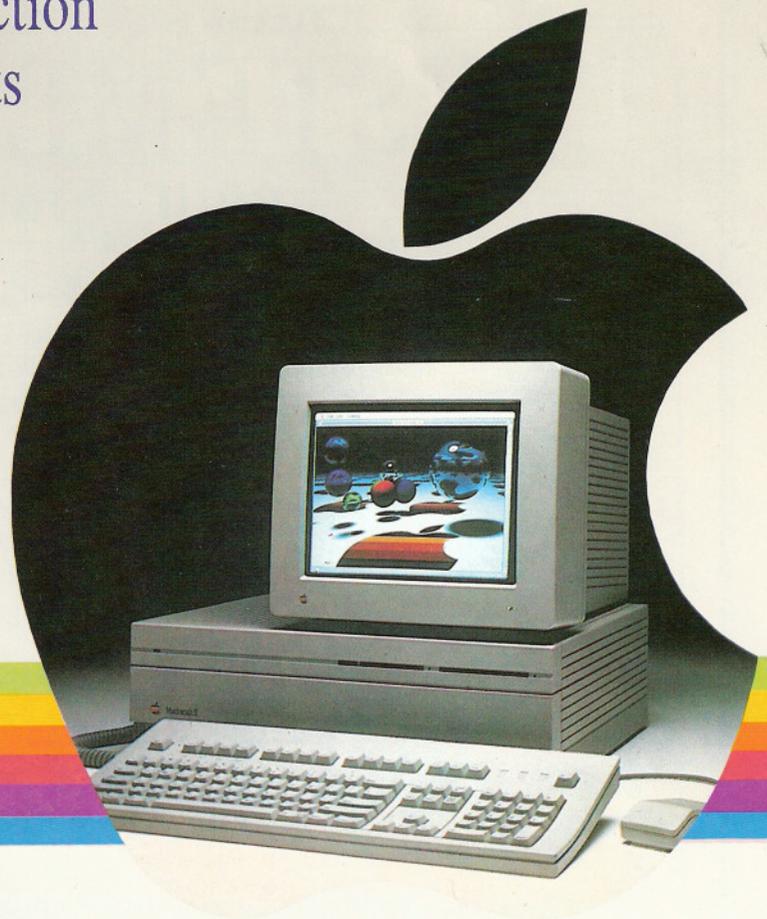


La revue francophone indépendante pour les utilisateurs des
Apple][+, //e, //e+, //c, IIGs™ et Macintosh™

pom's

- 🍏 Les nouveaux Macintosh : Mac SE, Mac II
- 🍏 Cryptage de fichiers confidentiels
- 🍏 Disque virtuel et AppleWriter
- 🍏 Essai 'routier' de PageMaker
- 🍏 Kyram : nibble et protection
- 🍏 ProDOS : fichiers détruits
- 🍏 Conversion HGR/DHGR
- 🍏 Méthode 'Simplexe'
- 🍏 Recherche d'octets
- 🍏 **Minitel & Apple :**
 - graphisme
 - ligne artificielle



M 2366 - 29 - 45,00 F



NUMERO 29 - PRIX 45 F

3792366045005 00290 ISSN : 0294-6068

SOYEZ BRANCHÉS AVEC LES MODEMS OLITEC

- TRANSFORMEZ VOTRE ORDINATEUR EN SUPER MINITEL
- ACCÉDEZ AUX RÉSEAUX NATIONAUX, INTERNATIONAUX
- CRÉEZ VOTRE PROPRE SERVEUR

A PARTIR DE

280F TTC

Apple 2 E, 2 +

- Coffret n° 1 :
 - 1 interface Minitel/Série (vous utilisez le modem du Minitel)
 - 1 interface série RS232
 - 1 logiciel de communication universel
 - 1 émulation Minitel**890,00F TTC**
- Coffret n° 2 :
 - 1 Modem Olitec 12 Modes (V 24, V 21, V 23, Bell 103, Bell 202)
 - 1 interface série RS232
 - 1 logiciel de communication universel
 - 1 émulation Minitel**1 990,00F TTC**

Apple 2 C, 2 GS

- Même configuration sans interface série.
- N° 1 Prix **490,00F TTC**
 - N° 2 Prix **1 590,00F TTC**
 - N° 3 Prix **2 080,00F TTC**

MODEMS, Interfaces

- Modem 12 modes pour Apple 2, 2+, 2 e, 2 c, 2 GS, Mac... **1 490F TTC**
- Modem 16 modes à réponse automatique **1 990F TTC**
- Interface Minitel/Série (RS232 C) **280F TTC**

- Coffret n° 3 :
 - 1 Modem Olitec 16 Modes à réponse automatique (V25, V24, V 21, V 23, Bell 103, Bell 202)
 - 1 interface série RS232
 - 1 logiciel de communication universel
 - 1 émulation Minitel**2 480,00F TTC**



CONTACTEZ-NOUS :

STÉ OLITEC, 20, rue de Remenauville - 54000 NANCY
Téléphone : 83.35.00.65

SYSTEME EXPERT pour Macintosh

Installez un Système Expert aux fonctionnalités professionnelles sur votre Mac (128 K, 512K, 512/800 ou MacPlus) :

- Création, modification, sauvegarde de vos propres «Bases de Connaissances» pouvant contenir jusqu'à 250 Règles exprimées en français courant
- Chaque Règle peut comporter de 1 à 6 conditions
- Liste et Dictionnaire des Bases de Règles et de Faits
- Moteur d'inférence fonctionnant en logique propositionnelle
- Simplicité d'utilisation, exploitant toute la convivialité du Mac
- Logiciel structuré écrit en Microsoft® MBASIC 2.0
- Programme source livré non protégé
- Logiciel proposé complet, port inclus, au prix de

350 F TTC

FONCTIONNE EN

- Chainage avant
- Chainage arrière
- Vérification d'hypothèses
- Modes "déduction" et "vérification" assistées, proposés automatiquement en cas de non-résolution

Commandez dès aujourd'hui votre disquette, contenant le Programme source + Mode d'emploi et Documentation complète + 3 Bases de Connaissances proposées en exemple (que vous pourrez éditer et modifier) en Botanique, Diagnostic automobile et Graphologie

Adressez votre chèque à **BOYER-LARUET, 22 Soudanes, 78430 LOUVECIENNES**

Numéro 29
mars-avril 1987

Éditorial

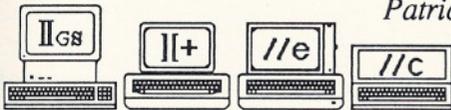
Hervé Thiriez



Page 5

Recherche d'octets

Patrick Covare



Page 6

Apple & Minitel : les caractères semi-graphiques



Page 10

Conversions HGR/DHGR

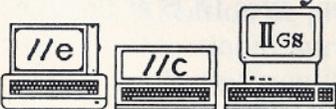
Alexandre
Avrane



Page 11

De l'octet au nibble : Kyram

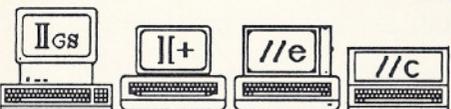
Gildas
Ménier



Page 13

ProDOS et fichiers détruits

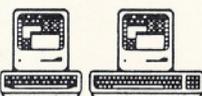
Patrice Neveu



Page 26

PageMaker : essai 'routier'

Philippe Mathieu



Page 40

Les nouveau-nés : Macintosh SE & II



Page 43

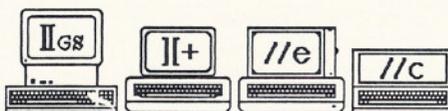
Cryptage de fichiers : Kruptos



Jean-Luc Bazanegue

Page 45

Cryptage de fichiers : Kruptos

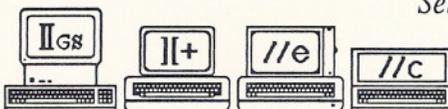


Christian
Piard

Page 53

Simplexe

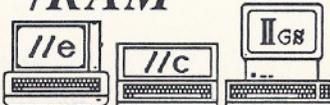
Serge Cattan



Page 57

AppleWriter & /RAM

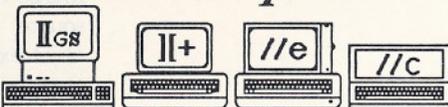
Christian Piard



Page 63

Programme WPL : tabulations automatiques

Bernard Bel



Page 64

Apple & Minitel : ligne 'artificielle'



Jean-Louis Chaulot-Talmon

Page 66

Micro-informations : page 68. Un nouveau produit Pom's, BananaSoft : page 71. Courrier des Lecteurs : page 73.

Les annonceurs ; Apple : pages 38 et 39. Boyer-Larvet : page 2. Olitec : page 2.

Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Tél. : (1) 39 51 24 43. Directeur de la publication : Hervé Thiriez

Disquettes d'accompagnement Apple II : DOS & ProDOS, 800 Ko et 140 Ko

À partir de ce numéro, les disquettes d'accompagnement de la revue vous sont proposées sous deux formes : 140 Ko 5,25 pouces et 800 Ko 3,5 pouces, et ce, pour penser à nos lecteurs qui ont 'craqué' pour le IIGS et qui n'ont plus de drive 140 Ko.

Autre nouveauté, nos disquettes 140 Ko vous sont proposées en **double face** :

- Recto formaté en DOS 3.3
- Verso formaté en ProDOS (nommée /POMS29/)

sans changement de prix : 60,00 F.

La disquette 800 Ko coûte 80,00 F.

Les deux faces de la disquette 140 Ko et la disquette 800 Ko comportent tous les fichiers relatifs à la revue sans tenir compte du

système d'exploitation nécessaire à tel ou tel programme : vous pouvez ainsi récupérer les fichiers plus facilement pour les adapter ou en extraire les routines.

Pour plus de commodité, sur la disquette 800 Ko et sur la face ProDOS de la disquette 140 Ko, mettez les fichiers 'ProDOS' et 'BASIC.SYSTEM' pour que vous puissiez démarrer directement.

Voici la liste des fichiers des disquettes d'accompagnement de ce numéro avec indication du système d'exploitation supporté pour leur fonctionnement. Bien entendu, les sources (tous en format TEXT) peuvent être utilisés indifféremment avec DOS et ProDOS.

Fichier	DOS	ProDOS	Remarques
RECHERCHE	•		objet exécutable (BRUN)
T.RECHERCHE	•	•	source
DHGR.DEMO	•	•	programme de démonstration Basic (RUN ou -)
DHGR	•	•	objet appelé par DHGR.DEMO
DHGR.PIC	•	•	image DHGR (Extasie) utilisée par DHGR.DEMO
T.DHGR	•	•	source
KYRAM	•		objet exécutable (BRUN)
T.KYRAM	•	•	source
INDEX		•	objet exécutable (BRUN ou -)
BLOC		•	objet exécutable (BRUN ou -)
T.INDEX.CODE	•	•	source
T.BLOC.CODE	•	•	source
KRUPTOS		•	objet exécutable (BRUN ou -)
KRUPTOS.S	•	•	source
KRUPTOS.CODE		•	objet non exécutable
SECRET	•	•	fichier à décrypter par vos soins
SIMPLEXE	•		programme Basic (RUN)
PATISSERIE			fichier de démonstration utilisé par Simplexe
WPL.TAB	•	•	programme WPL pour AppleWriter
AW1.C		•	objet à 'bloader' pour patcher AppleWriter
AW2.C		•	objet à 'bloader' pour patcher AppleWriter
AW1	•	•	source
AW2	•	•	source
GRAPH.VIDEOTEX	•	•	programme Basic (RUN)

Éditorial

Enfin, nous venons de voir la sortie du Mac ouvert, baptisé Mac II, et du Mac SE (Super-Entrouvert ?), ainsi que l'apparition de toute une famille de nouveaux produits Mac, présentés dans ce numéro. Le show typiquement "Apple" pour la sortie de ces nouveaux produits nous a convaincu, une fois de plus, que cette société doit détenir le brevet d'une nouvelle unité de mesure informatique, le Mo (*Mégalo-octet*).

Le plaisir de voir apparaître ces nouveaux produits a été quelque peu estompé par la disparition, depuis la sortie de notre dernier numéro, des revues Golden (disparition complète) et Infomag (transmutation ?). Après avoir été la première revue dédiée à une famille de matériels micro, serons-nous la dernière ? Il est certain qu'il en reste aujourd'hui à peine une dizaine, sur les plus de 50 créées depuis le premier numéro de Pom's. Rassurez-vous, l'équipe de Pom's dispose toujours de l'enthousiasme et du dynamisme de tout 'micro-maniaque' mais également de curiosité face aux nouvelles machines et de projets qui se concrétisent au fil des numéros.

Avec ce numéro, nous vous proposons sur la disquette Macintosh 29 (et probablement de façon régulière sur les prochaines) des logiciels domaine public ou Mac Honor. Bien entendu, nous les mettons sur nos disquettes sans en modifier le prix.

Autres nouveautés :

- les disquettes Apple // existent aujourd'hui en 800 Ko ProDOS et 140 Ko DOS et ProDOS... sans modification de prix,
- EPE 5.1 fonctionne sur l'Apple IIgs et existe sur disquette 800 Ko,
- BananaSoft, une nouvelle disquette qui jette un pont entre l'AppleSoft et le Basic Microsoft 5.x.

Nous vous donnons bien entendu rendez-vous à Apple Expo, pas immédiatement, Apple ayant décidé de maintenir le lieu de La Vilette, mais de modifier la date : début octobre.

Hervé Thiriez

Ont collaboré à ce numéro : Alexandre Avrane, Jean-Luc Bazanegue, Bernard Bel, Serge Cattan, Jean-Louis Chaulot-Talmon, Patrick Covare, Jean-Michel Gourévitch, Olivier Herz, Philippe Mathieu, Gildas Ménier, Gérard Michel, Patrice Neveu, Christian Piard, Hervé Thiriez.

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 : Berger-Levrault - 18, rue des Glacis - 54000 Nancy. Tél. : 83.35.61.44.

Pom's est une revue indépendante non rattachée à Apple Computer, Inc. ni à Apple Computer France S.A.R.L. Apple, le logo Apple, Mac et le logo Macintosh sont des marques déposées d'Apple Computer, Inc.

Recherche d'octets

Patrick Covare

Ce programme permet de rechercher, en mémoire ou sur disquette, une expression quelconque.

Il peut être très utile, par exemple pour retrouver tous les appels à une routine quelconque, l'endroit où une mémoire particulière est modifiée, etc.

Écrit en assembleur, le programme est rapide. Les 35 pistes d'une disquette sont examinées en moins de 30 secondes, et une recherche en mémoire ne demande que 2 ou 3 secondes pour 64Ko.

Commandes disponibles

BRUN RECHERCHE lance le programme pour la première fois. Par la suite, on pourra indifféremment utiliser & (l'ampersand) ou Ctrl-Y à partir du moniteur.

Les commandes 'A' et 'Z' permettent de définir les adresses hexadécimales de début et fin de recherche en mémoire vive.

Ces adresses ne peuvent se situer ni dans l'espace \$C000-\$C0FF réservé aux entrées/sorties (sinon bonjour les dégâts!), ni dans la plage \$9500-\$95FF où est conservée l'expression à rechercher. Pendant l'exécution, ces plages sont automatiquement évitées par le programme.

Si l'adresse de départ est supérieure à celle de fin, RECHERCHE boucle sur \$0000 après avoir examiné l'adresse \$FFFF.

Les commandes 'P' et 'F' permettent de saisir, toujours en hexadécimal, les numéros de

pistes de début et fin de recherche sur la disquette. S'ils sont identiques, la recherche ne s'effectuera que sur une seule piste.

La commande 'D' autorise le changement de lecteur (1 ou 2).

La commande 'E' permet d'entrer une expression à rechercher, expression composée au maximum de 256 caractères.

Très pratique : inclure des caractères inconnus (wildcards) en tapant un ou plusieurs '='. Par exemple : "20 == FD" trouvera les appels à COUT (\$FDED), COUT0 (\$FDF0), KEYIN (\$FD1B), etc.

Afin de ne pas surcharger l'écran, seuls les 11 premiers octets sont affichés sur le menu ; néanmoins, l'expression étant conservée à l'adresse \$9500, elle peut être visualisée par un simple appel du moniteur.

Les commandes 'R' et 'S' lancent une recherche en mémoire ou sur disquette.

Dans ce dernier cas, la touche ESCAPE permet d'interrompre la recherche en cours.

Si des expressions identiques sont détectées, leurs adresses, et éventuellement les numéros de pistes et secteurs, sont affichés.

Si rien n'a été trouvé, on revient au menu.

Pour une recherche sur disquette, l'adresse et ses deux derniers chiffres complètent l'affichage.

Exemple :

\$862A P:\$12 S:\$9

informe que l'expression a été trouvée à l'octet \$2A du secteur 9, piste \$12, et qu'elle se trouve

actuellement en mémoire à l'adresse \$862A.

Lors d'une recherche sur la disquette, les pistes sont chargées une à une, dans un buffer situé en \$8000-\$8FFF. Les secteurs y sont stockés dans l'ordre inverse (afin d'améliorer le temps de traitement) : page \$80 pour le secteur \$F, \$81 pour \$E, jusqu'à \$8F pour le secteur 0.

Bien entendu, le programme se charge également de l'affichage lorsque l'expression est trouvée plusieurs fois sur la même page ou le même secteur (ce qui n'est pas le cas de Nibbles Away).

La commande 'M' dirige vers le moniteur de l'Apple (retour par Ctrl-Y).

La commande 'Q' permet de quitter le programme.

Fonctionnement technique

La configuration mémoire suivante est utilisée :

\$0000...\$000A

vecteurs page zéro ;

\$8000...\$8FFF

buffer de chargement d'une piste ;

\$9000...\$94A0

implantation du programme ;

\$9500...\$95FF

buffer de l'expression à rechercher ;

\$9600...\$96FF

buffer des caractères inconnus.

Le listing source est amplement commenté et ne devrait pas poser de problème de compréhension. Cependant, il est bon de détailler l'algorithme de recherche.

Le sous-programme de comparaison utilise les buffers EXPRESS et WILD, initialisés lors de la saisie de l'expression. EXPRESS contient la suite d'octets à rechercher, WILD stocke l'emplacement des caractères inconnus.

Prenons un exemple. Soit l'expression saisie "A8 =B F3". Le buffer WILD, utilisé pour

effectuer un ET logique, contient la valeur 'FF0FFF'. Le zéro indique la présence d'un caractère inconnu. Le buffer EXPRESS contient les codes 'A80BF3'. Les espaces tapés au clavier ont été supprimés, et le troisième caractère a été remplacé par un zéro.

Si, lors de la recherche, le

programme rencontre la suite d'octets 'A85BF3', celle-ci est considérée comme identique car :

A8 AND FF = A8
5B AND 0F = 0B
F3 AND FF = F3

ce qui correspond exactement au contenu du buffer EXPRESS.



Source 'T.RECHERCHE' Assembleur BigMac

```

1 *****
2 *
3 *      -- RECHERCHE --
4 *
5 * PATRICK COVARE      02/1986
6 *
7 *      Assembleur Big Mac
8 *****
9
10
11 MDEBUT = $0      ;Adresse de debut de recherche memoire
12 MFIN   = $2      ;Adresse de fin de recherche memoire
13 PDEBUT = $4      ;Piste de debut de recherche sur disquette
14 MDRIVE = $5      ;Numero du lecteur en service
15 PFIN   = $6      ;Piste de fin de recherche sur disquette
16 LONG   = $7      ;Longueur de l'expression a rechercher
17 OPTION = $8      ;Numero de l'option choisie multipliee par 2
18 TOUR   = $9      ;Compteur utilise par plusieurs routines
19 LG      = $A      ;Memoire utilisee par la routine MENU
20 CH      = $24
21 PROMPT  = $33
22 A2L     = $3E
23 A2H     = $3F
24 LOCRTS  = $3E3
25 AMPERS  = $3F5
26 CTRL Y  = $3F8
27 EXPRESS = $9500  ;Buffer contenant l'expression a rechercher
28 WILD    = $9600  ;Buffer de position des caracteres inconnus
29 RMTS    = $B7B5
30 RNDRV   = $B7EA
31 RNVLM   = $B7EB
32 RNPST   = $B7EC
33 RNSCT   = $B7ED
34 RNBUF   = $B7F0
35 RNREAD  = $B7F4
36 PRNTAX = $F941
37 BLANK   = $F94A
38 TABV    = $FB5B
39 HOME    = $FC58
40 CLREOL  = $FC9C
41 RDKEY   = $FDOC
42 GETLN   = $FD6A
43 GETEXP  = $FD6F
44 CROUT   = $FD8E
45 PRBYTE  = $FDDA
46 PRHEX  = $FDE3
47 COUT    = $FDED
48 BELL    = $FF3A
49 MONZ    = $FF69
50 GETNUM  = $FFA7
51 ZMODE   = $FFC7
52
53
54 ORG $9000
55 OBJ $9000
56
57
58 LDX #7      ;Chargement de la chaine d'initialisation
59 INIT LDA CODINIT,X ;pour le 1er affichage du menu.
60 STA MDEBUT,X
61 DEX
62 BPL INIT
63 *
64 *****
65 *
66 BEGIN LDA #$4C  ;Initialisation des vecteurs
67 STA AMPERS ;AMPERSAND et
68 STA CTRL Y ;CTRL Y
69 LDA #BEGIN
70 STA AMPERS+1
71 STA CTRL Y+1
72 LDA #>BEGIN
73 STA AMPERS+2
74 STA CTRL Y+2
75 JSR MENU ;Affichage du menu,
76 CHOIX JSR RDKEY ;puis lecture d'une touche.
77 LDX #9
78 VERIF CMP JOB,X ;La touche enfoncee est-elle valide?
79 BEQ BON ;Oui, on saute a BON.

```

```

80 DEX ;Non, on poursuit le test.
81 BPL VERIF
82 JSR BELL ;La touche etait incorrecte --> BIP,
83 JMP CHOIX ;puis retour pour une autre lecture clavier.
84 BON TXA ;Le numero de ligne de l'option choisie
85 ASL ;est multipliee par 2,
86 STA OPTION ;et sauvegarde en memoire.
87 *
88 *****
89 *
90 JOBZPF CPX #4 * ROUTINE DE MODIF ADRESSES MEMOIRE ET PISTES *
91 BCS JOBZ ;Saut a JOBZ si option <> A, Z, P ou F.
92 ADC #5
93 JSR TABV ;Initialisation verticale,
94 LDA #20 ;et horizontale du curseur.
95 STA CH
96 LDA #"$"
97 STA PROMPT ;Modif du prompteur pour l'affichage d'un $.
98 JSR GETLN ;Lecture du clavier.
99 JSR ZMODE
100 JSR GETNUM ;Sauvegarde dans A2L,A2H.
101 DEY ;A-t-on entre une chaine vide?
102 BEQ BEGIN ;Oui, retour au menu.
103 LDX OPTION
104 CPX #4 ;Option choisie = A ou Z ?
105 BCC Z1 ;Oui, saut a Z1.
106 LDA A2L ;Non, il s'agit donc d'une modif de piste,
107 CMP #523 ;et on verifie le numero indique.
108 BCC Z2 ;Branchement en Z2 si piste < 35.
109 ERREUR JSR BELL ;Sinon BIP,
110 JMP BEGIN ;puis retour au menu.
111 Z1 LDA A2H
112 CMP #595 ;Partie haute de l'adresse = 595 ?
113 BEQ ERREUR ;C'est interdit --> BIP.
114 CMP #5C0 ;Partie haute de l'adresse = 5C0 ?
115 BEQ ERREUR ;C'est egalement interdit --> BIP.
116 STA MDEBUT+1,X ;Sauvegarde
117 LDA A2L ;de
118 Z2 STA MDEBUT,X ;l'adresse,
119 JMP BEGIN ;et retour au menu.
120 *
121 *****
122 *
123 JOBZ CPX #5 * ROUTINE DE MODIF DU NUMERO DE LECTEUR *
124 BCS JOBE ;Saut a JOBE si option <> D.
125 DEC MDRIVE ;Si MDRIVE=2 --> MDRIVE=1,
126 BNE DRIVE ;et inversement.
127 LDA #2
128 STA MDRIVE
129 DRIVE LDA MDRIVE ;Affichage
130 ORA #5B0 ;en mode normal
131 STA $6CF ;du numero de lecteur,
132 JMP CHOIX ;et retour pour une autre lecture du clavier.
133 *
134 *****
135 *
136 JOBE CPX #6 * ROUTINE SAISIE DE L'EXPRESSION A RECHERCHER *
137 BCS JOBR ;Saut a JOBR si option <> E.
138 LDA #15
139 JSR TABV ;Initialisation verticale,
140 LDA #8 ;et horizontale du curseur.
141 STA CH
142 JSR CLREOL ;L'ancienne expression est effacee.
143 JSR GETEXP ;Lecture du clavier.
144 DEX
145 BMI STOP ;Retour au menu si on entre une chaine vide.
146
147 LDX #0 ;VERIF DE CHAQUE CARACTERE DE L'EXPRESSION
148 LDY #0
149 ENCORE LDA #2
150 STA TOUR ;TOUR=2 --> debut de verification d'un octet.
151 LDA #0
152 STA EXPRESS,X
153 LDA #5FF
154 STA WILD,X
155 IN LDA $200,Y ;Chargement d'un caractere.
156 INY
157 CMP #5A0 ;Est-ce un espace?
158 BEQ IN ;Oui, on passe au caractere suivant.
159 CMP #"-";Est-ce un caractere inconnu?
160 BEQ EGAL ;Oui, on saute a EGAL.
161 EOR #5B0
162 CMP #50A ;Est-il compris entre 0 et 9 ?

```

DOS 3.3

II+
//e
//e+
//c
//gs

```

163      BCC DIGIT      ;Oui, on saute a DIGIT.
164      ADC #588
165      CMP #5FA      ;Est-il compris entre A et F ?
166      BCS DIGIT      ;Oui, saut a DIGIT.
167      STX LONG      ;Non, on sauvegarde la longueur de l'expression,
168      JMP BEGIN      ;et on retourne au menu.
169      AND #50F
170      DEC TOUR      ;Si TOUR=0, un 1er caract a deja ete verifie,
171      BEQ DEUX      ;et on se branche alors a DEUX.
172      STA EXPRESS,X  ;Sinon, on sauvegarde le 1er caractere,
173      JMP IN        ;puis on saute a IN pour le verif du 2eme.
174      DEC TOUR
175      BEQ EGAL2     ;Saut a EGAL2 si TOUR=0.
176      LDA #50F
177      STA WILD,X    ;On charge le buffer WILD avec 'OF',
178      JMP IN        ;et on saute a IN pour le test du 2eme caract.
179      EGAL2      LDA WILD,X
180      AND #5F0
181      STA WILD,X    ;Modification du buffer WILD,
182      LDA #0        ;et 0 dans l'accu pour modif du buffer EXPRESS.
183      DEUX      PHA
184      LDA EXPRESS,X
185      ASL          ;Decalage a gauche du 1er caractere
186      ASL          ;stocke precedemment dans EXPRESS.
187      ASL
188      ASL
189      STA EXPRESS,X
190      PLA
191      CLC
192      ADC EXPRESS,X
193      STA EXPRESS,X ;Sauvegarde du second caractere,
194      INX
195      JMP ENCORE   ;et verification de l'octet suivant.
196 *
197 *****
198 *
199      JOBR      CPX #7      * ROUTINE DE RECHERCHE EN MEMOIRE *
200      BCS JOBS      ;Saut a JOBS si option <> R.
201      LDA LONG
202      BEQ RT1      ;Retour menu si expr source de longueur nulle.
203      LDA MDEBUT
204      STA TSTA+1   ;Initialisation
205      LDA MDEBUT+1 ;avec les parametres
206      STA TSTA+2   ;definis par l'utilisateur.
207      LDA MFIN
208      STA FINTST+1
209      LDA MFIN+1
210      STA FINTST+8
211      JSR TEST     ;Saut a la routine de comparaison d'octets.
212      RT1      JMP BEGIN
213 *
214 *****
215 *
216      JOBS      CPX #8      * ROUTINE DE RECHERCHE SUR LA DISQUETTE *
217      BCS JOBM      ;Saut a JOBM si option <> S.
218      LDA LONG
219      BEQ RT2      ;retour menu si expr source de longueur nulle.
220      LDA MDRIVE
221      STA RWDRV     ;Initialisation
222      LDA PDEBUT    ;des routines RWTS et TEST
223      STA RWPST     ;avec les parametres
224      LDY #0        ;definis par l'utilisateur.
225      STY RWLM
226      STY RWBUF
227      STY TSTA+1
228      STY FINTST+1
229      INY
230      STY RWREAD    ;RWTS en mode lecture.
231      LDA #580      ;Adresse haute du buffer de data RWTS.
232      STA RWBUF+1
233      LDA #590      ;Adresse haute de fin de la recherche.
234      STA FINTST+8
235      JMP VRFPST
236      DISK      LDA #5F      ;Numero du 1er secteur a lire.
237      STA RWSCT
238      READ      JSR LCCRNTS  ;Lecture d'un secteur.
239      JSR RWTS
240      BCS ERR      ;Branchement a ERR en cas d'anomalie.
241      INC RWBUF+1   ;Incrementation du buffer de lecture.
242      DEC RWSCT     ;Decrementation du numero de secteur a lire.
243      BPL READ     ;Saut si lecture piste pas encore terminee.
244      LDA #580      ;Adresse haute du buffer de data RWTS.
245      STA TSTA+2
246      STA RWBUF+1
247      JSR TEST     ;Saut a la routine de comparaison d'octets.
248      LDA $C000     ;A-t-on appuye sur une touche?
249      BPL INCPST   ;Non, on passe a la piste suivante.
250      BIT $C010
251      CMP #59B     ;S'il s'agit de la touche ESCAPE,
252      BEQ RT2      ;on retourne au menu.
253      INCPST     INC RWPST   ;Incrementation du numero de piste.
254      VRFPST     LDA PFIN
255      CMP RWPST    ;Faut-il lire d'autres pistes?
256      BCS DISK     ;Oui, saut a DISK.
257      JMP BEGIN    ;Non, on retourne au menu.
258      ERR      JSR HOME
259      LDA #IOERR   ;Affichage d'un message d'erreur.
260      LDX #>IOERR
261      JSR AFFICHE
262      JSR RDKEY    ;Lecture d'une touche,
263      RT2      JMP BEGIN    ;puis retour au menu.
264 *
265 *****
266 *
267      JOBM      CPX #9      * ROUTINE D'APPEL DU MODE MONITEUR *
268      BEQ JOBQ     ;Saut a JOBQ si option <> M.
269      LDA #RTMON   ;Initialise le vecteur CTRL Y,
270      STA CTRL+1   ;pour un retour a RTMON.
271      LDA #>RTMON
272      STA CTRL+2
273      JSR HOME
274      JSR COUT
275      JMP MONZ     ;Entree dans le moniteur.
276      PLA         ;Reinitialise le pointeur de pile.
277      PLA
278      LDA OPTION
279      CMP #16      ;Le moniteur a-t-il ete appele via le menu?
280      BEQ RT2     ;Oui, on retourne au depart.
281      JMP QUEST    ;Non, on se branche dans la routine de test.
282 *
283 *****
284 *
285      JOBQ      JSR HOME    * ROUTINE DE SORTIE DU PROGRAMME *
286      JMP $3D0
287 *
288 *****
289 *
290      TEST     LDA #0      * ROUTINE DE COMPARAISON DES OCTETS *
291      STA TOUR   ;Memorise le nombre d'expressions retrouvees.
292      LDY #0
293      TSTA     LDA $FFFF,Y ;Charge un octet a tester,
294      AND WILD,Y ;modifie l'accu si c'est un caract inconnu,
295      CMP EXPRESS,Y ;puis compare avec l'expression source.
296      BNE INCR   ;Saut a INCR si ce n'est pas identique.
297      INY
298      CPY LONG   ;Expression source entierement testee?
299      BNE TSTA   ;Non, on continue les comparaisons.
300      LDA TOUR
301      BNE PRTADR ;Pas d'effacement d'ecran si TOUR > 0.
302      JSR HOME
303      PRTADR   LDA #5"
304      JSR COUT  ;Affiche un $.
305      LDA TSTA+2
306      LDX TSTA+1
307      JSR PRNTAX ;Affiche l'adresse de l'expression retrouvee.
308      LDA OPTION
309      CMP #12
310      BEQ CR    ;Saut a CR s'il s'agit d'une recherche memoire.
311      LDA #PS
312      LDX #>PS
313      JSR AFFICHE ;Affiche 'P.$' et '$.$'.
314      LDA #13
315      STA CH     ;Place le curseur en colonne 13.
316      LDA RWPST
317      JSR PRBYTE ;Affiche le numero de la piste.
318      LDA #23
319      STA CH     ;Place ensuite le curseur en colonne 23.
320      SEC
321      LDA #59F    ;Adresse haute du 16eme secteur lu par RWTS.
322      SBC TSTA+2 ;Calcule et
323      JSR PRHEX  ;affiche le numero du secteur.
324      CR      JSR CROUT    ;Saute une ligne.
325      INC TOUR
326      LDA TOUR
327      CMP #22    ;22 expressions ont-elles ete retrouvees?
328      BEQ QUEST  ;Oui, saut a QUEST.
329      INCR      INC TSTA+1 ;Incremente la partie basse de l'adr de test,
330      BNE T1     ;et saute a T1 si le resultat n'est pas nul.
331      INCR2     INC TSTA+2 ;Incremente la partie haute de l'adr de test.
332      LDA TSTA+2
333      CMP #595   ;Sommes-nous dans le buffer EXPRESS?
334      BEQ INCR2  ;Oui, on passe a l'adresse suivante.
335      CMP #5C0   ;Sommes-nous dans les E/S ($C000..$C0FF)?
336      BEQ INCR2 ;Oui, on passe a l'adresse suivante.
337      T1      LDA TSTA+1
338      CMP #500   ;Est-ce la fin de la zone a tester?
339      BNE TSTY
340      LDA TSTA+2
341      CMP #500
342      BNE TSTY   ;Non, on continue les comparaisons.
343      LDA TOUR   ;Des expressions ont-elles ete retrouvees?
344      BEQ RT3    ;Non, retour au programme principal.
345      QUEST     LDA #KBDCHX ;Affiche 'M-MONITEUR ...'
346      LDX #>KBDCHX
347      JSR AFFICHE
348      JSR RDKEY  ;Lecture d'une touche.
349      CMP #5"    ;Est-ce un M ?
350      BNE T2
351      JMP JOBM   ;Oui, saut a la routine moniteur.
352      T2      CMP #59B   ;Est-ce la touche ESCAPE ?
353      BNE CONT  ;Non, saut a CONT pour continuer.
354      PLA      ;Oui, on annule l'adresse de retour,
355      PLA
356      JMP BEGIN ;et on se rebranche au depart.
357      CONT     LDA TOUR
358      CMP #22   ;La fin du test est-elle terminee?
359      BNE RT3   ;Oui, retour au programme principal.
360      LDA #0    ;Non, il reste encore des octets a tester,
361      STA TOUR
362      JMP INCR  ;alors, on continue les comparaisons.
363      RT3      RTS
364 *
365 *****
366 *
367      MENU     LDA #0      * ROUTINE D'AFFICHAGE DU MENU *
368      STA CH    ;On place le curseur a gauche
369      JSR TABV  ;et en haut de l'ecran,
370      LDX #10
371      JSR BLANK
372      LDA #AFFMENU ;puis on affiche le menu.

```

```

373 LDX #>AFFMENU
374 JSR AFFICHE
375 LDA #13
376 STA CH
377 JSR ECRAN
378
379 LDA #5 ;AFFICHAGE DES PARAMETRES ADRESSES ET PISTES
380 JSR TABV ;Le curseur est place sur la ligne 5.
381 LDX #0
382 LDY #35
383 DOLLAR STY CH ;Initialisation horizontale de curseur.
384 LDA #S"
385 JSR COUT ;Affichage d'un $.
386 CPX #4
387 BCS TWO ;Saut si le parametre ne comporte que 2 caract.
388 LDA MDEBUT+1,X
389 JSR PRBYTE ;Affiche 2 caracteres.
390 TWO LDA MDEBUT,X
391 JSR PRBYTE ;Affiche 2 caracteres,
392 JSR CROUT ;et saute une ligne.
393 INX
394 INX
395 CPX #4
396 BCC DOLLAR ;Saut si le 2eme parametre pas encore affiche.
397 LDY #37
398 CPX #8
399 BCC DOLLAR ;Saut si le 4eme parametre pas encore affiche.
400
401 LDA MDRIVE ;AFFICHAGE DU NUMERO DE LECTEUR EN SERVICE
402 ORA #SBO
403 STA S6CF
404
405 LDY LONG ;AFFICHAGE DE L'EXPRESSION SOURCE
406 BEQ CURS ;Saut a CURS si expression de longueur nulle.
407 CPY #12 ;Expression < 12 octets?
408 BCC INF ;Oui, on saute a INF.
409 LDY #11 ;Non, on limite l'affich aux 11 premiers octets.
410 INF STY LG
411 LDA #15
412 JSR TABV ;On place le curseur sur la ligne 15,
413 LDA #41
414 SEC ;puis on calcule,
415 SBC LG
416 SBC LG ;selon la longueur de l'expression,
417 SBC LG
418 STA CH ;la position horizontale du curseur.
419 LDX #0
420 ENCOR LDA WILD,X ;Debut d'affichage d'un octet de l'expression.
421 PHA
422 BMI P1 ;Saut a P1 si le 1er caract de l'octet <> '-'.
423 LDA #"-
424 JSR COUT ;Affiche le signe = .
425 JMP P2
426 P1 LDA EXPRESS,X
427 LSR
428 LSR
429 LSR
430 LSR
431 JSR PRHEX ;Affichage du 1er caractere.
432 P2 PLA
433 ROR
434 BCS P3 ;Saut a P3 si le 2eme caract de l'octet <> '-'.
435 LDA #"-
436 JSR COUT ;Affiche le signe = .
437 JMP P4
438 P3 LDA EXPRESS,X
439 JSR PRHEX ;Affichage du 2eme caractere.
440 P4 INC CH ;Saut a un espace.
441 INX
442 DEY
443 BNE ENCOR ;Saut si l'affich de l'expr n'est pas termine.
444 CURS LDA #35
445 STA CH ;Place le curseur en bas du menu,
446 JMP TABVBAS ;puis retour via le RTS de TABVBAS.
447 *
448 *****
449 *
450 AFFICHE STA ECRAN+1 * ROUTINE D'AFFICHAGE SUR L'ECRAN *
451 STX ECRAN+2 ;A et X contiennent l'adresse du message.
452 LDX #0
453 ECRAN LDA $FFFF,X ;Chargement d'un caractere a afficher.
454 BPL DERNIER ;Si bit 7=0, il s'agit du dernier caractere.
455 CMP #",
456 BNE NON
457 JSR CLREOL ;Si c'est une virgule on efface la fin de ligne,
458 LDA #SBD ;et on charge l'accu pour un saut de ligne.
459 NON JSR COUT ;Affichage du caractere.
460 INX
461 JMP ECRAN
462 DERNIER ORA #SBO
463 JSR COUT ;Le dernier caract est affiche en mode normal,
464 INX
465 JMP CLREOL ;la fin de ligne effacee, puis RTS via CLREOL.
466 *
467 *****
468 *
469 CODINIT HEX 0000FFFF0012200
470
471 JOB ASC "AZPFDRSMQ"
472
473 AFFMENU DCI "- RECHERCHE D'OCTETS -., "
474 ASC "(PATRICK COVARE),,,"
475 ASC "A - MEMOIRE DEBUT,,"
476 ASC "2 - MEMOIRE FIN,,"
477 ASC "P - PISTE DEBUT,,"
478 ASC "F - PISTE FIN,,"
479 ASC "D - DRIVE,,"
480 ASC "E - EXP,,"
481 ASC "R - RECH MEMOIRE,,"
482 ASC "S - RECH DISQUETTE,,"
483 ASC "M - MONITEUR,,"
484 DCI "Q - QUITTER VOTRE CHOIX:"
485
486 PS DCI " P.$ S.$"
487
488 KBDCHX DCI ",M=MONITEUR SP=CONTINUER ESC=MENU "
489
490 IOERR HEX 87
491 DCI "ANOMALIE LECTURE DISQUETTE,,TAPEZ UNE TOUCHE "
492 TABVBAS LDA #S17
493 JMP SFB5B

```

Récapitulation 'RECHERCHE'

Après avoir saisi ce code sous
moniteur, vous le sauvegarderez par :
BSAVE RECHERCHE,A\$9000,L\$4A1

```

9000:A2 07 BD 46 93 95 00 CA
9008:10 F8 A9 4C 8D F5 03 8D
9010:F8 03 A9 0A 8D F6 03 8D
9018:F9 03 A9 0A 8D F7 03 8D
9020:FA 03 20 7D 92 20 0C FD
9028:A2 09 DD 4E 93 F0 09 CA
9030:10 F8 20 3A FF 4C 25 90
9038:8A 0A 85 08 E0 04 B0 3E
9040:69 05 20 5B FB A9 14 85
9048:24 A9 A4 85 33 20 6A FD
9050:20 C7 FF 20 A7 FF 88 F0
9058:B1 A6 08 E0 04 90 0C A5
9060:3E C9 23 90 14 20 3A FF
9068:4C 0A 90 A5 3F C9 95 F0
9070:F4 C9 C0 F0 95 01 A5
9078:3E 95 00 4C 0A 90 E0 05
9080:B0 12 C6 05 D0 04 A9 02

```

```

9088:85 05 A5 05 09 B0 8D CF
9090:06 4C 25 90 E0 06 B0 7A
9098:A9 0F 20 5B FB A9 08 85
90A0:24 20 9C FC 20 6F FD CA
90A8:30 2C A2 00 A0 00 A9 02
90B0:85 09 A9 00 9D 00 95 A9
90B8:FF 9D 00 96 B9 00 02 C8
90C0:C9 A0 F0 F8 C9 BD F0 1D
90C8:49 B0 C9 0A 90 0B 69 88
90D0:C9 FA B0 05 86 07 4C 0A
90D8:90 29 0F C6 09 F0 1C 9D
90E0:00 95 4C BC 90 C6 09 F0
90E8:08 A9 0F 9D 00 96 4C BC
90F0:90 BD 00 96 29 F0 9D 00
90F8:96 A9 00 48 BD 00 95 0A
9100:0A 0A 0A 9D 00 95 68 18
9108:7D 00 95 9D 00 95 E8 4C
9110:AE 90 E0 07 B0 1E A5 07
9118:F0 17 A5 00 8D DD 91 A5
9120:01 8D DE 91 A5 02 8D 47
9128:92 A5 03 8D 4E 92 20 D6
9130:91 4C 0A 90 E0 08 B0 76
9138:A5 07 F0 6F A5 05 8D EA
9140:B7 A5 04 8D EC B7 A0 00
9148:8C EB B7 8C F0 B7 8C DD

```

```

9150:91 8C 47 92 C8 8C F4 B7
9158:A9 80 8D F1 B7 A9 90 8D
9160:4E 92 4C 94 91 A9 0F 8D
9168:ED B7 20 E3 03 20 B5 B7
9170:B0 2C EE F1 B7 CE ED B7
9178:10 F0 A9 80 8D DE 91 8D
9180:F1 B7 20 D6 91 AD 00 C0
9188:10 07 2C 10 C0 C9 9B F0
9190:1A EE EC B7 A5 06 CD EC
9198:B7 B0 CA 4C 0A 90 20 58
91A0:FC A9 6C A2 94 20 20 93
91A8:20 0C FD 4C 0A 90 E0 09
91B0:F0 1E A9 C5 8D F9 03 A9
91B8:91 8D FA 03 20 58 FC 20
91C0:ED FD 4C 69 FF 68 68 A5
91C8:08 C9 10 F0 DE 4C 55 92
91D0:20 58 FC 4C D0 03 A9 00
91D8:85 09 A0 00 B9 FF FF 39
91E0:00 96 D9 00 95 D0 49 C8
91E8:C4 07 D0 F0 A5 09 D0 03
91F0:20 58 FC A9 A4 20 ED FD
91F8:AD DE 91 AE DD 91 20 41
9200:F9 A5 08 C9 0C F0 1E A9
9208:33 A2 94 20 20 93 A9 0D
9210:85 24 AD EC B7 20 DA FD

```

9218:A9 17 85 24 38 A9 8F ED
 9220:DE 91 20 E3 FD 20 8E FD
 9228:E6 09 A5 09 C9 16 F0 25
 9230:EE DD 91 D0 0E EE DE 91
 9238:AD DE 91 C9 95 F0 F6 C9
 9240:CO F0 F2 AD DD 91 C9 00
 9248:D0 90 AD DE 91 C9 00 D0
 9250:89 A5 09 F0 27 A9 45 A2
 9258:94 20 20 93 20 0C FD C9
 9260:CD D0 03 4C AE 91 C9 9B
 9268:D0 05 68 68 4C 0A 90 A5
 9270:09 C9 16 D0 07 A9 00 85
 9278:09 4C 30 92 60 A9 00 85
 9280:24 20 5B FB A2 0A 20 4A
 9288:F9 A9 58 A2 93 20 20 93
 9290:A9 0D 85 24 20 28 93 A9
 9298:05 20 5B FB A2 00 A0 23
 92A0:84 24 A9 A4 20 ED FD E0
 92A8:04 B0 05 B5 01 20 DA FD
 92B0:B5 00 20 DA FD 20 8E FD
 92B8:E8 E8 E0 04 90 E2 A0 25
 92C0:E0 08 90 DC A5 05 09 B0
 92C8:8D CF 06 A4 07 F0 4A C0
 92D0:0C 90 02 A0 0B 84 0A A9
 92D8:0F 20 5B FB A9 29 38 E5
 92E0:0A E5 0A E5 0A 85 24 A2
 92E8:00 BD 00 96 48 30 08 A9
 92F0:BD 20 ED FD 4C 01 93 BD

92F8:00 95 4A 4A 4A 4A 20 E3
 9300:FD 68 6A B0 08 A9 BD 20
 9308:ED FD 4C 13 93 BD 00 95
 9310:20 E3 FD E6 24 E8 88 D0
 9318:D0 A9 23 85 24 4C 9A 94
 9320:8D 29 93 8E 2A 93 A2 00
 9328:BD FF FF 10 10 C9 AC D0
 9330:05 20 9C FC A9 8D 20 ED
 9338:FD E8 4C 28 93 09 80 20
 9340:ED FD E8 4C 9C FC 00 00
 9348:FF FF 00 01 22 00 C1 DA
 9350:D0 C6 C4 C5 D2 D3 CD D1
 9358:AD A0 D2 C5 C3 C8 C5 D2
 9360:C3 C8 C5 A0 C4 A7 CF C3
 9368:D4 C5 D4 D3 A0 AD AC AC
 9370:20 A8 D0 C1 D4 D2 C9 C3
 9378:CB A0 C3 CF D6 C1 D2 C5
 9380:A9 AC AC AC C1 A0 AD A0
 9388:CD C5 CD CF C9 D2 C5 A0
 9390:C4 C5 C2 D5 D4 AC AC DA
 9398:A0 AD A0 CD C5 CD CF C9
 93A0:D2 C5 A0 C6 C9 CE AC AC
 93A8:D0 A0 AD A0 D0 C9 D3 D4
 93B0:C5 A0 C4 C5 C2 D5 D4 AC
 93B8:AC C6 A0 AD A0 D0 C9 D3
 93C0:D4 C5 A0 C6 C9 CE AC AC
 93C8:C4 A0 AD A0 C4 D2 C9 D6

93D0:C5 AC AC C5 A0 AD A0 C5
 93D8:D8 D0 AE AC AC D2 A0 AD
 93E0:A0 D2 C5 C3 C8 A0 CD C5
 93E8:CD CF C9 D2 C5 AC CD D3
 93F0:A0 AD A0 D2 C5 C3 C8 A0
 93F8:C4 C9 D3 D1 D5 C5 D4 D4
 9400:C5 AC AC CD A0 AD A0 CD
 9408:CF CE C9 D4 C5 D5 D2 AC
 9410:AC D1 A0 AD A0 D1 D5 C9
 9418:D4 D4 C5 D2 A0 A0 A0 A0
 9420:A0 A0 A0 A0 A0 A0 A0 D6
 9428:CF D4 D2 C5 A0 C3 C8 CF
 9430:C9 D8 3A A0 A0 A0 A0 A0
 9438:D0 AE A4 A0 A0 A0 A0 A0
 9440:A0 A0 D3 AE 24 AC CD BD
 9448:CD CF CE C9 D4 C5 D5 D2
 9450:A0 A0 A0 D3 D0 BD C3 CF
 9458:CE D4 C9 CE D5 C5 D2 A0
 9460:A0 A0 C5 D3 C3 BD CD C5
 9468:CE D5 A0 20 87 C1 CE CF
 9470:CD C1 CC C9 C5 A0 CC C5
 9478:C3 D4 D5 D2 C5 A0 C4 C9
 9480:D3 D1 D5 C5 D4 D4 C5 AC
 9488:AC D4 C1 D0 C5 DA A0 D5
 9490:CE C5 A0 D4 CF D5 C3 C8
 9498:C5 20 A9 17 4C 5B FB 00
 94A0:00

Apple & Minitel : les caractères semi-graphiques

L'écriture d'un logiciel destiné au Minitel passe presque invariablement par l'exploitation des caractères semi-graphiques ; nous vous proposons donc une table des codes de ces derniers.

