# VOYAGE AU CENTRE DE LA HP28c/s

# VOYAGE AU CENTRE DE LA HP28c/s

Par
Paul COURBIS
&
Sébastien LALANDE

---

PARIS, Editions de la Règle à Calcul
65, Bd Saint-Germain. BP 300. 75228 PARIS Cedex.

**Paul COURBIS**
21 ans, élève-ingénieur à l'Ecole Nationale Supérieure des Mines de Saint-Etienne, ancien élève de mathématiques spéciales au lycée Pasteur (Neuilly-sur-Seine).

**Sébastien LALANDE**
19 ans, élève de mathématiques spéciales au lycée Pasteur (Neuilly-sur-Seine).

Nous remercions PHOTO CHIC PHOTO CHOC, 1 Place de l'Eglise, 78110 Le Vésinet. pour le soin apporté à la réalisation des photographies de ce livre.

# Sommaire

## Seconde partie: LE HARD



## ANNEXES

# ANNEXES (suite)

## B) Programmes mathématiques :

**Pour toute HP28 :**



**Pour HP28c seulement :**



## C) Programmes divers :

# INTRODUCTION

Cet ouvrage est destiné à tous les possesseurs d'HP28 quel que soit leur niveau et la version de leur machine.

En effet vous trouverez ici, aussi bien des informations faciles à comprendre que d'autres qui demanderont une certaine réflexion personnelle:

- Des programmes simples, courts, écrits en langage normal dans les domaines de la musique, du calcul, des tests, des utilitaires...

- De l'initiation au langage machine accessible à n'importe qui.

- Des transformations électroniques qui feront de votre calculatrice un véritable ordinateur super accéléré, possédant 66k et plus, pouvant communiquer avec presque n'importe quelle autre machine, et sauvegarder ses programmes sur disquettes...

- Des programmes plus complexes en langage machine et en langage habituel dans le domaine des effets spéciaux, des graphiques, des utilitaires, des calculs mathématiques avancés, des points d'accès de la Rom, ainsi qu'une grande quantité d'adresses de références.

Tout ceci est bien entendu adapté au trois versions de 28 déjà sorties:

| HP28C-1BB | HP28C-1CC | HP28S-2BB |
|---|---|---|

Toutefois un tel recueil ne gagnerait pas à être lu de la première à la dernière ligne. Si vous cherchez une information spécifique, il vous suffit de consulter la table des matières (si un chapitre entier lui est consacré ou si le titre du chapitre vous semble contenir ce que vous cherchez); sinon consultez tout simplement l'index.

Bien que nous ayons tenté de rassembler ici le maximum d'informations, la liste n'est pas exhaustive et il reste encore beaucoup à découvrir. C'est pourquoi si vous créez un programme ou découvrez une adresse, un effet intéressant, envoyez-nous vos suggestions, vous serez alors cité dans notre tome complémentaire.

Pour faciliter la lecture (et la rédaction de l'ouvrage), nous avons utilisé un certain nombre d'abréviations et de dénominations dont vous trouverez la liste dans le glossaire.

Il ne nous reste plus qu'à espérer que vous trouverez ici tout ce dont vous avez besoin pour que vous puissiez laisser courir librement votre imagination dans le monde passionnant des 28.

## COMMENT DETERMINER LA VERSION DE VOTRE MACHINE:

- Allumez votre HP28.
- Placez-vous dans le menu BINARY ([SHIFT] [B]).
- Appuyez sur la touche du menu [HEX] pour vous placer en mode hexadécimal (base 16 (on compte ainsi: 0 1 2 3 4 5 6 7 8 9 A B C D E F 10 11 12 ... 19 1A 1B 1C 1D 1E 1F 20...)).
- Tapez **#A** [ENTER].
- Puis tapez **SYSEVAL** [ENTER].
- La version de votre machine s'affiche en inverse sur la ligne supérieure de votre écran. Il y a trois versions possibles 1BB (28C 1ère version) 1CC (28C 2ème version) 2BB (28S 1ère version).

Pour chaque thème, nous préciserons à quelle machine il s'applique par les abréviations suivantes:

***ttv : toute version.***

***!c : seulement les 28C.***

***!s : seulement les 28S.***

***1BB : seulement les 1BB.***

***1CC : seulement les 1CC.***

# AVERTISSEMENT

Pour apprendre à entrer les codes des programmes en langage machine, lisez l'avertissement page 189.

Tout au long de cette partie nous proposerons des transformations physiques de votre HP28 qui auront pour objet d'améliorer ses capacités.

Les indications sont les plus claires possible, de manière à ce qu'un bricoleur moyen, puisse les réussir sans trop de risque.

MAIS :

**!!!!! ATTENTION !!!!!**

**VOUS ETES L'UNIQUE RESPONSABLE DE TOUTE ACTION PHYSIQUE OU SOFTWARE APPORTEE PAR VOUS OU UN TIERS A VOTRE MACHINE.**

Nous précisons, pour toutes les opérations théoriquement suggérées ici ou ailleurs, que, ni les auteurs, ni l'éditeur, ni l'imprimeur, ni le distributeur, ni la société Hewlett Packard ne peuvent être tenus responsables pour quelque endommagement que vous puissiez faire à votre machine.

De plus, il est rappelé que le simple fait d' **OUVRIR LA MACHINE ANNULE LA GARANTIE** et l'action du service après-vente de la société Hewlett Packard. Ces opérations proposées sont théoriquement exactes mais ne doivent pas être réalisées en pratique. Légalement, vous avez le droit de faire ce qu'il vous plaît avec votre machine une fois que vous l'avez achetée mais ces opérations sont à faire à vos propres risques, sous votre unique responsabilité. HP n'assure pas la réparation de machines modifiées par des particuliers, et décourage avec raison quiconque d'essayer de transformer une calculatrice, car toute machine mise sur le marché par HP est techniquement parfaite et ne nécessite aucune amélioration.

**!!!!! ATTENTION !!!!!**

**VOUS ETES L'UNIQUE RESPONSABLE DE TOUTE ACTION PHYSIQUE OU SOFTWARE APPORTEE PAR VOUS OU UN TIERS A VOTRE MACHINE.**

Toutefois nous signalons quand même à titre indicatif que chacune des propositions faites ici ont été réalisées avec 100% de succès sur de nombreuses machines.

---

# PREMIERE PARTIE

# LE SOFT

# Avertissement

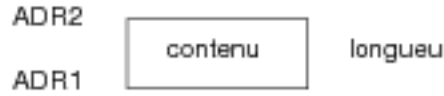
Le but de cette première partie est d'expliquer, ou plutôt de tenter d'expliquer, la manière de programmer en assembleur la HP28 (c'est-à-dire en langage machine). Pour cela, une bonne connaissance de la structure interne de la machine est nécessaire et les premiers chapitres sont consacrés à cela.

A la première lecture, toutes ces informations peuvent paraître difficiles à comprendre et à retenir mais il ne faut pas oublier que seuls les essais et l'utilisation des exemples proposés permettent de bien mémoriser (et de bien utiliser) ces informations... Il ne faut donc pas être découragé par les nombreux termes apparaissant dans le texte: les exemples sont là pour les éclairer et ils sont repris dans le glossaire (en annexe).

Courage, persévérance... Ce sont les seuls moyens de réussir, ici comme ailleurs...

***A propos des notations :***

Dans les chapitres III et IV sont présentés plusieurs tableaux représentant le contenu de la mémoire. Ils sont constitués de cases du type:




*On a toujours :*

ADR1 < ADR2

ADR1 = adresse de début de la zone (en hexadécimal).

ADR2 = adresse de début de la zone suivante (en hexadécimal).

"Longueur" = ADR2-ADR1 = nombre de quartets de la zone (en décimal).

"Contenu" = rappel de la nature du contenu, (détaillé plus loin).

***Remarque :***

L'utilisation des exemples proposés ne peut se faire qu'après la lecture des premiers chapitres: il est donc conseillé de lire une première fois ces premiers chapitres, pour apprendre à utiliser les exemples, puis de reprendre ces exemples, en les utilisant, de manière à bien les mémoriser et à bien les comprendre, ce qui sera très utile pour aborder le reste de ce livre.

# Chapitre I
# INTRODUCTION

Tout d'abord, il faut savoir que le microprocesseur du HP28 est quasiment le même que celui du HP71 (microprocesseur SATURN), ce qui permet de connaître les codes des différentes instructions du processeur (et d'utiliser certains manuels de référence du HP71, comme les HDS [Hardware internal Design Specification, HP N° 00071-90071], pour avoir la liste et l'explication de ces instructions, [qui sont rappelées en annexe]).

Ce microprocesseur est un 4 bits (l'unité élémentaire de mémoire est donc le quartet (et non l'octet comme sur beaucoup d'ordinateurs), c'est à dire une case mémoire dont la valeur est comprise entre 0 et 15 (décimal) ou encore entre 0 et F (en hexadécimal)).

Le SATURN possède 9 registres de 64 bits (A, B, C, D, R0, R1, R2, R3, R4) , 1 registre 1 bit (CARRY), 2 registres 4 bits (P, HWS), 1 registre 12 bits (OUTPUT), 2 registres 16 bits (INPUT, STATUS) , 3 registres 20 bits (D0, D1, PC), et une pile de 8 étages de 20 bits (RSTK). Une adresse est un nombre de 20 bits.

Une particularité du microprocesseur est de 'retourner' les informations lors de leur lecture: si on trouve en mémoire la suite de quartets (en hexadécimal) 780F4 la lecture dans un des registres, donnera, comme contenu de ce registre: 4F087... Ainsi toutes les adresses inscrites en mémoire vive (Ram) sont "écrites à l'envers" (tous les détails qui précèdent sont repris et détaillés dans l'annexe concernant le microprocesseur).

Cependant le SATURN de la HP28 n'est pas exactement le même que celui du HP71: il faut savoir que plusieurs instructions nouvelles viennent compléter celles présentes sur le HP71:

- **PC=(A) de code 808C** . Cette instruction lit un groupe de 5 quartets situés à l'adresse contenue dans les 20 premiers bits du registre A et charge ces 5 quartets (c'est à dire une adresse) dans le registre PC (Program Counter) ce qui provoque un saut à cette adresse.

- **RSI (reset system interrupt) de code 80810** dont le rôle n'a pas été explicité par HP.

- **Sur la HP28s** on rencontre encore de nouvelles instructions dont le code est en 808... comme:

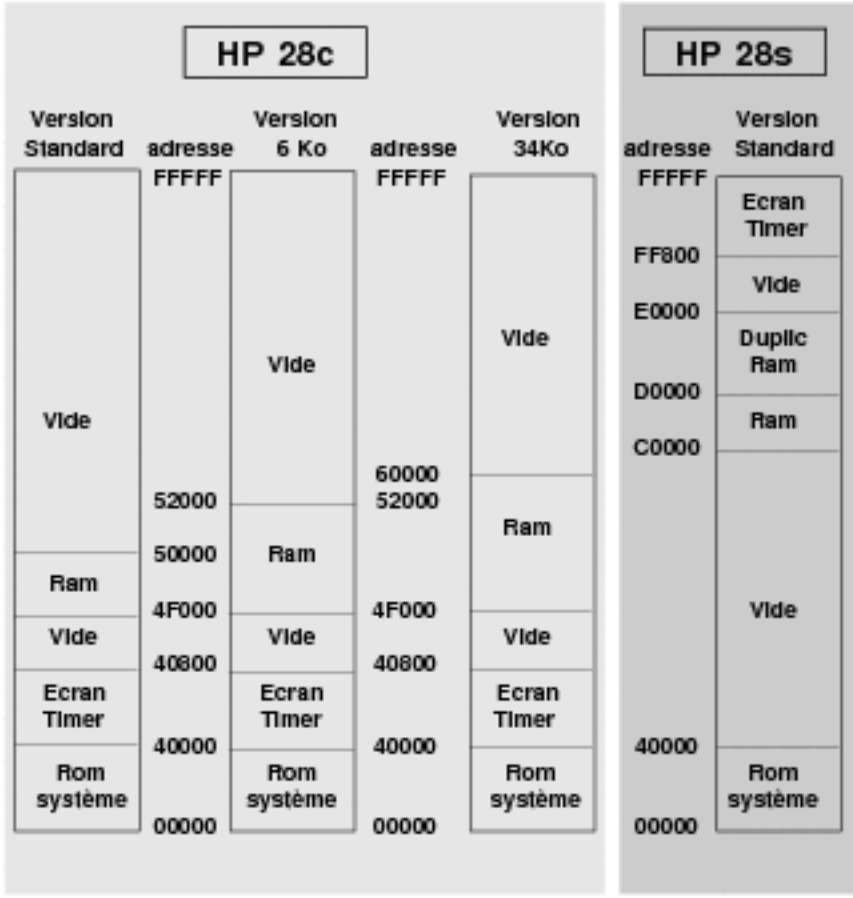
  **808A ?CBIT=0** **808B ?CBIT=1**
  **808D BUSCD**

  dont seuls les noms me sont connus... Ainsi que d'autres dont j'ai rencontré le code en désassemblant certaines parties de la Rom.

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 (c'est à dire un entier compris entre 0 et FFFFF [hexa] ), lit 5 quartets à cette adresse et les place dans le registre PC.

Il est à noter que, contrairement à une croyance répandue, l'instruction SYSEVAL ne peut, en aucun cas, détruire la machine. Le seul effet négatif possible peut être un Memory Lost, mais seulement si on utilise SYSEVAL au hasard. Tous les SYSEVAL présentés dans cet ouvrage ont été étudiés et sont absolument sans danger (pas de Memory Lost à craindre): le lecteur ne risque pas d'endommager sa machine par l'usage des séquences proposées qui utilisent cette mystérieuse instruction...

**Voici la structure de la mémoire pour les différentes versions de HP28:**



La mémoire (Rom, Ram, Ecran & timer) contient plusieurs types de groupes d'informations (suite 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 dernier type de groupe de quartets se rencontre aussi bien dans la Rom que dans la Ram; en particulier, les instructions RPL (Reverse Polish Lisp, c'est le langage du HP28) sont des objets au même titre que ceux créés par l'utilisateur.

---

# Chapitre II
# LES OBJETS

Pour bien comprendre le fonctionnement du HP28, 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 (c'est en fait une adresse en Rom) qui indique leur type, éventuellement suivi par des renseignements sur leur longueur, leur dimension...

Voici les 19 objets (leur caractéristiques détaillées suivent le tableau):

| 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 |
| Rom/Ram Pair (*) | 02AB8 |
| Algebraic | 02ADA |
| Program | 02C67 |
| Assembly Code (*) | 02C96 |
| Global Name | 02D12 |
| Local Name (*) | 02D37 |
| Rom Pointer (*) | 02D5C |

*(*) Ces objets sont impossibles à créer directement par l'utilisateur et apparaissent tous dans la pile sous la forme d'un "System Object" (sauf Local Name).*

## Caractéristiques des objets

Chaque objet est décrit en détail par une fiche du type suivant:

---

**Nom** [ nom en français ]

*Prologue :* xxxxxh (le h signifie que le chiffre est en hexadécimale mais n'est pas réellement stocké dans la mémoire)

*Structure :* tableau représentant la manière dont les quartets représentant l'objet sont stockés en mémoire.

*Exemple* de codage d'objets.

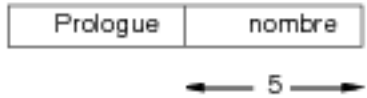
Cas d'utilisation d'un tel objet.

---

**Short integer** [ petit entier ]

*Prologue :* ***02911h***

*Structure :*




*Exemples :* 1192054321h est le short integer 12345h.
1192000000h est le short integer 00000h.

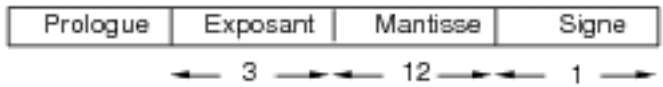
Ce type d'objets est utilisé par l'operating system de la HP pour passer des valeurs (par l'intermédiaire de la pile) entre deux routines. Par exemple une des sous-routines du BEEP produit un son à partir de ce genre d'objets pris dans la pile.

---

**Real** [ réel ]

*Prologue :* ***02933h***

*Structure :*




L'exposant et la mantisse sont en décimal codé binaire (DCB), c'est-à-dire qu'un nombre décimal est stocké comme si il était hexadécimal (ainsi 20d sera codé 20h [20h est un nombre hexadécimal qui est égal à 32 en décimal]). L'exposant varie de 000 à 499 (positif) et de 999 à 501 (négatif) [999 représentant l' exposant -1]. La mantisse contient 12 chiffres. Le signe est le signe de la mantisse: 0 si positif, 9 si négatif.

*Exemple :* π * 10 ^ 5 est représenté, en valeur numérique, par: 339205009535629514130h.

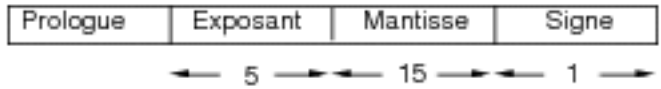
Utilisation : pour les calculs, par la machine comme par l'utilisateur.

---

**Extended Real** [ réel étendu ]

*Prologue :* ***02955h***

*Structure :*



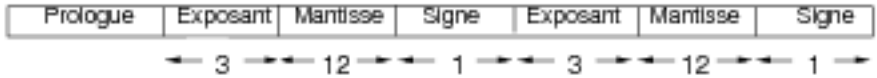

Mêmes remarques que pour Real et exemple similaire avec 3 décimales de plus et un exposant compris entre -49999 et +49999.

Utilisation : par la machine pour des calculs en haute précision (calcul de SIN, COS...).

**Complex** [ complexe ]

*Prologue :* ***02977h***

*Structure :*



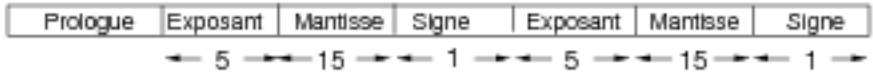

On trouve d'abord la partie réelle puis la partie imaginaire, sinon même structure que Real.
*Exemple :* le complexe (10,20) est représenté par
77920100000000000001010000000000020h.

Utilisation : par la machine, comme par l'utilisateur, pour les calculs, l'affichage de points...

**Extended Complex** [ complexe étendu ]

*Prologue :* ***0299Dh***

*Structure :*



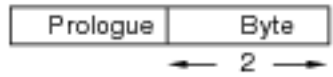

Mêmes remarques que pour Real, exemple similaire à Complex avec 2 * 3 décimales en plus et des exposants à 5 chiffres de -49999 à +49999.

Utilisation : en calculs internes (Comme Extended Real).

**Byte** [ octet ]

*Prologue :* ***029BFh***

*Structure :*




*Exemple :* octet 01: FB92010h

Utilisation : par l'operating system, comme le Short Integer.

**Premier objet inconnu**

*Prologue :* ***029E1h***

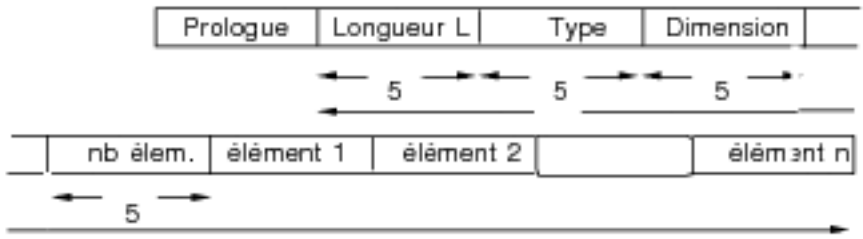
*Structure :* ???

Rôle : ???

**Array** [ vecteur/matrice ]

*Prologue :* ***02A0Ah***

*Structure :*




Le type est soit réel, soit complexe, et est représenté par le prologue Real ou Complex (si on crée un vecteur avec un autre prologue comme type, le vecteur est automatiquement considéré comme complexe). 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 identique. Cependant, "dimension" vaut 00002 et on a 2 * 5 quartets pour le nombre d'éléments (nombre de lignes / nombre de colonnes).

*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 |

Utilisation : par la machine comme par l'utilisateur.

---

**Second objet inconnu**

*Prologue* : ***02A2Ch***

*Structure* : ???

Rôle : ???

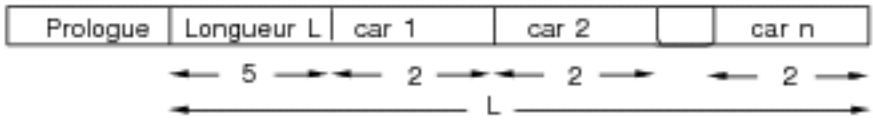
Cet objet, ainsi que le précédent inconnu, sont impossibles à charger par un STO sur la HP28c (ils provoquent un message Range Exception).

---

**String** [ chaîne ]

*Prologue* : ***02A4Eh***

*Structure* :




*Exemple* : "ABC" est représenté par E4A20B0000142434h

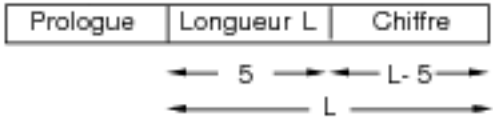
Utilisation : par la machine comme par l'utilisateur.

---

**Binary integer** [ entier ]

*Prologue :* ***02A70h***

*Structure :*




Habituellement L = 15h 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 07A20600000h.

*Exemple :* #123456789ABCDEF0h est représenté par:
07A20510000FEDCBA987654321h

Utilisation : par la machine, par l'utilisateur.

---

**List** [ liste ]

*Prologue :* ***02A96h***

*Structure :*


Fin de liste est le groupe de quartets 02F90.

*Exemple :* { #123456789ABCDEF0 "ABC" } est représenté par:

| | |
|---|---|
| 69A20 | prologue List |
| 07A20 | prologue Binary integer |
| 51000 | longueur de l'entier |
| 0FEDCBA987654321 | l'entier binaire |
| E4A20 | prologue String |
| B0000 | longueur de la chaine |
| 142434 | la chaîne |
| 09F20 | fin de liste |

Utilisation : par la machine, par l'utilisateur.

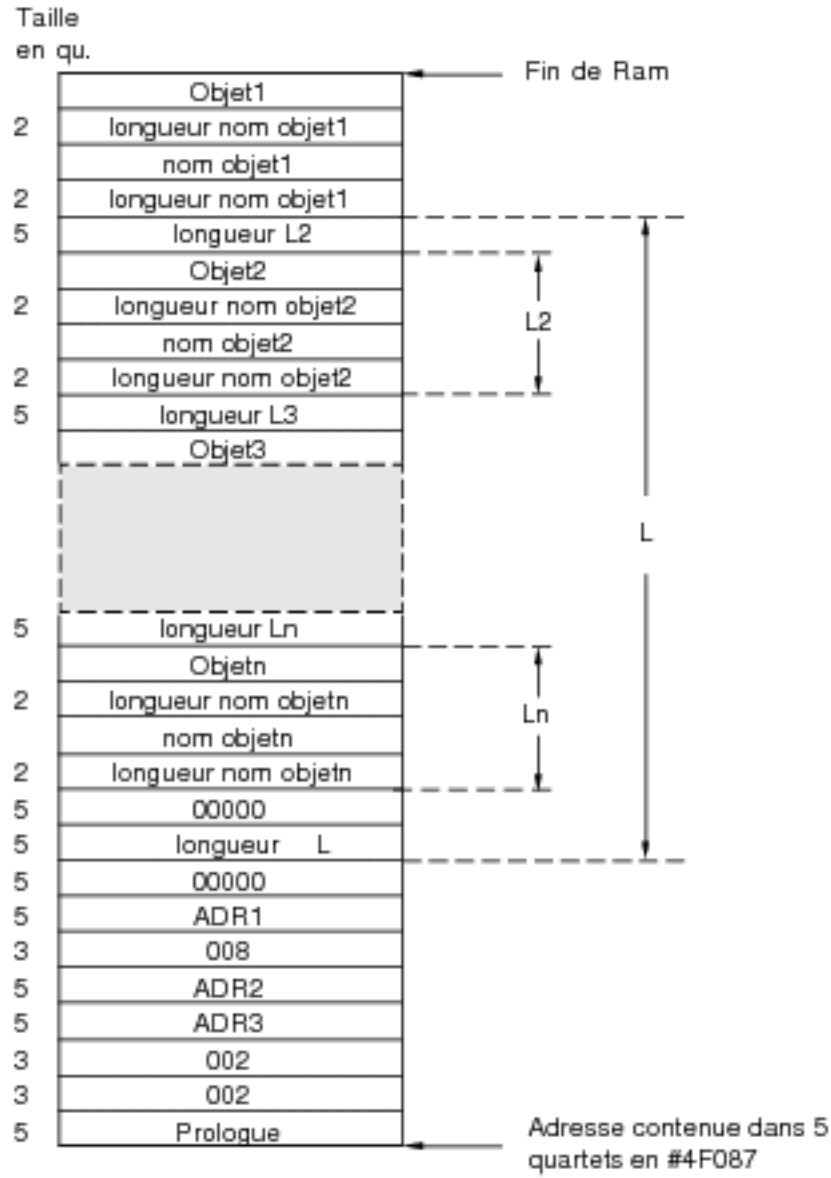
---

**Rom/Ram Pair** [ couple Rom/Ram ]

*Prologue :* ***02AB8h***

Cet objet constitue en fait la Ram USER ou aussi, pour la HP28s, un sous menu du menu 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 utilisateurs. En voici la structure :

**Pour les HP28c 1BB et 1CC:**

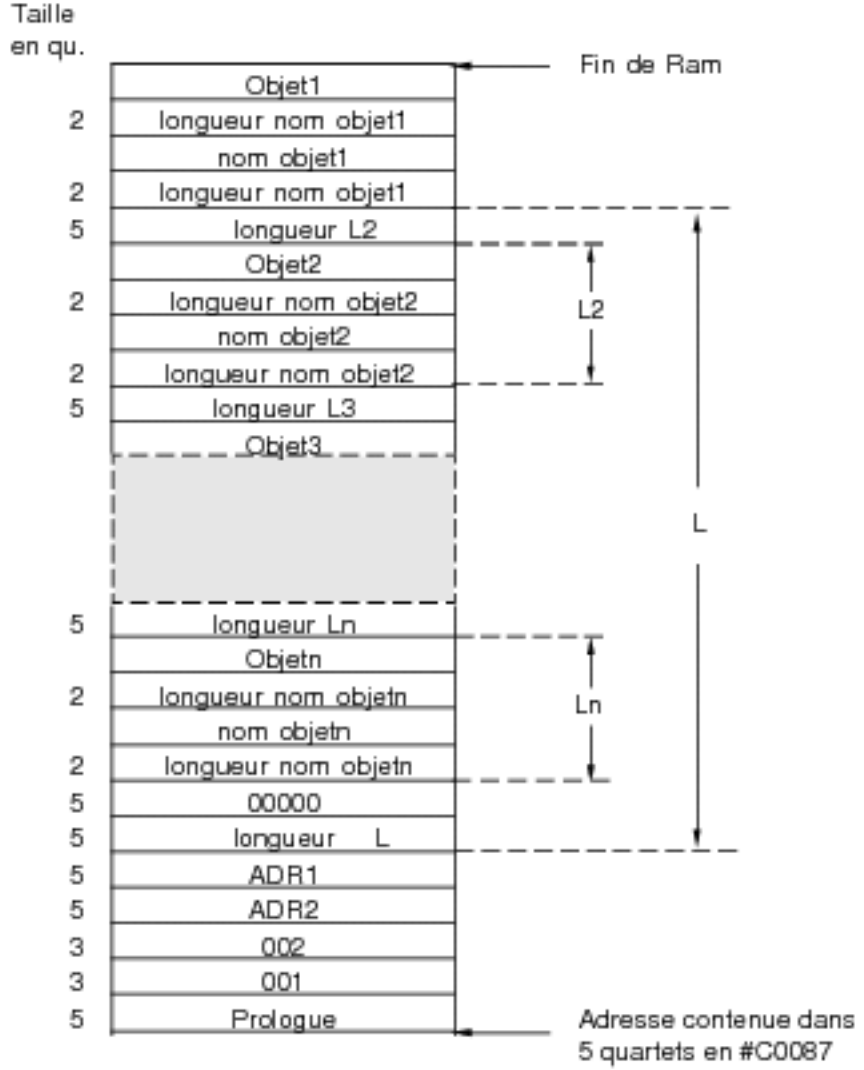



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| ADR1 | vaut | 27676 | pour la | 1BB, | 27758 | pour la | 1CC |
| ADR2 | vaut | 1C28C | pour la | 1BB, | 1C378 | pour la | 1CC |
| ADR3 | vaut | 1CB5D | pour la | 1BB, | 1CC49 | pour la | 1CC |

Ces trois groupes de 5 quartets sont probablement des adresses concernant les instructions RPL situées en Rom.

**Pour la HP28s :**

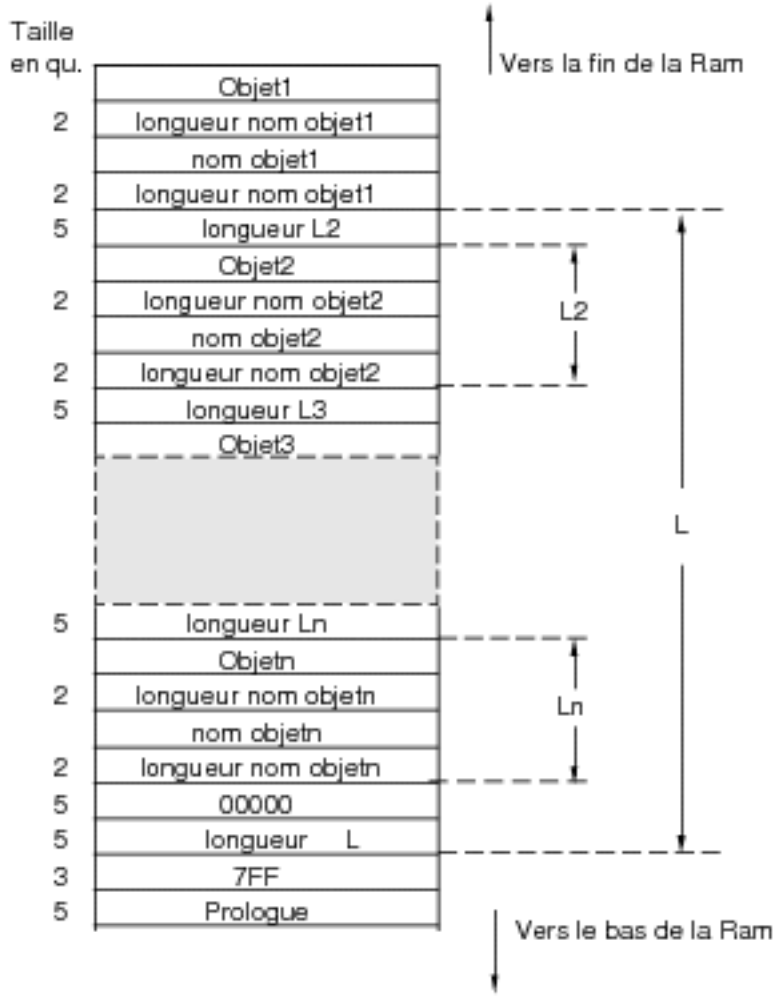
Tout d'abord dans le cas où il s'agit de la Rom/Ram pair constituant le menu HOME (menu principal) :




| | | |
|---|---|---|
| ADR1 | vaut | 0C291h |
| ADR2 | vaut | 0CBDCh |

Ces deux adresses concernent probablement les instructions RPL en Rom.

Pour une Rom/Ram pair constituant un sous-menu (c'est alors un objet contenu dans une autre Rom/Ram pair):




***Dans tous les cas:***

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

—> dans le sens "bas de Ram —>fin de Ram": on rencontre la taille du nom avant le nom ce qui permet de le "passer", puis le prologue avant l'objet ce qui permet de déterminer son type donc sa longueur.

—> dans l'autre sens: on rencontre les longueurs des zones avant celles-ci : il n'y a donc pas de problème.

*Exemple :* un tel exemple serait fastidieux et peu intéressant; le lecteur désirant avoir un exemple d'un tel objet n'aura qu'à utiliser le programme PEEK présenté en annexe en effectuant la séquence de commandes:

Mode hexadécimal puis :

*Pour 28c :*

**#4F087 #5 PEEK**

*Pour 28s :*

**#C0087 #5 PEEK**

Lire l'adresse en la retournant et la taper (ex "E18F4" -> #4F81E).
Entrer l'adresse de fin de Ram (voir plan mémoire du chap. I).

Faire: **OVER - PEEK**

Il y aura alors dans la pile une chaîne du type:
"8BA20200200...
C'est le listing des codes de l'objet Rom/Ram pair...

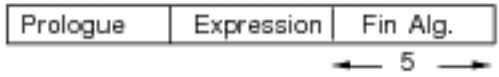
Utilisation : par la machine, pour le stockage des objets du menu USER.

---

**Algebraic** [ expression algébrique ]

*Prologue :* ***02ADAh***

*Structure :*




Fin d'algèbre vaut 02F90h. 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.

*Exemple :* 1.33 + 7.42 + SQR(74) [SQR remplaçant le signe racine carrée dans le texte] sera représenté par:

| | | |
|---|---|---|
| **Pour la HP28c 1BB:** | 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 | SQR |
| | D9F81 | + |
| | 09F20 | fin d'Algebraic |

| | | |
|---|---|---|
| **Pour la HP28c 1CC** | :ADA20 | prologue Algebraic |
| | 33920 | prologue Real |
| | 0000000000003310 | réel 1.33 |
| | 33920 | prologue Real |
| | 0000000000002470 | réel 7.42 |
| | 98091 | + |
| | 33920 | prologue Real |
| | 1000000000000470 | réel 74 |
| | 9C691 | SQR |
| | 98091 | + |
| | 09F20 | fin d'Algebraic |

| | | |
|---|---|---|
| **Pour la HP28s :** | ADA20 | prologue Algebraic |
| | 33920 | prologue Real |
| | 0000000000003310 | réel 1.33 |
| | 33920 | prologue Real |
| | 0000000000002470 | réel 7.42 |
| | EEE80 | + |
| | 33920 | prologue Real |
| | 1000000000000470 | réel 74 |
| | 42590 | SQR |
| | EEE80 | + |
| | 09F20 | fin d'Algebraic |

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

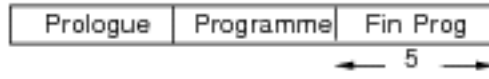
Utilisation : par la machine, par l'utilisateur.

---

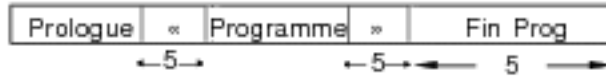
**Program** [ programme ]

*Prologue :* ***02C67h***

*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 un peu plus complexe car elle est de la forme:




« vaut: pour la 28c 1BB:27F0A, pour la 28c 1CC:27FEC, pour la 28s:0E9D0
» vaut: pour la 28c 1BB:27F1F, pour la 28c 1CC:28001, pour la 28s:0E9E6

C'est encore un objet composite.
*Exemple* : « "ABC" #123456789ABCDEF0 » est représenté par:

**Pour 28c 1BB:**

| | |
|---|---|
| 76C20 | prologue |
| A0F72 | « |
| E4A20 | prologue String |
| B0000 | Longueur chaîne |
| 142434 | ABC |
| 07A20 | prologue Binary Integer |
| 51000 | Longueur entier |
| 0FEDCBA987654321 | #123456789ABCDEF0 |
| F1F72 | » |
| 09F20 | Fin de programme |

---

**Pour 28c 1CC:**

| | |
|---|---|
| 76C20 | prologue |
| CEF72 | « |
| E4A20 | prologue String |
| B0000 | Longueur chaîne |
| 142434 | ABC |
| 07A20 | prologue Binary Integer |
| 51000 | Longueur entier |
| 0FEDCBA987654321 | #123456789ABCDEF0 |
| 10082 | » |
| 09F20 | Fin de programme |

---

**Pour 28s:**

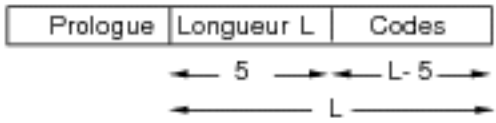
| | |
|---|---|
| 76C20 | prologue |
| 0D9E0 | « |
| E4A20 | prologue String |
| B0000 | Longueur chaîne |
| 142434 | ABC |
| 07A20 | prologue Binary Integer |
| 51000 | Longueur entier |
| 0FEDCBA987654321 | #123456789ABCDEF0 |
| 6E9E0 | » |
| 09F20 | Fin de programme |

Utilisation : par la machine, par l'utilisateur.

**Assembly Code** [ programme assembleur ]

*Prologue :* ***02C96h***

*Structure :*



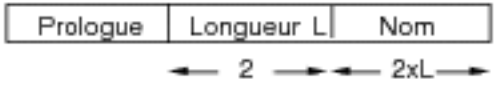

"Codes" représente la suite des codes des instructions constituant le programme assembleur.

Utilisation : par la machine, par l'utilisateur pour la programmation assembleur du HP28.

**Global Name** [ nom global ]

*Prologue :* ***02D12h***

*Structure :*




"Longueur" est le nombre de caractères du nom.

*Exemple :* 'ABC' est représenté par 21D2030142434.

Utilisation : par la machine, par l'utilisateur.

**Local Name** [ nom local ]

*Prologue :* ***02D37h***

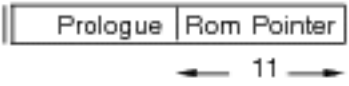
*Structure :* identique à un nom global, seul le prologue change.

Utilisation : par la machine, par l'utilisateur.

**Rom Pointer** [ pointeur de Rom ]

*Prologue:* ***02D5Ch***

*Structure :*




Rôle inconnu.

Utilisation : ???

A ces 19 objets, il convient d'ajouter:

- les instructions RPL,
- des instructions internes du HP28,
- les noms (globaux et locaux),
- diverses constantes;

tous codés sur 5 quartets (l'adresse de l'objet en question).

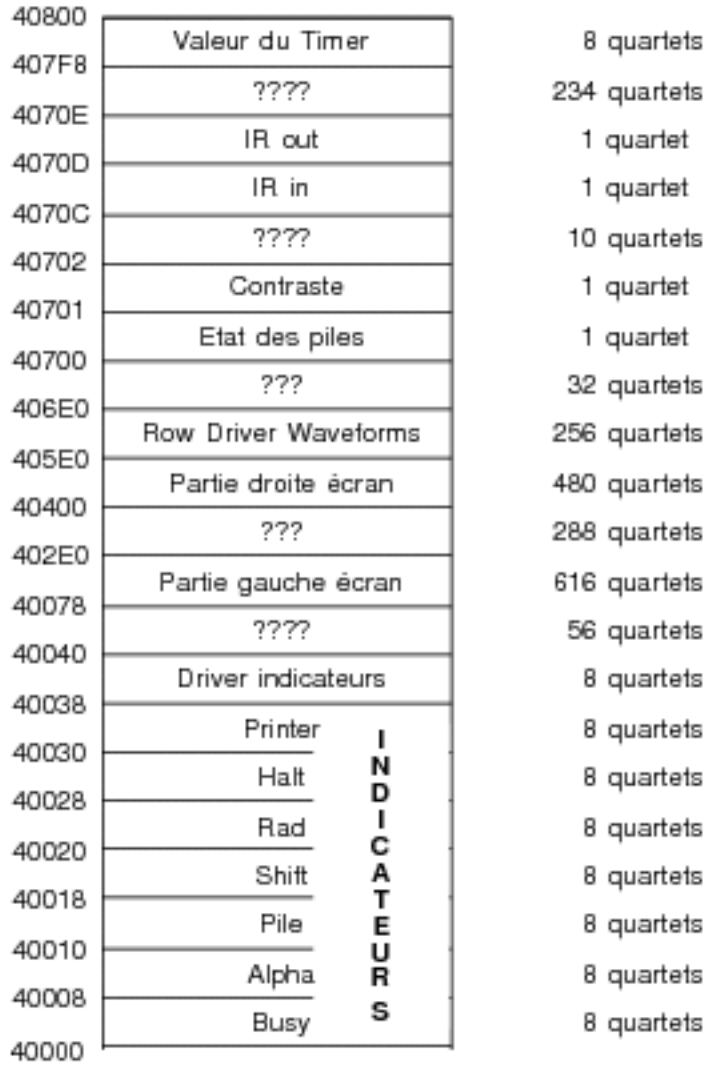
Ces objets sont tous considérés comme des instructions et sont en fait représentés par leur adresse d'exécution (utilisable par SYSEVAL).

Une liste (non exhaustive) de tels objets est donnée en annexe.

---

# Chapitre III
# ZONE D'ENTREE / SORTIE

**1) Pour les HP28 IBB et 1CC :**




**Indicateurs**

Si les 8 quartets valent FFFFF000, l'indicateur concerné est allumé, sinon il est éteint. En mettant plus ou moins de quartets à 'F' on peut jouer sur le contraste de l'indicateur concerné.

### Driver indicateurs

La valeur normale de ce driver est 00000000. Si on remplace certains '0' par des 'F' on joue sur le contraste de tous les indicateurs et, pour plus de trois 'F', on inverse l'affichage.

### L'écran

Chaque pixel est représenté par un bit, soit 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 composant électronique assurant la gestion de l'écran du HP28) le numéro de la ligne logique, située dans la Ram précédemment décrite, à afficher en une ligne physique donnée (sur l'afficheur proprement dit):

Appelons L1 ... L32 les 32 lignes physiques et l1 ... l32 les 32 lignes logiques. Chaque ligne physique est concernée par 32 bits (8 quartets) dans l'ordre L1, L32, L2, L31... Pour une ligne physique Ln donnée, on aura:

- si les 32 bits sont à 0, la ligne Ln sera éteinte;
- si plus d'un des 32 bits est à 1, la ligne Ln apparaîtra noire (plus ou moins foncée).
- si le m$^{ième}$ bit et lui seulement est à 1, la ligne lm sera affichée en Ln.

L'état normal est donc:

| | | | |
|---|---|---|---|
| 10000000000000000000000000000000 | L1 | 00000000100000000000000000000000 | L9 |
| 00000000000000000000000000000001 | L32 | 00000000000000000000000100000000 | L24 |
| 01000000000000000000000000000000 | L2 | 00000000010000000000000000000000 | L10 |
| 00000000000000000000000000000010 | L31 | 00000000000000000000001000000000 | L23 |
| 00100000000000000000000000000000 | L3 | 00000000001000000000000000000000 | L11 |
| 00000000000000000000000000000100 | L30 | 00000000000000000000010000000000 | L22 |
| 00010000000000000000000000000000 | L4 | 00000000000100000000000000000000 | L12 |
| 00000000000000000000000000001000 | L29 | 00000000000000000000100000000000 | L21 |
| 00001000000000000000000000000000 | L5 | 00000000000010000000000000000000 | L13 |
| 00000000000000000000000000010000 | L28 | 00000000000000000001000000000000 | L20 |
| 00000100000000000000000000000000 | L6 | 00000000000001000000000000000000 | L14 |
| 00000000000000000000000000100000 | L27 | 00000000000000000010000000000000 | L19 |
| 00000010000000000000000000000000 | L7 | 00000000000000100000000000000000 | L15 |
| 00000000000000000000000001000000 | L26 | 00000000000000001000000000000000 | L18 |
| 00000001000000000000000000000000 | L8 | 00000000000000010000000000000000 | L16 |
| 00000000000000000000000010000000 | L25 | 00000000000000001000000000000000 | L17 |

soit donc, en hexadécimal :

10000000 00000008 20000000 00000004 40000000 00000002 80000000 00000001
01000000 00000080 02000000 00000040 04000000 00000020 08000000 00000010
00100000 00000800 00200000 00000400 00400000 00000200 00800000 00000100
00010000 00008000 00020000 00004000 00040000 00002000 00080000 00001000

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

### Etat des piles

Le bit 0 de ce quartet indique si les piles sont bonnes ou non: si il est à 1 les piles sont déchargées. La mise à jour de ce bit se fait en écrivant dans le bit 3 du quartet (opération réalisée régulièrement par la machine).

### 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.

### IR in

C'est l'entrée du HP28. La valeur normale est non nulle et passe à zéro s'il y a réception ou si la diode IR est déconnectée (ce qui permet de tester la présence des périphériques proposés dans le présent livre).

### IR out

Ce quartet commande la diode infra-rouge du HP28. A zéro, il n'y a pas émission, pour 8, émission très faible (indétectable) pour 4, faible, pour 2, moyenne, pour 1, forte.

### 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 (sur 28c 1BB) ou par #1266 SYSEVAL (sur 1CC) est calculée en faisant la différence entre un offset de correction situé en Ram et la valeur du timer.

**2) Pour la HP28s:**

| Adresse | | Taille (décimale) |
|---|---|---|
| 00000 | Valeur du Timer | 8 quartets |
| FFFF8 | ???? | 234 quartets |
| FFF0E | IR out | 1 quartet |
| FFF0D | IR in (?) | 1 quartet |
| FFF0C | ???? | 3 quartets |
| FFF09 | Etat des piles | 1 quartet |
| FFF08 | ???? | 4 quartets |
| FFF04 | Etat écran | 1 quartet |
| FFF03 | ???? | 1 quartet |
| FFF02 | Contraste | 1 quartet |
| FFF01 | Speed | 1 quartet |
| FFF00 | 0000000.....0000000 | 160 quartets |
| FFE60 | ???? | 14 quartets |
| FFE52 | Interruptions | 42 quartets |
| FFE28 | Partie droite écran | 552 quartets |
| FFC00 | ???? | 9 quartets |
| FFBF7 | 3333333.....3333333 | 231 quartets |
| FFB10 | interruptions | 16 quartets |
| FFB00 | 0000000.....0000000 | 160 quartets |
| FFA60 | Partie gauche écran | 544 quartets |
| FF840 | Driver indicateurs | 8 quartets |
| FF838 | Printer (INDICATEURS) | 8 quartets |
| FF830 | Halt (INDICATEURS) | 8 quartets |
| FF828 | Rad (INDICATEURS) | 8 quartets |
| FF820 | Shift (INDICATEURS) | 8 quartets |
| FF818 | Pile (INDICATEURS) | 8 quartets |
| FF810 | Alpha (INDICATEURS) | 8 quartets |
| FF808 | Busy (INDICATEURS) | 8 quartets |
| FF800 | | |

Il est à noter que le Row Driver Waveform a disparu...

### Indicateurs

Si les 8 quartets valent FFFFF000, l'indicateur concerné est allumé, sinon il est éteint. En mettant plus ou moins de quartets à 'F', on peut jouer sur le contraste de l'indicateur concerné.

### Driver indicateurs

La valeur normale de ce driver est 00000000. Si on remplace certains '0' par des 'F' on joue sur le contraste de tous les indicateurs et, pour plus de trois 'F', on inverse l'affichage.

### 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).

### Interruptions

Ce sont des zones utilisées par la routine d'interruption du HP28 pour la sauvegarde temporaire des registres.

### Speed

La valeur normale de ce quartet est 7. En augmentant la valeur, on accélère la HP28, en la diminuant, on la ralentit. Il est à noter que 0 et 1 ne sont pas distingués. En mettant le quartet à F, on double la vitesse de la machine. Ce quartet est remis à 7 par tout appui sur ON. Sa modification n'entraîne pas de variation du cycle d'horloge.

### Le contraste

C'est un quartet dont la valeur va de #0, contraste le plus faible, à #F, contraste le plus fort.

### Etat écran

Si ce quartet vaut A, l'écran est allumé, s'il vaut 0, il est éteint. Ceci permet de faire tourner de longs programmes en gardant l'écran OFF ce qui économise les piles.

### Etat des piles

Le bit 0 de ce quartet indique l'état des piles: s'il vaut 0, les piles sont bonnes, s'il vaut 1, elles sont déchargées. Contrairement aux HP28c, il n'y a pas d'échantillonnage à réaliser: la valeur de ce bit est continuellement mise à jour par le circuit électronique.

### IR in

C'est l'entrée du HP28 (non utilisable sur 28s ???). La valeur normale est non nulle et passe à zéro si il y a réception ou si la diode IR est déconnectée (ce qui permet de tester la présence des périphériques proposés dans le présent livre).

### IR out

Ce quartet commande la diode infra-rouge du HP28. A zéro, il n'y a pas émission, pour 8, émission très faible (indétectable) pour 4, faible, pour 2, moyenne, pour 1, forte.

### Le timer

C'est un groupe de 8 quartets dont la valeur va décroissant de #077FFF à #000000. La valeur exacte obtenue par #11CA SYSEVAL est calculée en faisant la différence entre un offset de correction situé en Ram et la valeur du timer.

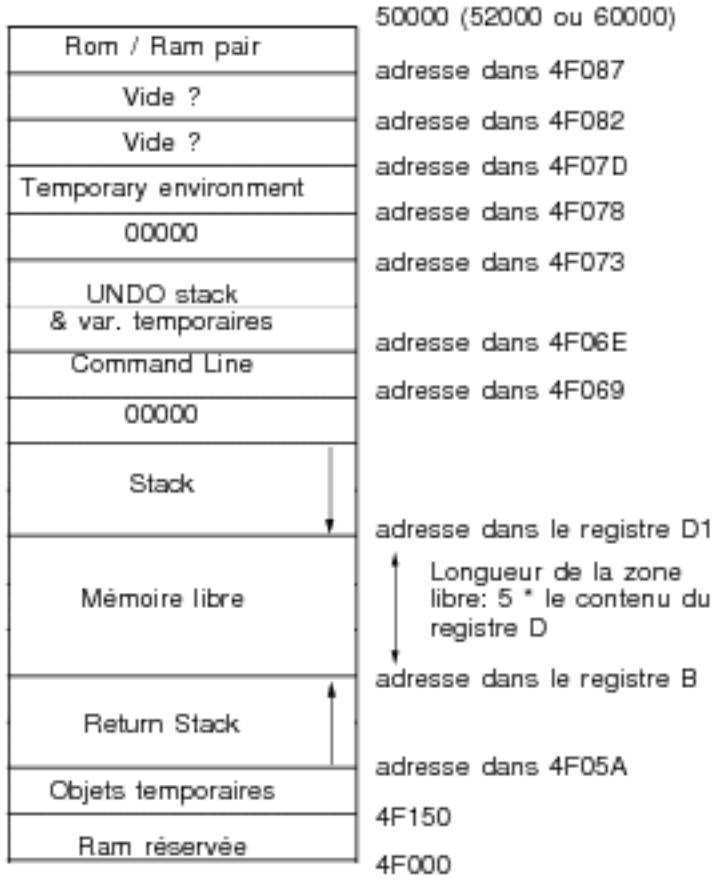
---

# Chapitre IV
# LA MEMOIRE

## 1) Pour la 28c:

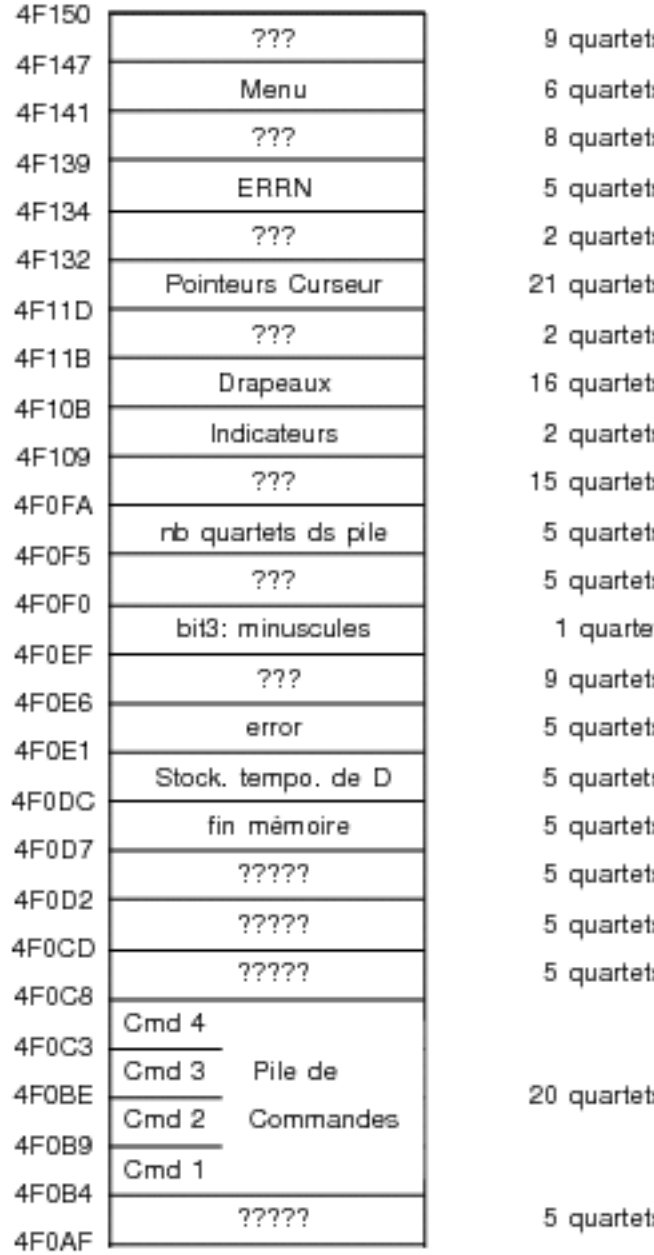
*A) Plan général de la Ram:*

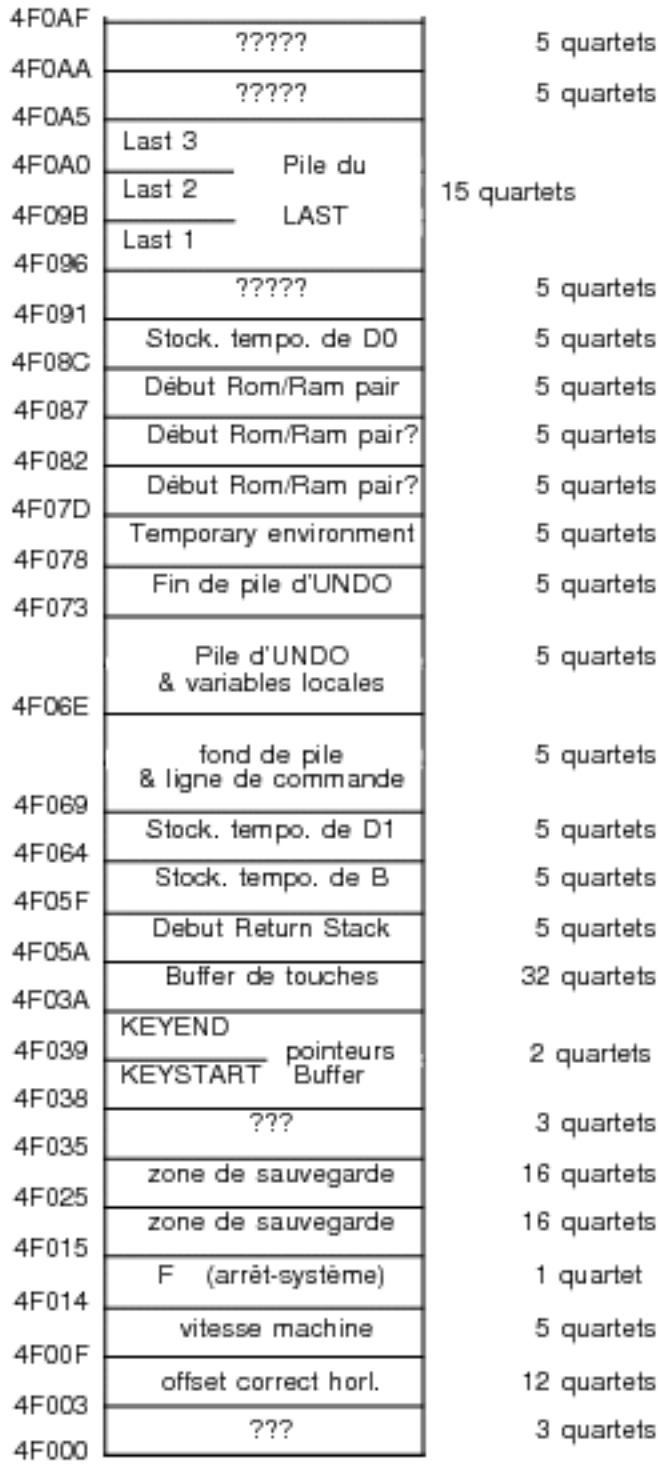
En version standard, elle va de #4F000h à #4FFFFh, en version 6 K-octets, de #4F000h à #51FFFh et en version 34 K-octets, de #4F000h à #5FFFFh.



*B) La Ram réservée:*

Elle contient la plupart des pointeurs nécessaires au fonctionnement de la HP28. A l'occasion de sa description, on verra les détails de la plupart des autres zones de la Ram.




4F0AF
?????
5 quartets
4F0AA
?????
5 quartets
4F0A5
Last 3
4F0A0
Pile du
Last 2
15 quartets
4F09B
LAST
Last 1
4F096
?????
5 quartets
4F091
Stock. tempo. de D0
5 quartets
4F08C
Début Rom/Ram pair
5 quartets
4F087
Début Rom/Ram pair?
5 quartets
4F082
Début Rom/Ram pair?
5 quartets
4F07D
Temporary environment
5 quartets
4F078
Fin de pile d'UNDO
5 quartets
4F073
Pile d'UNDO
& variables locales
5 quartets
4F06E
fond de pile
& ligne de commande
5 quartets
4F069
Stock. tempo. de D1
5 quartets
4F064
Stock. tempo. de B
5 quartets
4F05F
Debut Return Stack
5 quartets
4F05A
Buffer de touches
32 quartets
4F03A
KEYEND
4F039
pointeurs
2 quartets
KEYSTART
Buffer
4F038
???
3 quartets
4F035
zone de sauvegarde
16 quartets
4F025
zone de sauvegarde
16 quartets
4F015
F (arrêt-système)
1 quartet
4F014
vitesse machine
5 quartets
4F00F
offset correct horl.
12 quartets
4F003
???
3 quartets
4F000

### 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 (HP28c 1BB) ou de #1266 (1CC). On a temps réel = offset - timer, (timer est lu dans la Ram d'entrée/sortie).

### Vitesse machine

#4F00F à #4F013. Ces 5 quartets représentent la vitesse de la machine: ils sont utilisés par WAIT et BEEP de manière à ce que deux HP28c tournant à des vitesses différentes produisent les mêmes BEEP, effectuent les mêmes WAIT. En particulier, l'accélération proposée dans la première partie de cet ouvrage, n'influe pas sur BEEP et WAIT. (Sur la 28c en ma possession, la valeur de ce groupe de 5 quartets est 0961Dh).

### Arrêt-système et auto-tests

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

### Zones de sauvegarde

#4F015 à #4F024 et #4F025 à #4F034. Ces deux zones sont utilisées par la machine pour le stockage temporaire de certains registres.

