Au fond de la HP-41

Jean-Daniel Dodin - jdanield@dodin.net
www.dodin.net

Version du 16 Juin 2001 - Reproduction autorisée

AU FOND

DE LA

par

Jean-Daniel DODIN

Fig. 1 – Couverture originale

Depuis la mise en ligne de mon livre, je me dis qu'une édition plus propre est nécessaire. Je vais essayer de la faire maintenant. Je compte sur vous pour me signaler tout ce qui ne va pas... Il ne s'agit plus, comme précédemment, de faire une édition "fac simile" du livre initial, mais bien une nouvelle édition mise à jour si nécessaire.

Je ne pratique plus guère, les mises à jour ne seront donc pas de mon fait, mais j'attends vos remarques avec impatience... même si elles sont en anglais.

La société éditrice initiale était :

Editions du Cagire SARL au capital de 20000F RC Toulouse B 327 662 609 ISBN 2-86811 77 rue du Cagire 31100 TOULOUSE FRANCE Quatrième édition revue et corrigée Premier trimestre 1984 Bien sur, cette adresse ne correspond plus à rien (à rien qui ait un rapport avec la HP-41) et les Editions du Cagire n'existent plus. Je les avais créées moi-même pour éditer mes livres et je les ai dissoutes dès que je n'ai plus rien eu à éditer.

Chapitre 1

Introduction

1.1 Édition 2001

Cette nouvelle édition n'est pas très différente des précédentes, en effet je ne pratique plus la HP-41 (pas au niveau ou je le faisais dans le temps), mais j'ai abandonné l'idée de faire un "fac similé" de l'édition originale.

Du coup la mise en page n'est plus la même, les numéros ne sont plus les mêmes, mais vous bénéficiez de la correction des erreurs (j'en ai corrigées de nombreuses par rapport à l'édition web précédente) et d'une version PDF qui fait maintenant référence.

Vous pouvez m'aider de plusieurs façons.

— D'une part en me signalant les erreurs (il y en a certainement plein). Si il s'agit de fautes de frappe, envoyez-moi juste le numéro de la page, le numéro de la ligne et l'erreur à corriger.
S'il s'agit d'une erreur de fond (déjà présente dans l'original), dites-moi à quel endroit (paragraphe) il faut l'indiquer - je donnerait votre nom comme auteur de la correction (sauf opposition de votre part).
— Vous pouvez aussi refaire une partie des figures. Les figures que je donne là sont scannée sur le livre, elles avaient été faites "à la main", c'est à dire au stylo et à l'encre. Les refaire en DAO ne serait pas du luxe, mais je n'ai pas le temps.
— Vous pouvez *compléter* ce livre, si vous avez une expérience de la chose.
— Vous pouvez me fournir des photos, même en couleur.

L'original de cette édition est fait avec LyX, système d'exploitation Linux. Je peux exploiter tous les formats graphiques.

Normalement vous devez avoir sous les yeux une sortie Acrobat, mais j'espère pouvoir mettre en ligne une version HTML améliorée, si tout veut bien apparaître...

1.2 Remerciements

Ce livre n'aurait pas pu voir le jour sans l'existence du club PPC (Personnal Programming Center) qui rassemblait certainement les meilleurs spécialistes des calculatrices de poche HP. La somme de connaissances que le club a pu accumuler sur la HP-41C est proprement ahurissante. Sur les 6000 membres du club, nombreux sont ceux qui ont apporté leur pierre à la construction. Les principaux ouvriers eux-mêmes sont trop nombreux pour qu'il soit possible de les citer tous. Il me faudra me limiter aux architectes, que les autres m'excusent.

D'abord Richard J. Nelson (PPC n°l) sans qui rien de tout cela n'aurait pu exister. Fondateur de PPC et éditeur de PPC Journal, il a droit à toute ma reconnaissance. Ensuite Bill Wickes dont le livre sur la programmation synthétique m'a ouvert les yeux sur un monde nouveau. Et encore Paul Lind, génial découvreur des instructions microcode et John McGechie, coordinateur du Chapitre Australien de PPC qui a le premier assuré la diffusion des connaissances sur le microcode.

Et tant d'autres encore...John Dearing, Cary Reinstein, Robert Groom, Keith Jarett, Lynn vilkins.., merci à tous.

1.3 Avertissement

Ce livre est un peu particulier. Il comprend peu de programmes et la plupart d'entre vous n'auront jamais l'occasion d'utiliser certains chapitres. Mais après tout je n'aurai jamais l'occasion de faire l'escalade de l'Everest et j'ai pourtant pris grand plaisir à en lire le récit. J'espère que le présent ouvrage vous en apportera autant.

1.4 De quoi allons-nous parler ?

Le chapitre 2 page 5, "Géographie", vous donnera une description de la structure de votre HP-41C, des diverses zones qui se partagent la mémoire : autant de tiroirs qui peuvent contenir des trésors.

Le chapitre 3 page 16, "Signification des chiffres", vous montrera sur quelles bases travaille la 41C et vous permettra une meilleure compréhension de la structure des programmes.

Le chapitre 4 page 28, "une zone particulière", analyse la mémoire essentielle de la 41C, celle qui définit tout le reste, celle qui est en principe son domaine réservé mais qui maintenant est ouverte pour vous.

Le chapitre 5 page 33, "Au voleur !", fera pour vous le point sur une méthode particulière de construction des instructions artificielles qui vous permettra de résoudre simplement la plupart des problèmes de programmation synthétique et vous donnera quelques exemples d'application.

Le chapitre 6 page 43, "Microcode", dévoilera pour vous le saint des saints, le microcode. Vous verrez les instructions utilisables et leur mode de fonctionnement.

Le chapitre 7 page 58, "Utilisation", enfin, vous donnera des exemples de programmation en Microcode, pris dans votre machine ou réalisés par l'auteur.

Finalement des annexes 8 page 75 essaieront de répondre aux principales questions restantes.

1.5 Attention !

Les manipulations sur le logiciel, qu'il soit normal, synthétique ou en microcode sont sans danger pour la HP-41C, le pire qui puisse vous arriver est une indisponibilité d'une nuit de votre chère machine (cf. paragraphe 5.5 page :42).

Il n'en est pas de même des manipulations matérielles qui peuvent amener la destruction de l'unité centrale avec un coût de réparation de l'ordre de 650 F (1983 - cette réparation n'est plus possible en 2001), c'est pourquoi ce type de manipulation n'est pas décrit ici.

Le contenu de ce livre n'est d'aucune façon garanti par Hewlett-Packard. La programmation synthétique et le microcode ne sont d'aucune façon pris en charge par HP. Vous êtes adulte, non ? (en américain NOMAS : Not Manufacturer Supported : sans soutien du fabriquant).

L'auteur fournit ces renseignements de bonne foi mais n'accepte aucune responsabilité d'aucune sorte quant à l'usage qui peut en être fait.

1.6 Excuses

Depuis la première édition, l'auteur a perfectionné son matériel, ce qui explique la meilleure présentation de l'édition actuelle. Il reste sûrement encore des erreurs, excusez-le ! L'édition 2001 est très différente dans sa présentation.

Ceci dit, la chaîne de fabrication est un peu complexe, ce qui peut expliquer d'éventuels problèmes. Pour votre information : le texte est écrit sous Linux avec LyX. LyX est une sorte de traitement de texte qui sert d'interface graphique au formateur de texte très connu LaTeX. Donc, après avoir été écrit avec LyX, le texte est traduit en LaTeX.

Ensuite le fichier dvi résultant du travail de LaTeX est traduit en PostScript. Comme je ne suis pas sur que vous puissiez lire le PostScript, je le traduits ensuite en PDF, lisible avec Acrobat Reader.

Tout cela est fait automatiquement par LyX.

De plus, j'ai l'intention de faire une version HTML pour la lecture en ligne, mais là je ne sais pas trop comment certains caractères vont passer. Si vous lisez cette version, vous aurez peut-être des problèmes. Bon courage !

1.7 A suivre...

Ce livre ne vous apprendra pas tout, et loin de là, de la programmation de la 41c, qu'elle soit normale ou synthétique, ni même en microcode. ce n'est d'ailleurs pas son but. Il veut être un outil de travail de base à partir duquel chacun est libre d'élaborer ses propres applications. Le meilleur moyen de progresser est d'échanger ses connaissances avec d'autres. C'est Le but du club PPC et, particulièrement en ce qui nous concerne, de sa section française.

Certes, beaucoup de choses ont déjà été trouvées, mais des isolés les redécouvrent tous les jours et les sources sont souvent étrangères ou inaccessibles. Si vous avez apprécié ce Livre le club PPC est fait pour vous.

Je ne sais pas s'il existe encore des sections actives du club PPC, mais je crois que oui. Si vous les connaissez, envoyez-moi les coordonnées, que je les signale.

Pour l'instant il y a deux sources de renseignements, la liste de discussion PPCT (voir sur mon site internet www.dodin.net) et le HP Museum. On trouve du matériel souvent aux enchères sur e-bay.

Chapitre 2

Géographie

Notre chère machine a beaucoup de possibilités, mais son organisation interne est passablement compliquée et il n'est pas sans intérêt de s'y attarder. Nous avons connu la HP-41C, la HP-41CV (C 5), maintenant la HP-41CX, toutes voisines, chaque modèle incorporant simplement "dans la boite" les modules qui étaient à rajouter auparavant. Tout ceci n'a rien simplifié.

2.1 Géographie matérielle

La HP-41C comporte essentiellement 6 parties indépendantes :

1. L'affichage : comprend les afficheurs à cristaux liquides et leurs circuits de commande, y compris une horloge.
2. Le clavier : fait d'un circuit imprimé spécial, il assure non seulement sa fonction évidente mais encore la liaison entre l'afficheur, le circuit logique et les connecteurs.
3. Le circuit Logique ou carte mère (cf figure 2.4 page 7) : cerveau et mémoire de la 41C, il comprend un microprocesseur spécialisé, les mémoires mortes internes et les mémoires vives standard, ainsi qu'une horloge et les circuits annexes. C'est à ce niveau que les nouveaux modèles reçoivent les adjonctions. Voir aussi la figure 2.3 page suivante).
4. Les connecteurs d'interface (cf figure 2.1) : en effet, les circuits de connexion apparents dans les ouvertures au sommet de ta 41C sont mécaniquement indépendants et donc facilement interchangeables. Ils sont en fait composés d'un circuit imprimé souple replié de multiples fois sur lui même. ce circuit n'est pas soudé mais simplement posé sur le circuit du clavier ; il assure également la liaison avec l'alimentation.
5. L'alimentation (figure 2.2 page 6) : constituée soit de piles soit d'accumulateurs au format des piles, chargés en dehors de la machine, soit d'un bloc d'accumulateurs HP.
6. Le boîtier : en 3 parties de plastique résistant assemblées par des vis cachées sous les pieds en caoutchouc. Les vis sont vissées directement dans le plastique et il ne faut donc pas trop les forcer.

2.2 Géographie électrique

Elle est surtout notable en ce qui concerne les connecteurs des ports. Remarquer en particulier (fig. 2.7 et 2.8, pages 8 et 9) L'usage des plots 2 et 4. Ceux-ci ne sont pas reliés à l'intérieur de la HP mais uniquement, éventuellement, au plot 3. C'est ainsi que la HP reconnaît la position des modules et numérote les ports 1, 2, 3 et 4 (cf. la gravure au dos de la 41C).

Remarquons également (fig. 2.8, p. 9) la disposition de l'alimentation qui protège les piles contre une "charge" et permet la connexion de batteries extérieures ou d'une alimentation 6V continus branchée sur le secteur (attention, il n'est pas impossible que sur les nouveaux modèles cette disposition soit modifiée).

Fig. 2.1 – Les ports de la HP-41

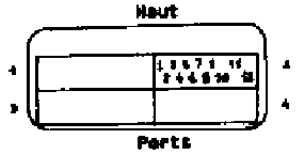


FIG. 2.2 – *La HP-41 vue de dessous*

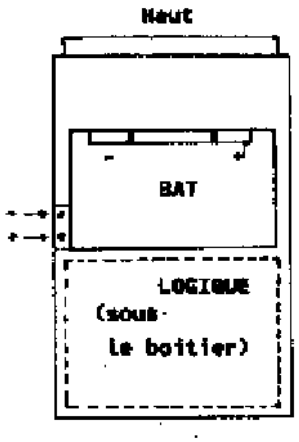


FIG. 2.3 – *La HP-41 vue de dessous, capot enlevé*

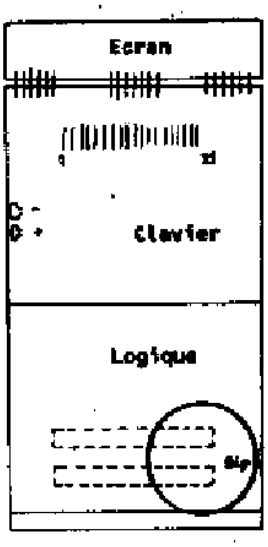


FIG. 2.4 – Carte mère de la HP-41

C = condensateur
R = résistance
CR = diode
L = bobine
U = circuit intégré
réf : PPCJ V6N6 p13

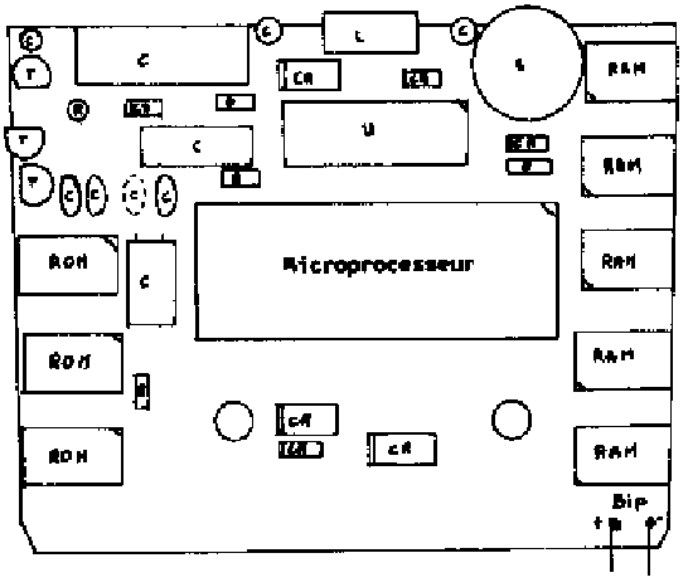


FIG. 2.5 – Circuit logique d'une HP-41C, Octobre 1982

Le circuit logique reproduit figure 7 a été dessiné en Octobre 1979 et celui ci-dessus en Octobre 1982. En trois ans il a perdu bon nombre de ses composants. Les transistors ont disparu. Est-ce là la conséquence du nouveau mode RESET (cf. 5.5 page 42), ou d'une intégration plus poussée ? Le fait est qu'il y a de la place pour de futurs composants.

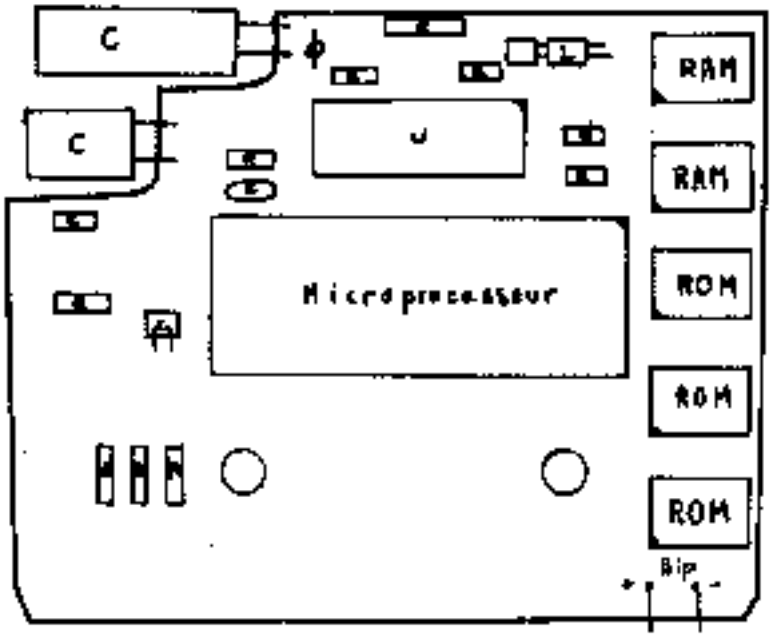


FIG. 2.6 – Piggy back

Compte tenu de la nécessité de rajouter des composants, les dernières plaquettes HP utilisent la méthode dite "piggy back" (se traduirait en français par "à la levrette"), consistant à placer deux circuits intégrés l'un sur l'autre.

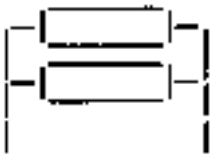
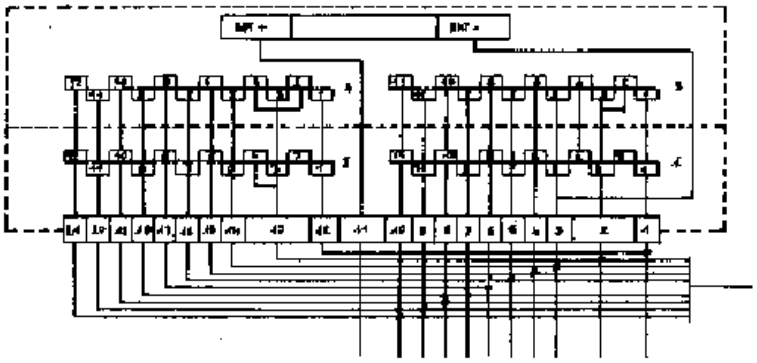


FIG. 2.7 – Vue de dessus des connecteurs (réf. PPCJ V6N6 p. 14)



Les premières 41C étaient prévues pour recevoir une telle alimentation par l'ouverture latérale utilisée par le chargeur de batteries. Si on enlevait le petit cache plastique on apercevait deux plots en laiton jaune qui dépassaient du plastique (le - côté écran).

Hélas, HP a abandonné l'idée de fabriquer un adaptateur 6V quand il a sorti son "battery pack" et a un jour supprimé ces plots ainsi que leurs guides en plastique et les petits ressorts de contact. Mais pendant longtemps le circuit du clavier n'a pas été modifié et comportait encore les contacts, il suffisait donc aux bricoleurs de reconstruire les plots en laiton pour pouvoir aisément brancher une alimentation extérieure. Ceci est-il encore possible sur les derniers modèles, et pour combien de temps ?

2.3 Géographie de la mémoire vive

2.3.1 Généralités

Il ne s'agit plus de matériel mais de logiciel, mais c'est toujours de la géographie !

La mémoire vive de la HP 41C est entièrement accessible à l'utilisateur avisé.

S'agissant de géographie logique, la mesure ne peut pas se faire en millimètres. l'unité de mémoire vive de la HP-41C est le registre.

Ce registre a comme sous multiple l'octet, le digit et le bit (cf. définitions, section 8.1 page 75). Sauf mention explicite, nous compterons toujours en base hexadécimale.

Le premier registre de la mémoire vive est R(000). Cette numérotation n'a rien à voir avec cette que vous utilisez avec RCL et STO le chiffre hexadécimal ( ) sera appelé "adresse absolue" du registre.

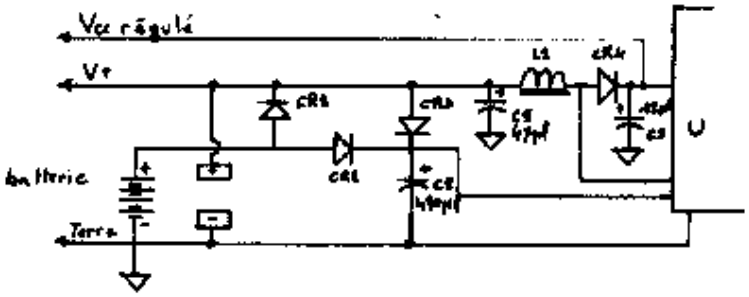
Le dernier registre possible (en 1983...) est R(3FF). Comme 3FF=1023 décimal, il y a donc de 0 à 1023=1024 registres possibles, soit exactement l k registres et donc à 7 octets par registre 7 k octets. Hélas, il s'en faut de beaucoup que ces registres existent tous.

La "carte" de la mémoire vive est représentée à la fin de ce livre, fig. 8.2 page 82.

Les zones suivantes peuvent éventuellement être rencontrées en mémoire vive :

1. les registres d'état
2. Un vide
3. les X-Mémoires
4. les assignements

FIG. 2.8 – Alimentation de la 41C (circulaire de PPCJ V6N7)



5. les alarmes
6. les buffers
7. les programmes
8. les données

2.3.2 Les registres d'état

De R(000) à R(00F), c'est l'aide mémoire de la HP-41C. Normalement vous ne devriez pas y avoir accès, mais nous ne sommes pas des gens normaux ! Ces registres sont si importants que nous leur consacreront un chapitre entier.

2.3.3 Un vide

Certains emplacements ont une adresse mais pas de contenu ! sur une HP-41C standard, c'est le cas de R(010) à R(OBF) et de R(200) à R(3FF). Ces vides sont occupés en partie par la mémoire vive du module d'extension de fonctions de R(040) à R(OBF) et les X-mémoires au dessus de R(1FF) (en standard sur la 41CX), ce qui laisse encore des vides. Ces vides ne sont pas tous inutiles, en particulier les registres R(010) à R(01F) doivent rester vides pour que la 41C puisse fonctionner (voir chapitre 6 page 43).

2.3.4 Les X-mémoires

Ce sont des registres ordinaires mais utilisés avec les fonctions spéciales du module X-fonctions. Elles sont placées de R(040) à R(OBF) pour les mémoires voisinant avec les X-Fonctions, de R(200) à R(2EF) et de R(301) à R(3EF) pour les modules possibles X-mémoires.

Moyennant quelques limitations, on peut étendre leur usage et même faire fonctionner des programmes à l'intérieur de ces registres. Ce n'est cependant pas leur usage principal. Leur étude complète sort du cadre de cet ouvrage. Il ne faut pas les confondre avec les autres registres de mémoire vive.

2.3.5 Les assignements

Ce sont des registres qui contiennent les références des touches assignées. Ces registres sont situés à partir de R(OCO) et en comptant vers les adresses croissantes selon le nombre de touches assignées. Ils ont le format suivant :

:F :O : : : : : : : : : : : : :

F0 est suivi de : code de fonction sur 4 digits + code touche sur 2 digits + code fonction + code touche, au total 14 digits. Quand on efface un assignement, les 3 octets ne sont pas immédiatement supprimés, seul l'octet donnant le code de la touche est remis à zéro (et l'index de touche baissé dans les registres ⊢ ou e). Cet emplacement est réutilisé par une nouvelle affectation, mais il n'est pas supprimé par un PACK. Par contre, il est supprimé par PK du PPC ROM.

Le code de la fonction est celui décrit dans le chapitre 3 page 16. Si la fonction n'utilise qu'un octet, les 2 digits de gauche du code fonction sont 04. Il y a donc deux assignements par registre. Ces registres sont remplis de droite à gauche et de bas en haut.

Par exemple, assigner la fonction LN à la touche A donne à l'affichage (en admettant qu'aucun autre assignement n'existe) :

FIG. 2.9 – Rôle des lignes de sortie

(réf. nos de la Fig. 2.7, page 8)

Ligne 1 Vbat (ou Vcc). Ligne isolée de la batterie par une diode (interdisant le retour vers la batterie) utilisée par les accessoires nécessitant un courant élevé et une tension non régulée (lecteur de cartes ou crayon lumineux).

Ligne 2 Sélectionne l'adresse (ports 3 et 4) par connexion à 3

Ligne 3 Vcc 6V régulé pour usage logique. Ne supporte pas plus de 20mA sans griller (nécessitant le remplacement du circuit logique). Tombe à la valeur de la batterie quand la HP est éteinte.

Ligne 4 L'autre sélection d'adresse (ports 2 et 4)

Ligne 5 Terre ou V-

Ligne 6 PWO (power on). 0V quand le microprocesseur est arrêté, Vcc quand il est actif. L'interruption de cette ligne permet de rendre "absent" un périphérique sans le débrancher physiquement

Ligne 7 Ligne de transfert de données et d'adresses de registres, en série avec le registre C du CPU. A 0 quand le CPU est arrêté, transmet des bits série quand le CPU est actif.

Ligne 8 Hand-shake ("poignée de main") avec périphériques asynchrones. A Vcc sauf impulsions à 0V.

Ligne 9 ISA. Transporte les adresses de ROM et renvoie 10 bits d'instruction prise à l'adresse indiquée. A 0 quand inactif.

Ligne l0 SYNC. Synchronisation entre les périphériques et la 41C. A 0 en sommeil profond, à Vcc en sommeil léger et modulée en fonctionnement.

Ligne 11 F2. Phase 2. Permet de compter les 56 pas d'un cycle machine.

Ligne 12 F1. Phase 1. Comme F2 mais déphasé d'une micro-seconde environ pour la lecture des données.

Ces données permettent aux spécialistes la construction d'interfaces. Ce genre de travail n'est pas à conseiller aux simples bricoleurs. Le risque de destruction du circuit logique est élevé en cas de fausse manoeuvre (extrait de PPCJ V7 N3 Avril 1980).

:F :0 :0 :0 :0 :0 :0 :0 :0 :1 :5 :0 :0 :1 :

Si nous assignons ensuite une autre fonction, la 41c va d'abord regarder si la partie gauche de R(0CO) est libre. Si c'est le cas, elle utilise la place ainsi disponible. Sinon la HP "pousse" vers le haut les registres d'assignement et place donc le contenu de R(OCO) en R(OC1). R(OCO) devient alors disponible...

La fig. 2.10, page 11, donne le code utilisé dans les registres d'assignement pour désigner les touches. Ce code est différent de celui qui apparaît à l'écran.

2.3.6 Les alarmes

Pour ceux qui possèdent le module horloge (ou une 41CX), sachez que la 41C utilise les registres situés au dessus des registres d'assignement pour enregistrer les alarmes et leurs messages. les registres ainsi utilisés sont encadrés par un registre d'état supérieur et un registre d'état inférieur (voir section 8.9, page 80).

2.3.7 Le buffer

Un autre type de mémoire est utilisé par HP, en particulier dans un module de diffusion réduite appelé HP-IL development. Un buffer est une mémoire tampon destinée à recevoir des données à titre transitoire, par exemple au cours d'un transfert sur la boucle HP-IL entre la mémoire de masse et l'imprimante. Ce buffer est normalement situé au dessus des registres d'assignement et au dessus (ou au dessous, par ordre chronologique) des alarmes.

Il est constitué d'un registre d'état inférieur, d'un registre d'état supérieur et entre les deux de la mémoire tampon proprement dite. Le registre d'état inférieur est de la forme :

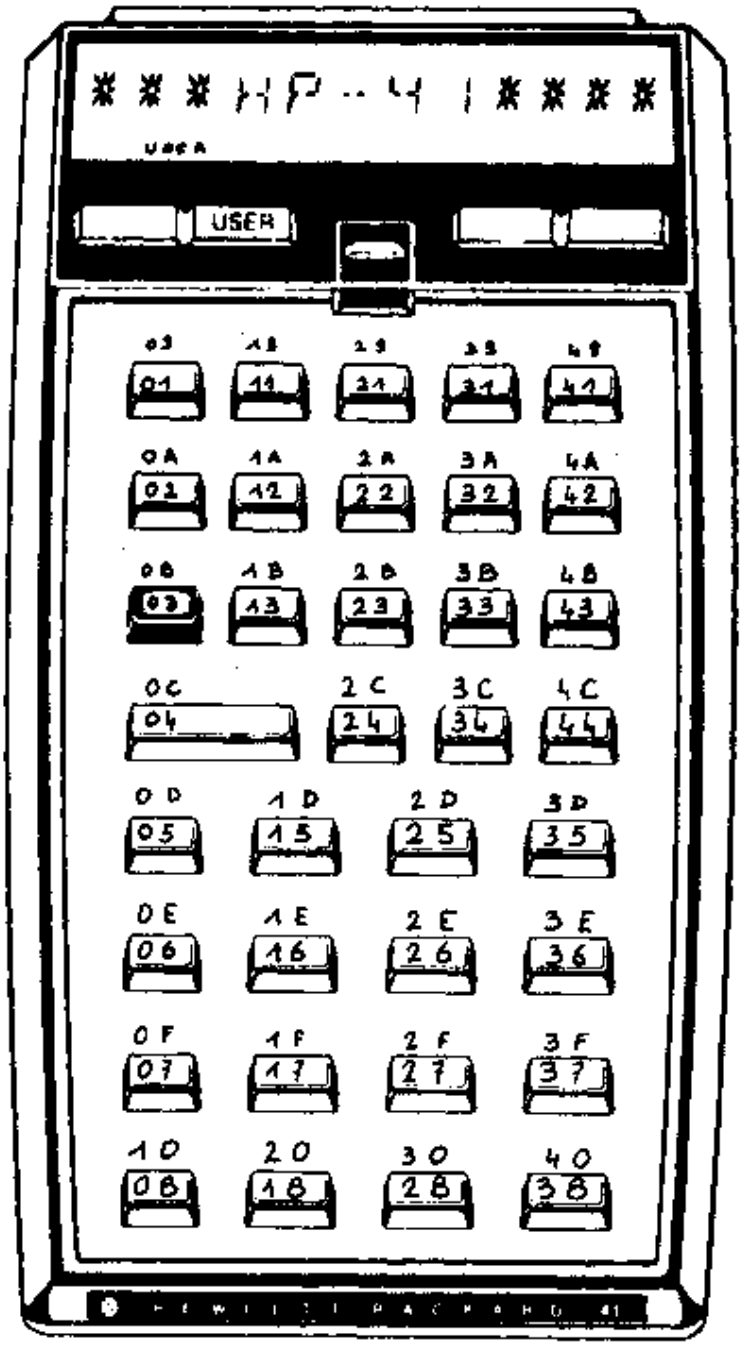
:B :B :F :5 :6 :4 :0 : : :...

BB est (toujours en hexadécimal, n'oubliez pas) le repère du registre, comme F0 pour les assignements et AA pour les alarmes, F5 est le nombre total de registres, état compris, ici 245 registres. 640 est la position du pointeur dans le buffer. Ici le pointeur désigne l'octet 640 (décimal 1600). Quand au registre d'état supérieur, il a pour valeur :

:1 :0 :4 :D :4 :F :4 :E :4 :9 :5 :4 :5 :2 :

FIG. 2.10 – Codes (hexadécimaux) des touches trouvés dans les registres d'assignement.

Au dessus de La touche, Le code de la touche shiftée. Remarquer que si on sait le faire on peut assigner une fonction à la touche SHIFT (03) et même à la touche SHIFT shiftée (!) (0B). Pour utiliser cette dernière il faut alors sortir du mode USER, presser SHIFT, USER, et une deuxième fois SHIFT. Le code "affichage" (utilisé par KA) est celui attendu (31 et -31). Essayez quand vous aurez vu KA (chap 5 page 33chapitre).



C'est une astuce de HP. Si vous commencez l'étude de la 41C avec ce livre vous n'avez pas encore les connaissances nécessaires pour la comprendre. Ces connaissances vous seront données au chapitre suivant. Dans le cas contraire vous avez peut-être déjà compris ? Solution 17.

2.3.8 Des Programmes

Les programmes sont situés dans une zone au dessus des trois types de mémoires vus ci-dessus. Cette zone est située entre deux limites définies par des adresses conservées dans les registres d'état de la 41C (cf. chapitre 4).

La limite inférieure est l'adresse du registre contenant le .END. permanent. Le .END. est la fin du dernier programme en mémoire, dans l'ordre du CAT 1. Il est situé quelque part au dessus des assignements-alarmes-buffer. Entre le .END. et ces registres spécialisés, il y a donc des registres disponibles qui sont attribués en fonction des besoins. Cette attribution est automatique. En cas d'insuffisance de registre, ce fait est en général signalé par un message "PACKING" parfois suivi de "TRY AGAIN" ("essayez encore").

La limite supérieure de la zone programme est en fait la limite inférieure de la zone données...

2.3.9 Les Données

Les données sont situées entre une limite inférieure et le sommet de la mémoire standard, variant avec le nombre de modules de mémoire vive avec un maximum de 1FF. Il n'y a donc pas de limite supérieure enregistrée. Ceci est dû à la possibilité qu'il y a sur une 41C d'enlever des modules de mémoire sans que la machine puisse le savoir.

La limite inférieure est l'adresse absolue de R00 (remarquez : pas de ( ) autour des chiffres) le registre auquel vous accédez par STO 00 ou RCL 00.

Ouf ! notre machine n'est vraiment pas simple. Accrochez-vous, ce n'est pas fini !

2.4 Géographie de l'unité centrale

Comment faire pour s'y reconnaître, dans toutes ces zones de mémoire ? Pourtant la 41C s'y retrouve. Elle sait aussi analyser les fonctions décrites au chapitre 3. Mais qu'est-ce que c'est que ce "elle" ?

C'est un circuit électronique dont le principe est maintenant bien connu et que l'on appelle microprocesseur.

Ce microprocesseur est à lui seul une petite calculatrice avec mémoire vive et mémoire morte et instructions particulières.

Les mémoires mortes du microprocesseur sont photo-gravées et il n'est pas en notre pouvoir de les lire.

Nous nous contenterons donc d'observer les effets des instructions, de leur donner un nom et de les utiliser au mieux.

Nous verrons au chapitre 6 comment est organisée la mémoire vive de ce microprocesseur. La figure 2.12, page 14 vous donne une idée du schéma standard d'un microprocesseur, le nôtre doit y ressembler beaucoup.

Notre unité centrale considère comme des périphériques aussi bien l'imprimante et le lecteur de cartes que l'affichage ou la mémoire vive que nous venons d'étudier longuement (Section 2.3, page 8). Elle prend ses instructions dans la mémoire morte qui va être décrite ci-après.

2.5 Géographie de la mémoire morte

Si la mémoire vive a comme unité le registre, la mémoire morte a comme unité le mot.. Par une bizarrerie du sort, sans doute explicable par des raisons historiques, le mot d'instruction du microprocesseur HP de la 41C a une longueur de 10 bits, donc un octet plus deux bits. En hexadécimal, un octet est représenté par deux digits (par exemple 3E). Pour représenter les deux bits supplémentaires il faut donc mobiliser un digit de plus.

