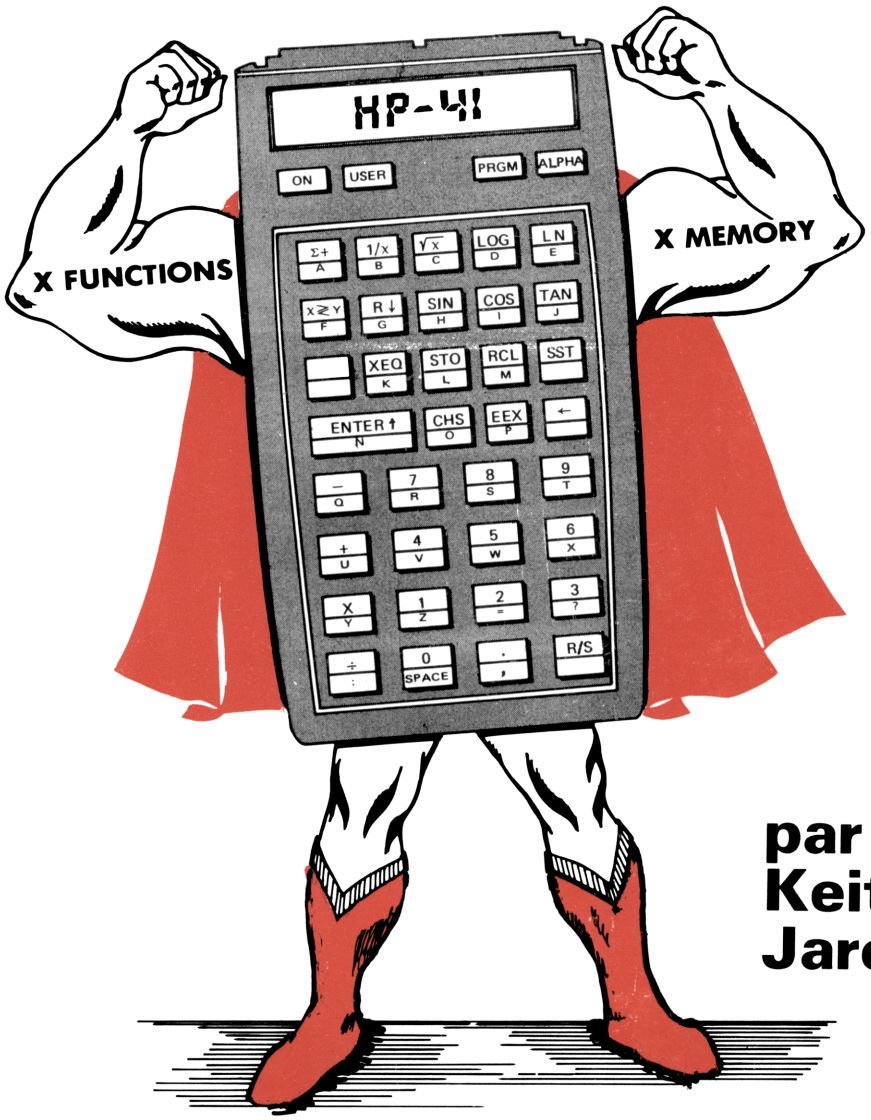
HP-41
LES FONCTIONS D'EXTENSION, C'est facile !



par
Keith
Jarett

module de fonctions d'extension mémoire pour HP-41C et CV ; HP-41CX

Editions du CAGIRE

HP-41 LES FONCTIONS D'EXTENSION, C'est facile !

module de fonctions d'extension mémoire pour HP-41C et CV ; HP-41CX

par Keith Jarett

traduit de l'américain par M.-D. Dodin

77 rue du Cagire
31100 Toulouse
France
ISBN 2-86811-008-8

Préface

DEDICACE

Ce livre est dédié à ma femme, Catherine Van de Rostyne, qui n'aurait pas pu être plus utile et plus encourageante.

REMERCIEMENTS

Les contributions les plus importantes à ce livre ont été faites par Clifford Stern, qui en a été, tout au long de son développement, le conseiller technique. Clifford fait partie de la poignée de "grands maitres" de la programmation de la HP-41. Il est probablement plus au fait des subtilités du système d'exploitation de la HP-41 que toute autre personne. Clifford a écrit la plupart des programmes utilitaires trés avancés du chapitre 10, procédé à la recherche d'éventuelles erreurs techniques dans ce livre, et fourni d'importants conseils pour son amélioration.

Les autres participants à ce livre sont Erik Christensen, qui a écrit le programme de formattage de texte et la documentation qui apparait dans le chapitre 8, Tapani Tarvainen et Gérard Westen, qui ont écrit le stupéfiant programme d'assignement du chapitre 10, et Alan McCornack, qui a écrit le programme de liste d'adresses du chapitre 7 et fourni beaucoup de suggestions trés utiles pendant la correction.

AU SUJET DE L'AUTEUR

Keith Jarett s'est adonné aux calculatrices Hewlett-Packard depuis qu'il acheta une HP-45 en 1973 et écrivit des "programmes", séquences de touches, pour elle. En 1980 et 1981, il coordonna le développement de 67 routines synthétiques pour le PPC ROM (voir annexe C), un module de programmes écrit par et pour les utilisateurs de la HP-41C.

Il est actuellement Ingénieur Système dans le groupe espace et communication de la société Hughes Aircraft. Il est diplômé de l'académie militaire Culver (1972), diplômé en génie électrique de l'université de Cornell en 1975 et de l'université de Stanford en 1976 et 1979.

POUR UNE INFORMATION SUR LE PRIX de ce livre, écrire à "Editions du Cagire, 77 rue du Cagire, 31100 Toulouse France". Remise aux distributeurs.

TABLE DES MATIERES

I N T R O D U C T I O N

Le module de fonctions d'extension mémoire, incorporé dans la nouvelle HP-41CX et disponible séparément pour les HP-41C et CV, procure beaucoup de nouvelles fonctions pour votre HP-41 (1). Il procure également 127 registres supplémentaires de mémoire. Jusqu'à deux modules de mémoire d'extension peuvent être ajoutés, chacun d'entre eux contenant 238 registres. Aussi, selon le nombre de modules d'extension que vous branchez (2) pour accompagner votre module de fonctions d'extension mémoire, vous avez entre 127 et 603 registres de mémoire d'extension disponibles.

La mémoire d'extension est un exemple de stockage "hors ligne". Les programmes en mémoire d'extension ne sont pas directement utilisables ; ils doivent d'abord être rappelés en mémoire principale, dite "en ligne". Une fois en mémoire principale, les programmes peuvent être modifiés ou exécutés à volonté. Des données peuvent également être stockées dans les mémoires d'extension.

De ce point de vue, le fonctionnement des mémoires d'extension est semblable à celui du lecteur de cartes HP82104A. Le lecteur de cartes a également la possibilité de sauver des programmes et des données à partir de la mémoire principale. Les différences principales entre la mémoire d'extension et le lecteur de cartes sont :

lecteur de cartes	mémoire d'extension
capacité illimitée	capacitée limitée (127 à 603 registres)
opérations manuelles	au clavier ou contrôle par programme
stockage permanent	stockage à court ou moyen terme (risque de MEMORY LOST)
Sauvegarde (WRTA), état, données	Programmes, données, fichiers

La mémoire d'extension équivalente à une carte magnétique est appelée un fichier. De la même façon qu'il y a différents formats pour les cartes magnétiques (cartes de programmes, de données, etc...), il y a différents types de fichiers en mémoire d'extension. Les trois types de fichiers sont programme, données et texte, aussi appelés ASCII (3).

Un fichier programme, comme son nom l'indique, contient un programme. Un fichier de données contient un ou plusieurs registres de données. Un fichier texte, ou ASCII, contient une série de caractères alphanumériques appelée chaine de caractères. Ces types de fichiers seront abordés et expliqués dans les trois premiers chapitres où des exemples de leur utilisation seront donnés.

---

(1) Le module de fonctions d'extension mémoire fournit 47 fonctions. La HP-41CX contient ces 47 fonctions plus 14 autres, pour un total de 61 fonctions d'extension.

(2) Ne branchez pas deux modules d'extension mémoire l'un au-dessus de l'autre. Le module de fonctions d'extension mémoire peut être branché n'importe où. Reportez-vous à la section 1 du manuel de votre module de fonctions d'extension mémoire pour des détails sur les configurations autorisées. Si vous n'avez qu'un module de mémoire d'extension, vous devez le brancher dans le port 1 ou le port 3 (en haut à gauche ou en bas à gauche,

vu de l'arrière de la calculatrice).

(3) American Standard Code for Information Interchange : code américain normalisé pour l'échange d'informations ; un système pour exprimer les caractères en valeurs de 7 éléments binaires. Chaque caractère utilise un octet (1/7 de registre) dans la HP-41.

" P A R A S I T E S "

Un "parasite" est défini comme une caractéristique d'une fonction (ou d'une machine) qui est soit indésirable soit inattendue, soit différente de ce qui est décrit dans le mode d'emploi. Du fait de l'incroyable complexité de la programmation interne dans le module de fonctions d'extension mémoire, certains parasites ont réussi à persister malgré tous les efforts de Hewlett-Packard. Dans certaines circonstances peu usuelles, certaines des X-fonctions peuvent avoir un effet désagréable, allant jusqu'au MEMORY LOST.

Si vous avez ou un module de fonctions d'extension mémoire ou un lecteur de cartes fabriqué avant Septembre 1983, vous devez lire ce chapitre avant d'aller plus loin. Sinon votre système HP-41 est pratiquement dépourvu de parasites, et vous pouvez pour l'instant sauter cette discussion.

Ce livre donne tous les détails et procédés nécessaires pour éviter les problèmes qui peuvent se poser avec les parasites du module de fonctions d'extension que vous pouvez avoir. Dans certains cas, vous pouvez même réparer après coup les dommages causés par le parasite. L'objet du présent résumé est de vous donner assez d'informations pour vous permettre d'éviter les ennuis jusqu'à ce que vous ayez pénétré plus avant dans ce livre, où les techniques de prévention et de correction sont expliquées en détail.

Vous ne devez pas reprocher à HP l'existence des parasites. Ils passent beaucoup de temps à tester leurs productions avant de les mettre dans le commerce, mais aucune série de test ne peut couvrir toutes les conditions de travail. Vient un moment où les essais doivent cesser, et la production commencer. Si vous voulez la perfection, vous devrez attendre longtemps.

Hewlett-Packard produit des calculatrices et des modules avec moins de problèmes que les autres fabriquants. Que quelques parasites subsistent est simplement le prix d'une production qui est en tête de la concurrence en performances et valeur.

Il y a trois parasites dans les premières versions du module de fonctions d'extension mémoire. Les techniques présentées dans ce livre éliminent tout problème avec ces parasites. Le plus souvent une procédure de prévention est décrite, mais une procédure de correction est aussi possible pour les parasites les plus fréquents.

Le premier parasite du module de fonctions d'extension mémoire est celui de SAVEP (save program : sauver un programme) et PCLPS (programmable clear programs : effacement programmable des programmes). Ces fonctions ne doivent pas être exécutées si la calculatrice est positionnée dans un programme d'un module d'application, en dehors de la mémoire principale. Ceci est étudié en détails pages 13 et 14. Si vous utilisez le programme "XF" (§10B) pour utiliser les X-fonctions à partir du clavier, ce problème sera automatiquement évité.

Le second parasite est celui de PURFL (purge file ; effacer un fichier). Cette fonction crée une situation dangereuse dans laquelle une fausse manoeuvre peut faire perdre l'accés à toute la mémoire d'extension (cf. p. 77). Heureusement, les techniques de programmation synthétiques (§10E) peuvent être utilisées pour réparer les dommages. D'autres mesures simples peuvent empêcher ce problème de prendre une trop grande place. Pour

plus d'informations sur PURFL, confert pages 17. Si vous n'avez jamais entendu parler de programmation synthétique, le §10A donne une brève description de ce qu'est la programmation synthétique et comment vous pouvez en apprendre plus à son sujet.

Le troisième parasite est celui des fonctions VER et 7CLREG du lecteur de cartes. Ces fonctions peuvent modifier le contenu de la mémoire d'extension. Tant que vous n'avez pas lu les détails donnés dans l'Annexe A, n'utilisez pas VER quand un module de mémoire d'extension est présent dans les ports 2 ou 4 ou dans un module combiné. Le module de fonctions d'extension mémoire peut être branché n'importe où sans risque. Un court programme synthétique §10D vous permettra d'utiliser VER sans abimer la mémoire d'extension.

Les deux premiers parasites ci-dessus n'apparaissent que dans la révision 1B du module de fonctions d'extension mémoire. Vous pouvez voir quelle révision vous avez en utilisant le catalogue 2 (tapez shift CATALOG 2). Quelque part dans la liste des fonctions des périphériques doit apparaitre le message :

-EXT FCN 1B
-EXT FCN 1C
ou -EXT FCN 2C (HP-41CX seulement)

Si le message passe trop vite, vous pouvez presser R/S pour interrompre le listage, puis utiliser BST pour le retrouver. Les révisions 1C et plus récentes sont exemptes des parasites de SAVEP/PCLPS et PURFL.

Le troisième parasite est dû au fonctionnement du lecteur de cartes. Il n'est corrigé que dans les plus récents lecteurs de cartes. Contrôlez votre lecteur de cartes à l'aide du catalogue 2. Si vous voyez :

CARD READER
CARD RDR 1D
CARD RDR 1E
ou CARD RDR 1F

Alors votre lecteur de cartes a le parasite VER. Si vous avez une révision 1G ou au dessus, votre lecteur de cartes est exempt de ce parasite.

Remarquez que la HP-41CX n'a aucun de ces parasites dans ses X-fonctions. Mais si vous avez un vieux lecteur de cartes, vous aurez toujours à éviter la fonction VER tant que vous n'aurez pas lû le §10D

Maintenant voyons ce qu'est la mémoire d'extension.

C H A P I T R E I

SAUVER DES PROGRAMMES DANS LA MEMOIRE D'EXTENSION

1A. Créer un fichier programme

Les opérations les plus utilisées sur la mémoire d'extension sont SAVEP et GETP. Ces opérations sauvent un programme dans la mémoire d'extension et le récupèrent de la mémoire d'extension. Pour illustrer ces fonctions, allumez votre HP-41, faites GTO.. et tapez le programme ci-contre.

Si vous voyez le message PACKING suivi de TRY AGAIN, vous devez réduire la taille (SIZE) pour rendre plus de registres disponibles pour les programmes. Une autre solution est d'utiliser la fonction CLP pour effacer un ou plusieurs programmes (choisissez ceux qui sont sacrifiables ou que vous avez déja sauvés sur bande ou carte) pour faire de la place.

Quand vous avez fini de taper le programme, faites GTO.. pour le compacter et lui attacher un END. Ceci est important pour réduire la place nécessaire pour stocker ce programme dans la mémoire d'extension. Ce que vous pouvez également faire avant de sauver le programme est d'exécuter chaque GTO (lignes 42 et 50) au moins une fois. Ceci est expliqué en détail page 132.

Pour exécuter ces lignes, passer en mode calcul (non programme) et pressez GTO.042. Ensuite pressez SST et maintenez la touche jusqu'à voir apparaitre la ligne de programme à l'affichage, puis relachez la touche. Quand l'affichage s'efface, vous savez que la ligne 42 a été exécutée. Pressez alors GTO.050, SST jusqu'a voir apparaitre la ligne 50, puis relachez. Le programme est maintenant prêt à être sauvé en mémoire d'extension. La procédure pour sauver un programme sera décrite dans le §C de ce chapitre.

Description du programme "JNX"

Le programme "JNX" calcule la fonction de Bessel de premier type d'ordre entier, $J_n(x)$, avec une précision de 8 chiffres. Ce programme peut vous être utile ou non, mais il est un bon exemple de la puissance de la HP-41. Le programme "JNX" de calcul de la fonction de Bessel sera utilisé au chapitre 6, aussi vous voudrez peut-être le sauver sur une carte magnétique ou une cassette avant de poursuivre. Pour contrôler si votre version de "JNX" fonctionne correctement, essayez :

2 ENTER↑ 1,2 XEQ"JNX"

pour calculer $J_2(1,2)$. Le résultat doit être $1,593490184 \times 10^{-1}$

Vous pouvez maintenant sauter au début du § suivant (page 11), à moins que vous ne soyez particulièrement intéressé par les fonctions de Bessel. La discussion détaillée qui suit décrit exactement le fonctionnement de "JNX".

Analyse ligne à ligne de "JNX"

L'algorithme utilisé par "JNX" est basé sur la relation de récurrence :

$$J_{i-1}(x) = (2m/x)J_i(x) - J_{i+1}(x)$$

```
01♦LBL "JNX"
 02 STO 03
 03 ABS
 04 5
 05 +
 06 X<>Y
 07 STO 00
 08 X<Y?
 09 X<>Y
 10 INT
 11 4
 12 *
 13 STO 07
 14 1/X
 15 STO 02
 16 STO 04
 17 STO 05
18♦LBL 00
 19 RCL 05
 20 CHS
 21 RCL 07
 22 RCL 03
 23 /
 24 RCL 04
 25 STO 05
 26 *
 27 +
 28 STO 04
 29 RCL 07
 30 4
 31 /
 32 FRC
 33 X≠0?
 34 /
 35 ST+ 02
 36 RCL 00
 37 2
 38 ST- 07
 39 *
 40 RCL 07
 41 X≠Y?
 42 GTO 01
 43 RCL 04
 44 STO 01
 45 RCL 05
 46 STO 06
 47 RCL 07
48♦LBL 01
 49 X≠0?
 50 GTO 00
 51 RCL 04
 52 ST- 02
 53 RCL 01
 54 RCL 02
 55 ST/ 01
 56 ST/ 04
 57 ST/ 05
 58 ST/ 06
 59 RCL 05
 60 RCL 04
 61 RCL 06
 62 RCL 01
 63 END
                80 BYTES
```

L'exécution commence avec les estimations initiales :

$$J_m(x) = J_{m+1}(x) = 0, \text{ où } m = 2*INT(max(n,x+5))$$

La relation de récurrence est évaluée par valeurs décroissantes de n, jusqu'à n=0. Pendant ce calcul, la somme :

$$J_0(x)+2J_2(x)+2J_4(x)+2J_6(x)+...$$

est évaluée. Comme cette somme est théoriquement égale à 1, elle est utilisée pour normaliser la précédente valeur calculée de $J_n(x)$.

Maintenant les particularités de ce programme. Les registres utilisés par "JNX" sont :

registres	contenus
00	n
01	$J_n(x)$
02	somme de normalisation
03	x
04	$J_i(x)$ en commençant par $\frac{1}{2}m$
05	$J_{i+1}(x)$ en commençant par $\frac{1}{2}m$
06	$J_{n+1}(x)$
07	2i en commençant par 2m

Les lignes 01-17 de "JNX" initialisent les registres de données. La boucle LBL 00 calcule $J_{i-1}(x)$ à partir des estimations précédentes $J_i(x)$ et $J_{i+1}(x)$. Cette nouvelle estimation remplace l'ancien $J_i(x)$ (ligne 28), tandis que $J_i(x)$ remplace l'ancien $J_{i+1}(x)$ (lignes 24 et 25).

Les lignes 30-35 ajoutent $2J_{i-1}(x)$ à la somme de normalisation, seulement si i est impair (c'est à dire si la partie fractionnaire de 2i/4 est 0,5).

Les lignes 37-38 décrémentent i (registre 07) pour le prochain passage dans la boucle. Ensuite, si i=n, les lignes 43-46 sauvent la valeur courante de $J_i(x)$ et $J_{i+1}(x)$ pour usage ultérieur. Sinon ces lignes sont sautées. A moins que i=0 (lignes 49-50), la boucle LBL 00 est répétée, un pas de plus vers $J_0(x)$.

Quand i=0 est atteint, les registres 04 et 05 contiennent des estimations de $J_0(x)$ et $J_1(x)$. Les lignes 51-52 ajustent la somme pour le terme supplémentaire $J_0(x)$ ajouté ligne 35. Alors le facteur de normalisation est appliqué aux quatres estimations de la fonction de Bessel. Quand le programme s'arrête, les résultats suivants sont dans la pile :

registre	contenus
T	$J_1(x)$
Z	$J_0(x)$
Y	$J_{n+1}(x)$
X	$J_n(x)$

1B. La fonction EMDIR

Avant de sauver le programme "JNX" en mémoire d'extension ; il est utile de contrôler l'état de cette mémoire. Pressez :

XEQ ALPHA E M D I R ALPHA

EMDIR est la fonction qui donne la table des matières de la mémoire

d'extension. Si vous avez une HP-41CX, un raccourci est disponible. Vous pouvez presser :

shift CATALOG 4

pour obtenir la table des matières de la mémoire d'extension.

Si vous n'avez pas encore utilisé la mémoire d'extension, vous devez voir le message DIR EMPTY (table vide). Si vous avez sauvé des programmes ou des données dans l'X-M, remarquez que chaque entrée de la table décrit un "fichier", qui est un jeu de registres de la mémoire d'extension alloué au stockage d'un programme ou d'un bloc de données. La description du fichier est constituée de trois éléments : le nom du fichier, le type du fichier et la taille du fichier.

Le nom du fichier est une chaine de 7 caractères au plus. Le type de fichier est désigné par une seule lettre : P pour programme, D pour données, et A pour ASCII (texte). Ils sont décrits respectivement dans les chapitres 1, 2 et 3. La taille du fichier est le nombre de registres de la mémoire d'extension alloués. En fait deux registres supplémentaires par fichier sont utilisés pour l'en-tête du fichier. Un registre d'en-tête contient le nom du fichier, tandis que l'autre contient le type, la longueur et les pointeurs (tous les détails sont donné §10C). Ainsi, si vous créez un fichier de 12 registres, le nombre de fichiers disponibles en mémoire d'extension diminue de 14.

Vous pouvez contrôler le nombre de registres disponibles dans la mémoire d'extension en laissant EMDIR se dérouler complètement. Ce nombre apparait alors en X (faisant monter la pile). Ce nombre est toujours inférieur de 2 au nombre de registres restant dans la mémoire d'extension, car la calculatrice soustrait automatiquement les deux registres d'état nécessaires pour le prochain fichier. Donc si EMDIR laisse un compte de registres de 53, il y a en fait 55 registres inutilisés, mais le plus grand fichier que vous pouvez crêer est un fichier de 53 registres.

La fonction EMDIR est en quelque sorte analogue au CATALOGUE 1 pour la mémoire principale. A cause de son utilité, vous devez envisager d'assigner EMDIR à une touche. Pressez :

shift ASN ALPHA E M D I R ALPHA

suivi par la touche de votre choix. Avec une HP-41CX, le déroulement de la table peut être interrompu, déroulé pas à pas en avant et en arrière, comme le catalogue 1. Avec une HP-41C ou CV vous ne pouvez pas l'interrompre. A la place, vous pouvez "geler" l'affichage en maintenant pressée une touche autre que R/S ou ON. Relacher la touche permet au listage de reprendre.

Sur la HP-41CX, il y a deux autres fonctions en rapport avec cette table. La fonction EMROOM renvoie le nombre de registres de la mémoire d'extension disponible pour des données, exactement comme à la fin de EMDIR. La différence est que la table n'est pas affichée. Ceci rend EMROOM plus commode que EMDIR pour utilisation dans un programme qui crée des fichiers dans la mémoire d'extension. Vous pouvez contrôler le nombre de registres disponibles avant d'essayer de crêer un fichier, en réduisant éventuellement la taille du fichier pour correspondre à EMROOM.

Egalement, sur la HP-41CX, on trouve la fonction EMDIRX. Quand vous mettez un nombre n en X et exécutez EMDIRX, le nom du nième fichier est renvoyé en ALPHA et le type de fichier retourné en X sous forme d'une chaine de deux caractères (PR, DA ou AS, pour programmes, données ou

fichiers ASCII).

1C.Utiliser SAVEP et GETP

Pour sauver le programme "JNX" dans la mémoire d'extension, pressez simplement :

ALPHA J N X ALPHA

suivi par :

XEQ ALPHA S A V E P ALPHA

L'affichage va s'effacer pendant quelques instants, à l'exception des indicateurs, pendant que l'opération est exécutée.

Cette procédure d'utilisation de **SAVEP** est similaire à la procédure que vous devrez utiliser avec beaucoup d'autres opérations sur la mémoire d'extension. D'abord vous placez le nom du programme ou du fichier en ALPHA, puis vous exécutez la fonction.

Contrôlez le résultat de SAVEP en exécutant EMDIR (ou CATALOG 4 pour la 41CX). Vous devez voir :

JNX P012

Si cela va trop vite, vous pouvez essayer EMDIR à nouveau. Sur une HP-41C ou CV, tenir pressée une touche "gèle" l'affichage jusqu'à ce que vous relachiez la touche. Sur une HP-41CX, pressez R/S pour arrêter le déroulement et SST ou BST pour le parcourir pas à pas.

Un autre moyen pour contrôler l'existence ou la taille d'un fichier de la mémoire d'extension est l'utilisation de la fonction FLSIZE. Mettez simplement le nom du fichier en ALPHA et exécutez FLSIZE. Dans l'exemple ci-dessus, vous devez presser :

ALPHA J N X ALPHA
XEQ ALPHA F L S I Z E ALPHA

Le résultat doit être le nombre 12 dans le registre X. C'est la taille du fichier "JNX", sans compter les deux registres d'en-tête nécessaires au fichier.

Quand vous utilisez **FLSIZE**, le résultat est la taille du fichier désigné s'il existe, ou le message d'erreur FL NOT FOUND si le fichier n'existe pas. FLSIZE fonctionne de la même façon pour les trois types de fichiers (programmes, données, ASCII). Si vous exécutez FLSIZE avec le registre ALPHA vide, vous obtenez la taille du fichier "de travail". le fichier "de travail" ou "courant" est le dernier fichier auquel vous vous êtes référé par son nom dans une fonction de la mémoire d'extension, comme SAVEP, ou le dernier fichier affiché par EMDIR. Des détails supplémentaires sur les fichiers de travail sont donnés page **20**.

AVERTISSEMENT: (révision 1B seulement) Quand vous utilisez SAVEP, vérifiez que la calculatrice est positionnée dans la mémoire principale, **non pas** dans un module d'application en mémoire morte (ROM). Méfiez-vous particulièrement quand un modules d'application (MATH, STANDARD, etc...) est branché. Même l'imprimante (ou le module HP-IL avec l'imprimante validée) contient trois programmes (PRAXIS, PRPLOT et PRPLOTP), en mémoire morte. Le programme "XF" présenté au chapitre 10 garanti que vous soyez en mémoire principale quand vous exécutez SAVEP. Cet avertissement ne s'applique pas à la 41CX, ni aux révisions 1C et au dessus du module de fonctions d'extension mémoire. (NDT: certains modules REV 1B ne présentent pas non plus ce bug. Faites l'essai).

Pour savoir si oui ou non vous êtes dans un programme d'un module d'application, en mode calcul pressez :

shift RTN

et passez en mode programme. Ceci vous déplace sur la ligne 00 du programme courant. Si vous voyez seulement :

00

vous êtes dans un module d'application. Vous pouvez presser :

shift CATALOG 1 ou GTO..

pour retourner dans la mémoire principale. Si vous connaissez le nom d'un programme particulier de la mémoire principale, vous pouvez presser :

GTO ALPHA (nom du programme) ALPHA

ce qui vous ramène aussi dans la mémoire principale.

Une fois dans la mémoire principale, si vous pressez :

shift RTN PRGM

pour venir ligne 00 du programme courant, vous devez voir :

00 REG nnn

où nnn est le nombre actuel de registres libres dans la mémoire principale.

Si vous faites accidentellement un SAVEP alors que vous êtes en dehors de la mémoire principale, faites un PURFL immédiatement. Le fichier de programme que SAVEP crée dans ces conditions risque d'être trop grand et il est certainement inutilisable. L'un de ces fichiers peut même bloquer le clavier du calculateur si vous essayez de le rappeler en mémoire principale.

Incidemment, si vous voulez transférer un programme d'un module d'application dans la mémoire d'extension, vous devez d'abord le copier (COPY) en mémoire principale.

Le même avertissement (êtes vous en dehors de la mémoire principale?) s'applique à la fonction PCLPS. En cas de PCLPS, la punition pour un faux pas est l'épouvantable MEMORY LOST. Encore une fois, cet avertissement ne s'applique pas à la HP-41CX ou aux révisions 1C et au dessus du module de fonctions d'extension mémoire.

Voyons maintenant comment récupérer le programme "JNX" qui se trouve dans la mémoire d'extension. Vérifiez que la registre ALPHA contient toujours "JNX", puis GTO.. et pressez ALPHA G E T P ALPHA. A condition que vous ayez assez de place disponible pour les programmes, vous devez maintenant avoir un deuxième programme "JNX" en mémoire. Exécutez CATALOG 1 et vous devez voir LBL"JNX, END, LBL"JNX, and .END. REG xx comme dernier élément listé.

Ceci illustre le fait que **GETP** retrouve le programme désigné et le place entre le dernier END et le .END. même si cela signifie qu'un programme va être recouvert. En particulier, si vous n'aviez pas exécuté GTO.. pour placer un END à la fin du programme "JNX" avant de le sauver, le GETP aurait recouvert l'ancienne version de "JNX" avec la nouvelle, laissant seulement une version au lieu de deux. Vous voudrez sans doute

vous entrainer davantage avec SAVEP et GETP avec vos propres programmes.

Quand elles sont assignées à des touches, les fonctions SAVEP et GETP procurent des équivallent en une seule pression de touche de l'enregistrement et de la lecture de cartes magnétiques de programmes. Une application typique est le déplacement d'un programme vers la fin du catalogue 1 pour obtenir une réponse plus rapide lors de corrections ou de PACK. (Quand vous insérez une instruction ou faites PACK dans un programme qui est situé près du début du catalogue 1, tous les programmes qui sont à la suite doivent être déplacés. Ceci ralentit la calculatrice notablement). Utilisez SAVEP pour sauver le programme, CLP pour l'effacer de la mémoire principale, et GETP pour le ramener à la fin du catalogue 1. Des techniques de ce type sont trés utiles, mais la pleine puissance de SAVEP et GETP est mise en oeuvre par l'usage de ces instructions dans vos programmes.

Avant d'explorer ce sujet, vous devez en savoir un peu plus au sujet de SAVEP et GETP.

1D POSSIBILITES AVANCEES DE SAVEP

Les exemples ci-dessus peuvent vous laisser penser qu'un programme ne peut être sauvé que dans un fichier de mémoire d'extension ayant le même nom que le programme. Avec un petit effort supplémentaire, il est possible de sauver un programme en donnant au fichier le nom que vous voulez. Au lieu de mettre simplement le nom du programme (en fait, le nom de n'importe quel label alpha, apparaissant dans le catalogue 1 du programme) dans le registre ALPHA, vous pouvez faire suivre le nom du programme du nom désiré pour le fichier, en utilisant une virgule comme séparateur. Si vous voulez sauver le programme courant, vous pouvez omettre le nom du programme et mettre seulement en ALPHA une virgule suivie du nom du fichier.

Le programme courant est celui qui apparaît quand vous passez en mode programme. C'est habituellement le programme que vous avez exécuté le plus récemment, à moins que vous n'ayiez pressé GTO.. ou CATALOG 1, car les deux peuvent vous amener à une partie différente de la mémoire programme. Les contenus du registre ALPHA valables pour SAVEP sont les suivants :

Contenus de ALPHA	Résultat
"nom du programme"	Le programme contenant un label ALPHA de ce nom est sauvé dans la mémoire d'extension dans un fichier du même nom.
"nom du programme,nom du fichier"	Le programme ainsi nommé est sauvé dans un fichier du nom donné. Attention : Ne pas laisser d'espace après la virgule, à moins que vous vouliez que cet espace fasse partie du nom du fichier.
",nom du fichier"	Le programme courant est sauvé dans un fichier de ce nom.

Note : les virgules ne sont pas admises dans les noms de fichier, puisque la virgule est interprétée comme séparateur. Les noms de fichier ne peuvent excéder 7 caractères (tout caractère supplémentaire n'est pas retenu).

1E. Les caractéristiques avancées de GETP, y compris GETSUB et PCLPS

Contrairement à SAVEP, la fonction GETP opère différemment quand elle est exécutée en tant que partie d'un programme plutôt que provenant du clavier. Dans les deux cas, GETP ramène le fichier de programme dont le nom est donné dans le registre ALPHA dans la mémoire programe, le plaçant juste après le dernier END dans le Catalogue 1 et avant le .END. . Comme quand on lit un programme à partir de cartes, de codes-barres, ou de cassette, les assignements des label ALPHA du programme ne prendront effet que si le GETP est exécuté en mode USER. Tout assignement conflictuel préexistant est recouvert.

Quand il est appellé du clavier, GETP place le calculateur à la première ligne du programme ramené. Cela permet commodément soit de passer en mode programme pour revoir le programme soit de presser R/S pour l'exécuter.

Quand on le trouve dans un programme en cours d'exécution, GETP lit le fichier programme nommé et continue à exécuter le programme original. Une exception est faite pour le cas où le programme original était le dernier programme en mémoire principale. Le dernier programme en mémoire principale est celui qui contient le .END. comme sa dernière ligne. Dans ce cas, lorsque le fichier programme est lu, le programme original est recouvert. En clair le programme original ne peut continuer à fonctionner. A la place l'exécution continue à la première ligne du nouveau programme.

Quand on écrit un programme qui utilise GETP, on doit soigneusement envisager les opérations GETP de façon à ne pas recouvrir accidentellement des programmes importants. Il est souvent utile de placer une note dans les instructions du programme, demandant que l'utilisateur soit efface le dernier programme en mémoire soit fasse GTO.. avant d'exécuter le programme. C'est une bonne méthode de travail de ne pas faire GTO.. avant d'être sûr soit que la dernière zone programme en mémoire principale est vide (pas de lignes autre que le .END.) soit qu'elle contient au moins un label ALPHA. Cette précaution aidera à prévenir la prolifération ennuyeuse de END en excès dans le catalogue 1.

Un programme peut utiliser GETP pour charger les sous-programmes qui lui sont nécessaires à partir de la mémoire d'extension, en effaçant à chaque fois le sous-programme utilisé précédemment. Cette technique appelée recouvrement est nécessaire pour les très grands programmes lorsque tous les sous-programmes ne rentrent pas dans la mémoire principale en même temps.

Les précautions à prendre quand GETP est utilisé peuvent vous inciter à utiliser GETSUB à la place. Le GETSUB est presque la même chose que GTO.. suivie de GETP. La différence est qu'un nouveau END est créé même si le dernier programme en mémoire principale est vide. Si vous prenez l'habitude d'utiliser GETSUB, vous vous rendrez bientôt compte que votre catalogue 1 est rempli de ENDs excédentaires. Ces END devront être effacés à l'aide de la procédure suivante :

Si vous avez un END excédentaire sans label ALPHA le précédant dans le catalogue 1, la seule façon de s'en débarrasser est d'exécuter le catalogue 1 et de l'interrompre au END excédentaire. Ensuite vous pouvez soit passer en mode programme et l'effacer soit faire :

XEQ ALPHA C L P ALPHA ALPHA ALPHA.

Quand aucun nom de programme n'est donné, la fonction CLP efface le programme courant. Si plusieurs END sont l'un derrière l'autre, arrêter le

catalogue 1 au premier, passer en mode programme et presser la touche d'effacement puis SST alternativement jusqu'à ce qu'apparaisse une ligne qui n'est pas un END, indiquant que le groupe entier de END a été effacé.

Que GETSUB soit exécuté à partir d'un programme ou à partir du clavier, le résultat est le même. Le .END. est transformé en END ; le fichier programme nommé dans le registre ALPHA est lu, et un nouveau .END. est ajouté. L'exécution n'est pas transférée au nouveau programme.

Le seul usage valable de GETSUB est de récupérer un programme à partir de la mémoire d'extension lorsque le dernier programme au Catalogue 1 contient le .END. comme dernière ligne et que vous ne voulez pas que le programme soit effacé. Cela peut se produire lorsque plusieurs fichiers programme doivent être lus à partir de la mémoire d'extension. Le premier fichier peut être lu en utilisant GETP, alors que GETSUB peut être utilisé pour les fichiers suivants. Cette procédure évite la création de END excédentaires. Cependant, si le dernier programme au Catalogue 1 comporte un END non permanent, ou si vous ne voulez pas sauver ce programme, alors utilisez GETP plutôt que GETSUB.

Incidemment, si vous connaissez bien les fonctions du lecteur de carte, exécuter GETP est justement analogue au fait de lire un jeu de cartes de programme, et GETSUB est presque analogue à l'exécution de la fonction RSUB du lecteur de carte. La différence est que RSUB ne transforme le .END. en END que si vous êtes dans le dernier programme en mémoire principale (celui qui comporte le .END. comme dernière ligne).

Quand vous avez fini d'utiliser tous les programmes qui étaient lus, vous pouvez utiliser la fonction PCLPS pour les effacer. Chargez simplement le registre ALPHA avec le nom du premier programme que vous lisez à partir de la mémoire d'extension (celui que vous lisez avec GETP, plutôt que GETSUB). Puis PCLPS effacera ce programme et tous les programmes qui le suivent dans le catalogue 1. Si le registre ALPHA est effacé, le programme courant et tous les autres programmes qui suivent seront effacés également. Cela se produira exactement de la même façon si PCLPS est appelé du clavier ou rencontré dans un programme en cours d'exécution. L'exécution du programme en cours continuera après que les programmes désignés aient été effacés.

AVERTISSEMENT : (Révision 1B seulement) si le calculateur est placé en dehors de la mémoire programme principale (dans un programme d'un module d'application) et que le registre ALPHA n'est pas effacé, l'exécution de PCLPS donnera un MEMORY LOST, après un retard de plusieurs secondes en guise d'effet dramatique. Se référer à l'avertissement de SAVEP (page 13) pour plus de détails. Le programme "XF" dans la section 10B a le bénéfice accidentel de prévenir l'apparition de ce problème.

Les lecteurs astucieux remarqueront que PCLPS a une exception très voisine de celle de GETSUB. PCLPS efface le programme nommé et tous les programmes qui le suivent. Si la fonction PCLPS est exécutée dans un programme qui est un des programmes en cours d'effacement, l'exécution se terminera au .END. Cependant, si l'instruction PCLPS faisait partie d'un sous-programme, le .END. sera exécuté et interprété comme un RTN. Le contrôle retournera au programme appelant qui peut ne plus exister si vous n'avez pas envisagé les choses correctement. Si le programme appelant n'existe pas (à cause de l'action de PCLPS), vous vous retrouverez en dehors de la zone programme de la HP-41, dans les registres d'assignement et du tampon d'entrée/sortie qui se trouvent sous le .END.. Cela se produira quel que soit le numéro de révision des fonctions étendues que

vous possédez. Bien que les programmeurs de synthétiques en savourent les possibilités, cette situation devrait être évitée.

Si vous êtes assez malchanceux pour que ce problème vous arrive, interrompez simplement le programme (si il ne s'arrête pas seul avec une erreur quelconque) et alors faites shift Catalogue 1 ou GTO .. pour replacer le calculateur dans la mémoire programme principale.

1F. Effacement d'un fichier à partir de la mémoire d'extension.

De même que la fonction CLP efface le programme désigné de la mémoire principale, la fonction PURFL supprime le fichier désigné de la mémoire d'extension. Le fichier désigné peut être un fichier programme, données ou ASCII. Après la suppression du fichier, la mémoire d'extension est automatiquement compactée pour récupérer l'espace autrefois utilisé par le fichier. Cette opération est quelque peu semblable au compactage réalisé par CLP.

Avertissement : (révision 1B seulement) la fonction PURFL a une caractéristique très dangereuse. Après l'exécution de PURFL, il n'y a plus de fichier travail. Si un fichier travail n'est pas rapidement rétabli, le contenu entier de la mémoire d'extension peut être rendu inaccessible. Toute instruction qui opère sur le fichier travail détruira la table de la mémoire d'extension si un fichier travail n'existe pas au moment où l'instruction est exécutée. Ensuite des techniques spéciales (section 10E) sont nécessaires pour restaurer la table.

NOTE : Si votre module de fonctions d'extension est une révision 1B, Ce "parasite" peut être utile. La séquence PURFL, RCLPT est une manière rapide et facile d'effacer la table de la mémoire d'extension sans gêner la mémoire principale. Toutefois, cette séquence ne devrait jamais être utilisée dans un programme.

Le fichier travail, qui est appelé fichier courant dans le mode d'emploi de la HP-41CX, est le dernier fichier utilisé ou créé, à moins qu'une instruction EMDIR (ou Catalogue 4 sur la HP-41CX) n'ai été exécutée depuis lors. Lorsque vous consultez la table de la mémoire d'extension sur une HP-41C ou CV, le dernier fichier devient le fichier travail.

Cela est vrai que la table soit interrompue ou continue jusqu'à la fin. Sur une HP-41CX, le fichier travail ne change que si vous ne laissez pas la table continuer jusqu'à la fin.

Le fichier travail en mémoire étendue est analogue au programme de travail ou courant en mémoire principale. Le programme courant en mémoire principale est le programme qui apparait quand vous passez en mode programme. Toutes les opérations relatives au programme qui ne spécifient pas un nom de label ALPHA opèrent sur le programme courant. Les instructions telles que GTO.001, LIST 999, DEL 005, et CLP ALPHA ALPHA opèrent toutes sur le programme courant. En mémoire principale, le programme courant est choisi entre deux façons. C'est normalement le programme auquel on a accédé en dernier par une instruction GTO "label" ou XEQ "label". Cependant, si vous exécutez par la suite le catalogue 1, le programme auquel le catalogue s'arrête devient le programme courant. Par conséquent en choisant soigneusement un point où arrêter le catalogue 1, vous pouvez choisir un programme courant sans avoir à épeler une instruction GTO "label" (si en fait le programme que vous voulez comporte un label ALPHA).

Juste comme le catalogue 1, EMDIR peut être prématurément arrêté en pressant R/S pour choisir un fichier de travail. Les autres façons

d'établir un fichier de travail sont de créer un nouveau fichier ou d'exécuter toute instruction qui se réfère à un fichier existant par son nom.

Maintenant supposez que vous venez d'exécuter PURFL, mais que vous n'avez pas encore établi un nouveau fichier de travail. Si maintenant vous exécutez une fonction qui opère sur le fichier de travail, par exemple FLSIZE avec le registre ALPHA vide, vous obtiendrez le message FL NOT FOUND. Ceci est certainement raisonnable, puisqu'il n'y avait pas de fichier de travail sur lequel travailler. Cependant, avec le module de fonctions d'extension mémoire révision 1B, si vous exécutez alors un EMDIR, vous verrez le message DIR EMPTY. Tout ce que vous aviez en mémoire étendue est parti.

Il y a plusieurs façons d'éviter le problème avec PURFL si vous avez un module de fonctions d'extension mémoire révision 1B. Votre première ligne de défense est de prendre l'habitude d'exécuter EMDIR ou d'établir un nouveau fichier de travail immédiatement après avoir utilisé PURFL. Cela inclut tout usage de PURFL dans vos programmes. Votre seconde ligne de défense est de vous demander "ai-je défini le bon fichier de travail" ? avant toute instruction qui opère sur le fichier de travail.

Là où une instruction qui peut opérer sur le fichier désigné ou le fichier de travail doit être utilisé dans un programme, vous pouvez la précéder par les pas ALENG 1/X pour vous assurer que le registre ALPHA n'est pas vide (voir page 48). En dernier lieu, un court programme appelé "PFF" dans la section 10E permet de réparer les dommages après coup. Ce programme comporte quelques instructions synthétiques qui ne peuvent être rentrées par des moyens normaux, mais les codes-barres pour le programme sont donnés dans l'appendice D. Une fois de plus, ce problème avec PURFL ne se produit pas avec la HP-41CX ou avec les révisions 1C et au dessus du module de fonctions d'extension mémoire.

Un détail supplémentaire sur SAVEP mérite d'être mentionné. Si vous sauvez une nouvelle version d'un programme qui est déjà sauvé dans la mémoire d'extension, l'ancien fichier sera automatiquement purgé et la nouvelle version sera ajoutée à la fin du directory de la mémoire d'extension. Le savoir à l'avance peut vous épargner quelques moments de panique quand vous consultez la table. Vous pouvez retenir l'ancien fichier de programme si vous choisissez d'utiliser un nom différent pour le nouveau fichier de programme (voir page 14).

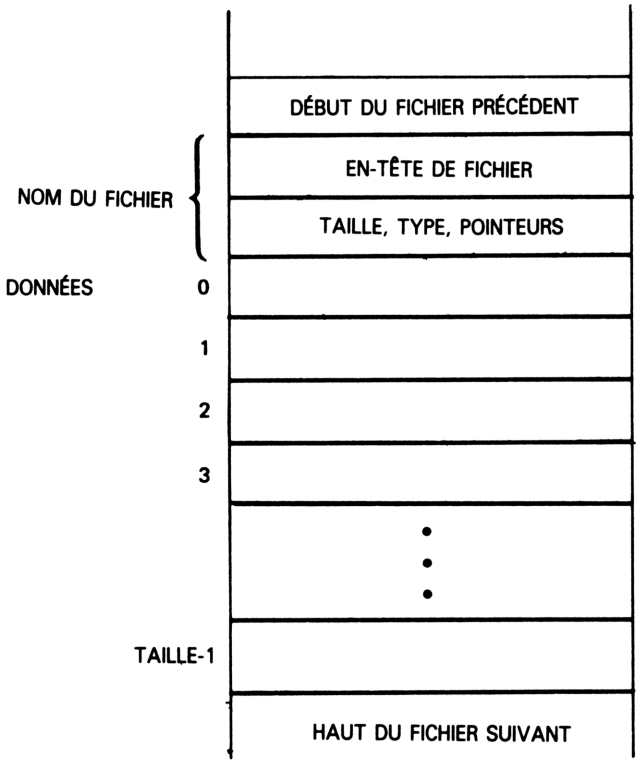


Figure 2.1 : Usage des registres par un fichier de la mémoire d'extension
Cette information est disponible à travers FLSIZE, EMDIR ...

C H A P I T R E 2

SAUVER DES DONNEES DANS LA MEMOIRE D'EXTENSION

2A. Structure des fichiers

Sauver des données dans la mémoire d'extension demande quelques pas de plus que pour sauver un programme. Plutôt que simplement sauver les données, nous devons d'abord créer un fichier de données vide dans la mémoire d'extension où sauver les données. La figure 2.1 montre la structure d'un tel fichier. Cette structure est la même pour les trois types de fichiers (fichiers programme, données et texte), sauf que l'information à l'intérieur du fichier est organisée différemment. Pour les fichiers de programme et les fichiers de texte, chaque registre vaut 7 octets.

Pour rendre les exemples de ce chapitre plus faciles à suivre, utilisez la petite routine qui suit pour charger à l'avance les registres de donnée avec les mêmes valeurs que les numéros de registre. Le registre de données 00 a la valeur 0, le registre 01 a la valeur 1, et ainsi de suite, jusqu'à ce qu'on rencontre un registre NONEXISTENT.

```
01 LBL "PRELOAD"
02 1                Charge la pile avec des 1
03 ENTER↑           en prévision des additions
04 ENTER↑
05 ENTER↑
06 CLX              Début avec X=0
07 LBL 01
08 STO IND X        Stocke X dans le registre numéro x
09 +                Ajoute 1
10 GTO 01           Retour ligne 07
11 END
```

2B. La fonction SAVERX et le fichier de travail.

Supposez que vous voulez sauver le contenu des registres de données 2 à 9. Si vous utilisiez des cartes magnétiques, vous rentrerez :

2,009 XEQ "WDTAX"

Pour sauver ces registres dans la mémoire d'extension vous devez d'abord créer un fichier de données d'au moins 8 registres. Il n'y a pas d'instruction analogue à SAVEP qui crée le fichier et transfère les données dans une seule opération. Donnons au fichier de données le nom : "ABC". Faites :

ALPHA A B C ALPHA

pour donner un nom au fichier, puis

8

pour donner la taille du fichier, suivi de :

XEQ ALPHA C R F L D ALPHA

pour créer un fichier vide de 8 registres. L'instruction CRFLD s'attend à

trouver le nom de fichier dans le registre ALPHA et la taille du fichier (le nombre des registres de données dans le fichier) dans X. CRFLD efface les registres dans un fichier au moment où il est créé. Exécutez EMDIR et vous devriez voir :

ABC D008

comme dernière entrée dans la table. Pour sauver le contenu des registres 2 à 9 dans le fichier de données "ABC", faites :

2,009 XEQ "SAVERX"
(faites XEQ ALPHA S A V E R X ALPHA)

La fonction SAVERX accepte un nombre dans le registre X de la forme bbb,eee.

Une séquence de registres de données commençant avec le numéro de registre bbb et se terminant par le numéro de registre eee est transférée au fichier de données de travail dans la mémoire d'extension. Le fichier de travail est le dernier fichier utilisé ou créé, à moins d'exécuter un EMDIR ou un PURFL. Quand vous consultez la table de la mémoire d'extension, le fichier de travail devient le fichier auquel vous arrêtez la table. Sur la HP-41CX, si vous laissez la table continuer jusqu'à la fin, le fichier de travail demeure inchangé. Sur la HP-41C ou CV, en laissant la table continuer jusqu'à la fin, le dernier fichier dans la table devient fichier de travail.

2C. La fonction GETRX et le pointeur de registres

Si vous y pensiez à l'avance, vous pourriez supposer que la séquence 12,019 XEQ "GETRX" récupèrerait les 8 nombres et les placerait dans les registres de données de 12 à 19. Aussi logique que cela puisse paraître, ce n'est pas le cas. Si vous essayez cette séquence, vous obtiendrez le message d'erreur END OF FL. Quelques explications sont nécessaires.

Un fichier de données dans une mémoire étendue peut être très grand. Un seul fichier de la sorte peut être utilisé pour plusieurs blocs de données. De plus, vous n'avez pas besoin de rappeler le fichier entier d'un coup. De petits blocs de données ou même des registres isolés peuvent être récupérés du fichier de données. Le prix pour cette flexibilité est qu'un pointeur est nécessaire pour spécifier où dans le ficher de données vous souhaitez stocker ou récupérer des données. Sans pointeur il serait impossible de garantir que GETRX récupèrerait le bon bloc de données.

Mais si un pointeur est nécessaire, pourquoi ne nous fallait il pas l'établir avant de faire l'opération SAVERX ? Normalement il est nécessaire d'établir le pointeur, mais dans ce cas, le fichier "ABC" venait d'être créé. Un nouveau fichier créé a un pointeur qui est automatiquement initialisé à zéro, signifiant que toute opération de SAVE ou GET est réalisée en commençant au registre 0, le premier registre du fichier. Les registres dans le fichier de données de la mémoire d'extension sont numérotés à partir de 0, juste comme sont numérotés les registres de données dans la mémoire principale.

La séquence : 2,009 SAVERX stockait les contenus des registres de données 2 à 9 dans les 8 premiers registres (et seulement 8 registres) du fichier de données "ABC".

Cette séquence a eu l'effet caché supplémentaire d'avancer le pointeur de 0 (le premier registre) à 8 (un registre après le dernier registre du fichier). Toute opération supplémentaire telle que GETRX donnera le message

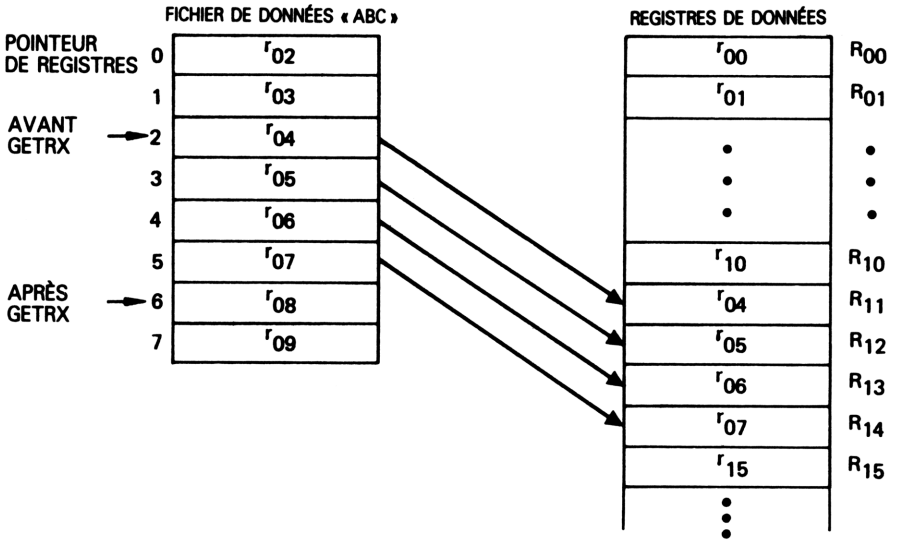


Figure 2.2 : Effet de la séquence : « ABC » 2 SEEKPTA 11,014 GETRX.

d'erreur END OF FL jusqu'à ce que le pointeur soit replacé.

Pour placer le pointeur, nous utilisons la fonction SEEKPTA. La fonction SEEKPTA place simplement le pointeur à la valeur spécifiée en X. Le nom du fichier devrait aussi être spécifié dans le registre ALPHA. Si le registre ALPHA est effacé, SEEKPTA va opérer sur le fichier de travail. Tout fichier de données a son propre pointeur, stocké dans un des 2 registres d'en-tête du fichier. Un SEEKPTA sur un fichier particulier n'affectera pas le pointeur d'un autre fichier.

Par exemple, supposez que vous voulez récupérer les premiers contenus des registres de données 4 à 7 du fichier de données "ABC" et placer ces quatre valeurs dans les registres de données 11 à 14. la figure 2.2 illustre cette opération. Notez sur la figure 2.2 que le premier contenu du registre de données 4 réside dans le troisième registre du fichier "ABC". Par conséquent nous pressons :

ALPHA A B C ALPHA 2 XEQ ALPHA S E E K P T A ALPHA

pour mettre la valeur du pointeur à 2, le troisième registre du fichier. Une fois cela fait nous pressons simplement :

11,014 XEQ ALPHA G E T R X ALPHA

pour recupérer les données. Utilisez RCL pour vérifier que les registres 11, 12, 13 et 14 comportent les mêmes valeurs que les registres 4, 5, 6 et 7.

La fonction GETRX accepte un nombre dans le registre X de la forme bbb,eee, désignant le bloc de registres de données dans lequel les données récupérées doivent être placées. GETRX récupère le nombre indiqué de registres de données du fichier de travail de la mémoire d'extension, commençant au registre courant du fichier. GETRX avance aussi le pointeur de registres au premier registre suivant le du bloc qui a été récupéré dans la mémoire d'extension.

SAVERX, comme GETRX, avance le pointeur de registres au premier registre suivant le du bloc qui a été écrit dans la mémoire d'extension. En fait, cette avancée automatique du pointeur de registres est commune à toutes les fonctions de fichier de données SAVE et GET.

La fonction RCLPTA fournit un moyen facile de vérifier la valeur courante de tout pointeur de fichier, au cas où vous ne vous en rappelleriez pas. Mettez simplement le nom du fichier dans ALPHA et faites RCLPTA, et la valeur du pointeur sera rappelée en X. Si ALPHA est effacé, RCLPTA va opérer sur le fichier de travail. Comme pour toute opération de RCL, la pile montera à moins que RCL soit immédiatement précédé d'un ENTER, CLX, ou d'une opération similaire empêchant la montée de la pile.

Comme exemple de RCLPTA, vérifions maintenant le pointeur. Puisque vous venez juste de rappeler les registres de 4 à 6 du fichier "ABC", si vous exécutez RCLPTA, le résultat devrait être 7.

Truc : RCLPTA est une façon commode de choisir un fichier qui doit être un fichier de travail sans altérer le contenu du fichier.