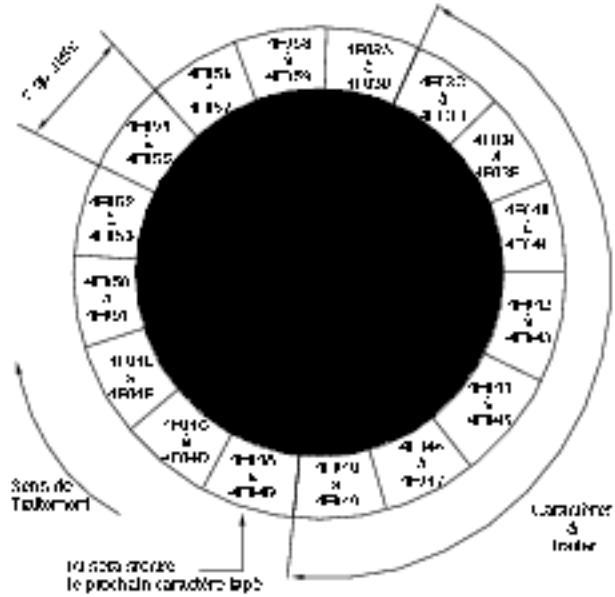
### Le buffer

#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 :

**Le Buffer**

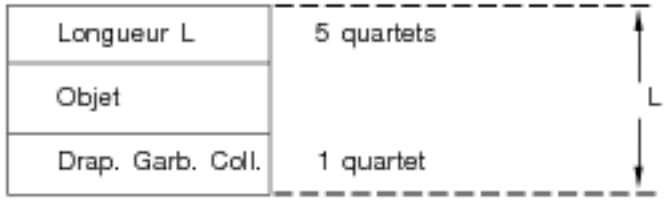


***Attention*** : les codes stockés dans le buffer sont les codes clavier et non les codes ASCII. En voici la liste :

| 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 | BACK | 33 |
| CHS | 34 | ENTER | 37 | SOLV | 3D |
| USER | 3E | NEXT | 3F | TRIG | 40 |
| Cursor | 43 | DOWN | 49 | LEFT | 4A |
| RIGHT | 4B | UP | 4C | DEL | 4F |
| INS | 51 | SHIFT | 80 | | |

### Zone des objets temporaires

Cette zone contient tous les objets qui ne sont pas stockés dans le menu USER. Elle contient les objets sous la forme suivante:



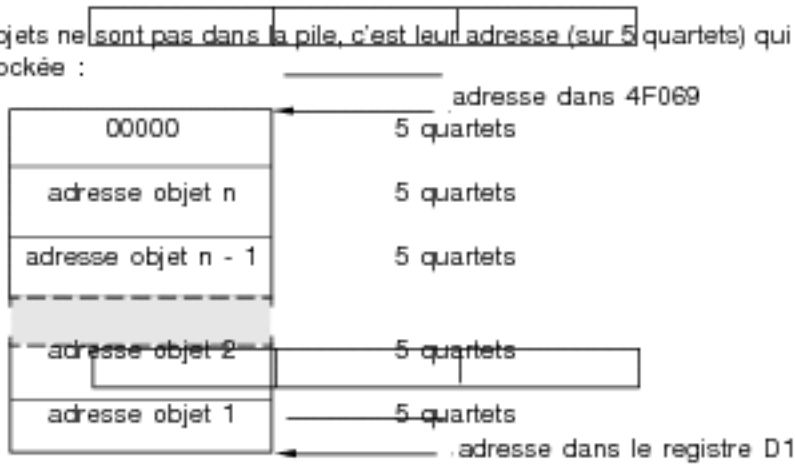

Drapeau du Garbage Collector sert lors du Garbage collector pour marquer les objets à garder (le Garb. Coll. se produit lorsqu'il n'y a plus assez de mémoire libre; cette opération consiste à détruire tous les objets qui ne sont plus utiles, c'est-à-dire ceux qui ne sont plus référencés).

### Return Stack

Cette pile d'adresses contient la suite des adresses de retour lors de l'appel de la routine de fin d'objet (09F20): chaque fois qu'il y a appel à un objet programme (prologue 76C20), il y a sauvegarde dans cette pile de l'adresse du programme appelant. A la fin du programme appelé (lors de la rencontre de 02F90), il y a "dépilage" de cette adresse pour pouvoir revenir au programme appelant...

### La pile

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

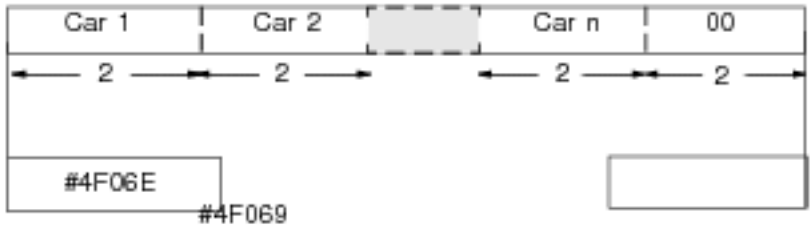



Ainsi pour obtenir l'adresse de l'objet situé au niveau 1 de la pile, il suffit de lire 5 quartets à partir de l'adresse contenue dans le registre D1.

Si, par exemple, le niveau 1 contient "ABC", à l'adresse ainsi lue, on trouvera E4A20B0000142434.

## 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 complétés à 23 par l'adjonction d'un ou de plusieurs caractères de code 00.

Des informations sur le curseur sont stockées de #4F11D à #4F131 :

| | | |
|---|---|---|
| 4F132 | | |
| 4F12F | pos. curseur ds ligne | 3 quartets |
| 4F12C | marge verticale | 3 quartets |
| 4F129 | ??? | 3 quartets |
| 4F126 | 1er étage à afficher | 3 quartets |
| 4F123 | ??? | 3 quartets |
| 4F120 | pos. curseur ds écran | 3 quartets |
| 4F11D | marge à gauche | 3 quartets |

marge à gauche : nombre de caractères cachés à gauche. Par exemple, si marge à gauche vaut 3, "123456" 1 DISP donnera ...56.

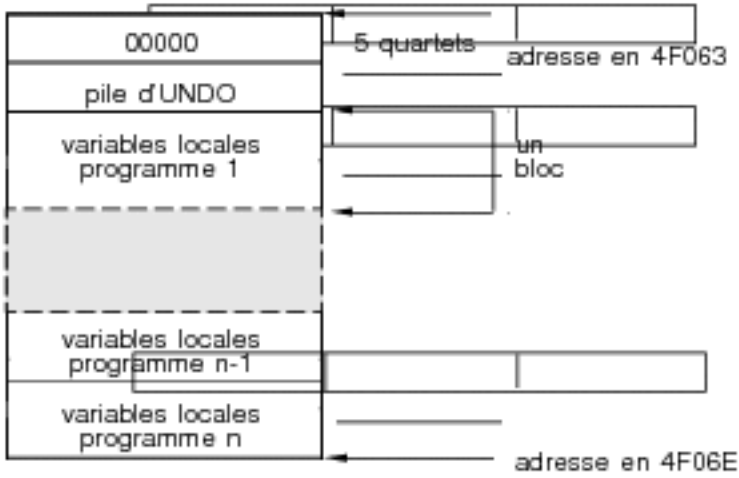
position curseur dans la ligne de commande : position du caractère sous le curseur dans la ligne de commande.

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

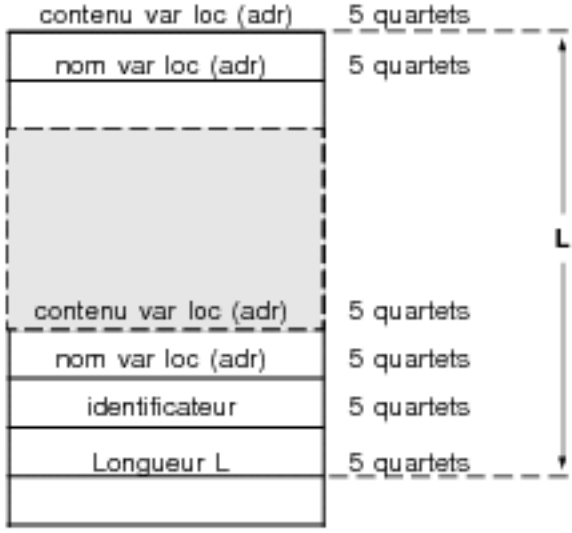
marge verticale : numéro de la ligne à partir de laquelle il faut afficher. Par exemple, si marge verticale vaut 3, alors: "1<br>2<br>3<br>4" 1 DISP<br>donnera : 3<br>4

position horizontale dans écran : de 0 à 16h.

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




Chaque bloc ayant la structure :




"nom variable locale" est l'adresse du nom de la variable locale dont l'adresse du contenu suit. Dans le cas de la pile d'UNDO, cette adresse est celle d'un nom vide ''.
"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 |
| ????? | 5 quartets |
| ????? | 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 |
| ????? | 5 quartets |
| ????? | 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 |
| ??? | 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 :

**Pour 1BB:**

0E059, 0E07C, 0E0B3, 0E0FE, 0E117, 0E130

**Pour 1CC:**

0E06F, 0E092, 0E0C9, 0E114, 0E12D, 0E146

et d'exécution :

**Pour 1BB** :

0DDBB, 0DDD9, 0DDF3, 0DE15, 0DEB3, 0DE51

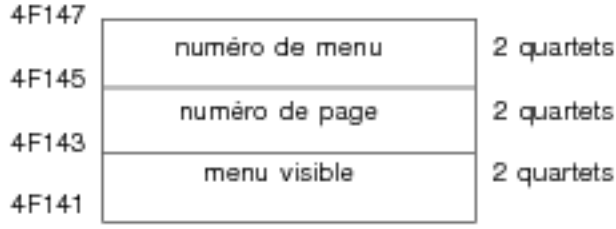
**Pour 1CC:**

0DDD1, 0DDEF, 0DE0D, 0DE2B, 0DE49, 0DE67

*Ces adresses ont pour rôle* :

- pour l'affichage de placer dans la pile le nom du n$^{ième}$ programme (chaîne de caractères).
- pour les secondes de déclencher l'exécution du n$^{ième}$ programme du menu.

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




"Menu visible" est égal à 00 si le menu est caché. Sinon, il est égal à 'numéro de menu'.
"Numéro de page" est le numéro de l'objet à partir duquel il faut commencer le menu (0,6,C,...).
"Numéro de menu" est 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 #4F087h à #4F08Bh. Elle a déja été étudiée précédemment (voir la rubrique sur les différents types d'objets).

### La pile du LAST

De #4F096h à #4F0A4h. 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

De #4F0B4h à #4F0C7h. 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 #4F109h à #4F10Ah.

**#4F109h**
-bit0: non utilisé ?
-bit1: imprimante
-bit2: halt
-bit3: radians

**#4F10Ah**

-bit0: shift
-bit1: pile
-bit2: alpha
-bit3: non utilisé ?

**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

Les drapeaux sont de #4F10Bh à #4F11Ah. Ce sont 64 bits, chaque bit représentant un drapeau (Flag 1 : Bit 0 de #4F10Bh, Flag 64 : Bit 3 de #4F11Ah).

### Adresses diverses

En #4F0D7h est inscrite l'adresse de fin de mémoire vive (#50000h en standard).

En #4F0E1h est écrit le numéro de la prochaine erreur à afficher. S'il vaut 00000, cela signifie qu'aucune erreur ne s'est produite.

Le bit 3 de #4F0EFh est à 1 si le mode "minuscules" est actif.

En #4F0F5h à #4F0F9h est inscrit le nombre de quartets dans la pile, c'est à dire 5 fois le nombre d'éléments dans la pile (5*DEPTH+5).

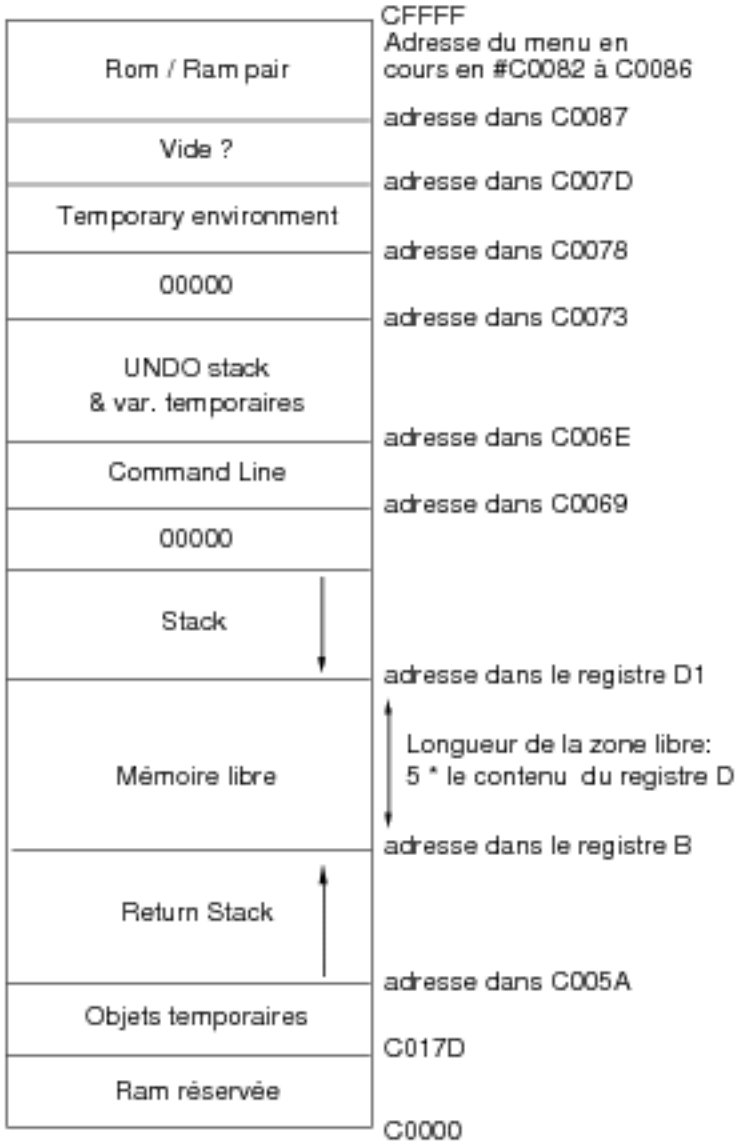
En #4F134h à #4F138h se trouve le numéro de la dernière erreur commise.

---

**2) Pour la 28s:**

*A) Plan général de la Ram :*

Elle va de #C0000 à #CFFFF et est dupliquée en #D0000 à #DFFFF (ceci étant probablement dû à la non connection d'un fil dans le module mémoire).



*B) La Ram réservée:*

Elle contient la plupart des pointeurs nécessaires au fonctionnement de la HP28. A l'occasion de sa description, on verra les détails de la plupart des autres zones de la Ram.

| | | |
|---|---|---|
| C017D | ??? | 26 quartets |
| C0163 | Command number | 5 quartets |
| C015F | ????? | 5 quartets |
| C015A | Menu | 9 quartets |
| C0151 | ?? | 8 quartets |
| C0149 | ERRN | 5 quartets |
| C0144 | Pointeurs du curseur | 35 quartets |
| C0121 | ??? | 2 quartets |
| C011F | Drapeaux | 16 quartets |
| C010F | Indicateurs | 2 quartets |
| C010D | ??? | 15 quartets |
| C00FE | nb quartets ds pile | 5 quartets |
| C00F9 | ?? | 6 quartets |
| C00F3 | bit1: mode 'menus' | 1 quartet |
| C00F2 | ? | 1 quartet |
| C00F1 | bit3: minuscules | 1 quartet |
| C00F0 | ??? | 10 quartets |
| C00E6 | error | 5 quartets |
| C00E1 | stock. tempo. D | 5 quartets |
| C00DC | fin mémoire | 5 quartets |
| C00D7 | ????? | 5 quartets |
| CC0D2 | ????? | 5 quartets |
| C00CD | ????? | 5 quartets |
| C00C8 | Cmd 4 | |
| C00C3 | Cmd 3 | |
| C00BE | Cmd 2 | Pile de Commande: 20 quartets |
| C00B9 | Cmd 1 | |
| C00B4 | ????? | 5 quartets |
| C00AF | ????? | 5 quartets |
| C00AA | ????? | 5 quartets |
| C00A5 | Last 3 | |
| C00A0 | Last 2 | Pile du Last: 15 quartets |
| C009B | Last 1 | |
| C0096 | ????? | 5 quartets |
| C0091 | Stock. tempo. de D0 | 5 quartets |
| C008C | Début Rom/Ram pair | 5 quartets |
| C0087 | Début menu en cours | 5 quartets |
| C0082 | | |

| | | |
|---|---|---|
| C0082 | Début Rom/Ram pair ? | 5 quartets |
| C007D | Temporary environment | 5 quartets |
| C0078 | Fin de pile d'UNDO | 5 quartets |
| C0073 | Pile d'UNDO | 5 quartets |
| | & variables locales | |
| C006E | fond de pile | 5 quartets |
| | & ligne de commande | |
| C0069 | | |
| | Stock. tempo. de D1 | 5 quartets |
| C0064 | Stock. tempo. de B | 5 quartets |
| C005F | Debut Return Stack | 5 quartets |
| C005A | Buffer de touches | 32 quartets |
| C003A | KEYEND | |
| C0039 | pointeurs Buffer | 2 quartets |
| | KEYSTART | |
| C0038 | ??? | 2 quartets |
| C0036 | inhibition clavier | 1 quartet |
| C0035 | zone de sauvegarde | 16 quartets |
| C0025 | zone de sauvegarde | 16 quartets |
| C0015 | 1 (arrêt-système) | 1 quartet |
| C0014 | vitesse machine | 5 quartets |
| C000F | offset correct horl. | 12 quartets |
| C0003 | ??? | 3 quartets |
| C0000 | | |

### Offset de correction de l'horloge

#C0003 à #C000E. Cette série de 12 quartets sert au calcul du temps effectif lors de l'appel de #11CA. On a temps réel = offset - timer, (timer est lu dans la Ram d'entrée/sortie).

### Vitesse machine

#C000F à #C0013. Ces 5 quartets représentent la vitesse de la machine: ils sont utilisés par WAIT et BEEP de manière à ce que deux HP28s tournant à des vitesses différentes produisent les mêmes BEEP, effectuent les mêmes WAIT. En particulier l'accélération proposée dans la seconde partie de cet ouvrage n'influe pas sur BEEP et WAIT. (Sur la 28s en ma possession la valeur de ce groupe de 5 quartets est 0F4E5h).

### Arrêt-système et auto-tests

#C0014. La valeur de ce quartet est normalement 1. En le mettant à 0, l'arrêt système [ON][UP] ainsi que les deux auto-tests n'ont plus aucun effet. Il est à noter que l'extinction de la machine réarme cet indicateur et permet à nouveau l'arrêt-système.

### Zones de sauvegarde

#C0015 à #C0024 et #C0025 à #C0034. Ces deux zones sont utilisées par la machine pour le stockage temporaire de certains registres.

### Inhibition clavier

Si #C0035 a une valeur non nulle, le clavier est inhibé (sauf ON). Ce quartet est remis à zéro à la fin de tout programme.

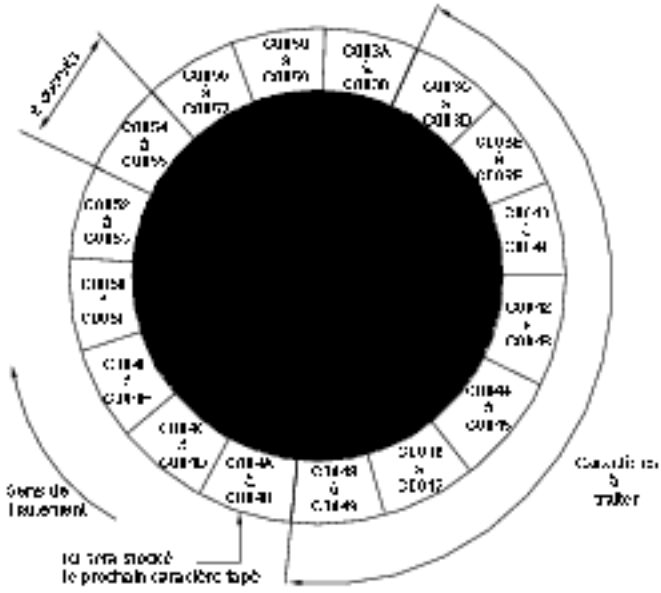
### Le buffer

#C0038 contient KEYSTART : pointeur de début des touches à traiter.

#C0039 contient KEYEND : pointeur de fin des touches à traiter.

#C003A à #C0059 : contient les codes clavier des touches. Les touches restant à traiter vont de #C003A + 2 * KEYSTART à #C003A + 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 :

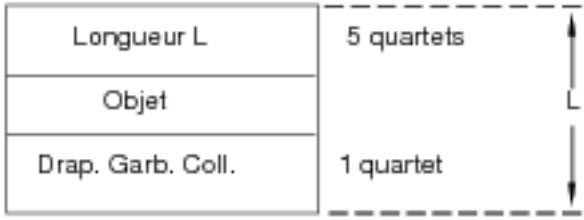


Attention : les codes stockés dans le buffer sont les codes clavier et non les codes ASCII. En voici la liste :

| Touche | Code | Touche | Code | Touche | Code |
|---|---|---|---|---|---|
| # | 13 | ' | 26 | ( | 17 |
| * | 1C | + | 04 | , | 03 |
| - | 10 | . | 0A | / | 28 |
| 0 | 08 | 1 | 14 | 2 | 16 |
| 3 | 0F | 4 | 20 | 5 | 22 |
| 6 | 1B | 7 | 2C | 8 | 2E |
| 9 | 27 | = | 06 | A | 45 |
| B | 48 | C | 43 | D | 42 |
| E | 3D | F | 47 | G | 39 |
| H | 3C | I | 37 | J | 36 |
| K | 31 | L | 3B | M | 2D |
| N | 30 | O | 2B | P | 2A |
| Q | 25 | R | 2F | S | 21 |
| T | 24 | U | 1F | V | 1E |
| W | 16 | X | 23 | Y | 15 |
| Z | 18 | [ | 0D | { | 12 |
| « | 07 | Space | 0C | α | 0B |
| lc | 01 | EVAL | 0E | STO | 1A |
| EEX | 33 | DROP | 34 | BACK | 35 |
| CHS | 3A | ENTER | 48 | SOLV | 3F |
| USER | 40 | NEXT | 41 | TRIG | 46 |
| Cursor | 44 | DOWN | 4B | LEFT | 4C |
| RIGHT | 4D | UP | 52 | DEL | 50 |
| INS | 4A | SHIFT | 80 | | |

**Zone des objets temporaires**

Cette zone contient tous les objets qui ne sont pas stockés dans le menu USER. Elle contient les objets sous la forme suivante:



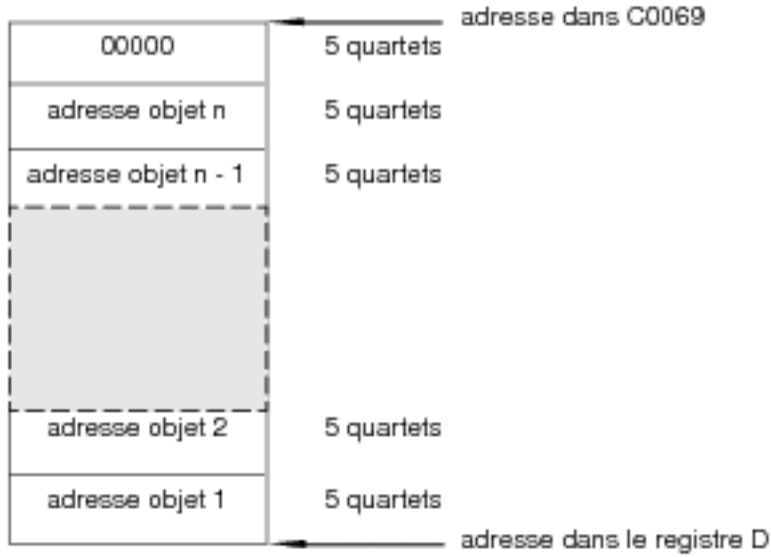

Drapeau du Garbage Collector sert lors du Garbage Collector pour marquer les objets à garder (le Garb. Coll. se produit lorsqu'il n'y a plus assez de mémoire libre; cette opération consiste à détruire tous les objets qui ne sont plus utiles, c'est-à-dire ceux qui ne sont plus référencés).

### Return Stack

Cette pile d'adresses contient la suite des adresses de retour lors de l'appel de la routine de fin d'objet (09F20): chaque fois qu'il y a appel à un objet programme (prologue 76C20), il y a sauvegarde dans cette pile de l'adresse du programme appelant. A la fin du programme appelé (lors de la rencontre de 02F90), il y a "dépilage" de cette adresse pour pouvoir revenir au programme appelant...

### La pile

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

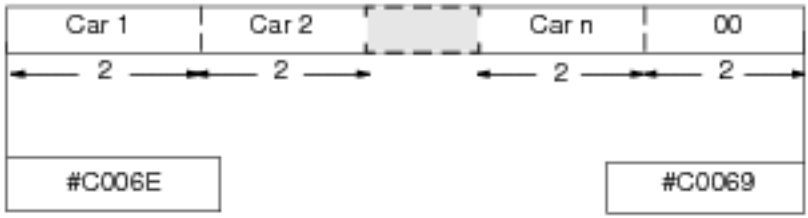



Ainsi pour obtenir l'adresse de l'objet situé au niveau 1 de la pile, il suffit de lire 5 quartets à partir de l'adresse contenue dans le registre D1.

Si, par exemple, le niveau 1 contient "ABC", à l'adresse ainsi lue, on trouvera E4A20B0000142434.

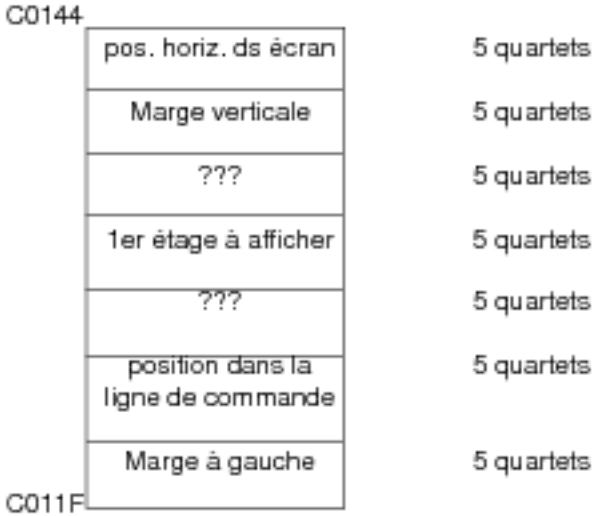
## La ligne de commandes

Elle se situe entre l'adresse pointée par #C0069 et celle pointée par #C006E (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 complétés à 23 par l'adjonction d'un ou de plusieurs caractères de code 00.

Des informations sur le curseur sont stockées de #C0121 à #C0143 :




marge à gauche : nombre de caractères cachés à gauche. Par exemple, si marge gauche vaut 3, "123456" 1 DISP donnera ...56.

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

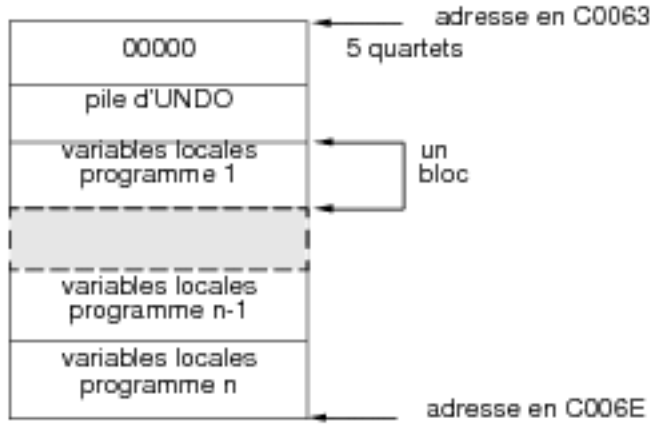
premier étage de la pile à afficher : premier étage visible de la pile.

marge verticale : numéro de la ligne à partir de laquelle il faut afficher. Par exemple, si marge verticale vaut 3, alors: "1
2
3
4" 1 DISP
donnera : 3
4

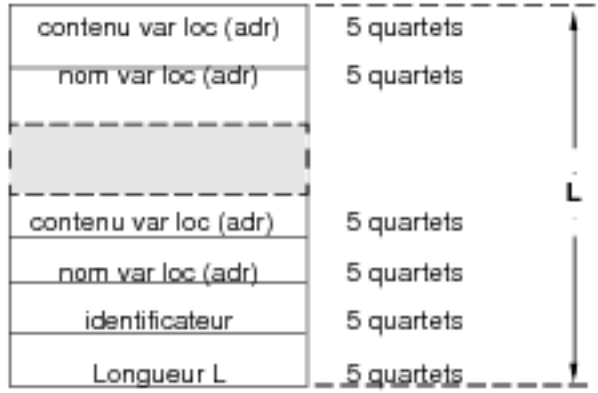
position horizontale dans écran : de 0 à 16h.

### La pile d'UNDO et des variables locales

Son adresse se trouve en #C006E sur 5 quartets.
La structure est la suivante :




Chaque bloc ayant la structure :



"nom variable" locale est l'adresse du nom de la variable locale dont l'adresse du contenu suit. Dans le cas de la pile d'UNDO, cette adresse est celle d'un nom vide ".

"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 #C0078 à #C007C. C'est une zone qui renseigne le HP-28 sur le menu à exécuter.

| | |
|---|---|
| 00000 | 5 quartets |
| ????? | 5 quartets |
| ????? | 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 |
| ????? | 5 quartets |
| ????? | 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 |
| ??? | 3 quartets |

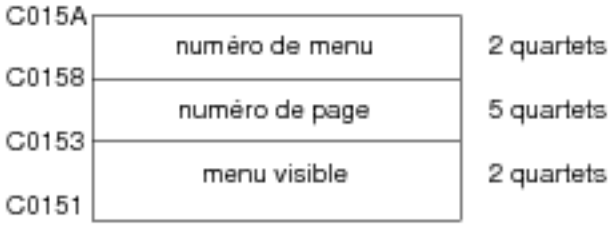
"Adresse disp n" pointe sur une routine déterminant le nom à placer dans la ligne de commande en mode ALPHA. Si le menu n'est pas le menu USER, le nom en question sera affiché dans le menu.

Pour le menu USER, les adresses disp sont dans l'ordre:
1DAEE, 1DB02, 1DB16, 1DB2A, 1DB3E, 1DB52
et d'exécution :
1D94F, 1D96D, 1D98B, 1D9A9, 1D9C7, 1D9E5

Ces adresses ont pour rôle :

- pour l'affichage de placer dans la pile le nom du n$^{ième}$ programme (chaîne de caractères).
- pour les secondes de déclencher l'exécution du n$^{ième}$ programme du menu.

Ces routines utilisent certainement les six quartets qui concernent aussi le menu et qui sont situés de #C0151 à #C0159.




Menu visible est égal à 00 si le menu est caché. Sinon, il est égal à 'numéro de menu'. Numéro de page est le numéro de l'objet à partir duquel il faut commencer le menu (0,6,C,...).
Numéro de menu est le numéro de menu à afficher. Voici la table relevée :

| | | | | |
|---|---|---|---|---|
| Array : 01 | Real : 06 | Stat : 0B | Mode : 10 | Solvr :18 |
| Binary : 02 | Stack : 07 | Print : 0C | Logs : 11 | Custom :EF |
| Cmplx : 03 | Store : 08 | Ctrl : 0D | Plot : 12 | |
| String : 04 | Memory : 09 | Branch : 0E | Solv : 16 | |
| List : 05 | Algebra : 0A | Test : 0F | User : 17 | |

A laquelle on peut ajouter :

19 : Menu FORM1 (Colct Expan Level Exget [<-] [->])
1A : Menu FORM2 (DNEG DINV *1 /1 ^1 +1-1)
1B : Menu FORM3 (-() <--> <-M M-> <-A A->)
1C : Menu FORM4 (AF)
1D : Menu FORM5 (1/() <--> <-D D-> <-A A->)
1E : Menu FORM6 (-() L() <-M M->)
1F : Menu FORM7 (1/() E() <-D D-> <-A A->)
20 : Menu FORM8 (1/() E^ D->)
21 : Menu FORM9 (-() L* D->)
22 : Menu FORM10 (->())
00 : Pas de menu

### Rom/Ram pair principale

Adresse en #C0087h à #C008Bh.
Elle a déjà été étudiée précédemment (voir la rubrique sur les différents types d'objets).

### Rom/Ram pair du menu en cours

Adresse en #C0082h à #C0086h.

### La pile du LAST

Adresses de #C0096h à #C00A4h.
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

Adresses de #C00B4h à #C00C7h.
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 #C010Dh à #C010Eh.

**#C010Dh**
- -bit0: non utilisé ?
- -bit1: imprimante
- -bit2: halt
- -bit3: radians

**#C010Eh**
- -bit0: shift
- -bit1: pile
- -bit2: alpha
- -bit3: mode algebraic/direct

*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

Les drapeaux sont de #C010Fh à #C011Eh. Ce sont 64 bits, chaque bit représentant un drapeau (Flag 1 : Bit 0 de #C010Fh, Flag 64 : Bit 3 de #C011Eh).

### Adresses diverses

En #C00D7h est inscrite l'adresse de fin de mémoire vive (#D0000h en standard).

En #C00E1h est écrit le numéro de la prochaine erreur à afficher. Si il vaut 00000 cela signifie qu'aucune erreur ne s'est produite.

Le bit 3 de #C00F0h est à 1 si le mode "minuscules" est actif.

Le bit 1 de #C00F2h est à 1 si le mode "menus" (Shift-Alpha) est actif.

En #C00F9h à #C00FDh est inscrit le nombre de quartets dans la pile, c'est à dire 5 fois le nombre d'éléments dans la pile (5*DEPTH+5).

En #C0144h à #C0148h se trouve le numéro de la dernière erreur commise.

---

# Chapitre V
# L'ACCES AU LANGAGE-MACHINE

Si vous ne savez pas encore ce qu'est le langage machine, reportez vous à l'annexe 1 qui l'explique...

Comme vous le savez maintenant, le langage machine est une suite de codes. Il faut donc réussir à placer cette suite de codes en mémoire. Pour ce faire, on ne créera les programmes-machine que sous la forme d'objets programmes-machine (prologue 02C96). Dans un premier temps, il va falloir effectuer une transcription entre la suite de codes '0' '1' ... 'E' 'F' qui constitue le codage de l' objet à créer et la suite de quartets correspondants. Pour des raisons de commodité, on stockera ces deux formes de codes dans des chaînes de caractères et on utilisera le programme 'ASS' présenté ci-dessous pour réaliser la transcription...

***Voici le listing de ASS :***

```
« -> LM « HEX "" 1 LM SIZE
        FOR X "#" LM X DUP2 1 + DUP SUB
            3 ROLLD DUP SUB + + STR-> B->R CHR +
          2 STEP » »
```

*Explication du programme :*

- On commence par ranger la liste des codes dans la variable locale LM ( -> LM);
- On se place en mode hexadécimal (HEX);
- On initie la chaîne qui contiendra les codes résultant de la transcription ("");
- On démarre une boucle pour le codage (1 LM SIZE FOR X);
- Comme un caractère de la chaîne LM représente un quartet et que l'on veut créer une chaîne comme résultat, on va coder 2 quartets par 2 quartets (1 caractère ASCII= 2 quartets=1 octet). Or on sait que le microprocesseur SATURN 'retourne' les codes (voir l'annexe sur le microprocesseur SATURN). Donc le doublet de quartet devra être 'retourné' avant d'être codé.

  C'est le rôle de la séquence LM X DUP2 1 + DUP SUB 3 ROLLD DUP SUB + (on prend le second caractère, puis le premier et on les additionne). Le codage est ensuite réalisé en ajoutant le caractère "#" en début des 2 quartets puis en effectuant un STR–>.

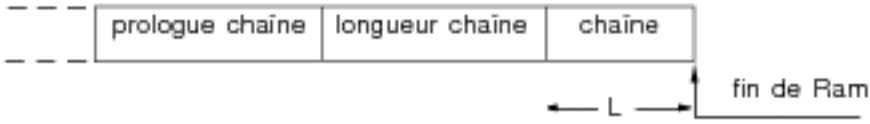
  On aboutit donc à la séquence: "#" LM X DUP2 1 + DUP SUB 3 ROLLD DUP SUB + + STR->.

- On passe ensuite au doublet suivant par 2 STEP.

Après avoir usé de ce programme sur la liste des codes (par exemple "76C2009F20" qui est l'objet programme vide), on obtient la chaîne contenant la liste des codes. Il faut à présent se 'débarrasser' des guillemets pour obtenir l'objet lui-même. La méthode la plus simple consiste à créer non pas l'objet lui-même, mais une liste contenant l'objet, puis à placer cette liste dans la pile en utilisant SYSEVAL et à effectuer la séquence 1 GET pour obtenir l'objet.

Cependant il faut connaître l'adresse du contenu de la chaîne pour faire ceci; mais le calcul de cette adresse est simple si on stocke celle-ci en début de menu USER (sur les 28s: en début de menu USER principal, c'est à dire en début de menu HOME).

En effet si on stocke l'objet de cette manière, la Ram se présente ainsi:




La valeur de L est 2 fois la longueur de la chaîne (chaque caractère occupe 2 quartets). Le programme ASS se transforme donc ainsi (ce nouveau programme sera appelé LASS comme List-ASS pour le distinguer du précédent)

**LASS**

```
« "69A20" SWAP + "09F20" + -> LM
   « HEX "" 1 LM SIZE
      FOR X "╬" LM X DUP2 1 + DUP SUB
            3 ROLLD DUP SUB + + STR-> CHR +
      2 STEP 'LM.C' DUP PURGE STO #fin.de.RAM
      LM.C SIZE 2 * - SYSEVAL 1 GET 1 ->LIST
      LIST-> DROP 'LM.C' PURGE » »
```

On ajoute "69A20" en début d'objet et "09F20" en fin pour placer l'objet à créer dans une liste. La chaîne résultat est placée dans LM.C en début de menu USER (pour ce faire, on commence par purger la variable). On calcule ensuite l' adresse du SYSEVAL par #fin.de.RAM LM.C SIZE 2 *. La liste est ensuite amenée dans la pile grâce au SYSEVAL. Enfin on extrait l'objet grâce à la séquence 1 GET, on oblige le calculateur à recréer l'objet grâce à 1 ->LIST LIST-> DROP et on détruit LM.C devenue inutile ('LM.C' PURGE).

*Remarques :*

#fin.de.RAM se détermine comme précisé au chapitre I, selon la version de la machine. (cet entier est soit #50000, soit #52000, soit #60000, soit #D0000).

**ATTENTION** : ces adresses sont valables en mode HEXADECIMAL.

Sur la 28s: rajouter l'instruction HOME en début de programme
(« HOME "69A20" -> LM ...).

Il est temps d'essayer quelques exemples. Vous pouvez utiliser les exemples du chapitre 3. En voici d'autres, tous très amusants !!!

**Empêcher le CLUSR :**

On va utiliser la particularité suivante: si une variable porte le même nom qu' une instruction RPL, alors la variable est prioritaire (lorsque son nom est entré alphabétiquement mais non si on utilise une touche de fonction). On va créer une variable de nom CLUSR. Cette création est normalement impossible mais devient facile avec LASS...

Le nom global 'CLUSR' se code ainsi (d'après le chapitre 3): 21D205034C4553525

*Voici ce qu'il faut faire :*

1) entrer l'objet à stocker. Par exemple:
« 1400 .07 BEEP "No CLUSR Avaible" 1 DISP »

2) créer le nom:
"21D205034C4553525" LASS [ENTER]
il apparaît alors 'CLUSR' dans la pile...

3) faire STO
Si on essaie de faire CLUSR, la machine "beepera" et affichera "No CLUSR Avaible", à la grande stupeur de vos amis!!!

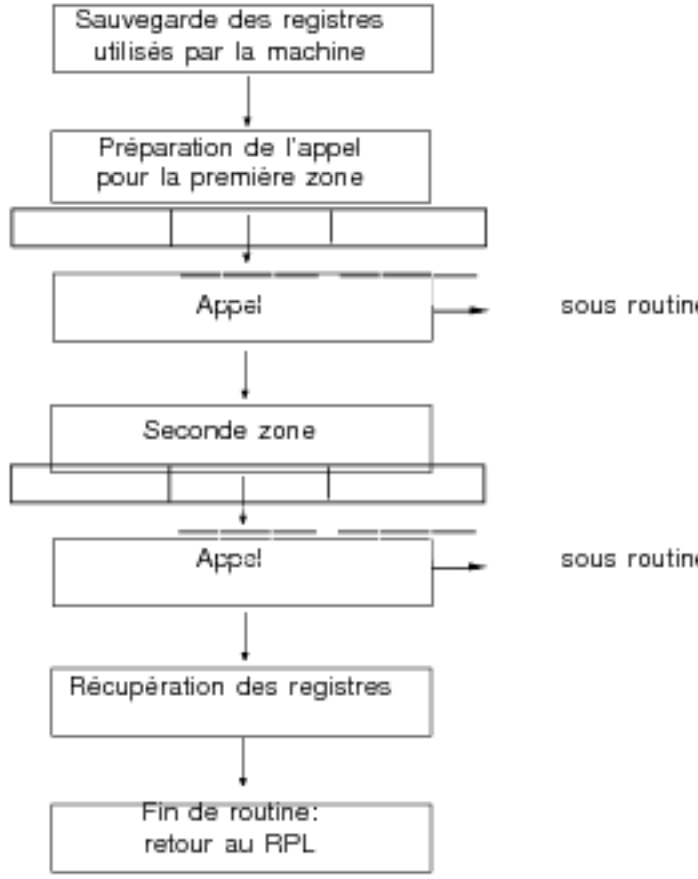
De la même manière, on peut créer des objets impossibles à créer directement...

**Inversion video :**

Le second exemple proposé ici est un programme en langage machine. Il suppose la connaissance des annexes 4 et 5 (au moins une bonne connaissance de l' annexe 4, l'annexe 5 n'étant à utiliser que comme référence). Ce programme effectue une inversion vidéo de l'écran de la HP28.

Comme l'écran est divisé en 2 parties, on va faire un programme appelant une sous-routine réalisant l'inversion d'une partie de l'écran précisée par le programme principal:

Voici l'organigramme du programme:




La sous routine effectue l'inversion entre deux adresses définies par le programme principal.

Si on ne fait que cela, l'écran va s'inverser puis repasser en 'normal' dès que le programme aura rendu la main au RPL. On va donc 'figer l'écran' grâce à une instruction RPL interne. Pour ajouter cette instruction au programme, on va placer ces deux objets dans une structure programme (prologue 02C67, fin 02F90).

Ce qui donnera un objet de la forme: Prologue programme
Figeage de l'écran
Objet programme LM
Fin de structure programme

Voici le listing résultat (les * représentent des quartets dépendants de la version de HP28 concernée)

```
Codes  Labels   Mnémoniques          Commentaires

76C20          CON(5)    #02C67      début de l'objet programme
*****          CON(5)    #*****      figeage écran (1BB:07C19
                                     1CC:07C4E 2BB:18A85)
69C20          CON(5)    #02C96      début de l'objet prgm
machine
E5000  début   CON(5)  (fin)-(début) longueur de cet objet
133            AD1EX
103            R3=A                  sauvegarde de D1
34*****        LC(5)     #*****      fin écran 1 (1BB,1CC:402DF
                                     2BB:FFA5F)
1F*****        D1=(5)    #*****      début écran 1 (1BB,1CC:40078
                                     2BB:FF840)
7220           GOSUB     sub         inversion
34*****        LC(5)     #*****      fin écran 2 (1BB,1CC:405DF
                                     2BB:FFE27)
1F*****        D1=(5)    #*****      début écran 2 (1BB,1CC:40400
                                     2BB:FFC00)
7010           GOSUB                 sub      inversion
113            A=R3
131            D1=A                  récupération de D1
142            A=DAT0  A             fin de routine
164            D0=D0+  5             (une routine LM se finit
                                     toujours par la séquence
809C           PC=(A)                A=DAT0  A   D0=D0+5  PC=(A))


27    sub      P=        7           on travaille sur une
colonne,                             c'est à dire sur 8 quartets:
                                     on fixe donc P à 7 pour que
                                     le champ WP contienne
                                     8 quartets (7 à 0)
1531  sub10   A=DAT1     WP          lecture d'une colonne
B9C            A=-A-1    WP          inversion(A=-A-1 correspond
                                     à A=NOT(A))
1511           DAT1=A    WP          écriture de la colonne
inversée
177            D1=D1+    8           colonne suivante
133            AD1EX                 copie de D1 dans A (on doit
                                     procéder en 2 étapes car
131            D1=A                  A=D1 n'existe pas)
8BA            ?A#C      A           test: doit on continuer ?
9E             GOYES     sub10       si oui -> on repart à sub10
20             P=        0           sinon: on met P à 0 (valeur
                                     standard)
01             RTN                   retour de sous routine
      fin
09F20          CON(5)    #02F90      fin de l'objet programme
```

Une fois codé, on obtient la séquence de quartets :

**Pour la 1BB:**

```
'76C20  91C'0  69C20  E5000  13310  334FD  2041F
8'004  '2203  4FD50  41F00  404'0  10113  13114
21648  08C2'  1531B  9C151  11'''1  33131  8BA9E
20010  9F20
```

**Pour la 1CC:**

```
'76C20  E4C'0  69C20  E5000  13310  334FD  2041F
8'004  '2203  4FD50  41F00  404'0  10113  13114
21648  08C2'  1531B  9C151  11'''1  33131  8BA9E
20010  9F20
```

**Pour la 2BB:**

```
'76C20  58A81  69C20  E5000  13310  334F5  AFF1F
048FF  '2203  4'2EF  F1F00  CFF'0  10113  13114
21648  08C2'  1531B  9C151  11'''1  33131  8BA9E
20010  9F20
```

**Pour entrer le programme: (lire l'avertissement Page 189).**

1) entrer la chaîne des codes ("76C20...)
2) exécuter LASS
3) stocker le résultat (deux 'System Object') dans 'INV.VID'

Le programme est alors prêt à fonctionner.
Pour le démarrer, il suffit d'appuyer sur la touche INV.VID du menu USER ou de taper INV.VID suivi de ENTER...

D'autres programmes LM sont présentés dans l'annexe 6 (Bibliothèque de programmes).

*Il ne faut jamais oublier :*
lorsqu'on réalise un programme LM il faut toujours sauvegarder les registres B, D, D1, D0 et R4 si on doit les modifier.
un programme LM se termine toujours par:

```
142   A=DAT0  A
164   D0=D0+5
808C  PC=(A)
```

Des routines utiles sont décrites en détails dans le chapitre 7.

---

# Chapitre VI

# COMMENT EN DECOUVRIR PLUS ?

Pour découvrir plus de choses, on peut uiliser deux voies:

1) On peut se servir de la liste des chaînes de caractères (située en annexe) et repérer dedans des chaînes pouvant servir à une routine intéressante:

   Chercher alors la routine utilisant cette chaîne (généralement elle est située juste après) et la désassembler. Dans le même ordre d'idées: on peut repérer une adresse Ram intéressante et chercher les routines qui y font appel, puis les désassembler (pour cela on peut utiliser le programme SEARCH, présenté en annexe, qui a pour but de chercher une suite de quartets donnée en Rom). On peut aussi désassembler une routine précise (instruction RPL) qui doit contenir des choses intéressantes.

2) Effectuer un désassemblage systématique de la Rom de la HP28. C'est une opération de longue haleine. On peut s'aider du programme de désassemblage, ou, méthode plus commode, transférer la Rom de la HP28 sur un plus gros ordinateur (en utilisant PEEK et l'interface d'entrées/sorties présentée dans la seconde partie. Puis exploiter la Rom sur ce gros système (ce qui permet en particulier de sauvegarder le désassemblage sur disquette, et de le transmettre à vos amis).

Personnellement, j'ai utilisé les deux méthodes (je n'ai cependant fait qu'entamer la seconde...) C'est ainsi que j'ai pu analyser la Ram, et découvrir les quelques adresses utiles présentées dans le chapitre suivant.

La meilleure façon de procéder est de bien assimiler ce qui a déjà été découvert (ce que contient ce livre), et de s'en servir comme base des découvertes ultérieures...

---

# Chapitre VII
# ROUTINES UTILES

Ces routines sont données par leur adresse d'appel (appel par GOSBVL ou GOVLNG). Attention: l'adresse d'appel dépend de la version de machine utilisée...

---

**Sauver les registres** utilisés par le système de la machine: **SAV.REG**

**Adresse : 1BB : 04EE2 1CC : 04EE2 2BB : 05081**

*Usage :* cette routine sauve les registres de travail de la HP28 (B, D, D0 et D1) dans la RAM réservée. Attention: le registre C est modifié. Cette routine doit nécessairement être utilisée si on veut réserver de la place dans la zone des objets temporaires (voir la routine RES.ROOM). Le contenu des registres B, D, D0 et D1 n'est pas modifié.

---

**Récupérer les registres** sauvés par SAV.REG: **LOAD.REG**

**Adresse : 1BB : 04F19 1CC : 04F19 2BB : 050B8**

*Usage :* cette routine récupère les valeurs sauvées par SAV.REG. Ces valeurs ont pu être modifiées par RES.ROOM et GARB.COLL. C est modifié par l'appel de cette routine.

---

**Réserver de la place dans la Ram des objets temporaires: RES.ROOM**

**Adresse : 1BB : 051B7 1CC : 051B7 2BB : 053AE**

*Usage :* C doit contenir le nombre de quartets à réserver. Après l'appel: si on a pu réserver la place, la carry vaut 0 et D0 contient l'adresse de début de la zone réservée. Sinon la carry vaut 1. Les registres A, B, C, D, D0, D1, P, carry sont modifiés. Les sauvegardes des registres (réalisées par SAV.REG) sont mises à jour.

---

**Effectuer un garbage collector**: **GARB.COLL**

**Adresse : 1BB : 0497D 1CC : 0497D 2BB : 04A94**

*Usage :* il suffit d'appeler cette adresse. Les sauvegardes réalisées par

---

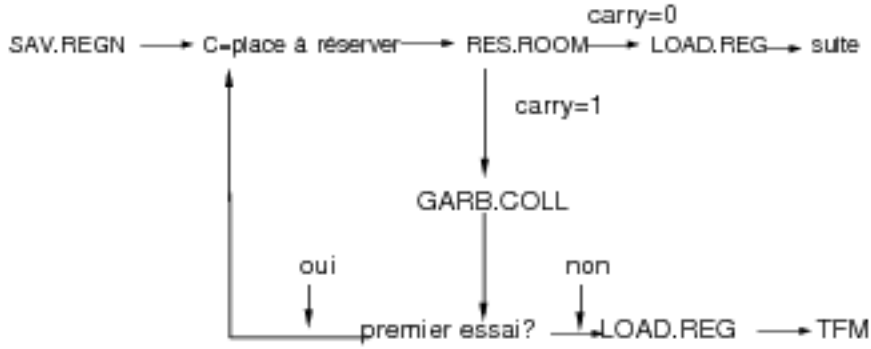
SAV.REG sont mises à jour.

**Afficher "Too Few Memory": TFM**

**Adresse : 1BB : 0332C 1CC : 0332C 2BB : 0393E**

*Usage :* à appeler par un GOVLNG. Affiche le message d'erreur et rend la main à l'interpréteur RPL.

Souvent on utilise conjointement ces routines:




**Afficher un message d'erreur quelconque: ERROR**

**Adresse : 1BB : 03358 1CC : 03358 2BB : 0396A**

*Usage :* charger le numéro de l'erreur dans le registre A. La routine affiche l'erreur correspondante et rend la main à l'interpréteur RPL.

**Emettre un BEEP: BEEP**

**Adresse : 1BB : 1016C 1CC : 101A3 2BB : 22AD3**

*Usage :* fréquence dans D champ A, durée dans C champ A (en millisecondes). Cette routine émet un BEEP de paramètres correspondants. Elle est en particulier utilisée dans le programme musical présenté en annexe.

Ces routines sont les plus utiles. Il existe d'autres routines, moins utiles que voici:

---

**Diviser le contenu de C champ A par 5: DIV5**

(attention: D est modifié).

**Adresse : 1BB : 0516D 1CC : 0516D 2BB : 05364**

*Usage :* n fait, c'est une multiplication par 3355444 / 16777216, qui vaut environ 0,2.

---

**Transférer une zone mémoire vers le haut: TRH**

**Adresse : 1BB : 04E00 1CC : 04E00 2BB : 04F9F**

*Usage :* D1=nouvelle adresse de fin D0=ancienne adresse de fin C=longueur à transférer. A n'utiliser que si les deux zones ne se recoupent pas ou si D1>D0.

---

# Conclusion
# QUE PEUT-ON FAIRE?

La réponse à cette question est simple: tout ou presque !!!

Voici quelques idées de programmes:

Au point de vue des graphiques: grâce à la connaissance de la Ram des entrées/sorties, on sait maintenant allumer et éteindre un point: on peut envisager la création de jeux utilisant les graphiques, de manière rapide. De plus, on peut créer des effets spéciaux en jouant sur le contraste...

Au point de vue sonore: l'adresse de la routine de BEEP est donnée dans le chapitre précédent, ce qui permet de réaliser des effets sonores surprenants... Un tel programme se trouve dans la bibliothèque.

Au point de vue calculs: ceux-ci sont plus rapides en LM mais moins faciles à mettre en oeuvre. Cependant, il pourra être utile de réaliser des routines LM pour ajouter certaines fonctions à la HP28.

Il y a certainement beaucoup d'autres choses à faire...

En guise de conclusion pour cette partie:

Cette première partie avait pour but de vous décrire les bases de la programmation-machine. Les exemples de la bibliothèque de programmes (en annexe) sont là pour vous aider à vous perfectionner. Il ne faut pas espérer être à même de réaliser du premier coup un programme très complexe : commencez donc par réaliser de petites routines pour vous entraîner... Voilà, à présent c'est à vous... Envoyez-moi vos réalisations: je serais très heureux de voir les idées que vous avez développées....

A présent : bonne programmation
(en langage machine bien sûr !!!)...

---

# SECONDE PARTIE

# LE HARD

# Chapitre I

## INTRODUCTION : DESCRIPTION EXTERIEURE

Vous êtes l'heureux possesseur d'une merveilleuse machine:

Elle possède des progammes mathématiques très puissants: dérivées, intégrales, développements limités; ainsi qu'une très grande facilité de calcul pour les matrices, les complexes, les expressions algébriques... sans parler de ses performances en langage machine. Toutefois les capacités de la 28C ne sont pas adaptées à ses qualités. La faible mémoire, la lenteur et l'absence de connections extérieures n'ont pu échapper à votre regard averti. Quant à la 28S il lui manque une connection extérieure pour sauvegarder des programmes ou des données que l'on aimerait placer à l'abri des "memory lost".

Vous trouverez donc dans cette partie les explications nécessaires pour faire en sorte que votre machine puisse parfaitement répondre à vos besoins :

- **Comment ouvrir votre 28. (ttv)**
- **Comment lui adjoindre une alimentation externe. (ttv)**
- **Comment l'accélérer. (ttv)**
- **Comment la connecter à un ordinateur. (ttv)**
- **Comment lui rajouter de la mémoire. (!c)**
- **Comment la refermer. (ttv)**
- **Des idées.**

**AVANT DE LIRE CE QUI SUIT, CONSULTEZ L'AVERTISSEMENT page (9).**

Une description extérieure de votre machine est nécessaire de manière à identifier les zones qui feront l'objet de nos propos.

Votre machine est formée de trois grandes parties faciles à reconnaître: le clavier, l'électronique, les piles. Le problème est de pouvoir les délimiter dans l'espace pour éviter de faire des bêtises irréparables en perçant, sciant...

## LE CLAVIER

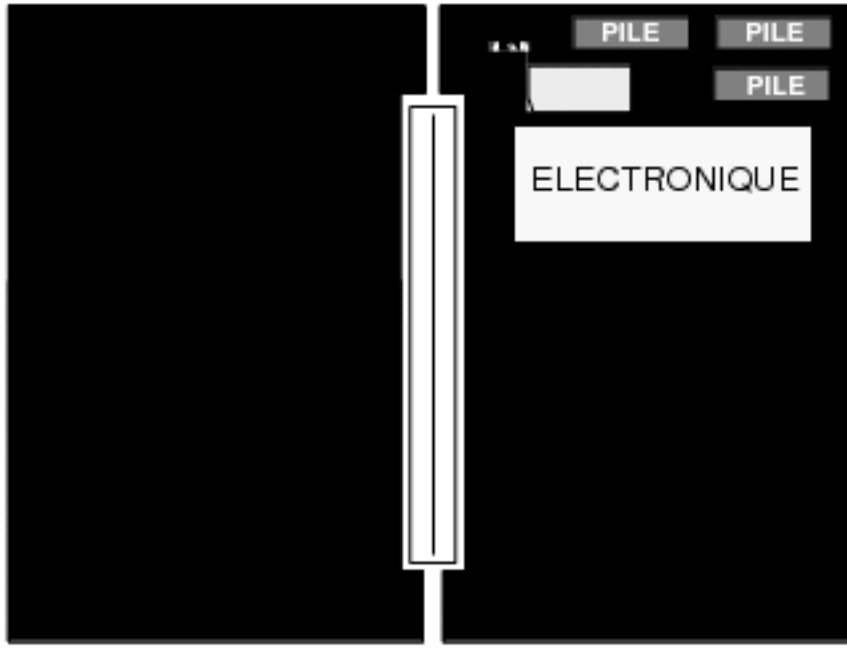
Il est formé de deux feuilles de plastique sur lesquelles sont dessinés les contacts des touches et d'une feuille de caoutchouc qui assure le contact physique entre les touches du clavier et les feuilles. Tous les dessins électriques aboutissent à une rangée de 21 pistes qui prennent contact avec la carte électronique par pression juste en haut du clavier de droite en dessous du bas de l'écran.

## L'ELECTRONIQUE

Elle est constituée par une seule carte qui se loge sous l'écran et comprend la diode infrarouge. Sur cette carte sont soudés les mémoires et autres composants. C'est là que nous travaillerons. La face composants que l'on modifiera est au-dessous.

## LES PILES

Elles occupent la place restante juste au-dessus de l'écran.



# Chapitre II
## OUVERTURE

**AVANT DE LIRE CE QUI SUIT, CONSULTEZ L'AVERTISSEMENT Page (9).**

Pour toute transformation électronique de la machine, il faut commencer par ouvrir votre HP de manière à pouvoir accéder à la carte mère. Malheureusement, contrairement à la plupart des autres machines, elle ne possède ni vis ni clapet qu'il suffirait de retirer pour que la machine s'ouvre.

Elle est fermée au moyen de plus d'une centaine de tiges de plastique qui partent du fond de la machine et dont la tête a été fondue puis écrasée sur la partie supérieure.

