

pom's

La revue francophone des utilisateurs de l'Apple

Initiation à l'assembleur (7)

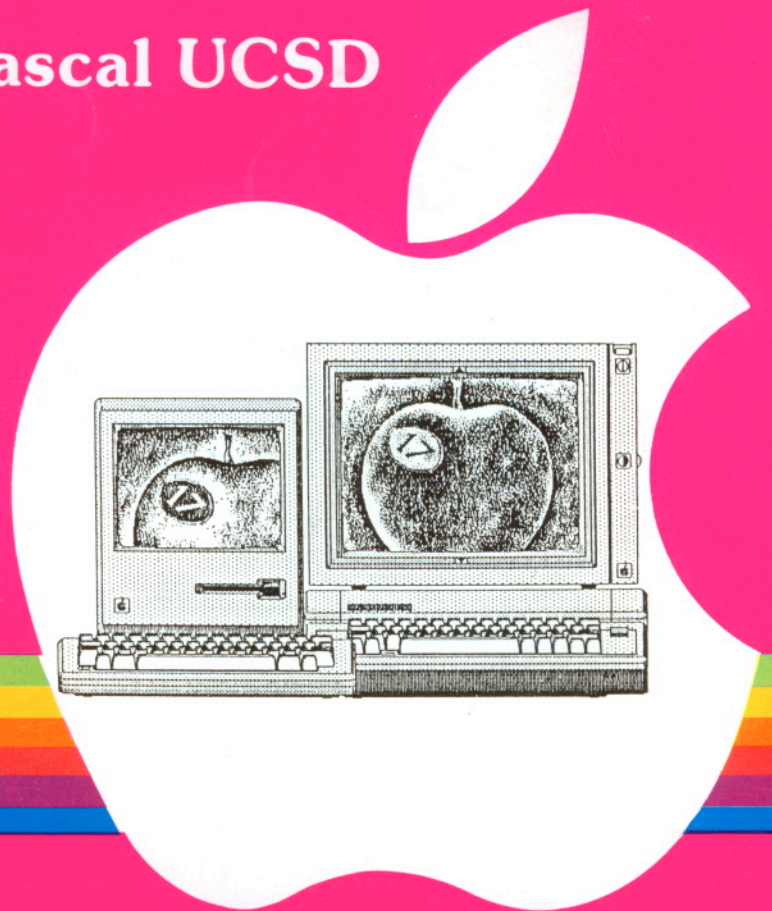
Animations graphiques

Copie graphique sur Epson

Macintosh : deux nouvelles fonctions

Mini-éditeur Basic

Les pointeurs en Pascal UCSD



NUMERO 17 • PRIX 40 F

ISSN : 0294-6068

Demandez le catalogue gratuit
chez votre revendeur habituel
ou chez :
PSI diffusion BP 86
77402 Lagny s/ Marne Cedex
Tél. (6) 006.44.35

APPLE A LIVRE OUVERT

DES LIVRES ET DES DISQUETTES POUR TOUS

POUR DÉBUTER

Apple pour tous ; la découverte de l'Apple ;
36 programmes Apple pour tous ; exercices pour
Apple ; 102 programmes pour Apple ; la pratique
de l'Apple tomes 1 et 2 ; le Basic et l'école tomes
1 et 2 ; Macintosh le magnifique ; programmer
le Macintosh ; multiplan pour Macintosh.

POUR GÉRER

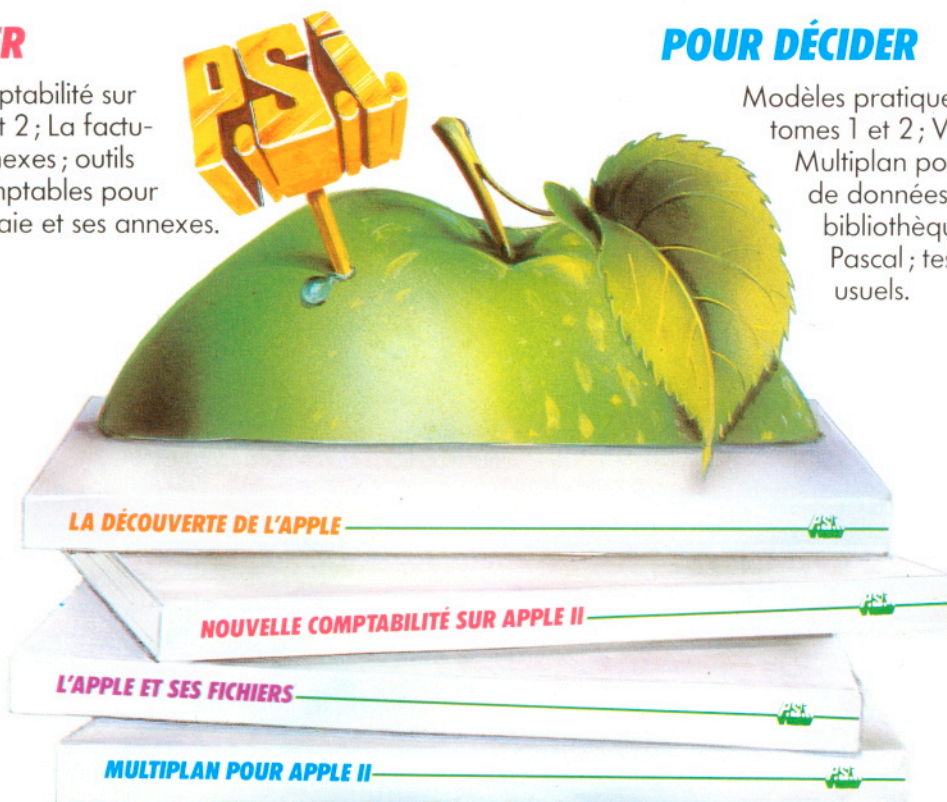
La nouvelle comptabilité sur
Apple tomes 1 et 2 ; La factu-
ration et ses annexes ; outils
financiers et comptables pour
l'entreprise ; la paie et ses annexes.

POUR PROGRESSER

Clefs pour Apple II ; clefs pour Apple //c ;
la pratique de l'Apple tome 3 ; l'Apple et ses
fichiers ; gestion de fichiers et de périphériques
pour AppleII/Pascal ; Microbook, base de données
pour Apple ; Pascal UCSD sur Apple tomes 1 et 2 ;
du Logo pour Apple ; ProDOS sur Apple ;
Pangraphe ; modèles d'expression graphique.

POUR DÉCIDER

Modèles pratiques de décision
tomes 1 et 2 ; Visicalc sur Apple ;
Multiplan pour Apple II ; bases
de données sur Apple II ;
bibliothèque scientifique en
Pascal ; tests statistiques
usuels.



Sommaire

	Page	Langage*	Matériel
Editorial par Hervé Thiriez	5		
Transfert rapide de tableaux par Gérard Michel	6	B-A	+, //e, //c
Les pointeurs en Pascal UCSD par Laurence Tichkowsky	13	P	+, //e, //c Mac
Animation graphiques par Alain Bellegarde	17	B-A	+, //e, //c
Initiation à l'assembleur (7) par Gérard Michel	23	A	+, //e, //c
Copie d'écran sur imprimante Espon par Alexandre Avrane	32	A	+, //e
Catalogue sur imprimante par Jean-Luc Bazanegue	40	B-A	Macintosh
BSAVE et BLOAD avec le Basic Microsoft par Marianne Sutz	47	B-A	Macintosh
Switch vidéo pour carte 80 colonnes par Eric Pascual	52		+, //e
Mini-éditeur Basic par Jean-François Rabasse	55	B-A	+, //e
Visualisation d'une distribution normale par Daniel Hirst	63	B	+, //e, //c
Transformez votre Apple //e en + par François Sermier	65	A	//e
Conversion chiffres-lettres par François Fleury	66	B	+, //e, //c
Micro-informations par Jean-Michel Gourévitch	69		+, //e, //c
Courrier des lecteurs par Olivier Herz	71		

*Langage : B(asic) - A(ssembleur) - P(ascal).

Les annonceurs

Apple : p. 38.39 / Computer 3 : p. 4 / Dynamite Computer : p. 46 / EMB : p. 51 / Hello : p. 75 / List : p. 75 / L'Ordinateur Individuel : p. 76 / PSI : p. 2 / Télécompo : p. 4.

Éditions MEV - 49, rue Lamartine - 78000 Versailles

Directeur de la publication : Hervé Thiriez

LA PHOTOCOMPOSITION EN PROLONGEMENT DE LA MICRO-INFORMATIQUE



TRANSMETTEZ-NOUS VOS TEXTES
PAR TÉLÉPHONE

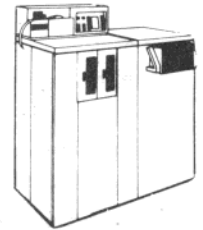
ou

DONNEZ-NOUS VOTRE DISQUETTE



Les textes de vos articles, catalogues, annuaires ou brochures saisis sur votre APPLE sont envoyés directement sur notre photocomposeuse.

Nous vous évitons ainsi, le coût et le temps de la saisie supplémentaire que nécessite le traitement traditionnel de la photocomposition avant l'impression des documents, si vous le désirez nous pouvons également nous charger de l'impression et du brochage.



NOTRE RÉFÉRENCE... LA REVUE POM'S

TELECOMPO 328.18.63

PHOTOCOMPOSITION
BUREAUTIQUE
TRANSMISSION DE DONNÉES

GESTION DE FICHIERS
MATÉRIEL DE
TRAITEMENT DE TEXTES

13 et 15 avenue du Petit Parc
94300 VINCENNES

BONJOUR LES PRIX !!

NOS PRIX SONT F TTC

Carte langage	400	Speech card	320
Carte 128 k ram	1550	Carte horloge	500
Carte 80 colonnes	640	Joystick	165
Interface série	520	Ventilateur	280
Super série	950	Contrôleur de drive auto switch 13/16	370
Interface parallèle	380	Lecteur Disk "Slim"	1500
Grappler + buffer 16 k	1350	Moniteur vert 12"	950
Carte modem	1085	Disquettes 5" 1/4 S.F./S.D. par 1 boîte	95/boîte
Carte Z 80	360	Disquettes 5" 1/4 S.F./D.D. par 1 boîte	175/boîte
Wildcard	400		

AU-DESSUS, NOUS CONSULTER.

Carte bleue et eurocard acceptées

Vente par correspondance: nous consulter.

Computer 3

3, rue Papillon 75009 Paris - Tél. 523.51.15

(metro Poissonnière) ouverture du lundi au samedi de 10 h à 19 h 30



Editorial

La galaxie Apple est décidément en état de mouvement perpétuel ! Steve Wozniak crée une nouvelle société spécialisée dans le gadget électronique pour écrans familiaux, quittant ainsi Apple qui, selon lui, n'a pas assez développé l'Apple II.

Avant son départ, lors d'une interview, Steve déclarait que le //c n'est plus appelé à être "étendu", en tous cas par Apple. La rubrique Micro-informations apporte de plus amples renseignements sur Apple et sa politique.

De notre côté de l'Atlantique, nous avons de bonnes raisons de pavoiser. Les deux meilleurs logiciels de gestion de fichiers sur le Macintosh, parmi une quinzaine de concurrents, sont de création française.

Contrôle X a passé un accord avec Hayden Software pour la commercialisation de CX MacBase aux Etats-Unis.

A.C.I. a réalisé une très belle performance en emportant, avec son programme ABCbase, le prix convoité de la Pomme d'Or.

Au niveau du matériel, Apple-Tell réalise un triomphe, ayant réussi à vendre en décembre sa 1000^e carte, et en voyant la Pomme d'Or couronner trois créateurs ayant réalisé leurs développements autour de cette carte.

Nous vous proposons dans ce numéro un questionnaire que vous pouvez compléter et nous renvoyer. Il y aura 15 abonnements (avec disquettes) à Pom's à gagner. Pour participer au tirage au sort, il vous suffit de nous le faire parvenir avant la fin du mois d'avril. Les lecteurs déjà abonnés pourront prolonger leur abonnement, ou l'offrir (nous encourageons toujours le prosélytisme...). La liste des gagnants pourra être consultée sur notre stand au Sicob Printemps ou à Apple Expo (toutefois, les gagnants seront prévenus par courrier). Bien entendu, un seul questionnaire peut être envoyé par lecteur. Cela nous permettra de mieux vous connaître et de mieux répondre à vos besoins.

Pom's vous propose deux nouveaux produits. Le premier est une disquette de démonstration de Max the Globe Trotter. Max the Globe Trotter est un cours de 50 heures d'anglais sur des supports différents (livre, cassettes audio et disquette). Il est vendu par Microlingua au prix de 1200 francs.

Le second, MAX (sans rapport avec le précédent), est un moniteur très étendu pour Apple //+ , //e ou //c, qui comporte un accès direct aux registres, un mini-assembleur, et bien d'autres choses encore...

Dans ce numéro, vous trouverez, outre les rubriques habituelles "Courrier des lecteurs" et "Micro-informations" que nous devons à **Olivier Herz** et **Jean-Michel Gourévitch**, la conclusion de la célèbre (et remarquable !) série "Initiation à l'assembleur", réalisée par **Gérard Michel**. Gérard nous propose aussi une très utile routine (en assembleur, bien sûr...) destinée aux transferts de tableaux de variables.

Pour rester au chapitre "initiation", **Laurence Tichkowsky**, qui fait désormais partie de l'équipe de rédaction, débute une série sur le Pascal UCSD qui s'adresse à tous les utilisateurs d'Apple, quel que soit leur matériel.

Alain Bellegarde met à notre disposition un programme d'animation de tables de formes, alors que **François Sermier** vous propose de transformer (sans fer à souder !) votre Apple //e en Apple II+.

Avec un fer à souder cette fois, les utilisateurs de cartes 80 colonnes munies d'une sortie vidéo pourront modifier cette dernière afin ne plus avoir à changer les branchements pour passer de 80 à 40 colonnes, ceci à condition de bien respecter les instructions que communique **Eric Pascual**.

Pour les utilisateurs d'imprimante Epson, **Alexandre Avrane** a écrit un programme de copie d'écran en assembleur, pendant que de son côté **Jean-François Rabasse** nous concoctait un mini-éditeur Basic. Vous trouverez aussi un programme écrit par **François Fleury**, qui effectue la conversion de valeurs numériques en leur équivalent en "toutes lettres".

Malgré la densité des informations destinées aux utilisateurs d'Apple II, les possesseurs de Macintosh ne sont pas oubliés. En effet, **Jean-Luc Bazanegue** propose un "Catalogue sur imprimante" et **Marianne Sutz** vous donne la marche à suivre pour ajouter deux nouvelles fonctions au Basic Microsoft. Ces deux programmes font largement appel au langage machine 68000, processeur qui pourrait bien devenir le standard de la micro.

Nombreux sont ceux qui ont souligné l'intérêt pédagogique des articles Mac, bien que ne possédant ni Mac ni carte 68000 sur Apple II ou //. Effectivement, il ne nous était pas apparu que, même sans Mac, on peut être passionné par le 68000.

En ce qui concerne les disquettes Macintosh, et après quelques mois d'expérience, nous avons décidé de publier une disquette Mac par numéro, au prix de 80,00 F. Cela ne fait pas beaucoup plus cher qu'une disquette vierge et vous évitera de longues heures de saisie. Bien entendu la disquette regroupant les programmes des numéros 14 à 16 reste disponible à son prix habituel.

Hervé Thiriez

Ont collaboré à ce numéro : Alexandre Avrane - Jean-Luc Bazanegue - Alain Bellegarde - Alexandre Duback - François Fleury - Jean-Michel Gourévitch - Gérard Michel - Daniel Hirst - Eric Pascual - Jean-François Rabasse - François Sermier - Marianne Sutz - Laurence Tichkowsky. **Rédacteurs :** Alexandre Avrane - Olivier Herz - Laurence Tichkowsky. **Directeur de la publication, rédacteur en chef :** Hervé Thiriez. **Dessins :** Laurent Bidot.

Siège social : Editions MEV - 49, rue Lamartine - 78000 Versailles.

Abonnements et rédaction : Editions MEV - 64-70, rue des Chantiers - 78000 Versailles - Tél. : (3) 951.24.43.

Publicité : Consulter les Editions MEV.

Diffusion N.M.P.P. : Sophie Marnez - Tél. : (1) 240.22.01.

Composition : Télécompo - 13-15, avenue du Petit Parc - 94300 Vincennes - Tél 328.18.63.

Impression : Rosay - 47, avenue de Paris - 94300 Vincennes - Tél. : 328.18.63.

Transfert rapide de tableaux

Gérard Michel

Dans le Pom's 12, nous vous avons présenté une petite routine en langage machine, appelée au moyen de l'ampersand, qui permettait d'effacer un tableau de variables en mémoire et d'incrémenter ou décrétement le nombre d'éléments dans un tableau à une seule dimension.

Nous vous en proposons ici une seconde, complémentaire de la précédente, qui permet de copier rapidement les valeurs des éléments d'un tableau dans un autre, sans passer par des boucles Applesoft, et ce quelle que soit la nature du tableau, dès lors que le tableau de départ et celui d'arrivée sont de même type.

Pour plus de commodité, nous désignerons par T1, T1% ou T1\$, le nom du tableau original (qui peut donc s'appeler en fait A\$, B\$, F%, HH...) et par T2, T2% ou T2\$ le tableau "duplicata".

Syntaxe

L'appel de la routine se fait par le biais d'une indirection dans la routine CHRGET (\$B1), utilisée par l'interpréteur Applesoft pour lire les caractères d'un programme.

Le crochet fermant (]) servira de caractère d'identification des instructions de transfert de tableaux. C'est lui qui indiquera à la routine si l'instruction la concerne ou si la main doit être rendue à l'Applesoft. En outre, pour assurer un fonctionnement correct, le] doit toujours être précédé de deux points (:) dans une ligne de programme, même s'il se trouve en tout début de ligne.

Par ailleurs, la définition préalable des deux tableaux (par DIM ou dimension par défaut) est indispensable pour l'emploi des instructions ci-dessous.

Copie pure et simple

- :JS T1,T2
- :JS T1%,T2%
- :JS T1\$,T2\$

Tous les octets composant le contenu de T1 (stockés dans la zone réservée aux variables dimensionnées) sont recopiés à l'identique aux adresses correspondant au contenu de T2.

Les deux tableaux doivent être exactement de même taille (même nombre de dimensions et même nombre d'éléments par dimension). Il s'agit donc d'effectuer un MOVE de la zone de mémoire définissant T1 dans celle allouée à T2.

Pour les tableaux de réels ou d'entiers (T1 ou T1%), l'instruction JS

réalise en T2 ou T2% une copie totalement indépendante de l'original, puisque c'est la valeur effective de chaque élément qui se trouve stockée dans la zone des tableaux. On peut ainsi, par exemple, modifier ensuite la valeur de T2%(4) sans altérer pour autant celle de T1%(4), et réciproquement.

Pour les tableaux de chaînes de caractères, en revanche, ce ne sont pas les chaînes elles-mêmes qui sont stockées dans la zone des tableaux, mais leurs longueurs et leurs adresses. Par "JS T1\$, T2\$", on copie donc en T2\$ les paramètres définissant les chaînes de T1\$, reliant ainsi une même chaîne à deux tableaux différents, sans rendre la copie indépendante de l'original. Si l'on modifie alors le contenu de T1\$(10), par exemple, on change celui de T2\$(10) qui reste automatiquement identique à T1\$(10), et réciproquement.

Ce type de manipulation présente l'avantage de ne pas encombrer la mémoire avec des duplicatas de chaînes lorsqu'ils ne sont pas réellement utiles. C'est notamment le cas pour une routine Applesoft d'affichage ou d'impression de tableaux alphanumériques, appelée en plusieurs endroits d'un programme par GOSUB pour traiter des tableaux différents (voir à ce propos l'exemple simplifié donné en première illustration dans le programme de démonstration listé en fin d'article).

Toutefois, il faut se méfier alors des routines de nettoyage - mémoires qui ne sont pas prévues pour traiter le cas de chaînes adressées dans deux variables distinctes et donnent des résultats incohérents. En conséquence, il convient de toujours détruire le duplicata après usage, au moyen de la routine DIM.VAR.OBJ du Pom's 12, dont le code objet est fourni en annexe (instruction &Nom du tableau).

Copie et duplication des chaînes

- :JD T1\$,T2\$

Même traitement initial que ci-dessus. Les chaînes de T1\$ sont ensuite copiées, en un autre endroit de la mémoire libre, et affectées à T2\$ qui devient ainsi indépendant de T1\$.

Au prix d'un encombrement mémoire en rapport, on réalise alors une véritable copie de l'original et de ses valeurs, du même type que celle obtenue par des instructions T2\$ (...) = T1\$ (...).

Là encore, T1\$ et T2\$ doivent être de même taille.

Copie de dimension

- :JT T1,T2,i,j,.../,x,y,.../
- :JT T1%,T2%,i,j,.../,x,y,.../
- :JT T1\$,T2\$,i,j,.../,x,y,.../

L'instruction JT permet de copier les valeurs de tous les éléments de la dimension la plus profonde de T1 dans la dimension la plus profonde de T2, avec duplication des chaînes s'il s'agit d'alphanumérique, en fonction de la valeur donnée pour les indices de niveau supérieur.

T1 et T2 peuvent n'avoir pas le même nombre de dimensions, dès lors que la dimension la plus profonde de T2 comporte un nombre d'éléments supérieur ou égal à celui de la dimension équivalente de T1.

Exemples

10 DIM A(10,5,30),B(30)

20 :JT A,B,2,3,/

Copie les valeurs de A(2, 3, 0) à A(2, 3, 30) en B(0) à B(30)

10 DIM comme ci-dessus

20 :JT B,A,/,8,3,/

Copie les valeurs de B(0) à B(30) en A(8, 3, 0) à A(8, 3, 30)

10 DIM BB\$(10,10),C\$(10,10,20)

20 :JT BB\$,C\$,4,/,2,5,/

Copie de BB\$(4, 0) à BB\$(4, 10) en C\$(2, 5, 0) à C\$(2, 5, 10)

La valeur des indices de niveau supérieur peut également être donnée sous forme de variables ou d'expressions numériques (voir programme de démonstration). Evidemment, la valeur de l'indice pour la dimension la plus profonde n'est jamais indiquée, puisque le transfert porte toujours sur tous les éléments de cette dimension.

Mode d'emploi

Vous pouvez suivre une procédure comparable à celle donnée au début du programme de démonstration :

- Chargement de la routine de transfert ST.3;

- Chargement du fichier INTER qui place l'indirection dans CHRGET et oriente sur ST.3 (d'où la nécessité de charger préalablement cette dernière);

- Chargement de DIM.VAR.OBJ et initialisation du vecteur &.

Après ces initialisations, vous pouvez utiliser sans problème les instructions de transfert dans vos programmes Applesoft.

Pour revenir au Basic "standard", il suffit de charger le petit fichier CHRGET, qui rend à la routine commençant en \$B1 son aspect initial.

Précisions sur la structure des tableaux

Rappelons brièvement les éléments les plus simples, déjà largement commentés par ailleurs, et dans le Pom's 12 en particulier.

La représentation d'un tableau en mémoire commence toujours par :

- Deux octets pour son nom.
- Deux octets (poids faible / poids fort) pour le nombre total d'octets occupés par le tableau.

- Un octet pour le nombre de dimensions.

- Deux octets par dimension (poids fort / poids faible) pour indiquer le nombre de ses éléments. On trouve d'abord le nombre d'éléments de la dimension la plus profonde et on remonte ensuite jusqu'à la première, qui occupe ainsi les deux octets précédant le contenu du premier élément du tableau.

Pour calculer l'adresse du premier élément d'un tableau à N dimensions, il faut donc ajouter $(2 * N) + 5$ à l'adresse du début du tableau (premier caractère du nom) dans la zone de stockage pointée par \$6B - \$6C.

Chacun de ces éléments occupe 2 octets (valeur entière) dans un tableau d'entiers, 3 octets pour des chaînes (longueur et adresse de la chaîne), et 5 octets pour les réels (format flottant compacté).

Le problème se complique un peu en ce qui concerne le mode de stockage de ces éléments, c'est-à-dire l'ordre dans lequel ils sont "rangés".

Le principe consiste en un classement hiérarchisé dans l'ordre croissant des indices les plus profonds. Prenons l'exemple d'un tableau à 4 dimensions (A,B,C,D), dont la valeur des indices respectifs sera notée a, b, c, d. En partant du début du tableau vers la fin, on trouvera les éléments

dans l'ordre suivant :

- Tous les éléments d'indice $d = 0$, avec d'abord tous ceux d'indices $c = d = 0$, parmi lesquels viennent d'abord tous ceux d'indices $b = c = d = 0$, en tête desquels on trouve bien sûr l'élément d'indice $a = b = c = d = 0$.

Cela nous donne, pour le tout début du tableau et en résumé, le classement : (0,0,0,0) - (1,0,0,0) - (2,0,0,0) ... (A,0,0,0) - (0,1,0,0) - (1,1,0,0) ... (A,1,0,0) - (0,2,0,0) - (1,2,0,0) ... (A,2,0,0) ... (A,B,0,0) - (0,0,1,0) - (1,0,1,0) ... () ... (A,B,1,0) - (0,0,2,0) ... () ... (A,B,C,0).

- Viennent ensuite les éléments d'indice $d = 1$, avec la même hiérarchie pour les indices supérieurs : de (0,0,0,1) à (A,0,0,1), puis (0,1,0,1) à (A,B,0,1), puis de (0,0,1,1) à (A,B,C,1).

- Et ainsi de suite jusqu'à, enfin, la série des éléments d'indice $d = D$, de (0,0,0,D) à (A,B,C,D).

Tout cela n'est pas particulièrement simple, il est vrai, mais conduit à des formules plus claires pour accéder aux éléments correspondant à une valeur donnée des indices de niveau supérieur et obtenir ainsi la série des éléments de la dimension la plus profonde pour cette valeur.

Il nous faut connaître l'adresse du premier élément appartenant à la série (indice = 0 pour la dimension la plus profonde) et la "distance" qui sépare deux éléments consécutifs dans cette série.

Pour l'adresse du début de série, le problème consiste à calculer le décalage en octets qui sépare le premier élément qui nous intéresse du tout premier élément du tableau. Pour notre exemple de dimension (A,B,C,D), le premier élément d'indice (a,b,c,0) se trouve à :

$$D = a + (A*b) + (A*B*c)$$

éléments du tout premier (0,0,0,0). Pour convertir ce décalage en octets, il suffit de multiplier D par le nombre

d'octets par élément (2, 3 ou 5). En ajoutant ensuite ce décalage à l'adresse de (0,0,0,0), on obtient l'adresse de (a,b,c,0).

La distance entre deux éléments est donnée par le produit des dimensions de niveau supérieur ($A*B*C$), lui-même multiplié par le nombre d'octets par élément.

De façon générale, pour un tableau à $n+1$ dimensions ($D1, D2 \dots, Dn, Dn+1$), dont chaque élément occupe OC octets, et pour des valeurs $V1, V2 \dots Vn$ des indices de niveau supérieur, on obtient la série des éléments de la dimension la plus profonde par les formules :

- Décalage en éléments = $D = V1 + D1*V2 + D1*D2*V3 + \dots + D1*D2*D3 \dots *Dn-1*Vn$
- Décalage en octets = $D*OC = DD$
- Adresse de l'élément ($V1, V2, \dots Vn, 0$) = adresse de (0, 0, ... 0, 0) + DD
- Distance entre deux éléments de la série = $DI = (D1*D2*\dots*Dn)*OC$

Partant de cette structure, le principe de la routine de copie des dimensions les plus profondes consiste à :

- Calculer pour les tableaux T1 et T2 les paramètres DD (puis l'adresse du premier élément) et DI.
- Transférer ensuite les éléments un par un, en passant au suivant par ajout de la valeur de DI dans l'original et de celle de DI dans la copie. Le nombre d'éléments à copier est indiqué par la dimension la plus profonde de T1.

Les informations complémentaires sont fournies dans les commentaires du programme source.

Source Big Mac

```

1 *****
2 *
3 *   COPIE RAPIDE DE TABLEAUX *
4 *   CODE = ST.3 *
5 *
6 *****
7 *
8     ORG $8DE0
9 O1  = $EE ;VARIABLES
    DE LA MULTIPLICATION
10 O2 = $EF ;O2-O1 (PO
    IDS FORT / POIDS FAIBLE) MULTIPLIE
    PAR M2-M1
11 S1 = $F9 ;AVEC RESU
    LTAT DANS S4-S3
12 S2 = $FA ;ET COPIE
    DE O2-O1 DANS S2-S1
13 M1 = $FC
14 M2 = $FD

```

Pom's n° 17

```

15 S3 = $FE
16 S4 = $FF
17     CMP £$5D ;CARACTERE
    "5" ?
18     BEQ KS0
19     CMP £$3A
20     BCC KS1
21 KS3 RTS
22 KS1 JMP $BE
23 KS0 JSR $B1
24     CMP £'T'
25     BNE SS0
26     JMP TT0
27 SS0 CMP £'D'
28     BNE KS2
29     JMP KS4
30 KS2 CMP £'S'
31     BNE KS3
32 SD  JSR SR
33     JMP KS5
34 SR  JSR $B1
35     JSR $F7D9 ;RECHERCHE

```

```

DE TABLEAU POINTE PAR $B8-$B9
36 LDA $9B ;$9B-$9C P
OINTE SUR LE NOM DU TABLEAU
37 STA $6
38 LDA $9C
39 STA $7
40 LDY £0
41 LDA ($B8),Y
42 CMP £$2C ;", " ?
43 BNE KS3
44 JSR $B1
45 JSR $F7D9 ;RECHERCHE
DU 2EME TABLEAU
46 RTS
47 KS5 CLC
48 LDA $6
49 ADC £4 ;ON FAIT L
E "MOVE" A PARTIR DU NOMBRE DE DIM
ENSIONS
50 STA $3C
51 LDA $7
52 ADC £0
53 STA $3D
54 LDY £2
55 LDA ($6),Y ;NBRE OCTE
TS DU TABLEAU -5 (POUR CALCUL FIN
MOVE)
56 SEC
57 SBC £5
58 STA $8
59 INY
60 LDA ($6),Y
61 SBC £0
62 STA $9
63 CLC
64 LDA $3C ;CALCUL AD
RESSE DE FIN DU "MOVE"
65 ADC $8
66 STA $3E
67 LDA $3D
68 ADC $9
69 STA $3F
70 CLC
71 LDA $9B ;ADRESSE D
U TRANSFERT (TABLEAU 2)
72 ADC £4
73 STA $42
74 LDA $9C
75 ADC £0
76 STA $43
77 LDY £0
78 JSR $FE2C ;M O U E
79 JSR $B7
80 RTS
81 KS4 JSR SD ;COPIE DES
CHAINES --> MOVE COMPLET DANS UN
IER TEMPS
82 LDA $9B
83 STA $CE
84 LDY £2
85 CLC
86 ADC ($9B),Y
87 STA $6 ;$6-$7 = A
DRESSE FIN DU TABLEAU + 1
88 LDA $9C ;(ADRESSE
TABLEAU + SON NOMBRE D'OCTETS)
89 STA $CF
90 INY
91 ADC ($9B),Y
92 STA $7
93 LDY £4 ;CALCUL DE
L'ADRESSE DU 1ER ELEMENT DU TABLE
AU
94 LDA ($CE),Y
95 ASL
96 CLC

```

```

97 ADC £5
98 BCC T0
99 INC $CF
100 CLC
101 T0 ADC $CE ;REMIS EN
$CE-$CF (DEBUT + (NBRE DIM * 2) +
5)
102 STA $CE
103 BCC T1
104 INC $CF
105 T1 JSR TR
106 JMP T4
107 TR LDY £2
108 LDA ($CE),Y ;POIDS FOR
T DE L'ADRESSE DE LA CHAINE
109 STA $8
110 DEY
111 LDA ($CE),Y ;POIDS FAI
BLE DE L'ADRESSE DE LA CHAINE
112 TAX
113 DEY
114 LDA $70 ;$6F-$70 E
T $71-$72 = ADRESSE CONTENUE EN $6
F-$70
115 STA $72 ;MOINS LON
GUEUR DE LA CHAINE
116 LDA $6F
117 SEC
118 SBC ($CE),Y
119 STA $71
120 STA $6F
121 BCS T2
122 DEC $72
123 DEC $70
124 T2 LDA ($CE),Y
125 LDY $8
126 JSR $E5E2 ;DEPLACE L
A CHAINE POINTEE PAR X-Y
127 LDY £2 ;EN POSITI
ON POINTEE PAR $71-$72
128 LDA $70
129 STA ($CE),Y ;MAJ NOUVE
LLE ADRESSE CHAINE DANS 2EME TAB (
T2)
130 DEY
131 LDA $6F
132 STA ($CE),Y
133 RTS
134 T4 LDA $CE ;CALCUL AD
RESSE ELEMENT SUIVANT S'IL EXISTE
135 CLC
136 ADC £3
137 STA $CE
138 BCC T3
139 INC $CF
140 T3 LDA $CF
141 CMP $7 ;ARRIVE A
LA FIN DU TABLEAU
142 BNE T1
143 LDA $CE
144 CMP $6
145 BNE T1
146 JSR $B7
147 RTS
148 TT0 JSR SR ;RECHERCHE
DES DEUX TABLEUX T1 ET T2
149 LDA $9B
150 STA $CE
151 LDA $9C
152 STA $CF
153 LDA £5
154 STA $8D ;CALCUL DU
NBRE D'OCTETS PAR ELEMENT (5, 3 0
U 2)
155 LDY £1
156 LDA ($6),Y

```



```

157      BPL TT00
158      LDA £3
159      STA $BD
160      DEY
161      LDA ($6),Y
162      BPL TT00
163      DEC $BD
164 TT00  LDY £5
165      LDA ($6),Y ;VERIFIE :
      DIM LA PLUS PROFONDE DE T2
166      STA $D7 ;SUP. OU E
      GAL A DIM LA PLUS PROFONDE DE T1
167      LDA ($CE),Y
168      CMP $D7
169      BEQ TT1
170      BCS TT100
171 TT2   JMP $DEC9
172 TT1   INY
173      LDA ($6),Y
174      STA $D6 ;ET $D6-$D
      7=NBRE ELE. DANS DIM PLUS PROFONDE
      T1
175      LDA ($CE),Y
176      CMP $D6
177      BCC TT2
178 TT100 JSR TT40
179      JMP TT20
180 TT40  LDA £0
181      STA $ED ;DONNERA N
      BRE DE DIM MOINS 2 EN FINAL
182 TT4   JSR $DEBE ;VERIFIE P
      RESENCE DE "," ET LIT CARACTERE SU
      IVANT
183      CMP £$CB ;TOKEN DE
      "/" ?
184      BEQ TT3
185      JSR $DD67 ;EVALUE EX
      PRESSION POINTEE PAR $B8-$B9
186      JSR $E752 ;CONVERTIT
      EN ENTIER A 2 OCTETS RANGE EN $50
      -$51
187      LDA $ED
188      ASL
189      TAX
190      LDA $50
191      STA D1,X ;STOCKE LA
      VALEUR DE L'INDICE
192      LDA $51
193      INX
194      STA D1,X
195      INC $ED
196      JMP TT4
197 TT3   INC $ED ;VERIFIE N
      BRE D'INDICES / NOMBRE DE DIMENSIO
      NS
198      LDY £4
199      LDA ($6),Y
200      CMP $ED
201      BEQ TT5
202      JMP TT2
203 TT5   DEC $ED ;RETOUR AU
      NOMBRE D'INDICES "UTILE"
204      DEC $ED
205      LDA $ED
206      BMI TT9 ;1 SEULE D
      IM DANS LE TABLEAU (DONC RIEN AVAN
      T /)
207      LDX £1
208      LDA ($6),Y
209      TAY
210      DEY
211      TYA
212      ASL
213      CLC
214      ADC £5
215      TAY ;Y POINTE
      SUR NBRE ELEMENTS DANS LA 1ERE DIM

```

```

216      STY $EB
217      STY $EC
218 TT10  LDA D1,X ;VERIFIE V
      ALEUR DE L'INDICE < NBRE ELEMENTS
      DANS LA DIM
219      CMP ($6),Y
220      BCC TT6
221      BEQ TT7
222 TT8   JMP TT2
223 TT7   INY
224      DEX
225      LDA D1,X
226      CMP ($6),Y
227      BCS TT8
228      INX
229      DEY
230 TT6   INX
231      INX
232      DEY
233      DEY
234      CPY £5 ;TOUS INDI
      CES TRAITES ?
235      BNE TT10
236 TT9   LDX £0
237      STX $18 ;$18-$19 =
      DECALAGE PREMIER ELEMENT "UTILE"
238      STX $19 ;PAR RAPPO
      RT AU 1ER ELEMENT DU TABLEAU
239      STX $1B ;$1A-$1B =
      DISTANCE ENTRE DEUX ELEMENTS "UTI
      LES"
240      LDA $ED
241      BPL TT11
242      INX
243      STX $1A ;SI 1 DIM
      => DECALAGE 0 ET DISTANCE = 1 FOIS
244      LDA £7 ;NBRE OCTE
      TS PAR ELEMENTS
245      JMP TT18 ;1ER ELEME
      NT EST A 7 OCTETS DU 1ER CARACTERE
      DU NOM
246 TT11  BNE TT12
247      LDY $EB
248      LDA ($6),Y ;SI 2 DIM
      X ET Y, DECALAGE = INDICE x
249      STA $1B ;ET DISTAN
      CE = X FOIS NBRE OCTETS PAR ELEMEN
      T
250      INY
251      LDA ($6),Y
252      STA $1A
253      LDA £9 ;1ER ELEME
      NT EST A 9 OCTETS DU 1ER CARACTERE
      DU NOM
254      JMP TT18
255 TT12  STX $1A
256      LDY $EC
257      LDA ($6),Y
258      STA 02 ;NBRE ELEM
      ENTS DANS 1ERE DIM (X) DANS 02-01
259      INY
260      LDA ($6),Y
261      STA 01
262 TT14  DEC $EC
263      DEC $EC
264      LDY $EC
265      LDA ($6),Y
266      STA M2 ;MULTIPLIC
      ATION PAR NBRE ELEMENTS DE LA DIM
      SUIVANTE
267      INY
268      LDA ($6),Y
269      STA M1
270      JSR MULT
271      LDA S3
272      STA $1A ;MAJ DU RE

```

SULTAT EN \$1A-\$1B

273 LDA S4
 274 STA \$1B
 275 LDA \$EC
 276 CMP £7
 277 BEQ TT13 ;ON ARRETE
 A L'AVANT DERNIERE DIM (A 7 OCTET
 S DU BORD)

278 LDA \$1A
 279 STA 01 ;RESULTAT
 EN 02-01 ET MULT PAR DIM SUIVANTE

280 LDA \$1B
 281 STA 02
 282 JMP TT14
 283 TT13 LDY \$EB
 284 STY \$EC
 285 LDA (\$6),Y ;X DANS 02
 -01

286 STA 02
 287 INY
 288 LDA (\$6),Y
 289 STA 01
 290 LDX £2 ;COMMENCE
 A LA VALEUR DE L'INDICE POUR LA 2E
 ME DIM

291 TT16 LDA D1,X
 292 STA M1
 293 INX
 294 TXA
 295 PHA
 296 LDA D1,X
 297 STA M2
 298 JSR MULT ;MULTIPLIC
 ATION PAR VALEUR DE L'INDICE POUR
 LA DIM SUIVANTE

299 CLC
 300 LDA S3
 301 ADC \$18 ;ACCUMULAT
 ION DES RESULTATS INTERMEDIAIRES

302 STA \$18
 303 LDA S4
 304 ADC \$19
 305 STA \$19
 306 DEC \$EC
 307 DEC \$EC
 308 LDY \$EC
 309 CPY £7 ;ARRIVE A
 L'AVANT DERNIERE DIM ?

