LIST

LE JOURNAL DES AMATEURS DE PROGRAMMATION

nº 5

DÉCEMBRE 1984

BASIC A L'ESSAI

- La jeune Alice 90 : un léger embonpoint
- Simons' Basic, un excitant pour le C.64
- MSX : le standard en vogue
- Les dix tests de LIST : treize nouvelles machines

SUR VOS ÉCRANS

- Comment loucher sur les fichiers Visicalc
- Les lutins jouent les envahisseurs
- Pascal habille les nombres

GROS PLAN SUR TROIS LOGICIELS

- Un Forth pour Spectrum
- Apprendre le Basic sur TO 7
- Un Assembleur pour Dai

(M 2712 - 5 - 20 F)

Belgique : 154 FB - Canada : 2.95$C - Suisse : 6 FS

20 F

GAGNEZ 250.000 F ET SAUVEZ LE MONDE !

LE TALISMAN TEMPOREL

Au cours de la dix-septième mission Apollo, les astronautes américains découvrent sur la Lune un étonnant cristal poli de 40 cm de côté. Réfractaire à toute analyse, le talisman explose sous l'effet de rayons laser, qui dispersent chacun de ses angles. Grâce à des ondes radio, 3 des coins sont retrouvés, mais 5 autres manquent toujours.

Comme par coïncidence, la Lune connaît de très fortes secousses sismiques. Selon le Dr Majid, elles sont liées à l'explosion du cristal et la lune serait elle-même sur le point d'exploser, bombardant la Terre de gigantesques météorites. Il affirme, d'autre part, que chaque coin manquant a été propulsé dans l'espace temps allant de la préhistoire à l'époque moderne.

Votre mission consiste, avec l'aide de la Nasa et de son Chronotron, à remonter le temps pour les rapporter.

A chaque étape vous devez user de toute votre intelligence pour résoudre chaque énigme ; votre adresse vous aidera à survivre. Eureka, en proposant un jeu d'Arcade suivi d'un jeu d'Aventure, teste parfaitement les qualités que l'aventure au réel exigerait de vous.

Vous désirez participer au Concours Eureka : c'est très simple. Il faut que vous possédiez, soit un système Commodore 64 ou Spectrum 48K et acheter le programme Eureka... A vous de jouer.

Pour découvrir la bonne réponse, il faut, tout en reconstituant le cristal, décoder les énigmes contenues dans les poèmes et les illustrations.

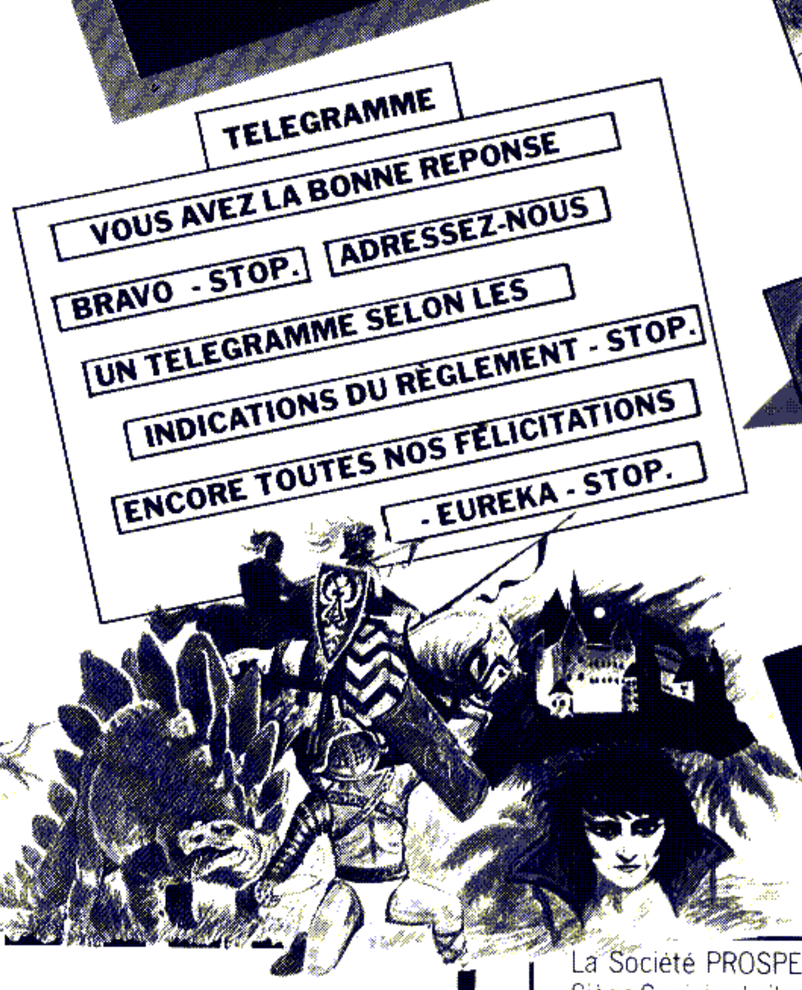
– Le premier à nous télégraphier la bonne réponse recevra le Chèque de 250.000 F.
– Le second gagnera un voyage d'une semaine pour 2 personnes.
– Les 3 suivants, des bons d'achat de 5.000 F TTC.
– Les 245 autres, des bons d'achat de 100 F TTC.

COMMENT GAGNER ?

Vous avez reçu la cassette de jeu et son fascicule. Un bon de garantie, surtout destiné à formaliser votre participation, comporte 2 volets : un que vous devez absolument conserver, l'autre que vous devez impérativement nous retourner dûment rempli. Il constitue la seule véritable preuve de votre participation. N'oubliez donc surtout pas de nous le retourner. Si, d'autre part, vous avez découvert la bonne réponse, ne nous téléphonez pas ! Adressez-nous un télégramme répondant aux conditions prévues dans le règlement.

Eureka est un programme aussi spectaculaire qu'une super production de cinéma. Les graphismes et les animations sont surprenantes de vérité. La bande sonore et les bruitages vous replongent, grâce à une simulation parfaite, dans chacune des cinq époques. Vous percevrez tout.

Il est joint à ce programme un fascicule détaillé qu'il est essentiel de lire attentivement. Illustré, il contient toutes les explications du jeu et les énigmes auxquelles il vous faudra répondre.



IAN LIVINGSTONE

Déjà auteur de romans d'aventure, vendus à plus de deux millions d'exemplaires, Ian Livingstone est le créateur d'Eureka.
Il a imaginé les énigmes et les pièges les plus retors. Il est d'ailleurs le seul, pour l'instant, à connaître la bonne réponse. Programmé par les équipes d'Andromeda, sous la direction de Donat Kiss et Andras Csascar, Eureka représente 5 années de travail et le concours de 4 graphistes , 2 musiciens et d'un professeur de logique.
Nous voulions une aventure qui vous pousse dans vos derniers retranchements. Ils l'ont fait !

Eureka, c'est cinq aventures en une seule cassette.
Remontez le temps et sauvez le monde !

AGE PREHISTORIQUE

Seul, sans défense, vous vous retrouvez à l'aube des temps. Autour de vous ? la jungle, le bruissement des feuilles, des pas dans les broussailles. Votre sang se fige. Une ombre immense vous recouvre. Le sol tremble.

LA ROME ANTIQUE

Le départ de la course de char va être donné. Les autres concurrents vous observent et vous louchent du regard. tous au même gabarit, ils vous dépassent de deux têtes et leurs chevaux semblent mieux entraînés que le vôtre. Vous parcourez le stade du regard. La course est partie !

LE MOYEN AGE

Les créneaux de la Tour de la Fée Morgane ne peuvent rien pour empêcher la brise glaciale de transpercer votre armure. Seul en haut de cette tour, vous entendez une voix qui vous crie de la rejoindre, là, dans l'ombre. Derrière vous, des pas résonnent. Vous vous décidez à descendre au plus profond de la tour, Des hurlements déchirent la nuit...

COLDITZ

Un long couloir, gris, sombre et glacé. Vous avancez pas à pas, attentif au moindre bruit. Du bout du couloir vous parvient une conversation assourdie.
A votre droite, une porte ! Sur la porte un mot : VERBOTEN !
Vous l'ouvrez. Le bruit métallique des bottes se rapproche. Vous serrez les poings.

LES CARAÏBES MODERNES

Vous remontez le temps à la vitesse de la lumière. Dans l'immense laboratoire du Dr Von Berg, l'écran de l'ordinateur affiche les informations.
Callé dans un fauteuil, les mains posées sur le clavier, vous vous préparez à un duel où l'arme est l'esprit...

BON DE COMMANDE

Je désire recevoir le JEU EUREKA SUR CASSETTE, au prix de 250 F. Je ne le recevrai qu'à partir du 1/12/1984, date de début du concours ; et mon règlement ne sera encaissé, au plus tôt, que 8 jours avant la livraison du jeu.

JEU EUREKA POUR COMMODORE 64 ☐ **250 F TTC.**
JEU EUREKA POUR SPECTRUM 48 K ☐ **250 F TTC.**

Nom _____ Prénom _____
Adresse _____
Ville _____ Code postal : |_|_|_|_|_|

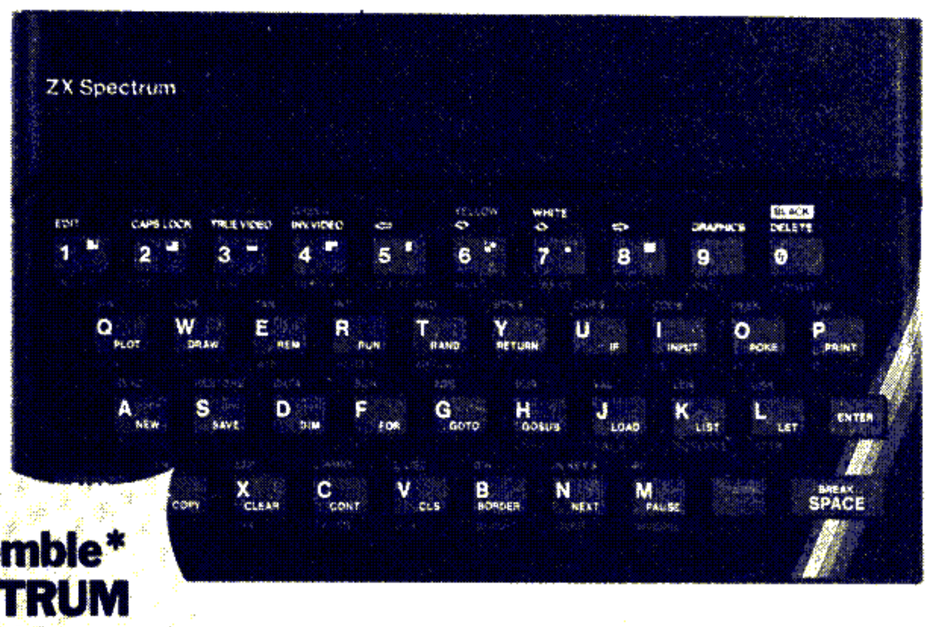
Ci-jointe la somme de F _____ TTC, par chèque bancaire à l'ordre de EUREKA INFORMATIQUE , 39/41, rue Victor-Massé - 75009 PARIS.

L T

EXTRAIT DU RÈGLEMENT

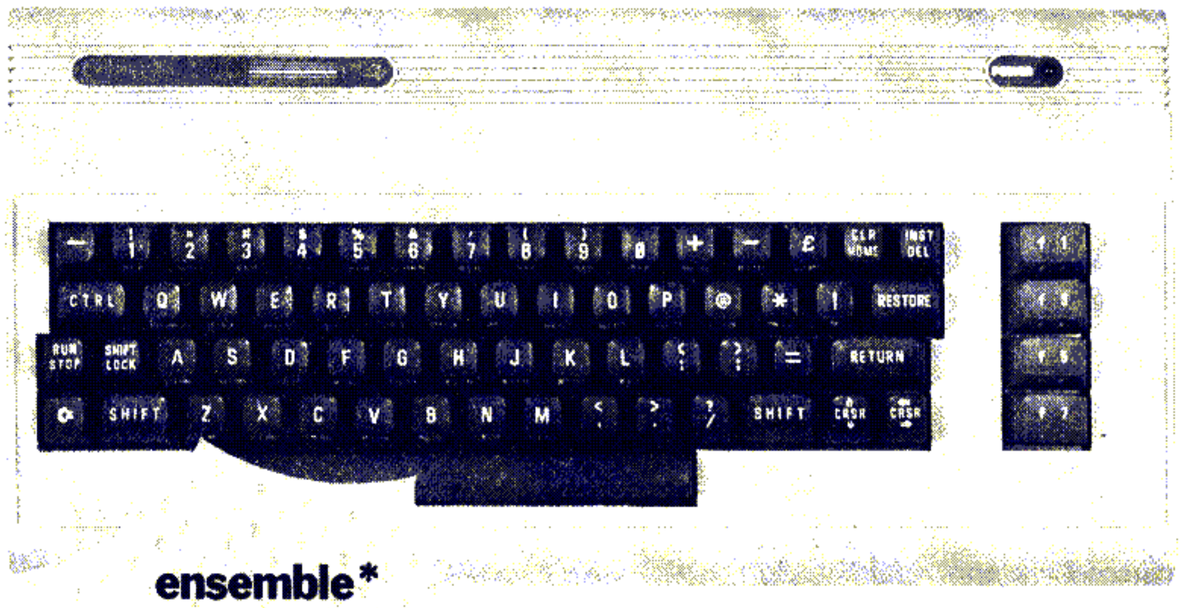
La Société PROSPECTIVE INTERNATIONALE DE DISTRIBUTION, dont le Siège Social est situé 39, rue Victor-Massé - 75009 PARIS, organise à partir du 1er Décembre 1984, un Concours avec obligation d'achat intitulé EUREKA. Le premier dépouillement aura lieu le 31 Mars 1985. Si nécessaire, un dépouillement mensuel sera ensuite effectué jusqu'à ce que les 250 Prix soient distribués. Ce concours est ouvert à toute personne physique résidant sur un territoire francophone, à l'exception du Personnel des Sociétés organisatrices ainsi que toute personne ayant participée à l'élaboration, la promotion, la distribution et la revente du jeu EUREKA.
Pour participer au Concours, il faut : acheter le jeu EUREKA et renvoyer le Bon de Participation joint ; pour gagner, il faut être l'une des 250 premières réponses aux énigmes contenues dans le jeu et ce, par télégramme.
Le réglement complet a été déposé chez Maître JAUNATRE, Huissier de Justices à PARIS, et peut-être obtenu, sur simple demande accompagnée d'une enveloppe timbrée à :
Concours EUREKA - 39, rue Victor Massé - 75009 PARIS.

OFFRE EXCEPTIONNELLE!



ensemble*
SPECTRUM 48K
2.890 F

Un ordinateur SINCLAIR SPECTRUM 48K / Un interface Peritel / Un magnetocassette / Un Interface Joystick / Un Joystick / Les divers câbles de branchement / Le programme EUREKA.

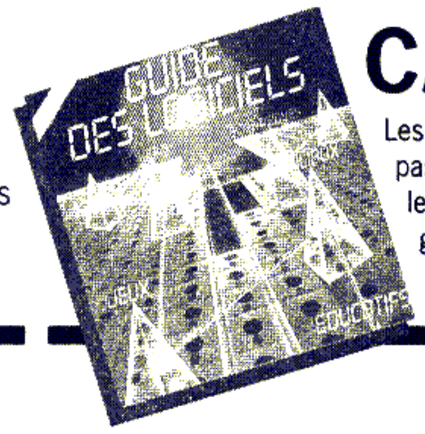


ensemble*
COMMODORE64
3.990 F

Un ordinateur Commodore 64 / Un interface Peritel / Un magnétocassette pour C64 / Les divers câbles de branchement (magnétophone, télé et secteur) / Un joystick / Le programme EUREKA.

Le concours n'ouvrant que le 1er décembref 1984, aucun jeu Eureka ne sera livré au public avant cette date. Les expéditions des programmes Eureka débuteront donc le 30 novembre 1984 et seront effectuées dans l'ordre d'arrivée des commandes. Les chèques seront encaissés 8 jours seulement avant la date d'expédition. Le matériel, lui, sera livré immédiatement.

*Les éléments de ces ensembles peuvent être acquis séparément : consulter le bon de commande ci-dessous ou notre catalogue VPC.



CADEAU

Les personnes ayant passé commande avant le 31.12.84. Recevront gratuitement le guide des logiciel 84.

BON DE COMMANDE

	PRIX EN FRANCS, TTC	QTÉ	Votre Commande en Francs
● ENSEMBLE COMMODORE PROMO EUREKA	3990	...	
COMMODORE 64 PAL SEUL	2790	...	
COMMODORE 64 PERITEL SEUL	3450	...	
COMMODORE SX 64 PORTABLE	8490	...	
UNITE DE DISQUETTES COMMODORE	3190	...	
LECTEUR DE CASSETTE POUR COMMODORE	400	...	
IMPRIMANTE COSMOS 80	2850	...	
INTERFACE CENTRONICS POUR IMPRIMANTE	240	...	
LOT DE 10 CASSETTES VIERGES	75	...	
BOITE DE 10 DISQUETTES VIERGES	190	...	
● ENSEMBLE SPECTRUM PROMOTION EUREKA	2890	...	
ORDINATEUR SPECTRUM 48K PAL	1890	...	
ORDINATEUR SPECTRUM 48K PERITEL	2220	...	
INTERFCE ZX1 SPECTRUM	745	...	
INTERFACE JOYSTICK	180	...	
MICRO-DRIVE SPECTRUM	745	...	
DOUBLE MICRODRIVE ROTRONICS	1990	...	
MICROCASSETTE POUR MICRODRIVE SINCLAIR	75	...	
LECTEUR DE CASSETTE	450	...	
IMPRIMANTE THERMIQUE ALPHACOM 32	920	...	
ROULEAU DE PAPIER THERMIQUE	30	...	
MONITEUR COULEUR CM14 FIDELITY	2750	...	
CABLE POUR BRANCHER UN COMMODORE 64 SUR CM 14	80	...	
CABLE POUR BRANCHER UN SPECTRUM SUR CM14	130	...	
MONITEUR N/V PHILIPS TP200	950	...	
JOYSTICK SPECTRAVIDEO	95	...	
JOYSTICK KRAFT	85	...	

Total de votre commande : _____ **F TTC.**

Signature :

Signature des parents
(Pour mineur)

Nom _____ Prénom _____

Adresse _____ Ville _____ Code postal _____

Ci-jointe la somme de F _____ TTC, par chèque bancaire à l'ordre de EUREKA INFORMATIQUE
39/41, rue Victor-Massé - 75009 PARIS.

Nom : _____

Adresse : _____

Code postal : |_|_|_|_|_|

Ville : _____

Je désire recevoir votre catalogue de vente par correspondance. Ci-joint 5 Francs en timbres-poste pour contribution aux frais d'expédition.

LIST
LE JOURNAL DES AMATEURS DE PROGRAMMATION

N° 5 - DÉCEMBRE 1984

SOMMAIRE

Ce numéro contient en encart, des bulletins d'abonnement paginés 11, 12, 77 et 78.

RÉDACTION-RÉALISATION
Directeur de la rédaction : Bernard Savonet
Rédacteur en chef : Jean Baptiste Comiti
Responsable de rubrique : Anne-Sophie Dreyfus
Conception graphique et secrétariat de rédaction : Eliane Gueylard
Assistante de rédaction : Maryse Gros
Administration : Marie-Hélène Muniz

Ont collaboré à ce numéro : *Olivier Arbey, Michel Arditti, Pierre Barnouin, François J. Bayard, Jean-Antoine Berro, Robin Bois, Jacques Boisgontier, Christian Boyer, Jean-Paul Carré, Thierry Chamoret, Raymonde Coudert, Robert Daguesse, Jacques Deconchat, Vincent Di Sanzo, Jean-Marie Donnat, Bruno Fiter, Philippe François, Florence Gautier-Louette, Pierre Ladislas Gedo, Max Hagenburger, Renée Koch, Jean-Christophe Krust, Jacques Labidurie, Jean-Pierre Lalevée, Bernard Lambey, Alain Lavenir, Thierry Lévy-Abégnoli, Marc Leygnac, Alain Mariatte, Pierrick Moigneau, Monti, Yvon Pérès, André Reeb-Gruber, Edouard Rencker, Pierre Ricard, Denis Sebbag, David Segard, Christophe Tavernier, Benoit Thonnart, André Warusfel.*

Illustrations : *Philippe Burel, C. Christ, Chimulus, Frapar, Bernard Helme, Fabien Lacaf, Alain Mangin, Pasty, Alain Prigent, Nestor Salas, Nicolas Spinga.*

ÉDITION-PUBLICITÉ-PROMOTION
Éditeur : Jean-Pierre Nizard
Éditeur-adjoint : Jean-Daniel Belfond
Administration : Maryse Marti, assistée d'Anne Stolkowski
Publicité : Béatrice Ginoux Defermon, assistée de Nadine Schops

VENTES
Diffusion NMPP : Béatrice Ginoux Defermon
Abonnements : Muriel Watremez
assistée de Cécilia Mollicone et Sylvie Trumel.

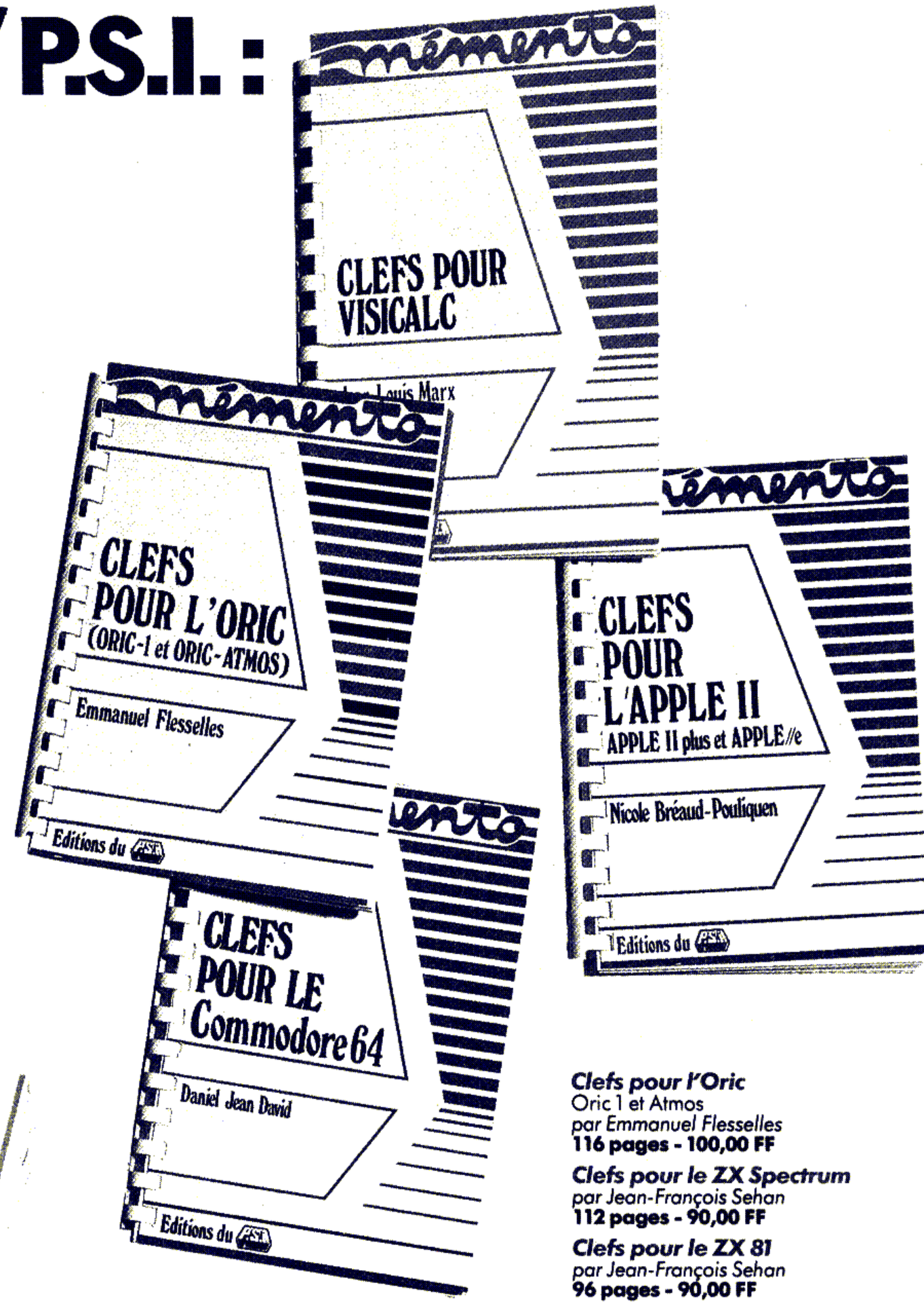
Directeur de la publication : Jean-Luc Verhoye

LIST est une publication du

5 place du Colonel Fabien - 75491 Paris Cédex 10
Téléphone : (1) 240 22 01 - Télex : LORDI 215 105 F

LES 'MÉMENTOS' P.S.I. : les clefs de votre ordinateur.

Des livres destinés à se trouver en permanence à côté de votre ordinateur individuel tandis que vous programmez?
Des livres qui renferment toutes les informations que vous avez besoin de retrouver rapidement: syntaxes des commandes, codes caractères, message d'erreur, codes machine, adresses utiles...?
Des livres qui vous donnent des "trucs" très précieux?
Ces livres sont les MEMENTOS P.S.I., pour une utilisation quotidienne de votre ordinateur.



NOUVEAU
Clefs pour Multiplan
par JL Marx et Alain Thibault
128 pages -100,00 FF

Clefs pour l'Apple II
par Nicole Bréaud Pouliquen
144 pages - 100,00 FF

Clefs pour le Commodore 64
par Daniel-Jean David
128 pages - 100,00 FF

Clefs pour Visicalc
par Jean-Louis Marx et Alain Thibault
104 pages - 100,00 FF

Clefs pour l'Oric
Oric 1 et Atmos
par Emmanuel Flesselles
116 pages - 100,00 FF

Clefs pour le ZX Spectrum
par Jean-François Sehan
112 pages - 90,00 FF

Clefs pour le ZX 81
par Jean-François Sehan
96 pages - 90,00 FF

Clefs pour le Vic
par Daniel-Jean David
120 pages - 90,00 FF

Le Basic de A à Z
par Jacques Boisgontier
176 pages - 110,00 FF

CP/ M mot par mot
par Yvon Dargery
112 pages - 90,00 FF

CHEZ VOTRE LIBRAIRE OU EN BOUTIQUE SPÉCIALISÉE

LM 12

P.S.I. DIFFUSION
BP 86 - 77402 Lagny-S/Marne Cedex
FRANCE
Téléphone (6) 006.44.35
P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Téléphone (2) 345.08.50
P.S.I. SUISSE
Case postale
Route neuve 1
1701 Fribourg
SUISSE
Tél. : (037) 23.18.28
CCP 17.56.84

Au Maroc
SMER DIFFUSION
3, rue Ghazza
Rabat
MAROC
Tél. : (7) 237.25

Au Canada
SCE Inc.
65, avenue Hillside
Montréal (Westmount)
Québec H321W1 - CANADA
Tél. : (514) 935.13.14

Table de conversions en Francs belges et Francs suisses
90 FF = 695 FB - 28.40 FS
100 FF = 770 FB - 31,50 FS
110 FF = 850 FB - 36.40 FS

Envoyer ce bon accompagné de votre reglement a P.S.I. DIFFUSION ou, pour la Belgique et le Luxembourg à P.S.I. BENELUX ou, pour la Suisse à P.S.I. SUISSE

DESIGNATION	NOMBRE	PRIX
	TOTAL	

par avion : ajouter 8 FF (75 FB) par livre
Pour la SUISSE frais de port sur tout envoi : 1.50 FS

Paiement par cheque joint ☐ Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement) paiement supérieur à 50 FF

Signature (obligatoire pour paiement par carte de crédit)

AGRAPH

☐ Je désire recevoir le catalogue P.S.I.

N° _____ Date d'expiration _____

NOM _____ PRENOM _____

rue _____ n° _____

Code postal _____ Ville _____

ISSN 0183 - 570X

L'ORDINATEUR INDIVIDUEL

LE TOUR DE L'APPLE 2c

LANGAGES : DÉPASSEZ VOTRE BASIC

LES ATOUTS POUR MIEUX PROGRAMMER

IBM FAIT SON AUTOCRITIQUE

PROFESSIONNEL

Data General One
Ericsson PC
Publi-base pour IBM PC
FileVision pour Macintosh

PERSONNEL

Commodore Plus/4
Einstein
Lecteur de disquettes Oric

RÉPERTOIRE 83-84 :
tous les essais, programmes, dossiers de L'ORDINATEUR INDIVIDUEL

EN VENTE DANS TOUS LES KIOSQUES

Le magazine de l'informatique pour tous - Décembre 1984 - N° 65
M 2946 - 65 - 23 F
Belgique : 156 FB - Suisse : 7,5 FS - Canada : 2,95 $ CAN

Pour l'assembleur
NE VOUS DISPERSEZ PAS !

Si vous maîtrisez déjà un ou plusieurs langages évolués, la connaissance du langage machine et de l'assembleur vous donnera un moyen efficace d'améliorer les performances de vos programmes.
P.S.I. vous propose :

des ouvrages dans lesquels vous trouverez :

- les particularités des langages machine et leurs avantages (vitesse d'exécution, occupation mémoire...) ;
- des détails et des exercices sur les codes machines, les jeux d'instructions avec leurs modes d'adressage, etc. La méthode adoptée par les auteurs de ces ouvrages consiste à transposer progressivement le langage Basic en langage machine ("La pratique de l'Apple" - tome 3 ; "L'assembleur de l'Oric",...)

les "Clefs pour..." qui seront votre outil de référence pour retrouver rapidement :

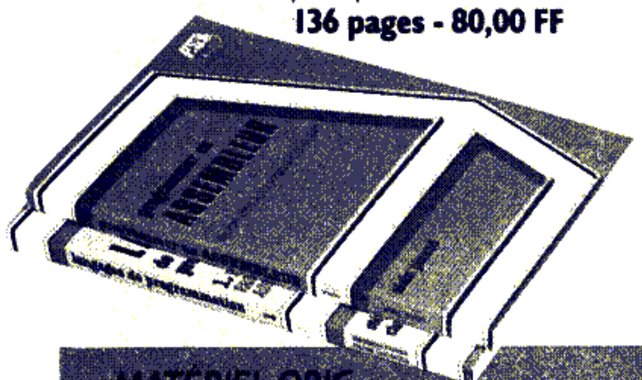
- les registres internes (6502, Z80,...) ;
- les jeux d'instructions ;
- le schéma des modes d'adressages ;
- le tableau de désassemblage.

Une introduction au langage machine et à son frère l'Assembleur illustrée d'exemples et d'exercices, indispensable pour mieux en comprendre les finesses.

Programmer en assembleur
par Alain Pinaud
144 pages - 90,00 FF

Quels sont les avantages et limites du Fortran, du LSE, du Pascal, du Cobol, de l'Assembleur et qu'est ce qui caractérise tous ces langages ? Des réponses à toutes ces questions dans le livre :

Langages de programmation
par Stéphane Berche et Claude Lhermitte
136 pages - 80,00 FF

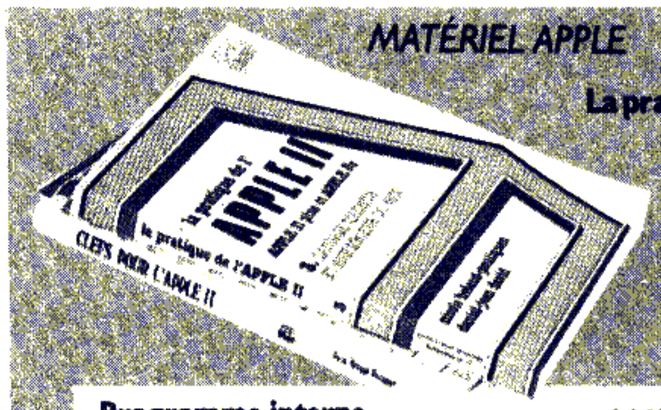


MATÉRIEL ORIC

NOUVEAU **L'assembleur de l'Oric et Oric Atmos**
Programmation en langage machine
par Marcel Henrot
160 pages - 90,00 FF

Clefs pour l'Oric - Oric 1 et Atmos
par Emmanuel Fiesselles
124 pages - 100,00 FF

MATÉRIEL APPLE



La pratique de l'Apple II tome 3
par Nicole bréaud-Pouliquen et Daniel-Jean David
176 pages - 90,00 FF

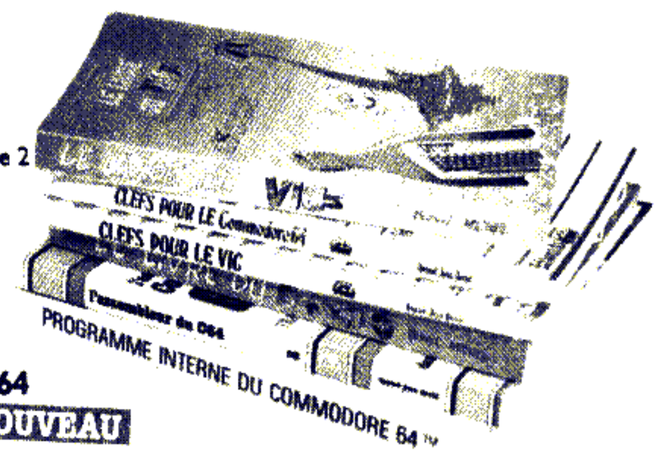
Clefs pour l'Apple II
par Nicole Bréaud-Pouliquen
144 pages - 100,00 FF

MATÉRIEL COMMODORE

Programme interne du Commodore 64
par Milton B. Bathurst
248 pages - 130,00 FF

L'assembleur du Commodore 64
Pratique du Commodore 64 - tome 2
par Daniel-Jean David
208 pages - 100,00 FF

Le livre du 64
par Benoît Michel
304 pages - 120,00 FF



Clefs pour le Commodore 64
par Daniel-Jean David
NOUVEAU
128 pages - 100,00 FF

Le livre du Vic
par Benoît Michel
248 pages - 110,00 FF

Clefs pour le Vic
par Daniel-Jean David
120 pages - 90,00 FF

MATÉRIEL TRS

Clefs pour le TRS-80 - tome 1
par Rémy Pineau
192 pages - 120,00 FF

MATÉRIEL SINCLAIR

ZX 81
La pratique du ZX-81 - tome 2
par Marcel Henrot
152 pages - 90,00 FF

Le petit livre du ZX-81
par Trevor Toms
136 pages - 80,00 FF

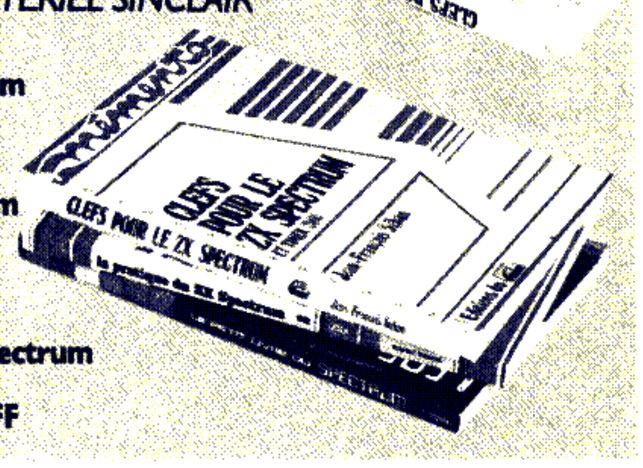
Clefs pour le ZX-81
par Jean-François Sehan
96 pages - 90,00 FF

MATÉRIEL SINCLAIR

ZX SPECTRUM
Le petit livre du Spectrum
par Trevor Toms
168 pages - 90,00 FF

La pratique du ZX Spectrum
tome 2
par Marcel Henrot
168 pages - 90,00 FF

Clefs pour le ZX Spectrum
par Jean-François Sehan
112 pages - 90,00 FF



EGALEMENT CHEZ VOTRE LIBRAIRE ET EN BOUTIQUE SPECIALISEE

LIL 12

P.S.I. DIFFUSION
BP 86 - 77402 Lagny-S/Marne Cedex
FRANCE
Telephone (6) 006.44.35
P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Telephone (2) 345.08.50
P.S.I. SUISSE
Case postale
Route neuve 1
1701 Fribourg
SUISSE
Tel. : (037) 23.18.28
CCP 17.56.84

Envoyer ce bon accompagné de votre règlement à P.S.I. DIFFUSION ou pour la Belgique et le Luxembourg à P.S.I. BENELUX ou pour la Suisse à P.S.I. SUISSE

DESIGNATION	NOMBRE	PRIX
	TOTAL	

par avion : ajouter 8 FF (75 FB) par livre
Pour la SUISSE frais de port sur tout envoi : 1,50 FS

Signature (obligatoire pour paiement par carte de crédit)

Paiement par chèque joint — Paiement en FF par carte bleue VISA (à P.S.I. DIFFUSION uniquement) paiement supérieur à 50 FF

☐ Je désire recevoir le catalogue P.S.I. gratuit ;

N° _____ Date d'expiration _____

NOM _____ PRENOM _____

rue _____ n° _____

Code postal _____ Ville _____

Au Maroc
SMER DIFFUSION
3, rue Ghazza
Rabat
MAROC
Tél. : (7) 237.25

Au Canada
SCE Inc.
65, avenue Hillside
Montréal (Westmount)
Québec H321W1 - CANADA
Tél. : (514) 935.13.14

Table de conversions en Francs belges et Francs suisses		
35 FF =	250 FB	12,20 FS
60 FF =	432 FB	19,10 FS
70 FF =	504 FB	22,20 FS
80 FF =	576 FB	25,30 FS
85 FF =	612 FB	26,90 FS
90 FF =	648 FB	28,40 FS
95 FF =	684 FB	29,90 FS
100 FF =	720 FB	31,50 FS
105 FF =	756 FB	33,00 FS
110 FF =	792 FB	34,60 FS
115 FF =	828 FB	36,10 FS
120 FF =	864 FB	37,80 FS
125 FF =	900 FB	39,10 FS
130 FF =	936 FB	40,80 FS
135 FF =	972 FB	42,20 FS
140 FF =	1.008 FB	43,70 FS
145 FF =	1.044 FB	45,20 FS
150 FF =	1.080 FB	46,70 FS
160 FF =	1.138 FB	49,70 FS
170 FF =	1.207 FB	52,60 FS
180 FF =	1.278 FB	55,60 FS
190 FF =	1.349 FB	58,60 FS
195 FF =	1.385 FB	60,00 FS
DISQUETTES		
195 FF =	1.500 FB	61,50 FS
295 FF =	2.275 FB	90,70 FS

AGRAPH

VOTRE spécial ORDINATEUR

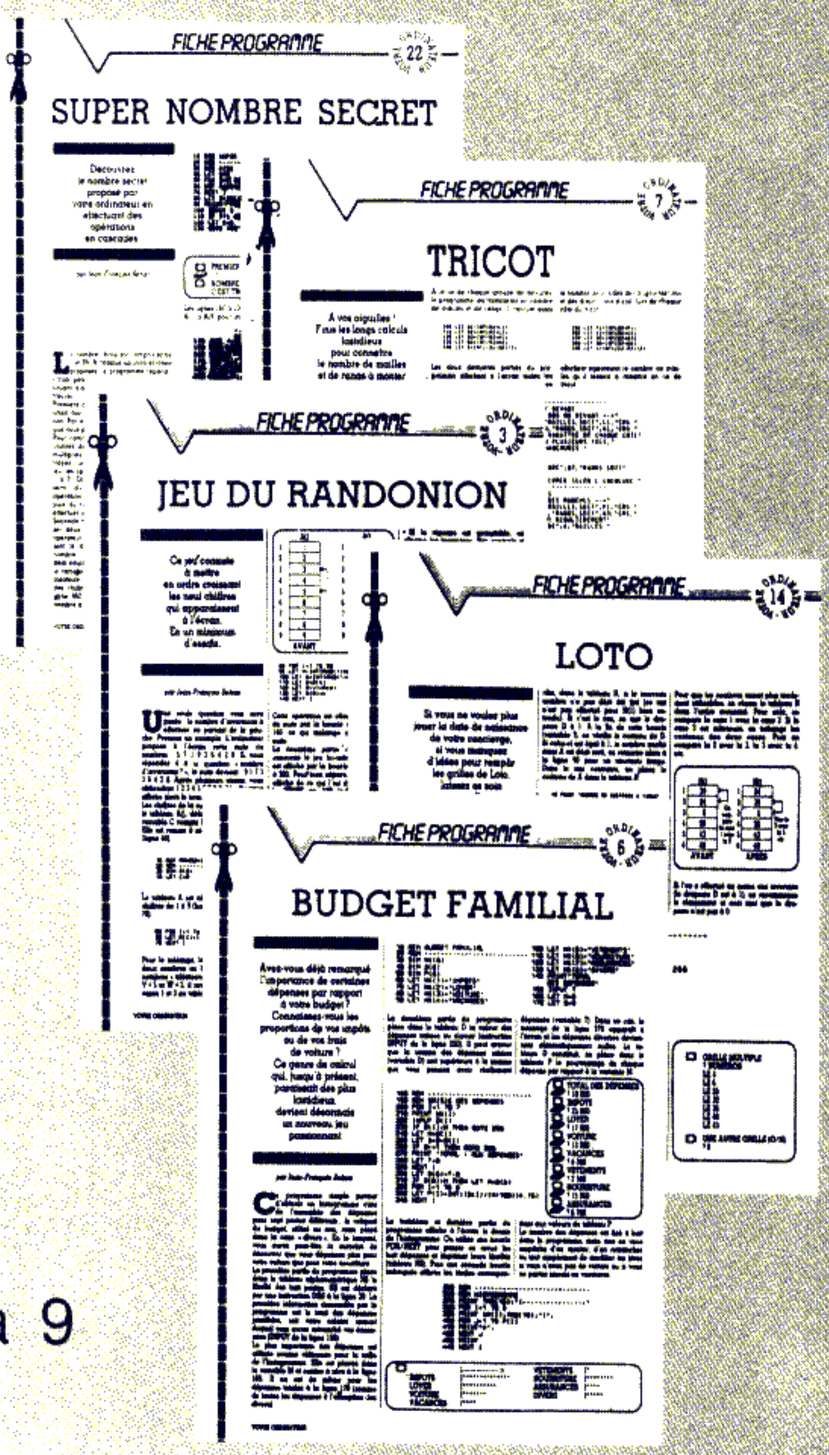
LE MAGAZINE DE L'INFORMATIQUE A LA MAISON

ISSN 0752-2363

HORS SERIE 25 F

le BASIC par la PRATIQUE

40 FICHES PROGRAMMES UNIVERSELLES



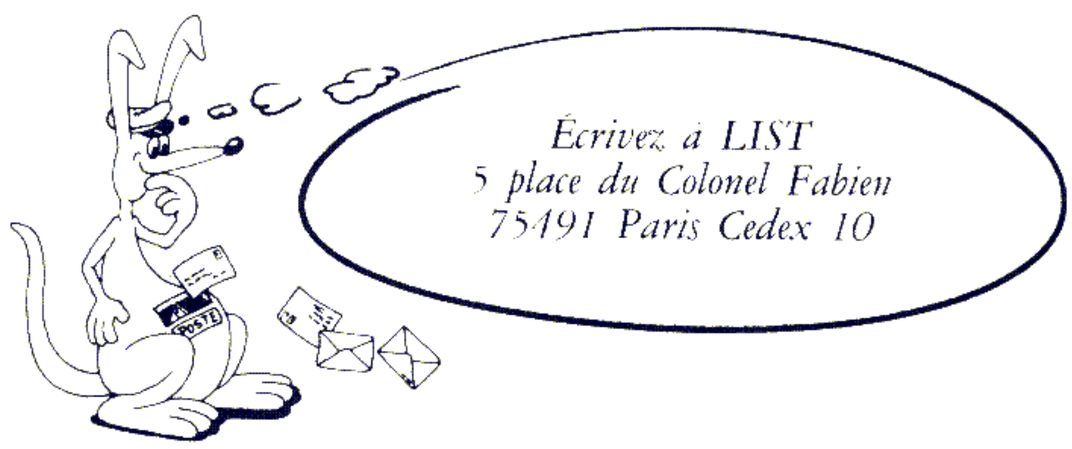
Recueil des fiches parues dans Votre Ordinateur n° 1 à 9

JEU **ENSEIGNEMENT** **VIE PRATIQUE** **MUSIQUE** DESSIN

Sur Alice - Apple - Atari - Atmos - Oric 1 - Commodore 64 - Vic 20 - Hector II HR
Thomson MO 5 - TO 7 - TO 7-70 - MSX - Yeno SC 3000 - ZX Spectrum et ZX 81.

en vente chez votre marchand de journaux

A VOS CLAVIERS



Comme quoi...

DEPUIS le premier numéro de *LIST*, mes craintes se sont (hélas) confirmées : avec l'élargissement de votre horizon, vous ne faites que rejoindre d'autres revues vulgarisant l'informatique individuelle.

Ce que je déplore, c'est que les ordinateurs de table se substituent totalement aux ordinateurs de poche. Les gros ordinateurs et leurs langages ont déjà un certain nombre de revues à leur disposition.

Toutefois, je sais que les programmes sont envoyés par les lecteurs, aussi faudrait-il, peut-être, que les fans des ordinateurs de poche se réveillent...

Néanmoins, je m'intéresse à votre revue et je pense même m'y abonner. Comme quoi !

Xavier PADRONA
83 La Garde

■ *Dans la mesure du possible,* LIST *accueille dans ses colonnes tout ce qui touche à la programmation, tant sur les ordinateurs de poche que sur les ordinateurs de table. A chacun de faire les adaptations nécessaires. Ainsi, un programme écrit dans un Basic standard devrait pouvoir, moyennant quelques transformations, tourner sur les différents types d'ordinateurs (à la taille de la mémoire près). Cela dit, comme vous l'avez certainement remarqué, nous ne publions pas seulement des listes de programme...*

Le bon Basic du BBC

AYANT lu le numéro 4 de *LIST*, je me permets de vous livrer mes réactions.

J'apprécie beaucoup que vous sortiez des sempiternelles listes de programmes pour parler d'autres choses : le langage Pascal, compilateur et interpréteur, récursivité, sans oublier dix tests permettant de se faire une « première idée de sa machine ».

A propos de Basic récursif, je trouve injuste que vous pénalisiez la magnifique machine qu'est le BBC en ne mentionnant pas que son Basic est récursif avec procédures (possibilité de variables locales) et fonctions définies par l'utilisateur, le tout dans la version de base.

Certains de vos articles se terminent par une bibliographie. Et c'est tant mieux.

Robert AURELLE
38 St Martin d'Hères

■ *Tout d'abord, merci de nous donner vos impressions sur notre journal.*

Dans l'article consacré au Basic récursif, publié dans LIST *4, nous n'avons pas, en effet, mentionné le BBC. Mais toutes les qualités de cette machine (rapidité, vocabulaire riche, récursivité, procédures, variables, etc.) ont été remarquées dans l'essai de son Basic, présenté dans* LIST *3 (pages 30 à 32). Nous ne pouvions tout de même pas nous répéter...*

Sous un nouvel angle

DANS l'article « Conversions d'angles » publié dans *LIST* 2 (page 97), on propose une formule pour arcsinus, asn (a) = atn ($a/\sqrt{1-a^2}$), qui n'aime pas a = 1 (car cela entraîne une division par zéro) et une formule pour arccosinus, acs (b) = atn ($\sqrt{1-b^2}/b$), qui n'aime pas b = 0.

Voici deux autres formules qui traitent toutes les valeurs de l'argument x comprises entre −1 et 1, bornes exclues. Ainsi, pour $-1 < x < 1$:

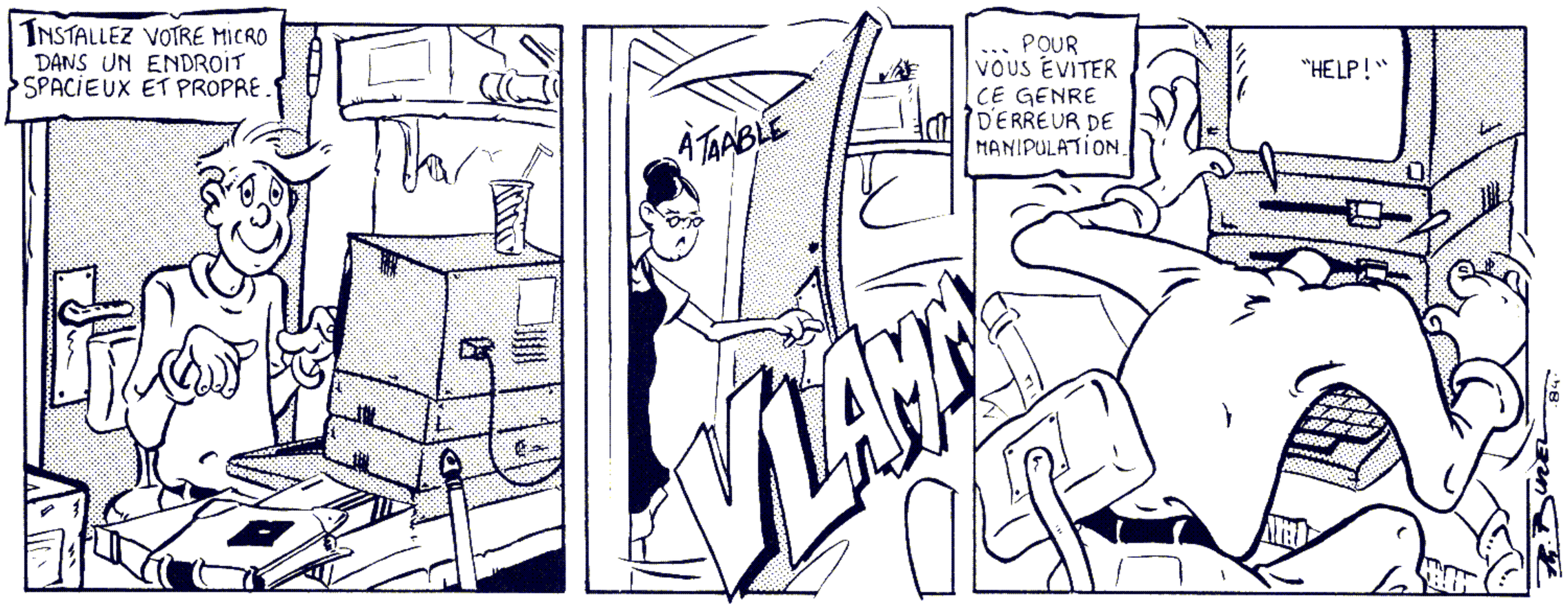
```
asn(x) = 2*atn(x/(1 + SQR(1 - x*x)))
acs(x) = 2*atn(1) - asn(x)
```

L'utilisation de ''2*atn(1)'' (égal à π/2 radians ou 90 degrés ou 100 grades) rend les formules indépendantes des unités d'angle employées par une machine.

Philippe KENT
Lausanne (Suisse)

Midi à sa porte

BRAVO à *LIST* qui me parvient tous les mois, ici en Angola, grâce aux bons soins de mon épouse.



Un seul point de critique : pourquoi parle-t-on si peu, sinon pas du tout du PB-700 ?

