les instructions cachées de la HP-41-C

Je n'aime pas suivre à la lettre les instructions d'un manuel. Mais parfois... des trésors cachés, et pour cause, apparaissent soudain. Un tel « accident » est survenu à une « vieille » HP41 qui faisait partie d'une génération aujourd'hui perdue : elle nous a permis de trouver le défaut de la cuirasse.

A l'origine il y eut un programme pas tout à fait au point (sic) s'arrêtant brutalement sur l'affichage d'erreur : ALPHA DATA. On efface le message. Surprise : le registre X est rempli de caractères chinois ! A l'origine de l'erreur une mauvaise gestion de la pile, un RCL IND X utilisant un argument aux alentours de 900.

Même avec 4 modules le registre 900 n'existe pas et la calculatrice doit, en principe, afficher NON EXISTENT, or, en lieu et place, elle est allée chercher « quelque chose », « quelque part ». Coups de téléphones enthousiastes aux copains, et là nouvelle surprise : sur les 4 HP 41 C dont nous disposons deux seulement ont cette faculté de manipuler les « mémoires hautes ». Ce sont les plus anciennes, acquises toutes deux en septembre 1979, dès la sortie de la HP 41 C. Les deux autres ont été achetées respectivement en novembre et décembre et déjà Hewlett-Packard avait rectifié le tir.

1 Dans le monde des HP41C, il faut casser les instructions pour recoller les morceaux

Les calculatrices de « première génération » ne sont plus. Vous êtes peut-être l'heureux possesseur de l'une d'elles : pour le vérifier entrez 999 RCL IND X, la calculatrice ne doit pas afficher NON EXISTENT. Dans ce cas choyez-là comme un bien précieux.

Nous nous sommes alors décidés à nous lancer dans une plongée profonde.

Il nous a fallu peu de temps pour comprendre que les « mémoires hautes » étaient des « fragments » de la mémoire programme. D'une part notre pêche ramenait en surface des lambeaux d'étiquettes alphabétiques, d'autre part, phénomène beaucoup plus gênant, cette lecture des registres programme s'avérait résolument destructive, puisqu'entraînant l'altération des programmes et éventuellement la perte de contrôle au clavier : ce n'était que le début des nombreux « accidents » qui allaient émailler nos recherches.

Notre premier but fut de décrypter les codes affichés, car ce ne sont pas les instructions en clair qui s'affichent lors du rappel des mémoires hautes, mais leur code.. Dans nos esprits cela pouvait autoriser des trucs « intéressants » comme l'autoprogrammation de la machine. Nous étions encore loin d'imaginer tout ce que nous allions découvrir...

2 Assistés d'une imprimante, nous partons pour un voyage dans l'hyper-espace adressable

L'affichage de la 41 C est très mal équipé pour permettre ce décodage, le tableau de la page 61 donne l'ensemble des caractères affichables avec leur code.

Sur 256 octets possibles la calculatrice en explicite seulement 80, avec une redondance pour les caractères 67 et 91 ([). Les octets restant sont matérialisés par le caractère plein ⊠. Les 21 caractères marqués d'une astérisque ne peuvent être générés directement au clavier. Nous disposions heureusement de trois imprimantes qui allaient nous être d'une utilité majeure.

Ce périphérique est susceptible d'imprimer des caractères différents pour les codes de 1 à 127, elle fournit ces mêmes caractères modulo 128 pour les codes de 129 à 255. Les codes 0 et 128 sont représentés par le caractère ♦ également obtenu avec les codes 10 et 138.

La fonction ACSPEC permet de plus de visualiser sous forme binaire un registre (7 octets) presque complet. La remarquable fonction BLDSPEC permet à l'inverse de générer l'ensemble des 256 octets (nous vous en parlerons dans un article suivant). Armé de ces différents outils et d'une bonne dose de patience il nous a été possible de décoder la totalité des instructions.

La table des codes est donnée dans le tableau ci-contre. Les codes 1 à 143, à l'exception des codes 29, 30 et 31 représentent les instructions d'un octet et n'appellent aucun commentaire particulier.

De 144 à 173 on trouve les instructions de deux octets, le premier octet indiquant la fonction à exécuter, le second étant l'argument de cette fonction. Ainsi :

STO 30 sera codé 145,30 SF 02 sera codé 168,02 TONE 1 sera codé 159,01 SCI 09 sera codé 157,09	(tous les codes sont donnés en décimal)

L'adressage direct des registres étant limité à 99, 7 bits seulement sont nécessaires pour le codage des adresses. Le huitième, le bit de poids fort définit le mode d'adressage : s'il est à zéro l'accès est direct, s'il est à 1 il y a indirection.

TONE IND 06 sera codé 159,134 (134 = 128 + 6)
STO IND 30 sera codé 145,158 (158 = 128 + 30)

Les instructions de branchement et les étiquettes relèvent d'une logique un peu plus complexe.

. Les labels courts codés sur 1 octet sont accessibles par les instructions GOTO codées sur 2 octets. Ces GOTO ont la structure suivante en binaire :

$1011 e_1 e_2 e_3 e_4$ octet 1	$S0_1 0_2 0_3 r_1 r_2 r_3 r_4$ octet 2

Les 4 bits e_1 à e_4 codent pour l'étiquette de 0 à 15. Avant « compilation » les bits du second octet sont tous à zéro. Après compi-

lation (une fois l'adresse relative du branchement calculée) ce deuxième octet indique la distance à laquelle se situe l'étiquette : de la fin du GOTO à l'octet qui précède le LBL. Cette distance en octets est donnée par :
$(r_1\ r_2\ r_3\ r_4) * 7 + 0_1 0_2 0_3$

Le bit S donne le signe de cette adresse relative :
s = 0 si le LBL est après le GTO
s = 1 si le LBL est situé avant GTO

On constate que l'espace maximum adressable est de 15 x 7 + 7 soit 112 octets de part et d'autre de l'instruction GTO ce qui est bien la valeur indiquée dans l'annexe G du manuel d'utilisation.

Les GTO de 3 octets et tous les XEQ ont un espace adressable beaucoup plus étendu. Leur structure est la suivante :

1101 = GOTO 1110 = XEQ

(1110) ou (1101) $0_1 0_2 0_3 r_1$
octet 1

$r_2\ r_3\ r_4\ r_5\ r_6\ r_7\ r_8\ r_9$ — octet 2
$se_1 e_2 e_3 e_4 e_5 e_6 e_7$ — octet 3

Les 7 bits de e_1 à e_7 codent pour l'étiquette de 0 à 127 (mais oui !). Les bits s, o_1 à o_3 et r_1 à r_9 codent pour l'adresse relative après compilation comme pour les GTO courts. La distance maximale est de 511 x 7 + 7 soit 3,5 K octets ce qui donne la taille maximum d'un programme.

Précisons au passage que les étiquettes A à J et a à e sont en fait des étiquettes numériques ayant pour code 102 à 112 pour les étiquettes de A à J et 124 à 127 pour les étiquettes a à e : ceci explique qu'elles ne soient pas accessibles indirectement.
Les étiquettes alpha et les END atteignent le sommet de la complexité.

LBL
$11000_1 0_2 0_3 r_1$ — octet 1
$r_2\ r_3\ r_4\ r_5\ r_6\ r_7\ r_8\ r_9$ — octet 2
$1111 l_1 l_2 l_3 l_4$ — octet 3
$t_1\ t_2 t_3 t_4 t_5 t_6 t_7 t_8$ « chaîne de caractères »
octet 4

Le codage des 2 premiers octets nous est maintenant familier : il donne l'adresse relative du label alpha ou du END *suivant* dans la mémoire programme. C'est cette procédure de chaînage qui permet l'accès rapide aux étiquettes globales.

. L'octet 3 annonce la présence d'une chaîne de caractères et donne sa longueur (maximum 15 caractères) codée sur les 4 bits l_1 à l_4.

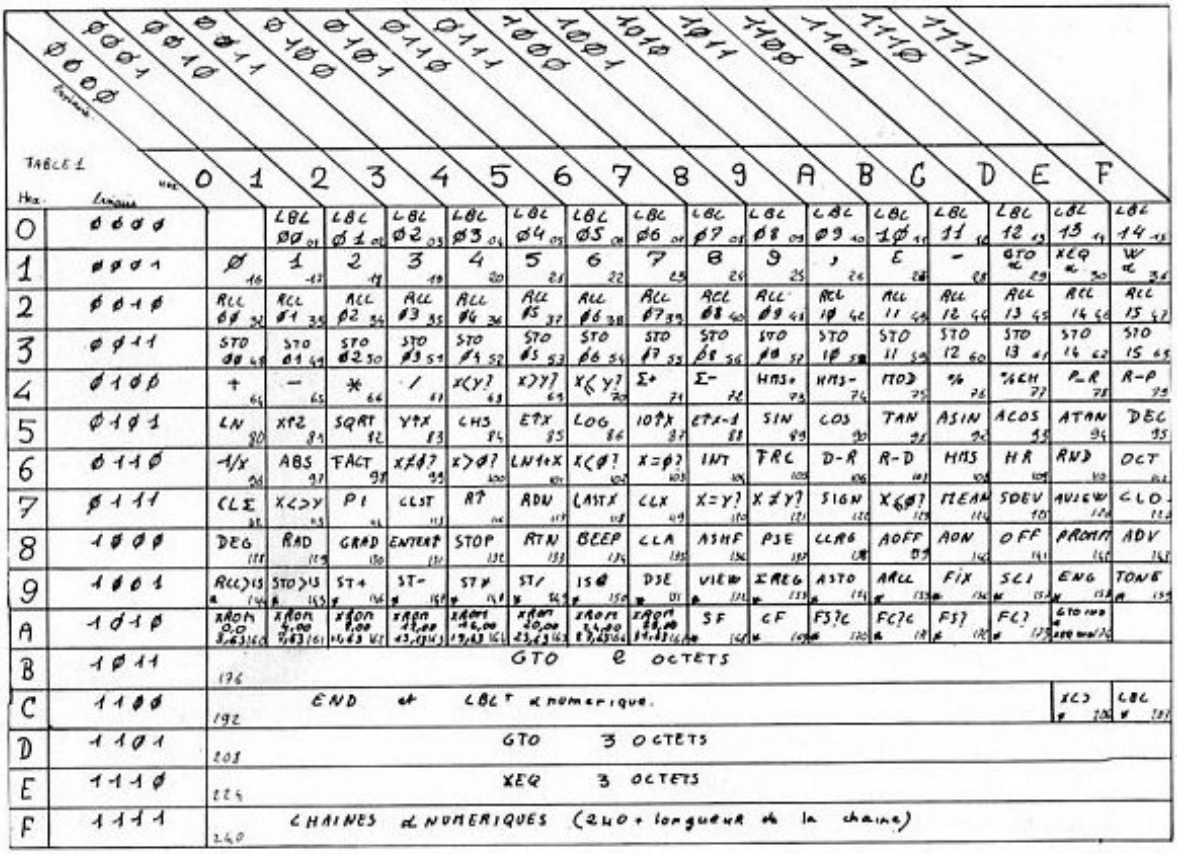
. L'octet 4 donne le code de la touche clavier dans le cas où l'étiquette a été assignée. Nous en reparlerons de façon plus détaillée dans un prochain article traitant des assignations.

Enfin les n octets suivants donnent les n caractères composant l'étiquette.

L'instruction END possède une structure plus simple, puisqu'elle n'occupe que trois octets. Les deux premiers sont rigoureusement identiques à ceux des étiquettes alpha. Le troisième à la structure suivante.
0p001001

Seul le bit p est apparemment variable : s'il est à zéro il ne se passe rien de particulier, s'il est un 1 alors le programme qui précède est PRIVATE.