310 BEQ TT15
 311 LDA (\$6),Y ;DIM SUIVA
 NTE EN M2-M1

312 STA M2
 313 INY
 314 LDA (\$6),Y
 315 STA M1
 316 LDA S1 ;S1-S2 CON
 TIENT TOUJOURS LE PRODUIT DES DIM

317 STA 01 ;PRECEDENT
 ES, REMIS EN 01-02

318 LDA S2
 319 STA 02
 320 JSR MULT
 321 LDA S3 ;PRODUIT D
 ES DIM REMIS EN 01-02

322 STA 01 ;(POUR ASS
 URER SA CONSERVATION EN S1-S2)

323 LDA S4
 324 STA 02
 325 PLA
 326 TAX
 327 INX ;PREND VAL
 EUR DE L'INDICE SUIVANT

328 JMP TT16 ;POUR CONT
 INUER LES MULTIPLICATIONS

329 TT15 PLA
 330 LDY \$EB ;ON VA SE

PLACER SUR LE TOUT PREMIER ELEMENT
 DU TABLEAU

331 INY
 332 INY
 333 TYA
 334 TT18 CLC
 335 ADC \$6
 336 STA \$6
 337 BCC TT17
 338 INC \$7
 339 CLC
 340 TT17 LDA \$ED
 341 BMI TT171 ;1 SEULE D
 IM ?

342 TT170 LDX £0
 343 LDA D1,X ;AJOUTE VA
 LEUR DU 1ER INDICE AU DECALAGE

344 ADC \$18
 345 STA \$18
 346 INX
 347 LDA D1,X
 348 ADC \$19
 349 STA \$19
 350 LDA £0 ;MULTIPLIE
 DECALAGE PAR NBRE D'OCTETS PAR EL
 EMENT

351 STA 02
 352 LDA \$BD
 353 STA 01
 354 LDA \$18
 355 STA M1
 356 LDA \$19
 357 STA M2
 358 JSR MULT
 359 CLC
 360 LDA S3
 361 ADC \$6 ;DECALAGE
 + ADR. 1ER ELE. --> 1ER ELEMENT "U
 TILE"

362 STA \$6
 363 LDA S4
 364 ADC \$7
 365 STA \$7
 366 TT171 LDA £0 ;MULTIPLIE
 DISTANCE PAR NBRE D'OCTETS PAR EL
 EMENT

367 STA 02
 368 LDA \$BD
 369 STA 01
 370 LDA \$1A
 371 STA M1
 372 LDA \$1B
 373 STA M2
 374 JSR MULT
 375 LDA S3
 376 STA \$1A
 377 LDA S4
 378 STA \$1B
 379 RTS

380 TT20 LDA \$6 ;SAUVEGARD
 E DES RESULTATS OBTENUS POUR LE 1E
 R TABLEAU T1

381 STA \$8
 382 LDA \$7
 383 STA \$9
 384 LDA \$1A
 385 STA \$1C
 386 LDA \$1B
 387 STA \$1D
 388 LDA \$D6
 389 STA \$1E
 390 LDA \$D7
 391 STA \$1F
 392 LDA \$CE ;CALCUL DE
 S PARAMETRES POUR LE SECOND TABLEA
 U T2

```

393 STA $6
394 LDA $CF
395 STA $7
396 JSR $B1
397 JSR TT40
398 LDA $6 ;SAUVEGARD
E POUR COPIE DES CHAINES SI NECESS
AIRE
399 STA $CE
400 LDA $7
401 STA $CF
402 TT25 LDY £0
403 TT21 LDA ($8),Y ;TRANSFERT
ELEMENT PAR ELEMENT
404 STA ($6),Y
405 INY
406 CPY $BD
407 BNE TT21
408 DEC $D6 ;$D6-$D7 =
NBRE ELEMENTS A TRANSFERER
409 BNE TT22
410 LDA $D7
411 BEQ TT23
412 TT22 LDA $D6
413 CMP £$FF
414 BNE TT24
415 DEC $D7
416 TT24 CLC ;AJOUTE DI
STANCE DANS T1
417 LDA $8
418 ADC $1C
419 STA $8
420 LDA $9
421 ADC $1D
422 STA $9
423 CLC ;AJOUTE DI
STANCE DANS T2
424 LDA $6
425 ADC $1A
426 STA $6
427 LDA $7
428 ADC $1B
429 STA $7
430 JMP TT25
431 TT23 LDA $BD
432 CMP £3 ;TABLEAU D
E CHAINES DE CARACTERES ?
433 BNE FIN
434 TT28 JSR TR ;COPIE DES

```

```

CHAINES UNE A UNE AVEC MAJ DANS T
2
435 DEC $1E ;$1E-$1F =
NBRE DE CHAINES A COPIER
436 BNE TT26
437 LDA $1F
438 BEQ FIN
439 TT26 LDA $1E
440 CMP £$FF
441 BNE TT27
442 DEC $1F
443 TT27 CLC ;AJOUTE DI
STANCE DANS T2 POUR PASSER A LA CH
AINE SUIVANTE
444 LDA $CE
445 ADC $1A
446 STA $CE
447 LDA $CF
448 ADC $1B
449 STA $CF
450 JMP TT28
451 FIN JMP $B1
452 MULT LDA 01 ;SOUS-ROUT
INE DE MULTIPLICATION
453 STA S1 ;16 BITS P
AR 16 BITS / RESULTAT SUR 16 BITS
454 LDA 02
455 STA S2
456 LDA £0
457 STA S3
458 STA S4
459 LDX £16
460 BS1 LSR M2
461 ROR M1
462 BCC BS0
463 CLC
464 LDA 01
465 ADC S3
466 STA S3
467 LDA 02
468 ADC S4
469 STA S4
470 BS0 ASL 01
471 ROL 02
472 DEX
473 BNE BS1
474 RTS
475 D1 DS 180 ;ZONE DE S
TOCKAGE DES VALEURS DES INDICES

```

TEST. ST3

```

1 PRINT CHR$(4)"BLOAD ST.3": PRINT CH
R$(4)"BLOAD INTER.3
5 HIMEM: 36319
6 PRINT CHR$(4)"BLOAD DIM.VAR.OBJ": PO
KE 1013,76: POKE 1014,7 * 16 + 12:
POKE 1015,9 * 16 + 2
10 DIM A$(300),B$(300),C$(300),D$(300),E
$(300),F$(300)
12 TEXT : HOME
20 GOTO 95
30 FOR I = 0 TO 300: PRINT A$(I): NEXT :
GET Z$: PRINT : & A$: DIM A$(300)
: RETURN
95 PRINT "INITIALISATION TABLEUX
100 FOR I = 0 TO 300:B$(I) = "TABLEAU B"
+ STR$(I):C$(I) = "TABLEAU C" +
STR$(I):D$(I) = "TABLEAU D" +
STR$(I):E$(I) = "TABLEAU E" + ST
R$(I):F$(I) = "TABLEAU F" + STR$(
I): NEXT
110 PRINT : PRINT "FIN INITIALISATION":
PRINT

```

```

120 :$SB$,A$: GOSUB 30:$SC$,A$: GOSUB 30:
$SD$,A$: GOSUB 30:$SE$,A$: GOSUB 3
0:$SF$,A$: GOSUB 30
130 CLEAR
135 DIM A%(100),A(200),A$(300),A2%(100),
A2(200),A2$(300)
140 PRINT : PRINT "INITIALISATION EN COU
RS": FOR I = 0 TO 100:A%(I) = I:A(
I) = 100 * I: NEXT : FOR I = 101 T
O 200:A(I) = 100 * I: NEXT : FOR I
= 0 TO 300:A$(I) = "ELEMENT " +
STR$(I): NEXT
145 PRINT "FIN INIT"
150 :$SA%,A2%:$SA,A2:$DA$,A2$
170 FOR I = 0 TO 100: PRINT A%(I),A2%(I)
: NEXT : PRINT : GET Z$: PRINT : F
OR I = 0 TO 200: PRINT A(I),A2(I):
NEXT : PRINT : GET Z$: PRINT
175 & A%: & A2%: & A: & A2
177 PRINT "MODIFICATION DE A2$": PRINT :
FOR I = 0 TO 300 STEP 10:A2$(I) =
A2$(I) + "MOD": NEXT
178 FOR I = 0 TO 300: PRINT A$(I),A2$(I)
: NEXT : PRINT
180 & A$: & A2$

```

```

190 DIM X1$(150): DIM X2$(2,2,150)
200 PRINT "INITIALISATION X1$": FOR I =
    0 TO 150:X1$(I) = "X1$ NO " + STR
    $(I): NEXT
205 PRINT "FIN INIT"
210 FOR J = 0 TO 2: FOR Z = 0 TO 2:$TX1$
    ,X2$, / ,J,Z, / : FOR I = 0 TO 150
    : PRINT X2$(J,Z,I),J" "Z" "I: NE
    XT : GET Z$: PRINT : NEXT : NEXT
220 & X1$: & X2$: DIM T1%(1,1,1,1,2,40),
    T2%(2,2,2,50)
230 PRINT "INITIALISATION T1%": PRINT

```

```

240 FOR I = 0 TO 40:T1%(1,1,1,1,1,I) = 1
    00 * I: NEXT
245 PRINT "FIN INIT"
250 FOR A = 0 TO 2: FOR B = 0 TO 2: FOR
    C = 0 TO 2:$TT1%,T2%,1,1,1,1, /
    ,A,B,C, / : FOR I = 0 TO 50: PRINT
    T2%(A,B,C,I),A" "B" "C" "I: NE
    XT : GET Z$: PRINT : NEXT C,B,A
270 PRINT : FOR A = 0 TO 2: FOR B = 0 TO
    2: FOR C = 0 TO 2: FOR D = 0 TO 5
    0: PRINT T2%(A,B,C,D)" "": NEXT D
    ,C,B,A

```

Récapitulation

*8DE0.9215

```

8DE0- C9 5D F0 08 C9 3A 90 01
8DE8- 60 4C BE 00 20 B1 00 C9
8DF0- 54 D0 03 4C E2 8E C9 44
8DF8- D0 03 4C 64 8E C9 53 D0
8E00- E7 20 07 8E 4C 24 8E 20
8E08- B1 00 20 D9 F7 A5 9B 85
8E10- 06 A5 9C 85 07 A0 00 B1
8E18- B8 C9 2C D0 CB 20 B1 00
8E20- 20 D9 F7 60 18 A5 06 69
8E28- 04 85 3C A5 07 69 00 85
8E30- 3D A0 02 B1 06 38 E9 05
8E38- 85 08 C8 B1 06 E9 00 85
8E40- 09 18 A5 3C 65 08 85 3E
8E48- A5 3D 65 09 85 3F 18 A5
8E50- 9B 69 04 85 42 A5 9C 69
8E58- 00 85 43 A0 00 20 2C FE
8E60- 20 B7 00 60 20 01 8E A5
8E68- 9B 85 CE A0 02 18 71 9B
8E70- 85 06 A5 9C 85 CF C8 71
8E78- 9B 85 07 A0 04 B1 CE 0A
8E80- 18 69 05 90 03 E6 CF 18
8E88- 65 CE 85 CE 90 02 E6 CF
8E90- 20 96 8E 4C C7 8E A0 02
8E98- B1 CE 85 08 88 B1 CE AA
8EA0- 88 A5 70 85 72 A5 6F 38
8EA8- F1 CE 85 71 85 6F 80 04
8EB0- C6 72 C6 70 B1 CE A4 08
8EB8- 20 E2 E5 A0 02 A5 70 91
8EC0- CE 88 A5 6F 91 CE 60 A5
8EC8- CE 18 69 03 85 CE 90 02
8ED0- E6 CF A5 CF C5 07 D0 B8
8ED8- A5 CE C5 06 D0 B2 20 B7
8EE0- 00 60 20 07 8E A5 9B 85
8EE8- CE A5 9C 85 CF A9 05 85
8EF0- BD A0 01 B1 06 10 08 A9
8EF8- 03 85 BD 88 B1 06 10 02
8F00- C6 BD A0 05 B1 06 85 D7
8F08- B1 CE C5 D7 F0 05 B0 0E
8F10- 4C C9 DE C8 B1 06 85 D6
8F18- B1 CE C5 D6 90 F2 20 24
8F20- 8F 4C A4 90 A9 00 85 ED
8F28- 20 BE DE C9 CB F0 1A 20
8F30- 67 D0 20 52 E7 A5 ED 0A
8F38- AA A5 50 9D 61 91 A5 51
8F40- E8 9D 61 91 E6 ED 4C 28
8F48- 8F E6 ED A0 04 B1 06 C5
8F50- ED F0 03 4C 10 8F C6 ED
8F58- C6 ED A5 ED 30 2F A2 01
8F60- B1 06 A8 88 98 0A 18 69
8F68- 05 A8 84 EB 84 EC BD 61
8F70- 91 D1 06 90 10 F0 03 4C
8F78- 10 8F C8 CA BD 61 91 D1
8F80- 06 B0 F4 E8 88 E8 E8 88
8F88- 88 C0 05 D0 E1 A2 00 86
8F90- 18 86 19 86 1B A5 ED 10
8F98- 08 E8 86 1A A9 07 4C 49
8FA0- 90 D0 10 A4 EB B1 06 85
8FA8- 1B C8 B1 06 85 1A A9 09
8FB0- 4C 49 90 86 1A A4 EC B1
8FB8- 06 85 EF C8 B1 06 85 EE
8FC0- C6 EC C6 EC A4 EC B1 06
8FC8- 85 FD C8 B1 06 85 FC 20
8FD0- 36 91 A5 FE 85 1A A5 FF
8FD8- 85 1B A5 EC C9 07 A0 0B
8FE0- A5 1A 85 EE A5 1B 85 EF

```

```

8FEB- 4C C0 8F A4 EB 84 EC B1
8FF0- 06 85 EF C8 B1 06 85 EE
8FF8- A2 02 BD 61 91 85 FC E8
9000- 8A 48 BD 61 91 85 FD 20
9008- 36 91 18 A5 FE 65 18 85
9010- 18 A5 FF 65 19 85 19 C6
9018- EC C6 EC A4 EC C0 07 F0
9020- 22 B1 06 85 FD C8 B1 06
9028- 85 FC A5 F9 85 EE A5 FA
9030- 85 EF 20 36 91 A5 FE 85
9038- EE A5 FF 85 EF 68 AA E8
9040- 4C FA 8F 68 A4 EB C8 C8
9048- 98 18 65 06 85 06 90 03
9050- E6 07 18 A5 ED 30 31 A2
9058- 00 BD 61 91 65 18 85 18
9060- E8 BD 61 91 65 19 85 19
9068- A9 00 85 EF A5 BD 85 EE
9070- A5 18 85 FC A5 19 85 FD
9078- 20 36 91 18 A5 FE 65 06
9080- 85 06 A5 FF 65 07 85 07
9088- A9 00 85 EF A5 BD 85 EE
9090- A5 1A 85 FC A5 18 85 FD
9098- 20 36 91 A5 FE 85 1A A5
90A0- FF 85 18 60 A5 06 85 08
90A8- A5 07 85 09 A5 1A 85 1C
90B0- A5 18 85 1D A5 D6 85 1E
90B8- A5 D7 85 1F A5 CE 85 06
90C0- A5 CF 85 07 20 B1 00 20
90C8- 24 8F A5 06 85 CE A5 07
90D0- 85 CF A0 00 B1 08 91 06
90D8- C8 C4 BD D0 F7 C6 D6 D0
90E0- 04 A5 D7 F0 25 A5 D6 C9
90E8- FF D0 C2 C6 D7 18 A5 08
90F0- 65 1C 85 08 A5 09 65 1D
90F8- 85 09 18 A5 06 65 1A 85
9100- 06 A5 07 65 18 85 07 4C
9108- D2 90 A5 BD C9 03 D0 23
9110- 20 96 8E C6 1E D0 04 A5
9118- 1F F0 18 A5 1E C9 FF D0
9120- 02 C6 1F 18 A5 CE 65 1A
9128- 85 CE A5 CF 65 18 85 CF
9130- 4C 10 91 4C B1 00 A5 EE
9138- 85 F9 A5 EF 85 FA A9 00
9140- 85 FE 85 FF A2 10 46 FD
9148- 66 FC 90 0D 18 A5 EE 65
9150- FE 85 FE A5 EF 65 FF 85
9158- FF 06 EE 26 EF CA D0 E6
9160- 60 FF 00 FF 00 FF 00 FF
9168- 00 FF 00 FF 00 FF 00 FF
9170- 00 FF 00 FF 00 FF 00 FF
9178- 00 FF 00 FF 00 FF 00 FF
9180- 00 FF 00 FF 00 FF 00 FF
9188- 00 FF 00 FF 00 FF 00 FF
9190- 00 FF 00 FF 00 FF 00 FF
9198- 00 FF 00 FF 00 FF 00 FF
91A0- 00 FF 00 FF 00 FF 00 FF
91A8- 00 FF 00 FF 00 FF 00 FF
91B0- 00 FF 00 FF 00 FF 00 FF
91B8- 00 FF 00 FF 00 FF 00 FF
91C0- 00 FF 00 FF 00 FF 00 FF
91C8- 00 FF 00 FF 00 FF 00 FF
91D0- 00 FF 00 FF 00 FF 00 FF
91D8- 00 FF 00 FF 00 FF 00 FF
91E0- 00 FF 00 FF 00 FF 00 FF
91E8- 00 FF 00 FF 00 FF 00 FF
91F0- 00 FF 00 FF 00 FF 00 FF
91F8- 00 FF 00 FF 00 FF 00 FF
9200- 00 FF 00 FF 00 FF 00 FF
9208- 00 FF 00 FF 00 FF 00 FF
9210- 00 FF 00 FF 00 00

```

*BLOAD DIM.VAR.OBJ

*927C.940C

```

927C- A4 6B 84 18
9280- A4 6C 84 19 A0 05 84 08
9288- A2 00 86 07 86 18 86 1C
9290- 86 1D 86 09 F0 19 20 B1
9298- 00 D0 04 E6 18 D0 35 C9
92A0- 24 F0 11 C9 25 F0 33 C9
92A8- C8 F0 21 C9 C9 F0 20 95
92B0- 06 E8 D0 E2 E6 1D A5 07
92B8- 09 80 85 07 A0 01 B1 88
92C0- 20 BA 00 F0 0F 20 B1 00
92C8- C9 C8 D0 03 4C 50 93 E6
92D0- 1C 4C 50 93 20 E5 92 4C
92D8- 11 93 E6 1D A5 06 09 80
92E0- 85 06 4C B4 92 A0 00 B1
92E8- 18 C5 06 D0 07 C8 B1 18
92F0- C5 07 F0 1C 20 00 93 85
92F8- 19 A5 1A 85 18 4C E5 92
9300- A0 02 B1 18 18 65 18 85
9308- 1A C8 B1 18 65 19 85 1E
9310- 60 A5 18 85 42 A5 19 85
9318- 43 C8 B1 18 85 08 C8 B1
9320- 18 85 09 20 00 93 A5 1A
9328- 85 3C A5 1E 85 3D A5 6D
9330- 85 3E A5 6E 85 3F A0 00
9338- 20 2C FE A5 6D 38 E5 08
9340- 85 6D A5 6E E5 09 85 6E
9348- A6 1B D0 03 20 B1 00 60
9350- 20 E5 92 A0 04 B1 18 C9
9358- 01 D0 F1 20 00 93 A5 1D
9360- F0 09 C6 08 A5 08 38 E5
9368- 1D 85 08 A5 1C F0 38 A0
9370- 06 B1 18 38 E9 01 91 18
9378- 88 B1 18 E9 00 91 18 A0
9380- 02 B1 18 38 E5 08 91 18
9388- C8 B1 18 E9 00 91 18 A5
9390- 1A 48 A5 1E 48 20 00 93
9398- 85 43 A5 1A 85 42 68 85
93A0- 30 68 85 3C 4C 2E 93 A0
93A8- 06 B1 18 18 69 01 91 18
93B0- 88 B1 18 69 00 91 18 A0
93B8- 02 B1 18 18 65 08 91 18
93C0- C8 B1 18 69 00 91 18 A5
93C8- 06 85 07 85 09 A5 6D 85
93D0- 6E 18 65 08 85 60 85 08
93D8- 90 04 E6 6E E6 09 A0 00
93E0- B1 06 91 08 A5 07 C5 1E
93E8- D0 09 A5 06 C5 1A D0 03
93F0- 4C 4C 93 C6 06 A5 06 C9
93F8- FF D0 02 C6 07 C6 08 A5
9400- 08 C9 FF D0 DB C6 09 4C
9408- E0 93 00 02 00

```

*BLOAD INTER.3

00BA- 4C E0 8D

*BLOAD CHRGET

*BA.BD

00BA- C9 3A B0 0A

Les pointeurs en Pascal UCSD

Laurence Tichkowsky

Le Pascal est un langage qui a fait couler beaucoup d'encre et dont on parle suffisamment pour éveiller l'intérêt de ceux qui programment. Il présente des particularités et offre des possibilités de développement que l'on ne trouve pas en Basic, en Fortran,...

Les avis sont très partagés; certains pensent qu'il est réservé aux universitaires, pendant que d'autres mettent au point des produits comme PFS, ABCbase,...

Le problème n'est pas de trancher sur la valeur du Pascal mais d'en parler.

La notion de pointeurs est très spécifique au Pascal. Elle est importante, et permet d'aller plus loin dans la connaissance du langage. C'est pourquoi nous lui consacrons, ici, ces quelques lignes.

Les enregistrements

A la lecture d'un programme, écrit en Pascal UCSD, vous avez sûrement déjà rencontré une déclaration de TYPE dans le genre de celle qui suit :

```
ADRESSE=RECORD
  RUE:STRING;
  CPOS:STRING;
  VILLE:STRING
END;
ELEMENT=RECORD
  NOM:STRING;
  ADR:ADRESSE;
  AGE:STRING
END;
```

ainsi pour définir la ville où habite Monsieur X on écrira, dans le programme principal ou dans la procédure, l'instruction d'affectation :

```
ELEMENT.ADR.VILLE:='PARIS';
```

La structure d'un enregistrement est un type (RECORD). Les éléments le composant sont appelés : "champs", ils peuvent être de types différents. La notion d'enregistrement est une notion récursive. En effet, un champ peut être défini comme un enregistrement (dans la déclaration donnée ci-dessus, le champ "ADR" est un enregistrement dont l'identificateur est "ADRESSE"; les champs le composant sont des sous-champs). Les variables de type enregistrement sont les seules qui permettent de définir des champs dont les types sont différents. Dans notre exemple, nous considérerons des champs de type chaîne de caractères (STRING).

Le système offre à cette structure de nombreux avantages.

Il lui donne la possibilité, grâce à l'instruction WITH, d'alléger l'écriture des affectations de chaque champ. Ainsi, pour définir les champs composant l'enregistrement ADRESSE on Pom's n° 17

écrira :

```
WITH ADRESSE DO
BEGIN
RUE:=';
CPOS:=';
VILLE:=';
END;
```

De ce fait, on évitera le préfixage de toutes les variables champs et donc des erreurs de frappe, malheureusement trop fréquentes !

On peut aussi définir des enregistrements à champs variables. Pour ce faire, on utilisera l'instruction CASE OF; ainsi, la déclaration suivante sera autorisée :

```
GENRE=(FEMININ,MASCULIN);
PERSONNE=RECORD
  NOM:STRING;
  ADR:ADRESSE;
  CASE XX: OF GENRE
    FEMININ:STRING;
    MASCULIN:STRING
  END
END;
```

Cette notion d'enregistrement est grandement liée à celle de fichiers tout en restant indépendante. En effet, pour créer un fichier de personnes, nous aurons recours aux enregistrements pour stocker l'ensemble des informations les concernant.

Les pointeurs

En premier lieu, il est important de faire la différence entre les variables statiques et dynamiques. Une variable déclarée comme suit :

```
VAR X:TYPE1;
```

est une variable statique. En effet, lors de la compilation, le compilateur alloue une case mémoire à la variable X de type TYPE1. L'adresse de cette case mémoire est fixée.

Par opposition, l'attribution d'une case mémoire à une variable dynamique se fait lors de son utilisation à l'exécution. Il est à noter que les pointeurs traitent les variables dynamiques.

Un pointeur est une variable, il est associé à un type défini préalablement; c'est pourquoi on rencontre des pointeurs de toutes sortes : pointeurs d'entiers, de caractères, de booléens, de tableaux, d'enregistrements... Ainsi, on trouvera aisément dans les déclarations d'un programme :

```
TYPE PTR=*ELEMENT;
ELEMENT=RECORD
  NOM:STRING;
  PRENOM:STRING;
  DATE:STRING
END;
```

```
VAR PELT:PTR;
```

ELEMENT est un enregistrement regroupant des informations comme : Nom, Prénom, Date de naissance, PTR définit des pointeurs d'ELEMENT et PELT sont les variables qui vont pointer sur les différents enregistrements.

Pour plus de compréhension, prenons un exemple et analysons-le. Notre exemple sera le programme TRI, listé ci-après. Celui-ci fait un tri alphabétique sur les noms des différents enregistrements stockés dans le fichier TRI1, de type TEXT. Les autres types de fichier seront étudiés ultérieurement.

Analyse des procédures

Procédure LIRE

Le tableau de pointeurs d'ELEMENT est utile dans la mesure où il nous permettra d'avoir accès aux différents enregistrements stockés, de façon séquentielle, dans notre fichier TRI1. Celui-ci n'aura pas sa raison d'être lorsque nous parlerons de listes.

Cette procédure est en fait celle qui nous donne la possibilité de saisir les enregistrements afin de remplir les champs composant le type ELEMENT.

L'appel de la procédure standard NEW, dont le paramètre est une variable pointeur, est très importante. En effet, NEW(VP) entraîne la création d'un nouvel élément, permet l'allocation mémoire nécessaire à une nouvelle variable et de ranger l'adresse mémoire de l'espace alloué dans la variable VP. Ainsi, pour accéder aux informations de ELEMENT il nous faudra passer par l'intermédiaire de VP.

Remarque : si l'enregistrement comporte des champs variables il faudra faire NEW(VP,C1,...,CN); Ci étant les valeurs correspondant aux différents choix d'une structure CASE, ils doivent être écrits dans l'ordre de la déclaration.

VP est de type PTR, lui-même définissant un pointeur d'ELEMENT, ELEMENT étant un enregistrement composé des champs Nom, Prénom et Date. Ainsi, afin de définir les différents champs considérés on écrira :

```
VP^.NOM ou VP^.PRENOM ou encore VP^.date. VP est une variable statique (selon la définition donnée plus haut) tandis que VP^ est dynamique. En effet, c'est le NEW(VP) qui lui permet d'exister. Toutes les autres instructions de cette procédure n'ont nullement besoin d'être détaillées.
```

Procédure LECTURE

Elle appelle LIRE autant de fois qu'il y a d'éléments à trier et définit le tableau de pointeurs.

Les procédures CLASS et ECHANGE

La procédure CLASS effectue un tri BULLE, c'est-à-dire que l'on considère deux éléments consécutifs du fichier, on les compare et les échange s'il y a lieu. Cette procédure appelle ECHANGE. Le témoin COMPTEUR nous permet de savoir quand le tri est terminé (il est égal à 0 quand le parcours de la liste n'a pas entraîné d'échange).

```

PROGRAM TRI (INPUT,OUTPUT);
TYPE PTR=^ELEMENT;
ELEMENT =RECORD
    NOM:STRING;
    PRENOM:STRING;
    DATE:STRING
END;

TAB=ARRAY[1..100] OF PTR;

VAR PT:TAB;
    VP:PTR;
    N,U,COMPTEUR :INTEGER;
    TRI1:TEXT;

PROCEDURE LIRE;
BEGIN
    NEW(VP);
    WRITELN('NOM :');
    READLN(VP^.NOM);
    WRITELN('TRI1,VP^.NOM);
    WRITELN('PRENOM :');
    READLN(VP^.PRENOM);
    WRITELN('TRI1,VP^.PRENOM);
    WRITELN('DATE DE NAISSANCE :');
    READLN (VP^.DATE);
    WRITELN (TRI1,VP^.DATE)
END;

PROCEDURE LECTURE;
BEGIN
    WRITELN ('LA LISTE A TRIER EST :');
    FOR N:=1 TO U DO
        BEGIN
            LIRE;
            PT[N]:=VP
        END
    END;

PROCEDURE ECHANGE (I:INTEGER);
BEGIN
    COMPTEUR:=0;
    VP:=PT[I];
    PT[I]:=PT[I+1];
    PT[I+1]:=VP;
    COMPTEUR:=1
END;

PROCEDURE CLASS;
VAR I:INTEGER;
BEGIN
    REPEAT
        BEGIN
            COMPTEUR:=0;
            FOR I:=1 TO U-1 DO
                BEGIN
                    IF PT[I]^NOM>PT[I+1]^NOM THEN ECHANGE (I)
                END
            END
        END
    UNTIL COMPTEUR=0
END

```

Procédure IMPRESSION

Comme son nom l'indique, elle imprime les résultats trouvés, c'est-à-dire, la liste des noms triés.

Les listes

Une des contraintes du Pascal réside dans la déclaration des tableaux. En effet, il est impossible de définir des tableaux de dimension variable. Ainsi, le plus souvent, on les surdimensionne et la place mémoire disponible se trouve réduite. Il serait donc intéressant d'exploiter la création dynamique au niveau des éléments d'un tableau.

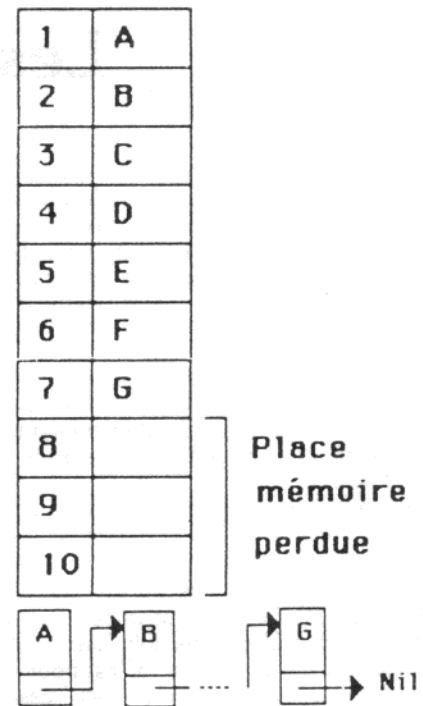
La structure d'un tableau établit une liaison entre celui-ci et ses éléments par l'intermédiaire des indices. Il faut donc trouver une représentation similaire. Cette représentation est celle des listes. Une liste est une suite d'éléments, créés dynamiquement, et reliés par une structure souple réalisée à l'aide des pointeurs.

Chaque élément de la liste se compose d'un minimum de deux informations :

- ses composantes propres, correspondant à l'élément du tableau (entier, réel, enregistrement,...)
- le pointeur vers l'élément suivant de la liste.

Le dessin ci-dessous montre un tableau

TAB=ARRAY[1..10] OF CHAR; dont les éléments sont A, B, C, D, E, F, G et sa représentation en liste. Ce schéma, bien que très primaire, met en évidence la structure d'une liste par rapport à celle d'un tableau.



Une des grandes différences entre un tableau et une liste réside dans l'accès aux éléments. Lorsqu'il s'agit d'un tableau, il suffit de connaître l'indice qui lui est associé; les éléments d'une liste, quant à eux, sont accessibles par le pointeur pointant sur chacun. Ainsi, pour accéder au 3^e élément il nous faut connaître son pointeur, celui-ci est contenu dans le 2^e, dont le pointeur est contenu dans le 1^{er}. De ce fait, le premier élément trouvé nous donnera l'adresse du second,...

Les éléments d'une liste sont, de part la structure du Pascal, des enregistrements. En effet, ils se composent de deux (ou plus) parties de types différents (dans notre exemple un caractère et un pointeur), et les seules variables autorisant le mélange des types sont les enregistrements. Ainsi, les éléments de la liste représentée ci-dessus seront déclarés comme suit :

```

TYPE PELT=^ELT;
ELT= RECORD
    CAR:CHAR;
    PTC:PELT
END;

```

Le champ CAR contient le caractère et PTC contient le pointeur vers l'élément suivant, lui-même de type ELT.

Comme nous l'avons précisé plus haut, les éléments d'une liste sont créés de façon dynamique (allocation mémoire au fur et à mesure de l'exécution du programme) tandis que le tableau est construit lors de la compilation.

Revenons à l'accès aux éléments de la liste. Il a été dit, précédemment, qu'il faut connaître le premier élément de la liste pour accéder aux suivants. De ce fait, un pointeur de début de liste est indispensable; par convention, le pointeur du dernier élément de la liste sera NIL. Lors de

```

UNTIL COMPTEUR=0
END;

PROCEDURE IMPRESSION;

VAR I:INTEGER;

BEGIN
WRITELN (TR11,'LA LISTE TRIEE EST :');
FOR I:=1 TO U DO
BEGIN
UP:=PT[I];
WRITELN (UP^.NOM);
WRITELN (TR11,UP^.NOM);
WRITELN (UP^.PRENOM);
WRITELN (TR11,UP^.PRENOM);
WRITELN (UP^.DATE);
WRITELN (TR11,UP^.DATE)
END
END;

(*****
(*      DEBUT DU PROGRAMME      *)
(*      PRINCIPAL                *)
(*****

BEGIN
REWRITE (TR11,'TR11');
WRITELN ('NOMBRE DE NOM INFERIEUR A 100');
READLN (U);
LECTURE;
CLASS;
WRITELN (TR11);
WRITELN (TR11);
IMPRESSION;
CLOSE (TR11,LOCK)
END.