Pour ouvrir, il suffirait donc de percer tous ces têtes, la partie inférieure se séparerait alors de la partie supérieure sans forcer. Ces têtes sont placées sous l'autocollant gris du clavier et l'autocollant marron qui possède l'inscription HP28 au-dessus de l'écran. Toutefois cette première méthode a l'inconvénient de désassembler aussi le clavier de droite et la fermeture ne sera pas parfaite.

Pour ouvrir, nous procéderons donc un peu différemment: nous allons scier la partie supérieure, entre le clavier et l'écran, de manière à extraire la partie écran-électronique-pile uniquement.

**MATERIEL :**

tournevis fin (3mm);
mini perceuse munie d'une scie en disque;
un cutter;
une petite pince plate;
une feuille de plastique (type intercallaire).

**DUREE : 1h**

**1-** Retirez les piles.

**2-** Retirez le cache gris du clavier en le décollant doucement, à l'aide du tournevis, en commençant par le coin inférieur droit. Attention ! ne le pliez pas. Une fois décollé, collez-le sur la feuille de plastique de manière à ce qu'au moment de la fermeture, il puisse être remis en place sans ajout de colle. (Photo II-1)

**PHOTO II-1**

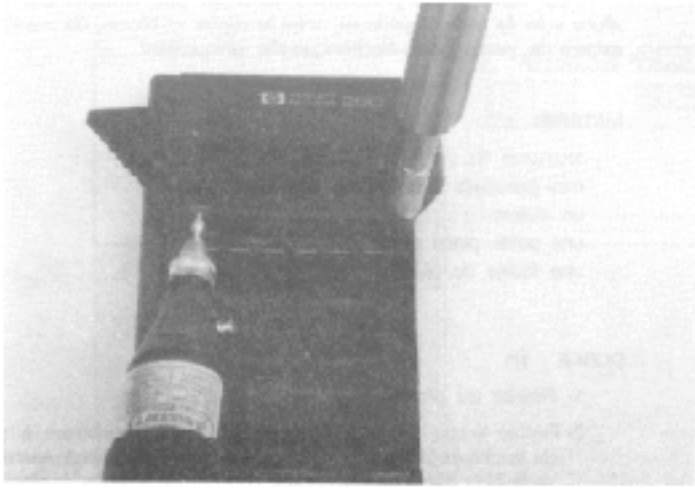


**PHOTO II-2**

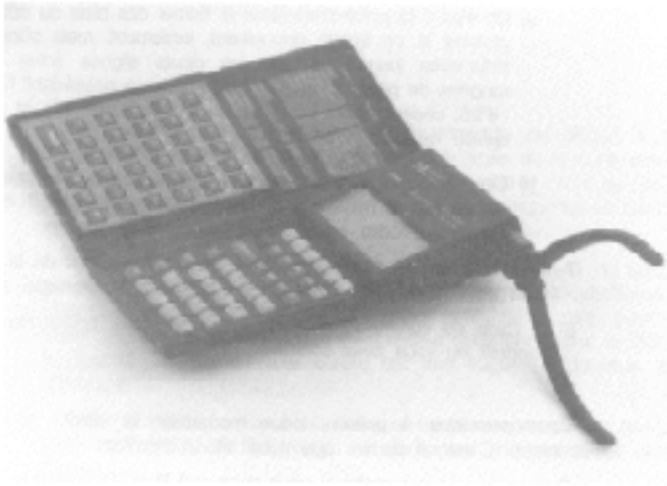


**PHOTO II-3**

**3-** Placez du sparadrap à l'endroit du coude au bas de l'écran pour que l'axe de la scie ne fasse pas de marques sur le plastique. (Photo II-2)

**4-** Sciez dans le creux juste en haut du clavier au bas de l'écran, en prenant soin de ne pas attaquer les bords à droite et à gauche qui appartiennent à la partie inférieure de la machine, de manière à ne pas altérer sa rigidité. ATTENTION ! il faut scier sur une profondeur n'excédant pas 3mm, car au-dessous se trouve la feuille de plastique assurant la liaison entre le clavier et la carte électronique. Pour les parties où le plastique ne sera pas entièrement scié, on finira au cutter (en l'occurence pour les bords).

**5-** Finir de couper avec le cutter: d'abord dans la fente en écartant pour s'assurer de la séparation effective entre le bas (le clavier) et le haut (l'écran); puis coupez les bords intérieurs (qui n'ont pu être sciés auparavant) avec le cutter.

**6-** Deux méthodes pour finir sont possibles :

**a)** On insère la pince plate dans le boîtier des piles du côté le plus profond et on écarte doucement, lentement, mais sûrement les mâchoires jusqu'à ce que les picots alignés entre les deux rangées de piles, sous l'autocollant marron possédant l'inscription HP28, cèdent. C'est la méthode que j'emploie; elle et plus sûre qu'elle ne paraît.

**b)** On décolle l'autocollant marron possédant les inscriptions HP28 au dessus de l'écran et on perce avec un forêt de 3 mm toutes les têtes rondes en plastique. (Photo II-3)

**7-** Avec le cutter, on écarte en haut à gauche au-dessus de la charnière pour aider à déboîter notre partie contenant électronique et écran.

**8-** A l'aide du cutter, achevez de disjoindre notre boîtier du clavier à l'endroit ou vous avez scié.

Vous possédez à présent deux morceaux: le clavier et le boîtier électronique. C'est ce dernier que nous allons modifier.

*Remarque* : si, par malheur, vous avez scié la feuille plastique du clavier, vous pouvez refaire les contacts altérés avec de la colle ou de la peinture conductrice.

Une fois cette étape terminée, il est recommandé d'épousseter vos deux parties de HP, de manière à éliminer les poussières de plastique engendrées par la scie.

---

# Chapitre III
# DESCRIPTION INTERIEURE

A présent, pour retirer la carte électronique du boîtier, il suffit d'insérer une lame de cutter à gauche ou à droite de la carte entre le boîtier et la languette métallique (qui constitue en fait le cadre de l'écran), puis de soulever un peu pour extraire la carte des encoches en plastique en ces deux endroits.

La carte a une face composants et une face écran. Il est possible de séparer l'écran de la carte en détordant les languettes métalliques qui s'appuient sur la carte mais ce n'est pas nécessaire pour les transformations proposées ici, et c'est même déconseillé pour la 28S car le dessus des puces (côté écran) est très fragile (le dessous aussi d'ailleurs).

Sur la face écran, on peut distinguer les points de contacts des piles juste au-dessous de l'infrarouge, ce sont deux grandes pistes carrées dorées.

Sur la face composants on peut apercevoir, (orientation de la carte: infrarouge vers le haut, face composants vers vous):

**POUR LES 28C:**

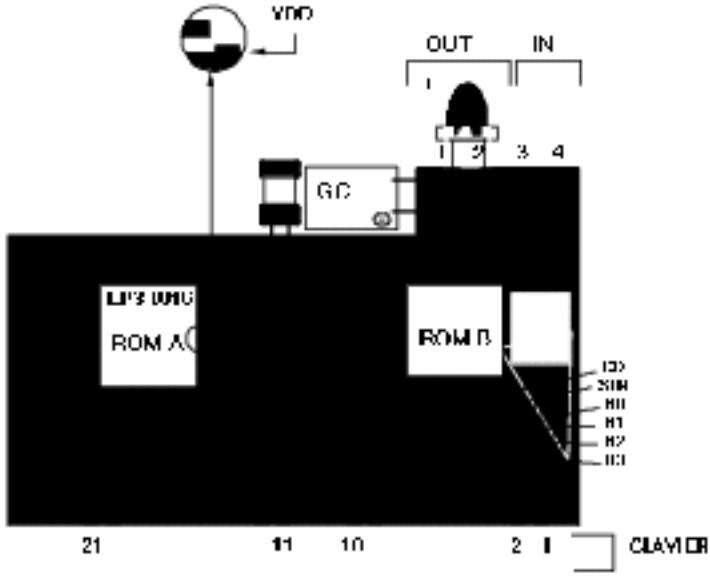



**FIGURE III-1**

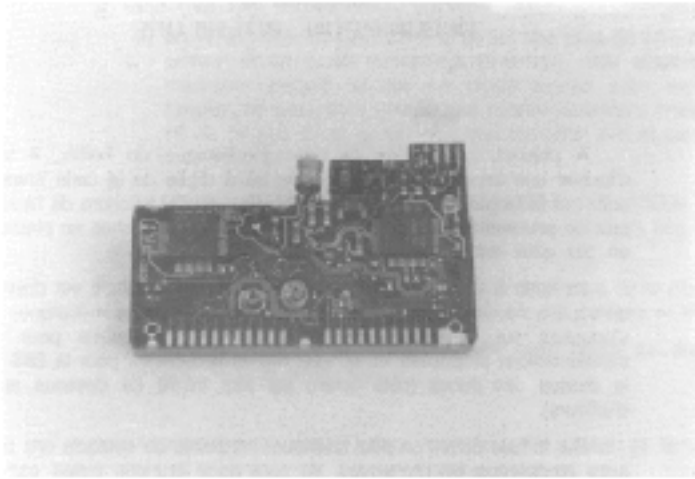


**PHOTO III-2**

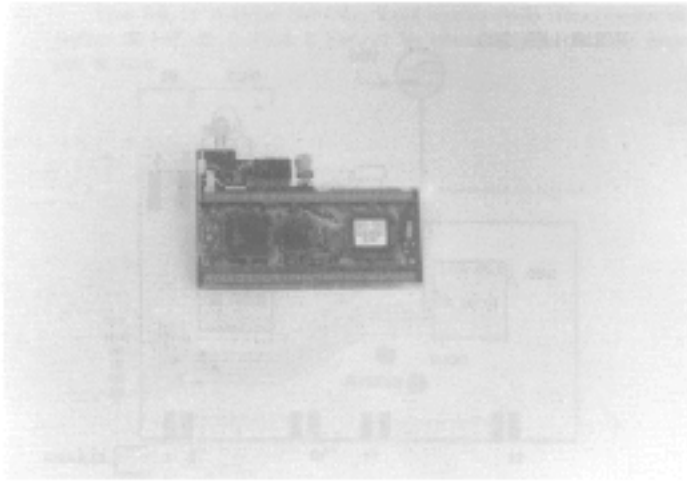


**PHOTO III-3**

- 2 blocs noirs avec beaucoup de pattes Ce sont les Roms. La Rom A est celle de gauche la Rom B est celle de droite.
- Une rangée de pistes parallèles sur le bas qui forment les contacts avec le clavier; il y en a 21.
- Un gros cylindre noir en haut qui sort de la carte: c'est le condensateur qui assure la régularité de l'alimentation d'entrée. (Photo III-2)
- Une série de 6 petits disques d'or en dessous de la Rom B qui sont les contacts pour les modules (bus, str et cd). Ce sont les 6 disques d'or placés au milieu des six premières fines pistes (à partir du bas) qui vont de la Rom A à la Rom B. (Photo III-3)
- Deux petites plaquettes parallèles beiges en dessous du deuxième condensateur cylindrique (blanc placé orthogonalement au gros noir) qui sont deux condensateurs de 33 pico Farrad et à leur droite un petit bloc bleu-noir qui est une inductance de 180 micro Henri. Ces trois composants contribuent à l'élaboration des oscillations dont la période détermine la rapidité de la machine.
- 4 pistes verticales en haut à droite dont les deux premières 1 et 2 sont connectées à l'infrarouge. Le 1 est le (+), le 2 est le (–). Les deux suivantes (3 et 4) forment une entrée: il suffit de mettre en contact ces deux bornes (les deux qui sont situées le plus à droite) pour que le quartet d'adresse juste inférieure à celui de l'émission infrarouge change de valeur. (#4070C pour les 28C et #FFF0C pour les 28S).
- Les deux ressorts sont les contacts pour le buzzer.

**POUR LES 28S :** (Photo III-4)

- En bas face composants, il y a une série de 21 pistes qui forment le contact avec le clavier comme pour les 28C.
- Au milieu, à droite et à gauche, les deux carrés brillants dans les trous sont les Roms Rams microprocesseur et display driver. Eh oui! tout cela est concentré dans ces deux véritables merveilles technologiques. Ces puces sont tenues sur la carte par suspension grâce à une centaine de fils d'un micron de diamètre qui assurent la liaison électrique entre le circuit et la puce.
- En haut se trouve un gros cylindre noir qui assure la régularité de l'alimentation d'entrée, comme pour la 28C.
- Un peu à sa droite, se trouve un cube noir avec une barre argentée, c'est un condensateur de 10 micro Farrad.

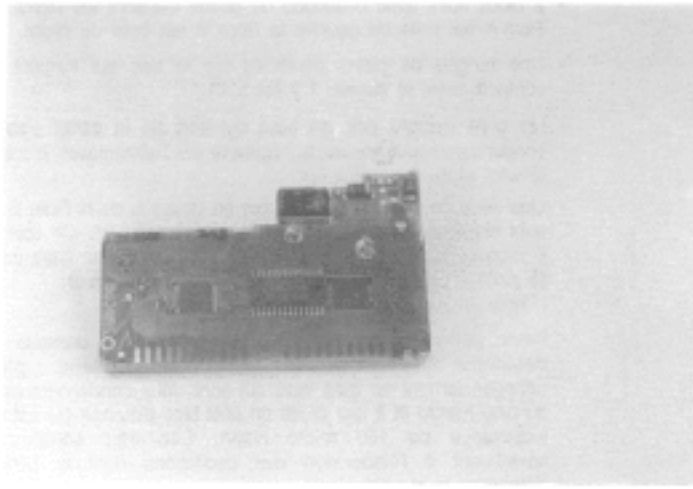


**PHOTO III-4**

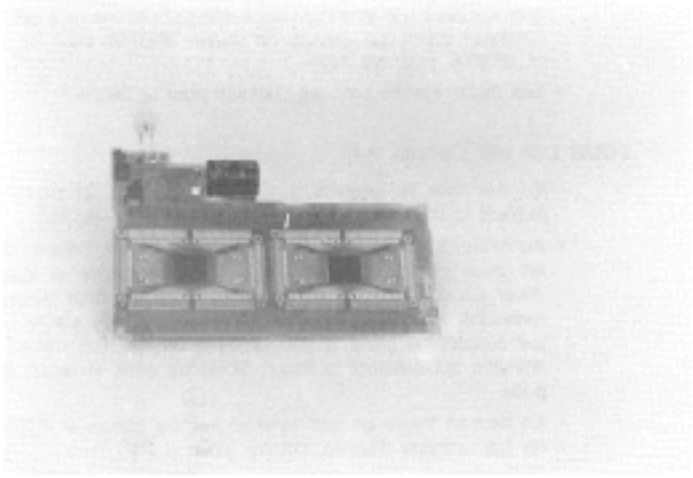


**PHOTO III-5**

- Au-dessus, se trouve la diode infrarouge et ses deux contacts sur le circuit.
- A droite se trouve, un bloc bleu noir: c'est une inductance de l'ordre de 100 micro henri qui intervient dans le circuit qui assure les oscillations responsables de la vitesse de la machines. les deux condensateurs qui l'accompagnaient dans la 28C ont été simulés au sein des puces, ce qui permet de changer la vitesse de la 28S par programme.
- A la gauche de ceci, semble se trouver un transistor de sortie pour l'infrarouge (le petit rectangle noir aux trois contacts).
- Les ressorts sont les contacts pour le buzzer.

---

# Chapitre IV
# LES TRANSFORMATIONS

## 1. ALIMENTATION EXTERNE (ttv)

**AVANT DE LIRE CE QUI SUIT, CONSULTEZ L'AVERTISSEMENT Page (9) .**

Il est possible de brancher une alimentation extérieure sur votre HP de telle sorte que, en branchant par exemple un transformateur délivrant du 4.5 volts continu, les piles soient déconnectées automatiquement; vous pourrez alors faire marcher votre HP des nuits entières sans consommer vos piles.

Nous utiliserons pour cela une prise minijack mono à interrupteur.

**MATERIEL :**

cutter;

perceuse;

scotch;

prise minijack mono femelle à interrupteur;

prise minijack mâle;

fer à souder avec prise à la masse pour l'isolation électrique;

étain;

fil électrique très fin avec gaine (fil de wraping gainé.).

**DUREE : 1h**

**1-** Suivez l'arrivée (+) du boîtier des piles vers la carte électronique. C'est celle qui est placée en haut. Placez un bout de scotch sur la piste de la carte électronique à cet endroit de manière à isoler l'arrivée directe du (+) sur la carte. (côté écran).

**2-** Replacez la carte dans le boîtier.

**3-** Coupez les deux supports verticaux en plastique de la diode infrarouge qui appartiennent au boîtier formant le fond de la machine. Ce sont deux plaquettes de plastique parallèles, en haut à gauche, possédant une encoche circulaire dans laquelle se loge la diode lorsque la machine est fermée. (Photo IV-1)

**4-** Percez à cet endroit, juste en dessous du lieu où se trouve la diode, du diamètre de votre prise minijack pour que vous puissiez la faire passer. Faites en sorte qu'une fois placée, elle soit le plus possible vers le fond, de manière à ce qu'il y ait suffisamment de place au-dessus pour les soudures, la diode et le circuit.

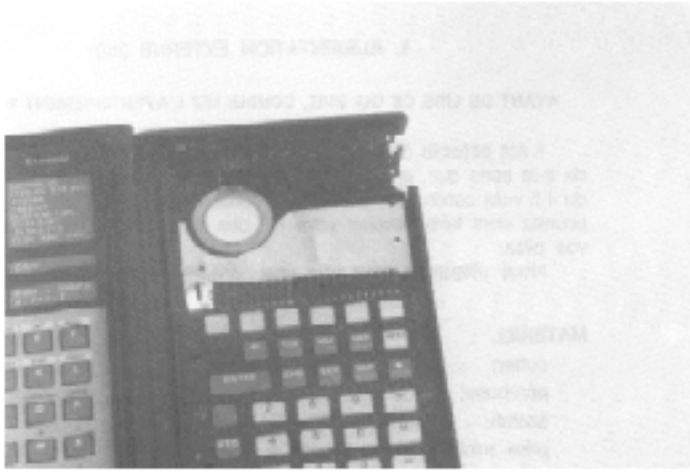


**PHOTO IV-1**

**5-** Installez la prise, contacts vers le haut, et collez-la avec de la glue avant de la visser pour qu'elle ne bouge absolument pas.

**6-** Usinez le boîtier supérieur qui contient la carte électronique, juste au dessous de la diode de manière à ce que ce boîtier puisse se refermer sur le socle sans buter contre la prise avant d'être totalement fermé.

**7-** Soudez le commun de la prise, (la bague externe), sur le (-) du gros condensateur noir, avec un fil court fin et souple.

**8-** Soudez de même la deuxième arrivée de la prise, (le centre), (ce qui sera le (+)) sur le (+) du gros condensateur noir (le côté qui n'est pas marqué d'une bande blanche).

**9-** Soudez enfin le dernier contact (celui qui forme l'interrupteur avec le second contact de la prise) (c'est à dire au (+)) sur la tige de fer derrière le ressort du (+) des piles.

Lorsque vous brancherez la prise mâle, les piles seront alors déconnectées pour laisser la place au transformateur qui aura intérêt à être alimenté, sinon vous perdrez votre mémoire par manque de courant. C'est pourquoi il est conseillé de faire le branchement machine éteinte.

**10-** Soudez le transformateur redresseur sur la prise minijack mâle. Lorsque vous brancherez votre transformateur, vérifiez donc bien qu'il soit en 4.5 volt et polarisé correctement, vérifiez aussi que votre machine est éteinte lors du branchement de la prise minijack de manière à éviter les "memory lost". Il existe dans le commerce des transformateurs 220 —> 3 Volts, redresseurs, dont l'une des prises de sortie est de type "minijack", ils conviennent parfaitement, car les tensions de sorties de ces appareils sont toujours surélevées.

---

## 2. ACCELERATION

**AVANT DE LIRE CE QUI SUIT, CONSULTER L'AVERTISSEMENT Page (9).**

### a) POUR LES 28 C

Votre HP28C est ultra performante, mais sa lenteur vous fera sourire, puis rager, puis pester! Quoi de plus frustrant que d'être si proche de la perfection sans pouvoir l'atteindre! Voici donc la solution à vos malheurs.

L'alimentation continue de votre machine arrive dans un circuit oscillant, faisant intervenir deux condensateurs "C" de 33 pF et une inductance "L" de 180 micro Henri. Cet oscillateur fournit des signaux périodiques assimilables à des créneaux de période "T".

Or "T" est directement reliée aux valeurs de "C" et "L". Donc en changeant ces valeurs, on modifie "T". Or "T" est le temps qui s'écoule entre deux créneaux. On modifie donc le nombre de créneaux par seconde. Or à chaque fois que la tension aux bornes du microprocesseur monte ou descend, celui-ci exécute une opération élémentaire, ce qui se produit chaque fois qu'il y a un créneau.

En résumé: la vitesse de la machine dépend directement de "C" et "L". Autrement dit, en changeant "C" ou "L", on modifie la vitesse générale de la machine.

**Tout d'abord "C"** :

Cette opération est d'une extrême simplicité; on supprime les deux condensateurs "C" et on les remplace si on le désire par d'autres.

Voici des exemples de valeurs de C:

| | |
|---|---|
| C=33 pF (normal) | Vitesse normale. |
| C=10 pF | Vitesse 1.7 * plus grande (comme la 28S et l'imprimante suit toujours). |
| C=0.2 pF | Vitesse 2.2 * plus grande cette capacité est la plus petite que l'on trouve sur le marché (l'imprimante ne suit plus). |
| C= rien | Vitesse 2.5 * plus grande (l'imprimante ne suit plus). Ne pas remplacer les condensateurs et laisser les pistes vierges revient à simuler une capacité de l'ordre du 0.05 pF. |

Même avec une accélération maximale de 2.5, la machine suit sans aucun problème puisque avec l'inductance il est possible d'aller jusqu'a 3.5. Mais là, c'est vraiment le maximum !

La différence entre une 28C accélérée à 2.5, et une 28C normale est stupéfiante et cette rapidité est d'un très grand confort. Exemple: pour tracer la courbe sinus en paramétrage par défaut; une 28C accélérée termine (4 bosses: une en haut, une en bas, une en haut, une en bas) lorsqu'une 28C normale vient juste d'achever le haut de la première bosse. Fantastique! Sans parler de l'auto-test !

## OPERATION

**MATERIEL :**

fer à souder 15 watt;
étain;
pince plate;
2 condensateurs si nécessaire.

**DUREE : 10min.**

**1-** Avec une tresse à dessouder, retirez la soudure aux deux bornes de chacun des condensateurs "C" après les avoir repérés sur la carte. (cf description intérieure). Ce sont deux petites plaquettes beiges parallèles au-dessous du cylindre blanc.

**2-** Avec une pince, faites pivoter l'un des condensateurs en chauffant l'une des deux pistes jusqu'à ce qu'il cède. Il ne vous resservira plus.

**3-** Avec le fer et la tresse, nettoyez les deux pistes des restes de soudure, et des morceaux éventuellement abandonnés par le condensateur après ce rude traitement.

**4-** Faites de même avec l'autre.

Si vous voulez vous servir d'une imprimante vous pouvez remettre deux "C" de 10 pF en les soudant sur les pistes citées ci-dessus à la place de leurs prédécesseurs.

**Pour "L" :**

Il faut abaisser la valeur de "L", soit en lui joignant une self en parallèle, soit en la remplaçant par une plus faible. La valeur de la self est à déterminer en fonction des condensateurs.

Il faut savoir que diminuer la self pose des problèmes de stabilité pour deux raisons : les selfs du marché nous donnent peu de choix, et la self ne peut être trop faible si les condensateurs le sont car il y a alors un déséquilibre du système oscillant, ce qui l'empêche d'osciller.

Une bonne solution est de mettre dans "C" des condensateurs de l'ordre de 5 pF et de mettre en parallèle avec "L" une self de 100 micro Henri. On peut atteindre ainsi une accélération de l'ordre de 3.2.

Toutefois, changer la self n'a qu'un intérêt limité; il est donc conseillé de s'en tenir à l'élimination des deux condensateurs.

**b) POUR LA 28S**

Comme vous pouvez le voir sur la photo de l'intérieur de la 28S il n'y a plus de condensateurs mais une self de 100 micro Henri. Il suffit donc de placer en parallèle une self de quelques dizaines de micro Henri. La vitesse pourrait être accélérée jusqu'à trois fois. Malheureusement pour nous, la théorie et la pratique sont ici dissociées par le fait que votre merveilleuse 28S sait faire une chose que vous ne soupçonnez certainement pas: en utilisant la vitesse d'horloge réglée par le quartz de l'écran LCD, elle change la valeur des "C" simulés dans ses puces. En d'autres termes, plus vous changez "L", plus elle corrige "C". (Jusqu'à un certain point de non retour!).

De plus cette accélération physique est inutile car il est possible sur cette machine (contrairement aux 28C) d'accélérer jusqu'à deux fois en pokant à une adresse de la manière adéquate, (voir le programme SPEED).

**Pour 28c seulement.**

Les 28s sont ce que l'on appelle "hard configured", ce qui signifie que les adresses attribuées aux cases mémoires sont fixées pour toujours de manière électronique. Donc si on rajoute de la mémoire sur une 28s, elle ne sera pas reconnue par le microprocesseur. Au contraire les 28c sont "soft configured", ce qui signifie que, au démarrage ou au cours d'un "memory lost", la mémoire est réorganisée et les adresses sont de nouveau attribuées. Ainsi une mémoire rajoutée sera reconnue après un "memory lost".

Comme toujours nous vous demandons de prendre connaissance de **L'AVERTISSEMENT** au début du livre vous signalant que si vous décidez de transformer votre machine, ce sera sous votre unique responsabilité.

Ce chapitre vous expliquera comment vous pouvez rajouter de la mémoire sur votre 28C.

Le processeur de votre merveilleuse 28 est très semblable à celui du HP71, il reconnaît donc parfaitement un module mémoire RAM pour HP71. Ces modules existent en 64, 32 et 4 kilooctets; ils ont tous à peu près la même taille et la même forme.

La première chose à faire est donc de se procurer un tel module.

Ces modules ne sont pas produits par HP, mais par la société CMT. Les 4K peuvent être trouvés d'occasion pour une somme allant de 200 à 500 Fr. Les 32K sont très rares en occasion et valent entre 1000 et 2000 Fr; neufs, on peut les trouver en France pour 2300 Fr mais il est plus économique de les commander aux USA. Les 64K sont introuvables d'occasion et ne peuvent être commandés, pour l'instant, qu'uniquement qu'aux USA et semble-t-il uniquement à la société Educalc. Celle-ci commercialise tout le matériel HP et CMT. Voici donc, à titre indicatif, les références et les prix (au catalogue N°38) qui pourraient vous être utiles.

**Educalc**

**adr :** 27953 Cabot Road
Laguna Niguel
CA 92677
U.S.A tél: 19-1-714-582-26-37 (depuis la France (attention! 9 heures de décalage en moins.))

| | | |
|---|---|---|
| **module:** | 64K RAM #71-656B | $294.95 |
| | 32K RAM #71-656 | $149.95 |
| | 4K RAM #82-420 | $ 65.95 |

Vous pouvez commandez par téléphone avec une carte bleue internationale.

Dès lors, si vous vous êtes procuré un module par quelque moyen que ce soit, il faut ouvrir votre machine: voir le chapitre correspondant.

Lorsque votre machine sera ouverte, il vous faudra situer 9 contacts sur la carte électronique à deux emplacements précis.

• **Tout d'abord, orientez votre carte comme suit :**

- vue de dos;
- face composants, vers vous;
- la diode infrarouge, vers le haut;
- la série des 21 pistes paralleles servant de connections au clavier lorsque la machine est fermée, vers le bas;
- les deux gros carrés noirs, à droite et à gauche;
- l'emplacement A est la Rom A, (celle de gauche);
- l'emplacement B est la série des 6 disques d'or en dessous de la Rom B, (voir description intérieure).

**contact 1 -** En A. Nous supposerons que dans cette orientation, la ROM possède 4 côtés NORD, SUD, EST, OUEST. sur la face EST (droite) la patte du milieu n'est pas reliée au circuit; en dessous de cette patte, se trouve une piste étamée : ce sera l'alimentation positive du module: VDD. (A, EST, 6ième piste.)

**contact 2 -** En B. C'est le premier disque en partant du bas. C'est la quatrième piste du bus: B3. (B, 1er disque.)

**contact 3 -** En B. On suit la seconde piste. Le disque que nous cherchons est un peu au-dessus du précédent plus à droite, c'est B2. (B, 2ème disque.)

**contact 4 -** En B. Idem : troisième piste, le disque est au-dessus du premier, un peu sur la gauche, c'est : B1. (B, 3ème disque.)

**contact 5 -** En B. Idem : quatrième piste, le disque est au-dessus de celui de B2 un peu sur la droite, c'est : B0. (B, 4ème disque.)

**contact 6 -** En B. Idem : cinquième piste, le disque est exactement au-dessus de B3 après trois fils de piste, c'est : STR. (B, 5ème disque.)

**contact 7 -** En B. Idem : sixième piste, le disque est au-dessus de B0 un peu à gauche, c'est: CD. (B, 6ème disque.)

**contact 8 -** En A. C'est ce qui sert à faire le chaînage avec la Ram préexistante. Sur la Rom A côté sud seconde patte à partir de la droite c'est: DOUT. (A, SUD, 2ème patte à partir de la droite.)

**contact 13 -** En A. C'est la masse que nous appellerons contact 13 pour la correspondance sur les pistes du module. Il est situé sur le côté ouest de la Rom A quatrième patte en partant du haut c'est: GND. (A, OUEST, 4ème patte en partant du haut.)

**CONSEILS :**

Cette opération contrairement aux autres : ouverture, accélération, alimentation, connecteurs, fermeture,... est beaucoup plus délicate puisque vous allez devoir souder sur le bus qui transporte les informations entre deux endroits fragiles: les Roms et le microprocesseur.

De plus, les endroits pour souder ne sont pas très larges. Nous vous conseillons donc un fer qui chauffe peu (15 Watt) possédant une tête très fine et très propre.

Il est fortement conseillé d'isoler votre fer le mieux possible à la terre car à la moindre fuite de courant dans le bus, vous pouvez griller le processeur.

Lorsque vous chauffez, restez sur le contact le moins de temps possible pour ne pas surchauffer les puces.

**MATERIEL :**

fer à souder 15 watt;

étain faible diamètre (1mm);

fil gainé, très fin, monofil que l'on utilise d'ordinaire pour wrapping;

un module mémoire ram pour HP71;

une miniperceuse munie d'une fraise (diamètre 5mm faible profondeur 2mm);

un pinceau (pour retirer les poussières).

**DUREE : 2 à 3 heures**

**1-** Retirez les piles de votre machine.

**2-** Ouvrez votre machine (voir chapitre correspondant), et court-cicuiter les bornes du gros condensateur noir pendant quelques secondes afin de vous assurer de l'absence de courant dans les mémoires de la machine.

**3-** Ouvrez le module en écartant le couvercle du boîtier avec une lame, en prenant soin de ne pas toucher l'intérieur avec cette même lame.

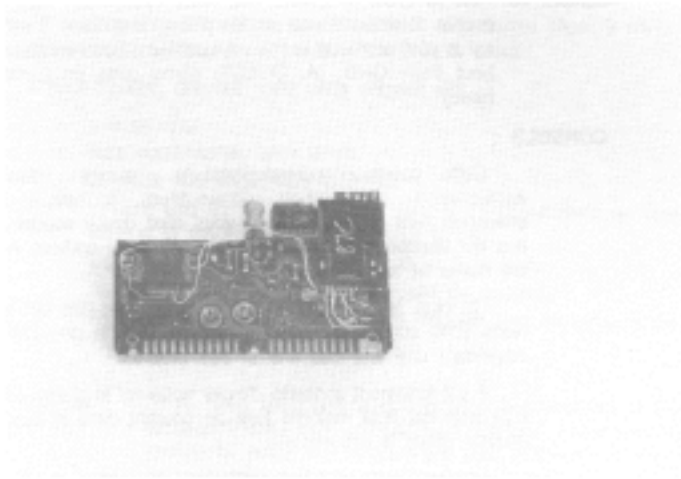
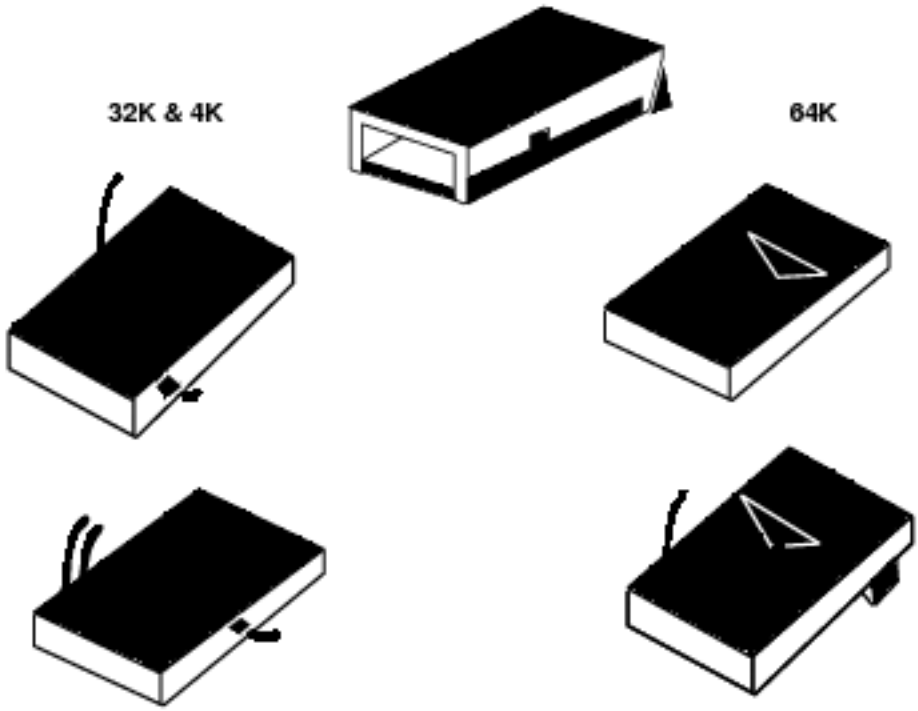


**PHOTO IV-3**



**4-** Dessoudez toutes les grandes pattes qui assurent normalement le contact entre le module et la carte du 71.

Ces pattes sont au nombre de 13, parallèles, et du même côté sur les 32K 4K et 64K. Sur les 64K en plus, de l'autre côté, deux pattes servent à prendre contact avec les micropiles plates qui assurent le maintien de la mémoire lorsque le module est déconnecté du 71. Nous n'en n'aurons pas besoin car le module sera toujours soudé sur l'alimentation de la machine. Vous les dessoudez donc de la même manière.

**5-** Mettez une pointe de soudure sur les neuf contacts de la carte électronique; en prenant bien garde que cette soudure ne touche pas les pistes ou pattes qui l'environnent.

**6-** Repérez l'emplacement des neufs contacts sur le module : face opposée à celle possédant les gros blocs noirs, vers vous; la série des 13 trous dans lesquels se logeaient les grandes pattes que vous avez dessoudées, vers le haut.

De gauche à droite les contacts correspondent aux numéros cités au début de 1 à 8 plus le 13.

**7-** Le module sera placé à droite par-dessus la Rom B. Pour les 4K et 32K, le module sera placé face composants (côté des blocs noirs) sur la Rom: le gros bloc noir étant à placer en dessous de la Rom B et contre celle-ci, (en quinconce), de manière à ce qu'il occupe le moins de place possible. (On peut aussi le placer au dessus de la Rom A de la même façon, cela n'a aucune importance). (Photo-IV-3)

Pour les 64K: face verte sur la Rom B, face noire vers vous, série des 13 contacts vers le clavier, placé le plus possible vers l'avant (vers l'infrarouge), en restant dans la partie du boîtier qui contient la carte électronique sans déborder sur l'emplacement des piles. En résumé, placez-le de telle sorte qu'il prenne le moins de place possible en épaisseur.

**8-** Avec le fil de wrapping, connectez les 9 contacts en tenant compte de la disposition du module et en donnant juste aux fils la longueur nécessaire.

Vous pouvez recouvrir toute la partie électronique une fois modifiée de la sorte (sauf sur les ressorts du buzzer et les contacts du clavier) avec une large bande de sparadrap; dans le but d'empêcher les fils de se casser ou de se dessouder par les chocs éventuels et répétitifs que votre machine pourrait subir. Mais ce n'est pas indispensable.

**9-** Avec la fraise, fraisez le fond de la machine en vis-à-vis du module, en prenant de la marge. creusez le plus profond possible sans pour autant percer la paroi. Les 64K sont plus fins et rentrent très bien. Mais, pour les autres, il est conseillé de fraiser au maximum, si possible de percer la paroi juste de la taille du module et de mettre un cache de plastique par derrière une fois la machine refermée.

Normalement il n'est nécessaire de percer que si le module force trop au cours de la fermeture.

**10- ATTENTION** : Vérifiez que la fermeture de la machine ne serre pas trop le module pour éviter de le détruire. Si c'est nécessaire, fraisez plus profond ou percez la paroi que vous recouvrirez par un cache de l'extérieur.

**11-** Refermez la machine, (voir chapitre correspondant) ou passez à une autre transformation.

**12-** Si le capot a tendance à se décoller et à se réouvrir après la fermeture telle qu'elle est indiquée au chapitre correspondant et si le module ne force pas trop: vous pouvez, en maintenant la machine fermée à fond, percez sur les flancs gauche et droit, à la hauteur du bas de l'écran, un trou de l'ordre de 1 mm de diamètre et de 2mm de profondeur (soit juste les deux épaisseurs de plastique: celle qui englobe l'électronique, et celle qui constitue le fond de la machine).

Dans ce trou vous placerez un morceau de clou fin (ou une fine tige) du diamètre du trou, enduit de superglue de manière à empêcher le boîtier de se soulever.

L'utilisation de cette mémoire est identique à celle que vous possédiez auparavant. Remettez les piles.

Faites "memory lost" ([ON] [INS] [RIGHT] simultanément), puis MEM, puis lancez l'autotest ([ON] [LEFT]) : chaque page graphique de points aléatoires constitue 512 octets de mémoire. Plus il y en a, plus vous avez de la mémoire.

**En cas de problème:**

- Si l'auto-test marche mais les pages de points aléatoires s'arrêtent avant la fin; ou, il y a une mauvaise soudure: reprenez-les; ou bien, vous avez trop écrasé votre module : repassez le fer sur les soudures du bloc noir du module et faites en sorte qu'il ne soit plus écrasé.

- Si la machine ne s'allume pas ou reste bloquée, il est possible que lors du montage, il soit resté des résidus d'électricité dans les mémoires, ce qui a désorienté votre processeur; il suffit très souvent de court-circuiter l'entrée des piles dans la machine pour résoudre ce problème.
- Si au cours du test clavier, il vous semble qu'une ligne ou une colonne de touches ne marche plus, assurez-vous tout d'abord qu'elle ne marche effectivement plus. Si c'est le cas, il faut réouvrir la machine et nettoyer les contacts du clavier et ceux de la carte qui leur correspondent, à l'aide d'un bout de chiffon et d'un pinceau pour épousseter.

**Pour chaîner plusieurs modules:**

Le numéro 9 du module est le DOUT. Pour chaîner branchez tout en parallèle sauf le DIN et DOUT. Connectez le DOUT de la machine sur le DIN du premier module, c'est-à-dire le DOUT de la machine: contact 8, sur le numéro 8 du module; le DOUT du premier sur le DIN du second, soit le numéro 9 du premier sur le numéro 8 du second, le DIN du second sur le DOUT du troisième... et le dernier DOUT reste non connecté.

Vous pouvez chaîner ainsi jusqu'à concurrence de 512K, sachant que vous avez déja au départ de la mémoire utilisée: 128K de Rom et 2K de Ram, vous pouvez donc rajouter jusqu'à 380K de Ram.

---

# Chapitre V
# ENTREES / SORTIES (ttv)

## A] DESCRIPTIONS

### 1. LES SORTIES

*a] LA DIODE INFRAROUGE*

Cette diode est la meilleure sortie que nous ayons; en effet elle offre quatre niveaux d'éclairages en fonction de la valeur d'un quartet de la Ram. Le bit de poids le plus fort détermine le degré d'éclairement. On a alors:

| valeur du quartet | force d'émission |
|---|---|
| 0000 = 0 | nulle (éteinte). |
| 0001 = 1 | max. |
| 0010 = 2 | moy. |
| 0011 = 3 | max. |
| 0100 = 4 | mini. |
| 0110 = 6 | moy. |
| 0111 = 7 | max. |
| 1000 = 8 | nul. |
| 1001 = 9 | max. |
| 1010 = A | moy. |
| 1011 = B | max. |
| 1100 = C | min. |
| 1101 = D | max. |
| 1110 = E | moy. |
| 1111 = F | max. |

Le bit de poids le plus fort est le premier non nul en partant de la droite. Il y a peu de différence entre max et moy et peu entre nul et min. **Attention !** Même lorsque la machine est éteinte, la diode reste allumée. Pour l'éteindre il faut faire un arrêt système, un CR, ou mettre un zéro dans le quartet en question.

L'adresse de ce quartet est #adresse: #4070D pour les 28C;
#FFF0D pour les 28S.

Pour allumer la diode au maximum il suffit donc de faire:

**IRON** « #adresse "1" POKE »

et pour l'éteindre:

**IROFF** « #adresse "0" POKE »

Ou encore de faire le programme en langage machine suivant. Ce qui a pour effet d'allumer la diode sur une durée écrite en hexadécimale, (écrite à l'envers dans le programme), cette valeur étant de l'ordre de 400(en hexa) pour une seconde.

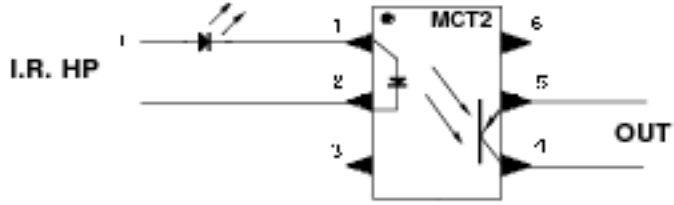
**IR1**

```
pour 28C   modif.28S                    commentaire

69C20                    CON(5) prologue prgm LM.
E3000                    CON(5) longueur du prgm.
8F2EE40     8F18050      GOSBVL #*****   sauvegarde-regis-
                                         tres D0 D1 B et D.
1BD0704     1BD0FFF      D0=*****       )D0=adresse de l'IR.
301                      C=1        P   ) force d'émission.
1540                     DAT0=C     P   ) allumage de la
                                           diode.
3400010                  LCHEX  01000   ) LONGUEUR dans C.
CE                  L1   C=C-1      A   )
5DF                      GONC L1         boucle.
300                      C=0        P    force d'émi.nulle.
1540                     DAT0=C     P     diode éteinte
8F91F40     8F8B050      GOSBVL #*****   récupération reg.
142                      A=DAT0 A
164                      D0=D0+S
808C                     PC=(A)
```

Les deux bornes, sur lesquelles est branchée la diode émettrice, délivrent du 4,5 Volts faible ampérage (niveau TTL). Il suffit donc de connecter une diode lumineuse (pour voir quand vous émettez) en série avec un photocoupleur à la place de la diode infrarouge pour obtenir une sortie. Le (+) est (vue face composants vers vous), sur la carte électronique, la borne de gauche de l'infrarouge et (-) celle de droite.

*Voici le schéma de principe:*

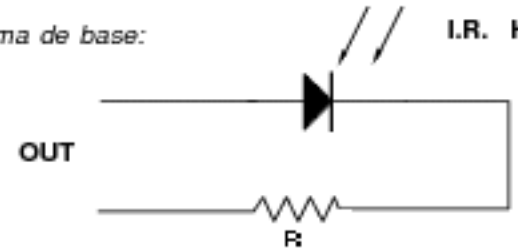



Nous verrons dans le chapitre suivant, l'utilisation pratique de cette sortie: en effet, nous supprimerons la diode infrarouge et mettrons un petit connecteur discret à la place, sur lequel nous

pourrons brancher soit la diode infrarouge, soit notre sortie à base de photocoupleur (MCT2 par exemple).

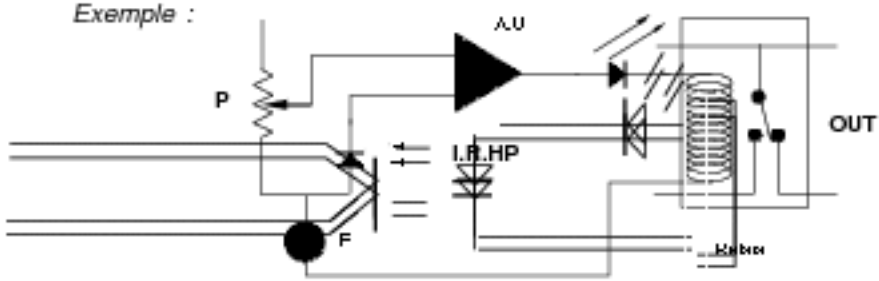
Il est aussi possible de recevoir les informations directement par les émissions de la diode sans avoir à ouvrir la machine, ceci à l'aide d'un phototransistor ou récepteur infrarouge qui se présente sous forme de diode qui est passante lorsqu'elle reçoit un flux suffisant d'infrarouge et non passante dans le cas inverse.

*Voici le schéma de base:*




R étant à fixer en fonction de l'utilisation, qui pourrait être par exemple un circuit comparateur à base d'amplificateur opérationnel.

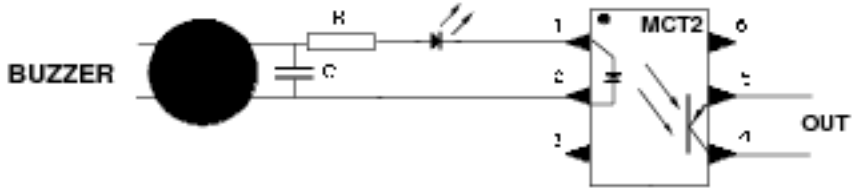
*Exemple :*




(E peut être l'alimentation positive de l'AO. P un potentiomètre de 10K et R un micro relais ILS Gunther 3753.1231.051)(mais ceci est un choix totalement arbitraire!!)

*b] LE BUZZER*

Les deux bornes du buzzer forment une sortie tout à fait utilisable comme pour l'infrarouge. Le circuit de sortie est le même que pour l'IR ; cependant, il faut rajouter un redresseur entre les bornes du buzzer et le photocoupleur: (R=50K C=1micro).



Il suffit alors de BEEPer en RPN ou en LM

**OUT1** « 1000 1 BEEP »

**OUT2**

- Dans C on met la durée multipliée par mille, écrite en Hexa inversé.
- Dans D on met la fréquence en Hz tranformés en Hexa inversé et on appelle la routine du BEEP.

```
pour 28C   modif.28S
69C20                        CON(5)  prologue
43000                        CON(5)  longueur
8F2EE40     8F18050          GOSBVL  #*****
34  00400                    LCHEX   00400
D7                           D=C     A
34  00500                    LCHEX   00500
8F  *****                    *****   =C6101  BEEP  1BB
                                     =3A101  BEEP  1CC
                                     =3DA22  BEEP  2BB
8F91F40     8F8B050          GOSBVL  #*****
142                          A=DAT0  A
164                          D0=D0+5
809C                            PC=(A)
```

## 2. LES ENTREES

*a] LE CLAVIER*

Le clavier est bien entendu le maître. En effet il permet à lui seul un choix de 79 entrées.

Sur le plan de la carte électronique de la 28C les contacts du clavier sont numérotés de 1 à 21 (orientation face composants vers le haut, infrarouge vers l'avant, contacts clavier vers vous, on numérote de droite à gauche de 1 à 21 sans compter la grosse piste complètement à droite qui est une masse); cette numérotation et cette disposition sont aussi valables pour la 28S.

Pour simuler l'appui d'une touche, il suffit de placer une résistance entre le numéro de la ligne et celui de la colonne correspondant à la touche. On réalise alors que l'ont peut simuler des touches qui n'existent pas au clavier elles sont au nombre de 13 on les appelera donc S1 ... S9 SA SB SC SD.

Voici le tableau de correspondance de ces touches:

| | 9 | 8 | 1 | 2 | 3 | 5 | 4 | 7 | 10 | 13 | 15 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **21** | (S1) | (S2) | (S3) | (S4) | (S5) | (S6) | INS | DEL | UP | DOWN | LEFT | RIGHT |
| **20** | A | B | C | D | E | F | SHIFT | MODE | TRIG | SOLV | USER | NEXT |
| **19** | G | H | I | J | K | L | (S9) | ENTER | CHS | EEX | DROP | BACK |
| **17** | M | N | O | P | Q | R | ' | 7 | 8 | 9 | / | (SD) |
| **16** | S | T | U | V | W | X | STO | 4 | 5 | 6 | * | (SC) |
| **14** | Y | Z | # | [ | [ | ( | EVAL | 1 | 2 | 3 | - | (SB) |
| **11** | (S7) | SPC | " | = | LC | α | (S8) | 0 | . | , | + | (SA) |

| | 6 |
|---|---|
| **12** | ON |

La résistance à placer entre une ligne et une colonne est de l'ordre 400 Ohms pour les 28C et 4 000 Ohms pour les 28S. L'utilisation du clavier peut être ou bien en RPN ou bien en LM.

**RPN :**

- Vous détectez la touche appuyée par un

```
« DO UNTIL KEY END »;
```

- Ou bien vous détectez le temps entre deux appuis de touche à l'aide de l'horloge (#horloge : 1BB=#123E, 1CC=#1266, 2BB=#11CA).

```
« DO UNTIL KEY END DROP
   #horloge SYSEVAL
   DO UNTIL KEY END DROP
   #horloge SYSEVAL SWAP - B->R 8192 / »
```

(le temps renvoyé est en secondes).

**LM :**

- La routine employée est

**pour les 28C:**

| | |
|---|---|
| 32zyx | LCHEX xyz |
| 801 | OUT=C |
| 803 | C=IN |

**pour les 28S:**

| | |
|---|---|
| 32zyx | LCHEX xyz |
| 8F1DA10 | GOSBVL 01AD1 routine faisant OUT=C C=IN |

Où xyz est une valeur hexadécimale qui signale sur quelle(s) ligne(s) on envoie du courant. Après cette routine, on reçoit la

valeur dans "C" qui correspond à la ou les colonnes frappées sur cette ou ces lignes sélectionnées.

*Exemple :* nous voulons tester si la touche V est appuyée, on fixe donc xyz à la valeur 008 et si V est appuyé, on recevra 400 pour les 28C 020 pour les 28S, et 000 si cette touche n'a pas été appuyée.

*Autre exemple :* nous voulons tester l'appui de la touche SPC et V simultanément. On envoie donc xyz=008+002=00A; si on reçoit 000: aucune des deux touches n'a été appuyée; 400 (28C) 020 (28S) on a eu V; 020 (28C) 800 (28S), on a eu SPC; 400+020=420 (28c), 020+800=820 (28S), on a eu SPC et V simultanément.

*Dernier exemple :* on veut tester toutes les touches. On envoie xyz=002+004+008+010+020+040+080=0FE; si on reçoit 000 aucune touche n'a été sélectionnée, dans le cas contraire, une touche a été appuyée. C'est l'équivalent d'un DO UNTIL KEY END DROP.

Voici le tableau des valeurs de C en entrée (valeur des lignes (ce sont les mêmes pour les 28C et les 28S)) et des valeurs en colonnes en fonction des touches:

| OUT=C 28S | | | | | | | | | | | | C=IN |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 100 | 800 | 040 | 020 | 001 | 400 | 002 | 080 | 200 | 004 | 008 | 010 | |
| (S1) | (S2) | (S3) | (S4) | (S5) | (S6) | INS | DEL | UP | DWN | LFT | RGT | 080 |
| A | B | C | D | E | F | SFT | MOD | TRI | SOL | USR | NXT | 040 |
| G | H | I | J | K | L | (S9) | ENT | CHS | EEX | DRP | SWP | 020 |
| M | N | O | P | Q | R | ' | 7 | 8 | 9 | / | (SD) | 010 |
| S | T | U | V | W | X | STO | 4 | 5 | 6 | * | (SC) | 008 |
| Y | Z | # | [ | [ | ( | EVL | 1 | 2 | 3 | - | (SB) | 004 |
| (S7) | SPC | « | = | LC | α | (S8) | 0 | . | , | + | (SA) | 002 |
| 010 | 020 | 800 | 400 | 200 | 080 | 100 | 040 | 008 | 001 | 002 | 004 | |
| OUT=C 28C | | | | | | | | | | | | C=IN |

Un exemple de programme utilisant ceci est WUS ; programme LM réalisant un DO UNTIL KEY END DROP très rapide.

La meilleure application pratique est de mettre tout bonnement la sortie d'un photocoupleur (MCT2 par exemple) en série avec la résistance adéquate aux bornes de la ligne et de la colonne correspondant à la touche que l'on désire simuler. Par convention et pour assurer un bon fonctionnement, il est

conseillé de brancher en borne 5, (collecteur du phototransistor interne au photocoupleur MCT2) sur une ligne et le 4 (émetteur du phototransistor en question) sur une colonne.

Une autre utilisation pratique est de brancher ligne et colonne en sortie d'un relais, microrelais, ou ILS.

L'intérêt de l'ILS est d'être une petite gélule de verre faisant office d'interrupteur, dont la commande de fermeture et d'ouverture se fait par un champ magnétique.

*Application :* on insère la gélule au fond de la machine sous la diode; on connecte ses deux bornes aux contacts de l'une des touches, (S7) par exemple; de l'extérieur on place alors un électro-aimant à proximité de l'emplacement de la gélule (et le plus loin possible du centre de la machine; attention aux Roms); lorsque le courant traverse l'électro-aimant, la touche (S7) est appuyée et peut être reconnue par les programmes de réception habituels.

*b] L'INFRAROUGE (!C)*

Eh oui! seules les 28C ont cette entrée de façon suffisamment apparente pour qu'elle soit utilisable.

A droite des deux bornes de la diode IR, se trouvent deux pistes parallèles entre elles et parallèles aux pistes de la sortie infrarouge. Il suffit de mettre en contact ces deux bornes pour changer la valeur du quartet d'adresse juste inférieure à celle de l'émission, soit #4070C pour les 28C; et #FFF0C pour les 28S si vous réussissez à trouver les deux bornes en question sur cette machine diabolique.

*c] LES PILES*

Fantastiques! On avait une sortie qui ne nécessitait pas l'ouverture de la machine : la diode infrarouge. Et maintenant on a une entrée qui ne nécessite pas non plus l'ouverture de la machine.

Malheureusement la communication est très lente et peu fiable pour les longues transmissions; toutefois un projet est à l'étude.

*Voici le principe :* si la tension aux bornes de la machine devient relativement faible, une case mémoire change et un programme prenant appui sur cette adresse déclenche l'allumage de l'indicateur de piles faibles. Il suffit donc de faire passer l'alimentation de 4,5 à 3 volts par exemple. Pour cette adresse, consultez la partie soft.

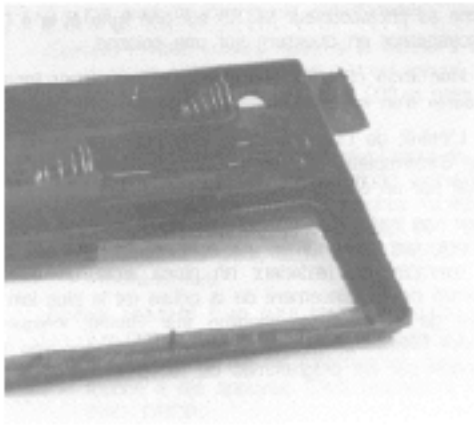


**PHOTO V-6**

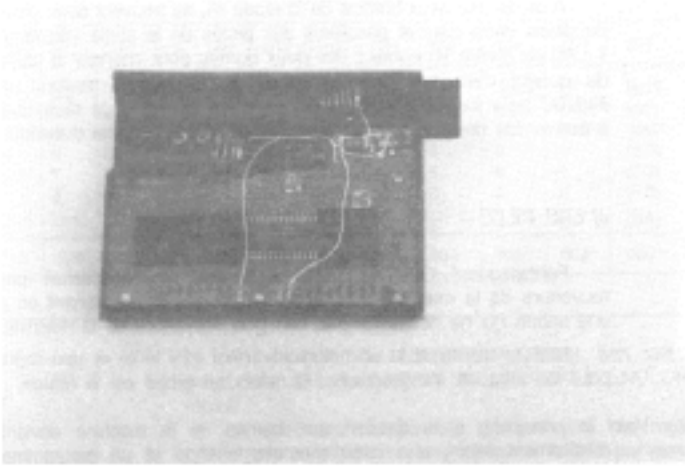


**PHOTO V-7**

**Connecteur sur 28S.**

## B] TRANSFORMATION (ttv)

**AVANT DE LIRE CE QUI SUIT, CONSULTEZ L'AVERTISSEMENT Page (9).**

Vous avez une HP28, vous avez souvent beaucoup de choses en mémoire, et vous êtes à la merci du premier "memory lost". Bien que conscient du problème, vous ne pouvez rien faire d'autre que de prier et à la rigueur recopier à la main ou à l'aide d'une imprimante ces programmes qui vous sont si chers et que vous aurez à retaper un jour ou l'autre. Vous désirez brancher un joystick, faire un composeur téléphonique, vous connecter à un ordinateur, sauvegarder vos programmes sur disquette, vous connecter à la machine d'un copain...

Avec ce petit connecteur qui se loge sans se faire remarquer dans le rectangle de l'infrarouge, tous vos problèmes seront résolus puisque vous pourrez aussi bien sauvegarder vos programmes sur disquette que vous connecter directement sur une autre 28 par le moyen d'un câble.

### MATERIEL

miniperceuse munie d'une fraise;

superglue;

fil très fin gainé (fil de wrapping);

ciseaux;

fer à souder et étain;

résistance de 390 ohms pour les 28C et 3 900 ohms pour les 28S;

sparadrap;

connecteur 4 broches en ligne femelle si possible noir (que l'on appelle parfois embase) (dimensions: 10mm*4mmm*18mm).

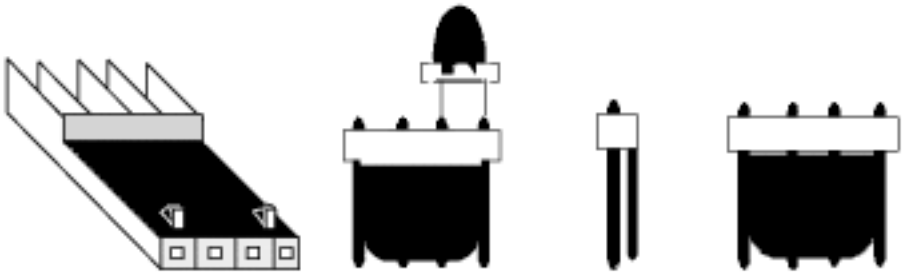


**FIGURE V-5**

**DUREE: 1h**

**1-** Ouvrez la machine (voir chapitre correspondant).

**2-** Sortez la carte électronique du boîtier.

**3-** Usinage du boîtier. A l'aide de la fraise usinez, dans le rectangle de la diode infrarouge sur le boîtier le rectangle nécessaire à la mise en place du connecteur femelle.

