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

pom's

- Du graphique en Turbo Pascal sous CP/M
- Routines binaires et carte langage
- Le Jeu de la Vie sur le Macintosh
- Le //c composeur téléphonique
- Vos listings Basic formatés
- Routine de saisie sur Mac
- Vos impôts sur Multiplan
- Désassembleur 65C02
- Rubrique MacAstuces
- Marche arrière Basic
- Patches au DOS 3.3
- Une UNIT Pascal
- SoftCopie ProDOS
- Un décruncher

NUMERO 23 - PRIX 40 F

M2366-23-40 F

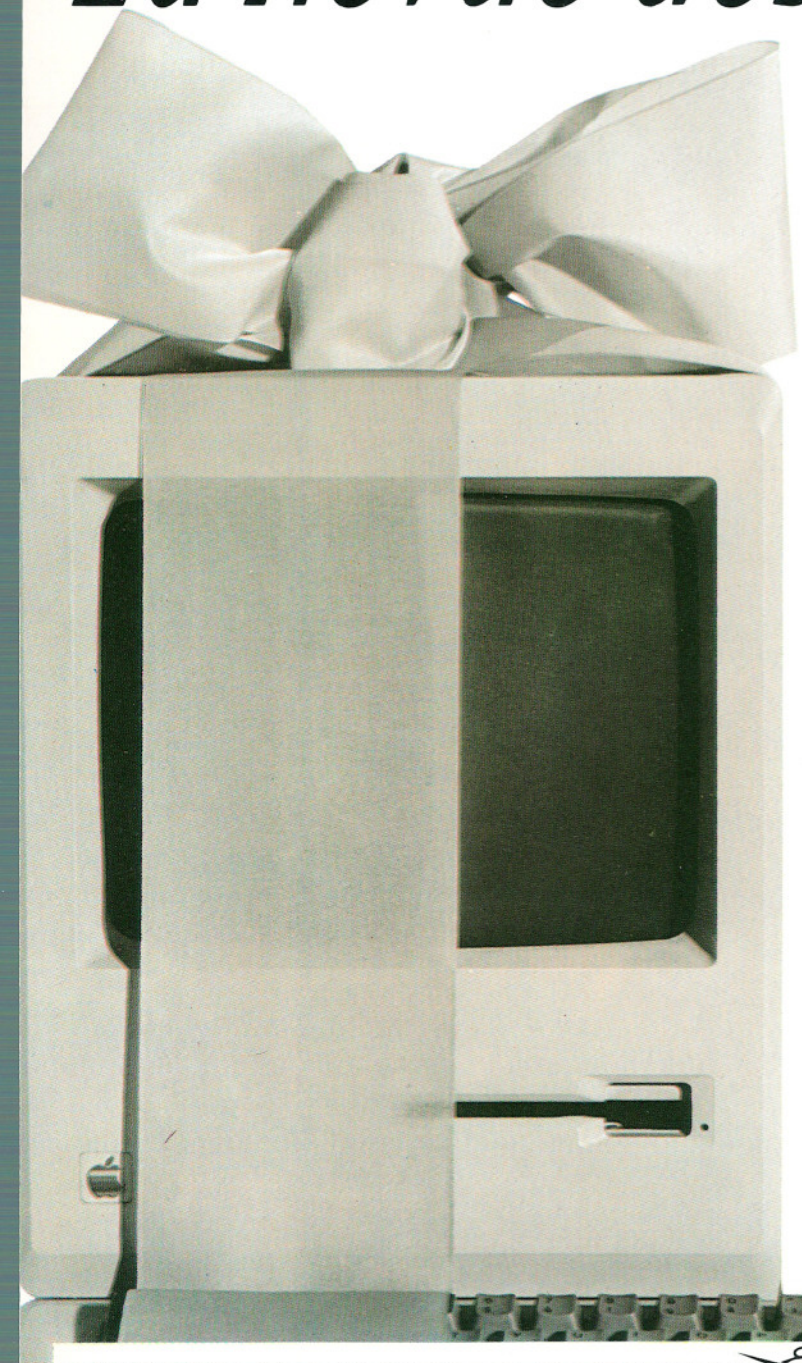
ISSN : 0294-6068

EXCEPTIONNEL ! le guide des 600 logiciels et périphériques

infomac

La Revue des Macintosh

ISSN 0766-4915



infomac
La Revue des Macintosh
5, PLACE DU COLONEL-FABIEN - 75491 PARIS CEDEX 10 - TÉLÉPHONE : (1) 42.40.22.01

- EXCLUSIF : le nouveau Mac !
- MUSCLÉ : les disques durs
- PRO : la micro édition

BON DE COMMANDE

à retourner à INFOMAG, 5 place du Colonel Fabien, 75491 Paris cedex 10

OUI je désire m'abonner pour 7 numéros à INFOMAG au prix de 215 F.
- Je désire recevoir, au prix de 35 F l'un : ☐ le numéro 1 ☐ le numéro 2 ☐ le numéro 3

Je vous adresse mon règlement par chèque bancaire à l'ordre de INFOMAG

NOM _____ PRENOM _____

ADRESSE _____

35 F DÉCEMBRE 85 N° 3

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

Éditorial

Ce numéro de printemps, outre ses treize programmes, vous présente deux nouveautés : le recueil n°3 d'abord, qui regroupe les numéros 9 à 12 de Pom's. Il vous séduira particulièrement pour ses listings sortis sur la LaserWriter (toujours la communication entre Apple // et Mac !). Une soixantaine de programmes et plus de 200 pages : une mine de renseignements à portée de main.
Deuxième nouveauté : E.P.E. 5.0. Nombre d'entre vous avait en effet pris la plume pour déplorer l'impossibilité d'éditer avec E.P.E. sous ProDOS en 80 colonnes. Dès aujourd'hui, cette nouvelle version est disponible, et elle fonctionnera également sur votre //c.
Vous n'avez pas E.P.E. ? au prix de 200,00 F, c'est un éditeur sans égal.
Vous avez E.P.E. ? retournez-nous la disquette originale, nous la mettrons à niveau au prix de 80,00 F.

Depuis plus de deux années (Pom's n° 9), vous trouviez les disquettes d'accompagnement Apple // à 55,00 F ou à 280,00 F par abonnement : elles resteront sans concurrence, mais nous devons avouer que nous étudions une 'remise à jour' de ces prix. Ne tardez pas à renouveler votre abonnement !

Pascal et Turbo Pascal, CP/M, DOS et ProDOS, assembleur et Basic, 6502, 65C02 et 68000 : vous trouverez certainement ce numéro très riche ; vos contributions nous montrent que vous êtes particulièrement actifs devant vos claviers. Dans le numéro 24 probablement, vous trouverez le très beau fruit de cette activité sous la forme d'un programme en CP/M, CP/M que nous avions quelque peu délaissé.

Nous vous donnons rendez-vous au ***Sicob Boutique****, du 14 au 19 avril, stand* ***2A203*** *et n'oubliez pas notre petit questionnaire page 73 !*

Hervé Thiriez

Ont collaboré à ce numéro : Jean-Paul Arbel, Alexandre Avrane, Jean-Luc Bazanegue, Dominique Bernardi, Gilles Caraux, Gaëtan Dagron, Delphine Declercq, Alexandre Duback, Bruno Fénard, Sylvie Gallet, Jean-Michel Gourévitch, Olivier Herz, Jacques Honorez, Bernard Hoyez, Yvan Koenig, Niels Køge, Jean-Claude Lengrand, Gérard Michel, Nicolas Montsarrat, Jean-Luc Nail, Patrice Neveu, Christian Piard, Hervé Thiriez.
Directeur de la publication, rédacteur en chef : Hervé Thiriez.
Rédacteurs : Alexandre Avrane, Olivier Herz.
Siège social : Editions MEV - 12, rue d'Anjou - 78000 Versailles. Tél. : (1) 39.51.24.43.
Publicité : Editions MEV.
Diffusion : N.M.P.P.
Impression : Rosay - 47, avenue de Paris - 94300 Vincennes. Tél. : (1) 43.28.18.63.

TurboPascal

Une approche, Une application

Nicolas Montsarrat

Commercialisé en France par FRACIEL - 42, rue des Prébendes - 37000 TOURS au prix de 625,00 francs HT (soit 741,25 francs TTC) dans sa version CP/M, le TURBO-PASCAL est un des meilleurs compilateurs Pascal disponibles sur Apple][+, //e et //c. Le système comporte le compilateur TURBO.COM résidant en mémoire centrale, avec éditeur incorporé, le programme d'installation TINST.COM, TLIST.COM qui liste les fichiers-source, ainsi qu'un petit tableur de démonstration nommé "Microcalc".

Le système TURBO-PASCAL

Après avoir appelé TURBO .COM, le programme propose un menu permettant d'éditer un fichier-source, de le compiler, de l'exécuter, de le sauver. Il propose quelques commandes de manipulation des disquettes (catalogue, appel d'un programme extérieur, etc.), et des commandes diverses (compilation en mémoire centrale ou sur disquette, adresse de départ du programme objet,...).

L'Éditeur

L'éditeur reprend la majorité des commandes de WORDSTAR, le traitement de texte bien connu tournant sous CP/M. Outre les commandes de contrôle du curseur, on y trouve des commandes de déplacements et de copie de blocs de texte, de recherche et de remplacement, ainsi que de mise en place du système de tabulation automatique.

La place disponible pour les textes est d'environ 22 Ko sous CP/M 60Ko et de 18 Ko sous CP/M 56Ko. C'est largement suffisant pour les programmes de taille moyenne, d'autant plus que le compilateur permet de compiler des programmes dont le source est composé de plusieurs fichiers ("include files")

Cet éditeur est pratique à utiliser, rapide et puissant. Un petit défaut toutefois : pour les possesseurs d'Apple][+, il y a des problèmes de reconnaissance de la vidéo inverse sur certaines cartes 80 colonnes destinées au][+ (Par exemple la carte U-Term), en particulier le curseur n'est pas visible à l'écran, ce qui n'est pas sans poser quelques problèmes... Pour les cartes 80 colonnes des //e et //c, aucun problème.

Le Compilateur

Le compilateur, résident en mémoire centrale, est sûrement le compilateur Pascal le plus rapide disponible sur Apple : Il compile les programmes en une seule passe, directement en code machine (et non en code-P comme le Pascal UCSD). Deux options sont proposées : la compilation en mémoire, où le code et le source résident en mémoire, et la compilation sur fichier COM, où seul le source est en mémoire. Cette deuxième possibilité permet de compiler des programmes dont le source est trop important pour laisser de la place au code, ou simplement pour obtenir un fichier exécutable indépendant du système TURBO-PASCAL.

La vitesse d'exécution des fichiers objet est également impressionante : un petit peu plus d'une seconde pour une boucle de 1 à 10000, contre 8 pour le Pascal UCSD, bien que le code généré soit du code Z80, plus lent que l'équivalent 6502.

En conclusion, un compilateur simple à utiliser, dont les performances sont inégalées sur Apple.

Le langage Pascal proprement dit

Le TURBO-PASCAL reprend toutes les caractéristiques du Pascal standard, plus quelques extensions :

Le type BYTE permet de définir des nombres entiers entre 0 et 255, n'occupant qu'un seul octet en mémoire. Les conversions en INTEGER se font automatiquement, sans générer d'erreurs.

Le type REAL définit les nombres réel entre 1^{-38} et 1^{+38} sur une mantisse de 11 chiffres, on est loin des maigres 6 chiffres du Pascal UCSD.

Les chaînes de caractères sont présentes, avec le type STRING et les instructions DELETE, INSERT, STR, VAL et les fonctions COPY, CONCAT, LENGTH, POS.

Les types énumérés (par exemple VAR HUMAIN : [HOMME, FEMME]) sont utilisables, contrairement au Pascal UCSD, sans écrire de routines de conversion : ainsi on peut écrire WRITELN (HUMAIN) et on aura à l'écran "HOMME" ou "FEMME" en toutes lettres. De même pour READLN (HUMAIN), qui attend les caractères HOMME ou FEMME.

Les "TYPED CONSTANTS" permettent de forcer le type d'une constante. Ainsi on peut définir CONST CR : CHAR = ^M qui définit le caractère 'Retour chariot'. On peut utiliser ces constantes comme paramètres de procédures de fonctions, contrairement à certaines constantes normales. On peut de même les utiliser pour initialiser des tableaux de constantes, permettant de simuler les READ/DATA du basic : on peut définir CONST DEUX : ARRAY [0..7] OF BYTE = (1,2,4,8,16,32,64,128), et manipuler DEUX comme un tableau (voir programme GRAPHIC).

La gestion des fichiers est comparable à la gestion UCSD, avec fichiers séquentiels et fichiers directs. Comme en Pascal UCSD, on accède aux périphériques par des fichiers portant le nom des 'devices' CP/M : "CON:" pour l'écran, "KBD:" pour le clavier sans écho, "LST:" pour l'imprimante, "AUX:" pour la carte série, etc.

On trouve également les UNTYPED FILES du Pascal UCSD, qui permettent d'accéder à des fichiers CP/M par blocs de 128 octets.

Toutes les procédures de gestion de l'écran sont présentes, avec vidéo inverse (sur //e et //c) et contrôle du curseur.

Dans la version CP/M, on peut définir des variables à une position absolue en mémoire (directive ABSOLUTE, voir programme GRAPHIC), chaîner des programmes, et appeler des sous programmes en assembleur : il suffit d'insérer les codes hexadécimaux de la routine.

Quelques restrictions : pour optimiser la vitesse d'exécution, le Turbo Pascal ne génère de code pour les appels récursif que sur demande de l'utilisateur par l'option {$A+}, et n'effectue la vérification des indices de tableau qu'avec l'option {$X+}.

Les limites

On peut faire au Turbo Pascal deux reproches principaux :

- dans la version Apple CP/M, il n'y a aucune gestion possible des graphiques. On peut lever cette barrière partiellement (voir programme GRAPHIC), mais les routines ne seraient très performantes qu'en assembleur ;
- il n'y a pas de possibilité de créer des 'librairies' et des 'unités' : la seule entité compilable est un PROGRAM. On ne peut donc pas créer de gros programmes, séparés en modules d'une dizaine de Ko s'appelant les uns les autres.

Conclusion

On l'a vu, le TURBO-PASCAL possède des performances époustouflantes en compilation et en vitesse d'exécution. Les débutants pourront l'utiliser de manière beaucoup plus pratique que le Pascal UCSD pour se former, le dialogue avec l'utilisateur étant très intéractif, et les attentes nettement moins longues... L'utilisateur chevronné pourra s'en servir pour exécuter des programmes plus rapidement, la conversion depuis le système UCSD étant relativement aisée avec un programme comme UNIVERSAL FILE CONVERSION.

Programme 'GRAPH.PAS'

Le signe ¬ indique que le retour chariot précédent ne doit être saisi. Il ne sert qu'à la mise en page.

```
PROGRAM GRAPHIC;
{Pour executer ce programme, il faut le compiler en Option C
 OM et demander une START ADDRESS de $5000. apres compilatio
 n, quitter turbo pascal et executer le programme sous CPM}
{ATTENTION en mode graphique, il ne faut pas fairede WRITE n
 i de WRITELN}
CONST DEUX:ARRAY[0..7] OF BYTE=(1,2,4,8,16,32,64,128);
TYPE PA=PACKED ARRAY[0..1] OF 0..255;
     MAGIC=RECORD
             CASE BOOLEAN OF
                TRUE:(INT:INTEGER);
                FALSE:(PTR:^PA);
             END;
VAR ECRAN:PACKED ARRAY[0..8191] OF BYTE ABSOLUTE $3000;
    A,R:REAL;
    OX,OY,X,Y:INTEGER;
FUNCTION PEEK(X:INTEGER):INTEGER;
VAR C:MAGIC;
BEGIN
  C.INT:=X;
  PEEK:=C.PTR^[0]
```

```
END;
PROCEDURE POKE(X,Y:INTEGER);
VAR C:MAGIC;
BEGIN
  C.INT:=X;
  C.PTR^[0]:=Y
END;
PROCEDURE GRAFIXON;
VAR A:INTEGER;
BEGIN
  POKE($E00C,0); POKE($E000,0); A:=PEEK($E055)+PEEK($E050)+P
  ¬EEK($E057)+PEEK($E052);
END;
PROCEDURE TEXT;
VAR A:INTEGER;
BEGIN
  A:=PEEK($E051)+PEEK($E054); POKE($E001,0); POKE($E00D,0);
CLRSCR;
END;
PROCEDURE HGR;
BEGIN
  GRAFIXON;
  FILLCHAR(ECRAN,8192,0)
END;
PROCEDURE PLOT(X,Y:INTEGER);
VAR D,I:INTEGER;
BEGIN
  D:=Y MOD 64;
  I:=1024*(D MOD 8)+128*(D DIV 8)+40*(Y DIV 64)+(XDIV 7);
  ECRAN[I]:=ECRAN[I] OR DEUX[X MOD 7]
END;
PROCEDURE LINE(X1,Y1,X2,Y2:INTEGER);
VAR XT,TX,XX,X,Y:REAL;
    C,I:INTEGER;
BEGIN
  IF X1>X2 THEN BEGIN
    C:=X1; X1:=X2; X2:=C;
    C:=Y1; Y1:=Y2; Y2:=C
  END;
  IF X1=X2 THEN
    IF Y1<Y2 THEN FOR I:=Y1 TO Y2 DO PLOT(X1,I)
             ELSE FOR I:=Y2 TO Y1 DO PLOT(X1,I)
    ELSE IF Y1=Y2 THEN FOR I:=X1 TO X2 DO PLOT(I,Y1)
                  ELSE BEGIN
                    TX:=(Y2-Y1)/(X2-X1); XT:=ABS(1/TX);
                    IF XT>1 THEN XT:=1; X:=X1; Y:=Y1; XX:=TX
                    ¬*XT;
                    REPEAT
                      PLOT(TRUNC(X),TRUNC(Y));
                      X:=X+XT; Y:=Y+XX
                    UNTIL X>X2
                  END
END;
BEGIN
  HGR;
  R:=0.0; A:=0; OX:=140; OY:=95;
  REPEAT
    X:=TRUNC(140+R*COS(A)); Y:=TRUNC(95+R*SIN(A));
    LINE(OX,OY,X,Y); OX:=X; OY:=Y;
    R:=R+1.5; A:=A+1.5533;
  UNTIL (R>95) OR KEYPRESSED;
  READLN; TEXT
END.
```

Une application

Une des limitations du TURBO-PASCAL est de ne pas permettre la gestion des graphiques. Ce petit programme y remédie, de facon rudimentaire car ses performances ne sont pas celles d'une routine en assembleur.

Pour transférer le programme de la disquette Pom's vers une disquette CP/M sous le nom GRAPH.PAS, il est nécessaire d'avoir UNIVERSAL FILE CONVERSION (dans le cas contraire, à vos claviers !). Il faut donc transférer le fichier GRAPH.PAS puis passer sous CP/M et faire un REN GRAPH.PAS=GRAPHPAS. Ensuite, passer sous TURBO PASCAL, puis compiler le programme en option COM FILE en spécifiant une START ADDRESS de $5000.

Le programme lui-même trace un petit dessin de démonstration, un carré qui tourne. Les procédures utilisables sont :

HGR passe en mode graphique et efface l'écran.

GRAFIXON passe en mode graphique sans effacer l'écran.

TEXT passe en mode texte.

PLOT(X,Y) met un point en X,Y à l'écran

LINE(X1,Y1,X2,Y2) trace une ligne entre X1,Y1 et X2,Y2.

Attention, une restriction : il ne faut pas faire de WRITE ni de WRITELN en mode graphique, sous peine de plantage...

Un composeur sur le //c

Bernard Hoyez,
Jean-Luc Nail,
Bruno Fénart

Les propriétaires d'un Apple //c ont sans doute été déçus en s'apercevant que le programme de Bernard Hoyez et Jean-luc Nail permettant de composer automatiquement un numéro de téléphone (Pom's n° 22), ne pouvait pas fonctionner sur leur machine. Qu'ils se réjouissent, voici une version d'AGENDA et de COMPOSEUR qui permet de l'adapter à votre machine favorite.

Pour cela, on utilisera le port série n° 2 "consacré" normalement au modem et plus particulièrement le niveau logique DTR "Data Terminal Ready", qui se trouve sur la broche 1 du connecteur (voir schéma). La valeur de cette sortie est soit -12V soit +12V. C'est cette sortie que l'on va utiliser à la place de AN0 pour commander le relais, bien que les valeurs de AN0 soient 0 et +5V.

Pour protéger le transistor on rajoutera une diode en dérivation.

Pour l'alimentation, la meilleure solution est d'utiliser une pile (de 4,5 ou 9 volts suivant le relais utilisé). Le reste du schéma est identique.

Pour commander le niveau de la broche DTR, il faut utiliser le registre de commande du port considéré (adresses : Port 1 = $C09A - Port 2 = $C0AA). Suivant la valeur de ce registre, DTR et par suite la tension de la broche 1, passe d'un niveau à l'autre (voir le tableau).
Les modifications à apporter au programme AGENDA sont les suivantes :

```
380 POKE 49322,8 Prise de ligne
550 POKE 49322,0 Coupe la ligne
```

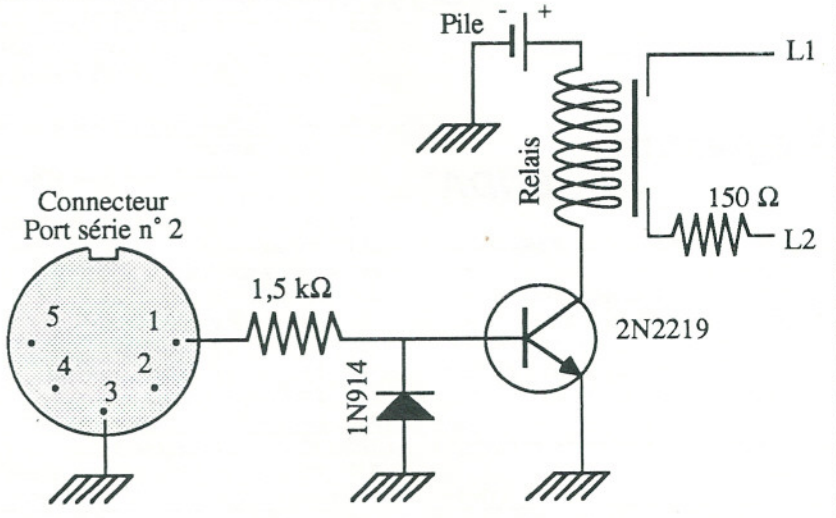


Schéma de branchement

Broche	*Port 1*	*Port 2*	*Signification*	
1	*DTR1*	*DTR2*	*Data Terminal Ready*	*Sortie*
2	*TD1*	*TD2*	*Transmit Data*	*Sortie*
3	*GND*	*GND*	*Masse mécanique/signal*	
4	*RD1*	*RD2*	*Read Data*	*Entrée*
5	*DSR1*	*DSR2*	*Data Set Ready*	*Entrée*

Les broches du connecteur série

	Adresse hexadécimale	*Adresse décimale*
Port 1	*$C09A*	*49306*
Port 2	*$C0AA*	*49322*

Octet :

7	6	5	4	3	2	1	0

Valeur du Registre		$\overline{RTS}$	*Tension Broche 1*
$00	*0 0*	*Haut*	*- 12 V*
$04	*0 1*	*Bas*	*+ 12 V*
$08	*1 0*	*Bas*	*+ 12 V*
$0E	*1 1*	*Bas*	*+ 12 V*

Signification des bits du registre de commande

Pour COMPOSEUR.S voir le listing du nouveau source.
Remarque : on a utilisé le port n° 2, pour le port n° 1 il suffirait de changer l'adresse du registre de commande.

Bibliographie : The Apple //c Reference Manual - Volume 1.

Programme 'AGENDA'

```
10  REM ***COMPOSEUR DE NUMERO***
20  REM *        par            *
30  REM *     Bernard HOYEZ     *
40  REM *  et Jean-Luc NAIL     *
50  REM ***************************
60  REM ==>ENTRER LES NOMS ET LES NUM
    EROS EN DATA A LA SUITE DES AUTR
    ES
70  REM ***************************
80  HOME
90  ONERR  GOTO 570
100  PRINT "      COMPOSITION AUTOMAT
    IQUE DE"
110  PRINT "          NUMERO TELEPHON
    IQUE"
120  PRINT  CHR$ (4);"BLOADCOMPOSEUR"
130  DIM T$(100,2)
140 N = 0
150 N = N + 1
160  FOR J = 1 TO 2
170  READ T$(N,J)
180  NEXT J
190  GOTO 150
200  VTAB (8)
210  PRINT "UTILISEZ LES FLECHES ";:
    INVERSE : PRINT "=>";: NORMAL :
    PRINT " ET ";: INVERSE : PRINT "
    <=": NORMAL
220  PRINT "POUR SELECTIONNER LE NOM
    DE L'ABONNE"
230  PRINT : PRINT "VALIDEZ AVEC LA T
    OUCHE RETURN"
240 I = 0
250  GET A$
260 A% =  ASC (A$)
270  IF A% <  > 13 AND A% <  > 21 AND
     A% <  > 8 THEN 260
280  IF A% = 13 THEN 360
290  IF A% = 21 THEN I = I + 1
300  IF A% = 8 THEN I = I - 1
310  IF I > N - 1 THEN I = N - 1
320  IF I < 1 THEN I = 1
330  VTAB 15
340  CALL  - 868: PRINT T$(I,1),T$(I,
    2)
350  GOTO 250
360 X$ = T$(I,2)
370  REM **COMPOSITION DU NUMERO*
380  POKE 49322,8: REM Prise de ligne
390  FOR I = 1 TO 700: NEXT I
400  HOME
410  PRINT : PRINT "JE COMPOSE LE NUM
    ERO ";
420  FOR I = 1 TO  LEN (X$)
430 A =  ASC ( MID$ (X$,I,1)) + 128
440  POKE 7,A
450  CALL 768
460  PRINT  CHR$ (A);
470  FOR T = 1 TO 600: NEXT T
480  NEXT I
490  HTAB 1: VTAB 5: PRINT "DECROCHEZ
     LE COMBINE, MAINTENANT..."
500  VTAB 8: PRINT ".....AVANT"
510  FOR I = 400 TO 1 STEP  - 1
520  VTAB (15): HTAB (11)
530  PRINT  INT (I / 50);" SECONDES"
540  NEXT I
550  POKE 49322,0: REM Coupe
560  GOTO 580
570  IF  PEEK (222) = 42 THEN 200
580  HOME : RESTORE
590  PRINT "...PRET POUR UN AUTRE APP
    EL ...": GOTO 140
600  DATA  RENSEIGNEMENTS,3612
620  DATA  AUTRES RENSEIGNEMENTS,3611
630  DATA  JULIETTE,12345678
```

Source 'COMPOSEUR.S'

Assembleur Big Mac

```
 1 ***************************
 2 *                         *
 3 *       Composeur         *
 4 *                         *
 5 ***************************
 6
 7 * 8 novembre 85
 8
 9
10 * ADRESSE
11
12 COMREG   =    $C0AA ;Port 2.
     $C09A pour le port 1
13 WAIT     =    $FCA8
14 CHIFFRE  =    7
15 ERREUR   =    8
16
17          ORG  $300
18
19 ENTREE        =          *
20
21          LDA  CHIFFRE
22          CMP  #"0"-1
23          BCC  ERROR
24          CMP  #"9"+1
25          BCS  ERROR
26
27          AND  #%1111
      ;filtrage
28          BNE  COMPOSE
29          LDA  #10
      ;CHIFFRE=0 => <A>=10
30 COMPOSE  TAX
31
32 BOUCLE   =    *
33          JSR  PULSE
34          DEX
35          BNE  BOUCLE
36          RTS
37
38 ERROR    =    *
     ;ce n'etait pas un chiffre
39          LDA  #$FF
40          STA  ERREUR
41          RTS
42
43 PULSE    LDA  #159
44          LDY  #$00
45          STY  COMREG
46          JSR  WAIT
47          LDA  #108
48          LDY  #$08
49          STY  COMREG
50          JSR  WAIT
51          RTS
```

Récapitulation 'COMPOSEUR'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE COMPOSEUR,A$300,L$32

```
0300- A5 07 C9 AF 90 12 C9 BA
0308- B0 0E 29 0F D0 02 A9 0A
0310- AA 20 1D 03 CA D0 FA 60
0318- A9 FF 85 08 60 A9 9F A0
0320- 00 8C AA C0 20 A8 FC A9
0328- 6C A0 08 8C AA C0 20 A8
0330- FC 60
```

GENUTILE : des utilitaires indispensables dans une 'UNIT'

Jean-Claude Lengrand

GENUTILE est un ensemble de PROCEDUREs et FUNCTIONs d'intérêt général constituant une INTRINSIC UNIT au sens du compilateur PASCAL, et qui est incorporé à la SYSTEM LIBRARY sous les numéros de SEGMENTs 16 (programme) et 17 (données).

GENUTILE définit un TYPE CHOIDECA:SET OF CHAR, et une variable ALFANUM de ce type. ALFANUM désigne l'ensemble des caractères alphanumériques depuis " " (espace) jusqu'à "~" (caret).

GENUTILE définit diverses variables de TYPE CHAR qui, lorsqu'elles sont envoyées sur OUTPUT, ont l'effet suivant :

SON	émet un BIP
BS	recul du curseur
EFL	efface la fin de ligne
EFB	efface le bas de l'écran
INV	met l'affichage en mode inverse
NORM	met l'affichage en mode normal
CR	envoie le curseur à la ligne
HOME	efface l'écran ou va à la page

GENUTILE définit aussi *ESC* et *FLD* (flèche droite) de TYPE CHAR. Toutes ces variables sont définies comme le CHR( ) de 7, 8, 29, 11, 18, 20, 13, 12, 27, 21 respectivement.

PROCEDURE PRENRETURN

Attend qu'un <RETURN> ait été frappé au clavier.

Programme 'GENUTILE'

```
(* LENGRAND, LE  9/12/82 *)
(**)
(*$S+*)
      UNIT GENUTILE;INTRINSIC CODE 16 DATA 17;

INTERFACE

  TYPE CHOIDECA=SET OF CHAR;
  VAR BS,SON,EFL,EFB,INV,NORM,CR,ESC,FLD,HOME:CHAR;
      ALFANUM:CHOIDECA;

  PROCEDURE PRENRETURN;
  FUNCTION  PRENCAR(BONSET:CHOIDECA):CHAR;
  FUNCTION  OUI:BOOLEAN;
  PROCEDURE MESSAGE(Y:INTEGER;S:STRING);
  PROCEDURE ALARME(S:STRING);
  PROCEDURE PRENCHAINE(LONGMAX:INTEGER;BONSET:CHOIDECA;VAR S:STRING);
  PROCEDURE ENTIER(LONGMAX:INTEGER;VAR S:INTEGER);
  PROCEDURE CONTINU(Y:INTEGER);
  PROCEDURE ALITEXT(LONGMAX:INTEGER;S:STRING);
  PROCEDURE SAISIE(X,Y,LONGMAX:INTEGER;BONSET:CHOIDECA;LIBELLE:STRING;
            VAR S:STRING);

  FUNCTION  PEEK(ADRESSE:INTEGER):INTEGER;
  PROCEDURE POKE(ADRESSE,CONTENU:INTEGER);

  PROCEDURE ERROR(L:INTEGER;VAR S:STRING);
  PROCEDURE VALBOOL(S:STRING;VAR N:BOOLEAN);
  PROCEDURE VALREEL(S:STRING;VAR K:REAL);
  PROCEDURE VALENT (S:STRING;VAR K:INTEGER);
  PROCEDURE STRBOOL(N:BOOLEAN;L:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER);
  PROCEDURE STRFIX(N:REAL;L,D:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER);
  PROCEDURE STRFLOT(N:REAL;L,D:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER);
  PROCEDURE STRENT(N,L:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER);

IMPLEMENTATION

  PROCEDURE PRENRETURN;
  VAR SORT  :CHAR;
  BEGIN
    REPEAT
      READ(KEYBOARD,SORT)
    UNTIL EOLN(KEYBOARD)
  END;

  FUNCTION PRENCAR;
  VAR CH    :CHAR;
      BON   :BOOLEAN;
  BEGIN
    REPEAT
      READ(KEYBOARD,CH);
      IF EOLN(KEYBOARD) THEN CH:=CR;
      BON:=CH IN BONSET;
      IF NOT BON THEN WRITE(SON)
        ELSE IF CH IN [' '..'^'] THEN WRITE(CH)
    UNTIL BON;
    PRENCAR:=CH
  END;

  FUNCTION OUI;
  BEGIN
    OUI:=PRENCAR(['Y','O','N']) IN ['Y','O']
  END;

  PROCEDURE MESSAGE;
  BEGIN
```

FUNCTION PRENCAR (BONSET:CHOIDECA): CHAR

Renvoie un caractère frappé au clavier. Ce caractère doit faire partie de BONSET, défini préalablement comme un SET OF CHAR, sinon, émet un BIP et attend un autre caractère.

FUNCTION OUI:BOOLEAN

Attend un caractère au clavier et renvoie TRUE s'il s'agit de O ou de Y et renvoie FALSE s'il s'agit de N. Émet un BIP et attend un autre caractère dans tous les autres cas.

PROCEDURE MESSAGE (Y:INTEGER;S:STRING)

Écrit le message S à partir du début de la ligne Y (0<Y<23) et efface la fin de la ligne. Le curseur reste immédiatement après le message.

PROCEDURE ALARME(S:STRING)

Efface l'écran sauf la 1ère ligne, affiche le message S et attend un <RETURN>.

PROCEDURE PRENCHAINE (LONGMAX:INTEGER; BONSET:CHOIDECA; VAR S:STRING)

Saisit au clavier la chaine S, avec limitation à LONGMAX caractères choisis dans BONSET. Si l'on répond par un simple <RETURN>, la valeur précédente de S est conservée. La correction par la flèche gauche (BS) est permise.

PROCEDURE ENTIER(LONGMAX: INTEGER; VAR S:INTEGER)

Saisit au clavier l'entier positif S, avec limitation à LONGMAX caractères choisis dans '0'..'9'. Si l'on repond par un simple <RETURN>, la valeur précédente

```
  GOTOXY(0,Y);WRITE(S,EFL)
END;

PROCEDURE ALARME;
BEGIN
  GOTOXY(0,1);WRITE(SON,EFB);
  MESSAGE(10,S);
  MESSAGE(12,'FAITES <RETURN> ');
  PRENRETURN
END;

PROCEDURE PRENCHAINE;
 (**)
 (*  SAISIT UNE CHAINE DE CARACTERES AVEC
     LONGUEUR MAXI ET VALEUR PAR DEFAUT *)
 (**)
 VAR S1    :STRING[1];
     CONT  :STRING;
 BEGIN
   S1:=' ';
   CONT:='';
   REPEAT
     IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(BONSET+[CR])
       ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:=PRENCAR([CR,BS])
              ELSE S1[1]:=PRENCAR(BONSET+[CR,BS]);
     IF S1[1] IN BONSET THEN CONT:=CONCAT(CONT,S1)
       ELSE IF S1[1]=BS THEN
         BEGIN
           WRITE(BS,' ',BS);
           DELETE(CONT,LENGTH(CONT),1)
         END;
   UNTIL S1[1]=CR;
   IF LENGTH(CONT)<>0
     THEN  S:=CONT
     ELSE WRITE(S)
 END;

 PROCEDURE ENTIER;
 (**)
 (*  SAISIT UN NOMBRE ENTIER POSITIF *)
 (**)
 VAR S1    :STRING[1];
     I     :INTEGER;
     CONT  :STRING;
     OKSET :CHOIDECA;
 BEGIN
   OKSET:=['0'..'9'];S1:=' ';CONT:='';
   REPEAT
     IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(OKSET+[CR])
       ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:=PRENCAR([CR,BS])
         ELSE S1[1]:=PRENCAR(OKSET+[CR,BS]);
     IF S1[1] IN OKSET THEN CONT:=CONCAT(CONT,S1)
       ELSE IF S1[1]=BS THEN
         BEGIN
           WRITE(BS,' ',BS);
           DELETE(CONT,LENGTH(CONT),1)
         END;
   UNTIL S1[1]=CR;
   IF LENGTH(CONT)<>0 THEN
    BEGIN
      S:=0;
      FOR I:=1 TO LENGTH(CONT) DO
      BEGIN
        S:=S*10;S:=S+(ORD(CONT[I])-ORD('0'))
      END;
    END ELSE WRITE(S)
 END;

 PROCEDURE CONTINU;
 (**)
 (*  INVITE L'OPERATEUR A CONTINUER *)
 (**)
 BEGIN
   WRITE(INV);
   MESSAGE(23,'FAIRE <RETURN>POUR CONTINUER');
   WRITE(NORM);
   PRENRETURN;
   GOTOXY(0,Y);
   WRITE(EFB)
 END;
 (**)
 PROCEDURE ALITEXT;
```

de S est conservée. La correction par la flèche gauche (BS) est permise.

PROCEDURE CONTINU(Y:INTEGER)

Affiche sur la dernière ligne de l'écran une invitation à frapper <RETURN> pour continuer, attend <RETURN>, envoie le curseur en (0,Y) et efface le bas de l'écran.

PROCEDURE ALITEXT(LONGMAX: INTEGER;S:STRING)

Affiche le texte S cadré à gauche sur LONGMAX positions, après troncature si nécessaire.

PROCEDURE SAISIE(X,Y,LONGMAX: INTEGER;BONSET: CHOIDECA;LIBELLE: STRING);VAR S:STRING)

Même fonction que PRENCHAINE, mais avec écriture de LIBELLE au point de l'écran (X,Y), juste avant la saisie. La valeur par défaut est affichée pendant la saisie, elle est éventuellement tronquée à LONGMAX caractères.

FUNCTION PEEK(ADRESSE: INTEGER)

Renvoie le contenu décimal de la mémoire ADRESSE.

PROCEDURE POKE(ADRESSE, CONTENU:INTEGER)

Met CONTENU dans la mémoire ADRESSE. Aucun test n'est effectué sur la valeur des paramètres. C'est une procédure à utiliser avec précaution.

PROCEDURE ERROR(L:INTEGER;VAR S:STRING)

Remplit une chaîne S de longueur L avec L astérisques.

```
  (**)
  (*   EDITE UNE CHAINE CADREE A GAUCHE *)
  (**)
  BEGIN
    IF LENGTH(S)>LONGMAX THEN S:=COPY(S,1,LONGMAX);
    WRITE(S,'':LONGMAX-LENGTH(S))
  END;
  (**)
  PROCEDURE SAISIE;
  (**)
  (*  SAISIT UNE CHAINE DE CARACTERES EN UN POINT DONNE
      DE L'ECRAN, AVEC LONGUEUR MAXI ET VALEUR PAR DEFAUT *)
  (**)
  BEGIN
    IF LENGTH(S)>LONGMAX THEN S:=COPY(S,1,LONGMAX);
    GOTOXY(X,Y);WRITE(LIBELLE);ALITEXT(LONGMAX,S);
    GOTOXY(X+LENGTH(LIBELLE),Y);
    PRENCHAINE(LONGMAX,BONSET,S);
    WRITE('':LONGMAX-LENGTH(S))
  END;

FUNCTION PEEK;EXTERNAL;

PROCEDURE POKE;EXTERNAL;

(**)
PROCEDURE ERROR;
(**)
(*  REMPLIT UNE CHAINE S DE L ASTERISQUES *)
(**)
VAR I:INTEGER;
BEGIN S:='';
      FOR I:=1 TO L DO S:=CONCAT('*',S);
END;

(**)
PROCEDURE VALBOOL;
(**)
(*  CONVERTIT UNE CHAINE EN BOOLEEN *)
(**)
BEGIN
  IF LENGTH(S)<>0 THEN
  CASE S[1] OF
    'T','V','Y','O','1':N:=TRUE;
          'F','N','0':N:=FALSE
  END
END;
(**)
PROCEDURE VALREEL;
(**)
(*  CONVERTIT UNE CHAINE EN REEL *)
(**)
TYPE PARTIE=(BLK,SGN,ENT,DEC,SEXP,EXPO);

VAR REGION       :PARTIE;
    SIGNE,PUISS,I,L  :INTEGER;
    FAC              :REAL;
    C                :CHAR;

PROCEDURE CALCULE;
  VAR I:INTEGER;
  BEGIN
    IF PUISS<-37
      THEN K:=0.0
      ELSE IF PUISS>37
             THEN K:=1.01E37
             ELSE IF PUISS<0
                    THEN FOR I:=1 TO -PUISS DO K:=K/10.0
                    ELSE FOR I:=1 TO  PUISS DO K:=K*10.0
  END;

BEGIN
  K:=0.0;SIGNE:=1;FAC:=1.0;PUISS:=0;
  REGION:=BLK;L:=LENGTH(S);
  IF L=0 THEN EXIT(VALREEL);
  FOR I:=1 TO L DO
    BEGIN C:=S[I];
      CASE REGION OF
        BLK :BEGIN
               IF NOT( C IN [' ','-','+','.','0'..'9'])THEN EXIT(VALREEL);
               CASE C OF
                 '+':REGION:=SGN;
                 '-':BEGIN
```

PROCEDURE VALBOOL(S:STRING;VAR N:BOOLEAN)

Si S commence par Y,O,T,V,1, rend N égal à TRUE.
Si S commence par N,F,0, rend N égal à FALSE.
Laisse N inchange dans les autres cas.

PROCEDURE VALREEL(S:STRING;VAR K:REAL)

Renvoie dans K la valeur réelle représentée par la chaîne S. L'interprétation s'arrête dès que la procédure trouve un caractère inattendu, comme la fonction VAL du BASIC. En particulier, elle s'arrête sur le premier espace rencontré. La forme avec exposant est reconnue. Le nombre peut être précédé de 1 ou plusieurs espaces placés avant tout caractère significatif. La forme de l'exposant est libre. Il peut être absent. Le point décimal peut être absent. Le nombre peut être précédé de un ou plusieurs signes qui se combinent. Ex: +-+3.14E sera interprété comme -3.14 (exposant=0). S peut être une chaîne vide. K vaut alors 0. Une valeur absolue inférieure à 1E-37 est considérée comme nulle. Une valeur absolue supérieure à 1E37 est prise égale à 1.01E37, un test sur la valeur renvoyée peut permettre de détecter un dépassement éventuel.

PROCEDURE VALENT(S:STRING;VAR K:INTEGER)

Remplit la même fonction que VALREEL, notamment pour la reconnaissance d'un nombre sous n'importe qu'elle forme, mais renvoie la valeur entière la plus proche du nombre représenté. Les entiers dont la valeur absolue excède MAXINT, soit 32767, sont ramenés à +-32767.

PROCEDURE STRBOOL(N:BOOLEAN; L:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER)

```
                    REGION:=SGN;SIGNE:=-SIGNE
                  END;
                '0','1','2','3','4','5','6','7','8','9':BEGIN
                    REGION:=ENT;K:=ORD(C)-ORD('0')
                  END;
                '.':REGION:=DEC;
              END;
            END;
        SGN :BEGIN
              IF NOT ( C IN ['-','+','.','0'..'9']) THEN EXIT(VALREEL);
              CASE C OF
                '-':SIGNE:=-SIGNE;
                '0','1','2','3','4','5','6','7','8','9':BEGIN
                    REGION:=ENT;K:=SIGNE*(ORD(C)-ORD('0'))
                  END;
                '.':REGION:=DEC;
              END;
            END;
        ENT :BEGIN
              IF NOT( C IN ['.','E','0'..'9']) THEN EXIT(VALREEL);
              CASE C OF
                '.':REGION:=DEC;
                'E':REGION:=SEXP;
      '0','1','2','3','4','5','6','7','8','9':K:=K*10.0+SIGNE*(ORD(C)-ORD('0'))
              END;
            END;
        DEC :BEGIN
              IF NOT( C IN ['0'..'9','E']) THEN EXIT(VALREEL);
              CASE C OF
                'E':REGION:=SEXP;
                '0','1','2','3','4','5','6','7','8','9':BEGIN
                    FAC:=FAC/10.0;K:=K+SIGNE*FAC*(ORD(C)-ORD('0'))
                  END;
              END;
            END;
        SEXP:BEGIN
              IF NOT( C IN ['-','+','0'..'9']) THEN EXIT(VALREEL);
              IF  C='-' THEN SIGNE:=-1
                        ELSE SIGNE:=1;
              IF C IN ['0'..'9'] THEN PUISS:=SIGNE*(ORD(C)-ORD('0'));
              REGION:=EXPO
            END;
        EXPO:BEGIN
              IF (NOT( C IN ['0'..'9'])) OR (ABS(PUISS)>MAXINT DIV 10)
                THEN BEGIN CALCULE;EXIT(VALREEL) END;
              PUISS:=PUISS*10+SIGNE*(ORD(C)-ORD('0'))
            END;
      END (* CASE *)
    END;  (* FOR  *)
  CALCULE;
END;

(**)
PROCEDURE VALENT;
(**)
(*  CONVERTIT UNE CHAINE EN ENTIER *)
(**)
VAR X:REAL;
BEGIN
  VALREEL(S,X);
  IF ABS(X)>MAXINT
    THEN IF X>0.0 THEN K:=MAXINT
                  ELSE K:=-MAXINT
    ELSE K:=ROUND(X)
END;

(**)
PROCEDURE STRBOOL;
(**)
(*   CONVERTIT UN BOOLEEN EN CHAINE *)
(**)
VAR I:INTEGER;
BEGIN ERREUR:=0;
      IF N THEN S:='VRAI'
           ELSE S:='FAUX';
      IF L<1 THEN BEGIN ERREUR:=1;S:='';EXIT(STRBOOL) END;
      IF L>80 THEN L:=80;
      IF L>4 THEN FOR I:=5 TO L DO S:=CONCAT(' ',S);
      IF L<4 THEN S:=COPY(S,1,L)
END;

(**)
```

renvoie une chaîne de longueur L contenant VRAI ou FAUX, éventuellement tronquée ou cadrée à droite, selon la valeur de N. ERREUR prend la valeur 1 si L<1, et 0 dans les autres cas.

PROCEDURE STRFIX(N:REAL;L, D:INTEGER:VAR S:STRING;VAR ERREUR:INTEGER)

Renvoie une chaîne S de longueur L représentant le nombre réel N avec D décimales, en format FIXE : -xxx.yy
Si L<3 ou D<0 ou L<D+2, on obtient ERREUR=1.
Si le nombre est trop grand (/n/ >32767), on obtient l'erreur 2.
Si le nombre est trop grand pour le nombre de caractères total imposé, on obtient l'erreur 3.
Dans les autres cas, ERREUR reste nul.
Si L est supérieur à 80, il est considéré comme étant égal à 80, sans que cela soit signalé.
Le nombre de décimales peut être réduit d'office par la procédure si la partie décimale est représentée par un entier superieur à 32767. Aucune erreur n'est alors signalée, et le nombre total de caractères est respecté.
Si l'on est sûr que le nombre est positif, on peut ne pas prévoir de place pour le signe.

PROCEDURE STRFLOT(N:REAL; L,D:INTEGER;VAR S:STRING;VAR ERREUR:INTEGER)

Même fonction que STRFIX, mais la chaîne contient le nombre en représentation flottante : -1.2344E+03 avec un chiffre avant le point décimal et un exposant de 4 caractères.
L'erreur 2 de STRFIX ne peut pas se produire. Les conditions d'apparition de l'erreur 1 sont : L<7 ou D<0 ou L<D+6.

PROCEDURE STRENT(N,L:INTEGER; VAR S:STRING; VAR ERREUR:INTEGER)

Renvoie une chaîne de longueur L

```
PROCEDURE STRFIX;
(**)
(*  CONVERTIT UN REEL EN CHAINE DE TYPE -XXXX.XX  *)
VAR I,J    :INTEGER;
   NEGATIF:BOOLEAN;
    ND,EX :REAL;
    S1    :STRING;
BEGIN
  IF L>80 THEN L:=80;
  IF (L<3) OR (D<0) OR (L<D+2)
    THEN BEGIN ERROR(L,S);ERREUR:=1;EXIT(STRFIX) END
    ELSE ERREUR:=0;
  NEGATIF:=(N<0.0);N:=ABS(N);
  IF N>32767.0 THEN
    BEGIN ERROR(L,S);ERREUR:=2;EXIT(STRFIX) END;
  IF D=0
    THEN
      BEGIN
        STR(ROUND(N),S1);S:=''
      END
    ELSE
      BEGIN
        STR(TRUNC(N),S1);
        REPEAT
          EX:=1;FOR I:=1 TO D DO EX:=EX*10.0;
          ND:=EX*(N-TRUNC(N));D:=D-1;
        UNTIL ND<=32767.0;
        J:=ROUND(ND);STR(J,S);D:=D+1;
        FOR J:=LENGTH(S)+1 TO D DO S:=CONCAT('0',S);
      END;
  S:=CONCAT(S1,'.',S);
  IF NEGATIF THEN S:=CONCAT('-',S);
  IF LENGTH(S)>L THEN
    BEGIN ERROR(L,S);ERREUR:=3;EXIT(STRFIX) END;
  FOR J:=LENGTH(S)+1 TO L DO S:=CONCAT(' ',S);
END;

(**)
PROCEDURE STRFLOT;
(**)
(*  CONVERTIT UN REEL EN CHAINE DE TYPE -XXXX.XXE+YY  *)
(**)
VAR S1     :STRING;
    EX     :INTEGER;
    NEGATIF:BOOLEAN;
BEGIN
  IF L>80 THEN L:=80;
  IF (L<7) OR (D<0) OR (L<D+6)
    THEN BEGIN ERROR(L,S);ERREUR:=1;EXIT(STRFLOT) END
    ELSE ERREUR:=0;
  NEGATIF:=(N<0.0);N:=ABS(N);EX:=0;
  IF N<>0.0 THEN
    BEGIN
      WHILE N>=10.0 DO
        BEGIN N:=N/10.0;EX:=EX+1 END;
      WHILE N<1.0 DO
        BEGIN N:=N*10.0;EX:=EX-1 END;
    END;
  STR(ABS(EX),S);
  IF ABS(EX)<10 THEN S:=CONCAT('0',S);
  IF EX<0 THEN S:=CONCAT('E-',S)
          ELSE S:=CONCAT('E+',S);
  IF NEGATIF THEN N:=-N;
  L:=L-4;
  STRFIX(N,L,D,S1,ERREUR);
  S:=CONCAT(S1,S)
END;

(**)
PROCEDURE STRENT;
(**)
(*  CONVERTIT UN ENTIER EN CHAINE *)
(**)
VAR I,J:INTEGER;
BEGIN
  IF L>80 THEN L:=0;
  ERREUR:=0;
  STR(N,S);I:=LENGTH(S);
  IF L>I THEN FOR J:=1 TO L-I DO S:=CONCAT(' ',S);
  IF L<I THEN BEGIN ERROR(L,S);ERREUR:=1 END
END;

    (* INITIALISATION *)
```

représentant l'entier N. L'erreur 1 se produit si L<1 ou si L>80.

Les PROCEDURES PRENCHAINE et SAISIE peuvent provoquer des diagnostics à la compilation si la paramètre actuel S est une chaîne de longueur déclarée (STRING{xx}). Il faut alors compiler le programme (ou la fraction de programme concernée) avec l'option (*$V-*) du compilateur pour éviter le diagnostic.

En cas d'erreur dans les procédures STRxxxx, la chaîne qui est renvoyée ne contient que des *** mais cela n'entraîne pas d'erreur d'exécution.

En cas d'espace en trop dans ces procédures, les chaînes sont cadrées à droite.

La version PASCAL de GENUTILE (obtenue après transfert de la disquette Pom's et nommée alors UTIDIV.TEXT) déclare PEEK et POKE comme étant EXTERNAL. PEEK et POKE ont été écrits en assembleur (fichier PEEKPOKE. TEXT). On doit donc

- compiler UTIDIV.TEXT,
- assembler PEEKPOKE. TEXT,
- linker PEEKPOKE.CODE dans UTIDIV.CODE pour obtenir GENUTILE.CODE,
- placer GENUTILE.CODE dans SYSTEM.LIBRARY par l'utilitaire LIBRARY.CODE.

```
BEGIN
  HOME:=CHR(12);  ESC :=CHR(27);     EFL:=CHR(29);     BS  :=CHR(8);
  CR  :=CHR(13);  FLD:=CHR(21);      SON :=CHR(7);     INV:=CHR(18);
  NORM:=CHR(20);  EFB:=CHR(11);
  ALFANUM:=[' '..'^']
  END.
```

Les identificateurs déclarés par GENUTILE sont :

ALFANUM	ALARME	ALITEXT	BS
CHOIDECA	CONTINU	CR	EFB
EFL	ENTIER	ERROR	ESC
HOME	INV	MESSAGE	NORM
OUI	PEEK	POKE	PRENCAR
PRENCHAINE	PRENRETURN	SAISIE	SON
STRBOOL	STRENT	STRFIX	STRFLOT
VALBOOL	VALENT	VALREEL.	

Source 'PEEKPOKE.TEXT'

```
    .FUNC PEEK,1
RET .EQU 0
ADR .EQU 2
    PLA
    STA RET
    PLA
    STA RET+1
    PLA
    PLA
    PLA
    PLA
    PLA
    STA ADR
    PLA
    STA ADR+1
    LDX #0
    TXA
    PHA
    LDA @ADR,X
    PHA
    LDA RET+1
    PHA
    LDA RET
    PHA
    RTS
    .PROC POKE,2
RET .EQU 0
ADR .EQU 2
VAL .EQU 4
    PLA
    STA RET
    PLA
    STA RET+1
    PLA
    STA VAL
    PLA
    PLA
    STA ADR
    PLA
    STA ADR+1
    LDX #0
    LDA VAL
    STA @ADR,X
    LDA RET+1
    PHA
    LDA RET
    PHA
    RTS
    .END
```

Un formateur de listing Basic

Sylvie Gallet

Qu'est-ce qu'un formateur de programme Applesoft ? C'est une routine qui affiche ou imprime un programme Basic pour en faciliter la lecture. La mise au point en sera donc facilitée et, une fois terminée, on obtient une documentation d'allure professionnelle.

Inspiré d'une routine écrite elle-même en Basic dans Call-Apple, FORMATEUR est réalisé entièrement en assembleur pour des raisons évidentes de rapidité. Une fois chargé en mémoire, il peut être utilisé à tout moment.