```

la création d'une liste le pointeur de tête sera défini et restera inchangé tout au long du traitement; tandis que le pointeur de fin de liste sera modifié. Un pointeur intermédiaire est nécessaire à l'insertion de nouveaux éléments, de ce fait le pointeur de fin sera déplacé en conséquence. La structure de liste permet, avec souplesse, d'insérer et de supprimer des éléments. Écrivons un programme regroupant la création d'une liste, l'insertion et la suppression d'un élément dans celle-ci.

Ce programme listé ci-après crée une liste d'enregistrements dont les champs sont :

- un nom
- et un pointeur.

Il permet aussi d'insérer un élément grâce à la procédure INSERER. La procédure SUPPRIMER supprime un élément de la liste.

Explication du programme

Création

Cette procédure, comme son nom l'indique, est celle qui permet de créer une liste. Au début, la liste est vide; ainsi, les pointeurs de début et de fin de liste sont égaux. Puis on arrive dans la boucle WHILE, par la-

```

PROGRAM LISTEALPHA (INPUT,OUTPUT);

TYPE PELT=^ELEMENT;
ELEMENT=RECORD
    NOM:STRING;
    PTSUIV:PELT
END;

VAR PTDEB,PTFIN,PTINT,POINT : PELT;
(* PTDEB: POINTEUR DE DEBUT DE LISTE *)
(* PTFIN: POINTEUR DE FIN DE LISTE *)
(* PTINT: POINTEUR INTERMEDIAIRE *)
(* NECESSAIRE AU TRAITEMENT *)

A,B,C : STRING;
X,Y : INTEGER;

PROCEDURE CREATION;

BEGIN
NEW (PTDEB);
(* POUR LA CREATION DU 1ER ELEMENT *)
PTFIN:=PTDEB;
(* LISTE VIDE => POINTEUR DE FIN = *)
(* POINTEUR DE DEBUT DE LISTE *)
WRITELN ('ENTREZ LE 1ER NOM ');
READLN (C);

(* LECTURE DU NOM *)
(* PAR CONVENTION L'ARRET SE *)
(* FERA PAR ENTREE D'UNE CHAINE VIDE *)
(* ET LE POINTEUR DE FIN SERA NIL *)

WHILE (C <> '') DO
BEGIN
NEW (PTINT);
(* CREATION D'UN NOUVEL ELEMENT *)
PTINT^.NOM:= C;
(* AFFECTATION DU CHAMP *)
PTFIN^.PTSUIV:= PTINT;
(* LIAISON AVEC ELEMENT PRECEDENT *)
PTFIN:=PTINT;
(* POINTEUR DE FIN DEPLACÉ *)
PTFIN^.PTSUIV:= NIL;
READLN (C);
(* LECTURE DU PROCHAIN ELEMENT *)
END

```

```

END;

PROCEDURE LIRE;

BEGIN
PTINT:=PTDEB;
(* ON COMMENCE A LIRE DEPUIS LE DEBUT *)
(* ON VA AFFICHER LES ELEMENTS DE LA LISTE *)
(* LE TEST D'ARRET SERA LA VALEUR NIL *)
(* DU POINTEUR DE FIN DE LISTE *)

WHILE (PTINT^.PTSUIV <> NIL) DO
BEGIN
WRITELN (PTINT^.NOM);
PTINT := PTINT^.PTSUIV
END;
WRITELN (PTINT^.NOM)
(* NECESSAIRE POUR DE DERNIER ELEMENT *)
END;

PROCEDURE INSERER (A : STRING);

VAR L : INTEGER;
PTANC:PELT;

BEGIN
(* INITIALISATION *)
PTINT:=PTDEB;
L:=0;
WHILE (L<>1) AND (PTINT <> NIL) DO
BEGIN
IF PTINT^.NOM<A THEN BEGIN PTANC:= PTINT;
PTINT:=PTINT^.PTSUIV
END
ELSE L:=1
END;

(* AU SORTIR DE LA BOUCLE IL FAUT TESTER *)
(* LE RESULTAT DE LA RECHERCHE *)
(* SI OUI : ALLOCATION MEMOIRE POUR LE *)
(* NOUVEL ELEMENT ET AFFECTATION DU CHAMP *)

NEW (POINT);
POINT^.PTSUIV:=PTANC^.PTSUIV;
POINT^.NOM:=A;
PTANC^.PTSUIV:=POINT
END;

```

quelle on définit et affecte les champs de chaque enregistrement. L'instruction NEW(PTINT), va créer un nouvel élément de type ELEMENT. Ainsi, le premier élément de la liste existe et peut donc être initialisé. Comme nous l'avons vu plus haut, un élément de la liste n'est accessible que par son pointeur, lui-même contenu dans le champ pointeur de l'élément précédent. Il est donc indispensable de relier les éléments de la liste entre eux. Cette liaison est réalisée par l'instruction :

```
PTFIN^.PTSUIV:=PTINT;
```

Ainsi, on peut déplacer le pointeur de fin de liste et continuer la création des éléments de la liste. La saisie des différents noms des éléments de la liste se terminera par l'entrée (au clavier) d'une chaîne vide.

Lecture

Cette procédure n'a nullement besoin d'être commentée. Elle permet l'affichage des éléments de la liste.

Insérer

Pour insérer un élément dans une liste, il faut déterminer l'endroit de l'insertion. Une insertion au début ou en fin de liste, ne pose pas de problème, il suffit de déplacer les pointeurs de début et de fin de liste. L'insertion en milieu de liste est plus délicate. Dans notre exemple, pour plus de facilité, le programme affiche les éléments existants et vous demande après quel nom vous désirez insérer le nouveau nom. Ainsi, le programme recherche le nom existant, crée un nouvel élément et met à jour les pointeurs.

Insertion alphabétique

Cette procédure n'est intéressante que si la liste à traiter n'a qu'un seul élément. Ainsi, la liste des noms sera triée par ordre alphabétique. Pour des enregistrements comportants d'autres champs, le tri pourra, bien sûr, s'effectuer sur un prénom, une date de naissance,...

La procédure recherche l'endroit de l'insertion, crée un nouvel élément, et enfin déplace les pointeurs en conséquence.

Supprimer

Cette procédure supprime un nom existant et déplace les pointeurs en conséquence.

La structure de liste apporte un plus au traitement, quand les éléments de celle-ci doivent être rangés et prélevés dans un ordre bien défini. L'utilisation d'une liste plutôt que d'un tableau se fait en fonction :

- de la difficulté à connaître la dimension du tableau;
- des différentes modifications susceptibles d'intervenir, quant au nombre d'éléments du tableau;
- du nombre de manipulations à effectuer sur les éléments. ■

```
PROCEDURE SUPPRIMER (A:STRING);
VAR L:INTEGER;
BEGIN
  PTINT:=PTDEB;
  L:=0;
  WHILE (L<>1) AND (PTINT <> NIL) DO
    BEGIN
      IF PTINT^.NOM=A THEN L:=1
      ELSE BEGIN
        POINT:=PTINT;
        PTINT:=PTINT^.PTSUIV
      END;
    END;
  IF L=1 THEN BEGIN
    IF PTINT=PTDEB THEN PTDEB:=PTINT^.PTSUIV
    ELSE POINT^.PTSUIV:=PTINT^.PTSUIV
  END
  ELSE WRITELN ('LE NOM ',A,' EXISTE PAS')
END;
```

```
(* ***** *)
(* DEBUT DU PROGRAMME PRINCIPAL *)
(* ***** *)
```

```
BEGIN
  CREATION;
  LIRE;
  X:=1;
  WHILE X=1 DO
    BEGIN
      WRITELN ('0- POUR SUPPRIMER');
      WRITELN ('1- POUR INSERER ');
      WRITELN ('2- RIEN FAIRE');
      READLN (Y);
      CASE Y OF
        0 : BEGIN
          WRITELN ('NOM A SUPPRIMER');
          READLN (A);
          SUPPRIMER (A);
          LIRE;
        END;
        1 : BEGIN
          WRITELN ('NOM A INSERER ');
          READLN (A);
          INSERER (A);
          LIRE;
        END;
        2 : BEGIN
          WRITELN ('VOUS VOULEZ VRAIMENT RIEN FAIRE');
          READLN (X)
        END
      END
    END
  END.
```

```
PROCEDURE INSERER (A,B : STRING);
VAR L : INTEGER;
BEGIN
  (* INITIALISATION *)
  PTINT:=PTDEB;
  L:=0;
  WHILE (L<>1) AND (PTINT <> NIL) DO
    BEGIN
      IF PTINT^.NOM=B THEN L:=1
      ELSE PTINT:=PTINT^.PTSUIV
    END;
  (* AU SORTIR DE LA BOUCLE IL FAUT TESTER *)
  (* LE RESULTAT DE LA RECHERCHE *)
  (* SI OUI : ALLOCATION MEMOIRE POUR LE *)
  (* NOUVEL ELEMENT ET AFFECTATION DU CHAMP *)
  IF L=1 THEN BEGIN
    NEW (POINT);
    POINT^.PTSUIV:=PTINT^.PTSUIV;
    POINT^.NOM:=A;
    PTINT^.PTSUIV:=POINT
  END
  ELSE WRITELN ('LE NOM ',B,' EXISTE PAS')
END;
```


Animations graphiques

Alain Bellegarde

L'écran graphique de l'Apple II permet d'effectuer des animations de bonne qualité grâce à sa définition de 280 par 192 points.

L'écran haute résolution se situe dans la zone mémoire \$2000 à \$3FFF pour la première page et \$4000 à \$5FFF pour la seconde.

Chacune des 192 lignes de l'écran est constituée de 40 octets consécutifs, dont 7 bits seulement sont affichés à l'écran. Le huitième bit (bit de poids fort ou MSB) est utilisé pour la sélection des couleurs. Lorsque ce bit est à 1, la couleur d'un point isolé en colonne paire sera bleue alors qu'un point isolé en colonne impaire apparaîtra orange. Lorsque le bit "couleur" est à 0, la couleur d'un point isolé sera violette pour les colonnes paires et verte pour les colonnes impaires.

Les animations réalisées sur l'Apple peuvent être classées en deux grandes catégories :

- Celles utilisant les vecteurs "shapes" de l'Applesoft ou des procédures similaires.
- Celles utilisant les blocs d'image.

Les vecteurs shapes

Dans ce type d'animation, les tracés sur l'écran sont effectués par déplacement d'un vecteur unitaire "allumant" ou "éteignant" un point de l'écran, c'est-à-dire mettant à 1 ou à 0 l'un des 7 bits d'un octet de la page graphique.

Ce type de tracé, défini dans le manuel Applesoft, permet de modifier facilement l'échelle et l'orientation d'un dessin. Ces modifications sont présentes en Applesoft sous les commandes "ROT" et "SCALE".

Par contre, l'utilisation en est assez lente puisque, pour traiter un octet de la page graphique, il convient de répéter 7 fois la procédure "déplacement, mise à 0 ou à 1 d'un octet".

Les blocs d'image

Le principe du bloc image est de sauvegarder les octets constituant un rectangle d'image comprenant le dessin à animer. L'avantage de ce type d'animation réside dans sa rapidité. En effet, le traitement est effectué par octet et non par bit, ce qui permet de charger beaucoup plus rapidement un dessin en un endroit quelconque de la page graphique.

Par contre, il est difficile, voire impossible, d'effectuer des rotations et des agrandissements du dessin.

Un autre inconvénient de l'utilisation des blocs image est lié à l'utilisation de l'octet comme unité de travail. Il s'agit des saccades obtenues dans les déplacements horizontaux lorsque l'on utilise tel quel le bloc sauvegardé. Cette saccade provient du fait que les déplacements à droite ou à gauche sont effectués au pas de un octet soit 7 bits ou 7 points de l'écran.

Un remède possible à cet inconvénient consiste en l'utilisation d'un programme décalant les bits d'un octet, ce qui permet de réduire les déplacements à un pas inférieur.

Le programme peut être exécuté soit au moment de chaque tracé, soit avant le tracé, les différents blocs modifiés étant à leur tour sauvegardés en mémoire et utilisés au moment convenable.

Dans le premier cas, le programme de déplacement des bits dans un octet conduit à une augmentation du temps de tracé et annule une partie de l'avantage des blocs image.

Dans le second cas, la rapidité d'exécution est maintenue, mais il est nécessaire de disposer d'une place suffisante en mémoire pour loger les différents blocs modifiés.

Cette dernière technique, qui porte en anglais le nom de "pre-shifted shapes" est employée par le programme Applesoft BLOC.BAS, qui utilise la routine assembleur BLOC.OBJ (dont le code source est BLOC.TXT, sous Lisa 2.5).

Afin d'utiliser ce programme, vous devez disposer d'un dessin sauvegardé sous forme de page graphique, et réalisé par HPLOT et consorts. Il s'agit bien là d'un dessin standard, et non d'une shape qui serait inexploitable pour BLOC.BAS.

Après avoir effacé la page HGR 1, le cas échéant, et chargé votre dessin de base, vous pourrez définir le pas de déplacement (1 à 6) et le nombre de blocs à inclure dans la séquence de déplacements (1 à 7).

La taille du bloc peut être donnée par Joystick ou clavier, en indiquant simplement le point haut - gauche et le point bas - droit du cadre dans lequel sera inscrit le motif à déplacer. La séquence de déplacements ainsi constituée vous sera ensuite démontrée à l'écran et vous pourrez enfin l'enregistrer sur disquette.

L'ensemble des blocs composant la séquence est sauvegardé à partir de

l'adresse \$6000 selon le schéma suivant :

- \$6000 : hauteur du bloc (nombre de lignes)
- \$6001 : largeur du bloc (en octets, soit 7 points)
- \$6002 : pointeurs du début du premier bloc
- \$6003 : (adresse basse et adresse haute)
- \$6004 : pointeurs du début du second bloc
- \$6005 : (adresse basse et adresse haute)
- ...
- \$600E : pointeurs du début du 7ème bloc
- \$600F
- \$6010 et suivants : octets du bloc image

Utilisation de la séquence

L'utilisation de la séquence créée par le programme BLOC.BAS est réalisée par l'intermédiaire du programme TRACE.OBJ, qui est en fait une version améliorée d'une partie du programme BLOC.OBJ.

Les paramètres nécessaires à l'exécution du tracé sont :

- le numéro du bloc utilisé, qui doit être stocké en \$FE (254)
- les coordonnées du point haut et gauche du bloc sous la forme du numéro de ligne stocké en \$FC (252 - si on utilise une valeur telle que la somme numéro de ligne et hauteur du dessin soit supérieure à 192, le dessin sera coupé dans le bas) et le numéro de la colonne gauche en octet (0 à 39) stocké en \$FD (253 - si on utilise une valeur inférieure à 0 ou telle que la colonne indiquée plus la largeur du dessin soit supérieure à 40, le dessin sera coupé à gauche ou à droite).

Tracé avec Joystick

Le programme TRACE.BAS utilise la routine précédente pour une animation de la séquence ROBOT.BLC, donnée à titre d'exemple. Pour l'exécuter, il suffit de taper RUN TRACE.BAS et d'entrer les coordonnées de départ de l'animation en réponse aux questions du programme. ■

BLOC - BAS

```

10 LOMEM: 36864
20 REM A BELLEGARDE 8/1984
30 POKE 232,239: POKE 233,64: POKE 230,3
  2: HCOLOR= 1: SCALE= 1: ROT= 0: PR
  INT CHR$(4);"BLOAD BLOC.OBJ"
40 REM 1ERE PHASE
50 ONERR GOTO 60
60 HOME : TEXT : VTAB 1: HTAB 13: INVERS
  E : PRINT "1ERE PHASE": NORMAL
70 VTAB 5: PRINT " VOUS POUVEZ : "
80 VTAB 8: PRINT "- 1 - CHARGER UN DESSI
  N "
90 VTAB 10: PRINT "- 2 - CHARGER UN BLOC
  "
100 VTAB 12: PRINT "- 3 - PASSER A LA PH
  ASE SUIVANTE"
110 VTAB 14: PRINT "- 4 - AFFICHER LA PA
  GE GRAPHIQUE"
120 VTAB 16: PRINT "- 5 - EFFACER LA PAG
  E GRAPHIQUE"
130 VTAB 18: PRINT "- 0 - SORTIR DU PROG
  RAMME"
140 VTAB 21: INPUT " VOTRE CHOIX :
  ";C%
150 IF C% < 0 OR C% > 5 THEN VTAB 23: P
  RINT : GOTO 140
160 IF C% = 0 THEN HOME : END
170 ON C% GOTO 180,250,470,430,440
180 ONERR GOTO 240
190 HOME : VTAB 1: HTAB 8: PRINT "CHARGE
  MENT D'UN DESSIN": NORMAL : VTAB 5
200 INPUT "NOM DU DESSIN A CHARGER 'C' D
  ONNE LE CATALOGUE ET 'RETURN' A
  NNULE LA PROCEDURE : ";T$
210 IF T$ = "C" THEN PRINT CHR$(4);"C
  ATALOG": PRINT : GOTO 200
220 IF T$ = "" THEN 60
230 PRINT CHR$(4);"BLOAD ";T$;";A$2000
  "; GOTO 50
240 PRINT : INPUT " ERREUR TAPER 'RETURN
  ' POUR CONTINUER ";T$: PRINT : GOT
  O 200
250 ONERR GOTO 420
260 HOME : VTAB 1: HTAB 8: PRINT " CHARG
  EMENT D'UN BLOC": VTAB 5
270 INPUT "NOM DE LA SEQUENCE A CHARGER
  'C' DONNE LE CATALOGUE ET 'RETURN
  ' ANNULE LA PROCEDURE : ";T$
280 IF T$ = "C" THEN PRINT CHR$(4);"C
  ATALOG": PRINT : GOTO 270
290 IF T$ = "" THEN 60
300 PRINT CHR$(4);"BLOAD ";T$;";A$6000
  "
310 ONERR GOTO 320
320 PRINT : INPUT " NUMERO DU BLOC A AFF
  ICHER <1/7> : ";NUM%
330 PRINT : PRINT " COORDONNEES D'AFFICH
  AGE "; HTAB 12: PRINT "(POINT HAUT
  - GAUCHE)"
340 PRINT : INPUT " X (HAUT-GAUCHE) <0,2
  79> : ";X%
350 IF X < 0 OR X > 279 THEN 340
360 INPUT "Y (HAUT-GAUCHE) <0,191> : ";Y
  %
370 IF Y < 0 OR Y > 191 THEN 360
380 POKE 250,16: POKE 251,96: POKE 252,Y
  %: POKE 253, INT (X% / 7): POKE 25
  4,NUM%
390 POKE - 16304,0: POKE - 16300,0: PO
  KE - 16297,0
400 CALL 16559
410 VTAB 21: INPUT "TAPER 'RETURN' POUR
  CONTINUER ";T$: GOTO 50
420 PRINT : INPUT " ERREUR TAPER 'RETURN
  ' POUR CONTINUER ";T$: PRINT : GOT
  O 270
430 POKE - 16304,0: POKE - 16302,0: PO
  KE - 16297,0: POKE - 16300,0: GE
  T T$: GOTO 60
440 HOME : VTAB 5: INPUT "VOULEZ-VOUS RE
  ELLEMENT EFFACER LA PAGE GRAPHIQU
  E <O/N> ";T$
450 IF T$ < > "O" THEN 60
460 HGR : TEXT : GOTO 60
470 REM 2EME PHASE
480 ONERR GOTO 490
490 HOME : VTAB 1: HTAB 13: INVERSE : PR
  INT "2EME PHASE": NORMAL
500 VTAB 5: PRINT " VOUS POUVEZ : "
510 VTAB 8: PRINT "- 1 - DEFINIR LE PAS
  ET LE NOMBRE DE BLOCS DE
  LA SEQUENCE"
520 VTAB 11: PRINT "- 2 - RETOURNER A LA
  1ERE PHASE"
530 VTAB 14: PRINT "- 3 - PASSER A LA 3E
  ME PHASE"
540 VTAB 17: PRINT "- 0 - SORTIR DU PROG
  RAMME"
550 VTAB 20: INPUT " VOTRE CHOIX :
  ";C%
560 IF C% < 0 OR C% > 3 THEN VTAB 20: P
  RINT : GOTO 550
570 IF C% = 0 THEN HOME : END
580 ON C% GOTO 590,50,750
590 ONERR GOTO 600
600 HOME : VTAB 1: HTAB 8: INVERSE : PRI
  NT "DEFINITION DE LA SEQUENCE": NO
  RMAL
610 VTAB 4: PRINT " - PAS DE LA SEQUENCE
  "
620 VTAB 6: PRINT "LE PAS DE LA SEQUENCE
  EST EGAL AU DEPLACEMENT ENT
  RE LES BLOCS"
630 VTAB 9: PRINT "UN PAS = 1 EST LE PLU
  S FIN MAIS LA COULEUR DU DESS
  IN EST MODIFIEE A CHAQUE TRACE (SA
  UF SI LE DESSIN EST NOIR/BLANC)UN
  PAS = 2 PERMET DE CONSERVER LES
  COULEURS"
640 VTAB 15: INPUT "PAS CHOISI <1/6> : "
  ;PAS%
650 IF PAS% < 1 OR PAS% > 6 THEN VTAB 1
  5: PRINT : GOTO 640
660 ONERR GOTO 670
670 VTAB 18: PRINT " - NOMBRE DE BLOCS D
  E LA SEQUENCE"
680 VTAB 20: PRINT "UN NOMBRE = 7 PERMET
  DE DEFINIR TOUTE LES POSSIBILIT
  ES DE DESSIN"
690 VTAB 23: INPUT "NOMBRE DE BLOCS <1/7
  > : ";NOMBRE%
700 IF NOMBRE% < 1 OR NOMBRE% > 7 THEN
  VTAB 23: PRINT : GOTO 690
710 HOME : VTAB 7: PRINT "CES PARAMETRES
  NECESSITENT DE DISPOSER DE ";7 *
  INT (.9 + (PAS% * NOMBRE%) / 7);
  " POINTS GRAPHIQUES A LA DROITE DU
  "
720 VTAB 9: PRINT "DESSIN POUR PERMETTRE
  D'EFFECTUER LES DEPLACEMENTS"
730 VTAB 12: PRINT "CETTE DISPOSITION SE
  RA VERIFIEE PAR LE PROGRAMME DANS
  LA PHASE SUIVANTE TAPER SU
  R UNE TOUCHE POUR CONTINUER ";: GE

```

```

T T#: GOTO 490
740 REM 3EME PHASE
750 ONERR GOTO 760
760 HOME : VTAB 1: HTAB 13: INVERSE : PR
INT "3EME PHASE": NORMAL
770 VTAB 5: PRINT "VOUS POUVEZ : "
780 VTAB 8: PRINT "-- 1 - DEFINIR LE BLOC
DE REFERENCE DE LA SEQUEN
CE AVEC LES PADDLES"
790 VTAB 11: PRINT "-- 2 - DEFINIR LE BLO
C DE REFERENCE DE LA SEQUENCE
AVEC LE CLAVIER TOUCHES I,
J,K,M ET 'RETURN' "
800 VTAB 15: PRINT "-- 3 - RETOURNER A LA
2EME PHASE"
810 VTAB 18: PRINT "-- 0 - SORTIR DU PROG
RAMME"
820 VTAB 22: INPUT " VOTRE CHOIX :
";C%
830 IF C% < 0 OR C% > 3 THEN VTAB 21: P
RINT : GOTO 820
840 IF C% = 0 THEN HOME : END
850 ON C% GOTO 860,930,470
860 POKE - 16304,0: POKE - 16302,0: PO
KE - 16297,0: POKE - 16300,0: HC
OLOR= 3
870 GOSUB 1560
880 X1 = X:Y1 = Y
890 IF PEEK ( - 16287) > 127 THEN 890
900 GOSUB 1560
910 X2 = X:Y2 = Y
920 GOTO 990
930 POKE - 16304,0: POKE - 16302,0: PO
KE - 16297,0: POKE - 16300,0: HC
OLOR= 3
940 GOSUB 1610
950 X1 = X:Y1 = Y
960 GOSUB 1610
970 X2 = X:Y2 = Y
980 XDRAW 1 AT X1,Y1: XDRAW 1 AT X2,Y2
990 IF X1 > X2 THEN TEMP = X1:X1 = X2:X2
= TEMP
1000 IF Y1 > Y2 THEN TEMP = Y1:Y1 = Y2:Y
2 = TEMP
1010 XDRAW 1 AT X1,Y1: XDRAW 1 AT X2,Y2
1015 X3 = X1:Y3 = Y1:X4 = X2:Y4 = Y2
1020 X1 = X1 - 1:Y1 = Y1 - 5:Y2 = Y2 - 1
1030 POKE 238, INT (.9 + (PAS% * NOMBRE%
) / 7)
1040 POKE 250,16: POKE 251,96: POKE 252,
Y1 + 1: POKE 253,(X1 + 1) / 7: POK
E 254,1
1050 POKE 24576,Y2 - Y1 - 1: POKE 24577,
INT ((X2 - X1) / 7)
1060 CALL 16510
1070 IF PEEK (238) < > 0 THEN POKE -
16301,0: PRINT " ATTENTION, LA PL
ACE PREVUE A DROITE DU DESSIN EST
INSUFFISANTE POUR EFFECTUER LES
DEPLACEMENTS. PRESSER UNE TOUCHE
POUR CONTINUER ";: GET T#
1080 HCOLOR= 3: HPLLOT X1,Y1 TO X1,Y2 TO
X2,Y2 TO X2,Y1 TO X1,Y1
1090 POKE - 16301,0: HOME : VTAB 21: IN
PUT "EST-CE QUE LE BLOC CONVIENT <
O/N> ? ";T#
1100 HCOLOR= 0: HPLLOT X1,Y1 TO X1,Y2 TO
X2,Y2 TO X2,Y1 TO X1,Y1: DRAW 1 AT
X3,Y3: DRAW 1 AT X4,Y4: HCOLOR= 3
: IF T# = "N" THEN TEXT : GOTO 75
0
1110 IF T# < > "O" THEN 1080
1120 REM 4EME PHASE

```

```

1130 ONERR GOTO 1140
1140 HOME : TEXT : VTAB 1: HTAB 13: INVE
RSE : PRINT "4EME PHASE": NORMAL
1150 VTAB 5: PRINT "VOUS POUVEZ : "
1160 VTAB 8: PRINT "-- 1 - CREER LA SEQUE
NCE DEFINIE"
1170 VTAB 11: PRINT "-- 2 - RETOURNER A L
A 3EME PHASE"
1180 VTAB 14: PRINT "-- 0 - SORTIR DU PRO
GRAMME"
1190 VTAB 18: INPUT " VOTRE CHOIX :
";C%
1200 IF C% < 0 OR C% > 2 THEN VTAB 15:
PRINT : GOTO 1190
1210 IF C% = 0 THEN HOME : END
1220 ON C% GOTO 1230,750
1230 POKE - 16304,0: POKE - 16302,0: P
OKE - 16300,0: POKE - 16297,0
1240 NOMBRE = NOMBRE%:PAS = PAS%
1250 IF NOMBRE = 1 THEN 1340
1260 FOR I = 2 TO NOMBRE
1270 CALL 16384
1280 FOR J = 1 TO PAS
1290 CALL 16450
1300 NEXT J
1310 POKE - 16301,0: HOME : VTAB 21: PR
INT "BLOC NUMERO ";I;" ON CONTINUE
<O/N> ";: INPUT T#
1320 IF T# = "N" THEN 1130
1330 NEXT I
1340 CALL 16384
1350 REM 5EME PHASE
1360 ONERR GOTO 1370
1370 HOME : TEXT : VTAB 1: HTAB 13: INVE
RSE : PRINT "5EME PHASE": NORMAL
1380 VTAB 5: PRINT "VOUS POUVEZ : "
1390 VTAB 8: PRINT "-- 1 - SAUVER LA SEQU
ENCE CREEE"
1400 VTAB 11: PRINT "-- 2 - RETOURNER A L
A 4EME PHASE"
1410 VTAB 14: PRINT "-- 3 - RETOURNER A L
A 1ERE PHASE"
1420 VTAB 17: PRINT "-- 0 - SORTIR DU PRO
GRAMME"
1430 VTAB 21: INPUT " VOTRE CHOIX :
";C%
1440 IF C% < 0 OR C% > 3 THEN VTAB 21:
PRINT : GOTO 1430
1450 IF C% = 0 THEN HOME : END
1460 ON C% GOTO 1480,1130,50
1470 ONERR GOTO 1550
1480 HOME : VTAB 1: HTAB 8: PRINT "SAUVE
GARDE D'UNE SEQUENCE": VTAB 5
1490 INPUT "NOM DE LA SEQUENCE A SAUVER
'C' DONNE LE CATALOGUE ET 'RETUR
N' ANNULE LA PROCEDURE : ";T#
1500 IF T# = "C" THEN PRINT CHR# (4);"
CATALOG": PRINT : GOTO 1490
1510 IF T# = "" THEN 1360
1520 FIN = 256 * PEEK (251) + PEEK (250
):LO = FIN - 24384
1530 PRINT CHR# (4);"BSAVE ";T#;"A#600
0,L";LO
1540 GOTO 1360
1550 PRINT : INPUT " ERREUR TAPER 'RETUR
N' POUR CONTINUER ";T#: PRINT : GO
TO 1490
1560 REM CROIX GRAPHIQUE
1570 X = 7 * INT (40 * ( PDL (0) / 255))
: IF X = 280 THEN X = 279
1580 Y = INT (191 * ( PDL (1) / 255))
1590 XDRAW 1 AT X,Y: IF PEEK ( - 16287)
< 128 THEN XDRAW 1 AT X,Y: GOTO 1570

```

```

1600 RETURN
1610 X = 140:Y = 90
1620 POKE - 16368,0
1630 XDRAW 1 AT X,Y
1640 XDRAW 1 AT X,Y: IF PEEK ( - 16384)
    < 141 THEN 1630
1650 IF PEEK ( - 16384) = 201 THEN Y =
    Y - 1: IF Y < 0 THEN Y = 0
1660 IF PEEK ( - 16384) = 205 THEN Y =
    Y + 1: IF Y > 191 THEN Y = 191
1670 IF PEEK ( - 16384) = 202 THEN X =
    X - 7: IF X < 0 THEN X = 0
1680 IF PEEK ( - 16384) = 203 THEN X =
    X + 7: IF X > 279 THEN X = 279
1690 IF PEEK ( - 16384) = 141 THEN XDR
    AW 1 AT X,Y: RETURN
1700 GOTO 1620

```

JLOAD TRACE.BAS

```

1 REM TRACE.BAS
2 REM A. BELLEGARDE 8/84
3 HOME : TEXT : VTAB 1: HTAB 8: INVERSE
  : PRINT "ANIMATION AVEC JOYSTICK":
  NORMAL
7 REM CHARGEMENT DE LA SEQUENCE
8 PRINT CHR$(4);"BLOAD ROBOT.BLC"

```

```

9 PRINT CHR$(4);"BLOAD TRACE.OBJ"
10 VTAB 8: INPUT "COLONNE DE DEPART: ";C
    OL:T = COL
20 VTAB 12: INPUT "LIGNE DE DEPART : ";
    LIG
30 VTAB 16: INPUT "BLOC DE DEPART : ";
    BLO
40 HGR : POKE - 16302,0
50 POKE 252,LIG: POKE 253,T: POKE 254,BL
    0
60 CALL 7936
70 X = PDL (0):Y = PDL (1)
80 IF X > 80 AND X < 160 THEN 150
90 IF X > 159 THEN 120
100 BLO = BLO - 1: IF BLO < 1 THEN COL =
    COL - 2:BLO = 7:T = COL
110 GOTO 130
120 BLO = BLO + 1: IF BLO > 7 THEN COL =
    COL + 2:BLO = 1:T = COL
130 IF COL > 128 THEN COL = 128
140 IF COL < 0 THEN T = 256 + COL
150 IF Y > 80 AND Y < 160 THEN 200
160 IF Y > 159 THEN 190
170 LIG = LIG - 2: IF LIG < 0 THEN LIG =
    0
180 GOTO 50
190 LIG = LIG + 2: IF LIG > 255 THEN LIG
    = 255
200 GOTO 50

```

```

1      TTL "BLOC.TXT"
2 *
3 *****
4 *
5 *      A. BELLEGARDE      8/84      *
6 *
7 *****
8 *
9      ORG $4000
10 HBASL EPZ $26
11 HBASH EPZ $27
12 TEMP1 EPZ $EB          ;VARIAB
    LE TEMPORAIRE
13 TEMP2 EPZ $EC
14 TEMP3 EPZ $ED
15 TEMP4 EPZ $EE
16 LSTOCK EPZ $FA          ;ADRESS
    E BASSE DE STOCKAGE DU BLOC
17 HSTOCK EPZ $FB          ;ADRESS
    E HAUTE DE STOCKAGE DU BLOC
18 LIGHAUT EPZ $FC          ;LIGNE
    HAUTE DU DESSIN
19 COLGAU EPZ $FD          ;COLONN
    E GAUCHE DU DESSIN
20 NUMSHA EPZ $FE          ;NUMERO
    DU BLOC UTILISE
21 HAUT EQU $6000          ;HAUTEU
    R DU BLOC
22 LARGE EQU $6001          ;LARGEU
    R DU BLOC
23 HPOSN EQU $F411
24 *
25 * SAUVEGARDE DES OCTETS DU BLOC IMAG
    E
26 *
27      LDA NUMSHA
28      ASL
29      TAX
30      LDA LSTOCK
31      STA HAUT,X
32      INX

```

```

33      LDA HSTOCK
34      STA HAUT,X
35      INC NUMSHA
36      LDA HAUT
37      STA TEMP3
38      LDA LIGHAUT
39      STA TEMP2
40 LOOP1 LDA TEMP2
41      LDX #$00
42      LDY #$00
43      JSR HPOSN
44      LDY COLGAU
45      LDX #$00
46      LDA LARGE
47      STA TEMP1
48 LOOP2 LDA (HBASL),Y
49      STA (LSTOCK,X)
50      INY
51      INC LSTOCK
52      BNE SUITE
53      INC HSTOCK
54 SUITE DEC TEMP1
55      BNE LOOP2
56      INC TEMP2
57      DEC TEMP3
58      BNE LOOP1
59      RTS
60 *
61 * ROL DU BLOC IMAGE
62 *
63      LDA HAUT
64      STA TEMP3
65      LDA LIGHAUT
66      STA TEMP2
67 LOOP5 LDA TEMP2
68      LDX #$00
69      LDY #$00
70      JSR HPOSN
71      LDX LARGE
72      LDY COLGAU
73      CLC

```

```

74 LOOP3   LDA (HBASL),Y
75         ROL
76         ROL
77         STA (HBASL),Y
78         INY
79         DEX
80         BNE LOOP3
81         LDX LARGE
82         LDY COLGAU
83 LOOP4   CLC
84         LDA (HBASL),Y
85         ROR
86         BCC SUITE2
87         ORA ##80
88 SUITE2  STA (HBASL),Y
89         INY
90         DEX
91         BNE LOOP4
92         INC TEMP2
93         DEC TEMP3
94         BNE LOOP5
95         RTS
96 *
97 * POSSIBILITE DE DEPLACER LE BLOC IM
  AGE
98 *
99         LDA COLGAU
100        ADC LARGE
101        STA TEMP1
102        DEC TEMP1
103 LOOP7   LDA HAUT
104        STA TEMP3
105        LDA LIGHAUT
106        STA TEMP2
107 LOOP6   LDA TEMP2
108        LDX ##00
109        LDY ##00
110        JSR HPOSN
111        LDY TEMP1
112        LDA (HBASL),Y
113        BNE RETOUR
114        INC TEMP2
115        DEC TEMP3
116        BNE LOOP6
117        DEC TEMP1
118        DEC TEMP4
119        BNE LOOP7
120        RTS
121 RETOUR  INC TEMP4
122        RTS
123 *
124 * DESSIN DU BLOC IMAGE
125 *
126        LDA NUMSHA
127        ASL
128        TAX
129        LDA HAUT,X
130        STA LSTOCK
131        INX
132        LDA HAUT,X
133        STA HSTOCK
134        LDA HAUT
135        STA TEMP3
136        LDA LIGHAUT
137        STA TEMP2
138 LOOP8   LDA TEMP2
139        LDX ##00
140        LDY ##00
141        JSR HPOSN
142        LDY COLGAU
143        LDX ##00
144        LDA LARGE
145        STA TEMP1

```

```

146 LOOP9  LDA (LSTOCK,X)
147        STA (HBASL),Y
148        INY
149        INC LSTOCK
150        BNE SUITE4
151        INC HSTOCK
152 SUITE4  DEC TEMP1
153        BNE LOOP9
154        INC TEMP2
155        DEC TEMP3
156        BNE LOOP8
157        RTS
158 *
159 * CROIX GRAPHIQUE
160 *
161        HEX 01000400242020961B6F492
        D00
162        END