Tout le problème est de savoir si ces deux bits supplémentaires sont pris à gauche ou à droite (pas de politique, je vous prie) ; expliquons nous sur un exemple. L'instruction RTN, pour le microprocesseur doit être codée :

1 1 1 1 1 0 0 0 0 0

soit 10 bits. Pour les représenter en hexadécimal nous pouvons regrouper 2 bits à gauche, puis 4 bits, puis les derniers 4 bits, obtenant le mode "244" ; dans ce mode, RTN s'écrit :

FIG. 2.11 – *carte de la mémoire morte*

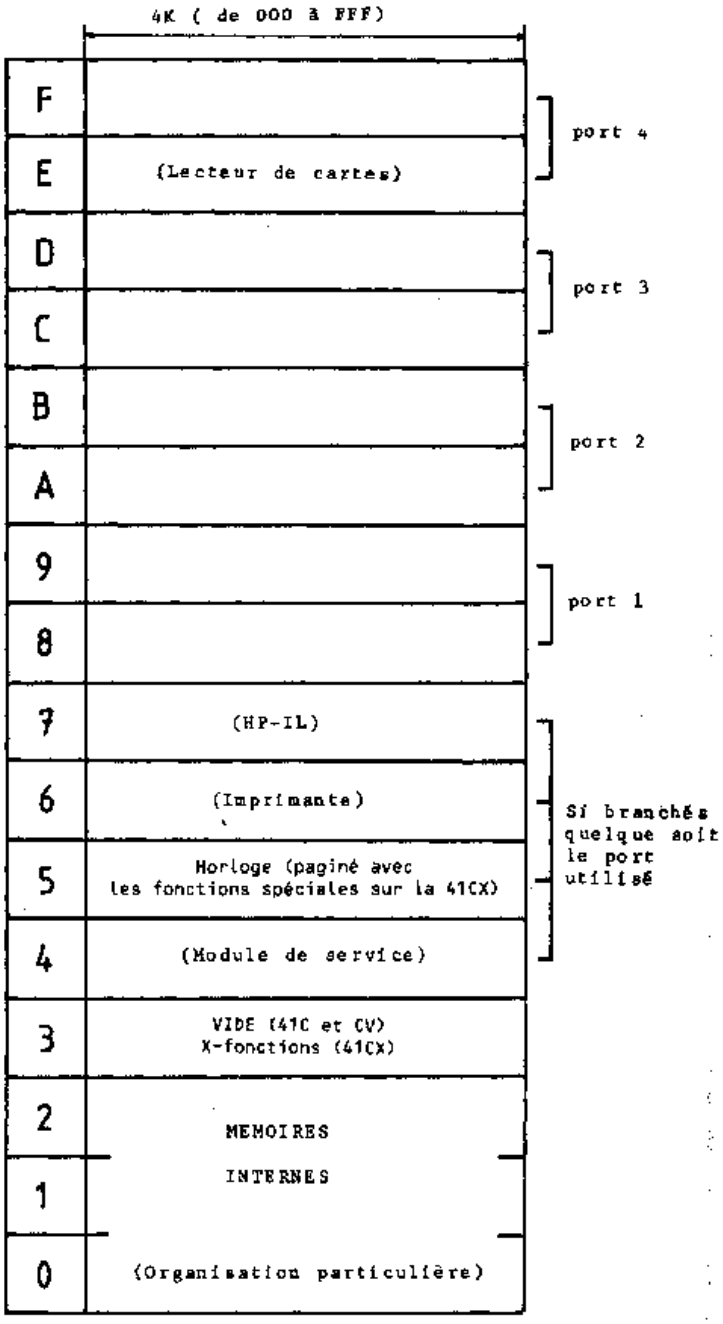
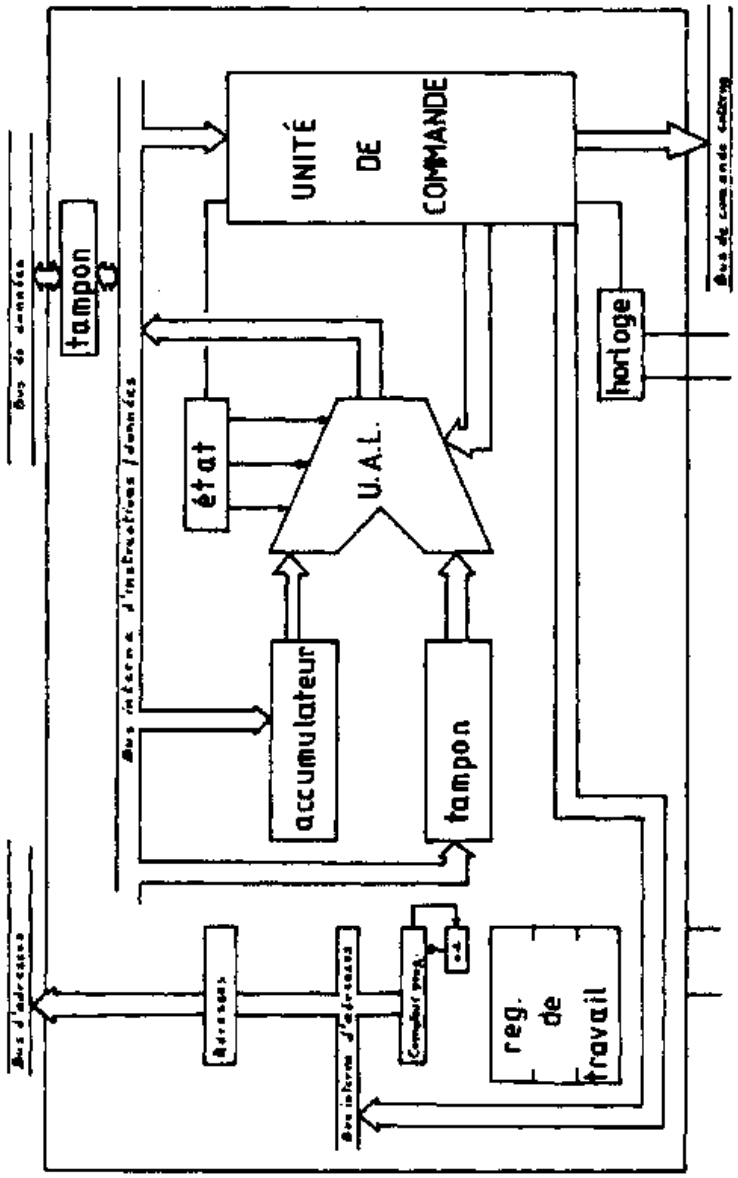


FIG. 2.12 – *Architecture générale d'un microprocesseur*



11 1110 0000

3 E 0 =3E0

Nous pouvons aussi regrouper 4 bits à gauche, les 4 suivants et laisser les deux de droite tout seuls. Nous avons le mode 442 :

1111 1000 00

F 8 0 =F80

Ces mots sont comptés un par un, il n'y a pas de structure semblable au registre.

En résumé l'unité de mémoire morte est le mot, la mémoire morte est organisée en colonnes de 16 mots, en pages de 16 colonnes et 256 mots et en modules de 16 pages ou 4096 mots (4K). le microprocesseur est capable de reconnaître 16 modules différents existant simultanément en mémoire, numérotés de 0 à F, soient 65536 mots (64K).

Les modules 0, 1 et 2 sont réservés aux fonctions construites d'origine dans la 41C, c'est la *mémoire interne*. C'est le contenu de ces modules qui fait de la 41C ce qu'elle est.

Le module 3 est utilisé uniquement sur la HP-41CX, on y trouve les X-fonctions.

Le module 4 est celui du service. HP utilise dans ses ateliers un "module de service" chargé de contrôler le bon fonctionnement de la 41C et qui est placé à cette adresse. En fait, la 41C teste la présence de ce module à chaque mise en service, auquel cas le module se substitue entièrement aux mémoires internes.

Cette possibilité pourrait être utilisée par un programmeur expert pour modifier complètement le fonctionnement de la 41C. Cette adresse est utilisée par le module HP-IL. L'adresse de la partie imprimante de ce module est en binaire 0110 (6). Quand on utilise une imprimante non IL, il faut "DISABLE" (déconnecter) ce module IL à l'aide d'un curseur situé sous le boîtier du module. Ce curseur se contente de mettre à la masse (à 0) une des lignes d'adresse, transformant 0110 (6) en 0100 (4) et t'adresse passe de 6 à 4. Cela marche car les premiers mots de ce module, lus comme instructions aboutissent finalement à un retour. 0n a en effet :

```
4000 01D                Carry n'est pas levé, donc
4001 01B ?C GO 0607     ne saute pas
4002 007 JC +00         ne saute pas
4003 IBB JNC +37        saute et arrive sur
....
403A 3E0 RTN
```

Que vous vous expliquerez mieux après la lecture du chapitre 58.

Le module 5 est celui de l'horloge. la présence de ce module est testée parfois par l'HP-IL, l'utilisation de cette adresse est donc à éviter si l'HP-IL est présente en mémoire à moins d'y mettre justement l'horloge. Le module horloge est présent d'origine sur la CX. Sur la CX cette même adresse est utilisée pour les fonctions supplémentaires, par commutation dans le temps (on a soit un module soit l'autre), sans que cela soit apparent pour l'utilisateur.

Le module 6 est celui de l'imprimante (tous modèles), les adresses C, D, E, F de ce module sont souvent testées par la mémoire interne. A condition d'en tenir compte on peut utiliser cette adresse. En particulier une éventuelle extension vidéo pourrait prendre en compte ces appels à l'imprimante. Ce n'est semble-t-il pas le cas actuellement. Le module 7 est celui de l'HP-IL, mémoire de masse et contrôle. En l'absence de l'IL on peut l'utiliser sans problème.

Les modules 8 à F sont libres à l'utilisation sauf le module E si le lecteur de cartes est présent car le dit lecteur de cartes prend toujours cette place.

A ce moment il est important de noter ceci : les accessoires cités plus haut ont leurs programmes placés aux adresses citées quelle que soit leur position matérielle à l'arrière de la 41C, sur une extension, ou incorporés directement dans la machine.

Tous les autres accessoires ont une adresse qui correspond à celle du port dans lequel ils sont insérés. Si deux accessoires ont la même adresse rien ne marche. Le plus souvent la 41C se bloque dés l'allumage. Il est donc important de noter la "carte mémoire morte" de votre 41C surtout si vous utilisez une extension ou un lecteur d'Eprom.

Ce lecteur d'Eprom dont il sera question plus loin est un cas particulier, puisque son adresse est réglable de façon interne (voir notice du constructeur).

Si vous utilisez un module de 4K son adresse sera 8, A, C ou E selon qu'il est dans le port 1, 2, 3 ou 4. Certains modules sont placés par construction aux adresses "hautes" soit 9, B, D ou F. Si le module fait 8K il occupe les adresses 8+9, A+B, C+D. ou E+F.

Chapitre 3

Signification des chiffres

La 41c, semblable en cela à tous les ordinateurs, ne traite que de signaux électriques. Quelle que soit la zone mémoire, ces signaux sont transcrits sous forme de chiffres, ces chiffres sont regroupés par Registres, Octets ou groupes d'octets.

3.1 Nombre ou lettres ? NNN, normalisation

Certaines instructions de la 41C agissent sur un registre complet. Les plus connues sont STO ou RCL, mais nous avons aussi ASTO, ARCL, VIEW et AVIEW...

Citer simplement ces instructions montre que le contenu de ces registres peut être considéré soit comme un nombre, soit comme des caractères (lettres, signes ou chiffres).

Comment savoir ?

1. un registre contient 7 octets ou encore 14 digits ou encore 56 bits. Ces trois façons de parler désignent le même contenu.
2. On convient par commodité de donner à certaines zones d'un registre un nom lié à la représentation des nombres et de numéroter les digits de 0 à 13 (de droite à gauche). Le nombre $+1,234567890^{21}$, par exemple, est représenté par :

	0	1	2	3	4	5	6	7	8	9	0	0	2	1
Numéro du bit	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Signe de la mantisse (MS)	↑													
Mantisse (M)		↑	↑	↑	↑	↑	↑	↑	↑	↑	↑			
Signe de l'exposant (XS)												↑		
Exposant													↑	↑

3. nous pouvons trouver dans un registre :
   — un chiffre positif ou négatif, avec un exposant positif ou négatif,
   — des caractères alphanumériques,
   — autre chose.., que nous appellerons Nombre Non Normalisé (NNN).
4. Si les digits 0 et 1 et de 3 à 12 sont compris entre 0 et 9, si MS=0 ou 9 et XS=0 ou 9, le registre contient un nombre décimal. Le signe + est représenté par 0, le signe - par 9. De surcroît, si l'exposant est négatif il est représenté par son complément.
   Par exemple, pour connaître la représentation de E-21 (E pour exposant de 10) faire 1000-21=979, on obtient les 3 digits de droite (signe compris) :

$$+1,234567890^{-21}$$

:0 :1 :2 :3 :4 :5 :6 :7 :8 :9 :0 :9 :7 :9 :

   Remarquons également que le nombre est toujours stocké en SCI 10 et que ni la virgule ni le E de l'exposant ne sont codés, étant toujours à la même place.
   Les variantes FIX ou ENG n'interviennent qu'à l'affichage.
5. Si le premier digit est 1, la HP considère que :
   — le premier octet n'est qu'un indicateur,
   — les 6 octets suivants sont des caractères et affichés comme tels.
   Les octets nuls présents à gauche, à droite ou au milieu d'une chaîne de caractères sont ignorés à l'affichage, celui-ci étant alors justifié à gauche.

Table 3.1 – Notation naturelle

Hex	affiché
A	⊗
B	;
C	<
D	=
E	>
F	?

6. Toute autre valeur des digits, par exemple une valeur supérieure à 9, entraîne l'existence d'un NNN.
   Un NNN peut être stocké et rappelé sans problème dans un registre d'état.
   Un NNN peut être stocké dans un registre habituel mais il est alors modifié par les instructions RCL, VIEW ou X<>–. Attention, cette modification intervient non seulement sur la valeur rappelée, mais encore sur le registre lui même.
   Cette modification est appelée normalisation.
   Si le signe de la mantisse était 0 ou 9 le NNN est transformé en décimal ordinaire, ne contenant aucun digit supérieur à 9, un digit supérieur à 9, par exemple B, est alors interprété comme sa contre valeur décimale 11 et le 1 de "retenue" ajouté au digit de gauche :

   $$1B ===> 21$$

   Si seul XS est anormal, la normalisation reporte sur le digit de gauche la retenue éventuelle. Si, de ce fait, il reste 0 en XS (cas du digit A) ce 0 est maintenu. Sinon un 9 est mis en place.
   Dans tous les autres cas la "normalisation" revient à remplacer la valeur MS par 1, le registre étant alors considéré comme alphanumérique.
   Un exposant "anormal", outre les problèmes de normalisation, provoque un affichage curieux, sans grande application pratique connue jusqu'à présent.
   La programmation synthétique ne permettant pas d'éviter les inconvénients de la normalisation, des fonctions en microcode ont été crées par le club PPC pour cela, mais pour s'en servir... voir chapitre 6.
7. les NNN sont en fait formés de digits hexadécimaux. Sous certaines conditions (FIX 9 ...) la HP est capable d'afficher ces digits hexadécimaux.
   Elle utilise les caractères disponibles à la suite des chiffres 0-9 dans la table des codes. On appelle cela la "notation naturelle"(voir table 3.1).

3.2 Des caractères

3.2.1 Représentation

Nous avons vu que la HP ne traite que des nombres, mais que ceux-ci sont parfois interprétés comme des caractères, lettres, signes ou chiffres.

Pourquoi faire simple, quand on peut faire compliqué ? Selon les moments, un même caractère pourra être représenté par un code ou un autre. Pour l'instant 3 cas sont répertoriés :
- l'affichage
- les imprimantes
- le microcode

Les deux premiers cas, que tout programmeur peut rencontrer sont indiqués dans la table des codes. Le troisième cas est donné Fig. 3.1, page 18 et Fig. 3.2, page 18.

Remarquons que dans les trois cas ces codes ne seront considérés comme des caractères que si la HP a été "prévenue" qu'il doit en être ainsi, c'est à dire :
- dans le registre ALPHA, quand on utilise les fonctions spécialisées ou AON, elle considère alors que tout ce qui se trouve dans ce registre représente un caractère,
- dans un registre normal aux conditions indiquées section 5, page 16,
- dans un registre normal avec la fonction ARCL, si le contenu est numérique, il est transformé en son équivalent en caractères et placé en alpha.

Astuce HP (buffer) : Comprenez-vous maintenant que (dans un registre) 10 4D 4F 4E 54 52 signifie :

10 : les caractères suivant sont alphanumériques

Fig. 3.1 – Caractères microcode : Affichage

Code des caractères quand ils sont utilisés par le microcode (référence : PPC-T Australie et Robert Groom, révisé par John McGechie)

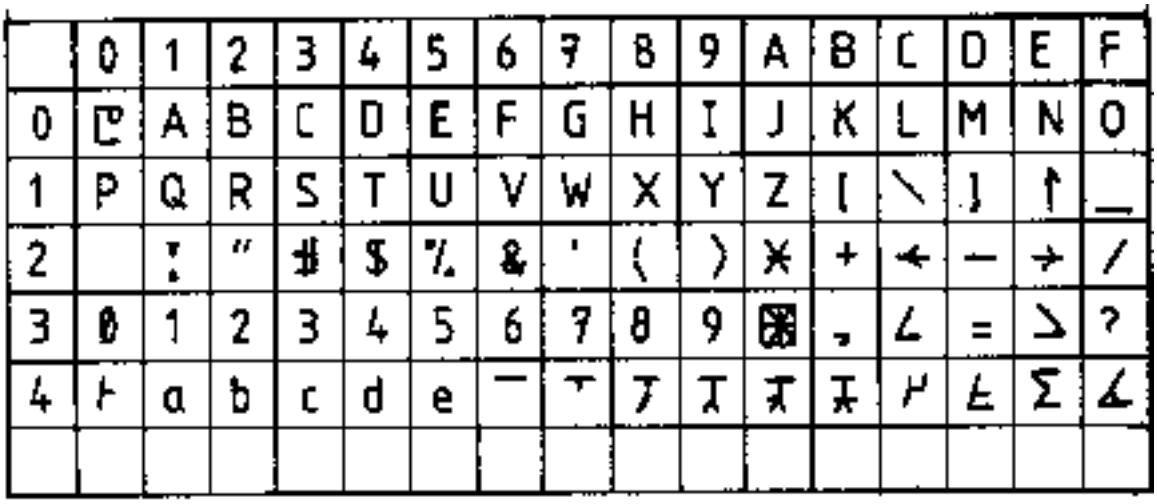


Fig. 3.2 – Caractères microcode : Imprimante

Code des caractères quand ils sont utilisés par le microcode (référence : PPC-T Australie et Robert Groom, révisé par John McGechie)

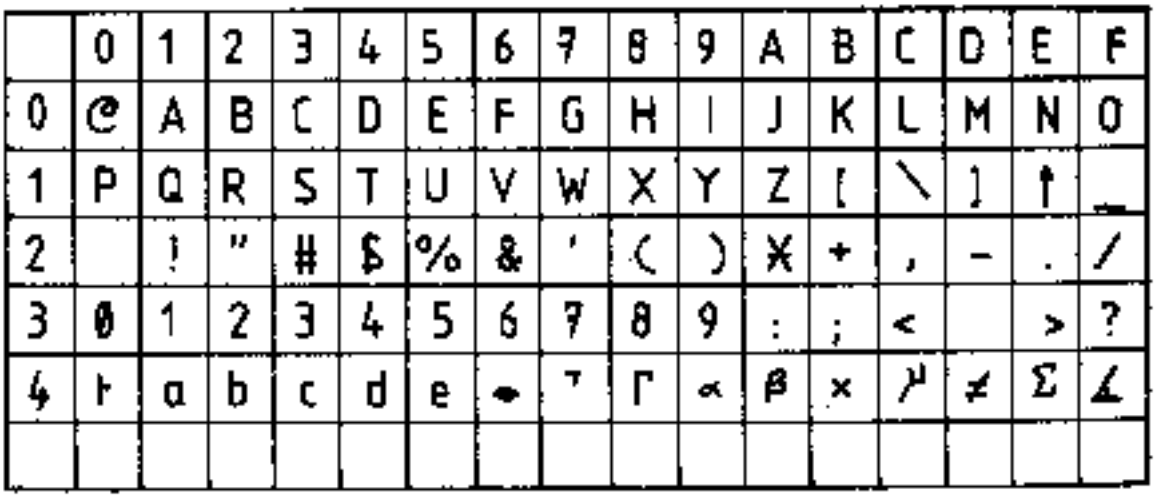
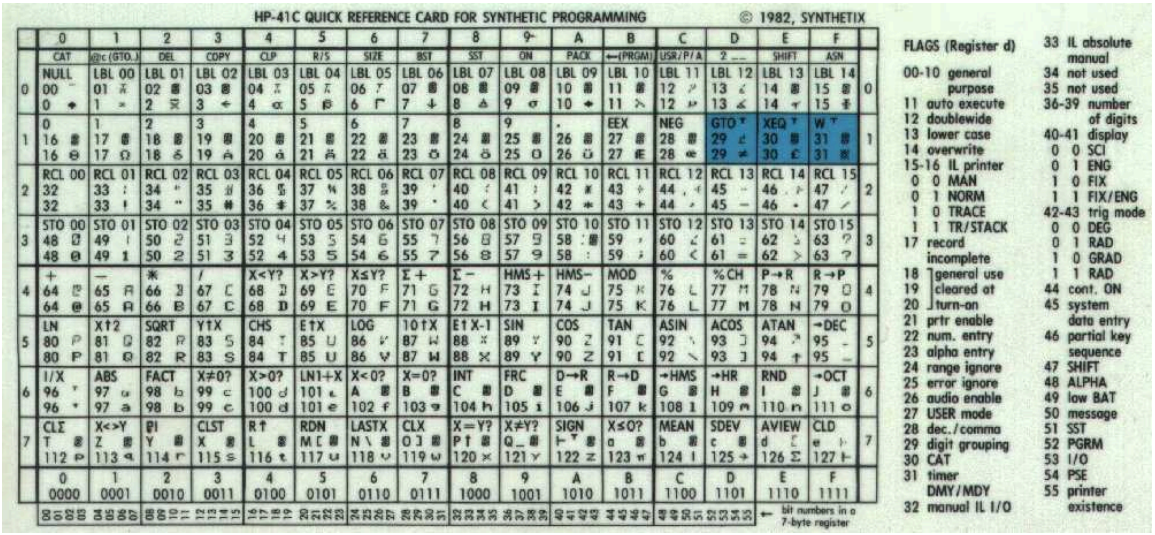


FIG. 3.3 – table des codes - recto, due à Keith Jarret



4D : M

4F : O

4E : N

49 : I

54 : T

52 : R

Si nous pouvions rappeler ce registre nous aurions MONITR à l'affichage. En fait si vous avez les moyens d'expérimenter avec le buffer, vous avez sans doute aussi accès aux fonctions en microcode de PPC ou au PPC ROM.

Il faut alors remarquer qu'une version primitive de ce module, diffusée sur EPROM au sein du club s'appelait HP-IL MONITOR. Dans le monitor comme dans le HP-IL Development on retrouve, aux adresses 879 (HEX) et suivantes du code les caractères qui sont implantés dans le registre d'état du buffer.

Dans une machine effacée (MEMORY LOST) faites 5, BSIZEX (SETBUFX avec l'EPROM MONITOR) suivi de 194 CHS NRCL (microcode) ou 194 RX (PPC ROM), vous obtenez bien "MONITR".

3.2.2 La table des codes

La table des codes (recto figure 3.3, verso figure 3.4) est un résumé des significations les plus courantes des codes de la HP-41C. Elle est organisée de façon que chaque valeur possible que peut prendre un octet soit représentée par une case.

Un octet peut-être représenté par 2 digits hexadécimaux, par exemple : 3A=digit 3 et digit A

Dans la table, le digit de gauche est celui du rang, il doit donc être lu à l'extrême droite de la table.

Le digit 3 désigne donc la quatrième ligne à partir du haut de la table, celle qui commence par STO 00.

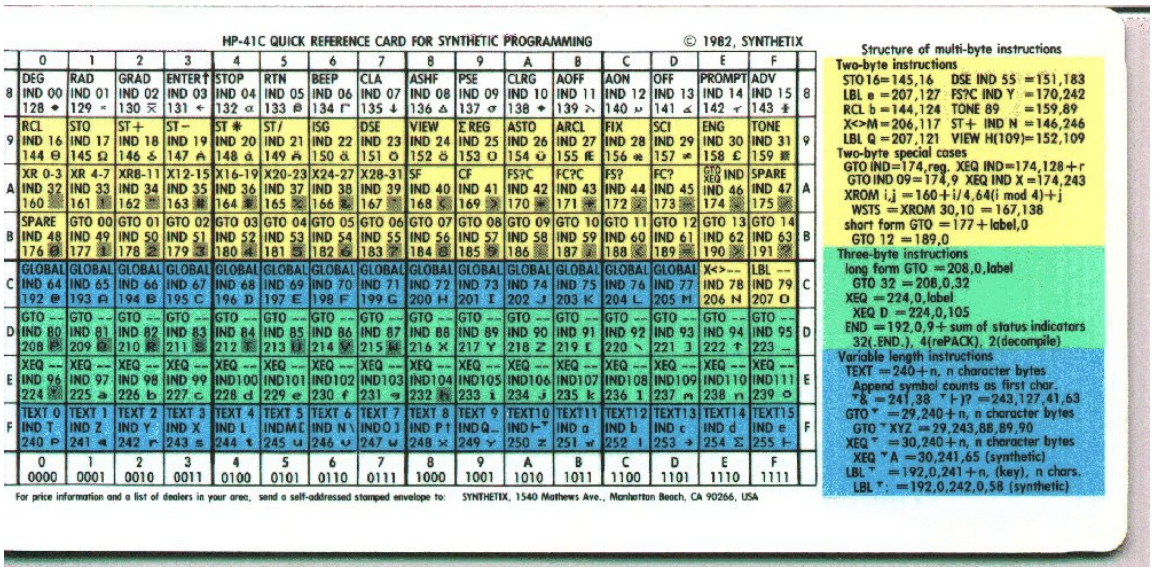
Le digit de droite est celui de la colonne ; il doit donc être lu au dessus ou au dessous de la table. Le digit A désigne donc la colonne commençant par LBL09.

A l'intersection de cette ligne et de cette colonne se trouve la case de l'octet 3A.

	STO 10	←zone 1
zone 4 →	58 : ⊗	←zone 2 (2 caractères)
zone 5 →	58 :	←zone 3

Nous distinguons 5 zones dans cette case, qui seront analysées ci-après. Disons immédiatement que la zone 1 représente la signification du code dans un programme (cf. 20et que la zone 5 représente la valeur décimale de l'octet considéré.

FIG. 3.4 – table des codes - verso, due à Keith Jarret



3.2.3 Zone 2

La zone 2 de la case représente l'interprétation du code faite par l'affichage. N'oubliez jamais que l'affichage est un périphérique de la HP comme le lecteur de cartes ou l'imprimante. La HP lui envoie des codes, l'affichage en fait ce qu'il veut. Voir en annexe 8.11, page 81, son comportement exact. Les octets 2C, 2E et 3A (celui de l'exemple), ont deux caractères représentés en zone 2. Remarquez qu'il s'agit des ",", "." et " :", qui disposent à l'affichage d'une colonne spéciale. Le deuxième caractère n'apparaît que dans des situations très rares.

L'affichage représente la nova ⊗ pour tous les octets de la deuxième moitié de la table, cette zone y a donc été supprimée.

3.2.4 Zone 3

La zone 3 de la case représente l'interprétation des codes, faite par les imprimantes thermique HP. C'est l'interprétation au standard HP ("8 bits"), l'imprimante HP-IL peut dans certains cas réagir différemment (se reporter au manuel).

Il est important de remarquer que l'imprimante ne réagit normalement qu'aux caractères de la première moitié de la table (code 00 à 7F).

Les codes de la deuxième moitié sont parfois ignorés, parfois imprimés, c'est pour cela qu'ils sont quand même signalés.

Mais surtout certains codes, quand ils sont transmis à l'imprimante, provoquent l'exécution de fonctions de l'imprimante.

On peut ainsi provoquer des sauts de caractères ou de colonnes (SKPCHR ou SKPCOL), imprimer le buffer de l'imprimante ou obtenir le même effet que les flags 12 et 13 (double largeur ou minuscules, voir tables 3.2, 3.3 ou 3.4).

L'imprimante HP-IL va même plus loin.

Remarquez que certains codes agissent même au cours du listage d'un programme, ce qui donne des résultats assez curieux, comme vous pouvez le voir avec le programme de démonstration, figures 3.5, 3.6, 3.7, 3.8, 3.9.

Pour Le microcode, L'utilisation se fait avec des instructions spéciales (voir chapitre 6).

3.2.5 Zone 4

La zone 4 : donne la valeur de L'octet utilisé comme postfixe (cf. section 3.3.2, page 22).

3.3 Des instructions (la zone 1)

Un octet permet 256 combinaisons. Ce nombre important ne l'est cependant pas assez pour une HP-41, et certaines instructions comportent plusieurs octets, même si elles n'occupent qu'une ligne de programme.

Table 3.2 – codes de contrôles des imprimantes thermiques HP de A0 à BF

A0 à AF	saute 0 à F caractères
B0 à B7	saute de 10 à 17 (décimal 16 à 23) caractères
B8 à BF	saute 0 à 7 colonnes

Table 3.3 – Codes de contrôle des imprimantes thermiques HP

	Largeur	type sortie	Majuscule/minuscule
D0	simple	caractère	M
D1	simple	caractère	m
D2	simple	colonne	M
D3	simple	colonne	m
D4	double	caractère	M
D5	double	caractère	m
D6	double	colonne	M
D7	double	colonne	m
E0	comme PRBUF		
E8	comme ADV		

3.3.1 Fonctions à un seul octet

Elles apparaissent dans la zone 1 des cases de la table, elles utilisent les codes Dl à 8F sauf les codes 1D, lE et lF. Quand la HP rencontre ces codes, elle exécute la fonction correspondante. Il n'y a pas de problème particulier à signaler sauf...

L'octet 00 (nul). Rencontré dans un programme, il est simplement ignoré, il sera généralement éliminé au prochain "PACKING".

Lors d'une modification d'un programme, une fonction effacée est remplacée par autant de nuls qu'elle comporte d'octets. Une fonction ajoutée prend la place des nuls disponibles là où on veut la mettre. S'il n'y en a pas assez la HP libère un registre entier en poussant les instructions suivantes vers le bas.

Le registre ainsi libéré est rempli de nuls.

Si deux nombres sont présents en mémoire programme sans séparateur (faites 1 alpha, alpha, 2) ils seront cependant séparés par un nul inamovible.

Des nuls peuvent également exister comme éléments de fonctions à plusieurs octets.

L'octet F0, tout à fait en bas à gauche de la table, est lui aussi une instruction d'un seul octet. Compte tenu de sa parenté avec les lignes de texte il sera étudié section 3.3.4 page 25.

Les octets lB et lC représentent respectivement EEX et NEG. EEX est l'Entrée d'un EXposant et apparaît sous la forme E (1 E2) dans un programme ou en alpha.

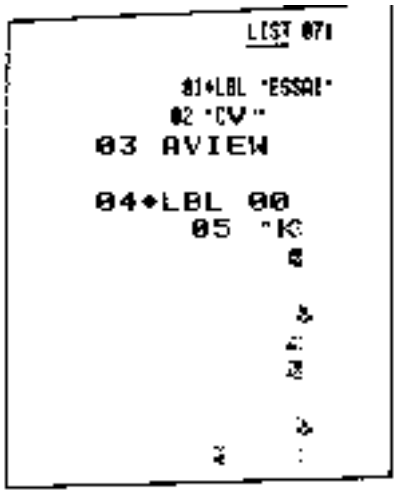
Table 3.4 – avec en plus, pour l'imprimante HP-IL en mode 8 bits

0D	Retour chariot
0A	A la ligne
80 à 8F	en mode colonne, prépare à lire et imprimer à octets de codes barre
C0	format : si dans la première cellule, centré, si au milieu texte séparé en 2, l'un à gauche, l'autre à droite
FC ou FD	Mode échappement
FE	ADV active
FF	ADV ignorée

Table 3.5 – affichage des postfixes

M = [	P = ↑
N = \	Q = _
O =]	⊢ = ⊤

Fig. 3.5 – list essai



NEG est l'effet de CHS agissant directement sur une entrée numérique (négation) comme le - de 1 E-2. Curieusement 25 CHS dans un programme est exécuté plus rapidement que -25 (avec le même effet, ne pas confondre avec la soustraction), allez savoir pourquoi ? (au fait, pour le faire, 25 ALPHA ALPHA CHS).

Les octets AF et 80 sont des NOP (Non OPérants) comme préfixe.

3.3.2 Fonctions sur deux octets

Le premier octet est appelé Préfixe et le second Postfixe. Les préfixes sont donnés aux cases 90 à BF plus CE et CF. La zone 1 de la case représente la fonction. Tous les octets de 00 à FF peuvent être utilisés comme postfixes. Le sens est alors celui de la zone 4 de la case. Ainsi :

RCL 79 est codé 90 4F

RCL IND 32 est codé 90 A0

Remarquons que, pour les codes 00 à 63, la valeur du postfixe n'est autre que la valeur décimale de l'octet. Remarquons surtout que, par des moyens décrits plus loin (chap. 5 page 33), il est possible effectivement d'affecter, à n'importe qu'elle fonction à 2 octets, n'importe quel postfixe (Le résultat n'est cependant pas toujours utile).

La zone 4 donne la façon dont ce postfixe est affiché par la HP et, quand elle est différente, l'interprétation par l'imprimante (voir tableau 3.5)

Remarquons aussi que les postfixes de la deuxième moitié de la table correspondent aux fonctions indirectes. Cette indirection est donnée par le postfixe et non pas par le préfixe.

L'octet AE a une double fonction : GTO IND si le postfixe vient des octets OO-7F, XEQ IND s'il vient des octets 80 à FF.

AE 08 est GTO IND 08 alors que AE 88 est XEQ IND 08

Les octets A0 à A7 s'appellent XROM. Ces octets sont des préfixes, mais pas exactement dans le même sens que les autres.

Chaque fonction d'un périphérique est associée à un code à deux octets. Celui-ci apparaît quand le périphérique est absent ou si le programme, préalablement enregistré sur carte ou cassette, est lu par une machine non pourvue de ces fonctions. C'est Le cas de WDTA (Write DaTA, enregistrement des données du Lecteur de cartes) ou d'ACSPEC (ACCumulate SPECiaL, traitement d'un caractère spécial par l'imprimante), y compris les fonctions non programmables (VERification, du lecteur de cartes), Le nom des modules comme "CARD READER", et même si le module est intégré dans la machine, comme les fonctions horloge de la 41CX.