Ce connecteur possède sur le dessus deux petites surélévations de 1mm entre lesquelles viendra se loger l'ergot de la prise mâle qui détermine le sens de branchement. En effet la prise mâle correspondante possède à l'origine une rangée de 4 picots métalliques encadrée de deux languettes de plastique; l'une devant passer sous le connecteur femelle et l'autre au-dessus, entre les deux surélévations.

On supprimera la languette large du dessous pour ne conserver que celle du dessus, il faudra lui réserver sa place entre les deux surélévations dans la machine. Le connecteur femelle se placera dans l'usinage du couvercle, sortie des trous au ras de la sortie du rectangle de la diode infrarouge, côté des surélévations vers le fond du couvercle. Assurez vous que le connecteur rentre bien dans le trou (mais ne faites pas le trou trop large pour autant). (Photo V-6)

**4-** Dans le fond de la machine (partie qui contient le clavier après ouverture) se trouvent, à l'endroit où devrait se trouver la diode lorsque la machine est fermée, deux languettes de plastique parallèles possédant une encoche ronde pour loger la diode. Supprimez ces deux languettes.

**5-** Placez un petit morceau de sparadrap sur les tiges métalliques qui partent des ressorts des piles pour que le connecteur ne fasse pas de court-circuit. Toutefois prenez garde à ne pas en mettre trop, n'oubliez pas que le connecteur mâle possède une languette qui viendra presque jusqu'au niveau de la première tige et il ne faut pas qu'elle bute sur le sparadrap.

**6-** Connecteur vu de dos, c'est-à-dire la surélévation derrière, la partie lisse vers vous, trous vers l'avant: nous numéroterons les canaux de droite à gauche de 1 à 4.

**7-** Soudez aux broches 1, 2 et 4 des fils de 10cm de long.

**8-** Soudez la résistance à ras sur la broche 3.

**9-** Pliez la diode à 90 degrés vers la gauche dans le plan du connecteur.

**10-** Soudez un fil de 10cm à l'autre extrémité de la résistance.

**11-** Numérotez les fils de 1 à 4 avec des étiquettes ou des bandes dessinées au feutre.

**12-** Avec de la superglue, collez le connecteur à sa place, en mettant de la colle sur les côtés, sur le dessus, mais pas à l'endroit où se logera la languette de la prise mâle, soit sur le devant du dessus. Toutefois collez-la suffisamment bien pour qu'elle tienne longtemps.

**13-** Placez un bout de sparadrap sur la connection entre la résistance et le fil 3.

**14-** Dessoudez la diode infrarouge de la carte électronique.

**15-** Nettoyez l'écran et replacez la carte électronique bien à fond en faisant passer les fils à droite du gros cylindre noir.

**16-** Soudez 1, en coupant l'excédent de fil, sur la borne la plus à gauche (+) de la diode infrarouge. (Photo V-7)

**17-** Soudez 2 sur l'autre borne de la diode: la seconde (la masse).

**18-** Soudez 3 sur la piste numéro 11 des contacts au clavier.

**19-** Soudez 4 sur la 9.

**20-** Dans le fond de la machine, il y a, à la sortie du clavier, l'extrémité d'une feuille de plastique qui dépasse. Cette feuille contient 21 pistes. En comptant à partir de la gauche trouvez la neuvième puis la onzième piste.

A l'aide de ciseaux, découpez sur plus d'un demi-centimètre le plastique transparent à droite et à gauche de 9 comme pour faire une frange de 9, sans toucher 8 et 10. Ceci pour que l'excédent de soudure apporté en 9 n'empêche pas, du fait de son épaisseur, le contact de se faire entre la carte et le clavier en 10 et 8. Faites de même à droite de 11.

**21-** Passez à une autre transformation ou refermez la machine. Une fois la machine refermée vous devez avoir de gauche à droite sur le connecteur: colonne de (S7), ligne de (S7), (-) de la diode, et (+) de la diode.

Si vous voulez utilisez une imprimante HP, il suffit d'acheter le connecteur mâle correspondant au connecteur de votre machine, ce connecteur possède 2 plaques de plastique qui encadrent la rangée de picots, supprimer la plus large des deux. Retordre les pattes de votre diode de manière à ce qu'elles soient droites, et dessoudez la diode sur les picots

correspondants aux deux trous les plus à droite de votre connecteur, de manière à ce que la plus grosse partie visible dans la diode soit à gauche (c'est le moins). (voir figure V-5 page 113).

N'oubliez pas de limer le dessous de la languette côté picots pour supprimer la petite dent. Sinon, une fois le connecteur inséré, votre 28 ne voudra plus vous le rendre.

L'intérêt de (S7) est qu'on ne peut pas le simuler directement au clavier donc pas d'interférence en cours de transmission (à moins que l'on ne fasse SPC, Z et Y simultanément).

De plus, pour des raisons de compatibilité avec la 28S, on ne peut pas utiliser l'entrée infrarouge de la 28C.

Il ne vous reste plus à présent qu'à faire les câbles de connections, (voir la partie INTERFACES).

---

# Chapitre VI
# FERMETURE

**AVANT DE LIRE CE QUI SUIT, CONSULTEZ L'AVERTISSEMENT Page (9).**

L'intérêt de ce chapitre n'est autre que de rendre présentable votre 28 après tous les outrages que vous lui avez fait subir.

**MATERIEL :**

colle plastique (UHU par exemple);

superglue;

petits étaux (au minimum 2) (4 c'est parfait) (l'écartement nécessaire est de 2 à 3 cm).

**DUREE : 10min**

+ temps de séchage pour un meilleur collage : une nuit.

**1-** Assurez-vous que la carte est bien enfoncée dans le couvercle.

**2-** Vérifiez que rien n'empêche la fermeture après les transformations faites. Essayez de fermer la machine.

**3-** Nettoyez les contacts sur la feuille de plastique du clavier comme sur la carte électronique. En passant un chiffon puis en époussetant avec un pinceau.

**4-** Fermez la machine sans colle. Maintenez-la fermée avec la main, insérez les piles, allumez, et lancez l'auto-test manuel. Si tout va bien, vérification du clavier comprise, vous pouvez poursuivre. Sinon: si c'est le test clavier, nettoyez les contacts ou si vous avez placer un connecteur assurez-vous que vous avez bien séparé les pistes, sélectionnées pour l'entrée, sur la feuille de plastique du clavier, (par rapport aux autres et parallèlement à elles mêmes).

Si votre machine ne marche toujours pas, assurez-vous que vous avez bien mis les piles, que vous maintenez bien la machine fermée à fond lorsque vous l'allumez ou quand vous appuyez sur une touche, essayez de court-circuiter les bornes du gros condensateur noir à l'intérieur (gros cylindre noir) pour tout remettre à zéro dans la mémoire et vérifiez toutes les étapes des transformations faites.

**5-** Si tout va bien placez de la colle plastique sur le fond de la machine, sur les bords, le fond et les picots qui vont rentrer en contact avec le couvercle mais sans que la colle ne touche la partie électronique. Mettre de la colle partout, sauf sur les bords à droite et à gauche des contacs du clavier sur 1 cm, car vous allez y mettre deux pointes de superglue.

**ATTENTION !** un abus de superglue est néfaste pour la machine à cause des oxydations engendrées par les émanations.

**6-** Maintenez la machine fermée et serrée le mieux possible pendant que vous placez des petits étaux ou des clips aux lieux stratégiques suivants , par ordre d'importance:

- En bas de l'écran à gauche (appuyez-vous sur le bords et non sur l'écran).
- En bas de l'écran à droite.
- En haut à gauche au-dessus de l'infrarouge ou du connecteur.
- En haut à droite au-dessus du boîtier des piles. Il est recommandé de placer un morceau de papier entre les surfaces d'appui des étaux et la machine.

**ATTENTION !** Ne serrez pas trop fort, surtout si vous avez un module mémoire, ne serrez **JAMAIS** au-dessous de l'endroit où vous avez fraisé, car dans ce cas vous serreriez directement le module contre la carte électronique et ce serait un massacre. Serrez juste ce qu'il faut pour que le couvercle soit en place dans le fond. Si vous avez mis un module, relisez la note au sujet de la fermeture dans le chapitre ajout de mémoire.

**6-** Maintenez la machine fermée et serrée le mieux possible

**7-** Attendez une nuit.

**8-** Défaites les étaux.

**9-** Mettez les piles et testez votre machine.

**10-** Placez le cache gris du clavier de telle sorte qu'il masque presque toute la fente du haut du clavier. Passez votre doigt sur tout le cache pour bien le faire adhérer.

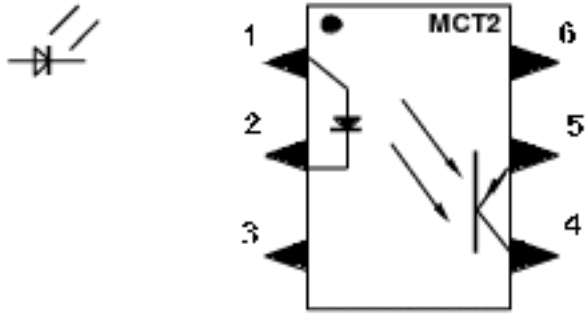
*Nota :* pour réouvrir ultérieurement, il suffira de passer un cutter là où vous avez mis de la colle.

# Chapitre VII
# INTERFACES D'ENTREES / SORTIES

Nous utiliserons en entrée la simulation de la touche (s7) et en sortie, la diode infrarouge, ceci aussi bien pour les 28C que pour les 28S pour des raisons de compatibilité.

Pour protéger votre HP contre des erreurs extérieures de manipulation, nous utiliserons en entrée comme en sortie des photocoupleurs MCT2 de manière à isoler entièrement l'électronique de la 28 de l'électronique de l'extérieur. Le seul lien entre les deux sera photonique (par la lumière) et non directement électrique.

Voici le câblage du MCT2 ( le 1 est repéré par le point ):




La numérotation des bornes du connecteur de votre HP: se fait de gauche à droite de 1 à 4, lorsque votre HP se présente normalement ouverte devant vous. Electroniquement parlant:

1= entrée clavier (colonne (-)), connectée au clavier 9.

2= entrée clavier (ligne (+)), connectée au clavier 11 avec la résistance.

3= sortie infrarouge (-), connectée à la borne infrarouge 2.

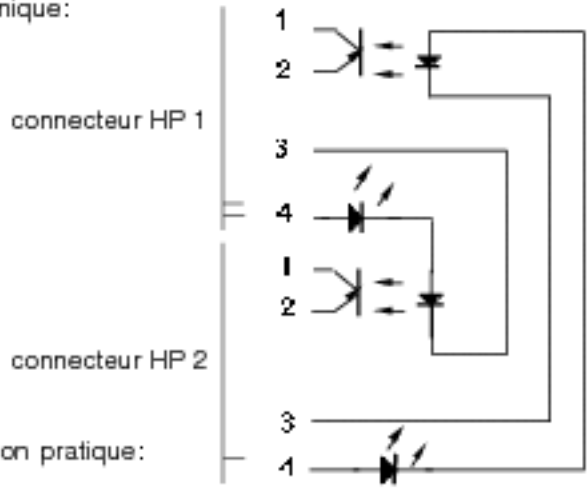
4= sortie infrarouge (+), connectée à la borne infrarouge 1.

Nous rappellons que les lignes du clavier sont des (+), malgré la faiblesse du courant qui les traverse, les colonnes sont des (-).

## 1. CABLE HP <-> HP

Ceci permet en utilisant les programmes IN et OUT de recevoir et d'émettre des chaînes de caractères entre deux HP28.

Schéma électronique:




Voici la réalisation pratique:

**MATERIEL :**

1 mètre de câble à 4 fils;
2 photocoupleurs MCT2;
2 diodes lumineuses rouges (led) les plus petites possibles (3mm);
du fil (le wrapping gainé par exemple);
2 micro connecteurs mâles (embases noirs) 4 broches en ligne;
2 plaques époxie quadrillées trouées de 4 trous sur 6;
fer à souder et étain.

**DUREE : 20min**

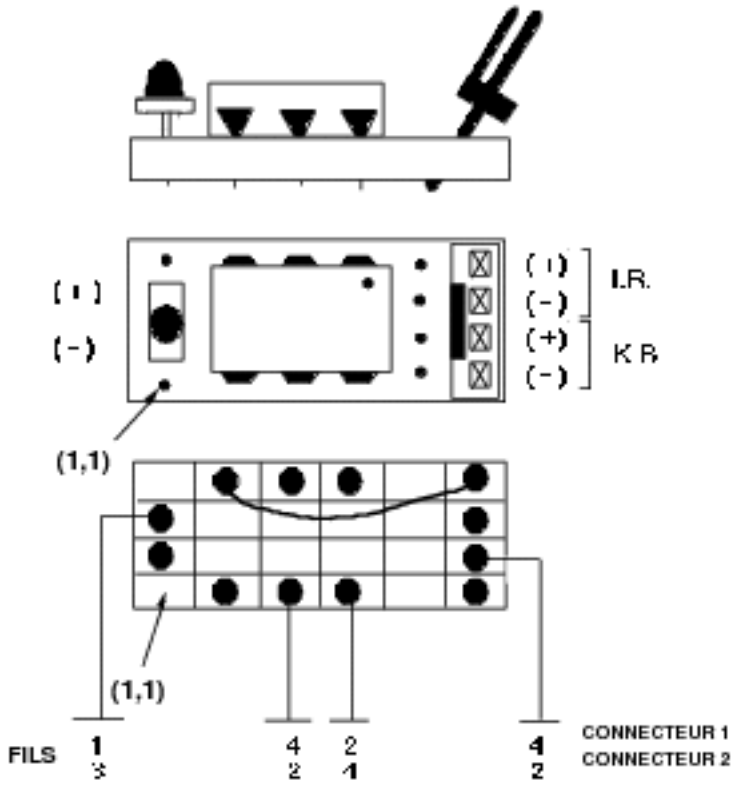
1. Nous orienterons la plaque à souder en époxie de la façon suivante: soudures derrière, grande longueur horizontale. On note les coordonnée d'un point (x,y) x sur l'horizontale y, sur la verticale (1,1) est le coin en bas à gauche.
2. Plaque 1, soudez le MCT2 numéro 1 en (4,4), 2 en (3,4) ...
3. La diode en (1,2) et (1,3) le (-) en (1,2). (-) = patte correspondant à la partie métallique la plus grosse dans la diode.
4. Le connecteur mâle possède deux plaques, l'une étant plus petite que l'autre. Sciez à la base la plaque la plus large du connecteur 4 broches. Laissez la plus petite. Limez la petite encoche sous cette plaque pour faciliter l'introduction dans le connecteur de la machine.

5. Soudez l'embase (le connecteur) sur la plaque en (6,1) à (6,4) en laissant le côté qui n'a pas été scié du côté des composants. Arrangez-vous pour que le connecteur ne soit pas parfaitement vertical mais plutôt un peu incliné vers l'extérieur de la plaque.

6. Choisissez 4 numéros de 1 à 4 pour les couleurs de chacun des fils de votre câble.

7. Retournez votre plaque époxie de manière à ce qu'elle soit toujours horizontale et le connecteur sur votre droite, on place le nouveau (1,1) en bas à gauche.

8. Raccordez:
   - la sortie (+) de l'IR au (+) de la diode lumineuse soit (1,2) à (6,1);
   - le 5 du MCT2 au 2 du connecteur soit (3,4) à (6,3);
   - le 4 du MCT2 au 1 du connecteur soit (2,4) à (6,4);
   - le 1 du cable au (-) de la diode soit au (1,3);
   - le 2 du cable au (-) de l'IR soit au (6,2);
   - le 3 du cable au 1 du MCT2 soit au (4,1);
   - le 4 du cable au 2 du MCT2 soit au (3,1).



**9.** Répétez toutes ces opérations pour la seconde plaque, sans connecter le câble.

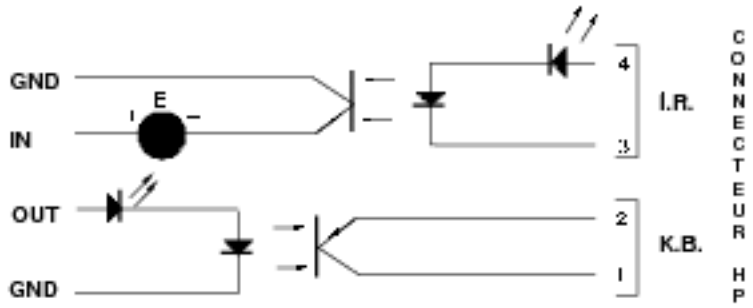
**10.** Connectez le câble sur la plaque 2 avec la même orientation vue de dos:

- le 1 au (4,1);
- le 2 au (3,1);
- le 3 au (1,3);
- le 4 au (6,2).

**11.** Pour évitez que le câble ne se dessoude au cours de manipulations prolongées, vous pouvez le fixer avec du sparadrap d'électricien (de préférence noir (pour l'esthétique)) mais ne cachez pas les diodes lumineuses et ne remplissez pas la rangée entre le MCT2 et le connecteur; cet emplacement est réservé pour l'enfilage sur la machine. Ce câble est compatible avec les programmes IN et OUT.

## 2. HP <-> RS232

La théorie est identique à la précédente. Voici le schéma de principe:




où E est une alimentation continue émanant d'une pile de 9V ou d'un transformateur 220->9V continu.

**MATERIEL :**

2 photocoupleurs MCT2;

2 diodes lumineuses LED une rouge et une verte (les plus petites possibles (3mm));

une prise pour votre alimentation 9V;

une prise DB25 pour la RS232;

1 plaque époxie quadrillée à souder de 4 trous par 9;
1 connecteur embase mâle 4 broches en ligne comme précédemment;
1 câble 3 fils de 1 mètre;
fer à souder plus étain.

**DUREE : 20min**

**1-** Usinez le connecteur mâle voir étape 4 de HP<->HP.

**2-** Plaque orientée soudures derrière, grande longueur horizontale, (1,1) en bas à gauche.

**3-** Soudez la diode verte en (1,1) (-) et en (1,2) (+). Le (-) est la patte qui correspond à la partie la plus grosse dans la diode. Soudez de même la diode rouge en (1,3) (-) et en (1,4) (+).

**4-** Soudez le premier MCT2 le 1 en (2,1) le 2 en (3,1)...

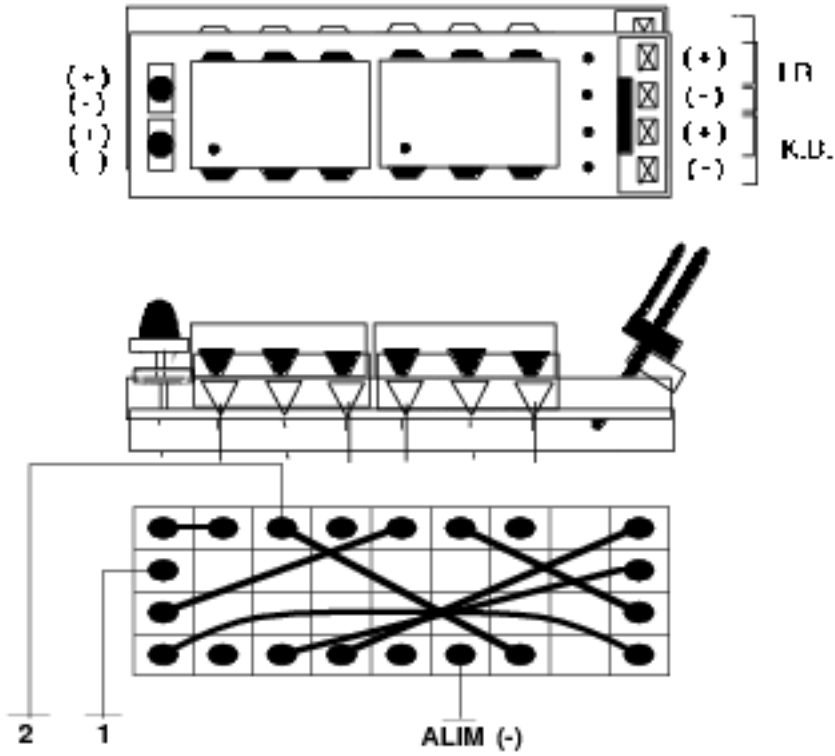
**5-** Soudez le second MCT2 le 1 en (5,1) le 2 en (6,1)...

**6-** Soudez l'embase verticalement comme pour le câble HP<->HP

**7-** Retournez le connecteur. Grande longueur toujours horizontale, nouvelle numérotation : (1,1) en bas à gauche. Numérotez les fils du câble de 1 à 4 en fonction de la couleur.

**8-** Connectez:

- (1,3) au fil 1 ((+) de la diode au OUT de la RS232);
- (1,4) au (2,4) ((-) de la diode au + du MCT2);
- (3,4) au fil 2 ((-) du MCT2 à la masse de la RS232: GND);
- (3,1) au (9,3) ((+) clavier au collecteur du MCT2);
- (4,1) au (9,4) ((-) clavier à l'émeteur du MCT2);
- Le (-) de votre prise alimentation au fil 3 ((-) alim au IN);
- Le (+) de votre prise alimentation au (6,1) ((+) alim au collecteur du MCT2);
- (7,1) au (3,4) (émeteur du MCT2 au GND);
- (1,1) au (9,1) ((+) IR au (+) diode rouge);
- (1,2) au (5,4) ((-) diode rouge au (+) MCT2);
- (6,4) au (9,2) ((-) MCT2 au (-) IR).




**9-** Fixer le câble et les fils de la prise alimentation au montage avec du sparadrap, sans recouvrir les colonnes 8 et 9.

**10-** Repérez sur le manuel de votre ordinateur les connections IN OUT et GND, trouvez leur correspondant sur votre DB25 et faites les raccords suivants :

- soudez le fil 1 au OUT;
- soudez le fil 2 au GND;
- soudez le fil 3 au IN;
- refermez la prise.

L'opération est terminée. Lorsque vous mettez la sortie RS232 au niveau haut ou 1, la diode verte doit s'allumer et simuler l'appui de la touche (s7). Lorsque vous émettez : la diode rouge s'allume et le IN passe au niveau haut dans votre ordinateur : les programmes IN et OUT cités dans la partie soft sont toujours valables du côté de votre HP. Mais la programmation de votre ordinateur est spécifique à celui-ci.

Le protocole de transmission choisi est le suivant:

on émet la longueur sur 16 quartets;
on attend une réponse simple de l'autre;
on émet la chaîne quartet par quartet.

### 3. câble HP<->APPLE II prise JOYSTICK

*(Pour Apple II+, IIe ou IIGS).*

L'avantage de cette méthode sur la RS232 est qu'elle ne nécessite pas d'alimentation externe et que la programmation est plus facile.

La seule différence avec le cas précédent, est l'utilisation d'une prise 16 broches à sertir pour apple joystick et non une DB25.

Les connections sont:

AN0 au lieu de OUT, (fil 1);

GND en (3,4), (fil 2) on ne connecte pas (3,4) à (7,1);

+5 au lieu de ALIM+ (fil 4);

PB0 en (7,1) (fil 3) et on place une résistance de l'ordre de 500 Ohms au lieu de la connection entre (3,4) et (7,1).

On allume alors la sortie par : POKE -16295,0;
et on éteint par : POKE -16296,0.

Voici les programmes de réception, sauvegarde, et émission sur APPLE II et sur HP.

### 1] PROGRAMMES D'EMISSION SUR HP

#### EMIT

Voir annexe programme (page 219); avec comme coefficient de l'ordre de c=00010 pour la longueur d'un bit 1 (V2) et de même pour l'attente entre bit (V3): à régler en décroissant à partir de C=01000 jusqu'à ce que la réception sur l'Apple ne soit plus bonne. Prendre ensuite la plus petite valeur correcte, avec un peu de marge.

#### DUK

Ce programme en LM fait un DO UNTIL KEY END DROP

**HP28C**

```
69C20
94000
8F 2EE40
808F
34 EF000
801 803
8AA
0F
34 EF000
```

**HP28S**

```
69C20
85000
8F 18050
1F5300C30F15D0
34 EF000
8F 1DA10
8AA
FE
34 EF000
```

```
801  803                    8F  1DA10
8AE                         8AE
0F                          FE
8080                        30015D0
8F  91F40                   8F  8B050
142                         142
164                         164
808C                        808C
```

**SEND**

2: "chaîne"
1: "nom sous forme de chaine"
et envoi la compilation de ceci qui est :

**"abc NOM**
**CHAINE"**

"a" est le CHR des centaines et des milliers de la longueur totale de cette grande chaîne; "b" est le CHR des unités et dizaines de cette longueur; "c" est le CHR de la longueur du nom. Donc on estime que la grande longueur n'éxcédera pas 9999, ni celle du nom 255.

```
« DUP  SIZE  CHR  SWAP  +  10  CHR
        +  SWAP  +  DUP  SIZE  2  +  100  /  DUP
        IP  CHR  SWAP  FP  100  *  CHR  +  SWAP  +
        DUP  SIZE  #0  +  EMIT  DROP  DUK  1  WAIT
   EMIT  DROP
»
```

**VAROUT**

1: 'NOM'
Il rappelle le contenu du nom, le transforme en chaîne et l'envoie.

```
« DUP  RCL  ->STR  SWAP  ->STR  2  OVER  SIZE  1  -  SUB
SEND »
```

**LOUT**

1: { liste de noms }

Il envoie une liste de variable. Après chaque réception, répondez par O (pour oui) à la question de l'Apple "ENCORE ? (O/N)", puis appuyezsur une touche de la HP.

```
« LIST->  1  SWAP  START  CLLCD  DUP  1  DISP  VAROUT
   "PRESS  A  KEY..."  3  DISP  DUK  1  WAIT  NEXT  CLMF
»
```

## 2] PROGRAMME DE RECEPTION SUR APPLE II

( Le programme en langage machine de l'apple "REC.." est nécessaire).

### RECAPPLE

```
10  PRINT CHR$(4)"BLOAD REC.."
15  GOTO 1000

20  POKE 3*256+16*S+2,4
25  POKE 771,15: POKE 772,0
30  CALL 768
40  L=0: W=1
50  FOR X=0 TO 4
60  L=L+W*PEEK(AD+X):W=W*16
70  NEXT
75  PRINT "SIZE= ";L
80  LE=2*L
85  POKE 771,LE-256*INT(LE/256)
90  POKE 772,INT(LE/256)
95  POKE -16295,0: POKE -16296,0
100 CALL 768
110 RETURN

500 HOME: PRINT "28 LINK": PRINT "(C) S.LALANDE & P.COURBIS"
505 PRINT: PRINT: PRINT "PRET A RECEVOIR...."
510 AD=32*256: GOSUB 20
515 N$= ""
520 A=PEEK(AD+4)+16*PEEK(AD+5)
530 FOR X=0 TO A-1
540 N$= N$+ B$( PEEK(AD+6+2*X) + 16 * PEEK(AD+7+2*X) )
550 NEXT
560 PRINT "NAME= ";N$
600 PRINT "SAUVEGARDE"
610 D$=CHR$(4): PRINT D$;"BSAVE";N$;",A$2000,L"; STR$(LE)
680 PRINT "OBJET="
690 AD= AD + 2 * A + 8
700 FOR X=0 TO L-A-5
710 PRINT A$( PEEK(AD+2*X) + 16 * PEEK(AD+2*X+1) );
720 NEXT
800 PRINT: PRINT
810 PRINT "ENCORE ? (O/N) ";: GET A$
820 IF A$<>"N" THEN GOTO 500
900 END

1000 DIM A$(256): RESTORE
1010 FOR I=0 TO 255
1020 IF I=10 THEN A$(I)= CHR$(13): NEXT
1030 IF I<32 THEN A$(I)= ".": NEXT
1040 IF I<96 THEN A$(I)= CHR$(I): NEXT
1050 IF I=96 OR I>122 AND I < 148 THEN READ C$: A$(I)=C$: NEXT
1060 IF I<123 THEN A$(I)= CHR$(I-32): NEXT
1070 IF I>148 THEN A$(I)= ".": NEXT
1100 DIM B$(256)
1110 FOR I=0 TO 255
1120 IF I>32 AND I<96 THEN B$(I)= CHR$(I): NEXT
1130 IF I>96 AND I<123 THEN B$(I)= CHR$(I-32): NEXT
1140 B$(I)= " ": NEXT
1150 GOTO 500
1200 DATA   "'","(","!",")","-","#"," ","/","^","V","S","E","D",
            "#","&","=<",">=","#","A","->","<- ","U","LF","'",
            "<<",">>"
```

Programme assembleur de réception sur Apple II:

**REC..**

```
Adresse Codes        Mnémoniques    Commentaires
0300:   4C 2D 03     JMP   $032D    aller au début réel du programme

0303:   00           BRK            nbre de quartets à recevoir
0304:   00           BRK            (réglé par le prgm basic)

0305:   8D FF FF     STA   $FFFF    sauvegarde d'un quartet
0308:   18           CLC            incrémentation de l'adresse de
                                     sauvegarde
0309:   AD 06 03     LDA   $0306
030C:   69 01        ADC   #$01
030E:   8D 06 03     STA   $0306
0311:   AD 07 03     LDA   $0307
0314:   69 00        ADC   #$00
0316:   8D 07 03     STA   $0307
0319:   38           SEC
031A:   AD 03 03     LDA   $0303    décrémentation du nbre de quartets
031D:   E9 01        SBC   #$01
031F:   8D 03 03     STA   $0303
0322:   AD 04 03     LDA   $0304
0325:   E9 00        SBC   #$00
0327:   8D 04 03     STA   $0304
032A:   B0 1C        BCS   $0348    si pas fini -> 348
032C:   60           RTS            sinon: retour au basic

032D:   A9 00        LDA   #$00     initialisation de l'adresse de
                                     sauvegarde
032F:   8D 06 03     STA   $0306    (mise à $2000)
0332:   A9 20        LDA   #$20
0334:   8D 07 03     STA   $0307
0337:   38           SEC
0338:   AD 03 03     LDA   $0303    décrémentation du nbre de quartets à
                                     recevoir
033B:   E9 01        SBC   #$01     (ceci est rendu nécessaire par le
                                    type de test de fin de réception)
033D:   8D 03 03     STA   $0303
0340:   AD 04 03     LDA   $0304
0343:   E9 00        SBC   #$00
0345:   8D 04 03     STA   $0304
0348:   20 4F 03     JSR   $034F    reçoit un quartet
034B:   8A           TXA
034C:   4C 05 03     JMP   $0305    on le sauve

034F:   A2 00        LDX   #$00     réception d'un quartet
0351:   A0 04        LDY   #$04     nbre de bits à recevoir
0353:   8A           TXA
0354:   0A           ASL
0355:   AA           TAX
0356:   20 62 03     JSR   $0362    carry=bit reçu
0359:   8A           TXA
```

```
035A:   69 00       ADC   #$00      prise en compte de ce bit
035C:   AA          TAX
035D:   88          DEY
035E:   D0 F3       BNE   $0353     si il reste des bits à recevoir ->
353
0360:   60          RTS

0361:   00          BRK             longueur reçue
0362:   20 6C 03    JSR $03C6       réception d'un signal
0365:   AD 61 03    LDA $0361       A=longueur signal
0368:   18          CLC             comparaison avec un offset d'où:
0369:   69 F0       ADC   #$F0 (1)  - carry=0 pour un signal court
036B:   60          RTS             - carry=1 sinon

036C:   A9 00       LDA   #$00      initialisation de la longueur reçue
036E:   8D 61 03    STA   $0361
0371:   AD 61 C0    LDA   $C061     attente début signal
0374:   C9 80       CMP   #$80
0376:   30 F9       BMI   $0371
0378:   EE 61 03    INC   $0361     mesure longueur
037B:   F0 08       BEQ   $0385     si longueur > 255 on va en 385
037D:   AD 61 C0    LDA   $C061
0380:   C9 80       CMP   #$80
0382:   10 F4       BPL   $0378
0384:   60          RTS

0385:   A9 FF       LDA   #$FF      la longueur est fixée à 255
0387:   8D 61 03    STA   $0361
038A:   AD 61 C0    LDA   $C061     attente fin signal
038D:   C9 80       CMP   #$80
038F:   10 F9       BPL   $038A
0391:   60          RTS
```

(1) Cette valeur (V) permet de déterminer si le signal correspond à un 0 ou à un 1. Elle dépend de la vitesse de la ZX et doit être telle que (V)+longueur du signal court ≤ 256 et (V)+longueur du signal long > 256.

**Entrée du programme sur Apple:**

- passer en mode moniteur par CALL-151
- taper `300:4C 2D 03` [RETURN]
  `303:00 00` [RETURN]

  etc...
- sortir du moniteur par `3D0G` [RETURN]
- enfin, sauver par `BSAVE REC..,A$300,L$92`

## 3] PROGRAMME D'EMISSION SUR APPLE

(B1 est à régler. C'est la longueur d'émission d'un bit long, si la sensibilité de la machine est suffisamment bonne pour détecter la différence entre les longs et les courts, vous pouvez remplacer la boucle en 230 par une commande plus courte en exécution: E=0 (par exemple)).

### EMIAPPLE

```
5  B1=1
10 HOME: PRINT "28 LINK": print"(C) S.LALANDE & P.COURBIS"
15 PRINT "EMSSION HP"
20 PRINT: PRINT
25 PRINT: PRINT "(* POUR CATALOG)"
30 INPUT "NOM = ";N$
35 IF N$="*" THEN PRINT CHR$(4)"CATALOG": GOTO 25
40 PRINT CHR$(4)"BLOAD";N$
45 AD= 32 * 256
50 L=PEEK(AD)*100+PEEK(AD+1)*1600+PEEK(AD+2)+PEEK(AD+3)*16
55 PRINT "LONGUEUR TOTALE = ";L
60 LE=2*L
65 PRINT "DEMAREZ LE PROGRAMME 'IN' SUR L'HP..."
70 PRINT "PRESS A KEY TO START";: GET A$
75 L=4: GOSUB 100
80 WAIT -16287,128
85 L=LE: GOSUB 100
90 GOTO 500

100 FOR X=0 TO L/2-1
110 T= PEEK(AD+2*X): GOSUB 200
120 T= PEEK(AD+2*X+1): GOSUB 200
130 NEXT
140 RETURN

200 FOR I=3 TO 0 STEP -1
210 W= INT( T / 2^I ): T= T - W * 2^I
220 IF W=0 THEN POKE -16295,0: POKE -16296,0: GOTO 240
230 POKE -16295,0: FOR E=1 TO B1: NEXT: POKE -16296,0
240 NEXT
250 RETURN

500 PRINT: PRINT "THE END!"
510 PRINT "ENCORE ? (O/N) ";: GET A$
520 IF A$<>"N" THEN GOTO 10
530 END
```

## 4] PROGRAMMES DE RECEPTION SUR HP

### RECEP

Voir annexes. Le coefficient d'attente (V4) est de l'ordre de C=00100, à règler en décroissant à partir de 180 et en fixant B1 dans EMIAPPLE à 400. Une fois V4 réglé, faites décroitre B1.

### IR1

Celui qui est avec le programme RECEP dans l'annexe, avec un délai de C=00004.

### CREAC

Idem voir RECEP en annexe.

### IN

A déclencher avant l'émission apple.

```
« " " " " + RECEP DUP 1 1 SUB NUM 100 * SWAP
  2 2 SUB NUM + CREAC IR1 RECEP 3 9999 SUB DUP DUP
   1 1 SUB NUM 1 + -> A
 « A 2 SWAP SUB "'" SWAP OVER + +
   STR-> SWAP A 1 + 9999 SUB
   IF DUP 1 1 SUB "«" ==
   THEN STR->
   END
   SWAP
   »
»
```

## 4. PROTOCOLE DE TRANSMISSION

Les "1" sont des signaux longs les "0" sont des courts.

**EMISSION HP**

Il doit être déclenché après le programme de réception sur HP.

**1-** Par le procédé suivant de 2 à 6 on émet le nom sous forme de chaîne de caractères puis l'objet.

C'est le programme IN exactement:

**2-** On calcule la longueur de la chaîne par SIZE.

**3-** On transforme en entier binaire hexadécimal par #0 +.

**4-** Que l'on envoie par le programme EMIT qui fait :

.Lecture de la longueur de l'objet (si c'est un entier binaire c'est F en hexa).

.Lecture du premier quartet, émission de ce quartet.

.Décrémentation de la longueur.

.Lecture du deuxième ....

...

.Jusqu'à ce que la longueur soit nulle.

**5-** Attente d'une réponse brève par un DO UNTIL KEY END DROP ou mieux le programme WUS ou DUK.

**6-** Emission de l'objet par EMIT.

**RECEPTION HP ou ORDINATEUR**

**1-** Par le procédé suivant de 2 à 6 on reçoit le nom puis l'objet.

**2-** On prépare la place pour recevoir 16 quartets.

**3-** On reçoit les 16 quartets bit par bit selon la méthode suivante:

.On attend de recevoir quelque chose.

.On incrémente une boucle jusqu'à ce que la réception cesse.

.On compare avec la valeur fixée pour un 1, si elle est supérieure c'est un 1, si elle est inférieur c'est un 0.

**4-** On prépare la place selon la longueur que l'on vient de recevoir pour recevoir la suite.

**5-** On émet un coup bref.

**6-** On reçoit l'objet.

**7-** On sauvegarde l'objet sous le format désiré sous le nom reçu.

**EMISSION HP ou ORDINATEUR**

**1-** On demande le nom; on le recherche; on stocke son contenu dans la mémoire, on émet par le procédé suivant le nom puis l'objet.

**2-** On émet la longueur de l'objet 16 quartets.

**3-** On attend le signal pour continuer.

**4-** On émet l'objet.

**RECEPTION HP**

**1-** Par le procédé suivant on reçoit le nom sous forme de chaine de caractères puis l'objet en répétant deux fois les étapes de 2 à 6. C'est le programme OUT exactement:

**2-** On crée un entier binaire que l'on remplira par la réception.

**3-** On reçoit la longueur que l'on écrit dans l'entier binaire.

**4-** On crée la chaîne de la longueur requise par CREAC.

**5-** On émet un coup bref.

**6-** On reçoit la chaîne entière.

**7-** On sauvegarde la chaîne sous le nom désiré.

---

# CONCLUSION

Ce qui a été expliqué jusqu'à présent ne constitue que des bases qui peuvent servir d'échafaudage à tout un ensemble d'idées:

Joystick.
Composeur téléphonique.
Commande de plusieurs moteurs.
Un robot dont le cerveau serait votre HP.
Une table traçante.
Donner un autre clavier à votre machine et un autre look.
Rajouter encore et toujours de la mémoire.
...

## JOYSTICK

Les joysticks modernes ne sont plus à potentiomètre mais à interrupteur; en effet, il y a quatre interrupteurs pour les quatre directions et un autre pour le bouton.

Il suffit donc de se faire un connecteur sur la machine possédant non pas une rangée de 4 broches mais deux rangées. On connecte toujours sur les deux premières de droite l'IR et sur les 6 autres les bornes 4 10 13 15, 18 au clavier et 21 par l'intermédiaire d'une résistance adéquate au clavier. (Voir la partie connecteur). Ainsi vous simulerez les 4 touches du curseur plus INS il suffit alors de raccorder ces broches aux interrupteurs du joystick.

Il existe de tout petits joysticks avec des tiges de 2cm.

Vous pourrez alors promener le curseur en mode EDIT dans les GRAPHICS et vous pourrez faire des jeux en testant le clavier de la manière décrite dans le soft et dans les descriptions des entrées sorties.

## COMPOSEUR TELEPHONIQUE

Branchez sur la sortie infrarouge un photocoupleur MCT2 de la manière classique. Branchez sur la sortie du photocoupleur une alimentation de 5 à 9 volts en série avec un micro-relais. Branchez enfin, une résistance de 150 Ohms entre un des fils du téléphone et la sortie du micro-relais, l'autre sortie du relais se branche sur le second fil du téléphone. Lorsque vous envoyez un signal sur l'IR, un clic se produit sur le fil du téléphone.

Il ne reste plus qu'à simuler par programme les séries de tic, tic, tic.... qui correspondent aux chiffres.

Pour commander plusieurs moteurs on s'inspire de la transmission par "longs" et "courts" déjà abondamment utilisée. On commande un moteur avec un photo-coupleur et un relais, en analysant les longueurs des signaux grâce à des circuits logiques et des delays. Le plus court commande le premier moteur, ... le plus long, le dernier.

### UN ROBOT

On peut diriger plusieurs moteurs et on a autant d'entrées voulues par le clavier. A vous de jouer.

### UNE TABLE TRACANTE

Deux moteurs suffisent, et, pour perfectionner on peut utiliser les entrées afin de signaler la position du crayon au programme.

### UN AUTRE LOOK

Retirez la carte électronique, branchez-la sur un autre clavier, une autre alimentation, dans un autre boîtier (celui d'une autre calculatrice par exemple...

En se donnant de la peine tout est possible avec un peu d'effort. Mais ce n'est jamais facile. Il est toujours formateur d'innover.

Bonne chance !!

# ANNEXES

# Annexe I
# LE LANGAGE MACHINE

Si vous savez déjà ce que sont assembleur et langage machine vous pouvez passer directement à l'annexe suivante... Sinon il ne vous reste qu'à lire ce qui suit...

L'exemple le plus classique utilisé pour comparer le langage évolué d'une machine avec son assembleur est celui de l'électricien:

Imaginons que Monsieur Dupont ait besoin d'installer de nouvelles prises électriques chez lui. Deux solutions s'offrent à lui:

- Appeler un électricien et lui dire: "posez une prise ici et là..."
- Se rendre chez son quincailler, acheter du fil et des prises, rentrer chez lui, préparer son matériel, poser prises et fils, vérifier son installation...

On peut dire que dans le premier cas Monsieur Dupont a utilisé un langage évolué en donnant un ordre qui recouvre de nombreuses opérations élémentaires (chercher du fil, chercher une prise...) alors que dans le second cas il a, en quelque sorte, utilisé un langage assembleur en réalisant lui-même et directement les différentes opérations élémentaires...

L'exemple proposé correspond très bien au problème. En effet: le langage évolué est:

- simple à utiliser;
- lent (comme l'électricien à venir...);
- ne "colle" pas toujours exactement aux désirs de l'utilisateur (!!!);
- certaines choses sont impossibles à réaliser (essayer donc de demander à votre électricien de venir changer une lampe [comme ça, pour voir...]).

Quant à l'assembleur:

- il est complexe à utiliser (on doit utiliser de nombreuses opérations élémentaires);
- il est rapide;
- on fait exactement ce que l'on veut;
- on peut faire des choses impossibles à réaliser à l'aide du langage évolué.

En fait les deux langages sont complémentaires: les deux sont très utiles, les deux sont agréables à utiliser , mais ils ne permettent pas les mêmes choses...

L'assembleur a été évoqué. Précisons exactement sa nature, ainsi que celle du langage machine qui lui est associé:

Un programme assembleur est la liste de noms des opérations élémentaires à effectuer pour réaliser une tâche donnée.

Cependant la machine n'est pas à même de comprendre ces noms: il faut les transformer en une suite de chiffres (de codes) qui lui soient compréhensibles.

Ce passage de l'assembleur au langage machine s'appelle l'assemblage (l'opération inverse se nomme le désassemblage). Comme il est quasiment impossible de comprendre un programme en langage machine, on écrit d'abord le programme assembleur correspondant. Ensuite, on effectue l'assemblage pour pouvoir faire exécuter le programme par la machine.

*Il faut savoir que :*

Dans le cas du HP28 l'assemblage se fera manuellement (un assembleur existe sur IBM PC et INTEGRAL PC [HP]. Il a été écrit par Mrs Pierre DAVID et Janick TAILLANDIER et est disponible auprès du club PPC PARIS, BP 604, 75028 PARIS CEDEX 01, FRANCE). En effet un programme d'assemblage est très long, et serait trop lent en exécution sur la HP28...

Un programme de désassemblage est proposé en annexe (dans la bibliothèque de programmes) et "tourne" sur tout HP28c ayant une mémoire étendue ainsi que sur tout HP28s...

L'annexe suivante présente les caractéristiques du microprocesseur du HP28 ainsi que les caractéristiques de son langage-machine (la liste des codes se trouve dans l'annexe suivante).

---

# Annexe II
# LE MICROPROCESSEUR SATURN

Le microprocesseur du HP-28 est un processeur SATURN 4 bits. Il s'agit sensiblement du même microprocesseur que celui du HP71.

Le SATURN possède un jeu de 19 "registres" ("registre" est le nom d'une mémoire du microprocesseur. Un "registre" ne contient que des entiers positifs)

---

**Tout d'abord les deux registres d'entrée / sortie:**

**OUTPOUT** (12 bits) :

Il permet d'envoyer du courant sur un ou plusieurs des 12 fils du clavier et du buzzer. Ce registre ne peut-être qu'écrit.

**INPUT** (16 bits) :

Il permet de lire l'état des 16 entrées.

Ces deux "registres" sont utilisés pour l'émission d'un BEEP (écriture dans OUTPUT) ainsi que pour l'échantillonage du clavier (on envoie du courant sur une ligne de touche et on regarde si ce courant ressort sur une colonne de touches ce qui permet de savoir si le bouton, situé à l'intersection ligne/colonne, est enfoncé ou non [s'il est enfoncé, le courant passe...]).

---

**Les registres-drapeaux:**

**CARRY** (1 bit) :

C'est le bit de retenue; lorsqu'une opération donne lieu à une retenue, (soustraction, addition) ce drapeau est armé: il peut ensuite être utilisé lors d'un test.

**HST** (hardware status) (4 bits) :

C'est un registre de 4 drapeaux (MP module pulled, SR service request, SB sticky bit, XM external module missing). Seul SB semble être encore utilisé sur le HP28: comme indicateur de débordement pour certaines opérations (divisions et mutiplications par 2).

**STATUS** (16 bits) :

Il s'agit de drapeaux utilisés par la machine. En fait, seulement les drapeaux 12 à 15 sont utilisés: les drapeaux 0 à 11 sont à disposition de l'utilisateur pour la réalisation de programmes). Dans la liste des codes 'd' représente le numéro de drapeau de ST ($0 \leq d \leq Fh$).

---

**Les pointeurs:**

**P** (4 bits) :

Il est utilisé pour les champs (voir paragraphe plus loin).

**D0 , D1** (20 bits chacun) :

Ils sont utilisés pour la lecture et l'écriture en mémoire.

**PC** (program counter) (20 bits) :

Il contient l'adresse de l'instruction en cours d'utilisation.

---

**Les registres de sauvegarde:**

**RSTK** (return stack) (8 étages de 20 bits chacun) :

Il s'agit d'une pile de 8 étages utilisée pour la sauvegarde d'adresses. Cette pile se comporte exactement comme la pile RPL du HP28 avec toutefois une différence : même vide, elle contient des 00000. Elle sert à la sauvegarde d'informations et en particulier à la sauvegarde de l'adresse de retour lors de l'appel à une sous-routine.

**R0 R1 R2 R3 et R4** (64 bits chacun) :

Ils sont utilisés pour la sauvegarde des registres de calcul.

---

**Les registres de calcul:**

**A B C et D** (64 bits chacun) :

Ce sont les registres de travail. A et C sont spécialement dédiés aux opérations de lecture et d'écriture mémoire (ils sont alors utilisés avec D0 et D1).

Certain de ces registres sont utilisés en permanence par le HP28: si on désire les utiliser dans un programme en langage machine, il conviendra de sauvegarder leur contenu avant tout... C'est le cas pour:

- **D0** qui est le pointeur RPL: ce registre contient l'adresse de l'instruction RPL en cours d'exécution;
- **D1** qui pointe la fin de pile RPL;
- **B** qui pointe le haut de la RETURN STACK RPL;
- **D** qui indique la place mémoire encore libre; semble être lui aussi utilisé (par les interruptions ???).

---

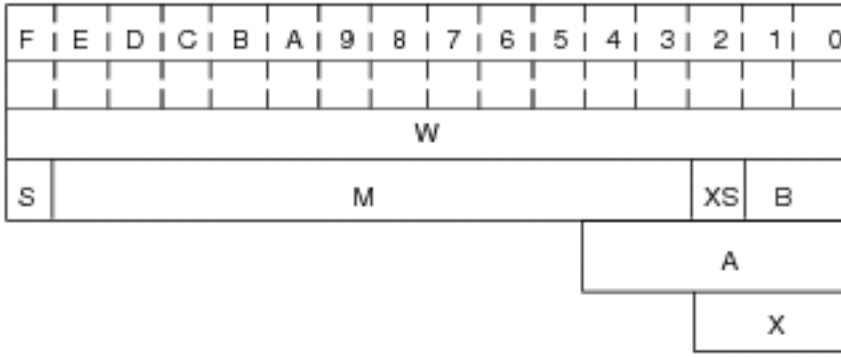
**Les champs:**

**Les registres A B C et D** sont très longs (64 bits) :

Ils sont divisés en zones plus petites appelées "champs". Ces champs, si ils ne se recouvrent pas, peuvent être utilisés indépendamment les uns des autres. Ceci permet de conduire simultanément des calculs différents avec peu de registres occupés.

Voici un tableau des champs:

N° quartet:




Ainsi le champ M représente-t-il les quartets E à 3, A les quartets 4 à 0, W la totalité du registre...

Pour se rappeler la signification de ces symboles, il suffit de savoir qu'ils représentent les initiales de mots anglais:

- **A** comme Address car le champ A fait 5 quartets de long (qui est la longueur d'une adresse) et qui est prévu pour contenir des adresses.

**B** comme Byte (octet) car 2 quartets valent un octet.
**M** comme Mantisse (un réel RPL fait 64 bits de long [voir la seconde partie, chapitre II] et la mantisse recouvre cette zone).
**S** comme Sign (pour la même raison que précédement).
**X** comme eXponent (comme M et S...).
**XS** comme eXponent-Sign (sur le 71 l'exposant allait de 0 à 99: il fallait aussi stocker son signe...).
**W** comme Wide.

Il existe deux autres types de champs: le champ P et le champ WP (wide-P) dont la taille dépent du registre P: le champ P est le quartet de numéro P , WP contient les quartets 0 à P.

---

**Codage des champs:**

On écrit après l'opération le nom du champ concerné:
Par exemple ?C=0 A signifie: est-ce que le champ A du registre C vaut zéro?

Pour le codage des instructions il y a deux cas:

- soit l'opération existe directement avec le champ concerné (c'est toujours le cas pour le champ A, quelquefois pour le champ B), alors le code est donné directement;
- Sinon le code est donné avec une lettre minuscule (a ou b) à remplacer par le quartet signifiant le champ donné.

*Voici le tableau de convertion:*

| Champ | a | b |
|---|---|---|
| P | 0 | 8 |
| WP | 1 | 9 |
| XS | 2 | A |
| X | 3 | B |
| S | 4 | C |
| M | 5 | D |
| B | 6 | E |
| W | 7 | F |

Ainsi lorsque l'on rencontre, dans le résumé des différents codes, une ligne du type: **Ab0 A=0 b**

Pour coder A=0 W, on utilisera les codes AF0 (F pour W puisque le symbole est b).

Un autre type de codage est de définir le nombre de quartets sur lequel doit porter l'opération: dans la liste, ceci est symbolisé par x+1 ou x. Par exemple LCHEX qui charge une constante hexadécimale dans le registre C, est présenté de la manière: 3xh1....h# LCHEX h#...h1 x=#1. Ce qui signifie que 3 est le premier code, suivi par x qui représente le nombre de chiffres de la constante moins 1 suivi de la constante (retournée). Par exemple pour placer 123 dans C on écrira 321.

**ATTENTION** : l'affectation des chiffres commence au quartet de n° P et se poursuit cycliquement: si dans l'affectation précédente on avait P=E et C=0000000000000000 avant l'affectation, alors, après celle ci, C vaudra: 2300000000000001...

---

**Ecriture et lecture en mémoire:**

Celle-ci se fait en deux étapes:

- définition de l'adresse de travail dans un des deux registres D0 ou D1;
- écriture ou lecture à l'aide de A ou C.

**ATTENTION** : la lecture et l'écriture retournent les quartets lus.

Par exemple, si la mémoire contient les quartets 0123456789ABCDEF en #4F810, après la séquence:

**D1=(5) 4F810**, affectation de l'adresse 4F810 à D1;

**C=DAT1 W** lecture de 16 quartets (champ W) en D1, stockés dans C. C contiendra le nombre **FEDCBA9876543210.**

De même :

| | |
|---|---|
| **D1=(5) 4F810** | affectation à D1 de l'adresse 4F810; |
| **LCHEX (16) 0123456789ABCDEF** | affectation à C de la valeur 0123456789ABCDEF; |
| **DAT1=C W** | écriture des 16 quartets de C (champ W) en D1; |

placera en 4F810 les quartets **FEDCBA9876543210.**

---

**Les tests :**

Ils ont tous un nom commençant par [?]. Ils sont toujours suivis par un saut ou un retour conditionnel: Goyes ou RTNyes effectué si le test est vérifié. RTNyes à pour code 00, Goyes est codé par le nombre de quartets du saut.

---

**Sauts:**

Il existe plusieurs types de saut qui se différencient:

**a) Par leur nature**

- saut de type GOTO: le programme se poursuit à l'adresse indiquée;
- saut de type GOSUB: on exécute la sous-routine située à l'adresse indiquée et l'exécution du programme principal se poursuit dès qu'une instruction de retour est rencontrée dans le sous-programme.

**b) Par leur adressage:**

Il y a deux types d'adressage:

- l'adressage absolu:

  dans ce cas les 5 chiffres suivant l'instruction forment l'adresse absolue à laquelle doit se poursuivre l'exécution du programme;

- l'adressage relatif :

  les chiffres suivant l'instruction de saut indiquent, une valeur à ajouter à l'adresse courante pour obtenir l'adresse à laquelle le programme doit se poursuivre. La valeur en question est considérée comme négative si le bit le plus fort est à un.

  Par exemple 10h est positif (00010000 en binaire) , A0h négatif (10100000 en binaire) et est considéré comme valant -60h (soit -96 en décimal) [cette valeur se calcule en faisant #0 - le nombre et en ne gardant que les chiffres les plus à droite du résultat].

**ATTENTION :** Pour les sauts relatifs: dans le cas d'un saut de type GOTO, on compte le saut relatif à partir de l'adresse du premier quartet de la valeur de saut, tandis que dans le cas d'un saut de type GOSUB,on compte à partir de l'adresse de l'instruction suivante.

**Divers:**

Enfin sont symbolisée par n ou n+1 dans la liste des codes, des constantes.

La liste des codes qui suit ne contient que des explications très succintes: la signification exacte et détaillée de tous les codes est contenue dans le manuel publié par HP: HP-71 Hardware internal Design Spécification (HDS) publié en 1984 par HP sous le numéro de référence 00071-90071.

Ce manuel contient toutes les explications souhaitables à l'exception de celles concernant les nouvelles instructions du SATURN. Parmi celles-ci, une seule est parfaitement connue: il s'agit de l'instruction PC=(A):

**PC=(A) Code: 808C**

Effet : cette instruction utilise l'adresse contenue dans le champ A du registre A: elle lit 5 quartets à cette adresse et place ces 5 quartets (qui forment eux-mêmes une adresse) dans le registre PC provoquant ainsi un saut à cette dernière adresse. Cette instruction est abondamment utilisée sur le HP28.

*Les autres instructions existantes sont:*

**RSI** (Reset System Interrupt) de code 80810 dont le rôle n'a pas été indiqué par HP.

*Et sur le HP28s:*

**?CBIT=0** de code **808A**
**?CBIT=1** de code **808B**
**BUSCD** de code **808D**

Il semble y avoir d'autres nouvelles instructions en 808: 8082, 8083 (???).

---

# Annexe III
# LISTE DES INSTRUCTIONS

Les instructions sont données dans l'ordre des codes. Ceci complète donc le manuel HDS de HP qui les donne dans l'ordre alphabétique (en particulier cette liste est très utile pour désassembler des programmes en Rom...).

Remarque : dans le cas d'instructions similaires seule la première de ces instructions est commentée...

