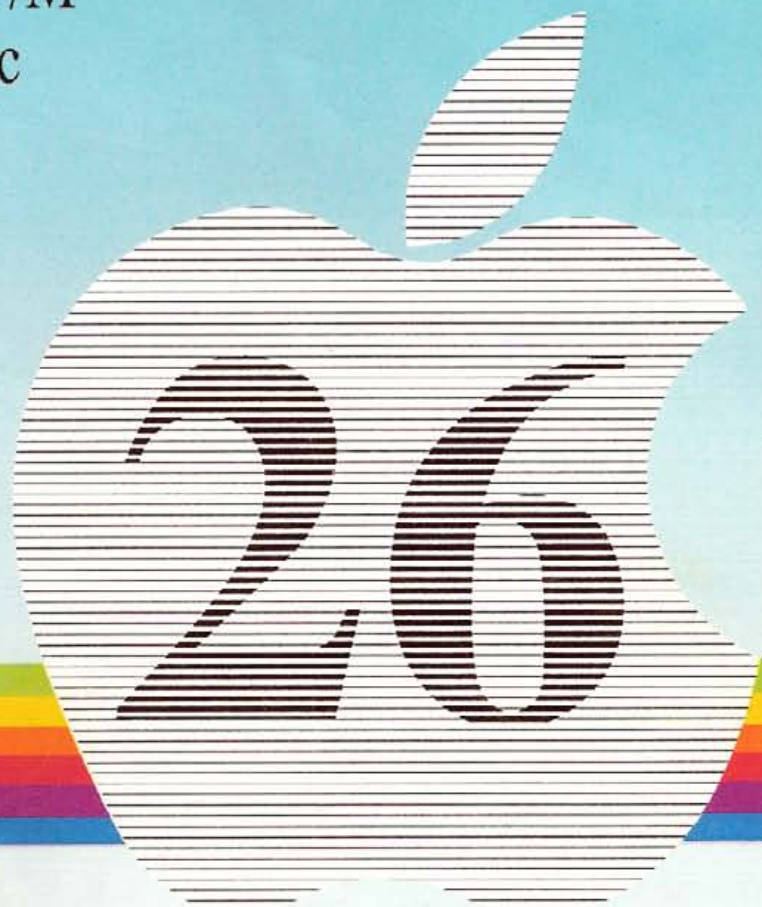
La revue francophone indépendante pour les utilisateurs des Apple ][+, //e, //e+, //c™ et Macintosh™

# pom's

- Disquettes spéciales avec BOOTDATA
- Lissajous : évolution de la courbe
- Fenêtres en HGR sur l'Apple //
- Tri de Chaînes sur Macintosh
- Carte Super Série et CP/M
- Courbes avec le MSBasic
- 68000's : un accessoire
- Startup Pascal complet
- Commandes ProDOS
- Programme éducatif
- Loupe pour HGR
- CALLRWTS




*NUMERO 26 - PRIX 40 F*

M2366-26-40 F

ISSN : 0294-6068

# SERVEURS?

Des dizaines et des dizaines d'Apple II ont déjà été transformés en serveurs par leurs propriétaires, avec l'un ou plusieurs des systèmes conçus pour exploiter les performances uniques de la carte Apple-Tell*.

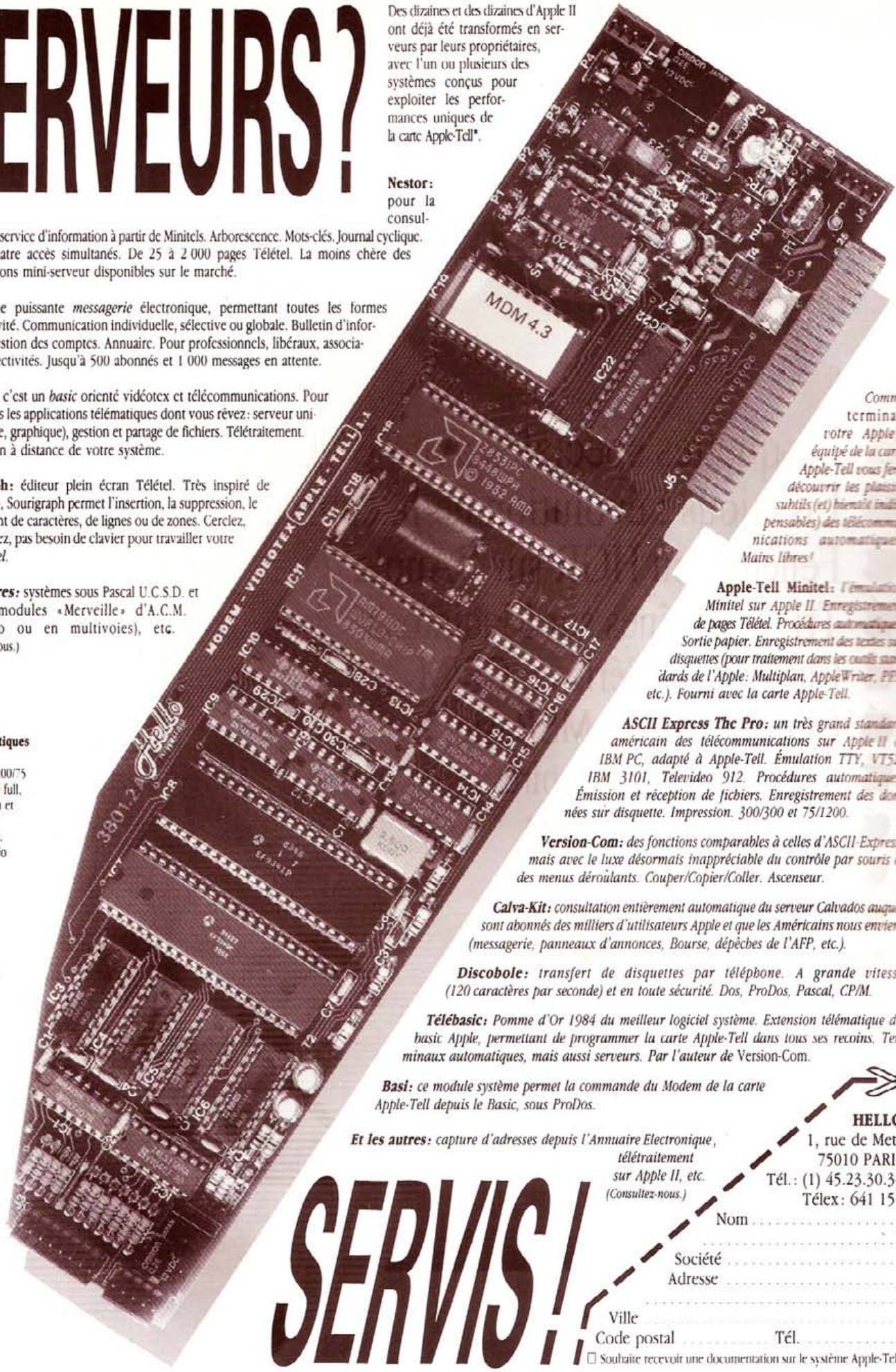
**Nestor:** pour la consultation d'un service d'information à partir de Minitels. Arborescence. Mots-clés. Journal cyclique. Jusqu'à quatre accès simultanés. De 25 à 2 000 pages Télétel. La moins chère des configurations mini-serveur disponibles sur le marché.

**Fakir:** une puissante *messagerie* électronique, permettant toutes les formes d'interactivité. Communication individuelle, sélective ou globale. Bulletin d'information. Gestion des comptes. Annuaire. Pour professionnels, libéraux, associations, collectivités. Jusqu'à 500 abonnés et 1 000 messages en attente.

**Télépom:** c'est un *basic* orienté vidéotex et télécommunications. Pour créer toutes les applications télématiques dont vous rêvez: serveur universel (texte, graphique), gestion et partage de fichiers. Télétraitement. Exploitation à distance de votre système.

**Sourigraph:** éditeur plein écran Télétel. Très inspiré de «MacPaint», Sourigraph permet l'insertion, la suppression, le déplacement de caractères, de lignes ou de zones. Cerclez, tirez, cliquez, pas besoin de clavier pour travailler votre *page Télétel*.

***Et les autres:*** systèmes sous Pascal U.C.S.D. et ProDos, modules «Merveille» d'A.C.M. (en mono ou en multivoies), etc. (Consultez-nous.)




*Comme terminal, votre Apple II équipé de la carte Apple-Tell vous fera découvrir les plaisirs subtils (et) bientôt indispensables) des télécommunications automatiques. Mains libres!*

**Apple-Tell Minitel:** *l'émulation Minitel sur Apple II. Enregistrement de pages Télétel. Procédures automatiques. Sortie papier. Enregistrement des textes sur disquettes (pour traitement dans les outils standards de l'Apple: Multiplan, AppleWriter, PFS, etc.). Fourni avec la carte Apple-Tell.*

***ASCII Express The Pro:*** *un très grand standard américain des télécommunications sur Apple II et IBM PC, adapté à Apple-Tell. Émulation TTY, VT52, IBM 3101, Televideo 912. Procédures automatiques. Émission et réception de fichiers. Enregistrement des données sur disquette. Impression. 300/300 et 75/1200.*

***Version-Com:*** *des fonctions comparables à celles d'ASCII-Express, mais avec le luxe désormais inappréciable du contrôle par souris et des menus déroulants. Couper/Copier/Coller. Ascenseur.*

***Calva-Kit:*** *consultation entièrement automatique du serveur Calvados auquel sont abonnés des milliers d'utilisateurs Apple et que les Américains nous envient (messagerie, panneaux d'annonces, Bourse, dépêches de l'AFP, etc.).*

***Discobole:*** *transfert de disquettes par téléphone. A grande vitesse (120 caractères par seconde) et en toute sécurité. Dos, ProDos, Pascal, CP/M.*

***Télébasic:*** *Pomme d'Or 1984 du meilleur logiciel système. Extension télématique du basic Apple, permettant de programmer la carte Apple-Tell dans tous ses recoins. Terminaux automatiques, mais aussi serveurs. Par l'auteur de* Version-Com.

***Basi:*** *ce module système permet la commande du Modem de la carte Apple-Tell depuis le Basic, sous ProDos.*

***Et les autres:*** *capture d'adresses depuis l'Annuaire Electronique, télétraitement sur Apple II, etc.* *(Consultez-nous.)*

***Caractéristiques générales:**
- modem 1 200/75 (Ccitt) et 300 full, numérotation et connexion automatiques.
- sorties vidéo composite (N&B) et Péritel (couleur).

# SERVIS!

**HELLO**
1, rue de Metz
75010 PARIS
Tél.: (1) 45.23.30.34
Télex: 641 155

Nom .....
Société .....
Adresse .....
Ville .....
Code postal ..... Tél. .....
☐ Souhaite recevoir une documentation sur le système Apple-Tell.

S.M.16

*Numéro 26*
*septembre-octobre 86*



**Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Directeur de la publication : Hervé Thiriez**

# De MacPaint à MacWrite : Pourquoi passer par le bureau ? "Raccourci" évite ce détour.

**Le chemin des écoliers**, tel est souvent le passage par le Bureau. Pour changer d'application, il suffit aujourd'hui d'appeler l'accessoire "Raccourci". Après avoir confirmé votre intention de quitter le programme en cours, une fenêtre de sélection apparaît et un simple double-clic vous fait passer de Basic à ScreenMaker, de MacPaint à Multiplan et, pourquoi pas, de MacForth à MacPoker…

## Des avantages :

**Sur un 128 Ko**, ne pas passer par le Finder c'est éviter des manipulations gourmandes en temps et en patience.

**Sur un 512 Ko**, les applications disposent de toute la place mémoire.

**Sur un Mac Plus**, les changements sont quasi-instantanés et les 768 Ko de mémoire cache sont disponibles.

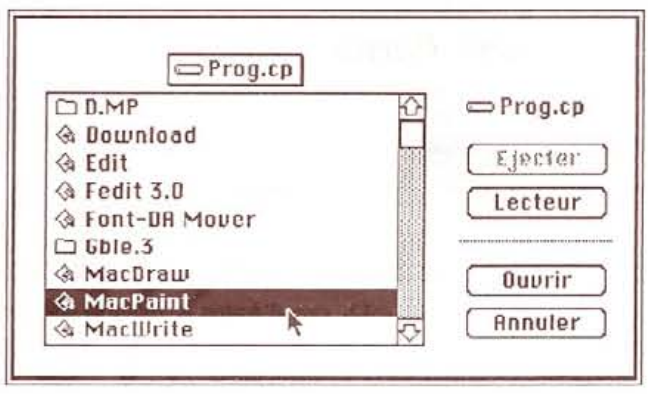
**Compatible avec Switcher :** vous pourrez changer l'une ou l'autre des applications gérées par Switcher directement par cet accessoire.

**Compatible disques durs :** même si votre disque dur est partagé via AppleTalk…

**Disponible** à tout moment, "Raccourci" est toujours utilisable car il ne nécessite pas une installation préalable des applications (sait-on toujours à l'avance quelles applications seront nécessaires à un travail donné ?).

Avec Pom's, travaillez *directement* !






**Disquette "Raccourci" - 200,00 F franco - Bon de commande page 74 - ©Éditions MEV**

Hervé Thiriez

# Éditorial

Serions-nous à l'étroit dans nos 140Ko ? L'adoption d'une nouvelle mise en pages et d'une typographie plus dense pour les listings nous permet de vous proposer un "Pom's" toujours plus riche, mais il devient difficile de mettre tous les programmes Apple // sur la même disquette. Sur celle du précédent numéro comme sur celle-ci, nous avons été conduits à supprimer les remarques des sources en assembleur, considérant qu'elles sont imprimées dans la revue et que leur absence ne gêne en rien l'utilisation. Nous ne doutons pas qu'entre toutes les REMs et plus de programmes, chacun aurait fait le même choix... Rappelons à ce sujet que les sources sont maintenant sauvegardés sous format TEXT pour une utilisation avec tous les assembleurs. Les problèmes qui proviennent de la différence de syntaxe entre assembleurs seront abordés dans un prochain article.

Notre stock d'articles en attente d'impression nous assure, pour de nombreux numéros, densité, qualité et originalité. Dans ces colonnes, prochainement, une élégante gestion d'adresses, un interpréteur doté entre autre d'un buffer clavier, l'& au secours de la souris, un écran virtuel style *MagicWindow*, un calculateur, des accessoires de bureau. En résumé : une véritable assurance chômage pour Apple // et Macintosh.

Pas de chômage non plus pour votre Z80 : J.-F. Rabasse s'est attaché à la communication en CP/M . La première partie de cette étude vous offre une commande pour contrôler de façon simple la carte série. Insistons sur le caractère autonome de chacune des deux parties ; désagréables en effet ces programmes qui ne fonctionnent qu'à la réception du énième numéro !

Dans ces pages, outre Basic, Pascal, et Assembleur, un programme pour nos tout-petits, illustrant l'utilisation de ToolKit d'Apple et le fort intéressant courrier de Michel Duroc à propos d'AppleWriter : vous avez dit diversité ?

Ordico de Roland Jost est une nouvelle disquette Pom's pour cruciverbistes, scrabbleurs et autres *joueurs de mots*. Au-delà de la qualité de programmation et de la rapidité de recherche, cette mini-base de données (d'aucuns diraient mini-BDD...) vaut par l'étendue de ses références dont la présentation en page 57 donne une idée.

Nous vous donnons rendez-vous dans nos pages de novembre avec peut-être quelques éléments concrets sur le nouvel Apple //...

**Ont collaboré à ce numéro :** Alexandre Avrane, Jean-Luc Bazanegue, Bruno Fénard, Jean-Michel Gourévitch, Olivier Herz, Alain Idelon-Ritton, Frédéric Ivsic, Gérard Michel, Daniel Mueller, Patrice Neveu, Dominique Ottello, Christian Piard, Jean-François Rabasse, Frédéric Rosay, Marianne Sutz, Hervé Thiricz, Julien Thomas, Norbert Vurlod.

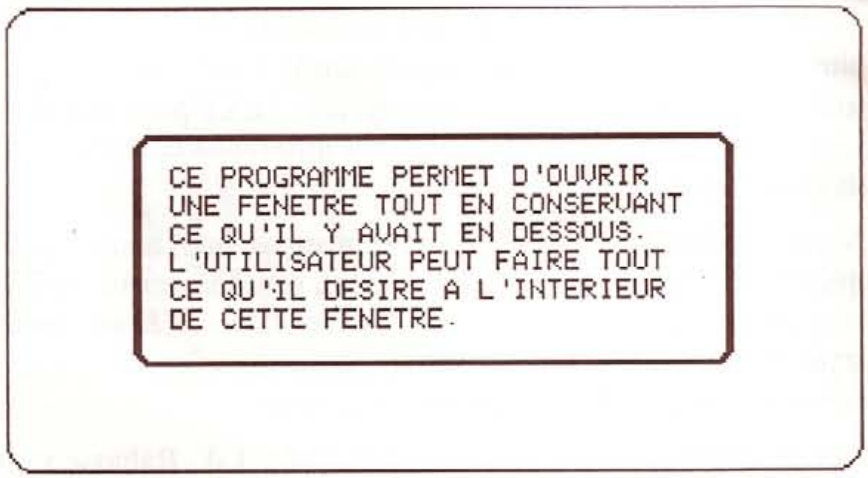
**Directeur de la publication, rédacteur en chef :** Hervé Thiriez.
**Rédacteurs :** Alexandre Avrane, Olivier Herz.
**Siège social :** Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Tél. : (1) 39.51.24.43.
**Publicité :** Éditions MEV.
**Diffusion :** N.M.P.P.
**Impression :** Rosay - 47, avenue de Paris - 94300 Vincennes. Tél. : (1) 43.28.18.63.

# Fenêtrage : Des fenêtres en HGR

Frédéric Rosay

Ce programme, sans prétendre égaler les performances du Mac, permet de donner une touche très professionnelle à vos programmes en leur permettant d'utiliser le principe des fenêtres graphiques.

Il permet d'ouvrir simultanément jusqu'à deux fenêtres graphiques (ce qui est suffisant dans la plupart des cas), la deuxième venant se superposer à la première qui est elle-même sauvegardée puis redessinée.




## Syntaxe

FENETRAGE est conçu pour être appelé à partir d'un programme Applesoft, et utilise l'ampersand (&) comme indicatif d'appel.

### & OUVRE haut, bas, droite, gauche

(& OUVRE 56, 136, 40, 239 ou & OUVRE HT,136,DR,GH par exemple) ouvre une nouvelle fenêtre d'après les coordonnées graphiques. Le programme dessine automatiquement un cadre autour de la fenêtre.
Les valeurs peuvent être données de manière statique ou par l'intermédiaire de variables Applesoft. Elles doivent être comprises dans la plage 0 à 191 en vertical, 0 à 279 pour l'horizontal. Si haut < bas, ou droite < gauche, le programme inverse automatiquement les coordonnées. Les dimensions minimum d'une fenêtre sont : 10 points verticaux, 14 points horizontaux.

### & FERME

ferme la dernière fenêtre ouverte en restituant l'affichage de la première fenêtre.

Le programme utilisateur peut écrire sur les fenêtres graphiques par tout moyen à sa convenance (HPLOT, DRAW, etc.) mais a la responsabilité de ne pas dépasser les bornes de la fenêtre qu'il a définie.

Afin de minimiser l'occupation en mémoire de la routine FENETRAGE, celui-ci ne contrôle pas certains cas d'erreurs : par exemple, l'ouverture d'une troisième fenêtre graphique est autorisée mais détruira la première fenêtre qui ne pourra plus être restaurée.

## Fonctionnement

Le programme tourne indifféremment sous DOS 3.3 ou ProDOS, et quelle que soit la configuration mémoire de votre Apple (compatibilité...). Très schématiquement, lorsqu'il est appelé, les opérations suivantes sont entreprises :

- si une ouverture est demandée :
  - contrôle de la syntaxe
  - sauvegarde de l'affichage graphique actuel, soit en carte langage (si DOS 3.3 et une carte langage), soit en dessous d'Himem
  - dessin du cadre autour des coordonnées de la nouvelle fenêtre
  - mise à blanc dans le cadre de la nouvelle fenêtre
- si une fermeture est demandée :
  - restauration de l'ancienne fenêtre

Il se loge à partir de l'adresse $9200 et manipule Himem ainsi que le vecteur de l'ampersand.

**Fichiers utilisés**

**FENETRAGE** code objet exécutable

**FENETRAGE.S** source en Big Mac

**FENETRAGE.DEMO** démonstration en Applesoft

**HGRECR.LIB** routine d'Eric Ringot (Pom's 15)

Le programme FENETRAGE.DEMO utilise la routine HGRECR d'Eric Ringot parue dans le numéro 15 de Pom's pour écrire facilement des caractères en HGR sans avoir à utiliser une moisson de HPLOT.

## Programme 'FENETRE.DEMO'

```
1  REM   FENETRAGE.DEMO
5  PRINT  CHR$ (17)
10  PRINT  CHR$ (4)"BLOAD HGRECR.LIB,A$8000
   "
20  PRINT  CHR$ (4)"BRUN FENETRAGE"
30 HM = 32768
35  HOME
40  HGR : POKE 49234,0
50  HCOLOR= 3: ROT= 0: SCALE= 1
60  & OUVRE56,136,40,239
70 A$ = "CE PROGRAMME PERMET D'OUVRIR ": CA
   LL HM,A$,50,70
80 A$ = "UNE FENETRE TOUT EN CONSERVANT": C
   ALL HM,A$,50,80
90 A$ = "CE QU'IL Y AVAIT EN DESSOUS.": CAL
   L HM,A$,50,90
100 A$ = "L'UTILISATEUR PEUT FAIRE TOUT": C
   ALL HM,A$,50,100
110 A$ = "CE QU'IL DESIRE A L'INTERIEUR": C
   ALL HM,A$,50,110
120 A$ = "DE CETTE FENETRE.": CALL HM,A$,50
   ,120
125 A$ = "APPUYEZ SUR TOUCHE, S.V.P.": CALL
    HM,A$,10,185
130  GET A$: & FERME
140  & OUVRE30,161,80,199
150 A$ = "SI L'APPLICATION": CALL HM,A$,90,
   40
160 A$ = "SORT DE LA FENETRE": CALL HM,A$,9
   0,50
170 A$ = "ALORS, A LA": CALL HM,A$,90,60
180 A$ = "FERMETURE DE CETTE": CALL HM,A$,9
   0,70
190 A$ = "DERNIERE, TOUT CE": CALL HM,A$,90
   ,80
200 A$ = "QUI A ETE MODIFIE": CALL HM,A$,90
   ,90
210 A$ = "EN DEHORS DE LA": CALL HM,A$,90,1
   00
220 A$ = "FENETRE RESTERA": CALL HM,A$,90,1
   10
230 A$ = "TEL QUEL.": CALL HM,A$,90,120
240 A$ = "PAR EXEMPLE:": CALL HM,A$,90,140
250  GET A$: & OUVRE10,175,70,209
260  FOR J = 7 TO 180 STEP 10: FOR I = 7 TO
    279 STEP 15
270 A$ = "X": CALL HM,A$,I,J
280  NEXT I,J
290  GET A$: & FERME: GET A$
300  HGR : POKE 49234,0
310 A$ = "APPUYEZ SUR TOUCHE, S.V.P.": CALL
    HM,A$,10,185
320  & OUVRE10,170,30,249
330 A$ = "L A  S Y N T A X E": CALL HM,A$,9
   0,20
340 A$ = "CE PROGRAMME EST FAIT POUR ETRE":
    CALL HM,A$,40,40
350 A$ = "UTILISE EN BASIC, DONC POSSEDE":
   CALL HM,A$,40,50
360 A$ = "UNE SYNTAXE TRES SIMPLE.": CALL H
   M,A$,40,60
370 A$ = "POUR OUVRIR UNE FENETRE, IL": CAL
   L HM,A$,40,70
380 A$ = "SUFFIT DE FAIRE:": CALL HM,A$,40,
   80
390 A$ = "- & OUVRE HAUT,BAS,DROITE,GAUCHE"
   : CALL HM,A$,40,90
400 A$ = "SI HAUT EST INFERIEUR A BAS": CAL
   L HM,A$,50,100
410 A$ = "LE PROGRAMME INVERSE LES": CALL H
   M,A$,50,110
420 A$ = "DEUX COORDONNEES (DE MEME POUR":
   CALL HM,A$,50,120
430 A$ = "DROITE ET GAUCHE).": CALL HM,A$,5
   0,130
440 A$ = "SI BAS-HAUT<10 ALORS BAS=HAUT+10"
   : CALL HM,A$,50,140
450 A$ = "DE MEME SI GAUCHE-DROITE<14 ALORS
   ": CALL HM,A$,50,150
460 A$ = "GAUCHE=DROITE+14": CALL HM,A$,50,
   160
470  GET A$: & FERME
480  & OUVRE10,170,30,249
490 A$ = "L A  S Y N T A X E": CALL HM,A$,9
   0,20
500 A$ = "CECI POUR DES RAISONS DE COMMODIT
   ES": CALL HM,A$,40,40
510 A$ = "AU NIVEAU DU PROGRAMME ET DE": CA
   LL HM,A$,40,50
520 A$ = "L'UTILISATION (ON NE PEUT RIEN":
   CALL HM,A$,40,60
530 A$ = "FAIRE DANS UNE FENETRE PLUS PETIT
   E)": CALL HM,A$,40,70
540 A$ = "POUR REFERMER UNE FENETRE IL SUFF
   IT": CALL HM,A$,40,90
550 A$ = "DE FAIRE:": CALL HM,A$,40,100
560 A$ = "- & FERME": CALL HM,A$,40,110
570 A$ = "ALORS LA DERNIERE FENETRE OUVERTE
   ": CALL HM,A$,50,120
580 A$ = "EST REFERMEE.": CALL HM,A$,50,130
590  GET A$: & FERME
600  & OUVRE10,170,90,189
610 A$ = "FENETRAGE EST LE": CALL HM,A$,95,
   20
```

```
620 A$ = "PROGRAMME DE": CALL HM,A$,95,30
630 A$ = "DE FENETRE,": CALL HM,A$,95,40
640 A$ = "FENETRAGE.S EST": CALL HM,A$,95,5
    0
650 A$ = "LE CODE BIG MAC.": CALL HM,A$,95,
    60
660 A$ = "LE PROGRAMME": CALL HM,A$,95,80
670 A$ = "PEUT AUSSI": CALL HM,A$,95,90
680 A$ = "TOURNER SOUS": CALL HM,A$,95,100
690 A$ = "PRODOS.": CALL HM,A$,95,110
700  GET A$: & FERME
710  & OUVRE20,90,20,259
720 A$ = "CONCU ET REALISE PAR:": CALL HM,A
    $,25,30
730 A$ = "ROZAY FREDERIC": CALL HM,A$,25,40
750 A$ = "UN GRAND MERCI A ERICK RINGOT": C
    ALL HM,A$,25,70
760 A$ = "POUR LE PROGRAMME HGRECR": CALL H
    M,A$,25,80
770  GET A$: & FERME
800  FOR I = 1 TO 1000: NEXT : TEXT
```

## Source 'FENETRAGE.S'

### Assembleur Big Mac

```
 1
 2 ********************************
 3 *                              *
 4 * AFFICHE / EFFACE UNE FENETRE *
 5 * EN GRAPHIQUE HAUTE RESOLUTION *
 6 *                              *
 7 *  PAR F. ROZAY                *
 8 *                              *
 9 ********************************
10
11
12          LST  OFF
13
14          ORG  $9200
15
16 DEP      =    $06        Ptr pour dessin fenetre
17 HBAS     =    $26        Adresse de la ligne HGR,
              obtenue par HPOSN
18 LINNUM   =    $50        entier extrait de FAC par
              GETADR
19 HIMEM    =    $73        Valeur de HIMEM
20 CHRGET   =    $B1
21 CHRGOT   =    $B7
22 XD7      =    $E5        No de l'octet dans lequel se
              trouve le point HGR
23 HPAGE    =    $E6        No de la page HGR utilisée
              ($20:page1 $40:page2)
24
25 AMPERV   =    $03F6      Adresse du sous-programme à
26 *                        utiliser si & est exécutée
27
28 ERROR    =    $D412      Emission d'un message
              d'erreur (code dans X-REG)
29 DATA     =    $D995      Positionne TXTPTR sur
              l'instruction BASIC suivante
30 FRMNUM   =    $DD67      Prend le chiffre ou la valeur
31 *             de la variable pointée par TXTPTR et le
              range dans FAC
32 GETBYTC  =    $E6F5      meme chose que FRMNUM mais
              avec des chiffres <256
33 GETADR   =    $E752      Transforme FAC en un entier à
34 *             2 octets et le range dans LINNUM
35 HPOSN    =    $F411      Donne l'adresse de la ligne
36 * avec les coordonnées du point (A:VERT X:H-LO Y:H-HI)
37
38 ********************
39 *   BRANCHE LE &   *
40 * SUR LE PROGRAMME *
41 *     PRINCIPAL    *
42 ********************
43
44 HIGH       LDA  #<MAIN      partie basse et partie haute
45           STA  AMPERV      de l'adresse du début du
               programme, les range
46           LDA  #>MAIN      dans AMPERV pour que le
               programme soit exécuté
47           STA  AMPERV+1    si un '&' est rencontré dans
               le programme BASIC
48           LDA  #<HIGH      change la valeur du HIMEM
               pour que le programme
49           STA  HIMEM       ne soit pas ecrasé par les
               variables du programme
50           LDA  #>HIGH      BASIC
51           STA  HIMEM+1
52           LDA  $BF00       voit si on est sous ProDOS
53           CMP  #$4C
54           BEQ  YAPRODOS
55           BIT  $C083       voit si on a une carte
               langage
56           BIT  $C083
57           LDA  $FFFF
58           CMP  #$FA
59           BNE  YA16K
60 YAPRODOS  LDA  #$70        ProDOS alors on sauvera
61           STA  HSAUVE      l'image à partir de $7000
62           BIT  $C081
63           LDA  #$FF
64           STA  HIMEM       et on rechange la valeur du
65           LDA  #$6F        HIMEM (->$6FFF)
66           STA  HIMEM+1
67           RTS
68 YA16K     LDA  #$E0        si on a une carte langage
69           STA  HSAUVE      alors on sauvera l'image à
70           BIT  $C081       partir de $E000
71           RTS
72
73 * PROGRAMME PRINCIPAL
74
75 MAIN       EQU  *
76           LDY  #0
77           JSR  CHRGOT
78 MAIN1     CMP  CMD,Y       prend ce qu'il y a après le
               '&' et le compare
79           BNE  AUTRCMD1    aux nouvelles instructions
80           JSR  CHRGET      ici on compare avec 'OUVRE'
81           INY
82           CPY  #4
83           BNE  MAIN1
84           JSR  GETBYTC     prend la première coordonnée
85           CPX  #192        compare avec le bas de
               l'écran
86           BCS  ILLEGAL     si c'est supérieur alors
               'ILLEGAL QUANTITY'
87           STX  TABLE
88           JSR  GETBYTC     fait de meme avec la deuxième
               coordonnée
89           CPX  #192
90           BCS  ILLEGAL
91           STX  TABLE+1
92           JSR  CHRGET
93           JSR  FRMNUM      cette fois ci c'est plus
               sérieux
94           JSR  GETADR      les coordonnées peuvent
               dépasser 256
95           JSR  SEPTAL
96           CPX  #40         toujours cette comparaison
               pour voir si on sort
97           BCS  ILLEGAL     de la fenetre
98           STX  TABLE+2
99           JSR  CHRGET
100          JSR  FRMNUM      et encore pour la dernière
               coordonnée
101          JSR  GETADR
102          JSR  SEPTAL
103          CPX  #40
```

```
104             BCS  ILLEGAL
105             STX  TABLE+3
106             JSR  VERIFIE
107             JSR  COORTK
108             JSR  SAVE        on sauve ce qu'il y a sous la
                   fenetre
109             JSR  COORTK
110             JSR  CADRE       et on affiche la fenetre
111
112             JMP  DATA        retour au programme BASIC
113
114 AUTRCMD1  EQU  *
115             LDY  #5
116             JSR  CHRGOT
117 MAIN2       CMP  CMD,Y       cette fois ci on compare avec
                   la commande 'FERME'
118             BNE  AUTRCMD2
119             JSR  CHRGET
120             INY
121             CPY  #9
122             BNE  MAIN2
123             JSR  COORTK
124             JSR  RECUP       et on referme tout simplement
125
126             JMP  DATA        retour au programme BASIC
127
128 ILLEGAL     LDX  #$35
129             JMP  ERROR       produit un 'ILLEGAL QUANTITY
                   ERROR'
130
131 AUTRCMD2 LDX  #$10
132             JMP  ERROR       produit un 'SYNTAX ERROR'
133
134 *+++++++++++++++*
135 *                *
136 *   FENETRE      *
137 *                *
138 *+++++++++++++++*
139
140 CADRE      CLC
141             LDA  BAS
142             SBC  #3
143             STA  BAS
144             LDA  HAUT
145             STA  BUFF
146             LDA  HAUT
147             LDX  #0
148             LDY  #0
149             JSR  HPOSN
150             LDY  GAUCHE
151             LDA  (HBAS),Y    affiche d'abord l'octet dans
                   le coin gauche en haut
152             ORA  #$70
153             STA  (HBAS),Y
154             INY
155             LDA  #$7F
156 CADRE0      STA  (HBAS),Y    puis va jusqu'au bout de la
                   ligne en affichant
157             INY              un trait
158             CPY  DROITE
159             BNE  CADRE0
160             LDA  (HBAS),Y
161             ORA  #$07        affiche l'octet en haut à
                   droite
162             STA  (HBAS),Y
163             INC  BUFF
164             LDA  BUFF
165             LDX  #0
166             LDY  #0
167             JSR  HPOSN
168             LDY  GAUCHE      fait de meme avec la seconde
                   ligne
169             LDA  (HBAS),Y
170             ORA  #$78
171             STA  (HBAS),Y
172             INY
173             LDA  #$7F
174 CADRE1      STA  (HBAS),Y
175             INY
176             CPY  DROITE
177             BNE  CADRE1      affiche le trait
178             LDA  (HBAS),Y
179             ORA  #$0F
180             STA  (HBAS),Y
181             INC  BUFF        et la partie droite
182             LDA  BUFF
183             LDX  #0
184             LDY  #0
185             JSR  HPOSN
186             LDY  GAUCHE
187             LDA  (HBAS),Y    et encore avec la troisième
                   ligne
188             AND  #$1F
189             ORA  #$1C
190             STA  (HBAS),Y
191             INY
192             LDA  #$00
193 CADRE4      STA  (HBAS),Y    maintenant remplit avec des
                   octets 'vides'
194             INY              pour avoir une fenetre noire
                   à l'intérieur
195             CPY  DROITE
196             BNE  CADRE4
197             LDA  (HBAS),Y
198             AND  #$F8        affiche la partie du cadre
                   qui se touve à droite
199             ORA  #$1C
200             STA  (HBAS),Y
201             INC  BUFF
202             LDA  BUFF
203             LDX  #0
204             LDY  #0
205             JSR  HPOSN
206             LDY  GAUCHE
207             LDA  (HBAS),Y
208             AND  #$0F
209             ORA  #$0E
210             STA  (HBAS),Y
211             INY
212             LDA  #$0
213 CADRE5      STA  (HBAS),Y    le vide
214             INY
215             CPY  DROITE
216             BNE  CADRE5
217             LDA  (HBAS),Y
218             AND  #$F0
219             ORA  #$38
220             STA  (HBAS),Y
221 CADRE2      INC  BUFF        cette fois ci on affiche
                   toute la partie
222             LDA  BUFF        verticale de la fenetre
223             LDX  #0
224             LDY  #0
225             JSR  HPOSN
226             LDY  GAUCHE
227             LDA  #$07
228             STA  (HBAS),Y    avec d'abord l'octet à gauche
229             INY
230             LDA  #$0
231 CADRE3      STA  (HBAS),Y
232             INY              puis le milieu
233             CPY  DROITE
234             BNE  CADRE3
235             LDA  #$70
236             STA  (HBAS),Y    et enfin l'octet à droite
237             LDX  BUFF
238             CPX  BAS
239             BNE  CADRE2
240             INC  BUFF
241             LDA  BUFF
242             LDX  #0
243             LDY  #0          une fois le milieu fini on
                   fait la meme
244             JSR  HPOSN       chose qu'en haut de la
                   fenetre mais en sens
245             LDY  GAUCHE      inverse
246             LDA  (HBAS),Y
247             AND  #$0F
```

```
248             ORA  #$0E
249             STA  (HBAS),Y
250             INY
251             LDA  #$0
252 CADRE6      STA  (HBAS),Y
253             INY
254             CPY  DROITE
255             BNE  CADRE6
256             LDA  (HBAS),Y
257             AND  #$F0
258             ORA  #$38
259             STA  (HBAS),Y
260             INC  BUFF
261             LDA  BUFF
262             LDX  #0
263             LDY  #0
264             JSR  HPOSN
265             LDY  GAUCHE
266             LDA  (HBAS),Y
267             AND  #$1F
268             ORA  #$1C
269             STA  (HBAS),Y
270             INY
271             LDA  #$00
272 CADRE7      STA  (HBAS),Y
273             INY
274             CPY  DROITE
275             BNE  CADRE7
276             LDA  (HBAS),Y
277             AND  #$F8
278             ORA  #$1C
279             STA  (HBAS),Y
280             INC  BUFF
281             LDA  BUFF
282             LDX  #0
283             LDY  #0
284             JSR  HPOSN
285             LDY  GAUCHE
286             LDA  (HBAS),Y
287             ORA  #$78
288             STA  (HBAS),Y
289             INY
290             LDA  #$7F
291 CADRE9      STA  (HBAS),Y
292             INY
293             CPY  DROITE
294             BNE  CADRE9
295             LDA  (HBAS),Y
296             ORA  #$0F
297             STA  (HBAS),Y
298             INC  BUFF
299             LDA  BUFF
300             LDX  #0
301             LDY  #0
302             JSR  HPOSN
303             LDY  GAUCHE
304             LDA  (HBAS),Y
305             ORA  #$70
306             STA  (HBAS),Y
307             INY
308             LDA  #$7F
309 CADRE8      STA  (HBAS),Y
310             INY
311             CPY  DROITE
312             BNE  CADRE8
313             LDA  (HBAS),Y
314             ORA  #$07
315             STA  (HBAS),Y
316             RTS
317
318 *+++++++++++++++++*
319 *                 *
320 *      SAUVE      *
321 *   IMAGE->RAM    *
322 *                 *
323 *+++++++++++++++++*
324
325 SAVE        BIT  $C081      sauve ce qu'il y a sous la
                                fenetre dans la page
326             LDA  HAUT       choisie au début du programme
327             STA  BUFF
328             INC  BAS
329 SAVE1       LDA  BUFF
330             LDX  #0
331             LDY  #0
332             JSR  HPOSN
333             LDA  HBAS
334             STA  DEP
335             LDA  HBAS+1
336             STA  DEP+1
337             LDA  HPAGE
338             PHA
339             LDA  HSAUVE
340             STA  HPAGE
341             LDA  BUFF
342             LDX  #0
343             LDY  #0
344             JSR  HPOSN
345             PLA
346             STA  HPAGE
347             LDY  GAUCHE
348             DEY
349 SAVE2       INY
350             LDA  (DEP),Y
351             STA  (HBAS),Y
352             CPY  DROITE
353             BNE  SAVE2
354             INC  BUFF
355             LDX  BUFF
356             CPX  BAS
357             BNE  SAVE1
358             RTS
359
360 *+++++++++++++++++*
361 *                 *
362 *    RECUPERE     *
363 *   RAM->IMAGE    *
364 *                 *
365 *+++++++++++++++++*
366
367 RECUP       LDA  HAUT       récupère ce qu'il y avait
                                sous la fenetre
368             STA  BUFF
369             INC  BAS
370 RECUP1      BIT  $C081
371             LDA  BUFF
372             LDX  #0
373             LDY  #0
374             JSR  HPOSN
375             LDA  HBAS
376             STA  DEP
377             LDA  HBAS+1
378             STA  DEP+1
379             LDA  HPAGE
380             PHA
381             LDA  HSAUVE
382             STA  HPAGE
383             LDA  BUFF
384             LDX  #0
385             LDY  #0
386             JSR  HPOSN
387             PLA
388             STA  HPAGE
389             LDY  GAUCHE
390             DEY
391 RECUP2      INY
392             BIT  $C083
393             BIT  $C083
394             LDA  (HBAS),Y
395             STA  (DEP),Y
396             CPY  DROITE
397             BNE  RECUP2
398             INC  BUFF
399             LDX  BUFF
400             CPX  BAS
401             BNE  RECUP1
402             BIT  $C081
403             RTS
```

```
404
405 *+++++++++++++++++*
406 *                 *
407 *     SEPTAL      *
408 *    DIVISION     *
409 *      PAR 7      *
410 *                 *
411 *+++++++++++++++++*
412
413 SEPTAL   LDA  #0          divise les coordonnées
                horizontales par 7
414          LDX  LINNUM      pour obtenir le numéro de
                l'octet dans lequel
415          LDY  LINNUM+1    se trouve le point HGR
416          JSR  HPOSN
417          LDX  XD7
418          RTS
419
420 * PREND LES
421 * COORDONNEES
422 * POUR LA FENETRE
423
424 COORTK   LDA  TABLE
425          STA  HAUT
426          LDA  TABLE+1
427          STA  BAS
428          LDA  TABLE+2
429          STA  GAUCHE
430          LDA  TABLE+3
431          STA  DROITE
432          RTS
433
434 *+++++++++++++++*
435 *               *
436 *   VERIFIE     *
437 *               *
438 *+++++++++++++++*
439
440 VERIFIE  LDA  TABLE+1    compare les 2 coordonnées
                verticales
441          CMP  TABLE       les arrange à la valeur la
                plus petite de la
442          BCC  INVERSE     fenetre ou les inverse si la
                première est
443 VERIF2   LDA  TABLE+1     supérieure à la seconde
444          SEC
445          SBC  TABLE
446          CMP  #8
447          BCC  RECALE
448
449 VERIF1   LDA  TABLE+3     fait de meme avec les
                coordonnées horizontales
450          SEC
451          SBC  TABLE+2
452          BMI  INVERSE1
453          CMP  #2
454          BCC  RECALE1
455
456          RTS
457
458 INVERSE  LDA  TABLE
459          LDX  TABLE+1
460          STA  TABLE+1
461          STX  TABLE
462          JMP  VERIF2
463
464 RECALE   LDA  TABLE
465          CLC
466          ADC  #8
467          STA  TABLE+1
468          JMP  VERIF1
469
470 INVERSE1 LDA  TABLE+2
471          LDX  TABLE+3
472          STA  TABLE+3
473          STX  TABLE+2
474          JMP  VERIF1
475
476 RECALE1  LDA  TABLE+2
477          CLC
478          ADC  #2
479          STA  TABLE+3
480          RTS
481
482 TABLE    HEX  00000000
483 CMD      ASC  'OUVREFERME'
484 HAUT     HEX  00
485 BAS      HEX  00
486 GAUCHE   HEX  00
487 DROITE   HEX  00
488 BUFF     HEX  00
489 HSAUVE   HEX  00
```

# Récapitulation 'FENETRAGE'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE FENETRAGE,A$9200,L$376*

```
9200- A9 40 8D F6 03 A9 92 8D
9208- F7 03 A9 00 85 73 A9 92
9210- 85 74 AD 00 BF C9 4C F0
9218- 0D 2C 83 C0 2C 83 C0 AD
9220- FF FF C9 FA D0 11 A9 70
9228- 8D 75 95 2C 81 C0 A9 FF
9230- 85 73 A9 6F 85 74 60 A9
9238- E0 8D 75 95 2C 81 C0 60
9240- A0 00 20 B7 00 D9 66 95
9248- D0 55 20 B1 00 C8 C0 04
9250- D0 F3 20 F5 E6 E0 C0 B0
9258- 62 8E 62 95 20 F5 E6 E0
9260- C0 B0 58 8E 63 95 20 B1
9268- 00 20 67 DD 20 52 E7 20
9270- E8 94 E0 28 B0 45 8E 64
9278- 95 20 B1 00 20 67 DD 20
9280- 52 E7 20 E8 94 E0 28 B0
9288- 32 8E 65 95 20 0D 95 20
9290- F4 94 20 45 94 20 F4 94
9298- 20 C5 92 4C 95 D9 60 A0
92A0- 05 20 B7 00 D9 66 95 D0
92A8- 17 20 B1 00 C8 C0 09 D0
92B0- F3 20 F4 94 20 92 94 4C
92B8- 95 D9 60 A2 35 4C 12 D4
92C0- A2 10 4C 12 D4 18 AD 71
92C8- 95 E9 03 8D 71 95 AD 70
92D0- 95 8D 74 95 AD 70 95 A2
92D8- 00 A0 00 20 11 F4 AC 72
92E0- 95 B1 26 09 70 91 26 C8
92E8- A9 7F 91 26 C8 CC 73 95
92F0- D0 F8 B1 26 09 07 91 26
92F8- EE 74 95 AD 74 95 A2 00
9300- A0 00 20 11 F4 AC 72 95
9308- B1 26 09 78 91 26 C8 A9
```

SI L'APPLICATION SORT DE LA FENETRE ALORS, A LA FERMETURE DE CETTE DERNIERE, TOUT CE QUI A ETE MODIFIE EN DEHORS DE LA FENETRE RESTERA TEL QUEL.

```
9310- 7F 91 26 C8 CC 73 95 D0
9318- F8 B1 26 09 0F 91 26 EE
9320- 74 95 AD 74 95 A2 00 A0
9328- 00 20 11 F4 AC 72 95 B1
9330- 26 29 1F 09 1C 91 26 C8
9338- A9 00 91 26 C8 CC 73 95
9340- D0 F8 B1 26 29 F8 09 1C
9348- 91 26 EE 74 95 AD 74 95
9350- A2 00 A0 00 20 11 F4 AC
9358- 72 95 B1 26 29 0F 09 0E
9360- 91 26 C8 A9 00 91 26 C8
9368- CC 73 95 D0 F8 B1 26 29
9370- F0 09 38 91 26 EE 74 95
9378- AD 74 95 A2 00 A0 00 20
9380- 11 F4 AC 72 95 A9 07 91
9388- 26 C8 A9 00 91 26 C8 CC
9390- 73 95 D0 F8 A9 70 91 26
9398- AE 74 95 EC 71 95 D0 D5
93A0- EE 74 95 AD 74 95 A2 00
93A8- A0 00 20 11 F4 AC 72 95
93B0- B1 26 29 0F 09 0E 91 26
93B8- C8 A9 00 91 26 C8 CC 73
93C0- 95 D0 F8 B1 26 29 F0 09
93C8- 38 91 26 EE 74 95 AD 74
93D0- 95 A2 00 A0 00 20 11 F4
93D8- AC 72 95 B1 26 29 1F 09
93E0- 1C 91 26 C8 A9 00 91 26
93E8- C8 CC 73 95 D0 F8 B1 26
93F0- 29 F8 09 1C 91 26 EE 74
93F8- 95 AD 74 95 A2 00 A0 00
9400- 20 11 F4 AC 72 95 B1 26
9408- 09 78 91 26 C8 A9 7F 91
9410- 26 C8 CC 73 95 D0 F8 B1
9418- 26 09 0F 91 26 EE 74 95
9420- AD 74 95 A2 00 A0 00 20
9428- 11 F4 AC 72 95 B1 26 09
9430- 70 91 26 C8 A9 7F 91 26
9438- C8 CC 73 95 D0 F8 B1 26
9440- 09 07 91 26 60 2C 81 C0
9448- AD 70 95 8D 74 95 EE 71
9450- 95 AD 74 95 A2 00 A0 00
9458- 20 11 F4 A5 26 85 06 A5
9460- 27 85 07 A5 E6 48 AD 75
9468- 95 85 E6 AD 74 95 A2 00
9470- A0 00 20 11 F4 68 85 E6
9478- AC 72 95 88 C8 B1 06 91
9480- 26 CC 73 95 D0 F6 EE 74
9488- 95 AE 74 95 EC 71 95 D0
9490- C0 60 AD 70 95 8D 74 95
9498- EE 71 95 2C 81 C0 AD 74
94A0- 95 A2 00 A0 00 20 11 F4
94A8- A5 26 85 06 A5 27 85 07
94B0- A5 E6 48 AD 75 95 85 E6
94B8- AD 74 95 A2 00 A0 00 20
94C0- 11 F4 68 85 E6 AC 72 95
94C8- 88 C8 2C 83 C0 2C 83 C0
94D0- B1 26 91 06 CC 73 95 D0
94D8- F0 EE 74 95 AE 74 95 EC
94E0- 71 95 D0 B7 2C 81 C0 60
94E8- A9 00 A6 50 A4 51 20 11
94F0- F4 A6 E5 60 AD 62 95 8D
94F8- 70 95 AD 63 95 8D 71 95
9500- AD 64 95 8D 72 95 AD 65
9508- 95 8D 73 95 60 AD 63 95
9510- CD 62 95 90 19 AD 63 95
9518- 38 ED 62 95 C9 08 90 1D
9520- AD 65 95 38 ED 64 95 30
9528- 20 C9 02 90 2B 60 AD 62
9530- 95 AE 63 95 8D 63 95 8E
9538- 62 95 4C 15 95 AD 62 95
9540- 18 69 08 8D 63 95 4C 20
9548- 95 AD 64 95 AE 65 95 8D
9550- 65 95 8E 64 95 4C 20 95
9558- AD 64 95 18 69 02 8D 65
9560- 95 60 00 00 00 00 4F 55
9568- 56 52 45 46 45 52 4D 45
9570- 00 00 00 00 00 00
```

## Récapitulation 'HGRECR.LIB'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE HGRECR.LIB,A$1000,L$39B*

```
1000- 20 58 FF BA CA CA 9A 18
1008- 68 69 A6 85 CE 68 69 00
1010- 85 CF A5 E4 48 A5 E7 48
1018- 20 BE DE 20 E3 DF 20 6C
1020- DD A0 00 B1 83 85 08 C8
1028- B1 83 85 06 C8 B1 83 85
1030- 07 20 BE DE 20 05 E1 A5
1038- A1 85 E0 A5 A0 85 E1 20
1040- F5 E6 86 E2 A9 01 85 E7
1048- A0 00 98 48 B1 06 48 A9
1050- 00 85 E4 A9 3F 20 58 FF
1058- 50 26 A9 7F 85 E4 68 38
1060- E9 1F 20 58 FF 50 19 18
1068- A5 E0 69 06 85 E0 D0 02
1070- E6 E1 68 A8 C8 C4 08 D0
1078- D1 68 85 E7 68 85 E4 60
1080- BA CA CA 9A 0A A8 B1 CE
1088- 18 65 CE 48 C8 B1 CE 65
1090- CF 48 A6 E0 A4 E1 A5 E2
1098- 20 11 F4 68 A8 68 AA A9
10A0- 00 20 01 F6 2C 58 FF 60
10A8- 3F 00 80 00 84 00 8A 00
10B0- 90 00 9D 00 AA 00 B5 00
10B8- C1 00 C5 00 CD 00 D5 00
10C0- E2 00 EA 00 EE 00 F3 00
10C8- F6 00 FD 00 0B 01 13 01
10D0- 1D 01 28 01 32 01 3E 01
10D8- 48 01 51 01 5C 01 67 01
10E0- 6B 01 70 01 79 01 81 01
10E8- 8A 01 94 01 A1 01 AD 01
10F0- B9 01 C4 01 D0 01 DD 01
10F8- E7 01 F2 01 FE 01 07 02
1100- 0F 02 1C 02 25 02 31 02
1108- 3D 02 49 02 53 02 5F 02
1110- 6C 02 77 02 80 02 8B 02
1118- 96 02 A2 02 AE 02 B7 02
1120- C2 02 CA 02 D1 02 D9 02
1128- 5B 49 02 00 52 22 20 24
1130- 2C 00 08 24 1F 36 06 00
1138- 3A 67 3C 0C 6C BE 2D 1E
1140- 2E 1E FE 2C 00 E7 0C 25
1148- 15 F5 AB 15 1F 15 3F 77
1150- 29 00 0C 0C DC 3B 2E 96
1158- 17 4D 2E 24 00 60 1C BF
1160- AE 17 76 65 1C 0D 16 07
1168- 00 08 24 05 00 92 1C 1C
1170- 24 0C 0C 06 00 92 0C 0C
1178- 24 1C 1C 06 00 3C 1C 4C
1180- 6E 1E 16 3F 17 0D 0D DE
1188- 07 00 20 8D 3A 3F 77 31
1190- 05 00 89 F6 04 00 3F 4C
1198- 11 35 00 12 05 00 0C 0C
11A0- D6 DA 1E 06 00 0C 25 1C
11A8- 3F 17 36 2E 1E 0E 2D 0C
11B0- 24 07 00 24 BC 96 31 17
11B8- 2D 04 00 65 E4 3F 17 95
11C0- BA 2E 2D 25 00 25 0C 3C
11C8- 3F B7 92 15 2D 0C 24 00
11D0- 3A 27 0C 0C 0C 36 36 F5
11D8- 3E 00 38 27 2C 2D F5 AA
11E0- 36 1E 3F 1C 04 00 AD F6
11E8- 3F 1C 24 25 0C 0C 35 00
11F0- 0C 0C 3C 3F 77 92 36 05
11F8- 00 E7 64 2D 15 BE 15 F6
1200- 3F 1C 2C 00 E7 64 2D 15
1208- 36 77 1E 17 3F 04 00 08
1210- 16 06 00 08 16 BE 05 00
1218- 91 E2 1C 1C 0C 0C 0C 06
1220- 00 38 67 89 B5 3F 3F 04
1228- 00 93 62 0C 0C 1C 1C 1C
1230- 06 00 0C 0C 1C 3F 17 95
1238- 0A 16 05 00 30 2E 2C 24
1240- 1C 3F 17 36 36 0E 2D 25
1248- 00 3A 37 6E 09 24 67 E4
1250- 1C 1E 1E 2E 00 3F 24 2C
1258- 2D 15 BE 0E BE 3F 27 2C
1260- 00 89 F2 3F 1C 24 24 0C
1268- 2D 15 06 00 09 36 1E 3F
1270- 27 24 24 2C 2D 15 3E 00
1278- 39 B7 3A 24 24 24 2D 2D
1280- 96 92 3F 04 00 39 B7 1A
1288- 24 24 24 2D 2D 06 00 11
1290- 35 3E 3F 1C 24 24 0C 2D
1298- 35 00 2B 2D 24 FC 1B 36
12A0- 36 36 4D 21 24 00 52 3A
12A8- 67 24 24 3C 2D 06 00 9B
12B0- 72 2D 0C 24 24 3C 00 73
12B8- 0E 15 DF 23 24 24 6C 09
12C0- 1E 1E 06 00 89 12 3F 3F
12C8- 24 24 24 05 00 E0 1C 36
12D0- 36 36 4D 21 24 24 BC 06
12D8- 00 0E 56 24 24 24 DF 33
12E0- 2E 1E 36 2E 00 92 E7 24
12E8- 24 0C 2D 15 36 36 17 05
12F0- 00 65 3C 38 3F 36 2E 1E
12F8- 36 05 00 AA 15 1F 3F 20
1300- 24 64 2D 15 36 36 00 77
1308- 15 15 DF 23 24 24 2C 2D
1310- 15 BE 06 00 E7 64 2D 15
1318- 97 15 F6 3F 1C 04 00 24
1320- 1F 28 2D F5 92 33 2E 00
1328- 92 E7 24 24 6C 09 36 36
1330- BE 05 00 92 1C 1C 24 24
1338- 4D 31 36 BE 06 00 F6 1E
1340- 24 24 24 4D 31 36 BE 35
1348- 07 00 0C 0C FC 1B 76 16
1350- 17 6E 09 E4 04 00 1C 1C
1358- 6C 09 F6 D6 36 05 00 0C
1360- 0C 3C 3F 77 92 17 2E 2D
1368- 25 00 24 2C B5 D2 33 2E
1370- 2D 00 1C 1C 56 4A 0E 06
1378- 00 24 3C B7 52 31 3E 3F
1380- 00 25 3F 36 2D 25 24 3F
1388- 3F 36 36 2D 2D 25 24 24
1390- 3F 3F 3F 36 36 36 2D 2D
1398- 2D 05 00
```

# *Des disquettes sans DOS...*
# *Des disquettes qui se présentent...*
# *Des disquettes en 35 ou 36 pistes...*
# ...BOOTDATA

Dominique Ottello

Suite aux diverses interventions de lecteurs concernant l'utilisation de la 36ème piste (voir par exemple le numéro 23 de Pom's), voici un petit utilitaire qui permet d'automatiser le processus d'agrandissement d'une disquette DOS 3.3.