Le programme Applesoft est listé de la façon suivante:

- une seule instruction par ligne (les ':' excédentaires entre deux instructions sont ignorés) ;
- une ligne blanche entre deux lignes de programme ;
- décalage de 4 caractères dans les boucles FOR-NEXT; remplacement des NEXT X,Y,Z par NEXT X : NEXT Y : NEXT Z ;
- décalage de 4 caractères après THEN ;
- décalage de 4 caractères pour les instructions dont la longueur dépasse une ligne (sauf cas de force majeure, les mots ne sont pas coupés) ;
- en début de ligne, on rajoute LET avant l'affectation d'une variable.

Bien entendu, le programme listé n'est absolument pas modifié par FORMATEUR.

Le format par défaut est 80 colonnes sur 66 lignes, mais l'utilisateur a la possibilité de modifier ces valeurs. Le saut de page et la pagination sont automatiques. Enfin, on peut également imprimer un titre et une date.

Utilisation

Le programme se charge par *BRUN FORMAT*. Les commandes disponibles sont :

&L
Liste l'ensemble du programme à l'écran. Comme la commande LIST de l'Applesoft, on peut la faire suivre d'un numéro de ligne :

```
&L 10          sont des
&L 10,20       formats
&L 10,         acceptés
&L ,10
```

&LP
Liste sur l'imprimante. Le format est identique que `&L`. L'imprimante est supposée connectée en slot 1. Notez que les instructions PR#n ne sont pas imprimées.

&T
Saisie du titre sur au plus 32 caractères. Le format est libre. Effectuer la séquence :

`&T` <Return> titre <Return>

On peut également y inclure des codes de contrôle pour l'imprimante (sauf ESC).

&D
Saisie de la date sur 32 caractères en format libre.

&P n
n définit la longueur d'une page (de 21 à 127 sinon message d'erreur).

&C n
n définit la largeur d'une page (de 40 à 127 sinon message d'erreur).

Le listing peut être interrompu par CTRL-C. Utilisez CTRL-S pour une interruption temporaire.

Fonctionnement

Sous DOS 3.3, le programme se charge en $90D1 et fixe Himem à $9100. Le programme initialise l'Ampersand et les valeurs par défaut du format, puis rend la main.

Lorsqu'il est appelé pour une liste, il scanne le programme Basic en mémoire en effectuant, d'une part le même travail que la routine LIST de l'Applesoft (conversion des numéros de ligne stockés en binaire en valeurs affichables, traduction des tokens en mots-clés, etc.), d'autre part en améliorant l'affichage comme décrit précédemment.

FORMATEUR utilise un grand nombre d'adresses inutilisées en page zéro ; cela ne pose pas de problèmes car il est prévu pour être activé en mode direct et non à l'intérieur d'un programme. De plus, la zone $300-$33F sert à sauvegarder le titre et la date éventuellement saisis.

N.D.L.R. : les routines d'accès aux périphériques du programme ont été légèrement modifiées afin de permettre un fonctionnement normal avec une carte 80 colonnes, ainsi qu'une exécution au choix sous DOS 3.3 ou ProDOS.
Le programme T.FORMAT est le fichier source au format Big Mac. L'assemblage sous DOS 3.3 se fait simplement en mettant la variable PRODOS à 0.

Pour ProDOS, elle doit être mise à 1 et le source doit être resauvé sur disque. L'assemblage s'effectue en chargeant DARWIN.S (pour les détails d'utilisation, voir Pom's 22) ; la dernière ligne doit devenir "PUT FORMAT" et on assemble en donnant les paramètres :

```
DARWIN_D = $90D1
DARWIN_F = $95FF
VA_DEBUT = 1
```

Le module objet peut alors être chargé sous ProDOS par :

```
BRUN FORMAT.PRODOS,A$6000
```

(il se relogera au dessus des buffers de ProDOS).

```
03 mars 1986                                    PAGE 1

Convert


400     LET J = J + 1

410     NEXT

420     LET N2 = J - 1

430     REM  INCLUSION DES BRK

440     ONERR  GOTO 460

450     LET F$ = F$(3) :
        PRINT D$"BLOAD"F$ :
        GOTO 470

460     CALL 768 :
        GOSUB 1080 :
        GOTO 440

470     CALL 37500

480     FOR I = 1 TO N1

490         IF  INT (I / 500) = I / 500 THEN
                PRINT  :
                PRINT "PATIENCE JE FAIS LE MENAGE!"; :
                CALL 37500 :
                PRINT "C'EST FINI" :
                PRINT

500         FOR J = 1 TO 12

510             IF  MID$ (A$(I),J,1) <  > "J" THEN
                    670

520             LET J = J + 1 :
                IF  MID$ (A$(I),J,1) <  > "S" THEN
                    550

530             LET J = J + 1 :
                IF  MID$ (A$(I),J,1) <  > "R" THEN
                    670

540             LET Z$ = Z1$ :
                GOTO 590

550             IF  MID$ (A$(I),J,1) <  > "M" THEN
                    670

560             LET J = J + 1 :
                IF  MID$ (A$(I),J,1) <  > "P" THEN
                    670

570             LET Z$ = Z2$

580             REM  ON A UN JSR OU JMP
```

Un listing formaté par 'FORMAT'

Ce listing partiel de 'CONVERT' publié dans ces pages, a été obtenu à l'imprimante en faisant :

- ***BRUN FORMAT***
- ***LOAD CONVERT***
- ***&T** pour la saisie du titre 'Convert'*
- ***&D** pour la date '03 mars 1986'*
- ***&LP 400,680** pour le listing proprement dit.*

Note : sous ProDOS, il faut faire BRUN FORMAT.PRODOS,A$6000 et il vous faut un Basic.System dont la version est supérieure à 1.1.

```
03 mars 1986                                    PAGE 2

Convert

590             LET M = J + 2

600             LET Y$ = ""

610             IF J > 4 THEN
                    LET Y$ =  LEFT$ (A$(I),J - 4)

620             FOR K = 1 TO N2

630                 LET L =  LEN (Z$(K))

640                 IF  LEN (A$(I)) - M + 1 < L THEN
                        660

650                 IF  MID$ (A$(I),M,L) = Z$(K) THEN
                        LET A$(I) = Y$ + Z$ + Z$(K) :
                        PRINT A$(I),"LIGNE N[ ";I :
                        PRINT  :
                        LET K = N2

660             NEXT K :
                LET J = 12

670         NEXT J :
        NEXT I

680     HOME  :
        PRINT "SAUVER LE FICHIER TRANSFORME ?" :
        PRINT  :
        PRINT "          SUR DISQUETTE? <O,N>";
```

Source 'DARWIN.S'
Assembleur BIG MAC

```
1  ****************************
2  *         DARWIN           *
3  ****************************
4
5  * Routine d'initialisation à ProDOS
6  * d'un programme assembleur DOS 3.3
7
8  * (C) 1985 Alexandre Avrane
9  * Modifié: 24/11/85
10 * Créé:    16/11/85
11
12 * Ce source est assemblé en conjonction
13 * avec le programme à adapter ProDOS
14
15 DARWIN_D KBD              ;adresse de début DOS 3.3
16 DARWIN_F KBD              ;adresse de fin DOS 3.3
17 VA_DEBUT KBD              ;saute en début de pgm après chargement?
18
19 * Calcul des adresses sous ProDOS
20 DARWINV1 =    DARWIN_F/$100+1
21 DARWINV2 =    DARWIN_D/$100
22 DARWINV3 =    DARWINV1-DARWINV2
23 DARWINV4 =    $100*DARWINV2
24 DARWINV5 =    DARWIN_D-DARWINV4
25 DARWINV6 =    DARWINV3*$100-DARWINV5
26
27 DARWIN_A =    $9A00-DARWINV6 ; adresse de début ProDOS
28 DARWIN_L =    DARWIN_F-DARWIN_D ; longueur ProDOS
29
30         ORG   DARWIN_A-73 ;DARWIN prend 72 octets
31
32 * ================================
33 * 1 - Recherche de la place mémoire
34 * ================================
35         LDA   $BFFD      IVERSION
36         BEQ   DARWIN10   Basic.System 1.0
37         LDA   #0-DARWIN_A/$100+$9B
38         JSR   $BEF5      GETBUFR
39         BCC   DARWIN11
40 DARWIN10 LDA  #14        PROGRAM TOO LARGE
41         JMP   $BE09
42 DARWIN11 =    *
43
44 * ========================
45 * 2 - Copie vers l'adresse d'implantation
46 * ========================
47 DARWIN20 JSR  $FF58       contient un RTS
48         TSX
49         DEX
50         CLC
51         LDA   $100,X     recherche sur la pile
52         ADC   #DARWIN99-DARWIN20-2 ;adresse début routine
53         STA   $3C        A1
54         LDA   $101,X
55         ADC   #0
56         STA   $3C+1
57         LDA   $3C        adresse de fin
58         ADC   #<DARWIN_L
59         STA   $3E        A2
60         LDA   $3C+1
61         ADC   #>DARWIN_L
62         STA   $3E+1
63         LDA   #<DARWIN_A adresse d'arrivée
64         STA   $42        A4
65         PHA
66         LDA   #>DARWIN_A
67         STA   $42+1
68         PHA
69         LDY   #0
70         JSR   $FE2C      déplace le bloc par MOVE
71
72 * ============================
73 * 3 - Appel du module et sortie
74 * ============================
75         PLA
76         STA   $42+1
77         PLA
78         STA   $42
79         DO    VA_DEBUT
80         JMP   ($42)      saute en début de programme...
81         ELSE
82         RTS              ...ou retour à l'Applesoft
83         NOP
84         NOP
85         FIN
86 DARWIN99 =    *
87 * Inclut le source du programme à adapter (format TEXT)
88         PUT   FORMAT
89         END
```

Récapitulation en version ProDOS 'FORMAT.PRODOS'

Après avoir saisi ce code sous moniteur, à partir de l'adresse $6000, vous le sauvegarderez par BSAVE FORMAT,A$6000,L$55E

```
6000- AD FD BF F0 07 A9 07 20
6008- F5 BE 90 05 A9 0E 4C 09
6010- BE 20 58 FF BA CA 18 BD
6018- 00 01 69 36 85 3C BD 01
6020- 01 69 00 85 3D A5 3C 69
6028- 20 85 3E A5 3D 69 05 85
6030- 3F A9 00 85 42 48 A9 94
6038- 85 43 48 A0 00 20 2C FE
6040- 68 85 43 68 85 42 6C 42
6048- 00 A9 4C 8D F5 03 A9 2B
6050- 8D F6 03 A9 94 8D F7 03
6058- A9 42 85 19 A9 50 85 1B
6060- A9 07 85 09 A0 3F A9 03
6068- 85 CF A9 00 85 CE 20 1A
6070- 98 85 D6 60 C9 4C F0 03
6078- 4C C3 97 A9 00 85 FB 85
6080- FC 85 FD A9 FF 85 FE 85
6088- FF 20 B1 00 90 2C C9 50
6090- F0 07 C9 00 D0 11 4C A9
6098- 94 A5 D6 F0 05 C6 D6 4C
60A0- C9 DE E6 D6 4C 40 94 C9
60A8- 2C F0 03 4C C9 DE A5 FB
60B0- F0 03 4C C9 DE E6 FB 4C
60B8- 40 94 20 0C DA A5 FB D0
60C0- 0D A5 50 85 FC A5 51 85
60C8- FD 20 B7 00 D0 BE A5 50
60D0- 85 FE A5 51 85 FF 20 B7
60D8- 00 F0 03 4C C9 DE A5 FD
60E0- C5 FF 90 0E D0 06 A5 FE
60E8- C5 FC B0 06 20 F3 98 4C
60F0- 00 BE AD 31 BE 49 FD 8D
60F8- 14 99 F0 05 A9 33 8D 14
6100- 99 A5 D6 F0 07 A9 31 20
6108- F6 98 C6 D6 85 FB 85 18
6110- 85 1A 85 1D 85 1C 85 1E
6118- 85 EC 85 D7 A5 FC 85 50
6120- A5 FD 85 51 20 1A D6 A0
6128- 03 B0 0A A0 00 F0 0F 4C
6130- FE 94 4C A3 94 20 49 97
6138- 20 68 97 4C 2B 95 20 49
6140- 97 20 68 97 4C 01 95 20
6148- F8 96 20 F8 96 F0 9D AD
6150- 00 C0 10 0A C9 83 D0 06
6158- AD 10 C0 4C A3 94 20 F8
6160- 96 85 50 20 F8 96 85 51
6168- C5 FF 90 08 D0 C4 A5 FE
6170- C5 50 90 BE 20 49 97 20
6178- 49 97 A5 51 A6 50 84 07
6180- 20 24 ED 84 06 A9 00 18
6188- 69 07 85 1A 38 E5 06 A8
6190- 88 A9 20 20 5C DB 88 D0
6198- FA A9 00 85 1E 85 1D 85
61A0- D7 85 EB A4 07 20 F8 96
61A8- D0 03 4C FE 94 C9 3A F0
61B0- F4 C9 82 D0 2C A9 01 85
61B8- EB 38 A5 EC E9 04 10 02
61C0- A9 00 85 EC AA E8 A9 20
61C8- 20 F2 96 CA D0 FA A2 00
61D0- BD D0 98 F0 06 20 F2 96
61D8- E8 D0 F5 4C FE 95 4C 63
61E0- 96 85 07 A6 EC E8 A9 20
61E8- 20 F2 96 CA D0 FA A5 07
61F0- 30 EC C9 30 90 32 C9 3A
61F8- B0 0D A9 20 20 F2 96 A5
6200- 07 20 F2 96 4C FE 95 C9
6208- 41 90 1D C9 5B B0 19 A2
6210- 00 85 07 BD D7 98 F0 06
6218- 20 F2 96 E8 D0 F5 A5 07
6220- 20 F2 96 4C FE 95 A5 07
6228- C9 22 F0 03 4C D4 96 A5
6230- 1D 49 01 85 1D A9 20 20
6238- F2 96 A9 22 4C E9 96 A9
6240- 3A 4C E9 96 4C FE 94 20
6248- F8 96 F0 F8 30 5E A6 FB
6250- E0 01 D0 0D A2 00 86 FB
6258- 85 07 A9 20 20 F2 96 A5
6260- 07 C9 22 D0 0B A5 1D 49
6268- 01 85 1D A9 22 4C E9 96
6270- C9 3A D0 1A A5 1D D0 C7
```

Source 'T.FORMAT'

Assembleur BIG MAC

Sauvegarde sous format TEXT (voir texte)

```
1           LST  ON
2 *
3 *--------------------------------
4 *!                              !
5 *! FORMATEUR DE LISTINGS BASIC  !
6 *!                              !
7 *! COPYLEFT Gisèle PERREAULT et !
8 *!                              !
9 *!        Sylvie  GALLET        !
10*!                              !
11*--------------------------------
12*
13*--------------------------------
14*! Source pour DOS 3.3/ProDOS    !
15PRODOS   =    0
16*! (mettre 0 pour DOS           !
17*!  ou 1 pour ProDOS)           !
18*!                              !
19*! Assemblable directement pour !
20*! DOS 3.3; utiliser DARWIN pour!
21*! ProDOS:                      !
22*! avec les paramètres:         !
23*! DARWIN_D = $9000             !
24*! DARWIN_F = $9520             !
25*! VA_DEBUT = 1                 !
26*--------------------------------
27*! Syntaxe:                     !
28*! &L   listing à l'écran       !
29*! &LP  listing à l'imprimante  !
30*! &D   saisie de la date       !
31*! &T   saisie du titre         !
32*! &Pn  longueur d'une page     !
33*! &Cn  largeur d'une page      !
34SLOT      =    '1'  slot imprimante
35*--------------------------------
36          DO   1-PRODOS   pas d'ORG pour P
                 roDOS
37          ORG  $90D1
38* OBJ $800
39          FIN
40
41CHRGET    EQU  $B1
42CHRGOT    EQU  $B7
43TXTPTR    EQU  $B8
44LINNUM    EQU  $50
45HIMEM     EQU  $73
46
47FAC       EQU  $9D
48LOWTR     EQU  $9B
49STOCK     EQU  $CE          *$CE-$CF
50NL        EQU  $18          *Compteur lignes
51DEC       EQU  $F9          *Décalage
52MAX       EQU  $FA
53VIRGULE   EQU  $FB
54DF        EQU  $FB
55M         EQU  $FC          *1ère ligne a li
                 ster
56N         EQU  $FE          *Dernière ligne
                 a lister
57AUX       EQU  $06
58MARGE     EQU  $09          *marge gauche
59NLMAX     EQU  $19          *Longueur page
60NC        EQU  $1A          *Compteur caract
61NCMAX     EQU  $1B          *Largeur page
62PAGE      EQU  $1C          *Numéro de page
63QUOTE     EQU  $1D          *Drapeau "
64TF        EQU  $1E          *Décalag si then
65PRT       EQU  $D6
66RF        EQU  $D7
67NF        EQU  $EB
68FF        EQU  $EC          *Décal. for-next
69* ($1F est utilisé par 80 col. du //e)
70*
71GETLN     EQU  $FD6A
72CRDO      EQU  $DAFB
73AMPER     EQU  $3F5
74BASIC     EQU  $FEB3
75STXERR    EQU  $DEC9
76OUTDO     EQU  $DB5C
77LINGET    EQU  $DA0C
78LINPRNT   EQU  $ED24
79FNDLIN    EQU  $D61A
80PRBL3     EQU  $F94C
81
82*
83*--------------------------------
84*
85* INITIALISATION
86*
87*--------------------------------
88*
89          LDA  #$4C
90          STA  AMPER
91          LDA  #DEBUT
92          STA  AMPER+1
93          DO   1-PRODOS   pas touche pas à
                 Himem sous ProDOS!
94          STA  HIMEM
95          FIN
96          LDA  #>DEBUT
97          STA  AMPER+2
98          DO   1-PRODOS
99          STA  HIMEM+1
100         FIN
101         LDA  #$42       * Longueur 66 li
                gnes par défaut
102         STA  NLMAX
103         LDA  #$50       * Largeur 80 par
                défaut
104         STA  NCMAX
105         LDA  #$7        * Largeur marge
                gauche
106         STA  MARGE
107         LDY  #$3F       *
108         LDA  #$03       *
109         STA  STOCK+1    * Efface zone st
                ockage titre et
110         LDA  #$0        * date ($300 à $
                3ff)
111         STA  STOCK      *
112         JSR  V2         *
113         STA  PRT        * Indicateur imp
                rimante
114         RTS
115*
116*--------------------------------
117*
118* COMMANDE LIST
119*
120*--------------------------------
121*
122DEBUT     =    *
123         CMP  #'L'
124         BEQ  L00        *
125         JMP  TITRE      * &L Liste le pr
                ogramme entier
126L00       LDA  #$0        *
127         STA  VIRGULE
128         STA  M
129         STA  M+1
130         LDA  #$FF
131         STA  N
132         STA  N+1
133*
134L01       JSR  CHRGET
135L02       BCC  L08        *
136         CMP  #'P'       * &LP liste sur
                imprimante
137         BEQ  L03        *
138         CMP  #$0
139         BNE  L05
140         JMP  LISTE
141L03      LDA  PRT
142         BEQ  L04
143         DEC  PRT
144         JMP  STXERR
145L04      INC  PRT
146         JMP  L01
```

```
6278- A5 D7 D0 C3 85 EB A9 20
6280- 20 5C DB A9 3A 20 F2 96
6288- 20 31 97 4C 5C 95 C9 2C
6290- F0 03 4C D4 96 A6 EB D0
6298- 03 4C D4 96 A9 20 20 5C
62A0- DB A9 3A 20 F2 96 20 31
62A8- 97 4C 6C 95 A2 00 86 FB
62B0- 38 E9 7F AA 85 07 C9 33
62B8- D0 07 A9 01 85 D7 4C A2
62C0- 96 C9 02 D0 09 A5 EC 69
62C8- 03 85 EC 4C A2 96 C9 45
62D0- D0 19 A2 00 BD DD 98 F0
62D8- 06 20 5C DB E8 D0 F5 A5
62E0- 1E 69 03 85 1E 20 31 97
62E8- 4C 5C 95 A5 07 AA 84 06
62F0- A0 D0 84 9D A0 CF 84 9E
62F8- A0 FF CA F0 0A 20 00 97
6300- 10 FB 30 F6 20 08 97 A9
6308- 20 20 F2 96 20 00 97 30
6310- 05 20 F2 96 D0 F6 20 F2
6318- 96 A4 06 A9 20 C9 0A D0
6320- 0A 20 4D 97 A6 1A A9 A0
6328- 20 4C F9 C9 08 D0 03 4C
6330- FE 95 20 F2 96 20 08 97
6338- 4C FE 95 20 5C DB E6 1A
6340- 60 C8 D0 02 E6 9C B1 9B
6348- 60 C8 D0 02 E6 9E B1 9D
6350- 60 85 07 38 A5 1B E9 0B
6358- 85 FA A6 1A E4 FA 90 2D
6360- A5 07 C9 20 F0 0A C9 2C
6368- F0 06 A6 1A E4 1B D0 1D
6370- A9 04 65 EC 85 F9 A9 01
6378- 85 FB 20 49 97 A5 09 18
6380- 65 1E 65 F9 85 1A AA CA
6388- A9 A0 20 4C F9 A9 00 85
6390- F9 60 A9 00 85 1A 20 FB
6398- DA E6 18 A5 18 C5 19 90
63A0- 03 20 5C 97 60 A5 1C F0
63A8- 08 A2 04 20 FB DA CA D0
63B0- FA 84 07 E6 1C A9 08 85
63B8- 18 A9 20 85 CE 20 B4 9
63C0- 84 1A A5 1B 38 E9 0F E5
63C8- 1A 10 02 A9 02 AA A9 A0
63D0- 20 4C F9 A2 00 BD E3 98
63D8- F0 06 20 5C DB E8 D0 F5
63E0- A9 00 A6 1C 20 24 ED 20
63E8- FB DA 20 FB DA A9 00 85
63F0- CE 20 B4 97 20 FB DA 20
63F8- FB DA A4 07 60 A0 00 B1
6400- CE F0 08 20 5C DB C8 C0
6408- 21 90 F4 60 C9 54 F0 07
6410- C9 44 F0 11 4C 36 98 A9
6418- 00 85 CE A9 03 85 CF 20
6420- 16 98 4C E7 97 A9 20 85
6428- CE A9 03 85 CF 20 16 98
6430- A5 38 48 A5 39 48 A9 FF
6438- 85 38 A9 97 85 39 20 6A
6440- FD E0 21 90 03 4C 20 98
6448- A0 00 B9 00 02 C9 8D F0
6450- 05 91 CE C8 D0 F4 68 85
6458- 39 68 85 38 4C B3 FE A9
6460- 00 A0 1F 91 CE 88 10 FB
6468- 60 20 FB DA A2 00 BD C7
6470- 98 F0 06 20 5C DB E8 D0
6478- F5 20 FB DA 4C B3 FE C9
6480- 50 F0 03 4C 59 98 20 B1
6488- 00 90 03 4C A3 94 20 0C
6490- DA A5 51 D0 30 A5 50 30
6498- 2C C9 15 90 3B 85 19 4C
64A0- A3 94 C9 43 F0 03 4C C9
64A8- DE 20 B1 00 90 03 4C A3
64B0- 94 20 0C DA A5 51 D0 0D
64B8- A5 50 30 09 C9 28 90 34
64C0- 85 1B 4C A3 94 20 FB DA
64C8- A2 00 BD E9 98 F0 06 20
64D0- 5C DB E8 D0 F5 4C A3 94
64D8- 20 FB DA A2 00 BD A2 98
64E0- F0 06 20 5C DB E8 D0 F5
64E8- 4C A3 94 07 07 4D 49 4E
64F0- 2E 32 31 00 20 FB DA A2
64F8- 00 BD BE 98 F0 06 20 5C
6500- DB E8 D0 F5 4C A3 94 07
6508- 07 4D 49 4E 2E 34 30 00
6510- 07 07 4D 41 58 2E 33 32
6518- 00 20 4E 45 58 54 20 00
6520- 20 4C 45 54 20 00 20 54
6528- 48 45 4E 00 50 41 47 45
6530- 20 00 07 07 4D 41 58 2E
6538- 31 32 37 00 AD 14 99 8D
6540- 11 99 A2 00 BD 0E 99 F0
6548- 08 09 80 9D 00 02 E8 D0
6550- F3 20 03 BE A9 00 60 50
6558- 52 23 31 0D 00 31
```

```
147 *
148 L05     CMP  #','
149          BEQ  L06
150          JMP  STXERR
151 *
152 L06     LDA  VIRGULE
153          BEQ  L07
154          JMP  STXERR
155 L07     INC  VIRGULE
156          JMP  L01
157 *
158 L08     JSR  LINGET
159          LDA  VIRGULE
160          BNE  L09
161          LDA  LINNUM
162          STA  M
163          LDA  LINNUM+1
164          STA  M+1
165          JSR  CHRGOT
166          BNE  L02
167 *
168 L09     LDA  LINNUM
169          STA  N
170          LDA  LINNUM+1
171          STA  N+1
172          JSR  CHRGOT
173          BEQ  L10
174          JMP  STXERR
175 *
176 L10     LDA  M+1
177          CMP  N+1
178          BCC  LISTE
179          BNE  RTS
180          LDA  N
181          CMP  M
182          BCS  LISTE
183 *
184 *---------------------------------
185 *
186 * FIN DU PROGRAMME
187 *
188 *---------------------------------
189 *
190 RTS      =    *
191          DO   1-PRODOS
192          LDA  SLOTSAVE
193          BEQ  L12
194 L11     LDA  $C000
195          BPL  L11        attente clavier
                  car PR#3 peut effacer l'éc
                  ran
196          LDA  $C010
197          FIN
198 L12     JSR  GO_PR
199          DO   1-PRODOS
200          JMP  BASIC
201          ELSE
202          JMP  $BE00
203          FIN
204 *
205 *---------------------------------
206 *
207 * LISTE
208 *
209 *---------------------------------
210 *
211 LISTE    =    *
212          DO   1-PRODOS
213          LDA  $AA54
214          ELSE
215          LDA  $BE31
216          FIN
217          EOR  #$FD       ;$FDF0 ?
218          STA  SLOTSAVE
219          BEQ  L20
220          LDA  #'3'       ;sinon slot 3
221          STA  SLOTSAVE
222 L20     LDA  PRT
223          BEQ  L22
224          LDA  #SLOT
225          JSR  GO_PR2
226 L21     DEC  PRT
227 L22     STA  DF
228          STA  NL
229          STA  NC
230          STA  QUOTE
231          STA  PAGE
232          STA  TF
233          STA  FF
234          STA  RF
235          LDA  M
236          STA  LINNUM
237          LDA  M+1
238          STA  LINNUM+1
239          JSR  FNDLIN
240          LDY  #$3
241          BCS  L23
242          LDY  #$0
243          BEQ  PREMLGN
244          JMP  CHAINAGE
245 *
246 RTS1     JMP  RTS
247 *
248 L23     JSR  CRDO1
249          JSR  ENTETE
250          JMP  L26
251 PREMLGN  =    *
252          JSR  CRDO1
253          JSR  ENTETE
254          JMP  L24
255 CHAINAGE JSR  LIRE1
256 L24     JSR  LIRE1
257          BEQ  RTS        * Fin programme
258          LDA  $C000      * Arrêt du lis-
259          BPL  L25        * ting si Ctrl C
260          CMP  #$83
261          BNE  L25
262          LDA  $C010
263          JMP  RTS
264 L25     JSR  LIRE1
265          STA  LINNUM
266          JSR  LIRE1
267          STA  LINNUM+1
268          CMP  N+1
269          BCC  L26
270          BNE  RTS1
271          LDA  N
272          CMP  LINNUM
273          BCC  RTS1
274 L26     JSR  CRDO1
275          JSR  CRDO1
276          LDA  LINNUM+1
277          LDX  LINNUM
278          STY  AUX+1
279          JSR  LINPRNT    * Impression du
                  no de ligne
280          STY  AUX
281          LDA  #$0        *
282          CLC  *
283          ADC  #$7        *
284          STA  NC         * Marge après nu
                  méro de ligne
285          SEC  *
286          SBC  AUX        *
287          TAY  *
288          DEY  *
289          LDA  #$20       *
290 L27     JSR  OUTDO
291          DEY
292          BNE  L27
293          LDA  #$0
294          STA  TF
295          STA  QUOTE
296          STA  RF
297          STA  NF
298          LDY  AUX+1
299 *
300 *---------------------------------
301 *
302 * LECTURE 1ER OCTET D'UNE LIGNE
303 *
304 *---------------------------------
305 *
306 DEBLGN   JSR  LIRE1
307          BNE  L31
308          JMP  CHAINAGE
309 L31     CMP  #':'       * Elimine les ":
                  " en trop
```

Récapitulation en version DOS FORMAT

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE FORMAT,A$90D1,L$527

```
90D1- A9 4C 8D F5 03 A9 00
90D8- 8D F6 03 85 73 A9 91 8D
90E0- F7 03 85 74 A9 42 85 19
90E8- A9 50 85 1B A9 07 85 09
90F0- A0 3F A9 03 85 CF A9 00
90F8- 85 CE 20 FC 94 85 D6 60
9100- C9 4C F0 03 4C A5 94 A9
9108- 00 85 FB 85 FC 85 FD A9
9110- FF 85 FE 85 FF 20 B1 00
9118- 90 2C C9 50 F0 07 C9 00
9120- D0 11 4C 8B 91 A5 D6 F0
9128- 05 C6 D6 4C C9 DE E6 D6
9130- 4C 15 91 C9 2C F0 03 4C
9138- C9 DE A5 FB F0 03 4C C9
9140- DE E6 FB 4C 15 91 20 0C
9148- DA A5 FB D0 0D A5 50 85
9150- FC A5 51 85 FD 20 B7 00
9158- D0 BE A5 50 85 FE A5 51
9160- 85 FF 20 B7 00 F0 03 4C
9168- C9 DE A5 FD C5 FF 90 1B
9170- D0 06 A5 FE C5 FC B0 13
9178- AD F7 95 F0 08 AD 00 C0
9180- 10 FB AD 10 C0 20 D5 95
9188- 4C B3 FE AD 54 AA 49 FD
9190- 8D F7 95 F0 05 A9 33 8D
9198- F7 95 A5 D6 F0 07 A9 31
91A0- 20 D8 95 C6 D6 85 FB 85
91A8- 18 85 1A 85 1D 85 1C 85
91B0- 1E 85 EC 85 D7 A5 FC 85
91B8- 50 A5 FD 85 51 20 1A D6
91C0- A0 03 B0 0A A0 00 F0 0F
91C8- 4C E0 91 4C 78 91 20 2B
91D0- 94 20 4A 94 4C 0D 92 20
91D8- 2B 94 20 4A 94 4C E3 91
91E0- 20 DA 93 20 DA 93 F0 90
91E8- AD 00 C0 10 0A C9 83 D0
91F0- 06 AD 10 C0 4C 78 91 20
91F8- DA 93 85 50 20 DA 93 85
9200- 51 C5 FF 90 08 D0 C4 A5
9208- FE C5 50 90 BE 20 2B 94
9210- 20 2B 94 A5 51 A6 50 84
9218- 07 20 24 ED 84 06 A9 00
9220- 18 69 07 85 1A 38 E5 06
9228- A8 88 A9 20 20 5C DB 88
9230- D0 FA A9 00 85 1E 85 1D
9238- 85 D7 85 EB A4 07 20 DA
9240- 93 D0 03 4C E0 91 C9 3A
9248- F0 F4 C9 82 D0 2C A9 01
9250- 85 EB 38 A5 EC E9 04 10
9258- 02 A9 00 85 EC AA E8 A9
9260- 20 20 D4 93 CA D0 FA A2
9268- 00 BD B2 95 F0 06 20 D4
9270- 93 E8 D0 F5 4C E0 92 4C
9278- 45 93 85 07 A6 EC E8 A9
9280- 20 20 D4 93 CA D0 FA A5
9288- 07 30 EC C9 30 90 32 C9
9290- 3A B0 0D A9 20 20 D4 93
9298- A5 07 20 D4 93 4C E0 92
92A0- C9 41 90 1D C9 5B B0 19
92A8- A2 00 85 07 BD B9 95 F0
92B0- 06 20 D4 93 E8 D0 F5 A5
92B8- 07 20 D4 93 4C E0 92 A5
92C0- 07 C9 22 F0 03 4C B6 93
92C8- A5 1D 49 01 85 1D A9 20
92D0- 20 D4 93 A9 22 4C CB 93
92D8- A9 3A 4C CB 93 4C E0 91
92E0- 20 DA 93 F0 F8 30 5E A6
92E8- FB E0 01 D0 0D A2 00 86
92F0- FB 85 07 A9 20 20 D4 93
92F8- A5 07 C9 22 D0 0B A5 1D
9300- 49 01 85 1D A9 22 4C CB
9308- 93 C9 3A D0 1A A5 1D D0
9310- C7 A5 D7 D0 C3 85 EB A9
9318- 20 20 5C DB A9 3A 20 D4
9320- 93 20 13 94 4C 3E 92 C9
```

```
310          BEQ  DEBLGN
311          CMP  #$82        * Next ?
312          BNE  L41
313 NEXT     LDA  #$1
314          STA  NF
315          SEC  *           Réduit le décala
                ge et imprime
316          LDA  FF          * NEXT
317          SBC  #$4
318          BPL  L32
319          LDA  #$0
320 L32      STA  FF
321          TAX
322          INX
323          LDA  #$20
324 L33      JSR  OUTDO1
325          DEX
326          BNE  L33
327          LDX  #$0
328 L34      LDA  NXT,X
329          BEQ  L35
330          JSR  OUTDO1
331          INX
332          BNE  L34
333 L35      JMP  LIRE
334 *
335 TOKEN1   JMP  TOKEN
336 *
337 L41      STA  AUX+1       * Effectue déca-
338          LDX  FF          * lage for-next
339          INX
340          LDA  #$20
341 L42      JSR  OUTDO1
342          DEX
343          BNE  L42
344 L43      LDA  AUX+1
345          BMI  TOKEN1
346          CMP  #'0'        * Si numéro de
347          BCC  L47         * ligne après
348          CMP  #':'        * THEN décale
                d'1 espace en +
349          BCS  L44
350          LDA  #$20
351          JSR  OUTDO1
352          LDA  AUX+1
353          JSR  OUTDO1
354          JMP  LIRE
355 L44      CMP  #'A'        * Si nom de va-
356          BCC  L47         * riable en débu
                t instruction imprime LET
357          CMP  #$5B        *
358          BCS  L47
359          LDX  #$0
360          STA  AUX+1
361 L45      LDA  LET,X
362          BEQ  L46
363          JSR  OUTDO1
364          INX
365          BNE  L45
366 L46      LDA  AUX+1
367          JSR  OUTDO1
368          JMP  LIRE
369          LDA  AUX+1       * Le caractère l
                u n'est pas un nom
370 L47      CMP  #'"'
371          BEQ  L48
372          JMP  PRINT
373 L48      LDA  QUOTE
374          EOR  #$1
375          STA  QUOTE
376          LDA  #$20
377          JSR  OUTDO1
378          LDA  #'"'
379          JMP  PRINT2
380 *
381 CHAINE    LDA  #':'
382          JMP  PRINT2      * Le : est entre
                guillemets
383 *
384 XX       JMP  CHAINAGE
385 *
386 LIRE     JSR  LIRE1
387          BEQ  XX          * 0 de fin de li
                gne Basic
388 L51      BMI  TOKEN
389          LDX  DF
390          CPX  #$1
391          BNE  L52
392          LDX  #$0
393          STX  DF
394          STA  AUX+1
395          LDA  #$20        * Décale de 1 es
                p. le 1er caractère
396          JSR  OUTDO1      * alphanumérique
                si ligne coupée
397          LDA  AUX+1
398 L52      CMP  #'"'
399          BNE  L53
400          LDA  QUOTE
401          EOR  #$01
402          STA  QUOTE
403          LDA  #'"'
404          JMP  PRINT2
405 L53      CMP  #':'
406          BNE  L54
407          LDA  QUOTE
408          BNE  CHAINE
409          LDA  RF
410          BNE  CHAINE
411          STA  NF
412          LDA  #$20        * ":" Séparateur
413          JSR  OUTDO
414          LDA  #':'
415          JSR  OUTDO1
416          JSR  NI          * Saut ligne pou
                r nouvelle inst.
417          JMP  DEBLGN
418 L54      CMP  #','
419          BEQ  L55
420          JMP  PRINT
421 L55      LDX  NF
422          BNE  L56
423          JMP  PRINT
424 L56      LDA  #$20
425          JSR  OUTDO
426          LDA  #':'
427          JSR  OUTDO1
428          JSR  NI
429          JMP  NEXT
430 *
431 *---------------------------------
432 *
433 * LECTURE ET IMPRESSION DES TOKENS
434 *
435 *---------------------------------
436 *
437 TOKEN    LDX  #$0         * Annule indicat
                eur coupure de ligne
438 *
439          STX  DF
440 *
441          SEC
442          SBC  #$7F
443          TAX
444          STA  AUX+1       *
445          CMP  #$1
446          BNE  L60
447          LDA  #1
448          STA  RF
449          JMP  L64
450 L60      CMP  #$2         * Début FOR NEXT
451          BNE  L61         * Augmente compt
                eur décalage
452          LDA  FF
453          ADC  #$3
454          STA  FF
455          JMP  L64
456 L61      CMP  #$45
457          BNE  L64
458          LDX  #$0
459 L62      LDA  THEN,X
460          BEQ  L63
461          JSR  OUTDO
462          INX
463          BNE  L62
464 L63      LDA  TF
465          ADC  #$3
466          STA  TF
```

```
9328- 2C F0 03 4C B6 93 A6 EB
9330- D0 03 4C B6 93 A9 20 20
9338- 5C DB A9 3A 20 D4 93 20
9340- 13 94 4C 4E 92 A2 00 86
9348- FB 38 E9 7F AA 85 07 C9
9350- 33 D0 07 A9 01 85 D7 4C
9358- 84 93 C9 02 D0 09 A5 EC
9360- 69 03 85 EC 4C 84 93 C9
9368- 45 D0 19 A2 00 BD BF 95
9370- F0 06 20 5C DB E8 D0 F5
9378- A5 1E 69 03 85 1E 20 13
9380- 94 4C 3E 92 A5 07 AA 84
9388- 06 A0 D0 84 9D A0 CF 84
9390- 9E A0 FF CA F0 0A 20 E2
9398- 93 10 FB 30 F6 20 EA 93
93A0- A9 20 20 D4 93 20 E2 93
93A8- 30 05 20 D4 93 D0 F6 20
93B0- D4 93 A4 06 A9 20 C9 0A
93B8- D0 0A 20 2F 94 A6 1A A9
93C0- A0 20 4C F9 C9 08 D0 03
93C8- 4C E0 92 20 D4 93 20 EA
93D0- 93 4C E0 92 20 5C DB E6
93D8- 1A 60 C8 D0 02 E6 9C B1
93E0- 9B 60 C8 D0 02 E6 9E B1
93E8- 9D 60 85 07 38 A5 1B E9
93F0- 0B 85 FA A6 1A E4 FA 90
93F8- 2D A5 07 C9 20 F0 0A C9
9400- 2C F0 06 A6 1A E4 1B D0
9408- 1D A9 04 65 EC 85 F9 A9
9410- 01 85 FB 20 2B 94 A5 09
9418- 18 65 1E 65 F9 85 1A AA
9420- CA A9 A0 20 4C F9 A9 00
9428- 85 F9 60 A9 00 85 1A 20
9430- FB DA E6 18 A5 18 C5 19
9438- 90 03 20 3E 94 60 A5 1C
9440- F0 08 A2 04 20 FB DA CA
9448- D0 FA 84 07 E6 1C A9 08
9450- 85 18 A9 20 85 CE 20 96
9458- 94 84 1A A5 1B 38 E9 0F
9460- E5 1A 10 02 A9 02 AA A9
9468- A0 20 4C F9 A2 00 BD C5
9470- 95 F0 06 20 5C DB E8 D0
9478- F5 A9 00 A6 1C 20 24 ED
9480- 20 FB DA 20 FB DA A9 00
9488- 85 CE 20 96 94 20 FB DA
9490- 20 FB DA A4 07 60 A0 00
9498- B1 CE F0 08 20 5C DB C8
94A0- C0 21 90 F4 60 C9 54 F0
94A8- 07 C9 44 F0 11 4C 18 95
94B0- A9 00 85 CE A9 03 85 CF
94B8- 20 F8 94 4C C9 94 A9 20
94C0- 85 CE A9 03 85 CF 20 F8
94C8- 94 A5 38 48 A5 39 48 A9
94D0- E1 85 38 A9 94 85 39 20
94D8- 6A FD E0 21 90 03 4C 02
94E0- 95 A0 00 B9 00 02 C9 8D
94E8- F0 05 91 CE C8 D0 F4 68
94F0- 85 39 68 85 38 4C B3 FE
94F8- A9 00 A0 1F 91 CE 88 10
9500- FB 60 20 FB DA A2 00 BD
9508- A9 95 F0 06 20 5C DB E8
9510- D0 F5 20 FB DA 4C B3 FE
9518- C9 50 F0 03 4C 3B 95 20
9520- B1 00 90 03 4C 78 91 20
9528- 0C DA A5 51 D0 30 A5 50
9530- 30 2C C9 15 90 3B 85 19
9538- 4C 78 91 C9 43 F0 03 4C
9540- C9 DE 20 B1 00 90 03 4C
9548- 78 91 20 0C DA A5 51 D0
9550- 0D A5 50 30 09 C9 28 90
9558- 34 85 1B 4C 78 91 20 FB
9560- DA A2 00 BD CB 95 F0 06
9568- 20 5C DB E8 D0 F5 4C 78
9570- 91 20 FB DA A2 00 BD 84
9578- 95 F0 06 20 5C DB E8 D0
9580- F5 4C 78 91 07 07 4D 49
9588- 4E 2E 32 31 00 20 FB DA
9590- A2 00 BD A0 95 F0 06 20
9598- 5C DB E8 D0 F5 4C 78 91
95A0- 07 07 4D 49 4E 2E 34 30
95A8- 00 07 07 4D 41 58 2E 33
95B0- 32 00 20 4E 45 58 54 20
95B8- 00 20 4C 45 54 20 00 20
95C0- 54 48 45 4E 00 50 41 47
95C8- 45 20 00 07 07 4D 41 58
95D0- 2E 31 32 37 00 AD F7 95
95D8- 8D F4 95 20 FB DA A9 04
95E0- 20 5C DB A2 00 BD F1 95
95E8- F0 06 20 5C DB E8 D0 F5
95F0- 60 50 52 23 31 0D 00 31
```

```
467             JSR  NI
468             JMP  DEBLGN
469 L64         LDA  AUX+1
470             TAX
471             STY  AUX
472             LDY  #$D0
473             STY  FAC
474             LDY  #$CF
475             STY  FAC+1
476             LDY  #$FF
477 L65         DEX
478             BEQ  L67
479 L66         JSR  LIRE2
480             BPL  L66
481             BMI  L65
482             JSR  SAUTLGN
483 L67         LDA  #$20
484             JSR  OUTDO1
485 L68         JSR  LIRE2
486             BMI  L69
487             JSR  OUTDO1
488             BNE  L68
489 L69         JSR  OUTDO1
490             LDY  AUX
491             LDA  #$20
492 *
493 PRINT       =    *
494             CMP  #$0A
495             BNE  L72
496             JSR  CRDO2
497             LDX  NC
498             LDA  #$A0
499             JSR  PRBL3
500 L72         CMP  #$8
501             BNE  PRINT2
502             JMP  LIRE
503 PRINT2      JSR  OUTDO1
504             JSR  SAUTLGN
505             JMP  LIRE
506 *
507 *--------------------------------
508 *
509 * SOUS PROGRAMMES
510 *
511 *--------------------------------
512 *
513 OUTDO1      =    *
514             JSR  OUTDO
515             INC  NC
516             RTS
517 LIRE1       =    *
518             INY  *          LIT DANS BASIC
519             BNE  SL1
520             INC  LOWTR+1
521 SL1         LDA  (LOWTR),Y
522             RTS
523 *
524 LIRE2       =    *
525             INY  *          LIT DANS LA TABL
                      E DES MOTS CLES
526             BNE  SL2
527             INC  FAC+1
528 SL2         LDA  (FAC),Y
529             RTS
530 *
531 SAUTLGN     =    *
532             STA  AUX+1
533             SEC
534             LDA  NCMAX      * A 11 caract. d
                      e fin de ligne
535 * coupe
536 * * la ligne si " " ou ","
537             SBC  #$B
538             STA  MAX
539             LDX  NC
540             CPX  MAX
541             BCC  RETURN
542             LDA  AUX+1
543             CMP  #'
544             BEQ  DECALE
545             CMP  #','
546             BEQ  DECALE
547             LDX  NC
548             CPX  NCMAX
549             BNE  RETURN
550 DECALE      =    *
551             LDA  #$4
552             ADC  FF
553             STA  DEC        * Décalage a eff
                      ectuer
554             LDA  #$1        * Marqueur coupu
                      re de ligne
555             STA  DF
556 NI          JSR  CRDO1      * Décalage en dé
                      but d'instruction
557             LDA  MARGE
558             CLC
559             ADC  TF
560             ADC  DEC
561             STA  NC
562             TAX
563             DEX
564             LDA  #$A0
565             JSR  PRBL3
566 RETURN      =    *
567             LDA  #$0        * Annule décalag
                      e local
568             STA  DEC
569             RTS
570 CRDO1       =    *
571             LDA  #$0
572             STA  NC
573 CRDO2       JSR  CRDO
574             INC  NL
575             LDA  NL
576             CMP  NLMAX
577             BCC  SL11
578             JSR  SL12
579 SL11        RTS  *          Même page
580 SL12        LDA  PAGE       * Page suivante
581             BEQ  ENTETE     * Si 1ère page n
                      e fait pas de saut
582             LDX  #$4        * de 4 lignes
583 SL13        JSR  CRDO
584             DEX
585             BNE  SL13
586 ENTETE      STY  AUX+1
587             INC  PAGE
588             LDA  #$8
589             STA  NL
590             LDA  #$20
591             STA  STOCK      * DATE
592             JSR  PRTMG
593             STY  NC
594             LDA  NCMAX
595             SEC
596             SBC  #$F
597             SBC  NC
598             BPL  SL21
599             LDA  #$2
600 SL21        TAX
601             LDA  #$A0
602             JSR  PRBL3
603             LDX  #$0
604 SL22        LDA  NUMPAGE,X  * Imprime "page
605             BEQ  SL23
606             JSR  OUTDO
607             INX
608             BNE  SL22
609 SL23        LDA  #$0
610             LDX  PAGE
611             JSR  LINPRNT
612             JSR  CRDO
613             JSR  CRDO
614             LDA  #$0
615             STA  STOCK      * TITRE
616             JSR  PRTMG      * Imprime titre
617             JSR  CRDO
618             JSR  CRDO
619             LDY  AUX+1
620             RTS
621 *
622 *
623 PRTMG       LDY  #$0
624 SL41        LDA  (STOCK),Y
625             BEQ  SL42
626             JSR  OUTDO
627             INY
628             CPY  #$21
629             BCC  SL41
630 SL42        RTS
631 *
632 *
633 TITRE       =    *
634             CMP  #'T'       * Saisie du titr
                      e ou de la date
635             BEQ  TTR
636             CMP  #'D'
637             BEQ  DTE
638             JMP  LONGPG
639 *
640 * Sélection entrée date ou titre
641 *
642 TTR         LDA  #$0
643             STA  STOCK
644             LDA  #$03
645             STA  STOCK+1
646             JSR  VIDE
647             JMP  SL5
648 DTE         LDA  #$20
649             STA  STOCK
650             LDA  #$03
651             STA  STOCK+1
652             JSR  VIDE
653 SL5         LDA  $38
654             PHA
655             LDA  $39
656             PHA
657             LDA  #INPUT
658             STA  $38
659             LDA  #>INPUT
660             STA  $39
661             JSR  GETLN
662             CPX  #$21
663             BCC  INPUT
664             JMP  ERR
665 INPUT       LDY  #$0
666 SL61        LDA  $200,Y     * Transfert dans
                      zone de stockage
667             CMP  #$8D
668             BEQ  SL62
669             STA  (STOCK),Y
670             INY
671             BNE  SL61
672 SL62        PLA
673             STA  $39
674             PLA
675             STA  $38
676             JMP  BASIC
677 *
678 VIDE        =    *
679             LDA  #$0
680             LDY  #$1F
681 V2          STA  (STOCK),Y
682             DEY
683             BPL  V2
684             RTS
685 *
686 ERR         JSR  CRDO
687             LDX  #$0
688 SL71        LDA  MX,X
689             BEQ  SL72
690             JSR  OUTDO
691             INX
692             BNE  SL71
693 SL72        JSR  CRDO
694             JMP  BASIC
695 LONGPG      =    *
696             CMP  #'P'       * Longueur page
697             BEQ  SL80
698             JMP  LARGPG
699 SL80        JSR  CHRGET
700             BCC  SL81
701             JMP  RTS        *
702 SL81        JSR  LINGET
703             LDA  LINNUM+1
704             BNE  ERREUR
705             LDA  LINNUM
706             BMI  ERREUR     * Maxi 127 ligne
                      s
707             CMP  #$15
708             BCC  MINL
709             STA  NLMAX
710             JMP  RTS
711 *
712 LARGPG      =    *
713             CMP  #'C'       * Largeur de la
                      page
714             BEQ  SL90
715             JMP  STXERR
716 SL90        JSR  CHRGET
717             BCC  SL91
718             JMP  RTS
719 SL91        JSR  LINGET
720             LDA  LINNUM+1
721             BNE  ERREUR
722             LDA  LINNUM
723             BMI  ERREUR
724             CMP  #$28
725             BCC  MINC
726             STA  NCMAX
727             JMP  RTS
728 *
729 ERREUR      =    *
730             JSR  CRDO
731             LDX  #$0
732 SL101       LDA  MAXI,X
733             BEQ  SL102
734             JSR  OUTDO
735             INX
736             BNE  SL101
737 SL102       JMP  RTS
738 *
739 MINL        =    *
740             JSR  CRDO
741             LDX  #$0
742 SL111       LDA  MNL,X
743             BEQ  SL112
744             JSR  OUTDO
745             INX
746             BNE  SL111
747 SL112       JMP  RTS
748 MNL         HEX  0707
749             ASC  'MIN.21'
750             HEX  00
751 *
752 MINC        =    *
753             JSR  CRDO
754             LDX  #$0
755 SL121       LDA  MNC,X
756             BEQ  SL122
757             JSR  OUTDO
758             INX
759             BNE  SL121
760 SL122       JMP  RTS
761 MNC         HEX  0707
762             ASC  'MIN.40'
763             HEX  00
764 *
765 MX          HEX  0707
766             ASC  'MAX.32'
767             HEX  00
768 NXT         ASC  ' NEXT '
```

Navette

Jean-Paul Arbel

L'Applesoft est un peu à l'étroit dans la zone mémoire réservée à votre imagination débordante. Les 48Ko qui vous sont généreusement alloués se revèlent parfois insuffisants lorsque vos programmes Basic utilisent des images HGR et ou des routines en langage machine importantes. Si, de plus, votre programme utilise des tableaux alphanumériques, il ne vous reste qu'une peau de... pomme.

Le programme NAVETTE va vous permettre d'utiliser les 16 Ko de la carte langage pour placer l'ensemble des programmes binaires dont a besoin votre programme Basic.

Les solutions

Programmes binaires n'utilisant pas de routines Applesoft

Le plus simple consiste à recopier le Moniteur de la ROM vers la RAM aux mêmes adresses (de $F800 à $FFFF). Tout appel au Moniteur à partir de la carte langage sera alors dirigé vers la RAM. Ainsi, à chaque utilisation de notre programme binaire, une routine en RAM principale (sous l'adresse $9600), se chargera de transférer le contrôle à la carte langage, puis de reprendre la main en finale en commutant vers la ROM avant de retourner au programme Basic.

Programmes utilisant des routines Moniteur et Applesoft

Le problème se complique car il y a conflit entre l'utilitaire en langage machine et la ROM Applesoft ; ceux-ci commencent en effet à la même adresse (à partir de $D000). Comme dans le cas précédent, une routine fera la liaison entre le programme Basic et la carte langage, mais elle servira de plus d'aiguillage pour tout appel (et retour) de routine ROM (Applesoft ou Moniteur). C'est l'objet du programme NAVETTE.

Principe

Le programme NAVETTE va servir d'interface entre le programme en 'carte langage' et les sous-programmes en ROM. Son principe repose sur une utilisation de l'instruction BRK pour passer le relais à NAVETTE. Il suffit de placer, dans le programme destiné à être logé dans la RAM supérieure, un BREAK (BRK:00) avant chaque appel d'une routine de la ROM. A la rencontre de ce code, le moniteur saute à l'adresse vectorisée par $FFFE et $FFFF, en l'occurence, on saute à NAVETTE qui, comme sont nom l'indique, fera la navette entre la RAM et la ROM.

Pratiquement...

Dans votre programme Basic, charger NAVETTE et l'utilitaire par une séquence telle que :

```
10  D$= CHR$(4): PRINT D$"B
    RUN NAVETTE"
20  A = PEEK (-16255) : A =
    PEEK (-16255): PRINT D$
    "BLOAD UTILITAIRE": A =
    PEEK (-16254)
```

L'utilitaire sera ensuite exécuté par l'instruction '&' dite ampersand.

Modification de l'utilitaire

Assembler le programme que l'on veut charger dans la carte langage, à partir de l'adresse $D000, en ayant pris soin de placer des BRK avant chaque appel à la ROM par JSR ou JMP ou avant chaque instruction accédant à la ROM (par exemple LDA $FBB3 qui détermine le type de la machine).

Il est à noter que tout programme existant peut être modifié et assemblé de la sorte. Si ce programme utilise l'ampersand, il suffit de supprimer la partie

---

Suite du source 'T.FORMAT' listé pages précédentes

```
769             HEX   00
770 LET         ASC   ' LET '
771             HEX   00
772 THEN        ASC   ' THEN'
773             HEX   00
774 NUMPAGE     ASC   'PAGE '
775             HEX   00
776 MAXI        HEX   0707
777             ASC   'MAX.127'
778             HEX   00
779
780 GO_PR       =     *
781             LDA   SLOTSAVE
782 GO_PR2      STA   PR_DATA+3  ;acc=slot destin
                      ataire
783             DO    1-PRODOS
784             JSR   CRDO
785             LDA   #4              ctrl-d
786             JSR   OUTDO
787             FIN
788             LDX   #0
789 GO2         LDA   PR_DATA,X
790             BEQ   GO3
791             DO    1-PRODOS
792             JSR   OUTDO
793             ELSE
794             ORA   #$80
795             STA   $200,X
796             FIN
797             INX
798             BNE   GO2
799 GO3         =     *
800             DO    PRODOS
801             JSR   $BE03
802             LDA   #0
803             FIN
804             RTS
805
806 PR_DATA     ASC   'PR#1'
807             HEX   0D00
808 SLOTSAVE    ASC   '1'
809             DO    1-PRODOS
810             END
811             FIN
```

relative à l'initialisation de celui-ci. Par contre les instructions Basic utilisant cet artifice restent inchangées. L'utilisation de NAVETTE à l'intérieur d'un programme Basic est entièrement transparente.