2D. Les fonctions SEEKPT et RCLPT

La fonction SEEKPT opère identiquement à SEEKPTA, sauf que l'opération de mise en place du pointeur est réalisée sur le fichier de travail, plutôt que sur le fichier indiqué dans ALPHA. Si vous êtes sûr que le fichier est le fichier de travail, SEEKPT sauve quelques pas. Dans le cas contraire utilisez SEEKPTA.

Le même conseil s'applique pour l'emploi de SEEKPT dans un programme. Si un pas précédent du programme établissait le bon fichier de travail, vous pouvez utiliser SEEEKPT, sinon, utilisez SEEKPTA.

La fonction RCLPT est une version de RCLPTA qui opère sur le fichier de travail. Utilisez la au lieu de RCLPTA lorsque vous savez que le fichier est le fichier de travail.

2E. Fonctions supplémentaires pour fichier de données : SAVEX, GETX, SAVER, GETR, CLFL

La fonction SAVEX transfère les contenus du registre X au fichier de travail, qui doit être un fichier de données, dans la mémoire d'extension. La valeur courante du pointeur désigne quel registre du fichier de données est utilisé. Après que la valeur soit sauvée, la valeur du pointeur augmente de 1. Par exemple, pour stocker la valeur 15 dans le troisième registre (registre numéro 2) du fichier de données, faites :

```
2 XEQ "SEEKPT"
15 XEQ "SAVEX"
```

La valeur du pointeur est maintenant 2+1=3, de sorte qu'une seconde instruction SAVEX stockerait x dans le registre 3 du fichier.

Cette incrémentation automatique du pointeur de registre avec SAVEX est extrêmement utile. Vous pouvez écrire un programme qui calcule un résultat à chaque fois en faisant une boucle, avec une seule instruction SAVEX pour stocker le résultat :

```
"nom du fichier"
CLFL ou CRFLD (CLFL sera étudié à la page 26)
LBL 01
(insérez ici les pas pour calculer le résultat)
SAVEX
GTO 01
```

Il n'est pas nécessaire de faire appel au pointeur de registres ou au compteur ISG pour le stockage. Si vous avez besoin d'un compteur pour le calcul, vous pouvez utiliser RCLPT comme compteur incorporé. Si ça vous chante, vous pouvez même laisser le END OF FL terminer les calculs. Cela rend la structure du programme très simple.

La fonction GETX, est l'opération inverse de SAVEX. Le registre courant du fichier de travail est récupéré et amené en X. Le pointeur de registres est augmenté de 1, et la pile est levée pour recevoir les données récupérées. Exactement comme pour RCL.

Par exemple, pour récupérer le nombre 15 dans le registre 2 du fichier de données "ABC" (qui devrait être toujours le fichier de travail si vous vous conformez aux exemples), faites :

```
SEEKPT
GETX
```

Le résultat devrait être 15 dans le registre X. Le pointeur de registre passe de 2 à 3, comme le montrera l'exécution de RCLPT. Exactement comme pour SAVERX, cette incrémentation automatique du pointeur rend commode l'utilisation de GETX dans une boucle.

La fonction SAVER tranfère tous les registres de données au fichier dont le nom est en ALPHA, ou au fichier de travail si ALPHA est vide. Contrairement à SAVERX, SAVER ignore totalement le pointeur de registres. Le registre de données 00 entre dans le registre 0 du fichier, le registre de données 01 entre dans le registre de fichier 1 et ainsi de suite. Malheureusement, bien que SAVER n'emploie pas le pointeur, il le modifie !

Le pointeur est laissé juste après le du dernier registre inscrit dans le fichier.

Si le fichier de la mémoire d'extension n'est pas assez grand pour contenir tous les registres de données, SAVER affiche un message d'erreur END OF FL et refuse de tranférer même un seul registre. Cette caractéristique, qui ne peut pas être surmontée par le flag 25, limite l'utilité de SAVER.

A moins que la taille courante ne soit précisément égale à la quantité de données que vous avez à sauver (et n'excède pas le FLSIZE du fichier de données choisi), vous devriez envisagez d'utiliser SAVERX plutôt que SAVER pour éviter de gaspiller de l'espace dans la mémoire d'extension. Naturellement, vous pouvez toujours réduire la taille jusqu'au nombre de registres de données que vous voulez sauver. Par exemple, pour sauver les registres de données de 00 à 23, utilisez la séquence :

```
24
PSIZE
"nom du fichier"
SAVER
```

La fonction PSIZE réduit la taille à 24, jetant les données situées au dessus du registre 23. Le reste des données est alors sauvé par SAVER. Cependant, cette technique n'est pas beaucoup plus facile que cette autre :

```
"nom du fichier"
0
SEEKPTA
,023
SAVERX
```

mais il se peut qu'elle soit préférée pour certaines applications.

La fonction GETR est bien plus utile que SAVER. GETR récupère les données en commençant par le registre 0 du fichier indiqué (le fichier de travail si ALPHA est vide), et les place dans les registres de données à 00 et au-dessus. Cela signifie que le rappel d'un fichier entier de données est aussi simple que :

```
"nom du fichier"
GETR
```

Il peut-être suivi par une instruction REGMOVE ou REGSWAP pour déplacer les données sur un bloc différent de registres si vous ne voulez pas qu'elles commencent au registre 00.

Comme SAVER, GETR ignore le pointeur mais le place a la position END OF FL, 1 registre après le du dernier registre récupéré.

La fonction CLFL efface le contenu du fichier de données ; c'est à dire qu'elle place tous les registres à zéro. Le pointeur de registres est aussi mis à zéro, de façon à ce que vous puissiez immédiatement commencer à utiliser les instructions SAVE pour le stockage des données dans le fichier. Mettez simplement le nom du fichier dans ALPHA et faites CLFL. Une application typique de CLFL est l'initialisation avant de réutiliser un fichier de données préexistant. Puisque les fichiers de données sont effacés lorsqu'ils sont créés, vous n'avez pas besoin d'utiliser CLFL sur un fichier de données nouvellement créé.

S'il n'y a pas de nom de fichier en ALPHA lorsque vous exécutez CLFL, un message NAME ERR apparaîtra. CLFL n'opérera pas sur le fichier de travail. Cependant, comme les autres fonctions de manipulation de fichiers, CLFL fait bien du fichier indiqué le fichier de travail. Si vous essayez d'utiliser CLFL pour effacer un fichier de programme, un FL TYPE ERR en résultera.

Les fichiers de programme ne peuvent être remplacés que par un nouveau programme (à l'aide de SAVEP) ou totalement supprimés.

La fonction PURFL, élimine le fichier indiqué de la mémoire d'extension, libérant ces registres pour d'autres usages.

Comme pour CLFL, il doit y avoir un nom de fichier valide dans ALPHA. Voir page 17 pour un avertissement important en ce qui concerne PURFL.

Sur la HP-41CX, la fonction RESZFL change la taille d'un fichier de données préexistant ou d'un fichier de texte. RESZFL n'opère que sur le fichier de travail. Utilisez RCLPTA avec le nom du fichier souhaité en ALPHA, ou utilisez n'importe quel moyen pour choisir le fichier souhaité comme fichier de travail. Puis mettre le nouveau FLSIZE en X et faites RESZFL. Si vous diminuez la taille du fichier, les registres ayant les numéros les plus élevés seront éliminés. Si il y a des données différentes de zéro dans ces registres, le calculateur donnera un message FL SIZE ERR. Vous pouvez passer par dessus cette protection en donnant une valeur négative à la FLSIZE souhaitée en X.

Comme vous l'avez vu, la mémoire d'extension est beaucoup plus flexible dans la lecture des données que le lecteur de cartes. La mémoire d'extension permet d'accéder facilement aux registres individuels de données, et à des sous groupes de registres à l'intérieur d'un bloc de données enregistrées. Cela donne une méthode commode pour analyser de grandes bases de données sans immobiliser tous vos registres de données. Vous pouvez en tirer des nombres à volonté, par bloc ou un par un.

Toute la puissance des fichiers de données de la mémoire d'extension sera illustrée au chapitre 6 avec les programme d'application "SOLVE", "DERIV", et "INTEG". Ces programmes utilisent la mémoire d'extension pour conserver leur données quand un programme fourni par l'utilisateur est appelé pour évaluer une fonction f(x).

Impression typique de « VAS »

```
RECORD 0:
THIS EXAMPLE ILLUSTRATES
 THE PRINTOUT/DISPLAY PR
ODUCED BY VAS.
RECORD 1:
SHORT RECORDS USE 1 LINE
RECORD 2:
LONGER RECORDS SPILL OVE
R INTO TWO OR MORE LINES
RECORD 3:
END OF FL
```

Liste des programmes « VAS »/« PVAS »

```
01♦LBL "VAS"      09♦LBL 01         18 "┣:"           26 GTO 01         34 AVIEW
 02 .9            10 "RECORD "      19 XEQ 10         27 RTN            35 STOFLAG
                  11 LASTX                                               36 RDN
03♦LBL "PVAS"     12 INT            20♦LBL 02         28♦LBL 10         37 FS?C 25
 04 ALENG         13 RCLFLAG        21 GETREC         29 SF 25          38 FC? 21
 05 1/X           14 CF 29          22 XEQ 10         30 PRA            39 PSE
 06 RDN           15 FIX 0          23 FS? 17         31 RCLFLAG        40 END
 07 INT           16 ARCL Y         24 GTO 02         32 FS?C 21
 08 SEEKPTA       17 STOFLAG        25 ISG L          33 FC?C 25              89 BYTES
```

CHAPITRE TROIS

FICHIERS DE TEXTE DANS LA MEMOIRE D'EXTENSION.

3A. Qu'est-ce qu'un fichier de texte ?

12 des 47 fonctions du module de fonctions d'extension mémoire et 14 des 61 fonctions d'extension de la 41-CX traitent exclusivement des fichiers de texte. Ce chapitre explique comment les fichiers ASCII sont utilisés et de quelle manière ils fournissent une capacité de manipulation de chaînes de caractères nouvelle et puissante. Si aucune de vos applications n'utilise de longues chaînes de caractères ALPHA, il est possible que vous préfériez sauter ce chapitre pour l'instant.

Avant l'arrivée des mémoires d'extension, traiter de chaînes de caractère sur la HP-41 était encombrant. Les chaînes de caractères devaient être segmentées en 6 caractères ou moins, parce que l'opération ASTO ne peut pas mettre plus de 6 caractères dans un registre.

La mémoire d'extension offre une nouvelle façon de traiter les chaînes qui ne demande plus qu'une chaîne soit divisée en registres. A la place, les chaînes de caractères sont stockées sans être segmentées dans un fichier de texte, ou fichier ASCII de la mémoire d'extension (les termes "fichier de texte" et "fichier ASCII" sont utilisés indifféremment dans ce livre, comme dans la documentation HP). Chaque chaîne ou **enregistrement**, peut contenir jusqu'à 254 caractères. Le nombre des différentes chaînes de caractères que vous pouvez avoir dans un seul fichier de texte n'est limité que par la taille de la mémoire d'extension. Comme pour un fichier de données, vous devez spécifier le nombre de registres qui doit être attribué, lorsque vous créez un fichier ASCII. Ce nombre doit être au moins :

$$N_{registres} = INT((N_{enregistrements}+N_{caractères}+7)/7),$$

Où $N_{enregistrements}$ est le nombre maximum d'enregistrements dont vous aurez besoin, $N_{caractères}$ est le nombre maximum de caractères qui seront stockés. Le +7 tient compte d'un octet de fin de fichier (voir section 10C) et de l'arrondi.

Par exemple, si vous voulez stocker 20 noms d'au plus 25 caractères chacun, il vous faudra :

$$N_{registres}=INT((20+20*25+7)/7)=INT(75,29)=75 \text{ reg.}$$

C'est une bonne idée d'utiliser un nombre un peu plus grand qu'il ne faut lorsque vous créez un fichier ASCII au cas où votre stockage aurait besoin de s'élargir. Le mode d'emploi du module de fonctions d'extension suggère d'ajouter 20 % à votre meilleure estimation du nombre de caractères à stocker, et de diviser le résultat par 7. Si vous possédez une HP-41CX, vous n'avez pas besoin de tant de précautions, parce que la fonction RESZFL facilite l'augmentation de la taille du fichier par la suite. Deux programmes présentés dans la section G de ce chapitre donnent une capacité similaire de changement de taille de fichier aux HP-41C et CV.

Juste comme les fichiers de données de la mémoire d'extension ont un pointeur de registre courant, les fichiers de texte ont un pointeur d'enregistrement courant. De plus, les fichiers de texte ont un pointeur de position du caractère courant dans l'enregistrement. Le pointeur d'enregistrements eee et le pointeur de caractères ccc sont combinés dans un seul

nombre décimal eee,ccc pour toutes les opérations de pointeur comme SEEKPT.

Pour illustrer ces points, essayons un exemple. Il nous faudra un fichier de texte de 25 registres appelé "NAMES". Assurez vous qu'il y a au moins 25 registres disponibles pour les données dans la mémoire étendue. Pour cela, exécutez EMDIR et laissez la table continuer jusqu'à la fin. Sur la HP-41CX vous pouvez utiliser la fonction EMROOM ou CATALOG 4 au lieu de EMDIR dans ce but. Le nombre en X est alors le nombre de registres disponibles pour les données dans la mémoire d'extension. Puis, pour créer le fichier "NAMES", faites :

25 ALPHA N A M E S ALPHA XEQ ALPHA C R F L A S ALPHA.

L'usage de la fonction CRFLAS, est très similaire à l'usage de CRFLD. Vous mettez le nom du fichier en ALPHA, le nombre de registres en X, et vous exécutez "CRFLAS".

Après avoir créé le fichier de texte appelé "NAMES", le pas suivant est d'utiliser la fonction APPREC, pour charger quelques données dans le fichier. Supposez que vous vouliez stocker les trois noms.

numéro d'enregistrement	Nom
0	RICHARD NELSON
1	ROGER HILL
2	JOHN MCGECHIE

Le procédé de stockage du nom est simple. Chargez simplement un nom dans le registre ALPHA et XEQ "APPREC". La fonction APPREC ajoute un nouvel enregistrement au fichier de texte de travail en ajoutant tout le contenus du registre ALPHA au fichier. Le pointeur est avancé à un caractère après le dernier caractère ajouté. La séquence d'opérations suivante charge les trois noms :

```
"RICHARD NELSON"    XEQ "APPREC"
"ROGER HILL"        XEQ "APPREC"
"JOHN MCGECHIE"     XEQ "APPREC"
```

Chaque guillemet (") indique que la touche ALPHA doit être pressée. Charger des données ALPHA dans un fichier de texte est beaucoup plus facile que les stocker dans les registres de données. La fonction APPREC manipule jusqu'à 24 caractères, plutôt que les 6 que ASTO peut manipuler. Rien que pour stocker le nom "RICHARD NELSON" dans les registres de données il faut 5 instructions : ASTO 01, ASHF, ASTO 02, ASHF, ASTO 03. C'est nettement plus encombrant que l'unique instruction APPREC. Cela devient encore plus encombrant si la chaîne de caractères doit rester inchangée dans ALPHA (ajoutez CLA, ARCL 01, ARCL 02, ARCL 03).

La facilité du chargement n'est nullement le seul avantage qu'il y a à utiliser des fichiers de texte de la mémoire étendue pour contenir des données ALPHA. La vraie puissance des fichiers de texte repose dans l'accès aux données, les capacités d'insertion et d'effacement.

3B. L'accès aux fichiers de texte.

Il y a deux fonctions qui rappellent des données à partir du fichier de texte. Ce sont ARCLREC, et GETREC.

Comme son nom l'indique, ARCLREC rappelle les caractères du fichier de texte de travail, à partir du pointeur courant, jusqu'à ce que le registre ALPHA soit rempli ou que la fin de l'enregistrement soit atteinte. Le

pointeur de caractères est avancé une position après le dernier caractère rappelé. La fonction ARCLREC lève ou baisse le drapeau 17 (le drapeau "enregistrement incomplet") pour indiquer si oui ou non il reste encore des caractères dans l'enregistrement. ARCLREC opère come ARCL, en ce qu'il ajoute les caractères rappelés à tout caractère existant dans le registre ALPHA. La fonction GETREC est exactement équivalente à la séquence CLA, ARCLREC.

Comme exemple, supposez que vous vouliez revoir les données dans le fichier "NAMES" (qui devrait encore être le fichier de travail puisque vous venez juste de le créer). La séquence :

```
0 SEEKPT
GETREC AVIEW
```

vous montre le contenu du premier enregistrement, "RICHARD NELSON". Si vous essayez ensuite la séquence :

```
ARCLREC AVIEW
```

le résultat sera "RICHARD NELSONROGER HILL". Oh là ! Nous oublions de faire un CLA avant le ARCLREC. La plupart du temps, vous trouverez que GETREC est plus commode à utiliser que ARCLREC, parce que GETREC efface automatiquement le registre ALPHA avant de rappeler l'enregistrement. La fonction ARCLREC sera utile pour les cas spéciaux dans lesquels le contenu d'un enregistrement doit être rajouté à un message.

Dans l'exemple précédent, la fonction ARCLREC pouvait faire contenir l'enregistrement entier dans le registre ALPHA, ainsi le drapeau 17 était baissé. Si l'enregistrement ne tenait pas dans le registre ALPHA, ARCLREC aurait levé le drapeau 17 pour indiquer qu'il restait encore des caractères dans l'enregistrement. Lorsque vous écrirez un programme qui imprime des chaînes de caractères à partir de fichiers de texte, vous utiliserez des séquences qui testent le drapeau 17. Par exemple :

```
(numéro d'enregistrement)
SEEKPT          Place le pointeur au début de l'enregistrement
LBL 01
GETREC          Rappelle 24 caractères de l'enregistrement
ACA (ou OUTA)   envoie les caractères à l'imprimante, mais n'im-
                prime pas encore
FS? 17          Si l'enregistrement est incomplet, rappelle 24
                caractères de plus.
GTO 01
PRBUF           Autrement imprime la chaîne de caractères accu-
                mulée.
```

Quelques périphériques HP-IL utilisent automatiquement le statut du drapeau 17 après ARCLREC ou GETREC. Si le drapeau 17 est levé, le retour chariot est supprimé de façon que le reste de l'enregistrement puisse être inclu sur la même ligne.

3C. Entrée des données dans les fichiers de texte.

Supposez que vous voulez changer le premier enregistrement du nom "RICHARD NELSON" à "RICHARD NELSON, FOUNDER OF PPC". La première chose que vous devez faire est de placer le pointeur d'enregistrements à zéro, qui est le premier enregistrement du fichier. Si vous n'avez rien fait pour

désigner un autre fichier de travail, le fichier NAMES est encore le fichier de travail. La séquence : 0 SEEKPT placera par conséquent les pointeurs au caractère 0 (le premier caractère) de l'enregistrement zéro (le premier enregistrement).

L'instruction APPCHR ajoute le contenu du registre ALPHA à la fin de l'enregistrement courant, ignorant le pointeur de caractères. Le pointeur est avancé à la fin de l'enregistrement courant, une position après le caractère ajouté. Contrairement à APPREC, APPCHR ne crée pas un nouvel enregistrement. Pour opérer le changement désiré à l'enregistrement 0, Pressez :

", FOUNDER OF PPC" XEQ "APPCHR"

vous pouvez utiliser la séquence :

```
0 SEEKPT
GETREC AVIEW
GETREC AVIEW
```

pour vérifier vos résultats. C'est beaucoup plus facile que d'utiliser ARCL, APPEND, et ASTO pour modifier une chaîne de caractères stockée dans les registres de données.

L'exemple suivant d'insertion utilise l'instruction INSCHR. Le but est de changer le premier enregistrement de "RICHARD NELSON, FOUNDER OF PPC" en "RICHARD J. NELSON, FOUNDER OF PPC". Cela demande l'insertion des caractères "J. " en avant du "N" dans "NELSON".

Pour que l'instruction INSCHR puisse fonctionner avec succès, vous devez dire à la HP-41 précisément où insérer les caractères. Dans cet exemple, cela signifie que le pointeur d'enregistrements doit être à 0 (le premier enregistrement) et le pointeur de caractères à 8, correspondant au 9ième caractère, "N". INSCHR insère toujours le contenu de ALPHA en avant de la position courante du pointeur et avance le pointeur de caractères par le nombre de caractères insérés. Comme pour les autres fonctions d'insertion de données, le pointeur termine une position après le dernier caractère inséré. La séquence :

```
,008 SEEKPT
"J. " INSCHR (n'oubliez pas l'espace)
```

effectue l'insertion de l'initiale intermédiaire "J. ". Utilisez :

```
0 SEEKPT
GETREC AVIEW
GETREC AVIEW
```

pour vérifier vos résultats. La séquence qui serait exigée pour faire cette insertion à l'aide des instructions ARCL et ASTO défie toute description !

L'exemple final de l'insertion du fichier de texte est l'adjonction d'un nouvel enregistrement au milieu d'un fichier préexistant. La fonction INSREC est donnée dans ce but.

Comme INSCHR, INSREC insère un nouvel enregistrement en avant du pointeur courant d'enregistrement, le chargeant avec le contenu de ALPHA. INSREC avance aussi le pointeur à la fin du nouvel enregistrement, juste après le dernier caractère. Comme exemple de la fonction INSREC, essayez d'insérer le nom "CLIFFORD STERN" entre "ROGER HILL" et "JOHN MCGECHIE". Puisque l'insertion doit être faite en avant du troisième enregistrement (numéro d'enregistrement 2), la séquence est :

```
2 SEEKPT
"CLIFFORD STERN" XEQ " INSREC"
```

L'instruction POSFL décrite à la page 33, facilite l'insertion de caractères ou d'enregistrements à la bonne place relative à toute chaîne de caractères choisie dans un fichier. D'abord vous utilisez POSFL pour trouver la chaîne de caractères avant laquelle l'insertion doit être faite, puis vous utilisez INSCHR ou INSREC comme vous voulez.

3D. Effacement de données des fichiers de texte

En continuant l'exemple précédent, nous avons :

numéro d'enregistrement	nom
0	RICHARD J. NELSON, FOUNDER OF PPC
1	ROGER HILL
2	CLIFFORD STERN
3	JOHN MCGECHIE

Supposez que vous voulez effacer le dernier enregistrement du fichier, l'enregistrement numéro 3. La fonction voulue pour cette opération est DELREC. La fonction DELREC efface l'enregistrement courant (comme désigné par le pointeur d'enregistrements) du fichier de texte de travail. DELREC ne change pas le pointeur d'enregistrements, mais elle marque zéro au pointeur de caractères. Pour effacer l'enregistrement numéro 3, la séquence est :

```
3 SEEKPT
DELREC
```

Pour vérifier le résultat, utilisez GETREC (le pointeur d'enregistrement est toujours 3). Vous devriez obtenir un message d'erreur END OF FL, indiquant que l'enregistrement 3 n'existe plus. Si vous aviez effacé l'enregistrement numéro 1, les enregistrements 2 et 3 se seraient déplacés pour devenir respectivement les nouveaux enregistrements 1 et 2. Par ailleurs, DELREC et INSREC ne traitent que d'un seul enregistrement. Si vous devez insérer ou effacer plusieurs enregistrements en un seul point d'un fichier, il se peut que vous ayez besoin d'une courte séquence de bouclage contenant DELREC ou INSREC.

Maintenant supposez que vous vouliez effacer la chaîne de caractères, "FOUNDER OF PPC" de l'enregistrement 0. La fonction DELCHR efface les caractères à partir de la position courante du pointeur. Le nombre de caractères à effacer est spécifié par la partie entière du nombre dans le registre X. Si ce nombre est plus grand que le nombre de caractères de la position courante du pointeur à la fin de l'enregistrement, l'effacement n'est effectué que jusqu'à la fin de l'enregistrement. Les pointeurs d'enregistrements et de caractères et le registre X demeurent inchangés par DELREC. Pour cet exemple, la séquence :

```
,017 SEEKPT
16 DELCHR
```

effectue l'effacement de ", FOUNDER OF PPC". La virgule était le 18ième caractère (caractère numéro 17) de l'enregistrement 0. Le nombre de caractère à effacer était 16. En fait, puisque vous effaciez tous les caractères de l'enregistrement 0 qui restaient, vous n'aviez pas à compter le nombre de caractères à effacer. Le nombre 99 aurait été aussi valable

que le nombre 16 ; Il vous fallait seulement un nombre au moins aussi grand que le nombre de caractères qui restait dans l'enregistrement 0.

Pour supprimer le contenu entier d'un fichier de texte sans effacer le fichier lui même, utilisez l'instruction CLFL, avec le nom du fichier dans le registre ALPHA. CLFL a besoin d'un nom de fichier, et n'opérera pas sur le fichier de travail. Le fichier désigné devient le fichier de travail, et le nombre d'enregistrements est mis à zéro. Cette opération est utile pour initialiser un fichier de texte préexistant pour le réutiliser comme s'il s'agissait d'un nouveau fichier. L'instruction CLFL est la même que celle qui efface les fichiers de données.

Pour effacer le fichier de texte lui-même et libérer ces registres de mémoire d'extension pour d'autres usages, mettez le nom du fichier dans ALPHA (ceci n'est pas optionnel) et exécutez PURFL. Pour des détails supplémentaires sur PURFL, y compris un important avertissement, voir page 17.

3E. Opérations diverses sur les fichiers de texte : POSFL, SAVEAS, GETAS

La fonction POSFL explore le fichier de texte de travail, à partir de la position courante du pointeur, à la recherche d'une chaîne de caractères identique au contenu du registre ALPHA. Cette chaîne de caractères ne peut déborder d'un enregistrement ; elle doit être contenue entièrement dans un seul enregistrement. Si la recherche aboutit, le pointeur est déplacé au premier caractère de la chaîne et la nouvelle valeur du pointeur est placée dans le registre X. Si la recherche n'aboutit pas, aucun message d'erreur n'est affiché, mais le nombre -1 est placé en X. Par conséquent, si vous utilisez la fonction POSFL dan un programme, une simple instruction "X inférieur à 0 ?" vous dira si la chaîne de caractères n'a pas été trouvée.

POSFL peut travailler conjointement avec DELCHR ou DELREC pour effacer les chaînes de caractères ou les enregistrements, ou conjointement avec INSCHR ou INSREC pour insérer de nouvelles chaînes de caractères ou des enregistrements.

L'usage de la pile par POSFL est tout à fait inhabituel. Sur la 41CX, la pile est levée et LASTX n'est pas perturbée. C'est exactement comme si RCLPT était exécuté au point où la coïncidence est trouvée. Sur la HP-41C ou la CV, POSFL ne fonctionne de cette façon que si la chaîne de caractères est trouvée. Si vous utilisez une HP-41C ou une CV et que la chaîne de caractères n'est pas trouvée, POSFL recouvre le registre X avec le nombre -1 et place le contenu précédent de X dans LASTX.

Essayons un exemple, Supposez que vous vouliez trouver la position du nom "HILL" dans le fichier "NAMES". C'est l'enfance de l'art. Pressez simplement :

```
0 SEEKPT
ALPHA espace H I L L ALPHA
POSFL
```

Le résultat devrait être 1,005, indiquant que le caractère espace devant "HILL" est le caractère numéro 5 de l'enregistrement 1. Le caractère espace a été utilisé pour assurer que "HILL" n'a pas été trouvé comme prénom ou comme chaîne de caractères encastrée dans un autre nom.

Les fonctions SAVEAS et GETAS ne sont utilisables que si vous possédez un appareil de stockage de masse HP-IL, tel que le lecteur de cassette numérique 82161A. Ces fonctions sont décrites dans le mode d'emploi du module de fonctions d'extension mémoire.

Si vous envisagez de faire grand usage des fichiers ASCII, un appareil

de mémoire de masse HP-IL sera très utile. A travers SAVEAS et GETAS, il fournit un moyen commode de sauver de façon permanente vos fichiers ASCII. Si vous devez rassembler les contenus de deux fichiers ASCII, ce pour quoi SAVEAS et GETAS ne sont pas faits, vous pouvez utiliser les programmes présentés dans la section 3G.

3F. Visualisation des contenus d'un fichier ASCII.

Le programme "VAS" (p. 28) affichera le contenu entier d'un fichier de texte, un enregistrement à la fois. Il utilise quelques unes des fonctions d'extension qui sont expliquées au Chapitre 4, aussi vous devrez lire ce chapitre avant d'essayer de comprendre comment "VAS" fonctionne.

Pour visualiser un fichier de texte, mettre le nom du fichier dans le registre ALPHA et exécuter "VAS". Si une imprimante est branchée et allumée, le contenu du fichier ASCII sera imprimé. Autrement il sera affiché. Le sous-programme LBL 10 effectue cette opération d'impression ou d'affichage. C'est une excellente application pour les fonctions d'extension RCLFLAG et STOFLAG.

Ci-joint une impression typique produite par "VAS".

Si vous ne voulez lister qu'une partie du fichier, mettre un compteur d'enregistrements en X dans le format ISG (ddd,fff, où ddd est l'enregistrement de début et fff est l'enregistrement final à visualiser). Mettez le nom du fichier dans ALPHA et exécutez "PVAS".

Un piège à erreur est compris dans "VAS" et "PVAS". Si vous obtenez le message DATA ERROR à la ligne 05, vous devez charger un nom de fichier dans ALPHA, faire BST et R/S. Ce piège à erreur est prévu pour vous empêcher de perdre votre table de mémoire d'extension si vous avez un module de fonctions d'extension mémoire révision 1B et que vous venez juste d'utiliser PURFL. Avec ALPHA effacé, l'instruction SEEKPTA opère sur le fichier de travail, causant un désastre s'il n'y a pas de fichier de travail.

Analyse ligne par ligne de "VAS"/"PVAS"

La ligne 02 fournit un compteur d'enregistrements par défaut de 0,900 pour "VAS", de telle sorte que tous les enregistrements soient affichés. Les lignes 04 et 05 constituent le piège à erreur qui détecte un registre ALPHA vide (longueur de la chaîne en ALPHA = 0). Vous pouvez effacer ces deux lignes si vous possédez une HP-41CX ou une révision 1C ou au-dela du module de fonctions d'extension. Les lignes 07-08 placent le pointeur au commencement du premier enregistrement à visualiser. La séquence LBL 01 forme le message "RECORD n:" dans le registre ALPHA. Ensuite XEQ 10 (ligne 19) affiche ou imprime la chaîne. La séquence LBL 02 utilise GETREC pour rappeler les 24 caractères. Ensuite un XEQ 10 imprime ou affiche la chaîne. Si le drapeau 17 est levé, indiquant un enregistrement incomplet, un autre GETREC est exécuté. Autrement le compteur d'enregistrements est incrémenté (ligne 25). L'instruction GTO 01 provoque le même processus d'exécution pour imprimer l'enregistrement suivant. Quand le compteur atteint sa limite, le GTO 01 est sauté et le RTN est exécuté à la place. Lorsque "VAS" est utilisé, la terminaison sera provoquée par une erreur END OF FL à la ligne 21. Ceci est normal.

3G. Sauvegarde des fichiers de texte sur cartes magnétiques

Si vous possédez un lecteur de cartes, le programme "WAS" présenté ici peut être utilisé pour écrire un fichier de texte dans les registres de données, desquels un WDTAX transfère l'information aux cartes magnétiques.

Le programme "RAS" exécute l'opération inverse. Ces programmes n'ont qu'une seule contrainte : le fichier de texte ne doit contenir aucun caractère nul (code décimal 0). Ce n'est pas une contrainte sérieuse puisque les caractères nuls ne sont ordinairement pas utilisés.

Pour utiliser "WAS", mettre simplement le nom du fichier dans ALPHA et XEQ "WAS". Le nom du fichier doit être fourni pour éviter un arrêt dû à une erreur à la ligne 03. Ce piège à erreur est prévu pour empêcher l'instruction FLSIZE de démolir votre table de mémoire d'extension si vous venez d'utiliser PURFL. Si vous obtenez le message DATA ERROR, vous devez charger un nom de fichier dans ALPHA et presser BST et R/S. Le programme "WAS" s'assurera que la SIZE est suffisante pour contenir toutes les données, en utilisant la fonction d'extension PSIZE pour augmenter le SIZE si besoin est. Si vous obtenez un arrêt sur l'erreur NO ROOM à la ligne 20, vous devrez soit effacer quelques programmes pour faire plus de place soit utiliser le programme "PWAS" décrit ci-dessous. Le nombre en X à l'arrêt dû à l'erreur indique le SIZE demandé pour cette opération "WAS".

Lorsque le lecteur de carte demande : "RDY 01 OF nn", vous pouvez soit insérer la carte à enregistrer soit presser deux fois R/S pour éviter l'enregistrement d'une carte. Lorsque "WAS" est terminé, un nombre de la forme 0,nnn est dans le registre X. Ce nombre indique qu'une représentation des données du fichier de texte réside dans les registres de données de 00 à nnn. Le nombre total des registres de données utilisé est nnn+1, tandis que le nombre de pistes utilisé pour enregistrer les données est 1+INT (nnn/16).

Pour utiliser "RAS", mettre le nom du fichier dans ALPHA (non optionnel) et XEQ "RAS". Fournissez les cartes de données à la demande ou pressez R/S deux fois si la représentation du fichier réside déjà dans les registres de données. Le programme "RAS" sait automatiquement où se terminent les données. Vous n'avez pas besoin de spécifier un nombre de registres.

Les programmes "WAS" et "RAS" sont très utiles pour traiter un problème que l'on trouve habituellement avec les fichiers ASCII. Supposez que vous ayez créé un fichier ASCII de 50 registres et ayez commencé à charger vos données dans le fichier. Tout à coup, vous obtenez le message d'erreur END OF FL. Le fichier est plein ! Il apparaît que vous devez purger ce fichier, en créer un nouveau, plus grand, et recommencer à entrer les données depuis le début. Mais attendez ! Vous pouvez utiliser "WAS" pour écrire les contenus du fichier dans les registres de données avant de purger le fichier. Ensuite, après avoir créé le fichier plus grand, vous pouvez utiliser "RAS" pour recharger les données dans le nouveau fichier. Ceci sauve beaucoup de travail.

Sur la HP-41CX, la fonction RESZFL peut être utilisée à cet effet à la place de "WAS" et "RAS". Choisissez d'abord le fichier que vous voulez redimensionner comme fichier de travail. Vous pouvez le faire en interrompant la table de mémoire d'extension ou en nommant le fichier et en exécutant FLSIZE ou RCLPTA. Puis, mettez la taille du fichier désiré dans X et exécutez RESZFL. RESZFL vous permet d'augmenter ou de diminuer la taille du fichier de travail, dans la mesure où aucun enregistrement n'est perdu par la réduction de taille du fichier.

Attention : Même si vous mettez dans ALPHA un nom de fichier différent, RESZFL continuera à redimensionner le fichier de travail. Vous devez exécuter le catalogue 4 (EMDIR) après avoir utilisé RESZFL pour vérifier le résultat.

Si vous ne voulez enregistrer qu'une partie du fichier de texte sur les cartes, mettez un compteur d'enregistrement de la forme ddd,fff en x,

Liste des programmes « WAS »/« PWAS »/« RAS »/« PRAS »

```
01♦LBL "WAS"
 02 ALENG
 03 1/X
 04 SIZE?
 05 FLSIZE
 06 7
 07 *
 08 APPREC
 09 DELREC
 10 RCLPT
 11 5
 12 *
 13 4
 14 +
 15 +
 16 6
 17 /
 18 INT
 19 X>Y?
 20 PSIZE
 21 0.9
 22 ENTER↑
 23 GTO 00

24♦LBL "PWAS"
 25 ALENG
 26 1/X
 27 X<>Y
 28 SIZE?
 29 2
 30 -
 31 1 E3
 32 /
 33 X<>Y

 34♦LBL 00
 35 ENTER↑
 36 INT
 37 SEEKPTA
 38 SF 25
 39 DSE L

 40♦LBL 01
 41 GETREC
 42 FC? 17
 43 ISG L
 44 FC? 25
 45 GTO 03

 46♦LBL 02
 47 R↑
 48 R↑
49 ASTO IND X
 50 ISG X
 51 X<0?
 52 GTO 04
 53 ALENG
 54 11
 55 ASHF
 56 X<=Y?
 57 GTO 02
 58 RDN
 59 5
 60 X<Y?
 61 FS? 17
 62 GTO 01
 63 GTO 02

 64♦LBL 03
 65 LASTX
 66 R↑
 67 CLA
68 ASTO IND X
 69 INT
 70 1 E3
 71 /
 72 SF 25
 73 WDTAX
 74 CF 25
 75 RTN

 76♦LBL 04
 77 6
 78 R↑

 79♦LBL 05
 80 DSE Z

 81♦LBL 05
 82 CLA
83 ARCL IND Z
 84 RDN
 85 ALENG
 86 X=Y?
 87 GTO 05
 88 DSE L

 89♦LBL 04
 90 ISG Z
 91 ACOS
 92 GTO 03

93♦LBL "RAS"
 94 CF 05
 95 ALENG
 96 1/X
 97 CLFL
 98 .9
 99 SIGN
100 GTO 08

101♦LBL "PRAS"
 102 CF 05
 103 ALENG
 104 1/X
 105 RDN
 106 ENTER↑
 107 INT
 108 SF 25
 109 SEEKPTA
 110 FC? 25
 111 GTO 07

 112♦LBL 06
 113 DELREC
 114 FC? 25
 115 GTO 07
 116 ISG Y
 117 GTO 06
 118 CF 25

 119♦LBL 07
 120 APPREC
 121 DELREC
 122 RCLPT
 123 X<>Y
 124 SF 25
 125 SEEKPT
 126 CF 25
 127 X<Y?
 128 SF 05

 129♦LBL 08
 130 SF 25
 131 RDTA

 132 CF 25
 133 CLX
 134 SF 06
 135 LASTX
 136 6

 137♦LBL 09
 138 CLA
139 ARCL IND Z
 140 ALENG
 141 X=0?
 142 FC? 06
 143 X<0?
 144 RTN
 145 FC? 06
 146 APPCHR
 147 FC?C 06
 148 CLA
 149 FS? 05
 150 INSREC
 151 FC? 05
 152 APPREC
 153 X≠Y?
 154 SF 06
 155 X≠Y?
 156 FC? 05
 157 GTO 10
 158 RDN
 159 RCLPT
 160 INT
 161 ISG X

 162♦LBL 09
 163 SEEKPT

 164♦LBL 10
 165 RDN
 166 ISG Z
 167 ACOS
 168 FS? 06
 169 ISG Y
 170 GTO 09
 171 END


  291 BYTES
```

le nom du fichier en ALPHA, et exécutez "PWAS". Les enregistrements débutant avec ddd et finissant par fff seront copiés dans les registres de données, et dans les cartes magnétiques si vous choisissez cette manière. C'est utile lorsqu'une taille insuffisante est disponible pour "WAS", ou lorsque vous mélangez les parties de deux fichiers de texte différents.

Si vous voulez remplacer une partie des données dans un fichier de texte avec des données à partir de cartes, vous pouvez utiliser le programme "PRAS".

Mettez un numéro de contrôle d'enregistrement en X, le nom du fichier en ALPHA, et exécutez "PRAS". Les enregistrements de ddd à fff seront effacés et remplacés par les données à partir des cartes ou des registres de données. Si vous voulez ajouter des enregistrements à la fin d'un fichier, utilisez un numéro de contrôle d'enregistrements xx,9, où xx est plus grand que le nombre de registres dans le fichier, et 900-xx est plus grand que le nombre de registres à ajouter. Pour trouver le nombre d'enregistrements dans le fichier, voir le problème 3.1 à la page 41.

Analyse ligne par ligne de "WAS"/"PWAS"/"RAS"/"PRAS"

Les lignes 02 et 03 s'assurent que le registre ALPHA n'est pas vide, de façon que la fonction FLSIZE à la ligne 05 ne fonctionne pas sur le fichier de travail (qui pourrait ne pas exister). Voir page 47 pour une explication de la fonction ALENG. Si vous voulez être capable d'utiliser "WAS" et "RAS" avec le fichier de "travail", vous pouvez effacer les lignes 02, 03, 95, et 96. Si vous avez un module de fonctions d'extension révision 1B, voir page 17 pour une explication du risque que vous prenez en effaçant ces pièges à erreurs.

Les lignes de 05 à 18 calculent le nombre de registres de données nécessaire pour stocker la représentation du fichier de texte. Cette représentation est mieux démontrée par un exemple. Supposez que vous ayez le fichier de texte :

numéro d'enregistrement	Nom
0	RICHARD J.NELSON
1	ROGER HILL
2	CLIFFORD STERN

La représentation de ce fichier généré par "WAS" serait :

Registre de données	Contenus
00	"RICHAR"
01	"D J. N"
02	"ELSON"
03	"ROGER "
04	"HILL"
05	"CLIFFO"
06	"RD STE"
07	"RN"
08	"" (chaîne vide)

La fin de chaque enregistrement est marquée par une chaîne de moins de 6 caractères. Ce marqueur de fin d'enregistrement sera une chaîne vide si le nombre de caractères dans l'enregistrement est un multiple de 6. Une chaîne vide au début d'un nouvel enregistrement (registre 08 dans cet exemple) signifie la fin du fichier.

Cette représentation particulière du fichier de texte représente un bon compromis entre la vitesse et l'utilisation des registres. Un meilleur

compactage des données n'est possible que par l'utilisation des techniques de programmation synthétiques (voir Section 10I). Le nombre de registres de données nécessaires pour représenter un fichier de texte dépend de la taille N du fichier (en registres) et du nombre d'enregistrements R dans le fichier. Le programme a besoin de calculer une limite supérieure du nombre de registres de données voulu, de façon à ajuster la taille assez largement. Le cas plus critique est lorsque les R-1 premiers enregistrements ont 6 caractères chacun, ce qui signifie qu'ils prennent chacun 2 registres de données, et que le dernier enregistrement utilise tout l'espace qui reste dans le fichier. Dans ce cas, un long calcul montre que le nombre total de registres de données nécessaires pour stocker les données du fichier de texte ne peut excéder :

$$D = 2(R-1)+1+INT((7(N-R)+10)/6)$$
$$= INT((7N+5R+4)/6).$$

Dans "WAS", la ligne 05 calcule la taille du fichier N ; tandis que les lignes 08-10 calculent le nombre R d'enregistrements dans le fichier. Ce dernier calcul demande qu'il y ait plus de 8 caractères non utilisés dans le fichier de texte, de façon qu'un enregistrement comprenant le nom du fichier puisse être temporairement ajouté sans provoquer l'erreur END OF FL. Les lignes 19 et 20 comparent le nombre maximum de registres de données demandé avec la taille courante, et redimensionnent en cas de besoin. Tous ces registres ne seront probablement pas utilisés, mais c'est le prix à payer pour la commodité de l'utilisateur. La ligne 21 place le compteur d'enregistrements par défaut (0,9) de façon que tous les enregistrements soient écrits.

La séquence LBL 00 place le pointeur au début du premier enregistrement désigné. Le drapeau 25 est levé de façon que le GETREC sur la ligne 41 n'arrête pas le programme.

La boucle LBL 01 va chercher un enregistrement dans le fichier. Si il rencontre le END OF FL, GETREC efface le drapeau 25 et provoque le branchement GTO 03. Dans le cas contraire la boucle LBL 02 est utilisée pour stocker le contenu de registre ALPHA en éléments de 6 caractères. D'abord les 6 caractères de gauche sont stockés dans le registre 00. La ligne 43 incrémente le compteur d'enregistrements dans LASTX dés que la récupération de chaque nouvel enregistrement est établie. Les lignes 53-57 retournent à LBL 02 pour stocker 6 autres caractères si 12 caractères ou plus étaient présents dans ALPHA avant que le ASHF à la ligne 55 n'enlève les 6 qui viennent d'être stockés. S'il n'y avait pas plus de 12 caractères, alors au plus un ASTO supplémentaire sera nécessaire avant le prochain GETREC. S'il y avait 5 caractères ou moins, l'enregistrement est complet et a déjà été stocké. Dans ce cas, la ligne 60 provoque l'exécution de GTO 01. Si plus de 5 caractères étaient présents, une autre opération ASTO sera généralement nécessaire. La seule exception se produit quand le flag 17 est levé, indiquant que GETREC a récupéré les 24 caractères sans atteindre la fin de l'enregistrement. Dans ce cas, les 6 derniers de ces 24 caractères viennent d'être ASTO et nous n'avons pas besoin de stocker un marqueur de fin de fichier blanc.

Ainsi le test du drapeau 17 à la ligne 61 nous renvoie chercher plus de caractères de l'enregistrement courant. Si le drapeau 17 n'est pas levé, l'enregistrement est complet et il nous faut absolument un marqueur de fin d'enregistrement (qui contiendra de 0 à 5 caractères). L'instruction GTO 02 à la ligne 62 s'occupe ce cas. LBL 03 marque le début de la procédure de terminaison qui se produit après que le END OF FL soit atteint par GETREC ou après que le ISG L à la ligne 41 arrive à une situation de saut. Le

marqueur de fin de fichier (une chaîne vide) est stocké dans le registre de données courant, de façon que "WAS" sache où les données de "RAS" se terminent. Ensuite le nombre 0,nnn est construit pour l'utilisation de WDTAX. Le programme "RAS" commence par vérifier que le registre ALPHA n'est pas vide. Le fichier de texte désigné est effacé, ce qui place automatiquement son pointeur à l'enregistrement 0. Le compteur d'enregistrements par défaut de 0,9 est placé dans LASTX par l'instruction SIGN. "PRAS" commence de façon similaire, mais le fichier n'est pas effacé. Si le SEEKPTA à la ligne 109 échoue, le programme considère que les nouveaux enregistrements sont à ajouter, et qu'aucun enregistrement n'a besoin d'être effacé. Autrement la boucle LBL 06 efface le nombre d'enregistrements demandés par le compteur d'enregistrements qui était initialement placé en X. Le drapeau 25 est testé pour le cas où vous auriez spécifié trop d'enregistrements et ou vous rencontreriez le END OF FL. Le drapeau 05 est levé à la ligne 128 si le premier enregistrement désigné est dans le fichier, plutôt qu'à la fin du fichier. Cela signifie que INSREC sera utilisé plus tard à la place de APPREC.

Au LBL 08, les cartes sont lues (si on le désire) et le drapeau 06 est levé, indiquant que le prochain registre à lire commence un nouvel enregistrement. La valeur 0 en Z (ligne 133) est le pointeur de registre initial, la valeur en Y (ligne 135) est le compteur d'enregistrements pour ISG, et le 6 en X (ligne 136) doit être utilisé pour les comparaisons avec ALENG. Lorsque la longueur de la chaîne du registre de données est inférieure à 6, la fin d'un enregistrement a été atteinte. La boucle LBL 09 prend d'abord une chaîne de 0 à 6 caractères provenant du registre de données courant. Si la longueur est 0 et que le drapeau 06 est levé, indiquant que ce registre est censé commencer un nouvel enregistrement, alors le RTN à la ligne 144 termine "RAS". Dans le cas contraire, si le drapeau 06 est baissé, la fonction APPCHR à la ligne 146 ajoute les contenus ALPHA à l'enregistrement courant. Si le drapeau 06 est levé, APPREC ou INSREC est exécuté, selon le drapeau 05, pour utiliser le contenu de ALPHA pour commencer un nouvel enregistrement. Si la longueur de la chaîne n'était pas exactement de 6 caractères, alors le drapeau 06 est levé pour indiquer que la nouvelle chaîne rappelée commencera un nouvel enregistrement. Les lignes 155-164 avancent le pointeur au nouvel enregistrement si les drapeaux 05 et 06 sont levés, de façon que le prochain INSREC place le prochain enregistrement à la bonne place.

Le compteur de registres en Z est alors incrémenté de façon que le registre suivant puisse être rappelé.

3H. Fonctions additionnelles de fichier de texte sur la HP-41CX

La HP-41CX inclut 2 fonctions additionnelles dédiées aux fichiers de texte. La première d'entre elles est ASROOM. ASROOM retourne le nombre d'octets disponibles dans le fichier désigné, ou le fichier de travail si ALPHA est effacé. Si vous avez un fichier auquel vous n'ajouterez pas d'informations fréquemment, vous pouvez utiliser la séquence suivante pour minimiser cet usage des registres de mémoire étendue :

```
(nom de fichier)
FLSIZE              donne le nombre de registres alloués
ASROOM              donne le nombre d'octets disponibles
7
/
INT                 Nombre de registres disponibles
-                   nombre de registres utilisés
```

RESZFL Redimendimensionner au minimum.

Si vous possédez une HP-41C ou une CV, vous pouvez utiliser "WAS" et "RAS" pour réduire la taille du fichier au minimum, mais vous pouvez avoir à utiliser le petit programme qui suit pour dupliquer la fonction ASROOM dans la séquence précédente :

```
01 LBL "ASROOM"
02 ALENG          Ces lignes sont un piège à erreur
03 1/X            pour le parasite PURFL. Vous pouvez
04 RDN            les supprimer si vous avez la révision 1C ou plus
05 FLSIZE         Nombre de registres dans le fichier désigné
06 7
07 *
08 0
09 SEEKPTA        allez au début du fichier.
10 +              Nombre total d'octets dans le fichier.
11 SF 25          empêche l'arrêt sur l'erreur ligne 14.
12 LBL 01
13 CLA
14 GETREC
15 ALENG          Soustrait le nombre de caractères dans
16 -              cet enregistrement.
17 FC? 17         Soustrait un octet pour chaque enregistrement,
18 DSE X          un octet à la fin du fichier.
19 FS? 25
20 GTO 01         répéter si END OF FL n'est pas atteint.
21 END
```

Cette routine donne le vrai ASROOM dans la mesure où il n'y a pas d'octet nul dans le fichier de texte.

La seconde fonction de fichier de texte de la HP-41CX est ED. La fonction ED est décrite pleinement dans le manuel de l'utilisateur de la HP-41CX, et la description est trop longue pour la répéter ici. Lorsque vous exécutez ED, le clavier est redéfini pour permettre un déplacement commode à travers le fichier aussi bien qu'une insertion et un effacement des données.

Si vous possédez une HP-41C ou une CV, le chapitre 9 présente un programme éditeur de texte appelé "TE" qui, tout en étant plus lent que ED, en contient toutes les caractéristiques plus quelques autres. Vous trouverez "TE" ou ED tout à fait utiles pour la création et la modification des fichiers de textes.

PROBLEMES (les solutions suivent le chapitre 10)

3.1 Ecrire une courte séquence d'instructions pour déterminer combien d'enregistrements il y a dans un fichier de texte (considérez que le fichier est le fichier de travail).

3.2 Ecrire un court programme pour imprimer un fichier de texte entier, un enregistrement par ligne (à moins que l'enregistrement ne dépasse la ligne d'impression). Considérez que le nom du fichier est dans le registre ALPHA au début du programme.

decimal code	display char	printer char	decimal code	display char	printer char
0	(null)		32	(space)	(space)
1			33	!	!
2			34	"	"
3			35	#	#
4			36	$	$
5			37	%	%
6			38	&	&
7			39	'	'
8			40	(	(
9			41	)	)
10			42	*	*
11			43	+	+
12			44	,	,
13			45	-	-
14			46	.	.
15			47	/	/
16			48	0	0
17			49	1	1
18			50	2	2
19			51	3	3
20			52	4	4
21			53	5	5
22			54	6	6
23			55	7	7
24			56	8	8
25			57	9	9
26			58	:	:
27			59	;	;
28			60	<	<
29			61	=	=
30			62	>	>
31			63	?	?

decimal code	display char	printer char	decimal code	display char	printer char
64	@	@	96	`	`
65	A	A	97	a	a
66	B	B	98	b	b
67	C	C	99	c	c
68	D	D	100	d	d
69	E	E	101	e	e
70	F	F	102	f	f
71	G	G	103	g	g
72	H	H	104	h	h
73	I	I	105	i	i
74	J	J	106	j	j
75	K	K	107	k	k
76	L	L	108	l	l
77	M	M	109	m	m
78	N	N	110	n	n
79	O	O	111	o	o
80	P	P	112	p	p
81	Q	Q	113	q	q
82	R	R	114	r	r
83	S	S	115	s	s
84	T	T	116	t	t
85	U	U	117	u	u
86	V	V	118	v	v
87	W	W	119	w	w
88	X	X	120	x	x
89	Y	Y	121	y	y
90	Z	Z	122	z	z
91	[	[	123	{	{
92	\	\	124	|	|
93	]	]	125	}	}
94	↑	↑	126	~	~
95	_	_	127	←	←

NOTES SUR LA TABLE DES CARACTERES ASCII

Si vous utilisez une imprimante HP-IL, les codes décimaux 9, 10 et 27 ont une signification différente. Le code 9 génère un caractère "line feed" (aller à la ligne), le code 10 génère un "retour chariot" et le code 27 un caractère "échappement". Le caractère "échappement" signifie que les caractères suivants constituent un message de contrôle spécial pour l'imprimante. Ce message n'est pas imprimé. Le mode échappement est abandonné automatiquement aprés réception d'un nombre de caractères ASCII suffisant pour constituer une commande valide.

Les codes décimaux 128-255 donnent des novas (tous segments allumés) à l'affichage. Les caractères de l'imprimante pour les codes 128-255 sont les mêmes que pour 0-127, respectivement, sauf pour les trois codes de contrôle.

Le code décimal 0 donne un caractère "nul", qui est sans rapport avec le message "NULL" obtenu quand on maintient la pression sur une touche. A moins que vous fassiez beaucoup de programmation synthétique, il est probable que vous n'utiliserez jamais de caractère nul. Lisez l'appendice C du manuel du module de fonctions d'extension mémoire pour un examen complet des effets étranges qu'il peut produire.

Le caractère 255 a également quelques propriétés étranges. Quand il est affiché dans le registre ALPHA, il apparait comme une nova. Cependant si vous ASTOquez une chaine qui contient ce caractère puis affichez cette chaine, vous verrez un affichage surprenant. Le caractère 255 et tous les caractères qui le suivent seront invisibles. Si vous avez un module de fonctions d'extension révision 1B, vous devez observer une précaution au sujet du caractère 255 : ne stockez pas plus de 6 caractères 255 consécutifs dans un fichier texte. Vous risquez de perdre ce fichier et tous les fichiers suivants la prochaine fois que vous purgerez un fichier. Ceci est du au fait que la HP-41 utilise un registre de 7 caractères 255 pour marquer le dernier registre utilisé des mémoires d'extension. Ce problème n'existe pas pour les modules révision 1C ou plus ni pour la HP-41CX.

CHAPITRE QUATRE

AUTRES FONCTIONS D'EXTENSION

Toutes les fonctions incorporées dans le module de fonctions d'extension mémoire ou incorporées dans la HP41CX ne sont pas directement concernées par l'utilisation de la mémoire d'extension. 16 parmi les 47 fonctions (25 sur 61 pour la HP-41CX) assurent un renforcement du système d'exploitation qui aide considérablement à traiter les chaînes ALPHA, les drapeaux, les blocs de données, et les assignements de touches. Une seule fonction, GETKEY, a la capacité de permettre l'adaptation complète du clavier sous le contrôle d'un programme. Ceci est démontré dans les programmes d'application au chapitres 7, 8, et 9.

4A. Usage de la Pile et flexibilité d'entrée.

Il y a une importante différence entre les fonctions d'extension et les fonctions normales incorporées (catalogue 3), qui n'est pas mentionnée dans le manuel de l'utilisateur. La plupart des fonctions d'extension qui utilisent une entrée à partir du registre X laissent simplement l'entrée en X quand elles sont exécutés (X<>F, POSA, POSFL, et GETKEYX sont les seules exceptions). A l'exception de POSA, POSFL, et GETKEYX, elles ne copient même pas X dans LASTX. A cet égard, les fonctions d'extension ressemblent d'avantage aux fonctions indirectes comme ARCL IND X qu'aux fonctions directes telles que 1/x. Cette différence dans l'usage de la pile est facile à traiter dans vos programmes dès que vous la connaissez. Au pire il vous faudra une instruction supplémentaire de rotation de la pile vers le bas ici et là pour vous débarasser d'une entrée de fonction utilisée. Au mieux vous pourrez faire usage du fait que LASTX n'est pas perturbé en y conservant un compteur de boucle.