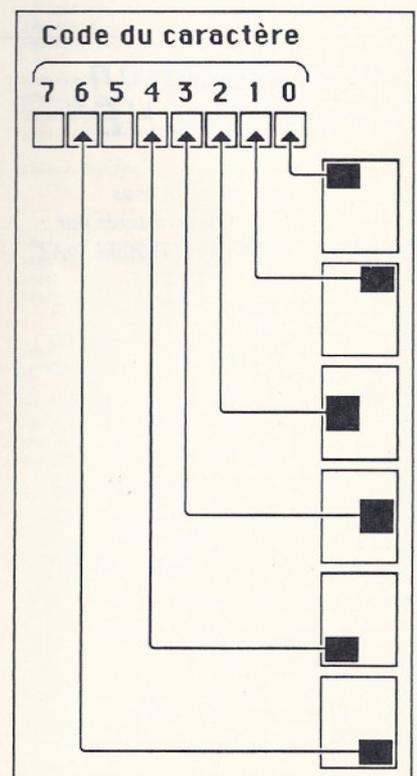
Chaque caractère semi-graphique vidéotex est contenu dans une matrice de 8 points (horizontalement) par 10 points (verticalement) et est divisé en 6 'pavés' élémentaires. L'état - 'on' ou 'off' - des pavés est déterminé par la valeur des bits 0, 1, 2, 3, 4 et 6 du code de caractère. Ainsi, comme le montre le schéma ci-joint, le bit 0 détermine l'état du bloc situé en haut et à gauche, le bit 1 détermine l'état du bloc situé en haut et à droite, etc.

Pour passer en mode semi-graphique, le Minitel attend le code \$0E (14). Le mode 'dis-joint', où les blocs élémentaires sont séparés par une ligne, est obtenu en envoyant les codes

\$1B et \$5A (27 et 90). \$1B et \$59 (27 et 89) retourne au mode 'joint'. Enfin, le code \$0F (15) permet de repasser en mode normal 'texte'.

\$20	\$21	\$22	\$23	\$24	\$25	\$26	\$27
\$28	\$29	\$2A	\$2B	\$2C	\$2D	\$2E	\$2F
\$30	\$31	\$32	\$33	\$34	\$35	\$36	\$37
\$38	\$39	\$3A	\$3B	\$3C	\$3D	\$3E	\$3F
\$60	\$61	\$62	\$63	\$64	\$65	\$66	\$67
\$68	\$69	\$6A	\$6B	\$6C	\$6D	\$6E	\$6F
\$70	\$71	\$72	\$73	\$74	\$75	\$76	\$77
\$78	\$79	\$7A	\$7B	\$7C	\$7D	\$7E	\$7F

Jeu de caractères Télétel joints



Conversions HGR ↔ DHGR

Alexandre Avrane

Qu'il est agréable de pouvoir disposer du graphisme double haute résolution (DHGR) disponible sur les Apple //c et //e équipés d'au moins 128K !

Encore faut-il pouvoir l'utiliser à sa guise : en effet, seuls quelques logiciels (par exemple Dazzle Draw ou Extasie) permettent de manipuler le DHGR.

Malheureusement, ce sont des programmes fermés et on peut être amené face à deux types de problèmes :

- vouloir imprimer l'image créée sur une configuration imprimante/interface qui n'a pas été prévue ;
- désirer utiliser la souplesse d'un langage de programmation (par exemple l'Applesoft) pour générer des séquences répétitives ;
- utiliser les fonctions puissantes des éditeurs HGR traditionnels (tels que MousePaint ou Blazing Paddles) sur une image DHGR.

Les images DHGR correspondent à une définition de 560 fois 192 points, alors qu'une image HGR standard en contient 280 fois 192. Elles sont stockées sur disque, par les éditeurs graphiques, comme des fichiers débutant à l'adresse \$2000

jusqu'en \$5FFF.

Néanmoins, elles ne peuvent être manipulées sans modification par votre routine préférée d'impression ou votre programme Applesoft car, après un BLOAD, la page HGR ne contient que les octets pairs de l'image pendant que la page HGR2 contient tous les octets impairs.

Pour vous en assurer essayez la séquence suivante :

```
HGR
BLOAD IMAGE, A$2000
CALL -151
C055 pour afficher la page
      HGR2
C054 pour retourner en page
      HGR
3D0G pour revenir au Basic
```

Le résultat affiché n'a généralement qu'une ressemblance fort vague avec l'image que vous avez si patiemment construite avec, par exemple, Dazzle Draw...

Il serait donc de bon ton de pouvoir disposer d'une routine qui convertisse le résultat pour afficher la partie gauche en page HGR et la partie droite en page HGR2. Ces deux parties pourraient alors être imprimées, manipulées à votre guise par un programme Applesoft, ou sauvegarder sur disque pour être modifiées par un éditeur HGR

traditionnel. Par la suite, il suffirait de recoller les deux parties pour obtenir à nouveau une image DHGR.

Coup de chance, le module DHGR effectue cette conversion ! C'est une routine (sans grande prétention) qui, chargée en \$8000, contient deux points d'entrée :

```
CALL 32768 ($8000)
convertit le fichier chargé en deux
images gauche/droite placées en
HGR et HGR2, c'est à dire
BSAVE IMAGE1, A$2000, L$2000
BSAVE IMAGE2, A$4000, L$2000
```

```
CALL 32771 ($8003)
effectue la conversion inverse
afin de sauver le résultat sur
disque, par
BSAVE IMAGE, A$2000, L$4000
et sa récupération ultérieure par
votre éditeur graphique.
```

DHGR est indépendant du système d'exploitation et pourra sans difficulté être utilisé sous DOS 3.3, ProDOS ou Pascal.

Le fichier T.DHGR contient le source en Big Mac au format texte ; le programme DHGR.DEMO illustre son utilisation en Applesoft.



Programme 'DHGR.DEMO'

```
100 REM DHGR.DEMO
110 :
120 :
130 REM Demo d'utilisation de
140 REM DHGR
150 :
160 :
200 D$ = CHR$ (4):F$ = "DHGR"
210 REM passe en graphique
220 HGR2 : HGR : PRINT CHR$ (27) CHR$ (
    17): REM Esc-Ctrl-Q
230 REM si on voulait charger une image
```

```
//e
//e+
//c
//gs
```

```
239 GOTO 250
240 PRINT D$"BLOAD"F$.PIC,A$2000"
250 REM charge la routine
260 PRINT D$"BLOAD"F$: REM A$8000
265 VTAB 20: PRINT "Image chargé
    e": GOSUB 1000
270 REM convertit en 2 images exploitabl
    es
280 CALL 32768
290 PRINT "Image convertie au format HGR
    ": GOSUB 1000
300 REM on va tracer un "X" sur l'image
310 HCOLOR= 3: REM blanc
315 REM surtout ne pas oublier d'initi
316 REM l'adresse 230 sinon gros bobo...
```

```

320 POKE 230,32: REM travaille sur HGR1
330 HPLOT 0,0 TO 279,95: HPLOT 279,96 TO
    0,191
340 POKE 230,64: REM travaille sur HGR2
350 HPLOT 0,96 TO 279,191: HPLOT 0,95 TO
    279,0
360 PRINT "Image terminée": GOSUB 1000
400 REM reconvertit au format DHGR
410 CALL 32771
412 PRINT "Image reconvertie au format D
    HGR": GOSUB 1000
415 END
420 REM et si on voulait sauver l'image
430 PRINT D$"BSAVE"F$.PIC,A$2000,L$4000
    "
440 REM mette L$4000 et non L$3FF8 pour
450 REM certains utilitaires DHGR
460 END
1000 REM Routine d'affichage
1010 PRINT "<1>/<2> affiche HGR1/HGR2":
    PRINT "<Return> continue ";
1020 GET K$: IF K$ = "1" THEN K = PEEK
    (49236): REM $C054
1030 IF K$ = "2" THEN K = PEEK (49237):
    REM $C055
1040 IF ASC (K$) = 13 THEN PRINT : RET
    URN
1050 GOTO 1020

```

Source 'T.DHGR'
 Assembleur Big Mac, format TEXT

DOS 3.3
ProDOS

```

1 *****
2 * CONVERSION DHGR/2 HGR *
3 *****
4
5 * Cette routine convertit des images DHGR
6 * stockées en un fichier binaire chargé en
7 * $2000 jusqu'en $5FFF, en deux images
8 * graphiques (parties gauche & droite)
9 * stockées en HGR et HGR2.
10 *
11 * Deux points d'entrée:
12 * CONVERT1 ($8000) pour conversion DHGR -> HGR+HGR2
13 * CONVERT2 ($8003) pour conversion HGR+HGR2 -> DHGR
14
15 * Modifié: 21/12/86
16 * Créé: 19/12/86
17 * (C) 1986 Alexandre Avrane
18
19 GBASL = $26
20 HPAG = $E6
21 HPOSN = $F411
22
23 CONVERT1 = * Entrée DHGR -> 2 HGR
24 CLC
25 BCC CONV^0 =jmp
26 CONVERT2 = * Entrée 2 HGR -> DHGR
27 SEC
28 CONV^0 PHP sauve carry
29 LDA #>$2000
30 STA HPAG page graphique
31 LDA #0
32 STA V1 1ère ligne
33

```

```

34 CONV^1 LDA V1
35 CMP #192 dernière ligne ?
36 BCS CONV^X oui!
37 LDX #0
38 LDY #0
39 JSR HPOSN calcule GBASL
40 PLP
41 PHP
42 BCS CONV^1A si Convert2
43 JSR CONV^2A
44 JSR CONV^2B
45 BMI CONV^1B =jmp
46 CONV^1A JSR CONV^2B
47 JSR CONV^2A
48 CONV^1B INC V1 ligne suivante
49 BNE CONV^1 =jmp
50 CONV^X PLP
51 RTS bye bye
52
53 *-----
54
55 CONV^2A LDY #40-1 40 colonnes par page HGR
56 LDX #80-1 80 colonnes par page DHGR
57 CONV^2 JSR SWAP HGR2
58 JSR CONV^2C alimente NEWLINE
59 DEX
60 JSR SWAP HGR1
61 JSR CONV^2C
62 DEX
63 DEY
64 BPL CONV^2
65 RTS
66
67 *-----
68
69 CONV^2B LDX #80-1
70 JSR SWAP HGR2
71 JSR CONV^3 vide NEWLINE vers HGR2
72 JSR SWAP HGR1
73 JSR CONV^3 vide NEWLINE vers HGR1
74 RTS
75
76 *-----
77
78 CONV^2C LDA (GBASL),Y
79 PHA
80 LDA NEWLINE,X
81 STA (GBASL),Y
82 PLA
83 STA NEWLINE,X échange NEWLINE et page HGR
84 RTS
85
86 *-----
87
88 CONV^3 LDY #40-1
89 CONV^4 JSR CONV^2C
90 DEX
91 DEY
92 BPL CONV^4
93 RTS
94
95 *-----
96
97 SWAP LDA GBASL+1
98 EOR #$60 échange vecteurs HGR1 & HGR2
99 STA GBASL+1
100 RTS
101
102 *-----
103
104 V1 DS 1
105 NEWLINE DS 80

```

Récapitulation
'DHGR'

Après avoir saisi ce code sous
 moniteur, vous le sauvegarderez
 par BSAVE DHGR,
 A\$8000,L\$C9

```

8000:18 90 01 38 08 A9 20 85
8008:E6 A9 00 8D 78 80 AD 78
8010:80 C9 C0 B0 1E A2 00 A0
8018:00 20 11 F4 28 08 B0 08
8020:20 35 80 20 4B 80 30 06
8028:20 4B 80 20 35 80 EE 78
8030:80 D0 DB 28 60 A0 27 A2
8038:4F 20 71 80 20 5A 80 CA
8040:20 71 80 20 5A 80 CA 88
8048:10 EF 60 A2 4F 20 71 80
8050:20 67 80 20 71 80 20 67
8058:80 60 B1 26 48 BD 79 80
8060:91 26 68 9D 79 80 60 A0
8068:27 20 5A 80 CA 88 10 F9
8070:60 A5 27 49 60 85 27 60
8078:A9 82 85 3D A9 D6 85 3E
8080:A9 83 85 3F A9 53 85 42
8088:A9 98 85 43 A0 00 20 2C
8090:FE A0 D6 84 3C A0 83 84
8098:3D A0 D5 84 3E A0 8B 84
80A0:3F A0 D8 84 43 A0 00 84
80A8:42 20 2C FE A9 1C 8D 1A
80B0:81 A9 81 8D 1B 81 20 11
80B8:81 C9 DD F0 06 9D 00 00
80C0:E8 D0 F3 A2 00 20 11 81
80C8:8D

```

De l'octet au nibble : Kyram

Gildas Ménier

Ce nouvel utilitaire n'est pas d'un usage évident en première approche ; garder le manuel du DOS 3.3 à portée de la main est une sage précaution. C'est dans un esprit d'utilisation pratique que le mode d'emploi qui suit est basé sur l'exemple.

Il vous permet de lire une disquette dont le formatage est fantaisiste, de savoir quel formatage a été utilisé, de protéger vos disquettes...

Pour lancer KYRAM, faire :

BRUN KYRAM

Un message du type :

L'AMPERSAND EST
OPERATIONNEL

doit s'afficher.

Présentation générale

Le DOS 3.3 ne permet pas d'utiliser à 100 % les disquettes, l'utilisateur n'ayant accès au support magnétique que dans les limites de ce que le DOS 3.3 autorise. Le catalogue DOIT être en piste \$11, par exemple. KYRAM se propose d'ajouter quelques commandes au Basic afin d'augmenter l'efficacité du SED (Système d'Exploitation de Disquettes), et de permettre à l'utilisateur d'ausculter en détail le contenu réel de la disquette.

Pour ajouter ces commandes, le programme exploite l'&. Quand l'interpréteur rencontre ce caractère, il fait un saut à l'adresse contenue en \$03F5, adresse de KYRAM, programme qui 'rendra la main' au Basic en fin de travail.

Chaque commande est un mot de cinq lettres précédé de &, et

éventuellement suivi d'une chaîne ou d'une variable alphanumérique (A\$ ou "TOTO" par exemple). Toutefois, si vous optez pour la solution variable alphanumérique, il est bon que l'ordre soit précédé d'une affectation de type A\$=A\$, ceci afin de préparer les routines Applesoft utilisées par le programme.

Note : il est utile d'essayer les commandes qui suivent au fur et à mesure de leur description.

Commandes générales

Ces commandes donnent des renseignements concernant le fonctionnement de KYRAM, les paramètres internes du DOS... Il s'agit de &QUID!, &DIAGN, &COMMS et &LRWTS.

&QUID!

Si vous tapez cette commande, puis CALL-151 et 320.400, divers nombres hexadécimaux apparaîtront, nombres qui représentent les prologues et épilogues des champs *adresse* et *données*.

&DIAGN (DIAGNostique)

Normalement, le message :

* KYRAM * PAS D'ERREUR A
LA DERNIERE COMMANDE

doit apparaître, vous indiquant que la dernière opération sur disquette a été menée à bien.

&COMMS (COMMandeS)

On demande par cet ordre la liste des commandes disponibles. Certaines commandes sont suivies de '_' qui rappelle que des paramètres doivent suivre. Si vous les omettez, vous obtiendrez :

ERREUR D'OPERANDE

&LRWTS (Liste RWTS)

L'écran est effacé et apparaissent divers mots et nombres hexadécimaux dont nous verrons la signification plus loin.

Accès au disque

La disquette est divisée en 35 pistes (en principe...), chacune étant divisée en 16 secteurs.

&SECRE (SECTor REad)

Cet ordre permet de lire un secteur. Pour lire le secteur \$00 de la piste \$01, il faut taper :

&SECRE "0001"

Le secteur est alors lu et écrit en mémoire, de l'adresse \$2000 à l'adresse \$20FF. Pour en être sûr, &DIAGN doit indiquer que tout s'est bien passé.

&SECRE "0F11"

commande la lecture du premier secteur du catalogue.

&BUFAS

Suite logique de la commande précédente, affiche sur l'écran le secteur qui vient d'être lu (un décalage s'affiche au début de chaque ligne).

&SECWR (SECTor WRite)

Pour écrire un secteur commençant en \$2000, il faut taper :

&SECWR "0001"

s'il devait être écrit sur le secteur 0 de la piste 1 (Changez de disquette avant de tenter l'opération).

&INITD (INIT Disquette)

Cette commande formatera la disquette SANS y mettre de

catalogue ni de DOS.

De l'octet au nibble

Avant écriture sur le disque, les octets composant le secteur sont d'abord codés en nibbles. Ils sont à la disquette ce que les octets sont à la mémoire. Un lecteur ne peut lire et écrire que des nibbles et comme il y a moins de 256 nibbles différents et valables, un secteur sur le disque occupe plus de 256 nibbles... De plus, comme les pistes sont circulaires (eh oui !), il faut bien que quelque chose indique au lecteur où commence le secteur qu'on veut lire.

Pour séparer ses secteurs, le DOS 3.3 utilise des blocs de nibbles (la plupart du temps des \$FF) un peu spéciaux, des nibbles synchronisés.

Pour en écrire un sur la disquette, le DOS utilise des routines dans lesquelles le temps d'exécution est primordial, et il ne sera pas écrit exactement comme les autres nibbles. Ainsi, les secteurs seront séparés.

Il faut maintenant que le lecteur trouve le secteur XX sur cette piste : le DOS 3.3 lit les premiers nibbles après les nibbles synchronisés et il recherche une série du type D5 AA 96 (ces nibbles sont réservés, c'est-à-dire qu'on ne peut pas trouver D5 dans un secteur codé en nibbles).

Cette série de trois nibbles constitue le *prologue* du champ adresse. Prologue parce qu'ils marquent le début d'une autre série qui indique, codés, le numéro de volume, le numéro de piste, le numéro de secteur et, enfin, un 'checksum' : tout ceci est le champ adresse. Le 'checksum' est une somme de contrôle qui permet à l'ordinateur d'être sûr que les différents numéros ont été bien lus. Si ce n'est pas le cas : I/O ERROR, après quelques autres tentatives bien entendues.

Huit nibbles après le prologue, on trouve DE AA EB, l'*épilogue*

du champ adresse. Plus loin, on trouve D5 AA AD qui est le prologue du champ données : ici débute le secteur codé, celui lu par &SECRE "...." (ou presque). DE AA EB ferme la marche (c'est l'épilogue du champ de données). Après, on retrouve une série de nibbles synchronisés.

&RDTRK (ReaD TRAcK)

Pour lire la piste 2 (par exemple) sans la décoder, taper :

```
&RDTRK "02"
```

puis :

```
CALL-151
```

puis :

```
2000.3F00 ('freiner' le listing par CTRL-S) et relire ce qui précède...
```

Pour que le DOS puisse lire un secteur, il faut que les valeurs des prologues, épilogues et autres 'checksum' correspondent parfaitement avec les valeurs D5 AA 96... sinon il n'y a pas de lecture possible. Certains programmes indiquent d'où vient l'erreur si erreur il y a (A : adresse, D : données). D'autres programmes essaient de recopier les secteurs même s'il manque un AA par exemple.

&MAXPI

La récupération de 4Ko supplémentaires sur la disquette (exemple classique) servira d'exemple. Taper :

```
&MAWPI "24"
```

puis :

```
&LRWTS
```

puis, avec une disquette vierge,

```
INIT HELLO.
```

Cette nouvelle disquette comporte \$24 pistes au lieu de \$23. Pour utiliser cette place disponible, il faut modifier la VTOC (table d'occupation du volume) ce qui indiquera au DOS les nouveaux secteurs accessibles (se reporter au manuel DOS pour l'organisation de la VTOC). Il faut pour ce faire, lire la VTOC :

```
&SECRE "0011"
```

puis :

```
CALL-151
```

puis :

```
20C0:FF FF 00 00 FF FF  
pour libérer la piste $24, puis :
```

```
2030:12 01 00 00 24
```

pour dire au SED qu'il peut compter sur une piste de plus. Bien sûr, il convient de revenir au Basic par Ctrl-C puis d'écrire cette VTOC là où on l'a trouvée :

```
&SECWR "0011"
```

Idée

Puisque c'est un octet dans la VTOC qui indique la position du catalogue, pourquoi ne pas transférer ce dernier sur cette piste \$24 ? Comme la plupart des copieurs ne copient pas cette piste, un double de cette disquette ainsi *customisée* ne serait guère utilisable...

La méthode

- formater une disquette \$24 pistes ;
- lire la VTOC ;
- modifier la position du catalogue (octet 1 et 2 de la VTOC, l'octet 0 étant inutilisé, voir annexe C du manuel DOS) ;
- réécrire la VTOC ;
- lire le premier secteur du catalogue en secteur 0F, piste \$11 et le réécrire en Secteur \$0F, Piste \$24 ;
- idem pour tous les autres secteurs de la piste (de \$0E à \$01) ;
- ne pas oublier de protéger dans la VTOC la piste \$24 qui, utilisée par le catalogue, n'est plus disponible ;
- libérer la piste \$11, ancien catalogue.

Malgré cette dernière modification, le DOS s'interdira d'écrire à la place de l'ancien catalogue. Pour l'y forcer, taper :

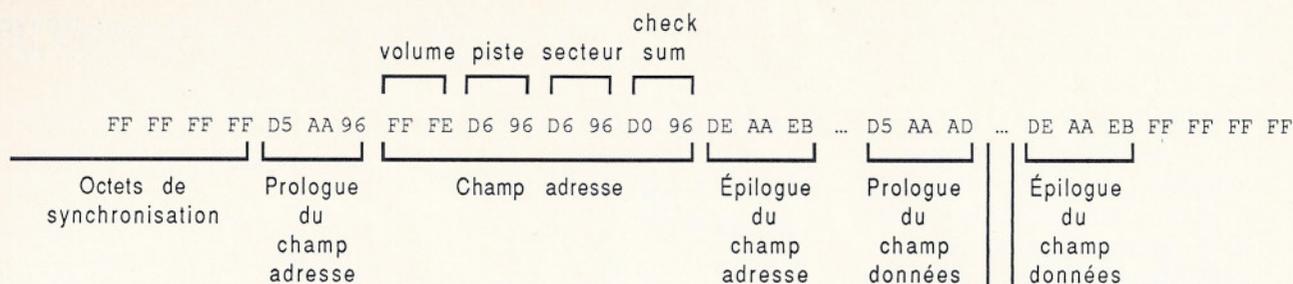
```
&ASAVU
```

avant de formater la disquette.

L'ordre inverse, pour éviter d'écrire sur la piste \$11 est :

```
&PSAVU.
```

(&LRWTS vous donne entre autre la valeur de SAU11 : \$A9 = le



Champ données (secteur codé en 342 nibbles pour 256 octets + checksum)

Le secteur sur la disquette

Les octets de synchronisation sont en fait constitués de 10 bits (1111111100). La méthode de synchronisation et le détail du codage du secteur (méthode 6/2) sont décrits en détail dans *Beneath Apple DOS* et *Beneath Apple ProDOS*, bibles de l'Applemaniaque...

DOS peut écrire piste \$11 ; \$69 = il ne peut pas)

Retaper cette liste d'ordres à chaque manipulation de disquette est fastidieux. Faites-en un programme Basic qu'il suffira de RUNer...

Idée (bis)

Vous avez 'deleté' un fichier et vous souhaitez le récupérer : opération simple s'il n'est pas déjà écrasé par un autre fichier. Trouver d'abord le secteur du catalogue où se trouve votre programme : taper

```
&SECRE "0F11" &BUFAS
&SECRE "0E11" &BUFAS
&SECRE "0D11" &BUFAS
```

jusqu'à apparition à l'écran du nom de votre programme, KYRAM par exemple.

Choisir l'offset le plus près du nom : \$00. Taper CALL - 151 puis :

```
2000.2020
```

réponse :

```
2000:00 00 FF 01 04 kk yy
rr aa mm A0 A0 A0...10
(kk yy rr aa mm représentent les
codes ASCII de KYRAM)
```

À la fin des A0 (espaces) vous tomberez par exemple sur 10.

FF indique que le fichier est effacé, 10 et 01 indiquent la piste et le secteur de début du fichier.

Remplacer FF par 10 et 10 par

A0 puis réécrire le secteur là où on l'a trouvé. Sauvegarder le fichier retrouvé sur une autre disquette ou bien le protéger dans la VTOC.

Pourquoi également ne pas modifier les messages d'erreur du DOS et, par exemple, remplacer l'entête du catalogue (Disk Volume...) par CTRL-D suivi de INIT HELLO...

Prologue, Épilogue, Kyram, et protection

En tapant &COMMS, on constate que certains ordres restent inconnus.

&WAPRO permet de modifier le prologue champ adresse dans les routines d'écriture, par exemple &WAPRO "D5 AA 96": le DOS lira la piste jusqu'à découvrir ce prologue qui lui indique qu'un champ adresse suit.

&RAPRO (comme Read Adress PROlogue) fait de même dans les routines de lecture.

Quand le DOS lit un secteur, il cherche la valeur de RAPRO et quand il écrit, celle de WAPRO.

Idem pour &RDPRO et &WDPRO qui se rapportent au champ données.

Idem pour &RAEPI, &RDEPI, &WAEPI et &DEPI qui concernent

l'épilogue.

Note : il ne faut préciser que deux nibbles pour &RAEPI et RDEPI, le DOS ne tenant pas compte du troisième.

Par exemple : &RAEPI "DEAA" (RAEPI comme Read Adress EPIlogue).

&SYNCR "FF", par exemple, permet de changer la valeur du nibble de synchronisation ce qui perturbera sensiblement la plupart des copieurs...

Un exemple

Créer une disquette pour étudier ces commandes, faire :

```
&WAPRO"D5BBCC" puis
INIT HELLO.
```

Maintenant, une tentative de CATALOG conduit à un I/O ERROR car le DOS cherchait comme prologue d'adresse D5 AA 96 et il ne pouvait trouver que D5 BB CC.

Comment lire maintenant ? En faisant &RAPRO "D5BBCC" pour qu'en lecture le DOS cherche le bon prologue.

Vous pouvez maintenant vous servir de cette disquette normalement, mais essayez de la copier...

Un problème subsiste : impossible de *booter* sur ce disque *new look*. En effet, la ROM C6 de l'Apple, sollicitée

lors du Boot cherche obstinément D5 AA 96. Avant de lire cette disquette, il faut donc charger Kyram et modifier les paramètres.

On peut de la même manière changer prologue et épilogue. Une autre idée serait de protéger un programme parmi d'autres : sur une disquette normale, changer le prologue données de façon à avoir D5 XX YY (écriture) puis sauver votre programme par BSAVE ou SAVE. Faire alors NEW, &WDPRO"D5 AA DD", &NOPRO, SAVE RIEN, DELETE RIEN, &TSTPR. &NOPRO demande au DOS d'ignorer XX et YY, SAVE et DELETE RIEN servent à réécrire le catalogue avec le prologue

normal. Un essai de LOAD ou BLOAD du programme conduit à un I/O ERROR. Pour le relire, un &NOPRO est de rigueur.

Face à une disquette inconnue

La disquette refuse d'être lue : Kyram devrait vous tirer d'affaire. Lire une piste avec &RDTRK, regarder les valeurs de prologue et d'épilogue. Neutraliser le contrôle de checksum par &NOCHK. Entrer les valeurs de prologue et d'épilogue trouvées à l'aide de &RDTRK. Essayer un catalogue, ça marche, tant mieux. Sinon, il faut trouver le catalogue et une boucle en

Basic utilisant &SECRE et &BUFAS rendra service pour trouver un secteur qui, de près ou de loin, ressemble à une VTOC...

En conclusion

Benjamin Franklin disait qu'il fallait pouvoir scier avec une lime et limer avec une scie. Kyram n'a peut-être pas d'applications en menuiserie, mais il semble que ses possibilités soient très étendues, l'imagination en est toutefois une limite. Les remarques et suggestions pourront être transmises à l'auteur par l'intermédiaire de la revue.



Source T.KYRAM Assembleur BigMac

```

1 *
2 *
3 *
4 *          KYRAM V2.1
5 *
6 *
7 *
8 *
9 * G.MENIER (c) 86          STG
10 *
11 * >PROGRAMME DE MODIFICATION DES PARAMETRES INTERNES DU DOS
12 * QUI PERMET, PAR EXEMPLE:
13 * -----
14 * L'UTILISATION DES MEMOIRES AUXILIAIRE (CARTE 80 COL)
15 * L'UTILISATION DES ROUTINES RWTS.
16 * LA MODIFICATION DES PARAMETRES INTERNES DU DOS
17 * LA CREATION DE DISKETTES FORMATEES A VOLONTE
18 * LA RELECTURE DE PROGRAMME SUR CES DISKETTES
19 * .....ETC
20 *
21 *
22 * -----
23 * PLAN PROGRAMME: - COPIE DU BUFFER BASIC (EN $200) DANS
24 * UN BUFFER EN $300
25 * - RECHERCHE DU MOT DANS UNE TABLE:LES
26 * MOTS FONT 5 LETTRES. UNE ERREUR CONDUIT
27 * A "**KYRAM* ERREUR DE SYNTAXE."
28 * - SINON BRANCHEMENT DIRECTEMENT AU MOT
29 * No.COMPT.
30 * - ON UTILISE POUR LE PASSAGE DE
31 * PARAMETRE, LES SP "DECODE.CONVERT.."
32 * -----
33 *
34 *          - TABLE DES SYMBOLES -
35 *
36 PASSE EQU $06          ;SERT COMME OCTET DE PASSAGE
37 SECONDE EQU $07        ;CONTIENT LA POSITION DE " No.2
38 BUFFIN EQU $08        ;POSITION DU BUFFER DE KYRAM
39 VALTYP EQU $11        ;CONTIENT LE TYPE DE LA VARIABLE
40 VOC EQU $18           ;SERT POUR LA RECHERCHE DU MOT
41 COMPT EQU $CE        ;CONTIENT LE No.DU MOT TROUVE
42 SAUV1 EQU $1E        ;DIVERS
43 SAUV2 EQU $1F        ; (VOUS AVEZ DIS VERT ?)
44 A1L EQU $3C          ;UTILISES PAR CERTAIN SP MONITEURS
45 A1H EQU $3D
46 A2L EQU $3E
47 A2H EQU $3F
48 A4L EQU $42
49 A4H EQU $43
50
51 STROUT EQU $DB3A      ;SORTIE ECRAN DU TXT POINTE /Y& A
52 CHRGET EQU $B1
53 PRBYTE EQU $FDDA     ;SORTIE ECRAN DE A EN HEXA
54 MYSEK EQU $BESA      ;ROUTINE DOS:POSITION DE LA TETE
55
56 MOVE EQU $C363        ;COPIE DE MEMOIRE PRINC-AUX
57 XFER EQU $C3B0        ;TRANSFERT D'EXECUTION PRINC-AUX
58
59
60
61 *-----
62
63          ORG $3FFA     ;ORIGINE
64
65 *-----
66
67
68 JSR DEBUT          ;PRESENTE+ VECTORISE &+RETOUR AUX
69 JMP $D43C          ;RETOUR AU BASIC (CMDLOOP)
70
71
72 KYRAM LDA $400      ;SAUVEGARDE DES 2 1ER CARACTERES
73 STA SAUV1          ;DE L'ECRAN
74 LDA $401           ;POUR AFFICHER 'KY'
75 STA SAUV2
76 LDA #'K'          ;ON AFFICHE 'KY'
77 STA $400
78 LDA #'Y'
79 STA $401
80
81 JSR GETLINE        ;COPIE DU BUFFER $200 EN BUFFIN
82 JSR LOOPPRIN      ;RECHERCHE DU MOT
83
84 LDA SAUV1          ;ON REMET LES 2 CARACTERES ECRAN
85 STA $400
86 LDA SAUV2          ;LA OU ON LES A TROUVE...
87 STA $401
88
89
90 TEST LDA #COMMANDE ;ON INITIALISE LE DEBUT DE LA
91 STA $1A            ;LISTE DES COMMANDES
92 LDA #/COMMANDE    ;POUR SAUTER EN INDIRECT
93 STA $1B            ;
94
95 LDA COMPT          ;No.DE COMMANDE
96 CMP #40            ;MAXI
97 BCS NON            ;PAS VALIDE:SN ERROR
98
99 ASL                ;A*2
100 CLC                ;
101 ADC COMPT          ;A*3
102 CLC                ;
103 ADC $1A            ;ON PREPARE LE SAUT / No.
104 BNE PASADJ        ;SI > 255 ON AJUSTE LE TIR
105 INC $1B           ;ADRESSE HAUTE
106 PASADJ STA $1A    ;
107
108
109 JMP ($1A)          ;ON SAUTE A LA COMMANDE
110
111 NON JMP ERROR     ;SN ERROR
112

```

Les commandes de Kyram

- passage de paramètres

L lecture

E écriture

les exemples sont les valeurs normales des paramètres

//e
//e+
//c
Igs

- &COMMS affiche la liste des commandes
- &LRWTS affiche les paramètres modifiables du DOS
- L &CHKSU provoque le contrôle du checksum en lecture (CHKSUM = 01)
- L &NOCHK rend inopérant ce test (CHKSUM = 00)
- L &TSTPR provoque le test des 2 derniers nibbles du prologue DATA lecture
- L &NOPRO ne teste que le premier nibble du prologue DATA lecture
- E - &WAPRO modifie le prologue adresse écriture : &WAPRO "D5AA96"
- E - &WDPRO modifie prologue DATA écriture : &WDPRO "D5AAAD"
- E - &WAEPI modifie épilogue adresse écriture : &WAEPI "DEAAEB"
- E - &WDEPI modifie épilogue DATA écriture : &WDEPI "DEAAEB"
- L - &RAPRO modifie le prologue adresse lecture : &RAPRO "D5AA96"
- L - &RDPRO modifie le prologue DATA lecture : &RDPRO "D5AAAD"
- L - &RAEPI modifie le prologue adresse lecture : &RAEPI "DEAA"
- L - &RDEPI modifie le prologue DATA lecture : &RDEPI "DEAA"
- E - &SYNCR modifie le nibble synchro formatage : &SYNCR "FF"
- E - &MAXPI nb maxi de pistes formatées : &MAXPI "23"
- E - &MAXSE nb maxi de secteurs formatés : &MAXSE "0F"
- LE- &VTSET position de VTOC : &VTSET "1100" ("ppss")
- E &ASAUV permet l'écriture sur la piste \$11 SAUII vaut A9
- E &PSAUV l'empêche SAUII vaut 69
- &QUID! en \$320 : prologues, épilogues, sync, VTOC
- L - &RDTRK lit la piste de \$2000 à \$3F00 : &RDTRK "01"
- &RESET Ctrl-Reset = Boot
- &NRESE Ctrl-Reset ≠ Boot
- L - &SECRE lit un secteur &SECRE "0001" ("sspp") résultat en \$2000
- E - &SECWR écrit un secteur &SECWR "0001" ("sspp") écriture de \$2000
- E &INITD formate le disque sans DOS ni VTOC ni catalogue
- &BUFAS affiche \$FF caractères en format TEXT à partir de \$2000 (avec offset)
- &DIAGN dernière erreur sur Kyram. Code d'erreur en \$300
- &KYRAM est branché
- &AMOVE transfert de xxxx à yyyy vers zzzz en mémoire auxiliaire :
&AMOVE "xxxxyyyyzzzz"
- &PMOVE transfert de xxxx à yyyy vers zzzz en mémoire principale :
&PMOVE "xxxxyyyyzzzz"
- &AEXEC transfert d'exécution vers mémoire auxiliaire
- &PEXEC transfert d'exécution vers mémoire principale
- &RETPR laisse en \$320 adresse de retour vers mémoire principale
- &RETAU laisse en \$320 adresse de retour vers mémoire auxiliaire

113	-----	123	JMP	MOTH	;WDPRO: ' DA PROLOG (')
114		124	JMP	MOTI	;WAEPI: ' AD EPILOG (')
115		125	JMP	MOTJ	;WDEPI: ' DA EPILOG (')
116	COMMANDE JMP MOTA	126	JMP	MOTK	;RAPRO: ' AD PROLOG (LECTU)
117	JMP MOTB	127	JMP	MOTL	;RDPRO: ' DA PROLOG (')
118	JMP MOTC	128	JMP	MOTM	;RAEPI: ' AD EPILOG (')
119	JMP MOTD	129	JMP	MOTN	;RDEPI: ' DA EPILOG (')
120	JMP MOTE	130	JMP	MOTO	;SYNCR:MODIFIE NIBBLE SYNCHRO
121	JMP MOTF	131	JMP	MOTP	;MAXPI:AU FORMATAGE:NB PISTES
122	JMP MOTG	132	JMP	MOTQ	;MAXSE: NB SECTEURS

```

133      JMP MOTR      ;VTSET:POSITION DE VTOC
134      JMP MOTS      ;ASAVU:PERMET ECRITURE TRK 11
135      JMP MOTT      ;PSAVU:L'EMPECHE
136      JMP MOTU      ;QUID!!LAISSE EN $320 HEAD+VTOC
137      JMP MOTV      ;RDTRK:LIRE EN $2000 LA PISTE N
138      JMP MOTW      ;RESET:VECTORISE VERS BOOT
139      JMP MOTX      ;NRESE:PAS DE BOOT SI RESET
140      JMP MOTY      ;SECRE:LIRE UN SECTEUR ("S.T.")
141      JMP MOTB1     ;SECWR:ECRIRE
142      JMP MOTZ      ;INITD:INIT SANS DOS SANS CATALOG
143      JMP MOTAL     ;BUFAS:AFFICHE $2000 EN FORMAT TXT
144      JMP MOTC1     ;DIAGN:AFFI. ERR/SECRE&SECWR&INITD
145      JMP KYRAM     ;KYRAM:VECTORISE L'AMPERSAND
146      JMP AMOV      ;SP DU Iie TRANSFERT DE MEV -> AUX
147      JMP PMOV      ;SP DU Iie TRANSFERT DE MEV -> PRI
148      JMP AEXE      ;SP DU Iie TRANSFERT EXEC -> AUX
149      JMP PEXE      ;SP DU Iie TRANSFERT EXEC -> PRIN
150      JMP RETP      ;LAISSE EN $320 BILLET DE RET PRIN
151      JMP RETA      ;LAISSE EN $320 BILLET DE RET AUXI
152
153 *-----
154 IOB    HEX 01600100
155 TRK    HEX 00      ;-> No DE PISTE
156 STR    HEX 00      ;-> No DE SECTEUR
157        HEX C240    ;-> TABLE CARACTERISTIQUES UNITEE
158        HEX 0020    ;BUFFER D'ENVOIS
159        HEX 0000
160 COM    HEX 00      ;COMMANDE (00,01,02,04)
161 ERR    HEX 00      ;CONTIENT ERR SI INDIC P A 1
162        HEX 006001
163 CARU   HEX 0001EFD8 ;TABLE DES CARACTERISTIQUES
164 *-----
165 MOTA   JSR $FC58    ;HOME
166        LDA #KY      ;AFFICHE TITRE *KYRAM*
167        LDY #/KY     ;
168        JSR STROUT  ;
169
170
171        LDA #CMD     ;AFFICHE L'ECRAN COMMANDES
172        LDY #/CMD    ;
173        JSR STROUT  ;
174
175
176        LDA #CMD1    ;
177        LDY #/CMD1   ;
178        JSR STROUT  ;
179
180        LDA #CMD2    ;
181        LDY #/CMD2   ;
182        JSR STROUT  ;
183
184        RTS          ;RETOUR PRG D'APPEL
185 MOTB   JSR $FC58    ;HOME
186        LDA #KY      ;AFFICHE *KYRAM*
187        LDY #/KY     ;
188        JSR STROUT  ;
189
190
191        LDA #ADREAD   ;AFFICHE "READ SYS..."
192        LDY #/ADREAD ;
193        JSR STROUT  ;
194
195
196        LDA $B955    ;ON VA CHERCHER READ ADD PROLOG
197        JSR PRBYTE   ;ON L'AFFICHE
198        LDA $B95F    ;
199        JSR PRBYTE   ;
200        LDA $B96A    ;
201        JSR PRBYTE   ;
202        JSR FINC     ;ON AFFICHE "FIN DE CHAMP:"
203        LDA $B991    ;ET ON AFFICHE
204        JSR PRBYTE   ;LE READ ADD EPILOG
205        LDA $B99B    ;
206        JSR PRBYTE   ;
207        JSR CHD      ;ON AFFICHE "CHAMP DONNEES:"
208        LDA $B8E7    ;ET ON L'AFFICHE
209        JSR PRBYTE   ;
210        LDA $B8F1    ;
211        JSR PRBYTE   ;
212        LDA $B8FC    ;
213        JSR PRBYTE   ;
214
215        JSR FINC     ;MEME CHOSE POUR "FIN DE CHAMP:"
216        LDA $B935    ;
217        JSR PRBYTE   ;
218        LDA $B93F    ;
219        JSR PRBYTE   ;
220
221        LDA #ADWRITE ;ON REFAIT TOUT CA AVEC
222        LDY #/ADWRITE ;"WRITE SYSTEM:..."
223        JSR STROUT  ;
224
225
226        LDA $BC7A    ;
227        JSR PRBYTE   ;
228        LDA $BC7F
229
230      JSR PRBYTE
231      LDA $BC84
232      JSR PRBYTE
233      JSR FINC
234      LDA $BCAE
235      JSR PRBYTE
236      LDA $BCB3
237      JSR PRBYTE
238      LDA $BCB8
239      JSR PRBYTE
240
241      JSR CHD
242      LDA $B853
243      JSR PRBYTE
244      LDA $B858
245      JSR PRBYTE
246      LDA $B85D
247      JSR PRBYTE
248
249      JSR FINC
250      LDA $B89E
251      JSR PRBYTE
252      LDA $B8A3
253      JSR PRBYTE
254      LDA $B8A8
255      JSR PRBYTE
256
257      LDA #CHKSUM   ;"CHECKSUM:"
258      LDY #/CHKSUM
259      JSR STROUT
260      LDX $B92E
261      LDA #01
262      CPX #$13      ;SI L'OCT CHERCHE A 13 (BNE L+13)
263      BEQ OUI       ;ON ECRIT "OUI"
264      LDA #00       ;SINON "NON"
265      JSR PRBYTE   ;AFFICHE A EN HEXAA
266
267
268      LDA #ADRTST   ;IDEM AVEC TEST DE L'EPILOG DATA
269      LDY #/ADRTST
270      JSR STROUT
271
272      LDX $B8F3
273      LDA #01
274      CPX #$F2
275      BEQ OUAIS
276      LDA #00
277      JSR PRBYTE   ;PRT A
278
279      LDA #SAUVE    ;ON AFFICHE OCTET SAUVEGARDE DE LA
280      LDY #/SAUVE   ;PISTE 11
281      JSR STROUT    ;69=>PROTEGE A9=>NON PROTEGE
282      LDA $B292
283      JSR PRBYTE
284
285      LDA #SYNC     ;ON AFFICHE LE NIBBLE DE SYNCHRO
286      LDY #/SYNC
287      JSR STROUT
288      LDA $BC60
289      JSR PRBYTE
290
291      LDA #NBPISTES ;ON AFFICHE LE NB MAXI DE PISTES
292      LDY #/NBPISTES
293      JSR STROUT
294      LDA $BEFE
295      JSR PRBYTE
296      LDA #NBSECT   ;ON ECRIT LE NB MAXI DE SECTEURS
297      LDY #/NBSECT ;EN CE QUI CONCERNE FORMAT.
298      JSR STROUT
299      LDA $BF2A
300      JSR PRBYTE
301
302      LDA #VTOC     ;VOILA LA POSITION DE VTOC
303      LDY #/VTOC
304      JSR STROUT
305      LDA $AC01
306      JSR PRBYTE
307      LDA #SECTEUR
308      LDY #/SECTEUR
309      JSR STROUT
310      LDA $B00D
311      JSR PRBYTE
312      LDA #TRKC     ;ON AFFICHE L'ADD DU + RECENT PRG
313      LDY #/TRKC
314      JSR STROUT
315      LDA $AA73
316      JSR PRBYTE
317      LDA $AA72
318      JSR PRBYTE
319      LDA #LD
320      LDY #/LD
321      JSR STROUT
322      LDA $AA61
323      JSR PRBYTE
324      LDA $AA60

```

```

325 JSR PRBYTE
326 LDA #CTRK ; TRACK COURANT POUR KYRAM
327 LDY #/CTRK
328 JSR STROUT
329 LDA TRK
330 JSR PRBYTE
331 LDA #DTRK ; ...ET AUSI POUR LE DOS.
332 LDY #/DTRK
333 JSR STROUT
334 LDA $3E7 ; PAS MARRANT A TROUVER:3E7-3E8
335 STA $06 ; CONTIENT UNE ADRESSE QUI
336 LDA $3E8 ; MEME L'ADRESSE DU DEBUT DE IOB ?
337 STA $07 ;
338 LDY # $00 ;
339 LDA ($06),Y
340 STA $1A
341 INY
342 LDA ($06),Y
343 STA $1B
344 LDY # $04 ; TRK C'EST LE 4IEME DE IOB
345 LDA ($1A),Y
346
347 JSR PRBYTE
348 LDA #DOTTED ; ON TRACE UNE LIGNE.
349 LDY #/DOTTED
350 JSR STROUT
351
352 RTS
353 MOTC LDA # $13 ; OBLIGE LE DOS A TESTER CHEKSUM
354 STA $B92E
355 RTS
356 MOTD LDA # $00 ; PERMET LES ERREURS SUR CHKSUM
357 STA $B92E
358 RTS
359 MOTE LDA # $F2 ; PAS D'ERR SUR LE PROLOG DATA
360 STA $B8F3
361 LDA # $E7
362 STA $B8FE
363 RTS
364 MOTF LDA # $00 ; LES ERR PASSENT A TRAVERS
365 STA $B8F3
366 STA $B8FE
367 RTS
368 MOTG LDA # $0C ; ON VEUT 3 OCTETS EN ENTREE
369 STA SECONDE ; 5(1ER")+6(3*2)+1=$0C
370 JSR SAUVEG ; SAUVE $B8 $B9 EN VUE D'UN RET BAS
371 JSR DECODE ; INITIALISE:"DEAA95"? OU A$?
372 JSR CONVERT ; LE 1ER TROUVE EST PLACE DS A
373 STA $BC7A ; ON STOCKE
374 JSR CONVERT ; LE 2ND EST PLACE DANS A
375 STA $BC7F ; ETC
376 JSR CONVERT
377 STA $BC84
378 JSR RESTORE
379 RTS
380 MOTH LDA # $0C ; CF MOTG
381 STA SECONDE
382 JSR SAUVEG
383 JSR DECODE
384 JSR CONVERT
385 STA $B853
386 JSR CONVERT
387 STA $B858
388 JSR CONVERT
389 STA $B85D
390 JSR RESTORE
391 RTS
392 MOTI LDA # $0C ; CF MOTG
393 STA SECONDE
394 JSR SAUVEG
395 JSR DECODE
396 JSR CONVERT
397 STA $BCAE
398 JSR CONVERT
399 STA $BCB3
400 JSR CONVERT
401 STA $BCB8
402 JSR RESTORE
403 RTS
404 MOTJ LDA # $0C ; CF MOTG
405 STA SECONDE
406 JSR SAUVEG
407 JSR DECODE
408 JSR CONVERT
409 STA $B89E
410 JSR CONVERT
411 STA $B8A3
412 JSR CONVERT
413 STA $B8A8
414 JSR RESTORE
415 RTS
416 MOTK LDA # $0C ; RAPRO
417 STA SECONDE
418 JSR SAUVEG
419 JSR DECODE
420 JSR CONVERT
421 STA $B955
422 JSR CONVERT
423 STA $B95F
424 JSR CONVERT
425 STA $B96A
426 JSR RESTORE
427 RTS
428 MOTL
429 LDA # $0C ; RDPRO
430 STA SECONDE
431 JSR SAUVEG
432 JSR DECODE
433 JSR CONVERT
434 STA $B8E7
435 JSR CONVERT
436 STA $B8F1
437 JSR CONVERT
438 STA $B8FC
439 JSR RESTORE
440 RTS
441 MOTM
442 LDA # $0A ; RAEPI
443 STA SECONDE
444 JSR SAUVEG
445 JSR DECODE
446 JSR CONVERT
447 STA $B991
448 JSR CONVERT
449 STA $B99B
450 JSR RESTORE
451 RTS
452 MOTN
453 LDA # $0A ; RDEPI
454 STA SECONDE
455 JSR SAUVEG
456 JSR DECODE
457 JSR CONVERT
458 STA $B935
459 JSR CONVERT
460 STA $B93F
461 JSR RESTORE
462 RTS
463 MOTO
464 LDA # $08 ; IDEM AVEC 1 OCTET POUR SYNCR
465 STA SECONDE ; SYNCR
466 JSR SAUVEG
467 JSR DECODE
468 JSR CONVERT
469 STA $BC60
470 STA $B83E
471 JSR RESTORE
472 RTS
473 MOTP
474 LDA # $08 ; MAXPI
475 STA SECONDE
476 JSR SAUVEG
477 JSR DECODE
478 JSR CONVERT
479 STA $BEFE
480 JSR RESTORE
481 RTS
482 MOTQ
483 LDA # $08 ; MAXSE
484 STA SECONDE
485 JSR SAUVEG
486 JSR DECODE
487 JSR CONVERT
488 STA $BF2A
489 JSR RESTORE
490 RTS
491 MOTR
492 LDA # $0A ; VTOCSET
493 STA SECONDE
494 JSR SAUVEG
495 JSR DECODE
496 JSR CONVERT
497 STA $AC01 ; PISTE
498 JSR CONVERT
499 STA $B00D ; SECTEUR
500 JSR RESTORE
501 RTS
502 MOTS
503 LDA # $A9 ; ON MODIFIE LE DOS (PISTE 11)
504 STA $B292 ; ANTI
505 RTS ; TESTDOS
506 MOTT
507 LDA # $69 ; RETABL
508 STA $B292 ; TEST
509 RTS
510 MOTU
511
512 LDA $BC7A ; QUID! TRANSFERT D'OCTETS -> $320
513 STA $320 ; ON TROUVE LES PROLOGUES EPILOGUES
514 LDA $BC7F ; DE READ ET WRITE + SYNCR ET VTOC
515 STA $321 ; PISTE EN TETE
516 LDA $BC84