Facilités

Le programme CONVERT permet d'ajouter les BRK dans les sources devant les instructions JSR et JMP appelant un sous-programme en ROM.

Ce programme se charge d'identifier toutes les adresses en ROM, définies par des labels grâce à l'opcode "EQU" ou "=", d'un programme source écrit avec BIG MAC, sauvegardé sous forme de fichier TEXT. Il insère ensuite les BRK automatiquement ; il sauve enfin sur disquette, le programme modifié, en ajoutant le suffixe ".LC".

Il utilise les routines suivantes :

- INPUT : input généralisé acceptant tous caractères ;
- DIM.VAR.OBJ : routine publiée dans Pom's 12 ;
- FRE (16) : routine de 'garbage' publiée dans Pom's 2.

Techniquement...

A chaque appel de l'ampersand le scénario est le suivant :

1] NAVETTE commute sur la mémoire supérieure (carte langage) ;

2] exécution de l'utilitaire jusqu'à la rencontre d'un BRK qui dirige sur NAVETTE ;

3] les registres sont alors sauvegardés. On recherche ensuite l'instruction suivant BRK. On commute alors l'accès à la ROM et cette instrution est exécutée ;

4] enfin, NAVETTE commute de nouveau sur notre utilitaire ;

5] le retour au Basic ne saurait se réaliser sans un petit passage (le dernier jusqu'au prochain) par NAVETTE, qui se charge de replacer correctement les commutateurs RAM/ROM, avant de rendre la main au programme Basic.

NAVETTE a été testé sur ICARE de P. CANTOT, version complète (SUPER PRINT et TORTUE inclus). Avec quelques aménagements bien sûr, il semble donner toute satisfaction.

Précisions, limites

- NAVETTE est assemblé en page 3 à partir de l'adresse $300, l'utilitaire étant placé au-dessus du DOS, HIMEM conserve sa valeur fixée par le DOS soit $9600 (38400) ;
- NAVETTE est incompatible avec les DOS déplacés dans la carte langage, de même qu'avec PRODOS qui utilise lui aussi cet espace mémoire ;
- NAVETTE commute sur le Banc 2 et la mémoire commune de 8Ko. Pour commuter sur le Banc 1 voir la remarque 5. Pour plus de précisions voir l'article de Gérard Michel dans le numéro 19 de Pom's, page 29 ;
- il ne faut pas que l'utilitaire fasse appel à des interruptions (BRK), les vecteurs d'interruptions étant modifiés par NAVETTE ;
- si vous voulez placer plusieurs utilitaires indépendants sur la carte langage, il vous suffit, avant l'appel de l'un d'eux, de POKer son adresse d'implantation en $31E (798) et $31F (799) dans l'ordre habituel : adresse basse puis haute. Si vous voulez utilisez le BANC 1, POKez 0 à la place de 128 en $324 (804).

Programme 'CONVERT'

```
10  REM
    !     INSERTION DES BRK     !
    !  DANS PGM SOURCE BIG MAC  !
    !                           !
    !      JEAN PAUL ARBEL      !
    !                           !
20  HIMEM: 37490: HOME : TEXT
30  REM  AMPERSAND -> DIM.VAR
40  POKE 1013,76: POKE 1014,124: POK
    E 1015,146
50  DIM A$(1): DIM Z$(1): DIM F$(3)
60 F$(1) = "INPUT":F$(2) = "DIM.VAR.
    OBJ":F$(3) = "FRE(16)"
70 D$ =  CHR$ (13) +  CHR$ (4)
80  ONERR  GOTO 100
90  FOR I = 1 TO 2:F$ = F$(I): PRINT
     D$"BLOAD"F$(I): NEXT : GOTO 110
100  CALL 768: GOSUB 1080: GOTO 80
110 Z1$ = " BRK" +  CHR$ (13) + " JS
    R "
120 Z2$ = " BRK" +  CHR$ (13) + " JM
    P "
130  REM  LECTURE FICHIER
140 VT = 10
150  VTAB VT: INVERSE :R$ = "NOM DU
    FICHIER SOURCE A TRANSFORMER?":
    GOSUB 1010
160  VTAB VT + 1: HTAB 3: CALL 782 I
    NPUT F1$
170  ONERR  GOTO 190
180  PRINT D$"OPEN"F1$: GOTO 210
190 E =  PEEK (222): IF E = 4 THEN E
    = 7
195  IF E <  > 11 THEN  PRINT D$"CLO
    SE"F1$
200  CALL 768: GOSUB 1090: GOTO 150
210  PRINT D$"READ"F1$
220  ONERR  GOTO 250
230  HOME :R$ = "CHARGEMENT DU FICHI
    ER:": GOSUB 1010: VTAB 5: INVERS
    E :R$ = F1$: GOSUB 1010
240  FOR I = 1 TO 2000: CALL 782 INP
    UT A$(I): & A$ + : NEXT
250  CALL 768:N1 = I - 1:J = 1
260  PRINT D$"CLOSE"F1$
270  IF I < 2 THEN  PRINT D$"DELETE"
    F1$:E = 7: GOTO 200
280  REM   RECHERCHE DES LABELS...
290  REM  ...APPLESOFT OU MONITEUR
300  HOME :R$ = "VOICI LA LISTE DES
    LABELS": GOSUB 1010: VTAB 3:R$ =
     "DE LA ROM APPLESOFT OU MONITEU
    R:": GOSUB 1010
310  FOR I = 1 TO N1:K =  LEN (A$(I)
    ) - 6: IF K < 3 THEN 410
320 B$ =  MID$ (A$(I),K,4):C$ =  LEF
    T$ (B$,1): IF C$ = "=" OR C$ = "
    U" THEN 340
330  GOTO 410
340 B$ =  RIGHT$ (B$,2)
350  IF B$ = "$D" OR B$ = "$E" OR B$
     = "$F" THEN 370
360  GOTO 410
370  IF C$ = "=" THEN Z$(J) =  LEFT$
     (A$(I),K - 2): & Z$ + : GOTO 39
    0
380  IF C$ = "U" THEN Z$(J) =  LEFT$
     (A$(I),K - 4): & Z$ +
390  PRINT Z$(J)
400 J = J + 1
410  NEXT
420 N2 = J - 1
430  REM  INCLUSION DES BRK
440  ONERR  GOTO 460
450 F$ = F$(3): PRINT D$"BLOAD"F$: G
    OTO 470
460  CALL 768: GOSUB 1080: GOTO 440
470  CALL 37500
480  FOR I = 1 TO N1
490  IF  INT (I / 500) = I / 500 THE
    N  PRINT : PRINT "PATIENCE JE FA
    IS LE MENAGE!";: CALL 37500: PRI
    NT "C'EST FINI": PRINT
500  FOR J = 1 TO 12
```

```
510  IF  MID$ (A$(I),J,1) <  > "J" T
     HEN 670
520 J = J + 1: IF  MID$ (A$(I),J,1)
     <  > "S" THEN 550
530 J = J + 1: IF  MID$ (A$(I),J,1)
     <  > "R" THEN 670
540 Z$ = Z1$: GOTO 590
550  IF  MID$ (A$(I),J,1) <  > "M" T
     HEN 670
560 J = J + 1: IF  MID$ (A$(I),J,1)
     <  > "P" THEN 670
570 Z$ = Z2$
580  REM  ON A UN JSR OU JMP
590 M = J + 2
600 Y$ = ""
610  IF J > 4 THEN Y$ =  LEFT$ (A$(I
     ),J - 4)
620  FOR K = 1 TO N2
630 L =  LEN (Z$(K))
640  IF  LEN (A$(I)) - M + 1 < L THE
     N 660
650  IF  MID$ (A$(I),M,L) = Z$(K) TH
     EN A$(I) = Y$ + Z$ + Z$(K): PRIN
     T A$(I),"LIGNE N[ ";I: PRINT :K
     = N2
660  NEXT K:J = 12
670  NEXT J: NEXT I
680  HOME : PRINT "VOULEZ-VOUS SAUVE
     R LE FICHIER TRANSFORME": PRINT
     : PRINT "           SUR DISQUETTE
     ? <O,N>";
690  GET R$
700  IF R$ = "N" THEN  END
710  IF R$ <  > "O" THEN 690
720  PRINT
730 F2$ = F1$ + ".LC"
740  ONERR  GOTO 760
750  PRINT D$"OPEN"F2$: PRINT D$"DEL
     ETE"F2$: PRINT D$"OPEN"F2$: PRIN
     T D$"WRITE"F2$: GOTO 770
760  CALL 768: GOSUB 1080: GOTO 740
770  FOR I = 1 TO N1
780  PRINT A$(I): NEXT
790  PRINT D$"CLOSE"F2$
800  END
1000  REM SOUS-PGM DE MESSAGES D'ERR
     EUR
1010  HTAB (40 -  LEN (R$)) / 2: PRI
     NT R$: NORMAL : RETURN
1020  VTAB 21: PRINT "APPUYEZ SUR RE
     TURN POUR UN NOUVEL ESSAI": VTA
     B 23: PRINT "  OU APPUYEZ SUR E
     SCAPE POUR QUITTER";: RETURN
1030  VTAB 19:R$ = "POUR VOIR LE CAT
     ALOGUE TAPEZ ?": GOSUB 1010: RE
     TURN
1040  VTAB 23: HTAB 38: GET R$: IF
     ASC (R$) = 13 THEN  HOME :VT =
     10: RETURN
1050  IF  ASC (R$) = 27 THEN  END
1055  ONERR  GOTO 190
1060  IF  ASC (R$) = 63 THEN  HOME :
      PRINT D$;"CATALOG": GET R$:VT
     = 23: RETURN
1070  GOTO 1040
1080 E =  PEEK (222)
1090  HOME : VTAB 5
1100  ON E GOTO 1190,1190,1190,1190,
     1210,1210,1130,1170,1110,1150,1
     250,1150,1150
1105  PRINT  CHR$ (7);"BREAK IN "; P
     EEK (218) +  PEEK (219) * 256:
     END : RESUME
1110 R$ = "DISQUETTE PLEINE": GOSUB
     1010: VTAB 10:R$ = "VEUILLEZ CH
     ANGER DE DISQUETTE": GOSUB 1010
1120  GOSUB 1020: GOSUB 1040: RETURN

1130 R$ = "FICHIER NON TROUVE"
1135  GOSUB 1010: VTAB 10:R$ = "VOUS
      AVEZ TAPE:": GOSUB 1010: VTAB
     12: INVERSE :R$ = F1$: GOSUB 10
     10
1140  GOSUB 1020: GOSUB 1030: GOSUB
     1040: RETURN
1150 R$ = "LE FICHIER SOURCE NE PEUT
      ETRE": GOSUB 1010: VTAB 7:R$ =
      "QU'UN FICHIER DE TYPE TEXTE":
      GOSUB 1010: VTAB 10:R$ = "CREE
      PAR L'ASSEMBLEUR BIG MAC": GOS
     UB 1010
1160  GOSUB 1020: GOSUB 1030: GOSUB
     1040: RETURN
1170 R$ = "VERIFIEZ LA FERMETURE DE
     LA PORTE": GOSUB 1010: VTAB 12:
     R$ = "OU PLACEZ UNE DISQUETTE D
     ANS LE LECTEUR": PRINT R$
1180  GOSUB 1020: GOSUB 1040: RETURN
1190 R$ = "DISQUETTE PROTEGEE": GOSU
     B 1010: VTAB 10:R$ = "VEUILLEZ
     ENLEVER LA PROTECTION ET": GOSU
     B 1010
1200  GOSUB 1020: GOSUB 1040: RETURN

1210 R$ = "LE FICHIER " + F$ + " EST
      ABSENT": GOSUB 1010:R$ = "VEUI
     LLEZ PLACER LA BONNE DISQUETTE
     ET": VTAB 7: GOSUB 1010
1220  GOSUB 1020: GOSUB 1040: RETURN
1250 R$ = "NOM DE FICHIER INCORRECT"
     : GOTO 1135
```

Source 'T.NAVETTE'

Assembleur Big Mac

```
 1 ******************************
 2 **                           **
 3 **       ...........         **
 4 **       :         :         **
 5 **       : NAVETTE :         **
 6 **       :.........:         **
 7 **                           **
 8 **   PAR Jean paul ARBEL     **
 9 **                           **
10 **                           **
11 ******************************
12
13          ORG  $300
14 AMPERV   =    $3F5
15 UTIL     =    $D000
16 RAM1     =    $C08B
17 RAM2     =    $C083
18 ROM      =    $C082
19 ROM1     =    $C089
20 ROM2     =    $C081
21 IRQ      =    $FFFE
22
23 ******************************
24 *** VECTORISE L'AMPERSAND ***
25
26          LDA  #$4C
27          STA  AMPERV
28          LDA  #<DEBUT
29          STA  AMPERV+1
30          LDA  #>DEBUT
31          STA  AMPERV+2
32          RTS
33
34 ******************************
35 ****  VECTORISE LE BREAK ****
36
37 DEBUT    JSR  COMMUT      ;Commute la R
              OM et le Bank 1 ou 2
38          LDA  #<GO        ;Vectorise le
              BRK vers NAVETTE
39          STA  IRQ
40          LDA  #>GO
41          STA  IRQ+1
42          JSR  UTIL        ;Exécute l'ut
              ilitaire
43          BIT  ROM         ;Retour au BA
              SIC
44          RTS
45 BANK     DFB  $80         ;0 = Bank 1 ,
              $80 = Bank 2
46
47
48 ************************************
49 ** NAVETTE  entre la RAM et la ROM  **
50 ************************************
51
52 *          Sauvegarde des  paramètres
53 GO        STA  SAVA        ;A
54          STY  SAVY        ;Y
55          PLA
56          STA  SAVP        ;P
57          LDA  $6
58          STA  SAV6        ;R6
59          LDA  $7
60          STA  SAV7        ;R7
61          PLA
62          STA  $6          ;ADRL BRK+2
63          PLA
64          STA  $7          ;ADRH BRK+2
65
66 * Recherche  l'instruction après le BR
             K
67          LDA  $6
68          BNE  SAUT
69          DEC  $7
70 SAUT     DEC  $6
71          LDY  #$2
72 BOUCLE   LDA  ($6),Y      ;Transmission
              de l'instruction
73          STA  INSTR,Y
74          DEY
75          BPL  BOUCLE
76          CMP  #$4C        ;OPCODE = JMP
              ?
77          BEQ  JUMP
78          CMP  #$6C        ;OPCODE = JMP
              Indirect ?
79          BEQ  JUMP
80          CMP  #$7C        ;OPCODE = JMP
              préindéxé ?
81          BEQ  JUMP
82
83 * Calcul adresse  de retour en Carte L
             angage (sauf si JMP)
84          CLC
85          LDA  $6
86          ADC  #$2         ;ADRL BRK+3
87          TAY
88          LDA  $7
89          ADC  #$0         ;ADRH BRK+3
90          PHA              ;ADRH Retour
              utilitaire - 1
91          TYA
92          PHA              ;ADRL Retour
              utilitaire - 1
93
94 * Restauration des paramètres
95
96 JUMP      BIT  BANK        ;Bank 1 ou 2
97          BPL  J1
98          BIT  ROM2
99          BIT  ROM2
100         JMP  J2
101 J1      BIT  ROM1
102         BIT  ROM1
103 J2      LDA  SAV6        ;R6
104         STA  $6
105         LDA  SAV7        ;R7
106         STA  $7
107         LDY  SAVY        ;Y
108         LDA  SAVP        ;P
109         PHA
110         LDA  SAVA        ;A
111         PLP
112         JSR  INSTR
113
114 ******************************
115 * COMMUTE  LA CARTE LANGAGE *
116
117 COMMUT   PHP
118         BIT  BANK        ;Bank 1 ou 2
119         BPL  C1
120         BIT  RAM2
121         BIT  RAM2
122         JMP  C2
123 C1      BIT  RAM1
124         BIT  RAM1
125 C2      PLP
126         RTS
127
128 ******************************
129 ** EXECUTE  L'INSTRUCTION  **
130
131 INSTR    DS   3           ;Instruction
              aprés le BRK
132         RTS
133
134 ******************************
135 **  SAUVEGARDE PARAMETRES  **
136
137 SAVA      DS   1
138 SAVY      DS   1
139 SAVP      DS   1
140 SAV6      DS   1
141 SAV7      DS   1
```

Routine 'FRE(16)'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE FRE(16),A$927C,L$1F9

```
927C- 4C 86 92 E5
9280- 6E C9 04 90 01 60 20 04
9288- 94 A6 73 A5 74 86 6F 85
```

```
9290- 70 A9 55 A2 00 85 5E 86
9298- 5F C5 52 F0 05 20 1F 93
92A0- F0 F7 A9 07 85 8F A5 69
92A8- A6 6A 85 5E 86 5F E4 6C
92B0- D0 04 C5 6B F0 05 20 12
92B8- 93 F0 F3 85 94 86 95 A9
92C0- 03 85 8F A5 94 A6 95 E4
92C8- 6E D0 07 C5 6D D0 03 4C
92D0- BA 93 85 5E 86 5F A0 00
92D8- B1 5E AA C8 B1 5E 08 C8
92E0- B1 5E 65 94 85 94 C8 B1
92E8- 5E 65 95 85 95 28 10 D3
92F0- 8A 30 D0 C8 B1 5E A0 00
92F8- 0A 69 05 65 5E 85 5E 90
9300- 02 E6 5F A6 5F E4 95 D0
9308- 04 C5 94 F0 BA 20 1F 93
9310- F0 F3 B1 5E 30 05 C8 B1
9318- 5E 30 03 4C A9 93 C8 B1
9320- 5E F0 F8 85 18 C8 B1 5E
9328- 85 19 C5 6F C8 B1 5E 85
9330- 1A E5 70 B0 E6 A5 19 CD
9338- 14 94 A5 1A ED 15 94 90
9340- 68 A5 19 C5 6D A5 1A E5
9348- 6E 90 D0 A2 21 CA CA A5
9350- 19 DD 13 94 A5 1A FD 14
9358- 94 90 F2 86 1B A2 03 BD
9360- 13 94 9D 11 94 BD 14 94
9368- 9D 12 94 BD 34 94 9D 32
9370- 94 BD 33 94 9D 31 94 BD
9378- 54 94 9D 52 94 BD 53 94
9380- 9D 51 94 E4 1B E8 E8 90
9388- D6 A6 1B A5 19 9D 13 94
9390- A5 1A 9D 14 94 A5 18 9D
9398- 34 94 A5 5E 9D 53 94 A5
93A0- 5F 9D 54 94 A5 8F 9D 33
93A8- 94 A5 8F 18 65 5E 85 5E
93B0- 90 02 E6 5F A6 5F A0 00
93B8- 60 60 A2 1F BD 34 94 A8
93C0- F0 F7 38 A5 6F FD 34 94
93C8- 85 6F A5 70 E9 00 85 70
93D0- BD 13 94 85 1C BD 14 94
93D8- 85 1D 88 C0 FF F0 06 B1
93E0- 1C 91 6F 90 F5 BD 33 94
93E8- 29 04 4A A8 C8 BD 53 94
93F0- 85 1C BD 54 94 85 1D A5
93F8- 6F 91 1C A5 70 C8 91 1C
9400- CA CA 10 B8 A2 1F A9 00
9408- 9D 34 94 9D 14 94 CA 10
9410- F7 4C 91 92 FF FF FF FF
9418- FF FF FF FF FF FF FF FF
9420- FF FF FF FF FF FF FF FF
9428- FF FF FF FF FF FF FF FF
9430- FF FF FF FF FF FF FF FF
9438- FF FF FF FF FF FF FF FF
9440- FF FF FF FF FF FF FF FF
9448- FF FF FF FF FF FF FF FF
9450- FF FF FF FF FF FF FF FF
9458- FF FF FF FF FF FF FF FF
9460- FF FF FF FF FF FF FF FF
9468- FF FF FF FF FF FF FF FF
9470- FF FF FF FF EA
```

Routine 'INPUT'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par : BSAVE INPUT,A$300,L$23

```
0300- A9 00 85 D8 68 A8 68 A6
0308- DF 9A 48 98 48 60 20 B1
0310- 00 20 E3 DF 85 85 84 86
0318- 20 2C D5 C8 E8 20 E9 E3
0320- 4C 9A DA
```

E.P.E. 5.0

Sous DOS,
Sous ProDOS,
en 40 colonnes,
en 80 colonnes,

Sur

Apple][+
Apple //e
Apple //e+
Apple //c

Vous le trouverez page 74. Vous avez déjà E.P.E. ? Nous le mettrons à niveau !

Routine 'DIM.VAR.OBJ'

Après saisie de ce code sous moniteur, vous le sauvegarderez par : BSAVE DIM.VAR.OBJ,A$927C,L$18E

```
927C- A4 6B 84 18
9280- A4 6C 84 19 A0 05 84 08
9288- A2 00 86 07 86 1B 86 1C
9290- 86 1D 86 09 F0 19 20 B1
9298- 00 D0 04 E6 1B D0 35 C9
92A0- 24 F0 11 C9 25 F0 33 C9
92A8- C8 F0 21 C9 C9 F0 20 95
92B0- 06 E8 D0 E2 E6 1D A5 07
92B8- 09 80 85 07 A0 01 B1 B8
92C0- 20 BA 00 F0 0F 20 B1 00
92C8- C9 C8 D0 03 4C 50 93 E6
92D0- 1C 4C 50 93 20 E5 92 4C
92D8- 11 93 E6 1D A5 06 09 80
92E0- 85 06 4C B4 92 A0 00 B1
92E8- 18 C5 06 D0 07 C8 B1 18
92F0- C5 07 F0 1C 20 00 93 85
92F8- 19 A5 1A 85 18 4C E5 92
9300- A0 02 B1 18 18 65 18 85
9308- 1A C8 B1 18 65 19 85 1E
9310- 60 A5 18 85 42 A5 19 85
9318- 43 C8 B1 18 85 08 C8 B1
9320- 18 85 09 20 00 93 A5 1A
9328- 85 3C A5 1E 85 3D A5 6D
9330- 85 3E A5 6E 85 3F A0 00
9338- 20 2C FE A5 6D 38 E5 08
9340- 85 6D A5 6E E5 09 85 6E
9348- A6 1B D0 03 20 B1 00 60
9350- 20 E5 92 A0 04 B1 18 C9
9358- 01 D0 F1 20 00 93 A5 1D
9360- F0 09 C6 08 A5 08 38 E5
9368- 1D 85 08 A5 1C F0 38 A0
9370- 06 B1 18 38 E9 01 91 18
9378- 88 B1 18 E9 00 91 18 A0
9380- 02 B1 18 38 E5 08 91 18
9388- C8 B1 18 E9 00 91 18 A5
9390- 1A 48 A5 1E 48 20 00 93
9398- 85 43 A5 1A 85 42 68 85
93A0- 3D 68 85 3C 4C 2E 93 A0
93A8- 06 B1 18 18 69 01 91 18
93B0- 88 B1 18 69 00 91 18 A0
93B8- 02 B1 18 18 65 08 91 18
93C0- C8 B1 18 69 00 91 18 A5
93C8- 6E 85 07 85 09 A5 6D 85
93D0- 06 18 65 08 85 6D 85 08
93D8- 90 04 E6 6E E6 09 A0 00
93E0- B1 06 91 08 A5 07 C5 1E
93E8- D0 09 A5 06 C5 1A D0 03
93F0- 4C 4C 93 C6 06 A5 06 C9
93F8- FF D0 02 C6 07 C6 08 A5
9400- 08 C9 FF D0 DB C6 09 4C
9408- E0 93
```

Récapitulation 'NAVETTE'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE NAVETTE,A$300,L$B6

```
0300- A9 4C 8D F5 03 A9 10 8D
0308- F6 03 A9 03 8D F7 03 60
0310- 20 96 03 A9 25 8D FE FF
0318- A9 03 8D FF FF 20 00 D0
0320- 2C 82 C0 60 80 8D B1 03
0328- 8C B2 03 68 8D B3 03 A5
0330- 06 8D B4 03 A5 07 8D B5
0338- 03 68 85 06 68 85 07 A5
0340- 06 D0 02 C6 07 C6 06 A0
0348- 02 B1 06 99 AD 03 88 10
0350- F8 C9 4C F0 15 C9 6C F0
0358- 11 C9 7C F0 0D 18 A5 06
0360- 69 02 A8 A5 07 69 00 48
0368- 98 48 2C 24 03 10 09 2C
0370- 81 C0 2C 81 C0 4C 7E 03
0378- 2C 89 C0 2C 89 C0 AD B4
0380- 03 85 06 AD B5 03 85 07
0388- AC B2 03 AD B3 03 48 AD
0390- B1 03 28 20 AD 03 08 2C
0398- 24 03 10 09 2C 83 C0 2C
03A0- 83 C0 4C AB 03 2C 8B C0
03A8- 2C 8B C0 28 60 00 00 00
03B0- 60 00 00 00 00 00
```

UN DECRUNCHER

Patrice Neveu

Je doute que personne n'ai jamais eu de problèmes de lecture ou de correction sur une ligne malheureusement trop grosse pour êre claire ou encore pour que le curseur puisse la parcourir d'une extrémité à l'autre. D'autre part, certains possèdent peut-être un CRUNCHER (celui du Pom's 22 par exemple...), programme qui installe sur une même ligne plusieurs autres consécutives en les séparant avec les 'deux-points' (:) habituels.

Il est toutefois utile de pouvoir transformer son programme Applesoft afin qu'il n'y ait qu'une instruction par ligne car il est alors facile de lire un listing, de le comprendre et de le modifier. C'est ce que le DECRUNCHER est à même de faire.

Comment faire ?

Pour l'utiliser, il suffit de charger par un LOAD le programme Basic comme vous le faites d'habitude, puis de taper :

BRUN DECRUNCHER ou
CALL 35072 s'il est déjà en mémoire.

Les changements que le DECRUNCHER effectue sont :

- une instruction par ligne sauf après IF étant donné que ce qui le suit est conditionné par le test ;
- un renumérotage de 1 en 1 avec modification des GOTO, GOSUB, DEL, des THEN, LIST, RUN lorsqu'ils sont suivis de numéros de ligne, et enfin, des ON XX GOTO/ GOSUB.

Il existe toutefois quelques limitations :

- le texte original écrit en Basic ne peut prendre plus de 14336 octets soit 14 Ko, auquel cas il affichera : TEXTE TROP LONG ;
- il ne doit pas avoir plus de 640 lignes à l'origine. (Le nombre de lignes qui résulte du décrunchage n'est limité que par l'Applesoft lui-même) ;
- le programme décrunché ne doit pas pouvoir faire plus de 18688 octets, soit 18,25 Ko. (Le programme décrunché étant généralement plus important que l'original) ;
- au cas où le programme découvre un GOTO, GOSUB, DEL, LIST, se référant à une ligne inexistante, il enverra : PAS DE LIGNE X.

Le principe

Il est tout d'abord nécessaire, voire indispensable d'étudier rapidement la structure du texte d'un programme Applesoft en mémoire. L'adresse de début du texte se trouve en TXTTAB ($67.68) et la fin en PRGEND ($AF.B0). Ce sont donc les octets situés entre ces deux adresses qu'il s'agira de traiter. Ce traitement se déroulera globalement ainsi :

- lire les 2 premiers octets qui représentent l'adresse de la prochaine ligne ;
- lire les 2 suivants qui ne sont autres que le numéro de la ligne ;
- puis étudier la suite de la ligne afin de trouver ':' ;
- si on les trouve, marquer la fin de la ligne actuelle par un octet nul puis commencer une nouvelle ligne sur laquelle on mettra la suite ;
- sinon, passer à la prochaine ligne.

En cours de route, à chaque fois qu'on lira une nouvelle ligne de l'original, on mémorisera son numéro ainsi que celui de celle qui suivra. (N'oublions pas qu'il y a renumérotation !) Avec un tel processus, on obtient le programme décrunché, mais sans que les GOTO, GOSUB etc. soient modifiés. Une deuxième passe s'impose alors ; il faut :

- lire la ligne comme précédemment mais en cherchant tous les mots-clé susceptibles d'être suivis par un ou plusieurs numéros de ligne ;
- identifier ce ou ces numéros et les rechercher dans la table construite précédemment ;
- si on a trouvé, échanger l'ancien numéro par le nouveau qui est issu de la renumérotation ;
- sinon, il y a une erreur et on s'arrête ;
- puis, si tout va bien, on passe à la prochaine instruction.

Il est alors utilisable, mais mieux vaut le sauver avant car, s'il empiète sur la page graphique 1 et qu'il l'utilise, gare aux problèmes !

Enfin, à titre d'exemple, voici un listing et sa représentation en mémoire :

```
10 TEXT : HOME
20 GOTO 10

0800- 00 09 08 0A 00 89 3A 97
0808- 00 11 08 14 00 AB 31 30
0810- 00 00 00
```

$801.802 : adresse de la prochaine ligne

$803.804 : numéro de ligne
$000A = 10
$805 : token de TEXT
$806 : :
$807 : HOME
$808 : fin de ligne
$809.80A : adresse de la prochaine ligne
$80B.80C : numéro de ligne
$0014 = 20
$80D : token de GOTO
$80E.80F : codes ASCII de 1 ($31) et de 0 ($30)
$810 : fin de ligne
$811.812 : fin de programme

Le même programme mais décrunché s'écrira ainsi :

```
0 TEXT
1 HOME
2 GOTO 0
```

```
800- 00 07 08 00 00 89 00 0D
808- 08 01 00 97 00 14 08 02
810- 00 AB 30 00 00 00
```

$801.802 : adresse de la prochaine ligne
$803.804 : numéro de la ligne
$805 : token de TEXT
$806 : fin de ligne
$807.808 : adresse de la prochaine ligne
$809.80A : numéro de la ligne
$80B : token de HOME
$80C : fin de ligne
$80D.80E : adresse de la prochaine ligne
$80F.810 : numéro de la ligne
$811 : token de GOTO
$812 : code ASCII de 0 ($30)
$813 : fin de ligne
$814.815 : fin de programme

Les adresses des lignes suivantes et les numéros de lignes sont inscrits avec l'octet bas suivi de l'octet haut.

Programme 'DECRUNCHER DEMO'

```
10  TEXT : HOME : INVERSE : PRINT "      D
    EMONSTRATION DE DECRUNCHAGE       ": NO
    RMAL : POKE 34,1
20  PRINT "VOICI LE LISTING DE CE PROGRAMM
    E:": PRINT : LIST
30  VTAB 22: FLASH : PRINT " FRAPPEZ UNE T
    OUCHE POUR LE DECRUNCHER   PUIS FAITES
    LIST POUR VOIR LE RESULTAT";: GET R$: P
    RINT : NORMAL
40  POKE 34,0: PRINT  CHR$ (4)"BRUN DECRUN
    CHER"
```

Le même programme, mais 'décrunché'

```
0  TEXT
1  HOME
2  INVERSE
3  PRINT "      DEMONSTRATION DE DECRUNCHA
   GE       "
4  NORMAL
5  POKE 34,1
6  PRINT "VOICI LE LISTING DE CE PROGRAMME
   :"
7  PRINT
8  LIST
9  VTAB 22
10  FLASH
11  PRINT " FRAPPEZ UNE TOUCHE POUR LE DEC
   RUNCHER   PUIS FAITES LIST POUR VOIR LE
    RESULTAT";
12  GET R$
13  PRINT
14  NORMAL
15  POKE 34,0
16  PRINT  CHR$ (4)"BRUN DECRUNCHER"
```

Programme 'DECRUNCHER.S'

Assembleur Big Mac

```
 1
 2          LST ON
 3
 4          ORG $8900
 5          OBJ $8000
 6
 7
 8 *********************************
 9 * DECRUNCHER 1.3                 *
10 *                               *
11 *************PAR*PATRICE*NEVEU***
12
13 *################################
14 *                               #
15 * - UNE INSTRUCTION PAR LIGNE   #
16 *   SAUF APRES 'IF'             #
17 * - RENUMEROTATION DE 1 EN 1    #
18 *   AVEC LES GOTO/GOSUB/RUN     #
19 *   LIST/ON/DEL REMIS A JOUR    #
20 *                               #
21 * - SOURCE MAXIMUM: 14K         #
22 *                   640 LIGNES  #
23 * - RESULTAT MAX. : 18.25K      #
24 *                   64000 LIGNES#
25 *                               #
26 *********************************
27
28
29
30
31
32 BASIC     EQU $03D0       RETOUR A L'APPLESOFT
33 STXTPT    EQU $D697
                             MET TXTPTR A LA VALEUR DE
                              (TXTTAB)-1
34 STROUT    EQU $DB3A       SOUS-PROGRAMME D'AFFICHAGE
                              D'UNE CHAINE
35 GETADR    EQU $E752       FAC --> 2 OCTETS EN LINNUM
36 AXFAC     EQU $EBA0       A&X --> FAC
37 FIN       EQU $EC4A       NOMBRE FLOTTANT --> FAC PA
                             R CHRGET
38 LINPTR    EQU $ED24       CONVERTIR ET AFFICHER X,A
                             EN DECIMAL
39 FOUT      EQU $ED34       FAC --> CHAINE DE CHIFFRE
40 CROUT     EQU $FC62       RETOUR CHARIOT
41 BELL      EQU $FF3A       BIP SONORE
42
43 LIMITE    EQU $4000       LES DEUX ZONES: $0800.3FFF
```

```
                                /$4000.4900
44 ESPACE    EQU $3800          ETENDUE DE LA 1ERE ZONE
45
46
47 LINADR    EQU $06            ADRESSE LIGNE COURANTE
48 TXTPTRS   EQU $08            SAUVEGARDE DE TXTPTR
49 NUMPTR    EQU $18            POINTEUR
50 LINLEN    EQU $1D            LONGUEUR NOUVELLE LIGNE
51 ONFLAG    EQU $1E            =1: ON XX, >=2: ON XX GOTO
                                , =0 SINON
52 LISTFLAG  EQU $1F            = 1 SI LIST (XXXX), 0 SINO
                                N
53 LINNUM    EQU $50            REGISTRE CONTENANT LE NUME
                                RO DE LIGNE
54 TXTTAB    EQU $67            DEBUT DU TEXTE BASIC
55 LOMEM     EQU $69            DEBUT DE LA ZONE DES VARIA
                                BLES SIMPLES
56 ARYTAB    EQU $6B            DEBUT  ZONE DES TABLEAUX
57 STREND    EQU $6D            FIN DES TABLEAUX, DEBUT DE
                                 ZONE LIBRE
58 HOHO      EQU $9E            2EME OCTET DE FAC
59 MOH       EQU $9F            3EME OCTET DE FAC
60 PGREND    EQU $AF            ADRESSE DE LA FIN DU TEXTE
                                 DU PROGRAMME
61 CHRGET    EQU $B1            PREND LE PROCHAIN CARACTER
                                E DU TEXTE
62 CHRGOT    EQU $B7            PREND LE CARACTERE POINTE
                                ACTUELLEMENT
63 TXTPTR    EQU $B8            POINTEUR DE CARACTERE DANS
                                 LE TEXTE
64 DELFLAG   EQU $CE            = 1 SI DEL XXXXX (,XXXXX),
                                 =0 SINON
65 TEMP1     EQU $EB            4 OCTETS ($EB.EE), PARTIE
                                DE CHRGET
66 CHAINPTR  EQU $EE            POINTEUR DS LA CHAINE DU N
                                O DE LIGNE
67 REMFLAG   EQU $FC            =1 SI ON EST DANS UN REM,
                                0 SINON
68 IFFLAG    EQU $FD            =1 SI ON EST DANS UN IF, 0
                                 SINON
69 ACC       EQU $FE            SAUVEGARDE DE L'ACCUMULATE
                                UR
70 GUILLFLAG EQU $FF            =1 SI ON EST DANS UNE CHAI
                                NE, 0 SINON
71
72 GUILL     EQU $22            TOKENS APPLESOFT
73 DEL       EQU $85
74 GOTO      EQU $AB
75 RUN       EQU $AC
76 IF        EQU $AD
77 GOSUB     EQU $B0
78 REM       EQU $B2
79 ON        EQU $B4
80 LIST      EQU $BC
81 THEN      EQU $C4
82
83
84 *--------------------------------
85 * INITIALISE
86 *--------------------------------
87
88            LDY #$EA          $EA = NOP
89            LDX #3            ON VA DONC MODIFIER CHRGET
                                 (ET CHRGOT)
90 INIT       LDA $BE,X         EN Y METTANT 4 NOP AFIN D'
                                EVITER DE
91            STA TEMP1,X       SAUTER LES ESPACES DANS UN
                                 REM OU UNE
92            STY $BE,X         CHAINE DE CARACTERES.
93            DEX
94            BPL INIT
95
96            INX               REMETTRE X=0 POUR L'INITIA
                                LISATION
97            STX GUILLFLAG     DU DRAPEAU A GUILLEMETS
98            STX IFFLAG        IF
99            STX REMFLAG       REMARQUES
100           STX ONFLAG        ON XXXXX GOTO/GOSUB
101           STX LISTFLAG      LIST (XXXX,XXXX)
102           STX DELFLAG       DELETE XXXX (,XXXX)
103           STX LINLEN        DE LA LONGUEUR DE LA LIGNE
104           STX LINNUM        ET DE SON NUMERO
105           STX LINNUM+1
106
107           LDA #>TABLE       PREPARE LA TABLE OU SERONT
                                 RANGES LES
108           STA TABPTR+2      ANCIENS NUMEROS DE LIGNE S
                                UIVIS DES
109           LDA #<TABLE       NOUVEAUX...
110           STA TABPTR+1
111
112           LDX #$0           MAIS IL FAUT LA REMPLIR DE
                                 $FF AVANT:
113           LDA #<TABLE
114           STA CLR+1
115           LDA #>TABLE
116           STA CLR+2
117           LDA #$FF
118           LDY #$00
119 CLR       STA TABLE,Y
120           INY
121           BNE CLR
122           INC CLR+2
123           INX
124           CPX #10
125           BNE CLR
126
127 *--------------------------------
128 * SAUVE LE TXTPTR
129 *--------------------------------
130
131           LDA TXTPTR
132           STA TXTPTRS
133           LDA TXTPTR+1
134           STA TXTPTRS+1
135
136 *--------------------------------
137 * VERIFIE SI LE PROGRAMME PEUT
138 * ETRE DECRUNCHE
139 *--------------------------------
140
141           LDA PGREND+1      C'EST LE HAUT DE LA FIN DU
                                 PROGRAMME
142           CMP #>LIMITE      DEPASSE LA LIMITE FIXEE ?
143           BCC BEGIN         NON, ON PEUT Y ALLER...
144
145 *--------------------------------
146 * ERREUR, IMPOSSIBLE A DECRUNCHER
147 *--------------------------------
148
149           LDY #>LENMSG
150           LDA #<LENMSG
151           JSR STROUT        AFFICHER LE CHARMANT MESSA
                                GE QUI SUIT:
152           JSR BELL          SUIVI D'UN BIP DE MAUVAIS
                                AUGURE
153           JMP WAOU          REMETTRE TOUT EN ORDRE AVA
                                NT DE SORTIR
154
155 LENMSG    HEX 8D            REVENIR A LA LIGNE
156           ASC "TEXTE TROP LONG"
157           HEX 8D00
158
159 *--------------------------------
160 * COMMENCE A DECRUNCHER
161 *--------------------------------
162
163 BEGIN     JSR STXTPT        FIXE TXTPTR AU DEBUT DU PR
                                OGRAMME -1
164           LDY TXTTAB        ADDITIONNE LA LONGUEUR DE
                                L'ESPACE
165           STY TXTPTR2+1     DISPONIBLE A L'ADRESSE DE
                                DEBUT DU
166           STY LINADR        TEXTE -> TXTPTR2 POUR CHRP
                                UT
167           LDA TXTTAB+1      -> LINADR
168           CLC
169           ADC #>ESPACE
170           STA TXTPTR2+2
171           STA LINADR+1
```

```
172
173             JSR CHRGET      PREND & STOKE L'ADRESSE DE
                                 LA PROCHAINE
174             JSR CHRPUT      LIGNE
175             JSR CHRGET
176             CLC
177             ADC #>ESPACE
178             JSR CHRPUT
179
180 LIGNE      JSR CHRGET      PREND LE NUMERO DE LA LIGN
                                E
181             STA NUMPTR      A METTRE DANS NUMPTR POUR
                                QUE LINTAB ET
182             JSR CHRGET      LINPUT S'EN SERVENT
183             STA NUMPTR+1
184             JSR LINTAB
185 LIN        JSR LINPUT
186
187 LINCHR     JSR CHRGET      PRENDRE UN CARACTERE
188             STA ACC
189             BEQ EOI         EOI=FIN D'INSTRUCTION:':',
                                 FIN DE LIGNE
190 LINCHR5    LDA ACC
191             JSR CHRPUT      METTRE LE CARACTERE POUR L
                                E RESULTAT
192             CMP #REM
193             BNE LINCHR6     MAIS CE N'EST PAS UN REM..
                                .
194             LDA #1          SINON IL FAUT SE LE SIGNAL
                                ER
195             STA REMFLAG
196 LINCHR6    CMP #IF
197             BNE LINCHR4     SI CE N'EST PAS UN IF...
198             LDA #01         AUTREMENT, LE SIGNALER
199             STA IFFLAG
200 LINCHR4    CMP #GUILL
201             BNE LINCHR1     SI CE N'EST RIEN DE TOUT C
                                ELA
202             LDA GUILLFLAG   SI PAS DE GUILLEMET (0) AL
                                ORS DEVIENT 1
203             EOR #$01        SI ON EST DEJA DANS UNE CH
                                AINE (1) CA
204             STA GUILLFLAG   DOIT DONC DEVENIR 0
205 LINCHR1    JMP LINCHR      PASSE AU PROCHAIN CARACTER
                                E
206 EOI         LDA ACC         CETTE FIN D'INSTRUCTION C'
                                EST QUOI ?
207             BNE DECRUNCH    C'EST UN : ALORS CREER UNE
                                 AUTRE LIGNE
208             JSR EOL         SINON C'EST UNE FIN DE LIG
                                NE (00)
209             JMP BOL
210
211 DECRUNCH   LDA GUILLFLAG   SI CE : ETAIT DANS UNE CHA
                                INE, UN REM
212             BNE LINCHR5     OU UN TEST (IF) ALORS ON N
                                E DOIT PAS
213             LDA REMFLAG     DECRUNCHER SOUS PEINE DE C
                                ATASTROPHE !
214             BNE LINCHR5
215             LDA IFFLAG
216             BNE LINCHR5
217             LDA #0
218             JSR EOL         CALCULER ADRESSE DE LA PRO
                                CHAINE LIGNE
219             LDA LINADR      QU'IL FAUT STOKER EN DEBUT
                                 DE LA LIGNE
220             JSR CHRPUT      ACTUELLE ET AUSSI A LA FIN
                                 POUR LA
221             LDA LINADR+1    PROCHAINE...
222             JSR CHRPUT
223
224             INC LINNUM      INCREMENTE LE NUMERO DE LI
                                GNE
225             BNE DC1         POUR FAIRE LA PROCHAINE...
226             INC LINNUM+1
227 DC1         JMP LIN         VA LE STOCKER PUIS LIT LA
                                LIGNE
228
229 BOL         JSR CHRGET      IL FAUT PRENDRE LES 2 PROC
                                HAINS OCTETS
230             JSR CHRGET      POUR SAVOIR SI ON A FINI
231             BEQ FINI        CE QUI EST LE CAS SI A=00
232             LDA LINADR      SINON ON REMPLACE CES 2 OC
                                TETS
233             JSR CHRPUT
234             LDA LINADR+1
235             JSR CHRPUT
236             INC LINNUM      ( EN FAIT ICI, ON EST EN D
                                ECRUNCHAGE )
237             BNE BOL1        ( MAIS ON FAIT PAREIL QUE
                                PRECEDEMENT)
238             INC LINNUM+1
239 BOL1        JMP LIGNE
240
241 FINI        LDA #$0         METTRE 3 OCTETS NULS POUR
                                MARQUER LA
242             JSR CHRPUT      FIN DU TEXTE DECRUNCHE
243             JSR CHRPUT
244             JSR CHRPUT
245             JMP AJUSTE
246
247 *-------------------------------
248 * C'EST DECRUNCHE, RENUMEROTE
249 *-------------------------------
250
251 WAOU        LDA #$00        MAINTENANT QUE LES GOTO ET
                                 GOSUB SONT
252             JSR CHRPUT      MIS A JOUR, ON MARQUE LA F
                                IN DU PROGR.
253             JSR CHRPUT      QUI EST DORENAVANT TERMINE
254             JSR CHRPUT
255             LDA TXTPTRS     REMET LE TXTPTR TEL QU'ON
                                L'A TROUVE
256             STA TXTPTR      AVANT DE DECRUNCHER
257             LDA TXTPTRS+1
258             STA TXTPTR+1
259             LDX #3          ET RETIRER LES NOP POUR RE
                                METTRE CHRGET
```

```
260 FINI1     LDA TEMP1,X     A SON ETAT D'ORIGINE
261           STA $BE,X
262           DEX
263           BPL FINI1
264           LDA TXTPTR2+1   ET FIXE DIFFERENTS POINTEU
                              RS
265           STA PGREND
266           STA LOMEM
267           STA ARYTAB
268           STA STREND
269           LDA TXTPTR2+2
270           STA PGREND+1
271           STA LOMEM+1
272           STA ARYTAB+1
273           STA STREND+1
274           PLA
275           PLA
276           JMP BASIC       HOP, C'EST FINI
277
278 *-------------------------------
279 * AJUSTE LES GOTO / GOSUB
280 *-------------------------------
281
282 AJUSTE    LDX #0          INITIALISE:
283           STX LINLEN      LA LONGUEUR DE LA LIGNE
284           STX TXTPTR      LE POINTEUR SOURCE
285           INX
286           STX TXTPTR2+1   LE POINTEUR RESULTAT
287           STX LINADR      L'ADRESSE DE LA LIGNE EN C
                              OURS
288           LDA TXTTAB+1
289           STA TXTPTR2+2
290           STA LINADR+1
291           LDA #>LIMITE
292           STA TXTPTR+1
293
294 COPYLIN   JSR EOL0        CALCULE ADR. LIGNE QUE L'O
                              N COMMENCE
295           JSR CHRGET      PRENDRE L'ADRESSE DE LA LI
                              GNE SUIVANTE
296           JSR CHRPUT      ET LA STOCKER COMME RESULT
                              AT
297           JSR CHRGET
298           BEQ WAOU        SI LA PARTIE HAUTE EST NUL
                              LE C'EST FINI
299           SEC             SINON CE RESULTAT COMMENCE
                               EN $0800
300           SBC #>ESPACE
301           JSR CHRPUT
302           JSR CHRGET      C'EST AU TOUR DU NUMERO DE
                               LIGNE
303           JSR CHRPUT      CONTENU EN 2 OCTETS
304           JSR CHRGET
305           JSR CHRPUT
306
307 CHRCHGO   LDX ONFLAG
308           CPX #2
309           BCS GOFOUND
310           LDX LISTFLAG
311           BNE TESTNUM
312           LDX DELFLAG
313           BNE GOFOUND1
314 CHRCHGO0  JSR CHRGET      ON RECOPIE LA LIGNE EN REC
                              HERCHANT UN
315 CHRCHGO1  JSR CHRPUT      SAUT (GOTO / GOSUB OU THEN
                               XXXXX)
316           BEQ COPYLIN     00 SIGNIFIE UNE FIN DE LIG
                              NE
317           CMP #ON
318           BNE CHRCHGO2
319           INC ONFLAG
320           BNE CHRCHGO0
321 CHRCHGO2  CMP #GOTO
322           BEQ GOFOUND     GOTO: TRAITEMENT SPECIAL D
                              E LA SUITE
323           CMP #GOSUB
324           BEQ GOFOUND     GOSUB EGALEMENT
325           CMP #RUN        LE RUN (XXXXX) EST TRAITE
                              COMME UN THEN
326           BEQ TESTNUM
327           CMP #LIST
328           BNE CHRCHGO3
329           INC LISTFLAG
330           BNE CHRCHGO
331 CHRCHGO3  CMP #DEL
332           BNE CHRCHGO4
333           INC DELFLAG
334           BNE CHRCHGO
335 CHRCHGO4  CMP #THEN       QUAND AU THEN, PEUT-ETRE..
                              .
336           BNE CHRCHGO     CE N'EST RIEN DE TOUT CELA
                              : CONTINUER
337
338 *-------------------------------
339 * POUR LE THEN IL FAUT UN NUMERO
340 *-------------------------------
341
342 TESTNUM   JSR CHRGET      ON PREND LE CARACTERE SUIV
                              ANT LE THEN
343           CMP #$30        ASCII INFERIEUR AU CODE DE
                               0
344           BCC CHRCHGO1    AU QUEL CAS ON CONTINUE DE
                               COPIER...
345           CMP #$3A        SINON PEUT-ETRE EST-CE SUP
                              ERIEUR A 9 ?
346           BCS CHRCHGO1    ET ALORS ON COPIE BETEMENT
                               LA SUITE
347           BCC GETEND      SINON ON TRITURE LE NUMERO
                               DE LIGNE !!
348
349 *-------------------------------
350 * POSITIONNE SUR LE NUMERO
351 *-------------------------------
352
353 GOFOUND   INC ONFLAG
354 GOFOUND1  JSR CHRGET      POUR LES GOTO / GOSUB (ET
                              ASSIMILES)
355
356 *-------------------------------
357 * TRANSFORME LA CHAINE EN 2 OCT.
358 *-------------------------------
359
360 GETEND    EQU *
361           JSR FIN         GRACE A CHRGET ON CHARGE D
                              ANS FAC LE NO
362           JSR GETADR      PUIS ON LE TRANSFORME EN 2
                               OCTETS
363
364 *-------------------------------
365 * FAIT L'ECHANGE
366 *-------------------------------
367
368           LDA #>TABLE     ON SE PLACE AU DEBUT DE LA
                               TABLE OU
369           STA NUMPTR+1    SONT RANGES LES ANCIENS NU
                              MEROS DE
370           LDA #<TABLE     LIGNES ET LEURS REMPLACANT
                              S
371           STA NUMPTR
372
373 IDENTIFY  LDY #0
374           LDA (NUMPTR),Y  LE BAS DU VIEUX NUMERO DE
                              LA TABLE EST
375           CMP LINNUM      COMPARE A CELUI QU'ON RECH
                              ERCHE
376           BNE NEXTID0     SI CE N'EST PAS LE MEME, A
                              U SUIVANT !
377           INY
378           LDA (NUMPTR),Y  PUIS ON PASSE AU HAUT DU N
                              UMERO
379           CMP #250        QUI DOIT ETRE INF. 249*256
                              +255 = 63999
380           BCS GOERR       SI C'EST PLUS ALORS ERREUR
                               !!!
381           CMP LINNUM+1    SINON ON COMPARE AU HAUT R
                              ECHERCHE
382           BEQ CHANGE      ET ON LE CHANGE SI C'EST T
                              ROUVE
383
384 NEXTID0   EQU *           ON PASSE AU NUMERO SUIVANT
```

```
                          DE LA TABLE
385           CMP #$FF    DES $FF SIGNALENT LA FIN D
                          E LA TABLE
386           BEQ GOERR   ALORS LE NUMERO N'EST PAS
                          TROUVE, ERR !
387 NEXTID    LDA NUMPTR  AUTREMENT, ON VA 4 OCTETS
                          PLUS LOIN
388           CLC         POUR TROUVER UN NOUVEAU NU
                          MERO
389           ADC #4      ( 2 POUR L'ANCIEN + 2 POUR
                           LE NOUVEAU )
390           STA NUMPTR
391           BCC NEXTEND PAS DE RETENUE POUR L'ADDI
                          TION
392           INC NUMPTR+1 SINON, EN TENIR COMPTE
393 NEXTEND   JMP IDENTIFY ET CA REPART POUR UN TOUR.
                          ..
394
395 *--------------------------------
396 * ERREUR, UN NUMERO DE LIGNE QUI
397 * EST APPELE N'EXISTE PAS
398 *--------------------------------
399
400 GOERR     LDY #>GOMSG AFFICHER UN IGNOBLE MESSAG
                          E D'ERREUR
401           LDA #<GOMSG
402           JSR STROUT
403           LDA LINNUM+1 ON PREND LE NUMERO DE LIGN
                          E RECHERCHE
404           LDX LINNUM
405           JSR LINPTR  POUR L'AFFICHER
406           JSR CROUT   PUIS ON PASSE A LE LIGNE
407           JSR BELL    AVEC UN BIP AGACANT
408           JMP WAOU
409
410 GOMSG     HEX 8D
411           ASC "PAS DE LIGNE "
412           HEX 00
413
414 *--------------------------------
415 * ICI S'EFFECTUE LE CHANGEMENT DE
416 * NUMERO DE LIGNE
417 *--------------------------------
418
419 CHANGE    INY         ON POINTE DANS LA TABLE SU
                          R LE NUMERO
420           LDA (NUMPTR),Y TROUVE APRES GOTO/GOSUB DO
                          NC ON
421           STA MOH     POURSUIT EN PRENANT LE NOU
                          VEAU QUI EST
422           INY         ISSU DE LA RENUMEROTATION
                          ET ON LE MET
423           LDA (NUMPTR),Y DANS FAC.
424           STA HOHO
425
426 *--------------------------------
427 * TRANSFORME LE NUMERO EN CHAINE
428 *--------------------------------
429
430           LDX #$90    CECI EST PRESQUE UN LINPTR
                           ($ED24)
431           SEC         CAR CA COMMENCE PAREIL,
432           JSR AXFAC
433           JSR FOUT
434           STY CHAINPTR+1 MAIS CA N'AFFICHE PAS FAC
                          !
435           STA CHAINPTR
436           LDY #0
437 WRITENUM  LDA (CHAINPTR),YPRENDRE LA CHAINE REPRESEN
                          TANT LE
438           BEQ WRITEND NOUVEAU NUMERO, ET L'INCLU
                          RE DANS LE
439           JSR CHRPUT  PROGRAMME DECRUNCHE APRES
                          LE GOTO
440           INY         LE GOSUB OU ENCORE LE THEN
441           BNE WRITENUM
442
443 WRITEND   JSR CHRGOT  PUIS IL FAUT LIRE LE DERNI
                          ER CARACTERE
444           JSR CHRPUT  LU APRES LE NOMBRE ET LE C
                          OPIER
445           BEQ WRITEND1 SI C'EST UN 0 -> FIN DE LI
                          GNE
446           JMP CHRCHGO SINON, SUITE DE LIGNE...
447 WRITEND1  EQU *
448           JMP COPYLIN
449
450 *--------------------------------
451 * SOUS-PROGRAMMES
452 *--------------------------------
453
454 EOL       EQU *       END OF LINE CALCULE L'ADRE
                          SSE DE LA
455           JSR CHRPUT  LIGNE SUIVANTE EN ADDITION
                          NANT LA
456 EOL0      LDA LINADR  LONGUEUR DE LA LIGNE ACTUE
                          LLE ET SON
457           LDX LINADR+1 ADRESSE DE DEBUT
458           LDY #0
459           CLC
460           ADC LINLEN
461           STA (LINADR),Y LE RESULTAT EST REMIS DANS
                           LE PROGRAMME
462           BCC EOL1    POUR CONSERVER SA COHERENC
                          E
463           INX
464 EOL1      TXA
465           INY
466           STA (LINADR),Y
467
468           LDA (LINADR),Y CECI SERVIRA A LA LIGNE SU
                          IVANTE
469           TAX
470           DEY
471           LDA (LINADR),Y
472           STA LINADR
473           STX LINADR+1
474
475           LDY #0      ON VA RECOMMENCER UNE LIGN
                          E DONC:
476           STY LINLEN  LONGUEUR DE LA LIGNE = 0
477           STY ONFLAG  ON XXXX GOTO/GOSUB EST TER
                          MINE
478           STY LISTFLAG LIST XXXXX,XXXXX EGALEMENT
                          , AINSI QUE
479           STY DELFLAG DEL XXXXX,XXXXX
480           STY GUILLFLAG LA CHAINE DE CARACTERE ENT
                          RE GUILLEMETS
481           STY IFFLAG  LE TEST AVEC IF
482           STY REMFLAG LE REM OU IL NE FALLAIT RI
                          EN CHANGER
483
484           RTS
485
486 *================================
487
488 CHRPUT    EQU *       CETTE ROUTINE EFFECTUE LA
                          TACHE OPPOSEE
489           CLC         A CELLE DE CHRGET, A SAVOI
                          R REMETTRE UN
490 TXTPTR2   STA $4000   OCTET PRIS PAR CHRGET, DAN
                          S LE RESULTAT
491           INC LINLEN  EN METTANT A JOUR LA LONGU
                          EUR DE LA
492           INC TXTPTR2+1 LIGNE.
493           BNE CHRPUTEND
494           INC TXTPTR2+2
495 CHRPUTEND TAX         CECI POUR CONSERVER LE REG
                          ISTRE D'ETAT
496           RTS
497
498 *================================
499
500 LINPUT    LDA LINNUM  STOCKE LE NUMERO DE LA LIG
                          NE ACTUELLE
501           JSR CHRPUT  EN 2 OCTETS MIS EN LINNUM
                          ($50.51)
502           LDA LINNUM+1
503           JSR CHRPUT
504           RTS
```

```
505
506 *================================
507
508 LINTAB    LDA NUMPTR      PRENDRE LE NUMERO DE LIGNE
                               DU PROGRAMME
509           JSR LINTOTAB    ORIGINAL POUR LE METTRE DA
                              NS LA TABLE
510           LDA NUMPTR+1
511           JSR LINTOTAB
512
513           LDA LINNUM      ET LE FAIRE SUIVRE DU NOUV
                              EAU NUMERO
514           JSR LINTOTAB    QU'IMPLIQUE LA RENUMEROTAT
                              ION LORS DU
515           LDA LINNUM+1    DECRUNCHAGE
516           JSR LINTOTAB
517           RTS
518
519 LINTOTAB   CLC
520 TABPTR     STA TABLE
521            INC TABPTR+1     IL FAUT BIEN-SUR FAIRE EVO
                                LUER LE
522            BNE LNTTABEND    POINTEUR (TABPTR)
523            INC TABPTR+2
524 LNTTABEND RTS
525
526 *================================
527 * TABLE DE CONVERSION DES LIGNES
528 * COMMENCE ICI
529 *================================
530
531 TABLE      EQU *
```

