

tremplin micro

**Initiation
au graphisme
de l'Apple**

**Tri
rapide**

**Sauveur
de
variables**

**Impression
en double
haute résolution
sur Epson**

L'occupation de la page zéro

**Une douzaine de routines
pratiques LM à utiliser
à partir du Basic**

N° 10 - Bimestriel - Deuxième année
3 Septembre - 3 Novembre 1986
254 FB - 11 FS - **33 F**



VOTRE BIBLIOTHÈQUE INFORMATIQUE

• LE MANUEL DE L'INTELLIGENCE ARTIFICIELLE

(A. Barr & E.A. Feigenbaum)

Vaste et captivant sujet que celui traité dans le Tome 1 de cet ouvrage collectif (traduit de l'américain par D. Tauszin-Raynaud). Si l'intelligence artificielle — l'IA pour les initiés — inquiète quelque peu certains esprits rétrogrades, elle en passionne beaucoup d'autres, heureusement plus ouverts aux techniques et théories nouvelles.

Avec un coût de l'informatique en chute libre, de nombreuses applications, hier réservées à la "grosse" informatique, deviennent possibles sur des machines plus modestes : la vôtre... par exemple.

Que l'on ne s'attende pourtant pas à trouver dans ce manuel des routines permettant de mettre au point un programme personnel. Nous sommes ici en présence d'un ouvrage de réflexion, d'une somme de connaissances résultant du travail cohérent de plusieurs dizaines de personnes : la recherche avec Anne Gardner, la représentation de la connaissance organisée par Avron Barr et James Davidson, la compréhension de la parole préparée par Lawrence Fagan, Paul Cohen et Avron Barr, etc. (Editions Eyrolles, 320 pages)

• INITIATION À LA GÉNÉRATION DE TEXTES EN LANGUE NATURELLE

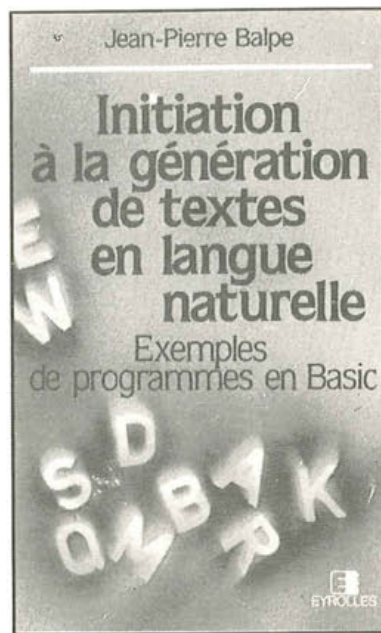
(Jean-Pierre Balpe)

Est-il possible de confier la production de textes à des ordinateurs ? La réponse vous est donnée dans cet excellent livre de Jean-Pierre Balpe... programmes en Basic à l'appui. Sur Apple ? Non, mais il s'agit d'un Basic Microsoft tout à fait conventionnel et sans aucune fonction graphique, ce qui vous permettra éventuellement de le traduire facilement en Applesoft. A noter que les programmes proposés par l'auteur ont tous été développés sur des machines dont la plus puissante n'avait pas plus de 32 Ko disponibles. Vous ne serez probablement pas surpris en apprenant qu'il est néanmoins

préférable de disposer d'un lecteur de disquette (nous préconisons l'Unidisk).

Nul doute que la lecture sérieuse (j'allais écrire "l'étude") de cette initiation autorisera bon nombre d'amateurs d'écrire des applications qui leur permettront la génération en langue naturelle de leurs propres "textes". Je ne serais pas autrement surpris si *Tremplin Micro* recevait bientôt des exemples aussi originaux qu'inédits. Pour ne rien vous cacher, je le souhaite car il s'agit là d'un sujet particulièrement intéressant.

(Editions Eyrolles, 204 pages)



• PROGRAMMATION EN ASSEMBLEUR 68000

(Louis Léon)

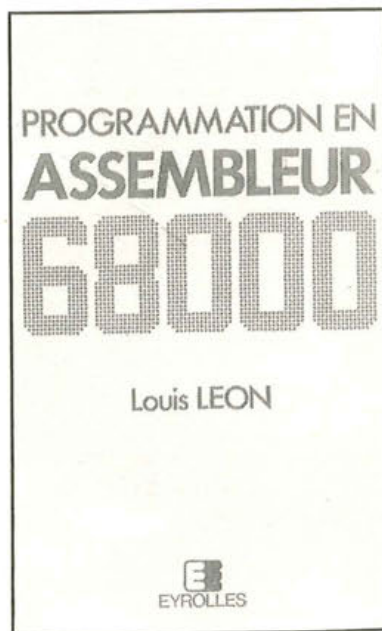
Je sais : votre Apple chéri n'est pas équipé d'un 68000, mais d'un 6502 ou encore d'un 65C02... en attendant le fameux 65C816 qui lui permettra (la nouvelle sera peut-être devenue officielle au moment où vous lirez ces lignes !) de rivaliser avec le PC d'IBM et ses innombrables copies. Mais certains de nos honorables Lecteurs (et aussi des Lectrices : je ne vous oublie

pas Mesdames) utilisent un Mac et rêvent de réécrire sur icelui les extraordinaires routines en langage machine qui ont permis à leur Apple de ne point paraître obsolète face à la redoutable concurrence des Atari, Amiga et autres machines géniales.

Il se trouve que Mac n'est plus le seul à être équipé d'un 68000, microprocesseur auquel les augures promirent un avenir beaucoup plus brillant que celui réservé au 6800 du même Motorola. Lisa prouva que le 68000 disposait réellement de possibilités extraordinaires, mais Lisa fut un échec et nous avons dû attendre l'arrivée du Mac pour que des auteurs compétents s'intéressent enfin à l'architecture et à la programmation de la bête.

Voilà qui est fait, et en français, par Louis Léon. Celui-ci vous guidera dans une approche prudente et logique du 68000 et vous prouvera que la programmation de ce microprocesseur n'est pas aussi rébarbative que vous le laisseront volontiers supposer certains beaux esprits, soucieux de vendre des machines... et des logiciels, mais ne désirant surtout pas aiguiller leurs clients vers la programmation.

(Editions Eyrolles, 380 pages)



tremplin micro

10

Apple et ProDOS (noms et logos) sont des marques déposées d'Apple Computer, Inc.

BIMESTRIEL

Le numéro : 33 F
Abonnement d'un an 190 F
(5 numéros)

EDITIONS JIBENA

Direction-Rédaction :

Editions JIBENA

Guy-HACHETTE

La Petite Motte — Senillé
63100 CHÂTELLERAULT.

Téléphone :
09-93-66-66

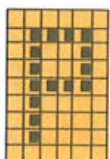
PUBLICITÉ :
Gratuite (même numéro)

Commission paritaire :
Demande en cours.

Les revues qui choisissent d'être
soumises au service du Lecteur,
ne l'obligeant pas à glaner,
dans plusieurs magazines, les
enseignements concernant sa
machine, ne bénéficient pas du
statut de Commission Paritaire,
et ne bénéficient pas des tarifs pos-
taux réduits.

TREMPLIN MICRO — Bimestriel —
est une publication des Editions
JIBENA, 4, rue de la Cour-des-
Foues, 75020 PARIS — S.A. au
capital de 3600000 F — Imprimé
par CITÉ-PRESS/PARIS — Dépôt
légal à la date de parution — Ins-
cription à la Commission Paritaire
des Publications et Agences de
Presse : en cours — Directeur de
Publication : Guy-Clément
GUY-HACHETTE — Diffusion N.M.P.P.

Au-dessus de la mêlée



POURQUOI vous obstinez-vous à éditer une revue dont le seul objectif est assurément d'aider les utilisateurs d'un Apple IIe ou IIc à mieux connaître leur machine ? C'est une question que l'on me pose souvent dans les milieux dits professionnels. Et c'est vrai que je n'ai certes pas choisi la facilité en créant Tremplin Micro, puis en poursuivant l'édition de ce titre.

Autre fait singulier : ce bimestriel de programmes ne semble guère ouvert à la publicité. D'où une autre question, au moins aussi singulière que la première : *Comment réussissez-vous à équilibrer votre budget ? Sans doute bénéficiez-vous d'une aide invisible d'Apple ?*

Est-il tellement invraisemblable, dans ce pays, qu'un directeur de publication soit suffisamment honnête pour ne point accepter de telles compromissions ? Tremplin Micro n'a rien à cacher et ne dispose pas de ressources occultes. Ses ventes lui suffisent et il peut même se permettre de rétribuer convenablement ses auteurs.

Par ailleurs, notre revue est contrôlée par l'OJD (c'est même l'un des rares titres de cette nature — publication dédiée à une seule marque — à se soumettre à une telle formalité). Jusqu'à maintenant, nous avons effectivement refusé les pages de quelques annonceurs (tout en les remerciant de leur confiance), trop peu nombreux pour justifier une augmentation de notre pagination. En effet, s'il est possible que Tremplin Micro s'ouvre un jour à la publicité (son directeur n'est pas publiphobe !), celle-ci sera regroupée dans un cahier spécial supplémentaire.

Lectrices et Lecteurs ont la presse qu'ils méritent. Il est parfois facile, dans notre pays, de réussir ce que, dans notre jargon professionnel, nous appelons des "coups". Je connais les recettes. Il est plus difficile d'imposer des publications dites "techniques et professionnelles". Heureusement, la France dispose, avec les Nouvelles Messageries de la Presse Parisienne et plusieurs dizaines de milliers de Dépositaires et Diffuseurs de Presse, du meilleur réseau de distribution du monde.

Tremplin Micro ne peut compter que sur ce réseau et sur ses Lectrices et Lecteurs. Merci aux animateurs du premier et à la fidélité des seconds... auxquels je promets de bons et nombreux moments en notre compagnie.

BONNE PROGRAMMATION ET À BIENTÔT !

GUY-HACHETTE.

SOMMAIRE 10

- Graphisme-démo (Maurice Chavelli) 3
- Instructions Basic Applesoft (Nestor) 6
- Greffe-LM (Nestor) 10

IMPRESSION EN DOUBLE HAUTE RÉOLUTION SUR EPSON

(Patrick Quettier et la collaboration d'Y. Koenig) . . . 12

- Sauveur de variables (Yvan Koenig) 23
- Tri rapide d'une liste de mots (Nestor) 25
- Draw et XDraw en langage machine (Géo) 28
- Un jeu de lettres : Treize (Géo) 30
- Essai de logiciel : Version Liste (Clément Renard) 35
- Cadre avec les caractères Souris (Nestor) 36
- Titre mystérieux (Michel Delacroix) 37
- Rideau (Clément Renard) 38
- Effets spéciaux (Nestor) 40
- Inversion d'écran DHR (Thierry Gauthier) 42
- Fenêtre de texte en HGR2 (Clément Renard) 45
- Défilement d'une ligne (G-H) 48
- Défilement de l'écran texte (G-H) 49
- Fenêtre rapide sur 40 colonnes (Argos) 50
- Minuscules transformées en capitales (Clément Renard) 52
- Affichage hexa (Nestor) 54
- Courrier des Lecteurs 56
- Biblio II, 5, 8, 34, III
- Essai de logiciel : ORCA/M 59
- Occupation de la page zéro (Marcel Cottini) 61
- Bulletin de commande et d'abonnement 63
- En souscription 64

Un message de notre ami Yvan KOENIG

Le courrier de *Tremplin Micro* montre à l'évidence que la recopie de programmes publiés en revue n'est pas chose aussi aisée qu'il y paraît au premier abord.

Les difficultés peuvent être classées en deux catégories : les fautes de frappe pures et les erreurs dues à des problèmes de lecture du listage imprimé (oublions les bogues et coquilles typographiques !).

Pour corriger ces erreurs il existe des utilitaires dont l'emploi est fortement conseillé. Pour le BASIC, POM'S diffuse une nouvelle version de son Editeur Plein Ecran (EPE) dont l'un des atouts est d'être un produit de conception française. On peut également utiliser PROCMD de CALL APPLE ou GALE de NIBBLE. Avec ces deux utilitaires on dispose de multiples fonctions dont la RENUMÉROTATION. Pour les routines en langage machine, NIBBLE a publié (et commercialise toujours) un remarquable éditeur de code HEXA sous le nom de MLE.

Il m'a pourtant semblé utile de fournir au lecteur un maximum d'éléments pour permettre une transcription plus sûre des programmes et simplifier la recherche d'éventuelles erreurs. C'est pourquoi *Tremplin Micro* (et pourquoi pas les autres revues ?) publiera désormais la plupart des listages sous une forme ENRICHIE*.

La présentation à laquelle vous êtes maintenant habitué(e) reste globalement inchangée, mais viennent s'y greffer quelques fioritures.

- Chaque ligne BASIC est accompagnée d'une SIGNATURE prenant en compte les différents octets de ladite ligne, à l'exclusion du numéro, du lien vers la ligne suivante et de tout ce qui se rattache à une REM éventuelle (REM proprement dite et : qui la précède le cas échéant). Par ailleurs, les minuscules non accentuées sont assimilées à leurs homologues majuscules.
- Les espaces parasites suivant REM ou DATA sont soigneusement filtrés. Dans une chaîne, si l'on a plusieurs espaces adjacents, les espaces de rang supérieur à un sont remplacés lors du listage par des trémas ce qui permet d'en contrôler aisément le nombre. Les caractères de contrôle — qu'il vaut mieux éviter (NIBBLE refuse carrément leur présence dans les programmes publiés) — sont affichés sous la forme développée 'ctX'. Sur écran 80 colonnes, ct s'affiche en INVERSE, CT en INVERSE remplace ct si 40 colonnes. Sur papier, sur IMW et EPSON récente ct sera souligné.
- Pour les routines HEXA les listes seront publiées à raison de 16 octets par ligne suivis là encore d'une SIGNATURE de contrôle.

Afin de vous permettre de contrôler votre travail de recopie, *Tremplin Micro* met en vente une disquette regroupant les programmes utilisés pour générer ces listages au prix de 30 francs.

* Quelques programmes du présent numéro utilisent déjà la formule de contrôle mise au point par Yvan KOENIG. Ce système se généralisera à partir de notre prochain numéro (3 novembre).

GRAPHISME.DÉMO

S I vous animez un club ou une classe de micro-informatique, n'hésitez pas à taper patiemment les 68 lignes de cette excellente démonstration des possibilités graphiques de l'Apple. Le même conseil s'adresse aux débutants et d'une manière générale, à celles et à ceux qui ne se sont pas encore intéressés aux instructions graphiques de l'Applesoft.

GRAPH.DEMO

```

1 REM -----
2 REM Un amper-interpréteur est caché là
3 REM -----
4 DATA 201,80,240,30,201,84,240,36,201,7
5 6,240,44,32,248,230,202,134,36,32,183,
6 0,240,10,32,245,230,202,138,133,37,32,
7 34,252,96,32,245
8 DATA 230,32,142,253,202,208,250,96,173
9 ,0,192,16,251,141,16,192,32,177,0,96,3
10 2,245,230,134,58,32,245,230,134,59,32,
11 245,230,138,76,99,254
12 FOR C = 768 TO 840: READ A%: POKE C,A%
13 : NEXT : POKE 1014,0: POKE 1015,3
14 POKE -16197,0: TEXT: HOME: & 11.5:
15 PRINT "Petite démonstration "
16 & 11: PRINT "-----"
17 & 20,11: PRINT "du": & 20: PRINT "--"
18 & 8,18: PRINT "Graphisme Haute Résolut
19 ion"
20 & 8: PRINT "-----"
21 ": & T: & P16: HGR
22 & 5: PRINT "L'instruction BASIC HGR af
23 fiche"
24 & 5: PRINT "et efface la PAGE 1 en lai
25 ssant"
26 & 7: PRINT "4 lignes de texte en bas..
27 ."
28 & 5: PRINT "<Pour l'assembleur: JSR $
29 F3E2>";: & T
30 & 4: PRINT "Pour couvrir la totalité d
31 e l'écran"
32 & 7: PRINT "il faut faire un POKE-1630
33 2,0"
34 & 3: PRINT "<Pour l'assembleur: BIT ou
35 LDA $C052>"
36 & 4: PRINT "Pour revenir en mixte POK
37 E-16301,0";: & T
38 HOME: TEXT: PRINT "De la meme manières
39 e l'instruction HGR2": & 11: PRINT "af
40 fiche la PAGE 2": & 5: PRINT "Celle-ci
41 n'a pas de mode mixte"
42 PRINT " (du moins pas avec la PAGE 1 d
43 u TEXTE)"
44 & 5: PRINT "<Pour l'assembleur: JSR $F
45 3D8>"
46 PRINT: PRINT " La page 1 occupe l'esp
47 ace mémoire:": PRINT: PRINT " $2000
48 / $3FF7... 8192 / 16375"
49 PRINT: PRINT " La page 2 occupe l'esp
50 ace mémoire:": PRINT: PRINT " $4000
51 / $5FF7... 16384 / 24567"
52 PRINT: PRINT "On peut afficher une PA
53 GE sans l'effacer": PRINT: PRINT " Po
54 ur la PAGE 1: POKE -16300,0 ($C054)":
55 PRINT " Pour la PAGE 2: POKE -16299,0
56 ($C055)": PRINT: & T: & P6
57 & 5: PRINT "On peut aussi écrire sur l
58 a PAGE": & 10: PRINT "qui n'est pas af
59 fichée"
60 & P2: PRINT "-> Ecriture PAGE 1: POKE2
61 30,32 ": & 6: PRINT "En assembleur: LD

```



GRAPH.DÉMO

(suite)

```
A $20": & 21: PRINT "STA $E6"
28 PRINT : PRINT "-> Ecriture PAGE 2: POK
E230,64 " : & 6: PRINT "En assembleur:
LDA $40": & 21: PRINT "STA $E6": & P5
: & T
29 HGR : & 6: PRINT "L'instruction HCOLOR
permet": & 10: PRINT "de fixer la cou
leur": & T
30 HOME : FOR C = 0 TO 7: HCOLOR= C: HPLO
T 0,0: CALL - 3082: & 15,23: PRINT "HC
OLOR = ";C:: & T: NEXT
31 GOSUB 68: & P2: PRINT "La routine HCOL
OR se situe en $F6F0": PRINT "Chargez
le code de la couleur dans X": & L240,
246,9: & P2
32 & 7: PRINT "Vous pouvez aussi charger"
: PRINT "directement $E4 pour gagner d
u temps...": & T
33 HGR : PRINT : & 8: PRINT "L'instructio
n HPLLOT permet": & 12: PRINT "de trace
r un point"
34 HPLLOT RND (1) * 280, RND (1) * 160: IF
PEEK ( - 16384) < 128 THEN 34
35 POKE - 16368,0: GOSUB 68
36 & P2: PRINT "La routine HPLLOT se situe
en $F457": & L87,244,7: & P2
37 PRINT " En $F411 se trouve la routine
HPOSN": PRINT "qui calcule à partir de
l'abscisse Y,X": & 10: PRINT "et de l
'ordonnée A:"
38 PRINT : PRINT "->L'adresse de début de
ligne ($26/$27)": PRINT "->HMASK ($30
)": PRINT "->HCOLOR1 ($1C)": PRINT "->
Le numéro d'OCTET dans la ligne ($E5)"
: & T
39 HOME : TEXT : HGR : PRINT "HPLLOT perme
t aussi de tracer des lignes": PRINT :
& 9: PRINT "HPLLOT X1,Y1 TO X2,Y2 TO..
"
40 HCOLOR= RND (1) * 8: HPLLOT RND (1) * 2
80, RND (1) * 160 TO RND (1) * 280, RN
D (1) * 160: IF PEEK ( - 16384) < 128
THEN 40
41 POKE - 16368,0: GOSUB 68
42 PRINT : & 5: PRINT "La routine HLIN es
t en $F53A": PRINT "Elle trace une lig
ne du dernier point": PRINT "traçé (do
nc qui est passé par HPOSN)": & 5: PRI
NT "au point:"
43 PRINT : PRINT : & 5: PRINT "-> ABSCISS
E X,Y": & 5: PRINT "-> ORDONNEE A": &
P2
44 & 5: PRINT "Une autre adresse bien uti
le!": PRINT "HFIND en $F5CB est l'inve
rse de HPOSN": PRINT "Il nous redonne
abscisse et ordonnée " : & P2: & 5: PRI
NT "-> ABSCISSE $E0,$E1": & 5: PRINT "
-> ORDONNEE $E2": & T
45 HOME : TEXT : HGR : HCOLOR= 7: PRINT "
HPLLOT permet aussi de tracer n'importe
": PRINT "quelle courbe à partir de so
n équation."
46 FOR X = 0 TO 14 STEP .08: HPLLOT X * 20
, SIN ( - X) * 50 + 100: NEXT : & T
47 TEXT : & P4: & 7: PRINT "Les formes ha
ute résolution": & 7: PRINT "-----
-----"
48 PRINT : & 3: PRINT "Une FORME est une
suite de vecteurs": & 5: PRINT "de dép
lacement dans les quatre " : & 5: PRINT
"directions avec ou sans traçé."
49 PRINT : PRINT " Chaque octet de la FO
RME est divisé": PRINT "en 3 sections,
chacune étant un vecteur": PRINT "Le
dernier octet ne contient que des 0."
50 PRINT : PRINT "Plusieurs formes peuen
t etre regroupées": PRINT : & 8: PRIN
T "dans une TABLE DE FORMES."
51 PRINT : PRINT " Cette TABLE contient e
n tete le nombre": PRINT " de formes p
uis leurs adresses relatives": & 6: PR
INT "La TABLE est ainsi relogeable."
52 PRINT : PRINT " Vous allez voir ce qu
'on peut faire": & 6: PRINT "avec un s
imple petit carré...": & T
53 DATA 1,0.4,0.45,54,63,36,0: FOR C =
850 TO 858: READ D%: POKE C,D%: NEXT
54 POKE 232,82: POKE 233,3: HOME : TEXT :
HGR : HCOLOR= 7: SCALE= 1: ROT= 0: PR
INT " DRAW permet de tracer une forme
en X,Y": PRINT : & 7: PRINT "Exemple:
DRAW 1 AT 100 , 50": DRAW 1 AT 100,50:
& T
55 PRINT : PRINT " SCALE donne la dimens
ion de la forme": PRINT : FOR D = 1 TO
40: & 10: PRINT "SCALE = ";D:: XDRAW
1 AT 100,50: SCALE= D: DRAW 1 AT 100,5
0: NEXT : & T: XDRAW 1 AT 100,50
56 & P2: PRINT " ROT permet de faire tou
rner la FORME": PRINT : SCALE= 24: FOR
D = 0 TO 64: & 10: PRINT "ROT = ";D::
XDRAW 1 AT 100,70: ROT= D: DRAW 1 AT
100,70: NEXT : & T: XDRAW 1 AT 100,70
```

```

57 & P2: PRINT "Vous pouvez créer ainsi d
e jolis effets": PRINT : FOR C = 1 TO
2: HCOLOR= C: FOR D = 0 TO 64: ROT= D:
DRAW 1 AT 130,70: NEXT D,C: & T
58 PRINT : PRINT " XDRAW dessine comme
DRAW mais avec": & 8: PRINT "la couleu
r complémentaire"
59 FOR C = 1 TO 2: FOR D = 0 TO 64: ROT=
D: XDRAW 1 AT 130,70: NEXT D,C: & T: G
OSUB 68
60 & P2: PRINT " XDRAW est en $F65D et DR
AW en $F601"
61 & P2: PRINT "Il faut mettre les coordo
nnées en:": PRINT : PRINT : & 5: PRINT
"-> ABSCISSE X,Y": & 5: PRINT "-> ORD
ONNEE A"
62 & P2: PRINT "Attention! Les coordonné
s sont celles": PRINT : & 8: PRINT "de
la FORME concernée.": PRINT : PRINT :
& T

```

```

63 & P8: PRINT "Et pour terminer un petit
kaléidoscope": PRINT : PRINT : & 4: P
RINT "Toujours avec le petit carré!":
& P10: & T
64 HGR2 : SCALE= 16:H = 1: FOR Y = 5 TO 1
55 STEP 50: HCOLOR= H:H = H + 1: IF H
= 3 THEN H = 5
65 FOR C = 1 TO 250: DRAW 1 AT C,Y: NEXT
C,Y
66 SCALE= RND (1) * 100: HCOLOR= RND (1)
* 7: DRAW 1 AT RND (1) * 280, RND (1)
* 160: IF PEEK ( - 16384) < 128 THEN 6
6
67 HOME : TEXT : END
68 HOME : TEXT : & 12: PRINT "Pour l'asse
mbleur": & 12: PRINT "-----
-": RETURN

```

PROGRAMME DE MAURICE CHAVELLI

MICRO.BIBLIO

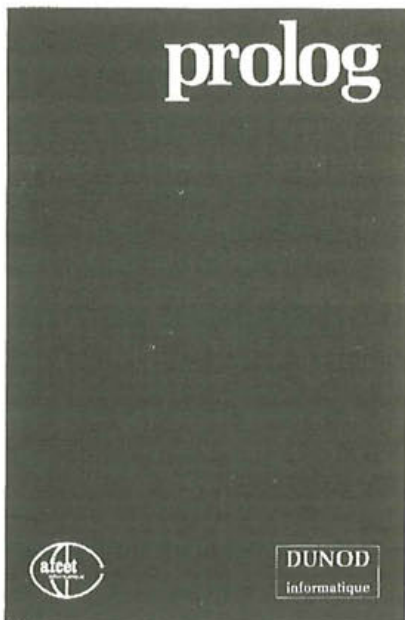
par Clément RENARD

• Prolog (Dunod)

On assiste dans les communautés scientifiques et industrielles françaises à un engouement croissant pour les langages de l'intelligence artificielle, les applications de systèmes experts et, de façon générale, les outils informatiques destinés, d'une part, à la représentation et à la manipulation de connaissances et, d'autre part, à la simulation de raisonnements.

En cinq ans, le langage PROLOG (PROgrammation LOGique) est passé d'une curiosité de laboratoire à une réalité industrielle et commerciale. Langage de programmation pour des applications d'intelligence artificielle, PROLOG est aussi un outil général pour la conception et l'exploitation de systèmes experts, pour la gestion de bases de données, le traitement des langues naturelles, la C.A.O. ou le génie logiciel. Très utilisé tant dans les laboratoires de recherche que dans les centres de développements industriels, il fait l'objet de nombreuses mises en œuvre destinées à la plupart des ordinateurs en usage aujourd'hui.

L'ouvrage collectif que présente Maryse Condillac décrit en profondeur le langage PROLOG et ses fonctions et donne un panorama de la grande variété de ses applications. Il le présente à plusieurs niveaux



d'abstraction en le replaçant dans le cadre plus général de la programmation en logique. A l'aide de nombreux exemples et applications qui feront apprécier la rapidité de sa programmation, la facilité de sa mise au point et le naturel de son expression, les auteurs expliquent ainsi les fondements théoriques et les concepts de base d'un langage en pleine évolution.

314 pages, broché (environ 170 F)

• Les meilleurs logiciels pour MOS, T07 et T09 (Marabout)

Nos lecteurs enseignants travaillent beaucoup sur Thomson... ce qui ne les empêche pas de continuer à tirer le meilleur parti de leur Apple chéri. Ils trouveront, dans ce nouveau volume de la collection *Marabout Service*, une sélection des meilleurs (du moins aux yeux de l'auteur) logiciels pour apprendre, s'amuser, rire, réfléchir, gérer, créer, dessiner, etc.

Il faut noter que la photocomposition du bouquin a été réalisée par l'auteur (Ilya Virgatchik) sur imprimante laser... une méthode d'édition qui tend à se généraliser !

Instructions Basic

APPLESOFT

V

OUS connaissez évidemment les 107 instructions du Basic Applesoft et vous savez qu'elles correspondent plus ou moins (nous y reviendrons plus loin) à des routines installées à demeure dans une ROM (mémoire morte, si vous préférez) de votre Apple. Le nom de cha-

que instruction est un mot réservé, c'est-à-dire que vous ne pouvez pas l'utiliser comme nom de variable... et pas davantage à l'intérieur d'un nom de variable (FOR=1 est interdit, mais REFORME=1 l'est aussi). Ces mots-clés du Basic Applesoft figurent tous dans une table qu'il est facile de lire à partir de l'adresse \$D0D0 (jusqu'à \$D25F). Détail amusant, cette table commence par le mot... END. Et ce n'est pas une plaisanterie ! Elle se termine par MID\$... et un beau zéro.

LE RÔLE DE L'INTERPRÉTEUR

C'est l'interpréteur qui, par l'intermédiaire de la routine PARSE (\$D56D), attribue son numéro de code à chacune de vos instructions Basic. Ce numéro ne peut être inférieur à \$80 (128) ou supérieur à \$EA (234).

Comment procède l'interpréteur ? Il lit la table caractère par caractère, non sans incrémenter son pointeur à chaque nouvelle instruction... mais il y a un hic. Consultez plutôt le début de cette fameuse table :

| | E | N | D | F | O | R | N | E |
|-------|----|----|----|----|----|----|----|----|
| D0D0- | 45 | 4E | C4 | 46 | 4F | D2 | 4E | 45 |
| D0D8- | 58 | D4 | 44 | 41 | 54 | C1 | 49 | 4E |

Vous constatez que les caractères s'y suivent sans aucune interruption : E N D F O R N E... sur quel critère l'interpréteur se base-t-il pour savoir que la lecture d'un mot-clé est terminée... et que celle d'un autre va commencer ? Je suis sûr que vous auriez trouvé la solution sans explication !

Les lettres de chaque instruction sont toutes codées à partir de la table ASCII normale (A=\$41, B=\$42, etc.), sauf la dernière qui est augmentée de \$80 (128) et devient donc : A=\$C1, B=\$C2, etc. Ce renseignement suffit à notre interpréteur pour s'y retrouver !

ADRESSES DES ROUTINES

Quand le CODE de l'instruction (ou son numéro d'ordre, si vous préférez) est connu, il est facile, pour l'interpréteur (toujours lui !), de consulter une autre table (de \$D000 à

\$D0B1) pour y lire l'adresse de la routine Applesoft correspondante, une adresse qu'il y trouve sur deux octets, comme il se doit. Mais je suis persuadé que vous avez mentalement fait un petit calcul dont le résultat vous chiffonne. Tenez ! je parie que vous vous préparez déjà à écrire à **Tremplin Micro** pour prétendre (ô le vilain... ou l'affreuse Jojette) que Nestor se moque de vous ?

Et oui : 2 octets multipliés par 107 instructions, cela nous donne effectivement 214 octets ou \$D6... or — voir plus haut — la longueur de la table n'est apparemment que de 178 octets ou \$B2... ce qui correspond à 89 instructions. Il existe effectivement 18 instructions sans point d'entrée (de \$C0 ou 192 à \$D1 ou 209).

ADRESSE FAUSSE ET ADRESSE VRAIE

Mais ce n'est pas tout. Les adresses des 64 premières routines sont fausses. Comme je vous le dis. Elles ne deviennent exactes qu'après leur avoir ajouté 1. Etrange, mais vrai. Quant aux adresses des 25 dernières, elles sont rigoureusement exactes. Nous n'allons pas nous en plaindre... et l'interpréteur pas davantage !

PASSONS AU PROGRAMME

Tout cela n'est d'ailleurs pas bien grave puisque notre petit programme va vous permettre, à partir du CODE HEXA ou DEC de chaque mot réservé (n'oubliez pas, de \$80 à \$EA) :

- d'afficher le nom de l'instruction (c'est le plus facile)
- son adresse positive... et négative (c'est parfois pratique)
- éventuellement de changer le mot-clé pour un autre de même longueur, à vos risques et périls bien entendu.

Je crois que vous voudrez tester la dernière "facilité" du programme et je vous conseille, dans un premier temps (et après avoir sauvegardé votre œuvre sur disquette) de changer REM pour ***.

Lorsque ce sera chose faite, tapez un CTRL-C pour sortir du programme sans RESET, puis un LIST qui vous prouvera que tous vos REM sont bel et bien devenus des ***, comme dans le listage de **Tremplin Micro**. Attention ! un RESET rétablit l'orthographe normale, sans aucune influence sur le programme en mémoire d'ailleurs.

Et maintenant, suivez les explications données en regard du listage. Amusez-vous bien !

NESTOR.

ATTENTION !

Tapez un REM à la place de ***

FONCTIONNEMENT :

Un simple RETURN vous donnera automatiquement la valeur suivante (la première étant \$80/128).

Pour aller en avant, taper + (plus) et ensuite des RETURN successifs.

Pour revenir en arrière, taper - (moins) et des RETURN.

Une valeur HEXA doit être précédée du \$ (dollar).

Une valeur DEC sera rentrée sans signe particulier.

Pour remplacer un mot-clé par un autre de même longueur, rentrer £MOT (ou dièse-mot avec un clavier QWERTY).

