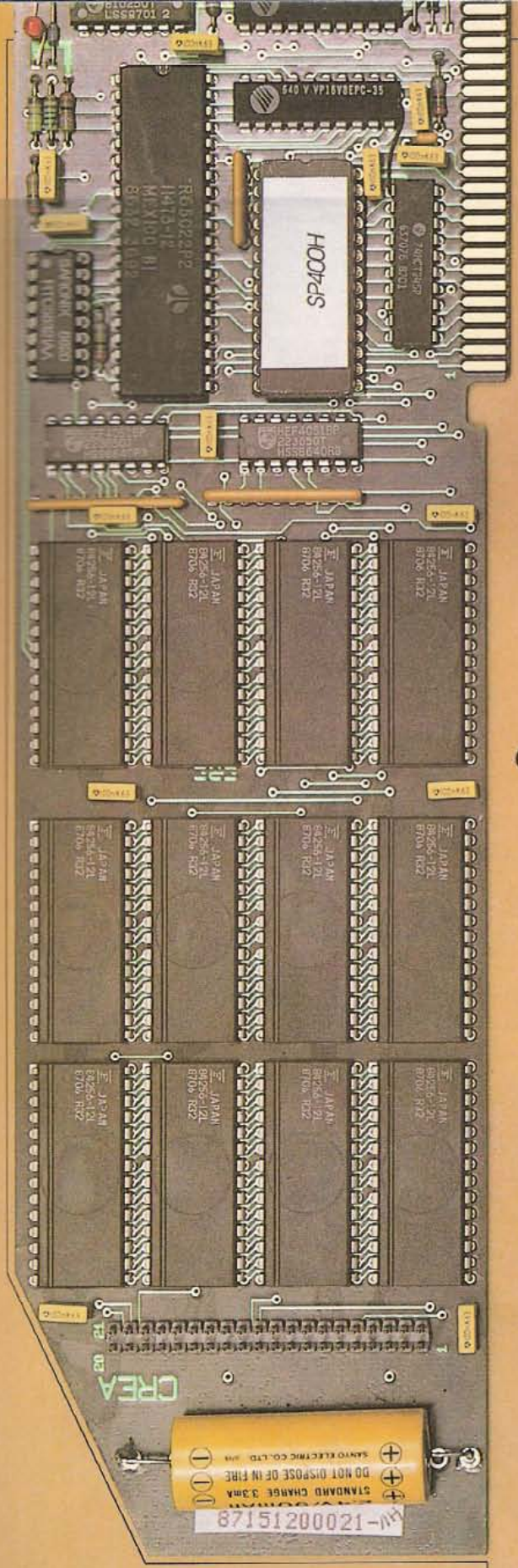
La revue [illegible] indépendante pour les utilisateurs des Apple[illegible], //e+, //c, IIGS™ et Macintosh™

Dans ce numéro : votre serveur sur Apple™

Som's

M 2366 - 34 - 45,00 F
3792366045005 00340
ISSN 0294-6068



L'anti-disque

Speedisk™, la RAM Card à mémoire permanente

Rapidité Temps d'accès à l'information : 0,2 ms (100 fois plus rapide que les disques durs...).
Par exemple :
démarrage sur Basic.System en 3 s,
AppleWriter disponible en à peine 1 s.

Fiabilité Constituée de circuit CMOS à très faible consommation, Speedisk™ est aussi fiable que l'ordinateur lui-même. Elle est insensible à l'environnement.

Capacité Speedisk™ est proposée en quatre versions :
1 Mo (1 048 576 octets)
384Ko extensible à 1Mo
avec horloge compatible ProDOS (pour les Apple)(+, //e) ou sans (IIGS)

Compatibilité 100% compatible avec ProDOS (c'est un volume), Speedisk™ fonctionne sur Apple)(+, sur Apple //e et sur Apple IIGS.

Prix Lecteur de Pom's, vous bénéficiez d'une remise de 10 % :

SP400		
384Ko	~~3 990,00~~	**3 591,00**
SP1000		
1 Mo	~~5 990,00~~	**5 391,00**
SP400H		
384Ko horloge	~~4 580,00~~	**4 122,00**
SP1000H		
1 Mo horloge	~~6 580,00~~	**5 922,00**

Garantie Speedisk™ est une fabrication française garantie un an par échange de la carte.

Banc d'essai dans la revue Pom's n° 31

Vente par correspondance, documentation, renseignements
Logma S.A.
12, rue d'Anjou
78000 Versailles
Tél : (1) 39 51 24 43

Speedisk™ est une marque déposée de Thot Informatique® – France

Speedisk™

Je désire recevoir – sans engagement – votre documentation sur les cartes Speedisk™

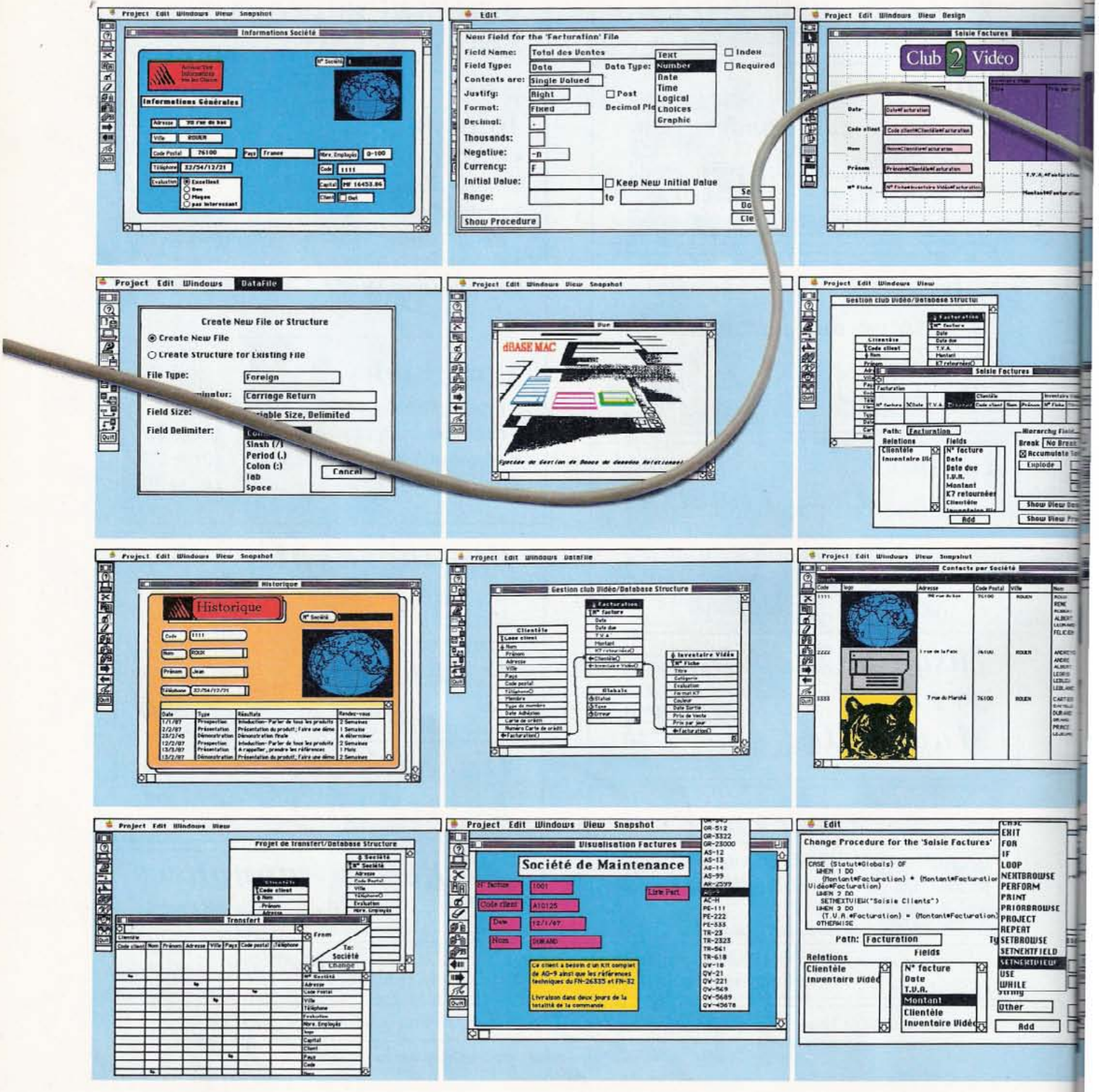
Nom.....
Adresse.....
.....
.....

Numéro 34
janvier-février 1988

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

Imprimé en France. N° d'impression 785066 — Dépôt légal janvier 1988

Ashton-Tate et La Co
présentent d



mmande Electronique

BASE Mac

DISPONIBLE EN FRANÇAIS

dBASE® Mac est disponible.
C'est le plus puissant système de gestion de base de données relationnelle pour Macintosh™.
A la différence des autres bases de données, vous l'utilisez sans aucune connaissance de la programmation.
Commencez par créer et relier vos fichiers. La modification d'une information entraînera la mise à jour des autres.
Automatiquement et immédiatement.
Et souvenez-vous qu'avec dBASE Mac, tout est réalisable sans programmer.
Cliquez simplement la souris et vous visualiserez vos fichiers de mille et une façons.
Toujours sans faire appel à la programmation.
En insérant des polices de caractères issues de MacDraw®, vous créez les rapports les plus simples aux plus sophistiqués. Ils apparaîtront à l'écran, tels qu'ils seront imprimés.
Nous ne parlons pas simplement du texte et des graphiques. Avec dBASE Mac, vous pouvez même incorporer des photos au sein des fichiers.
Toujours sans programmation, vous importez des données issues de dBASE III PLUS ou de n'importe quel fichier ASCII. Tout cela, grâce au système d'aide intégré et aux menus déroulants du Macintosh.
Mais si vous voulez aller encore plus loin, un puissant langage de programmation permettra de développer des applications.
Vous désirez une documentation complète ou assister à une présentation, alors, n'hésitez pas à contacter votre revendeur agréé Apple.
dBASE Mac est disponible au prix de 3 950 F (HT).

BON DE COMMANDE - 34 -

A retourner à :
La Commande Electronique
7,rue des Prias
27920 SAINT PIERRE DE BAILLEUL

Nom : .....

Société : .....

N° : .. Rue : .....

.....

Ville : .....

Code postal : .....

Téléphone : .....

☐ Je vous commande ..... exemplaires de la prise en main dBASE Mac, au prix de 50 F TTC. Règlement par chèque joint à la commande.

☐ Envoyez-moi une documentation complète sur le logiciel dBASE Mac.

La Commande Electronique
7, RUE DES PRIAS — 27920 SAINT-PIERRE DE BAILLEUL
TÉL. 32 52 54 02 TELEX LCE 180 855 FAX 32 52 54 46

ASHTON·TATE

Ont collaboré à ce numéro
Alexandre Avrane – Jean-Luc Bazanegue
Yvan Béard – Jean-Yves Bourdin
Jean-Jacques Colwhir – Stéphane Dedeyan
Alexandre Duback – Dimitri Geystor
Olivier Herz – Guillaume Lejeune
Gérard Michel – Christian Piard
Joëlle Piard – Hervé Thiriez
Bernard Toméno – Éric Weyland

Directeur de la publication
rédacteur en chef
Hervé Thiriez

Rédacteurs
Alexandre Avrane – Olivier Herz

Siège social
Éditions MEV – 12, rue d'Anjou
78000 Versailles – ☎ (1) 39.51.24.43

Publicité
Éditions MEV

Diffusion
N.M.P.P.

Impression
Berger-Levrault
18, rue des Glacis
54000 Nancy
☎ 83.35.61.44

Photos de couverture
CP & JLB

Photogravure
Graphotec
21, chemin de la Tour
92350 Le Plessis-Robinson
☎ (1) 46.30.44.49

Pom's est une revue indépendante non rattachée à Apple Computer, Inc. ni à Apple Computer France S.A.R.L.
Apple, le logo Apple, Mac et le logo Macintosh sont des marques déposées d'Apple Computer, Inc.
IBM est une marque déposée de International Business Machine.
PC et AT sont des marques déposées de la Société IBM.

Éditorial

Pom's sur Minitel, aujourd'hui

Pom's est bien plus la revue *des Apple* que celle des Apple // avec un cahier Mac : comme Minitel 27, Kruptos, InterPom's, T_Pom's, Clv, nous vous proposons à nouveau des programmes communs aux Apple // et au Macintosh. Il s'agit de Pom_Link et Répom'deur, deux de ces programmes qui donnent à un seul numéro la valeur d'un abonnement.

(1) 39 53 04 40, tel est le numéro de téléphone que Pom's met à la disposition de votre Minitel 24 heures sur 24. Dès aujourd'hui, vous y découvrirez le répondeur télématique de ce numéro, répondeur prêt à enregistrer vos messages et, pourquoi pas, vos commandes ; voici donc la possibilité d'essayer un nouveau programme de communication avant d'acheter la disquette. Qui dit mieux ? Ce nouveau numéro de téléphone est promis à bien des développements mais, rançon du succès, pardonnez-nous en cette période de mise en service quelques tonalités "occupé".

S'il est un service qui vous tient, et qui *nous* tient, à cœur, c'est bien l'assistance téléphonique pour la mise en œuvre de nos programmes ; c'est une charge importante, mais elle nous semble capitale.
Nous comprenons que ceux qui manifestent leur fidélité par un abonnement en attendent plus d'efficacité, aussi nous travaillons à des solutions : rendez-vous au numéro 35.

Comme tout service gratuit, cette assistance comporte un sympathique sottisier (qui occuperait bien un numéro complet) tel le cas de cet utilisateur d'InterPom's – au demeurant peu aimable – qui avait quelques soucis de communication : il s'agissait d'un pirateur de base dépourvu de mode d'emploi, qui exigeait une aide ! Heureusement, notre flegme nous aide...

Sondage ? nous recevons à l'heure de ces lignes encore bien des réponses : dépouillement dans le prochain numéro mais déjà une dominante apparaît, un succès pour la rubrique *Apple // for ever* – doutez-vous encore de l'inventivité des développeurs – et des programmes de communication.

Hervé Thiriez

Copie de l'écran texte 40 & 80 colonnes

Yvan BEARD

Le programme en assembleur présenté ici est une évolution de celui de Christian Guérin paru dans le numéro 3 de Pom's. Il fonctionne dans les mêmes conditions et s'utilise de la même façon. L'évolution réside dans le fait qu'il détermine la nature de l'écran et donne aussi bien la copie d'un écran en 40 colonnes qu'en 80 colonnes. Il fonctionne sous ProDOS et DOS 3.3.

Le petit programme BASIC ci-joint donne un exemple d'utilisation.

Comment faire ?

Il suffit de saisir et sauvegarder la récapitulation TXT.DMP.C (sauvegarde : *BSAVE TXT.DMP.C, A$300, L$60*). Le programme se charge en page 3 (ce n'est pas très original, mais sous ProDOS, quelle simplification) ; donc, l'une des premières lignes de votre programme Basic sera :

```
20 PRINT CHR$(4)"BLOADTXT.DMP.C"
```

À chaque fois que vous avez besoin d'envoyer une copie d'écran à un périphérique (ce sera bien souvent l'imprimante), faites :

```
PRINT CHR$(4)"PR£1": CALL 768: PRINT CHR$(
   4)"PR£0"
```

Ceci dans le cas ou le périphérique destinataire est connecté au port 1. Veillez à ce qu'il n'y ait pas d'écho à l'écran lors de l'envoi des caractères au périphérique (à l'imprimante), sinon l'écran sera défiguré au fur et à mesure de l'impression. En principe un :

```
PRINT CHR$(9)"80N"
```

avant le CALL 768 règle ce petit problème.

À noter enfin que le fichier binaire s'appelle TXT.DMP.C, mais qu'il conviendra à toute imprimante.

Source TXT.DMP
Assemblage par ProCODE

```
0  ********************************************************
1  *                                                      *
2  *         HARD COPIE TEXTE 40 ET 80 COLONNES           *
3  *                                                      *
4  ********************************************************
5  *
6            ORG  $300
7  *
8  ADL       EQU  $6          ;Adresse de début de ligne
9  LIGNE     EQU  $8          ;Numéro de ligne
10 OUT       EQU  $FDED       ;Ecriture d'un caractère
11 PAGE2OFF  EQU  $C054       ;Sélection page 1
12 PAGE2ON   EQU  $C055       ;Sélection page 1A
13 RD80COL   EQU  $C01F       ;Test 40 ou 80 colonnes
14 *
15 *-------------------------------------------------------
16 *     Début du programme
17 *-------------------------------------------------------
18 *
19 HCT       LDA  £0          ;Début Hard Copie ligne 0
20 HCT1      PHA              ;Sauvegarde numéro ligne
21 *
22 *-------------------------------------------------------
23 *   Calcul de l'adresse de début de la ligne
24 *-------------------------------------------------------
25 *
26           PHA
27           LSR
28           AND  £$3
29           ORA  £$4         ;Routine VTAB
30           STA  ADL+1
31           PLA
32           AND  £$10
33           BCC  HCT2
34           ADC  £$7F
35 HCT2      STA  ADL
36           ASL
37           ASL
38           ORA  ADL
39           STA  ADL
40 *
41 *-------------------------------------------------------
42 *       Ecriture d'une ligne
43 *-------------------------------------------------------
44 *
45           LDY  £0          ;Y pointe dans la ligne
46 *
47 HCT3      LDX  £$7F        ;Est-on en mode
48           CPX  RD80COL     ;    80 ou 40 colonnes?
49           BCS  HCT5        ;Si en 40 colonnes
50 *
51                            ;CARACTERE EN PAGE 1A
52           LDA  £0
53           STA  PAGE2ON     ;Activation page 1A
54           LDA  (ADL),Y
55           STA  PAGE2OFF    ;Retour en page 1
56           BMI  HCT4
57           AND  £$3F        ;Si le caractère est en
58           CMP  £$20        ;  inverse ou en flashant
59           BCS  HCT4        ;  on le transforme
60           ORA  £$40
61 *
62 HCT4      ORA  £$80
63           JSR  OUT         ;Ecriture sur l'imprimante
64 *
65                            ;CARACTERE EN PAGE 1
66 HCT5      LDA  (ADL),Y
```

```
67              BMI   HCT6
68              AND   £$3F
69              CMP   £$20
70              BCS   HCT6
71              ORA   £$40
72  HCT6        ORA   £$80
73              JSR   OUT
74  *
75              INY               ;Caractère suivant
76              CPY   £!40        ;Test si fin de ligne
77              BNE   HCT3
78  *
79  *-------------------------------------------------
80  *     Fin d'une ligne
81  *-------------------------------------------------
82  *
83              LDA   £"M"-$40    ;Envoi de 'RETURN'
84              JSR   OUT         ;  sur l'imprimante
85  *
86              PLA
87              TAY
88              INY               ;Passe à lig
89              TYA               ;suivante
90              CMP   £!24        ;Test si fin
91              BNE   HCT1        ;de la page
92  *
93              RTS
```


Récapitulation TXT.DMP.C

Après avoir saisi cette récapitulation sous moniteur, vous la sauvegarderez par : BSAVETXT.DMP.C,A$300,L$60

```
0300:A9 00 48 48 4A 29 03 09
0308:04 85 07 68 29 18 90 02
0310:69 7F 85 06 0A 0A 05 06
0318:85 06 A0 00 A2 7F EC 1F
0320:C0 B0 19 A9 00 8D 55 C0
0328:B1 06 8D 54 C0 30 08 29
0330:3F C9 20 B0 02 09 40 09
0338:80 20 ED FD B1 06 30 08
0340:29 3F C9 20 B0 02 09 40
0348:09 80 20 ED FD C8 C0 28
0350:D0 CA A9 8D 20 ED FD 68
0358:A8 C8 98 C9 18 D0 A3 60
```

Programme de démonstration TXT.DMP.DEMO

```
10  TEXT : HOME : PRINT  CHR$ (21);
20 D$ =  CHR$ (4):FLAG = 0
30  FOR J = 1 TO 39:A$ = A$ + "*": NEXT
40  PRINT D$;"BLOADTXT.DMP.C"
50 B$ = "POM'S": IF FLAG THEN B$ = " P O
   M ' S "
60 C$ = "HARD COPIE TEXT"
65  IF FLAG THEN C$ = " H A R D   C O P I
    E   T E X T "
70  IF FLAG THEN  PRINT D$"PR£3"
80  PRINT A$;: IF FLAG THEN  PRINT A$;"*"
   ;
90  FOR I = 1 TO 21
100  PRINT "**";
110  IF FLAG THEN  HTAB 80: GOTO 130
120  HTAB 40
130  NEXT I
140  PRINT A$"**";: IF FLAG THEN  PRINT A
   $"*";
150  VTAB 4: HTAB 7: IF FLAG THEN  PRINT
    TAB( 5);
160  INVERSE : PRINT B$: VTAB 11
170  IF FLAG THEN  HTAB 24:  INVERSE  :
   GOTO 190
180  HTAB 13: FLASH
190  PRINT C$
200  VTAB 18: NORMAL : HTAB 4: IF FLAG TH
   EN  PRINT  TAB( 20);
210  PRINT "Mettez votre imprimante en ma
   rche"
220  VTAB 20: HTAB 8: IF FLAG THEN  PRINT
    TAB( 20);
230  PRINT "puis tapez sur une touche"
240  WAIT  - 16384,128: POKE  - 16368,0
250  PRINT D$"PR£1": CALL 768: PRINT D$"P
   R£0"
260  HOME : IF FLAG THEN  END
270 FLAG = 1: GOTO 50
```

AppleWorks : des macro-commandes

Dimitri Geystor

L'un des reproches que l'on adresse à la version française 1.4 d'AppleWorks est qu'elle n'a pas (en tableur) les fonctions àROUND, àAND et àOR de sa sœur 2.0 américaine, pas plus qu'elle ne possède de programme de mailing.

Qu'à cela ne tienne, SuperMacroWorks permet de contourner ces lacunes avec élégance. Pour vous mettre en appétit, Pom's vous propose, dès ce mois-ci et dans la nouvelle rubrique consacrée à AppleWorks et SuperMacroWorks, les trois macros AND, OR et ROUND.

Rappelons que les fichiers de macro-commandes se présentent sous la forme de fichiers *traitement de texte AppleWorks*. Une fois que vous avez modifié votre AppleWorks avec le logiciel SMW, tout ce que vous avez à faire est de charger le fichier de macros, l'appeler à l'écran, et faire ⌘= pour compiler celles-ci et les rendre actives.

Les macros sont actionnées en appuyant simultanément sur la touche ⌘ et la touche d'identification de la macro. Pour plus de détails sur la syntaxe des macros, reportez vous au manuel français qui accompagne SMW.

Voici le listing de la macro AND.OR.ROUND. Vous trouverez ce fichier sur la disquette d'accompagnement de Pom's. Une dernière remarque : le pavé explicatif entre les deux traits d'astérisques n'est nullement nécessaire pour le bon fonctionnement des macros, mais c'est une bonne habitude à prendre : il fait exactement les dimensions d'un écran, et rappelle en quelques lignes les fonctions et commandes qui suivent.

```
**********************************************

      APPLEWORKS 1.4 - SuperMacroWorks
            AND.OR.ROUND

La version française 1.4 d'Appleworks ne
contient pas les fonctions àROUND, àAND et àOR
de la version américaine 2.0.

Mais il est tout à fait possible de les
introduire avec les macros ⌘R, ⌘A et ⌘O de ce
fichier.

Compilez avec ⌘=

puis allez dans un tableur, et expérimentez.
Les macros vous guident par des messages, il ne
vous reste qu'à introduire les valeurs ou
expressions.

**********************************************
Dimitri Geystor - Novembre 1987

START

O:<asp>àif(<msg>"
            "<msg>'OR : première expression, p
uis RETURN'<input>;1;(àif(<msg>" "<msg>'OR : de
uxième expression, puis RETURN'<input>;1;0)))<r
tn><msg>'Résultat du OR'!

A:<asp>àif(<msg>"
          "<msg>'AND : première expression, p
uis RETURN'<input>;(àif(<msg>" "<msg>'AND : deu
xième expression, puis RETURN'<input>;1;0));0)<
rtn><msg>'Résultat du AND'!

R:<asp>àint(((<msg>" "<msg>'Expression à arrond
ir , puis RETURN'<input>)+,005)*100)/100<rtn><m
sg>'Arrondi à deux décimales'!

END
```

Quelques mots d'explication sur ces macros. Elles tirent parti de trois formules classiques :

- pour l'arrondi à 2 décimales : àINT(((Expression) + ,005)*100)/100
- pour OR : àIF(Expr1;1;(àIF(Expr2;1;0))) - il suffit que Expr1 ou Expr2 soit vérifiée pour que la formule renvoie 1.
- pour AND : àIF(Exp1;(àIF(Exp2;1;0);0)) - il faut que Expr1 et Expr2 soient vérifiées pour que la formule renvoie 1.

L'introduction des expressions se fait par la fonction <input> : la macro s'arrête, et attend l'entrée de texte au clavier. La macro reprend aussitôt après un 'Return' au clavier (qui n'est pas ajouté à l'input). Pour que la macro soit compréhensible, il faut faire précéder <input> de la fonction <msg>, qui permet d'afficher un message au bas de l'écran, en caractères normaux (" "), inverses (' ') ou MouseText (* *). Dans le cas des macros ci-dessus, la série d'espaces entre " " efface le début de la ligne (où va s'inscrire l'input), et le message proprement dit, entre ' ', est volontairement déporté vers la droite.

Notes :

- attention aux apostrophes dans les messages en vidéo inverse : remplacez-les par un '°' si vous ne voulez pas que la suite du message se retrouve dans les endroits les plus inattendus...
- <input> accepte non seulement l'entrée de caractères, mais les déplacements par les flèches et la souris.

Voici, avant de vous quitter, une autre macro fort utile pour tous ceux qui font régulièrement de la rédaction :

```
*****************************************
      APPLEWORKS 1.4 - SuperMacroWorks
              COMPTE-MOTS

Cette macro permet de compter rapidement le
nombre de mots dans un fichier de Traitement de
texte. Par "mot" il faut entendre tout groupe
de caractères entre deux espaces.

Le compte se fait du curseur jusqu'à deux
caractères £ précédés d'un espace. Compiler
avec Ö=, placer les caractères £ et le curseur
aux endroits voulus, taper ÖS. Le nombre de
mots s'inscrira à la fin, à la place du double
£.

Cette macro utilise les mémoires de la Macro 0,
et les mémoires supplémentaires 1, 2, 3 et 4
pour stocker les valeurs des unités, dizaines,
centaines et milliers pendant le décompte.

*****************************************
Dimitri Geystor - Décembre 1987

START
Les macros suivantes s'appellent en cascade:

S:<all><insert><0=>0<save0>1<save0>2<save0>3<sa
-M>!  On démarre

M:<all><sa-C><0=>1<save0>4<sa-C><0=>2<save0>4<s
a-C><0=>3<save0>4!  Milliers

C:<all><sa-D><0=>1<save0>3<sa-D><0=>2<save0>3<s
a-D><0=>3<save0>3<sa-D> <0=>4<save0>3<sa-D><0=>
5<save0>3<sa-D><0=>6<save0>3<sa-D><0=>7<save0>3
 <sa-D><0=>8<save0>3<sa-D><0=>9<save0>3<sa-D><0
=>0<save0>3!  Centaines

D:<all><sa-U><0=>1<save0>2<sa-U><0=>2<save0>2<s
a-U><0=>3<save0>2<sa-U> <0=>4<save0>2<sa-U><0=>
5<save0>2<sa-U><0=>6<save0>2<sa-U><0=>7<save0>2
 <sa-U><0=>8<save0>2<sa-U><0=>9<save0>2<sa-U><0
=>0<save0>2!  Dizaines

U:<all><sa-K><0=>1<save0>1<sa-K><0=>2<save0>1<s
a-K><0=>3<save0>1<sa-K> <0=>4<save0>1<sa-K><0=>
5<save0>1<sa-K><0=>6<save0>1<sa-K><0=>7<save0>1
 <sa-K><0=>8<save0>1<sa-K><0=>9<save0>1<sa-K><0
=>0<save0>1!  Unités

K:<all><oa-right><sa-£>!  Compteur

£:<all><if>£<right><if>£<del>(<spc>=<spc><load0
>4<sa-0><load0>3<sa-0>< load0>2<sa-0><load0>1<s
a-0><spc>mots<right><del><spc>)<stop>!  Test et
 retour, ou arrêt et résultat
END
```

Pour l'explication du mécanisme de ces macros (comme pour toutes les macros en cascade), voir le manuel page 16. La macro appelée renvoie au point d'appel de la macro appelante :

- quand toutes les commandes qu'elle contient ont été exécutées ;

ou

- quand une condition qu'elle contient n'est pas remplie (si la condition est remplie, elle se poursuit normalement).

Note :

dans la macro "Compte-mots", certains groupes de caractères se répètent régulièrement, à peu de chose près. C'est l'occasion ou jamais d'utiliser, pour la copie au clavier, une autre particularité de la Macro 0 : écrire le groupe de caractères une seule fois, mettre le curseur sur le premier caractère, et répéter Ö/ (c'est-à-dire <read>) jusqu'à la fin du groupe à recopier. Maintenant la Macro 0 contient ce groupe, qu'on peut reproduire autant de fois qu'on le désire en faisant Ö0. Puis, en mode recouvrement, on apportera rapidement les quelques retouches supplémentaires.

Dans le prochain numéro, nous vous présenteront une macro plus élaborée, avec recours à des macros conditionnelles, au chaînage et à l'incrémentation de variables.

Elle permettra, en une seule commande Ö, d'aller chercher une adresse dans une *base de données* d'adresses, d'ouvrir un fichier *traitement de texte* et de faire la mise en page d'une lettre-type contenant votre en-tête, l'adresse du destinataire choisi et la date ; petit raffinement : cette macro par la même occasion (et sans intervention supplémentaire) renommera le fichier *traitement de texte* ouvert en lui donnant le nom qui figure dans l'adresse...

Les deux fichiers de macros AND.OR.ROUND et COMPTE.MOTS, prêts à l'emploi, sont inclus dans la disquette d'accompagnement de ce numéro de Pom's.

HGR/Minitel sur //c

Sur les Apple //c, la configuration de l'interface est perdue à chaque PR£. Aussi, dans le programme de transformation d'images HGR en images semi-graphiques Minitel (Pom's 33), il convient de modifier les lignes suivantes :

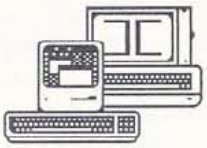
```
40080 PRINT CHR$(4)"PR£2": PRINT
    CHR$(1)"1D"CHR$(1)"3P"CHR$(1
    )"8B"
63040 PRINT CHR$(4)"PR£2": PRINT
    CHR$(1)"1D"CHR$(1)"3P"CHR$(1
    )"8B"
```

Sur la disquette d'accompagnement de la revue, ce petit problème avait été réglé.

J.-L. Bazanegue, C. Piard

Pom_Link 2.1 :

40 instructions qui servent



Objectivement, l'importance et les ressources des librairies Pom_Link ne les destinaient pas *a priori* à une publication dans les pages de Pom's mais plutôt en logiciel édité à part. Pom's se devait toutefois à la fidélité de ses lecteurs ; aussi, ce serveur Vidéotex vous est proposé en deux versions qui satisferont les besoins de chacun.

La version 2.1 dont nous discuterons ici regroupe une quarantaine d'instructions nouvelles, orientées télématique, ajoutées à celles de votre Basic (AppleSoft sur l'Apple //, Basic Microsoft 2.0 et plus sur le Macintosh). Cette version comporte le nécessaire à la réalisation d'un véritable serveur Vidéotex qui n'aura rien à envier aux serveurs du 3615. Pom_Link 2.1 est publiée dans ces pages, donc aussi sur les disquettes d'accompagnement Macintosh et Apple //. Les sources de la version Macintosh, beaucoup trop volumineux, n'ont pas pu être placés dans ces pages, mais ils sont toutefois sur la disquette Mac 34.

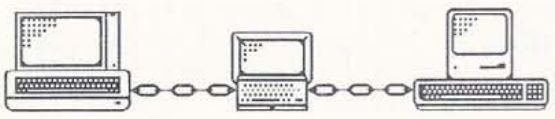
La version 3.0, disponible en *produit à part*, ouvre la porte au mode téléinformatique (style CalvaCom, mode ASCII 80 colonnes) mais va également plus loin dans l'exploitation du mode Vidéotex. Elle donne également à l'écran de l'ordinateur serveur un écho des entrées/sorties (utiles pour rechercher les bugs).

Pourquoi des librairies ?

Certes, Pom's pouvait publier comme produit unique et final le répondeur télématique donné en exemple mais le résultat aurait été figé. De nouvelles instructions ajoutées au Basic permettent à chacun :

- de modifier simplement ce répondeur et de l'adapter à ses besoins ;
- de créer son propre serveur spécialisé.

Les librairies permettront de mettre en œuvre votre imagination. De plus, ne doutons pas qu'elles serviront de base à bien d'autres programmes dans les prochains numéros.



Que font-elles ?

Chaque routine Pom_Link a une fonction d'envoi et/ou de réception de caractères vers ou depuis le Minitel 'servi'. Ainsi, il est plus agréable d'écrire dans son programme Basic :

```
DEPLACE 1, 16
```

pour positionner le curseur en ligne 1, colonne 16 plutôt que la suite d'instructions :

```
ouvrir port série en sortie
PRINT CHR$(31) CHR$(64+1) CHR$(64+16)
fermer port série
```

Réaliser un 'INPUT' sur le Minitel servi en Basic est quasi irréalisable alors que celui proposé ici est déjà beaucoup plus puissant que celui de l'AppleSoft. Chaque nouvelle fonction offre à chacun la possibilité de piloter un serveur sans pour autant maîtriser les égarements de la norme (?) Vidéotex ; il faut toutefois en dire un mot.

Le Vidéotex

Dans le descriptif de chaque instruction figurent souvent des remarques du fait de contraintes Vidéotex, et il faudra pardonner aux librairies de n'avoir pas totalement supprimé la lourdeur du système. La couleur du fond qui repasse au noir lors d'un déplacement, le soulignement qui n'est validé qu'après affichage d'un espace par exemple sont à prendre en considération lors de la programmation. Conséquence de ces petites contraintes, certaines instructions ne fonctionne pas – ou au moins pas normalement pour certaines – en *local*, c'est-à-dire Minitel serveur non connecté : cela compliquera légèrement la mise au point.

Mise en œuvre...

Une syntaxe commune à l'Apple // et au Macintosh à été retenue et, pour ne pas dupliquer les exemples, voici le mode d'utilisation sur chaque appareil :

...sur l'Apple //

La première instruction du programme Basic serveur doit être la suivante :

```
IF PEEK (104) * 256 + PEEK (103) < >
  7681 THEN POKE 103,1: POKE 104,30:
  POKE 7680,0: PRINT CHR$(4) "-nom"
```

où *nom* est le nom de votre programme Basic.

La ligne suivante sera :

```
PRINT  CHR$(4)"-POM.LINK.2.1"
```

Les nouvelles instructions sont alors disponibles avec la syntaxe suivante :

Instruction dans le mode d'emploi de Pom_Link :

```
ALERTE "interdit",1
```

Sur l'Apple // :

```
& ALERTE, "interdit",1
```

Autrement dit, les mots-clé des nouvelles instructions sont précédés de "&" et suivis de ",".

...sur le Macintosh

L'instruction suivante devra être placée avant l'occurrence de la première instruction de Pom_Link utilisée :

```
LIBRARY "Pom_Link 2.1"
```

Dès lors, les nouvelles instructions sont disponibles telles qu'elles sont décrites dans leur mode d'emploi :

```
ALERTE "Interdit",1
```

à condition toutefois d'avoir placé au début de votre programme l'instruction du Basic Microsoft :

```
DEFSNG A-Z
```

ou

```
DEFDBL A-Z
```

Si vos habitudes vous conduisent à utiliser `DEFINT` (ce qui est, à notre humble avis, la meilleure solution) ou `DEFSTR`, il vous faudra placer un point exclamation '!' ou – mais moins efficace – un dièse '#' juste après le nom de l'instruction. Vous obtiendrez alors :

```
ALERTE! "Interdit",1
ALERTE# "Interdit",1
```

Notons enfin que le nom des instructions peut être saisi indifféremment en majuscules ou minuscules ; ainsi :

```
AfficheR! "Bonjour chez vous..."
```

est aussi valide que

```
AFFICHER! "Bonjour chez vous..."
```

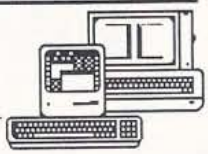
Optimiser HGR/Minitel

En ajoutant la ligne suivante au programme de conversion HGR/Minitel du numéro 33 de Pom's, on gagne un temps précieux :

50135 IF MN% THEN 50160

Pour convertir Lady, il faut avec cette ligne 2'55, et 4'45 sans...

G. Lauvergnier, Rennes



ACCEPT (ne fonctionne pas en local)

Syntaxe de l'instruction

ACCEPT *colonne,ligne,mode,sollicitation,longueur, retour,état*

But

Gère une zone de saisie contrôlée.

Explications

colonne et *ligne* sont des variables ou expressions entières indiquant la position de la zone de saisie sur l'écran du Minitel. *colonne* doit être compris entre 1 et 39 inclus. *ligne* doit être compris entre 1 et 24 inclus.

mode est une variable ou expression entière désignant la taille des caractères utilisés :

- mode = 0 : taille normale ;
- mode = 1 : double hauteur.

sollicitation est une variable ou expression chaîne qui sera affichée avant la zone de saisie proprement dite. Le nombre de caractères de *sollicitation* doit être compris entre 0 et 39 inclus.

longueur indique le nombre maximum de caractères qu'il sera possible d'entrer dans la zone de saisie. Afin de matérialiser la zone, ACCEPT affiche autant de points que nécessaire. *longueur* doit être compris entre 1 et 39 inclus.

retour est une variable chaîne dans laquelle sera retournée la chaîne de caractères entrée. Si la variable chaîne *retour* n'est pas vide, son contenu sera utilisé comme texte 'par défaut' et affiché en début de zone de saisie. La longueur de la chaîne de caractères représentée par *retour* ne peut pas être supérieure à la longueur de la zone de saisie.

état est une variable entière indiquant les conditions de sortie de l'instruction :

- état = 0 : pas de caractère saisi depuis plus de 2 minutes (voir le paragraphe "Côté utilisateur") ;
- état = 1 : sortie par ENVOI ou 'retour-chariot' ;
- état = 2 : sortie par RETOUR ;
- état = 3 : sortie par REPETITION ;
- état = 4 : sortie par GUIDE ;
- état = 5 : sortie par ANNULATION (voir le paragraphe "Côté utilisateur") ;
- état = 6 : sortie par SOMMAIRE ;
- état = 7 : sortie par SUITE ;
- état = 8 : sortie par CONNEXION.

Remarques

- la somme de *colonne* + longueur de *sollicitation* + *longueur* ne doit pas être supérieure à 40 ;
- ACCEPT provoque le passage en mode texte ;
- le curseur est invisible lors du retour au programme Basic.

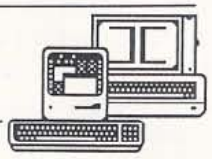
Côté utilisateur

- ACCEPT place le curseur sur la première position de la zone de saisie ou, si une chaîne par défaut est utilisée, sur la position suivant cette chaîne ;
- si l'on tente d'entrer plus de caractère que n'en prévoie la zone de saisie, ACCEPT émet un 'bip' et affiche le message "Fin de zone" dans la ligne 0 ;
- la touche CORRECTION déplace le curseur d'une position vers la gauche en effaçant un caractère. Si le curseur se trouve sur la première position de la zone de saisie, une action sur la touche CORRECTION provoque l'affichage du message "Rien à effacer" dans la ligne 0 ;
- si le curseur ne se trouve pas sur la première position de la zone de saisie, une action sur la touche ANNULATION vide la zone de saisie et place le curseur sur la première position. Si le curseur se trouve sur la première position de la zone de saisie, ANNULATION provoque un retour au programme Basic avec *état* = 5 ;
- toutes les touches de fonction provoquent un retour au programme Basic sauf CORRECTION, et ANNULATION dans le cas cité précédemment ;
- si le clavier du Minitel n'est pas sollicité pendant une minute, ACCEPT émet un 'bip' et affiche le message "Déconnexion imminente" dans la ligne 0. Si rien ne se passe pendant la minute suivante, il y a retour au programme Basic avec *état* = 0 ;
- avant retour au programme Basic, les points de la zone de saisie situés sur des positions inutilisées sont effacés ;
- lorsqu'un message est affiché dans la ligne 0, une action sur le clavier du Minitel provoque l'effacement de ladite ligne ;
- les caractères ',' et ':' sont remplacés par un point ('.') (uniquement avec l'Apple // comme serveur).

Exemples

```
10 REM Exemples d'utilisation d'ACCEPT
20 ACCEPT 5,8,1,"Nom : ",12,CH$,ET%
30 ACCEPT CO%,LI%+1,1,A$+": ",10,C$,ET%
40 ON ET%+1 GOSUB 100,200,300,400,500,...
```

Voir aussi ACCEPTF, SAISIE et SECRET.



ACCEPTF (ne fonctionne pas en local)

Syntaxe de l'instruction
ACCEPTF *état*

But

Attend pendant 2 minutes maximum une action sur une touche de fonction.

Explications

état est une variable entière contenant, au retour, le code de la touche de fonction :

- état = 0 : pas de caractère saisi depuis plus de 2 minutes (voir le paragraphe "Côté utilisateur") ;
- état = 1 : ENVOI ou 'retour-chariot' ;
- état = 2 : RETOUR ;
- état = 3 : REPETITION ;
- état = 4 : GUIDE ;
- état = 5 : ANNULATION ;
- état = 6 : SOMMAIRE ;
- état = 7 : SUITE ;
- état = 8 : CONNEXION ;
- état = 9 : CORRECTION.

Côté utilisateur

Si le clavier du Minitel n'est pas sollicité pendant une minute, ACCEPT émet un 'bip' et affiche le message "Déconnexion imminente" dans la ligne 0. Si rien ne se passe pendant la minute suivante, il y a retour au programme Basic avec *état* = 0.

Exemple

```
10 REM Exemple d'utilisation d'ACCEPTF
20 ACCEPTF ET%
```

Voir aussi ACCEPT, SAISIE et SECRET.

AFFICHE

Syntaxe de l'instruction
AFFICHE *chaîne* [;] (sur l'Apple //)
AFFICHE *chaîne* (sur le Macintosh)

But
Affiche une chaîne de caractères sur l'écran du Minitel utilisateur, à partir de la position courante du curseur.

Explications
chaîne est une variable ou une expression chaîne. Les caractères portant des codes différents sur l'ordinateur-serveur et le Minitel sont recodés. Sur l'Apple //, AFFICHE fonctionne comme un PRINT : un retour chariot est généré s'il n'y a pas de point-virgule à la fin de l'instruction ; seule la chaîne est affichée dans le cas contraire. Sur le Macintosh, seule la chaîne est affichée.

Remarque
Aux éventuels 'retour-chariot' (ASCII 13) contenus par la chaîne de caractères sont ajoutés des 'line-feed' (ASCII 10). Ceci autorise l'envoi direct d'un fichier. Par exemple, sur le Macintosh :

```
AFFICHE  INPUT$(LOF(1),1)
```

envoi la totalité du contenu du fichier #1 sur l'écran du Minitel.

Exemples

```
10 REM Exemples d'utilisation d'AFFICHE
20 AFFICHE "Début"; (uniquement Apple //)
30 AFFICHE "Page "+STR$(NP%)
40 AFFICHE C$
```

Voir aussi AFFICHER, ALERTE, CHARIOT, MTH.

AFFICHER (uniquement Macintosh)

Syntaxe de l'instruction
AFFICHER *chaîne*

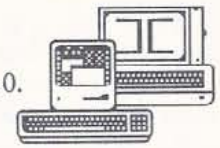
But
Identique à AFFICHE mais ajoute un 'retour-chariot' à la fin de la chaîne de caractères.

Voir aussi AFFICHE, ALERTE, CHARIOT.

ALERTE

Syntaxe de l'instruction
ALERTE *message,bip*

But
Affiche un message d'alerte dans la ligne 0.



Explications
message est une variable ou une expression chaîne. la longueur de la chaîne de caractères doit être comprise entre 0 et 36 inclus. Les caractères portant des codes différents sur l'ordinateur-serveur et le Minitel sont recodés.
bip est une variable ou expression entière qui indique si le message doit être accompagnés d'un 'bip' sonore :
bip = 0 : pas de 'bip' ;
bip <> 0 : émission d'un 'bip'.

Remarques
- la ligne 0 est effacée avant affichage du message d'alerte ;
- il est possible d'utiliser ALERTE pour effacer la ligne 0 en passant une chaîne de longueur 0 comme *message* ;
- la position courante du curseur est rétablie après exécution de l'instruction ALERTE. De même, l'ensemble des attributs Vidéotex sont préservés et restitués.

Exemples

```
10 REM Exemples d'utilisation d'ALERTE
20 ALERTE "Touche"+A$(T%)+"invalide",B%
30 ALERTE CH$,0
40 ALERTE "Communication interrompue",1
```

Voir aussi AFFICHE, AFFICHER, MTH.

APPEL

Syntaxe de l'instruction
APPEL *état*

But
Détecte la présence – ou l'absence – d'un appel téléphonique.

Explications
état est une variable entière contenant, au retour :
- 0 si aucun appel est détecté ;
- 1 si un appel est détecté (–1 sur le Macintosh).

Remarque
Si APPEL détecte quelque chose, l'instruction s'assure de la validité de l'appel en éliminant les impulsions inférieures à 25 centièmes de seconde.

Exemples

```
10 REM Exemples d'utilisation d'APPEL
20 APPEL ET%
30 IF NOT ET% THEN 20
40 REM APPEL
```

BAS (ne fonctionne pas en local)

Syntaxe de l'instruction
BAS *nombre*

But
Décale l'écran Vidéotex vers le bas.

Explication
nombre est une variable ou expression entière qui indique l'amplitude du décalage en lignes. *nombre* doit être compris entre 0 et 24 inclus.

Remarque
BAS vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).

Exemples
```
10 REM Exemples d'utilisation de BAS
20 BAS 10
30 BAS N%+10
40 BAS N$(A%)
```

Voir aussi HAUT.

BIP

Syntaxe de l'instruction
BIP

But
Provoque l'émission d'un 'bip' sonore.

Exemple
```
10 REM Exemple d'utilisation de BIP
20 BIP
```

Voir aussi ALERTE.