```

```

517 STA $322
518 LDA $BCAE
519 STA $323
520 LDA $BCB3
521 STA $324
522 LDA $BCB8
523 STA $325
524
525 LDA $B853
526 STA $326
527 LDA $B858
528 STA $327
529 LDA $B85D
530 STA $328
531 LDA $B89E
532 STA $329
533 LDA $B8A3
534 STA $32A
535 LDA $B8A8
536 STA $32B
537
538
539
540
541 LDA $B955
542 STA $32C
543 LDA $B95F
544 STA $32D
545 LDA $B96A
546 STA $32E
547 LDA $B991
548 STA $32F
549 LDA $B99B
550 STA $330
551
552 LDA $B8E7
553 STA $331
554 LDA $B8F1
555 STA $332
556 LDA $B8FC
557 STA $333
558 LDA $B935
559 STA $334
560 LDA $B93F
561 STA $335
562
563 LDA $B83E
564 STA $336 ;SYNC
565
566 LDA $AC01 ;VTPI
567 STA $337
568 LDA $B00D
569 STA $338 ;VTSEC
570 RTS
571
572
573 MOTV
574 LDA #$08 ;READ TRK : 1 PARAMETRE
575 STA SECONDE
576 JSR SAUVEG
577
578 JSR DECODE
579
580 LDA #$00
581 STA COM ;ON MET LE DISK EN MARCHE
582 JSR PIST
583
584 LDA TRK ;1) L'ENVOYER EN PISTE N
585 JSR MYSEEK ;C'EST FAIT.
586
587 LDA #$00 ;2) MAINTENANT ON VA LIRE NIBBLES
588 STA IM+1 ;ET LES STOCKER DE $2000 A $3FFF.
589 LDA #$20
590 STA IM+2
591 LDY #$00 ;IM+1 CONTIENT ADRESSE STOCKAGE
592
593 LDX #$60 ;NB SLOT*16
594 LDA $C08E,X ;STROBE LECTURE
595 TRKR LDA $C08C,X ;LIS NIIBLE
596 BPL TRKR ;TANT QU'IL N'EST PAS BON...
597 IM STA $2000 ;STOCKE
598 INC IM+1 ;ON BALADE LE POINTEUR
599 BNE TRKR
600 INC IM+2
601 LDA IM+2
602 CMP #$3F ;ON STOPPE EN $3F00
603 BNE TRKR
604
605 LDA $C088,X ;ON ARRETTE TOUT ! DISK:STOP!
606 JSR RESTORE ;ON RECUP. TXTPTR
607
608 RTS ;FINI.
609 MOTW
610 LDA #$00 ;ON VECTORISE RESET EN RAM
611 STA $3F4
612 LDA #$FF ;ET DS LE DOS SUR BOOT
613 STA $9E37
614 RTS
615
616 MOTX
617 LDA #$38 ;NORESET
618 STA $3F4
619 LDA $SA5
620 STA $9E37 ;PAS DE BOOT SI RESET
621 RTS
622
623 MOTY
624 LDA #$01 ;COMMANDE POUR READ
625 STA COM
626 JSR RWTS ;ON UTILISE RWTS
627 JSR RESTORE
628 RTS
629 RWTS LDA #$0A ;2 PARAMETRES
630 STA SECONDE ;MAIS *PISTE EN TETE*
631 JSR SAUVEG
632 JSR DECODE
633 JSR CONVERT
634
635 STA STR ; LE 1ER C'EST SECTEUR
636 PIST JSR CONVERT
637 STA TRK ; LE 2ND TRACK
638
639 LDA #$00
640 STA ERR ; COMME CA SI ERR<>00 Y A ERREUR
641
642
643 GO LDA #/IOB
644 LDY #IOB ;RWTS
645 JSR $3D9 ;ON SAUTE AUX VRAIS RWTS.
646 RTS
647
648 MOTZ LDA #$04
649 STA COM ;FORMATAGE SANS DOS SANS CATALOG
650 JMP GO
651
652 MOTA1 NOP ;BUFAS:BUFFER $2000 DE IOB EN ASCII
653 LDA #$00
654 STA $1A
655 LDA #$20
656 STA $1B ;ADRESSE DE DEPART
657 LDA #$00 ;PASSE
658 STA PASSE
659
660 JP LDA PASSE
661 JSR PRBYTE
662 LDA #"- "
663 JSR $FDF0 ;AFFICHE A
664 LDY #00
665 LOOP1 LDA ($1A),Y
666 JSR $FDF0
667 INY
668 CPY #32 ;ON FAIT 32 CARACTERES
669 BNE LOOP1
670 JSR $FC62 ; CR
671 LDA #32
672 CLC
673 ADC $1A
674 STA $1A
675 STA PASSE
676 CMP #$00 ;POUR 255 CARACTERES
677 BNE JP ; LIGNE SUIVANTE
678
679 JSR $FC62 ;CR
680
681 RTS
682
683
684 MOTB1
685 LDA #$02 ;ON ECRIS LE BUFFER
686 STA COM
687 JSR RWTS
688 JSR RESTORE ;WRITE
689 RTS
690
691 MOTC1 LDA ERR ;ON REGARDE SI ERR<>0
692 STA $300 ;AU PASSAGE ON LA LAISSE EN $300
693 CMP #$00
694 BNE ERREUR ;SI OUI ON L'AFFICHE
695 LDA #NONP ;SI NON...
696 LDY #/NONP
697 JSR STROUT
698 RTS
699 ERREUR CMP #$10
700 BNE VOLUME
701 LDA #WP
702 LDY #/WP
703 JSR STROUT
704 RTS
705 VOLUME CMP #$20
706 BNE BIZA
707 LDA #VOL
708 LDY #/VOL

```

```

709      JSR  STROUT
710 BIZA  CMP   #\$40
711      BNE  RTWA
712      LDA  #BIZ
713      LDY  #/BIZ
714      JSR  STROUT
715      RTS
716 RTWA  CMP   #\$80
717      BNE  CAALORS
718      LDA  #RTW
719      LDY  #/RTW
720      JSR  STROUT
721      RTS
722 CAALORS LDA #CA
723      LDY  #/CA
724      JSR  STROUT
725      RTS
726
727 KYRAM  LDA  #KYRAMP ;ON AFFICHE "KYRAM OK"
728      STA  \$3F6 ;ET ON VECTORISE VERS LE DEBUT
729      LDA  #/KYRAMP
730      STA  \$3F7
731
732      LDA  #KYM
733      LDY  #/KYM
734      JSR  STROUT
735      RTS
736
737 AMOV   STA  \$C007 ;ON VA UTILISER \$C100 MEM
738      LDA  #\$12 ;6 PARAMETRES
739      STA  SECONDE
740      JSR  SAUVEG
741      JSR  DECODE
742
743      JSR  CONVERT
744      STA  A1H ;PTFORT
745      JSR  CONVERT
746      STA  A1L ;PTFAIBLE :ADD
747
748      JSR  CONVERT
749      STA  A2H ;FIN (PTFORT)
750      JSR  CONVERT
751      STA  A2L ;(PTFAIBLE)
752
753      JSR  CONVERT
754      STA  A4H
755      JSR  CONVERT
756      STA  A4L
757
758      SEC  ;MEV PRINCIPALE VERS AUXI
759
760      JSR  MOVE
761
762      JSR  RESTORE
763      RTS
764 PMOV   STA  \$C007 ;MEME TRAFFIC QUE PRECED.
765      LDA  #\$12
766      STA  SECONDE
767      JSR  SAUVEG
768      JSR  DECODE
769
770      JSR  CONVERT
771      STA  A1H
772      JSR  CONVERT
773      STA  A1L
774
775      JSR  CONVERT
776      STA  A2H
777      JSR  CONVERT
778      STA  A2L
779
780      JSR  CONVERT
781      STA  A4H
782      JSR  CONVERT
783      STA  A4L
784
785      CLC  ;VERS MEMOIRE PRINCIPALE
786      JSR  MOVE
787
788      JSR  RESTORE
789      RTS
790
791 AEXE   STA  \$C007 ;MEM \$C100
792      LDA  \$3F2 ;RECUPERE L'ENTREE A CHAUD DU DOS
793      STA  \$3ED
794      LDA  \$3F3
795      STA  \$3EE
796      LDA  #\$FE
797      ADC  #\$FF ;ON MET L'INDIC V A 1
798      SEC
799      JSR  XFER ;C'EST PARTI...
800      RTS
801 PEXE   STA  \$C007
802      LDA  \$3F2
803      STA  \$3ED
804      LDA  \$3F3

805      STA  \$3EE
806      CLV
807      CLC
808      JSR  XFER
809      RTS
810
811 RETP   LDX  #\$00 ;TRANSFERT EN \$320 PEXE
812 TRANS1 LDA PEXE,X
813      STA  \$320,X
814      INX
815      CPX  #\$20
816      BNE  TRANS1
817      RTS
818
819 RETA   LDX  #\$00 ;TRANSFERT EN \$320 AEXE
820 TRANS2 LDA AEXE,X
821      STA  \$320,X
822      INX
823      CPX  #\$20
824      BNE  TRANS2
825      RTS
826
827 *-----
828
829 * CES ROUTINES PERMETTENT D'ALLER CHERCHER DANS A
830 * DES VALEURS DS LE BUFFER (EX "D5AA") OU EN MEV
831 * BASIC (EX AS).
832
833 SAUVEG LDA \$B8 ;ON SAUVEGARDE LES POINTEUR
834      STA  \$1A ;DU BASIC (TXTPTR)
835      LDA  \$B9
836      STA  \$1B
837      RTS
838 RESTORE LDA \$1A ;ON LES REMET OU ILS ETAIENT
839      STA  \$B8
840      LDA  \$1B
841      STA  \$B9
842      RTS
843
844 DECODE LDA BUFFIN ;POSITION DES POINTEURS SUR OCTETS
845      CLC ;QU'IL FAUT (PARAMETRES)
846      ADC  #\$05 ;ON SE POSITIONNE APRES LE MOT
847      LDY  BUFFIN+1
848      STA  \$B8
849      STY  \$B9
850
851      LDY  #\$05
852      LDA  (BUFFIN),Y ;ON A UNE 1ER " ?
853      CMP  #\$22 ;
854      BNE  VARIABLE ;NON, CA DOIT ETRE DU X\$
855      LDY  SECONDE ;N PARAMETRES PLUS LOIN...
856      LDA  (BUFFIN),Y
857      CMP  #\$22 ;ON A UNE 2ND " ?
858      BNE  OPERROR ;NON:ERREUR OPERANDE
859      INC  \$B8 ;C'EST BON, ON SE PREPARE A DECODER
860      RTS
861
862 OPERROR JSR \$FBE4 ;BEEP
863      LDA  #OPERRORM ;*KYRAM* ERREUR OPERANDE
864      LDY  #/OPERRORM
865      JSR  STROUT
866      JMP  \$D43C ;BASIC
867
868 VARIABLE JSR CHRGET ;CHERCHE VARIABLE POINTEE PAR (\$B8)
869
870      LDY  #\$01
871      LDA  (\$B3),Y ;ON PREND SON ADRESSE...
872      STA  \$B8
873      INY
874      LDA  (\$B3),Y
875      STA  \$B9
876
877      LDA  VALTYP ;SI JAMAIS C'EST PAS UNE CHAINE...
878      CMP  #\$FF ;
879      BNE  OPERROR ;...ON LAISSE TOMBER.
880
881      RTS
882 CODE   CMP  #\$30 ;A EST UN CHIFFRE ?
883      BCC  OPERROR ;NI CHIFFRE NI LETTRE:DEHORS!
884      CMP  #\$3A ;
885      BCS  HOP ;C'EST UNE LETTRE
886      CLC ;C'EST UN CHIFFRE ON CODE HEXA
887      SBC  #\$2F
888      RTS
889 HOP    CMP  #\$41 ;ENTRE A ET F?
890      BCC  OPERROR ;EUH, NON...
891      CMP  #\$47 ;
892      BCS  OPERROR ;NON PLUS...
893      CLC
894      SBC  #\$4 ;C'EST BON ON CODE HEXA
895      RTS
896
897 CONVERT LDY #\$00 ;ON PREND LE 1ER
898      LDA  (\$B8),Y
899      JSR  CODE ;ON LE CODE
900      ASL ;ON LE MULTIPLIE PAR 16

```

```

901 ASL
902 ASL
903 ASL
904 STA PASSE
905 INY
906 LDA (SB8),Y
907 JSR CODE
908 CLC
909 ADC PASSE ;ON LUI AJOUTE LE SECOND
910 TAX
911 LDA SB8
912 CLC
913 ADC #02 ;ON AJUSTE LES POINTEURS POUR LA
914 BCC JUMP ;SUITE.
915 INC SB9
916 JUMP STA SB8
917 TXA
918 RTS ;A CONTIENT LES 2 ASCII EN 1 HEXA
919
920 *-----
921
922 GETLINE LDA #000 ;ADRESSE BUFFER DE &INTERPRETEUR
923 STA BUFFIN ;
924 LDA #03 ;
925 STA BUFFIN+1 ;
926
927 LDX #00
928
929 JMP SAUT
930
931 CHARGE JSR CHRGET ;POINTE CARACT SUIVANT DS BUF BAS
932 INX ;ON UTILISE X COMME INDEX
933 SAUT LDY #00
934 LDA (SB8),Y ;GET CAR
935 PHA ;X->Y
936 TXA ;
937 TAY ;
938 PLA ;
939 STA (BUFFIN),Y ;TRANSFERT D'UN CARACTERE
940 CMP #3A ;SI "A" ALORS ON A FINI
941 BEQ STOP ;C'EST FAIT
942 CMP #00 ;SI C'EST FIN DE LA LIGNE PAREIL
943 BNE CHARGE ;ON CONTINUE SINON
944
945 STOP RTS ;FIN
946
947 LOOPPRIN LDA #00 ;ON COMMENCE PAR LE MOT No.0
948 STA COMPT
949
950 LDA #LISTEVOC ;OU EST LA LISTE DES MOTS?
951 STA VOC
952 LDA #/LISTEVOC
953 STA VOC+1
954
955 NOUVEAU LDY #00 ;ON COMPARE MOT A MOT
956 * ;SI ON UNE<> PASSE AU MOT SUIVANT
957 LOOPA LDA (BUFFIN),Y
958 CMP (VOC),Y
959 BNE SUIVANT
960 INY
961 CPY #05
962 BNE LOOPA
963
964 RTS
965
966 SUIVANT LDA VOC
967 CLC
968 ADC #05
969 STA VOC
970 LDA VOC+1
971 ADC #00
972 STA VOC+1
973
974 LDX COMPT
975 INX
976 STX COMPT
977 CPX #40 ;MAXI
978 BNE NOUVEAU ;ON TESTE UN NOUVEAU MOT.
979
980 RTS
981 ERROR JSR $FBE4 ;BIP
982 LDA #SNERROR ;SYNTAX ERROR:BOUM!
983 LDY #/SNERROR
984 JSR STROUT
985
986 JMP $D43C ;APPSOFT
987 *-----
988
989 * ICI ON A LA LISTE DES MOTS ADMIS
990
991 LISTEVOC ASC 'COMMSLRWTSCHKSU'
992 ASC 'NOCHKTSPRNOPRO'
993 ASC 'WAPROWDPROWAEPI'
994 ASC 'WDEPIRAPRODPRO'
995 ASC 'RAEPIRDEPISYNCR'
996 ASC 'MAXPIMAXSEVTSET'
997 ASC 'ASAUVPASUVAUQUID!'
998 ASC 'RDTRKRESETRNRESE'
999 ASC 'SECRESECWRINITD'
1000 ASC 'BUFASDIAGNKYRAM'
1001 ASC 'AMOVEPMOVEAEEXEC'
1002 ASC 'PEXECRETPRRETAU'
1003
1004 *-----
1005
1006 * DIVERSES ROUTINES OU DONNEES D'AFFICHAGE
1007
1008 FINC LDA #ADREAD1 ;AFFICHE "FIN DE CHAMP"
1009 LDY #/ADREAD1
1010 JSR STROUT
1011
1012
1013 RTS
1014
1015 CHD LDA #ADREAD2 ;AFFICHE "CHAMP DONNEES:"
1016 LDY #/ADREAD2
1017 JSR STROUT
1018
1019 RTS
1020
1021 OPERRORM HEX 8D ;TEXTES D'ERREURS...
1022 ASC "**KYRAM* ERREUR D'OPERANDE ."
1023 HEX 00
1024 SNERROR HEX 8D
1025 ASC "**KYRAM* ERREUR DE SYNTAXE ."
1026 HEX 00
1027
1028 KY HEX 8D
1029 ASC "STG *****"
1030 HEX 8D
1031 ASC " * KYRAM *"
1032 HEX 8D
1033 ASC " *****"
1034 HEX 8D8D00
1035 CMD ASC "-----COMMANDES-----"
1036 HEX 8D
1037 ASC "/ &COMMS / &LRWTS / &QUID! / &BUFAS "
1038 HEX 8D
1039 ASC "/ &CHKSU / &NOCHK / &TSTPR / &NOPRO "
1040 HEX 8D00
1041 CMD1
1042 ASC "/ &WAPRO_ / &WDPRO_ / &WAEPI_ / &WDEPI_"
1043
1044 HEX 8D
1045 ASC "/ &RAPRO_ / &RDPRO_ / &RAEPI_ / &RDEPI_"
1046
1047 HEX 8D
1048 ASC "/ &SYNCR_ / &MAXPI_ / &MAXSE_ / &VTSET_"
1049 HEX 8D
1050 ASC "/ &ASAUV_ / &PSAUV_ / &RESET_ / &NRESE_"
1051 HEX 8D00
1052 CMD2 ASC "/ &RDTRK_ / &SECRE_ / &SECWR_ / &INITD_"
1053 HEX 8D
1054 ASC "/ &AMOVE_ / &PMOVE_ / &AEEXEC_ / &PEXEC_"
1055 HEX 8D
1056 ASC "/ &RETPR_ / &RETAU_ / &DIAGN_ / &KYRAM_"
1057 HEX 8D
1058 ASC "-----"
1059 HEX 8D00
1060 ADREAD ASC "-----RWTS-PARAMETRES-----"
1061 HEX 8D
1062 ASC "L READ SYSTEM: CHAMP ADRESSE ð"
1063 HEX 00
1064 ADREAD1 HEX 8D
1065 ASC " FIN DE CHAMP ð"
1066 HEX 00
1067 ADREAD2 HEX 8D
1068 ASC " CHAMP DONNEES ð"
1069 HEX 00
1070 ADWRITE HEX 8D8D
1071 ASC "- WRIT SYSTEM: CHAMP ADRESSE ð"
1072 HEX 00
1073 CHKSUM HEX 8D8D
1074 ASC "- CHEKSUM ð"
1075 HEX 00
1076 SYNC HEX 8D
1077 ASC "- SYNCHRO ð"
1078 HEX 00
1079 ADRTST
1080 ASC " PROD ð"
1081 HEX 00
1082 VTOC HEX 8D
1083 ASC "- TABLE VTOC PISTEð"
1084 HEX 00
1085 PISTE ASC " PISTEð"
1086 HEX 00
1087 NBSECT ASC " MAXSEð"
1088 HEX 00
1089 SECTEUR ASC " SECTEURð"
1090 HEX 00
1091 NBPISTES ASC " MAXPIð"
1092 HEX 00

```

1093 SAUVE	ASC	" SAU110"	1118 WP	ASC	**KYRAM* LE DISK EST PROTEGE EN ECRITURE."
1094	HEX	00	1119	HEX	8D00
1095 TRKC	HEX	8D	1120 KYM	HEX	8D
1096	ASC	"- PRG COURANT A\$0"	1121	ASC	**KYRAM* L'AMPERSAND EST OPERATIONNEL."
1097	HEX	00	1122	HEX	8D00
1098 LD	ASC	" L\$0"	1123	*-----	
1099	HEX	00	1124	*	
1100 CTRK	HEX	8D	1125		
1101	ASC	"- TRK COURANT KYRAM0"	1126 PRESENT	HEX	8D
1102	HEX	00	1127	ASC	" ***** ** 86"
1103 DTRK	ASC	" DOS0"	1128	HEX	8D
1104	HEX	00	1129	ASC	"G.MENIER * KYRAM V2.2 * STG"
1105 DOTTED	HEX	8D	1130	HEX	8D
1106	ASC	"-----"	1131	ASC	" *****"
1107	HEX	00	1132	HEX	8D8D00
1108 NONP	ASC	**KYRAM* PAS D'ERREUR A LA DERNIERE CMD."	1133		
1109	HEX	8D00	1134 DEBUT	JSR	\$FC58 ;HOME
1110 VOL	ASC	**KYRAM* ERREUR CONCERNANT LE VOLUME."	1135	LDA	\$/PRESENT
1111	HEX	8D00	1136	LDY	\$/PRESENT
1112 BIZ	ASC	**KYRAM* ERREUR CONCERNANT LE LECTEUR."	1137	JSR	STROUT
1113	HEX	8D00	1138		
1114 RTW	ASC	**KYRAM* RWTS NE PEUT LIRE PROLOG&EPILOG."	1139	JSR	KYRAM
1115	HEX	8D00	1140	RTS	
1116 CA	ASC	**KYRAM* CODE NON HOMOLOG DOS/PRG <> G"	1141	*-----	
1117	HEX	8D00			

Récapitulation 'KYRAM'

Après avoir saisi ce code sous
moniteur, vous le sauvegarderez par
BSAVE KYRAM,A\$3FFA,L\$D68

3FFA-	20 54 4D 4C 3C D4	4140-	20 3A DB AD 7A BC 20 DA	42C0-	20 C5 46 8D 5D B8 20 64
4000-	AD 00 04 85 1E AD 01 04	4148-	FD AD 7F BC 20 DA FD AD	42C8-	46 60 A9 0C 85 07 20 5B
4008-	85 1F A9 4B 8D 00 04 A9	4150-	84 BC 20 DA FD 20 FF 47	42D0-	46 20 6D 46 20 C5 46 8D
4010-	59 8D 01 04 20 E9 46 20	4158-	AD AE BC 20 DA FD AD B3	42D8-	AE BC 20 C5 46 8D B3 BC
4018-	0D 47 A5 1E 8D 00 04 A5	4160-	BC 20 DA FD AD B8 BC 20	42E0-	20 C5 46 8D B8 BC 20 64
4020-	1F 8D 01 04 A9 45 85 1A	4168-	DA FD 20 07 48 AD 53 B8	42E8-	46 60 A9 0C 85 07 20 5B
4028-	A9 40 85 1B A5 CE C9 28	4170-	20 DA FD AD 58 B8 20 DA	42F0-	46 20 6D 46 20 C5 46 8D
4030-	B0 10 0A 18 65 CE 18 65	4178-	FD AD 5D B8 20 DA FD 20	42F8-	9E B8 20 C5 46 8D A3 B8
4038-	1A D0 02 E6 1B 85 1A 6C	4180-	1F 47 AD 9E B8 20 DA FD	4300-	20 C5 46 8D A8 B8 20 64
4040-	1A 00 4C 3E 47 4C C6 40	4188-	AD A3 B8 20 DA FD AD A8	4308-	46 60 A9 0C 85 07 20 5B
4048-	4C E6 40 4C 6A 42 4C 70	4190-	B8 20 DA FD A9 FB A0 4A	4310-	46 20 6D 46 20 C5 46 8D
4050-	42 4C 76 42 4C 81 42 4C	4198-	20 3A DB AE 2E B9 A9 01	4318-	55 B9 20 C5 46 8D 5F B9
4058-	8A 42 4C AA 42 4C CA 42	41A0-	E0 13 F0 02 A9 00 20 DA	4320-	20 C5 46 8D 6A B9 20 64
4060-	4C EA 42 4C 0A 43 4C 2A	41A8-	FD A9 16 A0 4B 20 3A DB	4328-	46 60 A9 0C 85 07 20 5B
4068-	43 4C 4A 43 4C 64 43 4C	41B0-	AE F3 B8 A9 01 E0 F2 F0	4330-	46 20 6D 46 20 C5 46 8D
4070-	7E 43 4C 95 43 4C A9 43	41B8-	02 A9 00 20 DA FD A9 5E	4338-	E7 B8 20 C5 46 8D F1 B8
4078-	4C BD 43 4C D7 43 4C DD	41C0-	A0 4B 20 3A DB AD 92 B2	4340-	20 C5 46 8D FC B8 20 64
4080-	43 4C E3 43 4C 7A 44 4C	41C8-	20 DA FD A9 09 A0 4B 20	4348-	46 60 A9 0A 85 07 20 5B
4088-	C1 44 4C CC 44 4C D7 44	41D0-	3A DB AD 60 BC 20 DA FD	4350-	46 20 6D 46 20 C5 46 8D
4090-	4C 45 45 4C 06 45 4C 0E	41D8-	A9 55 A0 20 3A DB AD	4358-	91 B9 20 C5 46 8D 9B B9
4098-	45 4C 51 45 4C 9A 45 4C	41E0-	FE BE 20 DA FD A9 3F A0	4360-	20 64 46 60 A9 0A 85 07
40A0-	AC 45 4C DF 45 4C 12 46	41E8-	4B 20 3A DB AD 2A BF 20	4368-	20 5B 46 20 6D 46 20 C5
40A8-	4C 2A 46 4C 3F 46 4C 4D	41F0-	DA FD A9 1F A0 4B 20 3A	4370-	46 8D 35 B9 20 C5 46 8D
40B0-	46 01 60 01 00 00 00 C2	41F8-	DB AD 01 AC 20 DA FD A9	4378-	3F B9 20 64 46 60 A9 08
40B8-	40 00 20 00 00 00 00 00	4200-	4A A0 4B 20 3A DB AD 0D	4380-	85 07 20 5B 46 20 6D 46
40C0-	60 01 00 01 EF D8 20 58	4208-	B0 20 DA FD A9 69 A0 4B	4388-	20 C5 46 8D 60 BC 8D 3E
40C8-	FC A9 49 A0 48 20 3A DB	4210-	20 3A DB AD 73 AA 20 DA	4390-	B8 20 64 46 60 A9 08 85
40D0-	A9 97 A0 48 20 3A DB A9	4218-	FD AD 72 AA 20 DA FD A9	4398-	07 20 5B 46 20 6D 46 20
40D8-	11 A0 49 20 3A DB A9 B2	4220-	80 A0 4B 20 3A DB AD 61	43A0-	C5 46 8D FE BE 20 64 46
40E0-	A0 49 20 3A DB 60 20 58	4228-	AA 20 DA FD AD 60 AA 20	43A8-	60 A9 08 85 07 20 5B 46
40E8-	FC A9 49 A0 48 20 3A DB	4230-	DA FD A9 89 A0 4B 20 3A	43B0-	20 6D 46 20 C5 46 8D 2A
40F0-	A9 52 A0 4A 20 3A DB AD	4238-	DB AD B5 40 20 DA FD A9	43B8-	BF 20 64 46 60 A9 0A 85
40F8-	55 B9 20 DA FD AD 5F B9	4240-	A0 A0 4B 20 3A DB AD E7	43C0-	07 20 5B 46 20 6D 46 20
4100-	20 DA FD AD 6A B9 20 DA	4248-	03 85 06 AD E8 03 85 07	43C8-	C5 46 8D 01 AC 20 C5 46
4108-	FD 20 FF 47 AD 91 B9 20	4250-	A0 00 B1 06 85 1A C8 B1	43D0-	8D 0D B0 20 64 46 60 A9
4110-	DA FD AD 9B B9 20 DA FD	4258-	06 85 1B A0 04 B1 1A 20	43D8-	A9 8D 92 B2 60 A9 69 8D
4118-	20 07 48 AD E7 B8 20 DA	4260-	DA FD A9 AB A0 4B 20 3A	43E0-	92 B2 60 AD 7A BC 8D 20
4120-	FD AD F1 B8 20 DA FD AD	4268-	DB 60 A9 13 8D 2E B9 60	43E8-	03 AD 7F BC 8D 21 03 AD
4128-	FC B8 20 DA FD 20 FF 47	4270-	A9 00 8D 2E B9 60 A9 F2	43F0-	84 BC 8D 22 03 AD AE BC
4130-	AD 35 B9 20 DA FD AD 3F	4278-	8D F3 B8 A9 E7 8D FE B8	43F8-	8D 23 03 AD B3 BC 8D 24
4138-	B9 20 DA FD A9 DA A0 4A	4280-	60 A9 00 8D F3 B8 8D FE	4400-	03 AD B8 BC 8D 25 03 AD
		4288-	B8 60 A9 0C 85 07 20 5B	4408-	53 B8 8D 26 03 AD 58 B8
		4290-	46 20 6D 46 20 C5 46 8D	4410-	8D 27 03 AD 5D B8 8D 28
		4298-	7A BC 20 C5 46 8D 7F BC	4418-	03 AD 9E B8 8D 29 03 AD
		42A0-	20 C5 46 8D 84 BC 20 64	4420-	A3 B8 8D 2A 03 AD A8 B8
		42A8-	46 60 A9 0C 85 07 20 5B	4428-	8D 2B 03 AD 55 B9 8D 2C
		42B0-	46 20 6D 46 20 C5 46 8D	4430-	03 AD 5F B9 8D 2D 03 AD
		42B8-	53 B8 20 C5 46 8D 58 B8	4438-	6A B9 8D 2E 03 AD 91 B9

4440- 8D 2F 03 AD 9B B9 8D 30
4448- 03 AD E7 B8 8D 31 03 AD
4450- F1 B8 8D 32 03 AD FC B8
4458- 8D 33 03 AD 35 B9 8D 34
4460- 03 AD 3F B9 8D 35 03 AD
4468- 3E B8 8D 36 03 AD 01 AC
4470- 8D 37 03 AD 0D B0 8D 38
4478- 03 60 A9 08 85 07 20 5B
4480- 46 20 6D 46 A9 00 8D BD
4488- 40 20 F3 44 AD B5 40 20
4490- 5A BE A9 00 8D A9 44 A9
4498- 20 8D AA 44 A0 00 A2 60
44A0- BD 8E C0 BD 8C C0 10 FB
44A8- 8D 00 20 EE A9 44 D0 F3
44B0- EE AA 44 AD AA 44 C9 3F
44B8- D0 E9 BD 88 C0 20 64 46
44C0- 60 A9 00 8D F4 03 A9 FF
44C8- 8D 37 9E 60 A9 38 8D F4
44D0- 03 A9 A5 8D 37 9E 60 A9
44D8- 01 8D BD 40 20 E3 44 20
44E0- 64 46 60 A9 0A 85 07 20
44E8- 5B 46 20 6D 46 20 C5 46
44F0- 8D B6 40 20 C5 46 8D B5
44F8- 40 A9 00 8D BE 40 A9 40
4500- A0 B1 20 D9 03 60 A9 04
4508- 8D BD 40 4C FE 44 EA A9
4510- 00 85 1A A9 20 85 1B A9
4518- 00 85 06 A5 06 20 DA FD
4520- A9 AD 20 F0 FD A0 00 B1
4528- 1A 20 F0 FD C8 C0 20 D0
4530- F6 20 62 FC A9 20 18 65
4538- 1A 85 1A 85 06 C9 00 D0
4540- DA 20 62 FC 60 A9 02 8D
4548- BD 40 20 E3 44 20 64 46
4550- 60 AD BE 40 8D 00 03 C9
4558- 00 D0 08 A9 D5 A0 4B 20
4560- 3A DB 60 C9 10 D0 08 A9
4568- 9C A0 4C 20 3A DB 60 C9
4570- 20 D0 07 A9 FE A0 4B 20
4578- 3A DB C9 40 D0 08 A9 24
4580- A0 4C 20 3A DB 60 C9 80
4588- D0 08 A9 4B A0 4C 20 3A
4590- DB 60 A9 75 A0 4C 20 3A
4598- DB 60 A9 00 8D F6 03 A9
45A0- 40 8D F7 03 A9 C6 A0 4C
45A8- 20 3A DB 60 8D 07 C0 A9
45B0- 12 85 07 20 5B 46 20 6D
45B8- 46 20 C5 46 85 3D 20 C5
45C0- 46 85 3C 20 C5 46 85 3E
45C8- 20 C5 46 85 3E 20 C5 46
45D0- 85 43 20 C5 46 85 42 38
45D8- 20 63 C3 20 64 46 60 8D
45E0- 07 C0 A9 12 85 07 20 5B
45E8- 46 20 6D 46 20 C5 46 85
45F0- 3D 20 C5 46 85 3C 20 C5
45F8- 46 85 3F 20 C5 46 85 3E
4600- 20 C5 46 85 43 20 C5 46
4608- 85 42 18 20 63 C3 20 64
4610- 46 60 8D 07 C0 AD F2 03
4618- 8D ED 03 AD F3 03 8D EE
4620- 03 A9 FE 69 FF 38 20 B0
4628- C3 60 8D 07 C0 AD F2 03
4630- 8D ED 03 AD F3 03 8D EE
4638- 03 B8 18 20 B0 C3 60 A2
4640- 00 BD 2A 46 9D 20 03 E8
4648- E0 20 D0 F5 60 A2 00 BD
4650- 12 46 9D 20 03 E8 E0 20
4658- D0 F5 60 A5 B8 85 1A A5
4660- B9 85 1B 60 A5 1A 85 B8
4668- A5 1B 85 B9 60 A5 08 18
4670- 69 05 A4 09 85 B8 84 B9
4678- A0 05 B1 08 C9 22 D0 18
4680- A4 07 B1 08 C9 22 D0 03
4688- E6 B8 60 20 E4 FB A9 0F
4690- A0 48 20 3A DB 4C 3C D4
4698- 20 B1 00 A0 01 B1 83 85
46A0- B8 C8 B1 83 85 B9 A5 11
46A8- C9 FF D0 DF 60 C9 30 90
46B0- DA C9 3A B0 04 18 E9 2F
46B8- 60 C9 41 90 CE C9 47 B0
46C0- CA 18 E9 36 60 A0 00 B1
46C8- B8 20 AD 46 0A 0A 0A
46D0- 85 06 C8 B1 B8 20 AD 46
46D8- 18 65 06 AA A5 B8 18 69
46E0- 02 90 02 E6 B9 85 B8 8A
46E8- 60 A9 00 85 08 A9 03 85
46F0- 09 A2 00 4C FA 46 20 B1
46F8- 00 E8 A0 00 B1 B8 48 8A
4700- A8 68 91 08 C9 3A F0 04
4708- C9 00 D0 EA 60 A9 00 85
4710- CE A9 4B 85 18 A9 47 85
4718- 19 A0 00 B1 08 D1 18 D0
4720- 06 C8 C0 05 D0 F5 60 A5
4728- 18 18 69 05 85 18 A5 19
4730- 69 00 85 19 A6 CE E8 86
4738- CE E0 28 D0 DC 60 20 E4
4740- FB A9 2C A0 48 20 3A DB
4748- 4C 3C D4 43 4F 4D 4D 53
4750- 4C 52 57 54 53 43 48 4B
4758- 53 55 4E 4F 43 48 4B 54
4760- 53 54 50 52 4E 4F 50 52
4768- 4F 57 41 50 52 4F 57 44
4770- 50 52 4F 57 41 45 50 49
4778- 57 44 45 50 49 52 41 50
4780- 52 4F 52 44 50 52 4F 52
4788- 41 45 50 49 52 44 45 50
4790- 49 53 59 4E 43 52 4D 41
4798- 58 50 49 4D 41 58 53 45
47A0- 56 54 53 45 54 41 53 41
47A8- 55 56 50 53 41 55 56 51
47B0- 55 49 44 21 52 44 54 52
47B8- 4B 52 45 53 45 54 4E 52
47C0- 45 53 45 53 45 43 52 45
47C8- 53 45 43 57 52 49 4E 49
47D0- 54 44 42 55 46 41 53 44
47D8- 49 41 47 4E 4B 59 52 41
47E0- 4D 41 4D 4F 56 45 50 4D
47E8- 4F 56 45 41 45 58 45 43
47F0- 50 45 58 45 43 52 45 54
47F8- 50 52 52 45 54 41 55 A9
4800- 9A A0 4A 20 3A DB 60 A9
4808- BA A0 4A 20 3A DB 60 8D
4810- AA CB D9 D2 C1 CD AA A0
4818- C5 D2 D2 C5 D5 D2 A0 C4
4820- A7 CF D0 C5 D2 C1 CE C4
4828- C5 A0 AE 00 8D AA CB D9
4830- D2 C1 CD AA A0 C5 D2 D2
4838- C5 D5 D2 A0 C4 C5 A0 D3
4840- D9 CE D4 C1 D8 C5 A0 AE
4848- 00 8D D3 D4 C7 A0 A0 A0
4850- A0 A0 A0 A0 A0 A0 A0 AA
4858- AA AA AA AA AA AA AA AA
4860- AA AA 8D A0 A0 A0 A0 A0
4868- A0 A0 A0 A0 A0 A0 A0 A0
4870- AA A0 A0 CB D9 D2 C1 CD
4878- A0 A0 AA 8D A0 A0 A0 A0
4880- A0 A0 A0 A0 A0 A0 A0 A0
4888- A0 AA AA AA AA AA AA AA
4890- AA AA AA AA 8D 8D 00 AD
4898- AD AD AD AD AD AD AD AD
48A0- AD AD AD AD AD C3 CF CD
48A8- CD C1 CE C4 C5 D3 AD AD
48B0- AD AD AD AD AD AD AD AD
48B8- AD AD AD AD AD AD AD 8D
48C0- AF A0 A6 C3 CF CD CD D3
48C8- A0 A0 AF A0 A6 CC D2 D7
48D0- D4 D3 A0 A0 AF A0 A6 D1
48D8- D5 C9 C4 A1 A0 A0 AF A0
48E0- A6 C2 D5 C6 C1 D3 A0 8D
48E8- AF A0 A6 C3 C8 CB D3 D5
48F0- A0 A0 AF A0 A6 CE CF C3
48F8- C8 CB A0 A0 AF A0 A6 D4
4900- D3 D4 D0 D2 A0 A0 AF A0
4908- A6 CE CF D0 D2 CF A0 8D
4910- 00 AF A0 A6 D7 C1 D0 D2
4918- CF DF A0 AF A0 A6 D7 C4
4920- D0 D2 CF DF A0 AF A0 A6
4928- D7 C1 C5 D0 C9 DF A0 AF
4930- A0 A6 D7 C4 C5 D0 C9 DF
4938- 8D AF A0 A6 D2 C1 D0 D2
4940- CF DF A0 AF A0 A6 D2 C4
4948- D0 D2 CF DF A0 AF A0 A6
4950- D2 C1 C5 D0 C9 DF A0 AF
4958- A0 A6 D2 C4 C5 D0 C9 DF
4960- 8D AF A0 A6 D3 D9 CE C3
4968- D2 DF A0 AF A0 A6 CD C1
4970- D8 D0 C9 DF A0 AF A0 A6
4978- CD C1 D8 D3 C5 DF A0 AF
4980- A0 A6 D6 D4 D3 C5 D4 DF
4988- 8D AF A0 A6 C1 D3 C1 D5
4990- D6 A0 A0 AF A0 A6 D0 D3
4998- C1 D5 D6 A0 A0 AF A0 A6
49A0- D2 C5 D3 C5 D4 A0 A0 AF
49A8- A0 A6 CE D2 C5 D3 C5 A0
49B0- 8D 00 AF A0 A6 D2 C4 D4
49B8- D2 CB DF A0 AF A0 A6 D3
49C0- C5 C3 D2 C5 DF A0 AF A0
49C8- A6 D3 C5 C3 D7 D2 DF A0
49D0- AF A0 A6 C9 CE C9 D4 C4
49D8- A0 8D AF A0 A6 C1 CD CF
49E0- D6 C5 DF A0 AF A0 A6 D0
49E8- CD CF D6 C5 DF A0 AF A0
49F0- A6 C1 C5 D8 C5 C3 A0 A0
49F8- AF A0 A6 D0 C5 D8 C5 C3
4A00- 8D AF A0 A6 D2 C5 D4 D0
4A08- D2 A0 A0 AF A0 A6 D2 C5
4A10- D4 C1 D5 A0 A0 AF A0 A6
4A18- C4 C9 C1 C7 CE A0 A0 AF
4A20- A0 A6 CB D9 D2 C1 CD 8D
4A28- AD AD AD AD AD AD AD AD
4A30- AD AD AD AD AD AD AD AD
4A38- AD AD AD AD AD AD AD AD
4A40- AD AD AD AD AD AD AD AD
4A48- AD AD AD AD AD AD AD AD
4A50- 8D 00 AD AD AD AD AD AD
4A58- AD AD AD AD AD AD AD D2
4A60- D7 D4 D3 AD D0 C1 D2 C1
4A68- CD C5 D4 D2 C5 D3 AD AD
4A70- AD AD AD AD AD AD AD AD
4A78- AD AD 8D AD A0 D2 C5 C1
4A80- C4 A0 D3 D9 D3 D4 C5 CD
4A88- BA A0 C3 C8 C1 CD D0 A0
4A90- C1 C4 D2 C5 D3 C3 C5 A0
4A98- FC 00 8D A0 A0 A0 A0 A0
4AA0- A0 A0 A0 A0 A0 A0 A0 A0
4AA8- A0 A0 C6 C9 CE A0 C4 C5
4AB0- A0 C3 C8 C1 CD D0 A0 A0

4AB8- FC 00 8D A0 A0 A0 A0 A0
 4AC0- A0 A0 A0 A0 A0 A0 A0 A0
 4AC8- A0 A0 C3 C8 C1 CD D0 A0
 4AD0- C4 CF CE CE C5 C5 D3 A0
 4AD8- FC 00 8D 8D AD A0 D7 D2
 4AE0- C9 D4 A0 D3 D9 D3 D4 C5
 4AE8- CD BA A0 C3 C8 C1 CD D0
 4AF0- A0 C1 C4 D2 C5 D3 D3 C5
 4AF8- A0 FC 00 8D 8D AD A0 C3
 4B00- C8 C5 CB D3 D5 CD A0 FC
 4B08- 00 8D AD A0 D3 D9 CE C3
 4B10- C8 D2 CF A0 FC 00 A0 A0
 4B18- D0 D2 CF C4 A0 FC 00 8D
 4B20- AD A0 D4 C1 C2 CC C5 A0
 4B28- D6 D4 CF C3 A0 A0 A0 D0
 4B30- C9 D3 D4 C5 FC 00 A0 A0
 4B38- D0 C9 D3 D4 C5 FC 00 A0
 4B40- A0 A0 A0 CD C1 D8 D3 C5
 4B48- FC 00 A0 A0 D3 C5 C3 D4
 4B50- C5 D5 D2 FC 00 A0 A0 CD
 4B58- C1 D8 D0 C9 FC 00 A0 A0
 4B60- A0 A0 D3 C1 D5 B1 B1 FC
 4B68- 00 8D AD A0 D0 D2 C7 A0
 4B70- C3 CF D5 D2 C1 CE D4 A0
 4B78- A0 A0 A0 A0 C1 A4 FC 00
 4B80- A0 A0 A0 A0 CC A4 FC
 4B88- 00 8D AD A0 D4 D2 CB A0
 4B90- C3 CF D5 D2 C1 CE D4 A0
 4B98- A0 CB D9 D2 C1 CD FC 00

4BA0- A0 A0 A0 A0 A0 A0 C4 CF
 4BA8- D3 FC 00 8D AD AD AD AD
 4BB0- AD AD AD AD AD AD AD AD
 4BB8- AD AD AD AD AD AD AD AD
 4BC0- AD AD AD AD AD AD AD AD
 4BC8- AD AD AD AD AD AD AD AD
 4BD0- AD AD AD AD 00 AA CB D9
 4BD8- D2 C1 CD AA A0 D0 C1 D3
 4BE0- A0 C4 A7 C5 D2 D2 C5 D5
 4BE8- D2 A0 C1 A0 CC C1 A0 C4
 4BF0- C5 D2 CE C9 C5 D2 C5 A0
 4BF8- C3 CD C4 AE 8D 00 AA CB
 4C00- D9 D2 C1 CD AA A0 C5 D2
 4C08- D2 C5 D5 D2 A0 C3 CF CE
 4C10- C3 C5 D2 CE C1 CE D4 A0
 4C18- CC C5 A0 D6 CF CC D5 CD
 4C20- C5 AE 8D 00 AA CB D9 D2
 4C28- C1 CD AA A0 C5 D2 D2 C5
 4C30- D5 D2 A0 C3 CF CE C3 C5
 4C38- D2 CE C1 CE D4 A0 CC C5
 4C40- A0 CC C5 C3 D4 C5 D5 D2
 4C48- AE 8D 00 AA CB D9 D2 C1
 4C50- CD AA A0 D2 D7 D4 D3 A0
 4C58- CE C5 A0 D0 C5 D5 D4 A0
 4C60- CC C9 D2 C5 A0 D0 D2 CF
 4C68- CC CF C7 A6 C5 D0 C9 CC
 4C70- CF C7 AE 8D 00 AA CB D9
 4C78- D2 C1 CD AA A0 C3 CF C4
 4C80- C5 A0 CE CF CE A0 C8 CF

4C88- CD CF CC CF C7 A0 C4 CF
 4C90- D3 AF D0 D2 C7 A0 BC BE
 4C98- A0 C7 8D 00 AA CB D9 D2
 4CA0- C1 CD AA A0 CC C5 A0 C4
 4CA8- C9 D3 CB A0 C5 D3 D4 A0
 4CB0- D0 D2 CF D4 C5 C7 C5 A0
 4CB8- C5 CE A0 C5 C3 D2 C9 D4
 4CC0- D5 D2 C5 AE 8D 00 8D AA
 4CC8- CB D9 D2 C1 CD AA A0 CC
 4CD0- A7 C1 CD D0 C5 D2 D3 C1
 4CD8- CE C4 A0 C5 D3 D4 A0 CF
 4CE0- D0 C5 D2 C1 D4 C9 CF CE
 4CE8- CE C5 CC AE 8D 00 8D A0
 4CF0- A0 A0 A0 A0 A0 A0 A0 A0
 4CF8- A0 A0 A0 A0 AA AA AA AA
 4D00- AA AA AA AA AA AA AA AA
 4D08- AA AA AA A0 8D C7 AE CD
 4D10- C5 CE C9 C5 D2 A0 A0 A0
 4D18- A0 A0 A0 AA A0 CB D9 D2
 4D20- C1 CD A0 D6 B2 AE B2 A0
 4D28- AA A0 A0 A0 A0 A0 A0 A0
 4D30- A0 D3 D4 C7 8D A0 A0 A0
 4D38- A0 A0 A0 A0 A0 A0 A0 A0
 4D40- A0 A0 A0 AA AA AA AA AA
 4D48- AA AA AA AA AA AA AA AA
 4D50- AA 8D 8D 00 20 58 FC A9
 4D58- EE A0 4C 20 3A DB 20 9A
 4D60- 45 60

Les reliures Pom's

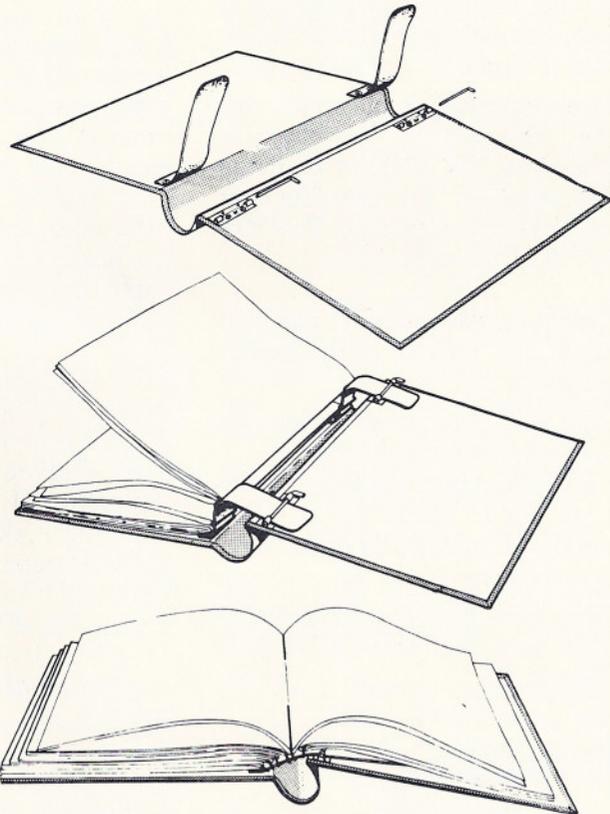
Pour nos lecteurs comme pour nous, la revue Pom's est une source précieuse d'information. À ce titre, nous sommes amenés à la consulter souvent.