Il permet en particulier :

- de formater une disquette sur 35 pistes ;
- de formater une disquette sur 36 pistes ;
- de libérer les pistes 1 et 2, en effaçant le DOS (sans effacer les fichiers éventuellement présents) ;
- d'inscrire un message d'avertissement sur la piste 0, secteur 0, affiché lors du boot de cette disquette.

Son utilisation est très simple :

*BRUN BOOTDATA*

affiche une suite de valeurs de CALL, qu'il suffit de lancer pour exécuter l'une des quatre opérations ci-dessus.

Il fonctionne avec le DOS 3.3 (48Ko) standard, ainsi qu'avec le Diversi-DOS. Le source est en Big Mac et contient, aux lignes 565 à 569, le texte du message à afficher lors du boot. Ce texte peut être remplacé par celui de votre choix, dans la limite de 256 octets entre les étiquettes TOS0 et FIN.

## Comment faire ?

Après avoir fait "BRUN BOOTDATA" cette liste de CALLs apparaît :

| | Opération | Lecteur 1 | Lecteur 2 |
|---|---|---|---|
| 1 | Message lors du boot sur la disquette de données : | 768 | 771 |
| 2 | Libération des pistes 1 & 2 occupées habituellement par le DOS : | 774 | 777 |
| 3 | Initialisation 35 pistes et opérations 1 et 2 : | 780 | 783 |
| 4 | Idem 3 mais en 36 pistes : | 786 | 789 |

Ainsi, pour créer une disquette de DATAs dans le lecteur 2, sur 36 pistes, vous ferez CALL 789. Pour libérer les pistes 1 & 2 puis inscrire un message en cas de boot, vous ferez CALL774 puis CALL 768 si l'opération concerne le lecteur 1.

Attention : disquettes DOS 3.3 seulement...

## *Source 'BOOTDATA.S' Assembleur Merlin*

```
         ORG  $3000      ;/
*
CTOSOD1  EQU  $300       ;CALL en 768
CTOSOD2  EQU  $303       ;CALL en 771
CP1P2D1  EQU  $306       ;CALL en 774
CP1P2D2  EQU  $309       ;CALL en 777
CIN35D1  EQU  $30C       ;CALL en 780
CIN35D2  EQU  $30F       ;CALL en 783
CIN36D1  EQU  $312       ;CALL en 786
CIN36D2  Equ  $315       ;CALL en 789
*
RWTS     EQU  $3D9       ;Routine DOS écriture secteur
COUT     EQU  $FDED      ;Routine sortie caractère
HOME     EQU  $FC58      ;Routine effacement écran
*
CATALOG  EQU  $AE9E      ;Routine DOS contruction de
*                         la piste $11 (catalog et VTOC)
MOTORON  EQU  $C089      ;Mise en route moteur
MOTOROFF EQU  $C088      ;Arrêt moteur
DISK01   EQU  $C08A      ;Lecteur 1
DISK02   EQU  $C08B      ;Lecteur 2
Q6H      EQU  $C08D      ;Utilisé pour recherche
Q7L      EQU  $C08E      ;disquette protégée
*
*
REG1     EQU  $19        ;Sauvegarde
REG2     EQU  $1A        ;.........
FLAGERR  EQU  $1C        ;Drapeau erreur 00 pas erreur
*
*
********************************
* IMPLANTATION DES CALL        *
********************************
*
*
         LDA  #$4C
         STA  CTOSOD1    ;Implantation CALL en 768
         STA  CTOSOD2    ;............ .... .. 771
         STA  CP1P2D1    ;............ .... .. 774
         STA  CP1P2D2    ;............ .... .. 777
         STA  CIN35D1    ;............ .... .. 780
```

```
        STA  CIN35D2    ;............ .... .. 783
        STA  CIN36D1    ;............ .... .. 786
        STA  CIN36D2    ;............ .... .. 789
        LDA  #IT0S0D1
        STA  CT0S0D1+1
        LDA  #>IT0S0D1
        STA  CT0S0D1+2
        LDA  #IT0S0D2
        STA  CT0S0D2+1
        LDA  #>IT0S0D2
        STA  CT0S0D2+2
        LDA  #P1P2D1
        STA  CP1P2D1+1
        LDA  #>P1P2D1
        STA  CP1P2D1+2
        LDA  #P1P2D2
        STA  CP1P2D2+1
        LDA  #>P1P2D2
        STA  CP1P2D2+2
        LDA  #IN35D1
        STA  CIN35D1+1
        LDA  #>IN35D1
        STA  CIN35D1+2
        LDA  #IN35D2
        STA  CIN35D2+1
        LDA  #>IN35D2
        STA  CIN35D2+2
        LDA  #IN36D1
        STA  CIN36D1+1
        LDA  #>IN36D1
        STA  CIN36D1+2
        LDA  #IN36D2
        STA  CIN36D2+1
        LDA  #>IN36D2
        STA  CIN36D2+2
*
*
*********************************
* MESSAGES DES CALL             *
*********************************
*
        JSR  $FE84      ;Mode NORNAL
        JSR  $FB2F      ;Mode TEXT
        JSR  $FE89      ;Clavier en entrée
        JSR  $FE93      ;Vidéo en sortie
        JSR  HOME
*
        LDA  #LISTCALL
        STA  REG1
        LDA  #>LISTCALL
        STA  REG2
        LDY  #$00
BCLCALL LDA  (REG1),Y
        BEQ  IREG
        CMP  #$FF
        BEQ  RETCALL
        JSR  COUT
IREG    INC  REG1
        BNE  CALL1
        INC  REG2
CALL1   BNE  BCLCALL
RETCALL LDA  #$10
        STA  $22        ;Fenetre texte haute
        RTS
*
LISTCALL ASC  "             "
        INV  "LISTE DES CALL"8D8D
P0      ASC  "Ecriture message piste 0 secteur 0"8D00
        ASC  "Lecteur 1:768 * Lecteur 2:771"8D8D
P12     ASC  "Libération piste 1 et 2"8D00
        ASC  "Lecteur 1:774 * Lecteur 2:777"8D8D
I35     ASC  "Initialisation 35 pistes"8D00
        ASC  "Lecteur 1:780 * Lecteur 2:783"8D8D
I36     ASC  "Initialisation 36 pistes"8D00
        ASC  "Lecteur 1:786 * Lecteur 2:789"8D8D
        ASC  "+----------- "
        INV  "TAPER CALL XXX"
        ASC  " -----------+"8D8D8D8DFF
VTO     ASC  "Construction CATALOG et VTOC"8D00
P36     ASC  "Libération piste 36"8D00
CAT     ASC  "Mise à jour VTOC"8D00
*
*
*********************************
* ECRITURE PISTE 0 SECTEUR 0    *
*********************************
*
*
IT0S0D1 LDA  #$01       ;Lecteur 1
        STA  LECTEUR
        BNE  IT0S0A
IT0S0D2 LDA  #$02       ;Lecteur 2
        STA  LECTEUR
IT0S0A  JSR  PROTEC
IT0S0   LDA  #$00
        STA  FLAGERR    ;Mise à zéro ERREUR
        LDA  #P0
        STA  REG1
        LDA  #>P0
        STA  REG2
        JSR  MESSAGE
        LDY  #$00       ;Chargement Tampon IOB pour BOOT
BCL0    LDA  T0S0,Y
        STA  TAMPON,Y
        INY
        CMP  #$FF
        BNE  BCL0
        LDA  #$00       ;Numéro volume
        STA  VOLUME
        LDA  #$00       ;Numéro piste
        STA  PISTE
        LDA  #$00       ;Numéro secteur
        STA  SECTEUR
        LDA  #$02       ;Commande écriture
        STA  COMMANDE
        JSR  RWDISK
        RTS
*
*
*********************************
* LIBERATION PISTES 1 ET 2      *
*********************************
*
*
P1P2D1  LDA  #$01
        STA  LECTEUR
        BNE  P1P2A
P1P2D2  LDA  #$02
        STA  LECTEUR
P1P2A   JSR  PROTEC
P1P2    LDA  #$00
        STA  VOLUME
        LDA  #$11       ;Piste $11 (dec:17) Catalogue
        STA  PISTE
        LDA  #$00       ;Secteur 0 VTOC
        STA  SECTEUR
        LDA  #$01       ;Commande lecture
        STA  COMMANDE
        LDA  #P12
        STA  REG1
        LDA  #>P12
        STA  REG2
        JSR  MESSAGE
        JSR  RWDISK
        LDA  #$FF       ;Correspond à 8 secteurs libres
        STA  TAMPON+$3C ;Piste 1 secteurs F à 8
        STA  TAMPON+$3D ;Piste 1 ........ 7 à 0
        STA  TAMPON+$40 ;Piste 2 ........ F à 8
        STA  TAMPON+$41 ;Piste 2 ........ 7 à 0
        LDA  #$02       ;Commande écriture
        STA  COMMANDE
        JSR  RWDISK
```

```
        RTS
*
*
*********************************
* INITIALISATION NORMALE (35)   *
*********************************
*
*
IN35D1  LDA  #$01
        STA  LECTEUR
        BNE  IN35A
IN35D2  LDA  #$02
        STA  LECTEUR
IN35A   JSR  PROTEC
        LDA  #I35
        STA  REG1
        LDA  #>I35
        STA  REG2
        JSR  MESSAGE
IN35    LDA  #$04       ;Commande FORMAT
        STA  COMMANDE
        LDA  #$00
        STA  VOLUME
        JSR  RWDISK
        LDA  #$60       ;Implantation d'un code RTS
*                        pour éviter l'écriture du DOS
*                        sur la disquette lors de l'INIT
        STA  $AEFF
        LDA  LECTEUR    ;Implantation numéro lecteur
*                        dans la routine INIT du DOS
        STA  $B5F8
        LDA  #VTO
        STA  REG1
        LDA  #>VTO
        STA  REG2
        JSR  MESSAGE
        JSR  CATALOG
        LDA  #$20       ;Restauration modif DOS
        STA  $AEFF
        JSR  IT0S0      ;Message piste 0 secteur 0
        LDA  FLAGERR
        CMP  #$FF
        BNE  LP1
        RTS
LP1     JSR  P1P2       ;Libération pistes 1 et 2
        RTS
*
*
*********************************
* INITIALISATION 36 PISTES      *
*********************************
*
*
IN36D1  LDA  #$01
        STA  LECTEUR
        BNE  IN36
IN36D2  LDA  #$02
        STA  LECTEUR
IN36    JSR  PROTEC
        LDA  #I36
        STA  REG1
        LDA  #>I36
        STA  REG2
        JSR  MESSAGE
        LDA  #$24
        STA  $BEFE      ;Autorise le DOS pour 36 pistes
        JSR  IN35       ;Initialisation
        LDA  #$23
        STA  $BEFE      ;Remet DOS à valeur initiale
        LDA  FLAGERR
        CMP  #$FF
        BNE  PIS23
        RTS
PIS23   LDA  #P36
        STA  REG1
        LDA  #>P36
        STA  REG2
        JSR  MESSAGE
        LDA  #CAT
        STA  REG1
        LDA  #>CAT
        STA  REG2
        JSR  MESSAGE
        LDA  #$11       ;Piste 17 Catalogue
        STA  PISTE
        LDA  #$00       ;Secteur 0 VTOC
        STA  SECTEUR
        LDA  #$01       ;Ordre de lecture
        STA  COMMANDE
        JSR  RWDISK
        LDA  #$FF       ;8 secteurs libres
        STA  TAMPON+$C4 ;Piste 36 secteurs F à 8
        STA  TAMPON+$C5 ;..... .. ........ 7 à 0
        LDA  #$24
        STA  TAMPON+$34 ;Indique $24 piste en VTOC
        LDA  #$02       ;Ordre écriture
        STA  COMMANDE
        JSR  RWDISK
        RTS
*
*
*********************************
*  LECTURE OU ECRITURE DES      *
*  SECTEURS DU DISQUE           *
*********************************
*
*
RWDISK  LDA  #$00       ;Mise à zéro des erreurs
        STA  ERREUR
        LDA  #$FF       ;Mise à FF du drapeau erreur
        STA  FLAGERR
        LDA  #>IOB      ;Partie haute adresse IOB
        LDY  #<IOB      ;Partie basse .....
        JSR  RWTS
        LDA  #$00       ;Remise à zéro location $48
        STA  $48        ;servant au moniteur
        LDY  #$00
*
*
*********************************
* CONTROLE DES ERREURS          *
*********************************
*
*
        LDA  ERREUR     ;Gestion des erreurs
        CMP  #$08       ;Erreur format ?
        BNE  PROTECT
        LDA  #FORM      ;Chargement adresse message
        STA  REG1
        LDA  #>FORM
        STA  REG2
        BNE  MESSAGE
PROTECT CMP  #$10       ;Erreur protection ?
        BNE  VOLU
        LDA  #PROT
        STA  REG1
        LDA  #>PROT
        STA  REG2
        BNE  MESSAGE
VOLU    CMP  #$20       ;Erreur volume ?
        BNE  DRIVE
        LDA  #VOL
        STA  REG1
        LDA  #>VOL
        STA  REG2
        BNE  MESSAGE
DRIVE   CMP  #$40       ;Erreur I/O ?
        BNE  LECTU
        LDA  #DRIV
        STA  REG1
        LDA  #>DRIV
        STA  REG2
```

```
        BNE  MESSAGE
LECTU   CMP  #$80       ;Erreur lecture ?
        BNE  OKMESS     ;Pas d'erreur
        LDA  #LECT
        STA  REG1
        LDA  #>LECT
        STA  REG2
        BNE  MESSAGE
OKMESS  LDA  #$00       ;Remise à zéro drapeau erreur
        STA  FLAGERR
        RTS
MESSAGE LDY  #$00
MESSA   LDA  (REG1),Y   ;Affichage du message
        BEQ  RETO
        INY
        JSR  COUT
        JMP  MESSA
RETO    RTS
*
*       Messages
*
FORM    INV  "ERREUR D'INITIALISATION"878D00
PROT    INV  "DISQUETTE PROTEGEE"878D00
VOL     INV  "ERREUR NUMERO DE VOLUME"878D00
DRIV    INV  "ERREUR LECTEUR (I/O ERROR)"878D00
LECT    INV  "ERREUR DE LECTURE"878D00
*
*
*********************************
* ROUTINE DE TEST DISQUETTE     *
* PROTEGEE OU NON ET DEMANDE    *
* POUR ENLEVER PROTECTION       *
*********************************
*
*
PROTEC  JSR  HOME
        LDX  #$60       ;Slot 6
        LDA  LECTEUR
        CMP  #$02
        BEQ  DRIVE2
        LDA  DISK01,X
        JMP  PRB
DRIVE2  LDA  DISK02,X
PRB     LDA  MOTORON,X  ;Mise en route moteur
        LDY  #$FF       ;Délai
PRB0    LDX  #$FF
PRB1    DEX
        BNE  PRB1
        DEY
        BNE  PRB0
        LDX  #$60       ;Slot 6
        LDA  Q6H,X      ;Recherche protection
        LDA  Q7L,X
        BMI  PROTIN
        LDA  MOTOROFF,X ;Arret moteur
        RTS
PROTIN  LDA  MOTOROFF,X ;Arret moteur
        LDY  #$00
BCLPR   LDA  PROT,Y
        CMP  #$00
        BEQ  RETPROT
        JSR  COUT
        INY
        BNE  BCLPR
RETPROT LDY  #$00
BCLPR1  LDA  ENLEVER,Y
        CMP  #$00
        BEQ  RETENL
        JSR  COUT
        INY
        BNE  BCLPR1
RETENL  JSR  $FD0C
        JMP  PROTEC
*
ENLEVER ASC  "ENLEVER PROTECTION PUIS <RETURN>"878D00
*
*
*
*********************************
* TOUT CE QUI EST ENTRE 'T0S0' *
* ET 'FIN' SERA ECRIT SUR LA    *
* PISTE 0 SECTEUR 0             *
*********************************
*
*
T0S0    HEX  01         ;Pour accepter ce qui suit
        STA  MOTOROFF,X ;Arret moteur lecteur
        LDA  $2B        ;Chargement numéro slot boot
        LSR
        LSR
        LSR
        LSR
        ORA  #$C0
        STA  $3F
        LDA  #$00
        STA  $3E
        CLD
        JSR  $FE84      ;Vidéo NORMAL
        JSR  $FB2F      ;Mode TEXT
        JSR  $FE89      ;Clavier en entrée
        JSR  $FE93      ;Vidéo en sortie
        JSR  HOME
        LDY  #$00
*
*                        Calcul d'adresse pour tenir
*                        compte de l'adresse $800 au
*                        BOOT de la disquette.
LAB1    LDA  TEXTE-T0S0+$800,Y
*
        CMP  #$00
        BEQ  LAB2
        JSR  COUT
        INY
        BNE  LAB1
LAB2    JSR  $FD0C      ;Attente touche clavier
        JMP  ($003E)
*
*
*********************************
* Message à afficher au BOOT    *
*********************************
*
*
TEXTE   ASC  "+--------------------------------+"8D
        ASC  "! DISQUETTE DE DONNEES SANS DOS !"8D
        ASC  "! METTRE LA DISQUETTE PROGRAMME !"8D
        ASC  "!      APPUYER SUR <RETURN>      !"8D
        ASC  "+--------------------------------+"8787878D
*
        HEX  00         ;Fin de message
*
FIN     HEX  FF         ;Fin de chargement total
*
*
*********************************
* PARAMETRES POUR RWTS (IOB)    *
* Input Output control Block    *
*********************************
*
*
IOB     HEX  0160       ;Table IOB
LECTEUR HEX  00         ;Numéro lecteur
VOLUME  HEX  00         ;Numéro volume
PISTE   HEX  00         ;Numéro piste
SECTEUR HEX  00         ;Numéro secteur
        DA   CARUNIT    ;Adresse caract. unité disque
        DA   TAMPON     ;Adresse mémoire tampon secteur
        HEX  0000
COMMANDE HEX 00         ;Code de commande
*                       ;$00 positionnement
*                       ;$01 lecture
*                       ;$02 écriture
```

```
*                       ;$04 formatage
ERREUR   HEX  00        ;Code erreur en retour
*                       ;$08 erreur format
*                       ;$10 protégé écriture
*                       ;$20 erreur volume
*                       ;$40 erreur lecteur
*                       ;$80 erreur lecture
         HEX  00        ;Volume précédent
         HEX  60        ;Interface précédente
         HEX  01        ;Lecteur précédent
*
*
*********************************
* CARACTERISTIQUES UNITE DE     *
* DISQUETTE (DCT)               *
```

```
* Device Characteristics Table *
*********************************
*
*
CARUNIT  HEX  0001EFD8   ;Caractéristiques unité disque
*
*
*********************************
* DEBUT DU TAMPON DE 256 OCTETS*
* correspondant au contenu d'un*
* secteur lu ou à ce qui sera  *
* écrit sur un secteur         *
*********************************
*
TAMPON   DS   256        ;Début du tampon (256 octets)
         END
```

# Récapitulation 'BOOTDATA'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE BOOTDATA,A$3000,L$6C2*

```
3000- A9 4C 8D 00 03 8D 03 03
3008- 8D 06 03 8D 09 03 8D 0C
3010- 03 8D 0F 03 8D 12 03 8D
3018- 15 03 A9 17 8D 01 03 A9
3020- 32 8D 02 03 A9 1E 8D 04
3028- 03 A9 32 8D 05 03 A9 5A
3030- 8D 07 03 A9 32 8D 08 03
3038- A9 61 8D 0A 03 A9 32 8D
3040- 0B 03 A9 A2 8D 0D 03 A9
3048- 32 8D 0E 03 A9 A9 8D 10
3050- 03 A9 32 8D 11 03 A9 F5
3058- 8D 13 03 A9 32 8D 14 03
3060- A9 FC 8D 16 03 A9 32 8D
3068- 17 03 20 84 FE 20 2F FB
3070- 20 89 FE 20 93 FE 20 58
3078- FC A9 9B 85 19 A9 30 85
3080- 1A A0 00 B1 19 F0 07 C9
3088- FF F0 0B 20 ED FD E6 19
3090- D0 02 E6 1A D0 ED A9 10
3098- 85 22 60 A0 A0 A0 A0 A0
30A0- A0 A0 A0 A0 A0 A0 A0 A0
30A8- 0C 09 13 14 05 20 04 05
30B0- 13 20 03 01 0C 0C 8D 8D
30B8- C5 E3 F2 E9 F4 F5 F2 E5
30C0- A0 ED E5 F3 F3 E1 E7 E5
30C8- A0 F0 E9 F3 F4 E5 A0 B0
30D0- A0 F3 E5 E3 F4 E5 F5 F2
30D8- A0 B0 8D 00 CC E5 E3 F4
30E0- E5 F5 F2 A0 B1 BA B7 B6
30E8- B8 A0 AA A0 CC E5 E3 F4
30F0- E5 F5 F2 A0 B2 BA B7 B7
30F8- B1 8D 8D CC E9 E2 FB F2
3100- E1 F4 E9 EF EE A0 F0 E9
3108- F3 F4 E5 A0 B1 A0 E5 F4
3110- A0 B2 8D 00 CC E5 E3 F4
3118- E5 F5 F2 A0 B1 BA B7 B7
3120- B4 A0 AA A0 CC E5 E3 F4
3128- E5 F5 F2 A0 B2 BA B7 B7
3130- B7 8D 8D C9 EE E9 F4 E9
3138- E1 EC E9 F3 E1 F4 E9 EF
3140- EE A0 B3 B5 A0 F0 E9 F3
3148- F4 E5 F3 8D 00 CC E5 E3
3150- F4 E5 F5 F2 A0 B1 BA B7
3158- B8 B0 A0 AA A0 CC E5 E3
3160- F4 E5 F5 F2 A0 B2 BA B7
3168- B8 B3 8D 8D C9 EE E9 F4
3170- E9 E1 EC E9 F3 E1 F4 E9
3178- EF EE A0 B3 B6 A0 F0 E9
3180- F3 F4 E5 F3 8D 00 CC E5
3188- E3 F4 E5 F5 F2 A0 B1 BA
3190- B7 B8 B6 A0 AA A0 CC E5
3198- E3 F4 E5 F5 F2 A0 B2 BA
31A0- B7 B8 B9 8D 8D AB AD AD
31A8- AD AD AD AD AD AD AD AD
31B0- AD A0 14 01 10 05 12 20
31B8- 03 01 0C 0C 20 18 18 18
31C0- A0 AD AD AD AD AD AD AD
31C8- AD AD AD AD AB 8D 8D 8D
31D0- 8D FF C3 EF EE F3 F4 F2
31D8- F5 E3 F4 E9 EF EE A0 C3
31E0- C1 D4 C1 CC CF C7 A0 E5
31E8- F4 A0 D6 D4 CF C3 8D 00
31F0- CC E9 E2 FB F2 E1 F4 E9
31F8- EF EE A0 F0 E9 F3 F4 E5
3200- A0 B3 B6 8D 00 CD E9 F3
3208- E5 A0 C0 A0 EA EF F5 F2
3210- A0 D6 D4 CF C3 8D 00 A9
3218- 01 8D AF 35 D0 05 A9 02
3220- 8D AF 35 20 4D 34 A9 00
3228- 85 1C A9 B8 85 19 A9 30
3230- 85 1A 20 C5 33 A0 00 B9
3238- C7 34 99 C2 35 C8 C9 FF
3240- D0 F5 A9 00 8D B0 35 A9
3248- 00 8D B1 35 A9 00 8D B2
3250- 35 A9 02 8D B9 35 20 61
3258- 33 60 A9 01 8D AF 35 D0
3260- 05 A9 02 8D AF 35 20 4D
3268- 34 A9 00 8D B0 35 A9 11
3270- 8D B1 35 A9 00 8D B2 35
3278- A9 01 8D B9 35 A9 FB 85
3280- 19 A9 30 85 1A 20 C5 33
3288- 20 61 33 A9 FF 8D FE 35
3290- 8D FF 35 8D 02 36 8D 03
3298- 36 A9 02 8D B9 35 20 61
32A0- 33 60 A9 01 8D AF 35 D0
32A8- 05 A9 02 8D AF 35 20 4D
32B0- 34 A9 33 85 19 A9 31 85
32B8- 1A 20 C5 33 A9 04 8D B9
32C0- 35 A9 00 8D B0 35 20 61
32C8- 33 A9 60 8D FF AE AD AF
32D0- 35 8D F8 B5 A9 D2 85 19
32D8- A9 31 85 1A 20 C5 33 20
32E0- 9E AE A9 20 8D FF AE 20
32E8- 26 32 A5 1C C9 FF D0 01
32F0- 60 20 69 32 60 A9 01 8D
32F8- AF 35 D0 05 A9 02 8D AF
3300- 35 20 4D 34 A9 6C 85 19
3308- A9 31 85 1A 20 C5 33 A9
3310- 24 8D FE BE 20 BC 32 A9
3318- 23 8D FE BE A5 1C C9 FF
3320- D0 01 60 A9 F0 85 19 A9
3328- 31 85 1A 20 C5 33 A9 05
3330- 85 19 A9 32 85 1A 20 C5
3338- 33 A9 11 8D B1 35 A9 00
3340- 8D B2 35 A9 01 8D B9 35
3348- 20 61 33 A9 FF 8D 86 36
3350- 8D 87 36 A9 24 8D F6 35
3358- A9 02 8D B9 35 20 61 33
3360- 60 A9 00 8D BA 35 A9 FF
3368- 85 1C A9 35 A0 AD 20 D9
3370- 03 A9 00 85 48 A0 00 AD
3378- BA 35 C9 08 D0 0A A9 D3
3380- 85 19 A9 33 85 1A D0 3D
3388- C9 10 D0 0A A9 ED 85 19
3390- A9 33 85 1A D0 2F C9 20
3398- D0 0A A9 02 85 19 A9 34
33A0- 85 1A D0 21 C9 40 D0 0A
33A8- A9 1C 85 19 A9 34 85 1A
33B0- D0 13 C9 80 D0 0A A9 39
33B8- 85 19 A9 34 85 1A D0 05
33C0- A9 00 85 1C 60 A0 00 B1
33C8- 19 F0 07 C8 20 ED FD 4C
33D0- C7 33 60 05 12 12 05 15
33D8- 12 20 04 27 09 0E 09 14
33E0- 09 01 0C 09 13 01 14 09
33E8- 0F 0E 87 8D 00 04 09 13
33F0- 11 15 05 14 14 05 20 10
33F8- 12 0F 14 05 07 05 05 87
3400- 8D 00 05 12 12 05 15 12
3408- 20 0E 15 0D 05 12 0F 20
3410- 04 05 20 16 0F 0C 15 0D
3418- 05 87 8D 00 05 12 12 05
3420- 15 12 20 0C 05 03 14 05
3428- 15 12 20 28 09 2F 0F 20
3430- 05 12 12 0F 12 29 87 8D
3438- 00 05 12 12 05 15 12 20
3440- 04 05 20 0C 05 03 14 15
3448- 12 05 87 8D 00 20 58 FC
3450- A2 60 AD AF 35 C9 02 F0
3458- 06 BD 8A C0 4C 62 34 BD
3460- 8B C0 BD 89 C0 A0 FF A2
3468- FF CA D0 FD 88 D0 F8 A2
3470- 60 BD 8D C0 BD 8E C0 30
```

```
3478- 04 BD 88 C0 60 BD 88 C0
3480- A0 00 B9 ED 33 C9 00 F0
3488- 06 20 ED FD C8 D0 F3 A0
3490- 00 B9 A4 34 C9 00 F0 06
3498- 20 ED FD C8 D0 F3 20 0C
34A0- FD 4C 4D 34 C5 CE CC C5
34A8- D6 C5 D2 A0 D0 D2 CF D4
34B0- C5 C3 D4 C9 CF CE A0 D0
34B8- D5 C9 D3 A0 BC D2 C5 D4
34C0- D5 D2 CE BE 87 8D 00 01
34C8- 9D 88 C0 A5 2B 4A 4A 4A
34D0- 4A 09 C0 85 3F A9 00 85
34D8- 3E D8 20 84 FE 20 2F FB
34E0- 20 89 FE 20 93 FE 20 58
34E8- FC A0 00 B9 37 08 C9 00
34F0- F0 06 20 ED FD C8 D0 F3
34F8- 20 0C FD 6C 3E 00 AB AD
3500- AD AD AD AD AD AD AD AD
3508- AD AD AD AD AD AD AD AD
3510- AD AD AD AD AD AD AD AD
3518- AD AD AD AD AD AD AB 8D
3520- A1 A0 C4 C9 D3 D1 D5 C5
3528- D4 D4 C5 A0 C4 C5 A0 C4
3530- CF CE CE C5 C5 D3 A0 D3
3538- C1 CE D3 A0 C4 CF D3 A0
3540- A1 8D A1 A0 CD C5 D4 D4
3548- D2 C5 A0 CC C1 A0 C4 C9
3550- D3 D1 D5 C5 D4 D4 C5 A0
3558- D0 D2 CF C7 D2 C1 CD CD
3560- C5 A0 A1 8D A1 A0 A0 A0
3568- A0 A0 C1 D0 D0 D5 D9 C5
3570- D2 A0 D3 D5 D2 A0 BC D2
3578- C5 D4 D5 D2 CE BE A0 A0
3580- A0 A0 A0 A0 A1 8D AB AD
3588- AD AD AD AD AD AD AD AD
3590- AD AD AD AD AD AD AD AD
3598- AD AD AD AD AD AD AD AD
35A0- AD AD AD AD AD AD AB 87
35A8- 87 87 8D 00 FF 01 60 00
35B0- 00 00 00 BE 35 C2 35 00
35B8- 00 00 00 00 60 01 00 01
35C0- EF D8 00 00 00 00 00 00
35C8- 00 00 00 00 00 00 00 00
35D0- 00 00 00 00 00 00 00 00
35D8- 00 00 00 00 00 00 00 00
35E0- 00 00 00 00 00 00 00 00
35E8- 00 00 00 00 00 00 00 00
35F0- 00 00 00 00 00 00 00 00
35F8- 00 00 00 00 00 00 00 00
3600- 00 00 00 00 00 00 00 00
3608- 00 00 00 00 00 00 00 00
3610- 00 00 00 00 00 00 00 00
3618- 00 00 00 00 00 00 00 00
3620- 00 00 00 00 00 00 00 00
3628- 00 00 00 00 00 00 00 00
3630- 00 00 00 00 00 00 00 00
3638- 00 00 00 00 00 00 00 00
3640- 00 00 00 00 00 00 00 00
3648- 00 00 00 00 00 00 00 00
3650- 00 00 00 00 00 00 00 00
3658- 00 00 00 00 00 00 00 00
3660- 00 00 00 00 00 00 00 00
3668- 00 00 00 00 00 00 00 00
3670- 00 00 00 00 00 00 00 00
3678- 00 00 00 00 00 00 00 00
3680- 00 00 00 00 00 00 00 00
3688- 00 00 00 00 00 00 00 00
3690- 00 00 00 00 00 00 00 00
3698- 00 00 00 00 00 00 00 00
36A0- 00 00 00 00 00 00 00 00
36A8- 00 00 00 00 00 00 00 00
36B0- 00 00 00 00 00 00 00 00
36B8- 00 00 00 00 00 00 00 00
36C0- 00 00
```

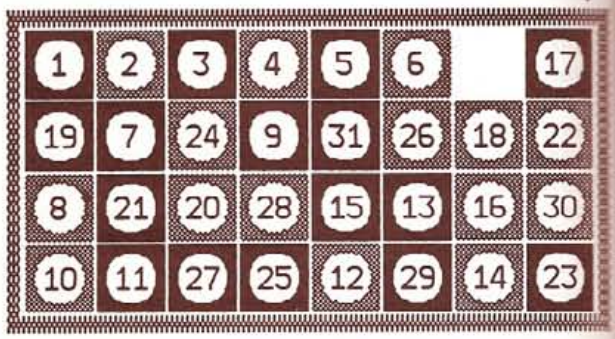
# Une nouvelle disquette de jeux : Ludologic

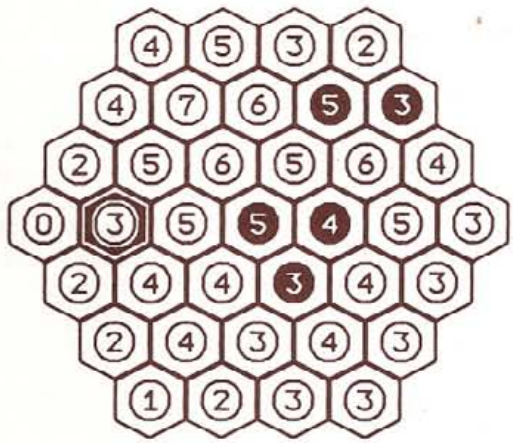
**Au sommaire de la disquette LUDOLOGIC, trois jeux de réflexion de difficulté croissante. Ces jeux qui nécessitent des neurones aussi calmes qu'entraînés, ne devraient pas décevoir les amateurs de puzzles et autres casse-têtes.**

**Il n'est pas nécessaire de présenter TAQUIN, ce pousse-pousse informatique ici fort bien présenté.**

**Nouvelle difficulté, NOIR & BLANC : 37 hexagones peuvent être noirs ou blancs mais au départ vous n'en connaissez pas la couleur. Chacun comporte un numéro qui représente le nombre de cellules voisines blanches... A vous de reconstituer le décor original !**

**HEXAGONE MAGIQUE est encore plus délicat, même principe que le carré magique, mais ici vous devrez installer les chiffres de 1 à 19 dans un hexagone de telle façon que les 5 horizontales et 10 obliques totalisent chacune 38 : bonne chance.**




Fidèle à son habitude, Pom's vous propose sur cette disquette les sources des routines écrites par Sylvie Gallet en assembleur Lisa 2.5. Bien entendu, le Basic est également listable. TAQUIN et NOIR & BLANC utilisent leur propre routine graphique qui permet de dessiner plus rapidement qu'avec des shapes.

**80,00 F Franco,**
**Bon de commande page 74**

# Commandes ProDOS : Renommez-les !

Bruno Fénart

Qui n'a jamais eu de problèmes en tapant LAOD au lieu de LOAD, ou CQTQLOG au lieu de CATALOG. N'avez vous jamais eu envie de raccourcir les commandes de ProDOS ou de les traduire, comme cela a déjà été proposé pour le DOS 3.3. Et si, pour le même prix, vous vous offriez deux commandes supplémentaires qui ne tarderont pas à vous paraître indispensables ?

## Des commandes personnalisées

Les versions de BASIC. SYSTEM sont innombrables et il serait imprudent de les 'patcher' directement comme en DOS 3.3. C'est pourquoi, plutôt que remplacer les commandes normales, on ajoute des commandes qui auront le même rôle. Cela aura pour avantage d'ajouter des commandes... sans en supprimer.

Après avoir lancé l'exécution de NEWCOM, vous pourrez taper "CATALOG" aussi bien que "CG" pour obtenir le catalogue d'une disquette.

*Les nouvelles commandes*

| | | |
|---|---|---|
| *BL* | = | *BLOAD* |
| *BR* | = | *BRUN* |
| *BS* | = | *BSAVE* |
| *CG* | = | *CATALOG* |
| *CT* | = | *CAT* |
| *CH* | = | *CHAIN* |
| *CL* | = | *CLOSE* |
| *CR* | = | *CREATE* |
| *DL* | = | *DELETE* |
| *EX* | = | *EXEC* |
| *I* | = | *IN£* |
| *LD* | = | *LOAD* |
| *LK* | = | *LOCK* |
| *PX* | = | *PREFIX* |
| *P* | = | *PR£* |
| *RN* | = | *RENAME* |
| *SA* | = | *SAVE* |
| *UN* | = | *UNLOCK* |
| *VE* | = | *VERIFY* |

La syntaxe des commandes raccourcies est la même que celles des commandes *normales*. Comme l'intérêt de renommer les commandes utilisables seulement dans un programme est faible, seules les commandes accessibles en mode direct ont été traduites. Cependant, vous pouvez changer cela si vous le désirez, grâce au programme CHANGE.

*Note : NEWCOM est compatible avec EPE 5.0, mais EPE devra être lancé avant car il n'est pas relogeable.*

## Choisir le nom de ses commandes

Vous pouvez choisir vous-mêmes les nouveaux noms des commandes ProDOS en utilisant le programme CHANGE. Ces noms ne doivent pas être constitués de plus de huit caractères. De plus, le premier caractère d'une commande doit obligatoirement être une lettre. Les caractères suivants peuvent être n'importe quoi (les minuscules sont automatiquement converties en majuscules) sauf un espace ou un caractère de contrôle. Il faut faire attention à ce choix : en effet, si vous prenez un nom qui ressemble à une commande ProDOS (respectivement commande Basic), votre propre commande sera interceptée (respectivement interceptera) par la commande en question. Ainsi, si vous choisissez de renommer "CATALOG" par "CATA", votre commande sera interceptée et reconnue comme "CAT" du sous volume A. Réciproquement, si vous choisissez "L" pour renommer "LOAD", votre commande interceptera la commande du Basic LIST et traduira par exemple "LIST,999" par "LOAD IST,999" (en mode direct uniquement et il suffirait de mettre ":" devant "LIST" pour éviter ce problème).

Pour ceux qui n'auraient pas suivi, pas de panique ! En utilisant directement NEWCOM sans changer les noms prédéfinis vous n'aurez pas les problèmes cités ci-dessus.

## Deux commandes indispensables :

## BYE et ONLINE

BYE permet d'enchaîner vers un autre interpréteur ou de relancer MOUSE.DESK si vous l'avez utilisé pour démarrer. Cette commande est d'ailleurs déjà disponible sur certaines versions de BASIC.SYSTEM (version 1.1) . Mais NEWCOM vous permet d'en disposer quelle que soit la version que vous utilisez. Quant à la commande ONLINE, elle existe sur la plupart des logiciels tournant sous ProDOS, mais malheureusement pas avec BASIC.SYSTEM, faute de place sans doute. Là encore NEWCOM pallie ce défaut et cette commande devient vite indispensable, surtout avec l'arrivée des unidisks 3'5.

Remarque : la commande ProDOS qui permet de quitter une application est un peu sommaire (sauf si vous utilisez Mouse.Desk). Pour l'utiliser sans problème, il est conseillé pour chacune de vos applications sous ProDOS de noter le préfixe de la disquette et le nom du programme (qui doit être de type System). Par exemple, pour AppleWriter, le préfixe est /AW et l'application est /AW.SYSTEM, ce qui est simple à retenir. Pour Épistole, le préfixe est /LB (?!) et l'application /MY.SYSTEM (?!).

## Utilisation de ProDOS en assembleur

L'utilisation de ProDOS en assembleur est assez délicate. En DOS 3.3 il suffisait d'envoyer par COUT la commande précédé de CTRL-D (exactement comme on l'aurait fait depuis le Basic).

Avec ProDOS il existe deux solutions :

• La première en appelant directement le MLI. Cette technique a été expliquée par Alexandre Avrane dans les colonnes du numéro 20 de Pom's. Elle est délicate et nécessite plusieurs opérations (GET-PREFIX ou ON-LINE, GET-FILE-INFO, OPEN, READ, CLOSE). Elle n'est indispensable que si vous souhaitez ne pas utiliser BASIC.SYSTEM (pour écrire votre propre fichier système, par exemple) ou pour des commandes non fournies par BASIC.SYSTEM (comme, par exemple, ONLINE et BYE).

• La seconde solution ressemble beaucoup à celle utilisée en DOS 3.3 . Il faut recopier en $200 la commande de BASIC.SYSTEM que l'on veut exécuter, suivie par un retour chariot (mais sans CTRL-D), puis appeler DOSCMD = $BE03. Puis on doit tester s'il y a eu une erreur et l'afficher à l'aide du sous programme PRINTERR= $BE0C (le code de l'erreur se trouve dans le registre A et non en ERRCODE = $BE0F comme l'indique le manuel de référence).

### Exemple simple

```
        LDX  #00
BCL     LDA  CMD,X      ;Recopie
                        la cde
        STA  $200,X
        INX
        CMP  #$8D       ;Dernier
                        caractère?
        BNE  BCL
        JSR  $BE03      ;Exécute
                        la cde
        BCC  NOERR      ;Erreur si
                        Carry set
        JSR  $BE0C      ;Si oui af
                        fiche err.
                        dont le n°
                        est dans A
NOERR   RTS
CMD     ASC  "CATALOG"  ;Commande
        DFB  $8D
```

C'est cette solution que nous utiliserons pour simuler les commandes de BASIC.SYSTEM en traduisant le nom des commandes et en décalant de 8 caractères (la commande la plus longue est POSITION) la liste des paramètres. Seuls la gestion et l'affichage de l'erreur sont un peu différents, car pris en charge directement quand on revient de la commande extérieure vers le Basic. Notez que la table des commandes est concaténée de la même façon que celle de BASIC.SYSTEM.

Enfin, pour terminer, signalons que le relogeur est le même que celui utilisé pour DATHEUR (Pom's n°24) en plus performant. Il tient compte de la structure spéciale des appels au MLI et permet de reloger les adresses des paramètres ainsi que celles des buffers ou des noms d'accès, suivant la commande ProDOS.