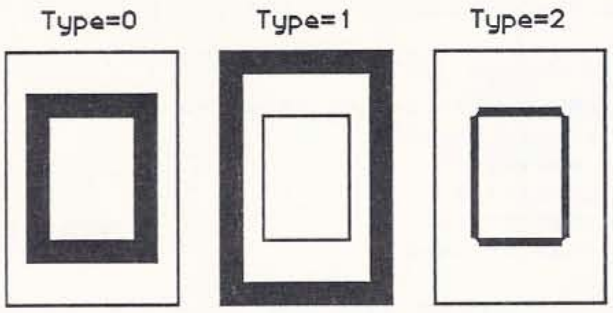
CADRE

Syntaxe de l'instruction
CADRE *x1,y1,x2,y2,type,lignage,couleur C,couleur F*

But
Trace un cadre à l'écran du Minitel utilisateur.

Explication
x1, *y1*, *x2* et *y2* : coordonnées du rectangle dans lequel doit s'inscrire le cadre. *x1* et *x2* doivent être compris entre 1 et 40 inclus. *y1* et *y2* doivent être compris entre 1 et 24 inclus.

type est une variable ou expression entière qui indique le type de cadre à afficher :

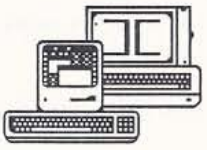


lignage est une variable ou expression entière qui indique si le cadre doit être tracé avec des caractères semi-graphiques joints ou disjoints :
- lignage = 0 : caractères joints ;
- lignage <> 0 : caractères disjoints.

Si *Type* = 2, *lignage* est ignoré.

couleur C est une variable ou expression entière représentant la couleur des caractères ;

couleur F est une variable ou expression entière représentant la couleur de fond :
- 0 : noir 0% de gris
- 1 : rouge 50% de gris
- 2 : vert 70% de gris
- 3 : jaune 90% de gris
- 4 : bleu 40% de gris
- 5 : magenta 60% de gris
- 6 : cyan 80% de gris
- 7 : blanc 100% de gris



Remarques
- le rectangle minimum est de 3 caractères par 3 ;
- après affichage du cadre, le curseur est placé dans l'angle supérieur gauche du cadre ;
- lors du retour au programme Basic, le Minitel utilisateur est en mode texte, curseur invisible.

Exemples
```
10 REM Exemples d'utilisation de CADRE
20 CADRE X1%,Y1%,X2%,Y2%,T%,L%,C%,F%
30 CADRE 10,A%+9,B%,24,1,1,C%+2,F%(A%)
```

Voir aussi VIDRECT.

CHARIOT

Syntaxe de l'instruction
CHARIOT [*nombre*]

But
Provoque *nombre* 'retour-chariot'.

Explications
nombre est une variable ou expression entière facultative

qui indique le nombre de retours à la ligne à effectuer. Si *nombre* est omis, CHARIOT provoque un seul retour à la ligne. *nombre* doit être compris entre 0 et 24 inclus.

Exemples

```
10 REM Exemples d'utilisation
20 REM de CHARIOT
30 CHARIOT
40 CHARIOT 3
50 CHARIOT N%
60 CHARIOT N%+2
```

Voir aussi AFFICHE, AFFICHER.

CNXN

Syntaxe de l'instruction
CNXN

But
Connexion du Minitel serveur.

Exemple

```
10 REM Exemple d'utilisation de CNXN
20 CNXN
```

Voir aussi CNXR, DECNX.

CNXR (ne fonctionne pas en local)

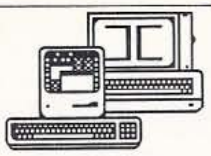
Syntaxe de l'instruction
CNXR *état*

But
Retournement et connexion du Minitel serveur.

Explication
Cette instruction est normalement utilisée lorsqu'on a détecté un appel téléphonique (à l'aide de l'instruction APPEL).
état est une variable entière dans laquelle on pourra trouver au retour dans le programme Basic :

- état = 0 si le Minitel de l'utilisateur n'a pas été connecté au bout d'une minute (dans ce cas, le Minitel serveur est déconnecté) ;
- état = 1 (–1 sur le Macintosh) si le Minitel de l'utilisateur a été connecté (dans ce cas, un message de 'copyright' est affiché pendant 4 secondes sur l'écran du Minitel utilisateur).

Remarque
CNXR vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).



Exemple

```
10 REM Exemple d'utilisation de CNXR
20 CNXR ET%
```

Voir aussi APPEL, CNXN, DECNX.

CURINVIS

Syntaxe de l'instruction
CURINVIS

But
Rend le curseur invisible.

Exemple

```
10 REM Exemple d'utilisation
20 REM de CURINVIS
30 CURINVIS
```

Voir aussi CURVIS.

CURVIS

Syntaxe de l'instruction
CURVIS

But
Rend le curseur visible.

Exemple

```
10 REM Exemple d'utilisation de CURVIS
20 CURVIS
```

Voir aussi CURINVIS.

DECNX

Syntaxe de l'instruction
DECNX

But
Déconnexion du Minitel serveur et du Minitel utilisateur.

Exemple

```
10 REM Exemple d'utilisation de DECNX
20 DECNX
```

Voir aussi CNXN, CNXR.

DEPLACE

Syntaxe de l'instruction
DEPLACE *colonne,ligne*

But
Positionnement du curseur.

Explications
colonne et *ligne* sont des variables ou expressions entières indiquant la position du curseur. *colonne* doit être compris entre 1 et 40 inclus. *ligne* doit être compris entre 1 et 24 inclus.

Exemples
```
10 REM Exemples d'utilisation
20 REM de DEPLACE
30 DEPLACE 2,10
40 DEPLACE 2,LI%
50 DEPLACE CO%+2,L%(N%+1)
```

Voir aussi LOCALISE.

ENLIGNE (ne fonctionne pas en local)

Syntaxe de l'instruction
ENLIGNE *état*

But
Permet de s'assurer de la présence du Minitel utilisateur.

Explication
état est une variable entière dans laquelle on pourra trouver au retour dans le programme Basic :
- état = 0 si le Minitel utilisateur ne répond pas ;
- état = 1 (–1 sur le Macintosh) si le Minitel utilisateur répond normalement.

Remarque
ENLIGNE vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).

Exemple
```
10 REM Exemple d'utilisation de ENLIGNE
20 ENLIGNE ET%
```

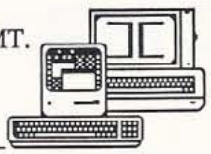
FIXE

Syntaxe de l'instruction
FIXE

But
Passe en affichage non clignotant.

Exemple
```
10 REM Exemple d'utilisation de FIXE
20 FIXE
```

Voir aussi FLASH, PARAMG, PARAMT.



FLASH

Syntaxe de l'instruction
FLASH

But
Passe en affichage clignotant.

Exemple
```
10 REM Exemple d'utilisation de FLASH
20 FLASH
```

Voir aussi FIXE, PARAMG, PARAMT.

GR

Syntaxe de l'instruction
GR

But
Passe en mode semi-graphique.

Exemple
```
10 REM Exemple d'utilisation de GR
20 GR
```

Voir aussi PARAMG, PARAMT, TXT.

HAUT (ne fonctionne pas en local)

Syntaxe de l'instruction
HAUT *nombre*

But
Décale l'écran Vidéotex vers le haut.

Explication
nombre est une variable ou expression entière qui indique l'amplitude du décalage en lignes. *nombre* doit être compris entre 0 et 24 inclus.

Remarque
HAUT vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).

Exemples

```
10 REM Exemples d'utilisation de HAUT
20 HAUT 10
30 HAUT N%+10
40 HAUT N%(A%)
```

Voir aussi BAS.

IDENT (ne fonctionne pas en local)

Syntaxe de l'instruction
IDENT *valeur 1,valeur 2,valeur 3,état*

But
Retourne les trois octets d'identification du Minitel utilisateur.

Explication
valeur 1 est une variable entière dans laquelle IDENT retourne le code constructeur du Minitel utilisateur ;
valeur 2 est une variable entière dans laquelle IDENT retourne le type du Minitel utilisateur ;
valeur 3 est une variable entière dans laquelle IDENT retourne le numéro de version du Minitel utilisateur ;
état est une variable entière dans laquelle on pourra trouver au retour dans le programme Basic :

- état = 0 si le Minitel utilisateur n'a pas répondu ;
- état = 1 (–1 sur le Macintosh) si le Minitel utilisateur a répondu normalement.

Remarques

- si *état* = 0, le contenu de *valeur 1*, *valeur 2* et *valeur 3* est indéfini.
- sur le Macintosh, IDENT vide le tampon d'entrée des caractères (voir PURGE).
- **certains Minitel sont munis d'un logiciel qui ne permet pas l'émission des identificateurs ; dans ce cas *état* sera égal à 0.**

Exemple

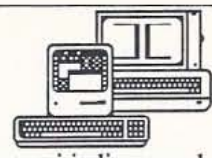
```
10 REM Exemple d'utilisation de IDENT
20 IDENT V1%,V2%,V3%,ET%
```

INIT (uniquement sur l'Apple //)

Syntaxe de l'instruction
INIT *slot*

But
Initialise les routines sur le slot indiqué et vérifie l'existence du port série.

Explication



slot est une variable ou expression entière qui indique quel port série devra être utilisé pour les entrées/sorties.

Exemples

```
10 REM Exemples d'utilisation de INIT
20 INIT 2
30 INIT S%
```

Voir aussi SERVOFF.

INVERSE

Syntaxe de l'instruction
INVERSE

But
Passe en affichage inversé.

Exemple

```
10 REM Exemple d'utilisation de INVERSE
20 INVERSE
```

Voir aussi NORMAL, PARAMG, PARAMT.

LIGNE

Syntaxe de l'instruction
LIGNE

But
Initialise le mode lignage ou souligné.

Explications

- si le Minitel utilisateur est en mode semi-graphique, LIGNE passe l'affichage en mode ligné (caractères semi-graphiques disjoints) ;
- si le Minitel utilisateur est en mode texte, LIGNE passe l'affichage en mode souligné ;

Remarque
Dans le cas du mode texte, LIGNE envoie un attribut latent qui ne sera validé par le Minitel utilisateur qu'à la réception d'un espace (ASCII 32).

Exemple

```
10 REM Exemple d'utilisation de LIGNE
20 LIGNE
```

Voir aussi PARAMG, PARAMT, SANSLIGNE.

LOCALISE (ne fonctionne pas en local)

Syntaxe de l'instruction
LOCALISE *colonne,ligne,état*

But
Retourne la position courante du curseur.

Explication
colonne et *ligne* sont des variables entières dans lesquelles LOCALISE retourne la position courante du curseur.
état est une variable entière dans laquelle on pourra trouver au retour dans le programme Basic :
- état = 0 si le Minitel utilisateur ne répond pas ;
- état = 1 (–1 sur le Macintosh) si le Minitel utilisateur répond normalement.

Remarques
- si *état* = 0, le contenu de *colonne* et *ligne* est indéfini.
- LOCALISE vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).

Exemple
```
10 REM Exemple d'utilisation
20 REM de LOCALISE
30 LOCALISE CO%,LI%,ET%
```

Voir aussi DEPLACE.

MODE

Syntaxe de l'instruction
MODE *mode*

But
Change le mode d'affichage.

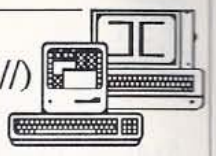
Explications
mode est une variable ou expression entière indiquant le mode d'affichage à utiliser :
- mode = 0 : caractères normaux ;
- mode = 1 : double hauteur ;
- mode = 2 : double largeur ;
- mode = 3 : double taille.

Remarque
MODE provoque le passage en mode texte.

Exemples
```
10 REM Exemples d'utilisation de MODE
20 MODE M%
30 MODE M%+A%(B%)
40 MODE 2
```

Voir aussi PARAMT.

MTH (uniquement sur l'Apple //)



Syntaxe de l'instruction
MTH *mode*

But
Modifie le recodage des caractères issus de l'instruction AFFICHE.

Explication
mode est une variable ou expression entière qui indique le type de recodage de certains caractères :
- mode = 0 : mode littéraire (par défaut) ;
- mode <> 0 : mode mathématique.

Apple //	Minitel littéraire	Minitel mathématique	
é	é	|	
è	è	|	
ç	ç	\	
à	à	@	
§	_	]	
°	°	[	
ù	ù	|	
£	£	#	
^a	â	↑a	autres voyelles accentuées : idem
¨a	ä	a̲	autres voyelles accentuées : idem
^t	t	↑t	lettres non accentuables : idem

Remarque
Tous les Minitels n'ayant pas les mêmes ROM d'affichage, les résultats peuvent varier légèrement d'un modèle à l'autre.

Exemples
```
10 REM Exemples d'utilisation de MTH
20 MTH 1
30 MTH M%
```

NORMAL

Syntaxe de l'instruction
NORMAL

But
Passe en affichage non inversé.

Exemple
```
10 REM Exemple d'utilisation de NORMAL
20 NORMAL
```

Voir aussi INVERSE, PARAMG, PARAMT.

PARAMG

Syntaxe de l'instruction
PARAMG *couleur C,couleur F,lignage,flash*

But
Passe en mode semi-graphique et établit les paramètres d'affichage.

Explications
couleur C est une variable ou expression entière représentant la couleur des caractères ;

couleur F est une variable ou expression entière représentant la couleur de fond ;

- 0 : noir 0% de gris
- 1 : rouge 50% de gris
- 2 : vert 70% de gris
- 3 : jaune 90% de gris
- 4 : bleu 40% de gris
- 5 : magenta 60% de gris
- 6 : cyan 80% de gris
- 7 : blanc 100% de gris

lignage est une variable ou expression entière qui indique si les caractères doivent être affichés avec des caractères semi-graphiques joints ou disjoints :

- lignage = 0 : caractères joints ;
- lignage <> 0 : caractères disjoints.

flash est une variable ou expression entière déterminant la fixité ou le clignotement (inversion des points du caractère une fois par seconde) :

- flash = 0 : caractères fixes ;
- flash <> 0 : caractères clignotants.

Exemples

```
10 REM Exemples d'utilisation de PARAMC
20 PARAMG CC%,CF%,LI%,FL%
30 PARAMG 7,0,1,1
40 PARAMG CC%+1,A%(CF%),INT(L%/2),0
```

Voir aussi FIXE, FLASH, GR, LIGNE, PARAMT, SANSLIGNE, TXT.

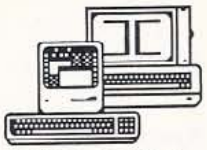
PARAMT

Syntaxe de l'instruction
PARAMT *mode,couleur C,couleur F, flash inversé,souligné*

But
Passe en mode texte et établit les paramètres d'affichage.

Explications
mode est une variable ou expression entière désignant la taille des caractères :

- mode = 0 : caractères normaux ;
- mode = 1 : double hauteur ;
- mode = 2 : double largeur ;
- mode = 3 : double taille.



couleur C est une variable ou expression entière représentant la couleur des caractères ;

couleur F est une variable ou expression entière représentant la couleur de fond ;

- 0 : noir 0% de gris
- 1 : rouge 50% de gris
- 2 : vert 70% de gris
- 3 : jaune 90% de gris
- 4 : bleu 40% de gris
- 5 : magenta 60% de gris
- 6 : cyan 80% de gris
- 7 : blanc 100% de gris

flash est une variable ou expression entière déterminant la fixité ou le clignotement (inversion des points du caractère une fois par seconde) :

- flash = 0 : caractères fixes ;
- flash <> 0 : caractères clignotants.

inversé est une variable ou expression entière indiquant si les couleurs de caractère et de fond doivent être inversés ou non :

- inversé = 0 : couleurs normales ;
- inversé <> 0 : couleurs inversées.

souligné est une variable ou expression entière qui indique si les caractères doivent être soulignés :

- souligné = 0 : caractères non soulignés ;
- souligné <> 0 : caractères soulignés.

Remarque
Les paramètres *couleur F* et *souligné* constituent des attributs latents qui ne seront validés par le Minitel utilisateur qu'à la réception d'un espace (ASCII 32).

Exemples

```
10 REM Exemples d'utilisation de PARAMT
20 PARAMT M%,CC%,CF%,FL%,IN%,SO%
30 PARAMT 2,0,7,1,1,1
40 PARAMT N%+(V%<3),0,7,1,1,SO%<>10
```

Voir aussi FIXE, FLASH, INVERSE, LIGNE, GR, NORMAL, PARAMG, SANSLIGNE, TXT.

PURGE (uniquement sur le Macintosh)

Syntaxe de l'instruction
PURGE

But
Vide le tampon d'entrée des caractères.

Explications
Les instructions de cette librairie qui reçoivent des caractères utilisent un tampon d'entrée capable de stocker

1024 caractères. Celui-ci permet à l'utilisateur d'anticiper les commandes ou le remplissage des zones de saisie. Cependant, suivant le type de traitement effectué par le serveur, il peut être utile de vider le tampon d'entrée pour être sûr de n'exploiter que des caractères 'frais' (par exemple avec SAISIE).

Remarque

Les instructions suivantes vident le tampon d'entrée :

- CNXR
- BAS
- ENLIGNE
- HAUT
- IDENT
- LOCALISE
- ROULEAU

Exemple

```
10 REM Exemple d'utilisation de PURGE
20 PURGE
```

REINIT

Syntaxe de l'instruction

REINIT

But

Réinitialisation du Minitel utilisateur.

Explications

Le Minitel utilisateur passe en :

- mode Vidéotex
- caractères blanc
- fond noir
- taille normale
- non souligné
- non inversé
- non flash

Remarque

REINIT émet un espace (ASCII 32) afin de valider le mode non souligné et la couleur de fond.

Exemple

```
10 REM Exemple d'utilisation de REINIT
20 REINIT
```

Voir aussi FIXE, FLASH, LIGNE, PARAMT, SANSLIGNE, TXT.

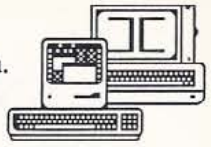
ROULEAU (ne fonctionne pas en local)

Syntaxe de l'instruction

ROULEAU *condition*

But

Passage en mode page ou en mode rouleau.



Explication

condition est une variable ou expression entière qui indique le type d'affichage à utiliser :

- condition = 0 : mode page ;
- condition <> 0 : mode rouleau.

Remarque

ROULEAU vide le tampon d'entrée des caractères (voir PURGE pour le Macintosh).

Exemples

```
10 REM Exemples d'utilisation
20 REM de ROULEAU
30 ROULEAU 1
40 ROULEAU R%
50 ROULEAU R%=0
```

SAISIE

Syntaxe de l'instruction

SAISIE *caractère*

But

Retourne une chaîne contenant le caractère frappé sur le clavier du Minitel utilisateur, ou une chaîne vide si aucun caractère n'a été sollicité.

Explications

caractère est une variable chaîne qui reçoit le caractère éventuel.

Si le code ASCII du caractère retourné est inférieur à 32 (espace), l'utilisateur a sollicité une touche de fonction :

- ASC(caractère) = 1 : ENVOI ;
- ASC(caractère) = 2 : RETOUR ;
- ASC(caractère) = 3 : REPETITION ;
- ASC(caractère) = 4 : GUIDE ;
- ASC(caractère) = 5 : ANNULATION
- ASC(caractère) = 6 : SOMMAIRE ;
- ASC(caractère) = 7 : SUITE ;
- ASC(caractère) = 8 : CONNEXION ;
- ASC(caractère) = 9 : CORRECTION ;
- ASC(caractère) = 13 : RETOUR-CHARIOT.

Remarques

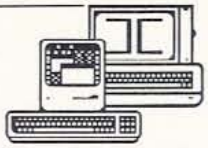
- sur le Macintosh, lorsque le tampon d'entrée contient plusieurs caractères, le caractère retourné par SAISIE est le premier caractère entré non lu (voir PURGE).
- les caractères dont les codes diffèrent sur le Minitel et l'ordinateur-serveur sont recodés.

Exemple

```
10 REM Exemple d'utilisation de SAISIE
20 SAISIE C%
```

Voir aussi ACCEPT, ACCEPTF, SECRET.

SANSLIGNE



Syntaxe de l'instruction
SANSLIGNE

But
Initialise le mode non ligné ou non souligné.

Explications
- si le Minitel utilisateur est en mode semi-graphique, SANSLIGNE passe l'affichage en mode non ligné (caractères semi-graphiques joints) ;
- si le Minitel utilisateur est en mode texte, SANSLIGNE passe l'affichage en mode non souligné ;

Remarque
Dans le cas du mode texte, SANSLIGNE envoie un attribut latent qui ne sera validé par le Minitel utilisateur qu'à la réception d'un espace (ASCII 32).

Exemple
```
10 REM Exemple d'utilisation
20 REM de SANSLIGNE
30 SANSLIGNE
```

Voir aussi PARAMG, PARAMT, LIGNE.

SECRET (ne fonctionne pas en local)

Syntaxe de l'instruction
SECRET *colonne,ligne,mode,sollicitation,longueur, retour,état*

But
Gère une zone de saisie contrôlée.

Explications
SECRET est en tous points identique à ACCEPT, mis à par le fait qu'au lieu de retourner les caractères frappés par l'utilisateur sur l'écran de son Minitel, on retourne uniquement des caractères '*' (astérisque).

Voir aussi ACCEPT, ACCEPTF, SAISIE.

SERVOFF (uniquement sur Apple //)

Syntaxe de l'instruction
SERVOFF

But
Libère l'interpréteur ampersand en restaurant l'ancien vecteur.

Exemple
```
10 REM Exemple d'utilisation de SERVOFF
20 SERVOFF
```

Voir aussi INIT.

TXT

Syntaxe de l'instruction
TXT

But
Passe en mode texte.

Exemple
```
10 REM Exemple d'utilisation de TXT
20 TXT
```

Voir aussi GR, PARAMG, PARAMT.

VIDECRAN

Syntaxe de l'instruction
VIDECRAN [*couleur*]

But
efface le contenu de l'écran en utilisant la couleur désignée.

Explications
couleur est une variable ou expression entière optionnelle qui désigne la couleur à utiliser pour effacer le contenu de l'écran :

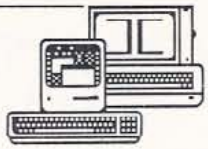
• 0 : noir	0% de gris
• 1 : rouge	50% de gris
• 2 : vert	70% de gris
• 3 : jaune	90% de gris
• 4 : bleu	40% de gris
• 5 : magenta	60% de gris
• 6 : cyan	80% de gris
• 7 : blanc	100% de gris

Remarque
Si *couleur* est omis, le noir est utilisé.

Exemples
```
10 REM Exemples d'utilisation
20 REM de VIDECRAN
30 VIDECRAN
40 VIDECRAN 2
50 VIDECRAN C%
60 VIDECRAN C%(I%)+1
```

Voir aussi VIDLIGNE, VIDRECT.

VIDLIGNE



Syntaxe de l'instruction
VIDLIGNE *ligne,couleur*

But
efface une ligne en utilisant la couleur désignée.

Explications
ligne est une variable ou expression entière indiquant la ligne à effacer. *ligne* doit être compris entre 1 et 24 inclus.
couleur est une variable ou expression entière qui désigne la couleur à utiliser pour effacer la ligne :

- 0 : noir 0% de gris
- 1 : rouge 50% de gris
- 2 : vert 70% de gris
- 3 : jaune 90% de gris
- 4 : bleu 40% de gris
- 5 : magenta 60% de gris
- 6 : cyan 80% de gris
- 7 : blanc 100% de gris

Exemples

```
10 REM Exemples d'utilisation
20 REM de VIDLIGNE
30 VIDLIGNE LI%,CO%
40 VIDLIGNE 4,3
50 VIDLIGNE 4,CO%(2)
60 VIDLIGNE L%+1,ASC(N$)-48
```

Voir aussi ALERTE, VIDECRAN, VIDRECT.

VIDRECT

Syntaxe de l'instruction
VIDRECT *x1,y1,x2,y2,couleur*

But
efface un rectangle en utilisant la couleur désignée.

Explication
x1, *y1*, *x2* et *y2* : coordonnées du rectangle à effacer. *x1* et *x2* doivent être compris entre 1 et 40 inclus. *y1* et *y2* doivent être compris entre 1 et 24 inclus.
couleur est une variable ou expression entière qui désigne la couleur à utiliser pour effacer le rectangle :

- 0 : noir 0% de gris
- 1 : rouge 50% de gris
- 2 : vert 70% de gris
- 3 : jaune 90% de gris
- 4 : bleu 40% de gris
- 5 : magenta 60% de gris
- 6 : cyan 80% de gris
- 7 : blanc 100% de gris

Remarque
x2 doit être supérieur ou égal à *x1*. *y2* doit être supérieur ou égal à *y1*.

Exemples

```
10 REM Exemples d'utilisation
20 REM de VIDRECT
30 VIDRECT X1%,Y1%,X2%,Y2%,CO%
40 VIDRECT 10,10,20,15,2
50 VIDRECT X%(X1%),3,X%(X2%),20,C%-1
```

Voir aussi VIDECRAN, VIDLIGNE.

Pom_Link 2.1 : les instructions

ACCEPT *colonne, ligne, mode, sollicitation, longueur, retour, état*
ACCEPTF *état*
AFFICHE *chaîne* [;] (sur l'Apple //)
AFFICHE *chaîne* (sur le Macintosh)
AFFICHER *chaîne* (uniquement Macintosh)
ALERTE *message, bip*
APPEL *état*
BAS *nombre*
BIP
CADRE *x1, y1, x2, y2, type, lignage, couleur C, couleur F*
CHARIOT [*nombre*]
CNXN
CNXR *état*
CURINVIS
CURVIS
DECNX
DEPLACE *colonne, ligne*
ENLIGNE *état*
FIXE
FLASH
GR
HAUT *nombre*
IDENT *valeur 1, valeur 2, valeur 3, état*
INIT *slot*
INVERSE
LIGNE
LOCALISE *colonne, ligne, état*
MODE *mode*
MTH *mode* (uniquement Apple //)
NORMAL
PARAMG *couleur C, couleur F, lignage, flash*
PARAMT *mode, couleur C, couleur F, flash, inversé, souligné*
PURGE (uniquement Macintosh)
REINIT
ROULEAU *condition*
SAISIE *caractère*
SANSLIGNE
SECRET *colonne, ligne, mode, sollicitation, longueur, retour, état*
SERVOFF (uniquement Apple //)
TXT
VIDECRAN [*couleur*]
VIDLIGNE *ligne, couleur*
VIDRECT *x1, y1, x2, y2, couleur*

Initiation

Jean-Jacques Colwhir

Le passage des paramètres

Écrire une routine autonome en assembleur et l'appeler depuis le Basic par un CALL ne présente que la difficulté de manipuler l'assembleur. Ainsi, pour faire un 'bip', il suffit de lire l'octet $C030 selon un cycle et une fréquence données. Depuis le Basic, on l'appellera par un *CALL adresse_de_début*, *CALL 768* par exemple. Si l'on veut depuis le Basic donner à la routine la longueur du bip et sa hauteur, le plus simple sera de mettre ces paramètres à des adresses convenues et depuis la routine, d'aller les rechercher aux mêmes adresses :

```
POKE 6,30 : POKE 7,150 : CALL 768
```

Dans la routine, on récupère les paramètres par :

```
        lda  $6
        sta  …
        …
        lda  $7
        jsr  …
        …
```

Passer les paramètres

L'objet de cet article est de montrer (de rappeler ?) comment passer des paramètres du Basic à la routine assembleur ; pour appeler notre bip par *CALL 768*, *LG*, *HT* ou *& BIP LG+3, 3 * (HT + K)* par exemple. Un prochain article présentera le passage aller et retour basé sur la réalisation d'un INPUT contrôlé.

Pour démonstration de cette première partie, nous montrerons un nouveau PRINT, capable d'envoyer une chaîne de caractères à un périphérique sans utiliser les fastidieux *PRINT D$"PR£"slot*. Il permettra également d'indiquer si les caractères devront être envoyés au périphérique avec le bit de poids fort à 1 ou à 0. Cette caractéristique est capitale pour l'émission de codes graphiques vers une imprimante puisque les huit bits doivent être significatifs (AppleSoft met systématiquement le bit de poids fort à 1). Cela servira également aux utilisateurs de LaserJet+ de Hewlett-Packard pour laquelle le bit 7 positionné à 1 pose un problème.
Ce PRINT ne fonctionnera que sous ProDOS en raison de la méthode de sortie de caractères adoptée mais le principe du passage des paramètres est valable également sous DOS 3.3.

Un nouveau PRINT

La syntaxe de notre instruction sera :

```
& PRINT chaîne, slot, poids_fort [;]
```

Chaîne sera une variable ou expression chaîne du type :

```
"Hello"
A$
A$(13) + "Hello"
STR$ (I*2) + LEFT$ (A$,3)
```

Slot sera une expression numérique représentant le numéro de port du périphérique destinataire, sous la forme :

```
1
SLOT
S(5)
```

Poids_fort sera également une expression numérique qui vaudra 255 ou 127 selon que les codes des caractères iront de 128 à 255 ou de 0 à 127 (pour les périphériques qui n'utilisent que 7 bits — Minitel par exemple — ce paramètre sera sans effet visible).

Voici des expressions valides :

```
& PRINT "Noe" + CHR$ (8) + "¨1",1,127
& PRINT A$(5),SLOT,PF
& PRINT STR$ (A = B),2,255
```

Pour les lignes qui suivent, nous prendrons l'exemple de la ligne Basic :

```
100 & PRINT "Essai",1,127 : IF A = …
      ↑
```

la flèche indique le caractère désigné par le pointeur de programme (TXTPTR en $B8, $B9).

L'ampersand

Première étape, l'installation du vecteur ampersand ('&' ou esperluette). Lorsque l'interpréteur Basic rencontre un &, il exécute un saut à l'adresse $3F5. Il nous suffit donc d'installer, à cette adresse, un vecteur (un saut) qui vise le début de notre interpréteur. Ainsi, notre routine prendra en charge l'analyse du programme Basic après lecture de l'ampersand.

```
        lda  £$4C       code de jump
        sta  $3F5
        lda  £<debut    adresse
        sta  $3F6
        lda  £>debut    adresse
        sta  $3F7
        rts             fin d'initialisation
debut   …
```

Lors du BRUN (démarrage du programme en langage-machine) de notre routine, seule l'installation du vecteur sera pratiquée ; on ne sautera à *debut* que lors de la rencontre d'un &. Notons qu'il s'agit là d'une méthode brutale : l'ancien vecteur est écrasé sans sauvegarde. La bonne méthode, pour que plusieurs routines ampersand puissent cohabiter consiste à noter le vecteur existant et à y faire un saut si la commande ne nous concernait pas.

Après rencontre d'un &, le pointeur de programme est disposé ainsi :

```
100 & PRINT "Essai",1,127 : IF A = …
      ↑
```

avec dans A le caractère qui suit l'&, en l'occurence, il doit s'agir du 'token' (code du mot-clé) de PRINT c'est-à-dire $BA (pour le trouver, il suffit de taper NEW : 10 PRINT et de passer en moniteur pour lire les octets à partir de $801).

```
        …
debut   cmp  £$BA
        beq  ok
        jmp  $DEC9      SNERR routine qui stoppe le
                        programme Basic avec le
                        message syntax error)
ok      jsr  $B1        CHRGET saisi le caractère
                        suivant et avance le pointeur
                        de programme d'un octet
```

Arrivés ici, nous nous sommes assurés qu'il s'agit bien de PRINT et le TXTPTR se présente ainsi :

```
100 & PRINT "Essai",1,127 : IF A = …
          ↑
```

Nous pourrions maintenant par une suite de JSR CHRGET saisir les caractères à afficher mais il est préférable de faire confiance à l'AppleSoft :

```
        …
        lda  £0         Réinitialise le calcul des
        sta  $52        chaînes temporaires
        jsr  $DD7B      FRMEVL, donne un Syntax
                        Error si besoin
```

FRMEVL est la routine d'évaluation des formules, ce qui permettra l'utilisation d'expression complexe du style STR$ (3 * AB) + "ERR" + CHR$(10). Elle évalue l'expression qui débute sous le TXTPTR et qui se termine par une virgule (ou une fin d'instruction ou de programme). Le résultat se trouve en FAC, zone de 6 octets à partir de $9D réservée aussi pour tous les calculs sur les nombres en virgule flottante. Si l'expression évaluée était une chaîne, et c'est le cas, seuls deux octets nous servent : $A0, $A1. Ils forment un pointeur sur le descripteur d'une chaîne temporaire que FRMEVL a créée à partir des constituants de l'expression.

Le descripteur, pointée par $A0, $A1 se présente sur l'Apple // ainsi :
octet 1 : longueur de la chaîne
octet 2 : poids faible de l'adresse du premier caractère
octet 3 : poids fort de l'adresse du premier caractère

Assurons-nous d'abord que l'expression évaluée par FRMEVL était bien une chaîne :

```
        …
        jsr  $DD6C      CHKSTR donne type mis-
                        match si ce n'est pas une
                        chaîne
```

Ceci fait, nous pouvons sauvegarder le pointeur sur le descripteur de la chaîne à afficher et nous intéresser aux paramètres suivants :

```
        …
        lda  $A0        FACMO, poids faible de
                        l'adresse
        sta  $85        FORPNT qui reçoit habituel-
                        lement le pointeur sur la
                        dernière variable utilisée
        lda  $A1        poids fort
        sta  $86
```

Notre TXTPTR est maintenant positionné ainsi :

```
100 & PRINT "Essai",1,127 : IF A = …
                  ↑
```

Pour évaluer le numéro de slot, nous utiliserons GETBYTC qui saute un caractère (en l'occurence la virgule) et qui évalue l'expression suivant ce caractère. Le résultat de cette évaluation doit tenir sur un octet (de 0 à 255) sinon nous aurons un Illegal Quantity Error. Le résultat est dans le registre X :

```
        …
        jsr  $E6F5      Getbytc
        stx  $6         on stocke en $6 le n° de port
```

Idem pour le sens du bit de poids fort :

```
        …
        jsr  $E6F5      Getbytc
        stx  $7         on stocke en $7 le sens du
                        poids fort
```

Nous avons tous les paramètres, vérifions si un ";" suit, le TXTPTR se présentant ainsi :

```
100 & PRINT "Essai",1,127 : IF A = …
                        ↑
```

Il faut lire le prochain caractère du Basic mais sans incrémenter le TXTPTR au cas où ce ne serait pas un point-virgule. C'est l'objet de CHRGOT, semblable à CHRGET mais sans incrément ; au sortir de CHRGOT, le caractère lu est dans A et le bit Z du registre d'état est positionné si on était en fin d'instruction.

```
        …
        lda  £0         initialise le drapeau de retour-
        sta  $8         chariot
        jsr  $B7        Chrgot
        beq  cr         si = 0, il n'y avait pas de ";"
        cmp  £$3B       est-ce ";" ?
        bne  $1
        jsr  $B1        saute le ";"
        jmp  pascr
$1      jmp  $DEC9      Syntax Error
cr      dec  $8         note qu'il faudra un retour-
                        chariot
pascr   …
```

Dans notre exemple, le TXTPTR n'a pas bougé puisqu'il n'y avait pas de point-virgule.

```
100 & PRINT "Essai",1,127 : IF A = …
                              ↑
```

Nous avons tous les éléments ; nous devons les contrôler :

```
        …
        lda  $6
        cmp  £8        pas de port au-dessus dc 7
        bcc  portok
err     jmp  rgerr     donne Range Error
portok  lda  $7        PF vaut 127 ou 255
        cmp  £$7F
        beq  pfok
        cmp  £$FF
        bne  err       sinon Range Error
pfok    …
```

Il faut maintenant rediriger la sortie de caractères vers le périphérique choisi. Le mieux, sous ProDOS, est de remplacer le *PR£slot* par une modification du vecteur de sortie de caractères dans VECTOUT en $BE30–$BE31 : sauvegarde l'ancien vecteur, remplacement par $C*n*00, *n* étant le numéro de port.

```
        …
pfok    lda  $BE30     sauve l'ancien vecteur
        sta  ancvec
        lda  $BE30+1
        sta  ancvec+1
        lda  £0        installe le nouveau
        sta  $BE30
        lda  $6        En $6, c'est le n° de port
        ora  £$C0      le résultat, c'est Cn
        sta  $BE31
```

Maintenant, affichons la chaîne en recopiant d'abord son descripteur en page 0 :

```
        …
        ldy  £2        copie le descripteur
$1      lda  ($85),y   en page 0
        sta  $18,y
        dey
        bpl  $1

$2      iny            y=0
        dec  $18       longueur chaîne
        bmi  fin
        lda  ($19),y   charge un caractère de la
                       chaîne
        ora  £$80
        and  $7        positionne poids fort
        jsr  $FDED     Cout, sortie de caractères
        jmp  $2
        …
```

Si demandé, on affiche un retour-chariot à la suite de la chaîne en veillant au bit de poids fort :

```
        …
fin     bit  $8        afficher un retour-chariot ?
        bpl  sortie    non si positif
        lda  £$8D      oui, on tient compte du poids
        and  $7        fort demandé
        jsr  $FDED     Cout
sortie …
```

Restituons enfin le vecteur de sortie de caractère (c'était l'écran 40 ou 80 colonnes ou peut-être un autre périphérique) :

```
        …
sortie lda  ancvec     restitue l'ancien vecteur
        sta  $BE30
        lda  ancvec+1
        sta  $BE31
        rts            et revient au Basic
```

Ajoutons la prise en charge d'un éventuel Range Error :

```
        …
rgerr   lda  £2
        jmp  $BE09
```

et le stockage de l'ancien vecteur :

```
ancvec ds   2
```

Source PRINT.S

Assembleur ProCODE

```
                    0           dsk  print
                    1           org  $300
                    2
0300: A9 4C         3           lda  £$4C
0302: 8D F5 03      4           sta  $3F5
0305: A9 10         5           lda  £<debut
0307: 8D F6 03      6           sta  $3F6
030A: A9 03         7           lda  £>debut
030C: 8D F7 03      8           sta  $3F7
030F: 60            9           rts
                   10
0310: C9 BA        11  debut    cmp  £$BA
0312: F0 03        12           beq  ok
0314: 4C C9 DE     13           jmp  $DEC9
0317: 20 B1 00     14  ok       jsr  $B1
                   15
031A: A9 00        16           lda  £0
031C: 85 52        17           sta  $52
031E: 20 7B DD     18           jsr  $DD7B
                   19
0321: 20 6C DD     20           jsr  $DD6C
                   21
0324: A5 A0        22           lda  $A0
0326: 85 85        23           sta  $85
0328: A5 A1        24           lda  $A1
032A: 85 86        25           sta  $86
                   26
032C: 20 F5 E6     27           jsr  $E6F5
032F: 86 06        28           stx  $6
                   29
0331: 20 F5 E6     30           jsr  $E6F5
0334: 86 07        31           stx  $7
                   32
0336: A9 00        33           lda  £0
0338: 85 08        34           sta  $8
033A: 20 B7 00     35           jsr  $B7
033D: F0 0D        36           beq  CR
033F: C9 3B        37           cmp  £$3B
0341: D0 06        38           bne  $1
0343: 20 B1 00     39           jsr  $B1
0346: 4C 4E 03     40           jmp  PASCR
                   41
0349: 4C C9 DE     42  $1       jmp  $DEC9
                   43
034C: C6 08        44  CR       dec  $8
                   45
```

```
034E: A5 06      46  PASCR   lda  $6
0350: C9 08      47           cmp  £8
0352: 90 03      48           bcc  portok
                 49
0354: 4C AC 03   50  err     jmp  rgerr
                 51
0357: A5 07      52  portok  lda  $7
0359: C9 7F      53           cmp  £$7F
035B: F0 04      54           beq  pfok
035D: C9 FF      55           cmp  £$FF
035F: D0 F3      56           bne  err
0361: AD 30 BE   57  pfok    lda  $BE30
0364: 8D B1 03   58           sta  ancvec
0367: AD 31 BE   59           lda  $BE30+1
036A: 8D B2 03   60           sta  ancvec+1
036D: A9 00      61           lda  £0
036F: 8D 30 BE   62           sta  $BE30
0372: A5 06      63           lda  $6
0374: 09 C0      64           ora  £$C0
0376: 8D 31 BE   65           sta  $BE31
                 66
0379: A0 02      67           ldy  £2
037B: B1 85      68  $1       lda  ($85),y
037D: 99 18 00   69           sta  $18,y
0380: 88         70           dey
0381: 10 F8      71           bpl  $1
                 72
0383: C8         73  $2       iny
0384: C6 18      74           dec  $18
0386: 30 0C      75           bmi  fin
0388: B1 19      76           lda  ($19),y
038A: 09 80      77           ora  £$80
038C: 25 07      78           and  $7
038E: 20 ED FD   79           jsr  $FDED
0391: 4C 83 03   80           jmp  $2
                 81
0394: 24 08      82  fin      bit  $8
0396: 10 07      83           bpl  sortie
0398: A9 8D      84           lda  £$8D
039A: 25 07      85           and  $7
039C: 20 ED FD   86           jsr  $FDED
                 87
039F: AD B1 03   88  sortie   lda  ancvec
03A2: 8D 30 BE   89           sta  $BE30
03A5: AD B2 03   90           lda  ancvec+1
03A8: 8D 31 BE   91           sta  $BE31
03AB: 60         92           rts
                 93
03AC: A9 02      94  rgerr    lda  £2
03AE: 4C 09 BE   95           jmp  $BE09
                 96
                 97  ancvec   ds   2
```


Pom's distribue SuperMacroWorks

Après l'article de Damien Nould dans Pom's 33, et l'éloge qu'en faisait aussi J.Y. Bourdin, vous étiez nombreux à rechercher vainement SuperMacroWorks chez les revendeurs.
Pom's a obtenu de Beagle Bros la possibilité de diffuser ce programme de Randy Brandt dans ses versions **française** et américaine.

Précisons que, si SuperMacroWorks est bien entendu compatible avec toute carte d'extension mémoire pour Apple // (cartes Apple, Applied Engineering, Checkmate et autres), il ne nécessite aucun équipement spécial : du moment que vous pouvez charger AppleWorks 1.4 tel qu'il est, vous pouvez utiliser SuperMacroWorks.

Pour les techniciens : SuperMacroWorks se loge dans la carte langage de la mémoire auxiliaire, ce qui n'est pas simple, sachant que vous n'avez plus alors de ROM et que vous devez utiliser la pile et la page zéro de la mémoire auxiliaire. C'est d'ailleurs pourquoi AppleWorks renonce à utiliser cet espace. Les seules exceptions imaginables à la compatibilité avec SuperMacroWorks seraient donc pour le propriétaire d'un][+ qui aurait modifié AppleWorks avec Plus-Works 2.0 ou le 'patch' fourni avec la carte Ramfactor d'Applied Engineering (ce qui est malin), ou l'improbable propriétaire d'un //e qui l'aurait laissé à 64Ko avec une carte 80 colonnes non étendue et aurait fait l'une des deux modifications précédentes pour charger quand même AppleWorks.

Attention : la version française que nous diffusons est la version 1.2. Cette version étant débuguée, il ne faut plus faire le patch de Pom's 33 page 67. Si vous avez une version antérieure, contactez votre revendeur. Si vous n'avez que la version US de SuperMacroworks, adressez-vous à Dimitri Geystor (Pom's 33 page 63).

Et si vous n'avez pas encore SuperMacroWorks...

Prix : 500,00 F TTC – abonnés : 450,00 F TTC – Port 20,00 F

A l'essai : les TimeOut

Eric Weyland

Comme c'est la lecture de l'article de J. Bourdin dans Pom's 33 qui m'a fait passer commande au père Noël de quelques programmes de la série des TimeOut de Beagle Bros je n'ai pas pu refuser quand il m'a demandé d'expliquer ici pourquoi je suis bien content d'avoir pensé à mettre mes sabots dans la cheminée.

Le principe : continuer AppleWorks

La série des *TimeOut* ajoute une quantité de fonctions nouvelles à *AppleWorks*, à la fois parfaitement indépendantes les unes des autres et totalement intégrées aussi bien à *AppleWorks* qu'entre elles : chaque disque *TimeOut* fournit en même temps une modification d'*AppleWorks* qui est le noyau intégrateur commun, véritable exploit de programmation signé *Alan Bird*, et une nouvelle "application" qui se charge en prenant une partie du bureau d'*AppleWorks*, fonctionnant en 'overlay' (un segment de programme chargé momentanément du disque) d'*AppleWorks*. Nul besoin de sauver puis recharger l'état présent de la mémoire *AppleWorks*, comme avec *Pinpoint*.

C'est là évidemment une ouverture fantastique pour *AppleWorks* : loin de fournir comme *Pinpoint* des accessoires supplémentaires dans une application, ou des applications extérieures à la première, *TimeOut* transforme *AppleWorks* en une sorte de super-système d'exploitation pilotant des applications intégrées. On peut parfaitement imaginer par exemple un assembleur *TimeOut* assemblant sous *AppleWorks* des sources constituées de fichiers traitement de texte. Disons que *TimeOut* réussit avec *AppleWorks*, sans menus déroulants ni fenêtres, mais avec une transparence parfaite, ce qui a été raté (à mon humble avis) sur *IBM* avec *Windows* : un intégrateur d'applications. Ou bien disons que, de même qu'il existe un "atelier du programmeur" (*Apple Programmer WorkShop*) sur GS, il existe maintenant un "atelier de l'utilisateur" (Apple... works !) sur //c, //e et IIGS : on peut tout faire sans sortir de son atelier.

Telle était la philosophie de *Bob Lissner*, l'auteur d'*AppleWorks* : il faut bien sûr intégrer l'interface utilisateur, mais ce ne sont pas les menus déroulants, souris, fenêtres, etc., qui font l'intégration. Il s'agit là seulement de l'expression visible d'une autre intégration, celle des opérations que fait l'utilisateur et des données qu'il manipule. Tous les "couper-coller", les accessoires de bureau, etc., ne sont qu'un pâle substitut, un pauvre "faute de mieux", de l'intégration vraie. C'est parce que *Beagle* est totalement fidèle à cette inspiration initiale d'*AppleWorks* que *TimeOut* est une telle réussite.

Une conséquence désagréable de cette intégration est que la série des *TimeOut* ne marche et ne marchera sans doute jamais que sur la version américaine 2.0 d'*AppleWorks*, et pas sur la version française 1.4. Non que cela soit techniquement impossible : mais il faudrait alors adapter et traduire chacune des applications *TimeOut* présentes et futures, ce qui représente énormément de travail. L'investissement est sans doute trop grand pour être rentable (à moins qu'un taux spectaculaire de ventes de *SuperMacroWorks* en France n'amène *Beagle* à réfléchir, qui sait ?).

Second point à savoir : il vous faut de la place sur le bureau (de la mémoire). Bien sûr, vous pouvez choisir de charger en mémoire certaines applications seulement au démarrage d'*AppleWorks*, et *TimeOut* vous permet d'appeler ensuite, sous *AppleWorks*, en cas de besoin, une autre application depuis le disque (y compris le disque Ram type *Speedisk*) ; il vous permet également de vider la mémoire et le bureau des applications dont vous ne vous servez plus, aussi facilement qu'on efface un fichier du bureau. Mais une application comme *TimeOut Graph*, chargée de faire des graphiques avec les fichiers tableur, prend à elle seule 56Ko sur le bureau d'*AppleWorks* : avec une machine de 128Ko, l'Apple // courant, vous ne pourrez pas charger, en même temps que l'application, le fichier tableur qu'elle doit traiter !