Pour faire suite à de nombreuses demandes, et afin de faciliter le rangement et les recherches dans Pom's, les Éditions MEV vous proposent des reliures toilées propres à recevoir six numéros, soit un an de documentation.

Une fois reliées, les revues forment un livre agréable à utiliser et insensible aux manipulations répétées.

Ces reliures sont de couleur Bordeaux, le logo 'Pom's' est frappé au dos et un aplat est réservé pour noter l'année.

Reliures Pom's : 60,00 Frs franco



ProDOS et fichiers détruits

Patrice Neveu

Si vous avez tendance à 'bidouiller' en assembleur dans les méandres de votre Apple, il peut vous arriver d'exciter les adresses fatidiques situées vers \$COE0, qui déclenchent les routines 'hardware' d'écriture sur disque.

Vue l'heure souvent tardive, le désordre régnant sur le bureau, et le walkman qui vous distille une mélodie hypnotique, la conséquence quasi-assurée est l'effacement total d'une piste, souvent la piste zéro.

Sous DOS 3.3, cela affectait rarement l'intégrité logique de la disquette. En revanche, sous ProDOS, l'ensemble des pointeurs internes permettant d'accéder aux fichiers ont disparu et la disquette est apparemment bonne pour la casse...

Apparemment seulement car, avec un peu de patience et d'habileté, les fichiers pourront être restaurés. En effet, ils sont toujours présents mais ProDOS ne sait plus les retrouver.

C'est donc par nécessité que sont nées les trois commandes suivantes :

- INDEX, pour retrouver tous les blocs index des fichiers ;
- RBLOC pour lire directement un bloc en mémoire ;
- WBLOC pour écrire directement un bloc en mémoire.

Elles sont entièrement compatibles avec la routine CMDLOAD (Pom's 20).

Utilisation

Récupérer l'intégrité des fichiers ne peut pas être une opération automatique (ne rêvons pas !). En revanche, ces commandes ad-

ditionnelles vous aideront considérablement, pourvu que vous ayez une connaissance minimum de la structure d'un volume ProDOS.

Blocs	Contenu
0-1	Image du programme exécuté lors du démarrage du disque
2	Premier bloc du directory
3-5	Suite du directory
6	Carte des blocs utilisés
7-279	Blocs libres, index ou données

Dans mon cas, le démarrage incontrôlé du disque a eu pour effet non seulement d'effacer toute la piste zéro donc les blocs 0 à 7, mais en plus de la rendre inutilisable. Elle avait été reformatée de façon fantaisiste, ce qui donnait des bruits inquiétants lorsque j'essayais de faire ne serait-ce qu'un CATALOG.

Pour observer les causes des désastres, il suffit de charger un *Nibble Editor* et de comparer la piste 0 avec celle d'une autre disquette. Il est nécessaire, de toute façon, de reformater la piste endommagée, avec un utilitaire permettant un formatage partiel d'une disquette, puis de recopier la piste 0 issue d'une autre disquette ProDOS (votre copieur préféré fait probablement l'affaire).

Ceci fait, on dispose d'une disquette bonne en apparence, mais toujours inutilisable, car le 'directory' recopié n'aiguille pas sur les bons fichiers. Ainsi, avant

de modifier le directory, il faudra connaître les emplacements de tous les fichiers à récupérer. Nous allons donc devoir les localiser, grâce à la commande INDEX.

La commande INDEX

Pour comprendre le fonctionnement de la commande index, il faut étudier la structure d'un fichier sous ProDOS. Dans la majorité des cas, c'est-à-dire pour les fichiers de plus de 512 octets et de moins de 128Ko (Sapling files), il est construit ainsi :

- un bloc index contient tous les numéros de blocs sur lesquels le programme est enregistré ;
- puis, éparpillés sur la disquette, on trouve ces blocs.

Sous ProDOS, les fichiers de moins de 512 octets (seedling files) sont pratiquement impossibles à retrouver car, étant donné qu'ils tiennent sur un bloc, ils ne sont pas localisés par un bloc index. Les fichiers de plus de 128Ko et moins de 16 mégas (tree files) sont localisés par un bloc index "maître", c'est-à-dire un index qui pointe vers un ensemble d'index, qui pointent à leur tour vers les blocs de données du fichier.

Ainsi, lorsqu'on ne peut plus se fier au directory pour retrouver ses fichiers, le mieux est de chercher les blocs index. La commande INDEX va lire les blocs du disque un par un, et tenter de déterminer si un bloc peut ou non, être un bloc index. Elle a toutefois besoin que vous lui fournissiez le nombre de blocs que contient la disquette :

- 280 pour un disque 5,25 pouces ;
- 1600 pour un disque 3,5 pouces ;
- 127 pour le disque virtuel en mémoire auxiliaire.

Les valeurs pour les ProFile et autres disques durs sont simples à déterminer, puisqu'un CATALOG affiche le nombre total de blocs. Bien sûr, mieux vaut avoir de la mémoire car, si c'est de votre disque dur dont il s'agit, vous n'en n'aurez peut-être pas un second sur lequel faire un essai...

En cas de doute, donnez un nombre supérieur à la réalité : à partir d'un certain moment, INDEX affichera une suite ininterrompue de numéros de bloc en inverse ; vous en déduirez que le nombre de blocs du volume est immédiatement inférieur à cette suite. En règle générale, tout numéro de bloc affiché en inverse indique qu'il est impossible de lire ce bloc.

INDEX s'inspire de la routine FIB publiée dans "Beneath Apple ProDOS", mais en améliorant grandement les performances. Avec INDEX, un bloc index est automatiquement détecté comme tel (à l'exception des derniers blocs de directory) ; au contraire FIB produisait une très longue liste de blocs potentiellement index, dont bien peu l'étaient en réalité. L'algorithme utilisé est, en effet, plus discriminatoire.

Durant l'exécution de la commande, des numéros de blocs seront affichés (en affichage normal). Ce sont les blocs présumés être index. Sortez-les sur imprimante, ou notez-les, car nous allons maintenant les utiliser.

Les commandes RBLOC & WBLOC

RBLOC et WBLOC sont deux commandes supplémentaires : il faudra lire un bloc par RBLOC,

n° bloc précédent n° bloc suivant

	BLOC2		
	0900:	00 00 03 00 F9 48 41 52	...yHAR
	0908:	44 43 4F 50 49 45 00 00	DCOPIE..
	0910:	00 00 00 00 00 00 00 00
	0918:	00 00 00 00 00 00 00 00
	0920:	01 00 C3 27 0D 12 00 06	..C'....
	0928:	00 18 01 27 48 41 52 44	...'HARD
premier fichier	0930:	4D 41 43 00 00 00 00 00	MAC.....
	0938:	00 00 00 06 08 00 03 00
	0940:	36 02 00 00 00 00 00 00	6.....
	0948:	00 03 00 70 00 00 00 00
second fichier	0950:	02 00 26 50 52 4F 44 4F	..&PRODO
	0958:	53 00 00 00 00 00 00 00	S.....
Ce fichier contient \$000236 octets (566 en décimal)	0960:	00 00 FF 0B 00 1E 00 00
	0968:	0A 00 00 00 00 00 00 00
	0970:	21 00 00 32 A9 00 00 02	!. 2)....
	0978:	00 2C 42 41 53 49 43 2E	..BASIC..
Le nom	0980:	53 59 53 54 45 4D 00 00	SYSTEM..
	0988:	00 FF 29 00 15 00 00 28	.)....(
	0990:	00 00 00 00 00 00 00 21!
	0998:	00 20 D2 A8 00 00 02 00	. R(....
	09A0:	27 43 4F 4E 56 45 52 54	'CONVERT
	09A8:	00 00 00 00 00 00 00 00

\$12 = 18 fichiers dans le catalogue

\$27 :
2 = fichier de 2 à 256 blocs
7 = longueur du nom

\$6 : c'est un fichier binaire
nb de blocs : 3

le bloc n° \$0008 est le bloc d'index, c'est ce n° qu'on modifiera

AUX_TYPE : \$7000
Comme ce fichier est du type binaire, il s'agit de son adresse de chargement.

Le bloc n° 2 (premier bloc du catalogue), tel qu'il est affiché à l'écran après la commande 'RBLOC 2'. Pour modifier le numéro de bloc index du premier fichier par exemple (n° trouvé à l'aide de la commande INDEX), passer en moniteur (CALL - 151), puis avec le curseur, relire la ligne '0938:...' en modifiant au passage le numéro. Réécrire le bloc en faisant WBLOC 2.

Pour bénéficier des commandes RBLOC, WBLOC et INDEX, il aura fallu faire '- BLOC' et '- INDEX'.

le modifier en mémoire en utilisant le moniteur de l'Apple (accessible par CALL - 151), puis réécrire le bloc par WBLOC.

L'exécution des deux commandes affiche également à l'écran les premiers octets du bloc et son adresse d'implantation. Ainsi, si la première adresse est \$900, alors le bloc se trouve en \$900-\$AFF. Les routines chargent et déchargent uniquement la mémoire vive située après un éventuel programme Applesoft et ses variables simples. En conséquence, il est préférable de ne pas modifier un programme, ou l'exécuter, entre le RBLOC et le WBLOC correspondant.

Il est maintenant préférable de s'assurer, par l'utilisation de RBLOC, que chaque bloc détecté par la routine INDEX correspond bien à un index de fichier. En particulier, le même numéro de bloc ne doit pas apparaître deux fois.

La structure du Directory

Dès lors, nous avons toutes les informations nécessaires pour ré-insérer nos fichiers dans le directory. Pour lire le directory, il faut se rappeler que celui-ci commence toujours en bloc 2 ; c'est le bloc-clé. Puis, selon le type de disque, il continue sur plusieurs blocs consécutifs. En ce qui concerne la structure du directory, seuls les octets impliqués dans la récupération des fichiers sont signalés dans la liste qui suit.

En début de chaque bloc directory, on trouve toujours ceci :

- \$00-01 numéro du bloc précédent du directory précédent. Si nul, on est dans le premier bloc ;
- \$02-03 numéro du bloc suivant du directory. Si nul, on est en fin de directory.

On ne touche pas à ces blocs

qui ne servent qu'au chaînage interne, immuable par type de support.

\$25-26 FILE_COUNT
Nombre de fichiers actifs dans le catalogue.

À mettre à jour, en fin de travail, lorsque l'on saura exactement combien de fichiers sont à récupérer et à lister lors d'un CATALOG.

Ensuite, à partir de l'octet \$2B, sont placées 13 séries de \$27 (39) octets qui décrivent individuellement les fichiers stockés. Pour chaque série, on trouve :

**\$00 STORAGE_TYPE/
NAME_LENGTH**
type de fichier/longueur du nom, au format TTTTLLLL.

Le type peut prendre les valeurs :

- \$0 Entrée effacée, disponible pour réutilisation
- \$1 Seedling file (1 bloc)
- \$2 Sapling file (2 à 256 blocs)
- \$3 Tree file (257 à 32768 blocs)
- \$D Sous-catalogue
- \$E Première entrée d'un sous-catalogue
- \$F Première entrée du catalogue (voir précédemment)

\$01-0F FILE_NAME
nom du fichier.

Au départ, mieux vaut soit ne pas y toucher, soit y mettre un caractère alphabétique. Les commandes RENAME seront pour bien plus tard.

\$10 FILE_TYPE
type du fichier

Les types qui peuvent nous intéresser sont :

- \$04 TXT fichier ASCII
- \$06 BIN binaire
- \$0F DIR sous-directory
- \$19 ADB base de donnée AppleWorks
- \$1A AWP traitement de texte AppleWorks
- \$1B ASP tableur AppleWorks

- \$EF PAS PASCAL sous ProDOS
- \$FC BAS BASIC Applesoft
- \$FE REL programme langage machine relogeable
- \$FF SYS système

Pour déterminer le type approprié, il sera préférable d'utiliser la routine TDUMP du numéro 20 de Pom's afin d'afficher le fichier et en déduire sa nature.

\$11-12 KEY_POINTEUR
numéro du bloc index.

À modifier aussitôt que l'on a utilisé INDEX et vérifié l'authenticité des blocs renvoyés.

\$13-14 BLOCKS_USED
Nombre de blocs utilisés par le fichier.

Cela comprend les blocs index plus les blocs de données. Modifier ceci en fonction encore de ce que le bloc index a donné.

\$15-17 EOF
Longueur du fichier sur 3 octets.

Au départ, y mettre 512 fois le nombre de blocs trouvé. Ensuite, il sera toujours plus facile de le modifier par voie logicielle.

\$1F-20 AUX_TYPE
dépend de FILE_TYPE

TXT Longueur d'enregistrement (paramètre L dans OPEN)

BIN Adresse de chargement (paramètre A dans BSAVE)

BAS Adresse de chargement (lorsque sauvé par SAVE, normalement \$801)

VAR Adresse des variables (sauvées par STORE)

SYS Adresse de chargement (habituellement \$2000)

Il est quasiment impossible de fixer ces adresses, à moins de se les rappeler, ou que le fichier soit issu d'un logiciel qui sauve toujours à la même adresse.

Les autres octets de chaque série sont hors-propos ici, et d'ailleurs non indispensables (par exemple les dates de création et modification).

Après avoir retrouvé les fichiers

Voilà, logiquement une fois tout ceci terminé, on devrait pouvoir prendre les utilitaires ProDOS et sauvegarder tous nos fichiers sur une bonne disquette ProDOS, car rapellons-le, ce que nous avons fait suffit seulement à reconnecter des fichiers au directory, et non à remettre la disquette entièrement en état. Entre autre, le schéma d'occupation des blocs n'est pas mis à jour, et il est donc impératif de ne sauvegarder aucun fichier sur le disque.

Syntaxe des commandes

Après avoir fait :

-INDEX
et

-BLOC,

vous disposez des commandes suivantes :

```
INDEX <nb de blocs>  
[,S<numéro de slot>]  
[,D<numéro de drive>]
```

```
RBLOC <numéro du bloc>  
[,S<numéro de slot>]  
[,D<numéro de drive>]
```

```
WBLOC <numéro du bloc>  
[,S<numéro de slot>]  
[,D<numéro de drive>]
```

Par exemple :

```
INDEX 280,S6
```

```
INDEX 1600
```

```
RBLOC 173,S5,D2
```

```
WBLOC 173
```

Pour le disque virtuel /RAM, prendre S3,D2.

Liste des fichiers sur la disquette Pom's

T. INDEX.CODE
source Big Mac (format texte),
commande INDEX

T. BLOC. CODE
idem pour commandes
RBLOC & WBLOC

INDEX
module exécutable

BLOC
module exécutable

Les commandes s'installent par
un simple BRUN sur les modules
exécutables.

ProDOS



II+
//e
//e+
//c
Igs

Au verso de la disquette d'accompagnement (face
nommée /POMS29/), les fichiers sont au format ProDOS.

Source 'T.INDEX.CODE'
Assembleur Big Mac, format TEXT

```

1      LST OFF
2
3      ORG $2100
4
5 *****
6 * INDEX: Permet de localiser tous les *
7 * blocs susceptibles d'etre des *
8 * blocs index. *
9 *
10 *****
11
12 *-----*
13 * Syntaxe: *
14 * *
15 * INDEX <total> [,S<slot>] [,D<drive>] *
16 * *
17 * Le programme va lire tous les blocs du disque *
18 * jusqu'au nombre total qu'on lui indique dans *
19 * commande. Si le bloc est susceptible d'etre un *
20 * bloc index, il affiche son numéro. *
21 * *
22 *-----*
23
24 *-----*
25 * Labels *
26 *-----*
27
28 SYNT_ERR = $40      ; Code du message 'SYNTAX
                       ; ERROR'
29 PTR       = $48      ; Pointeur temporaire
30 LINNUM    = $50      ; Résultat de GETADR
31 STREND    = $6D      ; Adresse du début de la
                       ; zone libre
32 PRETOP    = $6F      ; Adresse de la fin de la
                       ; zone libre
33 OLDXTXTPTR = $79     ; Sauvegarde de TTXTPTR
34 CR        = $8D      ; Code du Carriage Return
35 BUFADR1   = $AB      ; Adresse de la première
                       ; moitié du bloc lu
36 BUFADR2   = $AD      ; Adresse de la seconde
                       ; moitié du bloc lu
37 CHRGET    = $B1      ; Routine de prise de
                       ; caractère
38 TTXTPTR   = $B8      ; Pointe sur le caractère
                       ; à prendre
39
40 IN        = $0200    ; Buffer clavier
41
42 PRINTERR  = $BE0C    ; Affichage de l'erreur
43 DEFSLT    = $BE3C    ; Slot par défaut
44 DEFDRV    = $BE3D    ; Drive par défaut
45 XTADDR    = $BE50    ; Pointe adresse de la
                       ; commande externe
46 XLEN      = $BE52    ; Longueur du nom de la
                       ; commande moins un
47 XCNUM     = $BE53    ; Numéro de commande
48 PBITS     = $BE54    ; Indiquent les
                       ; paramètres variables
49 VPATH1    = $BE6C    ; Pointe sur le nom de
                       ; volume
50 BADCALL   = $BE8B    ; Traduit l'erreur pour
                       ; le BI
51 ENTRY     = $BF00    ; Point d'entrée du MLI
52 SYSERR    = $BF0F    ; Va à la routine de
                       ; traitement des erreurs
53
54 FRMNUM    = $DD67    ; Range valeur pointée
                       ; par TTXTPTR dans FAC
55 GETADR    = $E752    ; Transforme FAC en
                       ; entier 2 oct (LINNUM)
56 LINPTR    = $ED24    ; Affiche X et A
57 COUT      = $FDED    ; Affiche le caractère se
                       ; trouvant dans A
58 INVERSE   = $FE80    ; Affiche en inverse
59 NORMAL    = $FE84    ; Affiche en normal
60 RTS       = $FF58    ; Contient un RTS
61
62 *-----*
63 * Validité de la CMD *
64 *-----*
65
66 START     CLD        ; Obligatoire pour ProDOS
67 LDA #>FIN+$0100 ; Convention pour
                       ; CMDLOAD
68 LDA #>LONG-$0100
69 V_OLDCMD  LDA RTS

```

```

70      LDA VPATH1      ; Pointe vers commande
                       ; pour déterminer si
                       ; c'est la notre
71      STA PTR
72      LDA VPATH1+1
73      STA PTR+1
74      LDY #1
75 COMPAR    LDA (PTR),Y ; C'est INDEX ?
76      CMP COMMAND-1,Y
77      BNE NO_CMD      ; non, alors retour au
                       ; ProDOS
78
79      INY
80      CPY #5+1
81      BCC COMPAR
82      BCS BONNECMD    ; C'est INDEX alors on
                       ; peut y aller
83 NO_CMD    SEC        ; Indique que la commande
                       ; n'est pas à nous
84
85      JMP (V_OLDCMD+1)
86 *-----*
87 * ProDOS étudiée la ligne *
88 *-----*
89
90 BONNECMD
91      DEY              ; Demande à ProDOS
                       ; d'examiner la commande
                       ; Il ne doit pas y avoir
                       ; de paramètres
92
93      DEY
94
95      STY XLEN
96      LDA #0
97      STA SYSERR
98      STA XCNUM
99      LDA #0           ; Pas de paramètres: $00
                       ; en (Pbits)
100     STA PBITS+1
101 V_SUITE  LDA SUITE   ; Indique l'adresse du
                       ; retour
102
103     LDA V_SUITE+1
104     STA XTADDR
105     LDA V_SUITE+2
106     STA XTADDR+1
107     CLC              ; C'était bien pour nous
108     RTS
109
110     CMP #0           ; Si pas d'erreur, on
                       ; continue
111
112     BNE NO_CMD
113 *-----*
114 * Lecture du nombre de blocs *
115 *-----*
116
117     LDA TTXTPTR      ; Sauvegarde de TTXTPTR
                       ; pour que l'exécution
                       ; d'un programme Basic
                       ; puisse se continuer.
118
119     LDX TTXTPTR+1
120
121     LDY XLEN         ; Où se trouve la fin du
                       ; mot de commande ?
122
123     INY              ; Sert à lire la suite:
                       ; numéro de bloc,
                       ; puis les paramètres
                       ; éventuels (S et D)
124
125     TYA              ; Pour cela, on se sert
                       ; de TTXTPTR:
126
127     CLC              ; On additionne la
                       ; longueur du mot-clé
                       ; (INDEX) au début de la
                       ; ligne de commande.
128
129     LDA VPATH1+1
130
131     ADC #0           ; C'est ce qu'on met en
                       ; TTXTPTR.
132
133     STA TTXTPTR+1
134
135     JSR RD_NBR       ; On va lire le nombre
                       ; qui est pointé
                       ; Ce sera le bloc final
136 *-----*
137 * Lecture des paramètres s'il y en a *
138 *-----*
139

```

Récapitulation
'INDEX'

Après avoir saisi ce code sous
moniteur, vous le sauvegarderez
par BSAVE INDEX,A\$2000,L\$2D5

Les codes de \$2000 à \$20FF
correspondent au programme
CMDLOAD (Pom's 20).

```

2000:AD 00 BF C9 4C F0 05 A9
2008:87 4C ED FD AD 4D BE F0
2010:05 A9 15 4C 09 BE AD 04
2018:21 69 00 20 98 20 90 05
2020:A9 0E 4C 09 BE CD 02 21
2028:90 F6 AE 08 BE 8D 08 BE
2030:8E 07 21 AE 07 BE 8E 06
2038:21 A0 00 8C 07 BE 48 E9
2040:21 85 3C 68 38 E9 04 85
2048:74 A9 21 85 49 84 48 A8
2050:00 B1 48 F0 27 20 8E F0
2058:A4 2F C0 02 D0 0F B1 48
2060:C9 21 90 09 CD 02 21 B0
2068:04 65 3C 91 48 A5 48 38
2070:65 2F 85 48 A5 49 69 00
2078:85 49 D0 D3 A0 00 A9 21
2080:84 3C 85 3D 18 6D 04 21
2088:84 42 88 84 3E 85 3F AD
2090:08 BE 85 43 C8 4C 2C FE
2098:8D FB 20 A5 74 18 69 04
20A0:8D FC 20 86 3D CE FC 20
20A8:F0 47 AD FC 20 8D FD 20
20B0:AD FD 20 48 4A 4A 4A AA
20B8:68 29 07 A8 B9 F3 20 3D
20C0:58 BF D0 E1 A5 3D D0 09
20C8:B9 F3 20 1D 58 BF 9D 58
20D0:BF AD FC 20 38 CE FD 20
20D8:ED FD 20 CD FB 20 D0 D0
20E0:A5 3D D0 07 18 AE FD 20
20E8:E8 8A 60 A9 00 85 3D F0
20F0:B9 38 60 80 40 20 10 08
20F8:04 02 01 00 00 00 AD 99
2100:D8 A9 23 A9 01 AD 58 FF
2108:AD 6C BE 85 48 AD 6D BE
2110:85 49 A0 01 B1 48 D9 CA
2118:22 D0 07 C8 C0 06 90 F4
2120:B0 04 38 6C 06 21 88 88
2128:8C 52 BE A9 00 8D 0F BF
2130:8D 53 BE A9 00 8D 54 BE
2138:A9 00 8D 55 BE AD 4E 21
2140:AD 3E 21 8D 50 BE AD 3F
2148:21 8D 51 BE 18 60 C9 00
2150:D0 D0 A5 B8 A6 B9 85 79
2158:86 7A AC 52 BE C8 C8 98
2160:18 6D 6C BE 85 B8 AD 6D
2168:BE 69 00 85 B9 20 82 22
2170:8D D0 22 8E D1 22 20 AF
2178:22 20 B8 22 F0 28 C9 AC
2180:D0 F7 20 B8 22 C9 D3 D0
2188:0B 20 B8 22 29 07 8D 3C
2190:BE 4C 79 21 C9 C4 F0 03
2198:4C 8E 22 20 B8 22 29 03
21A0:8D 3D BE 4C 79 21 A9 00
21A8:8D CA 22 8D C9 22 AD C5
21B0:22 AD AF 21 AE B0 21 8D

```

140	JSR	IN_PTR	; Refixe le pointeur au début du buffer	224	BNE	CHKNG2	peut etre un index ; Sinon, on fait une autre vérification	21B8:C2 22 8E C3 22 AD 3D BE
141				225				21C0:29 02 0A 0A 0D 3C BE 0A
142	CH_VRGL	JSR	CHR_GET ; Prend caractère jusqu'à ce que soit une	226	VRFZ	LDA	(BUFADR1),Y ; La méthode a été entièrement changée	21C8:0A 0A 0A 8D C6 22 A6 6E
143	BEQ	EOL	; virgule, car cela signifierait un numéro	227	ORA	(BUFADR2),Y ; par rapport à 'Beneath Apple ProDOS'	21D0:E8 E4 70 90 05 A9 56 4C	
144	CMP	"#,"	; de drive et/ou de slot à lire.	228	BNE	VRFRS	; Car elle donnait trop de blocs non-index	21D8:90 22 8E C8 22 20 BE 22
145	BNE	CH_VRGL		229	INY		; Avec celle-ci tous les essais faits	21E0:B0 F5 AC C7 22 AE C8 22
146				230	BNE	VRFZ	; seul dernier bloc du catalog est indiqué	21E8:84 AB 86 AC E8 84 AD 86
147	JSR	CHR_GET	; Prend le caractère de commande	231	VRFRS	RTS	; à tort comme étant un index.	21F0:AE A0 00 20 FA 21 F0 6D
148	CMP	"#S"	; C'est le changement de Slot ?	232				21F8:D0 0A B1 AB 11 AD D0 03
149	BNE	TEST_D	; Non, alors peut-etre est-ce le drive ?	233	CHKNG2			2200:C8 D0 F7 60 A0 00 8C D2
150	JSR	CHR_GET	; Quel est le nouveau Slot ?	234	LDY	#0	; Le principe: tester si tous les numéros	2208:22 AC D2 22 B1 AB F0 3A
151	AND	##00000111		235	STY	COMPTEUR	; 16 bits différents les uns des autres,	2210:8D D3 22 B1 AD 8D D4 22
152	STA	DEFSLT	; A mettre chez ProDOS	236	CHKNG3		; puisque index ne peut contenir 2 fois le	2218:C8 F0 2A B1 AD CD D1 22
153	JMP	CH_VRGL		237	LDY	COMPTEUR	; meme bloc.	2220:90 09 D0 41 B1 AB CD D0
154				238	LDA	(BUFADR1),Y	; Numéro zéro indique la fin de la liste	2228:22 B0 3A B1 AB D0 08 C8
155	TEST_D	CMP	"#D" ; C'est le changement de Drive ?	239	BEQ	GOOD_ONE	; Donc si on y est, c'est un index	2230:20 FA 21 F0 10 D0 2E CD
156	BEQ	NOERRS		240	STA	TEMP	; Sinon on stocke pour la vérification	2238:D3 22 D0 DC B1 AD CD D4
157	JMP	ERR_STX		241	LDA	(BUFADR2),Y		2240:22 D0 D5 F0 20 EE D2 22
158	NOERRS	JSR	CHR_GET ; Quel est le nouveau drive ?	242	STA	TEMP+1		2248:D0 BF C0 00 F0 17 20 54
159	AND	##00000011		243				2250:22 4C 65 22 AE C9 22 AD
160	STA	DEFDRV		244	NXTBYTE	INY	; On compare au numéro d'après	2258:CA 22 20 24 ED 20 84 FE
161	JMP	CH_VRGL		245	BEQ	EOI	; Si on est à la fin du bloc	2260:A9 AC 4C ED FD EE C9 22
162	EOL			246	LDA	(BUFADR2),Y		2268:D0 03 EE CA 22 AD CA 22
163				247	CMP	TOT_BLOC+1	; Numéro ne peut etre supérieur au nbr max	2270:CD D1 22 F0 03 4C AE 21
164	*	-----		248	BCC	V1		2278:AD C9 22 CD D0 22 90 F5
165	*	Boucle de	lecture des blocs	249	BNE	NXTBLK	; Sinon, pas un bloc index qu'on teste	2280:18 60 A0 00 B1 B8 C9 30
166	*	-----		250	LDA	(BUFADR1),Y		2288:90 04 C9 3A 90 16 A9 40
167				251	CMP	TOT_BLOC		2290:C9 27 F0 0A C9 28 F0 06
168	LDA	#0	; Commence au bloc 0	252	BCS	NXTBLK		2298:20 8B BE 4C 0C BE 20 80
169	STA	BLOC_DEP+1		253	V1	LDA	(BUFADR1),Y ; Si on tombe sur un zéro, ça signifie ; peut-etre une fin de liste.	22A0:FE 4C 4E 22 20 67 DD 20
170	STA	BLOC_DEP		254	BNE	V2		22A8:52 E7 A5 50 A6 51 60 A0
171				255				22B0:00 A2 02 84 B8 86 B9 60
172	*	-----		256	INY		; Donc on regarde si ce qui suit est nul ; Va vérifier	22B8:20 B1 00 C9 8D 60 20 00
173	*	Lecture du	bloc indiqué	257	JSR	VRFZ		22C0:BF 80 C5 22 60 03 60 00
174	*	-----		258	BEQ	EOI		22C8:00 00 49 4E 44 45 58
175				259	BNE	NXTBLK	; Sinon, ce n'était pas un index	22D0:00 00 00 00 00
176	VC_PARM	LDA	PARM ; Nécessaire à cause du relogement	260				
177	LDA	VC_PARM+1	; Indique où est la table de paramètres	261	V2	CMP	TEMP ; Ce n'était pas un zéro, donc on vérifie	
178	LDX	VC_PARM+2		262	BNE	NXTBYTE	; le bas du numéro.	
179	STA	VC_MLI		263	LDA	(BUFADR2),Y	; Si pareil en bas, il se peut que le	
180	STX	VC_MLI+1		264	CMP	TEMP+1	; haut soit différent...	
181				265	BNE	NXTBYTE	; Si oui, alors prochain octet	
182	LDA	DEFDRV	; Codage du numéro d'unité sous la forme	266	BEQ	NXTBLK	; Si non, prochain bloc	
183	AND	##00000010	; DSSS0000	267	INC	COMPTEUR	; Tous les octets étaient différent à celui	
184	ASL			268	EOI			
185	ASL			269	INC	COMPTEUR	; testé, donc on prend le suivant.	
186	ORA	DEFSLT		270	BNE	CHKNG3		
187	ASL			271	GOOD_ONE	CPY	#0 ; Si ler octet du bloc est un zéro, alors	
188	ASL			272		BEQ	NXTBLK ; ça ne peut pas etre un index	
189	ASL			273	CONT	JSR	PBLOCK ; Affiche le numéro de bloc	
190	ASL			274	JMP	NXTBLK	; Passe au prochain bloc	
191	STA	PARM+1		275	*	-----		
192				276	*	-----		
193	LDX	STREND+1	; Installe le buffer dans la zone libre	277	*	-----		
194	INX			278	*	-----		
195	CPX	FRETOP+1		279	*	-----		
196	BCC	FIX_BUF		280	*	-----		
197				281	*	-----		
198	LDA	#\$56	; S'il n'y en a pas assez, alors on sort	282	*	-----		
199	ERR	JMP	ERR_EXIT ; 'NO BUFFER AVAILABLE'	283	*	-----		
200				284	PBLOCK	LDX	BLOC_DEP ; Affichage en décimal du numéro de bloc,	
201	FIX_BUF	STX	BUFF+1 ; Sinon, enregistre la page du buffer	285		LDA	BLOC_DEP+1 ; suivi d'une virgule pour tous les séparer	
202	JSR	RD_BLOC	; Puis va lire le bloc en question	286		JSR	LINPTR	
203	BCS	ERR	; Au cas où il y aurait une erreur	287		JSR	NORMAL	
204				288		LDA	"#,"	
205	*	-----		289		JMP	COUT	
206	*	Analyse du	bloc	290	*	-----		
207	*	-----		291	*	-----		
208				292	*	-----		
209	LDY	BUFF	; Prépare des pointeurs pour indiquer	293	*	-----		
210	LDX	BUFF+1	; l'adresse de la première page du buffer,	294	*	-----		
211	STY	BUFADR1	; et celle de la seconde page du buffer.	295	*	-----		
212	STX	BUFADR1+1		296	NXTBLK	INC	BLOC_DEP ; Passe au prochain bloc tout en vérifiant	
213	INX			297				
214	STY	BUFADR2						
215	STX	BUFADR2+1						
216								
217	*	-----						
218	*	Teste si	le bloc peut etre un index					
219	*	-----						
220								
221	LDY	#0	; Se place au début du bloc					
222	JSR	VRFZ	; Va vérifier s'il est complètement nul					
223	BEQ	NXTBLK	; Si oui, alors ça ne					

Récapitulation 'BLOC'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE BLOC,AS2000,LS2A4

Les codes de \$2000 à \$20FF correspondent au programme CMDLOAD (Pom's 20).

```

2000:AD 00 BF C9 4C F0 05 A9
2008:87 4C ED FD AD 4D BE F0
2010:05 A9 15 4C 09 BE AD 04
2018:21 69 00 20 98 20 90 05
2020:A9 0E 4C 09 BE CD 02 21
2028:90 F6 AE 08 BE 8D 08 BE
2030:8E 07 21 AE 07 BE 8E 06
2038:21 A0 00 8C 07 BE 48 E9
2040:21 85 3C 68 38 E9 04 85
2048:74 A9 21 85 49 84 48 A0
2050:00 B1 48 F0 27 20 8E F8
2058:A4 2F C0 02 D0 0F B1 48
2060:C9 21 90 09 CD 02 21 B0
2068:04 65 3C 91 48 A5 48 38
2070:65 2F 85 48 A5 49 69 00
2078:85 49 D0 D3 A0 00 A9 21
2080:84 3C 85 3D 18 6D 04 21
2088:84 42 88 84 3E 85 3F AD
2090:08 BE 85 43 C8 4C 2C FE
2098:8D FB 20 A5 74 18 69 04
20A0:8D FC 20 86 3D CE FC 20
20A8:F0 47 AD FC 20 8D FD 20
20B0:AD FD 20 48 4A 4A AA
20B8:68 29 07 A8 B9 F3 20 3D
20C0:58 BF D0 E1 A5 3D D0 09
20C8:B9 F3 20 1D 58 BF 9D 58
20D0:BF AD FC 20 38 CE FD 20
20D8:ED FD 20 CD FB 20 D0 D0
20E0:A5 3D D0 07 18 AE FD 20
20E8:E8 8A 60 A9 00 85 3D F0

```

```

298 BNE EOD ; qu'on ne sort pas des
; limites fixées par
299 INC BLOC_DEP+1 ; la commande. De toute
; façon, si on
300 EOD LDA BLOC_DEP+1 ; demande à lire plus de
; blocs qu'il n'en
301 CMP TOT_BLOC+1 ; existe, une erreur se
; produira. Il n'est
302 BEQ EOD1 ; donc possible de donner
; qu'on nombre
303 GO_LIT_BL JMP VC_PARM ; inférieur au nombre
; maximum
304
305 EOD1 LDA BLOC_DEP
306 CMP TOT_BLOC
307 BCC GO_LIT_BL
308
309 *-----*
310 * Sortie de commande
311 *-----*
312
313 CLC
314 RTS
315
316 *-----*
317 * Lecture du nombre pointé par TXTPTR
318 *-----*
319
320 RD_NBR
321 LDY #0 ; FRMNUM doit tomber sur
; un chiffre.
322 LDA (TXTPTR),Y ; S'assure que s'en est
; un, mais sans
323 CMP #'0 ; passer par CHRGET, qui
; modif. TXTPTR.
324 BCC ERR_STX ; Si on a un chiffre,
; c'est bon, car FRMNUM
325 CMP #' ; s'arrêtera à la fin du
; nombre.
326 BCC NO_ERR ; Sinon, on signale une
; erreur de syntaxe.
327
328 ERR_STX LDA #SYNT_ERR ; Signaler soi-même, car
; FRMNUM tend
329 ERR_EXIT
330 CMP #S27
331 BEQ IO_ERR
332 CMP #S28
333 BEQ IO_ERR
334 JSR BADCALL ; à déconnecter ProDOS
; lorsqu'il génère une
; PRINTERR ; Erreur.
335
336 IO_ERR JSR INVERSE
337 JMP CONT
338
339 NO_ERR JSR FRMNUM ; Si on a bien un nombre,
; on le lit, le met
340 JSR GETADR ; dans FAC puis met en
; entier dans LINNUM
341 LDA LINNUM
342 LDX LINNUM+1
343 RTS
344
345 *-----*
346 * Met TXTPTR au début de IN
347 *-----*
348
349 IN_PTR
350 LDY #<IN ; Cette fois, on se place
; sur le buffer du
351 LDX #>IN ; Clavier. Ligne de
; commande y est entière
352 STY TXTPTR ; Et stockée en
; caractères ASCII avec le
353 STX TXTPTR+1 ; Dernier bit à 1
354 RTS
355
356 *-----*
357 * CHRGET modifié
358 *-----*
359
360 CHR_GET JSR CHRGET
361 CMP #CR
362 RTS
363
364 *-----*
365 * Divers
366 *-----*
367
368 RD_BLOC JSR ENTRY
369 DFB $80
370 VC_MLI DA PARM
371 RTS
372
373 PARM DFB 3
374 DFB $60
375 BUFF HEX 0000
376 BLOC_DEP HEX 0000
377
378 COMMAND ASC 'INDEX'
379
380 TOT_BLOC HEX 0000
381 COMPTEUR HEX 00
382 TEMP HEX 0000
383
384 FIN = *
385 LONG = FIN-START+$0100
386

```

387 LST ON

Source 'T.BLOC.CODE'

Assembleur Big Mac, format TEXT

```

1 LST OFF
2
3 ORG $2100
4
5 *****
6 * INDEX: Permet de localiser tous les *
7 * blocs susceptibles d'être des *
8 * blocs index. *
9 * *
10 *****
11
12 *-----*
13 * Syntaxe: *
14 * *
15 * INDEX <total> [,S<slot>] [,D<drive>] *
16 * *
17 * Le programme va lire tous les blocs du disque *
18 * jusqu'au nombre total qu'on lui indique dans *
19 * commande. Si le bloc est susceptible d'être un *
20 * bloc index, il affiche son numéro. *
21 * *
22 *-----*
23
24 *-----*
25 * Labels
26 *-----*
27
28 SYNT_ERR = $40 ; Code du message 'SYNTAX
; ERROR'
29 PTR = $48 ; Pointeur temporaire
30 LINNUM = $50 ; Résultat de GETADR
31 STREND = $6D ; Adresse du début de la
; zone libre
32 FRETOP = $6F ; Adresse de la fin de la
; zone libre
33 OLDTXTPTR = $79 ; Sauvegarde de TXTPTR
34 CR = $8D ; Code du Carriage Return
35 BUFADR1 = $AB ; Adresse de la première
; moitié du bloc lu
36 BUFADR2 = $AD ; Adresse de la seconde
; moitié du bloc lu
37 CHRGET = $B1 ; Routine de prise de
; caractère
38 TXTPTR = $B8 ; Pointe sur le caractère
; à prendre
39
40 IN = $0200 ; Buffer clavier
41
42 PRINTERR = $BE0C ; Affichage de l'erreur
43 DEFSLT = $BE3C ; Slot par défaut
44 DEFDRV = $BE3D ; Drive par défaut
45 XTADDR = $BE50 ; Pointe sur adresse de
; la commande externe
46 XLEN = $BE52 ; Longueur du nom de la
; commande moins un
47 XCNUM = $BE53 ; Numéro de commande
48 PBITS = $BE54 ; Indiquent les
; paramètres valables
49 VPATH1 = $BE6C ; Pointe sur le nom de
; volume
50 BADCALL = $BE8B ; Traduit l'erreur pour
; le BI
51 ENTRY = $BF00 ; Point d'entrée du MLI
52 SYSERR = $BF0F ; Va à la routine de
; traitement des erreurs
53
54 FRMNUM = $DD67 ; Range valeur pointée
; par TXTPTR dans FAC
55 GETADR = $E752 ; Transforme FAC en
; entier 2 oct (LINNUM)
56 LINPTR = $ED24 ; Affiche X et A
57 COUT = $FDED ; Affiche le caractère se
; trouvant dans A
58 INVERSE = $FE80 ; Affiche en inverse
59 NORMAL = $FE84 ; Affiche en normal
60 RTS = $FF58 ; Contient un RTS
61
62 *-----*
63 * Validité de la CMD
64 *-----*
65
66 START CLD ; Obligatoire pour ProDOS
67 LDA #>FIN+$0100 ; Convention pour
; CMDLOAD
68 LDA #>LONG-$0100
69 V_OLDCMD LDA RTS
70 LDA VPATH1 ; Pointe vers commande
; pour déterminer si
; c'est la notre
71
72 STA PTR
73 LDA VPATH1+1
74 STA PTR+1
75 LDY #1
76 COMPARE LDA (PTR),Y ; C'est INDEX ?
77 CMP COMMAND-1,Y
78 BNE NO_CMD ; non, alors retour au
; ProDOS
79
80 INY
81 CPY #5+1
82 BCC COMPARE
83 BCS BONNECMD ; C'est INDEX alors on
; peut y aller

```

```

20F0:B9 38 60 80 40 20 10 08
20F8:04 02 01 00 00 00 AD 99
2100:D8 A9 23 A9 01 AD 58 FF
2108:AD 6C BE 85 48 AD 6D BE
2110:85 49 A0 00 B1 48 8D A2
2118:22 A2 00 BD 91 22 8D A3
2120:22 E8 C8 B1 48 DD 91 22
2128:D0 08 E8 C8 C0 06 90 F3
2130:B0 0F CE A3 22 F0 06 A2
2138:06 A0 01 D0 E6 38 6C 06
2140:21 A9 80 AE A3 22 CA D0
2148:02 A9 81 8D 8D 22 88 88
2150:8C 52 BE A9 00 8D 0F BF
2158:8D 53 BE 8D 54 BE A9 00
2160:8D 55 BE AD 74 21 AD 64
2168:21 8D 50 BE AD 65 21 8D
2170:51 BE 18 60 A5 B8 A6 B9
2178:85 79 86 7A AC 52 BE C8
2180:C8 98 18 6D 6C BE 85 B8
2188:AD 6D BE 69 00 85 B9 A0
2190:00 B1 B8 C9 30 90 04 C9
2198:3A 90 07 A9 40 F0 03 4C
21A0:2E 22 20 67 DD 20 52 E7
21A8:A5 50 8D A0 22 A6 51 8E
21B0:A1 22 A0 00 A2 02 84 B8
21B8:86 B9 20 8E FD 20 1F 22
21C0:C9 AC D0 F9 20 1F 22 C9
21C8:D3 D0 0B 20 1F 22 29 07
21D0:8D 3C BE 4C BD 21 C9 C4
21D8:D0 C1 20 1F 22 29 03 8D
21E0:3D BE 4C BD 21 68 68 AD
21E8:3D BE 29 02 0A 0A 0D 3C
21F0:BE 0A 0A 0A 0A 8D 9D 22
21F8:A5 79 A6 7A 85 B8 86 B9
2200:AD 9C 22 AD 01 22 AE 02
2208:22 8D 8E 22 8E 8F 22 A6
2210:6E E8 8E 9F 22 E8 E4 70
2218:90 0F A9 56 4C 2E 22 20
2220:B1 00 C9 8D F0 BF 4C ED
2228:FD 20 8A 22 90 06 20 8B
2230:BE 4C 0C BE 20 8E FD AE
2238:9E 22 86 AB AC 9F 22 84
2240:AC A6 AB A4 AC 20 99 FD
2248:A2 01 20 4A F9 A0 00 B1
2250:AB 20 DA FD A9 A0 20 ED
2258:FD C8 C0 08 D0 F1 A2 01
2260:20 4A F9 A0 00 B1 AB 29
2268:7F C9 20 B0 02 A9 2E 09
2270:80 20 ED FD C8 C0 08 D0
2278:EC 20 8E FD A5 AB 18 69
2280:08 85 AB C9 B0 90 BA 18
2288:60 00 20 00 BF 80 9C 22
2290:60 02 52 42 4C 4F 43 57
2298:42 4C 4F 43 03 60 00 10
22A0:00 00 00 00

```

À compter de ce
numéro, les
disquettes
Pom's sont
formatées...