### *Comment faire ?*

*En premier lieu, saisir et sauvegarder le programme binaire NEWCOM. C'est ce programme qui ajoute les nouvelles commandes. Saisir puis sauvegarder CHANGE.*

*Vous voulez simplement utiliser les nouvelles commandes telles qu'elles sont notées dans l'encadré, faites :*

*-NEWCOM*

*Vous préférez modifier le nom des nouvelles commandes avant de les installer, faites :*

*-CHANGE*

*Vous apportez alors les modifications que vous souhaitez. Après validation, VOTRE nouvelle version de NEWCOM est sauvegardée sur la disquette. Il vous est alors proposé de lancer NEWCOM pour installer vos nouvelles commandes.*

## *Récapitulation 'NEWCOM'*

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE NEWCOM,A$2000,L$391*

```
2000- AD 08 BE 8D 40 21 AD 07
2008- BE 8D 3F 21 A9 03 20 F5
2010- BE C9 0C D0 03 4C 09 BE
2018- 8D FF 20 85 43 8D 08 BE
2020- A9 21 85 3D A9 FF 85 3E
2028- A9 23 85 3F A0 00 84 42
2030- B4 3C 8C 07 BE 20 2C FE
2038- 20 3C 20 60 AD FF 20 85
2040- 3B A9 00 85 3A A2 00 A1
2048- 3A F0 7F 20 8C F8 A4 2F
2050- C0 02 D0 5B B1 3A C9 BF
2058- D0 12 88 B1 3A C9 00 D0
```

# Programme 'CHANGE'

```
10 D$ = CHR$ (4): PRINT D$"BLOAD NEWCOM"
20 DIM PR$(30),NW$(30)
30 DATA "APPEND","BLOAD","BRUN","BSAVE","
   CATALOG","CAT","CHAIN","CLOSE","CREA
   TE","DELETE","EXEC","FLUSH","FRE","I
   N#","LOAD","LOCK"
40 DATA "NOMON","OPEN","POSITON","PREFIX"
   ,"PR#","READ","RENAME","RESTORE","RU
   N","SAVE","STORE","UNLOCK","VERIFY",
   "WRITE","-"
50 FOR I = 0 TO 30: READ K$:PR$(I) = K$:
   NEXT I
60 PRINT CHR$ (21): HOME
100 REM Lecture
110 FOR I = 8960 TO 9216: IF PEEK (I) =
   194 AND PEEK (I + 1) = 217 AND PEE
   K (I + 2) = 197 AND PEEK (I + 3) =
   141 THEN TB = I + 4:I = 9216
120 NEXT I:LT = 9216 - TB - 2
130 J = TB: FOR I = 0 TO 30:K$ = ""
140 J = J + 1:K = PEEK (J): IF K < > 0 T
   HEN K$ = K$ + CHR$ (K): GOTO 140
150 NW$(I) = K$:LT = LT - LEN (K$) - 1: N
   EXT I
200 REM Affichage
210 VTAB 2: PRINT "  Renommez les comma
   ndes de ProDOS": PRINT "_____
   _____";: PRIN
   T "_____";: HTAB 21: PRINT "_____
   "
220 VTAB 21: PRINT "_____
   _____": PRINT "Place d
   isponible: 000": PRINT "Escape pour
   quitter.";
230 FOR II = 0 TO 30: GOSUB 1000: INVERSE
   : PRINT PR$(II); TAB( H + 8);: NORM
   AL : PRINT " = ";NW$(II): NEXT II
240 II = 0
250 VTAB 23: HTAB 19: PRINT SPC( (LT < 1
   00) + (LT < 10));LT: GOSUB 1000
260 N$ = NW$(II):LT = LT + LEN (N$):LM =
   8 * (LT > 7) + LT * (LT < 8): GOSUB
   3000:NW$(II) = N$:LT = LT - LEN (N$
   )
270 IF K = 13 OR K = 10 THEN II = II + 1
   - 31 * (II = 30)
280 IF K = 11 THEN II = II - 1 + 31 * (II
   = 0)
290 IF K < > 27 GOTO 250
300 REM Quitter
310 J = TB: FOR II = 0 TO 30:N$ = NW$(II):
   L = LEN (N$): IF L = 0 GOTO 330
320 FOR I = 1 TO L:J = J + 1: POKE J, ASC
   ( MID$ (N$,I,1)): NEXT I
330 J = J + 1: POKE J,0
340 NEXT II
400 M$ = "Enregistrer les nouvelles comman
   des": GOSUB 2000: IF K$ = "N" GOTO 5
   00
450 PRINT CHR$ (4)"BSAVE NEWCOM"
500 M$ = "Lancer l'exécution de NEWCOM": G
   OSUB 2000: IF K$ = "N" GOTO 600
550 CALL 8192: HOME : PRINT "ONLINE": PRI
   NT CHR$ (4)"ONLINE"
600 END
1000 C = INT (II / 16):V = II + 5 - C * 1
   6:H = 1 + C * 20: VTAB V: HTAB H: RE
   TURN
2000 VTAB 23: HTAB 1: CALL - 958: PRINT
   M$;" ? (O/N) ";
2010 GET K$:K$ = CHR$ ( ASC (K$) - 32 *
   (K$ > "Z")): IF K$ < > "O" AND K$ <
   > "N" GOTO 2010
2020 RETURN
3000 REM Introduction d'un nom
3010 L = LEN (N$): IF L > LM THEN L = LM:
   N$ = LEFT$ (N$,LM)
3020 HTAB H + 11:I = L: PRINT N$;
3030 HTAB H + 11 + I: GET K$:K = ASC (K$
   )
3040 IF K = 27 GOTO 3150
3050 IF K = 10 OR K = 11 OR K = 13 GOTO 3
   140
3060 IF K = 8 AND I > 0 THEN I = I - 1: G
   OTO 3030
3070 IF K = 21 AND I < L THEN I = I + 1:
   GOTO 3030
3080 IF I = LM THEN PRINT CHR$ (7);: GO
   TO 3030
3090 IF 96 < K AND K < 123 THEN K = K - 3
   2:K$ = CHR$ (K)
3100 IF K = 127 OR K < = 32 OR I = 0 AND
   (K < 65 OR 90 < K) THEN PRINT CHR
   $ (7);: GOTO 3030
3110 PRINT K$;:I = I + 1: IF I > L THEN L
   = I
3120 N$ = MID$ (N$,1,I - 1) + K$ + MID$
   (N$,I + 1,L - I)
3130 GOTO 3030
3140 PRINT SPC( 8 - I);:N$ = MID$ (N$,1
   ,I)
3150 RETURN
```

```
2060- 0B 88 B1 3A C9 20 D0 04
2068- A0 05 84 2F A5 3A 85 FA
2070- A5 3B 85 FB A4 2F 20 D9
2078- 20 C0 05 D0 32 B1 3A 85
2080- FB 88 B1 3A 85 FA 88 B1
2088- 3A C9 40 F0 17 C9 80 F0
2090- 13 C9 81 F0 0F 38 E9 C0
2098- 90 15 C9 0C B0 11 AA BC
20A0- F3 20 F0 0B 20 D9 20 C0
20A8- 04 D0 04 A0 02 D0 F5 A5
20B0- 2F 38 65 3A 85 3A A5 3B
20B8- 69 00 85 3B 38 ED FF 20
20C0- A6 3A 20 CB 20 B0 03 4C
20C8- 45 20 60 C9 03 90 06 D0
20D0- 06 E0 00 B0 02 18 60 38
20D8- 60 88 B1 FA AA C8 B1 FA
20E0- 38 E9 21 90 0C 20 CB 20
20E8- B0 07 6D FF 20 91 FA 18
```

# Source 'T.NEWCOM'
## Assembleur ProCODE

*Assemblage par ProCODE*

```
*********************************
*                               *
*   NEWCOM  (C) Bruno Fénart  1986   *
*                               *
*********************************24/5/86

*  Permet de renommer les commandes
* de BASIC.SYSTEM
*
*  Ajoute en plus deux commandes :
* BYE - permet de quitter BASIC.SYSTEM
* ONLINE - permet de connaitre tous les volumes en lignes
* Syntaxe : Aucun paramètre pour ces deux commandes

START    EQU  $2000
DEBUT    EQU  START+$100
FIN      EQU  DEBUT+$300

*  Variables en page 0
*  -------------------
LENGTH   EQU  $2F        ;Longueur de l'instruction
PCL      EQU  $3A        ;Adresse de la ligne désassemblée
A1       EQU  $3C        ;Adresse de départ pour MOVE
A2       EQU  $3E        ;Adresse de fin      "    "
A4       EQU  $42        ;Adresse d'arrivée   "    "

ADR      EQU  $FA        ;Vecteur d'indirection temporaire

*  variables diverses
*  ------------------
BUFFIN   EQU  $200       ;Buffer utilisé temporairement

*  Sous programmes du moniteur
*  ---------------------------
INSDSP2  EQU  $F88C      ;Décode la ligne (PCL) avec X=0
COUT     EQU  $FDED      ;Sortie d'un caractère
MOVE     EQU  $FE2C      ;(A1)-(A2) vers (A4) avec Y=0

*  Page globale BASIC.SYSTEM  (BI)
*  -------------------------------

DOSCMD   EQU  $BE03      ;Exécution d'une commande du BI
EXTRNCMD EQU  $BE06      ;JMP vers interpréteur extérieur
ERROUT   EQU  $BE09      ;Affiche l'erreur et fin
BADCALL  EQU  $BE8B      ;Conversion code erreur MLI en BI
XRETURN  EQU  $BE9E      ;RTS connu en page globale du BI
GETBUFR  EQU  $BEF5      ;Réserve (A) page(s) sous le BI

XTRNADDR EQU  $BE50      ;Adresse d'une commande externe
XCNUM    EQU  $BE53      ;N° de la commande (0 si externe)
PBITS    EQU  $BE54      ;Paramètres optionnels autorisés
*FBITS EQU $BE56         ;Paramètres trouvés
VPATH1   EQU  $BE6C      ;Vecteur vers la commande entrée

* Page globale ProDOS & codes de fonction
* ---------------------------------------
MLI      EQU  $BF00      ;Entrée de ProDOS

DEVCNT   EQU  $BF31      ;Nombre de lecteurs
ONLINE   =    $C5        ;Code de ONLINE
QUIT     =    $65        ;Code de QUIT
ALLOC    =    $40        ;Code de ALLOC INTERRUPT
READBLOC =    $80        ;Code de READ BLOC
WRITEBLOC =   $81        ;Code de WRITE BLOC

         ORG  START

**********************
*                    *
*   INITIALISATION   *
*                    *
**********************

*  Installe la commande personnalisée
* entre BASIC.SYSTEM (BI) et ses buffers

         LDA  EXTRNCMD+2 ;Préserve la commande précédente
         STA  PRECMD+2
         LDA  EXTRNCMD+1
         STA  PRECMD+1

         LDA  #>FIN-DEBUT+$FF ;Réserve autant de pages qu'en
         JSR  GETBUFR    ; occupe le programme à reloger
         CMP  #$0C       ;Si pas d'erreur (A)= adr. buffer
         BNE  ]gothem    ;sinon (A) = code d'erreur BI
         JMP  ERROUT     ;Affiche NO BUFFERS AVAILABLE et fin

]gothem  STA  ADRESSE    ;Préserve l'adresse du relogement
         STA  A4+1       ;Adresse destination pour MOVE
         STA  EXTRNCMD+2 ;installée en commande extérieure

         LDA  #>DEBUT    ;Adresses du programme déplacé
         STA  A1+1
         LDA  #<FIN-1
         STA  A2
         LDA  #>FIN-1
         STA  A2+1

         LDY  #0         ;Poids faibles nuls
         STY  A4
         STY  A1
         STY  EXTRNCMD+1

         JSR  MOVE       ;Transfert du programme (Y = 0)
         JSR  RELOGE     ;Reloge le programme sous le BI
         RTS

* RELOGEur
* ========
*  Réassemble le programme situé à l'adresse = (ADRESSE)
* en fonction de son adresse d'assemblage = DEBUT
* Les deux adresses doivent être des multiples de 256
* Longueur du programme (données comprises) = FIN-LONGUEUR
* La zone des données est séparée par DFB $00 (= BRK)
* Le programme prend en compte la structure d'appel au MLI
*  et reloge, si nécessaire, les adresses des paramètres

RELOGE   LDA  ADRESSE    ;Adresse du relogement
         STA  PCL+1
         LDA  #0         ;Idem poids faible
         STA  PCL
```

```
20F0- 60 38 60 02 02 04 02 02
20F8- 03 02 02 04 00 03 03 00
2100- D8 AD 6C BE 85 FA AD 6D
2108- BE 85 FB A9 00 8D 92 22
2110- A2 FF A0 00 E8 C8 BD 41
2118- 23 29 7F D0 05 88 D0 21
2120- F0 10 D1 FA F0 EE 49 20
2128- D1 FA F0 E8 E8 BD 41 23
2130- D0 FA EE 92 22 AD 92 22
2138- CD 91 22 D0 D5 38 4C 9E
2140- BE 88 8C 93 22 AD 5F 21
2148- 8D 50 BE AD 60 21 8D 51
2150- BE A9 00 8D 54 BE 8D 55
2158- BE 8D 53 BE 18 60 4C 61
2160- 21 D8 AE 92 22 F0 43 CA
2168- F0 3A CA A0 00 B9 00 02
2170- C9 8D F0 03 C8 D0 F6 B9
2178- 00 02 99 08 02 88 CC 93
2180- 22 D0 F4 98 38 69 08 A8
2188- A9 A0 99 FF 01 88 D0 FA
2190- BC 22 23 BD 03 23 AA BD
2198- 93 22 99 FF 01 CA 88 D0
21A0- F6 4C 03 BE 20 00 BF 65
21A8- 58 22 20 00 BF C5 54 22
21B0- B0 78 AD 31 BF 8D 52 22
21B8- A9 D3 20 ED FD AD 52 22
21C0- 0A 0A 0A 0A A8 B9 00 02
21C8- 8D 53 22 29 7F 4A 4A 4A
21D0- 4A 09 B0 20 ED FD A9 AC
21D8- 20 ED FD A9 C4 20 ED FD
```

```
]décode  LDX  #0          ;Indispensable pour INSDSP2
         LDA  (PCL,X)     ;L'instruction est égale à BRK
         BEQ  ]fin        ;alors début des données
         JSR  INSDSP2     ;Décode une instruction et
         LDY  LENGTH      ; détermine sa longueur
         CPY  #2
         BNE  ]suivant    ;Longueur < 3 : Opérande inchangé

         LDA  (PCL),Y     ;L'instruction est-elle JSR MLI ?
         CMP  #>MLI
         BNE  ]nonmli
         DEY
         LDA  (PCL),Y
         CMP  #<MLI
         BNE  ]nonmli
         DEY
         LDA  (PCL),Y
         CMP  #$20
         BNE  ]nonmli
         LDY  #5          ;Instruction du MLI sur 6 octets
         STY  LENGTH

]nonmli  LDA  PCL         ;Passage de paramètre pour CALCUL
         STA  ADR
         LDA  PCL+1
         STA  ADR+1
         LDY  LENGTH      ;Instruction sur 3 ou 5 octets
         JSR  CALCUL      ;Calcul d'une adresse relogée
         CPY  #5          ;Est-ce JSR $BF00 ?
         BNE  ]suivant    ;Non alors instruction suivante
         LDA  (PCL),Y     ;Adresse des paramètres MLI
         STA  ADR+1       ;mise comme paramètre de CALCUL
         DEY
         LDA  (PCL),Y
         STA  ADR
         DEY              ;Offset des opérandes à changer
         LDA  (PCL),Y     ;suivant le code de fonction MLI
         CMP  #ALLOC      ;Offset = 3 pour ALLOC INTERRUPT
         BEQ  ]calcul
         CMP  #READBLOC   ; ainsi que pour READ BLOC
         BEQ  ]calcul
         CMP  #WRITEBLOC  ; ainsi que pour WRITE BLOC
         BEQ  ]calcul
         SEC
         SBC  #$C0
         BCC  ]suivant
         CMP  #$C
         BCS  ]suivant
         TAX
         LDY  OFFSET,X    ;Offset des codes de $C0 à $CB
         BEQ  ]suivant    ;Transformation sauf pour NEWLINE

]calcul  JSR  CALCUL
         CPY  #4          ;Cas de OPEN & RENAME (doublée)
         BNE  ]suivant    ;sinon une seule transformation
         LDY  #2          ;Offset du second paramètres
         BNE  ]calcul     ; = JMP

]suivant LDA  LENGTH      ;Longueur de l'instruction - 1
         SEC
         ADC  PCL         ;Calcul de l'adresse suivante
         STA  PCL
         LDA  PCL+1
         ADC  #0
         STA  PCL+1
         SEC
         SBC  ADRESSE     ;Longueur du code désassemblée
         LDX  PCL         ; mise en X, A
         JSR  TEST        ;Test si < longueur du programme
         BCS  ]fin
         JMP  ]décode     ;Instruction suivante
]fin     RTS

* TEST une longueur
* -----------------
*  Longueur en X, A (A poids fort) comparée à FIN-DEBUT
*  Si inférieur C = 0, sinon C = 1

TEST     CMP  #>FIN-DEBUT
         BCC  ]oui        ; ou opérande > adresse de fin
         BNE  ]non
         CPX  #<FIN-DEBUT;Test poids faible de l'opérande
         BCS  ]non
]oui     CLC              ;Compris entre les limites
         RTS
]non     SEC
         RTS

* CALCUL des opérandes à modifier
* -------------------------------
*  Poids fort de l'opérande à modifier en (ADR),Y
*  Retour C = 1 si aucune transformation, 0 sinon

CALCUL   DEY
         LDA  (ADR),Y     ;Poids faible de l'opérande
         TAX              ; en X
         INY
         LDA  (ADR),Y     ;Poids fort de l'opérande en A
         SEC              ;Comparaison de l'opérande et
         SBC  #>DEBUT     ;Adresse d'assemblage < opérande?
         BCC  ]non
         JSR  TEST        ;Opérande dans les limites?
         BCS  ]non        ;Non: instruction suivante
         ADC  ADRESSE     ;Si oui calcule l'octet de poids
         STA  (ADR),Y     ; fort de l'adresse translatée
         CLC
         RTS
]non     SEC
         RTS

* Variables du relogeur
* ---------------------
OFFSET   DFB  2,2,4,2,2,3,2,2,4,0,3,3;Offset pour les codes
           de $C0 à $CB
ADRESSE  DS   1           ;Adresse du relogement

         DS   DEBUT-*     ;Le début doit être aligné

*********************************
*                               *
*   Début du code relogeable    *
*                               *
*********************************

* Interpréteur des commandes
* --------------------------
```

```
21E0- A9 B1 2C 53 22 10 02 A9
21E8- B2 20 ED FD A9 A0 20 ED
21F0- FD A9 BD 20 ED FD A9 A0
21F8- 20 ED FD AD 53 22 29 0F
2200- 8D 53 22 D0 06 20 2F 22
2208- 18 90 13 A9 AF 20 ED FD
2210- C8 B9 00 02 09 80 20 ED
2218- FD CE 53 22 D0 F2 A9 8D
2220- 20 ED FD CE 52 22 10 90
2228- 18 60 20 8B BE 38 60 C8

2230- B9 00 02 A2 0A C9 28 F0
2238- 0C A2 1E C9 45 F0 06 C9
2240- 52 F0 02 A2 00 BD 5F 22
2248- F0 06 20 ED FD E8 D0 F5
2250- 60 00 00 00 02 00 00 02
2258- 04 00 00 00 00 00 00 C9
2260- AF CF A0 C5 D2 D2 CF D2
2268- 00 CE CF A0 C4 C5 D6 C9
2270- C3 C5 A0 C3 CF CE CE C5
2278- C3 D4 C5 C4 00 CE CF D4

2280- A0 C1 A0 D0 D2 CF C4 CF
2288- D3 A0 D6 CF CC D5 CD C5
2290- 00 21 00 00 C2 D3 C1 D6
2298- C5 D2 C9 C6 D9 C2 CC CF
22A0- C1 C4 C5 CC C5 D4 C5 C3
22A8- C1 D4 C1 CC CF C7 CF D0
22B0- C5 CE D7 D2 C9 D4 C5 D8
22B8- C5 C3 D2 C5 C1 D4 C5 C6
22C0- D2 C5 D3 D4 CF D2 C5 CE
22C8- C1 CD C5 C2 D2 D5 CE CC
```

```
* Retour à une éventuelle autre commande
* si aucune des commandes n'est reconnue

        CLD             ;Non indispensable mais...
        LDA  VPATH1     ;Adresse-1 de la commande entrée
        STA  ADR        ;mise dans un pointeur temporaire
        LDA  VPATH1+1
        STA  ADR+1

        LDA  #0         ;Initialise le compteur
        STA  NOCMD      ; des commandes
        LDX  #$FF       ;Initialise sur la lière commande
]cherche LDY #$00       ;Pointe ler caractère de l'entrée
]compare INX            ;Caractère suivant
        INY
        LDA  NEWCMD,X   ;Charge une lettre de la commande
        AND  #$7F
        BNE  ]suivant
        DEY             ;ler caractère ?
        BNE  trouvé     ;Non alors comparaison complète
        BEQ  ]comsui
]suivant CMP (ADR),Y    ;Comparaison avec l'entrée
        BEQ  ]compare   ;Identiques -> suivante
        EOR  #$20       ; sinon compare
        CMP  (ADR),Y    ; avec une minuscule
        BEQ  ]compare   ;Identiques -> suivante
]autre  INX             ;Recherche un pointeur commande
        LDA  NEWCMD,X
        BNE  ]autre     ;Début d'une commande ?
]comsui INC  NOCMD      ;Commande suivante
        LDA  NOCMD
        CMP  NBCMD      ;Encore une ?
        BNE  ]cherche   ;Va tester la commande suivante
        SEC             ;Commande non reconnue
PRECMD  JMP  XRETURN    ;Saut à la commande précédente

trouvé  DEY
        STY  LNCMD      ;Mémorise la longueur-1
        LDA  ]vecteur+1 ;Charge l'adresse de
        STA  XTRNADDR   ; retour vers la commande
        LDA  ]vecteur+2
        STA  XTRNADDR+1
        LDA  #0         ;Aucun paramètre autorisé
        STA  PBITS
        STA  PBITS+1
        STA  XCNUM      ;Indique commande externe
        CLC             ;La commande a été trouvé
        RTS

*  Vecteur d'indirection vers les commandes
*  ----------------------------------------
* (JMP est indispensable pour permettre le relogement)

]vecteur JMP  DISPATCH  ;Vecteurs des commandes


* DISPATCH des commandes installée
* --------------------------------
*  Essai des commandes externes l'une après l'autre

DISPATCH CLD
        LDX  NOCMD      ;Nombre de commandes
        BEQ  GONLINE
        DEX
        BEQ  GOBYE
        DEX             ;N° de la commande du BI
        LDY  #0
]nocr   LDA  BUFFIN,Y
        CMP  #$8D
        BEQ  ]décale
        INY
        BNE  ]nocr
]décale LDA  BUFFIN,Y
        STA  BUFFIN+8,Y ;Décalage de 8 octets
        DEY
        CPY  LNCMD      ;Compare avec la longueur
        BNE  ]décale
        TYA
        SEC
        ADC  #8
        TAY
        LDA  #$A0       ;Espace
]efface STA  BUFFIN-1,Y
        DEY
        BNE  ]efface
        LDY  PROLNG,X   ;Longueur de la commande BI
        LDA  PROVEC,X   ;Offset du dernier caractère
        TAX
]command LDA PROCMD-1,X
        STA  BUFFIN-1,Y
        DEX
        DEY
        BNE  ]command
        JMP  DOSCMD     ;Exécution de la commande, et fin
        (retour avec un traitement d'erreur eventuel)
GOBYE   JSR  MLI        ;Au revoir !
        DFB  QUIT
        DA   QUITPARA

GONLINE JSR  MLI        ;Teste tous les lecteurs
        DFB  ONLINE
        DA   ONLINEPA
        BCS  ]finerr
        LDA  DEVCNT
        STA  CNT

]device LDA  #"S"
        JSR  COUT
        LDA  CNT
        ASL
        ASL
        ASL
        ASL
        TAY
        LDA  BUFFIN,Y
        STA  BYTO
        AND  #$7F       ;Isole le slot
        LSR
        LSR
        LSR
        LSR
        ORA  #"0"       ;Conversion en code ascii
        JSR  COUT
        LDA  #","
        JSR  COUT
        LDA  #"D"
        JSR  COUT
        LDA  #"1"
        BIT  BYTO
        BPL  ]d2
```

```
22D0- CF C3 CB C3 C8 C1 C9 CE
22D8- A3 C6 CC D5 D3 C8 F2 E5
22E0- E1 E4 D0 CF D3 C9 D4 C9
22E8- CF CE CF CD CF CE D0 D2
22F0- A3 D0 D2 C5 C6 C9 D8 C3
22F8- CC CF D3 C5 C1 D0 D0 C5
2300- CE C4 AD 6E 0E 3B 05 1A
2308- 16 44 68 2B 13 26 4A 2E
2310- 45 0E 3F 5A 1E 56 63 5D
2318- 4E 37 33 3B 05 33 3F 09
2320- 23 6F 06 05 04 05 07 03
2328- 05 05 06 06 04 05 03 03
2330- 04 04 05 04 08 06 03 04
2338- 06 07 03 04 05 06 06 05
2340- 01 CF CE CC C9 CE C5 8D
2348- 00 C2 D9 C5 8D 00 00 C2
2350- CC 00 C2 D2 00 C2 D3 00
2358- C3 C7 00 C3 D4 00 C3 C8
2360- 00 C3 CC 00 C3 D2 00 C4
2368- CC 00 C5 D8 00 00 00 C9
2370- 00 4C 44 00 CC CB 00 00
2378- 00 00 D0 D8 00 D0 00 00
2380- D2 CE 00 00 00 D3 C1 00
2388- 00 D5 CE 00 D6 C5 00 00
2390- 00
```

```
        LDA  #"2"
]d2     JSR  COUT
        LDA  #" "
        JSR  COUT
        LDA  #"="
        JSR  COUT
        LDA  #" "
        JSR  COUT
        LDA  BYT0
        AND  #$0F       ;Isole la longueur du nom
        STA  BYT0
        BNE  ]noerr     ;Erreur ?
        JSR  PRINTERR
        CLC
        BCC  ]suite     ;Toujours pris
]noerr  LDA  #"/"       ;Affiche le nom du volume
        JSR  COUT
]affiche INY
        LDA  BUFFIN,Y
        ORA  #$80
        JSR  COUT
        DEC  BYT0
        BNE  ]affiche

]suite  LDA  #$8D       ;Saute une ligne
        JSR  COUT
        DEC  CNT        ;lecteur suivant
        BPL  ]device
        CLC             ;Pas d'erreur
        RTS             ;Retour au BI

]finerr JSR  BADCALL    ;Convertie l'erreur MLI en BI
        SEC
        RTS             ;Retour avec affichage de l'erreur

* Affichage de l'erreur
PRINTERR INY
        LDA  BUFFIN,Y   ;Code de l'erreur
        LDX  #<NOCONNEC-TABERR
        CMP  #$28
        BEQ  ]afferr
        LDX  #<NOPRODOS-TABERR
        CMP  #$45
        BEQ  ]afferr
        CMP  #$52
        BEQ  ]afferr
        LDX  #<IOERR-TABERR
]afferr LDA  TABERR,X
        BEQ  ]fin       ;Fin du message
        JSR  COUT
        INX
        BNE  ]afferr    ;Toujours pris
]fin    RTS

*  Variables et constantes
*  =======================
        DFB  0          ;Séparateur


* Paramètres et variables pour QUIT & ONLINE
* ------------------------------------------
CNT     DS   1
BYT0    DS   1
ONLINEPA DFB 2
        DFB  0          ;Device number = 0  <=> tous
        DA   BUFFIN     ;Buffer de $100 octets seulement
QUITPARA DFB 4
        DFB  0
        DA   0
        DFB  0
        DA   0
TABERR  EQU  *
IOERR   ASC  "I/O ERROR"
        DFB  0
NOCONNEC ASC "NO DEVICE CONNECTED"
        DFB  0
NOPRODOS ASC "NOT A PRODOS VOLUME"
        DFB  0
```

```
* Constantes et variables de l'interpréteur
* ------------------------------------------
NBCMD   DFB  PROLNG-PROVEC+2;Nombre de commandes
NOCMD   DS   1          ;Numéro de la commande trouvée
LNCMD   DS   1          ;Longueur de la commande trouvée

PROCMD  EQU  *          ;Table des commandes ProDOS
bsave   ASC  "B"
save    ASC  "SA"
verify  ASC  "VE"
bsave.  =    *
save.   =    *
        ASC  "RIFY"
verify. =    *
bload   ASC  "B"
load    ASC  "LOA"
delete  ASC  "D"
bload.  =    *
load.   =    *
        ASC  "ELETE"
delete. =    *
catalog =    *
cat     ASC  "CAT"
cat.    =    *
        ASC  "ALOG"
catalog. =   *
open    ASC  "OPEN"
open.   =    *
write   ASC  "WRIT"
exec    ASC  "E"
write.  =    *
        ASC  "XE"
create  ASC  "C"
exec.   =    *
        ASC  "REATE"
create. =    *
fre     ASC  "F"
restore ASC  "RE"
fre.    =    *
store   ASC  "STO"
rename  ASC  "RE"
restore. =   *
store.  =    *
        ASC  "NAME"
rename. =    *
brun    ASC  "B"
run     ASC  "R"
unlock  ASC  "UN"
brun.   =    *
run.    =    *
lock    ASC  "LOCK"
unlock. =    *
lock.   =    *
chain   ASC  "CHA"
in      ASC  "IN"
chain.  =    *
        ASC  "$"
in.     =    *
flush   ASC  "FLUSH"
flush.  =    *
read    ASC  "read"
read.   =    *
position ASC "POSITIO"
nomon   ASC  "N"
position. =  *
        ASC  "OMON"
nomon.  =    *
pr      ASC  "PR$"
pr.     =    *
prefix  ASC  "PREFIX"
prefix. =    *
close   ASC  "CLOSE"
close.  =    *
append  ASC  "APPEND"
append. =    *
tiret   ASC  "-"
tiret.  =    *
```

```
PROVEC  EQU  *          ;Offset des adresses des noms (der
                        nier caractère)
        DFB  append.-PROCMD
        DFB  bload.-PROCMD
        DFB  brun.-PROCMD
        DFB  bsave.-PROCMD
        DFB  catalog.-PROCMD
        DFB  cat.-PROCMD
        DFB  chain.-PROCMD
        DFB  close.-PROCMD
        DFB  create.-PROCMD
        DFB  delete.-PROCMD
        DFB  exec.-PROCMD
        DFB  flush.-PROCMD
        DFB  fre.-PROCMD
        DFB  in.-PROCMD
        DFB  load.-PROCMD
        DFB  lock.-PROCMD
        DFB  nomon.-PROCMD
        DFB  open.-PROCMD
        DFB  position.-PROCMD
        DFB  prefix.-PROCMD
        DFB  pr.-PROCMD
        DFB  read.-PROCMD
        DFB  rename.-PROCMD
        DFB  restore.-PROCMD
        DFB  run.-PROCMD
        DFB  save.-PROCMD
        DFB  store.-PROCMD
        DFB  unlock.-PROCMD
        DFB  verify.-PROCMD
        DFB  write.-PROCMD
        DFB  tiret.-PROCMD

PROLNG  EQU  *          ;Longueur des noms
        DFB  append.-append
        DFB  bload.-bload
        DFB  brun.-brun
        DFB  bsave.-bsave
        DFB  catalog.-catalog
        DFB  cat.-cat
        DFB  chain.-chain
        DFB  close.-close
        DFB  create.-create
        DFB  delete.-delete
        DFB  exec.-exec
        DFB  flush.-flush
        DFB  fre.-fre
        DFB  in.-in
        DFB  load.-load
        DFB  lock.-lock
        DFB  nomon.-nomon
        DFB  open.-open
        DFB  position.-position
        DFB  prefix.-prefix
        DFB  pr.-pr
        DFB  read.-read
        DFB  rename.-rename
        DFB  restore.-restore
        DFB  run.-run
        DFB  save.-save
        DFB  store.-store
        DFB  unlock.-unlock
        DFB  verify.-verify
        DFB  write.-write
        DFB  tiret.-tiret

NEWCMD  EQU  *          ;Table des nouvelles commandes
        ASC  "ONLINE"
        DFB  $8D,0      ;online + CR (pas de paramètre)
        ASC  "BYE"
        DFB  $8D,0      ;bye + CR (idem)
        DFB  0          ;append
        ASC  "BL"
        DFB  0          ;bload
        ASC  "BR"
        DFB  0          ;brun
        ASC  "BS"
        DFB  0          ;bsave
        ASC  "CG"
        DFB  0          ;catalog
        ASC  "CT"
        DFB  0          ;cat
        ASC  "CH"
        DFB  0          ;chain
        ASC  "CL"
        DFB  0          ;close
        ASC  "CR"
        DFB  0          ;create
        ASC  "DL"
        DFB  0          ;delete
        ASC  "EX"
        DFB  0          ;exec
        DFB  0          ;flush
        DFB  0          ;fre
        ASC  "I"
        DFB  0          ;in#
        ASC  "LD"
        DFB  0          ;load
        ASC  "LK"
        DFB  0          ;lock
        DFB  0          ;nomon
        DFB  0          ;open
        DFB  0          ;position
        ASC  "PX"
        DFB  0          ;prefix
        ASC  "P"
        DFB  0          ;pr#
        DFB  0          ;read
        ASC  "RN"
        DFB  0          ;rename
        DFB  0          ;restore
        DFB  0          ;run
        ASC  "SA"
        DFB  0          ;save
        DFB  0          ;store
        ASC  "UN"
        DFB  0          ;unlock
        ASC  "VE"
        DFB  0          ;verify
        DFB  0          ;write
        DFB  0          ;tiret
        DS   FIN-*
*BF/26/5/86
```

Accompagné d'une cinquantaine de pages de documentation, Disk Manager permet de recréer les commandes du Dos, redéfinir l'organisation d'une disquette, grâce à un jeu d'instructions qui en fait un langage simple d'accès à la disquette. Il offre également un programme d'édition à l'aide de commandes évoluées. 4 utilitaires figurent aussi sur la disquette :

- Utili-disque : reconstruction d'une disquette détruite, Vérification, Plan d'occupation
- Ultra-copie : pour un backup particulièrement rapide
- Edicat : Edition du catalogue, classement des fichiers, Titres...
- Multi-disque : pour le classement de tous vos programmes (tri instantané).

## δisk Manager le Dos en Kit

*de Dan Steerey*

# Pour un Boot ergonomique : Startup Pascal complet

*Daniel Mueller*

Le PASCAL UCSD offre la possibilité de dater les fichiers, mais on oublie très souvent de mettre à jour la date lors du Boot, car il faudrait appeler le FILER puis choisir l'option 'D)ate' puis 'Q)uit'.

C'est donc le but du programme qui suit avec des petits *plus*.

En effet, il permet :

- de mettre à jour la date en mémoire et sur disquette, la syntaxe étant la suivante :

```
jour-mois-annee
   jour-mois
  -mois-annee
  jour--annee
      jour
     -mois
    --annee
```

- de placer automatiquement le nom du volume 5 (lecteur 2) en mémoire comme préfixe courant ;
- d'afficher tous les fichiers 'CODE' du volume 5 et d'exécuter l'un d'eux en pressant simplement sur une touche ;
- d'appeler directement l'éditeur ou le Filer ;
- de quitter le programme.

Pour cela, il a fallu rechercher les adresses de la date et du préfixe en mémoire pour le système Pascal 1.2., à l'aide du programme Prg.1. TEXT. Elles sont différentes pour les versions 64Ko et 128Ko. Le programme Startup utilise celles du Pascal 1.2 128Ko.

## Le principe

La date occupe 2 octets et le préfixe 8. Sur disquette, il faut savoir que pour chaque fichier le système réserve 26 octets et le nombre maximum de fichiers est 78. La structure apparaît clairement dans le programme.

Pour lire et écrire le DIRECTORY (qui débute au deuxième bloc et occupe 4 blocs) nous utilisons respectivement les fonctions UNITREAD et UNITWRITE.

## Programme 'PRG.1.TEXT'

```
program recherche;

type T_date = record case boolean of
                    true  : (adresse : integer);
                    false : (datesys : ^R_date)
                   end;

     R_date = packed record
                mois   : 1..12;
                jour   : 1..31;
                annee  : 0..100
              end;


var date   : T_date;
    Ddate : R_date;

begin
    ( Initialise Ddate avec la date actuellement en memoire
      aller dans le FILER et mettre par ex: 1 jan 86 )
   with Ddate do
   begin
      jour  := 01;
      mois  := 01;
      annee := 86;
   end;
   with date do
   begin
      adresse := -32767; ( $8000 )
      while adresse < -16385 ( $BFFF ) do
      begin
         if datesys^ = Ddate then writeln ('adresse ',adresse);

                                  ( Comparaison de la date en memoire avec Ddate )

         adresse := adresse + 1                          ( Incremente l'adresse )
      end ( while )
   end  ( with )
end.

( Initialiser la date en memoire et Ddate du programme avec les meme valeurs.
  Lancer le programme ci-dessus et relever toutes le adresses affichees
  puis changer seulement la date en memoire (par le FILER) et relancer
  le programme. L'adresse de la date sera celle qui n'a pas ete affichee. )
```

Adresse de base version 1.2

| | 128Ko | 64Ko |
|---|---|---|
| Date | $B85A | $ACFC |
| Préfixe | $B84A | $ACEC |

Pour l'adresse de la date des versions 1.0 et 1.1, on peut se référer à l'article de M. Pascual paru dans le numéro 16, l'adresse du préfixe étant obtenue en ajoutant 16 à celle de la date.

Si l'on désire 'booter' sur ce programme il faudra le sauvegarder sous le nom de :

*SYSTEM.STARTUP*

sur le lecteur 1, ainsi il sera exécuté automatiquement.

Ce programme tourne sur Apple // possédant deux lecteurs de disquette. Pour un système avec un seul lecteur, il faudra modifier la ligne indiquée dans le listing. (procédure FIX_VOL)

## Bibliographie


APPLE Pascal, Operating system reference manual, Language reference manual,

Découvrez PASCAL (tome 2) sur Apple ][, //e, //c de John Colibri édité par MNEMODYME,

Le système Pascal UCSD (tome 2) de Thierry CHAMORET édité par EDITEST.


## Programme 'STARTUP.TEXT'

```
program STARTUP;

uses chainstuff { Setchain (FICHIER),
                  rajoute le suffixe .CODE a FICHIER
                        »puis l'execute };

const boot_vol = 4; { Drive 1 ou volume #4 }
      cour_vol = 5; { Drive 2 ou volume #5 }
      maxlong  = 7; { Long max. du nom de volume }
      adrdate  = -18342; { Adresse de la date pour Pas
                        »cal [1.2] 128K  0B85A}
      adrfix   = -18358; { Adresse du prefixe pour Pas
                        »cal [1.2] 128K  0B84A }
      dash     = '-';
      month    = '???JanFebMarAprMayJunJulAugSepOctNovDe
                        »c';

type  system   = record case integer of
                   1 : (datesys   : ^r_date); { Poke po
                         »ur la date }
                   3 : (volumesys : ^vol);    { Poke po
                         »ur le volume }
                   2 : (adresse   : integer)  { Adresse
                         » en memoire }
                 end; { record }

      r_mois   = 1..12;

      r_date   = packed record
                   mois  : r_mois;
                   jour  : 1..31;
                   annee : 0..100
                 end; { record }

      vol      = string [maxlong];

      volume   = 4..12; { Numero des volumes }

      ordre    = (lire,ecrire); { Ordre pour la proced
                        »ure IO_Disk }

      strfile  = string [18];

      longstring = string [255];

      filetyp  = (untyped,xdsk,code,text,info,data,gra
                        »f,foto,secure);
      { Types de fichiers utilises par Pascal }

      infodir  = packed record
                   prebloc    : integer;  { Premier blo
                         »c physique de la disq. }
                   derbloc    : integer;  { Dernier blo
                         »c du directory      }
                   case filetyp of
                     secure,untyped : { Seulement DIREC
                         »TORY[0]
                                      informations po
                         »ur le Volume }
                                                    (filler1
                         »: 0..2048;
       { Nom de la disquette }                      nom_disk
                         »: vol;
       { Numero du dernier bloc du volume }         bloc_nbre
                         »: integer;
       { Nombre max. de fichiers sur disquette }    file_nbre
                         »: 0..77;
       { Initialise - heure du dernier acces }       time
                          »: integer;
       { Date courante }                            boot_date
                         »: r_date);

                   xdsk,code,text,info,
                   data,graf,foto :
                                                    (filler
                         »: 0..1024;
       { Utilise par le FILER }                     etat
                         »: boolean;
       { Nom du fichier }                           filename
                         »: string[15];
       { Nbre d'octets ds le der. bloc du fichier} nbreoctet
                         »: 1..512;
       { Date de la derniere modification }         modifdate
                         »: r_date)

                 end { record };


var directory  : array [0..77] of infodir;
    prefix     : vol;
    fichier    : strfile;
    date       : r_date;
    memoire    : system;
    ch         : char;
    ok         : boolean;

{$U-}

procedure IO_DISK (commande : ordre; drive : volume);
{ Procedure gerant les entrees/sorties avec les lecteu
                        »rs disquettes
  et interceptant les erreurs }

 function no_ERREUR : boolean;
  var num_error : integer;

    procedure write_ERREUR;
     const inverse = 15; { Mode inverse }
           normal  = 14; { Mode normal  }
     var ch2 : char;
    begin
      page (output);
      gotoXY (10,08);
```

```
        write (chr (inverse));
        case num_error of
             2  : write ('NUMERO DE VOLUME INCORREECT'
                         »);
             5  : write ('PERTE DU VOLUME');
             7  : write ('NOM DE FICHIER ILLEGAL');
             9  : write ('LE VOLUME SELECTIONNE N''EST
                         » PAS EN LIGNE');
             64 : write ('ERREUR GENERALE ACCES DISQUE
                         '»)
        end; { case }
        write (chr (normal));
        gotoXY (10,10);
        write (chr (7),'CONTROLER VOTRE LECTEUR OU LA
                         »DISQUETTE',chr (7));
        gotoXY (10,12);
        write ('Pressez une touche quelconque pour con
                         »tinuer ');
        repeat
          read (keyboard,ch2);
        until ch2 > chr (0);
        page (output);
      end;  { Write_ErreuR }

  begin  { No_ErreuR }
     no_erreur := false;
     num_error := IORESULT; { Numero de l'erreur }
     if (num_error = 0) or (num_error = 16) { 16 -> di
                         »squette protegee }
        then no_erreur := true
        else write_ERREUR;
  end;  { No_ErreuR }

{$I-}

begin  { IO_disK }
   repeat
      case commande of
           ecrire : unitwrite (drive,directory,sizeof
                         »(directory),2);
           lire   : unitread  (drive,directory,sizeof
                         »(directory),2);
      end; { case }
   until no_erreur;
end;  { IO_disK }

{$I+}


procedure FIXE_VOL;
{ Fixe le nom du volume #5 en memoire }
begin
   { Lit sur le drive 2 le nom de la disquette }
   IO_disk (lire,cour_vol);
   { Pour un systeme avec un seul lecteur remplacer ce
                         »tte ligne par :
     IO_disk (lire,boot_vol) }
   prefix := directory[0].nom_disk;
   { Le fixe en memoire }
   with memoire do
   begin
      adresse := adrfix;
      volumesys^ := prefix;
      end; { with }
end { Fixe_voL };



procedure FIXE_DATE (disk : boolean);
{ Lit la date sur volume #4
  ou la met a jour en memoire et sur disquette }
begin
   if disk
      then begin
           directory[0].boot_date := date;

{ Si la disquette est protegee contre l'ecriture
  il n'y aura pas de message d'erreur
  la date ne sera remise a jour seulement en memoire }
           IO_disk (ecrire,boot_vol);
           end { if then }
      else begin
           IO_disk (lire,boot_vol);
           date := directory[0].boot_date;
           end { if else };
{ Fixe la date en memoire}
   with memoire do
   begin
      adresse := adrdate;
      datesys^ := date;
   end; { with }
end;  { Fixe_datE }


procedure CATALOG (var strn : strfile);
{ Lit le DIRECTORY de #5 est affiche seulement les fic
                         »hiers codes (max. 58)
  possibilite d'EXECUTER l'un de ces fichiers en pres
                         »sant
  la lettre correspondante }
 var n,k,col : integer;
     ch1     : char;
     pile    : packed array ['A'..'z'] of integer; { M
                         »AX 58 fichiers }
begin
   repeat
     strn := '';
     n := 0; { Compteur fichiers codes }
     k := 0; { Compteur general }
     fixe_vol;
     page (output);
     gotoXY (27,2);
     write ('DIRECTORY  : ',prefix);
     repeat
       k := k + 1; { On commence par directory[1] car
                         »directory[0] etant
                   reserve pour les informations gen
                         »erales de la disquette }
       with directory[k] do
          if pos ('.CODE',filename) <> 0 { Position de
                         » .CODE }
             then begin
                  strn := filename;
                  delete (strn,pos ('.CODE',strn),5);
                         »{ Efface .CODE }
                  { Mise en colonnes (15 p.col.) des f
                         »ichiers trouves }
                  col := (n div 15) * 20; { Quelle col
                         »onne ? }
                  gotoXY (col+2,4+n mod 15);
                  write ('[',chr (ord ('A')+n),'] ',st
                         »rn);
                  pile [chr (ord('A')+n)] := k; { Rang
                         » du fich. dans DIR }
                  n := n + 1;
                  end   { if then };
     until (k = directory[0].file_nbre) or (n = 58);
     if n = 0 then begin
                  gotoXY (10,15);
                  write (chr (7),'AUCUN FICHIER CODE
                         »!!!',chr (7));
                  for n := 1 to 2000 do;
                  ok := false;
                  exit (catalog);
                  end; { if then }
     gotoXY (16,23);
     write (directory[0].file_nbre,' fichiers dont ',n
                         »,' fichiers ''CODE''.');
     gotoXY (0,00);
     write ('Taper une lettre - <ESC> pour quitter - <
                         »SPACE> pour relire ? ');
     repeat
       read (keyboard,ch1);
     until ch1 in ['A'..chr (ord ('A')+n-1),chr (27),'
                         » '];
     if ord (ch1) = 27
        then begin
             ok := false;
```

```
            exit (catalog);
            end; { if then }
   until ch1 <> ' ';
   n := pile [ch1]; { Rang du fichier dans Directory }
   { Le nom du fichier est retourne dans STRN }
   strn := concat ('#5:',directory[n].filename);
end   { CataloG };


procedure DATESET;
{ Saisie de la date
   - possibilite de rentrer seulement le jour
   - le jour et le mois (sur trois lettres)
   - ou en entier
   - controle sa validite
   - sauvegarde sur la disquette du boot si differente
                              » }

 var temp      : integer;
     mois_lit  : string [3]; { Represente le mois sous
                              » sa forme litterale }
     fini      : boolean;
     strg_date : longstring; { String pour la saisie d
                              »e la date }

   procedure EFFACE;
   { Efface jusqu'au premier DASH ,y compris }
   begin
      delete (strg_date,1,scan (length (strg_date),=da
                              »sh,strg_date[1]));
      if length (strg_date) > 0
         then delete (strg_date,1,1);
   end;  { EffacE }

{ ------------------------------------------------------
  La function SCAN ( limite,expression,debut ) : integ
                              »er;

  limite     : integer; est le nombre maximum d'octets
                              » a explorer
  expression : boolean; caractere a rechercher
   EX : = CH   (le caractere a rechercher egal a CH)
        <> CH  ( "      "    "    "       different de
                              » CH)
        sont les deux seules possibiltes.
  debut      : n'importe quel type sauf FILE; est le p
                              »remier octet de la
               variable a partir duquel la recherche d
                              »ebute.

  La function SCAN retourne la position du premier oct
                              »et verifiant EXPRESSION
  Si elle est en tete de la valeur a explorer, la posi
                              »tion sera egale a zero.
  ------------------------------------------------------ }

   function TEST_ENTIER (max : integer) : boolean;
   { Transforme STRG_DATE en entier et teste si TEMP <
                              » MAX,
     la valeur sera dans TEMP }
    var x, stop : integer;
   begin { Test_entier }
      temp := 0;
      stop := scan (length (strg_date),=dash,strg_date
                              »[1]);
      for x := 1 to stop do
        if strg_date[x] in ['0'..'9']
        then temp := temp * 10 + ord (strg_date[x]) -
                              »ord ('0');
        test_entier := (temp > 0) and (temp <= max);
   end;   { Test_entieR }

   function ANALYSE_M : boolean;
   { Test si les 3 premiers carac. de SRTG_DATE sont d
                              »es lettres}
    var x : boolean;
        w : integer; { Compteur }
   begin
      w := 1;
      x := true;
      mois_lit := '???';
      if length (strg_date) > 0
         then while (w < 4) and x do
                if strg_date[w] in ['A'..'V']
                   then begin
                        if w = 1
                           then mois_lit[w] := strg_da
                                »te[w]
 { La premiere lettre en MAJUS. et les 2 suivantes en m
                                »inus. }
                           else  mois_lit[w] := chr (o
                                »rd (strg_date[w]) + 32);
                        w := w + 1;
                        end { if then }
                   else x := false
         else x := false;
      analyse_M := x;
end;  { Analyse_M }


function ESPACE : boolean ;
{ Supprime les espaces ainsi que les caracteres < 32 e
                              »t > 123
  et converti tout en majuscules }
 var w : integer; { Compteur }
begin
   w := 1;
   espace := true;
   while w <= length (strg_date) do
      if (ord (strg_date[w]) >= 33 ) and (ord (strg_da
                              »te[w]) <= 122)
         then begin
              if (strg_date[w] >= 'a') and (strg_date[
                              »w] <= 'z')
              then strg_date[w]
                   := chr (ord (strg_date[w]) - ord ('
                              »a') + ord ('A'));
              w := w + 1;
              end { if then }
         else
            delete (strg_date,w,1);
   if length (strg_date) = 0 then espace := false;
end;  { EspacE }


function TEST : boolean;
{ Test la validite de la date (nbre de jours dans chaq
                              »ue mois
  et annees bisextiles) }

 var mois_30 : set of r_mois; { MOIS_30 := mois de 30
                              »jours }
     ok1     : boolean;

  procedure MESSAGE;
   var wait : integer;
  begin
     gotoXY (3,20);
     write (chr (7),'<< ERREUR dans la DATE >>');
     for wait := 1 to 2000 do;
     gotoXY (3,20);
     write (chr (29));
  end;  { MessagE }

begin  { TesT }
   ok1 := true;
   test := true;
   mois_30 := [4,6,9,11]; { mois de 30 jours sont plus
                              » nombreux que ce deux 31
                              }
   with date do
   begin
      if (mois in mois_30) and (jour >= 31) then ok1 :
                              »= false;
      { Test le mois de fevrier }
      if (annee mod 4 = 0) and (mois = 2) and (jour >=
                              » 30) then ok1 := false;
      if (annee mod 4 > 0) and (mois = 2) and (jour >=
```

```
                        » 29) then ok1 := false;
    end; { with }
    { Si la date est fausse on restaure l'ancienne + me
                        »ssage d'erreur }
    if not ok1
       then begin message;
            test := false;
            with directory[0].boot_date do
            begin
               date.jour  := jour;
               date.mois  := mois;
               date.annee := annee;
            end;  { with }
       end;  { if then }
end;  { TesT }

begin { DateseT }
    fixe_date (false);
    page (output);
    gotoXY (0,2);
    writeln ('Saisie de la date : <1..31>-<Jan..Dec>-<0
                        »..99>');
    with date do
       write
       ('La date actuelle est le ',jour,dash,copy (mont
                        »h,mois*3+1,3),dash,annee)
                        »;
    repeat
      fini  := false;
      gotoXY (00,5);
      write ('Entrer la date ? ');
      write (chr (29)); { EOL efface jusqu'a la fin de
                        »la ligne }
      readln (strg_date); { Saisie de la date }
      gotoXY (0,0); { Question d'esthetique }
      if espace
         then with date do
              begin
                    { TRAITEMENT PRINCIPAL DE STRG_DA
                        »TE
                    - jour -      }
                 if test_entier (31)
                    then jour  := temp;
                 efface; { Efface jusqu'au prochain '-'
                        ») }

                 {      - mois -      }
                 temp := 1; { Initialise le compteur po
                        »ur 'mois' }
                 if (length (strg_date) > 2) and analys
                        »e_M
                    then while (temp < 13) and not fini
                        » do
                        { Verifie si 'mois_lit' exis
                        »te dans 'month' et
                        initialise mois  Ex: Jan=1
                        »,Fev=2,...}
                        if copy (month,temp * 3 + 1,
                        »3) = mois_lit
                          then begin
                               mois := temp;
                               fini := true;
                               end { if then }
                          else temp := temp + 1;
                 efface; { Efface jusqu'au prochain  '-
                        »') }

                 {     - annee -      }
                 if (length (strg_date) > 0) and test_e
                        »ntier (99)
                    then annee := temp;
              end   { with };
    until test;
    { Si la date est differente on la met a jour sur  d
                        »isque et memoire }
    if not (date = directory[0].boot_date) then fixe_da
                        »te (true)
end;  { DateseT }
```

## La méthode

*Vous n'avez pas la disquette d'accompagnement : il vous faut saisir et compiler le listing de façon classique.*

*Vous avez la disquette d'accompagnementt : Il faut initialiser une disquette Pascal à l'aide du 'FORMATER'. Booter alors sur la disquette Pom's et lancer le programme de transfert 'BASIC-PASCAL'. Transférer alors le fichier STARTUP.TEXT en suivant les indications fournies par ledit programme. Il convient maintenant de compiler et d'exécuter comme habituellement.*

```
begin { STARTUP }
   fixe_date (false);
   fixe_vol;
   repeat
     ok := true;
     page (output);
     write ('Startup (DM): Q)uit, D)ate, F)iler, E)dit
                        »or, eX)ecute ? [1.2]');
     gotoXY (0,1);
     writeln ('Le PREFIXE courant est : ',prefix);
     write   ('La DATE actuelle est   : ');
     with date do
          write (jour,dash,copy (month,mois*3+1,3),das
                        »h,annee);
     gotoXY (61,0);
     repeat
       read (keyboard,ch);
       { Tout en majuscules }
       if ord (ch) > 96 then ch := chr (ord (ch) - 32)
                        ;
     until ch in ['D','E','F','Q','X'];
     case ch of
      'D' : dateset;
      'E' : fichier := '*SYSTEM.EDITOR.'; { Le point e
                        »st obligatoire pour les }
      'F' : fichier := '*SYSTEM.FILER.';  { fichiers s
                        »ans suffixe .CODE }
      'X' : catalog (fichier);{ Positionne ok si aucun
                        » fichier }
      'Q' : begin
              page (output);
              exit (program); { ON QUITTE }
            end;
       end; { case }
   until (ch in ['E','F','X']) and ok;
   page (output);
   setchain (fichier); { OK let's go }
end. { STARTUP }
```

*N.B. : Les caractères qui suivent le signe » sont en fait la suite de la ligne précédente.*

# Carte SSC & CP/M

J-F Rabasse

(1)

Les transmissions de données en série (norme RS232C) sont fréquemment utilisées sur micro-ordinateurs. L'application la plus simple est la gestion d'une imprimante série. La transmission série devient indispensable pour échanger des données entre deux calculateurs, ou bien en local par liaison filaire, lorsque les deux machines ont des systèmes d'exploitation et formats de disquettes différents, ou bien en *distant* par liaison téléphonique.

Cet article, en deux parties autonomes, se propose de montrer la gestion d'une Carte Super Série, sur un Apple en environnement CP/M.

## Pourquoi CP/M ?

Tout d'abord parce que c'est un peu le "parent pauvre" sur Apple, et de ce fait les réalisations de transmissions séries sont rares, alors qu'il existe une impressionnante bibliothèque de logiciels de communication sous DOS 3.3. D'autre part, les transferts de données entre un Apple et une autre machine se feront vraisemblablement sous CP/M qui est un SED très répendu. Si vous devez tranférer vers une machine utilisant DOS 3.3 ou ProDOS, il y a fort à parier qu'un simple échange de disquette suffira !

## Pourquoi la carte SSC ?

C'est d'abord la carte série la plus courante sur Apple, et d'autre part c'est une carte très performante au niveau électronique alors que le logiciel de gestion existant sur la ROM interne présente quelques défauts qui le rendent quasi-inutilisable dans beaucoup d'applications. Il est alors indispensable de revoir complétement la gestion de cette carte.

## La transmission série

Cette transmission nécessite une conversion des données, données existantes sous forme d'octets en une série de bits émis à une vitesse précise. Cette conversion, au départ comme à l'arrivée, est effectuée par un circuit spécialisé nommé UART ou ACIA. En émission, il suffira de charger avec la donnée le registre de transmission de l'ACIA, en réception de lire l'octet reçu dans le registre de réception.

L'ACIA se charge d'autre part de transférer la donnée selon un format bien précis, nombre de bits de données, nombre de bits d'arrêt, parité, vitesse.
Il suffit donc en théorie d'un seul fil (plus la masse électrique) entre les deux machines. En pratique, pour fonctionner correctement, une liaison série doit en plus assurer un certain nombre de contrôles afin d'éviter la perte de données au cours du transfert.
Ces contrôles, qui constituent le "protocole de liaison" (Hand-Shake des anglo-saxons), sont assurés par voie électrique (Hardware Handshake) en utilisant les signaux de contrôle de l'interface série, ou par voie purement logicielle en utilisant un programme de transmission respectant un protocole particulier. Les protocoles existants sont très nombreux. Il en existe de simples (ENQ/ACK ou D1/D3 appelé aussi XON/ XOFF), de moyennement évolués comme XMODEM ou de très élaborés (KERMIT).

Nous allons détailler dans cette première partie un exemple de protocole Hardware, interfacé avec le BIOS CP/M, et, dans la deuxième un protocole logiciel : XMODEM.

## Signaux RS232

Sur le connecteur DB25 d'une interface série, on trouve les signaux suivants :

*TXD* (Broche 2) Sortie émission de données.

*RXD* (Broche 3) Entrée réception de données.

*RTS* (Broche 4) Sortie demande pour émettre.

*CTS* (Broche 5) Entrée autorisation d'émettre.

*DSR* (Broche 6) Entrée interlocuteur prêt.

*Masse* (broche 7).

*DCD* (Broche 8) Entrée détection de porteuse (avec Modem).

*DTR* (Broche 20) Sortie prêt.

Les brochages de ces signaux sont en principe standardisés. Malheureusement, un certain nombre de constructeurs appliquent les normes avec poésie...

Remarque : pour les lecteurs intéressés, l'interface série CE130T des ordinateurs de poche Sharp série PC ne fonctionnera correctement avec le programme présenté ici qu'en utilisant la Broche 11 comme signal DTR au lieu de la Broche 20. Il faudra donc prévoir un câble de liaison spécial pendant la durée de la garantie de votre interface CE130T puis, ensuite, ouvrir la boîte et strapper directement sur le circuit imprimé, après coupure de la piste DTR. Cette opération est très simple à réaliser.

## Liaison avec un Modem

Tout ces signaux sont utilisés lors d'une liaison avec un Modem, il suffit d'utiliser un câble "fil pour fil".

### Liaison locale avec un autre calculateur

En liaison directe locale de deux machines, on peut simplifier et supprimer le Modem. Il est alors indispensable d'utiliser un câble particulier appelé "Éliminateur de Modem" afin de *croiser* deux à deux, signaux de données et signaux de contrôle.
La carte SSC possède un "Jumper Block" à deux positions (Terminal et Modem). En théorie, ceci devrait permettre une liaison locale avec un câble Modem normal ; en pratique, le câblage adopté par Apple et le logiciel de la ROM SSC rendent cette option inutilisable, car sans contrôle de transmission.
Il faudra donc laisser ce Jumper Block en position Modem, et réaliser un câble éliminateur.

### Câble éliminateur

Il est simple à réaliser, et ne consiste qu'à "croiser deux lignes" :

Lignes de données : TXD d'une machine vers RXD de l'autre (et inversement).
Lignes de contrôle : DTR d'une machine vers CTS de l'autre (et inversement).

## Gestion de la SSC

La carte SSC est gérée et configurée à l'aide d'un certain nombre de registres. Ces registres sont à des adresses qui dépendent du slot dans lequel cette carte est située. En principe, sous CP/M comme sous Pascal, cette carte est en Slot 2. Ces registres permettent de positionner les paramètres de la transmission (vitesse, nombre de bits, parité), de lire ou écrire les données et de lire ou positionner les signaux de contrôles. Pour plus de détails, se reporter au manuel de la carte SSC.