Il faut donc soit avoir suffisamment de mémoire pour ne pas être gêné, soit renoncer aux *TimeOut* et quitter *AppleWorks* pour utiliser des programmes indépendants de lui (en l'occurrence *Visualizer* de *PBI* ou *Graphic Edge* de *Pinpoint*). Inutile donc de préciser que les *TimeOut* sont compatibles avec pratiquement toutes les sortes de carte mémoire qui permettent d'étendre le bureau d'*AppleWorks* : il s'agit ici d'une nécessité.

Troisième point à savoir : il vous faut de la place sur disque pour placer vos fichiers *TimeOut*, puisqu'en fait c'est

AppleWorks qui enfle. Si vous avez un lecteur 3,5' ou un disque dur, pas de problème (les *TimeOut* sont d'ailleurs livrés à la fois sur disquette 5,25' et sur disquette 3,5'). Si vous avez un Méga de Ram, vous pourrez vous débrouiller. Mais si vous n'avez que 128Ko et deux lecteurs 5,25', il va falloir jongler : au début vous pourrez placer vos applications *TimeOut* sur votre disque de données. Mais elles risquent vite de remplir la disquette. Bref, avec les *TimeOut*, c'est non seulement *AppleWorks*, mais l'Apple lui-même qui grandit. Les deux faces d'une disquette 5,25' ne lui suffisent déjà plus pour stocker toutes les polices de *TimeOut SuperFonts*. Il faut donc sérieusement songer aux extensions nécessaires (mémoire, lecteurs de disques).

L'indispensable : TimeOut SuperFonts

TimeOut Superfonts de *Mark Simonsen* est une vraie merveille. Il imprime les fichiers textes *AppleWorks* entièrement en caractères graphiques proportionnels, vous donne le choix entre quatre qualités d'impression possibles, dans les "styles" (italiques, inverse, gras, souligné, ombré, silhouette, indices, exposants) du Mac tous combinables sur la même ligne, avec les dessins (HGR ou DHGR mais pas SHGR) ou les parties de dessins que vous voulez, avec les polices que vous voulez (jusqu'à 64 dans le même texte : la disquette en fournit 47, et d'autres disquettes de polices sont promises à bon marché).

Le //c ou //c est transformé en Macintosh ou en GS monochrome : d'ailleurs il utilise les polices du Mac, qui sont aussi les polices système du GS. Seule différence avec le Mac : vous travaillez votre texte sur l'écran texte d'*AppleWorks*, et il faut choisir l'option "imprimer sur écran" pour voir à l'avance ce que donnera votre impression graphique. Il reconnaît toutes les options d'impressions du traitement de texte *AppleWorks*, auxquelles il ajoute les siennes (enfin la vraie justification à droite avec *AppleWorks* !). Il nous donne accès à tous les caractères des polices Mac, et utilise des polices de 8 à... 127 points !

Voilà un sérieux concurrent pour *MultiScribe*, *Printrix* et... *GSWrite* (il imprime plus vite que *GSWrite* – ce qui n'est guère difficile – tout en utilisant ses polices, et il reconnaît plus de 40 imprimantes). Seul problème : pour le moment, il ne dispose pas d'éditeur de polices (à ce propos : quel éditeur de polices système utilisez-vous sur GS ?). Pour nos accents, il faut donc par exemple remplacer chaque "é" par "<x2>N<x1>" avant l'impression avec les polices fournies sur la disquette. Voilà une belle macro *SuperMacroWorks* à faire... Mais *Mark Simonsen* nous promet pour bientôt un *Universal Font Editor*, toujours chez *Beagle*, qui permettra d'éditer les polices Mac ou GS Write, y compris sur //e.

Notons que *TimeOut SuperFonts* ne périme pas *GraphMerge* de *Pinpoint*, qui imprime aussi les images dans les fichiers *AppleWorks*, mais imprime le texte en mode texte au lieu du mode graphique de *SuperFonts*. *GraphMerge* s'accommode donc d'une police transférée dans l'imprimante, ce que ne peut faire *SuperFonts*.

Les utiles : FileMaster, Graph, SideSpread, QuickSpell

TimeOut File Master est une sorte de Filer évolué ou de *Copy][Plus* intégré à *AppleWorks*, mais sans fonctions DOS 3.3. Sa fonction de backup sélectif des fichiers qui ne sont pas déjà en double le rend particulièrement utile aux utilisateurs de disque dur. Et la possibilité de programmer des macros pour lui avec *SuperMacroWorks* me semble fort intéressante...

TimeOut Graph est un bon grapheur en DHGR, qui fait vite et bien ce qu'on attend de lui, et reconnaît l'essentiel des imprimantes. Il n'est pas en couleurs, mais *AppleWorks* non plus. J'ai particulièrement apprécié sa capacité à retracer automatiquement les graphiques dès que vous changez une valeur dans votre fichier tableur. Comme ce fichier est en mémoire en même temps que *TimeOut Graph*, on peut percevoir immédiatement ce qu'est l'intégration vraie. En fait, *AppleWorks* est tout simplement doté désormais de la fonction graphique intégrée qui lui manquait.

TimeOut SideSpread est de la même veine : il intègre tranquillement à *AppleWorks* la fonction d'impression verticale des fichiers tableurs trop larges qui demandait auparavant de sortir d'*AppleWorks* pour utiliser *FontWorks* ou *SideWays*, et reconnaît également l'essentiel des imprimantes.

TimeOut Quickspell est plus rapide que *Pinpoint Spelling Checker* pour corriger vos fautes en anglais, et fait aussi bien que *Pinpoint Document Checker*, (même en laissant son dictionnaire de 80 000 mots sur disque : si vous mettez celui-ci sur la carte *Speedisk*, c'est quasiment instantané), le tout sans avoir besoin de quitter *AppleWorks*.

Le superflu : TimeOut DeskTools

TimeOut DeskTools me semble un produit assez inutile, engendré uniquement par la nécessité purement commerciale de "couvrir tout le terrain" de ce qui se vend déjà bien sur Apple // (les accessoires *Pinpoint* en l'occurrence). Les fonctions de ces accessoires de bureau sont tout aussi bien remplies soit par *AppleWorks* lui-même (la calculette = le tableur ; le bloc-notes = le traitement de textes), soit par des macros *SuperMacroWorks* (calendrier, adresseur d'enveloppes, composeur téléphonique, horloge, conversion majuscules/minuscules, décompte des mots d'un texte, et même l'encryptage des fichiers).

Beagle ne nous avait pas habitués à ce genre de frivolités. Quoi qu'il en soit, l'abonnement à Pom's, à ses disques, et à ses programmes de macros pour *SuperMacroWorks* est un bien meilleur investissement que les *TimeOut DeskTools*.

— Mais – me direz-vous – les publicités de *DeskTools* parlent d'un utilitaire de conversion des données qui permet de passer directement par le presse-papier, sans l'intermédiaire d'un fichier DIF sur disque, des catégories de la base de données dans des colonnes du tableur, et vice versa, et c'est quand même plus pratique qu'une macro qui passe par un disque, même un disque Ram comme celui de la carte *Speedisk*, n'est-ce pas ?

— Bien sûr, mais ce que les publicités ne disent pas, c'est que cet accessoire est également fourni, en prime, avec *TimeOut Graph* et *TimeOut UltraMacros*...

PinPoint, TimeOut, Macros : compatibilités, incompatibilités

Une excellente nouvelle : tous les *TimeOut* (sauf *UltraMacros*) sont entièrement compatibles avec *Pinpoint*. À l'installation, vous 'patchez' d'abord *AppleWorks* avec *TimeOut*, puis vous y installez *Pinpoint*. Une fois *AppleWorks* chargé, il suffit de faire 🍎-P pour accéder à *Pinpoint*, et ⌫-Escape pour accéder à *TimeOut*.

Du coup (et c'est sans doute pourquoi *Beagle* ne crie pas sur les toits cette bonne nouvelle), il me semble que les *TimeOut* qui doublent les accessoires de *Pinpoint* (Correcteur, accessoires de bureau par exemple) ne sont pas indispensables à qui a déjà *Pinpoint*. D'autant que *Pinpoint*, lui, s'installe aussi dans Basic System, *Word Perfect* et autres, et pas seulement dans *AppleWorks*, et permet avec *Pinpoint ToolKit* de programmer soi-même ses propres accessoires. De plus *Pinpoint* reconnaît aussi *AppleWorks* 1.4 français (même si certains des accessoires tels le *Speller* ou les macros *KeyPlayer* ne marchent qu'avec *AppleWorks* américain).

Deuxième bonne nouvelle : les *TimeOut* sont compatibles avec *SuperMacroWorks* et *AutoWorks*. Mais... ces deux programmes de macros, ainsi d'ailleurs qu'*UltraMacros*, ne sont pas eux-mêmes compatibles avec *Pinpoint* (sauf *AutoWorks* qui offre une option de choix du banc mémoire aux propriétaires d'une carte d'extension mémoire type *RamWorks* ou *MultiRam* sur //e ou //c). Les gourmands qui voudront avoir tout en même temps (Macros ET accessoires *Pinpoint* ET applications *TimeOut*), devront donc utiliser *KeyPlayer*, le programme de macros de *Pinpoint*. Mais... *KeyPlayer* n'automatise que les fonctions *AppleWorks*, pas celles de *Pinpoint* ni celles de *TimeOut*, et ne marche pas avec *AppleWorks* français.

Bien entendu vous êtes complètement perdu dans le dédale des informations qui précèdent : cela prouve que vous êtes normalement constitué, j'ai du mal à m'y retrouver moi-même. Ne nous plaignons pas trop de notre difficulté à comptabiliser notre fortune, et simplifions.

Si vous utilisez *AppleWorks 1.4* français, vous ne pouvez rajouter à *AppleWorks* qu'un seul programme de macros, *SuperMacroWorks*, et un seul programme d'extensions, *Pinpoint*. Vous ne pourrez pas les utiliser en même temps, il faudra quitter l'un pour utiliser l'autre. Et vous n'aurez accès qu'à certains accessoires de Pinpoint.

Si vous utilisez *AppleWorks 2.0* américain, le tableau joint à cet article devrait vous éclairer un peu plus.

UltraMacroWorks ou... SuperMacroWorks ?

TimeOut UltraMacros, le dernier-né de *Randy Brandt*, est bel et bien encore plus doué que son aîné *SuperMacroWorks*. Mais il partage avec lui le tempérament fier et indépendant de son auteur. Car, quoi qu'en dise *Beagle*, ce n'est pas vraiment une application *TimeOut* comme les autres. Il ne se loge pas dans le bureau d'*AppleWorks*, mais dans la place laissée libre par celui-ci (la carte langage de la mémoire auxiliaire : c'est d'ailleurs pourquoi il est le seul *TimeOut* qu'on puisse recommander au même titre que *SuperMacroWorks* à ceux qui n'ont que 128Ko de mémoire). Il n'est pas un subalterne docile qui reste à sa place tant qu'on ne l'appelle pas, comme les autres *TimeOut*. Bien entendu il est compatible avec les autres *TimeOut* : mais c'est pour les commander. Il m'a fallu fouiller, mais j'ai pu découvrir page 26 de la documentation une de ses fonctions les plus intéressantes : il automatise également les fonctions des applications *TimeOut*.

UltraMacros est une sorte de langage de programmation complet pour *AppleWorks* et *TimeOut* qui intègre toutes les commandes de l'*Applesoft*, y compris les manipulations de chaînes, et en rajoute même (IF THEN ELSE). Il permet d'introduire ses propres routines machines dans *AppleWorks* avec les macros Peek, Poke et Call, de créer des macros avec 🍎 et ⌫, des "macros-réveil" qui se déclenchent à l'heure choisie, des "fichiers-tâches" qu'on lance depuis le sélecteur du ProDOS, il éteint provisoirement l'écran quand vous allez prendre un café, il corrige le bug du Control-à dans la définition des codes imprimantes.

Mais... il ne reconnaît pas la version française 1.4 d'*AppleWorks*, ni les macros déjà existantes de *SuperMacroWorks*, françaises ou non. Les macros que vous avez et aurez pour la version française de *SuperMacroWorks* ne tourneront pas sur *UltraMacros*. J'ai essayé de traduire les *macros agenda* de *Dimitri Geystor* : c'est un joli casse-tête (il faut traduire d'abord les commandes *AppleWorks 1.4 VF* en commandes *AppleWorks 2.0 US*, traduire ensuite les commandes *SuperMacroWorks VF* en commandes *SuperMacroWorks US*, traduire enfin les commandes *SuperMacroWorks* en commandes *UltraMacros* !).

SuperMacroWorks vous permet de transférer une police *Power Print* pour l'imprimante au chargement d'*AppleWorks*, ce qu'*UltraMacros* ne permet plus. Enfin – et surtout – un fait que *Beagle* s'abstient soigneusement de dire, et que doivent donc crier sur les toits les utilisateurs qui n'aiment pas ce genre de cachotteries : *SuperMacroWorks* aussi automatise les applications *TimeOut* ! Eh oui : quand la légion des *TimeOut* s'est formée, son commandant était déjà là !

UltraMacros me semble donc surtout destiné aux fanatiques de la programmation et du "bidouillage" qui voudraient un autre programme, en plus de *SuperMacroWorks*, pour créer des applications spéciales en anglais seulement.

Si vous tenez absolument à UltraMacros, il vous suffira d'envoyer 22,5 dollars à Beagle en même temps que votre disque original (version US) de SuperMacroWorks.

On est prié de ranger soi-même son bureau

En admettant que vous ayez un bureau (une mémoire) de taille suffisante, vous avez tellement d'accessoires et d'applications à y mettre qu'un sérieux problème de rangement se pose. La Ram supplémentaire doit à la fois vous servir à étendre le bureau d'*AppleWorks*, à sauver les fichiers transitoires (fichiers ASCII, fichiers DIF) utilisés pour communiquer entre deux fonctions d'*AppleWorks*, à entreposer provisoirement la mémoire de travail *AppleWorks* que *Pinpoint* y sauve de temps en temps, à stocker les applications *TimeOut* et les accessoires *Pinpoint*, et même à garder en permanence sous la main ProDOS, un sélecteur de programmes et vos applications favorites (dont *AppleWorks* lui-même bien sûr). 1 024Ko, c'est vraiment le minimum, mais il reste le rangement... Aïe !

En fait, le problème est plus simple qu'il n'y paraît : il se résume au "partitionnement", ce barbarisme désignant le partage de la mémoire entre *AppleWorks* et son bureau d'un côté, le disque Ram de l'autre. Sur le GS, pas de problème : le tableau de bord est là pour cela. Si vous avez une carte mémoire type *MultiRam* de *Checkmate*, *RamWorks*, *Z-Ram* ou *RamFactor* d'*Applied Engineering*, les logiciels d'installation du disque Ram et de modification d'*AppleWorks* permettent ce "partitionnement". Si vous avez le nouveau //c un Méga ou une carte type Apple dans votre //e, *AppleWorks* dévore toute la place disponible sur la carte pour son bureau, et... bigre, on ne peut plus rien ranger du tout !

La solution consiste à 'patcher' *AppleWorks* de façon à limiter son appétit de mémoire, avant toute modification par *SuperMacroWorks*, *TimeOut* et/ou *Pinpoint*. Deux programmes du domaine public se chargent de ce travail pour nous. L'un, publié par *Steve Stephenson* dans *Open Apple volume 3*, modifie *AppleWorks 2.0* américain. L'autre, publié par l'*Écho des Apple* d'Octobre 87, est l'adaptation du premier à *AppleWorks 1.4* français par *J.Y. Bourdin*. Les colonnes de Pom's n'étant pas destinées à être remplies de listings de programmes du domaine public, nous ne pouvons les reproduire ici. Mais nous les avons ajoutés gracieusement, en prime, sous les noms de *APLWorkS2.0.MOD* et *APLWorkS1.4.MOD*, sur la face ProDOS de la disquette Pom's 34 ... Bien entendu, ne les utilisez que sur une copie de votre disquette AppleWorks !

Conclusion : innovation ou conformisme ?

Vous comprenez maintenant pourquoi je suis si content de ce que j'ai trouvé dans ma cheminée : les *TimeOut* constituent dans leur conception fondamentale une innovation importante et extrêmement prometteuse pour tous les utilisateurs d'Apple II et d'*AppleWorks*. Cette innovation se situant dans la continuité de l'esprit même d'*AppleWorks*.

↙ compatible avec ←	SuperMacroWorks	UltraMacros	AutoWorks	KeyPlayer
AppleWorks 2.0 seul	OUI	OUI	OUI	NON
AppleWorks 2.0 avec Pinpoint	NON	NON	OUI/NON	OUI
AppleWorks 2.0 avec TimeOut	OUI	OUI	OUI	NON
AppleWorks 2.0 avec Pinpoint ET TimeOut	NON	NON	OUI/NON	OUI
Pilote AppleWorks ET les fonctions TimeOut	OUI	OUI	???	NON

La réussite commerciale me semble promise à cette série, et il faut que nous assurions cette réussite : j'incite fortement tous ceux que l'anglais ne rebute pas trop à s'offrir au moins *TimeOut SuperFonts* en plus de *SuperMacroWorks*.

Sachons-le, le nom de *Beagle Bros* ne signifie plus la même chose qu'avant : ce qui était un atelier d'artisans décontractés et sympathiques devient maintenant une entreprise moderne et performante, rivalisant avec les "pros" du logiciel, une puissance commerciale qui s'impose par l'excellence de ses produits. *Beagle* prend la liste des principaux succès de vente en matière de compléments à *AppleWorks* (*Copy][Plus*, *Pinpoint accessories*, *MultiScribe*, *Visualizer*, *SideWays*, *SuperMacroWorks*, *Pinpoint Spelling Checker*) et y substitue sa version, intégrée à *AppleWorks* et souvent meilleure que les autres. Sincèrement, non seulement je n'échangerais pas un GS avec *AppleWorks*, *SuperMacroWorks* et *TimeOut* contre 2 ou 3 barils d'ordinateurs-X avec *Windows*, mais je ne l'échangerais même pas contre un Mac avec *Microsoft Works* !

Cette stratégie de *Beagle* est certainement la bonne : une telle série de bons utilitaires réellement intégrés est effectivement un besoin des utilisateurs. Certaines de ces applications (au moins *SuperFonts*) sont tellement au-dessus du lot de ce qui existait déjà qu'elles sont en fait des innovations. De plus, cela permet d'assurer un bon départ, une bonne assise pour la série, ce qui permettra des développements nouveaux et peut-être plus risqués. Enfin cela utilise le succès mérité de programmes *Beagle* ou *SoftWare Touch* déjà existants (*AutoWorks*, *FontWorks*, *Triple Dump*) pour faire encore mieux.

Mais reconnaissons qu'il y a un risque de glissade vers ce conformisme, ce renoncement à la création et à l'innovation vraie, qu'engendrent les situations de concurrence effrénée dans un marché fermé (allumez votre téléviseur sur n'importe quelle chaîne pour en voir un exemple). Certaines applications *TimeOut* ne font que "doublonner" purement et simplement de bons programmes déjà existants. Elles travaillent souvent mieux et plus vite, mais elles font la même chose : elles n'innovent pas réellement. L'opinion de *J.Y. Bourdin* qui se réjouissait dans Pom's 33 de la fusion *Beagle/SoftWare Touch* me semble globalement confirmée par les *TimeOut* : mais ses craintes sur l'orientation exclusivement *utilisateur* imprimée par la nouvelle direction ne sont hélas pas non plus sans fondements.

J'en prends pour preuve un signe qui, bien que superficiel, me paraît néanmoins quelque peu inquiétant : fini de rire en lisant les documentations *Beagle* ; plus de gravures rétros et de légendes humoristiques, plus de complicité entre auteur et lecteur, plus de clin d'œil ; et plus non plus de cette pédagogie souriante qui enseignait sans qu'il s'en aperçoive au lecteur à maîtriser sa machine, et peut-être, qui sait, à devenir un jour un auteur *Beagle*. Nous sommes au royaume des "pros" : tenue stricte, documentations "clean", compartiments étanches ; utilisateurs, vous êtes priés de parler devant l'hygiaphone.

Gardons cependant confiance : les meilleurs programmeurs *Beagle* (dont *Randy*), et une majorité d'utilisateurs ont toujours "l'esprit Wozniak". *Randy Brandt*, qui n'a pas perdu l'ancienne mentalité, parsème ses programmes d'allusions, de clins d'œil et d'informations presque cachées. (Un conseil : lisez, listez, testez tous les fichiers des disquettes *TimeOut* !). *Randy* a même fondé une mini-entreprise, appelée *Jem Software*, qui diffuse pour un prix dérisoire des mini-programmes de l'ancien style *Beagle*. Si Hermès le Dieu messager y consent, je pourrai peut-être vous entretenir un jour d'un certain "Patchmania"...

Des cerveaux comme ceux d'*Alan Bird*, *Randy Brandt* ou *Mark Simonsen* sont strictement incapables de se limiter à améliorer le programme du concurrent. Ce sont par essence des créateurs : ils ont déjà su, avec *TimeOut*, inventer les chemins de l'avenir. Préparons-nous à d'autres belles surprises...

Beagle Bros/Père Noël
6215 Ferris Square, Suite 100,
San Diego, CA 92121, USA.

Jem Software/Randy Brandt
P.O. Box 20920, El Cajon,
CA 92021, USA.

PBI Software, Inc
1163 Triton Drive, Foster City,
CA 94404, USA.

Pinpoint Publishing
5901 Christie Avenue, Emeryville,
CA 94608, USA.

POM_LINK 3.0 : PLUS LOIN...

La version 3.0 de Pom_Link va plus loin dans l'exploitation des ressources Minitel. Elle ouvre la porte à la réalisation de serveurs très puissants.
Parmi les instructions nouvelles :

* gestion du mode téléinformatique ;
* mise en mode téléinformatique, mise en mode Vidéotex ;
* routine de temporisation pour allouer des délais ;
* affichage des caractères émis et reçus (debugging) ;
* affichage de la date et de l'heure ;
* instruction accept sans effacement des points ;
* instruction accept numérique contrôlée ;
* instruction spécialisées pour le changement de la couleur des caractères ou de la couleur de fond seule ;
* idem mais exprimé en pourcentage de niveau de gris ;
* réglage du délai imparti à la connexion ;
* calcul automatique de la durée de communication (en minutes entières sur l'Apple //, en secondes sur la Macintosh ;
* fonctions utilitaires (sur l'Apple // : bug de l'ONERR... sur le Macintosh : changement de curseur, chargement des icônes système...)

Répon'deur...

Apple //

...enregistreur, télématique, interrogeable...

Christian Piard

Le répondeur télématique que nous vous proposons ici n'est qu'une illustration des multiples possibilités des routines Pom_Link 2.1. Il n'en est pas moins complet, fiable et convivial. Il se distingue de 90 % des serveurs-kiosque par la rapidité et le contrôle des saisies au clavier : on ne risque pas de saisir 8 caractères là où on en attend 3 ; ainsi, l'aspect de l'écran est protégé. L'utilisateur reste toujours dans les zones qui lui sont allouées.

Fonctions du répondeur

Le principe de l'ensemble est le suivant :

Un client, un ami vous appelle en votre absence : l'apple // détecte cet appel, connecte votre Minitel. Votre client et néanmoins ami reçoit alors la tonalité familière aux habitués du Minitel et connecte son propre Minitel (avec la touche Connexion/Fin). Le dialogue peut s'engager entre votre ordinateur-serveur et l'ami et/ou relation d'affaires.

Il aura la possibilité de laisser un ou plusieurs messages qui seront enregistrés sur disque. En fin de communication, ordinateur-serveur et Minitel se remettent en veille dans l'attente du prochain appel.

Tout ceci se passe alors que vous êtes fort loin de votre domicile/bureau/atelier ? Appelez chez vous avec un Minitel sous la main : vous pourrez demander la lecture, depuis votre lieu de vacances par exemple, des messages télématiques qui ont été laissés sur l'ordinateur. Cet accès à vos messages est évidemment protégé par un mot de passe.Beaucoup de messages sont arrivés, vous craignez la saturation du disque ? Vous avez également la possibilité d'effacer le fichier pour libérer de la place. Luxe supplémentaire, vous changerez si nécessaire le mot de passe (cas où vous l'auriez communiqué à un ami – client ? – pour démonstration).

Hard : comment faire ?

L'ordinateur. Ce programme fonctionne avec :

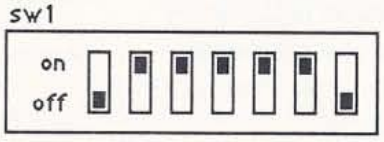
- Apple][+, Rom minuscules, carte langage, carte Super Série Apple ou ;
- Apple //e, carte Super Série Apple ou ;
- Apple //c ou ;
- Apple IIgs (sur port série intégré ou carte Super Série Apple).

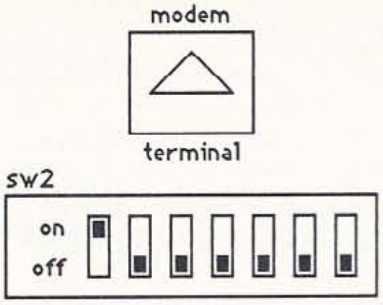
Le Minitel. Il sera nécessairement bi-standard (1B) c'est-à-dire avec modem *retournable* (aujourd'hui, c'est presque toujours le cas).

Le détecteur de sonnerie. Peu importe son type : celui dont le schéma a paru dans le numéro 33 de Pom's convient, celui vendu par votre revue préférée également. Notez toutefois que la fiabilité de l'ensemble est directement fonction de la qualité des composants utilisés.

Le câble Minitel/Apple. Il est maintenant bien connu. Vous l'utilisez déjà avec MinBas (Pom's 27), InterPom's (28), T.Pom's (30), Clv_Pom's, HGR/Minitel (33). S'il vous manque, Pom's en dispose.

Le port ou la carte série de l'Apple. Sur les Apple][+ ou //e, la carte série sera configurée ainsi :





Sur votre IIGS, à l'aide du tableau de bord, mettre toutes les options port modem par défaut sauf :

```
Data/Stop Bits : 7/1
Parity : Even
DCD Handshake : No
DSR/DTR Handshake : No
```

Soft : comment faire ?

Ce paragraphe ne vous concerne que si vous ne disposez pas de la disquette d'accompagnement de ce numéro. Il vous faut saisir et sauvegarder sur une disquette ProDOS le programme Basic *Repomdeur*. En ce qui concerne, Pom.link.2.1, même pensum ; il est toutefois plus fiable de reprendre le source et de l'assembler car les erreurs sont en principe décelées plus facilement.

Sur votre disquette, n'oubliez pas de mettre ProDOS 8 (version 1.1.1 ou plus) et Basic.System.

Avant de lancer l'ensemble, modifier la ligne Basic 26 pour indiquer :

- votre mot de passe à la place de ABCDE (de 4 à 7 caractères ;
- votre numéro de téléphone à la place de (1) 12 34 56 78 (maximum 15 caractères) ;
- votre nom à la place de M. Xxxxxxxxxx (maximum 15 caractères).

L'ensemble est alors prêt à enregistrer fidèlement les messages ; faire simplement :

RUN REPOMDEUR

La première tâche du programme sera de créer le fichier (le message Erreur 18 s'affiche si la disquette est protégée en écriture...).

Changer de nom

Petite simplification de programmation, le changement du nom et du numéro de téléphone n'est pas prévu. La méthode est la suivante : effacer le fichier de message par

```
DELETE FIC.REPOMDEUR
```

puis modifier la ligne 26 et ...

```
RUN REPOMDEUR
```

Au menu

Quatre options sont proposées à l'écran de l'Apple // :

1 **Activer le serveur** : votre ordinateur se met à l'écoute de la ligne et attend patiemment le premier appel. On sort de ce mode par ESC ;
2 **Lire les messages** : vous avez pu aussi les lire à distance ;
3 **Effacer les messages** : c'est aussi une opération que vous pourriez faire à distance ;
4 **Quitter le programme.**

Le serveur en service

Rien de passionnant n'apparaît sur l'écran de l'Apple lors du fonctionnement du serveur ; trois messages seulement :

```
Attente d'un appel
Appel reçu, attente de connexion
Connexion ok, serveur actif.
```

Côté correspondant utilisateur, c'est plus complet ; après deux pages de présentation du répondeur le menu principal apparaît offrant :

```
1 Laisser un message
2 Fonctions de service
3 Quitter ce répondeur
```

L'option 1 invite sur une page écran à laisser *nom*, *prénom* et *numéro de téléphone* puis 7 *lignes de message*. Un écran de validation s'affiche ensuite pour inviter le correspondant (et toujours client) à confirmer, à corriger ou à annuler le message.

L'option 2, qui donne accès aux fonctions de service, demande en premier lieu le mot de passe. Seules trois tentatives sont permises avant déconnexion forcée (même si l'on revient au sommaire après chaque essai infructueux).

Le menu des fonctions de service est le suivant :

```
1 Lire les messages
2 Effacer le fichier
3 Changer le mot de passe
```

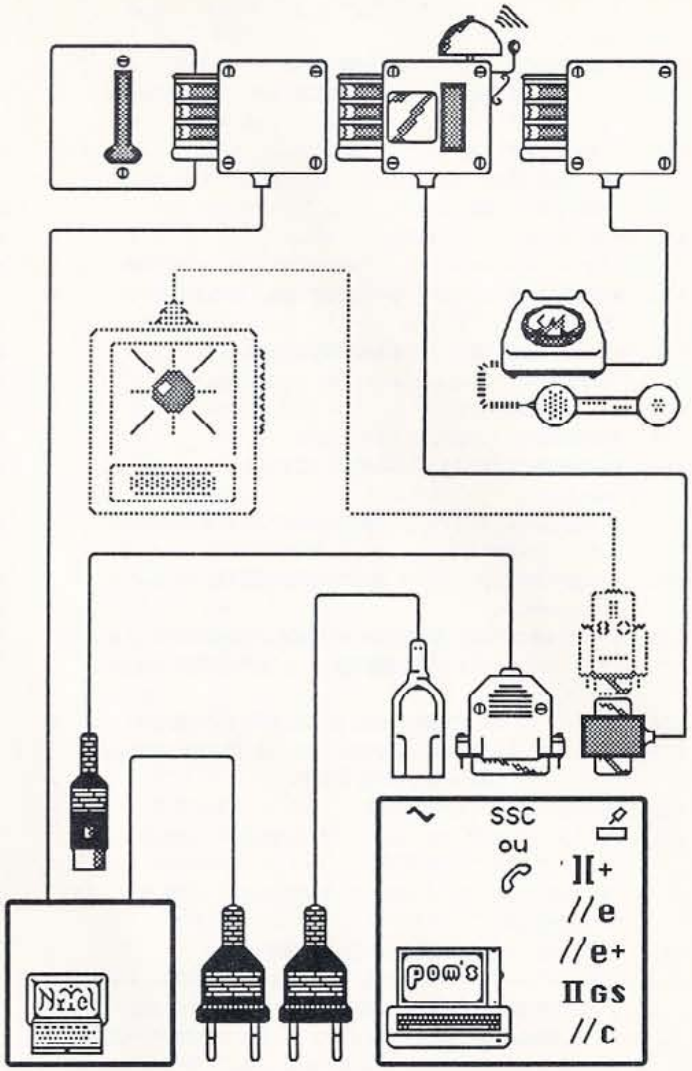
L'option 1 affiche le dernier message reçu ; on accède aux différents enregistrements avec SUITE et RETOUR.

L'option 2 évite les effacements involontaires de fichier en attendant les trois lettres OUI avant de supprimer les messages.

Le changement de mot de passe par l'option 3 est fiable : les caractères frappés au clavier sont remplacés à l'écran par des '*' par souci de discrétion et le nouveau mot est demandé deux fois pour être à l'abri des fautes de frappe.

Programme 'REPOMDEUR'

```
2  REM ----------reloge éventuellement---------
3  IF  PEEK (104) * 256 +  PEEK (103) < > 7681
     THEN  POKE 103,1: POKE 104,30: POKE 7680,
   0: PRINT  CHR$ (4)"-repomdeur"
6  PRINT  CHR$ (21): HOME
10  PRINT  CHR$ (4)"-pom.link.2.1
11  & INIT,2:N$ = "":PR$ = "":TL$ = "": FOR I
    = 0 TO 6:M$(I) = "": NEXT :PO = 0
12 D$ =  CHR$ (4):FI$ = "fic.repomdeur"
22  ONERR  GOTO 25
23  PRINT D$"open"FI$",l255": PRINT D$"read"FI$
    ",r0": INPUT LG,MO$,TE$,NO$
24  GOTO 30
25  ONERR  GOTO 20000
26 MO$ = "ABCDE":TE$ = "(1) 12 34 56 78":NO$ =
    "M. Xxxxxxxx"
27  PRINT D$"write"FI$",r0": PRINT 0: PRINT MO$
    : PRINT TE$: PRINT NO$
30  PRINT D$"close
38  GOTO 9000
40  HOME :PO = 0:ES = 0
45 U$ = "Attente d'un appel (Esc = menu)": GOSU
    B 8000
49  REM ----------attente de l'appel----------
50  & APPEL,A%
51  IF  PEEK (49152) = 155 THEN  POKE 49168,0:
    GOTO 9000
52  ON  NOT A% GOTO 50
55 U$ = "Appel reçu, attente de connexion": GOS
    UB 8000
60  & CNXR,A%
70  ON  NOT A% GOTO 10010
75 U$ = "Connexion ok, serveur actif": GOSUB 80
    00
97  REM ---------------1ère page---------------
98  & MTH,1
99  & DEPLACE,1,13
100  & AFFICHE," /""""ç   /"""ç  /""""""""""ç /
    ç /""""""ç"
101  & AFFICHE,"/      ç /     ç é          è é
    è é /"ç_/"
102  & AFFICHE,"é /""ç è é /"ç è é /"ç /"ç è é
    è é ç__"
103  & AFFICHE,"é é  è è é é è è é é è é è é é
    / ç    ç"
104  & AFFICHE,"é é  è è é é è è é é è é è é "
       ç__  ç"
105  & AFFICHE,"é ç__/ è é ç_/ è é é é é é é
         ç è"
106  & AFFICHE,"é      / ç     / é è é é é é
    /"ç_/ è"
107  & AFFICHE,"é ç___/   ç___/  ç_/ ç_/ ç_/
    ç_____/"
108  & AFFICHE,"é è": & AFFICHE,"é è": & AFF
    ICHE,"é è": & AFFICHE,"ç_/     ";
109  & PARAMT,1,2,0,0,0,0: & AFFICHE,"- La re
    vue des Apple -";
110  & HAUT,12: & BAS,4: & DEPLACE,9,17: & P
    ARAMT,0,2,0,0,0,0
112  & AFFICHE,"_" + CHR$ (18) + CHR$ (86);:
     & DEPLACE,12,18: & PARAMT,0,3,0,0,0,0:
     & AFFICHE,"_" + CHR$ (18) + CHR$ (92)
    ;
119  & MTH,0
120  FOR I = 1 TO 1000: NEXT
199  REM ----------2ème page----------
200  & VIDECRAN
210  & CADRE,1,1,40,7,2,0,5,0: & CADRE,1,8,40
    ,20,2,0,5,0
220  & DEPLACE,4,3: & AFFICHE,"Vous ^etes bi
    en au " + TE$;
230  & DEPLACE,4,5: & MODE,2: & AFFICHE,NO$;
240  & DEPLACE,4,10: & AFFICHE,"Votre interl
    ocuteur est un";
250  & DEPLACE,4,12: & AFFICHE,"ordinateur A
    pple pr^et à enregistrer";
260  & DEPLACE,4,14: & AFFICHE,"vos messages
     et confidences...";
270  & DEPLACE,13,18: & AFFICHE,"Accès au me
```



```
    nu : ";: & INVERSE : & AFFICHE," Sui
    te ";
280 & DEPLACE,18,19: & AFFICHE,"Renoncer :
    ";: & INVERSE : & AFFICHE," Connexion
    ";
290 & ACCEPTF,A%
300 ON NOT A% GOTO 10000: ON A% GOTO 500,340,
    200,340,340,500,500,10010,340
340 GOSUB 4000: GOTO 290
499 REM ---------------Sommaire---------------
500 PRINT D$"close": & VIDECRAN: & VIDRECT,
    1,1,40,6,4
510 & DEPLACE,9,4: & PARAMT,1,7,4,0,0,0: &
    AFFICHE," Répondeur télématique";
520 GOSUB 6000
550 & PARAMT,0,7,0,0,0,0
560 & DEPLACE,8,11: & AFFICHE,"Laisser un m
    essage";
570 & DEPLACE,8,14: & AFFICHE,"Fonctions de
     service";
580 & DEPLACE,8,17: & AFFICHE,"Quitter ce r
    épondeur";
590 & DEPLACE,15,22: & AFFICHE,"Votre choix
    :  + ";: & INVERSE : & AFFICHE," Env
    oi ";
600 CH$ = "": & ACCEPT,29,22,0,"",1,CH$,E%
610 ON NOT E% GOTO 10000: ON E% GOTO 650,200,
    500,620,600,600,650,10010
620 GOSUB 5000: GOTO 600
630 IF CH$ < "1" OR CH$ > "3" THEN GOSUB 7000
    : GOTO 600
650 IF CH$ < "1" OR CH$ > "3" THEN GOSUB 7000
    : GOTO 600
660 ON VAL (CH$) GOTO 700,3000,9900
700 REM -----------Enreg du message-----------
710 & VIDECRAN: & DEPLACE,3,2: & INVERSE
    : & PARAMT,0,7,4,0,0,0: & AFFICHE," En
    registrement de votre message ";: & CADR
    E,2,1,38,3,0,0,2,0
720 & DEPLACE,1,5: & AFFICHE,"Votre nom : .
    ..... + ";: & LIGNE: & AFFICH
    E," ";: & INVERSE : & AFFICHE," Suite
    ";
730 & DEPLACE,1,6: & AFFICHE," Prénom : .
    ..... + ";: & LIGNE: & AFFICH
    E," ";: & INVERSE : & AFFICHE," Suite
    ";
740 & DEPLACE,1,7: & AFFICHE,"Téléphone : .
    ..... + ";: & LIGNE: & AFFICH
    E," ";: & INVERSE : & AFFICHE," Suite
    ";
750 & CHARIOT: & NORMAL : & MODE,2: & AFF
    ICHE,"_____Message_____";
760 FOR I = 11 TO 17
770 & DEPLACE,1,I: & AFFICHE,".....
    ..... + ";: & LIGNE: & AFFICH
    E," ";: & INVERSE : & AFFICHE," Suite
    ";
780 NEXT
790 & DEPLACE,1,20: & AFFICHE,"Pour valider
     votre message :";: & LIGNE: & AFFICHE,
    " ";: & INVERSE : & AFFICHE," Envoi
    ";: & DEPLACE,13,21: & AFFICHE,"Pour l
    'annuler :";: & LIGNE: & AFFICHE," ";:
    & INVERSE : & AFFICHE," Sommaire ";
800 & DEPLACE,13,5: & AFFICHE,N$;
801 & DEPLACE,13,6: & AFFICHE,PR$;
802 & DEPLACE,13,7: & AFFICHE,TL$;
803 FOR I = 11 TO 17: & DEPLACE,1,I: & AFFIC
    HE,M$(I - 11);: NEXT
805 ON PO GOTO 810,850,880,910
810 PO = 1: & ACCEPT,13,5,0,"",16,N$,E%
820 ON NOT E% GOTO 10000: ON E% GOTO 840,830,
    700,835,830,500,840,10010
830 GOSUB 4000: GOTO 810
835 GOSUB 5000: GOTO 810
840 IF N$ = "" THEN & ALERTE,"Nom indispensa
    ble...",1: GOTO 810
845 IF E% = 1 GOTO 990
850 PO = 2: & ACCEPT,13,6,0,"",16,PR$,E%
860 ON NOT E% GOTO 10000: ON E% GOTO 990,810,
    700,875,870,500,880,10010
870 GOSUB 4000: GOTO 850
875 GOSUB 5000: GOTO 850
880 PO = 3: & ACCEPT,13,7,0,"",16,TL$,E%
890 ON NOT E% GOTO 10000: ON E% GOTO 990,850,
    700,905,900,500,910,10010
900 GOSUB 4000: GOTO 880
905 GOSUB 5000: GOTO 880
910 NN = 0
915 V = NN + 11
920 PO = 4: & DEPLACE,1,V: & ACCEPT,1,V,0,"",
    28,M$(NN),E%
930 ON NOT E% GOTO 10000: ON E% GOTO 990,980,
    700,935,940,500,950,10010
935 GOSUB 5000: GOTO 915
940 GOSUB 4000: GOTO 915
950 NN = NN + 1: ON NN = 7 GOTO 990: GOTO 915
980 NN = NN - 1: ON NN < > - 1 GOTO 915:NN =
    0: GOTO 880
989 REM ----------écran de validation---------
990 & VIDECRAN: & DEPLACE,1,3: & AFFICHE,"
    Voici votre message tel qu'il sera": & AF
    FICHE,"enregistré :"
1000 & DEPLACE,4,7: & AFFICHE," Nom : " +
    N$;
1002 & DEPLACE,4,8: & AFFICHE,"Prénom : " +
    PR$;
1004 & DEPLACE,4,9: & AFFICHE,"Téléph : " +
    TL$;
1006 FOR I = 0 TO 6: & DEPLACE,6,11 + I: & A
    FFICHE,M$(I);: NEXT
1010 & DEPLACE,1,21
1020 & AFFICHE,"Pour le valider.....";:
    & LIGNE: & AFFICHE," ";: & INVERSE :
    & AFFICHE," Envoi ": & NORMAL
1030 & AFFICHE,"Pour le modifier.....";:
    & LIGNE: & AFFICHE," ";: & INVERSE :
    & AFFICHE," Correction ": & NORMAL
1040 & AFFICHE,"Pour l'annuler.....";:
    & LIGNE: & AFFICHE," ";: & INVERSE :
    & AFFICHE," Annulation ": & NORMAL
1050 & ACCEPTF,A%
1060 ON NOT A% GOTO 10000: ON A% GOTO 1100,70
    0,990,1070,500,500,1065,10010,700
1065 & ALERTE,"Touche interdite",0: GOTO 1050
1070 GOSUB 5000: GOTO 1050
1099 REM -----------enregistrement----------
1100 & VIDECRAN: & DEPLACE,6,12: & AFFICHE
    ,"Enregistrement en cours";
1103 PRINT D$"open"FI$",1255"
1105 PRINT D$"read"FI$",r0
1110 INPUT LG
1120 LG = LG + 1
```

```
1130  PRINT D$"write"FI$",r0": PRINT LG: PRINT
      MO$: PRINT TE$: PRINT NO$
1140  PRINT D$"write"FI$",r"LG
1150  PRINT N$: PRINT PR$: PRINT TL$: FOR I = 0
       TO 6: PRINT M$(I): NEXT
1160  PRINT D$"close
1165  FOR I = 0 TO 6:M$(I) = "": NEXT
1169  REM ----------dernier écran----------
1170  & VIDECRAN: & DEPLACE,6,12: & AFFICHE
      ,"Votre message est enregistré...": & DEP
      LACE,11,14: & AFFICHE,"...faites ";: &
       INVERSE : & AFFICHE," Sommaire ";: & A
      CCEPTF,A%: ON A% = 0 OR A% = 8 GOTO 1001
      0:PO = 4: GOTO 500
3000  REM ----------fonctions de service-------
3010  & VIDECRAN: & VIDRECT,1,1,40,6,4: & DE
      PLACE,10,4: & PARAMT,1,7,4,0,0,0: & AFF
      ICHE," Fonctions de service";: & PARAMT,
      0,7,0,0,0,0
3012  & DEPLACE,32,12: & AFFICHE,"+ ";: &  I
      NVERSE : & AFFICHE," Envoi ";
3013  & DEPLACE,15,15: & AFFICHE,"Pour renon
      cer : ";: &  INVERSE : & AFFICHE," Somm
      aire ";
3014 MP$ = "": & SECRET,9,12,0,"Mot de passe :
       ",7,MP$,E%
3015  ON  NOT E% GOTO 10000: ON E% GOTO 3018,50
      0,3000,3016,3017,500,3018,10010
3016  GOSUB 5000: GOTO 3014
3017  GOSUB 4000: GOTO 3014
3018  IF MP$ = MO$ GOTO 3028
3020 ES = ES + 1
3022  IF ES = 1 THEN  & ALERTE,"Mot de passe e
      rroné",1: GOTO 3014
3023  IF ES = 2 THEN  & ALERTE,"Erreur, encore
       un essai",1: GOTO 3014
3024  & VIDECRAN: & DEPLACE,10,12: & AFFICH
      E,"Désolé, déconnexion forcée...";: GOTO
      10010
3028 ES = 0: & VIDLIGNE,12,0: & VIDLIGNE,15,
      0: GOSUB 6000
3030  & DEPLACE,8,11: & AFFICHE,"Lire les me
      ssages";: & DEPLACE,8,14: & AFFICHE,"Ef
      facer le fichier";: & DEPLACE,8,17: & AF
      FICHE,"Changer le mot de passe";
3040  & DEPLACE,15,22: & AFFICHE,"Votre choi
      x :   + ";: &  INVERSE : & AFFICHE," En
      voi ";
3048 CH$ = ""
3050 CH$ = "": & ACCEPT,29,22,0,"",1,CH$,E%
3060  ON  NOT E% GOTO 10000: ON E% GOTO 3070,30
      00,3065,3066,3065,500,3070,10010
3065  GOSUB 4000: GOTO 3050
3066  GOSUB 5000: GOTO 3050
3070  IF CH$ < "1" OR CH$ > "3" THEN  GOSUB 700
      0: GOTO 3048
3075  ON  VAL (CH$) GOTO 3100,3300,3500
3100  REM ----------lecture fic----------
3110  PRINT D$"open"FI$",1255
3120  PRINT D$"read"FI$",r0": INPUT LG
3130  IF LG = 0 THEN  PRINT D$"close": & ALERT
      E,"Aucun message reçu",1: GOTO 3050
3135 NO = LG
3140  & VIDECRAN: & PARAMT,0,2,0,0,0,0: & AF
      FICHE," " + CHR$ (18) + CHR$ (64 + 39);
3142  & DEPLACE,2,3: & MODE,1: & AFFICHE,"Co
      nsultation des messages enregistrés": & M
      ODE,0
3144  & PARAMT,0,2,0,0,0,0: & MTH,1: & AFFIC
      HE,"" + CHR$ (18) + CHR$ (64 + 39);: &
       MTH,0: & DEPLACE,9,5: & PARAMT,0,6,0,
      0,0,0: & AFFICHE,"Précédent :";
3146  & LIGNE: & AFFICHE," ";: &  INVERSE :
       & AFFICHE,"  Retour  ": & DEPLACE,11,6
      : & PARAMT,0,6,0,0,0,0: & AFFICHE,"Suiv
      ant :";: & LIGNE: & AFFICHE," ";: &  IN
      VERSE : & AFFICHE,"  Suite   "
3148  & DEPLACE,15,7: & PARAMT,0,6,0,0,0,0: &
       AFFICHE,"Fin :";: & LIGNE: & AFFICHE,
      " ";: &  INVERSE : & AFFICHE," Sommaire
       "
3150  & PARAMT,0,2,0,0,0,0: & AFFICHE,"_" +
      CHR$ (18) + CHR$ (64 + 39): & AFFICHE,"
       " + CHR$ (18) + CHR$ (64 + 39);
3152  & DEPLACE,1,12: & PARAMT,0,4,0,0,0,0: &
       AFFICHE,"`" + CHR$ (18) + CHR$ (64 +
      39);: & DEPLACE,1,16: & PARAMT,0,4,0,0,
      0,0: & AFFICHE,"`" + CHR$ (18) + CHR$
      (64 + 39)
3154  & DEPLACE,1,24: & PARAMT,0,2,0,0,0,0: &
       AFFICHE,"`" + CHR$ (18) + CHR$ (64 +
      39);
3170  PRINT D$"read"FI$",r"NO
3180  FOR I = 1 TO 3: INPUT I$(I): NEXT
3190  FOR I = 0 TO 6: INPUT MS$(I): NEXT
3192  & DEPLACE,2,10: & AFFICHE," Message n°
       " + STR$ (NO);
3194  & DEPLACE,2,13: & AFFICHE,"``> " + I$(
      1) + " " + CHR$ (24)
3196  & DEPLACE,2,14: & AFFICHE,"``> " + I$(
      2) + " " + CHR$ (24)
3198  & DEPLACE,2,15: & AFFICHE,"``> " + I$(
      3) + " " + CHR$ (24)
3200  FOR I = 1 TO 7: & DEPLACE,2,16 + I: & A
      FFICHE,"``` " + MS$(I - 1) + " " + CHR$
       (24);: NEXT
3220  & ACCEPTF,A%: ON  NOT A% GOTO 10000
3230  ON A% GOTO 3240,3250,3140,3235,3240,500,3
      260,10010,3240
3235  GOSUB 5000: GOTO 3220
3240  GOSUB 4000: GOTO 3220
3250 NO = NO - 1
3252  ON NO > 0 GOTO 3170:NO = 1: & ALERTE,"C'
      est le premier message",1: GOTO 3220
3260 NO = NO + 1
3262  ON NO <  = LG GOTO 3170:NO = LG: & ALERT
      E,"C'est le dernier",1: GOTO 3220
3299  REM ----------effacer le fichier---------
3300  & VIDECRAN: & VIDRECT,1,1,40,6,4: & DE
      PLACE,10,4: & PARAMT,1,7,4,0,0,0: & AFF
      ICHE," Effacement fichier ";: & PARAMT,0
      ,7,0,0,0,0
3310  & DEPLACE,5,10: & AFFICHE,"Cette comma
      nde supprime l'ensemble": & DEPLACE,5,11
      : & AFFICHE,"des messages stockés."
3320  & DEPLACE,5,13: & AFFICHE,"On ne peut
      annuler un effacement..."
3325 R$ = "": & DEPLACE,5,22: & AFFICHE,"Pou
      r effacer :  ";: &  INVERSE : & AFFICHE
      ," OUI + Envoi ";
3327  & DEPLACE,5,23: & AFFICHE,"Pour renonc
      er : ";: &  INVERSE : & AFFICHE,"  Somm
      aire   ";
3330 R$ = "": & ACCEPT,10,16,0,"Ok pour tout e
```

```
ffacer ? ",3,R$,E%
3340 ON NOT E% GOTO 10000: ON E% GOTO 3380,33
50,3300,3360,500,500,3350,10010
3350 GOSUB 4000: GOTO 3330
3360 GOSUB 5000: GOTO 3330
3380 IF R$ < > "OUI" THEN R$ = "": & ALERTE,
"Réponse incorrecte",0: GOTO 3330
3390 PRINT D$"open"FI$",l255"
3400 PRINT D$"write"FI$",r0": PRINT 0: PRINT M
O$: PRINT TE$: PRINT NO$
3410 PRINT D$"close"
3420 & VIDECRAN: & DEPLACE,6,12: & AFFICHE
,"Le fichier a été réinitialisé...": & DE
PLACE,11,14: & AFFICHE,"...faites ";: &
INVERSE : & AFFICHE," Sommaire ";: &
ACCEPTF,A%: ON A% = 0 OR A% = 8 GOTO 100
10: GOTO 500
3499 REM ----------changer le mot de passe----
3500 & VIDECRAN: & VIDRECT,1,1,40,6,4: & DE
PLACE,14,4: & PARAMT,1,7,4,0,0,0: & AFF
ICHE,"Mot de passe";: & PARAMT,0,7,0,0,0
,0
3510 & DEPLACE,1,23: & AFFICHE,"Pour renonc
er, faites : ";: & INVERSE : & AFFICHE
," Sommaire ";
3520 & DEPLACE,1,11: & AFFICHE,"Taper le no
uveau mot de passe + ";: & INVERSE : &
AFFICHE," Envoi ";
3530 & DEPLACE,1,16: & AFFICHE,"Taper le à
nouveau pour contr^ole";
3540 & CADRE,25,12,35,14,2,0,5,0
3550 & CADRE,25,17,35,19,2,0,5,0
3560 R1$ = "": & SECRET,27,13,0,"",7,R1$,E%
3570 ON NOT E% GOTO 10000
3580 ON E% GOTO 3600,3590,3500,3595,3590,500,3
600,10010
3590 GOSUB 4000: GOTO 3560
3595 GOSUB 5000: GOTO 3560
3600 IF LEN (R1$) < 4 THEN & ALERTE,"Minimu
m 4 caractères",1: GOTO 3560
3610 R2$ = "": & SECRET,27,18,0,"",7,R2$,E%
3620 ON NOT E% GOTO 10000
3630 ON E% GOTO 3680,3640,3500,3645,3640,500,3
680,10010
3640 GOSUB 4000: GOTO 3610
3645 GOSUB 5000: GOTO 3610
3680 IF R1$ < > R2$ THEN & ALERTE,"Les deux
mots de passe diffèrent",1: GOTO 3560
3690 MO$ = R1$
3700 PRINT D$"open"FI$",l255"
3710 PRINT D$"write"FI$",r0": PRINT LG: PRINT
MO$: PRINT TE$: PRINT NO$
3720 PRINT D$"close"
3730 & VIDECRAN: & DEPLACE,6,12: & AFFICHE
,"Le mot de passe a été changé... ": & D
EPLACE,11,14: & AFFICHE,"...faites ";:
& INVERSE : & AFFICHE," Sommaire ";: &
ACCEPTF,A%: ON A% = 0 OR A% = 8 GOTO 10
010: GOTO 500
3999 REM ----------sous-prog- ----------
4000 & ALERTE,"Touche inactive ici",0: RETURN

5000 & ALERTE,"Guide indisponible",0: RETURN
6000 & CADRE,4,10,6,12,2,0,5,0: & AFFICHE,"1
";: & CADRE,4,13,6,15,2,0,5,0: & AFFICH
E,"2";: & CADRE,4,16,6,18,2,0,5,0: & AFF
ICHE,"3";: RETURN
7000 & ALERTE,"Choix impossible",1: RETURN
8000 VTAB 10: PRINT U$;: CALL - 868: PRINT :
RETURN
9000 REM ----------menu général----------
9010 VTAB 23: FOR I = 1 TO 23: PRINT : NEXT
9020 PRINT : PRINT : PRINT "Voulez-vous :": PR
INT
9030 PRINT " 1 activer le serveur
9040 PRINT " 2 lire les messages
9050 PRINT " 3 effacer les messages
9055 PRINT " 4 quitter ce programme": PRINT
: PRINT
9060 VTAB 23: PRINT " Votre choix ";: G
ET R$: PRINT : ON R$ < "1" OR R$ > "4" GOT
O 9060
9070 ON VAL (R$) GOTO 40,9100,9400,9080
9080 HOME : END
9100 PRINT : PRINT
9110 PRINT D$"open"FI$",l255
9120 PRINT D$"read"FI$",r0": INPUT LG: PRINT D
$
9130 IF LG = 0 THEN PRINT D$"close": PRINT "A
ucun message reçu": PRINT : PRINT : GOTO 9
020
9140 PRINT "Vous avez "LG" message";: IF LG >
1 THEN PRINT "s";
9150 PRINT : PRINT
9160 INPUT "N° du message à lire (0 pour finir
): ";N1$:N1 = VAL (N1$)
9170 IF N1 < 0 OR N1 > LG THEN PRINT "De 1 à
"LG"...": GOTO 9150
9172 ON NOT N1 GOTO 9250
9175 PRINT : PRINT "Message n° "N1" :": PRINT
9180 PRINT D$"read"FI$",r"N1
9190 FOR I = 1 TO 3: INPUT A$: PRINT "--> "A$:
NEXT
9200 FOR I = 1 TO 7: INPUT A$: PRINT "* "A$: N
EXT
9205 PRINT D$
9210 GOTO 9150
9250 PRINT D$"close": GOTO 9020
9400 PRINT : PRINT "Ok pour effacer TOUS les m
essages ?"
9410 INPUT "Répondre OUI pour effacer ";N1$: O
N N1$ < > "OUI" GOTO 9500
9420 PRINT D$"open"FI$",l255
9430 PRINT D$"write"FI$",r0": PRINT 0: PRINT M
O$: PRINT TE$: PRINT NO$
9440 PRINT D$"close"
9450 PRINT : PRINT "Messages effacés": GOTO 90
20
9500 PRINT : PRINT "Ordre annulé": GOTO 9020
9899 REM ----------fin & déconnexion----------
9900 & VIDECRAN: & DEPLACE,10,12: & AFFICH
E,"Au revoir...";
9910 PRINT D$"close": & DECNX: GOTO 40
10000 & VIDECRAN: & DEPLACE,3,12: & FLASH
: & AFFICHE,"Minitel inactif depuis 2 m
inutes,";: & DEPLACE,9,14: & AFFICHE,"D
éconnexion forcée.";
10010 PRINT D$"close": & DECNX: GOTO 40
19999 REM ----------erreur----------
20000 & VIDECRAN: & AFFICHE,"Désolé, problè
me serveur": & DECNX: PRINT D$"close": PR
INT "erreur " PEEK (222): END
```