---

| Codes | Mnémoniques | Explications |
|---|---|---|
| 00 | RTNSXM | Retour de sous programme et XM mis à 1 |
| 01 | RTN | Retour de sous programme |
| 02 | RTNSC | Retour de sous programme et CARRY mis à 1 |
| 03 | RTNCC | Retour de sous programme et CARRY mis à 0 |
| 04 | SETHEX | Passage en mode hexadécimal |
| 05 | SETDEC | Passage en mode décimal |
| 06 | RSTK=C | Sauve le champ A de C dans RSTK |
| 07 | C=RSTK | Place le 1er étage de RSTK dans le champ A de C |
| 08 | CLRST | Vide RSTK |
| 09 | C=ST | Copie ST dans le champ X de C |
| 0A | ST=C | Copie le champ X de C dans ST |
| 0B | CSTEX | Echange ST et le champ X de C |
| 0C | P=P+1 | Incrémente P |
| 0D | P=P-1 | Décrémente P |
| 0EF0 | A=A&B A | Effectue un AND entre A champ A et B champ A |
| 0EF1 | B=B&C A | |
| 0EF2 | C=C&A A | |
| 0EF3 | D=D&C A | |
| 0EF4 | B=B&A A | |
| 0EF5 | C=C&B A | |
| 0EF6 | A=A&C A | |
| 0EF7 | C=C&D A | |
| 0EF8 | A=A!B A | Effectue un OR entre A champ A et B champ A |
| 0EF9 | B=B!C A | |
| 0EFA | C=C!A A | |
| 0EFB | D=D!C A | |
| 0EFC | B=B!A A | |
| 0EFD | C=C!B A | |
| 0EFE | A=A!C A | |
| 0EFF | C=C!D A | |
| 0Ea0 | A=A&B a | |
| 0Ea1 | B=B&C a | |
| 0Ea2 | C=C&A a | |
| 0Ea3 | D=D&C a | |

```
0Em4      B=B&A  a
0Em5      C=C&B  a
0Em6      A=A&C  a
0Em7      C=C&D  a
0Em8      A=A!B  a
0Em9      B=B!C  a
0EmA      C=C!A  a
0EmB      D=D!C  a
0EmC      B=B!A  a
0EmD      C=C!B  a
0EmE      A=A!C  a
0EmF      C=C!D  a
0F        RTI            Retour de routine d'interruption
```

---

```
100       R0=A           Copie A dans le registre R0
101       R1=A
102       R2=A
103       R3=A
104       R4=A
108       R0=C
109       R1=C
10A       R2=C
10B       R3=C
10C       R4=C
110       A=R0           Copie R0 dans le registre A
111       A=R1
112       A=R2
113       A=R3
114       A=R4
118       C=R0
119       C=R1
11A       C=R2
11B       C=R3
11C       C=R4
120       AR0EX          Echange les contenus de A et de R0
121       AR1EX
122       AR2EX
123       AR3EX
124       AR4EX
128       CR0EX
129       CR1EX
12A       CR2EX
12B       CR3EX
12C       CR4EX
130       D0=A           Copie le champ A de A dans D0
131       D1=A
132       AD0EX          Echange le contenu de A champ A et de D0
133       AD1EX
134       D0=C
135       D1=C
```

```
136       CD0EX
137       CD1EX
138       D0=AS          Copie les quartets 3 à 0 de A dans ceux de D0
139       D1=AS
13A       AD0XS          Echange les quartets 3 à 0 de A avec ceux de D0
13B       AD1XS
13C       D0=CS
13D       D1=CS
13E       CD0XS
13F       CD1XS
140       DAT0=A A       Ecrit A champ A à partir de l'adr. contenue ds D0
141       DAT1=A A
142       A=DAT0 A       Ecrit dans A champ A les qu. situés à partir de D0
143       A=DAT1 A
144       DAT0=C A
145       DAT1=C A
146       C=DAT0 A
147       C=DAT1 A
148       DAT0=A B
149       DAT1=A B
14A       A=DAT0 B
14B       A=DAT1 B
14C       DAT0=C B
14D       DAT1=C B
14E       C=DAT0 B
14F       C=DAT1 B
150a      DAT0=A a
151a      DAT1=A a
152a      A=DAT0 a
153a      A=DAT1 a
154a      DAT0=C a
155a      DAT1=C a
156a      C=DAT0 a
157a      C=DAT1 a
158x      DAT0=A x+1
159x      DAT1=A x+1
15Ax      A=DAT0 x+1
15Bx      A=DAT1 x+1
15Cx      DAT0=C x+1
15Dx      DAT1=C x+1
15Ex      C=DAT0 x+1
15Fx      C=DAT1 x+1
16n       D0=D0+ n+1     Ajoute à D0 la valeur n+1 (1h<=n+1<=10h)
17n       D1=D1+ n+1
18n       D0=D0- n+1     Soustrait à D0 la valeur n+1 (1h<=n+1<=10h)
19pq      D0=(2) qp      Affecte aux quartets 1 et 0 de D0 les valeurs & p
1Apqrs    D0=(4) srqp    Idem pour 3 2 1 et 0
1Bpqrst   D0=(5) tsrqp   Idem pour la totalité des quartets
1Cn       D1=D1- n+1
1Dpq      D1=(2) qp
1Epqrs    D1=(4) srqp
1Fpqrst   D1=(5) tsrqp
```

```
2n        P=     n      Affecte la valeur n (0<=n<=Fh)
```

```
3xh1..h#  LCHEX  h#..h1  x=#-1   Voir annexe précédente
```

```
400       RTNC           Retour de sous programme si CARRY=1
420       NOP3           No OPération
4yz       GOC    zy      Saut relatif si CARRY=1 (sinon: rien)
```

```
500       RTNNC          Retour de sous programme si CARRY=0
5yz       GONC   zy      Saut relatif si CARRY=0  (sinon: rien)
```

```
6300      NOP4
64000     NOP5
6yzt      GOTO   tzy     Saut relatif
```

```
7yzt      GOSUB  tzy     Saut relatif
```

```
800       OUT=CS         Le quartet faible de C est copié dans celui de
                         OUTPUT
801       OUT=C          C champ X est copié dans OUTPUT
802       A=IN           Copie INPUT dans A
803       C=IN
804       UNCNFG         Commande du Bus. Voir les HDS
805       CONFIG         Idem
806       C=ID           Idem
807       SHUTDN         Idem
80810     INTON          Autorise les interruptions
80810     RSI            ?????
808A      ?CBIT=0        ?????
808B      ?CBIT=1        ?????
808C      PC=(A)         Voir annexe précédente
808D      BUSCD          ?????
808F      INTOFF         Interdit les interruptions
809       C+P+1          C=C+P+1 champ A
80A       RESET          Commande du Bus. Voir les HDS
80B       BUSCC          Idem
80Cx      C=P    x       P est copié dans le quartet x de C
80Dx      P=C    x
80Fx      CPEX   x
```

```
810       ASLC          Rotation circ d'un quartet à gauche de A champ W
811       BSLC          Idem pour B
812       CSLC          Idem pour C
813       DSLC
814       ASRC          Idem mais vers la droite
815       BSRC
816       CSRC
817       DSRC
81C       ASRB          Décalage à droite d'un bit.Le bit sortant va ds
SB
81D       BSRB          (division par 2). Idem pour B
81E       CSRB          Idem pour C
81F       DSRB
821       XM=0          Affecte la valeur 0 à XM
822       SB=0
824       SR=0
828       MP=0
82F       CLRHST        Idem pour les 4 indicateurs XM, SB, SR, MP
831       ?XM=0         Test: XM est il nul?
832       ?SB=0
834       ?SR=0
838       ?MP=0
84d       ST=0   d      Met à zéro le drapeau d de ST
85d       ST=1   d
86d       ?ST=0  d
87d       ?ST=1  d
88n       ?P#    n      Test: P est-il différent de n?
89n       ?P=    n      Test: P est-il égal à n?
8A0       ?B=A   A      Test: B champ A est-il égal à A champ A?
8A1       ?C=B   A
8A2       ?A=C   A
8A3       ?C=D   A
8A4       ?B#A   A
8A5       ?C#B   A
8A6       ?A#C   A
8A7       ?D#C   A
8A8       ?A=0   A
8A9       ?B=0   A
8AA       ?C=0   A
8AB       ?D=0   A
8AC       ?A#0   A
8AD       ?B#0   A
8AE       ?C#0   A
8AF       ?D#0   A
8B0       ?A>B   A      Test de comparaison...
8B1       ?B>C   A
8B2       ?C>A   A
8B3       ?D>C   A
8B4       ?A<B   A
8B5       ?B<C   A
8B6       ?C<A   A
8B7       ?D<C   A
```

```
9B8       ?A≥B   A
9B9       ?B≥C   A
9BA       ?C≥A   A
9BB       ?D≥C   A
9BC       ?A≤B   A
9BD       ?B≤C   A
9BE       ?C≤A   A
9BF       ?D≤C   A
9Cpgrs    GOLONG srqp   GOTO relatif
9Dpgrst   GOVLNG tsrqp  GOTO absolu
9Epgrs    GOSUBL srqp   GOSUB relatif
9Fpgrst   GOSBVL tsrqp  GOSUB absolu
```

---

```
9a0       ?A=B   a      Tests...
9a1       ?B=C   a
9a2       ?C=A   a
9a3       ?C=D   a
9a4       ?A≠B   a
9a5       ?B≠C   a
9a6       ?C≠A   a
9a7       ?D≠C   a
9a8       ?A=0   a
9a9       ?B=0   a
9aA       ?C=0   a
9aB       ?D=0   a
9aC       ?A≠0   a
9aD       ?B≠0   a
9aE       ?C≠0   a
9aF       ?D≠0   a
9b0       ?A>B   b
9b1       ?B>C   b
9b2       ?C>A   b
9b3       ?D>C   b
9b4       ?A<B   b
9b5       ?B<C   b
9b6       ?C<A   b
9b7       ?D<C   b
9b8       ?A≥B   b
9b9       ?B≥C   b
9bA       ?C≥A   b
9bB       ?D≥C   b
9bC       ?A≤B   b
9bD       ?B≤C   b
9bE       ?C≤A   b
9bF       ?D≤C   b
```

---

```
Aa0        A=A+B  a    Affecte à A champ a la somme de A champ a et B
champ a
Aa1       B=B+C  a
Aa2       C=C+A  a
Aa3       D=D+C  a
Aa4       A=A+A  a
```

```
AmS       B=B+B   a
Am6       C=C+C   a
Am7       D=D+D   a
Am8       B=B+A   a
Am9       C=C+B   a
AmA       A=A+C   a
AmB       C=C+D   a
AmC       A=A-1   a       Décrémente le champ a de A
AmD       B=B-1   a
AmE       C=C-1   a
AmF       D=D-1   a
Ah0       A=0     b       Met à zéro le champ b de A
Ah1       B=0     b
Ah2       C=0     b
Ah3       D=0     b
Ah4       A=B     b       Copie B champ b dans A champ b
AhS       B=C     b
Ah6       C=A     b
Ah7       D=C     b
Ah8       B=A     b
Ah9       C=B     b
AhA       A=C     b
AhB       C=D     b
AhC       ABEX    b       Echange le champ b de A et celui de B
AhD       BCEX    b
AhE       ACEX    b
AhF       CDEX    b
```

---

```
Bm0       A=A-B   a
Bm1       B=B-C   a
Bm2       C=C-A   a
Bm3       D=D-C   a
Bm4       A=A+1   a
BmS       B=B+1   a
Bm6       C=C+1   a
Bm7       D=D+1   a
Bm8       B=B-A   a
Bm9       C=C-B   a
BmA       A=A-C   a
BmB       C=C-D   a
BmC       A=B-A   a
BmD       B=C-B   a
BmE       C=A-C   a
BmF       D=C-D   a
Bh0       ASL     b       Multiplie A champ b par 2 (le bit sortant est
perdu)
Bh1       BSL     b
Bh2       CSL     b
Bh3       DSL     b
Bh4       ASR     b       Divise A champ b par 2 (le bit sortant va dans
SB)
BhS       BSR     b
Bh6       CSR     b
Bh7       DSR     b
```

```
Bb8       A=-A   b      Inverse la valeur du champ b de A
Bb9       B=-B   b
BbA       C=-C   b
BbB       D=-D   b
BbC       A=-A-1 b      A=NOT(A) sur le champ b
BbD       B=-B-1 b
BbE       C=-C-1 b
BbF       D=-D-1 b
```

---

```
C0        A=A+B  A
C1        B=B+C  A
C2        C=C+A  A
C3        D=D+C  A
C4        A=A+A  A
C5        B=B+B  A
C6        C=C+C  A
C7        D=D+D  A
C8        B=B+A  A
C9        C=C+B  A
CA        A=A+C  A
CB        C=C+D  A
CC        A=A-1  A
CD        B=B-1  A
CE        C=C-1  A
CF        D=D-1  A
```

---

```
D0        A=0    A
D1        B=0    A
D2        C=0    A
D3        D=0    A
D4        A=B    A
D5        B=C    A
D6        C=A    A
D7        D=C    A
D8        B=A    A
D9        C=B    A
DA        A=C    A
DB        C=D    A
DC        ABEX   A
DD        BCEX   A
DE        ACEX   A
DF        CDEX   A
```

---

```
E0        A=A-B  A
E1        B=B-C  A
E2        C=C-A  A
E3        D=D-C  A
E4        A=A+1  A
E5        B=B+1  A
E6        C=C+1  A
```

```
E7          D=D+1  A
E8          B=B-A  A
E9          C=C-B  A
EA          A=A-C  A
EB          C=C-D  A
EC          A=B-A  A
ED          B=C-B  A
EE          C=A-C  A
EF          D=C-D  A
```

---

```
F0          ASL    A
F1          BSL    A
F2          CSL    A
F3          DSL    A
F4          ASR    A
F5          BSR    A
F6          CSR    A
F7          DSR    A
F8          A=-A   A
F9          B=-B   A
FA          C=-C   A
FB          D=-D   A
FC          A=-A-1 A
FD          B=-B-1 A
FE          C=-C-1 A
FF          D=-D-1 A
```

---

# Annexe IV
# LISTE EXHAUSTIVE DES MESSAGES D'ERREURS

Voici la liste complète des messages d'erreurs des 3 versions de la HP28. Certains des messages ne sont pas utilisés en tant que message d'erreurs, mais en tant que texte pour certaines opérations (CATALOG par exemple). C'est le cas pour des messages comme "Any Object" (#108h).

**Pour les HP28c (1BB & 1CC):**

# 1h: "Insufficient Memory"
# 2h: "Range Exception"
# 3h: "Undefined Local Name"
# 4h: "Undefined ROM Pointer"
# 5h: "Memory Lost"
# 101h: "No Room for UNDO"
# 102h: "Can't Edit CHR(0)"
# 103h: "Improper User Funct"
# 104h: "No Current Equation"
# 105h: "No Room to ENTER"
# 106h: "Syntax Error"
# 107h: ""
# 108h: "Any Object"
# 109h: "Real Number"
# 10Ah: "Complex Number"
# 10Bh: "String"
# 10Ch: "Array"
# 10Dh: "List"
# 10Eh: "Name"
# 10Fh: "Local Name"
# 110h: "Program"
# 111h: "Algebraic"
# 112h: "Algebraic or Name"
# 113h: "Binary Integer"
# 114h: "No Arguments"
# 115h: "1: N (a Real)"
# 116h: "Stack Depth=N+1"
# 117h: "NEXT"
# 118h: "PREV"
# 119h: "SCAN"
# 11Ah: "STOP"
# 11Bh: "USE"
# 11Ch: "FETCH"
# 11Dh: "QUIT"
# 11Eh: "Invalid PPAR"
#11Fh: "Non-real Result"
#120h: "Unable to Isolate"
#121h: "HALT not Allowed"
#122h: ""
#123h: ""
#124h: "UNDO Disabled"
#125h: "Command Stack Disabled"
#126h: "Edit line > 4096"
#127h: "Program Structure"
#128h: "Wrong Argument Count"
#129h: "Circular Reference"
#201h: "Too Few Arguments"
#202h: "Bad Argument Type"
#203h: "Bad Argument Value"
#204h: "Undefined Name"
#205h: "LAST Disabled"
#301h: "Positive Underflow"
#302h: "Negative Underflow"
#303h: "Overflow"
#304h: "Undefined Result"
#305h: "Infinite Result"
#501h: "Invalid Dimension"
#601h: "Invalid ∑DAT"
#602h: "Nonexistent ∑DAT"
#603h: "Insufficient ∑Data"
#604h: "Invalid ∑PAR"
#A01h: "Bad Guess(es)"
#A02h: "Constant?"
#A03h: "Interrupted"
#A04h: "Zero"
#A05h: "Sign Reversal"
#A06h: "Extremum"
#B01h: "Invalid Unit String"
#B02h: "Inconsistent Units"

**Pour la HP28s (2BB) :**

```
#     1h: "Insufficient Memory"
#     2h: ""
#     3h: "Undefined Local Name"
#     4h: ""
#     5h: "Memory Lost"
#     6h: "Power Lost"
# 101h: "No Room for UNDO"
# 102h: "Can't Edit CHR(0)"
# 103h: "Improper User Function"
# 104h: "No Current Equation"
# 105h: "No Room to ENTER"
# 106h: "Syntax Error"
# 107h: ""
# 108h: "Any Object"
# 109h: "Real Number"
# 10Ah: "Complex Number"
# 10Bh: "String"
# 10Ch: "Array"
# 10Dh: "List"
# 10Eh: "Global Name"
# 10Fh: "Local Name"
# 110h: "Program"
# 111h: "Algebraic"
# 112h: "Algebraic or Name"
# 113h: "Binary Integer"
# 114h: "No Arguments"
# 115h: "1: N (a Real)"
# 116h: "Stack Depth=N+1"
# 117h: "NEXT"
# 118h: "PREV"
# 119h: ""
# 11Ah: "STOP"
# 11Bh: "USE"
# 11Ch: "FETCH"
# 11Dh: "QUIT"
# 11Eh: "Invalid PPAR"
#11Fh: "Non-Real Result"
#120h: "Unable to Isolate"
#121h: "HALT Not Allowed"
#122h: ""
#123h: ""
#124h: "UNDO Disabled"
#125h: "Command Stack Disabled"
#126h: ""
#127h: "Program Structure"
#128h: "Wrong Argument Count"
#129h: "Circular Reference"
#12Ah: "Directory Not Allowed"
#12Bh: "Non-Empty Directory"
#201h: "Too Few Arguments"
#202h: "Bad Argument Type"
#203h: "Bad Argument Value"
#204h: "Untefined Name"
#205h: "LAST Disabled"
#301h: "Positive Underflow"
#302h: "Negative Underflow"
#303h: "Overflow"
#304h: "Undefined Result"
#305h: "Infinite Result"
#501h: "Invalid Dimension"
#601h: "Invalid ∑DAT"
#602h: "Nonexistent ∑DAT"
#603h: "Insufficient ∑Data"
#604h: "Invalid ∑PAR"
#A01h: "Bad Guess(es)"
#A02h: "Constant?"
#A03h: "Interrupted"
#A04h: "Zero"
#A05h: "Sign Reversal"
#A06h: "Extremum"
#B01h: "Invalid Unit String"
#B02h: "Inconsistent Units"
```

# Annexe V
# LISTE DES OBJETS SITUES EN ROM

Cette liste est divisée en trois parties: une par version de HP28. Chacune des parties comprend les listes suivantes:

**1) Liste des instructions RPL sous la forme:**

n° de l'instruction  #adresse  instruction

"n° de l'instruction" est un nombre de référence permettant la conversion adresse<—>instruction par la machine.

"#adresse" : adresse d'éxécution de l'instruction (utilisable par SYSEVAL).

*Exemple :* (pour la 28c 1BB) DUP est l'instruction 0 et a pour adresse #177F1 (ainsi faire #177F1 syseval est équivalent à effectuer DUP).

Les instructions sont subdivisées en deux listes selon leur type:

Type 1 : instructions sans saut;

Type 2 : instructions pouvant donner lieu à un saut (IF, IFERR, DO...).

**2) Liste des adresse des différents objets sous la forme:**

#adresse  objet

cette seconde liste est divisée par type d'objet dans l'ordre:

Bytes, Short Integers, Global names, Local Names, Rom pointers, Binary Integers, Reals, Extended Reals, Strings, Complexs, Extended complex.

"#adresse" est l'adresse de l'objet (ainsi #adresse SYSEVAL renvoit-il l'objet correspondant dans la pile...).

### Instructions de type 1

0 #1779F1h DUP
1 #17980Bh DUP2
2 #179825h SWAP
3 #17983Fh DROP
4 #179859h DROP2
5 #179873h ROT
6 #17988Dh OVER
7 #1798A7h DEPTH
8 #1798C6h DROPN
9 #1798E0h DUPN
10 #1798FFh PICK
11 #179919h ROLL
12 #179933h ROLLD
13 #17994Dh CLEAR
14 #179968h ->LIST
15 #179990h R->C
16 #1799EBh RE
17 #179A0Bh IM
18 #179A25h SUB
19 #179B0Bh LIST->
20 #179B43h C->R
21 #179B85h SIZE
22 #179C03h POS
23 #179C27h ->STR
24 #179C41h STR->
25 #179C65h NUM
26 #179C89h CHR
27 #179CADh TYPE
28 #179D62h ->ARRY
29 #179DDBh ARRY->
30 #179E36h RDM
31 #179EA0h CON
32 #179F7Dh IDN
33 #18014h TRN
34 #18065h PUT
35 #180BBh PUTI
36 #1830Ah GET
37 #18360h GETI
38 #189AEh SAME
39 #189D7h AND
40 #18950h OR
41 #189C9h NOT
42 #18A1Fh XOR
43 #18A98h ==
44 #18B0Ch ≠
45 #18B85h <
46 #18BF9h >
47 #18C6Dh ≤
48 #18CE1h ≥
49 #18D55h =
50 #18DD3h NEG
51 #18E1Fh ABS
52 #18E6Bh CONJ
53 #18EB7h x
54 #18EE5h MAXR
55 #18F13h MINR
56 #18F41h ∞
57 #18F6Fh 1
58 #18F9Dh +
59 #19057h -
60 #190FDh *
61 #191CBh /
62 #19271h ^
63 #19353h INV
64 #1939Fh ARG
65 #193F5h P->R
66 #1951Dh R->P
67 #1959Bh SIGN
68 #195DDh √
69 #1964Ch SQ
70 #196C0h SIN
71 #19702h COS
72 #19744h TAN
73 #19786h SINH
74 #197C8h COSH
75 #1980Ah TANH
76 #1984Ch ASIN
77 #198C5h ACOS
78 #1993Eh ATAN
79 #19980h ASINH
80 #199C2h ACOSH
81 #19A36h ATANH
82 #19AAFh EXP
83 #19AF1h LN
84 #19B60h LOG
85 #19BCFh ALOG
86 #19C11h LNP1
87 #19C49h EXPM
88 #19C81h FACT
89 #19CB9h IP
90 #19CF1h FP
91 #19D29h FLOOR
92 #19D61h CEIL
93 #19D99h XPON
94 #19DD1h MAX
95 #19E1Dh MIN
96 #19E69h MOD
97 #19EB5h MANT
98 #19EEDh D->R
99 #19F25h R->D
100 #19F53h ->HMS
101 #19F77h HMS->
102 #19F9Bh HMS+
103 #19FBFh HMS-
104 #19FE3h RNRM
105 #1A007h CNRM
106 #1A02Bh DET
107 #1A04Fh DOT
108 #1A073h CROSS
109 #1A097h RSD
110 #1A0C5h %
111 #1A111h %T
112 #1A15Dh %CH
113 #1A19Fh RAND
114 #1A1B9h RDZ
115 #1A1DDh RCL
116 #1A242h STO
117 #1A2B6h PURGE
118 #1A32Ah MEM
119 #1A362h ORDER
120 #1A3A4h CLUSR
121 #1A3C3h KILL
122 #1A3DDh ABORT
123 #1A3F7h ERRN
124 #1A411h ERRM
125 #1A42Bh EVAL
126 #1A463h IFTE
127 #1A527h IFT
128 #1A592h SYSEVAL
129 #1A5DCh DISP
130 #1A60Ah RND
131 #1A638h BEEP
132 #1A65Ch ->NUM
133 #1A67Bh LAST

134 #1A761h WAIT
135 #1A785h CLLCD
136 #1A79Fh KEY
137 #1A7B9h CLMF
138 #1A7DDh SF
139 #1A81Ah CF
140 #1A857h FS?
141 #1A894h FC?
142 #1A8D6h DEG
143 #1A8FAh RAD
144 #1A91Eh FIX
145 #1A956h SCI
146 #1A98Eh ENG
147 #1A9C6h STD
148 #1A9E0h FS?C
149 #1AA31h FC?C
150 #1AA87h BIN
151 #1AAA1h DEC
152 #1AABBh HEX
153 #1AAD5h OCT
154 #1AAEFh STWS
155 #1AB13h RCWS
156 #1AB32h RCLF
157 #1AB51h STOF
158 #1ABEDh STOΣ
159 #1AC25h CLΣ
160 #1AC3Fh RCLΣ
161 #1AC59h Σ+
162 #1AC9Bh Σ-
163 #1ACB5h MΣ
164 #1ACCFh CORR
165 #1ACE9h COV
166 #1AD03h MAXΣ
167 #1AD1Dh MEAN
168 #1AD37h MINΣ
169 #1AD51h SDEV
170 #1AD6Bh TOT
171 #1AD85h VAR
172 #1AD9Fh LR
173 #1ADB9h PREDV
174 #1ADDDh UTPC
175 #1AE01h UTPN
176 #1AE25h UTPF
177 #1AE49h UTPT
178 #1AE6Dh SCLΣ
179 #1AE87h DRMΣ
180 #1AEA1h COLΣ
181 #1AEC5h SINV
182 #1AF5Ch SNEG
183 #1AFF3h SCONJ
184 #1B076h STO+
185 #1B18Fh STO−
186 #1B2ADh STO/
187 #1B3CBh STO*
188 #1B683h EXGET
189 #1B6A7h EXSUB
190 #1B6DFh OBSUB
191 #1B703h OBGET
192 #1B727h FORM
193 #1B75Fh COLCT
194 #1B797h EXPAN
195 #1B7CFh ISOL
196 #1B7F3h QUAD
197 #1B817h SHOW
198 #1B845h TAYLR
199 #1B873h ∂
200 #1B914h RCEQ
201 #1B933h STEQ
202 #1B952h ROOT
203 #1B9F8h ∫
204 #1BB70h ASR
205 #1BB94h RL
206 #1BBB8h RLB
207 #1BBDCh RR
208 #1BC00h RRB
209 #1BC24h SL
210 #1BC48h SLB
211 #1BC6Ch SR
212 #1BC90h SRB
213 #1BCB4h R->B
214 #1BCD8h B->R
215 #1BCFCh CONVERT
216 #1BD3Eh INDEP
217 #1BD6Ch PMIN
218 #1BD90h PMAX
219 #1BDB4h AXES
220 #1BDD8h CENTR
221 #1BDFCh RES
222 #1BE20h *H
223 #1BE44h *W
224 #1BE68h DRAW
225 #1BE87h PIXEL
226 #1BEABh DRAX
227 #1BECAh PR1
228 #1BEE4h PRSTC
229 #1BEFEh PRST
230 #1BF18h CR
231 #1BF32h PRUSR
232 #1BF4Ch PRVAR
233 #1BF70h PRMD
234 #1BF8Ah PRLCD

## Instructions de type 2

0 #27CB4h IF
1 #27CC9h THEN(IF)
2 #27D29h ELSE(IF,IFERR)
3 #27D48h END(IF,IFERR)
4 #27D5Dh instr. incon
5 #27D77h ->
6 #27EAEh instr. incon
8 #27F0Ah «
9 #27F1Fh »
12 #27F63h WHILE
13 #27F82h REPEAT
14 #27FC4h END(WHILE)
15 #27FE3h DO
16 #29002h UNTIL
17 #29017h END(DO)
18 #29059h START
19 #290A0h FOR
20 #290FBh NEXT
21 #29229h STEP
22 #29264h IFERR
23 #2932Bh THEN(IFERR)
24 #29340h HALT

### Bytes HP28c 1BB

```
# 345Ch   41h
# 41FEh   A8h
# 440Dh   41h
# 6D0Bh   11h
# 86FDh   20h
# 8E18h   20h
# A441h   20h
# A6B9h   2Ch
# AB8Bh   3Ah
# AD60h   2Eh
# AD67h   2Ch
#113SFh   27h
#11464h   2Fh
#1147Ah   2Ah
#11490h   2Dh
#114A6h   2Bh
#114BCh   25h
#115S5h   5Eh
#115S8h   83h
#115E2h   84h
#115F8h   93h
#1160Eh   97h
#11624h   3Fh
#1163Ah   2Ch
#11650h   2Eh
#11666h   30h
#1167Ch   31h
#11692h   32h
#116A8h   33h
#116BEh   34h
#116D4h   35h
#116EAh   36h
#11700h   37h
#11716h   38h
#1172Ch   39h
#11742h   88h
#11779h   8Bh
#1350Ch   20h
#2E55Eh   00h
#3FC15h   0Ah
```

### Short Integers HP28c 1BB

```
# 1248h   0000Ch
# 34B8h   FFFFFh
# 3F09h   61441h
# 410Ch   A8241h
# 4166h   102A8h
# 41C2h   E13A8h
# 4CD9h   61441h
# 6C98h   02933h
# 6CA2h   02977h
# 6CACh   02A0Ah
# 6CB6h   02A4Eh
# 6CC0h   02A96h
# 6CCAh   02D12h
# 6CD4h   02911h
# 6CD9h   02911h
# 6CDEh   02C67h
# 6CE8h   02ADAh
# 6CF2h   02AB8h
# 6CFCh   02A70h
# 6D06h   029BFh
# 6D10h   02D37h
# 6D1Ah   02D5Ch
# 6D24h   02955h
# 6D2Eh   0299Dh
# 6D38h   02A2Ch
# 6D42h   02C96h
# 6D4Ch   029E1h
# 6D56h   00000h
# 6D60h   00001h
# 6D6Ah   00002h
# 6D74h   00003h
# 6D7Eh   00004h
# 6D88h   00005h
# 6D92h   00006h
# 6D9Ch   00007h
# 6DA6h   00008h
# 6DB0h   00009h
# 6DBAh   0000Ah
# 6DC4h   0000Bh
# 6DCEh   0000Ch
# 6DD8h   0000Dh
# 6DE2h   0000Eh
# 6DECh   0000Fh
# 6DF6h   00010h
# 6E00h   00011h
# 6E0Ah   00012h
# 6E14h   00013h
# 6E1Eh   00014h
# 6E28h   00015h
# 6E32h   00016h
# 6E3Ch   00017h
# 6E46h   00018h
# 6E50h   00019h
# 6E5Ah   0001Ah
# 6E64h   0001Bh
# 6E6Eh   0001Ch
# 6E78h   0001Dh
# 6E82h   0001Eh
# 6E8Ch   0001Fh
# 6E96h   00020h
# 6EA0h   00021h
# 6EAAh   00022h
# 6EB4h   00023h
# 6EBEh   00024h
# 6EC8h   00025h
# 6ED2h   00026h
# 6EDCh   00027h
# 6EE6h   00028h
# 6EF0h   00029h
# 6EFAh   0002Ah
# 6F04h   0002Bh
# 75E7h   00106h
# 75F1h   10080h
# 75FBh   04080h
# 7605h   00880h
# 760Fh   00180h
# 7619h   00280h
# 7623h   00480h
# 7FDDh   00EFFh
# 82F4h   00040h
# 8308h   0005Bh
# 8951h   0029Ah
# D167h   00016h
# D1C9h   00420h
# F06Ah   00015h
#1047Ch   00123h
#1049Ah   00122h
#13EB7h   00820h
#1415Ah   00120h
#163CCh   00017h
#1667Bh   00027h
#166FAh   00014h
#1671Dh   00015h
#16731h   00016h
#1682Bh   00017h
#168E4h   0001Dh
#16CB1h   000FAh
#17039h   00054h
#19302h   00304h
#1A699h   00205h
#1C7F1h   02955h
#1E5CCh   00039h
#1E5D6h   0003Ah
#1E5E0h   0003Bh
#1E5EAh   0003Ch
#1E5F4h   0003Dh
#1E5FEh   0003Eh
#1E608h   0003Fh
#1E612h   00040h
#24A16h   00501h
#26A6Eh   01111h
#2B6E2h   0011Fh
#2E89Bh   00A01h
#2E8A5h   00A02h
#2E8AFh   00A03h
#2E8B9h   00A04h
```

```
#2E8C3h  00A05h
#2E8CDh  00A06h
#2F531h  00017h
#31B4Eh  00010h
#32DC6h  4FA16h
#3F725h  0002Dh
#3F72Fh  00030h
#3F739h  00032h
#3F743h  00033h
#3F74Dh  00034h
#3F757h  00037h
#3F761h  0003Fh
#3F76Bh  00041h
#3F775h  00042h
#3F77Fh  00044h
#3F789h  00045h
#3F793h  00046h
#3F79Dh  00050h
#3F7A7h  00051h
#3F7B1h  00052h
#3F7BBh  00055h
#3F7C5h  00060h
#3F7CFh  00061h
#3F7D9h  00062h
#3F7E3h  00064h
#3F7EDh  00065h
#3F7F7h  00070h
#3F801h  00071h
#3F80Bh  00075h
#3F815h  00105h
#3F81Fh  00088h
#3F829h  00089h
#3F833h  000A1h
#3F83Dh  000A2h
#3F847h  000A6h
#3F851h  000A7h
#3F85Bh  000A9h
#3F865h  000AAh
#3F86Fh  000B1h
#3F879h  000BBh
#3F883h  000DDh
#3F88Dh  000DEh
#3F897h  000EDh
#3F8A1h  000EEh
#3F8ABh  00100h
#3F8B5h  00111h
#3F8BFh  00133h
#3F8C9h  00136h
#3F8D3h  00163h
#3F8DDh  00166h
#3F8E7h  00151h
#3F8F1h  00311h
#3F8FBh  00444h
#3F905h  00451h
#3F90Fh  00452h
#3F919h  00510h
#3F923h  00511h
#3F92Dh  00610h
#3F937h  00611h
#3F941h  00651h
#3F94Bh  00652h
#3F955h  00851h
#3F95Fh  00951h
#3F969h  00A11h
#3F973h  00A12h
#3F97Dh  00A15h
#3F987h  00A1Ah
#3F991h  00A21h
#3F99Bh  00A22h
#3F9A5h  00A2Ah
#3F9AFh  00A61h
#3F9B9h  00AA1h
#3F9C3h  00AA2h
#3F9CDh  00AAAh
#3F9D7h  FFFFFh
```

### Noms globaux HP28c 1BB

```
#11906h  'constant'
#1193Bh  'PPAR'
#13013h  'EXPR='
#13024h  'LEFT='
#13035h  'RT='
#13778h  ''
#1377Fh  'EQ'
#27597h  'ΣDAT'
#275A6h  'ΣPAR'
#27889h  'A'
#278AEh  'B'
#278BDh  'C'
#278CCh  'D'
#278DBh  'E'
#278EAh  'F'
#278F9h  'G'
#27908h  'H'
#27917h  'I'
#27926h  'J'
#27935h  'K'
#27944h  'L'
#27953h  'M'
#27962h  'N'
#27971h  'O'
#27980h  'P'
#2798Fh  'Q'
#2799Eh  'R'
#279ADh  'S'
#279BCh  'T'
#279CBh  'U'
#279DAh  'V'
#279E9h  'W'
#279F8h  'X'
#27A07h  'Y'
#27A16h  'Z'
#38915h  'n0'
#38933h  's0'
#3A154h  's1'
```

### Noms locaux HP28c 1BB

```
# A319h  'a'
# A322h  'b'
# A32Bh  'c'
# A334h  'd'
# A69Bh  'dim2'
# A733h  'dim2'
# A742h  'arr'
# A74Fh  'count'
# A8BAh  'lf'
#1039Dh  ''nohalt'
#103CBh  ''halt'
#132F0h  'right'
#13315h  'right'
#1335Fh  'solvid'
#13372h  'badlist'
#133D7h  'solvid'
#133EAh  'badlist'
#13545h  'msg'
#13561h  'msg'
#135D2h  'solvid'
#135F4h  'badlist'
#13618h  'badlist'
#160A2h  'wait?'
#1862Bh  'x'
#1868Eh  'x'
#18796h  'nb'
#187A1h  'i'
#187C3h  'n'
#1BACFh  ''int'
#23041h  '#b'
#2306Ah  '#b'
#230ACh  '#a'
#230B7h  '#b'
#230E5h  '#b'
#230F5h  '#a'
#2315Ah  '#b'
#23183h  '#b'
#231CAh  '#b'
#231F3h  '#b'
#28359h  ''stop'
#28366h  ''noname'
```

```
#28401h  'st'
#2840Ch  'ofs'
#28419h  'tok'
#28444h  'st'
#2844Fh  'ofs'
#2845Ch  'tok'
#29332h  'i'
#2933Bh  'j'
#2936Fh  'i'
#2937Fh  'j'
#29446h  'j'
#29468h  'i'
#2947Bh  'i'
#29D5Fh  ''ofs'
#29D66h  ''tok'
#2C87Bh  '#c'
#2CD90h  '#c'
#2CDAFh  '#c'
#2CE0Ah  '#c'
#2E55Fh  ''
#31111h  'fcn'
#3111Eh  'hex'
#3112Bh  'k'
#31134h  'i'
#3275Eh  ''
#34C99h  ''tcls'
#34CAAh  ''fcls'
#34CEDh  ''tcls'
#34D03h  ''fcls'
#34D3Ch  ''tcls'
#34D4Dh  ''fcls'
#34E4Fh  ''num'
#34E5Eh  ''fcn'
#34F21h  ''dvar'
#353EDh  ''xSYMfcn'
#35404h  ''xfcn'
#35D1Eh  ''nth'
#35E54h  ''scl'
#35E63h  ''xSYMfcn'
#35E7Ah  ''xfcn'
#35EDBh  ''xSYMfcn'
#35EF2h  ''xfcn'
#37A06h  ''dv'
#37E4Bh  ''nm'
#37E59h  ''op'
#388B9h  ''ni'
#388C5h  ''ns'
#38EE7h  ''*s'
#39179h  ''+s'
#39E8Dh  ''fl'
#3A110h  ''c'
#3A11Bh  ''b'
#3A126h  ''a'
#3A1FFh  ''n'
#3A20Ah  ''prog'
#3A428h  ''n'
#3A514h  ''n'
#3A5A6h  ''man1plams'
```

### Rom pointers HP28c 1BB

```
# 420Ah  5C8C5C2A7A8
# 5B1Bh  41211461441
# 6D1Fh  10295502911
#1C800h  E005B71C80A
#32E43h  4132E51F246
```

---

### Binary Integers HP28c 1BB

```
#2E2E6h  #8h
#2E2F1h  #9h
#2E2FCh  #Ah
#2E307h  #Bh
#2E312h  #Ch
#2E31Dh  #111h
#2E32Ah  #121h
#2E337h  #131h
#2E344h  #141h
#2E351h  #151h
#2E35Eh  #161h
#2E36Bh  #171h
#2E378h  #181h
#2E385h  #191h
#2E392h  #1A1h
#2E39Fh  #1B1h
#2E3ACh  #1C1h
#2E3B9h  #1D1h
#2E3C6h  #1E1h
#2E3D3h  #1F1h
#30D0Bh  #0000000000000000h
#30D25h  #0000000100000000h
```

### Réels HP28c 1BB

```
#1E2EAh  +0.00000000000E0
#1E2FFh  +1.00000000000E0
#1E314h  +2.00000000000E0
#1E329h  +3.00000000000E0
#1E33Eh  +4.00000000000E0
#1E353h  +5.00000000000E0
#1E368h  +6.00000000000E0
#1E37Dh  +7.00000000000E0
#1E392h  +8.00000000000E0
#1E3A7h  +9.00000000000E0
#1E3BCh  -1.00000000000E0
#1E3D1h  -2.00000000000E0
#1E3E6h  -3.00000000000E0
#1E3FBh  -4.00000000000E0
#1E410h  -5.00000000000E0
#1E425h  -6.00000000000E0
#1E43Ah  -7.00000000000E0
#1E44Fh  -8.00000000000E0
#1E464h  -9.00000000000E0
#1E479h  +3.14159265359E0
```

### Réels étendus HP28c 1BB

```
#1E48Eh   +3.141592653589793D00E0
#1E4FCh   +0.000000000000000000E0
#1E516h   +1.000000000000000000E0
#1E530h   +2.000000000000000000E0
#1E54Ah   +3.000000000000000000E0
#1E564h   +4.000000000000000000E0
#1E57Eh   +5.000000000000000000E0
#1FACAh   +7.000000000000000000E0
#1FBAAh   -1.211428571428571D00E0
#1FBC4h   +9.335849056603773D00E0
#2518Bh   +2.320000000000000000E0
#251A5h   +3.500000000000000000E0
#25EC1h   +3.300000000000000000E0
#30D59h   +1.000000000000000000E0
#3303Bh   +2.302585092994050D00E0
```

### Chaînes HP28c 1BB

```
# 39D9h   ""
# 7EB2h   " Low Memory!"
# 814Ah   "<>*}]]≤≥->∑°µ"
# 827Dh   "  «=lαYZ#{[[STUVWX
          MNOPQRGHIJKLABCDEF"
# 8AD2h   "SHIFT"
# 8C24h   "VISIT"
# 8D00h   "ENTER"
# 90B3h   "COMMAND"
# 90EEh   "CHS"
# 913Ah   "EEX"
# 916Dh   "1E"
# 9339h   "INS"
# 9349h   "DEL"
# 9359h   "UP"
# 936Fh   "DOWN"
# 937Fh   "LEFT"
# 938Bh   "RIGHT"
# A828h   "   " (3 espaces)
# A8D9h   "("
# A91Fh   ")"
# AA1Dh   "[Empty Stack]"
# ABB0h   "User Variables:"
# AC28h   "No User Variables"
# ACA9h   "Format "
# ACCBh   " Base "
# ACFFh   "RADIANS"
# AD1Fh   "DEGREES"
# AD3Eh   " Radix "
# AD7Dh   "Undo"
# ADA3h   "Command"
# ADCFh   "Last"
# ADEBh   "  Multiline"
# AEFBh   " ON "
# AF0Dh   " OFF"
# B007h   "
          "(CHR(10))
           (NEWLINE)
# C0ECh   "CURSOR"
# C3D2h   "EDIT"
# D1B2h   "BACK"
# E310h   "CLUSR"
# E324h   "ORDER"
# E338h   "MEM"
# E51Eh   "NEXT"
# E896h   "USER"
# EAF6h   "TRIG"
# EC39h   "SOLV"
# F254h   "PPAR"
# F293h   "DRW∑"
# F4DFh   "STD"
# F508h   "FIX"
# F531h   "SCI"
# F55Ah   "ENG"
# F583h   "DEG"
# F5A7h   "RAD"
# F5CBh   "RDX."
# F600h   "RDX,"
# F63Ah   "+CMD"
# F66Fh   "-CMD"
# F6A9h   "+LAST"
# F6E0h   "-LAST"
# F71Ch   "+UNDO"
# F753h   "-UNDO"
# F78Fh   "+ML"
# F7C2h   "-ML"
# F872h   "DEC"
# F89Bh   "HEX"
# F8C4h   "OCT"
# F8EDh   "BIN"
# F92Fh   "SST"
# F976h   "HALT"
# F9FBh   "TRACE"
# FA37h   "NORM"
# FA67h   " "
# FA73h   "  "
# FB3Bh   "STO"
# FC40h   "EVAL"
# FCC5h   "CONTINUE"
# FCF8h   "UNDO"
#11344h   "'"
#11375h   "CONVERT"
```

```
#1139Ch  "CLEAR"
#113BFh  "SWAP"
#113E0h  "STO"
#113FFh  "PURGE"
#11422h  "ROLL"
#11443h  "DROP"
#114EBh  "%CH"
#1150Fh  "INV"
#11533h  "SQ"
#11581h  "LAST"
#115A2h  "RCL"
#115C1h  "EVAL"
#117S8h  "->NUM"
#11B13h  " Not In
          Equation"
#11B56h  "Constant
          Equation"
#11B8Ch  "Using "
#12013h  "DRAW"
#12E8Dh  "SOLVR"
#13042h  " "
#131C5h  ": "
#13269h  "EXPR="
#132AFh  "LEFT="
#13330h  "RIGHT="
#1344Ah  "SOLVING FOR "
#13CB4h  "2: "
#13CD3h  "3: "
#13CE3h  "4: "
#13E5Bh  "1: "
#13E8Eh  "    "
         (3 espaces)
#1439Eh  "No Room to
          Show Stack"
#152E1h  "}"
#153E2h  "{"
#15980h  " { "
#15A62h  "{{ "
#15A95h  "]]"
#165C1h  " USAGE: "
#2986Eh  "HALT"
#29880h  "ELSE"
#29892h  "END"
#298A2h  "UNTIL"
#298B6h  "REPEAT"
#298CCh  "NEXT"
#298DEh  "STEP"
#298F0h  "IF"
#298FEh  "DO"
#2990Ch  "IFERR"
#29920h  "START"
#29934h  "FOR"
#29944h  "THEN"
#299S6h  "->"
#29962h  "WHILE"
#2BB6Fh  "+  "
#2BB7Fh  "-  "
#2BB99h  "?  "
#2F29Ch  "  "
#2FAC1h  "Kg"
#2FACFh  "n"
#2FADBh  "A"
#2FAE7h  "s"
#2FAF3h  "OK"
#2FB01h  "cd"
#2FB0Fh  "mol"
#2FB1Fh  "?"
#3ACDFh  "{<-}"
#3AD30h  "{->}"
#3AD4Ah  "EXGET"
#3AEEEh  "DNEG"
#3AF2Dh  "DINV"
#3AF6Ch  "*1"
#3AFA7h  "^1"
#3AFE2h  "/1"
#3B01Dh  "+1-1"
#3B0D4h  "EXPAN"
#3B115h  "COLCT"
#3B156h  "<-->"
#3B17Dh  "<-A"
#3B1A4h  "A->"
#3B1CBh  "AF"
#3B1F2h  "M->"
#3B219h  "<-M"
#3B240h  "-()"
#3B269h  "1/()"
#3B294h  "E()"
#3B2BDh  "L()"
#3B2E6h  "L*"
#3B30Dh  "E^"
#3B334h  "->()"
#3B35Dh  "<-D"
#3B384h  "D->"
#3EED2h  "
          " (23 espaces)
#3FA9Bh  "YES"
#3FAABh  "NO"
#3FAB9h  "Purge?"
#3FACFh  "Out Of Memory"
#3FAF3h  "Stack"
#3FB07h  "UNDO Stack"
#3FB25h  "Command Stack"
#3FB49h  "Last Argu-
          ments"
#3FB6Fh  "System Object"
#3FB93h  "]"
#3FB9Fh  "{ "
#3FBADh  "{"
#3FBB9h  "{"
#3FBC5h  "}"
#3FBD1h  "#"
#3FBDDh  "»"
#3FBE9h  "«"
#3FBF5h  "     "{S
espaces}
#3FC09h  "
          "(CHR(10))
           (NEWLINE)
#3FC1Ch  "der"
#3FC2Ch  "IFTE"
#3FC3Eh  "NOT"
#3FC4Eh  "NOT "
#3FC60h  "AND"
#3FC70h  " AND "
#3FC84h  "XOR"
#3FC94h  " XOR "
#3FCA8h  "OR"
#3FCB6h  " OR "
#3FCC8h  " "
#3FCD4h  " '"
#3FCE2h  "LEVEL "
#3FCF8h  "UNKNOWN"
#3FD10h  "*"
#3FD1Ch  "'"
#3FD28h  ","
#3FD34h  "."
#3FD40h  "["
#3FD4Ch  "]"
#3FD58h  "^"
#3FD64h  "*"
#3FD70h  "/"
#3FD7Ch  "+"
#3FD88h  "-"
#3FD94h  "="
#3FDA0h  "=="
#3FDAEh  "<"
#3FDBAh  ">"
#3FDC6h  "√"
#3FDD2h  "∂"
#3FDDEh  "≤"
#3FDEAh  "≥"
#3FDF6h  "≠"
#3FE02h  "1"
#3FE0Eh  "2"
#3FE1Ah  "3"
#3FE26h  "4"
#3FE32h  "5"
#3FE3Eh  "6"
#3FE4Ah  "7"
#3FE56h  "8"
#3FE62h  "9"
#3FE6Eh  " Version 1BB"
```

### Complexes HP28c 1BB

```
#119A0h  (-6.800000000000E0,-1.500000000000E0)
#119C5h  (+6.800000000000E0,+1.600000000000E0)
#119EFh  (+0.000000000000E0,+0.000000000000E0)
#32F39h  (+0.000000000000E0,+0.000000000000E0)
#32F8Dh  (+1.000000000000E0,+0.000000000000E0)
#32FB2h  (-1.000000000000E0,+0.000000000000E0)
#32FD7h  (+0.000000000000E0,+1.000000000000E0)
#32FFCh  (+0.000000000000E0,-1.000000000000E0)
#3FA5Fh  (+0.000000000000E0,+0.000000000000E0)
```

### Complexes étendus HP28c 1BB

```
#119A0h  (+0.000968000000000E0,+0.000029779150000E0)
#32F8Dh  (+0.000010000000000E0,+0.000029770000000E0)
#32FB2h  (+0.000910000000000E0,+0.000029770000000E0)
#32FD7h  (+0.000000000000000E0,+0.000029770100000E0)
#32FFCh  (+0.000000000000000E0,-9.999029559100000E0)
#3FA5Fh  (+0.000000000000000E0,+0.202933200000000E0)
```

---

### Instructions de type 1

0 #1778D3h DUP
1 #1778EDh DUP2
2 #17790Fh SWAP
3 #179211h DROP
4 #17933Bh DROP2
5 #179555h ROT
6 #17966Fh OVER
7 #179889h DEPTH
8 #179A8h DROPN
9 #179C2h DUPN
10 #179E1h PICK
11 #179FBh ROLL
12 #17A15h ROLLD
13 #17A2Fh CLEAR
14 #17A4Eh ->LIST
15 #17A72h R->C
16 #17ACDh RE
17 #17B2Dh IM
18 #17B97h SUB
19 #17BEDh LIST->
20 #17C25h C->R
21 #17C67h SIZE
22 #17CE5h POS
23 #17D09h ->STR
24 #17D23h STR->
25 #17D47h NUM
26 #17D6Bh CHR
27 #17D8Fh TYPE
28 #17E44h ->ARRY
29 #17EBDh ARRY->
30 #17F18h RDM
31 #17F82h CON
32 #1805Fh IDN
33 #180F6h TRN
34 #18147h PUT
35 #1819Dh PUTI
36 #183ECh GET
37 #18442h GETI
38 #18990h SAME
39 #189B9h AND
40 #18A32h OR
41 #18AABh NOT
42 #18B01h XOR
43 #18B7Ah ==
44 #18BF8h ≠
45 #18C71h <
46 #18CE5h >
47 #18D59h ≤
48 #18DCDh ≥
49 #18E41h =
50 #18EBFh NEG
51 #18F0Bh ABS
52 #18F57h CONJ
53 #18FA3h x
54 #18FD1h MAXR
55 #18FFFh MINR
56 #1902Dh ∞
57 #19058h 1
58 #19089h +
59 #19143h -
60 #191E9h *
61 #192B7h /
62 #19350h ^
63 #1943Fh INV
64 #19488h ARG
65 #194E1h P->R
66 #19609h R->P
67 #19687h SIGN
68 #196C9h √
69 #19738h SQ
70 #197ACh SIN
71 #197EEh COS
72 #19830h TAN
73 #19872h SINH
74 #198B4h COSH
75 #198F6h TANH
76 #19938h ASIN
77 #199B1h ACOS
78 #19A2Ah ATAN
79 #19A6Ch ASINH
80 #19AAEh ACOSH
81 #19B22h ATANH
82 #19B9Bh EXP
83 #19BDDh LN
84 #19C4Ch LOG
85 #19CBBh ALOG
86 #19CFDh LNP1
87 #19D35h EXPM
88 #19D6Dh FACT
89 #19DA5h IP
90 #19DDDh FP
91 #19E15h FLOOR
92 #19E4Dh CEIL
93 #19E85h XPON
94 #19EBDh MAX
95 #19F09h MIN
96 #19F55h MOD
97 #19FA1h MANT
98 #19FD9h D->R
99 #1A011h R->D
100 #1A03Fh ->HMS
101 #1A063h HMS->
102 #1A087h HMS+
103 #1A0ABh HMS-
104 #1A0CFh RNRM
105 #1A0F3h CNRM
106 #1A117h DET
107 #1A13Bh DOT
108 #1A15Fh CROSS
109 #1A183h RSD
110 #1A1B1h %
111 #1A1FDh %T
112 #1A249h %CH
113 #1A28Bh RAND
114 #1A2A5h RDZ
115 #1A2C9h RCL
116 #1A32Eh STO
117 #1A3A2h PURGE
118 #1A416h MEM
119 #1A44Eh ORDER
120 #1A490h CLUSR
121 #1A4AFh KILL
122 #1A4C9h ABORT
123 #1A4E3h ERRN
124 #1A4FDh ERRM
125 #1A517h EVAL
126 #1A54Fh IFTE
127 #1A613h IFT
128 #1A66Eh SYSEVAL
129 #1A6C8h DISP
130 #1A6F6h RND
131 #1A724h BEEP
132 #1A748h ->NUM
133 #1A76Ah LAST

```
134 #1A84Dh WAIT
135 #1A87Ah CLLCD
136 #1A88Bh KEY
137 #1A8A5h CLMF
138 #1A8C9h SF
139 #1A906h CF
140 #1A943h FS?
141 #1A980h FC?
142 #1A9C2h DEG
143 #1A9E6h RAD
144 #1AA0Ah FIX
145 #1AA42h SCI
146 #1AA7Ah ENG
147 #1AAB2h STD
148 #1AACCh FS?C
149 #1AB1Dh FC?C
150 #1AB73h BIN
151 #1AB8Dh DEC
152 #1ABA7h HEX
153 #1ABC1h OCT
154 #1ABDBh STWS
155 #1ABFFh RCWS
156 #1AC1Eh RCLF
157 #1AC3Dh STOF
158 #1ACD9h STOΣ
159 #1AD11h CLΣ
160 #1AD2Bh RCLΣ
161 #1AD45h Σ+
162 #1AD87h Σ-
163 #1ADA1h MΣ
164 #1ADBBh CORR
165 #1ADD5h COV
166 #1ADEFh MAXΣ
167 #1AE09h MEAN
168 #1AE23h MINΣ
169 #1AE3Dh SDEV
170 #1AE57h TOT
171 #1AE71h VAR
172 #1AE8Bh LR
173 #1AEA5h PREDV
174 #1AEC9h UTPC
175 #1AEEDh UTPN
176 #1AF11h UTPF
177 #1AF35h UTPT
178 #1AF59h SCLΣ
179 #1AF73h DRMΣ
180 #1AF8Dh COLΣ
181 #1AFB1h SINV
182 #1B049h SNEG
183 #1B0DFh SCONJ
184 #1B162h STO+
185 #1B27Bh STO-
186 #1B399h STO/
187 #1B4B7h STO*
188 #1B76Fh EXGET
189 #1B793h EXSUB
190 #1B7CBh OBSUB
191 #1B7EFh OBGET
192 #1B813h FORM
193 #1B84Bh COLCT
194 #1B883h EXPAN
195 #1B8BBh ISOL
196 #1B8DFh QUAD
197 #1B903h SHOW
198 #1B931h TAYLR
199 #1B95Fh ∂
200 #1BA00h RCEQ
201 #1BA1Fh STEQ
202 #1BA3Eh ROOT
203 #1BAE4h ∫
204 #1BC5Ch ASR
205 #1BC80h RL
206 #1BCA4h RLB
207 #1BCC8h RR
208 #1BCECh RRB
209 #1BD10h SL
210 #1BD34h SLB
211 #1BD58h SR
212 #1BD7Ch SRB
213 #1BDA0h R->B
214 #1BDC4h B->R
215 #1BDE8h CONVERT
216 #1BE2Ah INDEP
217 #1BE58h PMIN
218 #1BE7Ch PMAX
219 #1BEA0h AXES
220 #1BEC4h CENTR
221 #1BEE8h RES
222 #1BF0Ch *H
223 #1BF30h *W
224 #1BF54h DRAW
225 #1BF73h PIXEL
226 #1BF97h DRAX
227 #1BFB6h PR1
228 #1BFD0h PRSTC
229 #1BFEAh PRST
230 #1C004h CR
231 #1C01Eh PRUSR
232 #1C038h PRVAR
233 #1C05Ch PRMD
234 #1C076h PRLCD
```

### Instructions de type 2

```
0   #2D96h IF
1   #2DABh THEN(IF)
2   #2E0Bh ELSE(IF,IFERR)
3   #2E2Ah END(IF,IFERR)
4   #2E3Fh instr. incon
5   #2E59h ->
6   #2F90h instr. incon
8   #2FECh «
9   #28001h »
12  #28045h WHILE
13  #28064h REPEAT
14  #280A6h END(WHILE)
15  #280C5h DO
16  #280E4h UNTIL
17  #280F9h END(DO)
18  #2813Bh START
19  #28192h FOR
20  #281DDh NEXT
21  #2830Bh STEP
22  #28346h IFERR
23  #2840Dh THEN(IFERR)
24  #28422h HALT
```

### Bytes HP28c 1CC

# 345Ch 41h
# 41FEh A8h
# 440Dh 41h
# 6D0Bh 11h
# 8732h 20h
# 8E4Dh 20h
# 98B5h 2Dh
# A420h 20h
# A698h 2Ch
# AB83h 3Ah
# AD58h 2Eh
# AD5Fh 2Ch
#1139Ch 27h
#114A1h 2Fh
#114B7h 2Ah
#114CDh 2Dh
#114E3h 2Bh
#114F9h 25h
#11592h 5Eh
#115A8h 83h
#1161Fh 84h
#11635h 93h
#1164Bh 87h
#11661h 3Fh
#11677h 2Ch
#1168Dh 2Bh
#116A3h 30h
#116B9h 31h
#116CFh 32h
#116E5h 33h
#116FBh 34h
#11711h 35h
#11727h 36h
#1173Dh 37h
#11753h 38h
#11769h 39h
#11777Fh 88h
#117B6h 8Bh
#13SCAh 20h
#2E61Dh 00h
#3FD36h 0Ah

### Short Integers HP28c 1CC

# 1270h 0000Ch
# 34B8h FFFFFh
# 3F09h 61441h
# 410Ch A8241h
# 4166h 102A8h
# 41C2h E13A8h
# 4CD9h 61441h
# 6C98h 02933h
# 6CA2h 02977h
# 6CACh 02A0Ah
# 6CB6h 02A4Eh
# 6CC0h 02A96h
# 6CCAh 02D12h
# 6CD4h 02911h
# 6CD9h 02911h
# 6CDEh 02C67h
# 6CE8h 02ADAh
# 6CF2h 02AB8h
# 6CFCh 02A70h
# 6D06h 029BFh
# 6D10h 02D37h
# 6D1Ah 02D5Ch
# 6D24h 02955h
# 6D2Eh 0299Dh
# 6D38h 02A2Ch
# 6D42h 02C96h
# 6D4Ch 029E1h
# 6D56h 00000h
# 6D60h 00001h
# 6D6Ah 00002h
# 6D74h 00003h
# 6D7Eh 00004h
# 6D88h 00005h
# 6D92h 00006h
# 6D9Ch 00007h
# 6DA6h 00008h
# 6DB0h 00009h
# 6DBAh 0000Ah
# 6DC4h 0000Bh
# 6DCEh 0000Ch
# 6DD8h 0000Dh
# 6DE2h 0000Eh
# 6DECh 0000Fh
# 6DF6h 00010h
# 6E00h 00011h
# 6E0Ah 00012h
# 6E14h 00013h
# 6E1Eh 00014h
# 6E28h 00015h
# 6E32h 00016h
# 6E3Ch 00017h
# 6E46h 00018h
# 6E50h 00019h
# 6E5Ah 0001Ah
# 6E64h 0001Bh
# 6E6Eh 0001Ch
# 6E78h 0001Dh
# 6E82h 0001Eh
# 6E8Ch 0001Fh
# 6E96h 00020h
# 6EA0h 00021h
# 6EAAh 00022h
# 6EB4h 00023h
# 6EBEh 00024h
# 6EC8h 00025h
# 6ED2h 00026h
# 6EDCh 00027h
# 6EE6h 00028h
# 6EF0h 00029h
# 6EFAh 0002Ah
# 6F04h 0002Bh
# 7SE9h 00106h
# 7SF3h 10080h
# 7SFDh 04080h
# 7607h 00880h
# 7611h 00180h
# 761Bh 00280h
# 7625h 00480h
# 8012h 00EFFh
# 8329h 00040h
# 833Dh 0005Bh
# 8986h 0029Ah
# 990Ch 80000h
# D15Fh 00016h
# D1C1h 00420h
# F080h 00015h
#104B9h 00123h
#104D7h 00122h
#122C8h 00100h
#13FB1h 00820h
#14254h 00120h
#164C6h 00017h
#16775h 00027h
#167F4h 00014h
#16817h 00015h
#1682Bh 00016h
#16925h 00017h
#169DEh 0001Dh
#16DABh 000FAh
#17137h 00054h
#193EEh 00304h
#1A785h 00205h
#1C8DDh 02955h
#1E6B8h 00039h
#1E6C2h 0003Ah
#1E6CCh 0003Bh
#1E6D6h 0003Ch
#1E6E0h 0003Dh
#1E6EAh 0003Eh
#1E6F4h 0003Fh
#1E6FEh 00040h
#24AF8h 00501h
#26B50h 01111h
#2B792h 0011Fh

```
#2E95Ah  00A01h
#2E964h  00A02h
#2E96Eh  00A03h
#2E978h  00A04h
#2E982h  00A05h
#2E98Ch  00A06h
#2F5F0h  0001Fh
#31C43h  00010h
#32EBBh  9FA16h
#3F846h  0002Dh
#3F850h  00030h
#3F85Ah  00032h
#3F864h  00033h
#3F86Eh  00034h
#3F878h  0003Fh
#3F882h  0003Fh
#3F88Ch  00041h
#3F896h  00042h
#3F8A0h  00044h
#3F8AAh  00045h
#3F8B4h  00046h
#3F8BEh  00050h
#3F8C8h  00051h
#3F8D2h  00052h
#3F8DCh  00055h
#3F8E6h  00060h
#3F8F0h  00061h
#3F8FAh  00062h
#3F904h  00064h
#3F90Eh  00065h
#3F918h  0007Fh
#3F922h  0007Fh
#3F92Ch  0007Fh
#3F936h  00105h
#3F940h  00088h
#3F94Ah  00089h
#3F954h  000A1h
#3F95Eh  000A2h
#3F968h  000A6h
#3F972h  000AFh
#3F97Ch  000A9h
#3F986h  000AAh
#3F990h  000B1h
#3F99Ah  000BBh
#3F9A4h  000DDh
#3F9AEh  000DEh
#3F9B8h  000EDh
#3F9C2h  000EEh
#3F9CCh  00100h
#3F9D6h  00111h
#3F9E0h  00133h
#3F9EAh  00136h
#3F9F4h  00163h
#3F9FEh  00166h
```

```
#3FA08h  00151h
#3FA12h  00311h
#3FA1Ch  00444h
#3FA26h  00451h
#3FA30h  00452h
#3FA3Ah  00510h
#3FA44h  00511h
#3FA4Eh  00610h
#3FA58h  00611h
#3FA62h  00651h
#3FA6Ch  00652h
#3FA76h  00851h
#3FA80h  00951h
#3FA8Ah  00A11h
#3FA94h  00A12h
#3FA9Eh  00A15h
#3FAA8h  00A1Ah
#3FAB2h  00A21h
#3FABCh  00A22h
#3FAC6h  00A2Ah
#3FAD0h  00A61h
#3FADAh  00AA1h
#3FAE4h  00AA2h
#3FAEEh  00AAAh
#3FAF8h  FFFFFh
```