Alphonse KEMPF
Luanda (Angola)

■ *Vous êtes nombreux à nous écrire pour nous demander de parler plus, beaucoup plus, d'une machine particulière (celle que vous pratiquez) et moins des autres. C'est naturel. Mais vous comprenez bien que votre ordinateur n'est pas le seul modèle commercialisé.*

D'ailleurs, quand nous testons un ordinateur, nous testons surtout son langage de programmation : Basic d'origine, Forth, Pascal, Assembleur disponibles sous forme de logiciels, etc. Et les programmes que nous proposons valent par leur résultat, mais aussi et surtout par la manière de parvenir à leur conception.

Dans ce numéro justement, vous trouverez un programme présenté sur PB-700 (pages 44 et 45), mais dont les explications devraient permettre une adaptation à d'autres matériels.

Et si nous tenons à présenter les langages ou les programmes sur des matériels variés, c'est pour que, bon an, mal an, chacun s'y retrouve mieux et puisse comparer...

De Péritel à Secam

COMME je suis beaucoup trop jeune (j'ai 12 ans) pour pouvoir m'acheter une télé couleur avec prise Péritel, je vous écris pour savoir s'il existe un appareil quelconque qui transforme Secam en Péritel.

J'aimerais acheter un ZX Spectrum mais malheureusement il se trouve que ma télévision est équipée d'une prise Secam. Et le Spectrum ne peut pas se brancher dessus.

Longue vie à *LIST*.

Stéphane PETIT
20 Ajaccio

■ *La liaison entre la prise Péritel d'un ordinateur (ou d'un jeu vidéo) et la prise Secam de la télévision est possible grâce à certaines interfaces.*

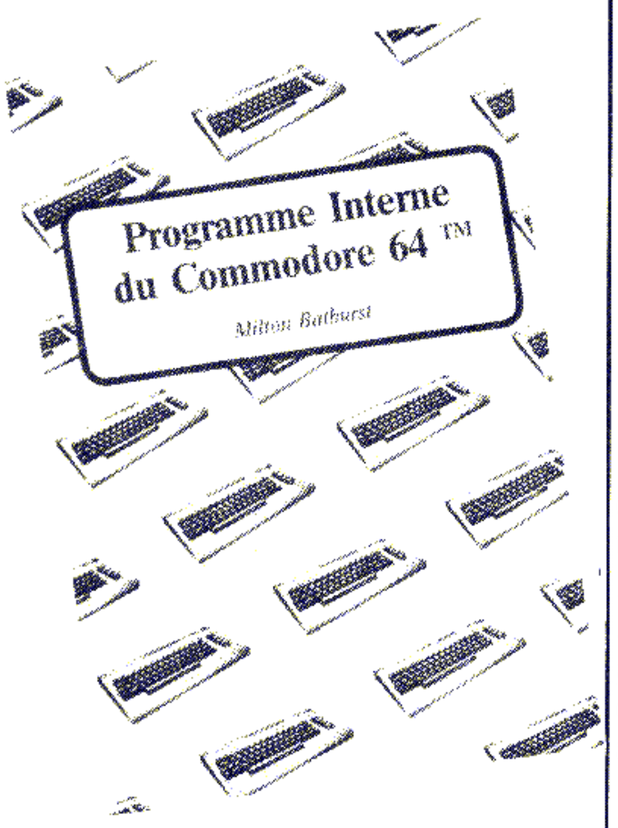
Nous savons que vous pouvez en trouver chez Video Match, distributeur national exclusif des productions de la Compagnie Générale de Vidéotechnique. Peut-être pourrez-vous, par leur intermédiaire, trouver un distributeur plus près de chez vous.

Video Match
8-10 rue Alexandre Dumas
67200 Strasbourg
Tél. : (88) 28 21 09

INDEX DES ANNONCEURS

LA GAZETTE DE LIST

UN LIVRE



Programme interne du Commodore 64
Milton Bathurst
Editions Data Cap
(PSI diffusion)
Belgique, 1984
Broché, 250 pages
Prix : 130 FF

TOUT programmateur chevronné sur Commodore 64, amateur de langage-machine, se doit de faire figurer ce livre en bonne place dans sa bibliothèque informatique. Un plan de lecture simple ouvre à une mine de trésors que peuvent creuser tous ceux qui veulent connaître à fond le fonctionnement logiciel de leur machine. Il propose en effet le listage complet de la MEM Basic, désassemblée et commentée, depuis l'adresse $A000 jusqu'à l'adresse $FFFF. A ces 16 Koctets analysés avec rigueur s'ajoutent une table des adresses de la page zéro et une table de références croisées.

Deux parties équilibrées correspondent au découpage en deux blocs de la MEM Basic du C.64. Le tout est rédigé de façon un peu sèche ; il est vrai qu'il ne s'agit pas d'un roman. Un livre à déconseiller aux débutants, mais qui rendra d'inestimables services aux autres.

JPL ■

Code Writer, nouveau générateur de programmes

IMAGINEZ les possibilités qui vous sont offertes par un programme qui écrit d'autres programmes. Les concepteurs de Code Writer ne s'y sont pas trompés. La programmation fait des adeptes et n'est cependant pas toujours une sinécure.

Destiné à l'IBM-PC, Victor, Apple et Commodore 64, Code Writer permettra de créer des programmes de gestion de fichiers avec saisie, consultation, mise à jour des enregistrements et des programmes d'édition.

Deux disquettes sont fournies. La première concerne la création de fichiers, la saisie et la mise à jour et permet de disposer de fonctions telles que *tests de validité, formatage automatique de zones*. La seconde concerne les états et les générateurs de menus. Jusque 21 menus et sous-menus peuvent être créés et liés entre eux. On pourra ensuite les modifier, remplacer des ordres ou en changer l'ordre. De plus des routines du type retour au menu principal ou retour au Basic sont intégrables directement.

Et pour finir, la **Sofitec**, distributrice de Code Writer en France, précise que tous les programmes générés par leur logiciel peuvent être librement vendus, sous réserve toutefois d'en indiquer l'origine sur les disquettes et l'écran. ■

Extension pour programmeurs en mal d'idée

MANQUE d'idée ou ordinateur esseulé ? La société **Sélia** devrait ouvrir de nouveaux horizons et permettre des connexions amusantes du type chaîne hi-fi, chaudière, ou aspirateur. Sélia distribue en effet une carte d'extension entrée/sortie, la PB 500 qui offrira aux propriétaires de Dragon les joies et les délires du contrôle informatique des appareils électroménagers et de loisir.

La PB 500 se branche directement sur le port d'extension et ne nécessite aucun ajustage, tous les canaux étant déjà étalonnés. Cette carte comprend en outre 8 entrées/sorties numériques, 4 entrées analogiques pour la mesure directe de température. Chaque borne d'entrée/sortie dispose d'une LED lumineuse de signalisation.

Pour les amateurs, Sélia précise que cette carte peut être utilisée pour la commande des circuits de trains miniatures. ■

UNE CASSETTE

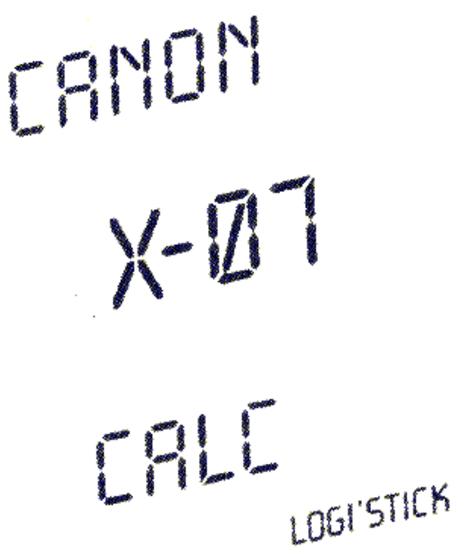
Calc
pour le Canon X-07
Logi'Stick
Distribuée par DDI
Prix : environ 130 FF

Le *Calc* de **Logi'Stick** est destiné à concrétiser la manipulation de tableaux chiffrés sur le X-07. Certes, il ne s'agit pas de transformer ce petit ordinateur en monstre, mais de tirer parti d'une machine très astucieuse — et modulaire.

La présentation du *Calc* est tout à fait classique : l'utilisateur peut créer un tableau de 66 à 324 cases en fonction de la MEV disponible (de 8 Ko à 24 Ko) et les répartir à son gré. L'écran du Canon joue le rôle d'une fenêtre qui présente simultanément 3 cases de 16 caractères. Les 4 curseurs, à droite de l'écran, permettent de déplacer cette fenêtre sur toute l'étendue du tableau.

Chaque case accepte jusqu'à 16 caractères, dont 7 signes d'opérations s'il s'agit d'une formule. A noter que le programme possède son propre interpréteur, et que les calculs formulés s'effectuent de gauche à droite sans priorité. L'utilisateur gère donc en fait un double tableau incluant données et formules, d'où une certaine perte de rapidité lors de l'interprétation des calculs ; mais on s'y habitue vite car à l'usage, l'ensemble s'avère efficace. Une bonne gestion des erreurs évite les accidents de première heure et cet atout n'est pas négligeable quand on utilise les fonctions élaborées du tableur.

Au total, onze fonctions sont disponibles. Elles permettent, par exemple, la transposition d'une case dans la ligne ou la colonne choisie ou la modification du contenu de la case de départ en fonction de la case d'arrivée.



On peut regretter l'absence d'exemples précis dans le mode d'emploi (14 pages trop légères). Hélas, on commence à le savoir : le plus souvent, les notices ne sont pas à la hauteur des produits — et cela vaut aussi bien pour les matériels que pour les logiciels.

La sauvegarde des données est habituellement une petite corvée lorsqu'elle s'effectue par l'intermédiaire du lent cassettophone. Le X-07 ne fait pas exception à la règle. Mais le logiciel propose une option séduisante : la sauvegarde sur carte mémoire de MEV alimentée par une pile autonome. Pour charger un nouveau tableau, il suffit de changer de carte en deux temps et trois mouvements.

Pour terminer, précisons qu'une routine d'impression classique permet d'utiliser l'imprimante X-710 pour conserver une trace écrite des modifications effectuées, ainsi que des résultats obtenus.

XB ■

Vidéo-clip Sord

MARIAGE de raison, ou d'argent, vidéo-clip et ordinateurs font bon ménage. Après **Coleco** et **Eureka, Sord** annonce à son tour une cassette vidéo pour son modèle portable l'IS 11. Elle dure 60 minutes, est aux normes VHS SECAM et, si elle ne vante pas les mérites inoubliables de l'appareil, elle en explique réellement le fonctionnement. Mise en marche, création de tableaux, formatage de colonnes, initialisation d'une cassette et programmation, tout est expliqué en vidéo.

La bande fait donc office en quelque sorte de manuel d'emploi et préfigure l'une des formes à venir de l'aide à la programmation. Seul inconvénient, il faut bien entendu un magnétoscope et la cassette coûte quand même 370 FF. ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Initiation au Forth
SEFI
Editions Cédic/Nathan
Paris, 1984
Broché, 166 pages
Prix : 95 FF

Forth pour Micros
Jean-Marie De Geeter
Editions Eyrolles
Paris, 1984
Broché, 180 pages
Prix : 90 FF

Forth, manuel d'application
Martin S. Ewing
Traduit par Benoît Berger
Editions Masson
Paris, 1984
Broché, 120 pages
Prix : 85 FF

Introduction au ZX Forth
Marc Petreman et
Michel Rousseau
Editions Eyrolles
Paris, 1984
Broché, 130 pages
Prix : 85 FF

Programmer le Pascal
André Bouckaert
Editions Marabout
Paris, 1984
Broché, 352 pages
Prix : 30 FF

La programmation des jeux de réflexion
Algorithmes en Pascal
Louis Jardonnet
Editions PSI
Lagny, 1984
Broché, 196 pages
Prix : 105 FF

Initiation à Logo
Doris Avram et
Michèle Weidenfeld
Editions Cédic/Nathan
Paris, 1984
Broché, 160 pages
Prix : 85 FF

LA GAZETTE DE LIST

Une imprimante pour Alice

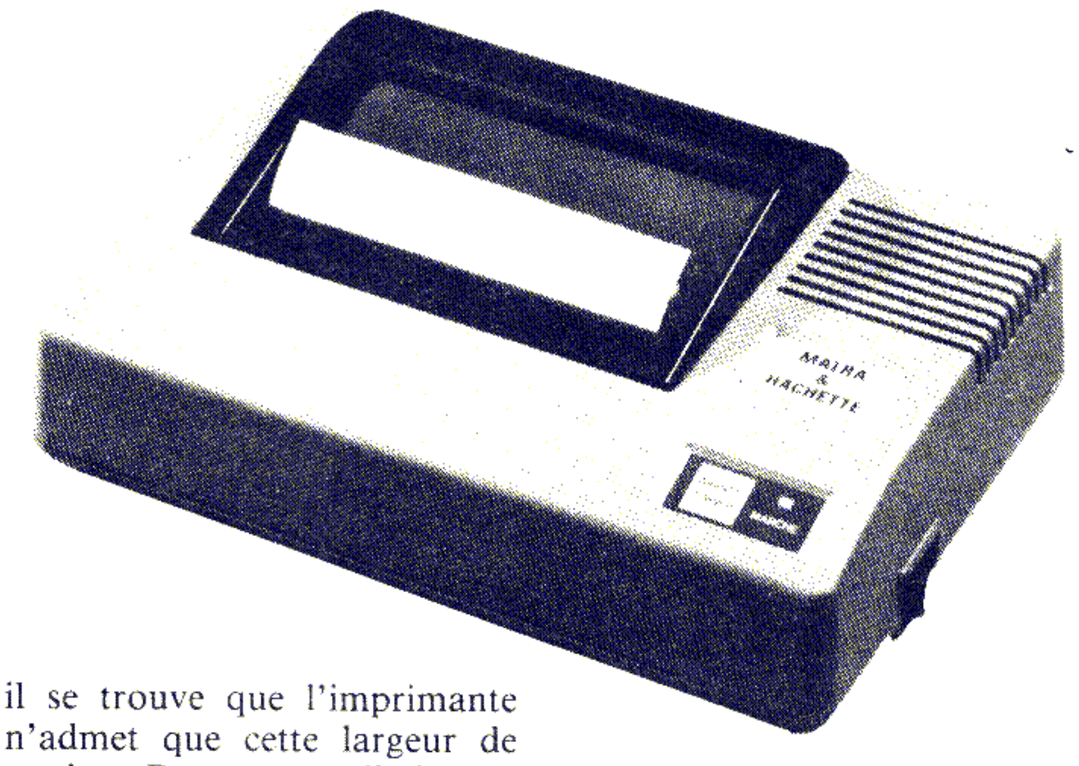
ALICE s'appelle toujours Alice. Et elle garde sa robe rouge. Après une première enfance un peu laborieuse, voilà qu'elle réapparaît. Qu'elles réapparaissent, devrais-je dire, puisqu'il y a désormais trois modèles en lice. La petite Alice d'origine, un peu à l'étroit dans ses 4 Ko, dispose maintenant d'une extension-mémoire, qu'elle partage d'ailleurs avec sa grande sœur Alice du coffret, dite encore Alice 32. Et il y a aussi Alice la grande, la plus sérieuse, Alice 90, le haut de gamme, quoi. Si Alice 32 et Alice 90 semblent compatibles côté logiciel, il n'en est pas tout à fait de même pour Alice 4 Ko : le Basic est bien le même, mais la gestion vidéo est différente, et les amateurs de PEEK et de POKE auront quelques surprises en essayant de passer de l'une à l'autre. Par contre, et c'est heureux, il y a une bonne compatibilité au niveau de la nouvelle imprimante.

D'apparence agréable, habillée elle aussi de rouge, cette imprimante thermique peut fonctionner en 32 colonnes (les 3 Alice disposent au moins d'un affichage de 16 lignes de 32 colonnes) et elle offre deux modes de fonctionnement accessibles par logiciel : le mode standard (32 caractères par ligne) et le mode élargi (16 caractères par ligne).

Les connexions sont simples : une prise pour se raccorder au secteur (le transformateur est incorporé), un cordon DIN-DIN pour se relier à la prise prévue à cet effet sur Alice, c'est une prise 4 broches. Un fil pour la détection de la porteuse (repéré CD), un fil pour la réception des données (repéré RD), un fil de masse (repéré GND), et un fil pour la transmission des données (repéré TD). La transmission est faite à 600 bauds, avec un bit de départ, 7 bits de données, 2 bits d'arrêt, et un retour-chariot automatique en fin de ligne. Une possibilité de test est incorporée dans l'imprimante : il suffit de maintenir le bouton d'avance-papier appuyé pendant la mise en route de l'appareil. Le jeu des caractères disponibles (non graphiques) est alors affiché.

Un voyant rouge signale la mise sous tension de l'appareil, qui comporte un interrupteur général sur le côté droit. Aucun réglage de contraste n'est accessible, et il n'y a pas de levier pour faciliter la mise en place ou le retrait du papier. Un couvercle rabattable, transparent, protège le rouleau, et le papier pourra facilement être déchiré grâce à une rangée de petites dentures ménagées sur le couvercle.

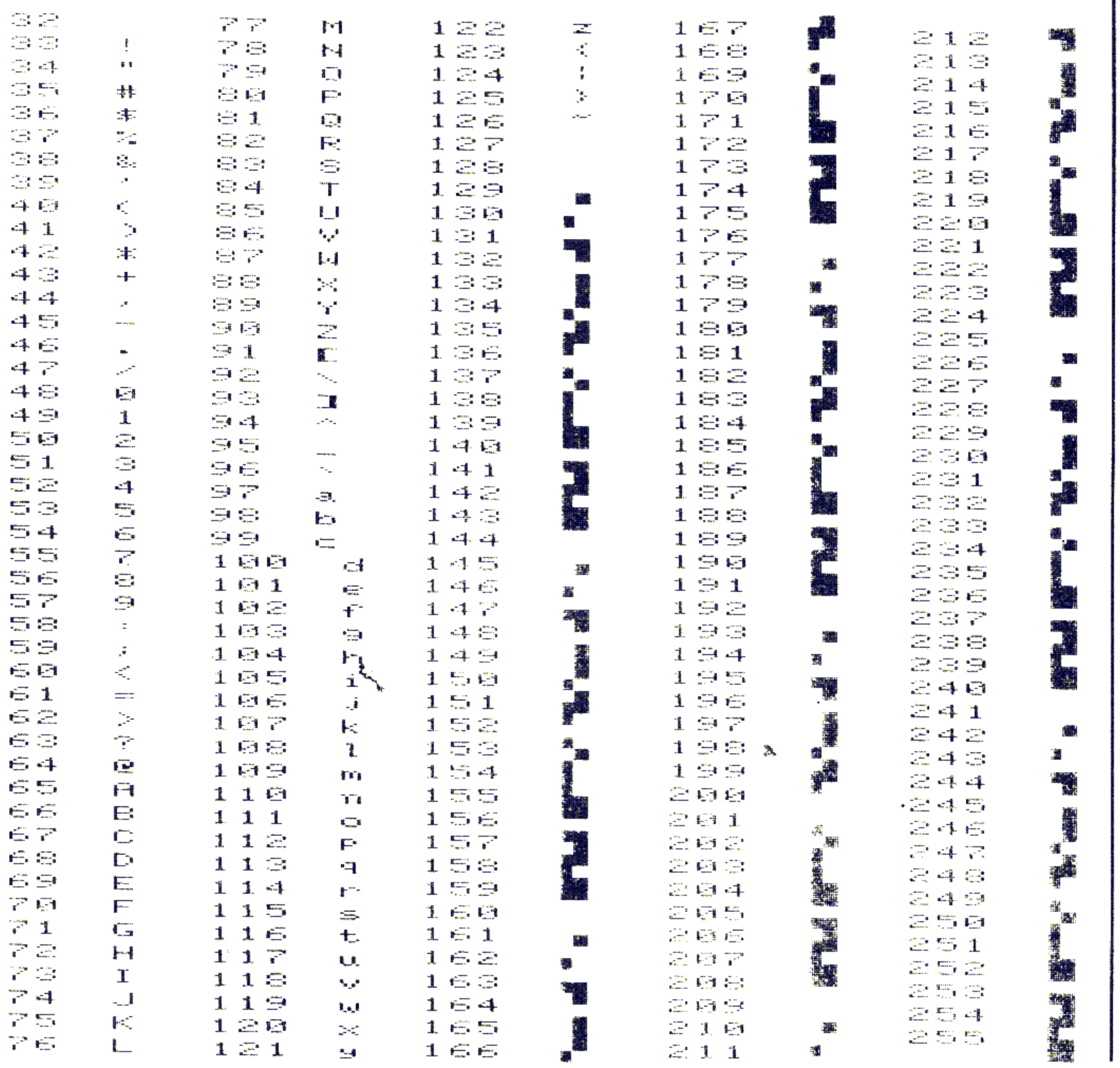
L'appareil dont je disposais me fournissait *au départ* une impression très pâlotte, et j'ai voulu essayer des papiers thermiques provenant d'autres machines (TO7, Alphacom...). Et là, ô surprise : aucun de ces papiers ne passe. Le papier thermique prévu pour Alice est en effet moins large d'environ 2 mm, et



il se trouve que l'imprimante n'admet que cette largeur de papier. Dommage : il faudra donc faire attention en achetant ; et bien préciser la largeur exacte du papier désiré (en principe 10,5 cm).

En désespoir de cause, je me suis résolu à ouvrir l'appareil pour voir s'il n'y avait pas un réglage de contraste interne. Rassurez-vous, c'est facile : 2 vis à enlever, et le couvercle se déboîte sans trop de mal (il faudra tout de même éviter de forcer pour ne pas briser les deux pattes arrière de maintien). L'intérieur est assez dépouillé, et la fabrication sérieuse. Un seul potentiomètre accessible : c'est

En mode élargi, les caractères correspondant aux codes 32 à 255



précisément celui-là qui règle le contraste. Ouf ! On referme soigneusement, et on rebranche. Ça marche.

L'impression s'effectue normalement sur 32 colonnes, et les caractères graphiques incorporés dans les PRINT ou dans des variables alphanumériques sont convenablement reproduits. On passe en 16 colonnes en tapant LPRINT CHR$(27) + CHR$(14), et on repasse en 32 colonnes en tapant LPRINT CHR$(27) + CHR$(15). Il n'y a pas de fonction de type COPY, permettant d'avoir directement une copie d'écran, mais il sera facile de l'obtenir par programme, tout au moins en mode texte 32 colonnes, puisque les caractères graphiques peuvent être reproduits. Evidemment, il ne faudra pas envisager de faire du traitement de texte intensif avec une telle machine (et d'ailleurs le prix assez élevé du papier thermique en dissuaderait vite), mais les possibilités sont largement suffisantes pour conserver une trace écrite de ses programmes, sous forme de listes où l'on pourra utiliser indifféremment des majuscules ou des minuscules. Le prix de vente (environ 1 095 FF) est très raisonnable. Que demander de plus à une imprimante thermique, par ailleurs assez rapide, très silencieuse et facile d'emploi ?

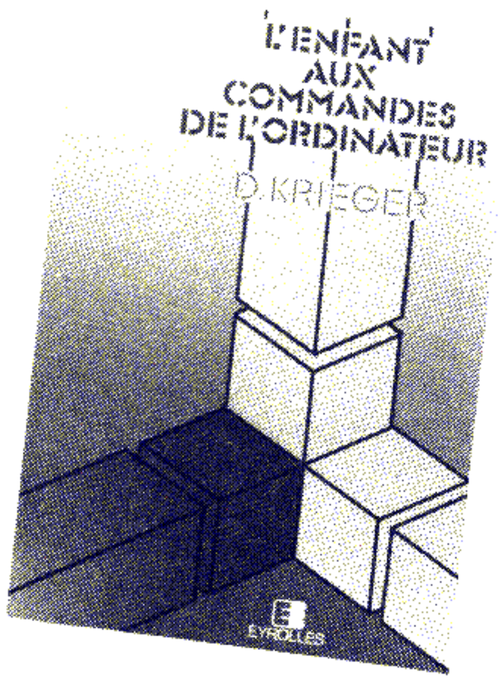
JD ■

Interface multi-micro

POUR ceux qui souhaitent utiliser leurs machines dans un circuit « ouvert », **Kap** propose des extensions entrée/sortie modulables pouvant accepter jusqu'à 32 bornes. Au menu des applications : réalisation d'automates, enregistrement de mesures, contrôle d'accès et systèmes

UN LIVRE

L'enfant aux commandes de l'ordinateur
Denis Krieger
Editions Eyrolles
Paris, 1984
Broché, 128 pages
Prix : 79 FF



VOICI un des trop rares ouvrages sur un thème pourtant très important : où en est l'analyse des rapports enfants-ordinateurs ? Si l'on peut penser que ces questions ne concernent que les parents, les enseignants et les enfants eux-mêmes, on ne doit pas oublier qu'elles vont conditionner la façon dont l'informatique va évoluer dans toute notre société.

Dans la première partie, l'auteur décrit et analyse plusieurs expériences informatiques, leur contenu pédagogique et les conclusions auxquelles elles ont permis d'aboutir. Cet exposé (parfois très « pédago ») trouve son intérêt, à notre sens, en ce qu'il offre une base de réflexion à tous ceux qui s'intéressent aux incidences de l'introduction de l'informatique en milieu scolaire.

La seconde partie de l'ouvrage propose un ensemble de 22 programmes utilisables en enseignement, avec listes pour Spectrum, commentaires et objectif pédagogique.

La conclusion pose une question qu'apparemment, au vu de l'ambiguïté de ses démarches expérimentales, l'Education Nationale n'a toujours pas résolue : l'ordinateur est-il pour l'enfant un outil d'acquisition de connaissances ou un outil développant la créativité ?

JL ■

LA GAZETTE DE LIST

de surveillance, ainsi qu'enseignement ou animation de maquettes.

On peut brancher ces cartes sur bon nombre de machines (Apple 2, Canon X-07, Commodore 64, Oric-Atmos, ZX 81 et Spectrum, Thomson M0/T07 IBM-PC...) et leur prix semble raisonnable. A titre d'exemple, Kap annonce la carte 8 sorties binaires pour Commodore à 700 FF ttc. ■

UN LIVRE

Maîtrisez les TO7 et TO7/70
Michel Oury
ETSF
Paris, 1984
Broché, 200 pages
Prix : 86 FF

CE nouvel ouvrage est la reprise du « Maîtrisez le TO7 », que l'auteur a revu et complété en fonction des quelques spécificités du TO7/70.

Dans sa première partie, il nous présente le « système TO7 » (Matériel/logiciel/extensions). Notons que les renseignements qu'il nous donne devraient se trouver sur un manuel technique accompagnant le TO7 ou, pour le moins, être disponible chez Thomson même. Ils sont très intéressants sur un plan technique : description des PIA, schéma général de la mémoire. Quant aux fanas d'Assembleur, ils peuvent se réjouir : ils liront une étude très bien faite de douze sous-programmes du moniteur réalisant la gestion des interfaces des périphériques. Signalons également les schémas des différents connecteurs et prise DB25 du TO7.

La deuxième partie est un manuel de référence Basic TO7, où toutes les instructions et fonctions sont présentées classées par type d'utilisation. L'annexe 8, qui est un index des mots-clés du Basic, permet au lecteur de trouver rapidement l'information qui l'intéresse.

Dans une troisième partie, Michel Oury nous permet de comprendre comment le TO7 gère son image graphique, et comment utiliser pour cela le PIA.

Enfin, le principal intérêt de ce livre est le chapitre du 6809. Le lecteur, même néophyte en langage-machine, aura ici l'occasion de se familiariser avec ce micro-processeur, en étudiant la description de sa structure, de ses modes d'adressage et de son jeu d'instructions (le tout très bien expliqué avec schémas).

En annexe, signalons les adresses remarquables du moniteur (page 0, adresses des portes des PIA et des routines système).

Un ouvrage utile donc, et portant bien son sous-titre (« Du Basic au langage-machine ») puisque, outre un manuel de référence Basic, l'utilisateur intéressé par l'Assembleur y trouvera une mine de renseignements directement utilisables.

JL ■

Le FX-750 P
Une supercalculatrice qui cause aussi « Basic »

LE FX-750 P de Casio à l'allure sérieuse des ordinateurs de poche (programmables en Basic) de la gamme Casio FX. Digne rejeton de l'ancêtre FX-702 P, celui-ci y ressemble par bien des côtés mais apporte un « plus » non négligeable dans le domaine du calcul scientifique et statistique.

L'originalité la plus marquante réside sans doute dans les cartes de mémoire, mémoire permanente même quand les cartes sont débranchées (autonomie : un an ou deux selon le modèle). Où est l'ordinateur ? Dans la carte ou bien dans cette boîte contenant un clavier et un afficheur (cristaux liquides, 24 beaux caractères) ? En fait, sans la carte, le FX-750 P ne fait *plus rien du tout !* La totalité des informations nécessaires et utiles — y compris la mémoire du système — est sur la carte. Enlevez-la et il n'y a plus aucun moyen de calculer 1 + 1.

Tout se passe donc comme si l'on changeait d'ordinateur en changeant de carte. Il y aura chez l'utilisateur une carte pour les calculs mathématiques, une pour la statistique, etc. Les cartes contiennent 2 ou 4 Ko (1 Ko représente, schématiquement, 1024 caractères). L'ordinateur est livré avec une première carte de 4 Ko. Notons au passage qu'il n'existe que deux logements pour ces extensions de la taille des carrés de crédit : 8 Ko au maximum de mémoire sont donc disponibles. En outre, 1296 octets (... « caractères ») sont systématiquement confisqués par la machine ; il faut bien que le FX-750 P se réserve un endroit pour sa petite cuisine personnelle.

Le FX-750 P est un très bon calculateur scientifique. Il possède, bien entendu, les grandes fonctions classiques : trigonométriques et inverses (en RAD, DEG ou GRAD), hyperboliques et inverses, logarithmiques et inverses, $\sqrt{}$, |x|, signe INT et

FRAC (troncatures de nombres), arrondis, nombres sexagésimaux et hexadécimaux, « hasard »,...

Mais encore — originalité marquante — le FX-750 P possède une jolie batterie de constantes préprogrammées : Pi, g (accélération en chute), c (vitesse de la lumière), h (constante de Plank), G (gravitation), e (charge élémentaire), me (masse de l'électron), U (masse atomique), NA (nombre d'Avogadro), k (constante de Boltzmann) et, enfin, Vm (volume d'un kilomole).

Le statisticien trouve aussi de quoi nourrir ses appétits avec une honnête préprogrammation des fonctions statistiques : CNT (Σn), SUMY (Σy), SUMX (Σx), SUMXY (Σxy), SUM X 2 (Σx^2), et SUMY 2 (Σy^2). Notons que ces expressions sont des *variables* dont on peut aisément, et d'autorité, changer les contenus. Cela permettra de ne pas réintroduire de longues séries dont on connaît déjà les informations caractéristiques, ou même d'illustrer certains calculs : « que se passe-t-il si telle variable statistique évolue ? »

Les fonctions proprement dites de calcul statistique sont les écarts types (population ou échantillon) et la régression linéaire avec calcul des paramètres de la droite (a et b), de valeurs estimées (extrapolation) et, enfin, du coefficient (r) de corrélation.

Tous les calculs s'effectuent sur 12 chiffres significatifs même s'ils ne sont visualisés que sur 10 chiffres (arrondis).

Le FX-750 P est aussi un ordinateur programmable en Basic. De ce point de vue, on retrouve le langage-type des poquettes Casio, avec dix zones programmables indépendamment. Un Basic honnête mais sans surprise. L'éditeur de lignes demeure rudimentaire ; le contrôle et diagnostic des erreurs est resté succinct.

Pas de PEEK, pas de POKE dans ce Basic : la machine est « fermée » de ce point de vue, le langage-machine ne sera pas accessible.

Concernant les périphériques, à côté des cartes de mémoire permanente, on peut aussi connecter une petite imprimante thermique (20 colonnes) qui fait office d'interface. C'est par ce canal que l'on raccordera un magnétophone. Les prix : 1 600 FF ttc pour l'ordinateur avec une carte de 4 Ko, 1 150 FF pour l'imprimante-interface K7, et il en coûtera 600 FF pour 4 Ko supplémentaires.

JCK ■

DU CÔTÉ DES CLUBS

Dans le département du Rhône

A Oullins, l'école primaire Moreaud accueille dans ses locaux les membres de l'OMI (Oullino Micro Informatique) qui vous invitent à les rejoindre et à participer à leurs activités tous les soirs de 18 à 20 heures. Leurs coordonnées sont les suivantes

Association OMI
Ecole Centre A
93 rue de la République
69600 Oullins
Tél. : (78) 51 34 31

Chez nos amis canadiens

LA SIAQ (Société d'Informatique Amateur du Québec), l'un des plus anciens groupes d'informaticiens amateurs de l'Est Canadien souhaite tisser un réseau de communication entre les différents clubs.

Son but est de favoriser les échanges d'expériences et d'informations, mais surtout, de créer avec les petits clubs éloignés des liens qui les rendront moins isolés.

Pour donner une nouvelle dimension à son ambitieuse entreprise, la SIAQ aimerait pouvoir entrer en contact avec des clubs français.

Si vous souhaitez vous aussi découvrir de nouveaux horizons... informatiques, vous pouvez lui écrire :

Société d'Informatique
Amateur du Québec
C.P. 9242
Sainte-Foy
P.Q. G 1 V 4 B 1
Canada

Exposition dans les Alpes-Maritimes

AU *Centre Jeunesse, Culture et Loisirs* de Mandelieu-la-Napoule (association loi de 1901) se tiendra, début décembre, la première exposition que cette ville consacre à l'informatique de loisir.

Cette manifestation dont l'entrée est gratuite se déroulera le samedi 8 décembre, de 14 à 19 heures, et le dimanche 9 décembre 1984, de 10 à 12 heures et de 14 à 19 heures.

C J C L
802 boulevard des écureuils
06210 Mandelieu-la-Napoule
Tél. : (93) 49 10 12

Faites-vous connaître...

Si vous faites partie d'un club d'informatique, sans but lucratif, où l'on apprend et pratique la programmation et si vous recherchez de nouveaux adhérents, signalez-nous votre existence. En lisant votre adresse dans ces colonnes, beaucoup de lecteurs seront contents d'apprendre que votre club n'est pas trop loin de chez eux. ■

LA GAZETTE DE LIST

Stockage de programme sur vidéodisque

DISQUETTES, cassettes et cartouches pourraient se retrouver bientôt en concurrence avec une nouvelle technique de stockage : le vidéodisque-laser.

Prématuré ? Peut-être, mais le couplage ordinateur-vidéodisque n'est résolument plus du domaine de la science-fiction. Une preuve de poids, le Japonais **JVC** (*Victor Company of Japan*) vient de lancer à Tokyo un nouvel MSX, le HC-7 qui réserve bien des surprises. Vendu aux alentours de 3 400 F, le HC-7 est pourvu d'une fonction surimpression-vidéo (ça devient classique) et dispose d'un autre atout, la compatibilité avec un lecteur de vidéodisques interactif. Le HC-7 prendra donc le contrôle du vidéodisque.

Pour accompagner le HC-7, JVC a par ailleurs annoncé la commercialisation prochaine d'un nouveau lecteur vidéodisque pourvu d'une RS 232 C et qui pourra se connecter sur la plupart des machines MSX. Le lecteur pourrait coûter aux environs de 6 000 F.

Quant aux programmes, on déclare chez JVC que de nombreuses applications seront développées prochainement sur vidéodisque. Les consommateurs français devront toutefois patienter. ■

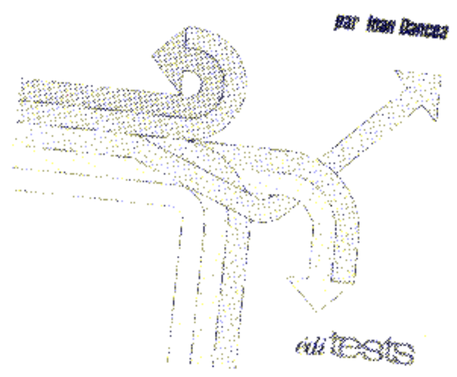
Kodak : « clic-clac » disquettes

KODAK, le roi de la photo a décidé de se lancer dans « l'aventure informatique » et commencera avec des disquettes. Dès le début de l'année prochaine, des supports aux couleurs rouge et jaune de la société seront donc disponibles en format 8,5.25 et 3.5 pouces avec le choix de simple, double et même quadruple densité.

Kodak a justifié cette décision par « un souci de diversification » et l'extension du marché des disquettes d'ici 1990 (plus de 50 % de croissance), qui devrait alors représenter un chiffre d'affaires de quelque 4 milliards de dollars. ■

UN LIVRE

Opérations arithmétiques dans les ordinateurs
Ioan Dancea
Editests
Paris, 1984
Broché, 170 pages
Prix : 100 FF



DE prime abord, on pourrait s'étonner qu'il faille 170 pages pour expliquer comment sont effectuées les quatre opérations élémentaires (addition, soustraction, multiplication et division) dans les ordinateurs. Mais le sujet est en fait beaucoup plus vaste qu'on ne l'imagine, et l'auteur, d'ailleurs, ne prétend pas l'avoir épuisé.

Comme toujours en informatique, les sujets apparemment les plus simples sont en fait très complexes. On aurait donc tort de croire que ce livre est facile à aborder, d'autant plus qu'il nécessite des connaissances de base en algèbre logique, en électronique et en arithmétique binaire. Cela vient du fait que le sujet est à la frontière entre l'électronique et l'informatique. S'il est indispensable de posséder des bases en arithmétique binaire (en ayant, par exemple, déjà réalisé des programmes en langage-machine), des lacunes en électronique n'empêcheront pas le lecteur de comprendre les données exposées, mais seulement quelques schémas de circuits électroniques.

Cela dit, bien qu'il ne soit pas destiné à tous les publics, le livre est passionnant. De très nombreuses méthodes sont passées en revue, aussi bien celles qui sont utilisées dans les calculatrices de poche que dans les ordinateurs de grande puissance.

Notons que les deux derniers chapitres exposent l'arithmétique des nombres flottants et des nombres décimaux. Ils paraissent un peu courts, car le sujet est plus complexe que dans les chapitres précédents, mais ils fournissent une bonne présentation des techniques utilisées.

TC ■

UNE CASSETTE

Logo Logic 1
Cassette pour C.64
et Vic 20
Edité par No Man's Land
Prix : 120 FF

LE logiciel Logo Logic 1 est fourni avec un fascicule de sept mini-pages au format d'une cassette. C'est peu, comparé aux livres qui accompagnent en général les véritables Logo.

Au premier abord, il semble que nous soyons en présence d'un Basic appauvri, baptisé langage, et présenté comme un Logo, avec deux modes de travail, les modes DIRECT et PROGRAMME.

En DIRECT, nous disposons des instructions RUN, LIST, SAVE, ... (qui nous rappellent un air connu) et TUE, avec en prime, lors de l'utilisation de cette dernière, une sorte de couinement émis par la machine, il aurait été préférable de conserver NEW.

On peut faire avancer un objet appelé TORTUE de 1 à 255 pas — on ne dépasse pas l'octet si facilement —, mais impossible de la faire reculer. La tortue peut pivoter vers la droite ou la gauche mais uniquement à angle droit. DROITE 1, par exemple, la fait tourner vers la droite de 90 degrés.

Si les enfants connaissent très jeunes le triangle qu'ils utilisent fort bien pour dessiner une maison, la toiture de No Man's Land n'évolue que dans un univers carré. C'est une faute grave.

Abordons le second mode. Pour créer un programme, il faut d'abord, et comme en Basic, numéroter les lignes. Une fois terminé, celui-ci forme un tout, et non un ensemble de procédures comme ce devrait être le cas pour un Logo. Une astuce permet cependant de définir un morceau de programme et de lui donner un nom. La « primitive » NOM est en fait un GOSUB déguisé.

Quant aux paramètres, ils ont disparu, remplacés par 26 variables globales (de A à Z). Chacune ne peut contenir qu'un nombre entier positif compris entre 0 et 255. La variation d'un contenu de variable n'est possible que par incrémentation (+ 1) ou décrémentation (– 1) et conduit à un message d'erreur lorsque le contenu n'est pas compris entre les bornes 0 et 255.

Les possibilités sont finalement assez limitées. Un exemple d'utilisation est proposé sous la forme d'une belle spirale carrée créée grâce au VA EN (GOTO ?), mais qui ne peut malheureusement s'arrêter sans message d'erreur car le SI n'existe pas. Difficile pourtant d'enseigner la logique de programmation en oubliant le SI.

Pas de Logo donc, et guère de « Logic », mais un simple jeu dont les résultats ne sont pas très motivants. Faire des cœurs sur des horizontales et des verticales est tout de même limité. Certains programmeurs ont réalisé en Basic des simulations de la tor-

tue Logo beaucoup plus sophistiquées et ils ont, eux, très vite renoncé à baptiser leur produit du nom de « mini-Logo ».

Que les possesseurs de Vic se consolent en écrivant eux-mêmes un programme de simulation graphique, en attendant de pratiquer un vrai Logo sur le Commodore 64.

RD ■

Pencil II : un australien aguichant

UN semi-professionnel à 1 500 F, il fallait être australien pour tenter le coup. C'est fait. La société **Hanimex** spécialisée jusqu'à présent dans les jeux électroniques et le matériel photo lance le Pencil II, ordinateur destiné à la fois à l'initiation, aux jeux, et aux « semi-pro ».

Côté initiation, un Basic « classique » (*simple, complet, évolutif* selon le constructeur). Côté jeu, un nom-clef, le Pencil II est compatible avec les cartouches **Coléco**. Pour le reste, d'indéniables qualités mais également des lacunes. Le Pencil revendique quatre générateurs de sons indépendants (dont un exclusivement pour les bruits), un Basic graphique « très complet » (il y en a donc deux), et une foule de périphériques dont une extension mémoire de 64 Ko, une carte 80 colonnes, une RS 232 C et deux lecteurs de disquette.

Enfin, Hanimex annonce la disponibilité de CP/M et la possibilité de reprendre la plupart des logiciels fonctionnant sous ce système. L'astuce : transférer ces logiciels d'un autre ordinateur via la RS 232. Reste que le clavier peut ne pas plaire à tous (encore ces touches-gomme), que la capacité mémoire en standard est assez faible et que l'aspect général de l'appareil inspire une certaine retenue. Mais il faut, bien sûr, tenir compte du prix de base, 1 500 F. ■

CASSETTES ET DISQUETTES

Haute résolution
permet d'obtenir une résolution graphique de 256 × 220 points
Cassette pour ZX81
Distribué par Direco
Prix : 120 FF

Origraph
Logiciel de création graphique
Cassette pour Oric1/Atmos
Edité par Micro Futur
Prix : 150 FF

Logomonde
utiliser Logo
cassette pour TO7, TO7-70 et MO5
Edité par Hatier
Prix : 185 FF

Basic Français
Cassette pour Oric1/Atmos
Edité par Loriciels
Prix : 180 FF

Désassembleur
Cassette pour Laser 200
Edité par ERE Informatique
Prix : 95 FF

Auteur
Cassette pour Oric1/Atmos
Distribué par Microprogrammes 5
Prix : 180 FF

Editeur assembleur
Cassette pour ZX Spectrum
Edité par Eyrolles
Prix : 120 FF

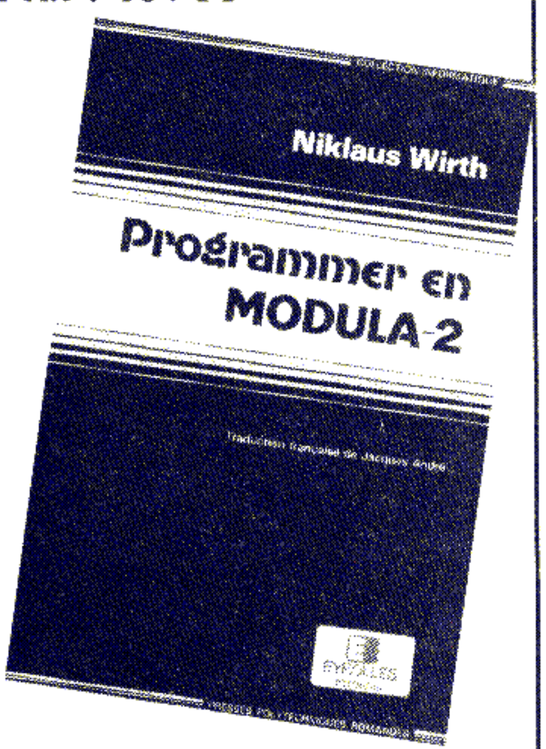
Star
Cassette pour Oric
Distribué par Microprogrammes 5
Prix : 180 FF

Basic étendu
Cassette pour Laser 200
Edité par ERE Informatique
Prix : 95 FF

LA GAZETTE DE LIST

UN LIVRE

Programmer en Modula-2
Niklaus Wirth
Traduit de l'anglais par
Jacques André
Presses Polytechniques
Romandes (Eyrolles diff.)
Lausanne, 1984
Broché, 264 pages
Prix : 184 FF



UN classique de 1982, déjà à sa seconde édition chez Springer Verlag. On ne présente pas Wirth, l'homme qui a fait sans aucun doute le plus pour une programmation intelligente depuis ses premiers travaux sur Algol W. Modula-2 (voir, dans le numéro 4 de *LIST*, Pascal et sa famille) est son dernier enfant, plus spécialement consacré à la multiprogrammation.

On pourrait croire, à la lecture d'ouvrages de base comme le « Manuel de l'Utilisateur de Pascal » et le « Rapport révisé » (même auteur, Eyrolles 1978) à un texte froid et pour tout dire peu digestible. Pas du tout ; je ne sais si on a changé Wirth, mais son livre est, cette fois-ci, plus agréable à lire, avec de nombreux exemples — entre autres graphiques —, même si le sujet reste évidemment assez ardu. Bien entendu les pascalophiles se sentiront tout à fait à l'aise devant cette extension apparemment assez puissante de leur outil de travail.

Je ne sais si Modula-2 restera un outil professionnel d'informaticien ou s'il se développera en micro-informatique comme héritier de Pascal (voire de Basic) ; voilà en tous cas un bon moyen de se faire une idée de ses possibilités.

AW ■

UN PETIT TOUR CHEZ LE LIBRAIRE

Logo, manuel de référence
Doris Avram, Tristan Savatier
et Michèle Weidenfeld
Editions Cédic/Nathan
Paris, 1984
Broché, 110 pages
Prix : 79 FF

Apprendre et appliquer le langage APL
Bernard Legrand
Editions Masson
Paris, 1984
Broché, 418 pages
Prix : 179 FF

Le langage B.A.L.
Initiation et pratique
Didier Lévy
Editests
Paris, 1984
Broché, 124 pages
Prix : 90 FF

Langage-machine, Trucs et astuces pour ZX Spectrum
Pascal Pellier
Editions Eyrolles
Paris, 1984
Broché, 138 pages
Prix : 89 FF

Le système Prologue Version 2.2
Pierre Giraud
Editests
Paris, 1984
Broché, 212 pages
Prix : 110 FF

Le système CP/M pour Z80
Fabienne et Philippe Gysel
Editests
Paris, 1984
Broché, 190 pages
Prix : 100 FF

Au cœur du HX 20
Laurent Besle
Editions Eyrolles
Paris, 1984
Broché, 160 pages
Prix : 130 FF

L'Assembleur de l'Oric 1 et Oric-Atmos
Marcel Henrot
Editions du PSI
Lagny, 1984
Broché, 160 pages
Prix : 90 FF

J'apprends le Basic
Michel Caut
Editions ETSF
Paris, 1984
Broché, 128 pages
Prix : 65 FF

Le Basic par l'exemple
Jean-François Schmid
Les Editions d'organisation
Paris, 1984
Broché, 196 pages
Prix : 91 FF

Guide du Basic Commodore 64 - Vic 20
Hergert Douglas
Traduit par Jean-Louis Gréco
Editions Sybex
Paris, 1984
Broché, 210 pages
Prix : 78 FF

Maîtrisez le MO5 du Basic au L.M.
Michel Oury
Editions ETSF
Paris, 1984
Broché, 198 pages
Prix : 86 FF

Aller plus loin en Basic TO7
Jean-Claude Wanner
Editions Eyrolles
Paris, 1984
Broché, 294 pages
Prix : 120 FF

Passeport pour Basic TO7 et TO7-70
Claudy Galais
Editions ETSF
Paris, 1984
Broché, 158 pages
Prix : 39 FF

Clefs pour le Commodore 64
Daniel-Jean David
Editions du PSI
Lagny, 1984
Broché, 126 pages
Prix : 100 FF

Utilitaires pour ZX 81
Marc Saal
Editions ETSF
Paris, 1984
Broché, 122 pages
Prix : 35 FF

Programmation du Commodore 64
Ian Sinclair
Traduit par Gérard Piloquet
Editions Belin
Collection Modulo
Paris, 1984
Broché, 126 pages
Prix : 95 FF

Pratique du TO7-70
Henri Lilen
Editions Radio
Paris, 1984
Niveau 1
Broché, 190 pages
Prix : 75 FF
Niveau 2
Broché, 174 pages
Prix : 100 FF

Logibul au pays de l'informatique
Sheila Dvorchik et
Lasley Wasylenki
Editions Belin
Collection Modulo
Paris, 1984
Broché, 112 pages
Prix : 95 FF

La transportabilité du logiciel
Olivier Lecarme et
Mireille Pellisier
Editions Masson
Paris, 1984
Broché, 262 pages
Prix : 140 FF

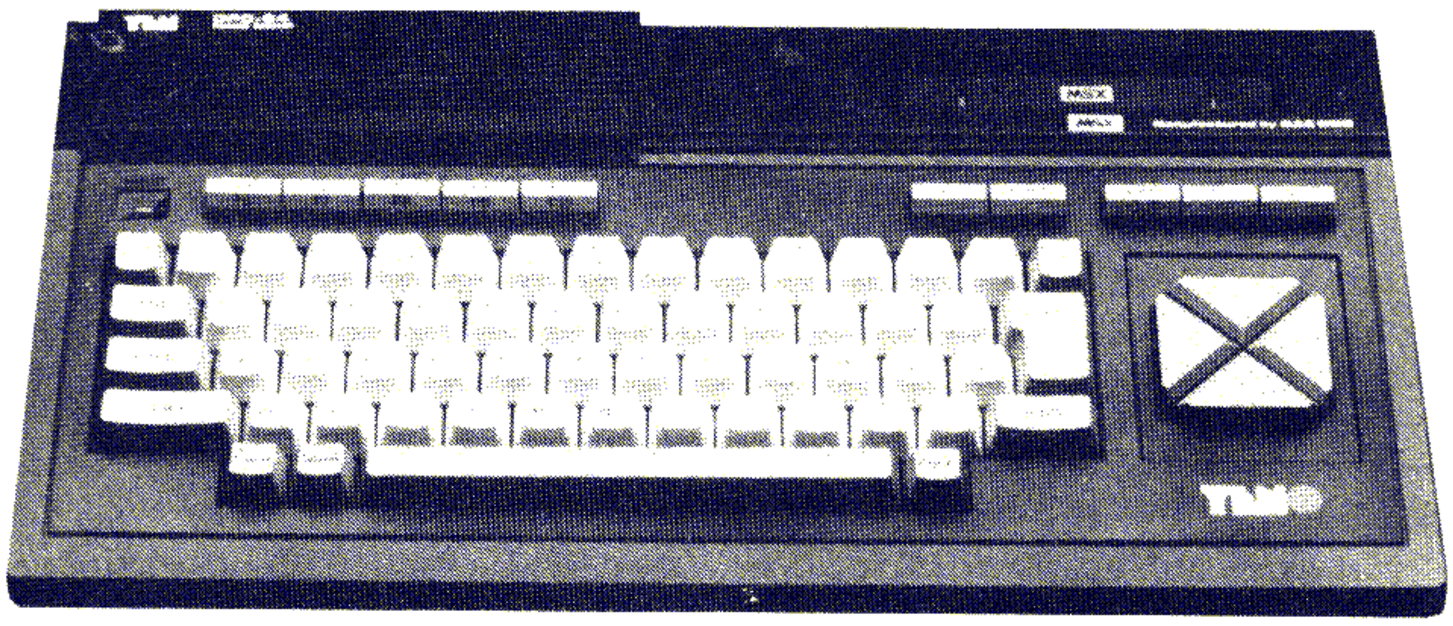
Mathématiques pour micro-informatique
Les nombres et leur traitement
W. Barden
Traduit par G. Revellin
Editions Dunod
Paris, 1984
Broché, 128 pages
Prix : 65 FF

Pratique du MO 5
Henri Lilen
Editions Radio
Paris, 1984
Niveau 1
Broché, 190 pages
Prix : 75 FF
Niveau 2
Broché, 174 pages
Prix : 100 FF ■

LIST A TESTÉ

LE BASIC DU STANDARD MSX

UN nombre croissant d'ordinateurs familiaux adoptent le standard MSX qui est en grande partie défini par le Basic MSX, autrement dit Basic Microsoft superétendu. La firme de logiciels, en mettant au point cette version du Basic, répondait à un appel d'offres lancé par l'industrie japonaise, désireuse de standardiser l'informatique familiale.