Voici toutes les instructions de votre HP-41 C :



3 Un cric pour lever le voile sur les mystères d'une calculatrice qui les cachait bien

Jusqu'à présent nous sommes restés très sages, ne faisant que décoder les instructions dont on peut disposer « légalement ». Mais quand on observe les instructions à deux octets on se rend compte qu'il y a des trous. Le deuxième octet permet un adressage de 0 à 127 et certaines fonctions sont loin d'occuper totalement cette possibilité.

La génération de ces fonctions correspondant aux trous, que nous avons appelées les *fonctions synthétiques* fut d'abord effectuée grâce à la fonction BLDSPEC (permise par l'imprimante) et aux calculatrices de première génération dont nous avions la chance de disposer. Le processus était complexe, long et... pas à la portée de toutes les bourses. Par la suite nous avons découvert un outil de synthèse très puissant qui peut être utilisé sur une 41C standard dans sa version de base (celle dont vous disposez aujourd'hui) : le **CRIC.**

CRIC est l'abréviation du « *Chargement dans le Registre alpha d'Instructions Codées* ». Nous en avons bien sûr découvert les propriétés par hasard, en essayant d'assigner des caractères. Il ne nous est pas possible dans ce premier article d'expliciter la séquence d'opérations qui va suivre. Mais ayez confiance, gardez donc les yeux grand ouverts et suivez-nous les yeux fermés (facile non !)

Prenez une calculatrice vierge de tout périphérique et effectuez une réinitialisation complète (éteindre la calculatrice puis, la touche d'effacement étant maintenue pressée, la rallumer. Lorsqu'on relâche cette touche la calculatrice affiche MEMORY LOST.) Assignons maintenant deux fonctions *dans l'ordre :*
ASN « BEEP » 11 (touche Σ +)
ASN « PACK » — 11 (touche Σ —)

Passez en mode programme et frappez
01 ENTER GTO.
CAT 1 et immédiatement R/S la calculatrice doit afficher END.

Appuyez sur ALPHA puis sur la touche d'effacement vous obtenez à l'écran :
4094 (!) Enter↑ (le point d'exclamation n'est pas affiché, mais généré par l'étonnement du 41Ciste de base qui pensait ne pas avoir autant de registres !).

Effectuez BST quatre fois, l'affichage indique 4090 BEEP.

Effacez cette instruction ainsi que LBL03 qui apparaît au pas 4089.

Appuyez maintenant sur la touche $\sqrt{x}$ pour introduire dans le programme le caractère « C » (vous êtes toujours en mode ALPHA). Il ne reste plus qu'à faire un GTO... pour quitter ces eaux étranges.

Nous venons de modifier directement la table d'assignation (sic). La fonction BEEP assignée à la touche Σ + est maintenant remplacée par une autre fonction : le CRIC. Si vous disposez d'un lecteur de carte, n'hésitez pas à enregistrer cette assignation à l'aide d'un WSTS.

Et maintenant quelques explications. L'exécution en mode calcul du CRIC (qui se traduit par l'affichage transitoire de XROM 05, 03) provoque deux phénomènes étroitement liés.

. Le chargement dans le registre alpha d'un certain nombre d'octets correspondant aux codes des instructions présentes en mémoire programme, à partir de l'adresse courante du pointeur programme. Le nombre d'octets transférés est donné par la valeur du dernier chiffre hexadécimal de l'instruction précédant celle actuellement affichée en mode programme (ça va, vous suivez toujours ?) Cette propriété apparaît supérieure au RCL IND des calculatrices de première génération, puisqu'elle s'effectue sans altération des registres de la mémoire programme. C'est en son honneur que nous avons baptisé le CRIC... qui de toute façon est seul disponible sur les machines de deuxième génération.

. La valeur du pointeur programme du registre b est décrémentée du nombre d'octets chargés en alpha et ceci sans tenir compte de la taille des instructions programmées (1, 2 ou plus de 2 octets). Il est donc possible de positionner le pointeur programme « à cheval » (vous me voyez venir) sur une instruction puis d'effectuer de légères modifications.

Introduisons la séquence d'instructions suivantes (après être revenu dans la zone programme !)

01 1	17		
02 RCL IND 16	144	144	codes décimaux (non affichés)
03 RCL IND 31	144	159	
04 RDN	117		

Positionnons le pointeur sur le premier RCL IND 16 au pas 02. Revenons en mode calcul et utilisons le CRIC. En mode alpha l'affichage ne montre qu'un seul caractère (un paté !). En effet l'instruction 1 a pour code Hexa 11. Revenons en mode programme. Apparemment rien n'est changé l'affichage indique toujours 02 RCL IND 16. Effaçons cette instruction et listons le programme. Nous obtenons :
01 1
02 RCL 00
03 TONE M ! !

Que s'est-il produit ? Pour le comprendre revenons à notre séquence de codes :

Code	17	144	144	144	159	117
Octet	1	2	3	4	5	6

Le pointeur programme est initialement positionné sur l'octet 2. L'utilisation du

Ensemble des caractères affichables de la HP-41 C les chiffres correspondent au code décimal, les dessins aux caractères.

0	✳	*(surligné)*	33	!	✳	*(point d'exclamation)*	43	+		64	✳	*(arobasque)*
1			34	"	✳	*(Guillemet)*	44	,		65 à 90	A à Z	
4			35	#	✳	*(Dièse)*	45	−		91	[	✳ *(crochet ouvrant)*
5			36	$			46	.		92	\	✳ *(antislash)*
6			37	%			47	/		93	]	✳ *(crochet fermant)*
12	μ	*(mu grec)*	38	&	✳	*(& et)*	48 à 57	0 à 9		94	↑	
13	∠		39	'	✳	*(apostrophe)*	58	:	✳ *(point virgule)*	95	_	✳ *(souligné)*
29	∠		40	(	✳	*parenthèses*	59	;		96	`	✳ *(texte)*
32		*(blanc)*	41	)	✳		60	<		97 à 101	a à e	
			42	*	✳		61	=		126	∑	
							62	>		127	⊢	✳ *(append)*
							63	?				

CRIC change cet octet en alpha (le pâté) et décrémente d'une unité l'adresse du pointeur, maintenant positionné sur l'octet 3. L'instruction RCL IND 16 affichée est alors la traduction des octets 3 et 4, et non plus 2 et 3. Lorsqu'on détruit cette instruction la séquence des codes devient :

17 144 00 00 159 117
RCL00 TONE M

Le quatrième octet nul disparaîtra aimablement au 1er PACK. Si on généralise le processus on obtient le schéma suivant :

1 — Introduire la séquence
1
RCL IND 16
RCL IND XX
Instruction.

XX est égal au code du premier octet de la fonction à générer moins 128 (c'est facile, une fois qu'on le sait).

La quatrième instruction doit avoir le code du deuxième octet de l'instruction finale.

2 — Après avoir positionné le programme sur RCL IND16 revenir en mode calcul et exécuter le CRIC.

3 — Revenir en mode programme et effacer l'instruction affichée.

4 — Eliminer les instructions 1 et RCL 00 pour ne conserver que la fonction désirée.

Avec le CRIC il vous est maintenant possible de combler les trous de notre tableau. La fonction TONE est la plus attirante ; pour notre délectation, les 117 TONE supplémentaires ne sont pas de simples doublons des dix sons de base. Six nouvelles fréquences apparaissent dans les basses, dont la durée varie considérablement : de cinq secondes pour les plus longs à quelques dizièmes de seconde pour certains, pratiquement inaudibles, mais qui dans une boucle donnent un superbe bruit de crécelle.

Les fonctions de stockage et rappel des mémoires utilisateurs sont également fort tentantes, pouvons-nous franchir la barrière des 99 ? Oui ! les codes de 100 à 111 donnent effectivement accès (directement) aux registres qui portent ces numéros sous réserve que le SIZE sélectionné soit suffisant. L'affichage de ces fonctions est cependant légèrement surprenant.

RCL 100 ET RCL 101 donnent respectivement RCL 00 et RCL 01
de RCL 102 à RCL 111 on obtient des registres étiquetés de A à J (on retrouve le code des étiquettes utilisateur déjà mentionné).

Les codes de 112 à 116 nous ramènent en terrain connu, ils correspondent aux registres TZYXL de la pile opérationnelle.
De 117 à 127 on génère les appellations suivantes

117	M
118	N
119	O
120	P
121	Q
122	⊢
123	a
124	b
125	c
126	d
127	e

Ces nouveaux registres **ne sont pas des registres de données** comme c'était le cas pour les registres 100 à 111

Quand le chargement d'une valeur quelconque dans c entraîna immédiatement un MEMORY LOST...

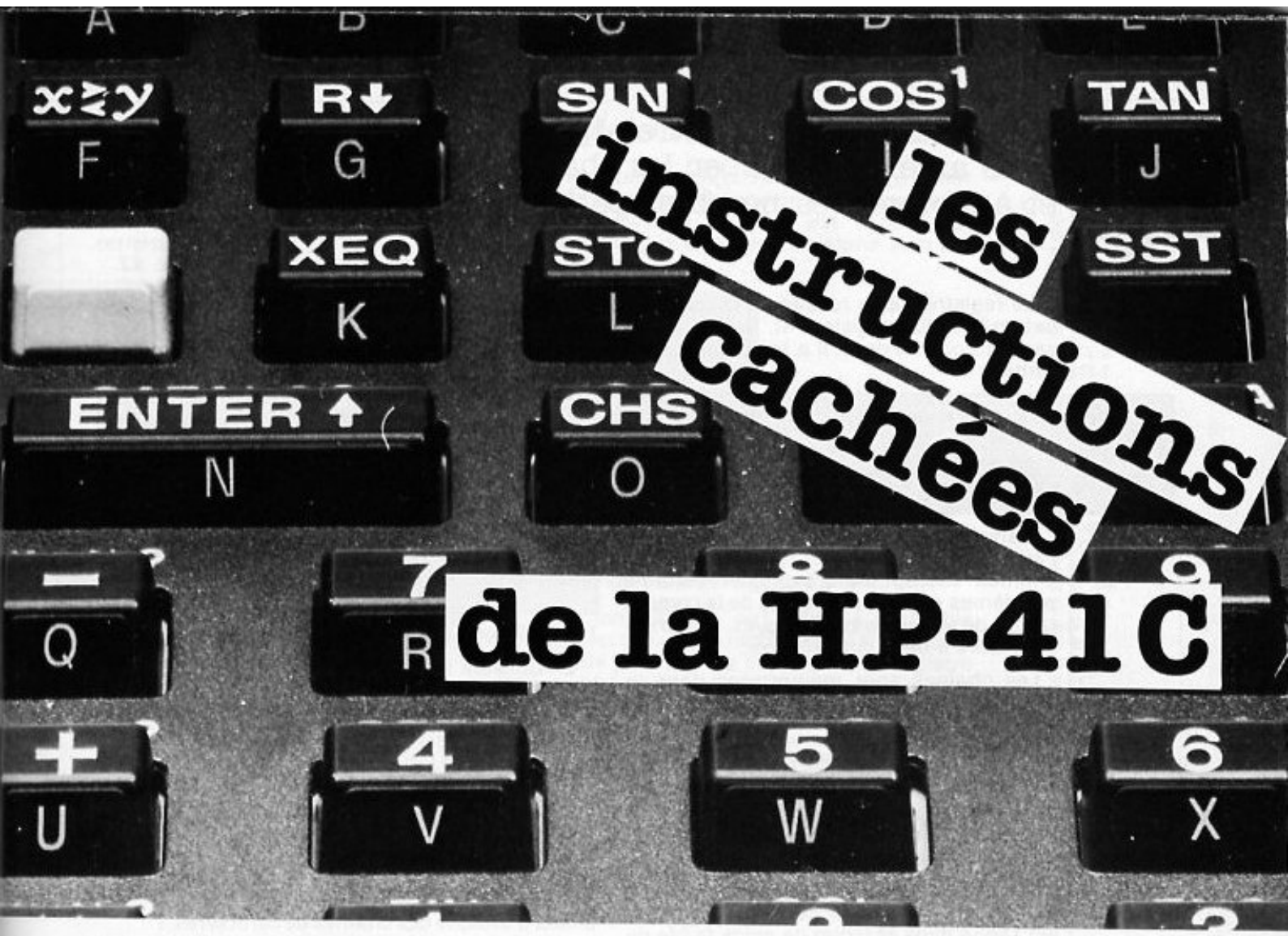
Quand la même opération avec les registres MNO et P remplissait le registre alpha de signes incongrus...