Pom_Link 2.1, les sources

Christian Piard

Les sources Procode, pour Apple //, présentés ici doivent permettre à chacun d'améliorer, d'adapter à ses propres besoins les nouvelles instructions serveur.

Ils sont présents en format texte sur la disquette d'accompagnement de ce numéro et sont récupérables par tout autre assembleur.

Source PL.0
Assembleur ProCODE

```
        lst  off
        dsk pom.link.2.1
        put  pl.1
        put  pl.2
        put  pl.3
        put  pl.4
        put  pl.sp
        put  pl.data
```

Source PL.1
Assembleur ProCODE

```
 0  *0701_1535
 1  *--------------------*
 2  *      NetPom's      *
 3  *--------------------*
 4
 5  chkcom  =   $DEBE
 6  chkstr  =   $DD6C
 7  chrget  =   $B1
 8  chrgot  =   $B7
 9  cout    =   $FDED
10  data    =   $18
11  error   =   $D412
12  errout  =   $BE09
13  facmo   =   $A0
14  forpnt  =   $85
15  frmevl  =   $DD7B
16  fretop  =   $6F
17  gethytc =   $E6F5
18  getspa  =   $E452
19  kbd     =   $C000
20  miserr  =   $DD76
21  movstr  =   $E5E2
22  ptr     =   $8
23  ptrget  =   $DFE3
24  snerr   =   $DEC9
25  status  =   $1A
26  varnam  =   $81
27  varpnt  =   $83
28
29          org  $801
30
31          lda  $3F6       ;vecteur & déjà
32          cmp  f<debut    ;installé ?
33          bne  $0
34          lda  $3F7
35          cmp  f>debut
36          beq  ok
37  $0      ldy  £2         ;sauve ancien
38  $1      lda  $3F5,y     ;vecteur
39          sta  vecteur,y
40          dey
41          bpl  $1
42          lda  f$4C       ;installe nouveau
43          sta  $3F5       ;vecteur &
44          lda  f<debut
45          sta  $3F6
46          lda  f>debut
47          sta  $3F7
48  ok      lda  £0         ;est-on sur un
49          sta  deuxc      ;//c ?
50          lda  $FBB3
51          cmp  £6
52          bne  $ok
53          lda  $FBC0
54          bne  $ok
55          dec  deuxc
56  $ok     rts
57
58  debut   ldy  £0         ;init longueur cmd
59          sty  cmd
60          sty  nocmd
61  $0      =    *          ;cherche
62  $1      inc  cmd        ;la commande
63          ldy  cmd
64          cpy  £12
65          bne  $11
66          jmp  snerr
67  $11     ora  £$A0       ;transforme en
68          sta  cmd,y      ;minuscule et
69          jsr  chrget     ;stocke les caract
70          beq  fincmd     ;jusqu'à 1 virgule
71          cmp  £','
72          bne  $0
73
74
75
76
77
78  fincmd  ldy  £$FF       ;compare à chacune
79          sty  ptrcmd     ;des commandes
80          inc  ptrcmd
81          ldy  ptrcmd
82          lda  commande,y
83  $1      cmp  cmd        ;compare d'abord
84          bne  $2         ;la longueur
85          ldx  £0         ;puis le nom
86  $100    inx             ;lui-meme
87          inc  ptrcmd
88          ldy  ptrcmd
89          lda  commande,y
90          cmp  cmd,x
91          bne  $2
92          cpx  cmd
93          beq  trouve
94          bne  $100
95  $2      inc  ptrcmd     ;cherche ordre
96          ldy  ptrcmd     ;suivant dans
97          lda  commande,y ;table
98          beq  pas_eu
99          bmi  $2
100         inc  nocmd      ;n° de la commande
101         jmp  $1
102
103 trouve  lda  £<adrcom   ;on a trouvé
104         sta  sautind+1  ;initialise
105         lda  £>adrcom   ;le saut
106         sta  sautind+2  ;vers la
107         ldy  nocmd      ;routine concernée
108         beq  sautind
109 $1      inc  sautind+1
110         bne  $2
111         inc  sautind+2
112 $2      inc  sautind+1
113         bne  $3
114         inc  sautind+2
115 $3      dey
116         bne  $1
117
118 sautind jmp  (adrcom)   ;saute
119
120 pas_eu  jmp  snerr      ;syntax error
```

Quel serait l'intérêt d'un serveur si l'ordinateur ne sait pas que le téléphone sonne ? Rendez-vous page 75 ou, pour les bricoleurs, dans le numéro 32.

Source PL.2
Assembleur ProCODE

```
*--------------------
* accept
*--------------------

accept   lda  £0         ;ce n'est pas
         sta  dsecret    ;secret
accept1  lda  £$FF
         sta  drap
         jsr  getbytc    ;position horiz
         stx  lhor
         jsr  getbytc    ;position verti
         stx  lver
         cpx  £25
         bcc  $a
         jmp  horslim
$a       jsr  getbytc    ;mode dble haut.
         cpx  £2         ;0 ou 1 seulement
         bcc  $a0
         jmp  horslim
$a0      stx  dblh
         ldy  lver       ;positionne
         ldx  lhor       ;le curseur
         jsr  position
         jsr  curseuri   ;invisible
         jsr  chkcom
         lda  £0
         sta  $52
         jsr  frmevl     ;évalue chaine
         jsr  chkstr     ;c'est 1 chaine ?
         lda  facmo      ;récupère pointeur
         sta  forpnt     ;sur chaine
         lda  facmo+1    ;temporaire
         sta  forpnt+1
         ldy  £0
         lda  (forpnt),y
         tax
         clc
         adc  lhor
         sta  lhor
         ldy  £2
$0       lda  (forpnt),y ;récupère adresse
         dey             ;de la chaine
```

```
        sta  ptr,y
        cpy  £0
        bne  $0
$1      dex                 ;l'affiche
        bmi  $finaff
        lda  (ptr),y
        jsr  envoicar
        iny
        jmp  $1
$finaff jsr  gethytc        ;attend lg%
        stx  lgatt
        cpx  £0             ;pas 0
        beq  $pas0
        cpx  £41
        bcc  $lgok
$pas0   jmp  horslim
$lgok   lda  lhor           ;controle si les
        clc                 ;pts tiennent
        adc  lgatt          ;dans l'écran
        cmp  £41
        bcc  $lgok1
        jmp  horslim
$lgok1  jmp  chaidef        ;chaine par défaut
debsais lda  £0             ;init compteurs
        sta  al
        sta  depasst
        inc  al             ;al=1:efface lg 0
debl    jsr  initcpt        ;init délai
deb2    =    *
        jsr  curseurv
$0      inc  tempo          ;inc les compt.
        bne  $1
        inc  tempo+1
        bne  $1
        inc  tempo+2
        bne  $1
        jmp  tpsdepas
$1      jsr  getcar         ;attend un caract.
        lda  lg
        beq  $0
        lda  $200
        and  £$7F
        sta  carrecu
        jsr  initcpt        ;compteurs à 0
        lda  al             ;si nécessaire,
        beq  $11            ;efface ligne 0
        jsr  effal
$11     lda  carrecu
        cmp  £13            ;retour chariot ?
        beq  $110           ;oui, = à envoi
        cmp  £10            ;touche fonction ?
        bcs  $2
$110    jmp  tchfonc        ;oui
$2      cmp  £$20           ;carac controle ?
        bcc  debl
        lda  nbcar          ;controle longueur
        cmp  lgatt
        bne  $20
        jmp  trop
$20     inc  nbcar
        ldy  nbcar
        lda  carrecu
        cmp  £','           ;transforme , & :
        bne  $21            ;en des .
        lda  £'.'
$21     cmp  £':'
        bne  $22
        lda  £'.'
$22     sta  $200,y         ;stocke car reçu
        bit  dsecret        ;(si secret,
        bpl  $23            ;* en écho)
        lda  £'*'
$23     jsr  envoi          ;en $201..2FF
        jmp  debl
trop    lda  £1             ;carac hors zone
        sta  al
        jsr  curseuri
        ldy  £0             ;affiche message
$1      lda  $lg0,y         ;fin de zone
        beq  $2
        jsr  envoi
        iny
        bne  $1
$2      jsr  curseurv
        jmp  debl
$lg0    dfb  7,$1F,$40,$41,$20
        dfb  $12,$64,$1F,$40,$41
        asc  'Fin de zone'
        dfb  $0A,0

tpsdepas lda  depasst       ;1 minute
        beq  $0             ;inactif
        jmp  nonrep         ;2 fois : fini
$0      lda  £1             ;1ère minute
        sta  al             ;message d'alerte
        sta  depasst        ;dcnx imminente
        jsr  curseuri
        ldy  £0
$1      lda  decim,y
        beq  $2
        jsr  envoi
        iny
        bne  $1
$2      jsr  curseurv
        jsr  initcpt1
        jmp  deb2

decim   dfb  7,$1F,$40,$41,$20
        dfb  $12,$64,$1F,$40,$41
        asc  'D'
        dfb  $19,$42
        asc  'econnexion imminente...
        dfb  $0A,0

tchfonc jsr  initcpt
        lda  carrecu
        cmp  £13            ;transforme RC
        bne  $00            ;en envoi
        lda  £1
        sta  carrecu
$00     cmp  £7             ;est-ce correct ?
        beq  $1
        cmp  £5             ;annulation ?
        beq  $2
        jmp  finacc         ;non, fin d'accept
$1      lda  nbcar          ;correction, 1 car
        beq  rien           ;à effacer ?
        lda  lhor           ;recule d'un cran
        clc
        adc  nbcar
        tax
        dex
        ldy  lver
        jsr  position
        lda  £'.'           ;affiche un point
        jsr  envoi
        jsr  position       ;repositionne
        dec  nbcar          ;car = car -1
        jmp  debl
$2      ldx  lhor           ;annulation,
        ldy  lver           ;curseur en début
        jsr  position       ;de zone
        jsr  curseuri
        ldx  lgatt          ;affich lg% points
        dex
$20     lda  £"."
        jsr  envoi
        dex
        bpl  $20
        ldx  lhor           ;positionn curseur
        ldy  lver           ;en début de zone
        jsr  position
        jsr  curseurv       ;curseur visible
        lda  nbcar          ;si zone vide, on
        bne  $21            ;sort, sinon on
        jmp  finacc         ;recommence
$21     lda  £0
        sta  nbcar
        jmp  debsais

rien    ldy  £0             ;pas de car à
$1      lda  $lg0,y         ;effacer
        beq  $2
        jsr  envoi
        iny
        bne  $1
$2      jsr  curseurv
        jsr  initcpt1
        lda  £1
        sta  al
        jmp  debl
$lg0    dfb  $1F,$40,$41,$20
        dfb  $12,$64,$1F,$40,$41
        asc  'Rien '
        dfb  $19,$41
        asc  'a effacer...'
        dfb  $0A,0

getcar  jmp  sais

initcpt lda  £0
        sta  depasst        ;dépasst = 0
initcpt1 lda  £0
        sta  tempo          ;tempo = 0
        sta  tempo+1
        bit  pint
        bpl  $1
        lda  £$FD
        bne  $2
$1      lda  £$F8
$2      sta  tempo+2
        rts

effal   jsr  curseuri
        ldy  £6             ;efface ligne 0
$1      lda  $efflg0,y
        jsr  envoi
        dey
        bpl  $1
        lda  £0
        sta  al
        rts
$efflg0 dfb  $0A,$64,$12
        dfb  $20,$41,$40,$1F

lgch    ds   1              ;long chaine
lgatt   ds   1              ;longueur attendue
nomch   ds   2              ;nom chaine retour
nbcar   ds   1
tempo   ds   3
depasst ds   1
al      ds   1              ;drapeau alerte
carrecu ds   1
dblh    ds   1

position jsr  curseuri      ;positionne
        stx  $1             ;le curseur
        sty  $2
        jsr  envoi1F
        lda  $2
        clc
        adc  £$40
        jsr  envoi
        lda  $1
        clc
        adc  £$40
        jsr  envoi
        lda  dblh           ;double hauteur ?
        beq  $0
        jsr  envoi1B
        lda  £$4D
$0      jmp  envoi
$1      ds   1
$2      ds   1
nonrep  lda  £0             ;Minit. répond pas
        sta  carrecu

finacc  =    *
$suite  lda  nbcar          ;cherche espace
        jsr  getspa         ;pour la chaine
        ldx  £1             ;chaine en $201
        ldy  £2
        jsr  movstr         ;la déplace
        lda  nbcar
        ldy  £0
        sta  (varpnt),y
        iny
        lda  fretop
        sta  (varpnt),y
        iny
        lda  fretop+1
        sta  (varpnt),y
        jsr  chkcom         ;attend variable
        jsr  ptrget         ;entière
        sta  varpnt
        sty  varpnt+1
        lda  varnam
        and  varnam+1
        bmi  $o
        jmp  miserr
$o      lda  carrecu        ;si fnct = 8 ou 9,
        cmp  £7             ;enlève 1
        bcc  $o1
        sbc  £1
$o1     ldy  £1             ;stocke état dans
        sta  (varpnt),y     ;variable entière
        ldy  £0
        tya
        sta  (varpnt),y
        jsr  curseuri
        ldx  nbcar
$eff    cpx  lgatt
        beq  $fin
        lda  £" "
        jsr  envoi
        inx
        bne  $eff
$fin    lda  £0
        sta  drap
        rts

chaidef jsr  chkcom         ;attend var chaine
        jsr  ptrget         ;pour afficher la
        lda  varnam         ;chaine par défaut
        sta  nomch
        bmi  $err
        lda  varnam+1
        sta  nomch+1
        bmi  $suite
$err    jmp  miserr
$suite  lda  varpnt
        sta  forpnt
        lda  varpnt+1
        sta  forpnt+1
        ldy  £0             ;charge lg de la
```

```
        lda  (forpnt),y ;chaine
        cmp  lgatt      ;si elle est plus
        bcc  $ok        ;longue que la lg
        beq  $ok        ;attendue : string
        jmp  strtool    ;too long
$ok     sta  nbcar
        sta  $n
        ldy  £2         ;prend l'adresse
$0      lda  (forpnt),y ;de la chaine
        dey
        sta  ptr,y
        cpy  £0
        bne  $0
        ldy  £0         ;affiche la chaine
$1      cpy  nbcar      ;et la stocke à
        beq  $finch     ;partir de $201
        lda  (ptr),y
        sta  $201,y
        jsr  envoi
        iny
        bne  $1
$finch  ldx  nbcar      ;complète
$2      cpx  lgatt      ;avec des points
        beq  $finp
        lda  £"."
        jsr  envoi
        inx
        bne  $2
$finp   lda  lhor       ;positionne curs
        clc             ;en fin de chaine
        adc  nbcar      ;par défaut
        tax
        ldy  lver
        jsr  position
        jmp  debsais
$n      ds   1

*--------------------
* acceptf
*--------------------

acceptf jsr  chkcom
        jsr  ptrget
        sta  varpnt
        sty  varpnt+1
        lda  varnam
        and  varnam+1
        bmi  $a1
        jmp  miserr
$a1     lda  £0         ;init compteurs
        sta  a1
        sta  depasst
        inc  a1

deb3    jsr  initcpt
deb4    =    *
$0      inc  tempo      ;incr compteurs
        bne  $1
        inc  tempo+1
        bne  $1
        inc  tempo+2
        bne  $1
        jmp  tpsdep
$1      jsr  getcar     ;attend un car
        ldx  lg
        beq  $0
        lda  $200
        and  £$7F
        sta  carrecu
        jsr  initcpt    ;compteurs à 0
        lda  a1
        beq  $11
        jsr  effa1
$11     lda  carrecu
        cmp  £13        ;retour chariot ?
        beq  fnrecu     ;oui, = à envoi
        cmp  £10        ;touche fonction ?
        bcc  fnrecu
$110    jmp  deb4       ;oui

nonrp   lda  £0
        sta  carrecu

fnrecu  lda  carrecu
        cmp  £13        ;transforme RC
        bne  $00        ;en envoi
        lda  £1
$00     cmp  £7
        bne  $01
        lda  £9
        bne  $o1
$01     bcc  $o1
        sbc  £1
$o1     ldy  £1         ;stocke état dans
        sta  (varpnt),y ;variab entière
        ldy  £0
        tya
        sta  (varpnt),y
        rts

tpsdep  lda  depasst    ;1 minute
        beq  $0         ;inactif
        jmp  nonrp      ;2 fois : fin
$0      lda  £1         ;1ère minute
        sta  a1         ;message d'alerte
        sta  depasst
        ldy  £0
$1      lda  declm,y
        beq  $2
        jsr  envoi
        iny
        bne  $1
$2      jsr  initcpt1
        jmp  deb4
```

Source PL.3
Assembleur ProCODE

```
*--------------------
* affiche
*--------------------

affiche jsr  chkcom
        lda  £0
        sta  $52
        jsr  frmevl     ;cherche chaine
        jsr  chkstr     ;c'est chaine ?
        lda  facmo
        sta  forpnt
        lda  facmo+1
        sta  forpnt+1
        ldy  £0
        lda  (forpnt),y ;récupère longueur
        tax
        ldy  £2
$0      lda  (forpnt),y ;récupère adresse
        dey             ;de la chaine
        sta  ptr,y
        cpy  £0
        bne  $0
$1      dex             ;affiche la chaine
        bmi  $fin
        lda  (ptr),y
        jsr  envoicar
        iny
        jmp  $1
$fin    jsr  chrgot     ;quel est le
        beq  $cr        ;caract suivant ?
        cmp  £$3B       ;point virgule ?
        beq  $fin1
        rts             ;-syntax error
$fin1   jmp  chrgot
$cr     lda  £10
        jsr  envoi
        lda  £13
        jmp  envoi

*--------------------
* alerte
*--------------------

alerte  jsr  chkcom
        lda  £0
        sta  $52
        jsr  frmevl     ;cherche chaine
        jsr  chkstr     ;c'est 1 chaine ?
        lda  facmo
        sta  forpnt
        lda  facmo+1
        sta  forpnt+1
        ldy  £0
        lda  (forpnt),y ;récupère longueur
        cmp  £37        ;maximum 36
        bcc  $000       ;caractères
        jmp  strtool
$000    sta  $lg
        ldy  £2
$0      lda  (forpnt),y ;récupère adresse
        dey             ;de la chaine
        sta  ptr,y
        cpy  £0
        bne  $0
        jsr  getbytc    ;attend variable%
        txa
        beq  $00
        jsr  bip
$00     ldy  £8         ;va en ligne 0
$01     lda  $ligne0,y  ;et l'efface
        jsr  envoi
        dey
        bpl  $01
        ldy  £0
        ldx  $lg
$1      dex             ;affiche la chaine
        bmi  $fin
        lda  (ptr),y
        jsr  envoicar
        iny
        jmp  $1
$fin    lda  £$0A
        jmp  envoi
$ligne0 dfb  $41,$40,$1F,$64
        dfb  $12,$20,$41,$40,$1F
$lg     ds   1

*--------------------
* appel
*--------------------

appel   jsr  chkcom     ;virgule ?
        jsr  ptrget     ;cherche variable
        sta  varpnt
        sty  varpnt+1
        lda  varnam     ;si var entière,
        and  varnam+1   ;les 2 octets sont
        bmi  $a1        ;négatifs
        jmp  miserr
$a1     ldy  £0         ;attend un instant
$a2     lda  $C061      ;avant de renoncer
        bmi  $sonne
        dey
        bne  $a2
$sonnepa lda £0         ;a=0 : sonne pas
$fin    ldy  £1
        sta  (varpnt),y
        dey
        tya
        sta  (varpnt),y
        rts
$sonne  ldy  £3         ;sonne-t-il au -
$s1     lda  £$C6       ;1/2 seconde ?
        jsr  $FCA8      ;(évite les
        bit  $C061      ;tintements)
        bpl  $sonnepa
        dey
        bne  $s1
        lda  £1
        bne  $fin

*--------------------
* bip
*--------------------

bip     lda  £7
        jmp  envoi

*--------------------
* cadre
*--------------------

cadre   jsr  getbytc    ;attend h1,
        cpx  £41
        bcs  $errf
        stx  $h1

        jsr  getbytc    ;v1,
        cpx  £25
        bcs  $errf
        stx  $v1

        jsr  getbytc    ;h2,
        cpx  £41
        bcs  $errf
        stx  $h2
        jsr  getbytc    ;v2,
        cpx  £25
        bcs  $errf
        stx  $v2
        jsr  getbytc    ;type,
        cpx  £3
        bcs  $errf
        stx  $type
        jsr  getbytc    ;lignage,
        stx  $lignage
        jsr  getbytc    ;couleur caract,
        cpx  £0
        bcs  $errf
        txa
        ora  £$40
        sta  $coulcar
        jsr  getbytc    ;et couleur fond.
        txa
        ora  £$50
        sta  $coulfd
        cmp  £$58
        bcc  $deb
$errf   jmp  horslim
$h1     ds   1
$h2     ds   1
$v1     ds   1
$v2     ds   1
$coulcar ds  1
$coulfd ds   1
$type   ds   1
```

```
$lignage ds   1
$deb     lda  $h2        ;h1 > h2 ?
         sec
         sbc  $h1
         cmp  £2
         bcc  $errf
         lda  $v2        ;v1 > v2 ?
         sec
         sbc  $v1
         cmp  £2
         bcc  $errf
         jsr  curseuri
         lda  £$59
         sta  lignag0
         lda  $lignage
         beq  $t00
         inc  lignag0
$t00     lda  $h1
         clc
         adc  £$40
         sta  hori
         lda  $type
         cmp  £2
         bne  $t01
         inc  hori
$t01     lda  $coulcar
         sta  coulc
         lda  $type
         cmp  £2
         beq  $type2
         lda  $coulfd
         sta  coulf
         lda  £$1B
         sta  coulc+1
         bne  $11
$type2   lda  £0         ;Si type 2, laisse
         sta  coulc+1    ;la couleur fond
         sta  coulf
$11      lda  $h2        ;calcule répétit.
         sec
         sbc  $h1
         adc  £$3D
         sta  repet
$t0      ldy  $type      ;caract ligne haut
         lda  lh,y
         sta  caract
         lda  ahg,y      ;angle haut gauche
         sta  angleg
         lda  ahd,y      ;angle haut droit
         sta  angled
         lda  graph1,y   ;type 0 & 1=graph
         sta  graph0     ;type 2 = texte
         lda  $v1        ;position vert.
         clc
         adc  £$40
         sta  vert
         jsr  ligne      ;trace lig du haut
         ldy  $type      ;caract ligne bas
         lda  lb,y
         sta  caract
         lda  abg,y      ;angle bas gauche
         sta  angleg
         lda  abd,y      ;angle bas droit
         sta  angled
         lda  $v2        ;position vert
         clc
         adc  £$40
         sta  vert
         jsr  ligne      ;trace lig du bas
         lda  £0         ;annule les angles
         sta  angleg
         sta  angled
         lda  $h1        ;colonne gauche :
         clc             ;position horiz
         adc  £$40
         sta  hori
         lda  $v1        ;position vert
         clc
         adc  £$40
         sta  vert
$cg      inc  vert       ;boucle de tracé
         ldy  $type      ;charge type carac
         lda  cg,y
         sta  caract
         jsr  colonne    ;trace la colonne
         lda  $v2
         clc
         adc  £$3F       ;colonn terminée ?
         cmp  vert
         bne  $cg
$fincolg lda  $v1        ;colonne droite
         clc             ;idem
         adc  £$40
         sta  vert
         lda  $h2
         clc
         adc  £$40
         sta  hori
$colod   inc  vert
         ldy  $type
         lda  cd,y
         sta  caract
         jsr  colonne
         lda  $v2
         clc
         adc  £$3F
         cmp  vert
         bne  $colod
$finc    lda  £$0F       ;mode texte
         jsr  envoi
         jsr  envoi1F
         clc
         lda  $v1
         adc  £$41
         jsr  envoi
         clc
         lda  $h1
         adc  £$41
         jsr  envoi
         jsr  envoi1B
         lda  coulc
         jsr  envoi
         rts

ligne    ldy  £0
$L1      lda  chaine,y
         jsr  envoi
         iny
         cpy  £15
         bne  $L1
         rts

colonne  ldy  £0
$L1      lda  chaine,y
         jsr  envoi
         iny
         cpy  £11
         bne  $L1
         lda  caract
         jmp  envoi

chaine   dfb  $1F
vert     dfb  $40
hori     dfb  $40
graph0   dfb  $0E,$1B
lignag0  dfb  $59,$1B
coulc    dfb  $40,$1B
coulf    dfb  $50
angleg   dfb  $00
caract   dfb  $00,$12
repet    dfb  $40
angled   dfb  $00
lh       dfb  $70,$23,$5F
lb       dfb  $23,$70,$7E
cg       dfb  $6A,$35,$7D
cd       dfb  $35,$6A,$7B
ahg      dfb  $60,$37,$00
ahd      dfb  $30,$6B,$00
abg      dfb  $22,$75,$00
abd      dfb  $21,$7A,$00
graph1   dfb  $0E,$0E,$0F

*--------------------
* chariot
*--------------------

chariot  jsr  chrgot
         beq  $1seul
         cmp  £','
         beq  $00
         jmp  snerr
$1seul   ldx  £1
         bne  $0
$00      jsr  getbytc
         cpx  £25
         bcs  $hlm
$0       dex
         bmi  $1
         lda  £10
         jsr  envoi
         lda  £13
         jsr  envoi
         jmp  $0
$1       rts
$hlm     jmp  horslim

*--------------------
* cnxn
*--------------------

cnxn     jsr  recoit
         ldy  £2
$1       lda  $2,y
         jsr  envoi
         dey
         bpl  $1
         jsr  recoit
         jmp  recoit1

$2       dfb  104,57,27

*--------------------
* cnxr
*--------------------

cnxr     ldy  £5         ;envoie les carac
$a       lda  $2,y       ;de connexion
         jsr  envoi      ;retournée
         dey
         bpl  $a
         jsr  chkcom     ;virgule ?
         jsr  ptrget     ;cherche variable
         sta  varpnt     ;entière
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $a1
         jmp  miserr
$a1      bit  pint
         bpl  $a2
         lda  £15
         bne  $a3
$a2      lda  £60        ;60 essais = 1 mn
$a3      sta  $essai
$ok      jsr  recoit1    ;attend la réponse
         bcs  $1         ;carac reçu
         dec  $essai
         lda  $essai
         bne  $ok        ;60 tentatives
         beq  $reponse   ;on a échoué
$1       cmp  £$13       ;reçu $13 ?
         bne  $ok        ;non, on boucle
$11      jsr  recoit1    ;oui, attend un
         cmp  £$13       ;caractère. $13?
         beq  $11
         cmp  £$53       ;=$53 ? ou
         beq  $110
         cmp  £$51       ;= $51
         bne  $ok        ;sinon on boucle
$110     lda  £1         ;drap = 1 : ok
$reponse ldy  £1         ;met la réponse
         pha             ;dans var%
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         pla
         beq  $echoue
         jsr  wait
         ldy  £0
$bcl     lda  $copyr,y
         beq  $finc
         eor  £$A5
         jsr  envoi
         iny
         bne  $bcl
$finc    jsr  wait       ;attend 1 instant
         jsr  wait
         jsr  wait
         jsr  wait
         lda  £12        ;efface l'écran et
         jmp  envoi      ;sort
$echoue  jmp  decnx      ;decnx et sort
$essai   ds   1
$2       dfb  104,57,27,111,57,27
$copyr   DFB  $BA,$E5,$E4,$85,$B7,$C1,$BA
         DFB  $E5,$E4,$BE,$9C,$DA,$A9
         DFB  $BA,$EF,$ED,$EB,$CA,$DC,$C4
         DFB  $D0,$85,$D6,$C0,$D7,$D3
         DFB  $C0,$D0,$D7,$85,$F5,$CA,$C8
         DFB  $FA,$E9,$CC,$CB,$CE,$85
         DFB  $97,$8B,$94,$BA,$EE,$ED,$D5
         DFB  $CA,$D0,$D7,$85,$E4,$D5
         DFB  $D5,$C9,$C0,$85,$EC,$EC,$85
         DFB  $C0,$D1,$85,$E8,$C4,$C6
         DFB  $CC,$CB,$D1,$CA,$D6,$CD,$BA
         DFB  $E9,$EC,$8D,$C6,$8C,$85
         DFB  $94,$9C,$9D,$9D,$85,$E6,$F5
         DFB  $89,$85,$EF,$E9,$E7,$85
         DFB  $83,$85,$F5,$CA,$C8,$82,$D6
         DFB  $BA,$E8,$F6,$FA,$FA,$FA
         DFB  $BA,$EA,$E8,$8D,$94,$8C,$85
         DFB  $96,$9C,$8B,$90,$94,$8B
         DFB  $97,$91,$8B,$91,$96,$00

*--------------------
* curseurvi
*--------------------

curseurv lda  £$11
         jmp  envoi

*--------------------
* curseurinvi
*--------------------

curseuri lda  £$14
         jmp  envoi
```

```
*--------------------
* decalebas
*--------------------

decaleb  jsr  getbytc
         cpx  £25
         bcc  $ok
         jmp  horslim
$ok      ldy  £11
$0       lda  $dcb,y
         jsr  envoi
         dey
         bpl  $0
$1       dex
         bmi  $fin
         lda  £$0B
         jsr  envoi
         jmp  $1
$fin     ldy  £7
$2       lda  $page,y
         jsr  envoi
         dey
         bpl  $2
         rts
$dcb     dfb  $43,$69,$3A,$1B,4,$66,$3A,$1B
         dfb  $41,$41,$1F,$14
$page    dfb  $43,$6A,$3A,$1B,4,$66,$3A,$1B

*--------------------
* decalehaut
*--------------------

decaleh  jsr  getbytc
         cpx  £25
         bcc  $ok
         jmp  horslim
$ok      ldy  £11
$0       lda  $dch,y
         jsr  envoi
         dey
         bpl  $0
$1       dex
         bmi  $fin
         lda  £$0A
         jsr  envoi
         jmp  $1
$fin     ldy  £7
$2       lda  $page,y
         jsr  envoi
         dey
         bpl  $2
         rts
$dch     dfb  $43,$69,$3A,$1B,4,$66,$3A,$1B
         dfb  $41,$58,$1F,$14
$page    dfb  $43,$6A,$3A,$1B,4,$66,$3A,$1B

*--------------------
* decnx
*--------------------

decnx    jsr  recoit
         ldy  £2
$1       lda  $2,y
         jsr  envoi
         dey
         bpl  $1
         jsr  recoit1
         jsr  recoit
         jmp  recoit
$2       dfb  103,57,27

*--------------------
*  deplace
*--------------------

deplace  jsr  getbytc
         txa
         beq  $err
         clc
         adc  £$40
         sta  posh
         cmp  £$69
         bcs  $err
         jsr  getbytc
         txa
         beq  $err
         clc
         adc  £$40
         sta  posv
         cmp  £$59
         bcs  $err
         lda  £$1F
         jsr  envoi
         lda  posv
         jsr  envoi
         lda  posh
         jmp  envoi
$err     jmp  horslim

*--------------------
* enligne
*--------------------

enligne  jsr  chkcom
         jsr  ptrget
         sta  varpnt
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $ok
         jmp  miserr
         lda  £6
         sta  $essai
$ok      ldy  £5
$ok1     lda  $chaine,y
         jsr  envoi
         dey
         bpl  $ok1
$0       jsr  recoit1
         bcs  $1
         dec  $essai
         bit  $essai
         bne  $ok
         lda  £0
         beq  $reponse
$1       cmp  £$1F
         bne  $0
         jsr  recoit1
         cmp  £$3F
         bcc  $ok
         jsr  recoit1
         cmp  £$3F
         bcc  $ok
         lda  £1
$reponse ldy  £1
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         rts
$essai   ds   1
$chaine  hex  611B02663A1B

*--------------------
* fixe
*--------------------

fixe     jsr  envoi1B
         lda  £$49
         jmp  envoi

*--------------------
* flash
*--------------------

flash    jsr  envoi1B
         lda  £$48
         jmp  envoi

*--------------------
* gr
*--------------------

gr       lda  £$0E
         jmp  envoi
```

