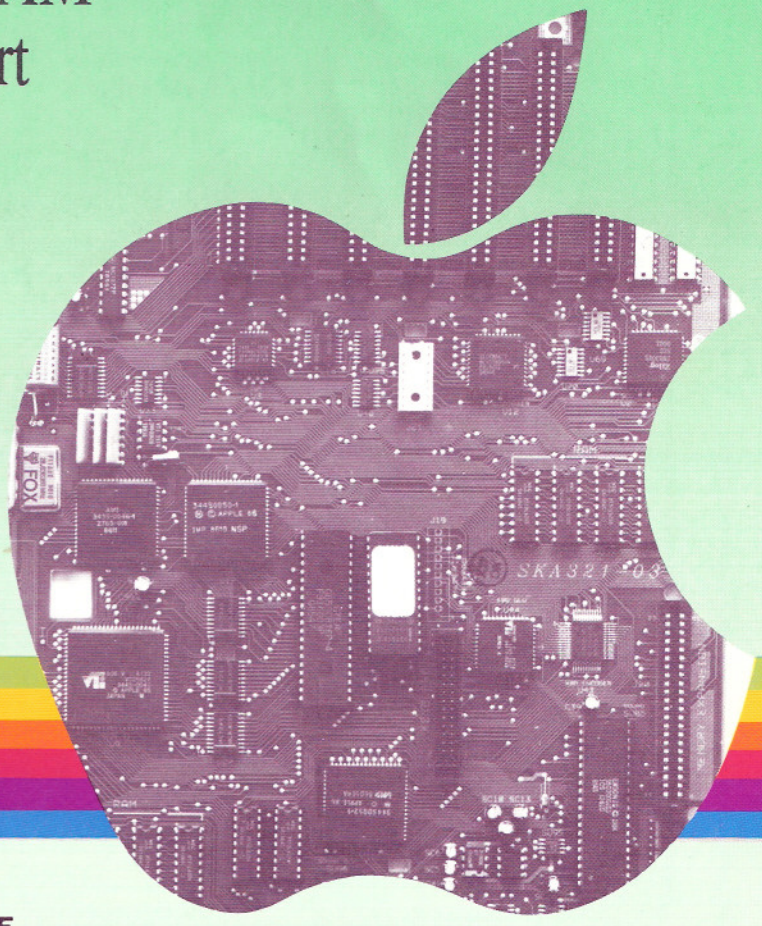
La revue francophone indépendante pour les utilisateurs des
Apple][+, //e, //e+, //c, IIGS™ et Macintosh™

pom's

MINITEL, Apple // et Macintosh

- Téléchargement de fichiers via Minitel
- Le debugging sous ProDOS
- ProDOS, fichiers et RAM
- Un système pico-Expert
- Codage de messages
- Scrollings en HGR
- JustText à l'essai
- Extasie à l'essai



M 2366 - 28 - 45,00 F

3792366045005 00280

NUMERO 28 - PRIX 45 F

ISSN : 0294-6068

NOUVEAU

TOPIQUES PASCAL est un livre qui vous permet d'acquérir la maîtrise complète de votre Apple en PASCAL. Ce livre présente:

- le fonctionnement des périphériques et les techniques PASCAL pour les gérer efficacement (gestion directe du clavier, multifenêtage, gestion de plusieurs curseurs, inversion directe de l'écran, test de la disponibilité de l'imprimante, copie de la page texte ou graphique, utilisation de l'Image Writer, paramétrage de la carte Super série, communications, redirection de l'écran vers la disquette, de l'imprimante vers l'écran)
- les extensions UCSD pour gérer la mémoire (tableaux dynamiques, tableaux creux, éditeur de texte, tableur, objet pointé de taille variable, codeur LISP) ou les disquettes (fichier dynamique, fiche de taille variable à accès direct)
- les moyens pour réaliser de vastes programmes (morcellement du texte, segmentation du code, librairies, gestion des erreurs, générateur de programmes, générateurs de générateurs, traces, types abstraits, programmation orientée objet)
- le fonctionnement système et les détails de l'implémentation Apple (types, formats fichiers, plans mémoire, adresses du BIOS et de l'interprète, appels système).

En tout **848 pages**, 141 programmes, 247 figures, un index complet.

Envoyez votre commande accompagnée du chèque à:

L'INSTITUT PASCAL
26 rue Lamartine - 75009 PARIS - Tel. 42.85.10.82

__ TOPIQUES PASCAL (290F TTC, étranger +20F)
__ les 4 disquettes source (100F TTC)

NOM: _____
Adresse:

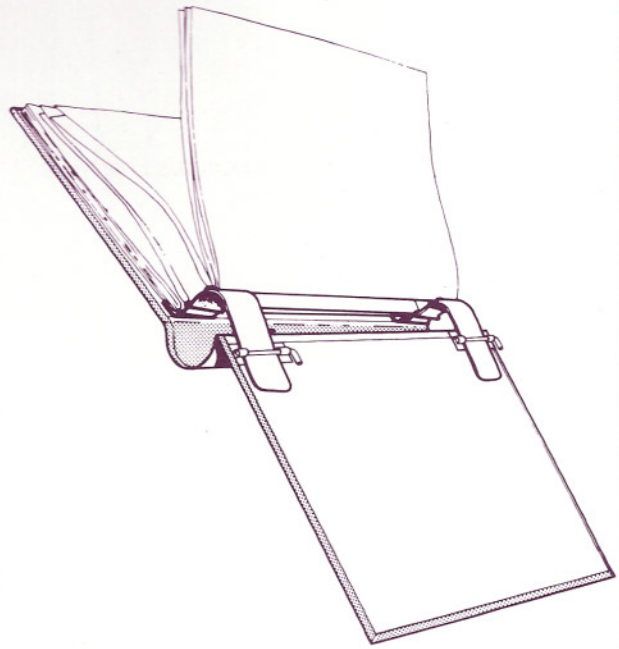
Les reliures Pom's

Pom's vous sert de référence, et à ce titre vous le consultez souvent.

Pour faciliter le rangement en bibliothèque et la consultation, Pom's vous propose des reliures toilées réunissant 6 numéros, soit un an de documentation.

Une fois reliées, les revues forment un livre agréable à utiliser, insensible à la manipulation.

Ces reliures sont couleur Bordeaux, le logo Pom's est frappé sur le dos et un aplat est réservé pour noter l'année. La reliure se fait par brides.



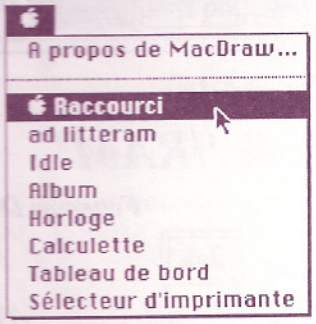
Reliures Pom's
60,00 F franco

Numéro 28
janvier-février 1987

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

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

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



Des avantages :

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

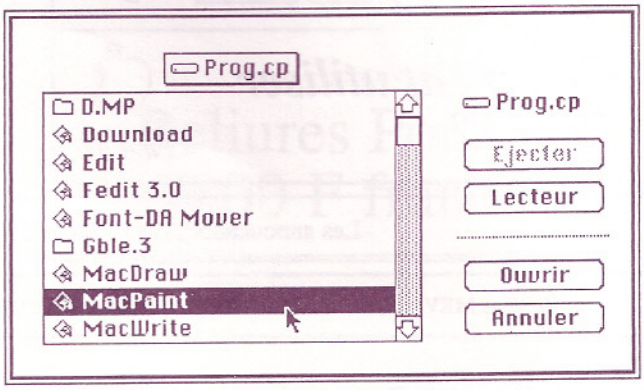
Sur un 512 Ko, les applications disposent de toute la place mémoire.

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

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

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

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



Avec Pom's, travaillez *directement* !

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

Éditorial

L'année 1987 sera celle du Mac ouvert ; il est en effet attendu depuis déjà longtemps. Sera-t-elle aussi celle de l'Apple IIGS ? Personne ne saurait aujourd'hui le dire, la richesse des possibilités de ce matériel et l'excellente compatibilité avec le reste de la gamme Apple // étant hélas compensées par un prix relativement élevé. Les transformations //e en IIGS dont on parle aujourd'hui résoudront certainement ce handicap.

Au niveau du logiciel Macintosh, signalons que – depuis juin 86 – on nous promet Writer Plus pour la semaine prochaine… Tôt ou tard, cela devrait devenir vrai ! Autre retard : dBase Mac qui, une semaine avant sa sortie aux États-Unis, a brutalement été stoppé par Ashton-Tate. Un développeur avait en effet constaté que la gestion de la mémoire virtuelle ralentissait énormément le fonctionnement à partir d'une certaine taille de fichier. Retour donc au *drawing-board*, la sortie du produit étant repoussée au milieu de l'année, car il est actuellement reprogrammé en assembleur.

Ceci dit, les produits nouveaux qui sortent actuellement dans les gammes Mac et IIGS sont nombreux, comme vous pourrez le voir dans la rubrique Micro-informations.

Nous savions vous avoir proposé des programmes séduisants dans le numéro 27, mais, avouons-le, nous n'attendions pas un tel succès. Vos courriers et appels téléphoniques se feront certainement aussi nombreux et enthousiastes pour les programmes de transmission proposés dans ces pages. Notons que, comme fréquemment, Pom's vous offre au prix du numéro des logiciels de haute qualité et de haute fiabilité dont la valeur dépasse le coût d'un abonnement revue plus disquette. Rappelons que les disquettes d'accompagnement (60,00 F pour Apple // et 80,00 F pour Macintosh) comportent les sources bien sûr mais également les programmes exécutables.

Pour vous simplifier la communication, Pom's fait dans le 'hard' : nous tenons à votre disposition des câbles de liaison Apple/Minitel. Peut-être serviront-ils pour de prochains programmes…

Hervé Thiriez

Ont collaboré à ce numéro Alexandre Avrane, Jean-Luc Bazanegue, François Dreyfuss, Jean-Michel Gourévitch, Roland Jost, Olivier Herz, Yves Martin, Gérard Michel, Christian Piard, Jacques Rey, Hervé Thiriez.
Directeur de la publication, rédacteur en chef : Hervé Thiriez.
Rédacteurs : Alexandre Avrane, Olivier Herz.
Siège social : Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Tél. : (1) 39.51.24.43.
Publicité : Éditions MEV.
Diffusion : N.M.P.P.
Impression : Rosay - 47, avenue de Paris - 94300 Vincennes. Tél. : (1) 43.28.18.63.

SHOW, I, C, O : le debugging sous ProDOS

Alexandre Avrane

Comme vous le savez tous, la recherche des erreurs d'un programme est une des activités principales de l'informatique. Tâche ô combien exaltante, bien entendu…

Malheureusement, il n'existe pas (encore) de méthodes infaillibles permettant de les éliminer dès la naissance, c'est-à-dire la conception d'un logiciel. En revanche, une panoplie d'outils destinés au *debugging* permet de les déceler plus rapidement. Examinons-les aujourd'hui, dans l'optique d'un programme écrit en assembleur ou en Applesoft, et dans un environnement ProDOS.

Les méthodes

Trois types d'approches sont souvent envisagées :

• D'abord, la méthode de la grande patience pas toujours récompensée : c'est l'analyse pas-à-pas des instructions exécutées ; le TRACE du Basic, ou un logiciel spécifique (tel que Bugbyter livré sur certaines disquettes ProDOS) pour les routines écrites en 6502. C'est long, fastidieux et éprouvant.

• Cette première approche peut être complétée par des ordres d'arrêt afin que le programme rende la main et qu'il soit ainsi possible d'examiner la valeurs des variables Applesoft ou de certains octets de la mémoire. Il s'agit, respectivement, des instructions STOP et BRK.

• Enfin, une mise au point affinée nécessite souvent une analyse détaillée des appels vers le système d'exploitation. Les commandes MON et NOMON du DOS 3.3 permettent ainsi d'obtenir des informations pertinentes en un minimum de temps.

En complément, on peut trouver un certain nombre d'utilitaires afin d'accélérer le temps passé au debugging : par exemple D-CODE (Beagle-Bros) et certaines routines proposées dans Beneath Apple DOS (et bien sûr dans Pom's).

Remarquons que le TRACE de l'Applesoft n'était pas disponible sous DOS 3.3 ; en revanche, les instructions MON et NOMON sont ignorées sous ProDOS. Aujourd'hui, notre objectif sera donc d'en créer les équivalents grâce aux commandes SHOW et NOSHOW.

MON & NOMON

Pour rappel, ces deux instructions permettent d'afficher à volonté, sous DOS 3.3 uniquement, l'ensemble des commandes passées par le programme Basic vers le système d'exploitation (CATALOG, READ, VERIFY, etc.) par l'exécution de lignes telle que :

```
260 PRINT CHR$(4);"OPEN FI
    CHIER1,D2"
```

Trois paramètres facultatifs définissent le type d'affichage : C pour n'afficher que les commandes, I (input) pour les données lues par un READ, O (output) pour celles écrites par un WRITE.
Deux points essentiels semblent dignes d'amélioration :

• D'abord, MON n'a d'effet que pour un programme écrit en Applesoft (ou en Integer Basic, pardons aux puristes). En particulier, une routine en assembleur qui appelle le File Manager ou RWTS ne peut être examinée.

• Ensuite, l'affichage à l'écran brouille très rapidement celui-ci si il est utilisé à d'autres fins (grille de saisie par exemple), ce qui occasionne une gêne non négligeable.

Les commandes offriront donc les particularités suivantes :

Les commandes affichées sont celles qui appellent effectivement le noyau de ProDOS (MLI), et non pas la chaîne de caractères qui suit le PRINT CHR$(4) d'un programme Basic. Par ailleurs, l'interception de cette chaîne par le Basic.System n'est pas documentée par Apple ; il faudrait donc, pour la réaliser, modifier directement le Basic.System hors ses pages globales, avec tous les risques liés à une fâcheuse dépendance vis-à-vis des versions actuelles de ProDOS.

Les anciens paramètres (C, I, O) sont conservés. De plus, le paramètre P permet d'envoyer les informations également sur une imprimante connectée en slot 1. Il sera alors possible d'exécuter un programme en obtenant une trace papier des actions effectuées.

Enfin, nos nouvelles commandes ne peuvent malheureusement pas s'appeler MON et NOMON pour assurer une compatibilité totale avec les programmes écrits sous DOS 3.3 car, erreur de conception du Basic.System, les commandes de celui-ci (dont MON et NOMON qui sont reconnues mais ignorées) ne peuvent être redéfinies par une commande externe. Il aurait été préférable que le Basic.System s'intéresse à l'existence d'éventuelles commandes externes avant de s'occuper des siennes, et non l'inverse.

Exemple d'utilisation

A titre d'illustration, le programme en Applesoft:

```
100  D$ = CHR$(4)
110  PRINT D$"SHOW,C,O,P"
120  PRINT D$"OPEN FICHIER1"
130  PRINT D$"WRITE FICHIER1"
140  PRINT "ABCDEF"
150  PRINT "GHIJKL"
160  PRINT D$"CLOSE FICHIER1"
170  PRINT D$"NOSHOW,C,O,P"
```

fournira, lors de sa première exécution, sur l'imprimante connectée :

```
C5: Online S6,D1
C4: Get_file_info
    /DISK1/FICHIER1 (46)
C0: Create /DISK1/FICHIER1
C8: Open /DISK1/FICHIER1,#01
CB: Write #01,L$000E
ABCDEF
GHIJKL
CC: Close #01
```

Chaque ligne affichée comprend (toutes les valeurs sont fournies en hexadécimal):

- le numérique d'appel à ProDOS,
- le libellé correspondant à cet appel,
- éventuellement, le ou les paramètres pertinents,
- enfin, un éventuel code-retour entre parenthèses si une erreur s'est produite.

Cette méthode de visualisation des informations peut donc aussi servir à mieux comprendre le fonctionnement du Basic.System, ou à développer son propre interpréteur en assembleur. On pourra d'ailleurs en profiter pour relever une certaine redondance du Basic.System qui a parfois tendance à demander des informations qu'il connait déjà.

En reprenant les étapes de l'exemple précédent, on note successivement:

- Un appel Online afin de connaître le nom du préfixe courant.
- Un Get_file_info pour vérifier qu'il s'agit bien d'un fichier texte. À la première exécution, ce fichier n'existe pas et on obtient le code erreur correspondant. En règle générale, d'ailleurs, il est chaudement recommandé de toujours faire précéder toute manipulation de fichier par un Get_file_info.
- Un Create, uniquement à la première exécution, pour créer le fichier avec ses attributs (type, date/heure, etc).
- Un Open où le programme indique le nom du fichier, et reçoit en retour un numéro de référence.
- Un Write, correspondant avec le numéro de référence précédent, sur une longueur de $000E octets (14 octets, c'est-à-dire 2 enregistrements de 6 caractères suivis d'un retour-chariot).
- L'affichage des données "ABCDEF" et "GHIJKL" car on l'avait demandé (option O)
- Enfin, la fermeture du fichier.

Bien entendu, on aurait obtenu le même résultat si le programme Basic ci-dessus avait été écrit en assembleur.

Informations affichées

Les paramètres affichés par la commande SHOW reprennent en partie l'ensemble des paramètres utilisés pour chaque appel à ProDOS. En particulier :

- les noms de fichiers utilisés et/ou le préfixe ;
- le numéro de référence fourni par ProDOS à l'ouverture ;
- la longueur demandée des lectures/écritures de données ;
- la longueur effectivement obtenue si elle diffère de celle demandée ;
- la position courante sur le fichier, en nombre d'octets depuis le début ;
- les indications de slot/drive ;
- le numéro de priorité pour les appels de gestion des interruptions ;
- le numéro de bloc lu ou écrit ;
- le caractère de fin de ligne suivi de son octet de masque.

Afin d'alléger la densité de la présentation, seules les informations pertinentes pour l'appel en cours sont fournies.

De plus, lorsque les paramètres I ou O sont actifs, les octets de données des fichiers texte lus ou écrits sont envoyés à l'affichage. Afin d'éviter les crises d'épilepsie des cartes d'interface, les caractères de contrôle éventuellement rencontrés sont ignorés.

Il n'est pas prévu de rappeler aujourd'hui le détail des types d'appel à ProDOS, qui peuvent être obtenus dans le *Technical Reference Manual* d'Apple ou de *Beneath Apple ProDOS*. (On pourra également consulter le numéro 20 de Pom's).

En revanche, les codes-erreur possibles sont :

```
00 pas d'erreur
01 fonction inconnue
04 nombre de paramètres
   incorrect pour la fonction
25 table d'interruption pleine
27 i/o error
28 pas d'unité dans ce
   slot/drive
2B volume protégé contre
   l'écriture
2E un volume, sur lequel se
   trouve des fichiers ouverts,
   a été enlevé
40 syntaxe incorrecte du
   préfixe
42 pas plus de 8 fichiers
   ouvert simultanément
43 numéro de référence fourni
   non alloué à un fichier
   ouvert
44 le chemin d'accès du préfixe
   n'a pu être trouvé
45 volume actuellement non
   monté
46 fichier inconnu
47 impossible de créer ou
   renommer vers un nom de
   fichier déjà existant
48 plus de place sur le volume
49 plus de place sur le
   directory du volume
4A version de ProDOS trop
   ancienne pour lire ce
   fichier
4B type de stockage du fichier
   inconnu
4C lecture impossible après la
   fin du fichier
4D impossible de se positionner
   après la fin du fichier
4E erreur d'accès au fichier
   (fichier verouillé)
50 impossible d'ouvrir,
   renommer ou détruire un
```

fichier ouvert
51 directory endommagé
52 volume non formaté pour ProDOS
53 valeur hors champs d'un paramètre d'appel
55 pas plus de 8 volumes connus simultanément par ProDOS
56 adresse de buffer incorrect
57 deux volumes montés ont le même nom
58 carte du volume endommagée

À l'exception du code-retour 00, qui indique un accomplissement correct, les codes erreurs sont affichés, à la suite des paramètres, entre parenthèses.

Installation

Les commandes SHOW et NOSHOW sont gérées par une routine en assembleur qui utilise le modèle donné par les commandes TDUMP et INIT de Pom's 20. Le fichier source SHOW.CODE.S, assemblé par Big Mac, fournit l'objet SHOW.CODE.

Ce dernier est ensuite chaîné avec la routine CMDLOAD par le programme CMDLINK pour fournir un fichier SHOW directement exécutable. (CMDLOAD et CMDLINK ont été explicités en détail dans le numéro 20 de Pom's).

Pour initialiser les commandes SHOW et NOSHOW, il suffit de faire :

-SHOW

ou

BRUN SHOW

Il est également possible de charger les commandes par:

BLOAD SHOW.CODE
BRUN CMDLOAD

Après chargement du programme, aucune différente n'est visible tant qu'une instruction telle que SHOW,C n'est pas exécutée.

Le paramètre d'assemblage SLOT permet de spécifier le slot de l'imprimante si celle-ci n'est pas connectée en slot 1. Le programme fonctionne néanmoins parfaitement sur une configuration sans imprimante, pourvue que la commande SHOW,P ne soit jamais exécutée.

Enfin, notons que l'utilisation de ces deux nouvelles commandes est compatible avec celle des autres commandes utilisant le principe décrit par CMDLOAD. En particulier, la commande TDUMP de Pom's 20 (qui permet de vérifier entre autre le contenu des fichiers texte) peut être utilisée conjointement.

De même, SHOW gère les tentatives d'interruption et peut donc co-exister sans problèmes avec des programmes qui les génèrent, comme DATHEUR de Pom's 24 qui utilise les interruptions de la carte souris.

Utilisation hors Basic

Il est possible d'obtenir, dans un environnement où l'interpréteur système n'est plus le Basic.System (par exemple le Filer, Convert, Exerciser, etc), les informations affichées à chaque appel du MLI. Dans ce cas, il faut modifier le source et :

- fixer le paramètre BASIC à 0 (il est normalement à 1)
- fixer le paramètre ADRESSE à une adresse où la routine pourra se loger sans interférer avec l'interpréteur système actif (par exemple $8000 pour CONVERT)
- fixer le paramètre OUT à 0 si on désire une sortie uniquement sur écran, à 1 pour une sortie supplémentaire sur imprimante
- fixer le paramètre XSYSIO à 1 pour n'afficher que les appels au MLI, à 0 pour obtenir en plus les données lues ou écrites vers tous les types de fichiers
- assembler le source ainsi modifié
- ne pas chaîner le module objet avec CMDLOAD mais l'exécuter directement (*BRUN SHOW.CODE*)

Les informations de debugging seront alors toujours envoyées (impossible de moduler leur apparition).

Ainsi utilisé, SHOW permet par exemple d'observer les algorithmes des programmes utilitaires fournis par Apple sans devoir les désassembler dans le détail.
Ainsi, ne cherchez plus pourquoi CONVERT 1.0 est si long pour lire un catalogue DOS 3.3 : il se contente seulement de lire consécutivement jusqu'à cinq fois les mêmes blocs de la piste $11 !

Attention, le paramètre ADRESSE, s'il est mal choisi, peut entraîner un magnifique *plantage* du système. Il est donc très important de le déterminer avec soin afin de charger SHOW.CODE dans une zone mémoire qui ne sera jamais affectée par l'interpréteur actif.

Cette utilisation du programme SHOW est surtout destinée à un but pédagogique (pour mieux discerner le fonctionnement du Filer par exemple), ainsi qu'aux ambitieux qui écrivent leur propre interpréteur système.

Fonctionnement interne

Le principe de fonctionnement du programme repose sur l'interception des appels au MLI qui ont comme point d'entrée l'adresse $BF00. En modifiant le JMP qui se trouve à cette adresse, une routine prend la main sur tous les accès effectués vers ProDOS.

En examinant alors la pile, SHOW détermine l'adresse de la routine appelante, ainsi que le code d'appel au MLI et l'adresse de la table des paramètres associée. On peut alors appeler directement MLI en lui fournissant les valeurs précédemment trouvées. En retour, il reste à gérer l'affichage des informations concernant cet accès, puis rendre la main à la suite de la routine appelante.

Bien entendu, il faut également s'occuper de certains détails,

comme les interruptions (car une carte souris peut en générer très fréquemment), les vecteurs d'affichage (en déconnectant temporairement le Basic.System qui ne comprend plus rien si on lui dit ce qu'on affiche...) et d'autres babioles telles que la sauvegarde/restitution des registres ou la sortie en catastrophe si jamais la routine appelante se met à charger un nouvel interpréteur système.

SHOW gère également les codes d'appel inconnus, en affichant alors le libellé "Unidentified" suivi d'un beep péremptoire. On pourra remarquer que l'EXERCISER de la disquette Apple ne se gêne pas pour envoyer quelques codes inconnus...

Fichiers sur la disquette Pom's

T.SHOW.CODE
: source en Big Mac, type TEXT

SHOW
: objet exécutable après chaînage avec CMDLOAD (Pom's 20)

SHOW ne prétend nullement être l'outil complet de debugging sous ProDOS mais pourra rendre de grands services pour dépanner certains problèmes de plantage. En particulier, couplé avec la commande TDUMP, il permet une réparation très rapide des programmes qui accèdent aux fichiers texte. Enfin, c'est un aide pour la mise au point rapide (c'est à dire sans faire de pas-à-pas) des routines en assembleur qui manipulent ProDOS.

Source T.SHOW.CODE
(Assembleur Big Mac)

```
*
        LST  OFF

****************************************
*                                      *
*         ProDOS  SHOW,I,C,O,P         *
*                                      *
****************************************

****************************************
*                                      *
* Commandes SHOW et NOSHOW pour ProDOS *
*                                      *
* Copyright (c) 1986 Alexandre Avrane  *
*                                      *
* Modifié: 05/07/86                    *
* Créé:    07/01/86                    *
*                                      *
* Assembleur: Big Mac                  *
*                                      *
****************************************

* Fonction:
* ========
* Simule les commandes MON et NOMON du DOS 3.3, avec les
* paramètres C (commandes), I (input data) et O (output data),
* P (envoi des informations sur l'imprimante).
* Les commandes correspondent aux appels au MLI, et non
* les instructions issues de l'Applesoft.

* Syntaxe:
* =======
* SHOW   <paramètres> active les modes de debugging
* NOSHOW <paramètres> désactive
* paramètres:
*  C  affichages des appels de l'interpréteur vers ProDOS
*  I  affichage des octets lus d'un fichier texte
*  O  affichages des octets écrits vers un fichier texte
*  P  envoi de l'affichage vers l'imprimante
* Les paramètres peuvent etre précédés d'une virgule
* (pour assurer la compatibilité avec le DOS 3.3).
* Si aucun paramètre, la commande est ignorée.

* Ce programme doit etre linké avec CMDLOAD (Pom's 20)

* Configuration
* =============
* Apple II+, //e, //e+, //c, ///
* Vidéo: 40 colonnes (avec minuscules) ou 80 colonnes
* Processeur: 6502, 65C02, R65C02
* ProDOS + Basic.System (toutes versions)

BASIC    =    0
* Ce paramètre d'assemblage doit etre mis à zéro
* pour utiliser ce programme sans Basic.System;
* les paramètres sont alors tous considérés actifs.
* La routine doit alors impérativement etre chargée
* dans une zone d'adresses n'interférant pas
* avec l'interpréteur système actif.

SLOT     =    1
* Localisation du slot de l'imprimante

         DO   1-BASIC
ADRESSE  =    $8000      adresse de chargement
OUT      =    0          par défaut, pas de sortie vers l'impr
XSYSIO   =    1          par défaut, seulement show,c
         FIN

CH       =    $24        position horizontale
CSW      =    $36        vecteur de sortie des caractères
PTR1     =    $3C
PTR2     =    $3E
         DO   BASIC
IN_BUFF  =    $200       buffer de saisie
EXTRNCMD =    $BE06      commande externe Basic.System
BI_OUT   =    $BE30      vecteur réel de sortie pour Basic.Sys
XTRNADDR =    $BE50      rappel après extraction des paramètres
XLEN     =    $BE52      longueur de la commande externe
XCNUM    =    $BE53      numéro de commande Basic.System
PBITS    =    $BE54      flags de recherche par Basic.System
FILETYPE =    $BEB8      type de fichier pour Basic.System
         FIN
MLI      =    $BF00      accès au ProDOS MLI
CROUT    =    $FD8E      retour-chariot
PRBYTE   =    $FDDA      affiche l'accumulateur en hexadécimal
COUT     =    $FDED      affiche l'accumulateur en ascii
RTS      =    $FF58      instruction RTS immuable (on espère!)

         DO   BASIC
         ORG  $2100      ($2000-20FF pour CMDLOAD)
         ELSE
         ORG  ADRESSE
         FIN

* ======
* Macros
* ======
MOVE     MAC
         LDA  ]1
         STA  ]2
         LDA  ]1+1
         STA  ]2+1
         <<<

GET      MAC
         LDY  #]1
         LDA  (PTR1),Y
         <<<

DISPLAY  MAC
         LDA  #]1
```

```
        JSR  COUT
        <<<

SAVE_REG MAC
        PHP
        PHA
        TXA
        PHA
        TYA
        PHA
        <<<

LOAD_REG MAC
        PLA
        TAY
        PLA
        TAX
        PLA
        PLP
        <<<

        EXP  OFF

* ======================================
* Examen de la commande et des paramètres
* ======================================
* On arrive ici à chaque retour-chariot en mode immédiat,
* et pour chaque commande précédée de ctrl-d en mode différé.

START   =    *
        DO   BASIC
        CLD              convention avec ProDOS
        LDA  #>FIN+$100 conventions pour CMDLOAD
        LDA  #>LONG-$100
V_OLDCMD LDA RTS

        SEI
        LDA  #<IN_BUFF  le buffer d'entrée contient la saisie
        STA  PTR1
        LDA  #>IN_BUFF
        STA  PTR1+1
        LDY  #$FF
        CLC
CMD_L1  INY
        LDA  (PTR1),Y   prend un octet saisi
        BCS  CMD_L2     on sait déjà que c'est peut-etre SHOW
        CMP  CMD,Y      NOSHOW?
        BNE  CMD_L2     non
        CPY  #6-1       6 caractères dans NOSHOW
        BEQ  FOUND      oui
        BNE  CMD_L1     pas encore certain (jmp avec carry=0)

CMD_L2  EOR  CMD+2,Y    SHOW?
        BNE  NO_CMD     non, c'est pas pour moi
        CPY  #4-1       4 caractères dans SHOW
        BEQ  FOUND      oui
        SEC             indique qu'on cherche sur SHOW...
        BCS  CMD_L1     ...et on continue
FOUND   TYA
        EOR  #3         donne 0 si SHOW, 6 si NOSHOW
        STA  PTR2+1
FOUND_L1 INY
        LDX  #4-1       4 paramètres possibles
        LDA  (PTR1),Y
        CMP  #","       on ignore les virgules...
        BEQ  FOUND_L1
        CMP  #" "       ...et les blancs
        BEQ  FOUND_L1
        CMP  #$8D       si retour-chariot...
        BEQ  CMD_OK     ...fin de la saisie
FOUND_L2 CMP PARM,X     un des paramètres?
        BNE  FOUND_L3   je ne sais pas encore
        LDA  PTR2+1
        STA  FLAG,X     les flags seront nuls si activés
        BPL  FOUND_L1   ...et on recherche d'autres paramètres
FOUND_L3 DEX
        BPL  FOUND_L2   peut-etre le suivant
        BMI  NO_CMD     inconnu

NO_CMD  SEC
        CLI
        JMP  (V_OLDCMD+1) ni SHOW, ni NOSHOW...

CMD_OK  CLI
        LDA  #0
        STA  XCNUM      commande externe à Basic.System
        STA  PBITS      aucun paramètre à faire extraire
        STA  PBITS+1
        STY  XLEN       longueur de la commande
        >>>  MOVE,CMD_BACK+1;XTRNADDR adresse de retour
        ELSE
        LDA  #0         si utilisation sans Basic.System...
        STA  FLAG       ...les paramètres sont actifs en
        STA  FLAG+1
        STA  FLAG+2
        LDA  #OUT
        STA  FLAG+3
        FIN
        DO   BASIC
        CLC             c'est bien notre commande externe!
        RTS             on nous rappelle à la ligne suivante...
        FIN

* ==============================
* Mise en place des interceptions
* ==============================
* On arrive ici dès que ProDOS a compris qu'il n'avait pas
* à extraire des paramètres pour notre commande

CMD_NEXT LDA MLI+2      l'interception est-elle déjà là?
        CMP  #>MLI
        BCS  CMD_SET    non: il faut la placer
        LDA  FLAG
        ADC  FLAG+1
        ADC  FLAG+2     aucun paramètre C,I,O actif?
        CMP  #6*3
        BNE  CIAO       ...au moins un
OFF_CMD SEI
        >>>  MOVE,SV_MLI+1;MLI+1 déconnecte l'interception
        CLI
        CLC
        RTS
CMD_SET >>>  MOVE,MLI+1;SV_MLI+1 stocke l'adresse réelle d'entr
        SEI
        >>>  MOVE,MY_MLI+1;MLI+1 intercepte les appels vers MLI
CIAO    CLC             pas d'erreur à l'exécution de notre comm
        CLI
        RTS             cette fois-ci, c'est fini.
CMD_BACK LDA CMD_NEXT   (adresse pour etre relogée)

* =======================
* Interception des appels
* =======================
* On arrive ici lorsqu'une routine quelconque appelle MLI

MY_MLI  JMP  *+3
        >>>  SAVE_REG

        TSX
        SEI
        LDA  $105,X     récupère l'adresse du programme appelant
        STA  GO_PGM+1   ...et sauve pour l'appeler plus tard
        STA  PTR1
        LDA  $106,X
        STA  GO_PGM+2
        STA  PTR1+1

        >>>  GET,1      code d'appel à MLI
        STA  MLI_CODE
        INY
        LDA  (PTR1),Y
        STA  MLI_PARM
        INY
        LDA  (PTR1),Y   adresse des paramètres
        STA  MLI_PARM+1
        CLI

        DO   BASIC
        LDA  MLI_CODE
        EOR  #$CA       lecture demandée?
        BNE  GO_MLI
        LDA  FILETYPE
        CMP  #$FF       sur un fichier système?
        BNE  GO_MLI     si oui: on va charger un nouvel interpr
        >>>  MOVE,V_OLDCMD+1;EXTRNCMD déconnecte la commande ext
        JSR  OFF_CMD    déconnecte l'interception vers MLI
        >>>  LOAD_REG   on restore tout l'environnement...
        JMP  MLI        ...et on rend l'ame en plongeant sur MLI.
        FIN
```

```
GO_MLI   >>>  LOAD_REG
         STA  TEMP
         PLA
         PLA            ;dépile le JSR MLI du pgm appelant
         LDA  TEMP
SV_MLI   JSR  $0000     on appelle MLI
MLI_CODE NOP            nop pour etre relogeable
MLI_PARM NOP
         NOP

* ==========================
* Gestion des retours d'appel
* ==========================
* On arrive ici dès que le MLI a fait son boulot

         STA  MLI_RC    récupère le code-retour MLI
         >>>  SAVE_REG

         >>>  MOVE,CSW;SV_CSW  sauve l'ancien vecteur de sortie
         LDA  CH
         STA  SV_CH
         SEI            ;inhibibe toutes les interruptions
         LDA  FLAG+3    sortie vers l'imprimante?
         BEQ  PRINTER
         DO   BASIC
         >>>  MOVE,BI_OUT;CSW  ne passe plus par Basic.System
         FIN
         BNE  CHECK_C   =jmp
PRINTER  STA  CSW
         STA  CH
         LDA  #SLOT+$C0
         STA  CSW+1     imprimante activée (vers $C100)

CHECK_C  >>>  MOVE,MLI_PARM;PTR1
         LDA  FLAG      show,c actif ?
         BEQ  BACK
         JMP  IO        non

BACK     LDA  MLI_CODE
         JSR  PRINT_L4  affiche le code
         >>>  DISPLAY,":"

T_CALL_V LDA  T_CALL
         >>>  MOVE,T_CALL_V+1;PTR2
         LDY  #0-2
SEEK     INY
         INY
         LDA  (PTR2),Y  cherche le code utilisé
         BEQ  TRANS     code inconnu!
         EOR  MLI_CODE
         BNE  SEEK

TRANS    INY
         TYA
         CLC
         ADC  (PTR2),Y
         BCC  TRANS2
         INC  PTR2+1
         CLC
TRANS2   ADC  PTR2
         STA  PTR2      translate le vecteur vers la 2e tabl
         LDA  PTR2+1
         ADC  #0
         STA  PTR2+1

         LDY  #0
LOOP     LDA  (PTR2),Y
         BPL  PARAMS
         JSR  COUT      affiche le libellé
         INY
         BNE  LOOP

PARAMS   PHA
         >>>  DISPLAY,"  "
         PLA

* ========================================
* Affichage des informations selon l'appel
* ========================================
* Table des types de paramètres
* octet = %0gfedcba
*  a = 1  ==> type1: pathname en +1/+2
*  b = 1  ==> type2: block number en +4/+5
*  c = 1  ==> type3: unit number en +1
*  d = 1  ==> type4: reference en +1
*  e = 1  ==> type5: reference en +5
*  f = 1  ==> type6: length en +4/+5
*  g = 1  ==> type7: position en +2/+3/+4
*  f=g=1  ==> type8: caractère newline en +3, masque en +2

TYPE1    LSR
         PHA
         BCC  TYPE2
         >>>  GET,1
         STA  PTR2
         INY
         LDA  (PTR1),Y
         STA  PTR2+1
         LDY  #0
         LDA  (PTR2),Y  longueur du pathname
         BEQ  TYPE2     préfixe nul
         STA  TEMP
TYPE1_L  INY
         LDA  (PTR2),Y
         ORA  #$80
         JSR  COUT      affiche le pathname
         DEC  TEMP
         BNE  TYPE1_L

TYPE2    PLA
         LSR
         PHA
         BCC  TYPE3
         >>>  DISPLAY,"$"
         JSR  PRINT_1   affiche le numéro de bloc
         >>>  DISPLAY,","

TYPE3    PLA
         LSR
         PHA
         BCC  TYPE4
         >>>  DISPLAY,"S"
         >>>  GET,1
         ASL
         PHP
         LSR
         LSR
         LSR
         LSR
         LSR
         ORA  #$B0
         JSR  COUT      affiche le slot
         >>>  DISPLAY,","
         >>>  DISPLAY,"D"
         LDA  #1
         PLP
         BCC  TYPE3_L
         ASL
TYPE3_L  ORA  #$B0
         JSR  COUT      affiche le drive

TYPE4    PLA
         LSR
         PHA
         BCC  TYPE5
         >>>  DISPLAY,","
         >>>  DISPLAY,"#"
         LDY  #5
         JSR  PRINT_3   affiche la référence à l'ouverture

TYPE5    PLA
         LSR
         PHA
         BCC  TYPE6
         >>>  DISPLAY,"#"
         LDY  #1
         JSR  PRINT_3   affiche la référence utilisée

TYPE6    PLA
         BEQ  THE_END
         PHA
         >>>  DISPLAY,","
         PLA
         LSR
         PHA
         BCC  TYPE7
         BNE  TYPE8
         >>>  DISPLAY,"L"
         >>>  DISPLAY,"$"
         JSR  PRINT_1   affiche la longueur...
```

```
        >>>  GET,6
        LDY  #4
        CMP  (PTR1),Y   ....et compare avec la longueur effect
        BEQ  TYPE7
        PHA
        >>>  DISPLAY,"-"
        >>>  DISPLAY,">"
        >>>  GET,7
        JSR  PRINT_L4   si une différence modulo $100, on l'aff
        PLA
        JSR  PRINT_L4

TYPE7   PLA
        LSR
        BCC  THE_END
        >>>  DISPLAY,"B"
        >>>  DISPLAY,"$"
        LDY  #4
        JSR  PRINT_2    position (relative byte address)
        DEY
        JSR  PRINT_3
        CLC
        BCC  THE_END    =jmp

TYPE8   PLA
        >>>  DISPLAY,"$"
        LDY  #3
        JSR  PRINT_3
        >>>  DISPLAY,"("
        DEY
        JSR  PRINT_3
        >>>  DISPLAY,")"

THE_END LDA  MLI_RC
        PHA
        BEQ  RC_0       code-retour 0 ...
        >>>  DISPLAY,"  "
        >>>  DISPLAY,"("
        PLA
        PHA
        JSR  PRINT_L4   ...sinon on l'affiche...
        >>>  DISPLAY,")"
RC_0    JSR  CROUT
        PLA
        BNE  GOBACK     ...et on termine

* ==================================================
* C'était un "open" ?  pour un fichier texte ?
* alors les "read" et "write" afficheront les données
* ==================================================
IO      LDA  MLI_CODE
        CMP  #$C8       open?
        BNE  SHOW_IO    ...non
        >>>  GET,5      obtient le # de référence
        TAY
        DO   BASIC
        LDA  FILETYPE   type texte ?
        EOR  #$04       ...get_file_info, au moins sous Basic.S
        ELSE
        LDA  #XSYSIO
        FIN
        STA  TXT_FLAG,Y  0 <=> fichier texte
        CLC
        BCC  GOBACK     =jmp

* ==================================
* Affichages des données si show,i,o
* ==================================
SHOW_IO CMP  #$CA       read ?
        BEQ  SHOW_IO2
        CMP  #$CB       write?
        BNE  GOBACK
        LDY  FLAG+2     show,o actif ?
        BNE  GOBACK
        BEQ  SHOW_OK
SHOW_IO2 LDY FLAG+1     show,i actif ?
        BNE  GOBACK
SHOW_OK >>>  GET,1      # de référence du read/write
        TAY
        LDA  TXT_FLAG,Y est-ce un fichier texte?
        BNE  GOBACK     non!
        >>>  GET,3
        STA  IO_PTR+1
        DEY
        LDA  (PTR1),Y
        STA  IO_PTR     obtient l'adresse du buffer d'entrée/s
        >>>  GET,6
        STA  PTR2
        INY
        LDA  (PTR1),Y
        STA  PTR2+1     stocke la longueur effectivement lue

IO_LOOP LDA  PTR2
        BNE  IO_LONG
        LDA  PTR2+1
        BEQ  GOBACK     jusqu'à la fin...
        DEC  PTR2+1
IO_LONG DEC  PTR2       ...on décrémente la longueur
        LDA  $FFFF      (adresse gérée dynamiquement)
IO_PTR  =    *-2
        AND  #$7F
        CMP  #$0D
        BEQ  IO_PRINT   sauf les retour-chariots...
        CMP  #$20
        BCC  IO_INC     ...on élimine les codes de controle
IO_PRINT ORA #$80       force le bit 7 à 1
        JSR  COUT
IO_INC  INC  IO_PTR     maintenant allons cueillir son voisin
        BNE  IO_LOOP
        INC  IO_PTR+1
        BNE  IO_LOOP    =jmp

* ==================================
* Retour vers le programme appelant
* ==================================
GOBACK  =    *
        >>>  MOVE,SV_CSW;CSW restore la sortie vidéo Basic.S
        LDA  SV_CH
        STA  CH
        CLI
        CLC
        LDA  GO_PGM+1   calcule l'adresse de suite du prog
        ADC  #4
        STA  GO_PGM+1
        LDA  GO_PGM+2
        ADC  #0
        STA  GO_PGM+2

        >>>  LOAD_REG   restore les registres...
GO_PGM  JMP  $0000      ...et c'est fini!

* ===================
* Routine d'affichage
* ===================
PRINT_1 LDY  #5
PRINT_2 LDA  (PTR1),Y
        JSR  PRINT_L4
        DEY
PRINT_3 LDA  (PTR1),Y
PRINT_L4 JMP PRBYTE

* ================
* Zones de données
* ================
        BRK             marque la fin de la zone relogeable
TXT_FLAG =   *-1
        DS   8          indicateur fichiers texte
SV_CSW  DA   0          ancien vecteur de sortie
MLI_RC  DFB  0          code-retour MLI
TEMP    DFB  0          compteur temporaire
SV_CH   DFB  0          position horizontale
        DO   BASIC
CMD     ASC  "NO"
        ASC  "SHOW"
PARM    ASC  "CIOP"
        FIN
FLAG    DFB  6,6,6,6    par défaut désactivés

T_CALL  =    *          table des types d'appel
        HEX  40
        DFB  C40-*
        HEX  41
        DFB  C41-*
        HEX  65
        DFB  C65-*
        HEX  80
        DFB  C80-*
        HEX  81
        DFB  C81-*
        HEX  B2
        DFB  CB2-*
```

```
        HEX  C0
        DFB  CC0-*
        HEX  C1
        DFB  CC1-*
        HEX  C2
        DFB  CC2-*
        HEX  C3
        DFB  CC3-*
        HEX  C4
        DFB  CC4-*
        HEX  C5
        DFB  CC5-*
        HEX  C6
        DFB  CC6-*
        HEX  C7
        DFB  CC7-*
        HEX  C8
        DFB  CC8-*
        HEX  C9
        DFB  CC9-*
        HEX  CA
        DFB  CCA-*
        HEX  CB
        DFB  CCB-*
        HEX  CC
        DFB  CCC-*
        HEX  CD
        DFB  CCD-*
        HEX  CE
        DFB  CCE-*
        HEX  CF
        DFB  CCF-*
        HEX  D0
        DFB  CD0-*
        HEX  D1
        DFB  CD1-*
        HEX  D2
        DFB  CD2-*
        HEX  D3
        DFB  CD3-*
        HEX  00          code inconnu: doit etre le dernier
        DFB  C00-*       (doit etre < $FF)

C40     ASC  "Alloc_interrupt"
        DFB  %00010000
C41     ASC  "Dealloc_interrupt"
        DFB  %00010000
C65     ASC  "Quit"
        DFB  %00000000
C80     ASC  "Read_block"
        DFB  %00000110
C81     ASC  "Write_block"
        DFB  %00000110
CB2     ASC  "Get_time"
        DFB  %00000000
CC0     ASC  "Create"
        DFB  %00000001
CC1     ASC  "Destroy"
        DFB  %00000001
CC2     ASC  "Rename"
        DFB  %00000001
CC3     ASC  "Set_file_info"
        DFB  %00000001
CC4     ASC  "Get_file_info"
        DFB  %00000001
CC5     ASC  "Online"
        DFB  %00000100
CC6     ASC  "Set_prefix"
        DFB  %00000001
CC7     ASC  "Get_prefix"
        DFB  %00000001
CC8     ASC  "Open"
        DFB  %00001001
CC9     ASC  "Newline"
        DFB  %01110000
CCA     ASC  "Read"
        DFB  %00110000
CCB     ASC  "Write"
        DFB  %00110000
CCC     ASC  "Close"
        DFB  %00010000
CCD     ASC  "Flush"
        DFB  %00010000
CCE     ASC  "Set_mark"
        DFB  %01010000
CCF     ASC  "Get_mark"
        DFB  %01010000
CD0     ASC  "Set_eof"
        DFB  %01010000
CD1     ASC  "Get_eof"
        DFB  %01010000
CD2     ASC  "Set_buf"
        DFB  %00010000
CD3     ASC  "Get_buf"
        DFB  %00010000
C00     ASC  "Unidentified"
        DFB  $87         beep!
        DFB  %00000000

FIN     =    *
LONG    =    FIN-START+$100
        LST  ON
```

Récapitulation SHOW

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par :
BSAVE SHOW,A$2000,L$600

```
2000- AD 00 BF C9 4C F0 05 A9
2008- 87 4C ED FD AD 4D BE F0
2010- 05 A9 15 4C 09 BE AD 04
2018- 21 69 00 20 98 20 90 05
2020- A9 0E 4C 09 BE CD 02 21
2028- 90 F6 AE 08 BE 8D 08 BE
2030- 8E 07 21 AE 07 BE 8E 06
2038- 21 A0 00 8C 07 BE 48 E9
2040- 21 85 3C 68 38 E9 04 85
2048- 74 A9 21 85 49 84 48 A0
2050- 00 B1 48 F0 27 20 8E F8
2058- A4 2F C0 02 D0 0F B1 48
2060- C9 21 90 09 CD 02 21 B0
2068- 04 65 3C 91 48 A5 48 38
2070- 65 2F 85 48 A5 49 69 00
2078- 85 49 D0 D3 A0 00 A9 21
2080- 84 3C 85 3D 18 6D 04 21
2088- 84 42 88 84 3E 85 3F AD
2090- 08 BE 85 43 C8 4C 2C FE
2098- 8D FB 20 A5 74 18 69 04
20A0- 8D FC 20 86 3D CE FC 20
20A8- F0 47 AD FC 20 8D FD 20
20B0- AD FD 20 48 4A 4A 4A AA
20B8- 68 29 07 A8 B9 F3 20 3D
20C0- 58 BF D0 E1 A5 3D D0 09
20C8- B9 F3 20 1D 58 BF 9D 58
20D0- BF AD FC 20 38 CE FD 20
20D8- ED FD 20 CD FB 20 D0 D0
20E0- A5 3D D0 07 18 AE FD 20
20E8- E8 8A 60 A9 00 85 3D F0
20F0- B9 38 60 80 40 20 10 08
20F8- 04 02 01 00 00 00 AD 99
2100- D8 A9 26 A9 04 AD 58 FF
2108- 78 A9 00 85 3C A9 02 85
2110- 3D A0 FF 18 C8 B1 3C B0
2118- 0B D9 88 24 D0 06 C0 05
2120- F0 0E D0 F0 59 8A 24 D0
2128- 2E C0 03 F0 03 38 B0 E4
2130- 98 49 03 85 3F C8 A2 03
2138- B1 3C C9 AC F0 F7 C9 A0
2140- F0 F3 C9 8D F0 16 DD 8E
2148- 24 D0 07 A5 3F 9D 92 24
2150- 10 E3 CA 10 F1 30 00 38
2158- 58 6C 06 21 58 A9 00 8D
2160- 53 BE 8D 54 BE 8D 55 BE
2168- 8C 52 BE AD BA 21 8D 50
2170- BE AD BB 21 8D 51 BE 18
2178- 60 AD 02 BF C9 BF B0 1D
2180- AD 92 24 6D 93 24 6D 94
2188- 24 C9 12 D0 29 78 AD 20
2190- 22 8D 01 BF AD 21 22 8D
2198- 02 BF 58 18 60 AD 01 BF
21A0- 8D 20 22 AD 02 BF 8D 21
21A8- 22 78 AD BD 21 8D 01 BF
21B0- AD BE 21 8D 02 BF 18 58
21B8- 60 AD 79 21 4C BF 21 08
21C0- 48 8A 48 98 48 BA 78 BD
21C8- 05 01 8D 6B 24 85 3C BD
21D0- 06 01 8D 6C 24 85 3D A0
21D8- 01 B1 3C 8D 22 22 C8 B1
21E0- 3C 8D 23 22 C8 B1 3C 8D
21E8- 24 22 58 AD 22 22 49 CA
21F0- D0 1F AD B8 BE C9 FF D0
21F8- 18 AD 06 21 8D 06 BE AD
2200- 07 21 8D 07 BE 20 8D 21
2208- 68 A8 68 AA 68 28 4C 00
2210- BF 68 A8 68 AA 68 28 8D
2218- 86 24 68 68 AD 86 24 20
```

```
2220- 00 00 EA EA EA 8D 85 24
2228- 08 48 8A 48 98 48 A5 36
2230- 8D 83 24 A5 37 8D 84 24
2238- A5 24 8D 87 24 78 AD 95
2240- 24 F0 0C AD 30 BE 85 36
2248- AD 31 BE 85 37 D0 08 85
2250- 36 85 24 A9 C1 85 37 AD
2258- 23 22 85 3C AD 24 22 85
2260- 3D AD 92 24 F0 03 4C CE
2268- 23 AD 22 22 20 77 24 A9
2270- BA 20 ED FD AD 96 24 AD
2278- 75 22 85 3E AD 76 22 85
2280- 3F A0 FE C8 C8 B1 3E F0
2288- 05 4D 22 22 D0 F5 C8 98
2290- 18 71 3E 90 03 E6 3F 18
2298- 65 3E 85 3E A5 3F 69 00
22A0- 85 3F A0 00 B1 3E 10 06
22A8- 20 ED FD C8 D0 F6 48 A9
22B0- A0 20 ED FD 68 4A 48 90
22B8- 21 A0 01 B1 3C 85 3E C8
22C0- B1 3C 85 3F A0 00 B1 3E
22C8- F0 10 8D 86 24 C8 B1 3E
22D0- 09 80 20 ED FD CE 86 24
22D8- D0 F3 68 4A 48 90 0D A9
22E0- A4 20 ED FD 20 6D 24 A9
22E8- AC 20 ED FD 68 4A 48 90
22F0- 2A A9 D3 20 ED FD A0 01
22F8- B1 3C 0A 08 4A 4A 4A 4A
2300- 4A 09 B0 20 ED FD A9 AC
2308- 20 ED FD A9 C4 20 ED FD
2310- A9 01 28 90 01 0A 09 B0
2318- 20 ED FD 68 4A 48 90 0F
2320- A9 AC 20 ED FD A9 A3 20
2328- ED FD A0 05 20 75 24 68
2330- 4A 48 90 0A A9 A3 20 ED
2338- FD A0 01 20 75 24 68 F0
2340- 6D 48 A9 AC 20 ED FD 68
2348- 4A 48 90 2F D0 47 A9 CC
2350- 20 ED FD A9 A4 20 ED FD
2358- 20 6D 24 A0 06 B1 3C A0
2360- 04 D1 3C F0 16 48 A9 AD
2368- 20 ED FD A9 BE 20 ED FD
2370- A0 07 B1 3C 20 77 24 68
2378- 20 77 24 68 4A 90 2F A9
2380- C2 20 ED FD A9 A4 20 ED
2388- FD A0 04 20 6F 24 88 20
2390- 75 24 18 90 19 68 A9 A4
2398- 20 ED FD A0 03 20 75 24
23A0- A9 A8 20 ED FD 88 20 75
23A8- 24 A9 A9 20 ED FD AD 85
23B0- 24 48 F0 14 A9 A0 20 ED
23B8- FD A9 A8 20 ED FD 68 48
23C0- 20 77 24 A9 A9 20 ED FD
23C8- 20 8E FD 68 D0 75 AD 22
23D0- 22 C9 C8 D0 10 A0 05 B1
23D8- 3C A8 AD B8 BE 49 04 99
23E0- 7A 24 18 90 5E C9 CA F0
23E8- 0B C9 CB D0 56 AC 94 24
23F0- D0 51 F0 05 AC 93 24 D0
23F8- 4A A0 01 B1 3C A8 B9 7A
2400- 24 D0 40 A0 03 B1 3C 8D
2408- 29 24 88 B1 3C 8D 28 24
2410- A0 06 B1 3C 85 3E C8 B1
2418- 3C 85 3F A5 3E D0 06 A5
2420- 3F F0 20 C6 3F C6 3E AD
2428- FF FF 29 7F C9 0D F0 04
2430- C9 20 90 05 09 80 20 ED
2438- FD EE 28 24 D0 DD EE 29
2440- 24 D0 D8 AD 83 24 85 36
2448- AD 84 24 85 37 AD 87 24
2450- 85 24 58 18 AD 6B 24 69
2458- 04 8D 6B 24 AD 6C 24 69
2460- 00 8D 6C 24 68 A8 68 AA
2468- 68 28 4C 00 00 A0 05 B1
2470- 3C 20 77 24 88 B1 3C 4C
2478- DA FD 00 FD 00 A9 AD 8D
2480- BF 9D A9 00 00 00 00 00
2488- CE CF D3 C8 CF D7 C3 C9
2490- CF D0 06 06 06 06 40 35
2498- 41 43 65 53 80 56 81 5F
24A0- B2 69 C0 70 C1 75 C2 7B
24A8- C3 80 C4 8C C5 98 C6 9D
24B0- C7 A6 C8 AF C9 B2 CA B8
24B8- CB BB CC BF CD C3 CE C7
24C0- CF CE D0 D5 D1 DB D2 E1
24C8- D3 E7 00 ED C1 EC EC EF
24D0- E3 DF E9 EE F4 E5 F2 F2
24D8- F5 F0 F4 10 C4 E5 E1 EC
24E0- EC EF E3 DF E9 EE F4 E5
24E8- F2 F2 F5 F0 F4 10 D1 F5
24F0- E9 F4 00 D2 E5 E1 E4 DF
24F8- E2 EC EF E3 EB 06 D7 F2
2500- E9 F4 E5 DF E2 EC EF E3
2508- EB 06 C7 E5 F4 DF F4 E9
2510- ED E5 00 C3 F2 E5 E1 F4
2518- E5 01 C4 E5 F3 F4 F2 EF
2520- F9 01 D2 E5 EE E1 ED E5
2528- 01 D3 E5 F4 DF E6 E9 EC
2530- E5 DF E9 EE E6 EF 01 C7
2538- E5 F4 DF E6 E9 EC E5 DF
2540- E9 EE E6 EF 01 CF EE EC
2548- E9 EE E5 04 D3 E5 F4 DF
2550- F0 F2 E5 E6 E9 F8 01 C7
2558- E5 F4 DF F0 F2 E5 E6 E9
2560- F8 01 CF F0 E5 EE 09 CE
2568- E5 F7 EC E9 EE E5 70 D2
2570- E5 E1 E4 30 D7 F2 E9 F4
2578- E5 30 C3 EC EF F3 E5 10
2580- C6 EC F5 F3 E8 10 D3 E5
2588- F4 DF ED E1 F2 EB 50 C7
2590- E5 F4 DF ED E1 F2 EB 50
2598- D3 E5 F4 DF E5 EF E6 50
25A0- C7 E5 F4 DF E5 EF E6 50
25A8- D3 E5 F4 DF E2 F5 E6 10
25B0- C7 E5 F4 DF E2 F5 E6 10
25B8- D5 EE E9 E4 E5 EE F4 E9
25C0- E6 E9 E5 E4 87 00 D0 39
25C8- 8D 10 C0 F0 34 20 8A 9D
25D0- 20 6C 9C B0 25 20 62 9F
25D8- 90 04 CA BD 00 02 09 80
25E0- 9D 00 02 8A D0 F4 20 00
25E8- 9A 20 D5 9A 2C 53 BE 10
25F0- DC 60 20 8A 9D 20 48 9C
25F8- 90 F7 20 45 B2 C9 05 D0
```

InterPom's
Téléchargement via MINITEL

Christian Piard

Depuis la parution du numéro 27 de Pom's, le câble de liaison Apple/Minitel fait partie de votre équipement. Vous ne l'avez pas trouvé chez les revendeurs ? Vous ne l'avez pas réalisé vous-même ? Pom's vous le propose aujourd'hui par correspondance.