Récapitulation 'DECRUNCHER'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE DECRUNCHER,A$8900,L$2DD

```
8900- A0 EA A2 03 B5 BE 95 EB
8908- 94 BE CA 10 F7 E8 86 FF
8910- 86 FD 86 FC 86 1E 86 1F
8918- 86 CE 86 1D 86 50 86 51
8920- A9 8B 8D D3 8B A9 DD 8D
8928- D2 8B A2 00 A9 DD 8D 3B
8930- 89 A9 8B 8D 3C 89 A9 FF
8938- A0 00 99 DD 8B C8 D0 FA
8940- EE 3C 89 E8 E0 0A D0 F2
8948- A5 B8 85 08 A5 B9 85 09
8950- A5 B0 C9 40 90 1F A0 89
8958- A9 63 20 3A DB 20 3A FF
8960- 4C 28 8A 8D D4 C5 D8 D4
8968- C5 A0 D4 D2 CF D0 A0 CC
8970- CF CE C7 8D 00 20 97 D6
8978- A4 67 8C A2 8B 84 06 A5
8980- 68 18 69 38 8D A3 8B 85
8988- 07 20 B1 00 20 A0 8B 20
8990- B1 00 18 69 38 20 A0 8B
8998- 20 B1 00 85 18 20 B1 00
89A0- 85 19 20 BB 8B 20 B0 8B
89A8- 20 B1 00 85 FE F0 22 A5
89B0- FE 20 A0 8B C9 B2 D0 04
89B8- A9 01 85 FC C9 AD D0 04
89C0- A9 01 85 FD C9 22 D0 06
89C8- A5 FF 49 01 85 FF 4C A8
89D0- 89 A5 FE D0 06 20 70 8B
89D8- 4C FF 89 A5 FF D0 D0 A5
89E0- FC D0 CC A5 FD D0 C8 A9
89E8- 00 20 70 8B A5 06 20 A0
89F0- 8B A5 07 20 A0 8B E6 50
89F8- D0 02 E6 51 4C A5 89 20
8A00- B1 00 20 B1 00 F0 13 A5
8A08- 06 20 A0 8B A5 07 20 A0
8A10- 8B E6 50 D0 02 E6 51 4C
8A18- 98 89 A9 00 20 A0 8B 20
8A20- A0 8B 20 A0 8B 4C 5F 8A
8A28- A9 00 20 A0 8B 20 A0 8B
8A30- 20 A0 8B A5 08 85 B8 A5
8A38- 09 85 B9 A2 03 B5 EB 95
8A40- BE CA 10 F9 AD A2 8B 85
8A48- AF 85 69 85 6B 85 6D AD
8A50- A3 8B 85 B0 85 6A 85 6C
8A58- 85 6E 68 68 4C D0 03 A2
8A60- 00 86 1D 86 B8 E8 8E A2
8A68- 8B 86 06 A5 68 8D A3 8B
8A70- 85 07 A9 40 85 B9 20 73
8A78- 8B 20 B1 00 20 A0 8B 20
8A80- B1 00 F0 A4 38 E9 38 20
8A88- A0 8B 20 B1 00 20 A0 8B
8A90- 20 B1 00 20 A0 8B A6 1E
8A98- E0 02 B0 45 A6 1F D0 34
8AA0- A6 CE D0 3F 20 B1 00 20
8AA8- A0 8B F0 CA C9 B4 D0 04
8AB0- E6 1E D0 F0 C9 AB F0 29
8AB8- C9 B0 F0 25 C9 AC F0 14
8AC0- C9 BC D0 04 E6 1F D0 CE
8AC8- C9 85 D0 04 E6 CE D0 C6
8AD0- C9 C4 D0 C2 20 B1 00 C9
8AD8- 30 90 CC C9 3A B0 C8 90
8AE0- 05 E6 1E 20 B1 00 20 4A
8AE8- EC 20 52 E7 A9 8B 85 19
8AF0- A9 DD 85 18 A0 00 B1 18
8AF8- C5 50 D0 0B C8 B1 18 C9
8B00- FA B0 16 C5 51 F0 38 C9
8B08- FF F0 0E A5 18 18 69 04
8B10- 85 18 90 02 E6 19 4C F4
8B18- 8A A0 8B A9 30 20 3A DB
8B20- A5 51 A6 50 20 24 ED 20
8B28- 62 FC 20 3A FF 4C 28 8A
8B30- 8D D0 C1 D3 A0 C4 C5 A0
8B38- CC C9 C7 CE C5 A0 00 C8
8B40- B1 18 85 9F C8 B1 18 85
8B48- 9E A2 90 38 20 A0 EB 20
8B50- 34 ED 84 EF 85 EE A0 00
8B58- B1 EE F0 06 20 A0 8B C8
8B60- D0 F6 20 B7 00 20 A0 8B
8B68- F0 03 4C 96 8A 4C 76 8A
8B70- 20 A0 8B A5 06 A6 07 A0
8B78- 00 18 65 1D 91 06 90 01
8B80- E8 8A C8 91 06 B1 06 AA
8B88- 88 B1 06 85 06 86 07 A0
8B90- 00 84 1D 84 1E 84 1F 84
8B98- CE 84 FF 84 FD 84 FC 60
8BA0- 18 8D 00 40 E6 1D EE A2
8BA8- 8B D0 03 EE A3 8B AA 60
8BB0- A5 50 20 A0 8B A5 51 20
8BB8- A0 8B 60 A5 18 20 D0 8B
8BC0- A5 19 20 D0 8B A5 50 20
8BC8- D0 8B A5 51 20 D0 8B 60
8BD0- 18 8D DD 8B EE D2 8B D0
8BD8- 03 EE D3 8B 60
```

Yvan Koenig

Un désassembleur 65C02

Le problème : Vous disposez d'une routine binaire conçue pour l'Apple //c ou le nouvel Apple //e. Pour l'adapter à votre machine, dépourvu du 65C02, le listing désassemblé est indispensable. Comment l'obtenir ?

Jusqu'à présent, nous ne disposions que du désassemblage manuel à l'aide de la liste des codes, solution lente et source de nombreuses erreurs ; un nouveau désassembleur devenait donc indispensable.

Le programme ci-dessous vous permettra de franchir sans difficulté la première partie de votre travail : on copie en RAM le moniteur et on le modifie pour qu'il s'ouvre aux STZ, TRB et autres INA. Le fichier 'CODEC' n'est qu'un exemple.

Source 'DESASM/RAM.S'

Assembleur Big Mac

```
 1
 2 ********************************
 3 *
 4 * APPLE II ** DESASSEMBLEUR 65C02
 5 *
 6 * Version RAM.
 7 *
 8 * par Y.KOENIG
 9 *  le 11/03/85
10 *
11 ********************************
12 *
13 * Ce programme copie le MONITEUR
14 * en RAM et le modifie en
15 * s'inspirant du MONITEUR //c
16 * afin de permettre de
17 * désassembler les codes du 65C02
18 *
19 * Ctrl Y bascule d'un moniteur
20 *  à l'autre.
21 * Le désassemblage se fait alors
22 *  avec la syntaxe habituelle.
23 *
24 ********************************
25
26          ORG  $803
27
28 LOC0     =    $0        &1
29 LENGTH   =    $2F
30 A1L      =    $3C       &3D
31 A2L      =    $3E       &3F
32 A4L      =    $42       &43
33 ACC      =    $45
34 *
35 CTRLYV   =    $3F8
36 *
37 MSLOT    =    $7F8
38 *
39 FMT1     =    $F962
40 FIXSEV   =    $FA9B
41 NXTBYT   =    $FAC7
42 RDSP1    =    $FAE4
43 DISKID   =    $FAFF
44 HEADR    =    $FCC9
45 NNEWOP   =    HEADR+1
46 PRBYTE   =    $FDDA
47 MOVE     =    $FE2C
48 WRITE    =    $FECD
49
50          BIT  $C081
51          BIT  $C081       pour écrire
             en RAM
52          LDA  #$FF
53          STA  A2L
54          STA  A2L+1
55          LDA  #>$D000
56          STA  A1L+1
57          STA  A4L+1
58          LDY  #<$D000
59          STY  A1L
60          STY  A4L
61          JSR  MOVE        copie le mo
             niteur et applesoft en RAM
62
63          LDA  #$C9
64          STA  $FFF3       annule Writ
             e
65          STA  $FFF5       annule Read
66          LDA  #$B2
67          STA  $FFDC
68          STA  $FFDE
69
70          LDA  #$B9
71          STA  $F919
72          LDA  #$B3
73          STA  $F91F
74
75          LDX  #3
76          LDA  #$EA        'NOP'
77 LOOP1    STA  $F895,X
78          DEX
79          BPL  LOOP1       dépose 4 NO
             P
80
81          LDA  #$FC
82          STA  $F8A6
83
84          LDX  #NF8AF-NFAF4-1
85 LOOP2    LDA  NFAF4,X
86          STA  $FAF4,X
87          DEX
88          BPL  LOOP2
89
90          LDX  #NFAF4-NFAB6-1
91 LOOP3    LDA  NFAB6,X
92          STA  $FAB6,X
93          DEX
94          BPL  LOOP3
95
96          LDX  #NWRITE-NFMT1
97 LOOP4    LDA  NFMT1-1,X
98          STA  FMT1-1,X
99          DEX
100         BNE  LOOP4       copie la no
             uvelle table
101
102         LDX  #NHEADR-NWRITE-1
103 LOOP5   LDA  NWRITE,X
104         STA  WRITE,X
105         DEX
106         BPL  LOOP5       copie le PA
             TCHE dans WRITE
107
108         LDX  #NFAB6-NHEADR-1
109 LOOP6   LDA  NHEADR,X
110         STA  HEADR,X
111         DEX
112         BPL  LOOP6       copie la ta
             ble dans HEADR
113
114         LDX  #N3C2-NF8AF-1
115 LOOP7   LDA  NF8AF,X
116         STA  $F8AF,X
117         DEX
118         BPL  LOOP7       copie 2 ins
             tructions
119
120         BIT  $C082       protège la
             RAM
121
122         LDA  #<$3C2
123         STA  CTRLYV+1
124         LDA  #>$3C2
125         STA  CTRLYV+2    Initialise
             ctrl Y
126         LDX  #FIN-N3C2-1
127 LOOP8   LDA  N3C2,X
128         STA  $3C2,X
129         DEX
130         BPL  LOOP8
131         JMP  $3D0        ;C'est fini
             !
132
133 NFMT1    HEX  0F22FF33CB62FF730322FF
             33CB66FF77
134         HEX  0F20FF33CB60FF700F22FF
             39CB66FF7D
135         HEX  0B22FF33CBA6FF731122FF
             33CBA6FF87
136         HEX  0122FF33CB60FF700122FF
             33CB60FF70
137         HEX  24316578
138 FMT2    HEX  00218182594D9192864A85
             9D495A
139 * ATTENTION, astuce Apple, CHAR2 com
             plète FMT2
140 CHAR2   ASC  "Y"
141         BRK
142         ASC  "X$$"
143         BRK
144 CHAR1   ASC  ",),#($"
145
146 MNEML    HEX  1C8A1C235D8B1BA19D8A1D
             239D8B1DA1
147         HEX  1C2919AE69A8192324531B
             23245319A1
148         HEX  AD1A5B5BA5692424AEAEA8
             AD298A7C8B
149         HEX  159C6D9CA5692953841334
             11A56923A0
150 *
151 *
152 MNEMR    HEX  D8625A48266294885444C8
             546844E894
153         HEX  C4B4088474B4286E74F4CC
             4A72F2
154         HEX  A48A06AAA2A27474747244
             68B232B272
155         HEX  22721A1A2626727288C8C4
             CA26484444
156         HEX  A2C8
157
158
159 * A recopier en WRITE
160 NWRITE   RTS
161 PATCHE   AND  #3
162         STA  LENGTH
163 DECODNEW TYA
164         LDX  #22
165 DECOD0   CMP  NNEWOP,X
```

```
166           BEQ  DECOD1
167           DEX
168           BPL  DECOD0
169           RTS
170 DECOD1    LDA  NNEWOP+NEWOP0-NEWOP,X
171           LDY  #0
172           RTS
173
174 NEWFAF4   LDA  ACC+5,X
175           JSR  PRBYTE
176           INX
177           BMI  TORDSP1
178           RTS
179 TORDSP1   JMP  RDSP1
180
181 * A recopier en HEADR
182 NHEADR    RTS
183 NEWOP     HEX  12        ORA (zp)
184           HEX  14        TRB zp
185           HEX  1A        INA
186           HEX  1C        TRB abs
187           HEX  32        AND zp
188           HEX  34        BIT zp
189           HEX  3A        DEA
190           HEX  3C        BIT abs,X
191           HEX  52        EOR (zp)
192           HEX  5A        PHY
193           HEX  64        STZ zp
194           HEX  72        ADC zp
195           HEX  74        STZ zp,X
196           HEX  7A        PLY
197           HEX  7C        JMP abs(X)
198           HEX  89        BIT imm
199           HEX  92        STA (zp)
200           HEX  9C        STZ abs
201           HEX  9E        STZ abs,X
202           HEX  B2        LDA (zp)
203           HEX  D2        CMP (zp)
204           HEX  F2        SBC (zp)
205           HEX  FC        INVALIDE ->
                   NOP
206 NEWOP0    HEX  38
207           HEX  FB
208           HEX  37
209           HEX  FB
210           HEX  39
211           HEX  21
212           HEX  36
213           HEX  21
214           HEX  3A
215           HEX  F8
216           HEX  FA
217           HEX  3B
218           HEX  FA
219           HEX  F9
220           HEX  22
221           HEX  21
222           HEX  3C
223           HEX  FA
224           HEX  FA
225           HEX  3D
226           HEX  3E
227           HEX  3F
228           HEX  FC
229
230 NEWSLOOP  DEC  LOC0+1
231           LDA  LOC0+1
232           CMP  #$C0
233           BEQ  TOFIXSEV
234           STA  MSLOT
235           JMP  NXTBYT
236 TOFIXSEV  JMP  FIXSEV
237
238 * A recopier en $FAB6
239 NFAB6     BNE  FABE
240           HEX  8A8BA5AC00 ;Fin table
              MNEML
241           HEX  EA
242 FABE      STX  LOC0
243           STA  LOC0+1
244 NSLOOP    LDY  #7
245           JMP  HEADR+NEWSLOOP-NHEADR
246 NNXTBYT   LDA  (LOC0),Y
247           CMP  DISKID-1,Y
248           BNE  NSLOOP
249
250 * A recopier en $FAF4
251 NFAF4     JMP  WRITE+NEWFAF4-NWRITE
252           HEX  DA
253           HEX  747476C600 ;Fin de tab
              le MNEMR
254
255 NF8AF     JSR  WRITE+PATCHE-NWRITE
256           BEQ  *+26
257
258 * A recopier en $3C2
259 N3C2      LDA  $3CD
260           EOR  #$81
261           ORA  #$80
262           STA  $3CD
263           BIT  $C081
264           RTS
265 FIN       =    *
266           LST  OFF
```

Comment faire ?

Pour désassembler les codes du 65C02, il suffit de faire :

- *BRUN DESASM/RAM ;*
- *charger un fichier binaire comprenant les nouvelles instructions, le fichier 'CODEC' par exemple ;*
- *passer en moniteur (CALL -151). En utilisant alors la commande L du moniteur, vous obtiendriez de nombreux '???' ;*
- *faire CTRL Y <RETURN> ;*
- *utiliser alors la commande L (avec l'exemple CODEC, faire 300L).*

Vous obtenez :

```
0300-   DA          PHX
0301-   80 03       BRA   $0306
0303-   1A          INC
0304-   3A          DEC
0305-   5A          PHY
0306-   FA          PLX
0307-   7A          PLY
0308-   9C 03 03    STZ   $0303
030B-   9E 04 04    STZ   $0404,X
030E-   64 06       STZ   $06
0310-   74 07       STZ   $07,X
0312-   1C 02 02    TRB   $0202
0315-   14 09       TRB   $09
0317-   0C 09 09    TSB   $0909
031A-   04 04       TSB   $04
031C-   89 08       BIT   #$08
031E-   7C 07 07    JMP   ($0707,X)
0321-   3C 06 06    BIT   $0606,X
0324-   12 07       ORA   ($07)
0326-   92 05       STA   ($05)
```

- *CTRL Y vous permet de passer du désassembleur original au désassembleur enrichi.*

Récapitulation 'DESASM/RAM'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE DESASM/RAM,A$803,L$216

```
0803- 2C 81 C0 2C 81
0808- C0 A9 FF 85 3E 85 3F A9
0810- D0 85 3D 85 43 A0 00 84
0818- 3C 84 42 20 2C FE A9 C9
0820- 8D F3 FF 8D F5 FF A9 B2
0828- 8D DC FF 8D DE FF A9 B9
0830- 8D 19 F9 A9 B3 8D 1F F9
0838- A2 03 A9 EA 9D 95 F8 CA
0840- 10 FA A9 FC 8D A6 F8 A2
0848- 08 BD FD 09 9D F4 FA CA
0850- 10 F7 A2 17 BD E5 09 9D
0858- B6 FA CA 10 F7 A2 DE BD
0860- A3 08 9D 61 F9 CA D0 F7
0868- A2 22 BD 82 09 9D CD FE
0870- CA 10 F7 A2 3F BD A5 09
0878- 9D C9 FC CA 10 F7 A2 04
0880- BD 06 0A 9D AF F8 CA 10
0888- F7 2C 82 C0 A9 C2 8D F9
0890- 03 A9 03 8D FA 03 A2 0D
0898- BD 0B 0A 9D C2 03 CA 10
08A0- F7 4C D0 03 0F 22 FF 33
08A8- CB 62 FF 73 03 22 FF 33
08B0- CB 66 FF 77 0F 20 FF 33
08B8- CB 60 FF 70 0F 22 FF 39
08C0- CB 66 FF 7D 0B 22 FF 33
08C8- CB A6 FF 73 11 22 FF 33
08D0- CB A6 FF 87 01 22 FF 33
08D8- CB 60 FF 70 01 22 FF 33
08E0- CB 60 FF 70 24 31 65 78
08E8- 00 21 81 82 59 4D 91 92
08F0- 86 4A 85 9D 49 5A D9 00
```

```
08F8- D8 A4 A4 00 AC A9 AC A3
0900- A8 A4 1C 8A 1C 23 5D 8B
0908- 1B A1 9D 8A 1D 23 9D 8B
0910- 1D A1 1C 29 19 AE 69 A8
0918- 19 23 24 53 1B 23 24 53
0920- 19 A1 AD 1A 5B 5B A5 69
0928- 24 24 AE AE A8 AD 29 8A
0930- 7C 8B 15 9C 6D 9C A5 69
0938- 29 53 84 13 34 11 A5 69
0940- 23 A0 D8 62 5A 48 26 62
0948- 94 88 54 44 C8 54 68 44
0950- E8 94 C4 B4 08 84 74 B4
0958- 28 6E 74 F4 CC 4A 72 F2
0960- A4 8A 06 AA A2 A2 74 74
0968- 74 72 44 68 B2 32 B2 72
0970- 22 72 1A 1A 26 26 72 72
0978- 88 C8 C4 CA 26 48 44 44
```

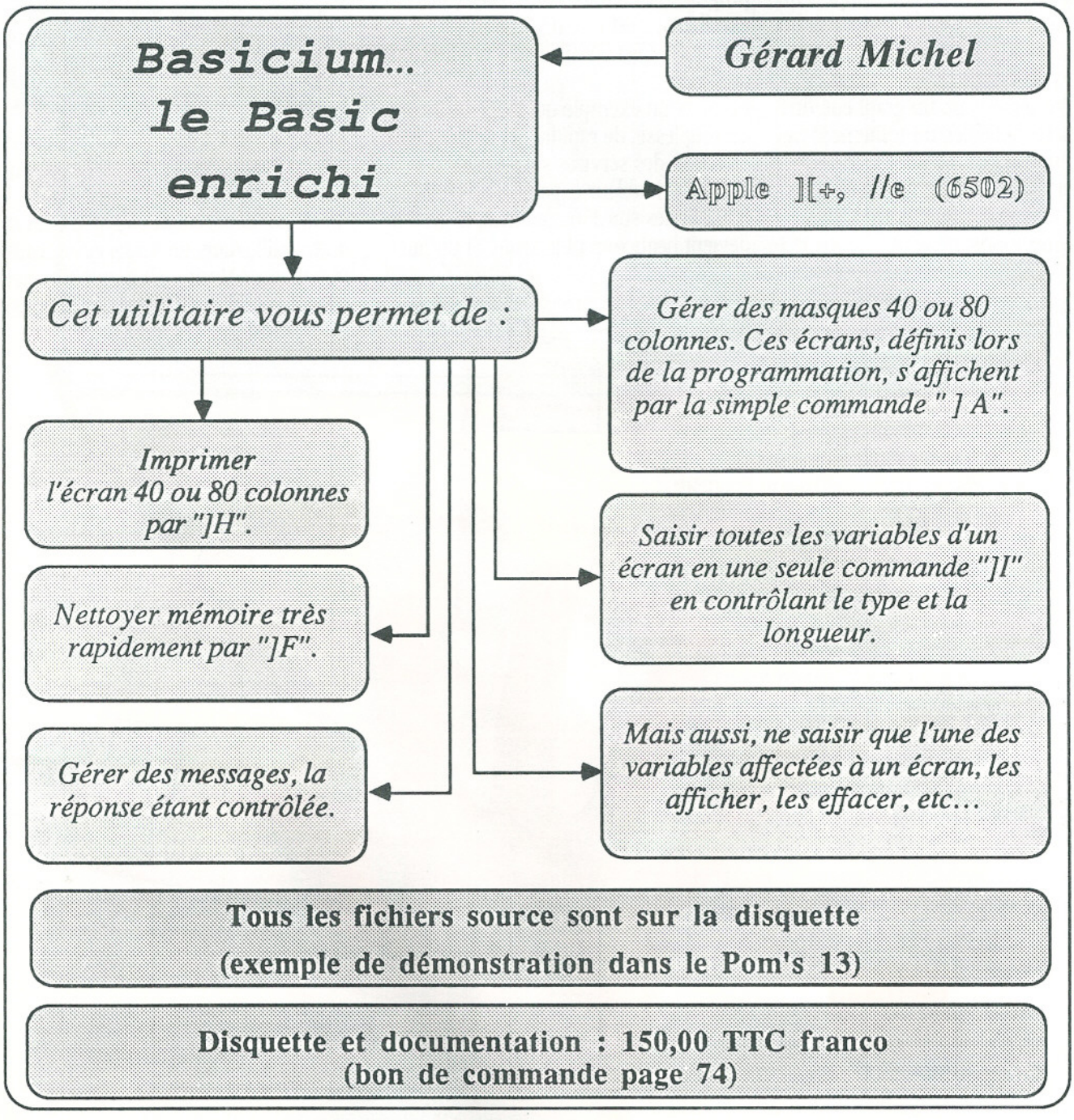
```
0980- A2 C8 60 29 03 85 2F 98
0988- A2 16 DD CA FC F0 04 CA
0990- 10 F8 60 BD E1 FC A0 00
0998- 60 B5 4A 20 DA FD E8 30
09A0- 01 60 4C E4 FA 60 12 14
09A8- 1A 1C 32 34 3A 3C 52 5A
09B0- 64 72 74 7A 7C 89 92 9C
09B8- 9E B2 D2 F2 FC 38 FB 37
09C0- FB 39 21 36 21 3A F8 FA
09C8- 3B FA F9 22 21 3C FA FA
09D0- 3D 3E 3F FC C6 01 A5 01
09D8- C9 C0 F0 06 8D F8 07 4C
09E0- C7 FA 4C 9B FA D0 06 8A
09E8- 8B A5 AC 00 EA 86 00 85
09F0- 01 A0 07 4C F8 FC B1 00
09F8- D9 FE FA D0 F4 4C E4 FE
0A00- DA 74 74 76 C6 00 20 CE
```

```
0A08- FE F0 18 AD CD 03 49 81
0A10- 09 80 8D CD 03 2C 81 C0
0A18- 60
```

Fichier exemple 'CODEC'

Après avoir saisi ce code sous moniteur, sauvegardez-le par BSAVE CODEC,A$300,L$29

```
0300- DA 80 03 1A 3A 5A FA 7A
0308- 9C 03 03 9E 04 04 64 06
0310- 74 07 1C 02 02 14 09 0C
0318- 09 09 04 04 89 08 7C 07
0320- 07 3C 06 06 12 07 92 05
0328- 00
```



Tant qu'à faire 5 choses à la fois,

Que ceux qui aiment travailler en faisant deux ou trois choses à la fois ne changent rien, au contraire, avec Apple et Jazz ils peuvent faire mieux.

Jazz de Lotus, c'est un programme créé pour Macintosh 512 Ko, équipé d'un lecteur externe qui permet de devenir un parfait jongleur professionnel.

Cinq programmes en concert, c'est-à-dire un tableur, un grapheur, un gestionnaire de fichier, un traitement de texte et un programme de communication réunis en un seul programme. Ou comment être à cinq sur la même souris.

Pouvoir gérer cinq applications à partir d'un seul écran, modifier une donnée dans une application et qu'elle se modifie automatiquement dans les autres, pouvoir sauter d'un graphe à un traitement de texte sans attendre les secondes qui durent une éternité pour changer de programme, c'est bien...

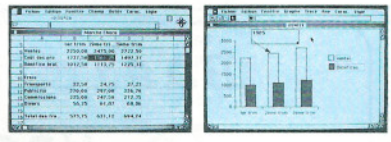
Quand on s'aperçoit que ces cinq programmes sont individuellement excellents, c'est une révolution.

Jazz, c'est l'outil idéal d'un directeur de service.

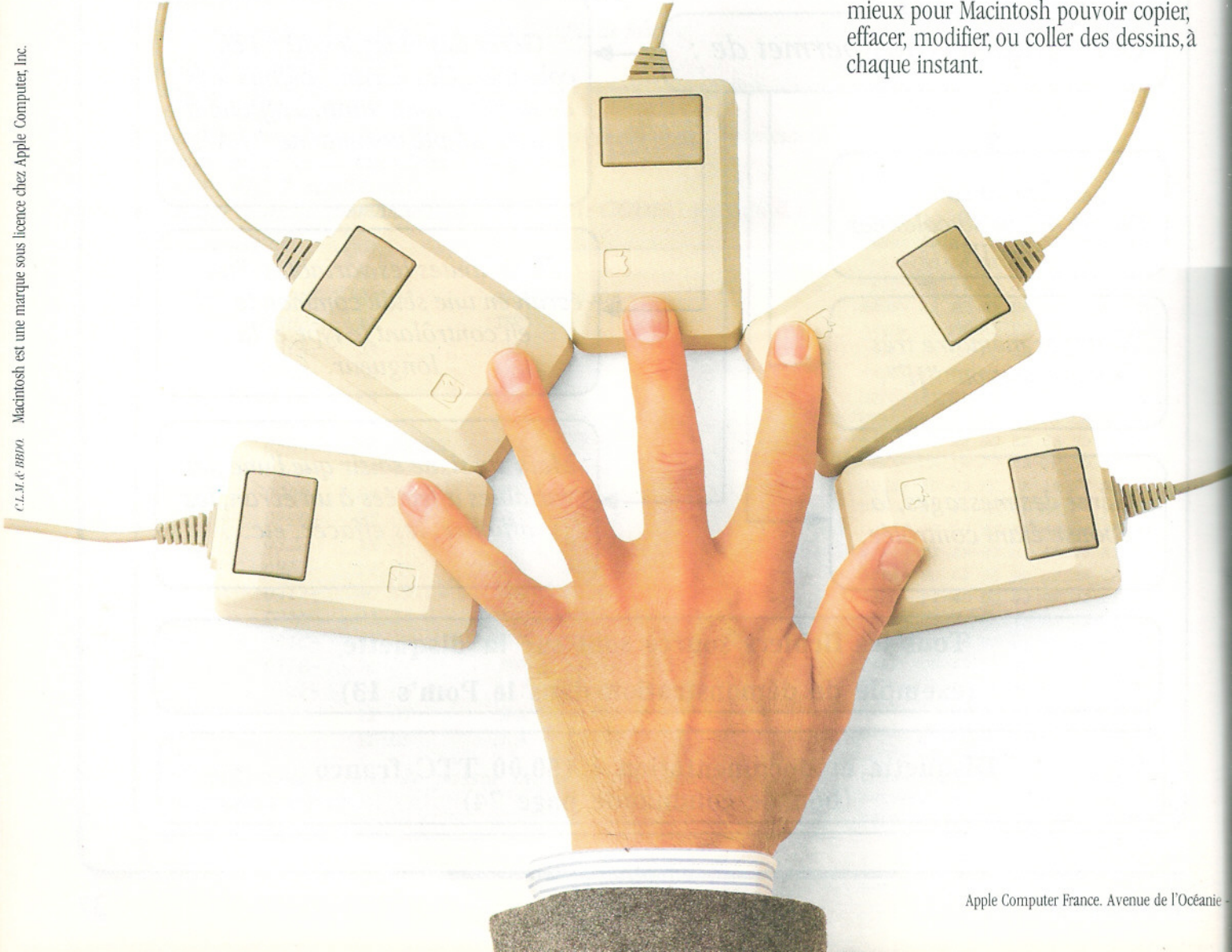
Prenons un exemple qui exige beaucoup de souplesse, de rapidité, et de doigté, la direction des services secrets :

– Avec un tableur capable d'afficher 8192 lignes sur 256 colonnes, la gestion devient beaucoup plus facile. Si un fait nouveau apparaît, rien n'est plus facile que d'étudier deux ou trois hypothèses afin de ne pas être pris de court.

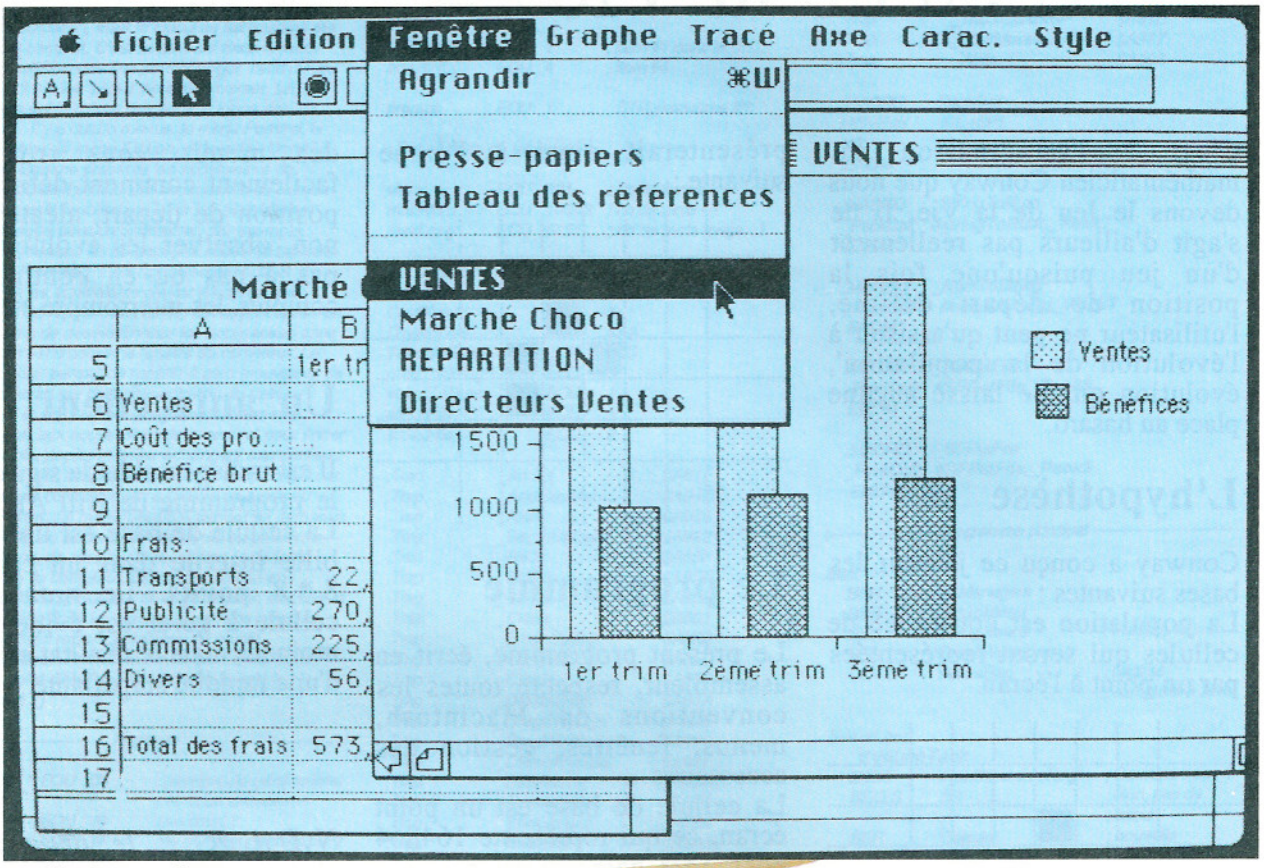
– Quand un projet est fin prêt, il faut bien le présenter à ses supérieurs et parfois même beaucoup plus haut; un grapheur permet de transformer toutes les données numériques obscures en graphiques lumineux.



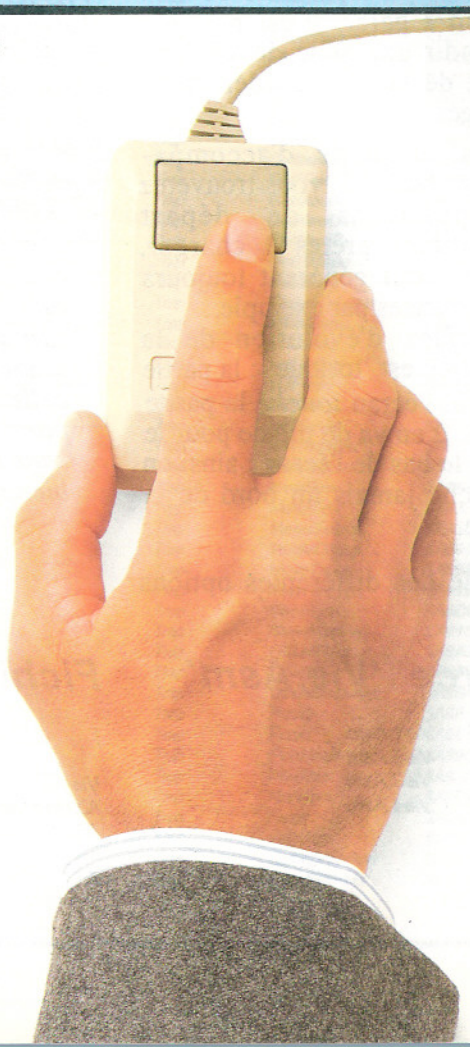
– Si le projet est accepté, un traitement de texte est nécessaire pour que chaque agent concerné soit au courant dans ses moindres détails. Evidemment quand on travaille dans un tel service il vaut mieux pour Macintosh pouvoir copier, effacer, modifier, ou coller des dessins, à chaque instant.



autant les faire en même temps.



Feuille de travail Jazz.



Grâce à son gestionnaire de fichier, Jazz fait la fusion automatique entre un mémo et une liste de correspondants à sélectionner selon un critère spécial.
Et avec un modem full duplex 1200 bauds en un instant votre correspondant reçoit toutes les informations précises à l'autre bout du monde.
Avec Jazz l'entreprise est rentable, rondement menée, précise et discrète, inutile d'avoir 35 collaborateurs pour rédiger le projet.

Jazz c'est cinq programmes liés entre eux si parfaitement que l'agent double est enfin enterré, place à l'agent quintuple.

Apple

B.P. 131, 91944 Les Ulis Cedex.

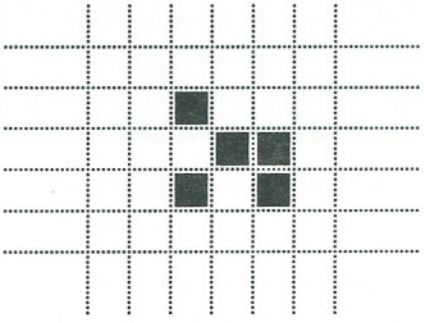
Jeu de la Vie

Dominique Bernardi

C'est à l'imagination du mathématicien Conway que nous devons le Jeu de la Vie. Il ne s'agit d'ailleurs pas réellement d'un jeu puisqu'une fois la position de départ définie, l'utilisateur ne peut qu'assister à l'évolution de la 'population', évolution qui ne laisse aucune place au hasard.

L'hypothèse

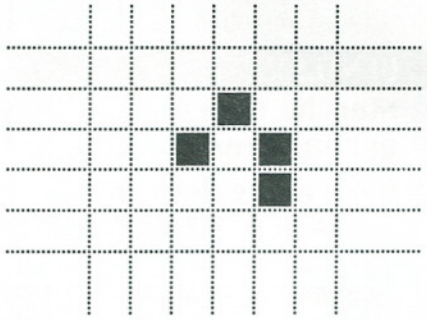
Conway a conçu ce jeu sur les bases suivantes :
La population est constituée de cellules qui seront représentées par un point à l'écran.



Chaque cellule peut avoir jusqu'à 8 voisines et cet entourage en détermine la survie :

- 0 ou 1 cellule adjacente : mort par isolement ;
- 2 cellules : survie ;
- 3 cellules adjacentes à une position vide : naissance ;
- plus de 4 cellules : mort par étouffement.

La position définie plus haut se présenterait ainsi à l'étape suivante :



Le programme

Le présent programme, écrit en assembleur, respecte toutes les conventions du Macintosh, menus, fenêtres, gestion des accessoires...
La cellule de base est un point écran, ce qui représente 164864 position à gérer. Il est possible d'agrandir une zone de 46 * 64 points, de la sauvegarder, de la recharger.

Sur la disquette d'accompagnement Mac 23, vous trouverez des positions de départ particulières prêtes à charger, positions qui génèrent toujours des naissances par exemple.
La surface d'évolution de la population est 'torique' : la partie droite de l'écran *touche* la partie gauche. Il en va de même pour le haut et le bas de l'écran ; ainsi on ne touche jamais un bord de la surface.

A l'aide des différentes options des menus vous trouverez facilement comment définir une position de départ, aléatoire ou non, observer les évolutions en pas à pas ou en continu, les compter, les interrompre etc.

Un antécédent

Il est indispensable de signaler ici le programme de Bill Atkinson. La cellule de base est une petite bille inscrite dans un carré de 8 * 8 points. Le nombre de cellule à gérer est évidemment moindre mais le résultat est alors d'une rapidité stupéfiante.

***N.D.L.R.** : le programme tourne indifféremment sur un Mac 512Ko ou un Mac 128Ko. En revanche, le chargement et l'édition des sources sont problèmatiques sur un 128Ko ; la disquette d'accompagnement Mac 23 contient donc les fichiers sources en deux versions :*

- *celles dont les noms débutent par 'Vie.', listées page suivantes et destinées aux Macs 512Ko ;*
- *celles dont les noms débutent par 'Vie128.', sauvegardées sans remarques et destinées, bien sûr , aux Macs 128Ko.*

Fichier 'Vie.Job' (Exec)

```
Asm       Vie.Asm     Exec     Edit
Link      Vie.Link    Exec     Edit
RMaker    Vie.R       Finder   Edit
```

Source 'Vie.Asm'

```
INCLUDE   Vie1.Asm
INCLUDE   Vie2.Asm
INCLUDE   Vie3.Asm
INCLUDE   Vie4.Asm
END
```

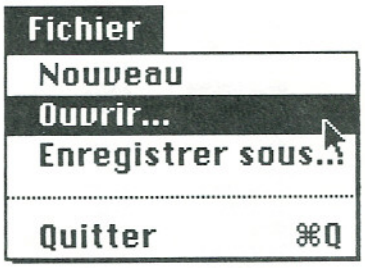
Fichier 'Vie.Link'

```
]

Vie
/Output Vie.Code
/Type 'TEMP'

$
```

Source 'Vie1.Asm'

```
;
; Le tableau principal occupe tout l'écran sauf la barre
; de menu. Il est formé de 322 lignes de 512 points.
; En fait, il est torique, c'est à dire que ce qui sort par la
; gauche réapparait à droite etc. On peut aussi
; agrandir un rectangle 46 x 64 extrait de ce tableau
; ('basse résolution'), mais le calcul se fait toujours sur
; le tableau complet. C'est seulement dans le mode
; basse résolution que l'on peut changer l'état du jeu
; en cliquant sur les 'pixels' pour les inverser. La
; procédure est la même que sous la loupe de
; MacPaint. Il y a quatre menus: le menu Pomme, le
; menu Fichier, le menu Edition, et le menu Contrôle.
; Le menu Pomme presente les accessoires de
; bureau et C'est la Vie... Les accessoires de bureau
; devraient tous fonctionner, et le jeu peut tourner
; avec l'horloge pour chronométrer, par exemple.
; Toutefois, comme on ne peut pas activer le tableau
; principal, l'accessoire est normalement actif et
; intercepte les équivalents clavier (l'horloge, par
; exemple, les rejette avec un bip désapprobateur. Le
; seul moyen de désélectionner les accessoires sans
; les fermer est d'activer la fenêtre du compteur. Le
; menu Fichier permet de repartir à zéro (nouveau), de
; lire ou d'écrire sur disque le contenu du petit
; rectangle et de quitter le programme. Le menu
; Edition sert aux accessoires de bureau et pour éditer
; le rectangle, copier et coller échangent son contenu
; avec celui d'un tampon, effacer agit uniquement sur
; ce rectangle contrairement à l'article nouveau du
; menu Fichier. Le menu Contrôle permet de démarrer
; et d'arrêter, ou de calculer une seule ou plusieurs
; generations, de changer la résolution et de faire
; apparaître le compteur. Ce dernier affiche en
; première ligne le numéro de la génération affichée et
; en deuxième ligne le nombre de cellules vivantes.
; Enfin, on peut remplir le tableau de points
; aleatoirement noirs ou blancs, la probabilité pour
; chaque point d'être noir étant réglable (article
; Probabilité...)
;
;---------- Registres ----------
statusReg   EQU  D3   ; nombre de générations
                      ; et flags ci-dessous
runBit      EQU  16   ; on court ?
countBit    EQU  17   ; le compteur est visible ?
resBit      EQU  18   ; Haute résolution ?
deterBit    EQU  19   ; La couleur du pixel à
                      ; mettre est connue ?
colorBit    EQU  20   ; bits 19 et 20 = 1 <=>
                      ; c'est du noir
quitBit     EQU  21   ; Quitter a été
                      ; sélectionné ?

ModifyReg   EQU  D4   ; état du bouton, des
                      ; touches spéciales etc.
MenuReg     EQU  D5   ; ID du menu choisi
MItemReg    EQU  D6   ; ID de l'article choisi
AppleHReg   EQU  D7   ; handle vers le menu
                      ; Pomme

Window1     EQU  A3   ; A3 pointe vers la
                      ; fenêtre principale
window2     EQU  A2   ; A2 pointe vers la
                      ; fenêtre du compteur

;---------- Équivalences ----------
AppleMenu   EQU  1
  AboutItem EQU  1

FileMenu    EQU  2
  NewItem   EQU  1
  OpenItem  EQU  2
  SaveItem  EQU  3
  QuitItem  EQU  5

EditMenu    EQU  3
  copyItem  EQU  4
  pasteItem EQU  5
  ClearItem EQU  6

CtlMenu     EQU  4
  GoItem    EQU  1
  StopItem  EQU  2
  StepItem  EQU  3
  NGenItem  EQU  4
  LoItem    EQU  6
  HiItem    EQU  7

  CountItem EQU  9
  ProbItem  EQU  11
  AlealItem EQU  12

AboutAlert  EQU  1    ; ressources ALRT et
                      ; DLOG
GenDialog   EQU  2
ProbaDialog EQU  3

OKItem      EQU  1    ; Item #1 dans la DITL
editText    EQU  4    ; Item #4

prompt      EQU  1    ; STR ressource #1

; Dimensions du tableau :

vMax        EQU  321      ; nb de lignes - 1
maxWord     EQU  10303    ; nb de mots - 1
maxLong     EQU  5151     ; nb de mots longs - 1

NoGrowDocProc   EQU  4
PlainDBox       EQU  2
CmdKey          EQU  8
DWindLen        EQU  $AA
WindowSize      EQU  $9C
numToString     EQU  0
stringToNum     EQU  1
sfPutFile       EQU  1
sfGetFile       EQU  2

.Trap   _AddPt          $A87E
.Trap   _AddResMenu     $A94D
.Trap   _Alert          $A985
.Trap   _BeginUpDate    $A922
.Trap   _BitClr         $A85F
.Trap   _BitSet         $A85E
.Trap   _BitTst         $A85D
.Trap   _Close          $A001
.Trap   _CloseDialog    $A982
.Trap   _CopyBits       $A8EC
.Trap   _Create         $A008
.Trap   _DisableItem    $A93A
.Trap   _DragGrayRgn    $A905
.Trap   _DragWindow     $A925
.Trap   _DrawMenuBar    $A937
.Trap   _DrawString     $A884
.Trap   _EnableItem     $A939
.Trap   _EndUpdate      $A923
.Trap   _EraseRect      $A8A3
.Trap   _ExitToShell    $A9F4
.Trap   _FindWindow     $A92C
.Trap   _FlushEvents    $A032
.Trap   _FlushVol       $A013
.Trap   _FrameRect      $A8A1
.Trap   _GetCursor      $A9B9
.Trap   _GetDItem       $A98D
.Trap   _GetItem        $A946
.Trap   _GetIText       $A990
.Trap   _GetMouse       $A972
.Trap   _GetNewDialog   $A97C
.Trap   _GetNextEvent   $A970
.Trap   _GetRMenu       $A9BF
.Trap   _GetString      $A9BA
.Trap   _GlobalToLocal  $A871
.Trap   _HiliteMenu     $A938
.Trap   _InitCursor     $A850
.Trap   _InitDialogs    $A97B
.Trap   _InitFonts      $A8FE
.Trap   _InitGraf       $A86E
.Trap   _InitMenus      $A930
.Trap   _InitWindows    $A912
.Trap   _InsertMenu     $A935
.Trap   _InverRect      $A8A4
.Trap   _MenuKey        $A93E
.Trap   _MenuSelect     $A93D
.Trap   _ModalDialog    $A991
.Trap   _MoveTo         $A893
.Trap   _NewPtr         $A11E
.Trap   _NewRgn         $A8D8
.Trap   _NewWindow      $A913
.Trap   _OffsetRect     $A8A8
.Trap   _Open           $A000
.Trap   _OpenDeskAcc    $A9B6
.Trap   _Pack3          $A9EA
.Trap   _Pack7          $A9EE
.Trap   _PenMode        $A89C
.Trap   _PtInRect       $A8AD
.Trap   _Random         $A861
.Trap   _Read           $A002
.Trap   _RectRgn        $A8DF
.Trap   _SelectWindow   $A91F
.Trap   _SendBehind     $A921
.Trap   _SetCursor      $A851
.Trap   _SetPort        $A873
.Trap   _SubPt          $A87F
.Trap   _SysEdit        $A9C2
.Trap   _SystemClick    $A9B3
.Trap   _SystemTask     $A9B4
.Trap   _TEInit         $A9CC
.Trap   _TextFont       $A887
.Trap   _TrackGoAway    $A91E
.Trap   _WaitMouseUp    $A977
.Trap   _Write          $A003

.MACRO  _PackCall
MOVE.W  %1,-(SP)
%2
.ENDM

.MACRO  _StringToNum
_PackCall  #stringToNum,_Pack7
.ENDM

.MACRO  _NumToString
_PackCall  #numToString,_Pack7
.ENDM

.MACRO  _SFGetFile
_PackCall  #SFGetFile,_Pack3
.ENDM

.MACRO  _SFPutFile
_PackCall  #SFPutFile,_Pack3
.ENDM

;---------- Programme principal ----------

Start
  BSR     InitManagers
  BSR     SetupMenu
  BSR     SetupBoard        ; Prépare le jeu
  BSR     CreateWindows
  CLR.L   statusReg         ; flags et
                            ; compteur à zéro

EventLoop
  _SystemTask
  BTST    #runBit,statusReg ; Courons-nous ?
  BEQ.S   @1                ; Non, pas de
                            ; calcul
  BSR     Calcule           ; nouvelle
                            ; génération
  BSR     Affiche           ; affichage
  BSR     AffCompte         ; affiche le
                            ; nouveau
                            ; compte
@1
  CLR     -(SP)
  MOVE    #$007F,-(SP)      ; 7 premiers
                            ; événements
  PEA     EventRecord
  _GetNextEvent             ; Événement
                            ; suivant
  TST     (SP)+             ; Existe-t-il ?
  BEQ.S   EventLoop         ; R.A.S... On
                            ; boucle
  BSR     HandleEvent
  BTST    #quitBit,statusReg
  BEQ.S   EventLoop
  RTS                       ; e finita la
                            ; commedia

;---------- InitManagers ----------

InitManagers                ; Standard
  PEA     -4(A5)
  _InitGraf
  _InitFonts
  MOVE.L  #$0000FFFF,D0
  _FlushEvents
  _InitWindows
  _InitMenus
  CLR.L   -(SP)
  _InitDialogs
  _TEInit
  _InitCursor
  RTS

;---------- SetupMenu ----------

SetupMenu
  CLR.L   -(SP)             ; espace pour
                            ; handle
  MOVE    #AppleMenu,-(SP)
```

```
        _GetRMenu
        MOVE.L  (SP),AppleHReg
        MOVE.L  (SP),-(SP)      ; Copie pour
                                ; AddResMenu
        CLR     -(SP)
        _InsertMenu
        MOVE.L  #'DRVR',-(SP)   ; Accessoires de
                                ; bureau
    _AddResMenu
Fichier:
    CLR.L   -(SP)
    MOVE    #FileMenu,-(SP)
    _GetRMenu
    CLR     -(SP)
    _InsertMenu
Edition:
    CLR.L   -(SP)
    MOVE    #EditMenu,-(SP)
    _GetRMenu
    CLR     -(SP)
    _InsertMenu
Controle:
    CLR.L   -(SP)
    MOVE    #CtlMenu,-(SP)
    _GetRMenu
    LEA     CtlHandle,A0
    MOVE.L  (SP),(A0)       ; mettons la
                            ; handle de côté
    CLR     -(SP)
    _InsertMenu
    _DrawMenuBar
    RTS

;---------- SetupBoard ----------

SetupBoard
    ;ATTENTION: NewPtr utilise des registres pour les
    ; arguments:
    ; taille du bloc à réserver en entrée dans D0, en
    ; sortie, le pointeur est dans A0, sauf si D0 contient
    ; un code d'erreur non nul 322 * 64 pour le tableau,
    ; 5 * 516 pour les lignes, 322 pour le tableau de
    ; booleens indiquant les rangées non vides enfin 2
    ; octets de 'bourrage' ---> 23512 octets en tout.
    MOVE.L  #23512,D0
    _NewPtr                 ; Demandons de
                            ; la place
    TST     D0              ; Erreur ?
    BEQ.S   @1              ; Non, tout va
                            ; bien.
    _ExitToShell            ; Faute de
                            ; mieux...
@1
    LEA     Board,A1        ; Sauvons les
                            ; différents
    MOVE.L  A0,(A1)         ; pointeurs
    ADDA    #20608,A0
    LEA     moins2,A1
    MOVE.L  A0,(A1)
    ADDA    #516,A0
    LEA     moins1,A1
    MOVE.L  A0,(A1)
    ADDA    #516,A0
    LEA     pas0,A1
    MOVE.L  A0,(A1)
    ADDA    #516,A0
    LEA     pas1,A1
    MOVE.L  A0,(A1)
    ADDA    #516,A0
    LEA     pas2,A1
    MOVE.L  A0,(A1)
    ADDA    #516,A0
    LEA     changed,A1
    MOVE.L  A0,(A1)

    MOVE.L  Board,A0        ; Mettons tout à
                            ; zéro
    MOVE    #5877,D0
@2
    CLR.L   (A0)+
    DBRA    D0,@2
    MOVE.L  Board,A0        ; Au départ, un
                            ; r-pentomino
    ADDA    #10271,A0       ; 160 * 64 + 31
    MOVE.B  #3,(A0)         ; au milieu de
                            ; l'écran
    MOVE.B  #6,64(A0)
    MOVE.B  #2,128(A0)
    MOVE.L  changed,A0      ; Ces trois lignes
    ST      160(A0)         ; sont non-vides
    ST      161(A0)
    ST      162(A0)
    RTS

;---------- Ouvrons les fenêtres ----------

CreateWindows
 Compteur:
  ; FUNCTION NewWindow (wStorage: Ptr;
  ; boundsRect: Rect;
  ; title: Str255; visible: BOOLEAN; procID:
  ; INTEGER;
  ; behind: WindowPtr; goAwayFlag: BOOLEAN;
  ; refCon: LongInt)
  ; : WindowPtr;
    CLR.L   -(SP)           ; Place pour le
                            ; résultat
    PEA                     ; WStorage2(A5)
    PEA                     ; counterRect
    PEA     'ww'
    MOVE.B  #1,-(SP)        ; Visible
    MOVE    #NoGrowDocProc,-(SP)
    MOVE.L  #1,-(SP)        ; Au sommet
    MOVE.B  #1,-(SP)        ; On veut pouvoir
                            ; la fermer
    CLR.L   -(SP)           ; RefCon = 0,
                            ; pourquoi pas ?
    _NewWindow              ; Dessinons la
                            ; fenêtre
    MOVE.L  (SP),window2    ; Sauvons son
                            ; adresse
    _SetPort                ; pointeur sur la
                            ; pile
    CLR     -(SP)           ; SysFont =
                            ; Chicago
    _TextFont
```