Source PL.4
Assembleur ProCODE

```
*--------------------
* ident
*--------------------

ident    lda  £3          ;cherche 3 fois
         sta  $essai      ;la réponse
$debut   ldy  £0
$d       lda  $interr,y
         beq  $0
         jsr  envoi
         iny
         bne  $d
$0       jsr  recoit1
         bcs  $1
         dec  $essai
         lda  $essai
         bne  $0
         sta  $reponse
         jmp  $fin
$interr  dfb  $1B,$3A,$66,$3,27,57,$7B,0
$1       cmp  £$1         ;est-ce le 1
         bne  $0
         jsr  recoit1     ;oui, attend 1er
         bcc  $debut      ;octet
         sta  $oc1
         jsr  recoit1     ;attend 2ème octet
         bcc  $debut
         sta  $oc2
         jsr  recoit1     ;attend 3ème octet
         bcc  $debut
         sta  $oc3
         jsr  recoit1     ;attend le 4 final
         bcs  $11
         jmp  $debut
$11      cmp  £4
         beq  $12
         jmp  $debut
$12      lda  £1
         sta  $reponse
         bne  $fin
$essai   ds   1
$oc1     ds   1
$oc2     ds   1
$oc3     ds   1
$reponse ds   1
$fin     jsr  chkcom
         jsr  ptrget
         sta  varpnt
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $2
         jmp  miserr
$2       lda  $oc1
         ldy  £1
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         jsr  chkcom
         jsr  ptrget
         sta  varpnt
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $3
         jmp  miserr
$3       lda  $oc2
         ldy  £1
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         jsr  chkcom
         jsr  ptrget
         sta  varpnt
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $4
         jmp  miserr
$4       lda  $oc3
         ldy  £1
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         jsr  chkcom
         jsr  ptrget
         sta  varpnt
         sty  varpnt+1
         lda  varnam
         and  varnam+1
         bmi  $5
         jmp  miserr
$5       lda  $reponse
         ldy  £1
         sta  (varpnt),y
         dey
         tya
         sta  (varpnt),y
         rts

*--------------------
* init
*--------------------

init     jsr  getbytc     ;attend n° de port
         stx  noslot
         txa
         ora  £$C0
         sta  $5+2
         sta  $4+2
         sta  $1+2
$1       lda  $C00C
         cmp  £$31        ;est-ce un port
         bne  $err        ;série ?
         lda  £$C0
         sta  status+1
         sta  data+1
         lda  noslot
         asl
         asl
         asl
```

```
        asl
        adc  £$88
        sta  data
        sta  status
        inc  status
        jsr  recoit
        jsr  recoit
        lda  deuxc
        beq  $2
        jmp  conf
$2      lda  £7
$4      jsr  $C000
$5      lda  $C000
        cmp  £$2C
        beq  $6
        jsr  portint
$6      rts
$err    lda  £3
        jmp  errout

conf    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
        jsr  $C311
        ldy  £1
$1      lda  $BE30,y
        sta  $av,y
        dey
        bpl  $1
        lda  £00
        sta  $BE30
        lda  £$C2
        sta  $BE31
        lda  £7
        jsr  cout
        ldy  £1
$2      lda  $av,y
        sta  $BE30,y
        dey
        bpl  $2
        rts
$av     ds   2
portint lda  £$FF
        sta  pint
        lda  noslot
        ora  £$C0
        sta  cn
        sta  pi1+1
        sta  piinit+1
        sta  piread+1
        sta  piwrite+1
        sta  pistat+1
        sta  $1+2
        sta  $r+2
        sta  $w+2
        sta  $s+2
$1      lda  $C00D
        sta  piinit
$r      lda  $C00E
        sta  piread
$w      lda  $C00F
        sta  piwrite
$s      lda  $C010
        sta  pistat
        lda  noslot
        asl
        asl
        asl
        asl
        sta  n0
        rts
pi      jmp  (pi1)
initpi  jmp  (piinit)
readpi  jmp  (piread)
writepi jmp  (piwrite)
statpi  jmp  (pistat)

pi1     da   $0000
piinit  da   $0000
piread  da   $0000
piwrite da   $0000
pistat  da   $0000

*--------------------
* inverse
*--------------------

inverse jsr  envoi1B
        lda  £$5D
        jmp  envoi

*--------------------
* ligne
*--------------------

xligne  jsr  envoi1B
        lda  £$5A
        jmp  envoi

*--------------------
* localise
*--------------------

localise jsr  locali
        jmp  losui

locali  lda  £3          ;cherche 3 fois
        sta  $essai      ;la réponse
$debut  ldy  £0          ;envoi
$00     lda  $chaine,y   ;transparence
        beq  $0          ;+ $1B,$61
        jsr  envoi       ;pour position
        iny              ;curseur
        bne  $00
$0      jsr  recoit1
        bcs  $1
        dec  $essai
        lda  $essai
        bne  $0
        sta  lreponse
        jmp  $fin
$1      cmp  £$1F        ;est-ce 1F ?
        bne  $0
        jsr  recoit1     ;oui, attend vert
        cmp  £$3F        ;supérieur à $40
        bcc  $debut      ;alors c'est bon
        sbc  £$40        ;soustrait $40
        sta  lver
        jsr  recoit1     ;attend horizontal
        cmp  £$3F
        bcc  $debut
        sbc  £$40
        sta  lhor
        lda  £1
        sta  lreponse
$fin    rts
$essai  ds   1
$chaine dfb  $1B,$3A,$66,$2,$1B,$61,0
lhor    ds   1
lver    ds   1
lreponse ds  1

losui   jsr  chkcom
        jsr  ptrget
        sta  varpnt
        sty  varpnt+1
        lda  varnam
        and  varnam+1
        bmi  $2
        jmp  miserr
$2      lda  lhor
        ldy  £1
        sta  (varpnt),y
        dey
        tya
        sta  (varpnt),y
        jsr  chkcom
        jsr  ptrget
        sta  varpnt
        sty  varpnt+1
        lda  varnam
        and  varnam+1
        bmi  $3
        jmp  miserr
$3      lda  lver
        ldy  £1
        sta  (varpnt),y
        dey
        tya
        sta  (varpnt),y
        jsr  chkcom
        jsr  ptrget
        sta  varpnt
        sty  varpnt+1
        lda  varnam
        and  varnam+1
        bmi  $4
        jmp  miserr
$4      lda  lreponse
        ldy  £1
        sta  (varpnt),y
        dey
        tya
        sta  (varpnt),y
        rts

*--------------------
* mode
*--------------------

mode    jsr  getbytc
        cpx  £4
        bcc  $1
        jmp  horslim
$1      stx  $2
        jsr  envoi1B
        clc
        lda  £$4C
        adc  $2
        jmp  envoi
$2      ds   1

*--------------------
* mth
*--------------------

mth     jsr  getbytc
        cpx  £0
        beq  $litt
        ldx  £$FF        ;mode mathématique
$litt   stx  drap
        rts

*--------------------
* normal
*--------------------

normal  jsr  envoi1B
        lda  £$5C
        jmp  envoi

*--------------------
* paramg
*--------------------

paramg  jsr  getbytc
        stx  $coulc
        jsr  getbytc
        stx  $coulf
        jsr  getbytc
        stx  $ligne
        jsr  getbytc
        stx  $flash
        lda  $coulc
        ora  $coulf
        cmp  £8
        bcc  $0
        jmp  horslim
$0      lda  £$0E        ;graphique
        jsr  envoi
        jsr  envoi1B     ;couleur carac
        clc
        lda  $coulc
        adc  £$40
        jsr  envoi
        jsr  envoi1B     ;couleur fond
        clc
        lda  $coulf
        adc  £$50
        jsr  envoi
        jsr  envoi1B     ;flash
        lda  £$49
        bit  $flash
        beq  $1
        sec
        sbc  £1
$1      jsr  envoi
        jsr  envoi1B     ;lignage
        lda  £$59
        ldx  $ligne
        beq  $2
        clc
        adc  £1
$2      jmp  envoi
$coulc  ds   1
$coulf  ds   1
$ligne  ds   1
$flash  ds   1

*--------------------
* paramt
*--------------------

paramt  jsr  getbytc
        stx  $mode
        jsr  getbytc
        stx  $coulc
        jsr  getbytc
        stx  $coulf
        jsr  getbytc
        stx  $flash
        jsr  getbytc
        stx  $inv
        jsr  getbytc
```

```
        stx  $souli
        lda  $mode      ;mode < 4
        cmp  £4
        bcc  $ok
$err    jmp  horslim
$ok     lda  $coulc     ;couleur carac et
        ora  $coulf     ;fond < 8
        cmp  £8
        bcs  $err
        lda  £$0F       ;mode texte
        jsr  envoi
        jsr  envoi1B    ;mode
        clc
        lda  £$4C
        adc  $mode
        jsr  envoi
        jsr  envoi1B    ;couleur carac
        clc
        lda  £$40
        adc  $coulc
        jsr  envoi
        jsr  envoi1B    ;couleur fond
        clc
        lda  £$50
        adc  $coulf
        jsr  envoi
        jsr  envoi1B    ;flash
        lda  £$49
        ldx  $flash
        beq  $1
        sec
        sbc  £1
$1      jsr  envoi
        jsr  envoi1B    ;inv
        lda  £$5C
        ldx  $inv
        beq  $2
        clc
        adc  £1
$2      jsr  envoi
        jsr  envoi1B    ;lignage
        lda  £$59
        ldx  $souli
        beq  $3
        clc
        adc  £1
$3      jmp  envoi
$mode   ds   1
$coulc  ds   1
$coulf  ds   1
$flash  ds   1
$inv    ds   1
$souli  ds   1

*--------------------
* reinit
*--------------------

reinit  ldy  £0
$1      lda  $2,y
        beq  $fin
        jsr  envoi
        iny
        bne  $1
$fin    rts

$2      dfb  27,57,$7F,$1B,$5C,$1B,$49
        dfb  $1B,$59,$1B,$4C,$1B,$47
        dfb  $1B,$50,$20,0

*--------------------
* rouleau
*--------------------

rouleau jsr  recoit
        jsr  getbytc
        cpx  £0
        beq  $off
$on     ldy  £7
$0      lda  $on1,y
        jsr  envoi
        dey
        bpl  $0
        rts
$off    ldy  £7
$1      lda  $off1,y
        jsr  envoi
        dey
        bpl  $1
        rts
$on1    dfb  $43,$69,$3A,$1B,4,$66,$3A,$1B
$off1   dfb  $43,$6A,$3A,$1B,4,$66,$3A,$1B

*--------------------
* saisie
*--------------------

saisie  jsr  chkcom
        jsr  ptrget     ;attend nom de
        bit  varnam     ;variable
        bmi  $err
        bit  varnam+1
        bmi  $1
$err    jmp  miserr
$1      jsr  sais
        jmp  finsais
sais    lda  £0
        sta  lg

        jsr  recoit
        bcc  pasrecu

reçu    cmp  £$13       ;fonction ?
        bne  $0
        jmp  control
$0      cmp  £$16       ;un caract spécial
        beq  spec       ;ou accentué ?
        cmp  £$19
        beq  spec

stocke  and  £$7F       ;caractère
        sta  $200       ;stocké en $200
        inc  lg         ;reçu 1 carac
pasrecu lda  lg
        rts

finsais lda  $200
        cmp  £7
        bne  $00
        lda  £9
        sta  $200
        bne  $1
$00     bcc  $1
        cmp  £10
        bcs  $1
        dec  $200
$1      jsr  getspa     ;cherche place
        ldx  £0         ;pour 1 carac qui
        ldy  £2         ;se trouve en $200
        jsr  movstr     ;range la chaine
        lda  lg
        ldy  £0
        sta  (varpnt),y
        iny
        lda  fretop
        sta  (varpnt),y
        iny
        lda  fretop+1
        sta  (varpnt),y
        rts
lg      ds   1
spec    jsr  recoit1    ;attend un 2ème
        bcs  $0         ;caractère
        jmp  pasrecu    ;pas reçu
$0      tax
        and  £$F0       ;si > à $40,
        cmp  £$40       ;c'est accentué
        beq  accents    ;sinon, c'est
        txa             ;un carac spécial
        ldy  £0         ;qu'on recode
$1      cmp  act1,y     ;avec la table
        beq  $2
        iny
        iny
        cpy  £34
        bne  $1
        jmp  pasrecu    ;pas dans la table
$2      iny
        lda  act1,y     ;est dans la table
        jmp  stocke

accents stx  $c1        ;sauve le 2è carac
        jsr  recoit1    ;en attend un 3ème
        bcs  $0
        jmp  pasrecu    ;pas reçu
$0      sta  $c2        ;compare le 2ème
        ldy  £0         ;dans la table
$1      lda  $c1
        cmp  act2,y
        bne  $2
        iny             ;compare le 3ème
        lda  $c2
        cmp  act2,y
        bne  $3
        iny
        lda  act2,y
        jmp  stocke     ;trouvé
$2      iny
$3      iny
        iny
        cpy  £42
        bne  $1
        jmp  pasrecu    ;pas trouvé
$c1     ds   1
$c2     ds   1

control jsr  recoit1    ;attend 2ème carac
        bcs  $1
$2      jmp  pasrecu
$1      cmp  £$13
        beq  control
        bit  £$40       ;doit etre entre
        beq  $2         ;$41 et $49
        cmp  £$4A
        bcs  $2
$3      and  £$0F       ;carac entre 1 & 9
        jmp  stocke

*--------------------
* sansligne
*--------------------

sansligne jsr envoi1B
        lda  £$59
        jmp  envoi

*--------------------
* secret
*--------------------

secret  lda  £$FF
        sta  dsecret
        jmp  accept1

dsecret ds   1

*--------------------
* servoff
*--------------------

serveuro ldy  £2
$1      lda  vecteur,y
        sta  $3F5,y
        dey
        bpl  $1
        rts

*--------------------
* txt
*--------------------

txt     lda  £$0F
        jmp  envoi

*--------------------
* videcran
*--------------------

videcran jsr  chrgot    ;si 0 paramètre,
        beq  $noi       ;c'est noir
        cmp  £','
        beq  $coulsui
        jmp  snerr
$noi    ldx  £0
        beq  $coulnoi
$coulsui jsr  getbytc   ;n° de couleur
$coulnoi lda  £$41      ;réinit position
        sta  $pos+1
        txa
        beq  $noir
        cmp  £8
        bcc  $00
$err    jmp  horslim
$00     jsr  curseuri   ;curseur invisible
        txa
        adc  £$40
        sta  $coul
        adc  £$10
        sta  $coul+2
$0      ldy  £12        ;efface une ligne
$1      lda  $eff,y
        jsr  envoi
        dey
        bpl  $1
        inc  $pos+1
        lda  $pos+1
        cmp  £$59       ;24 lignes en tout
        bne  $0
        jsr  ligne0
        rts
$noir   lda  £12
        jsr  envoi
        jsr  curseuri
        jmp  ligne0
$eff    dfb  $0F,103,$12,$5F
$coul   dfb  $40,$1B,$50,$1B,$0E
$pos    dfb  $41,$40,$1F,$14

ligne0  ldy  £6         ;va en ligne 0
$01     lda  $ligne0,y  ;et l'efface
        jsr  envoi
        dey
        bpl  $01
        rts
$ligne0 dfb  $0A,$64,$12,$20,$41,$40,$1F

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

```
* vidligne
*--------------------
vidligne jsr  getbytc    ;attend n° ligne
         txa             ;de 1 à 24.
         beq  $err
         cmp  £25
         bcs  $err
         adc  £$40
         sta  $pos+1
         jsr  getbytc    ;couleur de 0 à 7
         jsr  curseuri   ;curseur invisible
         txa
         cmp  £8
         bcc  $1
$err     jmp  horslim
$1       adc  £$40
         sta  $coul
         adc  £$10
         sta  $coul+2
         ldy  £12
$1       lda  $eff,y
         jsr  envoi
         dey
         bpl  $1
         rts
$eff     dfb  $0F,103,$12,$5F
$coul    dfb  $40,$1B,$50,$1B,$0E
$pos     dfb  $41,$40,$1F,$14

*--------------------
* vidrect
*--------------------
vidrect  jsr  getbytc    ;attend les 5
         cpx  £41        ;paramètres
         bcs  $errf
         stx  $h1
         jsr  getbytc
         cpx  £25
         bcs  $errf
         stx  $v1
         jsr  getbytc
         cpx  £41
         bcs  $errf
         stx  $h2
         jsr  getbytc
         cpx  £25
         bcs  $errf
         stx  $v2
         jsr  getbytc
         stx  $coul
         cpx  £8         ;couleur > 8 ?
         bcc  $deb
$errf    jmp  horslim
$deb     lda  $h1        ;h1 > h2 ?
         cmp  $h2
         bcs  $errf
         lda  $v1        ;v1 > v2
         cmp  $v2
         bcs  $errf
         jsr  curseuri
$bcl     jsr  envoi1F    ;envoi 1F
         lda  £$40       ;$40 + v1
         clc
         adc  $v1
         jsr  envoi
         lda  £$40       ;$40 + h1
         clc
         adc  $h1
         jsr  envoi
         lda  £$E
         jsr  envoi
         jsr  envoi1B
         lda  £$40
         clc
         adc  $coul
         pha
         jsr  envoi
         jsr  envoi1B
         pla
         adc  £$10
         jsr  envoi
         lda  £$5F       ;blanc
         jsr  envoi
         lda  £$12       ;répétition
         jsr  envoi
         lda  $h2        ;x fois
         sec
         sbc  $h1
         clc
         adc  £$40
         jsr  envoi
         inc  $v1        ;boucle
         lda  $v1
         cmp  $v2
         bcc  $bcl
         beq  $bcl
         lda  £$F        ;passe en texte et
         jmp  envoi      ;retour
$h1      ds   1
$h2      ds   1
$v1      ds   1
$v2      ds   1
$coul    ds   1
```

Source PL.SP Assembleur ProCODE

```
*--------------------
* range error
*--------------------
horslim  lda  £2         ;illegal quantity
         jmp  errout     ;error

*--------------------
* chaine trop longue
*--------------------
strtool  ldx  £$B0       ;string too long
         jmp  error

*--------------------
* envoi un caractère
*--------------------
envoicar jsr  savereg
         sta  carac
         jsr  recoit
         ldy  £0         ;est-ce un carac
$bcl     bit  drap       ;accentué ?
         bpl  $L
$M       lda  accent,y   ;mathématique
         jmp  $debut
$L       lda  accent0,y  ;littéraire
$debut   beq  $0
         cmp  carac
         beq  $acc
         iny
         iny
         iny
         iny
         bne  $bcl
$0       lda  carac      ;non
         jsr  envoi
$1       jmp  restreg
$acc     iny             ;oui
         ldx  £3
$acc1    bit  drap
         bpl  $L1
         lda  accent,y   ;envoi des 3 carac
         jmp  $suite
$L1      lda  accent0,y
$suite   jsr  envoi
         iny
         dex
         bne  $acc1
         beq  $1

envoi    bit  pint
         bmi  envoigs
         sta  $a
         stx  $x
$1       ldx  £0
         lda  (status,x)
         and  £$10
         beq  $1
         lda  $a
         sta  (data,x)
         ldx  $x
         jsr  ace
         rts
$a       ds   1
$x       ds   1

envoigs  pha
         stx  $x
         sty  $y
$attend  jsr  parapi
         lda  £0
         jsr  statpi
         bcc  $attend
         jsr  parapi
         pla
         jsr  writepi
         ldx  $x
         ldy  $y
         jsr  ace
         rts
$x       ds   1
$y       ds   1
carac    ds   1

*--------------------
* envoi 1F
*--------------------
envoi1F  lda  £$1F
         jmp  envoi

*--------------------
* envoi 1B
*--------------------
envoi1B  lda  £$1B
         jmp  envoi

*--------------------
* recevoir un caractère
* sec si reçu, clc sinon
*--------------------
recoit   bit  pint
         bmi  recoitgs
         stx  regx
         ldx  £0
         lda  (status,x)
         and  £%00001000
         bne  $1
         ldx  regx
         clc
         rts
$1       lda  (data,x)
         and  £%01111111
         ldx  regx
         sec
         jsr  acr
         rts

recoitgs stx  $x
         sty  $y
         jsr  parapi
         lda  £1
         jsr  statpi
         bcc  $fin
         jsr  parapi
         jsr  readpi
         and  £$7F
         jsr  acr
$fin     ldx  $x
         ldy  $y
         rts
$x       ds   1
$y       ds   1

parapi   ldx  CN
         ldy  n0
         rts

*--------------------
* reçoit un caract
*  sec si reçu
*  clc si pas reçu avant x tps
*--------------------
recoit1  lda  £$C0
         sta  strec
         sta  strec+1
$1       jsr  recoit
         bcc  $2
         rts
$2       inc  strec
         bne  $1
         inc  strec+1
         bne  $1
         rts

strec    ds   2

*--------------------
* attendre une seconde
*--------------------
wait     sty  regy
         stx  regx
         pha
         ldy  £10
$1       lda  £$C6
         jsr  $FCA8
         lda  kbd
         cmp  £$9B
         beq  $2
         dey
         bne  $1
$2       pla
         ldy  regy
         ldx  regx
         rts

*--------------------
* registres
*--------------------
```

```
savereg  sta  rega
         stx  regx
         sty  regy
         rts
restreg  lda  rega
         ldx  regx
         ldy  regy
         rts
ace      rts
acr      rts
```

Source PL.DATA
Assembleur ProCODE

```
accent0  dfb  '¯',$00,$19,$48
         dfb  $5E,$00,$19,$43;"¯"
         dfb  '$',$19,$27,$00
         dfb  '£',$19,$23,$00
         dfb  '°',$19,$30,$00
$0       dfb  'è',$19,$41,$65
         dfb  'ê',$19,$42,$65
         dfb  'à',$19,$41,$61
         dfb  'ù',$19,$41,$75
         dfb  'ç',$19,$4B,$63
accent   dfb  0
act1     dfb  $23,"£      ;table pour le
         dfb  $24,"$      ;recodage des
         dfb  $26,"£      ;caractères
         dfb  $27,"$      ;spéciaux
         dfb  $2C,"<
         dfb  $2D,"¯"
         dfb  $2E,">
         dfb  $2F,"v
         dfb  $30,"°
         dfb  $31,"+
         dfb  $38,"/
         dfb  $3C,"/
         dfb  $3D,"/
         dfb  $3E,"/
         dfb  $6A,"*
         dfb  $7A,"*
         dfb  $7B,"B
         dfb  0
act2     dfb  $41,$65,"è
         dfb  $42,$65,"é
         dfb  $43,$65,"e
         dfb  $48,$65,"e
         dfb  $41,$61,"à
         dfb  $43,$61,"a
         dfb  $48,$61,"a
         dfb  $43,$69,"i
         dfb  $48,$69,"i
         dfb  $43,$6F,"o
         dfb  $48,$6F,"o
         dfb  $41,$75,"ù
         dfb  $48,$75,"u
         dfb  $4B,$63,"ç
         dfb  0
adrcom   da   accept
         da   acceptf
         da   affiche
         da   alerte
         da   appel
         da   bip
         da   cadre
         da   chariot
         da   cnxn
         da   cnxr
         da   curseurv
         da   curseuri
         da   decaleb
         da   decaleh
         da   decnx
         da   deplace
         da   enligne
         da   fixe
         da   flash
         da   gr
         da   ident
         da   init
         da   inverse
         da   xligne
         da   localise
         da   mode
         da   mth
         da   normal
         da   paramg
         da   paramt
         da   reinit
         da   rouleau
         da   saisie
         da   sansligne
         da   secret
         da   serveuro
         da   txt
         da   videcran
         da   vidligne
         da   vidrect
ancvect  ds   2        ;vecteur précédent
deuxc    ds   1        ;=$FF si //c
controle dfb  $41,$4D  ;envoi (suit $13)
         dfb  $42,$52  ;retour
         dfb  $43,$53  ;répétition
         dfb  $44,$6D  ;guide
         dfb  $45,$51  ;annulation
         dfb  $46,$50  ;sommaire
         dfb  $47,$6C  ;correction
         dfb  $48,$6E  ;suite
         dfb  0
commande str  "accept"    ;1
         str  "acceptf"   ;2
         str  "affiche"   ;3
         str  "alerte"    ;4
         str  "appel"     ;5
         str  "bip"       ;6
         str  "cadre"     ;7
         str  "chariot    ;8
         str  "cnxn"      ;9
         str  "cnxr"      ;10
         str  "curvis"    ;11
         str  "curinvis"  ;12
         str  "bas"       ;13
         str  "haut"      ;14
         str  "decnx"     ;15
         str  "deplace"   ;16
         str  "enligne"   ;17
         str  "fixe"      ;18
         dfb  1,$BF       ;flash  $9F or $A0
         dfb  1,$A8       ;gr     $88 or $A0
         str  "ident      ;21
         str  "init"      ;22
         dfb  1,$BE       ;invers $9E or $A0
         str  "ligne"     ;24
         str  "localise"  ;25
         str  "mode"      ;26
         str  "mth"       ;27
         dfb  1,$BD       ;normal $9D or $A0
         str  "paramg"    ;29
         str  "paramt"    ;30
         str  "reinit"    ;31
         str  "rouleau"   ;32
         str  "saisie"    ;33
         str  "sansligne";34
         str  "secret"    ;35
         str  "servoff"   ;36
         str  "txt"       ;37
         str  "videcran"  ;38
         str  "vidligne"  ;39
         str  "vidrect"   ;40
         dfb  0
cmd      ds   13          ;contiend la cmd
cn       ds   1           ;=$C0+slot
n0       ds   1           ;=slot*$10
drap     ds   1           ;pour mode math
nocar    ds   1           ;n° de caractère
nocmd    ds   1           ;n° de la commande
noslot   ds   1           ;n° du port série
pint     dfb  0           ;=$FF sur IIgs
posh     ds   1           ;position horiz
posv     ds   1           ;position vert
ptrcmd   ds   1           ;pointeur sur cmd
rega     ds   1
regx     ds   1
regy     ds   1
vecteur  ds   3           ;ancien vecteur &
```

Récapitulation POM.LINK.2.1

Après avoir saisi cette récapitulation sous moniteur, vous la sauvegarderez par :

```
BSAVE POM.LINK.2.1,A$801,L5104
```

```
0801:AD F6 03 C9 3E D0 07 AD F7 03 C9 08 F0 1A A0
0810:02 B9 F5 03 99 ED 1B 88 10 F7 A9 4C 8D F5 03 A9
0820:3E 8D F6 03 A9 08 8D F7 03 A9 00 8D C6 1A AD B3
0830:FB C9 06 D0 08 AD C0 FB D0 03 CE C6 1A 60 A0 00
0840:8C D3 1B 8C E4 1B EE D3 1B AC D3 1B C0 0C D0 03
0850:4C C9 DE 09 A0 99 D3 1B 20 B1 00 F0 04 C9 2C D0
0860:E5 A0 FF 8C E9 1B EE E9 1B AC E9 1B B9 D8 1A CD
0870:D3 1B D0 18 A2 00 E8 EE E9 1B AC E9 1B B9 D8 1A
0880:DD D3 1B D0 07 EC D3 1B F0 15 D0 EA EE E9 1B AC
0890:E9 1B B9 D8 1A F0 2D 30 F3 EE E4 1B 4C 6F 08 A9
08A0:74 8D C2 08 A9 1A 8D C3 08 AC E4 1B F0 13 EE C2
08B0:08 D0 03 EE C3 08 EE C2 08 D0 03 EE C3 08 88 D0
08C0:ED 6C 9E 1A 4C C9 DE A9 00 8D 51 17 A9 FF 8D E2
08D0:1B 20 F5 E6 8E 82 14 20 F5 E6 8E 83 14 E0 19 90
08E0:03 4C BD 18 20 F5 E6 E0 02 90 03 4C BD 18 8E 47
08F0:0B AC 83 14 AE 82 14 20 48 0B 20 FE 10 20 BE DE
0900:A9 00 85 52 20 7B DD 20 6C DD A5 A0 85 85 A5 A1
0910:85 86 A0 00 B1 85 AA 18 6D 82 14 8D 82 14 A0 02
0920:B1 85 88 99 08 00 C0 00 D0 F6 CA 30 09 B1 08 20
0930:C7 18 C8 4C 2A 09 20 F5 E6 8E 3D 0B E0 00 F0 04
0940:E0 29 90 03 4C BD 18 AD 82 14 18 6D 3D 0B C9 29
0950:90 03 4C BD 18 4C D6 0B A9 00 8D 45 0B 8D 44 0B
0960:EE 45 0B 20 05 0B 20 F9 10 EE 41 0B D0 0D EE 42
0970:0B D0 08 EE 43 0B D0 03 4C 0A 0A 20 02 0B AD CC
0980:16 F0 E6 AD 00 02 29 7F 8D 46 0B 20 05 0B AD 45
0990:0B F0 03 20 21 0B AD 46 0B C9 0D F0 04 C9 0A B0
09A0:03 4C 59 0A C9 20 90 BB AD 40 0B CD 3D 0B D0 03
09B0:4C D8 09 EE 40 0B AC 40 0B AD 46 0B C9 2C D0 02
09C0:A9 2E C9 3A D0 02 A9 2E 99 00 02 2C 51 17 10 02
09D0:A9 2A 20 10 19 4C 63 09 A9 01 8D 45 0B 20 FE 10
09E0:A0 00 B9 F3 09 F0 06 20 10 19 C8 D0 F5 20 F9 10
09F0:4C 63 09 07 1F 40 41 20 12 64 1F 40 41 46 69 6E
0A00:20 64 65 20 7A 6F 6E 65 0A 00 AD 44 0B F0 03 4C
0A10:75 0B A9 01 8D 45 0B 8D 44 0B 20 FE 10 A0 00 B9
0A20:33 0A F0 06 20 10 19 C8 D0 F5 20 F9 10 20 0A 0B
0A30:4C 66 09 07 1F 40 41 20 12 64 1F 40 41 44 19 42
0A40:65 63 6F 6E 6E 65 78 69 6F 6E 20 69 6D 6D 69 6E
0A50:65 6E 74 65 2E 2E 2E 0A 00 20 05 0B AD 46 0B C9
0A60:0D D0 05 A9 01 8D 46 0B C9 07 F0 07 C9 05 F0 25
0A70:4C 7A 0B AD 40 0B F0 51 AD 82 14 18 6D 40 0B AA
0A80:CA AC 83 14 20 48 0B A9 2E 20 10 19 20 48 0B CE
0A90:40 0B 4C 63 09 AE 82 14 AC 83 14 20 48 0B 20 FE
0AA0:10 AE 3D 0B CA A9 AE 20 10 19 CA 10 F8 AE 82 14
0AB0:AC 83 14 20 48 0B 20 F9 10 AD 40 0B D0 03 4C 7A
0AC0:0B A9 00 8D 40 0B 4C 58 09 A0 00 B9 E4 0A F0 06
0AD0:20 10 19 C8 D0 F5 20 F9 10 20 0A 0B A9 01 8D 45
0AE0:0B 4C 63 09 1F 40 41 20 12 64 1F 40 41 52 69 65
0AF0:6E 20 19 41 61 20 65 66 66 61 63 65 72 2E 2E 2E
0B00:0A 00 4C 74 16 A9 00 8D 44 0B A9 00 8D 41 0B 8D
0B10:42 0B 2C E6 1B 10 04 A9 FD D0 02 A9 F8 8D 43 0B
0B20:60 20 FE 10 A0 06 B9 35 0B 20 10 19 88 10 F7 A9
```

```
0B30:00 8D 45 0B 60 0A 64 12 20 41 40 1F 00 00 00 00
0B40:00 00 00 00 00 00 00 00 20 FE 10 8E 73 0B 8C 74
0B50:0B 20 56 19 AD 74 0B 18 69 40 20 10 19 AD 73 0B
0B60:18 69 40 20 10 19 AD 47 0B F0 05 20 5B 19 A9 4D
0B70:4C 10 19 00 00 A9 00 8D 46 0B AD 40 0B 20 52 E4
0B80:A2 01 A0 02 20 E2 E5 AD 40 0B A0 00 91 83 C8 A5
0B90:6F 91 83 C8 A5 70 91 83 20 BE DE 20 E3 DF 85 83
0BA0:84 84 A5 81 25 82 30 03 4C 76 DD AD 46 0B C9 07
0BB0:90 02 E9 01 A0 01 91 83 A0 00 98 91 83 20 FE 10
0BC0:AE 40 0B EC 3D 0B F0 08 A9 A0 20 10 19 E8 D0 F3
0BD0:A9 00 8D E2 1B 60 20 BE DE 20 E3 DF A5 81 8D 3E
0BE0:0B 30 07 A5 82 8D 3F 0B 30 03 4C 76 DD A5 83 85
0BF0:85 A5 84 85 86 A0 00 B1 85 CD 3D 0B 90 05 F0 03
0C00:4C C2 18 8D 40 0B 8D 48 0C A0 02 B1 85 88 99 00
0C10:00 C0 00 D0 F6 A0 00 CC 40 0B F0 0B B1 08 99 01
0C20:02 20 10 19 C8 D0 F0 AE 40 0B EC 3D 0B F0 08 A9
0C30:AE 20 10 19 E8 D0 F3 AD 82 14 18 6D 40 0B AA AC
0C40:83 14 20 48 0B 4C 58 09 00 20 BE DE 20 E3 DF 85
0C50:83 84 84 A5 81 25 82 30 03 4C 76 DD A9 00 8D 45
0C60:0B 8D 44 0B EE 45 0B 20 05 0B EE 41 0B D0 0D EE
0C70:42 0B D0 08 EE 43 0B D0 03 4C C9 0C 20 02 0B AD
0C80:CC 16 F0 E6 AD 00 02 29 7F 8D 46 0B 20 05 0B AD
0C90:45 0B F0 03 20 21 0B AD 46 0B C9 0D F0 0C C9 0A
0CA0:90 08 4C 6A 0C A9 00 8D 46 0B AD 46 0B C9 0D D0
0CB0:02 A9 01 C9 07 D0 04 A9 09 D0 04 90 02 E9 01 A0
0CC0:01 91 83 A0 00 98 91 83 60 AD 44 0B F0 03 4C A5
0CD0:0C A9 01 8D 45 0B 8D 44 0B A0 00 B9 33 0A F0 06
0CE0:20 10 19 C8 D0 F5 20 0A 0B 4C 6A 0C 20 BE DE A9
0CF0:00 85 52 20 7B DD 20 6C DD A5 A0 85 85 A5 A1 85
0D00:86 A0 00 B1 85 AA A0 02 B1 85 88 99 08 00 C0 00
0D10:D0 F6 CA 30 09 B1 08 20 C7 18 C8 4C 12 0D 20 B7
0D20:00 F0 08 C9 3B F0 01 60 4C B1 00 A9 0A 20 10 19
0D30:A9 0D 4C 10 19 20 BE DE A9 00 85 52 20 7B DD 20
0D40:6C DD A5 A0 85 85 A5 A1 85 86 A0 00 B1 85 C9 25
0D50:90 03 4C C2 18 8D 97 0D A0 02 B1 85 88 99 08 00
0D60:C0 00 D0 F6 20 F5 E6 8A F0 03 20 D3 0D A0 08 B9
0D70:8E 0D 20 10 19 88 10 F7 A0 00 AE 97 0D CA 30 09
0D80:B1 08 20 C7 18 C8 4C 7D 0D A9 0A 4C 10 19 41 40
0D90:1F 64 12 20 41 40 1F 00 20 BE DE 20 E3 DF 85 83
0DA0:84 84 A5 81 25 82 30 03 4C 76 DD A0 00 AD 61 C0
0DB0:30 0E 88 D0 F8 A9 00 A0 01 91 83 88 98 91 83 60
0DC0:A0 03 A9 C6 20 A8 FC 2C 61 C0 10 E9 88 D0 F3 A9
0DD0:01 D0 E4 A9 07 4C 10 19 20 F5 E6 E0 29 B0 4B 8E
0DE0:2D 0E 20 F5 E6 E0 19 B0 41 8E 2F 0E 20 F5 E6 E0
0DF0:29 B0 37 8E 2E 0E 20 F5 E6 E0 19 B0 2D 8E 30 0E
0E00:20 F5 E6 E0 03 B0 23 8E 33 0E 20 F5 E6 8E 34 0E
0E10:20 F5 E6 E0 08 B0 13 8A 09 40 8D 31 0E 20 F5 E6
0E20:8A 09 50 8D 32 0E C9 58 90 0B 4C BD 18 00 00 00
0E30:00 00 00 00 00 AD 2E 0E 38 ED 2D 0E C9 02 90 EA
0E40:AD 30 0E 38 ED 2F 0E C9 02 90 DF 20 FE 10 A9 59
0E50:8D 8E 0F AD 34 0E F0 03 EE 8E 0F AD 2D 0E 18 69
0E60:40 8D 8B 0F AD 33 0E C9 02 D0 03 EE 8B 0F AD 31
0E70:0E 8D 90 0F AD 33 0E C9 02 F0 0D AD 32 0E 8D 92
0E80:0F A9 1B 8D 91 0F D0 08 A9 00 8D 91 0F 8D 92 0F
0E90:AD 2E 0E 38 ED 2D 0E 69 3D 8D 96 0F AC 33 0E B9
0EA0:98 0F 8D 94 0F B9 A4 0F 8D 93 0F B9 A7 0F 8D 97
0EB0:0F B9 B0 0F 8D 8C 0F AD 2F 0E 18 69 40 8D 8A 0F
0EC0:20 68 0F AC 33 0E B9 9B 0F 8D 94 0F B9 AA 0F 8D
0ED0:93 0F B9 AD 0F 8D 97 0F AD 30 0E 18 69 40 8D 8A
0EE0:0F 20 68 0F A9 00 8D 93 0F 8D 97 0F AD 2D 0E 18
0EF0:69 40 8D 8B 0F AD 2F 0E 18 69 40 8D 8A 0F EE 8A
0F00:0F AC 33 0E B9 9E 0F 8D 94 0F 20 76 0F AD 30 0E
0F10:18 69 3F CD 8A 0F D0 E6 AD 2F 0E 18 69 40 8D 8A
0F20:0F AD 2E 0E 18 69 40 8D 8B 0F EE 8A 0F AC 33 0E
0F30:B9 A1 0F 8D 94 0F 20 76 0F AD 30 0E 18 69 3F CD
0F40:8A 0F D0 E6 A9 0F 20 10 19 20 56 19 18 AD 2F 0E
0F50:69 41 20 10 19 18 AD 2D 0E 69 41 20 10 19 20 5B
0F60:19 AD 90 0F 20 10 19 60 A0 00 B9 89 0F 20 10 19
0F70:C8 C0 0F D0 F5 60 A0 00 B9 89 0F 20 10 19 C8 C0
0F80:0B D0 F5 AD 94 0F 4C 10 19 1F 40 40 0E 1B 59 1B
0F90:40 1B 50 00 00 12 40 00 70 23 5F 23 70 7E 6A 35
0FA0:7D 35 6A 7B 60 37 00 30 6B 00 22 75 00 21 7A 00
0FB0:0E 0E 0F 20 B7 00 F0 07 C9 2C F0 07 4C C9 DE A2
0FC0:01 D0 07 20 F5 E6 E0 19 B0 11 CA 30 0D A9 0A 20
0FD0:10 19 A9 0D 20 10 19 4C CA 0F 60 4C BD 18 20 60
0FE0:19 A0 02 B9 F2 0F 20 10 19 88 10 F7 20 60 19 4C
0FF0:AC 19 68 39 1B A0 05 B9 78 10 20 10 19 88 10 F7
```

```
1000:20 BE DE 20 E3 DF 85 83 84 84 A5 81 25 82 30 03
1010:4C 76 DD 2C E6 1B 10 04 A9 0F D0 02 A9 3C 8D 77
1020:10 20 AC 19 B0 0A CE 77 10 AD 77 10 D0 F3 F0 15
1030:C9 13 D0 ED 20 AC 19 C9 13 F0 F9 C9 53 F0 04 C9
1040:51 D0 DE A9 01 A0 01 48 91 83 88 98 91 83 68 F0
1050:23 20 C7 19 A0 00 B9 7E 10 F0 08 49 A5 20 10 19
1060:C8 D0 F3 20 C7 19 20 C7 19 20 C7 19 20 C7 19 A9
1070:0C 4C 10 19 4C 83 11 00 68 39 1B 6F 39 1B BA E5
1080:E4 85 B7 C1 BA E5 E4 BE 9C DA A9 BA EF ED EB CA
1090:DC C4 D0 85 D6 C0 D7 D3 C0 D0 D7 85 F5 CA C8 FA
10A0:E9 CC CB CE 85 97 8B 94 BA EE ED D5 CA D0 D7 85
10B0:E4 D5 D5 C9 C0 85 EC EC 85 C0 D1 85 E8 C4 C6 CC
10C0:CB D1 CA D6 CD BA E9 EC 8D C6 8C 85 94 9C 9D 9D
10D0:85 E6 F5 89 85 EF E9 E7 85 83 85 F5 CA C8 82 D6
10E0:BA E8 F6 FA FA FA BA EA E8 8D 94 8C 85 96 9C 8B
10F0:90 94 8B 97 91 8B 91 96 00 A9 11 4C 10 19 A9 14
1100:4C 10 19 20 F5 E6 E0 19 90 03 4C BD 18 A0 0B B9
1110:2F 11 20 10 19 88 10 F7 CA 30 08 A9 0B 20 10 19
1120:4C 18 11 A0 07 B9 3B 11 20 10 19 88 10 F7 60 43
1130:69 3A 1B 04 66 3A 1B 41 41 1F 14 43 6A 3A 1B 04
1140:66 3A 1B 20 F5 E6 E0 19 90 03 4C BD 18 A0 0B B9
1150:6F 11 20 10 19 88 10 F7 CA 30 08 A9 0A 20 10 19
1160:4C 58 11 A0 07 B9 7B 11 20 10 19 88 10 F7 60 43
1170:69 3A 1B 04 66 3A 1B 41 58 1F 14 43 6A 3A 1B 04
1180:66 3A 1B 20 60 19 A0 02 B9 9A 11 20 10 19 88 10
1190:F7 20 AC 19 20 60 19 4C 60 19 67 39 1B 20 F5 E6
11A0:8A F0 2B 18 69 40 8D E7 1B C9 69 B0 21 20 F5 E6
11B0:8A F0 1B 18 69 40 8D E8 1B C9 59 B0 11 A9 1F 20
11C0:10 19 AD E8 1B 20 10 19 AD E7 1B 4C 10 19 4C BD
11D0:18 20 BE DE 20 E3 DF 85 83 84 84 A5 81 25 82 30
11E0:08 4C 76 DD A9 06 8D 22 12 A0 05 B9 23 12 20 10
11F0:19 88 10 F7 20 AC 19 B0 0C CE 22 12 2C 22 12 D0
1200:E8 A9 00 F0 14 C9 1F D0 EB 20 AC 19 C9 3F 90 D9
1210:20 AC 19 C9 3F 90 D2 A9 01 A0 01 91 83 88 98 91
1220:83 60 00 61 1B 02 66 3A 1B 20 5B 19 A9 49 4C 10
1230:19 20 5B 19 A9 48 4C 10 19 A9 0E 4C 10 19 A9 03
1240:8D 9D 12 A0 00 B9 63 12 F0 06 20 10 19 C8 D0 F5
1250:20 AC 19 B0 16 CE 9D 12 AD 9D 12 D0 F3 8D A1 12
1260:4C A2 12 1B 3A 66 03 1B 39 7B 00 C9 01 D0 E1 20
1270:AC 19 90 CF 8D 9E 12 20 AC 19 90 C7 8D 9F 12 20
1280:AC 19 90 BF 8D A0 12 20 AC 19 B0 03 4C 43 12 C9
1290:04 F0 03 4C 43 12 A9 01 8D A1 12 D0 05 00 00 00
12A0:00 00 20 BE DE 20 E3 DF 85 83 84 84 A5 81 25 82
12B0:30 03 4C 76 DD AD 9E 12 A0 01 91 83 88 98 91 03
12C0:20 BE DE 20 E3 DF 85 83 84 84 A5 81 25 82 30 03
12D0:4C 76 DD AD 9F 12 A0 01 91 83 88 98 91 83 20 BE
12E0:DE 20 E3 DF 85 83 84 84 A5 81 25 82 30 03 4C 76
12F0:DD AD A0 12 A0 01 91 83 88 98 91 83 20 BE DE 20
1300:E3 DF 85 83 84 84 A5 81 25 82 30 03 4C 76 DD AD
1310:A1 12 A0 01 91 83 88 98 91 83 60 20 F5 E6 8E E5
1320:1B 8A 09 C0 8D 5E 13 8D 5B 13 8D 2F 13 AD 0C C2
1330:C9 31 D0 33 A9 C0 85 1B 85 19 AD E5 1B 0A 0A 0A
1340:0A 69 88 85 18 85 1A E6 1A 20 60 19 20 60 19 AD
1350:C6 1A F0 03 4C 6C 13 A9 07 20 00 C2 AD 00 C2 C9
1360:2C F0 03 20 B9 13 60 A9 03 4C 09 BE A9 38 8D 7C
1370:04 A9 6B 8D 7D 04 A9 81 8D 7E 04 A9 7C 85 42 85
1380:3C A9 04 85 43 85 3D 85 3F A9 7E 85 3E 38 20 11
1390:C3 A0 01 B9 30 BE 99 B7 13 88 10 F7 A9 00 8D 30
13A0:BE A9 C2 8D 31 BE A9 07 20 ED FD A0 01 B9 B7 13
13B0:99 30 BE 88 10 F7 60 00 00 A9 FF 8D E6 1B AD E5
13C0:1B 09 C0 8D E0 1B 8D 14 14 8D 16 14 8D 18 14 8D
13D0:1A 14 8D 1C 14 8D E3 13 8D E9 13 8D EF 13 8D F5
13E0:13 AD 0D C2 8D 15 14 AD 0E C2 8D 17 14 AD 0F C2
13F0:8D 19 14 AD 10 C2 8D 1B 14 AD E5 1B 0A 0A 0A 0A
1400:8D E1 1B 60 6C 13 14 6C 15 14 6C 17 14 6C 19 14
1410:6C 1B 14 00 C2 45 C2 46 C2 47 C2 48 C2 20 5B 19
1420:A9 5D 4C 10 19 20 5B 19 A9 5A 4C 10 19 20 33 14
1430:4C 85 14 A9 03 8D 7A 14 A0 00 B9 7B 14 F0 06 20
1440:10 19 C8 D0 F5 20 AC 19 B0 0E CE 7A 14 AD 7A 14
1450:D0 F3 8D 84 14 4C 79 14 C9 1F D0 E9 20 AC 19 C9
1460:3F 90 D5 E9 40 8D 83 14 20 AC 19 C9 3F 90 C9 E9
1470:40 8D 82 14 A9 01 8D 84 14 60 00 1B 3A 66 02 1B
1480:61 00 00 00 00 20 BE DE 20 E3 DF 85 83 84 84 A5
1490:81 25 82 30 03 4C 76 DD AD 82 14 A0 01 91 83 88
14A0:98 91 83 20 BE DE 20 E3 DF 85 83 84 84 A5 81 25
14B0:82 30 03 4C 76 DD AD 83 14 A0 01 91 83 88 98 91
14C0:83 20 BE DE 20 E3 DF 85 83 84 84 A5 81 25 82 30
```

```
14D0:03 4C 76 DD AD 84 14 A0 01 91 83 88 98 91 83 60
14E0:20 F5 E6 E0 04 90 03 4C BD 18 8E F9 14 20 5B 19
14F0:18 A9 4C 6D F9 14 4C 10 19 00 20 F5 E6 E0 00 F0
1500:02 A2 FF 8E E2 1B 60 20 5B 19 A9 5C 4C 10 19 20
1510:F5 E6 8E 71 15 20 F5 E6 8E 72 15 20 F5 E6 8E 73
1520:15 20 F5 E6 8E 74 15 AD 71 15 0D 72 15 C9 08 90
1530:03 4C BD 18 A9 0E 20 10 19 20 5B 19 18 AD 71 15
1540:69 40 20 10 19 20 5B 19 18 AD 72 15 69 50 20 10
1550:19 20 5B 19 A9 49 2C 74 15 F0 03 38 E9 01 20 10
1560:19 20 5B 19 A9 59 AE 73 15 F0 03 18 69 01 4C 10
1570:19 00 00 00 00 20 F5 E6 8E 06 16 20 F5 E6 8E 07
1580:16 20 F5 E6 8E 08 16 20 F5 E6 8E 09 16 20 F5 E6
1590:8E 0A 16 20 F5 E6 8E 0B 16 AD 06 16 C9 04 90 03
15A0:4C BD 18 AD 07 16 0D 08 16 C9 08 B0 F3 A9 0F 20
15B0:10 19 20 5B 19 18 A9 4C 6D 06 16 20 10 19 20 5B
15C0:19 18 A9 40 6D 07 16 20 10 19 20 5B 19 18 A9 50
15D0:6D 08 16 20 10 19 20 5B 19 A9 49 AE 09 16 F0 03
15E0:38 E9 01 20 10 19 20 5B 19 A9 5C AE 0A 16 F0 03
15F0:18 69 01 20 10 19 20 5B 19 A9 59 AE 0B 16 F0 03
1600:18 69 01 4C 10 19 00 00 00 00 00 A0 00 B9 1A
1610:16 F0 06 20 10 19 C8 D0 F5 60 1B 39 7F 1B 5C 1B
1620:49 1B 59 1B 4C 1B 47 1B 50 20 00 20 60 19 20 F5
1630:E6 E0 00 F0 0C A0 07 B9 4D 16 20 10 19 88 10 F7
1640:60 A0 07 B9 55 16 20 10 19 88 10 F7 60 43 69 3A
1650:1B 04 66 3A 1B 43 6A 3A 1B 04 66 3A 1B 20 BE DE
1660:20 E3 DF 24 81 30 04 24 82 30 03 4C 76 DD 20 74
1670:16 4C 99 16 A9 00 8D CC 16 20 60 19 90 17 C9 13
1680:D0 03 4C 28 17 C9 16 F0 44 C9 19 F0 40 29 7F 8D
1690:00 02 EE CC 16 AD CC 16 60 AD 00 02 C9 07 D0 07
16A0:A9 09 8D 00 02 D0 09 90 07 C9 0A B0 03 CE 00 02
16B0:20 52 E4 A2 00 A0 02 20 E2 E5 AD CC 16 A0 00 91
16C0:83 C8 A5 6F 91 83 C8 A5 70 91 83 60 00 20 AC 19
16D0:B0 03 4C 95 16 AA 29 F0 C9 40 F0 18 8A A0 00 D9
16E0:26 1A F0 09 C8 C8 C0 22 D0 F5 4C 95 16 C8 B9 26
16F0:1A 4C 8D 16 8E 26 17 20 AC 19 B0 03 4C 95 16 8D
1700:27 17 A0 00 AD 26 17 D9 49 1A D0 10 C8 AD 27 17
1710:D9 49 1A D0 08 C8 B9 49 1A 4C 8D 16 C8 C8 C8 C0
1720:2A D0 E1 4C 95 16 00 00 20 AC 19 B0 03 4C 95 16
1730:C9 13 F0 F4 89 40 F0 F5 C9 4A B0 F1 29 0F 4C 8D
1740:16 20 5B 19 A9 59 4C 10 19 A9 FF 8D 51 17 4C CC
1750:08 00 A0 02 B9 ED 1B 99 F5 03 88 10 F7 60 A9 0F
1760:4C 10 19 20 B7 00 F0 07 C9 2C F0 07 4C C9 DE A2
1770:00 F0 03 20 F5 E6 A9 41 8D C1 17 8A F0 2E C9 08
1780:90 03 4C DD 18 20 FE 10 8A 69 40 8D BB 17 69 10
1790:8D BD 17 A0 0C B9 B7 17 20 10 19 88 10 F7 EE C1
17A0:17 AD C1 17 C9 59 D0 EB 20 C4 17 60 A9 0C 20 10
17B0:19 20 FE 10 4C C4 17 0F 67 12 5F 40 1B 50 1B 0E
17C0:41 40 1F 14 A0 06 B9 D0 17 20 10 19 88 10 F7 60
17D0:0A 64 12 20 41 40 1F 20 F5 E6 8A F0 14 C9 19 B0
17E0:10 69 40 8D 14 18 20 F5 E6 20 FE 10 8A C9 08 90
17F0:03 4C BD 18 69 40 8D 0E 18 69 10 8D 10 18 A0 0C
1800:B9 0A 18 20 10 19 88 10 F7 60 0F 67 12 5F 40 1B
1810:50 1B 0E 41 40 1F 14 20 F5 E6 E0 29 B0 2B 8E B8
1820:18 20 F5 E6 E0 19 B0 21 8E BA 18 20 F5 E6 E0 29
1830:B0 17 8E B9 18 20 F5 E6 E0 19 B0 0D 8E BB 18 20
1840:F5 E6 8E BC 18 E0 08 90 03 4C BD 18 AD B8 18 CD
1850:B9 18 B0 F5 AD BA 18 CD BB 18 B0 ED 20 FE 10 20
1860:56 19 A9 40 18 6D BA 18 20 10 19 A9 40 18 6D B8
1870:18 20 10 19 A9 0E 20 10 19 20 5B 19 A9 40 18 6D
1880:BC 18 48 20 10 19 20 5B 19 68 69 10 20 10 19 A9
1890:5F 20 10 19 A9 12 20 10 19 AD B9 18 38 ED B8 18
18A0:18 69 40 20 10 19 EE BA 18 AD BA 18 CD BB 18 90
18B0:AE F0 AC A9 0F 4C 10 19 00 00 00 00 A9 02 4C
18C0:09 BE A2 B0 4C 12 D4 20 E7 19 8D 55 19 20 60 19
18D0:A0 00 2C E2 1B 10 06 B9 25 1A 4C E0 18 B9 FD 19
18E0:F0 0B CD 55 19 F0 0F C8 C8 C8 C8 D0 E5 AD 55 19
18F0:20 10 19 4C F1 19 C8 A2 03 2C E2 1B 10 06 B9 25
1900:1A 4C 07 19 B9 FD 19 20 10 19 C8 CA D0 EB F0 E3
1910:2C E6 1B 30 1C 8D 2F 19 8E 30 19 A2 00 A1 1A 29
1920:10 F0 F8 AD 2F 19 81 18 AE 30 19 20 FB 19 60 00
1930:00 48 8E 53 19 8C 54 19 20 A5 19 A9 00 20 10 14
1940:90 F6 20 A5 19 68 20 0D 14 AE 53 19 AC 54 19 20
1950:FB 19 60 00 00 00 A9 1F 4C 10 19 A9 1B 4C 10 19
1960:2C E6 1B 30 1C 8E EB 1B A2 00 A1 1A 29 08 D0 05
1970:AE EB 1B 18 60 A1 18 29 7F AE EB 1B 38 20 FC 19
1980:60 8E A3 19 8C A4 19 20 A5 19 A9 01 20 10 14 90
1990:0B 20 A5 19 20 0A 14 29 7F 20 FC 19 AE A3 19 AC
```