Cette liaison trouve ici une nouvelle application, la transmission (émission et réception) de fichiers par téléphone via Minitel entre Apple, même si votre correspondant ne dispose 'que' d'un Macintosh.

Ce programme vous permettra d'échanger plus facilement et rapidement des fichiers et programmes, et par exemple de travailler à domicile puis de télécharger le Macintosh de votre bureau...
Cette présentation ne concerne que la version Apple //, la version Macintosh se trouvant aux pages 41 à 53.

Tous fichiers, Toutes tailles, Tous supports.

Bien que plus délicate à mettre en œuvre, la transmission de fichiers a été préférée à la transmission de disquettes complètes pour une question de rapidité et pour ne pas être tributaire d'un type particulier de support.

Ce programme, conçu sous ProDOS, profite de la souplesse de ce système d'exploitation : tous les types de fichiers sont accessibles en suivant une unique procédure ce qui simplifie la programmation.

Côté utilisateur, cela signifie que tous les fichiers sont transférables par téléphone : fichiers Basic, TEXT, binaires, variables, AppleWorks tant texte que base de données ou tableur. Il peut également s'agir de tout type utilisateur (de $F1 à $F8) et de fichiers système. ProDOS, lui-même un fichier, est compris dans cette liste non limitative.

Deux exceptions toutefois :

- le type BAD est autorisé mais, regroupant par définition des blocs illisibles, l'issue d'une tentative de transmission sera l'échec.
- Le type DIR (directory) est interdit car créer des dossiers et sous-dossiers sur le disque du récepteur ne présenterait que des inconvénients.

Autre avantage de ProDOS, l'indépendance vis-à-vis du périphérique : le fichier à transmettre sera indifféremment sur une disquette 5,25 pouces, une 3,5', un disque dur et pourquoi pas le disque virtuel /RAM ?

La taille n'est limitée que par la durée (et le coût) de la communication : le fichier est chargé par fractions en mémoire. Il faut compter transmettre les données à raison de 4 kilo-octets par minute. Le programme Basic le plus volumineux ne demandera guère plus de 5 minutes.

Si le fichier à transmettre n'est pas au format ProDOS, il vous suffira de le convertir à l'aide de l'utilitaire 'CONVERT' ou, pour les possesseurs de //c, à l'aide d''Utilitaires Système'.

Le protocole

L'ordinateur émetteur transmet un bloc de données *transportant* 512 octets codés, un numéro d'ordre, un octet de contrôle de ce numéro, une somme de contrôle. L'ordinateur récepteur contrôle la validité du numéro d'ordre, le numéro lui-même, la somme de contrôle. Si tout va bien, il enregistre les données et envoie un acquièscement à l'émetteur qui va poursuivre. Dans le cas contraire, il demandera la réémission des données ou, en cas d'incohérence dans la succession des blocs, il demandera l'annulation de la transmission. D'autres contrôles sont effectués pour assurer une excellente fiabilité à l'ensemble : trop de temps entre deux caractères ? trop de temps entre deux blocs ?

Le premier bloc de données transmis comporte le type d'ordinateur émetteur, le nom du fichier, sa dimension, son type et les paramètres auxiliaires. Le récepteur saisit ainsi les éléments nécessaires à la création d'un fichier identique et vérifiera que le support dispose d'un nombre suffisant de blocs libres.

Le codage

Le modem du Minitel travaillant sur 7 bits et la gestion d'écran se réservant certains codes ASCII, il a été nécessaire de coder les données. La méthode retenue ici consiste à forcer à 1 le bit 6 de chaque octet afin de n'envoyer que des octets supérieurs à 63, les codes de contrôle du Minitel se trouvant au-dessous de cette valeur.

Il faut donc 4 octets composés de 6 bits utiles seulement pour représenter 3 octets. 683 octets seront envoyés par Minitel pour véhiculer les 512 octets de chaque blocs ProDOS. Ce type de codage, qui ne prend qu'une fraction de seconde, est plus

adapté à la transmission téléphonique (gain de temps) que la simple scission en deux octets, méthode utilisée dans les transmissions locales.

Le schéma page 31 montre le principe de codage.

Mode d'emploi

Si vous n'avez pas la disquette Pom's, deux solutions : saisir le source avec l'assembleur ProCODE, assembler, ou bien saisir l'objet en mode moniteur, et le sauvegarder sur une disquette ProDOS.
Si vous avez la disquette, transférer le fichier 'Minitel' sur une disquette ProDOS à l'aide de 'Convert' ou 'Utilitaires Système'.

Votre disquette ProDOS devra comporter les fichiers : ProDOS, Minitel et Basic.System. Ce dernier est indispensable pour que le programme puisse afficher le catalogue d'une disquette.
Il suffit alors de faire :

- INTERPOMS

Pour un démarrage direct sur ce programme, il faut le renommer *STARTUP*.

Une transmission de fichier peut se présenter ainsi :

- Communication téléphonique entre les intéressés ;
- Mise sous tension des Minitels (sans les connecter) ;
- Lancement des programmes. Émetteur et récepteur devront demander l'émission et la réception dans la même minute ;
- Les ordinateurs connectent eux-mêmes les Minitels. Les programmes déconnectant sans préavis le Minitel, il est préférable de ne pas raccrocher le combiné pendant la transmission. Différents messages vous informent du bon déroulement de la procédure.

Le programme

Le menu vous propose :

- l'envoi d'un fichier, indiquer alors le nom du fichier. Le préfixe par défaut sera automatiquement ajouté sauf si le "chemin d'accès" est complet, c'est-à-dire commençant par un "/".
- la réception d'un fichier, réception qui se fera en suivant le "chemin" indiqué par le préfixe,
- OnLine qui affiche tous les volumes en ligne,
- la modification du préfixe courant, préfixe qui peut viser un dossier c'est-à-dire être du type */DISK/CLIENTS/DEVIS*,
- le catalogue.

L'ordinateur récepteur constituera le ficher sur le disque sous le nom d'origine si l'émetteur est un Apple //, en suivant le chemin du préfixe, par exemple :

L'émetteur envoie

/FIC/DOS1/CALCUL

Le préfixe chez le récepteur est

/DISQUE.DUR/

Le fichier créé sera accessible par

/DISQUE.DUR/CALCUL

Au cas où un fichier *CALCUL* existerait déjà sur le volume récepteur, le programme essaiera de le créer en incrémentant le dernier caractère : *CALCUM*. S'il existe également, il le nommera *CALCUN* etc. Après *CALCUZ*, il tentera *CALCU0*, *CALCU1*... Au cas, peu probable, où vous transfèreriez 36 fois le même fichier, le programme ne pourra pas le créer.

Si l'émetteur est un Macintosh, en raison des nombreux caractères exotiques (¥ Δ √ ¶ ∞ ∂ ∫ ª ¸ ˘ ¢ †) autorisés dans les noms de fichiers, le programme donnera le nom *FICHIER.MAC.A*, puis *FICHIER.MAC.B*...

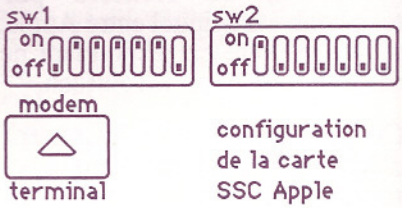
La routine de saisie des noms de fichiers et de préfixe en contrôle la syntaxe : minuscules converties en majuscules, pas de chiffre ni de point en premier caractère ou après un "/", succession de deux "/" interdite etc.

Le Minitel

Côté émetteur, il est indispensable que le Minitel soit "retournable" (ils le sont pratiquement tous aujourd'hui). Côté récepteur, c'est sans importance. Le Minitel échange ses données sur le réseau téléphonique de la manière suivante : il émet à 75 bauds et reçoit à 1200. Pour que la transmission soit rapide, il convient que le Minitel émetteur soit capable se "retourner", c'est-à-dire émettre à 1200 bauds. Le récepteur émettra ses acquièscements à 75 bauds.

La liaison série

Vous avez un Apple //e ou IIGS doté d'un carte Apple SuperSérie : installez la carte en port 2, et configurez la ainsi :



Vous avez un Apple //c, le programme se charge de la configuration du port série intégré.

Dans tous les cas, il vous faut un câble de liaison qui fera interface entre l'ordinateur et le Minitel. Le schéma est présenté ci-contre. Si vous doutez de vos capacités d'électroniciens, ce câble est disponible à la revue, par correspondance. N'omettez pas de débrancher les appareils avant d'effectuer le raccordement.

Correspondant Mac

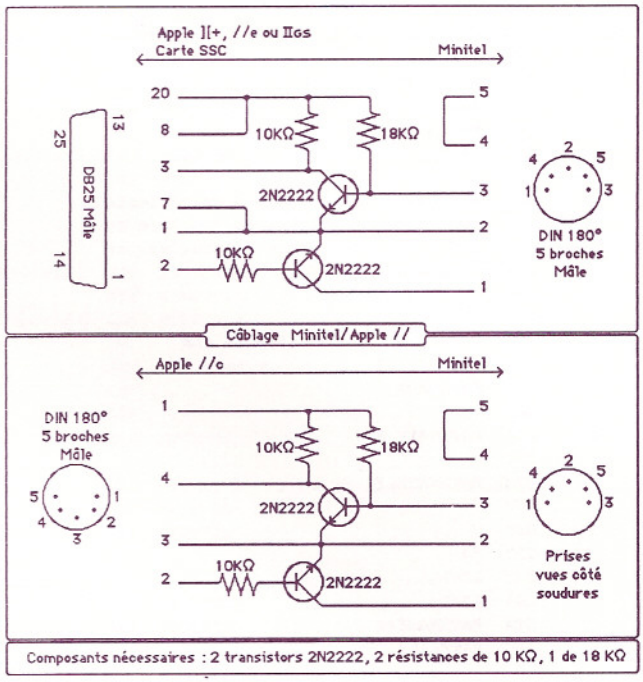
Si votre correspondant possède un Macintosh, vous pourrez lui envoyer n'importe quel type de fichier, mais lui ne pourra vous transmettre que des fichiers de type TEXT (MacWrite, Multiplan, Excel... en format TEXT). Certains caractères issus du Mac sont recodées à la réception pour être plus présentables sur l'Apple // (caractères • — … – ÷ " " ' ' « » Æ æ Œ œ ê ô î ï par exemple).

Source INTERPOMS.1
Assembleur ProCODE

```
        PUT   INTERPOMS.2
        PUT   INTERPOMS.3
```

Source INTERPOMS.2

```
*----------------
* INTERPOMS
* © Pom's & CP
*----------------

        LST   OFF
        DSK   INTERPOMS
        ORG   $2000

SOH     =     1
EOT     =     4
ACK     =     6
NACK    =     21
CAN     =     25
COMPTNC =     $18
COMPTC  =     $1A
FILEBUF =     $1000
BUFNC   =     $1400
BUFC    =     $1700
DEVCNT  =     $BF31
MLI     =     $BF00
CREATE  =     $C0
OPEN    =     $C8
READ    =     $CA
WRITE   =     $CB
CLOSE   =     $CC
GETFINFO =    $C4
ONLINE  =     $C5
STPREFIX =    $C6
GTPREFIX =    $C7
GETTIME =     $82
GETEOF  =     $D1
SETEOF  =     $D0
KBD     =     $C000
STROBE  =     $C010
STATUS  =     $6
DATA    =     $8
HP      =     $C030
COUT    =     $FDED
GBASL   =     $26
GBASCALC =    $F847
VTAB    =     $FC22
CV      =     $25
CROUT   =     $DAFB
WNDTOP  =     $22
WNDBTM  =     $23
CH      =     $24
STROUT  =     $DB3A
```



Composants nécessaires : 2 transistors 2N2222, 2 résistances de 10 KΩ, 1 de 18 KΩ

```
HOME    =     $FC58
TEXT    =     $FB39
BASIC   =     $3D0
CLREOP  =     $FC42
LINPTR  =     $ED24
LASTDEV =     $BF30

DEB00   LDX   #$FF        ; initialise la pile (utile
        TXS               ; pour repartir après RESET)
        LDA   $BF00       ; Est-on sous ProDOS ?
        CMP   #$4C
        BEQ   DEV0        ; oui, tvb
        LDA   #$7F
        JSR   ERREUR
        JMP   BASIC       ; Non, fin.

DEV0    LDA   $FBB3       ; Est-on sur un //c ?
        CMP   #6
        BNE   DEV0E
        LDA   $FBC0
        BNE   DEV0E0
        JSR   DEUXC       ; oui, on règle l'interface

        LDA   #2
        JSR   $FE95
        LDA   #7
        JSR   COUT
        JSR   $FE93
        JMP   DEV0E

DEV0E0  LDA   #7
```

Récapitulation 'INTERPOMS'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par :
BSAVE INTERPOMS,A$2000,L$1566
sur une disquette ProDOS.

```
2000- A2 FF 9A AD 00 BF C9 4C
2008- F0 08 A9 7F 20 D2 31 4C
2010- D0 03 AD B3 FB C9 06 D0
2018- 1D AD C0 FB D0 13 20 C4
2020- 34 A9 02 20 95 FE A9 07
2028- 20 ED FD 20 93 FE 4C 36
2030- 20 A9 07 20 00 C2 A9 C3
2038- 8D F2 03 A9 32 8D F3 03
2040- 49 A5 8D F4 03 AD 30 BF
2048- 8D 8B 20 20 00 BF C5 8A
2050- 20 B0 4A AD 6E 35 29 0F
2058- 8D 6E 35 A8 B9 6E 35 C8
2060- 99 6E 35 88 88 C0 00 D0
2068- F3 A9 AF 8D 6F 35 EE 6E
2070- 35 20 00 BF C6 8E 20 A9
2078- C0 85 07 85 09 A9 A8 85
2080- 08 A9 A9 85 06 4C 9D 20
2088- 00 00 02 00 6E 35 01 6E
2090- 35 03 22 08 26 72 2F 80
2098- 2D B9 2C BE 2B A9 15 20
20A0- ED FD 20 39 FB 20 58 FC
20A8- 20 71 2D A9 01 20 30 21
20B0- A9 16 20 30 21 A2 00 BD
20B8- CE 21 F0 06 20 ED FD E8
20C0- D0 F5 20 EA 2F A9 03 85
20C8- 22 A9 16 85 23 A0 0A B9
20D0- F8 21 99 9D 04 88 10 F7
20D8- 20 FD 25 A9 04 85 25 20
20E0- 22 FC 20 A7 31 A0 00 B9
20E8- 43 21 F0 07 20 ED FD C8
20F0- 4C E7 20 A9 01 8D B7 2C
20F8- A9 13 20 47 F8 A9 16 65
```

```
        JSR  $C200
DEV0E   LDA  #<RESET     ; Si RESET, on recommence
        STA  $3F2
        LDA  #>RESET
        STA  $3F3
        EOR  #$A5
        STA  $3F4

        LDA  LASTDEV     ; Récupère dernier lecteur
        STA  LD          ; utilisé pour mettre son
        JSR  MLI         ; préfixe en préfixe par
        DFB  ONLINE      ; défaut
        DA   PARLD0
        BCS  MENU
        LDA  PATHNAME    ; récupère le nom
        AND  #%00001111
        STA  PATHNAME
        TAY
DEV1    LDA  PATHNAME,Y
        INY
        STA  PATHNAME,Y
        DEY
        DEY
        CPY  #0
        BNE  DEV1
        LDA  #"/"
        STA  PATHNAME+1
        INC  PATHNAME

        JSR  MLI         ; met en préfixe par
        DFB  STPREFIX    ; défaut le nom trouvé
        DA   PARLD1

        LDA  #$C0        ; interface en slot 2
        STA  STATUS+1
        STA  DATA+1
        LDA  #$A8
        STA  DATA
        LDA  #$A9
        STA  STATUS

        JMP  MENU        ; On commence
OLDVEC  DS   2

PARLD0  DFB  2
LD      DS   1
        DA   PATHNAME
PARLD1  DFB  1
        DA   PATHNAME

*----------------
* MENU
*----------------

ADRCHOIX DA  ENVFICH     ; Adresse des différentes
         DA  RECFICH     ; options
         DA  MODPREF
         DA  ONL
         DA  CATAL
         DA  QUITTER
```

```
MENU    LDA  #21         ; désactive l'éventuelle
        JSR  COUT        ; carte 80 colonnes
        JSR  TEXT
        JSR  HOME
        JSR  CLOSEFIL    ; On ferme (cas du RESET)
        LDA  #1          ; Prépare écran
        JSR  LIGNE
        LDA  #22
        JSR  LIGNE
        LDX  #0
MENU0   LDA  MESTITRE,X
        BEQ  MENU1
        JSR  COUT
        INX
        BNE  MENU0
MENU1   JSR  LITPREF     ; lit et affiche préfixe
        LDA  #3          ; limite écran
        STA  WNDTOP
        LDA  #22
        STA  WNDBTM
        LDY  #10
P       LDA  POMS,Y
        STA  $49D,Y
        DEY
        BPL  P

CHOIX   JSR  EFFDATE
        LDA  #4          ; VTAB
        STA  CV
        JSR  VTAB
        JSR  RECOIT      ; au cas où...
        LDY  #0
CHOIX0  LDA  MESMENU,Y   ; affiche les menus
        BEQ  CHOIX1
        JSR  COUT
        INY
        JMP  CHOIX0
CHOIX1  LDA  #1          ; attend 1 seul carac
        STA  LONG
        LDA  #19         ; en ligne 19
        JSR  GBASCALC
        LDA  #22         ; colonne 22
        ADC  GBASL
        STA  GBASL
        BCC  CHOIX2
        INC  GBASL+1
CHOIX2  JSR  INPUT
        LDY  #5
        LDA  BUFFIN
CHOIX3  CMP  MESCHOIX,Y  ; est-ce un carac autorisé
        BEQ  CHOIX4      ; oui
        DEY
        BPL  CHOIX3
        JSR  BIP         ; non
        BCS  CHOIX1

CHOIX4  LDA  #<ADRCHOIX  ; initialise saut indirect
        STA  CHOIXIND+1
        TYA              ; Y = choix - 1
        BEQ  CHOIXIND    ; on incrémente le saut indirect
CHOIX5  INC  CHOIXIND+1  ; en f(n) du choix
```

```
2100- 26 85 26 90 02 E6 27 20
2108- C7 2B A0 05 AD 8E 2C D9
2110- 3D 21 F0 08 88 10 F8 20
2118- 82 32 B0 D7 A9 91 8D 2E
2120- 21 98 F0 09 EE 2E 21 EE
2128- 2E 21 88 D0 F7 6C 91 20
2130- 20 47 F8 A0 27 A9 DF 91
2138- 26 88 10 FB 60 C5 D2 CD
2140- CF C3 D1 8D 8D A0 A0 BC
2148- C5 BE EE F6 EF F9 E5 F2
2150- A0 F5 EE A0 E6 E9 E3 E8
2158- E9 E5 F2 8D 8D A0 A0 BC
2160- D2 BE E5 E3 E5 F6 EF E9
2168- F2 A0 F5 EE A0 E6 E9 E3
2170- E8 E9 E5 F2 8D 8D A0 A0
2178- BC CD BE EF E4 E9 E6 E9
2180- E5 F2 A0 EC E5 A0 F0 F2
2188- FB E6 E9 F8 E5 8D 8D A0
2190- A0 BC CF BE EE EC E9 EE
2198- E5 8D 8D A0 A0 BC C3 BE
21A0- E1 F4 E1 EC EF E7 F5 E5
21A8- 8D 8D A0 A0 BC D1 BE F5
21B0- E9 F4 F4 E5 F2 8D 8D 8D
21B8- A0 A0 A0 A0 A0 A0 A0 A0
21C0- D6 EF F4 F2 E5 A0 E3 E8
21C8- EF E9 F8 A0 BA 00 D4 FB
21D0- EC FB E3 E8 E1 F2 E7 E5
21D8- ED E5 EE F4 AF D4 FB EC
21E0- FB F2 FB E3 E5 F0 F4 E9
21E8- EF EE A0 DF DF DF DF DF
21F0- DF DF DF DF DF DF 8D 00
21F8- 20 09 0E 14 05 12 10 0F
2200- 0D 13 20 A9 03 85 22 20
2208- 58 FC 20 A7 31 20 EA 2F
2210- A9 07 85 25 20 22 FC A9
2218- 00 85 24 A0 24 A9 B8 20
2220- 3A DB A9 26 8D B7 2C A9
2228- 09 20 47 F8 20 C7 2B 90
2230- 03 4C 10 2E AC B8 2C D0
2238- 05 20 82 32 B0 D2 B9 8E
2240- 2C C9 AF D0 01 88 8C 6E
2248- 35 8C AE 35 B9 8E 2C C8
```

```
        INC  CHOIXIND+1
        DEY
        BNE  CHOIX5

CHOIXIND JMP  (ADRCHOIX) ; saut mis à jour par le choix

LIGNE   JSR  GBASCALC
        LDY  #39
        LDA  #"_"
LIGNE1  STA  (GBASL),Y
        DEY
        BPL  LIGNE1
        RTS

MESCHOIX ASC  "ERMOCQ"

MESMENU DFB  $8D,$8D
        ASC  "  <E>nvoyer un fichier"
        DFB  $8D,$8D
        ASC  "  <R>ecevoir un fichier"
        DFB  $8D,$8D
        ASC  "  <M>odifier le préfixe"
        DFB  $8D,$8D
        ASC  "  <O>nline
        dfb  $8D,$8D
        ASC  "  <C>atalogue"
        DFB  $8D,$8D
        ASC  "  <Q>uitter"
        DFB  $8D,$8D,$8D
        ASC  "        Votre choix :"
        DFB  0
MESTITRE ASC  "Téléchargement/Télér"
        ASC  "éception __________"
        DFB  $8D,0
POMS    INV  " INTERPOMS "

*----------------
* ENVOI DE FICHIER
*----------------

ENVFICH LDA  #3
        STA  WNDTOP
        JSR  HOME
        JSR  RECOIT     ; s'il reste un carac

        JSR  LITPREF
SAISIN1 LDA  #7         ; saisie du nom de fichier
        STA  CV
        JSR  VTAB
        LDA  #0
        STA  CH
        LDY  #>NOMFIC
        LDA  #<NOMFIC
        JSR  STROUT
        LDA  #38
        STA  LONG
        LDA  #9
        JSR  GBASCALC
        JSR  INPUT
        BCC  SA1
        JMP  VADEB
SA1     LDY  TPS
        BNE  SUITESAI   ; longueur nom > 0
        JSR  BIP        ; sinon pas d'accord
        BCS  SAISIN1

SUITESAI LDA  BUFFIN,Y  ; le dernier car est un / ?
        CMP  #"/"
        BNE  SUITEO
        DEY

SUITEO  STY  PATHNAME   ; stocke nom pour MLI
        STY  LNGPATHN   ; et garde la longueur
SUITSAI1 LDA  BUFFIN,Y
        INY
        STA  PATHNAME,Y
        DEY
        DEY
        BPL  SUITSAI1

        JSR  OUVREFIC   ; ouvre fichier demandé
        BCC  SUITOUV
        JMP  CLOTFIC

SUITOUV JSR  GFI        ; get file info
        BCC  SUITOUV0
        JMP  CLOTFIC

SUITOUV0 LDA  FILETYG
        CMP  #$F        ; C'est un directory ?
        BNE  SUIT01     ; non, tvb
        LDA  #$7E       ; oui, c'est interdit
        JSR  ERREUR
        JSR  CLOSEFIL
        JSR  GETRET
        JMP  ENVFICH

SUIT01  LDA  REFNUM
        STA  REFNUMG
        JSR  GETLONG    ; récupère longueur du fichier
        BCC  SUITOUV1
        JMP  CLOTFIC

SUITOUV1 LDA  #11
        STA  CV
        JSR  VTAB
        LDA  #0
        STA  CH
        LDX  BLOCKUS    ; affiche blocs à transférer
        LDA  BLOCKUS+1
        JSR  LINPTR
        LDY  #>MESBLOC
        LDA  #<MESBLOC
        JSR  STROUT     ; ok pour transfert ?
        JSR  GETRET
        BCC  BOUCLE0    ; oui
        JSR  CLOSEFIL   ; non
        JMP  ENVFICH

BOUCLE0 JSR  RETOURN    ; retourne le Minitel émetteur
        JSR  CONNECT    ; et le connecte
```

```
2250- 99 6E 35 88 88 10 F5 20
2258- 1F 2D 90 03 4C E6 23 20
2260- 54 2F 90 03 4C E6 23 AD
2268- 64 2F C9 0F D0 0E A9 7E
2270- 20 D2 31 20 71 2D 20 1E
2278- 32 4C 03 22 AD 30 2D 8D
2280- 1B 2D 20 0E 2D 90 03 4C
2288- E6 23 A9 0B 85 25 20 22
2290- FC A9 00 85 24 AE 68 2F
2298- AD 69 2F 20 24 ED A0 24
22A0- A9 D5 20 3A DB 20 1E 32
22A8- 90 06 20 71 2D 4C 03 22
22B0- 20 F8 34 20 E9 34 A9 0A
22B8- 20 30 21 A9 0B 85 22 20
22C0- 58 FC 20 FB DA E6 22 20
22C8- 16 35 A9 06 20 72 24 A9
22D0- 00 8D FA 25 8D FB 25 20
22D8- A7 31 B0 0D AD 00 C0 2C
22E0- 10 C0 C9 9B D0 F1 4C FA
22E8- 23 C9 19 F0 06 C9 15 F0
22F0- 05 D0 E4 4C 53 24 4C 87
22F8- 24 20 4B 2D 90 12 C9 4C
2300- D0 03 4C 91 23 20 D2 31
2308- A9 19 20 9A 31 4C E6 23
2310- 20 2B 30 20 16 35 A9 01
2318- 8D FC 25 20 72 24 AE FA
2320- 25 AD FB 25 20 24 ED 20
2328- 39 31 A9 0B 8D F7 25 20
2330- A7 31 B0 1A AD 00 C0 2C
2338- 10 C0 C9 9B F0 0D 20 88
2340- 31 CE F7 25 AD F7 25 F0
2348- 10 10 E4 4C FA 23 C9 19
2350- D0 03 4C 53 24 C9 15 D0
2358- 29 EE FC 25 AD FC 25 C9
2360- 0B D0 03 4C F2 23 20 16
2368- 35 AE FC 25 A9 00 20 24
2370- ED A9 02 20 72 24 AE FA
2378- 25 AD FB 25 20 24 ED 4C
2380- 27 23 C9 06 D0 A9 EE FA
2388- 25 D0 03 EE FB 25 4C F9
2390- 22 A9 01 8D FC 25 20 16
2398- 35 A9 08 20 72 24 AE FC
```

```
        LDA  #10
        JSR  LIGNE
        LDA  #11
        STA  WNDTOP
        JSR  HOME
        JSR  CROUT
        INC  WNDTOP
        JSR  MOUTKCR
        LDA  #6
        JSR  MOUT
        LDA  #0          ; initialise n° blocs
        STA  NUMBLOC
        STA  NUMBLOC+1

BOUCLE  JSR  RECOIT      ; attendre récepteur
        BCS  L1
        LDA  KBD
        BIT  STROBE      ; émetteur veut annuler ?
        CMP  #$9B
        BNE  BOUCLE
        JMP  ANNEME      ; oui
L1      CMP  #CAN        ; récepteur veut annuler ?
        BEQ  L2          ; oui
        CMP  #NACK       ; nack = commencer
        BEQ  L22
        BNE  BOUCLE
L2      JMP  ANNREC      ; on ferme

L22     JMP  CONTBLC     ; constitue le bloc 0
L3      JSR  LITFIC      ; lit tout le fichier
        BCC  DEC0
        CMP  #$4C        ; fin fichier ?
        BNE  L333        ; oui, le clore
        JMP  FINFICH

L333    JSR  ERREUR      ; non, afficher l'erreur
        LDA  #CAN        ; envoie annulation
        JSR  ENVOI
        JMP  CLOTFIC     ; et le fermer

DEC0    JSR  CODEBLOC    ; code le bloc
        JSR  MOUTKCR
        LDA  #1
        STA  NESSAI
        JSR  MOUT        ; Message 1
        LDX  NUMBLOC
        LDA  NUMBLOC+1
        JSR  LINPTR      ; n° du bloc
L30     JSR  ENVBLOC     ; et l'envoie
        LDA  #11
        STA  ATTENTE
L31     JSR  RECOIT      ; attendre acquiescement
        BCS  L4
        LDA  KBD         ; émetteur veut abandonner ?
        BIT  STROBE
        CMP  #$9B
        BEQ  L310
        JSR  WAIT        ; attend 10 X 1 seconde
        DEC  ATTENTE
        LDA  ATTENTE
        BEQ  L441
        BPL  L31
L310    JMP  ANNEME      ; oui
L4      CMP  #CAN        ; récepteur veut annuler ?
        BNE  L44
        JMP  ANNREC      ; oui
L44     CMP  #NACK       ; récepteur n'a pas compris ?
        BNE  L45         ; oui, on recommence max 10 fois
L441    INC  NESSAI
        LDA  NESSAI      ; on réémet 10 fois maximum
        CMP  #11
        BNE  L4411
        JMP  ANNEME0
L4411   JSR  MOUTKCR
        LDX  NESSAI
        LDA  #0          ; affiche n° de l'essai
        JSR  LINPTR
        LDA  #2          ; puis message 2
        JSR  MOUT
        LDX  NUMBLOC
        LDA  NUMBLOC+1
        JSR  LINPTR      ; puis numéro du bloc
        JMP  L30
L45     CMP  #ACK
        BNE  L31         ; récepteur a compris,
        INC  NUMBLOC     ; on passe au bloc suivant
        BNE  L450
        INC  NUMBLOC+1
L450    JMP  L3          ; on continue

FINFICH LDA  #1          ; on annonce fin de transm
        STA  NESSAI
FIN0    JSR  MOUTKCR     ; et on attend une réponse
        LDA  #8
        JSR  MOUT
        LDX  NESSAI
        LDA  #0
        JSR  LINPTR
        LDA  #EOT
        JSR  ENVOI

        LDA  #11
        STA  ATTENTE
FIN2    JSR  RECOIT
        BCS  FIN3
        LDA  KBD
        BIT  STROBE
        CMP  #$9B
        BEQ  FIN4
        JSR  WAIT
        DEC  ATTENTE
        LDA  ATTENTE
        BEQ  FIN5
        BNE  FIN2

FIN3    CMP  #CAN        ; on recoit CAN ou ACK ?
        BEQ  FIN4
        CMP  #ACK
        BEQ  FIN4

FIN5    INC  NESSAI      ; 10 essais maxi
```

```
23A0- 25 A9 00 20 24 ED A9 04
23A8- 20 9A 31 A9 0B 8D F7 25
23B0- 20 A7 31 B0 17 AD 00 C0
23B8- 2C 10 C0 C9 9B F0 1F 20
23C0- 88 31 CE F7 25 AD F7 25
23C8- F0 0A D0 E4 C9 19 F0 0E
23D0- C9 06 F0 0A EE FC 25 AD
23D8- FC 25 C9 0B D0 B8 20 16
23E0- 35 A9 04 20 72 24 20 71
23E8- 2D 20 07 35 20 1E 32 4C
23F0- 03 22 20 16 35 A9 07 20
23F8- 72 24 A9 01 8D FC 25 20
2400- 16 35 A9 05 20 72 24 AE
2408- FC 25 A9 00 20 24 ED A9
2410- 19 20 9A 31 A9 0B 8D F7
2418- 25 20 A7 31 B0 17 AD 00
2420- C0 2C 10 C0 C9 9B F0 1F
2428- 20 88 31 CE F7 25 AD F7
2430- 25 F0 0A 10 E4 C9 19 F0
2438- 0E C9 06 F0 0A EE FC 25
2440- AD FC 25 C9 0B D0 B8 20
2448- 07 35 20 71 2D 20 1E 32
2450- 4C 03 22 20 16 35 A9 03
2458- 20 72 24 4C E6 23 20 FB
2460- DA 20 FB DA 18 60 A0 03
2468- B9 F3 25 20 ED FD 88 10
2470- F7 60 A0 FF C8 D9 18 25
2478- D0 FA C8 B9 18 25 10 06
2480- 20 ED FD 4C 7A 24 60 A9
2488- 01 8D 00 14 A0 02 B9 1C
2490- 2D 99 5C 14 88 10 F7 A0
2498- 09 B9 60 2F 99 50 14 88
24A0- 10 F7 AD AE 35 8D 60 14
24A8- A8 C8 B9 6E 35 29 7F 99
24B0- 60 14 88 D0 F5 4C 10 23
24B8- CE EF ED A0 E4 F5 A0 E6
24C0- E9 E3 E8 E9 E5 F2 A0 C0
24C8- A0 F4 F2 E1 EE E6 FB F2
24D0- E5 F2 A0 BA 00 A0 E2 EC
24D8- EF E3 F3 A0 EC F5 E9 A0
24E0- F3 EF EE F4 A0 E1 E6 E6
24E8- E5 E3 F4 FB F3 AE 8D 8D
```

```
        LDA  NESSAI
        CMP  #11
        BNE  FIN0

FIN4    JSR  MOUTKCR
        LDA  #4
        JSR  MOUT

CLOTFIC JSR  CLOSEFIL   ; clore fichier
        JSR  DECONNE    ; déconnexion du Minitel
        JSR  GETRET
        JMP  ENVFICH

ANNEME0 JSR  MOUTKCR    ; on renonce
        LDA  #7
        JSR  MOUT
ANNEME  LDA  #1         ; annulation émetteur
        STA  NESSAI
ANNEME1 JSR  MOUTKCR
        LDA  #5
        JSR  MOUT
        LDX  NESSAI
        LDA  #0
        JSR  LINPTR
        LDA  #CAN
        JSR  ENVOI

        LDA  #11
        STA  ATTENTE
A32     JSR  RECOIT     ; attendre acquiescement
        BCS  A4         ; de l'annulation
        LDA  KBD        ; émetteur veut abandonner ?
        BIT  STROBE
        CMP  #$9B
        BEQ  A41
        JSR  WAIT
        DEC  ATTENTE
        LDA  ATTENTE
        BEQ  A40
        BPL  A32

A4      CMP  #CAN       ; si annulation ou compris
        BEQ  A41        ; => fini
        CMP  #ACK
        BEQ  A41

A40     INC  NESSAI     ; sinon, on recommence 10 fois
        LDA  NESSAI
        CMP  #11
        BNE  ANNEME1
A41     JSR  DECONNE
        JSR  CLOSEFIL
        JSR  GETRET
        JMP  ENVFICH

ANNREC  JSR  MOUTKCR    ; message 3
        LDA  #3
        JSR  MOUT
        JMP  CLOTFIC

MOUTCR  JSR  CROUT
```

```
Téléchargement/Téléréception
                                INTERFONS

Préfixe /UNIDISK3.5/TRANS/

Blocs libres sur ce disque : 126

Réception du fichier :
FICHIER.MAC.F

   * Attente du bloc 0
   * Bloc reçu
   * Nombre de blocs à recevoir : 17
   * Emetteur : Macintosh

Mercredi 14 Janvier 1987, 15:46.
```

```
        JSR  CROUT
        CLC
        RTS

MOUTK   LDY  #3
MOUT0   LDA  MT2,Y      ; affiche "  * " avant chaque
        JSR  COUT
        DEY
        BPL  MOUT0
        RTS

MOUT    LDY  #$FF
MOUT1   INY             ; message
        CMP  MT1,Y
        BNE  MOUT1
MOUT2   INY
        LDA  MT1,Y
        BPL  MOUT3
        JSR  COUT
        JMP  MOUT2
MOUT3   RTS

CONTBLC LDA  #1         ; prépare le bloc de controle
        STA  BUFNC      ; c'est un Apple //
        LDY  #2
CONT0   LDA  EOF,Y
        STA  BUFNC+80+12,Y
        DEY
        BPL  CONT0
        LDY  #9
CONT1   LDA  PARLSTG,Y  ; paramètres de GetFileInfo
        STA  BUFNC+80,Y
        DEY
        BPL  CONT1
        LDA  LNGPATHN   ; et nom précédé de lng
        STA  BUFNC+80+16
```

```
24F0- CD E5 F4 F4 E5 FA A0 EC
24F8- E5 A0 CD E9 EE E9 F4 E5
2500- EC A0 F3 EF F5 F3 A0 F4
2508- E5 EE F3 E9 EF EE AC A0
2510- F0 F5 E9 F3 AE AE AE 00
2518- 01 C5 ED E9 F3 F3 E9 EF
2520- EE A0 E4 F5 A0 E2 EC EF
2528- E3 A0 EE DB A0 02 FD ED
2530- E5 A0 F4 E5 EE F4 E1 F4
2538- E9 F6 E5 A0 F0 EF F5 F2
2540- A0 E2 EC EF E3 A0 EE DB
2548- A0 03 D2 FB E3 E5 F0 F4
2550- E5 F5 F2 A0 E1 A0 E4 E5
2558- ED E1 EE E4 FB A0 E1 EE
2560- EE F5 EC E1 F4 E9 EF EE
2568- 04 D4 F2 E1 EE F3 ED E9
2570- F3 F3 E9 EF EE A0 F4 E5
2578- F2 ED E9 EE FB E5 05 C1
2580- EE EE F5 EC E1 F4 E9 EF
2588- EE A0 E5 EE A0 E3 EF F5
2590- F2 F3 AC A0 E5 F3 F3 E1
2598- E9 A0 EE DB A0 06 C1 F4
25A0- F4 E5 EE F4 E5 A0 E4 F5
25A8- A0 F2 FB E3 E5 F0 F4 E5
25B0- F5 F2 AE AE AE 07 C8 FB
25B8- EC E1 F3 AC A0 E9 EC A0
25C0- E6 E1 F5 F4 A0 F2 E5 EE
25C8- EF EE E3 E5 F2 AE AE AE
25D0- 08 C1 EE EE EF EE E3 E5
25D8- A0 E4 E5 A0 E6 E9 EE A0
25E0- E4 E5 A0 F4 F2 E1 EE F3
25E8- ED E9 F3 F3 E9 EF EE AE
25F0- AE AE 00 A0 AA A0 A0 00
25F8- 00 00 00 00 00 A2 27 A9
2600- A0 9D CF 07 CA 10 FA 60
2608- 20 FD 25 A9 03 85 22 20
2610- 58 FC 20 EA 2F 20 FB DA
2618- AD AE 35 8D 6E 35 20 54
2620- 2F 90 06 20 1E 32 4C 10
2628- 2E A9 0E 20 36 29 AD 65
2630- 2F 38 ED 68 2F 8D BA 2B
2638- AA AD 66 2F ED 69 2F 8D
```

```
        TAY
        INY
CONT2   LDA  PATHNAME,Y
        AND  #%01111111
        STA  BUFNC+80+16,Y
        DEY
        BNE  CONT2
        JMP  DEC0

NOMFIC  ASC  "Nom du fichier à tranférer :"
        DFB  0
MESBLOC ASC  " blocs lui sont affectés."
        DFB  $8D,$8D
MESBLOCM ASC "Mettez le Minitel sous tension, puis..."
        DFB  0

MT1     DFB  1
        ASC  "Emission du bloc n° "
        DFB  2
        ASC  "ème tentative pour bloc n° "
        DFB  3
        ASC  "Récepteur a demandé annulation"
        DFB  4
        ASC  "Transmission terminée"
        DFB  5
        ASC  "Annulation en cours, essai n° "
        DFB  6
        ASC  "Attente du récepteur..."
        DFB  7
        ASC  "Hélas, il faut renoncer..."
        DFB  8
        ASC  "Annonce de fin de transmission..."
        DFB  0
MT2     ASC  " * "

ATTENTE DS   3
NUMBLOC DS   2
NESSAI  DS   1

EFFDATE LDX  #39        ; efface la date
        LDA  #" "
EFFDATE1 STA $7CF,X
        DEX
        BPL  EFFDATE1
        RTS

*----------------
* RECEPTION DE FICHIER
*----------------

RECFICH JSR  EFFDATE
        LDA  #3
        STA  WNDTOP
        JSR  HOME
        JSR  LITPREF
        JSR  CROUT
        LDA  LNGPATHN
        STA  PATHNAME
        JSR  GFI        ; place sur volume ?
        BCC  REC0
        JSR  GETRET
        JMP  VADEB
REC0    LDA  #14
        JSR  MOUTREC    ; l'affiche
        LDA  AUXTYPG
        SEC
        SBC  BLOCKUS
        STA  FREEVOL    ; garde place libre sur vol
        TAX
        LDA  AUXTYPG+1
        SBC  BLOCKUS+1
        STA  FREEVOL+1  ; id
        JSR  LINPTR

        LDA  FREEVOL    ; de la place sur ce vol. ?
        BNE  REC0R
        LDA  FREEVOL+1
        BNE  REC0R
        LDA  #$7D
        JSR  ERREUR
        JMP  FINRECP

REC0R   JSR  MOUTCR     ; réception du fichier...
        LDA  #1
        JSR  MOUTREC
        JSR  MOUTCR
        LDA  #10        ; limite fenetre
        JSR  LIGNE
        LDA  #11
        STA  WNDTOP
        JSR  HOME
        JSR  CROUT
        JSR  CROUT
        INC  WNDTOP

        LDY  #>MESBLOCM
        LDA  #<MESBLOCM ; mettre Minitel sous tension
        JSR  STROUT
        LDY  #>RECSTR
        LDA  #<RECSTR   ; faire RETURN ou ESC
        JSR  STROUT

RECS03  LDA  KBD
        BPL  RECS03
        BIT  STROBE
        CMP  #$8D
        BEQ  COMM
        CMP  #$9B
        BNE  RECS03
        JMP  FINRECP

COMM    JSR  HOME
        JSR  CONNECT
        LDA  #0         ; attend bloc 0
        STA  NUMBLOC
        STA  NUMBLOC+1
        STA  DRAPBL0

DEBREC  LDA  #1         ; initialise 10 essais
        STA  NESSAI
        LDA  DRAPBL0
        BEQ  REC11E
```

```
2640- BB 2B 20 24 ED AD BA 2B
2648- D0 0D AD BB 2B D0 08 A9
2650- 7D 20 D2 31 4C 29 29 20
2658- 5E 24 A9 01 20 36 29 20
2660- 5E 24 A9 0A 20 30 21 A9
2668- 0B 85 22 20 58 FC 20 FB
2670- DA 20 FB DA E6 22 A0 24
2678- A9 F0 20 3A DB A0 2A A9
2680- 26 20 3A DB AD 00 C0 10
2688- FB 2C 10 C0 C9 8D F0 07
2690- C9 9B D0 F0 4C 29 29 20
2698- 58 FC 20 E9 34 A9 00 8D
26A0- FA 25 8D FB 25 8D BC 2B
26A8- A9 01 8D FC 25 AD BC 2B
26B0- F0 05 A9 0A 8D FC 25 20
26B8- 16 35 A9 08 20 36 29 AE
26C0- FA 25 AD FB 25 20 24 ED
26C8- A9 00 8D F7 25 8D F8 25
26D0- 8D F9 25 AD FA 25 D0 0A
26D8- AD FB 25 D0 05 A9 15 20
26E0- 9A 31 20 A7 31 B0 38 EE
26E8- F7 25 D0 08 EE F8 25 D0
26F0- 03 EE F9 25 AD F7 25 D0
26F8- 0E AD F8 25 C9 87 D0 07
2700- AD F9 25 C9 02 F0 0C AD
2708- 00 C0 2C 10 C0 C9 9B F0
2710- 35 D0 CF EE FC 25 AD FC
2718- 25 C9 0B D0 9A F0 27 C9
2720- 04 D0 03 4C 10 29 C9 19
2728- D0 03 4C FA 28 C9 01 D0
2730- 03 4C 5C 27 A2 0A 20 88
2738- 31 CA D0 FA 20 A7 31 A9
2740- 15 20 9A 31 D0 9C 20 16
2748- 35 A9 09 20 36 29 20 5E
2750- 24 A9 19 20 9A 31 20 1C
2758- 35 4C 29 29 20 BB 30 EE
2760- F7 07 A9 00 8D 87 31 20
2768- B7 31 B0 03 4C E5 28 29
2770- 3F 8D FE 16 20 B7 31 B0
2778- 03 4C E5 28 29 3F 8D FF
2780- 16 20 B7 31 B0 03 4C E5
2788- 28 EE F7 07 20 B7 31 B0
```

```
        LDA  #10
        STA  NESSAI

REC11E  JSR  MOUTKCR    ; message attente bloc n°
        LDA  #8
        JSR  MOUTREC
        LDX  NUMBLOC
        LDA  NUMBLOC+1
        JSR  LINPTR

        LDA  #0
        STA  ATTENTE
        STA  ATTENTE+1
        STA  ATTENTE+2
        LDA  NUMBLOC    ; si bloc 0 envoyer
        BNE  REC00      ; régulièrement NACK
        LDA  NUMBLOC+1
        BNE  REC00
        LDA  #NACK
        JSR  ENVOI

REC00   JSR  RECOIT     ; attend qqchose
        BCS  REC1
        INC  ATTENTE    ; pendant 10 secondes
        BNE  REC001
        INC  ATTENTE+1
        BNE  REC001
        INC  ATTENTE+2
REC001  LDA  ATTENTE
        BNE  REC002
        LDA  ATTENTE+1
        CMP  #135
        BNE  REC002
        LDA  ATTENTE+2
        CMP  #2
        BEQ  REC003

REC002  LDA  KBD
        BIT  STROBE     ; si ESC on renonce
        CMP  #$9B
        BEQ  RECREN
        BNE  REC00

REC003  INC  NESSAI
        LDA  NESSAI
        CMP  #11
        BNE  REC11E
        BEQ  RECREN     ; on abandonne après 10 essais

REC1    CMP  #EOT       ; on a reçu EOT ?
        BNE  REC1A
        JMP  RECEOT
REC1A   CMP  #CAN       ; CAN ?
        BNE  REC1B
        JMP  RECCAN
REC1B   CMP  #SOH       ; début bloc ?
        BNE  REC1C
        JMP  RECBLOC

REC1C   LDX  #10
REC11   JSR  WAIT       ; autre chose ?
        DEX             ; attendre 9 secondes puis NACK
        BNE  REC11
        JSR  RECOIT     ; vide carte au cas où
        LDA  #NACK
        JSR  ENVOI
        BNE  REC00

RECREN  JSR  MOUTKCR    ; récepteur abandonne
        LDA  #9
        JSR  MOUTREC
        JSR  MOUTCR
        LDA  #CAN
        JSR  ENVOI
        JSR  CDG
        JMP  FINRECP

RECBLOC JSR  INITCOD    ; reçu SOH, c'est un bloc
        INC  $7F7
        LDA  #0
        STA  CHECKSUM

        JSR  RECOIT1
        BCS  RECS1
        JMP  RECERR
RECS1   AND  #%00111111
        STA  BUFC-2     ; sauve n° bloc

        JSR  RECOIT1
        BCS  RECS2
        JMP  RECERR
RECS2   AND  #%00111111
        STA  BUFC-1     ; sauve inverse n° bloc

        JSR  RECOIT1    ; code inutile pour Apple //
        BCS  RECBLOC1
        JMP  RECERR

RECBLOC1 INC $7F7
        JSR  RECOIT1    ; réception du bloc
        BCS  RECS5
        JMP  RECERR
RECS5   LDY  #0
        STA  (COMPTC),Y
        CLC
        ADC  CHECKSUM
        AND  #%00111111
        STA  CHECKSUM
        INC  COMPTC
        BNE  RECBLOC2
        INC  COMPTC+1
RECBLOC2 LDA COMPTC
        CMP  #$AC
        BNE  RECBLOC1
        LDA  COMPTC+1
        CMP  #$19
        BNE  RECBLOC1
        LDA  #" "
        STA  $7F7

        JSR  RECOIT1    ; reçoit checksum
        BCS  RECS04
        JMP  RECERR
```

```
2790- 03 4C E5 28 A0 00 91 1A
2798- 18 6D 87 31 29 3F 8D 87
27A0- 31 E6 1A D0 02 E6 1B A5
27A8- 1A C9 AC D0 DC A5 1B C9
27B0- 19 D0 D6 A9 A0 8D F7 07
27B8- 20 B7 31 B0 03 4C E5 28
27C0- 29 3F CD 87 31 F0 03 4C
27C8- E5 28 AD FE 16 49 3F CD
27D0- FF 16 F0 03 4C E5 28 AD
27D8- FA 25 29 3F CD FE 16 F0
27E0- 35 90 15 EE FE 16 AD FA
27E8- 25 29 3F CD FE 16 D0 08
27F0- A9 06 20 9A 31 4C A8 26
27F8- A9 19 20 9A 31 20 16 35
2800- A9 0D 20 36 29 20 16 35
2808- A9 09 20 36 29 20 5E 24
2810- 20 1C 35 4C 08 26 20 CA
2818- 30 20 16 35 A9 0A 20 36
2820- 29 AD BC 2B F0 13 AD BD
2828- 2B D0 03 20 9A 32 20 31
2830- 2D 90 03 4C A7 28 4C C5
2838- 28 20 4B 29 20 C4 29 20
2840- 16 35 A9 02 20 36 29 AE
2848- 58 14 AD 59 14 20 24 ED
2850- 20 16 35 A9 03 20 36 29
2858- AD 00 14 18 69 04 20 36
2860- 29 AD BD 2B D0 25 AD 54
2868- 14 C9 04 F0 1E 20 88 31
2870- A9 19 20 9A 31 20 16 35
2878- A0 2B A9 76 20 3A DB 20
2880- FB DA 20 82 32 20 1C 35
2888- 4C 29 29 AD BB 2B CD 59
2890- 14 F0 04 B0 0A 90 3E AD
2898- BA 2B CD 58 14 90 36 20
28A0- D5 29 90 0E 20 D2 31 A9
28A8- 19 20 9A 31 20 1C 35 4C
28B0- 08 26 AD AE 35 8D 6E 35
28B8- 20 1F 2D B0 EA AD BC 2B
28C0- D0 03 CE BC 2B EE FA 25
28C8- D0 03 EE FB 25 A9 06 20
28D0- 9A 31 4C A8 26 A9 7D 20
28D8- D2 31 A9 19 20 9A 31 20
```

```
RECS04   AND  #%00111111
         CMP  CHECKSUM   ; le vérifie
         BEQ  RECS4
         JMP  RECERR
RECS4    LDA  BUFC-2     ; n° bloc valide ?
         EOR  #%00111111
         CMP  BUFC-1
         BEQ  RECS41
         JMP  RECERR
RECS41   LDA  NUMBLOC    ; vérifier n° bloc
         AND  #%00111111
         CMP  BUFC-2
         BEQ  RSUITE
         BCC  RECMANQ

DEJAEU   INC  BUFC-2     ; est-ce le précédent ?
         LDA  NUMBLOC    ; non, on renonce
         AND  #%00111111
         CMP  BUFC-2
         BNE  RECMANQ    ; oui, on
         LDA  #ACK       ; envoie ack, boucler
         JSR  ENVOI      ; bloc déjà reçu
         JMP  DEBREC

RECMANQ  LDA  #CAN       ; on en a raté, renoncer
         JSR  ENVOI
         JSR  MOUTKCR
         LDA  #13
         JSR  MOUTREC
         JSR  MOUTKCR
         LDA  #9
         JSR  MOUTREC
         JSR  MOUTCR
         JSR  CDG        ; CDG = clore fichier, déconnecter
         JMP  RECFICH    ; et attendre RETURN

RSUITE   JSR  DEBUDEC    ; tvb, on décode
         JSR  MOUTKCR
         LDA  #10        ; bloc reçu
         JSR  MOUTREC
         LDA  DRAPBL0
         BEQ  RSUITE0    ; bloc 0, on crée et on ouvre
         LDA  TYPEAPP
         BNE  SUITTRA
         JSR  DECODMAC
SUITTRA  JSR  EF         ; enregistrer un bloc
         BCC  RECENROK
         JMP  ASSEZERR
RECENROK JMP  ASSEZ2

RSUITE0  JSR  FAITNOM    ; constitue le nom
         JSR  AFFNOM     ; l'affiche
         JSR  MOUTKCR    ; affiche nb blocs du fichier
         LDA  #2
         JSR  MOUTREC
         LDX  BUFNC+88   ; à recevoir
         LDA  BUFNC+89
         JSR  LINPTR

         JSR  MOUTKCR    ; quel ordinateur émet ?
         LDA  #3
         JSR  MOUTREC    ; on affiche son nom
         LDA  BUFNC
         CLC
         ADC  #4
         JSR  MOUTREC

         LDA  TYPEAPP    ; ça vient d'un Mac ?
         BNE  RSUITE01
         LDA  BUFNC+84
         CMP  #4         ; c'est du texte ?
         BEQ  RSUITE01
         JSR  WAIT
         LDA  #CAN
         JSR  ENVOI      ; non, alors annule
         JSR  MOUTKCR
         LDY  #>FICINC   ; car
         LDA  #<FICINC   ; inexploitable
         JSR  STROUT
         JSR  CROUT
         JSR  BIP
         JSR  CDG
         JMP  FINRECP
RSUITE01 LDA  FREEVOL+1  ; controle place disponible
         CMP  BUFNC+89   ; poids forts
         BEQ  ASS
         BCS  ASSEZ
         BCC  TROPEU
ASS      LDA  FREEVOL    ; poids faibles
         CMP  BUFNC+88
         BCC  TROPEU

ASSEZ    JSR  CREFIC     ; on créé le fichier
         BCC  ASSEZ11
         JSR  ERREUR

ASSEZERR LDA  #CAN       ; problème à la création,
         JSR  ENVOI      ; on demande annulation
         JSR  CDG
         JMP  RECFICH

ASSEZ11  LDA  LNGPATHN   ; on ouvre le fichier créé
         STA  PATHNAME
         JSR  OUVREFIC
         BCS  ASSEZERR
         LDA  DRAPBL0
         BNE  ASSEZ2
         DEC  DRAPBL0
ASSEZ2   INC  NUMBLOC
         BNE  ASSEZ1
         INC  NUMBLOC+1
ASSEZ1   LDA  #ACK       ; on acquiesce pour bloc 0
         JSR  ENVOI
         JMP  DEBREC

TROPEU   LDA  #$7D       ; pas de place
         JSR  ERREUR
         LDA  #CAN       ; on demande annulation
         JSR  ENVOI
         JSR  CDG
         JMP  FINRECP
```

```
28E0- 1C 35 4C 29 29 20 16 35
28E8- A9 0B 20 36 29 A9 A0 8D
28F0- F7 07 A9 15 20 9A 31 4C
28F8- A8 26 20 16 35 A9 0C 20
2900- 36 29 20 5E 24 A9 06 20
2908- 9A 31 20 1C 35 4C 29 29
2910- 20 5A 2D A9 06 20 9A 31
2918- 20 16 35 A9 0F 20 36 29
2920- 20 5E 24 20 1C 35 4C 29
2928- 29 A9 03 85 22 20 58 FC
2930- 20 EA 2F 4C D8 20 A0 FF
2938- C8 D9 83 2A D0 FA C8 B9
2940- 83 2A 10 06 20 ED FD 4C
2948- 3E 29 60 AD 00 14 8D BD
2950- 2B F0 2F AC 60 14 B9 61
2958- 14 09 80 C9 AF F0 06 88
2960- D0 F4 88 30 05 CC 60 14
2968- F0 F5 A2 FF E8 C8 B9 61
2970- 14 09 80 9D 6F 35 CC 60
2978- 14 D0 F1 8E 6E 35 8E AE
2980- 35 60 AC B6 29 B9 B6 29
2988- 99 6E 35 88 10 F7 AD 6E
2990- 35 8D AE 35 AE A0 14 BD
2998- A0 14 A0 00 D9 25 35 F0
29A0- 08 C8 C8 C0 42 D0 F5 F0
29A8- 04 C8 B9 25 35 09 80 9D
29B0- CF 07 CA D0 E2 60 0D C6
29B8- C9 C3 C8 C9 C5 D2 AE CD
29C0- C1 C3 AE C1 A9 09 20 47
29C8- F8 AC 6E 35 B9 6E 35 88
29D0- 91 26 D0 F8 60 A9 C3 8D
29D8- 05 2D A0 02 B9 54 14 99
29E0- 06 2D 88 10 F7 A9 01 8D
29E8- 09 2D 20 EA 2C B0 0C A0
29F0- 02 B9 5C 14 99 6E 2D 88
29F8- 10 F7 60 C9 47 F0 02 38
2A00- 60 AE 6E 35 FE 6E 35 BD
2A08- 6E 35 C9 BA F0 0D C9 DB
2A10- F0 10 9D 6E 35 20 C4 29
2A18- 4C D5 29 A9 C1 9D 6E 35
2A20- D0 F3 A9 B0 D0 F7 8D 8D
2A28- 8D 8D A0 A0 BC D2 C5 D4
```

```
RECERR   JSR  MOUTKCR    ; bloc mal reçu
         LDA  #11
         JSR  MOUTREC
         LDA  #" "
         STA  $7F7
         LDA  #NACK
         JSR  FNVOI
         JMP  DEBREC

RECCAN   JSR  MOUTKCR    ; reçu CAN
         LDA  #12
         JSR  MOUTREC
         JSR  MOUTCR
         LDA  #ACK       ; on acquièsce et
         JSR  ENVOI      ; on annule aussi
         JSR  CDG
         JMP  FINRECP

RECEOT   JSR  SETFINF    ; reçu EOT
         LDA  #ACK       ; on acquièsce
         JSR  ENVOI      ; et on ferme
         JSR  MOUTKCR    ; transmission OK
         LDA  #15
         JSR  MOUTREC
         JSR  MOUTCR
         JSR  CDG
         JMP  FINRECP

FINRECP  LDA  #3
         STA  WNDTOP
         JSR  HOME
         JSR  LITPREF
         JMP  CHOIX

MOUTREC  LDY  #$FF       ; affiche un message
MOUTRC1  INY
         CMP  MESSREC,Y
         BNE  MOUTRC1
MOUTRC2  INY
         LDA  MESSREC,Y
         BPL  MOUTRC3
         JSR  COUT
         JMP  MOUTRC2
MOUTRC3  RTS

FAITNOM  LDA  BUFNC
         STA  TYPEAPP
         BEQ  MAC        ; Si MAC émetteur, on renomme
         LDY  BUFNC+96   ; charge long pathname reçu
RECH0    LDA  BUFNC+97,Y
         ORA  #$80       ; tronque au dernier éventuel /
         CMP  #"/"
         BEQ  RECH1
RECH00   DEY
         BNE  RECH0
         DEY
         BMI  RECH11
RECH1    CPY  BUFNC+96
         BEQ  RECH00
RECH11   LDX  #$FF
RECH2    INX
         INY
         LDA  BUFNC+97,Y
         ORA  #$80
         STA  PATHNAME+1,X
         CPY  BUFNC+96
         BNE  RECH2
         STX  PATHNAME
         STX  LNGPATHN
         RTS
MAC      LDY  FICMAC     ; c'est un mac, on change le nom
MAC1     LDA  FICMAC,Y   ; (caractères exotiques)
         STA  PATHNAME,Y
         DEY
         BPL  MAC1
         LDA  PATHNAME
         STA  LNGPATHN

         LDX  BUFNC+160  ; affiche date
DATEMAC  LDA  BUFNC+160,X
         LDY  #0         ; en recodant carac du Mac
DATEMAC0 CMP  ACCENT,Y
         BEQ  DATEMAC1
         INY
         INY
         CPY  #66
         BNE  DATEMAC0
         BEQ  DATEMAC2
DATEMAC1 INY
         LDA  ACCENT,Y
DATEMAC2 ORA  #%10000000
         STA  $7CF,X
         DEX
         BNE  DATEMAC
         RTS

FICMAC   STR  "FICHIER.MAC.A"

AFFNOM   LDA  #9         ; affiche le nom reçu, tronqué
         JSR  GBASCALC
         LDY  PATHNAME
RECNM3   LDA  PATHNAME,Y
         DEY
         STA  (GBASL),Y
         BNE  RECNM3
         RTS

CREFIC   LDA  #$C3
         STA  ACCESBT    ; (unlock)
         LDY  #2
CREFIC1  LDA  BUFNC+80+4,Y
         STA  FILETYPE,Y ; type de fichier et
         DEY             ; aux file type
         BPL  CREFIC1
         LDA  #1
         STA  STORAGE    ; stockage seedling en standard
         JSR  CREFICH
         BCS  CREFIC2
         LDY  #2
EOF1     LDA  BUFNC+92,Y ; sauve l'EOF
         STA  POSEOF,Y
         DEY
```

```
2A30- D5 D2 CE BE A0 BD A0 E3
2A38- EF ED ED E5 EE E3 E5 F2
2A40- A0 EC E1 A0 F2 FB E3 E5
2A48- F0 F4 E9 EF EE 8D A0 A0
2A50- A0 A0 A0 A0 A0 A0 A0 A0
2A58- A0 A0 A0 F3 F5 F2 A0 E3
2A60- E5 A0 F6 EF EC F5 ED E5
2A68- 8D 8D A0 A0 A0 A0 A0 BC
2A70- C5 D3 C3 BE A0 BD A0 F2
2A78- E5 EE EF EE E3 E5 F2 AE
2A80- AE AE 00 01 D2 FB E3 E5
2A88- F0 F4 E9 EF EE A0 E4 F5
2A90- A0 E6 E9 E3 E8 E9 E5 F2
2A98- A0 BA A0 02 CE EF ED E2
2AA0- F2 E5 A0 E4 E5 A0 E2 EC
2AA8- EF E3 F3 A0 C0 A0 F2 E5
2AB0- E3 E5 F6 EF E9 F2 A0 BA
2AB8- A0 03 C5 ED E5 F4 F4 E5
2AC0- F5 F2 A0 BA A0 04 CD E1
2AC8- E3 E9 EE F4 EF F3 E8 05
2AD0- C1 F0 F0 EC E5 A0 AF AF
2AD8- 06 C9 C2 CD 07 D3 E5 F2
2AE0- F6 E5 F5 F2 A0 D0 EF ED
2AE8- A7 F3 08 C1 F4 F4 E5 EE
2AF0- F4 E5 A0 E4 F5 A0 E2 EC
2AF8- EF E3 A0 09 CF EE A0 F2
2B00- E5 EE EF EE E3 E5 0A C2
2B08- EC EF E3 A0 F2 E5 DC F5
2B10- 0B D2 FB FB ED E9 F3 F3
2B18- E9 EF EE A0 E4 F5 A0 E2
2B20- EC EF E3 A0 E4 E5 ED E1
2B28- EE E4 FB E5 0C C5 ED E5
2B30- F4 F4 E5 F5 F2 A0 F2 E5
2B38- EE EF EE E3 E5 0D C2 EC
2B40- EF E3 A0 ED E1 EE F1 F5
2B48- E1 EE F4 0E C2 EC EF E3
2B50- F3 A0 EC E9 E2 F2 E5 F3
2B58- A0 F3 F5 F2 A0 E3 E5 A0
2B60- E4 E9 F3 F1 F5 E5 A0 BA
2B68- A0 0F D4 E5 F2 ED E9 EE
2B70- FB AC A0 CF CB 00 C3 E5
2B78- A0 E6 E9 E3 E8 E9 E5 F2
```

```
        BPL  EOF1
        RTS
CREFIC2 CMP  #$47        ; nom existe déjà ?
        BEQ  CREFIC3
        SEC
        RTS
CREFIC3 LDX  PATHNAME    ; si oui, on incrémente le dernier
        INC  PATHNAME,X  ; caractère entre 0 et Z en évitant
        LDA  PATHNAME,X  ; les codes interdits
        CMP  #":"
        BEQ  CREFIC4
        CMP  #$DB        ; est-ce un "
        BEQ  CREFIC5
        STA  PATHNAME,X
CREFIC31 JSR AFFNOM
        JMP  CREFIC
CREFIC4 LDA  #"A"
CREFIC40 STA PATHNAME,X
        BNE  CREFIC31
CREFIC5 LDA  #"0"
        BNE  CREFIC40

RECSTR  DFB  $8D,$8D,$8D,$8D
        ASC  "  <RETURN> = commencer la réception"
        DFB  $8D
        ASC  "              sur ce volume"
        DFB  $8D,$8D
        ASC  "     <ESC> = renoncer..."
        DFB  0

MESSREC DFB  1
        ASC  "Réception du fichier : "
        DFB  2
        ASC  "Nombre de blocs à recevoir : "
        DFB  3
        ASC  "Emetteur : "
        DFB  4
        ASC  "Macintosh"
        DFB  5
        ASC  "Apple //"
        DFB  6
        ASC  "IBM"
        DFB  7
        ASC  "Serveur Pom's"
        DFB  8
        ASC  "Attente du bloc "
        ASC  "On renonce"
        DFB  10
        ASC  "Bloc reçu"
        DFB  11
        ASC  "Réémission du bloc demandée"
        DFB  12
        ASC  "Emetteur renonce"
        DFB  13
        ASC  "Bloc manquant"
        DFB  14
        ASC  "Blocs libres sur ce disque : "
        DFB  15
        ASC  "Terminé, OK"
        DFB  0

FICINC  ASC  "Ce fichier Mac est incompatible"
        DFB  $8D
        ASC  "    (seul le format TEXT convient)"
        DFB  $8D,0

FREEVOL DS   2
DRAPBL0 DS   1
TYPEAPP DS   1

*----------------
* QUITTER
*----------------

QUITTER JSR  TEXT
        JSR  HOME
        JMP  BASIC

*----------------
* INPUT
*----------------
INPUT   LDX  #0
        STX  POSC        ; position du curseur

        LDY  LONG        ; installe les points
        LDA  #"."
INPOINT DEY
        BEQ  INCLAV
        STA  (GBASL),Y
        BNE  INPOINT

INCLAV  LDA  #")"        ; initialise le curseur
        STA  (GBASL),Y

INCLAV1 LDA  (GBASL),Y   ; curseur clignotant
        EOR  #1
        STA  (GBASL),Y
        LDX  #$2C        ; dosage du scintillement
INCLAV11 DEX
        BNE  INCLAV11
        LDA  KBD
        BPL  INCLAV1
        BIT  STROBE

        CMP  #$9B        ; esc ?
        BEQ  INESC
        CMP  #$88        ; flèche gauche ?
        BEQ  INRETOUR
        CMP  #$FF        ; del ?
        BEQ  INRETOUR
        CMP  #$8D        ; return ?
        BEQ  INVALID
        CPY  LONG        ; dernier carac ?
        BEQ  INERR       ; oui, alors pas d'autre
        CMP  #"."        ; . ?
        BCC  INERR
        CMP  #"/"        ; / ?
        BEQ  SLASH
        CMP  #"A"        ; une lettre ?
        BCC  CHIFFRE
        AND  #%11011111  ; convertit minusc en MAJ
        CMP  #$DB        ; = "
```

```
2B80- A0 CD E1 E3 A0 E5 F3 F4
2B88- A0 E9 EE E3 EF ED F0 E1
2B90- F4 E9 E2 EC E5 8D A0 A0
2B98- A0 A0 A8 F3 E5 F5 EC A0
2BA0- EC E5 A0 E6 EF F2 ED E1
2BA8- F4 A0 D4 C5 D8 D4 A0 E3
2BB0- EF EE F6 E9 E5 EE F4 A9
2BB8- 8D 00 00 00 00 00 20 39
2BC0- FB 20 58 FC 4C D0 03 A2
2BC8- 00 8E B6 2C AC B7 2C A9
2BD0- AE 88 F0 04 91 26 D0 F9
2BD8- A9 A9 91 26 B1 26 49 01
2BE0- 91 26 A2 2C CA D0 FD AD
2BE8- 00 C0 10 F0 2C 10 C0 C9
2BF0- 9B F0 44 C9 88 F0 51 C9
2BF8- FF F0 4D C9 8D F0 6B CC
2C00- B7 2C F0 2E C9 AE 90 2A
2C08- C9 AF F0 0C C9 C1 90 0E
2C10- 29 DF C9 DB B0 1C 90 49
2C18- C0 00 F0 45 D0 08 C9 BA
2C20- B0 10 C0 00 F0 0C AA 88
2C28- B1 26 C8 C9 AF F0 03 8A
2C30- 30 2F 20 82 32 B0 A5 AC
2C38- B7 2C A9 A0 91 26 88 10
2C40- FB A9 00 8D 8E 2C 38 60
2C48- AD B6 2C F0 E5 CD B7 2C
2C50- D0 04 A9 A0 D0 02 A9 AE
2C58- 91 26 88 CE B6 2C 4C D8
2C60- 2B 91 26 EE B6 2C C8 4C
2C68- D8 2B 8C B8 2C A9 A0 91
2C70- 26 CC B7 2C F0 03 C8 D0
2C78- F6 AC B8 2C A9 00 99 8E
2C80- 2C 88 30 08 B1 26 99 8E
2C88- 2C 88 10 F8 18 60 00 00
2C90- 00 00 00 00 00 00 00 00
2C98- 00 00 00 00 00 00 00 00
2CA0- 00 00 00 00 00 00 00 00
2CA8- 00 00 00 00 00 00 00 00
2CB0- 00 00 00 00 00 00 00 00
2CB8- 00 20 58 FC A0 00 B9 E5
2CC0- 2C 99 00 02 C9 8D F0 03
2CC8- C8 D0 F3 20 03 BE C9 06
```

```
        BCS  INERR       ; > à Z
        BCC  INLETTRE

SLASH   CPY  #0          ; / est autorisé en premier
        BEQ  INLETTRE    ; caractère, mais pas après
        BNE  CHIFFRE1    ; un autre /

CHIFFRE CMP  #":"        ; c'est bien un chiffre ?
        BCS  INERR       ; non, c'est :;<=>? ou bien à
        CPY  #0          ; c'est le 1er carac ?
        BEQ  INERR       ; oui, pas de chiffre
CHIFFRE1 TAX             ; sauve carac
        DEY              ; le précédent est un / ?
        LDA  (GBASL),Y
        INY
        CMP  #"/"
        BEQ  INERR       ; oui alors pas de chiffre
        TXA              ; restaure carac
        BMI  INLETTRE    ; ok

INERR   JSR  BIP
        BCS  INCLAV1

INESC   LDY  LONG
        LDA  #" "
INESC1  STA  (GBASL),Y
        DEY
        BPL  INESC1
        LDA  #0
        STA  BUFFIN
        SEC              ; esc --> carry set
        RTS

INRETOUR LDA POSC        ; 1er carac ?
        BEQ  INERR       ; oui, pas de retour
        CMP  LONG        ; si dernier, remplace
        BNE  INR1        ; par espace
        LDA  #" "        ; sinon par un point
        BNE  INR2
INR1    LDA  #"."
INR2    STA  (GBASL),Y
        DEY
        DEC  POSC
        JMP  INCLAV

INLETTRE STA (GBASL),Y   ; affiche carac
        INC  POSC
        INY
        JMP  INCLAV

INVALID STY  TPS         ; on efface curseur
        LDA  #" "        ; et points restants
INVALID0 STA (GBASL),Y
        CPY  LONG
        BEQ  INVALID1
        INY
        BNE  INVALID0

INVALID1 LDY TPS
        LDA  #0          ; finir la chaine par 0
        STA  BUFFIN,Y
        DEY
        BMI  INVALID3    ; chaine vide = rien à recopier
INVALID2 LDA (GBASL),Y   ; stocke la chaine
        STA  BUFFIN,Y
        DEY
        BPL  INVALID2
INVALID3 CLC             ; quitte avec carry clear
        RTS

BUFFIN  DS   40
POSC    DS   1
LONG    DS   1
TPS     DS   1
```