DOS 3.3
ProDOS

82			143	BEQ	EOL	; virgule, car cela
83	NO_CMD	SEC				signifierait un numéro
						; de drive et/ou de slot
						à lire.
84		JMP (V_OLDCMD+1)	144	CMP	"#","	
85			145	BNE	CH_VRGL	
86	*	-----	146			
87	*	ProDOS étudie la ligne	147	JSR	CHR_GET	; Prend le caractère de
88	*	-----	148			commande
89			149	CMP	"#S"	; C'est le changement de
90	BONNECMD		150			slot ?
91	DEY	; Demande à ProDOS	151	BNE	TEST_D	; Non, alors peut-être
		d'examiner la commande	152			est-ce le drive ?
92	DEY	; Il ne doit pas y avoir	153	JSR	CHR_GET	Quel est le nouveau
		de paramètres	154			slot ?
93	STY	XLEN	155	AND	##00000111	
94	LDA	#0	156	STA	DEFSLT	; A mettre chez ProDOS
95	STA	SYSERR	157	JMP	CH_VRGL	
96	STA	XCNUM	158	TEST_D	CMP	"#D"
97	LDA	#0	159			; C'est le changement de
		; Pas de paramètres: \$00	160			Drive ?
		en (Pbits)	161	BEQ	NOERRS	
98	STA	PBITS	162	JMP	ERR_STX	
99	LDA	#0	163	NOERRS	JSR	CHR_GET
100	STA	PBITS+1	164			; Quel est le nouveau
101	V_SUITE	LDA SUITE	165			drive ?
		; Indique l'adresse du	166	AND	##00000011	
		retour	167	STA	DEFDRV	
102	LDA	V_SUITE+1	168	JMP	CH_VRGL	
103	STA	XTADDR	169			
104	LDA	V_SUITE+2	170			
105	STA	XTADDR+1	171			
106	CLC	; C'était bien pour nous	172	*	-----	
107	RTS		173	*	Boucle de lecture des blocs	
108	SUITE		174	*	-----	
109	CMP	#0	175			
		; Si pas d'erreur, on	176	LDA	#0	; Commence au bloc 0
		continue	177	STA	BLOC_DEP+1	
110	BNE	NO_CMD	178	STA	BLOC_DEP	
111			179			
112	*	-----	180			
113	*	Lecture du nombre de blocs	181	*	-----	
114	*	-----	182	*	Lecture du bloc indiqué	
115			183	*	-----	
116	LDA	TXTPTR	184			
		; Sauvegarde de TXTPTR	185	VC_PARM	LDA	PARM
		pour que l'exécution	186			; Nécessaire à cause du
		d'un programme Basic	187	LDA	VC_PARM+1	relogement
		puisse se continuer.	188			; Indique où est la table
117	LDX	TXTPTR+1	189			de paramètres
118	STA	OLDTXTPTR	190	LDX	VC_PARM+2	
119	STX	OLDTXTPTR+1	191	STA	VC_MLI	
120			192	STX	VC_MLI+1	
121	LDY	XLEN	193			
		; Où se trouve la fin du	194	LDA	DEFDRV	; Codage du numéro
		mot de commande ?	195			d'unité sous la forme
122	INY	; Cela sert à lire la	196	AND	##00000010	; DSSS0000
		suite: num de bloc,	197	ASL		
123	INY	; puis les paramètres	198	ASL		
		éventuels (S et D)	199	ORA	DEFSLT	
124	TYA	; Pour cela, on se sert	200	ASL		
		de TXTPTR:	201	ASL		
125	CLC	; On additionne la	202	ASL		
		longueur du mot-clé	203	ASL		
126	ADC	VPATH1	204	ASL		
		; (INDEX) au début de la	205	STA	PARM+1	
		ligne de commande.	206			
127	STA	TXTPTR	207	LDX	STREND+1	; Installe le buffer dans
		; On a alors le début de	208			la zone libre
		l'opérande.	209	INX		
128	LDA	VPATH1+1	210	CPX	FRETOP+1	
		; C'est ce qu'on met en	211	BCC	FIX_BUF	
		TXTPTR.	212			
129	ADC	#0	213	LDA	#\$56	; S'il n'y en a pas
130	STA	TXTPTR+1	214			assez, alors on sort
131			215	ERR	JMP	ERR_EXIT
132	JSR	RD_NBR	216			; 'NO BUFFER AVAIABLE'
		; On va lire le nombre	217	FIX_BUF	STX	BUFF+1
		qui est pointé	218			; Sinon, enregistre la
		; Ce sera le bloc final	219			page du buffer
133	STA	TOT_BLOC	220			; Puis va lire le bloc en
134	STX	TOT_BLOC+1	221			question
135			222			; Au cas où il y aurait
136	*	-----	223			une erreur
137	*	Lecture des paramètres s'il y en a	224	*	-----	
138	*	-----	225	*	Analyse du bloc	
139			226			
140	JSR	IN_PTR				
		; Refixe le pointeur au				
		début du buffer				
141						
142	CH_VRGL	JSR CHR_GET				
		; Prend caractère jusqu'à				
		ce que soit une				

```

207 *-----
208
209     LDY  BUFF      ; Prépare des pointeurs
                pour indiquer
210     LDX  BUFF+1    ; l'adresse de la
                première page du buffer,
211     STY  BUFADR1   ; et celle de la seconde
                page du buffer.
212     STX  BUFADR1+1
213     INX
214     STY  BUFADR2
215     STX  BUFADR2+1
216
217 *-----
218 *     Teste si      le bloc peut être un
                index
219 *-----
220
221     LDY  #0        ; Se place au début du
                bloc
222     JSR  VRFZ      ; Va vérifier s'il est
                complètement nul
223     BEQ  NXTBLK   ; Si oui, alors ça ne
                peut être un index
224     BNE  CHKNG2   ; Sinon, on fait une
                autre vérification
225
226 VRFZ     LDA  (BUFADR1),Y ; La méthode a été
                entièrement changée
227     ORA  (BUFADR2),Y ; par rapport à 'Beneath
                Apple ProDOS'
228     BNE  VRFRTS   ; Car elle donnait trop
                de blocs non-index
229     INY          ; Avec celle-ci et tous
                les essais faits
230     BNE  VRFZ     ; seul dernier bloc du
                catalog est indiqué
231 VRFRTS   RTS      ; à tort comme étant un
                index.
232
233 CHKNG2   LDY  #0        ; Le principe: tester si
                tous les numéros
234     STY  COMPTEUR ; 16 bits différents les
                uns des autres,
235     CHKNG3       ; puisque index ne peut
                contenir 2 fois le
236     LDY  COMPTEUR ; même bloc.
237     LDA  (BUFADR1),Y ; Numéro zéro indique la
                fin de la liste
238     BEQ  GOOD_ONE ; Donc si on y est, c'est
                un index
239     STA  TEMP     ; Sinon on stocke pour la
                vérification
240
241     LDA  (BUFADR2),Y
242     STA  TEMP+1
243
244 NXTBYTE  INY          ; On compare au numéro
                d'après
245     BEQ  EOI      ; Si on est à la fin du
                bloc
246     LDA  (BUFADR2),Y
247     CMP  TOT_BLOC+1 ; Numéro ne peut être
                supérieur au nbr max
248     BCC  V1
249     BNE  NXTBLK   ; Sinon, pas un bloc
                index qu'on teste
250     LDA  (BUFADR1),Y
251     CMP  TOT_BLOC
252     BCS  NXTBLK
253 V1      LDA  (BUFADR1),Y ; Si on tombe sur un
                zéro, ça signifie
254     BNE  V2      ; peut-être une fin de
                liste.
255
256     INY          ; Donc on regarde si ce
                qui suit est nul
257     JSR  VRFZ    ; Va vérifier
258     BEQ  EOI     ; Si oui, alors on a fini
                l'index
259     BNE  NXTBLK   ; Sinon, ce n'était pas
                un index
260
261 V2      CMP  TEMP    ; Ce n'était pas un zéro,
                donc on vérifie
262     BNE  NXTBYTE  ; le bas du numéro.
263
264     LDA  (BUFADR2),Y ; Si pareil en bas, il
                se peut que le
265     CMP  TEMP+1    ; haut soit différent...
266     BNE  NXTBYTE  ; Si oui, alors prochain
                octet
267     BEQ  NXTBLK   ; Si non, prochain bloc
268 EOI
269     INC  COMPTEUR ; Tous les octets étaient
                différent à celui
270     BNE  CHKNG3   ; testé, donc on prend le
                suivant.
271
272 GOOD_ONE
273     CPY  #0        ; Si 1er octet du bloc
                est un zéro, alors
274     BEQ  NXTBLK   ; ça ne peut pas être un
                index
275
276 CONT
277     JSR  PBLOCK   ; Affiche le numéro de
                bloc
278     JMP  NXTBLK   ; Passe au prochain bloc
279
280 *-----
281 *     Si le      bloc peut être index,
                afficher
282 *-----
283
284 PBLOCK   LDX  BLOC_DEP ; Affichage en décimal du
                numéro de bloc,
285     LDA  BLOC_DEP+1 ; suivi d'une virgule
                pour tous les séparer
286
287     JSR  LINPTR
288     JSR  NORMAL
289     LDA  #",
290     JMP  COUT
291
292 *-----
293 *     Passage au  prochain bloc
294 *-----
295
296 NXTBLK  INC  BLOC_DEP ; Passe au prochain bloc
                tout en vérifiant
297     BNE  EOD      ; qu'on ne sort pas des
                limites fixées par
298     INC  BLOC_DEP+1 ; la commande. De toute
                façon, si on
299     LDA  BLOC_DEP+1 ; demande à lire plus de
                blocs qu'il n'en
300     EOD   CMP  TOT_BLOC+1 ; existe, une erreur se
                produira. Il n'est
301     BEQ  EOD1    ; donc possible de donner
                qu'on nombre
302     GO_LIT_BL JMP VC_PARM ; inférieur au nombre
                maximum
303
304
305 EOD1    LDA  BLOC_DEP
306     CMP  TOT_BLOC
307     BCC  GO_LIT_BL
308
309 *-----
310 *     Sortie de  commande
311 *-----
312
313     CLC
314     RTS
315
316 *-----
317 *     Lecture du  nombre pointé par TXTPTR
318 *-----
319

```

```

320 RD_NBR
321 LDY #0 ; FRMNUM doit tomber sur
un chiffre.
322 LDA (TXTPTR),Y ; S'assure bien que s'en
est un, mais sans
323 CMP #'0 ; passer par CHRGET, qui
modif. TXTPTR.
324 BCC ERR_STX ; Si on a un chiffre,
c'est bon, car FRMNUM
325 CMP #' : ; s'arrêtera à la fin du
nombre.
326 BCC NO_ERR ; Sinon, on signale une
erreur de syntaxe.
327
328 ERR_STX LDA #SYNT_ERR ; Signaler soi-meme, car
FRMNUM tend
329 ERR_EXIT
330 CMP #27
331 BEQ IO_ERR
332 CMP #28
333 BEQ IO_ERR
334 JSR BADCALL ; à déconnecter ProDOS
lorsqu'il génère une
PRINTERR ; Erreur.
335 JMP
336 IO_ERR JSR INVERSE
337 JMP CONT
338
339 NO_ERR JSR FRMNUM ; Si on a bien un nombre,
on le lit, le met
340 JSR GETADR ; dans FAC puis le met
entier dans LINNUM
341 LDA LINNUM
342 LDX LINNUM+1
343 RTS
344
345 *-----
346 * Met TXTPTR au début de IN
347 *-----
348
349 IN_PTR
350 LDY #<IN ; Cette fois, on se place
sur le buffer du
351 LDX #>IN ; Clavier. La ligne de
commande est entière
352 STY TXTPTR ; Et stockée en
caractères ASCII avec le
353 STX TXTPTR+1 ; Dernier bit à 1
354 RTS
355
356 *-----
357 * CHRGET modifié
358 *-----
359
360 CHR_GET JSR CHRGET
361 CMP #CR
362 RTS
363
364 *-----
365 * Divers
366 *-----
367
368 RD_BLOC JSR ENTRY
369 DFB $80
370 VC_MLI DA PARM
371 RTS
372
373 PARM DFB 3
374 DFB $60
375 BUFF HEX 0000
376 BLOC_DEP HEX 0000
377
378 COMMAND ASC 'INDEX'
379
380 TOT_BLOC HEX 0000
381 COMPTEUR HEX 00
382 TEMP HEX 0000
383
384 FIN = *
385 LONG = FIN-START+$0100
386
387 LST ON
388 LST OFF
389
390 ORG $2100
391
392 *****
393 * BLOC: Permet de lire ou écrire un *
394 * block de disquette. *
395 * *
396 *****
397
398
399 *-----
400 * Syntaxe:
401 *
402 * RBLOC <numéro> [,S<slot>] [,D<drive>]
403 * WBLOC <numéro> [,S<slot>] [,D<drive>]
404 *
405 *-----
406
407 *-----
408 * Labels
409 *-----
410
411 SYNT_ERR = $40 ; Code de 'SYNTAX ERROR'
412 PTR = $48 ; Pointeur temporaire
413 LINNUM = $50 ; FAC devenu entier dans
deux octets
414 NO_BUFF = $56 ; Code de 'NO BUFFER
AVAILABLE'
415 STREND = $6D ; Page où commence la
zone libre
416 FRETOP = $6F ; Page où se termine la
zone libre
417 OLDTXTPTR = $79 ; Sauvegarde de TXTPTR
418 READ_BLOCK = $80 ; Code de l'instruction
MLI
419 WRITE_BLOCK = $81 ; Code de l'instruction
MLI
420 CR = $8D ; Code de Retour à la
ligne
421 BUFADR = $AB ; Pointe sur le début du
buffer
422 CHRGET = $B1 ; Routine de prise de
caractère
423 TXTPTR = $B8 ; Pointeur utilisé par
FRMNUM
424
425 IN = $0200 ; Buffer clavier
426
427 MLI = $BF00 ; Point d'entrée du MLI
428 PRINTERR = $BE0C ; Affiche l'erreur, met
dans A son code
429 XTADDR = $BE50 ; Adresse de la routine
externe
430 XLEN = $BE52 ; Longueur du nom de la
commande moins un
431 XCNUM = $BE53 ; Numéro de commande ($00
= externe)
432 PBITS = $BE54 ; Bits autorisant
l'utilisation
d'opérandes
433 DEFSLT = $BE3C ; Slot par défaut
434 DEFDRV = $BE3D ; Drive par défaut
435 VPATH1 = $BE6C ; Adresse de la ligne de
commande
436 BADCALL = $BE8B ; Code d'erreur en numéro
d'erreur BI
437 SYSERR = $BF0F ; Numéro d'erreur systeme
438
439 FRMNUM = $DD67 ; Evalue la formule
pointée par TXTPTR
440 GETADR = $E752 ; FAC -> LINNUM
441 PRBL2 = $F94A ; Affiche X espaces
442 PRBYTE = $FDDA ; Affiche l'accumulateur
en hexa
443 CROUT = $FD8E ; Retour à la ligne
444 PRNTXY3 = $FD99 ; Affiche X et Y en hexa,
suivis de "-"
445 COUT = $FDED ; Affiche le caractère

```

446	RTS	=	\$FF58	contenu dans A	504	*	Va	analyser	la commande
447				; Contient un RTS	505	*	-----		
448	*	-----			506				
449	*	Validité de		la CMD	507		DEY		; On passe par
450	*	-----							l'analyseur de ProDOS,
451									car
452	START	CLD		; Obligatoire pour ProDOS	508		DEY		; L'autre méthode ne
453		LDA	#>FIN+\$0100	; Convention pour	509		STY	XLEN	fonctionne pas.
				CMDLOAD					; Comme on analyse les
454		LDA	#>LONG-\$0100		510		LDA	#0	paramètres, ce
455	V_OLDCMD	LDA	RTS						; n'était pas utile de le
456		LDA	VPATH1	; Pointe vers commande	511		STA	SYSERR	faire.
				pour déterminer si	512		STA	XCNUM	
457		STA	PTR	; c'est la notre	513		STA	PBITS	
458		LDA	VPATH1+1		514		LDA	#0	
459		STA	PTR+1		515		STA	PBITS+1	
460					516	V_SUITE	LDA	SUITE	
461		LDY	#0	; Cela indique longueur	517		LDA	V_SUITE+1	
				de la ligne tapée	518		STA	XTADDR	
462		LDA	(PTR),Y		519		LDA	V_SUITE+2	
463		STA	CMD_LEN		520		STA	XTADDR+1	
464					521		CLC		
465		LDX	#0	; Cela indique nombre de	522		RTS		
				commande possibles	523				
466		LDA	COMMAND,X	; que BLOC peut	524	*	-----		
				comprendre et traiter.	525	*	Lecture du		numéro de bloc
467		STA	NBR_CMD	; (en fait: 2)	526	*	-----		
468					527				
469		INX			528	SUITE			
470		INY			529		LDA	TXTPTR	; Sauvegarde de TXTPTR
471	COMPAR	LDA	(PTR),Y	; C'est RBLOC ou WBLOC ?					pour que l'execution
472		CMP	COMMAND,X		530		LDX	TXTPTR+1	; d'un programme Basic
473		BNE	NO_CMD	; non, alors retour au					puisse se continuer.
				ProDOS	531		STA	OLDTXTPTR	
474		INX		; Oui, prochain caractère	532		STX	OLDTXTPTR+1	
				de chacune des	533				
475		INY		; commande: tapée et	534		LDY	XLEN	; Où se trouve la fin du
				possible.					mot de commande ?
476		CPY	#5+1	; Commande ne possède que	535		INY		; Cela sert à lire la
				5 caractères					suite: nu de bloc,
477		BCC	COMPAR	; Continue la boucle de	536		INY		; puis les paramètres
				comparaison					éventuels (S et D)
478		BCS	BONNECMD	; C'est la bonne	537		TYA		; Pour cela, on se sert
				commande...					de TXTPTR:
479					538		CLC		; On additionne la
480	NO_CMD								longueur du mot-clé
481		DEC	NBR_CMD	; Y-a-t-il une autre	539		ADC	VPATH1	; (RBLOC ou WBLOC) au
				commande possible ?					début de la ligne de
482		BEQ	NO_CMD1	; Si Z=1 alors non, on	540		STA	TXTPTR	; commande. On a alors
				sort.					début opérande.
483		LDX	#6	; Sinon, oui, alors on	541		LDA	VPATH1+1	; C'est ce qu'on met en
				voit l'autre,					TXTPTR.
484		LDY	#1	; et on redémarre sur la	542		ADC	#0	
				commande tapée	543		STA	TXTPTR+1	
485		BNE	COMPAR	; Puis on les compare	544				
486					545		LDY	#0	; FRMNUM doit tomber sur
487	NO_CMD1								un chiffre.
488		SEC		; Indique que la commande	546		LDA	(TXTPTR),Y	; S'assure que s'en est
				n'est pas à nous					un, mais sans
489		JMP	(V_OLDCMD+1)		547		CMP	#'0	; passer par CHRGET, qui
490									modif TXTPTR.
491	*	-----			548		BCC	ERR_STX	; Si on a un chiffre,
492	*	Détermine si		on écrit ou si on lit					c'est bon, car FRMNUM
493	*	-----			549		CMP	#':	; s'arrêtera à la fin du
494									nombre.
495	BONNECMD				550		BCC	NO_ERR	; Sinon, on signale une
496		LDA	#READ_BLOCK	; On décide					erreur de syntaxe.
				arbitrairement de lire	551	ERR_STX	LDA	#SYNT_ERR	; Il faut la signaler,
497		LDX	NBR_CMD	; Quelle commande a-t-on					car FRMNUM tend
				lu ?	552		BEQ	NO_ERR	; à déconnecter ProDOS
498		DEX		; La 1ère ou la 2nde ?					lorsqu'il génère une
499		BNE	MET_CMD	; Z=0 signifie la 1ère,	553		JMP	ERR_EXIT	; Erreur.
				donc RBLOC	554				
500		LDA	#WRITE_BLOCK	; Sinon, c'est WBLOCK	555	NO_ERR	JSR	FRMNUM	; Si on a bien un nombre,
501	MET_CMD	STA	GOTO_MLI+3	; On stocke ça dans la					on le lit, le met
				table des paramètres	556		JSR	GETADR	; dans FAC puis le met
502									entier dans LINNUM
503	*	-----			557				
					558		LDA	LINNUM	; LINNUM est numéro du

```

559      STA  BUFF+2      bloc sur lequel
                    ; On va maintenant
560      LDX  LINNUM+1    ; travailler.
                    ; Donc on le met dans
                    table de paramètres
561      STX  BUFF+3
562
563 *-----
564 *      Lecture des      paramètres s'il y en a
565 *-----
566
567      LDY  #<IN        ; Cette fois, on se place
                    ; sur le buffer du
568      LDX  #>IN        ; Clavier. Ligne de
                    ; commande y est entière
569      STY  TXTPTR      ; Et stockée en
                    ; caractères ASCII avec le
570      STX  TXTPTR+1    ; Dernier bit à 1
571
572      JSR  CROUT        ; Revient à la ligne
573 CH_VRGL JSR  CHR_GET   ; Scanne toute ligne à la
                    ; recherche d'une
574      CMP  #", "       ; Virgule, pour voir s'il
                    ; y a un paramètre.
575      BNE  CH_VRGL
576
577      JSR  CHR_GET      ; Prend le caractère de
                    ; commande
578      CMP  #"S"        ; C'est le changement de
                    ; Slot ?
579      BNE  TEST_D      ; Non, alors peut-être
                    ; est-ce le drive ?
580      JSR  CHR_GET      ; Quel est le nouveau
                    ; Slot ?
581      AND  #%00001111  ; On le transforme en
                    ; chiffre hexa
582      STA  DEFSLT      ; A mettre chez ProDOS
583      JMP  CH_VRGL
584
585 TEST_D  CMP  #"D"      ; C'est le changement de
                    ; Drive ?
586      BNE  ERR_STX
587      JSR  CHR_GET      ; Quel est le nouveau
                    ; drive ?
588      AND  #%00000011  ; Transformé en chiffre
                    ; hexa
589      STA  DEFDRV
590      JMP  CH_VRGL
591
592 EOL
593      PLA          ; Pour effacer le RTS de
                    ; CHR_GET
594      PLA
595
596 *-----
597 *      Modification du numéro d'unité
598 *-----
599
600      LDA  DEFDRV      ; On prend le numéro de
                    ; drive
601      AND  #%00000010  ; Dont on ne garde que le
                    ; 2ème octet
602      ASL          ; Que l'on décale 2 fois
                    ; à gauche
603      ASL          ; Pour obtenir ceci:
                    ; 0000D000
604      ORA  DEFSLT      ; Puis on y met aussi les
                    ; bits du slot
605      ASL          ; On a alors ceci:
                    ; 0000DSSS
606      ASL          ; Puis on décale le tout
                    ; 4 fois à droite
607      ASL          ; Pour avoir ceci:
                    ; DSSS0000
608      ASL
609      STA  RWBLP+1     ; C'est le numéro d'unité
610
611      LDA  OLDTXTPTR   ; Et on récupère le
                    ; TXTPTR du programme
612      LDX  OLDTXTPTR+1 ; Pour que celui-ci
                    ; puisse reprendre son
                    ; Execution normalement.
613      STA  TXTPTR
614      STX  TXTPTR+1
615
616 *-----
617 *      Lance les      opérations
618 *-----
619
620 PARM_AD
621      LDA  RWBLP      ; Ceci est nécessaire
                    ; étant donné que la
622      LDA  PARM_AD+1  ; Routine est relogée.
623      LDX  PARM_AD+2
624      STA  PARM_P
625      STX  PARM_P+1
626
627 *-----
628 *      Cherche un      buffer valable
629 *-----
630
631      LDX  STREND+1    ; On va se placer dans
                    ; espace libre, entre
632      INX          ; Variables dimensionnées
                    ; et chaînes de
633      STX  BUFF+1     ; Caractères.
634      INX
635      CPX  FRETOP+1
636      BCC  RW_BLOC
637
638      LDA  #NO_BUFF   ; S'il n'y a pas deux
                    ; pages minimum de
639      JMP  ERR_EXIT   ; Disponibles, alors 'NO
                    ; BUFFER AVAILABLE'
640
641 *-----
642 *      CHRGET amélioré
643 *-----
644
645 CHR_GET JSR  CHRGET   ; Prend le caractère
                    ; suivant,
646      CMP  #CR        ; Si Retour Chariot la
                    ; ligne est finie
647      BEQ  EOL        ; Oui, alors End Of Line
648      JMP  COUT       ; Non, alors on affiche,
                    ; puis on retourne
649
650 *-----
651 *      Va lire      ou écrire le disque
652 *-----
653
654 RW_BLOC
655      JSR  GOTO_MLI   ; Demande à exécuter la
                    ; commande tapée
656      BCC  EXIT       ; Si C=1, erreur, sinon
                    ; ça va
657
658 ERR_EXIT
659      JSR  BADCALL    ; On prépare le code
                    ; d'erreur
660      JMP  PRINTERR   ; Puis on l'affiche...
661
662 *-----
663 * Affichage de controle (176 octets)
664 *-----
665
666 EXIT
667      JSR  CROUT      ; Saute à la ligne après
                    ; rappelé le bloc
668
669      LDX  BUFF        ; Transfère l'adresse du
                    ; buffer au pointeur
670      STX  BUFADR
671      LDY  BUFF+1
672      STY  BUFADR+1
673
674 AFF_CTRL LDX  BUFADR  ; Affiche adresse du
                    ; début de la ligne qui
675      LDY  BUFADR+1   ; suit.
676      JSR  PRNTXY3
677      LDX  #1        ; Puis fait un espace

```

678	JSR PRBL2								pour un control
679				711	BCC	AFF_CTRL			
680	LDY #0		; On se place au début de la ligne	712					
681	AFF_CODE LDA (BUFADR),Y		; Puis on lit les octets un par un	713	*	-----			
682	JSR PRBYTE			714	*	Sortie de		la commande	
683	LDA #" "		; Met un espace entre chaque code	715	*	-----			
684	JSR COUT			716					
685	INY		; Passe au prochain octet	717	CLC			; Il n'y a pas eu de problème	
686	CPY #8		; On en fait 8 par ligne pour ne pas être désorienté.	718	RTS				
687	BNE AFF_CODE			719					
688				720	*	-----			
689	LDX #1		; Met 1 esp entre codes et caractères	721	*	Divers			
690	JSR PRBL2			722	*	-----			
691				723					
692	LDY #0		; Revient en début de ligne pour afficher correspondants.	724	BRK				
693	AFF_CAR LDA (BUFADR),Y		; tous les caractères correspondants.	725	GOTO_MLI				
694	AND #%01111111		; On filtre afin de pas avoir de caractères	726	JSR MLI			; Puis appel à celle-ci via le MLI	
695	CMP #\$20		; gêne affichage parcequ'interprétés par	727	DFB \$80			; Code de la commande	
696	BCS NORM		; la carte 80 colonnes (entre autre)	728	PARM_P DA RWBLP			; Adresse de la table des paramètres.	
697	LDA #'			729	RTS				
698	NORM ORA #*10000000			730					
699	JSR COUT			731	COMMAND DFB 02			; Il y a deux commandes possibles	
700	INY		; Prochain caractère de la série	732	ASC 'RBLOC'			; Liste des commandes disponibles	
701	CPY #8		; On en a mis 8	733	ASC 'WBLOC'				
702	BNE AFF_CAR			734					
703				735	RWBLP DFB \$03			; Il y a trois paramètres	
704	JSR CROUT		; Passe à la ligne	736	DFB \$60			; Numéro d'unité (\$E0 pour S6,D2)	
705	LDA BUFADR		; Puis change le pointeur pour avoir une nouvelle série de 8 octets	737	BUFF DA \$1000			; Adresse du buffer	
706				738	DA \$0000			; Numéro du bloc	
707	CLC			739					
708	ADC #8			740	CMD_LEN HEX 00			; Longueur de la ligne de commande	
709	STA BUFADR			741	NBR_CMD HEX 00			; Compteur pour savoir quelle commande	
710	CMP #\$B0		; 176 octets suffisent	742					
				743	FIN = *				
				744	LONG = FIN-START+\$0100				
				745					
				746	LST ON				

Éditeur Plein Écran **EPE 5.1**

Apple IIe, IIe+, IIc, **IIGS**
DOS 3.3 & ProDOS

Le Pacha

Listez vos programmes Basic en avant et en arrière.
Modifiez, insérez, effacez des caractères en plein écran
sans relire les lignes.

Recherchez toute chaîne de caractères.
Choisissez vous-même les codes de contrôle d'EPE.
Modifiez E.P.E. : le fichier source est sur la disquette.

200,00 F TTC franco (bon de commande page 74).

Échange contre version 2.2 ou 5.0 : 80,00 F

On trouve plus facilement avec 10 ans

Certaines évolutions sont aussi importantes que des révolutions.

Le plus difficile quand on débute, c'est de débiter. Car toutes les propositions d'emploi demandent une certaine expérience. Chercher un emploi est donc un problème insoluble. Pas pour l'Apple IIgs, il débute avec 10 ans d'expérience.

Comment? C'est très simple, il a une architecture double : un nouvel ordinateur puissant traitant le graphique et le son, entoure un Apple II classique et miniaturisé. Il profite ainsi de 10 ans de développement de logiciels et d'expérience. Un grand progrès réalisé en partie grâce au microprocesseur 65C816, un 16 bits descendant du 65C02 de l'Apple II. L'Apple IIgs fonctionne à deux vitesses : 2,8 MHz en mode natif ; en mode émulation (c'est-à-dire lorsque vous utilisez un

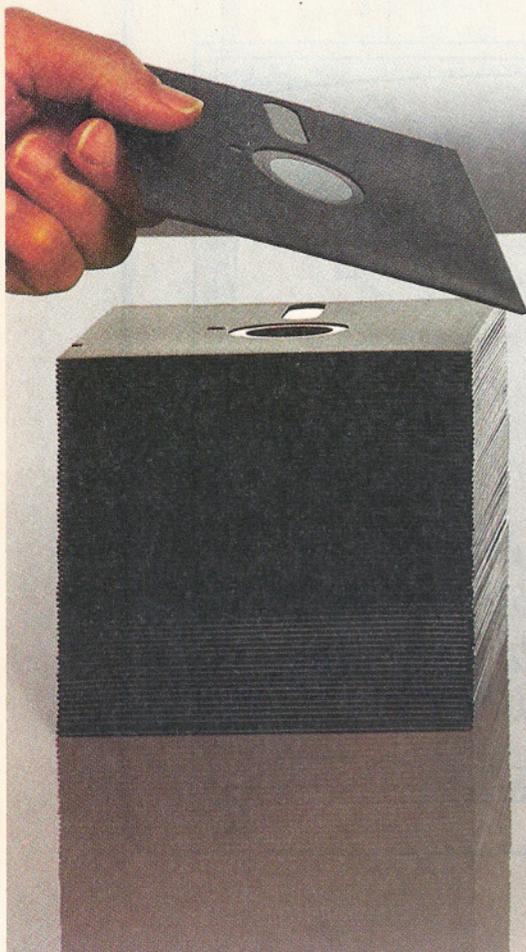
programme de votre ancien Apple II) vous avez le choix entre 1 MHz et 2,8 MHz.

Débiter connu, c'est 16.000 propositions d'emploi.

Avec l'IWM (Integrated Woz Machine*) l'Apple IIgs peut recevoir indifféremment des lecteurs de disquettes 3,5 pouces 800 Ko ou des lecteurs 5,25 pouces 140 Ko, ou les faire cohabiter, ce qui contribue à vous faire profiter de la plus grande bibliothèque de logiciels du monde. Le nombre de 16.000 logiciels étant estimatif car la seule chose qu'un Apple IIgs soit incapable de calculer, c'est le nombre d'emplois qu'il est capable de tenir.

Avec le nouveau système d'exploitation ProDOS, l'Apple IIgs permet la hiérarchisation des fichiers et des catalogues à la manière de

* En hommage à Steve Wozniak, co-fondateur de la Société Apple et créateur de l'Apple II.



un emploi quand on débute d'expérience.

Macintosh, et la connexion de nouveaux périphériques.

Gets, cela veut dire graphique et son.

L'imagination débridée des développeurs suffira-t-elle pour utiliser les 4096 nuances de couleurs disponibles ? C'est probable. Ce qui est certain, c'est qu'une résolution de 640 points sur 200 lignes pour 4 couleurs, 320 points sur 200 lignes pour 16 couleurs va donner beaucoup de relief à certaines démonstrations. Si vous n'en croyez pas vos yeux, vous n'en croirez pas vos oreilles non plus. Le coprocesseur "son" choisi par Apple a déjà une brillante carrière derrière lui puisqu'il est employé par les plus grands synthétiseurs du marché. C'est l'ENSONIQ, capable de générer 16 voix,

et il ne lui manque même pas la parole.

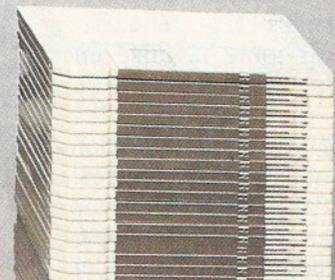
Les relations nécessaires pour réussir.

En 10 années, l'Apple II a eu tout le loisir de se faire d'excellentes relations de travail, imprimante, modem, disque dur, que l'Apple IIgs continuera à entretenir et développer ; celui-ci peut désormais être connecté au réseau AppleTalk et gérer la LaserWriter réservée jusqu'à présent à Macintosh. Il peut aussi recevoir le disque dur SCSI.

L'Apple IIgs possède 7 connecteurs d'extension permettant, à l'aide d'une multitude de cartes d'interfaces, de le relier à tous les types d'unités périphériques.

Acheter un Apple, c'est entrer dans le Club Apple pour échanger des informations, accéder au support technique par téléphone 7 jours sur 7, ou aux services télématiques du Club.

Apple présente l'Apple IIgs.



Apple

Apple, le logo Apple, Apple IIgs, ProDOS, Macintosh, Apple Talk, LaserWriter et GS Paint sont des marques déposées par Apple Computer Inc.

PageMaker : essai 'routier'



Philippe Mathieu

PageMaker, réalisé par Aldus et distribué par Ise Cegos, est le logiciel qui fit le succès de l'édition électronique sur Macintosh. Il se présente sous la forme de deux disquettes (Démarrage et Programme) accompagnées d'une documentation et d'une disquette de démonstration 'visite guidée'.

Il peut être utilisé sur un Macintosh 512Ko mais le travail sera plus aisé avec un Mac Plus. On peut l'installer sans difficulté sur un disque dur, l'original devant être introduit momentanément à la mise en route.

Démarrer avec PageMaker

Au démarrage, le menu Fichier permet soit d'ouvrir une publication (nom donné aux documents créés par PageMaker) existante, que l'on retrouve dans l'état où on l'avait laissée, soit d'ouvrir une nouvelle publication. Dans ce cas, une fenêtre de dialogue (Format d'impression) apparaît. Elle permet de déterminer le format de papier utilisé, l'orientation (paysage, portrait), l'option recto/verso, le nombre de pages ainsi que les marges.

Les format et orientation sont fixés définitivement. Les marges peuvent être modifiées ultérieurement mais cela oblige généralement à revoir toute la mise en page. En revanche, il est facile de supprimer ou d'ajouter des pages, sachant qu'une publication ne peut pas dépasser seize pages. Sur un 512Ko il sera même préférable, en raison de la capacité des disquettes, de ne pas aller au-delà de huit pages. Pour un document plus long, il suffira

d'enchaîner plusieurs publications, la pagination pouvant aller jusqu'à 999.

L'éditeur de texte interne

PageMaker offre des possibilités d'édition de texte mais, visible-ment, cet éditeur interne n'a pas été conçu pour être l'atout majeur du logiciel ! Il permet la création de textes autonomes ; cependant attention, il lui arrive de temps en temps (surtout avec la version 1.0) de ne pas respecter les marges et colonnes. On peut également l'utiliser pour modifier (insertion, suppression, etc.) des documents provenant de logiciels extérieurs mais, là encore, prudence : les résultats sont parfois aléatoires en ce qui concerne la mise en page.

Il est préférable de réserver l'usage de l'éditeur à l'ajout de titres ou de légendes, et de passer par un traitement de texte classique pour créer des textes longs. D'ailleurs, PageMaker ne prétend pas au titre de traitement de texte.

Un menu déroulant (Typo) permet de choisir le style des caractères (standard, gras, italique...), d'inverser leur 'couleur' et de déterminer l'alignement et la justification. Des raccourcis existent qui accélèrent certains de ces choix. On peut, à travers une fenêtre de dialogue ouverte par la commande *Caractères*, choisir la police de caractères, la taille, la position (indice, exposant) et l'interlignage.

Il est également possible par ce menu de déterminer la tabulation (commande *Tabulation*). Une règle apparaît alors à l'écran,

comportant un repère d'indentation ainsi que des taquets de tabulation gauche, droite, centrée ou décimale (cherchez bien ces taquets, ils sont cachés sous le repère d'indentation !).

On peut regretter l'absence de certaines fonctions classiques de traitement de texte : recherche/remplacement, glossaire. En revanche, PageMaker propose un trait d'union 'aléatoire' permettant une césure automatique des mots en cas de besoin (colonnes trop étroites).

Les outils semi-graphiques

Une trousse à outils, toujours présente à l'écran, permet d'effectuer toutes sortes de tracés (lignes, rectangles, carrés, cercles, ovales), la sélection du type de trait et de fond se faisant par deux menus déroulants. Les commandes *Premier plan* et *Second plan* du menu Édition (analogues à celles de MacDraw) se montrent très pratiques pour la réalisation d'encadrements variés.

Les graphiques et dessins placés sur une page peuvent être modifiés. Il faut pour cela les sélectionner à l'aide de la souris. Des poignées apparaissent alors, qui permettent de les agrandir ou de les réduire avec le pointeur en forme de flèche. Attention, ce genre de modifications, en déformant les caractères, rend parfois les légendes illisibles.

On trouve aussi dans la trousse des ciseaux permettant de couper des graphiques ou dessins. Ces "découpages" se font sans perte d'information car il est possible, toujours avec les ciseaux, de

revenir à l'état initial de l'image.

Placer un document

L'éditeur de texte et les outils sont des accessoires. PageMaker est essentiellement conçu pour mettre en page des documents créés avec d'autres logiciels et c'est bien sur ce point qu'il se montre le plus performant.

La mise en place de ces documents se fait soit par la commande *Placer* du menu Fichier, soit en passant par le Presse-papiers. L'utilisation de la commande *Placer* ouvre une fenêtre de dialogue permettant de choisir le document désiré, ce document devant se trouver sur une des disquettes en place ou sur le disque dur. Le choix d'un document fait apparaître un pointeur dont la forme varie selon le type du document. Il ne reste plus qu'à le placer à l'endroit voulu.

Les documents provenant d'un logiciel de traitement de texte (MacWrite, Word...) peuvent être placés directement en conservant la mise en page originale. Il est également possible de modifier cette mise en page par la création de colonnes et en utilisant la règle de tabulation. Nous reparlerons de l'utilisation de ces colonnes qui constituent un des attraits de PageMaker.

Pour un tableau, les choses se passent un peu moins simplement, le document voyant le plus souvent sa tabulation perturbée. Si votre tableur possède une fonction 'Copie de l'image' (comme Excel), aucun problème, le tableau est conservé dans le Presse-papiers avec ses tabulations. Si ce n'est pas le cas (comme, par exemple, avec Multiplan) le tableau se trouvera également dans le Presse-papiers, mais les tabulations seront perdues. Enfin, si votre tableur permet l'enregistrement sous forme 'texte', vous pourrez utiliser la commande *Placer* mais, là encore, vous devrez revoir la

tabulation.

Les dessins (type MacPaint ou MacDraw) sont également accessibles par la commande *Placer* ou à partir du Presse-papiers (pour MacDraw le document doit être enregistré en format 'Picture'). Ces dessins sont aisément modifiables : agrandissement, réduction, découpage grâce aux outils de la trousse.

La mise en page

Au démarrage, après avoir choisi le format d'impression, la première page s'affiche en taille écran. Ce format donne une vue d'ensemble de la page mais les textes sont illisibles. D'autres formats existent : taille réelle, réduction à 50% ou 70% et agrandissement à 200%. On trouve au bas de l'écran les icônes de chaque page ainsi que les icônes de la maquette.

La réalisation d'une page à l'aide de PageMaker se déroule en trois étapes : composition de la page en disposant divers repères, placement des documents extérieurs, finition à l'aide des outils de la trousse et de l'éditeur de texte.

La composition de la page se fait en général en taille écran (la page est alors entièrement visible). On peut demander l'affichage de règles horizontales et verticales qui permettent de placer des repères (en tirant avec la souris à partir des règles). Ces repères sont sans effet sur la disposition des éléments de la page, mais se montrent très commodes pour le travail, la commande *Magnétisme des repères* collant automatiquement contre le repère tout document placé à proximité. Le point d'intersection des règles, point zéro des graduations, peut être placé en n'importe quel endroit de la page, ce qui aidera au centrage de certains documents.

Il est également possible de placer des repères "actifs" : les colonnes, dont on choisira le nombre et l'espacement par la commande *Disposition* des colonnes. De plus, leur position et leur largeur pourront être modifiées par la

souris. En utilisant cette option, tout texte placé sur la page perdra sa tabulation et sa justification originale pour se dérouler à l'intérieur de la colonne, avec la mise en page que vous aurez choisie. Un texte trop long pour une colonne pourra être poursuivi dans la suivante, sans difficulté, à l'aide du pointeur.

Un texte qui vient d'être placé sur une page est entouré de poignées de sélection qui permettent, en tirant avec la souris, de le découper en pavés que l'on pourra déplacer indépendamment les uns des autres sans que le texte perde sa continuité (si l'on insère du texte dans un pavé, l'excédent est transféré dans le pavé suivant). De la même manière, il est possible de regrouper plusieurs pavés en un. Pour faire apparaître ces poignées il suffit de sélectionner le pavé avec la souris.

Le placement de dessins ou graphiques n'est pas affecté par les colonnes, mais l'utilisation du *Magnétisme des repères* permet des alignements parfaits. Il est possible d'intégrer ces images au texte soit en déplaçant des pavés, soit en modifiant la disposition des colonnes.

Ces opérations de placement des documents s'effectuent de préférence en taille réelle, la taille écran ne donnant pas une vision exacte du résultat. Évidemment, en taille réelle, la page n'apparaît pas totalement (à moins de disposer d'un écran pleine page) mais on peut se déplacer à l'aide des ascenseurs habituels du Macintosh.

En taille écran, on voit que la page n'occupe pas toute la place disponible. Cet espace libre est la table de montage (également utilisable en taille réelle). On effectuera sur cette table le bricolage : modification des documents, construction de certains motifs graphiques... En effet, le travail sur un élément d'une page se fait avec la souris, le risque étant alors de déplacer un élément correctement placé lorsqu'on veut en modifier un autre. Le travail sur la table de montage permettra d'éviter ces erreurs de manipulation, en particulier dans le cas

de pages complexes comportant un grand nombre d'éléments.

La maquette

Si certains éléments doivent se trouver sur toutes les pages d'une publication : repères, encadrements, logos, pagination... il suffit de créer une maquette, accessible par l'icône *recto* (ou deux icônes *recto* et *verso*, selon le choix du format d'impression). Toutes les fonctions de PageMaker sont disponibles sur ces maquettes.

Les éléments placés sur la maquette se retrouveront sur toutes les pages de la publication, mais il est possible de les faire disparaître de certaines pages par la commande *Supprimer* les motifs de la maquette.

Dans le cas d'une publication *recto/verso*, on définit deux maquettes qui peuvent être différentes. Il est possible, pour les comparer, de demander l'affichage de la double page. Cependant, on ne pourra pas travailler dessus, tous les outils et commandes étant alors inactivés. Cette commande *Afficher la double page* est utilisable pour comparer deux pages se faisant face dans la publication.

Le repère de foliotage peut se placer en tout point de la maquette et il est possible, dans le cas d'une publication *recto/verso*, de ne le placer que sur une des deux maquettes. La numérotation ne peut se faire qu'en chiffres arabes.

La manipulation des pages

On a vu qu'il est facile de supprimer une page ou d'en insérer une nouvelle (toujours dans la limite de seize). Toutefois, dans le cas d'une publication *recto/verso*, attention à la parité ! En effet si les maquettes des pages paires et impaires sont différentes, la mise en page peut être perturbée.

Il est possible d'invertir des

pages ou de recopier une page sur une autre. Il suffit de sélectionner toute la page (en taille écran) puis de la coller sur une autre page.

L'impression

PageMaker admet de nombreuses imprimantes (ImageWriter, Laser Writer ou toute imprimante ou photocomposeuse acceptant le langage PostScript) mais l'impression n'est pas très rapide. C'est bien sûr avec une imprimante à laser que l'on obtiendra les meilleurs résultats.

Avec la LaserWriter, différentes options sont disponibles : le lissage, qui améliore la qualité d'impression des textes et graphiques (soyez patient, l'impression est plus lente), des réductions ou agrandissements allant de 15% à 1000%, le montage en taille réduite de toutes les pages sur une même feuille, permettant une vue d'ensemble de la publication. Le résultat obtenu sur le papier est identique à ce que vous voyez à l'écran en taille réelle (What You See Is What You Get, le *WYSIWIG* à la mode !) mais, bien entendu, les différents repères et règles n'apparaissent pas. On observe cependant quelques différences entre l'écran et l'impression laser, particulièrement pour les styles autres que standard ainsi que pour les mélanges de textes avec des dessins de type MacPaint.

Apprendre à utiliser PageMaker

PageMaker est accompagné d'une disquette 'visite guidée' qui permet un premier contact avec le logiciel. La documentation commence également par une visite guidée suivie de l'exposé des différentes fonctions. Elle est claire et bien traduite, nous n'y avons pas décelé d'erreur. Elle est complétée par un glossaire et un index.

Le logiciel dispose en outre d'une fonction *assistance* facilement utilisable et suffisamment explicite.

Avantages et faiblesses

Au rang des faiblesses, nous placerons en premier lieu l'éditeur de texte aux fonctions limitées. À un moment où les logiciels de traitement de texte offrent de plus en plus de possibilités de mise en page, un logiciel d'édition électronique devrait permettre un traitement aisé des textes.

On regrettera également de ne pas pouvoir concevoir plusieurs maquettes pour une même publication ainsi que l'impossibilité de travailler sur plusieurs pages simultanément. La limitation à seize pages d'une publication est elle aussi assez gênante.

Parmi les points forts de PageMaker, on comptera tout d'abord sa grande simplicité d'utilisation lors du placement de documents externes. D'une manière générale, les différentes options s'utilisent commodément : création des maquettes, réduction ou agrandissement des pages, disposition des colonnes et repères dont le magnétisme est particulièrement efficace.

Autre avantage : le découpage des textes en pavés qui offre, avec un peu d'habitude, de nombreuses possibilités.

Un bon point aussi pour la trousse, toujours accessible, et dont les outils semi-graphiques sont intéressants.

Conclusion

PageMaker est tout à fait satisfaisant dès lors qu'il s'agit de regrouper des documents de provenances variées en leur ajoutant des encadrements, traits, etc. l'éditeur de texte n'étant utilisé que pour écrire titres et légendes ou modifier légèrement les textes.

Finalement, c'est quand même vous qui ferez qu'une page sera belle et agréable à lire !



Les nouveau-nés : Mac SE & Mac II

À l'issue de la récente annonce des nouveaux venus de la gamme Apple, nous vous proposons ici une présentation qui ne se laissera pas entraîner par notre enthousiasme face à un Mac en couleur. Regrettons seulement qu'il faille nécessairement, dans une telle approche, aborder la question du prix qui refroidit quelque peu.

Présentation

Mac SE : c'est quasiment un Macintosh Plus avec une légère évolution du clavier. Le coloris est platine, celle de l'Apple IIGs, mais aussi celle des Mac Plus dès ce mois-ci.

Mac II : l'unité centrale est plus volumineuse – on pense à un IBM – moniteur type IIGs, clavier détachable avec touches de fonctions et touches spéciales (pour MS/DOS ou UNIX). Il arbore un aspect 'professionnel' (plus que le Mac ?) que certains apprécient.

Ouvert, Fermé ?

Si le Mac n'est pas si fermé qu'on a bien voulu le dire (ou le médire), les nouveaux Mac sont

résolument ouverts. Il ne s'agit pas d'une ouverture 'à l'Apple II' (qui conduit, selon l'usage, à ne jamais reposer le capot !) mais d'une facilité d'évolution de la machine.

Mac SE : un seul connecteur – mais 96 broches – sur la carte mère pour installer des cartes d'extensions : adaptation au MS/DOS, carte vidéo, accélérateur, modem intégré... 6 périphériques SCSI peuvent être connectés en plus de l'éventuel disque dur interne.

Mac II : 6 connecteurs d'extension dont l'un occupé par la carte vidéo. 6 périphériques peuvent également être chaînés sur le port SCSI en plus du disque dur. La transmission des données sur les connecteurs utilise le protocole NuBus qui supprime le problème de la configuration manuelle. Une version du système d'exploitation UNIX, appelé A/UX, sera proposée à la fin de l'année. On parle de facilité de communication (Ethernet, Appletalk) sous A/UX.

Mac SE et Mac II : avec un lecteur 5,25 pouces, il sera possible de lire des fichiers MS/DOS sans carte spéciale grâce à un soft Apple : InterFile.

L'ouverture, c'est aussi deux ports série, un port lecteur de disquettes externe, une interface d'entrée (clavier, souris...) pouvant gérer 16 périphériques et un port son stéréo 4 voix.



Stockage

Mac SE : ce modèle sera proposé en deux versions, deux lecteurs de disquettes 800 Ko, ou un seul mais avec un disque dur de 20 Mo SCSI.

Mac II : il sera équipé d'un ou deux lecteurs de disquettes 800 Ko et d'un disque dur de 20, 40 ou 80 Mo.

La vidéo

Mac SE : monochrome.

Mac II : 640 x 480 points. La carte pilote un moniteur vidéo très haute résolution de 12 pouces monochrome ou 13 pouces en couleur. En standard, 16 couleurs ou nuances de gris parmi 16 millions et, avec extension mémoire, 256 couleurs ou nuances de gris ; cela ne laisse pas indifférent. Pour une présentation sophistiquée à l'écran, les programmeurs souffriront certainement... Les programmes tournant actuellement sur le Mac ne sont pas tous une démonstration de bon goût et de figinage alors, avec l'arrivée de la couleur et 307 200 points...