Celles des fonctions d'extension qui placent un résultat en X travaillent exactement comme RCL. La pile est levée, à moins qu'une fonction CLX, ENTER (ou autre fonction qui empêche la montée de la pile) vienne d'être exécutée. Il y a deux exceptions à cette règle. La première exception est POSA, qui recouvre X et sauve la valeur précédente de X dans LASTX. La seconde exception, qui ne s'applique qu'à la HP-41C ou la CV, est POSFL. Sur la HP-41C ou la CV, lorsque POSFL ne trouve pas la chaîne, X est recouvert et la valeur précédente de X est sauvée dans LASTX. Si la chaîne est trouvée, ou si vous utilisez une HP-41CX, la pile est levée et LASTX reste intact.

Un autre trait caractéristique commun aux fonctions d'extension est qu'elles ignorent tout chiffre en X au delà de ceux qui sont normalement demandés. Souvent cela signifie que la partie fractionnaire de X est ignorée. Par exemple, si vous voulez utiliser STOFLAG pour restaurer l'état des drapeaux 36-39, le nombre en X peut être 36,39xxxxxx, où les chiffres xxxxxx peuvent être différents de zéro. Un cas dans lequel cette caractéristique des fonctions d'extension peut être utile peut se trouver à la page 26. Là , la séquence :

```
"nom de fichier"
0
SEEKPTA
,023
SAVERX
```

était mentionnée comme une manière de sauver les registres de données de 0 à 23 dans le fichier de données. Parce que les pointeurs de fichiers de données sont toujours des nombres entiers, l'instruction SEEKPTA aurait ignoré toute partie fractionnaire du nombre en X. Donc vous auriez pu utiliser la séquence :

```
"nom de fichier"
,023
SEEKPTA
SAVERX
```

qui est un pas et un octet plus court. Un autre avantage et que la fonction SEEKPTA est réellement plus rapide avec ,023 en X qu'avec 0 en X. Ce n'est pas souvent que des situations de la sorte se produisent, mais si vous vous vous rappelez de cette flexibilité d'entrée des fonctions d'extension, vous pourrez écrire des programmes plus efficaces. Une autre caractéristique de la flexibilité d'entrée des fonctions d'extension est que les nombres négatifs sont d'ordinaire traités comme s'il s'agissait de nombres positifs.

Les exceptions sont AROT, discuté en section B de ce chapitre, et les fonctions RESZFL et GETKEYX. Ces deux dernières fonctions utilisent le signe de X comme drapeau pour passer outre à un piège à erreur et choisir un mode différent d'opération, respectivement. Un avantage possible de l'intérêt de ne pas tenir compte du signe est que les valeurs de pointeur négatives sont traitées comme si elles étaient positives. Vous pouvez donc simuler une instruction "décrémenter et sauter si inférieur à zéro" en utilisant un nombre entier négatif avec une instruction ISG. Incrémenter un nombre négatif décrémente sa valeur absolue.

4B. Manipulation ALPHA

Une HP-41C "nue" ou une CV a des capacités alphabétiques très limitées. Avec juste un registre ALPHA de 24 caractères et un jeu primitif d'opérations alpha (append, ASTO, ARCL, ASHF, etc.), ces capacités alpha sont bien adaptées à l'affichage d'un message mais inadéquates pour autre chose. La mêmoire d'extension y ajoute la possibilité de stocker des fichiers de texte (ensembles de chaînes ALPHA) et ajoute des instructions pour changer, rappeler ou trouver sélectivement une chaîne. De plus, il y a six fonctions relatives à AlPHA dans le jeu des fonctions d'extension, qui opèrent directement sur les contenus du registre ALPHA plutôt que sur des chaînes à l'intérieur d'un fichier de texte. Ces six fonctions, ALENG, ANUM, AROT, ATOX, POSA, et XTOA, ajoutent une capacité importante, mais ne permet toujours pas un traitement alphanumérique extensif.

Si vous vous rappelez que la HP-41 n'est pas prévue pour être capable de traiter les mots, vous réaliserez que ses capacités ALPHA, spécialement avec l'addition des fonctions d'extension, sont plus qu'adéquates pour son affichage de 12 caractères. La fonction AROT assure la rotation des contenus du registre ALPHA vers la gauche du nombre de positions de caractères spécifié en X. Un nombre négatif en X produit une rotation vers la droite. La valeur absolue de X doit être inférieure à 256, où un message DATA ERROR en résultera (nous employons ici le mot "rotation" par commodité, en termes exacts il faudrait dire "permutation circulaire").

L'usage primaire de AROT est d'amener un caractère selectionné à une extrémité de la chaîne ALPHA. Par exemple, un caractère sélectionné amené à l'extrémité gauche de ALPHA peut être décodé par la fonction ATOX (voir

ci-dessous). Un seul caractère qui vient d'être ajouté à l'extrémité droite de ALPHA peut être déplacé de sa position initiale à la fin de la chaîne, en début de chaîne par la séquence 1, CHS, AROT.

La fonction XTOA ajoute un seul caractère à la partie extrême droite de ALPHA. Ce caractère est désigné par un nombre décimal allant de 0 à 255 en X. Ce nombre décimal est appelé le code ASCII équivalent du caractère. La correspondance d'affichage et de représentation d'impression du code décimal ASCII est indiqué dans le tableau pages 60 et 61. Si ALPHA contient déjà 23 ou 24 caractères, un TONE est émis. Cet avertissement est effectué même si XTOA est utilisé dans un programme, à moins que le drapeau 26 ne soit baissé. Si X contient des données ALPHA, XTOA se comportera comme ARCLX, ajoutant les caractères à l'extrémité droite de ALPHA. Il est préférable d'utiliser ARCL X dans ce cas parce que sa signification est plus claire dans un listing de programme. La pile n'est pas baissée par XTOA et LASTX n'est pas mis à jour non plus.

La fonction XTOA peut être utilisée pour construire des chaînes ALPHA comportant des caractères qui n'existent pas au clavier tels que les parenthèses et le &. Par exemple, la séquence suivante crée la chaîne "X(1)=" dans le registre ALPHA :

```
"X"
40
XTOA
"┣1"
1
+
XTOA
"┣="
```

(notez que ces deux pas utilisent le fait que XTOA ne baisse pas la pile)

Une instruction ARCL et une instruction AVIEW peuvent alors être utilisée pour ajouter un nombre et afficher le message. XTOA peut également être utilisé pour former des chaînes comportant des caractères minuscules ou spéciaux pour imprimante, bien que la fonction de l'imprimante ACCHR fasse la même chose. A l'exception de a-e, ces caractères apparaîtront comme des novas dans l'affichage. Ceci est dû à la limitation d'un affichage à 14 segments.

Si le contenu de la chaîne est connu à l'avance, les techniques de programmation synthétique vous permettent de mettre une ligne de texte dans votre programme qui comporte tous les caractères spéciaux de cette sorte. Ceci est beaucoup plus efficace que d'utiliser XTOA. Voir La Programmation Synthétique, c'Est Facile !. XTOA est mieux adapté pour le rajout d'un ou deux caractères à ALPHA, là où le caractère réel à ajouter dépend du résultat d'un calcul dans le programme. Cette technique est utilisée dans la partie conversion de base du programme de la HP-16 au chapitre 9.

La fonction ATOX est presque l'inverse de XTOA. XTOA convertit un nombre décimal en caractère qui est ajouté à la droite du registre ALPHA. Au contraire, ATOX convertit le caractère à la gauche du registre ALPHA en code décimal correspondant. La pile est levée, mais LASTX reste intact. En plus de son usage primaire pour le décodage d'un caractère à partir d'ALPHA, ATOX est souvent utilisé simplement pour effacer un caractère du registre ALPHA. Une opération AROT peut être utilisée pour déplacer tout caractère désiré à l'avant de la chaîne dans ALPHA, où ATOX peut le supprimer et le décoder.

La fonction ALENG calcule le nombre de caractères dans le registre ALPHA, de 0 à 24. Ce nombre est placé dans le registre X, tandis que les premiers contenus de X, Y, et Z sont montés en Y, Z, et T. LASTX demeure

inchangé. Par exemple ; supposez que vous voulez vérifier si le registre ALPHA est vide et vous brancher si c'est le cas. La séquence : ALENG X=0? GTO 99 accomplira ceci. Si vous voulez voulez simplement générer un message d'erreur si ALPHA est vide, vous pouvez utiliser la séquence ALENG 1/X (donne DATA ERROR si X=0). Un autre usage pour ALENG est de déterminer combien d'opération ASTO et ASHF sont nécessaires pour stocker une longue chaîne ALPHA. Puisque chaque ASTO stocke 6 caractères (et que ASHF supprime alors ces 6 caractères) nous pouvons diviser la longueur par 6 et arrondir à l'entier supérieur le plus voisin pour déterminer combien d'ASTO seront nécessaires. Une autre approche est de vérifier la longueur après chaque ASHF et continuer tant que le registre ALPHA n'est pas vide.

Une application avancée de ALENG doit faciliter la rotation des chaînes contenant des caractères nuls. Si un caractère nul est déplacé à l'avant (la partie la plus à gauche) d'une chaîne, il disparaîtra. La seule façon de pouvoir dire que ceci s'est produit est de vérifier avec ALENG avant et après la rotation pour voir si la longueur a diminué. A moins que vous fassiez un bon nombre de programmation synthétique (voir section 10A), vous n'utiliserez probablement pas ALENG de cette manière. Mais, maintenant, quand vous voyez ALENG précédant et suivant AROT dans un programme synthétique, vous saurez pourquoi.

La fonction POSA accepte un code de caractère décimal en X. Elle cherche alors dans la chaîne contenue dans ALPHA, de gauche à droite, la première occurence du caractère spécifié. Un code de position est retourné au registre X, recouvrant le code de caractère. Le code de caractère est sauvé dans LASTX. POSA, POSFL , et GETKEYX sont les seules fonctions d'extension qui modifient LASTX.

Le code position retourné par POSA est un nombre entier allant de 0 à 23. Une valeur de 0 indique une occurence au premier caractère de ALPHA (le plus à gauche), tandis qu'une valeur de 23 indique un occurence au 24ième caractère. Si aucune occurence n'est trouvée, la valeur -1 est placée en X. Ces règles pour le code de position peuvent sembler étranges, mais elles sont concues avec une application spécifique à l'esprit.

Si vous voulez localiser un caractère particulier et le placer à l'avant de ALPHA, vous pouvez utiliser la séquence très simple : (code caractère) POSA X<0? SF 99 (si le caractère n'est pas trouvé alors affichez "NONEXISTENT") AROT. Le caractère localisé peut alors être supprimé par un ATOX.

Si vous avez des caractères séparateurs dans le registre ALPHA, POSA, AROT, ATOX fonctionnant ensemble peuvent trouver les séparateurs, les supprimer et préparer le registre ALPHA pour chaque chaîne séparée à traiter. Comme exemple de POSA, supposez que vous avez le nom de quelqu'un dans le registre ALPHA sous la forme standard "prénom nom", et que vous voulez le changer en "nom, prénom". La séquence suivante devrait faire l'affaire :

```
"ROGER HILL"        (par exemple)
"┣, "              place une virgule et un espace aprés le nom
32                  code décimal pour l'espace
POSA                localise l'espace après "ROGER"
AROT                le place à l'avant de la chaîne
ATOX                efface l'espace.
Résultat : "HILL, ROGER"
```

La fonction POSA possède un second mode de fonctionnement qui permet de rechercher dans le registre ALPHA une chaîne de 1 à 6 caractères. Au lieu de placer un code de caractère décimal en X, vous pouvez par exemple,

essayez la séquence suivante :

```
"FGH"              (pressez ALPHA F G H ALPHA)
ASTOX              (Pressez ALPHA shift STO . 6 ALPHA)
"WXYXABCDEFGHIJ"
POSA               (XEQ ALPHA P O S A ALPHA)
```

Le résultat devrait être 9, indiquant que la chaîne "FGH" commence au 10ième caractère de ALPHA. Notez que la chaîne "FGH" est toujours disponible dans LASTX si vous en avez besoin. Dans ce second mode, POSA est très similaire à la fonction POSFL. Parce qu'elle n'est pas limitée aux sous chaînes de 6 caractères ou aux chaînes de 24 caractères, la fonction POSFL est beaucoup plus utile que POSA pour rechercher les sous-chaînes. Cependant, pour la recherche d'un seul caractère, soit par le code décimal soit par une sous-chaîne d'un seul caractère, la fonction POSA est souvent plus simple à utiliser que POSFL. La fonction ANUM est presque l'inverse de ARCL. Le premier usage de ARCL est d'ajouter un nombre au registre ALPHA. La fonction ANUM extrait un nombre du registre ALPHA. Par exemple, si le registre ALPHA contient la chaîne "A=452", exécuter ANUM mettra le résultat 452 dans le registre X. La pile est levée et LASTX reste intact.

ANUM cherche dans le registre ALPHA de gauche à droite, retournant le premier nombre légitime qui a été trouvé. Ceci est affecté par les virgules et les points dans ALPHA, et l'état des drapeaux 28 et 29. Lorsque le registre ALPHA contient un ou plusieurs point ou virgules, les choses commencent à se compliquer. D'abord, le point et la virgule sont interprétés selon l'état des drapeaux 28 et 29. Si les drapeaux 28 et 29 sont levés, un point est interprété comme un point décimal et une virgule comme un séparateur de chiffres. Si le drapeau 28 est baissé et le 29 levé, une virgule est interprétée comme décimale et un point comme séparateur. C'est la notation européenne standard. Si le drapeau 29 est baissé, les séparateurs de chiffre (virgule si le drapeau 28 est levé, point si le drapeau 28 est baissé) sont traités comme des caractères ALPHA. Par conséquent si le nombre "12,003.05" est dans ALPHA et que vous exécutiez ANUM avec les drapeau 28 levé mais avec le drapeau 29 baissé, le résultat sera 12. Parce que le drapeau 29 est baissé, la virgule est considérée comme un caractère, séparant le nombre en deux parties. Pour plupart des applications, vous devriez essayer d'éviter ce problème en vous assurant que le drapeau 29 est levé avant d'utiliser ANUM. Une autre précaution à prendre : si vous utilisez ANUM avec un format de nombre non standardisé, les résultats peuvent ne pas être ce vous escomptiez. Par exemple "-34-" XEQ "ANUM" donne le résultat positif 34. Le second signe négatif annule l'effet du premier. Egalement, deux nombres ou plus séparés seulement par les symboles + ou - seront interprêtés comme un seul nombre.

PROBLEMES

4.1 Ecrire une séquence de 4 pas pour ajouter un caractère à la gauche du registre ALPHA

4.2 Ecrire une courte séquence pour ASTOquer les contenus du registre ALPHA sans perdre aucun registres sur les chaînes vides.

4.3 Ecrire les séquences pour effacer n caractère de ALPHA :
a) à gauche, b) à droite.

4.4 Modifier la séquence de rotation ci-dessus "nom, prénom" pour tenir

compte d'une possible initiale intermédiaire.

4C. Manipulations de drapeaux

Les fonctions d'extension procurent trois nouvelles fonctions qui sont très utiles pour le contrôle de l'état des drapeaux. Les plus importantes sont RCLFLAG et STOFLAG.

Considérez la situation suivante. Vous êtes en train d'écrire un programme qui nécessite d'arrondir ou d'afficher un résultat sous un certain format, par exemple FIX 2. Vous voudriez que ce programme soit capable de restituer le format d'affichage d'origine avant de renvoyer le contrôle à l'utilisateur.

Avant l'arrivée des fonctions d'extension, cette tâche, simple en apparence, était très difficile à faire. Des tests de drapeaux élaborés étaient nécessaires au début du programme pour déterminer la mise en état d'origine de l'affichage. Ensuite, après avoir changé la disposition de l'affichage, d'autres opérations étaient nécessaires pour rétablir la disposition originelle.

La disponibilité des fonctions RCFLAG et STOFLAG élimine toute cette difficulté. Vous utilisez simplement RCFLAG pour rappeler la disposition des drapeaux avant de changer l'affichage, ensuite utilisez STOFLAG pour rétablir l'état d'origine des drapeaux. Une séquence typique d'instructions peut ressembler à ceci :

```
RCLFLAG         Place une chaîne équivalente aux drapeaux en X
"AMT=$"
FIX 2
ARCL 01
AVIEW
STOFLAG         Utilise la chaîne pour rétablir les drapeaux.
```

Le sous-programme LBL 92 du programme "NAP" du chapitre 7 est un exemple de cette technique. Maintenant voyons quelques détails supplémentaires au sujet des fonctions RCLFLAG et STOFLAG.

La fonction RCLFLAG rappelle dans le registre X une chaîne alpha inintelligible qui représente l'état courant des drapeaux 0 à 43. Comme une instruction standard RCL, RCLFLAG lève les contenus de X, Y, et Z en Y, Z, et T, à moins d'être immédiatement précédée d'ENTER, CLX ou d'une autre opération interdisant la montée de la pile. LASTx reste intact. La chaîne alpha formée par RCLFLAG peut être stockée dans un registre de données ou conservée dans la pile. Le seul usage de cette chaîne est de rétablir plus tard une partie ou la totalité de l'état des drapeaux en utilisant STOFLAG.

La fonction STOFLAG possède deux modes d'opération. Celui illustré dans l'exemple ci-dessus est le plus simple. Placez simplement les données alpha de RCLFLAG en X, et exécutez STOFLAG. L'état d'origine des drapeaux 0 à 43 est rétabli (l'état où ils étaient lors de l'exécution de RCLFLAG). Le second mode d'opération de STOFLAG permet un rétablissement sélectif de l'état antérieur des drapeaux. Pour utiliser ce mode, placez la chaîne alpha de RCLFLAG dans le registre Y, et un nombre de la forme dd,ff (non pas dd,fff) en X. Puis, lorsque vous exécutez STOFLAG, le bloc de drapeaux du numéro de drapeau dd au numéro de drapeau ff (y compris dd et ff) sera rétabli à son état d'origine. Pour rétablir un seul drapeau, placez le numéro de drapeau dd en X.

Ce second mode d'opération vous permet, par exemple, de rétablir seulement l'état d'affichage, les drapeaux à usage général, ou simplement le mode trigonométrique. Pour rétablir seulement l'état d'affichage, vous devriez utiliser un séquence telle que :

```
RCLFLAG          Sauve l'état originel des drapeaux
STO 05           dans le registre de données 05
.
.                (pas de programme altérant l'affichage)
RCL 05           Ramener la chaîne de RCLFLAG
36.41            les drapeaux 36-41 contrôlent l'état de l'affi-
                 chage
STOFLAG          Ne rétablit que les drapeaux 36-41.
```

En utilisant RCLFLAG et STOFLAG, il est possible d'avoir plusieurs jeux de drapeaux adaptés aux différentes sections d'un programme. Chaque disposition de drapeau peut être stockée dans un registre de données séparé sous la forme d'une chaîne alpha de RCLFLAG. Chaque section du programme peut alors simplement utiliser RCLFLAG pour rétablir l'état des drapeaux, plutot que de traiter les drapeaux individuellement. Cela devrait remarquablement accélérer l'exécution des programmes.

Une autre application de RCLFLAG/STOFLAG est le court programme suivant qui imprimera les contenus du registre ALPHA si l'imprimante est allumée et validée (drapeau 21 levé), ou autrement sera équivalent à AVIEW et PSE. Ceci est supérieur à un simple AVIEW parce que :

1) Il ne s'arrête pas si le drapeau 21 est levé mais que l'imprimante est éteinte et

2) Il ne vous force pas à attendre le lent défilement de l'affichage si l'imprimante est allumée.

Cette séquence était utilisée dans le programme "VAS" de la section 3F. Voici un programme, "PVA" :

```
01 LBL "PVA"
02 SF 25
03 PRA           essayer d'imprimer ALPHA
04 RCLFLAG
05 FS?C 21       Baisser le drapeau 21 pour un AVIEW ultérieur
06 FC? 25        si l'impression n'était pas réussie,
07 AVIEW         ou si le drapeau était baissé, alors utilisez
                 AVIEW.
08 STOFLAG       Rétablir les drapeaux
09 RDN
10 FS?C 25       si l'impression n'était pas réussie,
11 FC? 21        ou si l'imprimante n'était pas validée,
12 PSE           alors faites PSE après le AVIEW.
13 END
```

Une autre application de RCLFLAG/STOFLAG vous permet d'obtenir un format d'affichage FIX/ENG en disposant des drapeaux 40 et 41. Ce mode d'affichage ressemble à un format FIX normal jusqu'à ce que le nombre en X soit assez grand ou assez petit pour qu'un exposant soit nécessaire. Alors le mode ENG prends le dessus. mettez simplement un nombre de 0 à 9 en X, et exécutez "FEX" pour placer le mode FIX/ENG avec le nombre spécifié de chiffres affichés à la droite du chiffre le plus signicatif.

```
01 LBL "FEX"          FIX/ENG INDirect X
02 ENG 0              lève le drapeau 41
03 RCLFLAG
04 FIX IND Y          lève le drapeau 40 et choisit
05 X<>Y               le nombre correct de chiffres
```

```
06 RDN
07 41
08 STOFLAG              lève le drapeau 41 (les autres demeurent
                        inchangés)
09 R↑
10 R↑                             remet la pile en ordre.
11 END
```

La troisième fonction relative au drapeau est X<>F. Cette fonction traite les drapeaux à usage général de 00 à 07 comme "mini registre", et réalise un échange avec ce registre. Ce mini registre ne peut contenir que des nombres entiers de 0 à 255 inclus. Par conséquent toute partie fractionnaire de X est rejetée avant la réalisation de l'échange, et le signe de X est ignoré. En effet, la fonction X<>F incorpore la séquence ABS, INT comme ses deux premiers pas, à l'exception de LASTX qui n'est pas modifié. En fait, la séquence X<>F, X<>F peut être utilisée pour réaliser ABS, INT sur un nombre inférieur à 255 sans modifier la pile ou LASTX. Aucun message DATA ERROR n'est donné par X<>F à moins que INT (ABS(X)) soit plus grand que 255.

La puissance de X<>F est que, comme STOFLAG et RCLFLAG, il vous donne la possibilité de maintenir plusieurs jeux de drapeaux à usage général dans les registres de données.

Après l'exécution de X<>F, l'état des drapeaux de 00 à 07 exprimé sous forme binaire la première valeur de X. Si vous êtes portés sur les mathématiques la formule est :

$$X \text{ antérieur} = \sum_{i=0}^{7} f_i * 2^i$$

ou f_i=1 si le drapeau i est levé, 0 si le drapeau i est baissé.

Dans cette représentation binaire, le drapeau 0 a la valeur 1 ; le drapeau 1 a la valeur 2, le drapeau 2 a la valeur 4 et ainsi de suite. cette équivalence peut être représentée sous forme de tableau. L'exemple indiqué ci-dessous donne la représentation binaire du nombre décimal 133.

Numéro de drapeau	valeur si levé	levé?	valeur courante
00	1	0	1
01	2	N	0
02	4	0	4
03	8	N	0
04	16	N	0
05	32	N	0
06	64	N	0
07	128	0	128
		total :	133

Comme simple exemple de l'utilité de X<>F, supposez que vous avez un programme qui commence par effacer les drapeaux de 00 à 03 et lever les drapeaux 04 et 05. Plutôt que d'utiliser la séquence :

```
CF 00
CF 01
```

```
CF 02
CF 03
SF 04
SF 05      (12 octets)
```

on peut utiliser la séquence :

```
48              (drapeau 04 = 16, drapeau 05 = 32)
X<>F            (4 octets).
```

Si vous n'êtes pas familier avec les équivalences binaires, vous pouvez vérifier que 48 est le nombre correct comme suit :

```
0               ceci efface les drapeaux de 00 à 07
X<>F            une technique très utile.
SF 04
SF 05
X<>F
```

Le résultat de cette séquence est 48. Ceci montre que le nombre 48, lorsque qu'il est suivi de X<>F, lèvera les drapeaux 04 et 05, tout en effaçant les autres. Si il était très important dans l'exemple ci-dessus de conserver l'état des drapeaux 06 ET 07, vous auriez pu utiliser cette séquence :

```
0
X<>F        Rappelle l'état des drapeaux
64
/           Les drapeaux 06 et 07 sont maintenant dans les
INT         chiffres 1 et 2
LASTX
*
48          Ces deux pas additionnent le nombre
+           pour lever les drapeaux 04 et 05.
X<>F
```

Une meilleure analyse de cette séquence est laissée comme exercice. Lorsque que vous le comprendrez, vous serez capable d'utiliser pleinement X<>F. Mais ne comettez pas l'erreur d'utiliser X<>F partout. Par exemple, dans l'exemple que l'on vient de montrer, un simple jeu de six instructions pour baisser les drapeaux 00-03 et lever les drapeaux 04-05 sauve deux octets par rapport à la méthode X<>F.

PROBLEMES

4.5 Ecrivez une séquence d'instructions qui évalue la fonction :

$$f(x)=\frac{SIN(PI*x)}{PI*x}$$

la fonction SIN doit être évaluée en mode RADian, mais le mode trigonométrique d'origine doit être rétabli.
4.6 Ecrire une séquence pour activer le mode FIX/ENG sans changer le nombre de chiffres courants (drapeaux 36-39).

Applications en programmation synthétique de RCLFLAG et STOFLAG

Lorsqu'une imprimante est en ligne, l'exécution du programme est

ralentie. Le ralentissement peut être réduit si vous baissez synthétiquement le drapeau 55. Le drapeau 55 restera baissé tant que le programme continuera à tourner. Dés qu'il s'arrêtera, les drapeaux 55 et 21 seront tous deux levés. La courte séquence suivante, développée par Steve Wandzura, baisse le drapeau 55 sans perturber aucun autre drapeau important :

```
RCLFLAG
SIGN          Stocke les drapeaux dans LASTX, fait X=0.
STO d         Baisse tous les drapeaux.
X<>L          Ramène les drapeaux en X.
STOFLAG       rétablit les drapeaux (jusqu'à 43).
RDN           rétablit la pile (à l'exception de T).
```

Le format exact de la chaîne ALPHA générée par RCLFLAG est, en hexadécimal :

1F Fd dd dd dd dd dd

où les d indiquent une information sur les drapeaux, correspondant aux drapeaux 0 à 43, de gauche à droite. Les drapeaux sont donc déplacés d'un octet et demi vers la droite de leur position normale dans le registre des drapeaux. Le déplacement supplémentaire d'un demi-octet peut être utile dans les applications avancées de la programmation synthétique.

4D. Les fonctions relatives à la SIZE

Deux des fonctions d'extension vous permettent de vérifier et d'ajuster la taille sous le contrôle du programme. C'est une nouvelle et puissante possibilité qui, avant l'introduction du module de fonctions d'extension mémoire, n'était disponible qu'à travers les techniques de programmation synthétique.

La fonction SIZE? trouve le nombre de registres de données actuellement attribués et place ce nombre dans le registre X. Ainsi, par exemple, si vous avez une SIZE de 020, et que vous exécutez SIZE?, le résultat sera le nombre 20 en X. La pile est levée comme pour une opération RCL.

La fonction SIZE? est l'exemple classique de la fonction système essentielle que les concepteurs avaient négligé dans la HP-41 d'origine. Si vous avez utilisé une HP-41 sans fonctions d'extension, vous le savez déjà. Combien de fois avez vous voulu vérifier la SIZE courante avant de commencer une entrée de données manuelle ou par programme? La procédure habituelle était d'essayer plusieurs opérations RCL dans le but d'arriver à une idée approximative de l'endroit où se trouve le premier registre NONEXISTENT. Cette procédure pourrait être automatisés par programme comme ce programme simple mais lent :

```
01 LBL "SZFIND"
02 CLX                  Ces deux lignes font X=0 et lèvent
03 SF 25                le flag 25 pour éviter l'arrêt ligne 05.
04 LBL 01
05 RCL IND X            Tentative pour rappeler un registre.
06 FC? 25               Si le registre était NONEXISTENT,
07 RTN                  la valeur en X est le SIZE.
08 RDN
09 1
10 +
11 GTO 01          Ajoute 1 au numéro de registre,
12 END                  ensuite essaye le suivant.
```

La fonction SIZE? est incomparablement plus rapide que cette approche, et beaucoup plus pratique aussi. Vous pourriez l'utiliser assez souvent pour que ça vaille la peine de l'assigner à une touche, mais même si vous ne le faites pas, elle est vite accessible par la séquence de touche : XEQ ALPHA S I Z E ? ALPHA.

La fonction PSIZE fait la même chose que SIZE, sauf qu'elle ne donne pas la familière demande par trois traits.

A la place, le SIZE est ajusté pour égaler la valeur en X. PSIZE peut être utilisée dans un programme en cours d'exécution, même dans un sous-programme de niveau six, sans effet néfaste pour l'exécution du programme. Cela signifie que vous pouvez écrire des programmes et des sous-programmes, qui augmentent ou diminuent automatiquement la taille comme nécessaire.

La courte séquence suivante vérifie si la SIZE courante est suffisante pour un but spécifique, et utilise PSIZE pour augmenter la taille si nécessaire.

```
(SIZE demandée)
SIZE?
X<>Y
X>Y?
PSIZE
```

Une autre variante donne un avertissement audible de l'opération PSIZE imminente, au cas où l'utilisateur de programme veut presser R/S pour empêcher de redimensionner :

```
(SIZE demandée)
SIZE?
X< Y?
GTO 01
TONE 9
X<>Y
PSE
PSIZE
LBL 01
```

4E. Opérations sur les blocs

Les fonctions d'extension REGMOVE et REGSWAP vous permettent de copier, d'échanger, où d'assurer la rotation de blocs de registres de données. La HP-41CX y ajoute la fonction CLRGX, qui efface un bloc de registres de données. Si vous possédez une HP-41C ou une CV, un court programme "d'effacement de bloc" fait le même travail.

La fonction REGMOVE accepte une entrée de la forme sss,dddnnn dans le registre X. L'exécution de REGMOVE copie un bloc de nnn registres commençant au registre sss vers un bloc de nnn registres commençant au registre ddd. Si nnn est zéro, un seul registre est copié. REGMOVE ne modifie pas la pile ou LASTX. Comme exemple, la séquence : 6,001003 REGMOVE, copie un bloc de 3 registres. Le bloc source est les registres 06, 07, et 08, tandis que le bloc destination est composé des registres 01, 02, et 03.

Pour faciliter les exemples à suivre, placez d'abord la SIZE à 020 et exécutez le programme "PRELOAD" de la page 21.

Ceci préparera tous vos registres de données. Lorsque vous pressez : XEQ "PRELOAD", la valeur 0 est stockée dans le registre 00, 1 dans le

registre 01, et ainsi de suite. La valeur dans chaque registre équivaut à son numéro. Comme exemple simple de REGMOVE, pressez : 3,007006 XEQ ALPHA R E G M O V E ALPHA. Cela entraînera la copie des registres 03-08 (bloc de 6 registres) dans les registres 07-12 comme indiqué ci-dessous.

Registres :	03	04	05	06	07	08	09	10	11	12
début :	3	4	5	6	7	8	9	10	11	12
										8
									7	
								6		
							5			
						4				
					3					
Résultat :	3	4	5	6	3	4	5	6	7	8

Les pas intermédiaires indiqués dans ce diagramme sont invisibles pour vous, ils sont inclus de façon que vous puissiez visualiser la manière dont le processus de copie est implémenté. Là où il n'y a pas d'entrée, le contenu des registres n'est pas changé à ce pas.

La fonction REGSWAP, échange les contenus des deux blocs de registres de données. Comme REGMOVE, elle accepte un nombre sous la forme sss,dddnnn en X, où sss dénote le commencement du bloc source, ddd le commencement du bloc destination, et nnn le nombre de registres dans chaque bloc. Si nnn est zéro, la HP-41 considère que vous voulez nnn=1 et elle n'échange que les registres sss et ddd. La pile et LASTX demeurent inchangés.

La programmation interne de REGSWAP échange une paire de registres à la fois. Si sss<ddd, le registre dont le numéro est le plus élevé est échangé en premier et le registre dont le numéro est le plus bas est échangé en dernier. Si sss>ddd, le registre dont le numéro est le plus bas est déplacé en premier et le registre dont le numéro est le plus élevé est déplacé en dernier. Cet ordre interne d'opérations est le même pour REGSWAP que pour REGMOVE. Normalement vous n'auriez pas besoin de savoir dans quel ordre ces opérations sont réalisées. Cependant, si les blocs source et destination sont imbriqués, l'ordre des opérations affecte le résultat. Comme exemple de REGSWAP, essayez ceci :

```
XEQ "INIT"
3,007006 XEQ "REGSWAP".
```

Le diagramme suivant montre comment l'échange des registres est réalisé.

Registres :	03	04	05	06	07	08	09	10	11	12
début :	3	4	5	6	7	8	9	10	11	12
						12				8
					11				7	
				10				6		
			9				5			
		12				4				
	11				3					
Résultat :	11	12	9	10	3	4	5	6	7	8

Comme vous pouvez le voir, REGSWAP peut réellement brouiller les registres quand il y a une imbrication significative des deux blocs. Cette caractéristique peut être transformée en avantage, cependant, en construi-

sant une fonction de "rotation de bloc". Considérez l'exemple suivant :

```
XEQ "PRELOAD"  (initialise les registres)
4,003009 XEQ "REGSWAP"
```

Les pas internes dans l'échange des registres sont montrés ci-dessous. Parce que 4 est plus grand que 3, l'échange procéde des registres inférieurs aux registres supérieurs.

Registres :	03	04	05	06	07	08	09	10	11	12
Début :	3	4	5	6	7	8	9	10	11	12
	4	3								
		5	3							
			6	3						
				7	3					
					8	3				
						9	3			
							10	3		
								11	3	
									12	3
Résultat :	4	5	6	7	8	9	10	11	12	3

Le résultat est que le bloc de 10 registres de 03 à 12 est permuté d'un registre vers le bas. Si vous aviez fait : 3,004009 XEQ "REGSWAP", le bloc de 10 registres aurait été permuté d'un registre vers le haut. Ce résultat peut être facilement généralisé. Pour assurer la permutation d'un bloc de registres nnn commençant au registre sss, utilisez l'entrée REGSWAP :

sss,(sss+1)(nnn-1) pour permuter d'un registre vers le haut
(sss+1),sss(nnn-1) pour permuter d'un registre vers le bas.

Si vous voulez assurer la permutation d'un bloc de n registres de r registres vers le haut ou vers le bas, vous pouvez être capable d'accomplir le résultat voulu avec une seule instruction REGSWAP. Si le nombre r divise n exactement (sans reste), utilisez :

sss,(sss+r)(nnn-r) pour permuter de r registres vers le haut
(sss+r),sss(nnn-r) pour permuter de r registres r vers le bas.

La fonction CLRGX sur la HP41CX accepte une entrée de la forme ddd,fffii dans le registre X, où ddd est le premier registre à effacer, ii est l'incrément entre les registres à effacer, et les registres au-dessus de fff ne sont pas perturbés. Si ii n'est pas fourni (ii=0), alors une valeur par défaut de ii=1 est prise, de sorte que les registres de ddd jusqu'à fff compris sont effacés. Si vous êtes familier avec l'instruction ISG sur la HP-41, ces règles ne seront pas nouvelles pour vous. Par exemple, pour effacer les registres de 04 à 08, vous devez presser : 4,008 XEQ ALPHA C L R G X ALPHA, pour effacer les registres 01, 03, 05, 07, et 09 vous devez presser : 1,00902 XEG ALPHA C L R G X ALPHA.

En fait il est très rare d'avoir à utiliser une valeur ii différente de 0. Par exemple quand vous avez stocké une matrice, une entrée par registre, dans un bloc de registres de données. Les valeurs de ii différentes de 0 permettent d'effacer sélectivement une seule colonne ou les éléments d'une diagonale. CLRGX laisse son entrée en X et ne perturbe pas LASTX.

Si vous possédez une HP-41C ou une CV, il est facile d'écrire une séquence d'instructions pour effacer un bloc de registres. Le programme suivant très simple effacera un bloc de registres de données commençant avec le registre ddd et se terminant avec le registre fff. Mettez

Liste du programme « ALSORT »

```
01♦LBL "ALSORT"      11 RCL X        21 GTO 03       32 GTO 02
    02 ENTER↑        12 RCL Z     22 X<> IND Y
    03 ISG Y         13 INT       23 STO IND L       33♦LBL 03
    04 X<0?          14 +          24 FS?C 06        34 R↑
    05 RTN           15 DSE X      25 GTO 03         35 R↑
    06 INT           16 X<0?       26 RDN            36 ISG Y
    07 1 E3          17 SF 06      27 ABS            37 GTO 01
    08 /          18 RCL IND L  28 RCL IND X        38 END
                                   29 DSE Y
    09♦LBL 01        19♦LBL 02     30 FS? 53
    10 CF 06         20 X>=NN?     31 SF 06       71 BYTES
```

Liste du programme « VREG »

```
01♦LBL "VREG"        09 CHS          19 -            28 GETKEYX
  02 CF 21           10 GETKEYX      20 X<0?         29 RDN
                     11 SIGN         21 CLX          30 X≠0?
  03♦LBL 01          12 X<>Y      22 VIEW IND X      31 GTO 03
  04 "REG?"          13 X=0?         23 .1           32 GTO 01
  05 AON             14 GTO 02       24 CHS          33 END
  06 AVIEW           15 X=Y?         25 SIGN
                     16 RTN
  07♦LBL 02          17 LASTX        26♦LBL 03      57 BYTES
  08 10              18 64           27 X<> L
```

Liste du programme « BC Σ »

```
01♦LBL "BCΣ"         08♦LBL 01       14 DSE X        19♦LBL 03
  02 6 E-5           09 CLΣ                          20 STO IND Y
  03 +            10 ΣREG IND X      15♦LBL 02        21 ISG Y
04 ΣREG IND X        11 ISG X        16 LASTX         22 GTO 03
  05 ISG X           12 GTO 01       17 -             23 END
  06 X<0?                            18 0
  07 GTO 02          13♦LBL 02                          44 BYTES
```

simplement le nombre ddd,fff en X et exécutez "BC".

```
01 LBL"BC"
02 SIGN              Stocke ddd,fff en LASTX
03 CLX               la valeur 0 à stocker.
04 LBL 03
05 STO IND L         Efface le registre.
06 ISG L             incrémente le compteur.
07 GTO 03
08 END
```

Si vous envisagez d'effacer de larges blocs de registres de données, vous pouvez utiliser le programme plus rapide "BCΣ" qui utilise la fonction CLΣ pour effacer 6 registres à la fois. Pour effacer 100 registres, "BC" utilise 13 secondes, tandis que "BCΣ" prend moins de quatre secondes. Pour utiliser "BCΣ", mettez simplement le numéro de contrôle ddd,fff en X et exécutez "BCΣ".

PROBLEME

4.7 Ecrivez un court programme pour assurer la rotation d'un bloc de nnn registres commençant au registre sss et allant vers le haut par rrr registres (vers le bas si rrr est négatif). Au début du programme, considérez que sss est en X, nnn est en Y, et rrr est en Z. N'utilisez que la pile et LASTx. (Ce n'est pas aussi facile que ça en à l'air).

4F. Contrôle d'assignement de touches

Deux fonctions d'extension, CLKEYS et PASN, augmentent vos capacités de contrôle des assignements de touches du mode USER. Une autre fonction d'extension, GETKEY (plus GETKEYX sur la HP-41CX), permet à votre programme de "lire" le clavier, assurant le nec plus ultra dans la redéfinition du clavier. D'abord quelques mots sur les assignements. La capacité d'assigner une fonction à une seule touche est un des traits qui distingue les meilleures calculatrices programmables. Les calculatrices programmables HP-65 et HP-67 avaient une rangée supérieure de touches étiquetées de A à E (la rangée supérieure shiftée était étiquetée de a à e). Un appui sur une de ces touches, vous permettait d'exécuter une section de programme de la calculatrice commencé avec le label correspondant (A-E ou a-e).

La HP-41 est une avancée considérable par rapport à ses prédécesseurs dans la capacité d'assignements des touches. Le mode USER de la HP-41 permet virtuellement à chaque touche d'être redéfinie avec un assignement d'un label global ou d'une fonction. Les labels globaux sont ces labels qui apparaissent dans le catalogue 1, tandis que les fonctions apparaissent dans le catalogue 2 ou le catalogue 3. Aussi, pour maintenir la compatibilité avec la HP-67 et la HP-97, la rangée supérieure de touches peut accéder aux labels locaux A-E (non shiftée) et a-e (shiftée) dans le programme courant. De plus, la seconde rangée non shiftée peut accéder aux labels locaux F-J. Ceci fonctionnera tant qu'il n'y aura pas de label global ou de fonction assignée à la touche en question. Cette propriété de recherche automatique de label est décrite sous le nom "labels locaux" dans le manuel d'utilisation de la HP-41. Si une touche située dans la rangée supérieure, shiftée ou non shiftée, ou dans la seconde rangée, non shiftée seulement, est pressée en mode USER, et qu'aucun label global ou fonction n'est assigné à cette touche, une recherche est entamée. Si le label local correspondant (de A à J ou de a à e) est trouvé dans le programme courant,

la calculatrice commence à exécuter le programme en ce point. Maintenez la touche baissée pour visualiser sa fonction.

Incidemment, comme cette recherche de label local peut prendre un temps relativement long, il est utile d'assigner les fonctions X<>Y et RDN à leur propres touches. Cet assignement a une priorité plus grande qu'un label local, aussi aucune recherche de label n'est réalisée.

La fonction PASN fonctionne à peu-près comme la fonction ASN à partir du clavier. Rappelez vous-en lorsque vous utilisez ASN, vous devez entrer dans le mode ALPHA et épeler un nom de fonction. C'est la même chose avec PASN, sauf que vous épelez le nom de fonction dans le registre ALPHA avant d'exécuter PASN. Avec ASN, vous désignez la touche à laquelle la fonction doit être assignée en pressant effectivement la touche après avoir épelé le nom de fonction si vous maintenez la touche baissée pendant un moment, un code de touche apparait à l'affichage. Ce code de touche est nombre à deux chiffres. Le premier chiffre est le numéro de rangée de la touche (de 1 à 8) tandis que le second chiffre indique la colonne (de 1 à 5). C'est ce code de touche rang-colonne que vous avez à mettre dans le registre X avant d'exécuter PASN.

La fonction ASN peut être utilisée pour effacer manuellement une touche de son assignement. Vous n'avez simplement qu'à presser ALPHA ALPHA pour le nom de fonction. Lorsqu'aucune fonction n'est désignée, la HP-41 considère que vous voulez que la touche soit libérée de tout assignement. Une fois de plus, PASN fonctionne de façon similaire. Assurez vous simplement que le registre ALPHA est vide, mettez le code de touche rangée-colonne en X, et exécutez PASN.

En résumé : pour utiliser PASN, chargez le registre ALPHA du nom de la fonction à assigner, mettez le code de touche rangée-colonne en X, et exécutez PASN. La fonction spécifiée sera assignée à la touche désignée. Si le registre ALPHA est vide, l'assignement de la touche désignée sera effacé. Ces instructions s'appliquent de façon identique si PASN est une instruction dans un programme ou si elle est exécutée au clavier.

La fonction PASN vous permet d'écrire des programmes qui font des assignements de touche. Par exemple, supposez que vous ayiez un programme pour mettre à jour des fichiers de textes. Il s'avèrerait très utile d'avoir des opérations telles que INSREC et DELREC assignées, même si ces fonctions ne sont pas assez utiles pour rester assignées en permanence. La réponse est d'utiliser PASN au début de votre programme pour assigner ces fonctions aux touches commodes. A la fin du programme vous pouvez utiliser PASN à nouveau, avec le registre ALPHA vide, pour effacer les assignements.

Le court programme listé ci-contre a été écrit par Alan McCornack. Il efface de la rangée supérieure (non shiftée seulement) toute fonction ou assignement de label global. Cette technique peut facilement être étendue selon vos besoins à l'effacement sélectif d'assignements.

```
01 LBL "CT3
02 CLA
03 11,015       Compteur ISG pour les codes touches 11 à 15
04 LBL 05
05 PASN         efface la touche X.
06 ISG X
07 GTO 05
08 END
```

La fonction CLKEYS efface tous les assignements des labels globaux ou des fonctions. Notez que lorsque vous utilisez CLKEYS pour effacer les assignements de fonctions, les pseudo-assignements de label locaux (les

touches A-J et a-e) ne seront plus masqués par la présence de label global ou d'assignement de fonction de priorité plus grande. CLKEYS est une solution drastique aux problèmes de conflits d'assignements. Dans la plupart des cas vous avez meilleur compte d'utiliser PASN pour effacer ou réassigner des touches individuelles. La section 10G vous propose même une meilleure méthode.

Voici une application typique de PASN et CLKEYS. Supposez que vous ayez deux jeux différents d'assignements que vous aimez utiliser avec votre HP-41, selon ses usages. Vous pouvez écrire deux programmes, pour la mise en place de chaque jeu d'assignements. Chaque programme aurait cette forme générale:

```
LBL "KB1"
CLKEYS              Elimine les assignements précédents
"fonction 1"
(code de touche 1)
PASN
"fonction 2"
(code de touche 2)
PASN
"fonction 3"
(code de touche 3)
PASN
.
.
.
END
```

Puisque PASN, comme ASN, recouvrira tout assignement précédent sur la touche désignée, vous pouvez trouver que CLKEYS est inutile ici, surtout si plusieurs assignements pour les différents claviers sont de la même fonction, ou utilisent les mêmes touches. Si vous voulez effacer sélectivement les assignements, incluez une séquence telle que :

```
CLA
(code de touche 1)
PASN
(code de touche 2)
PASN
etc.
```

dans le programme "KB1".

La troisième fonction d'extension qui est en relation avec les assignements est GETKEY. C'est une fonction très spéciale qui est peut-être plus puissante que tout autre fonction, comme vous le verrez aux chapitres 8 et 9. La fonction GETKEY est un type de fonction entièrement nouveau pour la HP-41. Lorsque vous exécutez GETKEY comme partie d'un programme, la calculatrice fait une pause de dix secondes en attendant que vous pressiez une touche. Si une touche est pressée, le code de touche rangée/colonne pour cette cette touche est placée dans le registre X. Si aucune touche n'est pressée dans les 10 secondes, le nombre 0 est placé en X. Dans les deux cas, la pile est levée et LASTX n'est pas perturbé. Si vous voulez que le programme continue à attendre jusqu'à ce qu'une touche soit pressée, une simple boucle fera l'affaire :

```
LBL 00
GETKEY
X=0?
GTO 00
```

Aussi lontemps qu'il n'y aura pas de touche pressée, ce segment de programme continuera à boucler. Pour éviter de lever la pile en cas d'insuccès de GETKEY, vous pouvez utiliser une instruction RDN entre LBL 00 et GETKEY. Contrairement à PASN, la fonction GETKEY a des codes de touches pour les quatre bascules supérieures. Pour GETKEY, ces touches sont assignées à un numéro de rangée de 0. La touche ON a un code de touche 01 et la touche ALPHA a un code 04. rappelez- vous que ces codes n'apparaissent que comme résultats de GETKEY ; ils ne travailleront pas avec PASN. Lorsque vous utilisez GETKEY, évitez les séquences telles que :

```
LBL 00
GETKEY
GTO 00
```

L'omission du test X=0? entraîne une boucle infinie. Mais vous ne pouvez pas arrêter celle-ci en pressant simplement R/S. Après tout, R/S n'est que la touche 84. Même presser la touche ON ne l'arrêtera pas. La seule façon de l'arrêter est d'enlever les batteries. Une meilleure façon est de presser et de maintenir la touche R/S, presser la touche ON, lacher la touche R/S, et lâcher la touche ON. Le meilleur parti est de s'assurer que vous avez une façon normale de sortir de la boucle GETKEY. Avec GETKEY, un programme peut simuler les assignements de labels locaux sur les 35 touches non shiftées. De plus, le programme peut faire ceci sans risque de conflit avec d'autres assignements de touches et sans la nécessité de mise en place du mode USER. L'interprétation de chaque touche peut même changer à l'intérieur d'un programme.

Le type de séquence le plus communément utilisé avec GETKEY est :

```
LBL 00
RDN
GETKEY
X=0?                    si aucune touche n'est pressée,
GTO 00                  alors essayez à nouveau.
XEQ IND X               Exécutez un sous-programme correspondant
.                       à la touche qui a été pressée.
.                       Cette portion du programme affiche
.                       des résultats ou exécute d'autres
.                       opérations qui sont les mêmes pour
                        toutes les touches.
RTN ou GTO 00
LBL 11                  cette section est exécutée si la touche
.                       de la rangée 1, colonne 1, était pressée.
.
RTN
.                       utilisez un LBL pour chaque touche à laquelle
.                       vous voulez que votre programme réponde.
.
END
```

Cette séquence attend jusqu'à ce qu'une touche soit pressée, puis

exécute tout pas de programme suivant le label local numérique correspondant. Par exemple, si vous deviez presser la touche d'effacement, la HP-41 chercherait le LBL 44 (rangée 4, colonne 4) et se mettrait à exécuter cette portion du programme comme sous-programme. Si vous voulez que plus de touches aient des fonctions, ajoutez simplement les labels numériques correspondants au programme, suivis des séqences qui font tout ce que vous voulez que fasse la touche. Les assignements n'entrent pas en conflit avec GETKEY, parce que la fonction GETKEY a priorité sur eux temporairement, de la même façon qu'elle a priorité sur les touches on/off et de sélection de mode. GETKEY peut fonctionner comme un autre niveau d'assignement à votre disposition.

Voici un exemple simple de la façon dont GETKEY peut être utilisé. Cette séquence demande une réponse par oui ou par non. R/S ou "0" (la touche de multiplication) sont acceptées comme une réponse positive, toute autre touche donne une réponse négative.

```
"message"
SF 25
LBL 00
AVIEW
GETKEY
RDN             met le code de touche dans le registre T de la
                pile.
GTO IND T
.               (une réponse "non" arrive ici)
.
RTN
LBL 71          (les réponses "0" viennent ici)
LBL 42          (les réponses R/S viennent ici)
CF 25
.
.
RTN
```

L'instruction GTO IND T se branche vers l'arrière en direction de LBL 00 si aucune touche n'a été pressée. Autrement il n'y a pas de LBL correspondant au code de touche. Ceci entraîne une erreur NONEXISTENT qui baisse le drapeau 25. L'exécution alors arrive dans la séquence de réponse par NON. Cette technique de GETKEY demande implicitement, comme le font la plupart des usages de GETKEY, qu'il n'y ait pas de LBL local imprévu qui ait un numéro ressemblant à code de touche. Cela signifie qu'à l'intérieur de ce programme, les labels suivants ne sont pas autorisés, sauf à l'endroit où un tel label est nécessaire comme l'objet d'une instruction GTO IND :

```
01 02   03 04       (les bascules)

11 12 13 14 15      (rang 1)
21 22 23 24 25      (rang 2)
31 32 33 34 35      (rang 3)
41    42 43 44      (rang 4)
51    52 53 54      (rang 5)
61    62 63 64      (rang 6)
71    72 73 74      (rang 7)
81    82 83 84      (rang 8)
```

Naturellement, vous pouvez violer cette contrainte dans vos programmes si vous ne vous préoccupez pas des résultats invalides lorsqu'une touche illégale est pressée en réponse à la demande GETKEY. Cette sorte de technique pour tester une réponse par oui ou par non est utilisée dans le programme d'adresses au chapitre 7. Deux options supplémentaires de réponse sont ajoutées, mais le principe est le même. Un exemple beaucoup plus élaboré de GETKEY est donné au chapitre 9, où un programme est présenté qui simule les seules fonctions de conversion de base de la calculatrice HP-16.

4G. Fonctions supplémentaires sur la HP-41CX

La HP-41CX inclut 14 fonctions d'extension de plus que le module de fonctions d'extension mémoire pour 41C ou CV. Six de ces fonctions ont déjà été décrites: EMROOM (page 12), EMDIRX (page 12), RESZFL pages 27 et 36), ED (page 41), ASROOM (page 40), et CLRGX (page 57). Les huit autres fonctions, GETKEYX, REG, X<NN?, X≤NN?, X=NN?, X≠NN?, X>=NN? et X>NN? seront décrites dans cette section.

La fonction GETKEYX est une extension de la fonction GETKEY. Un nombre en X jusqu'à 99,9 spécifie le nombre de secondes que la calculatrice attendra qu'une touche soit pressée quand vous exécuterez GETKEYX. Si X est inférieur à 0.1 la calculatrice fera de son mieux, mais il se peut que vous attendiez légèrement plus longtemps que ce que vous pensiez. GETKEYX retourne un code de touche au registre Y (non au registre X) et un code de caractère au registre X, comme expliqué ci-dessous. L'intervalle qui était spécifié en X est sauvegardé en LASTX, tandis que les premiers contenus des registres Y et Z de la pile sont levés à Z et T respectivement.

Si vous pressez une touche non shiftée dans le temps spécifié, le code de touche est placé dans le registre Y. Si vous pressez la touche shiftée, le code de touche 31 (rangée 3 colonne 1) n'est pas retourné comme il le serait si vous aviez utilisé GETKEY. A la place, la calculatrice recommence l'intervalle spécifié et attend qu'une autre touche soit pressée. Lorsque vous pressez la seconde touche, le négatif de son code de touche est placé dans le registre Y. Cette caractéristique vous permet d'obtenir les touche shiftées aussi bien que les touches non shiftées.

Si l'intervalle de temps expire avant qu'une touche soit pressée, la valeur zéro est placée en Y pour indiquer qu'aucune touche n'a été pressée. La valeur retournée au registre X dépend du statut du mode ALPHA (drapeau 48) et de quelle touche est pressée. Si le mode ALPHA est actif (drapeau 48 levé) et que vous pressiez une touche (ou shift plus une autre touche) qui correspond à un caractère, le code ASCII équivalent au caractère est retourné en X. Ceci rend simple de créer une copie du caractère choisi dans le registre ALPHA -utilisez une seule instruction XTOA.

Si le mode ALPHA est inactif (drapeau 48 baissé) lorsque vous utilisez GETKEYX, les codes ASCII ne sont retournés que pour les touches de chiffres, la touche virgule et la touche CHS. Une fois de plus, ce code vous permet de crêer une copie de la touche choisie dans le registre ALPHA simplement en utilisant XTOA. Si la touche pressée ne correspond pas à un caractère ALPHA (mode ALPHA actif), la valeur zéro est retournée au registre X.

Si vous spécifiez un intervalle de temps négatif dans le registre X pour GETKEYX, la calculatrice n'attendra pas que la touche soit relachée, comme il se doit normallement. Au lieu de cela, l'exécution s'arrêtera dès

que le contact électrique sera établi. Ainsi un séquence de la sorte :

```
LBL 01
.
.
.
-,1
GETKEYX
RDN
X=0?
GTO 01
```

continuera à boucler aussi longtemp que toute touche est maintenue baissée. Il y a peu de chance que vos applications vous amènent à utiliser cette caractéristique de GETKEYX, mais si vous devez l'utiliser, il y a un problème possible dont vous devriez prendre conscience. Bien que l'instruction GETKEYX lise effectivement le code de touche correct, le relâchement de la touche provoque l'exécution normale de la fonction. Pour toutes les touches à l'exception des touches ON et R/S, rien ne se produira parce que le programme est en train de fonctionner. En revanche, le relâchement de la touche ON éteindra la calculatrice, et le relâchement de la touche R/S arrêtera le programme. Ainsi lorque vous exécutez GETKEYX avec un nombre négatif en X, évitez de presser les touches R/S ou ON à moins que vous ayez fini d'utiliser le programme.

Voici deux exemples d'application pour GETKEYX. La première, "VREG", visualise un registre choisi aussi longtemps que la touche correspondante est maintenue baissée. Lorsque vous exécutez "VREG", la demande "REG?" apparaît, demandant le registre choisi. La touche "A" choisit le registre 01, la touche "B" choisit le registre 02, et la touche "Z" choisit le registre 26. Pressez la touche ON deux fois pour quitter le programme.