```

```

1          ORG $1F00
2 *****
3 *
4 *      A. BELLEGARDE      8/84      *
5 *
6 *****
7 *
8 HBASL   EPZ $26
9 HBASH   EPZ $27
10 TEMP1  EPZ $EB
11 TEMP2  EPZ $EC
12 TEMP3  EPZ $ED
13 TEMP4  EPZ $EE
14 LSTOCK  EPZ $FA
15 HSTOCK  EPZ $FB
16 LIGHAUT EPZ $FC
17 COLGAU EPZ $FD
18 NUMSHA  EPZ $FE
19 HAUT    EQU $6000
20 LARGE   EQU $6001
21 HPOSN   EQU $F411
22 *
23 * TRACE DU BLOC
24 *
25 TRACE  LDA NUMSHA
26         ASL
27         TAX
28         LDA HAUT,X
29         STA LSTOCK
30         INX
31         LDA HAUT,X
32         STA HSTOCK
33         LDA HAUT
34         STA TEMP3
35         LDA LIGHAUT
36         STA TEMP2
37 LOOP1   LDA COLGAU
38         STA TEMP4
39         LDA TEMP2
40         BPL CONT
41         CMP #191
42         BPL BASPAGE
43 CONT    LDX ##00
44         LDY ##00
45         JSR HPOSN
46         LDA LARGE
47         STA TEMP1
48 LOOP2   LDY TEMP4
49         BMI COUPE
50         CPY ##28
51         BPL COUPE

```

TRACE

```

52      LDX #*00
53      LDA (LSTOCK,X)
54      STA (HBASL),Y
55 COUPE INC TEMP4
56      INC LSTOCK
57      BNE SUITE
58      INC HSTOCK

```

```

59 SUITE DEC TEMP1
60      BNE LOOP2
61      INC TEMP2
62      DEC TEMP3
63      BNE LOOP1
64 BASPAGE RTS
65      END

```

Récapitulation

BLOC. OBJ

*4000.40FE

```

4000- A5 FE 0A AA A5 FA 9D 00
4008- 60 E8 A5 FB 9D 00 60 E6
4010- FE AD 00 60 85 ED A5 FC
4018- 85 EC A5 EC A2 00 A0 00
4020- 20 11 F4 A4 FD A2 00 AD
4028- 01 60 85 EB B1 26 81 FA
4030- C8 E6 FA D0 02 E6 FB C6
4038- EB D0 F1 E6 EC C6 ED D0
4040- D9 60 AD 00 60 85 ED A5
4048- FC 85 EC A5 EC A2 00 A0
4050- 00 20 11 F4 AE 01 60 A4
4058- FD 18 B1 26 2A 2A 91 26
4060- C8 CA D0 F6 AE 01 60 A4
4068- FD 18 B1 26 6A 90 02 09
4070- 80 91 26 C8 CA D0 F2 E6
4078- EC C6 ED D0 CE 60 A5 FD
4080- 60 01 60 85 EB C6 EB AD
4088- 00 60 85 ED A5 FC 85 EC
4090- A5 EC A2 00 A0 00 20 11
4098- F4 A4 EB B1 26 D0 0D E6
40A0- EC C6 ED D0 EB C6 EB C6
40A8- EE D0 DC 60 E6 EE 60 A5
40B0- FE 0A AA BD 00 60 85 FA
40B8- ES BD 00 60 85 FB AD 00

```

```

40C0- 60 85 ED A5 FC 85 EC A5
40C8- EC A2 00 A0 00 20 11 F4
40D0- A4 FD A2 00 AD 01 60 85
40D8- EB A1 FA 91 26 C8 E6 FA
40E0- D0 02 E6 FB C6 EB D0 F1
40E8- E6 EC C6 ED D0 D9 60 01
40F0- 00 04 00 24 20 20 96 1B
40F8- 6F 49 2D 00 00 00 00

```

*1F00.1F50 TRACE. OBJ

```

1F00- A5 FE 0A AA BD 00 60 85
1F08- FA E8 BD 00 60 85 FB AD
1F10- 00 60 85 ED A5 FC 85 EC
1F18- A5 FD 85 EE A5 EC 10 04
1F20- C9 BF 10 2C A2 00 A0 00
1F28- 20 11 F4 AD 01 60 85 EB
1F30- A4 EE 30 0A C0 28 10 06
1F38- A2 00 A1 FA 91 26 E6 EE
1F40- E6 FA D0 02 E6 FB C6 EB
1F48- D0 E6 E6 EC C6 ED D0 C8
1F50- 60

```

ROBOT. BLOC

*6000.60EF

```

6000- 20 07 10 60 F0 60 D0 61
6008- B0 62 90 63 70 64 50 65

```

```

6010- 00 00 00 00 00 00 00 00
6018- 00 00 00 00 00 00 00 00
6020- 00 00 00 00 00 00 00 7C
6028- 1F 00 00 00 00 00 04 10
6030- 00 00 00 00 00 07 70 00
6038- 00 00 00 00 01 40 00 00
6040- 00 00 00 01 40 00 00 00
6048- 00 00 19 4C 00 00 00 00
6050- 00 3D 5E 00 00 00 00 30
6058- 19 4C 06 00 00 00 30 01
6060- 40 06 00 00 00 30 05 50
6068- 06 00 00 00 30 7D 5F 06
6070- 00 00 00 30 01 40 06 00
6078- 00 00 30 01 40 06 00 00
6080- 00 30 01 40 06 00 00 00
6088- 30 01 40 06 00 00 00 30
6090- 01 40 06 00 00 00 3C 01
6098- 40 1E 00 00 00 04 01 40
60A0- 10 00 00 00 04 01 40 10
60A8- 00 00 00 00 01 40 00 00
60B0- 00 00 70 01 40 07 00 00
60B8- 00 10 00 00 04 00 00 00
60C0- 70 7F 7F 07 00 00 00 00
60C8- 07 70 00 00 00 00 40 08
60D0- 08 01 00 00 00 20 10 04
60D8- 02 00 00 00 70 3F 7E 07
60E0- 00 00 00 00 00 00 00 00
60E8- 00 00 00 00 00 00 00 00

```

LIST

LE JOURNAL
DES AMATEURS
DE PROGRAMMATION

Si programmer un ordinateur est devenu pour vous un loisir, un plaisir... une passion, sachez que **LIST** a été créé pour vous. **LIST** vous aide à tirer davantage de votre matériel, à vous perfectionner dans la conception des programmes qui "tourneront" sur votre machine. **LIST** vous initie aux langages informatiques et sélectionne les meilleurs livres pour progresser. **LIST** vous informe de l'actualité et vous fournit trucs, astuces et idées pour mieux programmer...

VOTRE CADEAU
Le **TRANS'BASIC** : table de conversion des instructions BASIC (24 pages - format 11,5x 25 cm).

40 F D'ECONOMIE + 1 CADEAU

BULLETIN D'ABONNEMENT à retourner à **LIST** (service Abonnements)
5, place du Colonel-Fabien - 75491 Paris Cedex 10

Nom : _____ Prénom : _____
Adresse : _____
Ville : _____
Code postal : [] [] [] [] Pays : _____

Veuillez m'abonner pour 10 numéros au prix avantageux de 160 F* au lieu de 200 F et m'adresser en cadeau **LE TRANS'BASIC**. Je fais ainsi une économie de 40 F sur le prix de vente au numéro. Je joins mon règlement indispensable libellé à l'ordre de **LIST**. *Tarif France - autres pays : nous consulter.

BASICIUM,

le Basic enrichi...

Gérard Michel

Apple II+, //e

Cet utilitaire vous permet de :

- Gérer des masques ou écrans** 40 ou 80 colonnes. Ces écrans, définis lors de la programmation, s'affichent par la simple commande **JA**.
- Saisir toutes les variables d'un écran** en une seule commande **Jl** en contrôlant le type et la longueur.
- Recopier l'écran** 40 ou 80 colonnes sur l'imprimante par **JH**.
- Faire un **nettoyage mémoire** (garbage collection) très rapide par **Jf**.
- Gérer des messages**, la réponse étant contrôlée.
- Mais aussi, **ne saisir que l'une des variables affectées à un écran**, les afficher, les effacer, etc...

Tous les fichiers source sont sur la disquette.

(Exemple de démonstration dans le Pom's 13)

Disquette et documentation : 150,00 F TTC franco (bon de commande page 74).

Initiation à l'assembleur (7)

Gérard Michel

Pour mettre un terme à cette série, dont l'univers impitoyable vous aura sans nul doute fascinés, nous vous proposons un dernier exercice de style. Une fois encore, son intérêt pratique vous paraîtra quelque peu limité, hormis certains cas d'application très spécifiques, mais il nous fournira l'occasion d'une large récapitulation. En particulier, nous approfondirons les possibilités d'interaction entre le Basic Applesoft et les routines en langage machine que vous pourriez juger bon de lui adjoindre dans le cadre de vos travaux de programmation.

Pour préciser notre sujet, rappelons tout d'abord que l'on conseille souvent aux programmeurs en Applesoft de placer en début de programme toutes les sous-routines faisant l'objet d'appels fréquents, afin de gagner en vitesse lors de l'exécution. Il n'est pas rare non plus de justifier ce conseil en expliquant que, lorsque l'Applesoft rencontre une instruction de branchement du type GOTO 100, ou GOSUB 50, il recherche la ligne sur laquelle doit s'effectuer le branchement en explorant le programme à partir du début jusqu'à ce qu'il ait trouvé le bon numéro de ligne.

En fait, si le conseil est généralement bon, l'explication correspondante n'est qu'à moitié correcte. En effet, les concepteurs de l'Applesoft n'ont pas oublié de réfléchir en écrivant leur Basic. Ainsi, la recherche de ligne ne part du début du programme que si le poids fort du numéro de la ligne où l'on doit se brancher (après conversion en hexadécimal, bien sûr) est inférieur ou égal au poids fort du numéro de la ligne dans laquelle on rencontre l'instruction de branchement (numéro stocké en \$75-\$76 par l'interpréteur). Dans le cas contraire, cette recherche part de la ligne suivant celle en cours d'exécution et se dirige vers la fin du programme.

Il en résulte que la structure suivante :

```
10 à 1000 : 500 lignes de Basic
1010 GOSUB 1030
1020 GOSUB 1030 : GOTO 1040
1030 : traitement quelconque
1040 : suite du programme...
```

```
sera plus efficace que celle-ci :
10 à 100 : 10 lignes de Basic
110 : traitement quelconque
120 à 1000 : 489 lignes de BASIC
1010 GOSUB 110
1020 GOSUB 110
1030 suite du programme...
```

Pour examiner les choses plus en détails, reportez-vous plus loin à l'analyse de la routine de traitement des instructions GOTO, qui débute en \$D93E. Cette routine est toujours utilisée, directement ou non, dès qu'un branchement doit être effectué, que ce soit par GOTO, GOSUB, ONERR ou ON...

Il est bien évident que ce problème de structure ne pèse significativement sur les performances d'un programme que lorsque les branchements s'avèrent particulièrement nombreux et que la taille de l'ensemble est importante. Dans ce cas, il devient rentable de se préoccuper de la position des différentes routines en fonction de leur fréquence d'appel et des lignes à partir desquelles elles sont appelées.

Par ailleurs, on pourrait s'affranchir de ces problèmes de structure si l'on stockait dans le programme l'adresse de la ligne à laquelle on doit se brancher et non son numéro.

Ainsi, GOSUB 1000 est mémorisé sous la forme :

```
B0 31 30 30 30
```

Si l'on sait que l'adresse de la ligne 1000 dans la zone du programme est \$20B2, si l'on peut alors stocker l'instruction sous la forme :

```
B0 B2 20
```

et faire en sorte que l'interprétation de GOSUB (token B0) puisse utiliser B2 20 pour se brancher directement en \$20B2, peu importent alors la structure du programme et les endroits d'où l'on appelle cette ligne 1000. La phase de recherche des lignes étant dissociée de l'exécution du programme, les branchements se feront toujours à la même vitesse, quelles que soient les positions relatives des points de départ et d'arrivée.

C'est à ce type de manipulation que nous allons donc nous attaquer, ce qui suppose deux catégories de traitements :

– Remplacement dans un programme des numéros de lignes de branchement par leurs adresses effectives.

– Adaptation de l'interpréteur Applesoft pour que les instructions de branchement puissent utiliser directement ces adresses.

Calcul des adresses

Notre programme Applesoft étant chargé en mémoire, il nous faut disposer d'une routine qui puisse l'ex-

plorer du début à la fin, repérer les instructions de branchement, calculer les adresses des lignes correspondantes et remplacer ensuite les numéros de ces lignes par les adresses ainsi obtenues.

Pour nous simplifier la tâche, nous utiliserons la routine de l'Applesoft qui traite le GOTO pour le calcul des adresses. Nous en ferons donc une brève analyse au moyen des deux listes ci-dessous.

*D93ELL

D93E-	20 0C DA	JSR	\$DA0C
D941-	20 A6 D9	JSR	\$D9A6
D944-	A5 76	LDA	\$76
D946-	C5 51	CMP	\$51
D948-	B0 0B	BCS	\$D955
D94A-	98	TYA	
D94B-	38	SEC	
D94C-	65 B8	ADC	\$B8
D94E-	A6 B9	LDX	\$B9
D950-	90 07	BCC	\$D959
D952-	E8	INX	
D953-	B0 04	BCS	\$D959
D955-	A5 67	LDA	\$67
D957-	A6 68	LDX	\$68
D959-	20 1E D6	JSR	\$D61E
D95C-	90 1E	BCC	\$D97C
D95E-	A5 9B	LDA	\$9B
D960-	E9 01	SBC	\$801
D962-	85 B8	STA	\$B8
D964-	A5 9C	LDA	\$9C
D966-	E9 00	SBC	\$800
D968-	85 B9	STA	\$B9
D96A-	60	RTS	

*D61ELL

D61E-	A0 01	LDY	\$801
D620-	85 9B	STA	\$9B
D622-	86 9C	STX	\$9C
D624-	B1 9B	LDA	(\$9B),Y
D626-	F0 1F	BEQ	\$D647
D628-	C8	INY	
D629-	C8	INY	
D62A-	A5 51	LDA	\$51
D62C-	D1 9B	CMP	(\$9B),Y
D62E-	90 18	BCC	\$D648
D630-	F0 03	BEQ	\$D635
D632-	88	DEY	
D633-	D0 09	BNE	\$D63E
D635-	A5 50	LDA	\$50
D637-	88	DEY	
D638-	D1 9B	CMP	(\$9B),Y
D63A-	90 0C	BCC	\$D648
D63C-	F0 0A	BEQ	\$D648
D63E-	88	DEY	
D63F-	B1 9B	LDA	(\$9B),Y
D641-	AA	TAX	
D642-	88	DEY	
D643-	B1 9B	LDA	(\$9B),Y
D645-	B0 D7	BCS	\$D61E
D647-	18	CLC	
D648-	60	RTS	

– Lorsqu'on arrive en \$D93E, \$B8-\$B9 pointe sur le premier octet du numéro de la ligne de branchement, qui est également contenu dans l'accumulateur A (voir l'Initiation à l'assembleur du Pom's 16).

La routine en \$DA0C convertit ce numéro de ligne en hexadécimal et le range en \$50-\$51 (poids faible / poids fort).

– Au retour de la routine en \$D9A6, Y contient le nombre d'octets restant dans la ligne courante après celui

pointé par \$B8-\$B9 (0 de fin de ligne compris).

– On compare ensuite le poids fort du numéro de ligne courante (en \$76) avec celui du numéro de la ligne de branchement (en \$51). Si le premier est supérieur ou égal au second (test BCS positif), la recherche partira du début du programme.

– Si le contenu de \$51 est supérieur à celui de \$76, on calcule $A = (\$B8) + Y + 1$ (notez le SEC avant ADC \$B8), et on met dans X la valeur de \$B9, éventuellement incrémentée de 1 si une retenue s'est dégagée du calcul de A. On a donc en A et X l'adresse du début de la ligne suivante (poids faible / poids fort) et on passe en \$D959.

Dans le cas contraire, on met dans A et X l'adresse du début du programme, pointée par \$67-\$68.

– Sous-routine en \$D61E :

- \$9B-\$9C contiendra toujours l'adresse de début de la ligne en cours d'examen. Lorsqu'on vient de \$D959, on initialise \$9B-\$9C à l'adresse de la ligne marquant le début de la recherche (début du programme, ou ligne suivant celle qui contient l'instruction de branchement).
- En \$D624, on récupère dans A le poids fort de l'adresse de la ligne suivant celle qu'on analyse. Si

cette valeur est nulle, on est en fin de programme et on saute en \$D647, où le CLC (C=0) permettra ensuite de générer un message d'erreur (numéro de ligne inexistant).

Sinon, on compare le poids fort du numéro de ligne cherché à celui du numéro de la ligne en cours.

- Si le test BCC de \$D62E est positif, c'est qu'il y a erreur quelque part, puisque le mode de choix du début de la recherche fait que la valeur de \$51 est normalement supérieure ou égale au poids fort du numéro de la première ligne examinée, et a fortiori des lignes suivantes.
 - Si c'est le même poids fort, on va comparer les poids faibles en \$D635. Sinon, on va passer à la ligne suivante en \$D63E.
 - En \$D635, même processus pour les poids faibles, avec erreur également si le test BCC de \$D63A est positif.
- Si le test BEQ de \$D63C est positif, on a trouvé notre ligne et son adresse de début est en \$9B-\$9C. Sinon, passage à la ligne suivante.
- On va mettre en A et X l'adresse de la ligne suivante et reprendre la routine au début. Le test BCS est obligatoirement positif, puisque l'on vient de passer sans succès

un BCC dans le cas des poids forts (\$D62E) comme dans celui des poids faibles (\$D63A).

– Au retour de \$D61E, en \$D95C, C=0 signale une erreur et on saute au traitement correspondant. Sinon, on met en \$B8-\$B9 l'adresse de la ligne trouvée et contenue en \$9B-\$9C, mais diminuée de 1 afin de respecter les règles d'analyse de l'interpréteur Applesoft (vois Pom's 16).

En résumé, la réalisation d'un branchement en Applesoft se traduit par :

- Recherche de la ligne dont le numéro est donné après l'instruction de branchement.
- Transfert de l'adresse de cette ligne dans la zone de programme dans le pointeur de programme \$B8-\$B9, afin que l'analyse de l'interpréteur reparte ensuite sur cette ligne (après le RTS de \$D96A, on arrivera sur le JMP \$D7D2 de \$D823).

Maintenant que nous savons comment fonctionne la routine de calcul d'adresses, examinons la routine que nous utiliserons pour l'exploiter.

Le programme Applesoft concerné étant chargé en mémoire, nous lancerons l'exécution de la routine dont le source est listé ci-dessous :

Source Big Mac Programme COMPG2.S

```

1 *****
2 *
3 *   REMPLACE LES NO DE LIGNES   *
4 *   DES BRANCHEMENTS PAR LES   *
5 *   ADRESSES DE CES LIGNES     *
6 *   CODE = COMPG2              *
7 *
8 *****
9 *
10          ORG   $300
11          LDA   $67              ;INITIALIS
          ATION POINTEUR DE PROGRAMME
12          STA   $B8
13          LDA   $68
14          STA   $B9
15 S2       LDY   £1              ;POIDS FOR
          T ADRESSE LIGNE SUIVANTE = 0
16          LDA   ($B8),Y         ;IMPLIQUE
          FIN DU PROGRAMME
17          BNE   S0
18          JMP   $D43C
19 S0       STA   $9              ;SALVEGARD
          E ADRESSE LIGNE SUIVANTE
20          DEY
21          LDA   ($B8),Y
22          STA   $8
23          LDY   £2
24          LDA   ($B8),Y         ;MAJ NUMER
          O DE LIGNE COURANTE
25          STA   $75
26          INY
27          LDA   ($B8),Y
28          STA   $76

```

```

29          LDA   $B8              ;PLACE $B8
          -$B9 AU DEBUT DE L'INSTRUCTION
30          CLC
31          ADC   £3
32          STA   $B8
33          BCC   S4
34          INC   $B9
35 S4       JSR   $B1              ;LECTURE C
          ARACTERE DU PROGRAMME
36          BNE   S1
37 S7       CMP   £$3A            ;": " ?
38          BEQ   S4              ;LECTURE C
          ARACTERE SUIVANT
39          LDA   $8
40          STA   $B8              ;PASSE A L
          A LIGNE SUIVANTE
41          LDA   $9
42          STA   $B9
43          BNE   S2
44 S1       CMP   £$AB            ;GOTO ?
45          BEQ   S3
46          CMP   £$B0            ;GOSUB ?
47          BEQ   S3
48          CMP   £$C4            ;THEN ?
49          BNE   S4
50          LDY   £1
51          LDA   ($B8),Y
52          CMP   £$3A
53          BCS   S4              ;CE N'EST
          PAS UN NO DE LIGNE QUI SUIT "THEN"
54          DEY
55          LDA   £$AB
56          STA   ($B8),Y         ;REPLACE
          "THEN" PAR "GOTO"
57 S3       JSR   $B1              ;LECTURE 1
          ER CHIFFRE DU NO DE LIGNE

```



```

58      LDY  $B8      ; SAUVEGARD
      E POINTEUR DE PROGRAMME
59      STY  $6
60      LDY  $B9
61      STY  $7
62      CMP  £$30      ; 0 "BOUCHE
      -TROU" ?
63      BNE  S5
64      JSR  $B1      ; 1ER CHIFF
      RE SIGNIFICATIF
65 S5   JSR  $B7      ; RECHARGE
      ACCUMULATEUR
66      JSR  $D93E      ; ROUTINE "
      GOTO" DE L'APPLESOFT
67      LDY  £0
68      LDA  $B8
69      STA  ($6),Y      ; REMPLACE
      NO DE LIGNE PAR SON ADRESSE
70      INY
71      LDA  $B9
72      STA  ($6),Y

```

```

73      LDA  $6
74      STA  $B8      ; RESTAURE
      POINTEUR DE PROGRAMME
75      LDA  $7
76      STA  $B9
77      INC  $B8      ; $B8-$B9 P
      OINTE SUR 2EME OCTET ADRESSE LIGNE
78      BNE  S6
79      INC  $B9
80 S6   LDY  £0
81 S8   JSR  $B1      ; LECTURE 0
      CTETS SUIVANTS ADRESSE LIGNE
82      BEQ  S7      ; FIN INSTR
      UCTION ?
83      BCS  S3      ; FIN DE "Q
      UEUE" DE NO DE LIGNE
84      LDA  £$3A      ; REMPLACE
      CHIFFRE PAR " : "
85      STA  ($B8),Y
86      BNE  S8

```

- Lignes 11 à 14 : initialisation du pointeur \$B8-\$B9 à l'adresse de début du programme.

- Lignes 15 à 18 : si le poids fort de l'adresse de la ligne suivante est nul, on est en fin de programme et on revient au Basic par un JMP \$D43C.

- Lignes 19 à 22 : on sauve l'adresse de la ligne suivante en \$8-\$9.

- Lignes 23 à 28 : on met le numéro de la ligne en cours de traitement en \$75-\$76.

- Lignes 29 à 34 : on met en \$B8-\$B9 l'adresse du premier octet de la première instruction de la ligne, diminuée de 1 (à 3 octets du début de la ligne).

- Lignes 35 et 36 : lecture d'un caractère de la ligne. Si ce n'est ni 0 ni ":", on passe en S1.

- Lignes 37 et 38 : si c'est ":" (\$3A), on passe au caractère suivant.

- Lignes 39 à 43 : on est en fin de ligne (A=0) et on passe à la suivante.

- Lignes 44 à 47 : si c'est le token de GOTO (\$AB) ou celui de GOSUB (\$B0), on va en S3.

- Lignes 48 et 49 : si ce n'est pas THEN (\$C4), on retourne en S4.

- Lignes 50 à 53 : si l'octet qui suit le token de THEN n'est pas inférieur à \$3A, c'est qu'il ne s'agit pas du début d'un numéro de ligne (instructions du type THEN CC=0). Dans ce cas, on retourne en S4.

- Lignes 54 à 56 : s'il s'agit bien d'un THEN suivi d'un branchement, on remplace le token de THEN par celui de GOTO, afin de n'avoir ensuite qu'un cas à examiner dans la modification de l'interpréteur. Conséquence : n'utilisez pas d'instructions redondantes "THEN GOTO" dans vos programmes "source", car "GOTO GOTO" serait plus tard rejeté par l'interpréteur.

- Lignes 57 à 61 : lecture de l'octet qui suit le token de l'instruction de branchement et sauvegarde de l'adresse de cet octet en \$6-\$7.

- Lignes 62 à 64 : l'adresse d'une ligne occupe toujours deux octets, alors que le numéro de cette ligne, stocké en ASCII, peut n'en prendre qu'un s'il est inférieur à 10. Afin de ne pas avoir à décaler le programme en mémoire dans ce cas, nous adopterons comme convention que, dans le "source" Applesoft, les numéros de ligne inférieurs à 10 doivent être précédés de 0. On écrira ainsi "GOTO 05" et non "GOTO 5". Cette solution n'est certes pas la plus élégante, mais elle simplifie et accélère la routine de calcul d'adresses.

On regarde donc ici si le premier octet du numéro de ligne est un 0 (\$30), auquel cas on doit lire le premier octet significatif de ce numéro par un autre JSR \$B1.

- Lignes 65 et 66 : relecture du premier octet du numéro de ligne par JSR \$B7, afin de s'assurer qu'il est bien chargé dans A et que les indicateurs du registre d'état sont correctement positionnés, puis appel de la routine GOTO qui va calculer l'adresse de la ligne.

- Lignes 67 à 72 : l'adresse de la ligne est en \$B8-\$B9. On la remet à la place des deux premiers octets du numéro de ligne, dont l'adresse est toujours contenue en \$6-\$7. On trouve donc maintenant derrière le token de l'instruction de branchement, l'adresse de la ligne sur laquelle doit se faire ledit branchement.

- Lignes 73 à 76 : restauration du pointeur \$B8-\$B9 à l'adresse du premier octet qui suit le token.

- Lignes 77 à 81 : lecture du premier octet qui suit l'adresse de la ligne que l'on vient de stocker dans le programme.

- Ligne 82 : si l'octet en question est 0 ou ":" (test BEQ positif), on a parcouru tous les octets résiduels de l'ex - numéro de ligne et on arrive à l'instruction suivante. On reprend donc la recherche des instructions de branchement éventuelles.

- Ligne 83 : normalement, ce test BCS ne sera positif que si l'octet lu par \$B1 est inférieur à \$30, et plus précisément s'il s'agit du code de la virgule (\$2C). Tout autre cas relèverait d'une erreur de syntaxe au niveau du programme Applesoft (instructions telles que "GOTO 100 Z = 0", par exemple).

Ce test vise donc à traiter le cas des instructions du type "ON Z GOTO 100, 200, 300". En retournant en S3, on va ainsi calculer les adresses des autres lignes de branchement possibles.

- Lignes 84 à 86 : l'octet lu est cette fois normalement un chiffre faisant partie du numéro de ligne initial (numéros à plus de deux chiffres). On le remplace par un ":" (\$3A) afin d'éviter les erreurs de syntaxe ultérieures lors de l'exécution du programme modifié. Ensuite, on va lire l'octet suivant pour voir si l'on est arrivé ou non à la fin du numéro de ligne.

Lorsque cette routine a terminé le calcul des adresses, elle vous rend la main en mode immédiat du Basic. Si vous faites un LIST du programme désormais en mémoire, vous risquez fort de voir apparaître des instructions très exotiques, comme par exemple :

```
GOTO HGR " : "
GOSUB POP 0 : : ....
```

En effet, une instruction GOTO 10000, qui était stockée sous la forme :

```
AB 31 30 30 30 30
```

le sera maintenant sous la forme :

AB 91 22 3A 3A 3A

si \$2291 est l'adresse, moins 1, de la ligne 10000.

En passant sur cette instruction, la routine de LIST ne pourra que décoder ce qu'elle trouve et vous affichera donc "GOTO HGR "...". N'en soyez pas trop surpris...

Il est évident, par ailleurs, que de telles instructions ne pourront être exécutées sans problème lors d'un RUN que si l'on modifie l'interpréteur Applesoft en conséquence. C'est un point délicat qu'il nous reste maintenant à examiner.

Interprétation des branchements sur adresses

Le programme Applesoft qui se trouve maintenant en mémoire comporte donc des instructions exotiques de branchement sur adresses dont il convient d'assurer l'exécution. Depuis Pom's 16, nous savons qu'il est possible de placer une indirection dans la routine CHRGET, qui commence à l'adresse \$B1, afin de constituer une "annexe" de l'interpréteur, dans laquelle on peut traiter des instructions originales ou modifier le traitement correspondant aux instructions standard. C'est encore la solution que nous adopterons ici pour la manipulation des branchements sur adresses. Ainsi, dans le cas présent, un saut "JMP \$9000" est implanté à l'adresse \$BA, et la routine commençant en \$9000 sera chargée de filtrer les instructions avant de les transmettre, éventuellement intactes, à l'interpréteur.

Avant d'analyser cette routine, il nous faut insister sur deux catégories de problèmes propres à la nature des opérations réalisées.

— La routine CHRGET retourne l'indicateur de retenue (C) à 0 dans le cas où le caractère lu est le code ASCII d'un chiffre (#\$30 à #\$39), et dans ce cas seulement. L'Applesoft utilise cette particularité chaque fois qu'il analyse des instructions qui doivent être suivies d'un nombre, telles que GOTO 100 : après lecture du premier caractère suivant le token de GOTO, la routine d'acquisition du numéro de ligne (\$DA0C) comporte des tests BCS, en retour de CHRGET, qui provoquent une erreur s'ils sont positifs.

Nos adresses de lignes, issues des calculs, comporteront inévitablement des octets à l'extérieur de la plage #\$30 - #\$39, qui fourniront un indicateur C=1 après leur lecture par CHRGET, là où les routines standard de l'Applesoft attendent C=0.

Ceci ne pose guère de problèmes pour l'analyse des tokens "maîtres",

comme dans l'instruction "10 GOTO adresse". En interceptant le GOTO, on sait qu'il nous faut ensuite récupérer l'adresse sans se préoccuper de la valeur du bit C et on peut contrôler l'ensemble du processus.

Il n'en va pas de même pour des instructions du type "10 IF <condition> GOTO adresse". Si l'on ne veut pas refaire ce qui existe, le traitement de IF et l'analyse de la condition seront laissés à l'Applesoft, évidemment, et l'interprétation ultérieure de "GOTO adresse" se fera sous contrôle de la routine de IF, qui se préoccupera de la valeur de C au moment de lire ce qu'elle considère comme un numéro de ligne et de passer la main à la routine de GOTO. De plus, le traitement de IF risque fort, à un moment ou un autre, de lire le GOTO pour vérifier que l'énoncé de la condition est terminé. Si nous interceptons alors le GOTO pour réaliser automatiquement le branchement, il n'y a plus de test et le IF n'est pas correctement exécuté.

La conséquence principale de ce problème réside dans le fait que l'on ne peut pas écrire une routine **générale** d'interception des GOTOs et GOSUBs, basée sur le principe d'une interprétation prioritaire. Une telle routine reposerait sur une interception au niveau du "JSR \$00B1" de l'adresse \$D81D (voir Pom's 16). Etant donné que l'on ne passe pas en \$D81D pour lire GOTO ou GOSUB lorsqu'ils ne sont pas tokens maîtres, il faudrait alors intercepter également les IFs et faire en sorte qu'ils acceptent des branchements sur adresses, en bout de course, si la condition est satisfaite. Cela ne ferait qu'élargir le problème à la modification du traitement d'une autre instruction, en sus des branchements.

Pour tourner la difficulté, nous adopterons une analyse a posteriori : la décision de branchement sera prise lorsque le caractère lu, avant celui que l'on est en train d'examiner, est le token d'une instruction de branchement, et dans la mesure où le sommet de la pile traduit bien le fait que l'interpréteur travaille effectivement sur une telle instruction. Énoncée de la sorte, la chose paraît certes un peu complexe, mais elle le sera moins lorsque nous entrerons dans les détails...

— Nous tenterons de traiter ici tous les branchements, simples comme GOTO, multiples comme ON ... GOTO, mais aussi le ONERR GOTO.

Dans ce dernier cas, il convient, d'une part, de placer l'adresse de la routine Basic de ONERR à l'endroit idoine et, d'autre part, de récupérer le contrôle des opérations lorsqu'une erreur se produit, ce qui suppose que

l'on sache qu'une erreur vient de se produire.

On retrouve en fait un problème général au niveau de notre routine : savoir d'où l'on vient, c'est-à-dire de quelle adresse a été appelée CHRGET et donc notre indirection en \$9000. Comme vous pourrez le constater plus loin, c'est la pile du processeur qui nous fournira les réponses.

Environnement

Le fonctionnement de la routine qui exécute les instructions (\$D7D2) a été détaillé dans l'article précédent.

De plus, nous avons vu plus haut comment procède la routine standard de GOTO. La routine de GOSUB utilise tout simplement la routine GOTO, mais dépose au préalable au sommet de la pile les éléments qui permettront au RETURN de revenir au bon endroit (vous n'aurez aucune peine à analyser la routine GOSUB, qui débute en \$D921). Ces deux routines seront complètement remplacées par les nôtres dans l'indirection \$9000.

Pour clarifier la suite du propos, il peut être bon de préciser par ailleurs comment fonctionnent les routines de IF et de ON ..., et comment se passe la prise en charge d'une erreur lorsque ONERR est actif.

Routine IF

La petite liste ci-dessous, extraite de la ROM Applesoft, constitue le coeur de la routine de IF :

```
*D9C9L
D9C9- 20 7B DD JSR $DD7B
D9CC- 20 B7 00 JSR $00B7
D9CF- C9 AB CMP $A8
D9D1- F0 05 BEQ $D9D8
D9D3- A9 C4 LDA $C4
D9D5- 20 C0 DE JSR $DECO
D9D8- A5 9D LDA $9D
D9DA- D0 05 BNE $D9E1
D9DC- 20 A6 D9 JSR $D9A6
D9DF- F0 B7 BEQ $D998
D9E1- 20 B7 00 JSR $00B7
D9E4- B0 03 BCS $D9E9
D9E6- 4C 3E D9 JMP $D93E
D9E9- 4C 28 D8 JMP $D828
```

\$DD7B est la partie la plus importante (analyse et test de la condition) mais elle ne concerne pas notre sujet.

Après l'analyse du JSR \$DD7B, on recharge en A le caractère pointé par \$B8-\$B9 (JSR \$B7), qui peut être GOTO (\$AB), et doit être THEN (\$C4) si ce n'est pas GOTO (sinon, \$DECO finira sur une SYNTAX ERROR. Si c'est THEN, \$DECO permettra de lire l'octet qui suit le THEN).

Si la condition n'est pas satisfaite, \$9D contient 0. Dans ce cas, on met le nombre d'octets restants dans la ligne (après celui pointé par \$B8-\$B9) dans Y par JSR \$D9A6. Le dernier caractère lu par \$D9A6 est le

0 de fin de ligne et le test BEQ amène donc en \$D998 qui pointe \$B8-\$B9 sur l'octet précédant la ligne suivante et amène au JMP \$D7D2 de l'adresse \$D823 par un simple RTS final. On vérifie ainsi que "condition non satisfaite" se traduit en Applesoft par "passage à la ligne suivante".

Si la condition est satisfaite, on arrive en \$D9E1. On relit le caractère pointé par \$B8-\$B9 (JSR \$B7), qui peut être soit le token de GOTO, soit l'octet suivant le token de THEN (c'est-à-dire un autre token, le premier chiffre d'un numéro de ligne ou le premier caractère d'un nom de variable). Si c'est un chiffre, C=0 au retour de \$B7, et on sort par JMP \$D93E (THEN 100 est donc bien traité comme GOTO 100). Si ce n'est pas un chiffre (C=1), on sort par JMP \$D828; on revient donc dans l'interpréteur comme si l'on venait de lire une variable ou un token "maître" (comme si l'on venait du JSR \$D828 de l'adresse \$D820).

Dans notre routine de calcul d'adresses, nous avons fait en sorte de n'avoir plus que des instructions du type "IF ... GOTO adresse", qui aboutiront donc en \$D828, exactement comme nos instructions directes "GOTO adresse". De même, les instructions " ... THEN GOSUB adresse" arriveront en \$D828 (comme un GOSUB simple) lorsque se posera le problème de réaliser le branchement.

Cette brève analyse de IF montre par ailleurs que l'on ne pourrait effectivement pas intercepter le GOTO directement au moment où il est lu par CHRGET : le JSR \$B7 de l'adresse \$D9CC, par exemple, provoquerait alors le branchement avant même que l'on sache si la condition est satisfaite pour justifier ce branchement. De plus, le fait que l'on passe par \$B7 pour relire le token de GOTO prouve que \$B8-\$B9 pointe déjà sur GOTO, qui a donc été lu, dans les traitements recouverts par JSR \$DD7B, au moyen d'un saut à \$B1. Les risques de branchement prématuré sont ainsi fort nombreux et justifient un traitement a posteriori, via le passage par \$D828 qui se termine, rappelons-le, par un JMP \$B1.