La carte comporte de plus des "switches" qui permettent de mémoriser une configuration par défaut et que tout programme de transmission peut aller lire.

### Émission

Pour émettre, il faut s'assurer que le registre de transmission est vide (afin d'éviter d'écraser l'octet précédemment émis, ceci surtout à basse vitesse), il faut ensuite s'assurer que le récepteur est bien prêt à recevoir la donnée, en lisant le signal CTS, et enfin écrire l'octet à envoyer dans le registre d'émission.

### Réception

Pour recevoir, il faut indiquer à l'émetteur que l'on est prêt en activant le signal DTR, attendre que le registre de réception soit plein, puis lire l'octet. Il est ensuite indispensable, avant de retourner au programme appelant, de désactiver le signal DTR, afin d'éviter que l'émetteur n'envoie un octet pendant que l'on aura le dos tourné. C'est cette précaution qu'Apple semble avoir oubliée dans son logiciel de gestion SSC, et qui seule vous permettra, par exemple, de renvoyer un fichier arrivant d'une autre machine, vers une imprimante ou tout périphérique lent, ou de stocker sur disque en temps réel un fichier.
Pour plus de sécurité, il est même préférable en lecture de n'activer DTR qu'après avoir vérifié qu'aucune donnée n'était présente, afin de solliciter l'émetteur.

## Programme SETSSC

Ce programme, sous la forme d'une commande CP/M (SETSSC .COM) a une double fonction. Il permet tout d'abord d'initialiser la carte SSC avec des paramètres en clair. En effet le codage de la liaison en vitesse, parité, etc. n'est pas toujours très parlant si l'on n'a pas le manuel de la SSC sous les yeux. Avec SETSSC on spécifiera directement les paramètres désirés.

Ensuite, ce programme implante dans le BIOS CP/M deux petites routines d'émission et de réception d'un octet, selon le protocole détaillé plus haut. Ces deux routines sont rendues accessibles à CP/M à travers les unités logiques **UR2:** et **UP2:** qui correspondent en standard à la lecture et l'écriture d'un perforateur de ruban, périphérique qui commence à se raréfier, surtout chez l'amateur !

Il n'y a besoin de rien d'autre, maintenant vous ferez vos transferts en utilisant ce bon vieux PIP qui est en fait un logiciel de transfert très pratique grâce à ses nombreuses options (tabulations, formatage, numérotations etc.)

## Utilisation

Votre carte SSC peut être configurée par défaut (switches) selon vos paramètres de transmission les plus fréquents.

L'appel du programme se fait au niveau 'commande' de CP/M par :

```
SETSSC <Par1>,<Par2>,…
```

Les paramètres sont optionnels et sont au nombre de quatre au maximum, séparés par des virgules (un espace entre SETSSC et les paramètres est nécessaire). Chaque paramètre est constitué d'une lettre suivie de caractères. On aura les commandes suivantes :

**Bxxxx** réglage de la vitesse où xxxx indique la vitesse en clair (et en bauds !). xxxx aura les valeurs 50 à 19200.

**Dx** nombre de bits de données. x a les valeurs 5 à 8.

**Sx** nombre de bits d'arrêt. x vaut 1 ou 2.

**Px** choix de la parité. x indique la parité, selon les dénominations françaises ou anglaises au choix. x vaudra :

| | | |
|---|---|---|
| 0 ou | S | parité 0 (Space) |
| 1 ou | M | parité 1 (Mark) |
| P ou | E | parité paire (Even) |
| I ou | O | parité impaire (Odd) |
| N | | pas de parité |

La commande SETSSC sans paramètres initialise la carte avec les valeurs par défauts des switches.

SETSSC avec des paramètres configure la carte selon les paramètres indiqués. Les paramètres non présents dans la ligne de commande sont laissés inchangés, ou sont lus sur les switches dans le cas d'un premier appel à SETSSC après un boot.

Par exemple :

**SETSSC** configure avec les switches.
**SETSSC B2400** passe la vitesse à 2400 bauds, le reste est inchangé.
**SETSSC D8,PN,S1** 8 bits de données, 1 stop, pas de parité, vitesse inchangée.
etc.

Toute commande incorrecte génère un message d'erreur et ne modifie rien sur la carte.
La configuration reste valide jusqu'au prochain démarrage à froid de CP/M.

La transmission de vos données se fera ensuite avec PIP. Par exemple :

Envoi du fichier TOTO.TXT vers la ligne série :

```
PIP  UP2:=TOTO.TXT
```

Réception d'un fichier et impression avec tabulation et numérotation :

```
PIP  LST:=UR2:[T8N]
```

etc.

### Remarque 1

Veillez à utiliser les unités logiques UP2: et UR2: plutôt que les unités PUN: et RDR: Ces deux dernières unités gèrent également le perforateur et le lecteur de ruban, mais réalisent en plus une "mise en page" du fichier transféré. Par exemple, avant émission du premier octet du fichier, CP/M envoie sur PUN: une série de 80 codes "Nul". C'est classique sur un ruban perforé, mais un certain nombre d'ordinateurs récepteurs risquent de ne pas apprécier la manœuvre.

### Remarque 2

L'utilisation de PIP implique l'utilisation de fichiers de type texte, terminés par Ctrl-Z, que PIP détecte comme fin de transmission. Il peut arriver que l'ordinateur qui vous envoie un fichier arrête son émission sans envoyer de Ctrl-Z (PIP le fait bien, pourquoi pas les autres ?). En ce cas, vous pouvez frapper un Ctrl-Z au clavier. La routine de lecture de la carte SSC, dans sa boucle d'attente, surveille aussi le clavier et retournera à CP/M tout caractère frappé comme provenant de la ligne série.
Ceci permet de reprendre la main correctement et fermer le fichier reçu.
(Ce problème n'existe pas avec les commandes de transmission des PC Sharp qui émettent effectivement le code fin de fichier.)

En émission, PIP arrête le transfert sur détection de Ctrl-Z, sans envoyer celui ci. Ceci peut éventuellement laisser l'ordinateur récepteur bloqué. Deux solutions :

- si l'ordinateur récepteur permet une reprise de main facile, pas de problème ;
- sinon, il faut disposer d'un moyen d'envoyer le Ctrl-Z sur la ligne série. En cas de besoin donc, il faudra *développer* (n'ayons pas peur des mots) le *logiciel* suivant :

Charger DDT. Assembler les instructions suivantes :

```
A100
   LHLD F390
   MVI  C,1A
   PCHL
```

puis Ctrl-C pour revenir à CP/M et SAVE 1 CONTROLZ.COM.

La commande CONTROLZ vous permettra donc de libérer éventuellement un ordinateur récepteur en envoyant le code Ctrl-Z sur la ligne.

### Remarque 3

La lecture/écriture de la carte série se fait de façon très simplifiée. Il n'y a en particulier pas de contrôle d'erreur de parité, de cadrage ou d'embouteillage. Ceci n'est pas très gênant car en pratique, cette utilisation est réservée à des transmissions filaires locales.

## Fonctionnement

Le programme source de SETSSC est documenté et ne pose pas de problème particulier. La majeure partie du code est le traitement de la ligne de commande. Des tables ont été créées pour transformer les paramètres spécifiés en valeurs de configuration des registres de l'ACIA. A noter que pour plus de commodité la vitesse de transmission a été codée d'ASCII en DCB plutôt qu'en binaire.
Le reste du code consiste à implanter les deux routines de lecture et écriture dans le BIOS dans la zone mémoire allouée aux Patches Slot 2 ( 0F280H ). Il faut ensuite entrer les adresses de ces routines dans la table des IOvecteurs aux emplacements réservés aux unités logiques UP2: et UR2:
Dernière précaution, lors du Coldstart, CP/M effectue une recherche des cartes présentes dans l'Apple et crée une table avec les types de ces cartes. Lors de chaque démarrage à chaud, (donc en particulier après chaque exécution de PIP) toutes les cartes sont réinitialisées. Ceci est très gênant car on perdrait ainsi la configuration définie par SETSSC pour revenir aux valeurs par défaut. Il faut donc "tuer" la carte SSC pour CP/M, en mettant un code 0 (Slot vide), pour que le Warmboot ne modifie rien.

Dans la deuxième partie de cet article, nous étudierons un protocole simple mais déjà très puissant. Deux programmes, sous forme de *commande* CP/M seront proposés, l'un émettant, l'autre recevant. Ces deux programmes sauront s'attendre si nécessaire, contrôler la réception des informations reçues, réémettre des informations mal transmises...

# Source 'SETSSC.SOURCE'

```
                TITLE   SETSSC
;
;===============================================
;
;  CONFIGURATION DE LA Super Serial Card
;       sous  CP/M 2.23
;
;  SETSSC Bxxxx,Dx,Sx,Px
;
;  Bxxxx        Vitesse en Bauds
;  Dx           Data bits, x=5 à 8
;  Sx           Stop bits, x=1 ou 2
;  Px           Parité, x=0 ou S   Space
;                       1 ou M   Mark
;                       E ou P   Paire
;                       O ou I   Impaire
;                       N        Non
;
;  SETSSC       configure avec les Switches
;
;  JF-R                 1986
;
;===============================================
;
BDOS    EQU     5       ;Entrée BDOS
BUFFER  EQU     80H     ;Buffer d'entrée
;
; Registres ACIA
; --------------
NSLOT   EQU     0E0A0H  ;Carte en slot 2
DIPSW1  EQU     NSLOT+1 ;Switches 1
DIPSW2  EQU     NSLOT+2 ;Switches 2
TDREG   EQU     NSLOT+8 ;Registre transmission
RDREG   EQU     NSLOT+8 ;Registre réception
STATUS  EQU     NSLOT+9 ;Registre d'état
COMMAND EQU     NSLOT+0AH ;Registre de commande
CONTROL EQU     NSLOT+0BH ;Registre de controle
;
; Adresses clavier
; ----------------
KBD     EQU     0E000H
KBDSTR  EQU     0E010H
;
; Adresses BIOS
; -------------
PADR    EQU     0F280H  ;Patches pour slot 2
IOVEC   EQU     0F380H  ;Table vecteurs IO
UR2     EQU     0CH     ;Offset User-Reader 2
UP2     EQU     10H     ;Offset User-Punch 2
CARDS   EQU     0F3B8H  ;Table cartes périphériques
                .PHASE  103H
                PAGE
                LD      IX,BUFFER
                LD      A,(IX+0)
                INC     IX
                OR      A       ;des paramètres ?
                JR      NZ,MODIF
                CALL    LEC_SWI ;récup. config. sw
                JP      CONFIG  ;config. ACIA et
;                                retour CCP
;
; Analyse de la ligne de commande
; -------------------------------
MODIF:          LD      HL,CARDS+2
                LD      A,(HL)
                OR      A       ;1ere fois ?
                JR      Z,MODIF1 ;Non
                CALL    LEC_SWI ;Oui, récup. des switches
                JR      LECTURE
;
MODIF1:         CALL    LEC_ACIA ;récup config ACIA
;
LECTURE:        LD      A,(IX+0) ;Ligne de commande
                INC     IX
                OR      A       ;ligne vide ?
                JP      Z,CONFIG
                CP      ' '
                JP      Z,LECTURE
                AND     0DFH    ;majusc. ou minusc.
                CP      'B'
                JP      Z,LEC_BAUD
                CP      'D'
                JP      Z,LEC_DATA
                CP      'S'
                JP      Z,LEC_STOP
                CP      'P'
                JP      Z,LEC_PARI
;
; Sortie sur erreur de commande
; -----------------------------
ERR_CMD:        LD      DE,ERR_MSG
                LD      C,9
                JP      BDOS    ;message et retour CCP
ERR_MSG         EQU     $
                DB      7,13,10
                ASC     'Fichtre, la commande !'
                DB      13,10,'$'
;
; Commande suivante (retour des routines LEC_xxx )
; ------------------
SUITE:          LD      A,(IX+0)
                INC     IX
                OR      A
                JP      Z,CONFIG
                CP      ','
                JR      NZ,ERR_CMD
```

## Comment faire ?

Vous avez la disquette Pom's

Les fichiers sont en format DOS 3.3. Vous les transférerez sur votre disquette CP/M à l'aide de "Universal File Conversion" ou APDOS.

Vous n'avez pas la disquette

Saisissez le dump avec l'utilitaire DDT à partir de l'adresse 0100.

Dans les deux cas

Utilisez la commande suivante pour configurer la carte :

SETSSC,Bxxxx,Dx,Sx,Px

| | |
|---|---|
| B = vitesse en bauds | 50 à 19200 |
| D = nombre de bits de données | 5 à 8 |
| S = nombre de bits de stop | 1 ou 2 |
| P = parité | 01PINSME0 (voir texte) |

Exemples

SETSSC B9600,PM
SETSSC S1,B300
SETSSC D8
SETSSC (sans paramètre = utilisation de la positions des switches)

```
                JR      LECTURE
                PAGE
;
; Routine de recherche d'une commande
; -----------------------------------
CHERCHE:        LD      A,(IX+0)
                INC     IX
                PUSH    HL
                CPDR
                POP     HL
                INC     HL
                ADD     HL,BC
                LD      A,(HL)
                RET     Z
                POP     HL
                JR      ERR_CMD
;
; Lecture Data bits
; -----------------
LEC_DATA        EQU     $
                LD      HL,DATA_T
                LD      BC,4
                CALL    CHERCHE
                LD      (DATA),A
                JP      SUITE
;
                DB      '5','6','7','8'
DATA_T          EQU     $-1
                DB      60H,40H,20H,0
;
; Lecture Stop bits
; -----------------
LEC_STOP        EQU     $
                LD      HL,STOP_T
                LD      BC,2
                CALL    CHERCHE
                LD      (STOP),A
                JP      SUITE
;
                DB      '1','2'
STOP_T          EQU     $-1
                DB      0,80H
;
; Lecture Parité
; --------------
LEC_PARI        EQU     $
                LD      HL,PARI_T
                LD      BC,16
                CALL    CHERCHE
                LD      (PARI),A
                JP      SUITE
;
                DB      'S','s','O','M','m','I'
                DB      'E','e','P','p','O','o'
                DB      'I','i','N','n'
PARI_T          EQU     $-1
                DB      0E0H,0E0H,0E0H,0A0H,0A0H,0A0H
                DB      60H,60H,60H,60H,20H,20H
                DB      20H,20H,0,0
;
; Conversion vitesse en DCB
; -------------------------
ACCU:           DW      0
CONV_BD:        LD      HL,ACCU+1
                XOR     A
                LD      (HL),A
                DEC     HL
                LD      (HL),A
;
DIGIT:          LD      A,(IX+0)
                OR      A
                RET     Z
                CP      ','
                RET     Z
                INC     IX
                AND     0FH
                RLD
                INC     HL
                RLD
                DEC     HL
                OR      A
                JR      Z,DIGIT
                LD      (CHIF5),A
                CP      1
                RET     Z
;
                POP     HL
                JP      ERR_CMD
;
; Lecture de la vitesse
; ---------------------
LEC_BAUD:       CALL    CONV_BD
                LD      IY,ACCU
                LD      HL,BAUD_T
                LD      B,15
LEC_B1:         DEC     HL
                LD      A,(HL)
                CP      (IY+1)
                DEC     HL
                JR      NZ,PAS_BON
                LD      A,(HL)
                CP      (IY+0)
                JR      Z,TROUVE
PAS_BON:        DJNZ    LEC_B1
;
                JP      ERR_CMD
;
TROUVE:         LD      DE,BAUD_T-32
                XOR     A
                SBC     HL,DE
                LD      A,L
                RRA
                LD      (BAUD),A
                CP      0FH     ;19200 bauds ?
                JP      NZ,SUITE
                LD      A,(CHIF5)
                CP      1
                JP      Z,SUITE
                JP      NZ,ERR_CMD
;
                DW      50H,75H,110H,135H,150H,300H
                DW      600H,1200H,1800H,2400H,3600H
                DW      4800H,7200H,9600H,9200H
BAUD_T          EQU     $
                PAGE
; Lecture des switches
; --------------------
LEC_SWI:        XOR     A
                LD      HL,DIPSW1
                RLD
                LD      (BAUD),A
                INC     HL
                LD      A,(HL)
                LD      B,A
                AND     80H
                LD      (STOP),A
                LD      A,40H
                AND     B
                SRL     A
                LD      (DATA),A
                LD      A,10H
                AND     B
                JR      Z,LEC_SWI1
                LD      A,30H
                AND     B
                RLCA
```

```
LEC_SWI1:       LD      (PARI),A
                RET
;
; Lecture de l'ACIA
; -----------------
LEC_ACIA:       LD      HL,COMMAND
                LD      A,(HL)
                AND     0E0H
                LD      (PARI),A
                INC     HL
                LD      B,(HL)
                LD      A,0FH
                AND     B
                LD      (BAUD),A
                LD      A,60H
                AND     B
                LD      (DATA),A
                LD      A,80H
                AND     B
                LD      (STOP),A
                RET
;
; Stockage des paramètres
; -----------------------
BAUD:           DS      1
DATA:           DS      1
STOP:           DS      1
PARI:           DS      1
CHIF5:          DS      1
;
; Configuration de l'ACIA
; -----------------------
CONFIG:         LD      HL,COMMAND
                LD      A,(PARI)
                OR      8       ;active RTS
                LD      (HL),A
                INC     HL
                LD      A,(BAUD)
                OR      10H
                LD      B,A
                LD      A,(DATA)
                OR      B
                LD      B,A
                LD      A,(STOP)
                OR      B
                LD      (HL),A
                LD      A,(RDREG)
;
; Mise en place des Drivers dans le BIOS
; --------------------------------------
                LD      HL,PUNCH
                LD      DE,PADR
                LD      BC,FIN-PUNCH
                LDIR            ;Transport du code
                LD      HL,PADR ;Adresse Punch
                LD      (IOVEC+UP2),HL
                LD      HL,PADR+READER-PUNCH
                LD      (IOVEC+UR2),HL
                LD      HL,CARDS+2      ;Slot 2
                XOR     A
                LD      (HL),A  ;On "tue" la carte
;
; Message Ok et retour CCP
; ------------------------
                LD      DE,OKMSG
                LD      C,9
                JP      BDOS
;
OKMSG:          DB      13,10
                ASC     'SSC  Ok...'
                DB      13,10,'$'
                PAGE
; Drivers d'entrée sortie série
```

## Dump hexadécimal de SETSSC.COM

```
0100  00 00 00 DD 21 80 00 DD 7E 00 DD 23 B7 20 06 CD
0110  64 02 C3 AD 02 21 BA F3 7E B7 28 05 CD 64 02 18
0120  03 CD 8A 02 DD 7E 00 DD 23 B7 CA AD 02 FE 20 CA
0130  24 01 E6 DF FE 42 CA 0C 02 FE 44 CA 8B 01 FE 53
0140  CA A2 01 FE 50 CA B5 01 11 50 01 0E 09 C3 05 00
0150  07 0D 0A 46 69 63 68 74 72 65 2C 20 6C 61 20 63
0160  6F 6D 6D 61 6E 64 65 20 21 0D 0A 24 DD 7E 00 DD
0170  23 B7 CA AD 02 FE 2C 20 CF 18 A9 DD 7E 00 DD 23
0180  E5 ED B9 E1 23 09 7E C8 E1 18 BD 21 9D 01 01 04
0190  00 CD 7B 01 32 A9 02 C3 6C 01 35 36 37 38 60 40
01A0  20 00 21 B2 01 01 02 00 CD 7B 01 32 AA 02 C3 6C
01B0  01 31 32 00 80 21 D3 01 01 10 00 CD 7B 01 32 AB
01C0  02 C3 6C 01 53 73 30 4D 6D 31 45 65 50 70 4F 6F
01D0  49 69 4E 6E E0 E0 E0 A0 A0 A0 60 60 60 60 20 20
01E0  20 20 00 00 00 00 21 E5 01 AF 77 2B 77 DD 7E 00
01F0  B7 C8 FE 2C C8 DD 23 E6 0F ED 6F 23 ED 6F 2B B7
0200  28 EB 32 AC 02 FE 01 C8 E1 C3 48 01 CD E6 01 FD
0210  21 E4 01 21 64 02 06 0F 2B 7E FD BE 01 2B 20 06
0220  7E FD BE 00 28 05 10 F0 C3 48 01 11 44 02 AF ED
0230  52 7D 1F 32 A8 02 FE 0F C2 6C 01 3A AC 02 FE 01
0240  CA 6C 01 C2 48 01 50 00 75 00 10 01 35 01 50 01
0250  00 03 00 06 00 12 00 18 00 24 00 36 00 48 00 72
0260  00 96 00 92 AF 21 A1 E0 ED 6F 32 A8 02 23 7E 47
0270  E6 80 32 AA 02 3E 40 A0 CB 3F 32 A9 02 3E 10 A0
0280  28 04 3E 30 A0 07 32 AB 02 C9 21 AA E0 7E E6 E0
0290  32 AB 02 23 46 3E 0F A0 32 A8 02 3E 60 A0 32 A9
02A0  02 3E 80 A0 32 AA 02 C9 E5 E5 E5 E5 E5 21 AA E0
02B0  3A AB 02 F6 08 77 23 3A A8 02 F6 10 47 3A A9 02
02C0  B0 47 3A AA 02 B0 77 3A A8 E0 21 FD 02 11 80 F2
02D0  01 3B 00 ED B0 21 80 F2 22 90 F3 21 93 F2 22 8C
02E0  F3 21 BA F3 AF 77 11 EE 02 0E 09 C3 05 00 0D 0A
02F0  53 53 43 20 20 4F 6B 2E 2E 2E 0D 0A 24 21 A9 E0
0300  CB 66 28 FC 21 A2 E0 CB 46 20 FC 21 A8 E0 71 C9
0310  21 A9 E0 CB 5E 28 03 2B 7E C9 23 CB C6 2B CB 5E
0320  20 0F 3A 00 E0 07 30 F6 CB 3F 23 CB 86 32 10 E0
0330  C9 2B 7E 23 23 CB 86 C9 00 00 00 00 00 00 00 00
```

```
; -------------------------------
PUNCH:          LD      HL,STATUS
W_TDR:          BIT     4,(HL)  ;TDR vide ?
                JR      Z,W_TDR
                LD      HL,DIPSW2
W_CTS:          BIT     0,(HL)  ;CTS ?
                JR      NZ,W_CTS
                LD      HL,TDREG
                LD      (HL),C  ;écriture
                RET
READER:         LD      HL,STATUS
                BIT     3,(HL)  ;RDR plein ?
                JR      Z,S_DTR
                DEC     HL
                LD      A,(HL)  ;lecture
                RET
S_DTR:          INC     HL
                SET     0,(HL)  ;active DTR
                DEC     HL
W_RDR:          BIT     3,(HL)  ;RDR plein ?
                JR      NZ,READER1
                LD      A,(KBD)
                RLCA
                JR      NC,W_RDR
                SRL     A
                INC     HL
                RES     0,(HL)
                LD      (KBDSTR),A
                RET
READER1:        DEC     HL
                LD      A,(HL)  ;lecture
                INC     HL
                INC     HL
                RES     0,(HL)  ;désactive DTR
                RET
FIN             EQU     $
                END
```

# Pom's

## Recueil n°1 (Pom's 1, 2, 3 & 4)



## Recueil n°2 (Pom's 5, 6, 7 & 8)



## Recueil n°3 (Pom's 9, 10, 11 & 12)



## Pom's n°13



## Pom's n°14

# SOMMAIRE DES NUMEROS 1 A 25



## Pom's n°15



## Pom's n°16



## Pom's n°17



## Pom's n°18



## Pom's n°19



## Pom's n°20



## Pom's n°21



## Pom's n°22



## Pom's n°23



## Pom's n°24



## Pom's n°25



A=Assembleur
B=Basic
P=Pascal
(B/A)=relatif au basic ou à l'assembleur
*-relatif au Macintosh*

Mac 128 512 Plus

# irT de acehîns

*Jean-Luc Bazanegue*

Le langage Basic, même aussi évolué que l'est celui de Microsoft pour le Macintosh, ne permet pas de tout faire (du moins si l'on veut conserver des temps d'exécution raisonnables). Le tri de chaînes de caractères est une de ces opérations souvent indispensables, mais qu'il vaut mieux confier à un langage certes primaire (mais ô! combien efficace !) : l'assembleur, quitte à appeler la routine ainsi écrite depuis un programme Basic ; c'est l'objet de cet article.

## Mode d'emploi

Vous pouvez fort bien avoir besoin d'une routine de tri pour vos programmes Basic, sans pour autant ressentir le besoin de savoir comment elle fonctionne ; c'est pourquoi nous passons directement au mode d'emploi, le Lecteur intéressé trouvera les détails techniques plus loin.

Deux méthodes existent pour l'installation de routines en assembleur dans un programme Basic (voir le numéro 21 de Pom's):

- routines sous la forme d'une 'Librairie' indépendante du programme Basic ;
- routines installées directement dans un tableau de variables entières et faisant partie intégrante du programme.

Comme il y a quelques différences quant à l'utilisation des deux méthodes, nous vous proposons un mode d'emploi en deux parties, la première concernant la

### Version 'Library'

Avant toute chose, il convient de signaler que le tri effectué par la routine pourrait être qualifié de 'tri indirect' puisqu'il est effectué sur un tableau de variables auxiliaire, à la manière de la routine écrite par Bernard Lambillon pour l'Apple //, et publiée dans le numéro 25 de Pom's.

Les chaînes à trier doivent se trouver dans un tableau de variables (alphanumériques, bien entendu) à une dimension (Chaine$(n) par exemple). Parallèlement, on devra avoir un tableau de variables entières à une dimension (comme Index%(n)), contenant dans chacun de ses éléments le numéro de la chaîne de caractères contenu par l'élément correspondant du tableau de variables alphanumériques (ouf!). Ce numéro sera compris entre 0 et le nombre de chaînes moins un. Rassurez-vous, ceci est beaucoup plus facile à faire qu'à dire :

On suppose un tableau de variables alphanumériques baptisé 'Chaine$', et constitué de 20 éléments (soit 0 à 19 avec, par défaut, 'Option Base 0'). Parallèlement, on a un tableau 'Index%', aussi de 20 éléments ; pour 'initialiser' ce dernier afin qu'il contienne les numéros des chaînes de caractères, il suffit de faire :

```
FOR I%=0 TO 19
Index%(I%)=I%
NEXT
```

Lorsque l'on sort de cette boucle (qui nous laisse tous pantois tant sa complexité est étonnante...), l'élément 0 contient 0, l'élément 1 contient 1... et le dernier contient 19.

On peut maintenant appeler la routine de tri ; en voici la syntaxe, à la manière du manuel du Basic Microsoft :

*TRILIB! premier élément du tableau de variables entières, premier élément du tableau de variables alphanumériques [,nombre de chaînes à trier]*

Le troisième argument, facultatif, correspond au nombre de chaînes à trier (toujours en partant de la première). Il doit être un entier (≤32767) et peut être passé à la routine de manière immédiate ('20'), sous la forme d'une expression ('Nombre%-10') ou d'une variable entière ('Nombre%'). Si cet argument est omis, toutes les chaînes sont triées. On pourrait donc avoir dans un programme un des appels suivants :

```
TriLib! Index%(0),Chaine$(0),20
TriLib! Index%(0),Chaine$(0),
Nombre%-10
TriLib! Index%(0),Chaine$(0),
  Nombre%
TriLib! Index%(0),Chaine$(0)
```

Au retour de la routine de tri, le tableau contenant les chaînes n'est pas modifié ; on trouve l'ordre des chaînes dans le tableau de variables entières. Ainsi, pour faire apparaître à l'écran le résultat du tri, on peut employer une boucle comme celle-ci :

```
FOR I%=0 TO 19
PRINT Chaine$(Index%(I%))
NEXT
```

Tout ceci est mis en évidence par le programme de démonstration 'TriLib.Démo', qui effectue le tri de 100 chaînes de 200 caractères placées dans un fichier à accès relatif.

### Protections

Afin d'éviter, autant que possible, les 'plantages' dus à des passages de paramètres incorrects, la routine en assembleur effectue quelques tests et, si un problème est rencontré, provoque l'affichage du message "Erreur de syntaxe" (ou "Syntax error", selon la version du Basic). Ceci apparaît dans les cas suivants :

- le premier argument ne correspond pas à un tableau de variables entières ;
- le tableau a plus d'une dimension ;
- le second argument ne correspond pas à un tableau de variables alphanumériques ;
- le troisième argument n'est pas un entier.

*Remarques : sans reprendre ce qui a été dit dans le numéro 21 de Pom's, il faut rappeler que la 'Library' doit se trouver de préférence sur la même disquette que le Basic et, sur un Mac Plus, au niveau 0 du HFS. De plus, il est indispensable d'initialiser la librairie avant le premier appel de la routine. Pour celle qui nous occupe aujourd'hui, il faut faire :*

```
LIBRARY "TriLib.Rscr"
```

### Version 'Tableau de variables'

Le fonctionnement de cette version, en ce qui concerne le tri, étant strictement identique, nous nous contenterons de citer ici les différences avec la version 'Library'.

Mises à part les petites manipulations liées à cette méthode et évoquées maintes fois dans des numéros précédents de Pom's (installation de la routine dans un tableau de variables, affectation de la variable du 'CALL' avec VARPTR...), il convient de signaler deux différences importantes. La première se rapporte à la syntaxe ; le premier argument doit ici être l'adresse du premier élément du tableau de variables entières (et non plus l'élément lui-même), soit :

```
VARPTR(Index%(0))
```

De même, le second argument doit être l'adresse du premier élément du tableau de variables alphanumériques :

```
VARPTR(Chaine$(0))
```

Enfin, la présence du troisième argument devient indispensable (voir les problèmes de décalages dans la pile, numéro 16 de Pom's).

L'appel de la routine avec cette méthode doit ressembler à :

```
A!=VARPTR(Routine%(0))
A! VARPTR(Index%(0)),
   VARPTR(Chaine$(0)),Nombre%
```

La seconde différence porte sur les protections : il n'y en a pas ! Ceci s'avère ne pas être très gênant si l'on prend quelques précautions lors de la mise au point du programme. Après, protections ou pas, le programme fonctionnera de la même façon.

Avant de passer à un autre point, nous aimerions ici réhabiliter le principe de l'installation d'une routine dans un tableau de variables, quelque peu dénigré ces derniers temps.

Si les 'Librairies' apportent des avantages reconnus, le tableau de variables a aussi des points forts :

- le programme Basic n'a pas le 'fil à la patte' supplémentaire (il a déjà l'interpréteur) que constitue la 'Library' (ce n'est jamais qu'un fichier) ;
- on a pas le 'temps mort' dû au chargement du code de la routine (on le charge de toute façon, mais cela passe inaperçu s'il est chargé en même temps que le programme Basic) ;
- si l'on doit passer à la routine de nombreux paramètres (ou peu de paramètres, mais souvent), ce passage se faisant plus directement par la pile, il est possible de gagner un temps appréciable sur l'exécution du programme ;
- enfin, nos Lecteurs qui ne possèdent pas le système de développement 68000 et n'ont pas la disquette Mac 26 ne peuvent pas utiliser la version 'Library', alors qu'il leur est possible d'employer la version 'Tableau' en recopiant simplement les "DATA" du programme de démonstration, et ce même s'ils disposent seulement de la version 1.0 du Basic (la 'Library' fonctionne avec les versions 2.0 et plus, binaires ou décimales).

*Note : avec la version 1.0, la syntaxe d'appel de la routine est :*

```
A!=VARPTR(Routine%(0))
CALL A!(VARPTR(Index%(0)),
   VARPTR(Chaine$(0)),Nombre%)
```

### Les limites

L'utilisation de la routine de tri n'apporte pas de contraites particulière pas rapport aux limites imposées par le Basic Microsoft, limites qui seront rarement atteintes : il est théoriquement possible de trier 32768 chaînes de 32767 caractères soit beaucoup plus d'octets que peut en adresser un MC68000, processeur qui n'est pourtant pas ridicule en la matière.

En résumé, on peut dire qu'en pratique la seule limite est la capacité mémoire de votre Macintosh.

## Méthode de Tri

Nous avons utilisé le tri par insertion, qui autorise des traitements d'assez courte durée de tableaux de chaînes de petite ou moyenne taille : un tableau de 100 chaînes de 255 caractères est trié en moins de trois secondes, ce qui est, compte tenu du système de comparaison des chaînes mis en œuvre (voir ci-dessous), tout à fait honorable.

## Méthode de comparaison

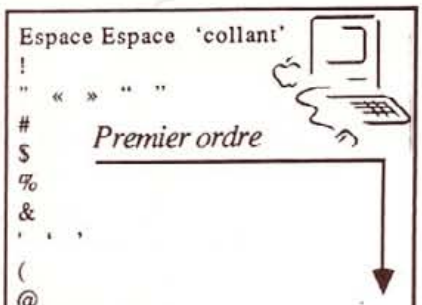
Généralement, les routines de tri comparent les chaînes de caractères en utilisant brutalement l'ordre des codes ASCII. Ainsi, la chaîne "Beaucoup" se retrouve placée avant "assez peu" puisque le code ASCII de "B" est 65 et le celui de "a" est 97. De même, la chaîne "œuvre" sera placée après "oxyde". La routine proposée par Apple dans un des 'Packages' du Macintosh utilise un mécanisme plus évolué qui permet d'obtenir des tris plus logiques.

Le tri s'effectue selon deux niveaux de priorité qui sont visualisés par le tableau joint à cet article. La priorité absolue est donné au premier ordre (de haut en bas sur notre tableau). Pour des caractères placés aux mêmes niveaux (sur la même ligne du tableau), le second ordre est employé (de gauche à droite). On retrouve un mécanisme équivalent pour les ligatures *Æ*, *æ*, *Œ* et *œ* (il y a des moments où l'on est vraiment content d'avoir l'accessoire "ad litteram").

La notion d'ordre prioritaire peut s'appliquer simultanément sur plusieurs caractères : la chaîne "acide" sera placée avant la chaîne "Adage" car la comparaison *A/a* correspond au second niveau de priorité, alors que *c/d* concerne le premier niveau.

Il est évident que cette méthode prend plus de temps que la comparaison pure et simple des codes ASCII, mais elle permet de ne pas se soucier de l'éventuelle présence de minuscules ou de caractères accentués. De plus, et si l'on fait abstraction du côté pratique, ce type de comparaison est intellectuellement plus satisfaisant que l'utilisation seule de l'ordre fixé par le code ASCII.

Néanmoins, si vous avez des notions d'assembleur et désirez accélérer le traitement, vous pouvez remplacer l'appel de la routine (clairement repéré car encadré par deux instructions 'MOVEM.L') par un branchement sur une routine de votre cru.




* *Pour le premier ordre, les caractères 'Æ', 'æ', 'Œ' et 'œ' sont considérés comme étant équivalents à 'A E', 'a e', 'O E' et 'o e'. Toutefois, pour le second ordre, 'Æ' est supérieur à 'A E', 'æ' est supérieur à 'a e', 'Œ' est supérieur à 'O E' et 'œ' est supérieur à 'o e'.*

***Ordres des caractères utilisés par la routine IUMagString pour comparer les chaînes***

## Programme TriLib.Démo'

### Démonstration de la routine 'TriLib' (version "Library" de la routine de tri)

*Note : sur un Mac 128Ko, ajouter 'CLEAR,26000,1024' en début de programme.*

```
ON ERROR GOTO Erreur
LIBRARY "TriLib.Rscr":' Ouverture de la 'Libra
   irie' contenant notre routine de tri.
DEFINT A-Z:DIM Index(99),Chaine$(99),Fenetres(
   7)
' Le tableau 'Index' contiendra les numéros d'or
   dre des chaînes (de 1 au nombre de chaînes).
' Le tableau 'Chaine$' recevra les 100 chaînes d
   e caractères.
' Le tableau 'Fenetre' n'a pas de rapport avec l
   a routine de Tri : il s'agit du tableau qui c
   ontiendra le code objet de la routine de ferm
   eture des fenêtres, publiée dans le numéro 19
    de Pom's. Les trois lignes suivantes concern
   ent aussi cette routine.
DATA &h42A7,&hA924,&h2E1F,&h6706,&h2F07,&hA916,
   &h60F2,&h4E75
FOR I=0 TO 7:READ Fenetres(I):NEXT
A!=VARPTR(Fenetres(0)):A!
' Ouverture de deux fenêtres. La fenêtre 1 sera
   utilisée pour les messages, alors que la fenê
   tre 2, beaucoup plus grande, autorisera la vi
   sualisation du début des chaînes de caractère
   s.
WINDOW 2,"",(8,46)-(503,337),3:TEXTSIZE 12:TE
   XTFONT 0:WINDOW 1,"",(8,24)-(503,41),3:TEX
   TSIZE 12:TEXTFONT 0
' On 'construit' 100 chaînes de caractères avec
   des caractères choisis de façon aléatoire ent
   re 'A' et 'Z'. Une fois qu'une chaîne contien
   t ses 200 caractères, on l'affiche.
BEEP:PRINT "Constitution aléatoire des chaînes
    (100 fois 200 caractères) en cours…";
WINDOW OUTPUT 2:FOR I=0 TO 99:RANDOMIZE TI
   MER:FOR J=1 TO 200:Chaine$(I)=Chaine$(I)+CH
   R$(65+INT(RND*26)):NEXT:PRINT "Chaîne" I+1
   ": " Chaine$(I):NEXT
' Les chaînes de caractères sont enregistrées no
   n triées dans un fichier à accès relatif, com
   portant 100 enregistrements de 200 caractères
   .
WINDOW OUTPUT 1:CLS:BEEP:PRINT "Enregistrem
   ent des chaînes non triées dans le fichier "C
   haînes"…";
OPEN"R",1,"Chaînes",200:FIELD 1,200 AS Tampon$
FOR I=0 TO 99:LSET Tampon$=Chaine$(I):PUT 1,I+
   1:NEXT:CLOSE
' Les chaînes sont maintenant relues (on considè
   re qu'elles ont été créées par un autre progr
   amme ou sont déjà existante) et, en même temp
   s, on initialise le tableau d'index.
CLS:BEEP:PRINT "Lecture en mémoire des chaînes
    non triées (depuis le fichier)…";
OPEN"R",1,"Chaînes",200:FIELD 1,200 AS Tampon$
FOR I=0 TO 99:GET 1,I+1:Chaine$(I)=Tampon$:Ind
   ex(I)=I:NEXT:CLOSE
' Appel de la routine de Tri. Comme nous avons l
   'intention de trier toutes les chaînes, il es
   t possible d'omettre le troisième argument (n
   ombre de chaînes à trier).
CLS:BEEP:PRINT "Tri en mémoire des 100 chaînes
    de caractères…";
TriLib! Index(0),Chaine$(0)
' Les chaînes sont triées. Rappelons que seul le
    contenu du tableau de variables entières 'In
   dice' reflète le travail effectué par la rout
   ine : il contient l'ordre des chaînes.
CLS:BEEP:PRINT "Affichage et enregistrement de
   s chaînes triées…";
WINDOW OUTPUT 2:CLS:OPEN"R",1,"Chaînes",200:
   FIELD 1,200 AS Tampon$:FOR I=0 TO 99:LSET
   Tampon$=Chaine$(Index(I)):PUT 1,I+1:PRINT Ch
   aine$(Index(I)):NEXT:CLOSE
' Le fichier 'Chaînes' contient maintenant les 1
   00 chaînes de caractères dans l'ordre.
WINDOW OUTPUT 1:CLS:BEEP:PRINT "Traitement
   terminé, appuyez sur une touche pour sortir."
   ;
WHILE INKEY$="":WEND:END
' Utilisé en cas d'erreur (disquette saturée, pa
   r exemple).
Erreur:RESUME Erreur2
Erreur2:WINDOW OUTPUT 1:CLS:BEEP:PRINT "Err
   eur ! Impossible de continuer…";
WHILE INKEY$="":WEND:END
```

## Programme 'Tri.Démo'

### Démonstration de la routine 'Tri' (version "tableau" de la routine de tri)

*Note : sur un Mac 128Ko, ajouter 'CLEAR,26000,1024' en début de programme.*

```
ON ERROR GOTO Erreur:DEFINT A-Z:DIM Index(99),Chaine$
   (99),Fenetres(7),Routine(100):' Le tableau 'Routine' re
   cevra le code objet de la routine de tri.
DATA &h42A7,&hA924,&h2E1F,&h6706,&h2F07,&hA916,&h60F2,&h
   4E75
' Code objet de la routine de tri.
DATA &h4E56,&hFFF2,&h206E,&hE,&h226E,&hA,&h302E,&h8,&h53
   40,&h3D40,&hFFFE,&h7201,&h3401,&hD442,&h3630,&h2000,&h5
   98F,&h3F03,&h6168,&h3D5F
DATA &hFFF6,&h2D5F,&hFFF2,&h3802,&h598F,&h3F30,&h40FE,&h
   6156,&h3D5F,&hFFFC,&h2D5F,&hFFF8,&h48E7,&hF0C0,&h558F,&
   h2F2E,&hFFF2,&h2F2E,&hFFF8,&h3F2E,&hFFF6,&h3F2E,&hFFFC,
   &h3F3C,&hA,&hA9ED,&h4A5F,&h4CDF,&h30F,&h6B06
DATA &h3183,&h4000,&h6016,&h31B0,&h40FE,&h4000,&hC44,&h2
   ,&h6606,&h3183,&h40FE,&h6004,&h5544,&h60B0,&hB26E,&hFFF
   E,&h6704,&h5241,&h608E,&h4E5E,&h4E75,&h4E56,&h0,&h48E7,
   &hC000,&h302E,&h8,&hC0FC,&h5,&h7200
DATA &h1231,&h2,&hE189,&h1231,&h3,&hE189,&h1231,&h4,&h2D
   41,&hA,&h1231,&h0,&hE149,&h1231,&h1,&h3D41,&h8,&h4CDF,&
   h3,&h4E5E,&h4E75
FOR I=0 TO 7:READ Fenetres(I):NEXT
' Installation de la routine de tri dans le tableau de va
   riables entières 'Routine'
FOR I=0 TO 100:READ Routine(I):NEXT
A!=VARPTR(Fenetres(0)):A!:WINDOW 2,"",(8,46)-(503,337),
   3:TEXTSIZE 12:TEXTFONT 0:WINDOW 1,"",(8,24)-(503,4
   1),3:TEXTSIZE 12:TEXTFONT 0:BEEP:PRINT "Constituti
   on aléatoire des chaînes (100 fois 200 caractères) en c
   ours…";:WINDOW OUTPUT 2:FOR I=0 TO 99
RANDOMIZE TIMER:FOR J=1 TO 200:Chaine$(I)=Chaine$(I)+
   CHR$(65+INT(RND*26)):NEXT:PRINT "Chaîne" I+1 ": " C
   haine$(I):NEXT:WINDOW OUTPUT 1:CLS:BEEP:PRINT "En
   registrement des chaînes non triées dans le fichier "Ch
   aînes"…";:OPEN"R",1,"Chaînes",200
```

```
FIELD 1,200 AS Tampon$:FOR I=0 TO 99:LSET Tampon$=Chai
  ne$(I):PUT 1,I+1:NEXT:CLOSE:CLS:BEEP:PRINT "Lectur
  e en mémoire des chaines non triées (depuis le fichier)
  …";:OPEN"R",1,"Chaines",200:FIELD 1,200 AS Tampon$:F
  OR I=0 TO 99:GET 1,I+1:Chaine$(I)=Tampon$
Index(I)=I:NEXT:CLOSE:CLS:BEEP:PRINT "Tri en mémoire
  des 100 chaines de caractères…";
' Appel de la routine de tri. Attention : avec cette vers
  ion, tous les arguments sont requis. De plus, aucun tes
  t n'est effectué quant à la validité des arguments.
Tri!=VARPTR(Routine(0)):Tri! VARPTR(Index(0)),VARPTR(C
  haine$(0)),100
```

```
CLS:BEEP:PRINT "Affichage et enregistreme
  nt des chaines triées…";:WINDOW OUTPUT
  2:CLS:OPEN"R",1,"Chaines",200:FIELD 1,2
  00 AS Tampon$:FOR I=0 TO 99:LSET Tampon
  $=Chaine$(Index(I)):PUT 1,I+1:PRINT Chai
  ne$(Index(I)):NEXT:CLOSE
WINDOW OUTPUT 1:CLS:BEEP:PRINT "Traitement terminé, a
  ppuyez sur une touche pour sortir.";
WHILE INKEY$="":WEND:END
Erreur:RESUME Erreur2
Erreur2:WINDOW OUTPUT 1:CLS:BEEP:PRINT "Erreur ! Imp
  ossible de continuer…";:WHILE INKEY$="":WEND:END
```

# Source 'TriLib.Asm' (version 'Librairie')

```
.Trap       _Pack6      $A9ED   ; Utilisé pour l'appel du 'Package' numéro 6, baptisé 'International Utilities Package'. Il est
                                ; situé dans le fichier 'système' (ressource INTL 6).
GetNextLibArg       EQU $2A     ; Cette routine est utilisée pour saisir un paramètre passé à la librairie depuis le Basic.
BasicError          EQU $42     ; Routine autorisant l'affichage de messages d'erreur du Basic.
IUMagString         EQU 10      ; Ce 'sélecteur' correspond à la routine de comparaison de chaînes située dans Pack 6.
VariableChaine      EQU 2       ; Valeur retournée dans D0 par 'GetNextLibArg' pour une variable de type 'chaîne'.
VariableEntiere     EQU 3       ; Valeur retournée dans D0 par 'GetNextLibArg' pour une variable de type 'entier'.
UneDimension        EQU 1       ; Pour s'assurer que le tableau de variables entières a bien une seule dimension.

; Saisies et vérifications des paramètres passés à la routine de tri depuis le programme Basic. Nous devons trouver, dans l'ordre, la
; première variable d'un tableau de variable entières (une seule dimension), la première variable d'un tableau de variables 'chaînes' et, enfin
; une variable entière, ou rien si le troisième argument (facultatif) est omis.
        JSR     GetNextLibArg(A5)       ; Saisie du premier paramètre. S'il s'agit d'une variable entière, 'GetNextLibAr' retourne
        CMPI    #VariableEntiere,D0     ; 3 dans les 16 bits de poids faible de D0. Si la valeur est différente (autre type de
        BNE.S   @3                      ; variable), on provoque l'affichage d'un message d'erreur. Cette comparaison nous permet
                                        ; aussi de détecter une éventuelle variable entière 'temporaire' (A%*10, INT(T%(2)/3)...) car,
                                        ; dans ce cas, le bit 15 de D0 est mis à 1. On obtiendrait donc $8003 au lieu de 3.
        CMPI.B  #UneDimension,-3(A2)    ; Au retour de la routine 'GetNextLibArg', et pour une variable entière, le registre
        BNE.S   @3                      ; d'adresse A2 pointe sur le contenu de cette variable. Pour s'assurer que le tableau de
                                        ; variable entières est bien à une seule dimension, on vérifie le contenu de l'octet situé trois
                                        ; octets avant l'adresse pointée par A2. Cet octet contient le nombre de dimensions du
                                        ; tableau (voir l'article 'Call : un exemple d'application', numéro 16 de Pom's).
        MOVE.L  A2,D6                   ; On sauvegarde l'adresse du premier élément du tableau de variables dans D6.
        JSR     GetNextLibArg(A5)       ; Saisie du second paramètre. S'il s'agit d'une variable 'chaîne', 'GetNextLibAr' retourne
        CMPI    #VariableChaine,D0      ; 2 dans les 16 bits de poids faible de D0. Si la valeur est différente, on provoque
        BNE.S   @3                      ; l'affichage d'un message d'erreur.
        MOVE.L  A2,D5                   ; Pour une variable 'chaîne', 'GetNextLibArg' retourne l'adresse du descripteur de la chaîne
                                        ; dans A2. Dans ce descripteur, on trouve dans l'ordre, la longueur de la chaîne sur deux
                                        ; octets suivie de son adresse sur 3 octets (24 bits). On sauvegarde l'adresse du premier
                                        ; descripteur dans D5.
        JSR     GetNextLibArg(A5)       ; Saisie (ou tentative) du troisième paramètre (qui doit être une variable entière).
        MOVEA.L D6,A0                   ; On place l'adresse du premier élément du tableau de variables entières dans le registre A0.
                                        ; On n'utilisera pas ce registre tout de suite, mais effectuer cette opération maintenant nous
                                        ; évitera de le faire deux fois, plus tard.
        TST     D0                      ; On teste le contenu de D0. S'il est nul, il n'y a plus de paramètre dans la liste passée
        BEQ.S   @1                      ; depuis le Basic. On se dirige vers l'étiquette @1 pour prendre le nombre par défaut.
        ANDI    #$7FFF,D0               ; On force le bit 15 de D0 à 0 car on peut ici avoir une variable entière temporaire (une
                                        ; expression comme "NombreChaine-10", par exemple).
        CMPI    #VariableEntiere,D0     ; On s'assure que le paramètre passé est bien un entier.
        BNE.S   @3                      ; Sinon, message d'erreur.
        MOVE    (A2),D0                 ; Comme vu précédemment, A2 pointe sur le contenu de la variable entière. On place
        BRA.S   @2                      ; cette valeur (nombre de chaînes à trier) dans D0 et on se dirige vers le tri.
@1      MOVE    -2(A0),D0               ; Si nous n'avions pas de troisième paramètre, il faut trier toutes les chaînes de
        BRA.S   @2                      ; caractères. Pour en connaître le nombre, on récupère le nombre de variables entières
                                        ; contenues dans le tableau. Ce nombre se trouve dans les deux octets précédents le premier
                                        ; élément (voir le numéro 16 de Pom's). La valeur trouvée est placée dans D0 et on se dirige
                                        ; vers la routine de tri.

; La routine ' BasicError' provoque l'affichage d'un des messages d'erreur du Basic Microsoft. L'argument doit être placé dans le
@3      MOVEQ   #2,D2                   ; registre de donné D2. 2 correspond à "SYNTAX ERROR" (ou son équivalent français).
        JSR     BasicError(A5)          ; Le retour à l'interpréteur est aussi assuré par 'BasicError'.

; Routine de tri. La méthode utilisée est le tri par insertion.
Tampon          Set     -14         ; On prévoie un tampon de 14 octets au sommet de la pile dans lequel nous placerons :
AdrChaineExt    Set     Tampon      ; • l'adresse -4 octets- de la première chaîne (chaîne extraite) ;
LongChaineExt   Set     -10         ; • la longueur -2 octets- de la première chaîne ;
AdrChaineComp   Set     -8          ; • l'adresse -4 octets- de la seconde chaîne (chaîne à comparer) ;
```

```
LongChaineComp      Set  -4          ; • la longueur -2 octets- de la seconde chaîne ;
NombreElements      Set  -2          ; • le nombre -2 octets- de chaînes à trier (moins un).
@2  LINK    A6,#Tampon               ; Création du tampon de 14 octets au sommet de la pile.
    MOVEA.L D5,A1                    ; L'adresse du descripteur de la première chaîne de caractères est placée dans le registre A1.
    SUBQ    #1,D0                    ; le registre D0, qui contient le nombre de chaînes à trier, est décrémenté de 1. On a donc
                                     ; dans D0 le nombre moins 1, ce qui convient mieux à une routine en assembleur (premier = 0,
                                     ; dernier = N-1).
    MOVE    D0,NombreElements(A6)    ; On sauvegarde le nombre dans le tampon au sommet de la pile.
    MOVEQ   #1,D1                    ; Le registre D1 sert de 'compteur d'extraction'. On y place la valeur 1, ce qui correspond à la
                                     ; seconde chaîne à trier.
BouclePrincipale
    MOVE    D1,D2                    ; Une copie du compteur est placée dans le registre D2.
    ADD     D2,D2                    ; Le contenu du registre D2 est multiplié par 2. Ceci nous donne le décalage entre le début du
                                     ; tableau de variables entières et la variable entière contenant l'index de la chaîne à extraire
                                     ; (rappelons que les index doivent être placés dans le tableau de variables entières par le
                                     ; programme Basic. Les index constituent les 'numéros d'ordre' des chaînes).
    MOVE    0(A0,D2),D3              ; L'adresse du tableau de variables entières (A0) plus le décalage par rapport au début du
                                     ; tableau (D2) nous donne l'adresse de l'index, que l'on place dans le registre D3.
    SUBQ.L  #4,SP                    ; On réserve une place de 4 octets au sommet de la pile, pour le retour par la routine
    MOVE    D3,-(SP)                 ; 'AdresseEtLongueur' de l'adresse de la chaîne extraite. Ensuite, l'index de la chaîne est
    BSR.S   AdresseEtLongueur        ; empilé. La longueur de la chaîne sera retournée dans la pile à l'endroit où est placé l'index.
    MOVE    (SP)+,LongChaineExt(A6)  ; La longueur de la chaîne est placée dans le tampon au sommet de la pile.
    MOVE.L  (SP)+,AdrChaineExt(A6)   ; L'adresse de la chaîne est aussi placée dans le tampon au sommet de la pile.
    MOVE    D2,D4                    ; Une copie du décalage est placée dans le registre D4.
BoucleSecondaire
    SUBQ.L  #4,SP                    ; Réserve une place pour le retour de l'adresse de la chaîne à comparer.
    MOVE    -2(A0,D4),-(SP)          ; A0+D4-2 nous donne l'adresse de l'index de la chaîne à comparer (au 'premier tour' de la
    BSR.S   AdresseEtLongueur        ; boucle secondaire, il s'agit de l'index précédent celui de la chaîne extraite). Cet index est
                                     ; empilé et on appelle la routine 'AdresseEtLongueur'.
    MOVE    (SP)+,LongChaineComp(A6) ; La longueur de la chaîne est placée dans le tampon au sommet de la pile.
    MOVE.L  (SP)+,AdrChaineComp(A6)  ; L'adresse de la chaîne est aussi placée dans le tampon au sommet de la pile.

    MOVEM.L A0-A1/D0-D3,-(SP)        ; Les registres qui sont modifiés par la routine 'Pack6' sont sauvegardés au sommet de la pile.
    SUBQ.L  #2,SP                    ; On prévoie un emplacement de 2 octets pour le retour du résultat de la comparaison.
    MOVE.L  AdrChaineExt(A6),-(SP)   ; L'adresse de la chaîne extraire (Chaîne A) est empilée.
    MOVE.L  AdrChaineComp(A6),-(SP)  ; L'adresse de la chaîne à comparer (Chaîne B) est empilée.
    MOVE    LongChaineExt(A6),-(SP)  ; La longueur de la chaîne extraire (Chaîne A) est empilée.
    MOVE    LongChaineComp(A6),-(SP) ; La longueur de la chaîne à comparer (Chaîne B) est empilée.
    MOVE    #IUMagString,-(SP)       ; On empile enfin le 'sélecteur de routine' correspondant à la routine de comparaison.
    _Pack6                           ; Appel de la routine 'Pack6'. Au retour, le résultat situé au sommet de la pile est égal à -1 si la
    TST     (SP)+                    ; chaîne A est inférieur à la chaîne B, 0 si les deux chaînes sont égales, 1 si la chaîne A est
                                     ; supérieure à la chaîne B. TST (SP)+ positionne les bits Z (zéro) et N (négatif) du registre
                                     ; d'état en fonction du résultat obtenu.
    MOVEM.L (SP)+,A0-A1/D0-D3        ; Les registres retrouvent leur contenu initial (comme avant l'appel de la routine 'Pack6').
    BMI.S   Inferieure               ; Vers l'étiquette 'inferieure' si la chaîne extraite est inférieure à la chaîne à comparer.
    MOVE    D3,0(A0,D4)              ; Si la chaîne extraite est supérieure ou égale à la chaîne comparée, on insère l'index de la
    BRA.S   SortieBoucleSecondaire   ; chaîne extraite dans le tableau de variables entières, et dans l'élément suivant celui qui
                                     ; contient l'index de la chaîne comparée. On sort ensuite de la boucle secondaire.
Inferieure
    MOVE    -2(A0,D4),0(A0,D4)       ; Le contenu de l'élément du tableau correspondant à l'index de la chaîne comparée est placé
                                     ; dans l'élément suivant (décalage vers le bas).
    CMPI    #2,D4                    ; Si D4 contient 2, la chaîne comparée était la première chaîne (puisqu'on utilise -2(A0,D4)).
    BNE.S   @1                       ; Si ce n'est pas le cas, branchement sur l'étiquette @1.
    MOVE    D3,-2(A0,D4)             ; L'index de la chaîne extraite est mis à la place de l'index de la chaîne comparée. Cette petite
    BRA.S   SortieBoucleSecondaire   ; manipulation supplémentaire autorise l'emploi du premier élément de chaque tableau. On sort
                                     ; ensuite de la boucle secondaire.
@1  SUBQ    #2,D4                    ; Le décalage depuis la base du tableau de variables entières est diminué de deux pour
    BRA.S   BoucleSecondaire         ; 'pointer' sur l'élément contenant l'index précédent celui de la chaîne qui vient d'être
                                     ; comparée.

SortieBoucleSecondaire
    CMP     NombreElements(A6),D1    ; On compare le compteur (D1) au nombre de chaînes à comparer. Si les valeurs sont égales,
    BEQ.S   Fin                      ; le traitement est terminé, retour au Basic.
    ADDQ    #1,D1                    ; Sinon, on incrémente le compteur de 1.
    BRA.S   BouclePrincipale         ; On va extraire la chaîne suivante.

Fin UNLK    A6                       ; Abandon du tampon au sommet de la pile.
    MOVEQ   #0,D0                    ; Pour signaler à l'interpréteur Basic qu'il n'y a pas d'erreur.
    RTS                              ; Retour au Basic.

; Cette routine retourne au sommet de la pile l'adresse et la longueur de la chaîne dont l'index a été empilé par le programme appelant.
Index       Set  8                   ; Décalage entre l'adresse pointée par la registre A6 et l'index empilé par le programme.
Longueur    Set  8                   ; Décalage entre l'adresse pointée par A6 et l'endroit dans la pile où sera placée la longueur.
```

```
Adresse         Set     10              ; Décalage entre l'adresse pointée par A6 et l'endroit où sera placée l'adresse.
AdresseEtLongueur
        LINK    A6,#0
        MOVEM.L D0-D1,-(SP)             ; Sauvegarde les registres dont le contenu est modifié par cette routine.
        MOVE    Index(A6),D0            ; L'index de la chaîne est placé dans D0.
        MULU    #5,D0                   ; On multiplie l'index par 5 afin d'obtenir le décalage par rapport au descripteur de la première
                                        ; chaîne à trier (voir le schéma représentant les descripteurs de chaînes et leurs contenus).
        MOVEQ   #0,D1                   ; Les 32 bits du registre D1 sont mis à zéro.
        MOVE.B  2(A1,D0),D1             ; Les bits 16 à 23 de l'adresse de la chaîne sont placés dans les bits 0 à 7 du registre D1.
        LSL.L   #8,D1                   ; On décale le contenu du registre D1 de 8 bits vers la gauche.
        MOVE.B  3(A1,D0),D1             ; Les bits 8 à 15 de l'adresse de la chaîne sont placés dans les bits 0 à 7 du registre D1.
        LSL.L   #8,D1                   ; On décale le contenu du registre D1 de 8 bits vers la gauche.
        MOVE.B  4(A1,D0),D1             ; Les bits 0 à 7 de l'adresse de la chaîne sont placés dans les bits 0 à 7 du registre D1.
        MOVE.L  D1,Adresse(A6)          ; L'adresse 'reconstituée' de la chaîne est placée dans la pile.
        MOVE.B  0(A1,D0),D1             ; Les bits 8 à 15 de la longueur de la chaîne sont placés dans les bits 0 à 7 du registre D1.
        LSL     #8,D1                   ; On décale le contenu du registre D1 de 8 bits vers la gauche.
        MOVE.B  1(A1,D0),D1             ; Les bits 0 à 7 de la longueur de la chaîne sont placés dans les bits 0 à 7 du registre D1.
        MOVE    D1,Longueur(A6)         ; La longueur de la chaîne est placée dans la pile.
                                        ; Toutes ces manipulations (transferts d'octets et rotations) sont rendues nécessaires par le
                                        ; fait que les descripteurs de chaînes sont alignés seulement une fois sur deux sur des
                                        ; adresses paires ; il n'est donc ici pas question d'effectuer des opérations sur des mots ou
                                        ; des longs mots.
        MOVEM.L (SP)+,D0-D1             ; Le contenu des registres D0 et D1 est restitué.
        UNLK    A6                      ; Retour au programme appelant.
        RTS
```