Quand l'altération des registres e et ⊢ bouleversait complètement nos assignations...

Quand la modification du registre d se répercutait aussitôt sur les drapeaux... nous avons compris que nous avions désormais accès aux registres internes de la calculatrice. Et c'est ainsi que nous sommes passés « de l'autre côté du miroir ». Mais ceci est une autre histoire...

à suivre .....

Philippe Descamps,
Bruno Langlois et Michel Sladki



les instructions cachées de la HP-41C

deuxième épisode

de l'autre côté du miroir

Résumé de l'épisode précédent : à la suite d'une opération de stockage indirect légèrement hérétique, Alice a suivi le lapin blanc dans les profondeurs de la HP-41 C . Elle est maintenant en face de 11 registres internes. Chacun porte l'inquiétante inscription « Programmez-moi »...

De ces 11 registres internes étiquetés M, N, O, P, Q, a, b, c, d, e, certains nous sont devenus familiers, d'autres conservent une part de leur mystère.

. les registres M, N, O et les trois octets de poids faible de P constituent les 24 octets du registre alphanumérique.

. le registre Q contient la dernière étiquette alpha appelée, il intervient probablement aussi au cours des opérations d'échanges avec les périphériques car il est périodiquement vidé si l'imprimante est connectée.

. les registres et e contiennent 36 drapeaux indiquant si une touche du clavier est ou non assignée. De plus e est également utilisé pour la numérotation des instructions.

. le registre a et une partie de b contiennent la pile d'adresse de retour des sous-programmes.

. les 12 bits de poids faibles de b codent l'adresse du pointeur programme.

. le registre c contient, codé en hexadécimal : l'adresse absolue du 1er registre statistique ; une constante (16916), qui modifiée, provoque immédiatement un MEMORY LOST ; l'adresse absolue de la séparation entre la zone programme et la zone données ; l'adresse du .END. final de la mémoire programme.

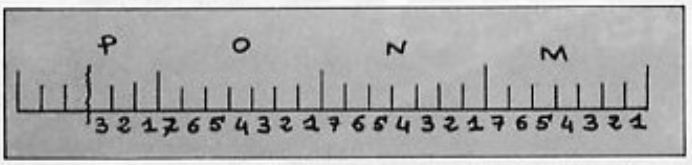
. les 56 bits du registre d correspondent aux 56 drapeaux (de 0 à 55) de la calculatrice.

Tout ceci est fort joli, mais est-ce d'une quelconque utilité ? La réponse est, heureusement, OUI !

4

L'exploitation du registre alpha va nous amener à couper les chaînes en 4 et à trouver les nombres pathologiques

Le registre Alpha résulte de la concaténation des trois registres M, N, O et d'une partie (3 octets) de P. Il a la structure suivante :



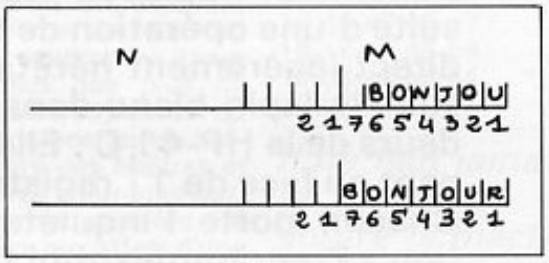
Dans un précédent article (L'OI n° 17), j'avais tenté d'imaginer un codage des chaînes susceptible d'expliquer certains problèmes observés au cours de la comparaison de valeurs alphabétiques. Je parvenais alors à deux conclusions :

. Les chaînes sont mémorisées dans le sens inverse du sens de lecture, et c'est faux.

. Chaque caractère est codé à cheval sur deux octets et c'est parfaitement faux.

Maintenant que nous pouvons réellement voir ce qui se passe, il est aisé d'en démonter le mécanisme.

Le registre Alpha fonctionne comme une pile : chaque nouveau caractère introduit est rangé dans M (octet de poids faible de M) et décale toute la chaîne d'un octet.



La fonction ASHF travaille en mettant simplement à zéro les six octets les plus à gauche de la chaîne. Est-il possible de faire mieux ?

Reprenons notre chaîne BONJOUR de 7 caractères, nous aimerions faire un « ASHF » partiel uniquement sur le premier caractère :

BONJOUR ASHF (1) ONJOUR

Méthode classique (séquence 1).

	ASTO X/ 2	ASHF / 1	ASTO Y / 2	« * * * * * » / 6	
Nb. octets	ARCL X/ 2		ARCL Y/ 2		ASHF 1

Total : 16 octets, X et Y sont perdus.

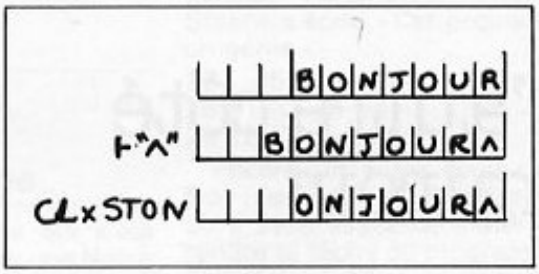
En utilisant les instructions d'accès aux registres internes
« ⊢⊓ » / CLX / STO N. (Séquence 2)

6 octets, seul X a été utilisé.

Si la chaîne est plus longue, son stockage préalable avant troncature utilise encore plus d'instructions. En « logique miroir », il suffit d'annuler les registres O ou P.

En résumé, l'accès direct aux registres alphanumériques, en multipliant les possibilités d'attaque des chaînes de caractères, en révolutionne le traitement.

Encore plus fort ! Etudions de plus près la séquence 2. Que s'est-il passé ?



Et si au lieu du CLX STO N nous avions fait RCL N... Obtenons-nous le caractère B dans le registre X ? La réponse est non !

Contrairement aux fonctions ASTO et ARCL les instructions RCL M, N, O, P, transfèrent les contenus des registres **sans modification.**

Dans la manipulation précédente le registre N contient en hexadécimal :

00 00 00 00 00 00 42

ce qui est le code d'un nombre positif d'exposant 42, la mantisse étant nulle la machine affiche :

0, E 42.

De tels nombres que j'appellerai désormais **pathologiques,** sont passionnants et curieusement très utiles à étudier. Nous y reviendrons une autre fois.

Manipulation des drapeaux (FL)

Le but de ce programme est de pouvoir manipuler tous les drapeaux (Flags) de la 41 C y compris, et surtout les drapeaux internes.

Le numéro du drapeau sélectionné est initialement en X, positif si on désire armer le drapeau, négatif pour le désarmer. En fin d'exécution le drapeau est dans l'état voulu ; la pile et le registre alpha sont détruits.

N°	Instructions	Données	Fonctions
1	Charger le programme		
2	Introduire le numéro du drapeau positif pour l'armer	(—)N	
	négatif pour le désarmer		
3	Exécuter la fonction		XEQ « FL »

Exemple :
Armer le drapeau 52
52 XEQ « FL » ⟶ La machine s'arrête en mode programme
Armer le drapeau 50
50 XEQ « FL » ⟶ Le « canard » reste suspendu à l'affichage (angoissant !)

Le programme commence par déterminer l'octet puis le bit de cet octet qui doit être modifié.

Le contenu du registre d est placé en M et cette chaîne décalée pour que l'octet cible devienne le troisième octet de N. Le registre N est alors rebasculé dans d. Le bit sélectionné (de 16 à 23) est modifié par les fonctions SF IND... ou CF IND... L'ensemble de d est à nouveau replacé dans N et encore décalé pour que les sept octets du registre modifié soient situés dans ☐. où ils sont récupérés et replacés dans d.

Ce programme représente un excellent exemple de manipulation de registre au niveau du bit .

(1) En utilisant les fonctions synthétiques, le listing sur imprimante change un peu (!). Voici les principaux changements : registres : M - crochet ouvrant ; N- anti-slash ; O- crochet fermant.

```
01*LBL "FL"
02 ENTER↑
03 ENTER↑
04 ABS
05 8
06 ST/ Z
07 MOD
08 16
09 +
10 RCL d
11 STO [
12 SF 29
13 SCI IND Z
14 ARCL Y
15 "+-"
16 X<> \
17 X<> d
18 RCL Z
19 X>0?
20 SF IND Z
21 X<0?
22 CF IND Z
23 RCL d
24 STO \
25 RDN
26 ABS
27 INT
28 7
29 X<>Y
30 -
31 FIX IND X
32 ARCL X
33 RCL ]
34 STO d
35 .END.(1)
```

5

Pour se guider dans cet espace étrange, il ne faut pas oublier d'agiter tous les drapeaux possibles

La 41 C offre 30 drapeaux manipulables par l'utilisateur. C'est beaucoup pour la plupart des applications, mais trop restreint pour quelques autres.

Désirez-vous mettre en mémoire la matrice binaire associée à un graphe ? Le nombre de drapeaux dont vous disposez se révèlera vite insuffisant. De l'autre côté du miroir on assiste au miracle de la multiplication des drapeaux.

Rappeler le registre d, le stocker dans une mémoire, générer un autre registre ou réutiliser d'autres valeurs binaires préalablement stockées. Tout ceci est possible... en prenant quelques précautions. La plus importante est de conserver les premiers drapeaux dans l'état suivant :

0 = désarmé 3 = armé.

Ceci a pour effet de donner la valeur 1 au premier digit hexadécimal. Par la suite, au cours des transferts d'un registre dans les mémoires cette valeur sera considérée comme une chaîne de caractères et non comme un nombre. La chose est importante car toute tentative pour **rappeler ou visualiser** un nombre pathologique d'une mémoire autre que les registres internes (pile opérationnelle comprise) normalise le nombre.

N'écoutez pas grand-mère HP quand elle vous dit qu'on ne peut pas manipuler les drapeaux internes de 30 à 55. On peut ! Il suffit de générer la bonne chaîne de caractères et d'en charger le registre d. Tout ceci, bien sûr, à vos risques et périls (veillez à ne pas avoir de programme capital en mémoire car il peut être tristement abimé).

Une autre possibilité intéressante du registre d est son aptitude à générer tous les caractères à la demande. Le processus est un peu plus long qu'avec la fonction BDLSPEC mais il a l'avantage de fonctionner sans imprimante, et comme vous ne disposez pas tous de ce coûteux joujou...