FIG. 3.6 – prp essai

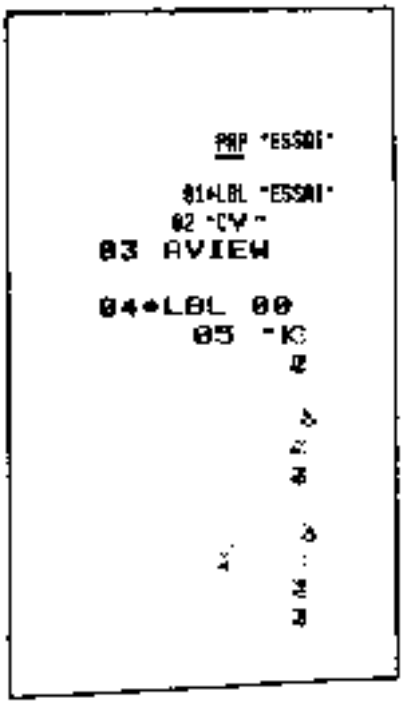


FIG. 3.7 – run essai

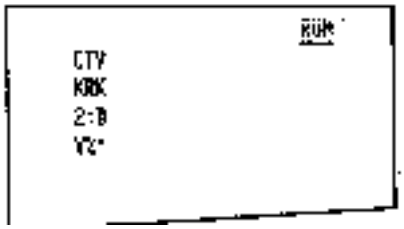


FIG. 3.8 – list cv

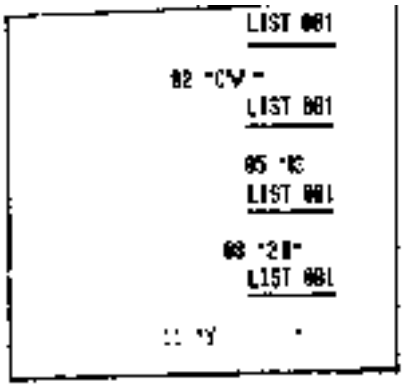


FIG. 3.9 – Affichage CV

Certains programmes sont bien difficiles à lister ! A l'affichage, on, voit pour les lignes de texte :
02 ⊤C⊗V
05 ⊤K⊗K
08 ⊤2⊗D
11 ⊤Y⊗⊗
Tirez-en les conclusions que vous voudrez !
(solution figure 3.10)

FIG. 3.10 – Solution

RUN donne la solution : le code de contrôle est donné par le deuxième caractère imprimé, à condition de le chercher dans la deuxième moitié de la table. Après %, n'importe quel caractère de la deuxième moitié de la table qui ne soit pas caractère de contrôle convient (ici caractère A2).

Ici il va falloir descendre au niveau du bit pour comprendre. Deux octets font 16 bits. Les cinq premiers bits (à gauche) servent de préfixe (10100), restent 11 bits pour Le postfixe. Ces 11 bits sont divisés en 5 bits pour Le numéro du périphérique et 6 bits pour Le numéro de fonction dans le périphérique.

Avec 6 bits on peut compter en décimal jusqu'à 64 (en fait de 0 à 63), avec 5 bits on peut compter jusqu'à 32. IL y a donc 32 numéros possibles pour les périphériques et 64 fonctions possibles dans Chaque périphérique.

Voyons un exemple :

WDTA a Le numéro XROM 30,07, soit en binaire

XROM=10100

30=11110

07=000111 soit

1010	0111	1000	0111
A	7	8	7

d'où les codes XROM 30,07=A7 87.

Remarquons également que L'octet de préfixe laisse de côté les deux bits de droite du numéro de périphérique. C'est ce qui explique que, à chaque case de la table, correspondent 4 périphériques (2 bits=4 possibilités).

Cases 90 et 91 : dans le but de gagner de la place en mémoire, HP a prévu des fonctions STO et RCL à un seul octet pour les registres les plus utilisés (rangs 2 et 3 de la table). Il est possible, mais peu profitable de construire des RCL et STO à deux octets avec des préfixes 90 et 91 et des postfixes de 00 à 0F. En fait, avant exécution, dans le microcode, la HP effectue cette transformation de 1 vers 2 octets.

Un problème similaire se pose pour les labels numériques. Une remarque supplémentaire s'impose ici. Quand on affecte à un Label (préfixe CF) un postfixe 66 à 6F, la table montre que l'on obtient LBL A à LBL J qui sont les labels "locaux" A-J. On voit ainsi que les labels locaux ne sont que des labels numériques camouflés (de 102 à 111).

Rang B : A part La case B0 qui n'est pas utilisée comme préfixe, nous avons affaire à un cas particulier de fonctions à deux octets.

Le préfixe est à lui tout seul la fonction, GTO 01 ou GTO 13. Le deuxième octet est nul à l'introduction du programme. A la première exécution du GTO, la HP va calculer la distance de saut et l'enregistrer dans cet octet libre. Cette opération est appelée compilation.

Un octet numéro 2 nul indique que La compilation n'a pas été faite. Après compilation, ce deuxième octet est organisé ainsi, si on numérote les bits 7 6 5 4 3 2 1 0 on a :

bit 7=sens de saut : 1 si le saut se fait vers une ligne de programme de numéro plus grand, 0 dans le cas contraire,

bits 0, 1, 2 et 3 = nombre entier de registres,

bits 4, 5 et 6 = nombre d'octets restants.

La distance est comptée ainsi : "nombre d'octets situés entre la fin du GTO et Le début du LBL", ce qui fait que dans un saut en avant ni Le GTO ni Le LBL ne sont comptés, alors que dans un saut en arrière les deux octets du GTO et L'octet du LBL sont comptés.

La distance maximale du label est donc F registres et 7 octets ou 112 octets.

La distance minimale est 0.

Si le label est trop Loin, Le deuxième octet du GTO est Laissé à 0 et la HP effectue tout ce travail à chaque fois...

3.3.3 Fonctions sur 3 octets

Ce sont Les rangs D "GTO" et E "XEQ".

Le cas est similaire à celui des GTO à 2 octets mais avec une distance de saut plus grande qui justifie l'encombrement supplémentaire. Remarquer que c'est la seule forme possible pour les XEQ.

Fig. 3.11 – Compléments sur Les XROM

On peut s'étonner que j'indique comme préfixe des XROM un nombre de 5 bits.
Ceci est dû au fait que Les XROM n'utilisent que la moitié de la ligne A, Le 5ème bit est donc toujours nul.
Cependant il nous faut parler ici d'un phénomène particulier qui est celui des affichages de XROM synthétiques.
Vous verrez (figure 5.2, page 39) un programme qui vous permet d'assigner à une touche n'importe quelle paire de codes qui viendront alors se placer dans les registres d'assignement aux emplacements prévus pour les codes de fonction.
Les seules fonctions qui peuvent légalement occuper deux octets dans les registres d'assignement sont les fonctions XROM.
Quand la HP lit une fonction assignée, le programme chargé d'afficher le nom de La fonction se contente de vérifier que le premier digit de la fonction est différent de 0 et affiche alors un numéro XROM calculé à partir des 3 digits suivants.
La HP suppose (bêtement ?) que si le premier digit est $\neq 0$, il ne peut être que A. Le même message XROM-,- correspond donc à 15 fonctions (premier digit compris entre 1 et F). Ce type d'XROM (qui s'affiche, mais ne s'inscrit pas dans un programme) peut donc apparaître jusqu'à XROM 63,63.
Si x est la valeur décimale du premier octet de la fonction et y la valeur du deuxième octet, XROM i,j est calculé par :
i = 4 (x MOD 16) + INT (y/64)
j = y M0D 64
On peut déduire de cette formule une méthode pour trouver sans calcul le numéro XROM à partir de la table des codes (méthode décrite par Keith Jarett).
Pour connaître le numéro XROM d'une instruction (ST+ IND M par exemple), voir que ST+ (le premier octet) est dans la même colonne (2) que XR 8-11, en effet le numéro de colonne n'est autre que le code de l'instruction modulo 16 ; donc nous avons un XROM 8, 9, 10 ou 11 selon le deuxième octet du code de l'instruction (IND M). Les lignes horizontales épaisses séparent la table en 4 zones horizontales qui correspondent respectivement au numéro. Ici, la première zone correspond à XROM 08,- ; la deuxième à XROM 09,- ; la troisième à XROM 10,- ; la quatrième à XROM 11,-.
La deuxième partie du XROM est donnée par La position de l'instruction dans La zone. Cette position est le numéro de La case dans la première zone ; un simple report à la première zone permet de le connaître quand le code vient des zones 2, 3 ou 4. Ici, IND M a Le numéro 53, ST+ IND M sera donc XROM 11,53.
De la même façon, si vous assignez avec le programme KA (figure 5.2, page 39) ADV IND e, donc 143/255 (décimal) vous aurez à l'affichage XROM 63,63 et en programme ADV, tout simplement. Bien sûr ADV IND e n'a aucun sens, et de plus l'usage de La table n'est pas si commode pour les XROM supérieurs à 31.

3 octets font 24 bits, ils sont organisés ainsi :

2 bits servent de préfixe : 11

2 bits donnent Le type GTO = 01, XEQ = 10

3 bits donnent Le nombre d'octets restant (de 0 à 7)

9 bits donnent Le nombre de registres complets (de 0 à 511)

1 bit donne la direction : 1 vers L'avant, 0 vers L'arrière

7 bits donnent La description du label par exemple :

1101 :	100	0 :0000 :0010 :	1	000 :0100
GTO	4 octets	2 registres	+	LBL 4

Cet exemple a été choisi pour montrer qu'il est parfaitement possible d'utiliser un GTO à 3 octets pour chercher un label à 1 octet... Les codes ici sont D8 02 84.

Le compte des octets se fait entre la fin du premier octet du GTO ou XEQ et le début du LBL. Donc dans un saut vers l'avant , les deux octets de compilation du GTO ou XEQ sont comptés, mais pas le LBL. Dans un saut vers l'arrière, le premier octet du GTO ou XEQ est compté ainsi que les 1 ou 2 octets du LBL.

3.3.4 Chaînes de caractères

Les octets de rang F sont des préfixes annonçant des chaînes de caractères dans un programme. Nous avons vu comment on identifie les caractères dans un registre. Dans un programme, cette identification se fait par un octet Fn où n est le nombre de codes qui, lus après le préfixe, doivent être considérés comme des caractères et placés en alpha.

Le maximum est donc FF, soit 15 caractères. Le minimum est F0. Il n'y a aucun caractère à mettre en alpha et effectivement alpha n'est pas perturbé. En fait ce code est utilisé pour caractériser les registres d'assignement et ne peut être obtenu simplement dans un programme.

Introduit dans un programme de façon artificielle, il se révèle non opérant (NOP) et par là même, utile comme bouche trou dans certains cas après ISG ou DSE.

Exemple :

"DODIN" = F5 44 4F 44 49 4E

Le caractère ⊢ (7F) placé juste après le préfixe signale à la HP que le registre alpha ne doit pas être effacé avant d'y placer la nouvelle chaîne (fonction APPEND).

3.3.5 Labels globaux - END

Les octets C0-CD "Global", jouent un double rôle : ils identifient à la fois le END et les labels alphanumériques.

Les END

Si le troisième octet d'une ligne commençant par l'octet Ca (avec a compris entre 0 et D) est un octet de texte Fn, alors la ligne est un label. Sinon c'est un END à 3 octets.

Dans les deux cas les deuxième, troisième et quatrième digits (à partir de la gauche) donnent la distance à laquelle se trouve le END ou LBL alpha précédent dans la mémoire (dans l'ordre du CAT 1). Cette distance est codée comme pour les STO à 3 octets ; elle est comptée de début de label en début de END etc.

Un label (ou END) commence par Cx avec en binaire :

1 1 0 0 a b c d

l100 = C

abc = reste des octets comptant la distance

d, joint à l'octet suivant= nombre de registres.

Attention ! CE = x<>– et CF = LBL numérique, donc on ne peut pas avoir abc = 111 car quelle que soit la valeur de d on aurait CE ou CF. Ceci n'est pas grave sauf que, quand le nombre de registres est maximum on ne peut y ajouter que 6 octets contre 7 pour les GTO et XEQ à trois octets.

Ceci procure le "chaînage" des labels en mémoire. Un GTO ou XEQ alpha commence à chercher un label à partir de la fin de la mémoire (le .END. permanent), et remonte de LBL en END jusqu'au premier label qui est donc identifié par les deux premiers octets C0 00.

Dans le END les 5ème et 6ème digits (à partir de la gauche) sont utilisés pour donner des informations :

Le 5ème digit est 2 pour END permanent (.END.), 4 pour un END privé, 6 si c'est le .END. et qu'il est privé.

Le 6ème digit est 0 pour le .END. après un MEMORY LOST, 9 si le programme n'est pas packé, D si Le programme est packé.

Les Labels

Il s'agit des labels alphabétiques. Les deux premiers octets sont donc formés de C suivi des 3 digits de distance. Le troisième octet est un préfixe de texte Fn, le quatrième octet et premier caractère, invisible à l'utilisateur sert à enregistrer un éventuel assignement (code de la touche, le même que dans les registres d'assignement). Les octets suivants épellent le nom. Du coup, le n de Fn est de 1 plus grand que le nombre de lettres du nom.

Un LBL "DE", seul en mémoire, assigné à la touche (A) est codé C0 00 F3 0l 44 45.

Les codes 16 et lE sont ceux de GTO et XEQ alpha, ils sont suivis d'une chaîne de caractères ordinaire, par exemple :

GTO "DODIN" = 16 F5 44 4F 44 49 4E

Enfin l'octet lF qui s'affiche $W^{\top}$ est inactif comme préfixe. Fin 1983 différents usages de ce code étaient discutés dans le club PPC-T sans qu'une application commode soit encore trouvée.

Les XEQ "ALPHA" se changent en XROM "ALPHA" si le programme en question (ALPHA) est un programme en langage courant (et non en microcode) placé dans un ROM.

C'est le cas par exemple des programmes du module Math.

3.4 Organisation des programmes en ROM

Cette organisation est en grande partie nécessitée par la possibilité d'usage de la fonction COPY, qui permet de recopier un programme en mémoire vive.

3.4.1 Les mots de contrôle

Le programme en ROM est précédé par un mot de code indiquant en hexa le nombre de registres nécessaire pour copier le programme, arrondi au dessus.

Le deuxième mot donne (codage 244) : Si le programme est privé, digit de gauche égal à 2 (binaire11), à 1 (01) sinon. Le digit de droite est toujours à 0 (0000). Le digit du milieu donne le nombre d'octets utilisés dans le registre incomplet.

Les mots suivants sont les octets du programme. Les deux bits supplémentaires (puisque les mots en ROM font 10 bits au lieu de 8 en RAM) sont à gauche. Ils valent 01 si l'octet est le premier d'une instruction, 00 autrement. En fin de programme, le dernier octet du END (et donc du programme) est 22F.

3.4.2 Les chaînages

Bien sûr, à moins d'être maso, il faut placer en ROM des programmes compilés et donc avec les octets de distance convenablement remplis.

Les sauts des GTO et XEQ sont codés directement en octets, ce qui permet une distance maxi légèrement supérieure à celle utilisable en RAM. Du fait de la compilation les labels ne sont plus utiles (sauf pour copier en RAM le programme). La distance se calcule avec : adresse du dernier octet inclus, moins adresse du premier octet inclus plus 1.

Le chaînage des labels alphabétiques est fait comme en RAM.

Pour le premier label une remarque s'impose. Lors de la copie en RAM la HP libère le nombre voulu de registres sous le .END.. Celui-ci est transformé en END normal. Le bloc de registres ainsi libéré est au fond de la mémoire, son END est donc le nouveau .END. et doit être cadré a droite de son registre.

L'ensemble du programme copié est donc tassé vers le bas et c'est le premier registre qui est incomplet on a :

x x x x E N D programme précédent en RAM

x x x L A B E début du programme copié

L x x x x x x

Par exemple. Le nombre d'octets occupés de ce premier registre étant le nombre qui figure dans le deuxième mot de tète. Il faut tenir compte de cette disposition pour calculer la distance entre le premier label du programme et le END suivant : les octets vides du premier registre doivent être comptés.

Chapitre 4

Une zone particulière : les registres d'état

Les registres d'état sont les 16 registres dont l'adresse est comprise entre 000 et 00F. Ils servent à la HP de brouillon, d'aide mémoire ou de mémoire tout court, la carte de ces registres est à la fin du livre, fig. 36, pl10.

4.1 La pile opérationnelle

La fameuse pile opérationnelle de HP est un groupe de 4 registres X, Y, Z, T.

Ces registres sont placés à la base de la mémoire ; T est le registre R(000), Z est R(001), Y est R(002) et X est R(003), le registre l (LASTX) est simplement R(004).

Ces registres n'ont pas d'autre caractéristique particulière. C'est leur manipulation par le microcode qui fait leur particularité.

Il y a quand même un autre détail. Nous venons de voir que la pile opérationnelle est à des adresses fixes en mémoire. Par contre les registres courants (R00...R99...R319) sont à des emplacements variables et peuvent même ne pas exister.

Nous allons voir que la HP doit vérifier l'existence de ces registres, puis calculer leur position à chaque fois qu'elle s'en sert. Par contre les registres de la pile sont toujours là. C'est pourquoi un ISG ou DSE est beaucoup plus rapide dans la pile que dans un registre.

Au fait, avez-vous remarqué que les instructions STO l (STO.l) ou RCL l (RCL.l) existent d'origine ? Si la première est parfois utile, la deuxième fait double emploi avec LASTX et utilise un octet de plus.

4.2 Le registre alpha

Là, ça devient plus intéressant !

"Le" registre alpha est connu pour faire 24 caractères, n'est-ce pas ? Et bien, en fait ce "registre" est constitué de 4 registres voisins appelés M, N, 0 et P qui sont les registres R(005), R(006), R(0071 et R(008).

Encore plus intéressant, ces registres peuvent être utilisés comme des registres ordinaires.

Rappelez-vous que nous avons dit que, dans un programme, à un préfixe donné peut être affecté un postfixe quelconque. Et bien, de 0 à 99 (décimal) nous avons les registres normaux, de l00 à 111 nous avons les registres de même numéro, et de 112 à 127 nous obtenons le registres d'état !

Le code 90 75 ou RCL 117 s'affiche en réalité RCL M. Il ne s'agit plus du 117 ième registre, mais du registre d'état M, registre de droite de alpha. Voyez chapitre 5 le mode d'emploi.

Quand vous introduisez un caractère dans le registre alpha, il est placé à droite du registre M. A mesure que vous introduisez des caractères en alpha, le premier caractère est poussé vers la gauche dans N, dans 0 puis dans P. Plus extraordinaire encore le caractère est poussé jusqu'à la gauche de P avant de disparaître à jamais. Nous avons alors 4 x 7 = 28 caractères en alpha ! Cela ne durera pas très longtemps, car les 4 octets gauche de P (et donc de alpha) sont souvent utilisés comme brouillon par la HP. De plus, les caractères qui se trouvent à cet endroit n'apparaissent pas à l'écran, cette propriété est cependant parfois utile.

Les registres M N 0 P peuvent donc être utilisés :
— normalement avec les fonctions alpha,

— directement avec STO, RCL, VIEW, ISG... etc, comme registres de stockage, de comptage... supplémentaires avec les. même avantages de rapidité que la pile.

Le mélange de ces deux façons de travailler permet un contrôle accru du contenu de ces registres.

Les autres registres d'état sont normalement invisibles pour l'utilisateur, mais pas pour nous !

Ils ne sont en général pas utilisés en bloc par la HP, mais des zones de certains registres ont un usage spécifique. Voyons les les uns après les autres.

4.3 Le registre P (adresse 008)

Les 4 octets de gauche du registre P sont utilisés comme brouillon par la machine. Par exemple, pendant un catalogue, le digit le plus à gauche (MS) de P contient le numéro de catalogue (1, 2 ou 3) et les autres digits le nombre de fonctions déjà affichées au cours du catalogue (en hexadécimal).

Il serait fastidieux et peu commode de lister les différents cas d'usage de P par la 41c. Un essai est préférable si l'on espère l'utiliser. Attention aux périphériques !

4.4 Le registre Q (adresse 009)

Ce registre est un brouillon, mais très important puisque c'est là que la HP place le nom des fonctions que vous épelez après un XEQ.

Ce nom est placé en ordre inverse, écrit de droite à gauche les octets non utilisés sont à 00. Ce registre est utilisé dans bien d'autres cas, en particulier par l'imprimante.

En fait, il est utilisé si souvent qu'il est impossible de s'en servir pour autre chose.

4.5 Le registre X (adresse 00A)

Ainsi appelé (si l'on peut dire) car il est affiché de cette façon par la HP.

Les 5 digits de droite sont utilisés comme brouillon. Mais les autres sont beaucoup plus intéressants. Ils représentent l'index des touches assignées.

Vous savez que chaque touche de la 41C peut être assignée. Il serait beaucoup trop long pour la HP de chercher dans toute la mémoire un éventuel assignement à chaque pression de touche. Par conséquent, la machine tient à jour un index des touches qui ont été assignées et n'entame la recherche que si cet index signale l'existence de l'assignation.

Il y a 35 touches au clavier ; il faut donc 35 index et donc 35 bits. 4 octets font 4 x 8 = 32 bits, il faut donc un digit de plus, soient au total 36 bits.

Il y en a un de trop, mais abondance de bien ne nuit pas !

La position du bit est représentée Fig. 8.3 page 83 par la lettre correspondant à la touche.

4.6 Les registres a et b (adresses 00B et 00C)

Ces registres sont les registres de *pointeurs*. Un pointeur est un registre (ne me dites pas qu'on tourne en rond) qui donne le numéro de l'octet de programme qui est en cours de traitement, autrement dit l'adresse de cet octet.

Le principal pointeur est le pointeur de programme, celui qui nous dit quel est l'octet de programme qui est en cours d'exécution. Ce pointeur est formé des deux octets de droite du registre b.

4.6.1 ROM

Quand le pointeur se trouve dans un module de mémoire morte, par exemple dans le PPC ROM, le pointeur de programme représente l'adresse de l'octet en cours de traitement, exprimée en 4 digits hexadécimaux de 0000 à FFFF, ce qui autorise 65 536 octets. Ces octets ont un numéro qui augmente dans le même sens que les lignes de programme.

4.6.2 RAM

Quand le pointeur se trouve dans la mémoire vive (RAM), par exemple en train d'exécuter votre programme favori, il a une forme différente.

Les 3 digits de droite du registre b donnent l'adresse du registre dans lequel l'octet traité se situe et le digit numéro 3 donne le numéro de l'octet dans le registre. les adresses de registres, décroissent quand le numéro de ligne augmente.

4.6.3 La pile de retour des sous programmes

Vous savez que, quand vous exécutez l'instruction XEQ dans un programme, la HP doit noter la position de cet XEQ pour pouvoir y revenir, après avoir exécuté le sous programme désigné. Où est donc notée cette adresse ?

Tout simplement à coté. Au moment du XEQ, avant de chercher l'adresse du programme à exécuter, la HP pousse le pointeur de 2 octets vers la gauche, il se retrouve donc dans les octets numéro 2 et numéro 3 du registre b. En fait, c'est tout le registre b qui est poussé à gauche, et ce qui sort à gauche de b rentre à droite de a. Ce qui est poussé à gauche de a est perdu.

Il y a donc en b le pointeur en cours d'usage, plus 2,5 pointeurs en réserve et on a le demi qui manque en b et 3 autres pointeurs en réserve, donc au total 6 appels de sous programme enregistrés. Si un septième appel a lieu, le premier appel est perdu.

Ce serait presque trop simple ! C'est une astuce dans le stockage des adresses de retour des sous programmes qui va limiter le plus la mémoire vive utilisable de la 41C, du moins pour l'exécution des programmes.

Dans la mémoire de la 41C il n'y a pas été prévu d'utiliser de programme à des adresses supérieures à 1FF. Donc le digit numéro 2 du pointeur est en binaire 0000 ou 0001. Il y a donc toujours 3 bits libres.

D'autre part, il n'y a que 7 octets par registre, donc le digit numéro 3 du pointeur est au maximum 7 = 0 1 1 1 et il n'y a que 3 bits utilisés. les ingénieurs HP n'ont pu résister à la tentation de placer les 3 bits du digit 3 dans les 3 bits vides du digit 2.

Pourquoi libérer ainsi un digit du pointeur ? Tout simplement pour faire la différence entre ROM et RAM. En effet, cette contraction n'est possible que dans la mémoire vive. De plus, les programmes situés en mémoire morte aux adresses 0xxx sont ceux de la mémoire interne dans laquelle il n'est pas question d'aller se promener. Donc si le digit numéro 3 est égal à 0, la HP considère que le retour doit se faire en RAM ; s'il est différent de 0, le retour se fait en ROM.

C'est ce phénomène qui fait que, s'il est possible (moyennant des manipulations spéciales) d'exécuter des programmes dans la mémoire du module x-fonctions, c'est beaucoup plus délicat dans les x-mémoires qui sont au dessus de 1FF et dans lesquelles, entre autre, il n'est pas possible de faire fonctionner des XEQ.

La manipulation de ces pointeurs de retour est à l'origine historique de l'exploration de la mémoire interne de la 41C.

En effet, comme nous l'avons vu, cette mémoire interne contient les adresses 0xxx, lxxx et 2xxx. les deux derniers groupes sont donc accessibles par construction d'une adresse de retour artificielle .

4.7 Le registre c (adresse 00D)

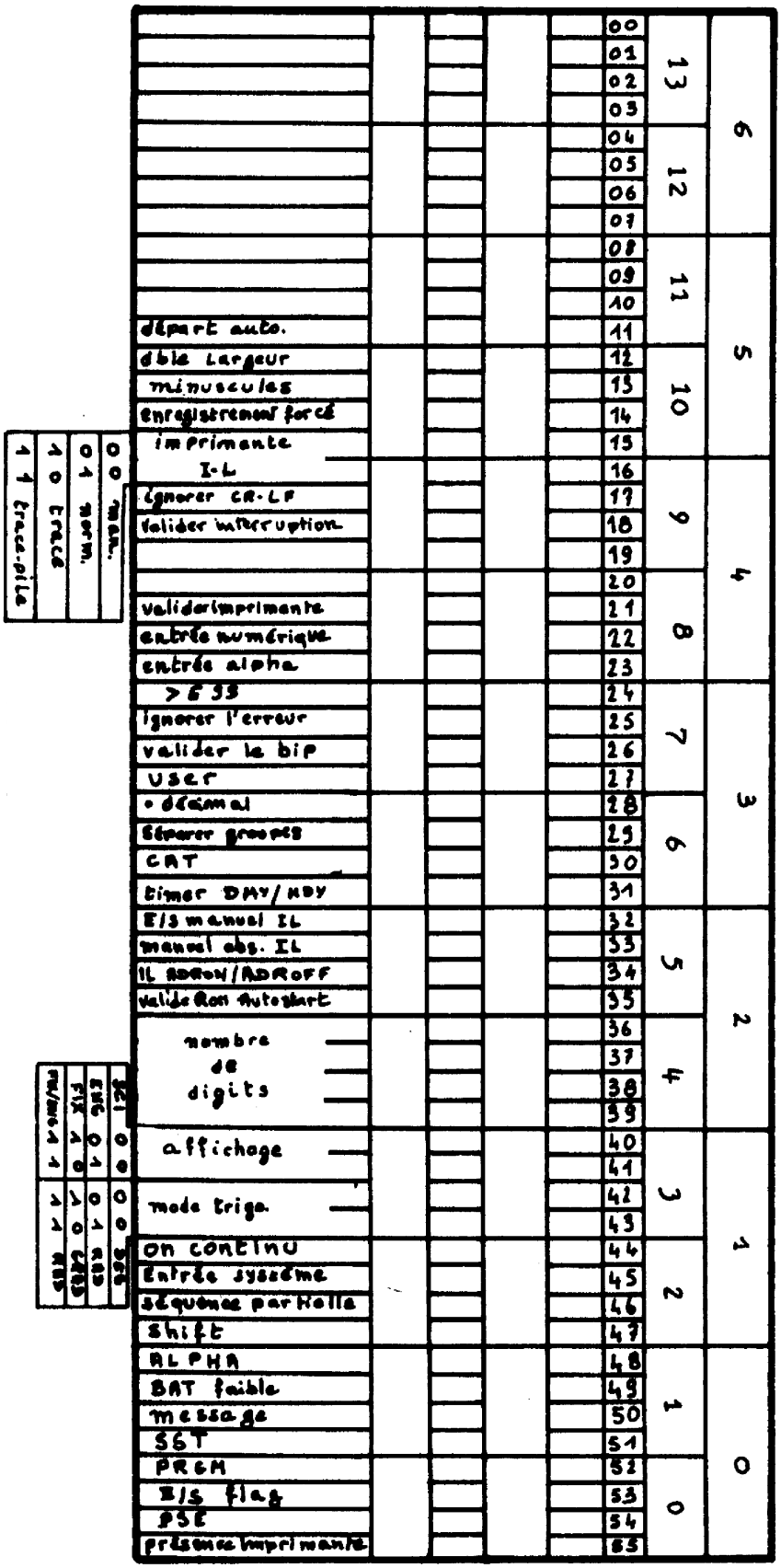
Un des plus dangereux et des plus intéressants des registres d'état. Ce registre contient, de droite à gauche :

1. Sur 3 digits, l'adresse du .END. permanent, c'est à dire l'adresse du registre dans lequel est situé ce .END. En effet, le .END. est toujours situé dans les 3 octets de droite de son registre. 3 digits suffisent donc à définir cette adresse. Ceci est la limite inférieure dont nous avons parlé dans le chapitre 1.
2. L'adresse du premier registre de données, celui que nous appelons R00, ici aussi 3 digits. C'est la limite inférieure de la zone des données et la limite supérieure de cette des programmes.

   On appelle rideau entre programmes et données ce registre.
3. Le nombre 169, appelé constante de départ à chaud. C'est une valeur qui ne doit pas être touchée. La HP la contrôle à chaque arrêt d'exécution et même parfois un peu plus souvent. Si elle trouve 169 dans cette zone, c'est que la 41C est "chaude" et tout va bien, sinon elle exécute un "départ à froid" et vous voyez apparaître le sinistre MEMORY LOST !

   Cependant, au cours d'un programme, ce contrôle n'est pas fréquent et on peut jouer un peu ; ne vous en privez pas.
4. Les 3 digits de gauche donnent l'adresse du premier registre statistique (REG).
5. Restent deux digits inutilisés (9 et 10) qui servent de brouillon, en particulier pour l'imprimante.

FIG. 4.1 – Les flags



4.8 Le registre d (adresse 00E)

Dans ce registre sont placés les 56 flags de la 41C, 1 flag par bit, comme on le voit dans le tableau joint (Fig 4.1 page précédente). La possibilité d'accéder directement à ce registre permet à la fois de traiter les 56 flags "d'un coup" et de traiter le contenu d'un registre bit par bit. Comme les flags ne sont testés par la 41C que de temps en temps, on peut le plus souvent les manipuler sans problème en cours de programme, à condition de reconstituer le registre avant l'arrêt du programme.

Méfiez-vous quand même des flags spécialisés comme 45 (entrée de données système) ou 21 (validation d'imprimante).

Ce registre est un des plus utilisés, au point que HP a fini par le reconnaître en incluant dans le module X-fonctions des fonctions travaillant sur lui.

4.9 Le registre e (adresse 00F)

Les trois digits de droite contiennent le numéro de la ligne de programme en cours d'exécution ou FFF si ce numéro doit être recalculé.

Les deux digits suivants sont utilisés comme brouillon à plusieurs occasions par la HP.

Les autres digits, de 8 à 13, contiennent les index d'assignation des touches secondaires (shiftées) comme vu dans le registre -.

Chapitre 5

Au voleur

Depuis la découverte des registres d'état et des instructions à postfixes anormaux fin 79, de très nombreuses applications ont été utilisées. La meilleure application est le module créé par le club PPC et appelé PPC ROM. Ce module contient, parmi de nombreux programmes d'intérêt général un certain nombre de petits programmes utilisant beaucoup de programmation synthétique, comme on appelle ce type spécial de programmation. Ces programmes sont perfectibles mais apportent au programmeur un mieux évident, croyez moi, la publicité que je fais au PPC ROM est gratuite ! Il est disponible pour tous, même non adhérents du club, à un prix d'environ 1000F qui n'est peut être pas accessible à tous.

Mais le PPC ROM n'est pas indispensable, heureusement ! Une méthode très commode a été trouvée.

5.1 Le byte grabber (BG ou "VOLEUR")

Suivez à la lettre les instructions ci-dessous (une HP-41C toute nue est non seulement utilisable mais souhaitable, cependant un lecteur de cartes est le bienvenu).

Cette procédure fonctionne avec toutes les machines. Certains accessoires peuvent parfois perturber le fonctionnement, aussi prenez une machine nue pour commencer (HP 41C ou CV). Les nombres de registres indiqués sont ceux obtenus à partir d'une 41C standard ; avec une CV ou une CX ces chiffres sont différents (aucune importance).

(remarque : ne marche pas apparemment sur ma CX #2352S41976 - note du 17 oct 1998 - marche sans problème sur une 41C #2144S11951 - si tant est que ces numéros, relevés sur le boîtier arrière, aient quoi que ce soit à voir avec la carte mère !)

Sur une CX ou si vous avez laissé le module X-Fonctions, les nuls peuvent être remplacés par autre chose, peu importe. Sortez du mode programme, faites GTO.., c'est fini.

Avant de poursuivre, si vous avez un lecteur de cartes, enregistrez une carte d'état (WSTS) sur ses deux pistes.

Explication :

a, b, c exploite une particularité de la machine appelée "BUG 9" par le PPC Club et d'ailleurs déjà décrite sous une forme à peine différente en France (CRIC).

Le numéro de ligne 4094 qui apparaît fugitivement puis le numéro 4093 précédent DEC n'ont pas de signification, sauf celle de vous montrer que vous êtes maintenant au dessous du .END., en fait dans le registre R(0C0), premier registre d'assignement.

Refaites les opérations de 1 à 5d, puis, au lieu de faire e) GTO.005, faites

plusieurs BST (soyez patient). Vous voyez :

Quand vous faites GTO.005, la HP se ressaisit un peu et reprend un compte "normal" des lignes à partir de F0, le 05 LBL 03 qui apparaît est donc celui qui précède le +.