```
19A0:A4 19 60 00 00 AE E0 1B AC E1 1B 60 A9 C0 8D C5
19B0:19 8D C6 19 20 60 19 90 01 60 EE C5 19 D0 F5 EE
19C0:C6 19 D0 F0 60 00 00 8C EC 1B 8E EB 1B 48 A0 0A
19D0:A9 C6 20 A8 FC AD 00 C0 C9 9B F0 03 88 D0 F1 68
19E0:AC EC 1B AE EB 1B 60 8D EA 1B 8E EB 1B 8C EC 1B
19F0:60 AD EA 1B AE EB 1B AC EC 1B 60 60 60 7E 00 19
1A00:48 5E 00 19 43 5D 19 27 00 23 19 23 00 5B 19 30
1A10:00 7D 19 41 65 7B 19 42 65 40 19 41 61 7C 19 41
1A20:75 5C 19 4B 63 00 23 A3 24 A4 26 A3 27 DD 2C BC
1A30:2D A2 2E BE 2F F6 30 DB 31 AB 38 AF 3C AF 3D AF
1A40:3E AF 6A AA 7A AA 7B C2 00 41 65 FD 42 65 FB 43
1A50:65 E5 48 65 E5 41 61 C0 43 61 E1 48 61 E1 43 69
1A60:E9 48 69 E9 43 6F EF 48 6F EF 41 75 FC 48 75 F5
1A70:4B 63 DC 00 C7 08 49 0C EC 0C 35 0D 98 0D D3 0D
1A80:D8 0D B3 0F DE 0F F5 0F F9 10 FE 10 03 11 43 11
1A90:83 11 9D 11 D1 11 29 12 31 12 39 12 3E 12 1B 13
1AA0:1D 14 25 14 2D 14 E0 14 FA 14 07 15 0F 15 75 15
1AB0:0C 16 2B 16 5D 16 41 17 49 17 52 17 5E 17 63 17
1AC0:D7 17 17 18 00 00 00 41 4D 42 52 43 53 44 6D 45
1AD0:51 46 50 47 6C 48 6E 00 06 E1 E3 E3 E5 F0 F4 07
1AE0:E1 E3 E3 E5 F0 F4 E6 07 E1 E6 E6 E9 E3 E8 E5 06
1AF0:E1 EC E5 F2 F4 E5 05 E1 F0 F0 E5 EC 03 E2 E9 F0
1B00:05 E3 E1 E4 F2 E5 07 E3 E8 E1 F2 E9 EF F4 04 E3
1B10:EE F8 EE 04 E3 EE F8 F2 06 E3 E5 F2 F6 E9 F3 08
1B20:E3 F5 F2 E9 EE F6 E9 F3 03 E2 E1 F3 04 E8 E1 F5
1B30:F4 05 E4 E5 E3 EE F8 07 E4 E5 F0 EC E1 E3 E5 07
1B40:E5 EE EC E9 E7 EE E5 04 E6 E9 F8 E5 01 BF 01 A8
1B50:05 E9 E4 E5 EE F4 04 E9 EF E9 F4 01 BE 05 EC E9
1B60:E7 EE E5 08 EC EF E3 E1 EC E9 F3 E5 04 ED EF E4
1B70:E5 03 ED F4 E8 01 BD 06 F0 E1 F2 E1 ED E7 06 F0
1B80:E1 F2 E1 ED F4 06 F2 E5 E9 EE E9 F4 07 F2 EF F5
1B90:EC E5 E1 F5 06 F3 E1 E9 F3 E9 E5 09 F3 E1 EE F3
1BA0:EC E9 E7 EE E5 06 F3 E5 E3 F2 E5 F4 07 F3 E5 F2
1BB0:F6 EF E6 E6 03 F4 F8 F4 08 F6 E9 E4 E5 E3 F2 E1
1BC0:EE 08 F6 E9 E4 EC E9 E7 EE E5 07 F6 E9 E4 F2 E5
1BD0:E3 F4 00 04 E9 EE E9 F4 00 00 00 00 00 00 00 00
1BE0:C2 20 00 00 15 02 FF 00 00 82 00 02 00 4C 03 BE
1BF0:30 30
```

Répom'deur...

Macintosh

...enregistreur, télématique, interrogeable...

Jean-Luc Bazanegue

Le numéro de Pom's que vous avez entre les mains contient un puissant ensemble de routines en langage machine, formant une librairie d'instructions télématiques qui s'ajoutent à celles du Basic Microsoft (2.0 et plus). Afin de donner un exemple de programmation sur la base de ces nouvelles fonctions, nous avons pensé qu'il serait judicieux d'écrire un serveur/répondeur télématique interrogeable à distance. Il ne faut cependant pas s'y tromper : malgré sont caractère 'démonstratif', ce serveur est parfaitement fiable, pratique, et son ergonomie pourrait faire rougir de honte les 'programmeurs' de la plupart des serveurs Vidéotex dits serveurs-kiosque.

Si vous ne programmez pas, et comme à l'accoutumée, cette application est disponible 'prête à lancer' sur la disquette d'accompagnement Mac 34 de Pom's.

Un répondeur télématique pour quoi faire ?

Il est inutile de présenter ici le traditionnel répondeur téléphonique, matériel pas très pratique (il faut enregistrer un message tout en ayant l'air décontracté, dynamico-branché – c'est à la mode – et intelligent – ce qui, en ce qui nous concerne, requiert beaucoup d'efforts ainsi qu'une attention soutenue), et particulièrement irritant pour la plupart des correspondants, qui généralement raccrochent rageusement avant d'appeler le maudit concurrent.

Cela ne va pas toujours jusqu'à la perte d'un client mais peut vous faire perdre une information, un rendez-vous, ou tout simplement un chaleureux et amical 'Comment vas-tu ?'. Nous passerons rapidement sur le 'yau de poil...' pour en venir à notre serveur/répondeur télématique.

La généralisation du Minitel, en particulier avec l'annuaire téléphonique et les grandes sociétés de vente par correspondance, fait que la tonalité aiguë de la porteuse, qui est devenue parfaitement familière, est en passe de provoquer le geste réflexe qui consiste en la mise sous tension du Minitel suivi d'une pression sur la touche 'Connexion/Fin' ; c'est ce que feront la plupart de vos correspondants, qui auront alors tout leur temps pour écrire – voire même peaufiner – un message à votre attention.

Répomdeur, puisque tel est le nom du serveur que Pom's vous propose, vous donnera donc, outre le style dynamico-branché cité plus haut et que vous ne recherchez pas forcément, la possibilité de stocker un maximum d'informations avec l'outil de communication qu'est votre Macintosh.

De plus, votre serveur-répondeur télématique *Répomdeur* est interrogeable à distance depuis n'importe quel Minitel raccordé à une ligne téléphonique. Il vous sera donc possible à tous moments d'appeler votre Macintosh pour lire les messages enregistrés et prendre immédiatement les mesures qui s'imposent en fonction des besoins ou propositions de vos amis et/ou clients.

À propos du programme "Envoi Vidéotex" (Pom's 33)

Le 'mini-protocole' utilisé par ce programme – qui envoie les images créées par "Paint ➡ Minitel" – n'est pas reconnu par tous les Minitel (demande d'identification). Dans certains cas, l'image Vidéotex n'est donc pas émise.

Afin de remédier à cela, nous avons placé sur la disquette d'accompagnement de ce numéro une version plus universelle.

Les disquettes d'accompagnement Macintosh maintenant en 800Ko

La taille des systèmes d'exploitation, des utilitaires, polices de caractères... n'allant pas en diminuant, il devient très difficile de tout loger sur une disquette simple face 400Ko.

Nous passons donc aux inévitables 800Ko ce qui nous permet, pour ce numéro 34 de Pom's, d'ajouter une 'démo' de "dBase Mac" en plus des programmes et fichiers publiés dans ces pages.

Le matériel

L'utilisation de *Répomdeur* requiert la présence du matériel suivant :

- une ligne téléphonique ;
- un Macintosh 512Ko, Plus, SE ou II ;
- un Minitel bi-standard dit 'retournable' (ils le sont pratiquement tous) ;
- un câble de liaison Macintosh ➡ Minitel tel que celui utilisé par les programmes destinés à l'utilisation du Minitel et publiés depuis le numéro 27 (si par hasard ce câble vous faisait défaut, les Éditions MEV peuvent vous le fournir) ;

- un détecteur d'appel téléphonique que vous pourrez réaliser à l'aide du schéma publié dans le numéro 32 de Pom's. Cet appareil, dont le double connecteur s'intercale entre la souris et le Macintosh, peut aussi vous être fourni par les Éditions MEV. Si vous décidez de construire vous-même l'objet, nous vous recommandons de faire très attention à la qualité et au branchements du relais ; une isolation galvanique parfaite est indispensable.

Le synoptique joint à cet article indique quelles sont les liaisons à effectuer entre les différents appareils. On peut noter que la présence de l'appareil téléphonique n'est pas indispensable.

Démarrage du serveur

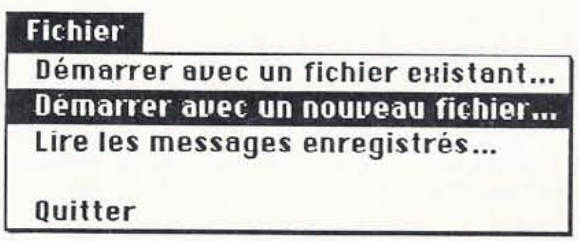
L'article "Démarrer avec un fichier existant..." permet l'utilisation d'un fichier qui contient déjà le mot de passe, votre nom – ou raison sociale – et votre numéro de téléphone. Le mot de passe vous sera indispensable si vous comptez lire vos messages à distance depuis un autre Minitel, alors que les deux derniers 'éléments' seront employés pour l'affichage sur l'écran du Minitel de votre correspondant d'un texte que nous pourrions qualifié 'd'écran de reconnaissance'. Par exemple, pour les Éditions MEV, nous affichons :

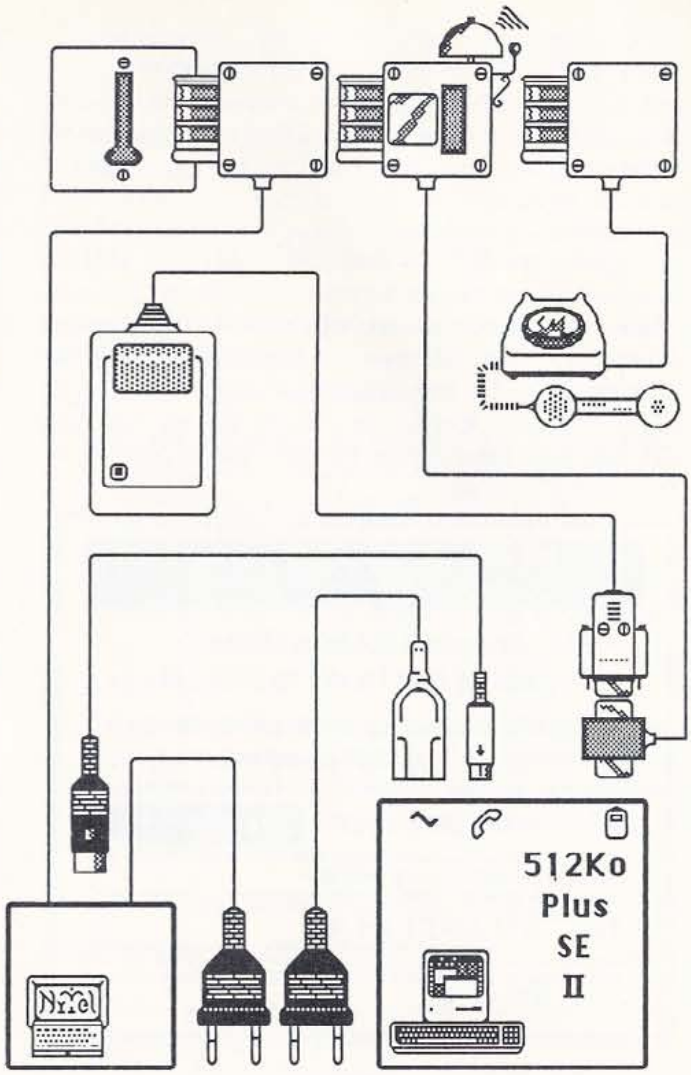
```
Vous êtes bien au (1) 39 51 24 43
- Editions MEV -
```

ce qui rassure tout de suite votre correspondant sur le bon aboutissement de son appel.

Cet article du menu "Fichier" conduit naturellement à l'affichage de la fenêtre de sélection de fichiers habituelle.

Note : au lancement du programme, la fenêtre de sélection s'affiche, permettant ainsi un démarrage direct du serveur. Si vous appeliez Répomdeur pour effectuer une autre opération (démarrage avec un nouveau fichier, lecture des messages...), faites simplement un 'clic' sur le bouton 'Annuler'.





L'ouverture d'un nouveau fichier sera parfois nécessaire, ne serait-ce que lors de la première utilisation du serveur. Dans ce but, *Répomdeur* affiche une fenêtre de saisie qui vous permettra l'initialisation de vos nom ou raison sociale et numéro de téléphone, ainsi que du mot de passe qui autorisera les 'fonctions de service' à distance (lecture des messages, remise à zéro du fichier et changement du mot de passe).

Le passage d'une zone de saisie à l'autre se fait classiquement par un 'clic', un 'retour-chariot' ou une action sur la touche de tabulation. Certains contrôles sont effectués par le programme afin d'éviter les plus grosses erreurs de saisie ; ainsi, pour empêcher l'apparition dans le mot de passe de caractères difficiles – voire même impossibles – à saisir sur le clavier du Minitel, on limite les possibilités à A ➡ Z et 0 ➡ 9. Au cas où un ou plusieurs caractères se trouveraient hors de ces gammes, l'erreur est signalée par un 'bip' suivi de l'affichage en inverse du texte indiquant les possibilités. Le même principe est utilisé pour la saisie des nom et numéro de téléphone dont la longueur est vérifiée afin que l'affichage des chaînes de caractères ne provoque pas de désordre sur l'écran du Minitel servi.

Le bouton 'OK' confirme vos choix et, si le contenu des zones de saisie est valide (les vérifications sont faites lors de la sollicitation de ce bouton), une seconde fenêtre apparaît, autorisant le baptême du fichier serveur. Une fois le fichier nommé, et si vous n'utilisez pas le bouton 'Annuler', le serveur démarre avec le nouveau fichier.

Quand le Macintosh sert...

... son écran affiche un minimum d'informations.
Il s'agit de :

```
Attente d'un appel (Commande+'.' pour
interrompre le serveur)
```

lorsque le Macintosh est en attente d'un appel ;

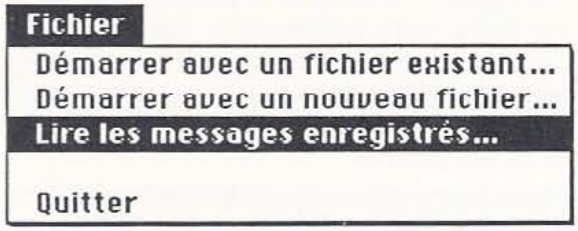
```
Appel reçu, attente de connexion…
```

en attendant que votre correspondant appuie sur Connexion/Fin ;

```
Connexion ok, serveur actif (connexion n° N)
```

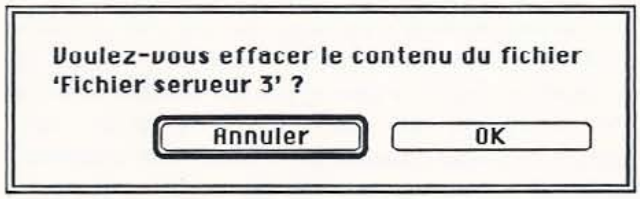
quand le Macintosh 'converse' avec le Minitel servi. N représente le nombre de connexions depuis le lancement de *Répomdeur*.

Lecture des messages



Nous ne parlerons ici que de la lecture des messages sur le Macintosh ; la consultation du fichier à distance sera évoquée un peu plus loin.

Le premier message affiché est le dernier reçu. Les boutons 'Précédent' et 'Suivant' seront utilisés pour passer d'un message à l'autre. Le bouton 'Vider' permet la remise à zéro du fichier, après confirmation bien sûr :



Côté Minitel servi

Après affichage des écrans de présentation, votre correspondant se trouve devant une 'page' qui lui propose trois choix possibles :

```
[1] Laisser un message
```

donne accès à différentes zones (parfaitement contrôlées par votre Macintosh-serveur), qui permettent la saisie d'un nom (obligatoire), prénom, numéro de téléphone et message de 196 caractères (7 lignes de 28 caractères). Nous avons particulièrement soigné l'ergonomie du serveur, pour la partie utilisateur ou la partie 'maître' (fonctions de service) ; ainsi, par exemple si votre correspondant désire passer plusieurs messages de suite, il trouvera lors des nouveaux affichages de la fenêtre de saisie ses nom, prénom, et numéro de téléphone affichés par défaut (mais modifiables) et le curseur placé sur la première position de la zone de saisie du message. Dans le même esprit, avant d'enregistrer le message, on le présente tel qu'il sera réellement enregistré, permettant ainsi au correspondant une éventuelle modification ou, pourquoi pas, annulation.

[2] Fonctions de service

Ceci vous est destiné. L'écran qui suit ce choix est un passage obligé pour aller vers les fonctions de service ; il faut ici frapper le mot de passe. Le serveur autorise trois essais au terme desquels l'indélicat est déconnecté. Afin de se protéger des indélicats futés, il n'est pas possible de faire plus de trois essais même en revenant à l'écran précédent entre deux tentatives. Notons que, par souci de discrétion, les caractères du mot de passe sont retournés à l'écran sous la forme d'astérisque ('*').

[3] Quitter ce répondeur

Pour ceux qui s'aperçoivent au dernier moment qu'il n'ont rien à faire là...

Fonctions de service

Vous seul pouvez accéder à ces fonctions puisque le mot de passe est indispensable.

[1] Lire les messages

Le fonctionnement est comparable à la lecture sur l'écran de votre Macintosh, les boutons 'Précédent' et 'Suivant' étant seulement remplacés par les touches 'Retour' et 'Suite'.

[2] Effacer le fichier

Identique à la fonction disponible sur le Macintosh avec, bien sûr, demande de confirmation.

[3] Changer le mot de passe

Ceci peut être utilisé si, par exemple, le mot de passe a été communiqué à fin de démonstration. Le changement de ce mot est doté de nombreuses protections qui garantissent une modification parfaitement fiable.

Programme 'Répom'deur'

* *les indentations indique la continuité de la ligne.*

```
'• Répom'deur — © 1988 Christi
  an Piard, Jean-Luc Bazanegue
  et Pom's •
LIBRARY "Pom_Link 2.1"
DEFINT A-Z
DIM Message$(6),Gris(3),Rect(3
  )
ON BREAK GOSUB Bloque:BREAK
   ON
GOSUB Initialisations
ON MENU GOSUB Menus:MENU O
  N
ON MOUSE GOSUB Souris
GOSUB OuvreFichier
AttenteOuverture:
IF Dr THEN FichierOuvert
GOTO AttenteOuverture
Menus:
MENU OFF
nMenu=MENU(0)
nArticle=MENU(1)
IF nMenu<>1 THEN FinMenus
ON nArticle GOSUB DemExistant
  ,DemNouveau,Lire,Rien,Quitte
  r
FinMenus:
MENU ON
MENU
RETURN
DemExistant:
GOSUB OuvreFichier
RETURN
DemNouveau:
GOSUB NouveauFichier
RETURN
Arreter:
WINDOW CLOSE 1
MENU 1,1,1
MENU 1,2,1
MENU 1,3,1
Dr=Faux
GOTO AttenteOuverture

Rien:
RETURN
Quitter:
LIBRARY CLOSE
WINDOW CLOSE 1
SYSTEM
FichierOuvert:
WINDOW 1,"État du serveur",(0
  ,39)-(512,342),1
TEXTFONT 0:TEXTSIZE 12
CLS
MENU 1,1,0
MENU 1,2,0
MENU 1,3,0
ON ERROR GOTO Erreur
RepriseAttente:
GOSUB VideVariables
ChaineEtat$="Attente d'un appe
  l ("+CHR$(17)+" + '.' pour i
  nterrompre le serveur)"
GOSUB spAfficheEtat
'• attente d'un appel •
AttenteAppel:
MENU 1,5,0
ON BREAK GOSUB ArretServeur
MENU OFF
MOUSE ON
Dr=Faux
DrArret=Faux
C$=""
Dlog=0
WHILE NOT Dr AND NOT DrArre
  t AND C$<>CHR$(27) AND Dlo
  g<>4
C$=INKEY$
Dlog=DIALOG(0)
WEND
MOUSE OFF
MENU ON
ON BREAK GOSUB Bloque
MENU 1,5,1
IF DrArret OR C$=CHR$(27) OR
   Dlog=4 THEN Arreter
OPEN "R",1,Fichier$,244
FIELD 1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
```

```
GET 1,1
Mot$=zMot$
NomServeur$=zNom$
TelephoneServeur$=zTel$
tlMot:
IF RIGHT$(Mot$,1)=" " THEN M
  ot$=LEFT$(Mot$,LEN(Mot$)-1)
  :GOTO tlMot
tlNom:
IF RIGHT$(NomServeur$,1)=" "
  THEN NomServeur$=LEFT$(NomS
  erveur$,LEN(NomServeur$)-1):
  GOTO tlNom
tlTel:
IF RIGHT$(TelephoneServeur$,1
  )=" " THEN TelephoneServeur$
  =LEFT$(TelephoneServeur$,LE
  N(TelephoneServeur$)-1):GOT
  O tlTel
ChaineEtat$="Appel reçu, atten
  te de connexion…"
GOSUB spAfficheEtat
CnxR! Dr
IF NOT Dr THEN Deconnexion
NombreConnexions=NombreConnexi
  ons+1
NombreConnexions$=STR$(NombreC
  onnexions)
ChaineEtat$="Connexion ok, ser
  veur actif (connexion n°"+No
  mbreConnexions$+")"
GOSUB spAfficheEtat

'• Première page •
Deplace! 1,13
AfficheR! " /~~~~\   /~~~\  /~
  ~~~~~~~~\ /\ /~~~~~\"
AfficheR! "/      \ /     \ {
          } {} { /~\_/"
AfficheR! "{ /~~\ } { /~\ } {
  /~\ /~\ } {} { \__"
AfficheR! "{ {  } } { { } } {
  } { } { } {/ \     \"
AfficheR! "{ {  } } { { } } {
  } { } { } ~   \__   \"
AfficheR! "{ \__/ } { \_/ } {
  } { } { }         \ }"
AfficheR! "{      / \     / {
  } { } { }    /~\_/ }"
AfficheR! "{ \___/   \___/  \_
  / \_/ \_/    \_____/"
AfficheR! "{ }"
AfficheR! "{ }"
AfficheR! "{ }"
Affiche! "\_/     "
ParamT! 1,2,0,0,0,0
Affiche! "- La revue des Apple
  -"
Haut! 12
Bas! 4
Deplace! 9,17
ParamT! 0,2,0,0,0,0
Affiche! "_"+CHR$(18)+CHR$(86
  )
Deplace! 12,18
ParamT! 0,3,0,0,0,0
Affiche! "_"+CHR$(18)+CHR$(92
  )
Temp!=TIMER
WHILE TIMER<Temp!+2
WEND
'• Seconde page •
Page2:
VidEcran!
Cadre! 1,1,40,7,2,0,5,0
Cadre! 1,8,40,20,2,0,5,0
Deplace! 5,3
Affiche! "Vous êtes bien au "+
  TelephoneServeur$
Deplace! 5,5
Mode! 2
AfficheR! NomServeur$
Deplace! 4,10
Affiche! "Votre interlocuteur
  est un"
Deplace! 4,12
Affiche! "ordinateur Apple prê
  t à enregistrer"
Deplace! 4,14
Affiche! "vos messages et conf
  idences..."
Deplace! 13,18
Affiche! "Accès au menu : "
Inverse!
Affiche! "   Suite   "
Deplace! 18,19
Affiche! "Renoncer : "
Inverse!
Affiche! " Connexion "
BouclePage2:
AcceptF! Dr
ON Dr+1 GOTO DeconnexionF,Som
  maire,Inact1,Page2,Inact1,In
  act1,Sommaire,Sommaire,Decon
  nexion,Inact1
Inact1:
GOSUB spToucheInactive
GOTO BouclePage2
'• Sommaire •
Sommaire:
VidEcran!
VidRect! 1,1,40,6,4
Deplace! 9,4
ParamT! 1,7,4,0,0,0
Affiche! " Répondeur télématiq
  ue"
GOSUB spCadresChoix
ParamT! 0,7,0,0,0,0
Deplace! 8,11
Affiche! "Laisser un message"
Deplace! 8,14
Affiche! "Fonctions de service
  "
Deplace! 8,17
Affiche! "Quitter ce répondeur
  "
Deplace! 15,22
Affiche! "Votre choix :   + "
Inverse!
Affiche! " Envoi "
BoucleChoix:
Chaine$=""
Accept! 29,22,0,"",1,Chaine$,D
  r
ON Dr+1 GOTO DeconnexionF,Cho
  ixImp2,Page2,Sommaire,GuideI
  mp2,BoucleChoix,BoucleChoix,
  ChoixImp2,Deconnexion
GuideImp2:
GOSUB GuideIndisponible
GOTO BoucleChoix
ChoixImp2:
IF Chaine$<"1" OR Chaine$>"3"
  THEN GOSUB ChoixImpossible
  :GOTO BoucleChoix
ON VAL(Chaine$) GOTO EnrMess
  age,Service,Quitte

'• Enregistrement du message •
EnrMessage:
VidEcran!
Deplace! 3,2
Inverse!
ParamT! 0,7,4,0,0,0
Affiche! "  Enregistrement de
  votre message  "
Cadre! 2,1,38,3,0,0,2,0
Deplace! 1,5
Affiche! "Votre nom : ........
  ........ +"
Ligne!
Affiche! " "
Inverse!
Affiche! " Suite "
Deplace! 1,6
Affiche! "   Prénom : ........
  ........ +"
Ligne!
Affiche! " "
Inverse!
Affiche! " Suite "
Deplace! 1,7
Affiche! "Téléphone : ........
  ........ +"
Ligne!
Affiche! " "
Inverse!
AfficheR! " Suite "
Chariot!
Normal!
Mode! 2
Affiche! "_____Message_____"
FOR Index=11 TO 17
Deplace! 1,Index
Affiche! "..................
  ........ +"
Ligne!
Affiche! " "
Inverse!
Affiche! " Suite "
NEXT
Deplace! 1,20
Affiche! "Pour valider votre m
  essage :"
```

```
Ligne!
Affiche! " "
Inverse!
Affiche! "  Envoi    "
Deplace! 13,21
Affiche! "Pour l'annuler :"
Ligne!
Affiche! " "
Inverse!
Affiche! " Sommaire "
Deplace! 13,5
AfficheR! Nom$
Deplace! 13,6
AfficheR! Prenom$
Deplace! 13,7
AfficheR! Telephone$
FOR Index=11 TO 17
Deplace! 1,Index
AfficheR! Message$(Index-11)
NEXT
ON Pointeur GOTO SaisieNom,Sa
  isiePrenom,SaisieTelephone,S
  aisieMessage

SaisieNom:
Pointeur=1
Accept! 13,5,0,"",16,Nom$,Dr
ON Dr+1 GOTO DeconnexionF,ErS
  aisieNom,Inact3,EnrMessage,G
  uideImp3,Inact3,Sommaire,ErS
  aisieNom,Deconnexion
Inact3:
GOSUB spToucheInactive
GOTO SaisieNom
GuideImp3:
GOSUB GuideIndisponible
GOTO SaisieNom
ErSaisieNom:
IF Nom$="" THEN Alerte! "Nom
  indispensable...",1:GOTO Sai
  sieNom
IF Dr=1 GOTO Validation
SaisiePrenom:
Pointeur=2
Accept! 13,6,0,"",16,Prenom$,D
  r
ON Dr+1 GOTO DeconnexionF,Val
  idation,SaisieNom,EnrMessage
  ,GuideImp4,Inact4,Sommaire,S
  aisieTelephone,Deconnexion

Inact4:
GOSUB spToucheInactive
GOTO SaisiePrenom
GuideImp4:
GOSUB GuideIndisponible
GOTO SaisiePrenom
SaisieTelephone:
Pointeur=3
Accept! 13,7,0,"",16,Telephone
  $,Dr
ON Dr+1 GOTO DeconnexionF,Val
  idation,SaisiePrenom,EnrMess
  age,GuideImp5,Inact5,Sommair
  e,SaisieMessage,Deconnexion
Inact5:
GOSUB spToucheInactive
GOTO SaisieTelephone
GuideImp5:
GOSUB GuideIndisponible
GOTO SaisieTelephone
SaisieMessage:
LigneMessage=0
SaisieMessage2:
NumLigne=LigneMessage+11
Pointeur=4
Deplace! 1,NumLigne
Accept! 1,NumLigne,0,"",28,Mes
  sage$(LigneMessage),Dr
ON Dr+1 GOTO DeconnexionF,Val
  idation,SaisieMessage3,EnrMe
  ssage,GuideImp6,Inact6,Somma
  ire,SaisieMessage4,Deconnexi
  on
GuideImp6:
GOSUB GuideIndisponible
GOTO SaisieMessage2
Inact6:
GOSUB spToucheInactive
GOTO SaisieMessage2
SaisieMessage4:
LigneMessage=LigneMessage+1
IF LigneMessage-7 THEN Valida
  tion
GOTO SaisieMessage2

SaisieMessage3:
LigneMessage=LigneMessage-1
IF LigneMessage<>-1 THEN Sais
  ieMessage2
LigneMessage=0:GOTO SaisieTele
  phone

'• Écran de validation •
Validation:
VidEcran!
Deplace! 1,3
AfficheR! "Voici votre message
   tel qu'il sera"
AfficheR! "enregistré :"
Deplace! 4,7
Affiche! "   Nom : "+Nom$
Deplace! 4,8
Affiche! "Prénom : "+Prenom$
Deplace! 4,9
Affiche! "Téléph : "+Telephone
  $
FOR Index=0 TO 6
Deplace! 6,11+Index
AfficheR! Message$(Index)
NEXT
Deplace! 1,21
Affiche! "Pour le valider  ...
  ...."
Ligne!
Affiche! " "
Inverse!
AfficheR! "   Envoi    "
Normal!
Affiche! "Pour le modifier ...
  ...."
Ligne!
Affiche! " "
Inverse!
AfficheR! " Correction "
Normal!
Affiche! "Pour l'annuler   ...
  ...."
Ligne!
Affiche! " "
Inverse!
AfficheR! " Annulation "
Normal!
AttenteValidation:
AcceptF! Dr
ON Dr+1 GOTO DeconnexionF,Enr
  egistrement,EnrMessage,Valid
  ation,GuideImp7,Sommaire,Som
  maire,ChoixImp7,Deconnexion,
  EnrMessage
ChoixImp7:
Alerte! "Touche interdite",0
GOTO AttenteValidation
GuideImp7:
GOSUB GuideIndisponible
GOTO AttenteValidation

'• Enregistrement •
Enregistrement:
VidEcran!
Deplace! 6,12
Affiche! "Enregistrement en co
  urs"
FIELD 1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
NombreMessage=CVI(zNombre$)
NombreMessage=NombreMessage+1
LSET zNombre$=MKI$(NombreMess
  age)
PUT 1,1
FIELD 1,16 AS uN$,16 AS uP$,1
  6 AS uT$,28 AS u1$,28 AS u2
  $,28 AS u3$,28 AS u4$,28 AS
   u5$,28 AS u6$,28 AS u7$
LSET uN$=Nom$
LSET uP$=Prenom$
LSET uT$=Telephone$
LSET u1$=Message$(0)
LSET u2$=Message$(1)
LSET u3$=Message$(2)
LSET u4$=Message$(3)
LSET u5$=Message$(4)
LSET u6$=Message$(5)
LSET u7$=Message$(6)
PUT 1,NombreMessage+1
FOR Index=0 TO 6
 Message$(Index)=""
NEXT
'• Dernier écran •
VidEcran!
Deplace! 6,12
AfficheR! "Votre message est e
  nregistré..."
Deplace! 11,14
Affiche! "...faites "
Inverse!
```

```
Affiche! " Sommaire "
AcceptF! Dr
IF Dr=Faux THEN DeconnexionF
IF Dr=8 THEN Deconnexion
Pointeur=4:GOTO Sommaire

'• Fonctions de service •
Service:
VidEcran!
VidRect! 1,1,40,6,4
Deplace! 10,4
ParamT! 1,7,4,0,0,0
Affiche! " Fonctions de servic
  e"
ParamT! 0,7,0,0,0,0
Deplace! 32,12
Affiche! "+ "
Inverse!
Affiche! " Envoi "
Deplace! 15,15
Affiche! "Pour renoncer : "
Inverse!
```

```
Affiche! " Sommaire "
AttenteMot:
MotPasse$=""
Secret! 9,12,0,"Mot de passe :
   ",7,MotPasse$,Dr

ON Dr+1 GOTO DeconnexionF,Att
  enteMot2,Sommaire,Service,Gu
  ideImp8,Inact8,Sommaire,Atte
  nteMot2,Deconnexion
GuideImp8:
GOSUB GuideIndisponible
GOTO AttenteMot
Inact8:
GOSUB spToucheInactive
GOTO AttenteMot
AttenteMot2:
IF MotPasse$=Mot$ GOTO Attent
  eMot3
Essai=Essai+1
IF Essai=1 THEN Alerte! "Mot
  de passe erroné",1:GOTO Atte
  nteMot
IF Essai=2 THEN Alerte! "Erre
  ur, encore un Essai",1:GOTO
  AttenteMot
VidEcran!
Deplace! 10,12
Affiche! "Désolé, déconnexion
  forcée..."
GOTO Deconnexion
AttenteMot3:
Essai=0
VidLigne! 12,0
VidLigne! 15,0
GOSUB spCadresChoix
Deplace! 8,11
Affiche! "Lire les messages"
Deplace! 8,14
Affiche! "Effacer le fichier"
Deplace! 8,17
Affiche! "Changer le mot de pa
  sse"
Deplace! 15,22
Affiche! "Votre choix :   + "
Inverse!
Affiche! " Envoi "
AttenteService:
Chaine$=""
AttenteService2:
Chaine$=""
Accept! 29,22,0,"",1,Chaine$,D
  r
ON Dr+1 GOTO DeconnexionF,Att
  enteService3,Service,Inact9,
  GuideImp9,Inact9,Sommaire,At
  tenteService3,Deconnexion
Inact9:
GOSUB spToucheInactive
GOTO AttenteService2
GuideImp9:
GOSUB GuideIndisponible
GOTO AttenteService2
AttenteService3:
IF Chaine$<"1" OR Chaine$>"3"
  THEN GOSUB ChoixImpossible:
  GOTO AttenteService
ON VAL(Chaine$) GOTO Lecture
  ,Effacer,ChangeMot

'• Lecture du fichier •
Lecture:
FIELD 1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
NombreMessage=CVI(zNombre$)
IF NombreMessage=0 THEN Alert
  e! "Aucun message reçu",1:GO
  TO AttenteService2
NumMessage=NombreMessage+1
FIELD 1,16 AS uN$,16 AS uP$,1
  6 AS uT$,28 AS u1$,28 AS u2
  $,28 AS u3$,28 AS u4$,28 AS
   u5$,28 AS u6$,28 AS u7$
Lecture3:
VidEcran!
ParamT! 0,2,0,0,0,0
```

Jusqu'à 72 points avec MacWrite

Le nouveau 'système' (4.2) est arrivé avec les polices LaserWriter *Courier*, *Helvetica* et *Times* (aussi utilisables avec une ImageWriter) en tailles 27, 30, 36, 42, 54 et 72. Malheureusement, le menu 'style' de MacWrite n'autorise que les tailles allant de 9 à 24 points. C'est pourquoi, afin de détourner le problème, nous avons créé avec un éditeur de fichiers un document MacWrite baptisé "MacWrite 9 à 72" et contenant des caractères en tailles 9 à 72. Ces caractères utilisent la police *Times* mais, ici, peu importe la police, c'est la taille qui compte. Par exemple, si vous voulez insérer dans un document le titre "Section 1" en *Courier 72* :

- fermez votre fichier ;
- ouvrez le fichier *MacWrite 9 à 72* ;
- sélectionnez et copier *72* (ou seulement le 7 ou le 2) ;
- fermez le fichier *MacWrite 9 à 72* ;
- ouvrez votre fichier ;
- coller *72* à l'endroit où vous voulez placer votre titre ;
- taper *Section 1* ;
- sélectionnez *Section 1* ;
- changer la police (*Courier* dans le menu Caractères) ;
- supprimez *72*.

Le fichier *Write 9 à 72* est sur la disquette d'accompagnement de ce numéro.