6 Il est temps de lever le rideau car la représentation débutera dans un instant

Abordons, maintenant, le registre C. En voici la structure précise :

Σ_1	Σ_2	Σ_3	Ø	Ø	1	6	9	R_1	R_2	R_3	L_1	L_2	L_3	(16)
7	6	5	4	3	2	1								

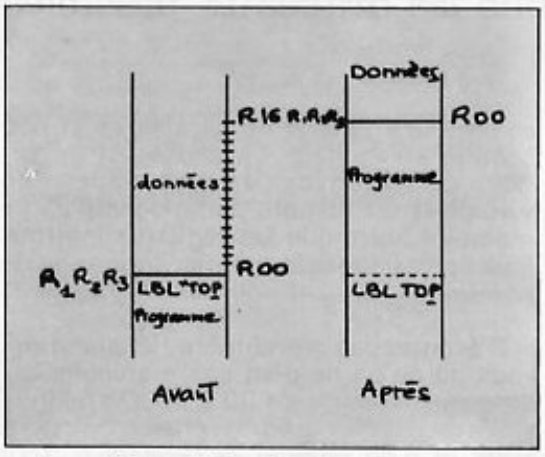
La localisation des premiers registres statistiques est donnée par :

$$\Sigma REG = (\Sigma_1 \Sigma_2 \Sigma_3)_{16} - (R_1 R_2 R_3)_{16}$$

C'est essentiellement la valeur R qui nous intéresse car elle donne en hexadécimal l'adresse absolue du premier registre (RO) des données, le registre immédiatement inférieur est le premier de la mémoire programme (cf. carte ci-contre).

L'opération SIZE est extrêmement complexe. Elle agit en décalant toutes les valeurs contenues dans les registres de données et de programmes en fonction du nouveau nombre de mémoires affectées aux données.

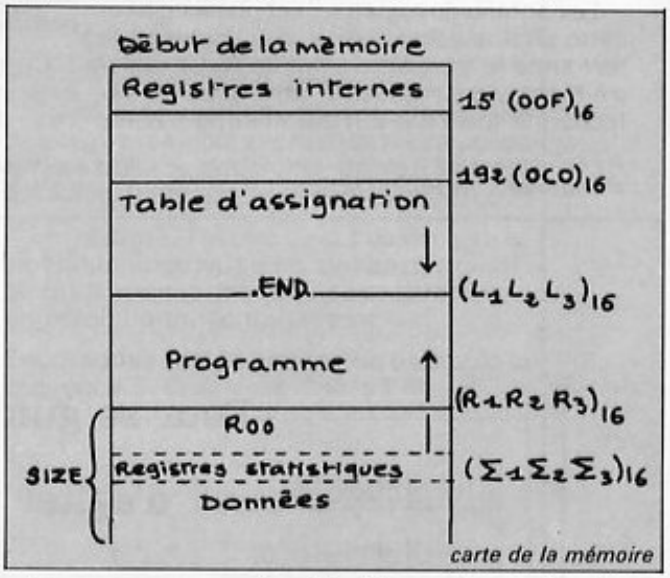
Une modification directe de la valeur R du registre c a un effet plus simple et plus intéressant : la frontière entre la zone donnée et la zone programme (ce que nous appelerons le rideau) est déplacée, sans altération des valeurs présentes dans les registres traversés. Ainsi, si nous augmentons de 16 la valeur de R on a le résultat suivant :



Le registre 16 devient le registre 0. Les 16 premières mémoires données constituent maintenant les premières instructions du premier programme en mémoire ! Immersion, transmutation : le rêve des alchimistes !

Ce phénomène présente deux applications majeures. D'une part une grande liberté dans la synthèse des programmes : génération de chaînes alphabétiques complexes, d'instructions à 3 ou 4 octets. D'autre part, cette technique constitue un véritable ascenseur pour permettre à un sous-programme, ne travaillant pas en adressage indirect, d'être « à la hauteur ».

Si un sous-programme manipule un vecteur implanté entre les adresses R01 à Rnn en utilisant un pointeur en R00, il pourra sans modification travailler sur un vecteur débutant en Rxx avec un pointeur en R(xx - 1) : il suffit de lever le rideau de (xx - 1) « registres ». Pendant cette opération les valeurs contenues dans les registres R00 et Rxx - 2 sont rangées en zone programme et parfaitement protégées si l'on n'effectue pas de tassement de la mémoire programme, ce qui aurait notamment pour effet d'éliminer tous les octets nuls.



carte de la mémoire

Nous avons vu aujourd'hui les registres M, N, O, P, c, d, qui semblent les plus riches sur le plan des applications pratiques. Le rôle du registre Q n'est pas totalement explicité. Les registres a et b permettent de voyager au travers des mémoires mais l'adressage absolu qu'ils utilisent s'accorde mal sur le plan de la programmation avec la plasticité de la mémoire programme.

Les registres ⊢— et e sont dédiés aux assignations. Or le mécanisme des assignations est le complément indispensable des techniques précédemment étudiées. Il nous a permis de créer et d'assigner toutes ces nouvelles fonctions et de les utiliser le plus normalement du monde. Mais ceci est une autre histoire.

à suivre .....

Philippe Descamps
Bruno Langlois

les instructions cachées de la HP-41C

troisième épisode:

La HP-41 persiste et assigne

Résumé de l'épisode précédent : perdue dans les profondeurs de la HP-41 C, Alice ne sait plus à quel registre interne se vouer. Elle en a trouvé 11 mais il lui manque la clé pour pouvoir utiliser toutes les fonctions à sa disposition. Il va lui falloir comprendre le mécanisme des assignations. Y parviendra-t-elle ? Vous le saurez en lisant cet épisode.

« Les affectations de fonctions standard de la HP 41 C à des touches du clavier utilisent un registre (7 octets) pour chaque affectation de rang impair effectuée. ».

Voici tout ce qui est dit dans votre manuel d'utilisation concernant le stockage des assignations : une petite note perdue au bas de la page 186.

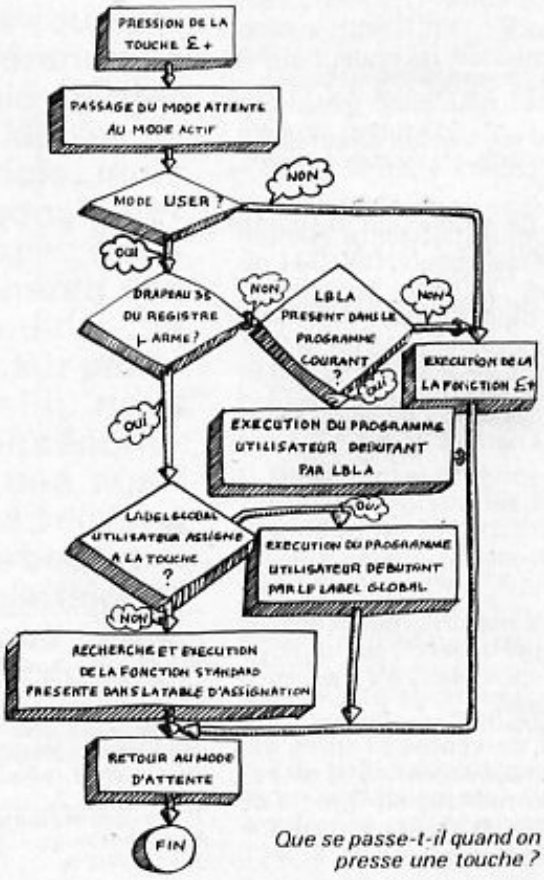
Pourtant le traitement des assignations est un excellent exemple du haut degré de sophistication atteint par le système d'exploitation de la 41C. L'organigramme ci-contre schématise les nombreuses opérations effectuées par la calculatrice lors de la simple pression d'une touche en mode calcul.

Notre but dans ce qui suit est de démonter le mécanisme des assignations pour découvrir de nouvelles possibilités de la HP 41 C.

Le lecteur assidu n'aura qu'un bref effort de mémoire à faire pour se souvenir de la carte mémoire de la 41 C publiée dans le numéro précédent. Du registre 192 au .END. permanent de la mémoire programme s'étend une zone que nous avons appelée table d'assignation. C'est un espace mystérieux théoriquement réservé au seul usage du microprocesseur et sur lequel l'œil de l'utilisateur ne devait jamais se poser.

C'est pourtant ce lieu que nous vous invitons à explorer en notre compagnie.

Mais avant d'effectuer ce voyage il est nécessaire de préparer la calculatrice. Elle ne doit contenir que deux assignations : la commande PACK affectée à la touche $\Sigma-$ et le CRIC, décrit dans le nº 24 de L'O.I., affecté à la touche $\Sigma+$. Vous pouvez parvenir



Que se passe-t-il quand on presse une touche ?

à ce résultat soit en recommençant les opérations nécessaires à l'obtention du CRIC, soit en lisant la deuxième piste d'une carte d'état si vous avez enregistré ces assignations grâce à la fonction WSTS.

Effectuons encore deux nouvelles assignations :
ASN « + » 12 (touche 1/x)
ASN « * » 13 (touche $\sqrt{x}$)

Passons maintenant en mode programme et introduisons la séquence ENTER / GTO... puis revenons en mode calcul. La calculatrice est maintenant prête pour la mise à feu, c'est le CRIC qui va nous servir de propulseur. Le pointeur programme est actuellement positionné sur le premier octet du .END. permanent. L'octet qui précède est le troisième octet du END généré par la calculatrice à la suite du GTO.., il a pour code hexadécimal 09.

Si nous utilisons le CRIC le pointeur programme s'incrémentera de 9 octets franchissant ainsi allègrement la barrière du .END. terminal.

Aussitôt dit, aussitôt fait. En mode USER pressons la touche Σ +. XROM 05, 03 s'affiche brièvement. Sans bruit, sans secousse, nous avons quitté l'atmosphère.

7

Envolés vers un espace mémoire interdit pourquoi ne pas tenter quelques expériences en orbite et vous livrer aux joies de l'apesanteur?

Attention : dès maintenant suivez scrupuleusement les instructions qui vous seront données, tout oubli ou erreur a de grandes chances d'entrainer un MEMORY LOST vous obligeant à tout recommencer.

Après ce conseil salutaire revenons au mode programme. Le paysage est encore familier : la calculatrice au pas OO nous indique le nombre de registres disponibles pour la programmation. Une suite de SST nous révèle un paysage plus exotique :

```
01 ᵀ
02 LBL 03
03 LBL 09
04 LBL 08      } Registre 2 (adresse 193)
05 ᵀ C
06 LBL 00
07 ᵀ
08 LBL 03
09 *
10 RCL 01      } Registre 1 (adresse 192)
11 LBL 03
12 +
13 1
```

Arrêtons nous ici, nous sommes à la fin de la table d'assignation. Nous pouvons à l'aide des fonctions BST et SST nous déplacer librement dans cet espace de 13 instructions. Essayons d'en dégager la structure.

Les instructions 01 et 07 sont des chaînes de zéro caractère (chaînes nulles, code 240) et ne sont pas accessibles directement au clavier, elles servent d'en-tête aux registres d'assignations. Leur destruction provoque de graves perturbations lors d'une tentative d'exécution des fonctions assignées, aussi nous les respecterons.

Au pas 09 et 12 nous retrouvons les deux fonctions * et + que nous avons assignées en dernier, chacune est précédée de l'instruction LBL 03. Pour vérifier qu'il s'agit bien de nos fonctions d'origine effaçons le signe + au pas 12 et insérons à la place la fonction BEEP. (Attention : dans la table d'assignation les insertions doivent toujours être précédées d'une destruction d'un nombre égal d'octets.). Revenons en mode calcul et pressons la touche 1/x. Le petit carillon retentit montrant que la fonction BEEP a bien remplacée la fonction +.

Si nous avons pu retrouver nos deux fonctions standards la commande PACK assignée à la touche Σ — est résolument absente de notre table d'assignation.

Rappellons qu'une commande est un ordre non programmable, elle n'a pas de code propre.

Dans ces conditions, quelle astuce la calculatrice utilise-t-elle pour mémoriser une commande ?