DEL 003 efface donc LBL 03, + et -, laissant trois nuls.

Quand vous introduisez la chaîne de caractères, elle recouvre les nuls : le premier octet est Fn où n va varier de 1 à 7 au fur et à mesure

de l'introduction des lettres, puis ?, octet 3F, puis A, octet 41 (tiens, le même code que - !) et les trois nuls sont recouverts. Mais vous êtes à droite de R(0C0) il n'y a rien dessous, les lettres suivantes tombent donc dans le vide et ne peuvent être enregistrées. Cependant, le n de Fn augmente bien à chaque fois d'une unité, et l'affichage montre comme des nuls (—) la

TABLE 5.1 – byte grabber

faire	voir
1) MEMORY LOST (en faisant, à partir d'une machine éteinte, ON en maintenant <- enfoncée)	MEMORY LOST
2) Assignez + à la touche LN	ASN + 15
3) Assignez DEL à la touche LOG	ASN DEL 14
4) Placez-vous en mode user	0.0000
5) Placez-vous en mode programme et faites ce qui suit	00 REG 45
a) SHIFT LBL ALPHA T ALPHA	01 LBL ⊤T
b) SHIFT CAT 1 et immédiatement R/S	LBL ⊤T
c) LOG A	DEL 001
	| 4094|
	|.END. REG 44 |
d) BST	[4093|
	| 4093 DEC |
e) SHIFT GTO. LN	GTO.005
	05 LBL 03
f) LOG C	DEL 003
	04 STO 01
g) ALPHA ? AAAAAA ALPHA	05 ⊤ ?A−−−−−

TABLE 5.2 – explication BG

Programme	code
4086⊤	F0
4087 LBL 03	04
4088 LBL 01	02 equivalent de DEL
4089 STO 01	31 code touche LOG
4090 LBL 03	04
4091 +	40 fonction +
4092 -	41 code touche LN
ici 176 registres vides ou le module X-F	
4093 DEC	ici sont les registres d'état

FIG. 5.1 – Le fonctionnement du Byte Grabber dévoilé

Faire, après un MEMORY LOST, GTO.. et entrez le programme suivant : O1 ENTER ; 02 ENTER ; 03 ENTER ; 04 ENTER ; 05 ENTER ; 06 ENTER ; 07 ENTER (7 fois ENTER) ; 08 "ABCDEFGHIJ" (FA suivi de 10 caractères = 11 caractères).
En tenant compte des 3 octets du .END. vous avez donc rempli 7+11+3=21 octets = 3 registres, un XEQ "PACK" vous l'assurera. Placez-vous sur la ligne 07 ENTER et effacez *3 lignes* vous voyez :
04 ENTER↑ BG, vous voyez alors
05 –⊗⊗ABCD
Que se passe-t-il ? Quand vous voulez introduire une fonction en mémoire, la
41C sait (croit savoir !) que cette fonction occupe au plus 3 octets (il y a une fonction de 3 octets qui peut être assignée, c'est ta fonction END ; en fait seul l'octet C0 figure dans le registre d'assignement, mais les octets suivants sont construits au moment de l'introduction de la fonction en mémoire). S'il n'y a pas assez de place disponible pour introduire la fonction souhaitée, la 41C libère un registre entier, donc 7 octets, ce qui lui parait plus que suffisant pour loger une instruction qui ne peut en faire que 3.
Mais le Byte Grabber a deux caractéristiques : c'est une fonction de trois octets (n'oublions pas que le Byte Grabber est une fonction qui recopie en mémoire les 3 octets contenus dans le registre d'assignement correspondant) et le premier octet qu'il place en mémoire est un octet F7 ; les deux octets suivants étant variables selon le BG utilisé. Le BG en mémoire est donc une chaîne de 7 caractères dont seuls les 2 premiers (et le préfixe) sont fournis ; les 5 suivants sont alors pris parmi les octets voisins dans la mémoire programme. Examinons les différents cas en fonction de la situation précédant l'exécution du BG.
1) aucun octet n'est libre. La HP libère 1 registre et place le F7 à gauche de ce registre. La chaîne affichée comprend F7, les deux octets suivants du BG, les 4 nuls restant du registre, et 1 octet du registre suivant, l'octet volé.
C'est le cas habituel d'utilisation.
2) Un seul octet est libre. Le F7 s'y place, la HP libère un registre pour les deux octets suivants du BG. La chaîne recouvre alors ces deux octets et les 5 nuls restants du registre libéré ; rien n'est volé mais tous les octets nuls ont été recouverts et on se retrouve dans le cas numéro1.
3) 2 octets sont libres. Le F7 s'y place, ainsi que le 2ième octet du BG ; la HP libère un registre pour le 3ième octet du BG ; la chaîne ne recouvre alors que 6 octets de ce registre, laissant un nul, ce qui nous ramène au cas de figure numéro 2.
4) 3 octets sont libres. Cela suffit pour le Byte Grabber, du coup aucun registre n'est libéré et le BG vole 5 caractères.
C'est le cas expérimenté ci-dessus. Si on augmente le nombre de nuls (les ENTER effacés), on peut donc choisir entre 0 et 5 le nombre de caractères volés.

partie de "vide" incluse dans la chaîne. Si le module X-F est présent ou si vous avez une Cx, il y a ici les registres de ce module et leur contenu, à la place du vide (attention, ce contenu va être modifié).

Vous terminez avec une chaîne de 7 caractères (?, A et 5 nuls) dont le premier caractère est ? et le deuxième A. Ce A est compris par la machine comme code de touche. Essayez une autre touche : TAN a pour code d'assignement 42, comme la lettre B. Essayez ? et 6 fois B.

Comment, ça ne marche pas ! Avez-vous réfléchi ? et oui, c'est maintenant la touche TAN qu'il faut assigner au pas 2.

Revenons au premier exemple (par exemple en lisant la carte enregistrée tout à l'heure) : Vous devez maintenant voir quand vous pressez la touche LN en mode calcul-user :

XROM 28,63

C'est un des multiples BG (Byte grabber = voleur d'octets) possibles.

Que fait le BG ? D'abord il est principalement utile en mode programme, et ne doit en aucun cas être utilisé avant un END ou dans une mémoire programme vide (méfiez-vous, c'est le cas actuellement).

Le BG crée en mémoire programme une ligne de texte de 7 caractères. Pour pouvoir la mettre en place, la HP libère un registre de 7 octets. Mais elle oublie que devant la chaîne il y a un F7 qui occupe un octet et que donc cette chaîne de 7 octets en demande en fait 8 en mémoire. Nous avons donc créé une chaîne

F7 00 3F 00 00 00 00

soit 7 octets, mais la HP veut à toute force trouver le 7ème caractère qui lui manque, et elle annexe purement et simplement le premier octet, nul ou non nul qui vient (pour les détails, voir figure 5.1).

Exemple, tapez :

0l LBL ⊤T

02 +

03 +

04 +

puis faites GTO.00l et BG (appuyez sur LN en mode user-programme) vous voyez :

02 ⊤− ?−−−−@ ; à droite vous avez le caractère qui a le code 40, le même code que +, qui a été volé et introduit dans la chaîne.

Effacez cette Ligne et tapez :

02 "ABCDEFG

faites BST et BG il reste :

01ªLBL".T

02 "ª ?ªªªª" = ⊤− ?−−−−@ <- Le F7 de de chaîne

03 -

04 *

05 /

06 X<Y ?

07 X>Y ?

08 X<=Y ?

09 S+

10 +

11 +

12 .END.

D'où viennent ces lignes de programme ? Ce sont simplement, comme vous pouvez le voir sur la table des codes, les instructions qui ont le même code que les lettres entrées précédemment. C'est le masquage du F7 de début de chaîne par le BG qui empêche la HP de savoir qu'il s'agit de lettres. Vous pouvez SST ou BST dans ce "programme" sans souci. Faites à nouveau GTO.001, BG vous voyez :

0l LBL "'T""

02 "ª ?ªªªª"

03 STO 15

04 "ABCDEFG

Que s'est-il passé ? le nouveau BG a volé le F7 du premier BG, c'est le voleur volé ! Les nuls ne se voient pas, STO 15 a le même code que ?, le F7 de la chaîne est libéré et la chaîne reprend son existence normale. Vous pouvez effacer les Lignes 2 et 3, packer et vous êtes revenu au départ. Nous appellerons cette opération "supprimer Le BG". Faites le, et voyons l'usage de cette nouvelle fonction.

A nouveau BG, vous retrouvez la disposition de la première figure. Effacez la ligne 06 X<Y ? et remplacez la par LBL 00 puis GTO.001 et supprimez le BG. Vous voyez :

"ABCxEFG

Vous avez remplacé Le D, dans la ligne de texte par un symbole appelé "l'homme complet". Si vous vous trompez en cours d'opération, faites simplement PACK.

Amusez-vous à essayer d'autres caractères.

Introduire un nul : Les nuls éventuellement nécessaires doivent être introduits en dernier, car on n'a plus le droit de faire PACK, donc plus le droit de se tromper. En partant de la situation ci-dessus, faisons GTO.001, BG.

Remarque : si, après un BG il n'y a aucun caractère à droite de la chaîne, c'est qu'il y avait des octets nuls qui traînaient, ne vous en faites pas, faites simplement BG plusieurs fois et laissez le nettoyage pour plus tard.

Dans ce cas, il peut se produire que le BG vole plus d'un octet.

Si cela vous gène, supprimez les BG en trop de la façon habituelle et faites PACK avant de recommencer, sinon, continuez comme si de rien n'était. Effacez LBL 00 et, sans faire PACK, BST jusqu'à la ligne qui contient le début de votre chaîne de caractères. Supprimez le BG, voyez :

"ABC-EFG

Un peu de ménage et un PACK ne feraient pas de mal, non ?

Remarque : avant l'exécution du BG, vous devez voir à l'écran la ligne qui précède l'instruction que vous voulez voler. Ces explications peuvent paraître complexes, mais à l'usage elles s'avèrent si commodes que j'ai toujours un BG assigné à une touche de ma 41C (note de la 4ème édition : j'ai maintenant remplacé le BG par une fonction microcode, CHARGE (cf. section 7.7.3 page 72), qui place directement en mémoire programme des instructions à partir de leurs codes décimaux, une version très améliorée de LB (ceci étant juste pour vous faire saliver !).

4094 : vous vous êtes sans doute demandés pourquoi on voit apparaître une ligne 4094 au cours de la création du BG. En fait, au cours d'un programme, la HP ne calcule pas les numéros de ligne de programme, elle n'en a pas besoin.

Mais si on arrête le programme pour le regarder, elle doit tout recalculer.

Pour se souvenir de la nécessité de ce calcul elle place dans le registre e le numéro de ligne FFF. Or FFF vaut 4095 en décimal. Dans le processus de création du BG, au moment où nous effaçons le LBL"T", la HP recule d'une ligne à partir de FFF sans s'apercevoir que ce numéro est invalide, d'où 4095-1= 4094.

5.2 Les postfixes artificiels

Le Byte Grabber peut vous servir à fabriquer en mémoire programme toutes les fonctions artificielles que vous voudrez. Il s'agit de fonctions à 2 octets,

Par exemple RCL, ISG, VIEW, FIX ou TONE. Les codes de préfixe de ces fonctions sont du rang 9. Nous pouvons construire en mémoire une chaîne d'octets qui mettra à la suite l'un de l'autre ce préfixe et un octet que nous finirons bien par transformer en postfixe. Nous allons essayer, par exemple, d'utiliser l'octet 75 comme postfixe. Nous allons résoudre le problème en plaçant en tête du train l'octet 90 ; nous aurons par exemple : 90 98 75

Que dit de cela la 41C ? 90 = RCL ; après RCL la 41C cherche un postfixe et prend donc le 98. Comme postfixe 98 est IND 24, reste un 75 tout seul qui est alors RDN et la HP écrit :

RCL IND 24

RDN

Êtes vous capable de créer ces deux lignes de programme ? Oui ? Alors vous pouvez tout faire. Avez-vous compris ? Nous allons voler (BG) le 90 et effacer purement et simplement la chaîne de caractères donnée par le BG. Restent les 2 derniers octets.

98 75

98 est lu VIEW

75 est alors un postfixe, il est affiché M

VIEW M

Sortez du mode programme et rentrez en ALPHA : ABCDEFGHIJ Pressez SST, ce qui exécute VIEW H, vous voyez (mode FIX 9) !

-4,4546474-50

C'est la copie des caractères DEFGHI, regardez les codes (et n'oubliez pas l'écriture spéciale des exposants négatifs).

Vous obtiendrez	avec
RCL M	RCL IND 16/ RDN
ISG M	RCL IND 22 / RDN
FIX M	RCL IND 28 / RDN

Mais là ! amusez vous à essayer plusieurs fois cette instruction... Il y a encore des choses à explorer...

TONE M	RCL IND 31 / RDN

Entendu ?

5.3 Tout assigner

Le programme ci-joint (fig, 5.2 page 39) est une variante d'un programme très courant dans le club PPC appelé traditionnellement KA (Key Assignement, assignement de touches), modifié pour être absolument sans risque de MEMORY LOST. Attention quand même, vous pouvez arrêter le programme, le SST, mais il faut impérativement aller jusqu'au bout de 2 assignements si vous voulez retrouver votre machine dans son état de départ.

Description du programme : Ce programme fabrique en alpha le contenu d'un registre d'assignement artificiel et place ce contenu dans R(0C0) qui a été précédemment occupé quand on a assigné les deux touches. Ces assignements préalables servent aussi à lever les index dans tes registres ⊢ ou e.

La construction de ce faux registre d'assignement est commencée ligne 05 où on place le caractère F0 en ALPHA.

A l'apparition de ??? entrez en décimal le préfixe de la fonction à assigner, ENTER, le postfixe, ENTER, le code de touche (ce même code que la 41C affiche quand on assigne : A = 11...) puis R/S. Le sous programme 02 convertit le préfixe en hexadécimal, puis le postfixe et place ces données en alpha, le code de la touche est ensuite mis en forme et placé en alpha. Puis un deuxième ??? demande le deuxième assignement. Il faut absolument répondre à cette 2ième demande, au besoin répéter simplement le premier assignement.

Ensuite la machine place le rideau en 00F (c'est l'effet de la ligne 12) le registre R(0C0) à remplir a alors l'adresse relative R177, d'où le recours au stockage indirect, puis on reconstitue l'état initial.

C'est cette partie du programme qui constitue son originalité. En effet, en général on se contente de placer le rideau en R(0C0), un simple STO 00 suffit alors au remplissage du registre. Mais si, à cet instant, on interrompt le programme ou on SST, il y a MEMORY LOST ! En effet, la 41C teste l'existence d'un registre sous le rideau : il y a toujours au moins un registre de programme à cette place, celui du .END..

Si le rideau est en R(0C0) il n'y a rien (sauf sur la CX, ou avec le module X-F) au dessous et la 41C réagit brutalement.

On peut recommencer indéfiniment l'opération et donc entrer autant d'assignements que désirés.

Réalisation du programme

Le BG va nous servir. La ligne 08 X<> M (l'imprimante fait des siennes, vérifiez sur la table pour MNOPQ⊢ la traduction par l'imprimante) s'obtient avec RCL IND 78 / RDN.

La ligne 09 est RCL IND16 / SDEV. La ligne11 (F5 01 69 00 F0 F0) est beaucoup plus difficile à obtenir (la ligne 05 Fl F0 est sur ce même modèle). Au fait, c'est un contenu pour le registre c, avez-vous vu le 169 ? Entrez ALPHA ABCDE ALPHA, BG le F5, reste - * / X<Y ? X>Y ?. Remplacez le - par LBL 00, remplacez le * par FRC, laissez le /, effacez les deux tests et faites *deux fois* "taper RCL IND.T et BG le RCL, reste F0". Faire PACK (pas GTO.., XEQ"PACK"), BST jusqu'au / et l'effacer. C'est fini, ne pas packer avant d'avoir supprimé le BG. Vous voyez :

"xX-XX (impossible à taper !)

Si à l'exécution, vous avez un MEMORY LOST en SST après cette ligne, c'est sans doute qu'il y a une erreur dans la ligne, refaites la. Mais cela ne devrait pas être.

Attention aussi à la ligne 05 qui est Fl F0, et non pas F0 tout seul, comme le listage pourrait le laisser croire.

La ligne 12 est ASTO c (RCL IND 26 / SDEV), la ligne 18 est STO c (RCL IND 17 / SDEV), la ligne 30 E 1 est un 1 E 1 dont on a volé le premier 1 ; l'essayer, c'est l'adopter (gain d'un octet et de temps).

La ligne 33 est STO 0 (RCL IND 17 / CLX), la ligne 53 est l'entrée d'un zéro, la ligne 54 X<>0 est RCL IND 78 / CLX, la ligne 64 est F5 7F 00 00 00 00. Pour cela, tapez "⊢ABCD", BG le F5, sauter le CLD et effacez - * et X<Y ? ; ne pas packer, supprimer le BG. Restent les lignes 70 et 83. X<> d est RCL IND 78 / AVIEW.

La ligne 87, x<> N est RCL IND 78 / LASTX, la ligne 88, STO M est RCL IND 17 / RDN.

5.4 L'artillerie lourde : LB

Quand il y a de nombreuses lignes de programme artificielles à créer, le BG devient fastidieux. De plus, il n'est pas possible d'obtenir ainsi certains codes des rangs C à E de la table. le programme ci-dessous, écrit par Lionel Ancelet, et publié dans le journal de PPC-Toulouse, crée directement en mémoire tous les octets qui peuvent être nécessaires. C'est une version d'un programme très répandu dans le club avec un nom classique LB (Load Byte = charge un octet) (cf. fig. 5.4 page 41).

Utilisation de LB :

Après avoir introduit LB en mémoire, faire GTO.. ; entrer en programme LBL "++" suivi d"un nombre de + supérieur de 12 au nombre d'octets artificiels à introduire, puis de XEQ "LB", END. Ne pas utiliser d'autres instructions. En mettre (des +) plutôt plus que moins, attention à ne pas avoir un END trop près sur lequel vous viendriez empiéter en cours de route.

Revenir en mode calcul et XEQ "LB". Après quelques secondes, un nombre de la forme 1,mn est affiché, ce qui signifie que l'on vous demande le premier octet d'un programme qui pourra en contenir nnn. Si vous n'avez pas rentré assez de +, une chaîne alpha ou un nombre est affiché.

Donnez alors la valeur décimale de l'octet à introduire, R/S , et ainsi de suite.

```
 01*LBL "KA"
 02*LBL 01
 03 "ASSIGNE
Z 2 †"
 04 PROMPT
 05 ""
 06 XEQ 02
 07 XEQ 02
 08 X<> [
 09 RCL c
 10 X<>Y
 11 "=1*"
 12 ASTO c
 13 177
 14 X<>Y
 15 STO IND
Y
 16 R↑
 17 R↑
 18 STO c
 19 CLST
 20 GTO 01
 21*LBL 02
 22 ASTO L
 23 "???"
 24 PROMPT
 25 CLA
 26 ARCL L
 27 X<> Z
 28 XEQ 04
 29 XEQ 04
 30  E1
 31 ST/ Y
 32 X<>Y
 33 STO J
 34 ABS
 35 INT
 36 LASTX
 37 FRC
 38 .1
 39 "
 40 ST* Z
 41 X=0?
 42 GTO 03
 43 RDN
 44 4
 45 X=Y?
 46 ISG Z
 47*LBL 03
 48 RDN
 49 X<>Y
 50 16
 51 *
 52 +
 53 ,
 54 X<> J
 55 SIGN
 56 8
 57 *
 58 X>0?
 59 CLX
 60 -
 61*LBL 04
 62 INT
 63 x=0?
 64 "+****"
 65 OCT
 66  E3
 67 /
 68  E1
 69 +
 70 X<> d
 71 FS?C 19
 72 SF 20
 73 FS?C 18
 74 SF 19
 75 FS?C 17
 76 SF 18
 77 FS?C 15
 78 SF 17
 79 FS?C 14
 80 SF 16
 81 CF 07
 82 SF 03
 83 X<> d
 84 ARCL X
 85 "+***"
 86 ,
 87 X<> \
 88 STO [
 89 RDN
 90 RDN
 91 END
```

Fig. 5.2 – Premier KA

Fig. 5.3 – Commentaires sur le programme KA

La ligne 03 est 03"ASSIGNEZ 2 T" pour "assignez 2 touches". vous devez alors assigner une fonction quelconque aux deux touches que vous comptez utiliser (assigner) avec le programme. Ceci a pour résultat de lever les index de touche dans le registre e (ou ⊢) et de placer un remplissage dans le registre 0C0, remplissage que KA va remplacer par les assignements définitifs.
Attention également : si, ayant des touches assignées auparavant, vous avez désassigné certaines d'entre elles avant d'utiliser KA le programme a de grandes chances de ne pas marcher. Il vaut mieux, au moins au début, utiliser KA sur la machine sans autre assignement que ceux réclamés par KA.
Deux assignements intéressants : le BG et eG0BEEP.
Le BG que nous avons créé place dans le registre d'assignement les octets F7 et 3F (décimaux 247/63), mais essayez d'assigner F7/80 (247/128). A même XROM, même affichage, ici avec le lecteur de cartes vous avez apparition de CARD READER (ou CARD RDR lx), une identification commode (F7/80=XROM 30,00).
eGOBEEP est l'assignement xx/167, xx étant compris entre 0 et F (0 à 15 décimal). C'est une "fonction imprévue" comme le Byte Grabber. Cette fonction a la propriété de créer (et éventuellement d'exécuter) les fonctions XROM 28 et 29 qui sont les fonctions de l'HP-IL, mémoire de masse et contrôle, et des imprimantes.
Cette fonction affiche eGOBEEP–. si vous répondez par un chiffre de 0 à 41 vous créez un XROM 28, chiffre qui correspond à l'HP-IL. Par exemple, le nombre12 donne XROM 28,12 qui est la fonction RENAME. si l'IL est présente ainsi que le lecteur de cassettes, cette fonction s'exécute (ou s'inscrit RENAME dans un programme) sinon on a NONEXISTENT (ou XROM 28,12 en programme).
Entre 42 et 63 il n'y a pas de fonction pour répondre. De 64 à 89 on obtient "XROM 29,(chiffre-64)", en quelque sorte il faut faire une retenue, et on a les fonctions de l'imprimante.
Attention, 89 est FMT (Format) et n'existe que sur l'HP-IL ; de plus, on peut ainsi "exécuter" les titres MASS STORAGE ou PRINTER. Avec des fortunes diverses, généralement mauvaises.

Si vous ne voulez plus rentrer d'octet, faites R/S sans rien introduire, le programme entre en phase terminale. Quand il s'arrête, un SST vous ramène au LBL"++", il ne vous reste plus qu'à vérifier votre programme et à supprimer les instructions inutiles.

Il existe une procédure de correction. Si vous vous êtes trompé, faites XEQ 01 au lieu de R/S et repartez au numéro de l'octet demandé, 1 pas ou 1 registre de 7 octets en arrière selon le cas.

Description de LB :

Quand on fait XEQ"LB", la machine n'est pas en mode alpha, donc le flag 48 est baissé, donc l'exécution saute au pas 04. AON lève le flag 48, CLST E RDN efface la pile et place 1 dans le registre T. L'exécution est ensuite transférée au LBL"++" ; la suite des "+" assure le comptage des octets, et on arrive au pas XEQ"LB" : celui-ci pousse dans le registre b l'adresse de retour du XEQ sous la forme compactée, et l'exécution est rendue au programme LB.

Comme cette fois le flag 48 est levé, l'exécution saute au LBL 00. AOFF rebaisse le flag 48, les instructions 11 à 39 calculent le nombre d'octets disponibles, les instructions 40 à 50 décodent l'adresse de retour, qui est celle du END de "++", et s'en servent pour construire un registre c provisoire, dans lequel cette adresse va être celle d'un registre R00 provisoire ; les octets synthétiques seront groupés par paquets de 7, et stockés registre par registre.

La boucle principale du programme est située aux lignes 73 à 130, le registre R00 contient le pointeur d'octet ; son contenu est affiché à la ligne 76, quand le programme s'arrête pour demander un octet. si on a pressé R/S sans rentrer d'octet, la valeur 0 est utilisée (lignes 77 et 78). les lignes 79 et 80 assurent que le nombre entré est dans l'intervalle O-255, et les lignes 81 à 102 construisent en alpha le caractère correspondant au code décimal entré (pour ceux qui ont le module X-F, ou la 41CX, ces lignes peuvent être remplacées par XTOA).

Ensuite, le programme regarde si on a fini de constituer un registre de 7 octets. Si c'est le cas, l'adresse relative au registre R00 provisoire est calculée aux lignes 109 à 120, puis le contenu de c est modifié pour placer R00 au bon endroit (lignes 121 et 122). Les lignes 123 et 124 stockent le registre constitué des 7 derniers octets entrés à l'adresse calculée précédemment, puis le contenu de c est restauré à sa valeur initiale. Alpha est alors effacé, et, s'il est encore possible de rentrer des octets, on continue.

Les pas 55 à 72 contiennent la procédure de correction. Le registre R00 est décrémenté, et le dernier caractère du registre alpha est effacé. Sachez enfin que le code hexa de la ligne 45 est : F4 10 00 0l 69.

Pas de difficulté particulière pour créer cette ligne si vous tenez compte du fait que le code de 10 étant celui du chiffre 0, il faut packer le programme après introduction du 0 (et avant effacement destiné à produire le nul), pour enlever le nul que la HP place avant tout chiffre en mémoire.

Donc taper "ABCD", BG le F4, entrer 0, effacer - et faire PACK, sauter *, entrer LBL 00 et FRC, effacer le * et supprimer le BG.

Vous devez maintenant savoir faire tous les autres codes.

Fig. 5.4 – Programme LB

```
01◆LBL "LB"
02 FS? 48
03 GTO 00
04 AON
05 CLST
06  E
07 RDN
08 GTO "++"
09◆LBL 00
10 AOFF
11 RCL b
12 STO [
13 "⊢◆◆◆"
14 RCL [
15 X<> d
16 CF 16
17 FS?C 07
18 SF 16
19 FS?C 06
20 SF 07
21 FS?C 05
22 SF 06
23 FS?C 04
24 SF 05
25 X<> d
26 FRC
27 STO [
28 LASTX
29 INT
30 R↑
31 +
32  E
33 -
34 STO 00
35 7
36 MOD
37 ST- 00
38  E3
39 ST/ 00
40 RCL [
41 X<> d
42 FS? 12
43 SF 03
44 X<> d
45 "0◆×i"
46 X<> [
47 STO \
48 "⊢◆◆◆"
49 RCL \
50 STO 01
51 FIX 3
52 SF 22
53 CLA
54 GTO 00
55◆LBL 01
56 SF 22
57  E
58 RCL 00
59 INT
60 X=Y?
61 GTO 02
62 7
63 MOD
64 X<>Y
65 X=Y?
66 LASTX
67 ST- 00
68  E1
69 ARCL X
70 ,
71 X<> \
72 STO [
73◆LBL 02
74 RCL 00
75 FS?C 22
76 STOP
77 FC? 22
78 ,
79 256
80 MOD
81 LASTX
82 +
83 OCT
84 X<> d
85 FS?C 11
86 SF 12
87 FS?C 10
88 SF 11
89 FS?C 09
90 SF 10
91 FS? 07
92 SF 09
93 FS? 06
94 SF 08
95 X<> d
96 X<> [
97 "⊢◆"
98 STO \
99 "⊢◆"
100 RCL \
101 CLA
102 STO [
103 RCL 00
104 INT
105 7
106 MOD
107 X≠0?
108 GTO 00
109 RCL 00
110 FRC
111  E3
112 *
113 RCL 00
114 INT
115 -
116 7
117 /
118 INT
119  E
120 +
121 RCL 01
122 X<> c
123 RCL [
124 STO IND
Z
125 X<>Y
126 STO c
127 CLA
128◆LBL 00
129 ISG 00
130 GTO 02
131 .END.
```

La séquence LBL"++" + + + ... XEQ"LB" est laborieuse à introduire. Pensez à la copier sur carte magnétique ; il suffit alors de travailler sur le dernier programme (celui dont le END est le .END.) et de placer la séquence en faisant XEQ"MRG". Vous découvrirez ainsi peut-être cette fonction si utile du lecteur de cartes.

L'entrée des GTO à 3 octets se fait pour LB en rentrant les valeurs décimales 208/0/ 0 à 127 (numéro du label).

N'essayez pas d'assigner ces valeurs, sinon, il vous faudra lire le prochain paragraphe !

5.5 En cas de malheur

Il peut arriver, plus souvent que vous ne l'espérez, que, à la suite d'une manipulation parfois anodine, la HP ne réagisse plus ou réagisse de façon anormale aux sollicitations.

Deux façons de réagir : RESET et MEMORY LOST

RESET est une fonction du microprocesseur prévue d'origine par HP pour être accessible à l'utilisateur.

Sur un modèle ancien, il faut retirer les piles et les remettre (attention à avoir débranché l'imprimante, si c'est une 83143 cette-ci alimente la HP - au fait, à part l'affichage, la HP fonctionne très bien sans pile quand cette imprimante est branchée).

Sur un modèle récent, il faut presser <- et donner une ou 2 pressions sur ON. Si on voit CLX, c'est débloqué.

MEMORY LOST s'obtient à partir d'une machine éteinte en pressant <- puis en effectuant une pression sur 0N et en lâchant tout.

Dans les cas graves, il faut laisser sans pile la HP plusieurs jours (enlever au préalable tous les accessoires). Généralement, les alarmes sont perdues, le réglage de l'horloge aussi.

Chapitre 6

Microcode

Nous voici maintenant dans les grandes profondeurs, comme déjà expliqué, vous sortirez sans doute frustrés de la lecture de ce chapitre, car les conditions à réunir pour utiliser le microcode ne sont pas immédiatement accessibles.

N'oubliez pas que Hewlett-Packard ne vous apportera aucune aide ni aucun matériel. Tout ce qui suit est l'oeuvre du club et n'est pris en compte -et encore !- que par le club PPC.

A y regarder de près, ces conditions ne sont pourtant pas si draconiennes.

Tous les accessoires indispensables sont en vente en France , et les performances sont au niveau des difficultés.

6.1 Comment s'en servir

Il peut paraître paradoxal de discuter du comment s'en servir avant d'avoir vu de quoi, mais ce point est très important ici.

Disons tout de suite qu'il est matériellement impossible de programmer en microcode dans la mémoire vive de la 41C.

Deux types d'appareils permettent cependant cette programmation :
— le lecteur d'EPROM
— le simulateur de ROM
Le premier type d'appareil (EPROM Box en anglais) est disponible dans le commerce sous de nombreuses formes, en France et à l'étranger. Son prix est un peu supérieur à celui du lecteur de cartes magnétiques, sa taille du même ordre que celle de la 41C à laquelle il est relié par un cordon identique à celui de l'imprimante (c'est la seule pièce d'origine HP). Un modèle récent est même identique d'aspect au lecteur de cartes.

Il permet seulement la lecture d'instructions préalablement enregistrées dans des EPROM placées dans le lecteur (et facilement interchangeables). Ces instructions doivent donc être enregistrées autrement.

Les mémoires utilisées sont des circuits intégrés courants du commerce, appelés EPROM. Ces mémoires peuvent être effacées à l'aide de rayons ultraviolets.

Elles sont remplies à l'aide d'un appareil spécialisé appelé programmeur d'EPROM. Il existe des versions de programmeur d'EPROM très bon marché (quelques centaines de francs), en général à fabriquer par l'amateur, mais d'un usage très laborieux. L'appareil professionnel est très coûteux et doit être branché à un ordinateur. Il existe quelques modèles intermédiaires d'un prix accessible à l'amateur, mais de toute façon son usage est peu fréquent, et donc son achat peu raisonnable. L'amateur a plutôt intérêt à se tourner vers un club ou un professionnel. Certains adhérents de PPC Toulouse, par exemple, possèdent un programmeur d'EPROM et peuvent faire le travail pour vous.

Ces EPROM une fois remplies, sont placées dans le lecteur d'EPROM. Celui-ci fonctionne alors exactement comme un module Math, PPC-ROM, X-fonction habituel.

On peut placer dans une EPROM des programmes dans notre langage habituel. Ces programmes deviennent alors disponibles à volonté et en permanence et laissent libres les 319 registres de la 41CV. On peut également y placer des programmes en microcode, qui s'utilisent alors exactement comme les fonctions standard LOG, SDEV ou +.

On peut placer dans un lecteur d'EPROM jusqu'à 32K octets (8 modules) selon le modèle... Je vous laisse rêver...

Le deuxième type d'appareil est maintenant disponible en France, ainsi que dans des modèles différents en Australie, aux Pays Bas et bien sûr aux USA. Il comporte dans la version française à la fois un lecteur d'EPROM de 4K et une mémoire vive de même taille.

Il s'agit d'une mémoire vive, et donc dans laquelle on peut écrire directement à partir de la 41C et qui, à la lecture, est prise pour un ROM.

Donc, à la lecture, même fonctionnement que le lecteur d'EPROM. A l'écriture, fonctionnement voisin de celui auquel vous êtes habitués. C'est donc un système très intéressant. Il a cependant des inconvénients, dont le plus important est d'être plus volumineux qu'un simple lecteur, au point de ne pas être vraiment transportable, surtout s'il est fabriqué par l'amateur. Ensuite, il ne dispose que de 4K de mémoire (1 module), ne permet pas d'écrire dans des EPROM et court les risques d'une mémoire vive (l'effacement, en cas de panne de l'alimentation un peu trop prolongée). Il est cependant possible d'enregistrer sur cartes magnétiques certains programmes courts (et d'enregistrer n'importe quoi sur cassette, même un module HP complet !), pour les réutiliser ultérieurement ; il est même possible de faire transférer ces cartes ou cassettes sur EPROM. C'est l'appareil idéal pour le développement de programmes en microcode. La meilleure configuration est un simulateur pour la mise au point des programmes et un lecteur d'EPROM pour l'utilisation. Une solution coûteuse, mais professionnellement très performante.

6.2 Le microcode, qu'est-ce que c'est ?

Vous connaissez les poupées russes qui contiennent une autre poupée, qui contient une autre... votre HP est un peu ainsi.

Quand vous utilisez le programme LB, par exempte, vous dialoguez non pas avec la HP mais avec ce programme. Les messages ont été prévus non pas par la 41C mais par le programmeur. Il faut vous y faire, on dialogue toujours avec un programme, jamais avec une. machine.