### Noms globaux HP28c 1CC

```
#11943h  'constant'
#11978h  'PPAR'
#11D1Dh  'EXPR='
#11E2Ah  'LEFT='
#11F3Fh  'RT='
#12072h  ''
#12079h  'EQ'
#26079h  '∑DAT'
#26088h  '∑PAR'
#27981h  'A'
#27990h  'B'
#2799Fh  'C'
#279AEh  'D'
#279BDh  'E'
#279CCh  'F'
#279DBh  'G'
#279EAh  'H'
#279F9h  'I'
#27A08h  'J'
#27A17h  'K'
#27A26h  'L'
#27A35h  'M'
#27A44h  'N'
```

```
#27A53h  'O'
#27A62h  'P'
#27A71h  'Q'
#27A80h  'R'
#27A8Fh  'S'
#27A9Eh  'T'
#27AADh  'U'
#27ABCh  'V'
#27ACBh  'W'
#27ADAh  'X'
#27AE9h  'Y'
#27AF8h  'Z'
#38A27h  'n0'
#38A45h  's0'
#3A266h  's1'
```

### Noms locaux HP28c 1CC

```
# A2F8h  'a'
# A301h  'b'
# A30Ah  'c'
# A313h  'd'
# A67Ah  'dim2'
# A712h  'dim2'
# A721h  'arr'
# A72Eh  'count'
# A899h  'lf'
#103DAh  ''nohalt'
#10408h  ''halt'
#133AEh  'right'
#133D3h  'right'
#1341Dh  'solvid'
#13430h  'badlist'
#13495h  'solvid'
#134A8h  'badlist'
#13603h  'msg'
#1361Fh  'msg'
#13690h  'solvid'
#136B2h  'badlist'
#136D6h  'badlist'
#1619Ch  'wait?'
#18700h  'x'
#18770h  'x'
#18878h  'nb'
#18883h  'i'
#188A5h  'n'
#1BBBBh  ''int'
#2315Ah  '#b'
#23183h  '#b'
#231C5h  '#a'
#231D0h  '#b'
#231FEh  '#b'
```

```
#2320Eh  '#a'
#2327Bh  '#b'
#2329Ch  '#b'
#232E3h  '#b'
#2330Ch  '#b'
#2843Bh  ''stop'
#2844Ch  ''noname'
#284E3h  'st'
#284EEh  'nfs'
#284FBh  'tok'
#28526h  'st'
#28531h  'nfs'
#2853Eh  'tok'
#29414h  'i'
#2941Dh  'j'
#29449h  'i'
#2945Fh  'j'
#29528h  'j'
#2954Ah  'i'
#2955Dh  'i'
#29E39h  ''nfs'
#29E48h  ''tok'
```

```
#2C93Ah  '#c'
#2CB4Fh  '#c'
#2CB6Eh  '#c'
#2CBC9h  '#c'
#2E616h  ''
#31206h  'fcn'
#31213h  'hex'
#31220h  'k'
#31229h  'i'
#32853h  ''
#34DA1h  ''tcls'
#34DB2h  ''fcls'
#34DF5h  ''tcls'
#34E0Bh  ''fcls'
#34E44h  ''tcls'
#34E55h  ''fcls'
#34F5Fh  ''num'
#34F66h  ''fcn'
#35029h  ''dvar'
#354F5h  ''xSYMfcn'
#355C0h  ''xfcn'
#35E26h  ''nth'
```

```
#35F5Ch  ''scl'
#35F6Bh  ''xSYMfcn'
#35F82h  ''xfcn'
#35FB3h  ''xSYMfcn'
#35FFAh  ''xfcn'
#37B0Eh  ''dv'
#37F53h  ''nm'
#37F60h  ''op'
#389CAh  ''ni'
#389D7h  ''ns'
#38FF9h  ''*s'
#3928Bh  ''+s'
#39F9Fh  ''fl'
#3A222h  ''c'
#3A22Dh  ''b'
#3A239h  ''a'
#3A316h  ''n'
#3A321h  ''prog'
#3A53Fh  ''n'
#3A62Bh  ''n'
#3A6B0h  ''maniplams'
```

---

### Rom pointers HP28c 1CC

```
# 420Ah  5C8C5C2A7A8
# 5B1Bh  41211461441
# 6D1Fh  102955502911
#1C8ECh  E005B71C8F6
#32F38h  4132F46F246
```

---

### Binary Integers HP28c 1CC

```
#2E3A5h  #8h
#2E3B0h  #9h
#2E3BBh  #Ah
#2E3C6h  #Bh
#2E3D1h  #Ch
#2E3DCh  #111h
#2E3E9h  #121h
#2E3F6h  #131h
#2E403h  #141h
#2E410h  #151h
#2E41Dh  #161h
#2E42Ah  #171h
#2E437h  #181h
```

```
#2E444h  #191h
#2E451h  #1A1h
#2E45Eh  #1B1h
#2E46Bh  #1C1h
#2E479h  #1D1h
#2E485h  #1E1h
#2E492h  #1F1h
#30E00h  #0000000000000000h
#30E1Ah  #0000000100000000h
```

### Réels HP28c 1CC

```
#1E3D6h   +0.000000000000E0
#1E3EBh   +1.000000000000E0
#1E400h   +2.000000000000E0
#1E415h   +3.000000000000E0
#1E42Ah   +4.000000000000E0
#1E43Fh   +5.000000000000E0
#1E454h   +6.000000000000E0
#1E469h   +7.000000000000E0
#1E47Eh   +8.000000000000E0
#1E493h   +9.000000000000E0
#1E4A8h   -1.000000000000E0
#1E4BDh   -2.000000000000E0
#1E4D2h   -3.000000000000E0
#1E4E7h   -4.000000000000E0
#1E4FCh   -5.000000000000E0
#1E511h   -6.000000000000E0
#1E526h   -7.000000000000E0
#1E53Bh   -8.000000000000E0
#1E550h   -9.000000000000E0
#1E565h   +3.14159265359E0
#3FB41h   +2.7182818284E0
```

### Réels étendus HP28c 1CC

```
#1E57Ah   +3.141592653589793000E0
#1E5E8h   +0.000000000000000000E0
#1E602h   +1.000000000000000000E0
#1E61Ch   +2.000000000000000000E0
#1E636h   +3.000000000000000000E0
#1E650h   +4.000000000000000000E0
#1E66Ah   +5.000000000000000000E0
#1FBB6h   +7.000000000000000000E0
#1FC96h   -1.211428571428571000E0
#1FCB0h   +9.335849056603770000E0
#2526Dh   +2.320000000000000000E0
#2528Fh   +3.500000000000000000E0
#25FA3h   +3.300000000000000000E0
#30E4Eh   +1.000000000000000000E0
#33130h   +2.302585092994050000E0
```

### Chaînes HP28c 1CC

```
# 39D9h   ""
# 7EE7h   " Low Memory!"
# 917Fh   "<>"}]]≤≥->∑*µ"
# 92A5h   "  «=lαγz#{[[STUVWX
           MNOPQRGHIJKLABCDEF"
# 9B07h   "SHIFT"
# 9C59h   "VISIT"
# 9D35h   "ENTER"
# 9115h   "COMMAND"
# 917Dh   "CHS"
# 91C9h   "EEX"
# 93B0h   "INS"
# 93C0h   "DEL"
# 93D0h   "UP"
# 93DEh   "DOWN"
# 93F0h   "LEFT"
# 9402h   "RIGHT"
# 99B6h   "E"
# 99C2h   "1E"
# A907h   "   "(3 espaces)
# A8B8h   "("
# A8F6h   ")"
# A9FCh   "[Empty Stack]"
# ABA8h   "User Variables:"
# AC20h   "No User Variables"
# ACA1h   "Format "
# ACC3h   " Base "
# ACF7h   "RADIANS"
# AD0Fh   "DEGREES"
# AD36h   " Radix "
# AD75h   "Undo"
# AD9Bh   "Command"
# ADC7h   "Last"
# ADE3h   "  Multiline"
# AEF3h   " ON "
# AF05h   " OFF"
# AFFFh   "
          "[CHR(10)]
           [NEWLINE]
# C0E4h   "CURSOR"
# C3CAh   "EDIT"
# D1AAh   "BACK"
# E326h   "CLUSR"
# E33Ah   "ORDER"
# E34Eh   "MEM"
# E534h   "NEXT"
# E8ACh   "USER"
# EB0Ch   "TRIG"
# EC4Fh   "SOLV"
# F26Ah   "PPAR"
# F2A9h   "DRW∑"
```

```
# F4F5h  "STD"
# F51Eh  "FIX"
# F547h  "SCI"
# F570h  "ENG"
# F599h  "DEG"
# F5B0h  "RAD"
# F5E1h  "RDX."
# F616h  "RDX,"
# F650h  "+CMD"
# F685h  "-CMD"
# F6BFh  "+LAST"
# F6F6h  "-LAST"
# F732h  "+UNDO"
# F769h  "-UNDO"
# F7A5h  "+ML"
# F7D8h  "-ML"
# F888h  "DEC"
# F8B1h  "HEX"
# F8DAh  "OCT"
# F903h  "BIN"
# F945h  "SSI"
# F98Ch  "HALT"
# FA11h  "TRACE"
# FA40h  "NORM"
# FA70h  " "
# FA89h  "  "
# FB51h  "STO"
# FC56h  "EVAL"
# FCDBh  "CONTINUE"
# FD0Eh  "UNDO"
#11381h  "'"
#113B2h  "CONVERT"
#113D9h  "CLEAR"
#113FCh  "SWAP"
#1141Dh  "STO"
#11143h  "PURGE"
#1145Fh  "ROLL"
#11480h  "DROP"
#11528h  "%CH"
#1154Ch  "INV"
#11570h  "SQ"
#115BEh  "LAST"
#115DFh  "RCL"
#115FEh  "EVAL"
#11795h  "->NUM"
#11B4Ah  " Not In
          Equation"
#11B8Dh  "Constant
          Equation"
#11BC3h  "Using "
#1204Ah  "DRAW"
#12F4Bh  "SOLVR"
#13100h  " "
#13283h  ": "
#13327h  "EXPR="
#13360h  "LEFT="
#133EEh  "RIGHT="
#13508h  "SOLVING FOR "
#13DAEh  "2: "
#13DCDh  "3: "
#13DDDh  "4: "
#13F55h  "1: "
#13F88h  "   "
#14498h  "No Room to
          Show Stack"
#153DBh  "}"
#154DCh  "{"
#15A7Ah  " | "
#15B5Ch  "[| "
#15B8Fh  "]]"
#166BBh  " USAGE: "
#29950h  "HALT"
#29962h  "ELSE"
#2997Ah  "END"
#29984h  "UNTIL"
#29998h  "REPEAT"
#299AEh  "NEXT"
#299C0h  "STEP"
#299D2h  "IF"
#299E0h  "DO"
#299EEh  "IFERR"
#29A02h  "START"
#29A16h  "FOR"
#29A26h  "THEN"
#29A38h  "->"
#29A44h  "WHILE"
#2BC24h  "+  "
#2BC34h  "-  "
#2BC4Eh  "?  "
#2F35Bh  "  "
#2FB94h  "Kg"
#2FBA2h  "m"
#2FBAEh  "A"
#2FBBAh  "s"
#2FBC6h  "oK"
#2FBD4h  "cd"
#2FBE2h  "mol"
#2FBF2h  "?"
#3ADFBh  "[<-]"
#3AE4Ch  "[->]"
#3AE66h  "EXGET"
#3B00Ah  "DNEG"
#3B049h  "DINV"
#3B088h  "*1"
#3B0C3h  "^1"
#3B0FEh  "/1"
#3B139h  "+1-1"
#3B1F5h  "EXPAN"
#3B236h  "COLCT"
#3B277h  "<-->"
#3B29Eh  "<-A"
#3B2C5h  "A->"
#3B2ECh  "AF"
#3B313h  "M->"
#3B33Ah  "<-M"
#3B361h  "-()"
#3B38Ah  "1/()"
#3B3B5h  "E()"
#3B3DEh  "L()"
#3B407h  "L*"
#3B42Eh  "E^"
#3B455h  "->()"
#3B47Eh  "<-D"
#3B4A5h  "D->"
#3EFF3h  "
           "(23 espaces)
#3FBBCh  "YES"
#3FBCCh  "NO"
#3FBDAh  "Purge?"
#3FBF0h  "Out Of Memory"
#3FC14h  "Stack"
#3FC28h  "UNDO Stack"
#3FC46h  "Command Stack"
#3FC6Ah  "Last Arguments"
#3FC90h  "System Object"
#3FCB4h  "]"
#3FCC0h  "| "
#3FCCEh  "|"
#3FCDAh  "{"
#3FCE6h  "}"
#3FCF2h  "#"
#3FCFEh  "»"
#3FD0Ah  "«"
#3FD16h  "     "
         (5 espaces)
#3FD2Ah  "
         "(CHR(10))
          (NEWLINE)
#3FD3Dh  "der"
#3FD4Dh  "IFTE"
#3FD5Fh  "NOT"
#3FD6Fh  "NOT "
#3FD81h  "AND"
#3FD91h  " AND "
#3FDA5h  "XOR"
#3FDB5h  " XOR "
#3FDC9h  "OR"
#3FDD7h  " OR "
#3FDE9h  " "
#3FDF5h  " '"
#3FE03h  "LEVEL "
#3FE19h  "UNKNOWN"
#3FE31h  "*"
#3FE3Dh  "'"
```

```
#3FE49h  ","
#3FE55h  "."
#3FE61h  "("
#3FE6Dh  ")"
#3FE79h  "^"
#3FE85h  "*"
#3FE91h  "/"
#3FE9Dh  "+"
#3FEA9h  "-"
#3FEB5h  "="
#3FEC1h  "=="
#3FECFh  "<"
#3FEDBh  ">"
#3FEE7h  "√"
#3FEF3h  "∂"
#3FEFFh  "≤"
#3FF0Bh  "≥"
#3FF17h  "≠"
#3FF23h  "1"
#3FF2Fh  "2"
#3FF3Bh  "3"
#3FF47h  "4"
#3FF53h  "5"
#3FF5Fh  "6"
#3FF6Bh  "7"
#3FF77h  "8"
#3FF83h  "9"
#3FF8Fh  " Version 1CC"
```

### Complexes HP28c 1CC

```
#119D7h  (-6.800000000000E0,-1.500000000000E0)
#119FCh  (+6.800000000000E0,+1.600000000000E0)
#11A26h  (+0.000000000000E0,+0.000000000000E0)
#3302Eh  (+0.000000000000E0,+0.000000000000E0)
#33082h  (+1.000000000000E0,+0.000000000000E0)
#330A7h  (-1.000000000000E0,+0.000000000000E0)
#330CCh  (+0.000000000000E0,+1.000000000000E0)
#330F1h  (+0.000000000000E0,-1.000000000000E0)
#3FB80h  (+0.000000000000E0,+0.000000000000E0)
```

---

### Complexes étendus HP28c 1CC

```
#119D7h  (+0.000968000000000E0,+0.000029770915000E0)
#33082h  (+0.000010000000000E0,+0.000029770000000E0)
#330A7h  (+0.000910000000000E0,+0.000029770000000E0)
#330CCh  (+0.000000000000000E0,+0.000029770100000E0)
#330F1h  (+0.000000000000000E0,-9.999029559100000E0)
#3FB80h  (+0.000000000000000E0,+0.202933200000000E0)
```

---

### Instructions de type 1

0 # 75DCh DUP
1 # 75F6h DUP2
2 # 7610h SWAP
3 # 762Ah DROP
4 # 7644h DROP2
5 # 765Eh ROT
6 # 7678h OVER
7 # 7692h DEPTH
8 # 76B1h DROPN
9 # 76CBh DUPN
10 # 76E5h PICK
11 # 76FFh ROLL
12 # 7719h ROLLD
13 # 7733h CLEAR
14 # 7752h ->LIST
15 # 776Ch R->C
16 # 77A4h RE
17 # 77F0h IM
18 # 7832h SUB
19 # 7888h LIST->
20 # 78C0h C->R
21 # 78EEh SIZE
22 # 796Ch POS
23 # 79AEh ->STR
24 # 79C8h STR->
25 # 79ECh NUM
26 # 7A10h CHR
27 # 7A34h TYPE
28 # 7AE4h ->ARRY
29 # 7B85h ARRY->
30 # 7BD6h RDM
31 # 7C5Eh CON
32 # 7D4Ah IDN
33 # 7DE1h TRN
34 # 7E32h PUT
35 # 7FB9h PUTI
36 # 8149Ah GET
37 # 8277h GETI
38 # 87C3h SAME
39 # 87ECh AND
40 # 886Fh OR
41 # 88F2h NOT
42 # 8952h XOR
43 # 89D5h ==
44 # 8A53h ≠
45 # 8AD6h <
46 # 8B4Ah >
47 # 8BBEh ≤
48 # 8C32h ≥
49 # 8CA6h =
50 # 8D24h NEG
51 # 8D70h ABS
52 # 8DBCh CONJ
53 # 8E08h π
54 # 8E36h MAXR
55 # 8E64h MINR
56 # 8E92h ∞
57 # 8EC0h 1
58 # 8EEEh +
59 # 8FD0h -
60 # 9076h *
61 # 9144h /
62 # 91FEh ^
63 # 92DBh INV
64 # 9327h ARG
65 # 9370h P->R
66 # 9469h R->P
67 # 94E2h SIGN
68 # 9524h √
69 # 958Eh SQ
70 # 9602h SIN
71 # 9644h COS
72 # 9686h TAN
73 # 96C8h SINH
74 # 970Ah COSH
75 # 974Ch TANH
76 # 978Eh ASIN
77 # 9802h ACOS
78 # 9876h ATAN
79 # 98B8h ASINH
80 # 98FAh ACOSH
81 # 9969h ATANH
82 # 99DDh EXP
83 # 9A1Fh LN
84 # 9A89h LOG
85 # 9AF3h ALOG
86 # 9B35h LNP1
87 # 9B6Dh EXPM
88 # 9BA5h FACT
89 # 9BDDh IP
90 # 9C15h FP
91 # 9C4Dh FLOOR
92 # 9C85h CEIL
93 # 9CBDh XPON
94 # 9CF5h MAX
95 # 9D41h MIN
96 # 9D8Dh MOD
97 # 9DD9h MANT
98 # 9E11h D->R
99 # 9E49h R->D
100 # 9E77h ->HMS
101 # 9E9Bh HMS->
102 # 9EBFh HMS+
103 # 9EE3h HMS-
104 # 9F07h RNRM
105 # 9F2Bh CNRM
106 # 9F4Fh DET
107 # 9F73h DOT
108 # 9F97h CROSS
109 # 9FBBh RSD
110 # 9FE9h %
111 # A035h %T
112 # A081h %CH
114 # A0DDh RDZ
115 # A101h COMB
115 # A101h RAND
116 # A125h PERM
117 # A149h LCD->
118 # A163h ->LCD
119 # A187h DGTIZ
120 # A1A1h MENU
121 # A1CFh RCL
122 # A220h STO
123 # A26Ch PURGE
124 # A32Bh MEM
125 # A359h ORDER
126 # A396h CLUSR
127 # A3B0h KILL
128 # A3CAh ABORT
129 # A3E4h ERRN
130 # A3FEh ERRM
131 # A418h EVAL
132 # A450h IFTE
133 # A500h IFT

```
134 # A54Ch SYSEVAL
135 # A5A6h DISP
136 # A5D4h RND
137 # A616h BEEP
138 # A63Ah ->NUM
139 # A654h LAST
140 # A73Ah WAIT
141 # A75Eh CLLCD
142 # A778h KEY
143 # A792h CLMF
144 # A7B6h SF
145 # A7F3h CF
146 # A830h FS?
147 # A86Dh FC?
148 # A8AFh DEG
149 # A8D3h RAD
150 # A8F7h FIX
151 # A92Fh SCI
152 # A967h ENG
153 # A99Fh STD
154 # A9B9h FS?C
155 # AA0Ah FC?C
156 # AA60h BIN
157 # AA7Ah DEC
158 # AA94h HEX
159 # AAAEh OCT
160 # AAC8h STWS
161 # AAECh RCWS
162 # AB06h RCLF
163 # AB20h STOF
164 # ABBCh STOΣ
165 # ABD6h CLΣ
166 # ABF0h RCLΣ
167 # AC0Ah Σ+
168 # AC38h Σ-
169 # AC52h MΣ
170 # AC6Ch CORR
171 # AC86h COV
172 # ACA0h MAXΣ
173 # ACBAh MEAN
174 # ACD4h MINΣ
175 # ACEEh SDEV
176 # AD08h TOT
177 # AD22h VAR
178 # AD3Ch LR
179 # AD56h PREDV
180 # AD7Ah UTPC
181 # AD9Eh UTPN
182 # ADC2h UTPF
183 # ADE6h UTPT
184 # AE0Ah SCLΣ
185 # AE24h DRWΣ
186 # AE3Eh COLΣ
187 # AE62h SINV
188 # AEF9h SNEG
189 # AF90h SCONJ
190 # B013h STO+
191 # B118h STO-
192 # B222h STO/
193 # B34Fh STO*
194 # B5C6h EXGET
195 # B5EAh EXSUB
196 # B622h OBSUB
197 # B646h OBGET
198 # B66Ah FORM
199 # B6A2h COLCT
200 # B6DAh EXPAN
201 # B712h ISOL
202 # B736h QUAD
203 # B75Ah SHOW
204 # B788h TAYLR
205 # B7B6h ∂
206 # B84Dh RCEQ
207 # B871h STEQ
208 # B88Bh ROOT
209 # B8E1h ∫
210 # BB53h ASR
211 # BB77h RL
212 # BB9Bh RLB
213 # BBBFh RR
214 # BBE3h RRB
215 # BC07h SL
216 # BC2Bh SLB
217 # BC4Fh SR
218 # BC73h SRB
219 # BC97h R->B
220 # BCBBh B->R
221 # BCDFh CONVERT
222 # BD21h INDEP
223 # BD45h PMIN
224 # BD69h PMAX
225 # BD8Dh AXES
226 # BDB1h CENTR
227 # BDD5h RES
228 # BDF9h *H
229 # BE1Dh *W
230 # BE41h DRAW
231 # BE5Bh PIXEL
232 # BE7Fh DRAX
233 # BE99h PR1
234 # BEB3h PRSTC
235 # BECDh PRST
236 # BEE7h CR
237 # BF01h PRUSR
238 # BF1Bh PRVAR
239 # BF5Dh PRMD
240 # BF77h PRLCD
241 # BF91h CRDIR
242 # BFB5h PATH
243 # BFCFh HOME
244 # BFE9h VARS
```

### Instructions de type 2

```
0   # E3E2h IF
1   # E3F8h THEN(IF)
2   # E459h ELSE(IF,IFERR)
3   # E479h END(IF,IFERR)
4   # E48Fh instr. incon
5   # E4D2h WHILE
6   # E4F2h REPEAT
7   # E535h DO
8   # E555h UNTIL
9   # E56Bh START
10  # E5B3h FOR
11  # E60Fh NEXT
12  # E673Eh STEP
13  # E77Ah IFERR
14  # E842h HALT
15  # E862h instr. incon
17  # E9B5h ->
18  # E9D0h «
19  # E9E6h »
22  # EA2Dh END(WHILE)
23  # EA4Dh END(DO)
24  # EA90h THEN(IFERR)
```

**Bytes HP28s 2BB**

```
# 27A2h  A8h
# 3B02h  41h
# 4462h  41h
# 715Eh  11h
#17125h  00h
#19696h  20h
#19DE7h  20h
#1A6CCh  3Ah
#1AB2Ah  2Eh
#1AB31h  2Ch
#1E4DAh  20h
#20BD2h  2Dh
#23CD5h  27h
#23DDAh  2Fh
#23DF0h  2Ah
#23E06h  2Dh
#23E1Ch  2Bh
#23E32h  25h
#23ECBh  5Eh
#23EE1h  83h
#23F58h  84h
#23F6Eh  93h
#23F84h  87h
#23F9Ah  3Fh
#23FB0h  2Ch
#23FC6h  2Eh
#23FDCh  30h
#23FF2h  31h
#24008h  32h
#2401Eh  33h
#24034h  34h
#2404Ah  35h
#24060h  36h
#24076h  37h
#2408Ch  38h
#240A2h  39h
#240B8h  88h
#240EFh  8Bh
#31AE9h  62h
#31AFAh  6Fh
#31B10h  64h
#31B17h  69h
#3E9F0h  20h
#3F0CDh  0Ah
```

**Short Integers HP28s 2BB**

```
# 11D4h  0000Ch
# 24ADh  61441h
# 26B0h  A8241h
# 270Ah  102A8h
# 2766h  E13A8h
# 3B5Eh  FFFFFh
# 4E78h  61441h
# 6B87h  007FFh
# 70EBh  02933h
# 70F5h  02977h
# 70FFh  02A0Ah
# 7109h  02A4Eh
# 7113h  02A96h
# 711Dh  02D12h
# 7127h  02911h
# 712Ch  02911h
# 7131h  02C67h
# 713Bh  02ADAh
# 7145h  02AB8h
# 714Fh  02A70h
# 7159h  029BFh
# 7163h  02D37h
# 716Dh  02D5Ch
# 7177h  02955h
# 7181h  0299Dh
# 718Bh  02A2Ch
# 7195h  02C96h
# 719Fh  029E1h
# 71A9h  00000h
# 71BDh  00002h
# 71C7h  00003h
# 71D1h  00004h
# 71DBh  00005h
# 71E5h  00006h
# 71EFh  00007h
# 71F9h  00008h
# 7203h  00009h
# 720Dh  0000Ah
# 7217h  0000Bh
# 7221h  0000Ch
# 722Bh  0000Dh
# 7235h  0000Eh
# 723Fh  0000Fh
# 7249h  00010h
# 7253h  00011h
# 725Dh  00012h
# 7267h  00013h
# 7271h  00014h
# 727Bh  00015h
# 7285h  00016h
# 728Fh  00017h
# 7299h  00018h
# 72A3h  00019h
# 72ADh  0001Ah
# 72B7h  0001Bh
# 72C1h  0001Ch
# 72CBh  0001Dh
# 72D5h  0001Eh
# 72DFh  0001Fh
# 72E9h  00020h
# 72F3h  00021h
# 72FDh  00022h
# 7307h  000A3h
# 7311h  00024h
# 731Bh  00025h
# 7325h  00026h
# 732Fh  00027h
# 7339h  00028h
# 7343h  00029h
# 734Dh  0002Ah
# 7357h  0002Bh
# 83EEh  0002Eh
# 928Fh  00304h
# A672h  00205h
# C893h  02955h
#11596h  00039h
#115A0h  0003Ah
#115AAh  0003Bh
#115B4h  0003Ch
#115BEh  0003Dh
#115C8h  0003Eh
#115D2h  0003Fh
#115DCh  00040h
#14FACh  00A05h
#15006h  00A06h
#15024h  0011Fh
#15042h  00A01h
#15056h  00A02h
#1510Ah  00A04h
#156E9h  00A03h
#192C0h  00106h
#192CAh  0010Fh
#192D4h  00280h
#192DEh  08080h
#192E8h  20080h
#192F2h  00480h
#192FCh  00880h
#19306h  01080h
#19310h  00080h
#1931Ah  00044h
#19324h  08040h
#19E7Bh  00EFFh
#19F02h  00104h
#1922Dh  00040h
#19241h  0005Bh
```

```
#1CCF6h  00016h
#1CD58h  01020h
#1F24Bh  00054h
#207AEh  000FEh
#20C24h  80000h
#22DE7h  00123h
#22E05h  00122h
#24C4Ch  00100h
#25FB3h  20020h
#26242h  00420h
#29C96h  01111h
#2D89Ch  00501h
#2E75Ah  0001Fh
#2FFA6h  000FBh
#3E85Fh  DFA16h
#3EB96h  0002Dh
#3EBA0h  00030h
#3EBAAh  00032h
#3EBB4h  00033h
#3EBBEh  00034h
#3EBC8h  00035h
#3EBD2h  00037h
#3EBDCh  00041h
#3EBE6h  00042h
#3EBF0h  00043h
#3EBFAh  00044h
#3EC04h  00045h
#3EC0Eh  00046h
#3EC18h  00050h
#3EC22h  00051h
#3EC2Ch  00052h
#3EC36h  00053h
#3EC40h  00054h
#3EC4Ah  00055h
#3EC54h  00056h
#3EC5Eh  00060h
#3EC68h  00061h
#3EC72h  00062h
#3EC86h  00065h
#3EC90h  00070h
#3EC9Ah  00071h
#3ECA4h  00075h
#3ECAEh  00105h
#3ECB8h  00088h
#3ECC2h  00089h
#3ECCCh  000A1h
#3ECD6h  000A2h
#3ECE0h  000A6h
#3ECEAh  000A7h
#3ECF4h  000A9h
#3ECFEh  000AAh
#3ED08h  000B1h
#3ED12h  000BBh
#3ED1Ch  000DDh
#3ED26h  000DEh
#3ED30h  000EDh
#3ED3Ah  000EEh
#3ED44h  00100h
#3ED4Eh  00111h
#3ED58h  00133h
#3ED62h  00136h
#3ED6Ch  00163h
#3ED76h  00166h
#3ED80h  00151h
#3ED8Ah  00311h
#3ED94h  00411h
#3ED9Eh  00412h
#3EDA8h  00444h
#3EDB2h  00451h
#3EDBCh  00452h
#3EDC6h  00510h
#3EDD0h  00511h
#3EDDAh  00550h
#3EDE4h  00610h
#3EDEEh  00611h
#3EDF8h  00650h
#3EE02h  00851h
#3EE0Ch  00861h
#3EE16h  00862h
#3EE20h  00865h
#3EE2Ah  00A11h
#3EE34h  00A12h
#3EE3Eh  00A15h
#3EE48h  00A1Ah
#3EE52h  00A21h
#3EE5Ch  00A22h
#3EE66h  00A2Ah
#3EE70h  00A51h
#3EE84h  00A62h
#3EE8Eh  00A65h
#3EE98h  00AA1h
#3EEA2h  00AA2h
#3EEACh  00AAAh
#3EEB6h  FFFFFh
#3F5B5h  00017h
#3F7ADh  00027h
#3F92Ch  00014h
#3F94Fh  00015h
#3F963h  00016h
#3F92Bh  00017h
#3F9C1h  0001Dh
#3FC75h  00104h
```

### Noms globaux HP28s 2BB

```
#1DFB6h  'EXPR='
#1DFC7h  'LEFT='
#1DFD8h  'RT='
#1E791h  ''
#1E798h  'EQ'
#24277h  'constant'
#242ACh  'PPAR'
#296ABh  '∑DAT'
#29BF6h  '∑PAR'
#36859h  'n0'
#36876h  's0'
#38095h  's1'
```

### Noms locaux HP28s 2BB

```
# EA9Fh  ''stop'
# EAB0h  ''noname'
# EB4Ch  'st'
# EB57h  'ofs'
# EB64h  'tok'
# EB8Fh  'st'
# EB9Ah  'ofs'
# EBA7h  'tok'
# FA87h  'i'
# FA90h  'j'
# FABCh  'i'
# FACAh  'j'
# FB9Bh  'j'
# FBBDh  'i'
# FBD0h  'i'
#10553h  ''ofs'
#10562h  ''tok'
#161D0h  '#c'
#1669Ah  '#c'
#166B9h  '#c'
#16714h  '#c'
#16814h  ''
#1A89Eh  '!46'
#1E154h  'a'
#1E15Dh  'b'
#1E2CDh  'right'
#1E2F2h  'right'
#1E33Ch  'solvid'
#1E34Fh  'badlist'
#1E3AAh  'solvid'
#1E3BDh  'badlist'
#1E513h  'msg'
#1E52Fh  'msg'
#1E59Bh  'solvid'
```

```
#1E5B0h  'badlist'
#1E5E1h  'badlist'
#22D12h  ''nohalt'
#22D40h  ''halt'
#2C3E0h  '#a'
#2C3F8h  '#b'
#2C42Bh  '#b'
#2C43Bh  '#a'
#2C4EBh  '#b'
#2C519h  '#b'
#32B32h  ''tcls'
#32B43h  ''fcls'
#32BD1h  ''tcls'
#32BE7h  ''fcls'
#32C20h  ''tcls'
#32C31h  ''fcls'
#32F54h  ''num'
#32F63h  ''fcn'
#33026h  ''dvar'
#33347Fh  ''xSYMfcn'
#33496h  ''xfcn'
#33DBFh  ''nth'
#33EF0h  ''scl'
#33EFFh  ''xSYMfcn'
#33F16h  ''xfcn'
#33F77h  ''xSYMfcn'
#33F8Eh  ''xfcn'
#35930h  ''dv'
#35D75h  ''nm'
#35D82h  ''op'
#367FBh  ''ni'
#36809h  ''ns'
#36E3Eh  ''*s'
#36E3Eh  ''+s'
#37DC4h  ''fl'
#38051h  ''c'
#3805Ch  ''b'
#38067h  ''a'
#38145h  ''n'
#38150h  ''prog'
#38382h  ''n'
#38482h  ''n'
#38514h  ''maniplams'
#3E0EAh  ''
```

### Rom pointers HP28s 2BB

```
# 27AEh  5C8C5C2A7A8
# 5D59h  41211461441
# 7172h  10295502911
# C8A2h  E005B70C8AC
#3E8DCh  413E8EAF246
```

### Binary Integers HP28s 2BB

```
#17970h  #00000h
#1797Fh  #00001h
#1798Eh  #00002h
#1799Dh  #00003h
#179ACh  #00004h
#2FEF0h  #0000000000000000h
#2FF0Ah  #0000000100000000h
```

### Réels HP28s 2BB

```
#112B4h  +0.000000000000E0
#112C9h  +1.000000000000E0
#112DEh  +2.000000000000E0
#112F3h  +3.000000000000E0
#11308h  +4.000000000000E0
#1131Dh  +5.000000000000E0
#11332h  +6.000000000000E0
#11347h  +7.000000000000E0
#1135Ch  +8.000000000000E0
#11371h  +9.000000000000E0
#11386h  -1.000000000000E0
#1139Bh  -2.000000000000E0
#113B0h  -3.000000000000E0
#113C5h  -4.000000000000E0
#113DAh  -5.000000000000E0
#113EFh  -6.000000000000E0
#11404h  -7.000000000000E0
#11419h  -8.000000000000E0
#1142Eh  -9.000000000000E0
#11443h  +3.14159265359E0
#2BFC8h  +3.000000000000E0
#3EEFFh  +2.71828182846E0
```

### Réels étendus HP28s 2BB

```
#11458h  +3.14159265358979000E0
#114C6h  +0.00000000000000000E0
#114E0h  +1.00000000000000000E0
#114FAh  +2.00000000000000000E0
#11514h  +3.00000000000000000E0
#1152Eh  +4.00000000000000000E0
#11548h  +5.00000000000000000E0
#12731h  +7.00000000000000000E0
#129C2h  +9.33584905660377000E0
#129EBh  -1.21142857142857000E0
#29499h  +2.32000000000000000E0
#294B3h  +3.50000000000000000E0
#2926Ch  +3.30000000000000000E0
#2FF3Eh  +1.00000000000000000E0
#30C92h  +2.30258509299405000E0
```

## Chaînes HP28s 2BB

```
# 407Fh  ""
# FFCBh  "HALT"
# FFDDh  "ELSE"
# FFEFh  "END"
# FFFFh  "UNTIL"
#10013h  "REPEAT"
#10029h  "NEXT"
#1003Bh  "STEP"
#1004Dh  "IF"
#1005Bh  "DO"
#10069h  "IFERR"
#1007Dh  "START"
#10091h  "FOR"
#100A1h  "THEN"
#100B3h  "->"
#100BFh  "WHILE"
#102D5h  "bodh"
#154CAh  "+  "
#154DAh  "-  "
#154F4h  "?  "
#18D50h  " Low Memory!"
#1903Dh  "<>"}]]≥≤->∑°µ"
#191BDh  "  «=lαyz#[[(STUVWX
         MNOPQRGHIJKLABCDEF"
#19A1Ah  "SHIFT"
#19BADh  "VISIT"
#19CB1h  "ENTER"
#1A08Ch  "COMMAND"
#1A0EAh  "CHS"
#1A136h  "EEX"
#1A318h  "INS"
#1A328h  "DEL"
#1A338h  "UP"
#1A346h  "DOWN"
#1A358h  "LEFT"
#1A36Ah  "RIGHT"
#1A4E1h  "[Empty Stack]"
#1A987h  "    "(4 espaces)
#1A9ADh  "Row "
#1AA73h  "Format "
#1AA95h  " Base "
#1AAC9h  "RADIANS"
#1AAE1h  "DEGREES"
#1AB08h  " Radix "
#1AB47h  "Undo"
#1AB6Dh  "Command"
#1AB99h  "Last"
#1ABB5h  "  Multiline"
#1ACCFh  " ON "
#1ACE1h  " OFF"
#1BCCBh  "CURSOR"
#1BFA7h  "EDIT"
#1CD41h  "BACK"
#1DE30h  "SOLVR"
#1DFE5h  " "
#1E1A2h  ": "
#1E246h  "EXPR="
#1E28Ch  "LEFT="
#1E30Dh  "RIGHT="
#1E413h  "SOLVING FOR "
#1F290h  "Custom Menu"
#20260h  "NEXT"
#20CB0h  "E"
#20CBCh  "1E"
#20E8Ch  " Error:"
#21310h  "USER"
#2154Ch  "TRIG"
#21681h  "SOLV"
#21CC6h  "PPAR"
#21D0Fh  "DRW∑"
#21F06h  "STD"
#21F2Fh  "FIX"
#21F58h  "SCI"
#21F81h  "ENG"
#21FAAh  "DEG"
#21FCEh  "RAD"
#21FF2h  "RDX,"
#22045h  "CMD"
#22073h  "LAST"
#220A3h  "UNDO"
#220D8h  "ML"
#2217Ch  "DEC"
#221A5h  "HEX"
#221CEh  "OCT"
#221F7h  "BIN"
#22239h  "SST"
#22280h  "HALT"
#222F6h  "TRAC"
#22344h  " "(2 espaces)
#22350h  "  "(3 espaces)
#22477h  "CLUSR"
#224A9h  "STO"
#225AEh  "EVAL"
#22633h  "CONTINUE"
#22666h  "UNDO"
#23CBAh  "'"
#23CEBh  "CONVERT"
#23D12h  "CLEAR"
#23D35h  "SWAP"
#23D56h  "STO"
#23D75h  "PURGE"
#23D98h  "ROLL"
#23DB9h  "DROP"
#23E61h  "%CH"
#23E85h  "INV"
```

```
#23EA9h  "SQ"
#23EF7h  "LAST"
#23F18h  "RCL"
#23F37h  "EVAL"
#240CEh  "->NUM"
#24474h  " Not In Equation"
#244BCh  "Constant Equation"
#244E0h  "Using "
#249D3h  "DRAW"
#25DABh  "2: "
#25DCAh  "3: "
#25DDAh  "4: "
#25F4Dh  "1: "
#25F8Ah  "   "
#2647Ch  "No Room to Show Stack"
#26569h  "Directory"
#26CE8h  "}"
#26DF8h  "{"
#27634h  " [ "
#27740h  "[[ "
#27780h  "]]"
#2E7D7h  "  " (2 espaces)
#2EDBAh  "kg"
#2EDC8h  "m"
#2EDD4h  "A"
#2EDE0h  "s"
#2EDECh  "°K"
#2EDFAh  "cd"
#2EE08h  "mol"
#2EE18h  "?"
#38C75h  "{<-}"
#38CC6h  "{->}"
#38CE0h  "EXGET"
#38E93h  "DNEG"
#38ED2h  "DINV"
#38F11h  "*1"
#38F4Ch  "^1"
#38F87h  "/1"
#38FC2h  "+1-1"
#39007Eh  "EXPAN"
#390BFh  "COLCT"
#39100h  "<-->"
#39127h  "A"
#3914Eh  "A->"
#39175h  "AF"
#3919Ch  "M->"
#391C3h  "<-M"
#391EAh  "-()"
#39213h  "1/()"
#3923Eh  "E()"
#39267h  "L()"
#39290h  "L*"
#392B7h  "E^"
#392DEh  "->()"
#39307h  "<-D"
#3932Eh  "D->"
#3CEDCh  "
          "(23 espaces)
#3EF53h  "YES"
#3EF63h  "NO"
#3EF71h  "Purge?"
#3EF87h  "Out of Memory"
#3EFABh  "Stack"
#3EFBFh  "UNDO Stack"
#3EFDDh  "Command Stack"
#3F001h  "Last Arguments"
#3F027h  "System Object"
#3F04Bh  "]"
#3F057h  "[ "
#3F065h  "["
#3F071h  "("
#3F07Dh  ")"
#3F089h  "#"
#3F095h  "»"
#3F0A1h  "«"
#3F0ADh  "     "(5 espaces)
#3F0C1h  "
          "[CHR(10)]
           [NEWLINE]
#3F0D4h  "der"
#3F0E4h  "IFTE"
#3F0F6h  "NOT"
#3F106h  "NOT "
#3F118h  "AND"
#3F128h  " AND "
#3F13Ch  "XOR"
#3F14Ch  " XOR "
#3F160h  "OR"
#3F16Eh  " OR "
#3F180h  " "
#3F18Ch  "LEVEL "
#3F1A2h  "UNKNOWN"
#3F1BAh  ""
#3F1C6h  "'"
#3F1D2h  ","
#3F1DEh  "."
#3F1EAh  "("
#3F1F6h  ")"
#3F202h  "^"
#3F20Eh  "*"
#3F21Ah  "/"
#3F226h  "+"
#3F232h  "-"
#3F23Eh  "="
#3F24Ah  "=="
#3F258h  "<"
#3F264h  ">"
#3F270h  "√"
```

```
#3F27Ch  "∂"
#3F288h  "≤"
#3F294h  "≥"
#3F2A0h  "≠"
#3F2ACh  "1"
#3F2B8h  "2"
#3F2C4h  "3"
#3F2D0h  "4"
#3F2DCh  "5"
#3F2E8h  "6"
#3F2F4h  "7"
#3F300h  "8"
#3F30Ch  "9"
#3F6F8h  " USAGE: "
#3FF8Bh  " Version 2BB"
#3FFADh  "Copyright HP 1986, 1987"
```

---

### Complexes HP28s 2BB

```
#2430Bh  (-6.80000000000E0,-1.50000000000E0)
#24330h  (+6.80000000000E0,+1.60000000000E0)
#2435Ah  (+0.00000000000E0,+0.00000000000E0)
#301E7h  (-1.00000000000E0,+0.00000000000E0)
#30E5Ah  (+0.00000000000E0,+0.00000000000E0)
#30EA7h  (+1.00000000000E0,+0.00000000000E0)
#3104Dh  (+0.00000000000E0,+1.00000000000E0)
#310F7h  (+0.00000000000E0,-1.00000000000E0)
```

### Complexes étendus HP28s 2BB

```
#2430Bh  (+0.00096800000000E0,+0.00002977915000E0)
```

---

# ANNEXE 6
# BIBLIOTHEQUE
# DE PROGRAMMES

# AVERTISSEMENT

Les programmes contenus dans cette partie ne peuvent être compris qu'après lecture du reste de cet ouvrage.

Cependant il est tout à fait possible de les utiliser sans connaître leur fonctionnement. Le lecteur pourra donc commencer par les essayer, puis, ayant vu les effets qu'ils provoquent, il pourra se servir de la partie sur le soft pour comprendre les méthodes utilisées.

En outre ces programmes fournissent des exemples aidant à la compréhension de la première partie (le soft) et montrent ce qu'il est possible de faire en langage machine...

Remarque: les * situés dans les listings de programmes en langage machine indiquent que le codage dépend de la version de la HP28 utilisée...

## COMMENT TAPER UN PROGRAMME EN LANGAGE MACHINE.

- Pour des raisons de facilité de lecture nous avons écrit les chaînes de caractères contenant les codes des programmes en langage machine, sous forme de séries de 5 codes hexadécimaux.

Ainsi: "69C20320001331F00FFF30F15D0131142164808C"

(Qui est le programme SPEED pour 28S)

sera écrit: 69C20 32000 1331F 00FFF 30F15 D0131 14216 4808C

Lorsque vous voyez une telle suite de codes il faut taper la chaîne correspondante sans espaces.

- Si vous pensez avoir commis une erreur en frappant une telle chaîne ou si vous désirez une lecture plus facile d'une chaîne de codes quelconque, le programme suivant vous intéressera: il insère un retour à la ligne tout les 5 caractères dans la chaîne.

1: "76C2007A206000009F20" -> 1: "76C20
07A20
60000
009F2
0"

**PAR5**

```
« "" OVER SIZE 1 SWAP
   FOR X
          OVER X DUP 4 + SUB 10 CHR + +
   5 STEP SWAP DROP
»
```

Après avoir modifié une chaîne de codes mise en forme à l'aide de PAR5, on peut réobtenir une chaîne de codes standard à l'aide du programme suivant :

```
REM5
« DUP SIZE 10 CHR -> S R
  « WHILE DUP R POS DUP
    REPEAT DUP2 1 - 1 SWAP SUB 3 ROLLD 1 +
           S SUB +
    END DROP »
»
```

***Remarque :*** **dans les listes de codes, les caractères gras représentent les séquences optionnelles**

---

# HORLOGE

Ce programme utilise l'horloge interne de la HP28 pour calculer l'heure...

Listing:

```
« RCWS 64 STWS #horloge SYSEVAL 8192 / B->R 3600
   / ->HMS offset HMS+ 24 MOD SWAP STWS »
```

#horloge est à remplacer par #123E pour la HP28c 1BB, par #1266 pour la 28c 1CC, par #11CA pour la HP28s 2BB... (toutes ces valeurs étant en hexadécimal).

offset est un réel à régler pour obtenir l'heure exacte.

**Explication du programme:**

RCWS rappelle la longueur d'entier binaire dans la pile 64 STWS fixe cette longueur à 64 (maximum), #horloge renvoie un entier qui représente le nombre de 8192èmes de seconde écoulés depuis la dernière mise en place des piles.

Par 8192 / B->R on transforme ce nombre en un réel entier en secondes. ->HMS ramène au format Heures, Minutes, Secondes.

Offset HMS+ 24 MOD calcule l'heure exacte (12 MOD pour un cycle de 12 heures). Enfin SWAP STWS redonne à WS sa valeur initiale (on récupère la valeur sauvée dans la pile).

On pourrait perfectionner le programme en lui adjoignant un calcul du jour, du mois, voir même de l'année (l'horloge interne de la 28 ne revient à zéro que tous les 1000 ans environ !!!!!).

# AUTOST

Programme en démarrage automatique: en lançant ce programme, vous provoquerez l'extinction de la machine (comme par SHIFT-OFF), mais lors du rallumage le programme continuera son exécution, ce qui permet le démarrage sur une application donnée...

La base du programme est l'utilisation d'un SYSEVAL (#7FBA sur HP28c 1BB, #7FEF sur HP28c 1CC, #18E58 sur HP28s 2BB) qui fait exactement la même chose que OFF (en fait il s'agit de la routine exécutée par la machine lorsqu'on l'éteint). Lorsqu'on lance le programme, il s'arrête lors de ce SYSEVAL, puis il se poursuit après un appui sur ON...

Donc, en résumé voici le mode d'emploi: taper le programme AUTOST, éteindre la machine en déclenchant ce programme, et non plus en appuyant sur OFF...

Voici un exemple de programme en démarrage automatique:

**Pour HP28c 1BB:**

```
« CLLCD #7FBA SYSEVAL 1400 .07 BEEP "HP-28c
version 1BB" 1 DISP 1000 .01 BEEP 1 WAIT
"N'oubliez pas d'etein-" 2 DISP 1000 .01
BEEP "-dre par AUTOST" 3 DISP 1000 .01 BEEP
1 WAIT "Have fun with your 28" 4 DISP 1000
.01 BEEP 1 WAIT CLMF »
```

**Pour HP28c 1CC:**

```
« CLLCD #7FEF SYSEVAL 1400 .07 BEEP "HP-28c
version 1CC" 1 DISP 1000 .01 BEEP 1 WAIT
"N'oubliez pas d'etein-" 2 DISP 1000 .01
BEEP "-dre par AUTOST" 3 DISP 1000 .01 BEEP
1 WAIT "Have fun with your 28" 4 DISP 1000
.01 BEEP 1 WAIT CLMF »
```

**Pour HP28s 2BB:**

```
« CLLCD #18E58 SYSEVAL 1400 .07 BEEP "HP-
28s version 2BB" 1 DISP 1000 .01 BEEP 1 WAIT
"N'oubliez pas d'etein-" 2 DISP 1000 .01
BEEP "-dre par AUTOST" 3 DISP 1000 .01 BEEP
1 WAIT "Have fun with your 28" 4 DISP 1000
.01 BEEP 1 WAIT CLMF »
```

# LASS

**LASS** : installation de programmes machine.
**Langage** : RPL (avec utilisation de SYSEVAL).

L'explication détaillée de ce programme est contenue dans le chapitre V de la seconde partie.

Entrée du programme: taper le listing suivant, puis ENTER, puis 'ASS' STO (se placer en mode HEXADECIMAL).

**Version pour HP28c :**

#fin.de.RAM est à remplacer par #50000 pour la version 2Ko, #52000 pour la version 6Ko, #60000 pour la version 34 Ko, #70000 pour la version 66 Ko... (toutes ces valeurs sont hexadécimales).

```
« "69A20" SWAP + "09F20" + -> LM « HEX ""
1 LM SIZE FOR X "#" LM X DUP2 1 + DUP SUB
3 ROLLD DUP SUB + + STR-> B->R CHR + 2 STEP
'LM.C' DUP PURGE STO #fin.RAM LM.C SIZE 2
* - SYSEVAL 1 GET 1 ->LIST LIST-> DROP
'LM.C' PURGE » »
```

**Version pour HP28s :**
(qui utilise le programme DOPATH listé plus loin):

```
« "69A20" SWAP + "09F20" + -> LM « PATH HOME
HEX "" 1 LM SIZE FOR X "#" LM X DUP2 1 +
DUP SUB 3 ROLLD DUP SUB + + STR-> B->R CHR
+ 2 STEP 'LM.C' DUP PURGE STO #D0000h LM.C
SIZE 2 * - SYSEVAL 1 GET 1 ->LIST LIST->
DROP 'LM.C' PURGE SWAP DOPATH » »
```

(Ce programme est légèrement différent de celui du chapitre V: on a ajouté la sauvegarde du chemin dans la pile (par PATH), puis on le reparcourt (par DOPATH) pour revenir au menu de départ).

# CRNAME

Le but de ce programme est la création de noms globaux impossibles à créer directement: soit parce qu'ils contiennent des caractères non admis (espace * / ...) soit parce qu'ils sont déjà référencés en ROM (ex 'SIN' ...). Avec CRNAME on peut créer de tels noms.

**En entrée:**
Le nom sous forme de chaîne: par ex "SIN". En sortie: le nom: par ex 'SIN'

*Remarque :* La séquence #1 #A CHK est optionnelle (ne la mettre que si le programme CHK est en mémoire).

**ATTENTION :** Le programme est à rentrer en mode HEXADECIMAL. Effectuer tout d'abord les deux séquences :

```
150 CHR 42 CHR + 32 CHR + 209 CHR + 2 CHR +
    'C.1' STO
144 CHR 47 CHR + 0 CHR + 'C.2' STO
```

Puis entrer le programme CRNAME lui-même:

**Pour HP28c :**

```
« #1 #A CHK 1 127 SUB C.1 OVER SIZE CHR + SWAP
+ C.2 + 'N.C' DUP PURGE STO #50000 N.C SIZE
2 * - SYSEVAL 1 GET 1 ->LIST LIST-> DROP 'N.C'
PURGE »
```

#50000 est l'adresse de fin de Ram. Elle est a modifier pour les HP28 à mémoire étendue (comme pour LASS).

**Pour HP28s:**

Il faut aussi entrer le programme DOPATH (situé plus avant dans cette bibliothèque).

```
« #1 #A CHK 1 127 SUB C.1 OVER SIZE CHR +
SWAP + C.2 + PATH SWAP HOME 'N.C' DUP
PURGE STO #D0000 N.C SIZE 2 * - SYSEVAL 1
GET 1 ->LIST LIST-> DROP 'N.C' PURGE SWAP
DOPATH »
```

**Explication du programme :**
Le but est de créer l'objet 'nom global' dans la RAM. Pour cela, on utilise une méthode calquée sur celle du programme LASS précédement décrit. En résumé: on code le nom dans une liste en ajoutant le prologue '02A96' et l'offset de fin de liste '02F90'.

On ajoute aussi le prologue de nom global. Ces trois adjonctions sont réalisées grâce aux deux additions de data (CRNAME.D1 et CRNAME.D2). On doit aussi coder la longueur du nom: ceci ce fait grâce à la séquence SIZE CHR +. Du point de vue sécurité: #1 #A CHK vérifie la présence de la chaîne de caractères dans la pile. 1 127 SUB ramène ensuite la chaîne à la longueur maximale autorisée (127 caractères). Le SYSEVAL place dans la pile l'objet { 'nom' } dans la pile.
Enfin la séquence 1 GET 1 ->LIST LIST-> DROP extrait l'objet et oblige le calculateur à le recréer, ceci afin de lui donner une existence réelle. Sur les HP28s, on se place en outre dans le menu HOME pour pouvoir calculer l'adresse du SYSEVAL.

## NOCLUSR

Supprimons le CLUSR... Pour cela on va créer un programme ayant pour nom 'CLUSR' (grâce au programme CRNAME) qui aura priorité sur le CLUSR situé en ROM...

Marche à suivre:

- entrer `CRNAME`
- entrer « `"No CLUSR Avaible" 1 DISP 1400 .07 BEEP` »
- taper `"CLUSR"` puis appeler `CRNAME`
- `'CLUSR'` apparaîtra dans la pile...
- taper `STO`. Dorénavant toute tentative pour faire un CLUSR se soldera par le nouveau message d'erreur `"No CLUSR Avaible"`.

Pour revenir à l'état standard: Taper `'CLUSR' PURGE`

## NOSYSEVAL

Supprimons à présent l'instruction SYSEVAL. On va proceder comme pour CLUSR:

- taper « `"No SYSEVAL Avaible" 1 DISP 1400 .07 BEEP` »
- taper `"SYSEVAL" CRNAME`
- `'SYSEVAL'` apparaît alors dans la pile
- taper `STO`

Retour à l'état normal: `'SYSEVAL' PURGE`

On peut de même supprimer d'autres instructions.

Il faut cependant savoir que le instructions recrées n'ont priorité sur les fonctions ROM que lorsqu'on tape le nom en toutes lettres (et non pas en l'exécutant via un menu).

Pour SYSEVAL et CLUSR, c'est toujours le cas, et c'est pour cela que l'on peut totalement 'dériver' ces instructions...

# INV.VID

Le programme suivant effectue une inversion vidéo quasi-instantannée de l'écran de la HP28...