Nous venons de voir que chaque instruction assignée était précédée de LBL 03, au pas 02 nous trouvons encore LBL 03 suivi de LBL 09. Serait-ce notre PACK ? Modifions encore la table et remplaçons LBL 09 par LBL 08, pour voir !

Un retour en mode calcul nous permet de constater que c'est maintenant la commande ON qui est assignée à la touche Σ—. Nous vous laissons le loisir et le plaisir de découvrir les commandes obtenues pour les autres LBL courts (de 0 à 14), et les instructions RCL ou STO d'un octet.

Intéressons nous maintenant au pas 05 nous y trouvons le caractère « C » sans trop de surprise : c'est le CRIC et c'est nous-même qui l'avons placé là. Mais il n'est pas précédé de LBL 03 et occupe 2 octets (code 241, 67), il faut donc supposer que le LBL 03 qui précède nos fonctions d'un octet n'est là qu'à titre de bouche-trou. Effaçons les pas 09 et 08 et entrons l'instruction SF

Correspondance entre la notation matricielle des touches et les codes instructions de la table d'assignation.

	1	2	3	4	5
1	LBL 08 LBL 00 35	9 1 27	RCL 09 RCL 01 19	STO 09 STO 01 11	HMS + — 3
2	LBL 09 LBL 01 34	. 2 26	RCL 10 RCL 02 18	STO 10 STO 02 10	HMS — * 2
3	LBL 10 LBL 02 33	E 3 25	RCL 11 RCL 03 17	STO 11 STO 03 9	MOD / 1
4	LBL 11 LBL 03 32	— * 4 24	RCL 12 RCL 04 16	STO 12 STO 04 8	% X<Y ? 0
5	LBL 12 LBL 04 31	GTO T 5 23	RCL 13 RCL 05 15	STO 13 STO 05 7	
6	LBL 13 LBL 05 30	XEQ T 6 22	RCL 14 RCL 06 14	STO 14 STO 06 6	
7	LBL 14 LBL 06 29	W T RCL 07 21	RCL 15 RCL 07 13	STO 15 STO 07 5	
8	0 LBL 07 28	RCL 00 8 20	STO 00 RCL 08 12	+. STO 08 4	

La première instruction correspond à la touche secondaire, la deuxième à la primaire. Le nombre qui suit indique le numéro du drapeau correspondant dans le registre ⊢ pour les touches primaires et dans le registre e pour les touches secondaires.

** Cette touche, recouverte par la touche ENTER ↑ est inexistante. Le contact est-il présent ?*

00. En mode calcul une pression prolongée de la touche 1/x déclenche l'affichage de XROM 32, 00 et c'est pourtant bien la fonction SF 00 qui est exécutée.

Première découverte majeure, il est possible d'assigner toutes les fonctions de deux octets et plus particulièrement les fonctions d'accès aux registres internes. Mais avant de réaliser cet exploit il nous reste à comprendre le rôle des instructions LBL 08, LBL 00, RCL 01 et 1 qui font suite aux fonctions et commandes assignées. Pour cela nous effectuerons une permutation : effaçons le LBL 00 au pas 06 et remplaçons-le par LBL 08, de même au pas 04 remplaçons LBL 08 par LBL 00. En mode calcul on constate que le CRIC est maintenant affecté à la touche $\Sigma-$ et la commande ON à $\Sigma+$. L'instruction qui suit une fonction a donc pour rôle de préciser la touche à laquelle cette fonction est assignée. Notons que cette modification de l'affectation des touches n'est réalisable que si les touches impliquées ont déjà reçue une assignation et donc que les indicateurs binaires des registres e ou ⊢ correspondants sont levés.

Le tableau ci-contre donne la correspondance entre le code matriciel des touches et le code de la table d'assignation, que la calculatrice visualise par les instructions correspondantes, ainsi que le numéro du drapeau dans les registre e ou ⊢. Le même codage des touches du clavier est également utilisé pour les assignations des fonctions de l'utilisateur. Cette information est stockée non pas dans la table d'assignation mais au niveau du programme dans le quatrième octet du LBL global correspondant (cf L'O.I. nº 24)

L'accès aux registres internes nous amène à une programmation synthétique sans larmes, pleurs et autres grincements de dents

8

Avant de vous décrire la marche à suivre reprenons le listing de nos instructions, listing quelque peu bouleversé depuis le début.

01 T 02 LBL 03 03 LBL 08 04 LBL 00 05 TC 06 LBL 08	Registre 1
07 T 08 SF 00 09 RCL 01 10 LBL 03 11 BEEP 12 1	Registre 2

Commençons par effacer les instructions 11, 10, 09 et 08 puis introduisons RCL IND 16, RCL IND 78, CLD.

Vous reconnaissez la suite d'instructions « classique » pour générer une instruction synthétique, ici l'instruction X<>e (code 206, 127) mais il manque un petit détail : l'instruction 1 qui précède l'instruction RCL IND 16, responsable du pas du CRIC est absente. Il est hors de question de faire disparaître la chaîne nulle qui précède, elle serait aussi difficile à régénérer que l'instruction synthétique que l'on désire obtenir. Il ne nous reste qu'une solution : sauter de plus haut ! Les pas 05 et 06 sont occupés

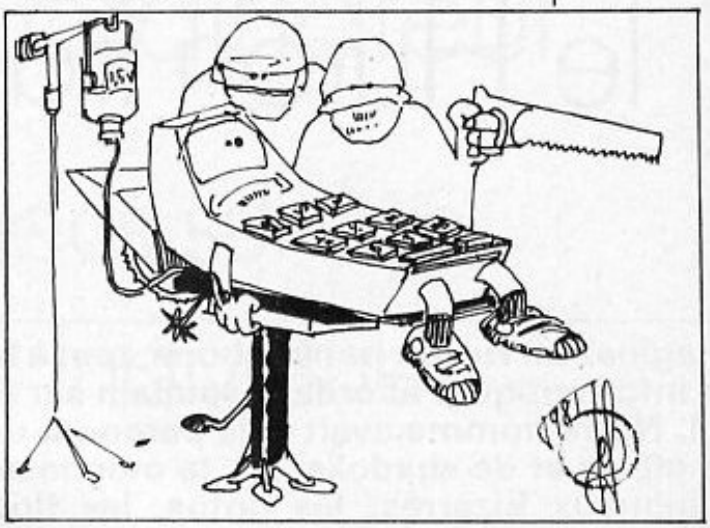
par le CRIC dont nous allons nous servir, ces instructions sont donc tabous.

Nous jetterons notre dévolu sur l'instruction 04 LBL 00. Il nous faut remplacer cette instruction par une autre dont le code hexadécimal a pour chiffre des unités 5. Attention (encore !) il ne faut en aucun cas utiliser le chiffre 5 (code 15 base 16) : la 41 C a la fâcheuse habitude d'insérer une instruction vide avant un nombre et une telle insertion est immédiatement mortelle (MEMORY LOST)

Remplaçons donc LBL 00 par LBL 04 (code 05 base 16) et plaçons le pointeur programme sur l'instruction suivante : le ^{T} C du CRIC.

La suite vous est maintenant familière. Retour au mode calcul, utilisation du CRIC (il est maintenant assigné à la touche $\Sigma-$), retour en mode programme pour effacer RCL IND 16 à l'affichage puis l'instruction RCL 00 qui suit ; X<>e est au pas 05. Il est préférable de faire un peu de ménage pour redonner à notre table une structure plus légale. On remplace au pas 01 LBL 04 par le LBL 00 initial, il faut aussi recréer une assignation pour la touche $\sqrt{x}$, après 04 ^{T} insérons LBL 03 LBL 09 RCL 01 pour réaffecter PACK.

Et voilà c'est terminé.



9

Il faut bien, hélas, revenir parfois sur terre où nous pourrons toujours jouer les passe-murailles avant de partir à la recherche du mu.

Il est temps pour nous d'achever ce périple, un simple CAT 1 nous ramènera en des territoires plus familiers.

La fonction synthétique x<>e est maintenant assignée à la touche 1/X. Que peut-on faire avec ? Essentiellement deux choses :

L'opération CLX X<>e efface toutes les assignations secondaires (obtenues après pression de shift)

Si le contenu du registre X est sauvegardé, par exemple dans M, puis rappelé, un nouveau X<>e restitue ces assignations.

Une autre possibilité plus amusante sinon plus utile (quoi que... !) est de modifier le compteur d'instructions également contenu dans le registre e.

Nous avons vu que le CRIC permettait de jouer les passe-murailles vers le début de la mémoire. Le registre e permet de franchir les barrières vers le haut.

Placez-vous dans un programme à un pas quelconque puis effectuez en mode calcul EEX 1 CHS X<>e. En mode programme vous retrouvez la même instruction au pas 2457 ; une série de BST vous permet de franchir les différents END qui cloisonnent les programmes ; en remontant encore davantage vous vous retrouvez en train d'interpréter votre mémoire donnée comme des lignes de programme. Si on insiste encore on ferme la boucle, on repart à zéro dans les registres d'état puis dans la table d'assignation : ce qui est en haut est en bas !

Les quelques expériences que nous avons eu le plaisir d'effectuer ensemble sont loin d'avoir épuisé le sujet. Vous possédez la clef pour accéder à ce jardin secret qu'est la table d'assignation, n'hésitez pas à vous y promener en solitaire, à faire vos propres découvertes.

De nos explorations personnelles le CRIC reste certes le plus beau fleuron, mais il y a aussi la fonction mu qui nous a grandement facilité la programmation synthétique des chaînes de caractères.

Mais ceci est une autre histoire....

à suivre .....

Philippe Descamps
Bruno Langlois

les instructions cachées de la HP-41C

Quatrième épisode:

La 41 C travaille à la chaîne

Dans la forêt des instructions de la 41 C, Alice vient de découvrir quelques chaînes qui ne manquaient pas de caractères. Las ! Comment les programmer ? Mais bon sang ! Pourquoi ne pas utiliser ces registres qui lui tendent les bras et pour lesquels elle dispose du Cric.
Suivons-la...

Les possibilités d'affichage de la 41 C sont beaucoup plus étendues que ne le laisse prévoir la description du clavier alpha. La première application qui vient à l'esprit est de pouvoir programmer ces caractères supplémentaires, inaccessibles au clavier, mais néanmoins présents dans les MEMs de la calculatrice. Il est agréable d'introduire dans les messages affichés des parenthèses, des crochets, apostrophes et guillemets. Sans oublier les caractères « chinois » dont l'utilité pour la programmation d'un jeu de pendu ne vous aura certes pas échappé.

Mais c'est en conjonction avec l'imprimante que la possibilité de générer de nouvelles chaînes de caractères prend toute sa valeur.

Envisageons l'exemple suivant : ne sachant comment vous y prendre pour déclarer votre flamme à une superbe blonde aux yeux de feu (ou à un grand brun au regard de velours) vous décidez de lui faire parvenir sur carte magnétique ce doux message : « Je t'Aime ! » qui s'imprimera automatiquement au grand ravissement de l'âme sœur.

Vous écririez probablement la séquence :CF 13 « Je » ACA, SF 13 « t » ACA 39 ACCHR CF 13 « A » ACA SF13 « ime » ACA 33 ACCHR PRBUF.

Soit un coquet total de 39 octets.

En utilisant les chaînes synthétiques on peut écrire directement « Je t'Aime » PRA soit 14 octets. Il nous reste incontestablement plus de place pour développer la suite.