C'est sur un ordinateur Yeno DPC 64 que le Basic MSX a été testé mais, en fait, la machine ici importe peu puisque ce langage est standard.

Nous commencerons par l'éditeur qui, bonne surprise, est un modèle du genre. Tout d'abord, il est plein écran. Microsoft semblait un peu fâché avec ce type d'éditeur, mais cela fait aujourd'hui partie du passé.

Bien gérer le graphisme

Les touches de déplacement du curseur forment un pavé indépendant, il est même possible de déplacer le curseur en diagonale ! La touche INSERT met l'éditeur en mode insertion : seule une nouvelle pression sur cette touche ou sur une touche du curseur permettra d'en sortir. Le clavier de la machine sur laquelle j'ai effectué cet essai est du type AZERTY accentué. De ce fait, comme sur toute machine à écrire française, les chiffres sont obtenus grâce à la touche SHIFT ; mais celle-ci peut être bloquée avec la touche CAPS qui, de plus, est munie d'une diode rouge de contrôle.

Les commandes paramétrables DELETE, RENUM et AUTO sont présentes et la longueur maximum d'une ligne est de 255 caractères.

Le Basic MSX est extrêmement complet dans le domaine de la gestion du graphisme. Mise à part l'existence d'un macro-langage, dont nous parlerons plus loin, il n'y a rien de révolutionnaire mais (presque ?) toutes les commandes existant dans ce domaine sont ici présentes. Les classiques PSET et PRESET allument et éteignent un point ; POINT teste l'état d'un point ; LINE trace une ligne ; CIRCLE trace un cercle ; PAINT remplit une surface avec la couleur désirée. Vous avez d'autre part à votre disposition tout ce que vous avez toujours voulu avoir pour manipuler les lutins (*sprite* en anglais). Grâce à la commande SCREEN, vous pouvez définir le mode d'affichage : texte (24 × 40 ou 24 × 32), graphique ou graphique multicolore (256 × 192 en 16 couleurs) et la taille des lutins (8 × 8 ou 16 × 16 points). SPRITE ON/OFF/STOP, SPRITE$, PUT SPRITE et ON SPRITE GOSUB permettent de créer, de faire évoluer les lutins et d'en gérer les interactions. Leurs mouvements peuvent s'effectuer sur 32 plans différents : une porte ouverte sur la troisième dimension.

Parlons maintenant de ce que Microsoft appelle « the graphic macrolanguage » et qui permet le tracé de figures complexes. Un exemple vaut mieux qu'un long commentaire. Après passage en mode graphique, l'exécution de

```
10 A$ = "R255D191L255U191"
20 DRAW A$
```

trace un cadre de la taille de l'écran. La chaîne de caractères stockée en A$ doit en fait être lue : RIGHT 255 (tracer 255 points en allant vers la droite), DOWN 191 (191 points vers le bas), LEFT 255 et UP 191. Il est également possible de dessiner des droites obliques en définissant l'angle, et de travailler en coordonnées relatives ou absolues.

Un autre macro-langage fonctionnant selon le même principe donne accès aux fonctions sonores de l'ordinateur :

```
10 A$ = "T250CDEFGAB"
20 PLAY A$
```

provoque l'émission de la gamme selon un tempo assez rapide (T250). Il est également possible de choisir entre huit octaves, et de déterminer la durée, le volume, l'enveloppe et l'attaque de chaque note. A cela s'ajoute enfin l'instruction BEEP qui produit un bip non modulable.

Les noms de variables ne comportent au maximum que deux caractères significatifs, mais il existe quatre types de variables, un pour les variables de caractères et trois pour les variables numériques. Les instructions DEFINT, DEFSNG, DEFDBL, DEFSTR ou les suffixes %, !, # et $ permettent de les déclarer selon chacun de ces types. Passons en revue les trois types de variables numériques.

Le type entier code chaque variable sur deux octets, soit 16 bits. De ce fait, le nombre contenu dans une telle variable est obligatoirement compris dans un intervalle de longueur $2^{16} = 65536$, cet intervalle va de -32768 à 32767. Avec ce type de variables, les calculs sont plus rapides et la place occupée moindre.

Le type simple précision autorise le codage de nombres en virgule flottante sur quatre octets. Les calculs sont effectués sur sept chiffres dont six sont ensuite stockés.

Une mémoire dynamisée

Le type double précision est celui que la machine adopte par défaut. Les calculs sont alors effectués sur 16 chiffres dont 14 sont finalement stockés. Les calculs faisant intervenir les fonctions transcendantales sont systématiquement effectués sur 16 chiffres, quel que soit le type des variables mis en œuvre ; seul le stockage final est susceptible de varier suivant ces types : le test 6 donne en effet les mêmes (piètres !) résultats, que les variables soient déclarées en simple ou en double précision (1).

Mis à part le classique DIM, qui permet la déclaration de tableaux de dimension presque quelconque (une ligne ne doit tout de même pas dépasser 255 caractères), il existe deux instructions très pratiques liées à la gestion des variables : SWAP qui échange les contenus de deux variables (utile, entre autres, pour les programmes de tri) et ERASE qui annule une instruction DIM. Ce Basic permet donc une gestion dynamique de la mémoire.

Microsoft a beaucoup perfectionné le traitement des chaînes de caractères et le Basic MSX offre tout ce qui est nécessaire pour couper, localiser, décoder les chaînes alphanumériques. Signalons rapidement MID$, RIGHT$ et LEFT$, ainsi que l'instruction INSTR, moins courante, grâce à laquelle on peut détecter et localiser une chaîne de caractères à l'intérieur d'une autre chaîne plus longue.

Il existe trois fonctions permettant de convertir les nombres de la base 10 à l'hexadécimal, l'octal et le binaire ; l'exécution de PRINT HEX$(15);" ";OCT$(15);" ";BIN$(15) produira l'affichage de F 17 1111. Notons au passage que les préfixes &H, &O et &B permettent de travailler directement dans l'une de ces bases.

(1) Voir Les dix tests de LIST, *pages 68 et 69 de ce numéro.*

Dans le domaine de la programmation structurée, nous remarquons que Microsoft n'a pas fait preuve d'audace particulière : pas d'instructions de structuration dérivée du Pascal, les sous-programmes se résument aux sempiternelles instructions GOSUB et RETURN et le passage d'argument doit toujours s'effectuer "manuellement" par l'intermédiaire de variables. On trouve cependant quelques instructions très pratiques pour contrôler le déroulement d'un programme. Ainsi ERROR permet à l'utilisateur de définir ses propres types d'erreur :

```
 10 ON ERROR GOTO 10000
    (...)
100 IFX < 0 THEN ERROR 200
    (...)
10000 IF ERR=200 PRINT"X
      NEGATIF"
```

Si X est négatif à la ligne 100, un branchement automatique à la ligne 10000 produira l'affichage de X NEGATIF.

Des instructions pour les jeux

Les instructions ON GOTO et ON GOSUB peuvent d'ailleurs fonctionner selon le même principe. Ainsi, ON KEY GOSUB provoquera un branchement à l'une des lignes spécifiées suivant la frappe d'une touche de fonction. Notons que KEY permet justement d'attribuer une séquence de touches à une des touches de fonction et que KEY LIST fait apparaître la liste de ces touches suivie de leurs définitions. ON STICK GOSUB est excellent pour programmer les poignées de jeux (STICK retourne simplement la direction, 0 à 8, indiquée par la manette). Les pseudo-variables PDL et PAD permettent de tester la position d'une manette à mouvement continu. ON STOP GOSUB provoque un branchement à la ligne spécifiée dès que l'exécution du programme est interrompue. Plus curieux : ON INTERVAL = ...GOSUB provoque un branchement à la ligne spécifiée, de manière rythmique en fonction de l'intervalle de temps choisi.

Toutes ces instructions peuvent être activées ou désactivées par des KEY ON/OFF, STOP ON/OFF, etc.

Notons que, fidèle à son habitude, Microsoft n'autorise pas le programmeur à faire suivre un GOTO ou un GOSUB d'une expression algébrique : seule, la constante numérique est permise. On a beau dire que ON GOTO permet de combler cette lacune, c'est pourtant bien utile dans certains cas, et l'on connaît des Basic qui cumulent les deux possibilités. En revanche, THEN peut être suivi de ELSE.

L'éventail des fonctions mathématiques est assez limité (COS, SIN, TAN, ATN, LN, LOG, EXP, SQR, MOD...) mais la précision est excellente (16 chiffres), et l'on peut définir de nouvelles fonctions avec DEFFN. On obtient ainsi arc sin par la séquence :

```
DEFFN ACS(X)=ATN (X/SQR (-X
*X+1))
```

Les fonctions logiques AND et OR sont complétées par XOR (*ou* exclusif), NOT, IMP (implication) et EQV (équivalence).

Un effort particulier a été produit

Différents Basic MSX disponibles en France

Nom	Constructeur (ordre alphabétique)	Prix public (en FF)	Mémoire vive totale (en Koctets)
V 20	Canon	3 000	64
PHC 28	Sanyo	3 000	32
YIS 503	Yamaha	3 400	48
YC 64	Yashica	3 950	80
DPC 64	Yeno	3 450	64
Mémoire morte	32 Koctets		
Langages	Basic MSX, Assembleur du Z 80		
Variables	Entiers de - 32 768 à 32 767 ; simple précision sur sept chiffres dont six stockés ; double précision sur 16 chiffres dont 14 stockés.		
Modes d'affichage	Texte, 24 lignes sur 32 ou 40 colonnes ; graphique multicolore, 64 sur 128 points et 256 sur 192 points en 16 couleurs ; lutins de 8 sur 8 ou 16 sur 16 points.		
Seule une partie de la mémoire vive totale annoncée est programmable en Basic. Pour en connaître la capacité, il faut faire FRE à la mise sous tension.			

A
ABS
AND
ASC
ATN*
AUTO

B
BASE
BEEP
BIN$
BLOAD
BSAVE

C
CALL
CDBL
CHR$
CINT
CIRCLE
CLEAR
CLOAD
CLOAD?
CLOSE
CLS*
COLOR
CONT*
COS*
CSAVE
CSNG
CSRLIN

D
DATA*
DEFDBL
DEF FN
DEFINT
DEFSNG
DEFSTR
DEFUSR
DELETE
DIM*
DRAW

E
ELSE
END*
EOF
EQV
ERASE
ERL
ERR
ERROR
EXP*

F
FIX
FOR*
FRE

G
GOSUB*
GOTO*

H
HEX$

I
IF*
IMP
INKEY$*
INP
INPUT*
INPUT #
INPUT$
INSTR
INT*
INTERVAL
ON/OFF/STOP

K
KEY
KEY LIST
KEY ON/OFF
KEY ON/OFF/STOP

L
LEFT$
LEN*
LET*
LINE
LINE INPUT
LINE INPUT #
LIST*
LLIST
LOAD
LOCATE
LOG*
LPOS
LPRINT
LPRINT USING

M
MAXFILES
MERGE
MID$
MOD
MOTOR

N
NEW*
NEXT*
NOT

O
OCT$
ON ERROR GOTO
ON GOTO*
ON GOSUB*
ON INTERVAL GOSUB
ON KEY GOSUB
ON SPRITE GOSUB
ON STOP GOSUB
ON STRIG GOSUB
OPEN
OR
OUT

P
PAD
PAINT
PDL
PEEK
PLAY
POINT
POKE
POS
PRESET
PRINT*
PRINT USING
PRINT #
PRINT # USING
PSET
PUT SPRITE

R
READ*
REM*
RENUM
RESTORE*
RESUME
RETURN*
RIGHT$
RND*
RUN*

S
SAVE
SCREEN
SGN
SIN*
SOUND
SPACE$
SPC
SPRITE ON/OFF/STOP
SPRITE$
SQR*
STEP*
STICK
STOP*
STOP ON/OFF/STOP
STRIG
STRIG ON/OFF/STOP
STRING$
STR$
SWAP

T
TAB
TAN*
THEN*
TIME
TRON
TROFF

U
USR

V
VAL
VARPTR
VDP
VPEEK
VPOKE

W
WAIT
WIDTH

X
XOR
&B, &H, &O

** Mots présents dans la plupart des Basic. Un programme écrit à l'aide de ces seuls mots peut être adapté très facilement d'un ordinateur à un autre (à condition de ne pas dépasser la taille de la mémoire).*

dans le domaine de la gestion des cassettes et des fichiers : le jeu d'instructions est en effet très fourni. Avec OPEN et CLOSE, on ouvre et l'on ferme des fichiers non seulement sur magnétophone, mais aussi sur l'écran texte, l'écran graphique, ou sur une imprimante.

Une conception prudente

Comme d'habitude, CSAVE et CLOAD permettent de sauvegarder et relire des programmes Basic alors que PRINT # et INPUT # en font autant pour les variables.

- SAVE et LOAD assurent la sauvegarde et la lecture de programmes non compactés, c'est-à-dire en codes ASCII.
- BSAVE et BLOAD en font autant, mais avec des programmes en langage-machine.
- MERGE charge les programmes en codes ASCII en les faisant cohabiter avec les lignes déjà en mémoire.
- CALL est une instruction assez originale : elle donne accès à de nouvelles fonctions provenant de cartouches de mémoire morte (MEM) enfichées.

A côté des classiques PEEK et POKE, on trouve VPEEK et VPOKE avec lesquels on peut lire et écrire en mémoire vidéo. Cette dernière est en effet hors de portée des 64 Ko directement adressables par le Z80, il est donc nécessaire de changer de page.

DEFUSR permet de définir l'emplacement de dix routines en langage-machine (DEFUSR0 à DEFUSR9). Elles pourront être appelées par USR0(X) à USR9(X) où X est un argument entier quelconque qui sera transféré automatiquement dans le registre HL du Z80.

Lors de la conception de ce Basic, la prudence a été de rigueur : le Basic MSX est effectivement étendu mais il n'a rien de révolutionnaire. Et s'il est attachant, il faut peut-être en rechercher la cause dans le poids des habitudes qui existent même dans un domaine de pointe comme l'informatique : quoi de plus rassurant pour un programmeur que d'apprendre qu'une tentative de standardisation repose sur le bon vieux Basic Microsoft ?

Thierry LÉVY-ABÉGNOLI

UN ORDINATEUR AU SECRET

QUELQUES lignes de Basic suffisent pour effectuer les transformations fondamentales de la cryptographie. Deux programmes (ici pour X-07) en fournissent la démonstration.

Née de la cryptographie, notamment grâce à Turing, l'informatique a complètement bouleversé cette discipline, mais continue à en dépendre pour la protection de ses banques de données.

L'histoire de la cryptographie répète à satiété le scénario de la victoire du canon sur la cuirasse : décideurs mal conseillés ou lésinant sur les moyens nécessaires, d'un côté et, de l'autre, succès gros de conséquences des décrypteurs grâce à un peu de génie, pas mal de chance, beaucoup de fautes de l'adversaire, et beaucoup d'inlassable labeur. Cette histoire est d'autant plus étrange que les chiffreurs disposent depuis longtemps de la protection inviolable que constitue la clef aléatoire « une fois ». Totalement impuissante contre elle, l'informatique permet au contraire de créer et de stocker par millions de caractères, d'utiliser et même d'effacer aussitôt après usage ces clefs « une fois » avec un maximum de commodité.

En attendant que l'on admette la nécessité de telles clefs, strictement aléatoires, les chiffres les moins mauvais s'appuieront sur une ou plusieurs clefs, aussi longues que le message et jamais réutilisées bien entendu, mais seulement « pseudo-aléatoires » parce que créées de façon convenue à partir de clefs secrètes beaucoup plus courtes.

Ici encore l'informatique intervient massivement, mais cette fois dans les deux plateaux de la balance.

D'une part, elle dote les décrypteurs de fantastiques moyens de calcul permettant l'analyse de milliards d'hypothèses. De l'autre, elle met à la disposition de tout un chacun d'excellents outils de chiffrement à bon marché : convenablement utilisé, un ordinateur de poche peut être aussi sûr que tous les codes ou machines à rotors qui traînent encore dans les chancelleries attardées ou les états-majors exotiques.

On peut dégager certaines règles à ce sujet.

- Matériel : n'importe quel ordinateur, même petit, disposant de préférence d'une petite imprimante et d'une dizaine de Ko de mémoire vive.
- Logiciel : quelques lignes de Basic suffisent pour créer les clefs et effectuer les transformations fondamentales de la cryptographie, transpositions et substitutions. Un degré assez élevé de complexité, inacceptable quand on chiffrait à la main, devient alors souhaitable. Aucun inconvénient, par exemple, à meubler d'une cascade d'opérations à sens unique tout le temps nécessaire à la frappe ou à l'impression.
- La mise en œuvre se réduit à l'entrée de la clef secrète et à la frappe du texte, avec impression de sa traduction (crypto ou clair) au fur et à mesure ou aussitôt après.
- Erreurs : une erreur ponctuelle de frappe ou de transmission ne doit pas rendre indéchiffrable toute une portion de texte.
- Sécurité : un système n'est acceptable que s'il est quasi impossible, non seulement de décrypter les messages (cela va

Lignes du programme de chiffrement	Lignes du programme de déchiffrement	Commentaires
1 à 4	1 à 4 et 6	Initialisation, entrée des paramètres
	5	Préparation de la transposition
5	10	Initialisation de la première séquence RND par K = RND(−X/Y)
6 à 8	7	Entrée, contrôle et affichage des textes
8	11	Première phase de la substitution par 2E4*RND(1)
9	5	Initialisation de la deuxième séquence RND par K = RND(−X*Y)
9	10	Effacement de la clef par X = 0
10 et 11	8 et 9	Transposition et deuxième phase de la substitution par 1E4*RND(1)
12		Impression et découpage du crypto en groupes de quatre lettres
	11 et 12	Impression du clair et restitution des espaces
13	12	Impression du groupe date.heure et de la longueur du message

Le programme dans ses grandes lignes

de soi), mais aussi de reconstituer les clefs à partir de tous les autres éléments supposés connus : matériel et programmes, procédures et collections de clairs et des cryptos correspondants. De plus, les clefs doivent être fréquemment renouvelées.

Une transposition enchevêtrée

Avant de passer à la description d'un exemple, ajoutons que les procédés de chiffrement à clefs révélées, qui passionnent les mathématiciens, mettent généralement en œuvre des cascades d'opérations portant sur des nombres de 100 à 200 chiffres, et ne paraissent donc guère convenir aux petits matériels. En outre, la moindre erreur de transmission brouille plusieurs dizaines de caractères.

Techniquement, le procédé programmé ci-après repose sur une transposition enchevêtrée avec une substitution complexe.

A partir de la clef secrète (X) et d'un groupe date.heure (Y) attribué au message, on initialise successivement deux séquences « RND » qui génèrent des valeurs aléatoires. L'une fournit sous la forme 2E4*RND(1) les éléments de la première partie de la clef de substitution, l'autre donne alternativement les éléments de la clef de transposition, sous la forme (L-I + 1)*RND(1), et ceux de la deuxième partie de la clef de substitution, sous la forme 1E4*RND(1). Le choix des multiplicateurs 1E4, 2E4 est lié à la grandeur limite des nombres entiers, ±32767 en Basic Microsoft. La transposition est un mélange aléatoire de l'ensemble du message. La substitution est opérée par des additions modulo 27, ce qui limite le clair aux majuscules et à l'espace (majuscules et signe @ dans le crypto). Il n'y a pratiquement aucun temps mort et les traductions s'impriment dès la fin de la frappe.

Les textes, clair ou crypto, sont entrés au kilomètre en INKEY$. Il *ne* faut donc *pas* chercher à corriger les fautes de frappe, qui seront généralement de peu de conséquence. Si toutefois une faute lors de la frappe du crypto provoquait un décalage, il faudrait recommencer le déchiffrement. Le découpage automatique du crypto en groupes de quatre lettres permet d'éviter assez aisément ce type d'erreur. On aura cependant intérêt à limiter la longueur des messages à quelques centaines de caractères, les tronçons d'un même message étant dotés de groupes date.heure différents.

On trouvera ici (voir ci-dessus) une description pas à pas du programme de chiffrement. Pour l'essentiel, le programme de déchiffrement effectue les mêmes opérations dans l'ordre inverse.

Pierre BARNOUIN

Pour adapter les programmes à d'autres matériels

Si votre ordinateur n'est pas un Canon X-07, vous ne devriez pas avoir de difficulté pour apporter les modifications nécessaires. Voici ce qu'il faut savoir.

- Diminuez le plafond de longueur (2999, lignes 1 et 5, chiffrement) si vous n'avez pas assez de mémoire.
- Il est indispensable de disposer d'une initialisation de séquences RND reproductibles, ce qui est le cas de la plupart des machines.
- En revanche, l'instruction INT permet de suppléer à l'absence de DEFINT, MOD, et ¥ (division entière).
- Utilisez des IF... THEN si vous ne disposiez pas de grandeurs booléennes telles que (K = 0). Celles-ci valent −1 sur le X-07 quand la condition est satisfaite.
- INSTR(Z$,A$) vaut bizarrement 1 et non 0 avec A$ = " ", Z$ étant une chaîne quelconque. Très utile parfois.
- Il est difficile de se passer d'imprimante en cryptographie, mais un modèle bas de gamme suffit.

Cryptographie
Programme pour X-07
Auteur Pierre Barnouin

Chiffrement

```
1 DEFSTRA:DEFINTB-K:DIMB(2999)
2 INPUT"Preparer IMPRIMANTE,CLEF";X
3 INPUT"DATE.HEURE: jj.hhmm";Y
4 PRINT"Taper le TEXTE puis #"
5 K=RND(-X/Y):FORI=0TO2999
6 A=INKEY$:ONINSTR("'#",A)GOTO6,9
7 B=117-ASC(A)+32*(A=" "):IFB<27<>1THEN6
8 PRINTA;:B(I)=2E4*RND(1)+B:NEXT
9 K=RND(-X*Y):X=0:FORJ=0TOI-1
10 K=J+(I-J)*RND(1):B=B(K)+1E4*RND(1)
11 B(K)=B(J):A=CHR$(64+BMOD27)
12 LPRINTA;:IFJMOD4=3THENLPRINT" ";
13 NEXT:LPRINT" ",Y;I
```

Déchiffrement

```
1 DEFSTRA:DEFINTB-K
2 INPUT"Preparer IMPRIMANTE,CLEF";X
3 INPUT"DATE.HEURE: jj.hhmm";Y
4 INPUT"LONGUEUR";L:L=L-1:DIMB(L),H(L)
5 FORI=0TOL:H(I)=I:NEXT:K=RND(-X*Y)
6 PRINT"Taper le CRYPTO":FORI=0TOL
7 A=INKEY$:ONINSTR(" ",A)GOTO7:PRINTA;
8 J=I+(L-I+1)*RND(1):K=H(J):H(J)=H(I)
9 H(I)=K:B(K)=1E4*RND(1)+90-ASC(A):NEXT
10 K=RND(-X/Y):X=0:FORI=0TOL
11 K=(2E4*RND(1)+B(I))MOD27:K=K+32*(K=0)
12 LPRINTCHR$(64+K);:NEXT:LPRINT" ",Y;I
```

Un exemple de message

```
CClef:123456712345671
DQQO TWAF GFRK FEPA FPUF WCNC G@U@ HXQL
LIAB MGTN TYJE WNXH NCLE WWPT MECY ZDUP
JDP@ FAHU UULR JYAP XRIM MLHX PFUD GTNX
OQXR UUUZ E@9M NC@T ZDDF PKYE LIYA BXKK
ZBWU              14.1645  132
```

Le même message, mais en clair

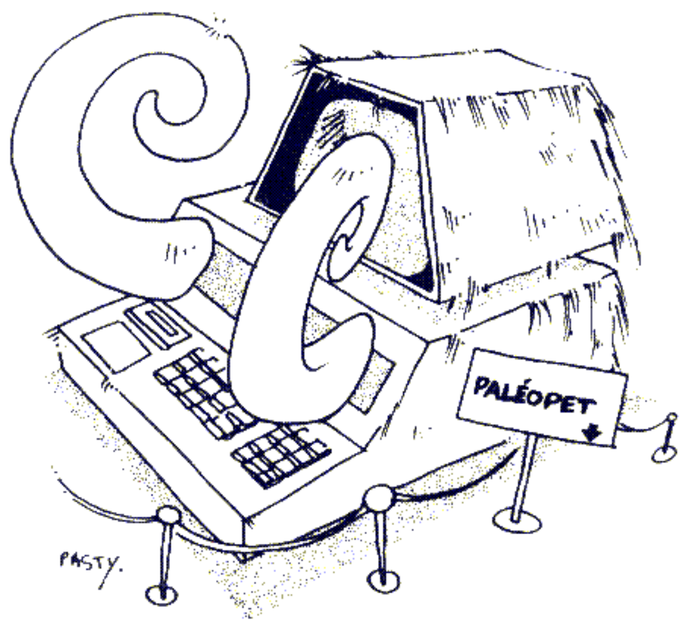
```
SI TOUS CEUX DONT ON LIT LES CHIFFRES DE
VAIENT SE PENDRE LA CORDE DE PENDU N AUR
AIT PLUS AUCUNE VERTU  IL Y EN AURAIT TR
OP  BAZERIES   14.1645  132
```

PARAMÉTRER L'IMPRESSION

Le nez de Cléopâtre, s'il eût été paramétré,
la face du monde en eût été changée.
(Attribué, peut-être à tort, à Blaise Pascal.)

L'OUVERTURE d'un canal périphérique à partir d'un programme ne tient qu'à un code. Sur Commodore, 1 ouvre le canal du lecteur de cassette, 3 celui de l'écran, 4 celui de l'imprimante, 8 celui du lecteur de disquette. Très vite, pour l'impression comme pour toute autre opération, paramétrer devient une nécessité.

Au Musée Archéologique, certains ont peut-être vu un PET premier modèle. Il remonte au Paléosilicien Inférieur. Grâce au carbone 14, on a pu le dater des alentours de 1979. Il comportait un clavier façon calculatrice, un magnétophone intégré, et le tout était surmonté d'un chapeau pointu contenant neuf pouces d'écran. L'ordinateur apparaissait comme un tout indivisible, et il fallait un réel effort pour en séparer les constituants (mentalement, bien sûr ; matériellement, un ouvre-boîte suffit).



Vu au Musée Archéologique

L'ordinateur et son entourage

Avec d'autres matériels, plus ou moins récents, on se trouve devant une collection de petites choses à relier entre elles par des tas de ficelles. Pratiquement, c'est la catastrophe majeure (surtout si on a un doberman). Pédagogiquement, c'est capable de montrer à l'apprenti le plus attardé qu'un ordinateur, c'est une Unité Centrale de Traitement — traduisez : une boîte noire — et que le reste, ce sont des périphériques. Ecran, clavier, imprimante(s), lecteurs de cassettes, de disques durs ou mous, tout ça c'est du périphérique.

A cet égard, la programmation des Commodore est d'une logique rigoureuse. Vous voulez sortir quelque chose sur cassette ? Ouvrez un canal périphérique avec le code 1 pour le lecteur de cassette et sortez ce que vous voulez sortir avec des instructions PRINT #... Vous voulez sortir quelque chose sur imprimante ? Ouvrez un canal périphérique avec le code 4 pour l'imprimante et sortez ce que vous voulez sortir avec des instructions PRINT #... Vous voulez sortir sur disquette ? Sachez que le code du lecteur de disquette est 8. Pour le reste, les différences sont légères. Quant à l'écran, on apprendra sans surprise que l'instruction PRINT n'est qu'un raccourci d'un processus plus complexe qui consiste à ouvrir un canal périphérique avec le code 3 pour l'écran, et à sortir ce qu'on veut sortir avec des instructions PRINT #...

Pour faire entrer un renseignement, c'est la même chose sauf que c'est le contraire : on ouvre un canal avec le code périphérique idoine, et on fait un INPUT #... Naturellement, on ne peut pas entrer depuis une imprimante ni sortir sur un clavier, mais l'INPUT est une entrée par l'écran, qui n'est qu'un raccourci d'un canal ouvert sur écran et d'un INPUT-dièse. Si vous ne me croyez pas, essayez le programme 1, et pensez-y pour sauvegarder vos œuvres d'art cathodiques.

Tout cela pour dire que, dès qu'on touche à un périphérique, d'une opération à l'autre, il y a plus de ressemblances que de différences et qu'une fois de plus, nous sautera aux yeux l'opportunité de... pa-ra-mé-trer !

Un premier exemple simple pour commencer : sortir une table de multiplication, comme dans les articles d'initiation. Si l'on veut l'avoir sur écran, on fera :

```
100 N=7
110 REM
120 FOR I=1 TO 10
130 PRINT I;"FOIS";N;" =";I*N
140 NEXT I
```

Si l'on veut la sortir sur imprimante, on fera :

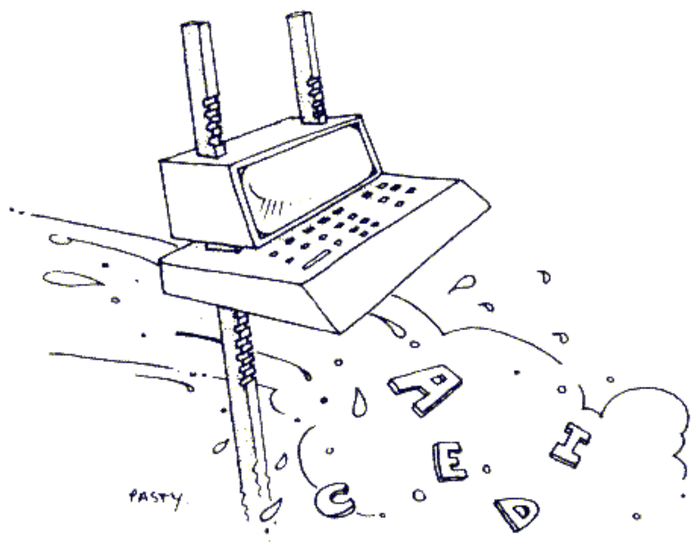
```
100 N=7
110 OPEN 1,4
120 FOR I=1 TO 10
130 PRINT#1,I;"FOIS";N;" =";I*
    N
140 NEXT I
150 CLOSE 1
```

Si l'on a un programme un tantinet plus complexe, avec trois cent douze ins-

LIST - PAGE 30

tructions PRINT, on les changera toutes en PRINT # ... Si, en revanche, on a prévu dès le début qu'on puisse vouloir choisir entre l'écran et l'imprimante, on peut éviter les trois cent douze modifications en se rappelant tout simplement que l'écran est un périphérique comme les autres, et qu'il porte le numéro 3 alors que l'imprimante porte le numéro 4. On ajoute deux lignes et on modifie la ligne 110 :

```
10 INPUT"ECRAN OU IMPRI
   MANTE";P$
```



Ouvrir un canal périphérique

```
20 P=3:IF LEFT$(P$,1)="I"
   THEN P=4
100 N=7
110 OPEN 1,P
120 FOR I=1 TO 10
130 PRINT#1,I;"FOIS";N;"=";
   I*N
140 NEXT I
150 CLOSE 1
```

Souvent, l'imprimante fait 80 colonnes de large, et l'écran 40. Parfait, on paramètre aussi la largeur d'affichage. Si le programme doit séparer divers éléments par un trait horizontal sur toute la largeur, on aura dans les initialisations :

```
30 LA=39:IF P=4 THEN LA=79
```

et dans le cours du programme, le trait sera tracé par :

```
500 FOR I=1 TO LA:PRINT#1,
   "-";:NEXT I:PRINT#
```

Echanger ses programmes

... Encore une variable ! Et quand il s'agit de mettre un titre en valeur, à l'écran, on donne dans l'inversion vidéo ; sur papier, il ne faut pas abuser si l'on tient à sa tête (d'impression). En revanche, les caractères expansés sont du plus bel effet. Or, l'inversion vidéo est déclenchée par un CHR$(18) et s'arrête sur un CHR$(146) ; les caractères expansés de l'imprimante sont déclenchés par un CHR$(1) et s'arrêtent sur un CHR$(129). Qu'à cela ne tienne. Dans les initialisations :

```
30 C1$=  CHR$(18):C0$=CHR$(146)
:IF P=4 THEN C1$=CHR$(1):C0$=
CHR$(129)
```

et dans le cours du programme, le titre à mettre en valeur sera écrit par :

```
500 PRINT#1,C0$;"TITRE";C1$
```

Vous commencez à entrevoir les économies de programme que l'abus des constantes empêchait de faire ?

Quand on échange des programmes avec des correspondants, ils ne sont pas à côté de vous pour vous expliquer comment ils veulent que vous répondiez à telle question, ou ce qu'il faut entendre par telle autre. La meilleure solution : un programme supplémentaire sur la cassette ou la disquette, constitué d'un paquet de PRINT pour apporter des informations sur le contenu de l'autre programme. Là encore, on peut souhaiter le lire sur l'écran ou, si l'on a une imprimante, le tirer une fois pour toutes : c'est plus commode à consulter pendant que l'autre programme tourne. On peut facilement concilier les deux :

```
10 INPUT"ECRAN OU IMPRI
   MANTE";P$
20 P=3:C1$=CHR$(18):C0$=
   CHR$(146):C$=CHR$(13)
30 IF LEFT$(P$,1)="I" THEN
   P=4:C1$=CHR$(1):C0$=
   CHR$(129):C$=CHR$(32)
100 OPEN 1,P
110 PRINT C1$;"TITRE";C0$
120 PRINT#1,"QUARANTE CAR
   ACTERES AU PLUS...";C$;
130 PRINT#1,"QUARANTE AU
   TRES CARACTERES AU
   PLUS..."
140 PRINT#1,"QUARANTE CA
   RACTERES AU PLUS...";C$;
150 PRINT#1,"QUARANTE AU
   TRES CARACTERES AU PLUS
   ..."
500 CLOSE 1
```

On a déjà vu le rôle de C1$ et de C0$.

Programme 1

```
100 PRINT"[clrd]";
110 FOR I=1 TO 20
120 FOR J=1 TO 39
130 X=INT(RND(1)*26+65)
140 PRINT CHR$(X);
150 NEXT J
160 PRINT
170 NEXT I
180 PRINT:PRINT"LECTURE DE L'ECRAN":PRINT
190 DIM A$(20)
200 OPEN 1,3:REM OUVERTURE D'UN CANAL
ECRAN
210 PRINT"[home]";
220 FOR I=1 TO 20
230 INPUT#1,A$(I):REM ENTREE DEPUIS L'ECRAN
240 NEXT I
250 FOR T=0 TO 1000:NEXT T
260  PRINT:PRINT:PRINT:PRINT"RECRITURE  DE
L'ECRAN":PRINT
270 FOR I=1 TO 20
280 PRINT A$(I)
290 NEXT I
```

TAB se meurt, TAB est mort

Le principe qui veut que PRINT-dièse et PRINT soient équivalents, surtout si l'on a ouvert un canal de sortie sur écran connaît une limitation notoire autant qu'irritante, le TAB.

Quand on fait un PRINT normal, donc sur l'écran, un TAB est un TAB. C'est-à-dire qu'en faisant :

```
100 PRINT "ZOZO";
110 PRINT TAB(10);"HULK";
120 PRINT TAB(20);"PROTZ"
```

ZOZO s'affiche en colonne 1, HULK en colonne 10 et PROTZ en colonne 20.

Mais, quand on fait un PRINT-dièse, un TAB n'est plus un TAB, mais un "SPC" :

```
 90 OPEN 1,4:REM SORTIE IMPRI
    MANTE
100 PRINT#1,"ZOZO";
110 PRINT#1,TAB(10);"HULK";
120 PRINT#1,TAB(20);"PROTZ"
130 CLOSE 1
```

ZOZO s'imprime en colonne 1, certes, mais HULK, au lieu de s'imprimer en colonne 10, va s'installer en colonne 14 (longueur de ZOZO + 10), et PROTZ en colonne 38 (14 + longueur de HULK + 20). En d'autres termes, le TAB ne compte plus les espaces à partir de la marge gauche, mais à partir de la dernière position imprimée, bref, il imprime le nombre d'espaces compris entre parenthèses.

Le constructeur est heureux de vous offrir la connaissance approfondie du traitement des chaînes de caractères que ne manqueront pas de vous apporter les heures passées par vous à pallier ces insufficances.

Reconnaissez que ça l'affiche mal !

PARAMÉTRER L'IMPRESSION

Programme 2

```
100 INPUT"ECRAN OU IMPRIMANTE ";P$
110 LE=39:NP=3
120 IF LEFT$(P$,1)="I" THEN LE=80:NP=4
130 OPEN 1,NP
140 :
150 READ L$
160 XX$=XX$+L$
170 IF LEN(XX$)<LE THEN 210
180 IF L$="*" THEN 280
190 GOTO 150
200 :
210 FOR I=LE TO 1 STEP -1
220 IF MID$(XX$,I,1)=" " THEN L=I:I=1:GOTO 240
230 NEXT I
240 PRINT#1,LEFT$(XX$,L-1)
250 XX$=MID$(XX$,L+1)
260 GOTO 170
270 :
280 PRINT#1,LEFT$(XX$,LEN(XX$)-1)
290 CLOSE 1:END
300 :
```

C$ est présent une ligne sur deux et suivi d'un point-virgule. Lorsque la sortie se fait sur écran, C$ contient CHR$(13), un retour-chariot, et les quarante autres caractères s'inscrivent sur la ligne suivante. Mais lorsqu'on sort sur imprimante, C$ contient CHR$(32), un simple espace, suivi d'un point-virgule : les quarante autres caractères s'inscriront sur la même ligne.

```
310 DATA "IL Y A CERTAINES CHOSES QUI
N'ONT L'AIR DE RIEN MAIS QUE L'ETRE "
320 DATA "HUMAIN N'ACCOMPLIT QU'AVEC DE
GROSSES DIFFICULTES. ON TROUVE MEME "
330 DATA "CERTAINES ACTIVITES, APPAREM
MENT SIMPLES COMME BONJOUR, QUI SONT "
340 DATA "PRATIQUEMENT IMPOSSIBLES A EF
FECTUER. AINSI PAR EXEMPLE (MAIS CET "
350 DATA "EXEMPLE N'A PAS ETE CHOISI AU
HASARD), DIRE N'IMPORTE QUOI. "
360 DATA "QUELS QUE SOIENT LES EFFORTS
QUE L'ON CONSENT ET LES PRECAUTIONS "
370 DATA "DONT ON S'ENTOURE, CE QUE L'ON
DIT A TOUJOURS UN SENS."
380 DATA *
```

Quelques pistes pour des améliorations

Il faut penser à terminer chaque ligne de DATA par un espace avant de fermer les guillemets, à moins de le mettre au début de la ligne suivante. Si l'on pose en principe qu'un mot ne sera jamais coupé dans les DATA, le programme peut-il s'occuper de ces espaces ?

Le texte sort sous forme de pavé. On convient de terminer une ligne de DATA par un signe donné (la flèche à gauche, CHR$(95), par exemple) lorsqu'on veut qu'elle termine un paragraphe. Le programme peut-il se charger de forcer un passage à la ligne, sans imprimer la flèche ?

En cas de texte particulièrement long, il va défiler sur l'écran sans qu'on ait le temps de le lire, ou va s'imprimer sur les pliures du papier de l'imprimante. On doit pouvoir installer un compteur de lignes qui, si la sortie se fait sur imprimante, se chargera du saut de page en imprimant six lignes vides lorsqu'une feuille sera remplie, et se remettra à zéro ; et qui, si la sortie se fait sur écran, affichera sur la dernière ligne un message du genre "FRAPPEZ RETURN POUR LA SUITE", et arrêtera le défilement tant que la touche n'aura pas été enfoncée.

Ajouter une ou deux lignes testant, par quelques PEEK judicieux, si l'ordinateur sur lequel tourne le programme est un Vic 20. Dans l'affirmative, l'affichage passe immédiatement à 23 lignes de 22 caractères, au lieu de 25 lignes de 40. Même test pour repérer le CBM 8000, avec passage spontané à 80 colonnes.

Trouver les deux lignes de Basic qui manquent pour transformer ce bref programme en un combiné de *Visicalc, Wordstar* et *Multiplan,* qui joue raisonnablement aux échecs et sache préparer un thé décent, ce qui est rare...

Le procédé est assez rudimentaire et ne demande qu'à être amélioré. Imaginez un peu que le texte à afficher figure dans le programme sous forme de DATA, et soit ensuite découpé par ce programme à une longueur que vous aurez fixée, et que vous pourrez faire varier en modifiant la valeur d'une seule... variable. Soit LE la largeur de l'écran ou la largeur d'une ligne de votre imprimante, et NP le numéro du périphérique de sortie, 4 pour l'imprimante ou 3 pour l'écran.

Si vous voulez afficher sur 15 colonnes, il n'y a, là encore, qu'un chiffre à changer. Le programme 2 peut constituer un point de départ pour vos méditations. Vous commencez par améliorer un embryon, et puis vous finissez auteur du meilleur traitement de texte de l'année.

Elles sont nombreuses, les améliorations qu'on peut apporter (voir encadré "Quelques pistes pour des améliorations"). Chacune d'elle revient à compliquer le programme, à l'alourdir tant parfois qu'il ne tourne plus, la correction d'une erreur en entraînant souvent dix autres. Alors, vous pestez un brin, vous dites que c'est complètement niais, l'informatique, et puis le lendemain, vous vous apercevez que le travail mal fait par douze lignes peut être bien fait par une seule, mieux placée et tellement simple qu'elle sait traiter trois cas au lieu d'un seul. Alors, c'est drôlement chouette, la programmation, et puis, tiens, si j'essayais de demander au programme de me faire...

C'est reparti ! Mais rappelez-vous : le principe de base est simple, il consiste seulement à laisser l'ordinateur faire votre travail.

François J. BAYARD

LA GRANDE MISÈRE DES FONCTIONS INVERSES

La précision des ordinateurs doit être interprétée avec beaucoup de prudence. Si tel modèle est réputé calculer avec 13 chiffres significatifs par exemple, cela signifie en général que les opérations élémentaires, telles que le produit de deux nombres, sont effectuées avec 13 chiffres. Mais il n'en va pas de même pour toutes les fonctions.

Dans un ordinateur, les fonctions mathématiques ne sont pas toutes logées à la même enseigne. Pour certaines, la précision laisse à désirer. Souvent même, cette précision diffère sensiblement entre une fonction et son inverse. Rappelons que si une fonction appliquée à une variable x donne pour résultat y, alors la fonction inverse appliquée à y donne pour résultat x. Ainsi l'inverse de y = log x est x = 10^y ou x = antilog y (les deux symboles sont équivalents). Par exemple, si log 2 = 0, 30103, alors 2 = $10^{0,30103}$ = antilog 0,30103 (1).

Les valeurs de ces fonctions inverses sont parfois — pour ne pas dire généralement — particulièrement maltraitées par les ordinateurs. Cette constatation peut surprendre, mais elle a une explication, sinon une excuse : le calcul interne de la fonction directe et celui de la fonction inverse homologue n'obéissent généralement pas au même algorithme et les précisions correspondantes n'ont aucun lien entre elles. Il faut simplement croire que les constructeurs attachent plus d'importance aux fonctions élémentaires (logarithmiques, trigonométriques, hyperboliques) qu'à leurs inverses. On trouvera, dans le tableau 1 (page suivante), quelques exemples concernant les antilogarithmes décimaux.

La question qui vient alors à l'esprit est la suivante : disposant d'une table de logarithmes acceptable et d'une table d'antilogarithmes médiocre, peut-on se servir de la première pour « réparer » la seconde ? La réponse est oui.

Un bon aller retour

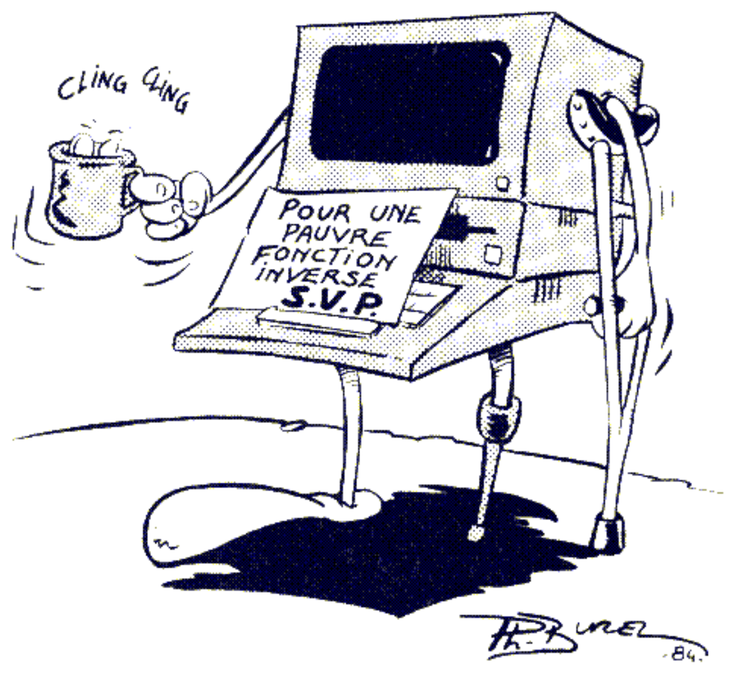
Pour effectuer cette réparation, il est possible d'établir une méthode très générale, applicable à chaque machine et à chaque fonction inverse, et fondée sur un phénomène que vous avez certainement eu l'occasion d'observer vous-même. En Basic, faites afficher SQR(7) ∧2, ce qui correspond à $(\sqrt{7})^2 = 7$.

L'écran indique probablement 7 (il faut l'espérer). Mais si vous entrez SQR (7) ∧2 - 7, il est presque certain que l'écran indiquera autre chose que ce qu'on attend, c'est-à-dire zéro. Sur le PC-1211, par exemple, il indique - 1,5 E - 10 alors que sur le TRS-80, il indique 1,43051 E-06. Ces résultats, incongrus, sont généralement accueillis avec un brin de condescendance, alors qu'ils méritent tout notre respect, car ils forment un outil formidable que nous n'allons pas nous priver d'exploiter.

Ecrire SQR(7)∧2-7, c'est mesurer l'erreur affectant une variable (en l'occurrence 7) lorsqu'elle effectue un aller retour entre une fonction (racine carrée) et son inverse (carré). Nous appellerons cette erreur « écart aller retour » et nous la désignerons par D.

Ce qui est intéressant, c'est l'enseignement que l'on peut tirer de ce paramètre : puisqu'il mesure un écart aller retour, si l'écart aller est nul (ou presque nul), il faut l'imputer intégralement au retour et inversement. Pour revenir à notre point de départ, mesurons D sur un trajet aller retour antilogarithme - logarithme - antilogarithme. Si le calcul du logarithme est correct, D mesure l'erreur commise sur l'évaluation de l'antilogarithme et représente donc la correction à apporter au résultat.

Le programme qui suit et l'équation qui lui correspond donnent une illustration respectivement informatique et mathématique de cette interprétation. Ils prennent pour exemple l'évaluation de la correction à apporter au calcul de x = antilog 0,778 151 250 4. La machine utilisée est un PC-1211, mais



(1) Nous désignons les logarithmes décimaux par log, les antilogarithmes décimaux par antilog et les logarithmes naturels par ln.

LA GRANDE MISÈRE DES FONCTIONS INVERSES

le principe est valable pour toute autre machine, à condition qu'elle traite correctement les logarithmes, même si elle maltraite les antilogarithmes.

Les résultats sont indiqués chiffres de garde compris. Pour les mettre en évidence, écrire l'opération à effectuer et, avant d'appuyer sur la touche ENTER, soustraire les deux premiers chiffres significatifs du résultat dont on connaît l'ordre de grandeur. Appuyer alors sur la touche ENTER : l'écran indique les 10 chiffres suivants. Par exemple, pour mettre en évidence les chiffres de garde de antilog 0,107 209 969 6, taper :

10 ∧ . 1072099696 - 1.2

Le résultat apparaît : 7,999999983 E-2. Ce sont les 10 derniers chiffres du résultat.

Les programmes permettant de calculer D sont extrêmement simples. Pour le calcul des antilogarithmes décimaux, on a le *programme 1 :*

```
500 INPUT Y
510 X = 10 ∧ Y
520 D = X - 10 ∧ LOG X
530 X = X + D
```

Ce programme est dépourvu d'affichage. Cette procédure serait illusoire, car même les machines qui calculent avec 13 chiffres n'en affichent généralement que 10 et l'incidence de la correction n'affecte souvent que les chiffres de garde, non affichés. L'essentiel est de retenir que Y désigne le nombre dont on calcule l'antilogarithme, qui se trouve en ligne 510 avant correction et en ligne 530 après correction. On peut également écrire d'une manière plus compacte :

```
X = 10 ∧ Y + (10 ∧ Y - 10 ∧ LOG (10 ∧ Y))
```

formule qui donne directement l'antilogarithme corrigé et dont la partie consacrée à la correction ne consomme que 18 octets (résistez à la tentation de « simplifier » cette expression en regroupant les deux termes 10 ∧ Y, sous peine de provoquer des erreurs d'arrondi).

Deux types de machines

Le *programme 1* est destiné aux machines dont les précisions de calcul et de stockage sont identiques ; nous dirons que ces machines sont de type A. Mais beaucoup de machines, que nous appellerons de type B, perdent une partie des derniers chiffres significatifs en stockant les résultats dans leurs mémoires. Ainsi le PC-1211 calcule avec 12 chiffres mais ne stocke en mémoire et ne restitue que 10 chiffres. Pour les machines de ce type, la formule que nous avons vue reste valable (à condition de supprimer X =) et nous obtenons le *programme 2 :*

```
500 INPUT Y
510 X = 10 ∧ Y
520 D = 10 ∧ Y - 10 ∧ LOG (10 ∧ Y)
530 A = 10 ∧ Y - X + D
```

Ce programme retourne la valeur affinée de 10 ∧ Y en deux parties : X est la valeur brute extraite de la mémoire X (généralement identique à l'affichage) ; A est la correction globale à ajouter à X, positive ou négative, égale à la somme des chiffres de garde et de la correction D.