Source INTERPOMS.3

```
*----------------
* CATALOG
*----------------

CATAL   JSR  HOME
        LDY  #0          ; Demande Catalog au
CAT0    LDA  CMD,Y       ; Basic Système
        STA  $200,Y      ; en stockant la commande
        CMP  #$8D        ; dans buffer clavier
        BEQ  CAT1
        INY
        BNE  CAT0

CAT1    JSR  $BE03
        CMP  #6
        BEQ  OKCAT0
        CMP  #8
        BNE  OKCAT
OKCAT0  JSR  ERREUR
OKCAT   JSR  GETRET      ; Attend Return
        JSR  HOME
        JSR  LITPREF     ; Réaffiche préfixe
        JMP  CHOIX       ; retourne au menu

CMD     ASC  "CAT"
        DFB  $8D
PP      DS   1

*----------------
* CHERCHE DATE/HEURE
* CREE FICHIER
*----------------

CREFICH JSR  MLI         ; répupère l'heure
        DFB  GETTIME     ; s'il y a une carte
        DA   0           ; horloge (non testé)
        LDY  #3
BCLC    LDA  $BF93,Y
        STA  DATECRE,Y
        DEY
        BPL  BCLC
```

```
2CD0- F0 04 C9 08 D0 03 20 D2
2CD8- 31 20 1E 32 20 58 FC 20
2CE0- EA 2F 4C D8 20 C3 C1 D4
2CE8- 8D 00 20 00 BF 82 00 00
2CF0- A0 03 B9 93 BF 99 0A 2D
2CF8- 88 10 F7 20 00 BF C0 02
2D00- 2D 60 07 6E 35 00 00 00
2D08- 00 00 00 00 00 00 20 00
2D10- BF D1 1A 2D 90 03 20 D2
2D18- 31 60 02 00 00 00 00 20
2D20- 00 BF C8 2B 2D 90 03 20
2D28- D2 31 60 03 6E 35 00 10
2D30- 00 AD 30 2D 8D 44 2D 20
2D38- 00 BF CB 43 2D 90 03 20
2D40- D2 31 60 04 00 00 14 00
2D48- 02 00 00 AD 30 2D 8D 44
2D50- 2D 20 00 BF CA 43 2D 90
2D58- 00 60 AD 30 2D 8D 6D 2D
2D60- 20 00 BF D0 6C 2D 90 03
2D68- 20 D2 31 60 02 00 00 00
2D70- 00 AD 30 2D 8D 7F 2D 20
2D78- 00 BF CC 7E 2D 60 01 00
2D80- 20 58 FC A0 2F A9 1F 20
2D88- 3A DB 20 FB DA 20 00 BF
2D90- C5 19 2E 90 03 20 D2 31
2D98- A9 0F 8D 1D 2F AD 1D 2F
2DA0- 0A 0A 0A 0A A8 B9 1D 2E
2DA8- 8D 1E 2F 29 0F F0 59 A2
2DB0- 00 BD 47 2F F0 06 20 ED
2DB8- FD E8 D0 F5 AD 1E 2F 4A
2DC0- 4A 4A 4A 29 07 09 B0 20
2DC8- ED FD A2 00 BD 4D 2F F0
2DD0- 06 20 ED FD E8 D0 F5 A9
2DD8- B1 2C 1E 2F 10 02 A9 B2
2DE0- 20 ED FD A9 A0 20 ED FD
2DE8- AD 1E 2F 29 0F 8D 1E 2F
2DF0- A9 AF 20 ED FD C8 B9 1D
2DF8- 2E 09 80 20 ED FD CE 1E
2E00- 2F D0 F2 A9 8D 20 ED FD
2E08- CE 1D 2F 10 90 20 1E 32
2E10- 20 58 FC 20 EA 2F 4C D8
2E18- 20 02 00 1D 2E 00 00 00
```

```
        JSR  MLI        ; créé le fichier
        DFB  CREATE
        DA   PARLST
        RTS
PARLST  DFB  7
ADPATHN DA   PATHNAME
ACCESBT DS   1
FILETYPE DS  1
AUXFLTP DS   2
STORAGE DS   1
DATECRE DS   2
HEURCRE DS   2

*----------------
* GET EOF
*----------------

GETLONG JSR  MLI        ; longueur logique du
        DFB  GETEOF     ; fichier ?
        DA   PAREOF
        BCC  OKGET
        JSR  ERREUR
OKGET   RTS
PAREOF  DFB  2
REFNUMG DS   1
EOF     DS   3

*----------------
* OUVRE FICHIER
*----------------

OUVREFIC JSR MLI        ; ouvre le fichier
        DFB  OPEN
        DA   PARLSTO
        BCC  OKO
        JSR  ERREUR
OKO     RTS
PARLSTO DFB  3
        DA   PATHNAME
        DA   FILEBUF
REFNUM  DS   1

*----------------
* ECRIT FICHIER
*----------------

EF      LDA  REFNUM     ; écrit un bloc de
        STA  REFNUMW    ; 512 caractères
        JSR  MLI
        DFB  WRITE
        DA   PARLSTW
        BCC  OKW
        JSR  ERREUR
OKW     RTS
PARLSTW DFB  4
REFNUMW DS   1
        DA   BUFNC
        DA   512
        DS   2

*----------------
```

```
* LIT FICHIER
*----------------

LITFIC  LDA  REFNUM     ; lit un bloc de 512
        STA  REFNUMW    ; caractères
        JSR  MLI
        DFB  READ
        DA   PARLSTW
        BCC  OKL
OKL     RTS

*----------------
* AJUSTE FIN DE FICHIER
*----------------

SETFINF LDA  REFNUM     ; en fonction de la longueur
        STA  REFNUM7    ; logique reçue.
        JSR  MLI        ; A défaut, la longueur
        DFB  SETEOF     ; physique (n X 512) serait
        DA   PARLST7    ; retenue
        BCC  OKSETEOF
        JSR  ERREUR
OKSETEOF RTS
PARLST7 DFB  2
REFNUM7 DS   1
POSEOF  DS   3

*----------------
* CLOT FICHIER
*----------------

CLOSEFIL LDA REFNUM     ; clot tout fichier
        STA  REFNUMC    ; sans préciser de n°
        JSR  MLI        ; de référence
        DFB  CLOSE
        DA   PARLSTC
        RTS
PARLSTC DFB  1
REFNUMC DFB  0

*----------------
* ONLINE
*----------------

ONL     JSR  HOME       ; quels sont les
        LDY  #>MESONL   ; volumes en lignes ?
        LDA  #<MESONL
        JSR  STROUT
        JSR  CROUT
        JSR  MLI
        DFB  ONLINE
        DA   PARLSTN
        BCC  ONLINE0
        JSR  ERREUR
ONLINE0 LDA  #15        ;examine les 15 x 16 octets
        STA  CNT
ONLINED LDA  CNT
        ASL
        ASL
        ASL
        ASL
```

```
2E20- 00 00 00 00 00 00 00 00
2E28- 00 00 00 00 00 00 00 00
2E30- 00 00 00 00 00 00 00 00
2E38- 00 00 00 00 00 00 00 00
2E40- 00 00 00 00 00 00 00 00
2E48- 00 00 00 00 00 00 00 00
2E50- 00 00 00 00 00 00 00 00
2E58- 00 00 00 00 00 00 00 00
2E60- 00 00 00 00 00 00 00 00
2E68- 00 00 00 00 00 00 00 00
2E70- 00 00 00 00 00 00 00 00
2E78- 00 00 00 00 00 00 00 00
2E80- 00 00 00 00 00 00 00 00
2E88- 00 00 00 00 00 00 00 00
2E90- 00 00 00 00 00 00 00 00
2E98- 00 00 00 00 00 00 00 00
2EA0- 00 00 00 00 00 00 00 00
2EA8- 00 00 00 00 00 00 00 00
2EB0- 00 00 00 00 00 00 00 00
2EB8- 00 00 00 00 00 00 00 00
2EC0- 00 00 00 00 00 00 00 00
2EC8- 00 00 00 00 00 00 00 00
2ED0- 00 00 00 00 00 00 00 00
2ED8- 00 00 00 00 00 00 00 00
2EE0- 00 00 00 00 00 00 00 00
2EE8- 00 00 00 00 00 00 00 00
2EF0- 00 00 00 00 00 00 00 00
2EF8- 00 00 00 00 00 00 00 00
2F00- 00 00 00 00 00 00 00 00
2F08- 00 00 00 00 00 00 00 00
2F10- 00 00 00 00 00 00 00 00
2F18- 00 00 00 00 00 00 00 8D
2F20- D6 EF EC F5 ED E5 F3 A0
2F28- E5 EE A0 EC E9 E7 EE E5
2F30- A0 BA 8D AD AD AD AD AD
2F38- AD AD AD AD AD AD AD AD
2F40- AD AD AD AD AD 8D 00 D0
2F48- EF F2 F4 A0 00 A0 CC E5
2F50- E3 F4 A0 00 20 00 BF C4
2F58- 60 2F 90 03 20 D2 31 60
2F60- 0A 6E 35 00 00 00 00 00
2F68- 00 00 00 00 00 00 00 00
```

```
         TAY
         LDA  BUFONLIN,Y ;entete 1er lecteur
         STA  LNG
         AND  #%00001111
         BEQ  ERR        ;longueur = 0 ?
         LDX  #0
ONLINEG  LDA  MESON0,X   ;envoie 'slot'
         BEQ  ONLINEH
         JSR  COUT
         INX
         BNE  ONLINEG
ONLINEH  LDA  LNG
         LSR             ;récupère n° slot
         LSR
         LSR
         LSR
         AND  #%00000111
         ORA  #%10110000
         JSR  COUT
         LDX  #0
ONLINEE  LDA  MESON1,X   ;envoie 'drive'
         BEQ  ONLINEF
         JSR  COUT
         INX
         BNE  ONLINEE
ONLINEF  LDA  #"1"
         BIT  LNG        ;teste si drive 1 ou 2
         BPL  ONLINE1
         LDA  #"2"
ONLINE1  JSR  COUT
         LDA  #" "
         JSR  COUT
         LDA  LNG
         AND  #$F
         STA  LNG
         LDA  #"/"
         JSR  COUT
AFF      INY             ;affiche nom volume
         LDA  BUFONLIN,Y
         ORA  #%10000000
         JSR  COUT
         DEC  LNG
         BNE  AFF
         LDA  #$8D
         JSR  COUT
ERR      DEC  CNT
         BPL  ONLINED
         JSR  GETRET
VADEB    JSR  HOME
         JSR  LITPREF
         JMP  CHOIX

PARLSTN  DFB  2
         DFB  0
         DA   BUFONLIN
BUFONLIN DS   256
CNT      DS   1
LNG      DS   1
MESONL   DFB  $8D
         ASC  "Volumes en ligne :"
         DFB  $8D
         ASC  "------------------"
         DFB  $8D,0
MESON0   ASC  "Port "
         DFB  0
MESON1   ASC  " Lect "
         DFB  0

*----------------
* GET FILE INFO
*----------------

GFI      JSR  MLI        ; récupères les infos
         DFB  GETFINFO   ; sur un fichier
         DA   PARLSTG
         BCC  OKG
         JSR  ERREUR
OKG      RTS
PARLSTG  DFB  $A
         DA   PATHNAME
ACCBITG  DS   1          ; lect/écrit/renam...
FILETYG  DS   1          ; type de fichier
AUXTYPG  DS   2          ; param auxiliaire (lng par ex)
STORAGG  DS   1          ; type de stockage
BLOCKUS  DS   2          ; nb de blocs affectés
HEURDAT  DS   8

*----------------
* CHANGE PREFIXE
*----------------

MODPREF  JSR  HOME
         JSR  LITPREF
MODPR    LDA  #7
         STA  CV
         JSR  VTAB
         LDA  #0
         STA  CH
         LDY  #>MESMOD
         LDA  #<MESMOD
         JSR  STROUT
         LDA  #38
         STA  LONG
         LDA  #9
         JSR  GBASCALC
         INC  GBASL
         JSR  INPUT
         BCS  OKCH       ; esc ?
         LDY  TPS
         BNE  MOD0
         JSR  BIP
         BEQ  MODPR
MOD0     STY  PATHNAME   ; stocke long
         INC  PATHNAME
         BEQ  MODMLI
         DEC  GBASL
MOD2     LDA  (GBASL),Y
         INY
         STA  PATHNAME,Y
         DEY
         DEY
         BPL  MOD2
```

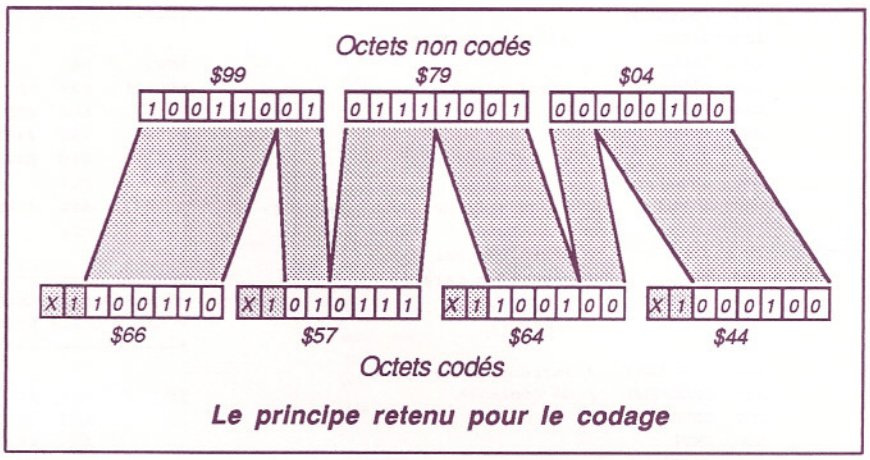
```
2F70- 00 00 20 58 FC 20 EA 2F
2F78- A9 07 85 25 20 22 FC A9
2F80- 00 85 24 A0 2F A9 D5 20
2F88- 3A DB A9 26 8D B7 2C A9
2F90- 09 20 47 F8 E6 26 20 C7
2F98- 2B B0 34 AC B8 2C D0 05
2FA0- 20 82 32 F0 D3 8C 6E 35
2FA8- EE 6E 35 F0 0C C6 26 B1
2FB0- 26 C8 99 6E 35 88 88 10
2FB8- F6 20 00 BF C6 D2 2F 90
2FC0- 0E 48 20 FB DA 68 20 D2
2FC8- 31 20 1E 32 4C 72 2F 4C
2FD0- 10 2E 01 6E 35 CE EF F5
2FD8- F6 E5 E1 F5 A0 F0 F2 FB
2FE0- E6 E9 F8 E5 A0 BA 8D 8D
2FE8- AF 00 20 00 BF C7 D2 2F
2FF0- 90 03 20 D2 31 A9 03 85
2FF8- 25 20 22 FC A2 00 BD 22
3000- 30 F0 06 20 ED FD E8 D0
3008- F5 AD 6E 35 8D AE 35 A2
3010- 01 BD 6E 35 09 80 20 ED
3018- FD E8 CE 6E 35 D0 F2 4C
3020- FB DA D0 F2 FB E6 E9 F8
3028- E5 A0 00 20 BB 30 A9 FF
3030- A2 03 9D 67 35 CA 10 FA
3038- A0 02 98 AA B1 18 9D 6B
3040- 35 CA 88 10 F7 A0 05 0E
3048- 6B 35 2E 67 35 88 10 F7
3050- A0 01 0E 6B 35 2E 68 35
3058- 88 10 F7 A0 03 0E 6C 35
3060- 2E 68 35 88 10 F7 A0 03
3068- 0E 6C 35 2E 69 35 88 10
3070- F7 A0 01 0E 6D 35 2E 69
3078- 35 88 10 F7 A0 05 0E 6D
3080- 35 2E 6A 35 88 10 F7 A0
3088- 03 B9 67 35 91 1A 88 10
3090- F8 A5 18 C9 01 D0 07 A5
3098- 19 C9 16 D0 01 60 20 A4
30A0- 30 4C 2E 30 18 A5 18 69
30A8- 03 85 18 90 02 E6 19 18
30B0- A5 1A 69 04 85 1A 90 02
30B8- E6 1B 60 A9 00 85 18 85
```

```
MODMLI   JSR  MLI
         DFB  STPREFIX
         DA   PARLSTCP
         BCC  OKCH
         PHA
         JSR  CROUT
         PLA
         JSR  ERREUR
         JSR  GETRET
         JMP  MODPREF
OKCH     JMP  VADEB
PARLSTCP DFB  1
         DA   PATHNAME
MESMOD   ASC  "Nouveau préfixe :"
         DFB  $8D,$8D
         ASC  "/"
         DFB  0

*----------------
* LIT PREFIXE
*----------------

LITPREF  JSR  MLI        ; le lit
         DFB  GTPREFIX
         DA   PARLSTCP
         BCC  LIT
         JSR  ERREUR

LIT      LDA  #3
         STA  CV
         JSR  VTAB


         LDX  #0
LIT0     LDA  MESSPR,X
         BEQ  LIT00
         JSR  COUT
         INX
         BNE  LIT0

LIT00    LDA  PATHNAME
         STA  LNGPATHN
         LDX  #1         ; l'affiche
LIT1     LDA  PATHNAME,X
         ORA  #%10000000
         JSR  COUT
         INX
         DEC  PATHNAME
         BNE  LIT1
         JMP  CROUT
MESSPR   ASC  "Préfixe "
         DFB  0

*----------------
* CODER UN BLOC
*----------------

CODEBLOC JSR  INITCOD

DEBCOD   LDA  #$FF       ;OCTETS DESTINATION A 11xxxxxx
         LDX  #3
```

```
CODE0    STA  D1,X
         DEX
         BPL  CODE0

         LDY  #2         ;PREND 3 OCTETS A TRAITER
         TYA
         TAX
CODE1    LDA  (COMPTNC),Y
         STA  S1,X
         DEX
         DEY
         BPL  CODE1

         LDY  #5         ;LES TRANSFORME EN 4 OCTETS
CODE3    ASL  S1         ;DE D1 A D4
         ROL  D1
         DEY
         BPL  CODE3

         LDY  #1
CODE4    ASL  S1
         ROL  D2
         DEY
         BPL  CODE4

         LDY  #3
CODE5    ASL  S2
         ROL  D2
         DEY
         BPL  CODE5

         LDY  #3
CODE6    ASL  S2
         ROL  D3
         DEY
         BPL  CODE6

         LDY  #1
CODE7    ASL  S3
         ROL  D3
         DEY
         BPL  CODE7

         LDY  #5
CODE8    ASL  S3
         ROL  D4
         DEY
         BPL  CODE8

         LDY  #3
CODE81   LDA  D1,Y
         STA  (COMPTC),Y
         DEY
         BPL  CODE81

         LDA  COMPTNC    ;ON A FINI LES 512 ?
         CMP  #1
         BNE  CODE9
         LDA  COMPTNC+1
         CMP  #$16
```

```
30C0- 1A A9 14 85 19 A9 17 85
30C8- 1B 60 20 BB 30 A0 03 A2
30D0- 03 B1 1A 9D 67 35 CA 88
30D8- 10 F7 A0 05 4E 6A 35 6E
30E0- 6D 35 88 10 F7 A0 01 4E
30E8- 69 35 6E 6D 35 88 10 F7
30F0- A0 03 4E 69 35 6E 6C 35
30F8- 88 10 F7 A0 03 4E 68 35
3100- 6E 6C 35 88 10 F7 A0 01
3108- 4E 68 35 6E 6B 35 88 10
3110- F7 A0 05 4E 67 35 6E 6B
3118- 35 88 10 F7 A0 02 B9 6B
3120- 35 91 18 88 10 F8 A5 18
3128- C9 01 D0 07 A5 19 C9 16
3130- D0 01 60 20 A4 30 4C CD
3138- 30 20 BB 30 A9 00 8D 87
3140- 31 A9 01 20 9A 31 AD FA
3148- 25 09 C0 20 9A 31 49 3F
3150- 20 9A 31 20 9A 31 EE F7
3158- 07 A0 00 B1 1A 20 9A 31
3160- 18 6D 87 31 8D 87 31 E6
3168- 1A D0 02 E6 1B A5 1A C9
3170- AC D0 E3 A5 1B C9 19 D0
3178- DD AD 87 31 09 C0 20 9A
3180- 31 A9 A0 8D F7 07 60 00
3188- A0 0A A9 C6 20 A8 FC AD
3190- 00 C0 C9 9B F0 03 88 D0
3198- F1 60 48 A2 00 A1 06 29
31A0- 10 F0 F8 68 81 08 60 A2
31A8- 00 A1 06 29 08 D0 02 18
31B0- 60 A1 08 29 7F 38 60 A9
31B8- 00 8D D0 31 8D D1 31 20
31C0- A7 31 90 01 60 EE D0 31
31C8- D0 F5 EE D1 31 D0 F0 60
31D0- 00 00 8D 1D 32 20 FB DA
31D8- 20 FB DA A9 00 85 24 A9
31E0- 14 85 25 20 22 FC A9 21
31E8- 20 ED FD A9 A0 20 ED FD
31F0- 20 82 32 A0 FF C8 B9 E2
31F8- 33 CD 1D 32 F0 04 C9 00
3200- D0 F3 C8 B9 E2 33 10 FA
3208- 20 ED FD C8 B9 E2 33 30
```

```
        BNE  CODE9
        RTS              ;OUI

CODE9   JSR  CODEINC     ;NON : INCREM
        JMP  DEBCOD

CODEINC CLC
        LDA  COMPTNC
        ADC  #3
        STA  COMPTNC
        BCC  CODE10
        INC  COMPTNC+1

CODE10  CLC
        LDA  COMPTC
        ADC  #4
        STA  COMPTC
        BCC  CODE11
        INC  COMPTC+1
CODE11  RTS

INITCOD LDA  #0
        STA  COMPTNC
        STA  COMPTC
        LDA  #>BUFNC
        STA  COMPTNC+1
        LDA  #>BUFC
        STA  COMPTC+1
        RTS

*---------------
* DECODER UN BLOC
*---------------

DEBUDEC JSR  INITCOD

DEBDECOD LDY #3
        LDX  #3
DECODE1 LDA  (COMPTC),Y
        STA  D1,X
        DEX
        DEY
        BPL  DECODE1

        LDY  #5
DECODE2 LSR  D4
        ROR  S3
        DEY
        BPL  DECODE2

        LDY  #1
DECODE3 LSR  D3
        ROR  S3
        DEY
        BPL  DECODE3

        LDY  #3
DECODE4 LSR  D3
        ROR  S2
        DEY
        BPL  DECODE4
```



Le principe retenu pour le codage

```
        LDY  #3
DECODE5 LSR  D2
        ROR  S2
        DEY
        BPL  DECODE5

        LDY  #1
DECODE6 LSR  D2
        ROR  S1
        DEY
        BPL  DECODE6

        LDY  #5
DECODE7 LSR  D1
        ROR  S1
        DEY
        BPL  DECODE7

        LDY  #2
DECODE8 LDA  S1,Y
        STA  (COMPTNC),Y
        DEY
        BPL  DECODE8

        LDA  COMPTNC
        CMP  #1
        BNE  DECODE9
        LDA  COMPTNC+1
        CMP  #$16
        BNE  DECODE9
        RTS

DECODE9 JSR  CODEINC
        JMP  DEBDECOD

*---------------
* ENVOI UN BLOC
*---------------

ENVBLOC JSR  INITCOD
        LDA  #0
```

```
3210- F7 A9 A0 20 ED FD A9 21
3218- 20 ED FD 38 60 00 20 FB
3220- DA A9 15 85 25 20 22 FC
3228- A9 00 85 24 A0 32 A9 5E
3230- 20 3A DB 2C 00 C0 10 FB
3238- 2C 10 C0 AD 00 C0 C9 0D
3240- F0 09 C9 1B F0 0A 20 82
3248- 32 B0 E8 20 55 32 18 60
3250- 20 55 32 38 60 A9 14 20
3258- 22 FC 20 42 FC 60 A0 A0
3260- A0 A0 AA AA AA A0 A0 C1
3268- F0 F0 F5 F9 E5 FA A0 F3
3270- F5 F2 A0 A0 BC D2 C5 D4
3278- D5 D2 CE BE A0 A0 AA AA
3280- AA 00 98 48 8A 48 A0 40
3288- 98 6A AA CA D0 FD 2C 30
3290- C0 88 D0 F4 38 68 AA 68
3298- A8 60 20 BB 30 A0 00 B1
32A0- 18 A2 00 DD 25 35 F0 13
32A8- E8 E8 E0 42 D0 F5 E6 18
32B0- D0 ED E6 19 A5 19 C9 16
32B8- D0 E5 60 E8 BD 25 35 91
32C0- 18 D0 EB A9 15 20 ED FD
32C8- 20 39 FB 20 58 FC 20 71
32D0- 2D A0 00 B9 E7 32 F0 09
32D8- 2C 30 C0 20 ED FD C8 D0
32E0- F2 20 1E 32 4C 00 20 8D
32E8- A0 A0 A0 A0 A0 DF DF DF
32F0- DF DF DF DF DF DF DF DF
32F8- DF DF DF DF DF DF DF DF
3300- DF DF DF DF DF DF DF DF
3308- DF DF 8D A0 A0 A0 A0 A0
3310- 20 09 0E 14 05 12 10 0F
3318- 0D 13 20 20 0D 09 0E 09
3320- 14 05 0C 20 20 20 20 16
3328- 20 31 2E 30 20 8D 8D 8D
3330- A0 A0 A0 A0 A0 A0 A0 A0
3338- A0 A8 E3 A9 A0 D0 EF ED
3340- A7 F3 A0 A6 A0 C3 D0 A0
3348- AD A0 B1 B9 B8 B7 8D 8D
3350- 8D 8D A0 A0 A0 A0 A0 D3
3358- EF F5 F2 E3 E5 A0 D0 F2
```

```
        STA  CHECKSUM
        LDA  #SOH        ; envoi code ASCII 1
        JSR  ENVOI
        LDA  NUMBLOC     ; envoi n° bloc
        ORA  #%11000000
        JSR  ENVOI
        EOR  #$3F        ; et en complément
        JSR  ENVOI
        JSR  ENVOI       ; utile pour Mac d'aligner pair

ENV0    INC  $7F7        ; flash sur bas écran
        LDY  #0          ; charge un caract
        LDA  (COMPTC),Y
        JSR  ENVOI       ; l'envoie
        CLC
        ADC  CHECKSUM    ; incrémente la somme
        STA  CHECKSUM    ; de controle
        INC  COMPTC
        BNE  ENV1
        INC  COMPTC+1
ENV1    LDA  COMPTC
        CMP  #$AC        ; en envoie 684 en tout
        BNE  ENV0
        LDA  COMPTC+1
        CMP  #$19
        BNE  ENV0
        LDA  CHECKSUM    ; envoie le checksum
        ORA  #%11000000
        JSR  ENVOI
        LDA  #$A0        ; efface bas écran
        STA  $7F7
        RTS

CHECKSUM DS   1

*---------------
* ATTENDRE 1 SEC
*---------------

WAIT    LDY  #10
W1      LDA  #$C6
        JSR  $FCA8
        LDA  KBD
        CMP  #$9B
        BEQ  W2
        DEY
        BNE  W1
W2      RTS

*---------------
* ENVOYER CARACTERE DE L'ACC
*---------------

ENVOI   PHA
ENVOI1  LDX  #0
        LDA  (STATUS,X)
        AND  #%00010000 ; attendre libération de
        BEQ  ENVOI1     ; la carte avant envoi
        PLA
        STA  (DATA,X)
        RTS

*---------------
* RECEVOIR UN CARAC DANS L'ACC
* CARRY CLEAR SI PAS DE CARACT
*---------------

RECOIT  LDX  #0
        LDA  (STATUS,X)
        AND  #%00001000
        BNE  OK
        CLC              ; on revient bredouille
        RTS
OK      LDA  (DATA,X)    ; on a eu un caract
        AND  #%01111111
        SEC
        RTS

*---------------
* RECEVOIR 1 CARAC
*     SEC SI RECU
*     CLC SI PAS RECU AVANT X TPS
*---------------

RECOIT1 LDA  #0
        STA  STREC
        STA  STREC+1
RECOIT11 JSR RECOIT
        BCC  RECOIT2
        RTS
RECOIT2 INC  STREC
        BNE  RECOIT11
        INC  STREC+1
        BNE  RECOIT11
        RTS

STREC   DS   2

*---------------
* ERREUR
*---------------
```

```
3360- EF C3 CF C4 C5 AC A0 ED
3368- EF E4 E5 A0 E4 A7 E5 ED
3370- F0 EC EF E9 8D 8D A0 A0
3378- A0 A0 A0 A0 A0 A0 A0 A0
3380- A0 E4 E1 EE F3 A0 A0 D0
3388- EF ED A7 F3 A0 EE DB A0
3390- B2 B8 A0 A0 A0 A0 A0 A0
3398- 8D 8D 8D A0 A0 A0 A0 A0
33A0- D2 E5 F6 F5 E5 A0 D0 EF
33A8- ED A7 F3 A0 BA A0 A8 B1
33B0- A9 A0 B3 B9 A0 B5 B1 A0
33B8- B2 B4 A0 B4 B3 8D 8D A0
33C0- A0 A0 A0 A0 DF DF DF DF
33C8- DF DF DF DF DF DF DF DF
33D0- DF DF DF DF DF DF DF DF
33D8- DF DF DF DF DF DF DF DF
33E0- DF 00 03 08 27 28 2F 5A
33E8- 4E C5 F2 F2 E5 F5 F2 A0
33F0- E4 A7 E1 E3 E3 FD F3 A0
33F8- E4 E9 F3 F1 F5 E5 06 07
3400- 45 46 C9 EE F4 F2 EF F5
3408- F6 E1 E2 EC E5 04 2B C4
3410- E9 F3 F1 F5 E5 A0 F0 F2
3418- EF F4 FB E7 FB 40 44 CE
3420- EF ED A0 E9 EE E3 EF F2
3428- F2 E5 E3 F4 11 48 49 C4
3430- E9 F3 F1 F5 E5 AF E4 E9
3438- F2 E5 E3 F4 EF F2 F9 A0
3440- F3 E1 F4 F5 F2 FB 4E D0
3448- F2 EF F4 FB E7 FB A0 E5
3450- EE A0 EC E5 E3 F4 F5 F2
3458- E5 57 C4 E5 F5 F8 A0 F6
3460- EF EC F5 ED E5 F3 A0 E8
3468- EF ED EF EE F9 ED E5 F3
3470- 7D D6 EF EC F5 ED E5 A0
3478- F3 E1 F4 F5 F2 FB 7E CC
3480- E5 A0 F4 F9 F0 E5 A0 A7
3488- C3 E1 F4 E1 EC EF E7 F5
3490- E5 A7 A0 E5 F3 F4 A0 E9
3498- EE F4 E5 F2 E4 E9 F4 7F
34A0- D0 F2 EF C4 CF D3 A0 E5
34A8- F3 F4 A0 E9 EE E4 E9 F3
34B0- F0 E5 EE F3 E1 E2 EC E5
34B8- 00 C5 F2 F2 E5 F5 F2 A0
34C0- BF BF BF 00 A9 38 8D 7C
34C8- 04 A9 6B 8D 7D 04 A9 81
34D0- 8D 7E 04 A9 7C 85 42 85
34D8- 3C A9 04 85 43 85 3D 85
34E0- 3F A9 7E 85 3E 38 4C 11
34E8- C3 A0 02 B9 F5 34 20 9A
34F0- 31 88 10 F7 60 68 39 1B
34F8- A0 02 B9 04 35 20 9A 31
3500- 88 10 F7 60 6F 39 1B A0
3508- 02 B9 13 35 20 9A 31 88
3510- 10 F7 60 67 39 1B 20 5E
3518- 24 4C 66 24 20 71 2D 20
3520- 07 35 4C 1E 32 00 A0 88
3528- C0 CB C1 A7 D3 A5 AD 8D
3530- DC 8E FB 8F FD 9D FC A1
3538- DB A3 A3 A4 DD 90 E5 91
3540- E5 94 E9 95 E9 99 EF 9A
3548- EF 9E F5 9F F5 CE AA CF
3550- AA C7 A2 C8 A2 D2 A2 D3
3558- A2 C9 AE D4 A7 D5 A7 D0
3560- AD D1 AD D6 AF CA A0
```

```
ERREUR   STA  CODERR
         JSR  CROUT
         JSR  CROUT
         LDA  #0
         STA  CH
         LDA  #20
         STA  CV
         JSR  VTAB
         LDA  #$21
         JSR  COUT
         LDA  #" "
         JSR  COUT
         JSR  BIP
         LDY  #$FF
ERREUR1  INY
         LDA  ERREURS,Y
         CMP  CODERR
         BEQ  ERREUR0
         CMP  #0
         BNE  ERREUR1

ERREUR0  INY
         LDA  ERREURS,Y
         BPL  ERREUR0
ERREUR2  JSR  COUT
         INY
         LDA  ERREURS,Y
         BMI  ERREUR2
         LDA  #" "
         JSR  COUT
         LDA  #$21
         JSR  COUT
         SEC
         RTS

CODERR   DS   1

*----------------
* ATTEND RETURN (OU ESC)
*----------------

GETRET   JSR  CROUT
         LDA  #21
         STA  CV
         JSR  VTAB
         LDA  #0
         STA  CH
         LDY  #>MESRET
         LDA  #<MESRET
         JSR  STROUT

RET1     BIT  KBD
         BPL  RET1

         BIT  STROBE
         LDA  KBD
         CMP  #$D
         BEQ  RET3
         CMP  #$1B
         BEQ  RET4
         JSR  BIP
         BCS  RET1

RET3     JSR  RET5
         CLC
         RTS

RET4     JSR  RET5
         SEC
         RTS

RET5     LDA  #20
         JSR  VTAB
         JSR  CLREOP
         RTS
MESRET   ASC  "    ***  Appuyez sur  <RETURN>  ***"
         DFB  0

*----------------
* BIP
*----------------

BIP      TYA
         PHA
         TXA
         PHA
         LDY  #$40
BIP1     TYA
         ROR
         TAX
BIP2     DEX
         BNE  BIP2
         BIT  HP
         DEY
         BNE  BIP1
         SEC
         PLA
         TAX
         PLA
         TAY
         RTS

*----------------
* DECODE ACCENTS MAC
*----------------

DECODMAC JSR  INITCOD
         LDY  #0
DEC00    LDA  (COMPTNC),Y
         LDX  #0
DEC1     CMP  ACCENT,X
         BEQ  DEC2
         INX
         INX
         CPX  #66
         BNE  DEC1
DEC3     INC  COMPTNC
         BNE  DEC00
         INC  COMPTNC+1
         LDA  COMPTNC+1
         CMP  #$16
         BNE  DEC00
         RTS

DEC2     INX
         LDA  ACCENT,X
         STA  (COMPTNC),Y
         BNE  DEC3

*----------------
* RESET
*----------------

RESET    LDA  #21        ; utile si Apple IIgs et
         JSR  COUT       ; 80 colonnes réactivée.
         JSR  TEXT       ; une page de pub...
         JSR  HOME
         JSR  CLOSEFIL

         LDY  #0
RESET1   LDA  MESRES1,Y
         BEQ  RESET2
         BIT  HP
RESET3   JSR  COUT
         INY
         BNE  RESET1

RESET2   JSR  GETRET
         JMP  DEB00

MESRES1  DFB  $8D
         ASC  "                                  "
         DFB  $8D
         ASC  "     "
         INV  " INTERPOMS  MINITEL    V 1.0 "
         DFB  $8D,$8D,$8D
         ASC  "          (c) Pom's & CP - 1987"
         DFB  $8D,$8D,$8D,$8D
         ASC  "     Source ProCODE, mode d'emploi"
         DFB  $8D,$8D
```

```
        ASC  "            dans  Pom's n° 28       "
        DFB  $8D,$8D,$8D
        ASC  "     Revue Pom's : (1) 39 51 24 43"
        DFB  $8D,$8D
        ASC  "                                    "
        DFB  0

*----------------
* MESSAGES D'ERREUR
*----------------

ERREURS DFB  3,8,$27,$28,$2F,$5A,$4E
        ASC  "Erreur d'accès disque"
        DFB  6,7,$45,$46
        ASC  "Introuvable"
        DFB  4,$2B
        ASC  "Disque protégé"
        DFB  $40,$44
        ASC  "Nom incorrect"
        DFB  17,$48,$49
        ASC  "Disque/directory saturé"
        DFB  $4E
        ASC  "Protégé en lecture"
        DFB  $57
        ASC  "Deux volumes homonymes"
        DFB  $7D
        ASC  "Volume saturé"
        DFB  $7E
        ASC  "Le type 'Catalogue' est interdit"
        DFB  $7F
        ASC  "ProDOS est indispensable"
        DFB  0
        ASC  "Erreur ???"
        DFB  0

*----------------
* INTERFACE //c
*----------------

DEUXC   LDA  #$38
        STA  $47C
        LDA  #$6B
        STA  $47D
        LDA  #$81
        STA  $47E
        LDA  #$7C
        STA  $42
        STA  $3C
        LDA  #4
        STA  $43
        STA  $3D
        STA  $3F
        LDA  #$7E
        STA  $3E
        SEC
        JMP  $C311      ; auxmove

*----------------
* DIVERS
*----------------

CONNECT LDY  #2
CONNECT1 LDA SCONNECT,Y
        JSR  ENVOI
        DEY
        BPL  CONNECT1
        RTS
SCONNECT DFB 104,57,27

RETOURN LDY  #2
RETOURN1 LDA SRETOURN,Y
        JSR  ENVOI
        DEY
        BPL  RETOURN1
        RTS
SRETOURN DFB 111,57,27
DECONNE LDY  #2
DECONNE1 LDA SDECONNE,Y
        JSR  ENVOI
        DEY
        BPL  DECONNE1
        RTS
SDECONNE DFB 103,57,27

MOUTKCR JSR  MOUTCR
        JMP  MOUTK

CDG     JSR  CLOSEFIL
        JSR  DECONNE
        JMP  GETRET

*----------------
* CORRESPONDANCE MAC/APPLE //
* QUELQUES CODES SEULEMENT
*----------------

ACCENT  DFB  0," "      ; pas de 0 dans fich TXT
        DFB  136,"à"
        DFB  203,"A"    ; Â
        DFB  167,"S"    ; ß
        DFB  165,"-"    ; •
        DFB  141,"ç"
        DFB  142,"é"
        DFB  143,"è"
        DFB  157,"ù"
        DFB  161,"`"
        DFB  163,"#"
        DFB  164,"$"
        DFB  144,"e"    ; ê
        DFB  145,"e"    ; ë
        DFB  148,"i"    ; î
        DFB  149,"i"    ; ï
        DFB  153,"o"    ; ô
        DFB  154,"o"    ; ö
        DFB  158,"u"    ; û
        DFB  159,"u"    ; ü
        DFB  206,"*"    ; Œ
        DFB  207,"*"    ; œ
        DFB  199,162    ; »
        DFB  200,162    ; «
        DFB  210,162    ; "
        DFB  211,162    ; "
        DFB  201,"."    ; …
        DFB  212,167    ; '
        DFB  213,167    ; '
        DFB  208,"-"    ; –
        DFB  209,"-"    ; —
        DFB  214,"/"    ; ÷
        DFB  202," "    ; espace collant

*----------------
* STOCK
*----------------

D1      DS   1
D2      DS   1
D3      DS   1
D4      DS   1
S1      DS   1
S2      DS   1
S3      DS   1
PATHNAME DS  64
LNGPATHN DS  1
```

C'est bien Marcel Cottini qui est l'auteur de Call-Rwts, programme paru dans le numéro 26 de Pom's et non P. Neveu, comme le laissait supposer la signature. Cette mauvaise attribution de paternité est imputable à un montage 'rapide'.