Le mot proposé doit compter le même nombre de caractères que le mot à remplacer.

```
100 TEXT : PRINT CHR$(21): HOME
105 ONERR GOTO 340: *** Fin de programme avec CTRL-C
110 GOTO 350: *** Vers routines LM
115 INVERSE : PRINT " POINTS D'ENTREE DES ROUTINES APPLESOF
T ": NORMAL : POKE 34,3
120 PRINT "La table commence à l'adresse $D000 et le nu
méro de code de chaque routine est compris entre $80 &
$EA (128-234).
125 VTAB 18: FOR I = 1 TO 10: PRINT "____";: NEXT : PRINT
130 VTAB 20: PRINT " POUR UNE MODIFICATION: ":: INVERS
E : PRINT "£NOM": NORMAL
135 VTAB 22: PRINT " RETURN POUR VALEUR SUIVANTE"
140 R = 0:PM = 1
145 VTAB 8: INVERSE : PRINT "NUMERO DEMANDE":: NORMAL : INP
UT "> ";CODE$
150 *** Il suffit de taper + ou - pour aller en AVANT ou en
ARRIERE
155 IF CO$ = "+" AND R < 107 THEN PM = 1:CO$ = "": GOTO 185
160 IF CO$ = "-" AND R >= 1 THEN PM = - 1:CO$ = "": GOTO 1
85
165 DI = 1: *** Adresse + 1 pour les 63 premiers tokens
170 IF CO$ = "" THEN 185
175 *** Si CO$ commence par £, changement de nom
180 IF ASC (CO$) = 35 AND LEN (CO$) - 2 = PEEK (24) THEN GO
TO 335
185 IF CO$ = "" THEN R = R + PM:D = R + 127: GOTO 225
190 IF LEN (CO$) < > 3 THEN 145
195 *** Si CO$ ne commence pas par un $ c'est une valeur dé
cimale (à priori)
200 IF ASC (CO$) < > 36 THEN D = VAL (CO$): GOTO 220
205 *** Conversion valeur HEXA en valeur DEC
210 D = 0:P = 0: FOR I = 3 TO 2 STEP - 1:A$ = MID$(CO$,I,1
):A = VAL (A$): IF A$ > "9" THEN A = ASC (A$) - 55
215 D = D + (A * 16 ^ P):P = P + 1: NEXT
220 R = D - 127: *** R=rang (de 1 à 107...$80-EA...128-234)
225 IF R < 1 THEN PM = 1:CO$ = "": GOTO 185: *** PM forcé à
+1
230 IF R > 107 THEN PM = - 1:CO$ = "": GOTO 185: *** PM for
cé à -1
235 VTAB 8: HTAB 17: PRINT D;: CALL - 868: HTAB 38: PRINT P
M;: IF PM = 1 THEN HTAB 37: PRINT "+"
240 *** 18 routines ne comportent pas de point d'entrée ($C
0-D1...191-209)
245 DR = D: IF D > 209 THEN DR = D - 18
250 *** Ne plus ajouter 1 pour obtenir l'adresse des 25 der
niers points d'entrée (donc DI=0)
255 IF D > 191 AND D < 210 THEN DR = 0:X = DR:Y = DR:DI = 0
: GOTO 270
260 *** Adresses des routines ($D000-$D0CF)
265 ADRESSE = 53248 + (2 * DR) - 256:X = PEEK (AD) + DI:Y =
PEEK (AD + 1): IF X > 255 THEN X = 0:Y = Y + 1
270 POKE 7,X: POKE 6,Y
275 PRINT : VTAB 11: HTAB 8
```



Routines Basic Applesoft

(suite)

```

280 *** Liste des tokens ($D0D0-$D25F ou 53456-53855)
285 POKE 25,R: CALL 776:A2 = PEEK (9) * 256 + PEEK (8):A1 =
    A2 - PEEK (24)
290 *** Affichage du nom
295 FOR J = A1 TO A2: PRINT CHR$ ( PEEK (J));: NEXT
300 *** Affichage de l'adresse décimale (+ ET -) et de l'ad-
    resse hexa
305 PRINT " ";: IF NOT DR THEN INVERSE : PRINT "PAS DE POIN-
    T D'ENTREE";: NORMAL : GOTO 320
310 AF = PEEK (AD + 1) * 256 + PEEK (AD) + DI
315 PRINT AF" "AF - 65536" $";: CALL 768
320 CALL - 868
325 PRINT : GOTO 145
330 *** Remplacement d'un mot-clé par celui de votre choix,
    mais de longueur identique
335 P = 1: FOR J = A1 TO A2 - 1:P = P + 1: POKE J, ASC ( MI-
    D$ (CO$,P,1)): NEXT : POKE A2, ASC ( RIGHT$ (CO$,1)) +
    128: GOTO 220
340 TEXT : HOME : END
345 *** Routines LM
350 FOR I = 768 TO 819: READ R: POKE I,R: NEXT
355 *** Copie de $D000-FFFF (ROM) en mémoire auxiliaire et
    lecture de cette zone commutée sur la mémoire vive (ann-
    ulé par un RESET)
360 A$ = "D000<D000.FFFFM D823G": FOR X = 1 TO LEN (A$): PO-
    KE 511 + X, ASC ( MID$ (A$,X,1)) + 128: NEXT : POKE 72,
    0: CALL - 144: POKE 49283,0: POKE 49283,0
365 *** La ligne précédente est inutile si l'on supprime le
    changement de nom
370 GOTO 115
375 DATA 165,6,166,7,32,65,249,96,162,0,134,24,169,208,133,
    8,133,9,160,0,161,8,201,128,144,14,230,24,165,24,197,25
    ,240,15,201,107,240,11,160,255,200,230,8,208,2,230,9,20
    8,227,132,24,96
    
```

ROM en RAM

La ligne 360 copie la mémoire morte en mémoire vive et commute la lecture de cette zone sur la mémoire vive, ce qui obligera l'interpréteur à lire vos éventuelles modifications... et à en tenir compte.

Comme cela a été précisé plus haut, un RESET rétablit la lecture en mémoire morte.

ADRESSES

*D000.D0B1 De la 1^{re} à la 64^e, adresse = adresse + 1 (exemple : \$D86F = \$D870... END).

```

D000- 6F D8 65 D7 F8 DC 94 D9
D008- B1 DB 30 F3 D8 DF E1 DB
D010- 8F F3 98 F3 E4 F1 DD F1
D018- D4 F1 24 F2 31 F2 40 F2
D020- D7 F3 E1 F3 E8 F6 FD F6
D028- 68 F7 6E F7 E6 F7 57 FC
D030- 20 F7 26 F7 74 F7 6C F2
D038- 6E F2 72 F2 76 F2 7F F2
D040- 4E F2 6A D9 55 F2 85 F2
D048- A5 F2 CA F2 17 F3 BB F3
D050- 9E F3 61 F2 45 DA 3D D9
D058- 11 D9 C8 D9 48 D8 F4 03
D060- 20 D9 6A D9 DB D9 6D D8
    
```

```

D068- EB D9 83 E7 C8 D8 AF D8
D070- 12 E3 7A E7 D4 DA 95 D8
D078- A4 D6 69 D6 9F DB 48 D6
    
```

De la 65^e à la 89^e, l'adresse est exacte (TOKEN \$D2 à \$EA).

```

D080- 90 EB 23 EC AF EB 0A 00
D088- DE E2 12 D4 CD DF FF E2
D090- 8D EE AE EF 41 E9 09 EF
D098- EA EF F1 EF 3A F0 9E F0
D0A0- 64 E7 D6 E6 C5 E3 07 E7
D0A8- E5 E6 46 E6 5A E6 86 E6
D0B0- 91 E6
    
```

Votre bibliothèque INFORMATIQUE

par Guy HACHETTE

• *Pratique des Apple, volume 4* (Editions Radio)*

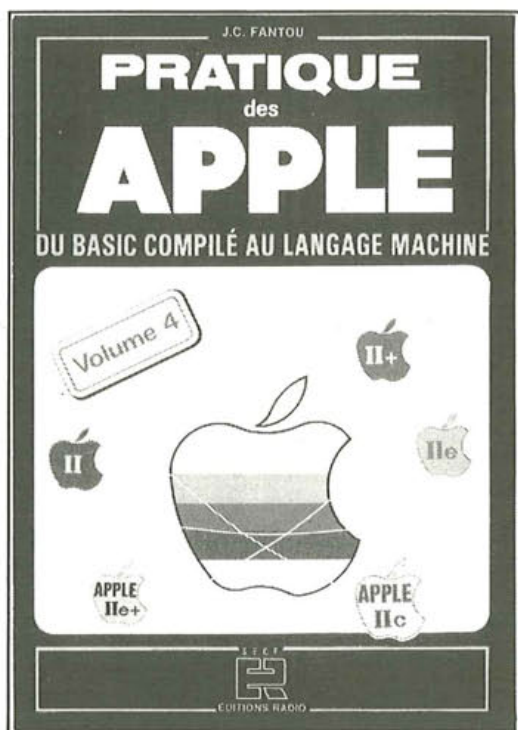
Il m'a été impossible de vous annoncer la parution de cet ouvrage dans le numéro 9 de *Tremplin Micro* et je le regrette : je suis en effet persuadé qu'il aurait rendu d'immenses services à bon nombre de nos Lectrices et Lecteurs pendant les vacances d'été.

Mais il n'est pas trop tard pour essayer de rattraper le temps perdu. Ce quatrième volet de la *Pratique des Apple* est consacré au Basic compilé et au langage machine, et je vous assure qu'il mérite de retenir votre attention. J.-C. Fantou, son auteur, sait y expliquer, très simplement, mais avec une rare maîtrise, comment court-circuiter l'interpréteur Basic pour accélérer les programmes et créer vos propres instructions Basic.

L'initiation au langage machine est progressive et très claire. Si vous êtes encore ignorant en la matière, il est certain que vous réaliserez, livre en main, de rapides progrès.

D'autres sujets retiendront également votre attention : les routines du mode graphique à double haute résolution, l'écriture en page graphique de texte 80 colonnes. Quant au chapitre consacré à l'apprentissage d'un assembleur tel que Toolkit, Lisa ou Big Mag, il est évident qu'il satisfera bien des curieux.

J'en aurai terminé quand j'aurai précisé que J.-C. Fantou — qui n'en est pas à son coup d'essai puisque nous lui devons déjà, dans la même excellente collection, *Graphisme et Son* — tient évidemment compte des instructions du 65C02. Chapeau !



* 226 pages / 210 x 295 mm / 150 F



* 448 pages / 155 x 240 mm / 240 F

• *Manuel de l'utilisateur des Apple II* (Editions Radio)*

Si je vous dis que les louanges adressées au précédent conviennent également à la méthode de Lon Poole (version française de J.-C. Fantou... toujours lui !), vous allez probablement me soupçonner d'avoir reçu d'icelui, pour prix de ma brosse à reluire, quelque ouvrage interdit de la collection X. Détrompez-vous : je ne connais pas J.-C. Fantou et pas davantage les responsables des Editions Radio, mais il se trouve que je suis emballé par cette deuxième édition.

Pour ne rien vous cacher, j'ai commencé mon apprentissage avec la première édition, il y a de cela quelques années. Elle m'avait beaucoup aidé, mais je me souviens avoir longuement séché sur certains de ses chapitres.

Les actuels acheteurs d'un Apple, et notamment d'un IIc ont beaucoup de chance. Les meilleurs livres américains ont été traduits et il existe de bons auteurs français (il suffit de lire régulièrement cette rubrique pour s'en convaincre).

Bref, je vous conseille ce *Manuel de l'utilisateur*. L'éditeur prétend que c'est une véritable "bible des Apple". Je ne le démentirai pas. D'abord parce qu'il traite — et fort bien — les points qui posent le plus de problèmes à nos Lectrices et Lecteurs : gestion des fichiers, programmation de la souris, son et musique, etc. Ensuite parce que, pour une fois, l'imprimante n'a pas été oubliée. Parfaitement : reconfiguration du port numéro un, tabulation et mise en page, micro-interrupteurs, définition de caractère à la demande... qui dit mieux ?

GREFFE-LM

C

COMMENT dissimuler une courte routine LM en tête d'un programme en Basic ? Cette question — ô combien indiscrète ! — m'a été posée par plusieurs Lectrices et Lecteurs de *Tremplin Micro*. Voici une solution simple et amusante.

Il s'agit ici de disposer d'un sous-programme capable de tracer rapidement un cadre (en 40 ou en 80 colonnes). Je me suis paresseusement contenté de reprendre celui paru dans les pages 40 et 41 du numéro 7 de *T.M.* Commencez par le taper (encadré, en bas de la page)...

*800.83F AVANT BLOAD GREFCADRE

```
0800- 00 35 08 64 00 B2 2A 2A
0808- 2A 2A 2A 2A 2A 2A 2A 2A
0810- 2A 2A 2A 2A 2A 2A 2A 2A
0818- 2A 2A 2A 2A 2A 2A 2A 2A
0820- 2A 2A 2A 2A 2A 2A 2A 2A
0828- 2A 2A 2A 2A 2A 2A 2A 2A
0830- 2A 2A 2A 2A 00 47 08 69
0838- 00 BA E7 28 34 29 22 50
```

*

*800.83F APRÈS BLOAD GREFCADRE

```
0800- 00 35 08 64 00 B2 20 58
0808- FC A5 06 A2 51 20 ED FD
0810- CA D0 FA A2 16 20 62 FC
0818- 20 10 FC A5 06 20 ED FD
0820- 20 ED FD CA D0 EF A2 4E
0828- 20 ED FD CA D0 FA 86 25
0830- 8D F7 07 60 00 47 08 69
0838- 00 BA E7 28 34 29 22 50
```

Marche à suivre : on ne peut plus simple !

- Connaître la longueur exacte de la routine LM ; dans le cas présent, celle-ci est de 46 octets (\$2E) ;
- Taper un NEW, puis le programme de la page 11, en veillant à ce que le nombre d'astérisques de la ligne 100 soit rigoureusement égal à la longueur de la routine LM ;
- Passer en mode Moniteur par un CALL-151 ;
- Afficher le début du programme par 800.83F: normalement, s'il n'y a pas d'erreur de saisie dans les lignes 100 et 105, le résultat doit être identique à l'exemple donné ci-contre. Si ce n'est pas le cas, vérifier le nombre d'astérisques ou rentrer une nouvelle ligne en supprimant les espaces ;
- Sauver cette première mouture sur disquette (on ne sait jamais !)
- Taper un BLOAD GREFCADRE, A\$806 ;
- Refaire un CALL-151, puis 800.83F... et comparer au modèle ci-contre ;
- Lister le résultat... la ligne 105 aura changé d'aspect, comme le montre mon exemple ;
- Run... et si ça ne marche pas, revérifier le tout car il y a forcément une erreur quelque part !
- Ne pas oublier de sauver cette ultime version et s'offrir le luxe de la soumettre à un copain (de préférence néophyte) qui s'interrogera longuement... et en pure perte sur le sens des étranges instructions de la ligne 100 ! Votre toujours dévoué : NESTOR

*300.32D

GREFCADRE

```
0300- 20 58 FC A5 06 A2 51 20
0308- ED FD CA D0 FA A2 16 20
0310- 62 FC 20 10 FC A5 06 20
0318- ED FD 20 ED FD CA D0 EF
0320- A2 4E 20 ED FD CA D0 FA
0328- 86 25 8D F7 07 60
```

BSAVE GREFCADRE, A\$300, L\$2E

Si vous êtes débutant(e), rentrez cette courte routine de la manière suivante :

```
10 FOR I = 768 TO 813: READ R: POKE I,R: NEXT
20 PRINT CHR$(4)"BSAVE GREFCADRE,A$300,L$2E"
30 DATA 32,88,252,165,6,162,81,32,237,253,202,2
  08,250,162,22,32,98,252,32,16,252,165,6,32,
  237,253,32,237,253,202,208,239,162,78,32,23
  7,253,202,208,250,134,37,141,247,7,96
Terminez par un RUN
```

AVANT BLOAD GREFCADRE

Naturellement, le nombre d'astérisques pourrait être supérieur à la longueur de la routine LM. L'essentiel est qu'il ne soit surtout pas plus petit !

QUESTION :

Le BLOAD GREFCADRE est-il indispensable ?

Réponse : non. Quand la routine LM est installée à l'adresse \$300, il suffit de passer en mode moniteur par un CALL-151, puis de réaliser un transfert de la manière suivante :

* 806 < 300. 32DM... et le tour est joué !

```
100 REM *****
105 PRINT CHR$(4)"PR£3": PRINT
110 VTAB 10: HTAB 3: PRINT "<1> 40 COLONNES <2> 80 COLONNES
    " : CALL - 198: GOSUB 150:N = PEEK (- 16384) - 48
115 IF N < > 1 AND N < > 2 THEN 130
120 POKE 6,160: CALL 2054: POKE 6,32: REM EFFACEMENT DU CADR
    E ANTERIEUR EVENTUEL
125 PRINT N = N - 48: POKE 2060,1 + 40 * N: POKE 2087,(40 *
    N) - 2: PRINT CHR$(16 + N): CALL 2054
130 IF N < > - 21 THEN 110
135 CALL 2054: VTAB 22: HTAB 3: PRINT "<1> MENU DE DISQUETTE
    <2> TERMINE " : CALL - 198: GET R$: PRINT : IF R$ = "1"
    THEN PRINT CHR$(4)"RUN MENU"
140 IF R$ = "2" THEN PRINT CHR$(21): HOME : END
145 GOTO 135
150 POKE - 16368,0: WAIT - 16384,128,127: POKE - 16368,0: PR
    INT : RETURN
```

LA LIGNE 100 APRÈS LE BLOAD

Vous constatez que votre Apple se débrouille, plutôt bien quand il s'agit d'afficher des inepties !

```
100 REM X ERRORS ONERR $ VTAB ) RETURN WITHOUT GOSUB h * = C
    AN'T CONTINUE VTAB $ b ERRORS $ ERRORS ONERR $ RETURN WI
    THOUT GOSUB h RETURN WITHOUT GOSUB h * = ILLEGAL QUANT
    ITY VTAB & RETURN WITHOUT GOSUB h * = CAN'T CONTINUE DI
    M % PLOT TYPE MISMATCH $`
```

AMUSEZ-VOUS UN PEU !

Si vous examinez l'extrait de la mémoire de votre Apple après le BLOAD... ou le TRANSFERT, vous constaterez que les octets \$801-802 contiennent l'adresse de la ligne suivante, en l'occurrence la ligne 105. Essayez donc de remplacer \$35 (en \$801), par \$47 (lu à l'adresse \$835), puis listez votre programme : la ligne 105 a tout bonnement disparu, ce qui ne gêne en rien le fonctionnement de la routine. On ne commencera plus par un PR£3 et c'est tout !

Moralité : à la suite d'une ligne de REM, on peut s'offrir certaines libertés !

UN DERNIER CONSEIL

Vous serez peut être tenté de supprimer le REM... mais je vous promets que votre Apple n'aimera pas cela du tout. Par contre, si vous désirez modifier le numéro de l'actuelle ligne 100, vous y êtes autorisé. Il est contenu dans l'adresse \$803 (\$64).

PROLONGEMENT...

A partir du même programme, changez la ligne 100 pour :

```
100 TEXT : HOME
101 REM + 46 astérisques
```

Ajoutez un POKE 2049,61 à la ligne 105, et + 8 un peu partout : CALL 2062 ; POKE 2068 ; POKE 2095... sans rien oublier et surtout pas de taper un POKE 2049,61.

Ensuite : BLOAD GREFCADRE, A\$80E Vous aurez ceci :

```
0800- 00 3D 08 64 00 89 3A 97
0808- 00 3D 08 65 00 B2 20 58
0810- FC A5 06 A2 29 20 ED FD
0818- CA D0 FA A2 16 20 62 FC
0820- 20 10 FC A5 06 20 ED FD
0828- 20 ED FD CA D0 EF A2 26
0830- 20 ED FD CA D0 FA 86 25
0838- 8D F7 07 60 00 4F 08 69
0840- 00 BA E7 28 34 29 22 50
0848- 52 23 33 22 3A BA 00 9A
0850- 08
```

Il n'y a plus de ligne 101, mais le programme fonctionne.

Vous trouverez la ligne 101 par LOAD et LIST, mais elle disparaîtra de nouveau après un RUN (le POKE de la ligne 105 !).

Double haute résolution sur imprimante EPSON

RAPPELS SUR LA DOUBLE RÉOLUTION :

On obtient une définition de 560 points sur 192 points. Elle occupe 8184 (= \$1FF8) octets de 8192 (= \$2000) à 16375 (= \$3FF7) simultanément en RAM Apple (colonnes 7 à 13, 21 à 27, etc.), et en RAM carte 80 colonnes (colonnes 0 à 6, 14 à 20, etc.). Il est possible d'enregistrer sur disquette un tel graphique. Un moyen très simple est de le stocker sous forme de deux fichiers binaires. Pour enregistrer un graphique sous le nom N\$ = "DESSIN", par exemple, il faut utiliser le petit programme suivant :

```
5 A$ = "300 : 38 B0 01 18 A2 20 86 3D 86 43 A2 00 86 3C
86 42 CA 86 3E A2 3F 8D 00 C0 20 11 C3 8D 01 C0 60
ND7D2G"
```

```
6 FOR I = 1 TO LEN(A$) : POKE 511 + I, ASC(MID$(A$,
I, 1)) + 128 : NEXT : POKE 72,0 : CALL -144
```

```
10 PRINT CHR$(4) "BSAVE"N$.IMP, A8192, L8184" : CALL
768 + 3 : PRINT CHR$(4) "BSAVE"N$.PAI, A8192,
L8184"
```

On obtient ainsi deux fichiers binaires appelés DESSIN.IMP (colonnes 7 à 13, 21 à 27, etc.), et DESSIN.PAI (colonnes 0 à 6, 14 à 20, etc.). De la même façon, pour le recharger en mémoire, il faut utiliser ce programme :

5 et 6 inchangées

```
20 PRINT CHR$(4) "BLOAD"N$.PAI, 8192" : CALL 768 :
PRINT CHR$(4) "BLOAD"N$.IMP, A8192"
```

Pour basculer l'affichage en mode graphique double haute résolution, il faut (le mode 80 colonnes étant déjà mis) utiliser la ligne suivante :

```
30 POKE 49246,0 : POKE 49239,0 : POKE 49234,0 : POKE
49232,0
```

Pour avoir un mode mixte (4 lignes de texte en bas), il faut remplacer 49234 par 49235.

Pour revenir au mode texte standard, il faut faire :

```
70 POKE 49233,0 : POKE 49238,0 : POKE 49247,0
```

L'Apple n'a pas la possibilité de créer ses propres graphiques (sous Basic) comme il le permet pour la haute résolution standard (HPLOT, etc.)... Toutefois, les possesseurs d'une carte Chat Mauve, pourront utiliser le logiciel ARLEQUIN (fourni avec FELINE) ou PURPLESOFT (fourni avec la carte EVE) et travailler en double haute

résolution. Il faut remarquer que ces logiciels fonctionnent aussi avec une carte 80 colonnes étendue (le DESSIN.1 de la disquette *Tremplin Micro* a été réalisé de cette façon), mais que seul ARLEQUIN fonctionne sur le IIc !

UTILISATION DE LA ROUTINE DANS UN PROGRAMME BASIC (précautions) :

Pour éviter tout conflit de mémoire entre le Basic, la page graphique haute résolution et la routine en langage machine, il est souhaitable de charger le Basic (par exemple) au-dessus de la page graphique : c'est-à-dire à partir de 16384 (= \$4000). Sous DOS 3.3, cela laisse environ 21K octets de libres pour le Basic et ses variables. On peut obtenir ce résultat de 3 façons :

- entrer au clavier **POKE 103,1 : POKE 104,64 : POKE 16384,0** puis RUNer le programme basic.

- écrire un petit programme Basic qui lancera le programme Basic principal (c'est le cas de DEMO.LANCE).

```
10 POKE 103,1 : POKE 104,64 : POKE 16384,0 : PRINT
CHR$(4) : "RUN Programme principal"
```

- placer en tête du programme principal la ligne :

```
1 IF PEEK(104) < 64 THEN POKE 103,1 : POKE
104,64 : POKE 16384,0 : PRINT CHR$(4) "RUN
Programme".
```

UTILISATION DE LA ROUTINE DANS UN PROGRAMME BASIC (mode d'emploi) :

La routine machine mise précédemment en place par les lignes 5 et 6 est ici inutile car **COPIE.HGR16.EPS** la contient. Le programme Basic devra, dès le début, contenir la ligne suivante (les numéros sont donnés à titre indicatif) :

```
10 PRINT CHR$(4) "PR#3" : PRINT : PRINT CHR$(4) "BLOAD
COPIE.HGR16.EPS"
```

L'objet de cette ligne est d'initialiser la carte 80 colonnes et de charger en mémoire la routine en langage machine. Il faut ensuite charger la page graphique à imprimer à partir d'une disquette. Voir plus haut l'exemple de la ligne Basic numéro 20. On peut ensuite l'afficher à la vidéo (ligne numéro 30). Avant de lancer l'impression, il est nécessaire de régler deux paramètres de la

routine binaire. On peut évidemment ne pas le faire et laisser les valeurs par défaut déjà présentes.

40 POKE ENTREE + 13,X (avec X compris entre 0 et 33). Cela correspond à la tabulation horizontale du dessin sur la feuille (0 = à gauche, 33 = à droite). La valeur prise par défaut est X = 0 (réglage I).

50 POKE ENTREE + 14, Y avec Y = 0 (dessin en positif) ou Y = 255 (dessin en négatif). La valeur prise par défaut est Y = 0 (réglage O).

Il faut alors lancer l'impression :

60 PRINT CHR\$(4) "PRÉ1" : PRINT : POKE 1657,80 : CALL ENTREE : PRINT CHR\$(4) "PRÉ3"

On peut ensuite remettre la page texte à l'écran (ligne numéro 70 page 12). En utilisant CALL ENTREE + 3 on exécute une copie d'écran à l'échelle deux.

EN RÉSUMÉ :

Voici, comme exemple, un (mini) programme qui utilise la routine pour imprimer un graphique appelé DESSIN en négatif et à droite de la feuille :

```
1  comme ci-dessus
10 D$ = CHR$(4) : PRINT D$"PRÉ3" : PRINT D$"BLOAD
   COPIE.HGR16.EPS"
15 N$ = "DESSIN" : ENTREE = 2048
20 PRINT D$"BLOAD"N$.PAI, A8192" : CALL ENTREE + 6
   : PRINT D$"BLOAD"N$.IMP, A8192"
30 POKE 49246,0 : POKE 49239,0 : POKE 49234,0 : POKE
   49232,0
40 POKE ENTREE + 13,33
50 POKE ENTREE + 14,255
60 PRINT D$"PRÉ1" : PRINT : POKE 1657,80 : CALL ENTREE
   : PRINT D$"PRÉ3"
70 POKE 49233,0 : POKE 49238,0 : POKE 49247,0
```

MODIFICATIONS POSSIBLES :

Après l'impression, l'imprimante se remet en interligne 1/8^e de pouce. Pour obtenir 1/6^e de pouce, il faut faire POKE ENTREE + 12, 178 avant de lancer la routine (réglage G). On peut aussi choisir de loger la routine à une autre adresse. Pour cela, il faut assembler le fichier COPIE.HGR16.EPS.S après avoir défini ORG avec la nouvelle valeur choisie pour ENTREE.

NOTE DE YVAN KOENIG :

Tremplin ne possédant pas d'imprimante EPSON, j'ai été amené à intervenir sur le programme de Patrick QUETTIER pour en contrôler le bon fonctionnement.

Aucun problème si ce n'est avec la routine de chargement qui, chargeant directement DESSIN.PAI en mémoire auxiliaire provoquait des bruits désagréables au niveau du lecteur de disquettes ce qui m'a conduit à introduire la routine de transfert MAIN ↔ AUX.

Il m'a semblé intéressant de mettre en œuvre quelques modifications annexes :

- ajout de l'option taille 2
- échange des positions des modules programme et

table d'adresses afin de permettre de coupler cette routine avec un programme reloqueur semblable à celui de PRO.TYPC (Tremplin Micro n°9) pour placer la routine entre ProDOS et ses buffers.

Pour les lecteurs ne disposant pas de la disquette Tremplin, j'ai ajouté deux programmes d'aide.

Si vous tapez directement le code objet, FAIT. ADRESSE va générer pour vous la table d'adresses, il vous suffira après avoir tapé la partie programme de faire BLOAD ADRESSE puis BSAVE COPIE.DHGR.EPS, A\$800, L\$300.

Si vous saisissez la source en assembleur c'est FAIT. T.ADRESSE qu'il vous faut. Si vous travaillez avec MERLIN, supprimez la ligne 125 puis faites RUN FAIT.T.ADRESSE. Vous pourrez charger la source de la table sous MERLIN en faisant appel à la commande R(ead) ADRESSE (ne pas taper le préfixe T.). Si vous travaillez avec un LISA (1.5, 2.5, 2.6) la ligne 105 est pour vous. Créez votre fichier T.ADRESSE par un run FAIT.T.ADRESSE. Sous LISA il vous suffira de faire CTRLD EXEC T.ADRESSE pour voir le code source se mettre en place automatiquement.

Pour les lecteurs qui souhaitent travailler avec la routine entre ProDOS et ses buffers nous publions un module reloqueur qui permet d'effectuer le déplacement de la routine en corrigeant les adresses qui doivent l'être. Ce module peut être configuré en reloqueur simple comme ici ou en reloqueur d'une commande externe comme PRO.TYPC de Tremplin n°9.

Dans le but d'éviter la publication répétitive du même module, celui-ci a été conçu de façon à rendre son adaptation à chaque cas aussi simple que possible. Après avoir tapé le code faites BSAVE RELOPRO10, A\$60D5, L\$12B. Pour utiliser avec un module commande externe, il faudrait modifier la zone 612D-613B et faire 615D: 8D 07 BE (STA EXTRNCMD+1). Par ailleurs :

60D6 contient le nombre de pages utilisées par la routine
6148 et 614C = ENDPROG-1 (octet bas/octet haut)
6150 et 6154 = ENDALL-1
619D et 61A1 = ENDADRS-1
619A vaut 90 s'il y a une table d'adresses, B0 sinon
61B5 vaut 90 s'il y a une table de données, B0 sinon
61C0/61C1 peut nécessiter une adaptation.

Le module actif nécessite quelques ajustements. Je suppose que vous avez sauvé RELOPRO10 et COPIE.DHGR.EPS. Faites BLOAD RELOPRO10 puis BLOAD COPIE.DHGR.EPS, A\$6200.

Maintenant attention, on ajuste :

Placez \$62 en 6202. 6205, 6208, 620B, 6242, 624B, 624E, 6251, 625B, 6261, 626B, 62B9, 62BF, 62E2, 62F0, 6300

Placez \$63 en 6268, 6272, 6278, 627D, 6282, 6299, 62BC, 62E7, 62FD, 6303, 6318, 631D, 632A

Placez \$64 en 629E

\$6257 : 78 63

\$625D : 7C 63

\$62F1 : 00 00 00 00 00 00 00 00

\$6378 : EA EA 26 FF 2F 62 30 63

BSAVE RELOCOPYDHGREPS, A\$60D5, L1067 qu'il faudra transférer sous ProDOS avec le FILER.

```

1
2 *****
3 *
4 * Copie double haute-résolution *
5 *
6 * Imprimante Epson parallèle *
7 *
8 * Patrick QUETTIER *
9 *
10 * 76190 Yvetot *
11 *
12 *****
13 * Assembleur MERLIN BIG-MAC * Révision 25/04/86
14 *****
15
16 *Début: 2048 $08.00
17 *Fin: 2815 $0A.FF
18 *Longueur: 768 $03.00
19 *Table: 2432 $09.80
20 *Longtable: 384 $01.80
21
22 *****
23 *Variables utilisées par le programme
24
25 * RAM MAIN = RAM carte mère
26 * RAM AUX = RAM carte 80 cols
27
28 Slot = 1
29 FLAG = 6 ;1-RAM AUX (1) - RAM MAIN (0)
30 CSW = $36 ;2-Adresse sortie caractères
31 START = $3C
32 END = $3E
33 DEST = $42
34 SAUVTAB = 237 ;1-Sauvegarde de VTAB
35 COBITVT = 239 ;1-Compteur bits de COLONNE
36 HGR = 249 ;2-Adresse en page HGR de la RAM
37 VTAB = 251 ;1-Tabulation verticale 0 à 191
38 COBITHT = 254 ;1-Compteur bits octet RAM HGR
39 COLONNE = 255 ;1-Colonne à imprimer
40
41 ENTREE = $800 ; 2048 en décimal
42
43 STOR800F = $C000
44 STORE80 = $C001 ;Utilisation pages carte 80
45 TXTPAGE1 = $C054 ;Commute RAM MAIN
46 TXTPAGE2 = $C055 ;Commute RAM AUX
47 LORES = $C056 ;Page texte
48 HIRES = $C057 ;Page HGR
49
50 CROUT = $FD8E
51
52 *****
53 ORG ENTREE
54 *****

```

Si vous n'utilisez pas un assembleur, vous n'avez rien à taper dans cette page...


```

55 *Début du programme - Appel par le basic
56
0800: 4C 30 08 57 JMP TAILLE1 ; CALL ENTREE
0803: 4C 3B 08 58 JMP TAILLE2 ; CALL ENTREE+3
0806: 4C 0F 08 59 JMP MAIN_AUX ; CALL ENTREE+6
0809: 4C 12 08 60 JMP AUX_MAIN ; CALL ENTREE+9
61
62 * Table de 3 octets pouvant rester dans
63 * la zone programme en cas d'adjonction
64 * d'un module relogeur qui croirait
65 * y trouver un BCS ... suivi d'un BRK.
66
080C: B0 67 INTERLIG DFB "0" ;Réglage POKE ENTREE+12, 176/178
080D: 00 68 MARGE DFB 0 ;Réglage POKE ENTREE+13,marge
080E: 00 69 MASQUE DFB 0 ;Réglage POKE ENTREE+14, 0/255
70
71 *****
72
080F: 38 73 MAIN_AUX SEC
0810: B0 01 74 BCS MOVER ;Toujours
75
0812: 18 76 AUX_MAIN CLC
0813: A2 20 77 MOVER LDX £>$2000
0815: 86 3D 78 STX START+1
0817: 86 43 79 STX DEST+1
0819: A2 00 80 LDX £<$2000
081B: 86 3C 81 STX START
081D: 86 42 82 STX DEST
081F: CA 83 DEX ;X=FF = £<$3FFF
0820: 86 3E 84 STX END
0822: A2 3F 85 LDX £>$3FFF
0824: 86 3F 86 STX END+1
0826: 8D 00 C0 87 STA STOR800F
0829: 20 11 C3 88 JSR $C311 ;Copie MAIN <-> AUX
082C: 8D 01 C0 89 STA STORE80
082F: 60 90 RTN RTS
91
92 *****
93
0830: A9 01 94 TAILLE1 LDA £2-1
0832: 48 95 PHA ;compteur pour modif
0833: A2 08 96 LDX £8
0835: A0 30 97 LDY £<560
0837: A9 02 98 LDA £>560
0839: D0 0E 99 BNE AJUSTE ;Toujours
100
083B: A9 03 101 TAILLE2 LDA £4-1
083D: 48 102 PHA
083E: A9 00 103 LDA £0
0840: 8D 0D 08 104 STA MARGE ;Marge nulle si double
0843: A2 04 105 LDX £4
0845: A0 60 106 LDY £<1120
0847: A9 04 107 LDA £>1120

```

(suite page 16)