Routine de ON

Vous trouverez ci-dessous un bref extrait de la routine ON :

```
*D9ECL
D9EC- 20 F8 E6 JSR  $E6F8
D9EF- 48 PHA
D9F0- C9 B0 CMP  $B0
D9F2- F0 04 BEQ  $D9F8
D9F4- C9 AB CMP  $AB
D9F6- D0 89 BNE  $D981
D9F8- C6 A1 DEC  $A1
D9FA- D0 04 BNE  $DA00
D9FC- 68 PLA
D9FD- 4C 2A D8 JMP  $D82A
```

```
DA00- 20 B1 00 JSR  $00B1
DA03- 20 0C DA JSR  $DA0C
DA06- C9 2C CMP  $2C
DA08- F0 EE BEQ  $D9F8
DA0A- 68 PLA
DA0B- 60 RTS
DA0C- A2 00 LDX  $00
DA0E- 86 50 STX  $50
DA10- 86 51 STX  $51
DA12- B0 F7 BCS  $DA0B
```

La sous-routine appelée par JSR \$E6F8 se charge d'évaluer l'expression dans l'instruction "ON <expression> GOTO / GOSUB". Au retour, la valeur concernée est stockée en \$A1, et \$B8-\$B9 pointe sur le premier octet qui suit l'expression (cet octet est également en A).

L'octet suivant l'expression est empilé. Ce peut être le token de GOSUB (\$B0) ou celui de GOTO (\$B7); sinon : SYNTAX ERROR.

En \$D9F8, on décrémente \$A1. Si la valeur de \$A1 tombe alors à 0, on dépile le token de l'instruction et on retourne dans l'interpréteur en \$D82A, c'est-à-dire, une fois encore, comme si l'on venait de lire GOTO ou GOSUB comme token "maître". Ainsi, si \$A1 contenait la valeur 1, on passe à 0 au premier DEC \$A1 et on saute en \$D82A en pointant sur GOTO ou GOSUB : l'interprétation du token réalisera alors le branchement sur le numéro de ligne qui suit l'octet pointé par \$B8-\$B9 à l'entrée dans \$D82A, donc le premier numéro donné dans l'instruction ON, ce qui est conforme aux résultats observés lorsque vous utilisez ce type d'instruction.

Si \$A1 ne tombe pas à 0, on lit l'octet suivant (JSR \$B1) et on appelle \$DA0C. Cette routine se charge d'évaluer le numéro de ligne pointé par \$B8-\$B9 à l'entrée. Elle procède par des JSR \$B1 successifs, jusqu'à ce que \$B1 lui retourne un C=1 (caractère lu n'est pas un chiffre). Au retour, A contient l'octet qui suit le numéro de ligne, octet par ailleurs pointé par \$B8-\$B9.

Normalement, le séparateur des numéros de ligne dans un ON est la virgule (\$2C). Si l'on récupère bien une virgule en retour de \$DA0C, on retourne alors en \$D9F8. Ainsi, si \$A1 contenait 2 au départ, on est passé à 1 lors du premier DEC \$A1. On passe donc le premier numéro de ligne et on pointe sur la première virgule après JSR \$DA0C. En retournant à \$D9F8, \$A1 tombe cette fois à 0 et on saute à \$D82A. Comme A contient le token de l'instruction GOTO ou GOSUB, l'analyse du token et le choix de la routine correspondante se fera correctement. Le branchement sera ensuite réalisé sur le numéro de ligne qui suit l'octet pointé par \$B8-\$B9 à l'entrée dans \$D82A, c'est-à-dire le numéro qui suit la première virgule, ou encore le second numéro indiqué, qui correspond bien à la valeur 2 pour l'ex-

pression de "ON <expression> ...".

Ce processus d'avancée du pointeur \$B8-\$B9 se poursuivra ainsi toujours jusqu'à la virgule qui précède le numéro de ligne dont le rang dans la liste des numéros de ligne correspond à la valeur de l'expression.

Par contre, si l'on ne récupère pas une virgule en retour de \$DA0C, on dépile simplement le token et on sort par RTS. Comme l'interprétation se fait sous le contrôle du JSR \$D828 de l'adresse \$D820, on revient en \$D823 qui nous renvoie en \$D7D2 pour l'analyse d'une autre instruction. Cette situation résulte soit d'une valeur nulle pour "expression", soit d'une valeur supérieure au nombre de numéros de ligne donnés dans la liste (la valeur nulle relève du même principe, puisqu'on part de \$FF après le premier DEC \$A1). Dans ce cas, on a avancé le pointeur \$B8-\$B9 jusqu'au premier octet qui suit la liste des numéros de ligne (normalement "." ou 0 de fin de ligne) : par \$D7D2 on passe donc à l'instruction ou à la ligne suivante. On vérifie bien que, si "expression" vaut 0 ou une valeur supérieure au nombre d'items prévus, l'instruction ON "expression" ... fait simplement passer à la suite, sans réaliser de branchement.

Pour mémoire, notons qu'une erreur de syntaxe dans le ON (mauvais séparateur de numéros de ligne) provoquerait également la sortie par RTS. Mais on ne serait pas alors dans une situation conforme pour \$D7D2, d'où SYNTAX ERROR.

Dans notre indirection, nous n'intercepterons pas les "ON", afin de ne pas devoir refaire tout ce qui précède la phase de branchement proprement dite. En revanche, nous prendrons la main lors du JSR \$B1 de l'adresse \$DA00, afin de déplacer nous-mêmes le pointeur \$B8-\$B9 et l'amener jusqu'à l'adresse de branchement adéquate. Quant au branchement, nous l'intercepterons ensuite a posteriori, via le passage de ON par \$D82A, selon les mêmes principes que pour IF.

Prise en charge d'erreur par ONERR

Lorsque ONERR est actif, la surveillance d'une erreur se traduit finalement par un branchement en \$F2E9. Cette routine rétablit certains paramètres pour donner la main au sous-programme de traitement des erreurs et, en particulier, elle met en \$B8-\$B9 le contenu des adresses \$F4-\$F5. Ces deux adresses contiennent l'adresse du numéro de la première ligne de traitement d'erreur donnée après l'instruction ONERR GOTO (par exemple, l'adresse du premier chiffre de 9000 si vous avez indiqué ONERR GOTO 9000). La mise en

place de \$F4-\$F5 est réalisée lors de l'interprétation de ONERR GOTO.

Ensuite, la routine repositionne les registres et les indicateurs du registre d'état sur le premier caractère de la première ligne de traitement d'erreur par un JSR \$B7 (on est toujours sur le 9 de ONERR GOTO 9000, par exemple). On exécute alors un GOTO par JSR \$D93E : le branchement se fait donc comme s'il s'agissait d'un GOTO 9000, traité en cas d'erreur, ce que traduit bien finalement l'instruction ONERR GOTO ... A l'issue du JSR \$D93E, \$B8-\$B9 pointe sur l'octet précédant la première ligne de traitement d'erreur (sur l'octet précédant la ligne 9000, par exemple), et on saute directement à l'exécution de cette ligne par JMP \$D7D2.

La fin de ce processus est résumée ci-dessous :

F30F-JSR \$00B7
F312-JSR \$D93E
F315-JMP \$D7D2

Notre indirection se chargera d'intercepter l'ordre initial de ONERR GOTO pour mettre directement en \$F4-\$F5 l'adresse (moins 1) de la première ligne de traitement d'erreur. On aurait ainsi de suite l'adresse de l'octet précédant la ligne 9000 et non l'adresse du 9 dans un ONERR GOTO 9000.

Ensuite, c'est au niveau du JSR \$B7 de l'adresse \$F30F que nous pourrions reprendre à notre compte le traitement d'erreur effectif, c'est-à-dire assurer le branchement au sous-programme de traitement d'erreur lorsque celle-ci s'est effectivement produite.

Indirection dans l'interpréteur

Le source de cette routine est donné en fin d'article.

Tout appel à \$B1 ou \$B7 nous amènera donc en \$9000; l'essentiel du travail consiste alors à filtrer les informations pour déterminer s'il nous faut prendre en charge les branchements ou restituer les octets lus sans rien faire. Dans cet effort d'analyse, la pile et son pointeur constitueront nos principaux soutiens. Rappelons, par ailleurs, qu'il est bon d'avoir bien étudié les mécanismes du RUN pour se pencher sur cette nouvelle routine.

Les indications ci-dessous vous permettront de compléter les commentaires du listing.

– Lignes 14 à 25 : l'appel à CHRGET vient-il de la routine de traitement du ON, ou encore, s'agit-il du JSR \$B1 de l'adresse \$DA00 ? Si tel est le cas, le sommet de la pile (plus précisément les deux octets juste en dessous du sommet) doivent correspondre à l'adresse de retour

moins 1 de ce JSR, soit \$DA02 (le \$02 se situant au-dessus du \$DA dans la pile).

Si tel est le cas, nous irons chercher les virgules éventuelles (JSR VIR), nous "sortirons" le JSR \$B1 de la pile par deux PLA, nous chargerons A avec le code de la virgule (\$#2C) et nous laisserons l'Applesoft continuer le travail en retournant en \$DA06. En résumé, on remplace la sous-routine Applesoft de positionnement sur virgules par la nôtre, qui doit tenir compte de ce que nous mémorisons des adresses de lignes et non des numéros de lignes.

– Lignes 26 à 29 : le seul token traité "en direct" (au niveau du JSR \$B1 de l'adresse \$D81D) est celui de ONERR (\$#A5). Nous devons alors placer les paramètres de branchement sur le sous-programme de traitement d'erreur (JMP ON1).

– Lignes 30 à 36 : traitement de l'erreur effective lorsque ONERR est actif. Dans ce cas, on doit venir du JSR \$B7 de l'adresse \$F30F (sommet de la pile égale retour en \$F311). Branchement sur notre routine ERR.

– Lignes 37 à 44 : analyse a posteriori des instructions de branchement. \$6-\$7 contiennent toujours l'adresse de l'octet lu par CHRGET avant celui que l'on est en train de lire par ce même CHRGET. Le branchement ne se fera que si le caractère précédant est un GOTO (\$#AB), un GOSUB (\$#B0) ou une virgule (\$#2C - traitement des ON ... lorsque le branchement ne se fait pas sur la première ligne donnée dans la liste).

Ceci permet d'éviter les branchements prématurés. Si l'on est encore sur la "même ligne" que l'instruction de branchement, sur laquelle on est déjà passé sans rien faire, c'est sans doute que l'Applesoft veut réellement exécuter un branchement et qu'il en cherche l'adresse. Il nous reste en sus un second verrou, vis-à-vis de ces branchements prématurés, que nous examinerons plus loin.

– Lignes 45 à 55 : l'octet lu n'est pas pour nous... On en stocke l'adresse en \$6-\$7 (il peut en effet s'agir du token d'une instruction qui nous intéresse), on restaure les registres et on reprend le cours normal du CHRGET standard.

– Lignes 56 à 64 : second verrou des branchements. Tous les branchements, simples ou complexes (IF ou ON), passent par \$D828 ou \$D82A dans l'interpréteur. On doit arriver alors du JMP \$B1 de l'adresse \$D83C, avec au sommet de la pile l'adresse moins 1 de la routine de traitement de GOTO (\$D93E) ou de GOSUB (\$D921).

Si tel est bien le cas, il devient cer-

tain que le branchement doit être effectué, en fonction des résultats obtenus précédemment par l'Applesoft (on est arrivé jusqu'au point où l'Applesoft se met à la recherche des adresses des numéros de lignes).

– Lignes 65 à 77 : traitement GOTO. On "sort" de la pile l'adresse moins 1 de la routine de GOTO. On trouve dans les deux octets consécutifs pointés par \$B8-\$B9 (là où serait le premier chiffre du numéro de ligne) l'adresse moins 1 de la ligne de branchement, que l'on transfère en \$B8-\$B9 (et en \$6-\$7 pour archivage de l'adresse du dernier octet lu).

On est toujours sous contrôle du JSR \$D828 de l'adresse \$D820 (directement ou non, comme on l'a vu en analysant IF et ON). Le RTS nous amène en \$D823, soit sur JMP \$D7D2 pour passer à l'instruction suivante pointée par \$B8-\$B9, c'est-à-dire la ligne de branchement. Mission accomplie !

– Lignes 78 à 98 : traitement de GOSUB : principes identiques à GOTO.

En sus, il faut toutefois contrôler la place disponible dans la pile et empiler ensuite :

- l'adresse de retour après exécution de la sous-routine (nous prenons ici l'adresse du premier "..." qui suit l'adresse de la sous-routine après GOSUB);
- le numéro de la ligne où l'on doit revenir (adresse de la ligne courante mémorisée en \$75-\$76);
- le token de GOSUB (\$#B0) pour éviter ensuite les erreurs "RETURN WITHOUT GOSUB".

On passe ensuite directement à la sous-routine par JMP \$D7D2, et non par RTS, car c'est la routine de traitement du RETURN qui est conçue dans l'Applesoft pour assurer la sortie du JSR \$D828 de l'adresse \$D820.

– Lignes 99 à 126 : mise en place des paramètres pour ONERR GOTO. On vérifie tout d'abord que l'on vient effectivement de l'analyse du token maître ONERR (sommet de la pile égale \$D81F si l'on vient du JSR \$B1 de \$D81D). Dans ce cas, on trouve l'adresse deux octets plus loin que la position pointée par \$B8-\$B9 (ONERR pour Y=0, GOTO pour Y=1, premier octet de l'adresse pour Y=2).

On stocke cette adresse du sous-programme de traitement d'erreur en \$F4-\$F5, on passe directement au premier octet qui suit cette adresse en ajoutant 4 à \$B8-\$B9 (plus archivage en \$6-\$7) et on signale à l'Applesoft que ONERR est actif en mettant \$#80 à l'adresse (drapeau) \$D8. C'est dans la mesure où ce drapeau est positionné à \$#80 que l'arrivée d'une erreur nous enverra en \$F2E9

(voir plus haut).

On sort ensuite de notre JSR \$B1 par deux PLA et on passe directement à la suite par JMP \$D7D2, puisque l'on a réalisé ici tout le traitement nécessaire pour notre propre manipulation des ONERR GOTO et des erreurs éventuelles.

- Lignes 127 à 133 : traitement des erreurs effectives. Le traitement standard Applesoft a déjà remis en \$B8-\$B9 la valeur de \$F4-\$F5 (adresse première ligne sous-programme d'erreur) et il nous reste à l'archiver en \$6-\$7. On restaure ensuite le pointeur de pile à la valeur qu'il avait lors de la rentrée en \$D7D2 (stockée en \$F8) pour analyse de l'instruction au cours de laquelle l'erreur s'est produite. En remontant en ONO (JMP \$D7D2) on commence directement l'exécution du sous-programme de traitement d'erreur.

A noter que notre manipulation de ONERR permet une utilisation répétée sans les problèmes de "OUT OF MEMORY" que l'on rencontre avec le traitement standard de l'Applesoft.

- Lignes 134 à 156 : recherche des virgules dans le cadre des instructions ON ...

Comme les adresses occupent toujours deux octets, la recherche peut commencer à deux octets de la position courante (token de GOTO ou GOSUB, ou précédente virgule), d'où un LDY #2 au départ.

Un numéro de ligne occupant au maximum 5 octets (soit deux octets d'adresse et trois "." après notre calcul d'adresses), on doit trouver normalement une virgule avant que Y atteigne la valeur 6. Si tel n'est pas le cas, il ne peut y avoir de branchement et on doit passer à l'instruction suivante en séquence, comme dans le traitement standard. On exécute donc deux PLA pour sortir du JSR VIR de notre routine, deux PLA pour sortir du JSR \$B1 de l'adresse \$DA00, un PLA pour récupérer le token empilé par la routine de ON. Le RTS de la ligne 156 nous enverra donc en \$D823 (retour du JSR \$D828 de l'adresse \$D820) pour passer à l'instruction suivante. De plus, on remet Y à 2, pour pointer sur le premier octet qui suit la dernière adresse de ligne donnée dans la liste (en principe, on a avancé régulièrement \$B8-\$B9 pour le faire pointer sur la virgule que l'on vient de trouver - voir ci-dessous. En partant de la dernière virgule trouvée, la fin de l'instruction ON GOTO ... se trouve deux octets plus loin, juste

après la dernière adresse de ligne donnée dans la liste).

Les lignes 147 à 155 permettent de positionner \$B8-\$B9 sur la virgule trouvée et d'en archiver l'adresse (on ajoute la valeur de Y à la précédente valeur de \$B8-\$B9). C'est approximativement le même traitement que celui réalisé par l'Applesoft dans sa routine \$DA0C. On revient ensuite pour redonner la main à la routine de ON, qui se chargera de savoir si l'on est arrivé ou non à la bonne adresse de branchement (à la bonne virgule).

Dans le cas du non branchement (expression vaut 0 ou supérieure au nombre d'items prévus), les lignes 147 à 155 permettent de faire pointer \$B8-\$B9 sur le premier octet qui suit la fin de l'instruction ON ...

Mode opératoire

Pour tester ce système de branchement sur adresses, nous vous conseillons de disposer sur une même disquette des codes objets des deux routines concernées. Sur la disquette Pom's, la routine de calcul des adresses est baptisée COMPG2 et celle d'analyse des instructions de branchement INTERG11.

Comme dans les exemples évoqués dans le Pom's 16, constituez également un petit fichier pour mettre en place l'indirection dans CHRGET. Ce fichier contiendra simplement les codes 4C 00 90 (JMP \$9000) et devra se charger à l'adresse \$BA. Un tel fichier est fourni sur la disquette d'accompagnement sous le nom INTER6.

Un second fichier permettra de rétablir CHRGET à sa structure normale. Il devra se charger également en \$BA et contiendra les 4 octets C9 3A B0 0A (CMP #\$3A et BCS \$00C8). Ce fichier s'appelle CHRGET sur la disquette d'accompagnement.

Constituez ensuite un fichier EXEC (tel COMPDEF sur la disquette) qui comprendra les instructions suivantes :

```
HIMEM:9*4096
BRUN COMPG2
BLOAD INTERG11
BLOAD INTER6
```

Une fois ce fichier présent sur la même disquette que les autres, chargez ou entrez au clavier votre programme "source" en Applesoft (sauvez-le sous cette forme normale si ce n'est déjà fait).

Tapez ensuite EXEC COMPDEF, pour faire exécuter le calcul des

adresses et mettre en place l'indirection dans CHRGET. Listez le résultat de ce travail, ne serait-ce que par curiosité...

Vous pouvez enfin faire un RUN du programme adapté en mémoire et contrôler le déroulement des opérations.

Pour revenir à l'Applesoft standard, tapez, en mode immédiat par exemple, BLOAD CHRGET (le programme en mémoire, avec les adresses calculées, ne pourra dès lors plus être exécuté sans erreur de syntaxe, à moins que vous ne rechargez l'aiguillage par BLOAD INTER6).

Un programme comportant des adresses en lieu et place des numéros de ligne ne peut généralement pas être sauvé sur disquette correctement, car la commande SAVE n'est pas prévue pour traiter des programmes comportant des 0 ailleurs qu'en fin de ligne (alors qu'on peut trouver des 0 dans nos adresses de lignes). Ceci justifie l'emploi d'un fichier EXEC pour traiter automatiquement un programme source en mémoire, chaque fois que l'on veut tester son fonctionnement avec des branchements sur adresses.

Comme vous pourrez le constater, ce mécanisme de branchement ne se traduit par des gains de temps réels que dans certaines structures de programme bien particulières : nombreux branchements sur des lignes situées très loin de leurs points d'appel. Dans les autres cas, le temps perdu à analyser chaque octet du programme dans la routine \$9000, que celui-ci la concerne ou non, annule le temps gagné sur les branchements. Cet exercice de style nous aura donc surtout servi de prétexte pour approfondir le fonctionnement de l'Applesoft et les processus de "greffe" que l'on peut lui appliquer par le biais de l'assembleur et du langage machine. Nul doute que l'utilité de la pile, en tant qu'outil de mémorisation des opérations passées et d'analyse de ces opérations, vous apparaîtra plus nettement après cet usage intensif du pointeur de pile.

L'étude en profondeur des routines de l'Applesoft pourra vous être souvent utile, si vous désirez persévérer dans la chirurgie plastique du langage de base de votre Apple. Afin de vous y aider un peu, vous trouverez en annexe la liste des points d'entrée des routines de traitement des tokens "maîtres". Vous serez ainsi en possession des clés, à vous d'ouvrir les portes...

Source Big Mac Programme INTERG 11. S

```
1 *****
2 *
3 * ROUTINE D'INTERPRETATION DES *
```

```
4 * INSTRUCTIONS DE BRANCHEMENT *
5 * SUR UNE ADRESSE *
6 * CODE = INTERG11 *
7 *
8 *****
9 *
```

```

10      ORG $9000
11      STA $BD          ;STOCKAGE
      CARACTERE ANALYSE
12      STY $8          ;SAUVEGARD
      E DES REGISTRES
13      STX $9
14      TSX
15      LDA $102,X      ;VIENT-ON
      DE LA ROUTINE DU "ON" ?
16      CMP £$DA
17      BNE ON2
18      LDA $101,X
19      CMP £$02
20      BNE ON2
21      JSR VIR          ;RECHERCHE
      DE L'ADRESSE DE BRANCHEMENT
22      PLA
23      PLA
24      LDA £$2C        ;", "
25      JMP $DA06       ;RETOUR A
      LA ROUTINE DE "ON"
26 ON2   LDA $BD
27      CMP £$A5        ;TOKEN DE
      "ONERR" ?
28      BNE SU00
29      JMP ON1
30 SU00  LDA $102,X      ;VIENT-ON
      DE LA ROUTINE QUI BRANCHE SUR
31 SU0   CMP £$F3        ;LA ROUTIN
      E D'ERREUR SI "ONERR"
32      BNE V00         ;(TRAITEME
      NT D'UNE ERREUR EFFECTIVE)
33      LDA $101,X
34      CMP £$11
35      BNE V00
36      JMP ERR
37 V00   LDY £0
38      LDA (£6),Y      ;LE CARACT
      ERE PRECEDENT EST-IL :
39      CMP £$AB        ;"GOTO" ?
40      BEQ SSD
41      CMP £$B0        ;"GOSUB" ?
42      BEQ SSD
43      CMP £$2C        ;", " ?
44      BEQ SSD
45 SDEB  LDA $B8        ;M.A.J. PO
      INTEUR $6-$7
46      STA $6
47      LDA $B9
48      STA $7
49      LDY $8
50      LDX $9
51      LDA $BD        ;REPRISE N
      ORMALE DE "CHRGET"
52      CMP £$3A
53      BCS S1
54      JMP $BE
55 S1    RTS
56 SSD   LDA $102,X      ;LE SOMMET
      DE LA PILE EST-IL L'ADRESSE DE
57      CMP £$D9        ;LA ROUTIN
      E DE "GOTO" ($D93E-1)
58      BNE SDEB
59 SSD1  LDA $101,X
60      CMP £$3D
61      BEQ SS0
62      CMP £$20        ;OU DE "GO
      SUB" ($D921-1)
63      BEQ S2
64      BNE SDEB
65 SS0   PLA          ;DEPILE SO
      MMET
66      PLA

```

```

67 S0    LDY £0          ;MET $B8-$
      B9 A L'ADRESSE DE BRANCHEMENT
68 SS1   LDA ($B8),Y
69      PHA
70      INY
71      LDA ($B8),Y
72      STA $B9
73      STA $7
74      PLA
75      STA $B8
76      STA $6
77      RTS
78 S2    PLA          ;TRAITEMEN
      T "GOSUB"
79      PLA
80      LDA £$3
81      JSR $D3D6       ;CONTROLE
      PLACE DISPONIBLE DANS LA PILE
82      LDA $B8        ;CALCULE E
      T EMPILE LES VALEURS CORRECTES
83      CLC          ;POUR LE "
      RETURN" A VENIR
84      ADC £2
85      STA $6
86      LDA $B9
87      ADC £0
88      PHA
89      LDA $6
90      PHA
91      LDA $76
92      PHA
93      LDA $75
94      PHA
95      LDA £$B0
96      PHA
97      JSR S0        ;M.A.J. DE
      $B8-$B9
98      JMP $D7D2       ;SAUTE A L
      A LIGNE DU "GOSUB"
99 ON1   LDA $102,X
100     CMP £$D8        ;VIENT-ON
      DU JSR $B1 DE
101     BEQ ON10       ;L'ADRESSE
      $D81D (TOKEN MAITRE)
102     JMP SU0
103 ON10 LDA $101,X
104     CMP £$1F
105     BEQ ONERR
106     JMP SU00
107 ONERR LDY £2        ;ADRESSE D
      E DEBUT-1 DE LA PREMIERE LIGNE
108     LDA ($B8),Y    ;DE LA ROU
      TINE BASIC APPELEE PAR "ONERR GOTO
      "
109     STA $F4        ;EST MISE
      EN $F4-$F5
110     INY
111     LDA ($B8),Y
112     STA $F5
113     LDA $B8        ;PASSE A L
      A FIN DE L'INSTRUCTION
114     CLC          ;"ONERR GO
      TO ADRESSE"
115     ADC £4
116     STA $B8
117     STA $6
118     LDA $B9
119     ADC £0
120     STA $B9
121     STA $7
122     LDA £$80
123     STA $D8        ;SIGNALA "
      ONERR" ACTIF

```

```

124      PLA                ;SORT DU *
      JSR $B1" DE $D81D
125      PLA
126 ONO   JMP $D7D2        ;PASSE A L
      'INSTRUCTION SUIVANTE
127 ERR   LDA $F4          ;SAUT A LA
      ROUTINE BASIC DE TRAITEMENT DES E
      RREURS
128      STA $6
129      LDA $F5
130      STA $7
131      LDX $F8
132      TXS
133      BNE ONO
134 VIR   LDY £2           ;CHERCHE V
      IRGULES DANS INSTRUCTION *ON GOTO
      ...
135 VIR1  LDA ($B8),Y
136      CMP £$2C
137      BEQ VIR2
138      INY
139      CPY £6           ;SI PAS DE
      ", " A 5 OCTETS DU TOKEN DE GOTO

```

```

140      BNE VIR1          ;OU GOSUB
      --> ERREUR
141      PLA                ;MAIS PASS
      AGE A INSTRUCTION SUIVANTE SI
142      PLA                ;VARIABLE=
      0 OU SUPERIEURE AU NOMBRE D'ITEMS
143      PLA
144      PLA
145      PLA
146      LDY £2
147 VIR2  TYA                ;PLACE $B8
      -$B9 SUR UNE VIRGULE DANS L'INSTRU
      CTION
148      CLC                ;*ON GOTO
      ...
149      ADC $B8
150      STA $B8
151      STA $6
152      BCC VIR3
153      INC $B9
154 VIR3  LDA $B9
155      STA $7
156      RTS

```

COMPG2

*300.39A

```

0300- A5 67 85 B8 A5 68 85 B9
0308- A0 01 B1 B8 D0 03 4C 3C
0310- D4 85 09 88 B1 B8 85 08
0318- A0 02 B1 B8 85 75 C8 B1
0320- B8 85 76 A5 B8 18 69 03
0328- 85 B8 90 02 E6 B9 20 B1
0330- 00 D0 0E C9 3A F0 F7 A5
0338- 08 85 B8 A5 09 85 B9 D0
0340- C7 C9 AB F0 15 C9 B0 F0
0348- 11 C9 C4 D0 E1 A0 01 B1
0350- B8 C9 3A B0 D9 88 A9 AB
0358- 91 B8 20 B1 00 A4 B8 84
0360- 06 A4 B9 84 07 C9 30 D0
0368- 03 20 B1 00 20 B7 00 20
0370- 3E D9 A0 00 A5 B8 91 06
0378- C8 A5 B9 91 06 A5 06 85
0380- B8 A5 07 85 B9 E6 B8 D0

```

```

0388- 02 E6 B9 A0 00 20 B1 00
0390- F0 A1 B0 C6 A9 3A 91 B8
0398- D0 F3 FF

```

INTERG 11

*9000.9117

```

9000- 85 BD 84 08 86 09 BA BD
9008- 02 01 C9 DA D0 11 BD 01
9010- 01 C9 02 D0 0A 20 F2 90
9018- 68 68 A9 2C 4C 06 DA A5
9020- BD C9 A5 D0 03 4C AC 90
9028- BD 02 01 C9 F3 D0 0A BD
9030- 01 01 C9 11 D0 03 4C E5
9038- 90 A0 00 B1 06 C9 AB F0
9040- 1E C9 B0 F0 1A C9 2C F0
9048- 16 A5 B8 85 06 A5 B9 85
9050- 07 A4 08 A6 09 A5 BD C9
9058- 3A B0 03 4C BE 00 60 BD
9060- 02 01 C9 D9 D0 E3 BD 01

```

```

9068- 01 C9 30 F0 06 C9 20 F0
9070- 16 D0 D6 68 68 A0 00 B1
9078- B8 48 C8 B1 B8 85 B9 85
9080- 07 68 85 B8 85 06 60 68
9088- 68 A9 03 20 06 D3 A5 B8
9090- 18 69 02 85 06 A5 B9 69
9098- 00 48 A5 06 48 A5 76 48
90A0- A5 75 48 A9 B0 48 20 75
90A8- 90 4C D2 D7 B0 02 01 C9
90B0- D8 F0 03 4C 28 90 BD 01
90B8- 01 C9 1F F0 03 4C 28 90
90C0- A0 02 B1 B8 85 F4 C8 B1
90C8- B8 85 F5 A5 B8 18 69 04
90D0- 85 B8 85 06 A5 B9 69 00
90D8- 85 B9 85 07 A9 80 85 D8
90E0- 68 68 4C D2 D7 A5 F4 85
90E8- 06 A5 F5 85 07 A6 F8 9A
90F0- D0 F0 A0 02 B1 B8 C9 2C
90F8- F0 0C C8 C0 06 D0 F5 68
9100- 68 68 68 68 A0 02 98 18
9108- 65 B8 85 B8 85 06 90 02
9110- E6 B9 A5 B9 85 07 60 00

```

F1D5	CALL	F6E9	HCOLOR=	F273	NORMAL	D96B	RETURN
D66A	CLEARR	F3E2	HGR	F26F	NOTRACE	F721	ROT=
F24F	COLOR=	F3D8	HGR2	D649	NEW	D912	RUN
D896	CONT	F286	HIMEM:	DCF9	NEXT	D8B0	SAVE
D995	DATA	F232	HLIN	D9EC	ON	F727	SCALE=
E313	DEF	FC58	HOME	F2CB	ONERR	F775	SHLORD
F331	DEL	F6FE	HPLLOT	F225	PLOT	F262	SPEED=
DFD9	DIM	F7E7	HTRB	E77B	POKE	D86E	STOP
F769	DRAW	D9C9	IF	D96B	POP	F39F	STORE
D870	END	F1DE	IN#	F1E5	PR#	F399	TEXT
F280	FLASH	DBB2	INPUT	DAD5	PRINT	F26D	TRACE
D766	FOR	F277	INVERSE	D8E2	READ	F241	ULIN
DBA0	GET	DA46	LET	F3BC	RECALL	F256	UTAB
D921	GOSUB	D6A5	LIST	D9DC	REM	E784	WAIT
D93E	GOTO	D8C9	LOAD	D849	RESTORE	F76F	WDRAW
F390	GR	F2A6	LOMEM:	F318	RESUME	D3F5	Ø

HGR DUMP sur Epson

Alexandre Avrane

Fatigué de n'utiliser votre imprimante graphique qu'au quart de ses possibilités ? Ce programme, prévu pour les imprimantes Epson mais adaptable à toutes imprimantes à aiguilles, apporte les possibilités d'une carte interface graphique... sans le coût.

Voici les fonctions possibles :

- impression simultanée (sur le même bloc horizontal) de un à trois écrans haute-résolution de l'Apple
- tabulations horizontales précédant l'impression des écrans
- rotation de l'image imprimée de 0, 90, 180 ou 270 degrés
- zoom possible dans les rapports 1/1, 2/1, 4/1
- impression en simple ou double densité
- impression normale ou inversée (noir / blanc)
- définition possible de la fenêtre de l'écran haute-résolution à imprimer.

Exécution du programme

Le programme s'initialise par un simple BRUN HGR DUMP. Il utilise l'omniprésent vecteur de l'ampersand (&). Une fois activé, il n'existe qu'une seule instruction pour l'exécuter, celle-ci comportant de très nombreux paramètres facultatifs; la syntaxe complète est :

& PR# <écrans + tabulations>
<paramètres>

Pas de panique ! En cas de trou de mémoire, tapez "& PR# ?" et vous obtiendrez à l'écran la liste des paramètres disponibles. De plus, tous les paramètres ont une valeur par défaut et peuvent être donnés dans n'importe quel ordre.

Voyons-les de plus près :

- & est l'instruction qui débranche l'Applesoft vers le programme. Pas question de l'utiliser sous le Basic Integer ou le moniteur !
- PR# est l'identificateur du vecteur Ampersand demandé. En effet le programme, lors de son chargement, ne détruit pas l'ancien vecteur qui était éventuellement en place pour une autre routine. Celui-ci est automatiquement appelé si le mot-clé PR# ne suit pas l'ampersand.

Définition des écrans

<écrans + tabulations> est une suite facultative de 1 à 3 écrans haute-résolution et éventuellement 1 à 3 tabulations horizontales qui les précé-

deront à l'impression. Les valeurs possibles des écrans sont : HGR, HGR2, HGR3 (zone mémoire non visualisable située en \$6000-\$7FFF); les tabulations s'expriment par TAB (n) où n est exprimé en nombre de caractères de l'imprimante.

Exemples :

& PR# HGR, HGR2
& PR# TAB(10), HGR2, TAB(5), HGR
& PR# HGR3
& PR# HGR, HGR3, TAB(10), HGR3

Attention : dans le dernier exemple, la tabulation indiquée s'appliquera avant l'impression du premier écran. Si une tabulation est nécessaire uniquement entre le 2ème et 3ème écran, il faut écrire : & PR# TAB(0), HGR, TAB(0), HGR(3), TAB(10), HGR3... c'est la rançon à payer pour saisir les paramètres dans n'importe quel ordre. Deuxième remarque : hormis le premier paramètre, tous les autres sont précédés par une virgule.

Si rien n'est précisé, la commande par défaut est :

& PR# TAB(0), HGR

et imprime donc le premier écran haute-résolution de l'Apple.

Un maximum de trois écrans (avec leur tabulation éventuelle) est autorisé. En saisissant un quatrième effacera le premier entré.

Définition des autres paramètres

Les autres paramètres sont :

- ROT=n où n vaut 0, 1, 2 ou 3, correspondant à une rotation de 0, 90, 180 ou 270 degrés de l'impression par rapport à l'image sur écran.
- SCALE=n où n vaut 1, 2 ou 4 (pas de 3 !) effectue un effet de zoom de même proportion.
- INVERSE imprime l'écran en inverse (impression des points éteints uniquement)
- FLASH demande une impression en double densité, donnant une meilleure définition du résultat (caractéristique de l'Epson).
- SCRN(a1, a2, b1, b2) précise la partie de l'écran à imprimer, en définissant les colonnes de départ (a1) et d'arrivée (a2) et les lignes de départ (b1) et d'arrivée (b2) d'une fenêtre. Les valeurs sont exprimées comme les HTAB et VTAB d'un écran texte, c'est à

dire dans les limites 0-40 et 0-24 respectivement.

Lorsqu'ils sont omis, ces paramètres ont comme valeur par défaut : ROT=0, SCALE=1, non FLASH, non INVERSE, SCRN(0,40,0,24)

Exemples d'utilisation :

& PR# HGR2, FLASH
& PR# INVERSE, ROT=1,
SCALE=2 (écran HGR par défaut)
& PR# HGR3, HGR, SCALE=4,
SCRN(10,30,0,24)

Dans la version actuelle du programme, tous les paramètres sont globaux et s'appliquent donc à l'ensemble des écrans demandés. Enfin, il est bien sûr de la responsabilité de l'utilisateur de charger les images vidéo en mémoire avant leur utilisation.

Messages d'erreur

SYNTAX ERROR est généré lorsque, la préfixe & PR# ayant été reconnu, la suite de la commande est incompréhensible.

ILLEGAL QUANTITY ERROR est généré lorsque les valeurs des paramètres ROT, SCALE ou SCRN dépassent les plages permises.

OVERFLOW ERROR, enfin, lorsque la longueur totale d'une ligne d'impression (nombre d'écrans x taille d'un écran + tabulations) dépasse la longueur du rouleau de l'imprimante.