Rappelons que Marcel Cottini est auteur de plusieurs ouvrages concernant l'Apple // (éd. Sybex, P.S.I.) et qu'il avait déjà signé des pages de Pom's.

CopyBasFiles, fichiers & '/RAM'

François Dreyfuss

Vous savez que ProDOS crée automatiquement un disque virtuel dans l'espace mémoire de la carte 80 colonnes étendue, disque baptisé /RAM. Il est donc tentant d'y recopier, lors du 'boot', la majorité des programmes d'une disquette ; par la suite, en les recherchant par priorité sur le disque virtuel, les temps de chargement seront extrêmement courts.

Le programme COPYBASFILES crée un fichier EXEC qui recopie les programmes Applesoft d'une disquette vers le volume virtuel. Il ne fonctionne, bien sûr, que sur //e ou //c, et se lance par un simple RUN. Il demande, pour chaque programme rencontré, si on désire qu'il soit recopié ou non. Le fichier EXEC créé s'appelle WORKFILE. Ainsi une simple instruction :

```
PRINT CHR$(4)"EXEC
  WORKFILE"
```

à la fin de votre programme de STARTUP alimentera votre disque virtuel.

Techniquement, il ne faut cependant pas oublier que le volume /RAM comprend moins de blocs disponibles qu'une disquette 140Ko... il sera donc parfois nécessaire d'effectuer une sélection impitoyable des fichiers qui devront être copiés. La ligne 164 de COPYBASFILES contient l'instruction de test sur le type de fichier ; modifiez-la en conséquence si vous ne désirez ne recopier que les fichier binaires, par exemple.

Programme 'COPYBASFILES'

```
0  TEXT : HOME
1  PRINT "CREATION FICHIER EXEC POUR COPIE
    DES"
2  PRINT "FICHIERS BASIC VERS DISQUE VIRTU
   EL"
3  PRINT : PRINT
5  DIM L$(40)
10 D$ =  CHR$ (4)
11  PRINT D$"PREFIX"
12  INPUT XX$:S$ =  MID$ (XX$,2,( LEN (XX$
   ) - 2))
13  PRINT "PREFIXE ORIGINE = "XX$
15 MM$ = "/" + S$
16  PRINT "PREFIXE ARRIVEE (DEFAUT=/RAM/):
   "
17  INPUT R$: IF R$ = "" THEN R$ = "/RAM"
18 MM$ = MM$ + "/": IF  RIGHT$ (R$,1) <  >
    "/" THEN R$ = R$ + "/"
19  PRINT : PRINT "REPONDRE <N> SI FICHIER
    INUTILE"
80  PRINT : PRINT "LECTURE DIRECTORY "S$
85 U$ = "/" + S$
86 S$ = U$
90  PRINT D$;"OPEN ";S$;" ,T15"
100  PRINT D$;"READ ";S$
120 I = 0
130  ONERR  GOTO 400
150  INPUT L$(I)
160 L$(I) =  MID$ (L$(I),2,19)
162 T$ =  RIGHT$ (L$(I),3)
164  IF T$ <  > "BAS" THEN 168
165 L$(I) =  LEFT$ (L$(I),16)
167 I = I + 1
168  REM
180  GOTO 150
400  PRINT D$;"CLOSE ";S$
405  ONERR  GOTO 850
410 I = I - 1
420  REM  I+1 BASIC FILES
510  IF I < 0 THEN  END
512  GOTO 900
515  REM  BUILD AN EXEC FILE USING LOAD
520 E$ = "WORKFILE"
530  IF I >  = 0 THEN 532
531  PRINT "PAS DE FICHIER": END
532  PRINT D$;"OPEN ";E$: PRINT D$;"CLOSE
   ";E$
533  PRINT D$;"DELETE ";E$
540  PRINT D$;"OPEN ";E$
550  PRINT D$;"APPEND ";E$
560 J = I
565  GOTO 590
570  PRINT "PREFIX,D1"
580  PRINT "PREFIX ";S$
590  FOR I = 0 TO J
600  PRINT "LOAD ";MM$;L$(I)
700  PRINT "SAVE "R$;L$(I)
710  NEXT I
725  PRINT "NEW"
730  PRINT "CAT"
750  PRINT D$;"CLOSE ";E$
800  PRINT "FICHIER ";E$;" PRET"
810  END
850  PRINT "ERREUR #" PEEK (222)
855  POKE 216,0: RESUME
860  END
900 J =  - 1
910  FOR K = 0 TO I
920  PRINT "RECOPIER ";L$(K);
930  GET Q$: IF Q$ <  > "N" THEN Q$ = "O"
935  PRINT Q$
940  IF Q$ = "N" THEN 980
950 J = J + 1
960 L$(J) = L$(K)
980  NEXT K
985 I = J
990  GOTO 515
```

Messages et Basic.System

Jacques Rey

Le DOS 3.3 nous avait habitué à beaucoup de fantaisies de structure et de nombreux 'patches' le rendait souvent incompatible avec lui-même. Avec ProDOS et BASIC. SYSTEM ces petits jeux sont finis, place à la rigueur.
Il reste tout de même un petit point qui mérite d'être modifié sans toucher à la cohérence du système : nous pouvons franciser les messages de BASIC. SYSTEM.

Or, dans ce domaine aussi, les choses ont bien changé. Dans DOS 3.3 il était facile de visualiser l'ensemble des messages avec un éditeur de secteurs (Disk Fixer, Mobby-disk de Pom's) et de les modifier directement pour peu que l'on respecte la longueur des phrases d'origine.

Un tel examen de Basic.System ne révèle qu'un ou deux messages ; où sont donc passés les *Syntax Error* et autres *Path Not Found* ?

Les concepteurs du système, soucieux d'économiser la place, ont adopté une astuce qui peut rendre de grands services au programmeur qui doit afficher beaucoup de texte : les mots sont codés à raison de deux caractères par octet, chaque demi-octet servant de pointeur de position dans une table alphabétique.

Un exemple

Soit la table alphabétique suivante :

Désassemblage PRINTERR.BAS.S
(Assembleur ProCODE)

```
*********************************
* Désassemblage de la routine   *
*     d'affichage des messages  *
*     d'erreur de Basic.system  *
*       J. Rey  Septembre 1986  *
*********************************
*
        org $9F88
*
cout     =    $FDED     ;routine de sortie de caractères du moniteur
comptcar =    $BE4B     ;drapeau de comptabilisation de caractères
dercar   =    $BCB6     ;drapeau indiquant la position du dernier
                        ;caractère dans "bufimpr".
bufimpr  =    $0201     ;les caractères du message y sont stockés avant
                        ;l'appel de "cout".
debmess  =    $BA13     ;table indiquant le début de chaque message
caract   =    $BA28     ;table de 30 caractères et signes constituant les mots
messcod  =    $BA48     ;table des messages codés

*********** ROUTINE PRINTERROR ***********

printerr TAY            ;A contient le code de l'erreur ($02 à $15)
         LDX debmess,Y  ;pointe sur le début du message correspondant
         JSR debut      ;saut à la routine récupérant les caractères

**** FIN : impression et retour

fin      LDX  comptcar  ;combien de caractères compte ce message ?
         STX  dercar    ;je range ce nombre pour plus tard
         JSR  return    ;un CR à vide pour détacher le message
dong     LDA  #$87      ;caractère BELL pour faire un peu de bruit !
         JSR  affiche
printcar LDX  #$00      ;on se place au début du buffer
]boucle  LDA  bufimpr,X ;et un caractère svp !
         JSR  affiche
         INX            ;caractère suivant
         CPX  dercar    ;n'est ce point le dernier ?
         BCC  ]boucle   ;pas encore
return   LDA  #$8D      ;pour finir il faut un CR
affiche  JMP  cout      ;c'est par là que le tout se retrouve à l'écran

**** DECODE

debut    LDY  #$00
         CLC            ;initialisation C = 0
         STY  comptcar  ;remise à zéro du compteur de caractères
bouclel  JSR  decode    ;saut à la routine qui lit la table des messages
         BNE  litcar    ;si le 1/2 octet n'est pas nul c'est un des 15
                        ;premiers caractères de la table.
         JSR  decode    ;si non on récupère le 1/2 octet suivant
         ORA  #$10      ;auquel on ajoute $10 pour lire l'un des 15
                        ;caractères suivants de la table.
litcar   TAY            ;le résultat de ci-dessus sert de pointeur de
                        ;table.
         LDA  caract,Y  ;et voila un caractère !
         BEQ  sortie    ;si NUL c'est la fin du message
range    LDY  comptcar
         STA  bufimpr,Y ;on range le caractère trouvé dans le tampon
```

$1000	:	00	C1	C2	C5	C9	CA	CC	CD	CE	CF	D0	D2	D3	D9	A0	00
caractère	:		A	B	E	I	J	L	M	N	O	P	R	S	U	esp	
position	:	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Et le message codé ci-dessous :

```
$1100:29 85 9D BE A9 7E CF
```

Chaque demi-octet donne la position d'un caractère ce qui nous donne :

```
BONJOUR POM S
```

Nous avons utilisé sept octets pour treize caractères, soit presque moitié moins.
En fait, l'économie n'est pas toujours aussi importante pour deux raisons.

D'une part, il faut terminer le message par un demi-octet pointant sur un caractère nul pour permettre à la routine de décodage de comprendre que la phrase est terminée (le F du dernier octet de l'exemple pointe sur la valeur 00). Dans le cas d'un message comportant un nombre pair de caractères et espaces il faudra utiliser un octet supplémentaire.

D'autre part un demi-octet ne peut adresser que quinze caractères (la valeur zéro est exclue, nous allons voir pourquoi) ce qui est peu pour constituer des phrases complexes. Ajoutons donc seize autres caractères à la table, auxquels nous accèderons de la façon suivante : dès que la routine de décodage rencontre un demi-octet égal à zéro, elle récupère le demi-octet suivant et lui ajoute la valeur $10. La valeur du pointeur s'étend alors de $01 à $16 soit 30 caractères ou signes (la valeur $10 est exclue) ce qui suffit à la majeure partie des cas.

```
        BNE   bouclel     ;branche toujours
sortie  RTS               ;retour à l'envoyeur pour affichage

decode  LDA   messcod,X   ;récupération d'un octet représentant 2 caractères
        BCS   mibas       ;C = 1 c'est le tour du 2 ème caractère
        BEQ   tabul       ;l'octet est nul le suivant est une tabulation
        LSR
        LSR
        LSR
        LSR               ;pour ne conserver que la partie haute de
                          ;l'octet.
        SEC               ;positionne C pour la récupération de la partie
                          ;basse.
        RTS

tabul   INX               ;lecture de l'octet suivant
        LDA   messcod,X   ;cet octet est considéré comme une tabulation
        STA   comptcar    ;il est placé dans "comptcar"
        INX               ;pour lire les caractères suivants
        BNE   decode      ;branche toujours

mibas   INX               ;prépare X pour la lecture du couple de caractères
        AND   #$0F        ;pour ne conserver que la partie basse de l'octet
        CLC               ;pour la lecture de l'octet suivant
        RTS
```

Dans Basic.System, nous trouvons en $BA28 la table de caractères et signes, en $BA4/$BB46 la table des messages codés et en $BA13 une table qui donne le début du message en fonction du code de celui-ci.
Ces tables sont exploitées par une routine appelée 'PRINTERR' située à partir de l'adresse $9F88, que vous trouverez ici désassemblée et commentée (assembleur ProCODE). Il suffit de placer le code de l'erreur dans l'accumulateur et d'effectuer un JSR $9F88 pour afficher le message correspondant, les codes valides vont de $02 à $15, de plus le code $24 affiche le copyright Apple.

Ces quelques éléments permettront à ceux qui le désirent de modifier ou franciser les messages facilement.

Comment utiliser Pom's ?

Pour les lecteurs qui ont pour la première fois Pom's dans les mains, voici quelques éléments pour en utiliser plus rapidement le contenu.

Chaque numéro de Pom's est accompagné de manière optionnelle d'une disquette regroupant tous les programmes, disquette disponible par correspondance.

...

Si vous n'avez pas la disquette, saisissez les programmes Basic et sauvegardez-les sur une disquette DOS ou ProDOS selon indication dans l'article par :

```
SAVE NOM_DE_PROGRAMME
```

Pour saisir les récapitulations de codes machine (issus des sources en assembleur), passez en moniteur par :

```
CALL -151
```

puis entrez les codes ainsi :

```
2000 : A2 14 86 08 A2 60 86 09
```

et non pas comme cela :

```
2000-  A2 14 86 08 A2 60 86 09
```

Revenez alors au Basic en faisant CTRL-C. L'ordre de sauvegarde est indiqué au début de chaque récapitulation. À la place de la récapitulation, vous pouvez saisir le source en assembleur si vous disposez d'un programme assembleur (Lisa, Merlin, Procode, Toolkit...). Assemblez. Le code généré est identique à celui de la récapitulation.

Si le programme est écrit en Pascal, vous devrez le saisir avec l'Éditeur Pascal (menu principal Pascal, demandez 'E'). Compilez, puis exécutez en faisant X.

...

Si vous avez la disquette Pom's (actuellement encore formatée en DOS 3.3), plusieurs cas se présentent :

Le programme fonctionne sous DOS 3.3, faites simplement :

```
RUN PROGRAMME
```

ou :

```
BRUN PROGRAMME
```

selon qu'il s'agit de Basic ou de programme en langage machine.

S'il fonctionne sous ProDOS, convertissez-le à ce format en utilisant 'CONVERT' de la disquette Système ou en utilisant la disquette 'Utilitaires Système' dans l'option 'Copier des fichiers'. Maintenant installé sur une disquette ProDOS, le programme sera lancé par :

```
- PROGRAMME
```

Si le programme est écrit en Pascal, vous le convertirez au format Pascal à l'aide de l'utilitaire 'BASIC-PASCAL' qui est sur chaque disquette Pom's. Avant conversion, ayez une disquette formatée Pascal à disposition.

Bien entendu, si vous vous heurtez à une difficulté dans l'exploitation des programmes de Pom's, n'hésitez pas à prendre contact avec un technicien au :

```
(1) 39 51 24 43
```

Il n'écrira pas vos programmes, mais il vous dépannera.

Cette fois Gutenberg n'

Gutenberg 1395-1468 a inventé l'imprimerie.

Gutenberg et ses associés, des gens très perfectionnistes au demeurant, n'hésitant pas à regarder à la loupe le moindre détail, n'avaient pourtant fait que la moitié du travail.

Le pouvoir de l'édition c'est bien, pouvoir éditer soi-même, c'est mieux.

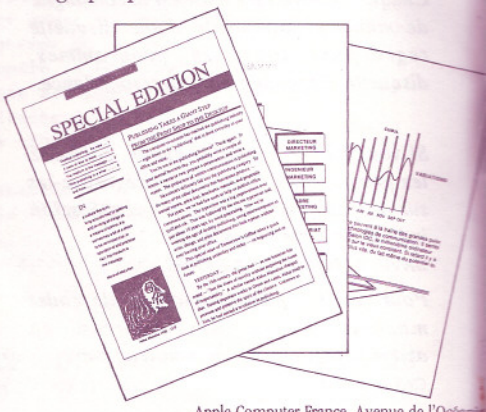
C'est d'ailleurs le seul moyen de pouvoir parvenir à la vraie liberté d'expression. Cela Gutenberg n'y avait pas songé.

En toute bonne foi, il croyait fermement dans les vertus de la vis à bois et d'une imprimerie pour tout le monde.

Chez Apple, nous croyons aux ressources de l'individu et à l'édition personnelle.

Par contre pour créer son atelier, Macintosh a fait comme Gutenberg, il s'est associé avec la "LaserWriter", l'imprimante à laser d'Apple. A la seule différence que si 31 assistants se bousculent autour d'une presse, 31 Macintosh reliés par AppleTalk se partagent aisément une LaserWriter.

Mais, malgré son million d'octets de mémoire morte (extension LaserWriter Plus) et sa résolution de 130 points/cm, la LaserWriter ne serait qu'une version légèrement améliorée de la presse en bois sans la puissance et les capacités graphiques de Macintosh Plus.



Apple Computer France. Avenue de l'Océanie

aura pas le dernier mot.

C'est-à-dire, un méga-octet de mémoire extensible à 4 méga-octets, un nouveau lecteur de disque interne double face 800K et en option un disque dur de 20 méga-octets, ce qui permet de stocker des milliers de pages de documents.

Et comme toujours, Macintosh Plus met à votre disposition tous ses fameux outils de bureau, pour couper, coller, remodeler le texte, choisir les caractères, mélanger texte et dessin, etc.

MacDraw

Au commencement de l'édition, il y a l'écriture : avec des logiciels de traitement de texte comme MacWrite, Word ou Writer Plus, le vrai problème des auteurs c'est l'inspiration.

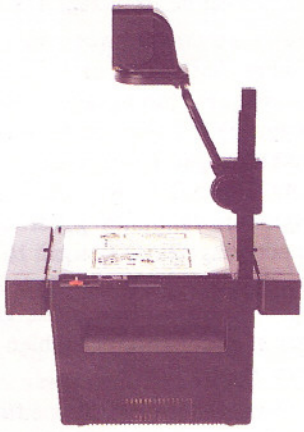
Le texte si brillant soit-il ne suffit pas. Avec MacPaint et MacDraw pour illustrer, tracer des schémas et des dessins techniques, définir des cadres, avec un logiciel comme Page Maker pour organiser et mettre en page, vos rapports d'entreprise, vos formulaires, vos manuels, votre journal interne, pour ne citer qu'eux, laisseront de vous une excellente impression. Et vous pourrez toujours tout modifier, y compris à la dernière minute !

MacWrite *MacDraw*

Pour Apple, il n'y a pas de petits et de grands écrivains, de littérature de bureau et de littérature tout court : tout le monde est logé à la même enseigne, celle de la qualité.

Avec la LaserWriter, une impression impeccable est à la portée de tous. Elle vous propose en effet onze familles de typographies (extension LaserWriter Plus), imprime sur papier, sur calque ou sur transparents et fournit des documents de qualité bromure.

Et si vous voulez vraiment faire les choses en grand, Macintosh peut aussi se connecter directement à une photocomposeuse d'imprimerie.



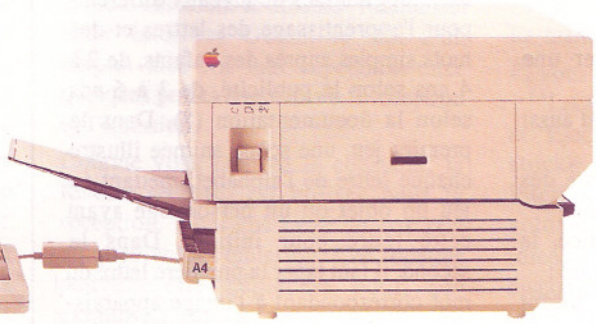
En fait, Apple ne vous offre rien de plus que votre imprimeur, sauf que vous n'aurez pratiquement plus besoin de lui.

Désormais, c'est vous, l'éditeur, qui éprouverez la sensation de Gutenberg il y a 436 ans lorsqu'il contempla son premier document.

C'est ainsi qu'Apple vous offre le meilleur de vous-même.

Page Maker

LaserWriter



Apple

Z.A. de Courtabœuf, B.P. 131, 91944 Les Ulis Cedex.

Record Holder

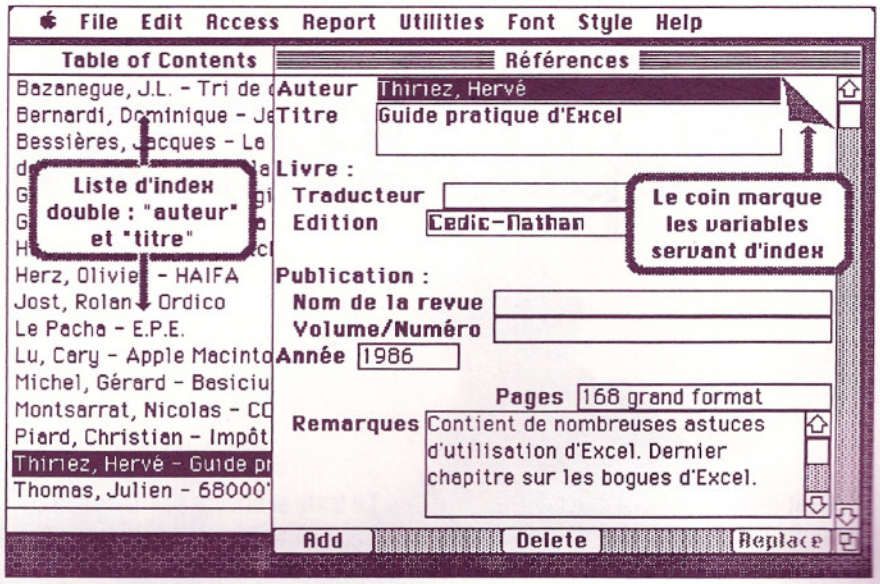
Vendu jusqu'au 30 septembre aux États-Unis au prix introductif de $49.95, ce programme devrait aujourd'hui coûter $20 ou $30 de plus, si la politique initiale a été maintenue. C'est un excellent gestionnaire de fichiers sans prétention, que j'ai appris à utiliser en moins d'une heure, aidé en cela par plus de 100 écrans d'assistance.

Les types de rubriques sont : text, nombre, monnaie, date, calculé et statistique (comptage, total et moyenne). Champ de texte : 32Ko maximum par information. Nombres jusqu'à 18 chiffres significatifs. Nombre maximum de fiches : limité par le support magnétique (jusqu'à 16 Mo sur un disque dur). Importation et exportation avec MacWrite. La copie d'écran ci-contre montre bien ce que donne Record Holder.

Nous avons récupéré, sans difficulté et en transitant par MacWrite, un fichier *File* de 270 enregistrements. Seul problème de transition : les virgules dans l'adresse (3, rue du Port) étaient interprétées comme des séparateurs de zones d'information; il a donc fallu, dans une seconde passe, les transformer en espaces, pour relire le fichier proprement dans Record Holder. À titre de comparaison, ce fichier se trie en 30 secondes dans File, et en moins de 15 dans Record Holder qui, par contre, ne fait des tris qu'en vue de l'impression ; heureusement, celle-ci peut être simulée à l'écran.

Avantages :

- il est très facile de rajouter une rubrique ;
- la mise en page d'une fiche est aussi facile qu'avec CX Base ;
- on peut voir à la fois la liste des fiches et une fiche particulière ;
- possibilité de choisir la police, la taille et le style de toute rubrique ;
- les champs calculés se construisent très facilement ;
- la gestion d'index avertit des doublons et les tolère éventuellement ;
- pas de protection, donc pas de problème avec un disque dur ;
- et, enfin, l'apprentissage est très rapide.



Inconvénients :

- pas de tri sauf pour l'impression ;
- l'impression d'étiquettes par 2 ou 3 de large n'est pas prévue ;
- pas d'importation d'images ;
- pas de décoration graphique des fiches.

Conclusion

Il y a peu de critiques à faire, en regard des avantages. Voilà donc un programme de gestion monofichier de grande qualité, à un rapport performance/prix le mettant hors concours.

Software Discoveries Inc., 99 Crestwood Road, Tolland, CT 06084, USA. Version US.

A B sCenes

Alliant graphismes et sons, ce sont trois programmes de niveaux différents pour l'apprentissage des lettres et des mots simples auprès des enfants, de 2 à 4 ans selon la publicité, de 3 à 6 ans selon la documentation (?). Dans le premier jeu, une scène animée illustre chaque lettre de l'alphabet, mettant en jeu un objet ou un personnage ayant cette lettre pour initiale. Dans le second, il faut taper la première lettre du mot correspondant à l'image apparaissant sur l'écran. Le troisième jeu demande la frappe du mot entier, toujours un mot simple.

Cette disquette est un bon exemple d'outil pédagogique simple et bien ficelé, en particulier par la variété des animations sur écran. A noter que la collection de Computeach comporte, pour diverses tranches d'âges et toujours au même prix, des disquettes d'arithmétique, de jeux de mots, d'apprentissage de signes, et même de création de scénario. Seule ABC cependant est disponible sur Macintosh ; certaines des autres n'existent que sur IBM PC.

Computeach, 240 Bradley Street, New Haven, CT 06511, USA.
Macintosh – Apple II – IBM PC – Prix $ 39,95.

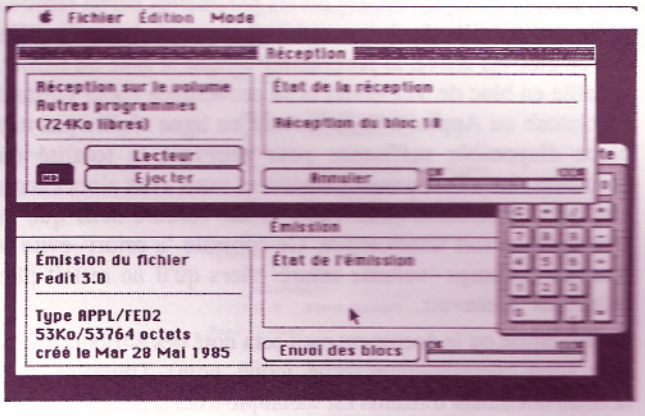
Les registres du +...

...ou plutôt le contenu des registres du MC68000 qui équipe votre Macintosh Plus, peut être, comme la mémoire (voir le numéro 27 de Pom's, page 42) visualisé :

- appuyer sur la touche 'INTERRUPT' ;
- taper 'D0' suivie d'un retour-chariot — le contenu (en hexadécimal) du registre d'adresse D0 apparaît ;
- même chose pour les autres registres de données (D0 à D7), ainsi que pour les registre d'adresses (A0 à A7, A7 étant le pointeur de pile) ;
- la valeur courante du registre d'état peut être obtenue en tapant 'SR' (Statut Register) ;
- celle du compteur ordinal en tapant 'PC' (Program Counter) ;
- 'G' provoque le retour au programme courant.

InterPom's
Téléchargement via MINITEL

Jean-Luc Bazanegue



InterPom's permet la transmission de n'importe quel type de fichier ou application, entre deux Macintosh — ou entre un Macintosh et un Apple // avec le programme compatible proposé page 15 — en utilisant le modem incorporé au Minitel. Le système utilise le même câble que le programme du précédent numéro de Pom's permettant, entre autres, le stockage et la restitution des écrans du Minitel ; si vous n'avez pas encore réalisé ce petit montage, nous joignons à cet article un schéma qui vous permettra de le réaliser, le détail du montage se trouve aussi dans le numéro 27. Si vous ne trouvez pas les connecteurs nécessaires à la réalisation ou si votre emploi du temps ne vous permet pas de faire le montage vous-même, il est désormais possible de se procurer un câble de liaison Macintosh/Minitel directement à la revue (n'oubliez pas de préciser de quel type de Mac il s'agit !). Le Minitel utilisé doit être du type 'retournable' (il doit pouvoir émettre à 1200 bauds), ce qui est la cas pour la quasi-majorité des modèles.

Possibilités et limites d'InterPom's

InterPom's peut faire passer tous les types de fichiers ou applications d'un ordinateur à l'autre par le réseau téléphonique par l'intermédiaire du Minitel. Le programme fonctionne indifféremment sur tous les Macintosh : 128Ko, 512Ko Macintosh Plus, en HFS, en MFS, sur disquettes (même avec un seul lecteur) et sur disques durs. La taille de la mémoire vive disponible n'a pas d'incidence directe sur la taille des fichiers transférables ; il est ainsi possible d'envoyer ou recevoir une application de 350Ko avec un Macintosh 128Ko, ou encore un fichier de 8 Méga (par exemple) si vous disposez d'un disque dur.

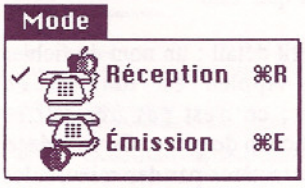
Les limites ne sont pas directement liées à InterPom's mais plutôt à son utilisation :

- grâce au type d'encodage utilisé, qui autorise des transmissions rapides

pour ce type de logiciel, il faut compter environ quinze secondes par Ko transmis : le fichier de 8 Mo précitée occuperait tout de même la ligne pendant un temps certain. Ce cas extrême (on ne transmet pas huit millions de caractères tous les jours !) peut constituer une limite. D'un point de vue plus pratique, une application comme MacPaint est transmise en 12 minutes, délai tout à fait acceptable ;

- une autre limite — puisqu'il faut en trouver une autre — est le fait qu'un

logiciel protégé le reste après transmission. Ainsi, s'il est possible d'envoyer Multiplan à un correspondant, le Multiplan recréé par InterPom's se comportera comme une copie 'normale' du logiciel, en réclamant l'insertion de l'original à l'ami récepteur. Gageons que cette limite — qui n'en est bien sûr pas vraiment une — fera plaisir aux auteurs de logiciels. En fait, s'il arrivait que certaines protections 'sautent' du fait d'une transmission par InterPom's, ce ne serait qu'involontaire.



Le programme, qui respecte scrupuleusement le célèbre 'Macintosh User Interface Guideline' est d'un emploi aisé ; il faut cependant citer quelques points communs à la réception et l'émission.

Comme vous pourrez le constater, la fenêtre du haut est utilisée pour la réception, alors que celle du bas sert uniquement à l'émission. Ces fenêtres peuvent être déplacées pour, par exemple, rendre plus visible un accessoire de bureau. Pour les remettre à leurs places respectives instantanément, il suffit le faire un 'clic' sur l'une ou l'autre des barres de déplacement en maintenant les touches 'Option' et 'Commande' enfoncées. Ceci ne modifie par la sélection et l'ordre courant des fenêtres.

Avant d'entreprendre une transmission de fichiers, il vous faudra bien sûr appeler votre correspondant, mais aussi mettre sous tension le Minitel (après on ne s'en occupe plus), sans avoir oublié la liaison Macintosh/Minitel.

Pendant la transmission, il convient de ne pas raccrocher le combiné téléphonique, ce qui vous permettra de converser avec votre correspondant entre ou après les transmissions.

Il y a trois solutions pour passer du mode Émission au mode Réception ou inversement :

- sélection de l'article du menu Mode ;
- Commande-E ou Commande-R ;
- sélection de la fenêtre par un 'clic'.

Réception d'un document

La manipulation est très simple :

- vous choisissez à l'aide des boutons 'Lecteur' (ou la touche 'Tabulation')

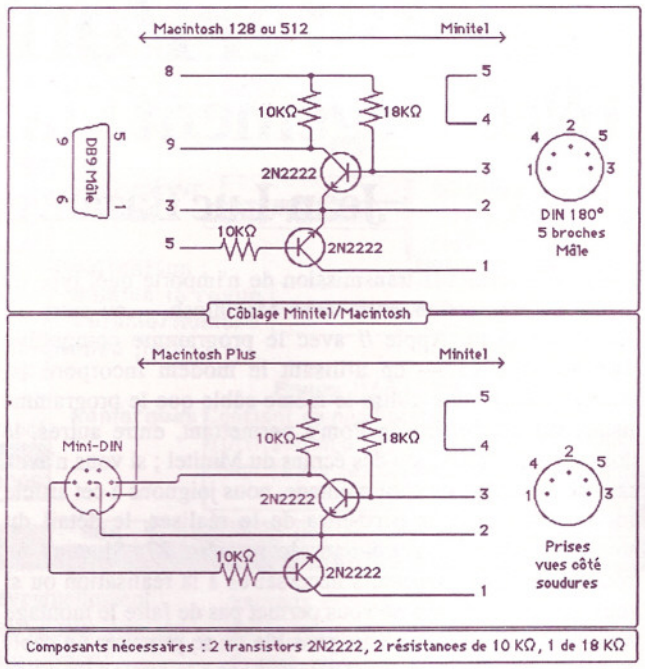
et 'Éjecter' le volume sur lequel sera stocké le document à recevoir. InterPom's vous indique le nom du volume en ligne, le type de volume (disquette ou disque dur) et la place disponible, en Ko. En cas de manque de place généralisé sur les volumes en lignes, il est possible d'insérer une disquette vierge que le programme se

chargera d'initialiser (après votre accord) ;

- prévenez votre correspondant et cliquez sur le bouton 'Envoi des Blocs'. Le programme connecte le Minitel, se met en attente du récepteur et, dès que "l'ami du bout du fil" clique sur 'Envoi des blocs' (il a environ une minute et demi pour le faire, après quoi la liaison est interrompue), la transmission commence.

Le premier bloc (chaque bloc correspond à 512 octets) est un bloc de contrôle qui contient divers paramètres parmi lesquels on trouve la taille du document, son nom, son type, etc. Dès que ce bloc est arrivé, le programme affiche le nom du fichier, sa taille en bloc de 512 octets et sa provenance (pour l'instant Macintosh ou Apple//). Si le volume en ligne n'offre pas une place disponible suffisante pour recevoir la totalité du document, InterPom's vous prévient et vous avez alors encore une minute et demi pour changer de volume sans que la transmission soit interrompue. Ce contrôle 'a priori' évite le frustrant message 'volume saturé' alors qu'il ne restait plus qu'un bloc à recevoir.

L'application ou le document reçu sera doté des attributs qu'il a chez votre correspondant (type, icône, nom). Toutefois, si le nom du document transmis est identique à celui d'un document existant sur le volume récepteur, InterPom's ajoute automatiquement le suffixe '.A' au nom actuel. Ainsi le fichier 'File' serait rebaptisé 'File.A'. Si un fichier 'File.A' existait aussi, 'File' serait rebaptisé 'File.B', et ainsi de suite...



Pendant la réception

Deux 'échelles' visualisent la transmission en cours. L'échelle supérieure (une simple ligne noire) représente le bloc de 512 octets alors que celle du bas, plus large indique le pourcentage de blocs reçus par rapport à la totalité.

Il est possible, aussi bien pour l'émetteur que pour le récepteur, d'interrompre la transmission en cours en cliquant sur 'Annuler'. Dans ce cas, le correspondant est prévenu, le Minitel est déconnecté et, au nom du fichier dont la réception était en cours est ajouté le suffixe '.INCOMPLET'.

Bien que les opérations se fassent d'une manière totalement automatique, des messages vous informent en permanence de l'état de la réception (Liaison avec l'émetteur, Attente du bloc N, Réception du bloc N, Réémission demandée, etc.).

Réception depuis un Apple //

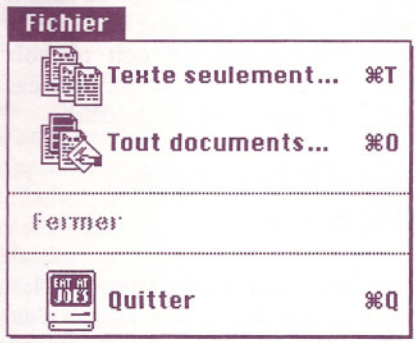
Si votre correspondant est muni d'un Apple // et de la version adéquate d'InterPom's, il peut vous envoyer des fichiers. Deux cas peuvent se présenter :

- il s'agit de fichiers de texte. Dans ce cas certains caractères sont recodés (é, è, à, {, etc.), le bit de poids fort de chaque caractère est forcé à 0 et InterPom's lui donne le type TEXT/MACA c'est-à-dire fichier texte/MacWrite. Ainsi, un double-clic sur le fichier créé provoquera le chargement de MacWrite, et le document sera directement exploitable ;
- il s'agit d'autres types de fichiers. Le contenu est laissé tel quel et on lui donne le type TEXT/EDIT, c'est-à-dire le type de l'éditeur utilisé avec, par exemple, le système de développement 68000. Le document sera aussi utilisable depuis MacWrite (ou tout autre programme de traitement de texte) mais ne pourra appeler par un double-clic que 'Edit'.

Dernier petit détail : un nom de fichier issu d'un Apple// est toujours en majuscule ; ce n'est pas très joli et prend beaucoup de place. On remplace donc les caractères par des minuscules en laissant tels quels le premier caractère du nom ainsi que, bien sûr, les caractères spéciaux comme '/'

Émission d'un document

La partie émission du programme à de nombreux points communs avec la section précédente, la plus grosse différence étant qu'au lieu de choisir un volume pour recevoir, on sélectionne un fichier à émettre. Cette opération se fera avec le menu Fichier.

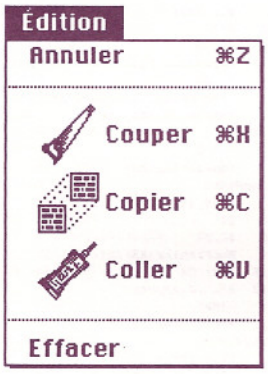


L'article 'Texte seulement...' (ou Commande-T) fait apparaître la fenêtre de sélection standard avec seulement les fichiers de type TEXT. L'article 'Tout documents...' (ou Commande-O) autorise la sélection de n'importe quel type de fichier, y compris les applications.

Dès qu'un fichier a été choisi, InterPom's affiche son nom, son type et son 'créateur' (ou APPL pour une application), sa taille en Ko et en octets ainsi que la date de sa création.

Pour le reste, la partie émission se comporte de la même façon que la réception.

Le menu Édition n'est pas utilisé par le programme ; il n'est donc valide que si un accessoire de bureau, qui pourrait utiliser ces fonctions, est sélectionné. Ceci est aussi valable pour l'article 'Fermer' du menu 'Fichier'.



Fichier 'InterPom's.Asm'

```
INCLUDE     Fichiers:InterPom's/1.Asm
INCLUDE     Fichiers:InterPom's/2.Asm
INCLUDE     Fichiers:InterPom's/3.Asm
INCLUDE     Fichiers:InterPom's/4.Asm
INCLUDE     Fichiers:InterPom's/5.Asm
END
```

Fichier 'InterPom's/1.Asm'

```
INCLUDE     Fichiers:FSEqu.Txt
INCLUDE     Fichiers:MacTraps.D
INCLUDE     Fichiers:SysEqu.D
INCLUDE     Fichiers:ToolEqu.D
INCLUDE     Fichiers:QuickEqu.D
INCLUDE     Fichiers:PackMacs.Txt
MenuPomme       EQU     1
 mAPropos       EQU     1
MenuFichier     EQU     2
 mOuvrir1       EQU     1
 mOuvrir2       EQU     2
 mFermer        EQU     4
 mQuitter       EQU     6
MenuEdition     EQU     3
 mAnnuler       EQU     1
 mCouper        EQU     3
 mCopier        EQU     4
 mColler        EQU     5
 mEffacer       EQU     7
MenuMode        EQU     4
 mReception     EQU     1
 mEnvoi         EQU     2
DialogAPropos   EQU     1
DialogBis       EQU     501
bOK             EQU     1
SOH             EQU     1
EOT             EQU     4
ACK             EQU     6
NACK            EQU     21
CAN             EQU     25
CAN_U           EQU     2
CAN_A           EQU     5
DIF             EQU     3
CHECK           EQU     7
AppleII         EQU     1
DebutBloc       EQU     0
NumBloc         EQU     1
NotNumBloc      EQU     2
TypeEmetteur    EQU     4
TypeTexteMac    EQU     5
NBlocsData      EQU     6
NBlocsResource  EQU     8
NBlocsTotal     EQU     10
TypeTexteApple  EQU     88
NBlocsApple     EQU     92
LongLog_Apple   EQU     96
NomFichierApple EQU     100
Date            EQU     164
Heure           EQU     250
LongueurBloc    EQU     512
LongueurBlocC   EQU     683
NombreMaxEssai  EQU     10
SixFaibles      EQU     $3F
DeuxForts       EQU     $C0
    BSR.S   Initialisations
    BSR.S   Menus
    BSR     OuvreSerie
    BSR     OuvertureF
    BRA     Evenements
```

```
Initialisations
    PEA     -4(A5)
    _InitGraf
    _InitFonts
    MOVE.L  #$0000FFFF,D0
    _FlushEvents
    _InitWindows
    _InitMenus
    CLR.L   -(SP)
    _InitDialogs
    _TEInit
    _InitCursor
    CLR.B   NomVolume(A5)
    CLR.B   NomFichier(A5)
    CLR.B   DefautEjecte(A5)
    CLR.B   DrChoixFichier(A5)
    MOVE.L  #$FF00FF00,DrMenuFichier(A5)
    CLR.B   $2F8
    MOVE    #DILoad,-(SP)
    _Pack2
    RTS
Menus
    SUBQ.L  #4,SP
    MOVE    #MenuPomme,-(SP)
    _GetRMenu
    MOVE.L  (SP),HandleMPomme(A5)
    MOVE.L  HandleMPomme(A5),-(SP)
    CLR     -(SP)
    _InsertMenu
    MOVE.L  #'DRVR',-(SP)
    _AddResMenu
    SUBQ.L  #4,SP
    MOVE    #MenuFichier,-(SP)
    _GetRMenu
    MOVE.L  (SP),HandleMFichier(A5)
    CLR     -(SP)
    _InsertMenu
    SUBQ.L  #4,SP
    MOVE    #MenuEdition,-(SP)
    _GetRMenu
    MOVE.L  (SP),HandleMEdition(A5)
    CLR     -(SP)
    _InsertMenu
    SUBQ.L  #4,SP
    MOVE    #MenuMode,-(SP)
    _GetRMenu
    MOVE.L  (SP),HandleMMode(A5)
    CLR     -(SP)
    _InsertMenu
    MOVE.L  HandleMEdition(A5),-(SP)
    CLR     -(SP)
    _DisableItem
    _DrawMenuBar
    RTS
Evenements
    TST.B   DefautEjecte(A5)
    BEQ.S   @1
    MOVEA.L defVCBPtr,A3
    TST     vcbDrvNum(A3)
    BEQ.S   @1
    CLR.B   DefautEjecte(A5)
    BSR     PlusieursVolumes
    BSR     VolumeSuivant2
    BSR     AffNom
@1  _SystemTask
    SUBQ.L  #2,SP
    MOVE    #$0FFF,-(SP)
    PEA     EnregEvents(A5)
    _GetNextEvent
    TST     (SP)+
    BEQ.S   Evenements
    BSR.S   Traitementevents
    BEQ.S   Evenements
    RTS
Traitementevents
    MOVE    EnregEvents+evtNum(A5),D0
    ADD     D0,D0
    MOVE    TableEvenements(D0),D0
    JMP     TableEvenements(D0)
TableEvenements
    DC      Eventsuivant-TableEvenements
    DC      Souris-TableEvenements
    DC      Eventsuivant-TableEvenements
    DC      Touche-TableEvenements
    DC      Eventsuivant-TableEvenements
    DC      Touche-TableEvenements
    DC      MiseJour-TableEvenements
    DC      Disquette-TableEvenements
    DC      Active-TableEvenements
    DC      Eventsuivant-TableEvenements
    DC      Eventsuivant-TableEvenements
    DC      Eventsuivant-TableEvenements
Eventsuivant
    MOVEQ   #0,D0
    RTS
Souris
    SUBQ.L  #2,SP
    MOVE.L  EnregEvents+evtMouse(A5),-(SP)
    PEA     EnregEvents+evtMeta+2(A5)
    _FindWindow
    MOVE    (SP)+,D0
    ADD     D0,D0
    MOVE    TableFenetre(D0),D0
    JMP     TableFenetre(D0)
TableFenetre
    DC      Eventsuivant-TableFenetre
    DC      BarMenu-TableFenetre
    DC      fenetreSys-TableFenetre
    DC      Contenu-TableFenetre
    DC      DeplaceFen-TableFenetre
    DC      Contenu-TableFenetre
    DC      Eventsuivant-TableFenetre
BarMenu
    SUBQ.L  #4,SP
    MOVE.L  EnregEvents+evtMouse(A5),-(SP)
    _MenuSelect
    MOVE    (SP)+,D1
    MOVE    (SP)+,D0
BarMenuC
    ADD     D1,D1
    MOVE    TableMenus(D1),D1
    JMP     TableMenus(D1)
TableMenus
    DC      Eventsuivant-TableMenus
    DC      TraitPomme-TableMenus
    DC      TraitFichier-TableMenus
    DC      TraitEdition-TableMenus
    DC      TraitMode-TableMenus
FinMenu
    CLR     -(SP)
    _HiliteMenu
    BRA.S   Eventsuivant
TraitFichier
    ADD     D0,D0
    MOVE    TableFichier(D0),D0
    JMP     TableFichier(D0)
TableFichier
    DC      FinMenu-TableFichier
    DC      TmOuvrir1-TableFichier
    DC      TmOuvrir2-TableFichier
    DC      FinMenu-TableFichier
    DC      TmFermer-TableFichier
    DC      FinMenu-TableFichier
    DC      TmQuitter-TableFichier
Touche
    BTST    #CmdKey,EnregEvents+evtMeta(A5)
    BNE.S   Commande
    TST.B   DrBoutonLecteur(A5)
    BNE.S   @1
    CMPI.B  #9,EnregEvents+evtMessage+3(A5)
    BNE.S   @1
    CLR.L   -(SP)
    BSR     Boutons
@1  BRA     Eventsuivant
Commande
    CLR.L   -(SP)
    MOVE    EnregEvents+evtMessage+2(A5),-(SP)
    _MenuKey
    MOVE    (SP)+,D1
    MOVE    (SP)+,D0
    BRA     BarMenuC
MiseJour
    BSR     SPMiseJour
    BRA     Eventsuivant
SPMiseJour
    MOVEA.L EnregEvents+evtMessage(A5),A4
    MOVE.L  A4,-(SP)
    MOVE.L  (SP),-(SP)
    _BeginUpdate
    _SetPort
    TST.L   wRefCon(A4)
    BNE.S   @1
    BSR     MiseJour1
    BRA.S   @2
@1  BSR     MiseJour2
@2  MOVE.L  A4,-(SP)
    _EndUpdate
    RTS
MiseJour1
    MOVE.L  A4,-(SP)
    _DrawControls
    BSR     AfficheContenuFR
    RTS
MiseJour2
    MOVE.L  A4,-(SP)
    _DrawControls
    BSR     ContenuCommun
    MOVEA.L (A5),A0
    PEA     gray(A0)
    _PenPat
    MOVE.L  #$00300010,-(SP)
    _MoveTo
    MOVE.L  #$003000B7,-(SP)
    _LineTo
    _PenNormal
    MOVE.L  #$00180010,-(SP)
    _MoveTo
    PEA     ChaineEmission1
    _DrawString
    MOVE.L  #$00480010,-(SP)
    _MoveTo
    PEA     ChaineEmission2
    _DrawString
    MOVE.L  #$00680010,-(SP)
    _MoveTo
```