Si vous faites CLX LN ; vous voyez DATA ERROR. Vous êtes en train de dialoguer avec un programme. Quand vous utilisez le module de math ou le PPC-ROM vous appelez des noms de programme et vous en attendez un résultat.

Quand vous utilisez les fonctions LN, + ou ENTER, vous faites la même chose.

Les programmes en microcode eux-même utilisent des instructions analogues à LN ou +. La différence c'est que ces instructions sont comprises par la 41C grâce à une disposition particulière des transistors qui réagissent à des stimulations électriques. Les différents niveaux sont donc les suivants :

— le programme "câblé" du microprocesseur de la 41C, qui, recevant des impulsions électriques sur certains fils, réagit.
— une définition codée de ces impulsions électriques qui forme ce que l'on appelle le microcode et que nous allons étudier.
— des modules programmés en microcode qui réagissent aux pressions de touches et forment les fonctions que nous utilisons tous les jours.
— les programmes que nous faisons habituellement qui sont simplement un chaînage organisé par nous de ces modules.

6.3 Le principe de fonctionnement

Notre machine, le croiriez-vous, est une marmotte. Elle dort chaque fois qu'elle a fini une fonction, c'est ce que l'on appelle le "sommeil léger".

Faites 2 ENTER 3 +, résultat 5. Quelques micro-secondes plus tard le microprocesseur de la 41C est arrêté. Seul l'écran reste éveillé, et encore pas pour longtemps, au bout de quelques minutes l'écran aussi s'éteint, c'est le "sommeil profond".

Mais la moindre pression de touche réveille le monstre ! La 41C entame alors une course poursuite pour mettre en ordre toute la mémoire avant de se précipiter au clavier pour voir ce qu'on lui demande.

Selon la touche pressée, une suite est attendue ou une fonction s'exécute. Quand le travail est fait, un nouveau nettoyage a lieu et la 41C se rendort.

Ce qui doit être remarqué c'est que le microcode ne connaît pas la fonction LN (par exemple), mais il constate que la touche pressée est la cinquième du premier rang, que le flag user n'est pas levé, qu'aucun programme n'est en cours d'exécution. Dans ce cas, il continue l'exécution en un point particulier de la mémoire, puis au dodo.

Si c'est un programme qui s'exécute, il y a fonctionnement continu, avec cependant un nettoyage entre chaque fonction.

Ce nettoyage de précaution prend une partie importante du temps d'exécution de la fonction. C'est l'économie de ce temps qui fait, entre autre, la rapidité d'exécution du microcode (jusqu'à 250 fois plus. vite...).

6.4 Géographie du microprocesseur

Le microprocesseur (CPU en anglais : Central Processing unit = unité centrale de traitement) possède pour ses propres besoins des registres de mémoire particuliers. J'utilise dans le texte les dénominations choisies par les pionniers américains du club PPC qui ont déchiffré tout cela (essentiellement Steve Jacobs), dans les tableaux les dénominations (mnémoniques) Hewlett-Packard sont également donnés (cf. fig. 6.1 page suivante et 6.2 page 47).

FIG. 6.1 – Schéma bloc de la HP-41

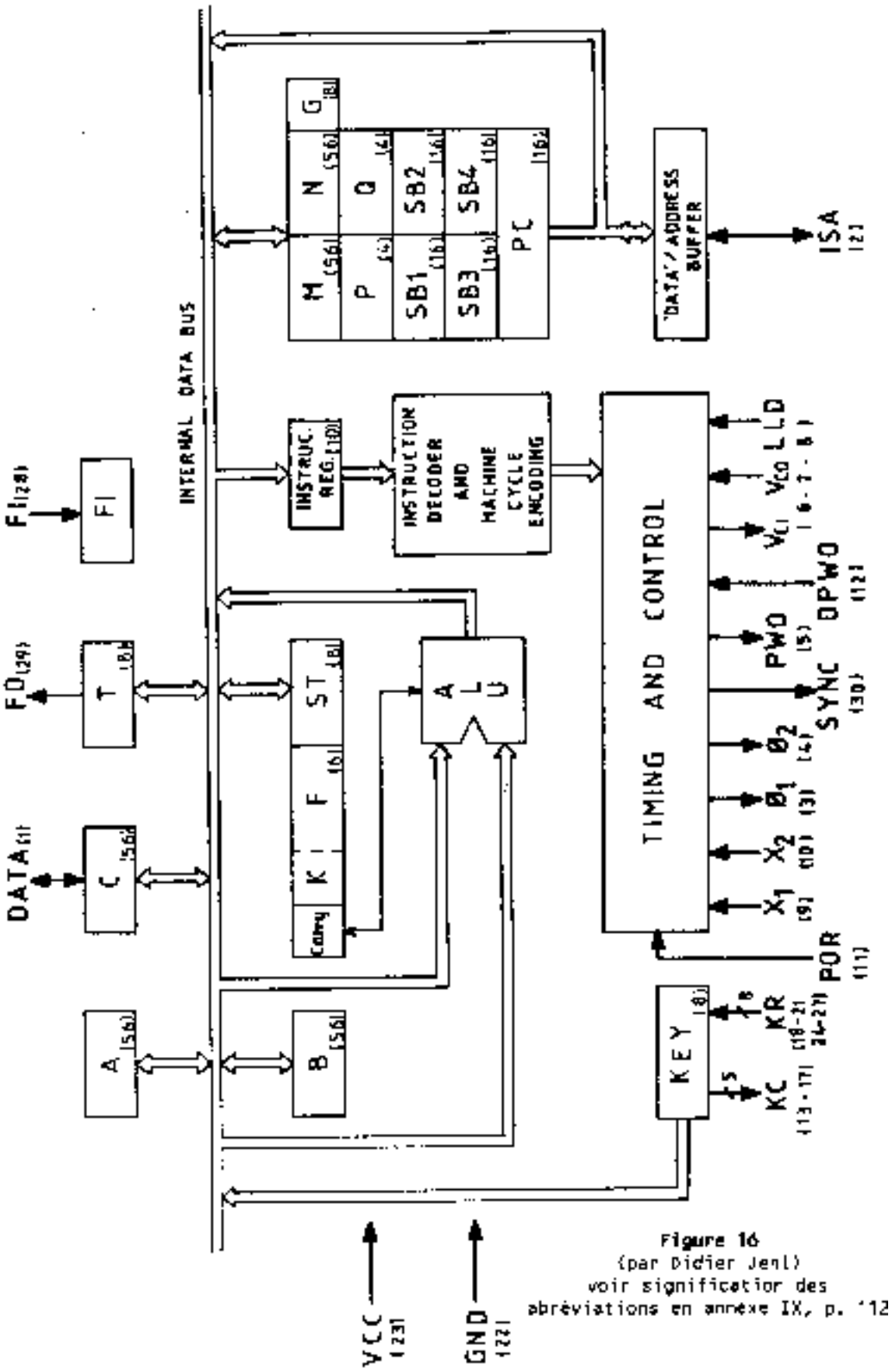


Figure 16
(par Didier Jenl)
voir signification des abréviations en annexe IX, p. 112

6.4.1 Le registre C

Ce registre de 56 bits est l'élément essentiel de la mémoire du microprocesseur, le point de passage obligé de toutes les entrées et de toutes les sorties. En plus fort, il joue un peu te rôle que joue usuellement le registre X, la plupart des opérations arithmétiques commencent et finissent en C. C'est en quelque sorte l'accumulateur principal.

6.4.2 Les registres A et B

Ce sont également de grands registres de 56 bits qui travaillent en relation avec C ou entre eux. Un peu l'équivalent de Y et Z dans la pile.

Attention ! il n'y a pas de mouvement de pile opérationnelle en microcode. En fait, il n'y a aucun mouvement automatique de registre (sauf SBR) ; tout doit être prévu par le programmeur.

Les registres A, B et C font 56 bits, comme n'importe quel registre habituel de la 41C. Ici aussi, on peut distinguer dans A, B et C des zones MS, XS et XP.

Mais en plus, on distingue deux zones supplémentaires : les digits 3 et 4 appelés zone KY (pour Key, c'est à dire "touche") et une zone formée des digits 3 à 6 appelé ADR (Adresse). Nous verrons l'usage de ces zones en même temps que les instructions en microcode (cf, fig. 6.3 page 48) sans oublier la zone formée de XP+XS et appelée S&X (Signe et exposant).

6.4.3 Les registres M et N

Encore deux grands registres, mais aucune arithmétique n'est possible avec eux, seulement une mise en réserve des données (STO) à partir de C, leur rappel (RCL) ou l'échange de leur contenu avec celui de C, et cela pour la totalité du registre.

6.4.4 Le registre G

Ce registre permet de stocker un seul octet choisi à volonté dans C, de le rappeler ou de l'échanger.

6.4.5 Les registres P et R

Registres très particuliers, ce sont des pointeurs. Ceci signifie que l'on peut désigner un digit particulier de A, B et C par un chiffre placé en P ou Q. P et Q comptent chacun 4 bits, ils peuvent donc contenir un chiffre hexadécimal de 0 à F.

Les 56 bits de A, B et C sont divisés en 14 digits de 4 bits (4 x 14 = 56), comptés de 0 à 13, de droite à gauche, comme d'habitude.

La 41C ne peut travailler à la fois avec P et Q, elle doit d'abord choisir P ou Q. Le pointeur choisi est alors désigné par la lettre R (on peut donc avoir R = P ou R = Q).

Le contenu de R peut être utilisé pour désigner un digit particulier de A, B ou C, par exemple, dans une instruction signifiant : "est-ce que le digit dont le numéro est en R est nul ?"

Mais P et Q peuvent aussi servir de compteur, un peu comme le registre i de certaines HP. on peut ajouter ou retrancher 1 au registre sélectionné R et repérer une valeur particulière de R (si R = 5 faire ceci, sinon...).

6.4.6 Le registre PC et les registres SBR

Nous avons vu (4.6 page 29) que la 41C possède un pointeur de programme et 6 registres de retour de sous programme.

Le microprocesseur, pour sa part, a bien sûr un registre (PC = Program Counter) dans lequel se trouve le numéro de ligne de ROM microcode en cours d'exécution, mais il n'a que 4 registres pour les adresses de retour de sous-programmes.

Chacun de ces registres fait 16 bits et peut donc compter jusqu'à 65536.

FIG. 6.2 – *Registres du CPU*

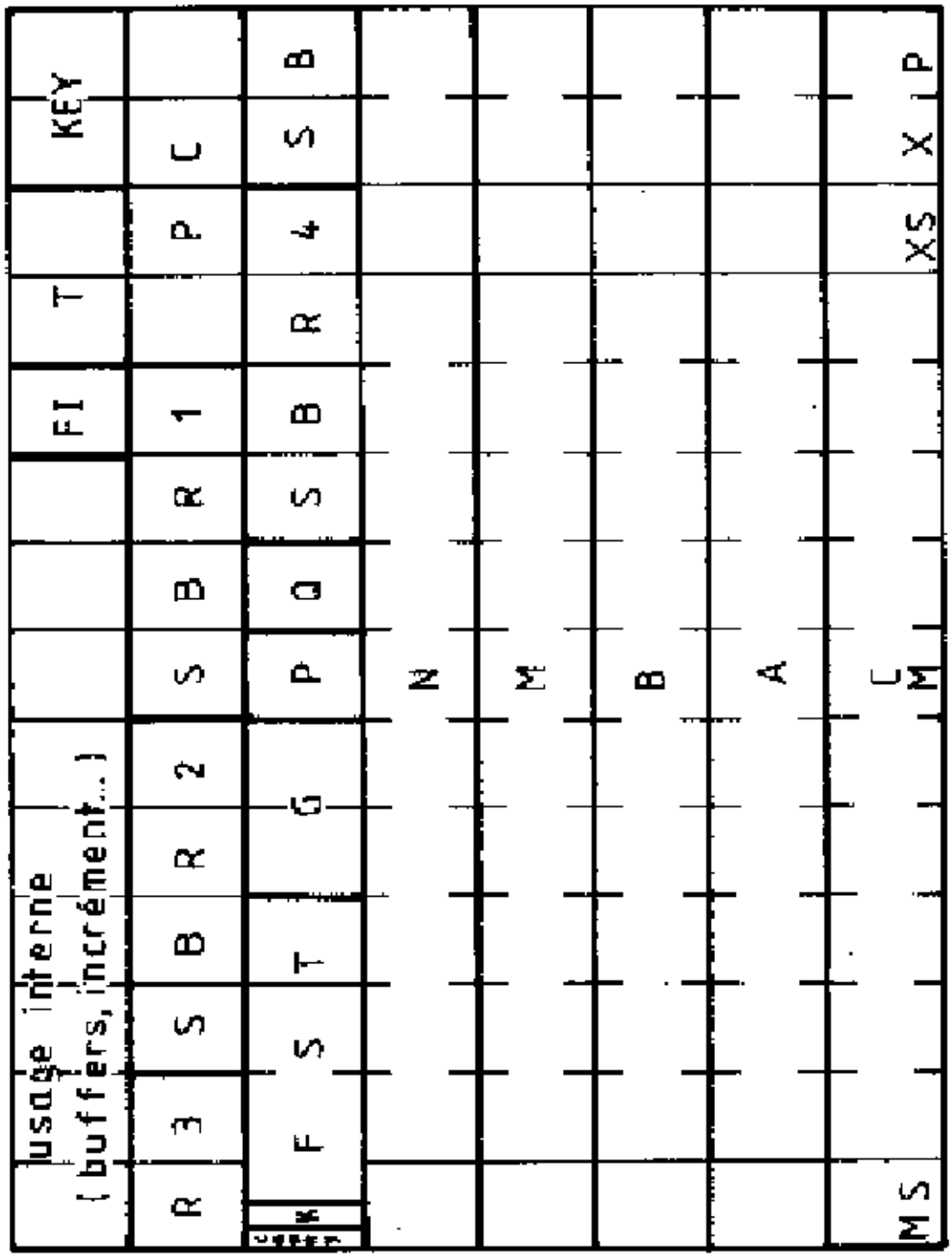


FIG. 6.3 – Suivi du registre microcode

Utilisez des photocopies de cette page pour suivre le contenu du registre C au cours de l'étude de programme microcode.

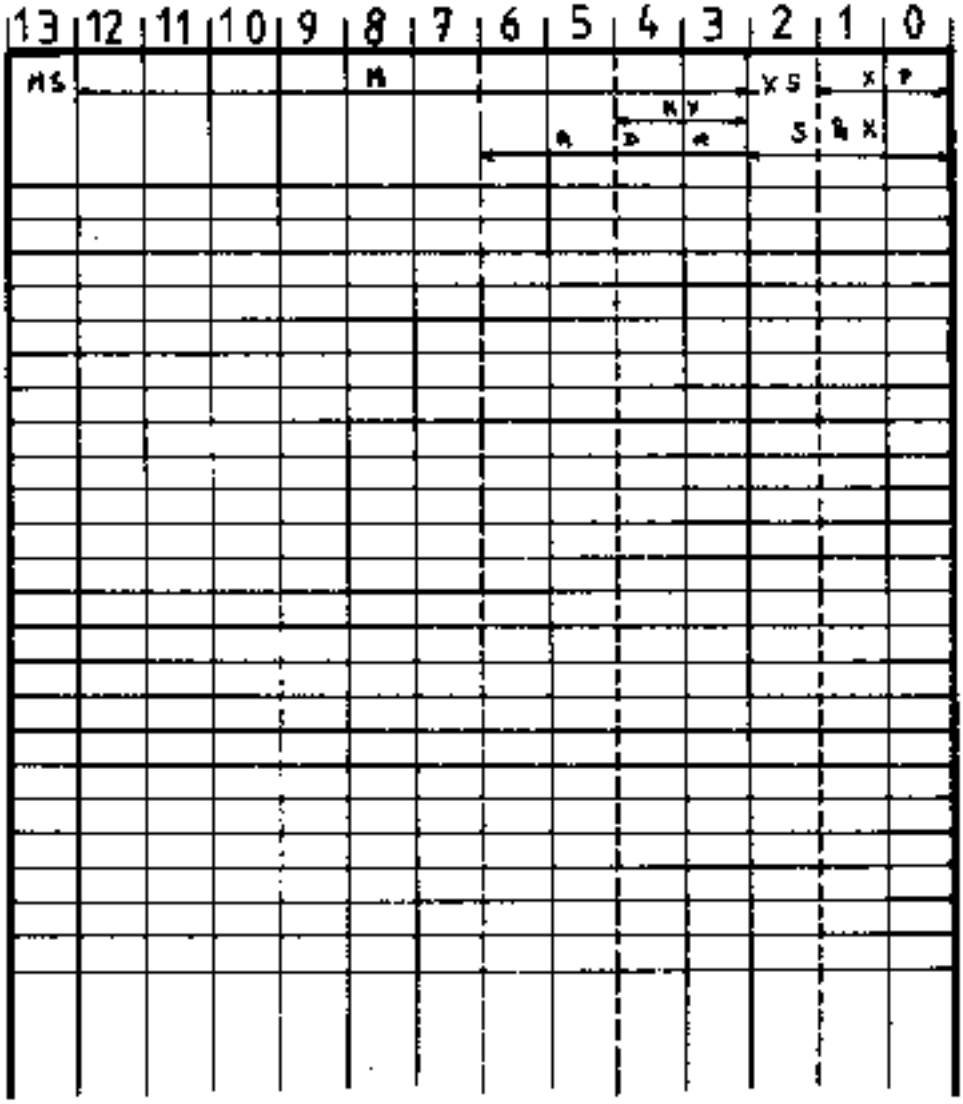
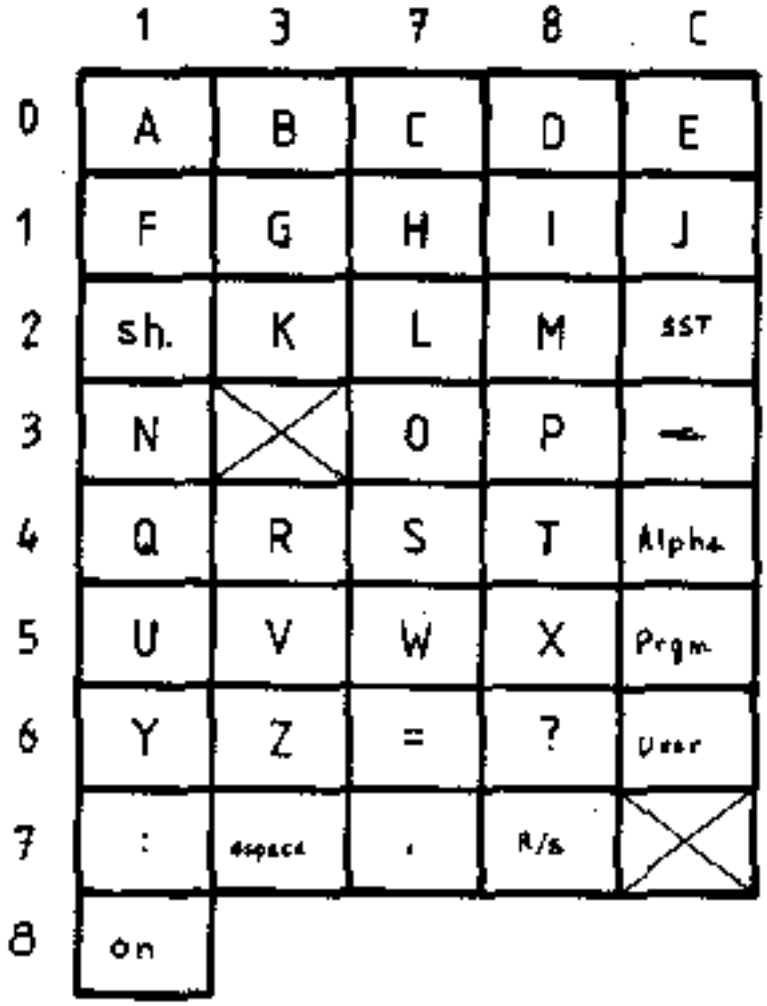


Fig. 6.4 – Valeurs trouvée dans KEY à la pression d'une touche colonne-rang <-=C3



6.4.7 Le registre ST

Appelé registre d'état (status, en anglais), c'est simplement un registre de flags de 14 bits, donc 14 flags.

Mais, de même que la X-fonction X<>F permet un échange entre X et une partie du registre des flags, il est possible d'échanger les 8 bits de droite (0 à 7, de droite à gauche) de ST et les 8 bits de droite (XP) de C. Sinon on peut lever, baisser ou tester les flags comme d'habitude. Cet échange avec C sert à tester les flags utilisateurs du registre d.

Les 6 flags 8 à 13 ont, dans le microcode des registres internes, un usage précis qui est le suivant :

F13 un programme est en cours d'exécution

F12 le programme est privé

F11 la montée de la pile (X->Y->Z->T->) est autorisée

F10 le pointeur de programme est dans un ROM

F9 et F8 usage général

Le plus souvent la zone 0-7 de ST contient ce que HP appelle "le jeu 0 des flags", c'est à dire les flags 48 à 55 du registre d (aux places 7 à O) qui sont les flags de contrôle de l'état de la machine.

6.4.8 Le registre T

Ce registre semble avoir le comportement suivant. Quand on y place 00 le bip acoustique est silencieux. Si on y place FF une impulsion est envoyée au bipper. En effet, les "TONE" sont réalisés en alternant dans T 00 et FF.

La fréquence est obtenue en faisant varier la durée entre deux échanges et la durée en faisant varier le nombre d'échanges (voir 7.6 page 67 le détail du fonctionnement).

6.4.9 Le registre KEY

Quand une touche est pressée, un code est placé dans ce registre. La relation code-touche est donnée fig, 6.4 page précédente.

6.4.10 Le registre Carry

Carry en anglais informatique, c'est la "retenue" (15 plus 9, je pose 4 et je retiens 1...).

Ici, c'est un flag spécial qui est levé dans les conditions suivantes (Seule la zone concernée par l'opération est considérée, elle peut être un digit au pointeur, à XS ou NS, ou plusieurs à M, XP,...).

— une addition donne un résultat supérieur au chiffre formé en mettant à 1 tous les bits de la zone (1111 soit F s'il n'y a qu'un digit, 1111 1111, soit FF s'il s'agit de XP, etc).
— une soustraction donne un résultat inférieur à zéro.
— ce flag Carry est levé en réponse à un test (lever Carry si...).

Remarque : en microcode, la HP fait ses additions aussi bien en hexadécimal qu'en décimal, à la demande. Dans le cas du travail en décimal Carry est levé à l'addition quand le plus grand nombre décimal pouvant être écrit dans la zone est dépassé.

Disons enfin que le flag Carry est baissé par toutes les instructions qui ne le lèvent pas... Il ne peut donc être testé que par l'instruction suivant immédiatement celle qui l'a levé.

6.4.11 .....

D'autres registres existent, nécessités par le fonctionnement interne. Ils ne nous sont pas accessibles en tant que tels, il n'est donc pas nécessaire d'insister.

6.5 Les instructions du microcode

Depuis la découverte des instructions par les pionniers du club, HP a fourni quelques renseignements permettant de connaître les mnémoniques utilisés par eux. Il y a donc deux façons d'appeler les mêmes instructions. Ces deux façons sont données dans les tableaux. Les commentaires utilisent les mnémoniques PPC.

Nous avons déjà dit que les instructions du microcode sont codées sur 10 bits.

Les deux bits de droite (Numéros 0 et 1) de l'instruction donnent le TYPE de l'instruction. Il y a donc 4 types d'instructions 0, 1, 2, et 3.

Dans les tableaux le code est donné sous la forme 244. Dans les listages les deux modes de codages sont indiqués sous La forme 244-442.

6.5.1 Instructions de type 00

Ces instructions sont formées ainsi : (cf. fig.20)

paramètre (4 bits) instruction (4 bits) type (00)

Le paramètre sert à préciser sur quoi agit l'instruction, par exemple :

0010 0001 00

signifie : paramètre=2, instruction CLRF, type 0 et se traduit : CLear FLag 5 (baisser le flag 5).

Cette instruction peut être écrite (cf. Il.5, p. 17) :

mode 442 : code 210

mode 244 : code 084

Remarquer que les instructions numéro 6 (et 13), 8 et 12 forment avec leur paramètre des instructions distinctes. Elles sont données fig. 21, p. 74 et 22, p. 75.

Ces instructions 6/0 (identiques aux instructions13/0 sont celles de l'accès aux registres G, M, T et ST. l'octet concerné par le registre G est celui formé par le digit situé au pointeur et le digit de numéro immédiatement supérieur.

Les instructions 8/0 sont variées. Remarquons ici l'instruction XQ-GO qui fait baisser d'un cran la pile de retour des sous programmes, "oubliant" l'adresse de retour la plus proche et plaçant 0 dans SBR4. GTO ADR remplace le GTO IND habituel et envoie le programme à une adresse calculée et placée dans la zone ADR de C.

Il n'y a pas de DSP ON, il faut faire DSP OFF, DSPTOG.

Plusieurs instructions importantes dans le dernier tableau (C/0 fig. 22, p. 75) :

PUSH ADR place le contenu de la zone ADR de C dans la pile de retour des sous programmes (SBR1). Cette pile de retour se comporte automatiquement comme la pile XYZT habituelle lors d'un RCL. lors d'un PUSH, le contenu de la pile monte d'un cran pour faire de la place à la nouvelle adresse. Le contenu de SBR 4 est perdu.

POP ADR. Quand cette instruction est exécutée, le contenu du premier registre de la pile (SBR 1) est placé dans la zone ADR de C et la pile descend d'un cran. Contrairement à la pile opérationnelle XYZT, il n'y a pas duplication du 4ième registre qui est mis à zéro. Exemple :

avant	après
SBR 4 : 0123	SBR 4 : 0000
SBR 3 : 6742	SBR 3 : 0123
SBR 2 : 2340	SBR 2 : 6742
SBR 1 : 1572	SBR 1 : 2340
C ADR : 6001	C ADR : 1572

6001 est perdu

Remarques sur SBR : Cette pile est du type dernier entré-premier sorti. Lors de la rencontre d'une instruction XQ, le microprocesseur place dans la pile l'adresse de retour, c'est à dire l'adresse du 2ème mot du XQ + 1. Cette adresse est en d'autres termes l'adresse du mot qui suit l'XQ de départ. C'est ainsi qu'il est possible de "passer des paramètres" à un sous programme. Il suffit de placer une donnée (par exempte un nombre de boucles à effectuer, le code d'un caractère à afficher ou le code d'un message d'erreur) après l'appel du sous programme.

Cette donnée va être récupérée par l'instruction FETCH S8X (to fetch = aller chercher) utilisée après un POP. RAM SLCT permet de sélectionner un registre de mémoire vive. Cette sélection se faisant à partir d'une adresse à 3 digits, il semble possible de sélectionner FFF soit 4096 registres. En fait, à l'expérience, seuls les 10 bits de droite de l'adresse répondent, ce qui limite à 1024 te nombre de registres accessibles - justement le maxi possible avec les x-mémoires.

PRPH SLCT sélectionne le périphérique dont le numéro est indiqué en C(XP).

Nous sommes ici dans une zone "incertaine", il semble bien que le digit XS n'ait pas d'influence sur le PRPH choisi, mais..., ce n'est pas sûr. La mémoire vive est un périphérique spécial (cf. RAM SLCT). Elle doit être invalidée avant toute sélection d'un autre périphérique ou par sélection d'une adresse vide, en général 010 (parfois 2FD).

L'affichage est le périphérique FD

Le lecteur de cartes est le périphérique FC

Le lecteur optique est le périphérique FE

Le module temps est le périphérique FB

Les instructions WRITE et READ envoient ou reçoivent des données vers ou en provenance d'un périphérique. S'il s'agit de la mémoire vive le paramètre indique la position du registre lu dans le groupe de 16 registres qui a été sélectionné. S'il s'agit de l'affichage, se reporter à l'annexe X.

Les autres périphériques (IL, imprimante 82143) utilisent des instructions spéciales. En fait, la HP-41C reconnaît des "adresses" ou numéros de code qui désignent une ligne (fil électrique), qui aboutit dans le périphérique à un registre. Si le périphérique comprend plusieurs registres (module HP-IL), à chaque registre correspond une adresse.

Écriture des variables (module HP-IL)

nnn étant le numéro du registre HP-IL destinataire, la HP utilise l'instruction 1nnn 0000 00 (binaire) pour envoyer dans ce registre le contenu de C S&X.

Écrire dans le registre 2 donne donc 280-A00 (244-442).

Sélection (instruction SELP d)

Fig. 6.5 – Instructions de type 0

les instructions précédées de * ne doivent pas être placées immédiatement après une instruction arithmétique.

mot : P3 P2 Pl P0 : I3 I2 I1 I0 : 0 0
(Paramètre : Instruction : Type)

n	bits	HP	PPC	Remarques
0	000		NOP	Nn opérant - paramètre inutilisé (cf. p. 75)
1	0001	SD=0	*CLRF f	Clear flag f - effacer flag f (1)
2	0010	SD=1	*SETF f	Set flag f - lever flag f (2)
3	0011	?SD=1	* ?FSET f	Lever Carry si le flag f est levé
4	0100	LC	LD@R d	Mettre d en C au pointeur R (4)
5	0101	?PT	* ?R= f	Lever Carry si le pointeur est à f
6	0110		divers	voir table 6/0
7	0111	PT=D	R= f	Placer le pointeur sélectionné à f (6)
8	1000		divers	voir table 8/0
9	1001	SELPRF	SELP d	Sélectionner périphérique d (7)
10	1010	REGN=C	WRIT r	Ecrire C vers le périphérique r
11	1011	?FN=1	?FI f	Test des flages des périphériques
12	1100		divers	Voir table C/0
13	1101		divers	comme table 6/0
14	1110	C=REGN	READ r	Lire un périphérique en C (8)
15	1111	RCR	RCR f	Rotation de C de f digits vers la droite (9)

(1) Sauf 3C4 (CLRST, ST=0) effacer le registre ST
(2) Sauf 3C8 (RST KB, CLRKEY) efface le flag de clavier
(3) Sauf 3CC (CHK KB, ?KEY) lève Carry si le flag K est levé
(4) Enlève 1 à la valeur du pointeur après exécution
(5) Sauf 3D4 (DEC PT, R=R-1) décrémenter le pointeur
(6) Sauf 3DC (INC PT, R=R+1) incrémenter le pointeur
(7) le CPU est inactif pendant l'exécution des instructions spéciales
(8) READ 0(T) s'écrit aussi READ DATA (C=DATA)
(9) RCR 15= RESET affichage (immobilisé pendant 24 cycles)

Fig. 6.6 – Type 0, Définition des paramètres

n	bits	d(D)	f(N)	r
0	000	0	3	0 (T)
1	0001	1	4	1 (Z)
2	0010	2	5	2 (Y)
3	0011	3	A(10)	3 (X)
4	0100	4	8	4 (L)
5	0101	5	6	5 (M)
6	0110	6		6 (N)
7	0111	7	B(11)	7 (O)
8	1000	8	-	8 (P)
9	1001	9	2	9 (Q)
10	1010	A(10)	9	10 (+)
11	1011	B(11)	7	11 (a)
12	1100	C(12)	D(13)	12 (b)
13	1101	D(13)	1	13 (c)
14	1110	E(14)	C(12)	14 (d)
15	1111	F(15)	0	15 (e)

Le décodage de toutes ces instructions est l'oeuvre de Paul Lind, Jim De Arras et Steven Jacobs, tous membres de PPC.

Fig. 6.7 – Type0, sous classe 6/0 (et 13/0)

244	HP	PPC	Remarques
018		Inutilisé	
058	G=C	G=C @R,+	Met les digits R et R+1 de C dans G
098	C=G	C=G @R,+	Met G dans C R et R+1
0D8	CGEX	C<>G @R,+	Echange G et deux digits de C
118			Inutilisé
158	M=C	M=C ALL	Met C dans M
198	C=M	C=M ALL	Met M dans C
1D8	CMEX	C<>M ALL	Echange de C et M
258	F=SB	T=ST	Met en T le contenu de ST
298	SB=F	ST=T	Met en ST le contenu de T
2D8	FEXSB	ST<>T	Echange T et ST
318			Inutilisé
358	ST=C	ST=C XP	Met en ST l'exposant de C
398	C=ST	C=St XP	Met en C(XP) le registre ST
3D8	CSTEX	*C<>ST XP	Echange de C(XP) et de ST

Note : ST consiste en les flags 0–7. Les autres flags doivent être levés et baissés individuellement .

Cette instruction a la forme dddd 1001 00 (d90 en mode 442). Elle arrête le microprocesseur de la HP-41C et sélectionne la ligne ou registre d. Elle peut être suivie de :

Lecture

L'instruction dddd 1110 1x, où dddd est en binaire le numéro du registre, lit ce registre et place sa valeur dans C S&X. Si x est égal à 1, la main est rendue au processeur de la 41C qui continue le travail.

Écriture des constantes

L'instruction aaaa aaaa 0x écrit dans le registre préalablement sélectionné l'octet aaaa aaaa. Si x est à 1, la main est rendue au processeur de la 41.

Écrire E2 dans le registre 0 donne : 024-090 389-E21.

Test de flag

L'instruction dddd 0000 11 teste le flag dddd du périphérique sélectionné et arme Carry si le flag est levé (le Carry de la 41) ; la main est toujours rendue à la 41. Dans le module HP-IL, la procédure suivante est utilisée,

Exemple : lire le registre 3

0E4-390 SELP 3

CFA-3E2 lire R3, laisser le temps à l'IL de travailler

0C3-303 tester le flag 3 et continuer le programme 41

Ici on ne rends pas la main à la 41 dés la seconde instruction pour permettre au module IL de compléter son travail.

6.5.2 Instructions de type 1

Ces instructions comportent deux mots successifs dans un programme. Il s'agit des sauts à une adresse (GOTO) ou à un sous programme (XQ) donné par le numéro de sa première ligne (il n'y a pas de label en microcode).

Comme il y a 16 bits pour l'adresse, on peut sauter à n'importe quel point de la mémoire (cf. figure 6.12 page 55).

6.5.3 Instructions de type 2

Opérations arithmétiques et logiques, particulièrement fournies ici. Elles peuvent être exécutées en mode hexadécimal ou en mode décimal et ne sont exécutées que sur le champ spécifié (cf. figure 6.14 page 56).