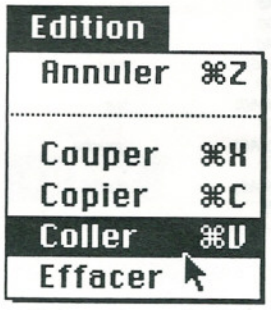


Source 'Vie2.Asm'

```
Tableau:
    CLR.L   -(SP)
    PEA     WStorage1(A5)
    PEA     windowRect      ; L'écran moins
                            ; les menus
    CLR.L   -(SP)           ; Pas de titre
    MOVE.B  #1,-(SP)        ; Visible
    MOVE    #PlainDBox,-(SP); Juste un
                            ; rectangle
    MOVE.L  #-1,-(SP)       ; Au sommet
    CLR     -(SP)           ; On ne la fermera
                            ; pas
    CLR.L   -(SP)
    _NewWindow
    MOVE.L  (SP),Window1
    _SetPort
    MOVE    #10,-(SP)       ; mode = patXor
    _PenMode

Rectangle:
    ; FUNCTION NewRgn : RgnHandle;
    CLR.L   -(SP)
    _NewRgn
    LEA     RectHandle,A0
    MOVE.L  (SP)+,(A0)

Grillage:
    MOVE.L  #21344,D0       ; 322 x 64 + 2 x
                            ; 46 x 8
    _NewPtr
    TST     D0
    BEQ.S   @1
    _ExitToShell
@1
    LEA     Grille,A1
    MOVE.L  A0,(A1)
    MOVE    #45,D1
@4
    MOVE    #63,D0
    MOVE    #-1,D2
@2
    MOVE.B  D2,(A0)+        ; une rangée
                            ; noire
    DBRA    D0,@2
    MOVE    #383,D0
    MOVEQ   #1,D2           ; et six rangées
                            ; de 00000001
@3
    MOVE.B  D2,(A0)+
    DBRA    D0,@3
    DBRA    D1,@4           ; 46 fois --> 322
                            ; rangées
    LEA     TamponVide,A1
    MOVE.L  A0,(A1)
    MOVE    #91,D0          ; Vidons-le
                            ; effectivement
@5
    CLR.L   (A0)+
    DBRA    D0,@5
    LEA     Tampon,A1
    MOVE.L  A0,(A1)

 Curseur:
    CLR.L   -(SP)           ; récupérons la
                            ; montre
    MOVE    #4,-(SP)        ; qui est dans le
                            ; fichier
    _GetCursor              ; System
    LEA     watchhandle,A0
    MOVE.L  (SP)+,(A0)

    RTS

;---------- Traitement des événements ----------

HandleEvent

    MOVE    Modify,ModifyReg
    MOVE    What,D0
    ADD     D0,D0
    MOVE    EventTable(D0),D0
    JMP     EventTable(D0)

EventTable

    DC.W    NextEvent-EventTable    ; Null Event
                                    ; (Inutilisé)
    DC.W    MouseDown-EventTable
    DC.W    NextEvent-EventTable    ; Mouse Up
                                    ; (Inutilisé)
    DC.W    KeyDown-EventTable
    DC.W    NextEvent-EventTable    ; Key Up
                                    ; (Inutilisé)
    DC.W    KeyDown-EventTable      ; Auto Key
    DC.W    Update-EventTable

; Normalement, la table est plus longue, mais nous ne
; faisons rien pour les activate/deactivate

;---------- Actions diverses ----------

Update
    MOVE.L  message,-(SP)
    _BeginUpDate
    MOVE.L  message,D0
    CMP.L   window1,D0
    BNE.S   @1
    BSR     Affiche
@1
    CMP.L   window2,D0
    BNE.S   @2
    BSR     AffCompte
@2
    MOVE.L  message,-(SP)
    _EndUpdate

NextEvent
    RTS                     ; retour à la
                            ; boucle

KeyDown
    BTST    #CmdKey,ModifyReg ; Seules les
                              ; commandes
    BEQ     NextEvent       ; nous
                            ; intéressent
    ; FUNCTION MenuKey (ch:CHAR): LongInt;
    CLR.L   -(SP)
    MOVE    Message+2,-(SP) ; Le caractère est
                            ; là
    _MenuKey                ; est-ce une
                            ; commande ?
    MOVE    (SP)+,MenuReg
    MOVE    (SP)+,MItemReg
    BRA     Choices         ; traitons-la

;---------- Pour Mouse Down c'est plus compliqué -------

MouseDown
    ; FUNCTION FindWindow (thePt: Point;
    ; VAR whichWindow: WindowPtr): INTEGER;
    CLR     -(SP)           ; place pour
```

```
                                ; résultat
    MOVE.L    Point,-(SP)       ; coordonnées de
                                ; la souris
    PEA       WWindow           ; Event Window
    _FindWindow
    MOVE      (SP)+,D0          ; numéro de
                                ; région
    ADD       D0,D0             ; *2 pour table
                                ; d'index
    MOVE      WindowTable(D0),D0
    JMP       WindowTable(D0)

WindowTable
    DC.W      NextEvent-WindowTable
    DC.W      InMenu-WindowTable
    DC.W      SystemEvent-WindowTable Window
    DC.W      Content-WindowTable
    DC.W      DoDrag-WindowTable
    DC.W      NextEvent-WindowTable
    DC.W      DoGoAway-WindowTable

SystemEvent
    ; PROCEDURE SystemClick (theEvent:
    ; EventRecord;
    ; theWindow: WindowPtr);
    PEA       EventRecord
    MOVE.L    WWindow,-(SP)
    _SystemClick
    BRA       NextEvent

Content
    MOVE.L    WWindow,D0
    CMP.L     window2,D0
    BNE.S     @1
    MOVE.L    window2,-(SP)     ; si c'est le
                                ; compteur
    _SelectWindow               ; sélectionner sa
                                ; fenêtre
    BRA       NextEvent
@1
    BTST      #runBit,statusReg
    BNE       NextEvent
    MOVE.L    Window1,-(SP)
    _SetPort
    PEA       Point
    _GlobalToLocal
    BTST      #resBit,statusReg
    BEQ       FatBits

DragRect:
    CLR       -(SP)
    MOVE.L    Point,-(SP)
    PEA       LoRect
    _PtInRect
    TST       (SP)+
    BEQ       NextEvent
    MOVE.L    rectHandle,-(SP)
    PEA       LoRect
    _RectRgn
    LEA       limites,A0
    MOVE.L    Point,(A0)
    MOVE.L    LoRect,-(SP)
    PEA       limites
    _SubPt
    LEA       limites,A0
    MOVE.L    (A0),4(A0)
    MOVE.L    #$011401C0,-(SP)  ; y = 276, x = 448
    PEA       limites+4
    _AddPt
    PEA       LoRect            ; préparation pour
                                ; Offset
; FUNCTION DragGrayRgn(theRgn : RgnHandle;
; startPoint : Point; limitRect,
; slopRect : Rect; axis : INTEGER; actionProc :
; ProcPtr) : LONGINT;
    CLR.L     -(SP)
    MOVE.L    RectHandle,-(SP)
    MOVE.L    Point,-(SP)       ; le coin en haut à
                                ; gauche
    PEA       limites
    PEA       HiRect
    CLR       -(SP)             ; pas de
                                ; contrainte
    CLR.L     -(SP)             ; ni d'actionProc

    _DragGrayRgn
    CMPI.L    #$80008000,(SP)   ; Renvoyé si le
                                ; bouton est
    BNE.S     @4                ; relaché hors
                                ; limites
    ADDA      #8,SP
    BRA       NextEvent
@4
    PEA       LoRect
    _FrameRect

    _OffsetRect
    PEA       LoRect
    _FrameRect
    BRA       NextEvent

FatBits:
    BCLR      #deterBit,statusReg ; couleur
                                  ; indeterminée
@3
    CLR.L     D0
    MOVE      yCoor,D0
    DIVU      #7,D0
    MOVE      D0,D2
    ADD       Top,D2
    MOVE.L    changed,A0
    ST        0(A0,D2)
    MULU      #512,D2
    MOVE.L    D2,A6
    ADDA      Left,A6
    MOVE      xCoor,D1

    ASR       #3,D1
    ADDA      D1,A6
    MULU      #7,D0
    LEA       pix,A0
    MOVE      D0,(A0)           ; haut
    ADDQ      #7,D0
    MOVE      D0,4(A0)          ; bas
    ASL       #3,D1
    MOVE      D1,2(A0)          ; gauche
    ADDI      #8,D1
    MOVE      D1,6(A0)          ; droite
    CLR       -(SP)
    MOVE.L    Board,-(SP)
    MOVE.L    A6,-(SP)
    _BitTst
    TST       (SP)+
    BNE.S     @7                ; le bit était noir

    BTST      #deterBit,statusReg
    BNE.S     @8                ; on sait déjà
    BSET      #deterBit,statusReg
    BSET      #colorBit,statusReg
    BRA.S     @4
@8
    BTST      #colorBit,statusReg
    BEQ.S     @10               ; blanc sur blanc
    BRA.S     @4

@7
    BTST      #deterBit,statusReg
    BNE.S     @9
    BSET      #deterBit,statusReg
    BCLR      #colorBit,statusReg
    BRA.S     @5
@9
    BTST      #colorBit,statusReg
    BNE.S     @10               ; noir sur noir
    BRA.S     @5

@4
    MOVE.L    Board,-(SP)
    MOVE.L    A6,-(SP)
    _BitSet
    BRA.S     @6

@5
    MOVE.L    Board,-(SP)
    MOVE.L    A6,-(SP)
    _BitClr

@6
    PEA       pix
    _InverRect

@10
    CLR       -(SP)
    _WaitMouseUp                ; le bouton a-t-il
                                ; été laché ?
    TST       (SP)+             ; Si oui, mise à
                                ; jour du
    BEQ       AffCompte         ; compteur et
                                ; retour
    PEA       Point
    _GetMouse
    CLR       -(SP)
    MOVE.L    Point,-(SP)
    PEA       HiRect
    _PtInRect
    TST       (SP)+
    BEQ.S     @10
    BRA       @3

DoDrag
    MOVE.L    WWindow,-(SP)
    MOVE.L    Point,-(SP)       ; coordonnées de
                                ; la souris
    PEA       windowRect        ; limites du
                                ; déplacement
    _DragWindow
    BTST      #CmdKey,ModifyReg ; cde ?
    BNE.S     @1
    MOVE.L    WWindow,-(SP)     ; non, activons
                                ; fenêtre
    _SelectWindow
@1
    BRA       NextEvent

DoGoAway
    ;FUNCTION TrackGoAway(theWindow :
    ; WindowPtr;
    ;startPoint : Point) : BOOLEAN
    CLR       -(SP)
    MOVE.L    WWindow,-(SP)
    PEA       Point
    _TrackGoAway
    TST       (SP)+             ; L'utilisateur
                                ; veut-il
    BEQ.S     @2                ; vraiment fermer
    MOVE.L    WWindow,-(SP)
    MOVE.L    Window1,-(SP)
    _SendBehind
    BCLR      #countBit,statusReg
@2
    BRA       NextEvent

; ---------- Articles de Menus ----------

InMenu
    ; FUNCTION MenuSelect  (startPt:Point) : LongInt;
    CLR.L     -(SP)
    MOVE.L    Point,-(SP)
    _MenuSelect
    MOVE      (SP)+,MenuReg
    MOVE      (SP)+,MItemReg
Choices
    CMP       #AppleMenu,MenuReg
    BEQ       InAppleMenu
    CMP       #FileMenu,MenuReg
    BEQ       InFileMenu
    CMP       #EditMenu,MenuReg
    BEQ       InEditMenu
    CMP       #CtlMenu,MenuReg
    BEQ       InCtlMenu

ChoiceReturn
    CLR       -(SP)
    _HiLiteMenu
    BRA       NextEvent

InAppleMenu
    CMP       #AboutItem,MItemReg
    BNE.S     Accessoire

    ;FUNCTION Alert(AlertID : INTEGER; Filter :
    ; ProcPtr) : INTEGER;
    CLR       -(SP)
    MOVE      #AboutAlert,-(SP)
    CLR.L     -(SP)
    _Alert
    ADDQ      #2,SP
    BRA       ChoiceReturn

Accessoire:
    ; PROCEDURE GetItem (menu: MenuHandle; item:
    ; INTEGER;
    ;           VAR itemString: Str255);
    MOVE.L    AppleHReg,-(SP)
    MOVE      MItemReg,-(SP)
    PEA       theString
    _GetItem
    ;FUNCTION OpenDeskAcc(theAcc : Str255) :
    ; INTEGER;
    CLR       -(SP)
    PEA       theString
    _OpenDeskAcc
    CLR       (SP)+             ; jetons le résultat
    BRA       ChoiceReturn


InFileMenu
    CMP       #NewItem,MItemReg
    BNE.S     Open
Effacе:
    MOVE.L    Board,A0          ; Effacer tout
    MOVE      #5877,D0
@1
    CLR.L     (A0)+

    DBRA      D0,@1
```

```
        CLR     statusReg
        BSR     Affiche
        BSR     AffCompte
        BRA     ChoiceReturn

Open:
        CMP     #OpenItem,MitemReg
        BNE     Save

    ; PROCEDURE SFGetFile(topLeft : Point;
    ; PromptString : Str255;
    ; fileFilter : ProcPtr; numTypes : INTEGER; typeList
    ; : SFTypeList;
    ; dialogHook : ProcPtr; VAR reply : SFReply);
    MOVE.L  #$00640052,-(SP)    ; x = 82, y = 100
    CLR.L   -(SP)
    CLR.L   -(SP)
    MOVE    #-1,-(SP)           ; tous les types
    CLR.L   -(SP)               ; pas de typeList
    CLR.L   -(SP)
    PEA     reply
    _SFGetFile
    LEA     Good,A0
    TST     (A0)
    BEQ     ChoiceReturn

    CLR.L   ParamBlock+12(A5)
    LEA     theString,A0
    MOVE.L  A0,ParamBlock+18(A5)
    MOVE    VrefNum,D0
    MOVE    D0,ParamBlock+22(A5)
    CLR     ParamBlock+26(A5)
    CLR.L   ParamBlock+28(A5)
    LEA     ParamBlock(A5),A0
    _Open
;Traitement d'erreur...
    MOVE.L  tampon,A1
    MOVE.L  A1,ParamBlock+32(A5)
    MOVE.L  #368,ParamBlock+36(A5)
    MOVE    #1,ParamBlock+44(A5)
    CLR.L   ParamBlock+46(A5)
    LEA     ParamBlock(A5),A0
    _Read
;Traitement d'erreur...
    LEA     ParamBlock(A5),A0
    _Close
    PEA     Tampon
    PEA     Board
    PEA     TampRect
    PEA     LoRect
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    MOVE.L  changed,A0
    ADDA    Top,A0
    MOVE    #45,D0
@6
    ST      (A0)+
    DBRA    D0,@6
    BSR     Affiche
    BSR     Affcompte
    BRA     ChoiceReturn

Save:
    PEA     Board
    PEA     Tampon
    PEA     LoRect
    PEA     TampRect
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    CMP     #SaveItem,MItemReg
    BNE     Quit
    ; PROCEDURE SFPutFile(topLeft : Point;
    ; PromptString : Str255;
    ; InitText : Str255; dialoghook : ProcPtr; VAR reply :
    ; SFReply);
    MOVE.L  #$00640068,-(SP)    ; x = 104, y = 100
    CLR.L   -(SP)
    MOVE    #prompt,-(SP)
    _GetString
    MOVE.L  (SP),A0             ; déréférençons
    MOVE.L  (A0),(SP)
    PEA     *
    CLR.L   -(SP)
    PEA                         reply
    _SFPutFile
    LEA     Good,A0
    TST     (A0)
    BEQ     ChoiceReturn

    CLR.L   ParamBlock+12(A5)
    LEA     theString,A0
    MOVE.L  A0,ParamBlock+18(A5)
    MOVE    VrefNum,D0
    MOVE    D0,ParamBlock+22(A5)
    CLR     ParamBlock+26(A5)
    LEA     ParamBlock(A5),A0
    _Create
    CLR.L   ParamBlock+28(A5)
    LEA     ParamBlock(A5),A0
    _Open
;Traitement d'erreur...
    MOVE.L  tampon,A1
    MOVE.L  A1,ParamBlock+32(A5)
    MOVE.L  #368,ParamBlock+36(A5)
    MOVE    #1,ParamBlock+44(A5)
    CLR.L   ParamBlock+46(A5)
    LEA     ParamBlock(A5),A0
    _Write
;Traitement d'erreur...
    LEA     ParamBlock(A5),A0
    _Close
    CLR.L   ParamBlock+18(A5)
    LEA     ParamBlock(A5),A0
    _FlushVol
    BRA     ChoiceReturn
```



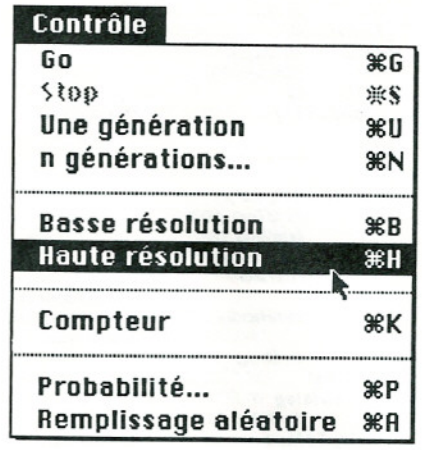
Source 'Vie3.Asm'

```
Quit:
    CMP     #QuitItem,MItemReg
    BNE     ChoiceReturn
    BSET    #quitBit,statusReg
    BRA     ChoiceReturn

InEditMenu
    CLR     -(SP)
    MOVE    MItemReg,-(SP)
    SUBQ    #1,(SP)             ; décalage pour
                                ; SystemEdit
    _SysEdit
    TST.B   (SP)+               ; A-t-il fait qqch ?
    BNE     ChoiceReturn
    CMP     #clearItem,MItemReg
    BNE.S   Copier

Effacer
    PEA     TamponVide          ; TamponVide -->
                                ; Rect
    PEA     Board
    PEA     TampVideRect
    PEA     LoRect
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    BSR     Affiche
    BSR     AffCompte
    BRA     ChoiceReturn

Copier
    CMP     #copyItem,MitemReg
    BNE.S   Coller
    PEA     Board
    PEA     Tampon
    PEA     LoRect
    PEA     TampRect
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    BRA     ChoiceReturn

Coller
    CMP     #pasteItem,MitemReg
    BNE     ChoiceReturn
    PEA     Tampon
    PEA     Board
    PEA     TampRect
    PEA     LoRect
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    BSR     Affiche
    BSR     AffCompte
    MOVE.L  changed,A0
    ADDA    Top,A0
    MOVE    #45,D0
@6
    ST      (A0)+
    DBRA    D0,@6
    BRA     ChoiceReturn

; ---------- Menu Contrôle ----------

InCtlMenu
    ADD     D6,D6
    MOVE    CtlMenuTable(D6),D6
    JMP     CtlMenuTable(D6)

CtlMenuTable
    DC.W    Dummy-CtlMenuTable ; pas d'item 0
    DC.W    DoGo-CtlMenuTable
    DC.W    DoStop-CtlMenuTable
    DC.W    DoStep-CtlMenuTable ; une
                                ; génération
    DC.W    DoNgen-CtlMenuTable ; n générations
    DC.W    Dummy-CtlMenuTable ; une ligne de
                                ; séparation
    DC.W    DoLo-CtlMenuTable  ; LoRes
    DC.W    DoHi-CtlMenuTable  ; HiRes
    DC.W    Dummy-CtlMenuTable ; une autre ligne
    DC.W    DoCounter-CtlMenuTable ; Montrer le
                                ; Compteur
    DC.W    Dummy-CtlMenuTable
    DC.W    DoProba-CtlMenuTable ; changer la
                                ; probabilité
    DC.W    DoAlea-CtlMenuTable ; Remplissage
                                ; aléatoire

DoGo
    BTST    #runBit,statusReg
    BNE     ChoiceReturn
    BSET    #runBit,statusReg
    MOVE.L  CtlHandle,-(SP)     ; handle vers le
                                ; menu
    MOVE.L  (SP),-(SP)          ; duplique
    MOVE.L  (SP),-(SP)          ; triplique
    MOVE.L  (SP),-(SP)          ; etc.
    MOVE    #GoItem,-(SP)
    _DisableItem
    MOVE    #StepItem,-(SP)
    _DisableItem
    MOVE    #StopItem,-(SP)
    _EnableItem
    MOVE    #nGenItem,-(SP)
    _DisableItem
Dummy
    BRA     ChoiceReturn

DoStop
    BTST    #runBit,statusReg
    BEQ     ChoiceReturn
    BCLR    #runBit,statusReg
    MOVE.L  CtlHandle,-(SP)
    MOVE.L  (SP),-(SP)
    MOVE.L  (SP),-(SP)
    MOVE.L  (SP),-(SP)
    MOVE    #GoItem,-(SP)
    _EnableItem
    MOVE    #StepItem,-(SP)
    _EnableItem
    MOVE    #StopItem,-(SP)
    _DisableItem
    MOVE    #nGenItem,-(SP)
    _EnableItem
    MOVE.L  Window1,-(SP)
    _SetPort
    BTST    #resBit,statusReg
    BEQ.S   @1
    PEA     LoRect
    _FrameRect
    BRA     ChoiceReturn
@1
    PEA     Grille
    MOVE.L  Window1,-(SP)
    ADDI.L  #2,(SP)
    PEA     HiRect
    PEA     HiRect
    MOVE    #2,-(SP)            ; mode srcXor
    CLR.L   -(SP)
    _CopyBits
    BRA     ChoiceReturn

DoStep
    BSR     Calcule
    BSR     Affiche
    BSR     AffCompte
    BRA     ChoiceReturn
```

```
DoNgen
   CLR.L    -(SP)
   MOVE     #GenDialog,-(SP)
   PEA      DStorage
   MOVE.L   #-1,-(SP)
   _GetNewDialog
   MOVE.L   (SP),-(SP)
   _SetPort
   CLR.L    -(SP)
   PEA      ItemHit
   _ModalDialog
   MOVE     ItemHit,D0
   CMP      #OKItem,D0
   BEQ.S    @1
   _CloseDialog                    ; Ce doit être
                                   ; annuler
   BRA      ChoiceReturn
@1
   MOVE.L   (SP),-(SP)
   MOVE     #editText,-(SP)
   PEA      itemType
   PEA      itemHandle
   PEA      dispRect
   _GetDItem                       ; récupération du
                                   ; nombre de
   MOVE.L   itemHandle,-(SP)       ; générations
                                   ; demandées
   PEA      theString
   _GetIText
   _CloseDialog
   LEA      theString,A0
   _StringToNum                    ; le nb est dans
                                   ; D0!
   TST      D0
   BEQ      ChoiceReturn           ; si nb = 0, ne rien
                                   ; faire
   SUBQ     #1,D0                  ; on va utiliser
                                   ; DBRA
   MOVE.L   D0,-(SP)               ; mettons le à
                                   ; l'abri
   MOVE.L   Window1,-(SP)          ; redessinons
                                   ; tout de suite
   _BeginUpDate                    ; le rectangle
                                   ; laissé en
   BSR      Affiche                ; blanc par
                                   ; CloseDialog
   MOVE.L   Window1,-(SP)
   _EndUpdate
   MOVE.L   watchHandle,A0         ; l'utilisateur va
                                   ; attendre
   MOVE.L   (A0),-(SP)             ; Déréferençons
   _SetCursor
   MOVE.L   (SP)+,D0               ; récupérons le
                                   ; nombre
@3
   BSR      Calcule
   DBRA     D0,@3
   _InitCursor
   BSR      Affiche
   BSR      AffCompte              ; actualisons le
                                   ; compteur
   BRA      ChoiceReturn

DoLo
   BTST     #resBit,statusReg
   BEQ      ChoiceReturn
   BCLR     #resBit,statusReg
   BSR      Affiche
   BRA      ChoiceReturn

DoHi
   BTST     #resBit,statusReg
   BNE      ChoiceReturn
   BSET     #resBit,statusReg
   BSR      Affiche
   BRA      ChoiceReturn

DoCounter
   MOVE.L   window2,-(SP)
   _SelectWindow
   BTST     #countBit,statusReg
   BNE      ChoiceReturn
   BSET     #countBit,statusReg
   BSR      AffCompte
   BRA      ChoiceReturn

DoProba
   CLR.L    -(SP)
   MOVE     #ProbaDialog,-(SP)
   PEA      DStorage
   MOVE.L   #-1,-(SP)
   _GetNewDialog
   MOVE.L   (SP),-(SP)
   _SetPort
   CLR.L    -(SP)
   PEA      ItemHit
   _ModalDialog
   MOVE     ItemHit,D0
   CMP      #OKItem,D0
   BEQ.S    @1
   _CloseDialog                    ; Ce doit être
                                   ; annuler
   BRA      ChoiceReturn
@1
   MOVE.L   (SP),-(SP)
   MOVE     #editText,-(SP)
   PEA      itemType
   PEA      itemHandle
   PEA      dispRect
   _GetDItem                       ; récupération du
   MOVE.L   itemHandle,-(SP)       ; pourcentage de
                                   ; points noirs
   PEA      theString
   _GetIText
   _CloseDialog
   LEA      theString,A0
   _StringToNum                    ; le % est dans
                                   ; D0!
   CMP      #100,D0
   BNE      @2                     ; 100 % est
                                   ; remplacé par
   MOVE     #255,D0                ; 255/256
   BRA.S    @3
@2
   MULU     #64,D0                 ; Pourcentage ->
                                   ; Octet
   DIVU     #25,D0
@3
   LEA      Proba,A0
   MOVE.B   D0,(A0)
   MOVE.L   Window1,-(SP)          ; redessinons
                                   ; tout de suite
   _BeginUpDate                    ; le rectangle
                                   ; laissé en
   BSR      Affiche                ; blanc par
                                   ; CloseDialog
   MOVE.L   Window1,-(SP)
   _EndUpdate                      ; et remplissons
                                   ; aussitôt.

; la probabilité est représentée sur un octet, soit huit
; chiffres binaires. Chaque bit du mot renvoyé par
; Random a une probabilité 1/2 d'être positionné. Un
; bit à 1 dans la proba est traduit par un OR et un bit à 0
; par un AND. C'est compliqué, mais
; mathématiquement correct. C'est plus rapide que de
; calculer bit par bit en faisant appel à Random tous les
; deux bits car toutes les manips se font au niveau du
; mot. D'autre part, avec la proba par défaut de 50%, il
; n'y a qu'un appel à Random pour 16 bits, ce que la
; méthode "naturelle" ne permettrait pas.

DoAlea
   MOVE.B   proba,D0               ; proba nulle :
                                   ; effacer
   BEQ      Effacer
   MOVE.L   watchHandle,A0         ; l'utilisateur va
                                   ; attendre
   MOVE.L   (A0),-(SP)             ; Déréferençons
   _SetCursor
   MOVEM.L  A4/D3-D5,-(SP)
   MOVE.B   proba,D3               ; proba sur 8 bits
   MOVEQ    #7,D4
@1
   ASR      #1,D3                  ; élimination des
                                   ; zéros
   DBCS     D4,@1                  ; à droite de proba
   MOVE.L   Board,A4
   MOVE     #maxWord,D5
@2
   CLR      -(SP)
   _Random
   MOVE     (SP)+,(A4)+
   DBRA     D5,@2
   BRA.S    @7
@3
   MOVE.L   Board,A4
   MOVE     #maxWord,D5
   ASR      #1,D3
   BCS      @4
@6
   CLR      -(SP)
   _Random
   MOVE     (SP)+,D0
   AND      D0,(A4)+
   DBRA     D5,@6
   BRA.S    @7
@4
   CLR      -(SP)
   _Random
   MOVE     (SP)+,D0
   OR       D0,(A4)+
   DBRA     D5,@4
@7
   DBRA     D4,@3
   MOVEM.L  (SP)+,D3-D5/A4
   CLR      StatusReg
   MOVE.L   changed,A0
   MOVE     #vmax,D0
@9
   ST       (A0)+
   DBRA     D0,@9
   _InitCursor
   BSR      Affiche
   BSR      AffCompte
   BRA      ChoiceReturn

; ---------- Affichage ----------

Affiche
   ;PROCEDURE CopyBits(FromB,toB : BitMap;
   ; from,to : Rect;
   ;
   ; mode : INTEGER; clipTo : RgnHandle);

   MOVE.L   Window1,-(SP)
   _SetPort
   BTST     #resBit,statusReg
   BEQ.S    @1
   PEA      Board
   MOVE.L   Window1,-(SP)          ; Offset 2 pour le
                                   ; bitmap
   ADDI.L   #2,(SP)                ; d'une fenêtre
   PEA      HiRect
   PEA      HiRect                 ; fenêtre entière
   CLR      -(SP)                  ; mode copy
   CLR.L    -(SP)                  ; pas de clipping
                                   ; supplémentaire
   _CopyBits
   BTST     #runBit,statusReg      ; le rectangle
                                   ; n'est dessiné
   BNE.S    @2                     ; qu'à l'arrêt
   PEA      LoRect
   _FrameRect
   BRA.S    @2
@1
   PEA      Board
   MOVE.L   Window1,-(SP)          ; Offset 2 pour le
                                   ; bitmap
   ADDI.L   #2,(SP)                ; d'une fenêtre
   PEA      LoRect                 ; une partie de la
                                   ; surface
   PEA      HiRect                 ; fenêtre entière
   CLR      -(SP)                  ; mode copy
   CLR.L    -(SP)                  ; pas de clipping
                                   ; supplémentaire
   _CopyBits
   BTST     #runBit,statusReg      ; la grille n'est
                                   ; dessinée
   BNE.S    @2                     ; qu'à l'arrêt
   PEA      Grille
   MOVE.L   Window1,-(SP)
   ADDI.L   #2,(SP)
   PEA      HiRect
   PEA      HiRect
   MOVE     #2,-(SP)               ; mode srcXor
   CLR.L    -(SP)
   _CopyBits
@2
   RTS

AffCompte
   BTST     #countBit,statusReg
   BEQ.S    @1
   MOVE.L   window2,-(SP)
   _SetPort
   PEA      cadran
   _EraseRect
   MOVE     #4,-(SP)               ; 1ère ligne du
                                   ; cadran
   MOVE     #13,-(SP)
   _MoveTo

; PROCEDURE NumToString(theNum : LONGINT;
; VAR theString : Str255);
; ATTENTION: Comme ne l'indique pas le livre de
; Chernikoff, NumToString utilise les registres A0 et
; D0 en entrée, laissant A0 intact en sortie

   CLR.L    D0
   MOVE     statusReg,D0           ; 16 bits de poids
                                   ; faible
   LEA      theString,A0
   _NumToString
```

```
        MOVE.L  A0,-(SP)
        _DrawString
        MOVE    #4,-(SP)  ; une ligne plus
                          ; bas
        MOVE    #25,-(SP)
        _MoveTo
        MOVE.L  D3,-(SP)  ; Sauvons un
                          ; registre
        CLR.L   D0        ; calcul de la
                          ; population
    MOVE.L  Board,A0
    MOVE    #maxLong,D1
@3
    TST.L   (A0)+
    BEQ.S   @2
    MOVE.L  -4(A0),D2
    MOVE    #31,D3
@5
    ASR.L   #1,D2
    BCC.S   @4
    ADDQ.L  #1,D0
@4
    DBRA    D3,@5
@2
    DBRA    D1,@3
    LEA     theString,A0
    _NumToString
    MOVE.L  A0,-(SP)
    _DrawString
    MOVE.L  (SP)+,D3          ; Récupération de
                              ; StatusReg
    MOVE.L  Window1,-(SP)
    _SetPort
@1
    RTS
```

```
; ---------------------- La Vie est ici ----------------------
; Chaque 'rangée' du tableau est parcourue à la
; recherche de bits à 1 (pixels 'vivants'). Pour chaque
; bit trouvé, on ajoute sa contribution aux 9 octets
; correspondant aux neuf cases du carré 3 x 3 dont le
; pixel est le centre, dans les 'lignes' précédente,
; présente et suivante respectivement. Cette
; contribution vaut 1 pour le centre et 2 pour tous les
; autres carrés. Ceci est le travail de la routine Calc.
; Quand une 'ligne' a été dans les trois positions, on
; peut calculer la valeur de la 'rangée' correspondante
; dans le tableau de la nouvelle génération. Comme la
; ligne était remplie au départ de -5, les pixels vivants
; correspondent aux cases contenant 0,1 ou 2, ce
; qu'un rapide calcul permet de vérifier. (L'intérêt de
; cette méthode tordue par rapport à une méthode
; plus 'naturelle' consistant par exemple à partir de 0,
; ajouter 1 ou 10 et à chercher les cases contenant 3,
; 12 ou 13 est de supprimer 5 ou 6 instructions à un
; endroit crucial : le gain de temps est de l'ordre de
; 30%! ) Ceci est fait par la routine Transfert. Il y a des
; cas particuliers pour les premières et dernières
; rangées, ainsi que pour les premiers et derniers bits
; de chaque rangée, parce que je voulais un terrain
; torique : un glisseur arrivant à un bord réapparait à
; l'autre bord au lieu de dégénérer en un misérable
; carré.
; NOTE : J'avais donné des noms symboliques aux
; registres, mais l'assembleur prétendait que les
; modes d'adressage complexes étaient illégaux. J'ai
; donc dû rendre aux registres leur nom d'origine, me
; contentant de donner ci-dessous leur signification.

base    EQU   A1   ; pointe vers le début d'une
                   ; rangée
pred    EQU   A2   ; 'ligne' précédente
act     EQU   A3   ; 'ligne' présente
suiv    EQU   A4   ; 'ligne' suivante
change  EQU   A0   ; pointe vers le booléen qui
                   ; indique si une rangée est non
                   ; vide
temp    EQU   D0   ; stockage temporaire
v       EQU   D2   ; coordonnée verticale (à
                   ; l'envers...)
hh      EQU   D3   ; offset du mot long étudié (croit
                   ; de g à d)
hl      EQU   D4   ; offset du bit (décroit de gauche
                   ; à droite)
h       EQU   D5   ; coordonnée horizontale (croit
                   ; de g à d)
bloc    EQU   D6   ; le mot long parcouru ou
                   ; construit
ch      EQU   D7   ; la nouvelle rangée a-t-elle un '
                   ; pixel vivant
```



Source 'Vie4.Asm'

```
Calc    ; pred, act, suiv et base doivent être
        ; corrects en entrée
        ; Calc ne les modifie pas (A1-A4), mais
        ; modifie D3-D6
    CLR     D3
    MOVE    #31,D5        ; en cas de
                          ; branchement
                          ; juste après
    MOVE.L  (A1),D6       ; premier mot long
    BEQ.S   @1            ; s'il est nul, mot
                          ; suivant
    BPL.S   @2            ; si le 1° bit est
                          ; nul, bit suivant
    ADDQ.B  #2,511(A2)    ; effet sur la
                          ; dernière case
    ADDQ.B  #2,511(A3)    ; des lignes
    ADDQ.B  #2,511(A4)
    ADDQ.B  #2,(A2)       ; et sur les
                          ; première
    ADDQ.B  #1,(A3)
    ADDQ.B  #2,(A4)
    ADDQ.B  #2,1(A2)      ; et secondes
                          ; cases
    ADDQ.B  #2,1(A3)
    ADDQ.B  #2,1(A4)
@2
    CLR     D5            ; modifions l'octet
                          ; le plus à gauche
    MOVE    #30,D4        ; le bit suivant est
                          ; le #30
@4
    BTST    D4,D6
    BEQ.S   @5            ; branchement si
                          ; bit à 0
    ADDQ.B  #2,0(A2,D5)
    ADDQ.B  #2,0(A3,D5)
    ADDQ.B  #2,0(A4,D5)
    ADDQ.B  #2,1(A2,D5)
    ADDQ.B  #1,1(A3,D5)
    ADDQ.B  #2,1(A4,D5)
    ADDQ.B  #2,2(A2,D5)
    ADDQ.B  #2,2(A3,D5)
    ADDQ.B  #2,2(A4,D5)
@5
    ADDQ    #1,D5         ; incrémenter h et
                          ; décrémenter
    DBRA    D4,@4         ; le n° du bit,
                          ; encore si >=0
@1
    MOVE    #31,D4
@6
    ADDQ    #4,D3
    CMP     #60,D3
    BEQ.S   @7
    MOVE.L  0(A1,D3),D6
    BNE.S   @4
    ADDI    #32,D5
    BRA.S   @6
@7
    MOVE.L  0(A1,D3),D6
    BEQ.S   @8
@9
    BTST    D4,D6
    BEQ.S   @10
    ADDQ.B  #2,0(A2,D5)
    ADDQ.B  #2,0(A3,D5)
    ADDQ.B  #2,0(A4,D5)
    ADDQ.B  #2,1(A2,D5)
    ADDQ.B  #1,1(A3,D5)
    ADDQ.B  #2,1(A4,D5)
    ADDQ.B  #2,(A2)       ; ce pixel de
                          ; droite a des
    ADDQ.B  #2,(A3)       ; effets aussi sur
                          ; l'octet
    ADDQ.B  #2,(A4)       ; le plus à gauche
@8
    SF      512(A2)       ; les lignes sont
                          ; 'sales'
    SF      512(A3)
    SF      512(A4)
    RTS                   ; Ouf!

Transfert
        ; la ligne vers laquelle pred=A2 pointe
        ; est parcourue. A chaque 0, 1 ou 2
        ; trouvé, on aura un pixel 'vivant' à la
        ; génération suivante. On met à un le bit
        ;correspondant. La rangée ainsi formée
        ; est écrite sur la rangée précédant celle
        ; vers laquelle base pointe. En sortie,
        ; ch=D7 vaut $FF si on a trouvé au
        ; moins un point vivant, à 0 sinon.
    MOVE    #-64,D3       ; offset de la
                          ; rangée
    CLR     D7            ; flag à zéro
    CLR     D5            ; premier élément
                          ; de la ligne
@1
    MOVE    #31,D4        ; bit de poids fort
                          ; (pixel gauche)
    CLR.L   D6
@5
    MOVE.B  0(A2,D5),D0   ; lisons la valeur
    BLT.S   @3
    CMP.B   #2,D0
    BGT.S   @3
    BSET    D4,D6         ; vivant si 0, 1 or 2
@3
    ADDQ    #1,D5
    DBRA    D4,@5         ; mot .L terminé ?
    MOVE.L  D6,0(A1,D3)   ; oui, écrivons-le
    BEQ.S   @6
    ST      D7            ; la rangée n'est
                          ; pas vide
@6
    ADDQ    #4,D3         ; fin de rangée
                          ; atteinte ?
    BNE.S   @1            ; non, mot suivant
    RTS

Calcule
    MOVEM.L A0-A4/D0-D7,-(SP) ; sauvons les
                              ; registres
    MOVE.L  moins2,A0
    MOVE    #644,D0
@1
    MOVE.L  #$FBFBFBFB,(A0)+ ; cinq lignes à
                             ; nettoyer
    DBRA    D0,@1
    MOVE.L  changed,change    ; change EQU A0
    TST.B   vMax(A0)          ; dernière rangée
                              ; vide ?
    BEQ.S   @2                ; oui, passons
    MOVE.L  moins2,A2
    MOVE.L  moins1,A3
    MOVE.L  pas0,A4
    MOVE.L  Board,A1          ; adresse de base
                              ; de la
    ADDA    #20544,A1         ; dernière rangée
    BSR     Calc
@2
    MOVE.L  Board,A1
    TST.B   (A0)              ; première rangée
                              ; blanche ?
    BEQ.S   @3
    MOVE.L  moins1,A2
    MOVE.L  pas0,A3
    MOVE.L  pas1,A4
    BSR     Calc
```

```
@3
   MOVE.L   pas0,A2
   MOVE.L   pas1,A3
   MOVE.L   pas2,A4
   MOVE     #vMax-4,v          ; 318 rangées
                               ; semblables
@8
   ADDA     #64,A1
   ADDQ.L   #1,A0
   TST.B    512(A4)            ; la ligne suivante
                               ; a-t-elle
   BNE.S    @4                 ; été utilisée ?
   MOVE     #128,D0            ; oui, nettoyons-la
   ADDA     #516,A4
@5
   MOVE.L   #$FBFBFBFB,-(A4)
   DBRA     D0,@5
@4
   TST.B    (A0)               ; rangée blanch ?
   BEQ.S    @6                 ; oui, sautons-la
   BSR      Calc               ; non, cacul de sa
                               ; contribution
@6
   TST.B    512(A2)            ; a-t-on écrit dans
                               ; la ligne
   BNE.S    @7                 ; précédente ?
   BSR      Transfert          ; oui, calcul de la
                               ; rangée
   MOVE.B   ch,-1(A0)          ; qui est vide ou
                               ; non
@7
   MOVE.L   A2,-(SP)           ; rotation entre les
                               ; lignes
   MOVE.L   A3,A2
   MOVE.L   A4,A3
   MOVE.L   (SP)+,A4
   DBRA     v,@8               ; une autre
                               ; rangée

Encore_deux                    ; une étiquette
                               ; globale ...
   MOVE.L   moins2,A4
   ADDA     #64,A1
   ADDQ.L   #1,A0
   TST.B    (A0)
   BEQ.S    @6
   BSR      Calc
@6 TST.B
   512(A2)
   BNE.S    @7
   BSR      Transfert
   MOVE.B   ch,-1(A0)
@7
   MOVE.L   A3,A2
   MOVE.L   A4,A3
   MOVE.L   moins1,A4
   ADDA     #64,A1
   ADDQ.L   #1,A0
   TST.B    (A0)
   BEQ.S    @8
   BSR      Calc
@8
   TST.B    512(A2)
   BNE.S    @9
   BSR      Transfert
   MOVE.B   ch,-1(A0)
@9
   MOVE.L   A3,A2
   MOVE.L   A4,A3
   ADDA     #64,A1
   ADDQ.L   #1,A0
   TST.B    512(A2)
   BNE.S    @10
   BSR      Transfert
   MOVE.B   ch,-1(A0)
@10
   MOVE.L   A3,A2
   ADDA     #64,A1
   ADDQ.L   #1,A0
   TST.B    512(A2)
   BNE.S    @11
   BSR      Transfert
   MOVE.B   ch,-1(A0)
@11
   MOVEM.L  (SP)+,A0-A4/D0-D7
   ADDQ     #1,statusReg
   RTS

; ---------- Données relogeables ----------

EventRecord                    ; Renvoyé par
                               ; GetNextEvent
   What:    DC     0          ; Nature de
                               ; l'événement
   Message: DC.L   0          ; précisions diverses
   When:    DC.L   0          ; nb de Ticks
Point:                         ; coordonnées de la
                               ; souris
   yCoor:   DC     0
   xCoor:   DC     0
   Modify:  DC     0          ; État de commande,
                               ; option...

WWindow     DC.L   0          ; Résultat de
                               ; FindWindow

CtlHandle   DC.L   0
DStorage    DCB.W  DWindLen,0  ; Dialogues

reply
   good:    DC     0
   fType:   DC.L   0
   vRefNum: DC     0
   version  DC     0
   theString: DCB.W 64,0      ; diverses chaines
                               ; y compris fName de
                               ; SFReply

WindowRect  DC     20,0,342,512
                               ; Tableau de jeu
counterRect DC     48,420,76,480
                               ; compteur

cadran      DC     0,0,28,60
```

```
ItemHit     DC     0
ItemType    DC     0
ItemHandle  DC.L   0
dispRect    DC     0,0,0,0

Board       DC.L   0          ; bitmap (tableau de
                              ; jeu)
 rowbytes:  DC     64         ; comme screenbits
 HiRect:    DC     0,0,322,512
                              ; comme portRect

limites     DC     0,0,276,448
                              ; pour dragGrayRgn

Grille      DC.L   0          ; un autre bitmap
            DC     64
            DC     0,0,322,512

Tampon      DC.L   0          ; un troisième
            DC     8
   TampRect: DC    0,0,46,64

   TamponVide:     DC.L   0
                              ; sert à effacer
            DC     8
   TampVideRect:   DC     0,0,46,64

LoRect      ; 46 x 64 centré (au début)
   Top:     DC     138
   Left:    DC     224
   Bottom:  DC     184
   Right:   DC     288

Proba       DC.B   128        ; proba 1/2 au début

Pix         DC     0,0,0,0    ; gros 'pixel'
                              ; rectangulaire

RectHandle  DC.L   0          ; Région rectangulaire
watchHandle DC.L   0          ; curseur montre

moins2      DC.L   0          ; pointeurs de ligne
moins1      DC.L   0
pas0        DC.L   0
pas1        DC.L   0
pas2        DC.L   0
changed     DC.L   0          ; pointe vers un tableau
                              ; de booléens

;---------- Données non relogeables ----------

WStorage1   DS.W   WindowSize
                              ; Fenêtre principale
WStorage2   DS.W   WindowSize
                              ; Fenêtre du compteur
ParamBlock  DS.W   50         ; Pour les
                              ; entrées/sorties
```

Fichier 'Vie.R'

```
Jeu de la Vie
APPLJDLV

INCLUDE Vie.Code

Type MENU
 ,1
\14
 C'est la vie...
 (-

 ,2
Fichier
 Nouveau
 Ouvrir...
 Enregistrer sous...
 (-
 Quitter/Q

 ,3
Edition
 Annuler/Z
 (-
 Couper/X
 Copier/C
 Coller/V
 Effacer

 ,4
Contrôle
 Go/G
 (Stop/S
 Une génération/U
 n générations.../N
 (-
 Basse résolution/B
 Haute résolution/H
 (-
 Compteur/K
 (-
 Probabilité.../P
 Remplissage aléatoire/A

Type ALRT (32)
 ,1
100 50 210 450
1
4444

Type DITL
 ,1
4
Button
80 230 100 290
OK

StaticText Disabled
15 20 36 300
Jeu inventé par J. Conway

StaticText Disabled
35 20 56 380
© Dominique Bernardi

StaticText Disabled
55 20 76 300
et Pom's

Type DLOG
 ,2

100 100 170 400
Visible NoGoAway
1
0
2

Type DITL
 ,2
4
Button
40 40 60 120
OK

Button
40 180 60 260
Annuler

StaticText Disabled
15 30 36 200
Nombre de générations:

EditText Disabled
16 220 31 270
25

Type DLOG
 ,3

100 100 170 400
Visible NoGoAway
1
0
3
Type DITL
 ,3
4

Button
40 40 60 120
OK

Button
40 180 60 260
Annuler

StaticText Disabled
15 30 36 200
Pourcentage de vivants:

EditText Disabled
16 220 31 270
50

Type STR
 ,1
Nom du nouveau fichier:
```

ROUTINE DE SAISIE

En passant de l'Apple // au Macintosh, le Basic de Microsoft a conservé les caractéristiques de son instruction standard de saisie au clavier ; de fait, l'instruction INPUT n'est pas plus performante aujourd'hui qu'elle ne l'était jadis.

Certes, les "évolutions Mac", du type EDIT FIELD ou autres EDIT$, apportent un plus incontestable, mais ne permettent pas de contrôler la saisie en fonction du type de données attendu. Par ailleurs, elles ne se prêtent guère aux opérations de saisie "intensive", car toute correction de frappe impose une utilisation alternative de la souris et du clavier...

Dans ses précédents numéros, Pom's vous a souvent proposé des solutions aux problèmes de saisie en Basic, sous forme de routines en assembleur ou en Applesoft, à des niveaux de complexité variés. Dans ce même esprit, nous vous présentons ici un exemple de routine Basic que vous pourriez intégrer dans vos programmes dès lors que la frappe y joue un rôle important.

Principes d'utilisation

Les données sont saisies par sous-ensembles correspondant à des tableaux de variables (par exemple, un tableau pour toutes les informations à saisir en un même écran). L'affichage doit toujours se faire avec une police non proportionnelle afin de permettre des déplacements corrects sur une ligne de saisie (voir plus loin).
Chacune des données est en outre identifiée par :

- un libellé précisant le contenu de l'information à fournir et les coordonnées horizontale et verticale du point à partir duquel doit commencer l'affichage du libellé (ces coordonnées sont exploitées par une instruction MOVETO) ;
- la position du point à partir duquel débutera la saisie (coordonnées horizontale et verticale) ;
- le nombre maximum de caractères autorisé pour la saisie de cette donnée ;
- le type de la donnée.

Ce dernier paramètre peut prendre les valeurs suivantes :

- 1 pour indiquer une donnée alphanumérique. La routine acceptera alors la frappe de tout caractère de code ASCII supérieur ou égal à 32, les guillemets, la virgule, les deux-points et le point-virgule étant toutefois remplacés par un point, afin d'éviter les problèmes d'exploitation ultérieure des informations entrées.
- 2 pour une donnée numérique. Ne sont alors acceptés que les chiffres, les signes "+" et "-", et le point décimal (la frappe d'une virgule est aussitôt transformée en point).
- 3 pour indiquer une date. Ne sont alors acceptés que les chiffres et la barre de fraction "/". La date doit normalement être saisie sous la forme JJ/MM/AA mais, si vous rentrez 1/1/86, par exemple, la routine complètera elle-même pour obtenir 01/01/86. De plus, un contrôle est effectué sur la valeur de JJ (inférieure à 32) et sur celle de MM (inférieure à 13).

La routine utilise quatre touches pour permettre les déplacements sur une ligne de saisie ou d'une ligne à l'autre à l'intérieur d'un tableau.

- RETURN permet de passer à la ligne suivante ou à la suite du programme s'il s'agit de la dernière ligne. En outre, cette touche valide la saisie en affectant à une chaîne de caractères tous les caractères qui se trouvent à la gauche de l'endroit où est entré RETURN (si vous tapez RETURN à l'intérieur d'une ligne de saisie, vous annulez la partie droite de la frappe).
Par ailleurs, si RETURN est la seule touche que vous frappez en tête de ligne, vous passez à la suivante tout en validant sans le modifier le contenu de la ligne. Cela vous sera notamment utile si vous vous déplacez à l'intérieur du tableau pour le modifier.
- Lorsque vous êtes au début d'une ligne (sauf la première..) et que vous n'avez encore entré aucun caractère, la frappe de ENTER (la touche qui se trouve juste à droite de la barre d'espace) vous permet de retourner au début de la ligne précédente. Ceci reste valable même lorsque des caractères sont affichés sur la ligne (modification d'une saisie antérieure), car c'est l'absence de frappe et non l'absence d'affichage qui est testée.
- Sur une ligne, la frappe de "Backspace" (en haut et à droite du clavier) permet de reculer le curseur (matérialisé par un signe _) d'un caractère. Les caractères situés alors à droite du curseur ne sont pas perdus, même si vous modifiez l'un de ceux qui les précèdent, dès lors que vous ramenez le curseur à l'extrémité de la ligne, au moyen de la touche décrite ci-dessous, avant de taper RETURN.
- Toujours sur une ligne, la frappe de la touche de tabulation permet de déplacer le curseur vers la droite et de restituer les caractères sur lesquels vous étiez préalablement passé avec la touche Backspace.