Voici une brève explication. La première boucle GETKEYX détecte le moment où la touche est pressée. Si le code de touche est 01 (la touche ON), un RTN arrête le programme (le relâchement de la touche ON éteint la calculatrice, et la seconde pression ranime la calculatrice).

Si le code de touche n'est pas 01, le code ASCII est récupéré de LASTX et ajusté en soustrayant 64. Ceci convertit "A" en 01 et "Z" en 26. Les deux lignes suivantes remplacent les valeurs négatives par zéro. Puis le registre sélectionné est visualisé. La seconde boucle GETKEYX continue simplement de boucler aussi longtemps qu'une touche est maintenue baissée. Lorsque la touche est relâchée, le code de touche zéro provoque un branchement vers LBL 01 en haut du programme, où la question REG? est reposée.

Une autre application plus évidente de GETKEYX est de permettre la sélection d'une touche pour un assignement demandé par un programme. Par exemple, supposez que vous ayiez un programme de conversion de base qui réalise les assignements des fonctions MOD, INT, et FRC. Dans la section 4F vous avez appris la manière dont la fonction PASN peut être utilisée pour faire des assignements sous le contrôle du programme. La fonction GETKEYX permet à l'utilisateur du programme de choisir des touches pour ces assignements "en temps réel", quand le programme est en train de fonctionner, sans avoir à représenter le code de la touche. Un programme typique pourrait utiliser une structure de la sorte :

```
"MOD"          Le sous-programme LBL 99 demande
```

```
XEQ 99          à l'utilisateur de presser une touche, utilise
"INT"           GETKEYX pour obtenir le code de touche, puis
XEQ 99          utilise PASN pour faire l'assigne-
"FRC"           ment demandé.
XEQ 99
.
.
.
LBL 99
"├ KEY? "       noter les espaces avant et après KEY?.
AVIEW           affiche le message demandant une touche.
10
LBL 00
GETKEYX         obtient le code de touche en Y.
X<>L
X>Y?            si code de touche<10 , essayez à nouveau.
GTO 00
RDN             ces cinq pas suppriment les 6 caractères
6               " KEY? " du registre ALPHA.
CHS
AROT
ASHF
RDN             le code de touche est maintenant en X.
PASN            faites l'assignement de touche demandé.
RTN
```

La fonction ΣREG? sur la HP-41CX donne l'emplacement courant du bloc de registres statistiques, un bloc de 6 registres utilisé par la calculatrice pour les opérations statistiques Σ+, Σ-, MEAN et SDEV. La fonction ΣREG du catalogue 3 choisit un registre de départ pour ce bloc de 6 registres. Lorsque vous exécutez ΣREG?, le nombre renvoyé est le même que le dernier emplacement choisi par ΣREG, ou 11 si vous n'avez pas exécuté ΣREG depuis la dernière fois que la calculatrice a été effacée. la pile est levée est LASTX n'est pas perturbé. Les fonctions des 6 registres du bloc de registre de statistique sont les suivantes :

$R_{\Sigma REG?}$	Σx
$R_{\Sigma REG?+1}$	Σx^2
$R_{\Sigma REG?+2}$	Σy
$R_{\Sigma REG?+3}$	Σy^2
$R_{\Sigma REG?+4}$	Σxy
$R_{\Sigma REG?+5}$	n

L'application primaire de la fonction ΣREG? est de rappeler les données à partir des registres statistiques sans tenir compte de l'endroit où ces registres sont placés. Par exemple, la séquence :

```
2
ΣREG?
+               ΣREG? en L
RDN             ΣREG? + 2 en T
RCL IND T       Σy
RCL IND L       ΣX
```

simule la fonction RCLΣde la HP-67/97, apportant la somme des valeurs de y dans le registre Y et la somme des valeurs de x dans le registre X. Vous pouvez modifier cette séquence pour rappeler n'importe lequel des 6 registres du bloc REG pour vos calculs.

Les six autres fonctions de la HP-41CX, X<NN?, X<=NN?, X=NN?, X≠NN?, X>=NN?, X>NN? vous permettent de comparer les contenus de X avec tout autre registre. L'emplacement de l'autre registre est désigné en Y. Si vous êtes familier avec les fonctions indirectes, ces fonctions sont effectivement des fonctions "X comparé indirect Y". Pour utiliser une de ces six fonctions, par exemple X>=NN?, placez simplement un numéro de registre en Y (de 0 à SIZE?-1) et pressez : XEQ ALPHA X Shift J = N N ? ALPHA. Au lieu d'un numéro de registre, Y peut contenir des données alpha désignant un registre de pile : "Z", "T", ou "L". "X" et "Y" fonctionneront mais ne seront pas utiles. Le résultat sera affiché : YES si les contenus de X sont plus grands ou égaux aux contenus du registre spécifié en Y, NO dans le cas contraire.

Si vous utilisez une de ces instructions dans un programme, l'affichage YES ou NO n'apparaîtra pas. A la place, l'instruction qui suit sera exécutée seulement si le résultat est YES, autrement elle sera sautée. Cette opération est conforme à la règle standard "exécutez si vrai" pour toutes les instructions de test.

Un trait important distingue ces six fonctions de comparaison de leur contreparties du catalogue 3. Ces fonctions indirectes de comparaison vous permettent de comparer les données alpha aussi bien que les valeurs numériques. Les chaînes sont comparées sur la base des codes ASCII. L'effet est le même que si vous utilisiez ATOX pour comparer les chaînes caractère par caractère de gauche à droite, s'arrêtant à la première position qui révélerait une différence entre les chaînes. L'ordre du code ASCII des chaînes alpha est semblable à l'ordre lexicologique normal sauf que :
1) les caractères numériques et de ponctuation sont inférieurs aux caractères alphabétiques, et
2) les caractères minuscules sont plus grands que les caractères majuscules.

Pour plus de détails sur l'ordre du code ASCII, voyez la table d'équivalence ASCII aux pages 42 et 43.

Le programme "ALSORT" listé ci-contre trie un bloc de registres du registre ddd au registre fff, inclus, dans l'ordre croissant. Il utilise un simple algorithme de tri à bulle. Placez simplement le nombre ddd,fff en X et exécutez "ALSORT".

Parce que la fonction X>=NN? est utilisée pour les comparaisons, le programme "ALSORT" triera les données numériques, les données alpha ou les deux. L'algorithme de tri à bulles est simple. En BASIC il ressemble à quelque chose comme ça :

```
FOR i= ddd+1 TO fff
     FOR J = i-1 TO ddd STEP -1
          IF R(J+1)  R(J), GOTO nouvel i
```

```
                ELSE SWAP (R_J+1,R_J)
                NEXT J
        nouvel i : NEXT i
```

Dans le programme "ALSORT", LBL 01 commence la boucle i, qui utilise un compteur ISG de la forme i,fff. LBL 02 commence la boucle J qui utilise un compteur DSE j,(ddd-1). Si vous voulez tracer l'utilisation de la pile de "ALSORT", il peut être utile de savoir que à LBL 01, les contenus importants de la pile sont :

X=0,(ddd-1) et Y=i,fff

Au LBL 02, la pile contient :

L=j+1, X=R_{j+1}, Y=j,(ddd-1), Z=0,(ddd-1), et T=i,fff.

CHAPITRE CINQ

UN COMPTEUR DE TAILLE DE PROGRAMME

Le mode d'emploi de la HP-41 mentionne que l'octet est l'unité de base de la mémoire programme, et que chaque instruction dans un programme occupe un ou plusieurs octets. En effet, le mode d'emploi donne un tableau résumé du nombre d'octets pour chaque type d'instruction. Si vous possédez une HP-41CX, et que vous voulez savoir la quantité d'octets qu'un de vos programmes occupe, vous pouvez simplement exécuter le catalogue 1 et presser R/S pour l'arrêter au END du programme sélectionné. Le nombre à droite de l'affichage indique le nombre d'octets de la mémoire principale que le programme occupe. Si la dernière instruction du programme est .END., il vous faudra presser GTO.. Pour donner au programme son dernier END. Aucun nombre d'octet n'est fourni avec le .END..

Si vous possédez une HP-41C ou CV et que vous voulez savoir la quantité d'octets qu'un de vos programme occupe, vous pouvez vous reporter au tableau résumé de votre mode d'emploi et compter les octets à la main. En divisant par sept et en arrondissant cela donne le nombre de registres demandés pour contenir le programme. Naturellement cette procédure de comptage manuelle semble une perte de temps lorsque vous possédez un puissant outil tel que la HP-41.

Avec le module de fonctions d'extensions, vous pouvez automatiser la procédure de comptage d'octet. Le court programme utilitaire "CBX" présenté ici fait tout le boulot. D'abord "CBX" sauve votre programme dans la mémoire d'extension, créant un nouveau fichier de programme (à moins que le programme n'ai déjà été sauvé dans la mémoire d'extension). Ensuite "CBX" exécute une instruction RCLPT qui, pour un fichier de programme, retourne le nombre d'octets du programme dans le registre X. Finalement, "CBX" efface le fichier de programme temporaire qu'il a créé. Si votre programme était déjà sauvé dans la mémoire d'extension, le fichier n'est pas effacé. Après que "CBX" ai pris le nombre d'octets, il calcule le nombre de registres de programmes demandés. Ce nombre serait égal à FLSIZE, sauf qu'il y a un octet supplémentaire dans le fichier pour la somme de contrôle du programme (voir page 126). Ainsi quelquefois le nombre de registres de programme demandé est inférieur de 1 à FLSIZE.

Instructions pour "CBX"

1. Assurez-vous que le programme que vous voulez mesurer a un END non permanent (pas le .END.) comme dernière ligne, et que le programme est PACKé. Ce sont ces mêmes choses que vous devriez faire avant de sauver un programme dans la mémoire d'extension, de façon à minimiser l'espace utilisé.

2. Chargez le registre ALPHA avec le nom du programme pour lequel vous voulez un nombre d'octets. Ce nom ne doit pas entrer en conflit avec le nom d'un fichier ASCII ou de données préexistant.

3. Exécutez "CBX" (pressez XEQ ALPHA C B X ALPHA).

4. Si le programme était déjà sauvé dans la mémoire d'extension, le résultat apparaîtra très vite. Le nombre d'octets sera en X, et le nombre de registres de programme sera en Y. Pour voir le nombre de registres de programme, pressez X<>Y ou RDN (rotation vers le bas).

5. Si le programme n'était pas déjà sauvé dans la mémoire d'extension, "CBX" prendra quelques secondes de plus pour obtenir le résultat. D'abord une table de mémoire d'extension apparaîtra. Environ une demi seconde après

que la table soit finie, le nombre d'octets apparaîtra en X, avec le nombre de registres en Y. Pour accélérer les choses, vous pouvez interrompre l'affichage de la table et redémarrer le programme "CBX" en pressant R/S deux fois.

"CBX" Exemple 1 :
Comptez le nombre d'octets dans "CBX" lui-même. Solution :

Chargez le registre ALPHA avec le nom du programme "CBX". Puis XEQ "CBX". Le résultat devrait être un compte de 52 octets en X, et un compte de 8 registres en Y. Si "CBX" n'était pas compacté, ou s'il n'avait pas de END non permanent, votre compte peut être légèrement différent.

Analyse de "CBX" ligne par ligne

Au début de "CBX", le registre ALPHA devrait contenir le nom du programme pour lequel le compte des octets est désiré.

La ligne 03 retournera le compte d'octets si le programme est déjà sauvé dans la mémoire d'extension. Si le programme n'est pas déjà sauvé dans la mémoire d'extension, l'instruction RCLPTA entraînera l'effacement du drapeau 25. La ligne 04 efface le drapeau 25 et se branche sur LBL 01, la séquence finale de calcul, si RCLPTA avait réussi. Autrement le programme désigné est sauvé dans un fichier temporaire en mémoire d'extension temporaire appelé "**CBX". La ligne 08, RCLPT, donne le compte d'octets à partir de ce fichier temporaire. Le fichier temporaire est alors purgé.

L'instruction EMDIR est incluse pour rétablir un fichier de travail après l'instruction PURFL. Si un fichier de travail n'est pas défini et que vous avez un module de fonctions d'extension révision 1B, la mémoire d'extension est en danger d'être effacée. Voir page 19 pour plus de détails. Si votre révision est 1C ou au dessus (y compris la HP-41CX), vous pouvez effacer en toute sécurité les lignes de 11 à 14. Les instructions ENTER↑ et RDN vous assurent que, table interrompue ou non, le registre X contiendra le compte d'octets. Une instruction EMDIR terminée lève la pile, donnant le nombre de registres libres dans la mémoire d'extension. Une instruction EMDIR interrompue ne lève pas la pile. Des deux façons, l'instruction RDN laissera le compte d'octets dans le registre X, puisque le compte d'octets était en X et Y avant l'instruction EMDIR. L'instruction CLD est inclue de telle sorte que la dernière entrée de la table ne reste pas dans l'affichage après la fin de "CBX".

La séquence LBL 01 commence avec le compte d'octets en X et calcule le nombre de registres de programme demandés.

$$N_{reg}=INT((N_{octets}+6)/7)$$

Cette formule accomplit la division par 7 et l'arrondi au nombre entier immédiatement supérieur. Le programme "CBX" termine avec N_{reg} en Y et N_{octets} en X.

"CBX" Exemple 2 :

Comptez les octets dans le programme "JNX" de la section 1B. Cet exemple considère que vous avez une copie de "JNX" ou bien dans la mémoire principale ou bien dans la mémoire d'extension.

Solution :

Chargez le registre ALPHA avec "JNX" et pressez XEQ "CBX". Le résultat devrait être 80 octets (12 registres).

Liste du programme « CBX »

```
01♦LBL "CBX"      07 SAVEP          13 CLD          18 +
 02 SF 25         08 RCLPT          14 RDN          19 7
 03 RCLPTA        09 "**CBX"                        20 /
 04 FS?C 25       10 PURFL          15♦LBL 01       21 INT
 05 GTO 01        11 ENTER↑         16 RCL X        22 X<>Y
06 "⊢,**CBX"      12 EMDIR          17 6            23 END      52 BYTES
```

Liste des programmes « SOLVE »/« DERIV »/« INTEG »

```
01♦LBL "SOLVE"
  02 ASTO 00
  03 STO 01
  04 1
  05 %
  06 +
  07 STO 02
 08 "**SOLVE"
  09 4
  10 XEQ 05
  11 2 E-3
  12 SAVERX
  13 RCL 01
14 XEQ IND 00
 15 "**SOLVE"
  16 SAVEX
  17 GETR

 18♦LBL 01
  19 CLX
  20 SEEKPTA
  21 3 E-3
  22 SAVERX
  23 RCL 02
  24 FS? 10
  25 VIEW X
26 XEQ IND 00
 27 "**SOLVE"
  28 GETR
  29 ENTER↑
  30 ENTER↑
  31 X<> 03
  32 -
  33 X≠0?
  34 /
  35 RCL 02
  36 ENTER↑
  37 X<> 01
  38 -
  39 *
  40 ST- 02
  41 RCL 01
  42 RND
  43 RCL 02
  44 RND
  45 X≠Y?
  46 GTO 01
  47 LASTX
  48 GTO 21

49♦LBL "DERIV"
  50 ASTO 03
  51 STO 04
  52 RDN
  53 STO 05
  54 3 E-3
  55 STO 06
 56 "**DERIV"
  57 7
  58 XEQ 05

  59♦LBL 02
  60 CLX
  61 SEEKPTA
  62 6 E-3
  63 SAVERX
  64 RCL 06
  65 INT
  66 RCL 05
  67 *
  68 RCL 04
  69 +
 70 XEQ IND 03
 71 "**DERIV"
  72 GETR
73 STO IND 06
  74 ISG 06
  75 GTO 02
  76 RCL 03
  77 RCL 02
  78 RCL 01
  79 ENTER↑
  80 +
  81 -
  82 9
  83 *
  84 -
  85 +
  86 RCL 00
  87 11
  88 *
  89 -
  90 6
  91 /
  92 RCL 05
  93 /

 94♦LBL 21
  95 ENTER↑
  96 PURFL
  97 EMDIR
  98 CLD
  99 RDN
 100 RTN

101♦LBL "INTEG"
 102 ASTO 00
 103 STO 01
 104 X<>Y
 105 -
 106 4
 107 /
 108 STO 02
 109 ST+ X
 110 ST- 01
 111 CLX
 112 STO 03
 113 STO 06
 114 STO 07
 115 SF 20
 116 20
117 "**INTEG"
 118 XEQ 05

 119♦LBL 22
 120 2
 121 RCL 03
 122 CHS
 123 Y↑X
 124 STO 05
 125 ST+ 05
 126 1
 127 -

 128♦LBL 03
 129 STO 04
130 "**INTEG"
 131 CLX
 132 SEEKPTA
 133 .019
 134 SAVERX
 135 FS? 49
 136 OFF
 137 3
 138 RCL 04
 139 X↑2
 140 -
 141 RCL 04
 142 *
 143 RCL 02
 144 *
 145 RCL 01
 146 +
147 XEQ IND 00
 148 "**INTEG"
 149 GETR
 150 1
 151 RCL 04
 152 X↑2
 153 -
 154 *
 155 ST+ 06
 156 1
 157 RCL 04
 158 RCL 05
 159 +
 160 X<Y?
 161 GTO 03
 162 RCL 03
 163 STO 04
 164 RDN
 165 7
 166 STO 05
 167 SIGN
 168 ST+ 03
 169 -
 170 RCL 02
 171 *
 172 RCL 06
 173 *
 174 3
 175 *

176♦LBL 04
 177 ENTER↑
178 X<> IND 05
 179 ST- Y
 180 RND
 181 X<> Z
 182 4
 183 *
 184 STO Z
 185 DSE X
 186 /
187 RCL IND 05
 188 +
 189 ISG 05
 190 LN
 191 DSE 04
 192 GTO 04
193 STO IND 05
 194 FS? 10
 195 VIEW X
 196 RND
 197 R↑
 198 FC?C 20
 199 X≠Y?
 200 GTO 22
 201 LASTX
 202 GTO 21

 203♦LBL 05
 204 SF 25
 205 CRFLD
 206 FS?C 25
 207 RTN
 208 SF 25
 209 PURFL
 210 FC?C 25
 211 GTO 06
 212 SF 25
 213 CRFLD
 214 FS?C 25
 215 RTN

 216♦LBL 06
217 "NO ROOM - EM"
 218 PROMPT
 219 END

      405 BYTES
```

CHAPITRE 6

APPLICATIONS DES FICHIERS DE DONNEES

6A. Un programme universel de recherche de racines

Une application fréquente des calculatrices programmables est de résoudre des équations de la forme f(x)=0 ; c'est à dire de trouver la valeur de x qui rend cette équation vraie pour une fonction fournie par l'utilisateur. Par exemple, supposez que le coût de la production de n articles utilisant la machine 1 est SQRT (n) tandis que le coût de la production de n articles sur la machine 2 est 10+LN(n+1). Parce que la fonction LN est "plus plate" que la fonction SQRT, la Machine 2 sera plus économique pour un très grand nombre de valeurs de N. Mais à quelle valeur de n la machine 2 devient-elle plus économique ? Pour trouver le point d'intersection, il nous faut résoudre l'équation SQRT(n)=10+LN(n+1) pour n. Cette équation peut être reécrite sous la forme f(x)=0, où f(x)=10+LN(x+1)-SQRT(x). Cet exemple sera résolu plus tard dans cette section.

Les problèmes de minima et de maxima peuvent être résolus sous la forme f(x)=0 en utilisant la dérivée première de f. Si la fonction en étant maximale ou minimale a une forme relativement simple, il est plus rapide d'utiliser le calcul pour trouver la dérivée première correcte. Cependant, si trouver la dérivée analytiquement n'est pas pratique, le programme "DERIV" dans la prochaine section peut la calculer numériquement.

Tout programme qui résoud f(x)=0 devra appeler le programme f(x) plusieurs fois. Le programme de recherche de racine aura besoin également de quelques registres de données pour son propre usage. Ces registres doivent être ceux qui ne sont pas perturbés par l'évaluation de f(x), de façon que l'information nécessaire à partir des évaluations antérieures de f(x) puissent être retenue.

Si le programme f(x) utilise des registres de données, la possibilité d'un conflit dans l'usage des registres ne peut pas être négligée. Peu importe quels registres de données le programme de recherche de racine utilisera, il pourra y avoir un programme f(x) qui utilisera ces mêmes registres de données.

Une "solution " à ce problème est de vérifier dans les programmes de recherche de racine et les programmes f(x) si il y a un conflit entre les registres, et réécrire un des programmes pour éliminer le conflit.

Le module de fonctions d'extension résoud les conflits dans l'usage de registres une fois pour toute. Il vous permet d'écrire un programme universel de recherche de racine qui fonctionnera avec tout programme f(x) (naturellement le programme f(x) doit avoir un label global de 6 caractères ou moins, de façon qu'il puisse être atteint par une instruction XEQ IND). Plutôt que de laisser ces données essentielles dans les registres numérotés, où elle seraient susceptible d'altération par le programme f(x), le programme de recherche de racine sauve ces données dans un fichier de mémoire d'extension avant d'appeler f(x). Après que f(x) ait retourné une valeur, le programme de recherche de racines peut appeler ses données essentielles, intactes, de la mémoire d'extension. C'est un exemple classique de la puissance de la mémoire d'extension.

Le listage du programme ci-contre inclut "SOLVE" plus deux autres programmes qui seront étudiés dans les deux prochaines sections. Ces trois programmes sont combinés en un seul parce qu'ils utilisent des séquences d'instruction communes, et parce qu'ils seront souvent utilisés ensemble. Tapez le programme tel quel de façon que vous puissiez utiliser les

exemples qui suivent.

Exigences en mémoire :

Programme	registres libres nécessaires pour fonctionner
"SOLVE"	4
"DERIV"	7
"INTEG"	20

"SOLVE" Exemple 1 :

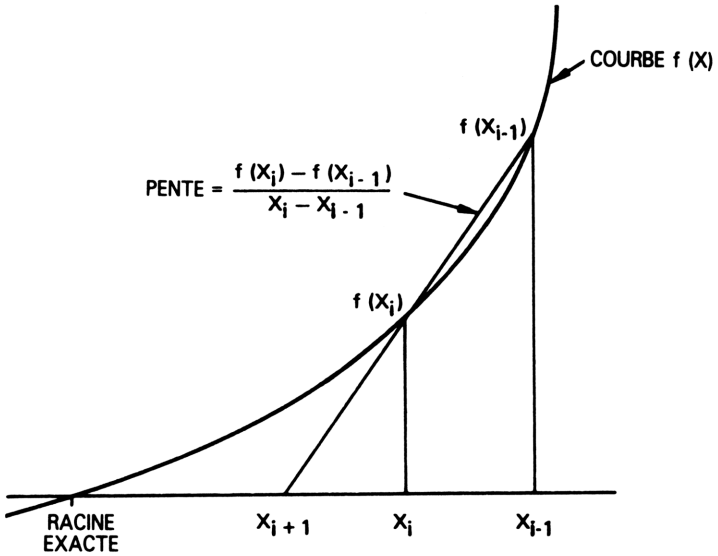
En continuant l'exemple donné au début de cette section, nous voulons trouver la valeur de n telle que SQRT(n)=10+LN(n+1). En dessous de cette valeur, la machine 1 sera plus économique, tandis qu'au dessus de cette valeur, la machine 2 sera meilleur marché à l'usage.

Le premier pas est d'écrire un programme pour calculer f(x). Dans ce cas x est le nombre d'unités à produire, et f est la différence de prix entre la machine 2 et la machine 1. Le programme suivant calcule la différence de prix :

```
01 LBL "CDIFF"      Commence avec n dans le registre X
02 SQRT             SQRT (n)
03 LASTX
04 1
05 +                n+1
06 LN               LN(n+1)
07 X<>Y
08 -                LN(n+1)-SQRT(n)
09 10
10 +                10+LN(n+1)-SQRT(n)
11 END
```

Maintenant que vous avez le programme "CDIFF" et le programme "SOLVE" prêts, obtenir la solution est simple :

1. Vous assurer que la SIZE est au moins 004.
2. Placer le nom de fonction dans le registre ALPHA (dans ce cas pressez ALPHA C D I F F ALPHA)
3. Rentrer une proposition de racine (dans cet exemple vous pouvez utiliser 100, puisqu'il n'y a qu'une seule racine, toute valeur positive devrait aller)
4. Choisir un mode d'affichage correspondant à la précision que vous voulez. Par exemple, si vous voulez quatre chiffres significatifs, tapez SCI 3. Si il vous faut une précision de 0,0001 (qui donne un nombre différent de chiffres significatifs selon la valeur de la racine), tapez FIX 4. Le programme de recherche de racine s'arrête quand deux approximations successives sont égales, dans la précision d'affichage spécifiée. N'utilisez pas FIX 9, ENG 9, ou SCI 9, parce que des erreurs d'arrondi peuvent nuire à la précision de la formule utilisée quand vous demandez trop de chiffres. Pour cet exemple, FIX 2 est suffisant.
5. Levez le drapeau 10 si vous voulez visualiser des approximations successives de la racine, baissez le drapeau 10 si vous voulez que le "canard volant" s'affiche.
6. XEQ "SOLVE" pour commencer la recherche de la racine.
7. La recherche de la racine se termine par une table de mémoire d'extension. Vous pouvez interrompre cette table pour voir la réponse, mais il n'est pas nécessaire de le faire. L'instruction EMDIR a été placée à la fin de "SOLVE" pour compenser le parasite PURFL. Si vos fonctions


COURBE f (X)
f (Xi-1)
PENTE = f (Xi) – f (Xi - 1) / Xi – Xi - 1
f (Xi)
RACINE EXACTE
Xi + 1
Xi
Xi-1

nécessaire pour traiter de telles fonctions n'a pas été comprise dans le programme "SOLVE". Vous devez également être conscient que comme x_i et x_{i-1} se rapprochent beaucoup l'un de l'autre (moins de 10^{-9} x_i), la calculatrice ne peut pas calculer précisément leur différence. La multiplication par $x_i - x_{i-1}$ peut de ce fait causer une erreur substantielle dans la valeur calculée de x_{i+1}. C'est pourquoi vous ne devez pas demander une précision de 10 chiffres à "SOLVE".

Analyse ligne par ligne de "SOLVE"

Les 7 premières lignes de "SOLVE" stockent le nom de fonction et la estimation initiale et calculent la seconde estimation comme 1.01 fois la première estimation. Vous pouvez changer cela pour permettre à l'utilisateur de rentrer la seconde proposition, si vous le voulez. L'usage des registres par "SOLVE" est :

Registre	Contenus
00	nom de fonction
01	approximation antérieure x_{i-1}
02	approximation courante x_i
03	$f(x_{i-1})$

Le sous-programme LBL 05 met en place un fichier de données de quatre registres appelé "**SOLVE". Ceci demande plus qu'une simple instruction CRFLD, parce que le programme doit être capable de manipuler automatiquement le cas dans lequel un fichier du nom de "SOLVE" existe déjà dans la mémoire d'extension. Ce cas peut se produire lorsque le programme "SOLVE" se termine anormalement, avant que l'instruction PURFL à la ligne 209 puisse être exécutée.

Les lignes 11 et 12 sauvent les contenus des registres 00 à 02 dans la mémoire d'extension en préparation de l'appel du programme f(x) (qui peut altérer le contenu de ces registres). Ensuite f(x) est évalué d'après l'estimation initiale x_0. Les lignes 15-17 sauvent la valeur de $f(x_0)$ dans le quatrième registre du fichier de données "**SOLVE" et utilisent GETR pour ramener toute les données. A ce point les quatre registres sont initialisés, et la procédure itérative peut commencer.

Les lignes 19-22 sauvent les contenus des registres 00 à 03 dans la préparation de l'exécution de f(x). Passer par la boucle LBL 01 n'est pas nécessaire la première fois, mais le sera dans les répétitions suivantes. Si le drapeau 10 est levé, x_i est visualisé avant le rappel de f(x). Après l'évaluation de f(x), GETR ramène les contenus des registres de 00 à 03.

Les lignes 29 à 40 mettent à jour les contenus de ces registres comme indiqués:

Registre	anciens contenus nom de la fonction	nouveaux contenus nom de la fonction
00		
01	x_{i-1}	x_i
02	x_i	$x_{i+1}=x_i+\frac{(x_i-x_{i-1})*f(x_i)}{(f(x_i)-f(x_{i-1}))}$
03	$f(x_{i-1})$	$f(x_i)$

Ensuite les contenus des registres 01 et 02 sont extraits, arrondis et comparés. Si les versions arrondies sont égales, l'exécution s'arrête avec le programme de nettoyage LBL 21. Autrement la boucle LBL 01 est répétée.

La séquence LBL 21 purge d'abord le fichier à partir de la mémoire d'extension. Ceci entraîne la mise en place d'une situation dangereuse si

vous avez un module de fonctions d'extension mémoire révision 1B, due au parasite PURFL. Après l'exécution de PURFL, il n'y a pas de fichier de "travail". Cela pourrait sembler ne pas être un problème, mais si vous exécutez accidentellement une instruction comme SEEKPT ou SAVERX qui opére sur le fichier de travail, le désastre se produira : toute votre table de mémoire d'extension disparaîtra ! La section 10E donne plus de détails et souligne la façon dont cette condition DIR EMPTY peut être corrigée à l'aide des techniques de programmation synthétique. Ce problème ne se produit pas avec les révisions 1C et plus haut, y compris la HP-41CX.

Pour éviter la catastrophe, la séquence 21 rétablit un fichier de travail de la seule façon possible sans intervention manuelle, avec une instruction EMDIR. Cependant, exécuter EMDIR dans un programme comporte deux effets indésirables. D'abord, l'affichage de la table prend un temps appréciable. Deuxièmement, lorsque la table est achevée, le nombre de registres libres dans la mémoire d'extension est placé en X. Puisque nous voulons que le résultat "SOLVE" soit en X, l'instruction EMDIR est précédée de ENTER et suivie de RDN. De cette façon, même si vous choisissez d'interrompre la table et de redémarrer le programme (comme dans le cas où "SOLVE" serait utilisé comme une sous routine d'un autre programme) le registre X contiendra toujours le résultat correct.

Si vous avez un module de fonctions d'extension de révision 1C ou plus haut, y compris la CX, vous pouvez effacer les lignes 95, 97, 98, et 99. Cela épargnera une quantité importante de temps d'exécution.

Ces détails sur la séquence LBL 21 peuvent ne pas être d'un intérêt immédiat pour vous, mais ils vous sont fournis comme référence. Si vous écrivez un programme qui utilise un fichier de données temporaire, et que vous voulez qu'il soit utillisable avec le module de fonctions d'extension révision 1B, vous aurez à traiter la même situation.

"SOLVE" exemple 2:

Trouvez le second zéro de $J_3(x)$, la fonction de Bessel de première espèce, d'ordre 3. C'est la seconde valeur de x différente de zéro pour laquelle $J_3(x)=0$. Naturellement vous aurez besoin d'une copie du programme "JNX" de la fonction de Bessel de la page 10. Puisque vous voulez calculer $J_3(X)$, vous pouvez construire un programme enveloppe :

```
01 LBL "J3X"
02 3
03 X<>Y
04 XEQ "JNX"
05 END
```

Ce programme prend simplement la valeur de X qui est donnée, et appelle "JNX" avec Y=3. Cette technique simple (et même banale) est souvent utilisée avec les programme tels que "SOLVE". Si vous avez une fonction qui a besoin de plus d'une entrée, vous devez soit modifier ce programme soit créer un programme enveloppe pour l'utiliser avec "SOLVE". Le nom du programme enveloppe ne doit pas dépasser six caractères, de façon que l'instruction XEQ IND dans "SOLVE" fonctionne correctement. En fait, c'est une bonne pratique de la programation de la HP-41 d'éviter l'utilisation des labels alpha de 7 lettres partout où vos vos programmes pourraient nécéssiter l'appel indirect de sous-programmes.

Maintenant continuons avec cet exemple. D'abord nous avons besoin de savoir quelque chose en ce qui concerne le comportement de $J_3(X)$. Assignez

"J3X" à une touche (shift ASN ALPHA J shift 3 X ALPHA) et essayez quelques valeurs de X pour avoir une idée générale de ce à quoi ressemble $J_3(x)$. Mais faites attention que le programme "JNX" donne DATA ERROR lorsque X=0.

x	J3(X)
0,01	2,1E-8
1	0,02
2	0,13
3	0,31
4	0,43
5	0,36
6	0,11
7	-0,17
8	-0,29
9	-0,18
10	0,06

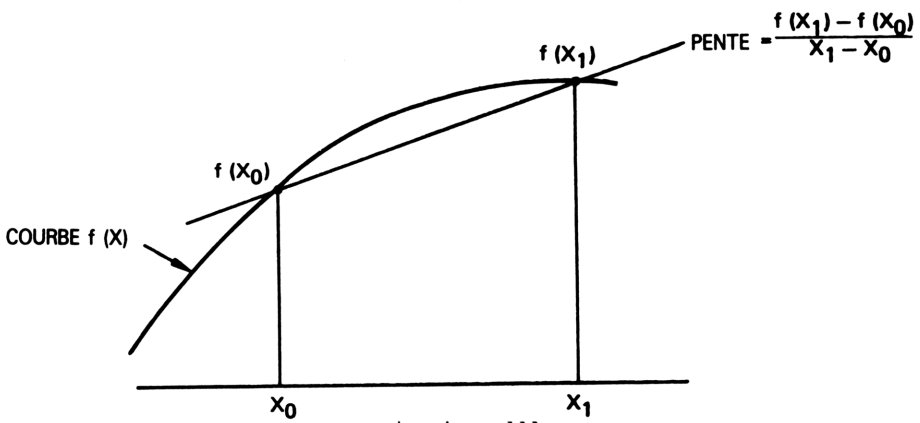
A partir de ces points, il est apparent que le second zéro de $J_3(x)$ est situé entre X=9 et X=10.

Pour trouver la valeur plus exactement, placez un SIZE de 008 ou plus, choisissez un mode d'affichage FIX 8, chargez le registre ALPHA avec "J3X", entrez une estimation initiale de 9,5, SF 10, et pressez XEQ "SOLVE". Vous verrez les approximations suivantes:

9.59500000
9.76155455
9.76102548
9.76102313

Pour d'autres informations sur la recherche de racine, y compris la discussion d'algorithmes plus sophistiqués, consultez n'importe quel bon livre sur l'analyse numérique ou lisez la description du programme "SV" dans le manuel du PPC ROM.

6B. Différentiation numérique



dessin p 111

De nombreux types de problèmes, particulièrement ceux impliquant des calculs de maxima et minima, demandent une évaluation numérique de la dérivée d'une fonction. La technique préférée est d'abord d'utiliser les règles du calculs pour construire une équation pour la dérivée, puis d'écrire un programme pour évaluer l'équation. Si la fonction n'a pas une forme simple, les méthodes numériques peuvent être utilisées. La plus simple de ces méthodes est d'évaluer la fonction en deux points rapprochés et de calculer une tangente basée sur ces deux valeurs.

Une estimation beaucoup plus précise de la dérivée première est donnée par l'expression :

$$f'(x)=(2f(x+3h)-9f(x+2h)+18f(x+h)-11f(x))/6h.$$

Cette estimation est exacte pour toute fonction polynomiale de degré trois ou moins. Autrement l'erreur est de l'ordre de h^3. Cependant, il ne s'en suit pas automatiquement que vous devriez utiliser la plus petite valeur possible pour h. Le problème est qu'une erreur d'arrondi peut détruire la précision qui serait obtenue autrement en diminuant la valeur de h.

Un exemple typique illustrant l'effet de ces erreur d'arrondi sera donné dans l'exemple 2 de cette section.

A cause de l'erreur d'arrondi dans la soustraction, la plus grande précision à laquelle on peut s'attendre dans l'estimation de la dérivée est de 6 chiffres. Toute autre précision est pure coïncidence.

Le programme "DERIV" qui fait partie du groupe "SOLVE"/ "DERIV"/ "INTEG" évalue l'équation ci-dessus pour f'(x) en appelant le programme f(x) fourni par l'utilisateur quatre fois. Comme le programme "SOLVE", le programme "DERIV" protège ces données du programme f(x) en créant un fichier temporaire de données en mémoire d'extension. "DERIV" est donc compatible avec n'importe quel programme f(x) fourni par l'utilisateur.

Instruction pour "DERIV"

Utiliser "DERIV" revient au même qu'utiliser "SOLVE". La SIZE doit être 007 ou plus. Le nom de la fonction fournie par l'utilisateur doit être dans le registre ALPHA. Le registre Y doit contenir le pas h, qui peut être positif ou négatif. Ceci permet au dérivées d'être évaluées là où la fonction est discontinue d'un côté. Utilisez h>0 pour évaluer la dérivée à droite, ou h<0 pour évaluer la dérivée à gauche. Le registre X doit contenir la valeur de x à laquelle f(x) doit être estimée.

L'exécution de "DERIV" produit alors l'estimation de la dérivée. La précision de l'estimation dépend de la taille du pas (voir exemple 2), mais en aucun cas on ne peut s'attendre à une précision de plus de 6 chiffres. Le format d'affichage n'affecte pas la précision, puisqu'aucun arrondi n'est réalisé.

"DERIV" Exemple 1:

Vérifiez la propriétés suivantes des dérivées des fonctions de Bessel :

$J_0'(x)=-J_1(x)$, et
$J_1'(x)=J_0(x)-J_1(x)/x$.

D'abord il vous faudra construire des fonctions enveloppe pour $J_0(x)$ et $J_1(x)$:

```
01 LBL "J0X"          01 LBL "J1X"
02 0                  02 1
03 X<>Y               03 X<>Y
04 XEQ "JNX"          04 XEQ "JNX"
05 END                05 END
```

Pour estimer la dérivée de $J_0(x)$ à x=1, pressez : 01 ENTER↑ 1 ALPHA J shift O X ALPHA XEQ ALPHA D E R I V ALPHA. La dimension du pas de ,01 donne une précision approximative de 6 chiffres. Des pas de 0,1 ou 0,001 donnent une précision de 4 chiffres, qui est aussi tout à fait raisonnable. Comparez vos résultats au tableau suivant :

	Dérivées estimées		Dérivées exactes	
x	$J_0'(x)$	$J_1'(x)$	$-J_1(x)$	$J_0(x)-J_1(x)/x$
1	-0,440050350	0,325147317	-0,440050586	0,325147101
2	-0,576724900	-0,064471700	-0,576724808	-0,064471625
3	-0,339059100	-0,373071483	-0,339058958	-0,373071608
4	0,066043167	-0,380638968	0,066043328	-0,380638978
5	0,327579167	-0,112081133	0,327579138	-0,112080944

Ces résultats ont été obtenus en utilisant le programme "COMPARE", qui automatise toute la procédure. Une fois utilisé avec une imprimante, un programme tel que "COMPARE" peut épargner de nombreuses minutes de frappe au clavier et de recopie des résultats. A la place, vous pouvez simplement mettre en route l'imprimante, démarrer le programme "COMPARE", et revenir quinze minutes plus tard pour vérifier les résultats.

Analyse ligne par ligne de "DERIV"

L'utilisation des registres de données par "DERIV" est :

Registre	contenu
00	f(x)
01	f(x+h)
02	f(x+2h)
03	nom de la fonction -- f(x+3h)
04	x
05	h
06	i compteur de boucle (à l'origine 0,003)

Lorsque "DERIV" démarre, la pile et le contenu de ALPHA sont :

Registre	contenu
Y	h, la taille du pas
X	x, le point auquel estimer f'(x)
ALPHA	nom de la fonction.

Les lignes 49-55 stockent les entrées dans les registres appropriés. Les lignes 56-58 mettent en place le fichier de données de 7 registres

appelé "**DERIV" dans la mémoire d'extension. Les lignes 59-75 constituent la boucle qui est exécutée quatre fois pour évaluer f(x), f(x+h), f(x+2h), et f(x+3h). Le registre 06 contient le compteur ISG de cette boucle. Le compteur est également utilisé comme pointeur indiquant où le résultat doit être stocké (ligne 73).

A l'intérieur de la boucle, les lignes 59-63 sauvent les contenus des registres 00 à 06 dans la mémoire d'extension. Les lignes 64-70 calculent f(x+ih). En fait la fonction INT doit être utilisée pour supprimer le 0.03 du compteur i de la boucle. Les lignes 71-73 rappellent les registres 00-06 de la mémoire d'extension et stockent le résultat f(x+ih) dans le registre de donnée i. Les lignes 74-75 provoquent la répétition de la boucle pour la prochaine valeur de i, jusqu'à ce que la fonction f ait été évaluée aux autres points.

Le dernier pas en calculant l'estimation de la dérivée utilise les quatre résultats des registres 00-03 pour former le résultat.

$$f'(x)=(2f(x+3h)-9f(x+2h)+18f(x+h)-11f(x))/6h$$

La factorisation qui est utilisée dans le calcul est :

$$f'(x)=(f(x+3h)+f(x+3h)-9(f(x+2h)-2f(x+h))-11f(x))/6h.$$

Le programme "DERIV" se termine par la même séquence EMDIR que "SOLVE". La même option d'interruption de la table de la mémoire d'extension s'applique.

Comme pour "SOLVE" vous pouvez effacer l'instruction EMDIR si vous avez une révision 1C ou plus élevée.

"DERIV" Exemple 2:

Utilisez "DERIV" pour calculer la dérivée de la fonction :

$$f(x)=x^4+10x^3+100x^2+1000x+10000$$

à x=1. Montrez comment la précision de l'estimation varie avec la taille du pas. A partir du calcul différentiel, la dérivée de f(x) est :

$$f'(x)=4x^3+30x^2+200x+1000$$
$$= 1234 \text{ à } x=1.$$

Vérifiez vos résultats d'après le tableau suivant :

taille du pas	estimation de dérivée	
1,0	1240,000000	
,1	1234,006000	
,01	1234,000000	(c'est la taille de pas optimale)
,001	1234,016667	
,0001	1233,833333	
,00001	1233,333333	

Si vous trouvez difficile de construire un programme pour évaluer f(x), étudiez la séquence ci-dessous. Elle utilise la factorisation :

$$f(x) = (((x+10)x+100)x+1000)x+10000.$$

Cette technique de factorisation peut être appliquée à toute fonction polynomiale.

```
01 LBL "FX"
02 ENTER↑
03 ENTER↑
04 ENTER↑
05 10
06 +
07 *
08 1E2          (pressez 1 EEX 2)
09 +
10 *
11 1E3
12 +
13 *
14 1E4
15 +
16 END
```

Trouver la dérivée à x=1 est simple. Pressez simplement ALPHA F X ALPHA, rentrez la taille du pas , ENTER↑, le nombre 1, et XEQ "DERIV". Vos résultats doivent concorder avec le tableau ci-dessus. Vous pouvez même souhaiter automatiser la procédure en écrivant un court programme comme celui-ci :

```
01 LBL "STEP"
02 "FX"
03 1
04 XEQ "DERIV"
05 END
```

Notez que ENTER↑ n'est pas inclus parce que les lignes 01 et 02 permettent la montée de la pile. Consultez le mode d'emploi de votre calculatrice pour des détails sur les mouvements de la pile.

Précision de "DERIV"

Comme l'exemple précédent l'a montré, la précision de l'estimation de la dérivée s'améliore à mesure que la taille du pas diminue, mais ensuite empire si la taille du pas est rendue trop petite. Il est possible d'écrire un programme qui appelle "DERIV" à répétition, diminuant à chaque fois la taille du pas. Les séries d'estimations de dérivée D_i doivent avoir les propriétés suivantes :

1) D_i doit être monotone, et

2) $|D_i - D_{i-1}|$ doit être monotone et décroissante.

Lorsque la taille du pas devient si petite que l'une de ces conditions est violée, la précédente estimation de la dérivée D_{i-1} est la meilleure estimation disponible. Cette approche est utilisée dans le programme "FD" du PPC ROM, où un facteur de 0,7 est utilisé pour diminuer la taille du pas dans chaque itération. Vérifiez page 146 du manuel de l'utilisateur du PPC ROM pour plus de détails. Le principal inconvénient de cette approche est qu'elle ralentit grandement l'évaluation de la dérivée. Dans la plupart des

cas, y compris maximisation et minimisation, la précision supplémentaire n'est pas nécessaire. De plus, si "DERIV" doit être appelé par "SOLVE", chaque estimation de dérivée doit être aussi rapide que possible. Votre application peut même se contenter de l'utilisation de la très simple estimation :

$$f(x)=(f(x+h)-f(x))/h,$$

qui est deux fois plus rapide que "DERIV".

Théorie de "DERIV"

L'expression f'(x)=(2f(x+3h)-9f(x+2h)+18f(x+h)-f(x))/6h est l'une des catégories d'estimation de dérivées. Ces estimations peuvent être déduites par l'intermédiaire d'une décomposition en séries de Taylor de f(x). Par exemple, une estimation à quatre points de la dérivée seconde peut être déduite comme suit :

$$\begin{aligned} f''(x) &= a_0 f(x)+a_1 f(x+h)+a_2 f(x+2h)+a_3 f(x+3h) \\ &= a_0 f(x) + a_1 f(x) + a_2 f(x) + a_3 f(x) \\ &\quad + a_1 h f'(x) + 2a_2 h f'(x) + 3a_3 h f'(x) \\ &\quad + a_1 \frac{h^2}{2} f''(x) + 4a_2 \frac{h^2}{2} f''(x) + 9a_3 \frac{h^2}{2} f''(x) \\ &\quad + a_1 \frac{h^3}{6} f'''(x) + 8a_2 \frac{h^3}{6} f'''(x) + 27a_3 \frac{h^3}{6} f'''(x) \end{aligned}$$

Les dérivées quatrièmes et au-dessus de f(x) sont omises de la décomposition en série de Taylor parce qu'une estimation à quatre points ne permet pas aux dérivées au delà de la troisième d'être considérées. Si l'équation ci-dessus doit être vraie pour toutes les valeurs de h, les équations suivantes doivent être vraies pour les coefficients a0, a1, a2, et a3 :

$$\begin{aligned} a_0 + a_1 + a_2 + a_3 &= 0 \\ + a_1 + 2a_2 + 3a_3 &= 0 \\ + a_1 + 4a_2 + 9a_3 &= 2/h^2 \\ + a_1 + 8a_2 + 27a_3 &= 0 \end{aligned}$$

Ces quatre équations sont suffisantes pour définir les coefficients. Ceci montre aussi pourquoi quatre coefficients ne sont pas suffisants pour considérer les dérivées quatrième et au dessus.

Lorsque ce système d'équation est résolu, le résultat est l'estimation à quatre points :

$$f''(x)=(2f(x)-5f(x+h)+4f(x+2h)-f(x+3h))/h^2.$$

Vous pouvez souhaiter écrire votre propre programme "DERIV2" analogue à "DERIV" et basé sur cette formule.

Un autre exercice intéressant est de dériver l'équation pour une estimation de f'(x) à quatre points et de vérifier que c'est la même que celle utilisée par "DERIV".

"DERIV" Exemple 3:

Trouvez la valeur maximum de $J_1(x)-J_0(x)$ qui se produit juste après la première pointe de $J_1(x)$. Bien que cette fonction soit analytiquement différentiable, cet exemple est destiné à illustrer la manière dont on utilise la combinaison "SOLVE"/DERIV".

L'idée est de résoudre pour la valeur de x qui rend la dérivée de $J_1(x)-J_0(x)$ égale à zéro.

Les programmes enveloppe suivants sont nécessaires pour la solution :

```
01 LBL "J1-J0"
02 0                    ce programme utilise le fait que
03 X<>Y                 J1(x) aboutit en Y
04 XEQ "JNX"            après que J0(x) soit calculé par "JNX".
05 -
06 END

01 LBL "DJ1-J0"
02 "J1-J0"              Ce programme utilise "DERIV" pour calculer
03 ,01                  la première dérivée de J1(x)-J0(x).
04 X<>Y
05 XEQ "DERIV"
06 END
```

Après avoir rentré ces programmes enveloppe, pressez "DJ1-J0", 2, FIX 4, XEQ "SOLVE". Ceci calculera l'emplacement de la pointe de $J_1(x)-J_0(x)$. Pour trouver la valeur de $J_1(x)-J_0(x)$ à la pointe, laissez l'emplacement dans le registre X et XEQ "J1-J0". Votre résultat devrait être :

Emplacement de la pointe	valeur de la pointe de J1(x)-J0(x)
2,9386	0,6002

Une grande précision dans la détermination de l'emplacement de la pointe n'est pas nécessaire pour trouver la valeur de cette pointe. L'aplatissement de la fonction au voisinage de la pointe tolère les erreurs en X.

6C. Programme universel d'integration

L'integration, comme la recherche de racine, est une application fréquente des calculatrices programmables. Le programme d'intégration présenté ici, "INTEG" débute avec une fonction et des valeurs a et b fournies par l'utilisateur. Ensuite "INTEG" calcule :

$$\int_a^b g(z)\,dz$$

De même que le programme de recherche de racine "SOLVE", "INTEG" permet des équations de la forme :

$$\int_a^b g(x,z)\,dz = c$$

qui doivent être résolues pour x. Le procédé d'intégration demande habituellement beaucoup plus d'évaluations de la fonction fournie par l'utilisateur g(z) que ne le ferait la différentiation ou la recherche de racine. Ce procédé rend le programme "INTEG" beaucoup plus lent à donner une réponse que "SOLVE" ou "DERIV". L'algorithme particulier utilisé dans "INTEG" est le même que celui utilisé dans le programme "IG" du PPC ROM, et il est très semblable à l'algorithme utilisé par la fonction "integrate" de la HP-15C.

Instructions pour "INTEG"

Pour calculer l'intégrale de la fonction g(z) de z=a à z=b, placez le nom du programme qui calcule g(z) dans le registre ALPHA, rentrez **a** ENTER↑ **b**, puis XEQ "INTEG".

Un SIZE de 020 ou plus est demandé. La disposition de l'affichage détermine la précision du résultat et la quantité de temps que le calcul prendra, tout comme avec "SOLVE". Le calcul est une procédure itérative qui s'arrête lorsque deux estimations successives sont égales, lorsqu'elles sont arrondies au format de l'affichage courant. Si vous levez le drapeau 10, les estimation successives seront affichées (et imprimées si une imprimante est connectée). Le temps nécessaire pour calculer chaque nouvelle estimation est approximativement le même que le temps total déjà utilisé. C'est à dire que le temps total écoulé double à chaque pas. Aussi assurez-vous de ne pas spécifier plus de précision que vous n'en avez besoin réellement.

"INTEG" Exemple 1 :

Utilisez "INTEG" pour calculer l'intégrale :

$$\int_{-\frac{1}{2}}^{\frac{1}{2}} 3(1-z^2)^{-\frac{1}{2}}\,dz$$

avec six chiffres significatifs.

La première étape est d'écrire un court programme pour calculer la fonction à intégrer :

```
01 LBL "I1"
02 X↑2
03 1
04 X<>Y
05 -
06 SQRT
07 1/X
08 3
09 *
10 END
```

Ensuite placez le mode d'affichage à SCI 5. Comme "INTEG" améliore de façon répétitive ses estimations jusqu'à ce que les deux estimations successives soient égales après arrondi, le mode SCI 5 laissera une précision d'à peu près 6 chiffres. Ensuite, placez le nom de la fonction "I1" dans le registre ALPHA. Levez le drapeau 10 de façon à voir la séquence des estimations. Ensuite rentrez les limites d'intégration : 5 CHS ENTER↑ 5 et XEQ "INTEG". Vous devez voir les estimations suivantes :

3,00000	00
3,14601	00
3,14214	00
3,14158	00
3,14159	00
3,14159	00

Si vous vous impatientez pendant le travail, vous pouvez presser R/S pour interrompre "INTEG", changer le mode d'affichage, et presser R/S pour redémarrer. La réponse correcte pour cette intégrale est PI.

"INTEG" est compatible avec les fonctions qui ne sont pas définies aux limites de l'intervalle d'intégration, parce que il n'essaie jamais d'évaluer la fonction aux limites. L'exemple suivant illustre ceci.

"INTEG" Exemple 2 :
Evaluer :

$$\int_0^1 z^{-\frac{1}{2}}\, dz$$

avec 4 chiffres significatifs.

Solution :

```
01 LBL "I2"
02 SQRT
03 1/x
04 END
```

Pressez SCI 3, "I2", 0 ENTER↑ 1, XEQ "INTEG". Les résultats doivent être :

1,414	
1,710	
1,865	
1,934	
1,967	
1,984	
1,992	
1,996	
1,998	
1,999	
1,999	(la vraie valeur de l'intégrale est 2)

Cet exemple illustre à quel point la convergence peut être lente quand la fonction augmente sans borne à l'une ou l'autre des limites de l'intégration. Le nombre maximum d'itérations que permet "INTEG" est 13, correspondant à 2^{13}-1 évaluations de la fonction. Ce nombre maximum a été choisi par ce qu'il prend plus de 8 heures pour se terminer, même avec la plus simple fonction. Si vous êtes très patient et que vous voulez essayer d'avantage d'itérations, changez simplement les lignes 116 et 133 pour refléter l'usage accru des registres de données. Bien sûr, vous n'avez pas besoin de faire de changement pour "INTEG" si la fonction n'altère pas les registres de données 20 et au dessus.

Théorie de "INTEG"

L'algorithme utilisé par "INTEG" est le même que celui utilisé par le programme "IG" du PPC ROM, et il est très semblable à l'algorithme utilisé par la HP-15C. L'essence de l'algorithme est un partage répété des intervalles. D'abord la fonction g(z) est évalué au point milieu et une estimation de l'intégrale est produite. Puis g(z) est évalué à deux points supplémentaires entre le point milieu et les limites d'intégration pour produire une estimation d'un histogramme à 3 points. Dans la troisième itération, 4 points supplémentaires sont ajoutés chevauchant les 3 points antérieurs. A chaque pas, le nombre de points double approximativement.

Il y a deux améliorations qui sont appliquées à cette procédure d'intégration. Deux estimations successives d'histogramme sont utilisées pour construire une estimation par la règle de Simpson, sans autre évaluation de g(z). Deux estimations successives par la règle de Simpson sont utilisées pour construire une estimation de Newton-Cotes. Cette amélioration des estimations est poursuivie pour un total de k-1 améliorations de la kième itération.

La seconde amélioration à la procédure d'intégration est l'utilisation d'un échantillonnage de points non uniforme pour couvrir l'intervale d'intégration plus rapidement. L'échantillonnage non uniforme est implémenté en réalisant le changement de variables :

$$I = \int_a^b g(z)\,dz$$

$$= 3(b-a)/4 * \int_{-1}^{1} f(u(3-u^2)(b-a)/4+(a+b)/2)*(1-u^2)\,du$$

et en utilisant une procédure d'échantillonnage uniforme pour la nouvelle fonction comme fonction de u. Comparez le partage uniforme d'intervalle pour les u_i à l'espacement non uniforme de z_i pour (a,b)=(-1,1) :

u_i	z_i
0	0
±,5	±,6875
±,25; ±,75	±,3672; ±,9141
±,125; ±,375; ±,625; ±,875	±,1865; ±,5361; ±,8154; ±,9775

Bien que les z_i approchent les limites de l'intégration beaucoup plus rapidement que les u_i, la pénalité est que la simple estimation par histogramme devient plus difficile à calculer. Chaque valeur $g(z_i)$ doit être pondérée par la largeur du sous-intervalle centré sur z_i.

La pleine explication de la façon dont opère "INTEG" prendrait plusieurs pages supplémentaires. Si vous voulez en trouver d'avantage, consultez les références suivantes :

1. Le manuel de l'utilisateur du PPC ROM, pages 222-224 après la description du programme "IG".

2. P.J. Davis et P.Rabinowitz, "Methods of Numerical Integration" section 6.3, presse académique, New York, 1975.

3. W.M Kahan, "Handheld Calculator Evaluates Integrals" (HP-34C), Hewlett-Packard Journal, août 1980.

4. B. Carnahan, H.A Luther, et J.O. Wilkes, "Applied Numerical Methods", section 2.7, J. Wiley, New York, 1969. La notation dans cette référence est Ti,j = M(i+j-1,j-1).