Le processeur

Mac SE : comme le Mac : MC68000 Motorola (32/16 bits) à 8 Mhz.

Mac II : MC68020 (32/32 bits) à 16 Mhz, coprocesseur arithmétique 68881 pour des calculs fulgurants (200 fois plus rapide qu'avec le 68020 seul).

La vitesse

Mac SE : bien qu'utilisant le même processeur 68000, 15 à 20 % sont gagnés grâce à des accès mémoire plus rapides. Les accès au disque dur SCSI sont deux fois plus rapide.

Mac II : un 68020 à 16 Mhz ne déçoit pas ; transfert via SCSI à 1 Mo/s

La mémoire vive

Mac SE : il est livré avec 1 Mo de Ram, extensible à 4 Mo.

Mac II : 1 Mo extensible à 8 Mo sur carte mère mais 15000 Mo sur cartes d'extension. Prix des 15 gigas ?

La mémoire 'morte'

Mac SE : 256 Ko (Contre 128Ko sur le Mac Plus)

Mac II : 256 Ko

Disponibilité

Mac SE : mars 87. **Mac II** : juin 87.
Extension 2 Mo (SE ou II) : mai.

Prix

Mac SE : avec 1 Mo de Ram et disque dur intégré : 35 000 Frs TTC.

Mac II : avec 1 Mo de Ram et disque dur intégré 20 Mo, deux lecteurs 800 Ko, moniteur monochrome, carte vidéo : 65 000 Frs TTC.



Cryptage de fichiers confidentiels : Kruptos



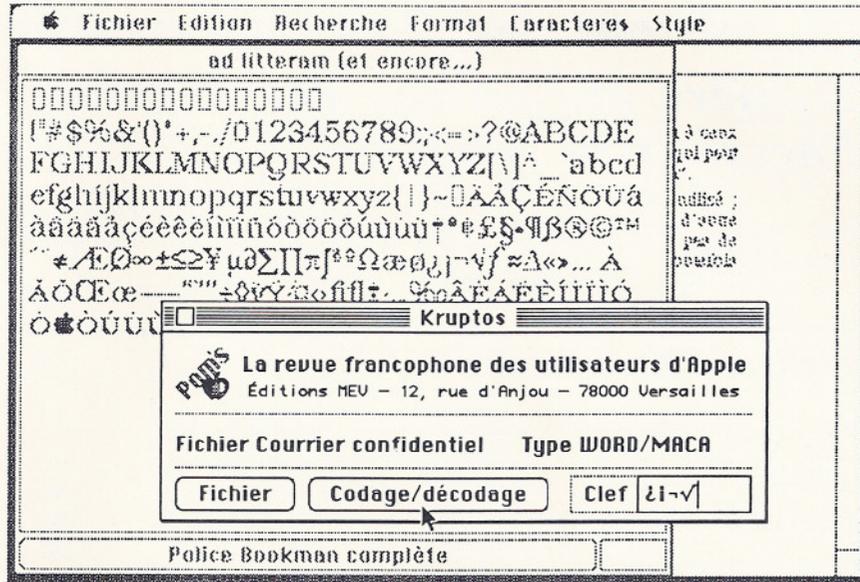
Jean-Luc Bazanegue

Le but de l'accessoire de bureau "Kruptos" est le cryptage – et, bien sûr, le décryptage – de fichiers confidentiels, ceci afin de les rendre incompréhensibles et, de manière générale, inexploitable par des tiers.

Effets de Kruptos

Un fichier codé à l'aide de Kruptos présente les caractéristiques suivantes :

- il n'a plus d'icône personnalisée et donc, vu du Finder, il est banalisé et l'éventuel 'espion' ne pourra pas faire la différence entre, par exemple, un fichier MacWrite et un fichier MacPaint ;
- il n'est plus 'double cliquable', et toute tentative d'ouverture par cette méthode se soldera par un message «Il n'y a pas d'application pour ouvrir ce document.» ;
- un essai de chargement depuis l'application correspondante – généralement avec l'article 'Ouvrir' du menu 'Fichier' – reste possible, mais aboutira selon les applications, prévoyantes ou pas, au mieux à un message pouvant ressembler à «Ceci n'est pas un fichier créé par Blurp», et au pire à la bombe. Il va sans dire que pour essayer d'ouvrir le fichier depuis l'application, il faut d'abord deviner quelle est cette dernière ;
- le contenu du fichier, après codage, étant très 'abîmé', les éditeurs de secteurs et autres outils de 'déplombage' ne seront d'aucun secours au curieux.



Fichier 'Secret' (fichier MacWrite en format texte)

5GWdè..m Fà..enx	B5 C7 57 64 7B 0C 97 ED A0 46 40 83 1C E5 EE 78
.Lf.ù>~tx.~1(1.>	1A CC E6 9C FC 3E FE F4 78 9F 7E DD 28 5D 17 3E
ù.UQU4,.Qz2<..W	FF 7C 1E D6 51 D5 B4 2C 1B 51 FA B2 BC 8C 00 57
..9.a1a7zç...ém	85 A7 9A 39 2E E1 B1 E1 B7 7A DC 06 06 06 7B 6D
oW./..éé.Q.!EC.K'	6F 57 84 AF 89 FD 7B 00 51 8E A1 45 C3 05 4B 4F
4J'Jx'3/tGhrEr!0	B4 CA 27 CA F8 A7 B3 2F 74 C7 48 F2 45 72 21 4F
NGv'ès..h2\$QI.eD	4E C7 F6 27 FD 73 14 9F E8 B2 24 51 5B 04 E5 C4
FiG45e#\$+Yt'>.A	C6 69 C7 B4 B5 E5 24 A3 A4 2B D9 74 27 3E 0E 41
Ko/N/QJ2ù.QJE ..	CB EF 2F 4E 2F 51 CA B2 FC 1A 51 CA 45 A0 95 10
'L4J..-sVetM8GCr	27 CC B4 CA FF 2E AD F3 56 E5 74 CD 38 47 43 72
Zù.EFLzi...d=..A	DA 7C 1E C5 46 CC FA 69 OD 14 9A E4 BD 97 07 41
.&."ux3RK1...cm	85 A5 82 26 22 F5 F8 B3 D2 4B EC 06 1B 19 63 6D
..U:J&,V.Uwl...&	2C OF D6 BA DD A5 2C 59 01 D6 F7 49 8C 00 04 A6
4..U-0z8jçCcC7>.	B4 85 20 D6 AD B0 FA 38 6A DC 43 E3 43 37 3E 0E
..é?>YH[u...A<	0A 83 FB 3F BE 26 59 C8 DB F5 07 14 18 41 BC 97
J-âf8rkLPb~ùâ*z	CA 2D C0 E6 B8 F2 6B FF CC 50 E2 7E 7C 40 23 7A
Ah?.iP1èd..HAr..	C1 E8 3F 07 69 50 DD FD E4 06 05 C8 41 F2 85 1C
-*..g0*	2D DE AA 83 E7 30 A3

Utiliser Kruptos

L'accessoire est très simple à utiliser. Il faut d'abord choisir le fichier à crypter en utilisant le bouton 'Fichier', qui provoque l'affichage de la fenêtre de sélection qui vous propose tous les fichiers sauf :

- les applications (type APPL/xxxx) ;
- les fichiers 'système' (type xxxx/MACS – system, Finder, Clipboard File...);
- les fichiers ouverts ou en cours d'utilisation.

Une fois le fichier sélectionné, l'accessoire affiche le nom du fichier, ainsi que son type. L'affichage du type – lui aussi codé – permet d'avoir une

indication, quand on n'en est plus bien sûr, sur le codage ou le non codage du fichier : dans le cas d'un fichier non codé, le type est généralement cohérent, par exemple 'MACA' pour un fichier MacWrite, alors que le traitement rend invariablement le type fantaisiste (†±Ö3, *I,â...). Afin que le type ne facilite pas la tâche du pirate, son codage est différent de celui du fichier proprement dit.

Il faut ensuite (ou avant la sélection du fichier, l'ordre n'a pas d'importance) choisir la clef à utiliser pour le codage. La clef est constituée de quatre

caractères quelconques qui peuvent être frappés directement au clavier, ou encore 'collés'. Dans ce dernier cas, et si l'on utilise un accessoire de bureau comme le 'ad litteram' du numéro 25, on dispose de 243 caractères différents, ce qui nous donne 3 486 784 401 combinaisons possibles.

Il ne reste plus qu'à actionner le bouton 'Codage/décodage' pour que l'opération soit effectuée, au rythme d'environ 80Ko par minute (le temps d'exécution peut varier sensiblement en fonction de la mémoire de masse utilisée).

Le codage peut être effectué plusieurs fois de suite avec des clefs différentes ou identiques, ce qui peut augmenter le nombre de possibilités jusqu'à 4 294 967 296. L'ordre utilisé pour le décodage d'un tel fichier peut ne pas se faire dans l'ordre. Par exemple, si l'on a utilisé les clefs 1234, ABCD et POMS, le fichier pourra être décodé en donnant d'abord ABCD, puis POMS, puis enfin 1234.

Anecdotique : cette version de Kruptos est compatible avec la version pour Apple][publiée dans ce numéro ; il est donc possible de coder un fichier sur un Mac et le décodé sur un Apple][, ou l'inverse.



Petit jeu : si vous êtes le premier à téléphoner pour nous communiquer la clef qui a servi à crypter le fichier 'secret' listé page précédente, vous gagnerez le cadeau indiqué dans le fichier. Pour vous faciliter la tâche : il s'agit d'un fichier créé avec MacWrite et sauvegardé en format 'texte seul'.

#1=UC#0XCEBWTIÉr *dwo=TuquOZ*o%Á=089^0evF...cEgKoojD -ceWE*trPKC#1-@M'afS=CÚr(AW4TfD=80te*20FgALio-

Fichiers et sources de l'accessoire 'Kruptos'

Note : le caractère "f" indique la continuité de la ligne courante.

Fichier 'Kruptos.Job'

Asm	Kruptos.Asm	Exec	Edit
Link	Kruptos.Link	Font/DA Mover	Edit

Fichier 'Kruptos.Link'

```

]
/Resources
Kruptos
/output Accessoire "Kruptos"
/Type 'DFIL' 'DMOV'
$

```

Source 'Kruptos.Asm'

```

INCLUDE Kruptos/1.Asm
INCLUDE Kruptos/2.Asm
END

```

Source 'Kruptos/1.Asm'

RESOURCE 'DRVR' 29 'Kruptos'

```

INCLUDE FSEqu.Txt
INCLUDE MacTraps.D
INCLUDE SysEqu.D
INCLUDE ToolEqu.D
INCLUDE QuickEqu.D
INCLUDE PackMacs.Txt

HandleCtlF EQU WindowSize
HandleCtlC EQU HandleCtlF+4
HandlText EQU HandleCtlC+4
HandleControle EQU HandlText+4
CreateurFichier EQU HandleControle+4
TypeFichier EQU CreateurFichier+6
FichierCourant EQU TypeFichier+6
ReponseGetFile EQU FichierCourant+64
TamponIO EQU ReponseGetFile+72
DrapeauMdf EQU TamponIO+ioFQEISize
DrapeauColle EQU DrapeauMdf+1
DrapeauCurs EQU DrapeauColle+1

```

DrapeauEOF	EQU	DrapeauCurs+1
Clef	EQU	DrapeauEOF+1
LongueurD	EQU	Clef+4
LongueurR	EQU	LongueurD+4
Tampon522	EQU	LongueurR+4
CompteurTiming	EQU	Tampon522+522
TailleTampon	EQU	CompteurTiming+4

Oui	EQU	\$100
MaxC	EQU	4
BS	EQU	8
CR	EQU	13

Base	DC	\$400
	DC	0
	DC	\$16A
	DC	0
	DC	Ouverture-Base
	DC	Status-Base
	DC	Controle-Base
	DC	Status-Base
	DC	Fermeture-Base

Titre DC.B 7,'Kruptos'

; Ouverture de l'accessoire

```

Ouverture
MOVEM.L D3-D7/A1-A4,-(SP)
MOVEA.L A1,A4
TST.L dCtlWindow(A4)
BNE Status4
SUBQ.L #4,SP
MOVE.L SP,-(SP)
_GetPort
MOVE.L #TailleTampon,D0
_NewPtr,clear
TST D0
BEQ.S MemoireOK
MOVE #7,-(SP)
_SysBeep
BRA Status2

```

```

MemoireOK
MOVEA.L A0,A3
SUBQ.L #4,SP
MOVE.L A3,-(SP)
PEA RectFenetre
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)

```

; Affichage des boutons

```

SUBQ.L #4,SP
MOVE.L A3,-(SP)
PEA RectBoutonF
PEA TitreBoutonF
MOVE #Oui,-(SP)
CLR.L -(SP)
CLR.L -(SP)
CLR.L -(SP)

```

```

_NewControl
MOVE.L (SP),HandleCtlF(A3)
MOVE.L A3,-(SP)
PEA RectBoutonC
PEA TitreBoutonC
MOVE #Oui,-(SP)
CLR.L -(SP)
CLR.L -(SP)
MOVEQ #1,D0
MOVE.L D0,-(SP)

```

```

_NewControl
MOVE.L (SP)+,HandleCtlC(A3)
ST DrapeauMdf(A3)
BSR InvalideBouton

```

; Enregistrement 'TextEdit'

```

BSR PoliceChicago
SUBQ.L #4,SP
PEA RectVisTexte
MOVE.L (SP)-,(SP)

```

```

_TENew
MOVE.L (SP),HandlText(A3)

```

```

_TEActivate
SF DrapeauCurs(A3)

```

```

_InitCursor

```

Status2

```

_SetPort

```

Status4

```

MOVE.L dCtlWindow(A4),A3

```

Status3

```

MOVE.M.L (SP)+,D3-D7/A1-A4

```

Status

```

MOVEQ #0,D0
RTS

```

; Fermeture de l'accessoire

Fermeture

```

MOVE.M.L A3-A4,-(SP)
MOVEA.L A1,A4
MOVEA.L dCtlWindow(A4),A3
MOVE.L HandlText(A3)-,(SP)

```

```

_TeDispose

```

```

MOVE.L A3,-(SP)

```

```

_DisposWindow

```

```

CLR.L dCtlWindow(A4)

```

```

MOVEA.L A4,A1

```

```

MOVE.M.L (SP)+,A3-A4

```

```

BRA Status

```

; Routine de contrôle

Controle

```

MOVE.M.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 CSCCode(A2),D0

```

```

CMPI #accEvent,D0

```

```

BEQ.S Evenement

```

```

CMPI #accCursor,D0

```

```

BEQ Curseur

```

```

CMPI #accCut,D0

```

```

BEQ Couper

```

```

CMPI #accCopy,D0

```

```

BEQ Copier

```

```

CMPI #accPaste,D0

```

```

BEQ Coller

```

```

CMPI #accClear,D0

```

```

BEQ Effacer

```

```

BRA Status3

```

; Aiguillage des événements

Evenement

```

MOVEA.L CSPParam(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 Status3

```

; 'click' dans la fenêtre de l'accessoire

Contenu

```

PEA EvtMouse(A2)

```

```

_GlobalToLocal

```

```

CLR -(SP)

```

```

MOVE.L EvtMouse(A2)-,(SP)

```

```

MOVE.L A3,-(SP)

```

```

PEA HandleControle(A3)

```

```

_FindControl

```

```

TST (SP)+

```

```

BEQ.S @1

```

```

CLR -(SP)

```

```

MOVE.L HandleControle(A3)-,(SP)

```

```

MOVE.L EvtMouse(A2)-,(SP)

```

```

CLR.L -(SP)

```

```

_TrackControl

```

```

TST (SP)+

```

```

BEQ.S @1

```

```

SUBQ.L #4,SP

```

```

MOVE.L HandleControle(A3)-,(SP)

```

```

_GetCRefCon

```

```

BSR Boutons

```

```

BRA Status3

```

@1

```

CLR -(SP)

```

```

MOVE.L EvtMouse(A2)-,(SP)

```

```

PEA RectVisTexte

```

```

_PtInRect

```

```

TST (SP)+

```

```

BEQ Status3

```

@3

```

MOVE.L evtMouse(A2)-,(SP)

```

```

BTST #shiftKey,evtMeta(A2)

```

```

SNE D0

```

```

MOVE.B D0,-(SP)

```

```

MOVE.L HandlText(A3)-,(SP)

```

```

_TEClick

```

*Sur la
disquette
Pom's 29, cet
accessoire est
installé dans le
menu ⌘ avec
les 5 autres
accessoires
décrits page
75.
Il est
également
sous la forme
d'un fichier
'Font
DAMover'*

```

BRA      Status3

; Action sur le clavier.
Touche
BTST    #0,evtMeta(A2)
BEQ.S   @4
MOVE    evtMessage+2(A2),D0
ANDI    #DF,D0
SUBI    #86,D0
BEQ     Coller
SUBQ    #2,D0
BEQ     Couper
SUBI    #FFEB,D0
BNE.S   @4
BRA     Copier
@4  CMPI.B #CR,evtMessage+3(A2)
BEQ.S   @5
CMPI.B #BS,evtMessage+3(A2)
BEQ.S   @1
BSR     DeReference
CMPI    #MaxC,teLength(A0)
BNE.S   @1
MOVE    teSelStart(A0),D0
CMP     teSelEnd(A0),D0
BNE.S   @1
@5  MOVE    #7,-(SP)
      _SysBeep
BRA     Status3
@1  MOVE    evtMessage+2(A2),-(SP)
      MOVE.L HandlText(A3),-(SP)
      _TEKey
Key2
BSR     DeReference
CMPI    #MaxC,teLength(A0)
BEQ.S   @1
BSR     InvalideBouton
BRA.S   @2
@1  BSR     ValideBouton
@2  BRA     Status3

```

; Routine de mise à jour

```

MiseJour
MOVE.L  A3,-(SP)
      _BeginUpdate
BSR     PoliceChicago
MOVEA.L (A5),A0
PEA     gray(A0)
      _PenPat
MOVE.L  #00300008,-(SP)
      _MoveTo
MOVE.L  #00300160,-(SP)
      _LineTo
MOVE.L  #00500008,-(SP)
      _MoveTo
MOVE.L  #00500160,-(SP)
      _LineTo
PEA     RectGrisTexte
      _FrameRect
      _PenNormal
PEA     RectTexte
      _FrameRect
BSR     AffichePoms
MOVE.L  #00170030,-(SP)
      _MoveTo
PEA     Chaine0
      _DrawString
MOVE.L  #00260034,-(SP)
      _MoveTo
BSR     PoliceMonaco

```

```

PEA     Chaine1
      _DrawString
BSR     PoliceChicago
TST.B   FichierCourant(A3)
BEQ.S   @2
BSR     AffChaineFichier
BSR     AffChaineType
@2  MOVE.L #006400FD,-(SP)
      _MoveTo
PEA     Chaine4
      _DrawString
@1  MOVE.L A3,-(SP)
      _DrawControls
PEA     RectVisTexte
      _EraseRect
PEA     portRect(A3)
MOVE.L  HandlText(A3),-(SP)
      _TEUpdate
MOVE.L  A3,-(SP)
      _EndUpdate
BRA     Status3

```

; Réponse à un message 'Activate' ou 'Deactivate'

```

Active
BTST    #activeFlag,evtMBut(A2)
BEQ.S   @1
MOVE.L  HandlText(A3),-(SP)
      _TEActivate
SF      DrapeauColle(A3)
BRA     Status3
@1  MOVE.L HandlText(A3),-(SP)
      _TEDeactivate
BRA     Status3

```

; Changement de la forme du curseur.

```

Curseur
Buffer  SET     -4
LINK    A6,#Buffer
MOVE.L  HandlText(A3),-(SP)
      _TEIdle
PEA     Buffer(A6)
      _GetMouse
CLR     -(SP)
MOVE.L  Buffer(A6),-(SP)
PEA     RectVisTexte
      _PtInRect
TST     (SP)+
BEQ.S   @1
TST.B   DrapeauCurs(A3)
BNE.S   @2
SUBQ.L  #4,SP
MOVE    #iBeamCursor,-(SP)
      _GetCursor
MOVEA.L (SP)+,A0
MOVE.L  (A0),-(SP)
      _SetCursor
ST      DrapeauCurs(A3)
BRA.S   @2
@1  TST.B DrapeauCurs(A3)
BEQ.S   @2
SF      DrapeauCurs(A3)
      _InitCursor
@2  UNLK   A6
BRA     Status3

```

```

Couper
BSR     VerifSelection
BEQ     Status3
MOVE.L  HandlText(A3),-(SP)

```





```
_TeCut
BSR TEToScrap
BSR ValideBouton
BRA Status3

Copier
BSR VerifSelection
BEQ Status3
MOVE.L HandlText(A3),-(SP)
_TeCopy
BSR TEToScrap
BRA Status3

Coller
TST.B DrapeauColle(A3)
BNE.S @2
ST DrapeauColle(A3)
BSR TEFromScrap
@2 MOVEA.L HandlText(A3),A0
MOVEA.L (A0),A0
MOVE TeScrpLength,D0
ADD teLength(A0),D0
MOVE teSelEnd(A0),D1
MOVE teSelStart(A0),D2
SUB D2,D1
SUB D1,D0
CMPI #MaxC+1,D0
BLT.S @1
MOVE #7,-(SP)
_SysBeep
BRA Status3
@1 TST TeScrpLength
BEQ Status3
MOVE.L HandlText(A3),-(SP)
_TePaste
BRA Key2

Effacer
BSR VerifSelection
BEQ Status3
MOVE.L HandlText(A3),-(SP)
_TEDelete
BSR ValideBouton
BRA Status3

ValideBouton
TST.B DrapeauMdf(A3)
BNE.S @1
TST.B FichierCourant(A3)
BEQ.S @1
BSR DeReference
CMPI #MaxC,teLength(A0)
BNE.S @1
MOVE.L HandleCtlC(A3),-(SP)
CLR -(SP)
_HiliteControl
ST DrapeauMdf(A3)
@1 RTS

InvalideBouton
TST.B DrapeauMdf(A3)
BEQ.S @1
MOVE.L HandleCtlC(A3),-(SP)
MOVE #$FF,-(SP)
_HiliteControl
SF DrapeauMdf(A3)
@1 RTS

TEFromScrap
MOVE.M.L A0-A1/D0-D2,-(SP)

SUBQ #4,SP
MOVE.L teScrpHandle,-(SP)
MOVE.L #'TEXT',-(SP)
PEA scratch8
_GetScrap
MOVE.L (SP)+,D0
BPL.S @1
MOVEQ #0,D0
@1 MOVE D0,TeScrpLength
MOVE.M.L (SP)+,A0-A1/D0-D2
RTS

TEToScrap
MOVE.M.L A0-A1/D0-D2,-(SP)
SUBQ #4,SP
_ZeroScrap
ADDQ #4,SP
MOVEA.L teScrpHandle,A0
SUBQ #4,SP
_GetHandleSize
MOVE.L D0,-(SP)
_HLock
MOVE.L #'TEXT',-(SP)
MOVE.L (A0),-(SP)
_PutScrap
MOVEA.L teScrpHandle,A0
_HUnlock
MOVE.L (SP)+,D0
MOVE.M.L (SP)+,A0-A1/D0-D2
RTS

DeReference
MOVEA.L HandlText(A3),A0
MOVEA.L (A0),A0
RTS

PoliceChicago
CLR -(SP)
_TextFont
MOVE #12,-(SP)
_TextSize
RTS

PoliceMonaco
MOVE #monaco,-(SP)
_TextFont
MOVE #9,-(SP)
_TextSize
RTS

VerifSelection
BSR.S DeReference
MOVE teSelStart(A0),D0
CMP teSelEnd(A0),D0
RTS

AffichePoms
Tampon SET -bitmapprec
LINK A6,#Tampon
MOVE.M.L A0-A4/D0-D2,-(SP)
LEA Poms,A0
MOVE.L A0,Tampon+baseAddr(A6)
MOVE #4,Tampon+rowBytes(A6)
CLR.L Tampon+bounds(A6)
MOVE.L #$200020,Tampon+bounds+bottom(A6)
PEA Tampon+baseAddr(A6)
MOVEA.L A3,A0
ADDQ.L #2,A0
MOVE.L A0,-(SP)
PEA Tampon+bounds(A6)
```

```

PEA RectPoms
CLR -(SP)
CLR.L -(SP)
_CopyBits
MOVEM.L (SP)+,A0-A4/D0-D2
UNLK A6
RTS

```



```

AffChaineFichier
MOVE.L #$00470008,-(SP)
_MoveTo
PEA Chaine2
_DrawString
PEA FichierCourant(A3)
_DrawString
RTS

```

```

AffChaineType
PEA RectEffType
_EraseRect
MOVE.L #$004700D8,-(SP)
_MoveTo
PEA Chaine3
_DrawString
PEA TypeFichier+1(A3)
_DrawString
MOVE #' /,-(SP)
_DrawChar
PEA CreateurFichier+1(A3)
_DrawString
RTS

```

```

InfoFichier
LEA TamponIO(A3),A0
LEA ReponseGetFile+rName(A3),A1
MOVE.L A1,ioFileName(A0)
MOVE ReponseGetFile+rVolume(A3),j
ioVRefNum(A0)
CLR ioFDirIndex(A0)
CLR.B ioFileType(A0)
_GetFileInfo
RTS

```

```

RectFenetre DC 41,4,152,365
RectGrisTexte DC 85,245,107,286
RectTexte DC 85,285,107,353
RectVisTexte DC 85+3,285+3,107-3,353-3
RectBoutonF DC 86,8,106,80
RectBoutonC DC 86,88,106,232
RectPoms DC 8,8,40,40
RectEffType DC 58,248,77,352

```

```

Chaine0 DC.B 45,'La revue francophone des ]
utilisateurs d"Apple'
Chaine1 DC.B 49,'Éditions MEV — 12, rue d"Anjou ]
— 78000 Versailles'
Chaine2 DC.B 8,'Fichier ',0
Chaine3 DC.B 5,'Type '
Chaine4 DC.B 4,'Clef',0
TitreBoutonF DC.B 7,'Fichier'
TitreBoutonC DC.B 15,'Codage/décodage'

```

.Align 2

```

Poms
DC.L $00000000,$00000050,$00000080,$00001100
DC.L $000008AA,$00000155,$000000AA,$00001401
DC.L $00002A02,$00004514,$0000A280,$00015140
DC.L $000228A0,$00051440,$00028A00,$00014560
DC.L $0028A2A0,$00545140,$008A2980,$00453D7C

```

```

DC.L $00A2FFD6,$1451FEAB,$2A28FD45,$4515FAA3
DC.L $A20AFD45,$5100FAA3,$2A007D46,$14007AAA
DC.L $0A003D54,$05001AA8,$02800DD0,$01000660

```

```

Curseur1
DC.L $07C01930,$21084104,$41048102,$8102FFFE
DC.L $81028102,$41044104,$21081930,$07C00000
DC.L 0,0,0,0,0,0,0
DC 7,7

```

```

Curseur2
DC.L $07C01830,$20484044,$4084B082,$8D028382
DC.L $8162821A,$42044404,$24081830,$07C00000
DC.L 0,0,0,0,0,0,0
DC 7,7

```

```

Curseur3
DC.L $07C01830,$20085014,$48248442,$82828102
DC.L $82828442,$48245014,$20081830,$07C00000
DC.L 0,0,0,0,0,0,0
DC 7,7

```

```

Curseur4
DC.L $07C01830,$24084404,$4204821A,$81628382
DC.L $8D02B082,$40844044,$20481830,$07C00000
DC.L 0,0,0,0,0,0,0
DC 7,7

```

Source 'Kruptos/2.Asm'

; Traitement en cas d'action sur un bouton.

```

Boutons
LINK A6,#0
TST 10(A6)
BEQ BoutonFichier

```

; Action sur le bouton 'Codage/décodage'

BoutonCodage

; Copie de la clef dans le tampon de quatre octets 'Clef'

```

BSR DeReference
MOVE.A.L teTextH(A0),A0
MOVE.A.L (A0),A0
LEA Clef(A3),A1
MOVEQ #3,D0
@1 MOVE.B (A0)+,(A1)+
DBRA D0,@1

```

; Vérification de la clef : on compare deux caractères consécutifs. S'il sont identiques, on les modifie.

```

LEA Clef(A3),A1
MOVEQ #2,D5
@2 MOVE.B 0(A1,D5),D0
CMP.B 1(A1,D5),D0
BNE.S @3
MOVE.B 1(A1,D5),D1
ADDQ #1,D5
ADD.B D5,D0
ADD.B D5,D1
SUBQ #1,D5
EORI.B #$FF,D0
MOVE.B D0,0(A1,D5)
MOVE.B D1,1(A1,D5)
@3 DBRA D5,@2

```

; Traitement de la partie 'données' du fichier. Si le fichier ne contient que des ressources, ce segment n'est pas exécuté.

```

TST.L LongueurD(A3)
BEQ.S PartieRessources
PartieDonnees

```

; Ouverture de la partie 'données'.

```
LEA    TamponIO(A3),A0
LEA    ReponseGetFile+rName(A3),A1
MOVE.L A1,ioFileName(A0)
MOVE   ReponseGetFile+rVolume(A3),]
       ioVRefNum(A0)
LEA    Tampon522(A3),A1
MOVE.L A1,ioOwnBuf(A0)
MOVE.B #fsRdWrPerm,ioPermssn(A0)
_Open
```

; Appel du sous-programme de codage.

```
BSR    Code_Decode
```

; Positionnement de la fin de fichier pour la partie 'données'.

```
LEA    TamponIO(A3),A0
MOVE.L LongueurD(A3),ioLEOF(A0)
_SetEOF
```

; Fermeture de la partie 'données' du fichier

```
LEA    TamponIO(A3),A0
_Close
```

; Traitement de la partie 'ressource' du fichier. Si le fichier ne
; contient que des ressources, ce segment n'est pas exécuté.

PartieRessources

```
TST.L  LongueurR(A3)
BEQ    FinCodage
```

; Ouverture de la partie 'ressource'.

```
LEA    TamponIO(A3),A0
LEA    ReponseGetFile+rName(A3),A1
MOVE.L A1,ioFileName(A0)
MOVE   ReponseGetFile+rVolume(A3),]
       ioVRefNum(A0)
LEA    Tampon522(A3),A1
MOVE.L A1,ioOwnBuf(A0)
MOVE.B #fsRdWrPerm,ioPermssn(A0)
_OpenRF
```

; Appel du sous-programme de codage.

```
BSR    Code_Decode
```

; Positionnement de la fin de fichier pour la partie 'ressource'.

```
LEA    TamponIO(A3),A0
MOVE.L LongueurR(A3),ioLEOF(A0)
_SetEOF
```

; Fermeture de la partie 'ressource' du fichier.

```
LEA    TamponIO(A3),A0
_Close
```

FinCodage

```
BSR    InfoFichier
```

; Codage du type du fichier.

```
MOVE.L Clef(A3),D5
MOVE.B Clef(A3),D0
ROR.B  #3,D0
MOVE.B D0,Clef(A3)
MOVE.B Clef+1(A3),D0
ROL.B  #2,D0
MOVE.B D0,Clef+1(A3)
MOVE.B Clef+2(A3),D0
ROR.B  #4,D0
MOVE.B D0,Clef+2(A3)
MOVE.B Clef+3(A3),D0
```

```
ROL.B  #5,D0
MOVE.B D0,Clef+3(A3)
MOVE.L Clef(A3),D0
ROL.L  #3,D0
MOVE.L D5,Clef(A3)
```

; Réécriture des informations du fichier avec le type modifié.

```
LEA    TamponIO(A3),A0
EOR.L  D0,ioFlUsrWds+fdCreator(A0)
MOVE.L ioFlUsrWds+fdCreator(A0),]
       CreateurFichier+2(A3)
```

_SetFileInfo

```
BSR    AffChaineType
```

; Mise à jour du volume.

```
LEA    TamponIO(A3),A0
CLR.L  ioFileName(A0)
MOVE   ReponseGetFile+rVolume(A3),]
       ioVRefNum(A0)
```

_FlushVol

; Effacement de la zone de saisie

```
MOVEQ  #0,D0
MOVE.L D0,-(SP)
MOVEQ  #4,D0
MOVE.L D0,-(SP)
MOVE.L HandlText(A3),-(SP)
_TESetSelect
MOVE.L HandlText(A3),-(SP)
_TEDelete
BSR    InvalideBouton
BRA    SortieBoutons
```

; Action sur le bouton 'Fichier'

BoutonFichier

; Affichage de la fenêtre de sélection. Les fichiers sont filtrés
; par la procédure 'FiltreFichiers'.

```
MOVE.L #$001E000C,-(SP)
CLR.L  -(SP)
PEA    FiltreFichiers
MOVE   #-1,-(SP)
CLR.L  -(SP)
CLR.L  -(SP)
PEA    ReponseGetFile(A3)
MOVE   #SFGetFile,-(SP)
_Pack3
TST    ReponseGetFile+rGood(A3)
BEQ.S  SortieBoutons
```

; Lecture des infos sur le fichier.

```
BSR    InfoFichier
```

; Déplacement du nom de fichier, et modification s'il est trop
; long pour tenir dans la place qui lui est réservé.

```
MOVEQ  #0,D0
MOVE.B ReponseGetFile+rName(A3),D0
ADDQ   #1,D0
LEA    ReponseGetFile+rName(A3),A0
LEA    FichierCourant(A3),A1
_BlockMove
CLR    -(SP)
PEA    FichierCourant(A3)
_StringWidth
CMPI   #144,(SP)+
BLT.S  @1
MOVEQ  #0,D4
```



```

LEA      FichierCourant(A3),A2
MOVE.B  (A2),D4
@2 MOVE.B #'...',0(A2,D4)
CLR      -(SP)
PEA      FichierCourant(A3)
_StringWidth
CMPI     #144,(SP)+
BLT.S    @1
SUBQ.B   #1,(A2)
SUBQ     #1,D4
BRA.S    @2

```



```

CMPI.L   #'APPL',ioFIUsrWds+fdType(A0)
BEQ.S    FiltreNon
CMPI.L   #'MACS',ioFIUsrWds+fdCreator(A0)
BEQ.S    FiltreNon
TST.B    ioFIAttrib(A0)
BMI.S    FiltreNon
MOVE     ioFIUsrWds+fdFlags(A0),D0
ANDI     #$5000,D0
BNE.S    FiltreNon
CLR      12(A6)
BRA.S    FiltreOK

```

; Mise en réserve des infos qui seront utilisées par la suite.

```

@1 MOVE.L  TamponIO+ioFIUsrWds+fdType(A3),]
      TypeFichier+2(A3)
      MOVE.L  TamponIO+ioFIUsrWds+fdCreator(A3),]
      CreateurFichier+2(A3)
      MOVE.B  #4,TypeFichier+1(A3)
      MOVE.B  #4,CreateurFichier+1(A3)
      MOVE.L  TamponIO+ioFILgLen(A3),LongueurD(A3)
      MOVE.L  TamponIO+ioFIRLgLen(A3),LongueurR(A3)

```

; Vers affichage des chaînes.

```

BSR      AffChaineFichier
BSR      AffChaineType
BSR      ValideBouton
SortieBoutons
UNLK     A6
MOVE.L   (SP)+,(SP)
RTS

```

; Routine de codage (ou décodage).

```

Code_Decode
PEA      Curseur1
_SetCursor
SUBQ.L   #4,SP
_TickCount
MOVE.L   (SP)+,CompteurTiming(A3)
MOVE.L   Clef(A3),D6
LEA      Scratch8,A4
LEA      Curseur1,A2
MOVEQ    #1,D4
MOVEQ    #-4,D5
MOVEQ    #4,D7
@1 BSR    CurseurAttente
LEA      TamponIO(A3),A0
MOVE.L   D7,ioReqCount(A0)
MOVE.L   A4,ioBuffer(A0)
MOVE     #fsAtMark,ioPosMode(A0)
_Read
BNE      @2
ROLL     #1,D6
EOR.L    D6,scratch8
MOVE.L   D7,ioReqCount(A0)
MOVE.L   A4,ioBuffer(A0)
MOVE     #fsFromMark,ioPosMode(A0)
MOVE.L   D5,ioPosOffset(A0)
_Write
BRA.S    @1
@2 SF     DrapeauCurs(A3)
_InitCursor
RTS

```

; Filtre pour la fenêtre de sélection des fichiers.

```

FiltreFichiers
LINK     A6,#0
MOVEM.L  D0-D2/A0-A2,-(SP)
MOVE.L   8(A6),A0

```

```

FiltreNon
MOVE     #$FFFF,12(A6)
FiltreOK
MOVEM.L  (SP)+,D0-D2/A0-A2
UNLK     A6
MOVE.L   (SP),4(SP)
ADDQ.L   #4,SP
RTS

```

; Gestion du curseur 'animé'.

```

CurseurAttente
SUBQ.L   #4,SP
_TickCount
MOVE.L   CompteurTiming(A3),D0
ADDQ.L   #2,D0
CMP.L    (SP)+,D0
BMI.S    ChangeCurseur
RTS

```

```

ChangeCurseur
MOVE     #68,D0
MULU    D4,D0
PEA     0(A2,D0)
_SetCursor
SUBQ.L   #4,SP
_TickCount
MOVE.L   (SP)+,CompteurTiming(A3)
ADDQ     #1,D4
ANDI     #$0003,D4
RTS

```

Apple et Minitel : les caractères semi-graphiques

L'article de la page 10 de ce numéro vous donne les codes des caractères semi-graphiques ainsi que le moyen d'y accéder ; voici un petit programme qui vous permettra de concrétiser la méthode. N'oubliez pas de brancher le câble Mac/Minitel !

```

DEFINT A-Z
OPEN "COM1:1200,E,7,1" FOR OUTPUT AS 1
PRINT#1,CHR$(&HC) CHR$(&HE);
GOSUB Affiche
PRINT#1,CHR$(&H1B) CHR$(&H5A);
GOSUB Affiche
PRINT#1,CHR$(&HF);
CLOSE:END

```

```

Affiche:
FOR I=&H20 TO &H3F:PRINT#1,CHR$(I);NEXT
FOR I=&H5F TO &H7E:PRINT#1,CHR$(I);NEXT
PRINT#1,CHR$(13) CHR$(10):RETURN

```

Cryptage de fichiers confidentiels : Kruptos

Christian Piard

Il arrive fréquemment que des fichiers utilisés sur un ordinateur doivent rester confidentiels, courrier privé, rapport professionnel, recette de la sauce aux câpres par exemple... Un programme Basic, un source assembleur, une feuille de calcul Multiplan en cours d'élaboration (pardon, en cours de *développement*), doivent parfois être hors d'atteinte pour éviter des modifications intempestives.

C'est l'objet de la commande externe ProDOS que nous vous proposons ici.

Nous ne reviendrons pas sur le principe des commandes externes ProDOS, abordé par A. Avrane dans le numéro 20 de Pom's ; rappelons simplement qu'une fois installée, une commande externe s'utilise en mode direct ou en mode programme tout comme CAT, VERIFY, ou RENAME par exemple.

Le cryptage

La clef comporte 4 caractères quelconques, y compris les caractères de contrôle, ce qui permet de choisir parmi plus de 200 000 000 de clef possibles... Le programme modifiera toutefois (sans le dire) les clefs comportant plusieurs caractères identiques.

Le OU EXCLUSIF

Un OU EXCLUSIF (EOR) est effectué entre les 4 premiers caractères du fichiers et la clef. La *table de vérité* du OU exclusif est la suivante :

- 1 EOR 1 = 0
- 1 EOR 0 = 1
- 0 EOR 1 = 1
- 0 EOR 0 = 0

Autrement dit, *l'un ou l'autre, mais pas les deux* (Initiation à

l'assembleur 4, G. Michel, Pom's 14).

Prenons par exemple les 4 premiers caractères d'un fichier : 'jour' ce qui correspond aux codes ASCII \$6A, \$6F, \$75, \$72.

Prenons pour clef : 'zss1' c'est-à-dire \$5A, \$53, \$53, \$31. Voici la représentation binaire de chacune des deux chaînes et l'opération OU exclusif effectuée entre les deux :

```
011101010011011110111010101110010
01011010010100110101001100110001
-----
00110000001111000010011001000011
```

Le résultat correspond à \$30, \$3C, \$26, \$43 soit : '0<&C'.

Ici, on peut donc écrire :

jour EOR zss1 = 0<&C
Le mot *jour* après codage est méconnaissable.

La rotation

Pour ne pas trop faciliter le travail de l'éventuel espion, avant chaque EOR entre 4 caractères du fichier et la clef, on effectue une rotation de cette clef d'un bit à gauche :

Prenons pour exemple la clef P159. Avant le premier codage, nous décalons tous les bits d'une position sur la gauche (ROL en assembleur). Il faut bien entendu récupérer le bit qui tombe à gauche dans le bit de retenue du registre d'état pour le faire entrer à droite.

```
P159 =
01010000001100000011010100111001
P159 décalé =
10100000011000000110101001110010
```

Le 0 de gauche est passé à droite et la clef utilisée réellement devient : '[espace]0jr '

Du fait de cette transformation à chaque étape, un fichier composé

uniquement d'espaces et crypté avec la clef 'P159' donne le résultat suivant :

```
'BJR `dtE!)Ij£3s5&FGJ,mnt8...'
(les caractères soulignés sont des caractères de contrôle).
```

Enfin, un jeu d'EOR et d'incréméntation transforme la clef au cas où deux caractères consécutifs seraient identiques (clef aabb par exemple).

Créer la commande

Si vous disposez de la disquette d'accompagnement de Pom's, ce paragraphe ne vous est pas indispensable.

Pour obtenir la commande, saisissez le fichier binaire KRUPTOS listé ci-après puis sauvegardez-le.

Si vous optez pour la saisie du source assembleur, après assemblage, vous obtenez le fichier KRUPTOS.CODE qui n'est pas directement exécutable. Il vous faut également le programme d'installation des commandes externes CMDLOAD d'A. Avrane (Pom's 20). Ce programme est constitué des codes qui vont de \$2000 à \$20FF dans le fichier KRUPTOS listé ci-après. Vous avez maintenant entre les mains les deux fichiers nécessaires ; faites alors :

```
BLOAD CMDLOAD
BLOAD KRUPTOS.CODE
BSAVE KRUPTOS,A$2000,L$300
```

Mode d'emploi

Pour installer votre nouvelle commande, faites :

```
-KRUPTOS
```

La syntaxe pour chaque cryptage et décryptage est la suivante :

KRUPTOS chemin d'accès, clef [,Sslot] [,Ddrive]

Chemin d'accès : nom de votre fichier, éventuellement précédé de préfixes,

Clef : 4 caractères,

Slot et Drive : sont optionnels.

Si la clef n'est pas valide, vous serez gratifié d'un INVALID PARAMETER.

L'opération étant réversible, pour décrypter un fichier, il faut employer la même commande, ...avec la bonne clef.

Si le fichier a été crypté à l'aide de deux clefs successivement, décryptez-le avec les deux (l'ordre n'est pas important).

Note

Tous les fichiers sont cryptables par cette commande sauf :

- les fichiers de type SYS (ProDOS, Aw.System, Basic, System...),
- ceux de type DIR (catalogues de la disquette),
- ceux de type \$F0 (commandes ProDOS),

et ce, par souci de précaution : un FILE TYPE MISMATCH vous

Fichier 'SECRET'

Il s'agit du listage d'un fichier TEXT, créé par AppleWriter. À vous de trouver le premier la clef qui a servi au cryptage et de nous appeler pour recevoir le cadeau indiqué ci-dessous...

```

0000: 5GWdè..m Fà..enx B5 C7 57 64 7B 0C 97 ED A0 46 40 83 1C E5 EE 78
0010: .Lf.ù>~tx.~](.)> 1A CC E6 9C FC 3E FE F4 78 9F 7E DD 28 5D 17 3E
0020: ù.VQU4,.Qz2<..W FF 7C 1E D6 51 D5 B4 2C 1B 51 FA B2 BC 8C 00 57
0030: '.9.ala7zç..ém 85 A7 9A 39 2E E1 B1 E1 B7 7A DC 06 06 06 7B 6D
0040: oW./..éé.Q!EC.K' 6F 57 84 AF 89 FD 7B 00 51 8E A1 45 C3 05 4B A7
0050: 4J'Jx'3/tGHRER!O B4 CA 27 CA F8 A7 B3 2F 74 C7 48 F2 45 72 21 4F
0060: NGv'ès..h2SQ[.eD 4E C7 F6 27 FD 73 14 9F E8 B2 24 51 5B 04 E5 C4
0070: FiG45e$#$S+Yt'>.A C6 69 C7 B4 B5 E5 24 A3 A4 2B D9 74 27 3E 0E 41
0080: Ko/N/QJ2ù.QJE .. CB EF 2F 4E 2F 51 CA B2 FC 1A 51 CA 45 A0 95 10
0090: 'L4J .-sVetM8GCr 27 CC B4 CA FF 2E AD F3 56 E5 74 CD 38 47 43 72
00A0: Zù.EFLzi...d=..A DA 7C 1E C5 46 CC FA 69 0D 14 9A E4 BD 97 07 41
00B0: .%.&"ux3Rk1...cm 85 A5 82 26 22 F5 F8 B3 D2 4B EC 06 1B 19 63 6D
00C0: .,V:J%,Y.VwI...& 2C 0F D6 BA DD A5 2C 59 01 D6 F7 49 8C 00 04 A6
00D0: 4. V-0z8jçCcC7>. B4 85 20 D6 AD B0 FA 38 6A DC 43 E3 43 37 3E 0E
00E0: ..é?&YH[u...A<. 0A 83 FB 3F BE 26 59 C8 DB F5 07 14 18 41 BC 97
00F0: J-áf8rk LPb-ùà#z CA 2D C0 E6 B8 F2 6B FF CC 50 E2 7E 7C 40 23 7A
0100: Ah?.iP]éd..HAr.. C1 E8 3F 07 69 50 DD FD E4 06 05 C8 41 F2 85 1C
0110: -^*.g0# 2D DE AA 83 E7 30 A3

```

rappellera à l'ordre. Pour lever ces limites, supprimer les lignes 161 à 167 du source.

Cette commande est à manipuler avec précaution car décrypter un fichier avec une mauvaise clef revient à le crypter deux fois. Dans ce cas, décryptez-le avec les deux clefs.

Il est vivement déconseillé d'utiliser des caractères de contrôle pour la clef : ceux-ci n'étant pas affichés, il est difficile de contrôler s'il n'y a pas de faute de frappe et on risque de ne pas savoir quel code a été utilisé...