Principales variables utilisées par la routine

nd% : nombre de données

Gérard Michel

(lignes) à saisir.
zc$(i) : ce tableau contient les données saisies en sortie de la routine. Chaque élément du tableau correspond à une ligne.
tt% : si cette variable vaut 0, elle indique une première saisie. Dans le cas contraire, il s'agit de modifier un tableau précédemment entré. Les valeurs initiales doivent alors être basculées dans zc$(i) avant l'appel de la routine.
li$(i) : contient les libellés à mettre en regard des données à saisir.
lh%(i) et **lv%(i)** contiennent les coordonnées horizontales et verticales des libellés.
zh%(i) et **zv%(i)** : coordonnées horizontales et verticales des points de départ de chaque ligne de saisie.
ll%(i) : longueurs maximales des données.
ty%(i) : types des données.
pc% : position du curseur sur la ligne (vaut 1 pour le début de la ligne).
l% : nombre de caractères entrés.
te% : vaut 0 si aucun caractère n'a été frappé.
dd% : ce paramètre est fixé à 7. Il correspond à la valeur qu'il faut ajouter ou retrancher à la coordonnée horizontale d'un MOVETO pour obtenir un déplacement équivalent à la place occupée sur l'écran par un caractère en police Monaco 9 points.
d% : indique la coordonnée horizontale du prochain caractère à saisir.
z$: reçoit le caractère frappé.
**c$** : chaîne tampon "archivant" les caractères avant leur validation par RETURN et leur affectation à zc$(i).
k$: contient le caractère sous le curseur et les caractères qui se trouvent à sa droite, en cas de retour en arrière.

Programme de démonstration

Il permet de saisir deux tableaux, l'un de 4 variables et l'autre de 3. Chaque tableau saisi est alors affiché pour contrôle.
En mettant à 1 la valeur de tt%, vous pouvez ensuite modifier votre première saisie pour ces 2 tableaux.

Cet exemple très "dépouillé" vous aidera simplement à analyser la façon dont vous pourrez appeler la routine de saisie dans vos propres programmes.

Programme de démonstration

```
' Initialisation et appels de la routine
OPTION BASE 1:tt%=0:nl%=1:DIM sv$(2,4
    )
' Définition des libellés du premier tab
    leau (premier écran)
CLS:DATA 4,LIBELLE 1,LIBELLE 2,LIBELLE
     3,LIBELLE 4,50,50,50,80,50,110,50,1
    40
' Paramètres des données du premier tabl
    eau
DATA 130,50,15,1,130,80,6,2,130,110,8,3
    ,130,140,50,1
' Définition des libellés du second tabl
    eau
DATA 3,SUITE 1,SUITE 2,SUITE 3,100,100,
    100,130,100,160
' Paramètres des données du second table
    au
DATA 180,100,20,1,180,130,10,2,180,160,
    8,3
' Début du programme de démonstration
' Saisie des deux tableaux, modification
     des mêmes tableaux et fin du progra
    mme
depart:
IF nl%>2 THEN tt%=tt%+1:RESTORE:nl%=1
    : ' Préparation de la modification
IF tt%>1 THEN END
GOSUB lecture:CLS:IF tt%=1 THEN FOR i
    =1 TO nd%:zc$(i)=sv$(nl%,i):NEXT
GOSUB saisie:CLS:FOR i=1 TO nd%:MOVET
    O 10,50+30*(i-1):PRINT li$(i)" : "z
    c$(i):NEXT:PRINT:PRINT:INPUT z$
FOR i=1 TO nd%:sv$(nl%,i)=zc$(i):NEXT:
    ' sauvegarde des données saisies
nl%=nl%+1:GOTO depart
' Lecture des libellés et des paramètres
lecture:
READ nd%:FOR i=1 TO nd%:READ li$(i):N
    EXT:FOR i=1 TO nd%:READ lh%(i),lv%
    (i):NEXT
FOR i=1 TO nd%:READ zh%(i),zv%(i),ll%(
    i),ty%(i):NEXT
RETURN
'
' Routine de saisie
'
saisie:
' Police non proportionnelle en 9 points
TEXTFONT 4:TEXTSIZE 9:TEXTMODE 0
```

```
IF tt%=0 THEN FOR i=1 TO nd%:z
    c$(i)="":NEXT:' tt%=0 => sa
    isie
FOR i=1 TO nd%:MOVETO lh%(i),l
    v%(i):PRINT li$(i)" : ";:M
    OVETO zh%(i),zv%(i):PRINT
    zc$(i);:NEXT
FOR i=1 TO nd%: ' Boucle de saisie des
    variables
debut:
' Initialisation des variables pour la s
    aisie
d%=zh%(i):zv%=zv%(i):MOVETO d%,zv%:dd%=
    7:k$=zc$(i):c$=zc$(i):l%=LEN(c$):pc
    %=1:ll%=ll%(i):ty%=ty%(i):te%=0
' Affichage du curseur et attente d'un c
    aractère au clavier
100 PRINT "_";:MOVETO d%-dd%,zv%:PRIN
    T k$;:MOVETO d%-dd%,zv%:z$=INKEY$
    :IF z$="" THEN 100
c%=ASC(z$):IF c%=13 AND te%=0 THEN PR
    INT c$;:GOTO fin
IF c%=3 AND te%=0 THEN IF i>1 THEN i=
    i-1:MOVETO d%-dd%,zv%:PRINT c$" ";
    :GOTO debut ELSE BEEP:GOTO 100
te%=1: ' Un caractère différent de RETUR
    N (13) ou ENTER (3) est frappé
' c%=8 => en arrière
tus: processing jo
IF c%=8 AND pc%=1 THEN BEEP:GOTO 100
IF c%=13 THEN c$=LEFT$(c$,pc%-1):MOVE
    TO d%-dd%,zv%:PRINT SPACE$(ll%-pc
    %+2);:GOTO fin
IF c%=9 AND pc%=l%+1 THEN BEEP:GOTO 1
    00
IF c%=9 THEN PRINT SPACE$(ll%-pc%+1);
    :MOVETO d%-dd%,zv%:PRINT MID$(c$,
    pc%,1);:GOTO 300
IF c%=8 THEN 200
IF pc%=ll%+1 THEN BEEP:GOTO 100
IF c%=34 OR c%=44 OR c%=58 OR c%=59 TH
    EN z$=".":c%=46
ON ty% GOTO alpha,num,date
alpha:
IF c%<32 THEN BEEP:GOTO 100 ELSE suit
    e
num:
IF c%=46 OR c%=45 OR c%=43 THEN suite
IF c%>57 OR c%<48 THEN BEEP:GOTO 100
    ELSE suite
date:
IF c%<47 OR c%>57 THEN BEEP:GOTO 100
suite:
IF k$<>"" THEN PRINT SPACE$(ll%-pc%);
k$=" ":MOVETO d%-dd%,zv%:PRINT k$;:MO
    VETO d%-dd%,zv%:PRINT z$;:d%=d%+dd
    %:pc%=pc%+1:IF pc%=l%+2 THEN l%=l%+
    1:c$=c$+z$:k$="":GOTO 110
MID$(c$,pc%-1,1)=z$:k$=MID$(c$,pc%,1)
110 MOVETO d%-dd%,zv%:GOTO 100
200 pc%=pc%-1:IF pc%<l% THEN MOVETO d%
    -dd%,zv%:k$=RIGHT$(c$,l%-pc%+1):PR
    INT k$;:GOTO 210
k$=MID$(c$,pc%,1)
210 MOVETO d%-2*dd%,zv%:d%=d%-dd%:GOTO
    100
300 pc%=pc%+1:k$=MID$(c$,pc%,1):d%=d%+d
    d%:GOTO 110
fin:
IF ty%<>3  OR c$="" THEN bc1
' Contrôle de conformité de la date sais
    ie
FOR j=1 TO 3:IF MID$(c$,j,1)<>"/" THE
    N d0
IF j<3 THEN c$="0"+c$:j=3
d0:
NEXT:IF MID$(c$,3,1)<>"/" OR VAL(LEF
    T$(c$,2))<1 OR VAL(LEFT$(c$,2))>31
    THEN BEEP:zc$(i)=c$:GOTO debut
FOR j=4 TO 6:IF MID$(c$,j,1)<>"/" THE
    N d1
IF j<6 THEN c$=LEFT$(c$,3)+"0"+MID$(c
    $,4):j=6
d1:
NEXT:IF MID$(c$,6,1)<>"/" OR VAL(MID
    $(c$,4,2))<1 OR VAL(MID$(c$,4,2))>
    12 OR LEN(c$)<>8 OR MID$(c$,7,1)="
    /" OR RIGHT$(c$,1)="/" THEN BEEP:
    zc$(i)=c$:GOTO debut
bc1:
' Affectation de la variable et suite...
zc$(i)=c$:IF c$="" THEN MOVETO d%-dd%,
    zv%:PRINT "  ";
NEXT:RETURN
```

A propos de '•-> Corbeille'

Avec le nouveau système (disquette 22), la fenêtre inférieure de cet accessoire (Pom's 21) est partiellement masquée par la fenêtre de sélection des fichiers, plus grande qu'avec les versions antérieure du système. Ceci n'empêche pas ce 'mini-programme' de fonctionner mais fait un peu 'désordre'. Pour que tout rentre dans l'ordre, il faut remplacer la ligne :

`MOVE.L #$00280014,-(SP)` par `MOVE.L #$00200014,-(SP)`

et la ligne :

`RectangleF DC 198,20,281,368` par `RectangleF DC 252,20,335,369`

Astuces

Gaëtan Dagron
Delphine Declercq
Jacques Honorez
Niels Køge
Hervé Thiriez

une rubrique ouverte aux lecteurs

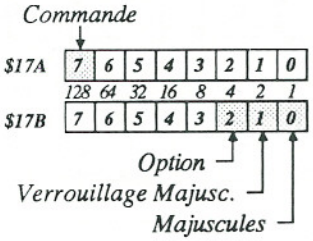
Pour la souris

Qui n'a pas sous la main une vieille radiographie ? Ce support anti-statique semble être l'idéal pour les évolution de la souris. Moins de risques de voir l'animal se salir !

Commande, Option, Majuscules ?

Pour savoir quel est l'état de ces touches, ainsi que 'verrouillage majuscules' depuis un programme Basic, il faut lire le contenu des adresses $17A et $17B :

- le bit 7 de $17A est à 1 si la touche 'Commande' est enfoncée ;
- le bit 0 de $17B est à 1 si l'une des touches 'Majuscules' est enfoncée ;
- le bit 1 de $17B est à 1 si la touche 'Verrouillage majuscules' est enfoncée ;
- le bit 2 de $17B est à 1 si l'une des touches 'Option' est enfoncée.



Dans un programme Basic, on pourrait donc trouver des lignes telles que :

```
IF PEEK(&h17A) AND
  128 THEN Commande
IF PEEK(&h17B) AND 1
  THEN Majuscule
IF PEEK(&h17B) AND 2
  THEN Verrouillage
IF PEEK(&h17B) AND 4
  THEN Option
```

ou encore, si l'on veut savoir si les touches 'Option' et 'Majuscule' sont enfoncées simultanément :

```
IF PEEK(&h17B) AND 5
  THEN Option.Maj.
```

N.D.L.R. : si vous programmez en assembleur, ces deux adresses correspondent à `KeyMap+6` et `KeyMap+7`.

LaserAstuces

Pour éviter de gâcher une feuille de papier et de l'encre avec l'auto-test à l'allumage de l'appareil, il suffit de retirer l'alimentation papier au moment de l'allumage et de la remettre lorsque le témoin vert ne clignote plus.

Titres encadrés avec MacPaint

Mettre un titre dans une zone comportant un décor n'est pas une sinécure avec MacPaint. Une première solution revient à entrer le texte une fois le décor créé. Le détourage du texte ainsi obtenu peut être jugé trop important :

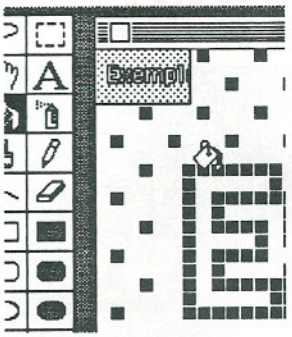
La seconde solution, bien moins lisible, consiste à taper le texte en dehors du cadre puis à l'y placer avec le lasso :

La procédure indiquée ci-dessous permet de construire, certes au prix de quelques manipulations, un détourage à la fois lisible et moins brutal que le détourage rectangulaire présenté en premier :

Première étape : activer la grille, si possible avant de dessiner le cadre. Cela nous permettra de meilleurs centrages.

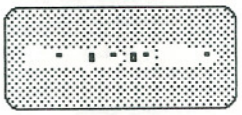
Seconde étape : taper le titre en dessous du cadre, le sélectionner, le copier, le coller (préserve l'original qui servira plus tard), demander une fois le contour et le placer avec le lasso dans le cadre.

Troisième étape : activer la loupe, choisir la peinture blanche, et verser la peinture sur le bord extérieur du texte :



L'utilisation de la loupe nous permet de verser la peinture, exactement sur le trait extérieur, sans difficulté.

Quatrième étape : se déplacer avec la main pour continuer la même opération de blanchissage (2 fois ici) sur le reste du texte, qui n'est pas forcément jointif. Le résultat est alors :



Cinquième étape : il ne reste plus qu'à y placer avec le lasso le texte qui était resté au naturel sous le cadre, ce qui nous donne le résultat final présenté plus haut.

Modifier «System»

On ne peut normalement modifier le nom du fichier 'System'. Pour y parvenir, dupliquez-le et modifiez alors si vous le voulez le nom de 'Copie de System' qui sera, lui, modifiable jusqu'à sa première utilisation sous le nom de 'System'.

L'intérêt de faire porter à un fichier 'System' un nom différent, c'est que cela vous permet de lancer un programme placé sur une disquette comportant son propre 'System' (rebaptisé) à partir d'une disquette ayant un 'System' différent, sans activer automatiquement le système du programme.

Exemple : la disquette Multiplan comporte son système normal, mais nous souhaitons travailler à l'écran à partir d'un système ne comportant que les polices minimales, pour voir plus de lignes et de colonnes à l'écran. Au moment de lancer l'impression, nous réactivons le système normal, afin d'obtenir de jolis caractères.

On ne copie pas !

Pour interdire les copies d'écran - sur papier ou sur disquette - avec la séquence de touche "Commande/Majuscules/3 ou 4" il faut utiliser :

```
POKE &h2F8,0
```

Notons que ceci interdit aussi l'éjection des disquettes avec "Commande/Majuscules/1 ou 2".

Pour de nouveau autoriser ces fonctions :

```
POKE &h2F8,1
```

Passage à Excel

Quand on charge des tableaux Multiplan dans Excel, une fenêtre de dialogue peut apparaître, demandant si l'on souhaite avoir seulement le total des erreurs : ce n'est pas la peine de demander plus de détail ; on ne pourra de toute façon rien y faire.
Les erreurs proviennent par exemple de fonctions telles Nbiter() ou Delta() qui, dans Excel, sont remplacées par des paramètres de calcul itératif placés dans des fenêtres.

De même, les formats d'Excel sont plus riches, sauf en ce qui concerne le format Histogramme qui n'existe plus, probablement à cause de la facilité de réaliser des graphiques avec Excel.

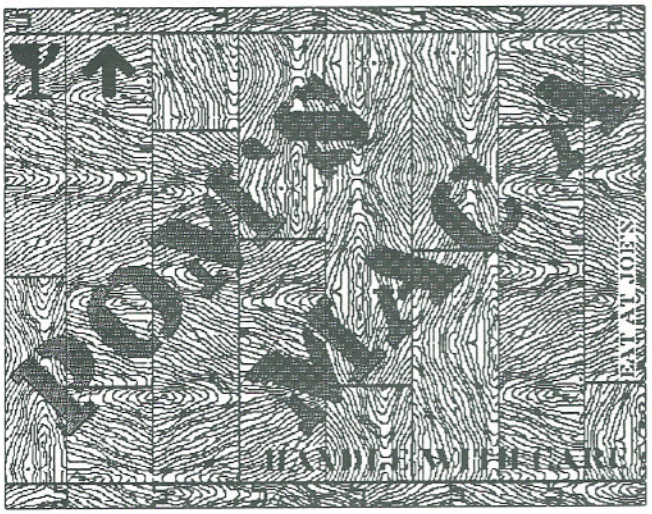
Si l'on traduit souvent des tableaux de Multiplan vers Excel, le mieux est de construire des macros qui automatisent autant que possible les conversions les plus fréquentes (par exemple pour les formats).

'Bip' ou pas 'Bip'

Pour mettre en ou hors fonction le haut parleur depuis un programme, il faut 'POKer' respectivement 1 ou 0 à l'adresse $260.
Quelqu'un connaît-il la méthode pour faire varier simplement le niveau sonore ?

DISQUETTE MAC A

SORT MENU : un utilitaire de tri de menus ;
PAINT MOVER : copier et manipuler les images MacPaint (uniquement sur 512Ko ou Plus) ;
ACCESSOIRES DE BUREAU
'MOCK' : mini traitement de texte, grapheur et terminal ;
CONVERT DAM : pour passer de DA Mover à Font/DA Mover ;
FEDIT : un éditeur de blocs et de fichiers (en anglais) ;
CUBE : un jeu en accessoire ;
IDLE : pour préserver votre écran durant de longs traitements ;
MULTISCRAP : accessoire permettant de disposer de plusieurs album à la fois ;
C CHECK : pour vérifier les niveaux de parenthèses en langage 'C' ;
HEX CALC : l'indispensable calculatrice hexadécimale ;
MEGAROÏDS : un jeu d'adresse ;
MEM WINDOW : permet de lister la mémoire du Mac.



Disquette Mac 'A' : 80,00 F TTC Franco, Bon de commande page 74

Vos impôts avec Multiplan sur Macintosh

La feuille de calcul Multiplan listée pages suivantes et destinée à l'Apple // est compatible avec Multiplan sur Macintosh. Il faut :

- saisir la liste avec un traitement de texte quelconque (MacWrite, Word, éditeur MDS) ;
- remplacer la première ligne (`ID;PMP`) par `ID;PMP;N` ;
- sauvegarder le tout sous format TEXTE ;
- charger le fichier résultant depuis Multiplan.

Une version spécialement modifiée pour le Macintosh est disponible sur la disquette Mac 23.

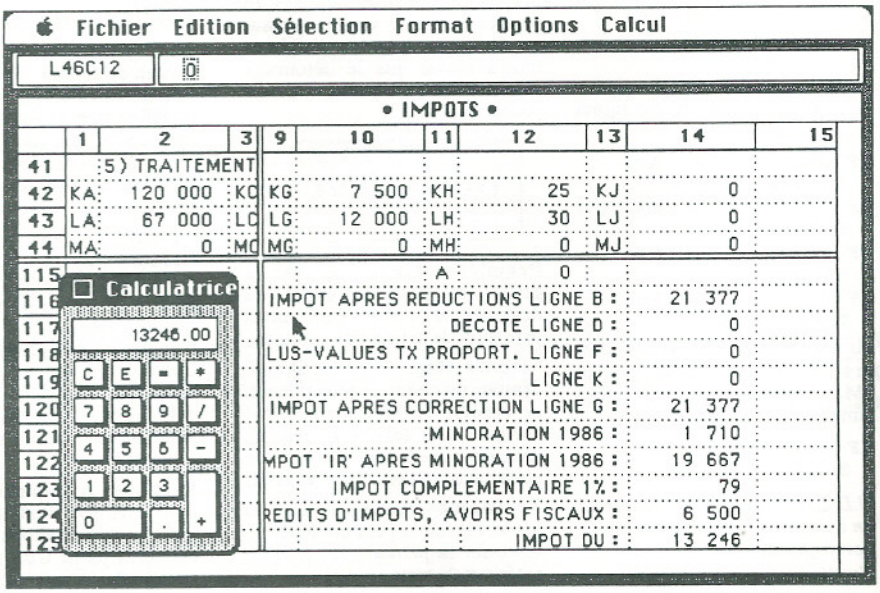
DISQUETTE MACINTOSH 23

Cette nouvelle disquette est accompagnée de la version **"Run Time"** du **Basic Microsoft 2.10**, qui ne comporte que l'interpréteur. Ceci permettra à ceux d'entre-vous qui ne disposent pas du Basic d'utiliser les programmes publiés dans Pom's. Cette version ne remplace pas le Basic Microsoft tel que l'on peut le trouver dans le commerce (elle permet seulement l'exécution des programmes), mais a l'avantage de prendre moins de place sur les disquettes.

Vous trouverez aussi une version 'Mac' de la **feuille de calcul Multiplan** listée pages suivantes, autorisant un calcul simple du montant de vos impôts.

Keeper est un petit utilitaire qui garde le Finder en mémoire, réduisant ainsi le temps de retour au bureau à mois d'une seconde (ne fonctionne pas sur un Macintosh 128Ko).

Et, bien sûr, les fichiers et programmes liés aux articles publiés dans ce numéro.



IMPÔTS MULTIPLAN

Christian Piard

Suite logique du pensum que constitue chaque année la déclaration de revenus, voici le nécessaire pour calculer simplement le montant de vos impôts.

Nous avons le privilège d'avoir un Code Général des Impôts détaillé (euphémisme !). Le résumer dans une feuille de calcul d'Apple n'entre heureusement pas dans nos ambitions : il n'est question ici que d'un calcul utilitaire intégrant les revenus de la grande majorité des contribuables. La présente feuille de calcul est le reflet de la *fiche de calcul facultatif*, pages 9 à 12 de votre *notice*.

Les calculs

Pour la détermination de l'impôt, il a été tenu compte des limites, déductions et abattements en tous genres avec leurs limites inférieures et supérieures, des éventuels 'abattements spéciaux', du plafonnement du quotient familial, des éventuelles décotes, de la minoration de 3 % 1985, du 1 % sur le revenu des capitaux mobiliers et autres avoirs fiscaux...

Les lecteurs d'Outre-Mer seront déçus : il n'a pas été possible d'y intégrer le plafonnement D.O.M.

Comment faire ?

Le fichier listé dans ces pages est en format symbolique. Si ne vous disposez pas de la disquette d'accompagnement, vous devrez saisir le-dit fichier à l'aide d'un traitement de texte, le sauvegarder en format TEXT, le relire sous Multiplan en précisant format symbolique. Cette feuille de calcul occupera 99 % de la mémoire disponible sur un Apple // 64 Ko, et 57 % sur un 128 Ko.

De la ligne 1 à la ligne 64, vous trouverez une copie de votre déclaration de revenus : il vous suffit de recopier celle-ci dans Multiplan.

Par exemple, vous avez déclaré à votre percepteur un salaire imposable de 100 000,00 F que vous avez inscrit dans le •5, ligne KA ; il vous suffit de reporter cette somme dans la feuille de calcul, cellule L42C2, celle qui vient juste après la mention KA. Il en va de même pour l'ensemble des sommes que vous avez portées sur votre déclaration. Ligne 125, vous devriez alors trouver le montant de l'impôt dû, les calculs intermédiaires se faisant dans les lignes 66-123.

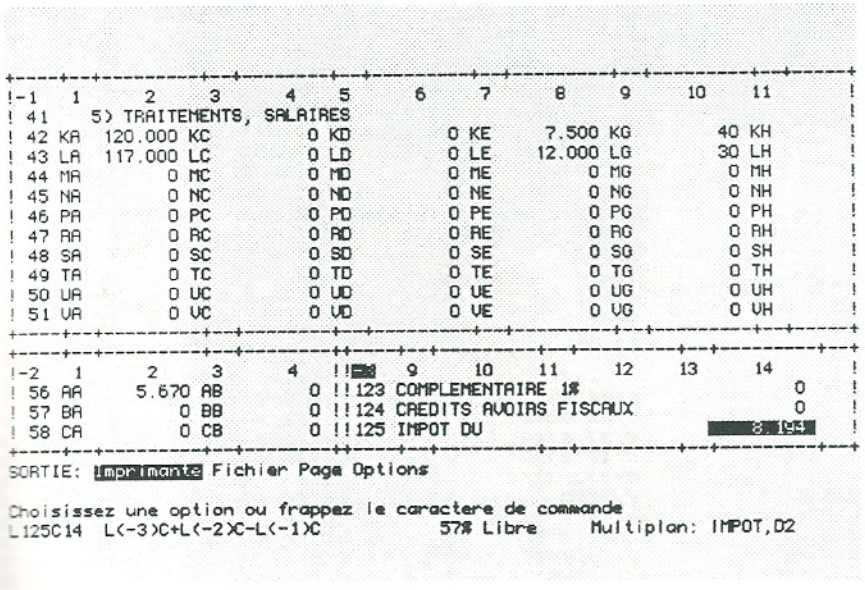
Attention, au hasard de vos essais, pour effacer un montant, mettez 0, mais ne laissez pas la cellule à blanc : les abattements et plafonnements deviendraient fantaisistes...

Quelques précisions

Seul un fiscaliste est capable de se retrouver dans les inextricables méandres de la législation. Cette feuille de calcul a été testée dans bien des cas, les planchers et plafonds vérifiées maintes fois mais, dans ce domaine, on ne peut être exhaustif ; il reste certainement des cas où l'interaction de telle condition avec telle autre pose un problème.

Le but de cette feuille est utilitaire, n'y cherchez pas une finesse de programmation confinant à l'Art. L'utilisateur ne peut d'ailleurs être que modeste devant les auteurs de Multiplan, auteurs que certains appellent les 'Paganinis de l'informatique'.

Des artistes quoi !



Fichier 'IMPOTS'

```
ID;PMP
F;K;DG0G10
F;W1 1 3
F;W3 3 3
F;W5 5 3
F;W7 7 3
F;W9 9 3
F;W11 11 3
F;W13 13 3
F;FI0C;C1
F;FI0C;C5
F;FI0C;C9
F;FI0C;C11
F;FI0G;C2
F;FI0G;C3
F;FI0G;C4
F;FI0G;C6
F;FI0C;C7
F;FI0G;C8
F;FI0G;C10
F;FI0G;C12
F;FI0G;C13
F;FI0G;C14
B;Y125;X15
C;Y1;X1;K"SITUATION DE
   FAMILLE (M=1/C=2/D=3
   /=V=4)"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X8;K1
C;Y2;X1;K"INVALIDE (0/1
   )"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"P"
C;X8;K0;G
C;Y3;S;R2;C8
C;Y4;S
C;Y5;S
C;Y6;S
C;Y7;S
C;Y8;S
C;Y9;S
C;Y3;X1;K"EPOUSE INVALI
   DE (0/1)"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"A"
C;Y4;X1;K"ENFANTS"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"F"
C;Y5;X1;K"ENFANTS INVAL
   IDES TITULAIRES CART
   E"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"G"
C;Y6;X1;K"ENFANTS INVAL
   IDES"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"H"
C;Y7;X1;K"ENFANTS CELIB
   ATAIRES RATTACHES"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"J"
C;Y8;X1;K"ENFANTS MARIE
   S RATTACHES"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"N"
C;Y9;X1;K"INVALIDES SOU
   S LE TOIT"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X7;K"R"
C;Y10;X1;K"AGE"
F;FC0R
F;X2;FC0R
F;X3;FC0R
F;X4;FC0R
F;X5;FC0R
F;X6;FC0R
C;X8;K30
C;Y11;X2;K"1) REVENUS D
   ES VALEURS ET CAPITA
   UX MOBILIERS"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;Y12;X11;K"S"
C;X12;K0
C;Y13;X11;K"U"
C;X12;K0;G
C;Y14;S;R13;C12
C;Y15;S
C;Y16;S
C;Y17;S
C;Y18;S
C;Y14;X11;K"V"
C;Y15;K"W"
C;Y16;K"X"
C;Y17;K"Y"
C;Y18;K"Z"
C;Y19;X2;K"2) REVENUS F
   ONCIERS"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;Y20;X9;K"A"
C;X10;K0
C;X11;K"B"
C;X12;K0
F;FI0R
C;Y21;X11;K"D"
C;X12;K0;G
F;FI0R
C;Y22;S;R21
F;FI0R
C;X9;K"M"
C;X10;K0;G
C;X11;K"N"
C;Y23;X2;K"3) REVENUS D
   ES PROFESSIONS NON S
   ALARIEES"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;Y24;X3;K"KA"
F;FI0C
C;X4;K0
C;X5;K"KB"
C;X6;K0
C;X7;K"KC"
C;X8;K0
C;X9;K"KD"
C;X10;K0
C;X11;K"KE"
C;X12;K0
C;Y25;X3;K"LA"
F;FI0C
C;X4;K0;G
C;Y26;S;R25;C4
C;Y28;S
C;Y29;S
C;Y31;S
C;Y32;S
C;Y33;S
C;Y34;S
C;Y25;X5;K"LB"
C;X6;K0;G
C;Y26;S;C6
C;Y27;S
C;Y28;S
C;Y29;S
C;Y30;S
C;Y31;S
C;Y32;S
C;Y33;S
C;Y25;X7;K"LC"
C;X8;K0;G
C;Y26;S;C8
C;Y27;S
C;Y28;S
C;Y29;S
C;Y30;S
C;Y31;S
C;Y32;S
C;Y33;S
C;Y25;X9;K"LD"
C;X10;K0
C;X11;K"LE"
C;X12;K0;G
C;Y26;S;C12
C;Y27;S
C;Y28;S
C;Y29;S
C;Y31;S
C;Y32;S
C;Y33;S
C;Y34;S
C;Y26;X3;K"MA"
F;FI0C
C;X5;K"MB"
C;X7;K"MC"
C;X9;K"MD"
C;X10;K0;G
C;Y27;S;R26;C10
C;Y28;S
C;Y29;S
C;Y30;S
C;Y31;S
C;Y32;S
C;Y33;S
C;Y35;S
C;Y26;X11;K"ME"
C;Y27;X3;K"NA"
F;FI0C
C;X4;K0
C;X5;K"NB"
C;X7;K"NC"
C;X9;K"ND"
C;X11;K"NE"
C;Y28;X3;K"PA"
F;FI0C
C;X5;K"PB"
C;X7;K"PC"
C;X9;K"PD"
C;X11;K"PE"
C;Y29;X3;K"RA"
F;FI0C
C;X5;K"RB"
C;X7;K"RC"
C;X9;K"RD"
C;X11;K"RE"
C;Y30;X3;K"SA"
F;FI0C
C;X4;K0
C;X5;K"SB"
C;X7;K"SC"
C;X9;K"SD"
C;Y31;X3;K"TA"
F;FI0C
C;X5;K"TB"
C;X7;K"TC"
C;X9;K"TD"
C;X11;K"TE"
C;Y32;X3;K"UA"
F;FI0C
C;X5;K"UB"
C;X7;K"UC"
C;X9;K"UD"
C;X11;K"UE"
C;Y33;X3;K"WA"
F;FI0C
C;X5;K"WB"
C;X7;K"WC"
C;X9;K"WD"
C;X11;K"WE"
C;Y34;X3;K"XA"
F;FI0C
C;X7;K"XC"
C;X8;K0
C;X9;K"XD"
C;X10;K0
C;X11;K"XE"
C;Y35;X7;K"YC"
C;X8;K0
C;X9;K"YD"
C;Y36;X2;K"4) PLUS-VALU
   ES DIVERSES ET PROFI
   TS DE CONSTRUCTION"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
F;X10;FC0L
C;Y37;X1;K"AJ"
C;X2;K0
C;X7;K"AM"
C;X8;K0
C;X9;K"AN"
C;X10;K0
C;X11;K"AP"
C;X12;K0
C;Y38;X1;K"BJ"
C;X2;K0
C;X3;K"BK"
F;FI0C
C;X4;K0;G
C;Y39;S;R38;C4
C;Y40;S
C;Y38;X7;K"BM"
C;X8;K0;G
C;Y39;S;C8
C;Y40;S
C;Y38;X9;K"BN"
C;X10;K0
C;X11;K"BP"
C;X12;K0
C;Y39;X3;K"CK"
C;X5;K"CL"
C;X6;K0
C;X7;K"CM"
C;X9;K"CN"
C;X10;K0
C;X11;K"CP"
C;X12;K0
C;Y40;X3;K"DK"
C;X5;K"DL"
C;X6;K0
C;X7;K"DM"
C;Y41;X2;K"5) TRAITEMEN
   TS, SALAIRES"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
C;Y42;X1;K"KA"
C;X2;K0
C;X3;K"KC"
F;FI0C
C;X4;K0;G
C;Y43;S;R42;C4
C;Y44;S
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y51;S
C;Y52;S
C;Y53;S
```

```
C;Y42;X5;K"KD"
C;X6;K0;G
C;Y43;S;C6
C;Y44;S
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y42;X7;K"KE"
C;X8;K0
C;X9;K"KG"
C;X10;K0
C;X11;K"KH"
C;X12;K0;G
C;Y43;S;C12
C;Y44;S
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y50;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y42;X13;K"KJ"
F;FI0C
C;X14;K0;G
C;Y43;S;C14
C;Y44;S
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y50;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y43;X1;K"LA"
C;X2;K0
C;X3;K"LC"
F;FI0C
C;X5;K"LD"
C;X7;K"LE"
C;X8;K0;G
C;Y44;S;R43;C8
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y50;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y43;X9;K"LG"
C;X10;K0;G
C;Y44;S;C10
C;Y45;S
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y50;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y43;X11;K"LH"
C;X13;K"LJ"
F;FI0C
C;Y44;X1;K"MA"
C;X2;K0;G
C;Y45;S;R44;C2
C;Y46;S
C;Y47;S
C;Y48;S
C;Y49;S
C;Y50;S
C;Y51;S
C;Y52;S
C;Y53;S
C;Y54;S
C;Y44;X3;K"MC"
F;FI0C
C;X5;K"MD"
C;X7;K"ME"
C;X9;K"MG"
C;X11;K"MH"
C;X13;K"MJ"
F;FI0C
C;Y45;X1;K"NA"
C;X3;K"NC"
F;FI0C
C;X5;K"ND"
C;X7;K"NE"
C;X9;K"NG"
C;X11;K"NH"
C;X13;K"NJ"
F;FI0C
C;Y46;X1;K"PA"
C;X3;K"PC"
F;FI0C
C;X5;K"PD"
C;X7;K"PE"
C;X9;K"PG"
C;X11;K"PH"
C;X13;K"PJ"
F;FI0C
C;Y47;X1;K"RA"
C;X3;K"RC"
F;FI0C
C;X5;K"RD"
C;X7;K"RE"
C;X9;K"RG"
C;X11;K"RH"
C;X13;K"RJ"
F;FI0C
C;Y48;X1;K"SA"
C;X3;K"SC"
F;FI0C
C;X5;K"SD"
C;X7;K"SE"
C;X9;K"SG"
C;X11;K"SH"
C;X13;K"SJ"
F;FI0C
C;Y49;X1;K"TA"
C;X3;K"TC"
F;FI0C
C;X5;K"TD"
C;X7;K"TE"
C;X9;K"TG"
C;X11;K"TH"
C;X13;K"TJ"
F;FI0C
C;Y50;X1;K"UA"
C;X3;K"UC"
F;FI0C
C;X4;K0
C;X5;K"UD"
C;X6;K0
C;X7;K"UE"
C;X9;K"UG"
C;X11;K"UH"
C;X13;K"UJ"
F;FI0C
C;Y51;X1;K"VA"
C;X3;K"VC"
F;FI0C
C;X5;K"VD"
C;X7;K"VE"
C;X9;K"VG"
C;X11;K"VH"
C;X13;K"VJ"
F;FI0C
C;Y52;X1;K"WA"
C;X3;K"WC"
F;FI0C
C;X5;K"WD"
C;X7;K"WE"
C;X9;K"WG"
C;X11;K"WH"
C;X13;K"WJ"
F;FI0C
C;Y53;X1;K"XA"
C;X3;K"XC"
F;FI0C
C;X5;K"XD"
C;X7;K"XE"
C;X9;K"XG"
C;X11;K"XH"
C;X13;K"XJ"
F;FI0C
C;Y54;X1;K"YA"
C;Y55;X2;K"6) CHARGES A
    DEDUIRE"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;Y56;X1;K"AA"
C;X2;K0
C;X3;K"AB"
F;FI0C
C;X4;K0
C;X5;K"AC"
C;X6;K0
C;X7;K"AD"
C;X8;K0
C;X9;K"AE"
C;X10;K0;G
C;Y59;S;R56;C10
C;Y56;X11;K"AP"
C;X12;K0
C;X13;K"AR"
C;X14;K0
C;Y57;X1;K"BA"
C;X2;K0
C;X3;K"BB"
F;FI0C
C;X4;K0
C;X5;K"BE"
C;X6;K0
C;X9;K"BR"
C;X10;K0
C;Y58;X1;K"CA"
C;X2;K0
C;X3;K"CB"
F;FI0C
C;X4;K0;G
C;Y59;S;R58;C4
C;Y58;X7;K"CP"
C;X8;K0
C;X9;K"CR"
C;X10;K0
C;X11;K"CS"
C;X12;K0
C;Y59;X1;K"DA"
C;X2;K0;G
C;X3;K"DB"
F;FI0C
C;X5;K"DE"
C;X6;K0;G
C;X7;K"DP"
C;X8;K0;G
C;X9;K"DR"
C;Y60;X2;K"7) CHARGES O
    UVRANT DROIT A DES R
    EDUCTIONS D'IMPOT"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;Y61;X1;K"GA"
C;X2;K0;G
C;Y63;S;R61;C2
C;Y64;S
C;Y61;X3;K"GB"
F;FI0C
C;X4;K0;G
C;Y62;S;C4
C;Y63;S
C;Y61;X5;K"GC"
C;X6;K0;G
C;Y62;S;C6
C;Y63;S
C;Y64;S
C;Y61;X7;K"GE"
C;X8;K0;G
C;Y62;S;C8
C;Y63;S
C;Y64;S
C;Y61;X9;K"GP"
C;X10;K0;G
C;Y62;S;C10
C;Y63;S
C;Y64;S
C;Y61;X11;K"GS"
C;X12;K0
C;Y62;X1;K"HA"
C;X2;K0
C;X3;K"HB"
F;FI0C
C;X5;K"HE"
C;X7;K"HP"
C;X9;K"HR"
C;Y63;X1;K"JA"
C;X3;K"JB"
F;FI0C
C;X5;K"JC"
C;X7;K"JE"
C;X9;K"JP"
C;Y64;X1;K"XA"
C;X5;K"XE"
C;X7;K"XP"
C;X9;K"XR"
C;Y68;X2;K"REVENUS DES
    VALEURS ET CAPITAUX
    MOBILIERS"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
C;Y69;X2;EIF(R[-56]C[+1
    0]>=1000,R[-56]C[+10
    ],0)+R[-55]C[+10];K0
F;FI0R
C;X4;EIF(R[-56]C[+8]>=1
    000,1000,0)+IF(R[-55
    ]C[+8]>=5000,5000,R[
    -55]C[+8]);K0
F;FI0R
C;X6;ERC[-4]-RC[-2];K0
F;FI0R
C;X8;EMAX(RC[-2],RC[-2]
    +IF(R[-54]C[+4]>=300
    0,R[-54]C[+4]-3000,0
    )+R[-53]C[+4]-R[-52]
    C[+4]+R[-57]C[+4]);K
    0
F;FI0R
C;X13;K"L 1"
F;FI0R
C;X14;ERC[-6];K0
F;FI0R
C;Y70;X2;K"REVENUS FONC
    IERS"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
C;X13;K"L 2"
F;FI0R
C;X14;EMAX(R[-50]C[-4]+
    R[-48]C[-4]-(R[-50]C
    [-2]+R[-49]C[-2]+R[-
    48]C[-2]),0);K0
F;FI0R
C;Y71;X2;K"REVENUS DES
    PROFESSIONS NON SALA
    RIEES"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
C;Y72;X4;ESUM(R[-48]C:R
    [-43]C)-SUM(R[-42]C:
    R[-39]C);D;K0
C;X6;S;R72;C4;K0
C;X8;S;K0
C;X10;S;K0
C;X12;S;K0
F;X5;FI0G
F;X7;FI0G
F;X9;FI0G
F;X11;FI0G
C;Y73;X10;EMAX(R[-1]C-I
    F(R[-1]C>3000,R[-1]C
    /2,1500),0);D;K0
C;X12;EMAX(R[-1]C-IF(R[
    -1]C>8000,R[-1]C*0.2
    5,2000),0);K0
C;Y74;X13;K"L 3"
F;FI0R
C;X14;EMAX(R[-40]C[-6]+
    R[-40]C[-4]-R[-39]C[
    -6]-R[-39]C[-4],0);K
    0
C;Y75;X13;K"L 4"
F;FI0R
C;X14;ESUM(R[-2]C[-4]:R
    [-2]C[-2])+SUM(R[-3]
    C[-6]:R[-3]C[-10]);K
    0
```

```
C;Y76;X2;K"PLUS-VALUES
   A COURT TERME ET PRO
   FITS DE CONSTRUCTION
   "
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
F;X10;FC0L
C;X13;K"L 5"
F;FI0R
C;X14;ER[-39]C[-12]+R[-
   38]C[-12]-R[-38]C[-1
   0]+MAX(SUM(R[-39]C[-
   4]:R[-37]C[-4])-SUM(
   R[-39]C[-2]:R[-37]C[
   -2]),0);K0
C;Y77;X2;K"TRAITEMENTS,
    SALAIRES, PENSIONS
   RENTES"
F;FC0L
F;X3;FC0L
F;X4;FC0L
F;X5;FC0L
F;X6;FC0L
F;X7;FC0L
F;X8;FC0L
F;X9;FC0L
C;Y78;X2;ER[-36]C+R[-36
   ]C[+2];K0
C;X4;ER[-35]C[-2]+R[-35
   ]C;K0
C;X6;ER[-34]C[-4]+R[-34
   ]C[-2];K0
C;X8;ER[-33]C[-6]+R[-33
   ]C[-4];K0
C;X10;ER[-32]C[-8]+R[-3
   2]C[-6];K0
C;X12;ER[-31]C[-10]+R[-
   31]C[-8];D;K0
C;X14;ESUM(R[-30]C[-12]
   :R[-29]C[-10]);K0
C;Y79;X2;EMIN(MAX(MIN(M
   AX(R[-1]C*0.1,1800),
   R[-1]C,57840),R[-37]
   C[+4]),R[-1]C);K0
C;X4;EMIN(MAX(MIN(MAX(R
   [-1]C*0.1,1800),R[-1
   ]C,57840),R[-36]C[+2
   ]),R[-1]C);K0
C;X6;EMIN(MAX(MIN(MAX(R
   [-1]C*0.1,1800),R[-1
   ]C,57840),R[-35]C),R
   [-1]C);K0
C;X8;EMIN(MAX(MIN(MAX(R
   [-1]C*0.1,1800),R[-1
   ]C,57840),R[-34]C[-2
   ]),R[-1]C);K0
C;X10;EMIN(MAX(MIN(MAX(
   R[-1]C*0.1,1800),R[-
   1]C,57840),R[-33]C[-
   4]),R[-1]C);K0
C;X12;EMIN(MAX(MIN(MAX(
   R[-1]C*0.1,1800),R[-
   1]C,57840),R[-32]C[-
   6]),R[-1]C);D;K0
C;X14;S;R79;C12;K0
C;Y80;X2;EMIN((R[-38]C[
   +6]*R[-38]C[+8]+R[-3
   8]C[+10]*R[-38]C[+12
   ])*0.009,50000);K0
C;X4;E(R[-37]C[+4]*R[-3
   7]C[+6]+R[-37]C[+8]*
   R[-37]C[+10])*0.009;
   K0
C;X6;E(R[-36]C[+2]*R[-3
   6]C[+4]+R[-36]C[+6]*
   R[-36]C[+8])*0.009;K
   0
C;X8;E(R[-35]C*R[-35]C[
   +2]+R[-35]C[+4]*R[-3
   5]C[+6])*0.009;K0
C;X10;E(R[-34]C[-2]*R[-
   34]C+R[-34]C[+2]*R[-
   34]C[+4])*0.009;K0
C;X12;E(R[-33]C[-4]*R[-
   33]C[-2]+R[-33]C*R[-
   33]C[+2])*0.009;D;K0
C;X14;E(R[-32]C[-6]*R[-
   32]C[-4]+R[-32]C[-2]
   *R[-32]C+R[-31]C[-6]
   *R[-31]C[-4]+R[-31]C
   [-2]*R[-31]C)*0.009;
   K0
C;Y81;X2;ER[-3]C-R[-2]C
   -R[-1]C;D;K0
C;X4;S;R81;C2;K0
C;X6;S;K0
C;X8;S;K0
C;X10;S;K0
C;X12;S;K0
C;X14;S;K0
C;Y82;X2;EIF(R[-32]C>18
   00,MAX(R[-32]C*0.1,1
   800),R[-32]C);D;K0
C;X4;EIF(R[-31]C[-2]>18
   00,MAX(R[-31]C[-2]*0
   .1,1800),R[-31]C[-2]
   );K0
C;X6;EIF(R[-30]C[-4]>18
   00,MAX(R[-30]C[-4]*0
   .1,1800),R[-30]C[-4]
   );K0
C;X8;EIF(R[-29]C[-6]>18
   00,MAX(R[-29]C[-6]*0
   .1,1800),R[-29]C[-6]
   );K0
C;X10;ESUM(RC[-8]:RC[-2
   ]);K0
C;Y83;X2;EIF(R[-1]C[+8]
   >24400,R[-1]C*24400/
   R[-1]C[+8],R[-1]C);D
   ;K0
C;X4;EIF(R[-1]C[+6]>244
   00,R[-1]C*24400/R[-1
   ]C[+6],R[-1]C);K0
C;X6;EIF(R[-1]C[+4]>244
   00,R[-1]C*24400/R[-1
   ]C[+4],R[-1]C);K0
C;X8;EIF(R[-1]C[+2]>244
   00,R[-1]C*24400/R[-1
   ]C[+2],R[-1]C);K0
C;Y84;X2;ER[-3]C+R[-34]
   C-R[-1]C;K0
C;X4;ER[-3]C+R[-33]C[-2
   ]-R[-1]C;K0
C;X6;ER[-3]C+R[-32]C[-4
   ]-R[-1]C;K0
C;X8;ER[-3]C+R[-31]C[-6
   ]-R[-1]C;K0
C;X10;ER[-3]C;D;K0
C;X12;S;R84;C10;K0
C;X14;S;K0
C;Y85;X2;EMIN(R[-1]C*0.
   2,104600);D;K0
C;X4;S;R85;C2;K0
C;X6;S;K0
C;X8;S;K0
C;X10;S;K0
C;X12;S;K0
F;X5;FI0G
F;X9;FI0G
F;X11;FI0G
C;X14;EIF(R[-37]C[-12]>
   192200,R[-1]C*0.1+17
   298,R[-1]C*0.2);K0
C;Y86;X2;ER[-2]C-R[-1]C
   ;D;K0
C;X4;S;R86;K0
C;X6;S;K0
C;X8;S;K0
C;X10;S;K0
C;X12;S;K0
C;X14;S;K0
F;X5;FI0G
F;X9;FI0G
F;X11;FI0G
C;Y87;X2;E(R[-37]C[+2]*
   R[-37]C[+4]/100)+(R[
   -37]C[+6]*R[-37]C[+8
   ]/100)+(R[-37]C[+10]
   *R[-37]C[+12]/100);K
   0
C;X4;E(R[-36]C*R[-36]C[
   +2]/100)+(R[-36]C[+4
   ]*R[-36]C[+6]/100)+(
   R[-36]C[+8]*R[-36]C[
   +10]/100);K0
C;X6;E(R[-35]C[-2]*R[-3
   5]C/100)+(R[-35]C[+2
   ]*R[-35]C[+4]/100)+(
   R[-35]C[+6]*R[-35]C[
   +8]/100);K0
C;X8;E(R[-34]C[-4]*R[-3
   4]C[-2]/100)+(R[-34]
   C*R[-34]C[+2]/100)+(
   R[-34]C[+4]*R[-34]C[
   +6]/100);K0
C;X13;K"L 6"
F;FI0R
C;X14;ESUM(R[-1]C:R[-1]
   C[-12]);K0
C;Y88;X13;K"L 7"
F;FI0R
C;X14;ESUM(R[-1]C[-12]:
   R[-1]C[-6]);K0
C;Y89;X13;K"L 8"
F;FI0R
C;X14;ER[-33]C[-4]+R[-3
   1]C[-10];K0
C;Y90;X13;K"L 9"
F;FI0R
C;X14;ESUM(R[-1]C:R[-3]
   C)+SUM(R[-14]C:R[-16
   ]C)+R[-20]C+R[-21]C;
   K0
C;Y91;X13;K"L10"
F;FI0R
C;X14;ESUM(R[-32]C[-12]
   :R[-32]C[-4]);K0
C;Y92;X13;K"L11"
F;FI0R
C;X14;ER[-2]C+R[-1]C;K0
C;Y93;X2;K"DEDUCTION DE
   S CHARGES"
F;FC0G
F;X3;FC0G
F;X4;FC0G
F;X5;FC0G
F;X6;FC0G
F;X7;FC0G
F;X8;FC0G
F;X9;FC0G
C;Y94;X1;K"J"
C;X2;EMIN(R[-38]C,5000*
   R[-38]C[+2]);K0
C;X3;K"K"
C;X4;ER[-38]C[+6]+MIN(R
   [-38]C[+2],16190*R[-
   38]C[+4]);K0
C;X5;K"L"
C;X6;EMIN(R[-36]C[-4],6
   000+R[-90]C[+2]*500+
   IF(R[-90]C[+2]>2,(R[
   -90]C[+2]-2)*500,0))
   ;K0
C;X7;K"M"
C;X8;EMIN(R[-36]C[+2],1
   3310)*R[-36]C[+4];K0
C;X9;K"P"
C;X10;ESUM(RC[-2]:RC[-8
   ]);K0
C;X12;ER[-2]C[+2]-RC[-2
   ];K0
C;Y95;X1;K"R"
C;X2;EIF(R[-1]C[+10]<10
   0000,MIN(25000,R[-39
   ]C[+10]),MIN(R[-1]C[
   +10]/4,R[-39]C[+10])
   );K0
C;X3;K"S"
C;X4;ER[-38]C[-2];K0
C;X5;K"T"
C;X6;EMIN(R[-38]C[-2],M
   AX(R[-1]C[+6]*0.01,I
   F(R[-38]C=0,R[-1]C[+
   6]*0.05,0)));K0
C;X7;K"V"
C;X8;EMIN(R[-38]C[+2],R
   [-1]C[+4]*0.1,400+R[
   -91]C*100);K0
C;X9;K"W"
C;X10;EMIN(R[-37]C[-2],
   R[-1]C[+2]/4);K0
C;X13;K"L12"
F;FI0R
C;X14;ER[-1]C[-4]+SUM(R
   C[-12]:RC[-4]);K0
C;Y96;X13;K"L13"
F;FI0R
C;X14;ER[-4]C-R[-1]C;K0
C;Y97;X13;K"L14"
F;FI0R
C;X14;EMIN(IF(AND(OR(R[
   -95]C[-6]=1,R[-94]C[
   -6]=1,R[-87]C[-6]>65
   ),R[-1]C<73600),IF(R
   [-1]C<45600,7360,368
   0),0),R[-1]C);K0
C;Y98;X10;K"NET IMPOSAB
   LE"
F;FC0G
F;X11;FC0G
F;X12;FC0G
F;X13;FC0G
C;X14;EINT((R[-2]C-R[-1
   ]C)/10)*10;K0
C;Y99;X2;K"NOMBRE DE PA
   RTS"
F;FC0G
F;X3;FC0G
F;X4;FC0G
F;X5;FI0R
C;X6;EIF(R[-98]C[+2]=1,
   2,1)+(R[-97]C[+2]+R[
   -96]C[+2]+R[-95]C[+2
   ]+R[-94]C[+2]+R[-92]
   C[+2]+R[-91]C[+2])/2
   +R[-90]C[+2];K2
F;FF1G
C;X10;K"QUOTIENT FAMILI
   AL"
F;FC0G
F;X11;FC0G
F;X12;FC0G
F;X13;FC0C
C;X14;ER[-1]C/RC[-8];K0
C;X15;ER[-1]C[-1]/IF(R[
   -98]C[-7]=1,2,1);K0
C;Y100;X4;K0
C;X6;K0
F;F%0G
C;Y101;X4;K15650
C;X6;K0.05
F;F%0R
C;X8;K782.5
F;FF2G
C;X10;K"TRANCHE RETENUE
   "
F;FC0G
F;X11;FC0G
F;X12;FC0G
F;X13;FC0C
C;X14;ELOOKUP(R[-2]C,R[
   -1]C[-10]:R[+12]C[-8
   ]);D;K0
F;F%0G
C;X15;ELOOKUP(R[-2]C,R[
   -1]C[-11]:R[+12]C[-9
   ]);K0
F;F%0G
C;Y102;X4;K16360
C;X6;K0.1
F;F%0R
C;X8;K1600.5
F;FF2G
C;X10;K"IMPOT AVANT COR
   RECTION"
F;FC0G
F;X11;FC0G
F;X12;FC0G
F;X13;FC0G
C;X14;ER[-4]C*R[-1]C-R[
   -3]C[-8]*LOOKUP(R[-3
   ]C,R[-2]C[-10]:R[+11
   ]C[-6]);K0
C;X15;ER[-4]C[-1]*R[-1]
   C-IF(R[-101]C[-7]=1,
   2,1)*LOOKUP(R[-3]C,R
   [-2]C[-11]:R[+11]C[-
   7]);K0
F;FI0G
C;Y103;X4;K19400
C;X6;K0.15
F;F%0R
C;X8;K2570.5
F;FF2G
C;X10;K"IMPOT APRES PLA
   FONNEMENT"
F;FC0G
F;X11;FC0G
F;X12;FC0G
F;X13;FC0G
C;Y104;X4;K30680
C;X6;K0.2
F;F%0R
C;X8;K4104.5
F;FF2G
C;X10;K"DU QUOTIENT FAM
   ILIAL"
F;FC0G
F;X11;FC0G
```

⇒ Suite de ce fichier page 60

SoftCopie

Bruno Fénart

SOFT.COPIE permet de recopier une disquette (non protégée) avec un seul drive, en seulement trois passes (contre neuf pour l'utilitaire FILER de ProDOS). Vous devez posséder un lecteur de disquette et 128Ko de RAM, la mémoire auxiliaire étant utilisée comme zone tampon.

Son principal intérêt est qu'il permet de recopier une disquette avec un seul lecteur en seulement trois passes. Il est donc particulièrement destiné aux propriétaires d'APPLE //c ne possédant pas le lecteur de disquette externe.

Il occupe très peu de place en mémoire principale et, en particulier, *il ne détruit pas un programme Applesoft en mémoire, ni ses variables s'il est en cours d'exécution, et il ne dérange pas les éventuels fichiers ouverts*. Vous pouvez donc l'incorporer comme utilitaire appelé par votre programme favori.