EPSON (suite)

| | | | | | |
|----------------|-----|--------------------------------------|-----|------------|----------------------------------|
| | 108 | | | | |
| 0849: 8E 92 08 | 109 | AJUSTE | STX | LBL1+1 | |
| 084C: 8C 7A 08 | 110 | | STY | LBL8B+1 | |
| 084F: 8D 7F 08 | 111 | | STA | LBL8H+1 | |
| 0852: 68 | 112 | | PLA | | |
| 0853: AA | 113 | | TAX | | |
| 0854: A0 01 | 114 | | LDY | £2-1 | |
| 0856: BD F1 08 | 115 | SL | LDA | VARITBL1,X | |
| 0859: 99 AD 08 | 116 | | STA | LBL31,Y | |
| 085C: BD F5 08 | 117 | | LDA | VARITBL2,X | |
| 085F: 99 BE 08 | 118 | | STA | LBL32+1,Y | |
| 0862: CA | 119 | | DEX | | |
| 0863: 88 | 120 | | DEY | | |
| 0864: 10 F0 | 121 | | BPL | SL | |
| | 122 | | | | |
| 0866: 20 0A 09 | 123 | | JSR | INITIMP | |
| | 124 | | | | |
| | 125 | | | | |
| | 126 | *----- *Impression de la page HGR | | | |
| | 127 | | | | |
| 0869: AE 0D 08 | 128 | LBL0 | LDX | MARGE | ;Met la marge de début de ligne |
| 086C: F0 08 | 129 | | BEG | LBL8 | ;Si marge nulle |
| 086E: A9 A0 | 130 | | LDA | £" " | |
| 0870: 20 20 09 | 131 | LBL9 | JSR | PRINT2 | |
| 0873: CA | 132 | | DEX | | |
| 0874: D0 FA | 133 | | BNE | LBL9 | |
| | 134 | | | | |
| 0876: 20 26 09 | 135 | LBL8 | JSR | ESC_L | |
| 0879: A9 30 | 136 | LBL8B | LDA | £<560 | ;1120 si double (octet bas) |
| 087B: 20 2D 09 | 137 | | JSR | PRINT1 | |
| 087E: A9 02 | 138 | LBL8H | LDA | £>560 | ;1120 si double (octet haut) |
| 0880: 20 2D 09 | 139 | | JSR | PRINT1 | ;pour 560 aiguilles (1120) |
| | 140 | | | | |
| 0883: A5 FB | 141 | | LDA | VTAB | ;Sauvegarde VTAB |
| 0885: 85 ED | 142 | | STA | SAUVTAB | |
| 0887: A0 01 | 143 | | LDY | £1 | |
| 0889: 84 FE | 144 | | STY | COBITHT | ;COBITHT=1 |
| 088B: 84 06 | 145 | | STY | FLAG | ;FLAG=1 |
| 088D: 88 | 146 | | DEY | | ;Y=0 (HTAB par le registre) |
| 088E: 2C 55 C0 | 147 | | BIT | TXTPAGE2 | ;Commute RAM carte 80 |
| | 148 | | | | |
| 0891: A9 08 | 149 | LBL1 | LDA | £8 | ;COBITVT=8 (4 si mode double) |
| 0893: 85 EF | 150 | | STA | COBITVT | |
| | 151 | | | | |
| 0895: A6 FB | 152 | LBL2 | LDX | VTAB | ;HGR=adresse début ligne VTAB |
| 0897: BD 80 09 | 153 | | LDA | GBAS_HI.X | |
| 089A: 85 FA | 154 | | STA | HGR+1 | |
| 089C: BD 40 0A | 155 | | LDA | GBAS_LO.X | |
| 089F: 85 F9 | 156 | | STA | HGR | |
| | 157 | | | | |
| 08A1: B1 F9 | 158 | | LDA | (HGR),Y | ;A=octet page HGR en (HTAB,VTAB) |
| 08A3: A6 FE | 159 | | LDX | COBITHT | ;Récupère bit COBITHT de A |
| 08A5: 4A | 160 | LBL3 | LSR | A | |
| 08A6: CA | 161 | | DEX | | |

```

08A7: D0 FC      162          BNE   LBL3
08A9: 08         163          PHP
08AA: 26 FF      164          ROL   COLONNE      ;Ajoute ce bit à COLONNE
08AC: 28         165          PLP
08AD: EA EA      166  LBL31     HEX   EA,EA        ;ROL COLONNE en mode double
08AF: E6 FB      167          INC   VTAB         ;Calcule prochain VTAB (+1)
08B1: C6 EF      168          DEC   COBITVT      ;Continue pour COBITVT=8 à 1
08B3: D0 E0      169          BNE   LBL2
                                170
08B5: A5 FF      171          LDA   COLONNE      ;Envoie COLONNE à l'imprimante
08B7: 4D 0E 08   172          EOR   MASQUE       ;Réglage 0
08BA: 20 2D 09   173          JSR   PRINT1
08BD: 20 2F 08   174  LBL32     JSR   RTN          ;JSR PRINT10 en mode double
                                175
08C0: E6 FE      176          INC   COBITHT      ;COBITHT=COBITHT+1
08C2: A5 FE      177          LDA   COBITHT
08C4: C9 08      178          CMP   #8
08C6: 90 0F      179          BCC   LBL4        ; COBITHT<8 continue meme octet
                                180
                                ;Ici C=1
08C8: A9 01      181          LDA   #1          ;Cas COBITHT=8 change d'octet
08CA: 85 FE      182          STA   COBITHT     ;COBITHT=1
08CC: E5 06      183          SBC   FLAG        ; 1->0    0->1
08CE: 85 06      184          STA   FLAG
                                185
                                ;On passe en RAM MAIN pour
08D0: F0 05      186          BEQ   LBL4        ;le meme HTAB si FLAG=0
                                187
                                ;On passe en RAM AUX pour
08D2: C8         188          INY
                                ;HTAB=HTAB+1 si FLAG=1
08D3: C0 28      189          CPY   #40
08D5: F0 0C      190          BEQ   LBL5        ;Cas HTAB=40 ligne suivante
                                191
08D7: A5 ED      192  LBL4     LDA   SAUVTAB     ;Cas continue meme ligne
08D9: 85 FB      193          STA   VTAB        ;Récupère VTAB
08DB: A6 06      194          LDX   FLAG        ;Commute la page FLAG
08DD: 9D 54 C0   195          STA   TXTPAGE1,X
08E0: 4C 91 08   196          JMP   LBL1
                                197
08E3: A9 8D      198  LBL5     LDA   #13+$80    ;CR
08E5: 20 20 09   199          JSR   PRINT2      ;fin de la ligne
08E8: A5 FB      200          LDA   VTAB        ;Continue pour VTAB=0 à 191
08EA: C9 C0      201          CMP   #192
08EC: F0 0B      202          BEQ   FIN
08EE: 4C 69 08   203          JMP   LBL0
                                204
                                205 *****
                                206
08F1: EA EA      207  VARITBL1  HEX   EA,EA     ; NOP NOP en mode simple
08F3: 26 FF      208          ROL   COLONNE    ; si mode double
                                209
08F5: 2F 08      210  VARITBL2  DA    RTN      ; JSR RTN en mode simple
08F7: 30 09      211          DA    PRINT10    ; JSR PRINT10 si mode double
                                212
                                213 *****
08F7: 30 09      214  * Tout ce qui suit fonctionne avec

```

(suite page 18)

```

215 * IIE ou //e et Imprimante parallèle EPSON
216 * Il serait nécessaire de modifier
217 * pour //c et imprimante série EPSON
218 *****
219
220 *Fin - Retour au basic
221
08F9: A9 9B 222 FIN LDA £27+$80 ;Imprimante remise en interligne
08FB: 20 20 09 223 JSR PRINT2 ;standard
08FE: AD 0C 08 224 LDA INTERLIG
0901: 20 20 09 225 JSR PRINT2
0904: 2C 56 C0 226 BIT LORES
0907: 4C 8E FD 227 JMP CROUT
228
229 *****
230
231 INITIMP
232 LDA £0 ;VTAB=0
233 STA VTAB
234 STA STORE80
235 BIT HIRES
236 LDA £27+$80 ;Initialisation de l'imprimante
237 JSR PRINT2 ;Interligne m=8
238 LDA £"A"
239 JSR PRINT2
240 LDA £8+$80
241
0920: 2C 54 C0 242 PRINT2 BIT TXTPAGE1
0923: 6C 36 00 243 JMP (CSW)
244
245 *****
246
0926: A9 1B 247 ESC_L LDA £27 ;ESC
0928: 20 2D 09 248 JSR PRINT1
092B: A9 4C 249 LDA £'L' ;mode double densité
250
251 * Saut à l'imprimante EPSON parallèle
252
092D: 2C 54 C0 253 PRINT1 BIT TXTPAGE1
0930: 2C C1 C1 254 PRINT10 BIT Slot*$100+$C0C1
0933: 30 FB 255 BMI PRINT10
0935: 8D 90 C0 256 STA Slot*$10+$C080
0938: 60 257 RTS
258
259 *****
260 *
261 * Logiquement, la carte EPSON NON graphique
262 * doit fonctionner également
263 * cf: POM'S 17
264 * On laisse de la place pour les
265 * modifications pour sortie série
266 *
0939: 00 00 00 267 DS $980-*

```

093C: 00 00 00 00 00 00 00 00
 0944: 00 00 00 00 00 00 00 00
 094C: 00 00 00 00 00 00 00 00
 0954: 00 00 00 00 00 00 00 00
 095C: 00 00 00 00 00 00 00 00
 0964: 00 00 00 00 00 00 00 00
 096C: 00 00 00 00 00 00 00 00
 0974: 00 00 00 00 00 00 00 00
 097C: 00 00 00 00

SUITE
 DE LA
 PAGE 18

Et maintenant, suivant que vous utiliserez ou non un assembleur, tapez l'un de ces deux programmes d'aide (lire page 13).

FAIT.ADRESSE

```

1 REM "FAIT.ADRESSE
10 PRINT CHR$(21)           A455
20 TEXT : HOME             1C5A
30 HGR : TEXT              7854
40 VTAB 10: PRINT "PATIENCE, JE CO
   NSTRUIS LA TABLE"      037E
50 D1 = 2432:D2 = D1 + 192  1E69
60 FOR I = 0 TO 191: HPL0T 0,I 4298
70 POKE D1 + I, PEEK (39)   AFOA
80 POKE D2 + I, PEEK (38)   1DOA
90 NEXT                    0582
100 PRINT CHR$(4)"BSAVE ADRESSE,A
    $980,L384"             D14B
  
```

FAIT.T.ADRESSE

```

1 REM "FAIT.T.ADRESSE
10 PRINT CHR$(21)           A455
20 TEXT : HOME             1C5A
30 HGR : POKE - 16302,0: POKE -
   16303,0: REM "Retour en TEXT 01BA
40 VTAB 10: PRINT "PATIENCE, JE CO
   NSTRUIS LA TABLE"      037E
50 D1 = 16384:D2 = D1 + 192  65A4
60 FOR I = 0 TO 191: HPL0T 0,I 4298
70 POKE D1 + I, PEEK (39)   AFOA
80 POKE D2 + I, PEEK (38)   1DOA
90 NEXT                    0582
100 POKE 768,169: POKE 770,76: POKE
   771,218: POKE 772,253    CA06
110 C1 = 768:C2 = 769:D$ = CHR$(4
   )                          EDEC
115 PA = 0:X = D1: PRINT D$"MONCO" F684
120 PRINT D$"OPEN T.ADRESSE": PRINT
   D$"WRITE T.ADRESSE"      4815
125 PRINT "INS": REM "Pour LISA, in
   utile avec MERLIN        90E8
130 PRINT "GBAS_HI = *"     EAB2
140 FOR I = 0 TO 23: PRINT " HEX "; ED88
150 FOR J = 0 TO 7: POKE C2, PEEK (
   X):X = X + 1: CALL C1: NEXT 0E8B
160 PRINT : NEXT           6576
170 PA = PA + 1: IF PA = 1 THEN PR
   INT "GBAS_LO = *": GOTO 140 BA5E
210 PRINT D$"CLOSE"         DBDC
220 PRINT D$"NOMONCO"      4B7F
  
```

DOUBLE HAUTE RÉOLUTION SUR IMPRIMANTE EPSON continue page 20.

Offrez-vous 31 routines

Vous avez souvent besoin de routines en langage machine pour compléter l'APPLESOFT, mais votre page 3 déborde ? Ne vous énervez pas, AMPERELO est là avec ses 31 routines RELOGEABLES (plus de problème d'implantation !):

PRINT USING aux normes françaises, **tri rapide**, **Input** contrôlé, **recherche** dans un tableau, **position** d'une sous-chaîne dans une chaîne, **remplacement** d'une sous-chaîne par une autre dans une chaîne, **SWAP** de variables ou de tableaux, **garnissage** de tableaux, **DIMDEL**, **IF THEN ELSE**, **GET** numérique contrôlé, **PEEK** et **POKE** sur deux octets,

prises au point par un spécialiste :
Yvan KOENIG

ils sont tous là. Et cela sans vous obliger à étudier le langage machine.

MOSAIQUE vous permettra en outre de couper-coller les modules dont vous aurez besoin.

En prime, **LECTURE.SOURCES** vous servira à lire sans assembleur, toutes les sources MERLIN/BIG/MAC. Il vous permettra également de transférer en mode TEXT les sources binaires de MERLIN pour en autoriser le traitement sous ProCODE.

Présentées en format DOS 3.3. les routines peuvent être converties pour tourner sous ProDOS.

UTILISEZ LE BON DE COMMANDE, À LA FIN DU JOURNAL.

EPSON (suite)

LA ROUTINE DE PATRICK QUETTIER est destinée à être incorporée, pour son utilisation, dans un programme principal en Basic. Cette routine réalise une copie sur imprimante d'une page graphique double haute résolution (560 points par 192 points).

Son fonctionnement nécessite la configuration suivante :

- Appel *Ile* équipé d'une carte 80 colonnes étendue (+ 64 K) ou d'une carte "FELINE" ou "EVE" Le Chat Mauve (80 colonnes couleur + 64 K).
- Imprimante EPSON MX-80 F/T type II ou FX-80.
- Interface imprimante : carte graphique EPSON (ROM B.APL ou D.APL). Avec une carte équipée d'une ROM E.APL ou YK.APL on peut utiliser PRINT CHR(9) "80N" au lieu de POKE 1657,80 pour supprimer l'écho écran. Il semble que l'on puisse également utiliser la carte EPSON non graphique.

Remarque : les premiers Apple *Ile* (révision A) ne peuvent travailler en double haute résolution (consulter APPLE, il y avait une procédure d'échange lors de l'introduction de la révision B). Moyennant quelques modifications des commandes imprimantes, la routine fonctionnerait sur un Apple *Ic...* à condition de l'équiper d'un adaptateur série-parallèle pour l'imprimante EPSON !

Et voici **RELOPRO10** (mode d'utilisation page 13)

```
60D5: A9 03 20 F5 BE 90 4E A0 16 B9 E9 ££5EB5
60E0: 61 20 ED FD 88 10 F7 4C 00 BE 04 15 04 2A 05 14 ££4D64
60F0: 08 2A 16 15 04 2A 15 15 08 2A 04 15 04 2A 15 14 ££C757
6100: 08 28 04 15 04 2A 15 15 08 2A 15 15 00 2A 15 15 ££6151
6110: 08 2A 15 15 04 2A 05 15 08 2A 05 15 04 2A 15 14 ££3547
6120: 08 28 05 15 04 2A 15 14 08 28 85 43 85 FE A9 00 ££8AC5
6130: 85 FD F0 07 20 2C FE 60 EA EA EA A9 00 85 3C 85 ££EFD0
6140: 04 A9 62 85 3D 85 05 A9 7B 85 3E A9 63 85 3F A9 ££A0BB
6150: FF 85 06 A9 64 85 07 D8 A9 00 85 42 EA EA EA 38 ££D361
6160: E5 04 85 02 A5 43 E5 05 85 03 A0 00 B1 3C 48 29 ££93C8
6170: 03 A8 68 4A 4A AA BD EA 60 88 30 04 4A 4A D0 F9 ££2771
6180: 29 03 AA A0 00 E0 02 90 05 C8 20 C2 61 88 B1 3C ££3A6D
6190: 91 42 20 B4 FC CA 10 F6 90 D2 90 19 A9 7F 85 3E ££1469
61A0: A9 63 85 3F 20 C2 61 A2 01 B1 3C 91 42 20 B4 FC ££C846
61B0: CA 10 F6 90 EF 90 80 A5 06 85 3E A5 07 85 3F 4C ££D589
61C0: 34 61 A5 06 D1 3C C8 A5 07 F1 3C 90 1A 88 A5 04 ££4DC9
61D0: D1 3C C8 A5 05 F1 3C B0 0E 88 B1 3C 65 02 91 3C ££1613
61E0: C8 B1 3C 65 03 91 3C 88 60 8D 87 C5 C3 C1 CC D0 ££DCCB
61F0: A0 C5 C4 A0 D3 C1 D0 A0 C1 A0 D9 A7 CE A0 CC C9 ££86B1
```

Naturellement, vous ne devez pas recopier la **signature** de chaque ligne (dernière colonne).

Vous pourrez vérifier l'exactitude de votre programme en utilisant la disquette que *Tremplin Micro* met à votre disposition (lisez à ce propos le message de notre ami Yvan KOENIG, page 2 du précédent numéro).

BSAVE RELOPRO10, A\$60D5, L\$12B

VERSION MODIFIÉE (à la suite de RELOPRO10)

Si vous désirez utiliser votre routine de recopie d'écran, vous serez sans doute amené à la reloger grâce à **RELOPRO10**.

Lisez alors les explications d'Yvan KOENIG, page 13 et n'oubliez pas que cette version travaille sous **ProDOS**.

Pour vous aider voici la routine **COPIE.DHGR.EPS** avec les modifications indiquées page 13... et les signatures de contrôle.

| | | | | | | | | | | | | | | | | | |
|-------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---------|
| 6200: | 4C | 30 | 62 | 4C | 3B | 62 | 4C | 0F | 62 | 4C | 12 | 62 | B0 | 00 | 00 | 38 | EE232C |
| 6210: | B0 | 01 | 18 | A2 | 20 | 86 | 3D | 86 | 43 | A2 | 00 | 86 | 3C | 86 | 42 | CA | EE280D |
| 6220: | 86 | 3E | A2 | 3F | 86 | 3F | 8D | 00 | C0 | 20 | 11 | C3 | 8D | 01 | C0 | 60 | EE6059 |
| 6230: | A9 | 01 | 48 | A2 | 08 | A0 | 30 | A9 | 02 | D0 | 0E | A9 | 03 | 48 | A9 | 00 | EE6992 |
| 6240: | 8D | 0D | 62 | A2 | 04 | A0 | 60 | A9 | 04 | 8E | 92 | 62 | 8C | 7A | 62 | 8D | EE1DC6 |
| 6250: | 7F | 62 | 68 | AA | A0 | 01 | BD | 78 | 63 | 99 | AD | 62 | BD | 7C | 63 | 99 | EE2809 |
| 6260: | BE | 62 | CA | 88 | 10 | F0 | 20 | 0A | 63 | AE | 0D | 62 | F0 | 08 | A9 | A0 | EE085D |
| 6270: | 20 | 20 | 63 | CA | D0 | FA | 20 | 26 | 63 | A9 | 30 | 20 | 2D | 63 | A9 | 02 | EE9714 |
| 6280: | 20 | 2D | 63 | A5 | FB | 85 | ED | A0 | 01 | 84 | FE | 84 | 06 | 88 | 2C | 55 | EE9078 |
| 6290: | C0 | A9 | 08 | 85 | EF | A6 | FB | BD | 80 | 63 | 85 | FA | BD | 40 | 64 | 85 | EE078B |
| 62A0: | F9 | B1 | F9 | A6 | FE | 4A | CA | D0 | FC | 08 | 26 | FF | 28 | EA | EA | E6 | EE1236 |
| 62B0: | FB | C6 | EF | D0 | E0 | A5 | FF | 4D | 0E | 62 | 20 | 2D | 63 | 20 | 2F | 62 | EE6E22 |
| 62C0: | E6 | FE | A5 | FE | C9 | 08 | 90 | 0F | A9 | 01 | 85 | FE | E5 | 06 | 85 | 06 | EEF29A |
| 62D0: | F0 | 05 | C8 | C0 | 28 | F0 | 0C | A5 | ED | 85 | FB | A6 | 06 | 9D | 54 | C0 | EEF610 |
| 62E0: | 4C | 91 | 62 | A9 | 8D | 20 | 20 | 63 | A5 | FB | C9 | C0 | F0 | 0B | 4C | 69 | EECDF1 |
| 62F0: | 62 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | A9 | 9B | 20 | 20 | 63 | AD | 0C | EE2702 |
| 6300: | 62 | 20 | 20 | 63 | 2C | 56 | C0 | 4C | 8E | FD | A9 | 00 | 85 | FB | 8D | 01 | EE0AD5 |
| 6310: | C0 | 2C | 57 | C0 | A9 | 9B | 20 | 20 | 63 | A9 | C1 | 20 | 20 | 63 | A9 | 88 | EE0E28 |
| 6320: | 2C | 54 | C0 | 6C | 36 | 00 | A9 | 1B | 20 | 2D | 63 | A9 | 4C | 2C | 54 | C0 | EE048B |
| 6330: | 2C | C1 | C1 | 30 | FB | 8D | 90 | C0 | 60 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | EE0316 |
| 6340: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | EE0000 |
| 6350: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | EE0000 |
| 6360: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | EE0000 |
| 6370: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | EA | EA | 26 | FF | 2F | 62 | 30 | 63 | EE001D |
| 6380: | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | EE14E0 |
| 6390: | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | EE0A7F0 |
| 63A0: | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | EE2A00 |
| 63B0: | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | EE0610 |
| 63C0: | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | EE14E0 |
| 63D0: | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | EE0A7F0 |
| 63E0: | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | EE2A00 |
| 63F0: | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | EE0610 |
| 6400: | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | 20 | 24 | 28 | 2C | 30 | 34 | 38 | 3C | EE14E0 |
| 6410: | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | 21 | 25 | 29 | 2D | 31 | 35 | 39 | 3D | EE0A7F0 |
| 6420: | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | 22 | 26 | 2A | 2E | 32 | 36 | 3A | 3E | EE2A00 |
| 6430: | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | 23 | 27 | 2B | 2F | 33 | 37 | 3B | 3F | EE0610 |
| 6440: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | EE0DC00 |
| 6450: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | EE0DC00 |
| 6460: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | EE0DC00 |
| 6470: | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | 80 | EE0DC00 |
| 6480: | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | EE2580 |
| 6490: | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | EE2580 |
| 64A0: | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | EE2580 |
| 64B0: | 28 | 28 | 28 | 28 | 28 | 28 | 28 | 28 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | A8 | EE2580 |
| 64C0: | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | EE3D00 |
| 64D0: | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | EE3D00 |
| 64E0: | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | EE3D00 |
| 64F0: | 50 | 50 | 50 | 50 | 50 | 50 | 50 | 50 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | D0 | EE3D00 |



Pour sauver le tout : **BSAVE RELCOPYDHGEP5, A\$60D5, L1067**

SCRIBE

Jean-Paul ARNAUD propose cette petite démonstration aux amis qui possèdent une SCRIBE.

| | | | |
|---------------------------------|------|---------------------------------|------|
| 10 REM * SCRIBE DEMO * | | 310 PRINT | 75BA |
| 20 : | 003A | 320 PRINT CHR\$(27)"z" | A2F9 |
| 30 REM J-P ARNAUD JUIN 1986 | | 330 PRINT "BROUILLON EN-TETE" | 2AC6 |
| 40 : | 003A | 340 PRINT | 75BA |
| 50 : | 003A | 350 PRINT CHR\$(15) | 7E58 |
| 60 TEXT : HOME | 1C5A | 360 PRINT CHR\$(27)"Q" | D8F8 |
| 70 VTAB 4: HTAB 10: PRINT "*" " | | 370 PRINT "BROUILLON COMPRESSE" | 7685 |
| < SCRIBE > "DEMO "*" | 59EA | 380 PRINT | 75BA |
| 80 INVERSE : SPEED= 195 | 1E20 | 390 PRINT CHR\$(27)"y" | 88F8 |
| 90 VTAB 23: FOR I = 6 TO 2 STE | | 400 PRINT "BROUILLON COMPRESSE | |
| P - 1: HTAB 2: VTAB I: PRIN | | INDICE" | 1F51 |
| T "*": NEXT | 720A | 410 PRINT | 75BA |
| 100 VTAB 2: HTAB 3: FOR I = 1 T | | 420 PRINT CHR\$(14) | A457 |
| O 37: PRINT "*": NEXT | 4A60 | 430 PRINT "BROUILLON COMPRESSE | |
| 110 VTAB 2: FOR I = 1 TO 5: HTA | | MIXTE" | 7B2C |
| B 39: PRINT "*": NEXT | D129 | 440 PRINT | 75BA |
| 120 POKE 34,6 | 8C82 | 450 PRINT CHR\$(27)"z" | A2F9 |
| 130 VTAB 6: FOR I = 39 TO 3 STE | | 460 PRINT "BROUILLON COMPRESSE | |
| P - 1: HTAB I: PRINT "*": | | EN-TETE" | 9397 |
| NEXT | 7B3F | 470 PRINT CHR\$(27)"c" | 45E2 |
| 140 SPEED= 255: VTAB 13: HTAB 9 | | 480 PRINT CHR\$(27)"L008" | EE83 |
| : FLASH : PRINT "ctGctGctG | | 490 PRINT CHR\$(27)"m" | 51EC |
| BRANCHEZ "L'IMPRIMANTE ": N | | 500 PRINT "LETTRE" | 2ECE |
| ORMAL | 4EC1 | 510 PRINT | 75BA |
| 150 VTAB 23: GET A\$ | 5D64 | 520 PRINT CHR\$(27)"y" | 88F8 |
| 160 HOME : INVERSE : VTAB 13: H | | 530 PRINT "LETTRE INDICE" | AD9A |
| TAB 7: PRINT " PROGRAMME DE | | 540 PRINT | 75BA |
| DEMONSTRATION " | A588 | 550 PRINT CHR\$(14) | A457 |
| 170 VTAB 16: HTAB 12: PRINT " S | | 560 PRINT "LETTRE MIXTE" | 2A75 |
| UR L'IMPRIMANTE ": NORMAL | E90E | 570 PRINT | 75BA |
| 180 VTAB 22: HTAB 16: FLASH : P | | 580 PRINT CHR\$(27)"z" | A2F9 |
| RINT " <RETURN> ": NORMAL : | | 590 PRINT "LETTRE EN-TETE" | 86E0 |
| GET W\$ | 5732 | 600 PRINT | 75BA |
| 190 REM * TITRE * | | 610 PRINT CHR\$(27)"c" | 45E2 |
| 200 : | 003A | 620 PRINT CHR\$(27)"L030" | D17E |
| 210 PRINT CHR\$(13): PRINT CH | | 630 PRINT CHR\$(27)"X" | 96F7 |
| R\$(4)"PR£1": PRINT CHR\$(| | 640 PRINT "* O P T I O N S *" | 76BE |
| 27)"c": PRINT CHR\$(14): P | | 650 PRINT CHR\$(27)"L008" | EE83 |
| RINT CHR\$(27)"L011" + CHR | | 660 PRINT | 75BA |
| \$(27)"X": PRINT "IMPRIMANT | | 670 PRINT "TEXTE SOULIGNE" | 0A0E |
| E SCRIBE" | 815F | 680 PRINT | 75BA |
| 220 PRINT CHR\$(27)"c" | 45E2 | 690 PRINT CHR\$(27)"Y" | 88F8 |
| 230 PRINT CHR\$(27)"L008" | EE83 | 700 PRINT CHR\$(27)"D"; CHR\$(| |
| 240 PRINT "BROUILLON" | 39B4 | 0); CHR\$(1) | 4F2A |
| 250 PRINT | 75BA | 710 PRINT "ZEROS BARRES "(000)" | D591 |
| 260 PRINT CHR\$(27)"y" | 88F8 | 720 PRINT CHR\$(27)"X" | 96F7 |
| 270 PRINT "BROUILLON INDICE" | 4180 | 730 PRINT "SOULIGNE ET ZEROS BA | |
| 280 PRINT | 75BA | RRES "(000)" | BBD0 |
| 290 PRINT CHR\$(14) | A457 | 740 PRINT CHR\$(27)"c" | 45E2 |
| 300 PRINT "BROUILLON MIXTE" | 7B5B | 750 PRINT CHR\$(4)"PR£0" | 815F |

Sauveur de variables

JE suis persuadé qu'un jour ou l'autre vous avez regretté de ne pouvoir sauver un tableau de chaînes sur disque comme il est possible de le faire pour un tableau numérique sous DOS 3.3.

Je vous propose un palliatif écrit en partant du programme CHAIN de la disquette master DOS 3.3.

Vous savez tous que le stockage des variables se fait en divers emplacements mémoire sous APPLESOFT. Les variables numériques sont les plus faciles à appréhender puisqu'elles sont stockées en une seule fois, entre (LOMEM) et (ARYTAB) pour des variables simples, entre (ARYTAB) et (STREND) pour des tableaux.

Pour les variables chaînes, c'est plus varié. APPLESOFT stocke un descripteur dans les zones définies ci-dessus tandis que les chaînes proprement dites peuvent être stockées en partie haute de la mémoire disponible c'est-à-dire entre (FREETOP) et (HIMEM) mais si elles sont définies par

CH\$ = "xxxxxxxx", le stockage reste dans la zone programme à l'endroit où la chaîne est définie.

Ajoutons que lorsqu'une chaîne change de contenu l'ancienne valeur est purement et simplement abandonnée sur place, le descripteur pointant désormais sur la nouvelle chaîne. La routine GARBAG que tout utilisateur APPLE a maudit un jour ou l'autre a pour fonction de faire le ménage de la mémoire en éliminant les chaînes mortes. Cette routine est hélas très lente mais l'accélérer n'est pas au programme aujourd'hui.

Le programme CHAIN qui permet de chaîner deux programmes effectue un ménage puis collecte toutes les chaînes "programme" pour les reporter dans la zone de stockage normal des chaînes. Ensuite, le bloc descripteurs est déplacé de façon à se retrouver juste sous les chaînes. ATTENTION, les fonctions FN sont maltraitées

(suite au verso).

TEST1.SAUVEUR1

| | | | |
|--|------|---|------|
| 10 TEXT : HOME | 1C5A | 140 KB = 49152:KS = 49168 | B916 |
| 20 D\$ = CHR\$(4) | B3A4 | 150 REM Collecte les chaînes actives | |
| 30 A\$ = "APPLE": PRINT A\$ | B444 | 160 REM Regroupe descripteurs et chaînes | |
| 40 B\$ = "TREMLIN ": PRINT B\$ | A95F | 170 PRINT D\$"BRUN SAUVEUR1" | 6A19 |
| 50 C\$ = "SOUPLE": PRINT C\$ | 31AE | 180 PRINT D\$"MON C" | F0B3 |
| 60 BC\$ = B\$ + C\$: PRINT BC\$ | 75AB | 190 REM Sauve le fichier VARIABLES | |
| 70 DIM C\$(15,2): FOR I = 0 TO 15 | 8D2D | ES | |
| 80 C\$ = STR\$(I):C\$(I,0) = "Z" + C\$:C\$(I,1) = "U" + C\$:C\$(I,2) = "D" + C\$ | E639 | 200 PRINT D\$"BSAVE VARIABLES1 ,A" PEEK(4) + 256 * PEEK(5) "-" ,L" PEEK(8) + 256 * PE | |
| 90 FOR J = 0 TO 2: HTAB 1 + J * 15: PRINT C\$(I,J): NEXT J | 9613 | EK(9) | DF6E |
| 100 PRINT | 75BA | 210 REM Sauve le fichier OFFSET | |
| 110 C\$ = "MICRO": PRINT C\$ | ED50 | 220 PRINT D\$"BSAVE OFFSET1 ,A0 ,L8" | B89C |
| 120 BC\$ = B\$ + C\$: PRINT BC\$ | 75AB | 230 PRINT D\$"NOMON C" | 6450 |
| 130 DIM NM(12): FOR I = 1 TO 12: NM(I) = I: NEXT | CF42 | 240 PRINT : PRINT "FAITES MAINTENANT 'RUN TEST2.SAUVEUR1'" | E73C |

puisque si leur descripteur est bien sauvé, leur définition reste dans la zone programme où vous ne la retrouverez donc pas tout à l'heure.

Lorsque ces opérations sont terminées, CHAIN charge le PROG2 puis rétablit les pointeurs indispensables, descend le bloc descripteur et HOP, le tour est joué, on exécute PROG2 avec les variables définies dans PROG1. En supprimant la fonction LOAD PROG2 et en ajoutant quelques opérations de stockage de pointeurs il était possible de faire la même chose mais, sans être obligé d'utiliser PROG2 juste après PROG1.

Après utilisation de PROG1 on sauve le bloc variables sous forme d'un fichier binaire que l'on

pourra rappeler plus tard lorsque l'on en aura besoin. Il vous sera désormais possible de garnir vos tableaux stables une fois pour toutes, en accès clavier ou par lecture LENTE de fichier(s) TEXT et de sauver tout cela en fichier binaire.

Lorsque vous aurez besoin d'utiliser ces tableaux, il vous suffira de faire un BLOAD VARIABLES or vous savez qu'un bload est rapide (et peut être accéléré). La production littéraire n'étant pas mon fort, je vous renvoie aux commentaires du source pour le fonctionnement de la routine SAUVEUR et aux programmes TEST pour l'utilisation mais, une fois encore, ATTENTION à ne pas transférer de fonction.