Fichier 'InterPom's.Job'

```
MDS:ASM         Fichiers:InterPom's.Asm     MDS:Exec    MDS:Edit
MDS:LINK        Fichiers:InterPom's.Link    MDS:Exec    MDS:Edit
MDS:RMAKER      Fichiers:InterPom's.R       MDS:Finder  MDS:Edit
```

```
    PEA      ChaineEmission5
    _DrawString
    MOVE.L   #$001800D0,-(SP)
    _MoveTo
    PEA      ChaineEtatE
    _DrawString
    TST.B    NomFichier(A5)
    BEQ.S    @1
    BSR      AfficheContenuF2
@1  RTS
Disquette
    MOVE     EnregEvents+evtMessage+2(A5),D5
    TST      EnregEvents+evtMessage(A5)
    BEQ.S    @3
    SUBQ.L   #2,SP
    MOVE.L   #$640064,-(SP)
    MOVE.L   EnregEvents+evtMessage(A5),-(SP)
    MOVE     #DIBadMount,-(SP)
    _Pack2
    TST      (SP)+
    BNE.S    @4
@3  MOVE.L   vcbQHdr+2,A3
@2  CMP      vcbDrvNum(A3),D5
    BEQ.S    @1
    MOVE.L   (A3),A3
    BRA.S    @2
@1  BSR      PlusieursVolumes
    BNE.S    @5
    BSR      PasVolume
    BRA.S    @4
@5  BSR      VolumeSuivant2
    BSR      AffNom
@4  BRA      Eventsuivant
Active
    MOVEA.L  EnregEvents+evtMessage(A5),A4
    CMPI     #userKind,windowKind(A4)
    BNE      Eventsuivant
    MOVE     EnregEvents+evtMeta(A5),D7
    BTST     #ActiveFlag,D7
    BEQ.S    DeActiver
    MOVE.L   A4,-(SP)
    _SetPort
DeActiver
    BSR      ValideMenus
    BRA      Eventsuivant
fenetreSys
    PEA      EnregEvents(A5)
    MOVE.L   EnregEvents+evtMeta+2(A5),-(SP)
    _SystemClick
    BRA      Eventsuivant
DeplaceFen
    MOVE     EnregEvents+evtMeta(A5),D0
    ANDI     #2048+256,D0
    CMPI     #2048+256,D0
    BNE.S    @1
    SUBQ.L   #4,SP
    _FrontWindow
    MOVE.L   (SP)+,D3
    MOVE.L   PointeurFenetre1(A5),-(SP)
    LEA      RectFenetre1,A3
    MOVE.L   (A3),-(SP)
    CMP.L    PointeurFenetre1(A5),D3
    SEQ      -(SP)
    _MoveWindow
    MOVE.L   PointeurFenetre2(A5),-(SP)
    MOVE.L   8(A3),-(SP)
    CMP.L    PointeurFenetre2(A5),D3
    SEQ      -(SP)
    _MoveWindow
    BRA.S    @2
@1  MOVE.L   EnregEvents+evtMeta+2(A5),A4
    MOVE.L   A4,-(SP)
    MOVE.L   EnregEvents+evtMouse(A5),-(SP)
    PEA      Rectanglelimite
    _DragWindow
    TST.L    wRefCon(A4)
    BNE.S    @3
    BSR      MarqueMode1
    BRA.S    @2
@3  BSR      MarqueMode2
@2  BRA      Eventsuivant
Contenu
    SUBQ.L   #4,SP
    _FrontWindow
    MOVE.L   (SP)+,D0
    MOVEA.L  EnregEvents+evtMeta+2(A5),A4
    CMP.L    A4,D0
    BEQ.S    @1
    MOVE.L   A4,-(SP)
    _SelectWindow
    MOVE.L   wRefCon(A4),D0
    BNE.S    @3
    BSR      MarqueMode1
    BRA.S    @2
@3  BSR      MarqueMode2
    BRA.S    @2
@1  MOVE.L   A4,-(SP)
    _SetPort
    PEA      EnregEvents+EvtMouse(A5)
    _GlobalToLocal
    SUBQ.L   #2,SP
    MOVE.L   EnregEvents+EvtMouse(A5),-(SP)
    MOVE.L   A4,-(SP)
    PEA      HandleControle(A5)
    _FindControl
    TST      (SP)+
    BEQ.S    @2
    SUBQ.L   #2,SP
    MOVE.L   HandleControle(A5),-(SP)
    MOVE.L   EnregEvents+EvtMouse(A5),-(SP)
    CLR.L    -(SP)
    _TrackControl
    TST      (SP)+
    BEQ.S    @2
    SUBQ.L   #4,SP
    MOVE.L   HandleControle(A5),-(SP)
    _GetCRefCon
    BSR      Boutons
@2  BRA      Eventsuivant
TraitPomme
    CMPI     #mAPropos,D0
    BEQ.S    TraiteAPropos
    MOVE.L   HandleMPomme(A5),-(SP)
    MOVE     D0,-(SP)
    PEA      NomAccessoire(A5)
    _GetItem
    SUBQ.L   #2,SP
    PEA      NomAccessoire(A5)
    _OpenDeskAcc
    ADDQ     #2,SP
    BRA      FinMenu
TraiteAPropos
    CLR.L    -(SP)
    MOVE     #DialogAPropos,-(SP)
    PEA      Dialogue(A5)
    MOVE.L   #-1,-(SP)
    _GetNewDialog
    MOVE.L   (SP),-(SP)
    _SetPort
Attente
    CLR.L    -(SP)
    PEA      Article(A5)
    _ModalDialog
    _CloseDialog
    SUBQ.L   #4,SP
    _FrontWindow
    MOVE.L   (SP)+,A4
    CMPI     #userKind,windowKind(A4)
@1  BNE      FinMenu
    MOVE.L   A4,-(SP)
    MOVE.L   (SP),-(SP)
    _SelectWindow
    _SetPort
    BRA.S    @1
TraitEdition
    BSR.S    VersSysteme
    BRA      FinMenu
VersSysteme
    MOVE     D0,D7
    CLR      -(SP)
    MOVE     D0,-(SP)
    SUBQ     #1,(SP)
    _SysEdit
    MOVE.B   (SP)+,D1
    RTS
TraitMode
    SUBQ     #1,D0
    BNE.S    Mode2
    MOVE.L   PointeurFenetre1(A5),-(SP)
    _SelectWindow
    BSR      MarqueMode1
    BRA.S    FinTraitMode
Mode2
    MOVE.L   PointeurFenetre2(A5),-(SP)
    _SelectWindow
    BSR      MarqueMode2
FinTraitMode
    BRA      FinMenu
TmOuvrir1
    LEA      TypeTEXT,A1
    MOVE.L   A1,D0
    MOVEQ    #1,D1
    BRA      TmOuvrir
TmOuvrir2
    MOVEQ    #0,D0
    MOVEQ    #-1,D1
TmOuvrir
    MOVE.L   #$640064,-(SP)
    CLR.L    -(SP)
    CLR.L    -(SP)
    MOVE     D1,-(SP)
    MOVE.L   D0,-(SP)
    CLR.L    -(SP)
    PEA      ReponseGetFile(A5)
    MOVE     #SFGetFile,-(SP)
    _Pack3
    TST      ReponseGetFile+rGood(A5)
    BEQ      FinMenu
    LEA      TamponIO(A5),A0
    LEA      ReponseGetFile+rName(A5),A1
    MOVE.L   A1,ioFileName(A0)
    MOVE     ReponseGetFile+rVolume(A5),
             ioVRefNum(A0)
    CLR      ioFDirIndex(A0)
    CLR.B    ioFileType(A0)
    _GetFileInfo
    BEQ.S    @7
    CLR      -(SP)
@11 BSR      Erreur
    BRA      FinMenu
@7  LEA      TamponIO(A5),A0
    TST.B    ioFlAttrib(A0)
    BGE.S    @10
    MOVE     #1,-(SP)
    BRA.S    @11
@10 MOVEQ    #0,D0
    MOVE.B   ReponseGetFile+rName(A5),D0
    ADDQ     #1,D0
    LEA      ReponseGetFile+rName(A5),A0
    LEA      NomFichier(A5),A1
    _BlockMove
    CLR      -(SP)
    PEA      NomFichier(A5)
    _StringWidth
    CMPI     #168,(SP)+
    BLT.S    @2
    MOVEQ    #0,D4
    LEA      NomFichier(A5),A4
    MOVE.B   (A4),D4
@3  MOVE.B   #'…',0(A4,D4)
    CLR      -(SP)
    PEA      NomFichier(A5)
    _StringWidth
    CMPI     #168,(SP)+
    BLT.S    @2
    SUBQ.B   #1,(A4)
    SUBQ     #1,D4
    BRA.S    @3
@2  LEA      TamponIO(A5),A0
    LEA      TamponTemporaire(A5),A1
    MOVEQ    #ioFQElSize,D0
    _BlockMove
    LEA      TamponIO(A5),A0
    MOVE.L   ioFlRLgLen(A0),D0
    ADD.L    ioFlLgLen(A0),D0
    MOVE.L   D0,D5
    LEA      NombreOctets(A5),A0
    CLR      -(SP)
    _Pack7
    MOVE.L   D5,D0
    LEA      ChaineEmission4,A0
    CMPI.L   #1,D5
    BGT.S    @4
    MOVE.B   #6,(A0)
    BRA.S    @5
@4  MOVE.B   #7,(A0)
@5  MOVE     #1024,D1
    DIVU     D1,D0
    SWAP     D0
    TST      D0
    BEQ.S    @9
    SWAP     D0
    ADDQ     #1,D0
    BRA.S    @1
@9  SWAP     D0
@1  ANDI.L   #$FFFF,D0
    LEA      NombreKoEnvoi(A5),A0
    CLR      -(SP)
    _Pack7
    MOVE.L   TamponIO+ioFlUsrWds+fdType(A5),
             TypeFichier+2(A5)
    MOVE.L   TamponIO+ioFlUsrWds+fdCreator(A5),
             CreateurFichier+2(A5)
    MOVE.B   #4,TypeFichier+1(A5)
    MOVE.B   #4,CreateurFichier+1(A5)
    MOVE.L   TamponIO+ioFlCrDat(A5),-(SP)
    MOVE.B   #abbrevDate,-(SP)
    PEA      ChaineDate(A5)
    MOVE     #IUDateString,-(SP)
    _Pack6
    PEA      SauvePort(A5)
    _GetPort
    MOVE.L   PointeurFenetre2(A5),-(SP)
    _SetPort
    PEA      RectangleEffaceFichier
    _EraseRect
    PEA      RectangleEffaceInfo1
    _EraseRect
    PEA      RectangleEffaceInfo2
    _EraseRect
    PEA      RectangleEffaceInfo3
    _EraseRect
    BSR      AfficheContenuF2
    TST.B    DrChoixFichier(A5)
    BNE.S    @8
    ST       DrChoixFichier(A5)
    MOVE.L   HandleControle1F2(A5),-(SP)
    CLR      -(SP)
    _HiliteControl
@8  MOVE.L   SauvePort(A5),-(SP)
    _SetPort
    BRA      FinMenu
TmFermer
    SUBQ.L   #4,SP
    _FrontWindow
    MOVEA.L  (SP)+,A4
    MOVE     windowKind(A4),-(SP)
    _CloseDeskAcc
    BRA      FinMenu
TmQuitter
    MOVE     #-1,D0
    RTS
OuvertureF
    BSR      Montre
    SUBQ.L   #4,SP
    CLR.L    -(SP)
    PEA      RectFenetre2
    PEA      Titre2
    MOVE.B   #1,-(SP)
    MOVE     #documentProc,-(SP)
```

```
    CLR.L   -(SP)
    CLR     -(SP)
    MOVEQ   #1,D0
    MOVE.L  D0,-(SP)
    _NewWindow
    MOVEA.L (SP),A4
    _SetPort
    MOVE    #sysFont,-(SP)
    _TextFont
    MOVE    #12,-(SP)
    _TextSize
    SUBQ.L  #4,SP
    MOVE.L  A4,-(SP)
    PEA     RectangleBouton3
    PEA     TitreBouton1F2
    MOVE.B  #1,-(SP)
    CLR.L   -(SP)
    CLR.L   -(SP)
    MOVEQ   #3,D0
    MOVE.L  D0,-(SP)
    _NewControl
    MOVE.L  (SP),HandleControle1F2(A5)
    MOVE    #-1,-(SP)
    _HiliteControl
    MOVE.L  A4,PointeurFenetre2(A5)
    SUBQ.L  #4,SP
    CLR.L   -(SP)
    PEA     RectFenetre1
    PEA     Titre1
    MOVE.B  #1,-(SP)
    MOVE    #documentProc,-(SP)
    MOVEQ   #-1,D0
    MOVE.L  D0,-(SP)
    CLR     -(SP)
    CLR.L   -(SP)
    _NewWindow
    MOVEA.L (SP),A4
    _SetPort
    MOVE    #sysFont,-(SP)
    _TextFont
    MOVE    #12,-(SP)
    _TextSize
    SUBQ.L  #4,SP
    MOVE.L  A4,-(SP)
    PEA     RectangleBouton1
    PEA     TitreBouton1
    MOVE.B  #1,-(SP)
    CLR.L   -(SP)
    CLR.L   -(SP)
    CLR.L   -(SP)
    _NewControl
    MOVE.L  (SP),HandleControle1(A5)
    MOVE.L  A4,-(SP)
    PEA     RectangleBouton2
    PEA     TitreBouton2
    MOVE.B  #1,-(SP)
    CLR.L   -(SP)
    CLR.L   -(SP)
    MOVEQ   #1,D0
    MOVE.L  D0,-(SP)
    _NewControl
    MOVE.L  (SP)+,HandleControle2(A5)
    SUBQ.L  #4,SP
    MOVE.L  A4,-(SP)
    PEA     RectangleBouton3
    PEA     TitreBouton3
    MOVE.B  #1,-(SP)
    CLR.L   -(SP)
    CLR.L   -(SP)
    MOVEQ   #2,D0
    MOVE.L  D0,-(SP)
    _NewControl
    MOVE.L  (SP)+,HandleControle3(A5)
    MOVE.L  A4,PointeurFenetre1(A5)
    BSR     PlusieursVolumes
    BNE.S   @1
    BSR     PasVolume
    BRA.S   @2
@1  BSR     VolumeSuivant
@2  _InitCursor
    RTS
Montre
    SUBQ.L  #4,SP
    MOVE    #watchCursor,-(SP)
    _GetCursor
    MOVEA.L (SP)+,A0
    MOVE.L  (A0),-(SP)
    _Setcursor
    RTS
AfficheContenuFR
    BSR     ContenuCommun
    MOVEA.L (A5),A0
    PEA     gray(A0)
    _PenPat
    MOVE.L  #$00400010,-(SP)
    _MoveTo
    MOVE.L  #$004000B7,-(SP)
    _LineTo
    _PenNormal
    MOVE.L  #$00180010,-(SP)
    _MoveTo
    PEA     ChaineReception1
    _DrawString
    MOVE.L  #$001800D0,-(SP)
    _MoveTo
    PEA     ChaineEtatR
    _DrawString
AffNom
    PEA     SauvePort(A5)
    _GetPort
    MOVE.L  PointeurFenetre1(A5),-(SP)
    _SetPort
    TST.B   NomVolume(A5)
    BEQ.S   @1
    PEA     RectangleEfface
    _EraseRect
    MOVE.L  #$00280010,-(SP)
    _MoveTo
    PEA     NomVolume(A5)
    _DrawString
    MOVE.L  #$00380010,-(SP)
    _MoveTo
    MOVE    #' (',-(SP)
    _DrawChar
    PEA     NombreK(A5)
    _DrawString
    PEA     ChaineReception2
    _DrawString
    BSR     AfficheDriver
    TST.B   DrapeauVV(A5)
    BEQ.S   @1
    MOVEA.L (A5),A0
    PEA     gray(A0)
    _PenPat
    MOVE    #notPatBic,-(SP)
    _PenMode
    PEA     RectangleIcone
    _PaintRect
    PEA     RectangleEfface
    _PaintRect
    _PenNormal
    MOVE.L  SauvePort(A5),-(SP)
    _SetPort
@1  RTS
AfficheDriver
Tampon  SET     -bitmaprec
    LINK    A6,#Tampon
    MOVEM.L A0-A4/D0-D2,-(SP)
    SUBQ.L  #4,SP
    MOVE    DriverCourant(A5),-(SP)
    _GetIcon
    MOVEA.L (SP)+,A0
    MOVE.L  (A0),Tampon+baseAddr(A6)
    MOVE    #4,Tampon+rowBytes(A6)
    CLR.L   Tampon+bounds(A6)
    MOVE.L  #$200020,Tampon+bounds+bottom(A6)
    PEA     Tampon+baseAddr(A6)
    MOVEA.L PointeurFenetre1(A5),A0
    ADDQ.L  #2,A0
    MOVE.L  A0,-(SP)
    PEA     Tampon+bounds(A6)
    PEA     RectangleIcone
    CLR     -(SP)
    CLR.L   -(SP)
    _CopyBits
    MOVEM.L (SP)+,A0-A4/D0-D2
    UNLK    A6
    RTS
Boutons
    LINK    A6,#0
    MOVE    10(A6),D0
    ADD     D0,D0
    MOVE    TableBoutons(D0),D0
    JMP     TableBoutons(D0)
TableBoutons
    DC      BoutonLecteur-TableBoutons
    DC      BoutonEjecter-TableBoutons
    DC      BoutonAttente-TableBoutons
    DC      BoutonEnvoi-TableBoutons
BoutonLecteur
    BSR     VolumeSuivant
    BSR     AffNom
    BRA.S   ETQ2
BoutonEjecter
    MOVE    VolumeCourant(A5),
            TamponIO+ioVRefNum(A5)
    CLR.L   TamponIO+ioCompletion(A5)
    CLR.L   TamponIO+ioVNPtr(A5)
    LEA     TamponIO(A5),A0
    _Eject
    BEQ.S   @1
    CLR     -(SP)
    BSR     Erreur
    BRA.S   SortieBoutons
@1  BSR     PlusieursVolumes
    BNE.S   @3
    BSR     PasVolume
    BRA.S   ETQ2
@3  BSR     VolumeSuivant
    BSR     AffNom
ETQ2
    MOVEA.L defVCBPtr,A1
    TST     vcbDrvNum(A1)
    SEQ     DefautEjecte(A5)
SortieBoutons
    UNLK    A6
    MOVE.L  (SP)+,(SP)
    RTS
BoutonAttente
    BSR     Reception
BoutonAttente2
    MOVEQ   #2,D0
    _FlushEvents
    BRA.S   SortieBoutons
BoutonEnvoi
    BSR     Emission
    BRA.S   BoutonAttente2
PlusieursVolumes
    LINK    A6,#0
    MOVEQ   #0,D5
    MOVE.L  vcbQHdr+2,A1
    BRA.S   @4
@2  MOVEA.L (A1),A1
@4  MOVE.L  A1,D1
    BEQ.S   @1
    TST     vcbDrvNum(A1)
    BEQ.S   @2
@3  ADDQ    #1,D5
    BRA.S   @2
@1  MOVE.L  HandleControle1(A5),-(SP)
    CMPI    #2,D5
    SMI     D4
    MOVE.B  D4,DrBoutonLecteur(A5)
    MOVE    D4,-(SP)
    _HiliteControl
    TST     D5
    UNLK    A6
    RTS
VolumeSuivant2
    LINK    A6,#0
    BRA.S   E8
VolumeSuivant
    LINK    A6,#0
    MOVE.L  vcbQHdr+2,A3
    TST.B   NomVolume(A5)
    BEQ.S   @1
    MOVE    VolumeCourant(A5),D0
@7  CMP     vcbVRefNum(A3),D0
    BEQ.S   @6
    MOVE.L  (A3),A3
    MOVE.L  A3,D1
    BNE.S   @7
    MOVE.L  vcbQHdr+2,A3
    BRA.S   @7
@6  MOVE.L  (A3),A3
    MOVE.L  A3,D1
    BNE.S   @1
    MOVE.L  vcbQHdr+2,A3
@1  TST     vcbDrvNum(A3)
    BNE.S   @8
    TST     vcbDRefNum(A3)
    BLE.S   @8
    MOVE    vcbVRefNum(A3),D0
    TST.B   NomVolume(A5)
    BNE.S   @7
    PEA     RectangleIcone
    _EraseRect
    PEA     RectangleEfface
    _EraseRect
    UNLK    A6
    RTS
@8
E8  MOVE    vcbVRefNum(A3),VolumeCourant(A5)
    MOVEQ   #0,D0
    MOVE.B  vcbVN(A3),D0
    ADDQ    #1,D0
    LEA     vcbVN(A3),A0
    LEA     NomVolume(A5),A1
    _BlockMove
    CLR     -(SP)
    PEA     NomVolume(A5)
    _StringWidth
    CMPI    #168,(SP)+
    BLT.S   @2
    MOVEQ   #0,D4
    LEA     NomVolume(A5),A4
    MOVE.B  (A4),D4
@3  MOVE.B  #'_',0(A4,D4)
    CLR     -(SP)
    PEA     NomVolume(A5)
    _StringWidth
    CMPI    #168,(SP)+
    BLT.S   @2
    SUBQ.B  #1,(A4)
    SUBQ    #1,D4
    BRA.S   @3
@2  MOVE.L  vcbAlBlkSiz(A3),D0
    MOVE    vcbFreeBks(A3),D1
    MOVEQ   #9,D2
    LSR.L   D2,D0
    MULU    D1,D0
    LSR.L   #1,D0
    MOVE.L  D0,NombreKONum(A5)
    LEA     ChaineReception2,A1
    CMPI    #1,D0
    SMI     PasDePlace(A5)
    BGT.S   @4
    MOVE.B  #9,(A1)
    MOVE.B  #')',9(A1)
    BRA.S   @5
@4  MOVE.B  #10,(A1)
    MOVE.B  #'s',9(A1)
@5  LEA     NombreK(A5),A0
    CLR     -(SP)
    _Pack7
    MOVE    vcbAtrb(A3),D0
    ANDI    #$8080,D0
    SNE     DrapeauVV(A5)
    BSR     ChercheDriver
    MOVE.L  HandleControle2(A5),-(SP)
    CMPI.B  #8,-3(A3)
    SEQ     D3
    MOVE    D3,-(SP)
```

```
    _HiliteControl
    MOVE.L  HandleControle3(A5),-(SP)
    MOVE.B  DrapeauVV(A5),D0
    OR.B    PasDePlace(A5),D0
    EXT     D0
    MOVE    D0,-(SP)
    _HiliteControl
    TST.B   D3
    BNE.S   @6
    MOVE    #267,DriverCourant(A5)
    BRA.S   @7
@6  MOVE    #268,DriverCourant(A5)
@7  UNLK    A6
    RTS
ChercheDriver
    MOVE    vcbDrvNum(A3),D0
    MOVEA.L drvQHdr+2,A3
@2  CMP     dQDrive(A3),D0
    BNE.S   @1
    RTS
@1  MOVEA.L qLink(A3),A3
    MOVE.L  A3,D1
    BNE.S   @2
    RTS
PasVolume
    PEA     RectangleIcone
    _EraseRect
    PEA     RectangleEfface
    _EraseRect
    MOVE.L  HandleControle2(A5),-(SP)
    MOVE    #-1,-(SP)
    _HiliteControl
    CLR.B   NomVolume(A5)
    RTS
ValideMenus
    SUBQ.L  #4,SP
    _FrontWindow
    MOVEA.L (SP)+,A4
    MOVEQ   #0,D5
    CMP     #UserKind,windowKind(A4)
    BNE.S   FenetreAccessoire
    TST.B   DrMenuFichier(A5)
    BNE.S   @1
    ST      DrMenuFichier(A5)
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mOuvrir1,-(SP)
    _EnableItem
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mOuvrir2,-(SP)
    _EnableItem
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mFermer,-(SP)
    _DisableItem
    ADDQ    #1,D5
@1  TST.B   DrMenuEdition(A5)
    BEQ.S   @2
    SF      DrMenuEdition(A5)
    MOVE.L  HandleMEdition(A5),-(SP)
    CLR     -(SP)
    _DisableItem
    ADDQ    #1,D5
@2  TST.B   DrMenuMode(A5)
    BNE.S   @3
    ST      DrMenuMode(A5)
    MOVE.L  HandleMMode(A5),-(SP)
    CLR     -(SP)
    _EnableItem
    ADDQ    #1,D5
@3  TST     D5
    BEQ     @4
    _DrawMenuBar
@4  RTS
FenetreAccessoire
    TST.B   DrMenuFichier(A5)
    BEQ.S   @1
    SF      DrMenuFichier(A5)
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mOuvrir1,-(SP)
    _DisableItem
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mOuvrir2,-(SP)
    _DisableItem
    MOVE.L  HandleMFichier(A5),-(SP)
    MOVE    #mFermer,-(SP)
    _EnableItem
    ADDQ    #1,D5
@1  TST.B   DrMenuEdition(A5)
    BNE.S   @2
    ST      DrMenuEdition(A5)
    MOVE.L  HandleMEdition(A5),-(SP)
    CLR     -(SP)
    _EnableItem
    ADDQ    #1,D5
@2  TST.B   DrMenuMode(A5)
    BEQ.S   @3
    SF      DrMenuMode(A5)
    MOVE.L  HandleMMode(A5),-(SP)
    CLR     -(SP)
    _DisableItem
    ADDQ    #1,D5
@3  TST     D5
    BEQ     @4
    _DrawMenuBar
@4  RTS
MarqueMode1
    MOVE.L  HandleMMode(A5),-(SP)
    MOVE    #mReception,-(SP)
    MOVE.B  #255,-(SP)
    _CheckItem
    MOVE.L  HandleMMode(A5),-(SP)
    MOVE    #mEnvoi,-(SP)
    CLR     -(SP)
    _CheckItem
    RTS
MarqueMode2
    MOVE.L  HandleMMode(A5),-(SP)
    MOVE    #mReception,-(SP)
    CLR     -(SP)
    _CheckItem
    MOVE.L  HandleMMode(A5),-(SP)
    MOVE    #mEnvoi,-(SP)
    MOVE.B  #255,-(SP)
    _CheckItem
    RTS
ContenuCommun
    MOVEA.L (A5),A0
    PEA     gray(A0)
    _PenPat
    PEA     RectangleVolume
    _FrameRect
    PEA     RectangleStatus.
    _FrameRect
    MOVE.L  #$006E00C8,-(SP)
    _MoveTo
    MOVE.L  #$006E01CD,-(SP)
    _LineTo
    MOVE.L  #$002000D0,-(SP)
    _MoveTo
    MOVE.L  #$002001C6,-(SP)
    _LineTo
    _PenNormal
    PEA     RectangleNiveau
    _FrameRect
    MOVE.L  #$0059014D,-(SP)
    _MoveTo
    MOVE.L  #$0061014D,-(SP)
    _LineTo
    MOVE.L  #$005901CD,-(SP)
    _MoveTo
    MOVE.L  #$006101CD,-(SP)
    _LineTo
    MOVE.L  #$0063014D,-(SP)
    _MoveTo
    MOVE.L  #$006301CD,-(SP)
    _LineTo
    MOVE    #monaco,-(SP)
    _TextFont
    MOVE    #9,-(SP)
    _TextSize
    MOVE.L  #$0060014F,-(SP)
    _MoveTo
    PEA     ChaineZero
    _DrawString
    MOVE.L  #$006001B5,-(SP)
    _MoveTo
    PEA     ChaineCent
    _DrawString
    MOVE    #sysFont,-(SP)
    _TextFont
    MOVE    #12,-(SP)
    _TextSize
    RTS
AfficheContenuF2
    MOVE.L  #$00280010,-(SP)
    _MoveTo
    PEA     NomFichier(A5)
    _DrawString
    MOVE.L  #$00480010,-(SP)
    _MoveTo
    PEA     ChaineEmission2
    _DrawString
    PEA     TypeFichier+1(A5)
    _DrawString
    MOVE    #' /',-(SP)
    _DrawChar
    PEA     CreateurFichier+1(A5)
    _DrawString
    MOVE.L  #$00580010,-(SP)
    _MoveTo
    PEA     NombreKoENvoi(A5)
    _DrawString
    PEA     ChaineEmission3
    _DrawString
    PEA     NombreOctets(A5)
    _DrawString
    PEA     ChaineEmission4
    _DrawString
    MOVE.L  #$00680040,-(SP)
    _MoveTo
    PEA     ChaineDate(A5)
    _DrawString
    RTS
OuvreSerie
    PEA     TamponIOSort(A5)
    PEA     Sort
    BSR     OuvreS
    PEA     TamponIOEntre(A5)
    PEA     Entre
    BSR     OuvreS
    RTS
OuvreS
    LINK    A6,#0
    MOVEA.L 12(A6),A4
    MOVEA.L A4,A0
    MOVE.L  8(A6),ioFileName(A0)
    CLR.B   ioPermssn(A0)
    _Open
    BNE.S   @1
    MOVEA.L A4,A0
    MOVE    #8,csCode(A0)
    MOVE    #94+1024+16384+12288,csParam(A0)
    _Control,Immed
    BNE.S   @1
    MOVE.L  #800,D0
    _NewPtr,clear
    BNE.S   @1
    MOVE.L  A0,A1
    MOVEA.L A4,A0
    MOVE    #9,csCode(A0)
    MOVE.L  A1,csParam(A0)
    MOVE    #800,csParam+4(A0)
    _Control,Immed
    BEQ.S   @2
@1  MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@2  UNLK    A6
    MOVE.L  (SP),8(SP)
    ADDQ.L  #8,SP
    RTS
RectFenetre1            DC      50,8,169,478
RectFenetre2            DC      200,8,319,478
Rectanglelimite         DC      28,4,338,508
RectangleVolume         DC      8,8,111,191
RectangleBouton1        DC      70,56,88,184
RectangleBouton2        DC      89,56,107,184
RectangleBouton3        DC      89,200,107,328
RectangleIcone          DC      72,16,104,48
RectangleEfface         DC      29,16,58,184
RectangleStatus         DC      8,200,78,462
RectangleEffaceFichier  DC      27,16,43,189
RectangleEffaceInfo1    DC      61,48,77,189
RectangleEffaceInfo2    DC      77,16,93,189
RectangleEffaceInfo3    DC      93,64,109,189
Titre1                  DC.B    9,'Réception'
Titre2                  DC.B    8,'Émission',0
ChaineReception1        DC.B    23,'Réception
                                sur le volume'
ChaineReception2        DC.B    10,'Ko libres)',0
ChaineEmission1         DC.B    19,'Émission du
                                fichier'
ChaineEmission2         DC.B    5,'Type '
ChaineEmission3         DC.B    3,'Ko/'
ChaineEmission4         DC.B    7,' octets'
ChaineEmission5         DC.B    8,'créé le ',0
ChaineEtatR             DC.B    20,'État de la
                                réception',0
ChaineEtatE             DC.B    18,'État de l'
                                'émission',0
ChaineZero              DC.B    2,'0%',0
ChaineCent              DC.B    4,'100%',0
TitreBouton1            DC.B    7,'Lecteur'
TitreBouton2            DC.B    7,'Ejecter'
TitreBouton3            DC.B    17,'Attente des
                                blocs'
TitreBouton1F2          DC.B    15,'Envoi des
                                blocs'
TitreBoutonAnnuler      DC.B    7,'Annuler'
TypeTEXT                DC.B    'TEXT'
CreateurWrite           DC.B    'MACA'
CreateurEdit            DC.B    'EDIT'
Createur                DS.L    1
Dialogue                DS.B    DWindLen
NomAccessoire           DS.B    32
EnregEvents             DS.B    20
HandleMPomme            DS.L    1
HandleMFichier          DS.L    1
HandleMEdition          DS.L    1
HandleMMode             DS.L    1
Article                 DS      1
SauvePort               DS.L    1
PointeurFenetre1        DS.L    1
PointeurFenetre2        DS.L    1
DriverCourant           DS      1
HandleControle          DS.L    1
HandleControle1         DS.L    1
HandleControle2         DS.L    1
HandleControle3         DS.L    1
HandleControle1F2       DS.L    1
NomVolume               DS.B    vcbMaxNam+1
NomFichier              DS.B    64
NomFichierRecu          DS.B    64
NomFichierModif         DS.B    64
NombreK                 DS      5
VolumeCourant           DS      1
DrapeauVV               DS      1
DefautEjecte            DS      1
TamponIO                DS.B    ioFQElSize
TamponTemporaire        DS.B    ioFQElSize
TamponTemporaireF       DS.B    ioFQElSize
DrMenuFichier           DS.B    1
DrMenuEdition           DS.B    1
DrMenuMode              DS.B    1
DrBoutonLecteur         DS.B    1
PasDePlace              DS.B    1
DrChoixFichier          DS.B    1
ReponseGetFile          DS.B    72
NombreKoEnvoi           DS.B    6
CreateurFichier         DS.B    6
NombreOctets            DS.B    10
TypeFichier             DS.B    6
ChaineDate              DS.B    18
```

```
Entre                   DC.B    4,'.AIn',0
Sort                    DC.B    5,'.AOut'
TamponIOEntre           DS.B    ioFQElSize
TamponIOSort            DS.B    ioFQElSize
TamponSortie            DS.B    700
TamponEntre             DS.B    700
RectEffMess1            DC      36,208,60,456
RectEffMess2            DC      60,208,76,456
RectEffMess12           DC      36,208,76,456
Code                    DS      1
NombreOctetsIO          DS.L    1
NumeroBloc              DS      1
NombreKONum             DS.L    1
NombreBlocRecu          DS.L    1
NombreOctetRecu         DS.L    1
NumeroBlocLettre        DS.B    6
CodeConnexion           DC.B    27,57,104,0
CodeDeconnexion         DC.B    27,57,103,0
CodeOpposition          DC.B    27,57,111,0
RectangleNiveauRB       DS      4
RectangleNiveau         DC      97,333,107,462
RectangleEffaceNiveau   DC      100,334,106,461
RectangleEffaceNiveau2  DC      98,334,99,461
NumeroRef               DS      1
FichierTexte2           DS      1
NOctets                 DS.L    1
DrAbandon               DS      1
NombreBlocsD            DS      1
NombreBlocsR            DS      1
NombreBlocsT            DS      1
NombreBlocEmis          DS      1
CompteurBlocs           DS      1
CompteurEssais          DS      1
TableEquival            DC.B    35,'£'
                        DC.B    64,'à'
                        DC.B    91,'°'
                        DC.B    92,'ç'
                        DC.B    93,'§'
                        DC.B    123,'é'
                        DC.B    124,'ù'
                        DC.B    125,'è'
                        DC.B    126,'¨'
CompteurTiming          DS.L    1
NombreBD                DS      1
NombreBR                DS      1

C1  DC.B    28,'Attente du bloc de contrôle…'
C2  DC.B    30,'Réception du bloc de contrôle…'
C18 DC.B    29,'Émission du bloc de contrôle…'
C15 DC.B    34,'Annulation demandée par
            l''émetteur'
C19 DC.B    26,'Annulation demandée par le'
C27 DC.B    9,'récepteur'
C23 DC.B    28,'3" pour prévenir l''émetteur…'
C16 DC.B    32,'Pas assez de place sur ce volume'
C3  DC.B    21,'Réception du fichier '
C4  DC.B    18,' blocs) depuis un '
C5  DC.B    16,'Attente du bloc '
C6  DC.B    18,'Réception du bloc '
C20 DC.B    17,'Émission du bloc '
C7  DC.B    30,'Il nous faut, hélas, renoncer.'
C8  DC.B    22,'Transmission terminée.'
C9  DC.B    9,'Macintosh'
C10 DC.B    8,'Apple ]['
C11 DC.B    2,'. '
C12 DC.B    3,'IBM'
C13 DC.B    13,'Serveur Pom''s'
C14 DC.B    19,'Annulation demandée'
C17 DC.B    26,'Liaison avec le récepteur…'
C21 DC.B    24,'Liaison avec l''émetteur…'
C22 DC.B    9,'ème essai'
C24 DC.B    20,'Réémission demandée.'
C25 DC.B    22,'Confirmation en cours…'
C26 DC.B    20,'Annulation en cours…'
C28 DC.B    15,' (automatique).'
C29 DC.B    11,' (demandé).'
C30 DC.B    17,' bloc) depuis un '
CMF DC.B    '.INCOMPLET'
    .ALIGN  2
```

Fichier 'InterPom's/2.Asm'

```
Reception
    BSR     Montre
    CLR     NumeroBloc(A5)
    MOVE.L  #$0064014E,RectangleNiveauRB+top(A5)
    MOVE    #106,RectangleNiveauRB+bottom(A5)
    PEA     CodeConnexion
    BSR     CodeMinitel
    MOVE.L  HandleControle3(A5),-(SP)
    PEA     TitreBoutonAnnuler
    _SetCTitle
    PEA     TamponIOEntre(A5)
    BSR     VideBuffer
AttenteBlocControle
@1  BSR     NACKx10
    MOVE    Code(A5),D0
    CMPI    #SOH,D0
    BEQ.S   LectureBlocControle
    CMPI    #CAN,D0
    BNE.S   @2
    BSR     AnnulationEmetteur
    BRA     FinReception
@2  CMPI    #CAN_U,D0
    BNE.S   @3
    BSR     AnnulationUtilisateur
    BRA     FinReception
@3  CMPI    #CAN_A,D0
    BNE.S   @4
    BSR     AnnulationAbandon
    BRA     FinReception
@4  BSR     CodeDifferent
    BRA.S   @1
LectureBlocControle
    BSR     BlocControle
    MOVE    Code(A5),D0
    BEQ.S   ContenuHeader
    CMPI    #CAN_A,D0
    BNE.S   @1
    BSR     AnnulationAbandon
    BRA     FinReception
@1  CMPI    #NACK,D0
    BNE.S   AttenteBlocControle
    BSR     CodeDifferent
    BRA     AttenteBlocControle
ContenuHeader
    MOVE.B  TamponEntre+TypeEmetteur(A5),D0
    BNE.S   @5
    BSR     ReceptionMac
    BRA.S   FinReception
@5  SUBQ.B  #1,D0
    BNE.S   @6
    BSR     ReceptionApple
    BRA.S   FinReception
@6  SUBQ.B  #1,D0
    BNE.S   @7
    BSR     ReceptionIBM
    BRA.S   FinReception
@7  BSR     ReceptionServeur
FinReception
    PEA     CodeDeconnexion
    BSR     CodeMinitel
    MOVE.L  HandleControle3(A5),-(SP)
    PEA     TitreBouton3
    _SetCTitle
    _InitCursor
    PEA     RectEffMess12
    _EraseRect
    PEA     RectangleEffaceNiveau
    _EraseRect
    PEA     RectangleEffaceNiveau2
    _EraseRect
    MOVE.L  vcbQHdr+2,A3
    MOVE    VolumeCourant(A5),D5
@2  CMP     vcbVRefNum(A3),D5
    BEQ.S   @1
    MOVE.L  (A3),A3
    BRA.S   @2
@1  BSR     VolumeSuivant2
    BSR     AffNom
    RTS
CodeDifferent
    PEA     C24
    BSR     AfficheM2
    PEA     RectangleEffaceNiveau2
    _EraseRect
    BSR     Attente6
    PEA     RectEffMess2
    _EraseRect
    RTS
ReceptionApple
    MOVE.B  TamponEntre+TypeTexteApple(A5),D0
    CMPI.B  #4,D0
    BEQ.S   @5
    CMPI.B  #$19,D0
    BEQ.S   @5
    CMPI.B  #$1A,D0
    BEQ.S   @5
    CMPI.B  #$1B,D0
    BNE.S   @6
@5  ST      FichierTexte2(A5)
    MOVE.L  CreateurWrite,Createur(A5)
    BRA.S   @7
@6  SF      FichierTexte2(A5)
    MOVE.L  CreateurEdit,Createur(A5)
@7  MOVE.L  NombreBlocRecu(A5),D0
    LSR.L   #1,D0
    CMP.L   NombreKONum(A5),D0
    BMI.S   @1
    BSR     Beep7
    PEA     C16
    BSR     AfficheM1
    BSR     Tempo
    RTS
@1  BSR     NouveauFichier
    BEQ.S   @15
    RTS
@15 MOVEQ   #0,D0
    BSR     OuvertureFichier
    BEQ     @4
    RTS
@4  BSR     ChangeNumeroBloc
@19 BSR     AfficheAttenteBloc
    MOVE    #ACK,-(SP)
    BSR     EnvoiCode
@2  BSR     TesteTimeOut
    BEQ.S   @28
    BSR     AnnulationAbandon
    BRA     AbandonReceptionApple
@28 BSR     TesteAnnulerR
    TST     Code(A5)
    BEQ.S   @20
    BSR     AnnulationUtilisateur
    BRA     AbandonReceptionApple
@20 BSR     AttenteLiaisonEmetteur
    TST     Code(A5)
    BEQ.S   @2
    BSR     GetTimer
    MOVE    Code(A5),D0
    CMPI    #SOH,D0
    BEQ.S   @18
    CMPI    #CAN,D0
    BNE.S   @22
    BSR     AnnulationEmetteur
    BRA     AbandonReceptionApple
@22 CMPI    #EOT,D0
    BEQ     FinReceptionApple
@24 BSR     CodeDifferent
@27 BSR     AfficheAttenteBloc
    MOVE    #NACK,-(SP)
    BSR     EnvoiCode
    BRA.S   @2
@18 PEA     C6
    BSR     AfficheM1
    PEA     NumeroBlocLettre(A5)
    _DrawString
    BSR     LireBloc
    MOVE    Code(A5),D0
    BEQ.S   @16
    CMPI    #CAN_A,D0
    BNE.S   @23
    BSR     AnnulationAbandon
    BRA     AbandonReceptionApple
@23 BRA.S   @24
@16 BSR     VerificationBloc
    MOVE    Code(A5),D0
    BEQ.S   @3
    CMPI    #DIF,D0
    BNE.S   @25
    BSR     AfficheAttenteBloc
    MOVE    #ACK,-(SP)
    BSR     EnvoiCode
    BRA     @2
@25 CMPI    #CAN_A,D0
    BNE     @27
    BSR     AnnulationAbandon
    BRA     AbandonReceptionApple
@3  BSR     DecodeBloc
    TST.B   FichierTexte2(A5)
    BEQ.S   @8
    BSR     ModifsFichierTexteA
@8  BSR     SPNiveauReception
    BSR     EcritBlocDisque
    BEQ     @4
    RTS
FinReceptionApple
    MOVE    #ACK,-(SP)
    BSR     EnvoiCode
    PEA     RectEffMess2
    _EraseRect
    PEA     C8
    BSR     AfficheM1
    LEA     TamponIO(A5),A0
    MOVE.L  NombreOctetRecu(A5),ioLEOF(A0)
    _SetEOF
@1  SF      DrAbandon(A5)
    BRA.S   ARA2
AbandonReceptionApple
    ST      DrAbandon(A5)
ARA2
    BSR     FermetureFichier
    BSR     LireInfoFichier
    BEQ.S   @3
    RTS
@3  LEA     TamponIO(A5),A0
    MOVE.L  #'TEXT',ioFlUsrWds+fdType(A0)
    MOVE.L  Createur(A5),ioFlUsrWds+fdCreator(A0)
    _SetFileInfo
@2  TST.B   DrAbandon(A5)
    BEQ.S   @1
    BSR     RenommerFichier
@1  BSR     MiseJourVolume
    BSR     Tempo
RetourReceptionApple
    RTS
AfficheAttenteBloc
    PEA     RectEffMess2
    _EraseRect
    PEA     C5
    BSR     AfficheM1
    PEA     NumeroBlocLettre(A5)
    _DrawString
    RTS
Attente6
    MOVE.L  #480,D3
    SUBQ    #4,SP
    _TickCount
    MOVE.L  (SP)+,D4
    ADD.L   D3,D4
@1  _SystemTask
    SUBQ    #4,SP
    _TickCount
    CMP.L   (SP)+,D4
    BHI.S   @1
    PEA     TamponIOEntre(A5)
    BSR     VideBuffer
    RTS
BlocControle
    PEA     RectEffMess12
    _EraseRect
    PEA     C2
```

```
    BSR     AfficheM1
    BSR     LireBloc
    TST     Code(A5)
    BEQ.S   @9
    RTS
@9  BSR     VerificationBloc
    TST     Code(A5)
    BEQ.S   @1
    RTS
@1  BSR     DecodeBloc
    BSR     DeplaceNFR
    CMPI.B  #AppleII,TamponEntre+TypeEmetteur(A5)
    BNE.S   @13
    BSR     MinusMajus
@13 BSR     LongueurNomFichierRecu
    MOVE    #7,-(SP)
    _SysBeep
    PEA     C3
    BSR     AfficheM1
    PEA     NomFichierRecu(A5)
    _DrawString
    CMPI.B  #AppleII,TamponEntre+TypeEmetteur(A5)
    BEQ.S   @10
    MOVE    TamponEntre+NBlocsTotal(A5),D0
    BRA.S   @7
@10 BSR     SPNombreOctets
    MOVE.L  NombreOctetRecu(A5),D0
    MOVE    #512,D1
    DIVU    D1,D0
    SWAP    D0
    TST     D0
    BEQ.S   @8
    SWAP    D0
    ADDQ    #1,D0
    BRA.S   @7
@8  SWAP    D0
@7  ANDI.L  #$FFFF,D0
    MOVE.L  D0,NombreBlocRecu(A5)
    LEA     scratch20,A0
    CLR     -(SP)
    _Pack7
    MOVE.L  #$004800D0,-(SP)
    _MoveTo
    MOVE    #' (',-(SP)
    _DrawChar
    PEA     scratch20
    _DrawString
    CMPI    #1,NombreBlocRecu+2(A5)
    BGT.S   @11
    PEA     C30
    BRA.S   @12
@11 PEA     C4
@12 _DrawString
    MOVE.B  TamponEntre+TypeEmetteur(A5),D0
    BNE.S   @2
    LEA     C9,A0
    BRA.S   @5
@2  SUBQ.B  #1,D0
    BNE.S   @3
    LEA     C10,A0
    BRA.S   @5
@3  SUBQ.B  #1,D0
    BNE.S   @4
    LEA     C12,A0
    BRA.S   @5
@4  LEA     C13,A0
@5  MOVE.L  A0,-(SP)
    _DrawString
    BSR     Tempo
    CLR     Code(A5)
    RTS

DeplaceNFR
    MOVEQ   #0,D0
    MOVE.B  TamponEntre+NomFichierApple(A5),D0
    ADDQ    #1,D0
    LEA     TamponEntre+NomFichierApple(A5),A0
    LEA     NomFichierRecu(A5),A1
    _BlockMove
    RTS

MinusMajus
    LEA     NomFichierRecu(A5),A0
    MOVE.B  (A0)+,D0
    EXT     D0
    ADDQ.L  #1,A0
    SUBQ    #2,D0
@1  MOVE.B  (A0),D1
    CMPI.B  #'A',D1
    BMI.S   @2
    CMPI.B  #'Z',D1
    BGT.S   @2
    ORI.B   #32,(A0)+
    BRA.S   @3
@2  ADDQ.L  #1,A0
@3  DBRA    D0,@1
    RTS

LireBloc
    MOVE.B  #1,TamponEntre(A5)
@2  LEA     TamponIOEntre(A5),A0
    MOVE    #2,csCode(A0)
    _Status,Immed
    BEQ.S   @6
@7  MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@6  LEA     TamponIOEntre(A5),A0
    TST.L   csParam(A0)
    BEQ.S   @2
    BSR     GetTimer
    LEA     TamponIOEntre(A5),A0
    MOVE.L  #1,ioReqCount(A0)
    LEA     scratch8,A3
    MOVE.L  A3,ioBuffer(A0)
    MOVE    #fsFromStart,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Read
    BNE     @7
    MOVE.B  (A3),TamponEntre+1(A5)
    PEA     RectangleEffaceNiveau2
    _EraseRect
    CLR.L   NOctets(A5)
@1  LEA     TamponIOEntre(A5),A0
    MOVE    #2,csCode(A0)
    _Status,Immed
    BNE     @7
    LEA     TamponIOEntre(A5),A0
    MOVE.L  csParam(A0),D5
    CMP.L   NOctets(A5),D5
    BNE.S   @4
    BSR     VerifTiming
    TST     Code(A5)
    BEQ.S   @3
    BRA.S   @5
@4  SUBQ.L  #4,SP
    _TickCount
    MOVE.L  (SP)+,CompteurTiming(A5)
    MOVE.L  D5,NOctets(A5)
    LSR     #4,D5
    MULU    #3,D5
    ADD     #334,D5
    MOVE.L  #$0062014E,-(SP)
    _MoveTo
    MOVE    D5,-(SP)
    MOVE    #$0062,-(SP)
    _LineTo
@3  CMPI.L  #687,D5
    BMI.S   @1
    MOVE.L  #687,ioReqCount(A0)
    LEA     TamponEntre+2(A5),A1
    MOVE.L  A1,ioBuffer(A0)
    MOVE    #fsFromStart,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Read
    BNE     @7
    CLR     Code(A5)
@5  RTS

VerifTiming
    SUBQ.L  #4,SP
    _TickCount
    MOVE.L  (SP)+,D0
    SUB.L   CompteurTiming(A5),D0
    CMPI.L  #30,D0
    BMI.S   @3
    MOVE    #NACK,Code(A5)
    BRA.S   @2
@3  CLR     Code(A5)
@2  RTS

NACKx10
    PEA     C21
    BSR     AfficheM1
    MOVEQ   #2,D5
@3  BSR     UneSeconde
    DBRA    D5,@3
    PEA     TamponIOEntre(A5)
    BSR     VideBuffer
    MOVEQ   #1,D5
@1  MOVE    D5,-(SP)
    BSR     AfficheNumEssai
    MOVE    #NACK,-(SP)
    BSR     EnvoiCode
    BSR     DixSecondes
    TST     Code(A5)
    BNE.S   @2
    ADDQ    #1,D5
    CMPI    #21,D5
    BNE.S   @1
    MOVE    #7,-(SP)
    _SysBeep
    MOVE    #CAN_A,Code(A5)
@2  RTS

DixSecondes
    MOVE.L  #300,D3
    SUBQ    #4,SP
    _TickCount
    MOVE.L  (SP)+,D4
    ADD.L   D3,D4
@1  BSR     TesteAnnulerR
    TST     Code(A5)
    BEQ.S   @4
    MOVE    #CAN_U,Code(A5)
    BRA.S   @3
@4  BSR     AttenteLiaisonEmetteur
    TST     Code(A5)
    BNE.S   @3
@2  SUBQ    #4,SP
    _TickCount
    CMP.L   (SP)+,D4
    BHI.S   @1
@3  RTS

AnnulationEmetteur
    MOVE    #7,-(SP)
    _SysBeep
    PEA     C15
    BSR     AfficheM1
    PEA     C25
    BSR     AfficheM2
    BSR     TroisACK
    RTS

AnnulationUtilisateur
    PEA     C26
    BSR     AfficheM1
    PEA     RectEffMess2
    _EraseRect
    BSR     TroisCANCEL
    RTS

AnnulationAbandon
    MOVE    #7,-(SP)
    _SysBeep
    PEA     C7
    BSR     AfficheM1
    PEA     C23
    BSR     AfficheM2
    BSR     TroisCANCEL
    RTS

TroisCANCEL
    MOVEQ   #2,D5
@1  MOVE    #CAN,-(SP)
    BSR     EnvoiCode
    BSR     UneSeconde
    DBRA    D5,@1
    RTS

TroisACK
    MOVEQ   #2,D5
@1  MOVE    #ACK,-(SP)
    BSR     EnvoiCode
    BSR     UneSeconde
    DBRA    D5,@1
    RTS

TroisEOT
    MOVEQ   #2,D5
@1  MOVE    #EOT,-(SP)
    BSR     EnvoiCode
    BSR     UneSeconde
    DBRA    D5,@1
    RTS

EnvoiCode
    LINK    A6,#0
    LEA     TamponIOSort(A5),A0
    MOVEQ   #1,D0
    MOVE.L  D0,ioReqCount(A0)
    LEA     TamponSortie(A5),A1
    MOVE.L  A1,ioBuffer(A0)
    MOVE.B  9(A6),TamponSortie(A5)
    MOVE    #fsFromStart,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Write
    BEQ.S   @1
    MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@1  UNLK    A6
    MOVE.L  (SP),2(SP)
    ADDQ.L  #2,SP
    RTS

UneSeconde
    MOVE.L  #60,D3
    SUBQ    #4,SP
    _TickCount
    MOVE.L  (SP)+,D4
    ADD.L   D3,D4
@1  _SystemTask
    SUBQ    #4,SP
    _TickCount
    CMP.L   (SP)+,D4
    BHI.S   @1
    RTS

Tempo
    MOVE.L  #180,D3
    SUBQ    #4,SP
    _TickCount
    MOVE.L  (SP)+,D4
    ADD.L   D3,D4
@1  SUBQ    #4,SP
    _TickCount
    CMP.L   (SP)+,D4
    BHI.S   @1
    RTS

CodeMinitel
    LINK    A6,#0
    LEA     TamponIOSort(A5),A0
    MOVEQ   #3,D0
    MOVE.L  D0,ioReqCount(A0)
    LEA     TamponSortie(A5),A1
    MOVE.L  A1,ioBuffer(A0)
    MOVE.L  8(A6),A1
    MOVE.L  (A1),TamponSortie(A5)
    MOVE    #fsFromStart,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Write
    BEQ.S   @1
    MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@1  UNLK    A6
    MOVE.L  (SP),4(SP)
    ADDQ.L  #4,SP
    RTS

NomExistant
    LEA     NomFichierRecu(A5),A0
    MOVE.B  (A0),D0
```

```
    CMPI.B  #62,D0
    BMI.S   Moins62
    MOVE.B  62(A0),D1
    CMPI.B  #'.',D1
    BEQ.S   DejaModifPlus62
    MOVE.B  #'.',62(A0)
    MOVE.B  #'A',63(A0)
    MOVE.B  #63,(A0)
    RTS
DejaModifPlus62
@1  ADDQ.B  #1,63(A0)
    CMPI.B  #':',63(A0)
    BEQ.S   @1
    RTS
Moins62
    EXT     D0
    MOVE.B  -1(A0,D0),D1
    CMPI.B  #'.',D1
    BEQ.S   DejaModif
    MOVE.B  #'.',1(A0,D0)
    MOVE.B  #'A',2(A0,D0)
    ADDQ.B  #2,(A0)
    RTS
DejaModif
@1  ADDQ.B  #1,0(A0,D0)
    CMPI.B  #':',0(A0,D0)
    BEQ.S   @1
    RTS

AfficheM1
    LINK    A6,#0
    PEA     RectEffMess1
    _EraseRect
    MOVE.L  #$003800D0,-(SP)
    _MoveTo
    MOVE.L  8(A6),-(SP)
    _DrawString
    UNLK    A6
    MOVE.L  (SP),4(SP)
    ADDQ.L  #4,SP
    RTS
AfficheM2
    LINK    A6,#0
    PEA     RectEffMess2
    _EraseRect
    MOVE.L  #$004800D0,-(SP)
    _MoveTo
    MOVE.L  8(A6),-(SP)
    _DrawString
    UNLK    A6
    MOVE.L  (SP),4(SP)
    ADDQ.L  #4,SP
    RTS

AfficheNumEssai
    LINK    A6,#0
    MOVE    8(A6),D5
    CMPI    #1,D5
    BEQ.S   @1
    PEA     RectEffMess2
    _EraseRect
    MOVE.L  #$004800D0,-(SP)
    _MoveTo
    MOVE    D5,D0
    EXT.L   D0
    LEA     scratch8,A0
    CLR     -(SP)
    _Pack7
    PEA     scratch8
    _DrawString
    PEA     C22
    _DrawString
@1  UNLK    A6
    MOVE.L  (SP),2(SP)
    ADDQ.L  #2,SP
    RTS

AttenteLiaisonEmetteur
    LEA     TamponIOEntre(A5),A0
    MOVE    #2,csCode(A0)
    _Status,Immed
    BEQ.S   @4
@5  MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@4  LEA     TamponIOEntre(A5),A0
    TST.L   csParam(A0)
    BEQ.S   @2
    MOVE.L  #1,ioReqCount(A0)
    LEA     scratch8,A3
    MOVE.L  A3,ioBuffer(A0)
    MOVE    #fsFromStart,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Read
    BNE.S   @5
    MOVE.B  (A3),D0
    ANDI.B  #$7F,D0
    EXT     D0
    MOVE    D0,Code(A5)
    BRA.S   @3
@2  CLR     Code(A5)
@3  RTS
ModifsFichierTexteA
    MOVEQ   #127,D0
    LEA     TamponEntre+TypeEmetteur(A5),A0
@1  ANDI.L  #$7F7F7F7F,(A0)+
    DBRA    D0,@1
    LEA     TableEquival,A2
    MOVEQ   #16,D2
@2  MOVE.B  0(A2,D2),D0
    MOVE    #511,D1
    LEA     TamponEntre+TypeEmetteur(A5),A0
@4  CMP.B   (A0),D0
    BEQ.S   @5
    ADDQ.L  #1,A0
    BRA.S   @6
@5  MOVE.B  1(A2,D2),(A0)+
@6  DBRA    D1,@4
    SUBQ    #2,D2
    BGE.S   @2
    RTS
```