Ces erreurs peuvent être interceptées par l'instruction ONERR GOTO de l'Applesoft.

Fonctionnement

Organisation interne

Le programme nécessite une configuration minimum de 48K, ainsi que le moniteur Autostart et l'Applesoft sur la carte mère ou la carte langage. Il se charge à partir de l'adresse \$8000 (32768), donc juste au-dessus de la troisième page graphique, et laisse plus de 4K octets disponibles pour d'autres routines situées avant les buffers du DOS (ou de ProDOS).

Il est organisé en quatre parties principales :

- un interpréteur de commandes qui gère l'initialisation et la lecture des paramètres
- un algorithme d'impression pour des rotations à l'horizontale (ROT=0 ou 2)
- un algorithme d'impression pour des rotations à la verticale (ROT=1 ou 3)
- une batterie de modules de service

Il est malheureusement impossible d'entrer plus en détail dans le fonctionnement des deux algorithmes en raison de leur complexité, la conception de la haute résolution sur l'Ap-

ple II ayant dû être inspirée par un casse-tête chinois particulièrement vicieux...

Gestion des interruptions et de l'environnement

Le programme peut être interrompu momentanément par la touche ESC à la fin de l'impression d'une ligne. L'interruption est définitive par la touche CTRL-C.

Le programme sauvegarde l'environnement utilisé (adresses en page zéro, vecteur de Reset, adresse de la routine de sortie de caractères) avant son exécution. L'activation de RESET rétablit les valeurs d'origine; l'utilisation répétée très rapidement de Reset si ROT= 2 ou 3 peut amener la destruction de l'image vidéo.

Paramètres d'assemblage

Le fichier source HGR DUMP.S est utilisé par Big Mac.

PR SLOT définit le slot de l'imprimante (actuellement 1).

MX80, MX82, MX100 indiquent le modèle d'Epson utilisé.

TWINLF doit être à 0 si l'imprimante ne possède pas de saut de ligne interne, à 1 sinon.

MSG0 est le message envoyé (en ordre inversé) à l'imprimante pour initialiser la taille en pouces d'un saut de ligne.

MSG1 est le message lui indiquant que les PIXLNO1 * 256 + PIXLNO2 octets suivants qu'elle recevra devront être considérés comme des représentations graphiques pour la tête d'écriture à 8 aiguilles, en sim-

ple (DUMPTYPE = "K") ou double ("L") densité.

Améliorations possibles

Il est possible de perfectionner le programme pour lui faire supporter d'autres types d'imprimantes à aiguilles (en modifiant les valeurs de MSG0 et MSG1 contenant les codes de contrôles).

On pourrait également rendre les paramètres locaux, donc autoriser l'impression, sur un même bloc horizontal, d'un écran normal suivi par un écran à taille double et tourné de 90 degrés par exemple.

Enfin, on peut le modifier pour gérer les couleurs, ou la double haute-résolution (560x192) de l'Apple //e.

A vous de jouer !

Source Big Mac

```

1          LST OFF
2
3 *****
4 * EPSON HGR SCREEN DUMP *
5 *****
6
7 *Copyright [C] 1984 A.Avrane
8
9 *M.A.J.:09/07/84
10 *Creation: 21/04/84
11
12 *Parametres d'assemblage:
13 PRSLOT = 1          slot impr
14   mante
15 TWINLF = 1          saut de li
16   gne interne
17 MX80 = 0            modele de
18   l'Epson
19 MX82 = 1
20 MX100 = 0
21
22 *Constantes d'assemblage:
23 DO MX80
24 MAXWIDTH = 80      colonnes
25   FIN
26 DO MX82
27 MAXWIDTH = 100
28   FIN
29 DO MX100
30 MAXWIDTH = 136
31   FIN
32
33 LGMSG0 = 4
34 LGMSG1 = 8
35 LGMSG2 = 4
36 ESC = 27
37 CI = 9
38 CR = %8D
39 DEL = %7F
40 PAUSE = 27+%80     escape
41 ABORT = 03+%80     ctrl-c
42 PRSLOT*2 = PRSLOT*%10
43 PRSLOT*3 = PRSLOT*%100
44
45 *Adresses d'assemblage:
46 SAUGBASL = %00
47 GBASL = %24
48 CSW = %36
49 TRAP = %48
50 CHRGET = %B1
51 CHRGOT = %B7
52 TXPTR = %B8
53 HPAG = %E6
54 SOFTEV = %3F2
55 AMPER = %3F5
56 KEYBOARD = %C000
57 STROBE = %C010
58 FLUSH = %C080+PRSLOT*2
59 CARDACK = %C0C1+PRSLOT*3
60 ERROR = %D412
61 CHKCLS = %DEB8
62 CHKCOM = %DEBE
63 SYNERR = %DEC9
64 ILQTYERR = %E199
65 GETBYTC = %E6F5
66 GETBYT = %E6F8
67 HPOSN = %F411
68 PRBL2 = %F94A
69 SETPWRC = %FB6F

```

```

67 CROUT = %FD8E
68 COUT = %FDED
69 COUT1 = %FDF0
70 OUTPORT = %FE95
71
72
73 *****
74 *MAJ vecteur &, garde ancien
75 *****
76
77 *verifie vieux=nouveau, evite boucl
78   es
79 LDA AMPER+1
80 CMP #<ENTRY
81 BNE SETAMPER
82 LDA AMPER+2
83 CMP #>ENTRY
84 BEQ SETDONE yes
85
86 *Sauve ancien vecteur pour autres a
87   ppels
88 SETAMPER LDA AMPER+1 save old
89 STA SAVAMPER
90 LDA AMPER+2
91 STA SAVAMPER+1
92
93 *Met en place nouveau vecteur
94 LDA %$4C "jmp"
95 STA AMPER
96 LDA #<ENTRY
97 STA AMPER+1 set new
98 LDA #>ENTRY vector
99 STA AMPER+2
100 SETDONE LDA #0
101 STA TRAP dos bug
102 RTS
103 AST 70
104
105 *****
106 *Stockage des variables
107 *****
108
109 *Parametres:
110 PARAM = *
111 NEGATIVE DFB 0
112 DBSTRIKE DFB 0
113 ROTATION DFB 0
114 FLIPFLAG DFB 0
115 SIZE DFB 1
116 SIZE2 DFB 7
117 TAB0 = *
118 TAB3 DFB 0
119 TAB2 DFB 0
120 TAB1 DFB 0
121 SCRNMINH DFB 0
122 SCRNMACH DFB 40
123 SCRNMINV DFB 0
124 SCRNMACHV DFB 24
125 HPAG0 = *
126 HPAG3 DFB 0
127 HPAG2 DFB 0
128 HPAG1 DFB 0
129 NBDFLT = **PARAM
130
131 *****
132 *Drapeaux et compteurs programme:
133 HORIZON DFB 0
134 VERTICAL DFB 0
135 VERTICA2 DFB 0
136 PIXLHPOS DFB 0
137 PIXLVPOS DFB 0
138 MASK1 DFB 0
139 MASK2 DFB 0

```

```

136 MATRIX DFB 0
137 SCRWIDTH DFB 0
138 NBHPAG DFB 0
139 SAVEBYTE DFB 0
140 TRANSIT2 DFB 0
141 AST 70
142
143 *****
144 *Entree &, verif syntaxe
145 *****
146 ENTRY = *
147 CMP %$BA "PR# ?
148 BEQ ITSFORUS oui
149 JMP (SAVAMPER) ancien &
150
151 *****
152 *Initialise valeurs par default
153 *****
154 ITSFORUS = *
155 LDX %NBDFLT-1
156 INITDFLT LDA DEFAULT,X
157 STA PARAM,X
158 DEX
159 BPL INITDFLT
160 LDA #*K simple den
161 STA DUMPTYPE
162
163 *****
164 *Acquisition des commandes
165 *****
166 JSR CHRGET
167 LDY #1
168 BNE READNEXT =jmp
169 READPARG = *
170 JSR CHRGET
171 BEQ NOWEWO
172 JSR CHKCOM che
173
174 ck ,
175 READNEXT LDA #>READPARG-1
176 PHA
177 LDA #<READPARG-1
178 PHA
179 JSR CHRGOT
180 BEQ HEREWEGO eol or :
181 LDX #8
182 NEXTPARG CMP CMDNAME,X
183 BEQ PARGFUND
184 DEX
185 BPL NEXTPARG
186 PLA
187 PLA
188 JMP SYNERR error
189 PARGFUND TXA
190 ASL
191 TAX
192 LDA CMDADDR,X
193 PHA
194 LDA CMDADDR,X
195 PHA
196 RTS
197
198 *****
199 HEREWEGO PLA
200 PLA
201 NOWEWO = *
202
203 *****
204 *Verifie longueur < largeur imprima
205   nte

```

```

203 *****
204
205 *Calcule largeur d'un ecran
206 CLD
207 LDA SCRNMACH
208 SEC
209 SBC SCRNMACH calcul
210 LDY ROTATION largeur
211 BEQ GETTAB0 ecran
212 LDA SCRNMACH
213 SEC
214 SBC SCRNMACH
215 GETTAB0 STA SCRWIDTH 1-40
216 LDX SIZE
217 GETTAB1 DEX
218 BEQ GETTAB2
219 CLC
220 ADC SCRWIDTH
221 BNE GETTAB1 =jmp
222 GETTAB2 STA SCRWIDTH 1-160
223
224 *MAJ No matrices pour 1 ecran
225 *(7 x pixels si rot=0, sinon 8)
226 STA MASK2
227 STA PIXLN02
228 LDX #0
229 STX MASK1
230 STX PIXLN01
231 LDX #2
232 BIT DBSTRIKE flash?
233 BPL SETPIXLO no
234 INX
235 ASL MASK2
236 ROL MASK1
237 SETPIXLO ASL PIXLN02
238 ROL PIXLN01
239 DEX
240 BPL SETPIXLO
241 DEY ;rot=1?
242 BPL CHKTAB0 oui
243 SEC
244 LDA PIXLN02
245 SBC MASK2
246 STA PIXLN02
247 LDA PIXLN01
248 SBC MASK1
249 STA PIXLN01
250
251 *verifie longueur et largeur imprim
ante
252 CHKTAB0 LDA HPAG1
253 BNE CHKTAB1 default=hgr
254 LDA #20
255 STA HPAG1
256
257 CHKTAB1 CLC
258 LDA SCRWIDTH
259 ADC TAB1
260 ADC TAB2
261 LDX HPAG2
262 BEQ CHKTAB2 no 2eme
263 ADC SCRWIDTH
264 LDX HPAG3
265 BEQ CHKTAB2 no 3eme
266 ADC SCRWIDTH
267 CHKTAB2 BCC CHKTAB4 < 255
268 CHKTAB3 LDX #45 overflow
269 JMP ERROR
270 CHKTAB4 CMP #MAXWIDTH+1
271 BCS CHKTAB3
272
273 *****
274 *Sauve page zero,csw,pile,reset
275 *****
276 LDX #0F
277 SAVE LDA SAUGBASL,X
278 STA SAVEAREA,X
279 DEX
280 BPL SAVE
281
282 LDA CSW
283 STA SAUCSW
284 LDA CSW+1
285 STA SAUCSW+1
286 LDA SOFTEV
287 STA SAURESET
288 LDA SOFTEV+1
289 STA SAURESET+1
290 TSX
291 STX SAVSTACK
292
293 *MAJ vecteur Reset
294 LDA #RESET
295 STA SOFTEV
296 LDA #RESET
297 STA SOFTEV+1
298 JSR SETPWRC
299 JSR FLIP
300
301 *****
302 *Configure Epson
303 *****

```

```

304 LDA #PRSL0T
305 JSR OUTPORT P
306
307 *Uide buffer imprimante:
308 JSR PURGE
309
310 *Initialise interface:
311 LDY #LMSG0-1
312 PRINT0 LDA MSG0,Y
313 JSR COUT
314 DEY
315 BPL PRINT0
316
317 *Initialise linefeed:
318 LDY #LMSG1-1
319 PRINT1 LDA MSG1,Y
320 JSR COUT
321 DEY
322 BPL PRINT1
323
324 *****
325 *Va au dump horizontal/vertical
326 *****
327 LDA ROTATION
328 BEQ DUMP1 rot=0
329 JSR DUMP2 rot=1
330
331 *****
332 *Sortie, restaure tout et quitte
333 *****
334 EXIT = *
335 LDX #0F
336 EXIT:1 LDA SAVEAREA,X
337 STA SAUGBASL,X
338 DEX
339 BPL EXIT:1
340
341 JSR FLIP
342 LDX SAVSTACK restaure p
ile
343 TXS
344 LDA SAUCSW
345 STA CSW
346 LDA SAUCSW+1
347 STA CSW+1
348 LDA SAURESET
349 STA SOFTEV
350 LDA SAURESET+1
351 STA SOFTEV+1
352 JMP SETPWRC -> retour
a l'appel
353
354 *****
355 *Initialise imprimante pour graphiq
ue
356 *****
357 GRAPHICS = *
358
359 *Pause ou abandon ?
360 LDA KEYBOARD
361 BPL GRAPHIC5
362 STA STROBE
363 CMP #PAUSE
364 BNE GRAPHIC4
365 GRAPHIC2 LDA KEYBOARD
366 BPL GRAPHIC2
367 STA STROBE
368 GRAPHIC4 CMP #ABORT
369 BEQ EXIT
370
371 GRAPHIC5 LDY #LMSG2-1
372 GRAPHIC6 LDA MSG2,Y
373 JSR COUT
374 DEY
375 BPL GRAPHIC6
376 RTS
377 AST 70
378
379 *****
380 DUMP1 = * rot = 0
381 *****
382
383 *Dump de;ande = 1 to 3 screens
384 *1 ecran = 24 blocs de 8 lignes
385 *1 bloc = 40 caracteres
386 *1 caractere = 8 matrice
387 *1 matrice = 8 pixels
388
389 *****
390 *Dump rotation normale: selectionne
391 *ecrans, blocs de 8 lignes
392 *****
393 LDA SCRNMACH 0-23
394 ASL
395 ASL
396 ASL
397 TAY
398 STY VERTICAL 0-184
399
400 *Ligne superieure pour chaque bloc
401 SCRNB:1 LDY VERTICAL 0-191

```

```

402 STY VERTICA2 sauve lign
e superieure
403 TYA
404 LSR
405 LSR
406 LSR ;0-23
407 CMP SCRNMACH dernier bl
oc ?
408 BCS EXIT oui
409
410 LDY #0
411 STY NBHPAG
412
413 *Prend 1 ecran, place Tab
414 SCRNB:2 LDX TAB0,Y y=nbhpag
415 BEQ SCRNB:5 pas de Tab
416 JSR PRBL2 Print
417
418 SCRNB:5 LDX HPAG0,Y y=nbhpag
419 BEQ SCRNB:7 pas 1 ecran
n
420 STX HPAG
421 DEY ; premier ecran ?
422 BMI SCRNB:6 oui
423 LDA VERTICA2
424 STA VERTICAL restaure l
igne superieure
425
426 *Saisit et sauve adresse base pour
chaque bloc
427 SCRNB:6 LDA VERTICAL 0-191
428 INC VERTICAL
429 PHA
430 LDX #0
431 LDY #0
432 JSR HPOSN -> GBASL (
adresse base pour ligne)
433
434 PLA
435 AND SIZE2 7,3,1
436 ASL
437 TAX
438 PHA
439 LDA GBASL
440 ADC SCRNMACH carry=0
441 STA SAUGBASL,X
442 INX
443 LDA GBASL+1
444 STA SAUGBASL,X
445 PLA
446 LSR ;0-7
447 CMP SIZE2 derniere l
igne du groupe ?
448 BCC SCRNB:6 non
449
450 *Dump du bloc, PRINT CR, va a la su
ite
451 JSR LINE8
452 SCRNB:7 INC NBHPAG
453 LDY NBHPAG
454 CPY #3 dernier ec
ran fini ?
455 BCC SCRNB:2 no
456 JSR CROUT
457 JMP SCRNB:1 va au bloc
suivant
458
459 *****
460 *Sort le caractere forme par 8
461 *pixels horizontaux par bloc
462 *****
463 LINE8 = *
464 JSR GRAPHICS
465
466 *Verifie fin de bloc
467 LDY SCRNMACH 0-39
468 DEY
469 STY HORIZON set htab
470 LINE8:2 INC HORIZON
471 LDY HORIZON
472 CPY SCRNMACH 1-40
473 BCS CHAR8:3 sortie
474 JSR CHAR8
475
476 *MAJ adresse du groupe de 8x8-pixel
s
477 LDX #16-2
478 LINE8:3 INC SAUGBASL,X
479 DEX
480 DEX
481 BPL LINE8:3
482 *Boucle sur l'octet HTAB suivant
483 BMI LINE8:2 =jmp
484
485 *****
486 *Selectionne la matrice du caracte
re
487 *****
488 CHAR8 = *
489 LDA #00000001 MAJ compte
ur
490 STA PIXLHPOS

```

```

491 CHAR8*2 JSR PIXL8
492 CLC
493 ROL PIXLHPOS derniere m
   atrice ?
494 BPL CHAR8*2 non
495 CHAR8*3 RTS
496
497 *****
498 *Dump 8 pixels des 8 lignes
499 *****
500 PIXL8 = *
501 LDX #0
502 STX MATRIX initialisa
   tion
503 LDA #80
504 STA PIXLVPOS
505 PIXL8*1 LDA SIZE 1,2,4
506 STA SAVEBYTE
507
508 PIXL8*2 LDA (SAVGBASL,X) saisie o
   ctet
509 AND PIXLHPOS selectionn
   e bit, ignore la couleur
510 BEQ PIXL8*3
511 LDA PIXLVPOS
512 ORA MATRIX MAJ matric
   e
513 STA MATRIX
514
515 PIXL8*3 LSR PIXLVPOS
516 BEQ DELIVER execute 8
   fois
517 DEC SAVEBYTE verifie di
   mension
518 BEQ PIXL8*4
519 BNE PIXL8*2 =jmp
520 PIXL8*4 INX
521 INX
522 BNE PIXL8*1 =jmp
523 AST 70
524
525 *****
526 *Envoi au port I/O de l'imprimante
527 *****
528 DELIVER LDA DBSTRIKE 0,80
529 ASL ;met carry a 1 si Fla
   sh
530 LDA SIZE 1,2,4
531 BCC DELIVER2
532 ASL ;2,4,8
533 DELIVER2 TAX
534 DELIVER3 LDA MATRIX
535 EOR NEGATIVE
536 DELIVER4 BIT CARDACK
537 BMI DELIVER4
538 STA FLUSH
539 DEX
540 BNE DELIVER3
541 RTS
542 AST 70
543
544 *****
545 DUMP2 = * rot = 1
546 *****
547
548 *Dump demande = 1 a 3 ecrans
549 *1 ecran = 40 blocs de colonnes
550 *1 bloc = 192 groupes de ligne
551 *1 groupe de ligne = 2 octets
552 *1 matrice = 8 pixels > 1 octet
553
554 *****
555 *Boucle sur les colonnes
556 *****
557 LDY SCRNMINH 0-39
558 DEY
559 STY HORIZON
560 COL7*1 LDY #0
561 STY TRANSIT2
562 INC HORIZON
563 COL7*2 LDA HORIZON
564 CMP SCRNMAXH derniere T
   ab ?
565 BCS COL7*9 oui:sortie
566 *MAJ masques pour Tabs courant et s
   uivants
567 AND SIZE2 modulo 8,4
   ,2
568 STA SAVEBYTE 0-7/3/1
569
570 LDA SIZE 1,2,4
571 CMP #4
572 BNE COL7*20
573 LDA #3
574 COL7*20 PHA
575 TAY
576 COL7*21 DEY
577 BEQ COL7*22
578 ASL SAVEBYTE
579 BPL COL7*21 =jmp
580
581 COL7*22 PLA ;1,2,3
582 ASL
583 ASL

```

```

584 ASL
585 ADC SAVEBYTE c=0
586 ADC TRANSIT2
587 ASL
588 TAY
589 LDA MASKTABL-16,Y saisie
590 STA MASK1 masques
591 BEQ COL7*1 Geme=ier
592 INY
593 LDA MASKTABL-16,Y
594 STA MASK2
595 LDY #0
596 STY NBHPAG
597
598 *Saisit 1 ecran et imprime Tab
599 COL7*3 LDX TAB0,Y y=nbhpag
600 BEQ COL7*4
601 JSR PRBL2
602 COL7*4 LDX HPAG0,Y y=nbhpag
603 BEQ COL7*7
604 STX HPAG
605
606 JSR LINE7
607 COL7*7 INC NBHPAG
608 LDY NBHPAG
609 CPY #3 derniere ec
   ran execute ?
610 BCC COL7*3 non
611
612 JSR CROUT
613 INC TRANSIT2
614 LDA TRANSIT2
615 CMP SIZE
616 BCC COL7*2
617 BCS COL7*1 =jmp
618 COL7*9 RTS sortie
619
620 *****
621 *Dump toutes lignes pour 1 colonne
622 *****
623 LINE7 = *
624 JSR GRAPHICS
625 LDA SCRNMAYX 1-24
626 ASL
627 ASL
628 ASL
629 STA VERTICAL 8-192
630 LINE7*2 DEC VERTICAL
631 LDA VERTICAL saisie lig
   ne
632 LDX #0
633 LDY #0
634 JSR HPOSN -> GBASL
635 JSR BYTE7 dump 1 Tab
   sur 1 ligne
636 LDA SCRNMINV 0-23
637 ASL
638 ASL
639 ASL
640 CMP VERTICAL derniere l
   igne?
641 BNE LINE7*2 non
642 RTS
643
644 *****
645 *Saisie octets pour 1 tab sur 1 col
   onne
646 *****
647 BYTE7 = *
648 LDY HORIZON 1er octet
649 LDA (GBASL),Y
650 AND MASK1
651 STA SAVEBYTE
652 INY
653 LDA (GBASL),Y 2eme octet
654 AND MASK2
655 STA MATRIX
656
657 *Matrice inversee des octets
658 TSX
659 LDA MASK1
660 MERGE7*1 ASL SAVEBYTE
661 PHP
662 ASL
663 BCC MERGE7*1
664 PLP
665 ROL MATRIX
666 CMP #0
667 BNE MERGE7*1
668 TXS
669
670 *Etend matrice si taille large
671 LDX SIZE 1,2,4
672 DEX
673 BEQ PIXL7*1
674 TXA ;1,3
675 PHA
676 EOR %00000010
677 TAY ;3,1
678 SWAP7*1 LSR MATRIX
679 PHP
680 SWAP7*2 PLP

```

```

681 PHP
682 ROR SAVEBYTE
683 DEX
684 BPL SWAP7*2
685 PLP
686 PLA
687 PHA
688 TAX
689 DEY
690 BPL SWAP7*1
691 PLA
692 LDA SAVEBYTE
693 STA MATRIX
694
695 *Permute les bits de la matrice 2 p
   our 2
696 PIXL7*1 CLC
697 LDX #8
698 PIXL7*2 ROL MATRIX
699 PHP
700 DEX
701 BNE PIXL7*2
702 LDX #8
703 PIXL7*3 PLP
704 ROL MATRIX
705 DEX
706 BNE PIXL7*3
707 JMP DELIVER sortie
708 AST 70
709
710 *****
711 *Commandes des Parametres
712 *****
713 HGR = *
714 LDY #1
715 LDA (TXTPTR),Y
716 CMP #3' hgr3 ?
717 PHP
718 LDA #20
719 PLP
720 BNE HGR*GO
721 JSR CHRGET
722 LDA #60
723 BNE HGR*GO =jmp
724 HGR2 = *
725 LDA #540
726 HGR*GO LDX HPAG2
727 STX HPAG3
728 LDX HPAG1
729 STX HPAG2
730 STA HPAG1
731 RTS
732
733 TAB = *
734 JSR GETBYTC
735 CPX #MAXWIDTH
736 BCS PARM*KO
737 TXA
738 LDX TAB2
739 STX TAB3
740 LDX TAB1
741 STX TAB2
742 STA TAB1
743 JSR CHKCLS
744 JMP DECRTXT
745
746 INVERSE = *
747 LDA #5FF
748 STA NEGATIVE %FF
749 RTS
750 FLASH = *
751 LDA #80
752 STA DBSTRIKE
753 LDA #L*
754 STA DUMPTYPE
755 RTS
756
757 ROT = *
758 JSR GETBYTC
759 CPX #4
760 BCS PARM*KO
761 CPX #2
762 BCC ROT*2
763 LDA #80
764 STA FLIPFLAG
765 DEX
766 DEX
767 ROT*2 STX ROTATION
768 PARM*X JMP DECRTXT
769 PARM*KO PLA
770 PLA
771 PLA
772 PLA
773 JMP ILQTYERR illegal qu
   antity
774
775 SCALE = *
776 JSR GETBYTC
777 CPX #5
778 BCS PARM*KO
779 CPX #3
780 BEQ PARM*KO
781 STX SIZE
782 *Determine size2 (pour rot=0)

```

```

783 LDA #15
784 STA SIZE2
785 TXA
786 SCALE*2 LSR SIZE2 7,3,1
787 LSR
788 BNE SCALE*2
789 BEQ PARM*X =jmp
790
791 SCRNI = *
792 JSR GETBYTC 0-39
793 STX SCRNMINH
794 STX SCRWIDTH
795 JSR CHKCDM
796 JSR GETBYT 1-40
797 TXA
798 BEQ PARM*KO maxh=0
799 CPX #41
800 BCS PARM*KO maxh>40
801 CPX SCRWIDTH
802 BCC PARM*KO maxh<minh
803 BEQ PARM*KO maxh=minh
804 STX SCRMAXH
805
806 JSR CHKCDM
807 JSR GETBYT 0-23
808 STX SCRNMINH
809 STX SCRWIDTH
810 JSR CHKCDM
811 JSR GETBYT 1-24
812 TXA
813 BEQ PARM*KO maxv=0
814 CPX #25
815 BCS PARM*KO maxv>24
816 CPX SCRWIDTH
817 BCC PARM*KO maxv<minv
818 BEQ PARM*KO maxv=minv
819 STX SCRMAXV
820 JSR CHKCLS )
821
822 DECRTXT = * soustrait
823 LDX TXTPTR l de
824 PHP ;txtptr
825 DEX
826 STX TXTPTR
827 PLP
828 BNE DECRTXT2
829 DEC TXTPTR+1
830 DECRTXT2 RTS
831 AST 70
832
833 *****
834 *Rappel des commandes
835 *****
836 HELP = *
837 LDY #0
838 HELP2 LDA MSGHELP,Y
839 BEQ HELP3
840 JSR COUT1
841 INY
842 BNE HELP2 =jmp
843 HELP3 PLA
844 PLA
845 JMP CHRGET
846 AST 70
847
848 *****
849 *Reset: sortie d'urgence
850 *****
851 RESET = *
852 JSR PURGE
853 TSX
854 STX SAVSTACK
855 JSR EXIT
856 JMP (SAVRESET) retour au
Reset normal
857 PURGE = *
858 *Vide le buffer de l'imprimante
859 LDA #0
860 TAX
861 TAY
862 LDA #DEL
863 PURGE*2 STA FLUSH
864 DEX
865 BNE PURGE*2
866 DEY
867 BNE PURGE*2
868 RTS
869 AST 70
870
871
872 *****
873 *Flip-flop: rotation ecran
374 *****
375 H1 = PIXLHPOS
376 H2 = PIXLUPDS
377 V1 = MASK1
378 V2 = MASK2
379 FLIP = *
380 BIT FLIPFLAG rot>1?
381 BPL FLIP*8 non:sortie
382 LDX #3-1
383 STX NBHPAG
384 FLIP*0 LDX NBHPAG
385 LDA HPAGO,X
386 BEQ FLIP*7 inutilise
387 STA HPAG
388 LDA #0
389 STA V1
390 LDA #191
391 STA V2
392
393 FLIP*05 LDA #0
394 STA H1
395 LDA #39
396 STA H2
397 LDA V1
398 LDX #0
399 LDY #0
400 JSR HPOSN
401 LDA GBASL
402 STA SAVGBASL
403 LDA GBASL+1
404 STA SAVGBASL+1
405 LDA V2
406 LDX #0
407 LDY #0
408 JSR HPOSN
409
410 FLIP*1 LDY H1
411 LDA (SAVGBASL),Y
412 JSR FLOP
413 PHA
414 LDY H2
415 LDA (GBASL),Y
416 JSR FLOP
417 LDY H1
418 STA (SAVGBASL),Y
419 PLA
420 LDY H2
421 STA (GBASL),Y
422 INC H1
423 DEC H2
424 BPL FLIP*1
425 INC V1
426 DEC V2
427 LDA V2
428 CMP #5F
429 BNE FLIP*05
430 FLIP*7 DEC NBHPAG ecran suiv
431
432 BPL FLIP*0
433 FLIP*8 RTS
434
435 FLOP = *
436 LDX #8
437 ROL
438 PHP
439 DEX
440 BNE FLOP*1
441 FLOP*2 PLP
442 ROL
443 DEX
444 BNE FLOP*2
445 ROL
446 PLP
447 ROR
448 RTS
449
450 *****
451 *Stockage des constantes
452 *****
453
454 *Parametres par default:
455 DEFAULT DFB 0,0,0,0,1,7,0,0,0,0,4
0,0,24,0,0,0
456
457 *Sauvegarde de l'environnement:
458 SAVAMPER DA $FF58
459 SAVAREA DA 0,0,0,0,0,0,0
460 SAVCSW DA $FFD0
461 SAVRESET DA $FF69
462 SAVSTACK DFB $FF
463
684 *Constantes du programme:
685 MASKTABL = * de 0 a 16
pour scale=1/2/4
686 DFB %01111111,%00000001
687 DFB %01111110,%00000011
688 DFB %01111100,%00000111
689 DFB %01110000,%00001111
690 DFB %01100000,%00011111
691 DFB %01000000,%00111111
692 DFB %01000000,%01111111
693 DFB 0,0
694 DFB %00001111,%00000000
695 DFB %01110000,%00000001
696 DFB %00011110,%00000000
697 DFB %01100000,%00000011
698 DFB %00111100,%00000000
699 DFB %01000000,%00000111
700 DFB %01111000,%00000000
701 DFB 0,0
702 DFB %00000011,%00000000
703 DFB %00001100,%00000000
704 DFB %00110000,%00000000
705 DFB %01000000,%00000001
706 DFB %00000110,%00000000
707 DFB %00011000,%00000000
708 DFB %01100000,%00000000
709 DFB 0,0
710 CMDNAME DFB $91 HGR
711 DFB $90 HGR2
712 DFB $C0 TAB(
713 DFB $9E INVERSE
714 DFB $9F FLASH
715 DFB $98 ROT=
716 DFB $99 SCALE=
717 DFB $D7 SCRNI(
718 DFB $8A ? (PRINT)
719
720 DDB HGR-1
721 DDB HGR2-1
722 DDB TAB-1
723 DDB INVERSE-1
724 DDB FLASH-1
725 DDB ROT-1
726 DDB SCALE-1
727 DDB SCRNI-1
728 DDB HELP-1
729
730 *Rappel des commandes (help):
731 MSGHELP = *
732 DFB CR,CR
733 INU *EPSON HGR SCREEN DU
734
735 MP **
736
737 DFB CR
738 ASC "[C] 1984 ALEXANDRE A
739
740 VRANE*
741 DFB CR,CR,CR
742 INU "& PR#"
743 ASC " [TAB(T)] [,HGR/HGR2
744 /HGR3]*
745
746 DFB CR
747 ASC " [,INVERSE]"
748 DFB CR
749 ASC " [,FLASH]"
750 DFB CR
751 ASC " [,ROT=0/1]"
752 DFB CR
753 ASC " [,SCALE=1/2/4]"
754 DFB CR
755 ASC " [,SCRNI(C1,C2,L1,L2
756 ])*
757 DFB CR,0
758
759 *Codes de controle des imprimantes
Epson :
760 MSG2 = *
761 DFB "N","0","8",CI
762 MSG1 = *
763 DO TWINLF
764 DFB 4
765 ELSE
766 DFB 8
767 FIN
768 DFB "A",ESC,"?",ESC,0,"R"
769 ,ESC
770 MSG2 = *
771 PIXLN01 DFB 0
772 PIXLN02 DFB 0
773 DUMPTYE DFB "K"
774 SAVEVE DFB ESC
775
776 SKP 70
777 LST ON
778 END

```

Récapitulation

*8000.868B

```

8000- AD F6 03 C9 4A D0 07 AD
8008- F7 03 C9 80 F0 1B AD F6
8010- 03 8D 74 85 AD F7 03 8D
8018- 75 85 A9 4C 8D F5 03 A9

```

```

8020- 4A 8D F6 03 A9 80 8D F7
8028- 03 A9 00 85 48 60 00 00
8030- 00 00 01 07 00 00 00 00
8038- 28 00 18 00 00 00 00 00
8040- 00 00 00 00 00 00 00 00
8048- 00 00 C9 8A F0 03 6C 74
8050- 85 A2 0F 8D 64 85 9D 2E

```

```

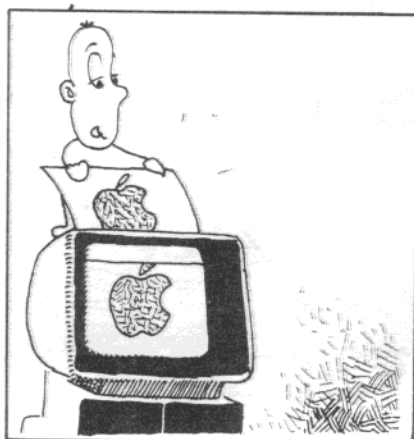
8058- 80 CA 10 F7 A9 CB 8D 8A
8060- 86 20 B1 00 A0 01 D0 08
8068- 20 B1 00 F0 2C 20 BE DE
8070- A9 80 48 A9 67 48 20 B7
8078- 00 F0 1C A2 08 DD BB 85
8080- F0 08 CA 10 F8 68 68 4C
8088- C9 DE 8A 0A AA BD C4 85

```

8090- 48 E3 B0 C4 85 48 60 68
8098- 68 D8 AD 38 80 38 ED 37
80A0- 80 AC 30 80 F0 07 AD 3A
80A8- 80 38 ED 39 80 80 46 80
80B0- AE 32 80 CA F0 06 18 6D
80B8- 46 80 D0 F7 80 46 80 8D
80C0- 44 80 8D 89 86 A2 00 8E
80C8- 43 80 8E 88 86 A2 02 2C
80D0- 2F 80 10 07 E8 0E 44 80
80D8- 2E 43 80 0E 89 86 2E 98
80E0- 86 CA 10 F7 98 10 13 38
80E8- AD 89 86 ED 44 80 8D 89
80F0- 86 AD 88 86 ED 43 80 8D
80F8- 88 86 AD 30 80 D0 05 A9
8100- 20 8D 3D 80 18 AD 46 80
8108- 6D 36 80 6D 35 80 AE 3C
8110- 80 F0 0B 6D 46 80 AE 38
8118- 80 F0 03 6D 46 80 90 05
8120- A2 45 4C 12 D4 C9 65 80
8128- F7 A2 0F B5 00 9D 76 85
8130- CA 10 F8 A5 36 8D 86 85
8138- A5 37 8D 87 85 AD F2 03
8140- 8D 88 85 AD F3 03 8D 89
8148- 85 BA 8E 8A 85 A9 BA 8D
8150- F2 03 A9 84 8D F3 03 20
8158- 6F FB 20 D7 84 A9 01 20
8160- 95 FE 20 C7 84 A0 03 89
8168- 7C 86 20 ED FD 88 10 F7
8170- A0 07 B9 80 86 2D ED FD
8178- 88 10 F7 AD 30 80 F0 51
8180- 20 C0 82 A2 0F 8D 76 85
8188- 95 00 CA 10 F8 20 D7 84
8190- AE 8A 85 9A AD 86 85 85
8198- 36 AD 87 85 85 37 AD 88
81A0- 85 80 F2 03 AD 89 85 8D
81A8- F3 03 4C 6F FB AD 00 C0
81B0- 10 13 8D 10 C0 C9 98 D0
81B8- 08 AD 00 C0 10 FB 8D 10
81C0- C0 C9 83 F0 BE AD 03 B9
81C8- 88 86 20 ED FD 88 10 F7
81D0- 6D AD 39 80 0A 0A 0A A8
81D8- 8C 3F 80 AC 3F 80 8C 40
81E0- 80 98 4A 4A 4A CD 3A 80
81E8- 80 99 A0 00 8C 47 80 BE
81F0- 34 80 F0 03 20 4A F9 BE
81F8- 38 80 F0 36 86 E6 88 30
8200- 06 AD 40 80 8D 3F 80 AD
8208- 3F 80 EE 3F 80 48 A2 00
8210- A0 00 20 11 F4 68 2D 33
8218- 80 0A AA 48 A5 26 6D 37
8220- 80 95 00 E8 A5 27 95 00
8228- 68 4A CD 33 80 90 D8 20
8230- 42 82 EE 47 80 AC 47 80
8238- C0 03 90 B3 20 8E FD 4C
8240- DB 81 20 AD 81 AC 37 80
8248- 88 8C 3E 80 EE 3E 80 AC
8250- 3E 80 CC 38 80 B0 1B 20
8258- 64 82 A2 0E F6 00 CA CA
8260- 10 FA 30 E8 A9 01 8D 41
8268- 80 20 73 82 18 2E 41 80
8270- 10 F7 60 A2 00 8E 45 80
8278- A9 80 8D 42 80 AD 32 80
8280- 8D 48 80 A1 00 2D 41 80
8288- F0 09 AD 42 80 0D 45 80