Si le *programme 2* est intégré à un programme ou s'il est traité en sous-programme, après l'exécution de la ligne 530, on remplacera X par X + A.

Seuls les antilogarithmes des nombres inférieurs à 1, sont justiciables de la méthode décrite. Pour les nombres supérieurs à 1 ne traiter que leur partie décimale. Par exemple, pour calculer antilog 7,35, il suffit d'écrire antilog 7,35 = 10^7 × antilog 0,35.

Si vous utilisez les *programmes 1* ou *2*, filtrez donc vos entrées pour n'intro-

Tableau 1
Antilogarithmes décimaux

y	Valeur réelle de x = antilog y	PC-1211				TI-58 C			
		Sans correction		Après correction		Sans correction		Après correction	
		Valeur de x (1)	Erreur sur 12e chiffre	Valeur de x (1)	Erreur sur 12e chiffre	Valeur de x (1)	Erreur sur 13e chiffre	Valeur de x (1)	Erreur sur 13e chiffre
0,107 209 969 6	1,279 999 999 | 858 92	...83	– 3	...86	0	...856	– 3	...858	– 1
0,204 119 982 7	1,600 000 000 | 162 38	13	– 3	17	+ 1	159	– 3	162	0
0,301 029 995 7	2,000 000 000 | 165 87	16	– 1	17	0	162	– 4	166	0
0,397 940 008 7	2,500 000 000 | 160 96	13	– 3	16	0	154	– 7	159	– 2
0,477 121 254 7	2,999 999 999 | 864 18	83	– 3	87	+ 1	854	– 10	863	– 1
0,602 059 991 3	3,999 999 999 | 742 46	69	– 5	76	+ 2	732	– 10	748	+ 6
0,698 970 004 3	4,999 999 999 | 585 32	53	– 6	58	– 1	569	– 16	589	+ 4
0,778 151 250 4	6,000 000 000 | 225 97	16	– 7	25	+ 2	202	– 24	226	0
0,845 098 040 0	6,999 999 999 | 770 21	66	– 11	78	+ 1	744	– 26	779	+ 9
0,903 089 987 0	8,000 000 000 | 148 40	14	– 1	16	+ 1	113	– 35	145	– 3
0,954 242 509 4	8,999 999 999 | 185 06	12	– 7	20	+ 1	143	– 42	188	+ 3
0,991 226 075 7	9,800 000 000 | 169 36	07	– 10	18	+ 1	122	– 47	171	+ 2

(1) Valeur calculée par la machine, chiffres de garde compris. Afin de ne pas surcharger le tableau, seuls les derniers chiffres significatifs sont indiqués. Les chiffres omis sont identiques à ceux situés à gauche du pointillé de la deuxième colonne.

duire que des nombres inférieurs à 1, en mémorisant l'exposant à appliquer. Pour couvrir tous les cas, insérez les lignes :

• 505 E = INT Y : Y = Y - E

• pour les machines de type A

535 X = X * 10 ∧ E

• pour les machines de type B

535 X = X * 10 ∧ E : A = A *10 ∧ E

Et si vous ne craignez pas les instructions-fleuves, la première formule qui donnait X devient :

X = (10 ∧ (Y - INT Y) + (10 ∧ (Y-INT Y) - 10 ∧ LOG (10 ∧ (Y-INT Y))) * 10 ∧ INT Y

Elle couvre tous les cas, sans surconsommation de mémoire, et tous les types de machines, tout au moins ceux que nous avons essayés (pour les machines de type B, supprimer X =).

Pour élargir notre étude, nous avons adapté ces formules à une machine dont le langage est aux antipodes du Basic : la TI-58 C, programmable en AOS (Algebraic Operating System). La formule de correction devient très simple. Pour obtenir un antilogarithme décimal, remplacer INV log par :

INV log + (CE - log INV log))

Cela ne coûte que neuf pas de programme par rapport à l'instruction normale (la dernière parenthèse, fermée mais non ouverte, remplace le signe =, à éviter si la formule doit être traitée en sous-programme).

Pour les valeurs supérieures ou inférieures à l'unité, utiliser la formule :

RCL 0 INV Int INV log + (CE - log INV log)) × RCL 0 Int INV log EE INV EE)

dans laquelle le nombre introduit est supposé être en mémoire 0 (zéro). Là encore, les parenthèses fermées mais non ouvertes correspondent au signe =.

Les résultats obtenus à l'aide de la méthode décrite sont consignés dans le tableau 1. Les deux premières colonnes ne constituent qu'une table d'antilogarithmes à 15 chiffres servant de repère. Les autres colonnes indiquent les valeurs des antilogarithmes calculées par le PC-1211 et la TI-58 C, avec les erreurs sur le dernier chiffre significatif, avant et après correction. On peut constater que dans la quasi-totalité des cas, les erreurs produites par les machines sont largement absorbées.

On peut songer à transposer cette méthode de correction aux antilogarithmes naturels, ceux qui correspondent aux logarithmes naturels, ln : y = ln x est équivalent à x = e^y (prononcer « exponentielle de y ») et peut encore s'écrire x = antiln y. Cette transposition est inutile sur la TI-58 C, qui calcule les exponentielles avec une excellente précision (nous n'avons pas décelé d'erreur supérieure à deux unités du treizième ordre). Quant au PC-1211, il donne des résultats à peu près corrects pour les nombres inférieurs à 1, mais les performances se dégradent très vite ensuite (voir tableau 2). On peut tenter d'y remédier avec une formule utilisant les logarithmes naturels :

X = EXP Y + (EXP Y - EXP LN EXP Y)

Mais on constate que la correction, si elle va dans le bon sens, est exagérée. Il convient alors de n'imputer qu'une partie de l'écart aller retour sur la correction, en l'occurrence la moitié. Notre formule devient donc :

X = EXP Y + (EXP Y - EXP LN EXP Y)/2

Ce qui donne un programme pour chaque type de machines.

• Type A

500 INPUT Y
510 X = EXP Y
520 D = (X - EXP LN X) / 2
530 X = X + D

• Type B

500 INPUT Y
510 X = EXP Y
520 D = (EXP Y - EXP LN EXP Y)/2
530 A = EXP Y - X + D

C'est à l'aide de ces formules que nous avons dressé le tableau 2, qui permet de constater une restitution de table des exponentielles correcte, à partir de valeurs incorrectes.

La règle générale

La méthode de correction décrite peut finalement se résumer en une formule très générale : soit une fonction inverse quelconque, désignée par f^{-1}, d'une variable y. Si l'ordinateur fournit un résultat x qu'il y a tout lieu de suspecter, ajoutez à x le terme correctif D :

$D = f^{-1}(y) - f^{-1}(f(f^{-1}(y)))$

Cela donne par exemple, pour certaines fonctions trigonométriques :

ASN Y + (ASN Y - ASN SIN ASN Y), pour arc sin y ;

ACS Y + (ACS Y - ACS COS ACS Y), pour arc cos y ;

ATN Y + (ATN Y - ATN TAN ATN Y), pour arc tg y.

Nous avons vérifié par sondage l'effet bénéfique de ces corrections sur un PC-1211. Par exemple pour arc cos 0,5, la machine calcule (en degrés) :

ACS 0,5 = 60 - 4 **E** - 10

Valeur affinée :

ACS 0,5 + (ACS 0,5 - ACS COS ACS 0,5) = 60 - 1 **E**-10.

La valeur réelle de arc cos 0,5 étant 60, on voit que la correction absorbe les trois quarts de l'erreur d'origine.

En résumé, le mode de correction décrit est tellement élémentaire et sobre en octets, qu'à défaut de le trouver inclus dans la mémoire morte de votre ordinateur, vous pourrez l'incorporer utilement dans vos programmes nécessitant le maximum de précision possible.

Pierre Ladislas GEDO

Tableau 2
Exponentielles

y	Valeur réelle de $x = e^y$		PC-1211			
			Sans correction		Après correction	
			Valeur de x (1)	Erreur sur 12e chiffre	Valeur de x (1)	Erreur sur 12e chiffre
0,2	1,221 402 758	160 17	...14	− 2	...15	− 1
0,4	1,491 824 697	641 27	62	− 2	64	0
0,5	1,648 721 270	700 13	66	− 4	68	− 2
1	2,718 281 828	459 05	44	− 2	45	− 1
2	7,389 056 098	930 65	86	− 7	94	+ 1
4	54,598 150	033 144 2	032 0	− 11	033 2	+ 1
2,302 585 093	10,000 000	000 059 5	000 0	− 1	000 1	0
3,401 197 382	30,000 000	010 135 3	009 4	− 7	010 1	0
4,094 344 562	59,999 999	986 667 4	985 0	− 17	986 4	− 3
4,605 170 186	100,000 000	001 191	000	− 1	002	+ 1
5,298 317 366	199,999 999	890 393	885	− 5	890	0
5,298 317 367	200,000 000	090 393	085	− 5	090	0

(1) Valeur calculée par la machine, chiffres de garde compris. Afin de ne pas surcharger le tableau, seuls les derniers chiffres significatifs sont indiqués. Les chiffres omis sont identiques à ceux situés à gauche du pointillé de la deuxième colonne.

FORTH POUR SPECTRUM

SI LIST, par ses articles enthousiastes sur le Forth, a réussi à vous convaincre de la puissance et de la rapidité de ce langage, vous serez très certainement intéressé par son implantation sur votre Spectrum. A condition, bien sûr, que vous disposiez d'un lecteur de micro-cartouches...

Plusieurs versions de Forth sur Spectrum sont disponibles dans la logithèque déjà très riche de ce petit ordinateur. Mais le Forth présenté ici a l'avantage de fonctionner avec les microdrives, non seulement pour charger le vocabulaire mais aussi pour les chargements et sauvegardes de blocs ou d'écrans. C'est une orientation encore relativement rare, peut-être en raison du prix exorbitant des cartouches pour microdrives.

L'ensemble du logiciel est présenté dans un classeur cartonné de petit format, dans lequel sont insérées trois pochettes plastiques. La première contient deux cartouches de microdrives, l'une référencée *SMF 1.1 Source*, l'autre *SMF 1.1 Blocs*. La seconde pochette contient une documentation succincte de sept feuillets imprimés. Ceci pourrait paraître très insuffisant si la troisième pochette ne contenait un livre de 260 pages, « Programmer le Forth » de Robert Van Loo (aux éditions Marabout).

Cette présentation « artisanale » (sans nuance péjorative, bien au contraire), est ici un gage de qualité. La société Sémaphore importe des logiciels d'Angleterre et ne se contente pas de livrer les produits à l'état brut : elle les adapte à notre langue tant au niveau des messages des programmes que de la documentation. De plus, la société s'engage à communiquer les éventuelles erreurs aux acheteurs dûment enregistrés chez eux.

Un noyau de base de 7 455 octets

La cartouche *Source* contient un petit programme « run » permettant de charger le code du Forth (sur la même cartouche) et de l'initialiser. Le programme « run » est enregistré 80 fois sur la même cartouche et le code l'est six fois. Cela peut paraître coûteux de réquisitionner toute la place d'une cartouche de 90 Koctets pour un contenu réel d'environ sept Koctets. Mais cet apparent gaspillage fait que l'on se trouve presque instantanément sous Forth après avoir tapé le mot-clé RUN. Par ailleurs, cela donne une relative sécurité quant à l'effacement accidentel de certaines portions de la bande. Cependant la grande vulnérabilité des cartouches ne vous promet pas de disposer éternellement de votre Forth. Une copie de sauvegarde n'est malheureusement pas proposée, ni même la possibilité d'en réaliser une soi-même par des moyens classiques.

Le noyau du Forth de base occupe 7455 octets et non pas les 7443 octets annoncés dans la documentation.

La cartouche *Blocs* a subi un formatage légèrement différent de celui normalement réalisé par l'instruction FORMAT du Basic. Un catalogue de cette cartouche imprimera des informations incompréhensibles. Pour avoir un catalogue lisible vous devrez le faire sous Forth par l'intermédiaire du mot INDEX prédéfini.

Sur cette cartouche, vous trouverez quelques utilitaires pour accroître le vocabulaire de base. Ainsi, en plus d'un éditeur ne faisant pas partie du noyau de base, les concepteurs du logiciel ont ajouté :

- un bloc d'extension graphisme et sons permettant de récupérer les fonctions présentes en Basic ;
- un bloc d'extension arithmétique double précision ;
- quelques blocs d'un intérêt mineur dont un programme de démonstration recherchant les 92 solutions du problème des huit reines aux échecs ;
- et, surtout, un programme de formatage de cartouches sous Forth.

Ce formatage est indispensable pour utiliser une cartouche qui contiendra les

écrans que vous écrirez. Pour que la cartouche puisse être conforme aux exigences du code Forth, il faut avoir 190 secteurs de 512 octets contigus sur la bande, sans discontinuité. Ceci peut parfois être difficile à réaliser en raison de la boucle au niveau de la bande sans fin. Ce peut être même impossible pour certaines cartouches et vous devrez alors utiliser les cartouches réfractaires pour le Basic, ainsi qu'il est précisé dans la notice. Au cas où le formatage aurait réussi vous disposeriez ainsi de 85 blocs de 1 Koctet par cartouche.

Outre ces blocs déjà présents, la société Sémaphore annonce la sortie imminente de blocs plus sophistiqués : assembleur moniteur, virgule flottante, etc.

Le Forth ainsi implanté est au standard Fig Forth 79. Il est donc tout à fait transportable si on n'utilise pas les mots non standardisés des blocs précités.

Se prémunir contre les pirates

Les mots non présents par rapport à ceux de l'ouvrage de Robert Van Loo sont mentionnés dans la documentation. Ils sont rares mais la liste est néanmoins incomplète. Dans l'énumération des mots manquants sont oubliés les mots POP, PUSH, R, etc.

Deux ou trois mots ont une syntaxe quelque peu différente. Par contre des mots du noyau de base du Forth Spectrum non mentionnés dans le livre ne sont pas explicités et nécessiteront peut-être un autre ouvrage de référence sur le Forth pour leur compréhension.

Malgré ces petites lacunes, les informations disponibles permettent rapidement de jongler avec tous les mots et les piles ! L'atout principal de cette version de Forth est le dialogue possible avec les microdrives. Cette mémoire de masse d'accès relativement rapide permet d'obtenir une souplesse et une rapidité dans l'esprit du Forth.

Dans le but de se prémunir contre les « pirates » (bien mal cependant) un redémarrage à froid est effectué lors d'appel de routines en mémoire morte qui entraîne normalement un compte rendu d'erreur en Basic pour certains arguments. On peut le vérifier rapidement avec le bloc d'extension graphique et la fonction PLOT par exemple. Il faudra tenir compte de ceci et sauvegarder le programme avant de le tester.

Configuration Forth

Pour réaliser votre propre configuration Forth, vous pouvez faire en sorte que le vocabulaire que vous avez défini devienne partie intégrante du vocabulaire primitif.

Après avoir chargé le Forth à partir de la cartouche originale, exécutez les étapes suivantes :

```
FORTH DEFINITIONS HEX
' TASK NFA FENCE !
FORGET TASK
```

Si, par exemple, vous désirez disposer de l'éditeur dès le chargement du programme-source, faites :

```
0A LOAD
: TASK ;
DP @ DUP 619A ! 619C !
' TASK NFA 618E !
713E 619E !
COLD
```

Notez bien la valeur donnée par « HERE U. »

Disposez ensuite d'un écran, soit le 1 s'il est vierge, puis :

```
EDITOR
1 CLEAR
1 P HEX
2 P CREATE RES FF2A, 2BFF, 2BF9, 222B, 5C3D,
3 P 32AF, 5C71, 01CD, CB16, 0D6E, 3B21,
4 P CB5C, 239E, EECB, FDFB, 3136,
5 P C302, 12A9, SMUDGE ; S
1 LOAD
```

Vous pouvez maintenant exécuter RES et, en appuyant sur ENTER, vous êtes sous Basic.

Prenez alors une cartouche vierge et tapez en mode commande SAVE *"m";1;"forth1" CODE 24576, longueur.

La longueur correspond à la différence entre la valeur « HERE U. » trouvée précédemment et la valeur 24576.

Faites ensuite : VERIFY *"m";1;"forth1" CODE

Après avoir réinitialisé le Spectrum par un PRINT USR 0 ou une coupure secteur, il ne reste plus qu'à recréer l'interface Basic de chargement du programme-source Forth en entrant ces lignes :

```
10 CLEAR #
20 LOAD *"m";1;"forth1" CODE : RANDOMIZE USR 24606
```

et faire en mode commande :

```
SAVE *"m";1;"forth1" LINE 10
```

puis

```
VERIFY *"m";1;"forth"
```

Vous disposez alors sur la cartouche de votre propre version Forth, que vous chargerez tout simplement par :

```
LOAD *"m";1;"forth"
```

La société Sémaphore signale le comportement parfois imprévisible de l'interface ZX1 afin de dégager sa responsabilité en cas de mauvais fonctionnement. C'est vrai, elle comporte des bogues ! Pour en corriger certaines, Sinclair a réalisé une deuxième version de l'interface ZX1. Cette version n'est, aux dires de l'importateur, pas encore arrivée en France, mais il faut savoir dès maintenant que certaines adresses de routines de la mémoire morte de l'interface ZX1 ont été modifiées. Le noyau Forth de base fonctionnera encore correctement car il n'utilise que les *Hook Codes*. Mais le bloc de formatage ne fonctionnera pas car il utilise des adresses absolues dans la mémoire morte de l'interface ZX1. Espérons que Sémaphore tiendra compte de ce problème.

Un Forth compatible microdrive, y compris pour les entrées-sorties, permet d'aller plus loin et plus vite dans la programmation en Forth. Ce logiciel est assez satisfaisant malgré quelques lacunes et surtout un prix d'achat (500 FF ttc) très différent de ceux habituellement pratiqués pour les logiciels Spectrum.

Le logiciel en quelques lignes

Nom : *Forth 1.1*
Ordinateur : *ZX Spectrum*
Forme : *deux cartouches de microdrives*
Edité et distribué par : *Sémaphore Logiciels (La Plaine-Genève, Suisse)*
Prix public : *500 FF*
Principale orientation : *langage Forth*

Benoît THONNART

SIMONS' BASIC
BASIC ÉTENDU
POUR COMMODORE 64

NOMBREUX sont les utilitaires destinés à apporter au Basic du Commodore 64 un peu de puissance, pour ne pas dire d'efficacité ! L'un d'eux, Simons' Basic, détonne par l'étendue de ses possibilités : il enrichit le Basic de 114 commandes et instructions supplémentaires permettant de gérer sons et graphismes, et de mettre au point avec facilité des programmes structurés, bref, de tout faire avec une souplesse inégalée. Tel est donc Simons' Basic, utilitaire majeur destiné au C.64.

Edité par Commodore, et distribué en France par Procep, *Simons' Basic* passe souvent pour être *le* programme utilitaire qu'il faut connaître. Voyons de plus près...

Présenté sous la forme d'une classique cartouche enfichable à l'arrière du C.64, sa mise en œuvre est donc d'une parfaite simplicité. Dès que la cartouche est en place et l'unité centrale allumée, *Simons' Basic* devient intégralement disponible.

Lors de notre premier contact avec cet utilitaire, nous avons trouvé la cartouche dans une boîte cartonnée, accompagnée d'un livret grand format. A l'ouverture de la boîte, un second livret (petit format, celui-là) est apparu. Peu habitués à cette profusion de documentation, et étonnés comme il se doit, nous avons donc observé le phénomène plus attentivement pour découvrir que les deux livrets avaient exactement le même contenu... Le plus petit étant rédigé en anglais !

Cette intéressante particularité est une preuve des efforts consentis par Procep pour la francisation des logiciels Commodore. C'est aussi une explication au fait que les prix de certains logiciels sont plus élevés en France. Mais pourquoi donc avoir laissé dans l'emballage la documentation en anglais ?

Comme nous en avons l'habitude, un coup d'œil rapide au manuel d'utilisation nous donnera les moyens de maîtriser plus facilement le logiciel. Plus enclins à employer notre langue maternelle, nous prenons donc celui qui est rédigé en français.

Un manuel bien fait

Il nous apprend que 114 nouvelles commandes viennent muscler le Basic quelque peu anémique du C.64 ; et ceci dans une douzaine de domaines différents : aide à la programmation, édition de texte et validation d'entrées, aides numériques, gestion des disquettes, graphismes, manipulations d'écran, *sprites* et caractères, programmation structurée, gestion des erreurs, commandes musicales, commandes de périphériques de « jeu ».

Le manuel est divisé en chapitres correspondant à chacune de ces grandes directions d'emploi. L'utilisateur non averti ne peut manquer d'être surpris par la variété du champ d'application présentée pour chacune des instructions nouvelles.

Le manuel est bien conçu, clair et précis. Seuls manquent quelques détails permettant à l'utilisateur de retrouver facilement une instruction particulière : rien n'attire l'attention sur elle, si ce n'est le numéro qui la précède. Un index ou une table alphabétique aurait été bien utile.

Pour chaque instruction, on trouve : ses caractéristiques syntaxiques, une description de ses effets, un ou plusieurs exemples d'utilisation très précis, décrivant les actions de l'utilisateur et les résultats obtenus.

Le livret se termine par quelques exemples de programmes destinés à illustrer les possibilités du *Simons' Basic,* et une table des dix nouveaux messages d'erreur spécifiques de cette extension.

Le grand format du manuel et sa reliure spirale le rendent facile à manipuler. C'est un avantage sur le manuel d'origine en anglais.

Evidemment, nous ne pouvons pas

décrire ici chacune des 114 nouvelles fonctions : un numéro entier n'y suffirait pas !

Voyons plutôt quelques-unes des fonctions les plus spectaculaires (ce ne sont d'ailleurs pas forcément les plus intéressantes pour tout le monde, question de goût !). Le mieux est de comparer ces fonctions avec ce que l'on devrait faire pour obtenir le même effet en Basic standard du C.64.

Plus rapide et plus simple

Prenons par exemple l'instruction qui permet de mettre en place la haute résolution graphique. En *Simons'*, l'instruction adéquate est HIRES *T, F...* dans laquelle T représente la couleur du tracé et F la couleur du fond.

En Basic standard, il aurait fallu écrire :

```
100 POKE 53265,59
110 POKE 53272,24
120 FOR E = 1024 TO 2023 : POKE E,
    T + 16 * F : NEXT E
130 FOR E = 8192 TO 16383 : POKE
    E, 0 : NEXT E
140 POKE 53280,F
150 END
```

La lenteur exaspérante d'un tel programme rend d'autant plus précieuse l'instruction correspondante HIRES.

Second exemple, l'allumage de points sur l'écran haute résolution est loin d'être une sinécure en Basic :

```
100 INPUT X, Y
110 IF X<0 OR X>319 OR Y<0 OR
    Y>199 THEN 100
120 X1 = 8 * INT (X/8)
130 Y1 = 320 * INT (Y/8)+(Y AND 7)
140 X2 = 2! (7-(X AND 7))
150 P = 8192 + X1 + Y1
160 POKE P,PEEK(P) OR X2
170 END
```

...Mais ça devient un vrai plaisir avec *Simons'* grâce à PLOT *X, Y*. Et ne parlons pas des tracés de lignes et de cercles dont le traitement en Basic suffit à faire perdre son sang-froid à tout individu, même très calme !

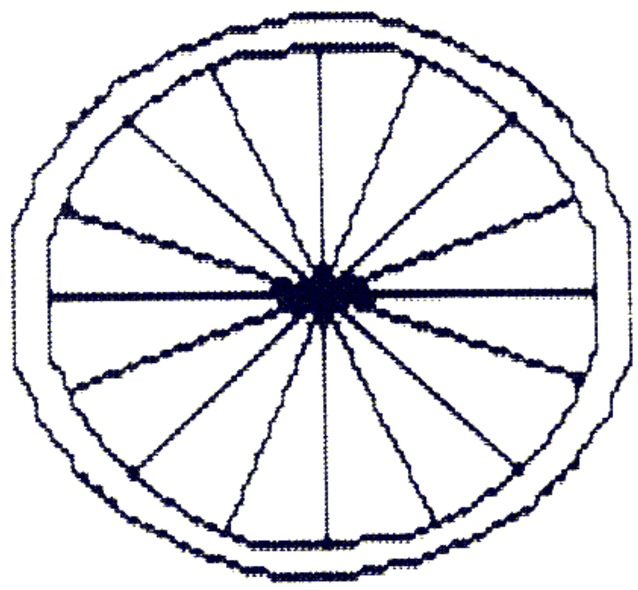
De plus, *Simons'* autorise les conversions binaire-décimal et hexadécimal-décimal avec une vitesse inégalée.

Il est possible de combiner simplement ces deux fonctions pour des calculs arithmétiques entre bases différentes : PRINT %10110101 + $EB38, addition d'un nombre binaire et d'un nombre hexadécimal, rend pour résultat 60397 en décimal.

Pour réaliser cette opération, un programme Basic normal devrait par exemple effectuer la conversion binaire-décimal, puis la conversion hexadécimal-décimal, pour enfin pouvoir effectuer l'opération et rendre le résultat. Beaucoup d'efforts pour peu d'effet. Il est vrai que nous n'avons pas besoin tous les jours de ce genre de possibilités...

```
*** EXTENDED CBM V2 BASIC ***
38719 BASIC BYTES FREE

READY.
10HIRES 0,1
20CIRCLE 160,100,40,40,1
30CIRCLE 160,100,45,45,1
40FOR X=0 TO 360 STEP 22.5
50ANGL 160,100,X,40,40,1
60NEXT
100GOTO100
```



En sept lignes, Basic épaulé par Simons' permet le tracé d'une roue

Plus courante d'emploi est la fonction MOD (X, Y) qui rend le reste de la division de X par Y. En Basic, la formule est : PRINT X – INT(X/Y)*Y. Elle est là encore beaucoup plus efficace avec *Simons'*.

Concernant la programmation structurée, *Simons'* est capable d'apporter aux programmeurs des habitudes de programmation meilleures, et d'éliminer bien des « sacs de nœuds » présents dans des programmes écrits sans réflexion préalable, à grands renforts de GOTO.

Pour cela, IF...THEN...ELSE est encadré de REPEAT...UNTIL, LOOP... EXIT IF...END LOOP, tandis que PROC et END PROC, combinés à CALL et EXEC permettent de définir et d'utiliser des PROCédures.

Pour les amateurs de jeux, PENX, PENY, JOY et autres POT(X) fournissent un arsenal précieux.

Quant aux amateurs de musique, ou à ceux qui l'auraient été si le C.64 avait bien voulu leur faciliter la tâche, ils ne sont pas oubliés : les commandes WAVE, VOL, ENVELOPE, MUSIC, PLAY sont là pour le plaisir de la programmation et celui des oreilles...

Nous pourrions continuer encore longtemps cette énumération : *Simons' Basic* n'est pas un utilitaire comme les autres, que l'on teste en quelques jours. Pour lui, une longue pratique est indispensable. Nous n'avons pu, pendant la durée de notre test, qu'approcher certaines possibilités de cette cartouche,

Liste des mots réservés de Simons' Basic

ANGL
ARC
AT
AUTO

BFLASH
BFLASH 0
BLOCK

CALL
CENTRE
CGOTO
CHAR
CHECK
CIRCLE
CMOB
COLD
COLOUR
COPY
CSET

DELAY
DESIGN
DETECT
DIR
DISABLE
DISAPA
DISK
DISPLAY
DIV
DRAW
DUMP
DUP

END PROC
ENVELOPE
EXEC
EXOR

FCHR
FCOL
FETCH
FILL
FIND
FLASH
FRAC

GLOBAL
GRAPHICS

HI COL
HIRES
HRDCPY

IF...THEN...ELSE
INKEY
INSERT
INST
INV

JOY

KEY

LIN
LINE
LOCAL
LOOP...EXIT IF...END LOOP
LOW COL

MEM
MERGE
MMOB
MOBOFF
MOB SET
MOD
MOVE
MULTI
MUSIC

NO ERROR
NRM

OFF
OLD
ON ERROR
ON KEY
OPTION
OUT

PAGE
PAINT
PAUSE
PENX
PENY
PLACE
PLAY
PLOT
POT
PROC

RCOMP
REC
RENUMBER
REPEAT...UNTIL
RESET
RESUME
RETRACE
RLOCMOB
ROT

SCRNLD
SCRSY
SCROLLING
SECURE
SOUND

TEST
TEXT
TRACE

USE

VOL

WAVE

@
$
%

...et ceux que nous avons peut-être oubliés.

sans pouvoir même en effleurer d'autres. La richesse de ce logiciel est simplement exceptionnelle. Mais nous avons fait quelques amusantes expériences non prévues dans le manuel (ou trop bien cachées) : par exemple, GRAPHICS est une variable réservée dont la valeur est 53248 (première adresse du registre vidéo). Il suffit de taper PRINT GRAPHICS, pour vérifier. De même pour SOUND, qui vaut 54272 (première adresse du registre son).

Les programmeurs apprécieront

L'instruction MEM, utilisée pour la redéfinition des caractères peut être annulée par NRM (NoRMal), tandis que le manuel indique seulement STOP/RESTORE pour revenir à l'état antérieur.

Nous avons découvert que le mode TRACE devient inopérant lorsque MEM a été exécuté. De la même façon, il peut être intéressant de savoir que la fonction AT est utilisable dans une instruction du type : A$ = AT (10, 15) + "texte" : PRINT A$.

Les utilisateurs du *Simons' Basic* ont donc encore des choses à découvrir... Et que dire à ceux qui ne le possèdent pas ? A eux de voir si la puissance et la fonctionnalité de ce type de logiciel peuvent leur être utiles. Il faudra, bien entendu, qu'ils soient des programmeurs acharnés pour en tirer le parti maximum. Avec un manuel bien conçu, voici donc un fantastique programme qui, à notre connaissance, est le seul à permettre une utilisation aussi « totale » du C.64.

Le logiciel en quelques lignes

Nom : Simons' Basic
Ordinateur : Commodore 64
Forme : cartouche
Edité par : Commodore
Distribué par : Procep
Prix public : 750 FF ttc, avec manuel en français
Principales orientations : extension du Basic ; fonctions graphiques, de musique, d'aide à la programmation, de calcul.

Seules ombres au tableau : les programmes conçus à l'aide du *Simons' Basic* ne peuvent plus être utilisés sans lui. Vous aurez donc du mal à faire profiter vos amis de vos productions, si eux-mêmes ne le possèdent pas. Enfin, son prix relativement élevé : 750 FF ttc. Le confort se paie !

Robin BOIS

SPL
UN ASSEMBLEUR
POUR LE DAI PC

LE logiciel SPL, comme disent les initiés, est un des cinq éditeurs-assembleurs de langage-machine qui « tournent » sur le DAI PC. Sans doute est-il le plus puissant, car il se veut très proche de MACRO 80, un assembleur tout à fait professionnel.

SPL est livré sous forme de cassette audio (ou micro-cassette numérique) avec plusieurs utilitaires, dont DISPLAY, un très puissant désassembleur et TRANSLATOR, qui permet de récupérer des sources écrites avec les autres assembleurs. Voilà qui est bien pratique. La notice, en français, est longue de 34 pages ; c'est dire que les commandes sont nombreuses et qu'il convient de lire et relire le texte avant de prétendre maîtriser ce logiciel. Cela étant, *SPL* offre une grande souplesse d'emploi.

Le droit à l'erreur

D'abord, sa façon de compacter le fichier-source autorise la compilation de programmes très longs, avantage décisif pour toute application « sérieuse » : on peut assembler en une seule fois jusqu'à 12 Koctets de codes-machine ! Ensuite, l'écriture de la source sous éditeur plein écran (avec tabulation automatique pour chacune des zones labels, opérandes, etc.) simplifie grandement le travail : il est possible de déplacer le texte-source dans toutes les directions, de se positionner n'importe où pour corriger.

Le premier déverminage (ou débogage) se fait en sortie d'édition : en cas d'erreur de syntaxe, *SPL* réédite la source à partir de la ligne fautive, et le curseur est remplacé par une lettre clignotante mnémonique du type d'erreur détecté. Ce système fait gagner beaucoup de temps lors de l'écriture d'un programme. Les possibilités de travail sont très grandes, et il est hors de question, en quelques lignes, de les passer toutes en revue. Les grands « classiques » sont là, avec souvent un « plus » : copie ou déplacement de paragraphes, recherche ou remplacement d'étiquettes, paramétrage de la liste (nombre de lignes par page; affi-

chage en décimal, hexadécimal, octal, binaire, au choix, des adresses et/ou des opérandes). J'ai beaucoup apprécié la possibilité de demander un segment de liste ou d'édition par numéro de ligne ou par étiquette. Ainsi, la commande L 10 FIN déclenche-t-elle la liste de la source de la ligne 10 jusqu'à l'endroit où se trouve l'étiquette FIN. L'inconvénient d'une telle richesse est l'abondance des commandes : il vaut mieux garder le manuel près de soi pour s'y retrouver, du moins au début !

La liste 1 montre un exemple de source *SPL*, utilisant l'assemblage conditionnel, la gestion dynamique d'étiquettes, les macro-instructions, avec ou sans passage de paramètres. Ce programme est destiné à mesurer le temps que le Dai met à parcourir 255 instructions NOP (1). Les lignes 21 à 25 montrent comment expanser 255 instructions en peu de place : c'est l'assembleur qui fait la boucle en gérant dynamiquement l'étiquette COMPT.

Le début de liste (lignes 9 à 14) est un exemple d'assemblage conditionnel, utile pour faire plusieurs versions d'une même source. Ici, l'étiquette VERS (version) vaut 1, le début du code-machine sera compilé à partir de 400 (hex), sinon, à partir de 500 (hex). Ceci peut être encore affiné, car *SPL* permet les tests booléens sur les étiquettes, du genre IF VERS = 1 AND LABEL = 2 OR VERS < > 5, etc. Les macro-instructions peuvent aussi passer des paramètres (exemple : store passe l'adresse RESULT dans l'étiquette RE lors de l'expansion de la macro). La ligne 2 montre comment donner des directives à l'imprimante, lors du listage de la source. Ici, on commande le passage en écriture grasse sur une Epson.

L'Assembleur n'est pas tout

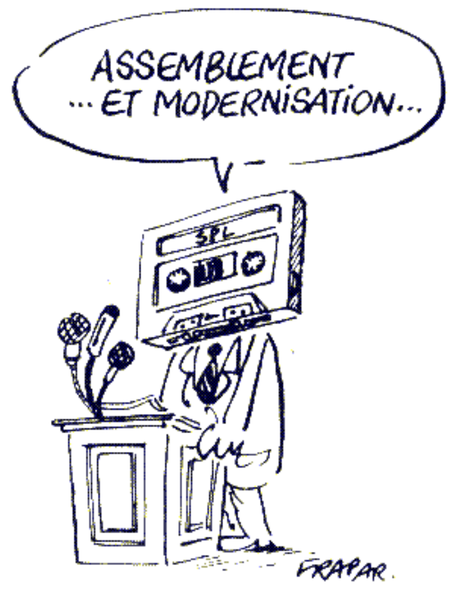
L'utilitaire DISPLAY, livré avec *SPL*, ajoute des fonctions à l'assembleur : il permet d'obtenir la liste hex et ASCII d'une zone mémoire. Il désassemble aussi un code-machine en recréant une source utilisable par *SPL*, avec des étiquettes, s'il vous plaît ! Sur la liste 2, on voit comment apparaît le désassemblage de la routine « sortie série » du Dai. Remarquez les étiquettes placées aux points de branchement de la routine. DISPLAY peut désassembler *SPL* lui-même (12 Ko de codes-machines !) et le tout (code et source) tient encore en mémoire vive. Enfin,

*(1) Si vous désirez savoir combien de temps le Dai met à parcourir 255 NOP, faites, sous Basic : PRINT (#FFFF - PEEK (#300) - (PEEK (#301) *256)) *20 ; "millisecondes".*

En fait, ce n'est qu'à partir de plusieurs milliers de NOP que la décrémentation devient visible.

Liste 1 : programme-source obtenu avec SPL

```
1       ;
2               PRT     1BH,45H         ;directives imprimante
3   ;
4               TITL     ESSAI SPL
5       ;
6               PUT     "H"             ;affichage hexadecimal
7               ORG     300H
8       TEMPS   DW      0FFFFH          ;initialisation
9       VERS    SET     1H              ;version 1
10              IF      VERS=1H
11              ORG     400H
12              ELSE
13              ORG     500H
14              ENDIF
15      ;
16      TIMER   EQU     1BEH            ;horloge du DAI
17      RESULT  EQU     300H            ;tampon du resultat
18      ;
19              push                    ;appel de macro
20              time
21      COMPT   SET     0FFH            ;label gere dynamiquement
22      loop    ###                     ;pt d entree de l expansion
23              NOP
24      COMPT   SET     COMPT-1H        ;decrement du label
25              loop    COMPT>0H        ;boucle tant que label>0
26      ;
27              store   RESULT          ;macro avec parametre
28              pop
29              RET
30      FIN     END
31      ;
32      ;zone des macro-instructions
33      ;
34      push    MACRO
35              PUSH B
36              MEND
37      time    MACRO
38              LHLD    TEMPS
39              SHLD    TIMER
40              MEND
41      store   MACRO   RE              ;passage du parametre
42              LHLD    TIMER
43              SHLD    RE
44              MEND
45      pop     MACRO
46              POP B
47              MEND
48      ;
```



Liste 2 : exemple de désassemblage d'une routine

```
0000                    ;
0000                    PRT         1BH,45H
0000                ;
0000                    ORG         0DD94H
DD94  F5      LHDD94    PUSH PSW
DD95  3A00FD  LHDD95    LDA         LHFD00
DD98  E608              ANI         8H
DD9A  CA95DD            JZ          LHDD95
DD9D  3AF3FF  LHDD9D    LDA         LHFFF3
DDA0  E610              ANI         10H
DDA2  CA9DDD            JZ          LHDD9D
DDA5  F1                POP PSW
DDA6  32F6FF            STA         LHFFF6
DDA9  FE0D              CPI         0DH
DDAB  C0                RNZ
DDAC  F5                PUSH PSW
DDAD  3E0A              MVI A       0AH
DDAF  CD94DD            CALL        LHDD94
DDB2  F1                POP PSW
DDB3  C9                RET
DDB4  à=FD00  LHFD00    EQU         0FD00H
DDB4  à=FFF3  LHFFF3    EQU         0FFF3H
DDB4  à=FFF6  LHFFF6    EQU         0FFF6H
DDB4                    END
```

SPL, UN ASSEMBLEUR POUR DAI

l'utilitaire IMPLEM permet de paramétrer *SPL*, si les options par défaut ne vous conviennent pas, et ce, sans POKEs fastidieux.

La fonction LINK (« accrochage », lors de la compilation, de sous-routines en bibliothèque) est absente de *SPL*. Elle est remplacée par un MERGE (commande Y) qui inclut dans la source de travail des segments provenant de la mémoire de masse.

L'interactivité de *SPL* se traduit en (nombreux !) messages signalant telle ou telle anomalie : fautes de syntaxe, fautes de structure (lors de la compilation), fautes d'introduction (commande erronée), etc. De plus, à chaque retour à *SPL*, un *check sum* de la source, du code et de *SPL* lui-même vérifie l'intégrité de ce qui est en mémoire (vous savez : un programme « pas tout à fait au point » et que l'on essaie, peut occasionner des ravages sournois !).

Voilà donc un assembleur très réussi et puissant qui ne dépayserait pas un utilisateur venant de machines n'ayant rien à voir avec un micro-ordinateur. Associé à *DDT (DAInamic debugging tool,* un excellent utilitaire de mise au point), *SPL* se devrait de figurer dans la panoplie de tout Daïste féru de langage-machine.

Alain MARIATTE

Le logiciel en quelques lignes

Nom : *Assembleur 8080*
Ordinateur : *Dai PC et Dai T (version professionnelle)*
Forme : *cassette audio ou cassette numérique*
Edité et distribué par : *Dainamic Mottaart - 20 3170 Herselt Belgique*
ou : *Dainamic France - 9, rue Lavoisier, 59140 Dunkerque*
Prix public : *1 100 F belges (165 F français, environ)*
Orientation principale : *assemblage de langage-machine*
Autres orientations : *désassembleur, translateur de sources, dump-mémoire*

LES COUPS D'ŒIL DE LIST

INITIATION AU LANGAGE BASIC POUR TO 7

PLUS qu'une initiation, c'est un cours de Basic appliqué au TO 7 qui est proposé en six volumes (douze cassettes) réalisés et distribués par Vifi-Nathan. L'ordinateur donnant des leçons d'informatique : un bon exemple d'Enseignement Assisté par Ordinateur.

Chacun des six volumes comporte deux cassettes et un fascicule d'une vingtaine de pages composé de fiches pratiques où sont résumées les instructions traitées et leur utilisation. Un lexique reprend les termes employés pour la première fois dans la leçon et une série d'exercices d'application est proposée en complément. On a droit a trois ou quatre réponses inexactes avant

Instructions traitées dans l'initiation au langage Basic de Vifi-Nathan

Vol. 1	Vol. 2	Vol. 3	Vol. 4	Vol. 5	Vol. 6

BOX
BOXF
CLS
COLOR
LINE
LOCATE
PLAY
PRINT
PSET
SCREEN
	AUTO
	CONT
	DELETE
	END
	LIST LOAD
	NEW
	REM
	RUN
	SAVE
	STOP
		EXPRESSIONS
		OPERATEURS
		VARIABLES
			DATA
			GOTO
			IF THEN
			ELSE
			INPUT
			INPUT PEN
			INPEN
			LINE INPUT
			READ
			RESTORE
				DIM
				FOR NEXT
				STEP
				GOSUB
				RETURN
				ON X
					ATTRB
					CONSOLE
					DEFGR$
					GR$
					INSTR
					LEFT$
					LEN
					MID$
					PRINT USING
					RIGHT$
					TAB

Le logiciel en quelques lignes

Nom : Initiation au langage Basic
Ordinateur : TO 7, TO 7-70
Forme : Cassettes (série de 6 coffrets contenant chacun deux cassettes et un livret d'accompagnement)
Edité par : Vifi-Nathan
Prix public : 195 FF ttc le coffret
Principale orientation : initiation à la programmation en Basic

que le programme ne donne la solution.

Quelques secondes suffisent au chargement de chaque leçon (un chargement en musique : le magnétophone du TO 7 travaillant sur deux pistes, une piste numérique et une piste « audio »).

La présentation de chaque leçon est attrayante, tout comme la transition entre les chapitres. Au total, une cinquantaine d'instructions sont « disséquées » et, surtout, leur effet est immédiatement visualisé à l'écran. C'est le principal avantage de cette formule que de présenter ainsi des applications. Un cours écrit, si bien fait soit-il, restera toujours un peu théorique.

Chaque volume développe un thème spécifique, c'est ainsi que l'on trouve dans l'ordre :

- la gestion d'écran, le maniement des couleurs et du graphisme et la génération des sons, toutes choses ayant un attrait particulier pour le profane ;
- les commandes d'édition, de lancement et de sauvegarde des programmes ;
- les expressions numériques, les opérateurs arithmétiques et les variables ;
- les notions de programmation proprement dites (branchements, compteurs et boucles) qui ne viennent qu'avec le quatrième volume, soit après l'acquisition d'un minimum de connaissances nécessaires à l'écriture d'un programme ;
- le volume 5 aborde les tableaux et les sous-programmes ;
- enfin, le volume 6 est consacré au traitement des chaînes de caractères (concaténation, extraction de sous-chaînes, etc.) ainsi qu'à la définition et à l'utilisation de caractères spéciaux créés par l'utilisateur.

Pendant le déroulement de chaque leçon, un menu, affiché au bas de l'écran, vous permet à tout moment de revenir en arrière ou de passer à la suite.

Une base solide

On ne trouvera pas dans ce cours, par ailleurs bien construit, les instructions étendues du Basic-disque, ni celles qui concernent par exemple le traitement des erreurs ou la gestion des manettes de jeu. Mais cette série de cassettes, très pédagogique, permet aux grands débutants d'acquérir une base solide.

Nous avons ici une bonne démonstration de l'EAO (enseignement assisté par ordinateur) dont le principal avantage est sans doute de s'adapter au rythme d'étude de chacun.

Jean-Paul CARRÉ

DES LUTINS ENVAHISSENT L'ÉCRAN

CHAQUE programme Basic tient du mouton à cinq pattes : non seulement il doit réaliser ce qu'on attend de lui, mais il doit également être le plus performant possible tout en occupant peu de place en mémoire... Ce dernier point est fondamental sur les ordinateurs de poche, qui ne sont pas toujours riches en Koctets. Sur tous les matériels, et sur le PB-700 en particulier, le fait de réaliser des programmes simples et concis présente de nombreux avantages : moins de temps passé au clavier, risques d'erreurs réduits, maintenance et modifications plus faciles, raisonnement plus pratique à suivre ou à expliquer...

Une fois lancé, le programme présenté ici propose un jeu : des « lutins » d'un type assez particulier envahissent l'écran par la droite. Il s'agit de les faire disparaître grâce à un petit rayon qui part de la gauche de l'écran et qui est dirigé à partir du clavier : P lance ce rayon, Q le déplace vers le haut et Z le déplace vers le bas. Si on le fait descendre alors qu'il est déjà en bas de l'écran, il réapparaît en haut. Et inversement. Chaque lutin touché donne un point.

Au départ, trois niveaux de difficulté sont proposés (de 1 à 3). La partie se termine dès qu'une rangée de lutins rejoint la colonne de gauche d'où part le rayon.

Ce programme pourrait n'être considéré que pour le résultat (passionnant) qu'il propose. Mais on peut aussi l'analyser afin de comprendre sa structure et surtout, de l'adapter à d'autres Basic que celui du PB-700.

Le programme se divise en cinq parties : initialisation (lignes 1 à 16), progression des lutins (lignes 18 à 25), déplacement du rayon (lignes 28 à 45), tir (lignes 48 à 70) et fin de partie (lignes 78 à 95).

Les tableaux de variables sont dimensionnés par la ligne 5 ; les lignes 10 à 15 servent à créer trois variables graphiques qui sont respectivement : une suite de symboles CHR$ (250) représentant les « lutins », une suite de tirets représentant le rayon qui va servir à faire disparaître les lutins, et une suite d'espaces qui permettra d'effacer ce rayon. Le programme taillera dans ces variables des longueurs correspondant à ses besoins avec une instruction MID$ (lignes 20, 55 et 60).

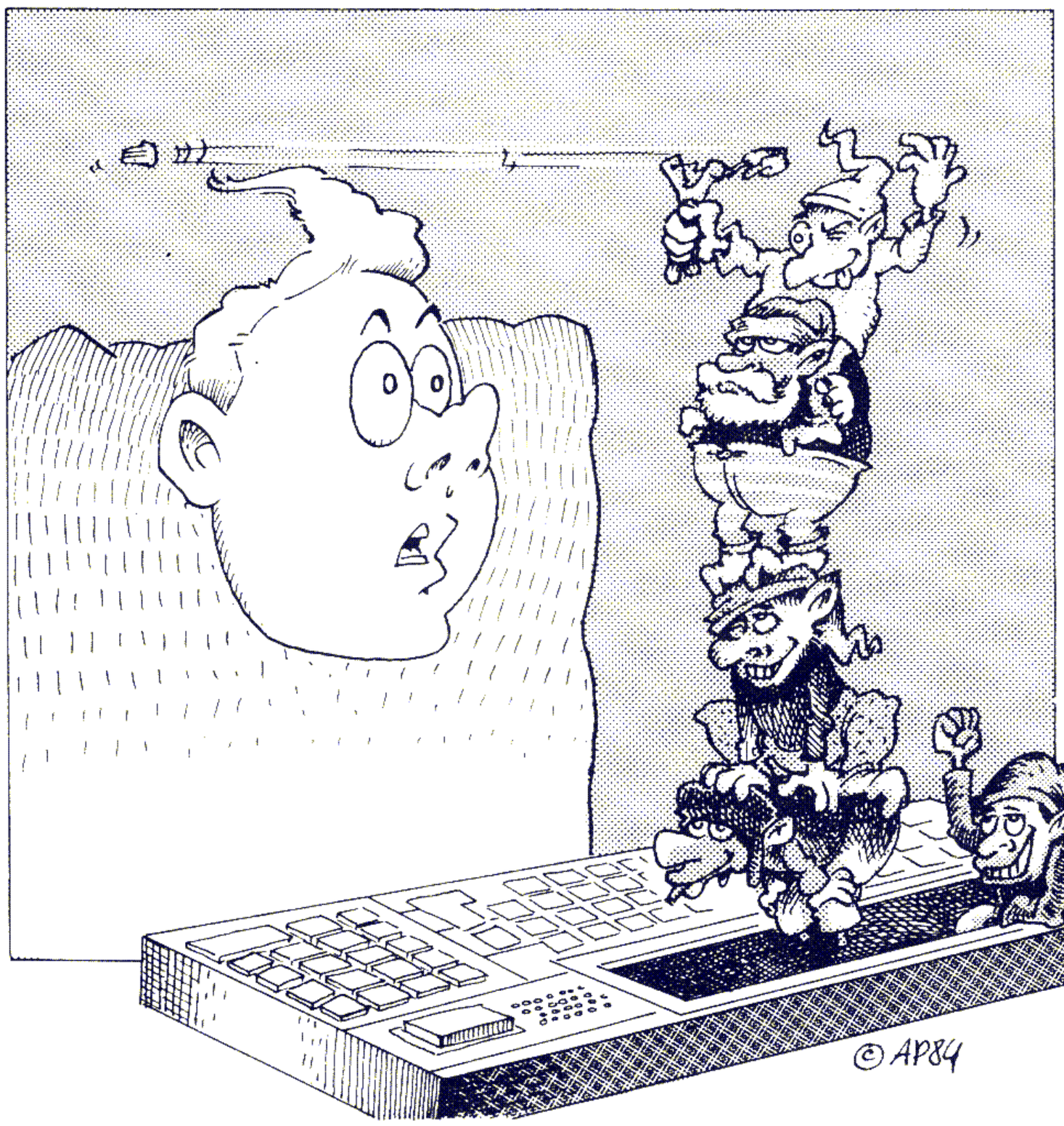
Ligne remplie, partie finie

Les lutins sont figurés par le symbole CHR$ (250) : une croix dans un carré ouvert (il manque un côté). Selon le niveau (T) choisi ligne 16, ils apparaîtront par paquets de 1, 2 ou 3.

N est une variable aléatoire qui peut prendre les valeurs 0, 1, 2 ou 3. On peut ainsi déterminer sur quelle rangée de l'écran vont apparaître les lutins, à chaque passage du pointeur ligne 20. Par exemple, en niveau 2, si N = 3, nous verrons arriver deux symboles CHR$ (250) sur la rangée du bas de l'écran.

La ligne 25 donne le signal de la fin d'une partie : il y a plus de 17 lutins sur une ligne.