Fichier 'InterPom's/3.Asm'

```
Emission
    BSR     Montre
    CLR     NumeroBloc(A5)
    CLR     NombreBlocEmis(A5)
    MOVEA.L (A5),A0
    PEA     ltgray(A0)
    _PenPat
    PEA     RectangleEffaceNiveau
    _PaintRect
    _PenNormal
    MOVE    #$0064,RectangleNiveauRB+top(A5)
    MOVE.L  #$006A01CD,RectangleNiveauRB+bottom(A5)
    PEA     TamponIOEntre(A5)
    BSR     VideBuffer
    PEA     CodeOpposition
    BSR     CodeMinitel
    PEA     CodeConnexion
    BSR     CodeMinitel
    MOVE.L  HandleControle1F2(A5),-(SP)
    PEA     TitreBoutonAnnuler
    _SetCTitle
    LEA     TamponTemporaire(A5),A0
    LEA     TamponSortie+TypeEmetteur(A5),A1
    MOVEQ   #ioFQElSize,D0
    _BlockMove
    MOVEQ   #0,D0
    MOVE.B  ReponseGetFile+rName(A5),D0
    ADDQ    #1,D0
    LEA     ReponseGetFile+rName(A5),A0
    LEA     TamponSortie+NomFichierApple(A5),A1
    _BlockMove
    LEA     TamponSortie(A5),A0
    MOVE.L  #$01C0FFFF,DebutBloc(A0)
    CLR.B   TypeEmetteur(A0)
    CMPI.L  #'TEXT',TypeEmetteur+ioFlUsrWds+
            fdType(A0)
    BNE.S   @1
    MOVE.L  #$4000000,TypeTexteApple(A0
    BRA.S   @2
@1  CLR.B   TypeTexteApple(A0)
@2  MOVE.B  TypeEmetteur+ioFlLgLen+1(A0),
            LongLog_Apple+2(A0)
    MOVE.B  TypeEmetteur+ioFlLgLen+2(A0),
            LongLog_Apple+1(A0)
    MOVE.B  TypeEmetteur+ioFlLgLen+3(A0),
            LongLog_Apple(A0)
    MOVE    #LongueurBloc,D1
    MOVE.L  TypeEmetteur+ioFlLgLen(A0),D0
    DIVU    D1,D0
    SWAP    D0
    TST     D0
    BEQ.S   @3
    SWAP    D0
    ADDQ    #1,D0
    BRA.S   @4
@3  SWAP    D0
@4  MOVE    D0,NBlocsData(A0)
    MOVE.L  TypeEmetteur+ioFlRLgLen(A0),D0
    DIVU    D1,D0
    SWAP    D0
    TST     D0
    BEQ.S   @5
    SWAP    D0
    ADDQ    #1,D0
    BRA.S   @6
@5  SWAP    D0
@6  MOVE    D0,NBlocsResource(A0)
    ADD     NBlocsData(A0),D0
    MOVE    D0,NBlocsTotal(A0)
    MOVE    D0,NombreBlocsT(A5)
    MOVE.B  NBlocsTotal+1(A0),NBlocsApple(A0)
    MOVE.B  NBlocsTotal(A0),NBlocsApple+1(A0)
    MOVE    NBlocsData(A0),NombreBlocsD(A5)
    MOVE    NBlocsResource(A0),NombreBlocsR(A5)
    BSR     Date_Heure
    BSR     CodeBloc
    BSR     CheckSum
    PEA     C17
    BSR     AfficheM1
    BSR     AttenteLiaison
    MOVE    Code(A5),D0
    BGE.S   @7
    BSR     AbandonEmission
    BRA     SortieEmission
@7  CMPI    #CAN_U,D0
    BNE.S   @9
    BSR     ArretEmission
    BRA     SortieEmission
@9  CMPI    #CAN,D0
    BNE.S   @8
    BSR     AnnulationRecepteur
    BRA     SortieEmission
@8  PEA     C18
    BSR     AfficheM1
    MOVE    #1,CompteurEssais(A5)
ETQ11
    BSR     Envoilbloc
@8  BSR     AttenteBlocOK
    TST     Code(A5)
    BNE.S   @9
    ADDQ    #1,CompteurEssais(A5)
    CMPI    #NombreMaxEssai+1,CompteurEssais(A5)
    BEQ.S   @1
    MOVE    CompteurEssais(A5),-(SP)
    BSR     AfficheNumEssai
    PEA     C28
    _DrawString
    BRA     ETQ11
@1  BSR     AbandonEmission
    BRA     SortieEmission
@9  MOVE    Code(A5),D0
    CMPI    #CAN,D0
    BNE.S   ETQ10
    BSR     AnnulationRecepteur
    BRA     SortieEmission
ETQ10
    CMPI    #CAN_U,D0
    BNE.S   @2
    BSR     ArretEmission
    BRA     SortieEmission
@2  CMPI    #NACK,D0
    BNE.S   @1
    ADDQ    #1,CompteurEssais(A5)
    MOVE    CompteurEssais(A5),-(SP)
    BSR     AfficheNumEssai
    PEA     C29
    _DrawString
    BRA     ETQ11
@1  TST     NombreBlocsD(A5)
    BEQ     BlocsRessource
BlocsDonnees
    MOVE    NombreBlocsD(A5),CompteurBlocs(A5)
    LEA     TamponIO(A5),A0
    LEA     ReponseGetFile+rName(A5),A1
    MOVE.L  A1,ioFileName(A0)
    MOVE    ReponseGetFile+rVolume(A5),
            ioVRefNum(A0)
    CLR.L   ioOwnBuf(A0)
    MOVE.B  #fsRdPerm,ioPermssn(A0)
    _Open
    BEQ.S   @1
    CLR     -(SP)
    BSR     Erreur
    BRA     SortieEmission
@1  BSR     SpEmission
    BNE     SortieEmission
    BSR     FermetureFichier
BlocsRessource
    TST     NombreBlocsR(A5)
    BEQ     SortieEmission2
    MOVE    NombreBlocsR(A5),CompteurBlocs(A5)
    LEA     TamponIO(A5),A0
    LEA     ReponseGetFile+rName(A5),A1
    MOVE.L  A1,ioFileName(A0)
    MOVE    ReponseGetFile+rVolume(A5),
            ioVRefNum(A0)
    CLR.L   ioOwnBuf(A0)
    MOVE.B  #fsRdPerm,ioPermssn(A0)
    _OpenRF
    BEQ.S   @1
    CLR     -(SP)
    BSR     Erreur
    BRA     SortieEmission
@1  BSR     SpEmission
    BNE     SortieEmission
FinTransmission
    BSR     FermetureFichier
SortieEmission2
    PEA     RectEffMess2
    _EraseRect
    PEA     C8
    BSR     AfficheM1
    BSR     TroisEOT
SortieEmission
    PEA     CodeDeconnexion
    BSR     CodeMinitel
    MOVE.L  HandleControle1F2(A5),-(SP)
    PEA     TitreBouton1F2
    _SetCTitle
    _InitCursor
    PEA     RectEffMess12
    _EraseRect
    PEA     RectangleEffaceNiveau
    _EraseRect
    PEA     RectangleEffaceNiveau2
    _EraseRect
    RTS
TesteAnnuler
    _SystemTask
    SUBQ    #2,SP
    MOVE    #2,-(SP)
    PEA     EnregEvents(A5)
    _GetNextEvent
    TST     (SP)+
    BEQ.S   RetourTesteAnnulerNon
@1  SUBQ    #2,SP
    MOVE.L  EnregEvents+evtMouse(A5),-(SP)
    PEA     EnregEvents+evtMeta+2(A5)
    _FindWindow
    CMPI    #inContent,(SP)+
```

```
        BNE.S   BeepTesteAnnuler
        MOVEA.L EnregEvents+evtMeta+2(A5),A4
        TST.L   wRefCon(A4)
        BEQ.S   BeepTesteAnnuler
        MOVE.L  A4,-(SP)
        _SetPort
        PEA     EnregEvents+EvtMouse(A5)
        _GlobalToLocal
        SUBQ.L  #2,SP
        MOVE.L  EnregEvents+EvtMouse(A5),-(SP)
        MOVE.L  A4,-(SP)
        PEA     HandleControle(A5)
        _FindControl
        TST     (SP)+
        BEQ.S   BeepTesteAnnuler
        SUBQ.L  #2,SP
        MOVE.L  HandleControle(A5),-(SP)
        MOVE.L  EnregEvents+EvtMouse(A5),-(SP)
        CLR.L   -(SP)
        _TrackControl
        TST     (SP)+
        BEQ.S   BeepTesteAnnuler
        SUBQ.L  #4,SP
        MOVE.L  HandleControle(A5),-(SP)
        _GetCRefCon
        MOVEQ   #3,D0
        CMP.L   (SP)+,D0
        BNE.S   BeepTesteAnnuler
        MOVE    #CAN_U,Code(A5)
        RTS
BeepTesteAnnuler
        BSR     Beep7
RetourTesteAnnulerNon
        CLR     Code(A5)
        RTS
Date_Heure
        MOVE.L  Time,-(SP)
        MOVE.B  #longDate,-(SP)
        PEA     TamponSortie+Date(A5)
        MOVE    #IUDateString,-(SP)
        _Pack6
        MOVE.L  Time,-(SP)
        CLR     -(SP)
        PEA     TamponSortie+Heure(A5)
        MOVE    #IUTimeString,-(SP)
        _Pack6
        LEA     TamponSortie+Date(A5),A0
        MOVE.B  (A0),D0
        EXT     D0
        MOVE.B  #',',1(A0,D0)
        MOVE.B  #' ',2(A0,D0)
        LEA     3(A0,D0),A1
        LEA     TamponSortie+Heure(A5),A0
        MOVE.B  (A0)+,D0
        EXT     D0
        SUBQ    #1,D0
@1      MOVE.B  (A0)+,(A1)+
        DBRA    D0,@1
        MOVE.B  TamponSortie+Date(A5),D0
        ADD.B   TamponSortie+Heure(A5),D0
        ADDQ.B  #3,D0
        MOVE.B  D0,TamponSortie+Date(A5)
        LEA     TamponSortie+Date(A5),A0
        MOVE.B  (A0),D0
        EXT     D0
        MOVE.B  #'.',0(A0,D0)
        RTS
CodeBloc
        LEA     TamponSortie+TypeEmetteur(A5),A3
        MOVE    #170,D5
        MOVE    #LongueurBlocC-3,D4
        MOVE    #LongueurBloc-2,D3
@1      CLR.L   -(SP)
        MOVE.B  0(A3,D3),1(SP)
        MOVE.B  1(A3,D3),2(SP)
        MOVE.B  2(A3,D3),3(SP)
        BSR     Codage
        MOVE.L  (SP)+,0(A3,D4)
        SUBQ    #4,D4
        SUBQ    #3,D3
        DBRA    D5,@1
        RTS
Codage
Nul     SET     8
Octet1  Set     9
Octet2  Set     10
Octet3  Set     11
        LINK    A6,#0
        MOVEM.L D0-D1,-(SP)
        MOVE.B  Octet1(A6),D0
        LSL     #6,D0
        MOVE.L  Nul(A6),D1
        LSR.L   #Nul,D1
        LSR     #4,D1
        MOVE.B  D1,D0
        LSL.L   #Nul,D0
        MOVE    Octet2(A6),D1
        LSR     #6,D1
        MOVE.B  D1,D0
        LSL.L   #Nul,D0
        MOVE.B  Octet3(A6),D0
        ORI.L   #$C0C0C0C0,D0
        MOVE.L  D0,Nul(A6)
        MOVEM.L (SP)+,D0-D1
        UNLK    A6
        RTS
CheckSum
        LEA     TamponSortie+TypeEmetteur(A5),A0
        MOVEQ   #0,D0
        MOVE    #683,D1
@1      MOVE.B  (A0)+,D2
        ANDI.B  #SixFaibles,D2
        ADD.B   D2,D0
        DBRA    D1,@1
        ORI.B   #DeuxForts,D0
        MOVE.B  D0,(A0)
        RTS
AttenteLiaison
        MOVE.L  #3600,D3
        SUBQ    #4,SP
        _TickCount
        MOVE.L  (SP)+,D4
        ADD.L   D3,D4
@1      BSR     TesteAnnuler
        TST     Code(A5)
        BNE.S   @2
        BSR     AttenteLiaisonRecepteur
        TST     Code(A5)
        BNE.S   @2
        SUBQ    #4,SP
        _TickCount
        CMP.L   (SP)+,D4
        BHI.S   @1
        MOVE    #-1,Code(A5)
@2      RTS
AttenteLiaisonRecepteur
        LEA     TamponIOEntre(A5),A0
@1      MOVE    #2,csCode(A0)
        _Status,Immed
        BEQ.S   @4
@6      MOVE    #2,-(SP)
        BSR     Erreur
        _ExitToShell
@4      LEA     TamponIOEntre(A5),A0
        TST.L   csParam(A0)
        BEQ.S   @2
        MOVE.L  #1,ioReqCount(A0)
        LEA     Scratch8,A3
        MOVE.L  A3,ioBuffer(A0)
        MOVE    #fsFromStart,ioPosMode(A0)
        CLR.L   ioPosOffset(A0)
        _Read
        BNE.S   @6
        MOVE.B  (A3),D0
        ANDI    #$7F,D0
        CMPI.B  #CAN,D0
        BNE.S   @3
        MOVE    D0,Code(A5)
        RTS
@3      CMPI.B  #NACK,D0
        BNE.S   @1
        MOVE    D0,Code(A5)
@2      RTS
AttenteConfirmeAnnulation
        LEA     TamponIOEntre(A5),A0
@1      MOVE    #2,csCode(A0)
        _Status,Immed
        BEQ.S   @4
@5      MOVE    #2,-(SP)
        BSR     Erreur
        _ExitToShell
@4      LEA     TamponIOEntre(A5),A0
        TST.L   csParam(A0)
        BEQ.S   @2
        MOVE.L  #1,ioReqCount(A0)
        LEA     scratch8,A3
        MOVE.L  A3,ioBuffer(A0)
        MOVE    #fsFromStart,ioPosMode(A0)
        CLR.L   ioPosOffset(A0)
        _Read
        BNE.S   @5
        MOVE.B  (A3),D0
        ANDI    #$7F,D0
        CMPI.B  #CAN,D0
        BNE.S   @3
        MOVE    D0,Code(A5)
        RTS
@3      CMPI.B  #ACK,D0
        BNE.S   @1
        MOVE    #NACK,Code(A5)
@2      RTS
AttenteBlocOK
        MOVE.L  #540,D3
        SUBQ    #4,SP
        _TickCount
        MOVE.L  (SP)+,D4
        ADD.L   D3,D4
@1      BSR     TesteAnnuler
        TST     Code(A5)
        BNE.S   @2
        BSR     AttenteCarBlocOK
        TST     Code(A5)
        BNE.S   @2
        SUBQ    #4,SP
        _TickCount
        CMP.L   (SP)+,D4
        BHI.S   @1
        CLR     Code(A5)
@2      RTS
AttenteCarBlocOK
        LEA     TamponIOEntre(A5),A0
@1      MOVE    #2,csCode(A0)
        _Status,Immed
        BEQ.S   @5
@6      MOVE    #2,-(SP)
        BSR     Erreur
        _ExitToShell
@5      LEA     TamponIOEntre(A5),A0
        TST.L   csParam(A0)
        BEQ.S   @2
        MOVE.L  #1,ioReqCount(A0)
        LEA     scratch8,A3
        MOVE.L  A3,ioBuffer(A0)
        MOVE    #fsFromStart,ioPosMode(A0)
        CLR.L   ioPosOffset(A0)
        _Read
        BNE.S   @6
        MOVE.B  (A3),D0
        ANDI    #$7F,D0
        CMPI.B  #CAN,D0
        BNE.S   @3
        MOVE    #CAN,Code(A5)
        BRA.S   @2
@3      CMPI.B  #ACK,D0
        BNE.S   @4
        MOVE    #ACK,Code(A5)
        BRA.S   @2
@4      CMPI.B  #NACK,D0
        BNE.S   @1
        MOVE    #NACK,Code(A5)
@2      RTS
Envoilbloc
        MOVE.L  #$0062014E,-(SP)
        _MoveTo
        MOVE.L  #$006201CD,-(SP)
        _LineTo
        MOVE.L  #LongueurBlocC+6,NOctets(A5)
        LEA     TamponIOSort(A5),A0
        CLR.L   ioCompletion(A0)
        MOVE.L  #LongueurBlocC+6,ioReqCount(A0)
        LEA     TamponSortie(A5),A1
        MOVE.L  A1,ioBuffer(A0)
        MOVE    #fsFromStart,ioPosMode(A0)
        CLR.L   ioPosOffset(A0)
        _Write,async
        BEQ.S   @1
        MOVE    #2,-(SP)
        BSR     Erreur
        BSR     FermetureErr
        _ExitToShell
@1      LEA     TamponIOSort(A5),A0
        TST     ioResult(A0)
        BNE.S   @2
        PEA     RectangleEffaceNiveau2
        _EraseRect
        RTS
@2      MOVE.L  ioActCount(A0),D5
        CMP.L   NOctets(A5),D5
        BEQ.S   @1
        MOVEA.L (A5),A0
        PEA     white(A0)
        _PenPat
        MOVE.L  D5,NOctets(A5)
        MOVE.L  #LongueurBlocC+6,D0
        SUB     D5,D0
        LSR     #4,D0
        MULU    #3,D0
        ADD     #334,D0
        MOVE    D0,-(SP)
        MOVE    #$0062,-(SP)
        _MoveTo
        MOVE.L  #$006201CC,-(SP)
        _LineTo
        _PenNormal
        BRA.S   @1
SPNiveauEmission
        MOVE.L  #16384,D5
        MOVE    NombreBlocsT(A5),D0
        BNE.S   @1
        MOVEQ   #1,D0
@1      DIVU    D0,D5
        MULU    NombreBlocEmis(A5),D5
        LSR.L   #7,D5
        MOVE    #128,D4
        SUB     D5,D4
        ADDI    #333,D4
        MOVE    D4,RectangleNiveauRB+left(A5)
        PEA     RectangleNiveauRB(A5)
        _EraseRect
        MOVE    D4,-(SP)
        MOVE    #$0064,-(SP)
        _MoveTo
        MOVE    D4,-(SP)
        MOVE    #$006A,-(SP)
        _LineTo
        RTS
AbandonEmission
        BSR     Beep7
        PEA     RectEffMess2
        _EraseRect
        PEA     C7
        BSR     AfficheM1
        BSR     FermetureFichier
        BSR     TroisCANCEL
        RTS
AnnulationRecepteur
        BSR     Beep7
        PEA     C19
        BSR     AfficheM1
        PEA     C27
        BSR     AfficheM2
```

```
    BSR     FermetureFichier
    BSR     TroisACK
    RTS
ArretEmission
    PEA     RectEffMess2
    _EraseRect
    PEA     C26
    BSR     AfficheM2
    BSR     FermetureFichier
    BSR     TroisCANCEL
    RTS

SpEmission
@1  ADDQ    #1,NumeroBloc(A5)
    ADDQ    #1,NombreBlocEmis(A5)
    MOVEQ   #0,D0
    MOVE    NumeroBloc(A5),D0
    LEA     NumeroBlocLettre(A5),A0
    CLR     -(SP)
    _Pack7
    LEA     TamponSortie(A5),A0
    MOVE.B  #SOH,(A0)
    MOVE.B  NumeroBloc+1(A5),DebutBloc+1(A0)
    ORI.B   #DeuxForts,DebutBloc+1(A0)
    MOVE.B  NumeroBloc+1(A5),D0
    EORI.B  #SixFaibles,D0
    ORI.B   #DeuxForts,D0
    MOVE.B  D0,DebutBloc+2(A0)
    MOVE.B  #$FF,DebutBloc+3(A0)
    LEA     TamponIO(A5),A0
    MOVE.L  #LongueurBloc,ioReqCount(A0)
    LEA     TamponSortie+TypeEmetteur(A5),A1
    MOVE.L  A1,ioBuffer(A0)
    MOVE    #fsAtMark,ioPosMode(A0)
    CLR.L   ioPosOffset(A0)
    _Read
@9  BSR     CodeBloc
    BSR     CheckSum
    PEA     RectEffMess2
    _EraseRect
    PEA     C20
    BSR     AfficheM1
    PEA     NumeroBlocLettre(A5)
    _DrawString
    MOVE    #1,CompteurEssais(A5)
@3  BSR     Envoilbloc
    BSR     SPNiveauEmission
@2  BSR     AttenteBlocOK
    TST     Code(A5)
    BNE.S   @4
    ADDQ    #1,CompteurEssais(A5)
    CMPI    #NombreMaxEssai+1,CompteurEssais(A5)
    BEQ.S   @6
    MOVE    CompteurEssais(A5),-(SP)
    BSR     AfficheNumEssai
    PEA     C28
    _DrawString
    BRA     @3
@6  BSR     AbandonEmission
    BRA     FinEmission
@4  MOVE    Code(A5),D0
    CMPI    #CAN_U,D0
    BNE.S   @8
    BSR     ArretEmission
    BRA     FinEmission
@8  CMPI    #CAN,D0
    BNE.S   @5
    BSR     AnnulationRecepteur
    BRA     FinEmission
@5  CMPI    #NACK,D0
    BNE.S   @7
    ADDQ    #1,CompteurEssais(A5)
    MOVE    CompteurEssais(A5),-(SP)
    BSR     AfficheNumEssai
    PEA     C29
    _DrawString
    BRA     @3
@7  SUBQ    #1,CompteurBlocs(A5)
    BNE     @1
    BSR     FermetureFichier
    MOVEQ   #0,D0
    RTS
FinEmission
    MOVEQ   #1,D0
    RTS
```

Fichier 'InterPom's/4.Asm'

```
ReceptionIBM
    RTS
ReceptionServeur
    RTS
VerificationBloc
    LEA     TamponEntre+TypeEmetteur(A5),A0
    MOVEQ   #0,D0
    MOVE    #LongueurBlocC,D1
@1  ANDI.B  #SixFaibles,(A0)
    ADD.B   (A0)+,D0
    DBRA    D1,@1
    ANDI.B  #SixFaibles,D0
    ANDI.B  #SixFaibles,(A0)
    CMP.B   (A0),D0
    BEQ.S   @2
@5  MOVE    #CHECK,Code(A5)
    BRA.S   @3
@2  MOVE    NumeroBloc(A5),D0
    ANDI.B  #SixFaibles,D0
    MOVE.B  TamponEntre+NumBloc(A5),D1
    ANDI.B  #SixFaibles,D1
    CMP.B   D1,D0
    BEQ.S   @4
    SUBQ.B  #1,D0
    CMP.B   D1,D0
    BNE.S   @6
    MOVE    #DIF,Code(A5)
    BRA.S   @3
@6  MOVE    #CAN_A,Code(A5)
    BRA.S   @3
@4  EORI.B  #SixFaibles,D1
    ANDI.B  #SixFaibles,TamponEntre+NotNumBloc(A5)
    CMP.B   TamponEntre+NotNumBloc(A5),D1
    BNE.S   @5
    CLR     Code(A5)
@3  RTS
LongueurNomFichierRecu
    CLR     -(SP)
    PEA     NomFichierRecu(A5)
    _StringWidth
    CMPI    #111,(SP)+
    BLT.S   @2
    MOVEQ   #0,D4
    LEA     NomFichierRecu(A5),A4
    MOVE.B  (A4),D4
@3  MOVE.B  #'…',0(A4,D4)
    CLR     -(SP)
    PEA     NomFichierRecu(A5)
    _StringWidth
    CMPI    #111,(SP)+
    BLT.S   @2
    SUBQ.B  #1,(A4)
    SUBQ    #1,D4
    BRA.S   @3
@2  RTS
DecodeBloc
    LEA     TamponEntre+TypeEmetteur(A5),A3
    MOVE    #170,D5
    MOVEQ   #0,D4
    MOVEQ   #0,D3
@1  MOVE.L  0(A3,D4),-(SP)
    BSR     Decode
    MOVE.B  1(SP),0(A3,D3)
    MOVE.B  2(SP),1(A3,D3)
    MOVE.B  3(SP),2(A3,D3)
    ADDQ.L  #4,SP
    ADDQ    #4,D4
    ADDQ    #3,D3
    DBRA    D5,@1
    RTS
Decode
Octet1  SET     8
Octet2  SET     9
Octet3  SET     10
Octet4  SET     11
    LINK    A6,#0
    MOVEM.L D0-D1,-(SP)
    ANDI.L  #$3F3F3F3F,Octet1(A6)
    MOVE    Octet3(A6),D0
    LSR     #2,D0
    ANDI.B  #$C0,D0
    OR.B    D0,Octet4(A6)
    LSR     #8,D0
    MOVE.B  Octet2(A6),D1
    LSL.B   #4,D1
    OR.B    D1,D0
    MOVE.B  D0,Octet3(A6)
    MOVE.B  Octet2(A6),D0
    LSR.B   #4,D0
    MOVE.B  Octet1(A6),D1
    LSL.B   #2,D1
    OR.B    D1,D0
    MOVE.B  D0,Octet2(A6)
    CLR.B   Octet1(A6)
    MOVEM.L (SP)+,D0-D1
    UNLK    A6
    RTS
VideBuffer
    LINK    A6,#0
    MOVE.L  8(A6),A0
    _KillIO,immed
    BEQ.S   @1
    MOVE    #2,-(SP)
    BSR     Erreur
    _ExitToShell
@1  UNLK    A6
    MOVE.L  (SP),4(SP)
    ADDQ.L  #4,SP
    RTS
SPNombreOctets
    MOVEQ   #0,D0
    MOVE.B  TamponEntre+98(A5),D0
    LSL     #8,D0
    MOVE.B  TamponEntre+97(A5),D0
    LSL.L   #8,D0
    MOVE.B  TamponEntre+96(A5),D0
    TST.L   D0
    BNE.S   @1
    MOVEQ   #1,D0
@1  MOVE.L  D0,NombreOctetRecu(A5)
    RTS
GetTimer
    SUBQ.L  #4,SP
    _TickCount
    MOVE.L  (SP)+,CompteurTiming(A5)
    RTS
TesteTimeOut
    SUBQ.L  #4,SP
    _TickCount
    MOVE.L  (SP)+,D0
    SUB.L   CompteurTiming(A5),D0
    CMPI.L  #720,D0
    BMI.S   @1
    MOVEQ   #1,D0
    RTS
@1  MOVEQ   #0,D0
    RTS
TesteAnnulerR
    _SystemTask
    SUBQ    #2,SP
    MOVE    #2,-(SP)
    PEA     EnregEvents(A5)
    _GetNextEvent
    TST     (SP)+
    BEQ.S   RetourTesteAnnulerNonR
@1  SUBQ    #2,SP
    MOVE.L  EnregEvents+evtMouse(A5),-(SP)
    PEA     EnregEvents+evtMeta+2(A5)
    _FindWindow
    CMPI    #inContent,(SP)+
    BNE.S   BeepTesteAnnulerR
    MOVEA.L EnregEvents+evtMeta+2(A5),A4
    TST.L   wRefCon(A4)
    BNE.S   BeepTesteAnnulerR
    MOVE.L  A4,-(SP)
    _SetPort
    PEA     EnregEvents+EvtMouse(A5)
    _GlobalToLocal
    SUBQ.L  #2,SP
    MOVE.L  EnregEvents+EvtMouse(A5),-(SP)
    MOVE.L  A4,-(SP)
    PEA     HandleControle(A5)
    _FindControl
    TST     (SP)+
    BEQ.S   BeepTesteAnnulerR
    SUBQ.L  #2,SP
    MOVE.L  HandleControle(A5),-(SP)
    MOVE.L  EnregEvents+EvtMouse(A5),-(SP)
    CLR.L   -(SP)
    _TrackControl
    TST     (SP)+
    BEQ.S   BeepTesteAnnulerR
    SUBQ.L  #4,SP
    MOVE.L  HandleControle(A5),-(SP)
    _GetCRefCon
    MOVEQ   #2,D0
    CMP.L   (SP)+,D0
    BNE.S   BeepTesteAnnulerR
    MOVE    #CAN,Code(A5)
    RTS
BeepTesteAnnulerR
    MOVE    #7,-(SP)
    _SysBeep
RetourTesteAnnulerNonR
    CLR     Code(A5)
    RTS
ChangeNumeroBloc
    ADDQ    #1,NumeroBloc(A5)
    MOVEQ   #0,D0
    MOVE    NumeroBloc(A5),D0
    LEA     NumeroBlocLettre(A5),A0
    CLR     -(SP)
    _Pack7
    RTS
SPNiveauReception
    MOVEA.L (A5),A0
    PEA     ltgray(A0)
    _PenPat
    MOVE.L  #16384,D5
    DIVU    NombreBlocRecu+2(A5),D5
    MULU    NumeroBloc(A5),D5
    LSR.L   #7,D5
    ADDI    #333,D5
    MOVE    D5,RectangleNiveauRB+right(A5)
    PEA     RectangleNiveauRB(A5)
    _PaintRect
    _PenNormal
    MOVE    D5,-(SP)
    MOVE    #$0064,-(SP)
    _MoveTo
    MOVE    D5,-(SP)
    MOVE    #$006A,-(SP)
    _LineTo
    RTS
FermetureFichier
    LEA     TamponIO(A5),A0
    _Close
    RTS
MiseJourVolume
    LEA     TamponIO(A5),A0
    CLR.L   ioFileName(A0)
    MOVE    VolumeCourant(A5),ioVRefNum(A0)
    _FlushVol
    BEQ.S   @1
    CLR     -(SP)
    BSR     Erreur
@1  RTS
LireInfoFichier
    LEA     TamponIO(A5),A0
    LEA     NomFichierRecu(A5),A1
    MOVE.L  A1,ioFileName(A0)
```

```
    MOVE     VolumeCourant(A5),ioVRefNum(A0)
    CLR.B    ioFileType(A0)
    CLR      ioFDirIndex(A0)
    _GetFileInfo
    BEQ.S    @1
    CLR      -(SP)
    BSR      Erreur
    MOVEQ    #1,D0
    RTS
@1  MOVEQ    #0,D0
    RTS
EcritBlocDisque
    LEA      TamponIO(A5),A0
    MOVE.L   #512,ioReqCount(A0)
    LEA      TamponEntre+TypeEmetteur(A5),A1
    MOVE.L   A1,ioBuffer(A0)
    MOVE     #fsAtMark,ioPosMode(A0)
    CLR.L    ioPosOffset(A0)
    _Write
    BEQ.S    @1
    CLR      -(SP)
    BSR      Erreur
    BSR      FermetureErr
    MOVEQ    #1,D0
    RTS
@1  MOVEQ    #0,D0
    RTS
NouveauFichier
    LEA      TamponIO(A5),A0
    LEA      NomFichierRecu(A5),A1
    MOVE.L   A1,ioFileName(A0)
    MOVE     VolumeCourant(A5),ioVRefNum(A0)
    _Create
    BEQ.S    NouveauFichierOuvert
    CMPI     #-48,D0
    BNE.S    @1
    BSR      NomExistant
    BRA.S    NouveauFichier
@1  MOVE     #3,-(SP)
    BSR      Erreur
    MOVEQ    #1,D0
    RTS
NouveauFichierOuvert
    MOVEQ    #0,D0
    RTS
OuvertureFichier
    LEA      TamponIO(A5),A0
    LEA      NomFichierRecu(A5),A1
    MOVE.L   A1,ioFileName(A0)
    CLR.L    ioOwnBuf(A0)
    MOVE.B   #fsWrPerm,ioPermssn(A0)
    MOVE     VolumeCourant(A5),ioVRefNum(A0)
    TST      D0
    BNE.S    @1
    _Open
    BNE.S    @2
@3  MOVEQ    #0,D0
    RTS
@1  _OpenRF
    BEQ.S    @3
@2  MOVEQ    #1,D0
    RTS
RenommerFichier
    LEA      NomFichierRecu(A5),A0
    LEA      NomFichierModif(A5),A1
    MOVEQ    #64,D0
    _BlockMove
    LEA      CMF,A0
    LEA      NomFichierModif(A5),A1
    MOVE.B   (A1),D0
    EXT      D0
    CMPI.B   #53,D0
    BLE.S    @1
    MOVEQ    #54,D0
    MOVE.B   #63,(A1)
    BRA.S    @3
@1  ADDQ     #1,D0
    ADD.B    #10,(A1)
@3  MOVEQ    #9,D1
@2  MOVE.B   (A0)+,0(A1,D0)
    ADDQ     #1,D0
    DBRA     D1,@2
    LEA      TamponIO(A5),A0
    LEA      NomFichierRecu(A5),A1
    MOVE.L   A1,ioFileName(A0)
    MOVE     VolumeCourant(A5),ioVRefNum(A0)
    LEA      NomFichierModif(A5),A1
    MOVE.L   A1,ioNewName(A0)
    CLR.B    ioFileType(A0)
    _Rename
    BEQ.S    @4
    CLR      -(SP)
    BSR      Erreur
@4  RTS
Beep7
    MOVE     #7,-(SP)
    _SysBeep
    RTS
Erreur
    LINK     A6,#0
    MOVE     8(A6),D4
    MOVE     D0,D3
    BSR      Beep7
    TST      D4
    BNE.S    @1
    BSR      TroisCANCEL
@5  MOVE     #1,D5
    BRA.S    FenetreErreur
@1  SUBQ     #1,D4
    BNE.S    @2
    MOVE     #2,D5
    BRA.S    FenetreErreur
@2  SUBQ     #1,D4
    BNE.S    @3
    MOVE     #3,D5
    BRA.S    FenetreErreur
@3  BSR      TroisCANCEL
    MOVE     #4,D5
FenetreErreur
    CLR.L    -(SP)
    MOVE.L   #'STR ',-(SP)
    MOVE     D5,-(SP)
    _GetResource
    MOVE.L   (SP)+,A2
    MOVE.L   A2,A0
    _HLock
    MOVE.L   (A2),-(SP)
    CLR.L    -(SP)
    CLR.L    -(SP)
    CLR.L    -(SP)
    _ParamText
    CLR      -(SP)
    MOVE     #128,-(SP)
    CLR.L    -(SP)
    _StopAlert
    MOVE     (SP)+,D0
    MOVE.L   A2,A0
    _HUnLock
@1  UNLK     A6
    MOVE.L   (SP),2(SP)
    ADDQ.L   #2,SP
    RTS
FermetureErr
    LEA      TamponIO(A5),A0
    _Close
    RTS
```

Fichier 'InterPom's/5.Asm'

```
ReceptionMac
    LEA      TamponEntre+TypeEmetteur(A5),A0
    LEA      TamponTemporaireF(A5),A1
    MOVEQ    #ioFQElSize,D0
    _BlockMove
    MOVE     TamponEntre+NBlocsData(A5),NombreBD(A5)
    MOVE     TamponEntre+NBlocsResource(A5),
             NombreBR(A5)
    BSR      DeplaceNFR
    MOVEQ    #0,D0
    MOVE     TamponEntre+NBlocsTotal(A5),D0
    LSR.L    #1,D0
    CMP.L    NombreKONum(A5),D0
    BMI.S    CreationFichierMac
    BSR      Beep7
    PEA      C16
    BSR      AfficheM1
    BSR      Tempo
    RTS
CreationFichierMac
    BSR      NouveauFichier
    BEQ.S    OuvertureFichierMac
    RTS
OuvertureFichierMac
    TST      NombreBD(A5)
    BEQ      OuvertureRessource
    MOVEQ    #0,D0
    BSR      OuvertureFichier
    BEQ.S    NumeroBlocMac
    RTS
NumeroBlocMac
    BSR      ChangeNumeroBloc
AttenteBlocMac
    BSR      AfficheAttenteBloc
    MOVE     #ACK,-(SP)
    BSR      EnvoiCode
Boucle1
    BSR      TesteTimeOut
    BEQ.S    @1
    BSR      AnnulationAbandon
    BRA      AbandonReceptionMac
@1  BSR      TesteAnnulerR
    TST      Code(A5)
    BEQ.S    @2
    BSR      AnnulationUtilisateur
    BRA      AbandonReceptionMac
@2  BSR      AttenteLiaisonEmetteur
    TST      Code(A5)
    BEQ.S    Boucle1
    BSR      GetTimer
    MOVE     Code(A5),D0
    CMPI     #SOH,D0
    BEQ.S    ReceptionBlocMac
    CMPI     #CAN,D0
    BNE.S    @3
    BSR      AnnulationEmetteur
    BRA      AbandonReceptionMac
@3  CMPI     #EOT,D0
    BEQ      FinReceptionMacData
DifMac
    BSR      CodeDifferent
DifMac2
    BSR      AfficheAttenteBloc
    MOVE     #NACK,-(SP)
    BSR      EnvoiCode
    BRA.S    Boucle1
ReceptionBlocMac
    PEA      C6
    BSR      AfficheM1
    PEA      NumeroBlocLettre(A5)
    _DrawString
    BSR      LireBloc
    MOVE     Code(A5),D0
    BEQ.S    VerificationBlocMac
    CMPI     #CAN_A,D0
    BNE.S    DifMac
    BSR      AnnulationAbandon
    BRA      AbandonReceptionMac
VerificationBlocMac
    BSR      VerificationBloc
    MOVE     Code(A5),D0
    BEQ.S    @1
    CMPI     #DIF,D0
    BNE.S    @2
    BSR      AfficheAttenteBloc
    MOVE     #ACK,-(SP)
    BSR      EnvoiCode
    BRA      Boucle1
@2  CMPI     #CAN_A,D0
    BNE      DifMac2
    BSR      AnnulationAbandon
    BRA      AbandonReceptionMac
@1  BSR      DecodeBloc
@3  BSR      SPNiveauReception
    BSR      EcritBlocDisque
    BEQ.S    @4
    RTS
@4  SUBQ     #1,NombreBD(A5)
    BNE      NumeroBlocMac
FinReceptionMacData
    LEA      TamponIO(A5),A0
    MOVE.L   TamponTemporaireF+ioFlLgLen(A5),
             ioLEOF(A0)
    _SetEOF
@1  BSR      FermetureFichier
    CMPI     #EOT,Code(A5)
    BEQ      FinReceptionMacData2
OuvertureRessource
    TST      NombreBR(A5)
    BEQ      FinReceptionMacData2
    MOVEQ    #1,D0
    BSR      OuvertureFichier
    BEQ.S    NumeroBlocMacR
    RTS
NumeroBlocMacR
    BSR      ChangeNumeroBloc
AttenteBlocMacR
    BSR      AfficheAttenteBloc
    MOVE     #ACK,-(SP)
    BSR      EnvoiCode
Boucle1R
    BSR      TesteTimeOut
    BEQ.S    @1
    BSR      AnnulationAbandon
    BRA      AbandonReceptionMac
@1  BSR      TesteAnnulerR
    TST      Code(A5)
    BEQ.S    @2
    BSR      AnnulationUtilisateur
    BRA      AbandonReceptionMac
@2  BSR      AttenteLiaisonEmetteur
    TST      Code(A5)
    BEQ.S    Boucle1R
    BSR      GetTimer
    MOVE     Code(A5),D0
    CMPI     #SOH,D0
    BEQ.S    ReceptionBlocMacR
    CMPI     #CAN,D0
    BNE.S    @3
    BSR      AnnulationEmetteur
    BRA      AbandonReceptionMac
@3  CMPI     #EOT,D0
    BEQ      FinReceptionMac
DifMacR
    BSR      CodeDifferent
DifMac2R
    BSR      AfficheAttenteBloc
    MOVE     #NACK,-(SP)
    BSR      EnvoiCode
    BRA.S    Boucle1R
ReceptionBlocMacR
    PEA      C6
    BSR      AfficheM1
    PEA      NumeroBlocLettre(A5)
    _DrawString
    BSR      LireBloc
    MOVE     Code(A5),D0
    BEQ.S    VerificationBlocMacR
    CMPI     #CAN_A,D0
    BNE.S    DifMacR
    BSR      AnnulationAbandon
    BRA      AbandonReceptionMac
VerificationBlocMacR
    BSR      VerificationBloc
    MOVE     Code(A5),D0
    BEQ.S    @1
    CMPI     #DIF,D0
    BNE.S    @2
    BSR      AfficheAttenteBloc
```

```
    MOVE    #ACK,-(SP)
    BSR     EnvoiCode
    BRA     Boucle1R
@2  CMPI    #CAN_A,D0
    BNE     DifMac2R
    BSR     AnnulationAbandon
    BRA     AbandonReceptionMac
@1  BSR     DecodeBloc
@3  BSR     SPNiveauReception
    BSR     EcritBlocDisque
    BEQ     NumeroBlocMacR
    RTS
FinReceptionMac
    LEA     TamponIO(A5),A0
    MOVE.L  TamponTemporaireF+ioFlRLgLen(A5),
            ioLEOF(A0)
    _SetEOF
@1  BSR     FermetureFichier
FinReceptionMacData2
    MOVE    #ACK,-(SP)
    BSR     EnvoiCode
    PEA     RectEffMess2
    _EraseRect
    PEA     C8
    BSR     AfficheM1
    BSR     LireInfoFichier
    BNE.S   @1
    LEA     TamponIO(A5),A0
    MOVE    TamponTemporaireF+ioFlUsrWds+
            fdFlags(A5),D0
    ANDI    #$2000,D0
    MOVE    D0,ioFlUsrWds+fdFlags(A0)
    MOVE.L  TamponTemporaireF+ioFlUsrWds+
            fdType(A5),ioFlUsrWds+fdType(A0)
    MOVE.L  TamponTemporaireF+ioFlUsrWds+
            fdCreator(A5),ioFlUsrWds+fdCreator(A0)
    CLR.L   ioFlUsrWds+fdLocation(A0)
    CLR     ioFlUsrWds+fdFldr(A0)
    _SetFileInfo
    BGE.S   @1
    CLR     -(SP)
    BSR     Erreur
@1  BSR     MiseJourVolume
    BSR     Tempo
RetourReceptionMac
    RTS
AbandonReceptionMac
    BSR     FermetureFichier
    BSR     RenommerFichier
    BSR     MiseJourVolume
    BSR     Tempo
    RTS
```

Fichier 'InterPom's.Link'

```
]

Fichiers:InterPom's
/Output Fichiers:InterPom's.Code
/Type 'TEMP'

$
```

Fichier 'InterPom's.R'

```
Fichiers:InterPom's
APPLTa2m
Type Ta2m = STR
  ,0
  Pom's - InterPom's Version 1.0 - Janvier 1987
Type MENU
  ,1
\14
  ^1À propos d'InterPom's…
  (-
  ,2
Fichier
  ^2Texte seulement…/T
  ^3Tout documents…/O
  (-
  (Fermer
  (-
  ^4Quitter/Q
  ,3
Édition
  Annuler/Z
  (-
  ^5Couper/X
  ^6Copier/C
  ^7Coller/V
  (-
  Effacer
  ,4
Mode
  !\12^8Réception/R
  ^9Émission/E
Type ALRT
     ,128 (4)
82 72 178 440
128
5555
Type DITL
     ,128 (4)
2
Button
56 256 80 352
OK
StaticText
16 64 48 352
^0^1^2^3
Type STR
     ,1 (4)
Erreur d'entrée/sortie.
     ,2 (4)
Fichier ouvert ou utilisé. Dupliquez-le pour l'envoyer.
     ,3 (4)
Problème d'interface série. Impossible de continuer.
     ,4 (4)
Le catalogue est saturé. Changez de volume.
Type DLOG
  ,1 (4)
82 72 298 440
Visible  NoGoAway
1
0
1

Type DITL
  ,1 (4)
13
StaticText
0 0 216 368
\20
IconItem
24 24 56 56
257
IconItem
168 312 200 344
266
StaticText
12 72 28 352
InterPom's - V 1.0/Mac (protocole ALC)
StaticText
36 72 52 352
© 1987 Pom's/Éditions MEV
StaticText
52 72 68 352
Jean-Luc Bazanegue et Christian Piard
StaticText
76 24 92 352
"InterPom's" permet la transmission de fichiers
StaticText
92 24 108 352
- ou applications - entre Macintosh, ou entre
StaticText
108 24 124 352
Apple][™ et Macintosh™.
StaticText
132 24 148 352
Le mode d'emploi et les 'sources' (MDS 68000) se
StaticText
148 24 164 352
trouvent dans le numéro 28 de "Pom's".
StaticText
172 24 188 296
Pom's/Éditions MEV - 12, rue d'Anjou -
StaticText
188 24 204 296
78000 Versailles - Tél. : (1) 39 51 24 43.
Type BNDL
   ,128(32)
   Ta2m 0
   ICN#
   0 128
   FREF
   0 128
Type FREF
   ,128(32)
   APPL 0
TYPE ICN# = GNRL
  ,128(32)
2
00000000 000000C0 00000140 00000280
00000300 00007A78 0000FFAC 003DFD56
0015FA8A 002DF546 0055FA8A 00A1F546
0140FA8C 0080F554 00307AA8 00503550
00A01BA0 00C00CC0 1E9E0100 3FEB0280
7F558500 7EA2AA00 7D51B400 7EA2A800
7D51BC00 3EA30000 3D550000 1EAA0000
0D540000 06E80000 03300000 00000000
*
000001E0 000003E0 000007E0 000007E0
0000FFFC 0001FFFE 007FFFFF 007FFFFF
007FFFFF 00FFFFFF 01FFFFFF 03FFFFFF
03F3FFFF 03F9FFFE 01F9FFFE 01F8FFFC
01F87FF8 3FFF3FF0 7FFF9FE0 FFFFCFC0
FFFFFFC0 FFFFFF80 FFFFFF00 FFFFFE00
FFFFFE00 FFFFFE00 7FFF8000 7FFF8000
3FFF0000 1FFE0000 0FFC0000 07F80000
TYPE ICON = GNRL
  ,257 (4)
00000000 000000C0 00000140 00000280
00000300 00007A78 0000FFAC 003DFD56
0015FA8A 002DF546 0055FA8A 00A1F546
0140FA8C 0080F554 00307AA8 00503550
00A01BA0 00C00CC0 1E9E0100 3FEB0280
7F558500 7EA2AA00 7D51B400 7EA2A800
7D51BC00 3EA30000 3D550000 1EAA0000
0D540000 06E80000 03300000 00000000
  ,258 (4)
0FFF0000 08018000 0B014000 0A81E000
0B6EA000 0A6EA000 F8002000 8BF7A000
B8002000 AB7FF800 B8400C00 ABD87FF8
8854400C BBDB580A 8853540F BAC05B75
885ED375 BBC04001 885BDBED B8404001
8FDF5BFD B7C04001 8057DDDD BEC04001
805E5BFD 80404001 FFCFDEFD 00404001
00405EDD 007FC001 00004001 00007FFF
  ,259 (4)
0FFF0000 08018000 0BFF4000 0BFFE000
0BFFA000 0BFFA000 F8002000 8BFFA000
BBFFA000 ABFFF800 B8400C00 ABD80A00
8BD40F00 BBDB7700 88537500 BBC00880
8BDE9040 BBC02020 8BDBC510 B8408A08
8FDF1404 B7C22802 80544781 BEC2087F
805F3E0F 8040880F FFCFC7FF 0040202F
00401040 007FF880 00000500 00000200
  ,260 (4)
0FFFFFF0 10000008 13FFFFC8 17FFFFE8
1668F468 16D5EAE8 1645E2E8 16D5EAE8
1655EAE8 17FFFFE8 17998A68 179494E8
17949CE8 17948E68 17949E68 16949E68
173988E8 17FFFFE8 13FFFFC8 10000008
10000008 10000008 100001C8 1200FFC8
10000008 10000008 10000008 0FFFFFF0
08000010 0A000050 08000010 07FFFFE0
  ,261 (4)
00000010 00000028 00000054 000000A8
00000110 00000230 00000420 00000840
00001040 00002080 00004100 0000C100
00018200 00034400 00068400 000D0800
001A1000 00351000 00EA2000 01F44000
03FA4000 1EFC8000 3CD90000 3CFD0000
3C7A0000 1E3C0000 07380000 03B80000
01F00000 00E00000 00C00000 00800000
  ,262 (4)
00007FFF 00014001 00005EED 00045EED
00004001 00105FBD 00005FBD 00424001
00005D7D 01005D7D 00104001 040057BD
000157BD 10804001 00007FFF 42080002
00000000 FFFE1088 80020000 BDDA4020
BDDA0000 80020880 BF7A0000 BF7A0200
80024000 BAFA0800 BAFA0000 80022000
AF7A0000 AF7A8000 80020000 FFFE0000
  ,263 (4)
00000000 00000004 0000000E 0000001C
00000038 000030D0 00007920 0000FEA0
0000D740 0001AB80 00035180 0006A0C0
000D54E0 000E1960 001D8AC0 0039E580
00F24A00 01E91400 07552800 0EB67000
3D14C000 7A498000 FD230000 6C950000
364A0000 1B140000 0DAC0000 04D80000
02700000 01300000 00A00000 00400000
  ,264 (4)
00000000 000000C0 00000140 00000280
00000300 00007A78 0000FFAC 0001FD56
0001FA8A 0001F546 0001FA8A 0001F546
0000FA8C 0000F554 0FFFFFA8 300000D6
40F0F021 81FFF812 FE8017F4 83000C12
7E36C7E4 04000208 0436C204 08000108
0836C110 08000108 1036C090 10000088
180001D0 17FFFEA0 08000100 07FFFE00
  ,265 (4)
00FFFFF0 0300000C 040F0F02 081FFF81
0FE8017F 083000C1 07E36C7E 00400020
00436C20 00800010 00836C10 00800010
01036C08 0100000E 01B00019 015FFFEA
00A00011 00FFFFE6 1E9E0004 3FEB0018
7F558060 7EA2AA80 7D51D500 7EA28000
7D518000 3EA30000 3D550000 1EAA0000
0D540000 06E80000 03300000 00000000
  ,266 (4)
00000000 00000050 00000080 00001100
000008AA 00000155 000000AA 00001401
00002A02 00004514 0000A280 00015140
000228A0 00051440 00028A00 00014560
0028A2A0 00545140 008A2980 00453D7C
00A2FFD6 1451FEAB 2A28FD45 4515FAA3
A20AFD45 5100FAA3 2A007D46 14007AAA
0A003D54 05001AA8 02800DD0 01000660
  ,267 (4)
7FFFFF70 AA8281A8 DD950154 AAAAF1AA
DDD4F156 AAA0F1AA DDC0F156 AA80F1AA
DD80F156 AA80F1AA DD800156 AFFFFEAA
D5555556 AAAAAAAA D5555556 AFFFFFEA
D8000036 ABFFFFAA D8000036 ABFFFFAA
D8000036 A99530AA DA1548B6 A91548AA
D8954BB6 AB0B34AA D84004B6 A84003AA
D8000036 ABFFFFAA D8000036 7FFFFFFC
  ,268 (4)
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000
00000000 1FFFFFF0 3FFFFFF8 7FFFFFFC
D5555556 AAAAAAAA D5555556 AAAAAAAA
D5555556 AFFFEAAA D8003556 AB6F2AAA
DB6DB556 ABEDAAAA DB6DB556 AB6F2BEA
D8003636 AFFFEBEA D5555556 7FFFFFFC
INCLUDE Fichiers:InterPom's.Code
```

Scrollings en HGR

Roland Jost

Après les effets de miroir de JANUS (Pom's 25), nous vous proposons ici de réaliser des défilements ('scrolling'), verticaux ou horizontaux, de la totalité de l'écran HGR ou simplement d'une partie de celui-ci.

Ce programme n'est pas le premier du genre : dans les premiers numéros de Pom's, Jean-François Duvivier réalisait un dédoublement d'écran. HAIFA (Pom's 5) et ICARE (Pom's 14) comportent aussi un module de 'scrolling'. Dans le numéro 21 de Pom's Patrick Quettier proposait un programme de défilement graphique, dans un sens seulement. Pour ne pas oublier la littérature étrangère, citons HI-RES FULL SCROLL dans CALL APPLE #V,(2)-1982 et surtout HIRES.HOUDINI (NIBBLE #V(10), 1984). Enfin, certains génériques de jeux ou de logiciels graphiques utilisent des effets de défilement.

Il fallait donc apporter un 'plus' par rapport aux programmes existants. Nous pensons que c'est le cas de SCROLLINGS.

Les possibilités de SCROLLINGS

- déroulement de l'écran HGR dans les 4 directions (gauche, droite, haut, bas) ;
- 3 modes possibles :
  - vider l'écran de manière progressive ;
  - obtenir un déroulement continu en entrant d'un côté de l'écran ce qui disparaît de l'autre ;
  - 'scrolling' sur deux pages, la page 2 venant combler la page 1. Dans ce cas, le contenu de la page 1 est perdu ;
- l'ampleur du mouvement peut être définie ;
- le déroulement peut se faire sur tout l'écran ou seulement dans une fenêtre définissable par l'utilisateur. Dans cette fenêtre, on peut faire défiler le contenu d'une page entière.

Le scrolling peut être stoppé temporairement (appui sur la touche <espace>) ou interrompu, avec reprise du cours normal du programme Basic appelant (touche <ESC>).

Quelques explications

Le principe du déroulement d'écran est simple : dans le sens vertical, pour obtenir le déplacement d'une ligne vers le haut, il suffit de copier chaque ligne de 280 points (40 octets) dans la ligne précédente. On pourra mettre des zéros dans la dernière ligne (191) pour obtenir un 'scrolling simple' avec vidage de l'écran. Si l'on a pris la précaution de sauver la ligne du haut (ligne 0) , on peut la recopier dans celle du bas et obtenir un scrolling en continu. Enfin, on peut recopier une ligne d'une autre page dans la ligne du bas et réaliser ainsi un 'scrolling' sur deux pages.

En répétant *n* fois la manœuvre, le déplacement sera plus ou moins important. Pour N=192, on opère sur toute la page.

De même, pour un 'scrolling' latéral, on opère sur les 40 colonnes de 192 octets de l'écran HGR.

On peut définir une fenêtre en ne travaillant que sur *nx* colonnes et *ny* lignes.

Le programme SCROLLINGS est loin d'être optimisé. Il se compose de 4 modules permettant chacun le défilement dans une direction. La construction des différents modules est similaire et on y retrouve des groupes d'instructions identiques qui auraient pu être mises en sous-programmes communs. Mais le lecteur pourra ainsi, s'il le désire, ne prendre que le module nécessaire à ses besoins.

Occupation de la mémoire

SCROLLINGS est chargé à l'adresse 37376 ($9200) et doit être protégé à partir du Basic par HIMEM:36864.

$9000: table des octets bas des adresses des 192 lignes de la page graphique 1.
$90C0: table des octets de poids fort.
$9180: stockage provisoire.
$9200: SCROLLINGS.
$94A3: fin du programme.

Utiliser SCROLLINGS

```
CALL 37376,Direction,
Type,Amplitude,
X,Y,X1,Y1
```

Direction = G, D, H, B.

Type = 0 : scrolling simple ;
1 : scrolling continu ;
2 : scrolling de la page 2 dans la page 1.

Amplitude : de 0 à 255.

X,Y : coordonnées du point supérieur gauche de la fenêtre :

```
0 <= X <= X1
0 <= Y <= Y1
```

X1,Y1 : coordonnées du coin inférieur droit :

```
X <= X1 <= 39
Y <= Y1 <= 191
```

graphiques.