8290- 8D 45 80 4E 42 80 F0 0B
8298- CE 48 80 F0 02 D0 E4 E8
82A0- E8 D0 DA AD 2F 80 0A AD
82A8- 32 80 90 01 0A AA AD 45
82B0- 80 40 2E 80 2C C1 C1 30
82B8- FB 8D 90 C0 CA D0 EF 60
82C0- AC 37 80 88 8C 3E 80 AD
82C8- 00 8C 49 80 EE 3E 80 A0
82D0- 3E 80 CD 38 80 B0 65 2D
82D8- 33 80 8D 48 9D AD 32 80
82E0- C9 04 D0 02 A9 03 48 A8
82E8- 88 F0 05 0E 48 80 10 F8
82F0- 68 0A 0A 0A 6D 48 80 6D
82F8- 49 80 0A A8 B9 78 85 8D
8300- 43 80 F0 C3 C8 B9 78 85
8308- 8D 44 80 A0 00 8C 47 80
8310- BE 34 80 F0 03 20 4A F9
8318- BE 38 80 F0 05 86 E6 20
8320- 3D 83 EE 47 80 AC 47 80
8328- C0 03 90 E4 20 8E FD EE
8330- 49 80 AD 49 80 CD 32 80
8338- 90 95 B0 88 60 20 AD 81
8340- AD 3A 80 0A 0A 0A 8D 3F
8348- 80 CE 3F 80 AD 3F 80 A2
8350- 00 A0 00 20 11 F4 20 65
8358- 83 AD 39 80 0A 0A 0A CD
8360- 3F 80 D0 E5 60 AC 3E 80
8368- B1 26 2D 43 80 8D 48 80
8370- C8 B1 26 2D 44 80 8D 45
8378- 80 BA AD 43 80 0E 48 80
8380- 08 0A 90 F9 28 2E 45 80
8388- C9 00 D0 F1 9A AE 32 80
8390- CA F0 1F 8A 48 49 02 A8
8398- 4E 45 80 08 28 08 6E 48
83A0- 80 CA 10 F8 28 68 48 AA
83A8- 88 10 ED 68 AD 48 80 8D
83B0- 45 80 18 A2 08 2E 45 80
83B8- 08 CA D0 F9 A2 08 28 2E
83C0- 45 80 CA D0 F9 4C A3 82
83C8- A0 01 B1 88 C9 33 08 A9
83D0- 20 28 D0 09 20 B1 00 A9
83D8- 6D 00 02 A9 40 AE 3C 80
83E0- 8E 38 80 AE 30 8E 3E 3C
83E8- 80 8D 3D 80 60 20 F5 E6
83F0- E0 44 80 3F 8A AE 35 80
83F8- 8E 34 80 AE 36 80 8E 35
8400- 80 8D 36 80 20 88 DE 4C
8408- 9C 84 A9 FF 8D 2E 80 6D
8410- A9 80 8D 2F 80 A9 CC 8D
8418- 8A 86 60 20 F5 E6 E0 04
8420- B0 11 E0 02 90 07 A9 80
8428- 8D 31 80 CA CA 8E 30 80
8430- 4C 9C 84 68 68 68 4C
8438- 99 E1 20 F5 E6 E0 05 80
8440- F2 E0 03 F0 EE 8E 32 80
8448- A9 0F 8D 33 80 8A 4E 33
8450- 80 4A D0 FA F0 DA 20 F5
8458- E6 8E 37 80 8E 46 80 20
8460- BE DE 20 F8 E6 8A F0 C8
8468- E0 29 B0 C7 EC 46 80 90
8470- C2 F0 C0 8E 38 80 20 BE
8478- DE 20 F8 E6 8E 39 80 8E
8480- 46 80 20 BE DE 20 F8 E6
8488- 8A F0 A8 E0 19 80 A4 EC

8490- 46 80 90 9F F0 9D 8E 3A
8498- 80 20 88 DE A6 88 08 CA
84A0- 86 88 29 D0 02 C6 89 60
84A8- A0 00 8D 06 85 F0 06 20
84B0- F0 FD C8 D0 F5 68 4C
84B8- B1 00 20 C7 84 BA 8E 8A
84C0- 85 20 83 81 6C 88 85 A9
84C8- 00 AA A8 A9 7F 8D 90 C0
84D0- CA D0 FA 88 D0 F7 60 2C
84D8- 31 80 10 75 A2 02 8E 47
84E0- 80 AE 47 80 8D 38 80 F0
84E8- 63 85 E6 A9 00 8D 43 80
84F0- A9 BF 8D 44 80 A9 00 8D
84F8- 41 80 A9 27 8D 42 80 AD
8500- 43 80 A2 00 A0 00 20 11
8508- F4 A5 26 95 00 A5 27 85
8510- 01 AD 44 80 A2 00 A0 00
8518- 20 11 F4 AC 41 80 81 00
8520- 20 52 85 48 AC 42 80 B1
8528- 26 20 52 85 AC 41 80 91
8530- 00 68 AC 42 80 91 26 EE
8538- 41 80 CE 42 80 10 DC EE
8540- 43 80 CE 44 80 AD 44 80
8548- C9 5F D0 A9 CE 47 80 10
8550- 90 60 A2 08 2A 08 CA D0
8558- FB A2 07 28 2A CA D0 FB
8560- 2A 28 6A 60 00 00 00 00
8568- 01 07 00 00 00 00 29 00
8570- 18 00 00 00 58 FF 00 00
8578- 00 00 00 00 00 00 00 00
8580- 00 00 00 00 00 00 F0 FD
8588- 69 FF FF 7F 01 7E 03 7C
8590- 07 78 0F 70 1F 60 3F 40
8598- 7F 00 00 0F 00 70 01 1E
85A0- 00 60 03 3C 00 40 07 78
85A8- 00 00 00 03 00 0C 00 30
85B0- 00 40 01 06 00 18 00 60
85B8- 00 00 00 91 90 C0 9E 9F
85C0- 98 99 07 BA 83 C7 83 DA
85C8- 83 EC 84 09 84 0F 84 1A
85D0- 84 39 84 55 84 A7 8D 8D
85D8- 2A 20 05 10 13 0F 0E 20
85E0- 08 07 12 20 13 03 12 05
85E8- 05 0E 20 04 15 0D 10 20
85F0- 2A 8D DB C3 DD A0 B1 89
85F8- B8 B4 A0 C1 CC C5 D8 C1
8600- CE C4 D2 C5 A0 C1 D6 D2
8608- C1 CE C5 8D 8D 8D 26 20
8610- 10 12 23 A0 DB D4 C1 C2
8618- A8 D4 A9 DD A0 DB AC C8
8620- C7 D2 AF C8 C7 D2 B2 AF
8628- C8 C7 D2 B3 D0 8D A0 A0
8630- DB AC C9 CE D6 C5 D2 D3
8638- C5 DD 8D A0 A0 DB AC C6
8640- CC C1 D3 C8 DD 8D A0 A0
8648- DB AC D2 CF D4 8D B0 AF
8650- B1 DD 8D A0 A0 DB AC D3
8658- C3 C1 CC C5 8D B1 AF B2
8660- AF B4 DD 8D A0 DB AC
8668- D3 C3 D2 CE A8 C3 B1 AC
8670- C3 B2 AC CC B1 AC CC B2
8678- A9 DD 8D 00 CE 8D B8 09
8680- 04 C1 1B C0 1B 00 D2 1B
8688- 00 00 CB 1B



Editeur plein écran : EPE

Le Pacha

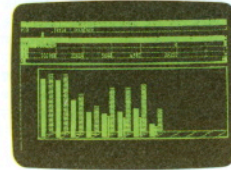
Apple II+, //e, //c

- o Listez vos programmes Basic en avant et en arrière.
- o Modifiez, insérez, effacez des caractères en plein écran sans relire les lignes.
- o Recherchez toute chaîne de caractères.
- o Choisissez vous-même les codes de contrôle d'EPE.
- o Modifiez EPE : le fichier source est sur la disquette.

150,00 F TTC franco (bon de commande page 74).

Puisque nous ne vous aviez besoin, nous

Apple et le logo Apple sont des marques déposées d'Apple Computer, Inc.

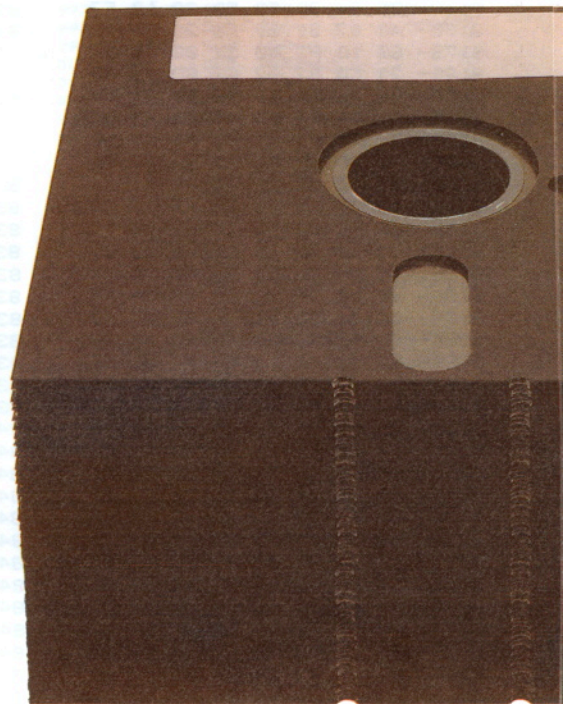


Version Calc

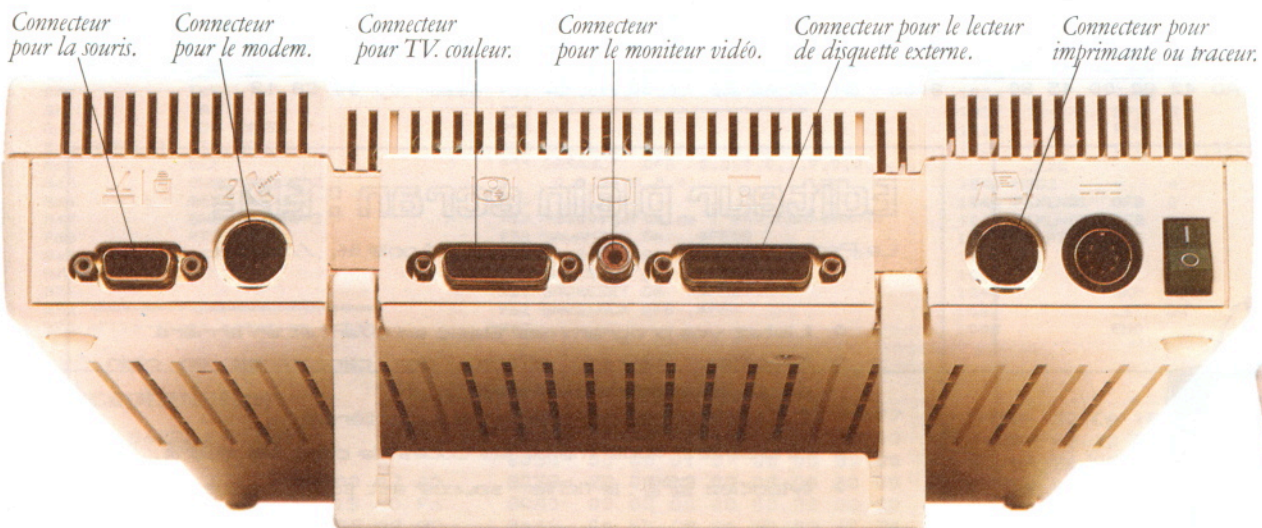


MousePaint

Une des plus grandes bibliothèques de logiciels programmes compatibles avec l'Apple IIc : jeux, gestion de base de données, analyse financière,



Un clavier 63 touches type AZERTY comprenant une accentuation complète et des caractères majuscules/minuscules intégrés.



Connecteur pour la souris.

Connecteur pour le modem.

Connecteur pour TV. couleur.

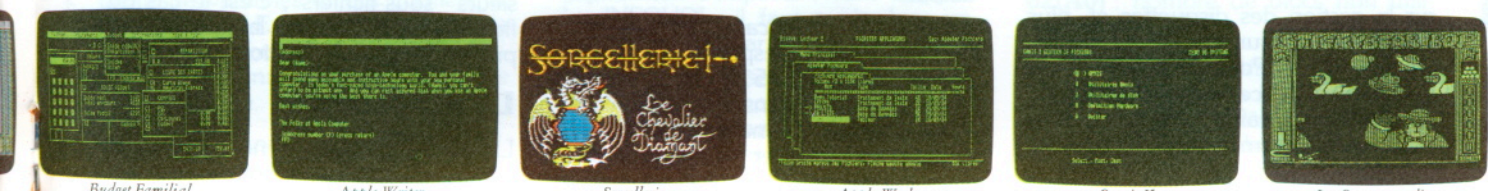
Connecteur pour le moniteur vidéo.

Connecteur pour le lecteur de disquette externe.

Connecteur pour imprimante ou traceur.

Voici comment l'écran de haute très facilement Si nous vous autant de mémoire en un seul appareil était indispensable.

... savions pas de quoi ... vous avons tout donné.



Budget Familial

Apple Writer

Sorcellerie

Apple Works

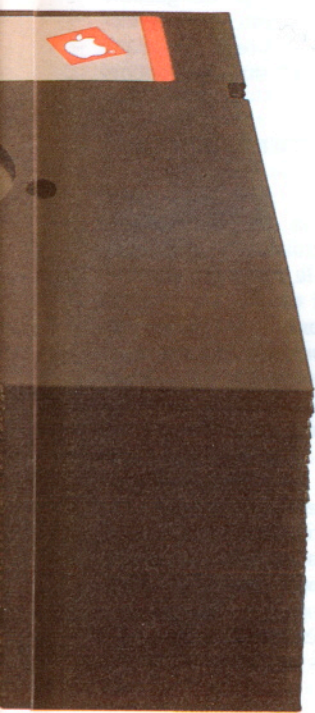
Omnis II

Les Oursins malins

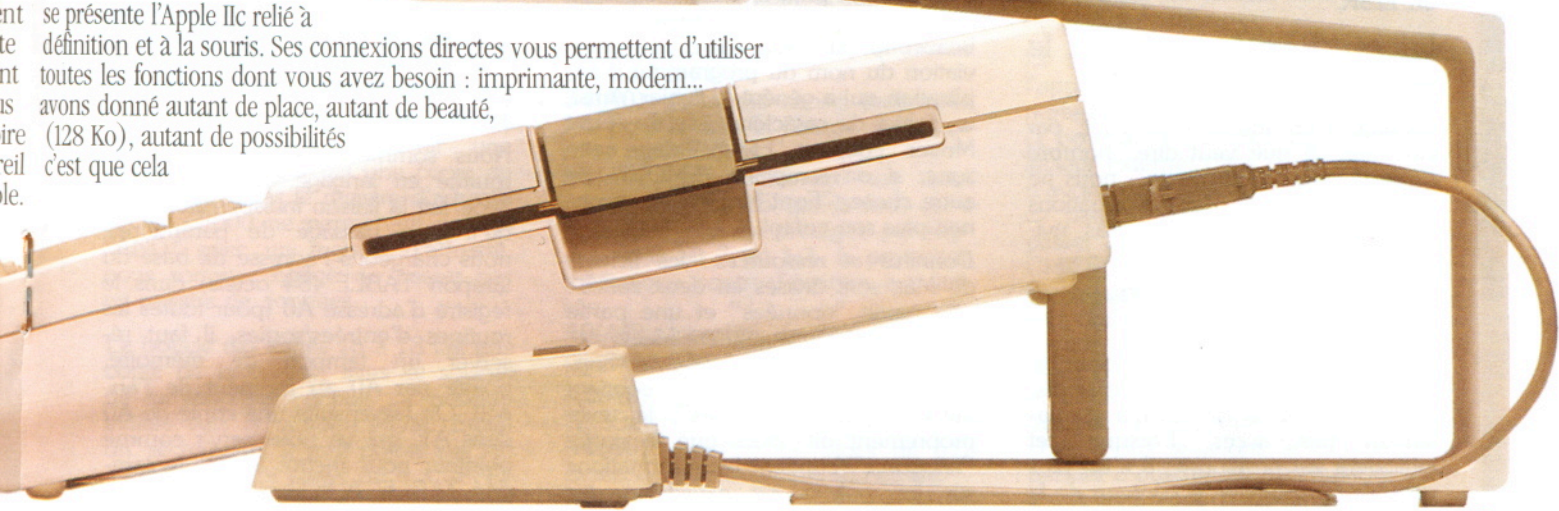
els
ux,
ère,
au monde ; 16.000
traitement de texte,
planification.

Voici 8 exemples de ce que vous pouvez afficher sur votre écran. 8 parmi 16.000 ! à vous de découvrir les autres.

Lorsque vous aurez essayé l'Apple IIc vous ne regretterez pas notre générosité.



ent se présente l'Apple IIc relié à
te définition et à la souris. Ses connexions directes vous permettent d'utiliser
nt toutes les fonctions dont vous avez besoin : imprimante, modem...
s avons donné autant de place, autant de beauté,
is (128 Ko), autant de possibilités
ire c'est que cela
reil
le.



Apple présente l'Apple IIc.



Catalogue sur Imprimante

Jean-Luc Bazanegue

Nos précédents articles destinés aux utilisateurs du Macintosh portaient essentiellement sur le graphisme. Nous nous sommes cette fois penché sur la gestion des disquettes et, plus généralement, sur les entrées/sorties du Macintosh. Pour illustrer le fonctionnement de certaines routines du système, nous avons écrit un programme qui permet d'obtenir simplement sur imprimante un catalogue complet de n'importe quelle disquette. En effet, si le "finder" autorise la visualisation du contenu d'une disquette, il ne permet pas un report aisé sur papier et, de plus, l'utilitaire en question nous cache plusieurs informations intéressantes.

Mode d'emploi

Le programme est écrit avec la version 2 du Basic Microsoft. Toutefois, les explications qui vont suivre permettront à ceux d'entre-vous qui ne possèdent pas encore cette version de l'adapter au "vieux" Basic. L'utilisation du programme est très simple et ne demande que très peu d'explications. Le menu "catalogue" permet d'obtenir les informations sur les fichiers soit sur imprimante, soit à l'écran, fichier par fichier.

Macintosh 128K

Avec ce modèle, l'impression ne peut se faire qu'en qualité "brouillon". Si une autre option est demandée, le programme ne se "plante" pas, mais passe à l'instruction suivante après quelques instants d'hésitation. Pour le reste, le fonctionnement est le même sur une version 512K ou 128K, mise à part la vitesse d'exécution, sensiblement moins élevée sur un 128K.

Informations fournies par le programme

Comme il est inutile d'indiquer, par exemple, ce que veut dire "nombre de fichiers sur la disquette", nous ne parlerons ici que des informations dont la signification n'est pas évidente.

Informations sur la disquette (volume)

Protection logicielle : il s'agit d'un indicateur dans le catalogue de la disquette, auquel l'utilisateur n'a normalement pas accès. Lorsque cet indicateur est positionné, il n'est plus possible de modifier le contenu de la disquette, tout comme avec la protection mécanique située sur le boî-

tier de la disquette.

Protection matérielle : la protection mécanique contre l'écriture citée ci-dessus.

Premier bloc du catalogue : le numéro indiqué correspond au premier des blocs logique (512 octets) contenant les informations sur les fichiers. C'est dans cette zone que notre programme va chercher les informations.

Nombre de blocs sur la disquette (sauf blocs du catalogue) : cette valeur varie en fonction de la taille des blocs.

Nombre d'octets par bloc : généralement 1024 octets.

Nombre minimum d'octets successifs : zone physique minimale pour que le système alloue un bloc à un fichier.

Premier bloc alloué : premier bloc occupé par un fichier.

Premier numéro de fichier disponible : le numéro qui sera attribué au prochain fichier créé. La valeur indiquée correspond rarement au nombre de fichiers présents sur la disquette + 1. Ceci est dû au fait que la destruction d'un fichier n'entraîne pas la mise à jour des numéros de fichiers. Ainsi, s'il y a 4 fichiers sur la disquette et que nous détruisons le quatrième, le prochain fichier créé se verra affecter 5 comme numéro, le numéro 4 n'étant plus utilisé.

Informations sur les fichiers

Type : indique le type de fichier qui peut être, par exemple, "TEXT" pour un fichier créé à partir du Basic, ou encore "APPL" pour un programme d'application.

Auteur : ceci peut être les initiales du créateur du programme, mais il s'agit beaucoup plus souvent d'une abréviation du nom du programme d'application qui a généré le fichier. Ainsi, un fichier de caractères issu de "Font Mover" contient "FMOV" dans cette zone; si on remplace "FMOV" par autre chose, "Font Mover" ne reconnaît plus son enfant.

Données et ressources : les fichiers du Mac sont divisés en deux zones : une partie "données" et une partie "ressource". Un document MacWrite, par exemple, sauvegardé avec l'option "le document entier", contient dans la partie "données", le texte proprement dit, alors que la partie "ressource" contient les informations sur les polices de caractères, les règles, etc... Certains fichiers n'utilisent que la partie "données" (un fi-

chier "TEXT" créé à partir du Basic), d'autres seulement la partie ressource (le Basic Microsoft par exemple). Lorsque qu'une des deux zones n'est pas utilisée, le programme affiche "0" pour le premier bloc.

Bundle : nous avons laissé le mot anglais car "tas" ou "paquet" ne nous semblait pas très élégant. Ce terme est affecté au fichier regroupant plusieurs "sous-fichiers", c'est le cas du fichier système et de la plupart des programmes d'application.

Le programme

La seule difficulté rencontrée pour l'écriture de ce programme était d'obtenir les informations situées sur la disquette, ce qui est impossible avec seulement le Basic. Nous avons dû faire appel à quelques routines faisant partie du système de gestion de disquettes du Macintosh. Les explications qui vont suivre porteront donc essentiellement sur l'utilisation de ces routines.

Lecture des informations sur le volume

L'instruction FILE\$, légèrement détournée de son utilisation normale, retourne dans F\$ le nom du volume (la disquette) suivi de "." et du nom du fichier sélectionné. Dans le cas présent, seul le nom du volume nous intéresse. Pour conserver uniquement ce nom, il suffit de repérer le séparateur, l'instruction LEFT\$ fait le reste. On peut noter que l'on conserve dans la nouvelle chaîne le séparateur, indispensable à la routine permettant de lire les informations sur le volume. On appelle ensuite la routine en langage machine VOLUME, dont le point d'entrée se trouve dans le 75ème élément du tableau de variables entières C, en lui passant comme argument l'adresse où se trouve, dans l'ordre, deux octets contenant la longueur de la chaîne F\$ et trois octets contenant l'adresse du premier caractère de la chaîne (voir l'article "CALL : exemple d'application", dans le numéro 16 de Pom's).

Nous sommes maintenant dans la routine en langage machine. Après avoir fait la liaison indispensable lorsqu'il y a passage de paramètres, nous chargeons l'adresse de base du tampon TABLE (84 octets) dans le registre d'adresse A0 (pour toutes les routines d'entrées/sorties, il faut réserver un tampon de mémoire, pointé par A0 au moment de l'appel). On fait ensuite une copie de A0 dans A1, qui va nous servir comme pointeur pour mettre à zéro le tampon TABLE. A la ligne 6, on charge la valeur 20, ce qui correspond au nombre de longs mots (4 octets)

moins un, dans le registre de données D0. La boucle BINIT met à zéro quatre octets à chaque tour, ceci en partant de l'adresse de base du tampon, jusqu'à ce que le contenu du registre DO soit égal à -1. Cette petite manoeuvre n'est pas indispensable lors de la lecture des informations sur le premier volume, puisque le démarrage de l'exécution du programme Basic remet à zéro le tableau de variables C qui contient le tampon TABLE, par contre, si on ne le vide pas pour les lectures suivantes, le fonctionnement de la routine serait perturbé.

Nous chargeons, à la ligne 9, l'adresse de base du tampon de mémoire "NOM", destiné à recevoir le nom du fichier, dans le registre d'adresse A2, et nous en faisons une copie dans l'adresse pointée par A0 + 18. L'adresse où se trouvent les informations sur la chaîne F\$ est, depuis le CALL, dans la pile dont le sommet est pointé par A6 (voir numéro 16). Entre notre adresse et le sommet de la pile, nous trouvons dans l'ordre l'ancienne valeur de A6 (4 octets) et l'adresse de retour au Basic (4 octets); l'adresse des informations sur la chaîne est donc à 8 octets de l'adresse pointée par A6. MOVE.L 8(A6),A1 effectue une copie de l'adresse en question dans le registre A1. Nous avons vu précédemment que les informations sur la chaîne débutent par deux octets indiquant la longueur de cette dernière, or, un nom de volume contient au maximum 28 caractères ce qui fait que seul le second octet est significatif dans le cas qui nous intéresse aujourd'hui. MOVE.B 1(A1),(A2)+ place une copie de cet octet dans le premier octet du tampon NOM pointé par A2; le registre A2 est auto-incrémenté et pointe maintenant sur le second octet du tampon NOM.

A la ligne 13, on place dans le registre D2, en allant du poids fort au poids faible, l'octet contenant le nombre de caractères et l'adresse de la chaîne sur trois octets. L'instruction ANDI.L #\$00FFFFFF,D2 force à 0 les 8 bits de poids fort de D2, ce qui nous permet d'obtenir une adresse valide. On fait une copie du registre D2 dans le registre d'adresse A3 (le 68000 ne permettant pas un "AND" avec un registre d'adresse, nous sommes obligé de passer par un registre de données).

La boucle des lignes 16 à 18 fait une copie de la chaîne, sans se soucier de sa longueur, depuis sa position initiale vers le tampon NOM. Enfin, on décrémente le mot pointé par A0 + 28 de 1, qui passe de 0 à \$FFFF soit, en complément à deux, une valeur négative.

Nous pouvons maintenant appeler la routine du Macintosh qui permet de lire les informations sur la disquette; nous l'appellerons INFOVOL. Pour appeler la routine INFOVOL, il faut introduire dans le programme le code illégal \$A007 (voir Pom's 16: "Appel des routines en ROM").

Avant d'étudier les informations retournées par la routine, une petite récapitulation s'impose. Avant d'appeler INFOVOL, il faut avoir dans un tampon pointé par A0 :

18(A0) : l'adresse (32 bits) où se trouve un octet contenant le nombre de caractères dans la chaîne, suivi des N caractères la constituant, y compris le ":",

28(A0) : une valeur négative sur deux octets.

ci est la méthode employée par notre programme pour indiquer le volume à la routine. Cependant, il existe d'autres possibilités : si A0 + 28 est positif, la routine utilise cette valeur (qui correspond à l'ordre d'insertion des disquettes dans le ou les lecteurs) pour déterminer le volume. Si A0 + 28 est négatif, la routine utilise le nom du volume, dont l'adresse est dans A0 + 18 (c'est le cas de notre programme). Si A0 + 28 est égal à 0, la routine utilise le numéro de référence de la disquette, situé en A0 + 22.

Informations retournées par INFOVOL

Les informations sont retournées dans le même tampon que celui qui nous a servi pour l'appel de la routine. Nous y trouvons :

16(A0) : un code d'erreur sur deux octets. Cette zone est à zéro si tout c'est bien passé.

18(A0) : l'adresse (4 octets) du nom du volume.

22(A0) : le numéro de référence de la disquette (2 octets).

30(A0) : la date et l'heure de création du volume, sous la forme de 4 octets contenant le nombre de secondes écoulées depuis le 1er janvier 1904 à 12H00 !

34(A0) : la même chose, mais cette fois pour la date et l'heure de la dernière modification.

38(A0) : un indicateur sur deux octets. Si le bit 15 (poids fort) est à 1, la disquette est protégée par logiciel. Si le bit 7 est à 1, la protection mécanique contre l'écriture est utilisée.

40(A0) : nombre de fichiers sur la disquette (2 octets).

42(A0) : premier bloc du catalogue (2 octets).

44(A0) : nombre de blocs de 512 octets dans le catalogue (2 octets).

46(A0) : nombre de blocs sur la disquette (ne comprend pas les blocs

du catalogue) sur 2 octets.

48(A0) : nombre d'octets par bloc (sauf pour les blocs du catalogue) sur 4 octets.

52(A0) : nombre minimum d'octets successifs pour que le système alloue un bloc à un fichier (4 octets).

56(A0) : premier bloc alloué à un fichier (2 octets).

58(A0) : premier numéro de fichier disponible (4 octets).

62(A0) : nombre de blocs non utilisés.

Lorsque nous retournons au Basic, l'exploitation de ces données ne posent pas de problème puisque le tampon TABLE de la routine en langage machine se trouve en fait au début du tableau de variables C, qui contient aussi le tampon NOM et la routine en langage machine.

Notre programme Basic doit exploiter des valeurs non signées retournées par une routine en langage machine; il faut donc prendre quelques précautions. En effet, les valeurs contenues par des variables entières sont exprimées en complément à deux et peuvent être à l'origine d'erreurs lors du calcul, par exemple, d'une donnée ou adresse sur 32 bits, si une variable entière contient un nombre supérieur à \$7FFF (32767), donc négatif.

A une expression du type :

VALEUR! = A%(N) * 65536 + A%(N+1)

on préférera :

A! = VARPTR(A%(N))
VALEUR! = PEEK(A!) * 16777216 + PEEK(A! + 1) * 65536 + PEEK(A! + 2) * 256 + PEEK(A! + 3)

Lecture des informations sur les fichiers

Grâce au travail effectué par la routine INFOVOL, l'obtention des informations sur les fichiers va être grandement simplifiée.

Le tampon TABLE contient, depuis l'appel de la routine INFOVOL, le numéro de référence de la disquette (1 pour le lecteur interne, 2 pour le lecteur externe) dans A0 + 22; nous n'avons donc pas à nous soucier de ce paramètre. Le seul paramètre à passer à la routine en assembleur FICHIER est le numéro du fichier, compris entre 1 et N, où N est la valeur retournée dans A0 + 40 par la routine INFOVOL. Après avoir déplacé N depuis la pile vers A0 + 28, et placé dans A0 + 18 l'adresse à laquelle sera reporté le nom du fichier, nous appelons la routine qui permet d'obtenir les informations sur les fichiers (que nous baptisons INFOFICH) avec le code illégal \$A00C.

Pour la routine INFOFICH, il faut dans :

18(A0) : l'adresse (4 octets) du tampon destiné au stockage du nom de fichier, précédé du nombre de caractères dans le nom.

22(A0) : le numéro (2 octets) de référence du volume sur lequel se trouve le fichier (1 pour le lecteur interne, 2 pour le lecteur externe).

28(A0) : le numéro (2 octets) du fichier concerné.

Comme pour la routine précédente, il existe plusieurs possibilités pour indiquer à la routine INFOFICH le fichier à traiter :

Si A0 + 28 est positif, la routine retourne les informations sur le fichier correspondant au nombre positif. Si A0 + 28 est égal ou inférieur à zéro, la routine retourne les informations sur le fichier dont le nom (précédé

d'un octets indiquant le nombre de caractères de ce nom) est pointé par le contenu (4 octets) de A0 + 18. Si deux fichiers situés sur des volumes différents portent le même nom, les informations retournées correspondent au fichier placé sur le volume spécifié par A0 + 22.

Informations retournées par INFOFICH

32(A0) : quatre caractères ASCII pour le type de fichier.

36(A0) : quatre caractères ASCII pour une abréviation du nom de l'auteur ou du programme qui a généré le fichier.

40(A0) : drapeaux (2 octets). Si le bit 15 est à 1, le fichier est protégé. Si le bit 14 est à 1, l'icône est invisible. Si le bit 13 est à 1, le fichier regroupe plusieurs "sous-fichiers". Si le bit 12 est à 1, le fichier est un "fi-

chier système".

42(A0) : quatre octets contenant la position de l'icône dans sa fenêtre. Les deux octets de poids fort contiennent la position horizontale, les deux octets de poids faible la position verticale.

46(A0) : deux octets pour la fenêtre où apparaît l'icône. Si A0 + 46 est égal à -3, l'icône apparaît dans la fenêtre de la corbeille. Si A0 + 46 est égal à -2, l'icône apparaît sur le bureau. Si A0 + 46 est égal à 0, l'icône apparaît dans la fenêtre du volume. Si A0 + 46 est supérieur à 0, l'icône apparaît dans la fenêtre d'un dossier.

48(A0) : numéro du fichier sur 2 octets. Il s'agit du numéro réel, qui ne correspond pas forcément à celui placé dans A0 + 28 avant l'appel de la routine.

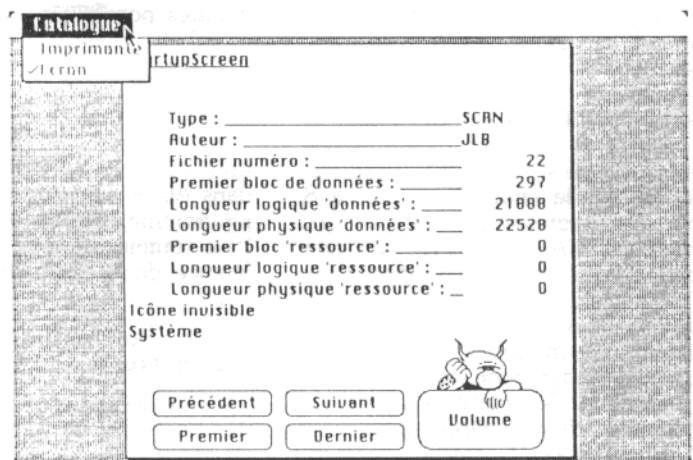
52(A0) : premier bloc pour la partie

Disquette Pom's

Protection logicielle : Non

Protection matérielle : Non

Nombre de fichiers sur la disquette : 30
 Premier bloc du catalogue : 4
 Nombre de blocs dans le catalogue : 12
 Nombre de blocs sur la disquette : 391
 Nombre d'octets par bloc : 1024
 Nombre minimum d'octets successifs : 8192
 Premier bloc alloué : 16
 Premier numéro de fichier disponible : 35
 Nombre de blocs libres : 0
 Nombre d'octets libres : 0



Num.	DONNEES			RESSOURCE			Info.bureau	Nom				
	Type Aut.	Bl.d.	Long.log	Long.phy	Bl.d.	Long.log			Long.phy			
1	FNDR	ERIK	0	0	0	2	7431	8192	DSQ	INV	Bureau	
2	ZSYS	MACS	182	5120	5120	4	153390	153600	VIS	SYS	SYSTEM	
3	PRES	MACS	0	0	0	187	17664	18432	VIS	SYS	IMAGewriter	
4	FNDR	MACS	0	0	0	205	46336	47104	VIS	SYS	FINDER	
5	APPL	QUIK	0	0	0	251	5632	6144	BUR	VIS	Copie de disque	
6	ZSYS	MACS	257	2048	2048	0	0	0	VIS		Calepin	
7	ZSYS	MACS	259	251	1024	0	0	0	VIS		Presse-papiers	
33	ZSYS	MACS	0	0	0	260	13595	14336	VIS		Album	
10	MSBB	MSBA	274	14029	14336	0	0	0	PRO	DSQ	VIS	editeur
11	TEXT		288	1720	2048	0	0	0	PRO	DSQ	VIS	XL
12	TEXT		290	1404	2048	0	0	0	PRO	DSQ	VIS	XP
13	MSBB	MSBA	292	367	1024	0	0	0	DSQ	VIS		64*216
14	MSBB	MSBA	293	395	1024	0	0	0	DSQ	VIS		256*54
15	TEXT	MSBA	294	468	1024	0	0	0	DSQ	VIS		128*108
16	MSBB	MSBA	295	133	1024	0	0	0	DSQ	VIS		4*64*54
9	TEXT	MSBA	273	221	1024	0	0	0	DSQ	VIS		Car. Estompés
18	MSBB	MSBA	296	801	1024	0	0	0	DSQ	VIS		Mac/Apple II
20	MSBB	MSBA	344	131	1024	0	0	0	DSQ	VIS		Backpat
21	MSBB	MSBA	345	573	1024	0	0	0	DSQ	VIS		Curseur
22	SCAN	JLB	297	21888	22528	0	0	0	DSQ	INV	SYS	StartupScreen
24	MSBB	MSBA	320	1309	2048	0	0	0	DSQ	VIS		Paint/Basic
25	MSBB	MSBA	322	635	1024	0	0	0	DSQ	VIS		Call
26	MSBB	MSBA	323	1375	2048	0	0	0	DSQ	VIS		Paint/Start
27	MSBB	MSBA	325	263	1024	0	0	0	DSQ	VIS		Chargement
28	MSBB	MSBA	326	449	1024	0	0	0	DSQ	VIS		Réservation
29	FFIL	FMOU	0	0	0	346	51082	51200	DSQ	VIS		Caractères
30	APPL	FMOU	0	0	0	331	12544	13312	DSQ	VIS		Font Mover
31	APPL	LCZR	0	0	0	154	18688	19456	BUR	VIS		Localizer
32	MSBB	MSBA	173	281	1024	0	0	0	DSQ	VIS		B/I
34	FNDR	ERIK	0	0	0	179	2616	3072	DSQ	INV		DeskTop

"données" du fichier (2 octets).