Formules utilisées par "INTEG" :

Nombre d'itérations $k = 0, 1, 2, \ldots$

Pour chaque valeur de k, les valeurs suivantes sont calculées :

Premier point d'échantillonnage $u_0=-1+2^{-k}$

i ème point d'échantillonnage $u_i=u_{i-1}+2^{1-k}$

kième estimatimation d'histogramme M(k,0)

$$S_k=S_{k-1}+\int_0^{2^k-1}(1-u_i^2)*f(u_i(3-u_i^2)(b-a)/4+(a+b)/2)$$

$$M(k,0)=(3(b-a)/4)*2^{-k}*S_k$$

L'amélioration des estimations utilise cette formule pour j=1, 2,... k,

$$M(k,j)=(k,j-1)+(M(k,j-1)-M(k-1,j-1))/4^j-1)$$

Les séries d'estimations M(k,k) convergent vers la vraie valeur de l'intégrale I. En fait, les estimations M(k,0) convergent également vers I, mais la convergence de M(k,k) est plus rapide. La quantité de calcul nécessaire pour construire M(k,k) est très petite par rapport à la quantité de calcul nécessaire pour construire M(0,0), M(1,0), ..., M(k,0) qui sont nécessaires pour obtenir M(k,k).

Analyse ligne par ligne de "INTEG"

Le programme "INTEG" est une version modifiée et optimisée du programme "IG" du PPC ROM, qui a été écrit et revisé par les membres de PPC Read Predmore et John Kennedy, respectivement. Des octets ont été récupérés à quelques endroits, un plus petit nombre de registres de données est utilisé, et, principalement, les capacités de la mémoire d'extension rendent "INTEG" compatible avec toute fonction à intégrer fournie par l'utilisateur.

L'utilisation des registres de données par "INTEG" est :

Registre	contenu	contenu pour la boucle LBL 06
00	nom de la fonction	
01	(a+b)/2	
02	(b-a)/4	
03	k	
04	u_j	k-j compte pour DSE
05	$u_{j+1}-u_j=2^{1-k}$	No de registre pour M(k,j)
06	S_{k-1}	
07	M(0,0)	
08	M(1,0)--M(1,1)	
09	M(2,0)--M(2,1)--M(2,2)	
10	M(3,0)-M(3,1)--M(3,2)--M(3,3)	
etc.		

Lorsque "INTEG" démarre, la pile et les contenus de ALPHA sont :

Registre	contenus
Y	a, la limite inférieure d'intégration
X	b, la limite supérieure d'intégration
ALPHA	nom de la fonction.

Les lignes 101-115 utilisent ces entrées pour initialiser les registres de données. Le drapeau 20 est levé de façon que le test de fin soit contourné pour k=0 (ligne 198). Ensuite, un fichier de données de 20 registres appelé "**INTEG" est créé dans la mémoire d'extension. La sous-routine LBL 05 traite automatiquement le cas dans lequel un fichier nommé "**INTEG" existe déja en mémoire d'extension.

La séquence LBL 22 calcule u_0 et l'increment $u_{j+1}-u_j=2^{1-k}$. C'est dans la boucle LBL 03 que la plupart du temps est dépensé. D'abord les registres sont récupérés dans la préparation du calcul $g(z_j)$. Les lignes 137-146 calculent $z_j=u_j(3-u_j^2)(b-a)/4+(a+b)/2$. Ensuite $g(z_j)$ est évalué et les contenus des registres sont rétablis. La valeur de $g(z_j)$ est pondérée par $(1-u_j^2)$ avant d'être ajoutée à la somme courante S_k. Les lignes 155-160 calculent $u_{j+1}=u_j+2^{1-k}$ et sortent de la boucle LBL 03 lorsque le résultat devient supérieur à 1, et par conséquent en dehors des limites de l'intégration.

Les lignes 135 et 136 sont prévues pour empêcher la perte de mémoire due à une faiblesse de la batterie. Certaines évaluations d'intégrales peuvent durer très longtemps, peut-être même dépasser la durée de vie d'une batterie chargée à bloc. Si une faiblesse de la batterie arrête "INTEG" à la ligne 136, vous pouvez la changer contre une batterie nouvelle et presser R/S pour continuer. Si vous n'avez pas une batterie de rechange, il serait plus sûr de laisser les batteries à l'extérieur pendant quelques heures. La plupart des machines HP-41 conservent le contenu de leur mémoire bien plus de quatre heures après le retrait des batteries. Même les HP-41 les plus anciennes semblent être bonnes pendant au moins 8 heures. Mais n'allumez pas la calculatrice pendant que vous enlevez la batterie. Cela aurait certainement comme résultat un MEMORY LOST.

A la fin de la boucle LBL 03, X est $1+2^{-k}$ et Y est 1. Les lignes 162-175 disposent les registres de données de la boucle LBL 04, ce qui applique la formule M(k,j) aux valeurs précédemment calculées M(k-1,0),

M(k-1,1), ..., M(k-1,k-1), qui sont stockées dans les registres 07 et au dessus. D'abord, k est stocké dans le registre 04 comme compteur de DSE, dont la valeur est k-j. Au premier passage dans la boucle j=zéro. Le nombre 7 est stocké dans le registre 05 comme pointeur du registre contenant M(k-1,0). Ce pointeur sera incrémenté à chaque passage dans la boucle LBL 04, tout comme le compteur dans le registre 04 sera décrémenté. Les lignes 167 et 168 incrémentent k pour le prochain passage à travers la boucle LBL 22. La ligne 169 obtient 2^{-k} en soustrayant 1 de la valeur qui était en X à la fin de la boucle LBL 03. Les lignes 170-175 calculent $M(k,0)=3*2^{-k}*S_k*(b-a)/4$. Notez qu'à ce point Y contient toujours 1. Le registre Y sera utilisé dans la boucle LBL 04 pour contenir la valeur 4^j, qui commence à 1 pour j=0 au premier passage à travers la boucle.

Au début de la boucle LBL 04, X contient M(k,j) et Y contient 4^j. Après les lignes 177-186, X contient $(M(k,j)-M(k-1,j))/(4^j-1)$, Y contient $4*4^j=4^{j+1}$ et Z contient une version arrondie de M(k-1,j). Ensuite, M(k,j) est ajouté à X, produisant la valeur M(k,j+1). Les lignes 189-192 incrémentent le compteur de registre et décrémentent le registre 4. Ceci met en place les conditions pour le prochain passage à travers la boucle LBL 04 avec la prochaine valeur de j. Après k passages à travers la boucle, l'instruction GTO 04 est sautée avec la valeur M(k,k) en X et la version arrondie de M(k-1,k-1) en Z et T. M(k,k) est stockée dans le registre voulu, ensuite elle est arrondie et comparée à M(k-1,k-1). Si les deux versions arrondies sont égales, M(k,k) est extrait de LASTX et retourné comme résultat. Autrement un branchement vers LBL 22 commence le calcul de M(k+1,0) et finalement M(k+1,K+1).

Comme "SOLVE" et "DERIV", "INTEG" se termine par une instruction, EMDIR de façon qu'il y ait un fichier de travail lorsque le programme s'arrête. Ce n'est nécessaire que pour les modules de fonctions d'extension de révision 1B.

"INTEG" Exemple 3 :
Vérifiez que :

$$\int_2^3 J_1(x)\,dx = J_0(2)-J_0(3)$$

Solution :
Faites FIX 4 et rentrez la fonction enveloppe :

```
01 LBL "J1X"
02 1
03 X<>Y
04 XEQ"JNX"
05 END
```

Pressez 2 ENTER↑ 3, "J1X", SF 10, et XEQ "INTEG". Le résultat est :

```
0,4971
0,4831
0,4839
0,4839
```

Pour comparer ceci à la vraie valeur de l'intégrale, pressez : 0

Liste du programme « COMPARE »

```
01♦LBL "COMPARE"
   02 1.005
   03 STO 09

   04♦LBL 01
   05 RCL 09
   06 INT
   07 STO 10
   08 "X="
   09 ARCL X
   10 AVIEW
   11 "J0X"
   12 .01
   13 X<>Y
14 XEQ "DERIV"
   15 "J0E="
   16 ARCL X
   17 AVIEW
   18 .01
   19 RCL 10
   20 "J1X"
21 XEQ "DERIV"
   22 "J1E="
   23 ARCL X
   24 AVIEW
   25 RCL 10
26 XEQ "J1X"
   27 CHS
   28 "J0T="
   29 ARCL X
   30 AVIEW
   31 RCL 10
   32 /
   33 X<> 10
34 XEQ "J0X"
   35 RCL 10
   36 +
   37 "J1T="
   38 ARCL X
   39 AVIEW
   40 ADV
   41 ISG 09
   42 GTO 01
   43 END

      115 BYTES
```

Liste du programme « NAP »

```
01♦LBL "NAP"
 02♦LBL A
 03 XEQ 90
 04 SF 25
 05 5
 06 CHS

 07♦LBL 05
 08 6
 09 +
 10 SEEKPT
 11 FS? 25
 12 GTO 05
 13 LASTX
 14 /
 15 "NAME "
 16 XEQ 92
 17 SIGN
 18 X<>Y
 19 AON
 20 XEQ 94
 21 XEQ 93
 22 XEQ 93
 23 XEQ 93
 24 "PHONE"
 25 XEQ 94
 26 "MISC. "
 27 XEQ 94
 28 AOFF
29 "RGS. = "
 30 7
 31 /
 32 XEQ 92

 33♦LBL 98
 34 RCLFLAG
 35 CF 21
 36 AVIEW
 37 STOFLAG
 38 RDN
 39 RTN
 40 GTO A

 41♦LBL 90
 42 0
 43 GTO 06

 44♦LBL 91
45 "ENTRY NO. ?"
 46 PROMPT
 47 6
 48 *

 49♦LBL 06
 50 "ML"
 51 SEEKPTA
 52 7
 53 RTN

 54♦LBL 92
 55 INT
 56 RCLFLAG
 57 FIX 0
 58 CF 29
 59 ARCL Y
 60 STOFLAG
 61 RDN
 62 RTN

 63♦LBL 93
 64 "ADD. L"
 65 X<>Y
 66 XEQ 92
 67 X<>Y
 68 ISG Y

 69♦LBL 94
 70 "⊦?"
 71 XEQ 98
 72 " "
 73 STOP
 74 APPREC

 75♦LBL 07
 76 1
 77 +
 78 RCLPT
 79 FRC
 80 1 E3
 81 *
 82 +
 83 RTN

 84♦LBL B
 85 XEQ 90
 86 "FIND?"
 87 AON
 88 STOP
 89 AOFF
 90 ASTO 00

 91♦LBL 08
 92 POSFL
 93 X<0?
 94 RTN
 95 6
 96 /
 97 LASTX
 98 X<>Y
99 "ENTRY NO. "
100 XEQ 92
101 XEQ 99
102 *
103 SEEKPT
104 RDN
105 XEQ 95
106 ARCL 00
107 GTO 08

108♦LBL 96
109 SF 25
110 ARCLREC
111 XEQ 99
112 CLA
113 FS? 25
114 GTO 07
115 RTN

116♦LBL C
117 XEQ 91

118♦LBL 95
119 ADV
120 CLA
121 XEQ 96
122 XEQ 96
123 XEQ 96
124 XEQ 96
125 " "
126 XEQ 96
127 " "
128 XEQ 96
129 ADV
130 CLD
131 RTN

132♦LBL D
133 XEQ 91

134♦LBL 09
135 DELREC
136 DSE L
137 GTO 09
138 RTN

139♦LBL c
140 XEQ 90
141 CLA

142♦LBL 10
143 SF 25
144 X<>Y
145 SEEKPT
146 6
147 +
148 X<>Y
149 FS? 25
150 XEQ 95
151 FS? 25
152 GTO 10
153 7
154 /
155 FLSIZE
156 X<>Y
157 XEQ 92
158 "⊦ : "
159 X<>Y
160 XEQ 92
161 "⊦ RGS."

162♦LBL 99
163 RCLFLAG
164 SF 25
165 PRA
166 RCLFLAG
167 FS?C 21
168 FC? 25
169 AVIEW
170 STOFLAG
171 RDN
172 FS? 21
173 FC? 25
174 PSE
175 STOFLAG
176 RDN
177 RTN

178♦LBL E
179 XEQ 91

180♦LBL 71
181♦LBL 84
182 SF 25
183 GETREC
184 FC? 25
185 GTO 01
186 XEQ 99
187 "OK?"
188 XEQ 98
189 GETKEY
190 CLD
191 RDN
192 GTO IND T
193 "LINE?"
194 XEQ 98
195 " "
196 AON
197 STOP
198 AOFF
199 DELREC
200 INSREC
201 GTO 84

202♦LBL 00
203♦LBL 04
204 RCLPT
205 INT
206 SEEKPT
207 GTO 84

208♦LBL 44
209♦LBL 01
210 CLST
211 END

        447 BYTES
```

CHAPITRE SEPT

UN PROGRAMME DE LISTE D'ADRESSES

Le programme "NAP" (Nom/ Adresse/ téléPhone) présenté dans ce chapitre illustre la manière dont les fichiers de texte peuvent être utilisés pour sauver des blocs de données ALPHA d'une manière organisée. Le programme a été écrit par Alan McCornack. Il est inclus ici d'abord comme un outil d'apprentissage, mais vous pouvez le trouver adapté pour un usage général. Son approche relativement évidente peut être un modèle pour vous aider à développer un programme pour organiser votre propre base de données. L'utilisation par Alan de GETKEY dans le programme "Corriger" est aussi instructive, montrant à quel point l'utilisation de GETKEY peut être restreinte à une portion d'un programme si on le désire.

Le programme "NAP" considère que le nom, l'adresse et le numéro de téléphone pour chaque entrée dans la liste a le format suivant :

élément	longueur maximale
Nom	24 caractères
Adresse	
Ligne 1	24 caractères.
Ligne 2	24 caractères
Ligne 3	24 caractères
Numéro de téléphone	22 caractères.
Renseignements divers	22 caractères

La limite de longueur de 24 caractères est imposée afin que chaque élément puisse être pleinement listé sur une seule ligne d'impression. les deux derniers éléments sont limités à 22 caractères de façon à apparaître sur une seule ligne après avoir été décalés de deux espaces. N'importe quel caractère peut être utilisé, incluant des caractères spéciaux accumulés dans le registre ALPHA à l'aide de XTOA.

Le programme "NAP" forme un bloc de 6 enregistrements pour chaque entrée. Les contenus des enregistrements sont comme ci dessous, où **n** est le numéro d'entrée :

numéro d'enregistrement	contenu
6n	nom
6n+1	adresse ligne 1
6n+2	adresse ligne 2
6n+3	adresse ligne 3
6n+4	numéro de téléphone
6n+5	Données diverses

Les entrées sont numérotées en séquence, commençant par zéro. Donc, si vous avez 10 entrées, le fichier de texte contiendra 60 enregistrements. Les entrées seront numérotées de 0 à 9.

Instructions pour l'utilisation du programme "NAP"

1. Avant d'utiliser "NAP" vous devez créer un fichier de texte nommé "ML". Prenez une taille de fichier, rentrez ce nombre en X, placez "ML" dans le registre ALPHA, et exécutez CRFLAS. La taille du fichier doit être assez grande pour contenir la liste d'adresses dont vous avez besoin. Chaque

entrée utilise un seul registre pour chaque groupe de sept caractère de texte plus un autre registre pour les 6 octets de séparateur d'enregistrements. Une entrée typique utilise de 8 à 12 registres. Le nombre maximum d'entrées qui peut être accomodé est par conséquent d'environ 50 à 75 avec un complément de 603 registres dans la mémoire d'extension. Si cette limite est trop basse, vous peuvez être en mesure de faire un peu mieux en abrégeant quelques un des renseignements pour diminuer le nombre de caractère utilisés.

2. Une fois le fichier ASCII "ML" en place, le programme "NAP" fournit plusieurs options. Ces options sont disponibles sur les 5 touches de la rangée supérieure en mode USER. Si vous avez déjà quelques fonctions ou labels globaux assignés à la rangée supérieure, vous pouvez utiliser le programme "SK" de la section 10G pour les désactiver temporairement. Autrement vous devrez effacer les assignements de ces touches en pressant : ASN ALPHA ALPHA (touche) ou en utilisant le programme "CT" de la page 84, puis en effaçant manuellement la touche X↑2. Les assignements de la rangée supérieure doivent être effacés parce que les labels globaux ou les assignements de fonction ont toujours priorité sur les labels locaux là où il y a un conflit.

3. Entrez dans le mode USER. Assurez-vous que la taille est au moins de 001, parce que le programme "NAP" utilise le registre 00. Levez le drapeau 26 manuellement ou bien en éteignant la calculatrice, puis en l'allumant. Effacez le drapeau 21 à moins que vous n'ayez une imprimante connectée et que vous vouliez une sortie imprimée.

4. Pressez GTO "NAP". Vous êtes maintenant dans le programme "NAP". Si vous avez supprimé ou suspendu tout assignement de touche de la rangée supérieure, les touches de la rangée supérieure auront les fonctions suivantes:

Ajouter Trouver Lister Effacer Corriger
Lister tout

La fonction "Lister tout" est obtenue en pressant "racine de x" (la touche "c"). Chacune de ces fonctions sera décrite en détails. Si vous commencez avec un fichier de texte "ML" vide, vous voudrez utiliser "Ajouter" d'abord, aussi pressez la touche "A" (rangée 1, colonne 1). En fait, vous pouvez commencer par la fonction "Ajouter" en utilisant XEQ "NAP" au lieu de GTO "NAP".

(A) Ajouter

Le programme demandera un nom pour remplir la première ligne de l'entrée de six lignes. Le programme s'arrêtera au mode ALPHA, de façon que vous puissiez entrer un nom d'au plus 24 caractères. Vous n'avez pas besoin de compter les caractères, parce que la calculatrice émettra un tone d'avertissement lorsque le 24ième caractère sera rentrée. Après avoir entré le nom, pressez R/S pour redémarrer le programme. La question suivante sera "ADD. L1?", ligne 1 de l'adresse. Rentrez-la et pressez R/S. Procédez de cette façon, en entrant la ligne 2 et la ligne 3 de l'adresse. Si vous voulez laisser une ligne blanche, vous pouvez simplement presser R/S sans entrée en réponse à la demande. Quand la question "PHONE"? apparaît, rentrez le numéro de téléphone. Assurez-vous que vous n'utilisez pas plus de 22 caractères si vous vous attendez à une sortie d'imprimante correcte.

Pour vérifier le nombre de caractères, ajoutez des espaces jusqu'à ce que la tonalité d'alarme résonne. Ceci indique que les 24 caractères sont présents. Ensuite pressez la touche d'effacement pour vous débarasser des espaces supplémentaire que vous avez mis. Pressez R/S lorsque l'entrée est prête à être traitée. La prochaine demande, "MISC.?" sera pour le champ des données diverses. Ce champ doit aussi être limité à 22 caractères si une imprimante est utilisée.

Lorsque vous presserez R/S, le programme stockera cette donnée et terminera. Après avoir fini de faire une entrée avec nom adresse et téléphone, vous pouvez presser R/S ou la touche "A" pour ajouter une autre entrée.

Avertissement: Ne jamais terminer cette partie du programme sans ajouter les 6 lignes de l'entrée. Si vous faites une faute, vous n'avez qu'à continuer à faire les entrées jusqu'à ce que toutes les 6 lignes soient faites, puis corrigez l'erreur à l'aide de la fonction de correction (touche "E") décrite plus loin. Si vous voulez effacer une entrée incomplète, vous pouvez utiliser la fonction d'effacement (touche "D") décrite à la page suivante, mais seulement si aucune autre entrée n'a encore été ajoutée au fichier.

(B) Trouver

Pressez "B", la touche 1/x, pour exécuter ce programme. Le programme vous demandera une chaîne à trouver. Le programme s'arrête en mode ALPHA ainsi vous pouvez rentrer la chaîne. La chaîne peut avoir jusqu'à 24 caractères. Le programme trouvera la chaîne, partout où elle apparaîtra dans le fichier, à condition que la chaîne soit contenue dans une même ligne.

Si on ne trouve pas la chaîne, une valeur de -1 est renvoyée en X. Si on trouve la chaîne, le message "ENTRY NO. n" vous dit quelle entrée contient la chaîne. Les 6 lignes de l'entrée seront alors montrées.

Pressez R/S pour arrêter le programme si c'est l'entrée que vous cherchez. Autrement la recherche continuera automatiquement. Cette seconde recherche (et toutes les recherches qui suivent) n'utiliseront que les 6 premiers caractères de la chaîne que vous avez entrés. Tant que vous n'interromprez pas la recherche en pressant R/S, la recherche continuera à afficher des entrées pour lesquelles une correspondance est trouvée. Lorsque le END OF FL est atteint et qu'aucune autre correspondance n'est trouvée, la valeur -1 revient en X et le programme s'arrête.

La chaîne à trouver n'a pas besoin d'être dans la ligne NAME. On la trouvera sur n'importe quelle ligne n'importe où dans le fichier "ML". Par exemple, supposez que vous vouliez enregistrer des anniversaires sur la ligne MISC. Vous pourriez utilisez la notation "bmm/dd/yy". La minuscule b est un "mot" qui indique les données d'anniversaire. Pour lister toutes les entrées qui comportent des anniversaires de juin, vous pourriez alors chercher la chaîne "b6".

Les caractères minuscules **a** à **e** font des trucs faciles à utiliser parce qu'ils sont faciles à entrer et peu souvent utilisés autrement.

(C) Lister

Pressez "C", la touche $\sqrt{x}$ pour exécuter cette routine. Le programme

demande le numéro de l'entrée à lister. Entrez simplement le numéro et pressez R/S. Rappelez-vous que la première entrée est le nombre zéro, et non 1. Les 6 lignes de l'entrée seront affichées en séquence, et imprimées si l'imprimante est connectée et que le drapeau 21 est levé.

(c) Lister tout

Pressez "c", shift $\sqrt{x}$, pour exécuter cette routine. Aucune entrée n'est nécessaire. Le nom entier et l'adresse seront affichés en séquence. Si l'imprimante est connectée et que le drapeau 21 est levé, la liste sera imprimée. Le numéro de téléphone et les diverses lignes seront indentées de deux espaces dans le listing imprimé. Les entrées successives seront séparées par deux espaces.

A la fin du listing, le nombre de registres utilisés et le nombre total de registres du fichier seront affichés (et imprimés si l'imprimante est connectée et validée).

(D) Effacer

Pressez "D", la touche LOG, pour exécuter cette routine. Le programme demande le numéro de l'entrée qui doit être effacé. Assurez-vous de savoir le bon numéro de l'entrée que vous voulez effacer. Lorsque vous serez sûr de l'entrée que vous voulez effacer, rentrez le numéro et pressez R/S. Rappelez-vous, une fois de plus que la première entrée est le nombre zéro.

(E) Corriger

Pressez "E", la touche LN, pour exécuter cette routine. Le programme vous demandera le numéro de l'entrée que vous voulez corriger. Entrez le numéro et pressez R/S. Le programme continuera à vous montrer chaque ligne de l'entrée (et l'imprimera si l'imprimante est connectée et validé).

Après une pause, la demande "OK?" apparaîtra. Si la ligne est OK et n'a pas besoin d'être changée, pressez "Y" ou R/S. Si vous voulez voir à nouveau la ligne, pressez la touche ALPHA. Pour laisser le mode EDIT, pressez la touche d'effacement ou la touche ON. Pressez "N" (la touche ENTER) si vous voulez changer de ligne. Si vous ne pressez aucune touche, la ligne sera affichée de nouveau et la demande "OK?" sera répétée, tout comme si vous aviez pressé la touche ALPHA. Si vous pressez toute autre touche que celles mentionnées ci -dessus, le résultat est le même que si vous pressiez "N".

Si vous pressez "Y" (la touche de multiplication) ou R/S pour indiquer que la ligne est OK, la ligne suivante sera affichée.

Si vous pressez "N" (la touche ENTER), et tout autre touche que ALPHA, "Y", R/S, ON, ou touche d'effacement, la demande "LINE?" apparaîtra. Rentrez une chaîne pour remplacer la ligne et pressez R/S. Si vous voulez effacer la ligne, vous n'avez qu'à presser R/S. Si vous faites une erreur et que vous ne voulez pas changer la ligne, pressez ALPHA pour sortir du mode ALPHA, puis pressez soit XEQ 84 soit la touche "E" pour redémarrer le processus de correction. La touche "E" redémarrera au début des entrées ; XEQ 84 continue avec la ligne suivante de l'entrée courante.

Chaque ligne de l'entrée est affichée en séquence, avec la demande "OK?". Toutes les fois que vous pouvez presser "Y" ou "N" pour indiquer si oui ou non vous pouvez avancer à la ligne suivante. Lorsque vous aurez fini de corriger toutes les 6 lignes de l'entrée, le programme continuera

automatiquement la prochaine entrée, s'il y en a une.

Lorsque vous voulez terminer, pressez simplement la touche d'effacement en réponse à la demande "OK?"

Analyse ligne par ligne de "NAP"

Le programme "NAP" contient plusieurs sous-programmes qui sont utilisés par plus d'un label local. Cette technique est connue sous le nom de programmation modulaire et réduit considérablement le nombre d'octets utilsés dan une tache complexe en divisant le travail en sections.

Le label 90 choisit un pointeur initial de zéro (début du fichier "ML"). Le label 91 choisit un pointeur au début de l'entrée choisie. La ligne 52 initialise le nombre de caractères dans le registre X. La partie entière de ce nombre de caractères sera affichée plus tard comme nombre de registres occupé.

Le label 92 ajoute la partie entière de X au registre ALPHA pour les besoins de l'affichage. Les fonctions RCLFLAG et STOFLAG garantissent que la disposition de l'affichage est restaurée. Le label 98 affiche le contenu du registre ALPHA sans l'imprimer. Le label 99 imprime le contenu du registre ALPHA, ou affiche ALPHA et fait une pause si l'imprimante est éteinte, non présente, ou invalidée. Cette routine diffère de la routine "PVA" présentée dans la section 4C en ce qu'elle conserve le statut du drapeau 25, qui est utilisé dans "NAP" pour détecter le END OF FL.

Le label 96 sort un enregistrement (ligne) du fichier "ML". la fonction ARCLREC est utilisée au lieu de GETREC de façon que les lignes PHONE et MISC puissent être indentées. Le label 99 est appelé pour la sortie effective. Si on n'a pas rencontré de END OF FL, l'instruction GTO 07 provoque l'exécution du programme de comptage du LBL 07. Cette séquence calcule le nombre de caractères dans l'enregistrement courant (RCLPT, FRC, 1E3, *), et ajoute ce nombre plus 1 (pour la longueur d'enregistrement en octet) au compteur de caractères en X. Les détails de l'utilisation des registres pour le fichier ASCII peuvent être trouvés dans la section 10C.

Le label 95 est utilisé par les labels B, C, et c pour sortir 6 enregistrements consécutifs du fichier.

Il avance le papier, efface ALPHA, puis sort les quatre premières lignes à l'aide du label 96. Les deux lignes suivantes sont précédées de 2 espaces qui sont chargés dans le registre ALPHA avant que le label 96 soit appelé.

Le label A ajoute une entrée au fichier de texte "ML". Les lignes 04-12 trouvent la fin du fichier et y placent le pointeur. Le drapeau 25 est utilisé à la fois pour supprimer le message d'erreur END OF FL et pour tester le moment où le END OF FL est atteint. Les lignes 15 à 28 demandent les 6 lignes de l'entrée et ajoutent les 6 enregistrements au fichier "ML". Le label 93 est un procédé de récupération d'octet qui génère les trois demandes "ADD. L1?", "ADD. L2?" et "ADD. L3?". Les lignes 17-18 disposent un compteur en Y avec une valeur initiale de 1. Le ISG Y à la ligne 68 incrémente ensuite ce compteur toutes les fois que le label 93 est appelé. Le label 94 ajoute simplement un guillemet, appelle le label 98 pour afficher la demande sans l'imprimer, charge le registre ALPHA avec un seul espace, et s'arrête pour les entrées. Si l'utilisateur se contente de presser R/S, le seul espace sera utilisé pour l'enregistrement. Autrement, qu'elle quelle soit, la chaîne ALPHA qui a été tapée sera ajoutée au fichier "ML" de la ligne 74. La séquence LBL 07 met à jour le compteur des caractères en X comme décrits ci-dessus.

Lorsque LBL A conclut, le nombre de registres utilisé est affiché, arrondi à l'entier voisin le plus élevé.

Le label B demande une chaîne à trouver, stockant les 6 caractères les plus à gauche dans le registre 00 pour les recherches à venir. Le label 08 réalise la recherche (ligne 92), calcule et affiche le numéro d'entrée (lignes 95-101), place le pointeur au début de l'entrée (102-103), et appelle le label 95 pour afficher l'entrée. La recherche est alors reprise, à l'aide des 6 caractères les plus à gauche de la chaîne cible. Lorsque le END OF FL est atteint, l'instruction POSFL retourne une valeur de -1 à X, et le test à la ligne 93 arrête le programme.

Le label C appelle le label 91 pour placer le pointeur à l'entrée choisie, puis tombe sur le label 95 pour afficher l'entrée... Le label C appelle d'abord le label 90 pour initialiser le compteur de caractère. Puis il place le pointeur au début du fichier (ligne 145). Les lignes 146-148 préparent la valeur du pointeur qui sera nécessaire au passage suivant de la boucle. Si le END OF FL n'était pas atteint, le label 95 affiche les 6 lignes de l'entrée, et le GTO 10 passe à l'entrée suivante. Lorsque le END OF FL est atteint, (drapeau 25 effacé), le compteur de caractère en X est divisé par 7, donnant le nombre de registres occupé.

Les lignes 155-161 construisent et affichent ou impriment un message qui compare ce nombre avec FLSIZE.

Le label D appelle le label 91 pour placer le pointeur au début d'une entrée, puis il efface 6 enregistrements.

Le label E appelle le label 91 pour choisir une entrée. Le premier enregistrement de l'entrée est alors imprimé ou affiché (ligne 186). Puis la demande "OK?" est affichée et GETKEY est exécutée. Après qu'une touche ai été pressée ou que le délai expire sans qu'une touche ne soit pressée, les lignes 190-192 effectuent un GTO IND X, branchant au label désigné par le code de touche. Les labels 71 ("Y") et 84 (R/S) provoquent l'afichage de la ligne suivante. Les labels 00 (pas de touche) et 04 (ALPHA) renvoient le pointeur au début de l'enregistrement courant, et provoquent le réaffichage de la ligne courante. Les labels 44 (touche d'effacement) et 01 (ON) provoquent l'arrêt du programme. Si aucune autre touche n'est pressée, l'exécution continuera à la ligne 193, parce que le drapeau 25 a été levé. Cette séquence demande une nouvelle ligne, puis utilise cette chaîne pour remplacer l'enregistrement courant (lignes 199-200).

CHAPITRE HUIT

TRAITEMENT DE TEXTE SUR HP-41

Ce chapitre introduit un programme de traitement de texte appelé "TE", qui permet à tout fichier de texte d'être revu et corrigé avec un minimum de frappes de touches. Comme le programme "HP-16" du chapitre 9, il utilise la fonction GETKEY pour obtenir un remarquable degré de commodité et d'agrément pour l'utilisateur. Si vous possédez une HP-41CX, sa fonction d'origine ED fait essentiellement le même travail que ce programme de traitement de texte. L'avantage majeur de ED est qu'il répond beaucoup plus rapidement que "TE". Les avantages de "TE" sont ses possibilités de recherche et d'entrée de caractères spéciaux (impossibles à taper). Même si vous possédez une HP-41CX, ne négligez pas les possibilités que fournit "TE".

Ce programme puissant et la documentation qui s'y rattache ont été écrits par Erick Christensen, et sont reproduits ici par autorisation.

"TE" est un programme de traitement de texte à utiliser avec une HP-41C ou une CV qui possède un module XFM branché. Le programme fonctionnera également avec une HP-41CX. Les modules XFM supplémentaires sont optionnels (on peut en utiliser jusqu'à deux). La HP-41 utilisée doit avoir au moins 115 registres de programme libres, et un registre de données libre.

Le programme "TE" fournit un moyen rapide de visualiser, d'ajouter, d'effacer, et de charger du texte dans un fichier de texte (ASCII) de mémoire d'extension. Un mode de correction est inclus qui redéfinit totalement le clavier pour le traitement du fichier. Dans ce mode de correction, vous pressez une certaine touche qui correspond à l'opération que vous souhaitez réaliser sur le fichier. Ensuite vous êtes questionné en conséquence, et la correction se poursuit. Vous voyez le fichier à travers une "fenêtre" de douze caractères, que vous pouvez déplacer tout le long du fichier à l'aide des différentes opérations à une seule touche. On vous indique si une erreur se produit, mais l'exécution n'est pas interrompue. Les pages suivantes décriront les différentes fonctions des touches (A) - (H). La dernière décrite sera la touche (E), car elle représente la majeure partie du programme, y compris le mode de correction.

Comme pour "NAP" au chapitre 7, il vous faut exécuter "SK" ou effacer tout assignements des touches (A) - (H) avant que vous puissiez utiliser effectivement "TE". La section 10G donne une pleine explication. Une fois que les assignements de ces touches sont effacés, exécutez "TE" pour démarrer le traitement de texte. Les touches (A) - (H) seront alors redéfinies comme suit :

rangée 1 : Ajouter Compter Effacer Supprimer Corriger

rangée 2 : Table Aller à Aide

Voici une brève explication de chacune de ces fonctions :

(A) Ajouter (un fichier à la mémoire)

Cette routine met en place l'attribution de la mémoire pour un nouveau

fichier de texte dans la mémoire d'extension. Le programme demande le nombre de lignes que le fichier de texte doit avoir ("LINES"). Tapez un nombre et ensuite pressez R/S. Le programme alors demande le nombre total de caractères qui seront attribués au fichier ("CHAR").

Tapez un nombre, et pressez R/S. La routine demande alors le nom du nouveau fichier("NAME?"). Tapez un nom de fichier qui a une longueur de 1 à 7 lettres, et pressez R/S. Si le nom tapé a déjà été utilisé comme nom d'un autre fichier, alors on vous redemandera un autre nom. S'il n'y a pas assez de mémoire libre disponible pour créer le fichier, on vous redemandera encore une fois le nombre de lignes et de caractères qui doivent être attribués. A la fin, l'affichage indiquera "OK" et le programme s'arrêtera. Vous pouvez maintenant utiliser n'importe lequel des autres labels locaux.

(B) Compter (le nombre de caractères libres dans un fichier)

Cette routine vérifie le nombre de caractères qui peut être ajouté au fichier spécifié. La routine demande le nom du fichier à analyser ("FILE NAME"). Tapez un nom de 1 à 7 caractères qui correspondent à un fichier déjà créé, et pressez R/S. Si vous spécifiez un nom qui n'est pas encore utilisé comme nom d'un fichier ou le nom d'un programme ou d'un fichier de données, la question vous sera reposée. Lorsqu'un fichier aura été choisi, le programme se mettra à compter les caractères libres, puis s'arrêtera avec "CHAR LEFT=n"où n équivaut au nombre de caractères libres. Pressez R/S et vous verrez "OK" à l'affichage. Vous pouvez maintenant utiliser n'importe quelle autre routine.

(C) Effacer (le contenu d'un fichier)

Cette routine efface tout le texte stocké dans un fichier, mais laisse le fichier intact, y compris le nom, l'attibution de la mémoire, et l'emplacement de la table. La routine demande le nom du fichier à effacer ("FILE NAME?"). Tapez le nom du fichier à effacer (1 à 7 caractères), et pressez R/S. Si vous spécifiez un nom qui n'a pas déjà été utilisé dans la mémoire, ou le nom d'un fichier de données ou de programme, on vous redemandera le nom du fichier. La routine se termine avec "OK" à l'affichage. Vous pouvez maintenant utiliser n'importe quelle autre routine.

(D) Supprimer (un fichier)

Cette routine purge un fichier de texte de la mémoire d'extension. Cette routine est celle à utiliser pour faire de la place pour de nouveaux fichiers, en effaçant les anciens. La routine demande le nom du fichier à effacer ("FILE NAME?"). Entrez le nom du fichier (longueur de 1 à 7 caractères) et pressez R/S. Si vous spécifiez un nom de fichier qui n'existe pas dans la mémoire, ou le nom d'un fichier de donnée ou de programme, la question vous sera reposée. La routine se termine avec "OK" à l'affichage. Vous pouvez maintenant utiliser n'importe quelle autre routine.

(F) Table (des fichiers)

Cette routine indiquera les nom, type et allocation mémoire pour chaque fichier de la mémoire d'extension, y compris les fichiers de données et de programme. L'ordre de visualisation est l'ordre dans lequel les fichiers ont été créés, ainsi le premier fichier montré est le premier

créé. L'affichage de chaque fichier est : FFFFFFF TMMM, où FFFFFFF est le nom du fichier de 1 à 7 caractères, T dénote le type de fichier, et MMM représente le nombre de registres de 7 caractères qui ont été alloués au fichier. T peut être A, D, ou P, où A=fichier de texte ASCII, D=fichier de données et P=fichier de programme. Lorsque tous les fichiers de la mémoire d'extension ont été listés, le programme s'arrête avec "FREE=n" à l'affichage, où n équivaut au nombre de caractères libres qui peuvent être alloués aux fichiers de la mémoire d'extension. Pressez R/S une fois de plus, et "OK" sera affiché. Vous pouvez maintenant utiliser n'importe quelle autre routine.

(G) Aller à (un fichier)

Cette routine est utilisée pour positionner le traitement de texte dans n'importe quel fichier de texte dans la mémoire. Une fois positionné dans un fichier de texte, vous pouvez réaliser n'importe quelle opération sur le fichier courant, en frappant simplement R/S après la demande "FILE NAME?". Ceci marche pour les routines (B) et (E). Pour positionner le traitement de texte dans un fichier spécifique, répondez à la question de cette routine ("FILE NAME?") avec le nom du fichier désiré, et pressez R/S. Si le fichier spécifié n'est pas trouvé, ou est le nom d'un programme ou d'un fichier de données, on vous redemandera le nom. A la fin, "OK" sera à l'affichage. Vous pouvez alors utiliser n'importe quelle autre routine.

(H) Aide (associée à une touche)

Cette routine est une routine de commodité pour déterminer la fonction d'une certaine touche de label local (A)-(G). Elle affichera un mnémonique qui correspond à la touche pressée. Il posera la question "PROBLEM KEY?". Répondez en pressant la touche qui vous pose problème. L'affichage indiquera un mnémonique de 3 caractères. Les codes des touches (A)-(G) sont comme suit : (A)=ADD, (B)=FRE, (C)=CLR, (D)=DEL, (E)=ED ,(F)=DIR, (G)=GTO (Note du traducteur : ces mnémoniques sont des abréviations des termes anglais, vous pouvez les remplacer à votre guise dans le programme ; nous n'avons pas fait ce remplacement pour éviter d'éventuelles erreurs de traduction des programmes). Le mnémonique apparaîtra environ une seconde, puis la routine reviendra demander une autre touche. Elle continuera à procéder de cette façon jusqu'à ce que vous pressiez une autre touche, ou lorsque vous presserez une tout autre touche que (A)-(G), la routine s'arrêtera de demander et affichera "OK", signifiant qu'il est possible d'utiliser n'importe quelle autre routine.

(E) Corriger (un fichier)

Lorsque vous pressez la touche "E", vous entrez dans en mode de correction. Le clavier est complètement redéfini pour les opérations sur fichiers. Toutes les entrées seront faites durant le fonctionnement du programme, à l'exception de l'entrée de caractères ALPHA, pour laquelle le programme s'arrêtera, et R/S sera nécessaire pour continuer.

Pour choisir le mode de correction, pressez la touche (E). Vous verrez le message "FILE NAME?" à l'affichage. Tapez le nom du fichier à corriger (1 à 7 caractères) et pressez R/S. Si vous spécifiez le nom d'un fichier de programme ou de données, on vous redemandera un nom. Si un fichier de texte a déjà été choisi à l'aide du label local (G), ce fichier sera utilisé si vous pressez simplement R/S sans entrée de nom. Après que le nom ait été

spécifié, vous verrez une portion du texte du fichier à travers une fenêtre de 12 caractères (qui peut être déplacée à l'aide de plusieurs commandes). La fenêtre sera initialement placée à la ligne 0, caractère 0. Vous verrez aussi l'annonciateur du drapeau 0. Cela signifie que la calculatrice est prête pour n'importe quelle fonction de correction de votre choix. Lorsque le drapeau 0 est levé, vous pouvez presser la touche qui correspond à la fonction que vous désirez. On vous demandera alors, si nécessaire, l'entrée pour la fonction choisie, et l'écran montrera de nouveau la fenêtre. Ensuite vous pourrez choisir d'autres touches pour corriger le fichier. Quand vous avez fini, pressez la touche R/S ou la touche ON lorsque le drapeau 0 est levé pour quitter le mode correction.

Le clavier du mode correction est disposé comme indiqué dans le tableau ci-dessous. Chaque touche réalise une opération différente. Note : Aucun assignement de touches du mode USER n'est affecté par ce clavier redéfini.

Rangée 0	quit	INS X			
rangée 1	ADD L	BEG L	CHA L	DEL L	INS L
rangée 2	BACK n	GOTO A	POINT	INS A	AHEADn
rangée 3	BACK 1	DEL A	VIEW L	CHA A	AHEAD1
rangée 4	ADD A		GTOCHR	POS FL	DELCHR
rangée 5	UP n	7		8	9
rangée 6	UP 1	4		5	6
rangée 7	DOWN 1	1		2	3
rangée 8	DOWN n	0	GOTO REC		STOP/CONT

Chaque opération du mode correction listée sur le diagramme du clavier ci-dessus sera maintenant décrite en détail. Les opérations seront identifiées par les mnémoniques sur le diagramme du clavier, et listées de haut en bas du clavier.

(ADD L) Ajouter une ligne de texte à la fin du fichier

Cette opération créera une nouvelle ligne de texte que vous spécifiez à la fin du fichier. La fenêtre sera diposée sur le texte nouvellement créé, à la fin du fichier. Si il n'y a pas assez de place allouée pour le nouveau texte, alors "ERROR" sera affiché, et l'opération ne sera pas réalisée. On vous demandera le nouveau texte à ajouter ("NEW TEXT?"). Tapez une chaîne de caractères qui peut avoir une longueur de 1 à 24 caractères, et pressez R/S. La routine retournera à l'affichage de la fenêtre.

(BEG L) Déplacer la fenêtre au début de la ligne

Cette routine positionnera la fenêtre aux 12 premiers caractères de la ligne sur laquelle la fenêtre est présentement positionnée. Si la fenêtre est déjà au début de la ligne, rien ne se produit, la routine ne pose pas de question, et retourne immédiatement à l'affichage de la fenêtre.

(CHA L) Changer le contenu d'une ligne

Cette routine changera le texte d'une ligne pour un nouveau texte, effaçant ainsi l'ancien texte. On vous demandera le texte qui doit recouvrir le texte précédent sur la ligne courante ("NEW TEXT?"). Répondez à la question en tapant une chaîne de 1 à 24 caractères qui remplacera le vieux texte, et pressez R/S. S'il n'y a pas assez de place pour le nouveau texte, alors "ERROR" sera affiché. La fenêtre est positionnée au début du texte nouvellement créé. La routine retournera à l'affichage de la fenêtre.

(DEL L) Effacer une ligne de texte

Cette routine effacera une ligne entière de texte de la mémoire. Toutes les lignes qui suivent remonteront d'un cran, ainsi la fenêtre sera positionnée sur la ligne suivante de la mémoire. Cette routine ne pose pas de question, et retourne immédiatement à l'affichage de la fenêtre.

(INS A) Insertion de texte dans une ligne

Cette routine vous permettra d'insérer un texte à la position courante de la fenêtre. Le texte inséré apparaîtra juste avant le texte couramment montré dans la fenêtre. La fenêtre sera positionnée au début du texte nouvellement inséré. La demande est "NEW TEXT?". Répondez avec la chaîne de texte de 1 à 24 caractères qui doit être insérée, et pressez R/S. S'il n'y a pas assez de place pour le texte à insérer, alors "ERROR" sera affiché. La routine retourne à l'affichage de la fenêtre.

(AHEAD n) Déplacement de la fenêtre en avant de n caractères

Cette routine déplacera la fenêtre en avant à travers la ligne courante de texte de n caractères. La routine demande "NUMBER?". Entrez une séquence de trois touches qui représente un nombre de trois chiffres pour n. La routine montre alors le nombre de trois chiffres à l'affichage, et retourne au nouvel affichage de la fenêtre. Si la fenêtre était déplacée au delà de la fin de la ligne courante alors "ERROR" sera affiché.

(BACK 1) Recul de la fenêtre d'un caractère

Cette routine reculera la fenêtre d'un caractère sur la ligne. Si vous reculez au delà du premier caractère, alors "ERROR" est affiché. La routine retourne alors à l'affichage de la fenêtre.

(DEL A) effacement du texte spécifié

Cette routine permettra à l'utilisateur d'effacer certaines chaînes de texte de la ligne. La routine demande le texte qui doit être effacée en affichant "OLD TEXT?". Entrez une chaîne de 1 à 24 caractères, et pressez R/S. Si le texte à effacer n'est pas trouvé, rien ne se produit alors. La recherche du texte est faite à partir de la position courante de la fenêtre à la fin du fichier. La routine retourne afficher la fenêtre.

(VIEW L) Visualisation d'une ligne entière

Cette routine visualise la ligne courante, 12 caractères à la fois. Elle va du début de la ligne à la fin. D'abord elle affiche le numéro de ligne "LINE n", où n équivaut au numéro de ligne. Puis elle visualise la

ligne et revient à l'affichage de la fenêtre.

(INS L) Insertion d'une ligne de texte

Cette routine insérera une ligne d'un texte spécifié par l'utilisateur dans le fichier. Le texte sera inséré juste avant la ligne courante. Après l'insertion du texte, la fenêtre sera positionnée au début de la nouvelle ligne de texte. S'il n'y a pas assez de place pour le nouveau texte, "ERROR" apparaîtra à l'affichage. La demande du texte qui doit être inséré est "NEW TEXT?". Répondez par une chaîne de 1 à 24 caractères à insérer, et pressez R/S. La routine retourne alors à l'affichage de la fenêtre.

(BACK n) recul de n caractères

Cette routine reculera la fenêtre à travers la ligne de n caractères où n peut être choisi. Si en reculant des n caractères la fenêtre dépassait le début du texte, la fenêtre serait alors positionnée au début du texte. La demande pour n est "NUMBER?". Répondez en pressant une séquence de trois touches représentant un nombre de trois chiffres pendant que le programme fonctionne. Le nombre de trois chiffres apparaîtra à l'affichage, et ensuite le programme retournera afficher la fenêtre à sa nouvelle position.

(GOTO A) aller à un texte dans une ligne

Cette routine cherchera dans la ligne courante une correspondance avec un texte spécifié, et si une correspondance est trouvée, elle déplacera la fenêtre au premier caractère du texte spécifié. La routine ne cherchera que dans la ligne courante, et si on ne trouve pas de correspondance, la fenêtre ne change pas de position. La demande est "TARGET TEXT?". Répondez en tapant la chaîne qui doit être retrouvée (1 à 24 caractères) et pressez R/S. La routine retournera alors afficher la fenêtre.

(POINT) Visualisation des valeurs courantes du pointeur

Cette routine montrera à combien de lignes et à combien de lettres à partir de la position la plus à gauche se trouve la fenêtre, par rapport au fichier. La routine ne pose pas de question. Elle affiche le message "LINE n CHR m" où n est le nombre de lignes vers le bas à partir du début, et m est le nombre de caractères à partir du début de la ligne.

(CHA A) Changement de texte dans une ligne

Cette routine vous permettra de remplacer un texte dans une ligne. On vous demande le texte qui doit être changé (OLD TEXT?"). Entrez une chaîne de 1 à 24 caractères représentant le texte à changer. Ce texte est alors effacé, si on peut le trouver. La routine demande alors "NEW TEXT?". Tapez le texte qui doit remplacer l'ancien texte (1 à 24 caractères), et pressez R/S. Le nouveau texte est alors inséré là où l'ancien texte se trouvait. S'il n'y a pas assez de place pour le nouveau texte, alors "ERROR" est affiché. La routine retourne afficher la fenêtre.

(AHEAD 1) déplacement de la fenêtre d'un caractère en avant

Cette routine déplace la fenêtre d'un caractère en avant sur la ligne. Si la fenêtre sort du texte, alors "ERROR" est affiché. La routine retourne afficher la fenêtre.

(ADD A) Ajouter du texte à la fin d'une ligne

Cette routine ajoutera un texte spécifié par l'utilisateur à la fin de la ligne courante. S'il n'y a pas assez de place pour le nouveau texte, alors le message "ERROR" sera affiché. La demande est "NEW TEXT?". Entrez de 1 à 24 caractères qui doivent être ajoutés à la ligne, et pressez R/S. La routine retourne à l'affichage de la fenêtre.

(GOTOCHR) Déplacer la fenêtre à un numéro de caractère absolu

Cette routine déplacera la fenêtre à un numéro de caractère spécifié (compté depuis le début de la ligne). S'il n'y a pas de caractère à la position spécifiée, alors le message "ERROR" sera affiché. A la demande "NUMBER?", tapez les trois chiffres représentant la position du caractère dans la ligne. Le numéro sera affiché, et la routine retournera à l'affichage de la nouvelle fenêtre.

(POS FL) Positioner la fenêtre sur un texte spécifié

Cette routine cherchera depuis le début du fichier une correspondance au texte spécifié. Si on trouve une correspondance, alors la fenêtre est positionnée au premier caractère du texte que l'on cherche. Si on ne trouve pas de correspondance, alors la fenêtre n'est pas changée de position. La demande pour le texte à retrouver est "TARGET TEXT?". Vous répondez par une chaîne de 1 à 24 caractères suivie de R/S. La routine retourne afficher la fenêtre à sa nouvelle position.

(DELCHR) Effacement de n caractères

Cette routine efface n caractères de la ligne courante. L'effacement commence au premier caractère affiché par la fenêtre, et traite n caractères à droite.

Si n est plus long que le nombre de caractères du début de la fenêtre à la fin de la ligne courante, tout ce qui part du premier caractère de la fenêtre jusqu'à la fin de la ligne est effacé. La demande pour n, le nombre de caractères qui doit être effacé, est "NUMBER?". Tapez un nombre n de trois chiffres, avec des zéros de tête si nécessaire. Ce nombre sera alors affiché brièvement, et la routine retournera montrer l'affichage de la fenêtre ainsi mise à jour.

(UP n) Déplacement de la fenêtre à n Lignes

Cette routine déplacera l'affichage de la fenêtre vers le haut d'un nombre de lignes spécifié. Si la fenêtre était déplacée au delà du début du fichier (première ligne) alors la fenêtre sera positionnée à la première ligne. La demande pour le nombre de lignes qui doit être déplacé vers le haut est "NUMBER?". Répondez par une séquence de trois chiffres, et le nombre de trois chiffres sera affiché. Puis la routine retournera visualiser l'affichage de la fenêtre.

(UP 1) Déplacement de la fenêtre d'une ligne vers le haut

Cette routine déplacera la fenêtre d'une ligne de texte vers le haut. Si la fenêtre est positionnée à la première ligne, alors rien ne se produira. Cette routine ne pose pas de question, et retourne directement à

l'affichage de la fenêtre.

(DOWN 1) Déplacement de la fenêtre d'une ligne vers le bas

Cette routine déplace la fenêtre d'une ligne de texte vers le bas. Si vous essayez de déplacer la fenêtre au delà de la dernière ligne de texte du fichier, alors "ERROR" sera affiché. Cette routine ne pose pas de question, et retourne directement à l'affichage de la fenêtre.

(DOWN n) Déplacement de la fenêtre de n lignes vers le bas

Cette routine déplace l'affichage de la fenêtre vers le bas d'un nombre de lignes spécifié. Si la fenêtre était déplacée au delà de la fin du fichier (dernière ligne) alors la position de la fenêtre sera inchangée, et "ERROR" sera affiché. La demande du nombre de lignes à descendre est "NUMBER?". Répondez par une séquence de trois chiffres, et le nombre de trois chiffres sera affiché. Puis la routine retournera visualiser l'affichage de la fenêtre.

(GOTO REC) Déplacement de la fenêtre vers une ligne spécifiée.

Cette routine déplacera la fenêtre vers une lignes spécifiée. Les lignes sont numérotées à partir de la ligne 0 (la première ligne du fichier). La demande pour n est "NUMBER?". Répondez par un nombre de trois chiffres pour le numéro de ligne n, avec des zéros de tête si nécessaire. Le nombre de trois chiffres sera alors affiché, et la routine retournera à l'affichage de la fenêtre. Si vous essayez de déplacer la fenêtre vers une ligne de texte qui n'a pas encore été créée, alors l'affichage montrera "ERROR" et la position de la fenêtre ne changera pas.

(STOP/CONT) Sortie du mode de correction

Cette routine sort du mode de correction, vous évitant d'utiliser n'importe quelle autre routine utilisant un label local (A)-(H). Elle rétablit le statut original des drapeaux, et se termine par "OK" à l'affichage. La touche R/S aussi bien que la touche ON provoqueront l'exécution du programme de sortie.

(INS X) Insertion d'un caractère spécial

Cette routine insèrera un caractère qui normalement ne peut être tapé dans le fichier, à la position courante de la fenêtre. L'entrée du programme est le code ASCII (un nombre entre 0-255).

Une liste de ces caractères spéciaux et de leurs codes correspondants se trouve aux pages 40 et 41. La routine demandera le code de caractère en affichant "NUMBER?". Tapez une séquence numérique de trois chiffres qui représente le code de caractère. Le code sera affiché, et ensuite la routine retournera à l'affichage de la fenêtre. S'il n'y a pas assez de place pour le nouveau caractère du fichier, alors "ERROR" sera affiché.

Si une de ces descriptions de fonction n'était pas complètement claire pour vous, vous devez créer un petit fichier ASCII dans lequel vous pouvez essayer les fonctions du mode de correction. Vous trouverez que les descriptions représentent une excellente référence après avoir effectivement utilisé "TE".

ADAPTATION DE "TE" A VOS BESOINS

L'information qui suit est fournie dans le cas où vous voudriez ajouter vos propres programmes de correction à "TE" ou changer un programme préexistant. Si vous n'envisagez pas de modifier "TE", vous pouvez passer au chapitre suivant.

Pour ajouter ou changer une routine :

1) Choisissez la touche à laquelle votre nouvelle routine doit être assignée. Pressez (XEQ) (ALPHA) G E T K E Y (ALPHA) et cette touche.

2) Le code de touche rangée/colonne en X obtenu au pas 1 est le numéro du label numérique qui débutera votre nouvelle routine. Ainsi, par exemple, si vous voyiez 41 à l'affichage provenant de l'exemple 1, alors votre routine doit démarrer avec le LBL 41.

3) Après le label, au début de votre routine, vous pouvez ajouter tout ce qui vous plait, suivi d'une instruction RTN.

4) Votre routine doit observer les contraintes suivantes :

FLAG/REG	AVANT	APRES
Drapeau 29	effacé	effacé
Drapeau 25	levé	levé
FIX	0	0
Drapeau 28	levé	levé
Drapeau 0-3,6	effacé	effacé
X	eee,ccc	eee,ccc
Y	non utilisé	peut être utilisé
Z	non utilisé	peut être utilisé
T	code de touche	peut être utilisé
L	non utilisé	peut être utilisé
ALPHA	la fenêtre	peut être utilisé
REG 00	état ini. des flags	ne pas utiliser

Lorsque votre routine arrive à l'instruction RTN, elle doit avoir la position de la nouvelle fenêtre en X sous la forme eee,ccc, où eee est le numéro d'enregistrement, et ccc le numéro de caractère du premier caractère de la fenêtre. Si votre routine efface tout le texte du fichier, elle ne doit pas se terminer par RTN, mais plutôt par GTO 26. Cela appelle une routine de traitement d'erreur.