9 10 12 14

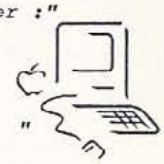
18 24 27 30

36 42 54 72

```
Affiche! "_"+CHR$(18)+CHR$(64
  +39)
Deplace! 2,3
Mode! 1
AfficheR! "Consultation des me
  ssages enregistrés"
Mode! 0
ParamT! 0,2,0,0,0,0
Affiche! "~"+CHR$(18)+CHR$(64
  +39)
Deplace! 9,5
ParamT! 0,6,0,0,0,0
Affiche! "Précédent :"
Ligne!
Affiche! " "
Inverse!
Affiche! "  RETOUR  "
Deplace! 11,6
ParamT! 0,6,0,0,0,0
Affiche! "Suivant :"
Ligne!
Affiche! " "
Inverse!
Affiche! "  SUITE   "
Deplace! 15,7
ParamT! 0,6,0,0,0,0
Affiche! "Fin :"
Ligne!
Affiche! " "
Inverse!
AfficheR! " SOMMAIRE "
ParamT! 0,2,0,0,0,0
AfficheR! "_"+CHR$(18)+CHR$(6
  4+39)
Affiche! " "+CHR$(18)+CHR$(64
  +39)
Deplace! 1,12
ParamT! 0,4,0,0,0,0
Affiche! "`"+CHR$(18)+CHR$(64
  +39)
Deplace! 1,16
ParamT! 0,4,0,0,0,0
Affiche! "`"+CHR$(18)+CHR$(64
  +39)
Deplace! 1,24
ParamT! 0,2,0,0,0,0
Affiche! "`"+CHR$(18)+CHR$(64
  +39)
Lecture2:
GET 1,NumMessage
Nom$=uN$
Prenom$=uP$
Telephone$=uT$
Message$(0)=u1$
Message$(1)=u2$
Message$(2)=u3$
Message$(3)=u4$
Message$(4)=u5$
Message$(5)=u6$
Message$(6)=u7$
Deplace! 2,10
Affiche! " Message n°" + STR$(
  NumMessage-1)
Deplace! 2,13
Affiche! "``> "+Nom$
Deplace! 2,14
Affiche! "``> "+Prenom$
Deplace! 2,15
Affiche! "``> "+Telephone$
FOR Index=1 TO 7
Deplace! 2,16+Index
Affiche! "``` "+Message$(Index
  -1)
NEXT
AttenteLecture:
AcceptF! Dr
ON Dr+1 GOTO DeconnexionF,Ina
  ct10,AttenteLectureR,Lecture
  3,GuideImp10,Inact10,Sommair
  eBis,AttenteLectureS,Deconne
  xion,Inact10
GuideImp10:
GOSUB GuideIndisponible
GOTO AttenteLecture
SommaireBis:
GOSUB VideVariables
GOTO Sommaire
Inact10:
GOSUB spToucheInactive
GOTO AttenteLecture
AttenteLectureR:'Retour
NumMessage=NumMessage-1
IF NumMessage>1 GOTO Lecture2
NumMessage=2:Alerte! "C'est le
   premier message",1:GOTO Att
  enteLecture
AttenteLectureS:'Suite
NumMessage=NumMessage+1
IF NumMessage<=NombreMessage+1
   GOTO Lecture2
NumMessage=NombreMessage+1:Ale
  rte! "C'est le dernier",1:GO
  TO AttenteLecture

'• Effacer le fichier •
Effacer:
VidEcran!
VidRect! 1,1,40,6,4
Deplace! 10,4
ParamT! 1,7,4,0,0,0
Affiche! " Effacement fichier
  "
ParamT! 0,7,0,0,0,0
Deplace! 5,10
AfficheR! "Cette commande supp
  rime l'ensemble"
Deplace! 5,11
AfficheR! "des messages stokés
  ."
Deplace! 5,13
AfficheR! "On ne peut annuler
  un effacement..."
R$=""
Deplace! 5,22
Affiche! "Pour effacer  :"
Ligne!
Affiche! " "
Inverse!
Affiche! " OUI + Envoi "
Deplace! 5,23
Affiche! "Pour renoncer :"
Ligne!
Affiche! " "
Inverse!
Affiche! "  Sommaire  "
AttenteEfface:
Reponse$=""
Accept! 10,16,0,"Ok pour tout
  effacer ? ",3,Reponse$,Dr
ON Dr+1 GOTO DeconnexionF,Att
  enteEfface2,Inact11,Effacer,
  GuideImp11,Sommaire,Sommaire
  ,Inact11,Deconnexion
Inact11:
GOSUB spToucheInactive
GOTO AttenteEfface
GuideImp11:
GOSUB GuideIndisponible
GOTO AttenteEfface
AttenteEfface2:
IF Reponse$<>"OUI" THEN Repon
  se$="":Alerte! "Réponse inco
  rrecte",0:GOTO AttenteEfface
FIELD#1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
LSET zNombre$=MKI$(0)
PUT 1,1
VidEcran!
Deplace! 6,12
AfficheR! "Le fichier a été ré
  initialisé..."
Deplace! 11,14
Affiche! "...faites "
Inverse!
Affiche! " Sommaire "
AcceptF! Dr
IF Dr=Faux THEN DeconnexionF
IF Dr=8 THEN Deconnexion
GOTO Sommaire

'• Changer le mot de passe •
ChangeMot:
VidEcran!
VidRect! 1,1,40,6,4
Deplace! 14,4
ParamT! 1,7,4,0,0,0
Affiche! "Mot de passe"
ParamT! 0,7,0,0,0,0
Deplace! 1,23
Affiche! "Pour renoncer, faite
  s : "
Inverse!
Affiche! " Sommaire "
Deplace! 1,11
Affiche! "Taper le nouveau mot
   de passe + "
Inverse!
Affiche! " Envoi "
Deplace! 1,16
Affiche! "Taper le à nouveau p
  our contrôle"
Cadre! 25,12,35,14,2,0,5,0
Cadre! 25,17,35,19,2,0,5,0
```



```
VerifMot:
Reponse2$=""
Secret! 27,13,0,"",7,Reponse2$
  ,Dr
ON Dr+1 GOTO DeconnexionF,Ver
  ifMot2,Inact12,ChangeMot,Gui
  deImpl2,Inact12,Sommaire,Ver
  ifMot2,Deconnexion
Inact12:
GOSUB spToucheInactive
GOTO VerifMot
GuideImpl2:
GOSUB GuideIndisponible
GOTO VerifMot
VerifMot2:
IF LEN (Reponse2$) < 4 THEN
  Alerte! "Minimum 4 caractère
  s",1:GOTO VerifMot
VerifMotBis:
Reponse3$=""
Secret! 27,18,0,"",7,Reponse3$
  ,Dr
ON Dr+1 GOTO DeconnexionF,Ver
  ifMotBis2,Inact13,ChangeMot,
  GuideImpl3,Inact13,Sommaire,
  VerifMotBis2,Deconnexion
Inact13:
GOSUB spToucheInactive
GOTO VerifMotBis
GuideImpl3:
GOSUB GuideIndisponible
GOTO VerifMotBis
VerifMotBis2:
IF Reponse2$<>Reponse3$ THEN
  Alerte! "Les deux mots de pa
  sse diffèrent",1:GOTO VerifM
  ot
Mot$=Reponse2$
FIELD 1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
LSET zMot$=Mot$
PUT 1,1
VidEcran!
Deplace! 6,12
AfficheR! "Le mot de passe a é
  té changé...  "
Deplace! 11,14
Affiche! "...faites "
Inverse!
Affiche! " Sommaire "
AcceptF! Dr
IF Dr=Faux THEN DeconnexionF
IF Dr=8 THEN Deconnexion
GOTO Sommaire
'• sous-programmes divers •
spToucheInactive:
Alerte! "Touche inactive ici",
  Faux
RETURN

GuideIndisponible:
Alerte! "Guide indisponible",F
  aux
RETURN
```

```
spCadresChoix:
Cadre! 4,10,6,12,2,0,5,0
Affiche! "1"
Cadre! 4,13,6,15,2,0,5,0
Affiche! "2"
Cadre! 4,16,6,18,2,0,5,0
Affiche! "3"
RETURN
ChoixImpossible:
Alerte! "Choix impossible",1
RETURN
spAfficheEtat:
PRINT ChaineEtat$
PRINT
RETURN
OuvreFichier:
Fichier$=FILES$(1,"PRép")
IF LEN(Fichier$) THEN Dr=Vra
  i ELSE Dr=Faux
RETURN
ArretServeur:
DrArret=Vrai
RETURN
VideVariables:
Essai=0
Pointeur=0
Nom$=""
Prenom$=""
Telephone$=""
FOR Index=0 TO 6
Message$(Index)=""
NEXT
RETURN
Souris:
Appel! Dr
RETURN

'• Fin et déconnexion •
Quitte:
VidEcran!
Deplace! 10,12
Affiche! "Au revoir..."
GOTO Deconnexion
DeconnexionF:
VidEcran!
Deplace! 3,12
Flash!
Affiche! "Minitel inactif depu
  is 2 minutes,"
Deplace! 9,14
Flash!
Affiche! "Déconnexion forcée."
Deconnexion:
DeCnx!
CLOSE
RESET
GOTO RepriseAttente
Erreur:
RESUME Erreur2
Erreur2:
VidEcran!
Affiche! "Désolé, problème ser
  veur"
GOTO Deconnexion
```

```
'• Ouverture d'un nouveau fich
  ier •
NouveauFichier:
DrInverse=Faux
WINDOW 2,"",(116,44)-(379,275
  ),-2
TEXTFONT 0:TEXTSIZE 12
PENPAT VARPTR(Gris(0))
MOVETO 8,120
LINETO 255,120
MOVETO 8,200
LINETO 255,200
PENNORMAL
MOVETO 16,16
PRINT "Mot de passe : de 4 à
  7 caractères"
MOVETO 16,32
PRINT "(A -> Z et 0 -> 9)."
MOVETO 16,56
PRINT "Nom : jusqu'à 16 carac
  tères"
MOVETO 16,72
PRINT "affichables par tous l
  es Minitel."
MOVETO 16,96
PRINT "Téléphone : jusqu'à 16
   caractères"
MOVETO 16,112
PRINT "affichables par tous l
  es Minitel."
MOVETO 56,141
PRINT "Mot de passe"
MOVETO 16,165
PRINT "Nom"
MOVETO 16,189
PRINT "Tél."
EDIT FIELD 1,"",(160,129)-(2
  48,144)
EDIT FIELD 2,"",(54,153)-(24
  8,168)
EDIT FIELD 3,"",(54,177)-(24
  8,192)
EDIT FIELD 1
NumEdit=1
BUTTON 1,1,"Annuler",(16,208)
  -(128,224)
BUTTON 2,1,"OK",(136,208)-(24
  8,224)
ON DIALOG GOSUB DialogueNou
  veau:DIALOG ON
DrSortie=Faux

BoucleD:
IF DrSortie THEN SortieNouvea
  u
GOTO BoucleD
SortieNouveau:
DIALOG OFF
WINDOW CLOSE 2
IF NOT Dr THEN RETURN
Fichier$=FILES$(0,"Nom du fich
  ier serveur :")
IF LEN(Fichier$) THEN Dr=Vra
  i ELSE Dr=Faux:RETURN
OPEN"R",1,Fichier$,244
```

```
FIELD#1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
LSET zNombre$=MKI$(0)
LSET zMot$=Mot$
LSET zNom$=NomServeur$
LSET zTel$=TelephoneServeur$
PUT 1,1
CLOSE
NAME Fichier$ AS Fichier$,"PR
  ép"
RETURN
DialogueNouveau:
DIALOG OFF
Dial0=DIALOG(0)
ON Dial0 GOSUB BoutonD,EditD,
  RienD,RienD,RienD,ReturnD,Re
  turnD
DIALOG ON
RETURN
BoutonD:
IF DrInverse THEN GOSUB Sign
  ale:DrInverse=Faux
Dial1=DIALOG(1)
IF Dial1=1 THEN Dr=Faux:DrSor
  tie=Vrai:RETURN
Mot$=EDIT$(1)
NomServeur$=EDIT$(2)
TelephoneServeur$=EDIT$(3)
LMot=LEN(Mot$)
IF LMot<4 OR LMot>7 THEN BEE
  P:DrInverse=1:GOSUB Signale
  :EDIT FIELD 1,Mot$,(160,12
  9)-(248,144):DrSortie=Faux:R
  ETURN
LNom=LEN(NomServeur$)
IF LNom<1 OR LNom>16 THEN BE
  EP:DrInverse=2:GOSUB Signal
  e:EDIT FIELD 2,NomServeur$
  ,(54,153)-(248,168):DrSortie
  =Faux:RETURN
LTel=LEN(TelephoneServeur$)
IF LTel<1 OR LTel>16 THEN BE
  EP:DrInverse=3:GOSUB Signal
  e:EDIT FIELD 3,TelephoneSe
  rveur$,(54,177)-(248,192):Dr
  Sortie=Faux:RETURN
DrErreur=Faux
FOR Index=1 TO LMot
C$=MID$(Mot$,Index,1)
IF (C$>"/" AND C$<":") OR (C$
  >"@" AND C$<"]") THEN testO
  K
DrErreur=Vrai
testOK:
NEXT
IF DrErreur THEN BEEP:DrInve
  rse=1:GOSUB Signale:EDIT F
  IELD 1,Mot$,(160,129)-(248,1
  44):DrSortie=Faux:RETURN
DrSortie=Vrai:Dr=Vrai
RETURN
EditD:
IF DrInverse THEN GOSUB Sign
  ale:DrInverse=Faux
NumEdit=DIALOG(2)
EDIT FIELD NumEdit
RETURN
RienD:
RETURN
Bloque:
RETURN
ReturnD:
IF DrInverse THEN GOSUB Sign
  ale:DrInverse=Faux
NumEdit=NumEdit+1
IF NumEdit=4 THEN NumEdit=1
EDIT FIELD NumEdit
RETURN
Signale:
Rect(top)=4+(DrInverse-1)*40
Rect(left)=8
Rect(bottom)=36+(DrInverse-1)*
  40
Rect(right)=255
INVERTROUNDRECT VARPTR(Rec
  t(top)),4,4
RETURN
'• Lecture des fiches sur le M
  acintosh •
Lire:
Fichier$=FILES$(1,"PRép")
IF Fichier$="" THEN RETURN
MENU 1,1,0
MENU 1,2,0
MENU 1,3,0
OPEN "R",1,Fichier$,244
FIELD 1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
NombreMessage=CVI(zNombre$)
IF NombreMessage=0 THEN CLOS
  E:GOSUB DialMessage0:GOTO
   RetourLire
GOSUB DialAffiche:CLOSE:GOTO
   RetourLire
RetourLire:
MENU 1,1,1
MENU 1,2,1
MENU 1,3,1
RETURN
'• Dialogue pas de message •
DialMessage0:
WINDOW 2,"",(100,120)-(411,19
  1),-2
TEXTFONT 0:TEXTSIZE 12
BEEP
MOVETO 16,24
PRINT "Ce fichier ne contient
   pas de message."
BUTTON 1,1,"OK",(192,40)-(296
  ,56)
Rect(top)=36
Rect(left)=188
Rect(bottom)=60
Rect(right)=300
PENSIZE 3,3
FRAMEROUNDRECT VARPTR(Rect
  (top)),12,12
PENNORMAL
Dlog=Faux
WHILE Dlog<>1 AND Dlog<>6
Dlog=DIALOG(0)
WEND
WINDOW CLOSE 2
RETURN
'• Dialogue message enregistré
  s •
DialAffiche:
WINDOW 2,"",(160,60)-(351,283
  ),2
TEXTFONT 0:TEXTSIZE 12
BUTTON 1,1,"Précédent",(16,16
  8)-(88,184)
IF NombreMessage=1 THEN BUTT
  ON 1,0
BUTTON 2,0,"Suivant",(104,168
  )-(176,184)
BUTTON 3,1,"Vider",(16,200)-(
  56,216)
BUTTON 4,1,"Annuler",(72,200)
  -(176,216)
TEXTFONT 4:TEXTSIZE 9:TEXT
  MODE 0
xNumMessage$=""
xNom$=""
xPrenom$=""
xTelephone$=""
FOR Index=0 TO 6
xLigneMessage$(Index)=""
NEXT
GOSUB MaJ
NumMessage=NombreMessage+1
NombreMessage$=STR$(NombreMess
  age)
IF LEFT$(NombreMessage$,1)="
  " THEN NombreMessage$=RIGHT
  $(NombreMessage$,LEN(NombreM
  essage$)-1)
FIELD 1,16 AS uN$,16 AS uP$,1
  6 AS uT$,28 AS u1$,28 AS u2
  $,28 AS u3$,28 AS u4$,28 AS
   u5$,28 AS u6$,28 AS u7$
LectureMac:
GET 1,NumMessage
xNom$=uN$
xPrenom$=uP$
xTelephone$=uT$
xLigneMessage$(0)=u1$
xLigneMessage$(1)=u2$
xLigneMessage$(2)=u3$
xLigneMessage$(3)=u4$
xLigneMessage$(4)=u5$
xLigneMessage$(5)=u6$
xLigneMessage$(6)=u7$
xNumMessage$="Message"+STR$(Nu
  mMessage-1)+"/"+NombreMessag
  e$
GOSUB MaJBis
BDial25:
Dlog=Faux
WHILE Dlog<>1 AND Dlog<>5
Dlog=DIALOG(0)
```

```
WEND
IF Dlog=5 THEN GOSUB MaJ:GO
  TO BDial25
Dlog=DIALOG(1)
ON Dlog GOTO Precedent,Suivan
  t,Vider,Annuler
Precedent:
IF NumMessage=2 THEN BDial25
NumMessage=NumMessage-1
IF NumMessage=2 THEN IF BUTT
  ON(1) THEN BUTTON 1,0
IF NumMessage<NombreMessage+1
  THEN IF BUTTON(2)=0 THEN
  BUTTON 2,1
GOTO LectureMac
Suivant:
IF NumMessage=NombreMessage+1
  THEN BDial25
NumMessage=NumMessage+1
IF NumMessage=NombreMessage+1
  THEN IF BUTTON(2) THEN B
  UTTON 2,0
IF NumMessage>2 THEN IF BUTT
  ON(1)=0 THEN BUTTON 1,1
GOTO LectureMac
Vider:
GOSUB Confirmation
IF NOT Dr THEN LectureMac
Dr=Faux
Annuler:
WINDOW CLOSE 2
RETURN
MaJ:
PENPAT VARPTR(Gris(0))
MOVETO 8,24
LINETO 184,24
MOVETO 8,72
LINETO 184,72
MOVETO 8,160
LINETO 184,160
MOVETO 8,192
LINETO 184,192
MaJBis:
MOVETO 12,16
PRINT xNumMessage$
MOVETO 12,40
PRINT xNom$
MOVETO 12,51
PRINT xPrenom$
MOVETO 12,62
PRINT xTelephone$
FOR Index=0 TO 6
MOVETO 12,88+11*Index
PRINT xLigneMessage$(Index)
NEXT
RETURN
'• Dialogue confirmation •
Confirmation:
WINDOW 1,"",(100,120)-(411,19
  1+16),-2
TEXTFONT 0:TEXTSIZE 12
C$=Fichier$
BoucleConfirm:
DP=INSTR(C$,":")
IF DP THEN C$=RIGHT$(C$,LEN
  (C$)-DP):GOTO BoucleConfirm
BEEP
MOVETO 16,24
PRINT "Voulez-vous effacer le
   contenu du fichier"
MOVETO 16,40
PRINT "'" C$ "' ?"
BUTTON 1,1,"Annuler",(72,56)-
  (176,72)
BUTTON 2,1,"OK",(192,56)-(296
  ,72)
Rect(top)=52
Rect(left)=68
Rect(bottom)=76
Rect(right)=180
PENSIZE 3,3
FRAMEROUNDRECT VARPTR(Rect
  (top)),12,12
PENNORMAL
Dlog=Faux
WHILE Dlog<>1 AND Dlog<>6
Dlog=DIALOG(0)
WEND
IF Dlog=6 THEN Dr=Faux:GOTO
  SortieConfirm
Dlog=DIALOG(1)
IF Dlog=1 THEN Dr=Faux:GOTO
  SortieConfirm
FIELD#1,2 AS zNombre$,8 AS zM
  ot$,16 AS zNom$,16 AS zTel$
GET 1,1
LSET zNombre$=MKI$(0)
PUT 1,1
CLOSE
Dr=Vrai
SortieConfirm:
WINDOW CLOSE 1
RETURN
'• Initialisations •
Initialisations:
FOR Index=0 TO 3
Gris(Index)=&HAA55
NEXT
Faux=0
Vrai=-1
top=0
left=1
bottom=2
right=3
MENU 1,0,1,"Fichier"
MENU 1,1,1,"Démarrer avec un f
  ichier existant…"
MENU 1,2,1,"Démarrer avec un n
  ouveau fichier…"
MENU 1,3,1,"Lire les messages
  enregistrés…"
MENU 1,4,0," "
MENU 1,5,1,"Quitter"
MENU 2,0,1,"Pom's"
MENU 2,1,1,"© 1988 JLB, CP &
  Éditions MEV — 39.51.24.43"
WINDOW CLOSE 1
RETURN
```

A l'ouverture de la nouvelle version de ce gestionnaire, le fichier se présente sous la forme d'un tableau : chaque colonne représente un champ, chaque ligne une fiche. Comme sur Works, on peut obtenir une présentation par fiches (l'ensemble d'une fiche apparaît à l'écran). Ces deux modes de présentation ne sont pas paramétrables et la seule police disponible est le Monaco 9. Le manuel d'utilisation est sobre lui aussi : les cents premières pages survolent tous les aspects du logiciel, permettant un rapide apprentissage de base. Pour le reste il faudra se plonger dans les 200 autres pages, arides mais efficaces.

Sobriété, tel semble être le mot clé pour définir l'aspect d'OverVue 2.1. S'il fallait choisir un mot pour en décrire les fonctionnalités, efficacité serait sans doute le plus approprié. En effet, OverVue 2.1 gère les fiches entièrement et uniquement en mémoire vive. Cela lui confère une rapidité impressionnante dans les opérations de tri. Mais cette rapidité est utilisée également pour effectuer un contrôle précis de la saisie, au moyen de trois fonctions astucieuses : *Non-Duplicatas* permet d'avertir l'utilisateur lorsqu'une même valeur est introduite deux fois dans une colonne, *Non-Unique* avertit l'utilisateur s'il vient d'entrer une valeur qui n'existait pas dans la colonne. Enfin, *Clairvoyance* permet, au cours de la saisie, de laisser OverVue "deviner" ce que vous êtes en train de taper en fonction de ce qui existe déjà dans la colonne. Il termine ainsi la frappe pour vous. Parfois, si plusieurs valeurs ont le même début, deux ou trois lettres s'imposeront pour assurer cette "clairvoyance". Une fois qu'OverVue a affiché vidéo inverse la valeur qu'il a deviné, vous pouvez soit la valider soit poursuivre normalement votre frappe pour un autre mot. Ces trois fonctions se montrent extrêmement confortables et accélèrent réellement la saisie, sans ralentir sensiblement le fonctionnement du programme.

Essai : OverVue 2.1

S. Dedeyan & G. Lejeune

Mais, plus encore, il est possible, pour une colonne donnée, de définir un format précis, par exemple la position des parenthèses et tirets s'il s'agit de noter des numéros de téléphones. Par la suite, il n'y aura plus qu'à saisir les chiffres. Bien sûr, si cette colonne a été cochée comme *Non-Unique* et numérique, on évitera de nombreuses erreurs de saisie. En cas de données fortement répétitives, comme les jours de la semaine, on peut établir un masque de saisie à choix multiples grâce a la fonction Barre de Saisie.

La gestion en mémoire vive offre d'autres atouts. Ainsi, OverVue se passe très bien de la fonction *Index*, présente sur toutes les bases de données conventionnelles. À quoi bon définir des priorités dans les recherches puisque même les plus gros fichiers sont triés en quelques secondes. Il existe cependant un petit inconvénient à cela : les tris ne portant que sur une colonne à la fois, il faut ensuite effectuer un autre tri sur le critère suivant, le premier ordonnancement n'étant pas oublié. En plus du tri, OverVue propose le regroupement des fiches, une fois triées. Par exemple, chaque ville d'un fichier d'adresses crée un groupe qui sera séparé des autres par une fiche spéciale appelée *sommaire*. Ce sommaire permet d'effectuer des calculs dans chaque groupe (moyenne, somme, mini, maxi, compteur...), mais se comporte comme une fiche ordinaire pour les tris et sélection. Ces possibilités sont souvent offertes dans les autres logiciels avec le module "rapports", rarement à l'écran. Les possibilités de recherche et de sélection sont tout à fait classiques et rien ne manque.

OverVue présente en outre deux caractéristiques qui le rapprochent du tableur : un nombre important de fonctions de calcul et la possibilité de représenter les données sous forme de graphiques. On peut ainsi remplir une colonne d'une valeur fixe ou d'une série, ajouter les valeurs de deux colonnes dans une troisième, multiplier ce résultat par trois, le diviser par 2 et envoyer le tout dans la colonne suivante. Le tout peut être assorti de fonctions logiques. Certes, il n'y a pas de fonctions mathématiques et financières, mais l'accent a été mis là où il le faut pour un fichier : l'alphabet. Une douzaine de fonctions sur les chaînes de caractères sont en effet disponibles (extraction, repérages de chiffres, conversion majuscule/minuscule, rotation...). Le module graphique est simple mais il est impossible d'imprimer un graphique à partir d'OverVue, il faudra le transférer via l'album.

On touche là un point crucial du logiciel. En effet, OverVue n'utilise pas l'ImageWriter ni la LaserWriter en mode graphique. Il utilise le propre générateur de caractères de l'imprimante. Ainsi, OverVue se limite aux seuls caractères romains sans italique ni gras. Les rapports imprimés permettent soit une simple copie de tableau (jusqu'à 200 caractères de large en ultra-condensé), soit du cousu-main. Dans ce cas, on se rapproche de l'emploi de masques des bases de données classiques : on crée des cases correspondantes aux colonnes choisies, on en détermine la longueur et la place sur le rapport, on ajoute du texte libre ça et là, on peut dater... rien que de très classique. Les fiches de sommaire s'intercalent à loisir. Il est possible de mémoriser huit formats de rapport. Avec de l'habitude, les rapports d'OverVue sont présentables, malgré leur limitation à un seul type de caractère. Notons qu'OverVue peut imprimer sur quatre colonnes et accepte toutes les hauteurs de papier; c'est donc un précieux générateur d'étiquettes.

La fusion de fichier est ce qui rend OverVue relationnel. Les deux fichiers peuvent être juxtaposés, chacun gardant ses caractéristiques propres, ou superposés s'ils sont du même type. On peut aussi insérer un fichier entier au milieu d'un autre grâce à la fonction *coller*. Enfin la fonction *Lier* permet une réelle fusion relationnelle : on précise quelle est la colonne de chaque fichier qui contient les données de type commun, puis quel nom donner à la colonne nouvelle qui accueillera le résultat de la fusion. Ces opérations sont sans danger dans la mesure où les deux fichiers d'origine subsistent toujours. Le seul problème que l'on peut rencontrer ici est la place disponible en mémoire vive.

OverVue dispose en outre d'un éditeur de macros-commande très complet. Chaque fichier peut gérer jusqu'à vingt macros. Une fois écrites et titrées elles apparaissent dans le menu *Do* et peuvent être lancée par un raccourci clavier. La définition de ces macros ne s'opère pas par enregistrement d'une séquence de commandes mais par un éditeur de texte, ce qui rend leur réalisation plus difficile. Elles peuvent effectuer toutes les fonctions d'OverVue sauf les entrées/sorties Finder, elles peuvent s'interrompre pour demander à l'utilisateur d'entrer une réponse. On dispose là d'un véritable langage de programmation.

En ce qui concerne l'ouverture sur l'extérieur, OverVue est plus doué pour l'Importation que pour l'Exportation. L'exportation n'est en effet possible que par l'album, le presse-papiers et l'écriture d'un rapport sur disque et non sur l'imprimante. Pour l'importation, les formats texte (avec choix de trois séparateurs : virgule, tabulation et retour-chariot), Sylk et Dif sont acceptés.

En définitive, OverVue est une gestion de fichier originale et simple d'emploi. Elle est particulièrement efficace et puissante, grâce à la gestion en mémoire vive. C'est pour l'instant le seul programme à proposer des fonctions aussi étendues pour le contrôle de la saisie. Ses macros, en combinant des fonctions astucieuses permettent de récupérer des heures de saisie et d'en économiser autant. OverVue apparaît presque comme un logiciel intégré qui serait orienté vers la base de données (contrairement à Excel qui lui est orienté vers le tableur). Cependant deux points faibles viennent ternir le tableau : l'aspect trop spartiate de la présentation à l'écran et surtout à l'impression ainsi que l'absence de fonction "Annuler". OverVue est remarquablement efficace pour les bases de données de taille moyenne ne nécessitant pas des rapports à la présentation irréprochable. Elle est agréable à utiliser mais donne un sentiment de frustration devant l'absence de certaines fonctionnalités primordiales comme l'impression soignée.

Apple // for ever

Jean-Yves Bourdin

Nouvelles du front

Il y a de l'angoisse dans l'air chez les propriétaires d'Apple // : l'abandon désormais officiel par Apple de la production du //e et du //c signifie-t-il la condamnation de l'Apple // ?

Du point de vue d'Apple, les choses sont claires : Apple est une entreprise dont toute la force repose dans l'innovation (technologique et logicielle). Quelles que soient les réticences de chacun d'entre nous au changement, nous avons donné de l'argent à Apple, et cet argent est utilisé par lui pour rester à la tête de l'innovation en matière d'informatique personnelle. Nul n'ignore que ni le Mac II ni le GS n'ont été précédés par une quelconque étude de marché, pas plus que l'Apple][de Wozniak ne l'avait été. Nous faisons de bons produits, et le marché suivra : c'est ainsi qu'Apple raisonne depuis Wozniak – avec quelques échecs (Apple III, Lisa) mais avec principalement, ma foi, pas mal de réussite.

Il est clair que la technologie 8 bits seule est une technologie dépassée. Ne nous voilons pas la face : le point d'entrée dans la gamme Apple, le bas de gamme d'Apple, c'est le GS.

Est-ce donc la fin de l'Apple // ? Évidemment pas. D'abord, parce qu'il y a trois millions d'Apple // installés dans le monde : c'est plus que toute autre machine. C'est un énorme marché. Ensuite parce que le GS est un Apple // à part entière. Le GS a été produit par Apple sur la demande des utilisateurs qui voulaient une machine à la fois moderne et compatible. Cette machine, j'en suis persuadé, a de gigantesques possibilités (avez-vous épluché le moniteur du GS ? Ses possibilités sonores ?). Les programmes, les cartes, les interfaces pour le GS commencent à foisonner. Le 65832 (processeur 32 bits compatible patte-à-patte avec le 65816 du GS) est en cours de production. Les accélérateurs GS vont bientôt sortir. Et ces machines 32 bits accélérées continueront à faire tourner l'Integer Basic et à répandre la "philosophie Wozniak". Bref : Apple // for ever.

D'autant que la technologie Apple 8 bits n'est pas morte. Faire tourner AppleWorks avec un bureau de 1 Méga suffit à 80% des besoins d'un utilisateur débutant. Simplement, l'Apple 8 bits passe quasiment dans le domaine public, je veux dire dans la production Taïwanaise à bas prix, rejoignant les compatibles IBM. Le successeur de l'Apple //c se trouve désormais dans les supermarchés des USA : un ordinateur compatible //e 65C02, avec 1 Méga de Ram, souris, sortie RVB, drive 5,25' intégré, acceptant les disques 3,5', avec pavé numérique, avec tous les interfaces du //c plus l'interface parallèle et un port d'extension, muni d'un processeur à 3,6 Mégaherz (plus rapide que le GS). C'est le **LASER 128 EX** de **Video Technology**, produit en Chine, vendu aux USA par **Central Point Software** (qui produit Copy][+ pour le // et le Mac). Son prix : 500 dollars, soit moins de 3000 francs. À ce prix, on a désormais un Apple // presque comme on a le téléphone. Apple // for everyone...

Si Apple abandonne le //c, il n'abandonne pas ses utilisateurs : il a créé aux États-Unis avec la coopérative **A.P.P.L.E.** une "Apple Programmers and Developers Association" (APDA) qui fournit à tous, et pas seulement aux développeurs patentés, les informations, documentations, outils de développement etc., nécessaires pour programmer sur les machines Apple. Les plus grands développeurs en sont membres tout aussi bien que le *bidouilleur* de banlieue. Les outils sont vendus à prix coûtant, et bien entendu sans aucune garantie, puisqu'il s'agit parfois de versions préalables, de *pre-Release*, de documentations provisoires polycopiées, qu'Apple diffuse dès qu'il le peut auprès de ceux qui en ont besoin. Cette politique d'ouverture maximale semblerait, à première vue, donner des armes à la concurrence : elle a, en fait, toujours fait le succès d'Apple. Apple France s'y met aussi : après le serveur Apple (en 36.14), voici **D.D.A., Documentations Développeurs Apple**, version française de l'APDA. C'est auprès d'elle que tout un chacun peut se procurer les documentations techniques sur le GS ou sur le Mac.

Toujours plus : un IBM dans l'Apple ?

Avec la carte **PC Transporter d'Applied Engineering**, le MS-Dos devient pour l'Apple // ce qu'était le CP/M auparavant : un deuxième système d'exploitation pour l'Apple. Cette carte n'est pas un émulateur, mais un véritable IBM dans l'Apple : elle a nécessité un énorme effort de développement et d'investissement. Son prix justifie une étude un peu serrée de ses performances.

Tout d'abord, en dehors de toute utilisation du MS-Dos, cette carte, qui entre dans n'importe quel slot sauf le slot 3 ou le slot auxiliaire, est à la fois une extension mémoire type Apple (768Ko) et une carte contrôleur de drives universelle pour l'Apple : elle pilote aussi bien les lecteurs GS 3,5' (mais pas les Unidisk //c et //c) que les drives 5,25' 40 pistes double face type IBM vendus par Applied Engineering (elle formate les disquettes 5,25' en 360Ko sous ProDOS), et elle permet de chaîner cinq lecteurs en tout.

Bien entendu, c'est aussi un compatible PC : microprocesseur Nec V30 à 7,16 MHz (trois fois plus rapide que l'IBM), 640Ko de Ram possibles en mode IBM, tous les modes vidéo CGA sur votre moniteur Apple, coprocesseur arithmétique en option, bref on peut faire tourner sur l'Apple l'adaptation sous MS-Dos de Visicalc appelée Lotus 1-2-3.

Les avantages principaux de cette carte sont bien entendu dans l'intégration qui permet d'utiliser tous les périphériques de l'Apple : carte horloge, imprimante, clavier du //e-IIGS (les touches de fonction IBM étant remplacées par des combinaisons avec les touches), souris, moniteur vidéo, son. Vous pouvez stocker des

données et programmes MS-Dos sur tout disque formaté en ProDOS (5,25', 3,5', disque dur, etc.). La carte permet même de lire et écrire des disquettes 3,5' formatées en MS-Dos avec un simple lecteur 3,5' Apple type IIGS.

Mais... il y a ce que les publicités ne disent pas : la carte PC Transporter ne permet pas de formater en MS-Dos les disquettes 3,5' avec un lecteur Apple 3,5' ni de démarrer directement ces disquettes. Il vous faudra donc avec la carte le duo-disk 5,25' type IBM vendu par Applied Engineering pour démarrer le MS-Dos. Et il est indispensable d'avoir accès à un compatible IBM muni d'un lecteur 3,5' pour formater en MS-Dos les disquettes 3,5'.

Faites les comptes : la carte avec 640Ko, plus le kit d'installation, plus le Duo-Disk 5,25', tout cela fait 1058 dollars aux prix d'Applied Engineering pour mettre le MS-Dos dans le GS. Rajoutons le fait que la carte en action tire 600 mA, et 1075 mA avec le coprocesseur arithmétique, et il est pratiquement indispensable de changer l'alimentation du //e pour celle plus puissante (110 et 220 V) vendue 69 dollars par Applied Engineering. C'est en fait le prix d'une configuration équivalente à base de *compatible-lambda made in China* vendu dans les supermarchés. L'avantage est évidemment de n'avoir qu'une seule machine sur son bureau, une seule imprimante, etc., et de pouvoir échanger les fichiers entre les systèmes d'exploitation.

...ou un Apple dans l'IBM ?

Plutôt que de mettre à vos frais dans votre Apple l'ordinateur du patron, il serait peut-être malin de faire mettre à ses frais un Apple dans son IBM : la carte **Trackstar 128** de **Compatible Peripherals** met un //e 65C02 128K dans un IBM ou compatible. Elle utilise les drives 5,25' IBM pour lire les disques Apple, y compris les disques protégés (sauf ceux qui utilisent les demi-pistes). Elle utilise les périphériques de l'IBM pour l'Apple, sauf le joystick (port pour joystick Apple sur la carte), elle permet les transferts de fichiers ProDOS/MS-Dos. Et elle ne coûte que 300 dollars. Mais ce n'est pas un GS, et vous n'aurez pas les caractères accentués français sur l'écran.

...ou une machine d'avenir ?

À chacun donc de faire ses choix. Le mien est ni l'un ni l'autre : un ordinateur dans un autre est une machine bloquée, qui ne pourra plus évoluer, et aussi bien l'IBM PC que l'Apple //c sont des machines technologiquement dépassées, d'ailleurs abandonnées toutes deux par leurs constructeurs respectifs. Ce ne sont pas des investissements d'aujourd'hui.

J'avoue également n'éprouver qu'une jouissance très modérée à la compilation en DBase III. Et plutôt que d'utiliser Lotus 1-2-3 ou Word Perfect sur IBM, essayez donc **Vip Professional** de **Vip Technologies**, parfait clone de Lotus 1-2-3, mais avec l'interface graphique-souris du GS, ou... **Word Perfect** de **Word Perfect Corporation** sur GS. Ces deux programmes existent d'ailleurs aussi pour //c-//c (Vip Professional nécessitant une extension mémoire type Ramworks ou Multiram). Peut-être la carte **PC Transporter** serait-elle un bon moyen de mettre à niveau un //e qui n'aurait que 128Ko et des lecteurs 5,25' : mais c'est le même prix qu'une unité centrale de GS. La seule machine qui peut concurrencer un GS, c'est le Mac.

Bon, je sais, le patron... Eh bien, faîtes-lui faire un tour au Sicob ou à la prochaine Apple Expo. Une entreprise ne peut pas se permettre longtemps sans dommage économique ce qui peut au contraire être l'intérêt bien compris du particulier : utiliser une informatique obsolète.

Le mange-disques

Depuis le temps qu'on l'attendait ! Ça y est, il est arrivé, il est là : **Merlin 8/16** de **Glen Bredon** chez **Roger Wagner Publications** est sorti ! Enfin la possibilité de programmer en assembleur de façon pratique pour le GS, finie la galère d'APW/ORCA ! Pourquoi ne pas en parler dans la rubrique "16 bits" ? Mais parce qu'il marche aussi sur le //c et le //e : vous avez bien lu, non seulement le Merlin 8, dernière version du Merlin-Pro sous Dos et ProDOS, mais également le Merlin 16 bits, tout cela marche sur tous les Apple // d'aujourd'hui. Bien entendu, il vous faut précisément un Apple // 88 : microprocesseur 65802, lecteur 3.5, mémoire. Voir la rubrique "sous le capot".

Entre autres, cela signifie qu'il est sous ProDOS 8 et ne s'occupe pas plus qu'AppleWorks du graphisme GS. Faudrait-il en conclure que même Glen Bredon estime que le ProDOS 16 n'est pas très au point, et que le graphisme GS ne fait que ralentir les choses dans les applications sérieuses ? Mais non voyons, puisque précisément il vous permet d'écrire quant à vous des applications sous ProDOS 16, des accessoires de bureau, de vous régaler avec les outils du GS. Un éditeur de liens plus rapide produisant du code relogeable, un éditeur plein écran, des bibliothèques de sous-routines, des bibliothèques de macros (incluant des macros d'accès aux outils GS), que nous faut-il de plus ?

Eh bien, un assembleur et un éditeur de liens co-résidents en mémoire avec l'éditeur, qui relient et assemblent jusqu'à 12 000 lignes de code par minute, sans qu'il soit nécessaire de passer par des modules séparés (suivez mon regard...). Et aussi un Sourceror 16 bits, ce magnifique désassembleur qui a autant fait pour l'Apple que le Locksmith (à ce propos : la dernière version du **Locksmith, 6.0 Révision C**, d'**Alpha Logic Business Systems**, fonctionne sur le GS). Sans compter la compatibilité avec tous les sources Merlin existants, et la possibilité de charger et assembler les fichiers APW avec un minimum de travail d'édition.

Voilà le genre d'outils qui remet les montres à l'heure et les choses à leur place : le langage privilégié de l'Apple //, c'est l'assembleur, n'en

déplaise aux adeptes d'autres jargons. C'est lui qui a fait et qui fera les plus grands programmes et les plus grands succès de l'Apple //. Apple // for ever, cela passe par Merlin for ever. Avec Merlin 16, TML Pascal et TML Basic, l'avenir du GS est assuré.

Documax, de Signum Microsystems, est exactement le genre de programmes qui me pousse à écrire dans ces colonnes. Un coût ridicule (25 dollars avec la documentation sur le disque), une interface utilisateur parfaite, implantable, pas de protection, des possibilités extraordinaires, et... personne ne le connaît ! Que fait-il ? D'abord, il compresse à 50% les fichiers textes, à 40% les fichiers traitement de texte et base de données d'AppleWorks, et les décompresse ensuite bien sûr. En quelques communications avec InterPom's, je le rembourse intégralement. Au lieu de copier un fichier pour le ranger, compressez-le : cela n'est pas plus long, et de toute façon Documax vous permet de lire ces fichiers et de travailler dessus. Étant sous ProDOS 8, il tourne sur //e, //c, IIGS et accepte bien entendu tous les disques reconnus par ProDOS.

Ensuite, il permet de chercher une référence, un nom, un morceau de phrase, un numéro de téléphone, dans l'ensemble de tous les fichiers texte ou AppleWorks que vous lui indiquez, compressés ou non (bien entendu, il s'occupe lui-même des catalogues ProDOS : vous n'avez jamais à taper un nom de fichier). Il vous crée si vous le voulez un fichier de références sur n'importe quel sujet avec nom de fichier, texte recherché avec son contexte, numéro de ligne. Vous faites ensuite de ce fichier une base de données, ou bien vous coupez et coller l'information dans d'autres fichiers. Il segmente et fusionne les fichiers à volonté. Il gère les catalogues ProDOS, y compris l'ordre des fichiers dans le catalogue. Il est sain de temps en temps de nous rappeler que l'informatique ne sert pas à cliquer des souris, mais à traiter et manipuler à notre guise l'information. Allez, chiche : trouvez le nom et le prix d'un programme qui en fasse autant sur une autre machine !

Sous le capot

Pour un prix inférieur à celui d'un lecteur 3,5', vous pouvez aujourd'hui mettre un disque dur 20 Mégas dans votre Apple. Vous avez remarqué que les clones IBM sont munis de disques durs bon marché. Évidemment, un petit malin a réalisé une carte d'interface entre ces disques durs et l'Apple][+, //e, IIGS. Il s'agit de la carte **Megaboard de Perlin Electronics**. La carte coûte 200 dollars, et supporte ProDOS, DOS 3.3, Pascal, CP/M. Pour 500 dollars (2 750 F au cours d'aujourd'hui), vous avez la carte et un disque dur de 20 Mégas. Reconnaissons cependant que la carte Speedisk, si elle ne fait qu'un Méga, est beaucoup plus rapide, silencieuse et sans doute durable (aucun élément mécanique).

Savez-vous que vous pouvez pour 25 dollars mettre un processeur 16 bits dans votre //e ou //c ? Il s'agit du **microprocesseur 65802**, vendu par **Roger Wagner Publishing**, et produit par le **Western Design Center**, concepteur du 65816, le microprocesseur du GS. Ce microprocesseur reconnaît toutes les instructions et tous les modes d'adressage du 65816, a les mêmes modes émulation et natif, mais bien entendu ne fait pas l'adressage mémoire sur 32 bits par page de 64Ko, et ne vous met pas un GS dans le //c. Il fait tourner les programmes 8 bits existants, plus ceux spécialement écrits pour lui (un seul aujourd'hui : Merlin 8/16). Il suffit d'enlever le 65C02 et de mettre à la place le 65802.

L'autre solution consiste à acheter une carte accélérateur **Transwarp d'Applied Engineering** pour le //e, avec un 65802 intégré. **Roger Wagner** la vend pour 245 dollars, et votre //e tourne plus vite que le GS (si !). Évidemment, Zip-Chip (voir Pom's 33) est moins cher et ne prend aucun slot : mais ce n'est pas un 65802, il ne fait pas tourner Merlin 16.

Toujours pour mettre votre //e ou //c à niveau, il faut étendre sa mémoire. Si vous avez un nouveau //c, poussez-le à un Méga (voir Pom's 33). Si vous avez un ancien //c, vérifiez qu'il a les Roms pour le lecteur 3,5', sinon faites-le mettre à niveau par Apple. Pour l'étendre à un Méga, les seules solutions seront ensuite la carte **Z-Ram d'Applied Engineering** ou la carte **Multiram CX de Checkmate Technology** (sur lesquelles vous pouvez ajouter une carte 65816 pour faire tourner Merlin 16).

Si vous avez un //e, vous pouvez l'étendre avec une des cartes qui se mettent dans le slot auxiliaire (Ramworks d'Applied Engineering ou Multiram //e de Checkmate Technology). Ces cartes ont l'avantage de remplacer la carte 80 colonnes, donc de ne pas prendre un slot de plus. Elles ont l'inconvénient d'être technologiquement dépassées et de ne pas être portables sur le GS.

Le plus intéressant, ce sont sans doute les cartes au standard Apple ("Slinky"), qui se mettent dans n'importe quel slot du][+, //e, IIGS (sauf le slot 3). Elles sont portables sur le GS, sont reconnues par AppleWorks pour y étendre son bureau. Choisissez entre la carte Apple (chère, mais Apple), la **Ramfactor d'Applied Engineering** qui est la meilleure, et la carte **Sprintdisk d'AST** qu'on trouve à des prix bradés en ce moment (179 dollars par exemple avec un Méga installé chez Microtech Consulting Company).

L'autre complément d'un Apple // pour le mettre à niveau, c'est le lecteur 3,5'. Le //c est normalement muni de cette capacité. Pour le //c, il faut une carte d'interface. Apple nous propose sa carte pour l'Unidisk 3,5'. Mais ce lecteur étant assez cher, on peut lui préférer la carte **Universal Disk Controller de Central Point Software** pour][+, //e, IIGS. Cette carte a l'avantage d'accepter aussi bien les lecteurs 5,25' Apple ou compatibles, que les Unidisks 3,5' d'Apple, que les lecteurs 3,5' 800Ko compatibles Mac, qui sont très bon marché, que les lecteurs 3,5' simple face 400Ko des vieux Mac, qu'il faut laisser aux dinosaures. Pour 261 dollars (plus le port) chez **Triad Software**, mon fournisseur favori,

vous aurez une carte et un drive 800Ko qui lira les disquettes GS sans problème, fera tourner Merlin 16 et rentrera AppleWorks, les accessoires Pinpoint et les applications TimeOut sur la même disquette, et démarrera automatiquement sur un //e 65C02.

Cette carte a cependant quelques inconvénients : d'abord, même si elle peut théoriquement rentrer dans tous les slots sauf le slot 3 du //e, elle déborde beaucoup à droite. En conséquence, elle est en fait pratiquement réservée au slot 2 du //e et au slot 7, ce qui marche impeccablement sous ProDOS, mais pas sous Pascal. Ensuite, elle accepte les drives Apple 3,5' du GS, mais n'obéit pas à l'éjection manuelle (ce qui n'est pas grave : c'est si facile par programme). Enfin, elle ne reconnaît pas les disquettes protégées du GS (raison de plus pour ne jamais utiliser un programme protégé sur GS). Ajoutons que comme les drives 800Ko sont pilotés par la carte et pas par l'unité centrale, elle n'accepte pas les programmes qui, comme Diversi-Cache, modifient les routines de lecture-écriture. Bill Basham a découvert pas mal de bugs dans cette carte, mais ils ne vont jamais jusqu'à la perte d'un fichier en mémoire ou la ruine d'une disquette. C'est donc malgré tout la solution de loin la plus économique pour tout Apple II démuni de lecteur 3,5'.

Voilà donc votre //e ou //c enfin à niveau, vous avez un "Apple // 88". Ceci dit, faites vos comptes : une carte accélérateur, plus un 65802, plus une interface pour lecteur 3,5', plus une carte mémoire de 1 Méga, c'est vrai, ça ronfle, et c'est au point. Mais c'est très en-dessous d'un GS, et ça n'est pas un GS, pour un prix qui lui est bien voisin !

Patchworks

Voici comment raccourcir le temps de chargement d'AppleWorks 1.4 pour ceux qui le chargent depuis un disque 3.5, un dur, ou un disque Ram, et/ou qui ont une horloge dans l'Apple (GS et autres) :

```
UNLOCK APLWORKS.SYSTEM↵
BLOAD APLWORKS.SYSTEM, A$2000,
    TSYS↵
POKE 14477,44↵ (Plus de «Pressez
    une touche pour continuer»)
POKE 14157,208 : POKE 14158,
    19↵ (Ne demande plus la date)
BSAVE APLWORKS.SYSTEM, A$2000,
    T$FF↵
```

Le patch suivant permet aux heureux propriétaires d'un GS et d'une carte Ram 1 Méga type Apple "slinky" (cartes Apple, Ramfactor, AST Sprintdisk, etc.) de faire croire à AppleWorks 1.4 qu'il est sur un //e et non sur un GS, le faisant ainsi se charger et mettre son bureau dans cette carte, en n'occupant que le Bank 0 de la Ram GS.

```
UNLOCK APLWORKS.SYSTEM↵
BLOAD APLWORKS.SYSTEM, A$2000,
    T$FF↵
POKE 13609,44↵
BSAVE APLWORKS.SYSTEM, A$2000,
    T255↵
```

Regardez à cette occasion le code que vous changez, et vous verrez comment AppleWorks s'y prend pour reconnaître qu'il est sur un GS.

Voici comment (Merci Dominique Ottello) permettre à AppleWriter ProDOS (version française) de marcher avec le port imprimante intégré du GS en court-circuitant sa routine d'initialisation du port série. C'est donc à vous qu'il appartient de configurer ce port en passant par le tableau de bord (Ó-control-esc). Pour cela, il faut d'abord avoir remplacé le ProDOS de la disquette par un ProDOS 8 plus récent qui autorise les interruptions.

```
PREFIX/AW↵
BLOADAWD.SYS,T$C,A$2000↵
CALL-151↵
4D98:60↵
4F8A:10↵
4F91:13↵
BSAVE AWD.SYS,T$0C,A$2000↵
```

Encore une partie et j'éteins

Garder un vieux][+ avec un joystick pour les jeux des enfants n'est plus un bon calcul : les grands du jeu sur Apple (Broderbund, Epyx) sortent maintenant systématiquement de magnifiques jeux d'action (arcade et simulation), de superbes animations en Double Haute Résolution. Après **Air Heart** de **Broderbund**, voici **California Games** d'**Epyx**, qui vous permet de faire du surf, du vélo-cross, du skate-board, du "freesbee", etc. Sous l'Apple, la plage...

— Range-moi cette disquette loin d'ici, tu vois bien que je travaille !

Après les diverses séries de simulation des Jeux Olympiques d'hiver et d'été, la Double Haute Résolution va nous offrir des simulations encore plus réalistes, puisque nous allons refaire nos parties de sports des rues, là où nous jouons sans arbitre avec le copain qui ne sait pas rattraper une balle contre celui qui ne sait pas la lancer (**Street Sports Base-Ball, Basket-Ball**, etc., toujours d'**Epyx**).

— Ne touche pas à ce joystick et laisse-moi écrire mon article !

Bien entendu, la Super Haute Résolution du GS permet aussi de magnifiques animations et simulations sportives : essayez donc **World Games GS**, du même **Epyx**, décidément très en forme en ce moment. Mais cela n'empêche pas les français de travailler aussi à de bons jeux sur Apple // : on parle beaucoup d'une version française pour Apple // de l'**Arkanoïd** de l'Atari, et **Froggy Software** nous offre des aventures inédites avec **Le justicier du bahut.**

Les jeux de l'Arlésienne : **Ultima V** et **Wizardry IV**. Cela fait longtemps qu'on les attend, les publicités sont parues, mais, à l'heure où j'écris ces lignes, je ne connais personne qui ait réussi à acheter autre chose qu'une promesse de vente. Curieuse stratégie commerciale.

— Bon, puisque c'est comme ça, je te prends au surf et je pulvérise ton high-score ! Bon sang, où as-tu mis la disquette de California Games ? J'avais dit qu'elle devait toujours rester à côté de l'Apple !!!