Fig. 6.8 – Type 0, sous classe 8/0

244	HP	PPC	Remarques
020	SPOPND	XQ-GO	Transforme XQ en GO en POPant la pile SBR
060	POWOFF	POWOFF	Arrêt du CPU, pas de l'affichage[1]
0A0	SELP	SLCT P	P comme pointeur actif
0E0	SELQ	SLCT Q	Q comme pointeur actif
120	?P=Q	?P=Q	Lève Carry si P=Q
160		?LOWBAT	Lève Carry si batterie faible
1A0	CLRABC	A=B=C=0	Efface A, B et C
1E0	GOTOC	GOTO ADR	Met dans PC les digits ADR de C
220	C=KEYS	C=KEY KY	Met en KY de C le contenu de KEY
260	SETHEX	SETHEX	Place le CPU en mode Hexa
2A0	SETDEC	SETDEC	Place le CPU en mode Décimal
2E0	DISOFF	DSPOFF	Eteint l'écran
320	DISTOG	DSPTOG	Inverse l'état de l'écran (on<>off)
360	RTN C	?C RTN	Retour si Carry levé
3A0	RTN NC	?NC RTN	Retour si Carry baissé
3E0	RTN	RTN	Retour impératif

Note 1 : POWOFF est une instruction à 2 octets, le deuxième étant 000.

Fig. 6.9 – type 0 sous classe C/0

244	HP	PPC	Remarques
030			Inutilisé
070	N=C	N=C ALL	Met C dans N
0B0	C=N	C=N ALL	Met N dans C
0F0	CN EX	C<>N ALL	Echange C et N
130	LDI	LDI S&X	Met dans C S&X le mot immédiatement suivant
170	STK=C	PUSH ADR	Met ADR de C dans SBR 1
180	C=STK	POP ADR	Met SBR 1 en ADR de C
1F0			Inutilisé ?
230	GOKEYS	GOTO KEY	Met dans PC le contenu de KEY
270	DADD=C	RAM SLCT	Choisit l'adresse RAM à C S&X
2B0	???	???	Ckear Data Register. Usage inconnu
2F0	DATA=C	WRITE DATA	Ecrit C vers un périphérique
330	CXISA	FETCH S&X	Met en C S&X le mot situé à l'adresse ADR
370	C=CORA	*C=C OR A	OU logique
3B0	C=C.A	*C=C AND A	ET logique
3F0	???	PRPH SLCT	Choisit le périphérique dont le numéro est en C S&X

Remarques Sur les NOP (000) : Seule l'instruction formée de dix zéros est un NOP réel, les autres peuvent être utilisées selon les besoins. C'est le cas des fonctions d'écriture des variables (Section 6.5.1 page 51), et des instructions 100 et 180 qui permettent à la 41CX de choisir entre les fonctions du module horloge et les fonctions supplémentaires.
Il n'est pas possible de passer d'un module 5 à l'autre à partir de n'importe quelle adresse. Il faut nécessairement utiliser les sous programmes ci-joints, que l'on trouve dans les deux modules :
5FC7 100 sélectionner la page 0
5FC8 3E0 RTN
5FC9 180 sélectionner la page 1
5FCA 3E0 RTN

Fig. 6.10 – Table de vérité de la fonction logique OU

C	A	C OU A
1	1	1
1	0	1
0	1	1
0	0	0

FIG. 6.11 – Table de vérité de la fonction logique ET

C	A	C ET A
1	1	1
1	0	0
0	1	0
0	0	0

FIG. 6.12 – Type 1 : GO/XQ absolus

Premier mot : A7 A6 AS A4 A3 A2 A0 0 1
Deuxième mot : A15 A14 A13 A12 A11 A10 A9 A8 G C
G et C selon tableau

G	C	HP	PPC	Remarques
0	0	GOSUB	?NC XQ	Exécute si Carry baissé
0	1		?C XQ	Exécute si Carry levé
1	0	GOLNC	?NC GO	Va si Carry baissé
1	1	GCL	?C GO	Va si Carry levé

FIG. 6.13 – Type 3 : Sauts relatifs

S6 S5 S4 S3 S2 S1 S0 C 1 1
C=condition

C	HP	PPC	Remarques
0	GONC	JNC xss	Saute si Carry baissé
1	GOC	JC xss	Saute si Carry levé

x= ici placer le signe (+ ou -)
ss= longueur du saut (HEXA, de +63 à -64)
Exemples

Code 244	Mnémonique PPC	Remarques
3F3	JNC -2	Saut en arrière de deux mots
0EF	JNC +1D	Saut en avant de 1D (29) mots

Pour faire le complément à 2, faire en Hexadécimal 80-saut

FIG. 6.14 – Opérations arithmétiques et logiques

Mot : I4 I3 I2 I1 I0 C2 C1 C0 1 0
Instruction champ type

Les mnémoniques HP sont identiques à ceux de PPC, sauf <> remplacé par EX et les shift remplacés par ASR, BSR, CSR et ASL.

n	bits	PPC	Remarques
0	00000	A=0	Efface le registre A
1	00001	B=0	Efface le registre B
2	00010	C=0	Efface le registre C
3	00011	A<>B	Echange A et B
4	00100	B=A	Met A dans B
5	00101	A<>C	Echange A et C
6	00110	C=B	Met B dans C
7	00111	C<>B	Echange B et C
8	01000	A=C	Met C dans A
9	01001	A=A+B	
10	01010	A=A+C	
11	01011	A=A+1	Incrémente A
12	01100	A=A-B	
13	01101	A=A-1	Décrémente A
14	01110	A=A-C	
15	01111	C=C+C	Multiplie par 2 le registre C. C'est aussi un décalage binaire de 1 bit vers la gauche
16	10000	C=C+A	
17	10001	C=C+1	Incrémente C
18	10010	C=A-C	
19	10011	C=C-1	Décrémente C
20	10100	C=0-C	Complément arithmétique
21	10101	C=-C-1	Devrait être 0-C-1, complément à 1 ou 9 selon le mode
22	10110	?B≠ 0	Lever Carry si...
23	10111	?C≠ 0	Lever Carry si...
24	11000	?A>C	Lever Carry si...
25	11001	?A<B	Lever Carry si...
26	11010	?A≠ 0	Lever Carry si...
27	11011	?A≠ *C*	Lever Carry si...
28	11100	RSHFA	Décaler A de 1 digit vers la droite
29	11101	RSHFB	Décaler B de 1 digit vers la droite
30	11110	RSHFC	Décaler C de 1 digit vers la droite
31	11111	LSHFA	Décaler A de 1 digit vers la gauche

Définition du champ (défini par HP comme "Durée validée")

n	bits	HP	PPC	Remarques
0	000	P	@R	Au digit spécifié par r
1	001	xx	S&X	A signe et exposant de C
2	010	PTR W	R<	à droite du pointeur (inclu)
3	011	W	ALL	Les 14 digits
4	100	PQ	P-Q	Entre les pointeurs
5	101	X	XS	Au signe de l'exposant
6	110	M	M	Mantisse seule (digits 3-12)
7	111	S	MS	Au signe de la Mantisse (13)

6.5.4 Instructions de type 3

Il s'agit de sauts relatifs. La valeur du saut est ajoutée au numéro de ligne de départ pour obtenir le numéro de ligne d'arrivée. A la main on compte en partant de 0 sur la ligne de départ (JNC ou JC), n'oubliez pas de compter en hexadécimal (bonne gymnastique intellectuelle). Les sauts négatifs sont comptés par leur complément (cf. exemple figure 6.13 page 55). Compte tenu du mode de notation des sauts négatifs par complément et des 7 bits donnant la longueur du saut, il est possible de sauter jusqu'à 64. En fait +63 (hexa) et -64.

Chapitre 7

Utilisation

Grâce à l'ingéniosité des adhérents de PPC, tous les modules ROM HP, y compris les modules internes, ont été déchiffrés, en informatique on dirait "désassemblés", cependant, connaître des listes d'instructions ne dit pas quel est leur but et il s'en faut de beaucoup que tout ait été éclairci.

Il y a au total plus de 500 pages de listages disponibles au club, il n'est donc pas question de les publier toutes ici, cependant, nous ne saurions terminer cette étude sans analyser quelques uns de ces éléments, choisis un peu arbitrairement comme illustrations.

7.1 Géographie logique d'un module

Comme tout le reste un module a sa géographie propre. En l'occurrence, il y a (modules externes seulement) :

1. à l'adresse x000 le numéro XROM du module et à l'adresse x001 le nombre de fonctions existant dans le module (tout ça en hexadécimal).
2. à partir de l'adresse X002 le catalogue des fonctions contenues dans le module. Ce catalogue est constitué de deux mots par fonction, donnant à la fois le type de programme et l'adresse d'entrée de la fonction. Les deux premiers bits du premier mot indiquent le type de programme. Ils sont à 00 s'il s'agit d'un programme en microcode et à 10 (2) si le programme est en langage standard.

   Les 4 derniers bits du premier mot et les 8 derniers bits du deuxième mot donnent les 3 digits d'adresse du programme à l'intérieur du module.

   Quand on exécute un CAT 2 les adresses des fonctions sont lues dans l'ordre où elles apparaissent dans le catalogue du module. Il est donc possible de faire lister les fonctions par ordre alphabétique si on en a envie, quelque soit la disposition des programmes dans le module. La première fonction listée est en général le nom du module. Comme tel ce nom a un numéro XROM. Bien sûr, ce nom ne correspond en général à aucune exécution possible.

   Si on prend la précaution de faire suivre ce nom de la fonction RTN, une tentative d'exécution sera sans inconvénient. Sinon un "plantage" risque fort d'en résulter.

   Le catalogue du module est terminé quand deux mots successifs sont à zéro (cf. Fig. 7.2 page 60).
3. à la fin du module ; aux adresses FFB, FFC, FFD et FFE se trouve le nom abrégé du R0M (première lettre en FFE, dernière en FFB).
4. Le dernier mot du module est la somme de contrôle (facultative).
5. Au dessus du nom du R0M, en fin de module, se trouvent 7 adresses particutiéres très importantes. Ces adresses sont interrogées (par un FETCH).

   Si elles contiennent 000 le programme interrogateur continue sans s'en préoccuper, sinon il se branche à cette adresse. Cette interrogation a lieu à plusieurs moments variant avec l'adresse (cf. fig. 7.2 page 60).
6. f) au milieu se trouvent des codes qui peuvent représenter :

— - un programme en langage utilisateur,
— - des instructions en microcode,
— - parmi ces instructions, des données.

La disposition d'un programme en langage utilisateur a été donnée en 3.4 page 26. Voici un exemple (Figure 7.1 page suivante

Les instructions en microcode et les données ne sont pas distinctes. Seule la façon d'utiliser le code fait la différence comme nous le verrons dans les exemples.

FIG. 7.1 – Programme en ROM

EN RAM (seul en fin de mémoire)	Programme	EN ROM(mots de contrôle 004 120)
C0	LBL“DO“	1C2
00		001
F3		0F3
00		000
44		044
4F		04F
02	LBL 01	inutile en ROM
F6	“MODULE“	1F6
4D		04D
4F		04F
44		044
55		055
46		046
45		045
7E	AVIEW	17E
89	PSE	189
B2	GTO 01	1B2
D1	compilé	00B
C8	END	1C6
02		02
29		22F

Les fonctions internes de la HP sont traitées par analyse directe des pressions de touches du clavier. Mais les fonctions des ROM sont exécutées à partir du point d'entrée.

Ce point d'entrée de la fonction donne accès à deux types de renseignements :

— - si la lecture se fait de bas en haut, les codes représentent les caractères composant le nom de la fonction, codés en microcode (cf. fig. 3.1 page 18).
La HP sait que le dernier caractère est atteint quand son code est augmenté de 080, ce qui revient à lever le bit de gauche de l'octet de code (bit numéro 7), qui sinon, est toujours à 0. De plus, les deux bits de gauche des premiers caractères du nom dans l'ordre de lecture gèrent les "prompts", ces traits (–) de demande d'argument qui suivent des fonctions comme STO–, GTO– (cf. figure 7.4 page 62).

— si la lecture se fait de haut en bas, c'est l'exécution de la fonction et les codes sont exécutés comme des instructions, si le premier mot est 000 NOP, la fonction est non programmable ; si le deuxième mot est également 000 NOP la fonction s'exécute immédiatement sans "NULL" possible.

7.2 Premier exemple, R↑

Cette fonction apparaît d'abord dans une table située en 14xx. Cette table donne les adresses de toutes les fonctions internes. Dans la fig. 7.3 page 61 vous pouvez voir de gauche à droite les numéros de ligne à partir de 1400. Ces numéros sont suivis du code ("microcode") en mode 244 puis 442. J'ai ajouté (à la main... avec l'aide de l'imprimante) le mnémonique de la fonction dont l'adresse est fournie par la table (microcode = adresse).

Vous trouvez R ligne 1474 avec un code 260. Rajoutez un devant le code et vous aurez l'adresse de la fonction R, c'est à dire 1260. Reportez-vous maintenant à la figure 7.4.

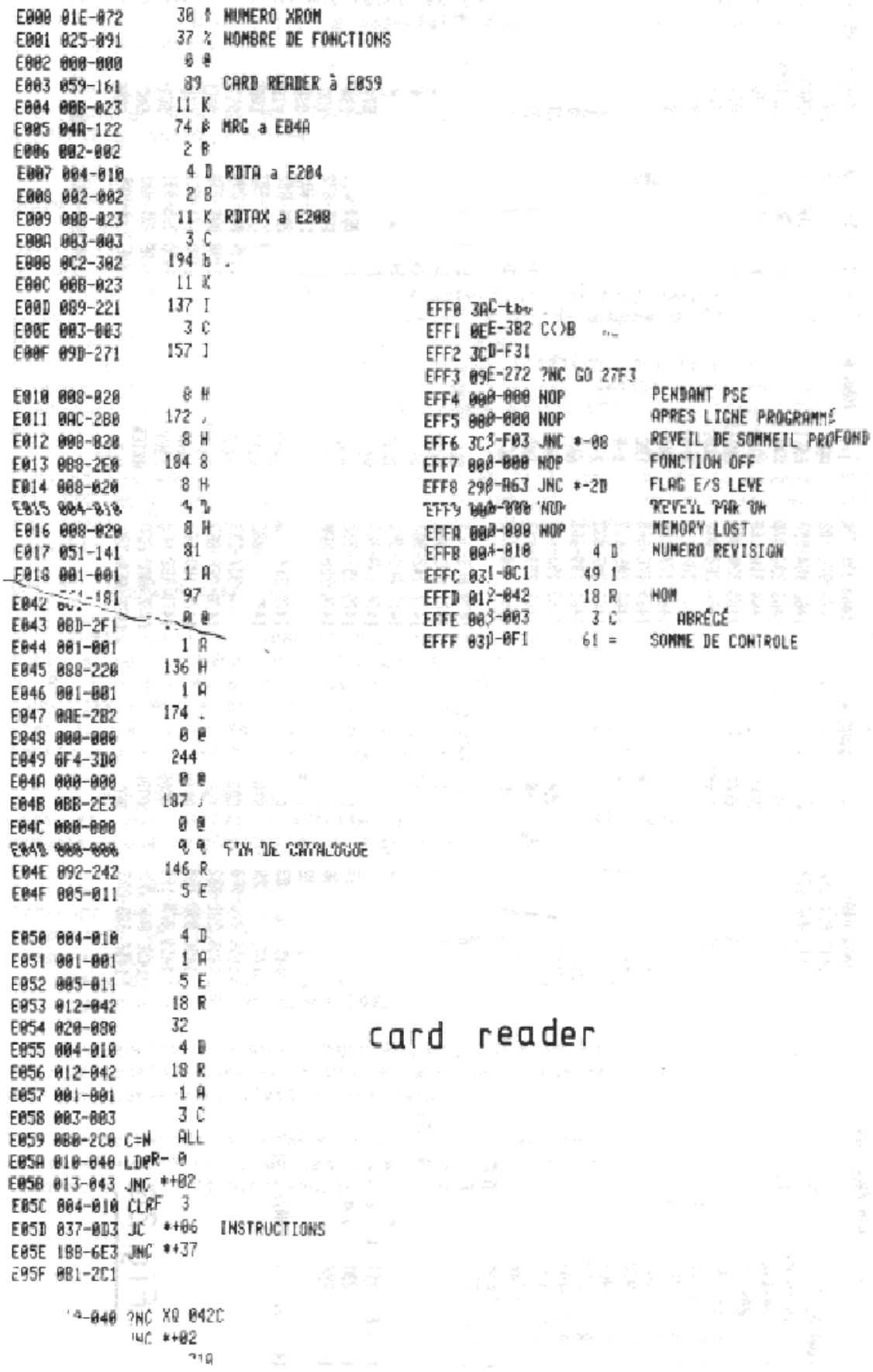
La ligne 1260 est celle qui est immédiatement en dessous du nom de la fonction que vous pouvez lire de bas en haut. L'espace blanc est là à cause du passage de 125F à 1260 qui correspond à un changement de paragraphe par l'imprimante.

Le code de ↑ est 1E, mais comme c'est le dernier caractère du nom, il est augmenté de 080 et donne 09E.

L'exécution commence donc en 1260 vers le bas. Vous serez peut-être déçu de constater qu'on ne trouve là qu'une nouvelle adresse où l'exécution doit se poursuivre, c'est à dire 14E5.

Cette "suite" elle même se contente d'exécuter un sous programme avant de terminer en 00EE, un des points de début des routines de "ménage" par lesquelles se terminent toutes les fonctions.

FIG. 7.2 – card reader



CODE	FONCTION
	NON PROGRAMMABLES
1400 0C8-320	CAT
1401 18C-630	GTO.
1402 124-490	DEL
1403 109-421	COPY
1404 0E7-393	CLP
1405 218-860	R/S
1406 292-A42	SIZE
1407 0C2-302	BST
1408 29E-A72	SST
1409 2A3-A83	ON
140A 1E7-793	PACK
140B 127-493	←
140C 34D-D31	MODE
140D 3E0-F80	
140E 348-D20	SHIFT
140F 09E-27?	ASN
1410 3EC-FB0	ENTREE DE DIGITS
	TRAITEMENT α
141D 0B5-211	GTO α
141E 0B5-2D1	XEQ α
141F 3E0-F80	
1420 3FF-FF3	RANGS 2 ET 3

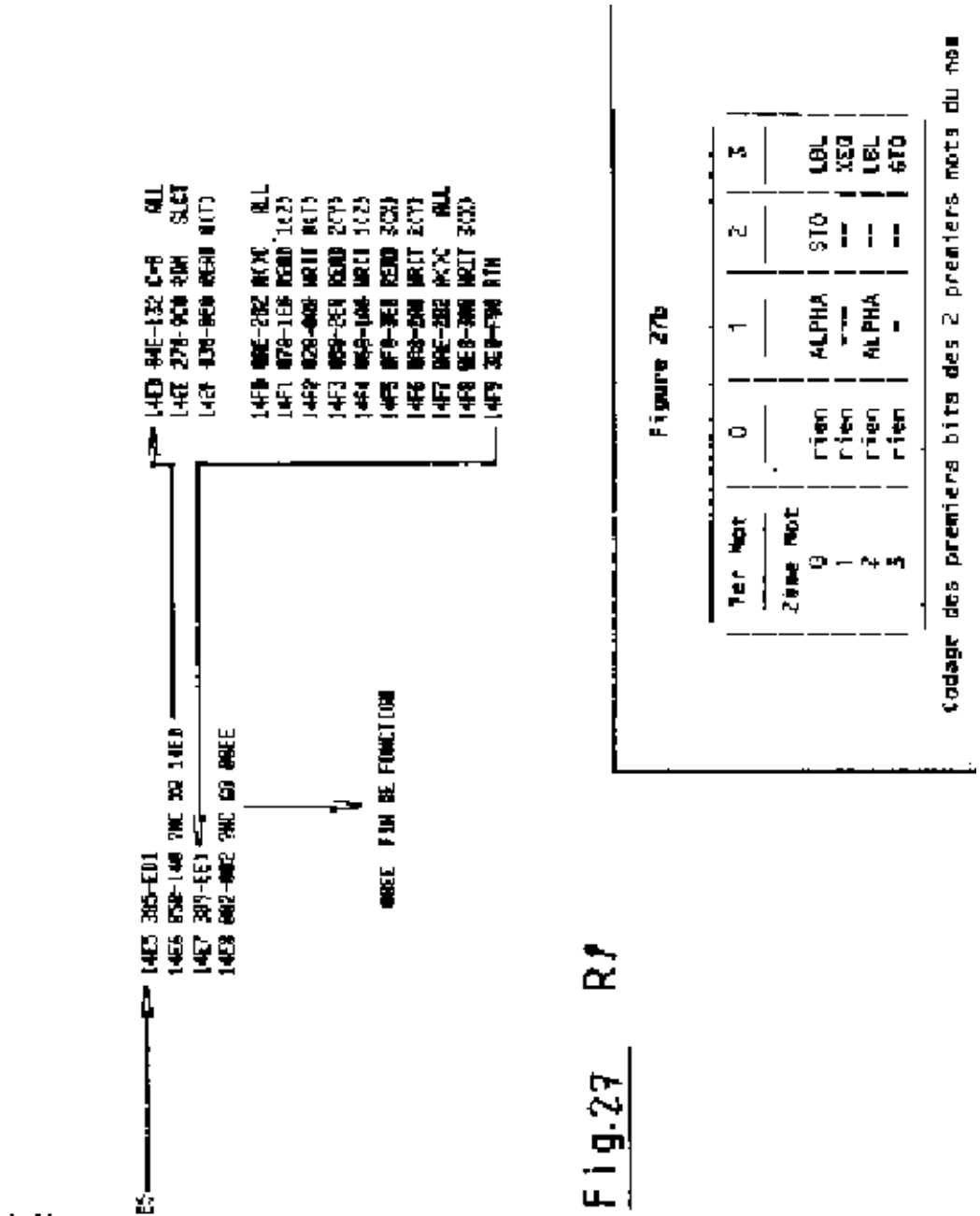
1440 04A-122	+	RANG 4
1441 054-150	-	
1442 05C-170	*	
1443 06F-1B3	/	
1444 308-C20	X<Y?	
1445 320-C80	X>Y?	
1446 2F6-BD2	X<=Y?	
1447 26D-9B1	Σ+	
1448 271-9C1	Σ-	
1449 032-0C2	HMS+	
144A 045-111	HMS-	
144B 04F-133	MOD	
144C 061-181	%	
144D 1EC-7B0	%CH	
144E 1BC-770	P-R	
144F 1C0-700	R-P	
1450 1A6-692	LN	RANG 5
1451 06B-1A3	X↑2	
1452 298-A60	SQRT	
1453 02A-0A2	Y↑X	
1454 23A-8E2	CHS	
1455 147-513	E↑X	
1456 1AC-6B0	LOG	
1457 2CA-B22	10↑X	
1458 163-583	E↑X-1	
1459 288-A20	SIN	
145A 27C-9F0	COS	
145B 282-A02	TAN	
145C 098-260	ASIN	
145D 07D-1F1	ACOS	
145E 0AA-2A2	ATAN	
145F 32B-CA3	DEC	

1460 1D6-752	1/X	RANG 6
1461 076-1D2	ABS	
1462 154-550	FACT	
1463 2DC-B70	X≠0 ?	
1464 31A-C62	X>0?	
1465 220-880	LN1+X	
1466 2E8-BA0	X<0?	
1467 30E-C32	X=0?	
1468 177-5D3	INT	
1469 17C-5F0	FRC	
146A 10E-4J2	D-R	
146B 20E-832	R-D	
146C 199-661	HMS	
146D 193-643	HR	
146E 257-953	RND	
146F 330-CC0	OCT	
1470 0F3-3C3	CLΣ	RANG 7
1471 2FC-BF0	X<>Y	
1472 242-902	PI	
1473 0F9-3E1	CLST	
1474 260-980	R↓	
1475 252-942	RDN	
1476 228-8A0	LAST X	
1477 101-401	CLX	
1478 314-C50	X=Y?	
1479 2E2-B82	X≠Y?	
147A 337-CD3	SIGN	
147B 2EF-BB3	X<=0?	
147C 1B9-6E1	MEAN	
147D 1B2-6C2	SDEV	
147E 0B2-2C2	AVIEW	
147F 0E0-380	CLD	

1480 114-450	DEG	RANG 8
1481 11F-473	RAD	
1482 11A-462	GRAD	
1483 13E-4F2	ENTER↑	
1484 215-851	STOP	
1485 25C-970	RTN	
1486 008-2E3	BEEP	
1487 0D1-341	CLA	
1488 092-247	ASHF	
1489 1FC-7F0	PSE	
148A 0ED-3B1	CLRG	
148B 345-D11	AOFF	
148C 33C-CF0	AON	
148D 1C8-720	OFF	
148E 209-821	PROMPT	
148F 14D-531	ADV	
1490 22E-8B2	RCL	RANG 9
1491 0B0-362	STO	
1492 2B0-AC0	ST+	
1493 2B9-AE1	ST-	
1494 2B8-A80	ST*	
1495 2C1-B01	ST/	
1496 19E-672	ISG	
1497 12D-481	DSE	
1498 2D6-B52	VIEW	
1499 277-9D3	SREG	
149A 0A4-290	ASTO	
149B 08C-230	ARCL	
149C 171-5C1	FIX	
149D 265-991	SCI	
149E 135-4B1	ENG	
149F 2B0-B40	TONE	
1480 3E7-F93	XROM (8)	

FIG. 7.3 – Codes des fonctions

FIG. 7.4 – *Fonction R↑*



Le coeur est donc ce sous programme qui commence en 14ED. Si vous programmez en microcode et que vous voulez exécuter R↑, il suffira d'appeler ce sous programme, c'est plus simple. Mais il n'est pas sûr que cette adresse sera la même sur tous les modèles de 41C. Ce sous programme commence par sélectionner le registre R(000), puis met en réserve en A le registre T. Ensuite nous voyons monter la pile : Z va en T, Y en Z, X en Y et finalement on récupère T pour le mettre en X. Simple, non ?

7.3 Deuxième exemple, validation de l'affichage

L'affichage est un des multiples périphériques de la 41C. Pour le valider et bien sûr ensuite l'utiliser, il faut :

07F6 130-4C0 LDI S&X

07F7 010-040 16 P

07F8 270-9C0 RAM SLCT

07F9 130-4C0 LDI S&X

07FA 0FD-3F1 253

07FB 3F0-FC0 PRPH SLCT

07FC 3E0-F80 RTN

07FD 000-000 NOP

Comme vous le voyez, il faut d'abord invalider la mémoire vive en sélectionnant des registres inexistants, de R(010) à R(01F) ceux qui sont situés juste au dessus du registre e, dans le vide.

Ensuite, on charge en C S&X l'adresse de l'affichage, soit 0FD et on le sélectionne. A partir de ce moment nous pouvons écrire à l'écran. Vous trouverez en annexe 8.11 page 81, le détail des instructions, mais voici tout de suite un exemple.

7.4 Messages

Nous trouvons parfois en mémoire des codes comme celui-ci (cf. fig. 7.5 page suivante, en haut à gauche). Ce groupe d'instructions commence par XQ 07EF, voyons donc ce que fait ce sous programme (à droite de la figure).

Tiens, un POP amène en ADR de C la première adresse de retour. Cette adresse nous la connaissons, c'est celle qui est juste en dessous de XQ 07EF, donc 2FF0.

La HP fetche, c'est à dire va chercher le contenu de cette adresse et le place en C S&X. Ce contenu est le code 018.

Ce code est écrit dans l'écran (qui a été préalablement sélectionné) en tant que caractère, ce n'était donc pas une instruction !

C=C+1 ajoute 1 à l'adresse qui devient 2FF1 et la HP teste le XS ; est-il différent de zéro ? Si oui, armer Carry. Ce n'est pas le cas ici, donc recommencer au FETCH.

On envoie donc à l'affichage successivement les codes 018, 012, 00F, 00D puis finalement 220. Mais là, il y a autre chose que 0 en XS, donc nous sortons de la boucle et par GTO ADR nous allons continuer l'exécution à l'adresse suivante.

Cette adresse contient un RTN et termine donc le sous programme par retour au programme principal. Mais que font tous ces codes dans l'affichage ? Nous allons le voir en corrigeant le listage pour faire apparaître les caractères. Vu ?

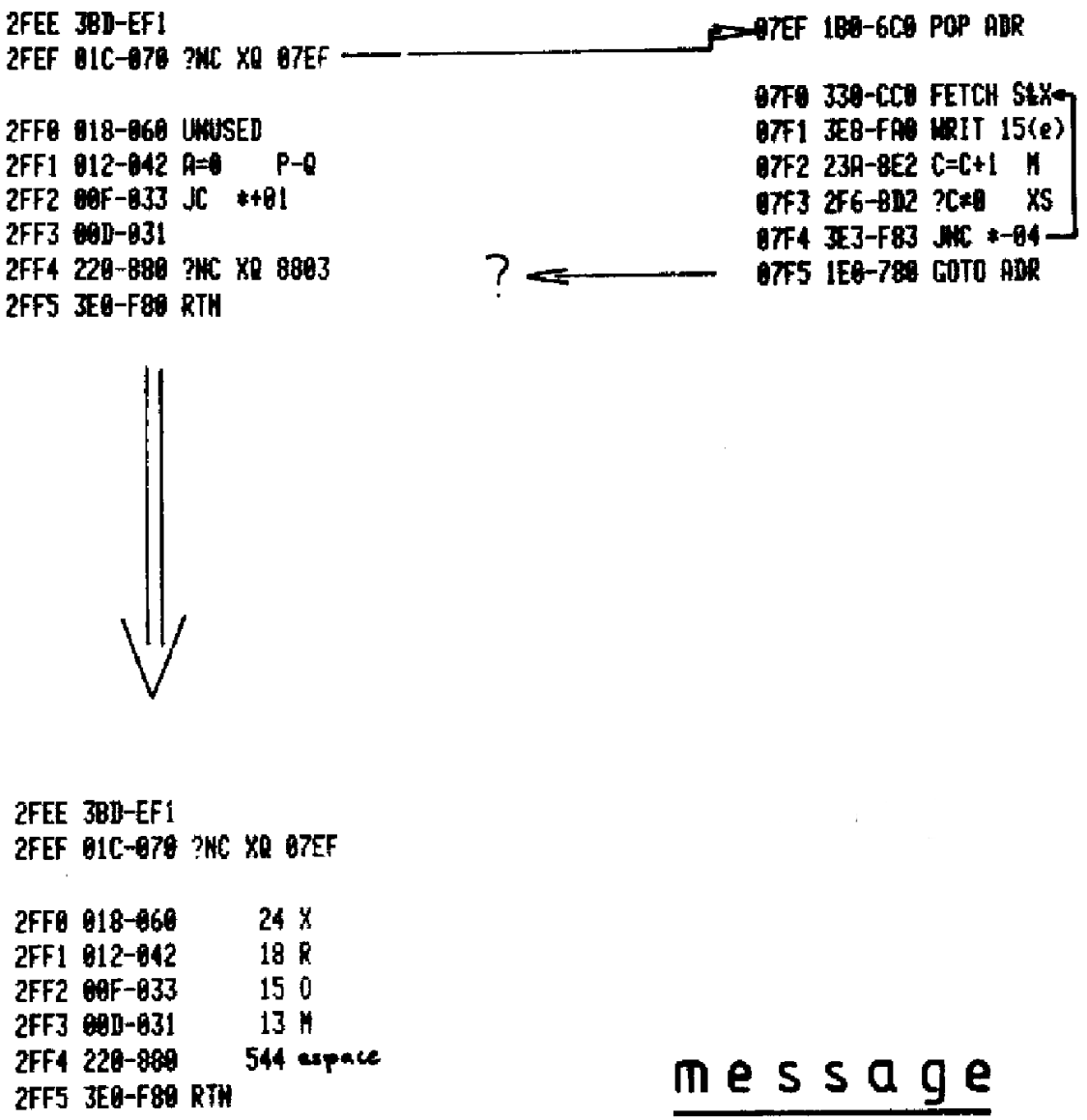
7.5 Départ à chaud ou départ à froid ?

Nous allons maintenant examiner un passage particulier de la mise en route de la HP (fig. 7.6 page 66).

Commençons en 01EA. La machine vient de s'allumer (du sommeil profond) et après avoir repris ses esprits elle vient voir si une autre touche est pressée (une autre que ON). Sinon, elle saute à 01F4. Si oui, elle teste pour savoir quelle est cette touche : elle charge 0C3 en C S&X, puis en A S&X, puis prend en C le contenu de registre KEY et transporte ce contenu en C S&X où elle peut le comparer au 0C3 qui se trouve maintenant en A, cette comparaison se faisant seulement sur les digits 0 et 1.

Si les deux éléments ne sont pas différents, c'est que la touche pressée a bien le code C3, qui est celui.., devinez ou allez voir Fig, 6.4 page 49. Dans ce cas allez au départ à froid.

FIG. 7.5 – *Message*



S'il s'agit de toute autre touche, on se retrouve dans le cas où il n'y a pas de touche pressée, en 01F4.

Il est alors temps de rallumer l'affichage et d'aller dare-dare vérifier si le flag 11 (autostart) est levé.

Pour cela, il faut lire le registre d, déplacer le flag 11 en bonne place pour être mis en ST et testé alors qu'il se trouve en position 0.

Si ce flag n'est pas levé, on peut terminer les opérations, si le flag est levé, on part exécuter le programme utilisateur courant. Mais avant, on exécute TONE 7 après avoir éteint l'affichage en manière de clin de d'oeil.

Nous sommes alors en 0204. Il serait logique de continuer par 0205. pourtant, dans la chronologie de la machine ce passage a été exécuté avant celui que nous venons de voir. Rien n'est simple ! D'abord, dans ce passage (appelé CHECKRAM, test de la mémoire vive) une remise à jour du clavier et du mode de calcul du CPU, puis une désactivation des périphériques et la sélection de RAM 0, les registres d'état, pour pouvoir lire c et vérifier si la constante de départ à chaud (ce 169 qui vient d'être mis en A) est bien présente en c.

Sinon, on va tout de suite faire un départ à froid. RCR 11 amène en S&X l'adresse de R00, on enlève 1 et on se trouve en tête des programmes. Si le fonctionnement est normal il doit toujours y avoir au moins un registre, contenant le .END., ici ; les instructions suivantes vont le vérifier.

Quand il n'y a rien en face, une lecture de registre donne toujours une série de bits à 1 ou une série de bits à 0 dans tout le registre C. Mais cette disposition peut aussi exister dans un registre normal, comment savoir ?

La HP prend le complément d'*une partie* du registre C et écrit le résultat dans le registre (prendre le complément c'est remplacer tous les 0 par des 1 et tous les 1 par des 0).