5) Pour appeler diverses routines de question, faites l'opération suivante:

PAS DE PROGRAMME	DESCRIPTION	UTILISE
XEQ 09	"NUMBER?" n° en X	T, Z, L, ALPHA
XEQ 06	"TEXT?" txt en A	
XEQ 07	"NEW TEXT?" txt en A	
XEQ 08	"TARGET TEST?" txt en A	

Exemple d'adjonction d'une routine

Ajoutez une routine qui effacera dans le fichier tout texte qui correspond au texte fourni par l'utilisateur.

Utilisez pour le texte qui doit être effacé "TARGET TEXT?". Cette nouvelle routine est assignée à la touche (ALPHA).

LISTE DE LA ROUTINE

```
01 LBL 04      débute le programme par les assignements de labels
02 ENTER↑      met eee,ccc en y
03 XEQ 08      appelle la demande "TARGET TEXT?'
04 LBL 88      Commence la boucle d'effacement
05 RDN         maintenant eee,ccc est en X
06 POSFL       recherche dans le fichier le texte en ALPHA
07 X 0?        le texte a-t-il été trouvé?
08 CLA         si non, efface alors alpha
09 RDN         Place eee,ccc en X
10 ALENG       prend la longueur de texte
11 DELCHR      efface ce nombre de caractères
12 X≠0?        alpha a-t-il été effacé ?
13 GTO 88      si non, reprend la boucle pour chercher encore
14 RDN         place eee,ccc en X
15 SF 25       lève le drapeau d'erreur
16 SEEKPT      Essaie de positionner la fenêtre à son ancienne
               position
17 FS? 25      avez-vous réussi?
18 RTN         oui, alors retour
19 GTO 26      si non, appelle le traitement d'erreur
```

Pour l'adjonction de cette routine pressez : (GTO) (ALPHA) T E (ALPHA) (SHIFT) (RTN) (PRGM) et tapez la routine.

Maintenant, toutes les fois que (ALPHA) est frappée dans le mode de correction, vous verrez une demande "TARGET TEXT?". Tapez le texte qui doit être effacé tout au long du fichier, et pressez (R/S). La routine retournera alors à l'affichage de la fenêtre.

DESCRIPTION DES LABELS NUMERIQUES

Description de label	Description de label
00 Mode de correction, entrée	24 INS A
02 INS	25 AHEAD n
06 demande "TEXT?"	26 traitement d'erreur pour effacement
07 demande "NEW TEXT?"	31 BACK 1
08 demande "TARGET TEXT?"	32 DEL A
09 demande "NUMBER?"	33 VIEW L
10 boucle pour VIEW L	34 CHA A
11 ADD L	35 AHEAD 1
12 BEG L	41 ADD A
13 CHA L	42 GOTO CHR
14 DEL L	43 POS FL
15 INS L	44 DEL CHR
16 demande "FILE NAME?"	51 UP n
17 demande "NAME?" pour LBL A	61 UP 1
18 demande "OK"	71 DOWN 1
19 boucle pour LBL B	81 DOWN n
20 boucle d'échappement pour LBL B	83 GOTO REC
21 BACK n	84 sortie de mode correction
22 GOTO A	99 Boucle d'affichage de fenêtre
23 POINT	

RESUME DES ERREURS POSSIBLES :

Fonction	Signification du message "ERROR"
ADD L	Pas assez de place pour un nouveau texte
BEG L	pas de situation d'erreur
CHA L	Pas assez de place pour un nouveau texte
DEL L	pas de situation d'erreur
INS A	pas assez de place pour un nouveau texte
DEL A	pas de situation d'erreur
VIEW L	pas de situation d'erreur
INS L	pas assez de place pour un nouveau texte
GOTO A	pas de situation d'erreur
POINT	pas de situation d'erreur
CHA A	pas assez de place pour un nouveau texte
ADD A	pas assez de place pour un nouveau texte
GTO CHR	la nouvelle position dépasse les limites du texte
POSFL	pas de situation d'erreur
DELCHR	pas de situation d'erreur
GTO REC	la nouvelle position dépasse les limites du texte
STOP-CONT	pas de situation d'erreur
INS X	pas assez de place ou code ASCII illegal
AHEAD n	la nouvelle position dépasse les limites du texte
AHEAD 1	la nouvelle position dépasse les limites du texte
BACK n	pas de situation d'erreur
BACK 1	pas de situation d'erreur
UP n	pas de situation d'erreur
DOWN 1	la nouvelle position dépasse les limites du texte
DOWN n	la nouvelle position dépasse les limites du texte
UP 1	pas de situation d'erreur

Liste du programme « TE »

```
01*LBL 31
02 1 E-3
03 -
04 X<0?
05 CLX
06 RTN

07*LBL 35
08 1 E-3
09 +
10 RTN

11*LBL 71
12 INT
13 1
14 +
15 RTN

16*LBL 61
17 INT
18 1
19 -
20 X<0?
21 CLX
22 RTN

23*LBL 41
24 XEQ 07
25 APPCHR
26 RTN

27*LBL 34
28 SF 10
29 XEQ 32
30 SF 25
31 GTO 24

32*LBL 11
33 XEQ 07
34 APPREC
35 RCLPT
36 INT
37 RTN

38*LBL 13
39 XEQ 07
40 DELREC
41 INSREC
42 GTO 26

43*LBL 32
44 "OLD"
45 XEQ 06
46 POSFL
47 X<0?
48 CLA
49 ALENG
50 DELCHR
51 GTO 26

52*LBL 24
53 XEQ 07
54 INSCHR
55 RTN

56*LBL 43
57 0
58 SEEKPT
59 RDN
60 XEQ 08
61 POSFL
62 X<0?
63 RDN
64 RTN

65*LBL 22
66 INT
67 XEQ 08
68 POSFL
69 INT
70 X<>Y
71 X=Y?
72 LASTX
73 RTN

74*LBL 08
75 "TARGET"
76 GTO 06

77*LBL 07
78 "NEW"

79*LBL 06
80 "├ TEXT?"
81 AVIEW
82 CLA
83 AON
84 STOP
85 AOFF
86 RTN

87*LBL 42
88 INT
89 GTO 25

90*LBL 44
91 XEQ 09
92 DELCHR
93 GTO 26

94*LBL 51
95 INT
96 XEQ 09
97 -
98 X<0?
99 CLX
100 RTN

101*LBL 25
102 XEQ 09
103 1 E3
104 /
105 +
106 RTN

107*LBL 21
108 XEQ 09
109 1 E3
110 /
111 -
112 X<0?
113 INT
114 RTN

115*LBL 81
116 INT
117 XEQ 09
118 +
119 RTN

120*LBL 02
121 XEQ 09
122 CLA
123 XTOA
124 FS? 25
125 INSCHR
126 RDN
127 RTN

128*LBL 83
129*LBL 09
130 "NUMBER?"
131 AVIEW
132 "RHIJ>?"
133 64
134 XTOA
135 RDN
136 "├456"
137 SF 01
138 GETKEY
139 SF 02
140 GETKEY
141 SF 03
142 GETKEY
143 CF 03
144 CF 02
145 CF 01
146 POSA
147 RDN
148 POSA
149 RDN
150 POSA
151 RDN
152 CLA
153 ARCL T
154 ARCL Z
155 ARCL Y
156 ANUM
157 X<0?
158 GTO 09
159 AVIEW
160 RTN

161*LBL 14
162 DELREC
163 XEQ 26
164 RTN

165*LBL 15
166 XEQ 07
167 INSREC
168 RTN

169*LBL 33
170 FS? 55
171 SF 21
172 INT
173 "LINE "
174 ARCL X
175 AVIEW
176 SEEKPT

177*LBL 10
178 CLA
179 ARCL 00
180 ARCL 00
181 ARCLREC
182 ASHF
183 ASHF
184 AVIEW
185 FS? 17
186 GTO 10
187 LASTX
188 CF 21
189 RTN

190*LBL 23
191 ENTER↑
192 INT
193 "LINE"
194 ARCL X
195 "├CHAR"
196 LASTX
197 FRC
198 1 E3
```

Liste du programme « TE » (suite)

```
199 *
200 ARCL X
201 AVIEW
202 RDN
203 RDN
204 RTN

205♦LBL 12
206 INT
207 RTN

208♦LBL 26
209 RCLPT
210 SF 25
211 SEEKPT
212 FC?C 10
213 FS? 25
214 RTN
215 INT
216 SF 25
217 SEEKPT
218 FC? 25
219 XEQ 61
220 RTN

221♦LBL 01
222♦LBL 84
223 RCL 00
224 STOFLAG
225 GTO 18

226♦LBL A
227 "LINES?"
228 PROMPT
229 "CHAR?"
230 PROMPT
231 +
232 7
233 /
234 1
235 +

236♦LBL 17
237 "NAME?"
238 AON
239 CLD
240 STOP
241 AOFF
242 SF 25
243 RCLPTA
244 FS? 25
245 GTO 17
246 SF 25
247 CRFLAS
248 FC? 25
249 GTO A
250 GTO 18

251♦LBL B
252 XEQ 16
253 .
254 SEEKPT
255 FLSIZE
256 7
257 *
258 1
259 -

260♦LBL 19
261 SF 25
262 GETREC
263 FC?C 25
264 GTO 20
265 ALENG
266 -
267 FS? 17
268 GTO 19
269 1
270 -
271 GTO 19

272♦LBL 20
273 "CHAR LEFT="
274 RCLFLAG
275 FIX 0
276 CF 29
277 ARCL Y
278 STOFLAG
279 AVIEW
280 RDN
281 STOP
282 GTO 18

283♦LBL C
284 XEQ 16
285 SF 25
286 CLFL
287 FC? 25
288 GTO C
289 GTO 18

290♦LBL D
291 XEQ 16
292 SF 25
293 PURFL
294 FC? 25
295 GTO D
296 GTO 18

297♦LBL E
298 XEQ 16
299 RCLFLAG
300 STO 00
301 RDN
302 .
303 SEEKPT
304 FIX 0
305 CF 29
306 CF 27
307 GTO 99

308♦LBL G
309 XEQ 16
310 GTO 18

311♦LBL 16
312 "FILE NAME?"
313 AON
314 AVIEW
315 CLA
316 STOP
317 AOFF
318 SF 25
319 RCLPTA
320 FC?C 25
321 GTO 16
322 SF 25
323 POSFL
324 FC?C 25
325 GTO 16
326 RTN

327♦LBL F
328 EMDIR
329 7
330 *
331 "FREE="
332 ARCL X
333 PROMPT
334 GTO 18

335♦LBL H
336 5
337 "PROBLEM KEY?"
338 AVIEW
339 "DIRED DELCLRFRE"
340 "┣ADDGTO"
341 GETKEY
342 23
343 X<=Y?
344 GTO 18
345 RDN
346 10
347 -
348 X>Y?
349 +
350 10
351 MOD
352 -3
353 *
354 AROT
355 ASHF
356 ASHF
357 ASHF
358 AVIEW
359 GTO H

360♦LBL "TE"
361 SF 27
362 CF 21
363 CF 10
364 SIZE?
365 1
366 X>Y?
367 PSIZE

368♦LBL 18
369 CF 25
370 CLST
371 "OK"
372 PROMPT
373 GTO 18

374♦LBL 99
375 RCLPT
376 SF 25
377 CLA
378 ARCL 00
379 ARCL 00
380 ARCLREC
381 ASHF
382 ASHF
383 SEEKPT
384 AVIEW
385 SF 25

386♦LBL 00
387 SF 00
388 GETKEY
389 CF 00
390 RDN
391 XEQ IND T
392 SEEKPT
393 "ERROR"
394 FC? 25
395 AVIEW
396 GTO 99
397 END

803 BYTES
```

Liste du programme « HP-16 »

```
01♦LBL "HP-16"
   02 2
   03 X<>F
   04 16
   05 STO 00
   06 CLST
   07 CLA
   08 CF 21

   09♦LBL 05
   10 RDN
   11 AVIEW
   12 SF 00
   13 GETKEY
   14 CF 00
   15 X=0?
   16 GTO 05
   17 RDN
   18 SF 25
  19 XEQ IND T
   20 R↑
   21 GTO 05

   22♦LBL 82
   23 0
   24 GTO 10

   25♦LBL 72
   26♦LBL 73
   27♦LBL 74
   28 R↑
   29 71
   30 -
   31 GTO 10

   32♦LBL 62
   33♦LBL 63
   34♦LBL 64
   35 R↑
   36 58
   37 -
   38 GTO 10

   39♦LBL 52
   40♦LBL 53
   41♦LBL 54
   42 R↑
   43 45
   44 -

   45♦LBL 10
   46 48
   47 GTO 06

   48♦LBL 11
   49♦LBL 12
   50♦LBL 13
   51♦LBL 14
   52♦LBL 15
   53 R↑
   54 1
   55 -
   56 GTO 10

   57♦LBL 21
   58 15

   59♦LBL 10
   60 55

   61♦LBL 06
   62 FS?C 05
   63 GTO 10
   64 X<>Y
   65 +
   66 FS? 07
   67 GTO 09
   68 0
   69 X<>Y
   70 CLA
   71 SF 07

   72♦LBL 09
   73 XTOA
   74 X<> L
   75 X<> 00
   76 ST* Y
   77 X<> 00
   78 +
   79 RTN

   80♦LBL 10
   81 RDN
   82 SIGN
   83 RDN
   84 FC? 06
  85 RCL IND L
  86 STO IND L
   87 FS?C 06
   88 RTN
   89 GTO 07

   90♦LBL 44
   91 RCL 00
   92 /
   93 INT
   94 1
   95 CHS
   96 AROT
   97 RDN
   98 ATOX
   99 RDN
  100 RTN

  101♦LBL 01
  102♦LBL 84
  103 0
  104 X<>F
  105 RDN
  106 CF 25
  107 CLD
  108 STOP
  109 RTN

  110♦LBL 22
  111 2
  112 16
  113 GTO 10

  114♦LBL 23
  115 4
  116 10
  117 GTO 10

  118♦LBL 24
  119 8
  120 ENTER↑
  121 GTO 10

  122♦LBL 25
  123 16
  124 2

  125♦LBL 10
  126 STO 00
  127 RDN
  128 X<>F
  129 RDN

  130♦LBL 07
  131 CF 07
  132 CLA
  133 ENTER↑

  134♦LBL 08
  135 ENTER↑
  136 X<> 00
  137 ST/ 00
  138 MOD
  139 9
  140 ST- Y
  141 RDN
  142 7
  143 X<>Y
  144 X>0?
  145 ST+ Y
  146 X<=0?
  147 X<>Y
  148 RDN
  149 57
  150 ST+ Y
  151 RDN
  152 XTOA
  153 X<> L
  154 X<> 00
  155 1
  156 CHS
  157 AROT
  158 RDN
  159 INT
  160 X>0?
  161 GTO 08
  162 RDN
  163 RTN

  164♦LBL 41
  165 ENTER↑
  166 CF 07
  167 RTN

  168♦LBL 32
  169 X<>Y
  170 GTO 07

  171♦LBL 51
  172 CHS

  173♦LBL 61
  174 X<>Y
  175 ST+ Y
  176 X<>Y
  177 ABS
  178 GTO 07

  179♦LBL 81
  180 1/X

  181♦LBL 71
  182 X<>Y
  183 ST* Y
  184 X<>Y
  185 INT
  186 GTO 07

  187♦LBL 33
  188 SF 06

  189♦LBL 34
  190 SF 05
  191 END
```

297 BYTES

CHAPITRE NEUF

UN PROGRAMME SIMULATEUR DE HP-16C

Ce chapitre présente un programme qui simule quelques unes des fonctions de la calculatrice HP-16C. Puisque le programme exemple réalise des conversions de base, un petit rappel sur les bases de nombres est nécessaire.

Un nombre décimal wxyz a la valeur :

$$wxyz_{10}=W.10^3+x.10^2+y.10+z$$

où w, x, y, et z sont tout chiffre de 0 à 9. L'indice 10 indique la base 10. La notation hexadécimale (base 16) fonctionne de la même façon. Un nombre hexadécimal $qrst_{16}$ a la valeur :

$$qrst_{16}=q.16^3+r.16^2+s.16+t$$

où q, r, s et t sont tout chiffre hexadécimal de zéro à quinze. Comme il n'y a pas de chiffres ordinaires qui corresponde aux nombres de dix à quinze, c'est une notation standard de les emprunter à l'alphabet :

$$A_{16}=10,\ B_{16}=11,\ C_{16}=12,\ D_{16}=13,\ E_{16}=14,\ \text{et}\ F_{16}=15.$$

Par exemple :

$$C5_{16}=12.16+5=197,\ \text{et}$$

$$FF_{16}=15.16+15=255.$$

Ces mêmes principes s'appliquent aux bases de nombres autres que 10 ou 16. Chaque chiffre dans la représentation représente un coefficient d'une puissance de la base.

La conversion de base est une application fréquente des calculatrices programmables. En fait, la HP-16C est spécialisé en conversion de base et en opérations en base 2 (binaire) base 8 (octale) base dix (décimale), et base 16 (hexadécimale). Les touches labellées A à F sont fournies pour faciliter l'entrée des nombres hexadécimaux.

Un nombre en base 16 peut être entré en mode HEX, puis converti simplement au mode décimal en tapant la touche DEC. La calculatrice alors interprète toutes les entrées suivantes comme des nombres décimaux jusqu'à ce que le mode soit changé.

Le programme "HP-16" simule les fonctions de conversion de base de la HP-16. Pressez simplement XEQ ALPHA H P shift - shift 1 shift 6 ALPHA pour exécuter le programme, et le clavier est redéfini comme ci-dessous dans le tableau d'accompagnement. Les touches qui n'apparaissent pas sur le tableau ne font rien quand on les presse, parce qu'il n'y a pas de label numérique correspondant dans le programme.

"Clavier GETKEY de la "HP-16"

rangée 1	A	B	C	D	E
rangée 2:	F	Hex	DEC	OCT	BIN
rangée 3:	---	X<>Y	STO	RCL	---
rangée 4:	ENTER↑		---	---	flèche
rangée 5:	-		7	8	9
rangée 6:	+		4	5	6
rangée 7:	*		1	2	3
rangée 8:	/		0	---	quit

Bien que le programme soit quelque peu paresseux, il est destiné à simuler une HP-16C avec une pile à deux niveaux (les registres X et Y seulement). Le mode est indiqué par les annonciateurs de drapeaux. Le drapeau 1 dénote le mode hexadécimal, le drapeau 2 le mode décimal, le drapeau 3 le mode octal, et le drapeau 4 le mode binaire. L'annonciateur du drapeau 0, lorsqu'il est allumé, indique que la HP-41 est prête. Lorsqu'il n'est pas allumé, l'annonciateur du drapeau 0 indique qu'un calcul est en cours. Lorsque vous pressez une touche, la disparition de l'annonciateur du drapeau 0 signifie que votre entrée a été reconnue. Attendez que l'annonciateur du drapeau 0 réapparaisse avant de presser une autre touche.

Une série d'exemples rendra le fonctionnement de ce programme plus clair. D'abord, exécutez "HP-16" pour démarrer le programme. L'annonciateur du drapeau 1 signifie que vous êtes en mode HEX.

Tapez la touche "C" (rangée 1 colonne 3). L'annonciateur du drapeau 0 disparait brièvement, puis réapparaît et un "C" apparait à l'affichage. Tapez la touche "2" et "C2" apparaîtra. Si vous faites une erreur, pressez la touche d'effacement et le chiffre le plus à droite du nombre affiché sera supprimé.

Maintenant convertissez ce nombre en base 8 en pressant la touche "OCT" (rangée 2, colonne 4). Après une courte attente, le résultat "302" apparaitra. L'annonciateur du drapeau 3 indique le mode octal.

Ajoutons 7 à ce nombre. Ceci est fait juste comme vous l'attendiez. Tapez simplement 7. Attendez que le 7 apparaisse à l'affichage, puis tapez +. Les deux nombres 302_8 et 7_8 seront ajoutés et le résultat octal, 311_8, apparaitra à l'affichage.

Vous pouvez convertir ce nombre en binaire en pressant la touche BIN (rangée 2 colonne 5). Ceci prend un peu de temps à cause du nombre de chiffres qui ont besoin d'être codés, mais le résultat est 11001001_2. L'annonciateur du drapeau 4 indique le mode binaire.

Pour trouver l'équivalent décimal, pressez la touche DEC (rangée 2, colonne 3). L'annonciateur du drapeau 2 indique le mode décimal, et le

résultat 201 apparaît.

Pressez la touche X<>Y (rangée 3 colonne 2, pas rangée 2, colonne 1) et vous verrez que le nombre $C2_{16}$=194 a été dupliqué dans le registre Y lorsque le 7 a été ajouté. Cette duplication automatique est similaire à la façon dont le registre T est dupliqué lorsqu'une opération comme l'addition est réalisée normalement.

La touche ENTER↑ fonctionne aussi comme attendu, dupliquant X en Y et terminant l'entrée de chiffre.

Le programme "HP-16" vous permet de stocker et rappeler les nombres à partir des registres de données 01 à 15. Vous pressez simplement la touches STO ou la touche RCL, attendez que l'annonciateur du drapeau 0 réapparaisse, et pressez une touche de 1 à 9 ou de A à F pour désigner le registre. Ne stockez rien dans le registre 0, parce que ce registre est réservé pour contenir la base de nombre. La disponibilité des opérations STO et RCL allège les limitations de la pile à deux niveaux. L'opération RCL lève la pile, aussi vous n'avez pas besoin de faire ENTER↑ avant de faire un RCL.

Quand vous aurez fini avec le programme, pressez simplement la touche R/S ou la touche ON pour terminer et effacer les drapeaux. Les contenus des registres X et Y seront en X et Y comme nombres décimaux, quel qu'ai été le mode dans lequel vous étiez lorsque vous avez pressé R/S.

Le programme "HP-16" est un exemple de la façon dont vous pouvez changer complètement la personnalité de votre HP-41 avec la seule fonction GETKEY et un programme de taille modérée. Si vous avez une application qui nécessite ce degré de commodité pour l'utilisateur, GETKEY peut être exactement ce dont vous avez besoin.

Analyse ligne par ligne de "HP-16"

Le programme "HP-16" est composé d'un bon nombre de petits morceaux, dont chacun obéit à quelques règles de base. D'abord, la base est contenue dans le registre de données 00. Les pas 16, STO 00 au haut du programme ont pour effet de placer le mode hexadécimal (base 16). Les contenus de registres "X" et "Y" du programme sont contenus dans la pile, en X et Y à la fois avant et après l'instruction XEQ IND T (ligne 16). Les nombres contenus dans la pile sont toujours en mode décimal. Le nombre affiché est en fait une chaîne dans le registre ALPHA. Si le nombre affiché change autrement que par addition ou suppression d'un chiffre, la sous-routine LBL 07 est appelée à reconstruire la représentation ALPHA à partir du nombre décimal en X. LBL 10 est utilisé de façon répétée pour de courts sauts vers l'avant et vers le bas. Un numéro de label peut seulement être réutilisé de cette façon si aucun de ces sauts ne se croise.

Le LBL 05 est la principale boucle de ce programme. Elle utilise GETKEY pour lire le clavier, ensuite lève le drapeau 0 et exécute la propre sous-routine de la touche qui a été pressée. Le drapeau 0 est alors effacé, le nouveau contenu de ALPHA est affiché, et on essaie un autre GETKEY. Le drapeau 25 est levé pour éviter les arrêts dus à l'erreur lorsqu'une touche invalide est pressée accidentellement. Par incidence, l'effacement du drapeau 21 à la ligne 08 évite que le programme ne s'arrête à l'instruction AVIEW si une imprimante et branchée ou éteinte. Si vous voulez imprimer, effacez l'instruction CF 21.

Les entrées de chiffre sont traitées en plaçant la valeur décimale en Y, et la différence de cette valeur avec le code ASCII en X. Par exemple, lorsque vous pressez "C" (touche 13) la valeur 12 est placée en Y, et la

valeur 55 en X. Pour les lettres de A à F, le code ASCII est 55 plus la valeur arithmétique de 10 à 15. La séquence LBL 06 ajoute le caractère ASCII correct à ALPHA. Le drapeau 07 indique qu'une entrée de chiffres est en cours. Quand un chiffre est pressé alors qu'une entrée de chiffres n'était pas déjà en cours, la pile sera levée. Ceci est effectué par la séquence 0, X<>Y, CLA. Cette séquence est contournée par l'instruction GTO 09 si une entrée de chiffres est en cours. La séquence LBL 09 met alors à jour le nombre décimal en X en multipliant par la base (lignes 75 et 76) et en additionnant la valeur du nouveau chiffre (lignes 77 et 78).

Si le chiffre est pressé comme une partie de STO ou RCL, le drapeau 5 sera levé et la section LBL 10 à la ligne 80 réalise l'opération nécessaire (STO si le drapeau 6 est levé, RCL si le drapeau 6 est baissé). Pour les opérations RCL, la séquence LBL 07 est utilisée pour reconstruire une chaîne ALPHA correspondant au nouveau registre X.

La routine LBL 44 divise simplement les contenus courants du registre X par la base, réalisant effectivement un décalage d'un chiffre, puis elle supprime le caractère le plus à droite d'ALPHA.

La séquence de fin (LBL 01 ou LBL 84) utilise X<>F pour baisser les drapeaux de 0 à 7, puis elle efface ce qui a été affiché, avant l'arrêt. Vous pouvez recommencer en pressant à nouveau R/S. Mais la base ne s'affichera pas dans les annonciateurs de drapeaux tant que vous n'avez pas pressé une touche de mode de base.

Les labels de sélection de mode, de 22 à 25, placent un nombre en Y correspondant au drapeau à lever, et un nombre en X correspondant à la nouvelle base. La base est alors stockée dans le registre 00, et la fonction X<>F est utilisée pour placer le bon drapeau et effacer tous les autres.

Une fois que la base a été changée, le registre ALPHA a besoin d'être mis à jour pour correspondre. La séquence LBL 07 le fait. Après qu'ALPHA soit effacé, les chiffres sont calculés et placés en ALPHA, en se déplaçant de droite à gauche. L'instruction ENTER! précédant immédiatement LBL 08 sert à garder une copie du nombre codé. Ceci est nécessaire parce que le codage ALPHA détruit le nombre en X.

Près du haut de la boucle de LBL 08, la fonction MOD donne la valeur arithmétique du chiffre courant. Les 13 lignes suivantes convertissent la valeur arithmétique en équivalent ASCII, puis une fonction XTOA crée le caractère. La fonction AROT permute le nouveau caractère vers l'avant de la chaîne.

La valeur décimale courante est divisée par la base (c'est la raison de l'instruction ST/ 00 précédente), et la partie entière est prise. Cette procédure effectue l'équivalent arithmétique d'un décalage d'un chiffre. Si le résultat n'est toujours pas zéro, des chiffres supplémentaires restent à décoder, et la séquence LBL 08 est à nouveau réalisée.

La séquence ENTER!, LBL 41, pousse simplement un zéro dans la pile et efface le registre ALPHA pour préparer l'entrée du chiffre. La séquence X<>Y, LBL 32, échange X et Y et va à la séquence LBL 07 pour reconstruire ALPHA. Les séquences -, +, /, et * se branchent aussi à LBL 07 pour reconstruire ALPHA. Les deux instructions X<>Y et les instructions ST+Y ou ST*Y permettent à la valeur précédente de Y de rester inchangée en Y. Comme la routine LBL 07 ne peut traiter les nombres négatifs ou non-entiers, INT est utilisé après division et ABS après une soustraction.

Les séquences STO et RCL lèvent toutes deux le drapeau 5. Ce drapeau est utilisé pour indiquer que la nouvelle entrée de chiffre est vraiment un

numéro de registre. Le drapeau 6 est utilisé pour déterminer si un STO ou un RCL doit être réalisé.

J'espère que cet exemple vous convainc de l'extraordinaire puissance de GETKEY. Vos applications peuvent être plus simples ou plus complexes, mais les mêmes principes s'appliquent.

Codes numériques des fonctions pour « XF »

(les numéros XROM sont aussi inclus pour référence)

	-EXT FCN 2C	
1	ALENG	25,01
2	ANUM	25,02
3	APPCHR	25,03
4	APPREC	25,04
5	ARCLREC	25,05
6	AROT	25,06
7	ATOX	25,07
8	CLFL	25,08
9	CLKEYS	25,09
10	CRFLAS	25,10
11	CRFLD	25,11
12	DELCHR	25,12
13	DELREC	25,13
14	EMDIR	25,14
15	FLSIZE	25,15
16	GETAS	25,16
17	GETKEY	25,17
18	GETP	25,18
19	GETR	25,19
20	GETREC	25,20
21	GETRX	25,21
22	GETSUB	25,22
23	GETX	25,23
24	INSCHR	25,24
25	INSREC	25,25
26	PASN	25,26
27	PCLPS	25,27
28	POSA	25,28
29	POSFL	25,29
30	PSIZE	25,30
31	PURFL	25,31
32	RCLFLAG	25,32
33	RCLPT	25,33
34	RCLPTA	25,34
35	REGMOVE	25,35
36	REGSWAP	25,36
37	SAVEAS	25,37
38	SAVEP	25,38
39	SAVER	25,39
40	SAVERX	25,40
41	SAVEX	25,41
42	SEEKPT	25,42
43	SEEKPTA	25,43
44	SIZE?	25,44
45	STOFLAG	25,45
46	X<>F	25,46
47	XTOA	25,47

	-X EXT FCN	
49	ASROOM	25,49
50	CLRGX	25,50
51	ED	25,51
52	EMDIRX	25,52
53	EMROOM	25,53
54	GETKEYX	25,54
55	RESZFL	25,55
56	ΣREG?	25,56
57	X=NN?	25,57
58	X≠NN?	25,58
59	X<NN?	25,59
60	X<=NN?	25,60
61	X>NN?	25,61
62	X>=NN?	25,62

	-TIME 2B	
65	ADATE	26,01
66	ALMCAT	26,02
67	ALMNOW	26,03
68	ATIME	26,04
69	ATIME24	26,05
70	CLK12	26,06
71	CLK24	26,07
72	CLKT	26,08
73	CLKTD	26,09
74	CLOCK	26,10
75	CORRECT	26,11
76	DATE	26,12
77	DATE+	26,13
78	DDAYS	26,14
79	DMY	26,15
80	DOW	26,16
81	MDY	26,17
82	RCLAF	26,18
83	RCLSW	26,19
84	RUNSW	26,20
85	SETAF	26,21
86	SETDATE	26,22
87	SETIME	26,23
88	SETSW	26,24
89	STOPSW	26,25
90	SW	26,26
91	T+X	26,27
92	TIME	26,28
93	XYZALM	26,29

	-X TIME	
95	CLALMA	26,31
96	CLALMX	26,32
97	CLRALMS	26,33
98	RCLALM	26,34
99	SWPT	26,35

	- WAND 1F -	
129	WNDDTA	27,01
130	WNDDTX	27,02
131	WNDLNK	27,03
132	WNDSUB	27,04
133	WNDSCN	27,05
134	"WNDTST	27,06

CHAPITRE DIX

LA PROGRAMMATION SYNTHETIQUE

10A. Quest-ce que la programmation synthétique?

Les instructions synthétiques sont celles qui ne peuvent être entrées à partir du clavier par des moyens normaux. La création et l'utilisation des instructions synthétiques est appelé programmation synthétique. Des milliers d'instructions synthétiques peuvent être crées, en partant des TONES non-standard jusqu'aux instructions puissantes qui donnent accès aux registres système à usage général. La programmation synthétique ne nuira en aucun cas à votre HP-41, bien que des plantages (blocage temporaire du clavier) et /ou des MEMORY LOST puissent se produire si vous expérimentez dans des territoires non familiers. Référez-vous à la section K de ce chapître pour des astuces sur la façon dont on reconnait et on récupère une situation de plantage.

La programmation synthétique fonctionnera sur toutes les calculatrices de la famille HP-41, sans se soucier de la date de fabrication. Elle ne dépend que des aspects fondamentaux du système d'opérations internes de la calculatrice qui sont communs à toutes les modèles.

Les programmes présentés dans ce chapitre utilisent les techniques synthétiques pour entrer dans les zones de mémoire qui ne sont pas normalement accessibles. Celles-ci comprennent les registres qui contiennent les informations sur les assignements de touches et les en-tête de registres dans la mémoire d'extension.

Comme quelques unes des instructions ne peuvent être directement tapées, des codes-barres sont fournis dans l'annexe D. Si vous n'avez pas accès à un crayon lumineux, l'éditeur de ce livre peut vous fournir une cassette magnétique contenant tous les programmes de ce livre.

Une autre alternative est d'apprendre suffisamment de choses concernant la programmation synthétique de façon à pouvoir entrer vous-même ces programmes. La possession du ZENROM (cf. références) rend la programmation synthétique aussi simple que la programmation normale. La manière la plus facile de démarrer avec la programmation synthétique est d'acheter un exemplaire du livre "La programmation synthétique, c'est facile !" directement chez l'éditeur du présent livre.

Si vous aimez programmer votre HP-41, vous devriez vraiment apprendre la programmation synthétique. C'est pour le moins comme trouver une toute nouvelle machine, cachée dans votre HP-41 familière.

J'espère que vous vous amuserez avec les programmes présentés dans ce chapitre. Si vous êtes un programmeur synthétique expérimenté, vous apprécierez leur puissance et leur versatilité. Si vous êtes un néophyte, ils vous donneront un aperçu des possibilités de la programmation synthétique.

10B. Exécution à une seule touche des fonctions d'extension

Le programme présenté dans cette section vous permet d'exécuter toutes fonctions à partir du jeu de fonctions d'extension, en spécifiant simplement un code numérique pour la fonction.

Ce programme a été écrit par Clifford Stern, un "grand maître" de la programmation synthétique. Clifford est spécialisé dans les routines économiques en pression de touches et les programmes compliqués.

C'est une affaire simple que d'utiliser la fonction incorporée ASN pour assigner plusieurs des fonctions communément utilisées. Vous avez probablement déjà assigné EMDIR, la fonction de la table de la mémoire d'extension, à une touche commode. Peut-être avez-vous également assigné SAVEP et GETP. Ou bien, si vous avez utilisé des fichiers de données, vous avez pu assigner RCLPT, SEEKPT, SAVERX, et GETRX aux touches. Ces assignements sont utiles, mais peuvent rapidement utiliser une portion importante de votre clavier en mode USER.

Voudriez vous pouvoir assigner toutes les fonctions d'extension à une seule touche ? Impossible ? Ca ne l'est pas avec la programmation synthétique ! Tout ce dont vous avez besoin est un exemplaire du programme "XF".

Pour exécuter n'importe quelle fonction d'extension, placez simplement le code numérique de la fonction en X, et exécutez "XF". Les codes numériques sont listés devant chaque nom de fonction dans le tableau ci-contre. Une courte digression devrait rendre l'équivalence du code numérique plus claire.

Si vous tapez des lignes de programme à l'aide des fonctions d'extension pendant que le module de fonctions d'extension mémoire est "nom de fonction" ; si vous supprimez plus tard le module de fonctions d'extension mémoire, les lignes du programme seront affichées et imprimées sous la forme : XROM 25,xx.

La désignation XROM indique que la fonction réside dans une mémoire morte externe. Le nombre 25 identifie le module de fonctions d'extension. Le nombre xx à deux chiffres identifie la fonction spécifique à l'intérieur du module. C'est ce même code de fonction à deux chiffres qu'utilise le programme "XF".

Incidemment, comme le tableau ci-contre le montre, "XF" vous permet aussi d'exécuter les fonctions du module horloge (XROM 26) et les fonctions du crayon lumineux (XROM 27). Les entrées 49-62 et 95-99 ne sont valables que pour la HP-41CX.

Les codes-barres de "XF" se trouvent dans l'annexe D.

Les lignes synthétiques et leurs équivalents décimaux par octet sont :
Lignes 04 : 206, 118 ; ligne 05 : 145, 123
Ligne 09 : 254, 127, 0, 0, 1, 105, 0, 18, 0, 123, 145, 125, 206, 116, 166
ligne 17 : 206, 118 ; ligne 18 : 206, 125 ; ligne 19 : 106, 123
ligne 20 : 144, 117 ; ligne 21 : 145, 246 ; ligne 22 : 206, 119
ligne 24 : 206, 117 ; ligne 26 : 206, 118 ; ligne 27 : 206, 123
ligne 28 : 145, 125
(Utilisez cette information si vous tapez le programme à l'aide d'un chargeur d'octets, un voleur d'octets, ou tout autre technique synthétique).

Mise en place du programme "XF":

Le programme "XF" doit être le premier dans le catalogue 1. Cela signifie que vous devez effacer tous les autres programmes de la mémoire principale avant de charger "XF". Vous pouvez utiliser SAVEP pour copier quelques uns des programmes dans la mémoire d'extension, ou vous pouvez utiliser des cartes magnétiques ou des cassettes. Pour nettoyer la mémoire

Liste du programme « XF »

```
01♦LBL "XF"      10 RDN          19 X<> a        28 STO c
02 X<>Y          11 XTOA         20 RCL [        29 X<> L
03 SIGN          12 RDN          21 STO IND \    30 SAVEP
04 X<> \         13 501          22 X<> ]        31 CLD
05 STO a         14 SIZE?        23 CLA          32 END
06 RDN           15 ST- Y        24 X<> [
07 64            16 RDN          25 RDN          74 BYTES
08 ST+ Y         17 X<> \        26 X<> \
09 "├♦♦×i♦δ♦ш→t  "  18 X<> c      27 X<> a
```

Liste du programme « EFTW »

```
01♦LBL "EFTW"          09 64           17 X<> \        25 STO c
  02 RCL [             10 +            18 X<> c        26 RDN
  03 CLA       11 "├♦♦×i♦δ♦uu→u  "     19 RCL [        27 SAVEP
  04 STO [             12 XTOA         20 STO IND \    28 CLD
  05 AON               13 CLX          21 X<> ]        29 END
  06 PSE               14 502          22 CLA
  07 AOFF              15 SIZE?        23 X<> [        67 BYTES
  08 CLX               16 -            24 RDN
```

programme, chargez simplement le registre ALPHA avec le nom du premier programme en mémoire principale (Catalogue 1) et exécutez PCLPS.

Ensuite chargez le programme "XF" dans votre calculatrice à l'aide des codes barre, de cartes magnétiques, ou à l'aide des techniques de la programmation synthétique décrites au chapitre 3 de "La programmation synthétique, c'est facile !".

Si vous possédez une HP-41C, vous pouvez avoir besoin de changer la ligne 13. Ce nombre, normalement 501, doit être 263+64n, où n est le nombre de modules simples branchés. Ceci se rapporte aux modules de la mémoire principale, non aux modules de la mémoire d'extension. Par exemple, si vous utilisez deux modules de mémoire simples, la ligne 13 devrait être 373. Si vous avez un module quadruple, qui est l'équivalent de 4 modules simples, le nombre 501 est correct. Attention : La mise en place d'un mauvais nombre à la ligne 13 peut entraîner un MEMORY LOST. Si vous débranchez un module de mémoire principale sans changer ce nombre, vous frôlez le désastre. Naturellement il est possible sans problème de débrancher des modules de mémoire d'extension.

Pour un maximum de commodité, assignez "XF" à votre touche favorite. Pour ce faire, pressez shift ASN ALPHA X F ALPHA suivi de la touche (ou de shift suivi de la touche pour un emplacement shifté) à laquelle vous voulez que "XF" soit assigné. Vous devez probablement éviter d'assigner "XF" (ou toute autre fonction) à une touche de chiffre ou à la touche XEQ. Si vous avez la fonction SIZE assignée à une touche, vous pouvez utiliser cette touche pour "XF".

Avec "XF", vous pouvez commodément redimensionner en tapant la taille souhaitée, ENTER!, 30 (le code numérique pour PSIZE), et en exécutant "XF".

"XF" est un programme qui se modifie tout seul et qui fonctionne en construisant et en stockant une courte séquence d'instructions contenant la fonction d'extension requise (ligne 30) dans le premier programme de la mémoire. Le premier programme sera changé, sans s'occuper de savoir si c'est "XF" ou non. Si "XF" est le premier programme, comme il le devrait, la séquence d'instruction stockée rentrera juste, aussi ce n'est que la fonction d'extension (ligne 30) qui changera. La ligne 31, CLD, peut être effacée si vous possédez une HP-41CX. Son but est d'effacer l'affichage après EMDIR.

ATTENTION : Ne changez pas le programme "XF" avant d'être sur de conserver le même nombre d'octets entre le haut du programme et la ligne 30. Si vous changez ce nombre d'octets, la séquence d'instruction stockée terminera à la mauvaise place.

"XF" est un exemple de la puissance de la programmation synthétique. Les programmes qui se modifient eux-mêmes sont d'habitude plutôt compliqués, mais cela montre comment un simple programme peut faire un travail qui ne peut être raisonnablement fait sans la programmation synthétique.

Exemple 1 pour "XF":

La fonction de mémoire d'extension la plus fréquemment utilisée est probablement la fonction "table de mémoire d'extension", EMDIR. Le tableau des entrées de "XF" indique que le code numérique correspondant est 14. Aussi si vous pressez 14 XEQ "XF" (ne pressez que la touche assignée) la

table de la mémoire d'extension sera affichée.

Instructions pour l'emploi de "XF"

1. S'assurer que "XF" est le premier programme dans la mémoire principale en exécutant le catalogue 1.

La première chose que vous devez voir est LBL "XF". Vous n'avez pas besoin de vérifier le catalogue 1 à chaque fois, mais vous devez être certain que "XF" est au sommet.

2. Charger les registres X, Y, Z, et le registre ALPHA avec tel contenu qui sera nécessaire au moment où la fonction est exécutée. La chaîne en ALPHA est limitée à 14 caractères. Si d'autres caractères sont placés en ALPHA, c'est seulement les 14 caractères les plus à droite qui seront utilisés et le reste des caractères sera perdu.

3. Presser ENTER↑ et mettre le code numérique de la fonction en X. Les nombres qui étaient en X, Y, et Z sont maintenant en Y, Z, et T. Ne pas utiliser le code zéro. Le code zéro n'a aucun effet sur la HP-41CX, mais sur la HP-41C ou la CV il détruirai tous les assignements de labels globaux, ne laissant que des assignements "fantôme" qui agissent comme la fonction ABS.

4. Presser la touche assignée pour exécuter "XF". La fonction désignée sera exécutée. "XF" construit réellement une séquence d'octets dans le registre ALPHA, transfère la séquence aux lignes 27-30 et ensuite exécute la séquence.

Si vous interrompez "XF" et ne repartez pas immédiatement, vous risquez un MEMORY LOST. Quelques sauvegardes ont été prévues, mais si vous vous arrêtez entre les lignes 18 et 28 et que vous ne permettez pas à "XF" de terminer normalement, cela peut provoquer un MEMORY LOST.

5. Lorsque la fonction est terminée, le "canard volant" disparait. Si une erreur se produit, vous verrez le message d'erreur correspondant. Par exemple si vous utilisez "XF" pour exécuter GETX (fonction numéro 23) lorsque le fichier de travail n'est pas un fichier de données, vous obtiendrez le message FL TYPE ERR.

6. En vérifiant la ligne 30 de "XF" (GTO. "XF" et GTO. 030), vous pouvez trouver la fonction d'extension exécutée la dernière à l'aide de "XF". Ceci peut être assez utile.

Le programme "XF" élimine le besoin d'assignement aux touches des fonctions d'extension dans la mesure où vous utilisez ces fonctions dans le mode RUN (non-programmable). Si vous tapez un long programme qui utilise des fonctions d'extension, vous pouvez toujours assigner temporairement quelques unes des fonctions les plus fréquemment rencontrées.

Deux précautions doivent s'appliquer à "XF". D'abord, ne pas utiliser "XF" pour exécuter PCLPS (fonction numéro 27) avec le registre ALPHA vide. Ceci effacera tous les programmes de la mémoire principale, y compris "XF" lui-même. A moins que ce soit le résultat que vous vouliez, vous devez nommer un programme avant d'exécuter PCLPS. Par exemple, si vous voulez effacer tous les programmes à l'exception de "XF", placez le nom du second programme du catalogue 1 en ALPHA, placez 27 en X, et exécutez "XF".

La seconde précaution est de ne pas appeler "XF" d'un sous-programme du second niveau ou plus. C'est à dire, qu'on ne doit pas appeler "XF"

lorque deux RTN ou plus sont en attente. Le programme "XF" efface le registre **a** du système d'exploitation, qui contient l'information utilisée pour les RTN du troisième au sixième niveau.

Une autre version de "XF", écrite également par Clifford Stern, épargne quelques frappes par rapport à "XF". Cette version, appelée "EFTW" s'arrête en mode ALPHA pour une entrée de 7 caractères au plus. Les instructions d'utilisation sont autrement les mêmes que celles d'"XF". La ligne 14 doit être 236+64n, où n est le nombre de modules de mémoire simple densité présents. De plus, XYZALM (fonction numéro 93) ne peut être utilisée là où une entrée z différente de zéro est nécessaire, parce que le registre Z est changé à zéro lorque XYZALM est exécuté.

Les codes barres pour "EFTW" se trouvent à l'annexe D.

Les lignes synthétiques et leurs équivalents décimaux par octet :
ligne 02 : 144, 117 ; ligne 04 : 145, 117
ligne 11 : 254, 127, 0, 0, 1, 105, 0, 18, 0, 117, 117, 145, 125, 117, 166
ligne 17 : 206, 118 ; ligne 18 : 206, 125 ; ligne 19 : 144, 117
ligne 20 : 145, 246 ; ligne 21 : 206, 119 ; ligne 23 : 206, 117
ligne 25 : 145, 125

10C. La structure interne de la mémoire d'extension

Cette section met en évidence la disposition générale des fichiers de la mémoire d'extension, montrant les zones qui sont abimées par les fonctions du lecteur de carte VER et 7CLREG. Ensuite, pour les programeurs de synthétique avancée, les détails de la structure d'en-tête des fichiers et les moyens d'éviter la normalisation des données ("La programmation synthétique, c'est facile !") sont couverts.

La mémoire d'extension est composée de un, deux ou trois blocs de registres, selon que zéro un ou deux modules d'extension mémoire sont branchés. Le module de fonctions d'extension mémoire contient 128 mémoire contient 239 registres. Les tailles annoncées pour ces modules sont 127 et 238 respectivement, parce que le dernier registre dans chaque module est retenu. Ce dernier registre contient un pointeur du début du module suivant et un autre pointeur de la fin du module précédent. Ces pointeurs sont nécessaires pour un bon chaînage de fichier parce que l'ordre dans lequel les modules de mémoire d'extension sont utilisés peut varier si les deux modules ne sont pas installés.
La figure 10.1 montre l'organisation de la mémoire d'extension avec plus de détails, contenant des adresses absolues de registres de données pour ceux qui sont assez téméraires pour farfouiller dedans.

A l'intérieur des zones de mémoire d'extension non réservées, des fichiers sont stockés dans le même ordre que celui dans lequel ils apparaissent dans la table de mémoire d'extension. Chaque fichier a deux registres d'en-tête, comme indiqué figure 2.1. Un code de partition spécial (hexadécimal FF FF FF FF FF FF FF pour les programmeurs synthétiques) est stocké juste aprés le dernier registre du dernier fichier. Ce code sépare les parties utilisées des parties non utilisées de la mémoire d'extension. Il dit à la calculatrice que le reste de la mémoire d'extension est disponible pour de nouveaux fichiers.

Lorsque vous démarrez avec une table de mémoire d'extension vide, le

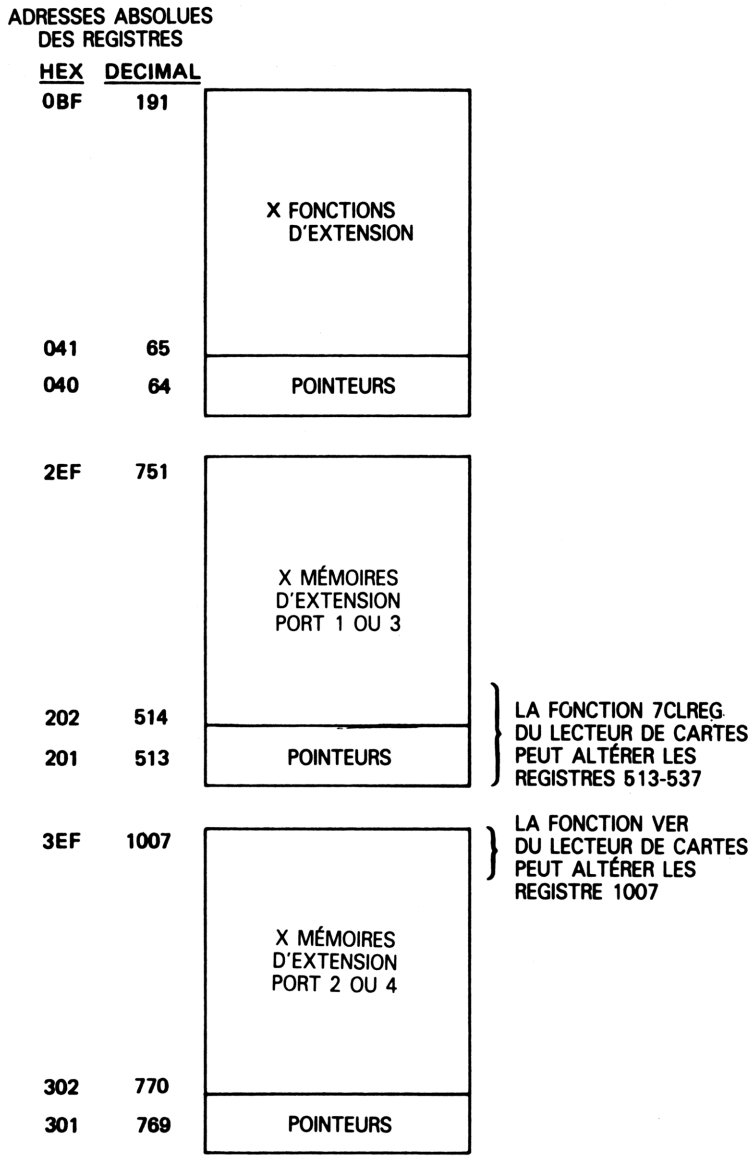


Figure 10.1 : Structure générale de la mémoire d'extension.

code de partition est au sommet du module de fonctions d'extension mémoire. Le premier fichier que vous créez occupera les registres supérieurs du module de fonctions d'extension mémoire, et déplacera le code de partion vers le bas. Lorsque vous créerez des fichiers, de nouveaux fichiers seront toujours ajoutés juste sous le dernier fichier, et le code de partition sera déplacé vers le bas.

En fin de compte vous utiliserez les 127 registres disponibles dans le premier bloc de la mémoire d'extension. S'il vous en faut davantage, le fichier déborde dans un module de mémoire d'extension. Habituellement le module de mémoire d'extension port 1 ou 3 sera utilisé avant le module port 2 ou 4. Les seules exceptions sont :

1) si il n'y a pas de module de mémoire d'extension au port 1 ou 3, ou

2) si le module port 2 ou 4 a été partiellement rempli avant que l'autre module soit installé.

Après un MEMORY LOST l'ordre naturel d'utilisation (port 1 ou 3 d'abord) sera restauré.

Structure détaillée des registres d'en-tête et de pointeur:

Chaque en-tête de fichier se compose de deux registres au sommet du fichier. Le premier de ces registres contient le nom de fichier, jusqu'à 7 caractères. Si le nom de fichier contient moins de 7 caractères, des espaces (hexadécimal 20) sont ajoutés à droite pour remplir les 7 octets du registre.

Le second registre d'en-tête de fichier contient plusieurs informations sur le fichier. La structure sera décrite ici en termes de chiffres, qui sont des chiffres hexadécimaux. Deux chiffres font un octet ; sept octets font un registre. Le chiffre le plus à gauche du second registre d'en-tête indique le type de fichier. Ce chiffre est 1 pour les fichiers de programmes, 2 pour les fichiers de données, et 3 pour les fichiers ASCII. Pour les fichiers de programmes, les 14 digits de ce registre sont:

10 00 00 00 BB BS SS

Où BBB est le nombre d'octets dans le programme sauvegardé (y compris le END) et SSS est le FLSIZE, en registres. Ces deux nombres sont en hexadécimal, non en décimal. Un programme dans le fichier de la mémoire d'extension a la même forme qu'un programme en mémoire principale, y compris le END. Le END est suivi par un octet de somme de contrôle qui contient la somme modulo 256 de tous les octets du programme. Ceci représente un octet "d'en-tête" en plus des deux registres d'en-tête de fichier de programme. Donc si le nombre d'octets du programme est 49 octets (7 registres), un fichier de 8 registres sera créé par SAVEP parce qu'un 50ième octet de somme de contrôle doit être inclu. Pour les fichiers de données, le second registre d'en-tête est :

2A, AA, 00, 00, RR, RS, SS

Où AAA est l'adresse absolue de ce second registre d'en-tête, RRR est le pointeur de registre, et SSS est la taille du fichier. Les registres sont numérotés 0,1, 2, etc., en commençant par le registre qui se trouve immédiatement sous le second registre d'en-tête. Pour les fichiers ASCII, le second registre d'en-tête est :

3A, AA, 00, CC, RR, RS, SS

Où CC est le pointeur de caractère, RR est le pointeur d'enregistrement, et AAA ainsi que SSS sont les mêmes que pour les fichiers de données.

Les registres de pointeur au bas de chaque bloc de la mémoire d'extension contiennent ces 14 digits :

OO, OW, WP, PP, NN, NT, TT

Où WW est le numéro du fichier de travail (01 et au-dessus), PPP est l'adresse absolue du registre inférieur du bloc précédent de la mémoire d'extension, NNN est l'adresse du registre supérieur du bloc suivant et TTT est l'adresse du registre supérieur de ce bloc. Le champ WW est utilisé seulement dans le module de fonctions d'extension mémoire. Le champ PPP n'est pas utilisé dans le module de fonctions d'extension mémoire, mais dans la HP-41CX il indique le fichier de travail précédent. Le champ NNN n'est pas utilisé dans le second module de mémoire d'extension. Tous ces registres de pointeurs sont initialisés quand un fichier est créé occupant une partie du module en question. Si une fonction liée à la mémoire d'extension a été utilisée, sans qu'aucun fichier ait été créé cependant, le champ TTT contiendra l'adresse du registre de pointeur lui-même.

Les digits de registres d'en-tête ou de pointeur qui ne sont pas utilisés n'ont pas besoin d'être à zéro. Par exemple, il est souvent commode pour un programme synthétique de changer le premier digit d'un registre de pointeur à 1, de façon que le registre puisse être rappelé comme donnée ALPHA.

Ceci soulève le sujet de la normalisation. Si un registre numéroté contient une représentation de bits qui ne représente pas un nombre et que la HP-41 ne reconnait pas comme donnée ALPHA, le contenu du registre peut être altérés lorsque le registre est rappelé. Ce point est discuté dans "La programmation synthétique, c'est facile !" ; les opérations qui normalisent les contenus d'un registre numéroté comprennent RCL ,ARCL, X<>, VIEW et toute opération indirecte.

Parmi les fonctions d'extension, plusieurs des opérations de SAVE et GET peuvent transférer les données sans normalisation. Cela rend possibles un bon nombre d'applications de la programmation synthétique avancée, y compris quelques uns des programmes de ce chapitre. GETX,SAVEX, GETR, SAVER, et GETRX, ne normalisent pas du tout les données. Les fonctions SAVERX et REGSWAP normalisent à la fois les extrêmes supérieurs et inférieurs du bloc de registres de données utilisé. L'opération REGMOVE normalise seulement le registre supérieur des données utilisé.

Lorsqu'un fichier est purgé de la mémoire d'extension, tous les fichiers situés sous ce fichier de la mémoire d'extension sont déplacés vers le haut pour remplir l'espace laissé par le fichier purgé. Si le fichier était le dernier dans la mémoire d'extension, aucun fichier n'est déplacé. Le code de partition est alors stocké juste au-dessous du dernier fichier qui reste. Les registres au delà de ce point ne sont pas effacés. Ils retiennent les même contenus qu'ils avaient avant que PURFL n'ait été exécuté, mais ils ne sont plus accessibles sauf par l'intermédiaire des techniques synthétiques.