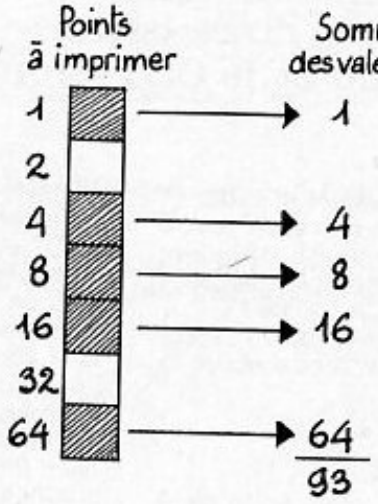
L'imprimante a également la possibilité d'utiliser des caractères spéciaux générés par les fonctions BLDSPEC et ACSPEC.

Instructions BLDSPEC et ACSPEC

L'instruction BLDSPEC sert à construire des caractères spéciaux. Sur l'imprimante, chaque caractère est inscrit dans une matrice 7x7. Pour définir un nouveau caractère, on en définit successivement les 7 colonnes de la manière suivante.

A chaque ligne est associée une valeur numérique, les valeurs des points que l'on veut imprimer sont ajoutées.

Dessinons un "papillon" et construisons notre caractère




```
56,00
     BLDSPEC
66,00
     BLDSPEC
73,00
     BLDSPEC
62,00
     BLDSPEC
73,00
     BLDSPEC
66,00
     BLDSPEC
56,00
     BLDSPEC
     ACSPEC
```

Une fois construit, nous « accumulons » dans un registre tampon (buffer) le caractère par la fonction ACSPEC. L'impression est déclenchée par l'instruction PRBUF.

Le fonctionnement de ces instructions est (presque) limpide : BLDSPEC permet de définir 7 bits du registre, simplement en convertissant en binaire le nombre rentré. Puis l'impression visualise le registre, 7 bits par 7 bits, le premier septet rentré étant à gauche, le bit de poids faible en haut.

Si donc nous accumulons un registre mémoire programme (rappelé par 999 RCL IND X), les 7 derniers bits du dernier octet forment la dernière colonne, le bit de poids fort étant rejeté en avant-dernière colonne. Il en est de même pour les octets comme le dessin l'indique, les octets suivants se « rangeront successivement » de la même manière.

```
3''2' 1  8
2''1' 8' 7
1''8''7' 6
  7''6' 5
  6''5' 4
  5''4' 3
  4''3' 2
```

Ceux qui se sont intéressés à cette technique ont pu observer que le processus de construction est particulièrement vorace en espace mémoire. La programmation du papillon donné en exemple dans l'encadré concernant la fonction BLDSPEC ne nécessite pas moins de 28 octets. Or la représentation finale de ce caractère tient dans 1 registre. Il suffit (!) de programmer cette chaîne de 7 octets pour que, en cours d'exécution, elle soit stockée dans le registre M et de là rappelée en X (RCLM).

Coût total de l'opération : 10 octets.

Laissons-là les possibilités d'impression pour envisager d'autres domaines d'applications moins immédiats.

Vous désirez utiliser la technique d'intégration de Gauss en seize points. Ceci suppose que vous devez introduire dans votre programme 16 constantes numériques ayant chacune dix chiffres significatifs plus deux chiffres et le signe pour l'exposant, soit 14 octets en moyenne.

« Que vient faire ici la programmation de chaîne de caractères ? » vous demandez-vous. Encore une fois la représentation finale de votre constante tient dans un registre et toute l'information est donc susceptible d'être condensée dans une chaîne de 7 caractères qui après transit par le registre M pourra être rappelée en X sous forme numérique, le tout en 10 octets, soit un gain de 4 octets par constante, au total une économie de plus de 9 registres est ainsi obtenue. Si l'on note également que l'interprétation de la chaîne de caractères est plus rapide que celle de la constante numérique correspondante, on peut obtenir un gain de temps à l'exécution de l'ordre de 40 %.

L'initialisation par chaîne du registre d est une autre possibilité intéressante. L'initialisation des drapeaux, le formatage de l'affichage, le choix du mode (USER, ALPHA, trigonométrique) peuvent utiliser un

nombre important de pas de programme ; de plus certaines possibilités comme la mise en alimentation continue ou la déconnection de l'imprimante par logiciel ne peuvent pas être directement programmés. Ici encore la programmation d'une chaîne de sept octets suivie de RCL M STO d règle tous ces problèmes en trois instructions.

Mais passons à la pratique et examinons quelques méthodes propres à concrétiser toutes ces possibilités de rêve.

10

Quelques instruments aratoires nous permettront de mieux cultiver notre jardin : la commande mu et le Cric feront-ils passer le message ?

Et d'abord le CRIC. Toujours le CRIC.

Maintenant assigné à demeure sur la calculatrice, il nous faut savoir comment l'utiliser : reprenons l'exemple passionnant (et passionné) de « Je t'Aime ! ».

Il s'agit de créer une chaîne de 11 caractères, notre premier acte sera d'écrire la séquence :

```
Ø1  1
Ø2  RCL IND 16
Ø3  ᵀ AAAAAAAAAAA
```

Troquons maintenant notre suite de A contre les caractères qui nous conviennent.

Revenons au pas Ø 2 et exécutons le CRIC : la calculatrice affiche RCL IND a ce qui ne nous surprend plus. Le préfixe de la chaîne de code 251 a été absorbé sur le deuxième octet de RCL IND 16. La calculatrice considère maintenant les octets qui suivent comme des instructions et non plus comme des caractères. Profitons-en et insérons :

03	HMS—	code	74	caractère	J
04	LN 1+X		101		e
05	RCLOO		32		espace
06	R/		116		t
07	RCL 07		39		
08	—		65		A
09	INT		105		i
10	HR		109		m
11	LN 1+ X		101		e
12	RCL 00		32		espace
13	RCL 01		33		!

Un BST et la calculatrice affiche

ᵀ Je ▩ 'A▩▩ e !

Les t, i et m minuscules ne sont pas inclus dans la MEM de la 41C et sont donc remplacés par des pâtés, mais l'imprimante elle reconnaîtra les siens.

Il ne nous reste plus qu'à effacer les instructions 1 et RCL IND 16 ainsi que les onze signes moins qui suivent la chaîne obtenue : ce sont les restes de la théorie de A refoulés en dehors de la zone de onze caractères qu'ils nous avaient permis de dimensionner.

Tout cela marche fort bien, mais des problèmes surviendront si vous tentez d'insérer des caractères ayant un code décimal compris entre 16 et 31. Cela se conçoit parfaitement pour les fonctions 29 et 30 : GTO et XEQ qui attendent des suffixes alphabétiques ; de même pour la fonction W non accessible au clavier (dont la seule chose que nous sachions avec certitude est qu'il vaut mieux ne pas la programmer !). Enfin pour les chiffres la calculatrice a la mauvaise habitude de les faire précéder d'un octet nul dont il sera difficile de se défaire (le PACK est inactif à l'intérieur des chaînes). De plus l'utilisation du CRIC nécessite la connaissance complète du code des caractères que l'on désire programmer.

Une autre manière de tourner la difficulté est d'utiliser la commande mu conjointement avec le registre Q.

Cette commande fut trouvée lors d'un voyage dans la table d'assignation. Elle est le résultat de l'affectation du chiffre 3.

Toujours à cause de l'insertion d'un octet nul, il est suicidaire de vouloir introduire ce chiffre directement dans la table d'assignation en utilisant la méthode simpliste qui consiste à presser la touche 3. L'encadré ci-contre décrit les différentes opérations nécessaires pour obtenir cette commande après l'assignation de la fonction BEEP aux touches LN et e^x.

L'utilisation de mu requiert également l'assignation des fonctions STO Q et RCL M (Cf L'OI n° 26 pour la manœuvre).

Quel est maintenant le rôle de cette commande mu enfin à votre disposition ? Son utilisation en mode programme charge automatiquement le chiffre 3 (non précédé de l'octet nul habituel) ; plus intéressante est l'instruction suivante ; il s'agit d'une chaîne alpha d'une longueur maximum de 7 caractères, elle est la représentation inversée du contenu de Q.

Un exemple clarifiera les choses.

Stockons la chaîne « ABCD » dans le registre alpha (non, non, ne cherchez pas, rappelez-vous qu'il suffit de passer en mode alpha et d'appuyer simplement sur les touches nécessaires).

Transférons cette chaîne dans X puis dans Q par STO Q. En mode programme exécutons maintenant la commande, la calculatrice affiche 3 puis au pas suivant nous découvrons la chaîne ᵀ DCBA▩.

Après l'assignation de BEEP aux touches In et e^x, vous devez trouver au plus profond de la table d'assignation la séquence

T
LBL 03
BEEP
HMS +
LBL 03
BEEP
—
LBL 00
RCL IND 16
RCL 19
LBL 00
RCL 00
3
LBL 03
BEEP
HMS +
LBL 03
3
—

Comment obtenir la fonction mu

Commençons par effacer les 5 premières instructions pour les remplacer par :

Le 1 habituel est remplacé par LBL 00 car les nombres sont prohibés dans la table d'assignation.

Le pointeur étant sur RCL IND 16 on exécute le CRIC et on efface le RCL IND 16 affiché.

Il ne reste plus qu'à effacer LBL 00 et RCL 00 et à restructurer la table d'assignation en introduisant après le T les quatre instructions initiales.

C'est notre chaîne initiale réinscrite à l'envers suivie de deux octets nuls et d'un pâté ; ce pâté qui a pour code décimal 16 indique que le registre contient une chaîne alpha et non un nombre. Nous découvrons au passage que dans un registre normal les chaînes sont cadrées à droite. Tout cela est fort intéressant mais seuls les palindromes peuvent subir impunément un traitement aussi... renversant.

Pour obtenir notre chaîne dans le bon sens il faut itérer le processus.

Chargeons notre chaîne inversée dans le registre M en exécutant simplement par un SST l'instruction que nous venons de générer, transférons le contenu de M dans X par un RCL M. La calculatrice affiche la constante —4,4342410 E10 où l'on reconnaît de loin des codes hexadécimaux des caractères ASCII A : 41, B : 42, C : 43... et une partie de D.

Si vous essayez d'extraire le logarithme ou la racine carrée du nombre affiché vous aurez quelques surprises, il est, voyez-vous un peu pathologique ! Il ne nous reste plus qu'à replacer ce nombre dans le registre Q et à réexécuter la fonction mu ; nous obtenons après le 3 de rigueur la chaîne

T ⊠ --ABCD

Programmons à la suite la fonction RCLM et après nous être débarrassés des instructions inutiles (3 divers et chaîne inverse) exécutons ce bref programme. Le contenu de X est bien la chaîne ABCD initiale.

Nous nous sommes donnés beaucoup de mal pour revenir à notre point de départ mais nous avons maintenant en programme une chaîne de 7 octets qui est l'exacte représentation du registre X.