Une petite accolade, CHR$(125), figure le départ du rayon ; elle se déplace verticalement sur la première




```
 1 REM *---INITIALISATION ---*
 5 CLEAR :DIM A(4),A$(1)*17,B$(1)*17,
   C$(1)*18:CLS
10 FOR I=1 TO 17
11 A$(1)=A$(1)+CHR$(250):NEXT I
13 B$(1)="----------------"
15 C$(1)="                "
16 INPUT "NIVEAU(1,2,3)";T:CLS
18 REM *---PROGRESSION DES LUTINS---*
20 N=INT(RND*4):A(N)=A(N)+T:LOCATE 19
   -A(N),N:PRINT MID$(A$(1),1,A(N));
25 IF A(N)>17 THEN 80
28 REM *---DEPLACEMENT DU RAYON ---*
30 LOCATE 0,X:PRINT " ";:IF INKEY$="Q
   " THEN X=X-1
31 IF INKEY$="Z" THEN X=X+1
35 IF X=-1 THEN X=3:IF X=4 THEN X=0
40 LOCATE 0,X:PRINT CHR$(125);
45 IF INKEY$<>"P" THEN 20
48 REM *--- TIR ----*
50 S=S+1:IF A(X)=0 THEN A(X)=1
55 A(X)=A(X)-1:LOCATE 1,X:PRINT MID$(
   B$(1),1,18-A(X));
60 LOCATE 1,X:PRINT MID$(C$(1),1,18-A
   (X));:BEEP
70 GOTO 20
78 REM *--- FIN DE PARTIE ---*
80 FOR I=1 TO 10
85 BEEP 1:BEEP :BEEP 1:NEXT I
90 CLS :LOCATE 5,2:PRINT "SCORE:";S;
95 IF INKEY$="" THEN 95 ELSE 5
```

Lutins en nombre
Programme pour **PB-700**
Auteur André Reeb-Gruber

Liste des variables

A(N)	longueur d'une rangée de lutins
A$(1)	suite de 17 symboles « lutins »
B$(1)	suite de 17 signes "–" représentant le rayon
C$(1)	suite de 18 espaces (effacement du rayon)
N	variable aléatoire de 0 à 3
T	niveau de difficulté (progression des lutins)
X	ordonnée du rayon

Les lutins arrivent à droite de l'écran ; le rayon qui part de la gauche essaie de contenir leur poussée

colonne de l'écran : ce qui revient à dire que ses coordonnées sont toujours sous la forme (0, X) avec X variant de 0 à 3. Quand la touche Z est activée (ligne 31), l'accolade se déplace vers le haut car X est incrémenté d'une unité ; inversement, une pression sur Q (ligne 30) la fera descendre d'un cran.

Un tour d'écran

La ligne 35 mérite d'être soulignée. Grâce à elle, en effet, l'accolade peut « faire le tour » de l'écran : si on la fait descendre alors qu'elle est en bas de l'écran, elle réapparaît en haut, et vice-versa. Cette ligne 35 est en fait un double test que l'on aurait pu écrire comme ceci : 35 IF X = −1 THEN X = 3 (passage du haut en bas de l'écran) et 36 IF X = 4 THEN X = 0 (passage du bas en haut de l'écran).

Quand X est ainsi déterminé, la ligne 40 affiche l'accolade à l'endroit voulu, sa position précédente ayant été effacée par la ligne 30.

S'il n'y a pas de tir à ce moment (ligne 45), les lutins continuent de se promener impunément (ligne 20 et suivantes).

Quand la touche P est activée, les lignes 50 à 60 interviennent pour déclencher le rayon. Ligne 50, on incrémente le score S en passant, et l'on s'assure que le rayon ne risque pas de déborder de l'écran : c'est ce qui se passerait si l'on tirait sur une rangée entièrement vide (A(X) = 0).

La ligne 55 forme le tir : le rayon est constitué d'une portion de la variable B$(1) (suite de signes "–") ; la longueur nécessaire est calculée grâce à A(X), de telle sorte que le rayon s'arrête sur le premier lutin de la rangée considérée.

La ligne 60 efface ce tir et le lutin touché : même principe d'action que la ligne 55, mais en remplaçant les signes "–" par des espaces. Puis l'on repart ligne 20 pour une nouvelle vague de lutins.

Lorsque les lutins ont réussi à atteindre l'accolade (signal donné par la ligne 25), ils claironnent leur victoire par des BEEP retentissants (lignes 80 et 85). Il ne reste plus qu'à afficher le score S (ligne 90) et, éventuellement, à repartir pour de nouvelles aventures (ligne 95) en appuyant sur une touche.

André REEB-GRUBER

PASCAL APPELLE LANGAGE-MACHINE

COMME on l'a vu dans LIST 4, l'interfaçage du Basic avec le langage-machine nécessite un certain « bricolage » sur Apple. De même, l'appel de sous-programmes en Assembleur à partir de programmes écrits en Pascal, doit suivre certaines règles.

La déclaration d'une procédure ou fonction en Assembleur se fait tout de suite après la déclaration des variables du programme Pascal (voir le programme BIDON).

Le système Pascal Apple permet alors de lier (*link*) les modules en langage d'assemblage (TOTO et ZOZO) avec des modules déjà compilés en « P-Codes ».

Supposons que nos deux programmes en Assembleur aient été assemblés et sauvés sur disquette. Le module TOTO est une fonction et le module ZOZO est une procédure. En langage d'assemblage, nous définirons TOTO et BIDON comme suit :

```
.PROC ZOZO,2        ;2 mots pour passer les 2 paramètres
...                 ;corps de la procédure
.END
```

De même la fonction TOTO sera définie par :

```
.FUNC TOTO,1        ; 1 mot pour le passage du paramètre
...                 ; corps de la fonction
.END
```

Le programme Pascal appelant devra de son côté définir la liste des paramètres nécessaires pour chacun des modules utilisés. TOTO et ZOZO seront alors définis comme des modules externes que nous lierons au programme Pascal après que ces modules aient été traduits en langage-machine par l'Assembleur Pascal (A)SSEM). Le programme BIDON définit TOTO et ZOZO comme modules externes. Le compilateur « passera » alors sur ces modules quand il les rencontrera ; le « linker » les liera au programme principal avant exécution :

```
L)INK
HOST FILE?BIDON
OPENING BIDON.CODE
LIB FILE?ZOZO
OPENING ZOZO.CODE
LIB FILE?TOTO
OPENING TOTO.CODE
LIB FILE <CR>
```

Le programme BIDON

```
PROGRAM BIDON
CONST . . . ;
. . . . . ;
VAR . . . . . ;
PROCEDURE ZOZO(VAR I, J:INTEGER);EXTERNAL; (* zozo en assembleur *)
FUNCTION TOTO(J:INTEGER):INTEGER;
BEGIN (* principal : bidon *)
  . . . . . ;
  ZOZO(A,B)
  .
  K: = TOTO(A);
  . . . . . ;
END.
```

```
MAP FILE? <CR>
READING BIDON
READING ZOZO
READING TOTO
OUTPUT FILE?BIDON.CODE
LINKING BIDON #1
    COPYING PROC ZOZO
    COPYING FUNC TOTO
```

Ici pas de déclaration de procédure ou fonction Assembleur « external » dans le programme Pascal de l'unité : il y a par contre obligation de passer par un SYSTEM.LIBRARY pour une unité sauf mention contraire dans le programme Pascal (voir l'unité ASZOZO).

Après assemblage de ZOZO et compilation de l'unité ASZOZO, on effectue une édition de liens ; l'unité est prête à être introduite dans la librairie.

Variables globales

Voici un programme Pascal contenant la procédure Assembleur ZOZO :

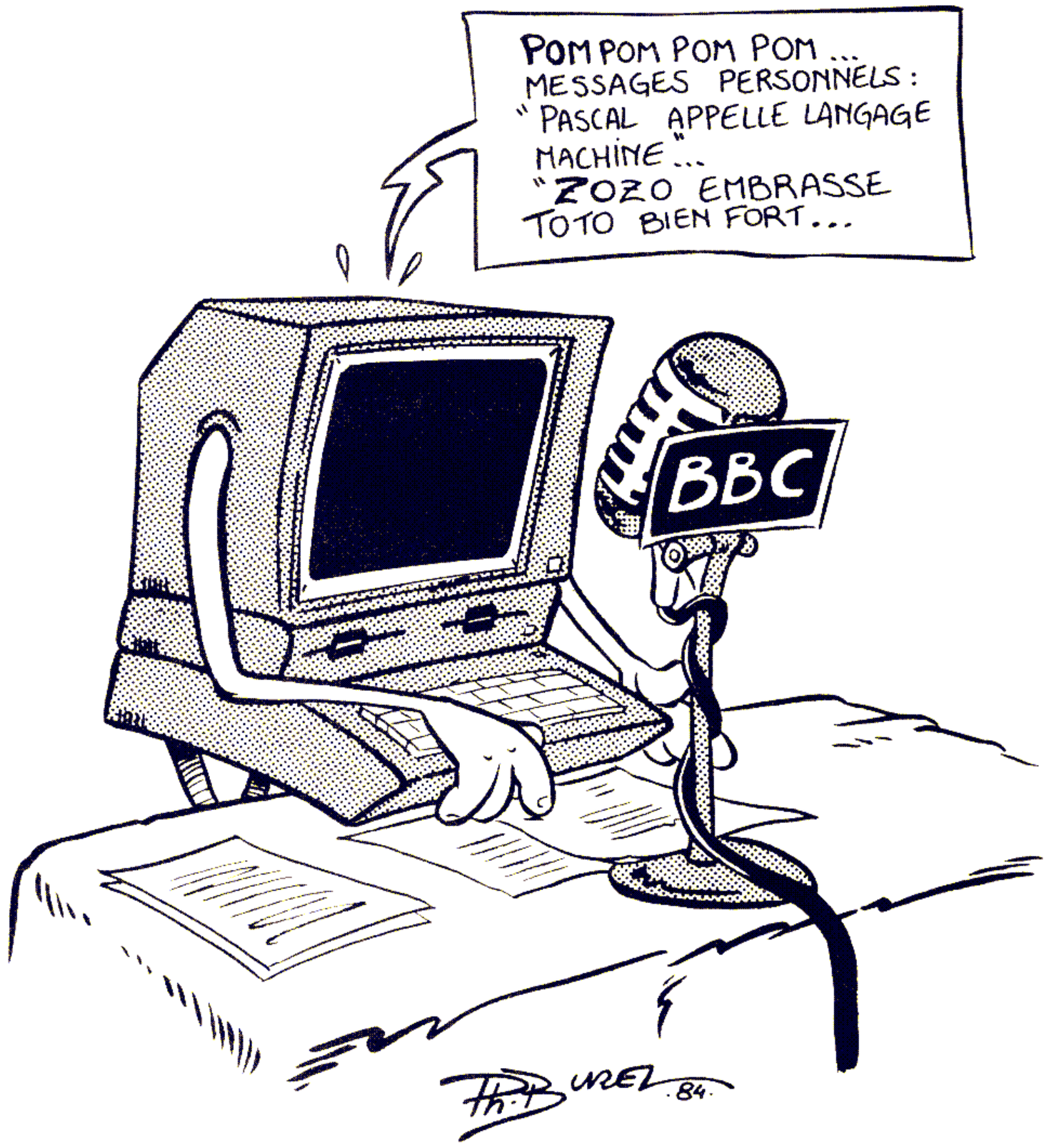
```
PROGRAM BIDON
USES ASZOZO;
CONST..;
VAR....;
BEGIN
    ..;
    ZOZO(...);
    ...;
END.
```

On compile le programme Pascal puis :

- si l'unité est régulière, on réalise l'édition de liens ;
- si l'unité est intrinsèque, il n'y a pas lieu de faire cette édition. L'unité est alors "pre linked", à condition qu'elle se trouve dans le SYSTEM.LIBRARY lors du lancement du programme.

Il faut remarquer que les variables déclarées au niveau de l'interface d'une unité sont globales par rapport au programme Pascal (ne pas les redéclarer au niveau global du programme !!). De plus, avec ou sans l'option "noload" lors de l'exécution du programme, il y a d'abord exécution des unités (mise en place des variables globales) puis exécution de l'initialisation.

Lors de l'appel d'une procédure en Assembleur, il y a toujours communication par empilage de mots sur la pile



L'unité ASZOZO

```
(*$SI + *)
UNIT ASZOZO; < INTRINSIC CODE 25 DATA 26; > (* < > - - >
unité intrinsèque *)
INTERFACE
  CONST . . . ;
  . . . . . ;   (* nécessite un segment data pour une unité intrinsèque *)
  VAR . . . . ;
  PROCEDURE UNTEL(. . . .);
  PROCEDURE ZOZO(. . .);
IMPLEMENTATION
  CONST . . . ;
  . . . . . ;
  VAR . . . . ;
  PROCEDURE ZOZO;EXTERNAL;
  PROCEDURE UNTEL;
    CONST . . . ;
    . . . . . ;
    VAR . . . . ;
      BEGIN
        . . . ;
        . . . ;
      END;
  BEGIN
  . . . . ;  (* initialisation s'exécute avant le programme Pascal*)
  . . . . ;
  END.
```

PASCAL APPELLE LANGAGE-MACHINE

(le mot minimum étant l'adresse retour qui doit être dépilée, sauvegardée puis empilée de nouveau lors de la sortie de la procédure Assembleur). Pour les mots, l'empilage a lieu dans l'ordre : octet de poids fort puis octet de poids faible. *Le mot* adresse *retour est toujours empilé en dernier.* Celui-ci n'est *jamais compté.*

Prenons un exemple pour plus de clarté :

PROCEDURE ZOZO(I:REEL;VAR J:REEL) ;

L'empilage a lieu dans l'ordre : mots pour le paramètre I puis mots pour le paramètre J. La procédure Assembleur ZOZO commencera par :

.PROC ZOZO,X

où X sera égal au nombre de mots décrivant les paramètres I et J.

Tous les passages par VARiables ont lieu par passage d'adresse, donc *un mot !* Tous les passages par valeurs ont lieu par le *nombre maximum de mots décrivant le paramètre.*

Déclarer les variables publiques

Quelques exceptions à la règle : STRING, RECORD, ARRAY sont passées par variable. Ainsi, ZOZO(VAR Z: STRING) ou ZOZO(Z:STRING) sont identiques : il y aura empilage d'un mot correspondant à l'adresse du paramètre Z.

Maintenant que l'on sait que dans ".PROC ZOZO,X", X est le nombre de mots et non le nombre de paramètres, on peut le calculer : deux mots décrivent la valeur de I et un mot contient l'adresse de J.

Les réels nécessitent deux mots, les entiers et les booléens, un seul, les entiers longs, ou ensembles, nécessitant le nombre maximum de mots alloués par la déclaration du programme Pascal (un mot indiquant ce nombre maximum). A part la fonction retournant une valeur qu'il est nécessaire d'empiler lors de la sortie Assembleur (ainsi que l'adresse retour), la communication entre Assembleur et Pascal passe par les directives du titre du paragraphe qui allouent un espace de communication de données.

Dans le programme Pascal, cela donne :

PROCEDURE ZOZO(VAR TOTO:< packed array of char >);EXTERNAL;

Et dans la procédure Assembleur :

```
.PROC ZOZO,1
.CONST A
.PUBLIC I, TOTO
.PRIVATE B,C:8
```

Toute variable publique doit être *déclarée en variable globale* dans le programme Pascal. De même toute constante doit être déclarée au niveau global (ici A). Si on utilise une unité, les déclarations seront faites au niveau de l'interface.

La référence privée indique une variable utilisable uniquement par le programme Assembleur. Mais la place de cette variable est définie dans le pro-

Programme 1

```
        .MACRO GLH
        ....
        ....
        .ENDM
;
        .MACRO MLF
        ....
        ....
        .ENDM
;
; * utilisable par toute routine *
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;
        .CONST AA,BB,..
        .PUBLIC ENTIER,TABLEAU,..
;  *  nécessite  une  déclaration  au  niveau  global  Pascal  .PRIVATE
AAA,BBB,...
;   *   accessible  uniquement  aux  routines  assembleur  localisation
programme hôte *
;
; * résolution lors de l'édition de liens *
;
;>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
;
SET     .EQU 00
;
; * peut servir à toute routine *
;
;-------------------------------------
;
        .PROC ZOZO,2
        .DEF  ALPHA,BETA,..
; * définition pouvant servir à d'autres routines *
        GLH SET
        ......
ALPHA   .....
        ......
        RTS
;
BETA    .WORD 0000
THETA   .BLOCK 012
;
;
;---------------------------------------;
        .PROC ZAZA
        .DEF MU,NU
        .REF ALPHA,BETA ; renvoie à une définition extérieure à ZAZA
;
        JSR  ALPHA
        MLF  BETA
        ....
        RTS
;
EPSILON .BYTE 00
;
;
        .....
        .....
        .END    ; fin de tout ce programme assembleur qui n'attend plus
que  son  hôte  qui  devra  déclarer  toutes les procédures assembleur
comme procédures externes.
```

gramme hôte (le segment des données globales reste ici durant tout le programme). Par défaut, un mot est réservé. Ici, on a réservé un mot pour B et un bloc de huit mots pour C.

Ceci peut être très intéressant lors de l'utilisation d'unités avec l'option "noload" ; on a ainsi un endroit pour sauvegarder des données. De même, entre deux appels de routines Assembleur, Pascal se sert de la page zéro, les données n'y sont donc pas à l'abri.

Le passage par la pile de TOTO permettra de mettre l'adresse de TOTO en page zéro (PTRTOTO & PTRTOTO + 1, seule façon de pouvoir utiliser l'adressage post ou pré-indexé).

Les actions qui modifient TOTO

La déclaration de ".PUBLIC I" permet de se servir de I comme étiquette adresse. On n'a donc pas besoin de l'adresse de I et il est inutile d'utiliser la pile.

La déclaration publique de variables permet de récupérer les actions de la routine Assembleur sur ces dernières (effet de bord).

".CONST A" permet d'accéder au niveau du programme Assembleur à la constante A déclarée au niveau global dans le programme hôte. On pourrait donc avoir dans la procédure Assembleur des mnémoniques de la forme :

```
LDA 12.
STA I ; write ln (I);  - - > 2572
LDA 0A
STA I+1

STA (PTRTOTO),Y ;adressage post-indexé
STA  TOTO, X    ;adressage indexé
```

TOTO sera modifié par ces actions.

Les routines séparées peuvent se partager données et sous-programmes en utilisant les directives .DEF et .REF.

L'Assembleur génère alors des informations permettant à un programme Assembleur d'appeler des sous-programmes qui se trouvent dans un autre programme Assembleur. L'utilisation de ".REF" et ".DEF" est similaire à celle de ".PUBLIC" associant des étiquettes ("labels") entre deux routines Assembleur plutôt qu'entre une fonction (ou une procédure Assembleur) et un programme hôte Pascal.

".DEF" identifie une étiquette pouvant être utilisée, au moyen de ".REF", à partir de ".PROC" ou ".FUNC" dans d'autres programmes d'assemblage.

Programme 2

Procédure ECRAN

```
                .PROC ECRAN,1
RETURN          .EQU 00
                PLA                 ;sauvetage de l'adresse de retour
                STA RETURN          ;
                PLA
                STA RETURN+1
                PLA                 ;on dépile la partie basse du paramètre
                AND #01;0- - >1
                EOR #01 ;1- - >0
                TAY
                LDA 0C054,Y         ;sélection de l'écran désiré
                PLA                 ;on dépile la partie haute du paramètre
                LDA RETURN+1        ;on restaure l'adresse de retour
                PHA
                LDA RETURN
                PHA
                RTS                 ;retour au Pascal
```

Programme Pascal

```
PROGRAM PAGE2;
VAR PAGE :INTEGER;
PROCEDURE ECRAN(PAGE:INTEGER);EXTERNAL;
PAGE:=1;
ECRAN(PAGE);
PAGE:=2;
ECRAN(PAGE);
```

```
.PROC ZOZO,3
.DEF SOL
      .....
      .....
;
;**
SOL   LDA
      ....
      ....
      RTS
;**
 LA     .WORD 0000
        .END
```

Ici, on définit une étiquette pouvant être utilisée par d'autres procédures en Assembleur. On renvoie alors à une définition de SOL externe à cette procédure.

```
.PROC ZAZA,1
.REF SOL, LA
.....
....
JSR SOL
...
ROR LA
...
.END
```

On peut maintenant réaliser une édition de liens entre ces routines et le "linker" résoudra les problèmes à condition que la routine ZAZA soit la routine hôte !

Attention : lors de l'édition de liens, ZAZA utilisera uniquement la routine de ZOZO comprise entre les étoiles.

Ensuite, il suffit de "linker" gaiement ZAZA avec un hôte Pascal se servant uniquement de ZAZA comme procédure externe (hôte : programme ou unité). On peut dire, dans ce cas, qu'une partie de ZOZO est une routine de ZAZA elle-même routine de l'hôte. ZAZA a piraté ZOZO !

Il est possible d'écrire un ensemble de routines qu'on assemble en une seule fois (voir programme 1).

Donnons pour finir un exemple réel : le Pascal fourni sur l'Apple dispose d'un ensemble de fonctions graphiques (*Turtlegraphics*) n'utilisant que la page graphique numéro un. On peut donc présenter un court programme qui permettra l'accès à la page numéro deux. Ce programme écrit sous "E)DIT", sera assemblé avec "A)SSEM" (voir programme 2).

Le Pascal, au moment de l'appel de la procédure ou de la fonction externe, empile sur la pile ("stack") la liste des paramètres exigés (deux mots pour les réels, un mot pour les entiers) puis l'adresse de retour au programme principal. La première chose à faire est donc de mettre à l'abri cette adresse afin de pouvoir la remettre sur la pile avant le RTS final.

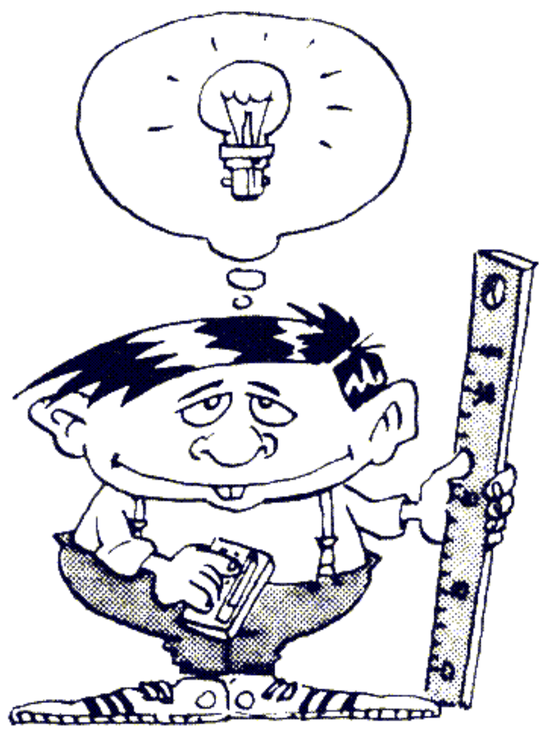
L'option "L)INK" reliera la procédure ECRAN au programme Pascal. Le passage des paramètres, ici le numéro de l'écran, se fait tout naturellement : il est mis sur la pile au moment de l'appel de la procédure ECRAN dépilé au moment opportun.

Philippe FRANÇOIS

MODULO-CI, MODULO-AH-LA-LA !

Jean-Christophe KRUST

SI jongler avec la pile opérationnelle de votre HP-41 C, traquer la milliseconde perdue et rogner le moindre octet est votre pain quotidien... Ou si, à l'inverse, vous échappe parfois un peu de la subtile recherche des programmes en Notation Polonaise Inverse...
Alors, voici qui doit vous intéresser.
En matière de programmation, est-on jamais certain d'avoir fait aussi bien que possible ?
Le mieux pourrait-il être l'ami du bien ?
Dans cette rubrique, les défis se succéderont : des programmes toujours plus courts, plus rapides...
Et les records tomberont !



Défi de LIST *n° 2, solution en n° 4 : Modulo - Suite et... fin ? Qui s'y frotte s'y pique... Entre les programmes MODULO 1 — le plus court — et MODULO 2 — le plus rapide pour les deux tests précisés — se situe une zone sombre de cas bien particuliers où MODULO 2 n'est plus, et de loin, optimal.*

Et c'est à Olivier Chararas que revient la tâche d'y jeter un peu de lumière. Suivons ses explications.

La méthode de la périodicité des congruences employée dans MODULO 2 (*LIST* n° 4) est très rapide dans de nombreux cas. Mais pas tous, car elle est très sensible au caractère premier de la puissance ou du diviseur.

Un contre-exemple choisi avec soin

Rappelons qu'il faut trouver a^b modulo c dans le cas où b notamment est un très grand nombre, par exemple : 8^{152} mod 3.

La périodicité maximale, égale à la fonction d'Euler de la puissance (ci-après), est rarement atteinte car, souvent, la puissance (b) et le diviseur (c) ne sont pas ensemble des nombres premiers.

La fonction d'Euler est définie par :

$$Q(n) = \prod_{i=1}^{i=m} (a_{i-1}) \times a_{i-1}^{(Pi-1)}$$

où les a_i successifs pour i = 1 à m sont les facteurs premiers décomposant n et les p_i, les puissances les affectant respectivement — voir la décomposition d'un nombre n en produit de nombres premiers :

$$n = \prod_{i=1}^{i=m} a_i^{Pi}$$

Le signe Π représente un produit au même titre que Σ représenterait une somme.

A l'appui de cette remarque, voici un contre-exemple (mesquinement choisi avec grand soin...) pour lequel MODULO 2 n'en finit pas... de ne pas trouver : 2^{223} modulo 211 (résultat 174) ou, pire encore, avec 2^{1117} modulo 1061 (résultat 302). Bien sûr, avec un seul module d'extension mémoire, c'est un

```
01♦LBL "MODUL03"    21 X<>Y
02 1                22 RDN
03 STO 00           23 X<>Y
04 X<> T            24 *
05 X<>Y             25 LASTX
06♦LBL 00           26 RDN
07 MOD              27 X<>Y
08 LASTX            28 MOD
09 X<> Z            29 STO 00
10 X=0?             30 LASTX
11 GTO 02           31 RDN
12 2                32♦LBL 01
13 /                33 X<> Z
14 ENTER↑           34 X↑2
15 INT              35 R↑
16 X<>Y             36 GTO 00
17 X=Y?             37♦LBL 02
18 GTO 01           38 R↑
19 RDN              39 END
20 RCL 00
```

Modulo 3
Programme pour **HP-41 C**
Auteur Olivier Chararas

peu court. Mais les quelque 300 registres disponibles d'une CV ne suffisent pas non plus dans tous les cas.

Une curiosité

Alors, voici MODULO 3 — programme encore optimisable à n'en point douter — qui donne, lui, le bon résultat dans la majorité des cas en moins de 6 secondes et n'use que d'un registre de mémoire.

Puisque la périodicité des congruences est liée à la fonction d'Euler, elle-même liée aux nombres premiers... Vous voyez la suite !

Calculez avec MODULO 3 des 2^x modulo x pour x premier ou non. Si x est un nombre premier, le résultat est 2. Et cela constitue un assez rapide test de primalité.

La réciproque n'est pas vérifiée : 2^x modulo x = 2 n'entraîne pas que x soit premier (par ex. : 2^{341} mod 341 = 2 mais 341 n'est pas premier). En revanche, 3^{341} mod 341 = 168 ce qui prouve que 341 n'est pas premier.

Ainsi, dans le doute, si n^x mod x = n quand n est un nombre premier on peut poursuivre les tests (n = 2, 3, 5, 7, 9, 11, etc.) jusqu'à la preuve que x n'est pas premier (n^x mod x ≠ n) ou bien qu'il l'est...

Olivier CHARARAS

QUI DIT MIEUX ?

Triangle de Pascal. Bateau ? Oui, mais pouvez-vous trouver les coefficients de n'importe quelle ligne (aux OUT OF RANGE près) du triangle de Pascal ?

Zoom arrière. Ci-dessous, se trouve un tableau de chiffres. Pour le construire, on a posé, à partir du sommet 1, que chaque chiffre d'une nouvelle ligne est égal à la somme de celui qui est situé juste au-dessus et de celui qui est à gauche de ce dernier. L'absence de chiffre vaut 0. Toutes sortes de symétries apparaissent, par lignes ou colonnes, en fonction du numéro de la ligne, etc.

N° de ligne	Tableau de Pascal						
0	1						
1	1	1					
2	1	2	1				
3	1	3	3	1			
4	1	4	6	4	1		
5	1	5	10	10	5	1	
6	1	6	15	20	15	6	1
	...						

③ + ③ ↓ ⑥

Sachant le numéro n de la ligne, trouvez la suite des chiffres qui la compose. Par exemple, n = 5, les chiffres sont 1, 5, 10, 10, 5 et 1.

Par commodité, et dans le souci de mesurer le temps de calcul, on stockera (ô sacrilège) dans les registres R00 à Rnn les nn + 1 chiffres de la ligne à trouver. Ce n'est que lorsque ces chiffres sont ainsi stockés qu'on stoppe le chronomètre.

On pourra programmer, à côté, une petite routine (hors concours) de visualisation des chiffres trouvés.

La solution réside, bien sûr, dans l'examen attentif de la logique du tableau. Mais aussi, dans le rappel de vos souvenirs sur le développement de $(a+b)^n$ (coefficients) ou loi binômiale.

MONTI

MOD sans MOD
(HP-41, HP-11, 12, 13, 14, 15, 16...)

Après lecture du numéro de septembre de « Misez p'tit » — ô rage — j'ai su la terrible vérité : ma HP-15 C ne connaît pas la fonction MOD.

Alors... si on faisait un programme qui calcule modulo : le reste de la division entière d'un nombre par un autre... sans employer du tout la fonction MOD. La HP-41 et les autres HP sur un pied total d'égalité ?

En deux mémoires et 19 octets, ce n'est pas l'idéal, mais ça tourne. Voici MOD sans MOD :

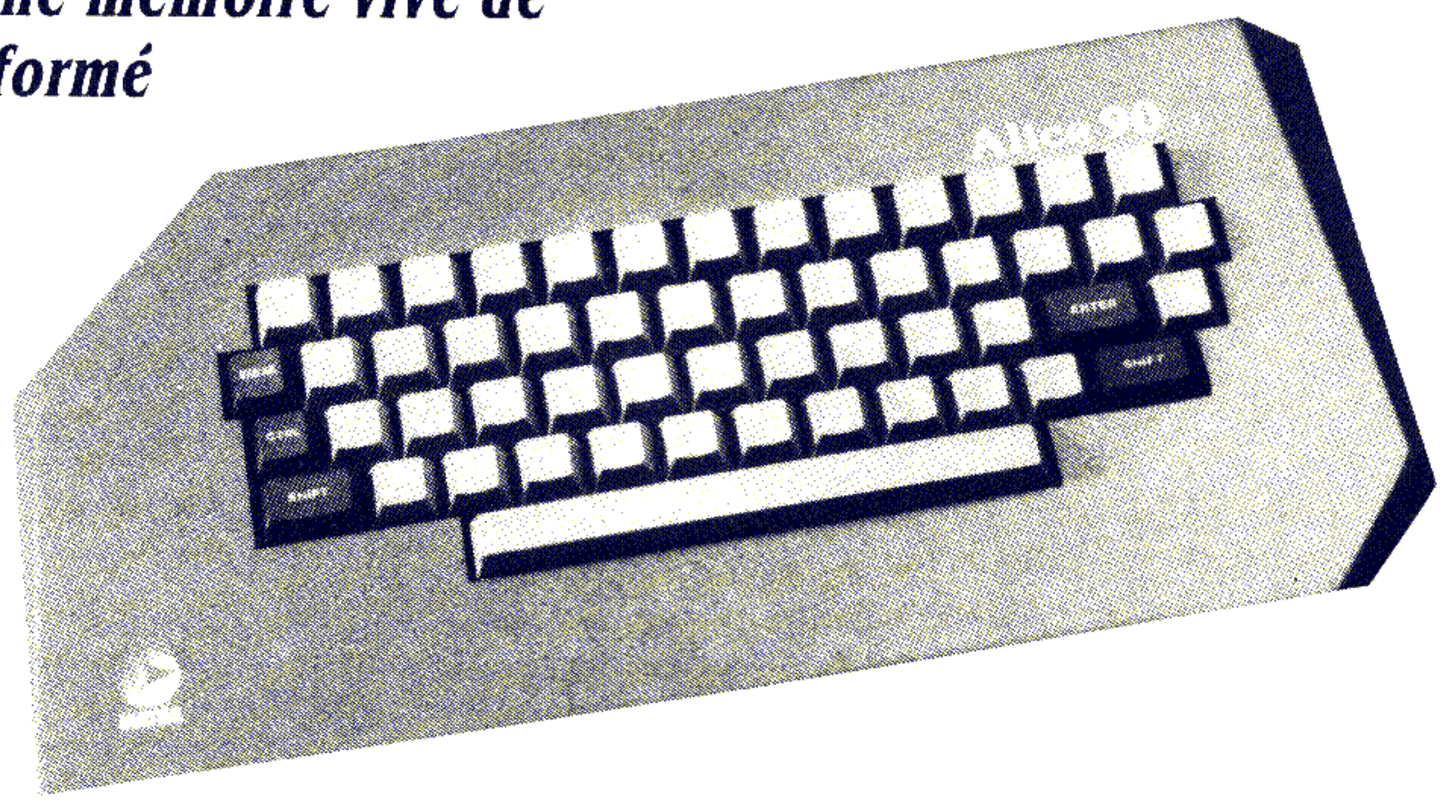
STO 8 R↓ STO 9 R↑ ÷ INT RCL 8 × RCL 9 — CHS RTN

Taper a, puis b (de a modulo b), lancer le programme et lire le résultat. Les ancêtres HP-33 E et autres reliques remplaceront R↑ par R↓ R↓ R↓.

Bruno FITER

LE BASIC D'ALICE 90

UN nouveau boîtier et une mémoire vive de 32 Koctets ont transformé Alice en Alice 90. Le Basic, lui, a peu changé. Il est très classique. Mais, sur ce nouveau modèle, il est aussi possible de programmer en Assembleur.



Il y a un an déjà, la collaboration de l'Américain Tandy et du Français Matra donnait le jour à un être bicéphale, baptisé MC-10 par le premier et Alice par le second. Aujourd'hui, sous la houlette de Hachette, Matra reprend le flambeau, en conservant pratiquement le même Basic, comptabilité oblige !

Sur ce tronc commun, Matra a développé deux nouveaux appareils. Dans son petit coffret exigu, le premier des deux présente une mémoire-utilisateur de 8 Koctets ; un connecteur permet d'ailleurs d'ajouter une extension de 16 Koctets, quoique pour un prix élevé. Une version plus luxueuse, équipée d'un véritable clavier mécanique, d'un boîtier de taille conventionnelle, et surtout de 32 Koctets, constitue le fleuron de la gamme, sous le nom d'Alice 90.

Les pionniers de l'informatique individuelle se souviennent sans doute du TRS-80. Ils reconnaîtront, dans la version de Basic qui équipe l'Alice, beaucoup des ordres qui faisaient leurs délices voilà plusieurs années.

Aligné cependant sur les performances de la concurrence, l'affichage a lieu selon trois modes différents : 32 colonnes sur 16 lignes, 40 sur 25 et, luxe suprême, 80 sur 25. Un simple ordre CLS suivi des quantités 32, 40 ou 80 se charge de l'opération. Toutefois, les possesseurs de téléviseurs couleur auront quelques difficultés à lire l'écran dans le dernier mode.

Un point fort : le clavier est mécanique

Sur le modèle 90, 53 touches (et ce sont de vraies touches) se proposent à la frappe. En plus des touches de caractères alphabétiques et numériques, quatre flèches du plus bel effet rappellent que le modèle originel était dépourvu d'éditeur et qu'il était impossible de déplacer simplement le curseur. Mais si les flèches sont bien là, leur pression n'est pas toujours suivie d'effets ; seule la flèche à gauche permet ici d'effacer le dernier caractère entré. Les autres servent en mode-éditeur.

Avec 32 ou 40 colonnes, le SHIFT n'affiche pas de minuscules, mais des caractères semi-graphiques sur 16 touches. Pour se faire une idée de leur aspect, il suffit de découper en quatre la surface occupée à l'écran par un caractère alphabétique normal, et d'allumer un ou plusieurs des quatre rectangles ainsi obtenus. Le fond de cette matrice 2 × 2 reste noir, mais la couleur des rectangles dépend de celle du curseur clignotant. Il est possible de la modifier en pressant une ou plusieurs fois CTRL O, pour obtenir l'une des huit variantes : vert, jaune, bleu, rouge, blanc, bleu clair, violet, orange. En outre, CTRL associée à 38 des touches, permet la frappe directe d'un mot-clé du

Basic, à l'image de nombreux appareils récents.

SHITF 0 effectue le passage à une impression en caractères inversés (vert sur fond noir) ; mais en mode 80 colonnes, par contre, elle assure une frappe en minuscules (non accentuées, sur un matériel français !). La répétition de cette séquence SHIFT 0 déclenche le retour à la normale.

Les lignes de programme Basic sont limitées à un maximum de 127 caractères. Seules les majuscules sont autorisées pour rentrer les instructions et fonctions diverses. L'ordre LIST permet de lister une ou plusieurs lignes, tandis que l'astérisque suivi d'un numéro permet d'accéder à un éditeur de ligne. Les flèches déplacent le curseur vers la droite ou la gauche, tandis que toute autre touche provoque l'insertion des caractères correspondants, à l'emplacement du curseur. Seule, la flèche vers le bas efface alors un caractère.

Un bon vieux Basic

Le temps de l'édition d'une ligne, SHIFT suivi d'un *espace* permet d'avoir un clavier à répétition. La pression de CTRL *espace* permet le retour à la normale.

Encore une fois, la société Microsoft a fait du bon travail. Le Basic est très standard avec à peine 55 mots-clé différents. Les impressions sont réduites au minimum. Les PRINT USING, HTAB ou VTAB, n'existent pas. Seule, la syntaxe PRINT @ suivie d'un nombre permet une impression en n'importe quel point de l'écran ; cette formule acrobatique oblige à calculer la position d'impression comme résultat du numéro de ligne souhaité multiplié par la largeur d'écran et auquel on rajoute la colonne d'impression. Ce qui s'écrirait VTAB 8:HTAB 5 dans un autre Basic, devient PRINT @ 8*40+5, pour un écran en 40 colonnes.

Les principales commandes et fonctions du Basic sont présentes. Les entrées sont obtenues de façon classique à partir d'un ordre INPUT et la saisie du clavier peut s'opérer à travers une variable INKEY$. Les tests sont limités à un simple IF...THEN sans ELSE, tandis que les opérateurs logiques comprennent les AND, OR et NOT. Appliqués à des valeurs numériques, ils servent aux opérations binaires ; ainsi 5 AND 3 prend la valeur 1 (en effet 0101 AND 0011 donne 0001).

Les calculs numériques s'effectuent sur 9 chiffres significatifs entre 10^{38} et 10^{-38}. Les opérateurs usuels sont +, −, *, /. Quant aux fonctions trigonométriques (sinus, cosinus et tangente), elles travaillent sur des valeurs exprimées en radians, à chacun de faire la conversion à partir des degrés.

Le traitement des chaînes de caractères s'effectue au moyen des fonctions usuelles : MID$, RIGHT$ et LEFT$. Les DATA peuvent être numériques ou alphanumériques, mais sont analysées en bloc. Il n'est pas possible de pointer une ligne de DATA particulière.

Les erreurs de syntaxe ne se manifestent qu'à l'exécution du programme, sous formes d'abréviations sibyllines en langue anglo-saxonne. Heureusement, le manuel, très clair, apporte sa traduction.

Le dessin est assurément le parent pauvre de l'Alice. Il est impossible de changer simplement la couleur des textes imprimés à l'écran. Dans les modes 32 et 40 colonnes, ceux-ci restent obstinément verts et noirs, même en changeant la couleur de fond par un ordre CLS suivi d'un chiffre de 0 à 8.

Si les graphiques peuvent coexister avec le texte sur un même écran, ils ne peuvent en aucun cas s'y superposer. Aucune fonction directe ne permet de tracer des cercles et des lignes : tout se fait point par point. Il faut découper l'écran en petits rectangles élémentaires : 64 × 32, 80 × 50 ou 160 × 125 points selon le mode écran choisi. En allumant ou en éteignant chaque rectangle, on crée un dessin.

En mode 32 ou 40 colonnes, l'ordre SET (X, Y, C) permet d'allumer un point de coordonnées X et Y dans la couleur C choisie parmi huit possibilités (noir exclu). L'instruction RESET effectue le contraire : l'extinction d'un point. Mais, en fait, tous les points contenus dans un même pavé de 2 × 2 doivent obligatoirement être de la même couleur. Si l'on désire travailler en huit couleurs, la résolution disponible est inférieure aux valeurs annoncées ci-dessus, sans toutefois être radicalement divisée par deux.

En mode 80 colonnes, on retrouve cet inconvénient, mais avec une palette de couleurs limitée à trois valeurs : la couleur du fond, celle d'impression du texte et celle du bord de l'écran.

Fiche technique d'Alice 90

Constructeur : *Matra*
Prix public : *2 490 FF ; version coffret avec lecteur de cassette et cinq cassettes de logiciel, 3 490 FF*
Mémoire vive : *56 Koctets dont 32 Koctets utilisateur*
Mémoire morte : *16 Koctets*
Langages : *Basic, Assembleur du 6803*
Modes d'affichage : *32 colonnes sur 16 lignes ; 40 sur 25 ; 80 sur 25*
Nombre de mots du Basic : *55*

La véritable nouveauté de l'Alice réside dans la présence en mémoire morte, d'un programme assembleur du microprocesseur 6803. Il faut auparavant réserver un espace mémoire à l'aide d'une instruction CLEAR. Il devient alors possible d'écrire un programme en langage-machine sous forme de mnémoniques. Quoique simplifié, cet assembleur est un réel outil de travail.

L'Assembleur n'est pas un jouet

Pour l'écriture du programme vous disposez d'un éditeur pleine page avec beaucoup de facilités : insertion de ligne, recherche de chaînes de caractères... Il est véritablement dommage de ne pas pouvoir en profiter en Basic.

Les labels sont autorisés dans l'écriture du programme qui peut être sauvegardé sur cassette sous forme d'un fichier texte. A la fin, l'assembleur se charge de traduire le programme en codes-machine exécutables à n'importe quelle adresse, et indique les éventuelles erreurs de syntaxe.

Au vu des possibilités somme toute restreintes du Basic, il est intéressant de se demander quel public vise Matra. S'agit-il du néophyte ? A prix égal, il trouvera d'autres machines équipées de synthétiseur de sons, de manettes de jeu, etc. Pour lutter contre la concurrence, seul le modèle de base a des atouts pour s'imposer : son manuel et son prix.

En revanche, le programmeur confirmé trouvera dans l'assembleur, un outil de développement en langage-machine. La capacité mémoire importante lui sera nécessaire.

Ironie du sort, ou revirement commercial, l'Alice qui semblait un appareil de première initiation se révèle dans sa version 90 un outil de travail très intéressant.

Pierre RICARD

PROGRAMMER AVEC VISICALC

Il arrive fréquemment que des scientifiques aient, comme les gestionnaires, à manipuler simultanément de nombreuses listes de nombres, surtout en physique (par exemple, des différences de potentiel en les sommets d'un maillage). Pour ce genre de calculs, il existe des outils merveilleux : les « calques » ou « tableurs ». On s'intéressera ici à Visicalc.

On peut lire dans un livre consacré à Visicalc (1), comment résoudre avec ce tableur une équation telle que : $\log((10+x)/(9-x)) = 0$ dont la solution est évidente ($x = -1/2$). On trouve un encadrement à 0,5 % près, en dix lignes. Les exemples seraient faciles à multiplier de cette programmation en langage « tableur ».

Une fois utilisées les merveilleuses fonctions d'un calque, il reste à exploiter ses résultats. Quand il faut recopier à la main une grande masse de données, ce n'est guère agréable. Quand il faut recopier ces nombres sur le clavier de l'ordinateur pour les traiter de nouveau, cela devient insupportable.

Ramasser les données

Il existe heureusement une solution au nom barbare : le format « DIF » (Data Interchange Format) mis au point par Software Arts pour Visicalc. Sur d'autres tableurs, cela porte d'autres noms, comme « SYLK » pour Multiplan de Microsoft, etc. Ce logiciel particulier est conçu pour ramasser tous les contenus d'un rectangle Visicalc en un fichier DIF, fichier que l'on peut utiliser aussitôt, par exemple, pour reverser ces contenus dans les mémoires de l'ordinateur hôte à fin de traitement en Basic, Pascal, Fortran ou Assembleur.

Donnons l'exemple (certes très stupide) d'un tableau de 4 lignes et 3 colonnes qu'il s'agit de récupérer par DIF :

ANDRE	11236	4700
BERTRAND	22756	2500
CATHERINE	1264	2000
DONALD	10681	300

Les instructions sont très simples. Il suffit de se placer en A1, coordonnées du coin en haut à gauche du tableau, d'entrer successivement les commandes / (slash) S # S et de donner un nom de fichier, comme TOTO/DIF, le suffixe DIF étant obligatoire. Le logiciel demande ensuite les coordonnées du coin en bas à droite (ici C4, quatrième ligne de la troisième colonne). Il reste enfin à préciser si l'on veut opérer par lignes (R) ou colonnes (C) : répondre R pour « row ». C'est fait : le fichier TOTO/DIF s'est inscrit sur la disquette ; on peut quitter Visicalc pour toujours.

Donnons une idée du contenu de ce fichier, listé ci-dessous (*liste du fichier TOTO/DIF*). Il comporte 44 lignes, regroupées ici par cinq pour gagner de la place.

On n'y comprend pas grand-chose à première vue mais, en fait, c'est très simple. Un peu d'attention montre que les trois colonnes de notre tableau initial s'y retrouvent bien, d'abord celle des prénoms, mis entre guillemets (ce sont des chaînes de caractères), puis les deux formées de nombres. Pas de confusion : la virgule ici n'est qu'un sépa-

Liste du fichier TOTO/DIF

TABLE	0,1	" "	VECTORS	0,1
" "	TUPLES	0,1	" "	DATA
0,0	" "	−1,0	BOT	1,0
"ANDRE"	1,0	"BERTRAND"	1,0	"CATHERINE"
1,0	"DONALD"	−1,0	BOT	0,11236
V	0,22756	V	0,1264	V
0,10681	V	−1,0	BOT	0,4700
V	0,2500	V	0,2000	V
0,300	V	−1,0	EOD	

(1) « Clefs pour Visicalc » *de Jean-Louis Marx et Alain Thibault, éditions du PSI, collection Mémento.*

rateur ; il ne s'agit pas de notre virgule bien française remplacée, le cas échéant, par le point décimal anglo-saxon.

Un examen de l'enregistrement sur la disquette (par exemple avec un utilitaire) montre facilement que notre fichier DIF est tout bêtement un fichier séquentiel, de maniement donc très commode ; il ne demande pas de précaution particulière.

Poursuivons l'observation : chaque chaîne est précédée des symboles [1,0] à la ligne supérieure, tandis qu'un nombre suit toujours immédiatement les symboles [0,] placés sur la même ligne que lui, avec un V sur la ligne inférieure.

C'est clair, le drapeau 1 signifie : la donnée est une chaîne. Il est alors suivi du nombre 0 (0 = rien ?) parce qu'il en faut bien un, et des caractères entre guillemets. Au contraire, le drapeau 0 dit qu'il s'agit d'une donnée numérique, qui lui succède aussitôt, et la chaîne de la ligne suivante est un banal V, sans guillemets, V comme Valeur mais aussi Vide... La structure est donc assez simple : toujours deux nombres suivis d'une chaîne de caractères.

La morphologie d'un fichier DIF

A chaque bout d'enregistrement de colonne, figurent deux lignes assez bizarres. La première est toujours [−1,0]. Bon sang, mais c'est bien sûr ! Toujours ce sacré drapeau : −1 = séparateur entre rangées, pour le distinguer d'un indicateur de nature de donnée (0 ou 1). La seconde ligne est [BOT], sauf tout à la fin où c'est [EOD]. Celle-là, on la reconnaît : c'est « end of data », « fin de données ». Quant à BOT, ce sont les initiales (anglaises) de « beginning of tuple », « commencement de tuple », ce qui ne dit pas grand-chose a priori. Patience.

Remontons tout en tête du fichier, et regardons les douze premières lignes. Pour simplifier, TABLE, VECTORS, TUPLES et DATA seront considérées ici comme des constantes des fichiers DIF, ainsi que les couples de guillemets [" "] qui les suivent. En fait, les seules choses susceptibles de varier pour nous, ce seront les nombres 4 et 3 qui terminent les lignes en-dessous de VECTORS et TUPLES. Quatre, c'est clairement le nombre de VECTORS (vecteurs), ici lignes puisque nous avons choisi R (comme *row*, ligne), à l'enregistrement ; et trois, celui des colonnes, appelées ici TUPLES (on dirait en argot de matheux t-uplets, quelque chose comme « liste de t machins », un terme quasi-synonyme de « vecteur », inutilisable car déjà employé pour les lignes).

En choisissant C (colonne) au lieu de R, Visicalc aurait simplement échangé les nombres 4 et 3, et le premier BOT aurait été suivi des six lignes que voici : 1,0 "ANDRE" 0,11236 V 0,4700 V au lieu des huits lignes listées plus haut. On aurait alors eu naturellement un [−1,0 BOT] de plus (4 TUPLES au lieu de 3). Le système est donc assez astucieux : les mots ROW et COLUMN — ligne et rangée — sont transformés, au gré du programmeur, en VECTORS et TUPLES, ce qui permet de ramasser les données dans un ordre ou dans l'autre suivant les cas. Ici VECTOR = ROW, TUPLE = COLUMN ; on sait qu'on aurait pu faire l'autre choix.

Nous connaissons maintenant la morphologie d'un fichier DIF. Rien n'empêche d'en fabriquer d'artificiels, pour mettre dans un Visicalc ce que nous voulons : figure en annexe un programme Basic Microsoft (sur TRS) ouvrant un fichier séquentiel appelé ESSAI/DIF sur le lecteur n° 1, simulant un fichier DIF obtenu d'après le tableau Visicalc imaginaire suivant :

DUPONT	123
MARTIN	456
SIMON	789

On peut trouver ce programme plutôt long — c'est vrai. En fait le problème n'est heureusement pas celui-là, mais plutôt de pouvoir mettre dans les mémoires de l'ordinateur le contenu d'une page de tableur. Et cela est bien plus facile, comme le montre le programme Basic suivant, destiné au cas où le Visicalc a construit uniquement une suite de nombres réels (35 dans l'exemple choisi, issu de 5 colonnes de 7 données) que l'on veut mettre dans A(I) pour I compris entre 0 et 34 :

```
10 DIM A(34) : D$ = "TOTO/DIF" :
   I = −1
20 OPEN "I", 1, D$
30 INPUT # 1, T$, T, N, S$
40 IF T$ < > "DATA" THEN 30
50 INPUT # 1, T, N, S$
60 IF T = 0 THEN I = I + 1: A(I) = N
70 IF S$ < > "EOD" THEN 50
80 CLOSE: END
```

Ce programme fonctionnerait aussi bien si les nombres que l'on recherche étaient moins nombreux que 35, et/ou s'ils figuraient dans un tableau bien plus vaste, mélangés à des chaînes de caractères. Il est bien entendu immédiat de l'adapter à toutes les demandes possibles, les fichiers DIF étant de simples fichiers séquentiels traités comme tous les autres fichiers. Bonne chance avec votre nouvel allié, le tableur !