Yvan KOENIG

TEST2.SAUVEUR1

| | | | |
|---|------|--|------|
| 10 TEXT : HOME | 1C5A | 110 CALL 926 | 3C2D |
| 20 PRINT CHR\$(4)"MON C" | 20B7 | 120 PRINT A\$: PRINT B\$: PRINT C\$: PRINT BC\$: FOR I = 1 TO 12 : PRINT NM(I): NEXT | 897F |
| 30 REM Charge le fichier VARIAB LES | | 130 PRINT : PRINT "UNE TOUCHE PO UR CONTINUER" | 48FF |
| 40 PRINT CHR\$(4)"BLOAD VARIAB LES1" | 07B6 | 140 POKE KS,0: WAIT KB,128: POKE KS,0: PRINT | FDD7 |
| 50 REM Charge le fichier OFFSET | | 150 FOR I = 0 TO 15: FOR J = 0 T O 2: HTAB 1 + 15 * J: PRINT C\$(I,J): NEXT : PRINT : NEX T : PRINT | D732 |
| 60 PRINT CHR\$(4)"BLOAD OFFSET 1" | 12E4 | 160 REM A\$,B\$,C\$,D\$,BC\$,KB,KS,C\$ (.),NM(.) | |
| 70 REM Place les descripteurs | | 170 REM SONT BIEN RECUPERES. | |
| 80 REM Ajuste les pointeurs | | | |
| 90 PRINT CHR\$(4)"BLOAD SAUVEU R1" | 2B48 | | |
| 100 PRINT CHR\$(4)"NOMON C": PR INT | C448 | | |

```

0289: 20 84 E4 A9 07 85 8F ££1B4C
0290: A5 69 A6 6A 85 9D 86 9E E4 6C D0 04 C5 6B F0 05 ££A2AD
02A0: 20 FC 02 F0 F3 85 9F 86 A0 A9 03 85 8F A5 9F A6 ££03F5
02B0: A0 E4 6E D0 07 C5 6D D0 03 4C 56 03 85 9D 86 9E ££1AB9
02C0: A0 00 B1 9D AA C8 B1 9D 08 C8 B1 9D 65 9F 85 9F ££01F4
02D0: C8 B1 9D 65 A0 85 A0 28 10 D3 8A 30 D0 C8 B1 9D ££20EB
02E0: A0 00 0A 69 05 65 9D 85 9D 90 02 E6 9E A6 9E E4 ££667A
02F0: A0 D0 04 C5 9F F0 BA 20 06 03 F0 F3 B1 9D 30 46 ££E252
0300: C8 B1 9D 10 41 C8 B1 9D F0 3C C8 B1 9D AA C8 B1 ££93E2
0310: 9D 85 9C 86 9B C5 B0 F0 02 B0 2B 88 88 B1 9D 48 ££0FC7
0320: 38 A5 6F 85 94 F1 9D C8 91 9D 85 6F C8 A5 70 85 ££773F
0330: 95 E9 00 91 9D 85 70 68 18 65 9B 85 96 A5 9C 69 ££2EE6
0340: 00 85 97 20 9A D3 A5 8F 18 65 9D 85 9D 90 02 E6 ££A791
0350: 9E A6 9E A0 00 60 A2 00 38 B5 6B E5 69 95 00 E8 ££40A7
0360: B5 6B E5 6A 95 00 E8 E0 03 90 ED A2 01 B5 69 95 ££C0A2
0370: 9B B5 6D 95 96 B5 6F 95 06 95 94 CA F0 EF 20 9A ££B133
0380: D3 A5 94 85 04 A6 95 E8 86 05 38 A5 73 E5 04 AA ££7026
0390: A5 74 E5 05 A8 E8 86 08 D0 01 C8 84 09 60 A5 06 ££E152
03A0: 85 3E 85 6F A5 07 85 3F 85 70 A2 00 18 A5 69 85 ££FC69
03B0: 42 75 00 95 6B E8 A5 6A 85 43 75 00 95 6B E8 E0 ££3AB3
03C0: 03 90 EA A5 04 35 3C A5 05 85 3D A0 00 4C 2C FE ££A269

```

SAUVEUR1

Quand vous aurez terminé, n'oubliez pas de sauver votre routine sur disquette par un **BSAVE SAUVEUR1, A\$289, L327**

PROGRAMME
SOURCE sur
la disquette
TM 11...

TRI RAPIDE

d'une liste de mots

Q

UI n'a jamais eu besoin, dans un programme en Basic, de trier rapidement une liste de mots ? Les routines classiques, non compilées, se révèlent d'une lenteur désespérante dès que le nombre de données dépasse la cinquantaine.

TRI.1 (156 octets) vous propose une solution simple et efficace. Cette routine en langage machine trie tous les éléments d'un tableau dimensionné en quelques secondes (moins de 10 pour 150 données). Elle est implantée à partir de l'adresse \$300 (768) et utilise quelques octets de la page zéro (vérifier que ceux-ci ne sont pas déjà monopolisés par le programme général... ou par d'autres sous-programmes en langage machine).

MODE D'EMPLOI : Pour trier les éléments d'un tableau A\$(, il suffit d'insérer dans votre programme en Basic une ligne du type :
CALL 768, A\$(0)

Les éventuelles variables de longueur 0 seront automatiquement classées en tête de liste.

PEUT-ON TRIER À PARTIR DU CARACTÈRE D'UN RANG DÉTERMINÉ ?

Oui. Inclure un **POKE 867, R - 1** (R=rang) avant la ligne ci-contre. Ainsi, **POKE 867,1** triera la liste sans tenir compte du premier caractère.

COMMENT CLASSER À PARTIR DE LA DONNÉE LA PLUS GRANDE ? (ordre alphabétique inverse)

Par quelques pokes bien ajustés : **POKE 869,7** ; **POKE 871,252** ; **POKE 887,144**. Ne pas oublier de rétablir les bonnes valeurs pour réaliser un tri normal.

TRI-DEMO : Cette démonstration vous montre comment utiliser TRI.1. Il n'est pas indispensable de la taper. Vous pouvez aussi vous limiter aux lignes 100-105, puis 210-240.

TRI-DEMO

```

100 IF PEEK (769) < > 190 THEN PRINT CHR
    $ (4)"BLOAD TRI.1"
105 PRINT CHR$ (4)"PR£3": PRINT : HOME
110 VTAB 12: PRINT "La machine va succes
    sivement trier 150 faux mots rangés
    dans trois tableaux: - A$(, B$(,
    C$(... puis elle vous invitera à fou
    rnrir le nombre de mots que vous alle
    z vous-meme taper dans le désordre."
115 PRINT : PRINT "La cloche vous indiqu
    era le début de chaque tri (quelques
    secondes)."
```

```
125 D = 0
```

```
130 N = 150
```

```
135 DIM A$(N),B$(N),C$(N)
```

```
140 FOR I = D TO N:A$(I) = CHR$ (65 + IN
    T ( RND (1) * 26)) + "000": NEXT
```

```
145 FOR I = D TO N:B$(I) = CHR$ (65 + IN
```

```

    T ( RND (1) * 26)) + "100": NEXT
150 FOR I = D TO N:C$(I) = CHR$ (65 + IN
    T ( RND (1) * 26)) + "200": NEXT
155 CALL - 198
160 PRINT : HOME :X = X + 1: ON X GOTO 1
    65,170,175
165 CALL 768,A$(0): GOTO 180
170 CALL 768,B$(0): GOTO 185
175 CALL 768,C$(0): GOTO 190
180 FOR I = D TO N: PRINT A$(I)" ";; NEX
    T : GOTO 195
185 FOR I = D TO N: PRINT B$(I)" ";; NEX
    T : GOTO 195
190 FOR I = D TO N: PRINT C$(I)" ";; NEX
    T
195 VTAB 22: PRINT : INVERSE : PRINT "PR
    ESSEZ UNE TOUCHE S.V.P" ;; NORMAL : P
    RINT "> " ;; GET R$: PRINT
```

(suite page 26)

TRI.DEMO

(suite et fin)

```
200 IF R$ = CHR$(27) THEN PRINT : END
205 IF X < 3 THEN 155
210 HOME
215 N = 50
220 INPUT "NOMBRE ? ";N: DIM M$(N)
225 PRINT : PRINT "SAISIE DE "N" MOTS":
PRINT
230 FOR I = 1 TO N: INPUT "> ";M$(I): NE
XT
```

```
235 CALL 768,M$(0)
240 FOR I = 1 TO N: PRINT M$(I): NEXT
245 PRINT : VTAB 23: PRINT "<1> ENCORE <
2> MENU DE LA DISQUETTE <3> TERMINE
POUR AUJOURD'HUI $";: GET R$: VTAB 2
0: PRINT R$
250 HOME : IF R$ = "1" THEN CLEAR : GOTO
100
255 IF R$ = "2" THEN PRINT CHR$(4)"RUN
MENU"
260 IF R$ < > "3" THEN 245
```

TRI.1 (BSAVE TRI.1, A\$ 300, L\$9C)

```
0300- 20 BE DE JSR $DEBE
0303- 20 E3 DF JSR $DFE3
0306- 38 SEC
0307- E9 01 SBC £$01
0309- 85 85 STA $85
030B- B0 01 BCS $030E
030D- 88 DEY
030E- 84 86 STY $86
0310- A2 00 LDX £$00
0312- A1 85 LDA ($85,X)
0314- C9 02 CMP £$02
0316- B0 01 BCS $0319
0318- 60 RTS
0319- 85 FA STA $FA
```

CHKCOM : teste la virgule dans TXTPTR
PTRGET : recherche la variable par son nom

Au retour, A et Y contiennent la bonne adresse.
On lit la longueur du tableau à cette adresse moins un.
Ainsi, A = A - 1... si A = FF, Y = Y - 1
Le résultat est placé dans \$85-86

et le contenu est lu grâce à l'adressage indirect avec X

Si la longueur est inférieure à 2, pas de TRI
Sinon on continue
RETOUR au Basic

Le nombre d'éléments à trier est placé dans \$FA

```
031B- A5 83 LDA $83
031D- 85 85 STA $85
031F- A5 84 LDA $84
0321- 85 86 STA $86
0323- A2 00 LDX £$00
0325- A9 00 LDA £$00
0327- 85 09 STA $09
```

RÉINITIALISATION

L'adresse de base est contenue dans \$83-84 et il est facile de la réinstaller dans \$85-86

Sert de pointeur pour compter les éléments triés
Si 9 = 0, il n'y a pas eu d'échange.
Si 9 = 1, on a modifié la liste.

```
0329- A0 02 LDY £$02
032B- B1 85 LDA ($85),Y
032D- 99 06 00 STA $0006,Y
0330- 88 DEY
0331- 10 F8 BPL $032B
0333- E8 INX
0334- E4 FA CPX $FA
0336- F0 56 BEQ $038E
0338- A5 85 LDA $85
033A- 85 18 STA $18
033C- 69 03 ADC £$03
033E- 85 85 STA $85
0340- A5 86 LDA $86
0342- 85 19 STA $19
```

ÉLÉMENT 0

On écrit page 0 :
— 6 : longueur de l'élément
— 7 :] adresse où il est implanté dans la mémoire
— 8 :]

X = X + 1 pour ÉLÉMENT 1 (le suivant)
Si X = \$FA (nombre d'éléments à traiter)
GOTO \$38E

L'adresse + 3 (élément suivant) est sauvegardée dans \$18-19 (où elle sera utilisée plus loin pour la permutation éventuelle) et placée dans \$85-86

| | | | |
|-------|-------|-----|--------|
| 0344- | 69 00 | ADC | £\$00 |
| 0346- | 85 86 | STA | \$86 |
| 0348- | A5 06 | LDA | \$06 |
| 034A- | F0 DD | BEQ | \$0329 |

Si la longueur de l'ÉLÉMENT 0 est nulle, GOTO \$329

ÉLÉMENT 1

| | | | |
|-------|----------|-----|----------|
| 034C- | A0 02 | LDY | £\$02 |
| 034E- | B1 85 | LDA | (\$85),Y |
| 0350- | 99 FB 00 | STA | \$00FB,Y |
| 0353- | 88 | DEY | |
| 0354- | 10 F8 | BPL | \$034E |

\$FB = longueur de l'élément 1
\$FC] adresse où il est implanté
\$FD]

| | | | |
|-------|-------|-----|--------|
| 0356- | A5 FB | LDA | \$FB |
| 0358- | F0 1F | BEQ | \$0379 |
| 035A- | C5 06 | CMP | \$06 |
| 035C- | 90 02 | BCC | \$0360 |
| 035E- | A5 06 | LDA | \$06 |
| 0360- | 85 FE | STA | \$FE |

Si \$FB = 0, permutation immédiate

Si la longueur de l'ÉLÉMENT 1 est plus petite que celle de l'ÉLÉMENT 0, le contenu de A est bon. Sinon on prend la plus petite longueur que l'on stocke dans \$FE

COMPARAISON

| | | | |
|-------|-------|-----|----------|
| 0362- | A0 00 | LDY | £\$00 |
| 0364- | B1 FC | LDA | (\$FC),Y |
| 0366- | D1 07 | CMP | (\$07),Y |
| 0368- | F0 04 | BEQ | \$036E |
| 036A- | B0 2A | BCS | \$0396 |
| 036C- | D0 0B | BNE | \$0379 |
| 036E- | C8 | INY | |
| 036F- | C4 FE | CPY | \$FE |
| 0371- | D0 F1 | BNE | \$0364 |
| 0373- | A5 FB | LDA | \$FB |
| 0375- | C5 06 | CMP | \$06 |
| 0377- | B0 1D | BCS | \$0396 |
| 0379- | A9 01 | LDA | £\$01 |
| 037B- | 85 09 | STA | \$09 |

Caractère de l'ÉLÉMENT 1 (de rang Y) comparé à celui de l'ÉLÉMENT 0

Si égalité, la boucle continue
Si plus grand, pas de permutation (GOTO \$396)
Si plus petit, permutation immédiate (GOTO \$379)

Comparaison terminée si Y est égal à \$FE

Si la longueur de l'ÉLÉMENT 1 est égale ou plus grande que celle de l'ÉLÉMENT 0, pas de PERMUTATION

09 est mis à 1

PERMUTATION

| | | | |
|-------|----------|-----|----------|
| 037D- | A0 02 | LDY | £\$02 |
| 037F- | B1 85 | LDA | (\$85),Y |
| 0381- | 48 | PHA | |
| 0382- | B1 18 | LDA | (\$18),Y |
| 0384- | 91 85 | STA | (\$85),Y |
| 0386- | 68 | PLA | |
| 0387- | 91 18 | STA | (\$18),Y |
| 0389- | 88 | DEY | |
| 038A- | 10 F3 | BPL | \$037F |
| 038C- | 30 08 | BMI | \$0396 |
| 038E- | A5 09 | LDA | \$09 |
| 0390- | F0 03 | BEQ | \$0395 |
| 0392- | 4C 1B 03 | JMP | \$031B |
| 0395- | 60 | RTS | |
| 0396- | E4 FA | CPX | \$FA |
| 0398- | B0 F4 | BCS | \$038E |
| 039A- | 90 8D | BCC | \$0329 |

L'adresse de l'ÉLÉMENT 1 prend la place de celle de l'ÉLÉMENT 0... et vice-versa

GOTO \$396

S'il n'y a eu aucune permutation, tri terminé

Sinon on recommence

RETOUR AU BASIC

X est-il égal au nombre d'éléments à comparer ?

Egal ou supérieur, ultime vérification en \$38E

Inférieur, affaire à suivre...

DRAW et XDRAW

en langage machine

L

A routine en langage machine peut être utilisée sans faire appel au programme DÉMO. Elle utilise la fonction XDRAW (\$F65D) et elle tournera aussi longtemps que vous n'aurez pas pressé une touche.

Que fait l'instruction XDRAW ? Elle affiche une forme avec la couleur inverse de celle qui se trouve aux points où est déjà tracée la forme. Ainsi, noir devient blanc et vice-versa. C'est pourquoi avec XDRAW le programme fait et défait indéfiniment l'écran.

La forme est réduite à un octet (\$4) qui est placé en page zéro à l'adresse \$8 et terminée par un 0 en \$9. Dans la démo, XDRAW devient DRAW (\$F601)... ce qui modifie l'écran. Toutefois, comme DRAW se contente de tracer (sans effacer), on stoppe le programme dès la fin de la boucle (continuer ne modifierait pas l'affichage).

QUELQUES RAPPELS

En Basic, pour utiliser DRAW ou XDRAW, il est indispensable d'indiquer le numéro de la forme dans la **table de formes** et aussi de renseigner l'Apple sur l'adresse de cette table. Ainsi, pour notre mini-forme, nous aurions dû exécuter les opérations suivantes : * FA: 1 0 4 0 4 0 * E8: FA 0

Le 1 indique le nombre de formes. Le premier 4 indique la partie basse du nombre d'octets séparant le début de la table de la première forme. Le dernier 4 (suivi du zéro) est la forme elle-même (ici, il n'y en a qu'une).

Et ensuite, en Basic :

```
10 HCOLOR = 3 : HGR : SCALE = 1 : ROT = 0 :
   X = 139 : Y = 95
20 DRAW 1 AT X,Y
```

XDRAW-DEMO

```
100 PRINT CHR$(21): HOME
110 PRINT CHR$(4)"BLOADXDRAW.L
    M": CALL 768
120 CALL - 198: GET R$: PRINT :
    POKE - 16301,0
130 VTAB 22: CALL - 958: PRINT
    "Le meme avec DRAW au lieu d
    e XDRAW " ;: GET R$
140 POKE 865,01: POKE 826,3: CALL
    768
150 CALL - 198: GET R$: PRINT :
    POKE - 16301,0
160 VTAB 22: PRINT "<1> ENCORE <
    2> MENU DISQUETTE <3> FIN " ;
    : GET R$
170 PRINT : POKE 826,225: POKE 8
    65,93: IF R$ = "1" THEN CALL
    768: GOTO 120
180 IF R$ = "2" THEN PRINT CHR$(
    4)"RUN MENU"
190 IF R$ < > "3" THEN 160
200 TEXT : HOME
```

Un RUN affichera un point. Puis, si vous tapez, en mode direct :

```
SCALE = 10 : GOTO 20
```

vous afficherez, 10 points... qu'un XDRAW 1 AT X,Y fera disparaître. C.Q.F.D.

En langage machine, il est inutile de renseigner les pointeurs \$E8-E9 sur l'adresse de la table de formes. Par contre, on devra placer dans X la partie BASSE de l'adresse du premier octet de la forme et dans Y la partie HAUTE de cette même adresse.

XDRAW.LM

| | | | | | |
|-------|----------|-----|-------|---|---|
| 0300- | A9 95 | LDA | £\$95 |] CTRL-U (envoyé par COUT) va désactiver la carte 80 colonnes | |
| 0302- | 20 ED FD | JSR | £FDED | | |
| 0305- | A9 20 | LDA | £\$20 | | On place \$20 (valeur pour HGR) dans HPAG (\$E6). |
| 0307- | 85 E6 | STA | £E6 | | Cette adresse contient \$40 pour HGR2 |
| 0309- | A9 7F | LDA | £\$7F | | Code de la couleur : \$7F=3 |
| 030B- | 85 E4 | STA | £E4 | | (0=0, \$2A=1, \$55=2, \$80=4, \$AA=5, \$D5=6, \$FF=7) |
| 030D- | A9 04 | LDA | £\$04 | |] Ça, c'est notre forme (un point) |
| 030F- | 85 08 | STA | £08 | | |
| 0311- | 20 F2 F3 | JSR | £F3F2 | | Efface l'écran (HCLR) |
| 0314- | 20 E2 F3 | JSR | £F3E2 | | Initialise la page HGR (\$F3D8 pour HGR2) |
| 0317- | 2C 52 C0 | BIT | £C052 | C'est POKE — 16302,0 : plus de fenêtre texte ! | |

INITIALISATION

| | | | | |
|-------|-------|-----|-------|--|
| 031A- | A9 00 | LDA | £\$00 | Accumulateur chargé avec 0 |
| 031C- | 85 06 | STA | £06 |] Pointeurs pour une boucle allant de 1 à 1024 |
| 031E- | 85 07 | STA | £07 | |
| 0320- | 85 09 | STA | £09 | Dernier octet de la forme (on profite de A=0) |
| 0322- | 85 F9 | STA | £F9 | ROT=0 |
| 0324- | A9 01 | LDA | £\$01 |] SCALE=1 (\$E7=0 correspond à SCALE=256) |
| 0326- | 85 E7 | STA | £E7 | |

BOUCLE

| | | | | |
|-------|----------|-----|-------|--|
| 0328- | E6 06 | INC | £06 | Plus un pour la boucle |
| 032A- | D0 13 | BNE | £033F | Si le résultat est différent de 0, on saute |
| 032C- | E6 07 | INC | £07 | Sinon, il faut incrémenter la partie haute de l'adresse |
| 032E- | A5 07 | LDA | £07 |] Celle-ci est-elle égale à 04 ? (autrement dit : est-on arrivé à \$400 ?) |
| 0330- | C9 04 | CMP | £\$04 | |
| 0332- | D0 0B | BNE | £033F | Si la réponse est non, on continue |
| 0334- | AD 00 C0 | LDA | £C000 |] Une touche a peut-être été pressée ? |
| 0337- | C9 7F | CMP | £\$7F | |
| 0339- | 90 E1 | BCC | £031C | Non, on continue. |
| 033B- | 2C 10 C0 | BIT | £C010 | Oui, on arrête, mais sans oublier un POKE — 16368,0 |
| 033E- | 60 | RTS | | Retour au Basic |

ON POSITIONNE LE CURSEUR

| | | | | |
|-------|----------|-----|-------|--|
| 033F- | A0 00 | LDY | £\$00 |] X, position horizontale, est placé dans les registres Y (partie haute) et X (partie basse). Y, position verticale, est placée dans l'accumulateur. |
| 0341- | A2 0B | LDX | £\$0B | |
| 0343- | A9 5F | LDA | £\$5F | |
| 0345- | 20 11 F4 | JSR | £F411 | |

XDRAW OU DRAW

| | | | | | |
|-------|----------|-----|-------|---|----------------------------|
| 0348- | A0 00 | LDY | £\$00 |] L'adresse de la forme est placée dans Y (partie haute) et X (partie basse). | |
| 034A- | A2 0B | LDX | £\$0B | | |
| 034C- | E6 F9 | INC | £F9 |] ROT (\$F9) = ROT + 1 Quand ROT = \$41 (65), on remet la valeur 1 dans \$F9 On en profite pour augmenter SCALE de 16 (\$10) Notons que la valeur de ROT doit être dans l'Accumulateur | |
| 034E- | A5 F9 | LDA | £F9 | | |
| 0350- | C9 41 | CMP | £\$41 | | |
| 0352- | D0 0C | BNE | £0360 | | |
| 0354- | A9 01 | LDA | £\$01 | | |
| 0356- | 85 F9 | STA | £F9 | | |
| 0358- | 48 | PHA | | | On empile la valeur de ROT |
| 0359- | A9 10 | LDA | £\$10 | |] SCALE = SCALE + \$10 |
| 035B- | 65 E7 | ADC | £E7 | | |
| 035D- | 85 E7 | STA | £E7 | | |
| 035F- | 68 | PLA | | | |
| 0360- | 20 5D F6 | JSR | £F65D | Récupération de ROT sur la pile XDRAW ou DRAW (\$F601) | |

BSAVE XDRAW.LM,
A\$300, L\$66

NEXT

| | | | | |
|-------|----------|-----|-------|------------------------------------|
| 0363- | 4C 28 03 | JMP | £0328 | Et ça repart pour un autre tracé ! |
|-------|----------|-----|-------|------------------------------------|

TREIZE

PETIT JEU DE LETTRES

L'

APPLE choisit un mot parmi ceux d'une liste présentée sous forme de DATAs. Il en mélange les lettres et les affiche sur l'écran graphique. Vous avez déjà compris qu'il s'agit ensuite de les placer, mais en bon ordre, dans les treize cases prévues à cet effet.

Pas de définition : si le mot est peu connu, c'est coton ! Heureusement, il suffit de taper un 0 (zéro) pour obtenir un renseignement. Celui-ci coûte un point, mais permet de placer exactement la première lettre (ou la suivante... si la première a déjà été localisée, et ainsi de suite).

Quand on croit connaître le mot, on tape seulement un T et on est invité à fournir la solution (attention ! en cas d'erreur, cela coûte un point !). Ce petit programme est entièrement en Basic, mais utilise une fonte spéciale de caractères (voir plus loin).

NOTA : Vous pourrez contrôler la saisie de TREIZE et de TREIZE.FONTE en utilisant la disquette VERIF de notre ami Yvan KOENIG (page 2 de couverture).

```

100 TEXT : PRINT CHR$(21): HOME : POKE 768
      ,0
      320D
105 LOMEM: 16384 + 1476:A = 16384: GOSUB 545
      : GOSUB 610
      6B9E
110 HOME :NM = 7: REM NOMBRE DE MOTS EN DATA 8E73
115 DIM L$(13),M$(13),M(13)
      032B
120 GOSUB 400
      FC4C
125 HGR : HCOLOR= 3: SCALE= 1: ROT= 0
      8D96
130 :
      003A
135 REM *** CASES POUR LE MOT ***
140 :
      003A
145 X = 3:Y = 27
      8627
150 HPL0T 0,0 TO 279,0 TO 279,30 TO 0,30 TO
      0,0
      F99D
155 HPL0T X,X TO 276,X TO 276,Y TO X,Y TO X,
      X
      5C73
160 FOR I = 24 TO 265 STEP 21: HPL0T I,X TO
      I,Y: HPL0T I + 2,X TO I + 2,Y: HCOLOR= 0
      : HPL0T I + 1,X TO I + 1,Y: HCOLOR= 3: N
      EXT
      BEA0
165 :
      003A
170 REM *** LETTRES MELANGEES ***
175 :
      003A
180 D = 9:B = 80
      F001
185 FOR I = 1 TO 13:E = 0:N = ASC (M$(I)) -
      64: GOSUB 590
      C4C4
190 DRAW N AT D + E,B
      3E66
195 D = D + 21: NEXT
      443F
200 :
      003A
205 REM *** ET MAINTENANT, JOUONS ***
210 :
      003A
215 VTAB 21: HTAB 1: PRINT "Renseignement: "
      ;: INVERSE : PRINT 0;: NORMAL : INPUT "
      (ou RANG + LETTRE)ctrlG ";R$
      3F27
220 IF R$ = "SOL" THEN PE = PE + 13 - TR: GO
      TO 430
      72FA
225 IF R$ = "T" THEN 425
      0CEA
230 IF R$ = "" THEN 215
      2A93
235 IF R$ < > "0" THEN 290
      8196
240 IF RS = 13 THEN T$ = "PLUS DE RENSEIGNEM
      ENT": GOSUB 575: GOTO 215
      DFBF
245 RS = RS + 1
      6A13
250 FOR I = 1 TO 13: IF M$(I) < > L$(RS) TH
      EN 260
      B944
255 IF M(I) THEN PE = PE + 1: GOTO 270
      B9C9
260 NEXT
      0582
265 GOTO 240
      4B41
270 IF I = RS THEN DR = 27
      AFFE
275 IF I > RS THEN DR = 28
      83FE
280 IF I < RS THEN DR = 29
      3801
285 D = 21 * I - 12: DRAW DR AT D,45:M(I) =
      0: GOTO 215
      0956

```



```

290 IF LEN (R$) = 1 THEN 315
295 L$ = RIGHT$ (R$,1): IF L$ < "A" OR L$ <
"Z" THEN 315
300 NU = VAL ( LEFT$ (R$, LEN (R$) - 1)): I
F NU < 1 OR NU > 13 THEN 315
305 IF L$(NU) = "" THEN T$ = "LETTRE DEJA TR
OUVEE": GOSUB 575: GOTO 215
310 GOTO 320
315 T$ = "ERREUR DE SAISIE": GOSUB 575: GOTO
215
320 FOR I = 1 TO 13: IF M$(I) = L$ THEN 330
325 NEXT : GOTO 405
330 IF M$(I) = L$(NU) THEN D = 21 * I - 12: N
= ASC (L$) - 64: GOSUB 590: XDRAW N AT
D + E,80:D = 21 * NU - 12: DRAW N AT D +
E,10:M$(I) = "":L$(NU) = "":M(I) = 0:TR
= TR + 1: IF TR < 13 THEN 215
335 IF TR < > 13 THEN 405
340 :
345 REM *** FIN DE LA PARTIE ***
350 :
355 PT = 13 - PE
360 VTAB 21: CALL - 958: INVERSE : PRINT "S
CORE:";: NORMAL : PRINT " ";: IF NOT PT
THEN PRINT " TRES MAUVAIS (<PT> POINT)"
: GOTO 385
365 IF PT < = 10 AND PT > = 6 THEN PRINT
PT" POINTS (BON RESULTAT)": GOTO 385
370 IF PT > 10 THEN PRINT PT" POINTS (EXCEL
LENT RESULTAT)": GOTO 385
375 PRINT " A VOUS DE JUGER! (<PT> POINT");:
IF PT > 1 THEN PRINT "S";
380 PRINT ")"
385 VTAB 23: PRINT "<1> ENCORE <0> TERMINE P
OUR CETTE FOIS ctrlG";: GET R$
390 IF R$ = "1" THEN CLEAR : GOTO 110
395 IF R$ < > "0" THEN VTAB 22: PRINT : GO
TO 385
400 TEXT : HOME : END
405 T$ = "MAUVAISE REPONSE": GOSUB 575:PE =
PE + 1: GOTO 215
410 :
415 REM *** SOLUTION ***
420 :
425 VTAB 21: CALL - 958: INPUT "MOT ENTIER
";T$: IF T$ < > M$ THEN GOTO 405
430 GOSUB 515
435 D = 9:B = 10
440 FOR I = 1 TO 13:E = 0:N = ASC (L$(I)) -
64: IF N = 9 THEN E = 4
445 DRAW N AT D + E,B
450 D = D + 21: NEXT
455 GOTO 355
460 :

```

```

5785
C002
DBB5
3CCB
2D40
3CF0
A17C
FE00
2589
15B4
003A
003A
4B36
ADD4
4965
5912
4723
7627
74E0
A960
6B7A
5514
8158
003A
003A
1C5E
EB4B
F9FA
5D86
3E66
443F
4948
003A

```

```

465 REM *** CHOIX ET TRAITEMENT ***
470 REM *** DU MOT (13 LETTRES) ***
475 :
480 CH = 1 + INT ( RND (1) * NM)
485 V = PEEK (768): IF V = NM OR NOT V THE
N POKE 768,1: POKE 769,CH: GOTO 500
490 FOR K = 1 TO V: IF PEEK (768 + K) = CH
THEN 480
495 NEXT : POKE 769 + V,CH: POKE 768,V + 1
500 RESTORE : FOR I = 1 TO CH: READ M$: NEXT
505 FOR I = 1 TO 13:L$(I) = MID$ (M$,I,1):
NEXT
510 FOR I = 13 TO 1 STEP - 1:J = INT ( RND
(1) * I + 1):M$(I) = L$(J):L$(J) = L$(I)
: NEXT
515 FOR I = 1 TO 13:L$(I) = MID$ (M$,I,1):M
(I) = 1: NEXT
520 RETURN
525 :
530 REM *** INSTALLATION DE LA ***
535 REM *** FONTE (30 CARACTERES) ***
540 :
545 PRINT CHR$ (4)"BLOAD TREIZE.FONTE,A",A
550 POKE 232,A - INT (A / 256) * 256: POKE
233, INT (A / 256)
555 RETURN
560 :
565 REM *** SOUS-PROGRAMMES ***
570 :
575 VTAB 22: CALL - 958: VTAB 23: HTAB (41
- LEN (T$)) / 2: INVERSE : PRINT T$: NOR
MAL : IF RV THEN RV = 0: RETURN
580 FOR K = 1 TO 2000: NEXT : VTAB 22: CALL
- 958: VTAB 23: HTAB 3: PRINT "Tapez <T>
si vous connaissez le mot"
585 RETURN
590 E = 0: IF N = 9 THEN E = 4
595 RETURN
600 :
605 REM *** PRESENTATION ***
610 :
615 HOME : INVERSE :T$ = " -----": VT
AB 1: PRINT T$: VTAB 3: PRINT T$: VTAB 2
: PRINT " T R E I Z E ";: NORMAL : HTAB
29: PRINT "REGLE DU JEU": VTAB 3: HTAB 2
9: PRINT "-----"
620 PRINT : PRINT "OBJET: Trouver un mot de
13 lettres qui ----- est fourni dans le
désordre."
625 PRINT : PRINT : PRINT " - Tout renseigne
ment demandé fait per- ---dre un point"
630 PRINT : PRINT " - Pénalité identique pou
r toute erreur"

```

(suite page 32).

TREIZE (suite)

```

635 PRINT : PRINT " - TAPER <SOL> pour annul
      er la partie."
640 VTAB 20: FOR I = 1 TO 10: PRINT " ____";
      NEXT : PRINT
645 VTAB 22:T$ = " PRESSEZ UNE TOUCHE SVP ":
      RV = 1: GOSUB 575
  
```

```

650 CALL - 198: WAIT - 16384,128,127: POKE
      - 16368,0: HOME : RETURN
655 :
660 REM *** MOTS DE 13 LETTRES ***
665 :
670 DATA INTERLOCUTEUR,CONFERENCIERE,DEMISTI
      FIANTE,FRAGILISATION,GALEOPITHEQUE,GIGAN
      TOMACHIE,HELICICULTURE
  
```

TREIZE.FONTE

```

4000: 31 02 41 00 7C 00 B5 00 E0 00 1E 01 4E 01 71 01 ££3A65
4010: A9 01 E1 01 F4 01 15 02 4A 02 69 02 AD 02 E5 02 ££BCE5
4020: 20 03 50 03 8E 03 C6 03 F6 03 14 04 4B 04 82 04 ££14B6
4030: C3 04 EC 04 1E 05 48 05 66 05 85 05 A3 05 C3 05 ££108C
4040: 00 00 2D 2D 2D 15 3F 3F 3F 3F 2E 4D 49 35 FF DB ££686B
4050: 3B 2E 4D 49 35 FF DB 3B 17 2D 2D 2D 2D 35 3F 3F ££A3C7
4060: 3F 3F 77 6D 49 29 3E DF DB 37 6D 49 29 3E DF DB ££4EDA
4070: 37 6D 49 29 3E DF DB B7 49 49 49 01 00 29 2D 2D ££D724
4080: AD 3F 3F 3F 37 6D 49 35 FF DB 37 6D 49 35 3F 3F ££FE06
4090: 3F 3F 2E 2D 2D 15 FF DB 37 6D 49 29 3E DF DB ££4230
40A0: 37 6D 49 29 3E DF DB 37 2D 2D 2D 2D 1E 3F 3F 3F ££33D4
40B0: B7 49 49 49 01 00 09 2D 2D 2D 15 3F 3F 3F 3F 17 ££7B4B
40C0: 2D 4D 49 F5 DB DB 3B 2E 3E 2E 3E 2E 3E 2E 4D 49 ££34B1
40D0: 29 3E DF DB 3F 0E 2D 2D 2D F5 3F 3F B7 49 49 49 ££AFFA
40E0: 00 29 2D 2D AD 3F 3F 3F 37 6D 49 29 15 FF DB 1B ££A10D
40F0: 37 6D 49 09 35 FF DB 1B 37 6D 49 09 35 FF DB 1B ££3740
4100: 37 6D 49 09 35 FF DB 1B 37 6D 49 09 35 FF DB 1B ££3740
4110: 37 2D 2D 2D 1E 3F 3F 3F B7 49 49 49 09 00 29 ££428A
4120: 2D 2D 2D 15 3F 3F 3F 3F 2E 4D 49 F5 DB DB 37 35 ££D573
4130: 37 2D 2D 2D 3E 3F 3F 37 35 37 6D 49 29 3E DF DB ££56F4
4140: 37 2D 2D 2D 1E 3F 3F 3F B7 49 49 49 01 00 29 ££3182
4150: 2D 2D 2D 15 3F 3F 3F 3F 2E 4D 49 F5 DB DB 37 35 ££D573
4160: 37 2D 2D 35 3F 3F 37 35 37 35 37 35 B7 49 49 49 ££821A
4170: 01 00 09 2D 2D 2D 15 3F 3F 3F 3F 17 2D 4D 49 F5 ££9D71
4180: DB DB 3B 2E 3E 2E 4D 49 09 3F FF DB 37 6D 49 2D ££53DD
4190: 35 FF DB 1B 37 6D 49 09 35 FF DB 3B 77 2D 2D 2D ££8168
41A0: 2D 1E 3F 3F 3F 56 49 49 09 00 29 4D 49 35 FF DB ££96C7
41B0: 3B 2E 4D 49 35 FF DB 3B 2E 4D 49 35 3F 3F 3F 3F ££AC3E
41C0: 2E 2D 2D 2D 35 FF DB 3B 2E 4D 49 35 FF DB 3B 2E ££AC3B
41D0: 4D 49 35 FF DB 3B 2E 4D 49 35 FF DB 3B 56 49 49 ££E0D6
41E0: 49 00 2D F5 37 35 37 35 37 35 37 35 37 35 37 AD ££D16B
41F0: 3F B7 49 09 00 49 49 29 3E 2E 3E 2E 3E 2E 3E 2E ££C9B3
4200: 3E DF DB 37 6D 49 29 3E FF DB 37 2D 2D 2D F5 3F ££C718
4210: 3F B7 49 49 00 29 4D 49 35 FF DB 3B 2E 4D 49 ££BF9E
4220: 35 FF DB 3B 2E 4D 49 F5 FF DB 37 6D 09 F5 3F 3F ££64FD
4230: 37 2D 2D 2D 15 FF DB 37 6D 49 29 3E DF DB 37 6D ££385F
4240: 49 29 3E DF DB B7 49 49 49 01 00 29 3E 2E 3E 2E ££7EFE
4250: 3E 2E 3E 2E 3E 2E 4D 49 35 FF DB 3B 2E 2D 2D 2D ££34D9
4260: F5 3F 3F 3F B7 49 49 49 01 00 09 2D 4D AD 3F 3F ££05F3
4270: 3F 3F 17 6D 09 6D 29 3E DF FF 1B 37 6D 09 6D 29 ££791B
  
```

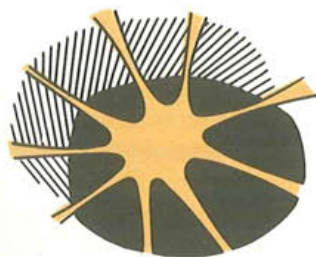
EXPLICATIONS DU PROGRAMME BASIC

- 100 :** Pourquoi POKE 768,0 ? explication à la ligne 485.
- 105 :** LOMEM est placé au-dessus de la fonte de caractères.
- 115 :** L\$(n) = lettres du mot
M\$(n) = lettres mêlées
M(n) = à 1 au début du jeu et à 0 quand M\$(n), la lettre correspondante, a été utilisée.
- 185 :** La variable E est égale à 0... sauf quand le caractère à afficher est un I (à ce moment, E=4... voir ligne 590). Le E=0 n'est pas utile (on le retrouve à la ligne 590).
- 215 :** Notez la présence d'un CTRL-G avant les guillemets de fin de ligne (pour le placer, appuyez simultanément sur les touches CTRL et G).
- 245 :** RS totalise le nombre de renseignements. Au début du jeu, on donne un tuyau sur la lettre n°1... et ainsi de suite.
- 250 :** Si L\$(RS) ne contient plus rien (c'est le cas quand la lettre a déjà été placée), on continue la boucle.
- 255 :** PE totalise les points de pénalité (un par renseignement et un par erreur manifeste). Si M(I) est égal à 0, on continue.
- 265 :** Le renseignement n'a pu être donné parce que la lettre n°RS a déjà été trouvée. RS va donc être incrémenté pour donner un renseignement sur le caractère suivant... s'il existe.