Tous ces détails sont d'un intérêt d'abord pour la programmation synthétique avancée, mais ils illustrent le nombre et la variété des pointeurs que la calculatrice doit maintenir pour garder les choses simples pour l'utilisateur de la mémoire d'extension.

10D. Une solution au parasite VER

Le programme "VER" présenté ici prend la place de la fonction VER incorporée du lecteur de carte, en s'assurant que la mémoire d'extension n'est pas endommagée. Ceci est un autre chef-d'oeuvre de Clifford Stern. A moins que le lecteur de carte soit très neuf (révision 1G ou plus haut) ou à moins que vous n'ayez pas de module de mémoire d'extension en port 2 ou en port 4, ce programme vous est nécessaire. Le numéro de révision de votre module de fonctions d'extension est sans influence ici.

Deux versions du programme "VER" sont fournies dans les codes-barres à l'annexe D. Normalement vous devez utiliser la première version. La seconde version n'est utilisée que dans deux cas:

1) Si vous avez un seul module de mémoire d'extension en port 1 ou 3, ou

2) Si un module de mémoire d'extension a été branché dans le port 1 ou 3 après qu'un module situé dans le port 2 ou 4 ait été partiellement rempli, et qu'un MEMORY LOST ne se soit pas produit depuis lors. Si vous avez branché les deux modules en même temps, la première version est la seule à utiliser.

Si vous n'avez qu'un module de mémoire d'extension (ou une HP-41CX) et que vous n'avez pas de modules de mémoire d'extension, ou si vous n'avez qu'un module de mémoire d'extension branché dans le port 1 ou 3, vous pouvez utiliser la fonction incorporée VER du lecteur de carte.

Si vous envisagez d'ajouter une autre mémoire d'extension, vous pourriez vouloir cependant vous accoutumer à utiliser le programme "VER".

Avertissement : Avant d'utiliser l'une ou l'autre des versions de "VER" vérifiez les deux éléments suivants:
1) Il y a au moins un assignement de touche qui n'est pas un assignement de label global, ou
2) Il n'y aucune alarme du module horloge utilisée et il y a au moins un registre de programme libre. Pour vérifier le nombre de registres de programme libres, pressez GTO. 000 et regardez l'affichage 00 REG nn en mode PRGM. Le nombre **nn** est le nombre de registres de programme libres.

Liste du programme « VER »

```
01♦LBL "VER"
 02 CF 25
 03 RCLFLAG
04 "×i♦Ä×♦0€♦x̄"
 05 X<> \
 06 ENTER↑
 07 X<> c
 08 RCL 04
  09 "0♦. "
 10 ASTO 63
 11 REGMOVE
 12 "├*"
 13 191
 14 STO 10
 15 R↑
 16 STO 06
 17 "├♦♦♦ū "
 18 X<> ]
19 STO IND Y
 20 X<> [
 21 STO IND Z
  22 DSE 10
23 STO IND 10
  24 R↑
  25 RCL \
  26 RCLPT
  27 GETX
28 "PRESS R/S"
  29 AON
  30 VER
  31 AOFF
  32 X<>Y
  33 SEEKPT
  34 RDN
  35 SAVEX
  36 X<>Y
37 STO IND 10
  38 LASTX
  39 X=Y?
  40 STO 63
  41 DSE 10
42 STO IND 10
  43 R↑
  44 STO 12
  45 END

112 BYTES
```

Liste du programme « EXM »

```
01♦LBL "EXM"
 02 190
 03 CLA
 04 XTOA
 05 RDN
 06 ASTO b
 07 END

19 BYTES
```

Liste du programme « PFF »

```
01♦LBL "PFF"
02 "├      "
 03 7
 04 AROT
 05 RCL c
 06 RCL [
  07 "×iλ♦"
 08 ASTO c
 09 STO 00
 10 X<>Y
 11 STO c
 12 CLST
 13 EMDIR
 14 CLD
 15 END

41 BYTES
```

Aucune de ces conditions n'est difficile à vérifier, et l'une ou l'autre assurera que "VER" fonctionne correctement.

Pour utiliser "VER" pressez XEQ ALPHA V E R ALPHA et la demande "CARD" apparaîtra. En même temps, vous verrez l'annonciateur du mode ALPHA. Si l'annonciateur d'ALPHA n'est pas allumé, le programme "VER" n'est pas présent et vous avez accidentellement exécuté la fonction incorporée VER. Dans ce cas, n'insérez aucune carte !

Après avoir vérifié la dernière carte, pressez R/S ou la touche d'effacement pour effacer de l'affichage la demande "CARD". Vous verrez alors le message "PRESS R/S".

AVERTISSEMENT: N'oubliez pas de taper R/S pour redémarrer le programme "VER". Si vous ne redémarrez pas le programme, le dommage causé par le parasite VER ne sera pas réparé. Même pire, d'importants pointeurs de système seront détruits, en provoquant le blocage du clavier et un MEMORY LOST. C'est une caractéristique commune à ce type de programmation synthétique. Elle est puissante et utile mais tout à fait impardonnable si vous ne l'utilisez pas correctement.

Précautions:

1) si vous n'enlevez pas votre doigt assez rapidement de la touche R/S, la calculatrice s'arrêtera quelques secondes ou plus tout en essayant de calculer un numéro de ligne. Durant cette pause, le message "PRESS R/S" restera à l'affichage, et l'annonciateur de PRGM sera éteint, tout comme si rien ne s'était produit. Ne faites pas l'idiot, et ne pressez pas R/S encore. Le programme démarrera lui-même. Lorsque le programme finira vous verrez probablement quelques novas et autres caractères non standards à l'affichage.

2) "VER" ne peut pas être appelé d'une profondeur de plus d'un niveau de sous-programme (c'est à dire, lorsque plus d'un RTN est en attente).

Les codes-barres pour "VER" se trouvent à l'annexe D.

Les lignes synthétiques et leurs équivalents décimaux (version 1) :
Ligne 04 : 252, 1, 105, 0, 19, 240, 1, 137, 0, 48, 3, 0, 2
Ligne 05 : 206, 118 ; ligne 07 : 206, 125
Ligne 09 : 245, 16, 0, 46, 240, 191
Ligne 17 : 247, 127, 0, 0, 0, 22, 191, 255
Ligne 18 : 206, 119 ; ligne 20 : 206, 117
Ligne 25 : 144, 118 ; ligne 30 : 167, 133 (pas synthétique)

Version 2, différences :
Ligne 09 : 245, 16, 0, 62, 240, 191
Ligne 17 : 247, 127, 0, 0, 0, 7, 223, 255

Le concept utilisé dans le programme "VER" est simple. Le but de ce programme est de rappeler l'emplacement 1007 (voir la carte de la mémoire d'extension), exécuter la fonction VER du lecteur de cartes, puis rétablir l'emplacement 1007. Les lignes 02 et 03 obligent à un arrêt en cas d'erreur si les fonctions d'extension ne sont pas présentes. Vous pouvez effacer ces lignes si vous voulez utiliser "VER" avec une HP-41CX.

De façon que GETX puisse être utilisé pour rappeler l'emplacement 1007, "VER" altère de façon provisoire le second registre d'en-tête du fichier supérieur pour simuler un fichier de données de 4095 registre. Cette puissante technique, inventée par Clifford Stern, permet d'accéder à toutes les mémoires d'extension, sans normalisation, par SAVEX et GETX. La

fonction REGMOVE (ligne 11) sauve les deux registres d'en-tête du fichier original supérieur de façon qu'ils puissent être rétablis avant que le programme se termine.

Contrairement aux apparences, les registres 04, 06, 10, et 63 ne sont pas affectés par "VER". Du fait de la manipulation par le programme des pointeurs du système d'exploitation, les instructions telles que DSE 10 et STO 06 ont vraiment accès aux registres internes du système d'exploitation. La fonction ASTO 63 altère le registre inférieur de pointeur du module de fonctions d'extension mémoire pour placer le pointeur de fichier de travail à 01.

10E. Une solution pour le parasite PURFL

Le programme "PFF" présenté ici est fourni spécialement pour les propriétaires des modules de fonctions d'extension mémoire révision 1B. Cela vous permet de rattraper une utilisation éventuelle d'une fonction nécessitant un fichier de travail lorsqu'aucun fichier de travail n'existe. Cette situation peut se produire après l'exécution d'un PURFL, comme expliqué à la page 19. Ce qui se produit réellement c'est que le registre supérieur de la mémoire d'extension est recouvert par le code de partition, qui a été mentionné à la section C de ce chapitre. Ce code de partition erroné dit à la HP-41 que la mémoire d'extension est vide.

La solution est simple, Le programme "PFF" remplace simplement le nom de fichier qui appartient au registre supérieur de la mémoire d'extension, emplacement 191 (décimal) figure 10.1 page 179). Comme aucun autre registre de la mémoire d'extension n'est dérangé par le "parasite", aucune autre action n'est nécessaire pour rétablir la table de la mémoire d'extension.

Le programme "PFF" qui réalise ce résultat a été écrit par Clifford Stern. Il utilise les techniques synthétiques, puisque le registre affecté n'est pas un registre de données normalement accessible.

Parce que le parasite PURFL n'est présent que dans le module de fonctions d'extension mémoire de la révision 1B, les propriétaires de la révision 1C ou de la HP-41CX peuvent sauter cette section.

Instructions pour "PFF":

1. D'abord vérifiez que votre table de mémoire d'extension est vraiment vide. Si vous ne l'avez pas vidée intentionnellement, et que le MEMORY LOST ne s'est pas produit, alors vous savez que le registre supérieur de la mémoire d'extension a été changé. Le programme "PFF" réparera alors le dommage, dans la mesure où vous n'avez pas créé de nouveaux fichiers dans la mémoire d'extension. Une fois que vous avez créé un nouveau fichier, les anciens sont recouverts et le dommage ne peut être réparé.

2. Mettez le nom du premier fichier de la mémoire d'extension, jusqu'à sept caractères, dans le registre ALPHA. Si vous ne vous rappelez pas du nom, n'importe quelle chaîne fera l'affaire. Par exemple, vous pourriez le nommer "TOP". Ensuite, après que "PFF" ai rétabli votre table de mémoire d'extension, vous pouvez obtenir le fichier "TOP", vérifier son contenu pour établir son identité, puis utiliser "PFF" à nouveau pour lui donner le nom correct.

3. Exécutez "PFF". Le programme se termine par une instruction EMDIR, à la fois pour établir un fichier de travail et pour vous montrer exactement quels fichiers de la mémoire d'extension vous avez récupérés. Vous pouvez interrompre la table de la mémoire d'extension si vous le voulez.

ATTENTION : "PFF" peut être SST, mais n'abandonnez jamais "PFF" entre les lignes 08 et 12, sinon il est probable qu'un MEMORY LOST s'ensuivra. Dans des conditions normales d'utilisation, "PFF" est sans danger.

La liste du programme "PFF" est p. 128. Les codes-barres se trouvent à l'appendice D.

Les lignes synthétiques et leurs équivalents décimaux par octet :
La ligne 02 est : APPEND 6 espaces
Ligne 05 : 144, 125 ; ligne 06 : 144, 117
Ligne 07 : 245, 1, 105, 11, 242, 0
Ligne 08 : 154, 125 ; ligne 11 : 145, 125

10F. Exécution d'un programme à l'intérieur de la mémoire d'extension

Si un fichier de programme est contenu entièrement dans les 127 registres du module de fonctions d'extension mémoire, il est possible d'exécuter ce programme sans faire d'abord un GETP. Naturellement, les techniques de la programmation synthétique sont nécessaires, mais rien de trop bizarre. Tout ce qu'il y a à faire est d'altérer le pointeur de programme, deux octets d'un registre interne qui désignent la part de mémoire qui est affichée quand vous entrez en mode programme.

AVERTISSEMENT : Avant d'essayer d'exécuter un programme dans la mémoire d'extension, vérifiez si tous les GTO ou XEQ sont compilés. C'est à dire que vous devez vérifier que chaque instruction GTO ou XEQ du programme a été exécutée au moins une fois depuis la dernière fois ou le programme a été corrigé ou compacté. Ceci s'applique aux instructions locales GTO et XEQ (celles qui se réfèrent aux labels 00-99, A-J ou a-e). Si vous oubliez de le faire, l'exécution du programme dans la mémoire d'extension peut invalider la somme de contrôle, ce qui amènera GETP à donner un message CHKSUM ERR. Si vous ne vous souciez pas de pouvoir récupérer le programme de la mémoire d'extension, vous pouvez ignorer cet avertissement. En particulier, si vous n'envisagez d'utiliser ce programme que dans la mémoire d'extension, l'impossibilité d'utiliser GETP n'est pas importante. De plus le programme "RPF" présenté dans la section 10I peut être utilisé à la place de GETP en cas d'erreur de somme de contrôle. Si vous avez besoin de rappeler le programme souvent ou par programme, il s'avèrera beaucoup plus commode de vérifier si les GTO ou XEQ sont compilés avant de sauver le programme. Une digression au sujet des branchements compilés devrait clarifier les choses.

La première exécution d'une instruction locale GTO ou XEQ provoque le stockage de la distance et de la direction du branchement dans les instructions elles-même. La calculatrice se rappelle alors de l'emplacement du label la prochaine fois que le GTO ou XEQ est rencontré. C'est ce stockage d'information sur la distance dans les instructions qui invalide la somme de contrôle du fichier de programme. Si les GTO et XEQ sont compilés avant que le programme soit sauvé, leur contenu n'est pas modifié par l'exécution dans la mémoire d'extension et le somme de contrôle reste valable. Les instructions indirectes et globales (ALPHA) GTO et XEQ (apparaissant au CAT 1) ne se compilent pas et ne posent donc pas de problème. Toutes les distances de saut sont modifiées si vous corrigez le programme (par insertion ou effacement), la machine efface alors les informations, devenues inexactes, des GTO et XEQ. La même situation se produit au PACKING, à moins que le programme n'ai déja été packé. Pour compiler, suivez la procédure suivante :

1. Le programme doit être en mémoire principale. S'il est déja en mémoire d'extension, vous tapez simplement un GETP.
2. Ensuite GTO... Pour chaque ligne qui contient un GTO ou un XEQ local, faites : GTO. nnn (allez à la ligne qui contient le GTO ou le XEQ), SST en mode RUN (non programme) pour exécuter l'instruction, pressez et maintenez la touche SST jusqu'à ce que l'instruction apparaisse à l'affichage. Ensuite lâchez la touche SST. Lorsque l'instruction disparait, elle a été exécutée. Répétez l'opération jusqu'à ce que vous ayez exécuté tous les GTO et les XEQ locaux.
3. Maintenant vous pouvez exécuter SAVEP avec le nom du programme dans le registre ALPHA pour sauver le programme.

Le programme "EXM" utilise une instruction synthétique, ASTO b. Cette instruction est utilisée ici pour transférer un caractère du registre ALPHA dans les deux octets les plus à droite du registre **b** du système d'exploitation, emplacement du pointeur de programme.

Exemple "EXM" : Supposez que votre premier fichier dans la mémoire d'extention est le programme "JNX". Vous pouvez utiliser "EXM" pour exécuter "JNX" sans l'amener dans la mémoire principale. Chargez simplement les registres Y et X avec les deux entrées nécessaires (n et x), et pressez :

XEQ ALPHA E X M ALPHA

Le résultat apparaîtra en X quand "JNX" sera terminé. Le pointeur de programme restera dans le programme "JNX" à moins que vous n'exécutiez le Catalogue 1 ou à moins que vous fassiez GTO ou XEQ un label du Catalogue 1. Comme il est listé ici, "EXM" ne permet que l'exécution du premier fichier de la mémoire d'extension. Si, cependant, vous connaissez l'adresse absolue du second registre d'en-tête du fichier de programme que vous voulez exécuter, vous pouvez utiliser ce nombre à la place du nombre 190 (ligne 02) pour exécuter un programme différent. Mais, répétons-le, le fichier de programme doit résider complètement dans le module de fonctions d'extension mémoire. Il ne doit pas déborder dans les modules de mémoire d'extension.

Notez que les programmes qui contiennent des sauts en arrière (comme LBL 00 .... GTO 00) doivent nécessairement être compilés avant de les sauver en mémoire d'extension si vous voulez pouvoir les exécuter directement dans la mémoire d'extension.

Instructions pour "EXM" :
1. Vérifiez que le fichier de programme que vous voulez exécuter est le premier fichier de la table de la mémoire d'extension. Si ce n'est pas le cas, calculez l'emplacement du second registre d'en-tête du fichier. C'est 190 moins le nombre de registres utilisés par les fichiers précédents. Rappelez-vous que le nombre de registres utilisés par un fichier est de 2 plus grand que le nombre indiqué dans la table de la mémoire d'extension. Remplacez la ligne 02 du programme "EXM" par ce nombre calculé.
2. Vérifiez que le fichier de programme que vous voulez exécuter est contenu entièrement dans le module de fonctions d'extensions. Ajoutez le nombre de registres utilisés par tous les fichiers jusqu'à et y compris le fichier à exécuter. Vérifiez d'inclure les deux registres d'en-tête de chaque fichier qui ne sont pas inclus dans l'affichage de la table de mémoire d'extension. Le total ne doit pas dépasser 127 registres.
3. Comme discuté dans l'avertissement ci-dessus, tous les GTO et XEQ du programme sauvé doivent être compilés si vous espérez pouvoir utiliser GETP

pour récupérer le fichier de programme plus tard. En cas d'urgence, vous pouvez utiliser le programme "RPF" présenté dans la section 10I.
4. Chargez les registres X, Y, et Z avec toutes les entrées demandées par la fonction. Le registre ALPHA ne peut pas être utilisé pour une entrée, parce qu'il est effacé par "EXM".
5. Exécutez "EXM". La ligne 06 de "EXM" entraîne un saut immédiat à la première ligne d'un programme qui se trouve immédiatement sous la position de l'adresse absolue de registre désignée ligne 02.

ATTENTION : Ne pas utiliser "EXM" pour exécuter un programme qui contient une instruction PSIZE. Toutes les fois que PSIZE change le SIZE, il révise le pointeur de programme pour compenser le fait qque tous les programmes de la mémoire principale ont été déplacés. Même si PSIZE ne déplace pas votre programme dans la mémoire d'extension, PSIZE révisera le pointeur de programme comme si le programme avait bougé. Ceci provoque un saut involontaire. Le seul cas où ce saut ne se produit pas, est lorsque l'entrée PSIZE arrive à égaler le SIZE courant, de façon que le SIZE reste inchangé.

10G. Suspendre et réactiver les assignements de touches du mode USER

Comme part de sa compatibilité avec le fonctionnement de la HP-67/97, la HP-41 a 15 touches (deux rangées supérieures non shiftées plus une rangée supérieure shiftée) qui, lorsque'elles sont pressées en mode USER, trouveront et exécuteront le label local correspondant (A-J et a-e). Mais cette caractéristique entre en conflit avec tout assignements de label global. Combien de fois avez-vous voulu utiliser l'assignement automatique des labels locaux A-J et a-e, mais avez trouvé une fonction ou un assignement de label global sur votre route? Vous pressez LOG pour exécuter LBL D, mais à la place vous obtenez une autre fonction que vous avez assignée à cette touche. Est-ce que ce ne serait pas chouette s'il y avait une manière d'éliminer les assignements de touches conflictuels, puis de les reprendre plus tard ?

Une fois de plus, la programmation synthétique arrive à la rescousse. Un très court programme synthétique appelé "SK" écrit par Tapani Tarvainen, désactive tous les assignements de touches de label global et les assignements de fonction du mode USER. Pour suspendre ces assignements, pressez :

XEQ ALPHA S K ALPHA

Un programme appelé "RK", écrit également par Tapani Tarvainen, vous permet de réactiver les assignements de touches dormants. Lorsque vous exécutez "RK" un GETP sera réalisé sur un fichier de programme synthétique spécial. Ce fichier synthétique doit être d'abord créé dans la mémoire d'extension en utilisant le programme "IN" décrit ci-dessous. En fait, vous pourriez réactiver les assignements de touche en récupérant un quelconque programme de la mémoire d'extension à l'aide de GETP. Dans le processus de récupération du programme, la calculatrice réactivera toutes les assignements de touches dormants. Cette réactivation se produit toutes les fois qu'un programme est amené en mémoire principale, que ce soit à partir de cartes magnétiques, code barre, cassettes, ou de mémoire d'extension. L'avantage de l'emploi de "RK" est que le dernier programme du Catalogue 1 ne sera pas dérangé. Tout autre type de fonctionnement de GETP recouvrira le dernier programme du catalogue 1 (voir page 16). A moins que ce soit ce que vous voulez, vous devriez utiliser "RK" plutôt que GETP.

10H. Sauvegarde de l'état des assignements de touches dans la mémoire d'extension

Le lecteur de cartes magnétiques HP82104A a une fonction WSTS qui vous permet d'enregistrer les informations d'assignements de touches sur cartes magnétiques. Cela permet facilement de conserver plusieurs jeux d'assignements de fonctions (les assignements de labels globaux ne sont pas enregistrés). Vous mettez simplement en place et enregistrez chaque jeu d'assignements de fonctions, en construisant une "bibliothèque" d'assignements de touches. Ensuite lorsque vous voulez utiliser une configuration particulière d'assignements de touches, vous vous contentez de lire la carte magnétique correspondante.

Les techniques de la programmation synthétique vous permettent d'utiliser la mémoire d'extension exactement comme vous utiliseriez les cartes magnétiques pour stocker les assignements de touches. Les programmes "SAVEK" et "GETK", ont été écrits par Tapani Tarvainen et révisés par Clifford Stern (section du label 04 ajoutée). Ces programmes sauvent les informations d'assignements de touches dans la mémoire d'extension et les récupèrent sur demande. Contrairement aux versions précédentes, ils sont entièrement compatibles avec les alarmes du module horloge et autres mémoires tampon I/O. Ces programmes comprennent également les programmes "SK" et "RK".

Attention : Avant d'utiliser l'un ou l'autre programme "RK" ou "GETK" pour la première fois, il vous faut utiliser un programme d'initialisation appelé "IN" qui est listé à la page 198. Les deux programmes "RK" et "GETK" se terminent par l'instruction GETP, qui a pour effet de réactiver tout nouvel assignement de touches ou tout assignement dormant. Une méthode unique, inventée par Tapani Tarvainen, utilise un fichier de programme synthétique comme l'objet de l'instruction GETP. Cette procédure élimine le recouvrement normal du dernier programme dans la mémoire principale lorsque GETP est exécuté. Le fichier de programme synthétique a le nom " " (un simple espace) et une longueur de zéro octet. Le programme "IN", une création ésotérique de Clifford Stern, automatise la procédure de création de fichier de programme synthétique dans la mémoire d'extension.
Avertissement : avant d'exécuter "IN", lisez l'avertissement sous "VER" page 128. Ensuite, si une des deux conditions listées là est satisfaite, vous pouvez presser XEQ "IN" pour créer le fichier de programme synthétique demandé par "RK" et GETK". Une fois ceci réalisé, vous n'avez pas besoin d'utiliser "IN" à nouveau tant que le fichier synthétique restera dans la mémoire d'extension. Pour en être sur, exécutez la table de mémoire d'extension. Vous devez voir une entrée qui s'affiche comme " POO1".

Instructions pour "SAVEK" et GETK"

1. Il doit y avoir un END au-dessus de LBL "SAVEK" au catalogue 1. Ne pas observer cette contrainte provoquera un éventuel MEMORY LOST
2. Si vous n'avez pas déjà procédé de cette façon, exécutez le programme "IN" comme décrit dans les deux derniers paragraphes pour la mise en place du fichier de programme synthétique de zéro octet appelé " " dans la mémoire d'extension.
3. Pour sauver le jeu courant d'assignements de fonctions, placez un nom de fichier de au plus 7 caractères dans le registre ALPHA et exécutez "SAVEK". Si vous obtenez un message d'erreur NO ROOM à la ligne 43, c'est que vous n'avez pas laissé assez de place dans la mémoire d'extension pour contenir

Liste des programmes « SAVEK »/« GETK »/« RK »/« SK »

```
 01 RTN
 02◆LBL "SAVEK"
 03 RCL [
 04 XEQ 04
 05 193
 06 X>Y?
 07 RTN
 08 SF 10
 09◆LBL 01
 10 -
 11  E3
 12 /
 13 "0"
 14 RCL [
 15 XEQ 10
 16 SIGN
 17◆LBL 02
 18 RDN
 19 RCL IND Y
 20 ""
 21 X<> [
 22 "├*"
 23 X<> \
 24 X≠Y?
 25 GTO 03
 26 ARCL c
 27 X<> \
 28 STO IND Z
 29 FC? 10
 30 SAVEX
 31 ISG Z
 32 GTO 02
 33◆LBL 03
 34 X<> L
 35 X<> c
 36 R↑
 37 CLA
 38 STO [
 39 R↑
 40 INT
 41 FC?C 10
 42 RTN
 43 CRFLD
 44  E
 45 GTO 01
 46◆LBL 04
 47 RCL c
 48 "*"
 49 X<> [
 50 STO \
 51 ASHF
 52 RDN
 53 ALENG
 54 8
 55 Y↑X
 56 ATOX
 57 *
 58 512
 59 MOD
 60 ATOX
 61 +
 62 RTN
 63◆LBL 10
 64 RCL c
 65 "@×i▒x̄◆"
 66 ASTO c
 67 RTN
 68◆LBL "GETK"
 69  E
 70 SIZE?
 71 +
 72 PSIZE
 73 LASTX
 74 PSIZE
 75 X<>Y
 76 FLSIZE
 77 XEQ 10
 78 CLKEYS
 79 X<> c
 80 RDN
 81 +
 82 PSIZE
 83 RDN
 84 PSIZE
 85 X<> L
 86 XEQ 04
 87 RCL Y
 88 -
 89 192
 90 -
 91  E3
 92 ST/ Z
 93 X↑2
 94 /
 95 +
 96 X<>Y
 97 X<> c
 98 X<>Y
 99 REGMOVE
100 GETR
101 X<>Y
102 STO c
103◆LBL "RK"
104 " "
105 GETP
106 RTN
107◆LBL "SK"
108 .
109 STO '
110 STO e
111 END     212 BYTES
```

Liste du programme « IN »

```
     01◆LBL "IN"
     02 ENDIR
     03  E
04 " ,×i◆Â×◆0←◆x̄"
     05 CRFLD
     06 +
     07 RCL \
     08 X<> c
     09 RCL 04
     10 REGMOVE
     11 "├*"
     12 RDN
     13 RCL 12
     14 STO 06
     15 "├◆×iλ"
     16 RCL [
     17 STO 12
     18 X<>Y
     19 STO 01
     20 X<> ]
     21 STO 03
     22 RDN
     23 ENDIR
     24 CLD
     25 ST- T
     26 X<>Y
     27 STO 01
     28 X<> \
     29 STO 02
     30 CLA
     31 STO [
     32 R↑
     33 X>0?
     34 E↑X
     35 SEEKPTA
     36 "×"
     37 X>0?
     38 ASTO L
     39 ASTO X
     40 SAVEX
     41 X<> L
     42 STO 01
     43 R↑
     44 X<> c
     45 END
```

93 BYTES

les informations d'assignements de touches. Vous avez la possibilité de faire de la place dans la mémoire d'extension si vous voulez toujours sauver les assignements de touches. Après avoir fait de la place, reprenez les instructions au début avec le nom de fichier en ALPHA. Lorsque "SAVEK" se termine, le nombre en X indique la taille du nouveau fichier de données des assignements de touches.
4. Pour utiliser "GETK", chargez le registre ALPHA avec le nom d'un fichier de données d'assignements de touches que vous avez créé avec "SAVEK". Ensuite pressez XEQ "GETK". Si vous obtenez un message d'erreur NO ROOM à la ligne 72, il n'y a pas de registres de programme libres. "GETK" nécessite la présence initiale d'un registre libre, et le piège à erreur à la ligne 72 vérifie son existence.

Donc, NO ROOM à la ligne 72 indique que vous devez réduire la taille de 1 ou effacer un programme pour faire de la place. Si vous obtenez le message NO ROOM à la ligne 82, c'est qu'il n'y a pas assez de registres de programme libres pour contenir les assignements de touches du fichier de données en question. La différence entre le nombre en X et la SIZE courant est le manque de registre. Vous devez à nouveau réduire la SIZE ou effacer un programme pour éliminer ce défaut. Si l'un ou l'autre de ces arrêts dus à une erreur se produit, vous devez soit recharger ALPHA et XEQ "GETK" soit faire XEQ "RK" pour simplement réactiver les assignements de labels globaux et terminer.

Le programme "SAVEK" sauve les assignements de touches du catalogue 2 ou des fonctions du catalogue 3, mais il ne sauve pas les assignements des labels du catalogue 1. Le programme "GETK" récupère les informations des assignements de touches de fonctions dans le fichier de mémoire d'extension en question. Ces assignements de fonctions fusionnent avec tout assignement de label global préexistant. Les assignements de fonction antérieurs sont effacés. En cas de conflit, où si un label global est déjà assigné à une des touches utilisées dans le stockage du jeu d'assignements de fonctions, l'assignement du label global prendra le pas.

Si vous possédez un PPC ROM (voir annexe C), vous pouvez effacer la section LBL 04, lignes 46-62, et remplacer les instructions XEQ 04 lignes 04 et 86 par XROM "E?".

Au cas où ça vous surprendrait, le RTN Ligne 01 est une part nécessaire du programme "GETK". Si "SAVEK"/"GETK" est le dernier programme en mémoire principale (c'est à dire, s'il a un .END. comme sa dernière ligne) le GETP à la ligne 105 transfèrera le contrôle à la ligne 01.

Les codes barres de ce programme se trouvent à l'annexe D.

Les lignes synthétiques et leurs équivalents décimaux :
Ligne 03 : 144, 117 ; ligne 11 : 27, 19 ; ligne 13 : 241, 16
Ligne 14 : 144, 117 ; ligne 20 : 241, 240 ; ligne 21 : 206, 117
Ligne 23 : 206, 118 ; ligne 26 : 155, 125 ; ligne 27 : 206, 118
Ligne 35 : 206, 125 ; ligne 38 : 145, 117 ; ligne 44 : 27
Ligne 47 : 144, 125 ; ligne 49 : 206, 117 ; ligne 50 : 145, 118
Ligne 64 : 144, 125 ; ligne 65 : 246, 64, 1, 105, 12, 2, 0
Ligne 66 : 154, 125 ; ligne 69 : 27 ; ligne 79 : 206, 125
Ligne 91 : 27, 19 ; ligne 97 : 206, 125 ; ligne 97 : 206, 125 ; ligne 102 : 145, 125
Ligne 109 : 145, 122 ; ligne 110 : 145, 127

Les codes-barres de "IN" se trouvent à l'annexe D.

Les lignes synthétiques et leurs équivalents décimaux pour "IN":
Ligne 03 : 27
Ligne 04 : 254, 32, 44, 1, 105, 0, 19, 240, 1, 137, 0, 48, 3, 0, 2
Ligne 07 : 144, 118 ; ligne 08 : 206, 125
Ligne 15 : 247, 127, 0, 1, 105, 11, 223, 255
Ligne 16 : 144, 117 ; ligne 20 : 206, 119 ; ligne 28 : 206, 118
Ligne 31 : 145, 117 ; ligne 36 : 241, 1 ; ligne 44 : 206, 125

Si vous possédez une HP-41CX, les lignes 02 et 23 peuvent être remplacées par EMROOM. Comme pour "VER", en dépit des apparences, aucun registre de données numéroté n'est perturbé par "IN".

En dépit de sa brièveté, "IN" est un programme très sophistiqué. En fait, si vous pensez que vous êtes un expert en programmation synthétique, vous pouvez essayer de comprendre coment il marche. Clifford Stern est probablement le seul à connaitre toutes les astuces qu'il contient.

10I. Sauvegarde des fichiers de mémoire d'extension sur cartes magnétiques

Le chapitre 3 introduit les programmes "WAS"/"RAS" qui vous permettent de transférer les données des fichiers ASCII aux cartes magnétiques et à partir d'elles. Les programmes "WFL" et "RFL" présentés dans cette section, ont été écrits par Clifford Stern. Ils permettent à tous les types de fichiers d'être écrits sur cartes magnétiques, ou juste introduits dans des registres de données pour un autre stockage. De plus, le nombre minimum absolu de registres est utilisé. Sept octets de données sont sauvés par registre, plutôt que les 6 octets par registre ou un nombre inférieur que sauve "WAS". Un troisième programme "RPF", permet la récupération des programmes de mémoire d'extension qui ont des erreurs de somme de contrôle. Par exemple, une erreur de somme de contrôle peut résulter du fonctionnement du programme en mémoire d'extension (voir section 10F) si tous les GTO et les XEQ n'étaient pas compilés avant que le programme n'ai été sauvé.

Contraintes communes à "WFL", "RFL", et "RPF"

1. Il doit y avoir un END au-dessus du programme "WFL"/"RFL"/"RPF" du catalogue 1. Ne pas observer cette contrainte conduira à un éventuel MEMORY LOST.

2. Vérifier qu'au moins une des conditions enregistrée à la page 128 est satisfaite (pas d'alarmes et un registre libre, ou au moins un assignement de touches qui ne soit pas un label).

3. Le registre ALPHA doit contenir un nom de fichier au début de chacun de ces programmes. La séquence ALENG, 1/X (lignes 14 et 15) ne peut être effacée du programme, parce que le nom de fichier est nécessaire au cours de l'exécution du programme. L'instruction POSA à la ligne 17 entraîne un arrêt du à l'erreur à la ligne 19 si le registre ALPHA comporte une virgule. Les virgules ne sont pas autorisées dans le nom de fichier, parce qu'une virgule est interprétée par la calculatrice comme un séparateur de nom. La virgule et tous les caractères qui la suivent sont ignorés de toutes les fonctions d'extension sauf SAVEP. Si une virgule était présente, le piège à erreur ALENG serait sans effet.

4. Ces programmes peuvent être appelés à partir d'un autre programme, mais pas d'un sous-programme. En d'autres mots, aucun RTN ne peut être en attente lorsqu'un de ces programmes est appelé.

Instructions pour "WFL"

1. Levez le drapeau 14 si vous avez l'intention d'écrire des données sur

cartes protégées.
2. Deux modes de fonctionnement sont disponibles pour "WFL". Le premier mode, obtenu en baissant le drapeau 01, écrit les contenus entiers du fichier. Le second mode assure un stockage plus lent mais plus économique des fichiers ASCII qui ne sont que partiellement remplis.

Il compte d'abord le nombre de caractères dans le fichier, ensuite il transfére seulement les registres qui sont réellement en usage vers les registres de données. Pour activer ce second mode, levez le drapeau 01. Après avoir vérifié l'état du drapeau 01, mettez le nom du fichier dans le registre ALPHA, et exécutez "WFL".
3. Si l'instruction PSIZE à la ligne 49 donne un message d'erreur NO ROOM, il vous faudra effacer quelques programmes ou assignements de touches pour faire assez de place pour les contenus des fichiers. Le nombre en X indique la taille demandée.
4. Lorsque une demande RDY 01 OF nn apparaît, vous pouvez soit insérer une carte magnétique pour enregistrer les données de fichiers ou pressez R/S deux fois pour éviter l'écriture des cartes magnétiques. Ne vous contentez pas de presser la touche d'effacement, ou votre fichier, qui a été temporairement changé pour un fichier de données, ne sera pas rétabli à son type de fichier original. Ne changez pas le contenu de la pile avant le redémarrage de "WFL". Un MEMORY LOST est le résultat probable. Si le lecteur de cartes n'est pas présent, la demande RDY 01 OF nn n'apparaîtra pas du tout. Les contenus seront simplement transférés aux registres de données.
5. Si vous omettez de lever le drapeau 14 et que vous voulez toujours écrire les données sur cartes protégées, pressez R/S deux fois pour éviter la demande RDY 01 OF nn. Lorsque le programme se termine, vous pouvez faire SF 14 et exécuter WDTA à partir du clavier pour écrire les cartes.
6. Quand la dernière carte magnétique est insérée, ou après que vous ayez pressé R/S deux fois, le programme se terminera par une séquence d'instructions qui laisse le nouveau SIZE en X. Ce nombre, qui représente la taille minimum de fichier demandée, doit être écrit sur les cartes. Il peut être nécessaire plus tard pour "RFL".

ATTENTION : Si votre but est de transférer immédiatement les données de fichier dans la mémoire d'extension plutôt que de les enregistrer sur cartes magnétiques, vous devez faire très attention de ne pas les perturber avant d'utiliser "RFL". Les données sont sous une forme "non normalisée" volatile, (voir page 127). Tout RCL, VIEW, ou opération similaire va altérer les données. Vous ne devez pas essayer de déplacer les données, ni même de les regarder, jusqu'à ce qu'elles aient été réécrites dans la mémoire d'extension. C'est le prix à payer pour l'efficacité du stockage de registre à registre du "WFL". Le format de données est le même que celui utilisé dans la mémoire d'extension, où toutes les données sont également non normalisées. Si vous rappelez ou visualisez accidentellement un registre de données écrit par "WFL", vous aurez à exécuter "WFL" à nouveau pour rétablir les bonne données avant d'utiliser "RFL".

Instructions pour "RFL"

1. Comme pour "WFL", deux modes de fonctionnement sont disponibles pour "RFL". Le mode est choisi par l'état du drapeau 01. Selon la quantité d'espace disponible dans la calculatrice, le premier mode (drapeau 01 baissé) peut fonctionner sans se préoccuper de savoir si le drapeau était levé pour "WFL". Pour le trouver, baissez le drapeau 01, mettez le nom de fichier dans le registre ALPHA, et exécutez "RFL".
2. Si un message d'erreur NO ROOM apparait à la ligne 49, et que le drapeau

01 était baissé lorsque vous avez utilisé "WFL" pour écrire les cartes, il vous faudra effacer quelques programmes pour faire de la place. Le nombre en X est le SIZE demandé. Si vous obtenez NO ROOM mais que le drapeau 01 était levé lorsque vous avez utilisé "WFL" pour écrire les cartes, alors vous avez une autre option.

Vous pouvez lever le drapeau 01 pour indiquer que le SIZE n'a pas besoin d'être augmenté au FLSIZE. Toutes les fois que vous utilisez "RFL" avec le drapeau 01 levé, vous devez redimensionner au nombre de registres écrits (c'est le même nombre que vous avez écrit sur les cartes). Ce drapeau option 01 peut aussi être utilisé pour lire les cartes de fichiers de données dans un plus grand fichier de données lorsque le SIZE ne peut être placé au nouveau FLSIZE. Quelque soit l'action que vous avez en réponse au message d'erreur NO ROOM , vous devez recharger le registre ALPHA et exécuter "RFL" à nouveau.

3. Si un lecteur de cartes est présent, la demande CARD apparaîtra. A ce point vous pouvez insérer les cartes de données que vous avez faites avec "WFL". Si les données sont déjà dans les registres, vous pouvez presser R/S deux fois pour éviter le traitement de lecture des cartes.

Comme pour "WFL", ne pressez pas la touche d'effacement en réponse à la demande CARD, ou bien le fichier, qui a été changé temporairement pour un fichier de données, ne sera pas rétabli à son type original. Aussi, ne perturbez pas la pile avant de redémarrer le programme. Si vous le faites, MEMORY LOST est le résultat probable. (S'il n'y a pas de lecteur de carte, la lecture des cartes est automatiquement évitée).

4. Le programme prendra les information des registres de données et les transférera dans le fichier de mémoire d'extension en question. Il n'est pas important que le fichier soit un fichier de programme, de données, ou de texte ASCII.

Une application utile de "WFL" et RFL" est de minimiser le nombre de registres nécessaires pour un fichier donné, auquel vous n'ajouterez pas d'information trop souvent. Vous pouvez créer un grand fichier ASCII, le remplir d'informations désirées, ensuite utilisez "WFL" avec le drapeau 01 levé pour écrire les enregistrements dans les registres de données et peut être dans les cartes magnétiques. Notez le SIZE résultant (qui apparaît en X à la fin de "WFL"). Purgez le fichier ASCII et créez un nouveau fichier ASCII du même nom avec un FLSIZE égal à la taille de "WFL". Puis exécutez "RFL", en pressant R/S deux fois à la demande CARD, pour relire l'information dans les registres de données dans le nouveau fichier ASCII, qui est la bonne taille des données.

Une autre application de "WFL"/"RFL" est de traiter le problème d'un fichier ASCII qui a dépassé sa taille de fichier (FLSIZE) originale. Baissez simplementle drapeau 01, placez le nom du fichier en ALPHA et exécutez "WFL" pour écrire tout le fichier vers les registres de données (et les cartes si vous le désirez). Puis purgez le fichier, créez un plus grand fichier du même nom, levez le drapeau 01, placez le nom du fichier en ALPHA et exécutez "RFL". Tant que vous n'aurez pas perturbé les registres de données dés l'exécution de "WFL", vous pouvez en toute sécurité presser deux fois R/S à la demande CARD pour l'éviter.

Une troisième application de "WFL" et RFL" est un enregistrement à un pas d'un fichier de données, sans aucun besoin d'une instruction GETR. Lorsqu'il est utilisé à la place de GETR, "WFL" élimine le besoin de redimensionner, puisque le programme le fait automatiquement. De plus, le pointeur de registre du fichier de données sera rétabli à sa valeur originale. C'est un légère amélioration par rapport à GETR. Vous pouvez aussi utiliser "WFL" pour enregistrer un fichier de programme directement à partir de la mémoire d'extension, si çà vous est égal que l'information

enregistrée soit dans un format qui ne peut être utilisé que par "RFL". Ceci pourrait être le cas, par exemple, si vous vouliez enregistrer un programme que vous ne voulez exécuter qu'en mémoire d'extension.

Le programme "RPF" récupère un programme à partir de la mémoire d'extension lorsqu'une erreur de somme de contrôle existe. Ceci peut être reconnu par le message CHKSUM ERR lorsque vous essayez d'exécuter un GETP ou un GETSUB. S'il n'y a pas d'erreur de somme de contrôle, GETP ou GETSUB est de loin préférable à "RPF", parce que "RPF" a l'effet secondaire de changer le fichier de programme récupéré en un fichier de données. Si vous soupçonnez un réel dommage pour ce programme, utilisez "RPF" en dernier ressort, seulement si le programme n'est pas disponible sur cartes magnétiques, cassette, ou codes barres. Naturellement si le dommage n'est dû qu'au fonctionnement du programme dans la mémoire d'extension, "RPF" récupérera une copie "propre" du programme.

La procédure utilisée pour "RPF" est un peu inhabituelle, et les opérations manuelles exigées empêchent d'utiliser "RPF" comme sous-programme. Suivez soigneusement et précisément les instructions.

Instructions pour "RPF"

1. GTO ..
2. Entrez en mode programme. Vérifiez s'il y a au moins un registre de libre. Puis exécutez SST pour obtenir le .END. à l'affichage, et insérez une instruction quelconque (ENTER! ira très bien), puis effacez cette instruction.
3. Placez le nom du fichier de programme endommagé dans le registre ALPHA et exécutez "RPF". Si vous obtenez un message d'erreur NO ROOM à la ligne 47, vous devrez effacer quelques programmes de la mémoire principale pour faire de la place pour le programme à récupérer. Ensuite recommencez au pas 1.
4. PACK (pressez XEQ ALPHA P A C K ALPHA, et non pas GTO..)
5. A la fin de "RPF", le SIZE sera de 000 et le fichier de programme sera changé en un fichier de données.
6. Vérifiez la copie du programme récupéré pour exactitude, au cas où autre chose que la compilation des GTO/XEQ ait entraîné le CHKSUM ERR.
7. Si le programme récupéré avait une longueur inférieure à 35 octets, il se peut qu'il ne soit pas possible d'utiliser "RPF" à nouveau sur le même fichier (qui est maintenant un fichier de données). Cette condition ne se produira que si le nombre d'octets modulo 7 dépasse le FLSIZE. Il est peu probable que vous rencontriez ce problème, parce que tout programme intéressant à exécuter en mémoire d'extension doit être beaucoup plus long que 34 octets.

Note: Si vous voulez supprimer la partie "RPF" de ce programme "WFL"/ RFL"/ "RPF", effacez les lignes 149-232, 122-123, 11-12, et 01-04. Ceci réduit le nombre d'octets pour donner un programme de lecture et d'écriture de fichiers de 265 octets. Vous pouvez aussi, si vous avez un PPC ROM (voir annexe C), remplacer les lignes 153-166 du programme par XROM "E?".

Les codes barre de ce programme se trouvent à l'annexe D.
Notez que la ligne 21 est "ajoutez 6 espaces".

Les lignes synthétiques et leurs équivalents décimaux par octet :
Ligne 24 : 206, 117
Ligne 48 : 252, 1, 105, 0, 19, 240, 1, 137, 0, 48, 3, 0, 2
Ligne 49 : 206, 118 ; ligne 50 : 206, 125 ; ligne 52 : 206, 117

Liste des programmes « WFL »/« RFL »/« RPF »

```
01♦LBL "RPF"
 02 CF 01
 03 SF 05
 04 GTO 02

05♦LBL "WFL"
 06 CF 06
 07 GTO 01

08♦LBL "RFL"
 09 SF 06

 10♦LBL 01
 11 CF 05

 12♦LBL 02
 13 CF 25
 14 ALENG
 15 1/X
 16 44
 17 POSA
 18 CHS
 19 LN
 20 FLSIZE
21 "⊢      "
 22 7
 23 AROT
 24 X<> [
 25 X<>Y
 26 FC? 01
 27 GTO 04
 28 SF 25
 29 FS? 06
 30 GTO 04
 31 CLX
 32 SEEKPT

 33♦LBL 03
 34 CLA
 35 GETREC
 36 ALENG
 37 +
 38 FS? 25
 39 GTO 03
 40 RCLPT
 41 +
 42 8
 43 +
 44 7
 45 /

 46♦LBL 04
 47 PSIZE
48 "×i◆à×◆0◆◆x̄"
 49 X<> \
 50 X<> c
 51 STO 10
 52 X<> [
 53 REGMOVE
 54 RCL c
 55 "⊢*"
 56 STO 06
  57 "⊢◆×iλ"
 58 CLX
 59 STO 07
 60 RCL \
 61 R↑
 62 CF 07
 63 X=Y?
 64 SF 07
 65 X<> [
 66 STO 12
 67 STO 01
 68 X<> ]
 69 STO 03
 70 2
 71 CHS
 72 LASTX
 73 FS? 07
 74 GTO 06
 75 .
76♦LBL 05
 77 STO ]
 78 LASTX
 79 STO 01
 80 R↑
 81 FLSIZE
 82 ST+ Y
 83 2
 84 ST+ Z
 85 CLX
 86 STO ]
 87 RCL c
 88 STO 01
 89 R↑
 90 SEEKPTA
 91 RCL [
 92 GETX
 93 X≠Y?
 94 GTO 05
 95 X<> \
 96 STO [
 97 RCLPT
 98 GETX

 99♦LBL 06
100 STO ]
101 "⊢◆"
102  E20
103 STO ↑
104  E
105 CHS
106 AROT
107 R↑
108 SEEKPT
109 R↑
110 .
111 X<> ]
112 FC? 07
113 SAVEX
114 FS? 07
115 SIGN
116 RDN
117 LASTX
118 STO 01
119 RCL a
120 X<> c
121 SF 25
122 FS? 05
123 GTO 07
124 TONE 8
125 FS? 06
126 RDTA
127 FS? 06
128 SAVER
129 FC? 06
130 GETR
131 FC? 06
132 WDTA
133 CF 25
134 STO c
135 STO 01
136 CLX
137 STO \
138 R↑
139 SEEKPTA
140 R↑
141 FC? 07
142 SAVEX
143 FC? 07
144 LASTX
145 STO 01
146 X<> a
147 STO c
148 SIZE?
149 RTN

150♦LBL 07
151 RCLPTA
152 STO 00
153 RCL c
154 "*"
155 X<> [
156 STO \
157 ASHF
158 ALENG
159 8
160 Y↑X
161 ATOX
162 *
163 512
164 MOD
165 ATOX
166 +
167 ENTER↑
168 "×i"
169 16
170 *
171 2
172 +
173 ENTER↑
174 XEQ 10
175 RCL 00
176 SIGN
177 CLX
178 XTOA
179 SEEKPT
180 PSIZE
181 X<> [
182 X<> c
183 FLSIZE
 184 "◆-"
185 STO ]
186 X<>Y
187 STO \
188 R↑
189 X<> [
190 STO 00

191♦LBL 08
 192 GETX
193 STO IND ]
 194 DSE ]
 195 GTO 08
196 X<> [
197 RCL \
198 LASTX
199 7
200 MOD
201 SEEKPT
202 CF 25
203 LASTX
204 -
205 AROT
206 CHS

207♦LBL 09
208 "⊢◆"
209 DSE X
210 GTO 09
211 X<>Y
212 X<> [
213 STO 01
214 2
215 CHS
216 AROT
217 R↑
218 ENTER↑
219 XEQ 10
220 X<> [
221 X<> c
222 BEEP
223 STOP

224♦LBL 10
225 256
226 MOD
227 X<>Y
228 LASTX
229 /
230 XTOA
231 RDN
232 XTOA
233 END

419 BYTES
```

Ligne 54 : 144, 125
Ligne 57 : 247, 127, 0, 1, 105, 11, 223, 255
Ligne 60 : 144, 118 ; ligne 65 : 206, 117 ; ligne 68 : 206, 119
Ligne 77 : 145, 119 ; ligne 86 : 145, 119 ; ligne 87 : 144, 125
Ligne 91 : 144, 117 ; ligne 95 : 206, 118 ; ligne 96 : 145, 117
Ligne 100 : 145, 119 ; ligne 101 : 242, 127, 0
Ligne 102 : 27, 18, 16 ; ligne 103 : 145, 120 ; ligne 104 : 27
Ligne 111 : 206, 119 ; ligne 119 : 144, 123
Ligne 120 : 206, 125 ; ligne 134 : 145, 125
Ligne 137 : 145, 118 ; ligne 146 : 206, 123
Ligne 147 : 145, 125 ; ligne 153 : 144, 125
Ligne 155 : 206, 117 ; ligne 156 : 145, 118
Ligne 168 : 242, 1, 105 ; ligne 181 : 206, 117
Ligne 182 : 206, 125 ; ligne 184 : 243, 192, 0, 45
Ligne 185 : 145, 119 ; ligne 187 : 145, 118
Ligne 189 : 206, 117 ; ligne 193 : 145, 247
Ligne 194 : 151, 119 ; ligne 196 : 206, 117
Ligne 197 : 144, 118 ; ligne 208 : 242, 127, 0
Ligne 212 : 206, 117 ; ligne 220 : 206, 117
Ligne 221 : 206, 125

10J. Assignements aux touches des fonctions synthétiques

Si vous faites de la programmation synthétique, il est tout à fait utile d'assigner quelques fonctions synthétiques de deux octets utilisées fréquemment. Le chapitre 4 de "La programmation synthétique, c'est facile !" contient deux des programmes les plus efficaces pour faire des assignements synthétiques de touches.

Le programme "ASG" (assign) présenté ici a été conçu par Tapani Tarvainen et optimisé par Tapani et Gerard Westen. Ce programme représente une étape décisive dans la programmation synthétique. Les programmes synthétiques précédents d'assignements de touches demandaient à l'utilisateur de spécifier la fonction à assigner en termes de ses deux équivalents décimaux par octet. "ASG" vous permet tout simplement d'épeler la fonction à assigner.

AVERTISSEMENT : un END doit précéder LBL "ASG" dans le catalogue 1. Si vous avez un "XF" comme premier programme du catalogue 1, c'est déjà réglé. Si vous exécutez accidentellement "ASG" quand il s'agit du premier programme du catalogue 1, le blocage du clavier est probable et un MEMORY LOST est possible.

"ASG" Exemple 1 :

Supposez que vous voulez assigner RCL b à une touche. Exécutez d'abord "ASG" (pressez XEQ ALPHA A S G ALPHA). Le message "ASN" apparaîtra à l'affichage, exactement comme pour la vraie fonction ASN, excepté que l'annonciateur du mode ALPHA sera allumé. Maintenant vous remplissez le nom de la fonction à assigner, dans ce cas RCL b. La calculatrice est déjà en mode ALPHA, aussi il vous suffit de presser :

R C L (space) Shift b

Ensuite pressez R/S pour redémarrer le programme. Attendez environ une seconde, puis pressez la touche à laquelle vous voulez assigner RCL b. Pour assigner RCL b à un emplacement shifté, pressez la touche shift, attendez que le signe moins apparaisse (indiquant un emplacement shifté), puis

pressez la touche à laquelle vous voulez que l'assignement soit fait.

Une fois que vous avez entré le nom de fonction et pressé la touche à laquelle l'assignement doit être fait, il vous suffit seulement d'attendre que le programme "ASG" termine son travail. La similitude entre la procédure et la fonction ASN incorporée est frappante. Le point d'entrée "PASG" (programmable assign) fournit une capacité d'assignement synthétique de touches similaire à PASN. Epelez la fonction synthétique en ALPHA, placez le code rang-colonne en X, et exécutez "PASG". Le code de touche est le même que vous utiliseriez pour PASN.

La partie "PASG" de "ASG" accomplit les propriété étonnante du décodage du nom de fonction dans ses équivalents décimaux. Ceci est fait en deux pas. Dans l'exemple du RCL b, "PASG" assigne d'abord la fonction RCL et extrait le code décimal des registres du système d'exploitation, puis le programme décode le suffix b en son équivalent décimal. Ces deux valeurs sont utilisées comme entrée pour un autre programme standard d'assignement de touches, "MKX", une création aussi de Tarvainen optimisée par Westen.

Pour les novices aventureux :

Si vous n'êtes pas familier avec la programmation synthétique et si vous avez été dérouté par les listages de programmes synthétiques, il se peut que vous vouliez utiliser "ASG" pour créer quelques instructions synthétiques. Quelques points de base vous aideront à éviter quelques uns des pièges les plus simples. D'abord et en tout premier lieu, n'altérez pas les contenus du registre c à moins de savoir exactement ce que vous faites.

Le résultat probable est un MEMORY LOST. Deuxièmement, faites attention que des lignes synthétiques dans les listages de l'imprimante apparaissent différemment qu'à l'affichage. Les plus remarquables sont les lignes de texte, dans lesquelles les caractères avec des codes 128-255 disparaissent, et les instructions qui ont accès aux quelques registres du système d'exploitation. L'équivalence est :

Affichage	listage d'imprimante
STO M	STO [
STO N	STO \
STO O	STO]
STO P	STO ↑
STO Q	STO _
STO ▸	STO ▸

Instructions pour "ASG"

1. Vérifiez s'il y a un END au-dessus de LBL "ASG" du catalogue 1, et qu'il n'y ait pas de LBL "ANUM" au catalogue 1. Ne pas observer ces restrictions entraînera un MEMORY LOST.
2. Exécutez "ASG". La demande ASN apparaîtra, et l'annonciateur du mode ALPHA sera allumé.
3. A l'aide des touches en mode ALPHA, épeler la fonction à assigner. Si le suffixe est un caractère synthétique, vous pouvez épeler à la place un nombre décimal de 0 à 255. Si vous le désirez, le préfixe peut aussi être un nombre décimal. Pour des fonctions indirectes comme GTO IND X, vous n'avez pas vraiment besoin d'épeler "IND", pourvu qu'il y ai 2 espaces entre GTO et X, la fonction GTO IND X sera assignée.
4. Pressez R/S pour redémarrer le programme.
5. Attendez une demi-seconde et pressez la touche à laquelle la fonction doit être assignée. Pour un assignement shifté pressez la touche shift, attendez un moment jusqu'à ce qu'apparaisse à l'affichage "-", puis pressez

la touche en question. Il n'est pas nécessaire de presser à nouveau R/S.
6. Le programme se mettra à faire des assignements synthétiques de fonctions. Si par hasard un arrêt provoqué par une erreur se produit, n'essayez surtout pas de redémarrer le programme. Commencez plutôt par réexécuter le pas 2 ci-dessus. Si l'erreur était NO ROOM à la ligne 50, réduisez la taille ou effacez un programme.
7. Pour faire un autre assignement, exécutez à nouveau "ASG".