André WARUSFEL

Programme d'ouverture d'un fichier ESSAI/DIF

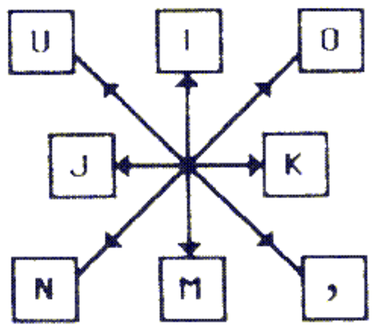
```
10 Z = 0:U = 1:M = −1
20 C$ = CHR$(34):A$ = C$ + C$:V$ = ","
30 B$ = "BOT":E$ = "EOD":DA$ = "DATA"
40 TA$ = "TABLE":VE$ = "VECTORS"
50 TU$ = "TUPLES":W$ = "V"
60 T = 3:D = 2
70 D1$ = "DUPONT":D1$ = C$ + D1$ + C$
80 D2$ = "MARTIN":D2$ = C$ + D2$ + C$
90 D3$ = "SIMON":D3$ = C$ + D3$ + C$
100 A = 123:B = 456:C = 789
110 OPEN"O", 1, "ESSAI/DIF:1"
120 PRINT # 1,TA$
130 PRINT # 1,Z; V$;U
140 PRINT # 1,A$
150 PRINT # 1,VE$
160 PRINT # 1,Z;V$;T
170 PRINT # 1,A$
180 PRINT # 1,TU$
190 PRINT # 1,Z;V$;D
200 PRINT # 1,A$
210 PRINT # 1,DA$
220 PRINT # 1,Z;V$;Z
230 PRINT # 1,A$
240 PRINT # 1,M;V$;Z
250 PRINT # 1,B$
260 PRINT # 1,U;V$;Z
270 PRINT # 1,D1$
280 PRINT # 1,U;V$;Z
290 PRINT # 1,D2$
300 PRINT # 1,U;V$;Z
310 PRINT # 1,D3$
320 PRINT # 1,M;V$;Z
330 PRINT # 1,B$
340 PRINT # 1,Z;V$;A
350 PRINT # 1,W$
360 PRINT # 1,Z;V$;B
370 PRINT # 1,W$
380 PRINT # 1,Z;V$;C
390 PRINT # 1,W$
400 PRINT # 1,M;V$;Z
410 PRINT # 1,E$
420 CLOSE
430 END
```

Programme Basic Microsoft construisant un « faux » fichier DIF de D = 2 colonnes et T = 3 lignes ; la première colonne contient des caractères, la seconde des nombres. On peut les modifier en changeant les lignes 70 à 100. Si l'on veut changer les nombres 2 et 3, il faut évidemment reprendre en conséquence les lignes 60, 160 et 190 mais, surtout, bouleverser les instructions entre 260 et 390.

MOI, JE DESSINE DES HORREURS

AUJOURD'HUI, je vous propose un moyen commode de dessiner sur l'écran d'un Oric ou d'un Apple. Dessins statiques pour le moment, mais nous verrons une prochaine fois comment animer vos créations.

J'ai suivi ici le principe de la programmation fonctionnelle : chaque commande est interactive (elle est suivie d'un effet immédiat). Nous construisons donc le programme par modules fonctionnels, ce qui permet, quand on le désire, d'ajouter facilement de nouvelles commandes.



Les huit touches du clavier commandant les déplacements

Sur le clavier, nous avons retenu un pavé de huit touches correspondant aux déplacements du curseur dans les huit directions habituelles (nord, nord-est, sud-est, sud, etc.). Pour cette partie des commandes, on se reportera au schéma ci-dessus. Le programme propose encore 14 autres commandes auxquelles on accède en pressant une ou deux touches. Ce sont respectivement :

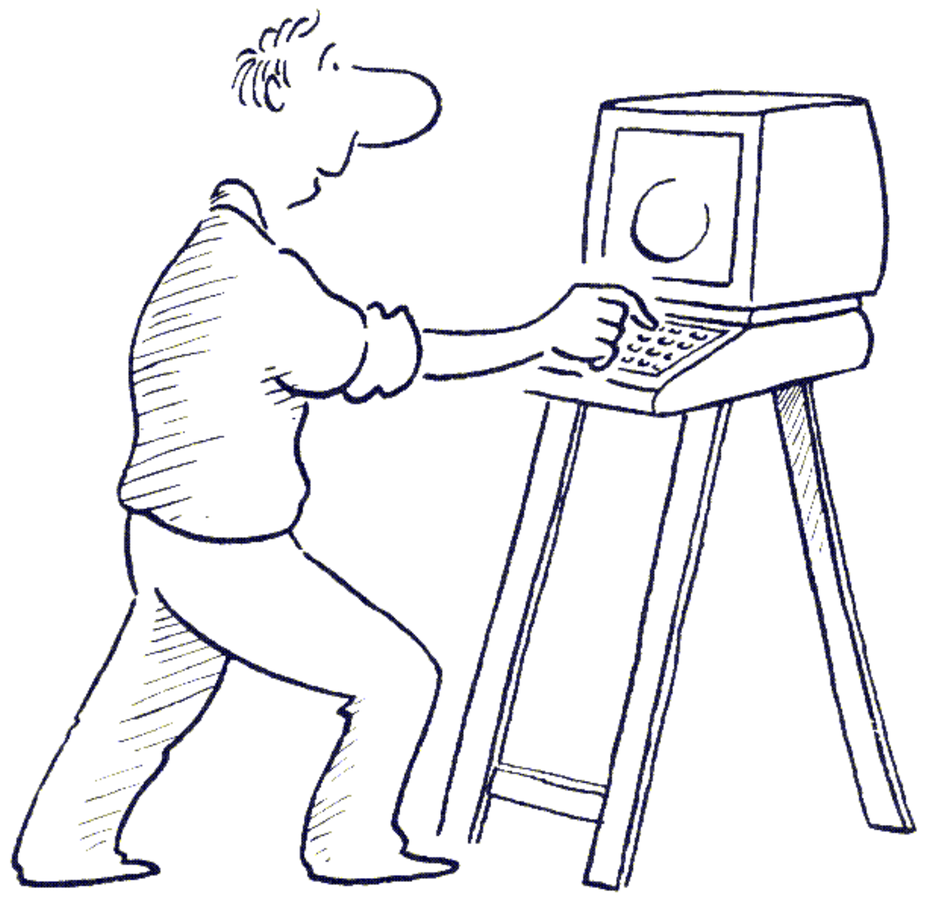
- X : coordonnées ;
- D : droite ;
- R : rectangle ;
- C : cercle ;
- E : effacement ;
- ? : position du curseur ;
- T : insertion d'un texte dans le dessin (pas de surprise : les majuscules sont obtenues par SHIFT, et DEL efface un caractère) ;
- ^ P, ^ E et ^ Y (ou Ctrl P, Ctrl E...) permettent de changer la couleur du papier, de l'encre, ou de les inverser ;
- ! : position haute ou basse du crayon virtuel (TRAce ou non) ;
- Z : choix du point zéro ;
- ^ Z (ou Ctrl Z) : retour du « crayon » au point zéro ;
- Q : terminer et quitter l'écran.

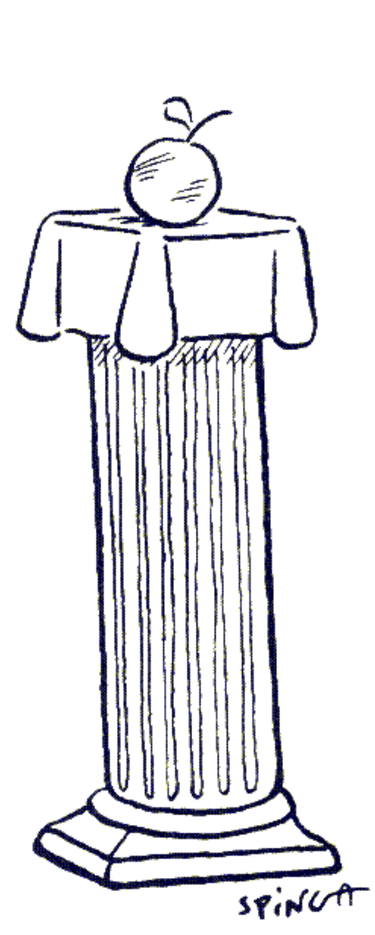
Le programme consiste donc en un ensemble de commandes (voir la ligne 10, REPEAT : GOSUB 20 : UNTIL NOT COMMAN) dont chacune est soit un déplacement, soit le tracé d'une droite, d'un rectangle, etc.

```
25 IF DEPLAC THEN GOSUB 300
30 IF XY THEN GOSUB 400
35 IF DROITE THEN GOSUB 500
40 IF RECTAN THEN GOSUB 600
(...)
```

Des commandes à répétition

Comme le programme, bien entendu, toutes les commandes qui le composent ont un début (identification de la commande) et une fin grâce à laquelle on passe éventuellement à une autre commande. Certaines d'entre elles, et notamment celles qui intéressent les



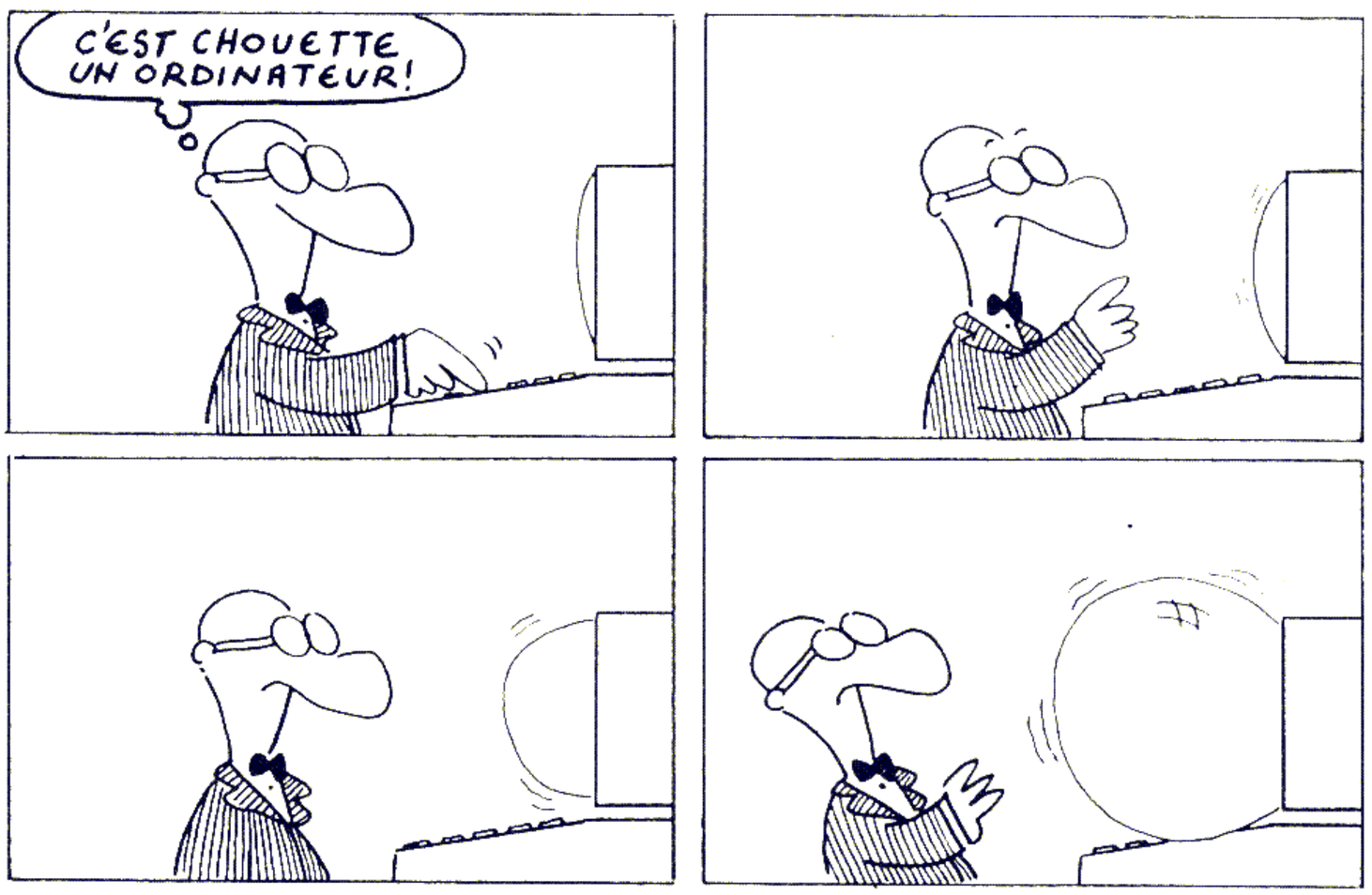


MOI, JE DESSINE DES HORREURS

Écran graphique
Programme pour Oric
Auteur Max Hagenburger


```
 0 REM ** ecran graphique **
 1 REM --------------------
 5 GOSUB 100 'repete les commandes
 10 REPEAT :GOSUB 20 :UNTIL NOT COMMAN
 15 GOSUB 2900 'fin -------
 20 GOSUB 200  'commande :
 25 IF DEPL THEN GOSUB 300
 30 IF XY   THEN GOSUB 400
 35 IF DROI THEN GOSUB 500
 40 IF RECT THEN GOSUB 600
 45 IF CERC THEN GOSUB 700
 50 IF EFFA THEN GOSUB 800
 55 IF OU   THEN GOSUB 900
 60 IF TXTE THEN GOSUB 1000
 65 IF CYPE THEN GOSUB 1100
 99 GOSUB 2800 :RETURN :REM ======
 100 REM initialisations limite/basic
 110 LB=#4F00 :DOKE 162,LB:DOKE 166,LB
 120 PAPER 0 :INK 7 :HIRES :CLS :PRINT
 130 X0=13 :X9=238 :Y0=1 :Y9=198
 140 CADRE=3100 :T=1 :GOSUB CADRE
 150 PRINT"ECRAN pour dessins animes
      "CHR$(96)" Max";
 160 H$=CHR$(8) :INPT=3000
 170 BLAN$=H$+CHR$(128)+CHR$(151)
 180 TRA=1 :VIT%=1 :PAP=0 :ENC=7
 180 X=120 :Y=100 :XZE=X :YZE=Y
 190 GET A$
 195 RETURN :REM -----
 200 REM debut une commande
 202 :A=ASC(A$)
 205 IF A$="!" THEN TRA=1-TRA
 210 IF A$="Z" THEN XZE=X :YZE=Y
 215 IF A=26 THEN X=XZE :Y=YZE 'Z
 220 DEPL=(A$="," OR A$=>"I" AND A$<="
O" AND A$<>"L" OR A$="U")
 225 DROI=(A$="D")
 230 RECT=(A$="R") :XY=(A$="X")
 235 CERC=(A$="C") :OU=(A$="?")
 240 EFFA=(A$="E")
 245 TXTE=(A$="T")
 250 CYPE=(A=5 OR A=16 OR A=25)
 290 :CURSET X,Y,3
 295 RETURN :REM -----
 300 PRINT "UIOJKNM, ";
 310 REPEAT :PRINT A$;
 320 IF A$="I" THEN Y=Y-1
 325 IF A$="J" THEN X=X-1
 330 IF A$="K" THEN X=X+1
 335 IF A$="M" THEN Y=Y+1
 340 IF A$="U" THEN X=X-1:Y=Y-1
 345 IF A$="O" THEN X=X+1:Y=Y-1
 350 IF A$="N" THEN X=X-1:Y=Y+1
 355 IF A$="," THEN X=X+1:Y=Y+1
 360 IF X<X0 THEN X=X0 :MUSIC 1,2,9,12
 365 IF X>X9 THEN X=X9 :MUSIC 1,2,9,12
 370 IF Y<Y0 THEN Y=Y0 :MUSIC 1,2,9,12
 375 IF Y>Y9 THEN Y=Y9 :MUSIC 1,2,9,12
 380 CURSET X,Y,TRA
 385 POKE 526,2 :MUSIC 1,0,1,0
 390 A$=KEY$ :UNTIL A$=""
 395 RETURN :REM -----
 400 REM Xy
 410 :GOSUB INPT
 420 X=XX :Y=YY
 495 RETURN :REM -----
 500 PRINT "Droite:";
 510 :GOSUB INPT
 520 IF X=XX AND Y=YY THEN RETURN
 530 DRAW XX-X,YY-Y,TRA
 540 X=XX :Y=YY
 595 RETURN :REM -----
 600 PRINT "Rectangle:";
 610 :GOSUB INPT
 620 IF X=XX OR Y=YY THEN RETURN
 630 DRAW 0,YY-Y,TRA :DRAW XX-X,0,TRA
:DRAW 0,Y-YY,TRA :DRAW X-XX,0,TRA
 640 PRINT"Plein O"H$;:GET O$
 650 PRINT O$;:IF O$<>"O" THEN RETURN
 660 FOR LI=Y TO YY STEP SGN(YY-Y)
 670 CURSET X,LI,TRA :DRAW XX-X,0,TRA
 680 NEXT
 695 RETURN :REM -----
 700 INPUT "Cercle rayon ";RA$
 710 RA=VAL(RA$)
 720 IF RA=0 OR X-RA<X0 OR X+RA>X9 OR
Y-RA<Y0 OR Y+RA>Y9 THEN PING :RETURN
 730 CIRCLE RA,TRA
 740 PRINT"Plein O"H$;:GET O$
 750 PRINT O$;:IF O$<>"O" THEN RETURN
 760 FOR LI=-RA TO RA
 770 CO=SQR(RA*RA-LI*LI)-RA/60
 780 CURSET X-CO,Y+LI,TRA :DRAW 2*CO,0
,TRA
 790 NEXT
 795 RETURN :REM -----
 800 PRINT "Efface l'ecran O"H$;
 810 GET O$ :IF O$<>"O" THEN RETURN
 820 TRA=1 :VIT%=1 :PAP=0 :ENC=7
 830 HIRES :PAPER PAP :INK ENC
 840 X0=13 :X9=238 :Y0=1 :Y9=198
 850 T=1 :GOSUB CADRE
 860 X=XZE :Y=YZE
 895 RETURN :REM -----
 900 PRINT "ou ?";
 910 CURSET X,Y,2:WAIT 9 :CURSET X,Y,2
:WAIT 9
 920 IF KEY$<>"" THEN 910
 995 RETURN :REM -----
 1000 PRINT "Texte (shi=maju) ";
 1010 IF X+6>X9 OR Y+8>Y9 THEN PING:RE
TURN
 1020 GET A$ :IF A$<" " THEN RETURN
 1030 A=ASC(A$) :SHI=(PEEK(521)>162)
 1040 IF NOT SHI AND A$=>"A" AND A$<="
Z" THEN A=A+32
 1050 IF A=127 AND X+6>X0 THEN X=X-6:C
URSET X,Y,3:CHAR A,0,1-TRA :GOTO 1020
 1060 CHAR A,0,TRA :PRINT CHR$(A);
 1070 X=X+6:IF X+5>X9 THEN X=X0:Y=Y+8
 1080 CURSET X,Y,3
 1090 GOTO 1010 'repete
 1095 RETURN :REM -----
 1100 PRINT "'Encre 'Papier 'Ynverse";
 1110 IF A=25 THEN R=PAP:PAP=ENC:ENC=R
 1120 EN=EN-(A= 5):IF ENC>7 THEN ENC=0
 1130 PA=PA-(A=16):IF PAP>7 THEN PAP=0
 1140 PAPER PAP :INK ENC
 1195 RETURN :REM -----

 2800 REM fin une commande
 2810 CLS
 2820 PRINT"UIOJKNM, Xy Dro Rec Cer Ef
Ze ? Qui !!";
 2830 IF TRA THEN PRINT H$H$H$BLAN$
 2840 PRINT"Txt 'EPY
 2850 PRINT"X"X-12"Y"Y;BLAN$"?"H$;
 2860 A=KEY$
 2870 GET A$ :IF A$="" THEN 2870
 2880 IF A$="Q" THEN INPUT"Quitte l'EC
RAN 'O'";O$ :IF O$<>"O" THEN A$="z"
 2890 COMMAN=(A$<>"Q")
 2895 RETURN :REM -----
 2900 CLS:PRINT"     fin du programme"
 2990 GET A$ :TEXT
 2995 END :REM =======
 3000 INPUT " X,Y ";XX$,YY$
 3010 XX=VAL(XX$) :YY=VAL(YY$)
 3020 IF LEFT$(XX$,1)="+" OR LEFT$(XX$
,1)="-" THEN XX=XX+X
 3030 IF LEFT$(YY$,1)="+" OR LEFT$(YY$
,1)="-" THEN YY=YY+Y
 3040 IF XX$=>"0" THEN XX=XX+12
 3050 IF XX>X0 OR XX>X9 OR YY<Y0 OR YY
>Y9 THEN PING :POP
 3095 RETURN :REM -----
 3100 REM cadre
 3110 CURSET X0-1,Y0-1,T
 3120 DRAW X9-X0+2,0,T:DRAW0,Y9-Y0+2,T
:DRAW X0-X9-2,0,T:DRAW 0,Y0-Y9-2,T
 3195 RETURN :REM =========
```



D'un Basic à l'autre

Pour adapter ce programme à un autre ordinateur, il faut considérer la limite de la mémoire Basic, les commandes spécifiques de la couleur, la définition de l'écran, l'inversion de l'affichage et les instructions graphiques. Ainsi, dans le cas de l'Oric, CURSET X, Y, 1 : DRAW XX-X, YY-Y, 1 place un point de coordonnées X et Y et trace un trait jusqu'à XX, YY ; le troisième paramètre, ici 1, détermine la brillance du tracé.

Quelques autres instructions, comme REPEAT... UNTIL ou MUSIC, ne se retrouvent pas sur toutes les versions du Basic. A titre d'exemple, on trouvera ci-dessous, adaptées à l'Apple II, la plupart des lignes qui font problème.

```
10  GOSUB 20: IF COMMAN THEN 10
110 LB = 8192: POKE 115,0: POKE 116,32          (limite Basic)
120  HGR : HOME : VTAB 22
130 X0 = 1:X9 = 278:Y0 = 1:Y9 = 158:TRA = 3
205  IF A$ = "!" THEN TRA = 3 - TRA
290  HPLOT X,Y
380  HCOLOR= TRA: HPLOT X,Y
390  GET A$: IF A$ ) " " THEN 315                (remplace REPEAT...UNTIL)
530  HCOLOR= TRA: HPLOT  TO XX,YY
630  HCOLOR= TRA: HPLOT  TO XX,Y TO XX,YY TO X,YY TO X,Y
670  HPLOT X,LI TO XX,LI
780  HPLOT X - C0,Y + LI TO X + C0,Y + LI
910  HCOLOR= 3: HPLOT X,Y: HCOLOR= 0: HPLOT X,Y
920 C =  PEEK ( - 16384): IF C = 63 OR C = 191 THEN 910
2810  HOME : VTAB 21
2830  IF TRAC THEN INVERSE : PRINT H$H$"!!";: NORMAL
3120  HPLOT X0 - 1,Y0 - 1 TO X9 + 1,Y0 - 1 TO X9 + 1,Y9 + 1
        TO X0 - 1,Y9 + 1 TO X0 - 1,Y0 - 1
```

En l'absence de la fonction CIRCLE, on devra programmer le calcul des points formant le cercle, ce qui nous donne, toujours pour Apple, à la ligne 730 :

```
HPLOT X + RA, Y
FOR I = 0 TO 6.3 STEP 3/RA
XI = X + RA * COS (I)
YI = Y + RA * SIN (I)
HPLOT TO XI, YI : NEXT
```

Moins commode : si l'on veut insérer du texte dans le dessin, il faudra décrire chaque caractère en DATA. Quant à l'inversion vidéo de l'écran, on l'obtiendra en utilisant par exemple le programme en langage-machine de *LIST* n° 2, page 80, dont voici une version décimale :

```
        1100  PRINT "'Ynverse";
        1110 A = 768:D$ =
"169032133007160000132006177006073255145006200208247230007165007201064208239096"
        1120  FOR I = 1 TO  LEN (D$) STEP 3
        1130 V =  VAL ( MID$ (D$,I,3)): POKE A,V:A = A + 1
        1140  NEXT
        1150  CALL 768
        1195  RETURN : REM -----
```

déplacements dans les huit directions, sont à répétition ; on maintient simplement la touche enfoncée.

Selon les cas, la commande est développée comme un ensemble d'instructions appelé par GOSUB et se terminant par RETURN, ou bien, si elle est très courte, exécutée directement (exemples : TRAce ou Z aux lignes 205 et 210). Remarquons d'ailleurs que TRA, variable de la trace, est une bascule. Vraie ou fausse, elle vaut 1 ou 0, d'où la formule de la ligne 205 : TRA = 1 - TRA (si TRA valait 1, elle vaut zéro et vice versa). Quant à la commande Z, elle permet de mettre en mémoire la position d'un point du dessin, position que l'on retrouve avec Ctrl Z.

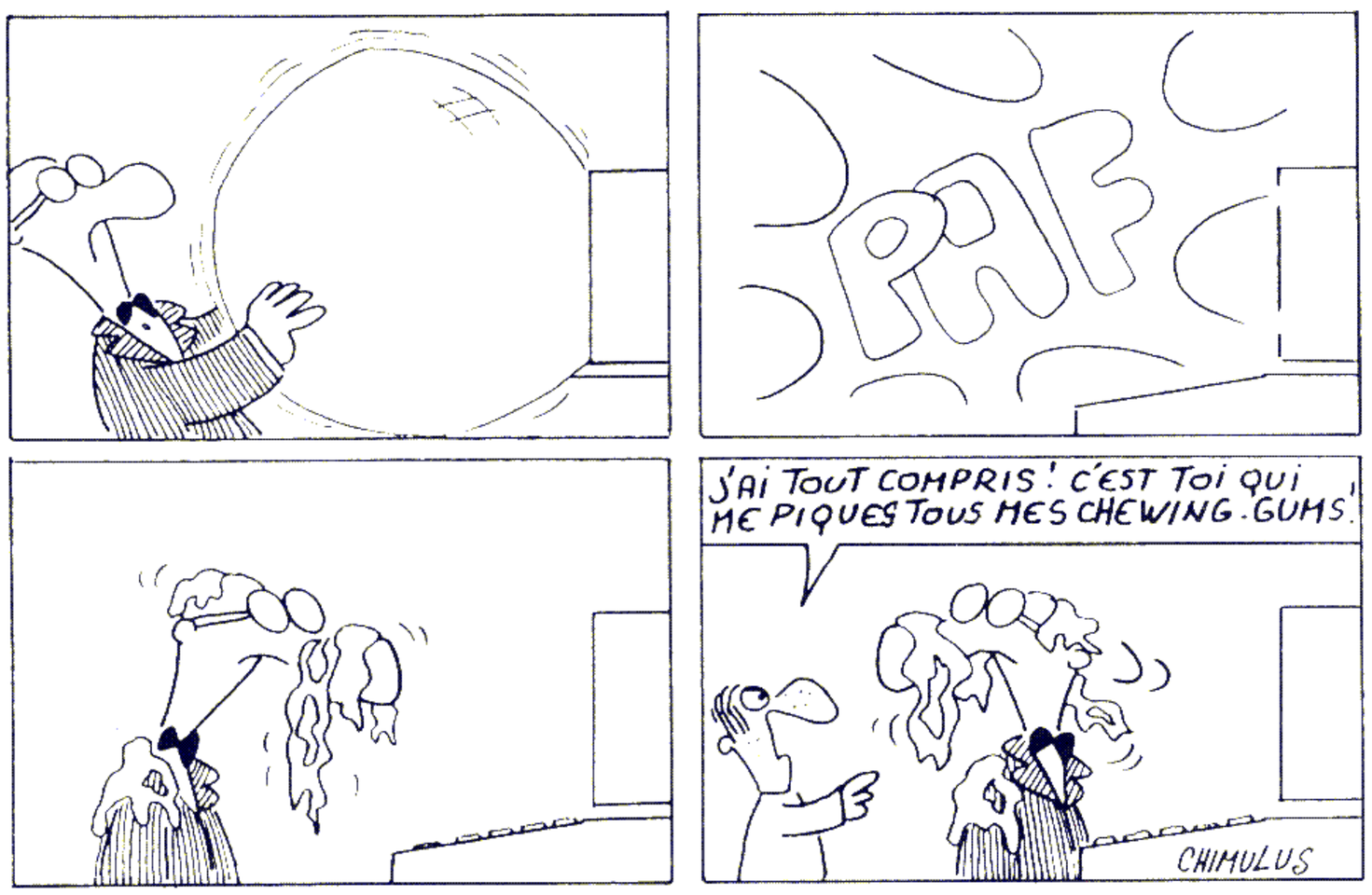
Une option pour rectangles et cercles

Certaines commandes requièrent l'entrée au clavier des coordonnées d'un ou de plusieurs points (voir le sous-programme INPT, lignes 3000 à 3095). Deux possibilités : les coordonnées sont absolues et commencent par un chiffre, ou bien elles sont relatives au point qui vient d'être tracé et elles commencent alors par les signes + ou −. En valeur absolue, il faut décaler de 12 points de façon à ne pas dessiner dans les zones de la couleur de l'Oric. A tout moment, le programme vérifie que le dessin demandé ne va pas... sortir de l'écran.

Une option pour les rectangles et pour les cercles : obtenir non pas seulement un contour, mais une figure pleine. A propos de l'instruction CIRCLE, on remarquera, ligne 770, la petite rectification nécessaire pour remplir le cercle : -RA/60.

Signalons encore deux petits trucs que vous ne connaissez peut-être pas et qui pourront vous servir dans d'autres programmes. L'expression A$ = KEY$ sert ici à nettoyer la mémoire-clavier avant la saisie par GET A$. Enfin, la valeur de l'octet 526 règle la répétition d'une touche et son clic.

Vous disposez maintenant de tout ce qu'il faut pour dessiner sur votre écran. Exercez-vous, et soignez spécialement les petites créatures, lutins, farfadets ou schtroumpfs de tout acabit : dans un prochain numéro de *LIST*, nous verrons comment gérer les dessins en mémoire, comment les mettre en mouvement, comment animer vos petits monstres...



Max HAGENBURGER

FORTH ET L'ARITHMÉTIQUE

FORTH est un langage ouvert : il accepte toute fonction, tout mot nouveau, même spécialisé. L'important n'est pas que ce vocabulaire soit standard, mais que son noyau le soit. Car on peut toujours ramener un mot nouveau, directement ou indirectement, à ce noyau. Il est donc légitime de prétendre que Forth est intégralement transportable. Alors, penchons-nous sur ce noyau, cassons-le, et... la transparence deviendra totale, « je vous le jure par Jupiter Ace » !

L'élément mémoire de base est la cellule de 16 bits qui est constituée de deux octets alignés. Calculer en "simple longueur" consistera à calculer avec des opérandes dont chacun sera mémorisé sur 16 bits. Et par voie de conséquence, la "double longueur" fera manier des chiffres occupant 32 bits. Il sera possible aussi de se livrer aux joies de la "triple longueur" ou de la "demi-longueur" qui se basent sur des nombres de 48 bits dans le premier cas et d'un octet dans le second. Il suffira d'utiliser des opérateurs adaptés, pris dans le glossaire standard, le "noyau", ou nouvellement forgés.

Limitons-nous, pour le moment, à la "simple longueur". La base binaire de tout langage informatique fait que les 16 bits d'une cellule peuvent être tous à zéro, valeur la plus basse de la cellule, ou tous à un, valeur la plus élevée. Toutes les valeurs intermédiaires, certains bits à zéro et d'autres à un, sont possibles, comme de juste.

Vérifions-le ! Pour cela, il faut forger un opérateur adapté :

```
: BIN 2 BASE ! ;
: B? BIN . DECIMAL ;
```

Chemin faisant, deux nouveaux opérateurs sont rencontrés. Ils doivent être définis.

BASE (--) : variable détenant la base de calcul courante ; son évocation empile son adresse.

! (n adr --) : prend l'adresse qui est en sommet de pile et y introduit les seize bits de n, sous-sommet de pile.

Dans le mot BIN, 2 est en sous-sommet de pile puisque BASE empile son adresse par-dessus et tout est prêt pour que le signe "!" introduise 2 dans la variable.

Ensuite dans B?, BIN fait passer en binaire, le point (.) affiche et DECIMAL ramène au décimal. On en déduit logiquement que la définition de l'opérateur DECIMAL est :

```
: DECIMAL 10 BASE ! ;
```

Changeons de base

Encore une remarque, cette procédure de chargement d'une variable est une procédure très importante, on la rencontrera constamment. Pour bien s'y habituer, chargeons :

8 BASE ! et l'on est maintenant en "octal"

16 BASE ! qui nous fait passer en "hexadécimal"

32 BASE ! puisque le "duotriacontal" ne nous fait pas peur !

(En "duotriacontal", le système des habitants de la troisième planète de l'étoile répertoriée au catalogue NGC sous le numéro 5678, qui ont huit doigts à chacune de leurs quatre mains, on compte 0 1 2 3 4 5 6 7 8 9 A B C D E F G H I J K L M N O P Q R S T U V 10 11 12, etc.).

Revenons à notre décimal avec DECIMAL. Convertissons allègrement les nombres décimaux en binaire :

0 B? et la réponse s'affiche : 0 OK.

44 B? donne 101100 OK.

255 B? donne 11111111 OK. Tiens,

Rappels de Forth

LE glossaire Forth comprend douze grands groupes de fonctions. Chacun d'eux a pour but de régler l'action synchronisée des opérateurs qui résident en mémoire, morte ou vive, et des opérandes (données sous forme de nombres ou de codes ASCII) qui ont été empilés dans la pile de données.

Ces données occupent maintenant des cellules (16 bits) ou des octets (8 bits, pour les amnésiques !). L'intérêt de pouvoir introduire une grande partie des opérateurs au clavier réside dans le fait que cela permettra de tester les programmes par morceaux, au fur et à mesure de leur entrée. Seuls les opérateurs de structuration sont réfractaires à l'utilisation en mode interprétation : pas question de passer DO...LOOP au clavier si ce n'est en mode compilation, c'est-à-dire entre les signes deux-points (:) et point-virgule (;).

Fonctions	Opérateurs
entrées, sorties et formatage	., ." , CLS, CR, VLIST
manipulation de piles	DROP, DUP, OVER, PICK, ROT, ROLL, SWAP, >R, R>, R
manipulation mémoire	?
tests numériques et caténiques	=, >, <, 0=, 0>, 0<
définition/compilation et sécurité	:, ;, (
structuration	DO, LOOP, IF, ELSE, THEN, I, J
arithmétique	+, −, *
dictionnaire et vocabulaires	FORGET

tiens ! Huit bits à 1, le maximum possible dans un octet sera donc 255.
9999 B? donne 10011100001111 OK et nous remarquons là qu'il y a 14 bits.

Mézalors, et si nous mettions les 16 bits de la cellule à 1, quelle valeur décimale cela nous donnerait-il ? Plutôt que de tâtonner avec B?, nous allons forger un autre outil.

Une réponse inattendue

Partant du principe que nous sommes en binaire, il suffit de repasser en décimal le temps de la conversion et de revenir en binaire aussitôt après. Cela s'écrit :

```
: D? DECIMAL . BIN ;
```

Nous voulons convertir du binaire en décimal, commençons donc par passer en binaire : BIN OK.

Et maintenant nous pouvons chercher :
1111 D? et la réponse apparaît : 31 OK.
11111111 D? donne bien 255 OK.
100000000 D? donne 256 OK. Voilà, il faut bien 9 bits pour passer de 255 à 256. Ils se répartissent sur les deux octets de la cellule. Le premier octet étant à zéro (00000000) et le deuxième n'ayant que le bit de droite à 1 (00000001). Ce qui donne une cellule de la forme : 0000000100000000. La confirmation nous est donnée en entrant ce dernier nombre binaire sur seize chiffres et en le faisant suivre de D?. La réponse est bien : 256 OK.

C'est très clair, non ??? Alors, essayons maintenant : 1111111111111 111 D? et la réponse ne se fait pas attendre, −1 OK !!!

Meeeuuuhnnnooonnn ! La machine n'est pas cassée ! Nous faisons simplement la connaissance des nombres signés. L'inventeur de Forth, Charles Moore, a pensé que puisque la plupart de nos calculs comportaient des valeurs faibles, le système numérique courant pouvait être conçu comme un cycle partant de 0 montant jusqu'à 32767, puis passant à −32768 et revenant vers 0 par les nombres négatifs.

Mais comme il peut nous être utile de nous passer des nombres négatifs, il a prévu la possibilité de travailler sur les valeurs absolues, les nombres "non-signés". De ce fait, nous pourrons aller

FORTH ET L'ARITHMÉTIQUE

Un programme Forth, pas à pas

```
: XROLL DUP 2 <
      IF DROP ."<2 ROLL IMPOSSIBLE"CR
      ELSE DUP 0
           DO SWAP R> R>
             ROT >R >R >R
           LOOP 1
           DO R> R> R> R> SWAP >R
             ROT ROT >R >R
           LOOP R>
   THEN ;
```

sans encombre de 0 à 65535, avec bien sûr, des opérateurs spéciaux. Découvrons le plus important de ceux-ci : U. (u --) qui affiche le sommet de pile 16 bits en non-signé.

La coutume veut que l'on remplace le n de la notation conventionnelle par u lorsqu'il s'agit d'une valeur non-signée, réservant le n à la valeur signée.

Forgeons maintenant un outil qui nous permettra de faire le même type de conversion que D? mais en travaillant, cette fois-ci, sur les valeurs absolues. Le principe est le même, il suffit de remplacer le point (.) par l'opérateur U. dans la définition du mot. Comme il s'agit de valeurs "Absolues", nous l'appellerons A? :

```
: A? DECIMAL U. BIN ;
```

Et puisque nous sommes toujours en binaire, nous pouvons faire, avec D? et A?, une série de comparaisons très instructives (voir encadré "Comparaisons instructives").

Une bogue ? Mais non !

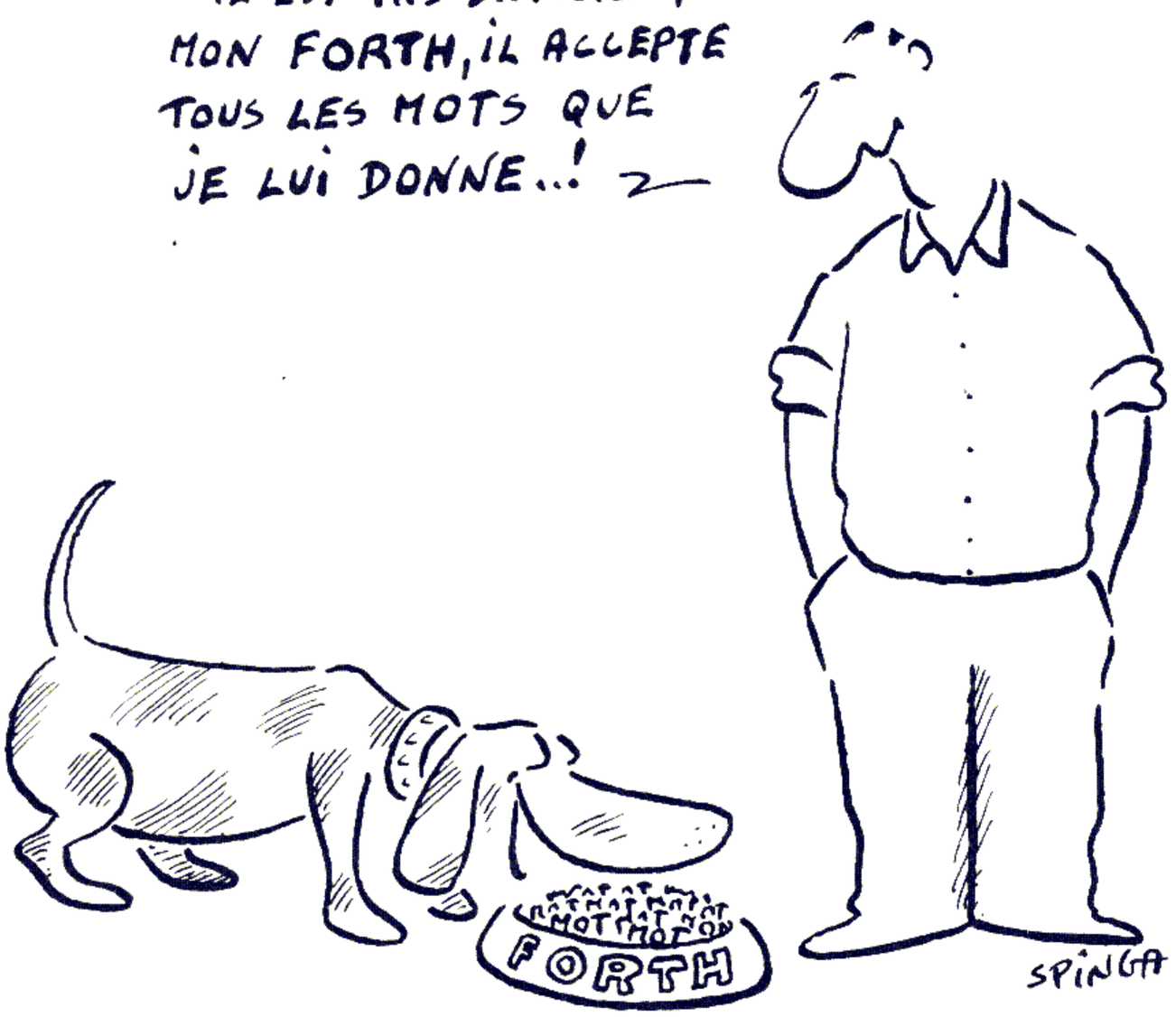
La première remarque à faire est que le premier bit du deuxième octet (le plus à gauche) est celui qui détient le signe des valeurs signées : s'il est à 1, la valeur décimale sera négative, alors que s'il est à 0, elle sera positive. La deuxième remarque est que le cycle positif-négatif est devenu apparent (avec les "non-signés", on franchit la barre de 32768 sans difficulté).

Enfin, nous constatons que −32768 et 32768 ont la même valeur binaire. Il

Le sommet de pile est conventionnellement à droite. La pile est supposée chargée de 1 2 3 4 5 6 7 8 9 10 (RETURN). Le "démontage" est fait ici avec 4 XROLL.

Liste	Pile données	Pile retour	Commentaires
XROLL	5 6 7 8 9 10 4	—	Début du mot en exécution
DUP	6 7 8 9 10 4 4	—	
2	7 8 9 10 4 4 2	—	Second terme comparaison
<	6 7 8 9 10 4 0	—	Booléen 0 car 4 < 2 faux
IF	5 6 7 8 9 10 4	—	Mange le booléen et renvoie à ELSE
DROP	idem	—	Si vrai, aurait détruit 4 et aurait affiché texte
." 2 XROLL IMPOSSIBLE"	idem	—	
CR	idem	—	Alinéa
ELSE	idem	—	Reprend traitement après booléen faux
DUP	6 7 8 9 10 4 4	—	Duplique n
0	7 8 9 10 4 4 0	—	Seconde borne du DO...LOOP
DO	5 6 7 8 9 10 4	4 0	Envoie bornes pile retour
SWAP	5 6 7 8 9 4 10	4 0	Interversion sur pile
R> R>	7 8 9 4 10 0 4	—	Vide pile retour
ROT	7 8 9 4 0 4 10	—	Rotation sur pile
>R >R >R	4 5 6 7 8 9 4	10 4 0	Recharge la pile retour
LOOP	idem	10 4 1	Incrémente l'index, compare et renvoie après DO
SWAP	4 5 6 7 8 4 9	10 4 1	SWAP pour la deuxième fois
R> R>	6 7 8 4 9 1 4	10	Réempile index et limite
ROT	6 7 8 4 1 4 9	10	Passe le 9 au sommet pile
>R >R >R	3 4 5 6 7 8 4	10 9 4 1	Passe sur pile retour 9 et limites (index au-dessus)
LOOP	idem	10 9 4 2	Incrémente l'index
SWAP	3 4 5 6 7 4 8	10 9 4 2	Interversion sur pile
R> R>	5 6 7 4 8 2 4	10 9	Passe limites sur pile
ROT	5 6 7 4 2 4 8	10 9	Rotation sur pile
>R >R >R	2 3 4 5 6 7 4	10 9 8 4 2	Pile retour se charge de plus en plus
LOOP	idem	10 9 8 4 3	Index approche limite
SWAP	2 3 4 5 6 4 7	10 9 8 4 3	Quatrième SWAP sur pile
R> R>	4 5 6 4 7 3 4	10 9 8	Evacuation limites sur pile
ROT	4 5 6 4 3 4 7	10 9 8	Rotation sur pile
>R >R >R	1 2 3 4 5 6 4	10 9 8 7 4 3	La bascule continue
LOOP	idem	10 9 8 7	LOOP incrémente l'index, constate l'égalité 4=4, écrête les limites en pile de retour et passe à la suite
1	2 3 4 5 6 4 1	10 9 8 7	Empilage de 1 (nouvel index)
DO	1 2 3 4 5 6	10 9 8 7 4 1	Bien sûr ! Il faut remettre la pile retour dans l'état primitif
R> R> R> R>	4 5 6 1 4 7 8	10 9	Fichtre, ça déboule ferme !
SWAP	4 5 6 1 4 8 7	10 9	Connu
>R ROT ROT	3 4 5 6 8 1 4	10 9 7	Accélérons, opérations sur les deux piles
>R >R	1 2 3 4 5 6 8	10 9 7 4 1	Envoi limites et index sur retour
LOOP	idem	10 9 7 4 2	Incrémentation et renvoi
R> R> R> R>	5 6 8 2 4 7 9	10	Pas de surprise !
SWAP	5 6 8 2 4 9 7	10	Là non plus !
>R ROT ROT	4 5 6 8 9 2 4	10 7	Même chose que plus haut
>R >R	2 3 4 5 6 8 9	10 7 4 2	Bascule des limites
LOOP	idem	10 7 4 3	DO...LOOP se termine
R> R> R> R>	6 8 9 3 4 7 10	vidée	On y arrive !...
SWAP	6 8 9 3 4 10 7	—	Connu
>R ROT ROT	5 6 8 9 10 3 4	7	L'ordre en pile revient
>R >R	3 4 5 6 8 9 10	7 4 3	Pour le LOOP
LOOP	idem	7	4=4, on saute à la suite
R>	4 5 6 8 9 10 7	vidée	Ouf ! on peut clore
THEN (et FIN par;)	idem	—	Sans oublier de refermer le "IF" !!!

Avec 4XROLL, on fait passer en sommet de pile le quatrième élément de la pile à partir du sommet : le 7. Il se retrouve bien au sommet maintenant ! (Les signes > doivent être collés au R, avant ou après, pour donner >R ou R>. Une erreur à ce niveau et tout est à refaire !)



Comparaisons instructives

Nombre binaire introduit	Valeur signée avec D?	Valeur absolue avec A?
0000000000000000	0 OK	0 OK
0111111111111111	32767 OK	32767 OK
1000000000000000	−32768 OK	32768 OK
1000000000000001	−32767 OK	32769 OK
1011111111111111	−16385 OK	49151 OK
1111111111111110	−2 OK	65534 OK
1111111111111111	−1 OK	65535 OK

faut de plus être très prudent avec la "base courante", sinon on se heurte à des remarques acides. Ainsi, 44 D? entraînera :
44 ?? N'EXISTE PAS

Ce qui, avouons-le, est un comble ! Mais il est sûr qu'en binaire, 44 n'existe pas !!

Et puisqu'on est dans les exercices, repassons en décimal et entrons :
8 BASE ! puis BASE ? donne 10 OK.
4 BASE ! puis BASE ? donne 10 OK.
32 BASE ! puis BASE ? donne 10 OK.

Quouaaahhh ? La base a changé trois fois, et le résultat reste 10 ! Eh oui, 8 en base 8 (l'octal), 4 en base 4, et même 32 en base 32 s'écrivent bien : 10.

Lors de mes débuts en Forth, j'avais rapporté ma console à la boutique, persuadé qu'elle ne marchait pas. Et le revendeur, après l'avoir testée pour obtenir confirmation de ce fait étonnant, était prêt à me l'échanger. Il a testé les autres consoles en magasin qui, oh ! horreur, faisaient la même erreur ! Il en déduisit que toute la fabrication avait une bogue ! Il me déclara gravement qu'il fallait attendre le résultat de la réclamation auprès du fabricant... Je n'ai bien entendu, et pour cause, jamais plus entendu parler de lui : c'est en réfléchissant que j'ai trouvé la raison de cette fausse anomalie (mais si, je vous assure, je suis génial !).

Cette petite histoire est rigoureusement authentique, et prouve combien le Forth est peu connu des réseaux commerciaux.

Puisque la "double longueur" existe, elle doit bien pouvoir être utilisée. En effet, Forth n'a nullement l'intention de s'arrêter à 65535. Mais aligner 32 bits pour faire une démonstration convaincante de ce qui se passe avec la juxtaposition de deux cellules, paraît difficile. Contentons-nous de raisonner par analogie.

Vers les grands nombres...

En fait, tout va se passer exactement comme pour les cellules de 16 bits. En nombres signés, le tout premier des 32 bits notera le signe, avec le même 0 pour le + et 1 pour −. Et cela permettra d'utiliser des valeurs de − 2 147 483 648 à + 2 147 483 647, avec le même cycle positif-négatif, simplement plus grand. Quant à l'échelle des valeurs non-signées, elle ira de 0 à 4 294 967 295 ! De quoi faire !

Pour manier de tels nombres, Forth a besoin d'une indication afin de savoir si un couple de cellules doit être pris comme un tout. Pour cela, il suffit d'introduire un point (.) dans un nombre, au début ou à la fin. La place a, en fait, peu d'importance. Forth sait que l'on est en double longueur lorsqu'il y a un point, et il attend l'espace pour considérer la valeur du nombre.

Mais d'abord, quelques opérateurs qui vont servir :
D+ (d1 d2 -- dr) : prend en sommet de pile deux nombres en double longueur, en fait la somme et empile le résultat en double longueur.
D− (d1 d2 -- dr) : prend les deux nombres en double longueur du sommet de pile, retranche d2 de d1, et empile le résultat en double longueur.
D. (d --) : prélève le sommet de pile en double longueur et l'affiche.

Il y en a bien d'autres, mais il est préférable de les inventorier selon les besoins. Pour l'instant, il nous manque l'équivalent de U., c'est-à-dire un opérateur qui affichera les valeurs double longueur en non-signé.

Comme il est de très faible utilité, cet opérateur n'existe pas dans le Forth de base de certaines machines. Aussi, il faut le fabriquer. Il comprend les trois mots principaux utilisés dans les opérations de formatage :
: UD. <# #S #> TYPE ;

Et maintenant, faisons les deux plus grandes additions possibles en double longueur :
2147483647. 2147483648. D+ UD. et la réponse sera 4294967295 OK.
2147483648. 2147483648. D+ UD. et la réponse sera 0 OK. Ce deuxième résultat n'a rien pour nous surprendre, il est analogue à celui rencontré en simple longueur non-signée.

Contrôlons à présent que la place du point dans le nombre double longueur est indifférente :
2000000000. 20000.00000 D+ UD. donnera la valeur 4000000000 OK. Voilà qui est décisif, non ?

Déjà, nous pouvons faire pas mal de choses. Par exemple, fabriquer des mots, puis d'autres mots qui utilisent les premiers : c'est cela programmer en Forth ! Ainsi, créer un mot pour compter de deux en deux, de 20 à 80 inclus ; un mot pour aligner les carrés des dix premiers nombres ; un mot pour aligner les carrés des nombres pairs de 20 à 80...

Et comme me disait encore récemment J. Delafontaine, un illustre pratiquant de notre langage : « Forth et méthode font plus que Basic ni que rage ».