Une nouvelle lecture redonnera le même contenu, si le registre est bon, il y aura donc à la fois des 0 et des 1 dans C.

Si le registre est mauvais, on retrouvera la même chose que lors de la première lecture,

La deuxième prise de complément va donc rétablir le contenu d'un registre correct et défaire celui d'un mauvais registre, ce qui est contrôlé au mot suivant. Si le registre est mauvais, on va au départ à froid, sinon, un WRITE DATA rétablit le registre dans son contenu original.

On remet en ST le jeu 0 des flags et, si le programme n'est pas en ROM, c'est tout,

Sinon on va procéder à une opération curieuse : vérifier si pendant le sommeil de la HP un petit malin n'a pas subtilisé le module dans lequel se trouve le pointeur !

Donc on lit le pointeur en b, on met à 0 les 3 digits de droite, ne reste donc plus que le numéro de port, on place cette adresse X000 qui est celle du premier mot du ROM en ADR, et on lit. S'il y a autre chose que 0, la HP suppose que le ROM est toujours là et termine son contrôle. S'il y a 0 elle renvoie le pointeur au début de la mémoire programme (c'est la signification des instructions qui suivent).

Cette analyse me suggère une félonie. La HP vérifie l'existence d'un ROM, mais lequel ? Voici une occasion d'aller mettre son nez dans le microcode,

Pour cela, il faut disposer d'un ROM possédant un programme en langage utilisateur et d'un autre ROM ayant à la même adresse un programme microcode.

J'ai fait l'expérience avec le PPC ROM et le module X-fonctions.

Placez dans un port le PPC ROM. Faites GTO "NK" puis éteignez la machine et placez à la place du PPC ROM le module X-F.

Rallumez, PRGM ON, vous voyez 0l STO 03. Il y a de drôles de choses dans ce module ! des SST maintiennent le même affichage, mais BST donne 01 P-R. Essayez LIST.., amusez-vous.

Revenons au microcode. Nous abordons maintenant la routine de départ à froid, un frisson va courir dans votre dos ! (cf. fig. 7.7 page 67 et suivantes).

Les 7 premières lignes vous mettent au diapason : tout à zéro. on pousse ensuite 0 dans les 4 niveaux de la pile SBR, on place le pointeur Q à R = 13, mais on laisse P en place.

Tous les flags sont alors effacés et la routine d'affichage des messages d'erreur est appelée en lC6C. Le mot suivant, utilisé par la routine indique le sinistre MEMORY LOST.

07F6 valide l'affichage, 0098 nettoie le registre de clavier.

Nous arrivons à un point crucial. La HP charge 3FF. C'est l'adresse du plus haut registre à effacer. HP prévoyait donc déjà les x mémoires ! Les ingénieurs HP ont hésité à choisir ce chiffre qui allonge sensiblement le départ à froid et ont failli le ramener à 1FF, ce qui aurait sauvé les X-mémoires. Avantage ou inconvénient ? Il faut faire avec.

Le C=0 WRITE DATA agit sur l'écran qui vient d'être validé, en effaçant les indicateurs (BAT, USER,...) ensuite la HP efface les 3FF registres possibles après avoir invalidé l'affichage (C=0 PRPH SLCT).

FIG. 7.6 – *Départ*

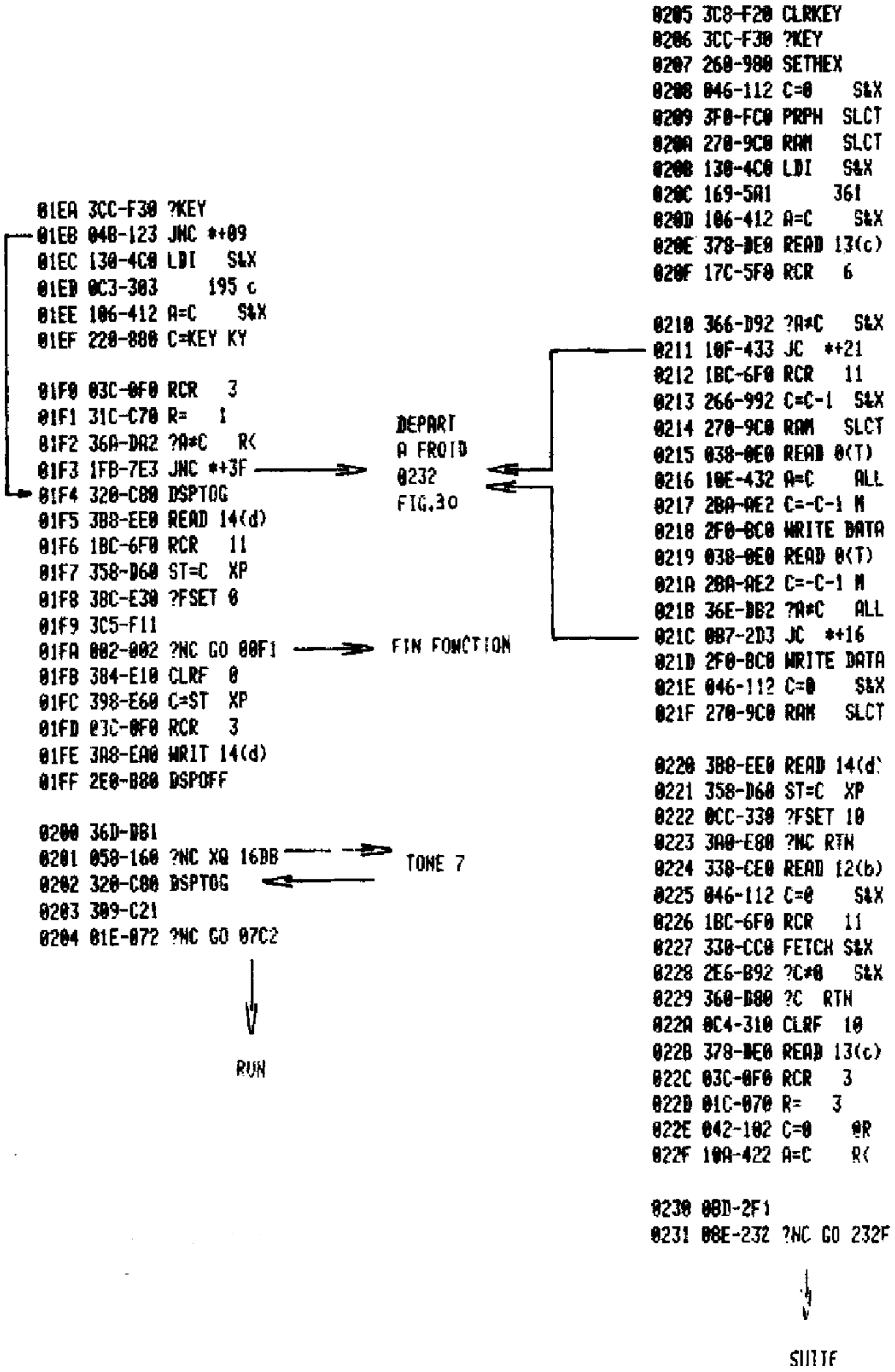


FIG. 7.7 – Départ à froid (1)

```
   DEPART A FROID

0232 260-980 SETHEX
0233 1A0-680 A=B=C=0
0234 150-560 M=C   ALL
0235 070-1C0 M=C   ALL
0236 05B-160 G=C   @R;+
0237 35B-D60 ST=C  XP
0238 258-960 T=ST
0239 170-5C0 PUSH ADR
023A 170-5C0 PUSH ADR
023B 170-5C0 PUSH ADR
023C 170-5C0 PUSH ADR
023D 0E0-380 SLECT Q
023E 2DC-870 R=   13
023F 0A0-280 SLECT P

0240 2C4-B10 CLRF  13
0241 344-D10 CLRF  12
0242 184-610 CLRF  11
0243 0C4-310 CLRF  10
0244 244-910 CLRF  9
0245 104-410 CLRF  8
0246 1B1-6C1
0247 070-1C0 ?NC XQ 1C6C ---
0248 02D-0B1      45 -       MEMORY LOST
0249 3D9-F61           <---
024A 01C-070 ?NC XQ 07F6 ---  VALIDE
024B 261-981           <---   ECRAN
024C 000-006 ?NC XQ 0009 ---  CLAVIER?
024D 130-4C8 LDI   S&X   <---
024E 3FF-FF3     1023     -
024F 10E-432 A=C    ALL
```

Il faut ensuite rétablir les valeurs par défaut OEF (41C et CV) pour R00, donc CEE pour .END. (pas de programme) et CFA pour REG.

Ensuite nous mettons à jour par défaut les flags.

Il faut lever le flag message (pour MEMORY LOST) puis aller faire le tour des périphériques (sous programme 27E6) à partir du point 6 (voir 5 page 58) pour voir s'ils réclament un service. Reste ensuite à mettre en place la constante de départ à chaud 169 qui indiquera que tout ce travail a été fait.

Pour finir, nous allons vérifier en 01F5 si un périphérique n'a pas levé le flag d'exécution automatique (cas de la lecture d'une carte magnétique idoine).

Nous avons vu ça tout à l'heure.

Voilà. Elle en fait du travail, cette petite ?

7.6 Un peu de musique

Nous allons maintenant examiner le fonctionnement des "TONE".

Vous allez voir que les TONE artificiels (postfixe supérieur à 9) ont une drôle d'origine ! (cf. fig. 7.10 page 69 et 7.11 page 70). L'entrée pour TONE est ligne 12D0 (après 149F). La HP a déjà en ST le numéro de tone à exécuter. La HP vérifie qu'elle a le droit de faire du bruit (flag 26) et en profite pour remettre en C le numéro du TONE. Elle met ensuite à 0 le registre T, ce qui prépare le BIP. En effet c'est le registre T qui commande, par sa variation, le bipper. Le numéro de TONE est à nouveau sauvé en ST et FF placé en C M puis en A M. Une opération curieuse prend place maintenant. N'oublions pas que le numéro de TONE normal est un chiffre décimal de O à 9. La HP prend donc son complément décimal. 9->0, 8->1, etc...

Les TONE ont une durée égale mais une fréquence variable ; un cycle de TONE de haute fréquence dure moins longtemps qu'un cycle de TONE de basse fréquence. Le nombre de cycles de TONE à exécuter pour une durée constante est donc variable.

La valeur précédente permet de choisir ce nombre (durée). Cette opération se réalise dans C MS, donc sur un seul digit. Puis nous allons choisir dans la table la valeur convenable. Pour cela, par une procédure très fréquente dans la HP, nous

FIG. 7.8 – Départ à froid (2)

```
0250 04E-132 C=0    ALL
0251 2F0-BC0 WRITE DATA
0252 2E0-B80 DSPOFF
0253 320-C80 DSPTOG
0254 3F0-FC0 PRPH  SLCT
0255 0AE-2B2 A<>C   ALL
0256 270-9C0 RAM   SLCT
0257 0AE-2B2 A<>C   ALL
0258 2F0-BC0 WRITE DATA
0259 1A6-692 A=A-1  S&X
025A 3B8-F63 JNC *-85
025B 130-4C0 LDI   S&X
025C 0EF-3B3       239
025D 13C-4F0 RCR   8
025E 130-4C0 LDI   S&X
025F 0FA-3E2       250

0260 03C-0F0 RCR   3
0261 130-4C0 LDI   S&X
0262 0EE-3B2       238
0263 368-DA0 WRIT 13(c)
0264 05A 162 C=0    M
0265 22E-8B2 C=C+1  ALL
0266 328-CA0 WRIT 12(b)
0267 270-9C0 RAM   SLCT
0268 04E-132 C=0    ALL
0269 09C-270 R=   5
026A 310-C40 LDAR- C
026B 130-4C0 LDI   S&X
026C 020-080       32
026D 3A8-EA0 WRIT 14(d)
026E 04E-132 C=0    ALL
026F 270-9C0 RAM   SLCT
```

FIG. 7.9 – Départ à froid (3)

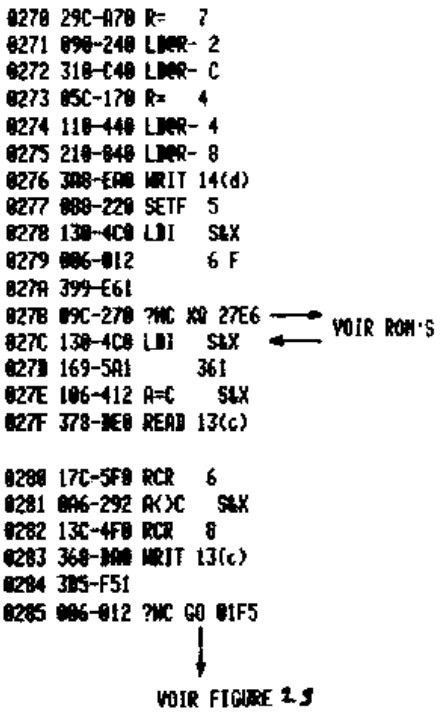


FIG. 7.10 – Programme Tone (1)

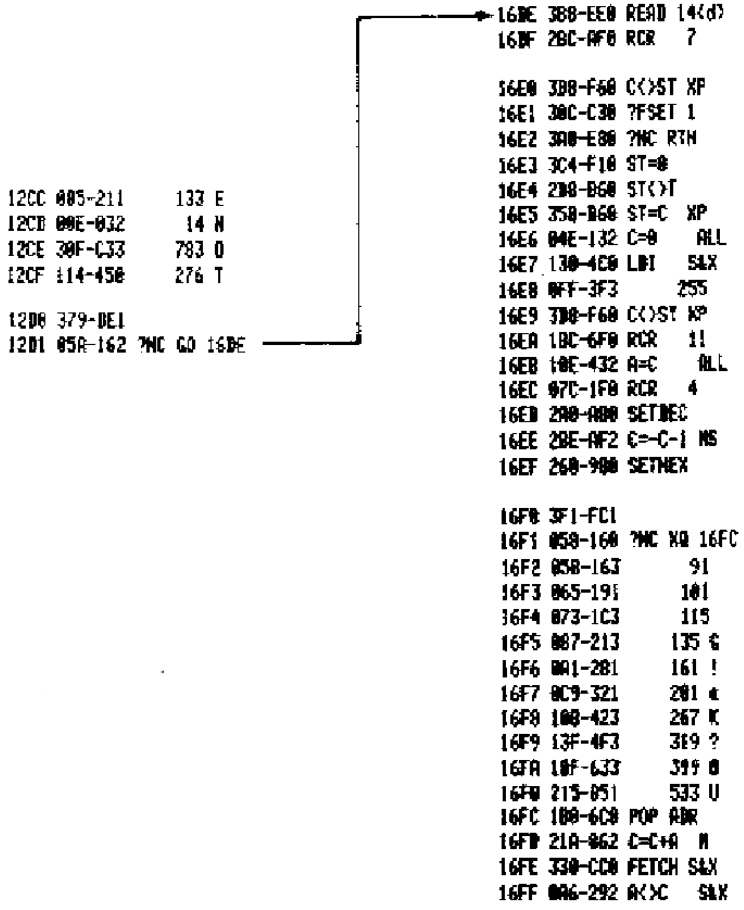


FIG. 7.11 – *Tone (2)*

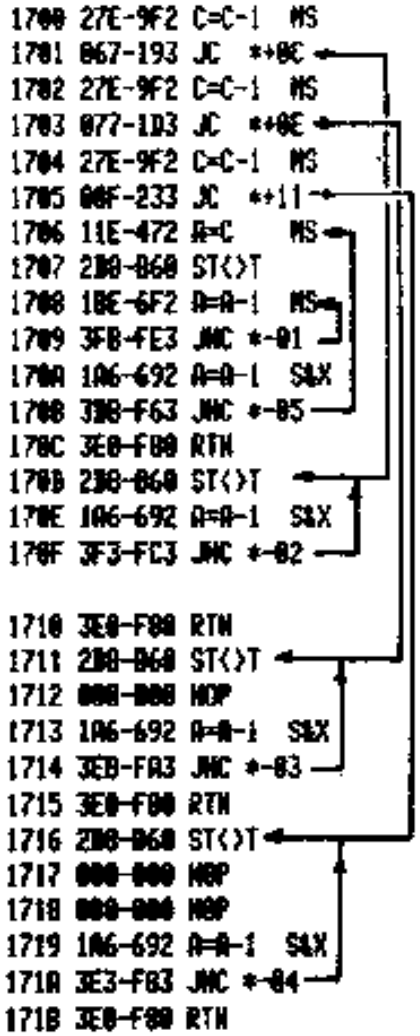
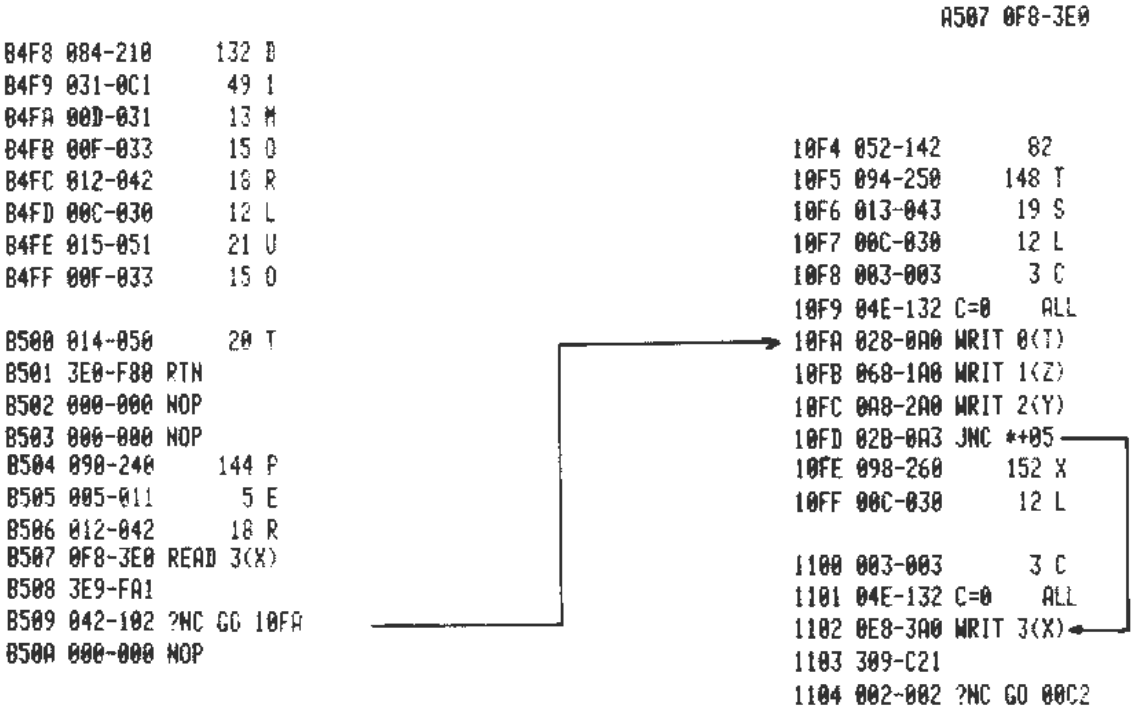


Fig. 7.12 – REP



sautons une zone de données pour arriver sur un POP ADR qui prend l'adresse 16F2 et y ajoute le numéro de TONE (de 0 à 9). Vous voyez donc que TONE 0 a le nombre de cycles 058, TONE 1 : 065... Cette constante de durée est mise en A S&X qui va servir de compteur.

Rappelons nous que nous avons en C MS la hauteur du TONE. Nous allons donc avoir divers branchements.

Pour TONE 9, C MS contient 0. Quand on enlève 1, on lève le flag de Carry et on saute à 170D où on produit autant d'échanges entre ST et T (donc valeurs FF et 00) qu'indiqué en A.

Il y a ici 3 lignes à exécuter, le minimum possible, donc la plus haute fréquence possible.

Pour TONE 8, C MS contient 1, on ne saute donc pas à la première soustraction mais à la deuxième, on va alors en 1711. Nous avons le même principe, mais un NOP est intercalé, donc une fréquence plus basse.

Pour des durées plus élevées, un double compteur est mis en place. Les TONE supplémentaires (supérieurs à 9) amènent plusieurs constatations :

— La constante de nombre de cycles est prise dans l'ordre des numéros de TONE. TONE 10 (décimal) prend donc comme constante le code du POP ADR... et ainsi de suite, la durée des TONE dépend des codes qui n'étaient pas prévus pour cela mais pour servir d'instructions.

— La valeur du TONE qui se trouve en XP peut-être supérieure à 9. Je ne sais pas trop comment réagit le microprocesseur quand on lui demande de prendre le complément décimal d'un chiffre supérieur à 9 ? Je n'ai pas jugé utile de faire l'essai en microcode, l'expérience à l'oreille suffit.

7.7 Le microcode par les utilisateurs : REP, XCAT et CHARGE

7.7.1 D'abord un exemple très simple : REP

Cette fonction, disponible sur une EPROM baptisée Toulrom 1D (ROM 1D fait à Toulouse) a comme objet de recopier x dans Y, Z et T en une seule instruction.

Cette opération s'effectue habituellement par ENTER, ENTER, ENTER, ce qui n'est pas bien sorcier et sert à utiliser la faculté de duplication du registre T lors des calculs de constantes : par exemple, si vous faites 1 REP + + +.., vous obtenez un compteur qui ajoute 1 à chaque +.

C'est le système utilisé dans LB pour calculer le nombre d'octets libres.

Pour réaliser une fonction en microcode, deux voies s'ouvrent à vous.

Construire vous même votre fonction à partir des instructions du microcode, et cela est parfaitement possible mais vous demande une parfaite connaissance de celles-ci et surtout une analyse parfaite du problème. N'oubliez pas qu'avec le microcode nous travaillons sans filet. En programmation normale, il est toujours possible de faire R/S si le programme refuse de sortir d'une boucle.

En microcode, si vous ne testez pas la touche R/S, la HP ne connaît pas son existence.

Autre inconvénient de cette méthode, elle conduit à des programmes longs.

La deuxième méthode consiste à faire un usage intensif des modules qui vous sont obligeamment fournis par HP dans la mémoire interne. Le seul ennui est qu'il faut les connaître, ce qui ne peut se faire, vu le volume, que par une collaboration au sein d'un club, où les documents nécessaires sont disponibles.

C'est cette dernière méthode que j'emploie ici, vous allez voir à quel point elle est efficace.

La fonction REP, au moins pour la partie qui figure dans I'EPROM, est limitée à son nom, à READ 3(x) et à ?NC GO 10FA (cf, fig. 7.12 page précédente).

Difficile de faire plus simple, non ? La seule chose faite est de lire le contenu de x et de le placer dans C.

Mais qu'y a-t-il en 10FA ? Comme vous le voyez sur la figure, c'est un des points d'entrée (ce n'est pas le point d'entrée habituel) de la routine CLST. Cette routine efface la pile en mettant C à O en recopiant C dans La pile. Je n'utilise donc que la partie recopie de cette routine. Remarquez au passage que la routine CLST utilise elle aussi sa consoeur CLX.

7.7.2 Du plus sérieux : XCAT

Tous ceux qui ont dans leur machine plusieurs modules ont été comme moi souvent exaspérés par la difficulté qu'il y a à retrouver l'orthographe d'une fonction à l'aide du CAT 2. Un de mes amis a mesuré environ 2mn 30s pour le total d'un catalogue 2.

XCAT a la particularité de commencer le CAT 2 au module que vous désignez par son numéro XROM. Par exemple, 30 XCAT commence le catalogue par le lecteur de cartes, qui se trouve pourtant toujours à l'adresse Exxx, l'avant dernière de la mémoire. Ensuite le catalogue continue normalement. Il est en particulier possible de SST et de BST à volonté.

Le programme (cf. fig. 7.13 page suivante) commence par son nom, comme usuel. Il lit en x le numéro du ROM dont on veut le catalogue, le traduit en binaire à l'aide de la routine interne BCDBIN qui commence en 02E3. Cette routine prend un nombre décimal de 3 chiffres en C et le traduit en binaire en S&X de C.

XCAT initialise ensuite les registres C et B du microprocesseur. Elle place le numéro du premier module possible moins 1, soit 4, dans le digit voulu de C.

Elle range en A le code XROM, replace le pointeur et entame la recherche du ROM demandé.

Pour cela, elle incrémente C au pointeur. Si C dépasse F (hexa) le flag Carry est levé et l'instruction suivante exécutée : si on dépasse sans avoir trouvé le ROM demandé c'est qu'il n'est pas en mémoire, on va donc en 02E0 qui provoque l'affichage de NONEXISTENT. Si la valeur au pointeur est inférieure à F on place (FETCH) en C S&X le premier mot du ROM testé. Ce mot, rappelons le, est le numéro XROM. on le compare au code mis en réserve en A. S'il est différent on prend le mot suivant qui est le nombre de fonctions du ROM testé.

On accumule ce nombre de fonctions en B, puis on teste le ROM suivant. Si c'est le bon ROM, on initialise le CAT 2 en plaçant en P le nombre de fonctions sautées et le numéro du catalogue (2) puis on se branche sur le catalogue 2 standard qui démarre à la première fonction du ROM désiré, en général son nom.

Comme vous êtes dans un CAT 2 standard, vous en avez tous les avantages : aller en avant ou en arrière pas à pas, ralentir le défilement en pressant une touche (ou l'accélérer sur 41CX) toutes choses qu'il aurait fallu programmer vous même sinon.

7.7.3 Une preuve de puissance : CHARGE (de Stéphane Barizien)

Ce programme a été créé en France par un jeune membre de PPC-T, Stéphane Barizien, qui est vite devenu un virtuose du microcode. Il illustre de façon éclatante ce que j'ai cherché à montrer avec REP, c'est à dire l'efficacité que l'on peut retirer d'un bon usage des modules internes de la HP-41. Cette fonction est non programmable, et donc peut s'exécuter même si la machine est en mode programme. Elle affiche CHARGE— et attend un nombre décimal de 3 chiffres. Ce nombre donné elle introduit en mémoire programme l'octet correspondant. Le seul défaut de ce programme, encore rustique, est de ne pas mettre à jour les numéros de ligne, et de placer l'octet au dessus de la ligne affichée. Ces défauts auraient pu être corrigés, mais en perdant la brièveté si nette du programme. Le mode opératoire est donc le suivant :

— Assigner à une touche la fonction CHARGE,
— Se mettre en mode programme

FIG. 7.13 – *Programme XCAT*

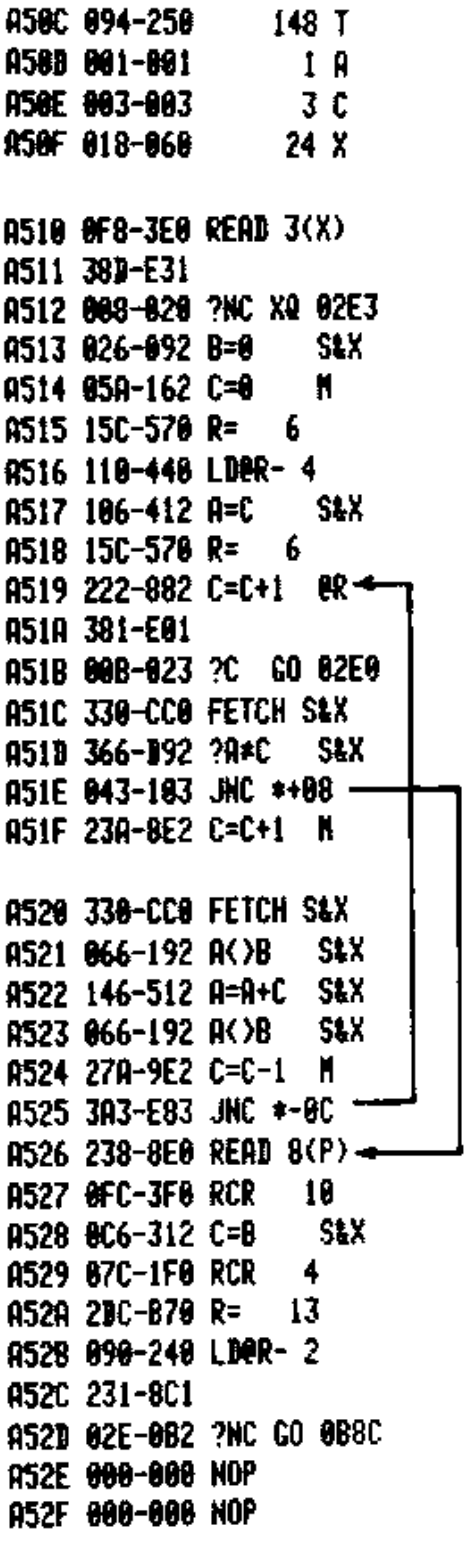
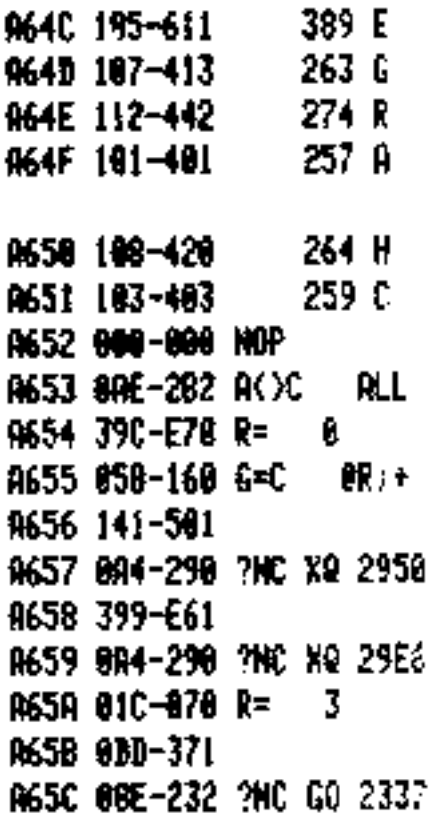


Fig. 7.14 – Programme CHARGE



— Taper ENTER
— Exécuter avec la touche assignée (en mode USER) CHARGE
— Fournir le code
— Recommencer autant de fois qu'il y a de codes à fournir.

Quand c'est fini effacer ENTER, dans le processus de retour en arrière qui suit, la HP renumérote les lignes.

Exemple : ENTER CHARGE 242, CHARGE 068, CHARGE 079, <-, place "DO" en mémoire.

Ce programme est l'équivalent microcode de LB. Comptez, si on ne tient pas compte du nom, 11 mots ! Qui dit mieux ! Dans le programme, le NOP indique une fonction non programmable, les bits de gauche du début du nom ont provoqué la saisie du nombre décimal à l'affichage, sa transformation en binaire et son stockage en A. le nombre est repris en C et stocké en G, ce qui, au passage le limite à 8 bits et, donc à 255. Si vous répondez 300 le programme prend 300-255. Charge récupère ensuite le compteur de programme (PT) en b (routine 2950, "GETPC"). La routine 29E6 ("INBYT") insère un octet contenu dans G dans le programme après avoir incrémenté le PC. 2337 ("PUTPC") remet en place le compteur, il nécessite R=3. Tout simple.

Chapitre 8

Annexes

8.1 Apprenons à compter (binaire, hexadécimal)

On sait, sans se tromper, si un interrupteur est ouvert ou fermé. Un ordinateur aussi le sait. Tout ce qu'il fait se résume à cela. Mais nous ne sommes jamais contents, il nous faut faire des mathématiques. Pour représenter un interrupteur il nous faut désigner deux états, et donc compter en base 2, binaire. Nous comptons alors :

0 zéro

1 un

10 dix

11 onze

etc...

On appelle bit (binary digit) chacun des chiffres 0 ou 1. Mais nous avons l'habitude de compter en décimal (base 10), d'où des conversions à faire.

Hélas, il se trouve que la traduction de base 2 en base 10 se fait mal, il n'y a pas de correspondance entre les chiffres obtenus en base 10 et les bits de la base 2, mais seulement correspondance entre les nombres. Par exemple :

1 en base 2 ou 10 s'écrit pareil

2 en base 10 donne 10 en base 2 (!!)

12 en base 10 ne donne pas 1 et 2 soit 110 en base 2 mais 1100.

Il faut donc traduire les nombres en bloc et c'est mal commode.

On peut traduire facilement en binaire toutes les bases qui sont une puissance entière de 2 :

2^1 = 2 pas de problème !

2^2 = 4 Trop voisin de 2, n'est pas utilisé

2^3 = 8 base octale, souvent utilisée, surtout aux USA (cf, fonctions OCT et DEC de la 41C)

2^4 = 16 base hexadécimale que nous utilisons constamment dans ce livre. Un chiffre hexadécimal est traduit exactement par 4 bits :

4 = 0100

6 = 0110

d'où 46 = 0100 0110 sans problème

Pour les chiffres supérieurs à 9 on utilise les lettres de l'alphabet de A à F. La ligne inférieure de la table des codes vous donne la correspondance entre hexadécimal et binaire. Nous appellerons digit un chiffre hexadécimal et octet (par référence au binaire) un groupe de 2 digits (8 bits). Nous mélangerons le décimal et les autres bases en utilisant uniquement le décimal pour dénombrer les éléments. Nous dirons "il y a 162 registres" mais "le registre A2". C'est dur, mais on s'y fait. Si vous avez des sous, offrez-vous une HP 16C, elle fera le travail pour vous...

8.2 Lexique

ROM Read Only Memory - mémoire morte. En général obtenue par des procédés photo, ne peut s'effacer. Les module HP sont des ROM.

RAM Random Acces Memory - mémoires vives. On y écrit, on les lit comme on veut.

EPROM Erasable, Programmable ROM. Circuit intégré courant dans le commerce, peut être effacé aux rayons ultraviolets et rempli avec un appareil spécial appelé programmeur d'EPROM. Dans le lecteur d'EPROM pour HP-41C il faut deux EPROMS au moins pour remplacer un module.

BIT Binary digit : élément binaire 0 ou 1 (voir 8.1 page précédente).

DIGIT Chiffre hexadécimal correspondant à un groupe de 4 bits.

OCTET 8 bits ou 2 digits.

MOT Dans les ROM HP ou EPROMS, les instructions du microcode utilisent| 10 bits, on parle alors de mot d'instruction.

^ OU ↑ symbole de l'élévation à une puissance en informatique ; sur la HP-41 peut indiquer la montée de la pile, avec ENTER ou R.

E précède l'exposant de 10 dans le registre alpha.

K d'habitude veut dire Kilo, mais en informatique on prononce "qua" et cela vaut 2^10=1024 éléments, octets, registres...