Instructions pour "PASG"
1. Vérifiez qu'il y ait un END quelque part au-dessus du LBL "PASG" du catalogue 1, et qu'il n'y ait pas de LBL "ANUM" au catalogue 1.
La pénalité si on omet d'observer ces restrictions est un MEMORY LOST.
2. Chargez le registre ALPHA avec une chaîne qui épelle la fonction à assigner. Voir article 3 dans les instructions de "ASG" pour une explication des divers types de chaînes qui sont permises.
3. Mettez le code de touche colonne/rangée en X. "PASG" fonctionne à cet égard exactement comme PASN.
4. Exécutez "PASG". Le programme réalisera l'assignement synthétique. Comme pour "ASG" n'essayez pas de redémarrer après que se produise un arrêt dû à une erreur. Redémarrez avec le pas 2.
5. Pour effectuer un autre assignement, chargez ALPHA en X et exécutez "PASG" à nouveau.

Instructions pour "MKX" :
1.Mettez le code préfixe décimal en Z, le code suffixe en Y, et le code de touche en X.
2. Exécutez "MKX" pour faire l'assignement en question.

Précautions générales pour "ASG", "PASG", et "MKX"
1. N'interrompez pas ces programmes ou ne faites pas SST. Si vous interrompez accidentellement le programme entre les lignes 158 et 161 ou si vous faites SST, il vous faudra réexécuter le programme. Si vous interrompez le programme après la ligne 176, vous devez redémarrer le programme pour éviter un éventuel MEMORY LOST.
2. Vérifiez qu'il n'ait pas de label global qui ait la même fonction qu'une fonction que vous voulez assigner. Par exemple, si vous avez un LBL "STO" au catalogue 1, "ASG" et "PASG" ne pourront pas assigner une instruction STO.
3. **Avertissement** : vérifiez qu'il n'y ait pas de label global "ANUM" au catalogue 1. Si vous avez un LBL "ANUM" au catalogue 1, la mémoire sera complètement effacée.
Les programmes "ASG", "PASG", et "MKX" sont entièrement compatibles avec les alarmes du module horloge et autres mémoires tampon I/O. "ASG" et "PASG" vous permettent également d'assigner les labels du catalogue 1 et les fonctions non synthétiques, de même que les fonctions synthétiques. En fait, "ASG" et "PASG" sont essentiellement des remplaçants directs des fonctions ASN et PASN. Ils accepteront toute entrée qu'acceptent ASN ou PASN, plus un beaucoup plus grand nombre qui correspond à des fonctions synthétiques.

Exemples de "ASG" et "PASG"
La liste suivante montre des assignements de touches typiques, à la fois synthétiques et non-synthétiques, et les entrées ALPHA de "ASG"/"PASG" nésessaires pour les obtenir. Plusieurs variations sur les entrées ALPHA sont d'ordinaire possibles, comme la liste le montre. Toute les fonctions qui ne montrent que des entrées décimales sont plus facilement assignées

par "MKX" en plaçant les entrées décimales dans la pile.

Fonctions	Entrées ALPHA
"ASG"	"ASG" tout label global peut être assigné à l'aide de "ASG" ou "PASG"
"VER"	"VER"
BST	"BST"
SIGN	"SIGN"
SF 14	"SF 14" ou "168 14"
STO N	"STO N", "STO 118", "145 N", ou "145 118"
X<>M	"X<>M", "X<>117", ou "206 117"
GTO IND X	"GTO IND X", "GTO X" (notez les 2 espaces), "GTO I 115", "GTO 243", "208 243", etc
RCL IND X	"RCL IND X", "RCL X" (2 espaces), "RCL I X", "RCL 243", "145 243", etc
XROM 29,08	"XROM 29,08" ou "X 29,08" (=PRA)
TONE 10	"TONE 10" ou "159 10"
TONE 89	"TONE 89" ou "159 89"
FIX 10	"FIX 10" ou "156 10"
XROM 28,35	"XROM 28,35 ou "X 28,35" (=OUTA)

fonctions synthétiques plus fantaisistes :

GTO.000	"199 133" (fonctionne en mode PRGM, seulement lorsque le lecteur de cartes est connecté)
eGOBEEP	"4 167" ou "0 167" (donne des fonctions de stokage de masse ou d'imprimante, expérimentez en mode PRGM)
Q-Loader	"27 0" (expérimentez en mode PRGM)
voleur d'octets	"247 63" (pas pour les novices, peut donner un MEMORY LOST s'il est inséré au-dessus d'un END)

Le programme "ASG" est peut-être le programme synthétique le plus avancé qui ai jamais été écrit, dans lequel il est fait usage d'une grande variété de techniques synthétiques pour assurer un très haut degré de commodité à l'utilisateur. J'espère que vous l'utiliserez avec joie.

Les codes-barres de ce programme peuvent se trouver à l'annexe D. La ligne 156 est une ligne de texte "ANUM", et non une instruction ANUM.

Les lignes Synthétiques et leurs équivalents décimaux par octet :
Ligne 04 : 242, 132, 128 ; ligne 05 : 154, 126
Ligne 55 : 242, 127, 0 ; ligne 57 : 243, 127, 64, 48
Ligne 122 : 206, 126 ; ligne 126 : 144, 121
Ligne 127 : 145, 120 ; ligne 130 : 206, 126
Ligne 158 : 144, 122 ; ligne 160 : 145, 120
Ligne 161 : 243, 127, 166, 66 ; ligne 162 : 154, 118
Ligne 163 : 155, 118 ; ligne 168 : 144 ; 125
Ligne 172 : 144, 117 ; ligne 173 : 144, 125
Ligne 174 : 246, 64, 1, 105, 11, 2, 0 ; ligne 175 : 154, 125
Ligne 180 : 145, 118 ; ligne 181 : 206, 117
Ligne 182 : 145, 119 ; ligne 183 : 154, 117
Ligne 185 : 206, 118 ; ligne 188 : 206, 117
Ligne 192 : 145, 118 ; ligne 193 : 245, 127, 132, 132, 132, 240
Ligne 194 : 206, 117 ; ligne 195 : 145, 119
Ligne 196 : 155, 125 ; ligne 197 : 206, 119
Ligne 205 : 145, 125

Pour des opérations plus rapides avec une HP-41CX ou une module

Liste des programmes « ASG »/« PASG »/« MKX »

```
01♦LBL "ASG"
 02 RCLFLAG
 03 SIGN
   04 ""
 05 ASTO d
 06 "ASN "
 07 STOP
 08 CF 21
 09 "⊢ "

 10♦LBL 01
 11 31
 12 AVIEW
 13 GETKEY
 14 X=0?
 15 GTO 01
 16 X≠Y?
 17 GTO 02
 18 "⊢-"
 19 FS?C 03
 20 GTO 01
 21 2
 22 CHS
 23 AROT
 24 ATOX
 25 ATOX
 26 SF 03
 27 GTO 01

 28♦LBL 02
 29 ARCL X
 30 AVIEW
 31 FC? 03
 32 CHS
 33 X>0?
 34 XTOA
 35 LASTX
 36 STOFLAG
 37 ATOX
 38 ATOX
 39 LN
 40 CHS
 41 AROT
 42 ASHF
 43 R↑

44♦LBL "PASG"
 45 AOFF
 46 32
 47 POSA
 48 X>0?
 49 GTO 03
 50 RDN
 51 PASN
 52 CLD
 53 RTN

 54♦LBL 03
 55 "⊢♦"
 56 AROT
 57 "⊢00"
 58 ATOX
 59 POSA
 60 ISG X
 61 AON
 62 AROT
 63 44
 64 POSA
 65 ISG X
 66 GTO 04
 67 AROT
 68 R↑
 69 ANUM
 70 ASHF
 71 ANUM
 72 640
 73 +
 74 64
 75 *
 76 +
 77 RCL X
 78 256
 79 ST/ Z
 80 MOD
 81 GTO 06

 82♦LBL 04
 83 R↑
 84 ANUM
 85 ATOX
 86 84
 87 -
 88 X(=0?
 89 GTO 04
 90 CHS
 91 7
 92 +
 93 X>0?
 94 GTO 05
 95 CHS
 96 31
 97 MOD
 98 2

 99♦LBL 04
100 X<0?
101 9
102 X>0?
103 +
104 X>0?
105 3
106 X>0?
107 +

108♦LBL 05
109 17
110 +
111 X>0?
112 95
113 X>0?
114 +
115 X>0?
116 X<>Y
117 CLX
118 POSA
119 X>0?
120 AROT
121 CLX
122 X<> d
123 R↑
124 SF 25
125 PASN
126 RCL _
127 STO ↑
128 RDN
129 X<>Y
130 X<> d
131 X≠0?
132 ATOX
133 ASHF
134 X=0?
135 ANUM
136 R↑
137 FS? 48
138 GTO 06
139 X<>F
140 R↑
141 208
142 R↑
143 X≠Y?
144 SF 07
145 X(=Y?
146 X=Y?
147 RDN
148 X>Y?
149 174
150 R↑
151 X<>F

152♦LBL 06
153 AOFF
154 R↑

155♦LBL "MKX"
 156 "ANUM"
 157 PASN
 158 RCL '
 159 CLA
 160 STO ↑
161 "⊢      B"
 162 ASTO \
 163 ARCL \
 164 R↑
 165 XTOA
 166 R↑
 167 XTOA
 168 RCL \
 169 ATOX
 170 SIGN
 171 AROT
 172 RCL [
 173 RCL c
 174 "@×iλx̄♦"
 175 ASTO c
 176 R↑

177♦LBL 07
178 RCL IND L
 179 "**"
 180 STO \
 181 X<> [
 182 STO ]
 183 ASTO [
 184 ASHF
 185 X<> \
 186 ASHF
 187 X≠Y?
 188 X<> [
 189 X=Y?
 190 R↑
 191 "⊢****"
 192 STO \
   193 "⊢"
 194 X<> [
 195 STO ]
 196 ARCL c
 197 X<> ]
198 STO IND L
 199 RDN
 200 X≠Y?
 201 ISG L
 202 X≠Y?
 203 GTO 07
 204 X<> Z
 205 STO c
 206 CLST
 207 CLA
 208 CLD
 209 END

372 BYTES
```

horloge utilisez :
Ligne 156 : "SW" ; ligne 161 : 243, 127, 166, 154

10K. Astuces de récupération des plantages

Un "plantage" est une situation dans laquelle le clavier est bloqué et omet de répondre, ou dans laquelle le Catalogue 1 est endommagé. Il n'y a habituellement aucun problème à reconnaitre un plantage, mais le récupérer est une autre histoire. Malheureusement, le MEMORY LOST est nécessaire pour récupérer de nombreux types de plantages.

Si le clavier "se bloque" et que vous ne pouvez obtenir aucune réponse de la touche R/S ou de la bascule ON, il y a plusieurs techniques qui peuvent vous aider à reprendre le contrôle :

1. Pressez et maintenez la touche d'effacement, pressez la touche R/S, relâchez la touche R/S, et relâchez la touche d'effacement.
2. Les HP-41 plus récentes (1982 ou plus tard) ont un mode de remise en place. Vérifiez votre mode d'emploi avant de l'essayer, parce qu'avec une ancienne HP-41 il donne un MEMORY LOST. Il donnera aussi un MEMORY LOST sur une HP plus récente si le clavier n'est pas bloqué. Pressez la touche d'effacement et maintenez le doigt sur elle. Pressez et relâchez la touche ON, puis relachez la touche d'effacement.
3. Enlevez les batteries pendant quelques secondes et remplacez-les. Celà effacera tout sauf les plantages les plus sérieux.
4. La prochaine chose à faire, si vous avez un lecteur de cartes, est d'insérer une carte (de n'importe quel type). Si la carte n'est pas entrainée, enlevez les batteries pendant quelques secondes et réinsérez les, avec la carte toujours en place. La carte doit être entrainée et l'affichage doit répondre, sans MEMORY LOST. Cette technique a été développée par Clifford Stern.
5. Enlevez les batteries et réinstallez-les avec chaque pile inversée (cela ne peut se faire avec la batterie rechargeable de HP). Pressez et maintenez la touche ON baissée pendant 10 secondes. Replacez les batteries avec la polarité normale et pressez la touche ON. Vous devriez obtenir un MEMORY LOST.
6. La simple suppression des batteries pendant la nuit n'effacera pas un plantage sérieux. Les plus anciennes HP-41 retiennent leur mémoire sans batteries pendant la nuit, et les plus récentes peuvent retenir leur mémoire pendant une semaine ou plus.

Si le clavier répond effectivement, mais que le Catalogue 1 n'est pas normal, vous devrez effacer la calculatrice (à l'aide des touches ON et la touche d'effacement dans la séquence décrite dans votre mode d'emploi). Cependant, si vous avez un PPC ROM (voir annexe C), il se peut que vous puissiez rétablir le catalogue 1 ainsi que tous vos programmes. Essayez cette séquence, développée par Clifford Stern :

ALPHA C 0 0 0 2 D ALPHA

(vous pouvez utiliser des espaces à la place des zéros pour gagner quelques frappes)

XEQ "HN", XEQ "E?"

Si le résultat est inférieur à 192 ou supérieur à 511, arrêtez ici. Autrement, continuez:

XEQ "SX", PACK

Vérifiez la catalogue 1 pour voir quels programmes ont été récupérés.

Cette séquence ne traitera pas des cas où les pointeurs du .END. ou du registre 00 ont été altérés. Pour ces cas vous avez besoin d'un PPC ROM, d'une connaissance de la structure du registre système c (voir "La programmation synthétique, c'est facile !"), de persévérence et de chance.

Note de l'éditeur : La HP-41 (en particulier les premiers modèles) est trés sensible à l'électricité statique. Ceci se manifeste surtout quand vous branchez ou débranchez des accessoires comme l'imprimante ou l'HP-IL, certains jours à la température et à l'humidité favorable. Les vètements et les moquettes en synthétiques sont beaucoup plus dangereux que la programmation du même nom. Ce phénomène peut entrainer des pannes définitives nécessitant un retour en service aprés vente. Mais si un jour votre machine refuse de reconnaitre le lecteur de carte, ou se comporte bizarrement, commencez par essayer les méthodes ci-dessus décrites, puis débranchez tout ce qui peut être débranché, posez la machine dans un coin tranquille et attendez. Le lendemain ou quelques jours plus tard, il se peut que tout soit à nouveau en ordre. Il arrive même que, batterie enlevée, au bout de quelques jours, la machine reprenne vie sans avoir perdu ses programmes.

SOLUTIONS AUX PROBLEMES

3.1. "*" APPREC DELREC RCLPT

3.2. voici une solution:

```
01 LBL "PAS"        (imprimez le fichier ASCII)
02 CLX
03 SEEKPTA
04 SF 25
05 LBL 01
06 GETREC
07 FC? 25
08 RTN
09 ACA
10 FS? 17
11 GTO 01
12 PRBUF
13 ADV
14 GTO 01
15 END
```

4.1 Une solution est :

```
XTOA          ajoute le caractère en question.
SIGN
CHS
AROT          le permute vers le début d'ALPHA.
```

La séquence SIGN, CHS est plus rapide qu'une entrée de chiffre -1.

4.2. Voici une séquence typique pour ASTOquer une chaîne :

```
(numéro du registre de départ)
ENTER!
LBL 01
ASTO IND Y
RDN
ALENG
X>0?
GTO 01
```

4.3a. Pour effacer n caractères de la gauche:

```
n             une ligne d'entrée de chiffre
X=0?
RTN           terminer si n=0
LBL 01
ATOX          effacer un caractère
RDN
DSE X         decrémenter n
GTO 01
```

4.3b. Pour effacer n caractères de droite:

```
n
CHS       permuter n caractères de la droite
AROT      à l'extrême gauche d'ALPHA.
CHS
continuer par la séquence 4.3a.
```

4.4. Commencée par "prénom initiale nom" ou "prénom nom" dans le registre ALPHA, la séquence suivante produira "nom, prénom initiale" ou "nom, prénom" :

```
32
POSA      trouver le premier espace
"-, "     ajouter une virgule et un espace
AROT      permuter le prénom derrière le nom
ATOX      supprimer l'espace qui est après le prénom
POSA      trouver l'espace suivant
44
POSA      trouver la virgule
X<>Y
X<Y?      si l'espace est devant la virgule
X<0?      et si l'espace a été trouvé
RTN       alors sauter le RTN et continuer
"⊢ "      ajouter un espace après le prénom
AROT      permuter l'initiale derrière le prénom
ATOX      supprimer l'espace placé après l'initiale
```

4.5.
```
PI
*
RCLFLAG
X<>Y
SIN
X<>Y
STOFLAG
X<>L
/
```

4.6. Le programme "FE" listé ici conserve l'état des drapeau 36-39, en levant les drapeaux 40 (FIX) et 41(ENG). L'approche est similaire à "FEX", sauf qu'un autre RCLFLAG est nécessaire au début pour sauver l'état des drapeaux 36-39 :

```
01 LBL "FE"
02 RCLFLAG          sauver l'état des drapeaux 0-39
03 ENG 0            lever le drapeau 41
04 RCLFLAG          sauver l'état du drapeau 41
05 FIX 0            lever le drapeau 40
06 X<>Y
07 ,39
08 STOFLAG          rétablir les drapeaux 0-39
09 R↑               cette séquence est plus rapide que
10 R↑               l'autre : RDN, RDN
11 41
12 STOFLAG          lever le drapeau 41
13 R↓
14 R↓
15 END
```

Liste du programme « BR »

```
01♦LBL "BR"     08 1 E-6        15 X<> T        22♦LBL 01
02 .1           09 *            16 SIGN         23 REGSWAP
03 %            10 ST+ Y        17 CHS          24 DSE Y
04 +            11 X<> L        18 X>0?         25 GTO 01
05 X<>Y         12 SQRT         19 X<>Y         26 END
06 1            13 R↑           20 RDN
07 -            14 ABS          21 +            43 BYTES
```

ANNEXE A

LES PARASITES VER ET 7CLREG

Si votre lecteur de cartes est une révision 1G ou plus élevée, vous pouvez passer à la discussion du parasite 7CLREG page suivante. Pour trouver quelle révision vous avez, effectuez le Catalogue 2. Si vous voyez un de ces en-tête:

```
CARD READER
CARD RDR 1D
CARD RDR 1E
CARD RDR 1F
```

alors votre lecteur de cartes a le parasite VER.

Voici toute l'histoire du parasite VER (pour les lecteurs de cartes jusqu'à 1F). Quand la fonction VER du lecteur de cartes est exécutée avec un module de mémoire d'extension branché dans le port 2 (les numéros des port sont indiqués au bas de votre HP-41 à côté du numéro de série), le premier registre de ce module est altéré. Le même avertissement s'applique dans le cas où l'on a un branchement du module d'extension dans le port 4 d'une extension de port.

Lorsque vous utiliserez VER dans ces conditions, un registre (emplacement décimal 1007) de vos informations de données ou de programmes dans la mémoire d'extension sera incorrect, à moins qu'il n'y ait pas de données dans le module port 2. Il est même possible que le registre altéré soit un registre d'en-tête de fichier, détruisant la table de la mémoire d'extension.

Si cette discussion n'est pas complètement claire pour vous, revenez-y après avoir lu les chapitres 2 et la section 10C. Pour l'instant, gardez vous d'exécuter la fonction VER du lecteur de cartes si vous avez un module de mémoire d'extension port 2. Si vous devez avoir un module XM port 2, assurez vous au moins que le module port 1 ou 3 soit rempli avant que soit utilisé le module port 2. C'est facile à faire :

1) si vous n'avez qu'un module XM, installez le dans le port 1 ou 3.

2) si vous avez deux modules XM, installez les en même temps (pendant que la calculatrice est éteinte, bien sur).

Si vous suivez cette procédure, le registre affecté par VER sera le 366ième registre de la mémoire d'extension. Si vous ajoutez les tailles de fichiers indiquées dans la table de la mémoire d'extension et que vous ajoutiez 2 autres registres pour chaque en-tête de fichier, vous pouvez comprendre quel fichier contient le 366ième registre. Ce fichier doit être vérifié ou purgé après une opération de VER. Si le 366ième registre est le second des deux registres d'en-tête d'un fichier, ce fichier et tous ceux qui suivent seront probablement perdus. Ce paragraphe deviendra clair après avoir lu la section 10C.

Maintenant passons aux bonnes nouvelles. Il est possible d'éliminer complètement la destruction du 366ième registre de mémoire d'extension. Le programme synthétique "VER" introduit dans la section 10D fait le travail ; en moins de temps qu'il n'en faut pour presser XEQ "VER". Si vous avez une extension de port (annexe C) une autre technique est tout aussi commode. Contentez-vous d'éteindre tous les modules de fonctions d'exntension et de

mémoire d'extension avant d'exécuter VER.

Tous les lecteurs de cartes ont un parasite 7CLREG. La fonction 7CLREG du lecteur de cartes est prévue pour simuler la fonction CLREG de la HP-67-97. Cette fonction 7CLREG peut détruire un module entier de mémoire d'extension. Si vous exécutez 7CLREG lorsque le SIZE est inférieur à 25, quelques une des données proches du registre 360 de la mémoire d'extension seront perdues. Ceci suppose que la procédure recommandée de branchement du module a été utilisée, de façon que le module port 1 ou 3 contienne le registre 365 comme son dernier registre. De plus, 7CLREG risque de causer le non accès à toutes les données d'extension commençant au registre 366. La solution est d'éviter l'emploi de 7CLREG, ou de le précéder de la séquence SIZE? 0,25, X>Y?, PSIZE (voir pages 74-75) pour assurer un SIZE d'au mois 25.

ANNEXE B

TEMPS D'EXECUTION POUR LES FONCTIONS D'EXTENSION

Toutes les fois que vous écrivez un programme, vous faites face à des choix entre différentes manières d'obtenir le même résultat. Un tableau de temps d'exécution de fonctions variées est un outil utile pour faire ces choix. Par exemple, si vous voulez mettre la valeur 1 en X, vous pourriez ne pas remarquer que la séquence CLX, SIGN est légèrement plus rapide que l'entrée du chiffre 1 de 40 millisecondes (65%), cette différence pourrait valoir la peine d'utiliser l'octet supplémentaire de l'espace programme.

Un tableau de temps d'exécution pour des fonctions importantes incorporées (catalogue 3) peut se trouver dans "La programmation synthétique, c'est facile !". Les temps d'exécution, en millisecondes, sont présentés dans cette annexe pour la majeure partie des fonctions d'extension, y compris celles pour lesquelles vous êtes censés avoir des choix, de façon que vous puissiez réaliser le meilleur choix de fonctions quand vous écrivez vos propres programmes.

Ces temps d'exécution on été mesurés par Clifford Stern, à l'aide de son programme d'application sur le module horloge qui a été présenté dans "La programmation synthétique, c'est facile !". Bien que chaque fonctionnement ait été automatisé le processus entier reste toujours un gros effort à cause du grand nombre de variables qui affectent le temps d'exécution.

Temps d'exécution pour les fonctions d'extension

Tous les temps sont listés en millisecondes (millième d'une seconde).

ALENG 168-3,5C-5,8 INT((C-1)/7), où C est le nombre de caractères dans le registre ALPHA

ANUM 95 à 390 (imprévisible)

APPCHR 237+10,2C+12,1R+incréments de fichier* où C est le nombre de caractères ajoutés, et R est le nombre de registres. Cette formule considère que R est le dernier enregistrement du fichier, autrement APPCHR sera plus lent et moins prévisible.

APPREC 211+6,9C+12,1R+incréments de fichiers*

ARCLREC 458+12R+incréments de fichiers*

AROT X>0 : 286-7,77C-6,6 INT((C-1)/7)-10,9X
X<0 : 287-7,77C-6,6 INT((C-1)/7)-10,9(C-|X|)

ATOX X=0 : 145
X>0 : 167-4,28C

CLFL 199+3,24.FLSIZE+incréments de fichiers*

CLKEYS 326+12,2G où G est le nombre de LBL globaux présents

CRFLAS 246+3,6.FLSIZE

CRFLD 246+3,6.FLSIZE

DELCHR imprévisible, mais lent

DELREC imprévisible, mais lent

FLSIZE 72,9+incréments de fichier*

GETP 1099+115.FLSIZE

GETR 58,5+12,5.FLSIZE+2,5(SIZE-FLSIZE)+incréments de fichier*

GETREC 350+12,1R+incréments de fichiers*

GETRX 110+9,9D+incréments de fichiers*, où D est le nombre de registres de données récupérés

GETX 63,5+incréments de fichiers*

INSCHR imprévisible, mais lent
INSREC imprévisible, mais lent
PASN dépend du temps de recherche des Catalogues 1, 2, et 3 jusqu'à ce que le label désigné ou la fonction soit trouvé.
PCLPS 700+1,0P+16G, où P est le nombre de registres de programme effacés
POSA 83,5 à 281
POSFL 190+17,0C+24.5R+incréments de fichiers*, où C est le nombre de caractères examinés, et R est le nombre d'enregistrements examinés
PSIZE 746+4,1X
PURFL 270+incréments de fichiers*, cette formule considère que vous purgez le dernier fichier de la table de la mémoire d'extension. Autrement PURFL sera plus lent et moins prévisible
RCLFLAG 36,8
RCLPT ou
RCLPTA 102+incréments de fichiers*
REGMOVE 77,6.5D, où D est le nombre de registres de données dans le bloc
REGSWAP 75,6+7.4D
SAVEP 350+90.FLSIZE+ increments de fichiers*
SAVER 56+12,5.SIZE+ incréments de fichiers*
SAVERX INT(X)=0 : 102,6+9,9D
INT(X)=0 : 109,3+9,9D
SAVEX 59,6+incréments de fichiers*
SEEKPT ou
SEEKPTA fichiers de données X=0 : 69,6+incréments de fichiers*
,001≤X≤1 : 65,2+incréments de fichiers*
fichiers ASCII X=0 : 102,3+incréments de fichiers*
,001≤X≤1 : 96,2+incréments de fichiers*
X≥1 plus lent et moins prévisible.
SIZE? 56+2,6X
STOFLAG X=donnée alpha : 35,1
X=dd,ff : approx. 58+20F.
où F est le nombre de drapeaux rétablis.
X<>F 100
XTOA X=numérique approx. 48
X=donnée alpha 47+3,7C
où C est le nombre de caractères ajoutés, plus 1 ou 2 ms si ALPHA n'est pas effacé
ASROOM imprévisible
CLRGX 67,6+1,19D
EMROOM 91+incréments de fichiers*
RESZFL imprévisible
ΣREG? 53
X comp. NN? 45 à 50ms, moins environ 4ms si Y=0.

*Les incréments de fichiers sont 12,9(N-1)+9E pour le fichier de travail, ou 136+12,3(N-1)+9E pour un fichier désigné, où N est le numéro de fichier dans la table de la mémoire d'extension (1 et au delà) et E est le numéro du bloc de mémoire d'extension dans lequel le fichier se trouve. E peut être 0, 1, ou 2 (voir figure 10.1).

ANNEXE C

LIVRES, PUBLICATIONS ET MODULES POUR LA HP-41C

Les Editions du Cagire sont fières de pouvoir vous offrir un choix très vaste et unique d'ouvrages et d'accessoires pour la HP-41C. Vous trouverez ci-dessous une liste des titres de livres en français et en anglais, ainsi qu'une liste des matériels les plus importants. Ces matériels ne sont pas d'origine HP (pour les produits HP, également très nombreux, voyez votre fournisseur habituel) et pour la plupart ne peuvent être trouvés que chez nous. C'est pourquoi nous avons développé un service de vente par correspondance et un catalogue qui contient tous les détails que vous pouvez souhaiter sur les ouvrages et les matériels. Demandez nous ce catalogue gratuit, si vous ne l'avez pas déja. De nouveaux ouvrages et de nouveaux matériels apparaissent tous les mois, nous vous informerons régulièrement si vous nous communiquez votre adresse.

Les livres en francais :

HP-41 La programmation synthétique, c'est facile ! de Keith Jarett (traduit de l'américain).
La programmation synthétique de la HP-41, par W.C. Wickes (traduit de l'américain).
Enter, la notation polonaise inverse, par J-D Dodin.
Au fond de la HP-41, par J-D Dodin.
Autour de la boucle HP-IL, par Janick Taillandier.
Calcul sur béton armé, par la SOCOTEC (Eyrolles)

Livres en anglais :

An Easy Course in Programming the HP-41, par Ted Wadman et Chris Coffin.
Calculator tips and routines for the HP-41, par John Dearing.
HP-41/HP-IL System dictionnary, par Cary Reinstein et PPC-USA
Curve fitting for programmable calculators, par William Kolb.

Autres références :

Le mode d'emploi de la HP-41CX, en deux volumes. Une excellente référence générale de la HP-41. Ce sont les manuels les plus complets et les meilleurs qu'HP ait jamais produit. Disponibles chez HP et ses revendeurs (**tous** les manuels HP sont vendus à qui les veut, indépendamment de la calculatrice ; il faut en général les commander).

LES PRINCIPALES REVUES :

En francais :

MICRO-REVUE, la revue du club français PPC-T, actuellement bimestrielle, 132 pages format 14,5x21,5. Sans doute la meilleure référence et la principale source de programmes (synthétiques ou non) pour la HP-41 en France. Cotisation annuelle incluant la revue, 200F (Oct. 84). Spécimen contre 40F en chèque à l'ordre de PPCT, 77 rue du Cagire 31100 Toulouse.

En anglais :

The PPC Calculator Journal, publié par Personal Programming Center (PPC), une association sans but lucratif dédiée à la programmation personnelle. Les numéros de Juillet 1979 (volume 6, numéro 4) a l'actuel contiennent une somme d'iformations incroyable sur le système HP-41. Le PPC Calculator Journal est la source la plus récente et la plus compréhensible pour de telles informations. Cotisation oct. 84 $37, plus $8 de droit d'entrée. S'adresser directement à PPC, POB 9599 Fountain Valley, California 92728-9599 USA. Joindre quelques cartes magnétiques ou coupons internationaux pour la réponse.

PPCTN (PPC Technical Notes) Revue Australienne particulièrement dédiée à la programmation synthétique et aux applications avancées. Diffusée en France par les Editions du Cagire (cf. catalogue).

Des revues existent en allemand, en danois,... nous consulter.

Les matériels :

LE PPC ROM, un module 8K conçu par les membres de PPC et fabriqué par Hewlett-Packard. Le PPC ROM contient 122 programmes à usage général, et il est accompagné d'un manuel de 500 pages, il est d'une grande valeur à la fois pour son utilité et comme outil d'apprentissage, parce que tous les programmes sont entièrement documentés et accompagnés d'une analyse ligne par ligne. Le délai de livraison varie de huit jours à 5 mois car il est très difficile de se le procurer. Il reste le chef d'oeuvre de PPC-USA. Il est écrit en langage normal.

MODULE ZENGRANGE (ZENROM). Le module (anglais et en langage machine) des passionnés de programmation synthétique et de microcode.

MODULE PANAME (français, en langage machine) module destiné à rendre la vie facile à ceux qui utilisent beaucoup de périphériques HP-IL ou qui traitent de tableaux.

MODULE CCD (allemand, en langage machine) orienté vers une amélioration des fonctions d'origine ASN, XEQ et CAT dont la puissance décuple, plus de nombreuses fonctions mathématiques et utilitaires.

LECTEUR DE DISQUETTES HP-IL utilisant des disquettes standard 5' au format IBM-PC. Lecteurs simples (360K) ou double (2x360K), à ne pas confondre avec le lecteur de microdisquettes d'origine HP.

INTERFACE VIDEO GRAPHIQUE qui vous apporte à la fois une interface 80 colonnes 24 lignes, une table tracante en langage HP-GL travaillant sur le même écran et une sortie centronic pour imprimante Epson avec recopie d'écran graphique.

Les programmes :

Un catalogue de programmes professionnels est en cours de constitution, nous le demander.

CASSETTE : Nous vous proposons également une cassette comportant tous les programmes difficiles des livres cités plus haut (cf. catalogue).

ANNEXE D

CODES-BARRES POUR PROGRAMMES

Les codes barre sont fournis ici pour tous les programmes de ce livre sauf les plus petits, de façon que vous puissiez commodément entrer ces programmes dans votre HP-41 à l'aide du crayon lumineux 82153A. Si vous avez un crayon lumineux ou si vous pouvez en emprunter un, celà vous épargnera du temps.

Protégez toujours la surface des codes barre avec une feuille de plastique transparent. Il peut être aussi utile de placer une feuille de papier sombre derrière les codes-barres pour améliorer le contraste.

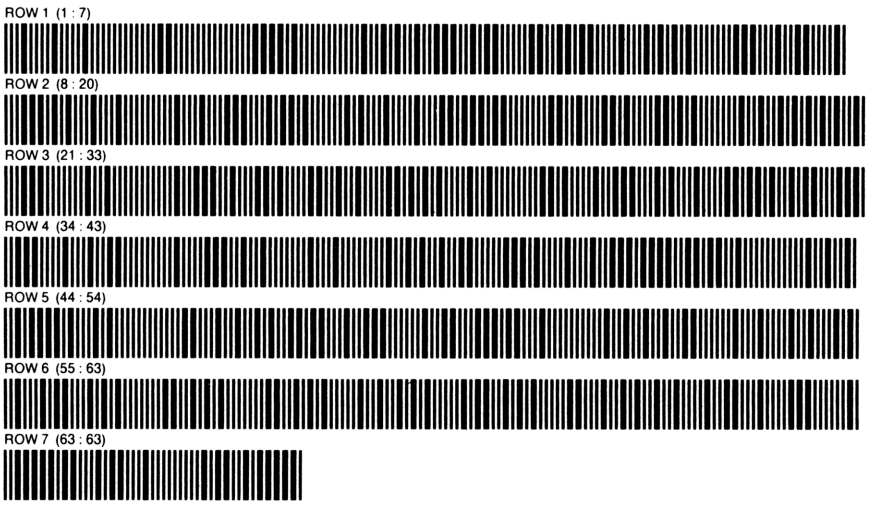
Si vos codes-barres ne sont pas lisibles, essayez d'encrer les barres incomplètes, en examinant les rangs plus rapidement à l'aide d'une règle, ou en maintenant le crayon lumineux à un angle différent. Si tout cela échoue, essayez en un autre crayon.

Si vous possédez un lecteur de cartes ou un lecteur de cassette, vous devez enregistrer ces programmes au cas où votre chien ou votre chat trouverait ce bouquin. La mémoire d'extension ne doit pas être considérée comme un stockage permanent, parce qu'elle est susceptible de MEMORY LOST.

BESSEL FUNCTION

PAGE 1
OF 1

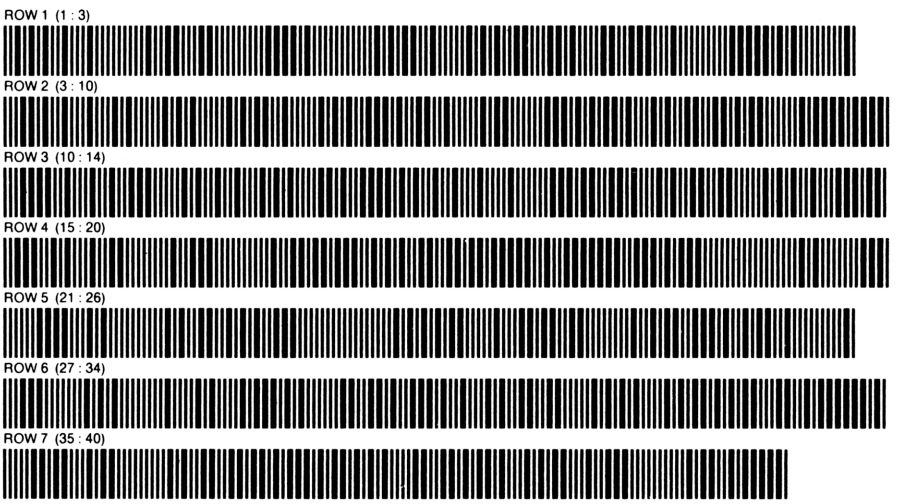
PROGRAM REGISTERS NEEDED: 12



VIEW ASCII FILE

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 13



ROW 1 (1 : 5)

ROW 2 (5 : 14)

ROW 3 (15 : 23)

ROW 4 (24 : 28)

ROW 5 (28 : 37)

ROW 6 (38 : 44)

ROW 7 (45 : 53)

ROW 8 (53 : 62)

ROW 9 (62 : 70)

ROW 10 (71 : 80)

ROW 11 (80 : 88)

ROW 12 (89 : 93)

ROW 13 (94 : 101)

ROW 14 (101 : 105)

ROW 15 (106 : 113)

ROW 16 (114 : 120)

ROW 17 (121 : 128)

ROW 18 (128 : 136)

ROW 19 (137 : 145)

ROW 20 (146 : 152)

ROW 21 (153 : 161)

ROW 22 (161 : 169)

ROW 23 (170 : 171)

BLOCK CLEAR USING Σ REG

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 7

ROW 1 (1 : 4)

ROW 2 (4 : 12)

ROW 3 (12 : 21)

ROW 4 (22 : 23)

BLOCK ROTATE

PROGRAM REGISTERS NEEDED: 7

ROW 1 (1 : 7)

ROW 2 (8 : 15)

ROW 3 (15 : 25)

ROW 4 (25 : 26)

VIEW REGISTERS

PROGRAM REGISTERS NEEDED: 9

ROW 1 (1 : 4)

ROW 2 (4 : 12)

ROW 3 (13 : 22)

ROW 4 (23 : 31)

ROW 5 (32 : 33)

ALPHA SORT

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 11

ROW 1 (1 : 3)

ROW 2 (4 : 12)

ROW 3 (12 : 20)

ROW 4 (21 : 28)

ROW 5 (28 : 36)

ROW 6 (36 : 38)

COUNT BYTES WITH XMEMORY

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 8

ROW 1 (1 : 4)

ROW 2 (5 : 8)

ROW 3 (8 : 13)

ROW 4 (14 : 23)

PROGRAM REGISTERS NEEDED: 58

ROW 1 (1 : 4)

ROW 2 (5 : 10)

ROW 3 (10 : 15)

ROW 4 (15 : 20)

ROW 5 (20 : 26)

ROW 6 (26 : 30)

ROW 7 (31 : 40)

ROW 8 (41 : 49)

ROW 9 (49 : 54)

ROW 10 (54 : 57)

ROW 11 (58 : 63)

ROW 12 (64 : 71)

ROW 13 (71 : 77)

ROW 14 (78 : 89)

ROW 15 (90 : 99)

ROW 16 (100 : 103)

ROW 17 (104 : 114)

ROW 18 (115 : 118)

ROW 19 (118 : 127)

ROW 20 (128 : 132)

ROW 21 (133 : 140)

ROW 22 (141 : 148)

ROW 23 (148 : 156)

ROW 24 (157 : 168)

ROW 25 (168 : 179)

ROW 26 (179 : 187)

ROW 27 (188 : 195)

ROW 28 (195 : 202)

ROW 29 (203 : 210)

ROW 30 (210 : 217)

ROW 31 (217 : 219)

ROW 32 (219 : 219)

ROW 1 (1 : 4)

ROW 2 (4 : 13)

ROW 3 (14 : 19)

ROW 4 (20 : 24)

ROW 5 (24 : 26)

ROW 6 (26 : 29)

ROW 7 (29 : 36)

ROW 8 (37 : 44)

ROW 9 (44 : 45)

ROW 10 (46 : 54)

ROW 11 (55 : 62)

ROW 12 (63 : 66)

ROW 13 (67 : 72)

ROW 14 (73 : 81)

ROW 15 (82 : 86)

ROW 16 (87 : 97)

ROW 17 (98 : 100)

ROW 18 (100 : 106)

ROW 19 (106 : 112)

ROW 20 (113 : 119)

ROW 21 (120 : 124)

ROW 22 (125 : 129)

ROW 23 (130 : 137)

ROW 24 (137 : 145)

ROW 25 (145 : 152)

ROW 26 (153 : 158)

ROW 27 (159 : 162)

ROW 28 (163 : 169)

ROW 29 (170 : 178)

ROW 30 (178 : 184)

ROW 31 (184 : 188)

ROW 32 (189 : 194)

ROW 33 (194 : 201)

ROW 34 (201 : 208)

ROW 35 (209 : 211)

TEXT EDITOR

PAGE 1
OF 4

PROGRAM REGISTERS NEEDED: 115

ROW 1 (1 : 8)

ROW 2 (8 : 16)

ROW 3 (17 : 25)

ROW 4 (26 : 31)

ROW 5 (32 : 39)

ROW 6 (39 : 44)

ROW 7 (44 : 51)

ROW 8 (51 : 57)

ROW 9 (58 : 65)

ROW 10 (66 : 75)

ROW 11 (75 : 78)

ROW 12 (79 : 84)

ROW 13 (85 : 91)

ROW 14 (92 : 98)

ROW 15 (99 : 106)

ROW 16 (107 : 114)

ROW 17 (115 : 122)

ROW 18 (123 : 130)

ROW 19 (130 : 132)

ROW 20 (132 : 137)

ROW 21 (138 : 144)

ROW 22 (144 : 152)

ROW 23 (153 : 160)

ROW 24 (161 : 167)

ROW 25 (167 : 173)

ROW 26 (173 : 180)

ROW 27 (181 : 189)

ROW 28 (190 : 195)

ROW 29 (195 : 201)

ROW 30 (202 : 211)

ROW 31 (211 : 218)

ROW 32 (219 : 225)

ROW 33 (226 : 229)

ROW 34 (229 : 237)

ROW 35 (237 : 244)

ROW 36 (244 : 249)

ROW 37 (250 : 255)

ROW 38 (256 : 264)

ROW 39 (264 : 271)

ROW 40 (271 : 273)

ROW 41 (273 : 279)

ROW 42 (280 : 286)

ROW 43 (286 : 291)

ROW 44 (291 : 296)

ROW 45 (297 : 304)

ROW 46 (304 : 309)

ROW 47 (310 : 312)

ROW 48 (312 : 320)

ROW 49 (320 : 325)

ROW 50 (326 : 331)

ROW 51 (332 : 337)

ROW 52 (337 : 339)

ROW 53 (339 : 339)

ROW 54 (340 : 343)

ROW 55 (344 : 352)

ROW 56 (352 : 360)

ROW 57 (360 : 365)

ROW 58 (366 : 373)

ROW 59 (373 : 379)

ROW 60 (380 : 388)

ROW 61 (388 : 393)

ROW 62 (393 : 397)

PROGRAM REGISTERS NEEDED: 43

ROW 1 (1 : 4)

ROW 2 (4 : 13)

ROW 3 (14 : 21)

ROW 4 (22 : 29)

ROW 5 (29 : 36)

ROW 6 (37 : 44)

ROW 7 (45 : 54)

ROW 8 (55 : 62)

ROW 9 (63 : 71)

ROW 10 (72 : 79)

ROW 11 (80 : 88)

ROW 12 (89 : 98)

ROW 13 (98 : 107)

ROW 14 (108 : 115)

ROW 15 (116 : 122)

ROW 16 (123 : 131)

ROW 17 (132 : 141)

ROW 18 (142 : 151)

ROW 19 (152 : 160)

ROW 20 (161 : 169)

ROW 21 (170 : 178)

ROW 22 (178 : 186)

ROW 23 (187 : 191)

EXTENDED FUNCTIONS

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 11

ROW 1 (1 : 6)

ROW 2 (7 : 9)

ROW 3 (9 : 13)

ROW 4 (14 : 20)

ROW 5 (21 : 28)

ROW 6 (28 : 32)

EXTENDED FUNCTIONS–
TIME MODULE–WAND

PAGE 1
OF 1

PROGRAM REGISTERS NEEDED: 10

ROW 1 (1 : 4)

ROW 2 (5 : 11)

ROW 3 (11 : 14)

ROW 4 (14 : 20)

ROW 5 (21 : 29)

ROW 6 (29 : 29)

VERSION 1
VERIFY CARD
PROGRAM REGISTERS NEEDED: 16

ROW 1 (1 : 4)

ROW 2 (4 : 5)

ROW 3 (6 : 11)

ROW 4 (11 : 17)

ROW 5 (17 : 21)

ROW 6 (22 : 28)

ROW 7 (28 : 32)

ROW 8 (33 : 41)

ROW 9 (41 : 45)

PURGE FILE FIX

PROGRAM REGISTERS NEEDED: 6

ROW 1 (1 : 2)

ROW 2 (2 : 7)

ROW 3 (7 : 15)

ROW 4 (15 : 15)

VERSION 2
VERIFY CARD
PROGRAM REGISTERS NEEDED: 16

PAGE 1
OF 1

ROW 1 (1 : 4)

ROW 2 (4 : 5)

ROW 3 (6 : 11)

ROW 4 (11 : 17)

ROW 5 (17 : 21)

ROW 6 (22 : 28)

ROW 7 (28 : 32)

ROW 8 (33 : 41)

ROW 9 (41 : 45)

EXECUTE PROGRAM IN
EXTENDED MEMORY
PROGRAM REGISTERS NEEDED: 3

PAGE 1
OF 1

ROW 1 (1 : 4)

ROW 2 (5 : 7)

SAVE, GET, SUSPEND, REACTIVATE
KEY ASSIGNMENTS
PROGRAM REGISTERS NEEDED: 31

ROW 1 (1 : 4)

ROW 2 (4 : 11)

ROW 3 (12 : 19)

ROW 4 (20 : 26)

ROW 5 (26 : 32)

ROW 6 (33 : 41)

ROW 7 (42 : 49)

ROW 8 (50 : 58)

ROW 9 (58 : 65)

ROW 10 (65 : 68)

ROW 11 (68 : 76)

ROW 12 (77 : 84)

ROW 13 (84 : 90)

ROW 14 (91 : 99)

ROW 15 (100 : 104)

ROW 16 (105 : 110)

ROW 17 (110 : 111)

PROGRAM REGISTERS NEEDED: 14

ROW 1 (1 : 4)

ROW 2 (4 : 5)

ROW 3 (6 : 13)

ROW 4 (14 : 18)

ROW 5 (19 : 28)

ROW 6 (28 : 37)

ROW 7 (38 : 45)

ROW 8 (45 : 45)

ROW 1 (1 : 4)

ROW 2 (5 : 8)

ROW 3 (8 : 13)

ROW 4 (14 : 21)

ROW 5 (21 : 26)

ROW 6 (26 : 33)

ROW 7 (34 : 41)

ROW 8 (42 : 48)

ROW 9 (48 : 52)

ROW 10 (52 : 57)

ROW 11 (57 : 64)

ROW 12 (64 : 73)

ROW 13 (74 : 82)

ROW 14 (83 : 91)

ROW 15 (91 : 98)

ROW 16 (98 : 104)

ROW 17 (105 : 113)

ROW 18 (113 : 121)

ROW 19 (122 : 128)

ROW 20 (128 : 134)

ROW 21 (135 : 143)

ROW 22 (143 : 151)

ROW 23 (152 : 159)

ROW 24 (160 : 168)

ROW 25 (168 : 176)

ROW 26 (177 : 183)

ROW 27 (184 : 190)

ROW 28 (191 : 197)

ROW 29 (198 : 207)

ROW 30 (208 : 215)

ROW 31 (216 : 223)

ROW 32 (224 : 232)

ROW 33 (233 : 233)

ROW 1 (1 : 4)

ROW 2 (5 : 9)

ROW 3 (10 : 18)

ROW 4 (18 : 25)

ROW 5 (25 : 33)

ROW 6 (34 : 41)

ROW 7 (42 : 46)

ROW 8 (47 : 55)

ROW 9 (55 : 60)

ROW 10 (61 : 67)

ROW 11 (68 : 75)

ROW 12 (76 : 83)

ROW 13 (84 : 92)

ROW 14 (93 : 103)

ROW 15 (104 : 114)

ROW 16 (115 : 124)

ROW 17 (124 : 132)

ROW 18 (132 : 140)

ROW 19 (141 : 149)

ROW 20 (149 : 155)

ROW 21 (155 : 160)

ROW 22 (161 : 167)

ROW 23 (167 : 174)

ROW 24 (174 : 179)

ROW 25 (179 : 185)

ROW 26 (186 : 192)

ROW 27 (192 : 196)

ROW 28 (197 : 204)

ROW 29 (205 : 209)

LEXIQUE DES FONCTIONS ET DES NOMS DE PROGRAMMES

FONCTIONS

ALENG Alpha LENGth - Longeur du contenu du registre ALPHA
ANUM Alpha NUMber - prendre un nombre en ALPHA
APPCHR APPend CHaRacter - Ajouter un caractère
APPREC APPend RECord - Ajouter un enregistrement
ARCLREC Alpha ReCaLl RECord - Rappel d'un enregistrement en ALPHA
AROT Alpha ROTation - Permutation circulaire du registre ALPHA
ATOX Alpha TO X - Du registre ALPHA vers le registre X
CLA CLear Alpha - Effacer le registre ALPHA
CLKEYS CLear KEYS - Effacer les (assignements des) touches
CRFLAS CReate FiLe AScii - Créer un fichier ASCII
CRFLD CReate FiLe Data - Créer un fichier de données
DELCHR DELete CHaRacter - Effacer un caractère
DELREC DELete RECord - Effacer un enregistrement
EMDIR Extended Memory DIRectory - Table de la mémoire d'extension
FLSIZE FiLe SIZE - Taille d'un fichier
GETAS GET AScii file - Récupérer un fichier ASCII
GETKEY GET KEY - Récupérer la frappe d'une touche
GETP GET Program - Récupérer un programme
GETR GET Registers - Récupérer les registres
GETREC GET RECord - Récupérer un enregistrement
GETRX GET Registrers by X - Récupérer les registres selon X
GETSUB GET SUBroutine - Récupérer un sous-programme
GETX GET X - Récupérer dans X
INSCHR INSert CHaRacter - Insérer un caractère
INSREC INSert RECord - Insérer un enregistrement
PASN Programmable ASsiGn - Assignement programmable
PCLPS Programmable CLear ProgramS - Effacement de programmes par programme
POSA POSition in Alpha - Position dans le registre ALPHA
POSFL POSition in FiLe - Position dans le fichier
PSIZE Programmable SIZE - Programmation du nombre de registres
PURFL PURge FiLe - Effacement d'un fichier
RCLFLAG ReCaLl FLAGs - Rappel des drapeaux
RCLPT ReCaLl PoinTers - Rappel des pointeurs
RCLPTA ReCaLl PoinTers by Alpha - Rappel des pointeurs du fichier nommé en ALPHA
REGMOVE REGisters MOVE - Déplacement des registres
REGSWAP REGisters SWAP - Echange des registres
SAVEAS SAVE AScii files - Sauvegarde des fichiers ASCII
SAVEP SAVE Program - Sauvegarde d'un programme
SAVER SAVE Registrers - Sauvegarde des registres
SAVERX SAVE Registrers by X - Sauvegarde des registres selon le contenu de X
SAVEX SAVE X register - Sauvegarde du registre X
SEEKPT SEEK PoinTers - Positionne les pointeurs
SEEKPTA SEEK PoinTers by Alpha - Positionne les pointeurs dans le fichier nommé en ALPHA
SIZE? SIZE ? - Quelle est la taille mémoire ?
STOFLAG STOre FLAGs - Stocker les drapeaux
X<>F X exchange Flags - Echanger X et les drapeaux
XTOA X TO Alpha - De X vers le registre ALPHA

41CX

ASROOM	AScii file ROOM - Place dans le fichier ASCII
CLRGX	CLear ReGisters by X - Effacement des registres selon X
ED	text EDitor - Traitement de texte
EMDIRX	Extended Memory DIRectory by X - Xième entrée de la table
EMROOM	Extended Memory ROOM - Place libre en mémoire étendue
GETKEYX	GET KEY by X - Attendre la frappe d'une touche pendant le temps X
RESZFL	ReSiZe FiLe - Redimentionner un fichier
ΣREG?	summation REGisters finder - Position des registres statistiques ?

Programmes
(par ordre d'apparition dans le volume)

JNX	$J_n(x)$
VAS	View AScii - Visualiser un fichier ASCII
PVAS	Programmable View AScii - Visualisation par programme d'un fichier ASCII
WAS	Write AScii - Ecrire un fichier ASCII
PWAS	Programmable Write AScii - Ecriture d'un fichier ASCII par programme
RAS	Read AScii - Lire un fichier ASCII
PRAS	Programmable Read AScii - Lire un fichier ASCII par programme
PVA	Print or View Alpha - Imprimer ou voir le registre ALPHA
FEX	Fix/Eng indirect X - mode FIX ou ENG selon X
SZFIND	SiZe FINDer - Trouveur de taille mémoire
BC	Block Clear - Effacement d'un bloc
BCΣ	Block Clear by summation registers - Effacement de bloc à l'aide des registres statistiques
CT	Clear Top row - Effacer la rangée supérieure
KB1	Key Board 1 - Clavier 1
VREG	View REGistrers - Montrer les registres
ALSORT	ALpha SORT - Tri alphabétique
CBX	Count Bytes using X-memory - Comptage des octets avec les fonctions d'extension
SOLVE	SOLVE - Résolution d'équation
DERIV	DERIVation - Dérivation
INTEG	INTEGrate - Intégration
COMPARE	COMPARE - Comparaison
NAP	Name/Adress/Phone - Nom/adresse/téléphone
TE	Text Editing - Traitement de texte
HP-16	HP-16
XF	eXtende Functions - Fonctions d'extension
EFTW	Extended Functions/Time module/Wand - Fonctions d'extension, module horloge, lecteur de codes-barres
VER	VERification
PFF	Purge File Fix - Correction du parasite de PURFL
EXM	Execute eXtended Memory - Exécuter les fonctions d'extension
SAVEK	SAVE Keys - Sauvegarder les assignements
GETK	GET Keys - Récupérer les assignements
RK	Reactivating Keys - Réactiver les assignements
SK	Suspend Keys - Suspendre les assignements
WFL	Write FiLe - Ecrire un fichier

RFL	Read FiLe - Lire un fichier
RPF	Retrieve Program File - Récupérer un programme
ASG	ASsiGn - Assigner
PASG	Programmable ASsiGn - Assigner par programme
MKX	Make Key assignments with eXtended memory - Assignements avec le module de fonctions d'extension

PARAGRAPHIC / 34, RUE DU TAUR 31000 TOULOUSE / FRANCE
Dépôt légal octobre 1984

DÉCHAÎNEZ LA PUISSANCE DE VOS FONCTIONS D'EXTENSION

Le module de fonctions d'extension mémoire, incorporé dans la HP-41CX et disponible séparément pour les HP-41C et CV, est le plus puissant module jamais vendu par Hewlett-Packard pour la HP-41. Malheureusement le manuel ne donne qu'une faible idée des possibilités réelles des mémoires d'extension.

HP-41 les fonctions d'extension, c'est facile ! est le fin du fin des livres sur les fonctions d'extension et la mémoire d'extension, par un des meilleurs experts du système HP-41. Le livre ne demande aucune connaissance préalable des fonctions d'extension ou de la mémoire d'extension. En fait il vous conduit pas à pas des concepts de base de la mémoire d'extension jusqu'à l'explication de chacune des fonctions d'extension et de courts exemples d'application.

La deuxième moitié du livre fournit plus de 30 programmes utilitaires, y compris un éditeur de texte, un gestionnaire de liste d'adresses, des programmes pour stocker les fichiers de texte sur cartes magnétiques, des programmes mathématiques (résolution d'équation, intégration, etc.) et beaucoup d'autres. Ces programmes rendent votre module de fonctions d'extension puis puissant, plus agréable et plus amusant à utiliser. Les codes-barres, inclus pour tous les programmes, les rendent aussi faciles à charger qu'à utiliser.

Si vous achetez une HP-41CX ou un module de fonctions d'extension mémoire, vous avec besoin de ce livre !

ISBN 2-86811-008-8