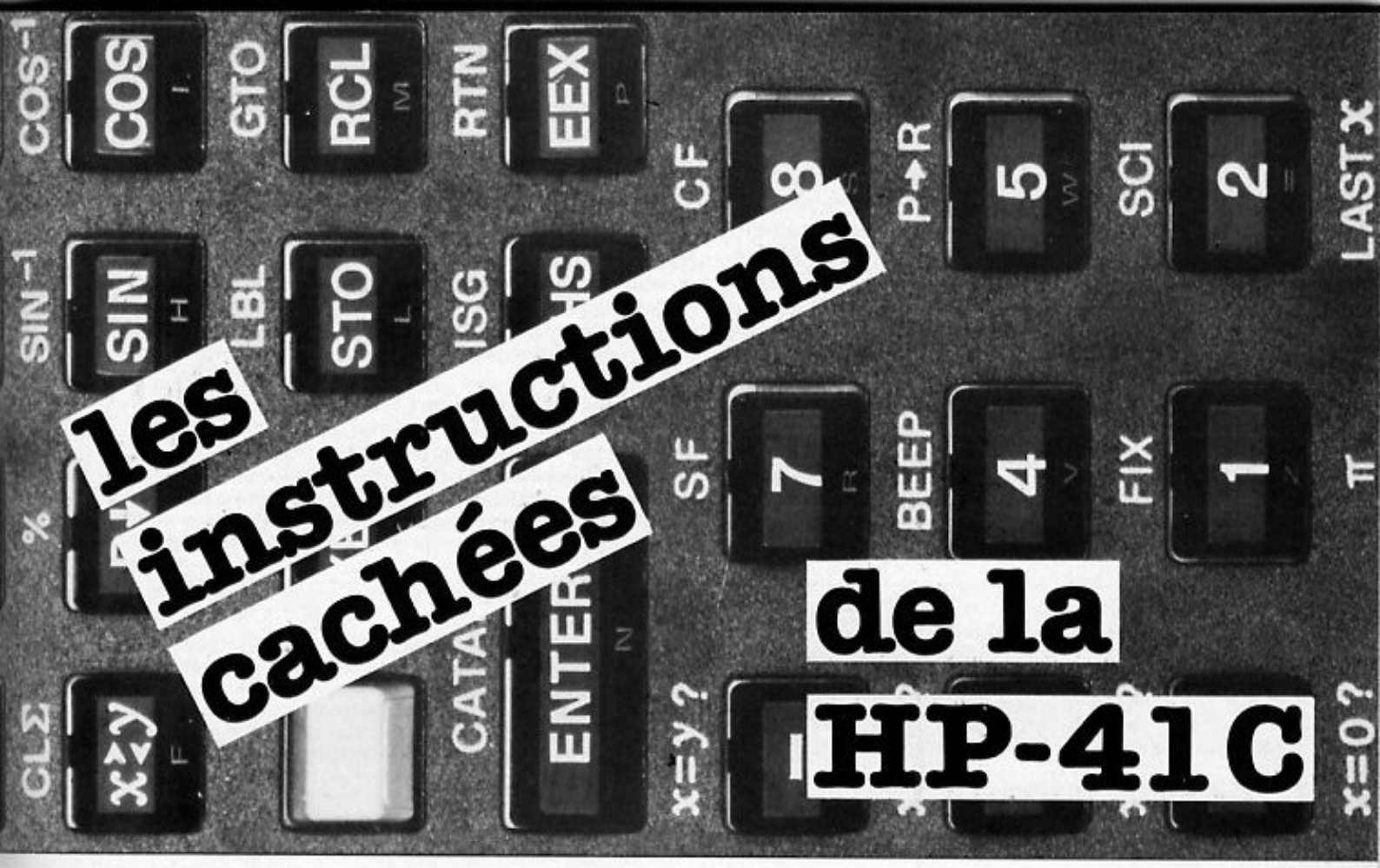
Ce truc fonctionne quel que soit le contenu de ce registre et par l'intermédiaire du registre Q, le contenu d'un registre quelconque peut être condensé en programme sous la forme d'une chaîne de 7 octets. Il peut s'agir d'une constante numérique, d'un code de caractère spécial généré par BLDSPEC, du contenu du registre d obtenu par RCLd. En fait de n'importe quoi qui tienne dans un registre.

Une petite ombre au tableau persiste malheureusement : si l'octet de poids faible du registre à programmer est nul rien ne va plus, la chaîne obtenue ne mesure plus que 6 octets et lors de l'inversion on obtient un catastrophique décalage qui réduit nos efforts à néant.

Enfin si vraiment vous ne pouvez obtenir par les techniques qui viennent d'être développées la chaîne dont vous rêvez il existe encore une méthode qui a l'avantage d'être infaillible. Il faut cependant que vous disposiez pour l'utiliser de trois programmes CD, CDR et MODC... et ceci est une autre histoire.

à suivre .....

Philippe Descamps
Patrick Imbault
Bruno Langlois



les instructions cachées de la HP-41C

Cinquième épisode :

La 41 C essaie de garder son sérieux

Alice arrive au bout de son long périple. Rien n'indique que tout doive s'écrouler comme un château de cartes, au contraire. La voici qui ressort bien décidée à utiliser tout ce qu'elle a découvert pour toutes sortes d'applications. Lesquelles ? L'histoire ne le dit pas : elle sont laissées au bon plaisir d'Alice.

L'utilisation des registres internes est-elle un simple jeu avec la machine, ou bien peut-elle déboucher sur des applications concrètes dans un domaine professionnel ? L'aspect ludique de la manipulation des fonctions synthétiques, que nous vous avons présentées ces derniers mois, est indéniable.

Jouer au prestidigitateur, faire apparaître des signes cabalistiques sur l'écran, modifier des registres sans avoir l'air d'y toucher, bouleverser les assignations et hisser de multiples drapeaux sont des occupations dignes d'occuper les longs jours de l'été. A tout ceci s'ajoute le plaisir du fruit défendu : les filles et fils d'Eve ont toujours aimé croquer les pommes (certains ayant fait de fort belles découvertes en les « regardant » simplement tomber). Les machines conçues par Hewlett-Packard étant considérées comme des produits sérieux, affoler ces pauvres bêtes et leur faire faire les pieds au mur a une incomparable saveur.

Mais l'effet de cette jubilation initiale s'atténuant avec le temps, considérons ces « trésors cachés » d'un œil plus critique. Et tout d'abord, qu'est-ce qui différencie les instructions d'accès aux registres internes des fonctions usuelles de la calculatrice ?

Leur origine n'est pas miraculeuse, la majorité d'entre elles ont manifestement été prévues par le constructeur, probablement pour des raisons de maintenance. Leurs seules différences avec les fonctions standard sont de deux types : dans un premier temps, elles ne sont pas directement accessibles au clavier. En fait ceci n'est plus tout à fait exact : générer une fonction en utilisant le CRIC nécessite il est vrai une

vingtaine de pressions de touches, et cela peut paraître beaucoup... si on oublie que la programmation de la fonction PROMPT en utilise neuf. Enfin cet argument s'effondre totalement quand ces fonctions sont assignées, elles sont alors aussi aisément disponibles que les fonctions « banales ». Dans ces conditions, pourquoi s'en priver ?

La deuxième différence est beaucoup plus sérieuse : ces fonctions sont délicates à manipuler.

La 41 C est d'un naturel assez chatouilleux, chose compréhensible chez une grande Diva de l'informatique de poche, mais si l'on va la taquiner au plus profond de ses circuits, alors là, c'est la grande pâmoison ! Les fonctions synthétiques ne sont pas couvertes par les tests d'erreur du système d'exploitation.

fastidieuse que l'utilisation de BLDSPEC ou ACCHR.

A ceux qui avanceraient qu'une calculatrice programmable est faite pour résoudre des problèmes et non pour en poser je répondrai que toutes ces fonctions artificielles ont leur place dans vos programmes où elles se révèleront aussi utiles voire plus nécessaires que les tests ou les appels de sous-programmes. Sans tarder, administrons une preuve de ce que nous avançons.

Le programme que nous vous proposons (encadré 1) permet de calculer les divers paramètres d'une régression linéaire : pente et ordonnée à l'origine de la droite de régression, coefficient de corrélation, estimation de y pour x donné et réciproquement.

Fonctions statistiques

A partir des valeurs contenues dans les registres statistiques (où qu'ils se trouvent !) Σx, Σy, Σxy, Σx^2 Σy^2 et n, ces programmes calculent:
— Pour RG : les paramètres de la droite de régression pente rendue en X et ordonnée à l'origine dans le registre Y.
— Pour COR : le coefficient de corrélation.
— Pour X? : une estimation de la variable x pour un y donné.
— Pour Y? : une estimation de la variable y pour un x donné.

Exemple :

xi 40, 5 38,6 37,9 36,2 35,1 34,6
yi 104,5 102 100 97,5 95,5 94

Introduire les couples yi ENTER↑ xi Σ +
— Pour calculer la pente : XEQ « RG » → 1,76
à la suite l'ordonnée à l'origine 33,53
— Pour calculer R (coefficient de corrélation)
XEQ « COR » → 0,995
— Pour x = 37 calculer ŷ : 37 XEQ « Y? » 98,65
— Pour y = 95,13 calculer x̂ : 95,13 XEQ « X? » 35,00

Allez donc utiliser cela comme sous-programme...

NB : tout à fait entre nous, c'est le genre de programme que j'aurais aimé trouver dans le module MEM d'application statistique. Hélas à l'appel de « Σ LIN » la calculatrice positionne les registres statistiques en R 10, efface les mémoires R 10 à R 15 et s'arrête dans un joyeux carillon en attendant vos instructions : vous avez la main !

```
01♦LBL "RG"
02 XEQ "FS"
03 ΣREG 00
04 RCL 04
05 RCL 00
06 RCL 02
07 *
08 RCL 05
09 /
10 -
11 SDEV
12 LASTX
13 X<>Y
14 X↑2
15 /
16 RCL 05
17 DSE X
18 /
19 MEAN
20 LASTX
21 *
22 LASTX
23 RDN
24 -
25 RCL [
26 STO c
27 RDN
28 RCL Z
29 RTH
30♦LBL "COR"
31 XEQ "RG"
32 SDEV
33 LASTX
34 *
35 X<>Y
36 /
37 RTH
38♦LBL "Y?"
39 XEQ "RG"
40 RCL Z
41 *
42 +
43 RTH
44♦LBL "X?"
45 XEQ "RG"
46 RDN
47 -
48 R↑
49 /
50 .END.
```

Avouons sereinement que nous avons tout fait pour cela. Si vous tentez une division par zéro la calculatrice refusera en affichant DATA ERROR. Si vous décidez de charger n'importe quoi dans le registre c, rien ne viendra arrêter votre main... et le résultat ne se fera guère attendre.

La seule réponse que l'on peut opposer à cet argument est qu'un MEMORY LOST n'a jamais tué personne (nous en sommes de vivantes preuves). Retirez les accus (parfois 48 heures dans les cas graves) et après cette brutale catharsis votre 41 C se réveillera oublieuse de vos mauvais traitements et aussi docile qu'auparavant.

Nous affirmons donc péremptoirement que les fonctions synthétiques sont des fonctions comme les autres. Il ne faut pas les maintenir dans le ghetto des gadgets et autres « abracadabras ». La manipulation du registre alpha n'est pas plus mystérieuse que celle de la pile opérationnelle, et la création de chaînes artificielles pas plus

De nombreuses calculatrices de milieu de gamme disposent de telles fonctions qui représentent le pain quotidien du statisticien. Alors, où est le problème et que vient faire ici la manipulation des registres internes ? Tout vient du fait que les registres statistiques de la 41 C n'occupent pas une position fixe en mémoire. L'utilisateur peut les assigner à la place qu'il désire... pour oublier ensuite où il a bien pu les mettre ! A cela, trois parades sont possibles :

Décider que les registres statistiques occupent une localisation définie en mémoire, et écrire tous ses programmes en accord avec cette décision.

C'est aussi sacrifier les avantages de la mobilité et cela peut aboutir à d'insolubles conflits quand l'utilisation d'un sous-programme en MEM, donc non modifiable, viendra interférer avec vos registres statistiques.