- 270 :** DR est le numéro du caractère de la fonte spéciale (celle-ci commence par A). On a :
27 = , 28 = et 29 = .
- 290 :** Si on n'a qu'un caractère, la réponse est forcément mauvaise. A cet endroit du programme, on attend une réponse de type 1A, 12T, etc.
- 330 :** XDRAW efface la lettre utilisée, mais on laisse les flèches correspondant aux renseignements donnés (cela permet de contrôler le nombre de pénalités !).
- 430 :** Avant d'afficher le mot entier, on recharge L\$(n) avec les lettres du mot. On pourrait éviter ce Gosub et utiliser la fonction MID\$(M\$,I,1) à la ligne 440.
- 480 :** Classique utilisation de RND. NM renferme le nombre de données (mots).
- 485 :** On utilise l'adresse 768 et les NM octets suivants pour y stocker le nombre de mots utilisés et le numéro d'ordre de chacun d'eux. CLEAR n'efface pas la page 3 et, de cette manière, on va aléatoirement choisir TOUS les mots de la liste. Quand celle-ci est épuisée... on annule tout et on recommence !
- 510 :** Mélange bien connu des caractères.
- 550 :** A = Adresse d'installation. Comme elle est connue (A=16384... \$4000), on pourrait poquer directement les valeurs :
— POKE 232,0 (\$0)
— POKE 233,64 (\$40)

Amusez-vous bien !

GÉO.



4280: 3E DF FF 1B 37 6D 09 6D 29 3E DF FF 1B 37 6D 09 ££A75E
 4290: 6D 29 3E DF FF 1B 37 6D 09 6D 29 3E DF FF 1B 37 ££F07E
 42A0: 6D 09 6D 29 3E DF FF 1B B7 49 49 49 00 09 2D ££5154
 42B0: 2D 2D 15 3F 3F 3F 3F 2E 4D 49 35 FF DB 3B 2E 4D ££B1F4
 42C0: 49 35 FF DB 3B 2E 4D 49 35 FF DB 3B 2E 4D 49 35 ££EF9A
 42D0: FF DB 3B 2E 4D 49 35 FF DB 3B 2E 4D 49 35 FF DB ££A8F6
 42E0: 3B 56 49 49 49 00 09 2D 2D AD 3F 3F 3F BF 6D 49 ££EDAE
 42F0: 09 35 FF DB 1B 37 6D 49 09 35 FF DB 1B 37 6D 49 ££7640
 4300: 09 35 FF DB 1B 37 6D 49 09 35 FF DB 1B 37 6D 49 ££7640
 4310: 09 35 FF DB 1B 77 2D 2D 2D F5 3F 3F B7 49 49 49 ££0936
 4320: 00 29 2D 2D 2D 15 3F 3F 3F 3F 2E 4D 49 35 FF DB ££C394
 4330: 3B 2E 4D 49 35 FF DB 3B 2E 4D 49 35 3F 3F 3F 3F ££AC3E
 4340: 37 2D 2D 2D 2D DE DB 3B 2E 3E 2E 3E 56 49 49 49 ££17E8
 4350: 00 09 2D 2D AD 3F 3F 3F BF 2D 4D 09 2D 3E DF DB ££C034
 4360: 3B 2E 4D 49 29 3E DF DB 3B 2E 4D 49 29 3E DF DB ££E940
 4370: 3B 2E 4D 49 29 3E DF DB 3B 2E 4D 29 0D 35 3F FF ££627F
 4380: 1B 3F 0E 2D 2D 2D AD FF 3F 3F 56 49 49 09 00 29 ££8D33
 4390: 2D 2D AD 3F 3F 3F 37 6D 49 29 3E DF DB 37 6D 49 ££4FBF
 43A0: 29 3E DF DB BF 2D 2D 2D 2D 1E 3F 3F 3F 77 6D 49 ££149C
 43B0: AD FF DB 3B 2E 4D 49 35 FF DB 3B 2E 4D 49 35 FF ££B6C8
 43C0: DB 3B 56 49 49 49 00 09 2D 2D 2D 15 3F 3F 3F 3F ££21E8
 43D0: 17 2D 4D 49 F5 DB DB 3B 2E 6D 3A 3F 0E 2D 2D AD ££1CE9
 43E0: 3F 77 29 35 37 35 3F DF DB 37 2D 2D 2D 2D 1E 3F ££32C1
 43F0: 3F 3F 56 49 49 49 00 2D 2D 2D 2D 35 3F 3F 3F 3F ££6A94
 4400: 37 6D 29 4D F5 DB 37 35 37 35 37 35 37 35 37 35 ££F006
 4410: B7 49 49 01 00 6D 49 29 3E DF DB 37 6D 49 29 3E ££B775
 4420: DF DB 37 6D 49 29 3E DF DB 37 6D 49 29 3E DF DB ££1CD6
 4430: 37 6D 49 29 3E DF DB 37 6D 49 2D 3E 3F DF 3B 2E ££B1ED
 4440: 2D 2D 0D 35 FF 3B 3F 56 49 49 09 00 6D 49 09 35 ££DAFA
 4450: FF DB 1B 37 6D 49 09 35 FF DB 1B 37 6D 49 09 35 ££4340
 4460: FF DB 1B 37 6D 49 09 35 FF DB 1B 37 6D 49 09 35 ££4340
 4470: 3F DF 1B 3F 0E 2D 4D 2D 1E 3F 3F 77 2D F5 B7 49 ££0462
 4480: 49 01 00 6D 09 6D 29 3E DF FF 1B 37 6D 09 6D 29 ££79D0
 4490: 3E DF FF 1B 37 6D 09 6D 29 3E DF FF 1B 37 6D 09 ££A75E
 44A0: 6D 29 3E DF FF 1B 37 6D 09 6D 29 3E DF FF 1B 37 ££F07E
 44B0: 6D 09 6D 29 3E FF FF 3B 0E 2D 2D 2D F5 FF 3B 56 ££8F9D
 44C0: 49 49 09 00 29 4D 09 35 FF DB 37 6D 49 35 FF DB ££6E25
 44D0: 77 6D 29 1E 3F 77 35 BF 2D AD FF 3B 17 6D 49 35 ££A5EB
 44E0: FF DB 37 6D 49 35 FF DB B7 49 49 49 00 2D 4D 09 ££FAEB
 44F0: 35 FF 1B 3F 0E 6D 09 35 FF DB BF 6D 49 29 3E DF ££19DC
 4500: DB 37 6D 49 2D 3E 3F DF 77 2D 6D 35 37 35 3F DF ££EF21
 4510: 1B 37 2D 2D 2D 2D 1E 3F 3F 3F 56 49 49 09 00 29 ££A4FB
 4520: 2D 2D 2D 35 3F 3F 3F 3F 4E 49 29 F5 3F 17 2D 1E ££450E
 4530: 3F 17 2D 3E BF 2D 3E BF 2D 3E 37 2D 2D 2D 2D 3E ££9C3E
 4540: 3F 3F 3F B7 49 49 49 01 00 92 4A 09 15 3F 17 2D ££37CD
 4550: 2D 15 3F 3F 3F 17 2D 2D 2D 2D DE 3B 37 2D 3E 37 ££B8BC
 4560: 2D 3E B7 49 49 01 00 92 4A 09 3E 17 2D 4D 09 3F ££4A31
 4570: 3F 3F 3F 17 2D 2D 2D 2D 35 3F 3F 3F 3F 0E 2D 3E ££9532
 4580: 0E 96 49 49 01 00 92 4A 49 15 37 2D 15 3F 3F 3F ££FEA7
 4590: 3F 2E 2D 2D 2D 2D 1E 3F 3F 3F 3F 4E 49 2D 1E 37 ££A954
 45A0: 96 49 49 00 29 2D 2D AD 3F 3F 3F 3F 2E 4D 49 29 ££0441
 45B0: 3E 2E 3E DF 9B 2D 2D F5 3F 3F 2E 3E 2E 9E 35 B7 ££2E15
 45C0: 49 49 09 00 ££729B

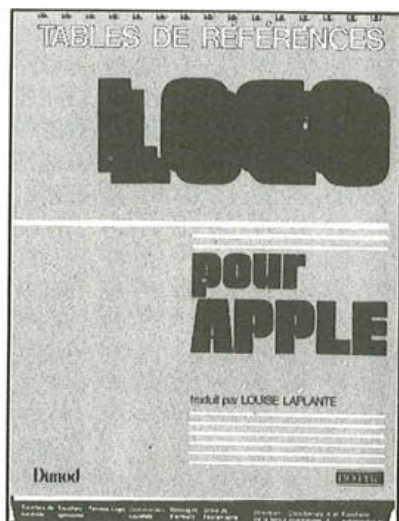
BSAVE TREIZE.FONTE, A\$4000, L\$5C4

Votre bibliothèque INFORMATIQUE

par Clément RENARD

• LOGO POUR APPLE (Dunod/BO-PRE)

Ces tables de références nous arrivent du Canada et nous les devons aux Editions BO-PRE. Elles ont été traduites de l'américain (*Logo Reference Flip Chart*) par Louise Laplante. La bonne idée que voilà ! Un système astucieux permet en effet de disposer ces tables debout, près de l'Apple, et de les consulter facilement, au fur et à mesure des nécessités. Si vous vous intéressez au Logo, vous saurez l'essentiel sur les touches de contrôle, les touches spéciales, les termes Logo, les commandes usuelles, les messages d'erreur, la grille de l'écran-texte, la direction de la tortue, les coordonnées, les fonctions mathématiques, les commandes pour fichiers et les exemples de procédures. C'est une belle édition, sur un papier de qualité. La partie chevalet est en carton super rigide. Bref : du bel ouvrage !



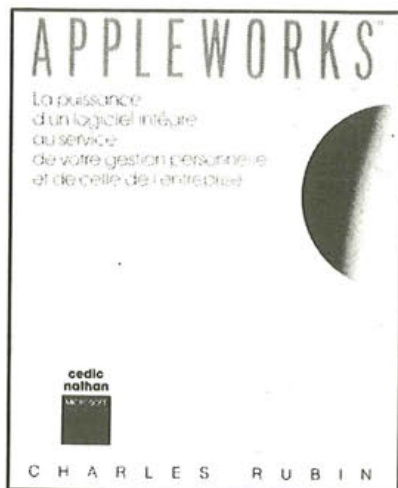
• LOTUS 1.2.3. POUR IBM PC (Dunod/BO-PRE)

Même formule que le LOGO POUR APPLE, mais il s'agit cette fois de

tables de références destinées aux utilisateurs du fameux LOTUS 1.2.3... sur IBM PC (et compatibles). Cette édition mérite les mêmes éloges que la précédente, mais elle est beaucoup plus consistante : LOTUS est un gros morceau !

• APPLEWORKS (Cédic-Nathan)

Voici une bonne traduction du livre de Charles Rubin, auteur auquel la Californie et *Appleworks* ont inspiré des expériences intéressantes... et un ouvrage qui ne l'est pas moins. On sait qu'*Appleworks* est en passe de devenir un produit modèle dans la gamme des logiciels Apple, mais en connaît-on réellement les multiples possibilités ? Il est permis d'en douter, surtout quand on lit, sous la plume de Rupert Lissner, le constat suivant : "Les développeurs de logiciels comme moi se demandent souvent quelle infime fraction des applications qu'ils ont si laborieusement construites pour leurs logiciels est effectivement utilisée. Un programmeur ne saurait cependant être plus satisfait de l'utilisateur nommé Charles Rubin qui réalise avec *Appleworks* des opérations que je n'aurais



pas moi-même envisagées. Par exemple, sa technique de création de lettres avec *Appleworks*, qui, sans rendre nécessaire l'achat d'un second programme, vaut à elle seule le prix de cet ouvrage...". Et Rupert Lissner sait de quoi il parle ! C'est dire tout l'intérêt de ce livre !



• LA ROM DE L'APPLE II (Sybex)

En quelques mois, au PSI et chez SYBEX, Marcel Cottini a publié plusieurs ouvrages magistraux sur Apple : *L'assembleur de l'Apple II* et *Programmation système de l'Apple II* (voir notre dernier numéro) et *La Rom de l'Apple II...* qui nous intéresse aujourd'hui.

Je connais Marcel Cottini et cela me gêne un peu car je n'ai qu'un mot à dire sur son bouquin : ACHETEZ ! Offrez-vous cette nouvelle Bible de l'Apple II, notamment si vous pratiquez plus ou moins le langage machine. Il vous éclairera sur bien des points de la ROM... et en français, ce qui est tout de même rare ! Notez qu'un autre volume sera consacré à l'Apple IIc.

UN LOGICIEL FRANÇAIS

aussi simple qu'efficace

V

ERSION Liste nécessite une souris, un Apple IIc ou IIe et un lecteur de disquettes. Comme son nom l'indique, ce logiciel offre un certain nombre de facilités à celles et à ceux qui sont amenés à utiliser des fichiers. Il permet en effet de créer une liste, de la trier, de l'imprimer, de la sauver dans les fichiers mailing d'Epistole... tout cela d'une manière simple et efficace.

Le manuel d'initiation :

Rien à redire. Comme la plupart des documentations accompagnant les produits de Version Soft, il se veut court, voir succinct, mais d'une grande clarté. Il autorise une prise en main qui, dans le pire des cas, ne dépassera pas quelques petites heures. Dans une première partie, on initie d'abord les débutants au maniement de la souris et des menus déroulants, mais la plupart des utilisateurs escamoteront sans doute ces quelques pages.

On passe ensuite au répertoire des diverses options de Version Liste, lesquelles sont étudiées dans un ordre on ne peut plus logique. Des photographies d'écrans complètent un texte dont on apprécie la concision. A noter que la dernière partie du manuel dévoile un certain nombre d'astuces capables d'éviter d'inutiles manipulations.

L'utilisation Ce qui saute d'abord aux yeux, c'est la qualité du graphisme des écrans. Double haute définition oblige ! Autre remarque : une rapidité de bon aloi. Version Liste fonctionne sous ProDOS et il est écrit en assembleur. Ceci explique évidemment cela. Détail intéressant : suivant le matériel de l'utilisateur, ce logiciel est fourni sur une disquette 5 pouces 1/4 ou sur une disquette 3 pouces 1/2. Je n'ai personnellement testé que la première version et je le regrette. Il est certain que l'Unidisk et des disquettes de 800 Ko offrent de meilleures possibilités.

En effet, dans certains cas, avec les disquettes 5 pouces 1/4, il est nécessaire d'introduire la face Exemples dans le lecteur principal... et l'on se passerait assurément de cette petite gymnastique. Reproche mineur, mais reproche tout de même : il aurait été facile de fournir deux disquettes (dont le coût n'aurait d'ailleurs pas dépassé celui d'une seule disquette 3 pouce 1/2). Je suis sûr que les responsables commerciaux de Version Soft vont remédier à cela dans de très brefs délais.

Car on ne saurait émettre de critiques malveillantes sur le reste. Comme le dit la notice, avec Version Liste, vous gérez vos fichiers en toute sécurité, et c'est apparemment exact (est-on jamais sûr de quelque chose en matière d'informatique ?). Tris rapides, sélection sur des critères numériques ou alphanuméri-

ques, possibilité de créer jusqu'à 40 rubriques. La souris existe et elle le prouve : copier, couper, coller, effacer, rechercher ou remplacer des mots, les éditer (sous forme de liste, de pages ou d'étiquettes)... On se croirait sur un Macintosh... mais avec quelques dizaines de milliers de francs en moins. C'est un constat fort agréable en vérité !

Des options intéressantes Je ne vais pas, dans le cadre de cette courte étude, passer en revue les différents menus déroulants de Version Liste. Je vous laisse le soin de les découvrir chez votre concessionnaire Apple. Toutefois, je m'en voudrais de ne point parler du menu "formats". Non content de mettre à votre disposition des fonctions de présentation (cadrer à droite ou à gauche, centrer), il offre des facilités assez inattendues :

- **Entier** convertit en nombres entiers les nombres contenus dans la colonne sélectionnée et les cadre systématiquement à droite de cette colonne. Toutefois, si la colonne contient des chaînes alphanumériques et numériques, seules les secondes resteront affichées et cadrées à droite.
- **Décimales** permet d'avoir deux chiffres après la virgule. Là aussi, le texte disparaît... mais peut être rappelé (voir ci-dessus).
- **Pourcentage** ajoute le signe % à la fin de chaque saisie de la colonne sélectionnée.
- **Francs** ajoute F à la fin de chaque nombre (vous vous en doutez un peu, non ?).

Et il est toujours possible, avec l'option alphanumérique de retrouver la colonne sous sa forme initiale.

Le logiciel transmettra éventuellement vos fichiers vers les autres utilitaires de Version Soft : Version Calc, Version Texte et Epistole.

Mention TRÈS BIEN pour Arnaud Lerondeau et Alban Liger, des Elèves de l'Ecole Supérieure d'Informatique et mention BIEN pour Pascale Couderc, la rédactrice du manuel. C. R.

CADRE-SOURIS

avec les nouveaux Apple IIe et IIc

POURQUOI ne pas utiliser les caractères souris des nouveaux Apple pour tracer un cadre (correct) en mode texte ? C'est ce que fait la routine LM que je vous propose ici. Elle peut évidemment être raccourcie (les commandes peuvent être données à partir du Basic). Taper la démo ci-contre ne vous fatiguera ni le poignet ni le bout des doigts. A vous de jouer !

CDRSRS

```

100 HOME : PRINT CHR$(4)"BLOAD CADRES
      OURIS"
110 CALL 768
120 GET R$
130 TEXT : HOME
140 VTAB 22: PRINT "<1> ENCORE <2> MEN
      U DISQUETTE <3> FIN S";: GET R$: P
      RINT
150 IF R$ = "1" THEN 110
160 IF R$ = "2" THEN VTAB 20: CALL - 9
      58: PRINT "TAPER RUN MENU": CALL -
      1438
170 IF R$ < > "3" THEN 140
180 HOME
  
```

CADRESOURIS

| | | | |
|-------|----------|-----|----------|
| 0300- | A9 03 | LDA | £\$03 |
| 0302- | 20 95 FE | JSR | \$FE95 |
| 0305- | A9 11 | LDA | £\$11 |
| 0307- | 20 ED FD | JSR | \$FDED |
| 030A- | A9 1B | LDA | £\$1B |
| 030C- | 20 ED FD | JSR | \$FDED |
| 030F- | 20 80 FE | JSR | \$FE80 |
| 0312- | A0 27 | LDY | £\$27 |
| 0314- | A9 4C | LDA | £\$4C |
| 0316- | 91 28 | STA | (\$28),Y |
| 0318- | 88 | DEY | |
| 0319- | 10 FB | BPL | \$0316 |
| 031B- | A0 00 | LDY | £\$00 |
| 031D- | A9 5A | LDA | £\$5A |
| 031F- | 91 28 | STA | (\$28),Y |
| 0321- | A0 27 | LDY | £\$27 |
| 0323- | A9 5F | LDA | £\$5F |
| 0325- | 91 28 | STA | (\$28),Y |
| 0327- | 20 66 FC | JSR | \$FC66 |
| 032A- | A5 25 | LDA | \$25 |
| 032C- | C9 17 | CMP | £\$17 |
| 032E- | 90 EB | BCC | \$031B |

INITIALISATION

\$FE95 (avec A = \$3) correspond à PR£3.

Ici, il s'agit de PRINT CHR\$(17) — 40 colonnes.

PRINT CHR\$(27) — CARACTÈRE SOURIS

SETINV : affichage en mode inverse.

CADRE HAUT

\$4C = L qui représente le souligné. Procédé classique avec adresse de la ligne lue en \$28-29 — HTAB est contenu dans Y.

BORDS GAUCHE ET DROIT

\$5A = Z qui représente le filet vertical gauche

\$5F = — qui représente le filet vertical droit

On poque d'abord le bord gauche, puis le bord droit (Y = 0, puis Y = \$27).

\$FC66 (LF) envoie un retour ligne

| | | | |
|-------|----------|-----|----------|
| 0330- | A9 D0 | LDA | £\$D0 |
| 0332- | 85 28 | STA | \$28 |
| 0334- | A0 26 | LDY | £\$26 |
| 0336- | A9 4C | LDA | £\$4C |
| 0338- | 91 28 | STA | (\$28),Y |
| 033A- | 88 | DEY | |
| 033B- | D0 FB | BNE | \$0338 |
| 033D- | A9 18 | LDA | £\$18 |
| 033F- | 20 ED FD | JSR | \$FDED |
| 0342- | A9 02 | LDA | £\$02 |
| 0344- | 85 25 | STA | \$25 |
| 0346- | 20 84 FE | JSR | \$FE84 |
| 0349- | 85 22 | STA | \$22 |
| 034B- | 85 20 | STA | \$20 |
| 034D- | A9 15 | LDA | £\$15 |
| 034F- | 85 23 | STA | \$23 |
| 0351- | A9 24 | LDA | £\$24 |
| 0353- | 85 21 | STA | \$21 |
| 0355- | 4C 6F F2 | JMP | \$F26F |

BASE DU CADRE

On place \$D0 dans \$28 (\$29 contient déjà \$7)... et on a bien \$7D0 dans \$28-29, ce qui permet de poquer le dernier bord du cadre.

TERMINÉ

PRINT CHR\$(24), pour les caractères normaux.
VTAB 02 puis mode normal avec \$FE84.

POKE 34,2 — POKE 32,2 puis POKE 35,21 — POKE 33,36.
Sur votre Apple IIc, terminez plutôt par 4C 22 FC qui placera le curseur au bon endroit.

BASIC SIMPLE

DOS 3.3

ProDOS

```

100 TEXT : HOME
110 DIM L$(38),M$(38)
120 VTAB 18: PRINT "ENTREZ UN TITRE (38 CARACTERES MAXI)"
130 VTAB 20: INPUT "":T$: HOME
140 IF T$ = "" THEN 280
150 L = LEN (T$): IF L > 38 THEN PRINT : PRINT "TROP LONG:
    PAS PLUS DE 38 CARACTERES!": GOTO 120
160 GOSUB 260
170 FOR I = L TO 1 STEP - 1:J = INT ( RND (1) * I + 1):M$(
    I) = L$(J):L$(J) = L$(I): NEXT
180 GOSUB 260
190 H = INT ((41 - L) / 2) - 1
200 FOR I = 1 TO L
210 FOR J = L TO 1 STEP - 1: IF L$(J) = M$(I) THEN M$(I) =
    "": GOTO 230
220 NEXT J
230 VTAB 3: HTAB H + J: INVERSE : PRINT L$(J): NORMAL :L$(
    J) = ""
240 GOSUB 270: NEXT I
250 GOTO 280
260 FOR I = 1 TO L:L$(I) = MID$( T$,I,1): NEXT : RETURN
270 FOR K = 1 TO 6:X = PEEK (49200): NEXT K: RETURN
280 VTAB 22: PRINT "<E>NCORE <M>ENU DISQUETTE <T>ERMINE $"
    : GET R$
285 PRINT R$: IF R$ = "E" THEN RUN
290 HOME : IF R$ = "M" THEN PRINT CHR$(4)"RUN MENU"
295 IF R$ < > "T" THEN 280

```

Titre mystérieux

Ce n'est ni génial ni très original, mais ce court programme en Basic a le mérite de montrer aux débutants comment mélanger aléatoirement les lettres d'une phrase, puis les remettre, en ordre en les affichant d'une manière un peu mystérieuse.

De telles routines nous sont souvent demandées par nos ami(e)s enseignant(e)s.

Merci à Michel Delacroix d'avoir pensé à eux !

Rideau

Trois petites routines en une :

- Remplissage de l'écran (de \$303 à \$318) ;
- Lever de rideau (de \$31D à \$334) ;
- Ouvrez les rideaux (de \$335 à \$362).

Commencez par taper le court programme de démonstration en Basic.

La formule $Y = \text{ABS}(Y - 1)$ donne alternativement la valeur 1 ou 0 à Y. Celle-ci est poquée à l'adresse \$6 (voir la partie **CHOIX** de la routine LM... et la ligne 150 de **RIDBAS**).

RIDEAU

| | | | |
|-------|----------|-----|-----------|
| 0300- | 20 58 FC | JSR | \$FC58 |
| 0303- | A9 20 | LDA | £\$20 |
| 0305- | A0 27 | LDY | £\$27 |
| 0307- | 91 28 | STA | (\$28), Y |
| 0309- | 88 | DEY | |
| 030A- | 10 FB | BPL | \$0307 |
| 030C- | E6 25 | INC | \$25 |
| 030E- | A5 25 | LDA | \$25 |
| 0310- | C9 18 | CMP | £\$18 |
| 0312- | B0 05 | BCS | \$0319 |
| 0314- | 20 22 FC | JSR | \$FC22 |
| 0317- | 90 EA | BCC | \$0303 |
| 0319- | A5 06 | LDA | \$06 |
| 031B- | D0 18 | BNE | \$0335 |
| 031D- | C6 25 | DEC | \$25 |
| 031F- | 20 1A FC | JSR | \$FC1A |
| 0322- | A9 A0 | LDA | £\$A0 |
| 0324- | A0 26 | LDY | £\$26 |
| 0326- | 91 28 | STA | (\$28), Y |
| 0328- | 88 | DEY | |

RIDBAS

```

100 PRINT CHR$(4)"BLOD RIDEAU"
110 HOME : VTAB 12: PRINT "ESCAPE POUR
    TERMINER LA DEMONSTRATION": GOTO
    130
120 CALL 768
130 CALL - 198: POKE - 16368,0: WAIT -
    16384,128,127: POKE - 16368,0:X =
    PEEK ( - 16384)
140 Y = ABS (Y - 1)
150 POKE 6,Y: IF X < > 27 THEN 120
160 POKE 34,10: POKE 32,10: POKE 33,20
    : POKE 35,20
170 VTAB 10: PRINT
180 PRINT : PRINT "<1> ENCORE": PRINT
    : PRINT "<2> MENU DISQUETTE": PRIN
    T : PRINT "<3> TERMINE ";
190 GET R$:R = ASC (R$) - 48: IF R < 1
    OR R > 3 THEN 190
200 POKE 35,24: POKE 32,0: POKE 33,40:
    POKE 34,0
210 PRINT : ON R GOTO 220,230,240
220 GOTO 120
230 PRINT CHR$(4)"RUN MENU"
240 HOME
  
```

HOME traditionnel.

REPLISSAGE DE L'ÉCRAN

HOME fait que CV (\$25) contient au départ la valeur 0 (ligne 1 de l'écran). Le registre Y est chargé avec \$27 (39). On lit l'adresse de base de la ligne en \$28-29. On poque un espace en mode inverse (\$20) à l'adresse lue en \$28-29 plus Y. La suite est facile à comprendre. Notez que \$FC22 déplace le curseur en CV (\$25).

CHOIX

Si le pointeur \$6 est différent de 0, on saute à la routine "Ouvrez les rideaux".

LEVER DE RIDEAU

On était à la dernière ligne. Donc on décrémente CV (\$25), puis \$FC1A monte le curseur. Cette fois, il s'agit d'effacer avec l'espace normal...

| | | | |
|-------|----------|-----|----------|
| 0329- | D0 FB | BNE | \$0326 |
| 032B- | 20 A8 FC | JSR | \$FCA8 |
| 032E- | A5 25 | LDA | \$25 |
| 0330- | C9 01 | CMP | £\$01 |
| 0332- | D0 EB | BNE | \$031F |
| 0334- | 60 | RTS | |
| 0335- | A9 13 | LDA | £\$13 |
| 0337- | 85 06 | STA | \$06 |
| 0339- | 85 07 | STA | \$07 |
| 033B- | E6 07 | INC | \$07 |
| 033D- | C6 25 | DEC | \$25 |
| 033F- | 20 1A FC | JSR | \$FC1A |
| 0342- | A9 A0 | LDA | £\$A0 |
| 0344- | A4 06 | LDY | \$06 |
| 0346- | 91 28 | STA | (\$28),Y |
| 0348- | A4 07 | LDY | \$07 |
| 034A- | 91 28 | STA | (\$28),Y |
| 034C- | A5 25 | LDA | \$25 |
| 034E- | C9 01 | CMP | £\$01 |
| 0350- | D0 ED | BNE | \$033F |
| 0352- | 8A | TXA | |
| 0353- | 20 A8 FC | JSR | \$FCA8 |
| 0356- | E6 07 | INC | \$07 |
| 0358- | C6 06 | DEC | \$06 |
| 035A- | F0 06 | BEQ | \$0362 |
| 035C- | A9 17 | LDA | £\$17 |
| 035E- | 85 25 | STA | \$25 |
| 0360- | D0 DD | BNE | \$033F |
| 0362- | 60 | RTS | |

et sans toucher aux bords de l'écran (de 01 à \$26) WAIT (\$FCA8) donne le temps de voir le rideau se lever... sinon ce serait trop rapide.
Quand CV (\$25) contient 01, on a terminé

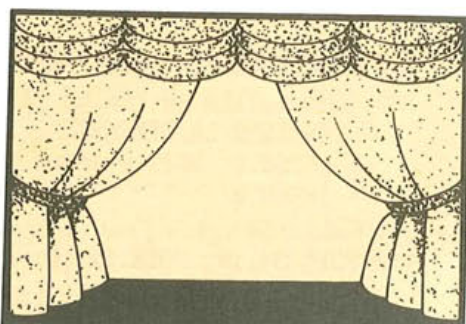
OUVREZ LES RIDEAUX

\$6 contient la position du rideau gauche et \$7 celle du rideau droit... et c'est parti. On efface successivement chaque rangée verticale, à gauche d'abord, puis à droite.

Là encore, il faut utiliser WAIT (qui est fonction de la valeur de A. Celle-ci est obtenue en passant X dans A (TXA)).

Le résultat est moins heureux, à nos yeux que celui de LEVER DE RIDEAU. On y note un effet de déchirure.

Il faudrait effacer **plus rapidement** chacune des rangées verticales de l'écran...



Retour au BASIC.

BSAVE RIDEAU, A\$300, L\$63

QUESTION :

Comment écrire, en langage machine et sous DOS 3.3, un programme Basic comme celui-ci ?

```
10 GET R$
20 IF R$ <> "C" THEN 60
30 HOME
40 PRINT
50 PRINT CHR$(4)
  "CATALOG"
60 RETURN
```

Michel DELETANG (ANNECY).

RÉPONSE :

| | | | | |
|-------|----------|----------|----------|--------------------------|
| 0300- | 20 0C FD | JSR | \$FD0C | GET R\$ |
| 0303- | C9 C3 | CMP | £\$C3 |] IF R\$ <> "C" THEN ... |
| 0305- | D0 11 | BNE | \$0318 | |
| 0307- | 20 58 FC | JSR | \$FC58 | HOME |
| 030A- | 20 8E FD | JSR | \$FD8E | PRINT OBLIGATOIRE |
| 030D- | A2 08 | LDX | £\$08 |] PRINT CHR\$(4) |
| 030F- | BD 19 03 | LDA | \$0319,X | |
| 0312- | 20 ED FD | JSR | \$FDED | "CATALOG" |
| 0315- | CA | DEX | | |
| 0316- | 10 F7 | BPL | \$030F | |
| 0318- | 60 | RTS | | |
| 0319- | 8D C7 CF | RETURN | G O | Tout est |
| 031C- | CC C1 D4 | L A T | | lu à |
| 031F- | C1 C3 | A C | | l'envers |
| 0321- | 84 | CHR\$(4) | | |

Effets spéciaux

l'écran se remplit, mais en colimaçon !