Utilisation

SOFT.COPIE est disponible en mode direct : il suffit de taper "BRUN SOFT.COPIE" (ou "-SOFT.COPIE"). A l'apparition du message, insérez votre disquette originale dans le lecteur, puis appuyez sur la touche <Return>. Au message suivant, insérez la disquette réception... Recommencez trois fois de suite, et c'est fini !

En fin de copie, il est possible de recommencer avec <Return>. Pour sortir de SOFT.COPIE, appuyez sur la touche <ESCape>. Cette dernière touche permet également d'abandonner une copie en cours.

Le programme vous signale les éventuelles erreurs rencontrées : disquette non insérée ou porte du lecteur ouverte, disquette protégée contre l'écriture, ou disquette non formatée.

En effet, la disquette de réception devra auparavant avoir été formatée en 16 secteurs, soit sous ProDOS par l'utilitaire FILER ou la commande INIT parue dans Pom's 20, soit sous DOS 3.3 ou PASCAL par la procédure correspondante.

SOFT.COPIE permet de recopier toute disquette formatée en 16 secteurs, c'est à dire utilisée avec les systèmes d'exploitation ProDOS, DOS 3.3, Pascal ou CP/M.
Ne confondons pas : SOFT.COPIE tourne sous ProDOS et nécessite, en particulier, le disque virtuel de 64Ko en mémoire auxiliaire. Il ne peut donc pas être utilisé avec un autre système d'exploitation (bien qu'il puisse recopier des disquettes de formats différents).

Le message d'erreur I/O ERROR est généré si vous tentez d'utiliser SOFT.COPIE sans avoir de disque virtuel installé par ProDOS, en particulier si vous ne disposez pas d'une configuration mémoire totale de 128 Ko ou plus.

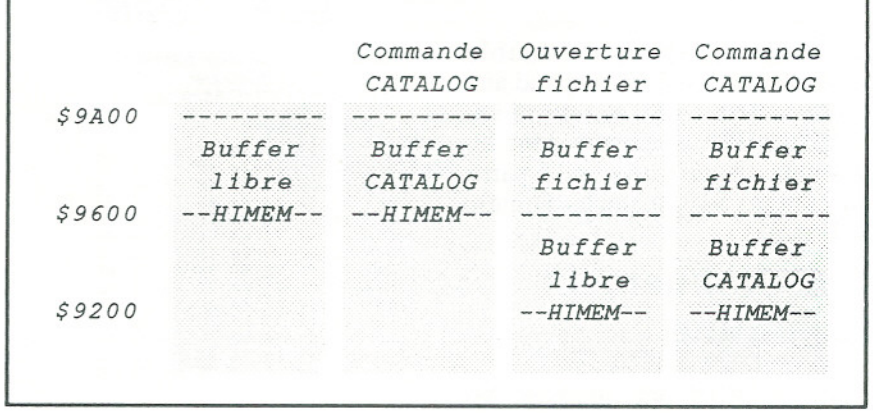
Description technique

En mémoire principale, SOFT.COPIE n'utilise que deux zones :

- la page 3 (adresses de $300 à $3CF) ;
- le buffer temporaire de 1Ko placé dynamiquement par ProDOS juste au-dessus de HIMEM:.
  La figure 1 illustre l'utilisation de ce buffer par ProDOS.

Le disque virtuel 64Ko en mémoire auxiliaire, initialisé par ProDOS lors de son chargement initial, est utilisé en quasi-totalité par SOFT.COPIE. Néanmoins, les blocs du disque virtuel qui correspondent au mode texte 80 colonnes et au graphique double haute-résolution sont préservés dans leur état initial.

Figure 1 : Occupation de la mémoire avant et pendant une commande CATALOG ou CAT



Ce disque virtuel, normalement dénommé /RAM, comporte 128 blocs (contre 280 pour une disquette). Le nombre de fichiers que peut contenir son répertoire principal est de 12 (contre 51).

La figure 2 montre la correspondance entre les blocs du volume /RAM et les adresses physiques en mémoire auxiliaire.

Les blocs 0 à 7 sont utilisés par ProDOS et ne doivent pas être modifiés directement :

- le bloc 2 contient le "Volume directory" ;
- le bloc 3 contient le "Volume Bit Map" ;
- le bloc 7 correspond aux adresses de la pile et de la page zéro de la mémoire auxiliaire. Il est donc fortement déconseillé de le modifier sous peine de graves problèmes ;
- les blocs 0,1,4,5 et 6 sont inaccessibles en lecture et en écriture. Une tentative de lecture de ces blocs donnera toujours zéro. Une tentative d'écriture ne les modifiera pas (ils resteront à zéro), ni les adresses en mémoire auxiliaire correspondant à ces blocs (adresses $200 à $BFF). Les affichages texte et graphique basse résolution en 80 colonnes sont donc automatiquement protégés.

Il est prévu que les futures versions de ProDOS utilisent d'autres adresses en mémoire auxiliaire. C'est déjà le cas pour les versions à partir de ProDOS 1.1.1 pour lesquelles le dernier bloc ($7F) est réservé. Le nombre total de blocs du disque virtuel devient 127.

La page graphique double haute-résolution 1 correspond aux blocs $08, $0A à $18. SOFT.COPIE n'utilisera donc les blocs du disque virtuel qu'à partir du numéro $19, jusqu'au numéro $7E inclus.

Fonctionnement technique

SOFT.COPIE va utiliser les caractéristiques du disque virtuel pour réaliser une recopie en trois passes.

Pourquoi trois passes ?

D'après les impératifs précédents, le volume /RAM comporte 93 blocs disponibles. Or, une disquette de 140 Ko en comporte 280, c'est-à-dire : 280 = 93 x 3 + 1.

La solution adoptée consiste donc à occuper la totalité de l'espace virtuel disponible ainsi qu'un buffer de 512 octets en mémoire principale, soit un total de 94 blocs utilisés 3 fois (afin de simplifier le programme, les blocs 93 et 196 sont recopiés deux fois).

L'avantage majeur est que SOFT.COPIE n'utilise alors que 208 octets et peut donc se loger dans la zone tranquille en page 3 ; en particulier, on n'a pas besoin de se soucier d'avoir à le reloger en mémoire haute comme on devrait le faire pour un programme plus important.

Figure 2 : Correspondance entre blocs du disque virtuel et numéros de page en mémoire auxiliaire 64 Ko

Page	0/1	2/3	4/5		8/9	A/B	C/D	E/F
$00	$07 (0,1,4,5,6 non accessibles)						$03	$02
$10	$09	$1A	$2B	$3C	$4D	$5D	$5E	$5F
$20	$08	$0A	$0B	$0C	$0D	$0E	$0F	$10
$30	$11	$12	$13	$14	$15	$16	$17	$18
$40	$19	$1B	$1C	$1D	$1E	$1F	$20	$21
$50	$22	$23	$24	$25	$26	$27	$28	$29
$60	$2A	$2C	$2D	$2E	$2F	$30	$31	$32
$70	$33	$34	$35	$36	$37	$38	$39	$3A
$80	$3B	$3D	$3E	$3F	$40	$41	$42	$43
$90	$44	$45	$46	$47	$48	$49	$4A	$4B
$A0	$4C	$4E	$4F	$50	$51	$52	$53	$54
$B0	$55	$56	$57	$58	$59	$5A	$5B	$5C
$C0	Zone des E/S et des commutateurs							
$D0 banc1	$60	$61	$62	$63	$64	$65	$66	$67
$D0 banc2	$68	$69	$6A	$6B	$6C	$6D	$6E	$6F
$E0	$70	$71	$72	$73	$74	$75	$76	$77
$F0	$78	$79	$7A	$7B	$7C	$7D	$7E	$7F

Récapitulation 'SOFT.COPIE'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE SOFT.COPIE,A$29D,L$131

```
029D- 4C 00 03
02A0- 49 6E 73 7B 72 65 7A 20
02A8- 6C 65 20 64 69 73 71 75
02B0- 65 20 00 73 6F 75 72 63
02B8- 65 3A 00 64 65 73 74 69
02C0- 6E 61 74 69 6F 6E 3A 00
02C8- 43 6F 70 69 65 20 00 61
02D0- 6E 6E 75 6C 7B 65 0D 00
02D8- 74 65 72 6D 69 6E 7B 65
02E0- 0D 00 45 72 72 65 75 72
02E8- 20 64 27 45 2F 53 07 00
02F0- 44 69 73 71 75 65 20 70
02F8- 72 6F 74 7B 67 7B 07 00
0300- 20 99 F3 20 73 F2 20 58
0308- FC A9 0C A4 73 D0 7B A5
0310- 74 8D C7 03 8D CD 03 8C
0318- C8 03 A9 B3 20 98 03 A9
0320- C4 A2 CA 20 45 03 8C C8
0328- 03 A9 BB 20 98 03 A9 CA
0330- A2 C4 20 45 03 CE C8 03
0338- C0 BA D0 DE A9 D8 A2 C8
0340- 20 9A 03 F0 C4 8D 5E 03
0348- 8E 66 03 A9 00 8D C9 03
0350- AC C8 03 A9 19 8D CE 03
0358- A2 77 20 00 BF 80 C4 03
0360- B0 19 20 00 BF 81 CA 03
0368- B0 11 EE C8 03 D0 03 EE
0370- C9 03 EE CE 03 EC CE 03
0378- D0 E0 60 A2 E2 C9 27 F0
0380- 0C A2 F0 C9 2B F0 06 20
0388- 8D BE 4C 09 BE 68 E9 0C
0390- 48 8C C8 03 20 BA 03 60
0398- A2 A0 48 20 BA 03 68 20
03A0- BE 03 AD 10 C0 20 0C FD
03A8- C9 8D F0 0D C9 9B D0 F5
03B0- 68 AA 68 A9 CF E0 42 D0
03B8- 85 60 20 8E FD 8A A0 02
03C0- 20 3A DB 60 03 60 00 00
03C8- 00 00 03 B0 00 00
```

Source 'T.SOFTCOPIE'

Assembleur ProCODE

```
*********************************************************
*                                                       *
*       SOFT.COPIE 4.2      (C)  Bruno Fénart  1985     *
*                                                       *
*  Créé: 15 Juillet 1985       Modifié: 14 Octobre 1985 *
*      copie rapide de disquettes avec un seul drive.   *
*                                                       *
*********************************************************
*  Présentation :
*  Ce programme permet de recopier une disquette non
*  protégée avec un seul drive, en seulement trois passes
*  (contre neuf pour l'utilitaire ProDOS).
*
*  Vous devez posséder un lecteur de disquette et
*  128K de RAM, la mémoire auxiliaire étant utilisée
*  comme zone tampon.
*  Constantes des disques réels et virtuels  *

UNITNB   EQU  $60        ;N° slot x $10 (+ $80 si drive 2)
RAM      EQU  $19        ;N° du lier bloc utilisé en /RAM
*   Codes des commandes ProDOS   *

RDBLOCK  EQU  $80        ;Lecture directe d'un bloc
WRBLOCK  EQU  $81        ;Ecriture directe d'un bloc
*   Variable en page zéro   *

HIMEM    EQU  $73        ;Fin de l'espace mémoire + 1
*   Zone d'E/S et des commutateurs   *

KBDSTRB  EQU  $C010      ;Echantillonnage du clavier
* Sous-programmes en pages globales ProDOS et BASIC.SYSTEM

ERROUT   EQU  $BE09      ;Affiche l'erreur (code BI en A)
BADCALL  EQU  $BE8D      ;Code erreur MLI --> code BI
MLI      EQU  $BF00      ;Entrée de ProDOS
*  Sous-programmes du moniteur et de l'interpréteur BASIC

STROUT   EQU  $DB3A      ;Affiche une chaîne (A,Y) fin 00
NORMAL   EQU  $F273      ;Passage en mode normal
TEXT     EQU  $F399      ;Passage en mode texte total
HOME     EQU  $FC58      ;Efface l'écran
RDKEY    EQU  $FD0C      ;Attend une touche, code mis en A
CROUT    EQU  $FD8E      ;Envoie un retour chariot
         ORG  $029D

***  Point d'entrée   ***

         JMP  DEBUT
*   Messages d'instructions   *

MESSAGE  EQU  *
DISQUE   ASC  'Insérez le disque '
         DFB  00
SOURCE   ASC  'source:'
         DFB  00
DESTIN   ASC  'destination:'
         DFB  00
COPIE    ASC  'Copie '
         DFB  00
ANNULE   ASC  'annulée'
         DFB  $0D        ;Saut de ligne
         DFB  00
TERMINE  ASC  'terminée'
         DFB  $0D        ;Saut de ligne
         DFB  00
IOERR    ASC  'Erreur d'E/S'
         DFB  $07        ;Gong
         DFB  00
WRIPRO   ASC  'Disque protégé'
         DFB  $07        ;Gong
         DFB  00
         DS   $300-*     ;Alignement
DEBUT    JSR  TEXT       ;Un écran tout neuf
         JSR  NORMAL
         JSR  HOME

AGAIN    LDA  #$0C       ;Code BI de NO BUFFERS AVAILABLE
         LDY  HIMEM
         BNE  BIERR      ; Erreur si HIMEM mal ajusté
         LDA  HIMEM+1    ; sinon les buffers de données
         STA  BUFFER+1   ; sont mis à l'adresse du buffer
         STA  XBUFFER+1  ; temporaire de BASIC.SYSTEM
         STY  BLOCK      ; Commence au bloc 0

                         ; Adr. retour d'erreur en lecture
PASSE    LDA  #<SOURCE   ; Affiche l'instruction:
         JSR  CLAVIER1   ; "Insérez le disque source:"
         LDA  #<DRIVPAR
         LDX  #<AUXPAR
         JSR  TRANS      ;Disquette source --> /RAM
         STY  BLOCK      ;Restitue le n° du bloc de départ

                         ;Adr. retour d'erreur d'écriture
         LDA  #<DESTIN   ;Affiche l'instruction:
         JSR  CLAVIER1   ;"Insérez le disque destination:"
         LDA  #<AUXPAR
         LDX  #<DRIVPAR
         JSR  TRANS      ;/RAM --> Disquette destination
         DEC  BLOCK      ;Un bloc commun entre les passes

         CPY  #$BA       ;Début dernière passe au bloc $BA
         BNE  PASSE

         LDA  #<TERMINE  ; "Copie terminée" ou
ESCAPE   LDX  #<COPIE    ; "Copie annulée"
         JSR  CLAVIER    ; puis test du clavier
         BEQ  AGAIN      ;Recommence tjs une autre copie
* Transfert des blocs du disque vers /RAM et vice versa *

TRANS    STA  LEC        ;Lecture du disque ou de /RAM
         STX  ECR        ;Ecriture du disque ou de /RAM

         LDA  #00        ;Poids fort toujours nul au départ
         STA  BLOCK+1
         LDY  BLOCK      ;Préserve le n° du bloc de départ
         LDA  #RAM       ; premier bloc en /RAM
         STA  XBLOCK
         LDX  #RAM+$5E   ;Dernier bloc en /RAM

GROUPE   JSR  MLI        ;Lecture d'un bloc
         DFB  RDBLOCK
LEC      DA   DRIVPAR
         BCS  ERREUR

         JSR  MLI        ;Ecriture d'un bloc
         DFB  WRBLOCK
ECR      DA   AUXPAR
         BCS  ERREUR

         INC  BLOCK      ;Blocs suivants
         BNE  NOCARRY
         INC  BLOCK+1    ;Drive sur 2 octets
NOCARRY  INC  XBLOCK     ;mémoire auxiliaire sur un seul.
         CPX  XBLOCK
         BNE  GROUPE     ;Copie par groupe de $5E blocs
         RTS
*    Gestion des erreurs    *

ERREUR   LDX  #<IOERR
         CMP  #$27       ;Erreur d'E/S ?
         BEQ  ERRIN
         LDX  #<WRIPRO
         CMP  #$2B       ;Ecriture protégée ?
         BEQ  ERRIN      ;Sinon autres erreurs:
         JSR  BADCALL    ;Code MLI --> code BI
BIERR    JMP  ERROUT     ;Affichage de l'erreur et arrêt

ERRIN    PLA             ;Calcul l'adresse de redémarrage
         SBC  #$0C       ;(SEC inutile après l'égalité)
         PHA
```

```
        STY  BLOCK      ;Réinitialise le n° de bloc
        JSR  AFFICHE1   ;Affiche l'erreur correspondante
        RTS             ;Retour 12 octets en arrière
*   Affichage des instructions et test du clavier   *
* CLAVIER: Message en 2 parties  (X = 1° ; Acc = 2°)
* CLAVIER1: Idem mais 1° partie = "Insérez le disque "

CLAVIER1 LDX  #<DISQUE   ;1° partie du message "Insérez...
CLAVIER  PHA             ;Préserve l'adr. de la 2° partie
        JSR  AFFICHE1   ;Affiche la 1° partie
        PLA             ;Récupère l'adresse de la 2°
        JSR  AFFICHE    ;Affiche la 2° partie (sans saut)

        LDA  KBDSTRB    ;Un clavier tout neuf
]boucle JSR  RDKEY      ;Affiche le curseur,
        CMP  #$8D       ;puis attend un retour chariot.
        BEQ  ONYVA
        CMP  #$9B       ;Si ce n'est pas le caractère ESC
        BNE  ]boucle    ;on attend un autre caractère,

        PLA             ;sinon on saute un retour
        TAX             ;Préserve l'adresse basse
        PLA             ;Dépile l'adresse haute
        LDA  #<ANNULE   ;Message "Annulé" si ESCAPE
        CPX  #<ESCAPE+4 ;Donne zéro en cas d'égalité
        BNE  ESCAPE     ;Revient au départ ou fin
ONYVA   RTS
* Affichages des messages.
* AFFICHE1:  Avec saut de ligne (X = Adr.)
* AFFICHE :  Sans saut de ligne (Acc = Adr.)

AFFICHE1 JSR  CROUT      ;Envoie un retour chariot
        TXA
AFFICHE  LDY  #>MESSAGE  ;Poids fort identique pour tous
        JSR  STROUT
        RTS
*   Table des paramètres de ProDOS   *

DRIVPAR  DFB  $03        ;Nombre de paramètres
        DFB  UNITNB     ;UNIT NUMBER du lecteur de disque
BUFFER   DA   $0000      ;Adresse du buffer de données
BLOCK    DA   $0000      ;Numéro du bloc

AUXPAR   DFB  $03        ;Nombre de paramètres
        DFB  $B0        ;UNIT NUMBER du disque virtuel
XBUFFER  DA   $0000      ;Adresse du buffer de données
XBLOCK   DA   $0000      ;Numéro du bloc
        DS   $3D0-*     ;Protège les vecteur du DOS
```

Suite de la page 56, fichier 'IMPOTS'

```
F;X12;FC0G
F;X13;FC0G
C;X14;EMAX(R[-2]C,R[-2]
   C[+1]-10520*2*(R[-5]
   C[-8]-IF(R[-103]C[-6
   ]=1,2,1)));K0
C;Y105;X4;K39440
C;X6;K0.25
F;F$0R
C;X8;K6076.5
F;FF2G
C;Y106;X4;K49550
C;X6;K0.3
F;F$0R
C;X8;K8554
F;FF2G
C;X10;K"REDUCTION D'IMP
   OTS"
F;FC0C
F;X11;FC0C
F;X12;FC0C
C;Y107;X4;K59950
C;X6;K0.35
F;F$0R
C;X8;K11551.5
F;FF2G
C;X11;K"A"
C;X12;EMIN(0.25*R[-46]C
   [-10]+0.2*(R[-46]C[-
   8]+R[-46]C[-6]+R[-46
   ]C[-4]),0.25*(15000+
   2000*(R[-103]C[-4]+R
   [-100]C[-4]+R[-99]C[
   -4]+R[-98]C[-4])));K
   0
C;Y108;X4;K69170
C;X6;K0.4
F;F$0R
C;X8;K15010
F;FF2G
C;X11;K"B"
C;X12;EMIN(R[-47]C[-2],
   12000+2000*(R[-104]C
   [-4]+R[-101]C[-4]+R[
   -100]C[-4]+R[-99]C[-
   4]));K0
C;Y109;X4;K115250
C;X6;K0.45
F;F$0R
C;X8;K20772.5
F;FF2G
C;X11;K"C"
C;X12;E0.25*(MIN(R[-48]
   C,IF(R[-108]C[-4]=1,
   16000,8000)+2000*(R[
   -105]C[-4]+R[-102]C[
   -4]+R[-101]C[-4]+R[-
   100]C[-4])));K0
C;Y110;X4;K158510
C;X6;K0.5
F;F$0R
C;X8;K28698
F;FF2G
C;X11;K"D"
C;X12;E(MIN(R[-48]C[-10
   ],4000+1000*(R[-106]
   C[-4]+R[-103]C[-4]+R
   [-102]C[-4]+R[-101]C
   [-4])))/4+MIN(R[-48]
   C[-8],7000+1500*(R[-
   106]C[-4]+R[-103]C[-
   4]+R[-102]C[-4]+R[-1
   01]C[-4]))/4;K0
C;Y111;X4;K187490
C;X6;K0.55
F;F$0R
C;X8;K38072.5
F;FF2G
C;X11;K"E"
C;X12;EMIN(R[-49]C[-6],
   2000);K0
C;Y112;X4;K213280
C;X6;K0.6
F;F$0R
C;X8;K48736.5
F;FF2G
C;X11;K"F"
C;X12;EMIN(R[-50]C[-4]/
   4,IF(R[-111]C[-4]=1,
   14000,7000));K0
C;Y113;X4;K241740
C;X6;K0.65
F;F$0R
C;X8;K60823.5
F;FF2G
C;X11;K"G"
C;X12;E(SUM(R[-50]C[-10
   ]:R[-50]C[-6]))/4;K0
C;Y114;X11;K"H"
C;X12;E(MIN((R[-51]C[-4
   ]+0.75*(R[-51]C[-2])
   ),IF(R[-113]C[-4]=1,
   400000,200000)))*0.0
   5;K0
C;Y115;X11;K"A"
C;X12;ESUM(R[-1]C:R[-8]
   C);K0
C;Y116;X9;K"APRES REDUC
   TION LIGNE B"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;ER[-12]C-R[-1]C[-
   2];K0
C;Y117;X9;K"DECOTE LIGN
   E D"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;EMIN(IF(AND(R[-1]
   C<4300,R[-1]C>=1700,
   R[-18]C[-8]=1),4300-
   R[-1]C,0)+IF(AND(R[-
   1]C<1700,R[-18]C[-8]
   =1.5),1700-R[-1]C,0)
   ,R[-1]C);K0
C;Y118;X9;K"IMPOTS SUR
   +VALUES LIGNE F"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;E(R[-79]C[-10]-R[
   -78]C[-10])*0.11+(R[
   -79]C[-8]-R[-78]C[-8
   ]+R[-80]C[-6])*0.16+
   (R[-79]C[-6]-R[-78]C
   [-6])*0.26+R[-81]C[-
   6]*0.51;K0
C;Y119;X9;K"LIGNE K"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;ER[-57]C[-4]+R[-5
   5]C[-4];K0
C;Y120;X9;K"APRES CORRE
   CTION LIGNE G"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;EIF((R[-4]C-R[-3]
   C+R[-2]C+R[-1]C)>339
   ,R[-4]C-R[-3]C+R[-2]
   C+R[-1]C,0);K0
C;Y121;X9;K"MINORATION
   1986"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;EIF(R[-1]C<22730,
   R[-1]C*0.08,0)+IF(AN
   D(R[-1]C>22729,R[-1]
   C<28411),4*(1420-R[-
   1]C*0.0425),0)+IF(AN
   D(R[-1]C>28410,R[-1]
   C<34091),R[-1]C*0.03
   ,0);K0
C;Y122;X9;K"APRES MINOR
   ATION 1986"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;ER[-2]C-R[-1]C;K0
C;Y123;X9;K"COMPLEMENTA
   IRE 1%"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;EIF(R[-3]C>339,R[
   -54]C*0.01,0);K0
C;Y124;X9;K"CREDITS AVO
   IRS FISCAUX"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;ER[-106]C[-2];K0
C;Y125;X9;K"IMPOT DU"
F;FC0C
F;X10;FC0C
F;X11;FC0C
F;X12;FC0C
F;X13;FC0C
C;X14;ER[-3]C+R[-2]C-R[
   -1]C;K0
W;N1;A70 12
E
```

Note : deux "retours chariots" sont indispensables à la fin de ce fichier.

Retour dans le Basic Halte aux Scrolls !

Gilles Caraux

Dans les lignes qui vont suivre, nous allons présenter deux utilitaires utilisables à l'aide de l'instruction ampersand (&) du langage Basic Applesoft.
La fécondité de cette instruction n'est plus à prouver. De jour en jour, de numéro de Pom's en numéro de Pom's, elle enrichit l'étendue des possibilités du Basic de nos Apple par l'accès facile à des programmes rédigés en langage machine.
Les deux utilitaires présentés ici regroupent des programmes permettant d'accomplir des tâches fort différentes et très utiles dans la pratique.

Le premier permet d'activer la touche ESC et ainsi de programmer la possibilité de revenir en arrière lors du déroulement séquentiel d'un programme écrit en Basic Applesoft.

Le second permet d'interrompre en bas de page le défilement continu de l'affichage à l'écran.

Pour utiliser cette bibliothèque ampersand, vous devez soit la charger par un BRUN, soit lancer l'instruction CALL 37555 lorsque la bibliothèque est déjà en mémoire. Celle-ci a été implantée dans les adresses $92B3... $949D. Toutefois, le programme étant relogeable, vous pouvez le déplacer facilement en utilisant le programme RELOCATE présenté par J.F. DUVIVIER dans le n°1 de Pom's.

Faites marche arrière dans un programme Basic

Utiliseriez vous une voiture sans marche arrière pour vous déplacer dans le dédale des rues d'une agglomération ? Non. Et pourtant, vous avez sûrement déjà utilisé ou même rédigé des programmes sans possibilité de revenir en arrière après un choix malencontreux. Ce n'est peut être pas tout-à-fait la même chose mais tout aussi fâcheux.
Nous vous proposons dans les lignes qui vont suivre, un moyen simple d'inclure dans vos programmes Basic les facilités du retour arrière programmé par la touche ESC. Cette touche sera programmable lors de chaque question posée à l'utilisateur de votre programme.
Quatre commandes utilisant l'ampersand (&) sont nécessaires pour accomplir cela. Elles gèrent une pile d'adresses de trois niveaux (implantées aux adresses $F9...$FE) permettant ainsi de faire un retour à trois étapes antérieures. Les deux premières permettent de pointer les instructions où l'on accepte de revenir en cas de demande, la troisième déclenche le retour arrière et la dernière interdit tout retour à une étape qui la précède.

&STORE

Cette commande permet de pointer une instruction Basic où l'on souhaite éventuellement revenir en cas de besoin. Comme le faisait le "Petit Pousset" avec des cailloux, elle laisse une trace du passage en certains points du programme dans l'intention de pouvoir y revenir.
Cette commande mémorise en $CE.$CF le numéro de l'instruction d'où elle est lancée. Vous pouvez la placer n'importe où dans votre programme (sauf dans un sous programme ou dans une boucle). Elle doit obligatoirement apparaître sur la ligne d'instructions où l'on désire revenir en cas de demande explicite.

La place privilégiée de &STORE se situe en général au début des instructions définissant un menu. En effet, c'est par le choix dans un menu que l'on débute normalement un traitement et c'est donc là en général qu'il est souhaitable de pouvoir revenir en cas de nécessité.
Cette commande prend soin de remettre à zéro la pile du processeur gérée par l'interpréteur Basic (adresses $100 à $1FF). Ceci permet de quitter une boucle ou un sous programme en ayant fait le ménage. Ceci interdit également, ce qui est normal, de revenir dans un sous-programme sans repasser par une instruction GOSUB et impose aussi de reprendre une boucle par l'instruction FOR. Vous ne pouvez donc pas programmer un retour à l'intérieur d'un sous-programme ou d'une boucle.

&VAL

Cette instruction est la deuxième étape de mémorisation d'un numéro d'instruction. Elle valide la demande exprimée par &STORE.
En effet &STORE n'a pas placé le numéro de l'instruction qu'elle pointe dans la pile contenant le numéro des instructions où l'on est susceptible de revenir.
Ceci était prématuré. Si, au début d'un menu, &STORE avait directement modifié la pile des numéros d'instructions ($F9... $FE), en appuyant sur "ESC" à la fin de ce menu, on retrouverait celui-ci et non celui qui le précède.
Sa place naturelle est située directement à la suite d'une instruction GET ou INPUT lorsque la réponse est autre que

"ESC".

Notons que l'on peut lancer plusieurs fois la commande &VAL après une commande &STORE. Seule la première sera prise en compte pour la modification de la pile.

&RETURN

L'exécution de cette commande vous fera revenir à la dernière instruction mémorisée par &STORE et validée par &VAL. On devra donc la placer à la suite d'un GET ou d'un INPUT après avoir testé si la réponse est CHR$(27) (touche "ESC").

&CLEAR

Cette commande remet à zéro la pile de travail définie en $F9...FE.
On l'utilisera pour interdire tout retour à une instruction précédant celle où elle apparaît. Ceci est parfois nécessaire quand pour des raisons de logique propre au programme, on ne peut retrouver par un retour arrière une situation propice à la reprise du traitement.

Halte aux SCROLLs

Le SCROLL est une sous-programme du système qui déclenche le décalage vers le haut des lignes de l'écran. Elle est utilisée pour dégager la dernière ligne du bas quand l'affichage nécessite plus de 24 lignes.
Ainsi, quand on édite un grand nombre de lignes, on provoque un défilement ininterrompu à l'écran.
La seule possibilité qui nous est donnée pour s'attarder sur l'affichage est d'utiliser CTL-S. Ceci est peu convivial et peu pratique.
Le programme en langage machine qui vous est proposé ici remédie à ce problème. Il permet une édition progressive écran par écran et non plus en continu. Il vient se substituer à la routine de sortie de caractères dont l'adresse est mémorisée en $36.$37.

&SCROLL DEL

A partir du moment où cette commande est donnée, l'affichage se fait page par page. Une fois l'écran rempli un message demande d'appuyer sur une touche pour continuer l'impression.
Pour revenir au mode d'affichage normal (ce qui est nécessaire quand on utilise les commandes DOS d'écriture sur disquette), vous devez lancer successivement les ordres : PR#0 : CALL 1002
La commande &SCROLL DEL est compatible avec les commandes de retour arrière présentées plus haut. Lors de la réponse au message apparaissant en bas de l'écran vous exécutez automatiquement &RETURN si vous appuyez sur la touche "ESC", sinon vous validez par &VAL la dernière demande de retour faite par &STORE.

Programme 'BIBLI.DEMO'

```
10  HIMEM: 37555:N% = 0:NM$ = ""
20 D$ =  CHR$ (13) +  CHR$ (4):ER$ =  CHR$
     (7):ESC$ =  CHR$ (27)
30  PRINT D$"BRUNBIBLI.AMPER"
40  &  STORE :N = 3: GOSUB 100: HTAB 7: PR
    INT "PROGRAMMES DE DEMONSTRATION": VTA
    B 6: HTAB 17: PRINT "MENU": NORMAL
50  VTAB 11: PRINT "1 - ECRIRE LES 300 PRE
    MIERS NOMBRES.": VTAB 14: PRINT "2 - A
    FFICHER LE CATALOG DE LA DISQUETTE.":
    VTAB 17: PRINT "3 - LISTER LES INSTRUC
    TIONS DU PROGRAMME": VTAB 20: PRINT "4
     - QUITER LE PROGRAMME.": GOSUB 170: O
    N O GOTO 70,110,80,90
60  GOSUB 150: &  VAL : PRINT D$"PR#0": CA
    LL 1002: TEXT : GOTO 40
70  &  VAL : &  STORE :N = 3: GOSUB 100: H
    TAB 6: PRINT "LISTES DES 300 PREMIERS
    NOMBRES": VTAB 10: NORMAL : & SCROLL D
    EL : FOR I = 1 TO 300: PRINT I;"  ";:
    NEXT : GOTO 60
80  &  VAL : &  STORE :N = 3: GOSUB 100: H
    TAB 8: PRINT "LISTING DU PROGRAMME BAS
    IC": VTAB 10: NORMAL : & SCROLL DEL :
    LIST : GOTO 60
90 N = 3: GOSUB 100: HTAB 15: PRINT "AU RE
    VOIR !": VTAB 23: NORMAL : END
100  TEXT : HOME : INVERSE : GOSUB 210: FO
    R I = 1 TO N: GOSUB 220: NEXT : GOSUB
    210: VTAB 3: RETURN
110  &  VAL : &  STORE :N = 5: GOSUB 100:
    HTAB 6: PRINT "LISTES DES FICHIERS SE
    TROUVANT": PRINT : HTAB 3: PRINT "ACTU
    ELLEMENT SUR LE DISQUE  NUMERO 1": VTA
    B 10: NORMAL
120  & RWTS: IF N% <  > 0 THEN  GOTO 140
130  NORMAL : HOME : TEXT : VTAB (15): PRI
    NT "AUCUN FICHIER N'EXISTE SUR LE DISQ
    UE !": GOTO 60
140  & SCROLL DEL : FOR I = 1 TO N%: &  RE
    AD NM$ > I: PRINT I;" - "NM$: PRINT :
    NEXT I: GOTO 60
150  VTAB 24: HTAB 1: INVERSE : PRINT "  P
    OUR CONTINUER TAPER SUR UNE TOUCHE.";:
     GET O$: NORMAL : IF O$ = ESC$ THEN  &
      RETURN : PRINT ER$: HTAB 1: RETURN
160  &  VAL : HTAB 1: RETURN
170  VTAB 24: HTAB 6: INVERSE : PRINT "REP
    ONDRE";: NORMAL : PRINT "  1 2 ... 4 :
    ";: HTAB 28: GET O$: IF O$ = ESC$ THEN
      &  RETURN
180 O =  VAL (O$): IF O > 0 AND O <  = 4 T
    HEN  HTAB 1: RETURN
190  HTAB 28: PRINT ER$;: PRINT O$;
200  HTAB 31: PRINT "ERREUR";: GOTO 170
210  PRINT "+";: FOR I = 1 TO 38: PRINT "-
    ";: NEXT I: PRINT "+";: RETURN
220  PRINT "I"; SPC( 38);"I";: RETURN
```

Source 'BIBLI.TEXT'

```
 1 *
 2
 3 ADDON   EQU  $D9
 4 BELL    EQU  $FF3A
 5 CONINT  EQU  $E6FB
 6 COUT1   EQU  $FDF0
 7 ERROR   EQU  $D412
 8 FNDLIN  EQU  $D61A
 9 FRMNUM  EQU  $DD67
10 LAB1    EQU  $DEC9
11 LAB2    EQU  $A851
12 LAB3    EQU  $D941
13 RDKEY   EQU  $FD0C
14 SCROLL  EQU  $FC70
15 SETTXT  EQU  $FB39
16 VP      EQU  $FC1A
17         ORG  $92B3
18
19
20 * CHARGEMENT DE AMPERV
21         LDX  #$02
22 B0      LDA  ADR0,X
23         STA  $03F5,X
24         DEX
25         BPL  B0
26 * CHARGEMENT DE
     L'ADRESSE DE LA
     TABLE IOB
27 *       POUR RWTS
28         LDX  #$01
29 B1      LDA  ADR1+1,X
30         STA  ADR3,X
31         DEX
32         BPL  B1
33 * CHARGEMENT DE
     L'ADRESSE DE LA
     TABLE DES
34 * CARACTERISTIQUES
     DU LECTEUR
35         LDX  #$00
36         LDA  ADR2+1,X
37         STA  ADR4+1,X
38         INX
39         LDA  ADR2+1,X
40         INX
41         STA  ADR4+1,X
42         JMP  ADR5
43 ADR0    JMP  ADR6
44 ADR1    JMP  ADR7
45 ADR2    JMP  ADR8
46 *
47 * DEBUT DE '&'
48 *
49 * INTERPRETATION
     DES COMMANDES
50 ADR6    LDX  #$00
51         STX  $19
52 B3      LDY  #$00
53 B2      LDA  TC1,X
54         BEQ  ADR9
55         CMP  #$FF
56         BEQ  ADR10
57         CMP  ($B8),Y
58         BNE  ADR11
59         INY
60         INX
61         BNE  B2
62 ADR10   JMP  LAB1
63 ADR11   INX
64         LDA  TC1,X
65         BNE  ADR11
66         INX
67         INC  $19
68         BNE  B3
69 ADR9    JSR  ADDON
70         ASL  $19
71         ASL  $19
72         LDX  $19
73         LDA  ADR12+2,X
74         PHA
75         LDA  ADR12+1,X
76         PHA
77         RTS
78 * TABLE DES
     COMMANDES DE '&'
79 TC1     HEX  A800
80         HEX  E500
81         HEX  B100
82         HEX  BD00
83         ASC  'SCROLL'
84         HEX  8500
85         ASC  'RWTS'
86         HEX  00
87         HEX  874E4D24
88         HEX  CF00FF
89 * ADRESSES DES
     COMMANDES
90 ADR12   JSR  ADR13+2
91         NOP
92         JSR  ADR14
93         NOP
94         JSR  ADR15
95         NOP
96         JSR  ADR16
97         NOP
98         JSR  ADR17
99         NOP
100        JSR  ADR18
101        NOP
102 ADR13  JSR  ADR19+2
103 *
104 * DEBUT DE &STORE
105 *
106 * CHARGEMENT DE
     "CE.CF"
107
108        LDA  $75
109        STA  $CE
110        LDA  $76
111        STA  $CF
112 * MISE A ZERO
     DE LA PILE
113        PLA
114        TAY
115        PLA
116        LDX  #$FF
117        TXS
118        PHA
119        TYA
120        PHA
121 ADR14  RTS
122 *
123 * DEBUT DE &VAL
124 *
125 * CE.CF =  0 ?
     SI OUI "RTS"
126 ADR30  LDA  $CE
127        CLC
128        ADC  $CF
129        BNE  ADR20
130        RTS
131 * CE.CF = F9.FA ?
     SI OUI "RTS"
132 ADR20  LDA  $CE
133        CMP  $F9
134        BNE  ADR21
135        LDA  $CF
136        CMP  $FA
137        BNE  ADR21
138        RTS
139 * EMPILEMENT DE
     CE.CF EN F9.FA
140 ADR21  LDX  #$03
141 B4     LDA  $F9,X
142        STA  $FB,X
143        DEX
144        BPL  B4
145        LDA  $CE
146        STA  $F9
147        LDA  $CF
148        STA  $FA
149 * MISE A ZERO DE CE.CF
150        LDA  #$00
151        STA  $CE
152        STA  $CF
153 ADR15  RTS
154 *
155 * DEBUT DE &RETURN
156 *
157 * FA.F9 = 0 ?
     SI OUI "RTS"
158 B6     LDA  $F9
159        CLC
160        ADC  $FA
161        BNE  ADR22
162        RTS
163 * CHARGEMENT LINNUM
164 ADR22  LDA  $F9
165        STA  $50
166        LDA  $FA
167        STA  $51
168 * DEPILEMENT DE F9.FA
169        LDX  #$00
170 B5     LDA  $FB,X
171        STA  $F9,X
172        INX
173        CPX  #$04
174        BCC  B5
175 * SI LINNUM=CE.CF
     REPRENDRE &RETURN
176        LDA  $CE
177        CMP  $50
178        BNE  ADR44
179        LDA  $CF
180        CMP  $51
181        BEQ  B6
182 ADR44  LDA  #$00
183        STA  $FD
184        STA  $FE
185 * REINITIALISATION
186        LDA  #$FF
187        STA  $32
188        LDA  #$F0
189        STA  $36
190        LDA  #$FD
191        STA  $37
192        JSR  SETTXT
193        JSR  LAB2
194 * GO TO LINNUM
     SINON ERREUR
195        JSR  FNDLIN
196        BCC  ADR23
197        JMP  LAB3
198 ADR23  LDX  #$5A
199        JMP  ERROR
200 ADR16  RTS
201 *
202 * DEBUT DE &CLEAR
203 *
204 * MISE A ZERO DE
     LA PILE F9...
205 ADR5   LDA  #$00
206        LDX  #$05
207 B7     STA  $F9,X
208        DEX
209        BPL  B7
210 *MISE A ZERO DE CE.CF
211        STA  $CE
212        STA  $CF
213 ADR17  RTS
214 *
215 * DEBUT DE &SCROLL DEL
216 *
217 * CHARGEMENT DE CSW:
218 * ADRESSE DE LA
     ROUTINE DE SORTIE
219 * DE CARACTERES.
220        LDA  ADR24+1
221        STA  $36
222        LDA  ADR24+2
223        STA  $37
224        JSR  LAB2
225        RTS
226 ADR24  JSR  ADR25
227 * ROUTINE DE SORTIE
     DE CARACTERES.
228 * SAUVEGARDE DES
     REGISTRES: X,Y ET A
229 ADR25  STX  $07
230        STY  $08
231        PHA
232 * SI POSITION DU
     CURSEUR < 23
233        LDA  $25
```

```
234           CMP  #$16
235           BEQ  ADR26
236 * ALORS:
      JMP FDF0 (COUT1)
237 B8        PLA
238           JMP  COUT1
239 * SINON: SI ACCUMULATEUR
      A<>RETURN OU
240 * POSITION HORIZONTALE
      DU CURSEUR < 39
241 * ALORS:
      JMP FDF0  (COUT1)
242 ADR26     PLA
243           PHA
244           CMP  #$8D
245           BEQ  ADR27
246           LDA  $24
247           CMP  #$27
248           BCC  B8
249 * SINON:
250 * IMPRIMER LE MESSAGE
      EN BAS DE L'ECRAN.
251 ADR27     LDY  #$00
252 B9        LDA  CH,Y
253           STA  $07D0,Y
254           INY
255           CPY  #$27
256           BNE  B9
257 * ENTREE D'UN
      CARACTERE AU CLAVIER
258           JSR  RDKEY
259           CMP  #$9B
260           BNE  ADR28
261 *  SI "ESC"
262 * ALORS: CONTROLE DE
      F9.FA PUIS &RETURN
263           LDA  $F9
264           CLC
265           ADC  $FA
266           BEQ  ADR29
267           PLA
268           STA  $06
269           LDA  #$BD
270           STA  $36
271           LDA  #$9E
272           STA  $37
273           JSR  LAB2
274           JMP  B6
275 ADR29     JSR  BELL
276 * SINON:
      EFFACER LE MESSAGE.
277 ADR28     LDY  #$00
278           LDA  #$A0
279 B10       STA  $07D0,Y
280           INY
281           CPY  #$27
282           BNE  B10
283 * REMONTER LE TEXTE
      DE 16 LIGNES.
284           LDY  #$00
285 B11       STY  $06
286           JSR  SCROLL
287           JSR  VP
288           LDY  $06
289           INY
290           CPY  #$0F
291           BNE  B11
292 * FAIRE &VAL PUIS
      IMPRIMER LE
      CARACTERE.
293           JSR  ADR30
294           PLA
295           LDX  $07
296           LDY  $08
297           JMP  COUT1
298 * MESSAGE EN
      MODE INVERSE:
299 * "POUR CONTINUER
      TAPER SUR UNE
      TOUCHE."
300 CH        HEX  2020100F1
      51220030F0E1409
301           HEX  0E1505122
      0140110051220
302           HEX  131512201
      50E0520140F1503
303           HEX  08052E202
      0A0A0A0A0A0
304 ADR18     BRK
305           LDA  #$10
306           STA  ADR31+1
307           LDA  #$1F
308           STA  ADR32
309           LDX  #$00
310 B12       INX
311           DEC  ADR31+1
312           INC  ADR32
313           TXA
314           PHA
315           JSR  ADR4
316           PLA
317           TAX
318           CPX  #$10
319           BNE  B12
320           CLC
321           BCC  ADR33
322 ADR4      LDY  #$C8
323           LDA  #$94
324           JSR  $03D9
325           CLD
326           RTS
327 ADR8      ORA  ($60,X)
328           ORA  ($00,X)
329 ADR31     ORA  ($00),Y
330 ADR3      CMP  $0094,Y
331 ADR32     HEX  2F
332           BRK
333           BRK
334           ORA  ($00,X)
335           INC  $0160,X
336 ADR7      BRK
337           ORA  ($EF,X)
338           CLD
339 ADR33     LDA  #$00
340           STA  $1D
341           LDA  #$30
342           STA  $1E
343           LDA  #$00
344           STA  $18
345 ADR43     LDA  #$20
346           STA  $1A
347 B14       LDA  #$0B
348           STA  $19
349           LDX  #$00
350 B13       LDY  #$00
351           LDA  ($19),Y
352           CMP  #$00
353           BEQ  ADR34
354           CMP  #$FF
355           BEQ  ADR35
356           LDY  #$02
357           LDA  ($19),Y
358           CMP  #$00
359           BCS  ADR36
360 ADR35     INX
361           CPX  #$07
362           BEQ  ADR37
363           CLC
364           LDA  $19
365           ADC  #$23
366           STA  $19
367           BCC  B13
368 ADR37     CLC
369           LDA  $1A
370           ADC  #$01
371           STA  $1A
372           BCC  B14
373 ADR34     LDY  #$02
374           LDA  #$00
375           STA  ($69),Y
376           INY
377           LDA  $18
378           STA  ($69),Y
379           RTS
380 ADR36     INC  $18
381           LDA  $19
382           STA  $1B
383           LDA  $1A
384           STA  $1C
385           CLC
386           BCC  ADR38
387 B16       LDA  #$0B
388           STA  $19
389           LDX  #$00
390 B15       LDY  #$00
391           LDA  ($19),Y
392           CMP  #$00
393           BEQ  ADR39
394           CMP  #$FF
395           BEQ  ADR38
396           LDY  #$02
397           LDA  ($19),Y
398           CMP  #$00
399           BCS  ADR45
400 ADR38     INX
401           CPX  #$07
402           BEQ  ADR40
403           CLC
404           LDA  $19
405           ADC  #$23
406           STA  $19
407           BCC  B15
408 ADR40     CLC
409           LDA  $1A
410           ADC  #$01
411           STA  $1A
412           BCC  B16
413 ADR45     LDY  #$03
414 B17       LDA  ($19),Y
415           CMP  ($1B),Y
416           BCC  ADR41
417           BNE  ADR38
418           INY
419           CPY  #$20
420           BNE  B17
421 ADR41     LDA  $19
422           STA  $1B
423           LDA  $1A
424           STA  $1C
425           CLC
426           BCC  ADR38
427 ADR39     LDY  #$01
428           LDA  $1B
429           CLC
430           ADC  #$03
431           STA  ($1D),Y
432           INY
433           LDA  $1C
434           STA  ($1D),Y
435           LDY  #$20
436 B18       LDA  ($1B),Y
437           CMP  #$A0
438           BNE  ADR42
439           DEY
440           CPY  #$03
441           BNE  B18
442 ADR42     TYA
443           TAX
444           SEC
445           SBC  #$02
446           LDY  #$00
447           STA  ($1D),Y
448           TXA
449           TAY
450 B19       LDA  ($1B),Y
451           AND  #$7F
452           STA  ($1B),Y
453           DEY
454           CPY  #$02
455           BNE  B19
456           CLC
457           LDA  $1D
458           ADC  #$03
459           STA  $1D
460           LDA  $1E
461           ADC  #$00
462           STA  $1E
463           LDY  #$00
464           LDA  #$FF
465           STA  ($1B),Y
466 ADR19     JMP  ADR43
467           JSR  FRMNUM
468           JSR  CONINT
469           CLC
470           LDA  $69
471           ADC  #$09
472           STA  $19
473           LDA  $6A
```

```
474             ADC  #$00
475             STA  $1A
476             LDA  #$FD
477             STA  $1B
478             LDA  #$2F
479             STA  $1C
480 B20         CLC
481             LDA  $1B
482             ADC  #$03
483             STA  $1B
484             LDA  $1C
485             ADC  #$00
486             STA  $1C
487             DEX
488             BNE  B20
489             LDY  #$03
490 B21         DEY
491             LDA  ($1B),Y
492             STA  ($19),Y
493             CPY  #$00
494             BNE  B21
495             RTS
```

Récapitulation 'BIBLI.AMPER'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE BIBLI.AMPER,A$92B3,L$34D

```
92B3- A2 02 BD DC 92
92B8- 9D F5 03 CA 10 F7 A2 01
92C0- BD E0 92 9D CE 94 CA 10
92C8- F7 A2 00 BD E3 92 9D C0
92D0- 94 E8 BD E3 92 E8 9D C0
92D8- 94 4C DF 93 4C E5 92 4C
92E0- D9 94 4C C8 94 A2 00 86
92E8- 19 A0 00 BD 1C 93 F0 1A
92F0- C9 FF F0 08 D1 B8 D0 07
92F8- C8 E8 D0 EF 4C C9 DE E8
9300- BD 1C 93 D0 FA E8 E6 19
9308- D0 DF 20 98 D9 06 19 06
9310- 19 A6 19 BD 3A 93 48 BD
9318- 39 93 48 60 A8 00 E5 00
9320- B1 00 BD 00 53 43 52 4F
9328- 4C 4C 85 00 52 57 54 53
9330- 00 87 4E 4D 24 CF 00 FF
9338- 20 52 93 EA 20 64 93 EA
9340- 20 91 93 EA 20 DE 93 EA
9348- 20 EC 93 EA 20 9D 94 EA
9350- 20 C8 95 A5 75 85 CE A5
9358- 76 85 CF 68 A8 68 A2 FF
9360- 9A 48 98 48 60 A5 CE 18
9368- 65 CF D0 01 60 A5 CE C5
9370- F9 D0 07 A5 CF C5 FA D0
9378- 01 60 A2 03 B5 F9 95 FB
9380- CA 10 F9 A5 CE 85 F9 A5
9388- CF 85 FA A9 00 85 CE 85
9390- CF 60 A5 F9 18 65 FA D0
9398- 01 60 A5 F9 85 50 A5 FA
93A0- 85 51 A2 00 B5 FB 95 F9
93A8- E8 E0 04 90 F7 A5 CE C5
93B0- 50 D0 06 A5 CF C5 51 F0
93B8- D9 A9 00 85 FD 85 FE A9
93C0- FF 85 32 A9 F0 85 36 A9
93C8- FD 85 37 20 39 FB 20 51
93D0- A8 20 1A D6 90 03 4C 41
93D8- D9 A2 5A 4C 12 D4 60 A9
93E0- 00 A2 05 95 F9 CA 10 FB
93E8- 85 CE 85 CF 60 AD FC 93
93F0- 85 36 AD FD 93 85 37 20
93F8- 51 A8 60 20 FE 93 86 07
9400- 84 08 48 A5 25 C9 16 F0
9408- 04 68 4C F0 FD 68 48 C9
9410- 8D F0 06 A5 24 C9 27 90
9418- F0 A0 00 B9 70 94 99 D0
9420- 07 C8 C0 27 D0 F5 20 0C
9428- FD C9 9B D0 1B A5 F9 18
9430- 65 FA F0 11 68 85 06 A9
9438- BD 85 36 A9 9E 85 37 20
9440- 51 A8 4C 92 93 20 3A FF
9448- A0 00 A9 A0 99 D0 07 C8
9450- C0 27 D0 F8 A0 00 84 06
9458- 20 70 FC 20 1A FC A4 06
9460- C8 C0 0F D0 F1 20 65 93
9468- 68 A6 07 A4 08 4C F0 FD
9470- 20 20 10 0F 15 12 20 03
9478- 0F 0E 14 09 0E 15 05 12
9480- 20 14 01 10 05 12 20 13
9488- 15 12 20 15 0E 05 20 14
9490- 0F 15 03 08 05 2E 20 20
9498- A0 A0 A0 A0 A0 00 A9 10
94A0- 8D CD 94 A9 1F 8D D1 94
94A8- A2 00 E8 CE CD 94 EE D1
94B0- 94 8A 48 20 BF 94 68 AA
94B8- E0 10 D0 EE 18 90 1E A0
94C0- C8 A9 94 20 D9 03 D8 60
94C8- 01 60 01 00 11 00 D9 94
94D0- 00 2F 00 00 01 00 FE 60
94D8- 01 00 01 EF D8 A9 00 85
94E0- 1D A9 30 85 1E A9 00 85
94E8- 18 A9 20 85 1A A9 0B 85
94F0- 19 A2 00 A0 00 B1 19 C9
94F8- 00 F0 23 C9 FF F0 08 A0
9500- 02 B1 19 C9 00 B0 23 E8
9508- E0 07 F0 09 18 A5 19 69
9510- 23 85 19 90 DE 18 A5 1A
9518- 69 01 85 1A 90 CF A0 02
9520- A9 00 91 69 C8 A5 18 91
9528- 69 60 E6 18 A5 19 85 1B
9530- A5 1A 85 1C 18 90 1A A9
9538- 0B 85 19 A2 00 A0 00 B1
9540- 19 C9 00 F0 3D C9 FF F0
9548- 08 A0 02 B1 19 C9 00 B0
9550- 17 E8 E0 07 F0 09 18 A5
9558- 19 69 23 85 19 90 DE 18
9560- A5 1A 69 01 85 1A 90 CF
9568- A0 03 B1 19 D1 1B 90 07
9570- D0 DF C8 C0 20 D0 F3 A5
9578- 19 85 1B A5 1A 85 1C 18
9580- 90 CF A0 01 A5 1B 18 69
9588- 03 91 1D C8 A5 1C 91 1D
9590- A0 20 B1 1B C9 A0 D0 05
9598- 88 C0 03 D0 F5 98 AA 38
95A0- E9 02 A0 00 91 1D 8A A8
95A8- B1 1B 29 7F 91 1B 88 C0
95B0- 02 D0 F5 18 A5 1D 69 03
95B8- 85 1D A5 1E 69 00 85 1E
95C0- A0 00 A9 FF 91 1B 4C E9
95C8- 94 20 67 DD 20 FB E6 18
95D0- A5 69 69 09 85 19 A5 6A
95D8- 69 00 85 1A A9 FD 85 1B
95E0- A9 2F 85 1C 18 A5 1B 69
95E8- 03 85 1B A5 1C 69 00 85
95F0- 1C CA D0 F0 A0 03 88 B1
95F8- 1B 91 19 C0 00 D0 F7 60
```

Patches au DOS 3.3

Patrice Neveu

Les petites améliorations au DOS 3.3 proposées ici sont les suivantes :

- obtention du CATALOG simplement par le slash (/). Nous devons être nombreux à frapper CATALOF ou à oublier des lettres en voulant taper trop vite.
- lancement des programmes Applesoft, Binaires ou fichiers Exec par l'unique commande '-'. Les fichiers INTEGER ne peuvent plus être utilisés car les patches occupent les emplacements de FP et INT.
- recherche du fichier désiré sur les deux lecteurs sans avoir à spécifier D1 ou D2 chaque fois qu'il est nécessaire. Bien-sûr, si le fichier n'existe sur aucun des deux lecteurs, le message FILE NOT FOUND apparaîtra comme d'habitude. En cas de VOLUME MISMATCH ou de I/O ERROR, le patch cherchera également sur l'autre lecteur.