Programme 'UTIL-SCROLLINGS'

```
1    REM *********************
2    REM * UTIL-SCROLLINGS    *
3    REM *                   *
4    REM *  R.JOST 1986       *
5    REM * UTILISE            *
6    REM *   SCROLLINGS.OBJ0 *
7    REM *   TABLES.HGR.OBJ0 *
8    REM *                   *
9    REM *********************
1000 ADR = 37376: HIMEM: 36864
1010 G$ =  CHR$ (7):D$ =  CHR$ (4)
1020  PRINT  CHR$ (4)"BLOAD TABLES.HGR.OBJ
     0"
1030  PRINT  CHR$ (4)"BLOAD SCROLLINGS.OBJ
     0"
1040 X = 0:X1 = 39:Y = 0:Y1 = 191
1050 MODE = 0
1060  GOSUB 8010: GET R$: PRINT R$
1070  GOSUB 5010
3000 :
3010  REM  COMMANDES
3020 :
3030  WAIT  - 16384,128,127:R =  PEEK ( -
     16384) - 128: POKE  - 16368,0
3040  IF R = 83 THEN  GOSUB 7010: GOTO 303
     0: REM S)AUVE LA PAGE 1
3050  IF R = 76 THEN  GOSUB 6010: GOTO 303
     0 -  REM L)OADPAGE1OU2
3060  IF R = 48 THEN MODE = 0: GOTO 3030:
     REM 0
3070  IF R = 49 THEN MODE = 1: GOTO 3030:
     REM 1
3080  IF R = 50 THEN MODE = 2: GOTO 3030:
     REM 2
3090  IF R = 77 THEN  GOSUB 4030: GOTO 303
     0: REM M)ODIFIE LA FENETRE
3100  IF R = 78 THEN X = 0:Y = 0:X1 = 39:Y
     1 = 191: GOTO 3030: REM FENETRE N)ORM
     ALE
3110  IF R = 73 THEN  GOSUB 8010: GOTO 303
     0: REM I)NSTRUCTIONS
3120  IF R = 11 THEN  CALL ADR,H,MODE,192,
     X,Y,X1,Y1: GOTO 3030: REM FLECHE HAUT
     E OU CTRL-K
3130  IF R = 10 THEN  CALL ADR,B,MODE,192,
     X,Y,X1,Y1: GOTO 3030: REM FLECHE BASS
     E OU CTRL-J
3140  IF R = 8 THEN  CALL ADR,G,MODE,39,X,
     Y,X1,Y1: GOTO 3030: REM FLECHE GAUCHE

3150  IF R = 21 THEN  CALL ADR,D,MODE,39,X
     ,Y,X1,Y1: GOTO 3030: REM FLECHE DROIT
     E
3160  IF R = 81 THEN  TEXT : END : REM QUI
     TTER
3170  GOTO 3030
4000 :
4010  REM  MODIFICATION DE LA FENETRE
4020 : REM
4030  TEXT : HOME : PRINT "MODIFICATION DE
      LA FENETRE": PRINT
4040  HTAB 1: VTAB 5: PRINT "COORDONNEES D
     U COIN SUPERIEUR GAUCHE:"
4050  HTAB 1: VTAB 7: PRINT "X ";:ZH =  PE
     EK (36) + 1: PRINT X;: HTAB ZH: VTAB
     7: INPUT "";X$: IF X$ <  > "" THEN X
     =  VAL (X$)
4060  IF X < 0 OR X > 39 THEN  PRINT G$: G
     OTO 4050
4070  HTAB 1: VTAB 8: PRINT "Y ";:ZH =  PE
     EK (36) + 1: PRINT Y;: HTAB ZH: VTAB
     8: INPUT "";Y$: IF Y$ <  > "" THEN Y
     =  VAL (Y$)
4080  IF Y < 0 OR Y > 191 THEN  PRINT G$:
     GOTO 4070
4090  HTAB 1: VTAB 10: PRINT "COORDONNEES
     DU COIN INFERIEUR DROIT:"
4100  HTAB 1: VTAB 11: PRINT "X1 ";:ZH =
     PEEK (36) + 1: PRINT X1;: HTAB ZH: VT
     AB 11: INPUT "";X1$: IF X1$ <  > "" T
     HEN X1 =  VAL (X1$)
4110  IF X1 > 39 OR X1 <  = X THEN  PRINT
     G$: GOTO 4100
4120  HTAB 1: VTAB 12: PRINT "Y1 ";:ZH =
     PEEK (36) + 1: PRINT Y1;: HTAB ZH: VT
     AB 12: INPUT "";Y1$: IF Y1$ <  > "" T
     HEN Y1 =  VAL (Y1$)
4130  IF Y1 > 191 OR Y1 <  = Y THEN  PRINT
      G$: GOTO 4120
4140  GOSUB 5030
4150  RETURN
5000 :
5010  REM  AFFICHAGE PAGE HGR SANS EFFACEM
     ENT
5020 :
5030 Z =  PEEK (49232) +  PEEK (49234) +
     PEEK (49236) +  PEEK (49239): RETURN
6000 :
6010  REM  CHARGEMENT D'UNE PAGE GRAPHIQUE
6020 :
6030 Z$ = "1"
6040  TEXT : HOME
6050  PRINT "CHARGEMENT EN PAGE ";:PH =  P
     EEK (36) + 1:PV =  PEEK (37) + 1: PRI
     NT Z$: HTAB PH: VTAB PV: GET Z$
6060  IF  VAL (Z$) = 0 THEN Z$ = "1"
6070  IF (Z$ <  > "2" AND Z$ <  > "1") THE
     N 6040
6080 AD$ = ",A$" +  STR$ (2 *  VAL (Z$)) +
      "000"
6090  PRINT Z$: PRINT
6100  INPUT "IMAGE A CHARGER ";N$: IF N$ =
      "" THEN  RETURN
6110  IF N$ = "?" THEN  PRINT  CHR$ (13);
     CHR$ (4)"CATALOG": GOTO 6100
6120 EE = 1: ONERR  GOTO 9010
6130  PRINT  CHR$ (13); CHR$ (4)"B
         LOAD "N$ + AD$
6140  GOSUB 5030: RETURN
7000 :
7010  REM  SAUVEGARDE D'UNE IMAGE
```

```
7020 :
7030  TEXT : HOME : PRINT "SAUVEGARDE DE L
     A PAGE 1": PRINT
7040  INPUT "NOM ";N$: IF N$ = "" THEN 707
     0
7050 EE = 2: ONERR  GOTO 9010
7060  PRINT  CHR$ (13); CHR$ (4)"BSAVE "N$
     ",A$2000,L$1FF8"
7070  GOSUB 5030: RETURN
8000 :
8010  REM  AFFICHAGE INSTRUCTIONS
8020 :
8030  TEXT : HOME
8040  INVERSE : PRINT "   SCROLLINGS  -  R
     .JOST  (C) 1985       ": NORMAL
8050  PRINT " SCROLLING DANS LES 4 DIRECTI
     ONS PAR          LES FLECHES "
8060  PRINT " (SUR LE II+ , UTILISER LES 2
      FLECHES,    CTRL-J ET CTRL-K.)": PRIN
     T
8070  PRINT " 0 : SCROLLING NORMAL ": PRIN
     T " 1 : SCROLLING 'PERPETUEL'": PRINT
      " 2 : PAGE 2 DANS PAGE 1": PRINT
8080  PRINT " L : CHARGEMENT D'UNE IMAGE E
     N PAGES     1 ET 2"
8090  PRINT " M : MODIFIER LA FENETRE ": P
     RINT
8100  PRINT " N : RETOUR A LA FENETRE STAN
     DARD"
8110  PRINT " Q : QUITTER LE PROGRAMME"
8120  PRINT " S : SAUVEGARDE DE LA PAGE 1"
8130  PRINT " I : AFFICHER LES INSTRUCTION
     S": PRINT
8140  PRINT " <ESPACE> POUR FAIRE UNE PAUS
     E "
8150  PRINT " <ESC> POUR STOPPER LE SCROLL
     ING"
8160  HTAB 1: VTAB 23: PRINT "TAPER <RETUR
     N> POUR CONTINUER ";: GET Z$
8170  GOSUB 5030
8180  RETURN
9000 :
9010  REM  TRAITEMENT ERREURS
9020 :
9030  TEXT : HOME
9040 ER =  PEEK (222)
9050  IF ER = 6 THEN  PRINT "CE FICHIER N'
     EXISTE PAS": GOTO 9090
9060  IF (ER = 4 OR ER = 9) THEN  PRINT "D
     ISK REMPLI OU PROTEGE": GOTO 9090
9070  IF ER = 8 THEN  PRINT "ERREUR I/O -
     ARRET DU PROGRAMME "
9080  POKE 216,0: END
9090  POKE 216,0: GET R$
9100  IF EE = 1 THEN R = 76: GOTO 3040
9110  IF EE = 2 THEN R = 83: GOTO 3040
9120  END
```

Source 'SCROLLINGS'

Assembleur ToolKit

```
 1 *
 2 *
 3 *  R.JOST 1985
 4 *
 5 * SCROLLINGS
 6 *
 7 *
 8 * CALL 37376,DIRECTION,MODE,N,X,Y,X1,Y1
 9 *
10 * DIRECTION = G)AUCHE, D)ROITE, H)AUT, B)AS
11 *
12 * MODE = 0 ->  DEFILEMENT SIMPLE
13 * MODE = 2 ->  RECOPIE PAGE2->PAGE1
14 * MODE = 1 ->  SCROLLING CONTINU
15 *
16 *
17 * N = AMPLITUDE DU DEPLACEMENT
18 *   N>0 ET N<=192
19 * X,Y ET X1,Y1 DEFINISSENT LA FENETRE
20 *
21 *
22 * EN COURS DE SCROLLING
23 *
24 * <ESC> PERMET DE REVENIR AU PROGRAMME BASIC A
                                  PPELANT
25 * <SPACE> PERMET UN ARRET TEMPORAIRE
26 * REPRISE PAR FRAPPE D'UNE TOUCHE
27 *
28 N        EQU  $06
29 C2       EQU  $07
30 C3       EQU  $08
31 DEPART   EQU  $18
32 ARRIVEE  EQU  $1A
33 TAMPON   EQU  $1C
34 ARR2     EQU  $1E
35 X1       EQU  $EB
36 X2       EQU  $EC
37 Y1       EQU  $ED
38 Y2       EQU  $EE
39 MODE     EQU  $EF
40 BUFFER   EQU  $FA
41 CHRGET   EQU  $00B1      ;
42 CHKCOM   EQU  $DEBE      ; teste la virgule
43 GETBYTC  EQU  $E6F5      ; saisie d'une valeur
44 *
45 TAB1L    EQU  $9000      ; TABLE ADRESSES (OCTET
                             S BAS)
46 TAB1H    EQU  $90C0      ; TABLE ADRESSES (OCTET
                             S HAUTS - PAGE 1)
47 *
48 KBD      EQU  $C000      ; TESTE LE CLAVIER
49 KBDSTRB  EQU  $C010      ; STROBE
50 NEWSTT   EQU  $D7D2      ; RETOUR AU BASIC A L'I
                             NSTRUCTION QUI SUIT
51 KBDWAIT  EQU  $FB88      ; ATTENTE TOUCHE
52 *
53 *
54 *
55          ORG  $9200
56 *
57 *
58 *
59 * Saisie de la direction du mouvement
60 * et dispatching vers les différents modules
61 *
62          JSR  CHKCOM
63          CMP  #$47       ; G)auche ?
64          BNE  OP2
65          JMP  LATERAL
66 OP2      CMP  #$44       ; D)roite ?
67          BNE  OP3
```

```
 68          JMP  LATERAL
 69 OP3      CMP  #$42       ; B)as
 70          BNE  OP4
 71          JMP  BAS
 72 OP4      CMP  #$48       ; H)aut
 73          BNE  OP5
 74          JMP  HAUT
 75 OP5      RTS
 76 *
 77 * Scrolling vers le HAUT
 78 *
 79 HAUT     JSR  CHRGET     ; saisie des paramètres
 80          JSR  GETBYTC
 81          STX  MODE
 82          JSR  GETBYTC
 83          STX  N
 84          JSR  GETBYTC
 85          STX  X1
 86          JSR  GETBYTC
 87          STX  Y1
 88          JSR  GETBYTC
 89          INX
 90          STX  X2
 91          JSR  GETBYTC
 92          STX  Y2
 93          LDA  #$00
 94          STA  C3
 95          LDA  #$80
 96          STA  BUFFER
 97          LDA  #$91
 98          STA  BUFFER+1
 99          LDX  Y2         ; adresse de la dernièr
100          LDA  TAB1L,X     e ligne
101          STA  ARR2
102          LDA  TAB1H,X
103          STA  ARR2+1
104 SCROLLUP JSR  TEST
105          LDX  Y1
106          STX  C2
107          LDA  TAB1L,X    ; adresse de la ligne e
                              n cours
108          STA  ARRIVEE
109          LDA  TAB1H,X
110          STA  ARRIVEE+1
111          LDY  X1
112 BE       LDA  (ARRIVEE),Y
113          STA  (BUFFER),Y ; stockage temporaire
114          INY
115          CPY  X2
116          BNE  BE
117 BOUCLE1  LDX  C2
118          LDA  TAB1L,X
119          STA  ARRIVEE
120          LDA  TAB1H,X
121          STA  ARRIVEE+1
122          INX
123          LDA  TAB1L,X
124          STA  DEPART
125          LDA  TAB1H,X
126          STA  DEPART+1
127          LDY  X1
128 BOUCLE2  LDA  (DEPART),Y ; on balaye la ligne e
                              n cours
129          STA  (ARRIVEE),Y
130          INY
131          CPY  X2
132          BNE  BOUCLE2
133          STX  C2
134          CPX  Y2
135          BNE  BOUCLE1
136 *
137 * Que faire de la dernière ligne ?
138 *
139          LDA  MODE
140          BNE  BOUCFI2
141          LDY  X1
142 BOUCF1   STA  (DEPART),Y ; scrolling simple
143          INY
144          CPY  X2
145          BNE  BOUCF1
146          DEC  N
147 SCROUP   BNE  SCROLLUP
148          RTS
149 BOUCFI2  CMP  #02
150          BNE  BOUCFI3
151          LDX  C3         ; page 2 -> page 1
152          LDA  TAB1L,X
153          STA  TAMPON
154          LDA  TAB1H,X
155          CLC
156          ADC  #$20
157          STA  TAMPON+1
158          INC  C3
159          LDY  X1
160 BOUCF2   LDA  (TAMPON),Y
161          STA  (ARR2),Y
162          INY
163          CPY  X2
164          BNE  BOUCF2
165          DEC  N
166          BNE  SCROLLUP
167          RTS
168 BOUCFI3  LDY  X1         ; scrolling continu
169 BOUCF3   LDA  (BUFFER),Y
170          STA  (ARR2),Y
171          INY
172          CPY  X2
173          BNE  BOUCF3
174          DEC  N
175          BEQ  FIN
176          BNE  SCROUP
177 FIN      RTS
178 *
179 *
180 *Scrolling vers le BAS
181 *
182 *
183 BAS      JSR  CHRGET     ; saisie des paramètres
184          JSR  GETBYTC
185          STX  MODE
186          JSR  GETBYTC
187          STX  N
188          JSR  GETBYTC
189          STX  X1
190          JSR  GETBYTC
191          STX  Y2
192          JSR  GETBYTC
193          INX
194          STX  X2
195          JSR  GETBYTC
196          STX  Y1
197          LDA  #191
198          STA  C3
199          LDA  #00
200          STA  BUFFER
201          LDA  #$60
202          STA  BUFFER+1
203          LDX  Y2
204          LDA  TAB1L,X
205          STA  ARR2
206          LDA  TAB1H,X
207          STA  ARR2+1
208 SCROLLDW JSR  TEST
209          LDX  Y1
210          STX  C2
211          LDA  TAB1L,X
212          STA  ARRIVEE
213          LDA  TAB1H,X
```

```
214          STA  ARRIVEE+1
215          LDY  X1
216 BEDW     LDA  (ARRIVEE),Y
217          STA  (BUFFER),Y
218          INY
219          CPY  X2
220          BNE  BEDW
221 BOUCDW1  LDX  C2
222          LDA  TAB1L,X
223          STA  ARRIVEE
224          LDA  TAB1H,X
225          STA  ARRIVEE+1
226          DEX
227          LDA  TAB1L,X
228          STA  DEPART
229          LDA  TAB1H,X
230          STA  DEPART+1
231          LDY  X1
232 BOUCDW2  LDA  (DEPART),Y
233          STA  (ARRIVEE),Y
234          INY
235          CPY  X2
236          BNE  BOUCDW2
237          STX  C2
238          CPX  Y2
239          BNE  BOUCDW1
240 *
241 * Que faire de la dernière ligne ?
242 *
243          LDA  MODE
244          BNE  BOUDWI2
245          LDY  X1          ; scrolling simple
246 BOUDW1   STA  (DEPART),Y
247          INY
248          CPY  X2
249          BNE  BOUDW1
250          DEC  N
251 SCRODW   BNE  SCROLLDW
252          RTS
253 BOUDWI2  CMP  #02
254          BNE  BOUDWI3
255          LDX  C3          ; page 2 -> page 1
256          LDA  TAB1L,X
257          STA  TAMPON
258          LDA  TAB1H,X
259          CLC
260          ADC  #$20
261          STA  TAMPON+1
262          DEC  C3
263          LDY  X1
264 BOUDW2   LDA  (TAMPON),Y
265          STA  (ARR2),Y
266          INY
267          CPY  X2
268          BNE  BOUDW2
269          DEC  N
270          BNE  SCROLLDW
271          RTS
272 BOUDWI3  LDY  X1          ; scrolling continu
273 BOUDW3   LDA  (BUFFER),Y
274          STA  (ARR2),Y
275          INY
276          CPY  X2
277          BNE  BOUDW3
278          DEC  N
279          BEQ  FINI
280          BNE  SCRODW
281 FINI     RTS
282 *
283 *
284 *Scrolling LATERAL
285 *
286 *
287 LATERAL STA  $09
288          JSR  CHRGET     ; saisie des paramètres
289          JSR  GETBYTC
290          STX  MODE
291          JSR  GETBYTC
292          STX  N
293          JSR  GETBYTC
294          STX  X1
295          JSR  GETBYTC
296          DEX
297          STX  Y1
298          JSR  GETBYTC
299          STX  X2
300          JSR  GETBYTC
301          STX  Y2
302          LDA  $09        ; à droite ou à gauche?
303          CMP  #$47
304          BEQ  SCRLEFT
305 *
306 * Scrolling à DROITE
307 *
308          LDX  #39
309          STX  C3
310 DEBUT    LDX  Y2         ; dernière ligne
311 DEBUT2   LDA  TAB1L,X
312          STA  DEPART
313          STA  ARR2
314          LDA  TAB1H,X
315          STA  DEPART+1
316          CLC
317          ADC  #$20
318          STA  ARR2+1
319          LDY  X2
320          LDA  MODE
321          BNE  MODE1
322          PHA
323          JMP  SRCLIG
324 MODE1    CMP  #01
325          BNE  MODE2
326          LDA  (DEPART),Y
327          PHA
328          JMP  SRCLIG
329 MODE2    CMP  #02
330          BNE  MODE3
331          LDY  C3
332          LDA  (ARR2),Y
333          PHA
334          LDY  X2
335          JMP  SRCLIG
336 MODE3    NOP
337 SRCLIG   DEY
338          LDA  (DEPART),Y
339          INY
340          STA  (DEPART),Y
341          DEY
342          CPY  X1
343          BNE  SRCLIG
344          PLA
345          STA  (DEPART),Y
346          DEX
347          CPX  Y1
348          BNE  DEBUT2
349          JSR  TEST
350          DEC  C3
351          DEC  N
352          LDA  N
353          CMP  #$FF
354          BNE  DEBUT
355          RTS
356 *
357 * Scrolling GAUCHE
358 *
359 SCRLEFT  LDX  #00
360          STX  C3
361 DEBLFT   LDX  Y2         ; derniere ligne
```

```
362 DEBLFT2 LDA  TAB1L,X
363         STA  DEPART
364         STA  ARR2
365         LDA  TAB1H,X
366         STA  DEPART+1
367         CLC
368         ADC  #$20
369         STA  ARR2+1
370         LDY  X1
371         LDA  MODE
372         BNE  CHX1
373         PHA
374         JMP  SRCLG
375 CHX1    CMP  #01
376         BNE  CHX2
377         LDA  (DEPART),Y
378         PHA
379         JMP  SRCLG
380 CHX2    CMP  #02
381         BNE  CHX3
382         LDY  C3
383         LDA  (ARR2),Y
384         PHA
385         LDY  X1
386         JMP  SRCLG
387 CHX3    NOP
388 SRCLG   INY
389         LDA  (DEPART),Y
390         DEY
391         STA  (DEPART),Y
392         INY
393         CPY  X2
394         BNE  SRCLG
395         PLA
396         STA  (DEPART),Y
397         DEX
398         CPX  Y1
399         BNE  DEBLFT2
400         JSR  TEST
401         INC  C3
402         DEC  N
403         LDA  N
404         CMP  #$FF
405         BNE  DEBLFT
406         RTS
407 *
408 * Arret du scrolling par frappe d'une touche
409 *
410 TEST    LDA  KBD
411         BIT  KBDSTRB
412         CMP  #$9B      ; <ESC> interrompt le s
                            crolling
413         BNE  SPACE
414         JMP  NEWSTT    ; retour au programme B
                            ASIC appelant
415 SPACE   CMP  #$A0      ; <ESPACE> fait une pau
                            se
416         BNE  CONT
417         JSR  KBDWAIT   ; attente d'une touche
418 CONT    RTS
```

Table 'TABLES.HGR. OBJ'

Après avoir saisi cette table sous moniteur, vous la sauvegarderez par BSAVE TABLES.HGR.OBJ,A$9000, L$180

```
9000- 00 00 00 00 00 00 00 00
9008- 80 80 80 80 80 80 80 80
9010- 00 00 00 00 00 00 00 00
9018- 80 80 80 80 80 80 80 80
9020- 00 00 00 00 00 00 00 00
9028- 80 80 80 80 80 80 80 80
9030- 00 00 00 00 00 00 00 00
9038- 80 80 80 80 80 80 80 80
9040- 28 28 28 28 28 28 28 28
9048- A8 A8 A8 A8 A8 A8 A8 A8
9050- 28 28 28 28 28 28 28 28
9058- A8 A8 A8 A8 A8 A8 A8 A8
9060- 28 28 28 28 28 28 28 28
9068- A8 A8 A8 A8 A8 A8 A8 A8
9070- 28 28 28 28 28 28 28 28
9078- A8 A8 A8 A8 A8 A8 A8 A8
9080- 50 50 50 50 50 50 50 50
9088- D0 D0 D0 D0 D0 D0 D0 D0
9090- 50 50 50 50 50 50 50 50
9098- D0 D0 D0 D0 D0 D0 D0 D0
90A0- 50 50 50 50 50 50 50 50
90A8- D0 D0 D0 D0 D0 D0 D0 D0
90B0- 50 50 50 50 50 50 50 50
90B8- D0 D0 D0 D0 D0 D0 D0 D0
90C0- 20 24 28 2C 30 34 38 3C
90C8- 20 24 28 2C 30 34 38 3C
90D0- 21 25 29 2D 31 35 39 3D
90D8- 21 25 29 2D 31 35 39 3D
90E0- 22 26 2A 2E 32 36 3A 3E
90E8- 22 26 2A 2E 32 36 3A 3E
90F0- 23 27 2B 2F 33 37 3B 3F
90F8- 23 27 2B 2F 33 37 3B 3F
9100- 20 24 28 2C 30 34 38 3C
9108- 20 24 28 2C 30 34 38 3C
9110- 21 25 29 2D 31 35 39 3D
9118- 21 25 29 2D 31 35 39 3D
9120- 22 26 2A 2E 32 36 3A 3E
9128- 22 26 2A 2E 32 36 3A 3E
9130- 23 27 2B 2F 33 37 3B 3F
9138- 23 27 2B 2F 33 37 3B 3F
9140- 20 24 28 2C 30 34 38 3C
9148- 20 24 28 2C 30 34 38 3C
9150- 21 25 29 2D 31 35 39 3D
9158- 21 25 29 2D 31 35 39 3D
9160- 22 26 2A 2E 32 36 3A 3E
9168- 22 26 2A 2E 32 36 3A 3E
9170- 23 27 2B 2F 33 37 3B 3F
9178- 23 27 2B 2F 33 37 3B 3F
```

Récapitulation 'SCROLLINGS. OBJ0'

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE SCROLLINGS.OBJ0,A$9200, L$2A3.

```
9200- 20 BE DE C9 47 D0 03 4C
9208- AE 93 C9 44 D0 03 4C AE
9210- 93 C9 42 D0 03 4C E7 92
9218- C9 48 D0 03 4C 20 92 60
9220- 20 B1 00 20 F5 E6 86 EF
9228- 20 F5 E6 86 06 20 F5 E6
9230- 86 EB 20 F5 E6 86 ED 20
9238- F5 E6 E8 86 EC 20 F5 E6
9240- 86 EE A9 00 85 08 A9 80
9248- 85 FA A9 91 85 FB A6 EE
9250- BD 00 90 85 1E BD C0 90
9258- 85 1F 20 8E 94 A6 ED 86
9260- 07 BD 00 90 85 1A BD C0
9268- 90 85 1B A4 EB B1 1A 91
9270- FA C8 C4 EC D0 F7 A6 07
9278- BD 00 90 85 1A BD C0 90
9280- 85 1B E8 BD 00 90 85 18
9288- BD C0 90 85 19 A4 EB B1
9290- 18 91 1A C8 C4 EC D0 F7
9298- 86 07 E4 EE D0 D8 A5 EF
92A0- D0 0E A4 EB 91 18 C8 C4
92A8- EC D0 F9 C6 06 D0 AB 60
92B0- C9 02 D0 21 A6 08 BD 00
92B8- 90 85 1C BD C0 90 18 69
92C0- 20 85 1D E6 08 A4 EB B1
92C8- 1C 91 1E C8 C4 EC D0 F7
92D0- C6 06 D0 86 60 A4 EB B1
92D8- FA 91 1E C8 C4 EC D0 F7
92E0- C6 06 F0 02 D0 C7 60 20
92E8- B1 00 20 F5 E6 86 EF 20
92F0- F5 E6 86 06 20 F5 E6 86
92F8- EB 20 F5 E6 86 EE 20 F5
9300- E6 E8 86 EC 20 F5 E6 86
9308- ED A9 BF 85 08 A9 00 85
9310- FA A9 60 85 FB A6 EE BD
9318- 00 90 85 1E BD C0 90 85
9320- 1F 20 8E 94 A6 ED 86 07
9328- BD 00 90 85 1A BD C0 90
9330- 85 1B A4 EB B1 1A 91 FA
9338- C8 C4 EC D0 F7 A6 07 BD
9340- 00 90 85 1A BD C0 90 85
9348- 1B CA BD 00 90 85 18 BD
9350- C0 90 85 19 A4 EB B1 18
9358- 91 1A C8 C4 EC D0 F7 86
9360- 07 E4 EE D0 D8 A5 EF D0
9368- 0E A4 EB 91 18 C8 C4 EC
```

```
9370- D0 F9 C6 06 D0 AB 60 C9
9378- 02 D0 21 A6 08 BD 00 90
9380- 85 1C BD C0 90 18 69 20
9388- 85 1D C6 08 A4 EB B1 1C
9390- 91 1E C8 C4 EC D0 F7 C6
9398- 06 D0 86 60 A4 EB B1 FA
93A0- 91 1E C8 C4 EC D0 F7 C6
93A8- 06 F0 02 D0 C7 60 85 09
93B0- 20 B1 00 20 F5 E6 86 EF
93B8- 20 F5 E6 86 06 20 F5 E6
93C0- 86 EB 20 F5 E6 CA 86 ED
93C8- 20 F5 E6 86 EC 20 F5 E6
93D0- 86 EE A5 09 C9 47 F0 5B
93D8- A2 27 86 08 A6 EE BD 00
93E0- 90 85 18 85 1E BD C0 90
93E8- 85 19 18 69 20 85 1F A4
93F0- EC A5 EF D0 04 48 4C 12
93F8- 94 C9 01 D0 06 B1 18 48
9400- 4C 12 94 C9 02 D0 0A A4
9408- 08 B1 1E 48 A4 EC 4C 12
9410- 94 EA 88 B1 18 C8 91 18
9418- 88 C4 EB D0 F5 68 91 18
9420- CA E4 ED D0 B9 20 8E 94
9428- C6 08 C6 06 A5 06 C9 FF
9430- D0 AA 60 A2 00 86 08 A6
9438- EE BD 00 90 85 18 85 1E
9440- BD C0 90 85 19 18 69 20
9448- 85 1F A4 EB A5 EF D0 04
9450- 48 4C 6D 94 C9 01 D0 06
9458- B1 18 48 4C 6D 94 C9 02
9460- D0 0A A4 08 B1 1E 48 A4
9468- EB 4C 6D 94 EA C8 B1 18
9470- 88 91 18 C8 C4 EC D0 F5
9478- 68 91 18 CA E4 ED D0 B9
9480- 20 8E 94 E6 08 C6 06 A5
9488- 06 C9 FF D0 AA 60 AD 00
9490- C0 2C 10 C0 C9 9B D0 03
9498- 4C D2 D7 C9 A0 D0 03 20
94A0- 88 FB 60
```

Extasie à l'essai

Nous devons ce programme de dessin, édité par Apple, à Pierre Berloquin et Serge Hervy. Il nécessite un Apple //c ou un //e avec carte Féline et souris.

La qualité de l'ergonomie permet de se lancer dans son premier dessin sans trop s'attarder sur la documentation d'ailleurs fort bien présentée. Une barre de menus déroulants, des fenêtres, pilotage à la souris, c'est presque MacPaint !

Les dessins en couleurs (16) et en double haute résolution sont réalisés à partir de fonctions de base : point, trait, rectangle, ellipse, aérographe et texte (à chasse proportionnelle) ; une loupe aide l'utilisateur à travailler au point près sur les 560 que comporte chaque ligne. On dispose également du remplissage à l'aide d'un motif choisi parmi seize, mais cette routine de remplissage – rapide – nous a semblé utiliser une astuce simplifiant plus la tâche du programmeur que celle de l'utilisateur : il faut balayer verticalement la zone à remplir et non simplement 'cliquer' dedans. Pas si simple quand la zone est effilée. Pas de vrais bugs mais la vitesse de tracé des rectangles reste en-deçà de l'extase.

La gestion est confiée à ProDOS, ce qui permet d'utiliser tous les supports mais là, le confort de l'utilisateur pêche un peu sans que ProDOS soit responsable :

- la routine ONLINE n'est pas disponible directement mais seulement en cas d'erreur ;
- en sauvegarde, une erreur est signalée deux fois, une lors de la tentative d'OPEN, une lors du CREATE ;
- on n'accède pas directement aux sous-catalogues lors des sauvegardes/chargements, il faut le préciser dans un menu ;
- dernier point faible : il n'est pas prévu de formater une disquette depuis le programme. ProDOS ne le permet pas mais adjoindre une telle routine à ce programme n'était pas un luxe.

Dans la fenêtre d'erreur subsistent deux nombres : l'un est le code de la fonction MLI appelé, l'autre le code d'erreur : un vestige de debugging ou initiation aux appels MLI ?

Finalités d'un programme de traitements de dessins, la sauvegarde et l'impression. Ici, la première se fera compactée (pour un gain de place non négligeable) ou non et la deuxième en couleurs (sur l'ImageWriterII).

Extasie est certainement un bon logiciel (double haute résolution, impression couleur, simplicité et confort dans les manipulations). On attend toutefois avec impatience une nouvelle version, plus rapide et aux accès disques plus conviviaux.

PicoExpert
Système Expert de degré 0

Yves Martin

Généralités

PICOEXPERT prend en compte deux fichiers. Dans le premier, il lit des assertions qui pourront être vraies ou inconnues. Dans le second, il lit les règles qui régissent ces assertions.

PICOEXPERT permet de faire sur ces assertions (on dira 'faits' dans la suite, même si c'est un abus de langage, car un fait devrait être une assertion vraie) soit de l'induction ou chaînage arrière, soit de la déduction ou chaînage avant.

Dans cet article, nous allons utiliser ce programme pour décrire brièvement comment fonctionne un système expert. Celui-ci est dit de degré 0 car les assertions de base ne contiennent pas de variable. Un système expert de degré 1, avec variables, serait beaucoup plus intéressant, mais c'est déjà moins simple à écrire, et il semble qu'il soit difficile de faire quelque chose qui soit à fois général et efficace : par exemple, un système expert en géométrie, outre qu'il doit savoir faire ce qu'on attend de lui, doit naturellement connaître des notions comme la transitivité, les classes d'équivalence par exemple, ce qui ne sera pas nécessaire pour un système expert en autre chose...

L'utilisation de la récursivité du Pascal et de ses pointeurs est tout à fait efficace ici, c'est presque faire de l'assembleur – car on gère constamment des adresses – avec un langage évolué. Si cet article pouvait arriver à convaincre les "Pascaliens" qui ne s'y sont pas encore mis à travailler les pointeurs...

Structure du programme

Le programme est divisé en différentes parties :

Procédures usuelles de Pom's et initialisations diverses

On remarquera la possibilité du '?' dans le OUI, ceci afin que l'on puisse signifier que l'on ne connaît pas la réponse à une question, alors que le NON pourrait signifier que l'on sait que la chose demandée n'est pas vraie ;

Procédures de manipulation des diverses listes

Ces procédures permettent d'ajouter ou retrancher des éléments d'une liste, ou tout simplement évaluer son contenu. Elles ne posent pas de problème et le débutant en pointeurs en Pascal peut commencer par défricher cette partie ;

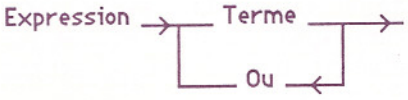
La construction de l'arbre des conditions

Ces procédures permettent de transformer les règles lues dans le fichier de règles en liste d'arbres. La structure retenue ici est la suivante :

- une règle est un enregistrement qui contient : une condition, une action et un pointeur de règle, ou encore les règles sont dans une liste qui contient une condition et une action ;
- une condition est une liste d'arbres ;
- une action est une liste d'entiers ;
- un arbre enfin est composé d'une INFO qui est un entier, et d'un Fil Gauche et d'un Fil Droit qui sont des pointeurs d'arbres.

Précisons tout d'abord que les assertions lues sur la disquette sont placées dans un tableau BF (pour Base de Faits) et qu'ensuite les faits sont repérés par leur indices dans ce tableau, c'est pour cela qu'une action est une liste d'entiers et que INFO d'un arbre est aussi un entier.

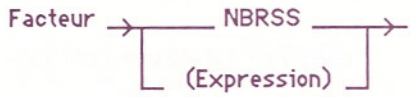
La grammaire pour construire la liste des règles est la grammaire usuelle de l'algèbre ordinaire, on a :



Par exemple :
A + B
A.C + B + D.(E + F + G)
sont deux expressions. Dans la première les deux termes sont A et B et dans la seconde les termes sont A.C puis B, puis D.(E + F +G). Autrement dit, une expression est une suite de termes séparées par des OU (bien sûr ici OU et ET sont respectivement représentés par "+" et ".").

Les TERMES eux, sont une suite de facteurs séparés par des ET (soit des "."). Dans le second exemple ci dessus, A.C est un terme dont les facteurs sont A et C, B est un terme à un seul facteur, et D.(E + F + G) est un terme dont les facteurs sont D d'une part et d'autre part (E + F + G) c'est-à-dire une EXPRESSION entre parenthèse. On retrouve là tout à fait l'algèbre ordinaire, et pour ceux qui

découvrent ce genre de choses ici, on s'aperçoit combien la récursivité est indispensable dans ces problèmes puisque la définition de FACTEUR contenue implicitement dans EXPRESSION contiendra aussi EXPRESSION.



Autrement dit, un facteur sera soit un NBRSS, un nombre sans signe, soit une expression placée entre parenthèses.

NBRSS : c'est une chaîne de caractères qui ne peut utiliser que les chiffres.

La construction proprement dite se fait dans la procédure CONSTRUIRE qui, partant d'une chaîne de caractères renvoit un pointeur d'arbres et un entier qui indique le nombre de feuilles dans toute la "forêt" de conditions ! La forêt en pratique n'ayant que 1 ou 2 arbres d'ailleurs...

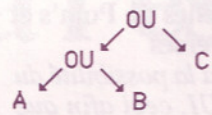
Ce nombre de feuilles devait servir à ordonner les arbres selon le nombre de feuilles décroissant, puis ensuite de pondérer les assertions afin que, les feuilles ayant du poids, on examine les arbres selon le poids d'inconnu que l'on a à son égard, mais cette dernière amélioration, pourtant assez facile à réaliser n'a pas été faite ici, ce serait un bon exercice. Donc dans la procédure CONSTRUIRE, les variables N, N1 et N2 ne considèrent que cette question et n'interfèrent pas la compréhension de la suite.

CONSTRUIRE utilise LICAR et NUCAR. LICAR lit tout simplement un caractère qu'elle place dans la variable CAR et fait pointer la variable NUCAR sur le prochain caractère à lire (les blancs sont sautés, bien sûr). Les procédures EXPRESSION et TERME sont identiques et basées sur l'algorithme suivant – donné pour EXPRESSION :

Pour EXPRESSION (V1 et V2 sont des PARBRES)

- Rechercher le premier TERME, et le mettre dans V1
- Tant qu'on rencontre un OU :
  - chercher le terme suivant, le mettre dans V2
  - construire un nouvel arbre ayant
    pour information : OU
    pour fils gauche V1
    pour fils droit V2
  - mettre dans V1 ce nouvel arbre
- EXPRESSION renverra la valeur de V1

Ainsi A + B + C sera codé ainsi :



même chose pour TERME.

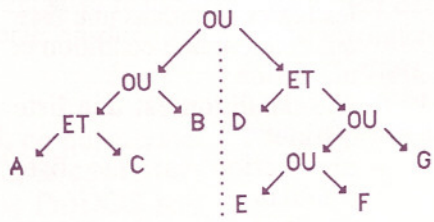
FACTEUR se traite de la même façon :

- Pour FACTEUR
- Chercher si on a affaire à un NBRSS
- Si oui on place ce nombre sans signe dans FACTEUR
- Sinon : Si l'expression commence par une parenthèse
  - alors chercher une EXPRESSION
  - si l'expression se termine par une parenthèse
    alors FACTEUR recoit EXPRESSION
    sinon Erreur ! Il manque ")"
  - sinon Erreur ! Il manque "("

Enfin NBRSS indique si l'on rencontre un nombre sans signe et, si oui, le place dans le INFO du PARBRE correspondant.

Ainsi A.C + B + D.(E + F + G) sera alors structuré comme suit:

Pointeur de l'expression



La condition d'une règle est une suite de Parbre, ceux-ci étant implicitement séparés par des ET. Cette possibilité est proposée dans la syntaxe par une '*' qui fait l'effet d'un "gros ET". Cette possibilité avait été laissée au départ dans l'optique de structurer une réorganisation du parcours des Parbres en fonction du poids des feuilles qui restaient inconnues ; mais ceci n'a pas été fait dans cette version, les feuilles, c'est-à-dire les assertions de départ n'ayant pas été pondérées.

Ainsi par exemple 2 * (3.4 + 5.6) est identique à 2.(3.4 + 5.6), seulement la première version donne deux Parbres pour l'expression alors que la seconde ne donne qu'un seul parbre.

Les procédures de déduction

Il s'agit du chaînage avant d'un système expert : Compte tenu des faits qui sont connus par la machine comme vrais, elle applique les règles qu'elle connaît et en déduit tout ce qu'elle peut.

La procédure DEDUIRE est le lien avec l'utilisateur qui indique quelles sont les assertions vraies. Une fois cela établi, on appelle DEDUIT.

DEDUIT parcours toutes les règles. Pour chaque règle, elle teste si celle-ci est applicable et, si oui, elle vérifie qu'en l'appliquant on obtient bien un fait nouveau. Si une fois au cours de tout ce parcours on obtient au moins un fait nouveau, alors on recommence tout le procédé depuis la première règle puisque la base de faits connus comme vrais s'est agrandie.
DEDUIT utilise donc les fonction booléennes CHAINAVANT, UTILISE, et TESTREGL.

Détaillons TESTREGL qui, elle, utilise la récursivité, ce qui nous préparera au chaînage arrière.
TESTREGL teste tout les Parbres de sa partie 'condition', tous ces arbres doivent être vrais car ils sont séparés par des ET ('*'). Pour cela TESTREGL contient une fonction TESTARB qui elle sera récursive, en effet son algorithme dans ces conditions est très simple :

Pour TESTARB(RAC:parbre)
Selon que le contenu INFO de la RACine est :
ET alors
si le Fils Gauche est vrai TESTARB est la valeur du Fils Droit
sinon TESTARB est faux
OU alors
si le Fils Gauche est faux TESTARB est la valeur du Fils Droit
sinon TESTARB est vrai

Une feuille alors TESTARB est vrai si la feuille est contenue dans la liste des feuilles connues comme vraies.
Là encore, si on débute dans l'approche récursive, cette petite application donne la mesure de la force de la récursivité.

Les procédures d'induction ou chaînage arrière

Cette partie est sans doute, avec la procédure de construction des arbres, la plus intéressante du programme, bien qu'elle soit très simple de compréhension et de réalisation.

Elle contient deux procédures en récursivité croisées, la seconde appelant une procédure elle-même récursive, situation tout à fait naturelle dans ce genre de situation, et donc très simple à écrire dans un langage lui-même récursif.

La fonction PROUVREGLE est vraie si la règle que l'on cherche à appliquer est effectivement applicable. La fonction VERIFIER se contente simplement de vérifier si un fait est vrai. Les deux s'articulent ainsi :

Pour VERIFIER qu'un fait est vrai, il faut :
- tester s'il est un élement de la liste des faits vrais (ici BASE), et si oui, c'est fini ;
- sinon, on parcours la liste de toutes les règles, et pour chaque règle dont ce fait là est une action, c'est-à-dire une conclusion, on est amené à tester si cette règle est applicable (PROUVREGLE) ;
- si le fait n'est dans aucune action, ou qu'aucune règle n'est applicable, alors il nous reste à demander à l'opérateur au clavier s'il sait quelque chose au sujet de ce fait là (aprés s'être assuré qu'on ne lui a pas déjà demandé – d'où une liste de QUESTIONS et une procédure ENLEVE pour enlever les faits connus des QUESTIONS).

Pour PROUVer une REGLE maintenant, il suffit de PROUVer les arbres qu'elle contient. Si tous les ARBres sont vrais, la règle est applicable et on l'applique (utilisation de PLACER).

Pour PROUVer un ARBre (PROUVARB), on procède comme pour le TESTer (TESTARB). Selon l'INFOrmation de la RACine :
– ET : on doit PROUVer les ARBres à la fois FG et FD
– OU : on doit PROUVer l'un des ARBres FG ou FD
– FEUILLE : on doit VERIFIER que cette feuille est bien connue comme vraie (et donc VERIFIER est appelée par PROUVARB qui est appelée par PROUVREGLE qui est appelée par VERIFIER elle-même).

La procédure CONFIRMER fait le lien avec l'opérateur au clavier et demande quelle assertion on veut prouver. Elle cherche alors à VERIFIER cette assertion selon l'algorithme décrit au dessus.

Les procédures de lecture des règles

Ces procédures lisent les fichiers de faits et de règles sur la disquette et construisent les arbres de règles.
LIREREGLE découpe la phrase en une partie CHaine ACTion (CHACT) et une partie CHaine CONDItion (CHCONDI). ATTENTION il est ici implicite que le signe d'implication soit "==>" et qu'il soit précédé d'un blanc.

Ensuite, PLACTION place les actions et PLCONDIT place les conditions, et donc appelle CONSTRUIRE. Remarquons que les actions sont séparées par des '.' et non des '*'. On peut le rajouter si on veut dans PLACTION il suffit alors d'ajouter '*' a l'ensemble des séparateurs ...IF S[I] IN...

Enfin, on a ajouté une procédure de vérification des règles qui est simplement une lecture de l'arbre de la liste des règles. Là encore, dans le cadre d'une découverte de la récursivité sur des arbres en pascal, on peut commencer à défricher ECRIREGLE et ECRIARB qui ne posent aucune difficulté.

Ce programme est accompagné de deux exemples en mathématique. Le premier sur les notions de parallélogramme, rectangle, losange et carré. On peut utiliser celui-ci pour se familiariser avec le fonction de PICOEXPERT.
Le second exemple montre les limites d'un tel progamme dans le cas où on veut rapidement travailler avec des variables et que l'on ne dispose pas au moins de l'équivalence, néanmoins PICOEXPERT est très *compétent* pour tout ce qui est générique, sans variables.

Les améliorations

On peut commencer par chercher à ce que le programme affiche non seulement ce qu'il déduit comme il le fait actuellement, mais qu'il affiche aussi pourquoi il déduit cela, c'est-à-dire qu'il indique, quand il utilise une règle, les faits qu'il utilise et eux seulement bien sûr. Ce serait une procédure qui serait appelée par PLACER par exemple, et elle serait intéressante à faire.

PICOEXPERT peut aussi être transformé en MICROEXPERT ou même MINIEXPERT en améliorant le chaînage arrière. En effet le programme actuel, à des fins didactiques, utilise soit un chaînage avant pur soit un chaînage arrière pur. Or dans la réalité, la démarche analytique, l'induction n'est jamais faite sans considérations de déduction, cela n'a pas de sens. Il faudrait donc repenser complètement

l'algorithme du chaînage arrière pour que de *pico-* le programme devienne *micro-* ou même *mini-* expert.

Pour cela, il y aurait assez peu à faire, encore que cela puisse être assez long. On pourrait dans un premier temps simplifier la grammaire et ne considérer, à la place des arbres, simplement des listes de ET, et dans ce cas on aurait effectivement plusieurs règles ayant la même action. Ensuite donner un coefficient de pondération à chaque assertion. Pour VERIFIER un fait on peut alors faire la liste des règles qui ont ce fait en action. Cette liste sera ordonnée selon la fraction d'inconnu dans ses prémisses :

Par exemple si A.B.C ==> D, que A , B , C soient affectés de a, b, c.

Si par exemple A et B sont connus alors cette règle sera affectée elle du coeficient c / (a+b+c) . Et ainsi les questions posées le seront avec beaucoup plus de pertinence. Et *pico-* deviendra *micro-* . Notons que ce micro-expert serait tout à fait efficace dans nombre d'applications : diagnostic de panne ou diagnostic d'utilisation de réglementations complexes. Pour avoir plus de place mémoire, il n'est pas nécessaire d'avoir le fichier 'faits' en mémoire centrale, il peut rester sur disque ou sur disque virtuel, et donc le tableau BF étant alors supprimé, il y a beaucoup plus de place pour les règles.

On peut chercher à améliorer les possibilités et introduire en plus de l'implication, l'équivalence logique. Ici cela n'a pas été fait car cela n'a d'intérêt d'une part que pour des applications mathématiques, et d'autre part, d'assez peu d'intérêt pour un système sans variable. Néanmoins, la chose est un bon exercice qui ne pose aucun problème et qui pourrait, dans le cadre d'application mathématiques, avoir un réel intéret pédagogique ou être une bonne base pour un projet pédagogique.

Pour cela, il suffit d'avoir un degré de profondeur de plus dans les procédures PROUV (règle ou arbre) et bien sûr un drapeau qui pointe sur la liste des équivalences que l'on est en train de tester pour ne pas boucler sur les équivalences. Ce degré de profondeur ne demande que la construction d'une autre procédure PROUV (PROUVGENE) et d'une procédure TESTEQU qui testera sur les équivalences une assertion donnée, moyennant de ne pas essayer une liste d'équivalences... dont peut être issu cet appel .
Et *micro-* deviendra *mini-*.

Bien sûr, cette partie sera grandement enrichie, mais cela devient moins simple, si il n'y a plus une base de faits vrais uniquement, mais aussi une base de faits faux (CONSTRUIRE se modifie facilement en mettant des nombres négatifs, cette partie ne présente pas de réelle difficulté) et que, c'est là qu'est la difficulté, le programme s'il rencontre A <==> B et qu'il sache que A est faux en conclura que B aussi, avec de nombreuses variantes sur les contraposées logiques, par exemple si A ou B ==> C
et que l'on sait que A est faux et que C aussi, on place automatiquement que B l'est aussi. Autrement dit, un programme qui, outre ses règles sur fichier, contiendra ses propres *méta-règles.*

La difficulté n'est pas tant dans l'écriture, mais surtout dans le fait que le chaînage arrière comme on l'a décrit ci-dessus ne serait plus alors aussi pertinent ; en effet dans ce cas, la récursivité si précieuse auparavant devrait pouvoir être court-circuitée ici a tout moment sous peine de poser à l'écran des questions qui ne seraient plus d'actualité. Il faut donc repenser alors complètement l'algorithme du chaînage arrière, pour qu'il ne pose que des questions pertinentes...
Et alors *mini-* deviendra vraiment expert.

Comme on le voit dans ces améliorations, travailler sur un système expert, même d'ordre zéro, peut être un travail vraiment passionnant puisqu'il amène à réfléchir sur la 'démarche intelligente' que l'on souhaite donner à l'analyse d'un problème, et les variantes sont nombreuses chacun programmant un peu sa propre façon d'analyser un problème...

NDLR : Le programme comprend 1.2+3.4 comme (1.2)+(3.4) et 1+2.3+4 comme 1+(2.3)+4. Il est donc conseillé de mettre des parenthèses dès qu'on a le moindre doute.

Éviter les règles qui se 'mordent la queue', surtout quand on a fait un chaînage arrière, car le programme peut alors boucler sans autre issue que le RESET. Par exemple :
(1.2) ==> (3)
(3) ==> (2)
Si l'on cherche à confirmer (2) on court au problème.

Si les règles sont incorrectement écrites, le programme ne s'en aperçoit pas toujours et peut planter *à la lecture du fichier de règles (erreur d'exécution ou bouclage).*

Ce système expert est très rudimentaire (par comparaison car il est déjà complexe), aussi, il ne sait interpréter que des assertions vraies :
Ainsi, dans le cas de :
(1.2) ==> (3)
(3) ==> (4)
si l'on veut confirmer (4), il demandera successivement :
1 vrai ?
2 vrai ?
3 vrai ?
4 vrai ?
Même si l'on affirme que 1 est faux, il demandera (car il ne sait pas le déduire) si 3 et 4 sont vrais.

Les fichiers

Le fichier GEOMETRIE s'intéresse aux parallélogrammes (#), rectanges, losanges, et carrés.

Le second fichier est une démonstration de base du theoreme de THALES, pour cela le programme est efficace seulement ou presque avec les hypotèses 1,5,6,13 et 10, c'est-à-dire : Soit ABC un triangle I milieu de AB, IJ//BC et J sur AC et K de AB tel que JK//AB. Alors le théorème dit que J est milieu de AC (et K milieu de BC).

Sur ce fichier on n'a pas pu mettre beaucoup d'autres règles car elles auraient fait boucler le programme . On voit ici que, même sans variables, avec des équivalences, on arriverait a d'excellents résultats.