AMUSANTS... ces remplissages successifs de l'écran par des astérisques (avec effet de scintillement). Vous les stopperez en appuyant sur la touche ESCAPE... et obtiendrez alors un cadre !

SPIRBAS

```

100 PRINT CHR$(4)"BLOAD SPIRECADRE"
105 HOME : VTAB 12: PRINT "ESCAPE POUR
    TERMINER LA DEMONSTRATION": POKE
    - 16368,0: WAIT - 16384,128: POKE
    - 16368,0
110 CALL 768
115 POKE 34,10: POKE 32,10: POKE 33,20
    : POKE 35,20
120 VTAB 10: PRINT
125 PRINT : PRINT "<1> ENCORE": PRINT
  
```

```

: PRINT "<2> MENU DISQUETTE": PRIN
T : PRINT "<3> TERMINE ";
130 GET R$:R = ASC (R$) - 48: IF R < 1
    OR R > 3 THEN 130
135 POKE 35,24: POKE 32,0: POKE 33,40:
    POKE 34,0
140 PRINT : ON R GOTO 145,150,155
145 GOTO 110
150 PRINT CHR$(4)"RUN MENU"
155 HOME
  
```

La routine LM commence à partir de l'adresse \$2F4, c'est-à-dire dans le BUFFER D'ENTRÉE. Au début d'un programme, il n'est pas interdit d'utiliser cette mémoire disponible (et qui le restera si aucun INPUT ne dépasse l'adresse \$2F4). Vous pourriez même — en modifiant les adresses qui contiennent \$2F4 — loger la totalité de SPIRECADRE dans le BUFFER. Essayez ! Comment ? **BLOAD SPIRECADRE, A\$27C**. Ensuite **CALL — 151**, puis tapez les valeurs suivantes : *2A3:7C N 2B2:7C N 2BE:7C N 2CE:7C

Pour terminer, remplacez le **CALL 768** de la ligne 110 par un **CALL 648 (636+12)**.

SPIRECADRE

| | | | |
|-------|----------|-----|----------|
| 02F4- | A9 20 | LDA | £\$20 |
| 02F6- | 91 28 | STA | (\$28),Y |
| 02F8- | 20 A8 FC | JSR | \$FCA8 |
| 02FB- | A9 AA | LDA | £\$AA |
| 02FD- | 91 28 | STA | (\$28),Y |
| 02FF- | 60 | RTS | |
| 0300- | 20 58 FC | JSR | \$FC58 |
| 0303- | A9 1B | LDA | £\$1B |
| 0305- | 85 09 | STA | \$09 |
| 0307- | A9 0C | LDA | £\$0C |

SOUS-PROGRAMME "POKE"

On lit en \$28-29 l'adresse de la ligne en cours.
Y = HTAB — \$FCA8 est une boucle d'attente (sa longueur dépend de la valeur de A et, ici, A = \$20).
On POKE un espace en mode inverse (\$20), puis l'astérisque (\$AA) en mode normal.

INITIALISATION

HOME traditionnel.
On utilise les adresses \$6 et \$9 de la page zéro comme pointeurs.

| | | | |
|-------|----------|-----|-----------|
| 0309- | 85 08 | STA | \$08 |
| 030B- | 85 07 | STA | \$07 |
| 030D- | A9 0B | LDA | £\$0B |
| 030F- | 85 06 | STA | \$06 |
| 0311- | 85 25 | STA | \$25 |
| 0313- | 20 22 FC | JSR | \$FC22 |
| 0316- | A4 07 | LDY | \$07 |
| 0318- | E6 09 | INC | \$09 |
| 031A- | 20 F4 02 | JSR | \$02F4 |
| 031D- | C4 09 | CPY | \$09 |
| 031F- | F0 03 | BEQ | \$0324 |
| 0321- | C8 | INY | |
| 0322- | D0 F6 | BNE | \$031A |
| 0324- | 20 66 FC | JSR | \$FC66 |
| 0327- | A4 09 | LDY | \$09 |
| 0329- | 20 F4 02 | JSR | \$02F4 |
| 032C- | A5 25 | LDA | \$25 |
| 032E- | C5 08 | CMP | \$08 |
| 0330- | D0 F2 | BNE | \$0324 |
| 0332- | C6 07 | DEC | \$07 |
| 0334- | 88 | DEY | |
| 0335- | 20 F4 02 | JSR | \$02F4 |
| 0338- | C4 07 | CPY | \$07 |
| 033A- | D0 F8 | BNE | \$0334 |
| 033C- | E6 08 | INC | \$08 |
| 033E- | C6 06 | DEC | \$06 |
| 0340- | C6 25 | DEC | \$25 |
| 0342- | 20 22 FC | JSR | \$FC22 |
| 0345- | 20 F4 02 | JSR | \$02F4 |
| 0348- | A5 25 | LDA | \$25 |
| 034A- | C5 06 | CMP | \$06 |
| 034C- | D0 F2 | BNE | \$0340 |
| 034E- | AD 00 04 | LDA | \$0400 |
| 0351- | C9 A0 | CMP | £\$A0 |
| 0353- | F0 C1 | BEQ | \$0316 |
| 0355- | AD 00 C0 | LDA | \$C000 |
| 0358- | C9 9B | CMP | £\$9B |
| 035A- | D0 A4 | BNE | \$0300 |
| 035C- | 2C 10 C0 | BIT | \$C010 |
| 035F- | A9 17 | LDA | £\$17 |
| 0361- | 85 25 | STA | \$25 |
| 0363- | 20 1A FC | JSR | \$FC1A |
| 0366- | A0 01 | LDY | £\$01 |
| 0368- | A9 A0 | LDA | £\$A0 |
| 036A- | 91 28 | STA | (\$28), Y |
| 036C- | C8 | INY | |
| 036D- | C0 27 | CPY | £\$27 |
| 036F- | D0 F9 | BNE | \$036A |
| 0371- | A5 25 | LDA | \$25 |
| 0373- | C9 01 | CMP | £\$01 |
| 0375- | D0 EC | BNE | \$0363 |
| 0377- | 60 | RTS | |

\$7 = Limite HTAB gauche — \$9 = Limite HTAB droite —
\$6 = Limite VTAB sup. — \$8 = Limite VTAB inf.

HORIZONTALEMENT (gauche à droite)

\$FC22 (VTAB) déplace le curseur en CV (\$25). L'adresse de la ligne, calculée par BASCALC est en \$28-29.

Y est chargé avec Limite HTAB gauche (\$7) et Limite HTAB droite (\$9) est incrémentée.

Quand Y est égal à \$9, on sort de la boucle.

VERTICALEMENT (du haut vers le bas)

\$FC66 (LF) fait passer à la ligne suivante — Y est chargé avec Limite HTAB droite (\$9) — Si la valeur de VTAB (\$25) n'est pas égale à Limite VTAB INF (\$8), on continue.

HORIZONTALEMENT (droite à gauche)

Même processus que plus haut, mais à reculons.

A la sortie, ne pas oublier les pointeurs pour la suite des opérations.

VERTICALEMENT (de bas en haut)

Là encore, même combat que plus haut.

TEST FIN

Si l'octet \$400 est toujours un espace (\$A0), on n'a pas terminé la page !

TEST ESCAPE

Si la touche ESCAPE n'a pas été enfoncée, on continue. Ça, c'est un POKE — 16368,0.

CADRE

VTAB (\$25) est chargé avec \$17 (23... dernière ligne), puis \$FC1A (VP) remonte le curseur à la ligne au-dessus. Y est chargé avec 1 et A avec \$A0 (espace)... et on efface la ligne... puis une autre, et ainsi de suite, jusqu'à la ligne n°1. Finalement, il ne reste que le cadre.

Les non-initié se contenteront de taper un CALL — 151, puis les différents codes du programme. Exemple :
*2F4: A9 20 91 28 20 A8 FC... return
*2FB: A9 AA 91 28 60... return... etc.
Pour terminer : BSAVE SPIRECADRE, \$A2F4, £\$84 ou BSAVE SPIRECADRE, A756, L132.

Retour au Basic.

Inversion d'écran

double haute résolution

Le succès du concept MAC INTOSH, avec ses menus déroulants, sa très haute résolution, la souris, etc. oblige en quelque sorte l'APPLE IIe à se mettre à la page, et pour cela il faut avoir recours à la double haute résolution. Malheureusement on trouve encore peu d'articles présentant des routines ou des utilitaires tirant partie des cartes CHAT MAUVE et EVE, pour ne citer qu'elles. Aussi j'espère que cette routine d'inversion d'écran vous inspirera. Une chose est sûre : elle sera très utile dans votre bibliothèque de routines graphiques.

PRINCIPE DU PROGRAMME :

L'algorithme d'inversion d'un octet est relativement simple, puisqu'il revient à compléter les bits du dit octet. Pour cela, le microprocesseur dispose d'une instruction efficace : EOR, qui réalise un "OU exclusif" de la valeur contenue dans l'accumulateur avec la valeur de VAL. Par exemple, si l'on a VAL=\$FF et \$05 dans l'accumulateur (soit, en binaire les valeurs 11111111 et 00000101), un "OU exclusif" changera la valeur \$05 en \$FA soit 11111010.

En mode double haute résolution, l'image est en quelque sorte un entrelassé d'octets.

La répartition de la mémoire est similaire à celle employée pour la haute résolution standard. C'est-à-dire que la première adresse de l'image commence en \$2000 (8192). Un bit mis à un indique qu'un point est allumé, un bit à zéro indique que ce point est éteint. Un octet représente une série de sept points sur l'écran. Pourquoi sept et pas huit ? Parce que, dans chaque octet, le bit de poids fort est ignoré. Un octet correspond donc à sept points, allumés ou non. Jusque-là rien de nouveau diront certains, le petit plus, c'est pour maintenant !

L'entrelassé se fait entre deux zones de la mémoire, les pages 1 et 1X. Ces pages ont les mêmes adresses (\$2000 à \$3FFF), mais la page 1 se trouve en mémoire principale alors que la page 1X se trouve en mémoire auxiliaire. L'affichage d'une image revient à visualiser 192 lignes de 40 paires de 14 points chacune.

Les octets d'ordre pair sont stockés en mémoire

auxiliaire (page 1X), les octets d'ordre impair sont en mémoire principale, mais ils ont la même adresse que ceux de la page 1X.

Pour une ligne, on a donc la séquence suivante : Points de 0 à 6, 14 à 20... 545 à 552 en page 1X, points de 7 à 13, 21 à 27... 553 à 559 sont en page 1.

Le programme devra donc inverser les octets des pages 1 et 1X.

L'accès en écriture sur la page 1X (mémoire auxiliaire) est rendu possible en positionnant des commutateurs logiciels présents dans la mémoire principale de l'APPLE. On utilisera 4 commutateurs :

| | |
|---|----------------|
| AN3 (C05E) doit être débranché. | 80STORE (C001) |
| 80COL. 80STORE et AN3 peuvent être correctement positionnés par un PR£3 | HIRES (C057) |
| | PAGE1 (C054) |
| | PAGE1X (C055) |

Le branchement des deux commutateurs HIRES et 80STORE est nécessaire pour modifier le rôle de PAGE1 et 1X. PAGE1 et PAGE1X décident si l'on commute la mémoire principale ou la mémoire auxiliaire.

La routine en assembleur a été écrite avec BIG MAC, elle prend un octet en mémoire principale l'inverse sur sept bits, donc avec une valeur \$7F, puis commute la page 1X, inverse l'octet et recommence jusqu'à la fin de la ligne. Elle calcule alors l'adresse de la nouvelle ligne à inverser, et ainsi de suite.

COMMENT UTILISER CETTE ROUTINE :

Le programme ci-après, écrit en BASIC, sert à positionner les commutateurs double haute résolution graphique, puis il dessine un cercle, un rectangle, inverse le cercle, le rectangle, puis tout l'écran et boucle sur lui-même.

Pour utiliser cette routine à partir du BASIC, il faut être en mode double haute résolution (560 X 192 points) et avoir défini une zone à inverser dont les valeurs extrêmes seront stockées dans des variables, elles-mêmes "pokées" à partir de l'adresse \$5FF0 (24560). Se reporter au listage BASIC. Cette routine est relogeable mais il faudra modifier les adresses \$5FF0 à \$5FF5 en conséquence.

Thierry GAUTHIER.

```

10 PRINT CHR$(4)"PRF3": PRINT :
   VTAB 10: HTAB 10: PRINT "UN INS
   TANT S.V.P....."
11 PRINT CHR$(4)"BLOADINVERT.SYS
   ": REM ROUTINE CHARGEE EN $6000
12 A = PEEK (49246): REM ANNONCIA
   TEUR 3 DEBRANCHE
13 A = PEEK (49232): REM MODE TEX
   TE A ZERO (GR)
14 A = PEEK (49239): REM MODE HAU
   TE RESOLUTION "ON"
15 A = PEEK (49165): REM AFF. 80C
   OLONNES "ON"
16 POKE 49153,0: REM 80STORE "ON"
   ECRITURE EN MEV-AUX
17 POKE 49156,0: REM RAMWRT ECRITU
   RE EN MEMOIRE PRINCIPALE
18 POKE 49154,0: REM RAMRD LECTURE
   EN MEMOIRE PRINCIPALE
19 POKE 49236,0: REM PAGE2 A ZERO
   = ACCES MEV//E
30 HGR : POKE 49237,0: REM PAGE2 A
   UN = ACCES MEV-AUX, ON EFFACE
   L'ECRAN
32 ADR = 24560:INVERT = 24576: REM
   ADRESSES $5FF0 ET $6000 DEBUT D
   E INVERT.SYS
35 PRINT CHR$(4)"PRF3"
40 CALL 62450: REM EFFACE L'ECRAN
   AVEC LA DERNIERE COULEUR UTILIS
   EE
41 POKE 49236,0: POKE 49234,0: REM
   ACCES MEV//E ET FULL HIRES
60 REM DESSIN DU CERCLE
65 HCOLOR= 3
70 FOR J = 0 TO 7 STEP 0.5 / 10
80 X = 270 + 2 * 10 * COS (J):B =
   96 + 10 * SIN (J)
83 GOSUB 250: REM S/P SELECTION DE
   S POINTS PAIRS ET IMPAIRS EN ME
   V//E OU EN MEV-AUX
85 NEXT J
90 REM DESSIN DU RECTANGLE
92 FOR X = 50 TO 100:B = 40: GOSUB
   250: NEXT X: REM UNE HORIZONTAL
   E
93 FOR B = 40 TO 55:X = 50: GOSUB
   250: NEXT B: REM UNE VERTICALE
94 FOR X = 50 TO 100:B = 55: GOSUB
   250: NEXT X: REM 2EME HORIZONTAL
   E
95 FOR B = 40 TO 55:X = 100: GOSUB
   250: NEXT B: REM LE DERNIER COT
   E

```

7F69
BED1
204D
B648
154F
344D
DF1B
091E
FC1C
1A1D
22E9
0492
8B62
4F8D
EE72
51C5
C449
1351
E747
A3CC
88A5
5C68
12AB
5994

```

109 REM INVERSE LE ROND
110 XB = 50: REM LONGUEUR EN X
120 YB = 25: REM LONGUEUR EN Y
130 CX = 245: REM PREMIER POINT A I
   NVERSER EN X
140 CY = 83: REM PREMIER POINT A IN
   VERSER EN Y
150 GOSUB 400: FOR T = . TO 200: NE
   XT T: GOSUB 400
151 REM INVERSE LE RECTANGLE
152 XB = 56:YB = 21:CX = 47:CY = 37
   : GOSUB 400: FOR T = 1 TO 200:
   NEXT T: GOSUB 400
160 REM INVERSE TOUT L'ECRAN
170 XB = 559:YB = 191:CX = 0:CY = 0
180 GOSUB 400 REM S/P INVERSION

```

Vous pourriez insérer ici une ligne comme celle-ci (il suffira alors de maintenir la touche POMME OUVERTE enfoncée... pour terminer la démo) :

```

184 IF PEEK (49249) > 127 THEN TEXT:
   HOME: END: REM POMME OUVERTE
185 GOTO 109
249 REM S/P PARTAGE DES POINTS A AF
   FICHER EN MEV//E ET MEV-AUX.
250 XX = INT (X / 7):PG = XX - 2 *
   INT (XX / 2)
260 XX = INT (XX / 2) * 7 + X - 7
   * XX
270 POKE 49237,0: IF PG THEN POKE
   49236,0
280 IF XX > 279 THEN RETURN
300 HPLLOT XX,B
310 POKE 49236,0: RETURN
400 REM S/P SEPARER ADRESSE PARTIE H
   AUTE ET PARTIE BASSE
410 SX = XB:SY = YB + 1:EX = CX
420 POKE ADR,EX - INT (EX / 256) *
   256: REM LGXL
430 POKE ADR + 1, INT (EX / 256): R
   EM LGXH
440 POKE ADR + 2,(EX + SX) - INT (
   (EX + SX) / 256) * 256: REM FIN
   DE LA LIGNE DANS EXL
450 POKE ADR + 3, INT ((EX + SX) /
   256): REM PARTIE HAUTE EXH
460 POKE ADR + 4,CY: REM LONGUEUR E
   N Y DANS LGY
470 POKE ADR + 5,CY + YB: REM FININ
   V = DERNIERE LIGNE A INVERSER
480 CALL INVERT + 7: REM APPEL INVE
   RT.SYS
490 RETURN

```

(suite page 44)

```

1      ORG $6000
2      ASSEMBLEUR BIG MAC
3
4
5      ROUTINE D'INVERSION
6      D'ÉCRAN
7      DOUBLE HAUTE
8      RÉOLUTION
9
10     INVERT.SYS
11
12     *
13     LGXL      =   $5FF0
14     LGXH      =   $5FF1
15     EXL       =   $5FF2
16     EXH       =   $5FF3
17     LGY       =   $5FF4
18     FININV    =   $5FF5
19     HPAG      =   $E6
20     GBASL     =   $26
21     GBASH     =   $27
22     *
23     PAGE1X    =   $C055
24     PAGE1     =   $C054
25     *
26     VARIABLES DE TRANSFERT
27     *
28     LXL       DS   1
29     LXH       DS   1
30     XL        DS   1
31     XH        DS   1
32     CY        DS   1
33     N2        DS   1
34     MASK      DS   1
35     *
36     DEBUT DU PROGRAMME
37     *
38         LDA   LGXL
39         LDY   LGXH
40         JSR   OCTETS
41         STX   LXL
42         STY   LXH
43         LDA   EXL
44         LDY   EXH
45         JSR   OCTETS
46         STX   XL
47         STY   XH
48         LDA   LGY
49         STA   CY
50     LIGNE1 JSR   CALCADR
51         LDY   LXL
52         LDX   LXH
53         LDA   LTABLE,X
54         STA   MASK

```

```

55     LIGNE   STY   N2
56         PHA
57         TYA
58         LSR
59         TAY
60         PLA
61         PHP
62         SEI
63         BIT   PAGE1
64         BCS   INV1
65         BIT   PAGE1X
66     INV1    LDA   (GBASL),Y
67         EOR   MASK
68         STA   (GBASL),Y
69         BIT   PAGE1
70         PLP
71         LDA   £$7F
72         STA   MASK
73         LDY   N2
74         INY
75         CPY   XL
76         BCC   LIGNE
77         PHP
78         LDX   XH
79         LDA   HTABLE,X
80         STA   MASK
81         PLP
82         BEQ   LIGNE
83         INC   CY
84         LDA   CY
85         CMP   FININV
86         BCC   LIGNE1
87     FINV    RTS
88     *
89     S/P INVERSE
90     *
91     OCTETS  LDX   £$FF
92     INV3    SEC
93     INV2    INX

```

```

94         SBC   £$07
95         BCS   INV2
96         DEY
97         BPL   INV3
98         AND   £$07
99         TAY
100        DEY
101        RTS
102     *
103     CALCUL ADRESSE
104     *
105     CALCADR LDA   CY
106         PHA
107         AND   £$C0
108         STA   GBASL
109         LSR
110         LSR
111         ORA   GBASL
112         STA   GBASL
113         PLA
114         STA   GBASH
115         ASL
116         ASL
117         ASL
118         ROL   GBASH
119         ASL
120         ROL   GBASH
121         ASL
122         ROR   GBASL
123         LDA   GBASH
124         AND   £$1F
125         ORA   HPAG
126         STA   GBASH
127         RTS
128     *
129     TABLE D'ADRESSES
130     *
131     LTABLE  HEX   7F7E7C78706040
132     HTABLE  HEX   0103070F1F3F7F

```

```

6000: 00 00 00 00 00 00 00 AD F0 5F AC F1 5F 20 7E 60  ££AAF6
6010: 8E 00 60 8C 01 60 AD F2 5F AC F3 5F 20 7E 60 8E  ££0963
6020: 02 60 8C 03 60 AD F4 5F 8D 04 60 20 8E 60 AC 00  ££96FC
6030: 60 AE 01 60 BD B3 60 8D 06 60 8C 05 60 48 98 4A  ££AA4D
6040: A8 68 08 78 2C 54 C0 B0 03 2C 55 C0 B1 26 4D 06  ££EEEE
6050: 60 91 26 2C 54 C0 28 A9 7F 8D 06 60 AC 05 60 C8  ££5473
6060: CC 02 60 90 D5 08 AE 03 60 BD BA 60 8D 06 60 28  ££069E
6070: F0 C8 EE 04 60 AD 04 60 CD F5 5F 90 AE 60 A2 FF  ££147B
6080: 38 E8 E9 07 B0 FB 88 10 F7 29 07 A8 88 60 AD 04  ££91BB
6090: 60 48 29 C0 85 26 4A 4A 05 26 85 26 68 85 27 0A  ££C0C4
60A0: 0A 0A 26 27 0A 26 27 0A 66 26 A5 27 29 1F 05 E6  ££554D
60B0: 85 27 60 7F 7E 7C 78 70 60 40 01 03 07 0F 1F 3F  ££8385
60C0: 7F ££FE7F

```

BSAVE INVERT.SYS, A\$6000, L\$C1

La fenêtre de texte de la page HGR2

S I vous avez déjà utilisé la seconde page graphique haute résolution (HGR2), vous savez qu'elle est placée entre les adresses \$4000 et \$5FFF (16384-24575). Comme la page graphique haute résolution normale (HGR), elle se présente sous la forme de 280 points de large (40 colonnes de 7 points) et de 192 points de haut (24 lignes de 8 points).

On l'initialise par un banal HGR2 qui efface évidemment la zone-mémoire concernée, mais n'a aucune incidence sur le contenu de la page TEXTE (\$400-\$7FF).

Est-il possible d'écrire, comme en HGR, dans la fenêtre de texte du bas de l'écran (lignes 21 à 24) ? La réponse est oui, mais vous avez pu déplorer les conséquences d'un POKE — 16301,0 (qui ouvre cette fenêtre et provoque l'affichage — ô combien désagréable ! — de caractères que l'on peut qualifier de "bizarres"). En fait, vous visionnez alors une petite partie du programme Applesoft, généralement chargé à partir de l'adresse \$801... début de la page TEXTE 2.

Pour être en mesure de tirer le meilleur parti de ces quatre lignes, il convient donc, dans un premier temps, de déplacer le programme en Basic. C'est ce que fait la ligne 0 de notre première démonstration. Grâce à ce

subterfuge, le Basic sera rechargé à partir de l'adresse 24577 (\$6001). Notez l'indispensable POKE 24576,0. L'octet précédant un programme en Basic doit obligatoirement être à zéro.

Et maintenant, tapez et sauvez HGR4L.ES, puis lancez-le par le RUN traditionnel. Normalement, votre Apple rechargera les excellentes lignes de votre routine à partir de l'adresse 24577, passera en mode HGR2, effacera les quatre lignes de texte... et réécrira quatre fois la phrase-bidon dont vous allez bientôt prendre connaissance.

Les adresses des quatre lignes de la fenêtre sont en DATA, mais il est possible de les obtenir par une boucle ressemblant à celle-ci :

```
10 FOR I = 21 TO 24 : POKE 36,0: VTAB I : PRINT
   (PEEK(41) * 256 + PEEK(40)) + 1024: NEXT
```

Explication :

On retrouve l'adresse de base de la ligne en 41-40... mais comme on désire travailler en page 2, il faut y ajouter 1024.

Je vous propose, en supplément, une démonstration plus élaborée... et plus rapide car elle utilise une routine en langage machine. Si vous êtes courageux (ou courageuse), essayez-la aussi. Bonne programmation !

Clément RENARD.

HGR4L.ES

Comme vous l'avez lu plus haut, la ligne 0 permet de poquer en 103 et 104, la nouvelle adresse à laquelle sera LOADée la routine. ■

```
0 IF PEEK (104) < > 96 THEN POKE 104,96: POKE 103,1: POK
  E 24576,0: PRINT CHR$ (4)"RUN HGR4L.ES"
10 HGR2
20 POKE - 16301,0
30 FOR I = 1 TO 4: READ R: FOR J = R TO R + 39: POKE J,16
  0: NEXT : NEXT
40 T$ = "LES QUATRE LIGNES-TEXTE DE LA PAGE HGR 2"
50 RESTORE : FOR I = 1 TO 4: READ R:X = 0: FOR J = R TO R
  + 39:X = X + 1: POKE J, ASC ( MID$ (T$,X,1)) + 128: N
  EXT : NEXT
60 DATA 2640,2768,2896,3024
```

HGR2.DEMO4L

| | |
|--|------|
| 100 D\$ = CHR\$(4): TEXT : PRINT CHR\$(21): HOME | FDC7 |
| 105 IF PEEK (104) < > 96 THEN POKE 104,96: POKE 103,1: POKE 24576,0: PRINT D\$"RUN HGR2.DEMO4L" | F807 |
| 110 PRINT D\$"BLOAD HGR4L" | 8949 |
| 115 CALL 768 | 8331 |
| 120 VTABLE 1: GOSUB 220 | 0351 |
| 125 FOR I = 2000 TO 2039: POKE I,160: NEXT | DFA6 |
| 130 GOSUB 220: HOME | 1A15 |
| 135 PRINT "DEPART PROGRAMME: ";: FLASH : PRINT PEEK (104) * 256 + PEEK (103): NORMAL | 5B7F |
| 140 LIST 100,130 | 0COD |
| 145 GOSUB 215: PRINT : HOME | 500D |
| 150 VTABLE 9: PRINT "LA FENETRE OUVERTE EN BAS DE LA "PAGE HGR2 EFFACE 4 ZONES DE LAMEMOIRE": PR INT | 9311 |
| 155 X = X + 1: ON X GOTO 160,165,170,175,180 | 211B |
| 160 T\$ = "A50.A77 D823G": GOTO 225 | 78D5 |
| 165 T\$ = "AD0.AF7 D823G": GOTO 225 | DBF3 |
| 170 T\$ = "B50.B77 D823G": GOTO 225 | BFD7 |
| 175 T\$ = "BD0.BF7 D823G": GOTO 225 | DFF5 |
| 180 VTABLE 24: PRINT "ON RETROUVE ICI LA LIGNE DE TEXTE";: G OSUB 220: PRINT | 9FAA |
| 185 VTABLE 22: CALL - 958 | 583B |
| 190 GOSUB 235: POKE - 16297,0: POKE - 16299,0: POKE - 1 6301,0: POKE - 16304,0: CALL 822: GOSUB 220: TEXT | 475A |
| 195 VTABLE 24: PRINT "<1> MENU DISQUETTE <2> FIN DE TRAVAIL ";: GET R\$ | 5F86 |
| 200 IF R\$ < > "1" AND R\$ < > "2" THEN VTABLE 23: PRINT : GOTO 195 | D4D4 |
| 205 POKE 2048,0: POKE 103,1: POKE 104,8: IF R\$ = "1" THEN PRINT : PRINT D\$"RUN MENU" | 29BD |
| 210 HOME : NEW | 5090 |
| 215 VTABLE 22: INVERSE : PRINT "PRESSEZ UNE TOUCHE S.V.P.": NORMAL | 8FAC |
| 220 CALL - 198: POKE - 16368,0: WAIT - 16384,128,127: P OKE - 16368,0: PRINT : RETURN | 2861 |
| 225 FOR I = 1 TO LEN (T\$): POKE 511 + I, ASC (MID\$ (T\$,I ,1)) + 128: NEXT : POKE 72,0: CALL - 144 | 3ABA |
| 230 GOSUB 220: PRINT : GOTO 155 | 17B8 |
| <hr/> | |
| 235 POKE 230,64: HCOLOR= 3 | 86E3 |
| 240 X1 = 40:Y1 = 130:N = 5: GOSUB 260 | 97F4 |
| 245 X1 = 90:Y1 = 130:N = 7: GOSUB 260 | ACFB |
| 250 X1 = 180:Y1 = 70: GOSUB 260 | 486F |
| 255 X1 = 230:Y1 = 130: GOSUB 260: GOTO 265 | 5D1A |
| 260 FOR I = 1 TO N: READ X: READ Y: HPLLOT X1,Y1 TO X,Y:X1 = X:Y1 = Y: NEXT : RETURN | 4436 |
| 265 HPLLOT 250,50 TO 250,130: HPLLOT 260,50 TO 260,130 | ADE2 |
| 270 RETURN | 63B1 |
| 275 DATA 40,130,40,50,80,50,80,90,40,90,90,130,90,50,130,5 0,130,90,90,90,130,90,130,130 | 755B |
| 280 DATA 180,70,180,50,140,50,140,130,180,130,180,90,155,9 0,230,130,190,130,190,90,220,90,190,90,190,50,230,50 | A1CE |

Cette seconde démonstration est un peu plus élaborée et peut être intéressante à étudier..

La routine en langage machine.

Original, non ? mais attendez de connaître le résultat pour répondre à la question !

Cette partie du programme se contente d'écrire des caractères géants dans la page HGR2.

| | | | |
|-------|----------|-----|--------|
| 0300- | A9 20 | LDA | £\$20 |
| 0302- | 85 E6 | STA | \$E6 |
| 0304- | A9 00 | LDA | £\$00 |
| 0306- | AA | TAX | |
| 0307- | A8 | TAY | |
| 0308- | 20 F4 F3 | JSR | \$F3F4 |
| 030B- | 20 E2 F3 | JSR | \$F3E2 |
| 030E- | 06 E6 | ASL | \$E6 |
| 0310- | 20 F4 F3 | JSR | \$F3F4 |
| 0313- | A9 13 | LDA | £\$13 |
| 0315- | 85 06 | STA | \$06 |
| 0317- | 85 25 | STA | \$25 |
| 0319- | 20 C1 FB | JSR | \$FBC1 |
| 031C- | A5 29 | LDA | \$29 |
| 031E- | 18 | CLC | |
| 031F- | 69 04 | ADC | £\$04 |
| 0321- | 85 29 | STA | \$29 |
| 0323- | A2 28 | LDX | £\$28 |
| 0325- | 20 4A F9 | JSR | \$F94A |
| 0328- | E6 06 | INC | \$06 |
| 032A- | A5 06 | LDA | \$06 |
| 032C- | C9 18 | CMP | £\$18 |
| 032E- | D0 E7 | BNE | \$0317 |
| 0330- | 2C 53 C0 | BIT | \$C053 |
| 0333- | 2C 55 C0 | BIT | \$C055 |
| 0336- | A9 D3 | LDA | £\$D3 |
| 0338- | 85 28 | STA | \$28 |
| 033A- | A9 0B | LDA | £\$0B |
| 033C- | 85 29 | STA | \$29 |
| 033E- | A0 22 | LDY | £\$22 |
| 0340- | 20 73 03 | JSR | \$0373 |
| 0343- | 10 FB | BPL | \$0340 |
| 0345- | 20 E4 FB | JSR | \$FBE4 |
| 0348- | AD 00 C0 | LDA | \$C000 |
| 034B- | C9 80 | CMP | £\$80 |
| 034D- | 90 F9 | BCC | \$0348 |
| 034F- | 2C 10 C0 | BIT | \$C010 |
| 0352- | 2C 54 C0 | BIT | \$C054 |
| 0355- | A9 07 | LDA | £\$07 |
| 0357- | 85 29 | STA | \$29 |
| 0359- | A0 22 | LDY | £\$22 |
| 035B- | 20 73 03 | JSR | \$0373 |
| 035E- | 10 FB | BPL | \$035B |
| 0360- | A9 B1 | LDA | £\$B1 |
| 0362- | 8D E2 07 | STA | \$07E2 |
| 0365- | AD 00 C0 | LDA | \$C000 |
| 0368- | C9 80 | CMP | £\$80 |
| 036A- | 90 F9 | BCC | \$0365 |
| 036C- | 2C 10 C0 | BIT | \$C010 |
| 036F- | 2C 51 C0 | BIT | \$C051 |
| 0372- | 60 | RTS | |

\$20 dans l'Accumulateur... pour que...
... HPAG connaisse le numéro de page (HGR=\$20 ; HGR2=\$40)

A passe dans le registre X...
...et dans le registre Y. Tout est à zéro !
BKGND : Efface l'écran avec la dernière valeur
HGR : initialise (\$20)
Décalage à gauche : 00100000 (\$20) devient 01000000 (\$40)
et BKGND efface HGR2
\$13 dans l'Accumulateur
Stockage en \$6 (pointeur)...
...et en \$25 (position verticale du curseur)

BASCALC calcule l'adresse de la ligne, adresse que l'on retrouve en \$28-29.
On augmente la partie haute de \$4 pour obtenir l'adresse correspondante de la page TEXTE 2 et on stocke le résultat en \$29. Le registre X est chargé avec la partie basse. \$F94A envoie X espaces.

\$06 = 06 + 1. Si l'on n'est pas encore à \$18, il reste une ligne à traiter.