Source KRUPTOS.S Assembleur ProCODE

```

0
2          DSK KRUPTOS.CODE
3
4 CLE      =      6
5 PTR      =     $48
6 HIMEM    =     $73
7 READ_BUF =     $200
8 PRINTERR =     $BE0C
9 XTADDR   =     $BE50
10 XLEN     =     $BE52
11 XCNUM    =     $BE53
12 PBITS    =     $BE54
13 VPATH1   =     $BE6C
14 VPATH2   =     $BE6E
15 GOSYSTEM =     $BE70
16 SSGINFO  =     $BEB4
17 SGET     =     $BEC6
18 SOPEN    =     $BECB
19 SREAD    =     $BED5
20 SCLOSE   =     $BEDD
21 SYSERR   =     $BF0F
24 RTS     =     $FF58
25 GFI      =     $C4
26 OPEN     =     $C8
27 READ     =     $CA
28 WRITE    =     $CB
29 CLOSE    =     $CC
30 SETMARK  =     $CE
31 GETMARK  =     $CF
32 SETEOF   =     $D0

```

I+
//e
//e+
//c
Igs

```

33 GETEOF   =     $D1
34
35          ORG     $2100
36
37 *-----
38 * Est-ce notre commande ?
39 *-----
40
41 START    CLD
42
43          LDA     #>FIN+$100
44          LDA     #>LONG-$100
45 V_OLDCMD LDA     RTS
46
47          LDA     VPATH1
48          STA     PTR
49          LDA     VPATH1+1
50          STA     PTR+1
51
52          LDY     #1          ;est-ce crypte ?
53 COMPAR   LDA     (PTR),Y
54          CMP     COMMAND-1,Y
55          BNE     NO_CMD
56          INY
57          CPY     #6+1
58          BCC     COMPAR
59
60 *-----
61 * Demande à ProDOS suite commande
62 *-----
63
64          DEY
65          DEY
66          STY     XLEN

```

ProDOS

```

67      LDA #0
68      STA SYSERR
69      STA XCNUM
70      LDA #00000011 ;2 noms de fichiers
71      STA PBITS
72      LDA #00000100 ;slot/drive autorisés
73      STA PBITS+1
74
75 V_SUITE LDA SUITE
76      LDA V_SUITE+1
77      STA XTADDR
78      LDA V_SUITE+2
79      STA XTADDR+1
80      CLC
81      RTS
82
83 NO_CMD SEC
84      JMP (V_OLDCMD+1)
85
86 *-----
87 * Controle validité clef
88 *-----
89
90 SUITE LDA VPATH2 ;vecteur nom de fichier
91      STA CLE
92      LDA VPATH2+1
93      STA CLE+1
94
95      LDY #0
96      LDA (CLE),Y
97
98      TAY ;cherche le /
99      LDX #0
100 BCL LDA (CLE),Y
101      CMP #'/'
102      BEQ SLASH
103      INX
104      DEY
105      BNE BCL
106
107 SLASH CPX #4 ;2ème nom a bien
108      BEQ CLE_OK ;4 caractères ?
109      LDA #SB ;non : invalid parameter
110      JMP PRINTERR
111
112 CLE_OK LDX #0 ;sauve la clef
113      INY
114 BCL1 LDA (CLE),Y
115      STA CLEF,X
116      INY
117      INX
118      CPX #4
119      BNE BCL1
120      DEX
121
122 BCL3 LDA CLEF,X
123      DEX
124      CMP CLEF,X
125      BNE CLE_VAL
126
127      TXA
128      ADC CLEF,X
129      EOR #11111111
130      STA CLEF,X
131      INX
132      TXA
133      ADC CLEF,X
134      STA CLEF,X
135      DEX
136
137 CLE_VAL CPX #0
138      BNE BCL3
139
140 *-----
141 * Ouverture fichier
142 *-----
143
144      LDA VPATH1
145      STA SOPEN+1
146      STA SSGINFO+1
147
148      LDA VPATH1+1
149      STA SOPEN+2
150      STA SSGINFO+2
151
152      LDA HIMEM
153      STA SOPEN+3
154      STA SREAD+5
155      LDA HIMEM+1
156      STA SOPEN+4
157
158      LDA #GFI ;get file info pour
159      JSR GOSYSTEM ;type de fichier
160      BCS ERREUR
161
162      LDA SSGINFO+4 ;controle type fichier
163      CMP #SOF ;un directory ?
164      BEQ ERRRTYPE
165      CMP #SFO ;une commande ProDOS ?
166      BEQ ERRRTYPE
167      CMP #SFF ;un fichier système ?
168      BEQ ERRRTYPE
169
170      LDA #OPEN ;ouvre le fichier
171      JSR GOSYSTEM
172      BCS ERREUR
173
174      LDA SOPEN+5 ;recopie n[ de référence
175      STA SREAD+1
176      STA SGET+1
177      STA SCLOSE+1
178
179      LDA #<READ_BUF
180      STA SREAD+2
181      LDA #>READ_BUF
182      STA SREAD+3
183
184      LDA #GETEOF ;cherche longueur fichier
185      JSR GOSYSTEM
186      BCS ERREUR
187
188      LDY #4 ;sauvegarde la longueur
189      BCLGET LDA SGET,Y
190      STA LONGUEUR,Y
191      DEY
192      BNE BCLGET
193
194 *-----
195 * Lecture/écriture
196 *-----
197
198      LDX #4 ;on lira 4 octets
199      STX SREAD+4
200
201      LOOP LDA #GETMARK ;relève la position
202      JSR GOSYSTEM ;dans le fichier
203      BCS ERREUR
204
205      LDY #2 ;la sauvegarde
206      BCLPOS LDA SGET+2,Y
207      STA POSITION,Y
208      DEY
209      BPL BCLPOS
210
211      LDA #READ ;lit fichier
212      JSR GOSYSTEM
213      BCC OK
214      CMP #5 ;fin fichier ?
215      BNE ERREUR ;non : erreur
216      BEQ FINAL
217
218      ERRRTYPE LDA #SD ;file type mismatch
219      ERREUR JMP PRINTERR
220
221      OK ROL CLEF ;modification
222      PHP ;de la clef
223      ROR CLEF
224      PLP
225      LDX #3
226      BCLCLE ROL CLEF,X
227      DEX

```

```

227      BPL BCLCLE
228
229      LDX #3          ;codage
230 BCL2  LDA READ_BUF,X
231      EOR CLEF,X
232      STA READ_BUF,X
233      DEX
234      BPL BCL2
235
236      LDY #2          ;restaure position
237 BCLPOS1 LDA POSITION,Y ;pour écriture
238      STA SGET+2,Y
239      DEY
240      BPL BCLPOS1
241
242      LDA #SETMARK    ;et repositionne
243      JSR GOSYSTEM    ;le pointeur
244      BCS ERREUR
245
246      LDA #WRITE      ;réécrit fichier
247      JSR GOSYSTEM
248      BCS ERREUR
249
250      BCC LOOP
251
252 FINAL  LDY #4          ;reprend longueur
253 BCLGET1 LDA LONGUEUR,Y ;car on a écrit
254      STA SGET,Y      ;N x 4 octets
255      DEY
256      BNE BCLGET1
257
258      LDA #SETEOF     ;remet à jour
259      JSR GOSYSTEM    ;longueur
260      BCS ERREUR
261
262      LDA #CLOSE      ;ferme fichier
263      JMP GOSYSTEM    ;terminé
264
265 *-----
266 * STOCK
267 *-----
268
269      BRK
270 COMMAND ASC 'CRYPTE'
271 CLEF DS 4
272 POSITION DS 3
273 LONGUEUR DS 4
274
275 FIN = *
276 LONG = FIN-START+$100
277

```

Récapitulation 'KRÜPTOS'

Après avoir saisi cette
récapitulation sous moniteur,
vous la sauvegarderez par :
BSAVE KRÜPTOS,A\$2000,
L\$28D

```

2000:AD 00 BF C9 4C F0 05 A9
2008:87 4C ED FD AD 4D BE F0
2010:05 A9 15 4C 09 BE AD 04
2018:21 69 00 20 98 20 90 05
2020:A9 0E 4C 09 BE CD 02 21
2028:90 F6 AE 08 BE 8D 08 BE
2030:8E 07 21 AE 07 BE 8E 06
2038:21 A0 00 8C 07 BE 48 E9
2040:21 85 3C 68 38 E9 04 85
2048:74 A9 21 85 49 84 48 A0
2050:00 B1 48 F0 27 20 8E F8
2058:A4 2F C0 02 D0 0F B1 48
2060:C9 21 90 09 CD 02 21 B0
2068:04 65 3C 91 48 A5 48 38
2070:65 2F 85 48 A5 49 69 00
2078:85 49 D0 D3 A0 00 A9 21
2080:84 3C 85 3D 18 6D 04 21
2088:84 42 88 84 3E 85 3F AD
2090:08 BE 85 43 C8 4C 2C FE
2098:8D FB 20 A5 74 18 69 04
20A0:8D FC 20 86 3D CE FC 20
20A8:F0 47 AD FC 20 8D FD 20
20B0:AD FD 20 48 4A 4A 4A AA
20B8:68 29 07 A8 B9 F3 20 3D
20C0:58 BF D0 E1 A5 3D D0 09
20C8:B9 F3 20 1D 58 BF 9D 58
20D0:BF AD FC 20 38 CE FD 20
20D8:ED FD 20 CD FB 20 D0 D0
20E0:A5 3D D0 07 18 AE FD 20
20E8:E8 8A 60 A9 00 85 3D F0

```

```

20F0:B9 38 60 80 40 20 10 08
20F8:04 02 01 00 00 00 AD 99
2100:D8 A9 23 A9 01 AD 58 FF
2108:AD 6C BE 85 48 AD 6D BE
2110:85 49 A0 01 B1 48 D9 7B
2118:22 D0 2D C8 C0 07 90 F4
2120:88 88 8C 52 BE A9 00 8D
2128:0F BF 8D 53 BE A9 03 8D
2130:54 BE A9 04 8D 55 BE AD
2138:4C 21 AD 38 21 8D 50 BE
2140:AD 39 21 8D 51 BE 18 60
2148:38 6C 06 21 AD 6E BE 85
2150:06 AD 6F BE 85 07 A0 00
2158:B1 06 A8 A2 00 B1 06 C9
2160:2F F0 04 E8 88 D0 F6 E0
2168:04 F0 05 A9 0B 4C 0C BE
2170:A2 00 C8 B1 06 9D 82 22
2178:C8 E8 E0 04 D0 F5 CA BD
2180:82 22 CA DD 82 22 D0 12
2188:8A 7D 82 22 49 FF 9D 82
2190:22 E8 8A 7D 82 22 9D 82
2198:22 CA E0 00 D0 E1 AD 6C
21A0:BE 8D CC BE 8D B5 BE AD
21A8:6D BE 8D CD BE 8D B6 BE
21B0:A5 73 8D CE BE 8D DA BE
21B8:A5 74 8D CF BE A9 C4 20
21C0:70 BE B0 64 AD B8 BE C9
21C8:0F F0 5B C9 F0 F0 57 C9
21D0:FF F0 53 A9 C8 20 70 BE
21D8:B0 4E AD D0 BE 8D D6 BE
21E0:8D C7 BE 8D DE BE A9 00
21E8:8D D7 BE A9 02 8D D8 BE
21F0:A9 D1 20 70 BE B0 31 A0
21F8:04 B9 C6 BE 99 89 22 88
2200:D0 F7 A2 04 8E D9 BE A9
2208:CF 20 70 BE B0 1A A0 02
2210:B9 C8 BE 99 86 22 88 10
2218:F7 A9 CA 20 70 BE 90 0B
2220:C9 05 D0 04 F0 3E A9 0D
2228:4C 0C BE 2E 82 22 08 6E
2230:82 22 28 A2 03 3E 82 22

```

```

2238:CA 10 FA A2 03 BD 00 02
2240:5D 82 22 9D 00 02 CA 10
2248:F4 A0 02 B9 86 22 99 C8
2250:BE 88 10 F7 A9 CE 20 70
2258:BE B0 CD A9 CB 20 70 BE
2260:B0 C6 90 A3 A0 04 B9 89
2268:22 99 C6 BE 88 D0 F7 A9
2270:D0 20 70 BE B0 B2 A9 CC
2278:4C 70 BE 00 43 52 59 50
2280:54 45 00 00 00 00 00 00
2288:00 00 00 00 00

```

Apple & Minitel caractères semi-graphiques

À titre d'illustration de
l'exploitation des caractères
semi-graphiques du Minitel (voir
page 10), voici un modeste
programme Basic qui fonctionne
sur tous les Apple // muni d'une
carte série en slot 2, configurée
ainsi : 1200 bauds, parité paire, 7
bits de données et 1 bit de stop.
Le câble de liaison décrit dans les
précédents numéros conviendra.

Programme GRAPH.VIDEOTEX

```

10 PRINT CHR$(4)"PR#2"
20 PRINT CHR$(12)CHR$(14);
30 GOSUB 100
40 PRINT CHR$(27)CHR$(90);
50 GOSUB 100
60 PRINT CHR$(15)
70 PRINT CHR$(4)"PR#0": HOME : END
100 FOR I = 32 TO 63: PRINT CHR$( I
);: NEXT : FOR I = 95 TO 126: PR
INT CHR$( I);: NEXT : PRINT CH
R$(13)CHR$(10): RETURN

```

Le SIMPLEXE est un algorithme permettant la résolution des problèmes de programmation linéaire, qui consiste à rechercher l'optimum d'une fonction linéaire à plusieurs variables liées par des équations ou des inéquations.

Développé en 1947 par Dantzig de l'US Air Force, cet algorithme est maintenant utilisé largement par les économistes et techniciens pour résoudre les problèmes les plus divers : affectation de personnel, préparation de mélanges industriels, stockages, etc.

Le programme SIMPLEXE reprend cet algorithme. Prenons l'exemple d'un pâtissier qui désire confectionner des soufflés au chocolat, des quatre-quarts et des mousses au chocolat (fichier PATISSERIE de la disquette Pom's 29). Il dispose de sucre et de chocolat en quantités limitées, et chaque type de gâteau est vendu à un prix différent. Combien de gâteaux de chaque type le pâtissier doit-il confectionner pour trouver un bénéfice maximum ?

Il faut 40 grammes de sucre par soufflé, 20 grammes par quatre-quarts et 50 grammes par mousse. Notre pâtissier ne dispose que de 2 kilos de sucre.

Pour le chocolat, les valeurs respectives sont 10, 28 et 40 ; il

n'y a qu'un seul kilo de chocolat disponible.

Soient x_1 , x_2 et x_3 les variables représentant le nombre de soufflés, quatre-quarts et mousses. Nous avons donc les CONTRAINTES :

$$40 x_1 + 20 x_2 + 50 x_3 \leq 2000$$

$$10 x_1 + 28 x_2 + 40 x_3 \leq 1000$$

Ces contraintes sont du type "inférieur ou égal". Il peut également exister des contraintes de type "supérieur ou égal" ou simplement "égal".

Le bénéfice de notre pâtissier est respectivement de 20, 20 et 40 centimes par type de produit. D'où la FONCTION ECONOMIQUE :

$$20 x_1 + 20 x_2 + 40 x_3$$

Cette fonction doit ici être maximisée ; pour d'autres exemples, elle pourrait être minimisée (si elle représentait un coût de fabrication).

Le programme SIMPLEXE permet d'éditer toutes ces données numériques et calcule le nombre de gâteaux à confectionner, compte tenu des contraintes, pour obtenir le plus grand bénéfice. Ici, il faudrait produire 27 soufflés et 18 mousses (tant pis pour les amateurs de quatre-quarts !).

Les données peuvent être sauvegardées sur disque pour être modifiées plus tard. Les résultats peuvent également être sortis sur une imprimante Epson RX80 (l'adaptation à d'autres imprimantes ne pose aucun problème). Le programme, s'il donne entière satisfaction pour la résolution des problèmes posés, est cependant volontairement limité à l'ossature afin d'alléger sa lecture, et ne comprend donc pas les multiples routines de gestion des erreurs possibles de saisie. À vous de le compléter suivant vos goûts.



Fichier 'PATISSERIE'

3
2
0
0
2
6
40
20
50
2000
10
28
20
1000
-20
-20
-40
2

//+
//e
//e+
//c
//gs

DOS 3.3

Programme 'SIMPLEXE'

Afin de faciliter la saisie, les espaces situés dans des chaînes de caractères ont été remplacés par des puces ('.').

```
100 REM *****
    REM *****
110 :
120 REM CATTAN Serge          ALGORI
    THME DU SIMPLEXE          09/85
130 :
140 REM *****
```

```
*****
150 REM ONERR GOTO 3470
160 T$ = "-----"
    -----"
170 D$ = CHR$(4)
180 HOME
190 INVERSE : PRINT SPC(40)
200 PRINT ".....ALGORITHME•DU•SIM
    PLEXE.....";
210 PRINT SPC(40)
220 NORMAL
230 POKE 34,4: VTAB 8
```

```

240 PRINT "ANALYSE•D'UN•PROBLEME•.....-
->•1"
250 PRINT "REPRISE•ANALYSE•EXISTANTE•...-
->•2"
260 PRINT "MODIFICATIONS•DES•DONNEES•...-
->•3"
270 PRINT "MODE•D'EMPLOI•.....-
->•4"
280 PRINT "RETOUR•AU•BASIC•.....-
->•5"
290 POKE 36,11: VTAB 18: INPUT "VOTRE•CH
OIX•?•";R$:R = VAL (R$)
300 ON R GOTO 340,320,4240,3490,330
310 GOTO 290
320 GOSUB 2540: GOTO 990
330 POKE 34,0: END
340 HOME
350 VTAB 23: INPUT "NOM•DE•L'ANALYSE•(•?
•POUR•LE•CATALOGUE•)•?•";NE$
360 IF LEN (NE$) < 1 OR NE$ = "?" THEN
PRINT D$;"CATALOG": GOTO 350
370 PRINT : INPUT "NOMBRE•D'INCONNUES•?•
";N$
380 N = VAL (N$): IF N = 0 THEN PRINT "
•UN•CHIFFRE•>•0•S.V.P.": GOTO 370
390 PRINT : INPUT "NOMBRE•DE•CONTRAINTES
•DU•TYPE•<=•?•";T1$: PRINT
400 IF LEN (T1$) = 0 THEN T1$ = ""
410 T1 = VAL (T1$): IF ASC (T1$) < 48 O
R ASC (T1$) > 57 THEN PRINT "•UN•CH
IFFRE•S.V.P.": GOTO 390
420 PRINT : INPUT "NOMBRE•DE•CONTRAINTES
•DU•TYPE•>=•?•";T2$: PRINT
430 IF LEN (T2$) = 0 THEN T2$ = ""
440 T2 = VAL (T2$): IF ASC (T2$) < 48 O
R ASC (T2$) > 57 THEN PRINT "•UN•CH
IFFRE•S.V.P.": GOTO 420
450 PRINT : INPUT "NOMBRE•DE•CONTRAINTES
•DU•TYPE•==•?•";T3$: PRINT
460 IF LEN (T3$) = 0 THEN T3$ = ""
470 T3 = VAL (T3$): IF ASC (T3$) < 48 O
R ASC (T3$) > 57 THEN PRINT "•UN•CH
IFFRE•S.V.P.": GOTO 450
480 D1 = T1 + T2 + T3:D2 = N + D1 + T2 +
1
490 GOSUB 510: GOTO 540
500 REM ===== DIM TABLEAU =====
=====
510 IF ME = 1 THEN RETURN
520 DIM PG(D1,D2),BA(D1),EC(2,D2),A(D1),
A1(D1)
530 ME = 1: RETURN
540 REM =====
550 REM SAISIE =====
560 HOME
570 IF T1 = 0 THEN 670
580 FOR K = 1 TO T1
590 PRINT K;
600 IF K = 1 THEN PRINT "IERE•CONTRAI
NT
E•DU•TYPE•<=•:": PRINT : GOTO 620
610 PRINT "IEME•CONTRAINTE•DU•TYPE•<=•:
": PRINT
620 FOR J = 0 TO N - 1
630 PRINT "COEFFICIENT•DE•X";J + 1;: INP
UT "•=•?•";B$:B = VAL (B$):PG(K - 1,
J) = B
640 NEXT J
650 INPUT "VALEUR•DU•SECOND•MEMBRE•=•?•"
;B$:B = VAL (B$):PG(K - 1,D2) = B: P
RINT
660 NEXT K
670 IF T2 = 0 THEN 780
680 L1 = T1 + 1:L2 = T2 + T1
690 FOR K = L1 TO L2
700 PRINT K - L1 + 1;
710 IF K - L1 + 1 = 1 THEN PRINT "IERE•
CONTRAINTE•DU•TYPE•>=•:": PRINT : GOT
O 730
720 PRINT "IEME•CONTRAINTE•DU•TYPE•>=•:
": PRINT
730 FOR J = 0 TO N - 1
740 PRINT "COEFFICIENT•DE•X";J + 1;: INP
UT "•=•?•";B$:B = VAL (B$):PG(K - 1,J
) = B
750 NEXT J
760 INPUT "VALEUR•DU•SECOND•MEMBRE•=•?•"
;B$:B = VAL (B$):PG(K - 1,D2) = B: P
RINT
770 NEXT K
780 IF T3 = 0 THEN 890
790 L1 = T1 + T2 + 1:L2 = T1 + T2 + T3
800 FOR K = L1 TO L2
810 PRINT K - L1 + 1;
820 IF K - L1 + 1 = 1 THEN PRINT "IERE•
CONTRAINTE•DU•TYPE•==•:": PRINT : GOT
O 840
830 PRINT "IEME•CONTRAINTE•DU•TYPE•==•:
": PRINT
840 FOR J = 0 TO N - 1
850 PRINT "COEFFICIENT•DE•X";J + 1;: INP
UT "•=•?•";B$:B = VAL (B$):PG(K - 1,
J) = B
860 NEXT J
870 INPUT "VALEUR•DU•SECOND•MEMBRE•=•?•"
;B$:B = VAL (B$):PG(K - 1,D2) = B: P
RINT
880 NEXT K
890 A1 = N + T1 - 1:A2 = A1 + T2:A3 = A2
+ T2:A4 = A3 + T3
900 PRINT "FONCTION•ECONOMIQUE•:•": PRIN
T
910 FOR J = 1 TO N
920 PRINT "COEFFICIENT•DE•X";J;: INPUT "
•=•";B$:CX = VAL (B$)
930 EC(1,J - 1) = - CX
940 NEXT J
950 PRINT : PRINT "TAPER•1•POUR•UNE•MINI
MISATION": HTAB 7: PRINT "2•POUR•UNE•
MAXIMISATION": PRINT : HTAB 7: INPUT
MO$
960 MO = VAL (MO$)
970 IF MO < > 1 AND MO < > 2 THEN PRI
NT : FLASH : PRINT "1•OU•2•S.V.P.": N
ORMAL : PRINT : GOTO 950
980 IF R < > 2 THEN GOSUB 2180
990 ZZ = 1E - 5

```

```

1000 IF MO = 2 THEN MO = - 1
1010 PRINT : FLASH : PRINT "CALCUL•EN•CO
      URS•...": NORMAL
1020 REM =====
1030 REM INIT VARIABLES AUXILIAIRES
1040 REM =====
1050 IF T1 = 0 THEN 1090
1060 FOR J = 1 TO T1
1070 PG(J - 1,N + J - 1) = 1:BA(J - 1) =
      N + J - 1
1080 NEXT J
1090 IF T2 = 0 THEN 1130
1100 FOR J = 1 TO T2
1110 PG(J + T1 - 1,A1 + J) = - 1:PG(J +
      T1 - 1,A2 + J) = 1:BA(J + T1 - 1) =
      A2 + J
1120 NEXT J
1130 IF T3 = 0 THEN 1170
1140 FOR J = 1 TO T3
1150 PG(J + T1 + T2 - 1,A3 + J) = 1:BA(J
      + T1 + T2 - 1) = A3 + J
1160 NEXT J
1170 IF T2 + T3 = 0 THEN 1440
1180 FOR J = 0 TO N + T1 + T2 - 1
1190 S = 0
1200 FOR I = T1 TO D1
1210 S = S + PG(I,J)
1220 NEXT I
1230 EC(0,J) = S
1240 NEXT J
1250 EC(0,D2) = 0
1260 FOR I = T1 TO D1
1270 EC(0,D2) = EC(0,D2) + PG(I,D2)
1280 NEXT I
1290 EX = 0:CO = 1
1300 GOSUB 1790
1310 IF RE = 2 THEN 1660
1320 IF EC(0,D2) > ZZ THEN 1670
1330 V = 0
1340 FOR C = 0 TO D1
1350 IF BA(C) > A2 THEN V = 1
1360 NEXT C
1370 IF V = 1 THEN 1680
1380 FOR J = A2 + 1 TO A4
1390 FOR I = 0 TO D1
1400 PG(I,J) = 0
1410 NEXT I
1420 EC(0,J) = 0:EC(1,J) = 0
1430 NEXT J
1440 EX = 1:CO = MO: GOSUB 1790
1450 IF RE = 2 THEN 1670
1460 HTAB 15: PRINT "*****": PRINT
1470 REM =====
1480 REM AFFICHAGE PROGRAMME OPTIMAL
1490 REM =====
1500 GOSUB 2110: REM ON IMPRIME ?
1510 HOME : PRINT
1520 PRINT "BASE•DE•DONNEES••":NE$
1530 PRINT "=====": PRINT
1540 PRINT "SOLUTION•OPTIMALE•POUR•": P
      RINT
1550 FOR I = 1 TO N
1560 FOR J = 0 TO D1
1570 IF BA(J) = I - 1 THEN PRINT "X";I;
      "•="";PG(J,D2):A(I - 1) = PG(J,D2):
      GOTO 1600
1580 NEXT J
1590 PRINT "X";I;"•="0"
1600 NEXT I
1610 PRINT : HTAB 15: PRINT "*****
      *"
1620 PRINT : PRINT CHR$(15): GOSUB 303
      0: REM AFFICHAGE EQUATIONS
1630 PRINT CHR$(18): REM RETOUR 80 C
      OLLONNES
1640 GOSUB 2160: POKE 34,0: PRINT D$;"PR
      #0": PRINT : RUN
1650 REM =====
1660 GOSUB 2110: HOME : PRINT "CE•PROBLE
      ME•N'A•PAS•DE•SOLUTION•OPTIMALE": PR
      INT : GOTO 1610
1670 GOSUB 2110: HOME : PRINT "CE•PROBLE
      ME•N'A•PAS•DE•SOLUTION•DU•TOUT": PRI
      NT : GOTO 1610
1680 FOR JJ = 0 TO A4
1690 IF EC(0,JJ) > ZZ THEN 1740
1700 FOR I = 0 TO D1
1710 PG(I,JJ) = 0
1720 NEXT I
1730 EC(0,JJ) = 0:EC(1,JJ) = 0
1740 NEXT JJ
1750 GOTO 1440
1760 REM =====
1770 REM ALGORITHME DU SIMPLEXE
1780 REM =====
1790 MA = ZZ
1800 FOR J = 0 TO D2 - 1
1810 IF MA < EC(EX,J) * CO THEN MA = EC(
      EX,J) * CO:J1 = J
1820 NEXT J
1830 IF MA = ZZ THEN RE = 1: RETURN
1840 MI = 1E20
1850 FOR I = 0 TO D1
1860 A = PG(I,J1)
1870 IF A < ZZ THEN 1900
1880 R = PG(I,D2) / A
1890 IF R < MI THEN MI = R:I1 = I
1900 NEXT I
1910 IF MI = 1E20 THEN RE = 2: RETURN
1920 BA(I1) = J1:PI = PG(I1,J1)
1930 FOR J = 0 TO D2
1940 PG(I1,J) = PG(I1,J) / PI
1950 NEXT J
1960 FOR I = 0 TO D1
1970 IF I1 = I THEN 2020
1980 B1 = PG(I,J1)
1990 FOR J = 0 TO D2
2000 PG(I,J) = PG(I,J) - B1 * PG(I1,J)
2010 NEXT J
2020 NEXT I
2030 B1 = EC(EX,J1):B2 = EC(EX + 1,J1)
2040 FOR J = 0 TO D2
2050 EC(EX,J) = EC(EX,J) - B1 * PG(I1,J)
2060 EC(EX + 1,J) = EC(EX + 1,J) - B2 * P
      G(I1,J)
2070 NEXT J

```

```

2080 GOTO 1790
2090 REM =====
2100 REM IMPRESSSION RESULTAT ?
2110 REM =====
2120 PRINT : INPUT "VOULEZ-VOUS•IMPRIMER
•CES•RESULTATS•?";R$
2130 IF LEFT$(R$,1) = "O" THEN IM = 1:
PRINT D$;"PR#1"
2140 RETURN
2150 REM =====
2160 IF IM = 0 THEN POKE 36,0: VTAB 23:
PRINT "APPUYEZ•SUR•UNE•TOUCHE•POUR•
LA•SUITE.": GET C$
2170 RETURN
2180 REM SAUVEGARDE SUR DISQUE
2190 REM =====
2200 PRINT
2210 PRINT D$;"OPEN";NE$
2220 PRINT D$;"WRITE";NE$
2230 PRINT N: PRINT T1: PRINT T2: PRINT
T3
2240 PRINT D1: PRINT D2
2250 IF T1 = 0 THEN 2320
2260 FOR K = 1 TO T1
2270 FOR J = 0 TO N - 1
2280 PRINT PG(K - 1,J)
2290 NEXT J
2300 PRINT PG(K - 1,D2)
2310 NEXT K
2320 IF T2 = 0 THEN 2400
2330 L1 = T1 + 1:L2 = T2 + T1
2340 FOR K = L1 TO L2
2350 FOR J = 0 TO N - 1
2360 PRINT PG(K - 1,J)
2370 NEXT J
2380 PRINT PG(K - 1,D2)
2390 NEXT K
2400 IF T3 = 0 THEN 2480
2410 L1 = T1 + T2 + 1:L2 = T1 + T2 + T3`
2420 FOR K = L1 TO L2
2430 FOR J = 0 TO N - 1
2440 PRINT PG(K - 1,J)
2450 NEXT J
2460 PRINT PG(K - 1,D2)
2470 NEXT K
2480 FOR J = 1 TO N
2490 PRINT EC(1,J - 1)
2500 NEXT J
2510 PRINT MO
2520 PRINT D$;"CLOSE"
2530 RETURN
2540 REM LECTURE ENREGISTREMENT
2550 REM =====
2560 VTAB 23: INPUT "NOM•DE•L'ANALYSE•(•
?•POUR•LE•CATALOGUE•)•?•";NE$
2570 IF LEN(NE$) < 1 OR NE$ = "?" THEN
PRINT D$;"CATALOG": GOTO 2560
2580 HOME : INVERSE : POKE 36,8: VTAB 23
: PRINT "ANALYSE•FICHIER•...";NE$: N
ORMAL
2590 PRINT
2600 PRINT D$;"OPEN";NE$
2610 PRINT D$;"READ";NE$
2620 INPUT N: INPUT T1: INPUT T2: INPUT
T3
2630 INPUT D1: INPUT D2
2640 GOSUB 510
2650 A1 = N + T1 - 1:A2 = A1 + T2:A3 = A2
+ T2:A4 = A3 + T3
2660 IF T1 = 0 THEN 2730
2670 FOR K = 1 TO T1
2680 FOR J = 0 TO N - 1
2690 INPUT PG(K - 1,J)
2700 NEXT J
2710 INPUT PG(K - 1,D2)
2720 NEXT K
2730 IF T2 = 0 THEN 2810
2740 L1 = T1 + 1:L2 = T2 + T1
2750 FOR K = L1 TO L2
2760 FOR J = 0 TO N - 1
2770 INPUT PG(K - 1,J)
2780 NEXT J
2790 INPUT PG(K - 1,D2)
2800 NEXT K
2810 IF T3 = 0 THEN 2890
2820 L1 = T1 + T2 + 1:L2 = T1 + T2 + T3
2830 FOR K = L1 TO L2
2840 FOR J = 0 TO N - 1
2850 INPUT PG(K - 1,J)
2860 NEXT J
2870 INPUT PG(K - 1,D2)
2880 NEXT K
2890 FOR J = 1 TO N
2900 INPUT EC(1,J - 1)
2910 NEXT J
2920 INPUT MO
2930 PRINT D$;"CLOSE"
2940 RETURN
2950 REM TRAITEMENT ERREUR=====
2960 REM =====
2970 IF PEEK(222) = 5 THEN VTAB 10: P
RINT "FICHIER•INEXISTANT": GOTO 2990
2980 PRINT "ERREUR"
2990 FOR IT = 1 TO 1000: NEXT
3000 POKE 34,0: GOTO 180
3010 RETURN
3020 REM =====
3030 REM LISTE DES DONNEES
3040 REM =====
3050 GOSUB 2600: REM RECUP INFOS FICHIE
R
3060 PRINT "EQUATIONS•DU•TYPE•<•:"
3070 PRINT "-----": PRIN
T
3080 FOR K = 1 TO T1
3090 FOR J = 0 TO N - 1
3100 PRINT PG(K - 1,J);"•x";J + 1;: IF J
< N - 1 THEN PRINT "•+•";
3110 A1(K) = PG(K - 1,J) * A(J) + A1(K)
3120 NEXT J
3130 PRINT "•<•";PG(K - 1,D2)
3140 PRINT "VALEUR•CALCULEE•:";A1(K): PR
INT
3150 NEXT K
3160 IF T2 = 0 THEN 3280
3170 L1 = T1 + 1:L2 = T2 + T1

```

```

3180 PRINT "EQUATIONS•DU•TYPE•>•:"
3190 PRINT "-----": PRIN
T
3200 FOR K = L1 TO L2
3210 FOR J = 0 TO N - 1
3220 PRINT PG(K - 1, J); "•x"; J + 1; : IF J
< N - 1 THEN PRINT "•+•";
3230 A1(K) = PG(K - 1, J) * A(J) + A1(K)
3240 NEXT J
3250 PRINT "•>•"; PG(K - 1, D2)
3260 PRINT "VALEUR•CALCULEE•:"; A1(K) : PR
INT
3270 NEXT K
3280 IF T3 = 0 THEN 3400
3290 L1 = T1 + T2 + 1; L2 = T1 + T2 + T3
3300 PRINT "EQUATIONS•DU•TYPE•==•:"
3310 PRINT "-----": PRIN
T
3320 FOR K = L1 TO L2
3330 FOR J = 0 TO N - 1
3340 PRINT PG(K - 1, J); "•x"; J + 1; : IF J
< N - 1 THEN PRINT "•+•";
3350 A1(K) = PG(K - 1, J) * A(J) + A1(K)
3360 NEXT J
3370 PRINT "•==•"; PG(K - 1, D2)
3380 PRINT "VALEUR•CALCULEE•:"; A1(K) : PR
INT
3390 NEXT K
3400 PRINT : PRINT "FONCTION•ECONOMIQUE•
:";
3410 IF MO = 1 THEN PRINT "POUR•UNE•MIN
IMISATION.": GOTO 3430
3420 PRINT "POUR•UNE•MAXIMISATION."
3430 PRINT "=====
=====": PRINT
3440 FOR J = 1 TO N
3450 PRINT EC(1, J - 1); "•x"; J; "•";
3460 NEXT J
3470 PRINT : PRINT "VALEUR•DE•LA•FONCTIO
N•ECONOMIQUE•==•"; EC(1, D2)
3480 RETURN
3490 REM MODE D'EMPLOI =====
3500 REM =====
3510 HOME : PRINT : PRINT : SPEED= 140
3520 PRINT "L'ALGORITHME•DU•SIMPLEXE•A•E
TE"
3530 PRINT "DEVELOPPE•PAR•G.B. DANTZIG•E
N•1947."
3540 PRINT "IL•S'AGIT•D'UN•OUTIL•DE•CALC
UL•TRES"
3550 PRINT "UTILISE•D'ABORD•PAR•LES•MILI
TAIRES,"
3560 PRINT "PUIS•PAR•LES•ECONOMISTES•ET•
LES"
3570 PRINT "INDUSTRIELS.": PRINT
3580 PRINT "LES•APPLICATIONS•SONT•NOMBRE
USES.": PRINT
3590 PRINT "--PREPARATION•DE•MELANGES•IND
USTRIELS,"
3600 PRINT "--AFFECTATION•DE•PERSONNEL,"
3610 PRINT "--PLAN•DE•PRODUCTION,"
3620 PRINT "--INVESTISSEMENTS•ETC."
3630 PRINT
3640 PRINT "POUR•SITUER•LE•PROBLEME, NOUS
•PRENDRONS"
3650 PRINT "L'EXEMPLE•D'UN•PATISSIER•QUI
•DESIRE
3660 PRINT "CONFECTIONNER•DES•SOUFFLES•A
U•CHOCOLAT,"
3670 PRINT "DES•QUATRE-QUARTS, ET•DES•MOU
SSES•AU"
3680 PRINT "CHOCOLAT."
3690 PRINT "POUR•CELA, IL•DISPOSE•DE•SUCR
E•ET•DE"
3700 PRINT "CHOCOLAT•EN•QUANTITES•LIMITE
ES, MAIS
3710 PRINT "SANS•LIMITATION•POUR•LES•AUT
RES"
3720 PRINT "INGREDIENTS."
3730 PRINT "DE•PLUS, CHAQUE•GATEAU•EST•VE
NDU•A•UN"
3740 PRINT "PRIX•QUI•TIENT•COMPTE•DU•COUT
•DES"
3750 PRINT "DIVERS•INGREDIENTS.": PRINT
3760 INVERSE : PRINT "PROBLEME.": NORME
L
3770 PRINT "COMBIEN•DE•GATEAUX•DE•CHAQUE
•TYPE"
3780 PRINT "LE•PATISSIER•DOIT•PREPARER•E
T•VENDRE"
3790 PRINT "POUR•EN•TIRER•UN•PROFIT•MAXI
MAL•?"
3800 PRINT "LES•MATIERES•PREMIERES•SE•RE
PARTISSENT"
3810 PRINT "AINSI.": PRINT
3820 PRINT "POUR•LE•SUCRE.:"
3830 PRINT "40•GR. POUR•LES•QUATRE-QUART
S,"
3840 PRINT "20•GR. POUR•LES•SOUFFLES,"
3850 PRINT "50•GR. POUR•LES•MOUSSES."
3860 PRINT : PRINT : PRINT "POUR•LE•CHOC
OLAT.:"
3870 PRINT "10•GR. POUR•LES•QUATRE-QUART
S,"
3880 PRINT "28•GR. POUR•LES•SOUFFLETS,"
3890 PRINT "20•GR. POUR•LES•MOUSSES."
3900 PRINT : PRINT "LE•POIDS•DE•SUCRE•DI
SPONIBLE•EST•2000•GR."
3910 PRINT "LE•POIDS•DE•CHOCOLAT•EST•100
0•GR."
3920 PRINT : PRINT "LES•INCONNUES•SONT•L
ES•GATEAUX."
3930 PRINT "LES•EQUATIONS•POUR•CE•PROBLE
ME•SERONT.": PRINT
3940 PRINT "40X1•+•20X2•+•50X3•<=•2000"
3950 PRINT "10X1•+•28X2•+•40X3•<=•1000"
3960 PRINT : PRINT "LES•EQUATIONS•QUI•SO
NT•ICI•DES"
3970 PRINT "INEGALITES•SONT•APPELEES•CON
TRAINTES."
3980 PRINT "ELLES•PEUVENT•ETRE•DE•3•TYPE
S.": PRINT
3990 PRINT "•-INFERIEUR•OU•EGAL•(<=),"
4000 PRINT "•-SUPERIEUR•OU•EGAL•(>=),"
4010 PRINT "•-EGAL•(=)."
4020 PRINT : PRINT "LES•VALEURS•SITUEES•

```

```

A•DROITE•DE"
4030 PRINT "L'EQUATION•S'APPELLENT•SECON
D•MEMBRE."
4040 PRINT : PRINT "D'AUTRE•PART, LE•BENE
FICE•POUR•NOTRE"
4050 PRINT "PATISSIER•SERA•DE•:"
4060 PRINT : PRINT "20•CTS•PAR•QUATRE•QU
ART,"
4070 PRINT "20•CTS•PAR•SOUFFLE,"
4080 PRINT "40•CTS•PAR•MOUSSE."
4090 PRINT : PRINT "L'EQUATION•CORRESPON
DANTE•SERA•:" : PRINT
4100 PRINT "BENEFICE•=•20•+•20•+•40"
4110 PRINT : PRINT "CETTE•EQUATION•SE•NO
MME•FONCTION"
4120 PRINT "ECONOMIQUE."
4130 PRINT "LA•FONCTION•ECONOMIQUE•PEUT•
ETRE"
4140 PRINT "MAXIMISEE, C'EST•LE•CAS•D'UN•
BENEFICE"
4150 PRINT "COMME•DANS•NOTRE•EXEMPLE, •OU
•MINIMISEE"
4160 PRINT "LORSQU'IL•S'AGIRA•D'UN•COUT•
PAR•EXEMPLE."
4170 PRINT : PRINT : PRINT "CE•PROGRAMME
•A•ETE•INSPIRE•PAR•L'ARTICLE";
4180 PRINT "DE•DANIEL•FERRO•PARU•DANS•LA
•REVUE"
4190 SPEED= 255
4200 PRINT "SCIENCE•ET•VIE."
4210 PRINT : PRINT "APPUYEZ•SUR•UNE•TOUC
HE•POUR•RETOUR•AU•••MENU." : GET R$
4220 HOME : GOTO 230
4230 REM =====
4240 REM MODIFICATIONS
4250 REM =====
4260 HOME : GOSUB 2540
4270 IF T1 = 0 THEN 4370
4280 FOR K = 1 TO T1
4290 PRINT : PRINT K;
4300 IF K = 1 THEN PRINT "IERE•CONTRAIN
TE•DU•TYPE•<=•:" : PRINT : GOTO 4320
4310 PRINT "IEME•CONTRAINTTE•DU•TYPE•<=•:
": PRINT
4320 FOR J = 0 TO N - 1
4330 PRINT "COEFFICIENT•DE•X"; J + 1; "•=•
"; : INVERSE : PRINT PG(K - 1, J); : NO
RMAL : INPUT "•"; B$: B = VAL (B$) : I
F LEN (B$) > 0 THEN PG(K - 1, J) = B
4340 NEXT J
4350 PRINT "VALEUR•DU•SECOND•MEMBRE•=•";
: INVERSE : PRINT PG(K - 1, D2); : NOR
MAL : INPUT "•"; B$: B = VAL (B$) : IF
LEN (B$) > 0 THEN PG(K - 1, D2) = B
4360 NEXT K
4370 IF T2 = 0 THEN 4480
4380 L1 = T1 + 1 : L2 = T2 + T1
4390 FOR K = L1 TO L2
4400 PRINT : PRINT K - L1 + 1;
4410 IF K - L1 + 1 = 1 THEN PRINT "IERE
•CONTRAINTTE•DU•TYPE•>=•:" : PRINT : G
OTO 4430
4420 PRINT "IEME•CONTRAINTTE•DU•TYPE•>=•:
": PRINT
4430 FOR J = 0 TO N - 1
4440 PRINT "COEFFICIENT•DE•X"; J + 1; "•=•
"; : INVERSE : PRINT PG(K - 1, J); : NO
RMAL : INPUT "•"; B$: B = VAL (B$) : I
F LEN (B$) > 0 THEN PG(K - 1, J) = B
4450 NEXT J
4460 PRINT "VALEUR•DU•SECOND•MEMBRE•=•";
: INVERSE : PRINT PG(K - 1, D2); : NOR
MAL : INPUT "•"; B$: B = VAL (B$) : IF
LEN (B$) > 0 THEN PG(K - 1, D2) = B
4470 NEXT K
4480 IF T3 = 0 THEN 4590
4490 L1 = T1 + T2 + 1 : L2 = T1 + T2 + T3
4500 FOR K = L1 TO L2
4510 PRINT : PRINT K - L1 + 1;
4520 IF K - L1 + 1 = 1 THEN PRINT "IERE
•CONTRAINTTE•DU•TYPE•=•:" : PRINT : G
OTO 4540
4530 PRINT "IEME•CONTRAINTTE•DU•TYPE•=•:
": PRINT
4540 FOR J = 0 TO N - 1
4550 PRINT "COEFFICIENT•DE•X"; J + 1; "•=•
"; : INVERSE : PRINT PG(K - 1, J); : NO
RMAL : INPUT "•"; B$: B = VAL (B$) : I
F LEN (B$) > 0 THEN PG(K - 1, J) = B
4560 NEXT J
4570 PRINT "VALEUR•DU•SECOND•MEMBRE•=•";
: INVERSE : PRINT PG(K - 1, D2); : NOR
MAL : INPUT "•"; B$: B = VAL (B$) : IF
LEN (B$) > 0 THEN PG(K - 1, D2) = B
4580 NEXT K
4590 A1 = N + T1 - 1 : A2 = A1 + T2 : A3 = A2
+ T2 : A4 = A3 + T3
4600 PRINT : PRINT "FONCTION•ECONOMIQUE
•:" : PRINT
4610 FOR J = 1 TO N
4620 PRINT "COEFFICIENT•DE•X"; J; "•=•"; :
INVERSE : PRINT - EC(1, J - 1); : NOR
MAL : INPUT "•"; B$: B = VAL (B$) : I
F LEN (B$) > 0 THEN EC(1, J - 1) =
- B
4630 NEXT J
4640 PRINT : PRINT "TAPER•1•POUR•UNE•MIN
IMISATION": HTAB 7: PRINT "2•POUR•UN
E•MAXIMISATION": PRINT : HTAB 7
4650 IF MO < > 1 THEN MO = 2
4660 PRINT "VALEUR•ACTUELLE•=•"; : INVERS
E : PRINT MO; : NORMAL : INPUT "•"; MO
$
4670 IF LEN (MO$) > 0 THEN MO = VAL (M
O$)
4680 IF MO < > 1 AND MO < > 2 THEN PR
INT : FLASH : PRINT "1•OU•2•S.V.P.":
NORMAL : PRINT : GOTO 4640
4690 IF MO = 2 THEN MO = - 1
4700 PRINT : INPUT "VOULEZ•VOUS•D'AUTRES
•MODIFICATIONS•?"; R$
4710 IF LEFT$ (R$, 1) = "O" THEN HOME :
GOTO 4270
4720 GOSUB 2180
4730 POKE 34, 0 : RUN

```

AppleWriter & /RAM

Christian Piard

AppleWriter ProDOS utilise les éventuels 64Ko de votre carte 80 colonnes pour permettre le travail sur des textes plus importants : la mémoire disponible passe, avec la carte 80 colonnes étendue, de 22269 caractères à 46845 soit un gain de 24Ko seulement.

Dans certains cas, il peut être intéressant de renoncer à cette extension, afin de récupérer le disque virtuel 64Ko que ProDOS installe dans cette 80 colonnes. Il est possible, par exemple, d'y installer les fichiers de travail WPL et ceux auxquels il est

fréquemment fait appel : les temps de traitement s'en trouveront considérablement réduits.

Le principe

Lors du 'boot', ProDOS effectue des tests concernant la configuration matérielle utilisée et note le résultat en \$BF98, octet nommé MACHID. Voici le codage de cet octet :

00xxx0xx	Apple II
01xxx0xx	Apple II+
10xxx0xx	Apple //e
11xxx0xx	Apple /// en émulation
10xxx1xx	Apple //c
xx01xxxx	48Ko
xx10xxxx	64Ko
xx11xxxx	128Ko
xxxxxx0x	pas de 80 colonnes
xxxxxx1x	80 colonnes
xxxxxxx0	pas d'horloge
xxxxxxx1	horloge compatible

Par exemple, sur un Apple //e, avec une carte 80 colonnes étendue, sans horloge, on trouvera \$B2 et sur un 64Ko : \$A2.

Il suffira donc de forcer à 0 le bit 5 pour indiquer à AppleWriter que l'on ne dispose que de 64Ko, les 64 qui restent serviront au disque virtuel.

Le programme système AW .SYSTEM, premier programme exécuté au lancement d'AppleWriter, lance en fonction de MACHID : AWB.SYS, AWC .SYS ou AWD.SYS. C'est donc AW .SYSTEM que nous modifierons.

En pratique

Il convient de n'effectuer la modification que sur une copie de sauvegarde, sur laquelle au moins deux blocs seront disponibles. S'assurer en faisant le catalogue

Source AW2 Assembleur ProCODE

ProDOS

//e
//e+
//c
//gs

* Source AW2

```

ORG $2000+473
LDA $BF98 ; charge MACHID
AND #%00010000 ; a-t-on 128Ko ?
BEQ NORMAL ; non, on ne fait rien
JSR $FC58 ; efface écran
LDY #0 ; affiche message
MESSAGE LDA MESS,Y
BEQ MODIF
JSR $FDED
INY
BNE MESSAGE
BIT $C010
MODIF LDA $C000 ; attend une touche
BPL MODIF
AND #%11011111 ; en fait une majuscule
BIT $C010
CMP #"N" ; si c'est N, on ne
BEQ NORMAL ; change rien
CMP #"M" ; si ce n'est pas M, on
BNE MODIF ; boucle
LDA $BF98 ; charge MACHID
AND #%11101111 ; force à 0 le bit 5
TAY
RTS ; on revient
NORMAL LDY $BF98
RTS

MESS DFB $8D,$8D
ASC "AppleWriter..."
DFB $8D
ASC " "
DFB $8D,$8D,$8D
ASC " <N>ormal : 46845 caractères"
DFB $8D,$8D
ASC " <M>odifié : 22269 caractères"
DFB $8D
ASC " + 60928 en disque virtuel"
DFB 0

```

que la longueur de AW.SYSTEM est bien de 473 octets, sinon cette modification ne convient pas.

Suivre les étapes suivantes :

- Booter sur la disquette Master ProDOS
- Mettre la disquette AppleWriter
- UNLOCKAW.SYSTEM
- BLOADAW.SYSTEM,A\$2000,TSYS
- Mettre la disquette Pom's*
- BLOADAW1.C
- BLOADAW2.C
- Mettre la disquette AppleWriter
- BSAVEAW.SYSTEM,A\$2000,L\$298,TSYS

Maintenant, lors du démarrage sur AppleWriter, un message apparaît, vous invitant à choisir entre AppleWriter 'normal' et AppleWriter 'modifié'.

Le premier vous accorde 46845 caractères, le second 22269 seulement mais 60928 dans le disque virtuel dont le préfixe est '/RAM'.