MANTISSE Un nombre est formé d'une mantisse et de son signe, d'un exposant et de son signe. La mantisse est la liste des chiffres du nombre. Cette appellation est tirée de l'étude des logarithmes

8.3 Le club PPC et ses publications

France PPC-Toulouse, principal club français, édite un Journal bimestriel en français contenant des articles originaux écrits par les adhérents du club (il ne s'agit pas d'une traduction du journal US). Le chapitre de Toulouse rassemble tous les francophones qui ne peuvent encore rejoindre un groupe local (environ 400 adhérents fin 83). PPC-T, 77 rue du CAGIRE 31100 Toulouse France tel. (61) 44 03 06. Chapitre de Paris : Philippe Guez, 56 rue JJ Rousseau 75001 Paris tel. (professionnel) (supprimé pour la version 2001 - n'existe plus).

Francophonie Lausanne Philippe Romascano Grand-Vennes 39, 1010 Lausanne

International La "maison mère" ; Personal Programming center journal de 32 pages mensuel en anglais, fait également une édition destinée au HP-75 et 80 édités par Richard Nelson 2545 V Camden place Santa Anna 92704 Californie USA

Australie Le chapitre PPC d'Australie publie PPC-TN (PPC-Technical Notes) journal en principe bimestriel mais parution très irrégulière, contenu très intéressant surtout consacré au microcode, c'est la première source à ce sujet. Édité par John McGechie, P0 BOX 512 Ringwood Victoria 3134 Australie (en anglais)

Grande Bretagne Le chapitre anglais publie un journal appelé DATAFILE en principe bimestriel, bien sûr en anglais. David N BURCH, ASTAGE Rectory Lane Windlesham, Surrey GU 206 Bv Royaume Uni

Allemagne Le club allemand ne se considère pas comme un chapitre de PPC, ce qui ne change pas grand chose, CCD publie un très beau journal mensuel en allemand. Contacter Andréas Marks- Cheffel, Nachtigallenweg 8 6246 Glasnutten 1 R F A.

Ci-dessus, je vous ai donné les principaux chapitres, ceux qui publient des journaux. Voici les adresses des autres (USA exclus, il sont trop nombreux).

Écrivez (en anglais, si vous ne connaissez pas la langue du pays) "PPC Chapter"...Peter Stanbrook 38 Bates Drive Thorneside, Qld 4158 AUSTRALIE. Jim Durtanovich P0 box C245 Clarence street Sydney, N s v 2000 AUSTRALIE. George Hathaway 85 Alcorn Av. Toronto, Ontario CANADA. Dr. P-G- Glockner 2536 Charlebois Dr Nv Calgary, Alberta T2L 0T6 Canada. Egon Jensen sp. Nollevej 62, EG 6700 Esbierg Danemark. Fernando Del Rey Poniente 25 2,A Madrid 16 ESPAGNE. Thomas Fange storangsgatan 24 413 19 Gotenborg SUÈDE. Paul Schlyter Nybrogatan 75 E s-l14 40 Stockholm Suède. Des Aubery 32 Joungfrau Naple Road Durban Natal AFRIQUE DU SUD. Harald Ommang Jac Saetresv 13 504ù Paradis NORVÈGE. liim Van Yperen pla Binnenwatersloot 30 2611 BK Delft PAYS BAS. Heinz-Jorg Plegge AH Roggen Kamp 20 D-4400 Nuenster RFA. Fleindert Kuipers Laan 2/10 9712 AV Groningen PAYS BAS HP-41 Club Franco Dal Flolin Plattenstr. 44, CH-8152 Glattbrugg, Suisse (en allemand). Tous ces groupes acceptent des adhésions extérieures, les contacter directement.

8.4 Les livres

En anglais :

Synthetic programming on the HP-41C par W.C. Wickes

Calculator Tips and Routines par John S Dearing

Curve Fitting par William M. Kolb

Synthetic programming made Easy par Keith Jarett

Extended function made Easy par Keith Jarett

An Easy course on programming the HP-41C par Chris Coffin et Ted Vadmann

Tous ces livres peuvent vous être fournis par les éditions du Cagire, soit sur stock, soit par commande aux USA.

En français :

Autour de la boucle (la boucle HP-IL) par Janick Taillandier (Editions du Cagire)

Programmer HP-41C par Philippe Descamps et Jean-Jacques Dhénin (Editions du PSI, en vente partout)

Calculs de béton armé sur HP-41 par L. Trione (Eyrolles)

Programmes d'acoustique sur HP-41 par P. Leys (Eyrolles)

Sur les HP-10,11, ... :

ENTER par Jean-Daniel Dodin (Editions du Cagire)

8.5 Matériel pour le microcode

SCIP (E. Poupée) 83 av. de Paris 94800 Villejuif (diffuse un lecteur d'Eprom de 16 Koctets et un simulateur de R0M).

Jacques Vaucelle "La Chasserie" 35133 Romagné (envisage de diffuser des simulateurs en kit).

Le Club PPC-T offre aussi des possibilités à ses adhérents.

Matériel étranger : Les Editions du Cagire peuvent vous fournir de nombreux matériels d'origine étrangère, si les fabrications françaises ne vous conviennent pas, détails sur demande.

8.6 Structure d'un jeu d'EPROM PPC

Structure définie par Jim De Arras

Une EPROM PPC, prête pour utilisation dans un lecteur d'EPROM (quel que soit le modèle) est composée en fait de deux EPROM au moins, baptisées U2 (pour Upper two, "les deux plus hauts") et L8 (pour Lover height, les huits plus bas).

L8 contient simplement les 8 bits de poids faibles (les plus à droite) du mot de même adresse du ROM que l'on veut construire.

U2 contient donc les 2 bits de poids forts (les plus à gauche). Ces deux bits, considérés comme une paire et lus dans l'ordre des adresses croissantes, sont rangés dans U2 de droite à gauche et de haut en bas. Exemple :

mots bits de droite

0xx 00

0xx 00

3xx 11

0xx 00

Sont rangés : 00 11 00 00 pour former un octet de U2, ici l'octet 30. De la même façon :

0003 donne 11 00 00 00 = C0

0300 donne 00 00 11 00 = OC

3000 donne 00 00 00 11 = 03

L'adresse 000 de U2 correspond aux adresses 000, 001, 002 et 003 de L8. On obtient donc l'adresse de l'octet précédemment reconstitué dans U2 en divisant l'adresse de L8 par 4 et en prenant la partie entière. Attention, vous travaillez en hexa. Si vous avez une HP-16C (veinard !), pas de problème, sinon utilisez un programme de changement de base (BD et TB du PPC ROM font très bien l'affaire).

Dans un jeu d'EPROM de 4K,avec une 2732 (4K octets) et une 2716 (2K octets) seul le premier ou le deuxième (au choix) Koctet de la 2716 est utilisé.

Je sais que tout cela n'est pas simple. Mais si vous venez à en avoir réellement besoin, vous aurez certainement en main un jeu d'EPROM "club" ou fourni par le fabriquant de votre lecteur d'EPROM et qui vous permettra d'expérimenter. Vous verrez que c'est passionnant.

8.7 Programme assembleur

Vous trouverez ci-joint figure 8.1un programme qui vous permettra de trouver le code 244 à partir de l'énoncé de l'instruction.

Ce programme vous permet d'écrire du microcode sans vous soucier des codes numériques en hexa, et ensuite d'établir les dits codes.

Le programme demande en entrée non pas les mnémoniques (qui sont d'ailleurs très désagréables à taper), mais le numéro d'ordre de la fonction, qui figure dans la première colonne du tableau (cf. Chap 6 page 43), et qui n'est autre que le code en décimal de l'instruction ou du paramètre.

Les tableaux particuliers (6/0, 8/0, C/0) donnant directement le code, l'assembleur ne s'en occupe pas.

Pour les instructions de type 0 (TO) et de type 2 (T2) entrer le numéro de l'instruction, ENTER, le numéro du champ, XEQ A pour TO ou XEQ C pour T2 vous obtenez à l'affichage le code.

Pour les instructions de type 1, placez en alpha l'adresse d'arrivée de la fonction et faites XEQ D pour ?NC XQ, XEQ E pour ?C XQ, XEQ F pour ?NC GO et XEQ G pour ?C GO.

Pour les instructions de type 3, mettez en x la valeur décimale du saut et XEQ H pour JNC ou XEQ I pour JC.

L'installation d'un overlay comme indiqué fig. 8.1 page suivante, permet avec les assignations par défaut des touches supérieures du clavier une utilisation commode et rapide (en mode USER).

Ce programme utilise deux routines du PPC ROM : QR (Quotient and remainder) qui donne M0D et quotient d'une division et BD (base to décimal) pour traduire l'adresse des sauts.

Pour ceux qui ne possèdent pas ce ROM j'ai reproduit les passages en question. BD utilise la mémoire 6 pour stocker la base, d'où la nécessité d'une SIZE 007.

La ligne 12 de BD est (HEXA) F3 7F 00 08, vous devez savoir tout faire.

8.8 Adressage ROM et RAM

par Didier Jehl (voir fig, 16, p. 62)

ROM, principe

1) La 41C envoie une adresse (A15...AO) sur la ligne ISA.

2) Chaque module ROM compare cette adresse avec celles qui lui sont assignées :
— s'il n'y a pas égalité, le module se déconnecte,
— s'il y a égalité, il prépare les données (DO à D9) correspondantes à l'adresse spécifiée (A15...AO) entre les phases 31 à 46, et les envoie en série à partir du front de montée du signal sync.

Les 2 types de modules HP

1. Module à adresse fixe (MAF) : Ce type de module aura vis à vis de la 41 une adresse fixe quelque soit le numéro du port dans lequel il est connecté (exemple : Horloge, imprimante).
2. Module à adresse variable (MAV) : ce type de module aura vis à vis de la 41 une adresse fixée par le numéro du port dans lequel il est connecté (exemple : math, PPC ROM,...).

adressage interne

T&C (Timing and Control) connecte la ligne ISA à l'interface d'entrée pendant les phases 16 à 31, permettant la comparaison des bits d'adresse (A15..A12) avec R2, R3, RI et R5. S'il y a égalité entre R2-R5 et A15-A12, un des blocs mémoire est sélectionné et délivre les 10 bits de données D0 à D9.

Fig. 8.1 – Programme assembleur

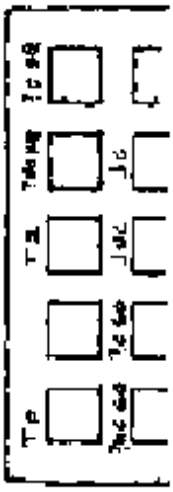
(a) Programme assembleur

```
01*LBL "BASS"
02*LBL L
03 SF 08
04*LBL M
05 CLA
06 WSB?
07 SF 01
08 RBS
09 63
10 X<Y?
11 GTO 19
12 RBM
13 J20
14 X<>Y
15 FS?C 01
16 -
17 32
18 XROM "QR"
19 XEQ IND Y
20 2
21 XROM "QR"
22 XEQ IND Y
23 2
24 *
25 L
26 X<>Y
27 FS?C 00
28 +
29 4
30 *
31 2
32 GTO 20
33*LBL D
34 XEQ 07
35 .
36 GTO 20
37*LBL E
38 XEQ 07
39 L
40 GTO 20
41*LBL F
42 XEQ 07
43 2
44 GTO 20
45*LBL G
46 XEQ 07
47 3
48 GTO 20
49*LBL 17
50 16
51 STO 06
52 XROM "BA"
53 256
54 XROM "QR"
55 XEQ L0
56 1
57 +
58 XEQ IND X
59 "+/-"
60 RBM
61 RBM
62 XEQ L8
63 RTN
64*LBL C
65 CLA
66 X<>Y
67 0
68 XROM "QR"
69 XEQ IND Y
70 X<>Y
71 RBM
72 8
73 *
74 +
75 4
76 XROM "QR"
77 XEQ IND Y
78 4
79 *
80 2
81 GTO 20
82*LBL A
83 CLA
84 XEQ L6
85 J
86*LBL 20
87 +
88 XEQ IND X
89 RNORM
90 STOP
91*LBL 08
92 16
93 XROM "QR"
94 X<>Y
95*LBL 16
96 4
97 XROM "QR"
98 XEQ IND Y
99 X<>Y
100 RBM
101 16
102 *
103 +
104 4
105 XROM "QR"
106 XEQ IND Y
107 +
108 *
109 RTN
110*LBL 00
111 "+00"
112 RTN
113*LBL 01
114 "+1"
115 RTN
116*LBL 02
117 "+2"
118 RTN
119*LBL 03
120 "+3"
121 RTN
122*LBL 04
123 "+4"
124 RTN
125*LBL 05
126 "+5"
127 RTN
128*LBL 06
129 "+6"
130 RTN
131*LBL 07
132 "+7"
133 RTN
134*LBL 08
135 "+8"
136 RTN
137*LBL 09
138 "+9"
139 RTN
140*LBL 10
141 "+A"
142 RTN
143*LBL 11
144 "+B"
145 RTN
146*LBL 12
147 "+C"
148 RTN
149*LBL 13
150 "+D"
151 RTN
152*LBL 14
153 "+E"
154 RTN
155*LBL 15
156 "+F"
157 .END.
```

```
01*LBL "BD"
02*LBL A
03 CLST
04*LBL 01
05 "I "
06 X<> J
07 X=0?
08 GTO 01
09 X<> I
10 RT
11 X<> K
12 "I+J"
13 X<> K
14 RBM
15 X<> I
16 E
17 *
18 39
19
20 X)0?
```

```
21 BASE X
22 9
23 +
24 X<0?
25 GTO 02
26 X<>Y
27 RCL 06
28 *
29 +
30 ,
31 GTO 01
32*LBL 02
33 RBM
34 CLA
35 RTN
```

```
01*LBL "QR"
02 X<>Y
03 STO J
04 X<>Y
05 RND
06 ST- J
07 LOG10
08 ST/ J
09 CLX
10 X<> J
11 X<>Y
12 RTN
13 END
```



Pour utiliser BD et QR, taper XEQ"BD" ou XEQ"QR". Si ces programmes sont trouvés dans le PPC ROM, la HP transformera elle même le XEQ en XROM"" comme dans le listage, sinon elle laissera les XEQ.

Assembleur

T&C détecte le front de montée de SYNC, connecte alors ISA avec l'interface de sortie et envoie les données D0-D9 jusqu'au front de descente de SYNC.

RAM, principe

1. La 41 envoie sur ISA l'instruction 270 (RAM SLCT) afin de prévenir les registres qu'elle va en sélectionner un.
2. Au début du cycle suivant, la ligne DATA envoie10 bits d'adresse. Le registre dont l'adresse correspond est alors sélectionné et le reste jusqu'à ce qu'un autre soit sélectionné.
3. Pour écrire dans le registre sélectionné, HP envoie sur la ligne ISA l'instruction 2F0 (WRITE DATA) et au début du cycle suivant les 56 bits présents sur DATA sont écrits dans le registre sélectionné. Ces 56 bits sont ceux qui sont contenus dans le registre C du microprocesseur.
4. Pour lire le registre sélectionné, HP envoie sur la ligne ISA l'instruction 038 READ 0(T), et au début du cycle suivant, les 56 bits du registre sélectionné sont envoyés sur DATA vers le registre C du micro.

adressage interne

Pour les modules simples, B3 et B4 (dont la valeur dépend uniquement du port) font partie de la comparaison des adresses. Pour le Quad memory, B3 et B4 = 0 (en l'air).

Un comparateur relié à ISA permet de détecter les différentes instructions 270, 2F0, 038. Lorsque 270 est détecté, TBC (Timing et Control) positionne le buffer en entrée et commande, au début du cycle suivant, le circuit latch (verrouillage) dés que 10 bits d'adresse présents sur DATA sont entrés dans latch. un comparateur compare (!), les 6 bits de poids fort avec les adresses assignées à chaque chip ; le chip dont l'adresse correspond est sélectionné et sélectionne à son tour un registre (parmi les 16 qu'il contient) à l'aide des 4 bits d'adresse de poids faible.

T&C attend que l'instruction 2F0 (ou 038) soit détectée pour commander l'écriture (ou la lecture) du registre sélectionné, et envoyer dans le registre (ou sur DATA) les 56 bits de DATA (ou du registre) à partir du cycle suivant l'instruction 2F0 (ou 038).

8.9 Les alarmes (module HP82182A)

Commençons l'étude à partir du bas de la mémoire et en remontant, c'est comme ça que la HP pratique. Suivons sur un exemple : le 25 Octobre 1983, à 12h, faisons afficher "Au fond de la HP-41C". Introduisons ce texte en ALPHA, 0 ENTER 10,251983 ENTER 12 XEQ "XYZALM".

Nous avons d'abord un registre d'état qui commence par les digits hexa AA, suivis d'un octet donnant le nombre total de registres de l'alarme, les digits suivants sont inutilisés. Avec notre exempte, AA 06 00 00 00 00 00.

Le registre suivant (au dessus), contient :

-dans les 11 digits de gauche et en 1/10éme de secondes, l'écart entre la date de l'alarme et le 1er Janvier 1900,

-dans le digit suivant (numéro 2), l'indication d'existence d'une répétition (si digit à 1),

-dans le digit suivant (numéro 1), l'indication d'une alarme "past due", c'est à dire passée sans avoir été déclenchée (digit à F).

-Le dernier digit (numéro 0), indique le nombre de registres occupés par le message alphanumérique (de 0 à 4), ce qui donne pour nous :

:_2_ :_6_ :_4_ :_4_ :_9_ :_2_ :_0_ :_0_ :_0_ :_0_ :_0_ :_0_ :_0_ :_3_ :

Ce message est situé à partir des registres suivants, lus de gauche à droite et de bas en haut

00 41 55 20 46 4E 4F
A U F 0 N
44 20 44 45 20 4C 41
D D E L A
20 48 50 2D 34 31 43
H P - 4 1 C

Le cas échéant, entre le registre donnant la date de l'alarme et le message, figure, dans un registre spécial, l'intervalle de répétition en secondes (digits 5 à 11 ; n'existe pas dans notre exemple).

Une nouvelle alarme est située immédiatement au dessus de la précédente.

Au dessus de la dernière alarme, se trouve un registre commençant par le code hexa F0, et vide pour le reste : F0 00 00.....

Vous pouvez utiliser la méthode décrite dans la construction du Byte Grabber pour pénétrer ces registres. Ils sont alors lus comme un programme, il faut traduire, bonne gymnastique.

8.10 Commentaire du schéma du microprocesseur de La HP-41C (cf, fig, 16, p. 62)

Les numéros () représentent les numéros de broche du microprocesseur.

Terme	Sens
KC	Key Column : numéro de colonne du clavier
KR	Key Row : numéro de rang du clavier
POR	détection touche ON utilisée (mise en ou hors sommeil profond)
xl, x2	connexions du circuit oscillateur
01	signal d'horloge pour lecture de données
02	signal d'horloge pour envoyer les données et calculer
SYNC	synchronisation
PWO	Power On : microprocesseur actif
DPWO	(Display Power On ?) commande L'affichage
VCI, VCO, LLD	commande de différents modes (sommeil profond, actif, sommeil léger)
ISA	Instructions et adresses ROM
FO	Flag output (T) vers bipper
FI	Flag In : détection de périphérique non synchronisé
DATA	transfert données, adresses RAM

]

]

RUN donne le solution ; le code de contrôle est donné par le deuxième caractère imprimé, à condition de le chercher dans la deuxième moitié de la table. Après %, n'importe quel caractère qui ne soit pas caractère de contrôle convient (ici caractère A2).

8.11 L'affichage de la HP-41C

Ces instructions ont été détaillées par Paul Lind et publiées pour la première fois dans le journal australien PPC-TN.

Chaque caractère est envoyé à l'écran sous forme d'une chaîne de 9 bits : :8 :7 :6 :5 :4 :3 :2 :1 :0 :

Le bit numéro 8 (à 1) indique le rang 4 de la table (fig. 8, p. 24) si les bits 0-5 donnent un nombre de 0 à F. Pour toute autre valeur des bits 0-5 l'écran montre un espace. Ce bit n'a pas d'effet sur la ponctuation.

Les bits 6 et 7 indiquent la ponctuation : pas de ponctuation : 00, un point (.) : 01, deux points (:) : 10 ou la virgule (,) : 11.

Les informations peuvent être envoyées à l'affichage par 1, 4, 8 ou 9 bits, caractère par caractère ou en bloc.

Voici l'effet des instructions :

Transfert par 9 bits : Les WRIT poussent un caractère dans l'affichage, il y a donc perte d'un caractère de l'autre côté. Tous les READ provoquent une permutation circulaire, le caractère mis en C n'étant pas effacé de l'affichage mais transporté de l'autre coté.

WRIT 14 (d) prend un caractère en C S&X et le pousse à gauche de l'écran

READ 14 (d) lit le caractère à droite de l'écran en C S&X

WRIT 15 (e) prend un caractère en C S&X et le pousse à droite de l'écran

READ 15 (e) lit un caractère à gauche de l'écran

WRIT 4 (L) pousse 4 caractères de C à gauche de l'écran

READ 4 (L) lit 4 caractères à gauche de l'écran

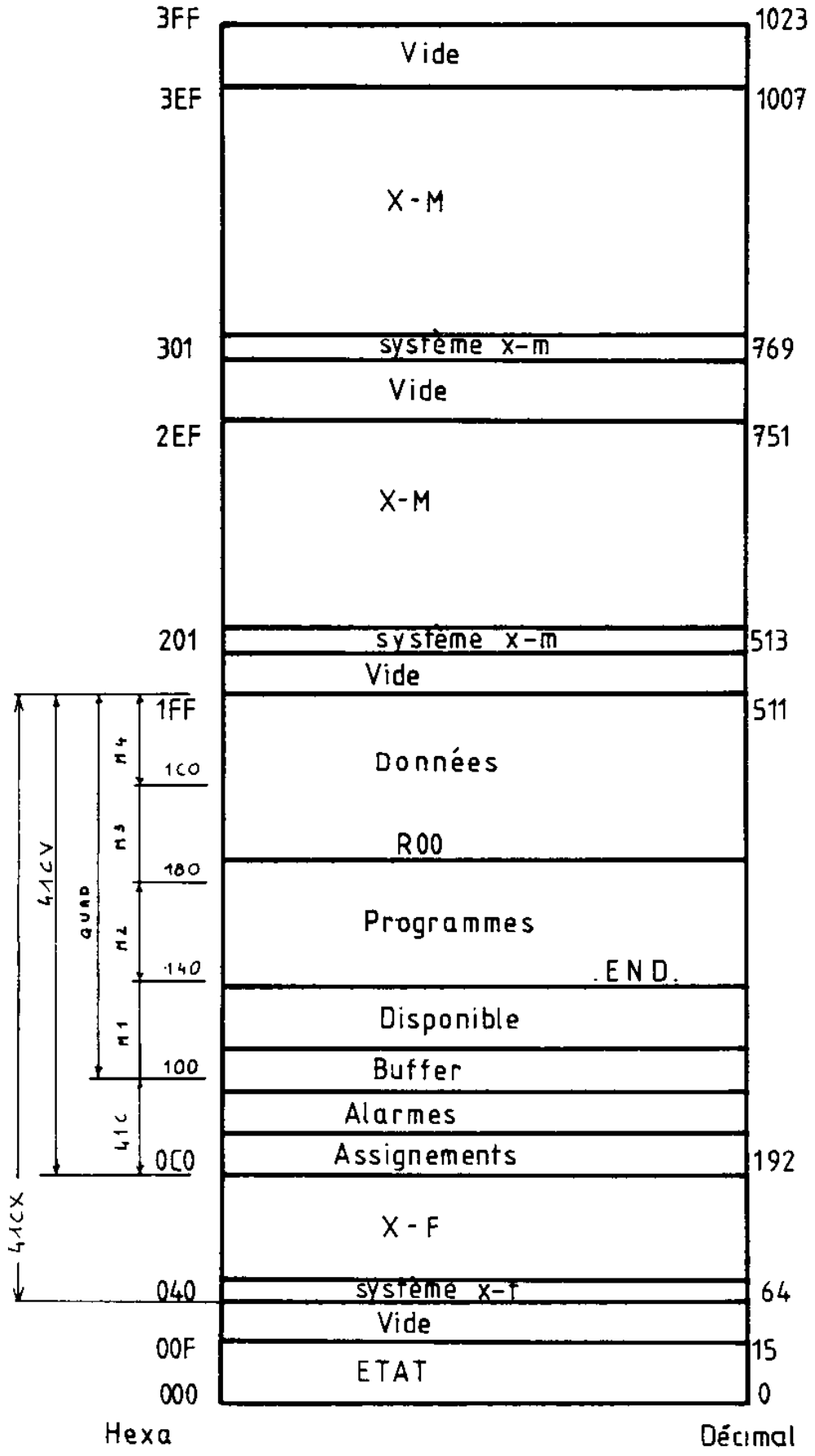
WRIT 6 (N) pousse 4 caractères à droite

READ 6 (N) lit 4 caractères à droite

Les instructions suivantes ne traitent que 8 bits. A l'écran *seuls les éléments considérés sont modifiés.*

WRIT 12 (b) pousse un caractère de XP de C à gauche de L'écran

FIG. 8.2 – *af109*


3FF
1023
Vide
3EF
1007
X-M
301
769
système x-m
Vide
2EF
751
X-M
201
513
système x-m
Vide
1FF
511
Données
R00
Programmes
.END.
Disponible
Buffer
Alarmes
Assignements
192
X-F
040
64
système x-f
Vide
00F
15
ETAT
000
0
Hexa
Décimal
41CV
QUAD
N4
N3
N2
N1
41C
41CX
100
180
140
100
0C0

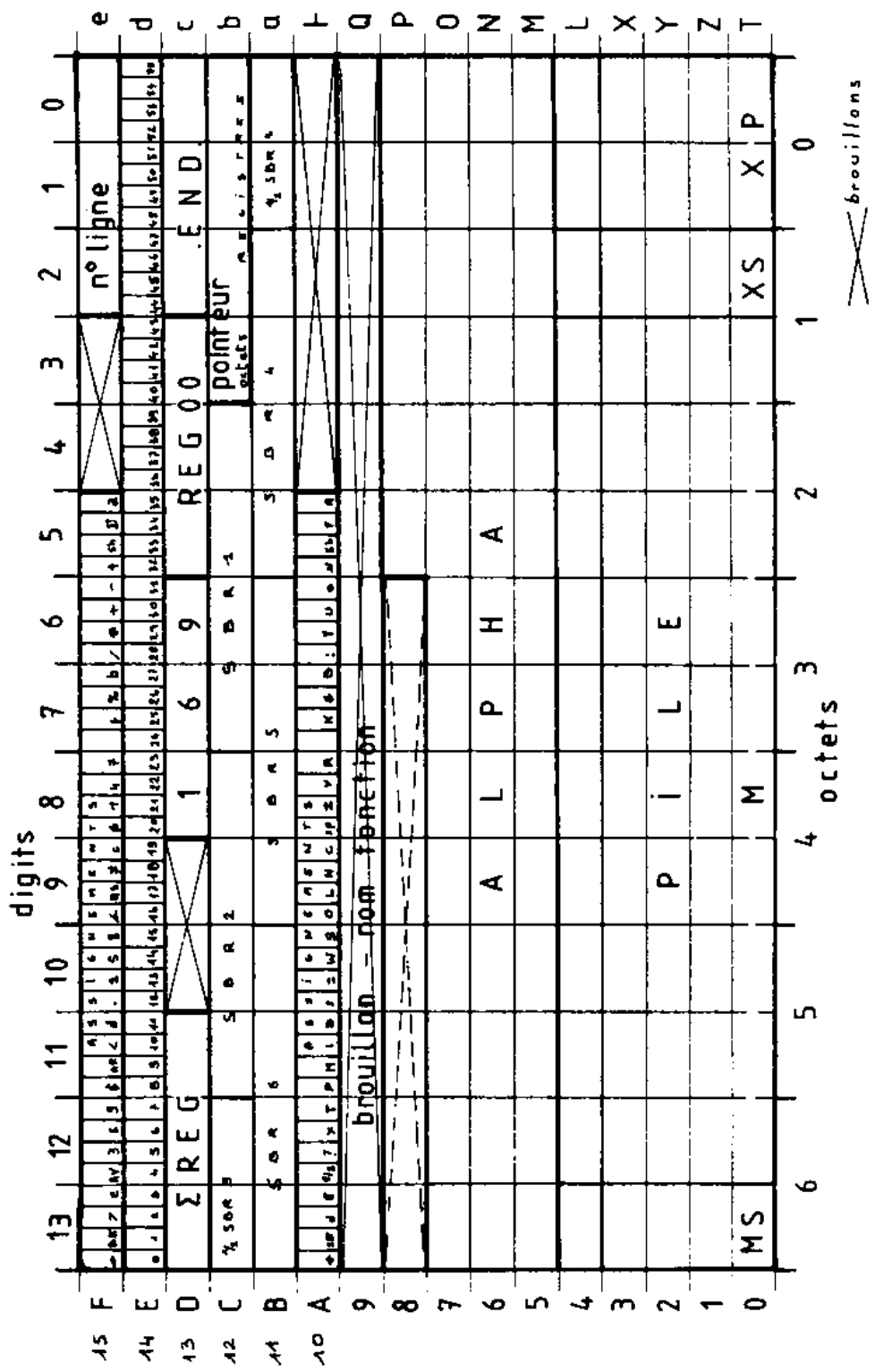


FIG. 8.3 – Utilisation des registres

READ 12 (b) lit les 8 bits inférieurs du caractère de gauche

WRIT 13 (c) pousse le caractère à droite

READ 13 (c) lit le caractère à droite

WRIT 3 (X) pousse 6 caractères à gauche

READ 3 (X) lit les 6 caractères de gauche

WRIT 5 (M) pousse 6 caractères à droite

READ 5 (M) fonction mal connue

Les instructions suivantes ne traitent que 4 bits ; haut (4 à 7) ou bas (0 à 3), par 1 ou 12 caractères.

WRIT pousse et READ lit :

instruction	haut	bas	1	12	gauche	droite
WRIT 7(0)		x	x		x	
READ 7(0)		x	x			x
WRIT 10(⊢)		x	x			x
READ 10(⊢)		x	x		x	
WRIT 10(⊤)		x		x	x	
READ 10(⊤)		x		x		x
WRIT 8(P)	x		x		x	
READ 8(P)	x		x			x
WRIT 11(a)	x		x			x
READ 11(a)	x		x		x	
WRIT 1(Z)	x		x	x		
READ 1(Z)	x			x		x

NOTA à l'édition 2001 : le WRIT 1(Z) me parait curieux (c'est ce qui figure dans l'addition originale, cependant). Peut-être y a-t-il une erreur (1 au lieu de gauche ?)

Les instructions suivantes ne concernent que le bit de gauche, elles servent en même temps que les instructions sur 4 bits.

WRIT 9(Q) comme WRIT 7(0) (mais pour bit 8)

READ 9(Q) comme READ 7(0) "

WRIT 2(Y) comme WRIT 0(T) "

READ 2(Y) comme READ 0(T) "

Les deux instructions READ ont un drôle de résultat.

Indicateurs :

Ils sont écrits-avec WRITE DATA et lus par READ 5(M), le tout avec C S&X :

bit	indicateur
0	ALPHA
1	PRGM
2	Flag 4
3	Flag 3
4	Flag 2
5	Flag 1
6	Flag 0
7	Shift
8	RAD
9	G (GRAD)
10	USER
11	BAT

8.12 Sélection des registres par RAM SELECT

RAM SELECT sélectionne Le bloc de 16 registres qui contient le registre de numéro donné. L'adresse est donc prise modulo 16 : 3=0, 18=16,... 3 RAM SELECT sélectionne les registres d'état, mais 0(T) est toujours le registre 0, pas le registre -3 ! On trouve dans le code des modules internes de la HP-41C, à partir de l'adresse 03F5, la suite d'instructions :

LDI

2FD

RAMSLCT

PRPHSLCT

READ(Y)

?NCGO

0205 MEMCHEK

Ces codes n'ont pas une grande signification, il s'agit d'une modification intervenue en cours d'étude de la machine, pour remédier à un problème de synchronisation entre les circuits de gestion de l'affichage. Mais HP utilise une astuce pour gagner de la place qui consiste à prendre la même constante pour sélectionner l'affichage et pour invalider la mémoire vive. Il faut savoir, en effet, que lors d'une opération READ ou WRITE, le microprocesseur de la 41C envoie toujours le contenu de C vers les mémoires vives, même si un autre périphérique a été sélectionné. Si, quand vous utilisez l'affichage, vous ne prêtez pas attention à cela, chaque message destiné à l'affichage envoie dans la mémoire précédemment sélectionné le contenu de C. On résout le problème en sélectionnant une partie de mémoire où il n'y a que du vide. Mais on rend ainsi inutilisable 16 emplacements de registres. Usuellement ce sont les registres 010 à 01F qui jouent ce rôle, mais l'astuce du programme ci-dessus fait également jouer ce rôle aux registres 2F0 à 2FF. Je ne sais pas si c'est la seule raison, mais quand HP a construit les modules d'X-mémoire, il a du coup été obligé de laisser vide ces 16 registres ! Tout ça pour gagner deux pas (LDI 010 avant RAMSLCT et report de LDI 2FD avant PRPHSLCT.

8.13 Liste des membres de PPC dont Les travaux ont été importants pour ce livre, (numéro PPC)

Richard J. Nelson (1)

Cary Reinstein (2046)

John Dearing (2791)

John McGechie (3324)

Bill Wickes (3735)

Keith Jarett (4660)

Jim De Arras (4706)

Valentin Albillo (4747)

Steven Jacobs (5358)

Paul Lind (6157)

Didier Jehl (8116 T80)

Lionel Ancelet (chapitre de Paris)

Lynn Vilkyn

Stéphane Barisien

et l'auteur

J. D. Dodin (7226 T1)

8.14 4ème de couverture :

Depuis la première édition de cet ouvrage, fin 1982, beaucoup d'évènements se sont produits. La connaissance des utilisateurs sur le microcode s'est approfondie et les appareils nécessaires sont apparus sur le marché.

Il est maintenant possible pour tous de pratiquer cette programmation, pour un prix de l'ordre de celui du lecteur de cartes.

En feuilletant ce Livre vous découvrirez bien des recoins cachés de votre calculatrice, et vous serez sans doute étonné de tout ce qu'elle contient.

Non seulement le microcode, mais la programmation synthétique, les registres d'état, le BG, feront maintenant partie de votre vocabulaire.

En route pour l'exploration...

Table des figures

Liste des tableaux

Index

Table des matières