Pour bénéficier de ces nouvelles commandes à chaque fois que vous booterez votre disquette, il suffira de taper 'BRUN DOSPATCH' puis 'INIT HELLO' comme pour initialiser une disquette avec un DOS normal.

Les routines utilisées

L'organisaton interne des tables du DOS

Les tables tout d'abord : en ce qui concerne les mots-clé du DOS tels que CATALOG, LOAD etc., elles sont au nombre de trois.

1) La liste des mots-clé ($A884.A908). Cette table contient tous leurs caractères ASCII. La dernière lettre de chaque commande a son bit 7 à 1. Ainsi, INIT est code $49 $4E $49 $D4

2) Juste après ($A909.A940) vient une table qui indique à l'aide de deux octets par commande quelles sont les modalités d'utilisation de chacune d'entre-elles.
Chaque octet étant constitué de 8 bits, nous avons donc 16 bits au total à prendre en compte. En considérant que dans $8000 c'est le bit 0 qui est à 1 (pour une fois on compte de gauche à droite !) on obtient la table rappelée en encadré.

3) La 3ème table ($9D1E.9D55) contient les points d'entrée de chaque commande, soit deux octets par commande. On apprend ainsi que la routine INIT commence en $A54F.

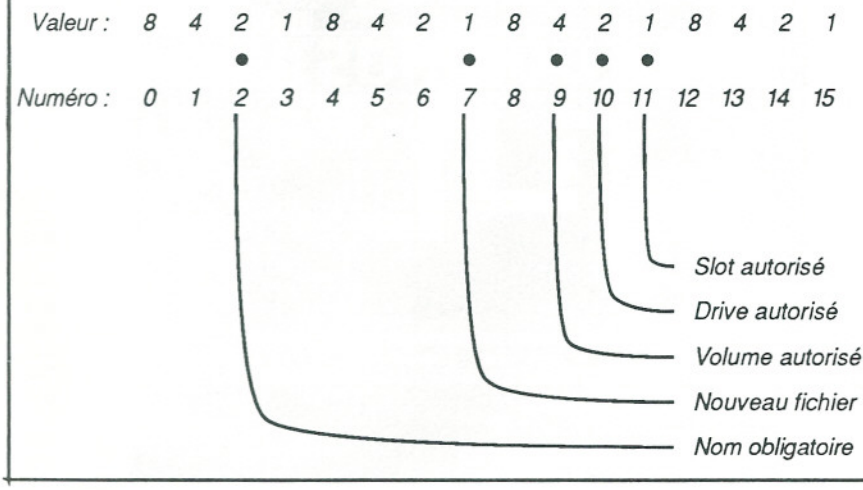
Avec ces tables il devient possible de modifier, rajouter ou enlever des commandes au DOS 3.3. Nous possédons en effet tous les éléments : la syntaxe des commandes, leurs modalités d'utilisation et les adresses de leurs routines.

Table d'utilisation des commandes

Bit	*Signification*
0	Le nom de fichier est optionnel
1	Il n'y a pas de paramètre de position (Catalog, Mon...)
2	Premier nom de fichier attendu
3	Deuxième nom de fichier attendu (Rename)
4	Numéro de slot indispensable (PR#, IN#)
5	Valeur de Maxfiles attendue comme opérande
6	Commande accessible uniquement par programme
7	Commande créant un nouveau fichier s'il n'existe pas
8	C, I, O, sont autorisés
9	V (Volume) est autorisé
10	D (Drive) est autorisé
11	S (Slot) est autorisé
12	L (Length) est autorisé
13	R (Range) est autorisé
14	B (Byte) est autorisé
15	A (Adress) est autorisé

Exemple :

Pour l'ordre INIT, on obtient 2070 :



Nous allons donc voir comme exemple le cas du remplacement de FP par le signe '-'. Au sujet de la syntaxe, nous avions en $A8EF:

FPINTBSAVEBLOADBRUNVERIFY

que l'on doit remplacer par $A8EF:

-INTBSAVEBLOADBRUNVERIFY0

(les caractères gras représentent ceux dont le bit de poids fort est à 1).

Pour plus de simplicité, la routine de Run Universel sera implantée à la place de celle de FP ($A57A.A59D) ; il n'est donc pas nécessaire de changer la 3ème table. Par contre, la seconde où sont les paramètres à utiliser , ne correspond plus ; il y avait $4070 et il faut $A075 (se reporter au tableau de signification des bits).

Et voilà, il ne reste plus qu'à écrire la routine qui sera exécutée à chaque '-'. Elle commence par ouvrir le fichier dont le nom suit ($A2A8). Ensuite, il faut connaître le type du fichier que l'on vient d'ouvrir. Ce renseignement est stocké en $B5C2 de cette manière : 0 = Text ; 2 = Applesoft ; 4 = Binaire. Selon le résultat trouvé on saute faire un BRUN ($A38E), un RUN($A4D1) ou un EXEC ($A5C6).

En ce qui concerne la recherche d'un fichier sur les deux lecteurs, la mise en place est plus délicate car on doit se brancher directement sur la routine de traitement des erreurs.

Voici donc les grandes lignes de son fonctionnement : dès qu'une erreur se produit, le DOS saute en $A6D5. Nous allons donc y placer un saut vers notre routine BIDRIVE implantée en $A594 à la place de INT. Celle-ci va lire le numéro de l'erreur ($B5C5). Si ce n'est pas le code de FILE NOT FOUND (6), ni de VOLUME MISMATCH (7) ou de I/O ERROR (8) alors on revient à la routine ERRHANDLE. Sinon, on change de lecteur puis on vérifie si un changement a déjà été effectué juste avant. Si oui, le fichier n'est pas dans un des lecteurs et on va alors afficher l'erreur en repassant par ERRHANDLE. Autrement, on lance la commande demandée par un saut en DOCMD ($A180).

Ce changement de lecteur fonctionne donc sur tous les ordres accompagnés d'un nom de fichier et ces patches peuvent être rajoutés non seulement au DOS 3.3 mais aussi au DIVERSI-DOS, et généralement sur tous les DOS 3.3 "améliorés".

Récapitulation 'DOSPATCH'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE DOSPATCH,A$2000,L$C5

```
2000- A9 A9 A2 00 8D 8D B7 8E
2008- 8E B7 A9 85 A2 4A 8D 8F
2010- B7 8E 90 B7 A9 60 8D 91
2018- B7 A9 0D 8D AC A6 A9 20
2020- A2 8D A0 B7 8D BA A6 8E
2028- BB A6 8C BC A6 A9 60 8D
2030- BD A6 A9 20 A2 94 A0 A5
2038- 8D D5 A6 8E D6 A6 8C D7
2040- A6 A0 15 B9 78 20 99 EF
2048- A8 88 10 F7 A2 A0 A0 75
2050- 8E 35 A9 8C 36 A9 A9 40
2058- A0 70 8D 37 A9 8C 38 A9
2060- AD 3E 9D 8D 4C 9D AD 3F
2068- 9D 8D 4D 9D A0 36 B9 8E
2070- 20 99 7A A5 88 10 F7 60
2078- AD AF 42 53 41 56 C5 42
2080- 4C 4F 41 C4 42 52 55 CE
2088- 56 45 52 49 46 D9 20 A8
2090- A2 AD C2 B5 29 7F F0 0A
2098- C9 02 F0 03 4C 8E A3 4C
20A0- D1 A4 4C C6 A5 AD C5 B5
20A8- C9 09 B0 18 C9 06 90 14
20B0- A9 03 ED 68 AA 8D 68 AA
20B8- A5 4A D0 05 E6 4A 4C 80
20C0- A1 4C 8D B7 60
```


Comment faire ?

Mettre au point ce programme était délicat, l'utiliser est particulièrement simple.

- Faire BRUN DOSPATCH, le DOS est alors modifié en mémoire ;
- Initialiser une disquette (INIT HELLO par exemple) à l'aide de ce DOS nouveau style.

Maintenant en 'bootant' sur la disquette ainsi préparée, vous disposez du *smart run* de type ProDOS, du CATALOG par le '/', mais surtout, inutile maintenant de préciser ",D1" ou ",D2"...

Source 'DOSPATCH.S'
Assembleur Big Mac

```
 1
 2          LST OFF
 3
 4 *********************************
 5 * PATCHES AU DOS 3.3             *
 6 *                               *
 7 * 1) / PERMET LE CATALOG.       *
 8 * 2) - LANCE LES FICHIERS A/B/T *
 9 * 3) FP ET INT SONT SUPPRIMES   *
10 * 4) RECHERCHE SUR 2 DRIVES LE  *
11 *    FICHIER & VOLUME DEMANDES  *
12 *                               *
13 ****PATRICE*NEVEU**LE*8/7/85*****
14
15          ORG $2000
16          OBJ $8000
17
18
19 DRIVESW   EQU $4A          = 1 APRES CHANGEMENT DE DR
                              IVE
20
21 DOSADR17  EQU $9D3E        ADRESSE DE CATALOG
22 DOSADR23  EQU $9D4C        REMPLACE CELLE DE FP
23 DOSORDRES EQU $A8EF        PARTIE DE LA TABLE DES COM
                              MANDES DOS.
24 TYPEFOUND EQU $B5C2        VOLUME DU FICHIER OUVERT
25 ERRNUM    EQU $B5C5        CODE D'ERREUR
26
27 CLOSEALL  EQU $A316        FERME TS FICHIERS OUVERTS
28 OPEN      EQU $A2A8        ROUTINE OUVERTURE FICHIER
29
30 DOCMD     EQU $A180        EXECUTE LA COMMANDE DOS
31 PATCH     EQU $A57A        ADRESSE DU PATCH DE RUN UN
                              IVERSEL
32 PATCH1    EQU $A594        ADRESSE DU BIDRIVE
33 PATCH2    EQU $B78D        ROUTINE FIXANT DRIVESW = 0
34 BRUNHAND  EQU $A38E        FAIT UN BRUN
35 RUNHAND   EQU $A4D1                RUN
36 EXECHAND  EQU $A5C6                EXEC
37 GOODOPEN  EQU $A6BA        VA ICI LORSQUE LE FICHIER
                              A ETE OUVERT
38 ERRHANDLE EQU $A6D5        EST LANCE LORS D'ERREURS
39 DRIVENUM  EQU $AA68        =1 SI DRIVE 1
                              =2 SI DRIVE 2 BRANCHE
40 PATCHDOS  EQU $BFE6        PATCH DES DERNIERS DOS 3.3
                               (JANVIER 83)
41
42 *-------------------------------
43 * ROUTINE ANNULANT DRIVESW
44 *-------------------------------
45
46          LDA #$A9          PATCH2  LDA #0
47          LDX #$0                   STA DRIVESW
48          STA PATCH2                RTS
49          STX PATCH2+1
50          LDA #$85
51          LDX #DRIVESW
52          STA PATCH2+2
53          STX PATCH2+3
54          LDA #$60
55          STA PATCH2+4
56
57 *-------------------------------
58 * SI LE FICHIER EST TROUVE, ALORS
59 * DRIVESW = 0.
60 *-------------------------------
61
62          LDA #$0D          LE BCC EST AVANCE DE 9 OCT
63          STA $A6AC         SI PAS D'ERREURS.
64
65          LDA #$20          IL VA ICI, OU 4 OCTETS ETA
                              IENT LIBRES
66          LDX #<PATCH2      MAINTENANT, ON SAUTE A PAT
                              CH2, C'EST A
67          LDY #>PATCH2      DIRE QU'ON FAIT DRIVESW=0
                              LORSQU'UNE
68          STA GOODOPEN      COMMANDE DOS A ETE EFFECTU
                              EE SANS
69          STX GOODOPEN+1    ERREUR.
70          STY GOODOPEN+2
71          LDA #$60          ET ON CONTINUE LE TRAITEME
                              NT NORMAL
72          STA GOODOPEN+3
73
74 *-------------------------------
75 * ERREUR --> CHANGE DE DRIVE
76 *-------------------------------
77
78          LDA #$20          $20 EST LE CODE DE 'JMP'
79          LDX #<PATCH1      ON PATCHE LE TRAITEMENT DE
                              S ERREURS:
80          LDY #>PATCH1      AVANT DE FAIRE COMME D'HAB
                              ITUDE, ON
81          STA ERRHANDLE     VERIFIE SI C'EST CODE6 (FI
                              LE NOT FOUND)
82          STX ERRHANDLE+1   CAR DANS CE CAS, TOUT N'ES
                              T PAS PERDU !
83          STY ERRHANDLE+2
84
85 *-------------------------------
86 * CHANGE LA TABLE DES MOT-CLES
87 *-------------------------------
88
89          LDY #MOTSEND-MOTSDOS CAR DES MOT-CLES ONT ETE
                                  RAJOUTES ET
90 CHNGMOTS  LDA MOTSDOS,Y       D'AUTRES ENLEVES:
91          STA DOSORDRES,Y   / - RAJOUTES, FP INT VERIF
                              Y ENLEVES
92          DEY
93          BPL CHNGMOTS
94
95 *-------------------------------
96 * CHANGE LA TABLE DES MOT-CLES
97 * POSSIBLES ($A909.A940)
98 *-------------------------------
99
100         LDX #$A0          LE RUN UNIVERSEL PEU ETRE
                              SUIVI
101         LDY #$75          DES MOTS: A (BRUN), R (EXE
                              C), S,D,V.
102         STX $A935         D'UN NOM OBLIGATOIRE (BRUN
                              /EXEC)
103         STY $A936         OU FACULTATIF (RUN)
104
105         LDA #$40          LE SLASH REPOND AUX MEMES
                              PARAMETRES
106         LDY #$70          QUE LE CATALOG.
107         STA $A937
108         STY $A938
109
110 *-------------------------------
111 * SIGNALE QUE / EST UN CATALOG
112 *-------------------------------
113
114         LDA DOSADR17
115         STA DOSADR23
116         LDA DOSADR17+1
117         STA DOSADR23+1
118
119 *-------------------------------
120 * INSTALLATION DE LA ROUTINE DE
121 * RUN UNIVERSEL ET BIDRIVE
122 *-------------------------------
123
124         LDY #BIDRIVEND-UNIRUN CALCUL DE LA LONGUEUR D
                                  E LA ROUTINE
125 INSTAL   LDA UNIRUN,Y        POUR POUVOIR LA TRANSFER
                              ER...
126         STA PATCH,Y
127         DEY
128         BPL INSTAL
129
130         RTS               LE DOS EST MAINTENANT MODI
                              FIE.
131
132 *-------------------------------
133
134 MOTSDOS  ASC "-"           LES MOT-CLES SONT CODES
                              AINSI:
135         ASC "/"           LES PREMIERS CODES ASCII
                              AVEC BIT 8 = 0
136         ASC 'BSAV'        PUIS LE DERNIER AVEC BIT 8
                              = 1
137         ASC "E"
138         ASC 'BLOA'
139         ASC "D"
140         ASC 'BRU'
141         ASC "N"
142         ASC 'VERIF'
143 MOTSEND  ASC "Y"
144
145 *-------------------------------
146 * ROUTINE 3 EN 1
147 *-------------------------------
148
149 UNIRUN   JSR OPEN          ON VA OUVRIR CE QUI NOUS I
                              NTERESSE
150         LDA TYPEFOUND     ON REGARDE LE TYPE DU FICH
                              IER
151         AND #$7F          ON LE DE-LOCK (LOCK --> BI
                              T 8 = 1)
152         BEQ EXEC          CODE0 = TEXT
153         CMP #2            CODE2 = APPLESOFT
154         BEQ RUN
```

Suite de 'DOSPATCH.S' page 72 ⇒

Courrier des Lecteurs

J'ai essayé sans succès de faire tourner le programme ORDRALPHABETIX (Pom's 13). Lorsque je le 'RUNe', les messages «Must be linked» puis «Proc cours indefined» s'affichent. Que faire ?

Monsieur Charles IZARD, 11400 CASTELNAUDARY

Pour créer le module exécutable ORDRALPHA.CODE, il est nécessaire d'effectuer les opérations suivantes :

- saisir les sources avec l'éditeur Pascal, ou utiliser le programme Basic->Pascal de la disquette Pom's pour transférer les fichiers du DOS 3.3 vers le Pascal. Vous devez alors obtenir les sources ORD1.TEXT et ORD2.TEXT sur un volume appelé, par exemple, 'TEST1:' ;
- compiler ORD1.TEXT pour obtenir ORD1.CODE (ORD1.TEXT utilise le SYSTEM. LIBRARY) ;
- assembler ORD2.TEXT pour obtenir ORD2.CODE ;
- 'linker' les deux modules en tapant successivement depuis le niveau global du Pascal :

```
L
(* pour appeler le Linker *)
TEST1:ORD1.CODE
(* au message "Host file?" *)
TEST1:ORD2
(* au message "Lib file?" *)
:SYSTEM.LIBRARY
<répondre par un RETURN>
N
(* au message "Map file?" *)
TEST1:ORDRALPHA.CODE
(* au message "Output file?" *)
```

Le fichier est alors directement utilisable par l'option X (Execute) du niveau global du Pascal 1.1. Attention : le programme assembleur n'est pas compatible avec le Pascal 1.2 et devrait être modifié en conséquence.

Possesseur d'un 'vieil' Apple //e, j'ai acquis le kit de mise à jour 65C02 dès sa sortie. Je me trouvais donc à la tête d'un nouveau //e, une sorte de super //c avec des slots.
Premier choc avec Locksmith 5.0 : il ne fonctionne plus. Second choc : le logiciel de la carte EVE également. Troisième choc : le passage en 80 colonnes devient risqué (perte aléatoire du programme en cours). La carte EVE accusée des problèmes est promptement remplacée par une Féline ; miracle, Locksmith et Purplesoft fonctionne à nouveau.

Questions techniques : Comment savoir si la touche CTRL est enfoncée (seule) ? idem pour SHIFT ? et CAPS LOCK ? et l'inverseur français/ américain.

Monsieur Thierry MECHAIN, 17300 ROCHEFORT

Merci pour votre expérience. Le chip clavier du //e n'envoie pas d'information sur le bus lorsque les touches CTRL ou SHIFT sont enfoncées seules. Il n'est donc pas possible d'en déterminer l'état par logiciel. Il en va de même pour CAPS-LOCK et l'inverseur de jeux de caractères.

La 36 ème piste, suite...

Il existe une autre solution, pratique et rapide, pour obtenir la 36ème piste : il suffit de demander au DOS qu'il formate correctement la disquette. Un petit patch suffit :

```
CALL -151
BEFE: 24   [au lieu de 23]
  (nombre de pistes)
B3EF: 24   [au lieu de 23]
AEB5: 90   [au lieu de 8C]
  (boucle de libération des
  pistes)
INIT HELLO
```

Et voilà ! De plus, le DOS contenu sur cette nouvelle disquette permet lui-aussi un formatage ultérieur en 36 pistes.

(N.D.L.R. : nous avons précisé entre crochets les anciennes valeurs contenues dans les adresses indiquées ; le lecteur devra les contrôler afin de ne pas modifier un DOS déjà patché ou un ProDOS réassemblé pour une nouvelle version)

A ce propos, un certain nombre d'autres lecteurs de disquettes compatibles permettent un formatage sur les 40 pistes d'une disquette 5 pouces 1/4 (le drive Chinon, par exemple). Il est donc tentant de faire la même expérience :

```
BEFE: 28
B3EF: 28
AEB5: A0
```

Le problème se pose différemment sous ProDOS: celui-ci est capable de gérer tout type de mémoire de masse, mais il contient également le programme d'accès aux Disk][.

Avec la version 1.0, la réponse se trouve dans Beneath Apple Prodos, mais il y a un léger bug, il faut taper :

```
UNLOCK PRODOS
BLOAD PRODOS, TSYS,
  A$2000
CALL-151
520D: 40   [au lieu de 18]
  ($118 à $140 blocs au total)
BSAVE PRODOS, TSYS,
  A$2000
```

Le FILER permet de formater une disquette et doit donc, lui-aussi, être modifié :

```
UNLOCK FILER
BLOAD FILER, TSYS,
  A$2000
4244: 40   [au lieu de 18]
79F4: 28   [au lieu de 23]
BSAVE FILER, TSYS,
  A$2000
```

Les routines s'étant déplacées dans la nouvelle version PRODOS 1.1.1, les modifications sont à faire ainsi :

Pour ProDOS :

```
56E3: 40
```

Pour le Filer

```
42A1: 40
7AF4: 28
```

Enfin, deux octets du CONVERT qui intéresseront les possesseurs de disquettes DOS 3.3 en format non standard :
30E4 contient $11
30EB contient $0F, c'est-à-dire l'adresse physique du début du catalogue.

En remplaçant les valeurs d'origine comme ci-dessous :

```
30F9: 29
30FA: 10
```

CONVERT sera en mesure de lire toute disquette DOS 3.3 formatée en 35, 36 ou 40 pistes.

Jacques REY, 77000 Melun

Micro-informations

Jean-Michel Gourévitch

Joli feu d'artifice de début d'année. Avec l'introduction du Macintosh Plus, Apple a tiré quelques solides cartouches. Nanti d'une mémoire augmentée, d'un clavier élargi et d'une interface pour périphéri- ques à transfert rapide, le nouveau Macintosh s'implantera- t-il dans les entreprises ? C'est ce que l'on souhaite à Cupertino, où l'objectif avoué de John Sculley est d'augmenter les ventes de Macintosh de 30%.

Compatibilité ?

En attendant, quelques possesseurs de Macintosh traditionnels se font des cheveux : leurs programmes et ustensiles favoris fonctionneront-ils sur le Macintosh Plus ? Pourquoi faut-il toujours que le progrès passe par des sacrifices ? En effet, après une période "sèche" en matière de logiciels, le Macintosh était arrivé à une sorte de maturité et les utilisateurs avaient pris leurs habitudes : dans le menu "Pomme", des accessoires de bureau bien utiles (comme *MacTracks*, un éditeur de macro commandes permettant de déclencher d'un doigt l'impression d'un document), reliés par les prises DB9 des accessoires comme le numériseur d'images *Thunderscan*. Las! Voici par exemple un logiciel et un accessoire qui ne fonctionnent plus sur le Macintosh Plus. Car, dans certains cas, le Macintosh Plus sera un "Macintosh Moins". En cause, pour le logiciel, le nouveau système d'organisation hiérarchique des fichiers : le *HFS* (Hierarchical File System). A cause de lui, de nombreux utilitaires et notamment *MacTracks* ne fonctionnent plus du tout sur le Plus. Tout comme les programmes *MacLion*, *MacExpert* et *Through the looking glass*. Toute une série d'autres exigent qu'on leur conserve leur système d'origine et qu'on n'y installe surtout pas le nouveau système avec le Finder 5.1 (comme *Crunch*, *Expert Lisp*, *Helix*, *Mac Publisher*, *Smooth Talker*, le *Macintosh Development System* - c'est un comble! - etc). Enfin, d'autres (comme le *Switcher*, *PageMaker*, *TK Solver*, *TéléMac*, *Omnis3*, *MacDraft*, etc) exigent que les fichiers ou fichier d'aide, ou applications (dans le cas du Switcher) se trouvent dans le même dossier.

Plus de mémoire

Les ennuis ne se bornent pas qu'au logiciel. L'utilisation de prises "Mini Din" non standardisées pour relier au Macintosh Plus l'imprimante ou le modem ont provoqué en février un vent de panique : personne n'en disposait, et les Macintosh Plus ne pouvaient imprimer. Et pas possible de bricoler un câble, car ces prises sont introuvables au magasin d'électronique du coin. Autre problème : ces prises ont été modifiées. L'un des contacts de la prise permettait d'extraire du Macintosh du courant en 5 volts, ce qui alimentait notamment le numériseur *Thunderscan*. Conclusion : même avec le nouveau câble, les accessoires comme le *Thunderscan* ne fonctionneront pas, il faudra leur ajouter une alimentation extérieure. Mais on pardonne tout avec la stupéfiante amélioration des performances du Macintosh Plus (qui sera dès l'été extensible à 4 Mégas de mémoire vive avec les nouveaux boîtiers mémoire de 1 Méga). Et aussi grâce aux prix peu élevés appliqués par Apple France pour la transformation des Mac existants en Macintosh Plus. Tout ceci suffira-t-il à assurer les ventes d'Apple ?

Le futur Mac modulaire

Il ne faudrait pas s'imaginer que la firme à la pomme entend désormais se reposer sur ses lauriers. Les efforts des ingénieurs de Cupertino se portent maintenant sur deux fronts : celui du Macintosh modulaire et celui de l'Apple //.
Un nouveau Mac en trois morceaux (clavier, unité centrale et moniteur) pourrait être présenté soit à la fin de l'été, soit en début 1987. Il comporterait le processeur de la série 68000 baptisé 68020 (le grand frère de celui du Mac). Il aurait surtout la particularité d'être une machine "ouverte". Acceptant ainsi des cartes d'extension, des disques durs internes etc.

Côté Apple //, on parle désormais de deux produits. L'un, de bas de gamme, serait un benjamin de l'Apple //e vendu 500 dollars ou moins. Le second serait le fameux Apple // équipé d'un processeur 16 bits : c'est le vaisseau fantôme de la micro informatique, on en parle depuis au moins deux ans, mais il finira bien par arriver. Rendez-vous peut être à l'automne.

Les efforts d'Apple vont surtout viser désormais trois directions : l'harmonisation de ses deux gammes de produits, la communication et le perfectionnement de l'édition électronique. Avec deux buts : assurer la cohérence des produits (et la compatibilité entre Macintosh et Apple //), et tenter par le biais de la communication (et la compatibilité avec les fichiers de l'IBM PC) et de l'édition d'implanter le Macintosh Plus dans les entreprises.

Harmonisation des gammes : le programme commun

Voici donc les Apple // et les Mac dotés d'une série de périphériques communs : modems, imprimantes, lecteurs de disquettes. C'est probablement ce changement là qui est le plus significatif : le Mac et l'Apple // utilisent dorénavant les mêmes disquettes de 800 Ko. On verra prochainement un utilitaire permettant de convertir les fichiers du Mac en Prodos pour pouvoir les utiliser sur l'Apple //. Et bientôt, les disques durs seront, eux aussi, communs aux deux gammes, en attendant un Apple // capable d'émuler le fonctionnement du Mac. Dans le sens inverse, il faut remarquer l'étonnant logiciel **Mac+II** dû à un jeune auteur prodige : **David Hemmo** et distribué par **JRD**. Ce produit se compose d'un câble destiné à relier l'Apple // au Mac et de deux logiciels, l'un pour l'Apple // (transférant le contenu des ROMs de celui-ci vers le Mac), l'autre pour le Macintosh (une adaptation du DOS 3.3). A la mise en route, le contenu des ROMs de l'Apple // est transféré vers le Mac. Un programme baptisé **MacCom** pour l'Apple // et un autre appelé **MacDos** pour le Macintosh permettent de transférer sur le Mac programmes et fichiers de l'Apple //. A en croire son auteur, ce programme permet de faire tourner sur le Mac pratiquement tous les programmes de l'Apple //. Prix : 1950,00 F. Bien utile pour les nouveaux convertis au Mac qui ont encore une vaste logithèque pour l'ancêtre.

La course de vitesse est engagée : aux États-Unis, un autre logiciel, **II in a Mac de Computer Applications** permet lui aussi de simuler le fonctionnement d'un Apple //. On voit même sur l'écran du Mac les lecteurs de disquettes, le modem et le joystick. Prix : 50 dollars.

La communication

C'est un atout indispensable à la réussite du Mac Plus qui devra aussi bien pour s'imposer pouvoir s'intégrer dans des réseaux locaux que pouvoir être connecté à des gros ordinateurs. Lors de la présentation du Mac Plus, John Sculley a donc bien insisté sur les accords signés avec d'importants partenaires. Avec **Northern Telecom**, un géant américain des télécommunications, pour permettre l'utilisation des lignes ordinaires et des centraux électroniques pour connecter les Mac à d'autres ordinateurs et périphériques. Avec **General Electric Information Services** (la filiale communication de General Electrics), pour pouvoir relier le Macintosh au réseau **Dealertalk** permettant à des sociétés de relier les sièges centraux aux filiales et aux concessionnaires. Ce réseau comporte un logiciel de base de données, un bulletin électronique et du courrier électronique. Le contrat permet d'utiliser les gros ordinateurs de General Electrics.

Un autre accord lie Apple à **Dow Jones**, pour l'accès aux informations boursières. Dow Jones a ainsi développé pour le Mac Plus un logiciel : **Straight Talk** permettant l'accès automatique aux bases de données de Dow Jones

Un produit existe déjà permettant de relier le Macintosh aux IBM PC. C'est le **PC MacBridge** de **Tangent Technologies**. La carte permet en fait de relier un IBM PC à un réseau AppleTalk.

En France, **Alpha Systèmes** a développé le **Netway 1000 A**, qui remplace un contrôleur 3274, émule un terminal 3278 et dessert un réseau AppleTalk permettant à 31 Macintosh (et à 16 d'entre eux simultanément) de se connecter à un ordinateur central.

Dans les entreprises, la communication passera par des serveurs de réseau. Celui d'Apple n'est toujours pas annoncé. En attendant, voici le **3Com 3 Server** de **3Com** composé d'un disque dur de 36 Méga-octets. Au travers des réseaux Ethernet et Appletalk, on peut relier entre eux des IBM PC et des Macintosh. Prix du serveur : 8000 dollars. La revue Infoworld qui l'a testé lui a accordé une note extrêmement élevée pour ses hautes performances. Et 3Com vient d'annoncer le **DiskPlus**, un disque dur de 70 Mégas à temps d'accès ultra court (30 millisecondes) se connectant à l'interface SCSI.

Un autre accès au réseau Ethernet est possible grâce au **Fastpath** de **Kinetics** : une boîte noire s'insèrent entre le réseau AppleTalk et Ethernet. Prix : 2500 dollars pour la boîte, plus 1500 pour le logiciel **K-Talk**.

Enfin, la dernière annonce concerne un réseau pouvant se partager les **Hyperdrive** de **General Computer** importés en France par **P-Ingénierie**. L'Hyperdrive, c'est cet étonnant

disque dur (disponible en version 10 ou 20 Mégas) qui s'installe à l'intérieur du Macintosh, et est désormais adaptable au Macintosh Plus. Ce disque était jusqu'alors le plus rapide. Nos essais personnels ont ainsi chronométré dans Excel un temps de 8 secondes pour ouvrir un document (contre 18 secondes avec un lecteur de disquettes).
Désormais, un logiciel baptisé **Hypernet** permet d'utiliser un Hyperdrive comme serveur et d'y relier un maximum de 31 Mac par AppleTalk. Prix : 25900,00 F HT pour un **Hyperdrive 20** et 3750,00 F HT pour **Hypernet**.

On trouve enfin d'autres serveurs aux États-Unis chez **Infosphere** avec **MacServe** et chez **Hayes** avec l'**Interbridge**, permettant de relier entre eux plusieurs réseaux AppleTalk. Enfin, la nouvelle version de **Mac Charlie** : **Mac Charlie Plus**, permet non seulement aux Macintosh de faire tourner des logiciels IBM, mais aussi de se connecter aux ordinateurs centraux et aux réseaux IBM. Pas de doute : 1986 devient l'année des serveurs pour le Mac.

Disques durs : baisses à l'horizon 1986

Ce sera aussi pour le Macintosh l'année des disques durs. Grâce à la nouvelle interface qu'a installé Apple sur le Macintosh Plus : la **SCSI** (les initiés prononcent "scuzzy"). Cette interface normalisée permet d'assembler en chaîne jusqu'à 7 périphériques avec une haute vitesse de transfert (1,5 Méga-octets par seconde). On peut donc désormais facilement relier au Mac des lecteurs de disques optiques, des scanners, des robots et des disques durs. Avec deux avantages :

- la vitesse, car les données du disque ne passent plus par un des "lents" ports série ;
- le coût, car l'interface étant normalisée, la carte contrôleur est peu onéreuse (moins de 200 dollars) et on peut y relier des disques durs déjà utilisés pour l'IBM.

Lecteurs, voici un pari : dans six mois, les disques durs auront baissé de 50%. Et ce n'est qu'un début : songez qu'aux États-Unis, un disque dur de 20 Méga-octets pour l'IBM PC est tombé au-dessous de 400 dollars.
Voici donc déjà annoncés toute une salve de disques durs pour le Mac se connectant sur le port SCSI.
Mention spéciale au disque dur de 10 Mégas de **Lodown** : son prix, 795 dollars, en fait le meilleur marché. Suivi par le **Macnifty Personal 10** de **Macnifty**. Mention spéciale aussi à l'**AST 4000** d'**AST** : c'est le plus gros avec ses 74 Mégas pour 7000 dollars.

L'Hyperdrive voit apparaître des concurrents qui s'installent à l'intérieur du Macintosh Plus pour un prix inférieur, ce sont les **Overdrive 10** et **20** de **Levco** (1775 et 1975 dollars) ainsi que les **Micahdrive 10 AT** et **20 AT** de **Micah** (1495 et 1895 dollars). A noter encore des disques durs reliés à l'interface SCSI chez **Supermac Technology** de **Mountain view**, **Univation Inc**, **Sunol Systemset MD Ideas** de **Foster City** (Californie), ainsi qu'une nouvelle version de la **Bernouilli Box** d'**Iomega**.

Des monstres

Comme si le Macintosh Plus n'était pas suffisant, voici que des constructeurs bricolent des Mac encore plus rapides et puissants. Voici ainsi le **Super 20** de **Levco**, comprenant un processeur 68020 cadencé à 16 Mégahertz, 2 ou 4 Méga-octets de mémoire vive, et l'emplacement d'un coprocesseur arithmétique 68881. Prix : 8500 dollars.

En France, **P-Ingénierie** importe l'**Hyperdrive 2000** de **General Computer** : il contient un processeur cadencé à 12 Mégahertz, un coprocesseur arithmétique 68881 et 2 Mégas de mémoire. Ce "superMac" permet d'accélérer certaines opérations entre 2 et 100 fois. Un dessin employant des routines du Macintosh prend ainsi 170 secondes sur le Mac contre 17 sur l'Hyperdrive 2000. Ce micro décidément gonflé est destiné à tous les amateurs de hautes performances en matière de DAO, de Finances, de simulations, mais aussi d'édition.

MaxRAM est une carte qui transforme un Macintosh 128Ko ou 512Ko en Mac 1536Ko, avec la possibilité de choisir entre la configuration 1 Mo de mémoire centrale / 400Ko de RAM disk, et la configuration 512Ko de mémoire / 1 Mo de RAM disk, cette dernière étant, à notre avis, la plus utile.Cette carte remplace la carte mère du Mac et ne pose apparemment pas de problème de compatibilité. Une des caractéristiques les plus intéressante du système est qu'il permet la récupération du contenu du RAM disk après une bombe ou un redémarrage. Le principe de travail avec cette extension est le suivant : démarrage du Mac avec chargement automatique en RAM (1 Mo) de MacWrite et MacPaint et tout autre logiciel de votre choix. A partir de là, tout se fait en RAM, le lecteur interne ne se servant plus qu'aux sauvegardes sur les disquettes. Les traitements deviennent très rapides ; ainsi, le chargement de MacWrite, qui prenait 15 secondes sur disquette (Finder 4.1 et Mac 512), 8" sur HyperDrive et 7" sur le MacPlus avec cache mémoire, ne prend plus que 5". Cette carte est vendue par la société **CCAM** 7300,00 F TTC aux possesseurs d'un 512Ko, 9900,00 F à ceux d'un 128 Ko.

Edition électronique au programme

Car l'autre axe stratégique d'Apple c'est le marché de l'édition électronique. Une autre prévision de Pom's : on verra cette année apparaître pour le Mac des logiciels de mise en page électronique révolutionnaires. Le marché de l'édition électronique est en pleine explosion. Dans les entreprises, il pourrait à lui seul justifier l'entrée du Macintosh Plus, (ou d'un nouveau frère doté d'un écran pleine page). Apple a visiblement décidé de tenter ce pari là. Au risque de voir le Mac devenir un terminal graphique des IBM.

En attendant une super imprimante à laser capable d'imprimer avec une densité de 1000 points par pouce, Apple a perfectionné sa **LaserWriter** (imprimant à 300 points par pouce). Elle inclut sept nouvelles polices de caractères et contient un Méga supplémentaire de mémoire morte. Pour l'exploiter, **Adobe**, l'auteur du contrôleur **PostScript** au cœur de l'imprimante a annoncé encore d'autres polices téléchargeables dans l'imprimante.

Côté logiciels, **Aldus** a annoncé une nouvelle version (1.2) de **PageMaker**, capable d'utiliser les 12 nouvelles polices de caractères d'**Adobe**, **Manhattan Graphics** sort la version 2.1 de **Ready Set To Go** capable de traiter des brochures de 40 pages, **Boston Software** a perfectionné **MacPublisher** connu en France sous le nom de **MacEditeur**, désormais capable de traiter des brochures de 96 pages, comportant de nouveaux formats (y compris le format tabloïd), des possibilités d'importer des éléments par télécommunications, et une palette graphique de 99 motifs pour réaliser des fonds. En prime, cette version est capable d'imprimer en couleurs sur la nouvelle imprimante Imagewriter II.

Déjà de nouveaux logiciels

Pour le Mac deux catégories de nouveaux logiciels, ceux qui fonctionnent sur le 512Ko, et ceux qui profitent déjà des caractéristiques du Macintosh Plus. Dans la première catégorie, **Microsoft** sort une nouvelle version de Multiplan : la 1.1, permettant d'utiliser simultanément plusieurs feuilles, et intégrant de nouvelles polices de caractères et sept fonctions financières nouvelles. **Statware** sort une version professionnelle de son programme statistique **Stat 80**. Il comporte notamment des analyses de corrélations des analyses factorielles et des manipulations de matrices avec 18 opérateurs et 50 fonctions. Prix : 400 dollars. Sur le front des langages, remarquons que Philippe Kahn, le patron français de **Borland** sort une version de son **Turbo Pascal** pour le Macintosh. Tandis que chez **P-Ingénierie**, sort le **TML Pascal**, présenté comme le premier environnement complet de développement de logiciels Pascal produisant du code natif 68000, et offrant ainsi les possibilités du défunt Lisa. Prix : 1400,00 F HT.

Voici enfin d'autres logiciels plus spécialement destinés au Mac Plus. Avec d'abord la nouvelle version de **Jazz**. Ce programme était à l'étroit sur le 512Ko, il pourra respirer sur le Mac Plus. Tout comme **SuperCrunch** de **Paladin Software**, le super tableur dont on dit qu'il concurrence Excel.

Et l'Apple //

Il ne faudrait pas croire que l'Apple // est abandonné. Voici pour lui des quantités de perfectionnements matériels.

Des cartes

Des cartes comme s'il en pleuvait. Deux fabricants de cartes pour l'IBM : **AST** et **Quadram** s'intéressent au //e. **AST** sort ainsi une carte de mémoire additionnelle d'un Méga-octets, une carte Ram Disk avec mémoire cache, une carte multi-fonctions (horloge, deux ports séries, etc), et un disque dur de 10 ou 20 Mégas. Tandis que **Quadram** propose carte horloge, carte 80 colonnes étendue, interface série avec buffer et la **Multicore**, une carte multi-fonctions (horloge, port série, port parallèlle, extension de 256Ko de mémoire). **Applied Engineering**, l'un des pionniers des extensions pour l'Apple //, a mis au point une carte accélérant les programmes d'un facteur de 3/1 et permettant en prime d'installer un processeur 16 bits dès que ceux ci deviendront disponibles. Avec 256Ko de mémoire vive. Prix : 279 dollars.

Mieux voir

L'amélioration de l'écran du // est possible. Grâce au **Screen Enhancer** de **Video 7**. Un modèle couleur permet de disposer de 4 couleurs de textes et 16 teintes de gris. Grâce aussi à un petit montage proposé dans la revue britannique **Apple User** par Chris Payne, permettant de transformer la sortie vidéo

monochrome en TTL (comme sur l'IBM). Chris Payne vend son montage tout fait pour 25 livres.

Améliorer Appleworks

Appleworks a été l'un des logiciels les plus vendus aux Etats-Unis. Rien d'étonnant donc si les Américains le perfectionnent. Le **Southern California Research Group** l'installe ainsi en ROM, sur une carte qui s'installe dans l'Apple //e. Chargement ultra rapide, opérations facilitées. Prix : 269 dollars. A noter qu'il faut leur envoyer une disquette contenant Appleworks.

La revue **Nibble** propose au travers de l'**International Apple Core** un programme permettant de réaliser des mailings avec **Appleworks** pour 30 dollars.

Communiquer

A bas prix. Mais si c'est possible ! Avec le Minitel. **Marvie** propose une interface de liaison (la **M 232 I**) permettant d'utiliser le modem du Minitel, et de stocker les pages vidéotex sur les disquettes de l'Apple, de transformer celui-ci en terminal 80 colonnes, de visualiser des pages ou d'éditer des textes. Prix : 995,00 F TTC avec carte série pour le //e et 695,00 F pour le //c.

Voler

Grâce au Macintosh, on l'attendait depuis longtemps. Eh bien tandis que Bill Budge, l'auteur inspiré de **Pinball Construction Set** concocte un simulateur de vol sur navette spatiale, **Sublogic** s'est enfin décidé à sortir pour le Mac son **Flight Simulator** le roi des simulateurs de vol. Avec des fenêtres tridimensionnelles multiples permettant de regarder simultanément plusieurs vues du paysage tout en volant. On peut même se regarder voler soi-même d'un point fixe extérieur. Vivement que ce simulateur arrive en France.

Formation

La société **KA** organise le jeudi 24 avril au PLM St Jacques un séminaire avancé sur Multiplan, animé par Hervé Thiriez et clôturé par Bernard Vergnes, président de Microsoft France.

Ce séminaire s'adresse à tous les utilisateurs confirmés de Multiplan pour, en une journée, leur présenter de multiples astuces d'utilisation de ce logiciel.

Mesurer

La revue **InCider** de février a consacré un article aux logiciels et interfaces permettant de transformer l'Apple // en instrument scientifique. On y trouve ainsi des logiciels d'expérimentation en chimie (**Chem Lab Simon & Schuster**) ou en température (**Temperature Lab d'Hayden** à 100 dollars) , d'anatomie (**The body in Focus de CBS Software** et **The Body transparent de Designware**), un système transformant l'Apple en électrocardiogramme d'effort (**Cardiovascular Fitness Lab de HRM** au prix de 175 dollars).

Adresses

JRD
11, place Sainte Croix
45000 Orléans
Tél. : 38 54 83 20

Tangent Technologies
5720 Peachtree Parkway Suite 100 Norcross GE 30092

Alpha Systèmes
29, avenue Gambetta
38000 Grenoble
Tél. : 76 43 28 40

Computer Applications
12813 Lindley Drive Raleigh NC 27614

3Com
1365 Shorbird Way P.O. Box 7390 Mountain View CA 94039

Kinetics
PO Box 3341 Walnut Creek CA 94598

P-Ingéniérie
226, Boulevard Raspail
75014 Paris

Statware
PO Box 510881 Salt Lake City UT 84151

Applied Engineering
PO Box 798 Carrollton TX 75006

Chris Payne
15 Braddenham Walk, Stoke Mandeville, Bucks, HP21 9DZ

Southern California Research Group
PO BOX 593 A Moorpark CA 93020

International Apple Core
Prod.Dept #N02 908 George Street Santa Clara CA95054

Marvie
37, rue des Mathurins
75008 Paris

CBS Software
One Fawcett Place Greenwich CT 06836

Designware
185 Berry St San Francisco CA 94107

HRM Software
175 Tompkins Av Pleasantville NY 10570

Simon & Schuster
Simon & Schuster Bldg 1203 Av of the Americas New York NY10020

Hayden
600 Suffolk St Lowell MA 01854
Sublogic Corp 713 Edgebrook Drive Champaign IL 618210

KA Service Formation
14, rue Magellan - 75008 Paris
Tél. : 47 23 72 00

CCAM
95, rue Lafayette - 75010 Paris
Tél. : 47 80 22 23

Suite du source 'DOSPATCH.S' (page 68)

```
155
156             JMP BRUNHAND        LA SEULE CHOSE QUI RESTE EST LE BINAIRE
157 RUN         JMP RUNHAND
158 EXEC        JMP EXECHAND
159
160 *--------------------------------
161 * NE SORT FILE NOT FOUND QU'APRES
162 * AVOIR ESSAYE LES 2 DRIVES DU
163 * SLOT ACTUEL.
164 *--------------------------------
165
166             LDA ERRNUM          PUIS VIENS LE PATCH: QUELLE ERREUR ?
167             CMP #9
168             BCS BIDRIVEND       UN FILE NOT FOUND, UN VOLUME MISMATCH ?
169             CMP #6              UN I/O ERROR ? NON ALORS TERMINE...
170             BCC BIDRIVEND
171
172 BIDRIVE    LDA #3              SINON, CHANGE DE DRIVE
173             SBC DRIVENUM
174             STA DRIVENUM
175
176             LDA DRIVESW         A-T-ON DEJA CHANGE DE DRIVE ?
177             BNE BIDRVEND1       OUI, ON EST BON POUR LE MESSAGE !!!
178             INC DRIVESW         NON, MAIS CE NE MARCHERA PAS 2 FOIS...
179             JMP DOCMD           DRIVE CHANGE, ON RE-ESSAYE
180
181 BIDRVEND1 JMP PATCH2            ANNULE DRIVESW POUR LA PROCHAINE FOIS
182 BIDRIVEND RTS                   ET RETOURNE AU TRAITEMENT DES ERREURS.
183
184 ********************************
185 *                              *
186 * TOUS LES RENSEIGNEMENTS SONT *
187 * TIRES DE 'BENEATH APPLE DOS' *
188 *                              *
189 ********************************
```

Vous... Votre Apple... ...et Pom's

Voici déjà un an, nous vous avions interrogé, sur vous, sur votre Apple et sur Pom's. Nous souhaitons en prendre l'habitude, pour mieux vous connaître, pour mieux apprécier l'évolution de votre équipement, pour mieux centrer nos articles sur vos aspirations. Comme l'an passé, nous procéderons à un tirage au sort qui permettra à dix lecteurs de bénéficier d'un abonnement avec disquettes.

Vous...

Votre âge
- de 15 ans ○
- de 15 à 19 ans ○
- de 20 à 24 ans ○
- de 25 à 34 ans ○
- de 35 à 45 ans ○
- + de 45 ans ○

Sexe
- féminin ○
- masculin ○

Langages pratiqués aisément
- Basic ○
- Pascal ○
- Forth ○
- Logo ○
- Assembleur 6502/65C02 ○
- Assembleur 68000 ○
- Autres..... ○

de préférence
- sous DOS ○
- sous ProDOS ○
- sous CP/M ○

Votre niveau en programmation
- débutant ○
- intermédiaire ○
- amateur averti ○
- haut niveau ○

Chaque semaine vous consacrez à votre micro
- - de 2 heures ○
- de 3 à 8 heures ○
- de 9 à 16 heures ○
- + de 16 heures ○

Votre matériel

Le micro
- Apple][+ ○
- Apple //e 6502 ○
- Apple //e 65C02 ○
- Apple //c ○
- Apple /// ○
- Macintosh 128 Ko ○
- Macintosh 512 Ko ○
- Macintosh Plus ○
- Lisa ○

Nombre de lecteurs, interne et externe : .....

L'imprimante
- ImageWriter ○
- ImageWriter // ○
- Apple DMP ○
- Epson ○
- Seïkosha ○
- Mannesman ○
- Centronics ○
- Oki ○

Autre .....

Les extensions
- Carte 80 colonnes ○
- 80 colonnes étendue ○
- Carte Z80 avec CP/M ○
- Carte souris ○
- Disque dur ○
- Table traçante ○

Ce que vous allez bientôt acquérir : .....

...et Pom's

Vous avez connu Pom's
- par une publicité ○
- dans un salon ○
- par un autre amateur ○
- en kiosque ○
- en boutique ○

De 5 (très bon) à 0 (très mauvais) votre opinion sur
- le niveau général.....
- la présentation.....
- l'intérêt des programmes.....
- la clarté des articles.....
- leur intérêt pédagogique.....
- les micro-informations.....
- les disquettes d'accompagnement.....

Quelles autres revues informatiques lisez-vous ?

.....
.....
.....
.....
.....
.....

Ce que vous préférez

.....
.....
.....
.....
.....
.....

Ce qu'il faut changer ou développer

.....
.....
.....
.....
.....
.....

Ce que vous détestez

.....
.....
.....
.....
.....
.....

Vos remarques particulières

.....
.....
.....
.....
.....
.....

Nom _____

Prénom _____

Adresse _____

_____

Ville _____

Code Postal _____

Abonné ○

Bon de commande

Disquettes

HAIFA source	(cf. Pom's n° 5)	.....	à 55,00 F	.....
H-BASIC	(cf. Pom's n° 8)	.....	à 150,00 F	.....
MUSIC	(cf. Pom's n° 10)	.....	à 80,00 F	.....
DISK-MANAGER	(cf. Pom's n° 11)	.....	à 450,00 F	.....
BASICIUM	(cf. Pom's n° 13)	.....	à 150,00 F	.....
E.P.E. 5.0	(cf. Pom's n° 23)	.....	à 200,00 F	.....
Échange E.P.E. 5.0	(cf. ce n° page 4)	.....	à 80,00 F	.....
PASCAL	(cf. ce n° page 4)	.....	à 80,00 F	.....
MAX (Moniteur étendu)	(cf. Pom's n° 18)	.....	à 150,00 F	.....
DOMINOS	(cf. Pom's n° 19)	.....	à 80,00 F	.....
MACASTUCES	(cf. Pom's n° 21)	.....	à 200,00 F	.....
P-FORMAT P-POLICE	(cf. Pom's n° 21)	.....	à 200,00 F	.....
COGO	(cf. Pom's n° 21)	.....	à 150,00 F	.....

Recueils

N°1, recueil des revues 1 à 4	.....	à 140,00 F	.....
Disquettes d'accompagnement 1 à 4	.....	à 150,00 F	.....
N°2, recueil des revues 5 à 8	.....	à 140,00 F	.....
Disquettes d'accompagnement 5 à 8	.....	à 190,00 F	.....
N°3, recueil des revues 9 à 12 . *(Disponible le 20 avril 1986)* ..	.....	à 140,00 F	.....
Disquettes d'accompagnement 9 à 12	.....	à 190,00 F	.....

Revues, disquettes

Revues 4 7 8	.....	à 35,00 F	.....
Revues 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.....	à 40,00 F	.....
Disquettes Apple II, //e, //c 1/2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23	.....	à 55,00 F	.....
Disquettes Macintosh			
14/15/16 groupées	.....	à 150,00 F	.....
17 18 19 20 21 22 23	.....	à 80,00 F	.....
Mac 'A'	.....	à 80,00 F	.....

Abonnements

Pour 6 numéros à partir du n° ☐

Abonnement à la revue seule	.....	à 200,00 F	.....
Abonnement revue + disquettes Apple II, //e, //c	.....	à 480,00 F	.....
Abonnement revue + disquettes Macintosh	.....	à 600,00 F	.....

Total TTC :

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

Montant du règlement :

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

Nom : .....

Adresse : .....

Imprimerie Rosay, 94300 Vincennes. Imprimé en France. Dépôt légal : Mars 1986. N° 3093

I.E.F.

Le spécialiste des PLUS de la Micro
vous invite dans le Nouveau Monde du

Macintosh Plus

Dernière heure
Floppy externe 800 K
2900 F HT

Caractéristiques

- Micro processeur 68000
- 128 K ROM intégrant des fonctions graphiques rapides et la gestion du bureau
- Clavier avec bloc numérique et touches curseur
- RAM 1 Méga Octets extensible à 4 Mégas
- Lecteur de disquette 800 K intégré
- Interface SCSI permettant de relier des périphériques puissants
- Système d'exploitation 5.1 avec architecture supérieure et mémoire cache

Plus rapide

Plus puissant

Plus ouvert

Plus connectable

Plus communiquant

Plus facile

Plus économique

A) Vous êtes équipé d'un Macintosh

IEF vous ouvre la porte des **plus** pour seulement :

- **4.500 F** HT (si vous êtes équipé d'un 512 K d'origine Apple)
- **6.500 F** HT (si vous êtes équipé d'un 128 K d'origine ou étendu)
- Pour ces prix, **IEF** vous change la plaque mère, le lecteur de disquettes et le clavier.
- La transformation est garantie 1 an par Apple.
- Ces prix ne sont valables que pendant une durée limitée, réservez dès aujourd'hui votre transformation.
- De plus, si vous achetez cette transformation, **IEF** vous offre son disque dur 20 Mégas au prix de **11.900 F** HT !

B) Vous n'êtes pas encore équipé d'un Macintosh

IEF vous offre **Macintosh Plus** exceptionnellement pour **24.900 F** HT

Promotion spéciale IEF de lancement :

1 Macintosh Plus + 1 disque dur 20 Méga Octets 34.900 F HT (offre limitée)

IEF propose des conditions spéciales pour les Grands Comptes et les établissements d'enseignement
Si vous voulez profiter d'une de ces offres, renvoyez vite le coupon réponse ci-dessous

I.E.F. 217, quai de Stalingrad 92130 ISSY LES MOULINEAUX Tél : (1) 45.57.14.14 Télex : 200210 F

Coupon réponse à retourner à : I.E.F. 217, quai de Stalingrad 92130 ISSY LES MOULINEAUX P 03 MC

NOM : ..... SOCIETE : .....

ACTIVITE : ..... TEL : .....

ADRESSE : .....

Je suis intéressé par : .....

l'actualité

les bancs d'essai

les guides d'achat

le dossier

les programmes

L'ORDINATEUR INDIVIDUEL

LA RÉFÉRENCE EN MICRO-INFORMATIQUE

A L'ORDINATEUR INDIVIDUEL, les rédacteurs, les conseillers techniques, les correspondants à l'étranger, l'équipe entière se mobilise pour vous fournir tous les mois une information complète et de qualité. Le monde de la micro bouge : L'O.I. teste pour vous les micros et logiciels qui apparaissent sur le marché. Il vous dit lesquels choisir et pourquoi. Vous êtes déjà équipé et vous souhaitez tirer le maximum de votre machine ? Les spécialistes de L'O.I. vous livrent conseils, programmes inédits et astuces d'utilisation. Lisez chaque mois L'ORDINATEUR INDIVIDUEL.