## Fichier 'TriLib.Job'

```
Asm     LIBInit.Asm Exec        Edit
Asm     TriLib.Asm  Exec        Edit
Link    TriLib.Link Exec        Edit
RmakerTriLib.R      Finder      Edit
```

## Source 'LIBinit.Asm'

```
LIBVER_Result   EQU     6

        CLR     LIBVER_Result(A0)
        MOVEQ   #0,D0
        RTS
```

## Fichier 'TriLib.R'

```
TriLib.Rscr
BLIB
TYPE CODE = GNRL
LIBinit,1
.R
TriLib CODE 1

TYPE CODE = GNRL
TriLib,2
.R
TriLib CODE 2
```

## Fichier 'TriLib.Link'

```
/ OUTPUT TriLib
LIBinit.Rel
<
TriLib.Rel
$
```

## *Source 'Tri.Asm'*
### *(version 'tableau de variables')*

```
                .Trap       _Pack6  $A9ED
                IUMagString EQU     10

                Tampon      Set     -14
                AChEx       Set     Tampon
                LChEx       Set     -10
                AChC        Set     -8
                LChC        Set     -4
                NbreElem    Set     -2
                Nombre      Set     8
                VarptrChaines Set   10
                VarptrEntiers Set   14

4E56 FFF2       LINK        A6,#Tampon
206E 000E       MOVEA.L     14(A6),A0
226E 000A       MOVEA.L     10(A6),A1
302E 0008       MOVE        8(A6),D0
5340            SUBQ        #1,D0
3D40 FFFE       MOVE        D0,NbreElem(A6)
7201            MOVEQ       #1,D1

                BouclePrincipale
3401            MOVE        D1,D2
D442            ADD         D2,D2
3630 2000       MOVE        0(A0,D2),D3
598F            SUBQ.L      #4,SP
3F03            MOVE        D3,-(SP)
6158            BSR.S       AdresseLongueur
2D5F FFF6       MOVE        (SP)+,LChEx(A6)
2D5F FFF2       MOVE.L      (SP)+,AChEx(A6)
3802            MOVE        D2,D4

                BoucleSecondaire
598F            SUBQ.L      #4,SP
3F30 40FE       MOVE        -2(A0,D4),-(SP)
6156            BSR.S       AdresseLongueur
2D5F FFFC       MOVE        (SP)+,LChC(A6)
2D5F FFF8       MOVE.L      (SP)+,AChC(A6)
48E7 F0C0       MOVEM.L     A0-A1/D0-D3,-(SP)
598F            SUBQ.L      #2,SP
2F2E FFF2       MOVE.L      AChEx(A6),-(SP)
2F2E FFF8       MOVE.L      AChC(A6),-(SP)
3F2E FFF6       MOVE        LChEx(A6),-(SP)
3F2E FFFC       MOVE        LChC(A6),-(SP)
3F3C 000A       MOVE        #IUMagString,-(SP)
A9ED            _Pack6
4A5F            TST         (SP)+
4CDF 030F       MOVEM.L     (SP)+,A0-A1/D0-D3
6B06            BMI.S       Interieure
3183 4000       MOVE        D3,0(A0,D4)
6016            BRA.S       SortieBoucle

                Interieure
31B0 40FE 4000  MOVE        -2(A0,D4),0(A0,D4)
0C44 0002       CMPI        #2,D4
6606            BNE.S       @1
3183 40FE       MOVE        D3,-2(A0,D4)
6004            BRA.S       SortieBoucle
5544            @1  SUBQ    #2,D4
60B0            BRA.S       BoucleSecondaire

                SortieBoucle
B26E FFFE       CMP         NbreElem(A6),D1
6704            BEQ.S       Fin
5241            ADDQ        #1,D1
60BE            BRA.S       BouclePrincipale

4E5E            Fin UNLK    A6
4E75            RTS
```

```
                Index       Set 8
                Longueur    Set 8
                Adresse     Set 10
                AdresseLongueur

4E56 0000       LINK        A6,#0
48E7 C000       MOVEM.L     D0-D1,-(SP)
302E 0008       MOVE        Index(A6),D0
C0FC 0005       MULU        #5,D0
7200            MOVEQ       #0,D1
1231 0002       MOVE.B      2(A1,D0),D1
E189            LSL.L       #8,D1
1231 0003       MOVE.B      3(A1,D0),D1
E189            LSL.L       #8,D1
1231 0004       MOVE.B      4(A1,D0),D1
2D41 000A       MOVE.L      D1,Adresse(A6)
1231 0000       MOVE.B      0(A1,D0),D1
E149            LSL         #8,D1
1231 0001       MOVE.B      1(A1,D0),D1
3D41 0008       MOVE        D1,Longueur(A6)
4CDF 0003       MOVEM.L     (SP)+,D0-D1
4E5E            UNLK        A6
4E75            RTS
```

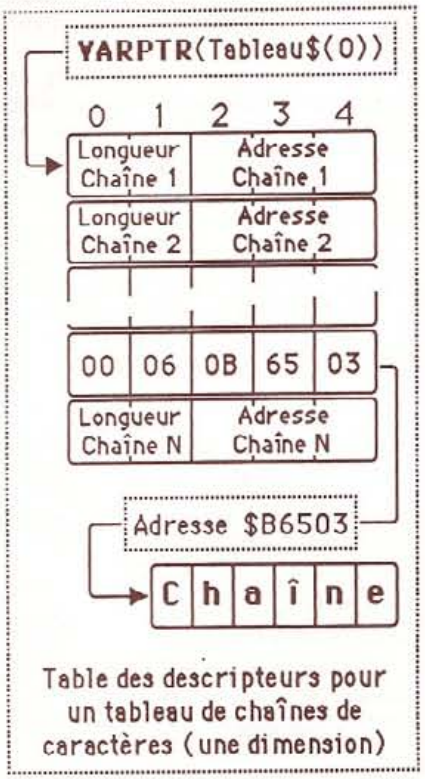



Table des descripteurs pour un tableau de chaînes de caractères (une dimension)



# 68000's

Julien Thomas

Quel que soit le langage que l'on utilise, il peut être intéressant d'avoir un minimum de connaissances en assembleur, ne serait-ce que pour écrire des petites routines destinées à pallier certaines déficiences du langage (traitement trop gourmand en temps ou simplement impossible).

Seulement, le 68000 étant un microprocesseur puissant, il offre un jeu d'instructions étendu, autorise souvent plusieurs syntaxes par instruction, chacune d'entre-elles offrant généralement de nombreux modes d'adressage possibles. Si l'on ajoute à cela la taille des opérandes (octets, mots ou longs mots) qui varie en fonction des instructions et des syntaxes, l'effet sur le contenu du registre d'état, plus quelques cas particuliers, on comprend qu'il n'est pas facile de mémoriser ce qu'il est possible de faire avec le 68000, ou plutôt ce qui ne l'est pas (c'est ce dernier point qui, souvent, nous oblige à faire un réassemblage).

Il est bien sûr envisageable de consulter un manuel de référence du 68000 lorsque cela se révèle nécessaire, mais l'expérience montre que cette méthode n'est pas très pratique, l'information recherchée étant généralement noyée dans d'autres (qui pratique l'assembleur et n'a jamais confondu les modes d'adressage autorisés pour l'opérande source, et ceux autorisés pour l'opérande destination ?).

D'où l'idée d'écrire un accessoire de bureau qui permettrait une consultation simple et immédiate lorsqu'une vérification s'impose.

## Utilisation

Nous avons voulu cet accessoire aussi pratique que possible ; pour cette raison, on peut le commander entièrement avec la souris bien sûr, mais aussi avec le clavier.

### Changement d'instruction

**Avec la souris** en cliquant dans le tableau des instructions (l'opération n'est prise en compte que lorsque le bouton est relaché).

**Avec le clavier :**

- la touche *tabulation* déplace la sélection d'une position vers la droite ;
- la touche *'back space'* déplace la sélection vers la gauche ;
- un *retour chariot* déplace la sélection vers le bas ;
- un *retour chariot* accompagné de la touche *majuscule* déplace la sélection vers le haut.

Dans tous les cas, un déplacement hors des limites sélectionne l'instruction située le long du bord opposé du tableau.

*Ces déplacements peuvent aussi être obtenus avec les touches fléchées du Macintosh Plus.*

La sélection d'une instruction entraîne :

- l'affichage de la syntaxe (ou des syntaxes, puisqu'il peut en exister jusqu'à trois) ;
- l'affichage de l'effet de l'instruction sur le registre d'état ;
- l'inversion des modes d'adressage valides pour les opérandes source et destination (pour la première syntaxe s'il y en a plusieurs) ;
- l'inversion des longueurs valides (pour la première syntaxe s'il y en a plusieurs) ;
- si nécessaire, un message signalant un cas particulier (pour la première syntaxe...).

### Changement de syntaxe

**Avec la souris** en cliquant sur le bouton correspondant, de la manière habituelle.


**Avec le clavier :**

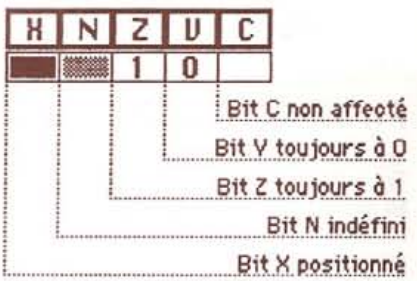
- sur un Mac 128 ou 512 en appuyant sur la touche *Entrée* ;
- sur un Mac Plus, en appuyant sur *Entrée* ou *Annulation*.

Les indications fournies par l'accessoire sont modifiées en conséquence.

## Le registre d'état

Pour indiquer l'effet des instructions sur les bits du registre d'état, nous avons employé la méthode suivante :

- case noircie pour un bit positionné (mis à 0 ou 1 en fonction du résultat) ;
- case grisé pour un bit dont l'état n'est pas défini ;
- un '1' si le bit est toujours à 1 ;
- un '0' si le bit est toujours à 0 ;
- une case vide si l'état du bit n'est pas modifié par l'instruction.




### Fichier 'F68000.Job'

```
Asm  S68000.Asm   Exec          Edit
Link F68000.Link  Font-DA Mover Edit
```

### Fichier 'F68000.Link'

```
]
/Resources
S68000
/output Accessoire 68000
/Type 'DFIL' 'DMOV'
$
```

### Source 'S68000.Asm'

```
INCLUDE  S68000/1.Asm
INCLUDE  S68000/2.Asm
         END
```

### Source 'S68000/1.Asm'

Note : le caractère '∫' indique la continuité d'une ligne.

```
RESOURCE 'DRVR' 29 'S8000*s'

.Trap  _BeginUpdate  $A922
.Trap  _CopyBits     $A8EC
.Trap  _DisposWindow $A914
.Trap  _DrawChar     $A883
.Trap  _DrawControls $A969
.Trap  _DrawString   $A884
.Trap  _EndUpdate    $A923
.Trap  _EraseRect    $A8A3
.Trap  _FindControl  $A96C
.Trap  _FrameRect    $A8A1
.Trap  _GetCRefCon   $A95A
.Trap  _GetCtlValue  $A960
.Trap  _GetCursor    $A9B9
.Trap  _GetMouse     $A972
.Trap  _GetPixel     $A865
.Trap  _GetPort      $A874
.Trap  _GlobalToLocal $A871
.Trap  _HideControl  $A958
.Trap  _InitCursor   $A850
.Trap  _InvertRect   $A8A4
.Trap  _LineTo       $A891
.Trap  _MoveTo       $A893
.Trap  _NewControl   $A954
.Trap  _NewPtr       $A11E
.Trap  _NewWindow    $A913
.Trap  _PaintRect    $A8A2
.Trap  _PenMode      $A89C
.Trap  _PenNormal    $A89E
.Trap  _PenPat       $A89D
.Trap  _PenSize      $A89B
.Trap  _PtInRect     $A8AD
.Trap  _SetCtlValue  $A963
.Trap  _SetCursor    $A851
.Trap  _SetPort      $A873
.Trap  _ShowControl  $A957
.Trap  _StillDown    $A973
.Trap  _SysBeep      $A9C8
.Trap  _TEInit       $A9CC
.Trap  _TextFont     $A887
```

```
.Trap   _TextMode       $A889
.Trap   _TextSize       $A88A
.Trap   _TrackControl   $A968
.Trap   _ValidRect      $A92A

SrcOr       Equ     1
SrcXOr      Equ     2
PatCopy     Equ     8
PatOr       Equ     9
PatXor      Equ     10
geneva      Equ     3
monaco      Equ     4
White       Equ     -8
Black       Equ     -16
Gray        Equ     -24
top         Equ     0
left        Equ     2
bottom      Equ     4
right       Equ     6
v           Equ     0
h           Equ     2
arrow       Equ     -108

mButDwnEvt  Equ     1
keyDwnEvt   Equ     3
keyUpEvt    Equ     4
autoKeyEvt  Equ     5
updatEvt    Equ     6
activateEvt Equ     8

evtNum      Equ     $0
evtMessage  Equ     $2
evtMouse    Equ     $A
evtMeta     Equ     $E
evtMBut     Equ     $F

accEvent    Equ     $40
accCursor   Equ     $42

activeFlag  Equ     0
shiftKey    Equ     9

portRect    Equ     $10
WindowKind  Equ     $6C
WindowSize  Equ     $9C

noGrowDocProc Equ   4
radioButProc  Equ   2
iBeamCursor   Equ   1

contrlRect  Equ     $8
contrlVis   Equ     $10

dCtlWindow  Equ     $1E
dCtlRefNum  Equ     $18

CSCode      Equ     $1A
CSParam     Equ     $1C

TempRect    Equ     $0FA
clear       Equ     $200

; Equivalences qui seront utilisées pour
; indiquer les modes d'adressage
; autorisés : PasM = Pas de Mode
; d'adressage, M1 = Mode 1 ( Dn)...
PasM    Equ     %0000000000000000
M1      Equ     %0000000000000001
M2      Equ     %0000000000000010
M3      Equ     %0000000000000100
M4      Equ     %0000000000001000
M5      Equ     %0000000000010000
M6      Equ     %0000000000100000
M7      Equ     %0000000001000000
M8      Equ     %0000000010000000
M9      Equ     %0000000100000000
M10     Equ     %0000001000000000
M11     Equ     %0000010000000000
M12     Equ     %0000100000000000
TouM    Equ     %0000111111111111

; Pour indiquer les longueurs autorisées :
; A_B = Adressage d'octets, A_W =
; adressage de mots...
A_B     Equ     %0001000000000000
A_W     Equ     %0010000000000000
A_L     Equ     %0100000000000000
A_BWL   Equ     %0111000000000000

; Pour l'effet des instruction sur le registre
; d'état : BXp = Bit X positionné, BNi = Bit N
; indéfini, BZna = Bit Z non affecté...
BXp     Equ     %0000000000000000
BXi     Equ     %0000000000000001
BXna    Equ     %0000000000000010
BX0     Equ     %0000000000000011
BX1     Equ     %0000000000000100
BNp     Equ     %0000000000000000
BNi     Equ     %0000000000001000
BNna    Equ     %0000000000010000
BN0     Equ     %0000000000011000
BN1     Equ     %0000000000100000
BZp     Equ     %0000000000000000
BZi     Equ     %0000000001000000
BZna    Equ     %0000000010000000
BZ0     Equ     %0000000011000000
BZ1     Equ     %0000000100000000
BVp     Equ     %0000000000000000
BVi     Equ     %0000001000000000
BVna    Equ     %0000010000000000
BV0     Equ     %0000011000000000
BV1     Equ     %0000100000000000
BCp     Equ     %0000000000000000
BCi     Equ     %0001000000000000
BCna    Equ     %0010000000000000
BC0     Equ     %0011000000000000
BC1     Equ     %0100000000000000

; Equivalences pour indiquer le numéro des
; messages des cas particuliers.
Msg1    Equ     %00100000
Msg2    Equ     %01000000
Msg3    Equ     %01100000
Msg4    Equ     %10000000
Msg5    Equ     %10100000

;Touches.
Ent     Equ     3       ; Entrée
BS      Equ     8       ; Back Space
Tab     Equ     9       ; Tabulation
CR      Equ     13      ; Ret. chariot
FlG     Equ     28      ; Flèche <-
FlD     Equ     29      ; Flèche ->
FlH     Equ     30      ; Flèche Haut
FlB     Equ     31      ; Flèche Bas
Ann     Equ     27      ; Annulation
Oui     Equ     $100

; Décalages par rapport à la base du tampon
; de mémoire créé par l'accessoire.
HandlesCtl      Equ     WindowSize
MnemoCourant    Equ     HandlesCtl+12
Bit15           Equ     MnemoCourant+2
BitMapIcone     Equ     Bit15+2
DrapeauCurs     Equ     BitMapIcone+14
NumeroSyntaxe   Equ     DrapeauCurs+1
HandleCtl       Equ     NumeroSyntaxe+1
Position        Equ     HandleCtl+4
IndMessages     Equ     Position+4
BoutonCourant   Equ     IndMessages+3
IndSources      Equ     BoutonCourant+1
IndDestinations Equ     IndSources+12
IndCCR          Equ     IndDestinations+12
IndLongueur     Equ     IndCCR+6
DrapeauCCR      Equ     IndLongueur+3
CodeCCR         Equ     DrapeauCCR+1
NbrsOctets      Equ     CodeCCR+2

Debut
        Dc      $400
        Dc      0
        Dc      $14A
        Dc      0
        Dc      Ouvre-Debut
        Dc      Etat-Debut
        Dc      Action-Debut
        Dc      Etat-Debut
        Dc      Ferme-Debut
Titre
        DC.B    7,'68000"s'

Ouvre
        Movem.L D3-D7/A1-A4,-(SP)
        Movea.L A1,A4
        Tst.L   dCtlWindow(A4)
        Bne     Etat3
        Subq.L  #4,SP
        Move.L  SP,-(SP)
        _GetPort
        Move.L  #NbreOctets,D0
        _NewPtr,clear
        Beq.S   Possible
        Move    #7,-(SP)
        _SysBeep
        Bra     Etat2
Possible
        Movea.L A0,A3
        Subq.L  #4,SP
        Move.L  A3,-(SP)
        Pea     CadreFenetre
        Pea     Titre
        Move    #Oui,-(SP)
        Move    #noGrowDocProc,-(SP)
        Moveq.L #-1,D0
        Move.L  D0,-(SP)
        Move    #Oui,-(SP)
        Clr.L   -(SP)
        _NewWindow
        _SetPort
        Move.L  A3,dCtlWindow(A4)
        Move.L  DctlRefNum(A4),∫
                WindowKind(A3)
        Bsr     Boutons
        Lea     IndMessages(A3),A0
        Moveq   #18,D0
@1      Clr     (A0)+
        Dbra    D0,@1
        Move    #$8000,Bit15(A3)
        Move    #41,MnemoCourant(A3)
        Bsr     Traitement
        Pea     RectSyntaxes1
        _ValidRect
        Pea     RectSyntaxes2
        _ValidRect
Etat2
        _SetPort
Etat3
        Movem.L (SP)+,D3-D7/A1-A4
Etat
        Moveq   #0,D0
        Rts

Ferme
        Movem.L A3-A4,-(SP)
        Movea.L A1,A4
        Move.L  dCtlWindow(A4),-(SP)
        _DisposWindow
        Clr.L   dCtlWindow(A4)
        Movea.L A4,A1
        Movem.L (SP)+,A3-A4
        Bra     Etat

Action
        Movem.L D3-D7/A1-A4,-(SP)
        Movea.L A1,A4
        Movea.L A0,A2
        Movea.L dCtlWindow(A4),A3
        Move.L  A3,-(SP)
        _SetPort
        Move    CSCode(A2),D0
        Cmpi    #accEvent,D0
        Beq.S   Evenement
        Cmpi    #accCursor,D0
        Beq     Curseur
        Bra     Etat3

Evenement
        Movea.L CSParam(A2),A2
        Move    EvtNum(A2),D0
        Cmpi    #mButDwnEvt,D0
        Beq.S   Contenu
        Cmpi    #keyDwnEvt,D0
        Beq     Touche
        Cmpi    #autoKeyEvt,D0
        Beq     Touche
        Cmpi    #updatEvt,D0
        Beq     MiseJour
        Cmpi    #activateEvt,D0
        Beq     Active
        Bra     Etat3

; Cherche l'emplacement où a eu lieu le 'clic'
Contenu
        Pea     EvtMouse(A2)
        _GlobalToLocal
        Clr     -(SP)
        Move.L  evtMouse(A2),-(SP)
        Pea     RectMnemo
        _PtInRect
        Tst     (SP)+
        Beq.S   @2
        Move    MnemoCourant(A3),D0
        Bsr     ZoneMnemo
        Cmp     MnemoCourant(A3),D0
        Beq.S   @1
        Bsr     Traitement
        Bra.S   @1
@2      Move.B  BoutonCourant(A3),D6
        Bmi.S   @1
        Clr-(SP)
        Move.L  EvtMouse(A2),-(SP)
        Move.L  A3,-(SP)
        Pea     HandleCtl(A3)
        _FindControl
        Tst     (SP)+
        Beq.S   @1
        Clr     -(SP)
        Move.L  HandleCtl(A3),-(SP)
        Move.L  EvtMouse(A2),-(SP)
        Clr.L   -(SP)
        _TrackControl
        Tst     (SP)+
        Beq.S   @1
        Subq    #4,SP
        Move.L  HandleCtl(A3),-(SP)
        _GetCRefCon
        Move    D6,D5
        Andi.B  #$30,D5
        Ext     D5
        Lsr     #4,D5
        Andi    #$0003,D6
        Move.L  (SP)+,D7
        Cmp     D6,D7
        Beq.S   @1
        Bsr     ChangI
                eSyntaxe
@1      Bra     Etat3
```

```
; En cas d'action sur une touche?
Touche
        Move.B  evtMessage+3(A2),D7
        Move    MnemoCourant(A3),D6
        Cmpi.B  #CR,D7
        Bne.S   @1
        Btst    #shiftKey,evtMeta(A2)
        Beq.S   Bas
        Bra.S   Haut
@1      Cmpi.B  #BS,D7
        Beq.S   Gauche
        Cmpi.B  #Tab,D7
        Beq.S   Droite
        Cmpi.B  #Ent,D7
        Beq.S   TrBoutons
        Sub.B   #27,D7
        Beq.S   TrBoutons
        Subq.B  #1,D7
        Beq.S   Gauche
        Subq.B  #1,D7
        Beq.S   Droite
        Subq.B  #1,D7
        Beq.S   Haut
        Subq.B  #1,D7
        Beq.S   Bas
        Bra     Etat3
Bas     ; Flèche bas ou Retour chariot
        Addq    #4,D6
        Cmpi    #83,D6
        Ble.S   SelectionTouche
        Sub     #84,D6
        Bra.S   SelectionTouche
Haut    ; Flèche haut ou Shift/Ret. chariot
        Subq    #4,D6
        Tst     D6
        Bge.S   SelectionTouche
        Add     #84,D6
        Bra.S   SelectionTouche
Gauche  ; Flèche <- ou Back Space
        Subq    #1,D6
        Move    D6,D7
        Andi    #3,D7
        Cmpi    #3,D7
        Bne.S   SelectionTouche
        Addq    #4,D6
        Bra.S   SelectionTouche
Droite  ; Flèche -> ou Tabulation
        Addq    #1,D6
        Move    D6,D7
        Andi    #3,D7
        Bne.S   SelectionTouche
        Subq    #4,D6
SelectionTouche
        Move    MnemoCourant(A3),-(SP)
        Move    Bit15(A3),D7
        Or      D7,(SP)
        Bsr     SelectionMnemo
        Move    D6,MnemoCourant(A3)
        Move    D6,-(SP)
        Or      D7,(SP)
        Bsr     SelectionMnemo
        Bsr     Traitement
        Bra     Etat3
TrBoutons       ; Entrée ou 'Annulation'
        Move.B  BoutonCourant(A3),D7
        Bmi.S   @1
        Move    D7,D5
        Andi.B  #$30,D5
        Ext     D5
        Lsr     #4,D5
        Andi    #$0003,D7
        Addq    #1,D7
        Cmp     D5,D7
        Ble.S   @2
        Moveq   #0,D7
@2      Bsr     ChangeSyntaxe
@1      Bra     Etat3

; Routine de mise à jour. Appelle tous les
; sous-programmes d'affichage.
MiseJour
        Clr     Bit15(A3)
        Move.L  A3,-(SP)
        Move.L  (SP),-(SP)
        _BeginUpdate
        _DrawControls
        Move    MnemoCourant(A3),-(SP)
        Bsr     SelectionMnemo
        Bsr     AfficheSelec
        Bsr     AfficheCCR
        Bsr     Tableaux
```

```
    Bsr     AfficheIcone
    Bsr     SyntaxeInstruction
    Bsr     AfficheMessage
    Move.L  A3,-(SP)
    _EndUpdate
    Move    #$8000,Bit15(A3)
    Bra     Etat3

; Fenêtre active au inactive.
Active
    Btst    #activeFlag,evtMBut(A2)
    Beq.S   @1
    _InitCursor
    Clr.B   DrapeauCurs(A3)
@1  Bra     Etat3

; Pour le changement de la forme du curseur
; en fonction de sa position.
Cursaur
Buffer  Set  -4
    Link    A6,#Buffer
    Pea     Buffer(A6)
    _GetMouse
    Move.L  Buffer(A6),D4
    Move.B  DrapeauCurs(A3),D6
    Moveq   #0,D7
    Clr     -(SP)
    Move.L  D4,-(SP)
    Pea     portRect(A3)
    _PtInRect
    Tst     (SP)+
    Bne.S   @1
    Tst.B   D6
    Beq.S   @2
    Movea.L (A5),A0
    Pea     arrow(A0)
    Bra.S   @3
@1  Moveq   #1,D7
    Clr     -(SP)
    Move.L  D4,-(SP)
    Pea     RectMnemo
    _PtInRect
    Tst     (SP)+
    Beq.S   @4
@6  Cmp.B   D7,D6
    Beq.S   @2
    Pea     Curseur1
    Bra.S   @3
@4  Lea     HandlesCtl(A3),A4
    Moveq   #8,D5
@7  Move.L  0(A4,D5),A0
    Move.L  (A0),A0
    Tst     contrlVis(A0)
    Beq.S   @5
    Clr     -(SP)
    Move.L  D4,-(SP)
    Pea     contrlRect(A0)
    _PtInRect
    Tst     (SP)+
    Bne.S   @6
@5  Subq    #4,D5
    Bge.S   @7
    Moveq   #2,D7
    Cmp.B   D7,D6
    Beq.S   @2
    Pea     Cursaur2
@3  _SetCursor
    Move.B  D7,DrapeauCurs(A3)
@2  Unlk    A6
    Bra     Etat3


; Affichage des tableaux.
Tableaux
    Moveq   #21,D4
    Moveq   #1,D5
    Moveq   #2,D6
    Move    #162,D7
@1  Move    D6,-(SP)
    Move    D5,-(SP)
    Move    D7,-(SP)
    Bsr     LigneHorizontale
    Addi    #12,D5
    Dbra    D4,@1

    Moveq   #4,D4
    Moveq   #1,D5
    Move    #253,D7
@2  Move    D6,-(SP)
    Move    D5,-(SP)
    Move    D7,-(SP)
    Bsr     LigneVerticale
    Addi    #40,D6
    Dbra    D4,@2
    Moveq   #0,D3
    Bsr     Tableaux2
    Moveq   #57,D3
    Bsr     Tableaux2
    Lea     Modes,A4
    Sf      -(SP)
    Move    #173,-(SP)
    Move    #14,-(SP)
    Pea     (A4)
    Bsr     AfficheChaine
    Sf      -(SP)
    Move    #236,-(SP)
    Move    #14,-(SP)
    Pea     7(A4)
    Bsr     AfficheChaine
    Bsr     Tableaux3
; Bits d'état.
    Lea     Bits,A4
    Moveq   #4,D4
    Move    #176,D7
@3  Move    D7,-(SP)
    Move    #236,-(SP)
    _MoveTo
    Move.B  (A4)+,D0
    Ext     D0
    Move    D0,-(SP)
    _DrawChar
    Add     #22,D7
    Dbra    D4,@3
; Affichage des codes mnémoniques.
    Bsr     PoliceMonaco
    Moveq   #0,D7
    Moveq   #20,D5
    Moveq   #11,D4
@4  Moveq   #5,D3
    Moveq   #3,D6
@5  Bsr.S   AdresseMnemo
    Move.B  (A4),-(SP)
    Bgt.S   @6
    Andi.B  #$7F,(A4)
    Addi.B  #2,(A4)
@6  Move    D7,D0
    Lsr     #1,D0
    Cmp     MnemoCourant(A3),D0
    Seq     -(SP)
    Move    D3,-(SP)
    Move    D4,-(SP)
    Move.L  A4,-(SP)
    Bsr     AfficheChaine
    Move.B  (SP)+,(A4)
    Addq    #2,D7
    Add     #40,D3
    Dbra    D6,@5
    Add     #12,D4
    Dbra    D5,@4
    Bra     SuiteTableau

; Pour obtenir l'adresse de base des
; informations sur l'instruction numéro N.
AdresseMnemo
    Move    TableMnemo(D7),D0
    Lea     TableMnemo(D0),A4
    Rts

TableMnemo
    Dc      xAbcd-TableMnemo
    Dc      xAdd-TableMnemo
    Dc      xAdda-TableMnemo
    Dc      xAddi-TableMnemo
    Dc      xAddq-TableMnemo
    Dc      xAddx-TableMnemo
    Dc      xAnd-TableMnemo
    Dc      xAndi-TableMnemo
    Dc      xAndC-TableMnemo
    Dc      xAndS-TableMnemo
    Dc      xAsl-TableMnemo
    Dc      xAsr-TableMnemo
    Dc      xBcc-TableMnemo
    Dc      xBchg-TableMnemo
    Dc      xBClr-TableMnemo
    Dc      xBra-TableMnemo
    Dc      xBset-TableMnemo
    Dc      xBsr-TableMnemo
    Dc      xBtst-TableMnemo
    Dc      xChk-TableMnemo
    Dc      xClr-TableMnemo
    Dc      xCmp-TableMnemo
    Dc      xCmpa-TableMnemo
    Dc      xCmpi-TableMnemo
    Dc      xCmpm-TableMnemo
    Dc      xDBcc-TableMnemo
    Dc      xDivs-TableMnemo
    Dc      xDivu-TableMnemo
    Dc      xEor-TableMnemo
    Dc      xEori-TableMnemo
    Dc      xEorC-TableMnemo
    Dc      xEorS-TableMnemo
    Dc      xExg-TableMnemo
    Dc      xExt-TableMnemo
    Dc      xIll-TableMnemo
    Dc      xJmp-TableMnemo
    Dc      xJsr-TableMnemo
    Dc      xLea-TableMnemo
    Dc      xLink-TableMnemo
    Dc      xLsl-TableMnemo
    Dc      xLsr-TableMnemo
    Dc      xMove-TableMnemo
    Dc      xMovC-TableMnemo
    Dc      xMovea-TableMnemo
    Dc      xMovU-TableMnemo
    Dc      xMovtS-TableMnemo
    Dc      xMovtS-TableMnemo
    Dc      xMovem-TableMnemo
    Dc      xMovep-TableMnemo
    Dc      xMoveq-TableMnemo
    Dc      xMuls-TableMnemo
    Dc      xMulu-TableMnemo
    Dc      xNbcd-TableMnemo
    Dc      xNeg-TableMnemo
    Dc      xNegx-TableMnemo
    Dc      xNop-TableMnemo
    Dc      xNot-TableMnemo
    Dc      xOr-TableMnemo
    Dc      xOri-TableMnemo
    Dc      xOrC-TableMnemo
    Dc      xOrS-TableMnemo
    Dc      xPea-TableMnemo
    Dc      xReset-TableMnemo
    Dc      xRol-TableMnemo
    Dc      xRor-TableMnemo
    Dc      xRoxl-TableMnemo
    Dc      xRoxr-TableMnemo
    Dc      xRte-TableMnemo
    Dc      xRtr-TableMnemo
    Dc      xRts-TableMnemo
    Dc      xSbcd-TableMnemo
    Dc      xScc-TableMnemo
    Dc      xStop-TableMnemo
    Dc      xSub-TableMnemo
    Dc      xSuba-TableMnemo
    Dc      xSubi-TableMnemo
    Dc      xSubq-TableMnemo
    Dc      xSubx-TableMnemo
    Dc      xSwap-TableMnemo
    Dc      xTas-TableMnemo
    Dc      xTrap-TableMnemo
    Dc      xTrapv-TableMnemo
    Dc      xTst-TableMnemo
    Dc      xUnlk-TableMnemo

SuiteTableau
    Move    #256,D7
    Move    #298,D6
    Move    D7,-(SP)
    Move    (SP),-(SP)
    Move    D6,-(SP)
    Bsr     LigneVerticale
    Move    #278,D5
    Move    D5,-(SP)
    Move    D7,-(SP)
    Move    D6,-(SP)
    Bsr     LigneVerticale
    Moveq   #3,D4
@7  Move    D7,-(SP)
    Move    D6,-(SP)
    Move    D5,-(SP)
    Bsr     LigneHorizontale
    Subi    #14,D6
    Dbra    D4,@7
; Affichage .B, .W et .L.
    Bsr     PoliceChicago
    Lea     Long,A4
    Moveq   #0,D5
    Move    D5,D3
    Addq    #4,D7
    Move    #268,D6
    Moveq   #2,D4
    Lea     IndLongueur(A3),A2
@8  Move.B  0(A2,D3),-(SP)
    Move    D7,-(SP)
    Move    D6,-(SP)
    Pea     0(A4,D5)
    Bsr     AfficheChaine
    Addq    #1,D3
    Addq    #3,D5
    Add     #14,D6
    Dbra    D4,@8
    Rts
Tableaux2
    Moveq   #19,D7
    Move    #168,D5
    Move    #221,D6
    Moveq   #12,D4
@1  Move    D5,-(SP)
    Add     D3,(SP)
    Move    D7,-(SP)
    Move    D6,-(SP)
    Add     D3,(SP)
    Bsr     LigneHorizontale
    Addi    #16,D7
    Dbra    D4,@1
    Subi    #16,D7
    Moveq   #19,D4
    Move    D5,-(SP)
    Add     D3,(SP)
    Move    D4,-(SP)
    Move    D7,-(SP)
    Bsr     LigneVerticale
    Move    D6,-(SP)
    Add     D3,(SP)
    Move    D4,-(SP)
    Move    D7,-(SP)
    Bsr     LigneVerticale
    Move.L  #$00020002,-(SP)
    _PenSize
    Lea     TempRect,A4
    Move    #1,(A4)
    Move    D5,left(A4)
    Move    #18,bottom(A4)
    Addq    #1,D6
    Move    D6,right(A4)
    Add     D3,left(A4)
    Add     D3,right(A4)
    Move.L  A4,-(SP)
    _FrameRect
    _PenNormal

; Affichage des modes d'adressage.
    Bsr     PoliceChicago
    Lea     Modes,A4
    Lea     IndSources,A2
    Move    #170,D6
    Moveq   #32,D7
    Moveq   #11,D5
    Moveq   #0,D4
    Moveq   #0,D2
@2  Tst     D3
    Bne.S   @3
    Move.B  0(A2,D2),-(SP)
    Bra.S   @4
@3  Move.B  12(A2,D2),-(SP)
@4  Move    D6,-(SP)
    Add     D3,(SP)
    Move    D7,-(SP)
    Pea     13(A4,D4)
    Bsr     AfficheChaine
    Move.B  13(A4,D4),D0
    Ext     D0
    Addq    #1,D0
    Add     D0,D4
    Add     #16,D7
    Dbra    D5,@2
    Rts
Tableaux3
    Move.L  #$00020002,-(SP)
    _PenSize
    Lea     TempRect,A4
    Move    #223,(A4)
    Move    #168,left(A4)
    Move    #240,bottom(A4)
    Move    #279,right(A4)
    Move.L  A4,-(SP)
    _FrameRect
    Moveq   #3,D0
    Move.L  D0,-(SP)
    _PenSize
    Moveq   #3,D4
    Move    #189,D7
@1  Move    D7,-(SP)
    Move    #225,-(SP)
    Move    #238,-(SP)
    Bsr     LigneVerticale
    Add     #22,D7
    Dbra    D4,@1
    _PenNormal
    Move    #241,(A4)
    Move    #168,left(A4)
    Move    #254,bottom(A4)
    Move    #279,right(A4)
    Move.L  A4,-(SP)
    _FrameRect
    Moveq   #3,D4
    Move    #190,D7
@2  Move    D7,-(SP)
    Move    #242,-(SP)
    Move    #252,-(SP)
    Bsr     LigneVerticale
    Add     #22,D7
    Dbra    D4,@2
    Rts

AfficheIcone
    Move.L  A4,-(SP)
    Lea     BitMapIcone(A3),A4
    Move    #4,4(A4)
    Clr.L   6(A4)
    Move.L  #$00200020,10(A4)
```

```
        Lea     Icone,A2
        Move.L  A2,(A4)
        Move.L  A4,-(SP)
        Movea.L A3,A4
        Addq.L  #2,A4
        Move.L  A4,-(SP)
        Pea     BitMapIcone+6(A3)
        Pea     RectDest
        Clr     -(SP)
        Clr.L   -(SP)
        _CopyBits
        Lea     BitMapIcone(A3),A4
        Move    #14,4(A4)
        Clr.L   6(A4)
        Move.L  #$0007006F,10(A4)
        Lea     Poms,A2
        Move.L  A2,(A4)
        Move.L  A4,-(SP)
        Movea.L A3,A4
        Addq.L  #2,A4
        Move.L  A4,-(SP)
        Pea     BitMapIcone+6(A3)
        Pea     RectDestPoms
        Clr     -(SP)
        Clr.L   -(SP)
        _CopyBits
        Movea.L (SP)+,A4
        Rts
```

; Initialisation des boutons de contrôle.