Bernard LAMBEY

UNE PETITE MUSIQUE DE POCHE

EN bricolant un peu le PC-1251, mais de façon purement logicielle, on peut le transformer en boîte à musique. C'est l'affaire de quelques octets en langage-machine.

Une remarque sur l'affichage du PC-1251 va nous permettre de déboucher sur des applications sonores auxquelles le bip de l'appareil ne nous a pas habitués.

La routine qui permet de maintenir l'affichage lors de l'exécution d'un programme a la forme suivante :

```
02      X       LIA     X
FF      B1      CAL     1FB1
37              RTN
```

La valeur X est donc placée en A avant l'appel des sous-programmes débutant à l'adresse 1FB1. Voyons quel est le sous-programme (1).

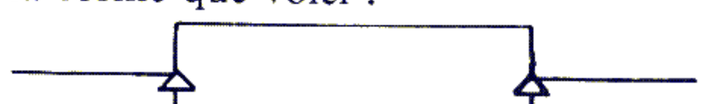
```
1FB1    12      5F      LIP     5F
        DB              EXAM
        DF              OUTC
37                      RTN
```

La valeur X se trouve donc transférée à l'adresse 5F qui est celle du port C, puis l'instruction OUTC envoie cette valeur aux périphériques. Le port C est un port de commande : chaque bit de la valeur X que l'on y place correspond à une fonction. Ainsi X = 1 (bit 0) correspond à l'affichage ; X = 32 (bit 5) au BEEP ; X = 8 (bit 3) à l'arrêt de la machine ; etc. Si X = 16 (bit 4), le haut-parleur du poquette émet un claquement très discret et reçoit alors non pas une brève impulsion, mais une tension continue.

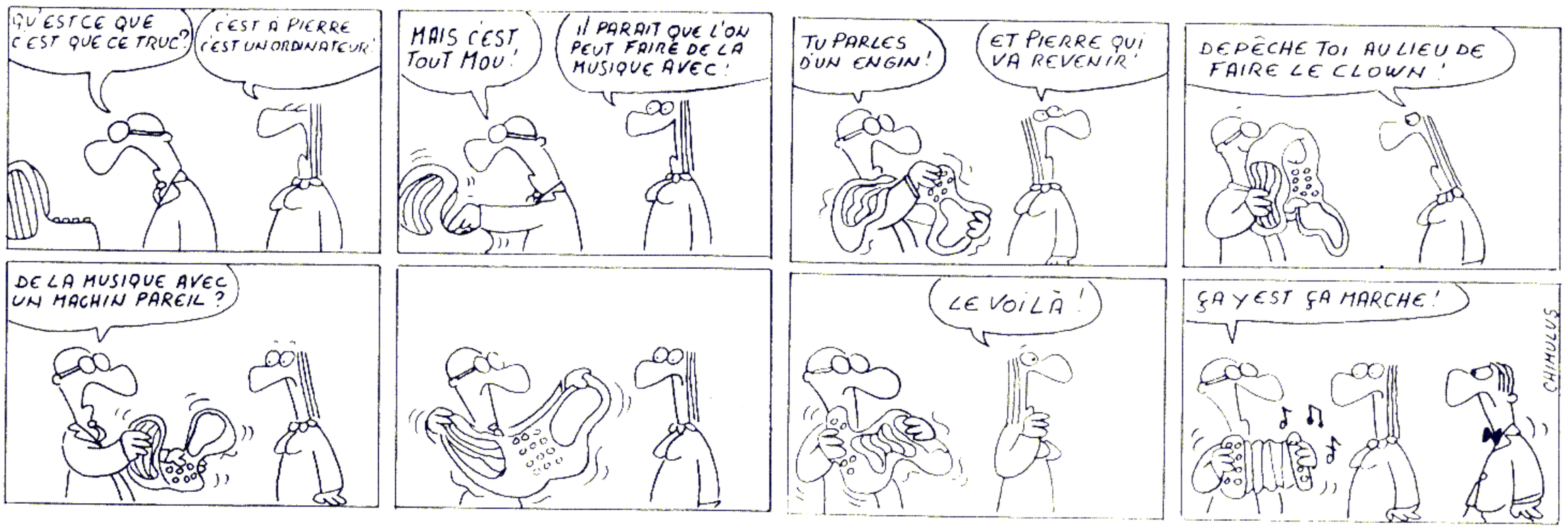
(1) Il se peut que l'adresse 1FB1 ne soit pas la bonne sur tous les modèles du PC-1251. L'ordre PEEK étant sans effet dans cette zone, on vérifiera la présence de cette routine grâce au procédé publié dans LIST *1, page 76.*

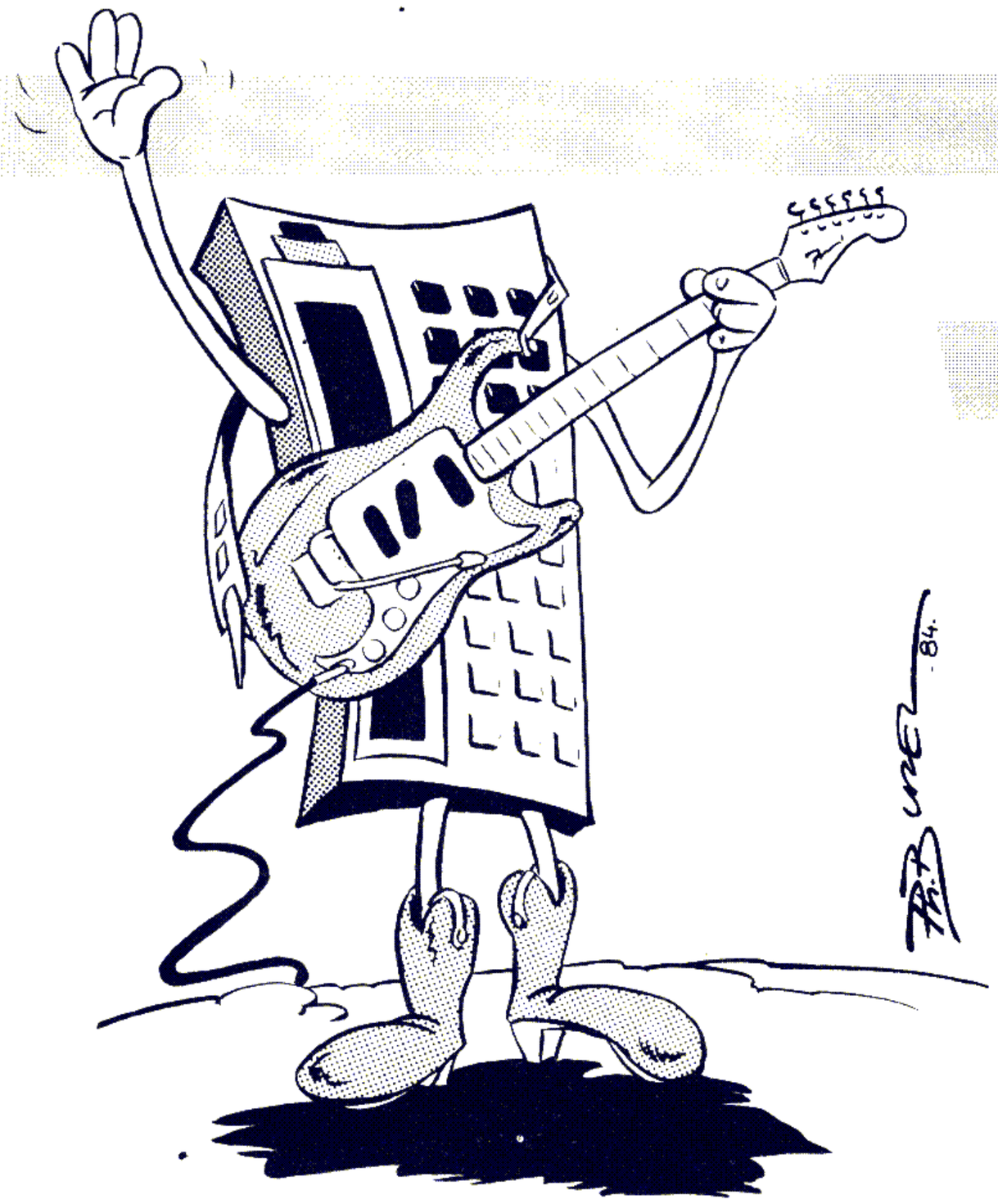
Le langage-machine s'impose

Pour vous en convaincre, essayez le programme 1. En approchant votre oreille de la machine au moment de l'exécution, vous discernerez à deux reprises un claquement. En effet, le signal que reçoit alors le haut-parleur a la forme que voici :



Cette particularité peut être mise à





Zizique de poche
Programmes pour PC-1251
Auteur Marc Leygnac

Programme 1

```
10:POKE &C000,&02,&10,&
   FF,&B1,&37
20:CALL &C000: FOR I=0
   TO 10: NEXT I
30:POKE &C001,0
40:CALL &C000: FOR I=0
   TO 10: NEXT I
50:BEEP 1
```

Programme 2

```
10:POKE &C000,&03,&00,&
   02,&10,&FF,&B1,&4E,&
   00,&02,&00,&FF,&B1,&
   C3,&29,&0C,&37
20:FOR I=0 TO 240 STEP
   16: POKE &C001,255-I
   : POKE &C007,I: CALL
   &C000: NEXT I
```

Programme 3
Un exemple d'utilisation

```
10:POKE &C000,&10,&C0,&
   0C,&03,&00,&83,&53,&
   02,&10,&FF,&B1
15:POKE &C00B,&4E,&00,&
   02,&00,&FF,&B1,&C2,&
   29,&0E,&37
20:FOR I=0 TO 4: CALL &
   C000: NEXT I
```

profit pour faire émettre par la machine des sons autres que le bip. Il suffit pour cela d'envoyer au haut-parleur plusieurs impulsions analogues à la précédente en jouant sur leur largeur (fréquence) et sur leur nombre (durée). Mais il faut que ces impulsions se succèdent très rapidement. C'est la raison pour laquelle le langage-machine s'impose.

Sur ce principe, voir ci-dessous la courte routine permettant de générer des sons.

La fréquence du son émis est ici inversement proportionnelle à T et sa durée proportionnelle à N × (α + T) où α est une constante correspondant aux instructions autres que WAIT T (pour obtenir des sons de durée fixe, quelle que soit leur fréquence, il faut donner à N une valeur fonction de T). Si l'on désire augmenter la durée d'un son, il faut alors un compteur de boucle non pas sur un octet (LIB N), mais sur deux.

La ligne 10 du programme 2 charge cette routine, puis les paramètres T et N sont « pokés » à la ligne 20 avant l'appel. Ici, tout est affaire de goût et d'imagination. Remplacez par exemple le quatrième octet (& 10) par & 20, et écoutez.

Le programme 3 en langage-machine

C 0 0 0	10	C0	0C	LIDP	C 0 0 C	DP pointe T
C 0 0 3	03	00		LIB	0	
→ C 0 0 5	83			LP	3	P pointe le registre B
C 0 0 6	53			MVDM		(DP) ← (P) : le contenu de B est placé en C00C
C 0 0 7	02	10		LIA	10	┐
C 0 0 9	FF	B1		CAL	1FB1	│
C 0 0 B	4E	00		WAIT	T	│ Impulsion de largeur
C 0 0 D	02	00		LIA	0	│ proportionnelle à (B)
C 0 0 F	FF	B1		CAL	1FB1	┘
C 0 1 1	C2			INCB		B ← B + 1
└ C 1 1 2	29	0E		JRNZM	0E	Si B ⩽ 255 on recommence
C 1 1 4	37			RTN		

0 3	N	L I B	N	N nombre d'impulsions
→ 0 2	1 0	L I A	1 0	Front montant
F F	B 1	C A L	1 F B 1	
4 E	T	W A I T	T	Attente de T cycles
0 2	0 0	L I A	0	Front descendant
F F	B 1	C A L	1 F B 1	
C 3		D E C B		B ← B - 1
└ 2 9	0 C	JRNZM	0 C	Si B ≠ 0 on recommence
3 7		RTN		

Routine de génération de sons

Avec le programme 3 et la routine correspondante, on utilise la possibilité qu'ont les programmes en langage-machine de se modifier en cours d'exécution : le PC-1251 émet des sons dont la fréquence varie. L'astuce consiste en fait à modifier à chaque boucle la valeur T de WAIT T.

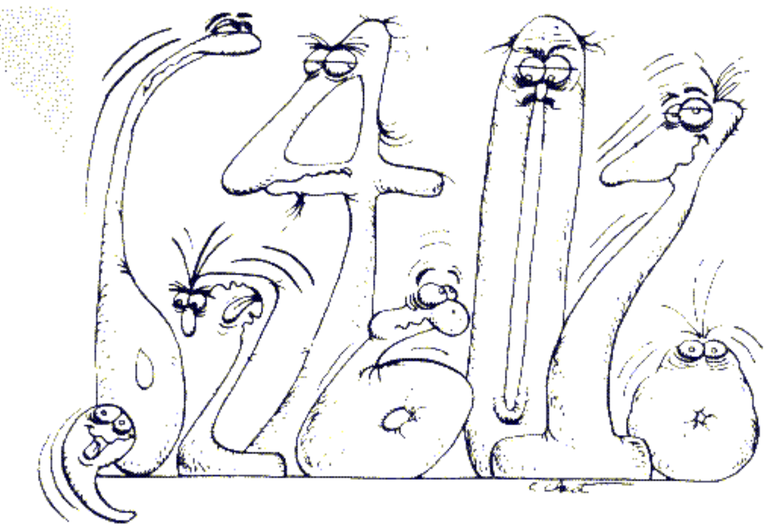
Dans l'exemple retenu, la fréquence du son diminue, mais on peut remplacer LIB 0 (03 00) par LIB 255 (03 FF) et INCB (C2) par DECB (C3) pour qu'elle augmente, ou mettre bout à bout les deux types de routines.

Et maintenant à vous de trouver de nouvelles variations musicales, ou de nouveaux bruitages.

Marc LEYGNAC

UNE PROCÉDURE PASCAL

LES NOMBRES ET LEUR FORMAT



LE langage Pascal ne dispose pas, à l'inverse du Basic, d'une instruction qui « formate » les valeurs numériques, c'est-à-dire qui les écrit sous la forme que l'on désire. Une procédure est donc nécessaire afin de préparer l'impression d'un numéro d'identification de sécurité sociale, d'un montant ou d'un code postal.

Dans la déclaration de la procédure Pascal présentée ici, procédure de « formatage », le premier paramètre, VALEUR, est une variable de type ENTIER. Il contient la valeur à formater. Son type doit être défini dans le programme ou l'unité de bibliothèque dans laquelle est insérée cette procédure. Selon les utilisations prévues, et le compilateur utilisé, ce type pourra être déclaré selon l'une des trois formes suivantes :

```
entier = integer ;
entier = integer [L] ;
entier = longinteger ;
```

Format, un paramètre essentiel

Le second paramètre indique le format à utiliser pour mettre en forme le nombre mémorisé dans VALEUR. La structure à utiliser pour les formats est spécifiée plus loin.

Enfin, le troisième paramètre, VALFOR, représente une valeur qui est retournée par la procédure. Il contient la VALeur du premier paramètre, FORmatée selon la règle indiquée par le second.

Dans cette procédure, le deuxième paramètre, FORMAT, est essentiel : il permet de spécifier la manière de présenter la VALEUR.

Ce format est une chaîne qui doit avoir la même longueur que celle désirée pour le résultat. N'importe quel caractère peut apparaître dans cette chaîne, à l'exclusion de cinq, qu'ils soient en majuscules ou en minuscules. Ces caractères ont une signification particulière, indiquée dans le tableau intitulé *Caractères interdits dans la définition du format*.

Le formatage du nombre s'effectue de la façon suivante : chaque caractère apparaissant sans le format, et qui n'est pas présent dans le tableau, est recopié tel quel dans la chaîne formatée. Pour les caractères présents dans ce tableau, la fonction correspondante est réalisée. Les exemples d'utilisation sont nombreux et très variés.

Les codes postaux : le format doit être « ZCCCC ». Le premier chiffre en effet est toujours un zéro, puisque les codes postaux sont composés de cinq caractères numériques. Le Z permet donc de forcer le premier chiffre, pour les départements de l'Ain (01) à l'Ariège (09) où les codes postaux n'ont donc que quatre chiffres lorsqu'ils sont représentés sous forme d'entiers.

Les nombres décimaux : il arrive souvent, en informatique, de mémoriser des nombres avec une unité différente de celle qui est couramment utilisée. C'est le cas de montants qui sont stockés en centimes, mais sont presque toujours exprimés en francs. Il est alors nécessaire d'insérer une virgule décimale entre l'antépénultième et l'avant-dernier chiffre. Mais un problème survient lorsque ces montants sont inférieurs à un franc. Pour représenter 2 centimes, par exemple, ils apparaissent sous la forme « ,02 » ou même « , 2 », au lieu de « 0,02 ». La procédure résoud ce problème avec le format « CCCCCZ,ZZ ». La virgule décimale est alors positionnée correctement, un zéro étant toujours placé dans la colonne des francs et des dizaines de centimes.

Les montants des chèques : les chèques émis automatiquement par les systèmes informatiques n'ont presque jamais de lignes où le montant est indiqué en toutes lettres. En contrepartie, il existe deux zones où la valeur doit être écrite sous forme numérique. Pour éviter les falsifications, il est nécessaire de faire précéder le premier chiffre de caractères de remplissage. On pourra donc spécifier à cette procédure un format comme « EEEEEEZ,ZZ » ou comme « TTTTTTZ,ZZ ». Le premier donnera, par exemple, ***3141,59 et le deuxième ---3141,59.

Les numéros de sécurité sociale : s'ils ne sont pas correctement formatés, les numéros de sécurité sociale, présentés sous la forme d'une suite de treize chiffres sans séparateur, sont difficiles à lire. Cette routine permet de formater de tels nombres. Pour les numéros de sécurité sociale, le format utilisé sera le suivant : « C.CC.CC.CC.CCC.CCC ».

Les sommes comptables : les éditions de comptabilité sont mieux présentées et plus faciles à exploiter lorsque les montants sont présentés sous forme de sommes. Le format utilisé pour un montant de 2718280,00 F sera donc celui-ci : « CC CCC CCZ,ZZ ». Il permettra d'afficher ou d'imprimer cette somme sous la forme : 2 718 280,00.

Malgré ses performances, cette procédure reste assez simple sur le plan de la programmation. Elle utilise une fonction interne, appelée CARACTERE. Son rôle est de retourner soit un chiffre appartenant au nombre qui est mémorisé dans le paramètre VALEUR de la procédure principale FORMATER, soit de retourner le caractère de remplace-

Procédure de formatage

```
procedure formater (valeur : entier ; format : string ; var valfor : string) ;
var     i      : integer ;
        signe  : char ;
        function caractere (remplacant : char) : char ;
        var    ancien : entier ;
        begin
          if valeur = 0
            then
              caractere : = remplacant
            else
              begin
                ancien : = valeur ;
                valeur : = valeur div 10 ;
                caractere : = chr (trunc (ancien-(10*valeur)) + ord('0')) ;
              end
        end ;
begin
  valfor : = format ;
  if valeur > = 0
    then
      signe : = '+'
    else
      begin
        signe : = '-' ;
        valeur : = -valeur
      end ;
  for i : = length (format) downto 1 do
    case format [i] of
      'C','c' : valfor [i] : = caractere (' ') ;
      'E','e' : valfor [i] : = caractere ('*') ;
      'S','s' : valfor [i] : = signe ;
      'T','t' : valfor [i] : = caractere ('-') ;
      'Z','z' : valfor [i] : = caractere ('0')
    end
end ;
```

Caractères interdits dans la définition du format

Caractère	Mnémonique	Signification
C ou c	Chiffre	Mettre à cet emplacement le prochain chiffre du nombre. Si tous les chiffres ont déjà été placés,mettre un espace.
E ou e	Etoile	Mettre à cet emplacement le prochain chiffre du nombre. Si tous les chiffres ont déjà été placés, mettre une étoile.
S ou s	Signe	Mettre à cet endroit le signe du nombre, c'est-à-dire le caractère "+" ou "-".
T ou t	Tiret	Mettre à cet emplacement le prochain chiffre du nombre. Si tous les chiffres ont déjà été placés, mettre un tiret.
Z ou z	Zéro	Mettre à cet emplacement le prochain chiffre du nombre. Si tous les chiffres ont déjà été placés, mettre un zéro.

ment qui lui est fourni s'il ne reste plus de chiffres dans le nombre VALEUR.

Pour réaliser ce traitement, cette fonction a besoin de l'opération modulo. Comme celui-ci n'est pas toujours implanté pour les entiers longs sur certains compilateurs Pascal, il est simulé à l'aide de la division entière.

Les symboles passés en revue

Avec cette fonction, le corps de la procédure FORMATER est assez court. Tout d'abord, le format à utiliser est stocké dans la chaîne qui sera formatée. Cela permet de dimensionner correctement cette variable, et de simplifier le traitement pour les caractères qui n'ont pas à être remplacés. Dans un second temps, cette procédure détermine la valeur de la variable SIGNE chargée de mémoriser un caractère représentant le signe du nombre.

Tous les symboles du format qui a été indiqué à la procédure sont ensuite examinés. Ils sont passés en revue à partir de la fin, puisque le nombre à formater doit toujours être cadré à droite. Les caractères ne représentant pas une commande du format ne sont pas modifiés, et sont intégralement recopiés du format à la chaîne formatée. Par contre, dès qu'un caractère indiquant une commande est détecté, il est modifié. La procédure fait donc dans certains cas un appel à la fonction CARACTERE, en lui indiquant le symbole de remplacement. Cette routine lui retourne alors, soit un chiffre de VALEUR, soit ce même caractère. Si la commande spécifiait un signe, celui mémorisé dans la variable SIGNE est tout simplement stocké dans la chaîne formatée.

Il faut noter que certains compilateurs, strictement conformes à la norme du Langage Pascal définie par Wirth, génèrent un code qui a un résultat indéfini lorsqu'une instruction CASE traite un cas qui n'a pas été prévu. Pour remédier à ce problème, il suffit de consulter l'encadré ci-dessous. Et tous les formatages deviennent alors possibles.

Thierry CHAMORET

Dans certains cas, le résultat est indéfini...

Lorsqu'une instruction CASE traite un cas qui n'a pas été prévu, certains compilateurs génèrent un code au résultat indéfini. Dans ce cas, il faut insérer dans la procédure de formatage, la condition :

if format [i] *in* ['C','E','S','T','Z','c','e','s','t','z'] *then*

Cette condition doit être placée avant l'instruction :

case format [i] *of*

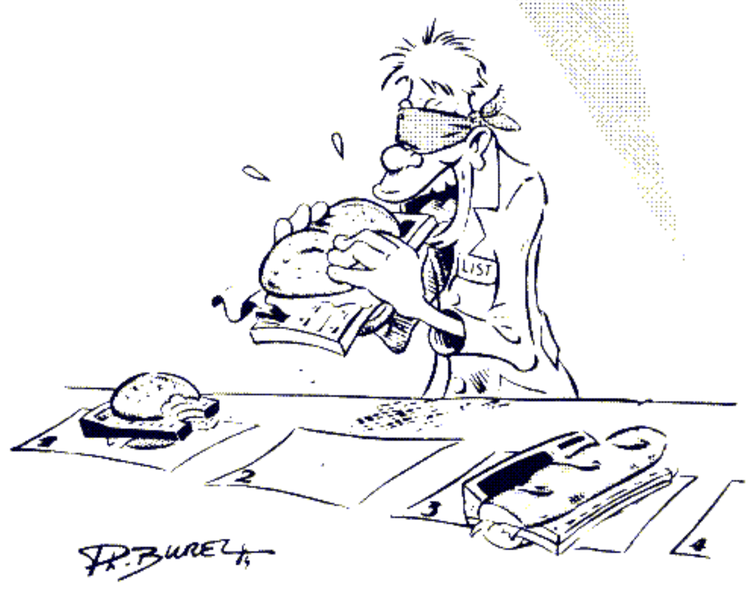
Elle permet ainsi d'éliminer les cas non prévus du sélecteur.

LES DIX TESTS DE LIST

MESURER LE BASIC

POUR évaluer en partie les performances d'un ordinateur donné, et plus précisément les qualités de son Basic, nous avons retenu provisoirement dix tests. Ils permettent de mesurer la vitesse avec laquelle la machine exécute ses appels de sous-programmes et diverses instructions de traitement de chaînes de caractères, de calculs arithmétiques, d'opérations sur les tableaux de variables, etc.

Si les dix tests présentés ici permettent de se faire une première idée de la rapidité de son ordinateur, il est souvent nécessaire de les adapter au Basic de chacun. Ainsi, les fonctions qui permettent de programmer le tracé graphique (test 8), l'écriture ou la lecture de fichiers (tests 9 et 10), quand elles existent, ne sont pas les mêmes sur tous les matériels. Par exemple, Alice 90, Vic 20, ZX81... ne possèdent pas de fonction graphique, et donc le test 8 ne peut pas leur être appliqué. De même, sur de nombreux ordinateurs, la lecture et l'écriture de fichiers sont impossibles à cause de l'absence de fonctions d'ouverture de fichiers (voir les cases « vides » du tableau).



En revanche, sur Dai 48K, l'écriture (ainsi que la lecture) d'un fichier (tests 9 et 10) peut avoir lieu de deux façons. L'écriture 10000 fois répétée d'un même secteur du disque dure environ 10 500 secondes (résultat publié dans *LIST* 4) : à chaque fois, la tête se positionne sur le secteur en question. Mais l'écriture de 10000 chaînes « LISTEST » sous forme de fichier à accès direct ne dure plus que... 77 secondes.

Pour ce qui est des comparaisons, elles ne peuvent être faites qu'entre des matériels « comparables ». Les performances d'un PC-1260, ordinateur de poche, ne peuvent pas être mesurées à celles d'un BBC, ordinateur « familial » au Basic très complet et très rapide (voir l'essai dans *LIST* 3, page 30). Mais il est toujours possible, et plus raisonnable, de comparer les ordinateurs de poche entre eux, ou les ordinateurs ayant un même processeur (Colour computer 2, TO7 et MO5, BBC et Electron... — voir le tableau).

LIST

Test 1 - *Boucle vide*

```
10  FOR I = 1 TO 10000
20  NEXT I
30  END
```

Test 2 - *Sous-programmes*

```
10  FOR I = 1 TO 10000
15  GOSUB 100
20  NEXT I
30  END
100 GOSUB 110
110 RETURN
```

Test 3 - *Matrice*

```
 5 DIM A(10,10)
10  FOR I = 1 TO 10
12  FOR J = 1 TO 10
13  FOR K = 1 TO 100
15  A(I,J) = K
17  NEXT K
18  NEXT J
20  NEXT I
30  END
```

Test 4 - *Opérations sur les chaînes de caractères*

```
 5  A$ = « LISTEST »
10  FOR I = 1 TO 10000
15  B$ = LEFT$(A$,2)
      + MID$(A$,3,3)
      + RIGHT$(A$,2)
20  NEXT I
30  END
```

Test 5 - *Arithmétique*

```
10  FOR I = 1 TO 10000
15  J = I*7 + 3/I
20  NEXT I
30  END
```

Test 6 - *Calcul scientifique*

```
10  FOR I = 1 TO 10000
15  J = SIN (LOG(I))
20  NEXT I
30  END
```

Test 7 - *Affichage*

```
10  FOR I = 1 TO 10000
15  PRINT CHR$(11) ;
      « LISTEST » ; I
20  NEXT I
30  END
```

Test 8 - *Tracé d'une ligne graphique*

```
10  FOR I = 1 TO 10000
15  LINE(0,0) – (319,199)
20  NEXT I
30  END
```

Test 9 - *Ecriture de fichiers*

```
 5  A$ = « LISTEST »
 6  OPEN « O », # 1,
      « FICHIER »
10  FOR I = 1 TO 10000
15  PRINT # 1, A$
20  NEXT I
25  CLOSE
30  END
```

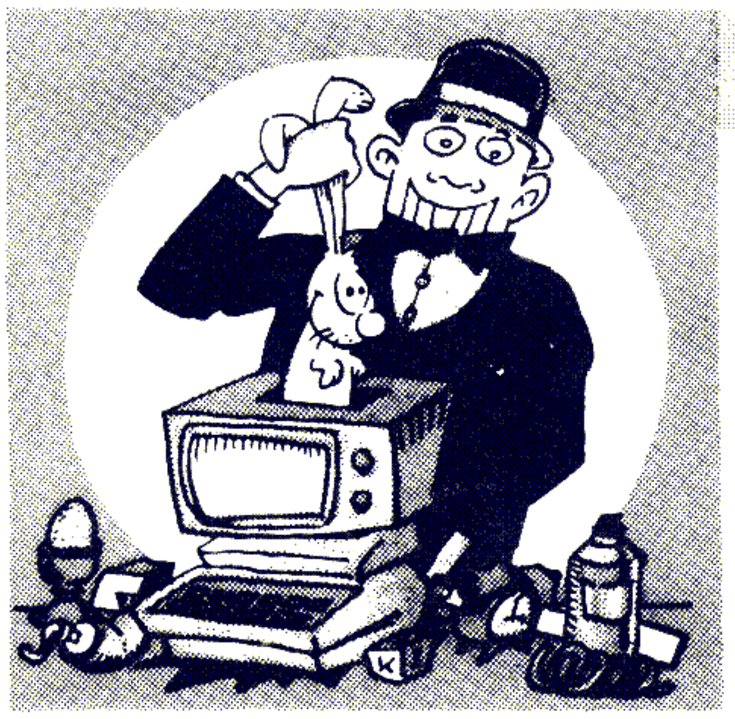
Test 10 - *Lecture de fichiers*

```
 6  OPEN « I », # 1,
      « FICHIER »
10  FOR I = 1 TO 10000
15  INPUT # 1, A$
20  NEXT I
25  CLOSE
30  END
```

Résultats des dix tests

Ordinateurs	Test 1	Test 2	Test 3	Test 4	Test 5	Test 6	Test 7	Test 8	Test 9	Test 10	Processeur
Alice 90	12	38	54	135	87	460 9 chiffres	237				6803
Apple IIe	13	43	65	136	93	479 9 chiffres	130	484 280×160	404 disquette	710 disquette	6502
Atari 600XL	22	88	78	271	204	2040 9 chiffres	225	1766 320×192			6502 C
Atmos	17	130	89	187	123	629 9 chiffres	244	766 240×200	141 disquette	77 disquette	6502
BBC	7	15	32	41	57	405 9 chiffres	114	239 160×256	1944 cassette	1944 cassette	6502 A
Canon X-07	37	79	100	236	383	1680 14 chiffres	2420	2830 120×32	43300 cassette	43300 cassette	NSC 800
Colour computer 2	14	47	69	173	116	598 9 chiffres	168	880 256×192	998 cassette	998 cassette	6809
Commodore 64	15	44	75	155	105	295 9 chiffres	200	731 320×200	394 disquette	323 disquette	6510
Dai 48K	5	30	21	87	87	799 6 chiffres	427	2518 336×256	77 disquette	62 disquette	8080 A
Dragon 32	14	49	68	162	112	590 9 chiffres	262	874 256×192	1103 cassette	1103 cassette	6809 E
Electron	9	20	43	55	64	578 9 chiffres	191	299 160×256			6502 A
Hector HRX	7	54	34	86	57	202 7 chiffres	1180	423 231×243			Z80
HP-71B	95	206	97	266	262	823 15 chiffres	1146		420 mémoire	200 mémoire	HP
HX-20	25	71	152	209	179	558 9 chiffres	1973	1234 120×32	2460 cassette	2400 cassette	6301
Lynx 48K	10	34	38	34	124	814 7 chiffres	4400	2520 256×248			Z80
MO 5	16	45	87	135	122	376 6 chiffres	295	1430 320×200	940 cassette	940 cassette	6809
MPF II	15	46	73	170	98	505 9 chiffres	337	517 280×192			6502
PB-700	115	547	705	1500	750	2100 12 chiffres	1450	2750 160×32			Casio
Oric-1	20	150	104	217	142	1058 9 chiffres	284	897 240×200			6502
PC-1260	75	296	399	970	740	3180 12 chiffres	1220				Sharp
PC-1350	69	316	384	940	710	3180 12 chiffres	1620	3500 150×32			Sharp
QX-10	21	56	85	146	99	305 16 chiffres	499	67 320×200	132 disquette	126 disquette	Z80
Spectrum	42	106	115	195	132	1180 8 chiffres	243	610 256×176			Z80
TI99/4A	28	58	138	2320	168	3190 14 chiffres	2500		1010 disquette	1840 disquette	TMS 9900
TO 7	18	46	100	138	124	417 6 chiffres	222	1037 320×200			6809
TRS-80 Modèle 1	27	86	111	213	166	516 6 chiffres	263				Z80
Vic 20	12	37	63	130	88	445 9 chiffres	173		369 disquette	310 disquette	6502
Yeno MSX	20	41	60	108	185	1937 16 chiffres	139	1165 256×192	1863 cassette	1863 cassette	Z80 A
ZX81 (mode fast)	45	100	100	160	125	1150 9 chiffres					Z80
ZX81 (mode slow)	176	380	400	640	500	4540 9 chiffres	5300				Z80

Les temps sont exprimés en secondes



LA BOÎTE A MALICES...

PRENEZ un programme et ôtez-en très soigneusement toutes les astuces, des plus élémentaires aux plus subtiles. Vous êtes certain de n'en avoir laissé passer aucune ? Bien. Que reste-t-il ? Rien, ou peut-être une bogue ou deux (tout le monde peut se tromper). En fait, tout programme n'est qu'une suite d'astuces. Dans les pages qui suivent, vous en trouverez un grand nombre. Certaines sont de portée très générale. D'autres ne valent que pour un matériel particulier. Mais dans tous les cas, vous aurez intérêt à être curieux, à fouiner dans la boîte à malices. Même s'il ne s'agit pas de votre machine, vous trouverez souvent des idées à reprendre. Par ailleurs, vous avez sans doute vos propres recettes, vos façons de faire... Si ce ne sont pas des secrets que vous cherchez à conserver jalousement, faites-en part au journal. Celles qui nous paraîtront les plus intéressantes enrichiront à leur tour la boîte à malices. Tous les lecteurs pourront ainsi en profiter.

LIST

BONNES ADRESSES

Voici une liste de variables système et de vecteurs, utiles en mémoire vive, qui va certainement intéresser les possesseurs d'Oric-1 et d'Atmos.

Les adresses énumérées sont en hexadécimal et elles sont souvent les mêmes sur les deux versions (Oric-1 et Atmos) ; lorsqu'elles diffèrent d'un ordinateur à l'autre, il en est fait mention.

12, 13 : pointeur de l'écran où sera effectué le prochain PRINT ou affichage avec les routines de la mémoire vive.

18, 19 : pointeur utilisé par l'interpréteur Basic pour stocker dans la table des mots du Basic (en C0EA) l'adresse du caractère qui est comparé aux caractères des instructions.

1A, 1B, 1C : contiennent le vecteur de retour à la saisie Basic. On y trouve le code machine JMP AFFCH qui affiche le message "Ready". Attention, sur Oric-1, AFFCH = CBED, mais sur Atmos AFFCH = CCB0.

Ce vecteur qui permet de détourner l'affichage de "Ready" pourra être utilisé en cas de protection de programmes ou de changement de message par exemple.

2F : adresse de stockage de l'octet traité par les routines cassettes.

30 : position horizontale du curseur. Une modification de cet octet n'a aucun effet.

31 : sur Oric-1, il s'agit de la longueur d'une ligne sur l'imprimante, modulo 13 (une des bogues du Basic de cette machine). Sur Atmos, c'est la longueur d'une ligne à l'écran.

33, 34 : utilisés pour le passage de paramètres entiers (2 octets). Attention, les valeurs déposées dans ces octets sont très "volatiles", il vaut mieux les traiter en langage-machine directement.

35 à 83 : mémoire tampon du clavier. Tous les caractères frappés au clavier y sont déposés, les uns après les autres, sous forme d'une liste (structure FIFO, First In, First Out, c'est-à-dire premier entré, premier sorti).

ET ATMOS

Après avoir été interprétés, les mots-clefs du Basic y sont redéposés sous forme de code et les autres caractères en code ASCII, le tout se terminant par un code nul.

91, 92 : utilisés entre autres par la routine de restauration des lignes Basic. Au retour de cette routine, ils contiennent l'adresse de la fin du programme. C'est particulièrement utile pour rétablir ce dernier après un NEW.

9A, 9B : contiennent l'adresse du début de la zone Basic. Par défaut, c'est l'adresse 501. L'adresse précédente (500) doit contenir un 0.

9C, 9D : où se trouve l'adresse de la fin du programme Basic et du début de la zone des variables.

Les problèmes que vous rencontrez avec l'Oric-1 lors du chargement de zones de mémoire viennent de ces octets. Ils sont positionnés à tort au sommet de la zone chargée et lors de la modification d'une ligne de programme, des blocs de mémoire sont intempestivement déplacés ce qui provoque à l'écran les résultats curieux que vous avez certainement pu déjà observer ; et si le pointeur est placé trop haut, vous obtenez le message OUT OF MEMORY ERROR.

9E, 9F : fin de la zone des variables simples.

A0, A1 : fin de la zone des variables simples et des variables en tableaux.

A2, A3 ou A4, A5 : fin de la zone des chaînes de caractères. Lorsqu'une variable alphanumérique est créée, les caractères qui la composent se placent en ordre décroissant à partir de cette adresse. Concluez-en qu'au départ, lorsqu'il n'y a pas encore de variables, ce pointeur contient l'adresse de la fin de la zone Basic (voir pointeur suivant).

A6, A7 : adresse de la fin de la zone Basic (ou HIMEM).

A8, A9 : numéro de la ligne Basic en cours d'exécution.

AA, AB ou AC, AD : numéro de ligne où a été effectué le dernier BREAK.

B0, B1 : adresse du prochain DATA à lire. Peut être modifié à votre gré pour simuler un RESTORE L,N par exemple.

B4, B5 : nom de la dernière variable créée tel qu'il est écrit en mémoire, c'est-à-dire avec deux codes ASCII dont le bit 7 dépend du type de variables.

B6, B7 ou B8, B9 : contient l'adresse où a été placée la dernière variable créée, numérique ou alphanumérique.

D0 à D5 : accumulateur flottant 1.

D8 à DD : accumulateur flottant 2.

La structure des accumulateurs est la suivante : le premier octet contient l'exposant, le bit 7 en indique le signe (1 = positif, 0 = négatif); sur les quatre octets suivants se trouve la mantisse 32 bits, et sur le dernier le signe de celle-ci (bit 7 à 1 s'il est positif, à 0 s'il est négatif).

E2 à F2 : routine GETCAR, place dans l'accumulateur du 6502 le code ASCII ou le code du mot Basic pointé par l'adresse contenue en E9, EA. On peut modifier cette routine pour redéfinir la syntaxe des instructions du Basic.

E9, EA : voir pointeur précédent.

FA à FE : contiennent en virgule flottante le nombre servant de base à la génération des nombres aléatoires (fonction RND). En modifiant ces deux octets, on peut simuler une instruction RANDOMIZE.

Cette liste indique quelles sont les principales variables système de la page zéro de la mémoire vive. Elle peut bien entendu être complétée.

Denis SEBBAG

PC-1251

PEUT-ON ENCORE RÉSERVER ?

Le PC-1251 présente 18 touches dites « de réserve » regroupées dans les deux rangées inférieures du clavier. L'utilisateur dispose de 48 octets de mémoire vive pour donner à ces touches, quand elles sont shiftées, la signification qu'il veut. C'est une disposition très pratique que de pouvoir ainsi personnaliser en partie le clavier.

Par ailleurs, comme sur la plupart des poquettes Sharp, l'ordre MEM indique la mémoire restant libre pour le programme ou les données. Mais il n'existe pas d'ordre permettant de savoir où l'on en est de la mémoire de réserve.

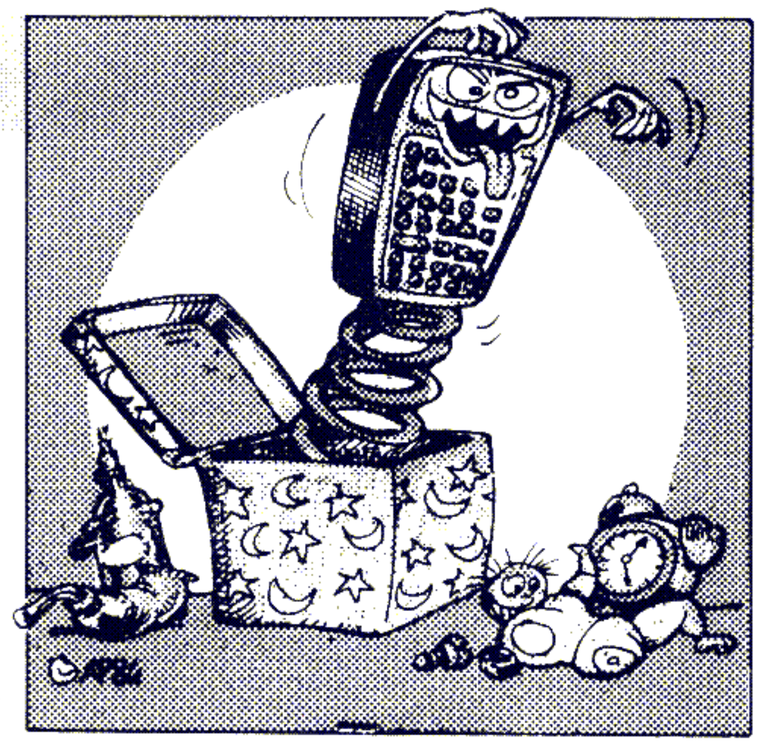
Voici, en trois lignes, un petit utilitaire qui affiche le nombre d'octets de réserve encore disponibles. Pour l'interroger, presser DEF M.

```
997 ''M'' A = 32767 : M = 49
998 M = M - 1 : A = A + 1 : GOTO (PEEK A = 0 OR PEEK A = 255) + 998
999 PRINT ''MEM RES :'';M
```

Jean-Antoine BERRO

BASIC

OBTENTION D'UNE DONNÉE MANQUANTE



Pour remplir un tableau de données à partir d'un clavier d'ordinateur, il suffit d'introduire ces données. Mais si certaines d'entre elles (dépendantes des autres) manquent, il faut les calculer, ou les faire calculer par un programme.

Au lieu de créer un programme spécifique pour chaque donnée manquante, on peut regrouper les différents traitements possibles dans un même programme qui orientera la donnée vers le traitement qui la concerne.

Par exemple, pour remplir un tableau à trois colonnes, DISTANCE, VITESSE, TEMPS, on utilisera le programme d'obtention d'une donnée manquante (qui devra être utilisé alors comme sous-programme).

Il traite automatiquement la donnée manquante, que ce soit la distance, la vitesse ou le temps. En effet, il la repère (lignes 120 à 140) grâce au zéro contenu dans D(T) qui tient lieu de drapeau. Il l'oriente alors (ligne 150) vers le traitement qui la concerne (lignes 200 à 420) et le résultat est affiché (ligne 160).

Il faut vérifier que le tableau dimensionné, ligne 10, a bien été initialisé (toutes les valeurs à zéro). Certains Basic ne possèdent pas l'instruction INKEY$ (lignes 60 et 70). Il faudra la remplacer par GET. Quant à la ligne 150, elle peut encore s'écrire avec ON...GOSUB.

L'exemple pris ici peut être étendu à d'autres, l'essentiel étant que les données soient liées entre elles par une relation (c'est bien le cas des variables vitesse, distance et temps).

Yvon PÉRÈS

Programme d'obtention d'une donnée manquante

```
 5 CLS
10 DIM D(3)
20 PRINT AT 2,2; "1.DISTANCE"
30 PRINT AT 4,2; "2.VITESSE"
40 PRINT AT 6,2; "3.TEMPS"
50 FOR T=1 TO 2
60 IF INKEY$="" THEN GOTO 60
70 LET A=VAL INKEY$
80 PRINT AT A*2,12;"= ?"
 90 INPUT D(A)
100 PRINT AT A*2,14;D(A)
110 NEXT T
120 FOR T=1 TO 3
130 IF D(T)=0 THEN LET A=T
140 NEXT T
150 GOSUB A*100+100
160 PRINT AT A*2,1; "/";TAB 12; "= ";X
170 STOP
200 REM CALCUL DE LA DISTANCE
210 LET X=D(2)*D(3)
220 RETURN
300 REM CALCUL DE LA VITESSE
310 LET X=D(1)/D(3)
320 RETURN
400 REM CALCUL DU TEMPS (DECIMAL)
410 LET X=D(1)/D(2)
420 RETURN
```

TI-57 LCD

TROMPE-L'ŒIL

LA TI-57 LCD semble avoir des réactions surprenantes parfois. J'en ai eu une preuve en l'allumant, alors qu'elle était éteinte, sans passer par ON. J'ai appuyé sans lâcher prise sur 2nd, puis sur LRN, enfin sur =, et j'ai relâché une par une ces touches dans le même ordre. L'écran s'est rempli alors de 4 et d'indicateurs. En effectuant l'opération inverse avec la touche + /– au lieu de =, j'ai tout effacé : + /– puis LRN et enfin 2nd. Comment expliquer un tel comportement ?

Mario TAGLIARINO

Une règle élémentaire permet d'allumer ou d'éteindre la TI-57 LCD, sans passer par ON ou OFF : joindre une des touches du haut du pavé avec l'une de celles placées sous le OFF (pour éteindre) ou sous le ON (pour allumer), à angle droit et par l'intermédiaire d'une ou de plusieurs touches. Cette dernière (ou ces dernières) opère la jonction entre la rangée horizontale et la rangée verticale.

Par exemple, la machine allumée sera éteinte par R/S LBL BST ou encore par 2nd STO 7 EE). Si elle est éteinte, elle pourra être allumée par OFF BST SST ou bien par INV 4 5 8 ×. Mais un tel allumage laisse la TI-57 LCD bien inactive. Cet affichage étrange ne serait-il qu'un trompe-l'œil ?

David SEGARD

TO7

CHEFS-D'ŒUVRE EN PÉRIL...

Suis-je bête ! Sur mon TO7, je viens d'ordonner NEW de façon irréfléchie et j'ai perdu mon programme.

Mais non, mon cher Watson, la commande NEW n'a pour effet que de remettre à zéro le pointeur de ligne situé à l'adresse 26101 et codé sur 2 octets. Pour repêcher un programme (en Basic, bien sûr), il suffit de repérer où se trouve le début de la ligne suivante. Il faut pour cela se souvenir exactement de la première ligne du programme perdu.

Supposons que cette première ligne était 10 CLS. Nous trouverons en 26101 la valeur 0, puisque nous avons demandé NEW, et 0 aussi en 26102 (deuxième octet du pointeur). En 26102 et 26103, sur deux octets donc, nous trouvons le numéro de la première ligne de programme. Ici, il s'agit de 10 ; nous aurons donc 0 en 26102 et 10 en 26103.

En 26104, nous trouverons 217 (c'est le code de CLS), et 0 (fin de ligne) en 26105. La ligne suivante commence par conséquent en 26106. Or 26106, c'est égal à 26106/256 = 101 pour l'octet de poids fort, plus 250 pour l'octet de poids faible. On recharge donc le pointeur en 26101 et 26102 par POKE 26101, 101 et POKE 26102, 250. Il n'en faut pas plus. Essayez, si vous ne me croyez pas...

Jean-Paul CARRÉ

DAI

TEXTE MIXÉ AU GRAPHIQUE

Sur le Dai, la structure de la mémoire d'écran ne permet pas d'emblée de mixer du texte au graphisme. C'est bien dommage, car cela force à utiliser des routines additionnelles qui *dessinent* les lettres du texte dans le mode graphique désiré. Réalisées en langage-machine, elles sont souvent assez rapides pour faire honnêtement illusion. Mais le prix à payer est une manipulation lourde, pas toujours évidente, et surtout un encombrement indésirable de la mémoire vive.

Il existe pourtant une autre solution, à laquelle on pense rarement : la page

Graphique et texte
Programme pour Dai
Auteur Alain Mariatte
Copyright *LIST* et l'auteur


```
10    REM
20    REM ******************************************************
30    REM ***    TEXTE MIXABLE DANS LA PAGE GRAPHIQUE          ***
40    REM ***          PROGAMME POUR DAI                       ***
50    REM ******************************************************
60    REM
70    REM
100   REM
110   REM
120   MODE 6A:COLORG 0 5 10 15:PRINT CHR$(12)
130   FOR I!=0.0 TO 100.0:DRAW 0,0 RND(XMAX),RND(YMAX) 20+(I! MOD 3)
140   NEXT
150   REM
160   REM *LIGNE 100 EN MODE TEXTE 4 COULEURS,7 SCANS
170   CB=#BFEF-100*90:POKE CB,#66
175   A$="***UNE LIGNE DE TEXTE EN 4 COULEURS********"
177   L!=LEN(A$)
180   GOSUB 1190
190   REM
200   REM *LIGNE 150 EN MODE TEXTE 16 COUL.,7 SCANS
210   CB=#BFEF-150*90:POKE CB,#E6
212   A$="***UNE LIGNE DE TEXTE EN 16 COULEURS********"
214   L!=LEN(A$)
220   FLAG!=1.0:GOSUB 1190
230   REM
240   REM
250   END
1190  REM *COULEUR 0 DANS LE REGISTRE 4 (POUR QUE LA FIN DE LIGNE SOIT PROPRE)
1200  POKE CB-1,#F0
1210  REM
1220  REM *LA LIGNE EST REMPLIE DE LETTRES
1225  AD=CB-2
1230  FOR I!=0.0 TO L!-1.0
1232  A!=ASC(MID$(A$,1!,1)):POKE AD,A!
1242  IF FLAG!=0.0 THEN POKE AD-3,0:GOTO 1246
1244  R!=RND(16.0) SHL 4.0:IF R!<16.0 THEN R!=16.0:REM eviter le zero
1245  POKE AD-3,R!
1246  AD=AD-2
1248  NEXT
1249  IF FLAG!=1.0 THEN POKE CB-90+2,32:REM CODE SPACE POUR NETTOYER FIN LIGNE
1250  REM
1260  REM *ON RECHARGE LA COULEUR 15 DANS LE 4eme REGISTRE
1270  POKE CB-1-90,#FF
1300  FLAG!=0.0:RETURN
```

vidéo du Dai est totalement redéfinissable, ligne à ligne, s'il le faut. A défaut de pouvoir mixer réellement du texte au dessin, au moins peut-on simplement définir des lignes « texte » et des lignes « graphique » à volonté. Il est même possible de faire cohabiter du texte et *plusieurs* modes graphiques simultanément à l'écran, du moins dans des parties distinctes de l'écran.

Ce « miracle » est possible si l'on joue sur les deux octets de contrôle qui codent l'état de chaque ligne d'écran. Le programme de démonstration place la page graphique en haute résolution (336 × 256) 4 couleurs (ligne 120), trace aléatoirement des lignes de couleur (c'est le « dessin » !), puis redéfinit deux endroits de l'écran pour y inscrire du texte, en « pokant » les codes ASCII dans deux lignes redevenues des lignes de MODE 0, au milieu des autres qui sont restées en MODE 6.