Fenêtre page 2 (POKE — 16301,0: POKE — 16299,0)

\$BD3 = 3027... adresse de la 24^e ligne + 3

\$22 (34) dans le registre Y (longueur du texte à afficher)
SAUT À LA ROUTINE D'AFFICHAGE

BIP

Teste si une touche a été pressée
Si plus petit que \$80 (128), le test continue

POKE — 16368,0
On bascule page 1...
et on va afficher le texte page 1... etc.

| AFFICHAGE | | | | |
|------------------|-------------------------|-----|-------------------------|--|
| 0373- | B9 7A 03 | LDA | \$037A, Y | |
| 0376- | 91 28 | STA | (\$28), Y | |
| 0378- | 88 | DEY | | |
| 0379- | 60 | RTS | | |
| 037A- | C1 C6 C6 C9 C3 C8 | |] T E X T E | |
| 0380- | C1 C7 C5 A0 10 01 07 05 | | | |
| 0388- | 20 32 A0 D0 D2 C5 D3 D3 | | | |
| 0390- | C5 DA A0 D5 CE C5 A0 D4 | | | |
| 0398- | CF D5 C3 C8 C5 | | | |
| | | | | |
| | | | | |

• Défilement d'une ligne

```

10 PRINT CHR$ (21):D$ = CHR$ (4): HOME : PRINT D$"BLOAD
   DEFIL.LM"
15 T$ = "1234567890ABCDEFGHIJKLMNPOQRSTUVWXYZabcd": POK
   E 769,0
20 PRINT T$:: CALL 768: GOTO 40
25 VTAB 22: INPUT "VTAB (1-23) -> ";V$: IF V$ = "" THEN
   45
30 V = VAL (V$): IF V < 1 OR V > 23 THEN 25
35 POKE 769,V - 1: VTAB V: GOTO 20
40 POKE - 16368,0: GOTO 25
45 PRINT D$"PR£3": CALL 809: PRINT
50 POKE 32,37: POKE 33,40: PRINT : HOME : FOR I = 1 TO
   23: VTAB I: PRINT " ": NEXT : POKE 32,39: LIST : PRI
   NT
55 VTAB 23: PRINT "<1> MENU DISQUE <2> TERMINE " : GET
   R$
60 IF R$ = "1" THEN TEXT : HOME : PRINT : PRINT D$"RUN
   MENU"
65 IF R$ < > "2" THEN VTAB 22: PRINT : GOTO 55
70 TEXT : HOME : END

```

DEFIL-DEMO

Un caractère sort à gauche de l'écran... et rentre à droite. Essayez cette petite démo.

Attention ! avec les anciens Apple IIe, supprimer la boucle de la ligne 50, créer une ligne 54 et modifier la ligne 55.

```

54 POKE 32,0: POKE 33,0
55 VTAB 23: POKE 1403,38 : PRINT
   "<1> MENU DISQUE <2> TERMINE";
   : GET R$

```

| | | | |
|-------|----------|-----|----------|
| 0300- | A9 00 | LDA | £\$00 |
| 0302- | 85 06 | STA | \$06 |
| 0304- | A5 06 | LDA | \$06 |
| 0306- | 20 C1 FB | JSR | \$FBC1 |
| 0309- | A0 00 | LDY | £\$00 |
| 030B- | B1 28 | LDA | (\$28),Y |
| 030D- | 48 | PHA | |
| 030E- | C8 | INY | |
| 030F- | B1 28 | LDA | (\$28),Y |
| 0311- | 88 | DEY | |
| 0312- | 91 28 | STA | (\$28),Y |
| 0314- | C8 | INY | |
| 0315- | C8 | INY | |
| 0316- | C0 28 | CPY | £\$28 |
| 0318- | D0 F5 | BNE | \$030F |
| 031A- | 88 | DEY | |
| 031B- | 68 | PLA | |
| 031C- | 91 28 | STA | (\$28),Y |
| 031E- | 20 A8 FC | JSR | \$FCA8 |
| 0321- | AD 00 C0 | LDA | \$C000 |
| 0324- | C9 80 | CMP | £\$80 |
| 0326- | 90 E1 | BCC | \$0309 |
| 0328- | 60 | RTS | |

DEFIL.LM

Le premier caractère de ligne est provisoirement empilé (\$30D: PHA), puis on lit successivement les caractères que l'on réécrit aussitôt à la même adresse... moins un. Finalement, le dernier caractère est récupéré sur la pile et vient en bout de ligne (\$31B: PLA). C'est BASCALC (\$FBC1) qui fournit l'adresse de la ligne sélectionnée.

Une boucle d'attente (\$FCA8... ou WAIT) freine le mouvement. Sa durée est fonction du contenu de l'accumulateur. On pourrait évidemment utiliser un autre système (empiler toute la ligne... puis la dépiler).

Cette partie est destinée à afficher votre routine LM (ligne Basic 45).

| | | | |
|-------|----------|-----|--------|
| 0329- | A9 00 | LDA | £\$00 |
| 032B- | 85 3A | STA | \$3A |
| 032D- | A0 03 | LDY | £\$03 |
| 032F- | 84 3B | STY | \$3B |
| 0331- | 20 53 F9 | JSR | \$F953 |
| 0334- | A9 17 | LDA | £\$17 |
| 0336- | 20 63 FE | JSR | \$FE63 |
| 0339- | 60 | RTS | |

• Défilement de l'écran texte

DÉFILEMENT

Cette démonstration vous montre parfaitement le fonctionnement de la routine, une simple extension de la précédente (p. 48).

Notez que :

- On poque à l'adresse \$7 la valeur de A (Wait).
- Presser une touche, stoppe le défilement.
- Presser pomme ouverte le relance.
- Escape permet de sortir de la routine.

```

100 TEXT : PRINT CHR$ (21): HOME : GOSUB 155
105 POKE 7,32
110 FOR I = 1 TO 24 STEP 2: FOR J = 1 TO 38 STEP 2: VTAB
    I: HTAB J: PRINT "S";: NEXT : NEXT
115 VTAB 17: PRINT : PRINT "    UNE TOUCHE ARRETE LE DEF
    ILEMENT    "
120 VTAB 19: PRINT "POMME OUVERTE LE RELANCE : ESCAPE =
    FIN    "
125 VTAB 1: CALL 768
130 PRINT : HOME
135 VTAB 22: PRINT "    <1> MENU DISQUETTE <2> TERMINE    "
    ;; GET R$
140 IF R$ = "1" THEN PRINT : PRINT CHR$ (4)"RUN MENU"
145 IF R$ < > "2" THEN PRINT : GOTO 135
150 HOME : END
155 FOR I = 768 TO 830: READ R%: POKE I,R%: NEXT : RETUR
    N
160 DATA 169,23,133,6,165,6,32,193,251,160,0,177,40,72,2
    00,177,40,136,145,40,200,200,192,40,208,245,136,104,
    145,40,165,7,32
165 DATA 168,252,198,6,16,221,173,0,192,201,155,240,16,2
    01,128,144,206,173,97,192,201,128,144,249,44,16,192,
    176,194,96
    
```

Voici, désassemblées, les 33 instructions installées par les lignes 155 à 165 de la démo.

| | | | |
|-------|----------|-----|----------|
| 0300- | A9 17 | LDA | £\$17 |
| 0302- | 85 06 | STA | \$06 |
| 0304- | A5 06 | LDA | \$06 |
| 0306- | 20 C1 FB | JSR | \$FBC1 |
| 0309- | A0 00 | LDY | £\$00 |
| 030B- | B1 28 | LDA | (\$28),Y |
| 030D- | 48 | PHA | |
| 030E- | C8 | INY | |
| 030F- | B1 28 | LDA | (\$28),Y |
| 0311- | 88 | DEY | |
| 0312- | 91 28 | STA | (\$28),Y |
| 0314- | C8 | INY | |
| 0315- | C8 | INY | |
| 0316- | C0 28 | CPY | £\$28 |
| 0318- | D0 F5 | BNE | \$030F |
| 031A- | 88 | DEY | |
| 031B- | 68 | PLA | |
| 031C- | 91 28 | STA | (\$28),Y |
| 031E- | A5 07 | LDA | \$07 |
| 0320- | 20 A8 FC | JSR | \$FCA8 |
| 0323- | C6 06 | DEC | \$06 |
| 0325- | 10 DD | BPL | \$0304 |

Au départ, l'adresse \$6 est chargée avec le numéro de la dernière ligne : \$17 (23).

Il suffit ensuite de décrémenter cette même adresse \$6, pour que toutes les lignes de l'écran soient traitées.

Pour obtenir la vitesse maximum, placez 255 dans la case mémoire \$7 (ligne 105 du programme de démonstration).

| | | | | |
|--------------|-------|----------|-----|--------|
| CMP \$9B | 0327- | AD 00 C0 | LDA | \$C000 |
| teste la | 032A- | C9 9B | CMP | £\$9B |
| touche | 032C- | F0 10 | BEQ | \$033E |
| ESCAPE. | 032E- | C9 80 | CMP | £\$80 |
| Quand une | 0330- | 90 CE | BCC | \$0300 |
| touche a | 0332- | AD 61 C0 | LDA | \$C061 |
| été pressée, | 0335- | C9 80 | CMP | £\$80 |
| la valeur | 0337- | 90 F9 | BCC | \$0332 |
| lue dépasse | 0339- | 2C 10 C0 | BIT | \$C010 |
| \$7F (127). | 033C- | B0 C2 | BCS | \$0300 |
| | 033E- | 60 | RTS | |

Des fenêtres rapides sur 40 colonnes

NOUS avons déjà traité le sujet des fenêtres sur 40 colonnes (*T.M.* n°4), mais la routine en langage machine était différente et sa philosophie de même. Ici, Argos vous propose une démo montrant comment remplir un écran (en un éclair)... et en vider instantanément une partie. A vous d'en tirer matière à gérer vos propres fenêtres (pour travailler en 80 colonnes, lire le n°9 de *T.M.*).

FENARGOS.BAS

```

100 TEXT : PRINT CHR$ (21): HOME
105 GOSUB 240
110 PRINT "REPLISSAGE D'ECRAN ET FENETRE          -----
----- par ARGOS"
115 VTAB 8: PRINT "CARACTERE DE REPLISSAGE ";; GET C$:
PRINT C$
120 VTAB 10: PRINT "CTRL-I POUR MODE INVERSE ";; GET R$:
IF R$ = CHR$ (9) THEN INVERSE : PRINT C$: NORMAL
125 PRINT :C = ASC (C$): IF C > 63 AND R$ = CHR$ (9) THE
N C = C - 64: GOTO 140
130 IF R$ < > CHR$ (9) THEN C = C + 128: GOTO 140
135 IF C > 63 THEN C = C + 64
140 VTAB 12: INPUT "HTAB FENETRE (0-38) ";R$
145 H1 = VAL (R$): IF H1 < 0 OR H1 > 38 THEN 140
150 VTAB 14: INPUT "LARGEUR FENETRE ";R$
155 H2 = VAL (R$): IF H2 < 1 OR H1 + H2 > 39 THEN 150
160 VTAB 16: INPUT "VTAB FENETRE (0-22) ";R$
165 V1 = VAL (R$): IF V1 < 0 OR V1 > 22 THEN 160
170 VTAB 18: INPUT "HAUTEUR FENETRE ";R$
175 V2 = VAL (R$): IF V2 < 1 OR V1 + V2 > 23 THEN 170
180 VTAB 1
185 POKE 7,C: CALL 768: REM REPLISSAGE
190 POKE 6,V1: POKE 7,V1 + V2: POKE 8,H1: POKE 9,H1 + H2
195 REM UN POKE 797,32 (ESPACE INVERSE) FERAIT UNE FENET
RE EN MODE INVERSE
200 CALL 807: REM FENETRE
205 GOSUB 235
210 VTAB 23: PRINT "": VTAB 24: HTAB 5: PRINT "<1> ENCOR
E <2> MENU <3> TERMINE ";; CALL - 198: GET R$
215 VTAB 1: PRINT : IF R$ = "1" THEN HOME : GOTO 110

```

REPLISSAGE DE L'ECRAN

On confie le caractère sélectionné à la case-mémoire \$7 de la page zéro. Le remplissage de l'écran, tel qu'il est réalisé, ligne par ligne, de la 23^e à la 0^e, n'affecte pas les 64 octets inutilisés pour la mémoire-écran qui sont réservés, vous le savez à chaque carte interface de périphérique.

Pour avoir un fond en mode inverse, il faut taper un espace (GET de la ligne 115), puis CTRL-I (GET de la ligne 120).

Si vous indiquez successivement les valeurs 1,38,1,22, vous obtiendrez évidemment un cadre (lignes 140, 150, 160 et 170).

FENÊTRE

On utilise les adresses \$6 à \$9 de la page zéro :

- \$6 : VTAB (0 à 22)
- \$7 : VTAB-FIN+1
- \$8 : HTAB (0 à 39)
- \$9 : HTAB-FIN+1

Lisez la REMarque de la ligne 195.

```

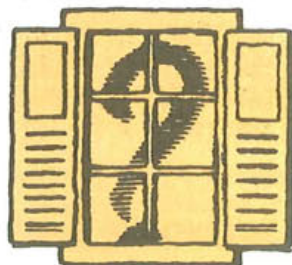
220 IF R$ = "2" THEN PRINT CHR$ (4)"RUN MENU"
225 IF R$ < > "3" THEN 210
230 HOME : END
235 CALL - 198: POKE - 16368,0: WAIT - 16384,128,127: PO
    KE - 16368,0: VTAB 1: RETURN
240 FOR I = 768 TO 813: READ R%: POKE I,R%: NEXT : RETUR
    N
245 DATA 169,23,133,6,165,6,32,193,251,165,7,160,39,145,
    40,136,16,251,198,6,16,238,96,32,193,251,164,8,169,1
    60,145,40,200,196,9,208,247,230,6,165,6,197,7,208,23
    4,96
    
```

FENARGOS.LM

Ces quelques octets sont ceux des lignes Basic 245-250

| | | | | |
|-------|----------|-----|-----------|---|
| 0300- | A9 17 | LDA | £\$17 |] INITIALISATION du pointeur \$6 (\$17 = 23, numéro de la dernière ligne). |
| 0302- | 85 06 | STA | \$06 | |
| 0304- | A5 06 | LDA | \$06 | |
| 0306- | 20 C1 FB | JSR | \$FBC1 | BASCALC (voir note en bas de page) |
| 0309- | A5 07 | LDA | \$07 | Caractère choisi dans l'Accumulateur |
| 030B- | A0 27 | LDY | £\$27 | Y = \$27 (39 en décimal) |
| 030D- | 91 28 | STA | (\$28), Y | On écrit A à l'adresse \$28-29 + Y |
| 030F- | 88 | DEY | | Y = Y - 1 |
| 0310- | 10 FB | BPL | \$030D | Si Y est égal ou supérieur à 0, on continue |
| 0312- | C6 06 | DEC | \$06 | \$6 = valeur en 1 |
| 0314- | 10 EE | BPL | \$0304 | Si plus grand ou égal zéro, suite |
| 0316- | 60 | RTS | | Retour au Basic |

| | | | | |
|-------|----------|-----|-----------|--|
| 0317- | 20 C1 FB | JSR | \$FBC1 |] Voir note dans le bas de la page Y est chargé avec HTAB-origine |
| 031A- | A4 08 | LDY | \$08 | |
| 031C- | A9 A0 | LDA | £\$A0 | \$A0 (espace) dans A |
| 031E- | 91 28 | STA | (\$28), Y | Ecriture à l'adresse \$28-29 + Y |
| 0320- | C8 | INY | | Y = Y + 1 |
| 0321- | C4 09 | CPY | \$09 | Y est-il égal à HTAB-FIN ? |
| 0323- | D0 F7 | BNE | \$031C | Non : on continue |
| 0325- | E6 06 | INC | \$06 | Sinon \$6(VTAB) est incrémenté |
| 0327- | A5 06 | LDA | \$06 |] Numéro de ligne chargé dans A Est-il égal à VTAB-FIN ? |
| 0329- | C5 07 | CMP | \$07 | |
| 032B- | D0 EA | BNE | \$0317 | Non : on continue |
| 032D- | 60 | RTS | | Ou retour au Basic |



\$FBC1 BASCALC

Cette routine calcule l'adresse de base qui correspond à la ligne courante et elle la range dans les mémoires BASH (\$28) et BASL (\$29). Au départ, A contient le numéro de ligne. ■

Quand les minuscules deviennent capitales

ON ne compte plus les routines capables de transformer les minuscules d'un programme Applesoft en autant de lettres capitales. En voici une autre, évidemment rapide puisqu'elle est écrite en langage machine. Pour bien la comprendre, nous vous conseillons d'examiner la mémoire de votre Apple... et de lire les explications de Clément Renard.

MINCAP.DEM

```

100 TEXT : HOME : IF PEEK (8) < > 77 AND PEEK (9) < > 77 THEN GOSUB 135      500E
105 PRINT "toto a été le premier à aller où ça po-": PRINT "sait un sérieux problème
      "
110 PRINT : PRINT : LIST 105,120                                           EEFB
115 IF PEEK (8) = 77 AND PEEK (9) = 77 THEN 150                          BAF9
120 VTAB 22: PRINT "tapez call 768, puis RUN ou LIST"                      1327
125 CALL - 198: POKE - 16368,0: WAIT - 16384,128,127: POKE - 16368,0     9E85
130 POKE 8,77: POKE 9,77: END                                             1D82
135 FOR I = 768 TO 841: READ R%: POKE I,R%: NEXT : RETURN                 220B
140 DATA 165,103,133,6,165,104,133,7,160,0,177,6,208,1,96,160,4,177,6,240,39,201,126 895B
      ,176,32,201,64,208,2,169,97,201,92,208,2,169,99,201,124,208,2,169,117
145 DATA 201,123,144,2,169,101,201,97,144,4,41,223,145,6,200,208,213,160,0,177,6,170 9E42
      ,200,177,6,134,6,133,7,208,190
150 VTAB 22: PRINT "<1> MENU DISQUETTE <2> TERMINE ";; CALL - 198: GET R$ 989A
155 POKE 8,0: POKE 9,0                                                  BB10
160 PRINT : HOME : IF R$ = "1" THEN PRINT CHR$ (4)"RUN MENU"           FCD5
165 IF R$ < > "2" THEN 150                                             EEA5
                                                                    9493

```

MINCAP.LM (lignes 140 et 145)

| | | | | |
|-------|-------|-----|---------|--|
| 0300- | A5 67 | LDA | \$67 |] On lit en \$67-\$68 l'adresse de départ du programme APPLESOFT. \$06-\$07 sont utilisés comme pointeurs. |
| 0302- | 85 06 | STA | \$06 | |
| 0304- | A5 68 | LDA | \$68 | |
| 0306- | 85 07 | STA | \$07 | |
| 0308- | A0 00 | LDY | £\$00 | Y = 0 |
| 030A- | B1 06 | LDA | (£06),Y | Contenu de l'adresse \$6-\$7 + Y dans A |
| 030C- | D0 01 | BNE | \$030F | Si différent de 0, on continue |
| 030E- | 60 | RTS | | |
| 030F- | A0 04 | LDY | £\$04 | Y = 4 |

| | | | | |
|-------|-------|-----|-----------|---|
| 0311- | B1 06 | LDA | (\$06), Y | Contenu de l'adresse \$6-\$7 + Y dans A |
| 0313- | F0 27 | BEQ | \$033C | Si c'est un 0, direction \$33C (une ligne est terminée) |
| 0315- | C9 7E | CMP | £\$7E | Est-ce supérieur à \$7E ? |
| 0317- | B0 20 | BCS | \$0339 | Si oui, C = 1 et GOTO \$339 |
| 0319- | C9 40 | CMP | £\$40 | Est-ce un "à" ? |
| 031B- | D0 02 | BNE | \$031F | Non : saut |
| 031D- | A9 61 | LDA | £\$61 | Oui : à devient a |
| 031F- | C9 5C | CMP | £\$5C | Est-ce un "ç" ? |
| 0321- | D0 02 | BNE | \$0325 | Non : saut |
| 0323- | A9 63 | LDA | £\$63 | Oui : ç devient c |
| 0325- | C9 7C | CMP | £\$7C | Est-ce un "ù" ? |
| 0327- | D0 02 | BNE | \$032B | Non : saut |
| 0329- | A9 75 | LDA | £\$75 | Oui : ù devient u |
| 032B- | C9 7B | CMP | £\$7B | Est-ce inférieur à \$7B(é) ? |
| 032D- | 90 02 | BCC | \$0331 | Si oui, C = 0 et pas de transformation |
| 032F- | A9 65 | LDA | £\$65 | é (ou è) devient E |
| 0331- | C9 61 | CMP | £\$61 | Est-ce inférieur à \$61 (a) |
| 0333- | 90 04 | BCC | \$0339 | Oui, on saute |
| 0335- | 29 DF | AND | £\$DF |] Le caractère minuscule devient majuscule |
| 0337- | 91 06 | STA | (\$06), Y | |
| 0339- | C8 | INY | | Y = Y + 1 |
| 033A- | D0 D5 | BNE | \$0311 | Encore un tour |
| 033C- | A0 00 | LDY | £\$00 | Y = 0 |
| 033E- | B1 06 | LDA | (\$06), Y | Lecture du contenu de \$6-\$7 + Y (= 0) |
| 0340- | AA | TAX | | A passe dans X |
| 0341- | C8 | INY | | Y = Y + 1 |
| 0342- | B1 06 | LDA | (\$06), Y | Lecture du contenu de \$6-\$7 + Y (= 1) |
| 0344- | 86 06 | STX | \$06 |] Nouvelle adresse |
| 0346- | 85 07 | STA | \$07 | |
| 0348- | D0 BE | BNE | \$0308 | Et ça repart ! |

Note de Tremplin Micro : en 33C, A = 0... donc on pourrait gagner un octet en passant tout simplement A dans Y (code : A8). Vous pouvez essayer en tapant 33C: A8 EA (EA est sans effet sur un programme).

§LIST

POUR MIEUX COMPRENDRE

10 REM DEMO
20 REM minuscules

Tapez ces deux lignes de "programme", puis tout ce qui est indiqué...

§CALL-151

*67.6A

0067- 01
0068- 08 1E 08
*801.81E

- \$67-68 = DÉPART PROGRAMME APPLESOFT. On y lit l'adresse de la ligne suivante qui commence ici en \$80B, où se trouve l'adresse de fin de programme, puisque celui-ci ne compte que 2 lignes. Comme l'interpréteur lira un zéro en \$81B, il saura que le programme est terminé.

- \$69-6A = LOMEM (approximativement fin de programme).

En examinant le contenu de la mémoire, de \$801 à 807, vous comprenez le pourquoi du Y = 4 de la ligne \$30F de notre routine.

0801- 0B 08 0A 00 B2 44 45
0808- 4D 4F 00 1B 08 14 00 B2
0810- 6D 69 6E 75 73 63 75 6C
0818- 65 73 00 00 00 0A 50

Les pointeurs (\$6-7) indiquent 0108 (801). L'adressage indirect permet de lire le contenu de \$801 + 4, soit \$805... en sautant le numéro de ligne. \$B2=REM, puis \$44-45-4D-4F=DEMO.

Affichage d'une ligne

Basic en hexadécimal

OBJET : Il est souvent intéressant de pouvoir lire, en hexadécimal, une ligne de programme (par exemple pour modifier un caractère sans retaper la ligne... dans le cas d'une ligne de longueur maximum). Cette petite routine est utilisée par l'intermédiaire de l'Ampersand (&) et permet, à partir d'un numéro de ligne, d'afficher :

- ce numéro en hexa (vous pouvez donc l'utiliser pour convertir une valeur décimale en valeur hexadécimale... même si ce n'est pas sa raison d'être) ;
- l'adresse hexa contenant... l'adresse de la ligne suivante ;
- l'adresse décimale effective du début de la ligne,
- tous les codes hexa de cette même ligne.

MODE D'EMPLOI : On ne peut plus simple. Taper *BRUN ADRLIGNE* (ou bien *BLOAD ADRLIGNE*, puis *CALL 768*). Ensuite, & *numéro de ligne*. Quand la ligne ne figure pas dans le programme, la routine se contente d'afficher la valeur hexa du numéro demandé et la mention "INEXISTANTE".

ADRLIGNE

| | | | |
|-------|----------|-----|-----------|
| 0300- | A9 4C | LDA | £\$4C |
| 0302- | 8D F5 03 | STA | \$03F5 |
| 0305- | A9 10 | LDA | £\$10 |
| 0307- | 8D F6 03 | STA | \$03F6 |
| 030A- | A9 03 | LDA | £\$03 |
| 030C- | 8D F7 03 | STA | \$03F7 |
| 030F- | 60 | RTS | |
| 0310- | 20 7B DD | JSR | \$DD7B |
| 0313- | 20 52 E7 | JSR | \$E752 |
| 0316- | 20 1A D6 | JSR | \$D61A |
| 0319- | 48 | PHA | |
| 031A- | A5 51 | LDA | \$51 |
| 031C- | F0 03 | BEQ | \$0321 |
| 031E- | 20 DA FD | JSR | \$FDFA |
| 0321- | 68 | PLA | |
| 0322- | 20 DA FD | JSR | \$FDFA |
| 0325- | A0 02 | LDY | £\$02 |
| 0327- | B1 9B | LDA | (\$9B), Y |
| 0329- | C5 50 | CMP | \$50 |

INSTALLATION D'AMPERV
JMP \$310 si &

FRMEVL : Evaluation de l'expression (résultat dans FAC)
GETADR : Convertit FAC en valeur entière (résultat dans \$50, \$51)
FNDLIN : La ligne fait-elle partie du programme (pointeur en \$9B, \$9C)
A est empilé
A chargé avec contenu de \$51
Si zéro pour partie haute, inutile d'afficher
PRBYTE : Affiche le contenu hexa de A
On dépile vers A
Affichage de la partie basse du numéro de ligne
Registre Y = 02
Lecture de la valeur trouvée à l'adresse \$9B-9C + Y
Est-elle égale à celle contenue dans \$50 ?

| | | | |
|-------|----------|-----|----------|
| 032B- | D0 3A | BNE | \$0367 |
| 032D- | C8 | INY | |
| 032E- | B1 9B | LDA | (\$9B),Y |
| 0330- | C5 51 | CMP | \$51 |
| 0332- | D0 33 | BNE | \$0367 |
| 0334- | 20 48 F9 | JSR | \$F948 |
| 0337- | A6 9B | LDX | \$9B |
| 0339- | A4 9C | LDY | \$9C |
| 033B- | 20 40 F9 | JSR | \$F940 |
| 033E- | E8 | INX | |
| 033F- | E8 | INX | |
| 0340- | D0 01 | BNE | \$0343 |
| 0342- | C8 | INY | |
| 0343- | 86 3C | STX | \$3C |
| 0345- | 84 3D | STY | \$3D |
| 0347- | A0 00 | LDY | £\$00 |
| 0349- | B1 9B | LDA | (\$9B),Y |
| 034B- | 85 3E | STA | \$3E |
| 034D- | C8 | INY | |
| 034E- | B1 9B | LDA | (\$9B),Y |
| 0350- | 85 3F | STA | \$3F |
| 0352- | C6 3E | DEC | \$3E |
| 0354- | A5 3E | LDA | \$3E |
| 0356- | C9 FF | CMP | £\$FF |
| 0358- | D0 02 | BNE | \$035C |
| 035A- | C6 3F | DEC | \$3F |
| 035C- | A5 3D | LDA | \$3D |
| 035E- | E6 24 | INC | \$24 |
| 0360- | 20 24 ED | JSR | \$ED24 |
| 0363- | 20 B3 FD | JSR | \$FDB3 |
| 0366- | 60 | RTS | |
| 0367- | A9 6F | LDA | £\$6F |
| 0369- | A0 03 | LDY | £\$03 |
| 036B- | 20 3A DB | JSR | \$DB3A |
| 036E- | 60 | RTS | |

Non... alors on saute !
Sinon $Y = Y + 1$.
Lecture de la partie haute...
... et comparaison avec \$51
Si c'est zéro, la ligne n'existe vraiment pas
PRBLNK affiche 3 espaces

X et Y chargés avec \$9B et \$9C...
pour que PRNTYX affiche la valeur hexa de l'adresse

$X = X + 2$ pour obtenir l'adresse effective du début de ligne
Si zéro, il ne faut pas oublier...
d'incrémenter la partie haute, c'est-à-dire Y

Et tout cela va en \$3C et \$3D (pour affichage des mémoires)

Pour que \$FDB3 (XAM) puisse afficher les mémoires contenant les lignes, il faut placer l'adresse de départ dans A1L/H (\$3C et \$3D) et l'adresse de fin dans A2L/H (\$3E et \$3F)... ce qui va être réalisé ici.

$CH = CH + 1$ (position du curseur)
LINPTR affiche les 2 octets de X, A (DEC)
XAM affiche les mémoires
Retour

Si ligne inexistante, STROUT affiche la chaîne pointée par Y-A (elle se termine par Ø)

| | |
|-------|----------|
| 036F- | A0 C9 |
| 0371- | CE C5 D8 |
| 0374- | C9 D3 |
| 0376- | D4 |
| 0377- | C1 CE |
| 0379- | D4 |
| 037A- | C5 00 |

Codes du mot
INEXISTANTE
(un espace au départ)

SAUVEGARDE :

BSAVE ADRLIGNE, A\$300, L\$7C

PAS D'ASSEMBLEUR ?

Tapez ces 8 lignes après un CALL-151 et sauveez sur disque, comme indiqué ci-dessus.

```

0300: A9 4C 8D F5 03 A9 10 8D F6 03 A9 03 8D F7 03 60 ££C64C
0310: 20 7B DD 20 52 E7 20 1A D6 48 A5 51 F0 03 20 DA ££940C
0320: FD 68 20 DA FD A0 02 B1 9B C5 50 D0 3A C8 B1 9B ££4A7D
0330: C5 51 D0 33 20 48 F9 A6 9B A4 9C 20 40 F9 E8 E8 ££D924
0340: D0 01 C8 86 3C 84 3D A0 00 B1 9B 85 3E C8 B1 9B ££F1DF
0350: 85 3F C6 3E A5 3E C9 FF D0 02 C6 3F A5 3D E6 24 ££2A36
0360: 20 24 ED 20 B3 FD 60 A9 6F A0 03 20 3A DB 60 A0 ££CF51
0370: C9 CE C5 D8 C9 D3 D4 C1 CE D4 C5 00 ££CFCC

```

Vous avez écrit à TREMLIN MICRO

• COLOR.HGR (n°7)

Je viens de copier la page 21 du n°7 de T.M. : j'ai obtenu un polygone entièrement teinté. J'ai modifié la ligne 130 afin d'obtenir un hexagone, puis un octogone, puis le contour d'un bonhomme. Et ceci plusieurs fois avec le même programme et tantôt j'obtiens la surface totalement colorée, tantôt partiellement, tantôt nullement colorée sans que j'arrive à en découvrir les raisons. J'ai dû plusieurs fois supprimer la ligne 110 car l'écran affichait : **ILLEGAL QUANTITE ERROR IN 230** ; j'ai copié exactement la page 21.

Quelle est cette **ERROR** qui ne se produit pas pour l'octogone et pour le polygone d'exemple, lesquels admettent très bien la ligne 110 ?

R. G. (76200 DIEPPE)

TM Dans notre exemple, Y2=140 (limite basse de l'image). Or, lorsque vous tracez votre octogone, vous utilisez la valeur 173... sans modifier Y2... et il est normal que l'image ne soit pas entièrement colorée.

• FONTE.LM (n°5)

Dans le dernier **TREMLIN MICRO** j'ai tapé **MASTER-MIND**, qui utilise notamment **FONTE.LM** de Tremplin Micro n°5, et malgré toutes les vérifications possibles je n'ai jamais pu faire fonctionner ce programme qui se "plante" à la ligne 230 (230 CALL AD).

J'ai retapé également **FONTE.LM**, mais c'est dans **FONTE.LM** qu'il y a des problèmes, malgré sa conformité absolue avec vos chiffres des pages 34 et 35. Je ne suis d'ailleurs pas autrement étonné, car je n'avais déjà pas pu faire fonctionner **FONTE.DEMO** de Tremplin Micro n°5 :

FONTE.DEMO se plantait aussi à la ligne 150 CALL 16384.

Monsieur P. (14000 CAEN)

TM Dans **FONTE.LM** il faut remplacer les 4 "EA" (à partir de l'adresse \$4004) par : A9 00 85 25.

• RUNTIME... ProDOS

J'ai écrit un programme en Basic, je l'ai compilé avec TASC, et ça marche sans problème sous DOS 3.3. Transféré en ProDOS, mon programme Basic



fonctionne. Une fois compilé, il fonctionne aussi. Mais dans ce cas, je suis obligé de **LOADER** le "RUN-TIME" de TASC en mode direct. Autrement dit :

1. Charger la disquette ProDOS
2. BLOAD RUNTIME
3. BRUN PROGRAMME

En DOS 3.3 je boote sur un Startup ainsi conçu :

```
4 PRINT CHR$(4) + "BLOAD RUNTIME" +  
CHR$(13) + CHR$(4) + "BRUN PROGRAMME".
```

Cela fonctionne. En ProDOS 1.1.1, ce même Startup se comporte de la façon suivante : je boote sur ce startup, Runtime est bien **LOADÉ**, mais ensuite le programme suivant (d'abord écrit sous DOS 3.3, puis compilé par TASC, puis transféré en ProDOS) n'est pas appelé : il s'affiche :

? SYNTAX ERROR IN 4

Daniel D. (75013 PARIS)

TM Vous avez beaucoup de chance si TASC fonctionne sous ProDOS !

Utilisez un programme EXEC réalisé sur le modèle suivant :

```
10 D$ = CHR$(4):G$ = CHR$(34)
```

```
20 PRINT D$"OPEN BOOT"  
30 PRINT D$"WRITE BOOT"  
40 POKE 33,30  
50 PRINT "BLOAD RUNTIME"  
60 PRINT "BRUN PROB.RUNT.OBJ"  
70 PRINT D$"CLOSE BOOT"  
80 POKE 33,40
```

• LOGIC.GRILLES (n°1)

J'ai deux questions à vous poser.

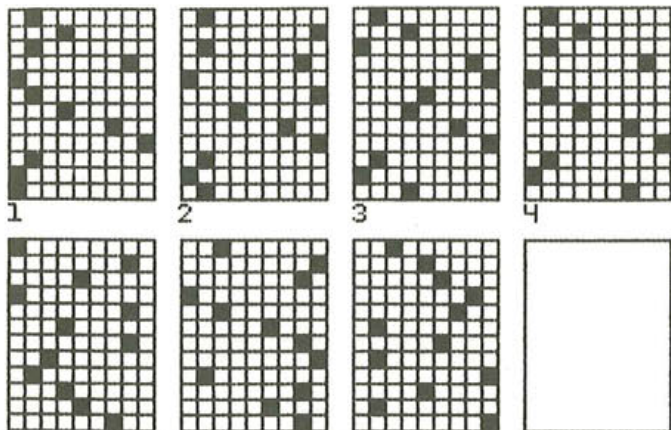
1. Peut-on sortir sur imprimante, et comment (avec ou sans la solution) les grilles du programme **LOGIC-GRILLES** du n°1 (avec **IMP.HGR** ou **GF**) ?
2. Je possède une imprimante **Imagewriter II** et contrairement à ce que vous écrivez la disquette **TOOLKIT** n'est pas fournie avec la machine. Impossible de la trouver (vous en faites si souvent référence !).