Le message ne sera pas affiché si vous n'avez pas de carte 80

colonnes étendue.

* ou, si vous ne l'avez pas, mettez la disquette sur laquelle vous aurez sauvegardé les fichiers AW1.C et AW2.C listés ci-contre.



Source AW1 Assembleur ProCODE

* On remplace l'original
* LDA \$BF98
* par un saut à notre
* sous-programme

```
ORG    $200D
JSR    $2000+473
```

Récapitulation AW1.C

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par : BSAVE AW1.C,A\$200D,L3

200D:20 D9 21

Récapitulation AW2.C

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par : BSAVE AW2.C,A\$21D9,L\$6F

```
21D9:AD 98 BF 29 10 F0 2C
21E0:20 58 FC A0 00 B9 10 22
21E8:F0 09 20 ED FD C8 D0 F5
21F0:2C 10 C0 AD 00 C0 10 FB
21F8:29 DF 2C 10 C0 C9 CE F0
2200:0B C9 CD D0 EE AD 98 BF
2208:29 EF A8 60 AC 98 BF 60
2210:8D 8D C1 F0 F0 EC E5 D7
2218:F2 E9 F4 E5 F2 AE AE AE
2220:8D DF DF DF DF DF DF DF
2228:DF DF DF DF DF DF DF 8D
2230:8D 8D A0 A0 BC CE BE EF
2238:F2 ED E1 EC A0 A0 BA A0
2240:B4 B6 B8 B4 B5 A0 E3 E1
2248:F2 E1 E3 F4 FD F2 E5 F3
2250:8D 8D A0 A0 BC CD BE EF
2258:E4 E9 E6 E9 FB A0 BA A0
2260:B2 B2 B2 B6 B9 A0 E3 E1
2268:F2 E1 E3 F4 FD F2 E5 F3
2270:8D A0 A0 A0 A0 A0 A0 A0
2278:A0 A0 A0 A0 A0 AB A0 B6
2280:B0 B9 B2 B8 A0 E5 EE A0
2288:E4 E9 F3 F1 F5 E5 A0 F6
2290:E9 F2 F4 F5 E5 EC 00
```

Un programme WPL : Tabulations automatiques

Bernard Bel

Si vous disposez de nombreux programmes assembleur réalisés sous DOS 3.3, il peut être nécessaire d'utiliser les sources sous un autre environnement (ProDOS, Pascal 1.2, etc.).

seul espace, ce qui donne un résultat peu esthétique pour un traitement de texte et souvent incompatible avec des assembleurs qui ne suivent pas ce protocole :

au lieu de :

```
LDX #0
LOOP LDA MSG,X
      JSR COUT ; $FDED
FL1  DEX
      BPL LOOP
```

Les assembleurs classiques (Big Mac, Lisa) permettent de sauver les sources sous format texte standard. Malheureusement, les tabulations sont converties en un

```
LDX #0
LOOP LDA MSG,X
      JSR COUT ; $FDED
FL1  DEX
      BPL LOOP
```

Il serait donc fort utile de disposer d'une routine qui 'allonge' le caractère espace pour créer une réelle tabulation. Le programme proposé ici est écrit

Apple & Minitel : Ligne téléphonique 'artificielle'

Jean-Louis Chaulot-Talmon

Que ce titre ne laisse pas supposer que nous avons découvert des lignes téléphoniques naturelles. Il s'agit simplement de vous proposer un montage simple permettant de simuler une ligne téléphonique pour la phase de mise au point d'un serveur télématique, par exemple. Ou, pourquoi pas, pour mettre en place un serveur interne ne dépendant pas du RTC, le Réseau Téléphonique Commuté.

Le problème est le suivant : votre Apple fait fonction de serveur (avec un modem ou celui d'un Minitel) et vous souhaitez tester le travail.

La méthode la plus simple — la plus coûteuse — consiste à disposer de deux lignes téléphoniques, l'une connectée à l'Apple serveur, l'autre au Minitel. Bien sûr, le temps des essais est dans ce cas taxé.

Relier l'Apple serveur au Minitel à l'aide de la prise péri-informatique n'est pas d'un grand secours non plus, car certaines fonctions du serveur sont alors inopérantes.

Cette *fausse ligne* résoud le problème ; nous trouvons reliés : l'Apple, son modem serveur, notre ligne, le Minitel qui consultera son serveur. Elle résoud accessoirement le problème familial que constitue à long terme l'occupation du téléphone...

C'est ce dispositif qui est décrit ici, après un bref rappel des notions de téléphonie sur lesquelles il s'appuie.

Circuit de liaison téléphonique

La liaison téléphonique est constituée de deux fils métalliques reliant le poste de l'abonné demandeur à celui de l'abonné demandé. Notre ligne sera donc constituée, de façon analogue, de deux fils reliant les broches 1 et 3 du Minitel aux broches 1 et 3 du modem micro-serveur. Toutefois, dans la communication réelle, les parties terminales d'un circuit — les lignes d'abonnés — assurent des fonctions complémentaires et il convient d'examiner celles qui incombent à la ligne de l'abonné demandé.

Appel et supervision de l'abonné demandé

Lorsque, conformément aux signaux émis par le cadran ou le clavier d'appel du demandeur, le central téléphonique établit la liaison avec le demandé, plusieurs opérations restent à effectuer :

- alerter le destinataire en actionnant la sonnerie ;
- détecter le décrochage afin d'arrêter la sonnerie et... taxer le demandeur ;
- se mettre en mesure de détecter le raccrochage pour libérer les lignes.

Les organes essentiels d'une ligne d'abonné

Les deux fils $L1$ et $L2$ de la ligne sont réunis l'un à l'autre, à l'intérieur de la ligne téléphonique

de l'abonné, de deux façons différentes :

- d'une part, et de façon permanente, par une liaison comprenant, disposés en série, un condensateur C (en général $2 \mu F$) et une sonnerie électromagnétique S d'une impédance supérieure à 1000Ω ;
- d'autre part, mais seulement quand le combiné a été soulevé de son support (l'interrupteur CC est alors fermé), une liaison métallique comprenant notamment les contacts d'impulsion du cadran d'appel (CA) — fermés au repos —, des enroulements d'une bobine d'induction In et la résistance variable du microphone.

Au repos, la ligne n'est parcourue par aucun courant puisque :

- le poste est raccroché : la seule liaison présente entre les fils de ligne est la liaison condensateur/sonnerie, or la tension B est continue ;
- l'interrupteur d'appel I est ouvert.

Quand le moment est venu d'appeler l'abonné, l'interrupteur I se ferme, le courant alternatif A débite à travers le condensateur et la sonnerie... sonne.

Dès que l'abonné décroche, l'interrupteur CC se ferme, le courant B passe.

Pendant toute la conversation, la ligne reste alimentée ainsi en courant continu, courant qui sera modulé par le microphone à l'image des signaux sonores qu'il reçoit.

Au raccrochage, CC s'ouvre, le courant continu est interrompu.

Réalisation de la ligne artificielle

Pour que les centraux téléphoniques puissent établir, maintenir, puis libérer la communication comme ils ont coutume de le faire pour une conversation normale, le modem simule aussi tous les états successifs d'un poste téléphonique. Cela signifie, à l'inverse, que la ligne artificielle doit reconstituer les fonctions essentielles — appel et alimentation — d'une ligne d'abonné normale pour pouvoir déclencher les états successifs des modems en présence.

Elle devra donc comprendre :

- une liaison métallique entre Minitel et micro-serveur (du fil et deux joncteurs normalisés PTT $\approx 20,00$ F) ;
- une alimentation en courant continu. Compte tenu de la faible intensité nécessaire (15 à 20 mA par modem), 4 piles plates de 4,5 V connectées en

série avec une résistance ajustable R1 conviendront ;

- une alimentation en courant alternatif. Un transformateur de sonnerie d'appartement 220/12v-50 mA convient ;
- un bouton poussoir Ba pour envoyer, à la demande, le courant d'appel sur la ligne artificielle. Là encore une résistance ajustable de quelques centaines d'ohms servira à régler le courant débité lors d'un appel.

Réglage de la ligne

Un poste téléphonique ordinaire relié à l'un des joncteurs, ajuster la résistance R1 pour que le courant continu envoyé vers le poste décroché, se situe entre 13 et 17 mA (350 à 400 Ω).

Poste raccroché, ajuster R2 en appuyant sur le bouton d'appel jusqu'à ce que la sonnerie tinte très faiblement. Le courant sera alors d'environ 10 mA soit une valeur de résistance d'environ

200 Ω .

Le système est alors prêt ; il suffit de relier le micro-serveur à une extrémité et le Minitel à l'autre pour commencer les essais... gratuitement.

À noter...

...que certains modems à réponse automatique nécessitent 2, 3 ou 4 trains de courant d'appel pour répondre. Selon le cas appuyer 2, 3 ou 4 fois sur le bouton d'appel ;
...qu'avec la faible consommation, le pile auront une durée de vie substantielle ;

...que, lors de l'utilisation, le poste téléphonique est inutile et sert donc normalement sur la ligne téléphonique PTT (une source de conflits en moins) ;

...qu'accessoirement, cette ligne artificielle permet de relier deux postes téléphoniques ordinaires, pour la plus grande joie des enfants.

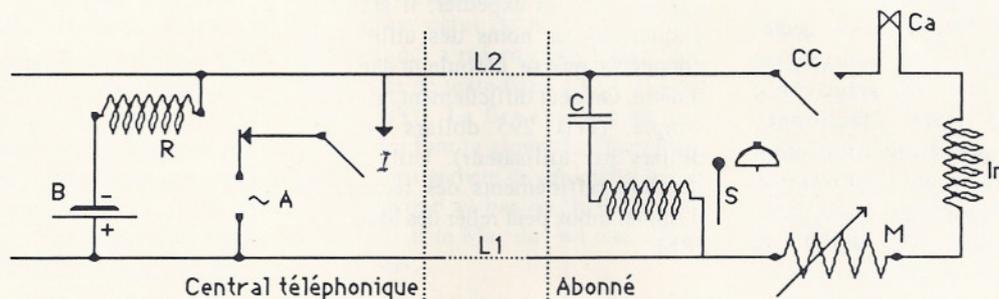


Schéma de principe Central/Poste d'abonné

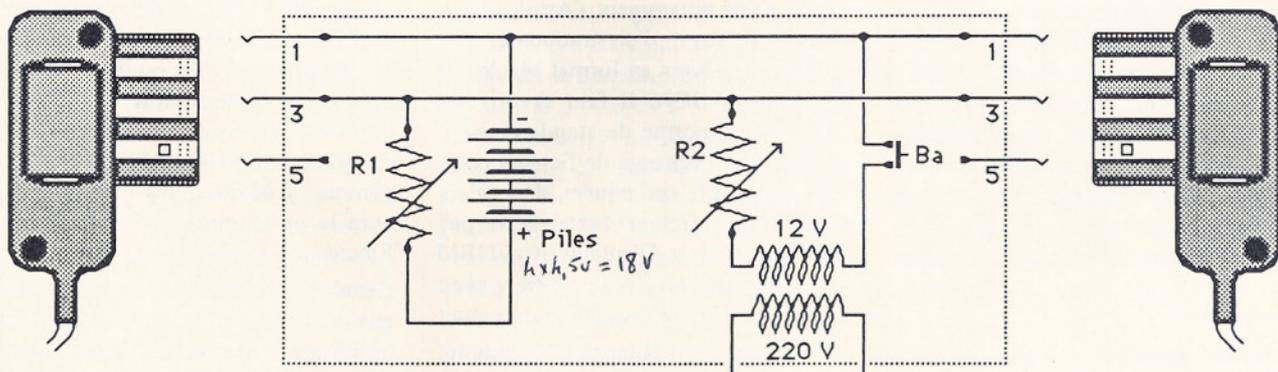


Schéma de la ligne 'artificielle'

Micro-informations

Jean-Michel Gourévitch

Vous découvrirez dans le cahier Mac les petits derniers d'Apple : l'un, le 'SE' fonctionne avec le processeur Motorola 68000, l'autre, le 'Mac II' véritable Mac 'ouvert' avec ses six *slots* pour cartes d'extension utilise un 68020. Mais d'autres nouveautés explosives concernent les communications et l'intégration du Macintosh dans des entreprises utilisant des IBM petits ou gros. On n'a pas fini d'en entendre parler, car c'est tout le monde de la micro-informatique qui va en être bouleversé.

Le nouveau serveur

La première nouveauté (annoncée d'ailleurs dans le dernier numéro de Pom's), c'est le nouveau serveur. Il répond au doux nom d'AppleShare. Et est simplement constitué d'un logiciel qui transforme un Mac en serveur dédié. Dédicé, car ce Mac ne peut alors (et c'est bien dommage) plus faire grand chose d'autre, sinon faire fonctionner simultanément une autre application tout en jouant en arrière fond son rôle de serveur. AppleShare vaut aux États-Unis quelque 800 dollars et permet de servir 25 utilisateurs. Le *plus*, c'est qu'on peut interconnecter entre eux les serveurs. Apple utilise ainsi 20 Mac servant 450 personnes. L'une des possibilités les plus révolutionnaires d'AppleShare (baptisée *multilaunching*) consiste dans la possibilité donnée à plusieurs utilisateurs de pouvoir ouvrir simultanément une application installée une seule fois. Les éditeurs de logiciels vont sûrement faire une drôle de tête. L'autre caractéristique du serveur AppleShare c'est d'être intégré à la dernière version (la 5.4) du Finder. Pour les utilisateurs c'est parfaitement transparent.

Il faudra que certains logiciels soient adaptés, mais d'ores et déjà, Tops de Centram (compagnie d'ailleurs rachetée par 3 COM qui se pose en spécialiste de

la communication sur Mac), Helix d'Odesta, les produits de General Computer (comme l'Hyperdrive) et, à coup sûr, la nouvelle version de 4e Dimension tournent sous AppleShare.

Le problème c'est donc que le Mac serveur ne peut pratiquement faire que cela. Ou presque. Car il peut faire tourner simultanément une application comme **Inbox** de **Think Technologies**. Inbox, c'est un fantastique programme de courrier électronique. On installe ce programme sur le serveur, puis sur le système de chaque utilisateur. On peut alors utiliser tout l'arsenal de boîtes à lettres, expédier des notes à certains ou des messages à tous, avec en prime l'heure d'envoi du message, la protection de certains messages par mots de passe, etc. Les messages sont limités à 30 pages. Pour les expédier, il suffit de cliquer sur les noms des utilisateurs connectés qui se déroulent dans une fenêtre. On peut difficilement faire plus simple. (Prix 295 dollars plus 75 dollars par utilisateur). Utilisant les derniers raffinements des techniques d'Apple, Inbox peut relier des Mac à des IBM.

IBM, nous voici

Car l'autre mouvement spectaculaire, c'est le pas fait par Apple vers l'univers IBM. Avec notamment l'**Apple DCA filter**. Il s'agit d'un traducteur qui traduit en fichiers au format MacWrite les fichiers DCA. Il faut savoir que DCA est la norme de standardisation d'IBM pour l'échange de fichiers entre PC. En clair, on pourra désormais traduire un fichier texte écrit par exemple sous le Displaywriter d'IBM pour le lire ou le travailler avec MacWrite. Encore faut-il le transmettre.

Le transmettre ? Rien de plus simple, car voici aussi qu'Apple sort (enfin) un produit qui était devenu presque mythique : sa carte permettant d'intégrer

tout compatible MS DOS au réseau AppleTalk. Voici une carte vendue 400 dollars qui permet donc à un IBM d'envoyer un fichier d'impression vers l'imprimante Laserwriter ou un fichier texte vers le Macintosh. Ou, suivez mon regard, de transférer un fichier d'édition électronique de PageMaker pour IBM vers PageMaker pour Mac...

Il ne manquerait plus que de pouvoir connecter le Mac à une unité centrale pour l'implanter solidement en entreprise. Vous avez dit unité centrale ? Voici l'**AppleLine 3270 File Transfer Software**. Lui aussi développé et vendu par Apple en personne. Il permet de récupérer sans douleur les fichiers des IBM 3270.

Voici pour les données, mais pourquoi s'arrêter en chemin ? On pourrait aussi prévoir un moyen de faire tourner MS/DOS sur le Mac. Comment ? C'est déjà fait ? Hunter Systems, une société de Palo Alto aurait développé un logiciel permettant aux possesseurs de Mac équipés de processeurs 68020 (les Mac *Ouverts*) de faire tourner les logiciels MS/DOS avec un système baptisé X DOS ? Mais alors, quel apanage reste-t-il donc à IBM ?

Du rififi dans les bases de données ?

Apple finit aux États-Unis de tester un SGBD connu sous le nom de code de *Silver Surfer* qui n'est autre que le fameux 4e Dimension d'ACI (dont la dernière version est tout à fait extraordinaire). C'est que la firme à la pomme va, en effet, distribuer sous son nom le programme écrit par Laurent Ribardière.

Cette décision n'est pas du goût de tous aux États-Unis, et les éditeurs des programmes concurrents, comme Helix, font grise mine et adjurent Apple de renoncer à sa décision. Quant à Ashton Tate, l'auteur de DBase III, qui a retardé

le lancement de son SGBD, il observe les événements. La décision d'Apple n'est apparemment pas facile. Il manque, en effet au Mac un logiciel de bases de données qui soit devenu un standard, comme DBase l'a été pour le PC. Dans cette catégorie, aucun produit ne s'est imposé encore comme Excel a pu le faire pour les tableurs. Dilemme d'Apple : patronner un produit extraordinaire comme 4D et risquer de décourager tout un secteur de l'édition de logiciels, ou attendre et risquer de voir naître plusieurs standards dont aucun ne s'imposerait. Je n'aimerais pas être dans la peau de celui qui a pris la décision...

Vive l'Apple IIGS

C'est le cri que vont pouvoir lancer les actuels possesseurs d'Apple //e. Il est en effet prévu qu'Apple procède en guise d'*upgrade* à un échange standard de leur machine. Ils pourront troquer leur ordinateur favori (repris 4151 Francs) contre un GS tout neuf. Et que va faire Apple des //e ? Bonne question. Ils seront récupérés et distribués dans les écoles. D'une pierre deux coups : on fait plaisir à de bons clients et on essaie d'en recruter d'autres. Futé non ?

Communications

Revenons au Mac qui s'impose comme un fabuleux outil de communication. Frédéric Lévy d'Hello Informatique a développé sa dernière version de Mac Tell : **MacTell3**. Voici un logiciel exemplaire. Il a été le premier vrai programme à transformer un Mac en Minitel. Il ne cesse de s'améliorer. Depuis la première version, on dispose d'un langage de procédures permettant d'automatiser, par exemple, la consultation d'un serveur. Eh bien désormais, ce langage de procédures est en français. On peut par exemple écrire : «*Composer le "36 15" (arrêter l'exécution si pas de connexion)*

Pause de 10 secondes

Taper "175040781", suivi de Envoyé

Plus besoin de jargon compliqué ou de simili Basic. Ce logiciel qui permet de tout automatiser, prend toute sa valeur avec le **Modem Diapason** d'Hello Informatique. On peut alors automatiser la consultation d'un serveur à une heure donnée. On peut aussi de façon ultra simple transformer le Mac en un mini serveur accessible depuis

n'importe quel Minitel. Parmi les autres améliorations, signalons l'enregistrement automatique de toute une consultation qu'on peut ensuite rejouer en local. **MacTell3** associé au modem Diapason forment vraiment un tandem inégalé.

Langages

Le Mac accueille deux nouveaux langages. Grâce à **Coral Software**, un Logo orienté objet et baptisé bien sur **Object Logo** incorporant les raffinements du Lisp. Ce langage de programmation de haut niveau est bien tentant avec son éditeur multi fenêtres, son *dévermineur* et son compilateur. Prix : 80 dollars.

Un autre éditeur américain : **Spencer** a, lui, développé un **APL 68000** pour le Mac. Les amateurs de ce langage là (il en existe pas mal) sont servis. La aussi on bénéficie d'une interface aux routines graphiques Quickdraw du Mac. Pour 295 dollars.

Textes et Édition électronique

D'abord un traitement de texte. Un de plus. C'est celui de la firme **Word Perfect**, qui a réécrit pour le Mac son traitement de texte (l'un des plus cotés pour le PC). La firme s'est pliée aux menus qui font la gloire du Macintosh, mais le changement de caractères se fait par une icône au bas de l'écran. **Word Perfect** pour le Mac devrait être vendu 400 dollars.

Poursuivons par une petite histoire peu morale. La firme Lettraset avait acheté la toute nouvelle version de **MacPublisher** qui devait s'appeler **Lettrapage** et publié des publicités luxueuses sur ce produit. Comme ce programme n'était pas prêt, elle s'est rabattu sur la dernière version de **Ready Set Go**, la 3.0, dont elle a acheté les droits. Conséquence pratique pour les utilisateurs : jusqu'à la fin mars, **Ready Set Go** est vendu aux États-Unis par son concepteur **Manhattan Graphics** pour 295 dollars. Ensuite, simplement rebaptisé **LettraPage**, il en coûtera 100 de plus. Aucune crainte de ce genre en France. C'est la très dynamique société **BIP** qui continue à importer, traduire (en deux semaines) et vendre **Ready Set Go 3**.

Rappelons que **Ready Set Go** version 3 n'a plus rien à voir avec les versions précédentes, qu'il permet de créer ou manipuler des objets, est doté d'un traitement de texte sophistiqué permettant de lier des textes pour le faire se déverser aux endroits prévus, permettant aussi le crénage (contrôle de l'espacement des lettres), la césure automatique, et la correction orthographique, grâce à un module qui sera développé avec l'université de Compiègne. **BIP** qui importe aussi le programme graphique **Cricket Draw**, ainsi que divers utilitaires, comme **Acta**, un traitement d'idées incorporé dans le menu Pomme vend le tout sous le nom d'**Edit 2000**. Un ensemble d'édition de bureau très complet qui pourrait faire quelque mal au quasi-monopole que s'était assuré **PageMaker**.

C'est encore **BIP**, qui importe **Guide**. **Guide** est un "Hypertexte". C'est à dire un système permettant d'accéder à des strates d'informations. Imaginons un dessin d'un moteur. On clique sur le couvercle cylindre, et dans une fenêtre s'inscrit sa référence. On peut encore cliquer dans cette référence pour voir par exemple apparaître le prix. Et ainsi de suite. On peut établir des systèmes de référence, des chronologies, etc. Ces systèmes sont particulièrement utiles pour retrouver des informations stockées par exemple sur un disque dur ou un de ces CD ROM, ces Compact Discs pour ordinateurs qui vont commencer à se répandre. L'hypertexte est donc un système électronique permettant de lier, référencer, retrouver des informations. Il n'est handicapé que par un traitement de texte insuffisant et un format non standardisé.

Memorandum de **Target Software** est un de ces logiciels qui vont faire un malheur. Il permet de réaliser des notes (comme ces notes adhésives qu'on colle partout) et de les relier à une cellule d'une feuille de calcul. Ce logiciel s'installe en accessoire de bureau dans le menu Pomme coûte 100 dollars et fonctionne avec Excel, Multiplan, Jazz ou Works.

À remarquer encore un logiciel d'édition électronique c'est **Xpress** de **Quark**. Il inclut bien sûr (dans sa version américaine) un correcteur orthogra-

phique de 80000 mots, la césure automatique, etc. Particulièrement remarquable la possibilité de faire se répandre automatiquement le texte autour d'un dessin aux formes irrégulières. Prix : 695 dollars.

Une fois un texte ou une publication réalisé, reste encore à l'imprimer. Un accessoire particulièrement utile permet d'opérer en tâche de fond, sans mobiliser le Macintosh, c'est le Spooler. **SuperMac Software** a développé **Superspool**, un formidable utilitaire permettant d'enregistrer un fichier d'impression sur le disque et de l'imprimer tout en continuant à travailler sur le Macintosh. Un accessoire qui change la vie des utilisateurs pour 60 dollars. **Laser Superspool** permet de réaliser la même opération avec une imprimante Laser et coûte 150 dollars dans la version mono-utilisateur et 395 dollars dans la version multi-utilisateurs (jusqu'à 5).

Un autre accessoire (cette fois matériel) permet aussi d'imprimer avec la Laser sans immobiliser le Mac, c'est le **Mac Buffer d'Ergotron**. Une petite boîte qui s'insère entre le Mac et la Laserwriter et permet de stocker 1 ou 2 Mégas de texte.

Outils graphiques

Ces programmes permettant au Mac d'exploiter toutes ses possibilités de calculs graphiques l'imposent dans les milieux les plus divers. Voici ainsi **MacSpin** vendu Par **Bruno Rives et Associés**.

C'est un logiciel d'analyse graphique des données multidimensionnelles permettant de découvrir des associations, des non linéarités, d'animer et visualiser un graphique et données en 3D. Son grand avantage est de pouvoir intégrer des informations catégorique ou du texte dans un graphique. Un outil utile dans le marketing, l'ingénierie ou la physique des particules, permettant de traiter des informations d'astronomies, de géophysique, etc.

Et l'Apple // ?

Les Apple IIGS commencent à être livrés et le II va son train. On attend pour avril deux nouveautés de taille destinées à lui permettre l'accès au

monde MS/DOS. **Dos Boot** développé par **Orange Micro** est une petite boîte contenant un lecteur de disquettes et une partie des composants d'un compatible IBM PC permettant au IIG d'utiliser ses programmes.

Applied Engineering a préféré étudier une carte permettant d'écrire et de lire sur un lecteur de disquettes indifféremment en format IBM ou Apple. La carte comprend aussi un processeur 8088 (celui des compatibles) mais cadencé à 8 Mhz. Tournant donc plus rapidement que celui de l'IBM. Je sais que de nombreux lecteurs de Pom's sont intéressés par cette possible compatibilité et je ne manquerai pas de leur donner dès que possible des informations supplémentaires sur ces produits.

En attendant, **AST Research** a déjà sauté dans le train de l'édition électronique et étudié toute une série de produits pour le IIG. Un système de numérisation permettant de transférer les images prises en vidéo et de les travailler avec **AST Vision Effects IIG** (dommage que ça ne fonctionne qu'avec le système couleur NTSC). Des disques durs, une carte de mémoires Ram, etc. **Applied Engineering** vend la feuille de calculs électronique **VIP professional** (rapide, puissante mais exigeant de la mémoire) avec une carte de mémoire pour 250 dollars. Quant à **Maxx de Icon Incorporated**, c'est tout simplement un volant d'avion qui s'installe dans la prise de jeux et permet d'utiliser de façon encore plus réaliste les simulateurs de vol. Prix 130 dollars.

Des programmes en ruban

Le **Softstrip** — déjà en vente depuis quelques mois aux États-Unis — permet de stocker des données sous la forme d'un ruban imprimé en noir et blanc. Un lecteur qu'on branche sur le Macintosh permet de retransformer ces données en fichiers ou programmes. Déjà, plusieurs revues d'informatiques publient leurs programmes sous cette forme aux États-Unis. Le lecteur permet en quelque 30 secondes de lire ces bandes et de récupérer le programme dans l'ordinateur sans avoir à les taper au clavier. Le prix de ce système très remarquable est de 2200 Francs. La firme qui le produit, **Cauzin**, s'est

alliée avec **Kodak** pour le distribuer dans le monde. Sera-ce un jour un nouveau standard ?

Un souhait

La période des déclarations d'impôts s'est une fois de plus écoulée sans qu'un éditeur ne propose un système complet permettant de déclarer et calculer ses impôts sur ordinateur. Ce marché est pourtant prospère aux États-Unis. Et nos Mac français alors ?

Adresses

Think Technologies
420 Bedford St, Lexington
MA 02173

Hello Informatique
1, rue de Metz
75010 Paris - Tél. : 45 23 30 34

Coral Software
Tél. : 617 868 7440 (aux
États-Unis)

Spencer Organization
PO Box 248 Westwood
NJ 07675

BIP
13, rue Duc
75018 Paris - Tél. : 42 55 44 63

SuperMac Software
950 N.Rengstorff Av
Mountain View CA 94043

Ergotron
PO Box 17013 Minneapolis,
MN 55417

ACI
6, avenue Franklin Roosevelt
75008 Paris - Tél. : 43 59 89 55

Bruno Rives et associés
6, avenue Franklin Roosevelt
75008 Paris - Tél. : 42 89 02 36

Target Software
Tél. : aux États-Unis :
(305) 252 0892

Applied Engineering
PO Box 798 Carrollton
TX 75006

Orange Micro
1400 n.Lakeview
Av. Anaheim CA 92807

AST Research
2121 Alton Av. Irvine CA 92714

Icon Incorporated
1611 116th Av NE Bellevue
WA 98004



Un nouveau produit Pom's : BananaSoft

Pom's vous propose un nouvel utilitaire qui simplifiera les rapports houleux qui existent souvent entre le programmeur et l'Applesoft. E.P.E. nous donnait un éditeur digne de ce nom ; voici maintenant une amélioration des possibilités du Basic ainsi que la correction de 'bugs'.

Le but

Les principaux objets de Bananasoft sont :

- réconcilier la carte 80 colonnes et le Basic (problèmes du GET, du HTAB...);
- aider le Basic dans son travail sur les chaînes (nouvelles fonctions, ramassage des poubelles 15 fois plus rapide) ;
- permettre la saisie au clavier sans attendre la disponibilité du Basic : il s'agit d'un véritable 'buffer clavier' de 32 caractères.

On se rapprochera aussi du Basic Microsoft 5.x par des fonctions telles SWAP, ERASE, LINE INPUT sans pour autant encombrer la mémoire (moins de 4Ko).

Quelle configuration ?

Bananasoft fonctionne sans problème sur Apple II+ (avec 48 Ko et ROM autostart), II/e, II/e 65C02 et II/c. Bien entendu, Bananasoft est à même de tirer parti de la présence éventuelle d'une carte langage.

Bananasoft reconnaît certaines cartes 80 colonnes :

- Apple Text card et Extended 80 col. pour l'Apple II/e ;
- Carte compatible (Eve Chat Mauve par exemple) ;
- Circuiterie interne de l'Apple II/c ;
- Carte 80 col. Videx Videoterm pour Apple 2+ (firmware 2.4 : si votre carte reconnaît les séquences ESC I, J, K, M au clavier, elle conviendra).

Le système d'exploitation supporté est le DOS 3.3 en version 48 Ko uniquement. Bananasoft lui apporte bien sûr quelques raffinements comme la possibilité d'entrer ses commandes en mode minuscules.

Tout programme Applesoft peut être exécuté sans modification sous Bananasoft, si l'une au moins de ces deux conditions est vérifiée :

- le buffer clavier est désactivé durant l'exécution dudit programme ;
- le programme n'accède au clavier que par les ordres GET, INPUT ou LINE INPUT (pas de PEEK/POKE/WAIT).

Les modifications à apporter aux programmes ne s'inscrivent pas dans ce cas là sont explicitées dans la documentation. Notons l'existence de deux nouvelles commandes DOS : INSTALL et INHIBIT gérant la réception de caractères dans le buffer, deux outils puissants pour résoudre le problème précité.

Bananasoft se reloge de lui-même entre le DOS et les buffers et est de cette façon totalement protégé d'une modification du nombre de ceux-ci (par la commande 'MAXFILES').

Les possibilités

Les améliorations portent sur 4 points :

Amélioration des fonctions du DOS

- Minuscules dans les ordres DOS. Ainsi :

```
]bsave TITL,a$2000,l$1fff
```

est acceptée par le DOS qui effectue une conversion minuscules/MAJUSCULES sur toute la ligne, exceptée les dénominations de fichiers.

- une nouvelle option est fournie pour l'ordre BSAVE, l'option 'E' (comme END).

```
]BSAVE TITL,A$2000,E$3FFF
```

est équivalent à :

```
]BSAVE TITL,A$2000,L$2000
```

naturellement, les 2 options 'E' et 'L' s'excluent mutuellement.

- Un nouvel ordre est apporté au système : il s'agit de 'L' (prononcer SMART RUN). Les utilisateurs de ProDOS connaissent déjà cet ordre ; il s'agit d'"exécuter" un fichier quel que soit son type. SMART RUN sera équivalent à un EXEC si le fichier ouvert est du type texte, à RUN si celui-ci est un programme BASIC Integer ou Applesoft, ou à BRUN s'il s'agit d'un fichier binaire.

- Les commandes INSTALL et INHIBIT ont été ajoutées au DOS afin de simplifier l'exploitation du buffer clavier.

- La commande FP ne déconnecte pas Bananasoft dans le cas où le langage courant était l'Applesoft mais le réinstalle si le langage était l'Integer.

- Et enfin, *the last but not least*, la commande INIT conserve son fonctionnement normal : elle installera sur disquette une version standard du DOS.

- Dernier petit détail, Bananasoft permet aux possesseurs d'Apple II+ (les pionniers !) de ne pas tomber en syncope devant des écrans "francophones", un mode de visualisation permet à ceux-ci :

- une conversion minuscules/MAJUSCULES des caractères sortant sur leur écrans
- une conversion des caractères 'èèàùç' en 'eeauc'...

Amélioration des fonctions standard d'Applesoft

- Les intructions BASIC 'PRE' et 'INE' ne déconnectent plus le DOS. Ainsi :

```
10 PRE 1
```

est devenu équivalent à :

```
10 PRINT CHR$(4);"PRE1"
```

- Un ramassage des poubelles ("garbage collection") est intégré dans Bananasoft, totalement transparent, il vous apporte une multiplication de la vitesse d'exécution des intructions portant sur les chaînes de caractères par un facteur supérieur à 2... Bananasoft reloge de lui-même une version améliorée de FRE(16) sur la carte langage si celle-ci a été détectée lors du boot, libérant ainsi plus de deux pages (1 page = 256 octets) pour le Basic. À noter que ce chargement ne perturbe en rien le fonctionnement du langage BASIC Integer si celui-ci est lui aussi chargé en RAM.

- Toutes les instructions Applesoft de positionnement du curseur (HTAB ainsi que le séparateur ',') dans l'ordre

PRINT) sont désormais compatibles avec les 80 colonnes à l'écran.

- L'instruction GET ne détectait pas sous Applesoft la frappe de la touche ESC lorsque les programmes de gestion 80 colonnes (anciennes ROM) de l'Apple //e était activé. Bananasoft résout ce problème.

- Les instructions HOME et HTAB sur Apple][+ ne fonctionnaient pas, ou mal, lorsque le périphérique de sortie était la carte 80 colonnes, ces petites misères ne sont plus maintenant qu'un mauvais souvenir...

- La fonction POS() est opérante maintenant également en 80 colonnes.

Nouveaux mots-clé au vocabulaire d'Applesoft

- **RESTORE** <n°-de-ligne>

Même signification que pour le Basic Microsoft 5.x, repositionne le pointeur de DATA au numéro de ligne spécifié, numéro qui est optionnel.

- **SWAP** nom-var, nom-var

Assure l'échange des valeurs prises par les deux variables spécifiées. Celles-ci doivent être du même type (entier, réel ou chaîne) sinon une erreur du type TYPE MISMATCH est signalée. Ex : SWAP A\$,FF\$

- **ERASE** nom-de-tableau

efface de la mémoire centrale le tableau de variable dont le nom est spécifié. Ex : ERASE G\$.

- **LINE INPUT** <constante-littérale;>nom-var.

Permet d'entrer toute séquence de touches frappée au clavier et de la stocker dans une variable chaîne de caractères. Les seuls caractères de contrôle étant interceptés par le programme sont RETURN pour valider, et flèches gauche et droite pour corriger. Ex :

```
LINE INPUT "init. imprimante ? ",II$
```

Ceci permet de saisir directement au clavier ESC-e par exemple pour faire passer l'imprimante en caractères Élite.

- ***LIST**<n°-de-ligne><><n°-de-ligne>

La syntaxe est identique à celle de l'instruction LIST de l'Applesoft. La différence tient dans le format d'affichage des mots-clé : ceux-ci ne se voient pas systématiquement entourés par des espaces. Ex :

```
*LIST 1009,1080
```

- **INSTR**(<indice de départ,>chaîne,chaîne)

Cette nouvelle fonction offerte par Bananasoft cherche la chaîne B\$ à l'intérieur de la chaîne A\$ en partant du caractère N, et retourne la position de la chaîne B\$ dans P.

```
P=INSTR(N,A$,B$)
```

Exemple :

```
A=INSTR(5,"SALUT LA COMPAGNIE! SALUT!","SALUT")
```

retournera la valeur 21 dans A, le premier SALUT après le cinquième caractère apparaissant en position 21.

- **STRING\$(N,C)**

Retourne une chaîne de N caractères C. Si C est une expression alphanumérique, le caractère utilisé sera le premier caractère résultant de l'expression. Exemple :

```
A$=STRING$(4,"01234")
```

retournera la chaîne "0000" dans A\$, A\$=STRING\$(4,48) retourne aussi "0000" dans A\$.

- Pour ceux pratiquant une programmation plus proche de la machine, Bananasoft leur permet des POKE et des PEEK sur

des mots de 16 bits (2 octets consécutifs de la mémoire) : les nouveaux termes seront respectivement :

DOKE adresse,mot_16 bits (adresse < poids faible - adresse+1 < poids fort)

DEEK(adresse) renvoie le mot_16 bits contenu dans adresse. Ex :

```
DOKE 1780,DEEK(&H300)
```

- Les constantes hexadécimales sont admises dans toute expression arithmétique : &HFF équivaut à 255. Exemple :
AC = &HFC58 : CALL AC

- Il est désormais possible de connaître l'adresse d'une variable en mémoire, ceci par emploi de la fonction **VARPTR** :

```
VARPTR(nom_variable)
```

renverra l'adresse où se trouve stockée la valeur d'une variable numérique ou le descripteur d'une variable chaîne de caractères. Une technique d'emploi de routine en langage machine dans un programme BASIC pourrait alors être celle-ci :

```
CALL DEEK(VARPTR(A$)+1)
```

où la routine relogeable se trouverait être dans la variable A\$. À noter que cette méthode est aussi populaire sur certaines machines (Macintosh, TRS80) que la routine SHLAM sur Apple //.

Le buffer clavier

À la mise en route du système, la réception des caractères est inhibée et la gestion clavier est normale ; c'est la commande DOS INSTALL qui autorise le buffer à fonctionner... La réception des caractères est lancée durant :

- tout appel à CHRGET/ CHRGOT en page zéro ;
- tout appel à COUT passant par le relais du DOS ;
- tout appel à RDKEY passant par le relais du DOS et n'appelant pas un périphérique clavier ;
- tout appel à RWTS.

Les caractères tapés au clavier seront stockés en attendant la fin du travail en cours (anticipation lors de calculs lents par exemple).

Lorsque le buffer clavier est activé, la frappe de certaines touches provoque divers processus.

- La frappe de <CTRL><X> provoque l'initialisation complète du buffer.

- La frappe de <CTRL><C>, outre qu'elle provoque elle aussi l'initialisation du contenu du buffer, n'est pas pris en compte par lui et le code reste dans l'adresse tampon de l'Apple (\$C000).

- Lorsque le périphérique de sortie est l'écran, la frappe de <CTRL><S> n'est pas pris en compte et son code reste dans l'adresse tampon de l'Apple. La fonction de cette touche (le "gel" de l'écran) est alors assuré par le driver vidéo.

L'adaptation (simple) de vos programmes au bénéfice de ce buffer est discuté en détail dans la documentation et fait l'objet d'un exemple sur la disquette.

La disquette

La disquette 5,25 pouces au format DOS 3.3, comporte les sources, l'objet et une démonstration. Des informations à destination des programmeurs sont apportées dans la documentation jointe.

Franco 200,00 F TTC, port avion hors CEE 15,00 F, bon de commande page 74.

Courrier des Lecteurs

Les slots du IIGS

Sur mon futur GS, pourrais-je utiliser simultanément les ports intégrés et mes cartes interfaces ?

M. P. Mykiéta, 92 Gennevilliers

Oui et non : un tableau de bord (accessible par CTRL-OPTION-ESC ou CTRL-OPTION-RESET) vous donne pour chacun des slots le choix entre votre carte ou celle qui est intégrée, par exemple en slot 4, votre interface ou la carte souris. À vous de choisir...

Minitel et Apple(s) ??

Que pensez de votre n° 27 sur la communication ? Le montage que vous proposez pour la liaison est séduisant mais le support technique d'un club me dit qu'il faut l'éviter, plusieurs personnes ayant claqué leur micro. S'il est aussi simple, pourquoi proposez-vous le câble à 225,00 F alors qu'on le trouve parfois à 900,00 F ? Un informaticien qui commercialise les liaisons PC-Minitel me dit que seul le modem Sectrad 300 convient et que la liaison //c Minitel est impossible.

M. Raymond, 77210 Samoreau

Rassurez-vous, la liaison proposée dans les Pom's 27 et 28 fonctionne sans souci, telle que décrite, chez nombre de nos lecteurs et nous pratiquons ici intensivement la transmission de fichiers à l'aide d'InterPom's.

Les conseils de tel ou tel ? Que dire si ce n'est de faire confiance aux interlocuteurs qui ont la pratique de la communication ? Le modem 1200 bauds du Minitel ne fatiguera par plus le //c qu'un modem 300 bauds (votre Apple transmet ses données à l'Image Writer à 9600 bauds...).

Le prix ? Nous ne saurions rien expliquer mais précisons que les câbles à 900,00 Frs sont

fréquemment accompagnés d'un logiciel d'émulation Minitel.

Enfin, en cas de difficultés de mise en œuvre des programmes Pom's, nous vous répondrons téléphoniquement :
(1) 39 51 24 43.

Le moniteur étendu...

...de T. Le Tallec (Recueil Pom's n°2) fonctionne normalement sur l'Apple //c après avoir apporté les modifications suivantes.

Dans le source :

L 71 CHAR1 = \$F9BA (F9B4)

L 72 CHAR2 = \$F9B4 (F9BA)

L 75 RTBL = \$FAD2 (FB19)

Dans l'objet :

9141:D7 (1E)

9142:F9 (FA)

93A7:BA (B4)

93AF:B4 (BA)

93B4:B4 (BA)

(entre parenthèses figurent les anciennes valeurs)

M. P. Adang, 52110 Blaiserives

Minitels face à face

Comment, par programme, faire que deux Minitels puissent se reconnaître après connexion ? Après une connexion manuelle, il n'y a pas de porteuse sur la ligne.

C. Laron, 13100 Aix.

Il est nécessaire que l'un des deux Minitels soit retournable, c'est-à-dire qu'il sache émettre à 1200 bauds et recevoir à 75 au lieu de l'inverse en fonctionnement sur serveur.

Vous pouvez vous inspirer des programmes "InterPom's" du numéro 28 :

Côté Minitel retournable, envoyer sur la prise péri-informatique les caractères suivants : 27, 57, 104, 27, 57, 111 ce qui, outre le retournement, provoque la connexion et l'émission d'une porteuse à 1700 Hz (± 400) qui

déclenchera la porteuse à 420 Hz (± 30) du Minitel de votre correspondant.

Prochainement

Pom's vous proposera de nouveaux programmes d'enregistrement et d'exploitation des pages Minitel pour Apple // et Macintosh. Écrits en assembleur, ils seront dotés de puissantes fonctions de filtrage et de récupération de fichiers, ils fonctionneront sur toutes les machines et le port communication du IIGS sera exploité. Les suggestions sont les bienvenues...

Dans les prochains numéros,

- ☛ un programme de gestion d'un écran virtuel,
- ☛ des courbes récursives en Pascal,
- ☛ un numéroteur de fichiers ProDOS,
- ☛ DhgrTools pour manipuler les pages graphiques,
- ☛ un programme de tracé de courbes,
- ☛ un jeu de réflexes et de réflexion,
- ☛ des essais hard et soft,
- ☛ des éléments pointus sur les nouveaux matériels...

Pour le Macintosh,

- ☛ des accessoires de bureau,
- ☛ des utilitaires,
- ☛ des applications autonomes de la veine d'InterPom's...

Bon de commande

Disquettes

BASICIUM.....	140 Ko (cf. Pom's n° 13)	à 200,00 F
E.P.E. 5.1.....	800 Ko ou 140 Ko (cf. Pom's n° 29)	à 200,00 F
Échange E.P.E. 5.1.....	800 Ko ou 140 Ko (cf. Pom's n° 29)	à 80,00 F
PASCAL.....	140 Ko (cf. Pom's n° 15)	à 80,00 F
MAX (Moniteur étendu).....	140 Ko (cf. Pom's n° 18)	à 150,00 F
DOMINOS.....	140 Ko (cf. Pom's n° 19)	à 80,00 F
COGO.....	140 Ko (cf. Pom's n° 21)	à 200,00 F
LUDOLOGIC.....	140 Ko (cf. Pom's n° 25)	à 80,00 F
ORDICO.....	140 Ko (cf. Pom's n° 26)	à 200,00 F
BANANASOFT.....	140 Ko (cf. Pom's n° 29)	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

Revue, disquettes

Revue 4 7 8	à 35,00 F
Revue 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	à 40,00 F
Revue 27 28 29	à 45,00 F
Disquettes Apple // 140 Ko 5,25 pouces			
1/2 3 4 5 6 7 8 9 10 11 12 13 14 15	à 60,00 F
16 17 18 19 20 21 22 23 24 25 26 27 28 29		
Disquettes Apple // 800 Ko 3,5 pouces			
29	à 80,00 F
Disquettes Macintosh			
14/15/16 groupées	à 150,00 F
17 18 19 20 21 22 23 24 25 26 27 28 29	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 225,00 F
Abonnement revues + disquettes Apple // 140 Ko 5,25'	à 525,00 F
Abonnement revues + disquettes Apple // 800 Ko 3,5'	à 625,00 F
Abonnement revues + disquettes Macintosh	à 625,00 F

Reliures toilées

Pour 6 numéros, un an de Pom's

Supplément avion hors CEE : ajoutez 15,00F par numéro et/ou disquette **Total TTC :**

Envoyez ce bon et votre règlement à : ÉDITIONS MEV - 12, rue d'Anjou 78000 VERSAILLES

Nom :

Adresse :

Carte bleue/VISA

n° de la carte : -----
 date d'expiration : -----
 montant : _____ F

signature : _____

Bon de commande 'Communication'

Pom's 26 : Apple II, Carte SuperSérie & CP/M. Programme permettant de configurer simplement le SSC pour CP/M

Pom's 27 : Apple II, Carte SuperSérie & CP/M 2ème partie. La communication selon le protocole EMUDISK.
Apple II & Minitel. Programme d'enregistrement et de restitution à lair des écrous Minitel (par l'interface Série).
Macintosh & Minitel. Programme d'enregistrement et de restitution à lair des écrous Minitel.

Pom's 28 : InterPom's Apple II et InterPom's Macintosh pour transmettre des fichiers par Minitel.

Je désire recevoir :	Câble Minitel/Apple II+, IIe, IIe+, IIgs avec SSC*	à	225,00
	Câble Minitel/Apple IIc	à	225,00
	Câble Minitel/Macintosh 128, 512	à	225,00
	Câble Minitel/Macintosh Plus	à	225,00

* les programmes des numéros 26 à 28 ne fonctionnent sur le IIgs qu'équipé d'une carte SuperSérie Apple. Pour relier votre IIgs au Minitel en utilisant le port intégré, commander un câble Macintosh Plus.

Total TTC

Supplément avion hors CEE : ajoutez 15,00F

Envoyez ce bon et votre règlement à : ÉDITIONS MEV - 12, rue d'Anjou 78000 VERSAILLES

Nom : _____

Adresse : _____

Carte bleue/VISA

n° de la carte : _____
date d'expiration : _____
montant : _____ F

signature : _____

Comme à l'accoutumée, la disquette Pom's "Mac 29" contient les fichiers et programme - exécutable et sources - publiés dans le cahier Macintosh de ce numéro. En plus, nous avons ajouté 5 accessoires de bureau :

- **dCAD Calculator**, une calculatrice scientifique à mémoire ;
- **Hex Calculator**, une calculatrice décimale/hexadécimale appréciée des programmeurs ;
- **Word Count**, pour connaître le nombre de mots, caractères et paragraphe dans un document ;
- **Telegraph** transforme en code morse visuel et sonore tout ce qui est frappé au clavier, ou collé ;
- **Extras** permet de connaître à tout moment la place disponible sur le disque, en mémoire, de renommer un fichier...

et une très intéressante application - **DA/Test** - qui permet d'essayer ou utiliser des accessoires de bureau sans les installer dans le menu . De fait, il évite aussi la saturation dudit menu avec des accessoires d'usage peu courant.

Disquette Pom's "Macintosh 29"
80,00 Frs - Bon de commande ci-contre.

Disquette Pom's "Macintosh 29"