Voyons cela de plus près : d'abord, en MODE 6, pour trouver le début d'une ligne (en partant du haut de l'écran), il suffit de faire : # BFEF-x*90, où x est le numéro cherché. L'octet présent à cette adresse est l'octet de mode de la ligne. Pour en faire une ligne de texte 4 couleurs, il faut y placer # 66 ; pour du texte 16 couleurs, # E6. En effet, # 66 s'écrit en binaire 01100110, ce qui donne :

- bits 7,6 : 01, c'est-à-dire mode 4 couleurs/caractères ;
- bits 5,4 : 10, c'est-à-dire haute résolution ou 40 caractères/ligne ;
- bits 3, 2, 1, 0 : 0110 qui code l'épaisseur de la ligne.

Pour 16 couleurs/caractères, on a # E6 (11100110) :

- bits 7,6 : 11, soit 16 couleurs/caractères ;
- les autres bits demeurent inchangés.

Pour le reste, le sous-programme débutant à la ligne 1190 place seulement le code des lettres de la chaîne A$ sur l'écran. N'oublions pas que nos deux lignes sont redéfinies en MODE 0, donc que, tout à fait classiquement, l'octet attribué à la première lettre à gauche de la ligne se trouve en « adresse du *Control Byte* - 2 » et que chaque lettre est séparée de sa voisine par *deux* octets. De même, pour 16 couleurs, n'oublions pas que l'octet de couleur d'une lettre se trouve 3 octets plus bas que la dite lettre.

Avec un peu d'entraînement, il est donc possible d'envisager la création de pages écran multimodes, superposant haute et basse résolutions, texte, etc. Intéressant, non ?

Alain MARIATTE

Apple II

CACHEZ CETTE LISTE...

Vous voulez rendre impossible le listage d'un de vos programmes ? Voici une méthode assez radicale. Pour obtenir la liste d'un programme, il faut d'abord en interrompre l'exécution. Pour cela, si aucun point d'arrêt n'est prévu, deux solutions :

- le CTRL C, facile à protéger par le ON ERR GOTO,
- le RESET.

Ce dernier travaille par indirection. C'est-à-dire qu'en cas de RESET, la main est donnée (en langage-machine) au programme débutant à l'adresse indiquée en $3F2-$3F3 (1010-1011). Attention, ceci n'est vrai que s'il s'agit d'un « démarrage à chaud » (la machine a déjà été initialisée). Le moniteur détecte le type de démarrage en examinant l'octet $3F4 (1012). Si le résultat d'un OU exclusif entre $3F3 et £$A5 correspond à ce qui se trouve en $3F4, alors c'est un démarrage à chaud, et il quitte le programme. Dans le cas contraire, c'est un démarrage à froid : la machine est ré-initialisée (équivalent de CTRL-Pomme ouverte-RESET).

Pour récupérer cette touche RESET, il va donc falloir réaliser un petit programme-machine et mettre en place ce vecteur d'indirection. Dans notre exemple, nous allons supposer l'implantation de ce programme en $300 (768). Attention, cette zone n'est pas disponible si vous utilisez le MEM/DOS 6502.

Il faut donc placer, en $3F2-$3F3 l'adresse $300, soit : POKE 1010,0 : POKE 1011,3.

Il faut ensuite mettre à jour l'octet $3F4 pour qu'il soit bien le résultat du OU exclusif entre $03 et $A5. Pour cela, point besoin de calculs, nous disposons d'une routine du Moniteur le faisant automatiquement. Elle se trouve à l'adresse $FB6F (64367 ou – 1169). Donc : CALL – 1169.

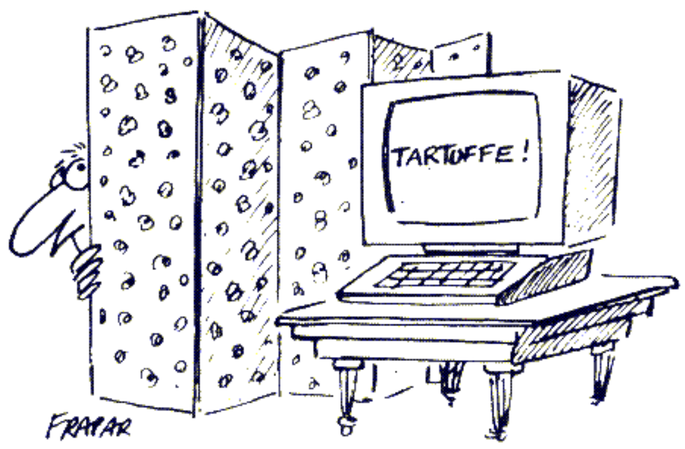
Reste à écrire notre programme. Dans l'exemple, nous allons simplement afficher à l'écran le texte "PRO-

Protection de programme pour Apple II

```
A1L       EQU   $3C
A1H       EQU   $3D
A2L       EQU   $3E
A2H       EQU   $3F
A4L       EQU   £42
A4H       EQU   $43
HOME      EQU   $FC58
MOVE      EQU   $FE2C
PWREDUP   EQU   £3F4
DEBUT     JSR   HOME      Effacement de l'écran
SUITE     LDA   £$A0
          STA   A1L       Mettre la valeur $A0 dans $3C
          LDA   £$B2
          STA   A2L       Mettre la valeur $B2 dans $3E
          LDA   £$03
          STA   A1H       Mettre la valeur $03 dans $3D
          STA   A2H       Mettre la valeur $03 dans $3F
          LDA   £$00
          STA   A4L       Mettra la valeur $00 dans $42
          STA   PWREDUP   Mettre la valeur $00 dans $03F4
          LDA   £$07
          STA   A4H       Mettre la valeur $07 dans $43
          JSR   MOVE      Transfert du texte en mémoire écran
          JMP   SUITE     Boucle sur le programme
```

Texte du message

```
03A0 – 50 52 4F 47 52 41 4D 4D
03A8 – 45 60 50 52 4F 54 45 47
03B0 – 45 60 61
```



GRAMME PROTEGE''. Nous allons donc écrire le texte en code-écran entre $3A0 et $3B2 (928 à 946) et utiliser, pour le transférer en mémoire d'écran, le sous-programme du moniteur se trouvant en $FE2C (–468). Il nécessite d'avoir positionné auparavant l'adresse de début de zone en $3C-$3D (60-61), l'adresse de fin en $3E-$3F (62-63) et l'adresse de destination en $42-$43 (66-67). Mettons-le à la quatrième ligne de l'écran, soit en $700. Un petit détail : si l'utilisateur, énervé par le message, fait un deuxième RESET, la machine sera ré-initialisée grâce à un procédé très simple, notre petit programme modifiera la valeur de l'octet $3F4 en le mettant par exemple à 0.

On trouvera page précédente le pro-

Huit lignes de Basic pour implanter la routine

```
0 ONERR GOTO 2
1 GOTO 3
2 CALL 768
3 POKE 1010,0: POKE 1011,3: CALL
      – 1169
4 DATA 32,88,252,169,160,133,60,
      169,178,133,62,169,3,133,61,
      133,63,169,0,133,66,141,244,
      3,169,7,133,67,32,44,254,76,
      3,3
5 FOR I = 768 TO 801: READ X:
POKE I,X: NEXT
6 DATA 80,82,79,71,82,65,77,77,6
      9,96,80,82,79,84,69,71,69,96,
      97
7 FOR I = 928 TO 946: READ X:
POKE I,X: NEXT
```

gramme en Assembleur. La liste en Basic permet d'implanter la routine correspondante en langage-machine et protège aussi le CTRL C. Attention, si vous la tapez au clavier, sauvez-la avant de faire un RESET...

Jacques LABIDURIE

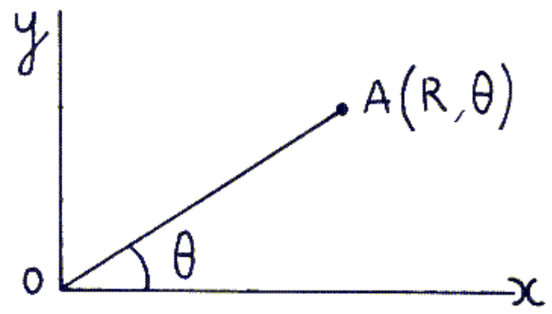
COORDONNÉES POLAIRES OU RECTANGULAIRES AU CHOIX

Le ZX 81 n'est pas doté de la fonction permettant de convertir les coordonnées polaires en rectangulaires, ni de la fonction inverse. Or ces fonctions sont très utiles pour bien des applications scientifiques et techniques, ou plus simplement pour certains programmes de jeu. Voici deux courts programmes qui permettent de les obtenir.

Programme 1
Polaires → Rectangulaires

```
5 PRINT "ANGLE ?"
6 INPUT A
10 PRINT AT 2,0 ; "R ?"
11 INPUT R
15 LET X = COS (A / 180 * PI) * R
20 LET Y = SIN (A / 180 * PI) * R
25 PRINT AT 0,7 ; A ; TAB 15 ; "X = " ; X
30 PRINT AT 2,7 ; R ; TAB 15 ; "Y = " ; Y
```

Coordonnées polaires

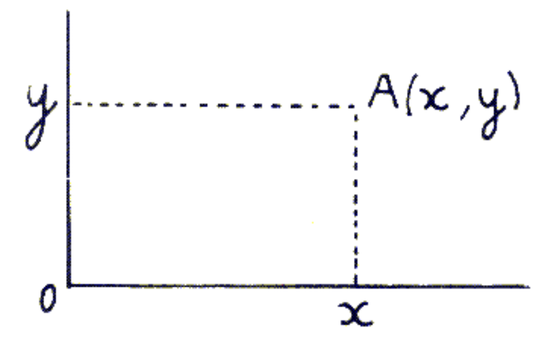


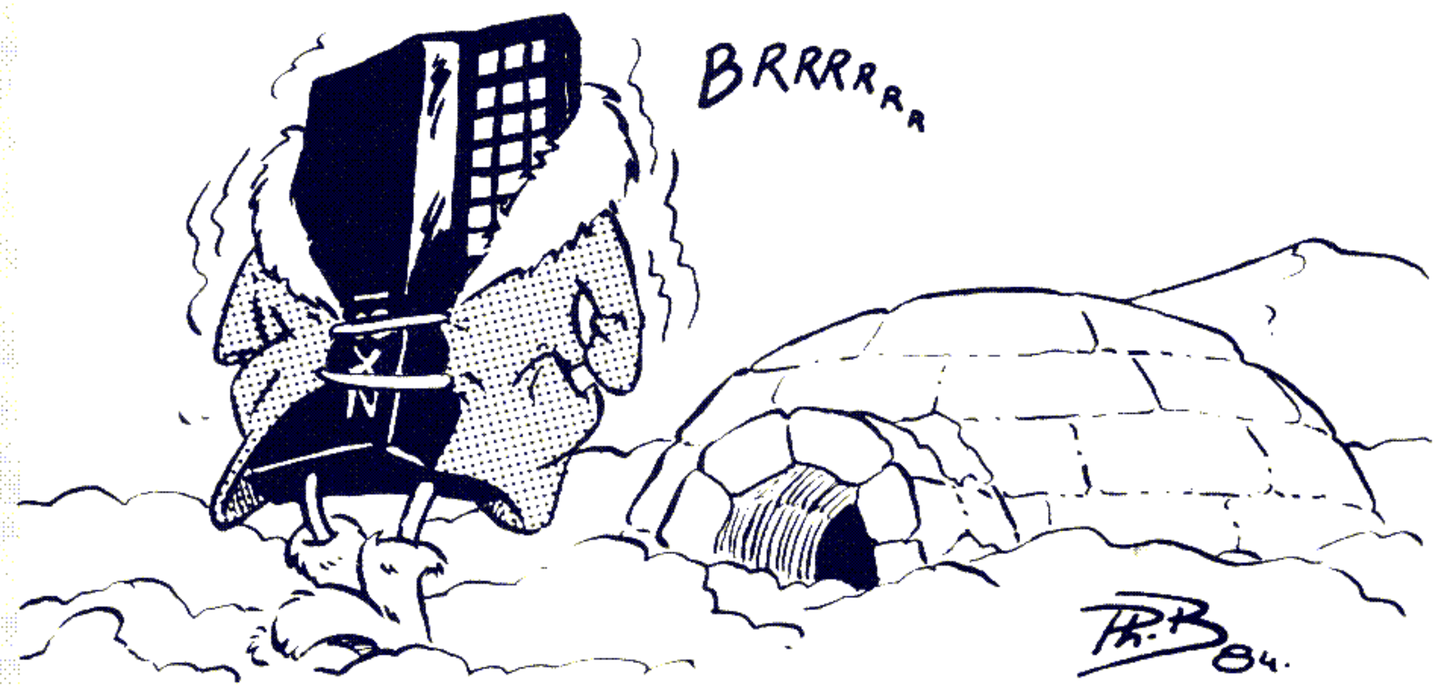
R : distance de O à A
θ : angle AOx

Programme 2
Rectangulaires → Polaires

```
5 PRINT "X ?"
6 INPUT X
10 PRINT AT 2,0, "Y ?"
11 INPUT Y
15 LET A = (ATN (Y/X) * 180/PI)
20 LET R = (Y/SIN (A/180 * PI))
25 PRINT AT 0,3 ; X ; TAB 10 ; "ANGLE =" ; A
30 PRINT AT 2,3 ; Y ; TAB 10 ; "R =" ; R
```

Coordonnées rectangulaires





On a choisi d'utiliser les degrés comme unité angulaire, mais, pour les fonctions trigonométriques, le ZX 81 ne travaille qu'en radians, d'où la conversion effectuée aux lignes 15 et 20.

Le premier des programmes (P → R), demande l'angle θ puis la distance R (du point O au point A) et il affiche les coordonnées rectangulaires X et Y du point A. Le second demande l'abscisse X et l'ordonnée Y avant d'afficher l'angle θ et la distance R.

Christophe TAVERNIER

UN BON DESSIN VAUT MIEUX...

Une quinzaine de commandes (dont chacune est ici attribuée à une touche particulière) permet de dessiner en couleurs sur l'écran de tout ordinateur au standard MSX. Le mode d'emploi résumé du programme est affiché en permanence.

Pour tracer, par exemple, un polygone segment par segment :
• appuyer sur V pour valider le premier point ;
• déplacer le curseur au moyen des quatre flèches ;
• appuyer sur T pour tirer une droite reliant le point précédent.

Pour obtenir une figure discontinue, presser sur V. Si l'on veut dessiner un cercle : on désigne son centre (une pression sur V) et l'un des points de sa circonférence (une pression sur C). Quant aux autres commandes, ce sont :
• P, peindre une figure fermée (on positionne le curseur à l'intérieur de la figure) ;
• A, annuler le dernier tracé (pour une droite) ;
• 1, 2, 3..., choisir la couleur d'écriture ;
• F , dessiner en couleur de fond sur une partie coloriée de l'écran (dans ce cas, le curseur disparaît sur une zone non peinte, et on le rend visible en changeant la couleur) ;
• G , gommer toute la surface d'un rectangle.

Une des directions possibles pour développer le programme consiste à stocker les commandes dans une table et à sauvegarder ainsi les dessins pour éventuellement les modifier.

Jacques BOISGONTIER

Dessinateur
Programme pour MSX
Auteur Jacques Boisgontier
Copyright *LIST* et l'auteur

```
10 '---------- DESSINATEUR
20 CF=15:CE=1                    couleur fond et ecriture
30 COLOR CE,CF
40 SCREEN 2
50 '
60 OPEN "GRP:" FOR OUTPUT AS #1
70 PRESET(18,135):PRINT #1,"CLAVIER MAJUSCULE"
80 PRESET(18,145):PRINT #1,"FLECHES POUR DEPLACER"
90 PRESET(18,155):PRINT #1,"V:VALIDATION 1ER POINT"
100 PRESET(18,165):PRINT #1,"T:TRACE DROITE/ C:CERCLE"
110 PRESET (18,175):PRINT #1,"A:ANNULATION TRACE / P:PEINDRE"
120 PRESET(18,185):PRINT #1,"1,2,3,.F COULEURS/ G:GOMMER"
130 X=100:Y=100                  ' Coordonnees depart
140 XA=X:YA=Y                    ' coordonnees Point Precedent
150 XB=XA:YB=YA
160 SPRITE$(1)=CHR$(191)         ' curseur
170 '--------------- curseur (sprite)
180 PUT SPRITE 1,(X,Y-1),CE,1
190 '
200 C$=INKEY$:IF C$="" THEN 200  ' test clavier
210 '       .
220 C=ASC(C$)
230 IF C=29 THEN X=X-1:GOTO 180 ' gauche
240 IF C=28 THEN X=X+1:GOTO 180 ' droite
250 IF C=31 THEN Y=Y+1:GOTO 180 ' bas
260 IF C=30 THEN Y=Y-1:GOTO 180 ' haut
270 IF C>32 THEN GOSUB 380
280 IF C$="V" THEN PSET(X,Y),CE:XA=X:YA=Y
290 IF C$="T" THEN LINE(XA,YA)-(X,Y),CE:XB=XA:YB=YA:XA=X:YA=Y
300 IF C$="C" THEN R=SQR((X-XA)^2+(Y-YA)^2):CIRCLE(XA,YA),R,CE
310 IF C$="G" THEN LINE(XA,YA)-(X,Y),CF,BF
320 IF C$="A" THEN LINE(XB,YB)-(X,Y),CF:X=XB:Y=YB:XA=X:YA=Y
330 IF C$="P" THEN PAINT (X,Y),CE
340 IF VAL(C$)<>0 THEN CE=VAL(C$)
350 IF C$="F" THEN CE=CF
360 GOTO 180
370 '--
380 PRESET(18,125):COLOR CF:PRINT #1,CHR$(200)
390 PRESET(18,125):COLOR CE:PRINT #1,C$
400 RETURN
```

On remarquera la gestion du curseur avec un « sprite ».

NOTATION POLONAISE

UN PROGRAMME TRIANGULEUX

Tout le monde sait qu'un triangle est parfaitement déterminé quand on connaît les coordonnées de ses sommets A, B, C (par rapport à des axes perpendiculaires dans un plan). On peut alors théoriquement calculer les longueurs a, b, c de ses côtés — traditionnellement a est la longueur de BC, b celle de CA et c celle de AB. On peut aussi calculer les trois hauteurs α, β, γ (c'est-à-dire les distances du sommet A au côté BC, de B à CA et de C à AB) et les mesures des angles en chacun des sommets : nous noterons $\hat{A}$, $\hat{B}$, $\hat{C}$ ces nombres avec le grade comme unité.

Le programme est donné en Langage Machine Spécialisé pour calculatrice Hewlett-Packard (c'est une 11C qui m'a servi, mais ceci serait valable pour pratiquement toutes les autres). Il est facilement adaptable à n'importe quel ordinateur de poche voisin. La traduction en Basic serait immédiate, car les formules utilisées sont très simples. On commence par calculer les côtés par des égalités du type :

$$a = \sqrt{(x_c - x_b)^2 + (y_c - y_b)^2}$$

(suite page 81)

LIST

le journal des amateurs de programmation

Si programmer un ordinateur est devenu pour vous un loisir, un plaisir... une passion, sachez que **LIST** a été créé pour vous. **LIST** vous aide à tirer davantage de votre matériel, à vous perfectionner dans la conception des programmes qui "tourneront" sur votre machine.
LIST vous initie aux langages informatiques et sélectionne les meilleurs livres pour progresser. **LIST** vous informe de l'actualité et vous fournit trucs, astuces et idées pour mieux programmer...
Pour être sûr de ne rater aucun numéro et pour recevoir **LIST** chez vous, **abonnez-vous** !

FAITES 40 F D'ECONOMIE !

LIST, LE PLAISIR DE PROGRAMMER

20 F chez votre marchand de journaux

BULLETIN D'ABONNEMENT

à retourner à **LIST**
(service Abonnements)
5, place du Colonel-Fabien
75491 Paris Cedex 10

Nom : _____
Adresse : _____
Ville : _____
Code postal : ☐☐☐☐☐ Pays : _____

Veuillez m'abonner pour 10 numéros au prix avantageux de **160 F*** au lieu de 200 F. Je fais ainsi une économie de 40 F sur le prix de vente au numéro.
Je joins mon règlement indispensable libellé à l'ordre de **LIST**.

* Belgique : **1330 FB** ; Suisse : **50 FS** ; Canada : **30 $ C** ; autres pays : **210 FF**.
Par avion : Afrique francophone : **245 FF** ; Amérique, autre Afrique, Océanie : **305 FF** ; Asie : **355 FF**.
Belgique : Soumillon, 26, av. Massenet, 1190 Bruxelles.
Versement Société Générale 2100405 835-39.
Suisse : 19, route du Grand-Mont, CH 1052, Le Mont-sur-Lausanne, versement Caisse d'Epargne et de Crédit, 10-2418 Le Mont CH 1052, compte courant n° 850 156-7.
Canada : LMPI, 9345, rue de Meaux, St Léonard (Québec), H 1R 3H3, Canada.

(suite page 81)

le demi-périmètre $p = (1/2)(a+b+c)$, l'aire S du triangle :

$S = \sqrt{p(p-a)\,(p-b)\,(p-c)}$

la hauteur $\alpha = 2S/a$, le rayon r du cercle inscrit (c'est-à-dire du cercle tangent intérieurement à chacun des côtés du triangle) égal à S/p et enfin les angles du triangle par les formules :

$$\text{tg}\,\frac{\hat{A}}{2} = \frac{r}{p-a}$$

Elles sont très faciles à transporter si l'on dispose de la fonction ATN (arc tangente) qui figure dans la plupart des Basic.

Voici le programme pour HP-11C :

```
LBL A : GRD : R/S : STO 5 : R↓ :
STO 4 : R/S : STO 7 : R↓ : STO 6 :
R/S : STO 9 : R↓ : STO 8 : RCL 6 :
− : RCL 9 : RCL 7 : − : R→ P : STO
1 : RCL 8 : RCL 4 : − : RCL 9 : RCL
5 : − : R→ P : STO 2 : RCL 6 : RCL
4 : − : RCL 7 : RCL 5 : − : R→ P
: STO 3 : RCL 2 : + : RCL 1 : +
2 : ÷ : STO 0 : RCL 1 : − : STO 7
: RCL 0 : RCL 2 : − : STO 8 : ×
RCL 0 : RCL 3 : − : STO 9 : ×
: RCL 0 : × : √x : STO I : RCL 1 :
÷ : 2 : × : STO 4 : RCL I : RCL 2
: ÷ : 2 : × : STO 5 : RCL I : RCL
3 : ÷ 2 : × : STO 6 : RCL I : RCL
0 : ÷ : STO 0 : RCL 7 : ÷ : TAN-1
: 2 : × : STO 7 : RCL 0 : RCL 8 :
÷ : TAN-1 : 2 : × : STO 8 : RCL 0
: RCL 9 : ÷ : TAN-1 : 2 : × : STO
9 : RCL 1 : R/S : RCL 2 : R/S : RCL
3 : R/S : RCL 4 : R/S : RCL 5 : R/S
: RCL 6 : R/S : RCL 7 : R/S : RCL
8 : R/S : RCL 9 : RTN.
```

Avant toute exécution, on choisit le format d'affichage par une instruction FIX n (n compris entre 2 et 9 par exemple), on lance le programme par la touche A, puis on commence à entrer les données dans l'ordre :

abscisse de A, ordonnée de A
abscisse de B, ordonnée de B
abscisse de C, ordonnée de C
(en attendant au moins une demi-seconde entre chaque entrée).

Au bout d'une quinzaine de secondes, s'affiche le côté a; par des RUN/STOP on obtient ensuite dans l'ordre les côtés b et c, puis les hauteurs α, β et γ, enfin les mesures (en grades) des angles $\hat{A}$, $\hat{B}$ et $\hat{C}$.

Si l'on veut afficher de nouveau ces nombres, on peut les trouver respectivement dans les mémoires de 1 à 9; le rayon r du cercle inscrit est, lui, en mémoire 0, et l'aire S en mémoire I. Ce programme de 120 pas occupe 11 mémoires.

André WARUSFEL

SPECTRUM ET MICRODRIVE

TOUT FICHIER A UNE FIN

DANS le n° 2 de LIST, *j'ai lu avec beaucoup d'intérêt votre article sur le microdrive du Spectrum et son catalogue. Sans doute connaissez-vous une solution au problème que je vais vous exposer.*

Pour écrire un fichier sur le microdrive, il y a l'instruction PRINT #. Si, par la suite, vous demandez de lire un élément qui n'existe pas dans un fichier, un message END OF FILE vient interrompre le programme. J'ai essayé de trouver une solution dans les variables-système, mais en vain. D'avance merci.

Gilles DESSARCE

Les microdrives vous permettent d'écrire sur cartouche des fichiers, malheureusement uniquement séquentiels. Si vous relisez un fichier par programme et que vous n'avez pas de compteur du nombre d'items inscrits sur le fichier, un désolant compte rendu END OF FILE arrêtera l'exécution lors de tentatives de lecture au-delà du dernier item.

Plusieurs solutions existent pour pallier cet inconvénient. La plus simple est sans doute de garder en mémoire ou au sein même de votre fichier un compteur du nombre d'items le composant. La mise en œuvre des autres solutions passe par la compréhension des variables-système. Regardons-y.

Pour déterminer l'adresse du canal-microdrive concerné, le programme moniteur du Spectrum utilise 3 variables-système dénommées CHANS, STRMS et CURCHL. CHANS pointe vers le début de la zone des canaux. STRMS permet de retrouver, en fonction de la voie utilisée, la valeur qu'il faut ajouter à CHANS pour obtenir l'adresse du canal courant, c'est-à-dire CURCHL.

Les deux octets de STRMS concernant une voie se situent à l'adresse

Comment éviter END OF FILE
Routine pour Spectrum
Auteur Benoît Thonnart


```
  1 REM  ---------- La routine de verification (45 octets) ----------
  2 DATA 253,126,118,135,198,22,111,38,92,94,35,86,27,42,79,92,25,229,221,225,
1,0,0,221,203,67,78,200,221,94,11,221,86,12,221,110,69,221,102,70,237,82,192,12,
201
  3
  4  REM  Implantation de la routine de verification (ici 23296)
  5 FOR n=1 TO 45: READ a: POKE 23295+n,a: NEXT n
  6
  7
  8 REM  ********** Exemple d'utilisation **********
  9
 10 REM Lecture et ecriture sans risquer le compte rendu END OF FILE
 20 OPEN # 4;"m";1;"origine"
 30 LET voie=4: POKE 23728,voie
 40
 50 IF  USR 23296 THEN  GO TO 80
 60 INPUT #4;n: PRINT n
 70 GO TO 50
 80 REM  Suite du programme
 90 CLOSE # 4
100 STOP
```

Lecture et écriture sur microdrive
Programme pour Spectrum (version Basic)
Auteur Benoît Thonnart

Le sous-programme permettant d'éviter le compte rendu END OF FILE occupe les lignes 9 000 à 9 510.

```
 10 REM Lecture et ecriture sans risquer le compte rendu END OF FILE
 20 OPEN # 4;"m";1;"origine"
 30 OPEN # 5;"m";1;"destin"
 40 LET voie=4: GO SUB 9000: REM  Verification
 50 IF a$="fini" THEN  GO TO 80
 60 INPUT #4;n: PRINT #5;n
 70 GO TO 40
 80 REM  Suite du programme
 90 CLOSE # 4: CLOSE # 5
100
200
300
400
500 STOP
9000 REM  ---------- END OF FILE ? ----------
9010 LET a$="encore"
9020 LET strms=23574+2*voie
9030 LET chans= FN p(23631)
9040 LET curchl= FN p(strms)+chans-1
9050 LET recflg= PEEK (curchl+67)
9060 IF recflg<2 THEN  RETURN : REM  Bit 1,(ix+67) -> Reset
9070 LET chbyte= FN p(curchl+11)
9080 LET reclen= FN p(curchl+69)
9090 IF chbyte=reclen THEN  LET a$="fini"
9100 RETURN
9200
9300
9400
9500 REM  ---------- Fonction DEEK (PEEK sur 2 octets) ----------
9510 DEF FN p(x)= PEEK x+256* PEEK (x+1)
```

23574 + 2 * le numéro de la voie. L'adresse moins un, pointée par ces deux octets, ajoutée à l'adresse pointée par CHANS donne l'adresse du canal courant.

L'ouverture d'un fichier, en lecture ou en écriture, crée une zone (appelée *Canal microdrive*) de 595 octets. Les 83 premiers octets de cette zone représentent un descripteur du fichier. Ils peuvent aussi être assimilés à des variables-système.

Trois de ces variables vont plus particulièrement nous intéresser, pour résoudre notre problème.

• RECFLG : le bit 1 de cet octet est à zéro si le secteur lu n'est pas le dernier du fichier ouvert en lecture ou en écriture, et il est à un si c'est le dernier secteur du fichier ;

• CHBYTE : compteur représentant le numéro de l'octet de la zone des données venant d'être lu ;

• RECLEN : nombre d'octets de données du fichier dans le secteur lu.

Muni de ces informations, on devine ce qu'il va falloir tester pour éviter un compte rendu END OF FILE lors de la lecture d'un fichier.

1. Le bit 1 de RECFLG est à 0 : on peut sans problème continuer la lecture.

2. Le bit 1 de RECFLG est à 1 : deux cas se présentent. Si le contenu de RECLEN est supérieur au contenu de CHBYTE, cela signifie qu'il y a encore des items à lire. Inversement, c'est-à-dire si le contenu de RECLEN est égal au contenu de CHBYTE, le contenu du fichier a été entièrement lu et il ne serait pas bon de poursuivre.

Un sous-programme de vérification en Basic tel que celui qui est présenté ci-contre (lignes 9 000 à 9 510) vous avisera par le contenu de la variable A$ si la lecture du fichier concerné peut se poursuivre. Le seul paramètre nécessaire au sous-programme est le numéro de la voie sur laquelle a été ouvert le fichier pour la lecture.

Pour qu'il reste le plus clair possible, ce sous-programme débutant en ligne 9 000 n'a pas été optimisé. Vous pourrez le rendre plus concis et plus rapide à l'exécution.

Un exemple d'utilisation de ce sous-programme vous est par ailleurs proposé. Le programme lit un fichier « origine » préalablement défini et écrit chaque item dans un nouveau fichier « destination », l'exécution se terminant correctement à la ligne 500.

La même fonction, mais en langage-machine, vous convaincra par sa concision et sa rapidité. La routine de la page précédente est entièrement relogeable et peut donc être implantée soit dans le tampon de l'imprimante soit au-dessus de RAMTOP.

Le seul paramètre nécessaire à la routine est le numéro de la voie sur laquelle a été ouvert le fichier. Ce nombre doit être « poké » par le Basic à l'adresse 23728 (variable-système inutilisée). La routine retourne la valeur 0 si l'on peut continuer la lecture et la valeur 1 s'il ne faut plus lire.

La bonne utilisation des variables-système permet de remédier à certaines déficiences et d'aller plus loin en programmation, notamment avec les microdrives.

Benoît THONNART

Assemblage de la routine

```
 10 ;Recherche END OF FILE
 20
 30 ; ORIGINE a preciser
 40
 50 ;No de voie POKE par le
 60 ;BASIC dans la variable
 70 ;systeme 23728
 80
 90         LD    A,(IY+118)
100
110 ;Recherche de l'adresse
120 ;du canal microdrive
130 ;concerne
140
150         ADD   A,A
160         ADD   A,#16
170         LD    L,A
180         LD    H,#5C
190         LD    E,(HL)
200         INC   HL
210         LD    D,(HL)
220         DEC   DE
230         LD    HL,(23631)
240         ADD   HL,DE
250         PUSH  HL
260         POP   IX
270         LD    BC,0
280
290 ;Si ce n'est pas le
300 ;dernier enregistrement
310 ;retourne la valeur 0
320
330         BIT   1,(IX+67)
340         RET   Z
350
360 ;DE=CHBYTE
370
380         LD    E,(IX+11)
390         LD    D,(IX+12)
400
410 ;HL=RECLEN
420
430         LD    L,(IX+69)
440         LD    H,(IX+70)
450
460 ; CHBYTE=RECLEN ?
470
480         SBC   HL,DE
490         RET   NZ
500
510 ;dernier enregistrement
520 ;lu, retourne 1 dans BC
530
540         INC   C
550         RET
```

LES JEUX ET CASSE-TÊTE INFORMATIQUES

de Thierry CHAMORET

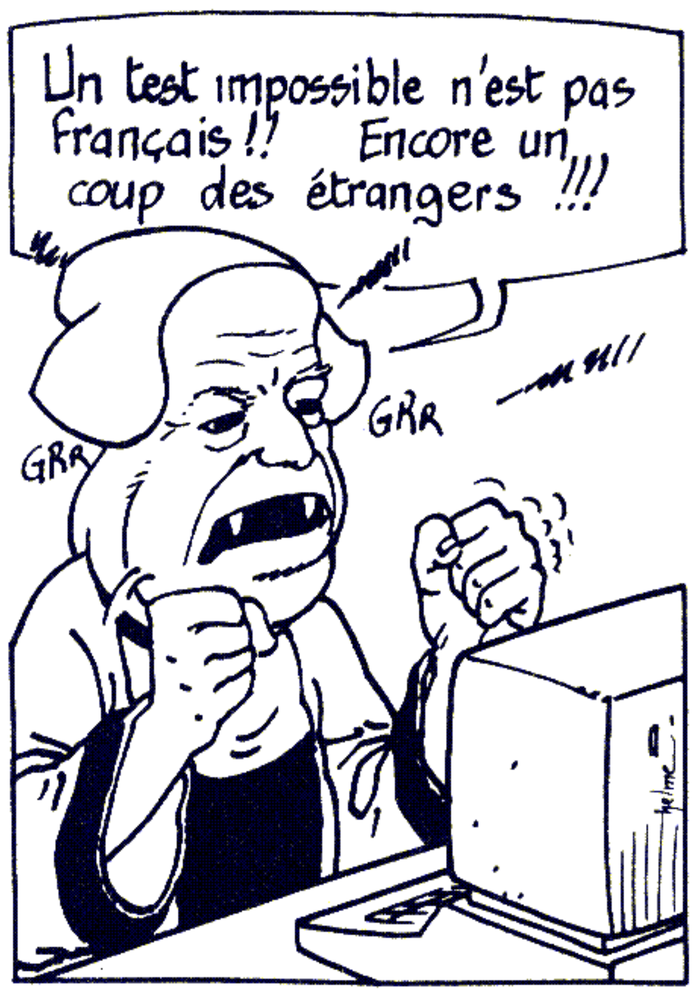
LES jeux et casse-tête qui vous sont proposés dans cette rubrique ont plusieurs aspects. Tout d'abord, ils peuvent être pris sous l'angle ludique, c'est-à-dire qu'il s'agit de jeux, de petits problèmes plus ou moins faciles à résoudre. Ils ont également un aspect pratique. Ils permettent en effet à chacun d'exercer son agilité logique. Et il n'est pas nécessaire, pour trouver la solution, d'avoir un ordinateur sous la main...

13

Un test impossible

Un test de comparaison comme celui-ci : if (A / B) < 1, ne pose apparemment aucun problème. Pourtant, si B prenait la valeur nulle ou une valeur proche de zéro, une erreur d'exécution, due à la division par zéro ou à un dépassement de capacité, se produirait. Un programmeur avisé, songeant à cela, décida de remplacer le test ci-dessus par : if A < B.

Ce test semble équivalent. Pourtant, après cette modification, le programme ne donna pas les résultats escomptés. Pouvez-vous dire pour quelle raison et comment corriger le test pour obtenir quelque chose d'équivalent ?



14

Toujours plus court

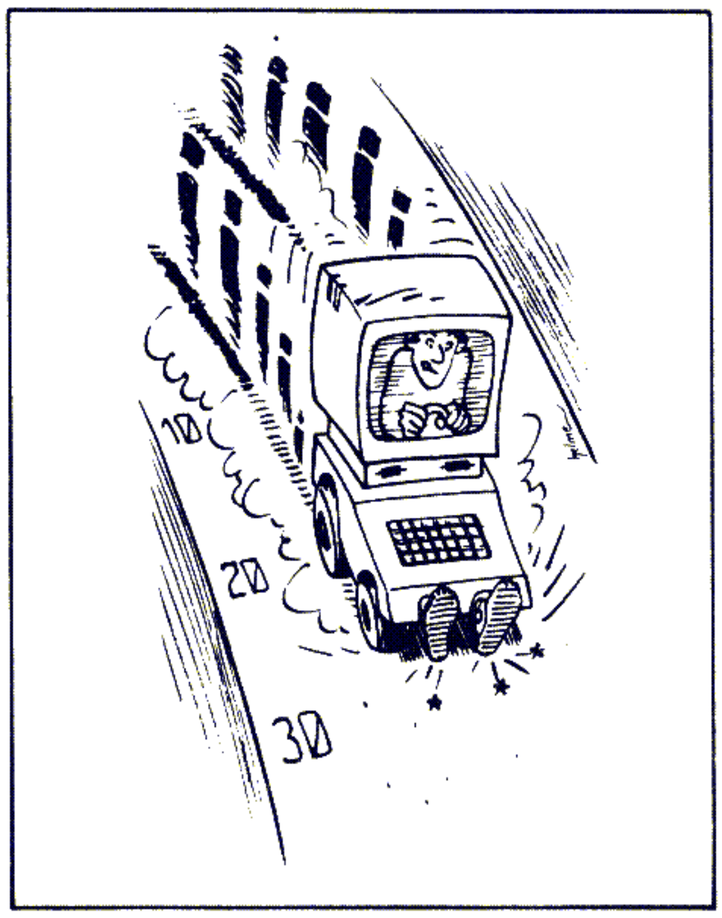
Nous avons trouvé, dans un vieux programme, la séquence d'instructions suivante :

```
10 for I = 1 to N
20 if I - 2 * int (I/2) = 0
30 then E (I) = int (I/2) + 1
40 else E (I) = int (I/2)
50 next I
```

Dans le langage qui est utilisé ici, *int* est une fonction qui retourne la valeur entière du nombre qui, ici, est passé en paramètre.

Ces instructions ont pour objet d'initialiser les N éléments du tableau E. Mais il faut dire que la technique de programmation utilisée n'est guère

explicite. Une fois que vous aurez compris ce que voulait faire le programme, il vous sera possible de le réécrire pour qu'il soit à la fois plus simple à comprendre et plus court (trois lignes sont suffisantes).



15

Une autre fonction aléatoire

La fonction RND retourne un nombre aléatoire compris entre 0 (inclus) et 1 (exclu). A l'intérieur d'une expression, elle permet d'obtenir les nombres entiers compris entre A et B, de manière aléatoire (voir la solution 12, « l'aléatoire encadré »).

Certains langages ne disposent pas de cette fonction. Par contre, ils disposent de RANDOM qui renvoie un nombre aléatoire compris entre 0 et 32767 (égal à 2 ∧ 15 − 1), tous deux inclus.

Vous trouverez certainement une ou deux expressions arithmétiques qui permettent, en une ligne et à partir d'une telle fonction RANDOM, d'obtenir un nombre pseudo-aléatoire entre A et B (tous deux inclus).

** Les solutions des jeux proposés ici paraîtront dans le prochain numéro de LIST.*

SOLUTIONS DU NUMÉRO PRÉCÉDENT

Les solutions présentées ici répondent aux problèmes posés dans le précédent numéro de LIST. Ce ne sont pas forcément les meilleures !

▸▸▸10

Gagner des microsecondes

Pour remettre à zéro un tableau T de 100 éléments, en gagnant des microsecondes, il est nécessaire d'utiliser la technique du « dépliage ». Si elle augmente la taille du programme, elle permet aussi d'en réduire le temps d'exécution. Nous pouvons ainsi écrire :

```
10 for I = 1 to 100 step 2
20   T(I) = 0
30   T(I + 1) = 0
40 next I
```

Le calcul du temps d'exécution de ce programme est le suivant :
50 fois la ligne 10 + 50 fois la ligne 20 + 50 fois la ligne 30. Soit, 42 500 + 44 000 + 55 000 = 141 500 microsecondes.

Le résultat satisfait largement à notre impératif de réduction du temps.

Par rapport à la simple boucle,

```
10 for I = 1 to 100
30 T(I) = 0
30 next I
```

qui s'exécutait en 173 000 microsecondes, la réduction du temps d'exécution du programme est de l'ordre de 18 %.

▸▸▸11

Trouver les mots

Les mots qui se rapportent aux définitions proposées sont :

1. ordinogramme (ou organigramme)
2. compilateur
3. boucle
4. processeur
5. baud
6. capteur
7. didacticiel
8. algorithme
9. mémoire
10. déboguer

▸▸▸12

L'aléatoire encadré

Une formule générale qui permet d'obtenir, à partir de la fonction RND, un nombre (entier) aléatoire compris entre A (inclus) et B (inclus), est :
INT((B − A + 1) * RND) + A
où INT(X) retourne la partie entière de X.

Sur certains Basic, la fonction RND s'écrit RND(0), RND(1), RAN, etc.

Pour simuler un jeu de dés, il suffit de prendre A = 1 et B = 6. L'expression devient :
INT(6 * RND) + 1

Imprimerie SIMA, rue des Epinettes, ZI de Torcy Sud, 77200 Torcy. Printed in France. Directeur de la publication : Jean-Luc Verhoye.
Dépôt légal novembre 1984. Diffusion NMPP. Numéro de commission paritaire : 63617

Chez Duriez : 15 micros portatifs + 9 domestiques

Imprimantes, Magnétophones, Moniteurs, Logiciels

ATARI, CANON, CASIO, COMMODORE, HEWLETT PACKARD, ORIC, SHARP, SINCLAIR, THOMSON.

Avez-vous vu les **300 prix**

Charter© Duriez ?

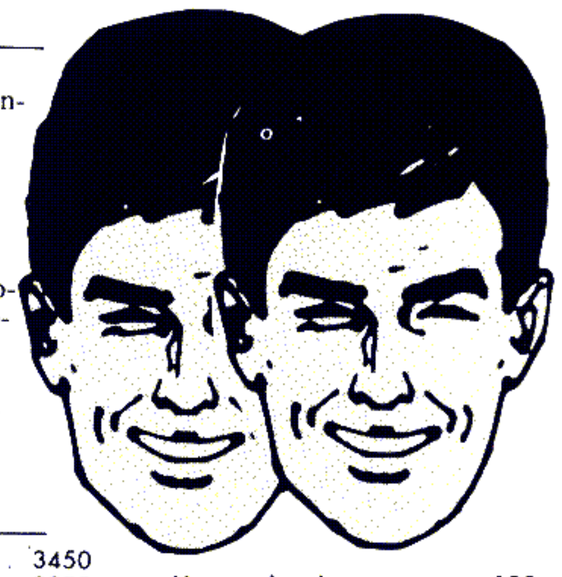
valables jusqu'au 15 Décembre

Au cœur du quartier latin, Duriez vend en magasin et par poste à prix charter.©

Il publie régulièrement bancs d'essai et Catalogues condensés de caractéristiques techniques précises, sans délayage publicitaire, complétés par des appréciations et des tests Duriez sans complaisance.

Ce banc d'essai est gratuit en magasin, ou envoyé par poste contre 3 timbres à 2,10 Frs.

POUR CHOISIR, pensez 2 fois.
1° Les performances de l'appareil ?
2° Les performances des programmes disponibles ?
Duriez fait des sélections pour vous éviter des regrets. Vous êtes tranquille.



Machines à écrire
- Photocopieurs
- Répondeurs téléphoniques
- Calculatrices
- Papeterie
- etc...

☐ Demandez le nouveau **catalogue général Duriez** contre 3 timbres à 2,10 F.

☐ **Duriez, 112 et 132 bld St-Germain 75006 Paris (M° Odéon, St-Michel)**

ATARI

600 XL Péritel	1990
800 XL Péritel	2490
Magnéto	426
Lecteur de disquette	2754
Imprimante courrier	3229
Traceur 4 couleurs	854
Manette de jeu	120
Imprimante FP10	610
Fx 750	**1550**
FA 20	**1150**
Carte 4 Ko	**600**
FP 200	2990
Extension 8K	623
Cordon magnéto	85
Traceur 4 coul.	2280
Lecteur de disquettes	4430
Clavier numérique	512
Secteur	225
Cordon impri. parallèle	390
Extension CETL (ROM)	809

AMSTRAD

CPC 464 + moniteur vert	2990
CPC 464 + moniteur couleur	4490

COMMODORE

Commodore 64 Pal	**2750**
Commodore 64 Péritel	**3450**

PERIPHERIQUES VIC20 et C64

Lecteur de cassettes	450
Lecteur de disque 1541	2250
Imprim. 50 cps MPS801	2690
Traceur 4 couleurs	1995
Interface RS232C	345
Manette de jeu	120
Crayon lumineux	475

LOGICIEL C64

Utilitaire

TOOL 64 (cart)	640
Master (disq)	950
64 Forth (cart)	588
Zoom Pascal (disq)	456
HES MON 64 (cart)	390

Professionnel

HES Writer (cart)	329
Omnicalc (cart)	329
Stat 64 (cart)	490
Graph 64 (cart)	380
Multiplan (disq)	1100
Vizawriter (disq)	1355
Super Base 64 (disq)	1190

Educatif

Turtle graphic (cart)	588
Paint brush (cart)	223
Sinthy 64 (K7)	326
Turtle Toyland (disq)	338
Coco (disq)	440

Jeux

Beach Head (K7)	150
Solo Flight (K7)	225
Summer Games (K7)	245
Harrier Attack (K7)	105

CANON

X07 mémoire 8K	1940
Traceur 4 coul. X710	1850
X07 + X710	3750
Interface video	2380
Extension 8K	750
Carte mém. 4K XM100	412
Carte mémoire 8K XM101	850
Cordon magnéto	65
Coupleur optique	470
Inter. RS232 + cordon	725
Cordons imp. parallèle	295
Secteur	82
Carte Fichiers	530
Carte Graphique	530
Carte Math/Stat	530

CASIO

PB 700	1480
Traceur 4 coul. FA10	2280
PB 700 + FA 10	3700
Extension 4KO R4	427
Magnéto intégré CM1	850
Interface FA4	865
Fx 702P	**1050**
Interface magnéto FA2	280

Ordinateurs MSX nouvelle norme Japonaise

DISPONIBLES NOVEMBRE 84
Yamaha, Sanyo, Yeno, Spectrovidéo

DISPONIBLE DÉCEMBRE 84
Canon

HEWLETT-PACKARD

HP 11C	810
HP 15C	1235
HP 12C	1235
HP 16C	1235
HP 41 CV	**2190**
HP 41 CX	2880
HP 71	**5100**
Extension mémoire 4K	**784**
Lecteur de cartes magnétiques (HP 71)	**1634**
Interface	**1318**
Lecteur de cartes (41C)	1850
Lecteur optique	1190
Imprimante 82 143	3690
Accus rechargeables	390
Chargeur	155
40 cartes magnétiques	239
Papier therm. noir (6b.)	120
Mémoire quadruple	809
Module X fonction	809
Module temps	809
Module mémoire tampon	809

PERIPHERIQUES HPIL

Module HPIL pour HP41	1348
Lecteur de cassette digit.	4770
Imprim. thermique HPIL	4770

EPSON

PX 8	**10300**
Extension mémoire 60K	**3300**
Extension mémoire 120K	**4660**
HX 20	5800
Magnéto intégré	1100
Extension 16K	1200
Modem + cordon	1755
Cassette Intext	780
Interface TV	3450
Interface moniteur	2290
10 mini cassettes digit.	990

OLIVETTI

M10 mémoire 8K	4950
M 10 mémoire 24K	**6990**
Traceur 4 coul.	2090
Secteur	98
Cordon imp. parallèle	199
Cordon imp. RS 232	498

ORIC ATMOS

Oric Atmos 48 K	**2250**
Cordon Péritel + alimentation 12 V	**95**
Traceur 4 coul. + cordon	1510
Cordon magnéto (jack)	45
Cordon imp. parallèle	150
Modulateur noir et blanc	210
Modulateur coul. SECAM	530
Lecteur de disquettes 3"	3600
disquette 3"	69

LOGICIELS EN K7

Aigle d'or (K7)	180
Categoic (K7)	95
Xenon (K7)	120
Zorgon (K7)	120
Hobit (K7)	249
Forth (K7)	180
Anglais Assimil (K7)	440
Author (K7)	187
Oric Calc (K7)	187
Poly Fichier	180

SHARP

PC 1500 A	**2065**
Traceur 4 coul. CE 150	1990
PC 1500 A + CE 150	**3990**
Extension 8K CE 155	790
Ext. 8K Protégée CE 159	1000
Ext. 16K Protégée CE 161	1700
Interf. RS232/Parallèle	1990
Cable imp. parallèle	480
Clavier sensitif	1265
PC 1251	1085
PC 1245	620
PC 1401	**1060**
PC 1260	1580
PC 1261	1990
Interface magnéto	169
Imprimante CE 126P	790
Imp. + magnéto CE 125	1695

SINCLAIR

ZX 81	580
Extension 16K	360
Spectrum 48K Péritel	2325
Spectrum 48K Pal	1965
Interface Péritel	360

TEXAS INSTRUMENTS

LOGICIEL

Jawbreaker II (cart)	190
Othello (cart)	188
Mash (cart)	190
The Attack (cart)	134
Star Trek (cart)	190
Return to Pirate I (cart)	190
Tombstone City (cart)	188
Super Demon Attack (cart)	190
TI Invaders (cart)	188
Hopper (cart)	190
Mind Challenger (cart)	134
Burger Time (cart)	190

THOMSON

MO 5	**2387**
Lecteur de K7	**598**
TO7-70	**3390**
Lecteur K7	690
Extension 64K	**1055**
Contrôleur de communic.	850
Manettes jeux et son	580
Lecteur dis. avec cont.	**3596**
Memo Basic	480
Cordon imp.	290
Interface SECAM	530

CASSETTE LOGICIELS TO7 MO5

Sauterelle	124
Mots croisés 1	188
Mots croisés 2	188
Mots en fleurs	188
Lire vite et bien	188
Un mot pour le compte	168
Multiplication casse-tête	168
Pulsar	140
Eliminator	120
Une affaire en or	155
Pilot	148

MONITEURS

Philips TP 200 (vert 12 pouces)	990
Fidelity (couleur 36 cm)	2980
Thomson (couleur 42 cm)	3430

IMPRIMANTES

Avec interface parallèle Centronics

Seikosha GP 50A	1250
Seikosha GP 500A	2500
Epsan RX 80	3440
Brother HR 5	1990
Brother 1009	2140

Avec interface RS 232C :

Brother EP 44	2690
Brother HR 5	1990
Brother 1009 (avec parallèle et RS 232)	2290

MAGNÉTOPHONES

Radiola RA 310	430
Canon X 730	690
Sanyo DR 202	690
Sanyo TRC 1550	795

Média Conseil

Je commande à Duriez : 132, Bd St-Germain, 75006 Paris.

1.1 décembre

☐ 1 Catalogue Duriez "Micros" (essais comparatifs des 20 micro-ordinateurs les plus vendus chez Duriez) contre 3 timbres à 2,10 F.

☐ 1 Catalogue général Duriez (Calculatrices, Machines à écrire, Répondeurs, Photocopieurs, Classeurs, Dicteurs, Papeterie, etc...) contre 3 timbres à 2F10.

☐ Le(s) article(s) entouré(s) sur cette page photocopiée (ou cités ci-dessous).
☐ Ci-joint chèque de .... F y compris Port et Emballage 40 F.
☐ Je paierai à réception (Contre-Remboursement) moyennant un supplément de 30 F + 40 F Port et Emballage.

Si changement de prix, je serai avisé avant expédition.
Mes Nom, Prénoms, Adresse (N°, Rue, Code, Ville) :
.....
.....
.....
Date et Signature .....
.....