Entrée du programme (lire avertissement page 189) :
Taper la chaîne des codes (située apres le listing du programme) en une seule ligne, sans espace, puis exécuter LASS. Stocker le résultat (System Object System Object) dans INV.VID.

```
Codes   Labels   Mnémoniques   Commentaires
76C20            CON(5) #02C67  Début de l'objet programme
*****            CON(5) #*****  figesage écran (1BB:07C19 1CC:07C4E 2BB:18A8S)
69C20            CON(5) #02C96  Début de l'objet prgm machine
E5000    début   CON(5) (fin)-(début)   longueur de cet objet
133              AD1EX
103              R3=A           Sauvegarde de D1
34*****          LC(5)  #*****  fin écran 1 (1BB,1CC:402DF 2BB:FFA5F)
1F*****          D1=(5) #*****  début écran 1 (1BB,1CC:40078 2BB:FF840)
7220             GOSUB  sub     inversion
34*****          LC(5)  #*****  fin écran 2 (1BB,1CC:405DF 2BB:FFE27)
1F*****          D1=(5) #*****  début écran 2 (1BB,1CC:40400 2BB:FFC00)
7010             GOSUB  sub     inversion
113              A=R3
131              D1=A           récupération de D1
142              A=DAT0 A       fin de routine
164              D0=D0+ 5       (une routine LM se finit toujours par la
808C             PC=(A)         séquence A=DAT0 A  D0=D0+ 5  PC=(A])

27       sub     P=     7       On travaille sur une colonne, c'est à dire sur
                                8 quartets: on fixe donc P à 7 pour que le
                                champ
                                WP contienne 8 quartets (7 à 0)
1531     sub10   A=DAT1 WP      lecture d'une colonne
B9C              A=-A-1 WP      inversion (A=-A-1 correspond à A=NOT(A))
1511             DAT1=A WP      écriture de la colonne inversée
177              D1=D1+ 8       colonne suivante
133              AD1EX          copie de D1 dans A (on doit procéder en deux
131              D1=A           étape car A=D1 n'existe pas)
8BA              ?A#=C  A       test: doit on continuer ?
9E               GOYES  sub10   si oui -> on repart à sub10
20               P=     0       sinon: on met P à 0 (valeur standard)
01               RTN            retour de sous routine
         fin

09F20            CON(5) #02F90 fin de l'objet programme
```

**Chaînes de codes:**

**Pour la 1BB:**

```
'/6C20  91C'/0  69C20  E5000  13310  334FD  2041F
8'/004  '/2203  4FD50  41F00  404'/0  10113  13114
21648  08C2'/  1531B  9C151  11'/'/1  33131  8BA9E
20010  9F20
```

**Pour la 1CC:**

```
'/6C20  E4C'/0  69C20  E5000  13310  334FD  2041F
8'/004  '/2203  4FD50  41F00  404'/0  10113  13114
21648  08C2'/  1531B  9C151  11'/'/1  33131  8BA9E
20010  9F20
```

**Pour la 2BB:**

```
'/6C20  58A81  69C20  E5000  13310  334F5  AFF1F
048FF  '/2203  4'/2EF  F1F00  CFF'/0  10113  13114
21648  08C2'/  1531B  9C151  11'/'/1  33131  8BA9E
20010  9F20
```

# MODUL/DEMOD

Il s'agit de programme permettant la création d'effets sonores intéressant : émission d'un son de fréquence variable par paliers (réglables en durée) de fréquences (incrémentation réglable). Les deux programmes sont très semblables (seul le sens d'incrémentation varie), et de ce fait seul le listing de MODUL est présenté. Les chaînes de codes sont présentées pour les deux programmes et pour les trois versions de machines...

Entrée du programme (lire avertissement page 189) :
Taper la chaîne de codes (située après le listing de MODUL), en une seule ligne, sans espace et stocker le résultat (System Object) dans MODUL ou DEMOD...

```
Codes     Labels   Mnémoniques     Commentaires
69C20              CON(5) 02C96    début d'objet routine LM
15000     debut    CON(5) fin-debut longueur du programme
8F*****            GOSBVL SAV.REG  sauvegarde D0, D1, B et D
3401000            LCHEX  00010    réglable: fréquence de départ
D7                 D=C    A
DB        L1       C=D    A
06                 RSTK=C          sauvegarde fréquence en cours dans RSTK
3405000            LCHEX  00050    réglable: durée d'un palier
8F*****            GOSBVL BEEP     émission du BEEP
07                 C=RSTK          récupération fréquence
D7                 D=C    A
3401000            LCHEX  00010    réglable: incrémentation fréquence
C3                 D=D+C  A        pour DEMOD: D=D-C A (E3)
3400010            LCHEX  01000    réglable: fréquence de fin
8BF                ?D≤C   A        pour DEMOD: ?D≥C A (8BB)
7D                 GOYES  L1
8F*****            GOSBVL LOAD.REG récupération D0 D1 B et D
142                A=DAT0 A        fin de routine
164                D0=D0+5
808C               PC=(A)
          fin
```

*Remarque :* Les fréquences de début, de fin, l'incrément et la durée d'un palier, sont réglables. Pour MODUL, on doit toujours avoir freq. de début ≤ freq. de fin, et la relation inverse pour DEMOD.

**Chaînes de codes:**

**Pour MODUL:**

**Pour HP28c 1BB:**

```
69C20  15000  8F2EE  40340  1000D  "/DB06  34050
008FC  61010  "/D"/34  01000  C3340  00108  BF"/D8
F91F4  01421  64808  C
```

**Pour HP28c 1CC:**

```
69C20  15000  8F2EE  40340  1000D  "/DB06  34050
008F3  A1010  "/D"/34  01000  C3340  00108  BF"/D8
F91F4  01421  64808  C
```

**Pour HP28s 2BB:**

```
69C20  15000  8F180  50340  1000D  "/DB06  34050
008F3  DA220  "/D"/34  01000  C3340  00108  BF"/D8
F8B05  01421  64808  C
```

**Pour DEMOD :**

**Pour HP28c 1BB:**

```
69C20  15000  8F2EE  40340  0010D  "/DB06  34050
008FC  61010  "/D"/34  01000  E3340  10008  BB"/D8
F91F4  01421  64808  C
```

**Pour HP28c 1CC:**

```
69C20  15000  8F2EE  40340  0010D  "/DB06  34050
008F3  A1010  "/D"/34  01000  E3340  10008  BB"/D8
F91F4  01421  64808  C
```

**Pour HP28s 2BB:**

```
69C20  15000  8F180  50340  0010D  "/DB06  34050
008F3  DA220  "/D"/34  01000  E3340  10008  BB"/D8
F8B05  01421  64808  C
```

# MUSICLM

MUSICLM est un programme permettant de jouer un air de musique précodé, à une vitesse très régulière, sans interruption entre les notes...

Entrée du programme (lire l'avertissement page 189) :
On tape la chaîne des codes, en une seule ligne, sans espace, et on exécute le programme LASS. Le résultat (System Object) sera stocké dans MUSICLM.

## Listing du programme:

```
Codes     Labels  Mnémoniques     Commentaires
69c20             CON(5) 02c96    début de l'objet programme-assembleur
.....             longueur à déterminer selon le nombre de notes programmées:
                       #4D (hexadécimal) + 10 (décimal) * nbre de notes
                           ex: pour 9 notes: 7A000
8r******          GOSBVL SAV.REG sauvegarde des registres de travail de la 28
7...              GOSUB L1       la longueur du saut est 10 * nbre de notes
                                 ex: pour 9 notes on codera le saut par 7A50
```

liste des notes: fréquence en hertz sur 5 quartets, suivi de la durée en millièmes de secondes, sur 5 quartets. Les valeurs sont à coder en HEXADECIMAL.

*Exemple:* pour un LA (440 Hz) de .759 secondes on codera:

```
                          8B100
                          6F200
```

ATTENTION: la dernière note doit impérativement être la note de fréquence 0 et de durée 0. On ne doit pas utiliser une telle note au milieu de la liste des notes, car le code de fréquence 00000 est utilisé pour indiquer la fin de liste.

```
07        L1      C=RSTK          C=adresse de début de la liste
135               D1=C
147               C=DAT1 A
D7                D=C A           D=fréquence de la note
174               D1=D1+5
143               A=DAT1 A
174               D1=D1+5
137               CD1EX
D6                RSTK=C          sauvegarde adresse en cours
D6                C=A A           C=durée de la note
8AA               ?C=0 A          a-t-on fini?
D0                GOYES L2
8r******          GOSBVL BEEP     émission du BEEP
69DF              GOTO L1         note suivante
07        L2      C=RSTK          on vide la pile RSTK
```

```
8r*****         GOSBVL LOAD.REG récupération des registres
142             A=DAT0 A        fin de routine
164             D0=D0+5
808C            PC=(A)
```

**Chaîne de codes:**

**Pour HP28c 1BB:**

```
69C20  .....  8F2EE  40'7..  .[...  notes    ....]
0'7135 14'D'7 1'7414 31'741 3'706D 68AAD    08FC6
10169  DF0'8  F91F4  01421  64808  C
```

**Pour HP28c 1CC:**

```
69C20  .....  8F2EE  40'7..  .[...  notes    ....]
0'7135 14'D'7 1'7414 31'741 3'706D 68AAD    08F3A
10169  DF0'8  F91F4  01421  64808  C
```

**Pour HP28s 2BB:**

```
69C20  .....  8F180  50'7..  .[...  notes    ....]
0'7135 14'D'7 1'7414 31'741 3'706D 68AAD    08F3D
A2269  DF0'8  F8B05  01421  64808  C
```

# Programme CHK

**But :** vérification du nombre et du type d'objets dans la pile.

*Remarque :* Le programme CHK est utilisé par de nombreux programmes en langage machine, présentés dans cette annexe. Si vous ne désirez pas le taper immédiatement, mais prévoyez de le faire par la suite, vous pouvez stocker, temporairement, « `DROP2` » dans **`'CHK'`**, pour en simuler la présence.

**Données en entrée :**

- niveau 2: nombre minimal d'objets;
- niveau 1: masque du type des objets (voir codes dans le listing du programme).

**En sortie:**

- rien si tout s'est bien passé;
- Too Few Arguments ou Bad Argument Type le cas échéant (dans ce cas les arguments de CHK restent dans la pile).

*Exemples* : (on représente l'état de la pile avant le CHK).

```
      3:              "ABC"
      2:                 #0
      1:                 #0
est correct (on demande la présence d'un nombre nul
d'objets)

      4:                 #0
      3:              "ABC"
      2:                 #2
      1:              #0B00
est aussi correct: on demande la présence de deux objets, le
premier de type quelconque (offset 00), le second de type 'entier
binaire' (offset 0B)

      2:                 #1
      1:                 #0
renverra un 'Too Few Arguments' car on demande 1 objet dans la
pile alors qu'il n'y en a pas...

      3:              "ABC"
      2:                 #1
      1:                #0B
renverra un 'Bad Argument Type' car on demande un entier et que
l'on a en fait une chaîne.
```

Entrée du programme (lire l'avertissement page 189):

Taper la liste des codes (située apres le listing commenté du programme) en une seule ligne, sans espace.

Exécuter ensuite LASS ou RASS et stocker le résultat (System Object System Object DROP2) dans CHK. Il est conseillé de stocker le programme sous ce nom, car il est utilisé comme sous-routine d'autres programmes présentés plus avant dans cette bibliothèque de programmes.

**Pour HP28c**

## Listing du programme:

| Codes | Labels | Instructions | Commentaires |
|---|---|---|---|
| 76C20 | | CON(5) 02C67 | début de structure programme |
| ***** | | CON(5) ***** | 1BB:1C35C 1CC:1CC48 2BB:0C36A |
| | | | met à jour STACKSIZE |
| 69C20 | | CON(5) 02C96 | début de l'objet codes-LM |
| ***** | deb | CON(5) fin-deb | longueur du programme: |
| | | | 1BB,1CC:0016D 2BB:00177 |
| 8F***** | | GOSBVL SAV.REG | sauve les registres |
| | | | 1BB,1CC:04EE2 2BB:05081 |
| AF2 | | C=0 W | |
| 1BF510C | | D0=C015F | mise à xéro de CMD NUMBER |
| 140 | | DAT0=C A | pour HP28s SEULEMENT |
| 84A | | ST=0 10 | première vérification... |
| 302 | | LCHEX(1) 2 | deux objets demandés |
| DA | | A=C A | A passe le nbr d'obj à la sous |
| | | | routine CHK1 |
| 33B0B0 | | LCHEX(4) 0B0B | 2 entiers demandés |
| AF5 | | B=C W | B passe ce paramètre |
| 7A40 | | GOSUB CHK1 | vérification... |
| 8F***** | | GOSBVL LOAD.REG | 1BB,1CC:04F19 2BB:050B9 |
| 85A | | ST=1 10 | seconde vérification |
| 143 | | A=DAT1 A | |
| 130 | | D0=A | D0=adresse objet 1 |
| 169 | | D0=D0+10 | D0=adresse mantisse objet 1 |
| 1527 | | A=DAT0 W | lecture de la liste des types |
| AF8 | | B=A | |
| 174 | | D1=D1+5 | objet suivant |
| 143 | | A=DAT1 A | |
| 130 | | D0=A | D0=adresse objet 2 |
| 169 | | D0=D0+10 | D0=adresse mantisse objet 2 |
| 142 | | A=DAT0 A | A=nbre d'objets |
| 3420000 | | LCHEX(5) 00002 | |
| CA | | A=A+C A | 2 objets en plus (les 2 |
| | | | entiers de CHK) |

```
174                 D1=D1+5       D1=debut de la liste des
                                  objets de la pile
7110                GOSUB CHK1    vérification...
8F*****             GOSBVL LOAD.REG
142                 A=DAT0 A
164                 D0=D0+5
808C                PC=(A)
1B*****   CHK1      D0=STACKSIZE  1BB,1CC:4F0F5 2BB:C00F9
100                 R0=A          sauvegarde de A dans R0
E4                  A=A+1         calcul du STACKSIZE
D6                  C=A A         correspondant à la valeur de A
C4                  A=A+A
AC4                 A=A+A  A
CA                  A=A+C  A
146                 C=DAT0 A       C=STACKSIZE réel
8BA                 ?C≥A   A       y a t'il assez d'objets?
91                  GOYES OK1
3410200             LCHEX(5) 00201 erreur 'Too Few Arguments'
DA        ERR       A=C    A
8F*****             GOSBVL LOAD.REG
8D*****             GOVLNG ERROR   affichage de l'erreur
7460      OK1       GOSUB L0       ce GOSUB permet la
                                   détermination de l'adresse
                                   de la liste qui suit
                                  prol.  objet:                offset
00000               CON(5) 00000   objet quelconque              00
11920               CON(5) 02911   Short Integer                 01
33920               CON(5) 02933   Real                          02
55920               CON(5) 02955   Extended Real                 03
77920               CON(5) 02977   Complex                       04
D9920               CON(5) 0299D   Extended Complex              05
FB920               CON(5) 029BF   Byte                          06
1E920               CON(5) 029E1   Unknown Object 1              07
A0A20               CON(5) 02A0A   Array                         08
C2A20               CON(5) 02A2C   Unknown Object 2              09
E4A20               CON(5) 02A4E   String                        0A
07A20               CON(5) 02A70   Binary Integer                0B
69A20               CON(5) 02A96   List                          0C
8BA20               CON(5) 02AB8   Ram/Rom Pair                 0D
ADA20               CON(5) 02ADA   Algebraic                     0E
76C20               CON(5) 02C67   Program                       0F
69C20               CON(5) 02C96   Assembly Code                 10
21D20               CON(5) 02D12   Global Name                   11
73D20               CON(5) 02D37   Local Name                    12
C5D20               CON(5) 02D5C   Rom Pointer                   13
110       L0        A=R0           récupération de A
07                  C=RSTK         C=adresse du début de
                                    la liste
108                 R0=C
```

```
86A             ?ST=0 A         premier essai?
60              GOYES L2
CC              A=A-1 A         mise à jour de A
CC      L1      A=A-1 A
8A8             ?A=0  A         a-t-on fini?
00      L2      RTNYES
118             C=R0
134             D0=C
969     L3      ?B=0  B         prologue atteint?
C0              GOYES L4
A6D             B=B-1 B         prologue suivant
164             D0=D0+5
64FF            GOTO L3
146     L4      C=DAT0 A        C=prologue théorique
8AA             ?C=0 A          obj. geq. ?
62              GOYES L5
D7              D=C A           prologue théorique dans D
101             R1=A            sauvegarde A
143             A=DAT1 A
133             AD1EX           D1=adresse objet étudié
147             C=DAT1 A        C=prologue objet
131             D1=A            récupération de D
1111            A=R1            récupération A
8A3             ?D=C A          prologue bon?
D0              GOYES L5        si oui: -> L5
3420200         LCHEX(5) 00202  sinon: erreur Bad Argument...
643F            GOTO ERR
174     L5      D1=D1+5         objet suivant
815             BSRC            offset prologue suivant
815             BSRC
6EAF            GOTO L1         on continue...
        fin
*****           CON(5) DROP2    1BB:17859 1CC:1793B 2BB:07644
                                [on élimine les deux entiers]
09F20           CON(5) 02F90    fin de structure programme
```

**Utilisation du programme:**

En lui-même, le programme CHK ne présente aucun intérêt. En fait, il ne peut être utilisé que comme outil de contrôle en entrée de fonctions-utilisateur, et en particulier pour les programmes assembleurs qui ne possèdent pas de message d'erreurs: une erreur dans le type de données peut quelquefois entraîner de graves conséquences: perte d'une partie des données, memory lost...

Il est à noter que le programme CHK est abondamment utilisé par les autres programmes assembleurs présentés, mais il faut savoir qu'il est optionnel: si l'utilisateur a la rigueur de faire une vérification des objets en entrée des fonctions assembleur, il peut éviter la frappe de CHK.

**Chaînes de codes :**

**Pour HP28c 1BB :**

```
'/6C20  C53C1  69C20  D6100  8F2EE  40AF2  84A30
2DA33  B0B0A  F5'/A4  08F91  F4085  A1431  30169
152'/A  F81'/4  14313  01691  42342  0000C  A1'/4'/
1108F  91F40  14216  4808C  1B5F0  F4100  E4D6C
4C4CA  1468B  A9134  10200  DA8F9  1F408  D8533
0'/460  00000  11920  33920  55920  '/'/920  D9920
FB920  1E920  A0A20  C2A20  E4A20  0'/A20  69A20
8BA20  ADA20  '/6C20  69C20  21D20  '/3D20  C5D20
1100'/  10886  A60CC  CC8A8  00118  13496  9C0A6
D1646  4FF14  68AA6  2D'/10  11431  3314'/  13111
18A3D  03420  20064  3F1'/4  81581  56EAF  958'/1
09F20
```

**Pour HP28c 1CC:**

```
'/6C20  84CC1  69C20  D6100  8F2EE  40AF2  84A30
2DA33  B0B0A  F5'/A4  08F91  F4085  A1431  30169
152'/A  F81'/4  14313  01691  42342  0000C  A1'/4'/
1108F  91F40  14216  4808C  1B5F0  F4100  E4D6C
4C4CA  1468B  A9134  10200  DA8F9  1F408  D8533
0'/460  00000  11920  33920  55920  '/'/920  D9920
FB920  1E920  A0A20  C2A20  E4A20  0'/A20  69A20
8BA20  ADA20  '/6C20  69C20  21D20  '/3D20  C5D20
1100'/  10886  A60CC  CC8A8  00118  13496  9C0A6
D1646  4FF14  68AA6  2D'/10  11431  3314'/  13111
18A3D  03420  20064  3F1'/4  81581  56EAF  B39'/1
09F20
```

**Pour HP28s 2BB:**

```
'/6C20  A63C0  69C20  '/'/100  8F180  50AF2  1BF51
0C144  84A30  2DA33  B0B0A  F5'/A4  08F8B  05085
A1431  30169  152'/A  F81'/4  14313  01691  42342
0000C  A1'/4'/  1108F  8B050  14216  4808C  1B9F0
0C100  E4D6C  4C4CA  1468B  A9134  10200  DA8F8
B0508  DA693  0'/460  00000  11920  33920  55920
'/'/920  D9920  FB920  1E920  A0A20  C2A20  E4A20
0'/A20  69A20  8BA20  ADA20  '/6C20  69C20  21D20
'/3D20  C5D20  1100'/  10886  A60CC  CC8A8  00118
13496  9C0A6  D1646  4FF14  68AA6  2D'/10  11431
3314'/  13111  18A3D  03420  20064  3F1'/4  81581
56EAF  446'/0  09F20
```

Ce programme est l'équivalent du programme LASS.

Données : **en entrée** : chaîne de caractères contenant les codes hexadécimaux représentant l'objet à créer;

**en sortie** : l'objet ou un message d'erreur:
*Bad Argument Type*
ou *Too Few Arguments*
ou *Too Few Memory*

Entrée du programme (lire l'avertissement page 189):
Taper la chaîne de codes (située apres le listing du programme) en une seule ligne, sans espace, puis exécuter LASS. Stocker ensuite le résultat dans RASS. Dorénavant toute référence à LASS dans un programme pourra être remplacée par RASS qui est quasi-instantané...

*Remarque :* La séquence #1 #0 + #0A CHK est optionnelle (ne la mettre que si le programme CHK est en mémoire).

**Listing du programme :**

```
Code      Labels    Instructions          Commentaires
76C20               CON(5) 02C67
**..**              #1 #0 + #0A CHK       vérification objets ds la pile
                                          (optionnel)
*****               DUP                   ces deux instructions provoquent
*****               DROP                  une sauvegarde dans LAST...
69C20               CON(5) 02C96          Assembly code
*****     début     CON(5) fin-début
8F******            GOSBVL SAV.REG        sauvegarde D0 D1 B D
1BF510C             D0=(5) C015F          mise à zéro de CMD.NUMBER
D2                  C=0    A              (HP28s seulement)
144                 DAT0=C A
84A                 ST=0 10               premier essai
143                 A=DAT1 A              A=adresse objet
131                 D1=A
174                 D1=D1+5
143                 A=DAT1 A              A=longueur chaîne
3450000             LCHEX 00005
8A6                 ?A#C A                chaîne vide ?
91                  GOYES OK
3410200             LCHEX 00201           erreur: Too Few Argument
DA                  A=C A
```

```
8F*****             GOSBVL LOAD.REG
8D*****             GOVLNG ERROR
EA        OK        A=A-C A
81C                 ASRB                 A=Nb codes dans la chaine
100                 R0=A
8F*****             GOSBVL LOAD.REG      récupération D0 D1 B et D
119       L2        C=R0                 C=place à réserver
8F*****             GOSBVL RES.ROOM
532                 GONC OK3             réservation réussie -> OK3
8F*****             GOSBVL LOAD.REG      récupération D0 D1 B D
86A                 ?ST=0 10             premier essai ?
90                  GOYES RETRY          oui -> on recommence
8D*****             GOLNG TFM            Too Few Memory
8F*****   RETRY     GOSBVL GARB.COLL     Garbage Collector
85A                 ST=1 10              second essai
65DF                GOTO L2
136       OK3       CD0EX                sauvegarde de D0 qui
109                 R1=C                 contient l'adresse de la place
                                         réservée
8F*****             GOSBVL LOAD.REG
119                 C=R1
134                 D0=C
110                 A=R0
D9                  B=A   A
CD                  B=B-1 A              B=nb. qu. à coder-1
143                 A=DAT1 A
131                 D1=A
179                 D1=D1+10             D1=adresse contenu de la chaîne
14B       L3        A=DAT1 B             lecture 1 carac.
3103                LCHEX 30             ASCII -> HEXA
B6A                 A=A-C B
3190                LCHEX 09
9EA                 ?A<=C B
90                  GOYES L4
3170                LCHEX 07
B6A                 A=A-C B
1580      L4        DAT0=A 1             écriture quartet
160                 D0=D0+1              quartet suivant
171                 D1=D1+2
CD                  B=B-1 A
59D                 GONC L3              si pas fini, on continue
8F*****             GOSBVL LOAD.REG      récupération D0 D1 B D
111                 A=R1
141                 DAT1=A A             résultat à la place de la chaîne
142                 A=DAT0 A             fin de routine
164                 D0=D0+5
909C                PC=(A)
09F20               CON(5) 02F90         fin de structure
          fin
```

**Chaînes de codes:**

### Pour HP28c 1BB :

```
'/6C20  07A20  60000  1B0D0  3D9F8  107A2  07000
0A021  D2030  3484B  41F'/  1F38'/  169C2  0ED00
08F2E  E4084  A1431  311'/4  14334  50000  8A691
34102  00DA8  F91F4  08D35  330EA  81C10  08F91
F4011  88F'/B  15053  28F91  F4086  A908D  C2330
8FD'/9  4085A  65DF1  36109  8F91F  40119  13411
0D8CD  14313  11'/91  4B310  3B6A3  1909E  A9031
'/0B6A  15801  601'/1  CD59D  8F91F  40111  14114
21648  08C09  F20
```

### Pour HP28c 1CC :

```
'/6C20  07A20  60000  100E0  39809  107A2  07000
0A021  D2030  3484B  43D8'/  1129'/  169C2  0ED00
08F2E  E4084  A1431  311'/4  14334  50000  8A691
34102  00DA8  F91F4  08D35  330EA  81C10  08F91
F4011  88F'/B  15053  28F91  F4086  A908D  C2330
8FD'/9  4085A  65DF1  36109  8F91F  40119  13411
0D8CD  14313  11'/91  4B310  3B6A3  1909E  A9031
'/0B6A  15801  601'/1  CD59D  8F91F  40111  14114
21648  08C09  F20
```

### Pour HP28s 2BB :

```
'/6C20  F7971  0FEF2  EEE80  07A20  70000  A021D
20303  484B4  CD5'/0  A26'/0  69C20  AE000  8F180
501BF  510CD  21448  4A143  1311'/  41433  45000
08A69  13410  200DA  8F8B0  508DA  6930E  A81C1
008F8  B0501  188FE  A3505  328F8  B0508  6A908
DE393  08F49  A4085  A65DF  13610  98F8B  05011
91341  10D8C  D1431  311'/9  14B31  03B6A  31909
EA903  1'/0B6  A1580  1601'/  1CD59  D8F8B  05011
11411  42164  808C0  9F20
```

**Syntaxe :**

En entrée : 2: #011CA adresse du peek
1: #10 longueur à lire

En sortie : chaîne contenant les codes hexadécimaux des quartets lus

*Exemple :* #0 #10 PEEK renvoie les 16 premiers quartets de la ROM: "8FDB400000CFE" (pour la 28s).

Messages d'erreur possibles :

*Too Few Arguments.*
*Bad Argument Type.*
*Too Few Memory.*

Entrée du programme (lire l'avertissement page 189):

Taper la chaîne des codes (située après le listing), en une seule ligne, sans espace. Exécuter LASS et stocker le résultat (#2 #0 + #B0B CHK DUP2 DROP2 System Object) dans PEEK.

*Remarque :* la séquence #2 #0 + #0B0B CHK est optionnelle (ne la mettre que si le programme CHK est en mémoire).

```
Code      Labels    Instructions        Commentaires
76C20               CON(5) 02C67        début de structure
**...**             #2 #0 + #0B0B CHK   vérification des objets (optionnel)
*****               DUP2                cette séquence réalise la
sauvegarde
*****               DROP2               dans LAST
69C20               CON(5) 02C96        prgm LM
*****     Début     CON(5) Fin-Début    longueur code
8F*****             GOSBVL SAV.REG      sauvegarde de D0 D1 B D
1BF510C             D0=C015F            mise à 0 de CMD.NUMBER
D2                  C=0 A               (28s seulement)
144                 DAT0=C A
84A       OK2       ST=0 A              premier essai de réservation
147                 C=DAT1 A
137                 CD1EX
179                 D1=D1+10
143                 A=DAT1 A            A=longueur du peek
C4                  A=A+A A
34A0000             LCHEX 0000A
C2                  C=A+C A             C=place à réserver
109                 R1=C
8F*****             GOSBVL LOAD.REG     récupération des registres
174                 D1=D1+5             objet au niveau 2
```

```
143                 A=DAT1 A
133                 AD1EX                 D1=adresse de cet objet
179                 D1=D1+10
147                 C=DAT1 A              C=adresse où peeker
108                 R0=C
131                 D1=A
119       L3        C=R1
8F*****             GOSBVL RES.ROOM       réservation place en RAM
532                 GONC OK3              si réussi->OK3
8F*****             GOSBVL LOAD.REG       sinon...
86A                 ?ST=0 10              premier essai?
90                  GOYES RETRY           oui->on essai encore
8D*****             GOVLNG TFM            non->Too Few Memory
8F*****   RETRY     GOSBVL GARB.COLL      garbage collector
85A                 ST=1 10               second essai
65DF                GOTO L3               on recommence
136       OK3       CD0EX
10A                 R2=C                  R2=adresse place réservée
134                 D0=C
34E4A20             LCHEX 02A4E
144                 DAT0=C A              écriture prologue chaîne
3450000             LCHEX 00005
164                 D0=D0+5
111                 A=R1
EA                  A=A-C A               A=longueur chaîne
140                 DAT0=A A              écriture longueur
164                 D0=D0+5               D0=adresse ou écrire
D8                  B=A A
110                 A=R0
131                 D1=A                  D1=adresse du peek
E1        L4        B=B-C A
8A9                 ?B=0 A
13                  GOYES L6
AE0                 A=0 B
15B0                A=DAT1 1
3103                LCHEX 30
A6A                 A=A+C B
3193                LCHEX 39              TRANSFERT
9EA                 ?A≤C  B
90                  GOYES L5
3170                LCHEX 07
A6A                 A=A+C B
148       L5        DAT0=A B
161                 D0=D0+2
170                 D1+D1+1
3120                LCHEX 02
6DCF                GOTO L4
8F*****   L6        GOSBVL LOAD.REG       récupération D0 D1 B D
174                 D1=D1+5               on drope la longueur
E7                  D=D+1
112                 A=R2
141                 DAT1=A A              on remplace l'adresse par la chaîne
142                 A=DAT0 A              fin de routine
164                 D0=D0+5
808C                PC=(A)
09F20     Fin       CON(5) 02F90          fin de structure
```

**Chaînes de codes :**

**Pour HP28c 1BB :**

```
'/6C20  07A20  60000  2B0D0  3D9F8  107A2  09000
0B0B0  21D20  30348  4B4B0  8'/195  8'/169  C205F
0008F  2EE40  84A14  '/13'/1  '/9143  C434A  0000C
21098  F91F4  01'/41  43133  1'/914  '/1081  31119
8F'/B1  50532  8F91F  4086A  908DC  23308  FD'/94
085A6  5DF13  610A1  3434E  4A201  44345  00001
64111  EA140  164D8  11013  1E18A  913AE  015B0
3103A  6A319  39EA9  031'/0  A6A14  81611  '/0312
06DCF  8F91F  401'/4  E'/112  14114  21648  08C09
F20
```

**Pour HP28c 1CC :**

```
'/6C20  07A20  60000  200E0  39809  107A2  09000
0B0B0  21D20  30348  4B4DE  8'/1B3  9'/169  C205F
0008F  2EE40  84A14  '/13'/1  '/9143  C434A  0000C
21098  F91F4  01'/41  43133  1'/914  '/1081  31119
8F'/B1  50532  8F91F  4086A  908DC  23308  FD'/94
085A6  5DF13  610A1  3434E  4A201  44345  00001
64111  EA140  164D8  11013  1E18A  913AE  015B0
3103A  6A319  39EA9  031'/0  A6A14  81611  '/0312
06DCF  8F91F  401'/4  E'/112  14114  21648  08C09
F20
```

**Pour HP28s 2BB :**

```
'/6C20  E8971  0FEF2  EEE80  07A20  90000  B0B02
1D203  03484  B46F5  '/0446  '/069C  20101  008F1
80501  BF510  CD214  484A1  4'/13'/  1'/914  3C434
A0000  C2109  8F8B0  501'/4  14313  31'/91  4'/108
13111  98FEA  35053  28F8B  05086  A908D  E3930
8F49A  4085A  65DF1  3610A  13434  E4A20  14434
50000  16411  1EA14  0164D  81101  31E18  A913A
E015B  03103  A6A31  939EA  9031'/  0A6A1  48161
1'/031  206DC  F8F8B  0501'/  4E'/11  21411  42164
808C0  9F20
```

# POKE

**Syntaxe:**

En entrée : 2: #01251 adresse où poker
1: "ABF4324D3" quartets à poker

Pas de sortie

*Exemple :* #FF840 "FFFFFFFF" place F (15 en décimal) dans les cases mémoire #FF840 à #FF847 (noircit la première colonne de l'écran sur la 28s, pour la 28c,remplacer #FFF840 par #40078).

Messages d'erreur possibles :

*Too Few Arguments*
*Bad Argument Type*

Entrée du programme (lire l'avertissement page 189):

Taper la chaîne des codes (située après le listing) une seule ligne, sans espace, puis exécuter LASS. Stocker le résultat (#2 #0 + #B0A CHK System Object DROP2) dans POKE

*Remarque :* La séquence #2 #0 + #0B0A CHK est optionnelle (ne la mettre que si le programme CHK est en mémoire)..

**Listing du programme:**

```
Code      Labels    Instructions        Commentaires

76C20               CON(5) 02C67        début de structure
**..**              #2 #0 + #0B0A CHK   vérification de objets (optionnel)
69C20               CON(5) 02C96        programme LM
A7000     Début     CON(5) Fin-Début    longueur du code
8F*****             GOSBVL SAV.REG      sauvegarde D0 D1 B D
143                 A=DAT1 A
132                 AD0EX
164                 D0=D0+5             D0=adresse longueur chaîne
146                 C=DAT0 A            C=longueur chaîne
164                 D0=D0+5             D0=adresse mantisse chaîne
D5                  B=C A
174                 D1=D1+5
143                 A=DAT1 A
131                 D1=A                D1=adresse objet 2
179                 D1=D1+10
143                 A=DAT1 A
131                 D1=A                D1=adresse où poker
3450000             LCHEX 00005
E1        L4        B=B-C A             TRANSFERT
8A9                 ?B=0 A
```

```
E2                  GOYES L3
14A                 A=DAT1 B
3103                LCHEX 30
B6A                 A=A-C B
3190                LCHEX 09
9EA                 ?A≤C B
90                  GOYES L5
31?0                LCHEX 0?
B6A                 A=A-C B
1590      L5        DAT0=A 1
1?0                 D0=D0+1
161                 D1=D1+2
3120                LCHEX 00002
60DF                GOTO L4
8F*****   L3        GOSBVL LOAD.REG
142                 A=DAT0 A            fin de routine
164                 D0=D0+5
808C                PC=(A)
*****     Fin       DROP2
09F20               CON(5) 02F90        fin de structure
```

**Chaînes de codes:**

**HP28c 1BB:**

```
?6C20  07A20  60000  2B0D0  3D9F8  107A2  09000
0A0B0  21D20  30348  4B469  C20A?  0008F  2EE40
14313  21641  46164  D51?4  14313  11?91  43131
34500  00E18  A9E21  4A310  3B6A3  1909E  A9031
?0B6A  15901  ?0161  31206  0DF8F  91F40  14216
4808C  958?1  09F20
```

**Pour HP28c 1CC :**

```
?6C20  07A20  60000  200E0  39809  107A2  09000
0A0B0  21D20  30348  4B469  C20A?  0008F  2EE40
14313  21641  46164  D51?4  14313  11?91  43131
34500  00E18  A9E21  4A310  3B6A3  1909E  A9031
?0B6A  15901  ?0161  31206  0DF8F  91F40  14216
4808C  B39?1  09F20
```

**Pour HP28s 2BB :**

```
?6C20  E8971  0FEF2  EEE80  07A20  90000  A0B02
1D203  03484  B469C  20A?0  008F1  80501  43132
16414  6164D  51?41  43131  1?914  31313  45000
0E18A  9E214  A3103  B6A31  909EA  9031?  0B6A1
5901?  01613  12060  DF8F8  B0501  42164  808C4
46?00  9F20
```

# REVERSE

**REVERSE** est une fonction qui a pour but de renverser les chaînes de caractères. Ainsi "ABC" REVERSE renverra "BCA". Cette fonction est particulièrement utile avec PEEK: si on désire lire 5 quartets contenant une adresse, celle-ci sera renvoyée à l'envers'. REVERSE permettra de la remettre dans le bon sens et de la lire ainsi facilement...

Entrée du programme (lire l'avertissement page 189):

taper la chaîne des codes (située après le listing du programme), en une seule ligne, sans espace. Exécuter ensuite LASS et stocker le résultat dans 'REVERSE'...

*Remarque :* La séquence #1 #0 + #0A CHK est facultative.

**Listing du programme**

```
Codes     Labels  Mnémoniques     Commentaires
76C20             CON(5) 02C67    début de structure programme
07A20600001       #1              création des arguments de CHK
*****             #0
*****             +
07A2070000A0      #0A
21D20303484B4     CHK             vérification de l'objet
*****             DUP             recréation de l'objet
E4A2050000        ""
*****             +
*****             SWAP
*****             DROP
69C20             CON(5) 02C96    prologue assembly-code
46000     début   CON(5) fin-début
132               AD0EX           sauvegarde D0
102               R2=A
143               A=DAT1 A        A=adresse chaîne
133               AD1EX
103               R3=A            sauvegarde de D1
174               D1=D1+5
137               CD1EX
135               D1=C
143               A=DAT1 A        A=longueur de l'objet chaîne
C2                C=C+A A         C=adresse de fin de l'objet
134               D0=C
174               D1=D1+5         D1=adresse de début des carac.
181               D0=D0-2         D0=adr. dernier carac.
14B       label   A=DAT1 B        lecture de deux caractères
14E               C=DAT0 B
14D               DAT1=C B        réécriture dans l'ordre inverse
148               DAT0=A B
171               D1=D1+2         paire suivante
181               D0=D0-2
133               AD1EX
131               D1=A
```

```
136             CD0EX
134             D0=C
8BA             ?A#C A          y en a t-il encore?
FD              GOYES label     oui -> label
112             A=R2            non: récupération D0, D1
130             D0=A
113             A=R3
131             D1=A
142             A=DAT0 A        fin de routine
164             D0=D0+5
808C            PC=(A)
09F20   fin     CON(5) 02F90    fin de structure
```

**Chaînes de codes :**

**Pour HP28c version 1BB :**

```
'/6C20  07A20  60000  1B0D0  3D9F8  107A2  07000
0A021  D2030  3484B  41F'/  1E4A2  05000  0D9F8
1528'/  1F38'/  169C2  04600  01321  02143  13310
31'/41  3'/135  143C2  1341'/  41811  4B14E  14D14
81'/11  81133  13113  61348  BAFD1  12130  11313
11421  64808  C09F2  0
```

**Pour HP28c version 1CC:**

```
'/6C20  07A20  60000  100E0  39809  107A2  07000
0A021  D2030  3484B  43D8'/  1E4A2  05000  09809
1'/09'/  1129'/  169C2  04600  01321  02143  13310
31'/41  3'/135  143C2  1341'/  41811  4B14E  14D14
81'/11  81133  13113  61348  BAFD1  12130  11313
11421  64808  C09F2  0
```

**Pour HP28s version 2BB :**

```
'/6C20  07A20  60000  10FEF  2EEE8  007A2  07000
0A021  D2030  3484B  4CD5'/  0E4A2  05000  0EEE8
0016'/  0A26'/  069C2  04600  01321  02143  13310
31'/41  3'/135  143C2  1341'/  41811  4B14E  14D14
81'/11  81133  13113  61348  BAFD1  12130  11313
11421  64808  C09F2  0
```

# UNPIXEL

*Remarque :* Ce programme nécessite l'entrée préalable de PEEK, de LASS et de CHK (optionnel).

On se sert du programme PIXEL (en ROM) modifié, et la syntaxe est la même que PIXEL, l'effet obtenu étant l'extinction d'un point...

Voici la séquence de commandes à effectuer:

**Sur 28c 1BB:**

Si CHK est en mémoire, taper la chaîne en une seul ligne :
"76C2007A20600001B0D03D9F8
107A20700004021D20303484B4"

Sinon: "76C20"
Puis, en mode hexadécimal: #11D3C #F0 PEEK
On obtient la chaîne: "BA6402C...
Taper + "163B0A212009F20" + LASS
Le résultat sera rangé dans UNPIXEL.

**Sur 28c 1CC:**

Si CHK est en mémoire, taper la chaîne en une seule ligne:
"76C2007A2060000100E039809
107A20700004021D20303484B4"

Sinon: "76C20"
Puis, en mode hexadécimal: #11D73 #F0 PEEK
On obtient la chaîne: "BA6409F...
Taper + "953B0A212009F20" + LASS
Le résultat sera rangé dans UNPIXEL.

**Sur 28s 2BB:**

Si CHK est en mémoire, taper la chaîne:
"76C20F79710FEF2EEE80CA97121D20303484B4"

Sinon: "76C20"
Puis en mode hexadécimal: #246D4 #E1 PEEK
On obtient la chaîne: "00740B4...
Taper + "2FDA1A112009F20" + LASS
Le résultat sera rangé dans UNPIXEL.

---

## SEARCH

Ce programme utilise PEEK. Son rôle est de rechercher dans la ROM, les différentes occurences d'un motif (suite de quartets). En entrée, on donne une chaîne contenant la séquence de quartets à rechercher. En sortie, on obtient la liste des différentes adresses où se situe le motif.

Listing du programme (à entrer en mode hexadécimal):

```
« #0 #100 3 PICK SIZE OVER + -> MOTIF AD LEN LENP
  « {} DO AD DUP 1 DISP LENP PEEK
          IF MOTIF POS AD OVER
          THEN + DUP 'AD' STO 1 - DUP
               2 DISP 1000 .07 BEEP 1 ->LIST +
          ELSE + LEN + 'AD' STO
          END
       UNTIL AD #40000 ≥
       END CLMF » »
```

*Remarque :* le #100 situé en début de programme est à moduler selon la place mémoire disponible...

---

## CONTRAST

Ce programme utilise POKE. Il est à entrer en mode HEXADECIMAL.

*Argument en entrée:* entier binaire de #0 à #15 en décimal (cad de #0 à #F en hexadécimal) représentant le contraste désiré (#0 contraste le plus faible, #F le plus fort).

**Pour HP28c :**

```
« HEX ->STR 3 3 SUB #407101 SWAP POKE »
```

**Pour HP28s :**

```
« HEX ->STR 3 3 SUB #FFF01 SWAP POKE »
```

# IN/OUT

Ces deux programmes permettent l'échange de données entre deux HP28 (transformées comme décrit dans la seconde partie) ou entre une HP28 modifiée et un autre ordinateur...

**Premier programme**:
OUT qui permet l'émission d'une chaîne de caractères.

**OUT**

```
« ->STR DUP SIZE #0 + EMIT DROP WUS 1 WAIT
EMIT DROP »
```

Ce programme utilise les deux autres programmes WUS (qui attend un signal de la part de l'autre machine) et EMIT qui envoie un objet dont le codage est de la forme prologue/longueur (sur cinq quartets)/objet, c'est à dire une chaîne, un entier binaire... (voir la liste des objets dans le chapitre 3 de la première partie).

**WUS et EMIT :**
Ce sont deux programmes en langage-machine. Pour les entrer, on tape la chaîne des codes, en une seule ligne, sans espace, et on exécute le programme LASS. Le résultat (un ou plusieur System-Object) est stocké comme un objet normal...

**Explication de OUT** :
On commence par émettre la longueur de l'objet. WUS permet d'attendre que l'autre machine ait fini de réserver la place mémoire nécessaire au stockage de la chaîne, que l'on émet ensuite...

**Listing de WUS:**

```
Codes    Labels   Mnémoniques      Commentaires
69C20             CON(5) 02C96     début de l'objet assembly code
*****    début    CON(5) fin-début longueur de l'objet
8F*****           GOSBVL SAV.REG   sauvegarde des registres de travail
**..**            interdiction des interruptions clavier:
                                   - 28c  INTOFF
                                   - 28s  D1=C0035  LCHEX F  DAT1=C 1
3q*****           LCHEX masque     28c: 01000     28s: 00100
D5                B=C A
3q20000 lbl       LCHEX 00002      masque de sortie
**..**            lecture clavier:
                                   - 28c  OUT=C  C=IN
                                   - 28s  GOSBVL 01AD1
```

```
0EF5            C=C&B A
8AA             ?C=0 A          si rien en entrée -> lbl
**              GOYES lbl
**..**          autorisation des interruptions clavier:
                                - 28c  INTON
                                - 28s  LCHEX 0  DAT1=C 1
8F******        GOSBVL LOAD.REG  récup des registres de travail
142             A=DAT0 A         fin de routine
164             D0=D0+5
808C            PC=(A)
        fin
```

**Chaîne de codes de WUS :**

**Pour la HP28c :**

```
69C20  44000  8F2EE  40808  F3401  000D5  34200
00801  8030E  F58AA  CE808  08F91  F4014  21648
08C
```

**Pour la HP28s:**

```
69C20  25000  8F180  501F5  300C3  0F15D  03400
100D5  34200  008F1  DA100  EF58A  ABE30  015D0
8F8B0  50142  16480  8C
```

**Listing de EMIT:**

```
Codes    Labels  Mnémoniques     Commentaires
69C20            CON(5) 02C96    debut de l'objet programme-machine
09000    début   CON(5) fin-début  longueur de l'objet
8F******         GOSBVL SAV.REG sauvegarde des registres de travail
143              A=DAT1 A        A=adresse de l'objet dans la pile
131              D1=A
174              D1=D1+5         D1=adresse de la longueur de cet objet
143              A=DAT1 A
130              D0=A            D0=longueur de l'objet
185              D0=D0-6         D0=nbre de quartets à émettre moins 1
136              CD0EX
1B******         D0=infra rouge (28c: 4070D   28s: FFF0D)
174              D1=D1+5         D1=adresse mantisse de l'objet
108      L0      R0=C
1530             A=DAT1 P        lecture d'un quartet
3430000          LCHEX 00003     nbre de bits à émettre
D7               D=C A
```

```
3400000  L3      LCHEX 00000    longueur du bit 0   A REGLER (voir note)
V1
A04              A=A+A P
590              GONC L1        si le bit à émettre est un 0 -> L1
3400010          LCHEX 01000    longueur du bit 1   A REGLER (voir note)
V2
D5       L1      B=C A
301              LCHEX 1
1540             DAT0=C P       activation de l'émetteur IR
CD       L2      B=B-1 A        temporisation
5DF              GONC L2
D2               C=0 A          désactivation
1540             DAT1=C P
3400010          LCHEX 01000    attente entre 2 bits  A REGLER (voir note)
V3
CE       L4      C=C-1 A
5DF              GONC L4
CF               D=D-1 A        y a-t-il encore des bits à émettre pour ce
                                quartet ?
59C              GONC L3        oui -> L3
170              D1=D1+1        quartet suivant
118              C=R0
CE               C=C-1 A
5EA              GONC L0        si il y en a encore -> L0
8F******         GOSBVL LOAD.REG  récupération des registres de travail
142              A=DAT0 A       fin de routine
164              D0=D0+5
808C             PC=(A)
```

**Note :** Les valeurs V1, V2 et V3 sont à régler. La méthode de réglage est décrite après tous les listings. ATTENTION: ce réglage est absolument nécessaire, et est fonction des deux machines reliées entre elles...

### Chaîne de codes de EMIT

**Pour la HP28c :**

```
69C20  09000  8F2EE  40143  1311'7  41431  30185
1361B  D0'704  1'7410  81530  34300  00D'73  40000
0A045  90340  0010D  53011  540CD  5DFD2  15403
40001  0CE5D  FCF59  C1'701  18CE5  EA8F9  1F401
42164  808C
```

**Pour la HP28s :**

```
69C20  09000  8F180  50143  1311'7  41431  30185
1361B  D0FFF  1'7410  81530  34300  00D'73  40000
0A045  90340  0010D  53011  540CD  5DFD2  15403
40001  0CE5D  FCF59  C1'701  18CE5  EA8F8  B0501
42164  808C
```

**Second programme**:

IN qui permet de recevoir une chaîne de caractères.

**IN**

```
« #0 DUP + RECEP B->R CREAC MEM DROP IR1 RECEP
»
```

RECEP : permet de remplir un objet (entier binaire, chaîne) qui est dans la pile.

CREAC: crée une chaîne de longueur spécifiée.

IR1 : envoie un court signal IR pour demander l'émission à l'autre machine (symétrique de WUS)

IN fonctionne donc ainsi: on reçoit la longueur de la chaîne à recevoir; on crée une chaîne d'espaces destinée à recevoir la chaîne qui va être émise. Enfin on reçoit la dite chaîne...

**CREAC** est un programme RPL simple dont voici le listing:

```
« -> L « " " 1 L LN 2 LN / CEIL
   START DUP +
   NEXT 1 L SUB »
```

**IRI** est le programme machine que voici:

```
Codes     Labels  Mnémoniques      Commentaires
69C20             CON(5) 02C96     prologue de l'objet assembly code
E3000     début   CON(5) fin-début  longueur
8F*****           GOSBVL SAV.REG   sauveg des regist de travail de la 28
1B*****           D0=Infra rouge   28c: 40/0D    28s: FFF0D
301               LCHEX 1
15C0              DAT0=C 1         activation de l'infra-rouge
3400100           LCHEX 00100
CE        L1      C=C-1 A          délai
5DF               GONC L1
300               LCHEX 0
15C0              DAT0=C 1         désactivation
8F*****           GOSBVL LOAD.REG  récupération des registres de travail
142               A=DAT0 A         fin de routine
164               D0=D0+5
808C              PC=(A)
```

**Chaîne de codes de IR1**

**Pour la HP28c :**

```
69C20  E3000  8F2EE  401BD  0/043  0115C  03400
100CE  5DF30  015C0  8F91F  40142  16480  8C
```

**Pour la HP28s :**

```
69C20  E3000  8F180  501BD  0FFF3  0115C  03400
100CE  5DF30  015C0  8F8B0  50142  16480  8C
```

## Listing de RECEP :

```
Codes     Labels   Mnémoniques      Commentaires
69C20              CON(5) 02C96     début de l'objet assembly-code
*****     début    CON(5) fin-debut longueur
8F*****            GOSBVL SAV.REG   sauvegarde des registres de travail
**..**             inhibition des interruptions:
                                    - 28c   INTOFF
                                    - 28s   D0=C0035  LCHEX F  DAT0=C 1
143                A=DAT1 A
131                D1=A             D1=adresse objet au niveau 1
174                D1=D1+5
143                A=DAT1 A         A=longueur objet
174                D1=D1+5          D1=adresse où on va écrire
130                D0=A
185                D0=D0-6          nbr de quartets à recevoir moins un
34*****            LCHEX masque de sortie  28c 00010    28s 00100
D5                 B=C A
136                CD0EX
108       L0       R0=C
AF3                D=0 W
2F                 P=15
304                LCHEX 4          nbre de bits par quartets
AC7                D=C S
20                 P=0
3420000   L1       LCHEX 00002      masque de sortie
**..**             échantillonage clavier:
                                    - 28c OUT=C  C=IN
                                    - 28s GOSBVL 01AD1
0EF5               C=C&B A          attente du début d'un signal
8AA                ?C=0 A
**                 GOYES L1
D0                 A=0 A
E4        L2       A=A+1 A          attente de la fin + mesure longueur
3420000            LCHEX 00002
**..**             échantillonage clavier
0EF5               C=C&B A
8AE                ?C#0  A
**                 GOYES L2
A07                D=D+D P
3405000            LCHEX discriminant  A REGLER voir explications    V4
8B2                ?A<C A           comparaison pour savoir si c'est un 1 ou un
50                 GOYES L3
B07                D=D+1 P
A4F       L3       D=D-1 S          un bit de moins à recevoir
94F                ?D#0  S          encore ?
**                 GOYES L1         oui -> L1
A8B                C=D P
1550               DAT1=C P         écriture du quartet reçu
170                D1=D1+1          quartet suivant
118                C=R0
CE                 C=C-1 A
5**                GONC L0          si il y en a encore à recevoir -> L0
**..**             autorisation des interruptions:
                                    - 28c   INTON
                                    - 28s   D0=C0035   LCHEX 0   DAT0=C 1
8F*****            GOSBVL LOAD.REG récupération des registres de travail
142                A=DAT0 A         fin de routine
164                D0=D0+5
808C               PC=(A)
          fin
```

**Chaîne de codes de RECEP :**

**Pour la HP28c :**

```
69C20  2B000  8F2EE  40808  F1431  311'4  1431'
41301  85340  1000D  51361  08AF3  2F304  AC'20
34200  00801  8030E  F58AA  CED0E  43420  00080
18030  EF58A  EAEA0  '3405  0008B  250B0  'A4F9
4F8BA  8B155  01'01  18CE5  69808  08F91  F4014
21648  08C
```

**Chaîne de codes de RECEP.**

**Pour la HP28s :**

```
69C20  8C000  8F180  501B5  30003  0F15C  01431
311'4  1431'  41301  85340  0100D  51361  08AF3
2F304  AC'20  34200  008F1  DA100  EF58A  ABED0
E4342  00008  F1DA1  00EF5  8AE9E  A0'34  05000
8B250  B0'A4  F94F6  BA8B1  5501'  0118C  E5491
B5300  C3001  5C08F  8B050  14216  4808C
```

Il y a certains réglages à effectuer: un bit 0 est représenté par un signal court, le 1 par un signal long; il faut donc régler la longueur de ces signaux, de manière à ce que les deux machines les reconnaissent...

**Réglage des valeurs V1, V2, V3 et V4:**

- **Première étape :**
  Taper le programme d'émission et celui de réception, un sur chaque machine; garder les chaînes de codes de RECEP et de EMIT: ces deux chaînes vont être à modifier.

- **Seconde étape :** les réglages proprement dits:

  1) **Réglage de V1** , longueur du signal court.

     **Du côté récepteur :**
     placer la chaîne " " dans la pile et déclencher RECEP

     **Du côté émetteur :**
     mettre la valeur V1 à 00000, coder la chaîne grâce à LASS et envoyer le caractère 0 par 0 CHR EMIT

  Si du côté récepteur, le programme RECEP s'arrête, tout est OK. Sinon, effectuer un arrêt système et recommencer l'opération en augmentant progressivement V1.

  *Remarque :* la valeur 00000 doit suffire...

### 2) Réglage de V4, discriminant.

V4 est le discriminant permettant de faire la différence entre un signal court et un signal long. Pour le régler, on va le diminuer jusqu'à ce que le récepteur ne reconnaisse que des 1, puis on le réaugmentera de quelques unités.

On émet donc le caractère 0 par 0 CHR EMIT tandis que, du côté récepteur on reçoit avec des valeurs décroissantes de V4. Lorsqu'on reçoit le caractère 255 au lieu de 0 (on obtient la valeur du caractère par l'instruction NUM) cela veut dire que l'on a dépassé la valeur correcte du discriminant. On fixera celui-ci à la valeur obtenue plus huit...

### 3) Réglage de V2, longueur d'un signal long.

Le principe est le même: on émet le caractère 255 en diminuant progressivement V2. Quand on reçoit le caractère 0 du côté récepteur, c'est que V2 est devenu trop faible. On le fixe alors à la valeur obtenue plus huit.

### 4) Réglage de V3, temps entre deux bits.

Le programme récepteur étant légèrement plus lent que le programme émetteur, il faut lui laisser du temps... On diminuera donc V3 jusqu'à ce que le récepteur ne fonctionne plus correctement (perte de bits, bits mal reçus), puis on le réaugmentera de quelques unités...

Une fois les quatre valeurs réglées, on peut transmettre des chaînes entières grâce à IN et OUT....

*Remarques:* Il est recommandé de noter les valeurs obtenues pour V1, V2, V3 et V4 car elles dépendent des machines en présence...

La HP28 prend, toutes les minutes, quelques instants pour réactualiser l'horloge. Ceci peut fausser la transmission. Il est donc recommandé de ne faire que de courtes émissions (pas plus de 1000 caractères) et de vérifier les résultats...

Pour réaliser des communications avec d'autres ordinateurs, les programmes IN et OUT restent les mêmes, mais il faut en réaliser pour les autres machines...

Les vitesses atteintes entre deux HP28 sont de l'ordre de 700 bauds (c'est à dire environ 87 caractères par secondes), de HP28 vers un plus gros système, on peut facilement atteindre des vitesses de l'ordre des 7000 bauds (environ 875 caractères par seconde); c'est en effet la réception qui conditionne la vitesse (en règle générale, un programme de réception est moins rapide qu'un programme d'émission). On peut envisager des liaisons via téléphone, radio (...) mais il faut concevoir les interfaces électroniques nécessaires.

Pour simplifier les programmes, il n'y a pas de vérification des données reçues. On pourrait facilement réaliser un programme effectuant une somme de contrôle, et demandant une réémission, mais il serait beaucoup plus long....

---

# TELECRAN

Ce programme utilise POKE, UNPIXEL, LCD-> et ->LCD

Il simule un mini-télécran sur la HP28. Les touches de commande sont:

**INS** pour terminer la session (sauve l'écran dans la pile).
**DEL** pour basculer entre le modes écriture (indicateur Alpha allumé) et effacement (indicateur Alpha éteint).
**Les quatres flèches** pour se déplacer.

**Démarrage du programme :**

Lancer le programme TELECRAN pour travailler sur un écran vide, lancer TELE pour travailler sur l'écran actuel, ou TEL pour travailler sur l'écran sauvé dans la pile...

Listing des programmes (se placer en mode HEXADECIMAL pour les entrer)

**ALPHON**

```
Pour HP28c « #40008 "FFFFF000" POKE »
Pour HP28s « #FF808 "FFFFF000" POKE »
```

**ALPHOFF**

```
Pour HP28c « #40008 "00000000" POKE »
Pour HP28s « #FF808 "00000000" POKE »
```

**TELAFF**

```
«X Y R->C IF DR 1 == THEN PIXEL ELSE UNPIXEL
END »
```

**TELINS**

```
«TELAFF LCD-> {X Y DR PPAR} PURGE CLMF ABORT»
```

**TELDEL**

```
«1 DR - DUP 'DR' STO IF 1 == THEN ALPHON ELSE
APLHOFF END»
```

**TELRIGHT**

```
«TELAFF X 1 + X 131 == - 'X' STO »
```

**TELLEFT**

```
«TELAFF X 1 - X 0 == + 'X' STO »
```

**TELDOWN**

```
«TELAFF Y 1 - Y 0 == + 'Y' STO »
```

**TELUP**

```
«TELAFF Y 1 + Y 32 == - 'Y' STO »
```

**TELE**

```
« {1,1} PMIN {131,32} PMAX 68 'X' STO 16 'Y'
STO 0 'DR' STO ALPHOFF DO "TEL" 1 DO DROP
"[INS[DEL[UP[RIGHT[DOWN[LEFT[" DO X Y R->C
DUP PIXEL UNPIXEL UNTIL KEY END UNTIL "["
DUP2 + + ROT SWAP POS END + STR-> UNTIL 0 END
»
```

**TEL**

```
« ->LCD TELE »
```

**TELECRAN**

```
« CLLCD TELE »
```

# DESASS (ttv)

Ce programme passionnant ne nécessite pas ou peu de commentaires. C'est un programme qui est monstrueux par sa taille mais il désassemble tout programme en langage machine.