André BOISSIÈRE (76330 N.D. DE GRAVENCHON)

TM 1. A partir d'**IMP.HGR.1** (paru dans T.M. n°6), vous pouvez imprimer vos **LOGIC.GRILLES**

TM en modifiant ainsi votre programme :

```
505 PRINT : PRINT CHR$(4) "BLOAD IMP.HGR.1,
A7000": POKE 6,1: POKE 7,1: POKE 253,0:
POKE 254,24
510 CALL 7000: PRINT : CALL 1002
511 PRINT CHR$(4) "PR£1": PRINT CHR$(27) "c":
PRINT "GRILLE-REPOSE = " H: PRINT
CHR$(12) : PRINT CHR$(4) "PR£0" : PRINT
```



Quelle grille du haut peut logiquement occuper l'emplacement vide ?

2. Votre concessionnaire peut sûrement vous fournir cette disquette TOOLKIT, d'ailleurs très pratique pour tester le fonctionnement de l'IWII avec le ruban couleur.

• La Boîte aux idées (n°8)

1. Dans le listage de la page 8, à la première ligne, on note une anomalie pour le moins étrange :
`300 A9 2C LDA £$7C`
 Que vient faire ici, le 7C... et comment avez-vous pu l'obtenir sur l'imprimante... les codes A9 2C générant obligatoirement un LDA £\$2C ?
2. Une colle : peut-on réellement chercher un mot de longueur \$FF (votre paragraphe RECHERCHE, page 6) ?
Maurice C. (76000 ROUEN)

- TM**
1. Pas de mystère. C'est bien 2C et la ligne avait été corrigée... par collage, mais la pastille qui recouvrait 7C s'est envolée !
 2. Non. C'est évidemment une plaisanterie... puisque la longueur des lignes ne dépasse pas 78 caractères, je crois. De plus, l'implantation de la routine à l'adresse \$300 n'autorise pas une telle fantaisie... car on écraserait d'intéressantes valeurs à partir de \$3D0.

• DIR-PRODOS

Pouvez-vous me fournir, succinctement, sans blabla la liste des commandes à envoyer pour :

1. Créer un sous-catalogue **ESSAI** (DIR) sur un disque **BD1** ?
2. Changer le préfixe **BD1** en **BD2** ?

Bernard D. (94410 SAINT-MAURICE)

TM Tout cela a déjà été expliqué dans *Trempin Micro...* Procurez-vous notre disquette **INDEX** !

Création sous-catalogue :

CREATE/BD1/ESSAI... et c'est tout ! Pour y accéder, vous taperez ensuite : PREFIX/BD1/ESSAI.

Changer de préfixe :

RENAME/BD1,/BD2 (en plaçant la virgule au bon endroit S.V.P.)

• BIT.MAP.RAM

Bravo pour le numéro 8, il est très intéressant ! Voici quelques éléments de réponse pour Jean-Claude H. d'Antony concernant la gestion des pages mémoires par ProDOS.

Le système Bit.Map est situé de \$BF58 à \$BF6F et il gère les 192 pages de \$00 à \$BF. Chaque octet concerne 8 pages et chaque page est représentée par 1 bit :

Bit à 0 = page libre

Bit à 1 = page occupée.

Il faut lire les octets dans le sens poids fort vers poids faible, voici un petit exemple :

\$BF58 = 11001111

Pages 0, 1, 4, 5, 6 et 7 occupées.

Voici ci-dessous un petit programme d'illustration qui affiche le Bit-Map.

Maurice CHAVELLI (13012 MARSEILLE)

```
100 DATA 162,6,134,37,162,0,230,37,32,34,252,169,
10,133,36,32,36,3,232,32,36,3,232,224,24,208,
235,173,0,192,16,251,141,16,192,96,189,88,191
,160,8,136,16,1,96,10,72,144,4,169,170,208,2,
169,160,32,237,253,104,76,41,3
105 FOR C = 768 TO 829: READ A%: POKE C,A%: NEXT
110 A$ = "0123456789ABCDEF"
115 HOME : HTAB 10: PRINT "BIT-MAP DE LA RAM"
120 PRINT TAB( 10)"-----"
125 PRINT TAB( 10)"(PAGES $00 A $BF)"
130 VTAB 7: HTAB 11: INVERSE : FOR C = 1 TO 16: P
RINT MID$( A$,C,1);: NEXT
135 VTAB 8: FOR C = 1 TO 12: HTAB 10: PRINT MID$(
A$,C,1): NEXT
140 VTAB 24: FLASH
145 HTAB 9: PRINT "PAGES OCCUPEES => *";
150 NORMAL : CALL 768: HOME
```

Suite du courrier page 58.

COURRIER

(suite)

• LIST X,Y

Pourquoi ai-je une SYNTAX ERROR IN 20 ?

```
10 INPUT "DEBUT LISTE: "; DL
```

```
15 INPUT "FIN LISTE: "; FL
```

```
20 LIST DL, FL
```

R.C. (34100 MONTPELLIER)

TM La commande LIST n'accepte pas les variables. Pourquoi vous obstinez-vous à contrarier votre Apple chéri et son Applesoft !

• TRACE HGR

Qui m'écrira, en langage machine (pour aller vite), une routine capable de tracer des lignes joignant des points disposés aléatoirement sur un écran HGR ?

J. C. (02100 ST-QUENTIN)

TM Oui... qui ?

• ETAT DES COMMUTATIONS

Comment savoir si l'on est ou non en mode TEXT, en HGR ou HGR2... avec ou sans fenêtre ?

S. C. (33220 STE-FOY)

TM En interrogeant les commutateurs logiciels... comme le fait (c'est seulement une petite DEMO) la courte routine que voici :

```
60000 HOME : VTAB 24: FOR I = 1 TO 4:
      READ V,T$,T1$
60010 IF PEEK (V) > 127 THEN T$ = T1
      $
60020 PRINT T$ " ";
60030 IF PEEK (49178) < 128 THEN GE
      T R$
60040 PRINT : NEXT
60050 DATA 49178,MODE GRAPHIQUE,MODE
      TEXTE,49179,SANS FENETRE,AVEC F
      ENETRE,49180,PAGE 1,PAGE 2,4918
      1,BASSE RESOLUTION,HAUTE RESOLU
      TION
```

Réponses à de nombreux lecteurs

IW.TAB.HOR

Qui oserait prétendre que les tabulations de l'IW ne fonctionnent pas ?

A essayer sans plus attendre !

```
100 TEXT : HOME
105 VTAB 7: INPUT "NOMBRE DE LIGNES PAR PAGE ":NP: IF NP <
    > 66 AND NP < > 72 THEN 105
110 VTAB 9: INPUT "NOMBRE DE LIGNES A EDITER PAR PAGE ":NE:
    IF NE > = NP THEN 105
115 NS = NP - NE:NE = NE - 2
120 CODE$ = CHR$ (27) + "c" + CHR$ (29) + "Aa": FOR I = 1
    TO NE:CO$ = CO$ + "àà": NEXT :CO$ = CO$ + "Ca": FOR I =
    1 TO NS:CO$ = CO$ + "àà": NEXT :CO$ = CO$ + "Aa" + CHR$
    (30)
125 POKE 768,165: POKE 769,6: POKE 770,32: POKE 771,237: POK
    E 772,253: POKE 773,96
130 PRINT CHR$ (4)"PR£1": PRINT
135 PRINT CHR$ (9) + STR$ ( LEN (CO$)) + "N": REM APPLE 2C
140 FOR I = 1 TO LEN (CO$): POKE 6, ASC ( MID$ (CO$,I,1)):
    CALL 768: NEXT
145 PRINT : PRINT CHR$ (4)"PR£0"
```

```
10 PRINT CHR$ (4)"PR£1"
15 PRINT CHR$ (27)"(010,020,035,050,070.
20 PRINT
25 PRINT CHR$ (9); CHR$ (1): REM DESACTIVE
    LA COMMANDE CTRL-I DE LA CARTE SERIE (E
    LLE DEVIENT CTRL-A)
30 FOR I = 1 TO 5
40 PRINT I;: NEXT
45 PRINT CHR$ (1); CHR$ (9)
50 PRINT CHR$ (4)"PR£0"
```



IW.LISTER

Tout y est, même le saut de page !

Normalement, cette petite routine fonctionne, sur DMP ainsi que sur l'ancienne et sur la nouvelle IMAGE-WRITER. ■

L'ASSEMBLEUR

le plus puissant du monde ?

LE Réseau Planétaire* diffuse ORCA/M, actuellement considéré comme l'un des meilleurs assembleurs du monde. La nouvelle version fonctionne sous ProDOS et accepte les instructions du 65C02. Mieux, il semble bien que cet ORCA/M soit parfaitement compatible avec le 65C816 (16 bits) des prochains APPLE, mais nous n'avons évidemment pas été en mesure de le vérifier.

Un point noir — mais uniquement pour ceux d'entre nous qui ne pratiquent pas l'anglais —, l'importante documentation (250 pages) de ce bel outil est rédigée en anglais. Dommage que le Réseau Planétaire ne puisse pas en fournir une traduction !

Sinon, j'avoue avoir été impressionné par les nombreuses possibilités qu'offre ORCA/M aux programmeurs.

Editeur pleine page :

La recopie ou la création d'un programme est réellement facile (les utilisateurs de ProCODE y sont habitués). Recherches et remplacements sont évidemment possibles, ainsi que la copie ou le déplacement de blocs entiers.

Routines graphiques :

Vous pouvez, grâce à ces routines graphiques — en simple et en double haute résolution ! — utiliser des couleurs, remplir des formes et dessiner des lignes. C'est une aide remarquable... quand on sait s'en servir.

Macros-instructions :

Si le langage d'ORCA vous autorise à écrire vos propres macros-instructions, il convient de remarquer que cet assembleur vous est fourni avec un catalogue de plus de 200 macros déjà écrites... et par des spécialistes.

APPRENTISSAGE

Si vous avez une bonne habitude des assembleurs américains (Big Mag, Merlin, Lisa), vous aurez tôt fait d'assimiler les diverses commandes d'ORCA/M, mais il

vous faudra néanmoins plusieurs jours pour être réellement en mesure d'en exploiter les richesses. Si vous débutez et n'avez aucune pratique de l'anglais, renoncez à ORCA/M et offrez-vous le ProCODE français de Version Soft.

Mais n'hésitez pas à prier le Réseau Planétaire de vous adresser une documentation complète sur ce champion.

Notez qu'ORCA/M est vendu 1200 F TTC, prix justifié par l'existence de la bibliothèque de macros-instructions. c.r.

AUTRES PRODUITS INTÉRESSANTS

Toujours au Réseau Planétaire :

- **Fontrix**, utilitaire multi-usages de graphismes et de composition de signes et caractères typographiques ;
- **Gutenberg**, considéré comme le traitement de texte le plus puissant, mais réservé à des utilisateurs avertis (un détail : le mode d'emploi — toujours en anglais — est un "petit" bouquin de 768 pages !) ;
- **Prime Plotter**, certainement indispensable à celles et à ceux qui sont amenés à réaliser de nombreux graphiques ;
- **L'Agenda Apple**, bon marché et entièrement en français (200 F).

BP 3 — 43260 SAINT-JULIEN CHAPTEUIL.

L'EXPERT

système expert

TENTATIVE fort intéressante dont l'objectif est de doter les utilisateurs d'Apple II d'un outil leur permettant d'utiliser des bases de connaissances personnelles dans la résolution de problèmes qui ne le sont pas moins.

L'EXPERT est composé de deux modules :

- L'Expert proprement dit
- Base Manager qui est le gestionnaire des bases de connaissances.

DOCUMENTATION :

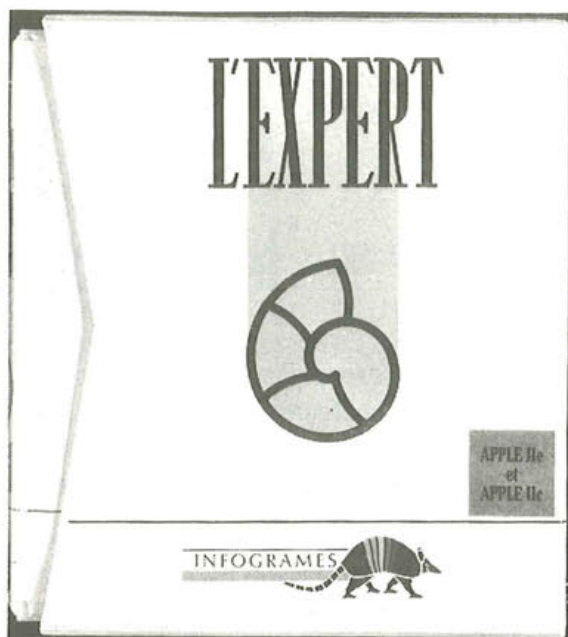
Livrée dans un classeur, sous emboîtement, elle est en français, et c'est un bon point. On pourrait peut-être lui reprocher de se révéler un peu trop succincte, mais il n'est pas sûr que cette critique soit encore justifiée après quelques heures d'utilisation. Elle explique comment utiliser L'Expert (fichiers, édition, facilités, déduire, vérifier une hypothèse... expertiser). On sait rapidement comment ajouter des faits lexicalement, formuler et sauvegarder un fait, utiliser le mode Pourquoi ou le mode Expertiser.

Son chapitre 2 est consacré au fonctionnement de Base Manager. Il est par ailleurs certain que les non-initiés en apprécieront le chapitre 3. Celui-ci donne en effet des informations générales sur les systèmes experts et je conseille vivement aux acheteurs de ce logiciel de commencer par là une initiation à l'utilisation de L'Expert.

MIND SOFT ET LES SYSTÈMES EXPERTS

L'existence du Club USE (Utilisateurs des Systèmes Experts) prouve que Mind Soft* désire réellement aller plus loin et ne point se contenter de développer et vendre des systèmes experts. Évidemment, l'adhésion au Club n'est pas gratuite (350 F par an), mais les adhérents reçoivent, en fonction de l'actualité de Mind Soft, des salons, etc., une lettre d'information qui contient des exemples de bases de connaissances développées par des utilisateurs, ainsi que des astuces et renseignements techniques pour le développement de ces mêmes bases. Les deux premiers bulletins m'ont paru intéressants.

Détail important : tout membre du Club a dit-on la possibilité, en priorité, de faire éditer ses propres bases de connaissances pro-



fessionnelles, validées au préalable par Mind Soft, puis de les faire commercialiser par le réseau de distribution de la marque.

On notera que, Mind Soft a passé un contrat de co-édition avec Infogrames, Société dont on connaît la brillante activité dans le secteur des logiciels de jeux.

UNE DÉMARCHÉ PERSONNELLE

Bon nombre de Lecteurs de *Tremplin Micro* aimeraient sans doute que je leur explique comment fonctionne ce logiciel. Ou encore que je leur suggère de courir chez leur revendeur Apple pour se l'offrir sans plus tarder... à charge pour moi, pour le cas où il les décevrait, de supporter leur mauvaise humeur ! Ils vont être déçus. L'Expert constitue une ouverture sur l'informatique personnelle de demain. Ce n'est pas encore de l'IA (intelligence artificielle), mais cela s'en rapproche. Les logiciels de cette famille permettent assurément aux esprits ouverts et imaginatifs de se lancer dans la grande aventure des systèmes experts. Aussi procureront-ils aux uns de nombreuses et agréables satisfactions, tandis qu'ils plongeront les autres dans des abîmes de perplexité.

Pour ma part, je souhaite que L'Expert suscite moult curiosités et inspire à nos lecteurs programmeurs un nombre égal d'idées originales et fructueuses !

C.R.

* 3, rue de l'Arrivée — BP 63 — 75749 PARIS CEDEX 15 (L'EXPERT est vendu, avec ses deux disquettes, 1590 F.)

OCCUPATION COMPLÈTE DE LA PAGE ZÉRO

Pour les chiffres et valeurs indiqués par la suite, la notation est donnée en base 16 (hexadécimale). Le \$ indique une adresse ou un pointeur : \$50 est l'adresse 50 en notation hexadécimale.

BYTE UTILISATION

| | |
|-------|---|
| 00-02 | Saut (JMP) vers la routine de démarrage à chaud de l'Apple-soft. JMP \$D43C avant le boot d'une disquette. JMP \$F128 après le boot d'une disquette. |
| 03-05 | Saut (JMP) vers le sous-programme de sortie de caractères. JMP \$DB3A avant le boot d'une disquette. JMP \$F128 après le boot d'une disquette. |
| 06-09 | Adresses inutilisées. |
| 0A-0C | Saut (JMP) défini par l'utilisateur pour l'utilisation de la fonction USR. Permet de communiquer des paramètres à des sous-programmes en langage machine et de faire un saut vers ces programmes. L'adresse est celle du sous-programme. Après un démarrage à froid, cette adresse contient : 4C 99 E1 ou JMP \$E199 |
| 0D-0E | Adresses utilisées comme registres d'aide pour la comparaison de caractères par la routine CHRGET : ENDCHR = 00 (fin de chaînes de caractères) CHARAC = " (début ou fin de chaînes) SEPAR = : (séparateur d'instructions) |
| 0F | Valeur de la dimension du dernier tableau déclaré. Longueur d'une nouvelle valeur de BASIC (chiffre). Compteur lors de la codification (Token) des instructions d'Applesoft. |
| 10 | Drapeau (Flag) pour DIM. |
| 11 | VALTYP. Drapeau de chaîne : — FF si c'est une chaîne — 00 si c'est un nombre. |
| 12 | Drapeau pour les entiers : — 80 si c'est un entier. — 00 dans les autres cas. |
| 13 | Drapeau Garbage Collection. Drapeau DATA lors de la codification. |
| 14 | SUBFLAG. Drapeau pour le traitement des variables : — 00 variables simples. — 40 tableaux de variables (avec STORE, sans indices). — 80 variables entières interdites. |
| 15 | Drapeau d'entrée : — 00 INPUT — 40 GET — 98 READ. |
| 16 | Registre de comparaison. Utilisé par Applesoft lors d'opérations de comparaison. Registre pour le calcul de TAN. |
| 17-19 | Adresses inutilisées. |
| 1A-1B | Pointeur sollicité par les sous-programmes pour le graphisme. |
| 1C | Sert de masque pour la couleur. |
| 1D | Sert de mémoire et compteur lors d'un tracé d'une ligne en graphique. |
| 1E-1F | Adresses inutilisées. |
| 20 | WNDLFT. Marge gauche de la fenêtre de texte (valeur entre 0 et 39 inclus). |
| 21 | WNDWDTH. Largeur de la fenêtre de texte (valeur entre 1 et 40 inclus). |
| 22 | WNDTOP. Haut de la fenêtre de texte (valeur entre 0 et 22 inclus). |

BYTE UTILISATION

• par Marcel COTTINI

| | |
|-------|--|
| 23 | WNCBTRM. Bas de la fenêtre de texte (valeur entre 1 et 24 inclus). |
| 24 | CH. Position horizontale du curseur (valeur entre 0 et 39 inclus). |
| 25 | CV. Position verticale du curseur (valeur entre 0 et 23 inclus). |
| 26-27 | GBASL & GBASH. Adresse de la première colonne dans la page graphique. |
| 28-29 | BASL & BASH. Adresse de la première colonne pour la page texte. La valeur de la ligne courante se trouve en CV (\$25). |
| 2A-2B | BAS2L & BAS2H. Pointeur de la ligne de destination (déroulement d'écran ou scroll). |
| 2C-2D | H2 & V2. Coordonnées pour HLINE et VLINE (GR). |
| 2E | MASK. Constantes pour le tracé en GR. |
| 2F | LASTIN. Valeur pour la lecture d'une cassette. |
| 30 | COLOR. Masque pour le codage d'un point en graphique haute résolution. Spécifie la couleur des lignes paires et impaires (respectivement, les 4 bits les plus à gauche et les 4 bits les plus à droite). |
| 31 | MODE. Pointeur pour le moniteur. Différencie le type d'information dans les commandes au moniteur. |
| 32 | INVFLG. Définit le mode d'affichage vidéo. — FF mode NORMAL — 3F mode INVERSE — 7F mode FLASH. |
| 33 | PROMPT. Contient la valeur du prompt (normalement AA). |
| 34-35 | YSAV & YSAV1. Sauvegarde du pointeur de la commande exécutée par le moniteur. |
| 36-37 | CSW1 & CSWH. Adresse de la routine de sortie de caractères : \$9EBD sous DOS 3.3 |
| 38-39 | KSWL & KSWH. Adresse de la routine d'entrée de caractères. |
| 3A-3B | PCL & PCH. Sauvegarde du compteur ordinal (registre PC). |
| 3C-3D | A1L & A1H. |
| 3E-3F | A2L & A2H. |
| 40-41 | A3L & A3H. Ce groupe d'adresses, de \$3C à \$41 correspond à la zone de sauvegarde et transmission de paramètres avant l'appel de certains sous-programmes du moniteur (par exemple MOVE. CALL — 468). |
| 42-43 | A4L & A4H. Zone de sauvegarde de paramètres avant l'appel de sous-programmes du moniteur. |
| 44-45 | A5L & A5H. Sauvegarde de l'accumulateur (registre A). |
| 46 | XREG. Sauvegarde du registre X. |
| 47 | YREG. Sauvegarde du registre Y. |
| 48 | STATUS. Sauvegarde du registre d'état PS. |
| 49 | SPNT. Sauvegarde du registre pointeur de pile. |
| 4A-4B | Utilisé par le DOS. |
| 4C-4D | Inutilisés. |
| 4E-4F | RNDL & RNDH. Génère un nombre aléatoire durant KEYIN. |
| 50-51 | LINNUM. Registre 16 bits contenant le numéro de la ligne actuelle du programme en cours. |
| 52 | Registre utilisé comme pointeur de pile pour le descripteur de pile qui se trouve de \$55 à \$5D. |
| 53-54 | Pointeur du haut du descripteur, de la pile du descripteur (\$55-\$5D). |
| 55-5D | Pile du descripteur pour un maximum de 3 descripteurs. |

(suite au dos)

BYTE UTILISATION

| | |
|-------|--|
| 5E-5F | INDEX. Pointeur temporaire pour déplacer des chaînes. |
| 60-61 | Pointeur pour Applesoft. |
| 62-65 | Registres utilisés pour les divisions et multiplications. |
| 66 | Est modifié lors du traitement d'une division. |
| 67-68 | XTAB. Début du texte d'un programme en BASIC Applesoft \$801 par défaut. |
| 69-6A | LOMEM. Adresse de début de la zone des variables simples. |
| 6B-6C | ARYTAB. Adresse de début des tableaux de variables, et fin des variables simples. |
| 6D-6E | STREND. Adresse de fin + 1 de la zone des tableaux de variables, et début de la zone libre. |
| 6F-70 | FRETOP. Adresse du début de la zone des chaînes de caractères, et fin de la zone libre. |
| 71-72 | FRESPC. Pointeur pour la gestion des chaînes de caractères. |
| 73-74 | HIMEM. Adresse la plus élevée de mémoire vive (FIN + 1 de la RAM). |
| 75-76 | CURLIN. Numéro de la ligne actuelle (FF en mode immédiat). |
| 77-78 | OLDLIN. Numéro de la dernière ligne exécutée ou de la ligne interrompue (maximum à 63999 avec Applesoft). |
| 79-7A | OLDTXPTR. Pointeur de la dernière instruction — 1 à exécuter, lors d'une interruption par ONERR. CONT l'utilise pour redémarrer. — \$7A contient FF en mode immédiat. |
| 7B-7C | Numéro de ligne de l'instruction DATA. |
| 7D-7E | Pointeur de l'élément DATA à lire par l'instruction READ. |
| 7F-80 | Pointeur utilisé par les routines d'entrées de caractères. Lorsque INPUT ou READ sont actifs, ce pointeur est mis à 201. |
| 81-82 | Registre pour le nom de la dernière variable (2 caractères ASCII). |
| 83-84 | VARPNT. Adresse de la valeur de la dernière variable utilisée. |
| 85-86 | FORPNT. Pointeur utilisé par des routines de la ROM Applesoft (LET, NEXT, etc.). |
| 87-88 | Pointeur utilisé par CHRGET. |
| 87 | Registre de priorité lors de l'évaluation de valeurs numériques. |
| 89 | Registre de comparaison utilisé par Applesoft. |
| 8A-8E | TEMP3 ou FAC5. Registre flottant temporaire sur 5 octets utilisé lors du traitement des puissances des nombres. |
| 8A-8B | Pointeur de la variable définie par l'instruction "DEF FN". Pointeur de la routine Garbage Collection pour déplacer les chaînes. |
| 8C-8D | Pointeur du descripteur de chaîne actuel. |
| 8F | Longueur d'une variable en mémoire. — 3 ou 7. |
| 90-92 | Saut (JMP) lors d'un appel d'une fonction Applesoft. L'adresse \$90 contient la valeur de saut "4C" et \$91, \$92 le pointeur de l'instruction appelée. |
| 93-97 | TEMP1 ou FAC3. Registre flottant sur 5 octets. — \$93 valeur de l'exposant. — \$94, \$95 pointe le premier élément d'un tableau de variables. — \$96, \$97 Fin + 1 d'un bloc d'octets (256) lors d'utilisation de certaines routines. |
| 98-9C | TEMP2 ou FAC4. Registre flottant sur 5 octets. — \$99 registre d'aide : FAC1 chaîne — \$9A registre pour le traitement des chaînes et constantes réelles. — \$9B, \$9C pointeurs divers. |
| 9D-A2 | FAC ou FAC1. Registre accumulateur sur 6 octets. — \$9D exposant de FAC1 ou indice signé — \$9E Mantisse1. — \$9F Mantisse2. — \$A0 Mantisse3. — \$A1 Mantisse4. — \$A2 signe provisoire de FAC1. Teste si le 7 ^e bit est significatif. |

BYTE UTILISATION

• par Marcel COTTINI

| | |
|-------|---|
| A3-A4 | Utilisé pour des fonctions mathématiques. — \$A3 registre pour traiter les polynômes. — \$A4 registre pour traiter les Mantisses. |
| A5-AA | ARG ou FAC2. Registre argument flottant sur 6 octets. — \$A5 exposant ou signature. — \$A6, \$A9 fonction math. — \$AA fonction signe (SGN). Signe provisoire de FAC2. |
| AB-AC | STRNG1. Pointeur utilisé pour déplacer une chaîne par MOVINS. |
| A B | Signe combiné. |
| AC | Adresse temporaire pour stocker la valeur arrondie de FAC1. |
| AD-AE | STRNG2. Pointeur du dernier caractère d'une chaîne. |
| AD | Registre pour le codage des instructions Applesoft (Token) |
| AF-B0 | PRGEND. Pointeur de la fin + 1 d'un programme. |
| B1-C8 | CHRGET. Sous-programme pour l'acquisition du prochain caractère. |
| B8-B9 | TXTPTR. Pointeur (ou compteur) de CHRGET. Adresse du dernier caractère obtenu par CHRGET. |
| B7 | CHRGOT. Deuxième point d'entrée de la routine CHRGET, pour la saisie du caractère actuel. |
| C9-CD | Nombre aléatoire sur 5 octets. Utilisé par RND. |
| CE-CF | Adresses inutilisées. |
| D0-D5 | Registres utilisés pour le graphique haute résolution. |
| D6 | Drapeau pour Autostart. — 80 toute commande dans un programme est considérée comme un "RUN". |
| D7 | Inutilisé. |
| D8 | ERRFLG. Drapeau pour ONERR. — 80 si actif. |
| D9 | RUNMOD. Drapeau pour RUN. — 80 si actif. |
| DA-DB | ERRLIN. Numéro de ligne où l'erreur s'est produite. |
| DC-DD | ERRPOS. Pointe la position du caractère d'où l'erreur a été prise en compte. |
| DE | ERRNUM. Code de l'erreur. |
| DF | ERRSTR. Sauvegarde du registre pointeur de pile avant l'erreur. |
| E0-E1 | Coordonnées horizontales du curseur en HGR (valeur de 0 à 279). |
| E2 | Coordonnées verticales du curseur en HGR (valeur de 0 à 191). |
| E3 | Inutilisé. |
| E4 | Masque pour la couleur en HGR, suivant le dernier HCOLOR déclaré. |
| E5 | Position d'un point pour certaines routines en HGR. Position déterminée suivant une matrice définie dans le sous-programme, et continuellement mise à jour. |
| E6 | Drapeau pour la page HGR. — 20 HGR (page 1). — 40 HGR2 (page 2). |
| E7 | Valeur de SCALE. |
| E8-E9 | Pointeur du début de la table des formes (Shape-Table). |
| EA | Compteur de Collision, utilisé en haute résolution (DRAW/XDRAW) |
| EB-EF | Inutilisés. |
| F0 | FIRST. Première position de HLIN et VLIN. |
| F1 | SPDBYT. Valeur de SPEED (256-SPEED). |
| F2 | Drapeau pour la fonction TRACE. |
| F3 | Drapeau pour la fonction FLASH. — 40 si FLASH actif. — 00 sinon. |
| F4-F8 | Pointeurs utilisés par ONERR. — \$F4, \$F5 pointe dans la zone du programme, la ligne de la déclaration de l'instruction "ONERR". — \$F6, \$F7 numéro de la ligne du programme où se trouve l'instruction "ONERR". — \$F8 pointeur de pile sauvegardé lors d'une erreur survenue avec "ONERR". |
| F9 | Valeur de ROT. |
| FA-FE | Adresses inutilisées. |
| FF | Adresse de début, utilisée pour la sauvegarde d'une chaîne, lors de l'utilisation de l'instruction "STR\$". |

VOTRE BIBLIOTHÈQUE INFORMATIQUE

• GUIDE DE PRODOS (P. Beaufils et W. Luther)

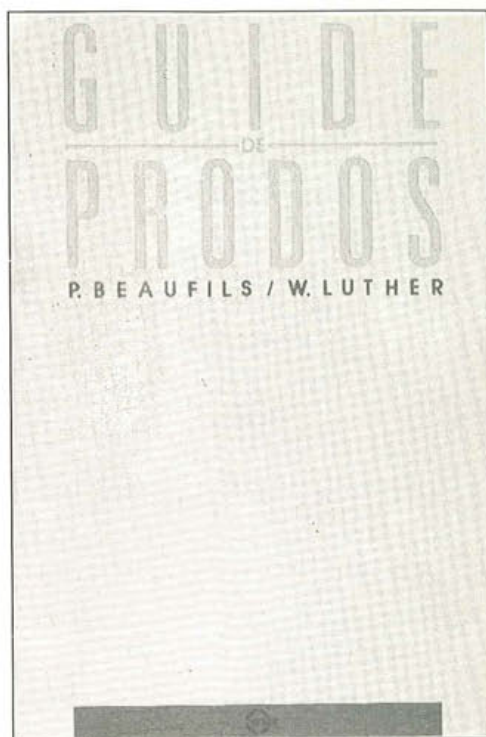
Dans leur courte introduction, les auteurs nous précisent que "ce livre est un cours de ProDOS, destiné à apprendre aux non-initiés l'utilisation des commandes de ce nouveau système d'exploitation Apple". On ne saurait les accuser de se montrer prétentieux : ils réussissent en effet — et fort bien ! —, non seulement à initier, mais à nous convaincre que ProDOS est à la fois un système sûr et sans surprises.

Ayant parfois la charge — ô combien délicate et difficile ! — de répondre à certaines questions des Lectrices et Lecteurs de *Tremplin Micro*, je ne surprendrai sans doute personne en souhaitant que tous les utilisateurs de ProDOS lisent ce guide, et de préférence sans en négliger aucun des intéressants chapitres. On a toujours quelque chose à apprendre dans un livre rédigé par des spécialistes... ce qui est le cas ici.

De plus, comme le disent les auteurs, "la programmation en langage machine est très facile et ce serait dommage de s'en priver"... et ils le prouvent dans les quelques exemples que je vous laisse le soin de découvrir.

Notons au passage que Pierre-Marc Beaufils est professeur agrégé de physique et Wolfram Luther docteur en mathématiques... ce qui n'empêche pas leur guide-cours d'être aussi clair et concis que possible.

(SYBEX, 6-8 Impasse du Curé, 75881 Paris Cedex 18)



• L'INFORMATISATION QUOTIDIENNE (ouvrage collectif du CREIS)

Le CREIS rassemble des enseignants et des chercheurs de toutes disciplines (et la Presse alors ?). L'Informatisation quotidienne s'adresse, à partir de l'examen de situations concrètes empruntées aux loisirs, à l'industrie, à l'agriculture, à l'école ou à l'hôpital, à tous ceux qui s'interrogent plus ou moins sur le devenir de notre société, face à l'envahissement de l'informatique.

J'ai repris à dessein certains des termes de la présentation de cet ouvrage collectif. Termes à mon avis excessifs puisqu'ils posent comme postulat que l'informatique est *envahissante*. Comme le fut, en d'autres temps et pour d'autres observateurs, la machine à vapeur...

Il reste que ce livre est intéressant et mérite d'être lu avec attention, ne serait-ce que pour éviter de tom-

ber dans des excès qu'il conviendrait alors de déplorer. Condamnés à l'informatique, les utilisateurs pourront-ils s'approprier ce nouvel outil ? C'est le sujet du dernier chapitre de *L'Informatisation quotidienne*. Je ne suis pas sûr que la réponse satisfasse les Lectrices et Lecteurs de *Tremplin Micro*. Pour eux, en effet, il n'existe qu'une solution : apprendre à maîtriser leur machine pour en tirer eux-mêmes de véritables satisfactions.

En effet, dans cette grande famille d'Apple-maniaques, nous savons apprécier le travail des experts et utilisons volontiers les logiciels qu'ils mettent au point, mais nous aimons aussi deviner ce qui se passe derrière des disquettes protégées qui n'en prennent alors que plus de valeur à nos yeux !

(Librairie Delagrave, 15 rue Soufflot, 75005 Paris).

• L'ANGLAIS COMMERCIAL PAR LES MOTS CROISÉS

Comment conjuguer récréation et

révision de votre anglais commercial ? En vous offrant le recueil des Editions Retz, puis en résolvant les 30 grilles de mots croisés et les 17 grilles de mots cachés que vous propose son auteur : M. A. Riccioli. Les définitions sont en français, mais les mots à trouver figurent parmi les 2800 utilisés dans le monde du commerce, des affaires et de la technologie. Toutes les réponses sont évidemment données dans le livre, mais il y a mieux : un glossaire regroupe tous les mots et expressions utilisés dans les définitions, ceci pour en connaître la traduction immédiate, si la mémoire fait défaut. On donne aussi une liste alphabétique des abréviations et sigles.

Michael Arthur Riccioli est chargé de cours à l'Institut britannique et à l'Institut catholique de Paris, chargé d'enseignement à l'I.U.T. de Paris V (CCIP). C'est aussi un lecteur de *Tremplin Micro* !

(Editions Retz, 2 rue du Roule, 75001 Paris).

Chasseur d'Images

Chaque mois,
le meilleur
de la
technique
et de la
pratique
photo !



Chez votre
marchand de journaux !