16 Bits

Si vous avez un GS, prenez rendez-vous avec votre revendeur : il vous remettra la dernière version du système (3.1, avec le Finder) et vous mettra à niveau votre machine (nouvelles Roms, nouvelle chip vidéo) si son numéro de série est inférieur à C725XXXXX. Tout cela gratuitement. Mais bien entendu ne jetez rien : ni l'ancien système ni les anciennes Roms. Certains programmes ne marchent pas avec les nouvelles Roms. La solution : charger les anciennes Roms en Ram, changer les vecteurs. Lecteurs GSphiles, au travail! Nous comptons sur vous.

Question :

Qu'est-ce qu'un Macintosh ?

Réponse :

Un ordinateur assez ancien de la société Apple, en noir et blanc dans ses versions grand public, mais dont le mérite fut d'avoir permis la réalisation d'un assez grand nombre de dessins et de fontes graphiques pour le nouvel ordinateur couleur de cette société, l'Apple IIGS.

Explication :

Vous avez dû remarquer, au milieu du tas de polycopiés et de disquettes qui accompagnait la documentation technique du GS, une disquette un peu spéciale : elle ne marchait pas. C'est en effet une disquette pour Macintosh, destinée à vous permettre de récupérer sous ProDOS les fontes graphiques, les images MacPaint, et les fichiers data divers du Mac. Bien entendu, la conversion fonctionne aussi dans l'autre sens, mais les fichiers GSPaint ont des couleurs bien pâles sur un Mac Plus.

Quelques conseils pour piller un Macintosh : choisissez-le avec suffisamment de mémoire pour avoir un disque Ram conséquent. Emportez votre drive GS avec vous pour faire un deuxième drive au Mac s'il n'en a pas déjà un. Emportez un stock de disquettes formatées en ProDOS.

La disquette contient aussi une application pour le Mac appelée "Emulateur Apple II GS" : Pom's a obtenu d'Apple l'assurance que vous pouvez donner cette application aux possesseurs de Mac. Quand vous aurez fini de soutirer pour votre GS toute la substantifique moelle du Mac de votre ami ou de votre patron, remerciez-le en lui laissant cette application et regardez-le l'essayer...

Vous l'avez remarqué, si vous voulez reconfigurer votre disque Ram et faire prendre en compte cette modification immédiatement à votre machine, le seul moyen, après avoir fait le nécessaire avec le tableau de bord, est... de l'éteindre. Le GS refuse le moindre risque de perdre un fichier du disque Ram. Comme il est aussi protégé contre les micro-coupures d'électricité, vous avez le temps de prendre un café avant qu'il oublie réellement ce qu'il a en mémoire et que vous puissiez le rallumer. Heureusement il y a un autre moyen : faites ♁-control-option-reset, et l'autotest de votre machine démarre. Attendez un peu, puis faites ♁-control-reset pour redémarrer le disque. Cette fois-ci, le GS vous obéit. Puisqu'il y a déjà un redémarrage à chaud et un redémarrage à froid , ceci doit être un redémarrage glacial...

Roger Wagner m'a écrit trois pages pour expliquer les raisons et formes de sa protection de Softswitch. En gros, il s'agit d'éviter, s'agissant d'un programme que l'utilisateur met sur toutes ses disquettes, une diffusion ou un passage involontaires dans le domaine public. Tout est copiable, mais l'installation du programme met un signe dans votre machine pour qu'il ne puisse pas marcher sur une autre.

Ceci dit, la dernière parution de Roger Wagner, Merlin 8/16, est un programme normal (non protégé). Et si quelqu'un devait craindre le passage dans le domaine public, c'est bien **Bill Basham**, de **Diversified Software Research**, dont le **Diversi-Cache**, qui s'intègre carrément au ProDOS 8, accélère de façon considérable la lecture et l'écriture sur le disque 3,5'. Eh bien, ce programme est diffusé en Shareware, avec d'autres indispensables (**Diversi-Copy**, le plus rapide des copieurs 3,5', **Diversi-Key**, qui met des macros partout, **Diversi-Hack**, un mini-Softswitch) : vous pouvez copier et distribuer gratuitement la disquette (la documentation est sur le disque). Si le programme vous plaît, vous êtes invité à payer directement à l'auteur. J'ai acheté Diversi-Copy, ce qui prouve que l'idée est bonne. Pour Diversi-Cache, j'aimerais qu'un lecteur fasse la comparaison avec le programme de cache fourni avec la carte GS-Ram d'Applied Engineering.

La place me manque pour faire aujourd'hui un tour d'horizon complet des langages sur le GS (j'ai dénombré jusqu'ici, en ne comptant que les langages 16 bits, au moins trois Pascal, deux Forth, cinq Basic, un C, plus les assembleurs). Disons seulement qu'Apple a fait son travail en sortant son interpréteur **GSBasic**, disponible chez **DDA**, fort ressemblant au Business Basic de l'Apple ///. Et parlons de l'autre. Vous qui commenciez à vous mettre au Pascal à cause de TML Pascal, mais qui regrettiez de ne pouvoir vraiment parler en Basic à votre machine, réjouissez-vous : le **TML Basic** est en vente. Attention : ce n'est pas un simple compilateur qui viendrait en complément d'un interpréteur, comme c'était le cas avec l'Applesoft. S'il est vrai qu'il compile les programmes du GSBasic, et que la plupart des programmes 100% Applesoft seront compilables après un petit nombre de modifications, TML Basic est un ensemble de programmation complet.

Sachant que TML Basic compile très vite, en mémoire, et lance aussitôt le programme compilé, sachant que son debugger intégré vous renvoie, en cas d'erreur, immédiatement dans l'éditeur, à la ligne fautive de votre programme toujours en mémoire, avec un message d'erreur explicite, la différence avec un interpréteur devient presque invisible pour le programmeur. Il n'y a que la vitesse d'exécution du programme qui change, mais de façon faramineuse. Bien entendu, il donne accès aux outils de la Rom, permet la programmation structurée et la récursivité, etc. L'interface avec le **TML Speech Toolkit** permet de faire parler le GS. La documentation est excellente. C'est un TML, et c'est un GS : donc, souris, couleurs, couper-coller à

travers quatre programmes chacun dans sa fenêtre, etc. De quoi plaindre sincèrement les pauvres programmeurs MS-DOS, même turbo.

À essayer : la carte **Memory Saver** de **Checkmate Technology**, qui pour 150 dollars transforme le disque Ram de n'importe quelle carte mémoire du slot auxiliaire du GS en disque Rom. Si vous avez essayé **Movie Studio** et **VS/Com** de **VersionSoft**, écrivez-moi. Même chose pour **Gribouille GS** de **Madeleine Hodé**.

À lire

Les ouvrages sur le GS commencent à foisonner. Il y a bien entendu de tout. Pour ma part, je vous conseille les suivants :

- Roger Wagner : **Apple IIGS, Machine Language for Beginners**, chez Roger Wagner Publishing ;
- David Eyes et Ron Lichty : **Programming the 65816 including the 6502**, chez Brady Book / Prentice Hall Press ;
- Gary Little : **Exploring the Apple IIGS**, chez Addison-Wesley.
- Michael Fischer : **Apple IIGS Technical Reference**, chez Osborne-McGraw Hill.
- Gary Bond : **Inside the Apple IIGS**, chez Sybex.

D'autre part, Apple publie peu à peu chez **Addison-Wesley** ses différents manuels pour le GS. Ce sont de vrais livres, bien plus utilisables que les milliers de pages photocopiées dont il fallait se contenter jusqu'ici. Votre patience sera récompensée si vous avez attendu pour acheter:

- Apple Computer : **Technical Introduction to the Apple IIGS**, chez Addison-Wesley.
- Apple Computer : **Apple IIGS Firmware Reference**, chez Addison-Wesley.
- Apple Computer : **Apple IIGS Hardware Reference**, chez Addison-Wesley.
- Apple Computer : **ProDOS 16 Reference Manual**, chez Addison-Wesley.

DDA devrait pouvoir fournir certains de ces ouvrages : demandez-lui son catalogue. Sinon, essayez avec l'APDA. Sauf pour **Sybex**, les éditeurs ne diffusent pas directement.

Pom's a découvert l'existence d'un sympathique *fanzine* de bonne facture pour les toqués de l'Apple // et du Mac, un produit du Sud et de l'association **Sun Data** appelé "Ampersand". Aux rubriques habituelles des fanzines (solutions de jeux, comment "protéger" vos disquettes), il joint des informations, des revues de logiciels, et des articles d'initiation. Il propose un catalogue de "Freeware" à la carte pour GS et Mac, et offre aux Marseillais l'accès à un serveur gratuit (91 79 30 60, code "&").

Pom's a également reçu le faire-part de naissance d'un futur confrère, une revue française spéciale GS et Mac : **GS Magazine**. Il est prévu que chaque numéro coûte 50 F, et soit accompagné d'une disquette 800Ko (en kiosque, ce sera une disquette Mac, que vous échangerez par la poste avec la disquette GS. Par abonnement, vous choisirez entre disquette GS et Mac, mais vous aurez le droit de l'échanger contre l'autre). La disquette contiendra des programmes en domaine public, Shareware, et des démonstrations de logiciels. La revue devrait être orientée principalement sur les services à l'utilisateur et au consommateur. Baptême prévu à la mi-février.

Adresses

A.P.D.A
290 S.W. 43rd Street, Renton,
WA 98055, USA.

A.P.P.L.E. - Coop
290 S.W. 43rd Street, Renton,
WA 98055, USA.

Alpha Logic Business Systems
4119 North Union Road, Woodstock,
IL 60098, USA.

Applied Engineering
P.O. Box 798, Carrollton,
TX 75006, USA.

Beagle Bros
6215 Ferris Square, Suite 100,
San Diego, CA 92121, USA.

Broderbund Software, Inc.
17 Paul Drive, San Rafael,
CA 94903, USA.

Central Point Software
9700 S.W. Capitol Highway,
#100, Portland, OR 97219, USA.

Checkmate Technology
509 South Rockford Drive, Tempe,
Arizona 85281, USA.

Compatibles peripherals, Inc.
6363 Taft Street, Suite 305,
Hollywood, Florida 33204, USA.

D.D.A.
Technopole - rue M. Faraday
78180 Montigny Le Bretonneux.
☎ (1) 30 45 26 62.

Diversified Software Research
34880 Bunker Hill, Farmington,
MI 48018-2728, USA.

Epyx, Inc.
600 Galveston Drive, Redwood City,
CA 94063, USA.

Froggy Software
33, avenue Philippe-Auguste
75011 Paris
☎ 43 58 25 98.

GS Magazine
320, rue Saint Honoré - 75001 Paris.

Gribouille
16, rue de Poules - 67000 Strasbourg.

Microtech Consulting Company
206 Angie Drive, Cedar Falls,
IA 50613, USA.

Mindscape, Inc.
3444 Dundee Road, Northbrook,
IL 60062, USA.

Perlin Electronics, Inc.
7394 Calle Real, Suite B, Goleta,
CA 93117, USA.

Pinpoint Publishing
5901 Christie Avenue, Emeryville,
CA 94608, USA.

Roger Wagner Publishing
1050 Pioneer Way, Suite P,
El Cajon, CA 92020, USA.

Signum Microsystems
120 Mountain Avenue, Bloomfield,
Connecticut 06002, USA.

Sun Data-Ampersand
16, rue Julia - 13005 Marseille

Sybex
6-8, impasse du Curé - 75881 Paris
Cedex 18 - ☎ 42 03 95 95

TML System
Customer Sales Department,
8837-B, Goodbys Executive Drive,

Jacksonville, Florida 32217, USA.

Triad Software
125 North Washington Street,
Papillion, NEbraska 68046, USA.

VersionSoft
94, rue Lauriston - 75116 Paris

☎ 47 27 71 72.

VIP Technologies
C/O ISD Marketing, 20 Steelcase
Road West, Markham
Ontario L3R 1D2, Canada.

Word Perfect Corporation
288 West Center Street, Orem
Utah 84057, USA.

Zip Chip
11926 Santa Monica Boulevard
Los Angeles, CA 90025, USA.

Bibliographie

Alexandre Avrane, Alexandre Duback

Systèmes d'exploitation et systèmes de protection sur Apple //, par Jean-Pierre Lagrange Édition Micro-application – 179 F

Qui n'a jamais rêvé de découvrir les secrets des protections de programmes. Voici donc un livre qui devrait assouvir l'appétit de découverte des chevronnés de l'Apple //. Que les éditeurs de logiciels soient néanmoins rassurés : il s'agit ici de montrer comment protéger les programmes et leurs disquettes ; il n'est jamais question de détailler le travail inverse. 'Déplombeurs' s'abstenir !

Avec un sujet de très haut niveau technique comme celui-ci, il est indispensable de posséder une excellente connaissance préalable de l'assembleur et des systèmes d'exploitation de l'Apple // (DOS 3.3 ou ProDOS). Il ne sera jamais question de faire un apprentissage de la pratique du 6502 : le début du livre démarre sur les chapeaux de roue et la suite continue à la même vitesse...

Au menu, on trouve d'abord les éditeurs habituels de disquette tels que CIA ou Watson. Suivent une description des méthodes 'préhistoriques' de protection en mémoire d'un programme (en commençant par rediriger les vecteurs du Reset). Puis, les protections sur disque sont progressivement détaillées : modifications du nombre de pistes, de secteurs, d'octets ou de nibbles précèdent des méthodes franchement abominables : nouvel encodage des nibbles, allongement des temps de synchronisation, synchronisations angulaires. Enfin, apparaissent les méthodes quasi-incopiables : interpistes ou spiraling par exemple.

Pour illustrer ces méthodes, l'auteur parsème les chapitres des listings des principales routines étudiées (à nouveau, seulement des routines de protection : aucun listing n'est fourni pour décrypter). Il ne reste plus au lecteur qu'à protéger ses programmes en créant un cocktail de sa composition pour faire amplement transpirer les héritiers d'Aldo Reset.

MatStatMath – Logma SA – 12, rue d'Anjou – 78000 Versailles – 450 F

Il ne s'agit pas là d'un livre mais d'une disquette proposée en annexe dans l'ouvrage *Excel efficace* publié chez *Cedic/nathan.*

MatStatMath regroupe un grand nombre de modèles de calcul utilisables avec Excel et destinés au étudiants, professeurs ou, plus généralement, tous ceux qui ont besoin de manipuler des statistiques ou nombres sous toutes leurs formes. Cette disquette contient également une police de caractères en taille 6 points, baptisée MicroCaractères.

MatStatMath est commercialisée par *Logma SA*, société qui distribue aussi les fantastiques cartes *Speedisk* pour Apple //.

Le livre de PostScript®, par Bernard-Paul Eminet – Édition du P.S.I – 200 F

Il ne faut pas rechercher dans cet ouvrage une description exhaustive des possibilités de PostScript. Il s'agit plus d'une présentation qui permet peut-être de débuter dans ce langage. Les exemples de ce livre sont pour l'essentiel tirés de l'excellent *PostScript Language – Tutorial and Cookbook* chez *Addison-Wesley Publishing Compagny* (fort beau livre, bien fait, réalisé avec goût et élégance dont le seul inconvénient est d'être en anglais).

Pour un livre qui traite du summum de la maîtrise de l'impression sur LaserWriter et autres imprimantes PostScript, on se demande pourquoi l'auteur (pourtant ingénieur chez Apple) liste des programmes sans montrer le résultat de leur exécution : avait-il une machine sous la main ? Cela ne simplifie pas la compréhension de l'ensemble.

On retrouve dans cet ouvrage, brutalement recopié, le programme paru dans Pom's 24 qui imprime joliment «PostScript» : aucune référence au nom de l'auteur, merci pour lui... Si l'on ajoute le mélange des polices de caractères et le manque de goût dans la mise en page et les croquis, voici un livre bien décevant.

ImageWriter™ II – Manuel de référence – Apple Computer France/InterÉditions

Comme toutes les documentations d'origine Apple, celle-ci est bien faite, claire et précise. De plus, les erreurs qui apparaissaient dans les éditions antérieures ont apparemment disparues. Cet ouvrage est donc indispensable lorsque l'on veut utiliser à fond l'imprimante ImageWriter, pratiquement toujours sous-employée.

Postscript

Boîte aux lettres

4) Transmission - 9 l.
De: Arian ZELWER (AZ10) - 11 jan 88 22h35

S'il vous plaît, une question :
- Ayant à côté du Mac+ un A//c avec lecteur 3,5 pouces, est-il possible de récupérer les fichiers textes - sous ProDOS/Appleworks - et les reprendre sous MacWrite ou Word ?
Puis-je échanger des fichiers par liaison directe entre ces deux machines ?

Merci de votre aide. *** AZ10 ***

Avec InterPom's sur le Mac et InterPom's sur le //c (et un câble de liaison ad hoc), vous pourrez transmettre les fichiers à 9600 bauds.
Si vous transmettez le fichiers AppleWorks, vous hériterez de codes de contrôle qu'il faudra éliminer en traitement de textes sur le Macintosh.
Un conseil, demander à AppleWorks l'«impression dans un fichier texte (ASCII) sur disque», puis transmettez ce fichier nouvellement créé : plus de codes de contrôle, peu de questions de présentation à régler à l'arrivée sur le Macintosh.

---

5) Bonjour et S.O.S. ... - 4l
De: Jean-Pierre BCJLETEX (JPB12) - 12 jan 88 08h31

J'ai un problème avec un fichier texte DOS 3.3 et ce message : Erreur code 107 ? Que doua-je faire ? Amitiés.
(PS) à quoi correspond le numéro de chaque disquette Mac dans le bon de commande ?

Erreur 107 : Bad Subscript error : vous tentez de sortir des limites d'un tableau ; par exemple, DIM A$(12) : PRINT A$(16). Devant une erreur dont on ne connait pas le code, le plus simple – à part consulter le manuel du Basic, du DOS ou de ProDOS – est de supprimer le ONERR qui récupère les erreurs. Un message en clair, et accessoirement en anglais, s'affichera.
Le numéro des disquettes Pom's représente le numéro de la revue à laquelle elles correspondent. Ainsi, la disquette 28 contient tous les fichiers listés dans la revue 28.

---

5) Date avec AppleWriter Pom's 33 - 9l
De: Jean-Marc MAYER (JMM13) - 02 jan 88 14h04

Je rencontre un petit problème avec "Date et AppleWriter" sur un GS.
Sans programme de boot qui met la date dans les octets $BF90-$BF91, l'utilitaire me dit : vous n'avez pas de carte horloge, ce qui est normal.
J'ai installé REBOOT.SYSTEM importé de CalvaCom mais lors d'un boot, j'ai toujours le message :
INSERER/REBOOT.
Que faire ?

En fait, vous n'avez pas besoin de REBOOT.SYSTEM puisque l'Apple IIGS comporte une carte horloge. C'est votre version de ProDOS qui ne la reconnaît pas : il faut installer la version 1.3 ou 1.4 de ProDOS 8 sur votre volume AppleWriter et tout rentrera dans l'ordre, les fichiers seront datés.

En ce qui concerne REBOOT, si vous préfériez cette solution, importez également MODIF.REBOOT qui permet de changer le nom de volume qu'attend cet utilitaire. Autre solution transitoire, renommer votre volume de travail /REBOOT.

---

3) Câble et Gestion bancaire (AR) - 35l
De:Thierry VICAIRE (TV10) - 30 dec 87 21h03

Je vous ennuie encore avec deux questions :
1) J'aurais besoin d'un câble de 3 m pour relier mon Minitel 1B à mon MacPlus : quel serait le prix ?
2) Concernant le programme de gestion de compte bancaire du numéro 20 de Pom's :
   - est-il compatible HFS ?
   - fonctionne-t-il obligatoirement avec Basic 2.0 ou son RunTime est-il suffisant ?
   - la ventilation des dépenses peut-elle servir à ventiler les recettes ?

1) 300,00 F franco
2) Ce programme ne fonctionnera normalement en HFS qu'avec la version 3.0 du Basic Microsoft.
- *Basic 2.0 ou simplement RunTime 2.0 conviennent mais pas en HFS.*
- *Non, seules les dépenses sont ventilées.*

---

14) Graphisme et assembleur *AR* - 91l.
De:Jean-Louis ROCHE (JLR10) - 27 dec 87 23h06

Problème :
Un petit bonhomme traverse l'écran ; 7 formes (shapes) définissent le mouvement et un petit programme Basic s'occupe de l'affichage. Cela fonctionne mais je me heurte à un clignotement de l'écran.
Même travail en assembleur, mais après de vains essais, je constate que "ça clignote toujours" ; on est loin du défilement harmonieux que j'attendais... J'en conclus qu'il y

a une astuce de programmation certainement bien connue depuis la préhistoire de l'informatique !

Nous ne publions pas le listing accompagnant ce message car la solution est effectivement connue :
Il suffit de tester l'octet $C019 et de n'afficher que lorsqu'il est positif, c'est-à-dire qu'il n'y a plus de balayage en cours.

```
...
$1   BIT  $C019
     BPL  $1
...routine d'affichage
```

Cette solution n'est valable qu'en assembleur ; pour une question de vitesse, pas question de tester en Basic ou même d'appeler depuis le Basic ce micro-programme avant l'XDRAW.

---

17) Clv'Pom_transfert - 8l
De: Alexis APLOGAN (AA11) - 29 dec 87 11h55

Avec Clv_pom's, je n'arrive pas à récupérer les applications Importés de CalvaCom. Lors de la transformation par BinHex 4, la lecture de l'application est interrompue par ERREUR CRC ($2A83).
Merci de votre réponse.

Il est probable que vous ayez importé votre application en mode "enregistrement de textes" ce qui a conduit à un recodage des caractères accentués, donc à une erreur de 'checksum'. Il convient de se mettre en mode "enregistrement de programmes".

---

10) Help Patchworks - 17l
De: Laurent AIUTI (LA24) - 26 dec 87 14h52

Réf. Pom's 33 Patchworks page 67
Le patch qui remplace le caractère "damier" par le caractère souris "return" sur AppleWorks 1.4 ne fonctionne pas, AppleWorks se plante et bloque le GS.
Pourtant j'ai exécuté à la lettre les indications.
Y a t-il une erreur dans les Pokes ?
Amicalement.

Que vous dire ? Non, il n'y a pas d'erreur, après exécution des pokes, nous obtenons le résultat escompté. Il ne peut que s'agir d'une erreur d'adresse lors de la manipulation.

---

6) Conseil... - 10l
De: François MULLER (FM17) - 11 dec 87 15h31

Je conseille aux utilisateurs du programme MinBas (Pom's 27) de faire la modification suivante s'ils ne l'ont déjà faite...

Ligne 795 ajouter CHR$(31) 9 après le CHR$(14).

Ceci provoquera un saut de 9 lignes sur votre IMW // et imprimera une page Minitel sur une page de papier , ce qui est plus pratique.
Amitiés ** FM **

Du côté des imprimantes LaserWriter

On parle d'une nouvelle version du PostScript intégré à la LaserWriter disponible aux États-Unis, mais quel est le numéro de version de votre imprimante ? Pour le savoir, envoyez-lui le programme listé ci-dessous. Il donne accessoirement le nombre de pages imprimées et le nom de l'imprimante ; c'est aussi l'occasion de voir des instructions non abordées dans le numéro 24 de Pom's.
L'envoi se fait depuis un Mac à l'aide de DownLoad ou JustText et depuis un Apple //, par le port série avec tout traitement de textes.

```
%! n° version PostScript
/Times-Roman findfont 12 scalefont setfont
100 700 moveto version (N\312 de version : ) show show
100 680 moveto 31 string
statusdict begin printername pagecount end
6 string cvs (Nombre de pages : ) show show
100 720 moveto (Imprimante : ) show show showpage
```

Trois nouvelles LaserWriter sont proposées par Apple : longévité triple, densité des noirs améliorée, cartouches plus durables, bac de 200 feuilles et, enfin !, un bac à enveloppes.

* *LaserWriter IISC, dédié à un seul Macintosh, pilotée par les routines QuickDraw dudit Mac : 4 polices, 1Mo Ram – 19 900 FHT*
* *LaserWriterIINT, dotée de la dernière version de PostScript (42 ?), rapide et partageable : 35 polices, 1Mo Rom, 2Mo Ram – 29 900 FHT.*
* *LaserWriter IINTX conçue avec un processeur 68020 à 16Mhz, 2 à 4 fois plus rapide que la Laser+. Extensions mémoire prévues : 35 polices, 1Mo Rom, jusqu'à 12Mo Ram – 39 900 FHT.*

Elles sont connectables à un IIGS ou à un PC à l'aide d'AppleTalk PC et un système de mise à niveau d'un modèle à l'autre est prévu.

Petites annonces

Ces petites annonces sont gratuites et réservées aux abonnés. Elles doivent bien entendu concerner l'informatique. Pour les ventes de logiciels, l'annonceur doit nous faire parvenir une photocopie de la facture d'achat.

Vends carte SSC (01/87) : 750 F — câble SSC/Minitel : 150 F — Disk][: 1000 F — carte horloge (driver ProDOS à adapter) : 100 F — collection Micro-Systèmes (20 à 72) : faire offre.
Jean-Marc Corazza - Le Téoulet - 81600 Gaillac ✆ 63 57 06 96

Vends imprimante Microline 80 parallèle connecteur Centronics : 1000 F.
Michel Le Port ✆ 47 20 81 88. Calva MLP10

Recherche Langage Fortran UCSD original pour Apple //.
André Moreau - 73, rue du Roleur - 59300 Valenciennes ✆ 27 45 16 12

Vends logiciels pour PC & compatibles sous emballages scellés, Multiplan 3, VP Expert : faire offre.
Daniel Ronxin - Laignelet - 35133 Fougères ✆ 99 99 33 71

Vends Apple //e 65C02 128Ko, Moniteur ambre, 2 lecteurs 140Ko, ventilateur, pavé numérique, joystick : 5000 F — Machine à écrire Underwood électronique à marguerite avec mémoire : 3500 F.
M. Calvet ✆ 45 97 44 82

Recherche Compilateur C Aztec d'origine, complet avec documentation pour Apple //e.
Norbert Steinberg - 24 avenue du Moulin - 78230 Le Pecq ✆ dom : 39 58 42 71 ✆ Bx 64 46 26 50

Contact Pour partager expérience en assembleur, prendre contact avec :
Emmanuel Bougeard - 3Bis boulevard des 3 croix - 35000 Rennes ✆ le week-end 99 59 37 06

Recherche programmes d'origine : The Bridge, MultiScribe 2.0, logiciels d'astrologie, Astronomie, Généalogie sur Apple //, Mac, IBM et tous logiciels de démonstration en version limitée.
Club informatique ForINext - 15 square Costes 91070 Bondoufle ✆ 64 97 67 09 CalvaCom CI10 & PB16

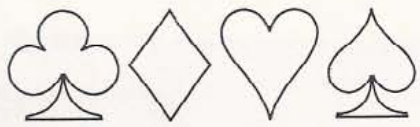
Vends machine à écrire IBM à sphères : 1500 F
Maurice Trecul - 3, boulevard de Verdun - Moigny sur école - 91490 Milly-la-Forêt ✆ (1) 64 98 42 35

Hasard...

Comme promis, nous avons tiré au sort des noms parmi les bulletins d'abonnement depuis parution du dernier numéro.

M. **Michel Cosset** de Caudebec en Caux à reçu un traitement de textes, M. **Jean Rumeau**, un traitement de textes doublé d'un programme de mailing, M. **Patrice Verdun** de Villiers le Mahieu, un traitement de textes, et M. **Bertrand Guillin** d'Épernay un tableur.

Félicitations et rendez-vous au prochain numéro pour le résultat de notre sondage et un autre tirage au sort parmi les abonnés (rappelons que nous offrons programmes de valeur et documentation techniques).



Solution du n° 33

P	E	T	I	T	M	A	T	I	N
R	O	U	T	I	N	I	E	R	E
A	L	T	E	R	E	R	■	R	I
E	I	R	■	O	S	■	T	A	G
S	E	I	G	N	E	U	R	I	E
I	N	C	A	S	■	L	A	S	■
D	■	E	L	■	A	T	M	O	S
I	I	■	B	O	N	I	■	N	O
U	N	I	E	M	E	M	E	N	T
M	O	L	E	S	T	A	T	E	S

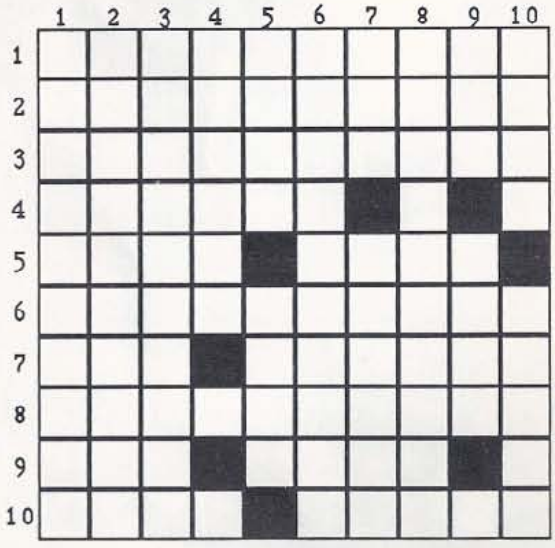
Horizontalement

1 - Courroies
2 - Tel Balzac
3 - Vraiment protestante
4 - Opaline
5 - Flotte - Bye-bye
6 - Il peut s'agir d'une zone
7 - Mal vieilli - Arc en ciel
8 - Empotés
9 - Va au fast foot - Débarrasée
10 - Dégoutter - Fins des fléaux

Verticalement

1 - Cuirs équestres
2 - Vit sur la Loire mais pas dans la Loire
3 - Vociférante
4 - Certaine argile
5 - Boue - Aufwiedersehen
6 - Peut tourner à la psychose
7 - Permet aux IBM d'y voir clair - Nacrés
8 - Retouchés
9 - Lunch - De moins en moins
10 - Exhaler - Chevilles

Problème 34 par Joëlle Piard



Nous avons découvert qu'on peut avoir le sens des affaires...

Avoir le sens des affaires est devenu monnaie courante.

Rien n'est plus banal que de commencer une entreprise à deux dans un garage, et de la retrouver 10 ans plus tard dans les 200 plus grandes entreprises américaines.

Ce qui est plus rare, c'est de conserver les vraies valeurs et d'avoir le sens de la famille.

De 1984 à 1987, Macintosh a beaucoup progressé.

Sa mémoire s'est considérablement étendue, de 1 à 8 méga-octets, ses menus se sont sophistiqués, hiérarchisés, sa bibliothèque de logiciels est devenue gigantesque.

Cependant, il n'a jamais oublié ce qu'il avait appris de plus important : l'homme.

Après Macintosh Plus et Macintosh SE, Macintosh II en est une nouvelle preuve.

et celui de la famille aussi.

Les menus déroulants sont toujours présents, bien qu'ils se déroulent beaucoup plus vite.

Les capacités graphiques sont toujours les mêmes à 16 millions de couleurs près.

C'est une machine toujours aussi intuitive, disposant de 1 à 8 mégaoctets de mémoire vive (et même plus), d'un disque dur interne de 20 à 80 mégaoctets, pour que toute la logique de votre travail y trouve son aise.

Ouvert à tous, Macintosh II est aujourd'hui ouvert à tout grâce à ses 6 connecteurs d'extension.

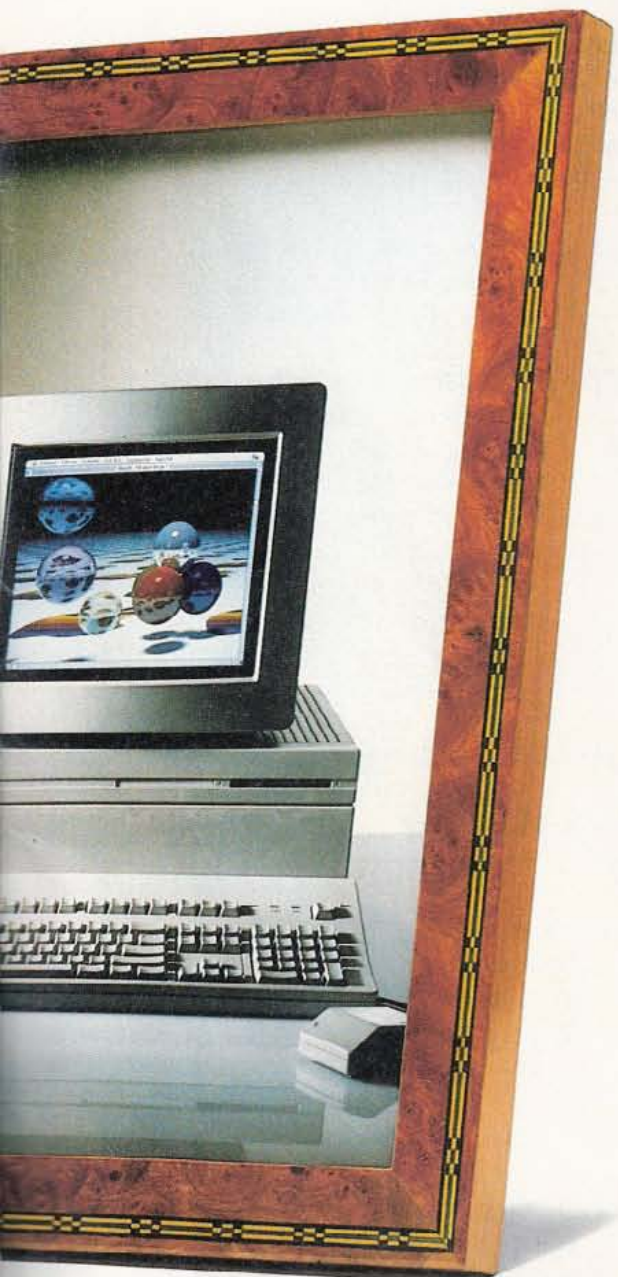
Mais pourquoi au juste les hommes ont-ils le sens de la famille?

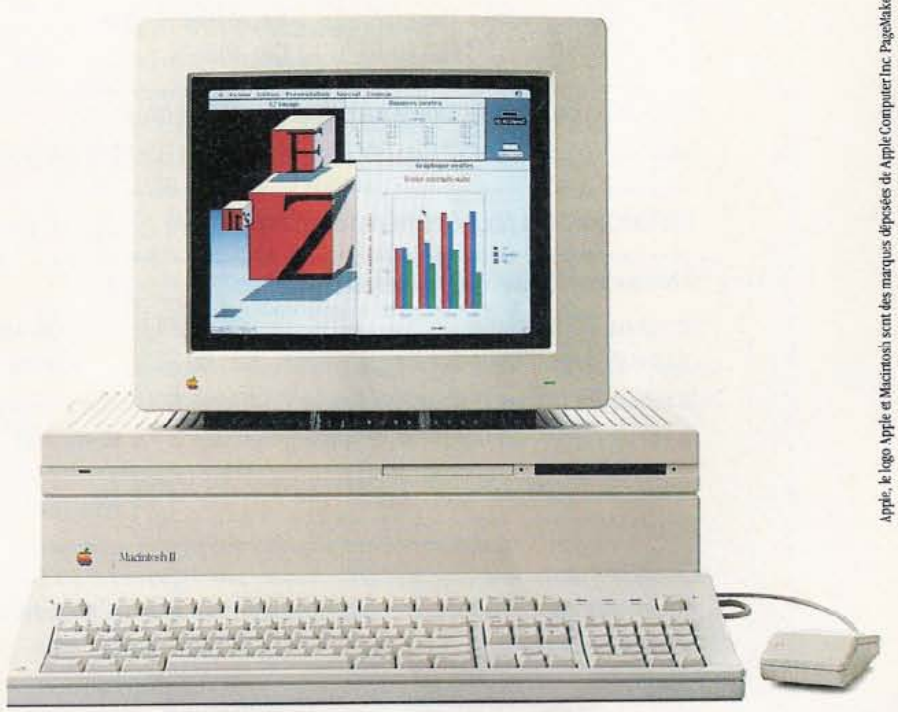
Parce qu'ils n'aiment pas renoncer à ce qui leur est familier, ni s'encombrer l'esprit de règles inutiles, et parce que le progrès et l'expérience doivent être vécus comme un enrichissement et non comme un changement systématique de méthode.

Et comme derrière chaque utilisateur se cache un homme, nous avons en définitive encore plus le sens des affaires.

Apple

Macintosh II





Revues Pom's

n° 8 35,00 F ☐	n° 10 40,00 F ☐	n° 11 40,00 F ☐	n° 12 40,00 F ☐	n° 13 40,00 F ☐
n° 14 40,00 F ☐	n° 15 40,00 F ☐	n° 16 40,00 F ☐	n° 17 40,00 F ☐	n° 18 40,00 F ☐
n° 19 40,00 F ☐	n° 20 40,00 F ☐	n° 21 40,00 F ☐	n° 22 40,00 F ☐	n° 23 40,00 F ☐
n° 24 40,00 F ☐	n° 25 40,00 F ☐	n° 26 40,00 F ☐	n° 27 45,00 F ☐	n° 28 45,00 F ☐
n° 29 45,00 F ☐	n° 30 45,00 F ☐	n° 31 45,00 F ☐	n° 32 45,00 F ☐	n° 33 45,00 F ☐
n° 34 45,00 F ☐				

Disquettes d'accompagnement de Pom's pour Apple][, en 140Ko – 5,25 pouces

n° 1+2 60,00 F ☐	n° 3 60,00 F ☐	n° 4 60,00 F ☐	n° 5 60,00 F ☐	n° 6 60,00 F ☐
n° 7 60,00 F ☐	n° 8 60,00 F ☐	n° 9 60,00 F ☐	n° 10 60,00 F ☐	n° 11 60,00 F ☐
n° 12 60,00 F ☐	n° 13 60,00 F ☐	n° 14 60,00 F ☐	n° 15 60,00 F ☐	n° 16 60,00 F ☐
n° 17 60,00 F ☐	n° 18 60,00 F ☐	n° 19 60,00 F ☐	n° 20 60,00 F ☐	n° 21 60,00 F ☐
n° 22 60,00 F ☐	n° 23 60,00 F ☐	n° 24 60,00 F ☐	n° 25 60,00 F ☐	n° 26 60,00 F ☐
n° 27 60,00 F ☐	n° 28 60,00 F ☐	n° 29 60,00 F ☐	n° 30 60,00 F ☐	n° 31 60,00 F ☐
n° 32 60,00 F ☐	n° 33 60,00 F ☐	n° 34 60,00 F ☐		

Disquettes d'accompagnement de Pom's pour Apple][, en 800Ko – 3,5 pouces

n° 29 80,00 F ☐	n° 30 80,00 F ☐	n° 31 80,00 F ☐	n° 32 80,00 F ☐	n° 33 80,00 F ☐
n° 34 80,00 F ☐				

Disquettes d'accompagnement de Pom's pour Macintosh

	n° 14+15+16 150,00 F ☐	n° 17 80,00 F ☐	n° 18 80,00 F ☐	n° 19 80,00 F ☐
n° 20 80,00 F ☐	n° 21 80,00 F ☐	n° 22 80,00 F ☐	n° 23 80,00 F ☐	n° 24 80,00 F ☐
n° 25 80,00 F ☐	n° 26 80,00 F ☐	n° 27 80,00 F ☐	n° 28 80,00 F ☐	n° 29 80,00 F ☐
n° 30 80,00 F ☐	n° 31 80,00 F ☐	n° 32 80,00 F ☐	n° 33 80,00 F ☐	n° 34 80,00 F ☐

Recueils de la revue Pom's (regroupent quatre numéros de Pom's)

n° 1 (revues 1 à 4) 140,00 F ☐	n° 2 (revues 5 à 8) 140,00 F ☐	n° 3 (revues 9 à 12) 140,00 F ☐
Disquette 1 à 4 200,00 F ☐	Disquette 5 à 8 200,00 F ☐	Disquette 9 à 12 200,00 F ☐

Logiciels pour Apple][

Pom_Link 3.0 140Ko 450,00 F ☐*	Pom_Link 3.0 800Ko 450,00 F ☐*	BananaSoft 140Ko 200,00 F ☐
Clv_Pom's 140Ko 200,00 F ☐	Clv_Pom's 800Ko 200,00 F ☐	Édit. Vidéotex 140Ko 200,00 F ☐
InterPom's 2.0 140Ko 200,00 F ☐	InterPom's 2.0 800Ko 200,00 F ☐	Ludologic 140Ko 80,00 F ☐
E.P.E 5.1 140Ko 200,00 F ☐	E.P.E 5.1 800Ko 200,00 F ☐	MaxMoniteur 140 Ko 150,00 F ☐
Ordico 140Ko 200,00 F ☐	Dominos 140Ko 200,00 F ☐	COGO (src) 140Ko 200,00 F ☐

Logiciels pour Macintosh

Excel efficace 400Ko 195,00 F ☐	Excel efficace 800Ko 175,00 F ☐	MacAstuces 200,00 F ☐
Pom_Link 3.0 *450,00 F ☐	Clv_Pom's 200,00 F ☐	InterPom's 2.0 450,00 F ☐

Disquettes de logiciels 'domaine public' pour Macintosh

Mac 'A' ☐ 'B' ☐ 'C' ☐ 'D' ☐ 'E' ☐ 'F' ☐ 'G' ☐ 'H' ☐ 'I' ☐ 'J' ☐ 80,00 F par disquette

Reliures toilées pour 6 numéros de Pom's (un an) : _____ exemplaire(s) à 60,00 F, soit _____ F

Abonnements pour six numéros à partir du _____, à :

la **revue** Pom's seule 225,00 F ☐	la **revue** et les **disquettes Apple][** 140K 525,00 F ☐
la **revue** et les **disquettes Apple][** 800K 625,00 F ☐	la **revue** et les **disquettes Macintosh** 625,00 F ☐
la **revue** Pom's, les **disquettes Apple][** 140Ko – 5' 1/4 et les **disquettes Macintosh** 925,00 F ☐	
la **revue** Pom's, les **disquettes Apple][** 800Ko – 3' 1/2 et les **disquettes Macintosh** 1025,00 F ☐	

Envoyez ce bon et votre règlement à : **Éditions MEV – 12, rue d'Anjou – 78000 Versailles**

Nom : _____

Adresse : _____

Règlement par : CB/Visa/Euro/MasterCard ☐ Chèque bancaire ☐ Chèque postal ☐ Mandat ☐

numéro de la carte _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ date d'expiration _ _ _ _ _ _

Montant _ _ _ _ , _ _ F Signature :

***Remise de 10% sur Pom_Link 3.1 pour les abonnés**

pour les envois par avion, ajoutez 15 F par numéro et/ou par disquette soit, par exemple, 90 F pour un abonnement avec ou sans disquettes.

Détecteur d'appels téléphoniques

Cet appareil, pour Apple // ou Macintosh, autorise une surveillance de la ligne téléphonique pour l'utilisation de l'Apple comme serveur avec un logiciel tel, par exemple, Répom'deur publié dans le numéro 34 de Pom's.

Câble-interface Apple → Minitel

Pour faire fonctionner les programmes suivants :

- *Minitel/1* pour Macintosh, *MinBas* pour Apple][+, //e, //e+, //c et IIGS : programme permettant l'enregistrement des écrans Minitel, la restitution à loisir hors réseau, le stockage et/ou l'impression de copies d'écran du Minitel, et aussi l'envoi de textes ou messages sur un serveur. Programme du numéro 27 de Pom's.
- *InterPom's 1.0* (et plus) pour Apple][+, //e, //e+, //c, IIGS et Macintosh : programme de téléchargement entre Apple][et/ou Apple][et Macintosh. Transmission de n'importe quel type de fichier (système, texte, binaire, Basic...) en utilisant le Modem du Minitel. Version 1.0 publiée dans le numéro 28 de Pom's.
- *T_Pom's* pour Apple][+, //e, //e+, //c, IIGS* et Macintosh : récupération de l'annuaire téléphonique sous la forme de fichiers texte. Numéro 30 de Pom's.
- *Clv_Pom's* pour Apple][+, //e, //e+, //c et IIGS* et Macintosh : programmes de communication pour CalvaCom et serveurs 'ASCII'. Numéro 31 de Pom's.
- *Paint → Minitel* pour Mac et *HGR → Minitel* pour Apple][+, //e, //e+, //c, IGS* : graphisme et Minitel, programmes proposés dans le numéro 33 de Pom's.
- *Répom'deur* pour Macintosh, Apple][+, //c, //c+, //c et IIGS* : répondeur/enregistreur télématique interrogeable à distance publié dans le numéro 34.

* *sur un Apple IIGS, ce programme fonctionne indifféremment avec le port série intégré ou la carte Super Série Apple. Pour connecter le port intégré du IIGS, utilisez un câble pour Macintosh Plus.*

✂

Je désire recevoir :

détecteur d'appels Apple //	_____	à 500,00 F	_____*
détecteur d'appels Macintosh	_____	à 500,00 F	_____*
câble Minitel/Apple // & SSC	_____	à 225,00 F	_____
câble Minitel/Apple //c	_____	à 225,00 F	_____
câble Minitel/Mac 128, 512K	_____	à 225,00 F	_____
câble Minitel/Mac Plus, IIGS	_____	à 225,00 F	_____
câble Minitel/IBM PC™	_____	à 225,00 F	_____
câble de liaison locale**	_____	à 225,00 F	_____

* *si vous êtes abonnés, vous bénéficiez d'une remise de 10% sur le prix du détecteur, soit 450 F au lieu de 500 F.*

** *préciser le type des deux machines à relier : Mac 512, Mac Plus, Apple //e, //c, IIGS, IBM PC™ .*

Envois par avion : ajoutez 15 F par câble et/ou détecteur

Éditions MEV – 12, rue d'Anjou – 78000 Versailles

Nom : _____

Adresse : _____

_____

En cas de règlement par CB/Visa/Eurocard/Mastercard :

numéro de la carte _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

date d'expiration _ _ _ _ _ _

Montant _ _ _ _ , _ _ F Signature :

AppleWorks™ décuplé

Automatiser
AppleWorks™
version française 1.4 ?
Utiliser la souris ?

SuperMacroWorks
de Randy Brandt

Programme américain
sur disquette 800Ko
sur disquette 140Ko

Version francisée
sur disquette 140Ko

Documentation
française
Disquette Bonus
de Dimitri Geystor

Documentation
américaine

500,00 F

abonnés à Pom's :
450,00 F

Frais de port 20,00 F
Banc d'essai : Pom's 33
Pom's – 12, rue d'Anjou
78000 Versailles
(1) 39 51 24 43

AN APPLEWORKS ENHANCEMENT

UNPROTECTED

Backups may be made using standard copying procedures.

COMPATIBLE

Apple IIe, IIc or IIGS
Requires AppleWorks 2.0 or newer

FOR APPLEWORKS 2.0 OR LATER
AppleWorks version 1.3 and earlier requires MACROWORKS (not Super)

SUPER
MACROWORKS

AppleWorks™ Macro Power! Turn any series of keystrokes into a new one-keystroke command. Adds many new features to your Word Processor, Data Base and Spreadsheet.