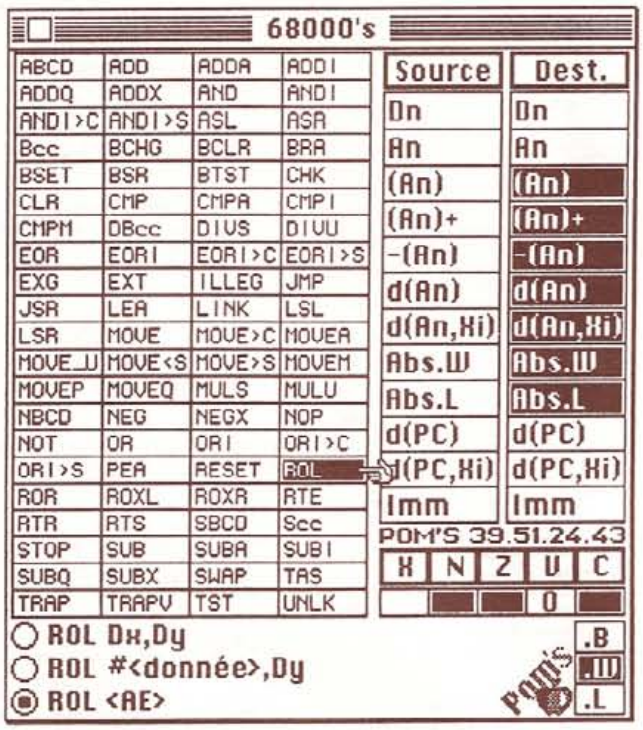
```
Boutons
        Moveq   #0,D7
        Moveq   #2,D6
        Lea     TempRect,A4
        Move    #256,(A4)
        Clr     left(A4)
        Move    #268,bottom(A4)
        Move    #14,right(A4)
@1      Bsr.S   Boutons2
        Addq    #1,D7
        Add     #15,(A4)
        Add     #15,bottom(A4)
        Dbra    D6,@1
        Rts
Boutons2
        Subq    #4,SP
        Move.L  A3,-(SP)
        Pea     TempRect
        Pea     PasTitre
        Clr.L   -(SP)
        Clr     -(SP)
        Move    #1,-(SP)
        Move    #radioButProc,-(SP)
        Move.L  D7,-(SP)
        _NewControl
        Lea     HandlesCtl(A3),A2
        Move    D7,D0
        Lsl     #2,D0
        Move.L  (SP)+,0(A2,D0)
        Rts
```

; La routine 'NumeroMnemoSouris' retourne
; au sommet de la pile le numéro de
; l'instruction 'cliquée', ou $FFXX si le 'clic' a
; eu lieu hors du tableau de sélection.

```
NumeroMnemoSouris
Parametre       Set     12
        Movem.L D0-D1,-(SP)
        Clr     -(SP)
        Move.L  evtMouse(A2),-(SP)
        Pea     RectMnemo
        _PtInRect
        Tst     (SP)+
        Seq     Parametre(SP)
        Beq.S   @1
        Move    evtMouse+v(A2),D0
        Subq    #1,D0
        Ext.L   D0
        Divu    #12,D0
        Lsl     #2,D0
        Move    evtMouse+h(A2),D1
        Subq    #2,D1
        Ext.L   D1
        Divu    #40,D1
        Add     D1,D0
        Move    D0,Parametre(SP)
@1      Movem.L (SP)+,D0-D1
        Rts
```

; La routine 'SelectionMnemo' met en
; inverse l'instruction selectionnée. Le
; numéro de l'instruction doit être empilé
; avec en plus le bit 15 à 0 pour une mise à
; jour, à 1 pour le traitement normal.

```
SelectionMnemo
DrInvert        Set     -2
BitInvert       Set     15
Parametre       Set     8
Tampon          Set     -10
        Link    A6,#Tampon
```




```
        Movem.L D0-D1,-(SP)
        Move    Parametre(A6),D0
        Bclr    #BitInvert,D0
        SNE     DrInvert(A6)
        Move    D0,D1
        Andi    #3,D1
        Lsr     #2,D0
        Mulu    #12,D0
        Addq    #3,D0
        Move    D0,Tampon+top(A6)
        Add     #9,D0
        Move    D0,Tampon+bottom(A6)
        Mulu    #40,D1
        Addq    #4,D1
        Move    D1,Tampon+left(A6)
        Add     #37,D1
        Move    D1,Tampon+right(A6)
        Tst.B   DrInvert(A6)
        Bne.S   @1
        Move    #PatCopy,-(SP)
        Bra.S   @2
@1      Move    #PatXor,-(SP)
@2      _PenMode
        Pea     Tampon(A6)
        _PaintRect
        _PenNormal
        Movem.L (SP)+,D0-D1
        Unlk    A6
        Move.L  (SP),2(SP)
        Addq.L  #2,SP
        Rts
```

; Routine utilisée en cas d'actions dans la
; zone des instructions.

```
ZoneMnemo
        Movem.L D0-D1,-(SP)
@2      Subq.L  #2,SP
        Bsr     NumeroMnemoSouris
        Move    (SP)+,D0
        Bmi.S   @3
@1      Cmp     MnemoCourant(A3),D0
        Beq.S   @3
        Move    MnemoCourant(A3),-(SP)
        Move    Bit15(A3),D1
        Or      D1,(SP)
        Bsr     SelectionMnemo
        Move    D0,MnemoCourant(A3)
        Move    D0,-(SP)
        Or      D1,(SP)
        Bsr     SelectionMnemo
@3      Subq.L  #2,SP
        _StillDown
        Tst     (SP)+
        Beq.S   @4
        Pea     evtMouse(A2)
        _GetMouse
        Bra.S   @2
@4      Movem.L (SP)+,D0-D1
        Rts
```

; La routine 'SelectionMode' met en inverse
; un des modes d'adressage valides. Le
; numéro du mode doit être empilé (de 0 à
; 11) avec en plus le bit 15 à 0 pour une mise
; à jour, à 1 pour le traitement normal ; ainsi
; que le bit 14 qui doit être à 1 s'il s'agit de
; l'opérande destination.

```
SelectionMode
DrInvert        Set     -2
BitDestination  Set     6
BitInvert       Set     15
Param           Set     8
Tampon          Set     -10
        Link    A6,#Tampon
        Move.L  D0,-(SP)
        Move    Param(A6),D0
        Bclr    #BitInvert,D0
        SNE     DrInvert(A6)
        Ext     D0
        Mulu    #16,D0
        Add     #21,D0
        Move    D0,Tampon+top(A6)
        Add     #13,D0
        Move    D0,Tampon+bottom(A6)
        Move    #170,Tampon+left(A6)
        Move    #220,Tampon+right(A6)
        Btst    #BitDestination,Param(A6)
        Beq.S   @3
        Move    #57,D0
        Add     D0,Tampon+left(A6)
        Add     D0,Tampon+right(A6)
@3      Tst.B   DrInvert(A6)
        Bne.S   @1
        Move    #PatCopy,-(SP)
        Bra.S   @2
@1      Move    #PatXor,-(SP)
@2      _PenMode
        Pea     Tampon(A6)
        _PaintRect
        _PenNormal
        Move.L  (SP)+,D0
        Unlk    A6
        Move.L  (SP),2(SP)
        Addq.L  #2,SP
        Rts
```

; Fonctionnement identique à celui de
; 'selectionMnemo', mais cette fois pour la
; sélection des longueurs valides (octet,
; mot et long mot).

```
SelectionLong
DrInvert        Set     -2
BitInvert       Set     15
Parametre       Set     8
Tampon          Set     -10
        Link    A6,#Tampon
        Movem.L D0-D1,-(SP)
        Move    Parametre(A6),D0
        Bclr    #BitInvert,D0
        SNE     DrInvert(A6)
        Move    #258,D1
        Mulu    #14,D0
        Add     D1,D0
        Move    D0,Tampon+top(A6)
        Move    D1,Tampon+left(A6)
        Add     #19,D1
        Move    D1,Tampon+right(A6)
        Add     #11,D0
        Move    D0,Tampon+bottom(A6)
        Tst.B   DrInvert(A6)
        Bne.S   @1
        Move    #PatCopy,-(SP)
        Bra.S   @2
@1      Move    #PatXor,-(SP)
@2      _PenMode
        Pea     Tampon(A6)
        _PaintRect
        _PenNormal
        Movem.L (SP)+,D0-D1
        Unlk    A6
        Move.L  (SP),2(SP)
        Addq.L  #2,SP
        Rts
```

; Routine pour visualiser l'effet d'une
; instruction sur le registre d'état. Il faut,
; avant l'appel de la routine, empiler un mot
; de 16 bits contenant, dans le poids fort, le
; numéro du bit (0 pour X, 1 pour N...) et
; dans le poids faible l'effet (0 pour
; positionné, 1 pour indéfini...).

```
PositionsBits
ParamNumero     Set     9
ParamPosition   Set     8
Tampon          Set     -8
Indefini        Set     1
Toujours0       Set     3
Toujours1       Set     4
        Link    A6,#Tampon
        Movem.L D6-D7,-(SP)
        Bsr     PoliceChicago
        Move.B  ParamNumero(A6),D7
        Ext     D7
        Mulu    #22,D7
        Add     #170,D7
        Move    D7,Tampon+left(A6)
        Add     #19,D7
        Move    D7,Tampon+right(A6)
        Move    #243,Tampon+top(A6)
        Move    #252,Tampon+bottom(A6)
        _PenNormal
        Movea.L (A5),A0
        Move.B  ParamPosition(A6),D7
        Ext     D7
        Bne.S   @1
        Pea     Black(A0)
        Bra.S   @2
@1      Cmpi    #indefini,D7
        Bne.S   @3
        Pea     Gray(A0)
        Bra.S   @2
@3      Pea     White(A0)
@2      _PenPat
        Pea     Tampon(A6)
        _PaintRect
        Cmpi    #Toujours0,D7
        Bmi.S   @4
        Move    D7,D6
        Move    Tampon+bottom(A6),D7
        Swap    D7
        Move    Tampon+left(A6),D7
        Addq    #5,D7
        Move.L  D7,-(SP)
        _MoveTo
        Cmpi    #Toujours1,D6
        Beq.S   @5
        Move    #'0',-(SP)
        Bra.S   @6
@5      Move    #'1',-(SP)
@6      _DrawChar
@4      Movem.L (SP)+,D6-D7
        _PenNormal
        Unlk    A6
        Move.L  (SP),2(SP)
        Addq.L  #2,SP
        Rts
```

; Routine d'affichage des chaînes de
; caractères. Il faut empiler, dans l'ordre, un
; drapeau à 0 pour un affichage normal et à
; $FF pour un affichage en inverse, la
; position dans la fenêtre (x,y), et l'adresse
; de la chaîne à afficher.

```
AfficheChaine
AdresseChaine Set       8
```

```
Coordonnees     Set     12
Drapeau         Set     16
                Link    A6,#0
                Movem.L A2/D2,-(SP)
                Tst     Drapeau(A6)
                Beq.S   @1
                Move    #SrcXOr,-(SP)
                Bra.S   @2
@1  Move        #SrcOr,-(SP)
@2  _TextMode
                Move.L  Coordonnees(A6),-(SP)
                _MoveTo
                Move.L  AdresseChaine(A6),-(SP)
                _DrawString
                Movem.L (SP)+,A2/D2
                Unlk    A6
                Move.L  (SP),10(SP)
                Add     #10,SP
                Rts

; Routine pour tracé de lignes horizontales
; ou verticales

LigneHorizontale
Tampon          Set     -2
X_1             Set     12
Y_0             Set     10
X_2             Set     8
                Link    A6,#Tampon
                Sf      Tampon(A6)
                Bra.S   Lignes
LigneVerticale
X_0             Set     12
Y_1             Set     10
Y_2             Set     8
                Link    A6,#Tampon
                ST      Tampon(A6)
Lignes
                Move.L  Y_0(A6),-(SP)
                _MoveTo
                Tst.B   Tampon(A6)
                Beq.S   @1
                Move    X_0(A6),-(SP)
                Move    Y_2(A6),-(SP)
                Bra.S   @2
@1  Move        X_2(A6),-(SP)
                Move    Y_0(A6),-(SP)
@2  _LineTo
                Unlk    A6
                Move.L  (SP),6(SP)
                Addq    #6,SP
                Rts

; Traitement consécutif à un changement de
; code mnémonique.

Traitement
                Bsr.S   SyntaxeInstruction
                Bsr     ConfigBoutons
                Pea     RectControles
                _ValidRect
                Clr.B   NumeroSyntaxe(A3)
                Bsr.S   AfficheMessage
                Clr     D6
                ST      DrapeauCCR(A3)
                Bsr     Indications
@1  Rts

; Affichage des messages pour les cas
; particuliers.

AfficheMessage
                Btst    #5,BoutonCourant(A3)
                Bne.S   @2
                Pea     RectSyntaxes2
                _EraseRect
@2  Lea         IndMessages(A3),A4
                Move.B  NumeroSyntaxe(A3),D7
                Ext     D7
                Move.B  0(A4,D7),D6
                Beq.S   @1
                Move    #geneva,-(SP)
                _TextFont
                Move    #9,-(SP)
                _TextSize
                Move.L  #$01270012,-(SP)
                _MoveTo
                Subq.B  #1,D6
                Bne.S   @3
                Pea     Mg1
                Bra.S   @4
@3  Subq.B      #1,D6
                Bne.S   @5
                Pea     Mg2
                Bra.S   @4
@5  Subq.B      #1,D6
                Bne.S   @6
                Pea     Mg3
                Bra.S   @4
@6  Subq.B      #1,D6
                Bne.S   @7
                Pea     Mg4
                Bra.S   @4
@7  Pea         Mg5
@4  _DrawString
@1  Rts

; Affichage des syntaxes.

SyntaxeInstruction
                Move    MnemoCourant(A3),D7
                Add     D7,D7
                Bsr     AdresseMnemo
                Moveq   #0,D4
                Move.B  (A4),D7
                Btst    #7,D7
                Beq.S   @1
                Moveq   #2,D4
                Andi    #$7F,D7
                Addq    #2,D7
@1  Move.B      1(A4,D7),D6
                Ext     D6
                Move    D6,-(SP)
                Move.B  (A4),-(SP)
                Move.L  A4,-(SP)
                Sub     D4,D7
                Move.B  D7,(A4)
                Bsr     PoliceChicago
                Move    #PatOr,-(SP)
                _TextMode
                Move    #266,D5
                Move.L  A4,D3
                Add     D4,D7
                Lea     2(A4,D7),A4
                Lea     IndMessages(A3),A2
                Moveq   #0,D7
                Pea     RectSyntaxes1
                _EraseRect
                Pea     RectSyntaxes2
                _EraseRect
@2  Move        #18,-(SP)
                Move    D5,-(SP)
                _MoveTo
                Move.L  D3,-(SP)
                _DrawString
                Move.B  0(A4,D7),D4
                Move.B  D4,D0
                Andi.B  #$E0,D0
                Lsr.B   #5,D0
                Move.B  D0,0(A2,D7)
                Andi    #$1F,D4
@4  Bsr         AdresseSyntaxe
                Pea     (A0)
                _DrawString
@3  Add         #15,D5
                Addq    #1,D7
                Dbra    D6,@2
                Move.L  (SP)+,A4
                Move.B  (SP)+,(A4)
                Move    (SP)+,D6
@6  Rts

; Pour trouver l'adresse de base des
; informations sur chaque instruction.

AdresseSyntaxe
                Add     D4,D4
                Move    TableSynt(D4),D4
                Lea     TableSynt(D4),A0
                Rts
TableSynt
                Dc      Sy0-TableSynt
                Dc      Sy1-TableSynt
                Dc      Sy2-TableSynt
                Dc      Sy3-TableSynt
                Dc      Sy4-TableSynt
                Dc      Sy5-TableSynt
                Dc      Sy6-TableSynt
                Dc      Sy7-TableSynt
                Dc      Sy8-TableSynt
                Dc      Sy9-TableSynt
                Dc      Sy10-TableSynt
                Dc      Sy11-TableSynt
                Dc      Sy12-TableSynt
                Dc      Sy13-TableSynt
                Dc      Sy14-TableSynt
                Dc      Sy15-TableSynt
                Dc      Sy16-TableSynt
                Dc      Sy17-TableSynt
                Dc      Sy18-TableSynt
                Dc      Sy19-TableSynt
                Dc      Sy20-TableSynt
                Dc      Sy21-TableSynt
                Dc      Sy22-TableSynt
                Dc      Sy23-TableSynt
                Dc      Sy24-TableSynt
                Dc      Sy25-TableSynt
                Dc      Sy26-TableSynt
                Dc      Sy27-TableSynt
                Dc      Sy28-TableSynt
                Dc      Sy29-TableSynt
                Dc      Sy30-TableSynt
                Dc      Sy31-TableSynt

; Police Chicago 12 points.

PoliceChicago
                Clr     -(SP)
                _TextFont
                Move    #12,-(SP)
                _TextSize
                Rts

; Police monaco 9 points.

PoliceMonaco
                Move    #monaco,-(SP)
                _TextFont
                Move    #9,-(SP)
                _TextSize
                Rts

; Configurations des boutons après chaque
; changement de code mnémonique.

ConfigBoutons
                Tst     D6
                Bne.S   @1
                Move.B  #$80,BoutonCourant(A3)
                Lea     HandlesCtl(A3),A4
                Moveq   #2,D7
                Moveq   #0,D5
@2  Move.L      0(A4,D5),-(SP)
                _HideControl
                Addq    #4,D5
                Dbra    D7,@2
                Rts
@1  Bsr.S       Bouton1
                Lea     HandlesCtl(A3),A4
                Moveq   #4,D7
                Bsr.S   Bouton23
                Subq    #1,D6
                Bne.S   @3
                Move.B  #$10,BoutonCourant(A3)
                Move.L  HandlesCtl+8(A3),-(SP)
                _HideControl
                Rts
@3  Move.B      #$20,BoutonCourant(A3)
                Moveq   #8,D7
                Bsr.S   Bouton23
                Rts
Bouton23
                Subq    #2,SP
                Move.L  0(A4,D7),-(SP)
                _GetCtlValue
                Tst     (SP)+
                Beq.S   @1
                Move.L  0(A4,D7),-(SP)
                Clr     -(SP)
                _SetCtlValue
@1  Move.L      0(A4,D7),-(SP)
                _ShowControl
                Rts
Bouton1
                Subq    #2,SP
                Move.L  HandlesCtl(A3),-(SP)
                _GetCtlValue
                Tst     (SP)+
                Bne.S   @1
                Move.L  HandlesCtl(A3),-(SP)
                Move    #1,-(SP)
                _SetCtlValue
@1  Move.L      HandlesCtl(A3),-(SP)
                _ShowControl
                Rts

; Affichages des indications : modes
; d'adressage autorisés, registre d'état et
; longueurs permises.

Indications
                Move    MnemoCourant(A3),D7
                Add     D7,D7
                Bsr     AdresseMnemo
                Move.B  (A4),D7
                Btst    #7,D7
                Beq.S   @1
                Andi.B  #$7F,D7
                Addq    #2,D7
@1  Addq        #1,D7
                Ext     D7
                Add.B   0(A4,D7),D7
                Addq    #4,D7
                Lea     0(A4,D7),A4
                Move    A4,D0
                Btst    #0,D0
                Beq.S   @2
                Addq.L  #1,A4
@2  Move        -2(A4),CodeCCR(A3)
                Lsl     #2,D6
                Move.L  0(A4,D6),D7
                Lea     IndDestinations(A3),A4
                Move    #$4000,D5
                Bsr     IndiqueModes
                Swap    D7
                Lea     IndSources(A3),A4
                Moveq   #0,D5
                Bsr.S   IndiqueModes
                Swap    D7
                Lea     IndLongueur(A3),A4
                Moveq   #12,D0
                Lsr     D0,D7
                Bsr.S   IndiqueLong
                Tst.B   DrapeauCCR(A3)
                Beq.S   @3
                Move    CodeCCR(A3),D7
                Lea     IndCCR(A3),A4
                Moveq   #0,D6
@4  Move        D7,D5
                Andi    #$0007,D5
                Cmp.B   0(A4,D6),D5
                Beq.S   @5
                Move.B  D5,0(A4,D6)
                Lsl     #8,D5
                Or      D6,D5
                Move    D5,-(SP)
                Bsr     PositionsBits
@5  Lsr         #3,D7
                Addq    #1,D6
                Cmpi    #5,D6
                Bne.S   @4
@3  Rts
IndiqueLong
                Moveq   #0,D6
@1  Btst        D6,D7
                Beq.S   @2
                Tst.B   0(A4,D6)
                Bne.S   @3
                ST      0(A4,D6)
@4  Move        D6,-(SP)
                Move    Bit15(A3),D0
                Or      D0,(SP)
                Bsr     SelectionLong
                Bra.S   @3
@2  Tst.B       0(A4,D6)
                Beq.S   @3
                Sf      0(A4,D6)
                Bra.S   @4
@3  Addq        #1,D6
                Cmpi    #3,D6
                Bne.S   @1
                Rts
IndiqueModes
                Moveq   #0,D6
@1  Btst        D6,D7
                Beq.S   @2
                Tst.B   0(A4,D6)
                Bne.S   @3
                ST      0(A4,D6)
@4  Move        D6,-(SP)
                Move    Bit15(A3),D0
                Or      D0,(SP)
                Or      D5,(SP)
                Bsr     SelectionMode
                Bra.S   @3
@2  Tst.B       0(A4,D6)
                Beq.S   @3
                Sf      0(A4,D6)
                Bra.S   @4
@3  Addq        #1,D6
                Cmpi    #12,D6
                Bne.S   @1
                Rts

AfficheSelec
                Lea     IndSources(A3),A4
                Move    Bit15(A3),D5
                Moveq   #0,D6
@1  Tst.B       0(A4,D6)
                Beq.S   @2
                Move    D6,-(SP)
                Or      D5,(SP)
                Bsr     SelectionMode
@2  Tst.B       12(A4,D6)
                Beq.S   @3
                Move    D6,-(SP)
                Or      D5,(SP)
                Ori     #$4000,(SP)
                Bsr     SelectionMode
@3  Addq        #1,D6
                Cmpi    #12,D6
                Bne.S   @1
                Lea     IndLongueur(A3),A4
                Moveq   #0,D6
@4  Tst.B       0(A4,D6)
                Beq.S   @5
                Move    D6,-(SP)
                Or      D5,(SP)
                Bsr     SelectionLong
@5  Addq        #1,D6
```

```
    Cmpi    #3,D6
    Bne.S   @4
    Rts
AfficheCCR
    Lea     IndCCR(A3),A4
    Moveq   #0,D6
@1  Move.B  0(A4,D6),D0
    Lsl     #8,D0
    Or      D6,D0
    Move    D0,-(SP)
    Bsr     PositionsBits
    Addq    #1,D6
    Cmpi    #5,D6
    Bne.S   @1
    Rts

; Pour faire suite à un changement de
; syntaxe (souris ou clavier).
ChangeSyntaxe
    Tst     D7
    Bne.S   @1
    Move.L  HandlesCtl(A3),-(SP)
    Move    #1,-(SP)
    _SetCtlValue
    Move.L  HandlesCtl+4(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Cmpi    #2,D5
    Bne.S   @2
    Move.L  HandlesCtl+8(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Bra.S   @2
@1  Cmpi    #1,D7
    Bne.S   @3
    Move.L  HandlesCtl(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Move.L  HandlesCtl+4(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Move.L  HandlesCtl+8(A3),-(SP)
    Move    #1,-(SP)
    _SetCtlValue
    Cmpi    #2,D5
    Bne.S   @2
    Move.L  HandlesCtl+8(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Bra.S   @2
@3  Move.L  HandlesCtl(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Move.L  HandlesCtl+4(A3),-(SP)
    Clr     -(SP)
    _SetCtlValue
    Move.L  HandlesCtl+8(A3),-(SP)
    Move    #1,-(SP)
    _SetCtlValue
@2  Sf      DrapeauCCR(A3)
    Move.B  D7,NumeroSyntaxe(A3)
    Move    D7,D6
    Bsr     Indications
    Andi.B  #$F0,BoutonCourant(A3)
    Move.B  NumeroSyntaxe(A3),D7
    Or.B    D7,BoutonCourant(A3)
    Bsr     AfficheMessage
    Rts
```

## *Source 'S68000/2.Asm'*

```
; Liste des syntaxes possibles.
Sy0     Dc.B    $01,' '
Sy1     Dc.B    $06,' Dy,Dx'
Sy2     Dc.B    $0C,' -(Ay),-(Ax)'
Sy3     Dc.B    $08,' <AE>,Dn'
Sy4     Dc.B    $08,' Dn,<AE>'
Sy5     Dc.B    $08,' <AE>,An'
Sy6     Dc.B    $0F,' #<donnée>,<AE>'
Sy7     Dc.B    $15,' #<donnée 8 bits>,CCR'
Sy8     Dc.B    $15,' #<donnée 16 bits>,SR'
Sy9     Dc.B    $06,' Dx,Dy'
Sy10    Dc.B    $0D,' #<donnée>,Dy'
Sy11    Dc.B    $05,' <AE>'
Sy12    Dc.B    $0C,' <étiquette>'
Sy13    Dc.B    $0C,' (Ay)+,(Ax)+'
Sy14    Dc.B    $0F,' Dn,<étiquette>'
Sy15    Dc.B    $06,' Rx,Ry'
Sy16    Dc.B    $03,' Dn'
Sy17    Dc.B    $1A,' An,#<déplacement 16 ⌡
                bits>'
Sy18    Dc.B    $0A,' <AE>,<AE>'
Sy19    Dc.B    $09,' <AE>,CCR'
Sy20    Dc.B    $07,' USP,An'
Sy21    Dc.B    $07,' An,USP'
Sy22    Dc.B    $08,' SR,<AE>'
Sy23    Dc.B    $08,' <AE>,SR'
Sy24    Dc.B    $1A,' <liste de ⌡
                registres>,<AE>'
Sy25    Dc.B    $1A,' <AE>,<liste de ⌡
                registres>'
Sy26    Dc.B    $09,' Dx,d(Ay)'
Sy27    Dc.B    $09,' d(Ay),Dx'
Sy28    Dc.B    $0D,' #<donnée>,Dn'
Sy29    Dc.B    $12,' #<donnée 16 bits>'
Sy30    Dc.B    $0B,' #<vecteur>'
Sy31    Dc.B    $03,' An'

; Pour affichage du tableau des modes
; d'adressage.
Modes   Dc.B    $06,'Source'
        Dc.B    $05,'Dest.'
        Dc.B    $02,'Dn'
        Dc.B    $02,'An'
        Dc.B    $04,'(An)'
        Dc.B    $05,'(An)+'
        Dc.B    $05,'-(An)'
        Dc.B    $05,'d(An)'
        Dc.B    $08,'d(An,Xi)'
        Dc.B    $05,'Abs.W'
        Dc.B    $05,'Abs.L'
        Dc.B    $05,'d(PC)'
        Dc.B    $08,'d(PC,Xi)'
        Dc.B    $03,'Imm'

; Pour affichages divers.
Bits    Dc.B    'XNZVC'
Long    Dc.B    2,'.B',2,'.W',2,'.L'
Zero    Dc.B    '0'
Un      Dc.B    '1'

; Messages des cas particuliers.
Mg1     Dc.B    44,'Si <AE>=An .W et .L ⌡
                seulement, CCR inchangé.'
Mg2     Dc.B    35,'Si Destination = Dn : .L, ⌡
                sinon .B.'
Mg3     Dc.B    29,'Si Source = An : .B ⌡
                interdit.'
Mg4     Dc.B    36,'Si Source = An : .W et .L ⌡
                seulement.'
Mg5     Dc.B    43,'Si donnée/adresse, Ry = ⌡
                registre d''adresse.'
        .Align  2

; Informations relatives aux instructions du
; MC68000. Pour chaque instruction, on
; trouve :
; - un octet indiquant la longueur du code
; mnémonique, avec le bit de signe (7) à 1 si
; il y a deux caractères de plus (comme pour
; MOVE to SR, pour lequel le code est
; MOVE, mais qui est représenté dans le
; tableau par MOVE>S) ;
; - les caractères du code ;
; - un octet indiquant le nombre de
; syntaxe - 1 ;
; - pour chaque syntaxe, un octet contenant
; le numéro de syntaxe plus, si nécessaire,
; le numéro du message lié à cette syntaxe ;
; - si nécessaire, un octet à -1 servant
; uniquement à aligner les informations
; suivantes sur des adresses paires ;
; - un mot de 16 bits pour les informations
; relatives aux bits d'état (3 bits * 5) ;
; - pour chaque syntaxe, un mot de 16 bits
; pour indiquer les modes d'adressage
; autorisés pour l'opérande source ;
; - pour chaque syntaxe, un mot de 16 bits
; pour indiquer les modes d'adressage
; autorisés pour l'opérande destination et
; les longueurs valides (octet, mot et long
; mot).
xAbcd   Dc.B    $04,'ABCD',1,1,2
        Dc      BXp+BNi+BZp+BVi+BCp
        Dc      M1,M1+A_B
        Dc      M5,M5+A_B
xAdd    Dc.B    $03,'ADD',1,3+Msg4,4,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      TouM,M1+A_BWL
        Dc      M1,TouM-M1-M2-M10-M11⌡
                -M12+A_BWL
xAdda   Dc.B    $04,'ADDA',0,5,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      TouM,M2+A_W+A_L
xAddi   Dc.B    $04,'ADDI',0,6,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_BWL
xAddq   Dc.B    $04,'ADDQ',0,6+Msg1,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,TouM-M10-M11-M12+⌡
                A_BWL
xAddx   Dc.B    $04,'ADDX',1,1,2
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M1,M1+A_BWL
        Dc      M5,M5+A_BWL
xAnd    Dc.B    $03,'AND',1,3,4,-1
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      TouM-M2,M1+A_BWL
        Dc      M1,TouM-M1-M2-M10-M11⌡
                -M12+A_BWL
xAndI   Dc.B    $04,'ANDI',0,6,-1
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_BWL
xAndC   Dc.B    $84,'ANDI>C',0,7,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,PasM+A_B
xAndS   Dc.B    $84,'ANDI>S',0,8,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,PasM+A_W
xAsl    Dc.B    $03,'ASL',2,9,10,11
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M1,M1+A_BWL
        Dc      M12,M1+A_BWL
        Dc      PasM,TouM-M1-M2-M10-⌡
                M11-M12+A_W
xAsr    Dc.B    $03,'ASR',2,9,10,11
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M1,M1+A_BWL
        Dc      M12,M1+A_BWL
        Dc      PasM,TouM-M1-M2-M10-⌡
                M11-M12+A_W
xBcc    Dc.B    $03,'Bcc',0,12
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,PasM+A_B+A_W
xBchg   Dc.B    $04,'BCHG',1,4+Msg2,⌡
                6+Msg2
        Dc      BXna+BNna+BZp+BVna+⌡
                BCna
        Dc      M1,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
xBclr   Dc.B    $04,'BCLR',1,4+Msg2,⌡
                6+Msg2
        Dc      BXna+BNna+BZp+BVna+⌡
                BCna
        Dc      M1,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
xBra    Dc.B    $03,'BRA',0,12
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,PasM+A_B+A_W
xBset   Dc.B    $04,'BSET',1,4+Msg2,⌡
                6+Msg2
        Dc      BXna+BNna+BZp+BVna+⌡
                BCna
        Dc      M1,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
xBsr    Dc.B    $03,'BSR',0,12
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,PasM+A_B+A_W
xBtst   Dc.B    $04,'BTST',1,4+Msg2,⌡
                6+Msg2
        Dc      BXna+BNna+BZp+BVna+⌡
                BCna
        Dc      M1,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_B+A_L
xChk    Dc.B    $03,'CHK',0,3
        Dc      BXna+BNp+BZi+BVi+BCi
        Dc      TouM-M2,M1+A_W
xClr    Dc.B    $03,'CLR',0,11
        Dc      BXna+BN0+BZ1+BV0+BC0
        Dc      PasM,TouM-M2-M10-M11⌡
                -M12+A_BWL
xCmp    Dc.B    $03,'CMP',0,3+Msg4
        Dc      BXna+BNp+BZp+BVp+BCp
        Dc      TouM,M1+A_BWL
xCmpa   Dc.B    $04,'CMPA',0,5,-1
        Dc      BXna+BNp+BZp+BVp+BCp
        Dc      TouM,M2+A_W+A_L
xCmpi   Dc.B    $04,'CMPI',0,6,-1
        Dc      BXna+BNp+BZp+BVp+BCp
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_BWL
xCmpm   Dc.B    $04,'CMPM',0,13,-1
        Dc      BXna+BNp+BZp+BVp+BCp
        Dc      M4,M4+A_BWL
xDBcc   Dc.B    $04,'DBcc',0,14,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      M1,PasM+A_W
xDivs   Dc.B    $04,'DIVS',0,3,-1
        Dc      BXna+BNp+BZp+BVp+BC0
        Dc      TouM-M2,M1+A_W
xDivu   Dc.B    $04,'DIVU',0,3,-1
```

```
        Dc      BXna+BNp+BZp+BVp+BC0
        Dc      TouM-M2,M1+A_W
xEor    Dc.B    $03,'EOR',0,4
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      M1,TouM-M2-M10-M11-⌡
                M12+A_BWL
xEorI   Dc.B    $04,'EORI',0,6,-1
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      M12,TouM-M2-M10-M11-⌡
                M12+A_BWL
xEorC   Dc.B    $84,'EORI>C',0,7,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,PasM+A_B
xEorS   Dc.B    $84,'EORI>S',0,8,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      M12,PasM+A_W
xExg    Dc.B    $03,'EXG',0,15+Msg5
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      M1+M2,M1+M2+A_L
xExt    Dc.B    $03,'EXT',0,16
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      PasM,M1+A_W+A_L
xIll    Dc.B    $05,'ILLEG',0,0
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,PasM
xJmp    Dc.B    $03,'JMP',0,11
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,TouM-M1-M2-M4-⌡
                M5-M12
xJsr    Dc.B    $03,'JSR',0,11
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,TouM-M1-M2-M4-⌡
                M5-M12
xLea    Dc.B    $03,'LEA',0,5
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      TouM-M1-M2-M4-M5-M12,⌡
                M2+A_L
xLink   Dc.B    $04,'LINK',0,17,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      M2,M12
xLsl    Dc.B    $03,'LSL',2,9,10,11
        Dc      BXp+BNp+BZp+BV0+BCp
        Dc      M1,M1+A_BWL
        Dc      M12,M1+A_BWL
        Dc      PasM,TouM-M1-M2-M10-⌡
                M11-M12+A_W
xLsr    Dc.B    $03,'LSR',2,9,10,11
        Dc      BXp+BNp+BZp+BV0+BCp
        Dc      M1,M1+A_BWL
        Dc      M12,M1+A_BWL
        Dc      PasM,TouM-M1-M2-M10-⌡
                M11-M12+A_W
xMove   Dc.B    $04,'MOVE',0,18+Msg3,-1
        Dc      BXna+BNp+BZp+BV0+BC0
        Dc      TouM,TouM-M2-M10-M11-⌡
                M12+A_BWL
xMovC   Dc.B    $84,'MOVE>C',0,19,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      TouM-M2,PasM+A_W
xMovea  Dc.B    $05,'MOVEA',0,5
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      TouM,M2+A_W+A_L
xMovU   Dc.B    $84,'MOVE_U',1,20,21
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,M2+A_L
        Dc      M2,PasM+A_L
xMovtS  Dc.B    $84,'MOVE<S',0,22,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      PasM,TouM-M2-M10-M11-⌡
                M12+A_W
xMovtS  Dc.B    $84,'MOVE>S',0,23,-1
        Dc      BXp+BNp+BZp+BVp+BCp
        Dc      TouM-M2,PasM+A_W
xMovem  Dc.B    $05,'MOVEM',1,24,25,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
        Dc      M1+M2,M3+M5+M6+M7+⌡
                M8+M9+A_W+A_L
        Dc      TouM-M1-M2-M5-M12,M1+⌡
                M2+A_W+A_L
xMovep  Dc.B    $05,'MOVEP',1,26,27,-1
        Dc      BXna+BNna+BZna+BVna+⌡
                BCna
```

```
                Dc      M1,M6+A_W+A_L
                Dc      M6,M1+A_W+A_L
xMoveq          Dc.B    $05,'MOVEQ',0,28
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      M12,M1+A_L
xMuls           Dc.B    $04,'MULS',0,3,-1
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      TouM-M2,M1+A_W
xMulu           Dc.B    $04,'MULU',0,3,-1
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      TouM-M2,M1+A_W
xNbcd           Dc.B    $04,'NBCD',0,11,-1
                Dc      BXp+BNi+BZp+BVi+BCp
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_B
xNeg            Dc.B    $03,'NEG',0,11
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_BWL
xNegx           Dc.B    $04,'NEGX',0,11,-1
                Dc      BXp+BNi+BZp+BVi+BCp
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_BWL
xNop            Dc.B    $03,'NOP',0,0
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,PasM
xNot            Dc.B    $03,'NOT',0,11
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_BWL
xOr             Dc.B    $02,'OR',1,3,4
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      TouM-M2,M1+A_BWL
                Dc      M1,TouM-M1-M2-M10-M11|
                        -M12+A_BWL
xOri            Dc.B    $03,'ORI',0,6
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      M12,TouM-M2-M10-M11-|
                        M12+A_BWL
xOriC           Dc.B    $83,'ORI>C',0,7
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      M12,PasM+A_B
xOriS           Dc.B    $83,'ORI>S',0,6
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      M12,PasM+A_W
xPea            Dc.B    $03,'PEA',0,11
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,TouM-M1-M2-M4-M5|
                        -M12+A_L
xReset          Dc.B    $05,'RESET',0,0
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,PasM
xRol            Dc.B    $03,'ROL',2,9,10,11
                Dc      BXna+BNp+BZp+BV0+BCp
                Dc      M1,M1+A_BWL
                Dc      M12,M1+A_BWL
                Dc      PasM,TouM-M1-M2-M10-|
                        M11-M12+A_W
xRor            Dc.B    $03,'ROR',2,9,10,11
                Dc      BXna+BNp+BZp+BV0+BCp
                Dc      M1,M1+A_BWL
                Dc      M12,M1+A_BWL
                Dc      PasM,TouM-M1-M2-M10-|
                        M11-M12+A_W
xRoxl           Dc.B    $04,'ROXL',2,9,10,11,-1
                Dc      BXp+BNp+BZp+BV0+BCp
                Dc      M1,M1+A_BWL
                Dc      M12,M1+A_BWL
                Dc      PasM,TouM-M1-M2-M10-|
                        M11-M12+A_W
xRoxr           Dc.B    $04,'ROXR',2,9,10,11,-1
                Dc      BXp+BNp+BZp+BV0+BCp
                Dc      M1,M1+A_BWL
                Dc      M12,M1+A_BWL
                Dc      PasM,TouM-M1-M2-M10-|
                        M11-M12+A_W
xRte            Dc.B    $03,'RTE',0,0
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      PasM,PasM
xRtr            Dc.B    $03,'RTR',0,0
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      PasM,PasM
xRts            Dc.B    $03,'RTS',0,0
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,PasM
xSbcd           Dc.B    $04,'SBCD',1,1,2
                Dc      BXp+BNi+BZp+BVi+BCp
                Dc      M1,M1+A_B
                Dc      M5,M5+A_B
xScc            Dc.B    $03,'Scc',0,11
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_B
xStop           Dc.B    $04,'STOP',0,29,-1
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      PasM,M12
xSub            Dc.B    $03,'SUB',1,3+Msg3,4,-1
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      TouM,M1+A_BWL
                Dc      M1,TouM-M1-M2-M10-M11|
                        -M12+A_BWL
xSuba           Dc.B    $04,'SUBA',0,5,-1
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      TouM,M2+A_W+A_L
xSubi           Dc.B    $04,'SUBI',0,6,-1
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      M12,TouM-M2-M10-M11-|
                        M12+A_BWL
xSubq           Dc.B    $04,'SUBQ',0,6+Msg1,-1
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      M12,TouM-M10-M11-M12|
                        +A_BWL
xSubx           Dc.B    $04,'SUBX',1,1,2
                Dc      BXp+BNp+BZp+BVp+BCp
                Dc      M1,M1+A_BWL
                Dc      M5,M5+A_BWL
xSwap           Dc.B    $04,'SWAP',0,16,-1
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      PasM,M1+A_W
xTas            Dc.B    $03,'TAS',0,11
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_B
xTrap           Dc.B    $04,'TRAP',0,30,-1
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,M12
xTrapv          Dc.B    $05,'TRAPV',0,0
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,PasM
xTst            Dc.B    $03,'TST',0,11
                Dc      BXna+BNp+BZp+BV0+BC0
                Dc      PasM,TouM-M2-M10-M11-|
                        M12+A_BWL
xUnlk           Dc.B    $04,'UNLK',0,31,-1
                Dc      BXna+BNna+BZna+BVna+|
                        BCna
                Dc      PasM,M2

; Définitions diverses
Icons           Dc.L    $00000000,$00000050
                Dc.L    $00000080,$00001100
                Dc.L    $000008AA,$00000155
                Dc.L    $000000AA,$00001401
                Dc.L    $00002A02,$00004514
                Dc.L    $0000A280,$00015140
                Dc.L    $000228A0,$00051440
                Dc.L    $00028A00,$00014560
                Dc.L    $0028A2A0,$00545140
                Dc.L    $008A2980,$00453D7C
                Dc.L    $00A2FFD6,$1451FEAB
                Dc.L    $2A28FD45,$4515FAA3
                Dc.L    $A20AFD45,$5100FAA3
                Dc.L    $2A007D46,$14007AAA
                Dc.L    $0A003D54,$05001AA8
                Dc.L    $02800DD0,$01000660

Poms            Dc.B    252,124,128,179,224
                Dc.B    62,62,7,243,3
                Dc.B    192,192,12,124
                Dc.B    198,198,193,182,16
                Dc.B    67,99,6,7,4
                Dc.B    97,128,24,134
                Dc.B    198,198,227,166,0
                Dc.B    3,99,7,227,0
                Dc.B    99,0,48,6
                Dc.B    198,198,213,131,224
                Dc.B    14,63,0,51,0
                Dc.B    198,0,96,28
                Dc.B    252,198,201,128
                Dc.B    48,3,3,0,51
                Dc.B    1,140,192,204,6
                Dc.B    192,198,193,132,48
                Dc.B    67,67,4,51,3
                Dc.B    15,224,254,134
                Dc.B    192,124,193,131,224
                Dc.B    62,62,35,227,39
                Dc.B    224,196,12,124

CadreFenetre    Dc      40,3,340,284
RectMnemo       Dc      1,2,253,162
RectSyntaxes1   Dc      256,16,284,230
RectSyntaxes2   Dc      284,16,298,223
RectControles   Dc      254,1,298,15
RectDest        Dc      267,223,299,255
RectDestPoms    Dc      214,168,221,279
PasTitre        Dc      0

Curseur1
                Dc      $0020,$0050,$0048,$0024
                Dc      $7FF2,$8009,$7FC9,$0825
                Dc      $07C3,$0421,$03C3,$01FE
                Dc      $0000,$0000,$0000,$0000
                Dc      $0020,$0070,$0078,$003C
                Dc      $7FFE,$FFFF,$7FFF,$0FFF
                Dc      $07FF,$07FF,$03FF,$01FE
                Dc      $0000,$0000,$0000,$0000
                Dc      5.0

Curseur2
                Dc      $00C0,$0140,$0180,$3D7C
                Dc      $7FD6,$FEAB,$FD45,$FAA3
                Dc      $FD45,$FAA3,$7D46,$7AAA
                Dc      $3D54,$1AA8,$0DD0,$0660
                Dc      $00C0,$01C0,$0180,$3D7C
                Dc      $7FFE,$FFFF,$FFFF,$FFFF
                Dc      $FFFF,$FFFF,$7FFE,$7FFE
                Dc      $3FFC,$1FF8,$0FF0,$0660
                Dc      0,8
```

Ces deux petits programmes, qui ont été utilisés pour la mise en pages du numéro 26 (titre de l'article 'Lissajous' et Éditorial), montre qu'il est possible, grâce à la définition de l'affichage du Macintosh, d'obtenir des images de qualité avec des fonctions trigonométriques simples.

Bien qu'ils ne soient pas très longs, les deux programmes se trouvent, comme à l'accoutumée, sur la disquette d'accompagnement de ce numéro. Rappelons qu'il n'est pas nécessaire d'avoir le Basic Microsoft pour utiliser les programmes Basic publiés dans Pom's : une version 'RUNTIME' de la version 2.00 se trouve sur chaque disquette Pom's depuis le numéro 23.

## Routine pour 'noircir' la barre des menus

*(incorporée aux programmes Basic)*

```
                ScrnBase    EQU     $824
                ScreenRow   EQU     $106
2078 0824               MOVEA.L ScrnBase,A0
7013                    MOVEQ   #19,D0
3238 0106       B0      MOVE    ScreenRow,D1
10FC 00FF       B1      MOVE.B  #$FF,(A0)+
5341                    SUBQ    #1,D1
66F8                    BNE.S   B1
51C8 FFF2               DBRA    D0,B0
4E75                    RTS
```

# Courbes

*Marianne Sutz*

## Programme 'Courbe1'

```
DEFINT A-Z:DIM C(15):PI!=3.1416:F
   OR I=0 TO 15:READ C(I):NEXT:W
   INDOW 1,"",(0,20)-(512,342),3:
   HIDECURSOR:BACKPAT VARPTR(C
   (12)):A!=VARPTR(C(0)):A!:CLS:P
   ENMODE 10
DATA &h2078,&h824,&h7013,&h3238,&h
   106,&h10FC,&hFF,&h5341,&h66F8,&h
   51C8,&hFFF2,&h4E75,-1,-1,-1,-1
FOR I!=0 TO PI!*2 STEP .001:X=256
   +1.5*(90*COS(I!)-110*SIN(I!)):
   Y=150+.7*(80*SIN(8*I!)-(120*CO
   S(I!))):IF C THEN PRESET(X,Y)
   :C=C-1 ELSE MOVETO X,Y:LINET
   O 256,190:C=9
NEXT:BEEP:WHILE NOT MOUSE(0):W
   END:MENU RESET:SHOWCURSOR
```

## Programme 'Courbe2'

```
DEFINT A-Z:DIM A!(520),C(15):PI!=
   3.1416:FOR I=0 TO 15:READ C(I)
   :NEXT:WINDOW 1,"",(0,20)-(512,
   342),3:HIDECURSOR:BACKPAT VA
   RPTR(C(12)):A!=VARPTR(C(0)):A!
   :CLS
DATA &h2078,&h824,&h7013,&h3238,&h
   106,&h10FC,&hFF,&h5341,&h66F8,&h
   51C8,&hFFF2,&h4E75,-1,-1,-1,-1
FOR X=0 TO 255:IF X MOD 3 THEN N
   =3 ELSE N=1
Y!=20*SIN(X*3*PI!/256)+X/2:FOR X2=
   0 TO 255 STEP N:Y2!=20*SIN(X2*
   3*PI!/256)+X2/2:IF Y!+Y2!>=A!(2
   60-X+X2) THEN A!(260-X+X2)=Y!+Y
   2!:PRESET(256-X+X2,280-A!(260-
   X+X2))
NEXT:NEXT:BEEP:WHILE NOT MOUSE
   (0):WEND:MENU RESET:SHOWCUR
   SOR
```

*Frédéric Ivsic*

Loupe

# Loupe

LOUPE est une routine écrite en assembleur 6502 qui réalise un effet de zoom sur une page graphique haute résolution. Elle agrandit et affiche en basse résolution une fenêtre extraite de cette image.

La fenêtre superposée à l'image haute résolution se déplace avec les touches traditionnelles I, J, K et M. La vitesse de déplacement peut être modifiée avec les touches 1 à 9.

La touche G permet de passer en basse résolution en recopiant le dessin situé dans la fenêtre, afin d'en étudier les détails. A l'inverse, la touche **H** fait revenir en mode haute résolution.

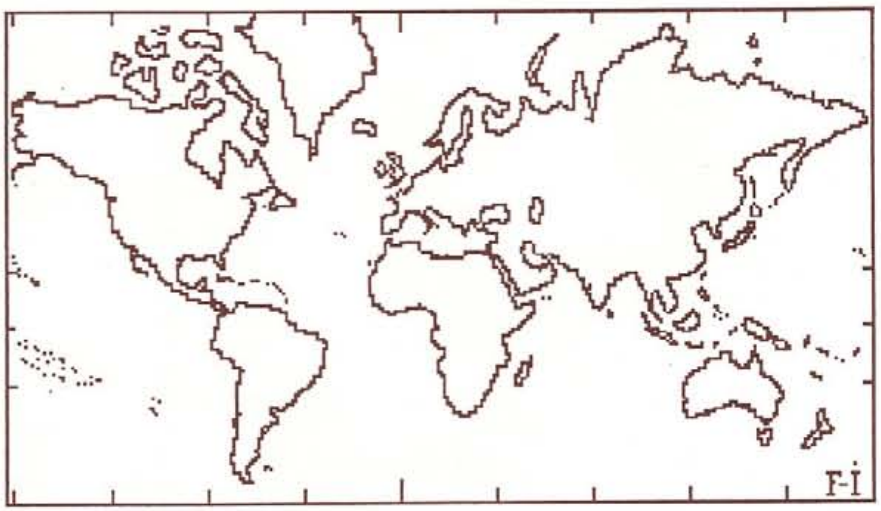
Enfin, **Escape** permet de sortir du programme en revenant au niveau du Basic.

Sur la disquette d'accompagnement, vous trouverez les fichiers :

**LOUPE.S** source en Big Mac
**LOUPE** fichier 6502 exécutable
**LOUPE.DEMO** petit programme de démonstration
**LOUPE.PIC** image graphique utilisée pour la démonstration

Le programme se charge en $7800 et utilise certaines routines de l'interpréteur Applesoft et du moniteur, en particulier HPOS et GBASCALC qui calculent les adresses respectives en haute ou basse résolution d'une ligne donnée.
Les octets 6 à 9 de la page zéro sont également utilisés.

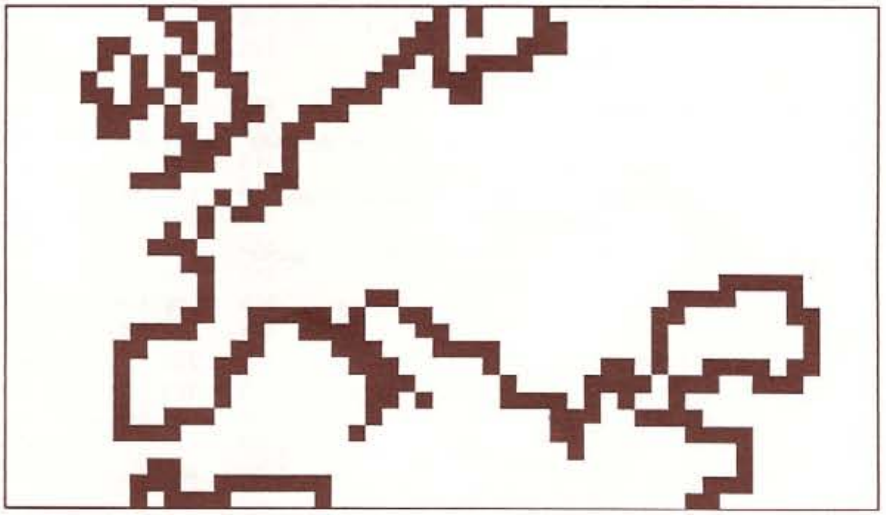
Aucun accès spécifique au DOS 3.3 n'étant utilisé, LOUPE est donc aisément portable vers un environnement ProDOS.




## Programme 'LOUPE.DEMO'

*NB : Pour utiliser ce programme, vous devez avoir sur la disquette une image graphique nommée LOUPE.PIC. Cette image est chargée à la ligne 54.*

```
5  REM  -- DEMO LOUPE --
6  REM
7  REM  (F.IVSIC)
8  REM
9  LOMEM: 16384
10 D$ =  CHR$ (4)
20  PRINT D$"BLOAD LOUPE,A$7800"
30  HOME : HTAB 15: INVERSE : PRINT "DE
   MO-LOUPE": NORMAL
34  PRINT
41  PRINT
43  PRINT "POUR PASSER EN HGR, TAPEZ (H
   )"
44  PRINT "ET DE MEME POUR GR, TAPEZ (G
   )"
45  PRINT : PRINT "
   I"
46  PRINT "LES DEPLACEMENTS: J<   >K"
48  PRINT "                    M"
50  PRINT : PRINT "LA VITESSE DE DEPLAC
   EMENT VERTICALE": PRINT "PEUT VARIER
    DE 1 A 9"
52  PRINT : PRINT "ENFIN, (ESC) POUR FI
   NIR."
53  PRINT : INPUT "TAPEZ RETURN..";R$
54  PRINT D$"BLOAD LOUPE.PIC,A$2000": R
   EM  CHARGE IMAGE HGR
55  REM  -- GR,NON MIXTE ,HTE RES.--
60  POKE 49239,0
62  POKE 49234,0
65  POKE 49232,0
100  CALL 30720
```



# Source 'LOUPE.S'
## Assembleur Big Mac

```
*
************************
**                     *
**   LOUPE HGR->GR     *
**                     *
**     F.IVSIC         *
**                     *
** DOS TOOL KIT        *
************************
*
COORDX   EQU  $6
COORDY   EQU  $7
VARX     EQU  $8
VARY     EQU  $9
HPOS     EQU  $F411
GBASL    EQU  $26
GBASH    EQU  $27
GBASCALC EQU  $F847
HOME     EQU  $FC58
BELL2    EQU  $FBE4
*
LIGHGR2  EQU  $F9        *Dernière ligne fenetre HGR
LIGHGR3  EQU  $FA        *Avant-dernière ligne HGR
LIGHGR   EQU  $FB        *Première ligne HGR
LIGGR    EQU  $FC        *Première ligne GR
OCTHGR   EQU  $FD        *Octet HGR (1[Colonne)
OCTHGR2  EQU  $FE        *Octet HGR (6[Colonne)
OCTGR    EQU  $FF        *Octet GR (1 à 40)
INDOCT   EQU  $CE        *Index Bit dans octet HGR
MASK     EQU  $CF        *Cadre HGR Col. D & G
MIXT     EQU  $D6
VITESSE  EQU  $D7        *Vitesse de déplacement en Y de 1..9
*
*
         ORG  $7800
*
INIT     LDX  #20        *fenetre au
         STX  VARX       *centre écran
         LDX  #96
         STX  VARY
         LDX  #1         *Vitesse = 1
         STX  VITESSE
*
KEY      LDA  VARX
         STA  COORDX     * - transfert de VARX & VARY vers
         LDA  VARY       *   COORDX & COORDY car ces derniers
         STA  COORDY     * sont modifiés après chaque visualisation
         JSR  L1         * par L1 et L2
         JSR  L2
         BIT $C010
         LDA  $C000      * Lect clavier
         PHP
         BIT  $C010      * Indic. touche enfoncée & RAZ
         PLP
         BPL  KEY        *si non, retour a KEY
         AND  #$7F       BIT 7 A ZERO
         TAY
*
* --- BOUCLE DE LECTURE DU CLAVIER ---
*
         CPY  #$1B       * Données = <ESC> ?
         BEQ  FIN          Fin de programme
         CPY  #$47       * = <G> ?
         BNE  A1
         LDX  #00        *Basse résolution
         STX  $C056
         JSR  BELL2
A1       CPY  #$48       * = <H> ?
         BNE  A2
         LDX  #00        *Haute résolutn
         STX  $C057
         JSR  BELL2
         JMP  A7
*
FIN      LDX  #00
         STX  $C051      * Mode text
         JSR  BELL2      *Beeeeeeep!!!
         JSR  HOME       *Efface écran text
         RTS             *retour BASIC
*
* --- Coordonnees Y ---
*
A2       CPY  #$49       * 'I' = haut
         BNE  A3
         SEC
         LDA  VARY
         SBC  VITESSE    * Mvt modulable par increment de 1 a 9
         STA  VARY
A3       CPY  #$4D       * 'M' = bas
         BNE  A4
         CLC
         LDA  VARY
         ADC  VITESSE    * Mvt modulable par increment de 1 a 9
         STA  VARY
*
* --- Coordonnees X ---
*
A4       CPY  #$4A       * 'J' = gauche
         BNE  A5
         SEC             * Ici le mouvement est constant car
         LDA  VARX       * celui-ci s'effectue par pas de 7
         SBC  #1         * bits (1 octet) sur l'écran graphique
         STA  VARX
A5       CPY  #$4B       * 'K' = droite
         BNE  A6
         CLC             * Idem ci-dessus
         LDA  VARX
         ADC  #1
         STA  VARX
*
A6       CPY  #$31       * Y<'1' alors A7
         BCC  A7
         CPY  #$3A       *si Y>=':'(code suivant '9'),alors A7
         BCS  A7
         SEC
         TYA
```

```
117           SBC  #$30        *1..9 = $31..$39
118           STA  VITESSE     *d'ou la soustraction
                de $30
119           JSR  BELL2
120 *
121 A7        JMP  KEY         *Fin de la boucle de
               lecture
122 *
123 * -- SAUT EFFACE CADRE HGR (bis) --
124 *
125 L2        JSR  CADREHGR    *pour effacer le
126           RTS              *cadre HGR après ler
               appel à la routine
127 *
128 * -- CALCUL DE X ET Y 1[ OCTET HGR --
129 *
130 L1        SEC              *principale.
131           LDA  COORDX      *coordonnées X,
132           SBC  #3          *Y octet HGR en
133           STA  COORDX      *haut à gauche
134           SEC              *fenetre HGR de
135           LDA  COORDY      * 6 oct x 24 lig
136           SBC  #12
137           STA  COORDY
138 *
139 * -- TEST LIMITE DE LA FENETRE --
140 *
141           LDA  COORDX      *X entre 0 et 33
142           BMI  S1          *(33=39-6)
143           CMP  #34
144           BCS  S2
145           JMP  S3
146 S1        LDA  #00
147           STA  COORDX
148           BEQ  S3
149 S2        LDA  #34
150           STA  COORDX
151 S3        LDA  COORDY      *Y entr 0 et 167
152           CMP  #243        *(167=191-24)
153           BCS  S4
154           CMP  #167
155           BCS  S5
156           JMP  S6
157 S4        LDA  #00
158           STA  COORDY
159           BEQ  S6
160 S5        LDA  #167
161           STA  COORDY
162 S6        JSR  CADREHGR    *'utilise
163           JMP  TRANSF      *CADREHGR seul pour e
               ffacer
164 *
165 * -- CALCUL PRELIMINAIRE --
166 *
167 CADREHGR  LDA  #22          *Pag HGR pleine
168           STA  MIXT
169 M2        LDA  COORDY      *Initialise
170           STA  LIGHGR      *lere ligne HGR
171           CLC
172           ADC  MIXT
173           STA  LIGHGR3     * Avant-dernière lign
               e
174           ADC  #1
175           STA  LIGHGR2     *Dernière ligne
176           LDA  COORDX
177           STA  OCTHGR      *  ler octet
178           CLC
179           ADC  #5
180           STA  OCTHGR2     *6ème octet (fin de 1
               igne)
181           CLC
182           LDA  #00
183           STA  LIGGR       *  ler ligne GR
184           STA  OCTGR       *  ler octet GR
185           STA  INDOCT      *  Index = 0
186           LDA  #$20        *$20 dans $E6 =
187           STA  $E6         *page 1 ds HPOS
188 *
189 * -- DESSIN CADRE EN HGR --
190 *
191           LDA  LIGHGR      * 1ère ligne
192           LDX  #00
193           LDY  #00
194           JSR  HPOS
195           JSR  LIGNE
196 *
197           LDA  LIGHGR2     *Dernière ligne
198           LDX  #00
199           LDY  #00
200           JSR  HPOS
201           JSR  LIGNE
202           JMP  SUITE
203 *
204 LIGNE     LDY  OCTHGR      * Trace trait en
205 LIG1      LDA  (GBASL),Y   * couleur complémenta
               ire
206           EOR  #$7F
207           STA  (GBASL),Y
208           CPY  OCTHGR2
209           BEQ  LIG2
210           INY
211           BNE  LIG1
212 LIG2      RTS
213 *
214 SUITE     LDA  #1          * Colonne de gauche a
               vec
215           STA  MASK        * le masque 0000 0001
216           LDY  OCTHGR
217           JSR  COLONE
218 *
219           LDA  #$40        * Colonne de droite a
               vec
220           STA  MASK        * le masque 0100 0000
221           LDY  OCTHGR2
222           JSR  COLONE
223           RTS
224 *
225 COLONE     LDA  LIGHGR      * Trace colonnes
226 COL1      CLC              * tjs en complt
227           ADC  #1
228           PHA
229           TAX
230           TYA
231           PHA
232           TXA
233           LDX  #00
234           LDY  #00
235           JSR  HPOS
236           PLA
237           TAY
238           LDA  (GBASL),Y
239           EOR  MASK
240           STA  (GBASL),Y
241           PLA
242           CMP  LIGHGR3
243           BEQ  COL3
```

```
244          BNE  COL1
245 COL3     RTS
246 *
247 * -- TRANSF HGR -> GR --
248 *
249 TRANSF   LDY  #00        *1ère ligne GR
250          LDA  #00
251          JSR  CAD1
252          INC  LIGGR
253          INC  LIGHGR
254 *
255 NEWLIG   LDA  LIGHGR     *Adresse ligne
256          LDX  #00        *en cours et chargeme
              nt
257          LDY  #00        *de l'octet HGR
258          JSR  HPOS
259          LDY  OCTHGR
260          LDA  (GBASL),Y
261 BYTE     ROR              ;Rotation, test(bit=1
              ) de
262          PHA              *6 bits affichés de l
              'octet
263          TYA
264          PHA
265          PHP
266          LDA  LIGGR
267          JSR  GBASCALC
268          LDY  OCTGR
269          LDA  #$FF        *Adresse en GR et tra
              nsfert
270          CPY  #00
271          BEQ  S14
272          CPY  #39
273          BEQ  S14
274          BCS  S11
275          PLP              *  bit 1 = $FF
276          BCS  S10         *  bit 0 = $00
277          LDA  #00
278 S10      STA  (GBASL),Y
279 S12      PLA
280          TAY
281          PLA
282          LDX  INDOCT      *Décompte de 7 bits d
              'un octet HGR
283          CPX  #7
284          BEQ  S8
285          INC  INDOCT
286          INC  OCTGR
287          BNE  BYTE
288 S8       LDX  #00         *Passage à l'octet HG
              R suivant sur la ligne
289          STX  INDOCT
290          LDX  OCTHGR
291          CPX  OCTHGR2
292          BEQ  S9
293          INC  OCTHGR
294          BNE  NEWLIG
295 S9       LDX  #00         *ligne suivante
296          STX  OCTGR
297          LDX  COORDX
298          STX  OCTHGR
299          LDX  LIGGR
300          CPX  MIXT
301          BEQ  CADREGR
302          INC  LIGGR
303          INC  LIGHGR
304          BNE  NEWLIG
305 *
306 S11      PLP
307          JMP  S12
308 *
309 S14      PLP
310          LDA  #$66
311          JMP  S10
312 *
313 CADREGR  LDY  #00         *Dernière ligne GR en
              trait plein
314          LDX  MIXT
315          INX
316          TXA
317          JSR  CAD1
318          RTS
319 *
320 CAD1     JSR  GBASCALC    *trace en plein la li
              gne de n° A
321 CAD2     LDA  #$66
322          STA  (GBASL),Y
323          CPY  #39
324          BEQ  CAD3
325          INY
326          BNE  CAD2
327 CAD3     RTS
```

## Récapitulation 'LOUPE'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE LOUPE,A$7800,L$1E7*

```
7800- A2 14 86 08 A2 60 86 09
7808- A2 01 86 D7 A5 08 85 06
7810- A5 09 85 07 20 96 78 20
7818- 92 78 AD 00 C0 08 2C 10
7820- C0 28 10 E8 29 7F A8 C0
7828- 1B F0 1B C0 47 D0 08 A2
7830- 00 8E 56 C0 20 E4 FB C0
7838- 48 D0 17 A2 00 8E 57 C0
7840- 20 E4 FB 4C 8F 78 A2 00
7848- 8E 51 C0 20 E4 FB 20 58
7850- FC 60 C0 49 D0 07 38 A5
7858- 09 E5 D7 85 09 C0 4D D0
7860- 07 18 A5 09 65 D7 85 09
7868- C0 4A D0 07 38 A5 08 E9
7870- 01 85 08 C0 4B D0 07 18
7878- A5 08 69 01 85 08 C0 31
7880- 90 0D C0 3A B0 09 38 98
7888- E9 30 85 D7 20 E4 FB 4C
7890- 0C 78 20 D6 78 60 38 A5
7898- 06 E9 03 85 06 38 A5 07
78A0- E9 0C 85 07 A5 06 30 07
78A8- C9 22 B0 09 4C B9 78 A9
78B0- 00 85 06 F0 04 A9 22 85
78B8- 06 A5 07 C9 F3 B0 07 C9
78C0- A7 B0 09 4C D0 78 A9 00
78C8- 85 07 F0 04 A9 A7 85 07
78D0- 20 D6 78 4C 5C 79 A9 16
78D8- 85 D6 A5 07 85 FB 18 65
78E0- D6 85 FA 69 01 85 F9 A5
78E8- 06 85 FD 18 69 05 85 FE
78F0- 18 A9 00 85 FC 85 FF 85
78F8- CE A9 20 85 E6 A5 FB A2
7900- 00 A0 00 20 11 F4 20 18
7908- 79 A5 F9 A2 00 A0 00 20
7910- 11 F4 20 18 79 4C 28 79
7918- A4 FD B1 26 49 7F 91 26
7920- C4 FE F0 03 C8 D0 F3 60
7928- A9 01 85 CF A4 FD 20 3B
7930- 79 A9 40 85 CF A4 FE 20
7938- 3B 79 60 A5 FB 18 69 01
7940- 48 AA 98 48 8A A2 00 A0
7948- 00 20 11 F4 68 A8 B1 26
7950- 45 CF 91 26 68 C5 FA F0
7958- 02 D0 E2 60 A0 00 A9 00
7960- 20 D8 79 E6 FC E6 FB A5
7968- FB A2 00 A0 00 20 11 F4
7970- A4 FD B1 26 6A 48 98 48
7978- 08 A5 FC 20 47 F8 A4 FF
7980- A9 FF C0 00 F0 42 C0 27
7988- F0 3E B0 38 28 B0 02 A9
7990- 00 91 26 68 A8 68 A6 CE
7998- E0 07 F0 06 E6 CE E6 FF
79A0- D0 D2 A2 00 86 CE A6 FD
79A8- E4 FE F0 04 E6 FD D0 B7
79B0- A2 00 86 FF A6 06 86 FD
79B8- A6 FC E4 D6 F0 10 E6 FC
79C0- E6 FB D0 A3 28 4C 93 79
79C8- 28 A9 66 4C 91 79 A0 00
79D0- A6 D6 E8 8A 20 D8 79 60
79D8- 20 47 F8 A9 66 91 26 C0
79E0- 27 F0 03 C8 D0 F5 60
```

## *Une nouvelle disquette Pom's :* Ordico

*Apple ][+, IIe, IIe+, IIc*

Destiné aux amateurs de mots croisés ou de Scrabble, cette base de données, due à Roland Jost, permet de trouver un mot de longueur donnée dont on ne connaît que quelques lettres.