Fichier FAITHALES.TEXT

```
I milieu de AB
J milieu de AC
K milieu de BC
I appartient AB
J appartient AC
K appartient BC
IA et IB meme longueur
IA et JC meme longueur
KB et KC meme longueur
JK  //  AB
JK  //  IA
JK  //  IB
IJ  //  BC
IJ  //  KB
IJ  //  KC
IK  //  AC
IK  //  JA
IK  //  JC
IJ et KB meme longueur
IJ et KC meme longueur
JK et IA meme longueur
JK et IB meme longueur
IK et JA meme longueur
IK et JC meme longueur
IJKB parallelogramme
IJCK parallelogramme
JKIA parallelogramme
```

Fichier REGLTHALES.TEXT

```
(14.12) ==> (25)
(25) ==> (22.19)
(1) ==>(4.7)
(7.22) ==> (21)
(11.21) ==> (27)
(13.6) ==> (14.15)
(10.4) ==> (11.12)
(27.5) ==> (23.17.18)
(15.18) ==> (26)
(26) ==> (24.20)
(23.24) ==> (8)
(8.5) ==> (2)
(20.19) ==> (9)
(9.6) ==> (3)
```

Fichier GEOMETRIE.TEXT

```
 1  On est dans le plan
 2  On a quatre points
    A,B,C,D
 3  (ABCD) est un
    quadrilatere
 4  Les cotes opposes sont deux
    a deux paralleles
 5  Deux cotes opposes sont
    paralleles et de meme
    longueur
 6  Les diagonales se coupent
    en leurs milieux
 7  (ABCD) est un #
 8  Les diagonales ont meme
    longueur
 9  Deux cotes sont
    orthogonaux
10  (ABCD) est un rectangle
11  Deux cotes consecutifs ont
    meme longueur
12  Les diagonales sont
    orthogonales
13  (ABCD) est un losange
14  (ABCD) est un carre
```

Fichier REGLEGEOM.TEXT

```
(1)
(2)
(4+5+6)*(3) ==> (7)
(1.2) ==> (3)
(7.(8+9)) ==> (10)
(7)*(11+12) ==> (13)
(10.13) ==> (14)
```

Programme 'PICOEXPERT'

Le symbole '⌡' indique la continuité de la ligne.

```
PROGRAM PICOEXPERT;  (* ---- Avec arbres de cond⌡
itions ET et OU ----  *)
CONST NMAX=55; ET=1000; OU=1010;
TYPE CHOIDECA=SET OF CHAR;
     PCELL=^CELL;
     PLISTE=^LISTE;
     PREGLE=^REGLE;
     PARBRE=^ARBRE;

     REGLE=RECORD
             CONDIT:PCELL;
             ACTION:PLISTE;
             SUIV:PREGLE;
           END;

     LISTE=RECORD
             ADRI:INTEGER;
             SUIV:PLISTE;
           END;

     CELL=RECORD
             ADRC:PARBRE;
             SUIV:PCELL;
           END;

     ARBRE=RECORD
             INFO:INTEGER;
             FG:PARBRE;
             FD:PARBRE;
           END;

VAR BASE,QUESTIONS,BASEREF,QUESTREF:PLISTE;
    REGLES:PREGLE;
    BF:ARRAY[1..NMAX] OF STRING;
    NOMFI2,NOMFI:STRING;
    INV,NORM,HOME,BS,EFL,EFB,SON,CR,CI,CO:CHAR;
    NBFAITS:INTEGER;
    FICH:TEXT;


(* ------- PROCEDURES USUELLES DE POM'S -------- *)

PROCEDURE PRENINFO;
   BEGIN HOME:=CHR(12);EFB:=CHR(11);EFL:=CHR(29);B⌡
   S:=CHR(8);
         CR:=CHR(13);SON:=CHR(7);INV:=CHR(15);NORM⌡
         :=CHR(14);
   END;

PROCEDURE MESS(U,V:INTEGER;S:STRING;K:INTEGER);
BEGIN GOTOXY(U,V);
CASE K OF
 0 : WRITE(S);
 1 : WRITE(S,EFL);
 2 : WRITE(S,EFB);
 3 : WRITE(S,SON);
 4 : WRITE(S,EFL,SON);
 5 : WRITE(S,EFB,SON);
 6 : WRITE(INV,S,NORM);
 7 : WRITE(INV,S,NORM,EFL);
 8 : WRITE(INV,S,NORM,EFB);
 9 : WRITE(INV,S,NORM,SON);
10 : WRITE(INV,S,NORM,EFL,SON);
11 : WRITE(INV,S,NORM,EFB,SON);
END;
END;

PROCEDURE MERR(S:STRING);
BEGIN MESS(0,23,S,4);END;

PROCEDURE EFMERR;
BEGIN MESS(0,23,' ',1);END;

PROCEDURE PRENRETURN;
   VAR SORT : CHAR;
   BEGIN GOTOXY(0,23);WRITE('''Return'' pour conti⌡
   nuer  ');
     REPEAT
        READ(KEYBOARD,SORT);
     UNTIL EOLN(KEYBOARD);
   EFMERR;
   END;

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

FUNCTION OUI:BOOLEAN;
   BEGIN
   OUI:=PRENCAR(['O','N','o','n','?']) IN ['O','o'⌡
   ];
   WRITELN;
   END;
PROCEDURE PRENENTIER(LONGMAX:INTEGER;VAR S:INTEGER⌡
);
   VAR S1 : STRING[1];
       I  : INTEGER;
       CONT : STRING;
       ENSCHIF :CHOIDECA;
   BEGIN
     ENSCHIF:=['0'..'9'];S1:=' ';CONT:='';
     REPEAT
       IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(ENSCH⌡
       IF+[CR])
          ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:⌡
          =PRENCAR([CR,BS])
                ELSE S1[1]:=PRENCAR(ENSCHIF+[CR,BS]⌡
                );
       IF S1[1] IN ENSCHIF THEN CONT:=CONCAT(CONT,⌡
       S1)
           ELSE IF S1[1]=BS THEN BEGIN
                                 WRITE(BS,' ',BS);
                                 DELETE(CONT,LENGT⌡
                                 H(CONT),1);
                                 END;
     UNTIL S1[1]=CR;
     IF LENGTH(CONT)<>0 THEN BEGIN
                             S:=0;
                             FOR I:=1 TO LENGTH(CO⌡
                             NT) DO
                             BEGIN
                             S:=S*10;S:=S+(ORD(CON⌡
```

```
                                T[I])-ORD('0'));
                                END;
                                END
                            ELSE WRITE(S);
  END;

PROCEDURE PRENCHAINE(LONGMAX:INTEGER;BONENS:CHOIDE⌋
CA;VAR S:STRING);
VAR S1   :STRING[1];
    CONT :STRING;

BEGIN
  S1:=' ';CONT:='';
  REPEAT
    IF LENGTH(CONT)=0 THEN S1[1]:=PRENCAR(BONENS+[⌋
    CR])
      ELSE IF LENGTH(CONT)=LONGMAX THEN S1[1]:=PRE⌋
      NCAR([CR,BS])
              ELSE S1[1]:=PRENCAR(BONENS+[CR,BS]);
   IF S1[1] IN BONENS 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;

(* -------- INITIALISATIONS DIVERSES ----------- *)

PROCEDURE LIRE(VAR LN:STRING);
BEGIN READLN(FICH,LN);
WHILE (NOT EOF(FICH)) AND ((LN='') OR (LN=' ')) DO⌋
READLN(FICH,LN);
END;

PROCEDURE LIREREGLE;FORWARD;

PROCEDURE ECRITEST;FORWARD;

PROCEDURE INIT;
CONST DIRERR='Mal orthographié ou mauvaise disquet⌋
te';
VAR LN:STRING;
    VRAI:BOOLEAN;
BEGIN
PRENINFO;
BASE:=NIL;REGLES:=NIL;QUESTIONS:=NIL;NBFAITS:=0;
WRITE(HOME);MESS(30,1,' SYSTEME EXPERT ',6);VRAI:=⌋
FALSE;
(*$I-*)
MESS(0,10,'Le fichier de faits sera en drive 2',1);
MESS(0,12,'Indiquer son nom sans le préfixe de dri⌋
ve',1);
MESS(0,13,'et sans le suffixe ".TEXT"',1);
REPEAT
MESS(0,18,'Nom du fichier de faits :   ',1);
PRENCHAINE(20,[' '..'z'],NOMFI2);EFMERR;
NOMFI2:=CONCAT('#5:',NOMFI2,'.TEXT');
RESET(FICH,NOMFI2);
VRAI:=IORESULT=0;
IF NOT VRAI THEN MERR(DIRERR)
            ELSE
WHILE NOT EOF(FICH) DO BEGIN LIRE(LN);
            IF ((LN<>'') AND (LN<>' ')) THEN BEGIN
                              NBFAITS:=NBFAITS+1;BF[⌋
                              NBFAITS]:=LN;
                              WRITELN(NBFAITS,'  -->⌋
                              ',BF[NBFAITS]);
                                                END;

            END;
CLOSE(FICH);
UNTIL VRAI;
MESS(0,3,' ',2);
VRAI:=FALSE;
MESS(0,10,'Le fichier de régles sera aussi en driv⌋
e 2',1);
MESS(0,13,'sans préfixe ni suffixe',1);
REPEAT
MESS(0,18,'Nom du fichier de régles :   ',1);
PRENCHAINE(20,[' '..'z'],NOMFI);EFMERR;
NOMFI:=CONCAT('#5:',NOMFI,'.TEXT');
RESET(FICH,NOMFI);
VRAI:=IORESULT=0;
IF NOT VRAI THEN MERR(DIRERR)
            ELSE WHILE NOT EOF(FICH) DO LIREREGLE;
CLOSE(FICH);
UNTIL VRAI;
(*$I+*)
WRITE(HOME,'Vérification des régles (O/N) ?  ');
IF OUI THEN BEGIN WRITELN;WRITELN;ECRITEST;PRENRET⌋
URN;END;
END;

(*  - Procedures de manipulation des diverses list⌋
es -  *)

FUNCTION MEMBRELIST(I:INTEGER;L:PLISTE):BOOLEAN;
BEGIN
IF L=NIL THEN MEMBRELIST:=FALSE
         ELSE BEGIN
              WHILE (L<>NIL) AND (I<>L^.ADRI) DO L⌋
              :=L^.SUIV;
              MEMBRELIST:=NOT(L=NIL);
              END;
END;

FUNCTION MEMBRECELL(A:PARBRE;C:PCELL):BOOLEAN;
BEGIN
IF C=NIL THEN MEMBRECELL:=FALSE
         ELSE BEGIN
              WHILE (C<>NIL) AND (A<>C^.ADRC) DO C⌋
              :=C^.SUIV;
              MEMBRECELL:=NOT(C=NIL);
              END;
END;

FUNCTION AJOUTLIST(I:INTEGER;L:PLISTE):PLISTE;
VAR P:PLISTE;
BEGIN NEW(P);P^.ADRI:=I;P^.SUIV:=L;AJOUTLIST:=P;EN⌋
D;

FUNCTION AJOUTCELL(A:PARBRE;C:PCELL):PCELL;
VAR P:PCELL;
BEGIN NEW(P);P^.ADRC:=A;P^.SUIV:=C;AJOUTCELL:=P;EN⌋
D;

FUNCTION AJOUTREGLE(C:PCELL;A:PLISTE;R:PREGLE):PRE⌋
GLE;
VAR P:PREGLE;
BEGIN NEW(P);P^.CONDIT:=C;P^.ACTION:=A;P^.SUIV:=R;⌋
AJOUTREGLE:=P;END;

FUNCTION EXISTE(I:INTEGER):BOOLEAN;
VAR LSTB:PLISTE;
BEGIN LSTB:=BASE;
WHILE (LSTB<>NIL) AND (I<>LSTB^.ADRI) DO LSTB:=LST⌋
B^.SUIV;
EXISTE:=NOT(LSTB=NIL);
END;

FUNCTION PLACER(I:INTEGER;AFFICH:CHAR):BOOLEAN;
VAR S:STRING;
BEGIN
IF AFFICH='D' THEN S:='On en deduit '
              ELSE S:='On place le fait ';
IF EXISTE(I) THEN PLACER:=FALSE
             ELSE BEGIN BASE:=AJOUTLIST(I,BASE);PL⌋
             ACER:=TRUE;WRITELN;
                  WRITELN(S,INV,' ',BF[I],' ',NORM⌋
                  );WRITELN;
                  END;
END;

PROCEDURE ENLEVE(I:INTEGER;VAR X:PLISTE);
VAR Y,YPREC:PLISTE;
    PASVU:BOOLEAN;
BEGIN Y:=X;YPREC:=X;PASVU:=TRUE;
WHILE (Y<>NIL) AND PASVU DO BEGIN
      IF I=Y^.ADRI THEN IF Y=X THEN BEGIN X:=X^.S⌋
      UIV;PASVU:=FALSE;END
                        ELSE BEGIN PASVU:=FALSE;YP⌋
                        REC^.SUIV:=Y^.SUIV;
                                        END;
                       YPREC:=Y;Y:=Y^.SUIV;
                       END;
END;

(* --- Les procedures de construction de l'arbre d⌋
e conditions --- *)

PROCEDURE CONSTRUIRE(DEPART:STRING;VAR EXPR:PARBRE;
VAR N:INTEGER);
CONST PASNOMBRE=990;
VAR NUCAR :INTEGER;
    CAR :CHAR;

PROCEDURE LICAR;
BEGIN
IF NUCAR<=LENGTH(DEPART) THEN BEGIN CAR:=DEPART[NU⌋
CAR];NUCAR:=NUCAR+1;
                                IF CAR=' ' THEN LICAR;
                                END;
END;

PROCEDURE EXPRESSION(VAR EXPR:PARBRE;VAR N:INTEGER⌋
);FORWARD;

PROCEDURE NBRSS(VAR PNB:PARBRE;VAR N:INTEGER);
VAR ENS:SET OF '0'..'9';
    V:PARBRE;
    P,FEUILLE:INTEGER;
BEGIN ENS:=['0'..'9'];FEUILLE:=0;P:=NUCAR;
WHILE CAR IN ENS DO BEGIN FEUILLE:=10*FEUILLE+ORD(⌋
CAR)-ORD('0');
                     LICAR;END;
NEW(V);V^.FG:=NIL;V^.FD:=NIL;
IF NUCAR=P THEN V^.INFO:=PASNOMBRE
           ELSE BEGIN V^.INFO:=FEUILLE;N:=N+1;END;
PNB:=V;
END;

PROCEDURE MERR(I:INTEGER);
BEGIN WRITELN(SON);
CASE I OF
 1: WRITELN('Erreur de syntaxe,il peut manquer une⌋
 ")" .');
 2: WRITELN('Erreur de syntaxe,il peut manquer une⌋
 "(" .');
END;
WRITELN(SON);
END;

PROCEDURE FACTEUR(VAR FACT:PARBRE;VAR N:INTEGER);
VAR V1:PARBRE;
    N1:INTEGER;
BEGIN  N1:=0;
NBRSS(V1,N1);
IF (V1^.INFO<>PASNOMBRE) THEN BEGIN FACT:=V1;N:=N1⌋
;END
                ELSE BEGIN
                      IF CAR='(' THEN BEGIN LICAR;⌋
                      EXPRESSION(V1,N1);
                           IF CAR=')' THEN BEGIN L⌋
                           ICAR;
                                   FACT:=V1;N:=N1;
                              END ELSE MERR(1);
                        END ELSE MERR(2);
                      END;
END;

PROCEDURE TERME(VAR TER:PARBRE;VAR N:INTEGER);
VAR N1,N2:INTEGER;
    V1,V2,V:PARBRE;
BEGIN N1:=0;N2:=0;
FACTEUR(V1,N1);
WHILE (CAR='.') DO BEGIN LICAR;FACTEUR(V2,N2);NEW(⌋
V);V^.INFO:=ET;
                V^.FG:=V1;V^.FD:=V2;N1:=N1+N2;V1:=V;
                END;
TER:=V1;N:=N1;
END;

PROCEDURE EXPRESSION;
VAR N1,N2:INTEGER;
    V1,V2,V:PARBRE;
BEGIN N1:=0;N2:=0;
TERME(V1,N1);
WHILE (CAR='+') DO BEGIN LICAR;TERME(V2,N2);NEW(V)⌋
;V^.INFO:=OU;
                V^.FG:=V1;V^.FD:=V2;N1:=N1+N2;V1:=V;
                END;
EXPR:=V1;N:=N1;
END;

BEGIN (* construire *)
NUCAR:=1;EXPR:=NIL;N:=0;
LICAR;EXPRESSION(EXPR,N);
END;(* fin ; appel de plcondit *)

(* ------ La deduction : chainage avant -------- *)

FUNCTION TESTREGL(REGLE:PREGLE):BOOLEAN;
VAR X:PCELL;
    VRAI:BOOLEAN;

  FUNCTION TESTARB(RAC:PARBRE):BOOLEAN;
  VAR TEST:BOOLEAN;
      INFORAC:INTEGER;
  BEGIN INFORAC:=RAC^.INFO;
  CASE INFORAC OF
    ET : IF TESTARB(RAC^.FG) THEN TEST:=TESTARB(RA⌋
    C^.FD)
                                ELSE TEST:=FALSE;
    OU : IF NOT(TESTARB(RAC^.FG)) THEN TEST:=TESTA⌋
    RB(RAC^.FD)
                                  ELSE TEST:=TRUE;
    END;
  IF INFORAC IN [1..NBFAITS] THEN TEST:=MEMBRELIST⌋
  (INFORAC,BASE);
  TESTARB:=TEST;
  END;

BEGIN X:=REGLE^.CONDIT;VRAI:=TRUE;
WHILE (X<>NIL) AND VRAI DO IF TESTARB(X^.ADRC) THE⌋
N X:=X^.SUIV
                                  ELSE VRAI:=FALSE;
TESTREGL:=VRAI;
END; (* TESTREGL *)

FUNCTION UTILISE(REGLE:PREGLE):BOOLEAN;
VAR CONCLUSION:PLISTE;
    VRAI:BOOLEAN;
BEGIN CONCLUSION:=REGLE^.ACTION;VRAI:=FALSE;
WHILE (CONCLUSION<>NIL) DO BEGIN
                IF PLACER(CONCLUSION^.ADRI,'D'⌋
                )THEN VRAI:=TRUE;
                CONCLUSION:=CONCLUSION^.SUIV;
                END;
UTILISE:=VRAI;
END;

PROCEDURE DEDUIT;
VAR CONTINUER:BOOLEAN;

  FUNCTION CHAINAVANT:BOOLEAN;
  VAR LSTR:PREGLE;
      VRAI:BOOLEAN;
  BEGIN LSTR:=REGLES;VRAI:=FALSE;
  WHILE (LSTR<>NIL) DO BEGIN
                  IF TESTREGL(LSTR) THEN IF U⌋
                  TILISE(LSTR) THEN VRAI:=TRUE;
                  LSTR:=LSTR^.SUIV;
                  END;
  CHAINAVANT:=VRAI;
  END;

BEGIN
REPEAT CONTINUER:=CHAINAVANT;UNTIL NOT(CONTINUER);
END;

PROCEDURE DEDUIRE;
```

```
VAR HYPOTHESE,I:INTEGER;

  FUNCTION POSSIBLE:BOOLEAN;
  VAR OK:BOOLEAN;
  BEGIN
  OK:= (HYPOTHESE IN [0..NBFAITS]);
  IF NOT OK THEN WRITE(SON);
  IF HYPOTHESE<10 THEN WRITE(BS,' ',BS)
                  ELSE WRITE(BS,BS,'  ',BS,BS);
  POSSIBLE:=OK;
  END;

BEGIN
WRITE(HOME);
MESS(0,3,'Donner vos hypothèses sous forme d''enti⌋
ers (0 pour quitter) .',0);
WRITELN;WRITELN;
REPEAT PRENENTIER(2,HYPOTHESE);UNTIL POSSIBLE;
IF HYPOTHESE<>0 THEN BEGIN I:=1;WRITELN(I:2,' : ',⌋
BF[HYPOTHESE]);END;
WHILE (HYPOTHESE<>0) DO BEGIN BASE:=AJOUTLIST(HYPO⌋
THESE,BASE);WRITELN;
              REPEAT PRENENTIER(2,HYPOTHESE);UNT⌋
              IL POSSIBLE;
              IF HYPOTHESE<>0 THEN BEGIN I:=I+1;
                                WRITELN(I:2,' : ',BF⌋
                                [HYPOTHESE]);
                                END;
              END;
DEDUIT;
END;

(* ------ L' induction : chainage arriere ------ *)

FUNCTION PROUVREGLE(LSTR:PREGLE):BOOLEAN;FORWARD;

FUNCTION VERIFIER(FAIT:INTEGER):BOOLEAN;
VAR LSTR:PREGLE;
    FAUX,I:BOOLEAN;
    C:CHAR;
BEGIN FAUX:=TRUE;LSTR:=REGLES;
IF MEMBRELIST(FAIT,BASE) THEN FAUX:=FALSE
                 ELSE WHILE (LSTR<>NIL) AND FAUX DO
IF MEMBRELIST(FAIT,LSTR^.ACTION) THEN IF PROUVREGL⌋
(LSTR) THEN FAUX:=FALSE
                              ELSE LSTR:=LSTR^.SUIV
                        ELSE LSTR:=LSTR^.SUIV;

IF FAUX AND (NOT(MEMBRELIST(FAIT,QUESTIONS))) THEN⌋
BEGIN WRITELN;
            WRITE('le fait ',INV,' ',BF[FAIT],' '⌋
            ,NORM,' est-il vrai ?  ');
            IF OUI THEN BEGIN WRITELN;I:=PLACER(F⌋
            AIT,'C');FAUX:=FALSE;
               END ELSE QUESTIONS:=AJOUTLIST(FAIT⌋
               ,QUESTIONS);

END;
VERIFIER:=NOT FAUX;
END;

FUNCTION PROUVARB(RAC:PARBRE):BOOLEAN;
VAR INFORAC:INTEGER;
    BRAC :BOOLEAN;
BEGIN
INFORAC:=RAC^.INFO;
CASE INFORAC OF
  ET : IF PROUVARB(RAC^.FG) THEN BRAC:=PROUVARB(RA⌋
  C^.FD)
                            ELSE BRAC:=FALSE;
  OU : IF NOT(PROUVARB(RAC^.FG)) THEN BRAC:=PROUVA⌋
  RB(RAC^.FD)
                            ELSE BRAC:=TRUE;
  END;
IF INFORAC IN [1..NBFAITS] THEN  BRAC:=VERIFIER(IN⌋
FORAC);
PROUVARB:=BRAC;
END;

FUNCTION PROUVREGL;
VAR LSTCAS:PCELL;
    VRAI,I:BOOLEAN;
    LACT:PLISTE;
BEGIN LSTCAS:=LSTR^.CONDIT;VRAI:=TRUE;
WHILE (LSTCAS<>NIL) AND VRAI DO
          IF PROUVARB(LSTCAS^.ADRC) THEN LSTCAS:=⌋
          LSTCAS^.SUIV
                            ELSE VRAI:=FALSE;
PROUVREGLE:=VRAI;
IF VRAI THEN BEGIN LACT:=LSTR^.ACTION;
            WHILE (LACT<>NIL) DO BEGIN I:=PLACER(⌋
            LACT^.ADRI,'D');
                    ENLEVE(LACT^.ADRI,QUESTIONS);
                    LACT:=LACT^.SUIV;
                    END;
            END;
END;

PROCEDURE CONFIRMER;
VAR I:INTEGER;
BEGIN WRITE(HOME);
REPEAT
 GOTOXY(0,3);
 WRITE('Que voulez-vous démontrer   (1..',NBFAITS,⌋
 ')  ?  ');PRENENTIER(2,I);
 IF NOT(I IN [1..NBFAITS]) THEN BEGIN WRITE(SON);
                IF I>9 THEN WRITE(BS,BS,'  ',BS,BS)
                            ELSE WRITE(BS,' ',BS);
                END;
UNTIL I IN [1..NBFAITS];
WRITE(HOME);WRITELN('A MONTRER : ',BF[I]);
IF VERIFIER(I) THEN BEGIN WRITELN;
          WRITE('L''hypothèse  ',INV,' ',BF[I]);
          WRITELN(' ',NORM,'  a été démontrée .');
       END ELSE BEGIN WRITELN;
          WRITELN(SON,'Hypothèse non prouvée');
          END;
END;
```

Un 'bug' dans le "carnet d'adresses" du numéro 27.

Les utilisateurs d'Apple //e *étendus* ont certainement constaté que, sur leur machine, le programme "Carnet d'adresses" publié dans le numéro 27 de Pom's un petit défaut : le composeur téléphonique ne fonctionne pas. Pour pallier cela, la variable IC de la ligne 10240 doit être mise à zéro.

Un grand merci à Monsieur Guyot - Sainte Eulalie en Born - 40200 Landes, qui nous a indiqué ce défaut.

```
(* ---- les procedure de lecture des regles ---- *)

PROCEDURE PLCONDIT(S:STRING;VAR LST:PCELL);
VAR LG,I,J,N :INTEGER;
    NPROD,FAUX:BOOLEAN;
    DEPART:STRING;
    EXPR :PARBRE;
    DJ,U,X :PCELL;
    V,Y :PLISTE;
BEGIN LG:=LENGTH(S);LST:=NIL;X:=NIL;Y:=NIL;I:=1;J:⌋
=1;NPROD:=TRUE;
REPEAT
  WHILE (I<=LG) AND NPROD DO IF S[I]='*' THEN NPRO⌋
  D:=FALSE
                                          ELSE I:=I+1;
  DEPART:=COPY(S,J,I-J);CONSTRUIRE(DEPART,EXPR,N);
  IF (NOT(NPROD))OR(J<>1) THEN BEGIN X:=AJOUTCELL(⌋
  EXPR,X);Y:=AJOUTLIST(N,Y);
                              J:=I+1;I:=I+1;NPROD:=TRUE;
                     END ELSE LST:=AJOUTCELL(EXPR,LST);⌋
                     (* On place  la sol *)
UNTIL (I>LG);
IF X<>NIL THEN BEGIN DJ:=NIL;
         REPEAT V:=Y;U:=X;N:=0;
         WHILE (U<>NIL) DO IF MEMBRECELL(U^.ADRC,D⌋
         J) THEN BEGIN
                                          V:=V^.SUIV;
                                          U:=U^.SUIV;
                                       END ELSE BEGIN
                              IF V^.ADRI>N THEN N:=V^.ADRI;
                                     V:=V^.SUIV;U:=U^.SUIV;
                                          END;
         IF N<>0 THEN BEGIN U:=X;V:=Y;FAUX:=TRUE;
                        WHILE (U<>NIL) AND FAUX DO
              IF MEMBRECELL(U^.ADRC,DJ) THEN BEGIN ⌋
              U:=U^.SUIV;
                                          V:=V^.SUIV;END
                                       ELSE
                              IF V^.ADRI=N THEN FAUX:=FALSE
                                       ELSE BEGIN
                                U:=U^.SUIV;V:=V^.SUIV;END;

              IF NOT FAUX THEN DJ:=AJOUTCELL(U^.A⌋
              DRC,DJ);
              END;
         UNTIL N=0;
         LST:=DJ;
         END;
END;

PROCEDURE PLACTION(S:STRING;VAR ACT:PLISTE);
VAR ENS:SET OF '0'..'9';
    L:PLISTE;
 LG,N,I:INTEGER;
BEGIN L:=NIL;ENS:=['0'..'9'];LG:=LENGTH(S);I:=1;
WHILE (I<LG) DO IF S[I] IN [' ','.','(',')'] THEN ⌋
I:=I+1
                                          ELSE BEGIN N:=0;
            WHILE S[I] IN ENS DO BEGIN N:=10*N+OR⌋
            D(S[I])-ORD('0');I:=I+1;END;
            IF (N<>0) THEN L:=AJOUTLIST(N,L)
                      ELSE WRITELN('Erreur d''ecr⌋
                      iture dans ',S,CHR(7));
                                          END;
ACT:=L;
END;

PROCEDURE LIREREGLE;
VAR CHACT,CHCONDI,LN:STRING;
    FAUX:BOOLEAN;
    I :INTEGER;
    LST:PCELL;
    ACT,PL:PLISTE;
BEGIN LIRE(LN);FAUX:=TRUE;I:=LENGTH(LN);WRITELN('-⌋
-> ',LN);
WHILE (I>0) AND FAUX DO IF LN[I]='>' THEN FAUX:=FA⌋
LSE
                                          ELSE I:=I-1;
IF I>0 THEN BEGIN CHACT:=COPY(LN,I+1,LENGTH(LN)-I);
          CHCONDI:=COPY(LN,1,I-4);
          (* on peut ajouter un WHILE qui enleve⌋
          les blancs finaux *)
   END ELSE CHACT:=LN;
PLACTION(CHACT,ACT);
IF I=0 THEN BEGIN PL:=ACT;
          WHILE (PL<>NIL) DO BEGIN BASE:=AJOUTLI⌋
          ST(PL^.ADRI,BASE);
                                     PL:=PL^.SUIV;END;
   END ELSE BEGIN PLCONDIT(CHCONDI,LST);
          REGLES:=AJOUTREGL(LST,ACT,REGLES);
          END;
END;
(* -- Procedure de verification de construction de⌋
s arbres --- *)

PROCEDURE ECRITEST;
VAR R:PREGLE;

PROCEDURE ECRIARB(RAC:PARBRE);
BEGIN
IF RAC<>NIL THEN
 IF RAC^.INFO=ET THEN BEGIN
        IF RAC^.FG^.INFO=OU THEN BEGIN WRITE('(');
                                 ECRIARB(RAC^.FG)⌋
                                 ;WRITE(').');
                       END ELSE BEGIN ECRIARB(RA⌋
                       C^.FG);
                                 WRITE('.');END;
        IF RAC^.FD^.INFO=OU THEN BEGIN WRITE('(');
                                 ECRIARB(RAC^.FD)⌋
                                 ;WRITE(')');
                       END ELSE BEGIN ECRIARB(RA⌋
                       C^.FD);END;
        END
    ELSE BEGIN ECRIARB(RAC^.FG);
        IF RAC^.INFO=OU THEN WRITE('+')
                       ELSE WRITE(RAC^.INFO);
        ECRIARB(RAC^.FD);
        END;
END;

PROCEDURE ECRIREGLE(R:PREGLE);
VAR X:PCELL;
    Y:PLISTE;
BEGIN X:=R^.CONDIT;
WHILE (X<>NIL) DO BEGIN WRITE('(');ECRIARB(X^.ADRC⌋
);WRITE(')');
          X:=X^.SUIV;IF (X<>NIL) THEN WRITE('*');
          END;
WRITE(' ==> (');Y:=R^.ACTION;
WHILE (Y<>NIL) DO BEGIN WRITE(Y^.ADRI);Y:=Y^.SUIV;
                IF (Y<>NIL) THEN WRITE('.');
                END;
WRITELN(')');WRITELN;
END;

BEGIN (* ecritest *)
R:=REGLES;
WHILE (R<>NIL) DO BEGIN ECRIREGLE(R);R:=R^.SUIV;EN⌋
D;
END;

(*  ---------- PROGRAMME PRINCIPAL ------------  *)

BEGIN
INIT;DEDUIT;
BASEREF:=BASE;QUESTREF:=QUESTIONS;
REPEAT
   WRITE(HOME);MESS(20,1,' MENU GENERAL ',6);WRITE⌋
   LN;WRITELN;
   WRITE('C)onfirmer   D)éduire   Q)uitter    --->⌋
   ');
   CO:=PRENCAR(['C','D','Q','c','d','q']);
   BASE:=BASEREF;QUESTIONS:=QUESTREF;
   CASE CO OF
     'C','c' :CONFIRMER;
     'D','d' :DEDUIRE;
   END;
   WRITELN;WRITELN;IF CO IN ['C','D','c','d'] THEN⌋
   PRENRETURN;
UNTIL CO IN ['Q','q'];
END.
```

Micro-Informations

Jean-Michel Gourévitch

Et si 1987 était l'année du Macintosh ? C'est visiblement le souhait exprimé à Cupertino où le constructeur à la pomme finit de fourbir les armes qu'il dévoilera cette année et qui devraient lui assurer l'avantage dans un monde de la micro-informatique déboussolé par les incertitudes du géant IBM.

Parmi toute une salve de nouveaux produits, dont les premiers sont apparus en ce mois de janvier (et notamment les imprimantes laser de bas et de haut de gamme), la vraie nouveauté chez Apple sera en 1987 ce Mac "ouvert", dont l'apparition est prévue pour le mois de mars. Un Mac ouvert, parce qu'on pourra ajouter à la machine toutes sortes de cartes supplémentaires, d'extensions de mémoire, de processeurs graphiques ou sonores. Et renouer ainsi avec l'ancêtre, cet Apple // qui fit la fortune de Wozniak et Jobs, les créateurs d'Apple. Les possesseurs d'Apple // ne seront pas oubliés. D'abord parce que le nouvel Apple IIGS sortira cette année à une vraie cadence industrielle des usines d'Apple. Ensuite parce que tous les nouveaux programmes permettant d'exploiter au mieux ses raffinements sonores et graphiques vont se bousculer dans les catalogues. Tout comme les perfectionnements matériels des fabricants de "add-on". Message personnel aux possesseurs d'Apple // : ne bradez pas vos machines, quelque chose me dit qu'Apple pourrait rééditer avec l'Apple // l'opération de mise à niveau à laquelle elle avait procédé pour le MacPlus et ce pour un prix intéressant (on parle de moins de 5000 Francs pour l'échange de la carte). Espérons seulement que l'attente sera moins importante que pour les utilisateurs de Macintosh.

Revenons à celui-ci, qui pourrait bien être la machine dont on va le plus parler cette année. Un mot d'abord sur ces mises à niveau : aux États-Unis, de nombreux Mac convertis en MacPlus ont connu des problèmes d'alimentation qu'il a fallu remplacer. Apparemment, les conversions ont été mieux pratiquées en France, car ce problème ne semble pas être apparu chez nous.

L'Apple de demain décrit aujourd'hui...

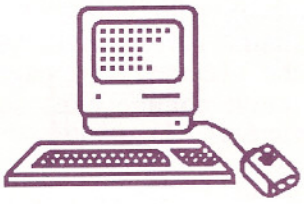
La grande question pour tous les passionnés d'Apple consistait à prévoir avant sa sortie l'apparence et les performances du nouveau Mac "ouvert". On a vu paraître tellement d'informations à ce sujet qu'on pensait ne plus rien ignorer des caractéristiques de la machine : processeur plus puissant et rapide Motorola 68020, port d'extension 'nubus', coprocesseur arithmétique 68881, mémoire vive standard de 4 Méga-octets. Une autre méthode pour prévoir les caractéristiques des machines du futur consiste à examiner ce que les fabricants de cartes et de périphériques mijotent pour la machine d'aujourd'hui : ces "superflus" seront souvent le "nécessaire" de la nouvelle machine. À ce jeu on pourrait prévoir, presque à coup sûr, pour le nouveau Mac un écran de grand format pouvant afficher une page pleine (au format 21,5cm sur 27,9cm) voire deux. Quelque chose me dit même que le Mac nouveau pourrait (en option, bien sûr) être doté de la couleur. Eh oui de la couleur, celle qui fait tant défaut, pour concurrencer complètement les IBM. Impossible ? Pas tellement : d'abord parce que les mémoires mortes du Mac contiennent tout ce qu'il faut pour cela, que les programmeurs des derniers logiciels parus ont prévu (souvent sans le dire) cette possibilité.

Enfin, parce qu'un constructeur (Vernont Microsystems) avait conçu, construit et vendu dès l'année passée un moniteur couleur "intelligent" le VM 8861 permettant d'afficher simultanément 256 couleurs choisies parmi une palette de 4096.

Mais la couleur n'est pas seulement le moyen de "griller" l'IBM. Ce n'est plus ce créneau que l'on vise chez Apple. Dans la ligne de mire du fabricant de Cupertino, on vise les "stations de travail" du type Sun Microsystems et autres. C'est à ce marché-là que le Mac ouvert (parfois baptisé Paris) aura la tâche de s'attaquer. Avec pour les *afficionados* deux conséquences mitigées. La première c'est que par rapport à ces systèmes extrêmement onéreux, le Mac en donnera incontestablement "plus pour chaque dollar dépensé". Mais il faut garder en mémoire que ces engins coûtent actuellement des fortunes (plus de

250 000 F). Le Mac ouvert proposera des possibilités analogues pour beaucoup moins cher. Probablement 40 à 60 000 F. Ce qui mettra quand même cet ordinateur, et c'est la seconde conséquence, bien moins agréable, hors de la portée de toutes les bourses. Mais Apple n'a jamais été un constructeur vendant des micro-ordinateurs bon marché, is'nt it ?

Le portrait robot du nouveau Mac, c'est donc celui d'une de ces "stations de travail", (connectables à des ordinateurs VAX) qui font fureur et font la fortune des constructeurs. À quoi servent-elles donc ? La question est importante, car de sa réponse vont découler les fonctionnalités du Mac nouveau. Ces stations de travail sont employées pour des études graphiques et des prototypes, pour concocter des études techniques et réaliser des documents et, enfin, à communiquer des informations. CAO, modélisation et graphiques d'un côté, édition personnelle et enfin communications : voici comme par hasard les thèmes généraux des derniers programmes sortis pour le Macintosh Plus d'aujourd'hui, et les domaines où il fera fureur avec son successeur cette année.



Des communications plus rapides et plus faciles

Juste un détail technique pour souligner l'importance du dernier cité de ces secteurs : la communication. La filiale de la firme Du Pont de Nemours spécialisée dans la *connectique* vient de produire un connecteur et un réseau à base de fibres optiques se substituant aux connecteurs et aux fils classiques AppleTalk. Avantage : les transferts de données s'effectuent sur des distances importantes (jusqu'à 1,5 Km) et à un débit de 600 kilobits par seconde (contre 238,6 pour AppleTalk, soit environ trois fois plus vite) et on prévoit de passer bientôt à un débit de 1,5 à 3 Mégabits par seconde (encore 3 à 5 fois plus vite), par le port SCSI. Ces vitesses considérables sont indispensables pour pouvoir faire travailler ensemble des micro ordinateurs, comme tous ceux qui les ont branchés en réseaux et piétinent devant les lenteurs des transferts actuels, ont pu le constater.

Les réseaux constituent un domaine essentiel de l'avenir du Mac. Aux États-Unis, **Farallon Computer** a développé le réseau **Phone Net** avec un contrôleur vendu 1000 dollars et permettant de relier 12 Mac par des fils simples torsadés jusqu'à 1 km de distance.

Apple va d'ailleurs venir cette année à la rescousse en fournissant dès le printemps un logiciel de serveur de réseaux aux performances qu'on dit déjà spectaculaires.

La liaison sur un réseau commun de Mac et d'IBM PC n'a déjà plus rien d'étonnant grâce aux prodiges réalisés par **TOPS** le fantastique système développé par **Centram**. Cette firme a récemment développé de nouveaux perfectionnements permettant d'une part de faire fonctionner l'ensemble sous le système UNIX, et d'autre part d'autoriser les IBM PC du réseau à imprimer directement sur la LaserWriter d'Apple.

Parmi les produits qui installent les Mac sur des réseaux, il faut encore noter **Intermail**, le système de courrier électronique d'**Internet**. Un système qui s'installe en accessoire de bureau et permet d'envoyer un message, immédiatement ou à une date choisie, de transférer des fichiers, pour un coût de 300 dollars (pour 1 à 4 utilisateurs), de 500 dollars (5 à 10 connectés), 750 (pour 11 à 20) etc.

Dans le même esprit, citons **Symbmail**, la messagerie de **Symbiotic**, pour passer messages et fichiers entre plusieurs Macintosh d'un même réseau AppleTalk

Et puis, pour finir ce panorama des communications, trois produits français :

«Charité bien ordonnée...», citons d'abord **Interpom's** programme gratuit pour vous qui avez Pom's entre les mains ou à 60 et 80 Francs sous forme de disquette Apple // ou Mac, programme qui utilise le modem du Minitel pour transmettre tous fichiers ou applications entre Mac, entre Apple // et même entre les deux types de machines. Les fichiers sont recréés à l'arrivée avec tous leurs attributs (types, icônes...). Qualité Pom's.

Le deuxième, **Self Serve d'ACI** est un logiciel permettant de créer un serveur Minitel en utilisant des modems branchés en cascade (et notamment les Diapasons d'Hello Informatique).

Le dernier, **TransLine** édité par **Bruno Rives & Associés**, est un logiciel de communication. Sa caractéristique principale : il n'est pas un émulateur de terminal. Il peut donc être utilisé dans n'importe quelle application ou presque pour transmettre ou recevoir des données, qui sont automatiquement converties au format de l'application qu'on utilise, pour remplir des tableaux ou récupérer des textes. On peut aussi communiquer entre ordinateurs par câbles (envoyer ainsi des documents MacDraw ou autres d'un Macintosh à l'autre).

Un outil d'édition et de publication

1986 a révélé toutes les possibilités du Macintosh dans le domaine de la publication assistée par ordinateur. Et la décision d'Apple de ne plus livrer en standard MacWrite a libéré l'imagination de tous les auteurs de traitements de textes. Résultat : il pleut des traitements de textes et des logiciels de mise en page.

Une pluie de traitements de textes

Le plus original est incontestablement **FeiMa d'Unisource**. Et pour cause : c'est le premier traitement de textes en chinois pour le Mac. Il utilise la souris pour sélectionner les caractères de préférence au clavier, permet de bénéficier d'un dictionnaire de caractères prédessinés, d'en ajouter d'autres. On peut aussi utiliser la méthode phonétique transformant des sons anglais en caractères chinois, et combiner dans un texte idéogrammes et caractères européens. Seul défaut : il n'imprime pas sur la LaserWriter. Prix 195 dollars pour une version abrégée et 545 dollars pour la version complète.

Pour certains, les caractères techniques seraient presque aussi compliqués que du chinois. C'est à eux que s'adresse **TechScriber** de **Mansfield Systems**. Plus que d'un traitement de textes, il s'agit presque d'un système de mise en page pour documents techniques permettant notamment d'entrer facilement les tables, expressions mathématiques compliquées, et de réaliser des manuels techniques comprenant diagrammes, éléments graphiques, etc. Prix : 395 dollars.

Il n'est d'ailleurs pas rare que les traitements de textes deviennent de vrais outils d'édition. C'est aussi le cas pour **Word 3.0**, la nouvelle version du traitement de textes vedette de **Microsoft**, appelée sûrement à un brillant avenir. Word comprend tout ce qu'on peut souhaiter dans un traitement de textes : correcteur orthographique incorporé, processeur d'idées lui aussi intégré, génération automatique d'index et de tables des matières, écriture en colonnes et possibilité d'entourer un texte, possibilité de visualiser sur l'écran une ou deux pages telles qu'elles seront finalement imprimées, possiblité de tourner sous Tops ou Ethernet, d'échanger des données avec les PC d'IBM en convertissant les fichiers au format DCA d'IBM, etc. Le tout pour 395 dollars. À noter encore qu'aux États-Unis, Word 3.0 serait livré avec la dernière version (5.1) du Switcher d'Apple permettant notamment lorsque l'on change un détail dans une des applications (par exemple à l'intérieur de MacPaint, dans un dessin transporté dans Word, de voir cette modification se réaliser automatiquement sur le dessin qu'on avait copié dans Word).

Autre futur best-seller, **Write Now** de **T/Maker** auquel Steve Jobs a prêté la main. Un traitement de textes comprenant un correcteur orthographique de 50 000 mots, l'ouverture simultanée d'un nombre illimité de documents, l'installation de une à quatre colonnes, la possibilité d'inclure des graphiques dans une ligne ou une phrase. Le tout avec la facilité de MacWrite et plus rapidement. Et pour 175 dollars.

Quant à **MindWrite** de **MindWork Software**, vendu seulement 125 dollars, il combine également un traitement d'idées et un traitement de textes. On dispose ainsi d'un traitement d'idées offrant toutes les facilités d'un MacWrite, permettant d'organiser tout en écrivant, et de voir à l'écran le résultat tel qu'il sera imprimé, de disposer d'une numérotation automatique des paragraphes, d'un compteur de mots, de caractères ou de paragraphes, de la faculté de déplacer une partie de texte avec la souris, sans recourir au copier coller, etc.

Full Write d'Ann Arbor Softworks (les créateurs de Full Paint) est un *processeur de documents*. En clair, il contient aussi des fonctions d'édition électronique. Parmi lesquelles, les possibilités d'utiliser des colonnes multiples de tailles différentes sur une page, d'enrouler automatiquement le texte autour de dessins aux formes irrégulières. Un module permet aux utilisateurs d'un texte d'inclure des notes dans un document pour les envoyer à l'auteur (ce module permettant de réaliser sur ordinateur ces petites notes adhésives de style *Post-It* pourrait d'ailleurs être versé par ses auteurs dans le domaine public). Full Write qui combine des fonctions d'édition qui ne figurent pas dans les traitements de textes et des fonctions sophistiquées de traitements de textes sera vendu environ 300 dollars.

Haba Word d'Haba Systems, qui a été offert pour 69 dollars aux possesseurs de MacWrite ou Word, permet, lui, 3 colonnes de tailles différentes sur une page, divise les fenêtres en quatre morceaux pour voir d'un coup plusieurs parties d'un texte, affiche à l'écran quatre pages en réduction comme elles seront imprimées.

Parmi les traitements de textes annoncés, on peut encore citer **Parangon Word**, de **Parangon Courseware** développé autour d'un éditeur de textes que certains utilisateurs du Mac connaissent sous le nom de Qued, promis pour juin 1987.

Des programmes d'édition électronique plus complets

Puisque les traitements de textes empruntent les possibilités gra-

phiques des programmes d'édition, rien d'étonnant s'il ne manque plus aux programmes d'édition électronique les facilités des traitements de textes. Ainsi de **Ready Set Go** de **Manhattan Graphics** (distribué en France par BIP) dont la dernière version 3.0 est un complet renouvellement. On y trouve même un correcteur orthographique (en français s'il vous plaît) et un guide de coupure de mots développé avec l'aide de l'Université de technologie de Compiègne. On peut ouvrir simultanément plusieurs documents, ajuster automatiquement textes et graphiques dans les colonnes après y avoir inséré un bloc image ou un texte, lire directement des fichiers MacPaint, MacWrite ou Works, on peut y inclure des documents spécialement préparés pour PostScript. Le nombre de pages est illimité, avec auto-numération, et impression de la date et de l'heure, etc.

Feu Mac Publisher est réapparu, vendu par la célèbre firme de lettrages **Letraset** sous le nom de **Letra Page**, perfectionné avec la possibilité de 48 colonnes par pages, un espacement des lettres contrôlable, une palette de 96 motifs disponibles, et de 9 couleurs, et lui aussi un guide de coupure de mots automatique, comprenant 93 000 mots (en anglais seulement pour le moment) etc. pour 495 dollars.

Et le petit dernier, **Solo** de **Mac America**, plus simple, ressemble à un banal Finder sur lequel on ouvre les différents documents à assembler. Une fenêtre pour un texte à incorporer, une fenêtre pour le texte assemblé. À noter que le texte qu'on ne prépare ne montre pas le texte avec ses attributs (gras, taille, etc.) mais avec des codes.

Des logiciels graphiques

À remarquer d'abord, l'**Illustrator** d'Adobe. Si le nom d'**Adobe** n'est pas inconnu aux utilisateurs de Macintosh, c'est que c'est cette compagnie qui a développé PostScript, le langage de composition utilisé par l'imprimante LaserWriter. C'est précisément pour optimiser le fonctionnement de l'imprimante Laser qu'Adobe a développé Adobe Illustrator. Ce programme rend accessible à des utilisateurs ne programmant pas en langage PostScript certaines de ses routines graphiques. En clair, les utilisateurs professionnels (et notamment les illustrateurs ou dessinateurs) disposent ainsi de possibilités de dessin beaucoup plus professionnelles. On commence à dessiner dans un mode de dessin, depuis zéro ou en important une esquisse réalisée avec MacDraw. Ensuite, on dispose d'une panoplie d'outils pour remplir les zones à colorier, contrôler l'épaisseur des lignes, les niveaux de gris, et même les couleurs (pour préparer la séparation des couleurs sur une photocomposeuse Linotronic). Un mode permet de visualiser l'image réalisée avec PostScript. Enfin, les dessins sont stockés sous forme de fichiers textes PostScript. Ce programme très "pro" sera vendu à peu près 500 dollars.

Du côté des programmes de dessins, on se bouscule aussi au portillon. MacPaint et MacDraw font vraiment figures d'ancêtres. Voici notamment **Cricket Draw** de **Cricket Software**, avec possibilité de rotation complète de tous les objets, ombrages contrôlés avec une échelle de gris, effets spéciaux, le tout utilisant PostScript, sans apparemment y toucher, tout en générant du code dans ce langage comme le précédent programme . Pour 295 dollars.

SuperPaint de **Silicon Beach Software** est un programme qui voici peu aurait encore paru "incroyable". On peut en effet travailler sur un dessin indifféremment en mode MacPaint ou en mode MacDraw. Ce qui permet de bénéficier des avantages de ces deux modes. On peut travailler en plein écran, ouvrir simultanément plusieurs documents, remplir des formes qui se poursuivent en dehors de l'écran, le déroulement de l'image sur l'écran est automatique, on dispose d'une échelle de gris de 32 niveaux, le texte peut être imprimé avec les caractères de la LaserWriter, il se reformate automatiquement lorsqu'on change les dimensions d'une case qui le contient. On peut imprimer en couleurs avec l'ImageWriter II. Prix : moins de 100 dollars.

Quant à **GraphicWorks**, il est publié par **Mindscape**, l'auteur de Comic Works et de Music Works. En fait, il utilise les mêmes outils que Comic Works. Rien d'étonnant si on peut y utiliser toute une gamme de ballons et de bulles (aussi appelés en français phylactères). Sa grande force, c'est de pouvoir manipuler indépendamment les objets de premier plan et le fond d'un paysage, de pouvoir installer des dessins en surimposition. Prix : 80 dollars.

À remarquer aussi la version 2 de **MacDraft** d'**IDD Software**. On y trouve notamment la possibilité de changer la taille des groupes et des *bits maps* de façon proportionnelle, un indicateur de position du curseur, de nouvelles échelles de dessin, un calculateur automatique de surface, et un choix de 64 motifs, etc. Prix : 269 dollars.

Des logiciels techniques

Très importants. Ces programmes sont ceux où l'on voit le plus l'usage qu'un Mac peut avoir pour des scientifiques, des ingénieurs, bref, tous les utilisateurs de ces omniprésentes "stations de travail". Ce sont ceux-ci qui permettent peut-être d'apprécier le mieux l'intelligence des programmeurs et des utilisateurs. Comme s'il s'agissait de torrents de matière grise en fusion.

Voici d'abord **Stella** de **High Performance Systems**. Lorsque l'on voit dans sa publicité, le directeur de la recherche de DuPont de Nemours s'exclamer «Nous avons des programmes de 2000 dollars pour nos unités centrales qui ne peuvent faire ce que Stella fait», ou encore l'analyste chargé du planning de Cray Research, le constructeur des plus puissants ordinateurs du monde expliquer que le programme les a aidé à «équilibrer les facteurs qui satisfont la demande de leurs ordinateurs avec les facteurs qui génèrent cette demande», on se dit que le programme doit être exceptionnel. Aucun doute, ce logiciel de modélisation baptisé Stella pour (Structural Thinking Experiential Learning Laboratory with Animation : laboratoire d'apprentissage expérimental de la pensée structurale avec animation) permet d'améliorer la façon de penser et de mettre en évidence des relations entre facteurs, comme un "Lego conceptuel". On peut simuler des modèles pour déterminer si les propositions génèrent des comportements correspondant bien à l'objet qu'on avait assigné. Qu'il s'agisse d'analyser des processus de fabrication complexe, ou comme l'a fait Procter et Gamble, d'analyser des stratégies pour le lancement d'un nouveau produit. Bref, une manière de miracle pour 395 dollars.

Voici ensuite **Prototyper** de **Smethers Barnes**, un programme à sortir en février permettant de construire des prototypes pour visualiser des idées avec des menus des boutons et des fenêtres qui répondent. Sans avoir à programmer.

Parameter Manager de **Structural Measurement Systems** est une matière de tableur spécialisé pour contrôler notamment les processus de fabrication. Utile en milieu industriel. Prix : 495 dollars.

Des classiques "revisités"

On pensait avoir tout vu en matière de tableurs. Et qu'après Excel, on aurait du mal à faire mieux. On peut en tout cas faire déja plus rapide. C'est l'objet de **Mac Calc** de **Bravo Technology**. Qui permet notamment (contrairement à Excel) de changer les caractères et leurs tailles dans chaque case. Ce tableur calcule de 10 à 15% plus vite qu'Excel, permet d'inclure des annotations relatives à une case) et dispose d'un choix de fonction pour moins de 100 dollars.

Quant à **Trapeze**, de **Data Tailor**, à paraître à la fin janvier, il présente la caractéristique de travailler en "blocs" de plusieurs cellules (pour lesquels le programme connaît les relations qui les unissent) et non case par case comme la majorité des tableurs. D'où la possibilité de faire des calculs sans se soucier de la position d'un bloc sur une page, et de réaliser des calculs financiers complexes (on dispose de 10 fonctions financières et d'ingéniérie dont certaines inédites). D'où la possibilité aussi de réaliser des présentations parfaites.

Des utilitaires géniaux

Génial, c'est bien le mot qui caractérise **Glue de Solutions Inc.** Ce programme permet de sauver sur le disque des graphiques venant de la plupart des applications pour Macintosh, et de les ouvrir sans avoir à ouvrir l'application. On peut par exemple sauver un graphique d'Excel, grâce à la partie du programme baptisé Image saver. Ensuite, on peut ouvrir ce fichier et visualiser le graphique ou l'imprimer avec l'autre partie baptisée le Viewer. Les fichiers sont facilement exportables. D'où la possibilité d'expédier facilement une lettre d'informations ou un dessins sauvés avec Glue sur un réseau ou par téléphone, avec InterPom's par exemple. Prix 49 dollars. Vivement que cet utilitaire (auquel un banc d'essai de la très sérieuse revue Inforworld a attribué une note maximale) débarque chez nous.

Utile, c'est **1st Aid** de **1st Aid Software** qui permet de récupérer des fichiers endommagés de sauver les données de disques "illisibles" ou de relire des fichiers endommagés ou effacés. Un kit de 4 disques vendu 90 dollars et que tout utilisateur de Mac devrait posséder.

Du matériel

Ces nouveaux écrans géants semblent désormais indispensables pour le Mac. Après le FDP de Radius, voici le **Big Picture** de **E Machines**. C'est un écran permettant de visualiser simultanément deux pages sur l'écran. Il faut bien sûr modifier le Mac avec une carte qui se clippe sur le processeur. On peut alors apprécier une visualisation de 1024 x 808 points avec une qualité remarquable. Pour un prix hélas supérieur à celui d'un Macintosh Plus complet.

Un jeu

C'est à Jean Pierre Brûlé, l'ex-Mr

Informatique en France (mais ausi champion de scrabble) que l'on doit **Anacrack** vendu par **ACI**. Un jeu basé sur l'utilisation de 90 000 mots avec 2 niveaux de vocabulaire 4 vitesses et trois formes de jeux rappelant les principes du célèbre Scrabble.

Adresses

Vernont Microsystems
11 Tigan St Winooski
VT 05404 Centram 2560 Ninth St, Berkeley, CA 94710

Internet
20 Amy Circle,
Waban MA 02168

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

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

Unisource
23 East St
Cambridge MA 02141

Mansfield Systems
550 Hamilton Av.
Suite 200 Palo Alto, CA 94301

T/Maker
1973 Landing Drive
Mt View, CA 94043

MindWork Software
PO Box 222280
Carmel CA 93922

Ann Arbor Softworks
2393 Teller Road, Suite 106
Newbury Park, CA 91320

Haba Systems
6711 Valjean Ave. VanNuys
CA 91406

Lettraset
PO Box 3900 Peoria, IL 61614

Mac America
18032C Lemon
Dr Yorba Linda CA 92686

Adobe
1870 Embarcadero
Rd N°100 Palo Alto, CA 94303

Cricket Software
3508 Market Street
Suite 206 Philadelphia PA 19104

Silicon Beach Software
PO Box 261430
San Diego CA 92126

Mindscape
3444 Dundee Rd
Northbrook IL 60062

IDD Software
1975 Willow Pass Rd Ste 8
Concord CA 94520

High Performance Systems
PO BOX B1167
Hanover NH 03755

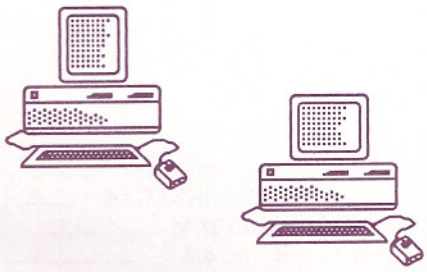
Smethers Barnes
Tél. :
(aux États-Unis) 503 24507270

Structural Measurement Systems
645 River Oak Pkwy
San Jose CA 95134

Bravo Technology
PO Box T Gilroy CA 95021

Data Tailor
1300 University Drive Suite 409
Fort Worth Texas 76107

Solutions Inc.
PO Box 989 h
Montpelier VT 05602



1st Aid Software
42 Radnor Road
Boston Ma 02135

E Machines
7945 S W Mohawk
St Tualatin OR 97062

Symbiotic
4, rue Robert Schumann
94220 Charenton
Tél. : 43 78 99 99

JustText™ à l'essai

JustText suppose la disposition d'un Macintosh et d'une LaserWriter, imprimante dont bien des PME sont équipées. Ce logiciel de traitement de textes déroute par son aspect 'réac' pour un programme Macintosh : comme avec AppleWriter, vous n'avez pas à l'écran une idée de ce que sera le document imprimé, tout simplement parce que l'écran ne peut donner les possibilités de la LaserWriter ou des composeuses Linotronic. Il faut donc *enrichir* le texte de codes de contrôle parfois (!) simples à retenir. Le résultat est une exploitation totale de l'imprimante : dimension et espacement des caractères, interlignes, largeur de colonnes sont réglables par point (0,35 mm), et même fractions de point.

Toutes les fonctions classiques des traitements de textes sont disponibles (tabulations, indentations), mais aussi les fantaisies des imprimeurs, 'vitrine' comme le premier J de ce texte, encadrer un texte ou cadrer à droite une fin de paragraphe... Les césures en fin de ligne sont automatiques en suivant un dictionnaire que vous enrichissez en fonction des besoins.

Lorsque lesdits besoins dépassent JustText, vous pouvez modifier directement les ordres PostScript avant qu'ils ne soient envoyés à l'imprimante : les limites sont alors l'imagination de l'opérateur et les possibilités de la machine (voir Pom's 24).

Au chapitre regrets : l'annulation du "Coller" qui n'est que partielle, la limitation des textes à 32 Ko (pénible lorsqu'on incorpore une image dont on *habille* les contours), quelques petits 'bugs' tels la validation des menus, la difficulté d'insérer des 'RETURN' au début de longs textes, les *A* avec accent grave qui s'impriment *Á*... Le mode d'emploi par l'exemple est efficace, mais en anglais.

D'un accès peu engageant, JustText est le programme à utiliser pour les travaux pointus.

Distribué par alpha systèmes à Grenoble

Bon de commande

Disquettes

HAIFA source	(cf. Pom's n° 5)	à	60,00 F	.....
DISK-MANAGER	(cf. Pom's n° 11)	à	450,00 F	.....
BASICIUM	(cf. Pom's n° 13)	à	150,00 F	.....
E.P.E. 5.0	(cf. Pom's n° 23)	à	200,00 F	.....
Échange E.P.E. 5.0	(cf. Pom's n° 23)	à	80,00 F	.....
PASCAL	(cf. Pom's n° 15)	à	80,00 F	.....
MAX (Moniteur étendu)	(cf. Pom's n° 18)	à	150,00 F	.....
DOMINOS	(cf. Pom's n° 19)	à	80,00 F	.....
COGO	(cf. Pom's n° 21)	à	150,00 F	.....
LUDOLOGIC	(cf. Pom's n° 25)	à	80,00 F	.....
ORDICO	(cf. Pom's n° 26)	à	200,00 F	.....

Recueils

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

Revues, disquettes

Revues 4 7 8	à	35,00 F	.....
Revues 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26	à	40,00 F	.....
Revues 27 28	à	45,00 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 24 25 26 27 28	à	60,00 F	.....
Disquettes Macintosh			
14/15/16 groupées	à	150,00 F	.....
17 18 19 20 21 22 23 24 25 26 27 28	à	80,00 F	.....
Mac 'A'	à	80,00 F	.....
MacAstuces	à	200,00 F	.....
"Raccourci"	à	200,00 F	.....

Abonnements

Pour 6 numéros à partir du n° .....

Abonnement à la revue seule	à	225,00 F	.....
Abonnement revue + disquettes Apple II, //e, //c	à	525,00 F	.....
Abonnement revue + disquettes Macintosh	à	625,00 F	.....

Reliures toilées

Pour 6 numéros, un an de Pom's	à	60,00 F	.....

Total TTC : .....

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

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 : Janvier 1987. N° 4122

Bon de commande 'Communication'

Pom's s'est intéressé ces derniers numéros à l'ouverture de votre ordinateur sur le monde extérieur.

Pom's 26 : **Apple //, Carte SuperSérie & CP/M. Programme permettant de configurer simplement la SSC sous CP/M**

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

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

Je désire recevoir :

Revue	26 27 28	à	45,00	.....
Disquette Apple //	26 27 28	à	60,00	.....
Disquette Macintosh	26 27 28	à	80,00	.....
»»»» Câble de liaison				
Apple][+, //e, //e+		à	225,00	.....
Apple //c		à	225,00	.....
Macintosh 128, 512		à	225,00	.....
Macintosh Plus		à	225,00	.....

Je désire m'abonner :

À compter du numéro _____
(6 numéros, Un an, Possibilité d'abonnement rétroactif)

Revues seules	à	225,00	.....
Revues et disquette Apple //	à	525,00	.....
Revues et disquettes Macintosh	à	625,00	.....
		Total TTC	.....

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

Envoyez ce bon et votre règlement à : Éditions MEV, 64, rue des Chantiers 78000 Versailles

Nom :

Adresse :