Il suffit de lui donner dans le niveau un de la pile une chaîne de caractères contenant la liste des codes hexadécimaux les uns à la suite des autres de la façon habituelle. Il n'est pas très rapide; il pourrait être optimiser aussi bien sur le plan de la longueur que de la vitesse, mais il marche! Tous les autres programmes sont annexes. La solution est sauvegardée dans SOL. En même temps que le programme, un exemple est donné. Ce programme ne désassemble que des codes machines, pas les prologues.

```
DESASS
« HEX 64 STWS 'Z' STO "       - DEBUT -" 10 CHR +
 'SOL' STO 1 'P' STO Z SIZE -> S
  « DO P 'I' STO L READ 1 + GET EVAL + STOS
    UNTIL P S >
    END
    "        - FIN - " STOS CLMF
  »
»

SOL       (A ne pas taper, ceci est un exemple).
"
133       AD1ex
103       R3=A
133       AD1ex
132       AD0ex
102       R2=A
AFC       BAex    W
101       R1=A
174       D1=D1+5
143       A=DAT1  A
133       AD1ex
179       D1=D1+10
143       A=DAT1  A
132       AD0ex
113       A=R3
133       AD1ex
143       A=DAT1  A
133       AD1ex
174       D1=D1+5
```

```
143       A=DAT1  A
DC        BAex    A
1'4       D1=D1+5
3450000   LCHEX   00005
808C      PC=(A)
        - FIN -
"
```

**Z** (A ne pas taper, ceci est un exemple).

```
"133103133132102AFC1011'41431331'9143132113133143133
'4143DC1'43450000808C"
```

**L**

```
{ A0 A1 A2 A3 A4 A5 A6 A7 A1 A9 AA AB AC AC AC AC }
```

**A0**

```
« INC READ DUP
     IF 14 ≠
     THEN { "RTNSXM" "RTN" "RTNSC" "RTNCC"
          "SETHEX" "SETDEC" "RSTK=C" "C=RSTK"
          "CLRST" "C=ST" "ST=C" "CSTex"
          "P=P+1" "P=P-1" 14 "RTI" }
          SWAP 1 + GET CODE SWAP
     ELSE DROP INC READ INC READ -> x y
          « y 8 < 38 CHR 33 CHR IFTE -> z
            « y 8 MOD 2 * 1 + "ABBCCADCBACBACCD" -> t u
              « u t DUP SUB u t 1 + DUP SUB -> a b
                « CODE a "=" a z b + + + + SPC2 +
                  IF x 15 ==
                  THEN "A"
                  ELSE x CH
                  END
                »
              »
            »
          » END
»
```

**A1**

```
« { N M } "18" READ ->STR POS GET INC
  READ 1 + GET EVAL
»
```

```
N
{ C0 C0 C0 C0 C4 C4 C6 C6 C6 C9 C9 C9 C6 C9 C9 C9 }

C0
« TAKE INC CODE "P"
3 ROLL + STR-> READ 1 + GET »

C4
« READ INC READ -> x y
  « { "DAT0=A" "DAT1=A" "A=DAT0" "A=DAT1"
    "DAT0=C" "DAT1=C" "C=DAT0" "C=DAT1" }
     y 8 MOD 1 + GET SPC2
    IF x 4 ==
    THEN
      IF y 8 <
      THEN "A"
      ELSE "B"
      END
    ELSE INC READ -> z
      «
        IF y 8 <
        THEN z CH
        ELSE READ 1 + ->STR
        END
      »
    END +
  » CODE SWAP
»

C6
« { "D0=D0+" "D1=D1+" "D0=D0-"
  "D1=D1-" } READ 5 - DUP 4 > 3 * - GET
  INC CODE SWAP READ 1 + ->STR +
»

C9
« READ 8 - DUP
  IF 3 >
  THEN 4 - "D1= "
  ELSE "D0= "
  END { 2 4 5 } 3 ROLL GET 1 - -> x
  « INC Z P DUP x + SUB REVERSE + P x + 'P' STO
    CODE SWAP
  »
»

P0
{ "R0=A" "R1=A" "R2=A" "R3=A" "R4=A" 5 6 '
  "R0=C" "R1=C" "R2=C" "R3=C" "R4=C" }
```

**P1**

```
{ "A=R0" "A=R1" "A=R2" "A=R3" "A=R4" 5 6 7
  "C=R0" "C=R1" "C=R2" "C=R3" "C=R4" }
```

**P2**

```
{ "AR0ex" "AR1ex" "AR2ex" "AR3ex" "AR4ex" 5 6 7
  "CR0ex" "CR1ex" "CR2ex" "CR3ex" "CR4ex" }
```

**P3**

```
{ "D0=A" "D1=A" "AD0ex" "AD1ex" "D0=C" "D1=C"
  "CD0ex" "CD1ex" "D0=AS" "D1=AS" "AD0XS" "AD1XS"
  "D0=CS" "D1=CS" "CD0XS" "CD1XS" }
```

**A2**

```
« INC CODE "p= " READ ->STR + »
```

**A3**

```
« INC READ -> x
  « "LCHEX " SPC2 2 INC P DUP x + DUP 'P' STO SUB REVERSE
    + CODE SWAP
  »
»
```

**A4**

```
« INC TAKE INC TAKE + DUP
  IF "00" ==
  THEN DROP "RTNC"
  ELSE DUP
    IF "20" ==
    THEN DROP "NOP3"
    ELSE REVERSE "GOC" SPC2 SWAP +  END
  END CODE SWAP
»
```

**A5**

```
« INC TAKE INC TAKE + DUP
  IF "00" ==
  THEN DROP "RTNNC"
  ELSE REVERSE "GONC" SPC2 SWAP +
  END CODE SWAP
»
```

**A6**

```
« Z INC P DUP 3 + SUB DUP
  IF 1 3 SUB "300" ==
  THEN DROP "NOP4"
  ELSE DUP
    IF "4000" ==
    THEN DROP "NOP5" INC
    ELSE 1 3 SUB REVERSE "GOTO" SPC2 SWAP +
    END
  END INC INC CODE
  SWAP
»
```

**A7**

```
« "GOSUB" SPC2 "" 1 3 START INC TAKE + NEXT
  REVERSE + CODE SWAP
»
```

**M**

```
{ B0 B1 B1 B3 B4 B4 B6 B6 B6 B6 BA BA BC BC BC BC }
```

**B0**

```
« U0 INC READ -> x
  « x 1 + GET
    IF x 8 ==
    THEN DROP V0 INC READ 1 + GET
    ELSE IF { 15 13 12 } x POS THEN SPC2 INC TAKE + END
    END CODE SWAP
    IF x 8 == READ DUP 11 == SWAP 10 == OR AND
    THEN GOYES
    END »
»
```

**B1**

```
« TAKE INC CODE "U" 3 ROLL + STR-> READ 1 + GET »
```

**B3**

```
« B1 GOYES »
```

**B4**

```
« { "ST=0" "ST=1" } READ 3 - GET INC SPC2 READ ->STR +
  CODE SWAP »
```

**B6**

```
« { "?ST=0" "?ST≠0" "?P≠" "?P=" } READ 5 - GET INC SPC2
  READ ->STR + CODE SWAP GOYES »
```

**BA**

```
« READ INC READ -> x y
  « CODE
    IF x 10 ==
    THEN A
    ELSE B
    END y 1 + GET SPC2 + "A"
    GOYES
  »
»
```

**BC**

```
« { "GOLONG" 4 "GOVLNG" 5 "GOSUBL" 4
  "GOSBVL" 5 } READ 2 * 23 - DUP 1 + SUB
  LIST-> DROP -> a b
    « a SPC2 Z P 1 + DUP b + 1 - SUB
      REVERSE + P b + 'P' STO CODE SWAP
    »
»
```

**U0**

```
{ "OUT=CS" "OUT=C" "A=IN" "C=IN"
  "UNCNFG" "CONFIG" "C=ID" "SHUTDN" 8 "C+P+1" "RESET"
  "BUSCC" "C=P" "P=C" E "CPex"}
```

**V0**

```
{ "INTON" 1 2 3 4 5 6 7 8 9 "?CBIT=0" "?CBIT=1"
"PC=(A)" "BUSCD" E "INTOFF" }
```

**U1**

```
{ "ASLC" "BSLC" "CSLC" "DSLC" "ASRC" "BSRC" "CSRC"
 "DSRC" 8 9 10 11 "ASRB" "BSRB" "CSRB" "DSRB" }
```

**U2**

```
{ 0 "XM=0" "SB=0" 3 "SR=0" 5 6 7 "MP=0"
9 10 11 12 13 14 "CLRHST" }
```

**U3**

```
{ 0 "?XM=0" "?SB=0" 3 "?SR=0" 5 6 7 "?MP=0" }
```

**A9**

```
« A B NORMAL GOYES »
```

**AA**

```
« C D NORMAL »
```

**AB**
```
« E F NORMAL »
```

**AC**
```
« { C D E F } READ 11 - GET EVAL INC CODE SWAP READ 1 + GET
SPC2 "A" + »
```

**A**
```
{ "?A=B" "?B=C" "?C=A" "?D=C" "?A≠B" "?B≠C" "?C≠A" "?D≠C"
"?A=0" "?B=0" "?C=0" "?D=0" "?A≠0" "?B≠0" "?C≠0" "?D≠0" }
```

**B**
```
{ "?A>B" "?B>C" "?C>A" "?D>C" "?A<B" "?B<C" "?C<A" "?D<C"
"?A≥B" "?B≥C" "?C≥A" "?D≥C" "?A≤B" "?B≤C" "?C≤A" "?D≤C" }
```

**C**
```
{ "A=A+B" "B=B+C" "C=C+A" "D=D+C" "A=A+A" "B=B+B" "C=C+C"
"D=D+D" "B=B+A" "C=C+B" "A=A+C" "C=C+D" "A=A-1" "B=B-1"
"C=C-1" "D=D-1" }
```

**D**
```
{ "A=0" "B=0" "C=0" "D=0" "A=B" "B=C" "C=A" "D=C" "B=A" "C=B"
"A=C" "C=D" "ABex" "CBex" "CAex" "CDex" }
```

**E**
```
{ "A=A-B" "B=B-C" "C=C-A" "D=D-C" "A=A+1" "B=B+1" "C=C+1"
"D=D+1" "B=B-A" "C=C-B" "A=A-C" "C=C-D" "A=B-A" "B=C-B"
"C=A-C" "D=C-D" }
```

**F**
```
{ "ASL" "BSL" "CSL" "DSL" "ASR" "BSR" "CSR" "DSR" "A=-A"
"B=-B" "C=-C" "D=-D" "A=-A-1" "B=-B-1" "C=-C-1" "D=-D-1" }
```

**SPC**
```
"       " (7 espaces)
```

**SPC1**
```
« SPC 1 7 4 PICK SIZE - SUB + " " + »
```

**SPC2**
```
« SPC 1 6 4 PICK SIZE - SUB + " " + »
```

**TAKE**
```
« Z P DUP SUB »
```

**READ**

```
« "╬" Z P DUP SUB + STR-> B->R »
```

**CODE**

```
« Z I P SUB SPC1 »
```

**GOYES**

```
« + INC TAKE INC TAKE + -> a
  « 10 CHR a
    IF a "00" ==
    THEN SPC1 "RTNYES"
    ELSE SPC1 "GOYES" SPC2 + a REVERSE
    END + +
  »
»
```

**NORMAL**

```
« -> a b
  « INC READ INC READ -> x y
    « CODE IF x 8 < THEN a ELSE b END
      y 1 + GET SPC2 x CH +
    »
  »
»
```

**CH**

```
« -> a
  « { "P" "WP" "XS" "X" "S" "M" "B" "W" }
    a 8 MOD 1 + GET »
»
```

**REVERSE**

```
« -> c « "" c SIZE 1 FOR x c x DUP SUB + -1 STEP » »
```

**INC**

```
« 1 'P' STO+ »
```

**STOS**

```
« "" 2 DISP DUP 1 DISP SOL SWAP + 10 CHR
 + 'SOL' STO INC »
```

# LCD->

Ce programme permet la sauvegarde de l'écran de la HP28, sous forme de chaîne dans la pile.

Entrée du programme (lire avertissemnet page 189):
Taper la chaîne de codes en une seule ligne, sans espace, et exécutez LASS. Le résultat (des System Object) sera stocké dans 'LCD->'

**Chaînes de codes:**

**Pour HP28 1BB:**

```
'6C20  C53C1  068B0  4'D60  8B430  92B30  0'8B0
4'D60  8B430  92B30  CD430  088B0  4'D60  8B430
92B30  CD430  098B0  4'D60  8B430  92B30  CD430
09F20
```

**Pour HP28 1CC:**

```
'6C20  844C1  858B0  4'D60  8B430  92B30  868B0
4'D60  8B430  92B30  CD430  8'8B0  4'D60  8B430
92B30  CD430  888B0  4'D60  8B430  92B30  CD430
09F20
```

**Explications** :

Pour réaliser le LCD-> on se sert des sous routines de PRLCD qui renvoient l'écran ligne par ligne, sous forme de chaînes dans la pile. Cependant il faut supprimer dans chacune de ces chaînes, les caractères de contrôle qui font passer l'imprimante en mode graphique (ce sont les 3 premiers caractères de chaque chaîne). De plus on additionne ensemble les 4 chaînes ainsi créées...

---

# ->LCD

Ce programme effectue l'opération inverse de LCD->: il prend une chaîne dans la pile et en fait un écran...

Entrée du programme (lire avertissement page 189):
Ttaper la chaîne de codes (située apres le listing), en une seule ligne, sans espace, et exécuter LASS. Stocker le résultat (#1 #0 + #0A CHK System Object System Object DROP) dans ->LCD.

**Listing du programme:**

```
Codes    Labels   Mnémoniques      Commentaires
76C20             CON(5) 02C67     début de structure programme
07A206000001      #1               )
07A206000000      #0               )
*****             +                )optionnel (vérif des objets dans la
07A20700000A0     #0A              )            pile)
21D20303484B4     CHK              )
*****             figeage de l'écran
69C20             CON(5) 02C96     début de programme assembleur
5A000    début    CON(5) fin-début longueur du code
8F2EE40           GOSBVL SAV.REG   sauvegarde D0,D1,B et D
143               A=DAT1 A
133               AD1EX            D1=adresse de la chaîne
174               D1=D1+5
143               A=DAT1 A         A=longueur de la chaîne
3450000           LCHEX 00005
EA                A=A-C A
D8                B=A A            B=2* nbr de caractères de la
chaîne
174               D1=D1+5          D1=adresse du premier caractère
1B87004           D0=40078         D0=adresse du début de l'écran
132               AD0EX
130      lbl0     D0=A
100               R0=A
8A9      lbl1     ?B=0 A           a-t-on fini ? (chaîne vide?)
B5                GOYES lbl3       si oui -> lbl3
14B               A=DAT1 B         sinon: lecture d'un caractère
148               DAT0=A B         écriture dans l'écran
171               D1=D1+2          caractère suivant
167               D0=D0+8          position suivante dans l'écran
CD                B=B-1 A
CD                B=B-1 A          décrémentation du nbr de
caractères
132               AD0EX
130               D0=A
340E204           LCHEX 402E0
8B2               ?C>A A           est-t-on dans le premier écran ?
A1                GOYES lbl2
```

```
3400404          LCHEX 40400
8BE              ?CSA A          est-on dans le second ?
E0               GOYES lbl2
3402100          LCHEX 00120     sinon: incrémentation
CA               A=A+C A         (passage du premier au second écran)
130              D0=A
340E504 lbl2     LCHEX 405E0     a-t-on fini une ligne?
8B2              ?A<C A          si tel n'est pas le cas:
7B               GOYES lbl1      on va en lbl1
110              A=R0            sinon:
E4               A=A+1 A         ligne suivante
E4               A=A+1 A
3408004          LCHEX 40080     a-t-on fait moins de 4 lignes ?
8B2              ?A<C A
E9               GOYES lbl0      si oui: ligne suvante
8F91F40 lbl3     GOSBVL LOAD.REG récupération de D0,D1,B et D
142              A=DAT0 A        fin de routine
164              D0=D0+5
808C             PC=(A)
*****   fin      CON(5) DROP     on droppe la chaîne
09F20            CON(5) 02F90    fin de structure
```

**Chaînes de codes:**

**Pour HP28c 1BB:**

```
76C20  07A20  60000  107A2  06000  00D9F  8107A
20700  00A02  1D203  03484  B491C  7069C  205A0
008F2  EE401  43133  17414  33450  000EA  D8174
1B870  04132  13010  08A9B  514B1  48171  167CD
CD132  13034  0E204  8B2A1  34004  048BE  E0340
2100C  A1303  40E50  48B27  B110E  4E434  08004
8B2E9  8F91F  40142  16480  8CF38  7109F  20
```

**Pour HP28c 1CC:**

```
76C20  07A20  60000  107A2  06000  00980  9107A
20700  00A02  1D203  03484  B4E4C  7069C  205A0
008F2  EE401  43133  17414  33450  000EA  D8174
1B870  04132  13010  08A9B  514B1  48171  167CD
CD132  13034  0E204  8B2A1  34004  048BE  E0340
2100C  A1303  40E50  48B27  B110E  4E434  08004
8B2E9  8F91F  40142  16480  8C129  7109F  20
```

## AFL1

**Pour HP28c uniquement.**

Ce programme utilise POKE et AFL2. Il permet de travailler avec seulement une ligne d'affichage, ce qui économise les piles...

Listing du programme (à entrer en mode HEXADECIMAL)

```
« AFL2 #40678 1 8 START DUP "00000000" POKE 16 + NEXT DROP
»
```

Pour revenir à l'état normal: faire un arrêt système...

---

## AFL2

**Pour HP28c uniquement.**

Ce programme utilise POKE. Il permet de travailler avec seulement deux lignes d'affichage, ce qu'économise les piles...

Listing du programme (à entrer en mode HEXADECIMAL)

```
« #405E0 1 16 START DUP "00000000" POKE 16 + NEXT DROP »
```

Pour revenir à l'état normal: faire un arrêt système...

---

## DBLMENU

**Pour HP28c uniquement.**

Ce programme utilise PEEK et POKE. Il permet de travailler avec une double ligne de menu (la ligne normale étant dupliquée en ligne 1)

Listing du programme (à entrer en mode HEXADECIMAL) :

```
« #405E0 #40658 1 8 START DUP2 #8 PEEK POKE 16 - SWAP 16 +
SWAP NEXT DROP2 »
```

Pour revenir à l'état normal: faire un arrêt système...

---

# ECROFF/ECRON

**28s exclusivement**

Ces deux programmes permettent l'extinction (ECROFF) et le rallumage (ECRON) du calculateur sans en arrêter le fonctionnement. Ainsi le programme:

```
« ECROFF 1 50 START 1400 .0'/ BEEP NEXT ECRON »
```

Eteindra l'écran (le calculateur semblera éteint) puis émettra 50 BEEPs et rallumera l'afficheur...

Entrée du programme: taper la chaîne des codes, en une seule ligne, sans espace, puis exécuter LASS. Stocker le résultat (System Object) dans ECRON ou ECROFF.

Listing des programmes:Remarque:les deux programmes étant très similaires, on ne donne que le listing de ECROFF avec en commentaire, la modification qui en fait le programme ECRON.

*Remarque* : Si vous avez déja tapé le programme POKE, les deux programmes sont inutiles car ils peuvent être remplacés par les deux programmes RPL:

**ECROFF** « #FFF03 "0" POKE »
**ECRON** « #FFF03 "A" POKE »

```
Codes     Labels  Mnémoniques      Commentaires
69C20             CON(5) 02C96     début de l'objet pgm LM
32000     deb     CON(5) fin-deb   longueur de code
133               AD1EX            sauvegarde de D1 dans A
1F30FFF           D1=FFF03
300               LCHEX(1) 0       pour allumer:A
15D0              DAT0=C 1
131               D1=A             récupération de D1
142               A=DAT0 A         fin de prgm LM
164               D0=D0+5
808C              PC=(A)
          fin
```

**Chaînes de codes :**

**Pour ECROFF:**

```
69C20  32000  1331F  30FFF  30015  D0133  14216
4808C
```

**Pour ECRON:**

```
69C20  32000  1331F  30FFF  30A15  D0133  14216
4808C
```

# SPEED

**28s exclusivement**

Ce programme rend la HP28s deux fois plus rapide.

**ATTENTION**: il y a modification du BEEP de WAIT... (les temps sont deux fois plus courts, la fréquence est altérée...) L' effet bénéfique de ce programme est supprimé par tout appui sur ON...Si vous avez tapé le programme POKE, voici une première version:

```
« #FFF00 "F" POKE »
```

Sinon : entrer la chaîne de codes (en une seule ligne, sans espace), exécuter LASS et stocker le résultat dans SPEED.

**Listing du programme :**

```
Codes       Labels  Mnémoniques      Commentaires
69C20               CON(5) 02C96     Prologue programme LM
32000       deb     CON(5) fin-deb   longueur code
133                 AD1EX            Sauvegarde D1 dans A
1F00FFF             D1=FFF00
30F                 LCHEX(1) F
15D0                DAT1=C 1
131                 D1=A             Récupération de D1
142                 A=DAT0 A         Fin de pgm LM
164                 D0=D0+5
808C                PC=(A)
            fin
```

**Chaîne de codes :**

```
69C20  32000  1331F  00FFF  30F15  D0131  14216
4808C
```

*Remarque :* On peut faire un prgm SPEEDN qui repasse en vitesse normale en remplaçant LCHEX(1) F (30F) par LCHEX(1) 8 (308). Pour des valeurs de 9 à E on accélère plus ou moins la 28s. De 0 à 7 on la ralentit jusqu'à quatre fois. Il est intéressant de savoir que 0 a le même effet que 1...

## PCAR (ttv)

Il calcule le polynome caractéristique d'une matrice. Il ne vous reste plus qu'à utiliser LAGU pour trouver les valeurs propres.

1: ARRAY (n,n) -> 1: 'poly. car'

```
« DUP IDN DUP SIZE LIST-> DROP2 -> M I N
  « 0 N FOR X M I X * - DET
       NEXT
       N 1 + 1 ->LIST ->ARRY N 1 + IDN 0 N
       FOR X 0 N
           FOR Y X 1 + Y 1 + 2 ->LIST X Y ^ PUT
           NEXT
       NEXT
       / ARRY-> DROP 0 N 0
       FOR X SWAP 'X' X ^ * +
       -1 STEP »
»
```

*Remarque :* En supprimant la séquence ARRY-> ... STEP, on obtient le polynome caractéristique sous forme de vecteur utilisable par LAGU.

---

## LAGU (ttv)

Il trouve toutes les racines réelles et complexes d'un polynôme à coefficients réels ou complexes. Vous placez en 1: le polynome sous forme de vecteur en puissances décroissantes des x^i 1: [ an ... a0 ], le coefficient ai correspond au coefficient devant le terme x^i. Le programme vous informera des différentes étapes qu'il rencontre et vous rendra à la fin dans le niveau 1: de la pile la liste des racines dont le nombre sera égale au degré du polynome.

L'algorithme employé est simple on fixe Z0 de la façon que l'on veut (à défaut d'une approximation de la racine on peut mettre 0 ou la valeur de la racine précédement trouvée (au cas ou elle serait multiple)) et on calcul Zk+1 = Zk + Sk où Sk est l'étape de LAGUERRE donnée par la formule:

```
Sk = -n P(Zk)/{ P'(Zk) + E [ ((n-1)P'(Zk))^2 - n
     (n-1) P(Zk) P''(Zk)]^(1/2)
```

n étant le degré du polynome

P le polynome

P' sa dérivée

P'' sa dérivée seconde

E étant à choisir = +1 ou -1 pour rendre le dénominateur le plus grand possible pour que l'étape de Laguerre soit la plus petite possible.

LAGU utilise les programmes VAL DER et DIV.

**LAGU**

```
« CLLCD {} 'SOL' STO 0 'Z' STO
  DO  DUP DUP2 'P' STO 1 DISP 'Z' VAL SWAP DER DUP
      'Z' VAL SWAP DER 'Z' VAL P SIZE LIST-> -
      DUP 1 - DUP SQ 3 PICK 3 PICK * NEG
    -> P0 P1 P2 N M A B
    « "Racine n°" N ->STR + 2 DISP Z
      WHILE   DUP 'Z' STO 3 DISP P0 EVAL DUP ABS 1E-10 >
      REPEAT  P1 EVAL P2 EVAL -> R S T
            « S A S SQ * B R T * * + √
              DUP2 DUP2 + ABS 3 ROLLD
              - ABS ≥ 2 * 1 - * + DUP
               IF ABS 0 ==
               THEN 50 .1 BEEP DROP RAND 40 * 20 -
                    RAND 40 * 20 - R->C "/0 Nouveau Z0"
                    2 DISP
               ELSE N NEG R * SWAP / Z +
               END
         »
      END DROP
    »
    SOL Z 1 ->LIST + 'SOL' STO P 1 Z NEG 2 1 ->LIST
    ->ARRY DIV DROP
  UNTIL DUP SIZE LIST-> ≤
  END DROP { Z P } PURGE SOL CLMF
»
```

**Attention** :

Si le polynome possède des racines de grande multiplicité, le procédé va osciller sans jamais converger.

Les approximations sont très bonnes pour un polynome de degré inferieur à 7 et de multiplicité maximale 4.

# DIV (ttv)

Ce programme effectue la division euclidienne du polynome A par le polynome B. Toutefois, le programme utilise une écriture des polynomes sous forme de vecteur. En effet ceci augmente la rapidité d'éxecution ainsi que la facilité de calcul et de lecture.

Les polynomes s'écrivent donc [ an an-1 ... a1 a0 ] où les ai sont les coefficients des termes de degré i, c'est à dire le terme se trouvant devant x^i. an devant être non nul, n étant le degré du polynome; et a0 se trouve être le terme constant. On a alors, A = [ an ... a0 ] et B = [ bp ... b0 ]. La division s'écrit donc A = B Q + R; avec Q = [ cq ... c0 ] et R = [ rm ... r0 ] et degré de R < degré de B soit q<p.

Il faut donc en entrée:

2: [ an ... a0 ]
1: [ bp ... b0 ]

On obtient en sortie:

2: [ cq ... c0 ]
1: [ rm ... r0 ]

**Voici le programme pour les 28C**

```
DIV
« DUP2 -> A B
  « B {1} GET
     A SIZE 1 GET
     B SIZE 1 GET
     DUP2 - -> c n p q
    « 0 q
       FOR x OVER {1} GET c / DUP 4 ROLLD * n x - 1 ->LIST
           RDM - ARRY-> 1 GET 1 - 1 ->LIST ->ARRY SWAP DROP B
       NEXT
       DROP q 2 + ROLLD q 1 + 1 ->LIST ->ARRY SWAP »
  »
»
```

**Voici le programme pour les 28S**

```
DIV
« DUP2 -> A B
  « B 1 GET
     A SIZE 1 GET
     B SIZE 1 GET
     DUP2 - -> c n p q
    « 0 q
       FOR x OVER 1 GET c / DUP 4 ROLLD * n x - 1 ->LIST
           RDM - ARRY-> 1 GET 1 - ->ARRY SWAP DROP B
       NEXT
       DROP q 2 + ROLLD q 1 + ->ARRY SWAP »
  »
»
```

# DIVC

Ce programme effectue comme DIV une division euclidienne de polynomes écrits sous forme de vecteur. La différence est que cette division s'effectue suivant les PUISSANCES CROISSANTES.

**Entrée:**

| | |
|---|---|
| 3: [a0 a1 a2 ... an] | A dividande |
| 2: [b0 b1 ... bp] | B diviseur |
| 1: d | degrée de la division (entier) |

**Sortie:**

| | |
|---|---|
| 2: [c0 c1 ... cq] | Q quotient |
| 1: [0 0 ... 0 rk ... rm] | R reste |

Avec A=BQ+R et valuation(R)=k+1>d, degré(Q)≤d, valuation(B)=0 et m=max(d+p,n).

**Exemple :** [1 2 3 4 5]=[1 2] [1 0 3]+[0 0 0 -2 5]

**Pour les 28C:**

```
DIVC
« -> A B d
  « B {1} GET
    A SIZE 1 GET
    B SIZE 1 GET
    -> c n p
    « IF d p + n >
      THEN d p + 1 ->LIST A OVER RDM B ROT RDM
      ELSE A B n 1 ->LIST RDM
      END
      -> A B
      « B A 0 d
        FOR x DUP2 x 1 + 1 ->LIST  GET c / DUP S ROLLD
              * - SWAP ARRY-> 1 GET SWAP OVER 1 +
              ROLLD 1 ->LIST ->ARRY SWAP
        NEXT
        SWAP DROP d 2 + ROLLD d 1 + 1 ->LIST ->ARRY SWAP
      »
    »
  »
»
```

**Pour les 28S :**

**DIVC**

```
« -> A B d
  « B 1 GET
    A SIZE 1 GET
    B SIZE 1 GET
    -> c n p
    « IF d p + n >
      THEN d p + 1 ->LIST A OVER RDM B ROT RDM
      ELSE A B n 1 ->LIST RDM
      END
      -> A B
      « B A 0 d
        FOR x DUP2 x 1 + GET c / DUP S ROLLD * -
              SWAP ARRY-> 1 GET SWAP OVER 1 +
              ROLLD ->ARRY SWAP
        NEXT
        SWAP DROP d 2 + ROLLD d 1 + ->ARRY SWAP
      »
    »
  »
»
```

---

## DER et DERC (ttv)

DER : dérive un polynome écrit sous forme de vecteur aux puissances décroissantes.

1: [an an-1 ... a0 ]

Renvoit 1: [n*an n-1*an-1 ... a1 ]

**DER**

```
« ARRY-> LIST-> - -> A
  « DROP
    IF A 0 == THEN [ 0 ] ELSE
     1 A FOR X X * A ROLLD
     NEXT A 1 ->LIST ->ARRY
    END »
»
```

**DERC** (dérivation selon les puissances croissantes)

```
« ARRY-> LIST-> - -> A
  « A 1 + ROLL DROP
    IF A 0 == THEN [ 0 ] ELSE
     1 A FOR X  A ROLL X *
     NEXT A 1 ->LIST ->ARRY
    END »
»
```

# VAL et VALC (ttv)

VAL : donne la valeur d'un polynome écrit sous forme de vecteur aux puissances décroissantes en un point x.

| | | | |
|---|---|---|---|
| | 2: | [ an ... a0 ] | P |
| | 1: | objet | x |
| Renvoit | 1: | objet | P(x) |

Objet peut être un real, complex, global name ou algebraic.

**VAL**

```
« -> X
  « ARRY-> LIST-> - -> A
    « 0 0 A FOR Y SWAP X Y ^ * + NEXT » »
»
```

**VALC** (donne la valeur d'un polynome sous forme de vecteurs aux puissances croissantes en un point x).

```
« -> X
  « ARRY-> 1 GET -> A
    « 0 A 1 - 0 FOR Y SWAP X Y ^ * + -1 STEP » »
»
```

---

# A->V et V->A (ttv)

Comme vous avez dû vous en rendre compte, on utilise très souvent les polynomes sous forme de vecteur. En effet seuls les coëfficients ont de l'importance. Pour rendre cohérent l'ensemble des calculs sur les polynomes il faut pouvoir passer d'un vecteur à une expression algebraic et vice et versa.

**A->V** 1: 'polynome algebraic' -> 1: [ an ... a0 ]

```
« {} 0 'I' STO
  DO 0 'X' STO OVER EVAL I FACT /
     1 ->LIST SWAP + SWAP 'X' DUP
     PURGE ∂ 1 'I' STO+ SWAP
  UNTIL OVER 0 SAME
  END
  SWAP DROP 'I' PURGE LIST-> 1 ->LIST ->ARRY
»
```

**V->A** 1: [ an ... a0 ] -> 1: 'polynome algebraic'

```
« 'X' VAL »
```

## PMAT (ttv)

Polynome de matrice. On rentre le polynome sous forme de vecteur des puissances décroissantes et la matrice carrée dont on veut calculer l'image par le polynome.

| | | | |
|---|---|---|---|
| | 2: | [ an ... a0 ] | P |
| | 1: | [[ a11 a12 ... a1n ] | |
| | | [ a21 ..... a2n ] | X |
| | | ... | |
| | | [ an1 ..... ann ]] | |
| renvoit | | | |
| | 1: | [[ b11...b1n ] | |
| | | ... | P(X) |
| | | [ bn1...bnn ]] | |

```
PMAT
« OVER SIZE 1 GET -> V X L
  « X 0 CON DUP IDN
    L 1 FOR Y DUP V Y 1 ->LIST GET
          * ROT + SWAP X *
    -1 STEP DROP
  »
»
```

---

## ->FRAC (ttv)

Voici un programme qui vous permettra d'obtenir la fraction approchée d'un réel. Si ce réel est un rationnel approché d'une manière suffisement précise vous obtiendrez la fraction réduite exacte qui lui correspond.

| | | | |
|---|---|---|---|
| en entrée : | 1: | réel | |
| en sortie : | 2: | entier | (numérateur) |
| | 1: | entier | (dénominateur) |
| ou : | 1: | 'numérateur / denominateur' selon le flag1 | |

```
« 1E11 * 1E11 -> n m
 « n ABS n
   DO SWAP OVER MOD
   UNTIL DUP 1000 / IP NOT
   END
   DROP n OVER / IP M ROT / IP
   DUP
   IF 1 ≠
   THEN 1 SF '1/1' SWAP 3 SWAP EXSUB
        SWAP 1 SWAP EXSUB
   ELSE DROP
   END » »
```

La dernière partie du programme (DUP IF ... END) met la fraction sous forme d'algébraic. Elle peut être supprimée si l'on désire obtenir le résultat sous la forme 2: Numérateur et 1: Dénominateur.

La valeur de 1000 peut être changée pour augmenter la précision. PI n'étant pas un rationnel, il est normal que cet algorithme puisse ne pas donner une fraction réduite pour un tel nombre. De plus si le réel est trop petit , la précision des chiffres significatif diminue et il se peut alors que le programme ne trouve pas la fraction. Il en va de même s'il est trop grand. Le programme fonctionne au mieux pour des valeurs comprises entre 10^3 est 10^-3.

La séquence 1 SF... ne sert que pour le programme ? -> FR.

---

# ?->FR (ttv)

->FRAC est nécessaire.

Ce programme prend n'importe quel objet dans la pile et transforme les réels qui s'y trouvent en fractions.

```
                    1: real
                       complex
                       real vector
                       complex vector
                       real matrix
                       complex matrix
                       algebraic
se transforme en    1: 'N/D'
                       'a/b+c/d*i'
                       "[ a1/b1 a2/b2 ...]"
                       "[ a1/b1+c1/d1*i ...]"
                       "[[ a1/b1 ....]
                        [..      ...]]"
                       "[[ a1/b1+c1/d1*i ...]
                        [..      ...]]"
                       'a/b*x+...+sin(c/d*x)+...'
```

**?->FR**

```
« IF DUP TYPE 0 ==
  THEN ->FRAC
  ELSE IF DUP TYPE 1 ==
       THEN 1 SF DUP RE ->FRAC SWAP IM DUP ABS ->FRAC
            '1' * SWAP IF 0 ≥ THEN + ELSE - END
       ELSE IF DUP TYPE DUP 3 == SWAP 4 == OR
            THEN ->STR "" SWAP
                 DO DUP " " POS DUP2 1 SWAP SUB
                    3 ROLLD 1 + 999 SUB SWAP RCLF 31 SF -> F
                  « IFERR STR->
                    THEN F STOF
                    ELSE F STOF 1 CF ?->FR ->STR
                         IF 1 FS? THEN DUP SIZE 1 - 2 SWAP SUB
                         END " " +
                    END »
                    ROT SWAP + SWAP
                 UNTIL DUP SIZE 2 ≤
                 END
                 + DUP 1 DISP
            ELSE IF DUP TYPE 9 ==
                 THEN -> A « A 1 1 A SIZE
                   FOR X -> Y « A X EXGET
                     IF DUP TYPE 2 <
                     THEN ?->FR DUP IF TYPE 2 < THEN 1 ELSE DUP SIZE
                          END -> S « Y SWAP EXSUB Y S + »
                     ELSE DROP Y 1 + END »
                   NEXT » DROP
                 END
            END
       END
  END
»
```

# M->DN

Ce programme a un effet semblable au ?->FR, pour une matrice, mais le calcul et le résultat sont différents. La chaîne S était inutilisable pour des calculs ultérieurs. L'idée est donc de déterminer le dénominateur commun à toutes les fractions de la matrice et de ne laisser dans la matrice que les numérateurs.

Exemple : vous avez la matrice

```
A=      [[  1  2  0 ]
         [  0 -4  6 ]
         [ -7  5  0 ]]
```

**On calcul INV(A)**

On a alors par ?->FR

```
S=      "[[ 5/19 0 -2/19 ]
         [ 7/19 0 1/19 ]
         [ 14/57 1/6 2/57]]"
```

Or on a aussi

```
INV(A)= [[ -30/-114 0/-114 12/-114 ]
         [ -42/-114 0/-114 -6/-114 ]
         [ -28/-114 -19/-114 -4/-114 ]]
```

Où -114 est le PPCM des dénominateurs (c'est en fait le déterminant).
Ce programme renvoie donc

```
2:  -114
1:  [[ -30 0 12 ]
     [ -42 0 -6 ]
     [ -28 -19 -4 ]]
```

c'est à dire

2: dénominateur commun
1: matrice des numérateurs

**M->DN**

```
« DUP DET INV SWAP OVER * ARRY-> -> a
  « a LIST-> DROP * 1 + -> b
    « 1 b START DUP SIGN SWAP OVER * .1 + IP * b ROLL
       NEXT
    »
    a ->ARRY
  »
»
```

# PARAM (ttv)

Ce programme est présenté sous deux formes:

Un seul et unique programme au début duquel se trouvent tous les paramètres nécessaires au tracé de la courbe. L'avantage est qu'il est court, mais il faut le visiter pour modifier l'un des paramètres. L'autre forme est une série de programmes de tracé et d'utilitaires que l'on utilise comme le menu PLOT.

**1.**

Ce programme s'exécute directement et utilise le paramétrage en cours. Le paramètre est T. La première expression algébrique est l'équation paramétrique sur l'axe x. La seconde est sur y. Le paramètre suivant est la borne inférieure du paramètre, le suivant est la borne supérieure. Le dernier paramètre est l'incrémentation qui correspond à la densité du tracé (plus il est petit, plus le tracé est précis, mais plus il est lent).

```
PARAM
« RCLF 31 CF 'T^3-T' '1.5*COS(T)-.3' '-x' 'x' .05
 ->NUM -> EQX EQY XMIN XMAX INC
 « CLLCD DRAX XMIN ->NUM XMAX ->NUM
   FOR X X 'T' STO
     IFERR EQX ->NUM EQY ->NUM
     THEN MAXR ->NUM DUP
     END
     R->C PIXEL
   INC STEP 'T' PURGE STOF
 »
»
```

**2.**

Entrez la série de programmes suivant. Pour tracer une courbe installez d'abord les programmes en lancant PARAM. Puis entrez les paramètres à modifier et lancez TRACE pour dessiner la courbe. Le paramètre est X.

**STEX**

```
« 'EQX' STO »
```

Stocke l'expression algébrique de la pile dans EQX qui est l'équation de x. Elle doit être impérativement une fonction de X.

**STEY**

```
« 'EQY' STO »
```

Stocke dans EQY qui est l'équation en y. Elle doit être une fonction de X.

**TMIN**

```
« 'PPAR°' RCL 1 ROT PUT 'PPAR°' STO »
```

Stocke dans PPAR° la valeur minimale du paramètre.

**TMAX**

```
« 'PPAR°' RCL 2 ROT PUT 'PPAR°' STO »
```

Idem valeur maximale.

**INC**

```
« 'PPAR°' RCL 3 ROT PUT 'PPAR°' STO »
```

Idem incrémentation.

**TRACE**

```
« RCLF 31 CF CLLCD RCEX 1 DISP RCEY 2 DISP
  PPAR° DUP 3 DISP .5 WAIT CLLCD DRAX
  LIST-> DROP ->NUM -> I
  « ->NUM SWAP ->NUM SWAP
    FOR T T 'X' STO
       IFERR EQX ->NUM EQY ->NUM
       THEN MAXR ->NUM DUP
       END
       R->C PIXEL
    I STEP STOP
  »
»
```

Trace la courbe sans s'interrompre aux endroits où les fonctions ne sont pas définies.

**RCEX**

```
« 'EQX' RCL »
```

Rappel EQX.

**RCEY**

```
« 'EQY' RCL »
```

Rappel EQY.

**EQX** (exemple à ne pas taper)

```
'X^3-X'
```

Variable équation des x.

**EQY** (exemple à ne pas taper)

```
'1.5*COS(X)-.3'
```

Variable équation des y.

**PPAR**
Paramètrage normal.

**PPAR°**
Paramètrage de courbe paramétrées.

**PARAM**

```
« 0 'X' STO RCLF 31 CF
  IFERR 'PPAR' RCL
  THEN { (-6.8,-1.5) (6.8,1.6) X 1 (0,0) }
       'PPAR' STO
  ELSE DROP END
  IFERR 'PPAR°' RCL
  THEN { '-π' 'π' .05 } 'PPAR°' STO
  ELSE DROP END
  { STEX STEY TMIN TMAX INC TRACE
  RCEX RCEY EQX EQY PPAR PPAR° PARAM X }
  ORDER STOF
»
```

Installe le menu de paramétrage.

---

# POL (ttv)

(En mode RAD)

Comme pour PARAM ce programme est présenté sous deux formes:

Un seul et unique programme au début duquel se trouvent tous les paramètres nécessaires au tracé de la courbe. L'avantage est qu'il est court, mais il faut le visiter pour modifier l'un des paramètres. L'autre forme est une série de programmes de tracé et d'utilitaires que l'on utilise un petit peu comme le menu PLOT.

**1.**

Ce programme s'exécute directement et utilise le paramétrage en cours. Le paramètre est X. La première expression algébrique est l'équation polaire R en fonction de l'angle X. Le paramètre suivant est la borne inférieure du paramètre, le suivant est la borne supérieure. Le dernier paramètre est l'incrémentation qui correspond à la densité du tracé, plus il est petit, plus le tracé est précis, mais plus il est lent.

**POL**

```
« RCLF 31 CF 'COS(X)' '-π' 'π' .05
 ->NUM -> EQP BI BS I
 « CLLCD DRAX BI ->NUM BS ->NUM
   FOR T T 'X' STO
     IFERR EQP ->NUM
     THEN MAXR ->NUM
     END
     T R->C P->R PIXEL
   I STEP
 » 'X' PURGE STOF
»
```

**2.**

Entrez la série de programmes suivants. Pour tracer une courbe installez d'abord les programmes en lançant POL. Puis entrez les paramètres à modifier et lancez DRPOL pour dessiner la courbe. Le paramètre est X.

**STPOL**

```
« 'EQPOL' STO »
```

Stocke l'expression algébrique de la pile dans EQPOL qui est l'équation R(X). Elle doit être impérativement une fonction de X.

**RCPOL**

```
« 'EQPOL' RCL »
```

**TMIN**

```
« 'PPAR°' RCL 1 ROT PUT 'PPAR°' STO »
```

Stocke dans PPAR° la valeur minimale du paramètre.

**TMAX**

```
« 'PPAR°' RCL 2 ROT PUT 'PPAR°' STO »
```

Idem valeur maximale.

**INC**

```
« 'PPAR°' RCL 3 ROT PUT 'PPAR°' STO »
```

Idem incrémentation.

**DRPOL**

```
« RCLF 31 CF CLLCD RCPOL 1 DISP
  PPAR° DUP 2 DISP .5 WAIT CLLCD DRAX
  LIST-> DROP ->NUM -> I
  « ->NUM SWAP ->NUM SWAP
    FOR T T 'X' STO
       IFERR EQPOL ->NUM
       THEN MAXR ->NUM
       END
       T R->C P->R PIXEL
    I STEP STOF
  »
»
```

Trace la courbe sans s'interrompre aux endroits où les fonctions ne sont pas définies.

**PPAR**

Paramètrage normale.

**PPAR°**

Paramètrage de courbes paramètrées.

**POL**

```
« RCLF 31 CF 0 'X' STO IFERR 'PPAR' RCL THEN
  { (-6.8,-1.5) (6.8,1.6) X 1 (0,0) }
  'PPAR' STO ELSE DROP END
  IFERR 'PPAR°' RCL THEN { '-π' 'π' .05 }
  'PPAR°' STO ELSE DROP END
  { STPOL RCPOL TMIN TMAX INC DRPOL
  EQPOL PPAR PPAR° POL X } ORDER STOF
»
```

Installe le menu de polaire.

# EQ (28C)

Tracer n'importe quelles courbes même sur des intervalles sur lesquels elles ne sont pas définies ou renvoient un complexe.

Tapez ce programme:

```
« IFERR EQ1 ->NUM DUP TYPE MAXR ROT IFTE
  THEN MAXR END
»
```

Faites STEQ
et stockez dans EQ1 la fonction que vous voulez tracer avec les programmes suivants:

**STE1**
« 'EQ1' STO »
Stocke la fonction

**RCE1**
« 'EQ1' RCL »
Rappelle la fonction

**EQ1** (exemple à ne pas taper).
'LN(√(-X^2+5))/SIN(X)'
Fonction à tracer.

---

## JINGLE (ttv)

Voici un programme enfin inutile. Son unique fonction est d'être agréable aux oreilles. Il n'est pas long, tapez le, vous verrez.

```
L
{ 390 .375 440 .15 275 .15
  350 .0375 350 .15 390 .0375 690 .15
  565 .15 390 .15 465 .15 565 .15
  590 .0375 390 .0375 390 .15 565 .3
  390 .3 350 .15 390 .15 515 .0375
  390 .0375 390 .15 465 .3 390 .3 }
```

Faites **L DUP + 'L' STO**

```
JINGLE
« L LIST-> 1 SWAP
  START MEM DROP BEEP
  2 STEP
»
```

---

## RABIP (ttv)

Génération aléatoire de Beep. Exécution directe. S'éteint en appuyant sur une touche.

```
RABIP
« DO 4400 RAND * .1 RAND * BEEP
  UNTIL KEY
  END DROP
»
```

---

# LABY

Le but du jeu est de sortir d'un labyrinthe de 16 cases sur 16. Pour se diriger on utilise les quatre programmes AV (avant), AR (arrière), GA (gauche) et DR (droite).

Pour jouer : taper les programmes, entrer un labyrinthe, stocker les coordonnées de la sortie dans la variable SO (sous forme complexe), lancer le programme INIT qui vous place à une position aléatoire et se diriger grâce aux quatre programmes précités. Si on veut que la vue se retrace: lancer le programme VUE.

Listing des différents programmes et objets:

**AL**

```
« 0 RDZ 16 RAND * IP »
```

**BL1**

BL1 est une variable qui doit contenir 127 CHR

**BL2**

BL2 est une variable qui doit contenir " "

**TS**

```
« R->C SO ≠ »
```

**SO**

SO est une variable contenant la position de la sortie, par exemple (11,16)

**TV**

```
« TVP SWAP TVP + »
```

**TVP**

```
« DUP 0 < SWAP 15 > + »
```

**ETAT**

```
« DUP2 IF TV THEN TS ELSE DUP2 8 / IP SWAP 8 / IP 2 * +
1 + LAB SWAP GET 3 ROLLD 8 MOD SWAP 8 MOD SWAP 16 SWAP ^
DUP #1 * * SWAP 2 SWAP ^ * AND #0 > END »
```

**LAB**

LAB est une variable contenant le codage du labyrinthe, sous la forme de 4 entiers binaires dans une liste, les bits à 1 représentant des murs, à 0 des passages. Voici un exemple de labyrinthe (à entrer en mode hexadécimal):

```
{ #A298AA2AEA02AE10 #FA0AAA6A09F28A #7406784576007708
#34873057515654 82 }
```

**I4**
```
« X 1 - Y »
```

**I3**
```
« X 1 + Y »
```

**I2**
```
« X Y 1 - »
```

**I1**
```
« X Y 1 + »
```

**CH**
```
« ETAT 95 * 32 + CHR »
```

**TEST**
```
« 1 'COUP' STO+ DUP2 IF TS THEN DUP2 IF ETAT THEN "MUR" 1 DISP
DROP2 ABORT END ELSE "BRAVO" 1 DISP DROP2 ABORT END 'Y' STO
'X' STO VUE »
```

**DR**
```
« I3 TEST »
```

**GA**
```
« I4 TEST »
```

**AR**
```
« I2 TEST »
```

**AV**
```
« I1 TEST »
```

**VUE**
```
« BL1 I1 CH BL1 + + I4 CH BL2 I3 CH + + BL1 I2 CH BL1 + + 3
DISP 2 DISP "        " COUP ->STR + + 1 DISP »
```

**INIT**
```
« 1 'COUP' STO 0 0 DO DROP2 AL AL DUP2 ETAT NOT UNTIL END 'Y'
STO 'X' STO VUE »
```

# MASTER

MASTER est un programme pour jouer au master-mind. Pour jouer: il faut tout d'abord entrer les programmes.

Pour le premier jeu: choisir la longueur de la combinaison à chercher (réel dans la pile) et lancer STOL (par la suite, il suffira de lancer le programme INIT). Ensuite on propose une combinaison, sous forme de liste de réels qu'on place dans la pile, et on appelle le programme MASTER...

**Listing des programmes :**

**PROG**

```
« 0 0 -> PR CP CT
  « PR 1 DISP 1 'CO' STO+ "COUP=" CO ->STR
     + 4 DISP SOL PR 1 L
     FOR X DUP X GET 3 PICK X GET
       IF ==
       THEN X -2 PUT SWAP X -1 PUT SWAP 1 CP + 'CP' STO
       END
     NEXT
     'PR' STO "PLACES=" CP ->STR + 2 DISP 1 L
     FOR X DUP X GET DUP
         IF -1 >
         THEN 1 L
            FOR Y PR Y DUP2 GET 4 PICK
              IF ==
              THEN -2 PUT 'PR' STO 1 CT + 'CT' STO 4 'Y' STO
              ELSE DROP2 END
            NEXT
         END DROP
     NEXT DROP "TROUVES=" CT ->STR + 3 DISP »
»

ERR
« DROP 1 STR-> »

STOL
« DUP IF TYPE 0 == THEN 'L' STO INIT ELSE ERR END »

INIT
« 0 'CO' STO 1 L START RAND 10 * IP NEXT L ->LIST 'SOL' STO
»
MASTER
« DUP IF TYPE 5 == DUP THEN DROP DUP SIZE L == END IF THEN
CLLCD PROG ELSE ERR END »
```

# ANAG

Ce programme prend une chaîne dans la pile et liste toutes les combinaisons de caractères possibles...Pour l'utiliser: placer la chaîne dans la pile et lancer ANAG.

**ANAG**

```
« -> A « A SIZE 'B' STO DEPTH 'C' STO 1 B FOR X A X DUP SUB
NEXT PRANAG PRDEPTH DROPN {B C} PURGE CLMF » »
```

**PRANAG**

```
« IF B 0 > THEN -1 'B' STO+ PRDEPTH DUP B - FOR X X ROLL PRANAG
X ROLLD -1 STEP 1 'B' STO+ ELSE PRDEPTH DUPN PRDEPTH 2 / 1
- 1 START + -1 STEP 1 DISP END »
```

**PRDEPTH**

```
« DEPTH C - »
```

---

# CARRE

Le but de ce jeu est de reconstituer le 'carré magique', c'est à dire le dessin suivant:

Pour ce faire on 'appuie' sur les carrés (en donnant le numéro correspondant), ce qui provoque une inversion de la couleur de ce carré, ainsi qu'une inversion de certains de ses voisins... Pour jouer: entrer les programmes; lancer le programme CARREM...

Listing des différents objets:

**CALC**

```
« "PRESS A KEY..." 1 DISP T 1
  DO DROP
     DO UNTIL KEY END
  UNTIL NUM DUP DUP 48 > SWAP 58 < *
  END 1000 .05 BEEP MESS 1 DISP
  48 - GET DUP 1 DUP ROT SIZE
  START GETI 1 - DUP 3 MOD 1 +
        WHILE DUP 3 >
        REPEAT 3 -
        END SWAP 3 / IP 1 + SWAP 2 ->LIST
        CAR SWAP DUP2 GET NOT PUT 'CAR' STO
  NEXT DROP2 »
```

**VISU**

```
« DO CAR {1 1} 1 3
     FOR X "" 1 3
          START 3 ROLLD GETI 95 * 32 + CHR 4 ROLL SWAP +
          NEXT M X GET SWAP + 142 CHR + 3 ROLLD
     NEXT DROP2 2 4
     FOR X X DISP NEXT CALC
  UNTIL CAR SOL ==
  END CLLCD "BRAVO..." 2 DISP 1 3
  START 1000 .2 BEEP NEXT »
```

**CARREM**

```
« CLLCD MESS 1 DISP 0 RDZ CAR
  DO {1 1} 1 9
     START RAND .5 > PUTI NEXT
     DROP DUP
  UNTIL  SOL ≠
  END 'CAR' STO VISU »
```

**SOL** et **CAR** doivent contenir la matrice

[[ 1 1 1 ]
[ 1 0 1 ]
[ 1 1 1 ]]

**M** contient

{" 789 ->'
" 456 ->"
" 123 ->'}

**T**

```
{ {1 2 4 5} {1 2 3} {2 3 5 6} {1 4 7} {2 4 5 6 8}
{3 6 9} {4 5 7 8} {7 8 9} {5 6 8 9} }
```

**MESS**

```
"EN CALCULS..."
```

# BIP BIP° (ttv)

Ce programme allume ou éteint le son de la même façon que ceux qui font HEX DEC RAD ... sur les 28S, c'est à dire avec un point qui apparaît à côté du nom lorsque le son est allumé et disparaît lorsqu'il est éteint.

**Un inconvénient :** Comme il marche par PURGE et STO, il revient toujours en début de menu pour les 28C.

**Un avantage :** Même si vous faites 51 SF ou CF en dehors du programme, celui-ci se rétablit automatiquement lorsque vous l'exécutez.

**Pour les 28C :**

```
BIP ou BIP°
« RCLF 31 CF IFERR 'BIP' RCL
  THEN 'BIP°'
  ELSE DROP 'BIP'
  END SWAP STOF
  DUP RCL SWAP PURGE
  IF 51 FS?
  THEN 'BIP°' STO 51 CF
  ELSE 'BIP' STO 51 SF
  END
»
```

**Pour les 28S :**

```
BIP ou BIP°
« RCLF 31 CF IFERR 'BIP' RCL
  THEN 'BIP°'
  ELSE DROP 'BIP'
  END SWAP STOF
  DUP DUP RCL VARS DUP 4 ROLL POS 1 - 1 SWAP SUB
  ROT PURGE
  IF 51 FS?
  THEN 'BIP°' 51 CF
  ELSE 'BIP' 51 SF
  END ROT OVER STO +
  ORDER
»
```

Au début vous pouvez le stocker sous n'importe lequel des deux noms, par exemple BIP.

## RENAME

Ce programme change le nom d'un programme et pour les 28S replace le programme avec le nouveau au même endroit que l'ancien.

2: 'ancien nom'
1: 'nouveau nom'

**Pour les 28C :**

```
RENAME
« OVER RCL SWAP STO
PURGE
»
```

**Pour les 28S :**

```
RENAME
« OVER RCL SWAP STO
  VARS DUP2 SWAP POS 2 SWAP
  SUB ORDER PURGE »
```

---

## UP

**Pour HP28s uniquement.**

Ce programme fait remonter d'un étage dans les directories.

```
« PATH DUP SIZE 1 - DUP 0 == + GET EVAL »
```

---

## DOPATH

28s exclusivement. Ce programme effectue un chemin (PATH). Il est utile pour revenir directement à un sous menu.

*Par exemple:* `HOME 'DIR1' CRDIR DIR1 'DIR2' CRDIR DIR2 PATH HOME`
Créera les menus DIR1 et DIR2, puis reviendra au menu HOME après avoir sauvé le chemin d'accès dans la pile grâce à PATH. DOPATH effectué sur ce chemin permettra de revenir directement dans DIR2.

```
« #1h #Ch CHK ->STR 2 OVER SIZE 1 - SUB STR-> »
```

*Remarque :* #1h #Ch CHK est optionnel. (Ne le mettre que si le programme CHK à déja été entré en mémoire).

*Explication:* #1h #Ah CHK vérifie la présence d'une liste dans la pile. La séquence ->STR 2 OVER SIZE 1 - SUB crée une chaîne contenant la liste des directories à appeler pour effectuer le chemin. Enfin STR-> exécute ces commandes.

---

# Annexe VII
# GLOSSAIRE

| | |
|---|---|
| Adresse | Entier compris entre 0 et FFFFF (hexadécimal) représentant l'endroit où se trouve une information donnée. |
| Assemblage | Action d'assembler. |
| Assembler | Traduire un programme assembleur en un programme machine. |
| Assembleur | Langage symbolique représentant le langage machine, programme réalisant la traduction assembleur —> LM. |
| Bit | Mémoire prenant la valeur 0 ou 1 et constituant les quartets. |
| Buffer | Zone mémoire destinée au stockage temporaire des touches frappées en attente (non encore traitées). |
| Champ | Partie d'un registre. |
| DCB | Décimal Codé Binaire: manière de stocker un nombre décimal en le représentant par le nombre hexadécimal identique (ex: 20, décimal, sera représenté par 20h [hexadécimal] qui vaut en fait 32 [décimal]). |
| Desassemblage | Action de désassembler. |
| Desassembler | Traduire un programme machine en un programme assembleur. |
| Desassembleur | Programme réalisant un désassemblage. |
| Drapeau | Mémoire ayant la valeur 0 ou 1 et servant d'indicateur. |
| Garbage Collector | Opération réalisée dès que la machine ne dispose plus d' assez de place mémoire. Cette opération consiste en la destruction des objets temporaires devenus inutiles. On peut déclencher le garbage collector en utilisant la commande MEM. |
| Hard | Abréviation de Hardware (matériel), plus généralement: tout ce qui concerne l'électronique (fonctionnement, modifications). |
| Hexadécimal | Base 16 (les "chiffres" sont: 0 1 2 3 4 5 6 7 8 9 A B C D E F). |

| | |
|---|---|
| Indicateur | Un des symboles pouvant s'allumer au sommet de l'afficheur du HP28 et indiquant l'état de la machine (Rad, Alpha...), plus généralement mémoire ou symbole pouvant prendre 2 états. |
| Kilo-Octet | Unité de mesure de la taille d'une mémoire. Correspond à 1024 ($2^{10}$) octets. |
| Langage machine | Suite de codes représentant une suite d'instructions élémentaires compréhensible par le microprocesseur. |
| LCD | Liquid Crystal Display, afficheur à cristaux liquides. |
| Objet | Tout élément avec lequel travaille le RPL. Par exemple: un réel. |
| Octet | Information sur 8 bits, unité de mesure de la taille d'une mémoire. |
| Peek | Instruction ou programme servant à lire le contenu d'une ou plusieurs cases mémoires situées à partir d'une adresse donnée. |
| Poke | Instruction ou programme servant à écrire une ou plusieurs cases mémoires situées à partir d'une adresse donnée. |
| Processeur | Voir microprocesseur. |
| Prologue | Groupe de 5 quartets identifiant l'objet débutant par ces quartets. |
| Quartet | Case mémoire élémentaire de 4 bits (valeur comprise entre 0 et 16 [décimal], entre 0 et F [hexadécimal]). |
| Ram | Random Acces Memory ou mémoire vive: circuit électronique pouvant stocker des informations (modifiables), zone mémoire contenant des informations modifiables. |
| Registre | Une mémoire du microprocesseur. Ne contient que des entiers positifs ou nuls. |
| Rom | Read Only Memory ou mémoire morte: circuit électronique stockant des informations (non modifiables), zone mémoire contenant des informations non modifiables (la Rom contient les programmes machines des instructions RPL). |

# Annexe VIII
# INDEX

Achevé d'imprimer
sur les presses de l'imprimerie IBP
à Rungis (Val-de-Marce 94) (1) 46.86.73.54
Dépôt légal - Janvier 1989
N° d'impression: 5078