Ordico contient plus de 15000 mots classés en 70 rubriques.
Recherches et affichages sont rapides : un fichier de 1500 mots est chargé en moins de 10 secondes et exploité quasi-instantanément.
Il est bien sûr possible d'ajouter des termes aux divers fichiers, de créer de nouvelles rubriques.

Voici quelques rubriques :

*1ère face :*

*Acteurs, Animaux, Armes/guerres, Auteurs américains, Auteurs anglais, Auteurs français, Chimie, Cinéastes, Coureurs cyclistes, Départements/régions, Dieux/déesses, Familles végétales, Femmes célèbres, Hommes politiques, Iles, Jeux/sports, Minéraux, Montagnes, Musiciens jazz, Musiciens, Parties du corps, Peintres étrangers, Peintres français, Rivières/fleuves, Saints/saintes, Savants/inventeurs, Sculpteurs, végétaux, Vêtements, Villes*

*2ème face :*

*Athlètes, Boxeurs, Cantatrices, Cols, Cosmonautes, Coureurs automobiles, Déserts, Détroits, Doctrines philosophiques, Drogues, Escrimeurs, Explorateurs, Gymnastes, Haltérophiles, Judokas, Lutteurs, Maladies, Maréchaux de France, Médicaments, Nageurs, Patineurs, Poissons, Présidents américains, Skieurs, Ski nordique, Unités, Villes olympiques*

---

Exemples :
Un musicien dont le nom comporte 7 lettres, les 2ème et 5ème sont des 'E'. Tapez : –E– –E– –. Vous obtenez instantanément :

*BENNETT DEBOECK DELEEUW GEVAERT LESUEUR PEDRELL PEETERS WELLESZ*

Dans les acteurs, –A– – – –E vous donnerait :

*CARETTE PALANCE RACETTE RANDONE RAYMONE VALLONE*

et – – – –H– :

*CAUCHY CEECHI ENIGHT VAUGHN WRIGHT*

– – – – – – – dans les femmes célèbres donnerait 32 noms...

---

**Disquette double face et documentation : 200,00 F franco. Bon de commande page 74**

# CALLRWTS

*Patrice Neveu*

CALLRWTS est un petit module en assembleur qui facilite les appels à RWTS. Pour les profanes, RWTS est le point d'entrée du DOS 3.3 qui permet d'accéder directement à une disquette, soit en lecture, soit en écriture (les fonctions de formatage ne sont pas exploitées).

Ce programme s'adresse à tous les débutants désireux de se familiariser avec la structure interne des disquettes, et qui ne disposent pas d'un éditeur de secteurs.

Son utilisation est très simple :

*BRUN CALLRWTS*

charge la routine et initialise le vecteur de l'ampersand qui est utilisé pour tous les appels.
L'utilisation s'étend à tout Apple ][, ][+, //e ou //c exploité sous DOS 3.3.
Les nouvelles fonctions disponibles sous Applesoft sont alors :

**& DISK** (slot,drive,volume)
définition de la disquette de travail

**& RSEC** (piste,secteur,adresse)
lecture d'un secteur

**& WSEC** (piste,secteur,adresse)
écriture d'un secteur

**& RTRK** (piste,adresse)
lecture d'une piste complète

**& WTRK** (piste,adresse)
écriture d'une piste complète

Ici "adresse" signifie l'adresse en décimal de début du buffer utilisé : 256 octets pour un secteur, 4096 octets pour une piste complète.

**& BELL**
**& BEL1** : deux fonctions envoyant un "beep" différent.

**& REPT** (nombre,"chaîne")
fonction répétitive d'une chaîne de caractères, indiquée soit directement, soit sous forme d'une variable alphanumérique.

Le source CALLRWTS.S est modifiable par l'assembleur Big Mac. Le module objet se charge en $9000 et fixe Himem à cette valeur, mais son adresse d'implantation peut être éventuellement modifiée par un nouvel assemblage.



## Source 'CALLRWTS.SCE'
### Assembleur Big Mac

```
 1 ****************************
 2 *                          *
 3 *  INSTRUCTIONS AMPERS     *
 4 *                          *
 5 *                          *
 6 * COPYRIGHT OCTOBRE 1984   *
 7 *                          *
 8 ****************************
 9 *                          *
10 *FONCTIONS: - REPT         *
11 *           - DISK         *
12 *           - RSEC         *
13 *           - WSEC         *
14 *           - RTRK         *
15 *           - WTRK         *
16 *           - BELL         *
17 *           - BEL1         *
18 *                          *
19 *                          *
20 ****************************
21 *                          *
22 *         BIG MAC          *
23 *                          *
24 ****************************
25 *
26 *
27 *
28 ***************
29 * IMPORTANT:  *
30 ***************
31 *
32 *
33 * NE PAS MODIFIER LE PROGRAMME
34 * SOUS PEINE DE NE PLUS FONCTIONNER
35 * CAR IL PRESENTE DES ADRESSES
36 * RESERVEES
37 *
38 *
39 *
40 *
41 *
42 HIMEM    =    $73        ;FIXE HIMEM
43 A1L      =    $3C        ;SAUVEGARDE DE
                              TRANSMISSION
44 A1H      =    $3D        ;DE PARAMETRES
45 A4L      =    $42        ;ZONE DE SAUVEGARDE
```

```
46 A4H      =   $43        ;AVANT L'APPEL DU
                 SOUS-PROGRAMME
47 LINNUM   =   $50        ;REGISTRE 16 BITS
48 VARPNT   =   $83        ;ADRESSE DE LA
                 DERNIERE VARIABLE
49 CHRGET   =   $B1        ;PROCHAIN CARACTERE
50 RWTS     =   $3D9       ;LECTURE/ECRITURE D'UN
                 SECTEUR
51 PTRGET   =   $DFE3      ;RESULTAT DE GETARYPT
52 FRMEVL   =   $DD76      ;EVALUATION DE TXTPTR
53 FRMNUM   =   $DD67      ;EVALUATION DE TXTPTR
                 ET RANGEMENT DANS LA FAC
54 CHKOPN   =   $DEBB      ;TXTPTR POINTE T'IL
                 SUR (
55 CHKCLS   =   $DEB8      ;TXTPTR POINTE T'IL
                 SUR )
56 CHKCOM   =   $DEBE      ;TXTPTR POINTE T'IL
                 SUR ,
57 CONINT   =   $E6FB      ;CONVERTIT LE FAC EN
                 UN SEUL OCTET
58 AYIN     =   $E199      ;REND ENTIER LE FAC
59 GETADR   =   $E752      ;TRANSFORME FAC EN
                 ENTIER A 2 OCTETS
60 OUTDO    =   $DB5C      ;AFFICHE
                 L'ACCUMULATEUR A
61 STROUT   =   $DB3A      ;AFFICHE LA CHAINE
                 POINTEE PAR (Y,A)
62 TXTPTR   =   $B8        ;ADRESSE DU DERNIER
                 CARACTERE OBTENU PAR CHRGET
63 CMD      =   $19
64 AMPER    =   $3F5       ;ADRESSE DE &
65 SNUM16   =   $B7E9      ;ADRESSE NUMERO DE
                 SLOT
66 DNUM     =   $B7EA      ;ADRESSE NUMERO DE
                 DRIVE
67 ADDON    =   $D998      ;AJOUTE 1 AU REGISTRE
                 Y
68 SNTX     =   $DEC9      ;SYNTAX ERROR
69 PRBYTE   =   $FDDA      ;AFFICHE A EN DEUX
                 CHIFFRES HEXA
70 SPKR     =   $C030      ;STIMULE LE HAUT
                 PARLEUR
71 VAR      =   $0478
72 SETTRK   =   $BE95      ;LECTURE ALLOCATION
                 DES PISTES
73 RWLANG   =   $C088
74 *
75 *
76          ORG  $9000
77 *
78 *
79 ************************
80 *  INITIALISE  &      *
81 ************************
82 *
83 *
84 ENTR     LDA  #<INTERPR
85          STA  AMPER+1
86          LDA  #>INTERPR
87          STA  AMPER+2
88 *
89 ***************************
90 *       FIXE  HIMEM       *
91 ***************************
92 *
93          LDA  #$00        ;ADRESSE BASSE
94          STA  HIMEM
95          LDA  #$90        ;ADRESSE HAUTE
96          STA  HIMEM+1
97 *
98 ***************************
99 *   MEMORISE SLOT & DRIVE*
100 ***************************
101 *
102         LDA  SNUM16      ;DERNIER NUMERO DE
                  SLOT
103         STA  NUMSLT
104         LDA  DNUM        ;DERNIER DRIVE UTILISE
105         STA  NUMDRV
106         RTS
107 *
108 *
109 *
110 ************************
111 * INTERPRETATION CMDE   *
112 ************************
113 *
114 *
115 INTERPR  LDX  #$00
116         STX  CMD
117 INTER1   LDY  #$00
118 LOOP1    LDA  CTBL,X
119         BEQ  TROUVE
120         CMP  #$FF
121         BEQ  ERREUR
122         CMP  (TXTPTR),Y
123         BNE  SUIVANT
124         INY
125         INX
126         BNE  LOOP1
127 ERREUR   JMP  SNTX
128 SUIVANT  INX
129         LDA  CTBL,X
130         BNE  SUIVANT
131         INX
132         INC  CMD
133         BNE  INTER1
134 TROUVE   JSR  ADDON
135         ASL  CMD
136         LDX  CMD
137         LDA  ATBL+1,X
138         PHA
139         LDA  ATBL,X
140         PHA
141         RTS
142 *
143 *
144 ************************
145 * TABLE DES COMMANDES *
146 ************************
147 *
148 *
149 CTBL      ASC  'REPT'
150         HEX  00
```

## Récapitulation 'CALLRWTS'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE CALLRWTS,A$9000,L$23C*

```
9000- A9 1F 8D F6 03 A9 90 8D
9008- F7 03 A9 00 85 73 A9 90
9010- 85 74 AD E9 B7 8D 18 91
9018- AD EA B7 8D 19 91 60 A2
9020- 00 86 19 A0 00 BD 54 90
9028- F0 1A C9 FF F0 08 D1 B8
9030- D0 07 C8 E8 D0 EF 4C C9
9038- DE E8 BD 54 90 D0 FA E8
9040- E6 19 D0 DF 20 98 D9 06
9048- 19 A6 19 BD 7E 90 48 BD
9050- 7D 90 48 60 52 45 50 54
9058- 00 44 49 53 4B 00 52 53
9060- 45 43 00 57 53 45 43 00
9068- 52 54 52 4B 00 57 54 52
9070- 4B 00 42 45 4C 4C 00 42
9078- 45 4C 31 00 FF E3 90 35
```

```
151           ASC  'DISK'
152           HEX  00
153           ASC  'RSEC'
154           HEX  00
155           ASC  'WSEC'
156           HEX  00
157           ASC  'RTRK'
158           HEX  00
159           ASC  'WTRK'
160           HEX  00
161           ASC  'BELL'
162           HEX  00
163           ASC  'BEL1'
164           HEX  00
165           HEX  FF          ;FF = FIN DE LA TABLE
166 *
167 *
168 ATBL      DA   REPT-1
169           DA   DISK-1      ;ADRESSE FONCTION DISK
170           DA   RSEC-1      ;ADRESSE DE READ
                     SECTOR
171           DA   WSEC-1      ;ADRESSE DE WRITE
                     SECTOR
172           DA   RTRK-1      ;ADRESSE DE READ TRACK
173           DA   WTRK-1      ;ADRESSE DE WRITE
                     TRACK
174           DA   BELL-1      ;SON DE CLOCHE
175           DA   BEL1-1      ;SON DE CLOCHE GRAVE
176 *
177 *
178 FINTBLE    HEX  00          ;FIN DE LA TABLE DES
                     COMMANDES
179 ZEROA1L    LDA  #$00
180           STA  A1L
181           JSR  $00B7
182           CMP  #$22
183           BNE  ECTXT
184           INC  TXTPTR
185           BNE  MAXSLT
186           INC  TXTPTR+1
187 MAXSLT    LDA  TXTPTR
188           STA  A4L
189           LDA  TXTPTR+1
190           STA  A4H
191           LDY  #$00
192 RGESLT    LDA  (TXTPTR),Y
193           CMP  #$22
194           BEQ  VAR3
195           INY
196           BNE  RGESLT
197 VAR3      STY  A1L
198           CLC
199           TYA
200           ADC  TXTPTR
201           STA  TXTPTR
202           LDA  #$00
203           ADC  TXTPTR+1
204           STA  TXTPTR+1
205           JMP  CHRGET
206 ECTXT     JSR  PTRGET
207           LDA  $81
208           BMI  FRMEV
209           LDA  $82
210           BMI  MZERO
211 FRMEV     JMP  FRMEVL
212 MZERO     LDY  #$00
213           LDA  (VARPNT),Y
214           STA  A1L
215           INY
216           LDA  (VARPNT),Y
217           PHA
218           INY
219           LDA  (VARPNT),Y
220           STA  A4H
221           PLA
222           STA  A4L
223           RTS
224 *
225 ***************************
226 * POINT D'ENTREE DE REPT *
227 ***************************
228 *
229 *
230 REPT      JSR  CHKOPN
231           JSR  FRMNUM
232           JSR  CONINT
233           STX  FINTBLE
234           JSR  CHKCOM
235           JSR  ZEROA1L
236           LDA  A1L
237           BEQ  FCHKCLS
238           LDA  FINTBLE
239           BEQ  FCHKCLS
240 REPT0     LDY  #$00
241           LDA  A1L
242           STA  A1H
243 STREPT    LDA  (A4L),Y
244           JSR  OUTDO
245           INY
246           DEC  A1H
247           BNE  STREPT
248           DEC  FINTBLE
249           BNE  REPT0
250 FCHKCLS   JMP  CHKCLS
251
252
253 ********************
254 *  TABLE I.O.B.    *
255 ********************
256
257
258 DEBUT     HEX  01          ;PARAMETRE DOIT ETRE A
                     01
259 NUMSLT    HEX  00
260 NUMDRV    HEX  00
261 VOLUME    HEX  00
262 PISTE     HEX  00
263 SECTEUR   HEX  00
264 TABLE     HEX  28          ;ADRESSE BASSE
265           HEX  91          ;ADRESSE HAUTE
266 BUFF      HEX  00
267 BUFF1     HEX  00
268           HEX  00
269           HEX  00
```

```
9080- 91 B5 91 C7 91 EA 91 F1
9088- 91 05 92 20 92 00 A9 00
9090- 85 3C 20 B7 00 C9 22 D0
9098- 2A E6 B8 D0 02 E6 B9 A5
90A0- B8 85 42 A5 B9 85 43 A0
90A8- 00 B1 B8 C9 22 F0 03 C8
90B0- D0 F7 84 3C 18 98 65 B8
90B8- 85 B8 A9 00 65 B9 85 B9
90C0- 4C B1 00 20 E3 DF A5 81
90C8- 30 04 A5 82 30 03 4C 76
90D0- DD A0 00 B1 83 85 3C C8
90D8- B1 83 48 C8 B1 83 85 43
90E0- 68 85 42 60 20 BB DE 20
90E8- 67 DD 20 FB E6 8E 8D 90
90F0- 20 BE DE 20 8E 90 A5 3C
90F8- F0 1A AD 8D 90 F0 15 A0
9100- 00 A5 3C 85 3D B1 42 20
9108- 5C DB C8 C6 3D D0 F6 CE
9110- 8D 90 D0 EB 4C B8 DE 01
9118- 00 00 00 00 00 28 91 00
9120- 00 00 00 00 00 00 60 00
9128- 00 01 EF D8 A0 17 A9 91
9130- 4C D9 03 00 00 00 20 BB
9138- DE 20 67 DD 20 FB E6 E0
```

```
MODE     HEX  00
CODERR   HEX  00
         HEX  00
         RTS
         HEX  00

******************
* TABLE DEVICE   *
******************


         HEX  00
         HEX  01
         HEX  EF
         HEX  D8

PARMS    LDY  #$17
         LDA  #$91
         JMP  RWTS
BUFF4    HEX  00
BUFF5    HEX  00
BUFFB    HEX  00

****************************
* POINT D'ENTREE DE DISK   *
****************************
*
*
DISK     JSR  CHKOPN
         JSR  FRMNUM
         JSR  CONINT
         CPX  #$08
         BCC  RETN
FAC      JMP  AYIN
RETN     STX  BUFF4
         TXA
         ASL
         ASL
         ASL
         ASL
         STA  NUMSLT
         JSR  CHKCOM
         JSR  FRMNUM
         JSR  CONINT
         CPX  #$05
         BCS  FAC
         STX  BUFF5
         STX  NUMDRV
         LDA  #$00
         STA  BUFFB
         STA  VOLUME
         JSR  $00B7
         CMP  #$2C
         BEQ  VDISK
         JMP  CHKCLS
VDISK    JSR  CHKCOM
         JSR  FRMNUM
         JSR  CONINT
         STX  BUFFB
         STX  VOLUME
         JMP  CHKCLS
TESTDISK JSR  CHKOPN
         JSR  FRMNUM
         JSR  CONINT
         STX  PISTE
         JSR  CHKCOM
TXTDISK  JSR  FRMNUM
         JSR  CONINT
         STX  SECTEUR
         JSR  CHKCOM
         JSR  FRMNUM
         JSR  GETADR
         LDA  LINNUM
         STA  BUFF
         LDA  LINNUM+1
         STA  BUFF1
         JMP  CHKCLS
*
*
****************************
* POINT D'ENTREE DE RSEC   *
****************************
*
RSEC     LDA  #$01
         STA  MODE
NTRSEC   JSR  TESTDISK
         JSR  PARMS
         BCC  ERROR
PRERR    JMP  $D412


         HEX  00
ERROR    RTS
*
*
****************************
* POINT D'ENTREE DE WSEC   *
****************************
*
*
WSEC     LDA  #$02
         STA  MODE
         BNE  NTRSEC
NBRESECT LDA  #$0F
         STA  SECTEUR
         CLC
         LDA  BUFF1
         ADC  #$0F
         STA  BUFF1
LECTECRT JSR  PARMS
         BCS  PRERR
         DEC  BUFF1
         DEC  SECTEUR
         BPL  LECTECRT
         RTS
*
****************************
* POINT D'ENTREE DE RTRK   *
****************************
*
```

```
9140- 08 90 03 4C 99 E1 8E 33
9148- 91 8A 0A 0A 0A 0A 8D 18
9150- 91 20 BE DE 20 67 DD 20
9158- FB E6 E0 05 B0 E5 8E 34
9160- 91 8E 19 91 A9 00 8D 35
9168- 91 8D 1A 91 20 B7 00 C9
9170- 2C F0 03 4C B8 DE 20 BE
9178- DE 20 67 DD 20 FB E6 8E

9180- 35 91 8E 1A 91 4C B8 DE
9188- 20 BB DE 20 67 DD 20 FB
9190- E6 8E 1B 91 20 BE DE 20
9198- 67 DD 20 FB E6 8E 1C 91
91A0- 20 BE DE 20 67 DD 20 52
91A8- E7 A5 50 8D 1F 91 A5 51
91B0- 8D 20 91 4C B8 DE A9 01
91B8- 8D 23 91 20 88 91 20 2C

91C0- 91 90 04 4C 12 D4 00 60
91C8- A9 02 8D 23 91 D0 EC A9
91D0- 0F 8D 1C 91 18 AD 20 91
91D8- 69 0F 8D 20 91 20 2C 91
91E0- B0 E1 CE 20 91 CE 1C 91
91E8- 10 F3 60 A9 01 8D 23 91
91F0- D0 05 A9 02 8D 23 91 20
91F8- BB DE 20 97 91 AD 1C 91
```

```
394 RTRK     LDA  #$01
395          STA  MODE
396          BNE  TESTWTRK
397 *
398 ************************
399 * POINT D'ENTREE DE WTRK *
400 ************************
401 *
402 WTRK     LDA  #$02
403          STA  MODE
404 TESTWTRK JSR  CHKOPN
405          JSR  TXTDISK
406          LDA  SECTEUR
407          STA  PISTE
408          JMP  NBRESECT
409 *
410 *
411 *
412 *
413 ***********************
414 *   SON DE CLOCHE     *
415 ***********************
416 *
417 *
418 BELL     LDX  #$FF
419 BELL1    LDA  #$5B
420 BELL2    LDY  #$1B
421 BELL3    DEY
422          BNE  BELL3
423          BIT  SPKR
424          TAY
425 BELL4    DEY
426          BNE  BELL4
427          SBC  #1
428          BEQ  BELL1
429          BIT  SPKR
430          DEX
431          BNE  BELL2
432          RTS
433 *
434 *
435 *
436 BEL1     LDX  #$50
437 BEL2     LDA  #$22
438 BEL3     LDY  #$02
439 BEL4     DEY
440          BNE  BEL4
441          BIT  SPKR
442          TAY
443 BELL5    DEY
444          BNE  BELL5
445          SBC  #1
446          BEQ  BEL2
447          BIT  SPKR
448          DEX
449          BNE  BEL3
450          RTS
451          LST  OFF
```

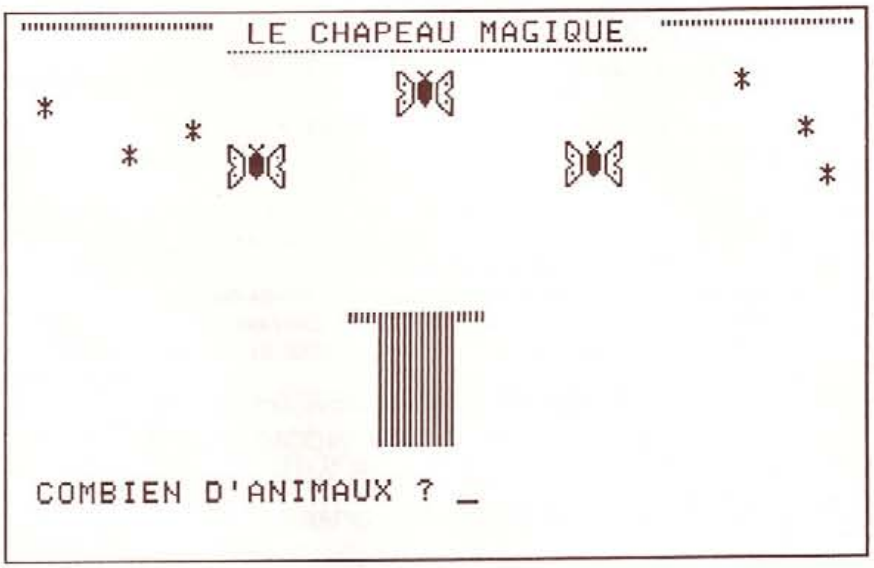
```
9200- 8D 1B 91 4C CF 91 A2 FF
9208- A9 5B A0 1B 88 D0 FD 2C
9210- 30 C0 A8 88 D0 FD E9 01
9218- F0 EE 2C 30 C0 CA D0 EA
9220- 60 A2 50 A9 22 A0 02 88
9228- D0 FD 2C 30 C0 A8 88 D0
9230- FD E9 01 F0 EE 2C 30 C0
9238- CA D0 EA 60
```

# Apprendre à compter avec des animaux

**Alain Idelon-Ritton**

Un jeu éducatif pour les petits, tel est l'objet du présent programme. Il s'agit d'apprendre les chiffres 1 à 9 aux enfants des grandes classes de maternelles et du cours préparatoire. De façon accessoire, il s'agit également de comprendre comment utiliser le module HRCG du Tool-Kit d'Apple.

Le principe est simple : un certain nombre d'animaux, compris entre 1 et 9, sortent d'un chapeau (magique, bien sûr !) ; l'enfant doit les compter et taper le bon chiffre ; un test complet comprend 10 essais. L'idée n'est pas neuve, mais les animaux intéressent les bambins.




Le DOS Tool-Kit d'Apple, qui permet de mixer graphismes et textes, était particulièrement adapté. Les fichiers :

- Animaux.Set (affichage des animaux)
- Gros.Chiffre.Set (affichage des chiffres)
- Bonhomme.Set (affichage du bonhomme MAXWELL)

ont été obtenus avec le programme ANIMATRIX de la disquette Tool-Kit.

Il ne restait plus qu'à les animer grâce à un programme fonctionnant sous l'environnement du module HRCG.

## Mode d'emploi

Avec une disquette contenant les sept fichiers suivants :

```
COMPTER.ANIMAUX
RBOOT
RLOAD
HRCG
ANIMAUX.SET
BONHOMME.SET
GROS.CHIFFRE.SET
```

il suffit de taper :

```
RUN COMPTER.ANIMAUX
```

Pour une bonne exploitation de ce programme, il est préférable qu'un adulte reste près de l'enfant pour le guider, l'aider à utiliser le clavier et reconnaître les touches correspondantes aux chiffres.

*N.D.L.R. : Pour des raisons de copyright, les fichiers RBOOT, RLOAD et HRCG ne figurent pas sur la disquette Pom's. Il vous faudra donc les y transférer, à partir de votre Tool-Kit, pour exécuter le programme.*
*Le programme fonctionne également sans modification sous environnement ProDOS. En revanche, les fichiers ci-dessus devront être extraits du Tool-Kit pour ProDOS diffusé par Apple.*

## Programme 'COMPTER.ANIMAUX'

```
10  REM *** COMPTER.ANIMAUX ***
100  LOMEM: 24576
110  GOSUB 5010
120  HGR : POKE  - 16302,0
130  DIM CO(9,20)
140 NR = 0
150  GOSUB 8010
160  GOSUB 7020
170  GOSUB 4000
180  GOSUB 3000
200  REM *** CORPS PRINCIPAL
210 NN =  INT ( RND (1) * 9) + 1
220 A =  INT ( RND (1) * 9) + 1
230 FR = 0
240  FOR J = 0 TO NN - 1: VTAB 10: HTAB 19: PRINT A
     N$(A):X =  PEEK ( - 16336): FOR K = 1 TO 20: N
```

```
    EXT K: VTAB 10: HTAB 19: PRINT VV$
250 X =  PEEK ( - 16336) +  PEEK ( - 16336) -  PEEK
    ( - 16336) +  PEEK ( - 16336)
260  VTAB CO(NN,2 * J): HTAB CO(NN,2 * J + 1): PRIN
    T AN$(A): NEXT
270  VTAB 20: HTAB 2: PRINT "COMBIEN D'ANIMAUX ? ";
    : POKE  - 16368,0: GET RE$: PRINT RE$
280  IF RE$ < "1" OR RE$ > "9" THEN  PRINT G$: GOTO
     270
290 RE =  VAL (RE$)
300  IF RE <  > NN THEN 340
310  HTAB 2: PRINT CA$"0"CI$"BRAVO"CN$;
320  POKE 769,0: POKE 773,5: POKE 777,1: POKE 790,1
    76: CALL 768
330  FOR I = 1 TO 200: NEXT I: HTAB 2: PRINT "
    ": GOTO 460
340 FR = FR + 1
350 QV =  INT ( RND (1) * 6) + 218: POKE 769,16: PO
    KE 773,3: POKE 777,105: POKE 790,QV: CALL 768
360  IF FR = 1 THEN  GOSUB 1000: GOTO 270
370  IF FR = 2 THEN  HTAB 2: PRINT G$G$G$CI$"OUH OU
    H OUH!!!"CN$;: VTAB 15: HTAB 3: PRINT CH$(NN);
    : FOR I = 1 TO 200: NEXT I: VTAB 21: HTAB 2: P
    RINT "                ": GOTO 270
380  IF FR = 3 THEN  VTAB 15: HTAB 36: PRINT G$CH$(
    NN): GOTO 270
390  IF FR = 4 THEN  VTAB 15: HTAB 7: PRINT G$CH$(N
    N): GOTO 270
400  IF FR = 5 THEN  VTAB 15: HTAB 32: PRINT G$CH$(
    NN): GOTO 270
410  IF FR = 6 THEN  VTAB 15: HTAB 11: PRINT G$CH$(
    NN): GOTO 270
420  IF FR = 7 THEN  VTAB 15: HTAB 28: PRINT G$CH$(
    NN): GOTO 270
430  IF FR = 8 THEN  VTAB 15: HTAB 15: PRINT G$CH$(
    NN): GOTO 270
440  IF FR = 9 THEN  VTAB 15: HTAB 24: PRINT G$CH$(
    NN): GOTO 270
450  IF FR = 10 THEN  VTAB 19: HTAB 21: PRINT G$G$G
    $CH$(NN): FOR K = 1 TO 500: NEXT K: VTAB 19: H
    TAB 21: PRINT VV$
460  FOR J = 0 TO NN - 1: VTAB CO(NN,2 * J): HTAB C
    O(NN,2 * J + 1): PRINT VV$: NEXT
470  FOR I = 1 TO 4: VTAB 15: HTAB 4 * I - 2: PRINT
     VV$: VTAB 15: HTAB 40 - 4 * I: PRINT VV$: NEX
    T I
480 NR = NR + 1
490  IF NR < 10 THEN 210
500  VTAB 3: HTAB 2: PRINT CV$: VTAB 23: HTAB 38: P
    RINT CW$CP$
510  VTAB 5: HTAB 6: PRINT "TAPER  ESC  POUR TERMIN
    ER": VTAB 7: HTAB 18: PRINT "OU"
520  HCOLOR= 1: HPLOT 88,28 TO 114,28 TO 114,42 TO
    88,42 TO 88,28: HPLOT 87,27 TO 115,27 TO 115,4
    3 TO 87,43 TO 87,27
530  VTAB 9: HTAB 6: PRINT "TAPER  R  POUR RECOMMEN
    CER": VTAB 11: HTAB 19
540  HCOLOR= 1: HPLOT 88,60 TO 101,60 TO 101,74 TO
    88,74 TO 88,60: HPLOT 87,59 TO 102,59 TO 102,7
    5 TO 87,75 TO 87,59
550  GET R$: IF  ASC (R$) <  > 27 AND R$ <  > "R" T
    HEN 550
560  IF R$ = "R" THEN NR = 0: PRINT CP$CY$:: GOSUB
    4000: GOTO 210
570  PRINT CO$CB$CY$CP$CO$CA$CY$CP$
580  VTAB 15: HTAB 17: PRINT "AU REVOIR"
590  FOR I = 1 TO 20: VTAB 17: HTAB 20: PRINT BO$(6
    )
600  FOR K = 1 TO 20: NEXT K: VTAB 17: HTAB 20: PRI
    NT BO$(5)
610  FOR K = 1 TO 20: NEXT K: VTAB 17: HTAB 20: PRI
    NT BO$(4): FOR K = 1 TO 20: NEXT K
620  NEXT I
630  VTAB 17: HTAB 20: PRINT BO$(3)
640  POKE 790,144: CALL 768: POKE 790,255: CALL 768
650  TEXT : HOME : PRINT "POUR UTILISER UN AUTRE PR
    OGRAMME": PRINT : HTAB 8: PRINT "EN TOUTE SECU
    RITE": PRINT
660  PRINT "APPUYER SUR RESET PUIS TAPER 'FP'."
670  END
```

## Fichier 'ANIMAUX.SET'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE ANIMAUX.SET,A$8AFF,L$300*

```
8AFF- 00
8B00- 00 00 00 00 00 00 00 00
8B08- 00 00 00 00 00 40 70 00
8B10- 00 00 40 60 20 7F 7F 00
8B18- 00 00 31 71 5B 7F 7F 7C
8B20- 7E 7F 7F 7E 7C 74 14 7F
8B28- 7F 7F 7F 7F 7F 2F 40 6E
8B30- 31 1F 00 00 00 01 02 00
8B38- 00 78 FE 06 FE 78 00 60
8B40- 70 39 1F 0E 8F 3D 70 01
8B48- 00 00 00 00 00 80 00 00
8B50- 01 01 03 06 7C 78 00 00
8B58- 00 00 00 00 00 00 00 00
8B60- 00 00 00 00 00 00 00 00
8B68- 00 00 00 00 00 00 00 00
8B70- 00 00 00 00 00 00 00 00
8B78- 00 00 00 00 00 00 00 00
8B80- 00 00 00 00 00 00 00 00
8B88- 00 00 00 00 00 00 00 00
8B90- 00 00 00 00 00 00 00 00
8B98- 00 00 00 00 00 00 00 00
8BA0- 00 00 00 00 00 00 00 00
8BA8- 00 00 00 00 00 00 00 00
8BB0- 00 00 00 00 00 00 00 00
8BB8- 00 00 00 00 00 00 00 00
8BC0- 00 00 00 00 00 00 00 00
8BC8- 00 00 00 00 00 00 00 E0
8BD0- E0 60 F8 BE 0F 01 00 01
8BD8- 81 01 00 00 00 00 00 42
8BE0- 64 38 64 42 00 00 00 03
8BE8- 07 7F 7F 03 00 00 40 00
8BF0- 00 00 01 03 83 03 81 00
8BF8- 00 00 00 00 00 00 00 00
8C00- 00 00 00 00 00 00 00 00
8C08- 00 30 78 7C 7E FF 7F 00
8C10- 00 00 00 01 81 C1 43 84
8C18- 0C 1C 3E BF 7F 7F 3F FF
8C20- EF 4E C6 00 00 00 00 E3
8C28- 67 F7 FF 7F FF 7E FE 1F
8C30- 0F 0F 8F 9F 37 B3 03 00
8C38- 00 00 00 00 00 00 00 7C
8C40- FC 78 00 00 00 00 00 03
8C48- 03 03 00 00 00 00 00 00
8C50- 00 00 60 F8 FC FE 7E 00
8C58- 00 00 7F 7F 7F 7F 7F 00
8C60- 00 00 00 03 87 8D 8F 00
8C68- 00 00 00 00 00 00 02 FD
8C70- 79 78 38 38 38 38 00 7F
8C78- 7F 7F 7F 60 60 60 00 8F
8C80- 7B 31 01 01 01 01 00 01
8C88- 00 00 00 00 00 00 00 00
8C90- 00 40 60 70 78 7E 79 A2
8C98- 24 3C 5A 3C 9C 18 18 00
8CA0- 00 00 00 00 00 78 7C 18
8CA8- 18 18 18 1C 1E 1F 8F 7E
8CB0- 7D 7D 0C 0C 0C 0C 84 0F
8CB8- 07 07 06 06 06 06 82 00
8CC0- 00 00 00 00 00 60 78 00
8CC8- C0 40 E0 40 43 5F 7F 00
8CD0- 01 03 0E 07 01 01 01 00
8CD8- 00 00 00 00 00 00 00 00
8CE0- 00 00 00 00 00 00 00 00
8CE8- 00 00 00 00 00 00 00 00
8CF0- 00 00 00 00 00 00 00 00
8CF8- 00 00 00 00 00 00 00 00
8D00- 00 00 00 00 00 00 00 7C
8D08- 78 60 40 40 C0 00 00 7F
```

*Vous avez un Apple //e avec Chat Mauve ou un //c ?*
*Vous avez Pascal 1.2 ?*

Utilisez

# COGO

Par Nicolas Montsarrat

**Apple //e, //c**

***Il s'agit d'un système graphique double-haute résolution écrit en Pascal. COGO vous permet de manipuler des graphiques grâce à un langage de description des objets - points, angles- et à l'emploi de fonctions primitives de manipulation très puissantes : cercle, tangente, intersections, parallèles, etc. Il est ainsi possible de tracer des grilles, des cercles, des segments de droite, des tangentes communes à deux cercles, de calculer des distances, des angles...***

***L'éditeur permet une saisie rapide du langage. Une instruction COGO peut-être exécutée dès la saisie pour faciliter la mise au point, ou au sein d'un programme.***

Ce programme, destiné à résoudre des problèmes de géométrie plane, comporte des instructions de stockage sur fichier afin de permettre la reprise d'un calcul.

**150,00 F TTC, franco**
**Bon de commande page 74**

```
8D10- 3F 1F 07 04 84 09 00 00
8D18- 00 00 00 00 00 00 00 00
8D20- 00 00 00 00 00 00 00 00
8D28- 00 67 6F 7F 7F 7F 7F 00
8D30- 00 00 01 03 86 1F 3F 06
8D38- 0A 12 22 4A 42 44 48 22
8D40- 14 08 1C 3E 3E 3E 3E 30
8D48- 28 24 22 29 21 11 09 44
8D50- 42 4A 22 12 0A 06 00 3E
8D58- 3E 1C 08 00 00 00 00 11
8D60- 21 29 22 24 28 30 00 00
8D68- 00 00 00 00 C0 C0 00 00
8D70- 12 9B 3F 2D BF BF 3F 00
8D78- 00 00 00 00 40 40 40 1E
8D80- 0C 1E 3F 3F 7F 7F 7F 40
8D88- 40 40 00 00 00 20 40 7F
8D90- 7F 3F 3F 1F 0E 08 07 00
8D98- 00 00 00 C0 C0 C0 00 00
8DA0- 00 01 03 87 1D 3F 07 00
8DA8- 40 60 60 70 70 78 78 03
8DB0- 07 07 0F 8F 57 27 07 78
8DB8- 7C FA 7A 99 8D 79 60 07
8DC0- 03 03 01 00 00 00 83 00
8DC8- 00 00 00 06 0C 58 F0 00
8DD0- 00 00 00 00 00 00 00 00
8DD8- 00 00 00 00 00 00 00 00
8DE0- 00 00 00 00 00 00 00 00
8DE8- 00 00 00 00 00 00 00 00
8DF0- 00 00 00 00 00 00 00 00
8DF8- 00 00 00 00 00 00 00
```

## Fichier 'BONHOMME.SET'

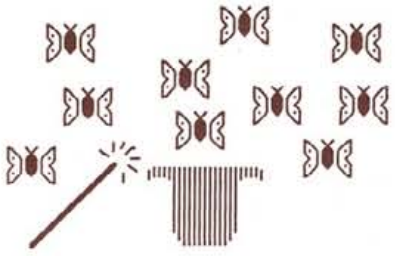
*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE BONHOMME.SET,A$6700,L$300*

```
6700- 00 00 00 00 00 00 00 00
6708- 00 00 00 00 00 00 00 00
6710- 00 00 00 00 00 00 00 00
6718- 00 00 00 00 00 00 00 00
6720- 00 00 00 00 00 00 00 00
6728- 00 00 00 00 00 00 00 00
6730- 00 00 00 00 00 00 00 00
6738- 00 00 00 00 00 00 00 00
6740- 00 00 00 00 00 00 00 00
6748- 00 00 00 00 00 00 00 00
6750- 00 00 00 00 00 00 00 00
6758- 00 00 00 00 00 00 00 00
6760- 00 00 00 00 00 00 00 00
6768- 00 00 00 00 00 00 00 00
6770- 00 00 00 00 00 00 00 00
6778- 00 00 00 00 00 00 00 00
6780- 0F 1F 1F 0F 07 07 07 07
6788- 60 60 60 60 60 60 60 70
6790- 06 06 0E 1C 38 70 30 10
6798- 00 30 78 78 78 30 70 78
67A0- 00 00 00 00 00 00 00 03
67A8- 00 00 00 40 60 20 00 00
67B0- 7C 7E 73 71 70 70 70 70
67B8- 07 0F 0D 0D 0C 0D 05 01
67C0- 78 38 78 70 70 30 30 38
67C8- 00 00 00 01 07 06 06 00
67D0- 00 00 00 00 00 00 00 00
67D8- 00 00 00 00 00 00 00 00
67E0- 00 00 00 00 00 00 00 00
67E8- 00 00 00 00 00 00 00 00
67F0- 00 00 00 00 00 00 00 00
67F8- 00 00 00 00 00 00 00 00
6800- 00 00 00 00 00 00 00 00
6808- 00 0C 1E 1E 1E 0C 7F 7F
```

```
1000  REM ***PAGE2
1010  PRINT CO$CB$CO$CD$
1020 W = 5
1030  FOR K = 1 TO NN
1040 X = 34:Y = 10
1050 I = 0: GOSUB 2000:I = 1: GOSUB 2000:X = X - 1:
     I = 2: GOSUB 2000: IF X > 4 * (K - 1) + 1 THEN
      1050
1060  IF K < NN AND NN <  > 2 THEN  VTAB 10: HTAB 4
      * K: PRINT VV$;
1070  IF K < NN AND NN = 2 THEN  VTAB 10: HTAB 4 *
     K + 1: PRINT VV$;
1080  VTAB 5: HTAB 4 * (K - 1) + 1: PRINT CH$(K)
1090  IF K > 1 THEN  FOR L = 1 TO K - 1: VTAB 5: HT
     AB 4 * (K - 2) + 1: PRINT VV$: NEXT L
1100 W = W + 5
1110  NEXT K
1120  IF NN = 2 THEN  VTAB Y: HTAB X + 4: PRINT BO$
     (4): GOTO 1140
1130  VTAB Y: HTAB X + 3: PRINT BO$(4)
1140  FOR W = 1 TO 500: NEXT W
1150  PRINT CO$CA$CO$CD$CO$CB$: FOR I = 4 TO 13: VT
     AB I: PRINT CE$: NEXT I: PRINT CO$CA$
1160  RETURN
2000  VTAB Y: HTAB X:GG =  PEEK ( - 16336): PRINT A
     N$(NN);: VTAB Y: PRINT BO$(I);: FOR J = 1 TO W
     : NEXT : RETURN
3000  REM  *** EFFETS SONORES
3010  FOR I = 0 TO 49: READ SO: POKE 768 + I,SO: NE
     XT I
3020  DATA 169,0,133,24,169,5,133
3030  DATA 25,169,1,133,26,164,25
3040  DATA 165,26,32,34,3,73,255
3050  DATA 73,208,32,34,3
3060  DATA 136,208,241,230
3070  DATA 26,16,235,96,44,48,192
3080  DATA 166,24,134,27,170,202
3090  DATA 208,253,198,27,16,248
3100  DATA 96
3110  REM *** POSITION DES ANIMAUX
3120  FOR I = 1 TO 9: FOR J = 0 TO I - 1: READ CO(I
     ,2 * J): READ CO(I,2 * J + 1): NEXT J: NEXT I
3130  DATA 3,19
3140  DATA 6,11,6,27
3150  DATA  6,11,3,19,6,27
3160  DATA 8,7,4,15,4,23,8,31
3170  DATA 11,5,6,11,3,19,6,27,11,32
3180  DATA 8,7,6,11,4,15,4,23,6,27,8,31
3190  DATA 8,7,6,11,4,15,3,19,4,23,6,27,8,31
3200  DATA 11,5,8,7,6,11,4,15,4,23,6,27,8,31,11,33
3210  DATA 11,5,8,7,6,11,4,15,3,19,4,23,6,27,8,31,1
     1,33
3220  RETURN
4000  REM  *** CHAPEAU & ETOILES
4010  HCOLOR= 5
4020  FOR Y = 100 TO 140: HPLOT 120,Y TO 145,Y: NEX
     T
4030  FOR Y = 96 TO 99: HPLOT 110,Y TO 155,Y: NEXT
4040  VTAB 4: HTAB 2: PRINT "*": VTAB 5: HTAB 38: P
     RINT "*": VTAB 6: HTAB 6: PRINT "*": VTAB 3: H
     TAB 35: PRINT "*": VTAB 5: HTAB 9: PRINT "*":
     VTAB 7: HTAB 39: PRINT "*"
4050  RETURN
5000  REM *** INITIALISATIONS
5010  ONERR  GOTO 6010
5020  TEXT : HOME : HGR :ADRS = 0
5030  PRINT  CHR$ (4)"BLOAD RBOOT": CALL 520
5040 ADRS =  USR (0),"HRCG"
5050  POKE 216,0
5060  IF ADRS < 0 THEN ADRS = ADRS + 65536
5070 CS = ADRS - 3 * 768: HIMEM: CS
5080 D$ =  CHR$ (4)
5090  PRINT D$"BLOAD ANIMAUX.SET,A";CS
5100  PRINT D$"BLOAD GROS.CHIFFRE.SET,A"CS + 768
5110  PRINT D$"BLOAD BONHOMME.SET,A"CS + 2 * 768
5120 CH =  INT (CS / 256):CL = CS - CH * 256
5130  POKE ADRS + 7,CL: POKE ADRS + 8,CH: CALL ADRS
     + 3
5140  RETURN
6000  REM *** TRAITEMENT ERREUR
6010  TEXT
```

```
6810- 7F 3F 3F 3F 1E 3F 3F 3F
6818- 33 33 33 33 33 33 33 73
6820- 00 00 00 00 00 00 40 60
6828- 70 30 30 30 30 30 30 20
6830- 00 00 00 00 00 00 00 40
6838- 00 00 00 00 00 00 00 01
6840- 03 03 03 03 03 03 03 01
6848- 58 0C 06 03 01 00 00 00
6850- 5E 07 00 00 00 00 00 00
6858- 00 00 00 00 00 00 60 70
6860- 00 00 00 00 00 00 60 78
6868- 40 00 00 00 00 00 00 00
6870- 00 00 00 00 00 00 7F 7E
6878- 00 00 00 00 03 0E 7C 70
6880- 01 02 06 0E 1C 38 70 60
6888- 10 10 18 18 18 18 78 78
6890- 06 0C 18 30 40 00 00 00
6898- 00 00 00 00 00 00 01 07
68A0- 1E 38 40 00 00 00 00 00
68A8- 00 00 00 00 00 00 7F 1F
68B0- 00 00 00 40 30 3C 0F 03
68B8- 20 10 18 1C 0E 07 03 01
68C0- 02 02 06 06 06 06 07 07
68C8- 00 00 00 00 00 00 00 00
68D0- 00 00 00 00 00 00 00 00
```

```
68D8- 00 00 00 00 00 00 00 00
68E0- 00 00 00 00 00 00 00 00
68E8- 00 00 00 00 00 00 00 00
68F0- 00 00 00 00 00 00 00 00
68F8- 00 00 00 00 00 00 00 00
6900- 00 00 00 00 00 00 00 00
6908- 00 18 3C 3C 3C 18 1C 3E
6910- 3E 3E 3E 7E 7C 3C 3C 3C
6918- 00 00 00 00 01 03 02 00
6920- 7C 6C 6C 4C 4C 0C 0C 1C
6928- 00 00 01 01 03 03 03 07
6930- 00 40 60 60 60 40 60 70
6938- 00 01 03 03 03 01 01 03
6940- 78 7C 7C 78 70 70 70 70
6948- 03 07 07 0F 0D 1B 33 23
6950- 30 30 38 1C 0E 07 06 04
6958- 03 03 03 03 03 03 03 07
6960- 00 00 00 00 00 00 00 60
6968- 00 06 0F 0F 0F 06 07 0F
6970- 70 78 58 58 18 58 50 40
6978- 1F 3F 67 47 07 07 07 07
6980- 00 00 00 01 03 02 00 00
6988- 00 00 00 40 70 30 30 00
6990- 0F 0E 0F 07 07 06 06 0E
6998- 00 0C 1E 1E 1E 0C 1C 3E
```



```
69A0- 00 00 00 00 40 60 20 00
69A8- 3E 3E 3E 3F 1F 1E 1E 1E
69B0- 00 00 40 40 60 60 60 70
69B8- 1F 1B 1B 19 19 18 18 1C
69C0- 00 40 60 60 60 40 40 60
69C8- 00 01 03 03 03 01 03 07
69D0- 60 70 70 78 58 6C 66 62
69D8- 00 00 00 00 00 00 00 00
69E0- 00 00 00 00 00 00 00 00
69E8- 00 00 00 00 00 00 00 00
69F0- 00 00 00 00 00 00 00 00
69F8- 00 00 00 00 00 00 00 00
```