La seconde solution consiste à sacrifier

Codage d'octet CD

Ce programme génère un caractère à partir de son code décimal compris entre 0 et 255. Ce code initialement en X est remplacé par le caractère correspondant, lequel est également concaténé au contenu initial de alpha, si ce dernier n'excède pas six caractères. Les registres L et T sont perdus, Y et X sont conservés. Le programme utilise un niveau de sous-programme.

n°	Instructions	Données	Fonctions	Affichage
1	Introduire le code décimal à l'affichage $0 \leqslant X \leqslant 255$	×		×
2	Exécuter la fonction		XEQ « CD »	« caractère »

Cette version utilise le registre d. Les fonctions FIX présentes en sous-programme ne sont pas des fonctions standard comme pourrait le laisser croire la liste du programme.

Il s'agit d'instruction synthétiques FIX 10 à FIX 15 ! (codes 156, 10 à 156,15).

Les modifications du format d'affichage n'ont pas d'importance (FIX 10 se comporte comme FIX 9, les autres comme FIX 0) par contre dans le registre d les drapeaux 36 à 39 sont positionnés pour représenter sous forme binaire les cinq « chiffres » hexadécimaux de A à F, exactement comme ils sont la représentation binaire des chiffres 0 à 9 avec les instructions FIX normales. Au total le programme fonctionne de la façon suivante :

1) — Décomposition du code décimal en base 16
2) — chargement du registre d avec la valeur 1,1 et sauvegarde de son contenu initial en X (fonction X<> d)
3) — Synthèse du « chiffre » hexadécimal (4 bits) de poids fort soit par la fonction FIX Ind pour les chiffres de 0 à 9 soit par les FIX synthétiques pour les valeurs de 10 à 15. Le drapeau 25 permet à la calculatrice de choisir entre ces deux options.
4) — Nouvel échange de d et X, décalage de 4 bits sur la gauche par la fonction FRC.
5) — Itération de la séquence n° 3 pour le deuxième chiffre hexadécimal.
6) — Il ne reste plus en utilisant le registre alpha qu'à isoler le troisième octet du registre qui contient la caractère nouvellement synthétisé.

On notera l'utilisation du registre N comme registre numérique transitoire, ce qui permet d'épargner la pile opérationnelle ; et les registres mémoires CD comporte un sous-programme pour traiter le cas où le code initial est nul. Le caractère vide (matérialisé à l'écran par un trait horizontal) ne peut être concaténé par les procédés habituels : ARCL et doit être inséré dans le registre alpha par une chaîne synthétique.

```
01♦LBL "CD"
02 X=0?
03 GTO 16
04 STO \
05 16
06 MOD
07 X<> \
08 LASTX
09 /
10 INT
11 1.1
12 X<> d
13 SF 25
14 FIX IND Y
15 XEQ IND Y
16 X<> d
17 FRC
18 X<> d
19 SF 25
20 FIX IND \
21 XEQ IND \
22 X<> d
23 STO \
24 X<> [
25 ASHF
26 ASTO Y
27 "-"
28 ARCL Y
29 ASHF
30 ASTO Y
31 STO [
32 ARCL Y
33 RDN
34 RTN
35♦LBL 16
36 "├♦"
37 RTN
38♦LBL 10
39 FIX 0
40 RTN
41♦LBL 11
42 FIX 1
43 RTN
44♦LBL 12
45 FIX 2
46 RTN
47♦LBL 13
48 FIX 3
49 RTN
50♦LBL 14
51 FIX 4
52 RTN
53♦LBL 15
54 FIX 5
55 END
```

```
01♦LBL "MODc"
02 ΣREG IND X
03♦LBL "FΣ"
04 RCL c
05 CLA
06 STO [
07 "├♦♦♦♦♦"
08 CLX
09 RCL [
10 X<> d
11 CF 00
12 CF 01
13 CF 02
14 CF 03
15 SCI 2
16 ARCL d
17 SCI 0
18 X<> ]
19 X<> d
20 SCI IND X
21 CLX
22 X<> ]
23 X<> d
24 STO [
25 "├--"
26 X<> \
27 STO L
28 X<> ]
29 "├-"
30 X<> [
31 "├×i"
32 X<> [
33 STO \
34 "├--"
35 CLX
36 LASTX
37 STO [
38 "├-"
39 X<> \
40 X<> c
41 STO [
42 RDN
43 .END.
```

une mémoire pour stocker l'adresse du premier registre statistique. Les possibilités d'interférences sont alors limitées à ce seul registre, mais il faut écrire les sous-programmes de traitement statistique en adressage indirect ce qui oblige à manipuler un pointeur, alourdit la programmation et ralentit l'exécution.

La troisième solution est celle que nous vous proposons (c'est peut-être la bonne !). Ce programme est divisé en deux parties. D'abord l'ensemble « RG », « COR », « X ? » et « Y ? » qui correspond aux fonctions statistiques désirées. A l'exception des pas 25 et 26, ce programme est classique et fonctionne en tenant compte du fait que les registres statistiques sont localisés dans les registres 00 à 05.

Les choses changent lorsqu'on considère le programme « MODc » : long de 43 instructions, il ne contient pas moins de 21 fonctions ou chaînes synthétiques, soit pratiquement la moitié. C'est un taux de « dopage » assez exceptionnel mais votre calculatrice le supportera très bien.

Le but de « F Σ » inclus dans « MODc » est de placer la frontière programme/données au niveau du premier registre statistique. En sélectionnant par les premières lignes du « MODc » la position de ce registre on peut ainsi choisir l'altitude à atteindre.

Ce programme a encore d'autres propriétés qui seront plus aisément compréhensibles une fois que son mécanisme aura été disséqué. Rappelons avant tout la structure du registre c déjà décrite dans L'O.I. n° 25.

	$\Sigma_1\Sigma_2$	Σ_3	X_1	X_2	1	6	9	R_1	R_2	R_3	L_1	L_2	L_3
octets:	7		6		5		4		3		2		1

Σ1 Σ2 Σ3 est l'adresse absolue en binaire; codée sur 12 bits, du premier registre statistique.

R1 R2 R3 est l'adresse absolue en binaire du premier registre de données. Le but de FΣ va être de permuter ces deux adresses.

Quand on échange simplement les octets 7 et 6 avec les octets 3 et 2 on parvient au résultat voulu, mais si la présence de L 1 dans l'octet 6 à la place de X 1 inutilisé par la machine est sans conséquence, la modification, même transitoire, de l'adresse du END final de la mémoire programme (L1 L2 L3), est mortelle, aboutissant à l'horrible MEMORY LOST.

Avant de faire l'échange il nous faut donc impérativement recopier les quatre bits de L 1 à la place de X 1. Voyons pas à pas comment y parvenir.

Initialement le registre c est basculé dans M (pas 4 à 6) puis la chaîne est décalée de 5 caractères par concaténation d'octets nuls (représentés par ◆ à l'impression). Le registre M est alors rappellé dans d où il est numérisé (pas 9 à 14) la partie entière de ce nombre étant L 1, tandis que L 2 et L 3 en constituent la partie fractionnaire. Si vous ne saisissez pas bien le pourquoi de ces manœuvres, rassurez-vous, c'est normal, dans quelques instants tout sera (presque) limpide.

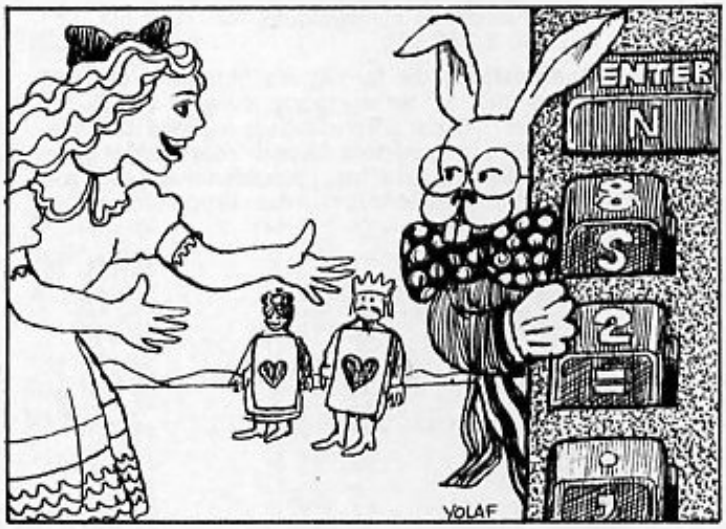
Le contenu de alpha est encore repoussé de 6 octets (pas 15 et 16), récupéré en partie dans le registre ♂ et permuté avec le registre d, le nombre de partie entière L 1 se retrouve alors dans X (pas 18 et 19). Le point capital est que la valeur X 1 du registre c initial occupe maintenant la position des drapeaux 36 à 39 qui, comme chacun sait, indiquent le nombre de chiffre décimaux (cf. le manuel de l'utilisateur page 170).

L'instruction SCI INDX va transférer dans le registre d le « chiffre » L 1 à la suite de Σ1 Σ2 Σ3. Le reste n'est que littérature, nous pouvons maintenant effectuer sans casse la permutation des octets 6, 7 et 3,2. Le registre modifié est replacé dans M pendant que d est restauré (pas 21 à 24). Le contenu d' alpha est encore décalé pour récupérer des octets contenant R1 R2 R3 L1 qui deviendront les octets 7 et 6 du futur registre c (pas 25 à 30).

La constante 0169 est concaténée sous la forme d'une chaîne alpha de code 1, 105 (pas 31). Enfin les septs octets du futur registre c sont complétés par les valeurs Σ1 Σ2 Σ3 L1 L2 L3 qui, pendant la bataille, étaient conservées dans le registre L (pas 32 à 37). Une dernière poussée d' alpha et un registre c tout neuf peut être mis en place, l'ancien étant, à toutes fins utiles, sauvegardé dans M (pas 38 à 42).

A bien y réfléchir, tout cela n'est pas plus compliqué que de jongler avec sept boules (les sept registres X, L, M, N, O, c, d) : il suffit simplement de ne pas laisser tomber les boules. Si le registre alpha a été complètement détruit dans l'opération, la pile opérationnelle a peu souffert puisque le contenu des registres X, Y, et Z est le même qu'au début des opérations. Même le registre des drapeaux, en dépit de tout ce que nous lui avons fait subir, est finalement intact.

Examinons maintenant l'outil dont nous disposons. Après l'appel de la fonction « F Σ » l'adresse des registres statistiques pointe maintenant dans la *zone programme*, au niveau de l'ex-registre 00. Une seconde exécution de « FΣ » restituera la partition initiale, à la condition expresse que Σ1 Σ2 Σ3 n'aient pas été modifiés entre temps (ce que le programme « RG » s'empresse de faire dès le pas 03). Dans ce dernier cas il reste la solution de replacer dans c son ancienne valeur conservée dans M (ce que fait « RG » aux pas 25 et 26).



Après avoir longuement démonté le mécanisme qui confère à « RG » sa transparence, intéressons-nous au devenir des données qui ont été immergées en zone programme : elles sont devenues un ensemble d'instructions. Et l'on se surprend à rêver : si l'on pouvait contrôler au bit près le contenu de ces registres, quelles merveilleuses possibilités nous seraient offertes, nous aurions alors tout loisir de générer n'importe quelle suite d'octets...

Et nous revoici plongés en eau profonde en des lieux fertiles en merveilles, il est décidément difficile de rester sérieux.

Comme à la fin de toute histoire il faut une morale, nous vous proposons celle-ci avant de vous quitter :
S'il te vient, lecteur, une idée délirante, ne la rejette pas comme telle.
Recueille-la, étudie-la et enfin essaye-la.
Une monstruosité logicielle n'a jamais fait exploser une calculatrice et parfois... ça marche.

Mais ceci est maintenant ton histoire...

Philippe Descamps, Patrick Imbault et Bruno Langlois

Voilà cette série est terminée... sauf pour les exemples d'utilisation que vous trouverez prochainement en Fiches Pratiques.