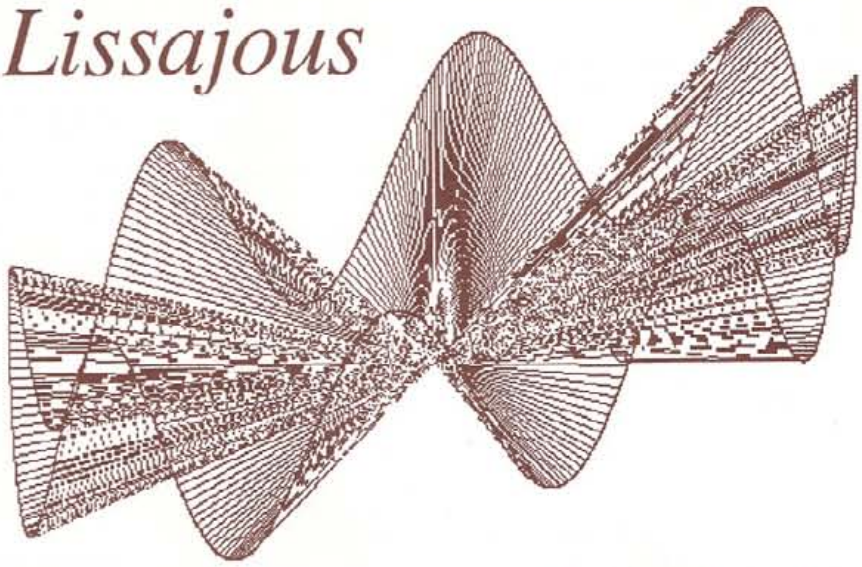
```
6020  PRINT "ERREUR DANS RLOAD OU RBOOT"
6030  POKE 216,0
6040  END
7000  REM *** DEFINITION DES
7010  REM *** FIGURINES
7020 AN$(1) = CA$ + "1" + CB$ + "ABC" + CC$ + "DEF"
      + CC$ + "GHI" + CD$ + CA$ + "0"
7030 AN$(2) = CA$ + "1" + CB$ + "JKLM" + CC$ + "NOP
     Q" + CD$ + CA$ + "0"
7040 AN$(3) = CA$ + "1" + CB$ + "R" + CHR$ (101) +
       CHR$ (102) + CD$ + CA$ + "0"
7050 AN$(4) = CA$ + "1" + CB$ + " S " + CC$ + "TU "
      + CC$ + "VW " + CD$ + CA$ + "0"
7060 AN$(5) = CA$ + "1" + CB$ + "XYZ" + CC$ +  CHR$
      (97) +  CHR$ (98) + " " +  + CD$ + CA$ + "0"
7070 AN$(6) = CA$ + "1" + CB$ +  CHR$ (103) +  CHR$
      (104) +  CHR$ (105) + CC$ +  CHR$ (106) +  CH
     R$ (107) +  CHR$ (108) + CD$ + CA$ + "0"
7080 AN$(7) = CA$ + "1" + CB$ +  CHR$ (109) +  CHR$
      (110) + " " + CC$ +  CHR$ (111) +  CHR$ (112)
      + " " + CC$ +  CHR$ (113) +  CHR$ (114) + " "
      + CD$ + CA$ + "0"
7090 AN$(8) = CA$ + "1" + CB$ + " " +  CHR$ (115) +
       CHR$ (116) + CC$ + " " +  CHR$ (117) +  CHR$
      (118) + CC$ +  CHR$ (121) +  CHR$ (119) +  CH
     R$ (120) + CD$ + CA$ + "0"
7100 AN$(9) = CA$ + "1" + CB$ + "!" +  CHR$ (34) +
     "#" + CC$ + "$%&" + CD$ + CA$ + "0"
7110 VV$ = CB$ + "    " + CC$ + "    " + CC$ + "
     " + CD$
7120  FOR I = 1 TO 9
7130 J = 6 * (I - 1)
7140 CH$(I) = CA$ + "2" + CB$ +  CHR$ (33 + J) +  C
     HR$ (34 + J) + CC$ +  CHR$ (35 + J) +  CHR$ (3
     6 + J) + CC$ +  CHR$ (37 + J) +  CHR$ (38 + J)
      + CD$ + CA$ + "0"
7150  NEXT I
7160 CH$(0) = CA$ + "2" + CB$ + "WX" + CC$ + "YZ" +
       CC$ +  CHR$ (91) +  CHR$ (92) + CD$ + CA$ + "
     0"
7170 BO$(0) = CB$ + CA$ + "3" + CL$ + " S " + CC$ +
      "TU " + CC$ + "VW " + CD$ + CA$ + "0"
7180 BO$(1) = CB$ + CA$ + "3" + CL$ + "XY " + CC$ +
      "Z0 " + CC$ + "12 " + CD$ + CA$ + "0"
7190 BO$(2) = CB$ + CA$ + "3" + CL$ + " 34" + CC$ +
      "567" + CC$ + " 89" + CD$ + CA$ + "0"
7200 BO$(3) = CB$ + CA$ + "3DAG" + CC$ + "EBH" + CC
     $ + "FC " + CD$ + CA$ + "0"
7210 BO$(4) = CB$ + CA$ + "3PAG" + CC$ + "MBH" + CC
     $ + "FC " + CD$ + CA$ + "0"
7220 BO$(5) = CB$ + CA$ + "3OAG" + CC$ + "MBH" + CC
     $ + "FC " + CD$ + CA$ + "0"
7230 BO$(6) = CB$ + CA$ + "3NAG" + CC$ + "MBH" + CC
     $ + "FC " + CD$ + CA$ + "0"
7240  RETURN
8000  REM *** INITIALISATIONS
8010  REM *** APPLESOFT
8020 CA$ =  CHR$ (1):CB$ =  CHR$ (2):CC$ =  CHR$ (3
     ):CD$ =  CHR$ (4):CE$ =  CHR$ (5):CF$ =  CHR$
     (6):C1$ =  CHR$ (9):CK$ =  CHR$ (11):CL$ =  CH
     R$ (12):CN$ =  CHR$ (14):CO$ =  CHR$ (15):CP$
     =  CHR$ (16):CS$ =  CHR$ (19):CT$ =  CHR$ (20)
8030 CV$ =  CHR$ (22):CW$ =  CHR$ (23):CY$ =  CHR$
     (25):CZ$ =  CHR$ (26)
8040 P1$ = CO$ + CA$:P2$ = CO$ + CB$:PT$ = CO$ + CT
     $:PN$ = CO$ + CP$
8050 KB =  - 16384:G$ =  CHR$ (7)
8060  PRINT CP$CK$: HTAB 11: VTAB 1: PRINT " LE CHA
     PEAU MAGIQUE "
8070  HCOLOR= 1: HPLOT 0,0 TO 0,191 TO 279,191 TO 2
     79,0: HPLOT 1,0 TO 1,190 TO 278,190 TO 278,1:
     HPLOT 1,0 TO 66,0: HPLOT 1,1 TO 66,1
8080  HPLOT 214,0 TO 279,0: HPLOT 214,1 TO 278,1
8090  HCOLOR= 5: HPLOT 70,9 TO 210,9
8100  POKE 230,64: HCOLOR= 6: HPLOT 1,1: CALL 62454
     : POKE 230,32
8110  PRINT CO$CB$: FOR I = 4 TO 13: VTAB I: PRINT
     CE$: NEXT I: PRINT CO$CA$
8120  RETURN
```

## Fichier 'GROS.CHIFFRES.SET'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE GROS.CHIFFRES.SET,A$8AFF,L$200*

```
8AFF- 00
8B00- 00 00 00 00 00 00 00 40
8B08- 40 E0 60 B0 30 98 18 07
8B10- 07 07 07 07 07 07 07 8C
8B18- 0C 86 06 00 00 00 00 07
8B20- 07 07 07 07 07 07 07 00
8B28- 00 00 00 00 00 00 00 07
8B30- 07 07 07 07 07 07 07 60
8B38- 78 7C 86 06 06 06 06 03
8B40- 0F 1F 9C B8 70 70 70 00
8B48- 00 00 00 40 60 70 78 B8
8B50- 38 9E 8F 87 83 01 00 3C
8B58- 9E 1E 8F 0F 7F 7F 7F 00
8B60- 00 00 00 00 7F 7F 7F 60
8B68- 78 0C 86 06 86 84 00 03
8B70- 0F 8E 9C B8 F8 70 70 00
8B78- 00 00 60 60 00 00 00 70
8B80- 38 1C 0F 07 1C 38 70 00
8B88- 84 86 06 86 0C 78 60 70
8B90- 70 F8 B8 9C 8E 0F 03 00
8B98- 40 E0 60 F0 70 B8 38 00
8BA0- 00 00 00 00 00 00 00 9C
8BA8- 1C 8E 0E 87 07 07 07 00
8BB0- 00 00 00 00 07 07 07 7F
8BB8- 7F 7F 00 00 00 00 00 7F
8BC0- 7F 7F 07 07 07 07 07 7F
8BC8- 7F 7F 07 07 07 07 07 3F
8BD0- 3F 3F 00 00 00 00 00 07
8BD8- 07 07 07 7F 7F 00 00 00
8BE0- 00 00 00 87 8F 1C 9C 00
8BE8- 00 00 07 87 8E FC 70 38
8BF0- 38 38 38 38 9C 8F 07 00
8BF8- 00 C0 40 E0 60 F0 70 00
8C00- 07 83 03 81 01 80 00 B8
8C08- 38 FC 7C FE 1E 8F 0F 00
8C10- 00 87 8F 9F BC 78 F8 0F
8C18- 0F 0F 8F 1E 7C 78 70 F8
8C20- F8 F8 78 3C 1F 0F 07 7F
8C28- 7F 7F 00 00 00 00 00 7F
8C30- 7F 7F 70 B8 38 9C 1C 00
8C38- 00 00 00 C0 78 FC 7C 8E
8C40- 0E 87 07 83 1F 8F 0F F0
8C48- 70 B8 38 9C 1C 8E 0E 80
8C50- 00 00 00 00 00 00 00 78
8C58- 7C 8F 0F 0F 0F 0F 0F 0F
8C60- 1F BC 78 F8 F8 F8 78 8F
8C68- 1E FC 7C FE 1E 8F 0F BC
8C70- 3C 8F 8F 9F BC 78 F8 0F
8C78- 0F 0F 8F 1E 7C 78 70 F8
8C80- F8 F8 78 3C 1F 0F 07 FC
8C88- 7C FE 1E 8F 0F 0F 0F 87
8C90- 8F 9F BC 78 F8 F8 F8 0F
8C98- 8F 1E 7C 78 70 00 00 F8
8CA0- 78 BC 3F 9F 1F 8E 0E 00
8CA8- 00 C0 40 E0 60 F0 00 87
8CB0- 07 83 03 81 01 80 00 FC
8CB8- 7C FE 1E 8F 0F 0F 0F 87
8CC0- 8F 9F BC 78 F8 F8 F8 0F
8CC8- 0F 0F 0F 0F 0F 0F 0F F8
8CD0- F8 F8 F8 F8 F8 F8 F8 8F
8CD8- 1E FE 7C FC 00 00 00 78
8CE0- BC 9F 8F 87 00 00 00 00
8CE8- 00 00 00 00 00 00 00 00
8CF0- 00 00 00 00 00 00 00 00
8CF8- 00 00 00 00 00 00 00
```

# Lissajous



Norbert Vurlod

Les courbes de Lissajous sont des figures que l'on peut obtenir sur l'écran d'un oscilloscope, en y faisant entrer deux courants alternatifs simultanément, mais l'un par l'entrée horizontale et l'autre par l'entrée verticale. L'effet obtenu dépend du rapport des deux fréquences, ainsi que du déphasage entre les deux courants.
Afin d'accélérer le tracé de cette courbe, le cercle trigonométrique est divisé ici en 256 parties ( au lieu de 2 π ) puis, en Basic, les sinus et cosinus de chaque angle sont calculés et ramenés aux coordonnées possibles de chaque point sur l'écran de l'Apple, puis stockés dans une table en mémoire. Cette pratique est très courante.

Le programme assembleur écrit en Tool-Kit est lui-même documenté, il suffit d'ajouter que le paddle 0 fait varier rapidement le déphasage entre les deux courants (effet de rotation horizontale plus rapide ), alors que le paddle 1 détermine le rapport des fréquences, toujours entre les deux courants. Il entraîne le changement de la forme de la courbe, du simple trait horizontal, à une sorte de couronne royale, en passant par le cercle dans le cas de 2 courants de même fréquence. Ce rapport est limité à 16 afin de conserver une définition suffisante à la courbe (sous moniteur, remplacer les LSR par des NOP 4A → EA pour vous en convaincre).

Quand la vue de ces courbes aura atteint son effet hypnotique maximum, vous pourrez toujours vous endormir après avoir appuyé sur ESC pour arrêter le programme...

## Fichier 'LISSAJOUS.DATA'

*Ce fichier binaire est une liste de sinus et cosinus que vous obtiendrez en lançant l'exécution du programme Basic LISSAJOUS.FAIT DATA*

### *Comment faire ?*

*1*
*créer le fichier LISSAJOUS.DATA en lançant l'exécution de LISSAJOUS.FAIT DATA (si vous ne disposez pas de la disquette Pom's)*
*2*
*saisir et sauvegarder l'objet LISSAJOUS.OBJ.*
*3*
*faire RUN LISSAJOUS.BAS.*
*4*
*régler fréquence et phase à l'aide des paddles.*

## Programme 'LISSAJOUS.FAIT DATA'

```
0  HIMEM: 20000:A = 20000:B =
   A + 256:PI2 = 8 *  ATN (1
   )
20  FOR I = 0 TO 255:J = PI2
   * I / 256: PRINT I,
21  POKE A + I,95 + 95 *  SIN
   (J): PRINT  PEEK (A + I),
22  POKE B + I,142 + 86 *  CO
   S (J): PRINT  PEEK (B + I)
26  NEXT I: PRINT  CHR$ (4)"B
   SAVE LISSAJOUS.DATA,A"A",L
   512"
```

## Programme 'LISSAJOUS.BAS'

```
10  PRINT  CHR$ (4)"BLOAD
    LISSAJOUS.DATA,A$6000"
20  PRINT  CHR$ (4)"BRUN LI
    SSAJOUS.OBJ"
```

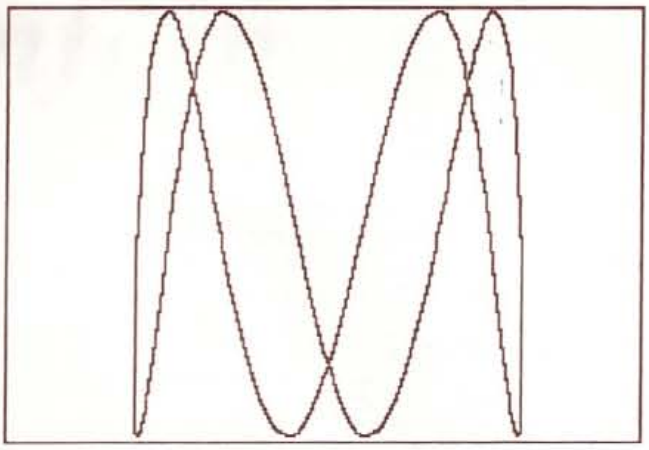
## Récapitulation 'LISSAJOUS.OBJ'

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE LISSAJOUS.OBJ,A$300,L$9E*

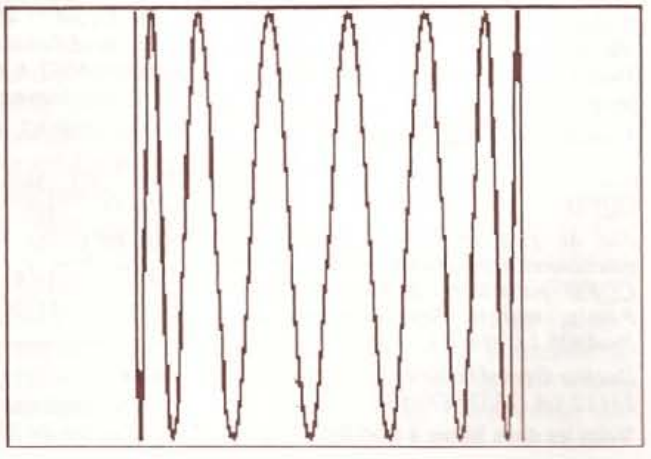
```
0300- A9 00 85 07 85 08 A9 20
0308- 85 E6 A9 55 8D 32 03 20
0310- F2 F3 2C 54 C0 2C 52 C0
0318- 2C 57 C0 2C 50 C0 A2 03
0320- 20 EC F6 A9 60 45 E6 85
0328- E6 A9 01 4D 32 03 8D 32
0330- 03 2C 00 C0 20 F2 F3 A2
0338- 00 20 1E FB 98 18 65 07
0340- 85 09 AA BD 00 60 A0 00
0348- AE 00 61 20 57 F4 A9 00
0350- 85 08 A2 01 20 1E FB 98
0358- 4A 4A 4A 4A 85 18 A5 09
0360- 18 65 18 85 09 AA BC 00
0368- 60 A6 08 BD 00 61 A2 00
0370- 20 3A F5 20 CB F5 E6 08
0378- D0 E4 A6 09 BC 00 60 A2
0380- 00 AD 00 61 20 3A F5 20
0388- CB F5 AD 00 C0 C9 9B F0
0390- 06 E6 07 D0 8E F0 8C 2C
0398- 10 C0 20 2F FB 60
```

# Source 'LISSAJOUS.SRCE'
## Assembleur Tool-Kit

```
1          ORG  $300
2  ************************************
3  * COURBE DE LISSAJOUS
4  ****************************
5  *
6  *
7  ANGLE1  EQU  $07        ;DESSINE LA COURBE
8  ANGLE2  EQU  $08        ;ROTATION HORIZONTALE
9  ORD     EQU  $09
10 RAP.FREQ EQU $18        ;RAPPORT DES FREQUENCES
11 *
12 HCLR    EQU  $F3F2
13 PREAD   EQU  $FB1E      ;LECTURE PADDLE
14 HLIN    EQU  $F53A
15 *
16 ************************************
17 * INITIALISATIONS
18 ************************************
19 *
20         LDA  #$00
21         STA  ANGLE1
22         STA  ANGLE2
23 *
24         LDA  #$20       ;TRACE EN P.1
25         STA  $E6
26         LDA  #$55       ;VISU DE  P.2
27         STA  PAGE
28         JSR  HCLR       ;EFFACE LA PAGE HGR1
29 *
30         BIT  $C054      ;COMMUTE P.1
31         BIT  $C052      ;MODE NON MIXTE
32         BIT  $C057      ;HIRES
33         BIT  $C050      ;MODE GRAPHIQUE P.1
34 *
35         LDX  #$03
36         JSR  $F6EC      ;HCOLOR=3
37 *
38 ************************************
39 * PROGRAMME PRINCIPAL
40 ************************************
41 *
42 BOUCLE1 EQU  *
43         LDA  #$60
44         EOR  $E6        ;CONTIENT $20 OU $40
45         STA  $E6        ;POUR P.1 OU P.2
46         LDA  #$01       ;TRACE SUR UNE PAGE
47         EOR  PAGE       ;PENDANT QUE L'AUTRE
48         STA  PAGE       ;EST AFFICHEE...
49         BIT  $C000      ;( $C054 OU $C055 )
50 PAGE    EQU  *-2
51         JSR  HCLR       ;EFFACE LA PAGE CACHEE
52         LDX  #$00       ;LECTURE DU
53         JSR  PREAD      ;PADDLE 0
54         TYA
55         CLC
56         ADC  ANGLE1
57         STA  ORD
58         TAX              ;TRACE DU
59         LDA  TAB.SIN,X ;PREMIER
60         LDY  #$00       ;POINT
61         LDX  TAB.COS
62         JSR  $F457      ;HPLOT X,Y
63 *
```



```
64         LDA  #$00
65         STA  ANGLE2
66 *
67         LDX  #$01       ;LECTURE DU
68         JSR  PREAD      ;PADDLE 1
69         TYA
70         LSR  A          ;DIVISION
71         LSR  A          ;PAR 16 DE
72         LSR  A          ;PDL(1)
73         LSR  A
74         STA  RAP.FREQ   ;MAXIMUM 16 PICS
75 ************************************
76 * TRACE DE LA BOUCLE
77 ************************************
78 BOUCLE2 EQU  *
79         LDA  ORD
80         CLC
81         ADC  RAP.FREQ
82         STA  ORD
83         TAX
84         LDY  TAB.SIN,X
85         LDX  ANGLE2
86         LDA  TAB.COS,X
87         LDX  #$00
88         JSR  HLIN       ;HPLOT TO X,Y
89         JSR  $F5CB      ;REINITIALISATION HGR
90         INC  ANGLE2
91         BNE  BOUCLE2
92 *
93 *
94 ************************************
95 * FERME LA BOUCLE
```



```
96 ****************************************
97          LDX  ORD
98          LDY  TAB.SIN,X
99          LDX  #$00
100         LDA  TAB.COS
101         JSR  HLIN
102         JSR  $F5CB      ;INIT HIRES
103 *
104         LDA  $C000      ;TEST CLAVIER
105         CMP  #128+27    ;TOUCHE ESC
106         BEQ  FIN
107 *
108         INC  ANGLE1
109         BNE  BOUCLE1    ;ON RECOMMENCE
110         BEQ  BOUCLE1
111 *
112 FIN     EQU  *
113         BIT  $C010      ;RAZ CLAVIER
114         JSR  $FB2F      ;TEXT
115         RTS
116 *
117 ****************************************
118 * TABLE DES SINUS ET DES COSINUS
119 ****************************************
120         DSECT
121         ORG  $6000
122 TAB.SIN DS   $100
123 TAB.COS DS   $100
124         DEND
```

# Courrier des Lecteurs

### FORMAT (numéro 23)

*Avez-vous essayé de répondre MON TITRE dans l'option &T du programme FORMAT de Pom's 23 ? Sous DOS 3.3, le DOS intercepte la saisie comme une commande et envoie donc un SYNTAX ERROR. Je ne parle pas des conséquences avec un titre tel que INITIALES...*

*Les 54 octets du patch ci-dessous éliminent ce petit problème. En bonus, j'ai rajouté quelques octets pour une gestion complémentaire des codes de contrôle :*

*CTRL-D inséré par exemple dans un programme sera listé ^D.*

```
BLOAD FORMAT
CALL-151
94AB:F0 7 4C 18 95 A9 0 F0 2 A9
  20 85 CE A9 3 85 CF 20 F8 94
94BF:A2 BF 86 33 A9 FF C5 76 D0
  2 86 76 A5 38 48 A5 39 48
94D1:A9 E1 85 38 A9 3 85 39 20
  6A FD E0 21 B0 22 EA
956E:60
9581:60
959D:60
93B6:C9 20 B0 11 69 40 48 A9 5E
  20 D4 93 68
93C3:EA EA EA EA EA EA EA EA
UNLOCK FORMAT
BSAVE FORMAT,A$90D1,L$527
```

*Ce patch tourne comme un fichier Exec (FORMAT.PATCH sur la disquette Pom's), à n'utiliser bien sûr que sur une version de sauvegarde... Il ne fonctionne que pour la version DOS 3.3 du programme. Son adaptation à ProDOS ne devrait pas vous poser de problèmes.*

*Yvan Koenig, 06220 Vallauris*

---

### CQFD

*J'ai de gros problèmes pour faire fonctionner le programme de conversion CQFD paru dans le numéro 16 de Pom's, malgré des tentatives sur ProDOS 1.0 et 1.1.1.*

*Docteur Gérard Lacurie*
*13112 LA DESTROUSSE*

Voici les deux lignes à modifier:

```
117  IF PEEK(48896) <> 76 THEN FLASH :
  PRINT "VOUS N'ETES PAS SOUS PRODOS!":
NORMAL:END
6021 POKE 811,163 : POKE 812,154
```

Cette dernière ligne appelle une routine du Basic.System en $9AA3, auparavant située en $9AAF, qui transfère les adresses $BE34-BE38 vers $36-39. Par manque de place (il faut faire "cohabiter" les deux DOS en mémoire), elle ne pouvait être incluse dans le programme CQFD.B, d'où des problèmes lorsque ProDOS évolue...

---

### AppleWriter et ^"

*Quelques mots pour vous indiquer le moyen de modifier AppleWriter ProDOS pour qu'il tienne compte des ^ et " en justification totale. A l'aide d'un éditeur de secteur, il convient de modifier les zones comme indiqué ci-dessous :*

*Piste 00, Secteur 0A, Octet 2A*
*mettre 5B au lieu de 19*
*Piste 0F, Secteur 0F, Octet E9*
*mettre 4C 20 60 EA EA*
*Piste 10, Secteur 00, Octet 09*
*mettre 4C 40 60*
*Piste 11, Secteur 09, Octet 20*
*mettre CD DE B8 D0 03 4C F5 49 C9 08 F0 03 4C EE 49 E6 75 E6 75 C4 75 F0 03 4C A8 49 4C 05 4A EA EA EA A4 75 B9 00 1C C9 20 F0 0D C9 08 F0 03 88 D0 F2 C6 75 C6 75 EA EA A4 75 4C EF 4B 00*

*Cette modification convient pour les Apple //e 6502 & 65C02 ; elle intervient sur le programme AWD.SYS.*

*Michel DUROC, 97220 Trinité*

Voici une solution non ambiguë des accents qui posent bien des problèmes inélégants même au mois d'août. Faire un essai sur une copie est une bonne précaution...

---

### Snake

*J'aurais souhaité utiliser le jeu Snake (numéro 24 de Pom's) avec de jeunes enfants et, pour cela, réduire la vitesse de déplacement du serpent. Philippe Krepper pourrait-il me fournir une méthode qui éviterait un réassemblage ?*

*Yves Robert, 36000 Châteauroux*

1ère solution
Programme SNAKE.FUSION :
supprimer de la ligne 50 le

```
PRINT D$"BSAVE"F$",A973,L3111"
```

et ajouter :

```
60 DATA 169,15,133,213,32,79,15
  ,198,213,208,249,234
70 FOR L = 0 TO 11: READ A: POKE
  3672 + L,A: NEXT
80 PRINT D$"BSAVE"F$",A973,L3111"
```

La valeur 15 de la ligne 60 est a déterminer expérimentalement : plus elle est grande, plus le serpent ralentit.

2ème solution
Elle nécessite un assembleur mais permet de faire varier à loisir la vitesse :
Insérer après la ligne 477 :

```
(I478)
478   RALENT  INC   BCL04+1
479           RTS
480   ACCELR  DEC   BCL04+1
481           BEQ   RALENT
482           RTS
(CTRL E)
```

Insérer après la ligne 466 :

```
(I467)
467           CMP   #$C1 ; "A"
468           BEQ   ACCELR
469           CMP   #$D2 ; "R"
470           BEQ   RALENT
(CTRL E)
```

Remplacer les lignes 338 à 346 par :

```
(M338,346)
338   BCL04   LDA   #$F
339           STA   $D5
340   BCL05   JSR   LITSNS
341           DEC   $D5
342           BNE   BCL05
(CTRL E)
```

Ensuite, assembler le programme et le sauver par :

```
BSAVE SNAKE.OBJ,ASC00,L$3F6
```

et enfin faire tourner le programme SNAKE.FUSION en ayant modifié la ligne 50 , L3111 devient , L3113.
En cours de partie, on peut alors modifier la vitesse par A et R.

# *Jean-Michel Gourévitch* Micro-informations

Plus de quatre millions de micro-ordinateurs vendus : c'est le total atteint cet été par Apple toutes machines confondues. Chiffre significatif : la firme à la pomme serait en train de gagner ses paris. A preuve l'article écrit en juillet dans la revue Infoworld par Dave Winer, fondateur de la firme Living Text (l'éditeur de Think Tank). Dave Winer y explique que les clients se bousculent dans les boutiques pour acheter des logiciels pour le Mac, que les développeurs se bagarrent pour lui écrire des programmes et qu'Apple a pris une avance technologique décisive sur IBM.

## Programmes : l'artillerie lourde

Exagérations ? Il est sûr que les ventes de Macintosh ont réellement décollées avec le MacPlus. Il est certain aussi que les programmes se font de plus en plus nombreux, que leur qualité dépasse celle des programmes du PC. Ainsi Excel, dont les amateurs de "Big Blue" attendent encore la transposition sur leur PC favori. Ainsi surtout d'**Ashton Tate**, qui a présenté à la Macworld Expo de Boston la version pour Macintosh de son célèbrissime "**DBase**", la reine des bases de données.

DBase pour le Mac est un gestionnaire de bases de données relationnel avec des relations illustrées comme dans **4ème Dimension** (pour relier entre eux jusqu'à 36 fichiers). Vendu outre Atlantique 495 dollars, ce programme fait une abondante utilisation des icônes et peut récupérer et utiliser les fichiers créés sur DBase II et III sur le PC. Le système comprend un langage utilisant les raffinements des déclarations conditionnelles, des fonctions mathématiques, on peut y créer des fenêtres d'alerte, des menus personnalisés, etc. «Il est supérieur à DBase III Plus pour le PC» a carrément assuré Edward Esber, le président d'Ashton Tate. On avait jadis attendu en vain la consécration du Macintosh par Jazz. Et si Ashton Tate réussissait là où Lotus a échoué ?

## Nouveautés : cris et chuchotements

A l'heure où vous lirez ces lignes, on saura probablement tout du nouvel Apple //. A dire vrai, le seul mystère résidait dans son prix, qu'on pensait difficilement pouvoir être inférieur à 15 000,00 F. Pour le reste, la très haute définition de son écran et ses possibilités sonores en faisaient une sorte d'Amiga à la sauce Apple, capable d'utiliser notamment les programmes existant pour le //.

La question du prix est importante. On s'attend en effet d'ici à Noël à une bagarre comme la micro-informatique n'en avait encore jamais vu. Côté PC, Amstrad et Tandy étaient à la fin juillet sur le point de franchir dans le bon sens (pour l'utilisateur) la limite des 4 000,00 F. Et Atari mijotait un ST utilisant un processeur 68020 (le plus rapide de la famille de celui du Mac), deux mégas de mémoire et le système Unix. Le tout à prix "canon". Dans ces conditions, le prix du nouvel Apple devenait un élément clé de sa survie.

Côté Mac, les rumeurs venues de Californie portaient d'abord sur l'amélioration (on parle d'un doublement et de 16 teintes de gris) de la résolution graphique du Macintosh. Ce problème de l'amélioration de la qualité de l'écran est semble t-il devenu si important qu'Alain Rossmann, et quelques autres génies de chez Apple ont quitté cet été la firme à la pomme pour monter une firme spécialisée dans les moniteurs de très haute qualité. Tiens, tiens...

Il se murmurait aussi que le nouveau Mac modulaire aurait précisément un écran de très haute résolution, et utilisant un processeur 68020, deviendrait pour moins de 5000 dollars un concurrent sérieux pour les "stations de travail" scientifiques du genre Apollo.

## Un outil pour scientifiques

Ce marché paraît si intéressant, que les fabricants de périphériques l'ont déja investi. Dans son numéro d'août, la revue Mac World a consacré sa couverture et tout un article à des améliorations permettant de faire travailler le Macintosh plus rapidement qu'un mini ordinateur Vax.

Le plus impressionnant est un Macintosh *gonflé* par la firme **Levco** et baptisé **Prodigy 4**. Avec notamment un processeur 68020, cadencé à 16 mégahertz, (permettant de disposer d'une puissance d'un MIPS : un million d'instructions par seconde), un coprocesseur 68881 et 4 mégas de mémoire vive. Plus rapide qu'un Vax, pour quelque 7000 dollars. Ce n'est pas donné, mais il paraît que les scientifiques y trouvent leur compte : «on a les avantages d'un Vax, sans être obligé de se le partager à 15» expliquait l'un d'eux au journal Infoworld. Autre Mac gonflé : celui de **Général Computer** : l'**Hyperdrive 2000** déja décrit dans cette rubrique, utilisant lui, un 68000 plus rapide et un coprocesseur 68881. Deux autres firmes ont choisi de se limiter à l'ajout de ce coprocesseur **Quesse**, avec le **Maccelerator**, utilisant un 32081 (Prix : 495 dollars) et Novy avec le même coprocesseur dans un "accélérateur à virgule flottante" (Prix : 449 dollars). A noter que l'accélérateur de Quesse s'installe à l'intérieur d'un Macintosh Plus, ce que ne peut faire celui de Novy.

## Un Mac "pro"

En attendant que ces merveilles fassent partie de l'équipement de base du Macintosh, on peut remarquer que les programmes deviennent à la fois plus scientifiques et plus professionnels.

Les scientifiques sont enfin servis. Avec toute une série de traitements de texte scientifiques. Toute une série sont dans le domaine public. **Weinberg** développé par **Allan Boardio** permet d'éditer et d'imprimer toutes sortes de formules mathématiques à base de $\Sigma$, de $\sqrt{}$, etc. **Alpha Systèmes** importe **Techfont**, une série de police de caractères techniques, (1 200,00 Francs) **SciFont**, des caractères scientifiques (635,00 Francs) et **Logifonts**, comprenant les composants des circuits logiques (510,00 Francs).

A noter que le britannique **Elite Software** a lui aussi réalisé un traitement de texte scientifique s'ajoutant à sa gamme **Format 80** pour les Apple //. Pour les amateurs de mathématiques, un programme permet de résoudre sans douleurs des équations : c'est **Power Math** de **Brain Power**, développé par un ancien élève du MIT et un ancien de Cambridge.

Avec ses possibilités graphiques, le Mac est un parfait outil d'analyse. A preuve, **Cricket Graph**, importé par **KA Informatique**, un grapheur comportant des fonctions statistiques, la possibilité de mixer les graphiques avec des textes et d'imprimer le tout en couleurs sur l'Imagewriter II. Rien de tel que l'image pour analyser les données.

C'est ce qu'a aussi compris **Abvent** avec son **Anatool**. Un outil permettant de modéliser les procédures dans une usine, de définir un dictionnaire de données, des spécifications de processus, de vérification de système, etc. Verra t-on davantage de Mac en usine grâce à ce programme ?

## Mac organisateur d'idées

**Living Videotext**, l'auteur de **Think Tank** a sorti **More**, qui permet de traduire graphiquement en organigramme, toute une hiérarchie de noms ou d'idées. Tandis que fleurissent les imitations. **MaxThink** de **Max Think Inc.** est vendu la moitié du prix de Think Tank : 195 dollars. **Calliope d'Innovision** utilise des icônes représentant les idées formulées dans des fenêtres différentes. Ces icônes peuvent être reliées entre elles. **MacSpec** de **LM Software** permet aussi d'organiser des spécifications ou des idées, en numérotant et indentant automatiquement, en renumérotant lorsque l'on déplace les sections, en établissant automatiquement des tables de matières (Prix : 200 dollars).

## Accessoires de bureau

Et voici surtout **Acta**. Avantage décisif : il s'installe dans le menu Pomme comme un accessoire de bureau. peut organiser des documents avec 2000 niveaux d'idées. Ses dossiers peuvent être enregistrés comme des documents Mac Write (et donc utilisables par Word). Son seul défaut : on ne peut imprimer les idées directement, il faut d'abord les transférer dans Macwrite ou Word. Prix : 60 dollars. **Mainstay** propose aussi, pour exactement le même prix, **Think Now**, lui aussi s'installe dans le menu Pomme et étend ou contracte le plan à la demande.

Les accessoires de bureau du menu Pomme sont décidément bien pratiques, puisque utilisables dans tous les environnements et dans tous les programmes. A noter parmi ceux qui sont particulièrement utiles, toujours chez **Mainstay**, **TypeNow**, qui transforme le Mac en machine à écrire : tout ce qu'on tape est immédiatement et directement imprimé sur l'ImageWriter. Prix raisonnable : 40 dollars. Idéal pour écrire un mémo ou des enveloppes. **Top Desk** de **Cortland Computer**, comprend pour 60 dollars, un éditeur de macros, un spooler permettant à l'imprimante d'imprimer en qualité graphique pendant que l'on continue à travailler, la possibilité d'ouvrir huit fenêtres dans Macwrite, un crypteur de fichiers pour les protéger des yeux indiscrets, un système désactivant l'écran pour le protéger des brûlures, la possibilité de transférer rapidement des éléments entre applications sans utiliser le finder, le tout en accessoires de bureau.

**Alpha Systèmes** importe **Locater**, un accessoire de bureau du menu Pomme qui permet d'afficher l'arborescence conduisant à un document d'après son nom, sa première lettre ou sa date de création. Quasiment indispensable à ceux qui utilisent le HFS avec un disque dur. Prix : 350,00 Francs.

Parmi les autres accessoires de bureau en voici deux bien utiles : **Art Grabber +** distribué par **Hayden** permet de prélever et coller immédiatement un dessin dans Macwrite sans quitter cette application et fonctionne avec MacPlus. **J Clock**, un accessoire du domaine public permet d'installer à demeure au démarrage une horloge visible en permanence dans le coin supérieur droit de tous les programmes.

Autre accessoire amusant qui ne s'installe pas dans le menu Pomme, **Postermaker**, de **Strider Software** permet pour 40 dollars de réaliser des posters de n'importe quel document Mac Paint agrandi jusqu'à 32 fois.

## Traitements de texte

La décision d'Apple de ne plus livrer systématiquement Macwrite avec les Macintosh ne cesse de provoquer des vocations. Du côté des anciens, remarquons d'abord une version 4.6 de Macwrite. Elle comporte seulement comme nouveauté en haut de l'écran une barre avec des icônes définissables par l'utilisateur représentant les choix des menus, et l'affichage de l'heure en permanence. **Word Handler d'ALS** vendu actuellement seulement 30

dollars comprend la possibilité d'ouvrir 4 fenêtres, de se déplacer latéralement comme dans Word, de lire les fichiers Mac Write, un compteur de mots, des abréviations automatiques et le passage automatique majuscules/minuscules.

En France, on attend avec impatience **Writer Plus d'ACI** avec (entre autres) ses possibilités d'écriture en colonnes et sa césure automatique. **Microsoft** a repoussé jusqu'à la fin de l'année sa nouvelle version de **Word** (comprenant notamment un correcteur intégré d'orthographes et un organisateur d'idées) pour laisser toutes ses chances à **Mac Works**, officiellement présenté en août à la MacWorld Expo de Boston.

**Works**, la version pour Mac d'Appleworks devrait faire un sacré tabac. Voici en effet un programme intégré utilisable par tous. Il a su éviter la lourdeur et la lenteur de Jazz grâce à une donnée importante : il travaille en mémoire vive. Les tris sur base de données s'effectuent à une vitesse impressionnante. Le tableur (avec grapheur incorporé) qui ne cède rien à Multiplan permet de disposer de 256 colonnes et 9999 rangées, de disposer de 54 fonctions et de plusieurs fenêtres parmi lesquelles on se déplace comme dans Excel. Le petit programme de communications fonctionne parfaitement et propose une innovation quasi révolutionnaire : les transferts s'effectuent en tâche de fond. En clair, on peut recevoir un fichier important tout en continuant à travailler sur le traitement de texte. Et quel traitement de texte. Il comporte à la fois la facilité de Macwrite, les possibilités de paramétrage des paragraphes de Word avec deux nouveautés : la possibilité de dessiner ou d'encadrer, incorporée à l'application, ainsi que la possibilité d'écrire un texte à côté d'une image ou d'un dessin qu'on a collé dans un texte. Souplesse, facilité, rapidité : Works offre pour 295 dollars, tout ce qu'un utilisateur moyen peut souhaiter. Avec son fonctionnement en RAM, il utilise au mieux les possibilités du Macintosh Plus. Avec d'un côté MacWorks et de l'autre Excel, voici Jazz très dangereusement pris en tenaille entre un intégré facile et un intégrable puissant.

## Du hard

Un peu monstrueuse la modification de **Custom Computer** baptisée **Double DD** installe tout simplement un second lecteur à l'intérieur du Macintosh, sa fente d'introduction se présente sur le côté droit du Mac. Aux États Unis, cette modification coûte 189 dollars pour un lecteur de 400Ko et immobilise le Mac pour deux jours. En France, aucun audacieux n'a encore tenté ça.

Aux États Unis, toujours, **General Computer** présente son premier disque dur... externe : le **FX 20**. Prix 1200 dollars pour 20 Mégas. Branchement sur le port SCSI.

## BDAO ?

Voici pour terminer, un traitement de texte un peu spécial, c'est **Comic Works** écrit par **MacroMind** (les auteurs de Videoworks et Musicworks). Comic Works est tout simplement un éditeur de bandes dessinées combinant les possibilités de MacWrite pour les textes et de Macpaint pour les dessins (avec notamment un aérographe avec buse ajustable). Tout un choix de bulles est prévu pour contenir les textes écrits avec des polices du genre "Vampire" ou "Crypt". Voici pour le grand public le premier programme de BDAO : la bande dessinée assistée par ordinateur.

## Formation

Le succès de son séminaire avancé sur Multiplan (avril 86) a conduit KA a renouveler l'opération le mercredi 8 octobre à l'**American Chamber of Commerce**, toujours en présence de Bernard Vergnes, Président de **Microsoft France**. Ce séminaire d'une journée, animé par Hervé Thiriez, s'adresse à tous les utilisateurs confirmés de Multiplan.

## Adresses

**Levco** - 66160 Lusk Boulevard
San Diego CA 92121

**Quesse Computer Co**
P.O. Box 922 Issaquah
WA 98027

**Novy Systems**
69 Ravenwood
Ct Ormond Beach FL 32074

**Allan Boardio Ass**
2000 Center St n°1225
Berkeley CA 94704

**Elite Software**
4 Hawthylands Drive - Hailsham
East Sussex BN271HE UK

**Alpha Systemes**
29 Bd Gambetta
38000 Grenoble
Tél. : 76 43 19 97

**Brain Power** - 24009 Ventura
Bd Calabasas CA 91302

**Abvent**
9903 Santa Monica Bd suite 268
Beverly Hills CA 90212

**MaxThink Inc.**
230 Crocket Ave, Piedmont
CA 94160

**Innovision**
PO Box 1317 Los Altos
CA 94023

**Symmetry**
761 E University Dr
Ste C. Mesa AZ 85203

**LM Software** - PO Box 93
Belmont CA 94002

**Mainstay** - 28611B Canwood
Agoura CA 91301

**Cortland Computer**
PO Box 9916
Berkeley CA 94709

**Strider Software**
Beecher Lake Road Pembine
Wisconsin WI 54156

**ALS** - 1283 Reamwood Ave
Sunnyvale CA 94089

**Custom Computer**
3601 Parkview Ln n°1C
Irvine CA 92715

**KA Service Formation**
14, rue Magellan - 75008 Paris
Tél.: 47 80 22 23

**American Chamber of Commerce**
21, avenue Georges V
75008 Paris

# Bon de commande

## Disquettes

| | | | |
|---|---|---|---|
| HAIFA source | (cf. Pom's n° 5) | à 60,00 F | |
| MUSIC | (cf. Pom's n° 10) | à 80,00 F | |
| DISK-MANAGER | (cf. Pom's n° 11) | à 450,00 F | |
| BASICIUM | (cf. Pom's n° 13) | à 150,00 F | |
| E.P.E. 5.0 | (cf. Pom's n° 23) | à 200,00 F | |
| Échange E.P.E. 5.0 | (cf. Pom's n° 23) | à 80,00 F | |
| PASCAL | (cf. Pom's n° 15) | à 80,00 F | |
| MAX (Moniteur étendu) | (cf. Pom's n° 18) | à 150,00 F | |
| DOMINOS | (cf. Pom's n° 19) | à 80,00 F | |
| P-FORMAT ][, P-POLICE ][ | (cf. Pom's n° 21) | à 200,00 F | |
| COGO | (cf. Pom's n° 21) | à 150,00 F | |
| LUDOLOGIC | (cf. Pom's n° 25) | à 80,00 F | |
| ORDICO | (cf. page 57) | à 200,00 F | |

## Recueils

| | | |
|---|---|---|
| N°1, recueil des revues 1 à 4 | à 140,00 F | |
| Disquettes d'accompagnement 1 à 4 | à 200,00 F | |
| N°2, recueil des revues 5 à 8 | à 140,00 F | |
| Disquettes d'accompagnement 5 à 8 | à 200,00 F | |
| N°3, recueil des revues 9 à 12 | à 140,00 F | |
| Disquettes d'accompagnement 9 à 12 | à 200,00 F | |

## Revues, disquettes

| | | |
|---|---|---|
| Revues 4 7 8 *(n°9 épuisé)* | à 35,00 F | |
| Revues 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 | à 40,00 F | |
| Disquettes Apple II, //e, //c<br>1/2 3 4 5 6 7 8 9 10 11 12 13<br>14 15 16 17 18 19 20 21 22 23 24 25 26 | à 60,00 F | |
| Disquettes Macintosh | | |
| 14/15/16 groupées | à 150,00 F | |
| 17 18 19 20 21 22 23 24 25 26 | à 80,00 F | |
| Mac 'A' | à 80,00 F | |
| MacAstuces | à 200,00 F | |
| "Raccourci" | à 200,00 F | |

## Abonnements

Pour 6 numéros à partir du n° .....

| | | |
|---|---|---|
| Abonnement à la revue seule | à 200,00 F | |
| Abonnement revue + disquettes Apple II, //e, //c | à 500,00 F | |
| Abonnement revue + disquettes Macintosh | à 600,00 F | |

**Total TTC :**

Supplément avion hors CEE : 15,00F par numéro et/ou disquette :

**Montant du règlement :**

*Envoyez ce bon et votre règlement à :* EDITIONS MEV, 64 rue des Chantiers 78000 VERSAILLES

**Nom :**

**Adresse :**

*Imprimerie Rosay, 94300 Vincennes. Imprimé en France. Dépôt Légal : Septembre 1986. N° 3553*

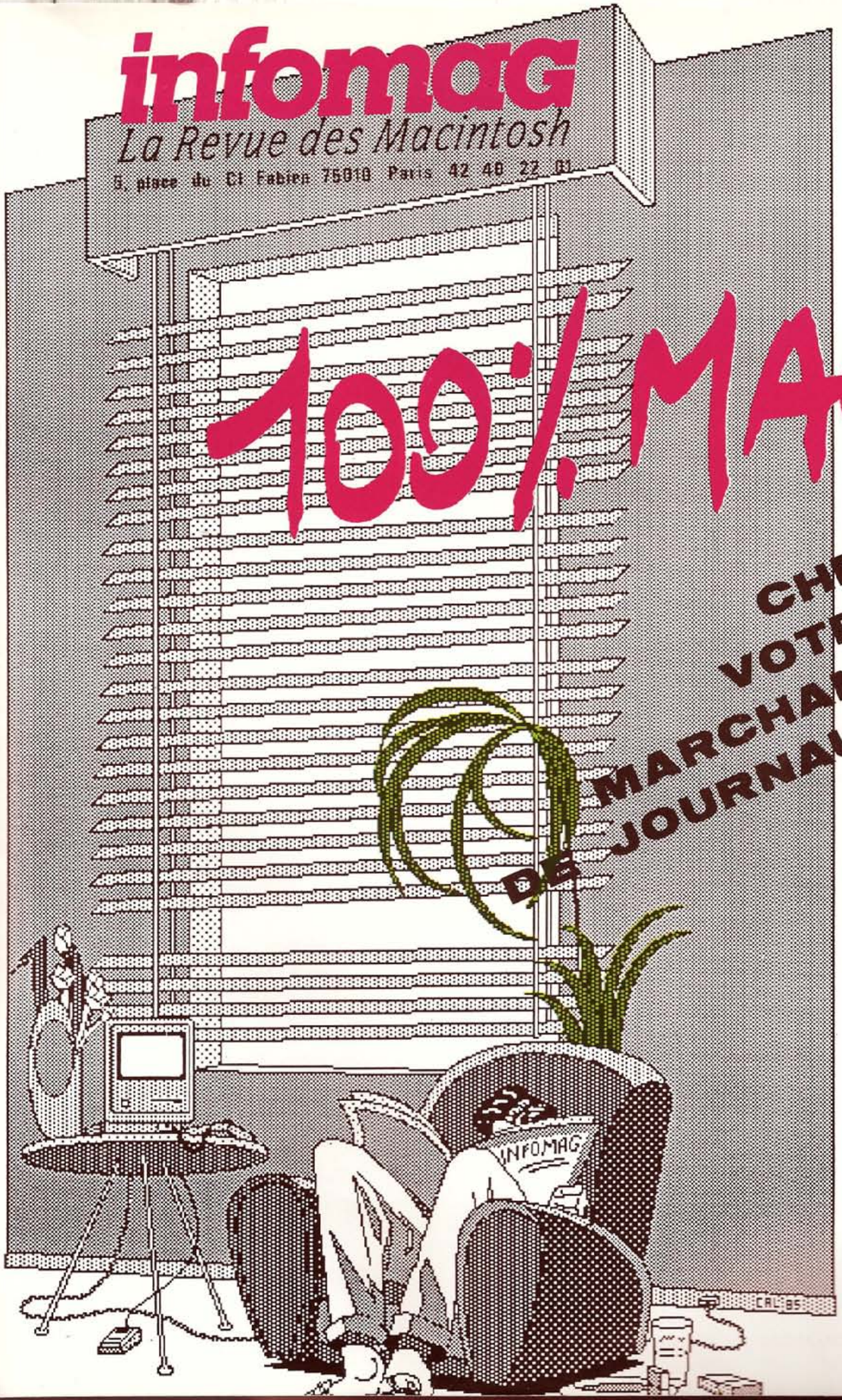
# infomag

*La Revue des Macintosh*

3, place du Cl Fabien 75019 Paris 42 40 22 01

# 100% MAC

CHEZ VOTRE MARCHAND DE JOURNAUX



# P.S.I.
# VISE JUSTE

## PROGRAMMER

**C**lefs pour Macintosch **150** FF – **B**asic Microsoft 2.0 sur Macintosh **250** FF – **B**asic + 80 routines sur Apple II **95** FF – **L**es ressources de l'Apple IIc **95** FF – **A**ssembleur de l'Apple **120** FF – **I**ntroduction à ProDOS sur Apple **85** FF – **S**ystème ProDOS sur Apple **190** FF – **P**rogrammation système de l'Apple II **190** FF – **A**pple, modems et serveurs **130** FF – **C**lefs pour l'Apple IIc et IIe 65CO2 **145** FF.

## UTILISER

**M**ac Astuces **150** FF – **M**ultiplan pour Macintosh **110** FF – **L**e livre de Jazz **220** FF – **5**0 modèles Multiplan pour gérer sur Apple et IBM/PC **130** FF – **A**ppleworks au travail **160** FF – **P**hotographie sur Apple et Amstrad **150** FF.

## DES LIVRES POUR

## CRÉER

**P**rogrammation des jeux d'Arcade sur Apple II **140** FF – **A**pple, logique et systèmes experts **120** FF – **C**réation et animation graphique sur Apple **335** FF.

## JOUER

**1**02 programmes pour Apple **120** FF – **S**uper jeux Apple **120** FF.

**DEMANDER LE CATALOGUE GRATUIT** à P.S.I. Diffusion - B.P. 86 - 77402 LAGNY CEDEX