54(A0) : longueur logique en octets pour la partie "données" (4 octets).

58(A0) : longueur physique en octets pour la partie "données" (4 octets).

62(A0) : premier bloc pour la partie "ressource" du fichier.

64(A0) : longueur logique en octets pour la partie "ressource" (4 octets).

68(A0) : longueur physique en octets pour la partie "ressource" (4 octets).

72(A0) : date et heure de création sous la forme de 32 bits indiquant le nombre de secondes écoulées depuis le premier janvier 1904 à 12H00.

76(A0) : identique à A0 + 72, mais pour la dernière modification.

Avant de conclure, il faut dire quelques mots sur la méthode employée pour supprimer les menus. Le Basic Microsoft ne permet pas la suppression pure et simple d'un menu, il est seulement possible de les masquer. La méthode n'est pas très élégante, prend beaucoup de place et ne permet pas de supprimer le menu "Pomme", qui est fort gênant car les accessoires de bureau qu'il permet d'appeler masquent les fenêtres du Basic, qui ne sont pas remises à jour à la fermeture des accessoires. Notre programme utilise deux routines. La première, que l'on appelle avec \$A936, supprime le menu dont le numéro se trouve au sommet de la pile, sur deux octets. Le numéro

d'un menu est celui que l'on utiliserait en Basic moins 1 (Le menu "Pomme" porte le numéro 1). La seconde routine, \$A937, réaffiche la barre des menus dans sa nouvelle configuration.

Nous savons maintenant lire, depuis le Basic et avec l'aide de quelques routines, les indications situées sur une disquette. Nous verrons prochainement qu'il est aussi possible de modifier les informations, ce qui nous permettra de lire depuis le Basic le contenu de certains fichiers comme, par exemple, les polices de caractères.

Source des routines en langage machine
incorporées au programme Basic

1	TABLE	DS.L 21
2	NOM	DS.L 16
3 4E56 0000	VOLUME	LINK A6,*0
4 41FA FF66		LEA TABLE,A0
5 2248		MOVEAL A0,A1
6 7014		MOVEQ #20,D0
7 4299	BINIT	CLR.L (A1)+
8 51C8 FFFC		DBRA D0,BINIT
9 45FA FFAC		LEA NOM,A2
10 214A 0012		MOVE.L A2,18(A0)
11 226E 0008		MOVEAL 8(A6),A1
12 14E9 0001		MOVE.B 1(A1),(A2)+
13 2429 0001		MOVE.L 1(A1),D2
14 0282 00FF FFFF		ANDI.L #\$00FFFFFF,D2
15 2642		MOVEAL D2,A3
16 721B		MOVEQ #27,D1
17 14DB	BNOM	MOVE.B (A3)+,(A2)+
18 51C9 FFFC		DBRA D1,BNOM
19 5368 001C		SUBQ #1,28(A0)
20 A007		DC \$A007
21 4E5E		UNLK A6
22 4E75		RTS
23 4E56 0000	FICHIERS	LINK A6,*0
24 41FA FF26		LEA TABLE,A0
25 316E 0008 001C		MOVE 8(A6),28(A0)
26 43FA FF70		LEA NOM,A1
27 2149 0012		MOVE.L A1,18(A0)
28 A00C		DC \$A00C
29 4E5E		UNLK A6
30 4E75		RTS
31 4E56 0000	MENU	LINK A6,*0
32 3F2E 0008		MOVE 8(A6),-(SP)
33 A936		DC \$A936
34 A937		DC \$A937
35 4E5E		UNLK A6
36 4E75		RTS

REM ***CATALOGUE***

```
WINDOW CLOSE 1:WIDTH 255:ON ERROR GOTO Erreur
DEFINT A-Z:DIM C(127),P(1),T(217),C$(17):POKE &H2F8,0
DEF FNV32(A1,N)=PEEK(A1+N)*2^24+PEEK(A1+N+1)*2^16+PEEK(A1+N+2)
*256+PEEK(A1+N+3)
DSK$="Disquette ":PL$="Protection logicielle : "
PM$="Protection matérielle :":NF$="Nombre de fichiers sur la disquette : "
PBC$="Premier bloc du catalogue :":NBC$="Nombre de blocs dans le catalogue : "
NBD$="Nombre de blocs sur la disquette : "
NOB$="Nombre d'octets par bloc : "
NMO$="Nombre minimum d'octets successifs :":PBA$="Premier bloc alloué : "
PNF$="Premier numéro de fichier disponible : "
NBL$="Nombre de blocs libres :":NOL$="Nombre d'octets libres : "
O$="Oui":N$="Non"
U9$="*****":U8$="*****":U6$="*****":U5$="*****"
C$(1)="00400036000400000000000000002000000000800000300000000
C$(2)="100000018000000030000000C000000060000000A0000000C000
C$(3)="0000A000000140000000A000000140000000A000000140000000
C$(4)="A000000140000000A01FF001400000090E00F02400000008B0000
C$(5)="E4400000004600001880000000230000050000000F200000030000
C$(6)="0030B00000010000004088000000800000083E0001FC040000104
C$(7)="1000C01820000108380000002000021044003FF0100002008201
C$(8)="F5AE100002010F023DE11000010210801FC0100001002080303
C$(9)="0100000804040C00C1000004044410002100001A00442000110
C$(10)="0006300242000120000B3803840000A000130402040000C000
C$(11)="386C06040000FFE0506C07040000FC1843080F820001F0045B1
C$(12)="01FF20001C00298403FFF00038002C180DFFFC00F0001E6038FF
C$(13)="FFFFE0003FFFFFFF05FFF000000000200200000000000040
C$(14)="02000000000000800100000000000800100000000000084010
C$(15)="0000000000128810000000000014911000000000014922000
C$(16)="000000014922000000000000C922000000000004924000000
C$(17)="00000049240000000000029380000000000000480000
FOR I=1 TO 17:C$(0)=C$(0)+C$(I):NEXT

REM Initialisation des variables (pour éviter les "mouvements de mémoire")
I=0:J=0:K=0:P=0:N=0:DR=0:X=0:Y=0:V=N
VOL=74: Point d'entrée dans le tableau pour la routine "Volume"
FIL=106: Point d'entrée pour la routine "Fichiers"
MEN=120: Point d'entrée pour la routine "Menu"

AI=VARPTR(T(0)):J=1
FOR I=0 TO 435:POKE AI+I,VAL("&H"+MID$(C$(0),J,2)):J=J+2:NEXT
ERASE C$
```

```

REM DATAs pour implantation de la routine en langage machine
DATA &h4E56,0,&h41FA,&hFF66,&h2248,&h7014,&h4299
DATA &h51C8,&hFFFC,&h45FA,&hFFAC,&h214A,18,&h226E
DATA &h14E9,1,&h2429,1,&h282,&hFF,-1,&h2642,&h721B
DATA &h14DB,&h51C9,&hFFFC,&h5368,&h1C,&hA007,&h4E5E
DATA &h4E75,&h4E56,0,&h41FA,&hFF26,&h316E,8,&h1C
DATA &h43FA,&hFF70,&h2149,18,&hA00C,&h4E5E,&h4E75
DATA &h4E56,0,&h3FE2,E,8
DATA &hA936,&hA937,&h4E5E,&h4E75

REM Implantation de la routine en mémoire
FOR I=74 TO 127:READ C(I):NEXT

MENU 5,0,0,"Catalogue":MENU 5,1,1,"Imprimante":MENU 5,2,2,"Ecran"
ON MENU GOSUB Tmenu:MENUE OFF:RMENU=VARPTR(C(MEN))
FOR I=1 TO 5:RMENU I:NEXT

REM Première ou nouvelle disquette
Nouvel:
WINDOW 1,,(110,220)-(502,330),2:TEXTFONT 0
PRINT "Pour obtenir le catalogue d'une disquette,"
PRINT "sélectionnez n'importe quel fichier situé sur cette disquette"
PRINT "et cliquer sur 'Ouvrir' (ou 'Open')."
PRINT "Vous pouvez aussi effectuer un 'double-clic' sur un nom de fichier."
PRINT
PRINT "Utilisez 'Annuler' (ou 'Cancel') pour interrompre le programme.;"
F$=FILES$(1):IF F$="" THEN Fin

REM Informations sur le volume
Rvol:
WINDOW 1,,(95,40)-(417,330),4:TEXTFONT 0
FOR I=1 TO LEN(F$):IF MID$(F$,I,1)="" THEN V=1
NEXT
V$=LEFT$(F$,V):GETVOL=VARPTR(C(VOL)):GETVOL=VARPTR(V$)
GOSUB Infovol
MENU 5,0,1:MENU ON
IF DR THEN BUTTON 1,1,"Impression",5,260-(95,280),1:GOTO Bout:
BUTTON 1,1,"Fichiers",5,260-(95,280),1
Bout:
BUTTON 2,1,"Autre disquette",105,260-(215,280),1
BUTTON 3,1,"Annuler",225,260-(315,280),1
WHILE DIALOG(0)>1:WEND
MENU 5,0,0:MENUE OFF:IF DIALOG(1)=2 THEN Nouvel
IF DIALOG(1)=3 THEN Fin
REM C(20) contient le nombre de fichiers sur la disquette

```

```

N=C(20):IF DR THEN Impr
REM Affichage des informations sur les fichiers à l'écran
WINDOW 1,,(85,25)-(422,335),4:WIDTH 42
TEXTFONT 0
BUTTON 5,1,"Volume",220,260-(315,305),1
BUTTON 4,0,"Premier",20,285-(110,305),1
BUTTON 1,0,"Précédent",20,260-(110,280),1
IF N=1 THEN J=0 ELSE J=1
BUTTON 2,J,"Suivant",120,260-(210,280),1
BUTTON 3,J,"Dernier",120,285-(210,305),1
PUT(235,221),T,PSET
PRINT:PRINT:PRINT:J=1
PRINT TAB(5) "Type: ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Auteur: ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Fichier numéro: ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Premier bloc de données: ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Longueur logique 'données': ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Longueur physique 'données': ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Premier bloc 'ressource': ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Longueur logique 'ressource': ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0)):PRINT
PRINT TAB(5) "Longueur physique 'ressource': ";GETPEN VARPTR(P(0))
LINE(P(1),P(0))-(254,P(0))

REM Premier ou nouveau fichier
Nfich:
LINE(1,1)-(335,50),30,BF:LINE(255,51)-(335,190),30,BF
LINE(1,191)-(335,210),30,BF:LINE(1,211)-(200,259),30,BF
GETFILI=VARPTR(C(FIL)):GETFILI
REM Le nombre de caractères dans le nom du fichier se trouve
REM dans le poids fort de la variable C(42). Les octets suivants
REM contiennent les caractères.
AI=VARPTR(C(42)):K=PEEK(AI):TEXTFACE 4:MOVETO 2,15
FOR J=1 TO K:PRINT CHR$(PEEK(AI+J)):NEXT
REM Les informations sur le fichier sont au début du tableau C
AI=VARPTR(C(0)):TEXTFACE 0
MOVETO 255,60:P=0:GOSUB Type:MOVETO 255,76:P=4:GOSUB Type

```

```

MOVETO 255,92:PRINT USING U6$,FNV32(AI,48)
MOVETO 255,108:PRINT USING U8$,C(26)
MOVETO 255,124:PRINT USING U8$,FNV32(AI,54)
MOVETO 255,140:PRINT USING U8$,FNV32(AI,58)
MOVETO 255,156:PRINT USING U8$,C(31)
MOVETO 255,172:PRINT USING U8$,FNV32(AI,64)
MOVETO 255,188:PRINT USING U8$,FNV32(AI,68):MOVETO 2,204
IF C(20) AND &H4000 THEN PRINT "Icône invisible":GOTO Suite
PRINT "Icône visible";
IF C(23)=-3 THEN PRINT "dans la fenêtre de la corbeille":GOTO Suite
IF C(23)=-2 THEN PRINT "sur le bureau":GOTO Suite
IF C(23)=0 THEN PRINT "dans la fenêtre de la disquette":GOTO Suite
PRINT "dans la fenêtre d'un dossier"
Suite:
IF C(20) AND &H8000 THEN PRINT "Fichier protégé"
IF C(20) AND &H1000 THEN PRINT "Système"
IF C(20) AND &H2000 THEN PRINT "Bundle"
WHILE DIALOG(0)<>1:WEND
IF DIALOG(1)=5 THEN LINE(235,221)-(299,259),30,BF:WINDOW CLOSE
1:WIDTH 255:GOTO Rvol
IF DIALOG(1)=1 THEN I=I-1:BUTTON 2,1:BUTTON 3,1:IF I=1 THEN
BUTTON 1,0:BUTTON 4,0
IF DIALOG(1)=2 THEN I=I+1:BUTTON 1,1:BUTTON 4,1:IF I=N THEN
BUTTON 2,0:BUTTON 3,0
IF DIALOG(1)=4 THEN I=I-1:BUTTON 1,0:BUTTON 4,0:BUTTON 2,1:BUTTON
3,1
IF DIALOG(1)=3 THEN I=N:BUTTON 2,0:BUTTON 3,0:BUTTON 1,1:BUTTON
4,1
GOTO Nfich
REM Impression des informations sur le volume et les fichiers
Impr:
OPEN "LPT1:PROMPT" FOR OUTPUT AS 1
Serr:
WINDOW OUTPUT 1:CLS:GOSUB Infovol
WINDOW 2,,(80,290)-(432,330),2:TEXTFONT 0
MOVETO 35,22:PRINT "Enregistrement de l'impression en cours.";
WINDOW OUTPUT *1:TEXTFONT 0:GOSUB Infovol:PRINT:TEXTSIZE 9
TEXTFONT 4
REM 16 espaces entre "DONNEES" et "RESSOURCE"
PRINT TAB(25),"DONNEES" RESSOURCE"
REM E=espace, S=souligné
REM 235,1E,235
PRINT TAB(17)_____

```

```

PRINT:PRINT "Num. Type Aut. Bl.d. Long.log Long.phy Bl.d. Long.log Long.";
PRINT "phy Info.bureau Nom"
REM 55,1E,45,1E,45,1E,55,1E,85,1E,85,1E,5S,1E,5S,1E
PRINT _____;
REM 8S,1E,8S,1E,15S,1E,20S
PRINT _____;

```

```

PRINT
FOR I=1 TO N:GETFIL:=VARPTR(C(FIL)):GETFILL:=AI:=VARPTR(C(I))
PRINT USING U6$,FNV32(AI,48):PRINT TAB(7):P=0:GOSUB Type
PRINT TAB(12):P=4:GOSUB Type:PRINT USING U6$,C(26);
PRINT USING U9$,FNV32(AI,54):PRINT USING U9$,FNV32(AI,58);
PRINT USING U6$,C(31):PRINT USING U9$,FNV32(AI,64);
PRINT USING U9$,FNV32(AI,68):PRINT TAB(65);
IF C(20) AND &H8000 THEN PRINT "PRO";
PRINT TAB(69):IF C(23)=-3 THEN PRINT "COR";
IF C(23)=-2 THEN PRINT "BUR";
IF C(23)=0 THEN PRINT "DSO";
IF C(23)>0 THEN PRINT "DOS";
PRINT TAB(73);
IF C(20) AND &H4000 THEN PRINT "INV"; ELSE PRINT "VIS";
PRINT TAB(77):IF C(20) AND &H1000 THEN PRINT "SYS";
PRINT TAB(81):AI:=VARPTR(C(42)):TEXTFACE I:K=PEEK(AI)
IF K>22 THEN Plus22
FOR J=1 TO K:PRINT CHR$(PEEK(AI+J)):NEXT:GOTO Finfich
Plus22:
FOR J=1 TO 19:PRINT CHR$(PEEK(AI+J)):NEXT:PRINT "...";
Finfich:
TEXTFACE 0:PRINT
NEXT
WINDOW OUTPUT 2:CLS:MOVETO 30,22:TEXTFONT 0
PRINT "Impression en cours. Pour interrompre : " CHR$(17) " ";
CLOSE 1:WINDOW CLOSE 2:GOTO Nouveau

```

```

REM Sous-programme pour informations sur le volume
Infovol:
AI:=VARPTR(C(0)):TEXTFACE 4:PRINT DSK$ LEFT$(V$,V-1):TEXTFACE 0
PRINT PRINT PL$;:IF C(19) AND &H8000 THEN PRINT 0$ ELSE PRINT N$
PRINT PM$;:IF C(19) AND &H80 THEN PRINT 0$ ELSE PRINT N$
PRINT PRINT NF$ TAB(34) USING U6$,C(20)
PRINT PBC$ TAB(34) USING U6$,C(21)
PRINT NBC$ TAB(34) USING U6$,C(22)
PRINT NBD$ TAB(34) USING U6$,C(23)
PRINT NOB$ TAB(34) USING U6$,FNV32(AI,48)

```

```

PRINT NMO$ TAB(34) USING U6$,FNV32!(A!,52)
PRINT PBA$ TAB(34) USING U6$,C(28)
PRINT PNF$ TAB(34) USING U6$,FNV32!(A!,58)
PRINT NBL$ TAB(34) USING U6$,C(31)
PRINT NOL$ TAB(31) USING U9$,C(31)*1024
RETURN

```

REM Sous-programme pour barre des menus

```

Tmenu:
MENU
IF MENU(1)=1 THEN DR=-1:MENU 5,1,2:MENU 5,2,1:BUTTON
1,1,"Impression",(5,260)-(95,280),1
IF MENU(1)=2 THEN DR=0:MENU 5,2,2:MENU 5,1,1:BUTTON
1,1,"Fichiers",(5,260)-(95,280),1
RETURN

```

REM Sous-programme pour affichage type et auteur du fichier

```

Type:
FOR j=32+P TO 35+P:K=PEEK(A!+j):IF K>31 AND K<126 THEN PRINT
CHR$(K); ELSE PRINT " ";
NEXT
RETURN

```

```

Erreur:
OPEN "LPT1:" FOR OUTPUT AS 1:RESUME Serr

```

```

Fin:
WINDOW CLOSE 1:MENU RESET:END

```

A propos de l'article "Personnalisez vos disquettes Macintosh" (Pom's 16)

Le programme permettant de constituer un fichier "StartupScreen", mis au point avec la version 1 du Basic Microsoft, ne fonctionne pas avec la version 2. Pour remédier à cela, il convient de remplacer :

```

280 FOR J = 1 TO M: PRINT #2,
CHR$(A(J) / 256) ; CHR$(A(J)
AND &HFF) ;; NEXT: NEXT:
CLOSE: PRINT "Conversion effectuée." : GOTO 300

```

```

par :
280 AD = 0 : A! = VARPTR (A(1)) :
FOR AD = 0 TO 63: PRINT #2 ,
CHR$(PEEK (A! + AD)) ;; NEXT:
NEXT: CLOSE: PRINT "Conversion effectuée." : GOTO 300

```



54, rue de Dunkerque
75009 PARIS Tél.: 282.17.09
Métro: Gare du Nord (100 m)

SURPRIS ... LES PRIX!!!

PRIX T.T.C.!! POUR APPLE ET COMPATIBLES

Diskettes U.S 5" 1/4 SF/SD

Diskettes 5" 1/4 SF/DD

- Lecteur diskettes pour APPLE (mécan. Japonais, entr. direct)
- Carte synthétiseur de voix
- Carte mémoire/langage 16 K Ram
- Carte mémoire 128 K
- Carte drive 13/16 sect.
- Carte 80 colonnes
- Carte imprimante parallèle
- Carte imprimante + Buffer 32 K
- Carte série
- Carte super série
- Carte Z80 - CP/M
- Joystick de luxe 2 +, 2E, 2C

139 F/boîte 10 (exp. min. 5 boîtes Port 27 F)

170 F/boîte 10 (exp. min. 5 boîtes Port 27 F)

- Carte wilcard (déplombage)
- Carte communication
- Carte IEEE - 488
- Carte copieur Eprom
- Carte A/D - D/A 12 bit
- Carte horloge
- Carte musique
- Carte RGB + Prise TV Secam
- Carte 6522 Via
- Port pour une carte
- Ventilo 10 W. super silencieux

Port Urgent ajouter 5,50 F

NOUVEAU: Pince spéciale pour diskettes (100 000 trous min.) (port : 13 F) **69 F**
Ordinateur multicompatible Forth, Basic, CP/M, MS-Dos, CP/M86

Écrivez, nous vous enverrons une liste plus complète de nos articles. **Revendeurs, contactez-nous.**

Notre devise : "DYNAMIT COMPUTER : MOINS CHER QUE MOI TU MEURS !!"

BSAVE et BLOAD avec le Basic Microsoft

Marianne Sutz

Par rapport au Basic Applesoft, le Basic Microsoft (surtout la version 2) offre une gamme d'instructions extrêmement étendue. Malheureusement, il manque deux commandes du DOS : BLOAD et BSAVE. Ces dernières sont très pratiques lorsqu'il s'agit de sauvegarder ou charger rapidement des fichiers binaires qui peuvent être, par exemple, des données graphiques, des tableaux de variables ou encore des tables de références employées pour des fichiers à accès direct.

Nous nous sommes donc honteusement inspiré du DOS de l'Apple II pour créer un petit utilitaire propre à exécuter ces tâches. Un gros avantage par rapport à l'Apple II : la vitesse de transfert mémoire vers disquette ou disquette vers mémoire est d'environ 15 Ko par seconde, ce qui est nettement plus performant que sur un Apple II, même avec un DOS rapide.

Mode d'emploi

Le programme Basic (qui fonctionne avec les deux versions du Basic Microsoft) proposé avec cet article n'est qu'un exemple destiné à visualiser le passage de paramètres aux routines. Ce mode d'emploi est donc celui des routines proprement dites.

Implantation des routines dans vos programmes

Si vous pensez utiliser ces routines dans de nombreux programmes, vous avez tout intérêt à en faire un sous-programme MERGEable, si possible en utilisant la possibilité de faire des sous-programmes à variables locales que nous offre la nouvelle version du Basic Microsoft. Cette méthode héritée du Pascal permet de ne pas se soucier d'une éventuelle interaction entre le programme principal et le ou les sous-programmes.

Dans le programme exemple, les codes hexadécimaux correspondants à la routine en langage machine apparaissent sous forme de chaînes de caractères car cela prend beaucoup moins de place que les traditionnels DATAs. Nous avons placé ces codes dans onze chaînes différentes afin d'améliorer la lisibilité du programme mais, dans vos programmes, vous pouvez mettre tous les codes dans seulement deux chaînes, ce qui réduira encore la place occupée par la routine.

La routine "BSAVE"

Cette routine sauvegarde sur dis-

quette une zone de mémoire de N octets à partir de l'adresse A.

Le point d'entrée de la routine se situe dans le 85ème élément du tableau de variables entières dans lequel vous placez la routine. Ainsi, si le tableau de variables est baptisé C%, `VARPTR(C%(84))` retourne l'adresse à utiliser pour l'appel de la routine (avec `OPTION BASE 1`, il faudrait prendre 85 au lieu de 84). La syntaxe en est :

`CALL BSAVE! (AF!, D%, AT!, N!)` avec la première version du Basic Microsoft;
`BSAVE! AF!, D%, AT!, N!` avec la version actuelle.

BSAVE!, ou tout autre nom de variable en simple précision, contient l'adresse du point d'entrée dans la routine.

AF! représente une variable en simple précision qui contient l'adresse du descripteur de la chaîne contenant le nom du fichier. Dans la pratique, le plus simple est de placer l'expression `VARPTR(F$)` (où `F$` contient le nom du fichier) dans le `CALL`.

D% représente une variable entière contenant le numéro du lecteur : 1 pour interne, 2 pour externe. On peut aussi placer le numéro "en clair" dans le `CALL`.

AT! représente une variable en simple précision contenant l'adresse de base du tampon de mémoire à sauvegarder. Pour la sauvegarde d'un tableau de variables numériques, on peut placer dans le `CALL` l'expression `VARPTR(X(Y))`. Il est aussi possible de placer l'adresse "en clair" dans le `CALL`, à condition que cette adresse soit supérieure à 32767 (\$7FFF). Si l'on indique une adresse inférieure de cette façon, le `CALL` empilera seulement deux octets et, comme notre routine s'attend à trouver quatre octets dans la pile, il y a peu de chances pour que tout se passe bien.

N! représente une variable en simple précision contenant le nombre d'octets à sauvegarder. Il est aussi possible d'indiquer le nombre d'octets "en clair", avec les mêmes réserves que pour **AT!**.

Le premier élément du tableau de variables entières où se trouve la routine en langage machine est utilisé pour retourner un code correspondant à une erreur éventuelle. Si ce mot de 16 bits contient une valeur nulle au retour de la routine, cela indique que tout s'est bien passé. Dans

le cas contraire, un problème est survenu. Voici les codes possibles et leur signification :

- 33 : le catalogue des fichiers est saturé.
- 34 : la disquette est saturée.
- 35 : le volume indiqué n'existe pas.
- 36 : erreur d'entrées/sorties.
- 37 : nom de fichier incorrect.
- 44 : la protection mécanique contre l'écriture est en place.
- 46 : le volume est protégé par logiciel (voir "Catalogue sur imprimante").
- 48 : un fichier portant le nom du fichier indiqué existe déjà.
- 56 : numéro de lecteur incorrect.

La routine refuse "d'écraser" un fichier existant portant le même nom que celui indiqué, et retourne le code -48 dans le premier élément du tableau de variables. Si vous voulez outrepasser cette protection, vous pouvez écrire votre programme ainsi :

```
100 BSAVE! = VARPTR (C(84))
110 CALL BSAVE! (VARPTR(F$),
1, AT!, N!)
120 IF C(0) = -48 THEN KILL F$ :
GOTO 110
```

La routine "BLOAD"

Cette routine charge en mémoire les N octets contenus par le fichier F, à partir de l'adresse A.

Le point d'entrée de la routine se situe dans le 156ème élément du tableau de variables entières. La syntaxe de cette routine est :

`CALL BLOAD! (AF!, D%, AT!)` avec la première version du Basic Microsoft;
`BLOAD! AF!, D%, AT!` avec la version actuelle.

Les paramètres sont les mêmes que pour la routine BSAVE, mis à part le nombre d'octets, qui a ici disparu car il est déterminé automatiquement par la routine. Au retour de cette routine, le premier élément du tableau contient aussi un code d'erreur :

- 35 : le volume indiqué n'existe pas.
- 36 : erreur d'entrées/sorties.
- 37 : nom de fichier incorrect.
- 43 : fichier introuvable.
- 56 : numéro de lecteur incorrect.

Fonctionnement des routines

Les deux utilitaires emploient des zones communes : le tampon de 80

octets TABLE, indispensable aux routines d'entrées/sorties du Macintosh; le tampon de 60 octets NOM, qui recevra le nom du fichier à traiter; la sous-routine RETOUR qui, comme son nom l'indique, sera utilisée pour le retour au Basic, après avoir passé un code d'erreur éventuel dans le premier élément du tableau de variables qui reçoit les routines; enfin, la sous-routine INITTABLE, qui vide le tampon TABLE à chaque appel de l'une ou l'autre des routines.

Plutôt que d'analyser la source des routines ligne par ligne, ce qui ferait double emploi avec les explications du programme "Catalogue" publié dans ce numéro, nous vous proposons d'étudier les routines "systèmes" mises en oeuvre pour arriver à nos fins. Comme pour toutes les routines ayant un rapport avec les entrées/sorties, le registre d'adresse A0 doit pointer sur l'adresse de base d'un tampon où doivent se trouver les paramètres à passer à la routine. Chacune de ces routines retournent aussi un code d'erreur dans les deux octets de poids faible de D0.

Création d'un fichier : \$A008

Les paramètres à placer dans le tampon pointé par A0 sont :

18(A0) : l'adresse de l'octet contenant le nombre de caractères dans le nom du fichier, suivi par les caractères eux-mêmes.

22(A0) : numéro du lecteur (1 pour le lecteur interne, 2 pour le lecteur externe).

Si, au retour de la routine, A0 + 16 est différent de zéro, une erreur est survenue; son code est retourné dans les deux octets de poids faible du registre de données D0.

Mise en place d'informations : \$A00D

Nous utilisons cette routine pour placer la chaîne "TEXT" dans la zone du catalogue affectée au type de fichier. Ceci permettra d'utiliser les fichiers créés depuis d'autres programmes d'application comme, par exemple, MacWrite. Cette routine est généralement utilisée après la routine qui permet de lire les informations sur les fichiers (\$A00C, voir l'article "Catalogue" dans ce numéro). Cette routine place dans le tampon les informations déjà présentes dans le catalogue; il suffit de les modifier avant d'appeler la routine \$A00D.

16(A0) : erreur si différent de zéro (au retour).

18(A0) : adresse du nom de fichier.

22(A0) : numéro du lecteur.

32(A0) : quatre caractères ASCII pour fixer le type de fichier.

36(A0) : quatre caractères ASCII






Sur la disquette Macintosh numéro 1

vous trouverez les programmes Basic :

- **Editeur** : permet l'édition de formes (dessins) ou curseurs à l'écran, dans le but de les réutiliser depuis d'autres programmes Basic.
- **Mac/Apple II** : transmission de fichiers texte entre un Macintosh et un Apple II.
- **Paint/Basic** : Récupération de document MacPaint pour les utiliser depuis un programme Basic.
- **Paint/Start** : Récupération d'un document MacPaint pour en faire une image qui sera visualisée lors de l'insertion de vos disquettes dans le lecteur du Macintosh.
- **et d'autres programmes "exemples"** employés pour l'illustration des articles publiés dans les numéros 15 et 16 de Pom's.

Les polices de Caractères :

- **Los angeles 24** et 12 points
-  (CAIRO)
- **Mos Eisley 24** et 12 points
- **Hollywood** ()
- **Manhattan**  Return

et les programmes :

- **LOCALIZER** : configuration du clavier AZERTY pour la France (ou un autre pays).
- **Copie de disque** : pour effectuer des copies de disquettes avec un seul lecteur sur un Macintosh 128Ko, en seulement quatre passages.

 (ce qui, en gros, veut dire "150,00 F TTC franco, bon de commande page 74")

pour fixer une abréviation du nom de l'auteur ou du programme qui a généré le fichier.

40(A0) : (2 octets). Si le bit 15 est à 1, le fichier sera protégé. Si le bit 14 est à 1, l'icône sera invisible. Si le bit 13 est à 1, le fichier sera sensé regrouper plusieurs "sous-fichiers". Si le bit 12 est à 1, le fichier sera un "fichier système".

42(A0) : quatre octets contenant la position de l'icône dans sa fenêtre. Les deux octets de poids fort contiennent la position horizontale, les deux octets de poids faible la position verticale.

46(A0) : deux octets pour la fenêtre où apparaît l'icône. Si A0 + 46 est égal à -3, l'icône apparaîtra dans la fenêtre de la corbeille. Si A0 + 46 est égal à -2, l'icône apparaîtra sur le bureau. Si A0 + 46 est égal à 0, l'icône apparaîtra dans la fenêtre du volume. Si A0 + 46 est supérieur à 0, l'icône apparaîtra dans la fenêtre d'un dossier.

Cette routine ne permet pas de modifier les dates et heures, ainsi que les informations sur la position et le nombre d'octets occupés par les fichiers.

Ouverture d'un fichier : \$A000

Cette routine ne donne accès qu'à la partie "données" d'un fichier. Pour la partie "ressource", il faut utiliser une autre routine, dont le fonction-

nement est similaire, mais dont le code est \$A00A.

Paramètres passés et retournés par la routine :

16(A0) : erreur si différent de 0 (au retour).

18(A0) : adresse pointant sur le nom du fichier.

22(A0) : numéro du lecteur.

24(A0) : deux octets indiquant un numéro d'accès (au retour).

27(A0) : indicateur sur un octet. Si le contenu de l'octet est égal à 3, les lectures et écritures seront autorisées. Pour 2, seules les écritures seront admises. Pour 1, lectures seulement. Si le contenu est nul, les autorisations en cours sont conservées.

28(A0) : adresse (4 octets) d'un tampon de mémoire de 522 octets utilisé pour l'accès au fichier. Si A0 + 28 contient une valeur nulle, le tampon de mémoire réservé au volume sera employé (c'est la solution retenue pour nos routines).

Ecriture sur un fichier : \$A003

16(A0) : erreur si différent de 0 (au retour).

24(A0) : numéro d'accès au fichier (retourné précédemment par la routine d'ouverture).

32(A0) : adresse de base du tampon de mémoire à transférer vers le fichier.

36(A0) : nombre (32 bits) d'octets à transférer vers le fichier.

Lecture d'un fichier : \$A002

16(A0) : erreur si différent de 0 (au retour).

24(A0) : numéro d'accès au fichier (retourné précédemment par la routine d'ouverture).

32(A0) : adresse de base du tampon de mémoire où devront être stockés les octets lus.

36(A0) : nombre (32 bits) d'octets à lire depuis le fichier.

Fermeture d'un fichier : \$A001

16(A0) : erreur si différent de 0 (au retour).

24(A0) : numéro d'accès au fichier.

Mise à jour du catalogue de la disquette : \$A013

Il faut absolument appeler cette routine, qui ne requiert pas de paramètres, lorsque l'on vient de créer un fichier, pour mettre à jour le catalogue de la disquette. Si cela n'est pas fait, on risque de ne pas retrouver le nouveau fichier.

Nous venons de faire connaissance avec quelques routines du DOS du Macintosh. Elles offrent d'autres possibilités que nous n'avons pas citées ici; le peu de pratique que nous en avons ne nous permet pas d'en parler sans dire de grosses bêtises. Il en va de même pour d'autres routines, mais gageons que nous n'en resterons pas là...

10 Exemple d'utilisation des routines
"BSAVE" et "BLOAD"

```
20
30 DEFINT A-Z
40 DIM C(211),C$(11),D(3382),P(3),R(3)
50 C$(1)="544558544E713140FFFFE4E5E4E7522487
   013429951C8FFFC4E75
60 C$(2)="4E56000041FAFF5461EA45FAFF9E214A0
   012226E001214E90001
70 C$(3)="24290001028200FFFFFF2642723A14DB5
   1C9FFFC316E00100016
80 C$(4)="A0083228001066B0A00C3228001066A8
   43FAFFA021510020A00D
90 C$(5)="32280010669842A8001CA00032280010
   668C216E000C0020216E
100 C$(6)="000800244268002C42A8002EA003322
   800106600FF70A0013228
110 C$(7)="00106600FF66A0134EFAFF604E560000
   41FAFEC84EBAFF5C45FA
120 C$(8)="FF10214A0012226E000E14E90001242
   90001028200FFFFFF2642
130 C$(9)="723A14DB51C9FFFC316E000C001642A
   8001CA000322800106600
140 C$(10)="FF1CA00C322800106600FF12216E00
   080020216800360024
```

```
150 C$(11)="4268002C42A8002EA0023228001066
   00FEF4A0014EFAFEE
160 FOR I=1 TO 11:C$(0)=C$(0)+C$(I):NEXT
170 J=1:A1=VARPTR(C(71))
180 FOR I=0 TO 281:POKE A1+I,VAL("&H"+MID$(
   C$(0),J,2)):J=J+2:NEXT
190 ERASE C$
200 CALL TEXTFONT(0)
210 SAV=84
220 LOA=155
230 F$="XYZ"
240 R(1)=10:R(3)=330
250 FOR Y=10 TO 162 STEP 8
260 RANDOMIZE TIMER:A1=VARPTR(P(0))
270 FOR I=0 TO 7:POKE A1+I,INT(RND*256):NEXT
280 R(0)=Y:R(2)=Y+8
290 CALL FILLRECT(VARPTR(R(0)),
   VARPTR(P(0)))
300 NEXT
310 LINE(10,10)-(330,170),,B
320 GET(10,10)-(330,170),D
330 CALL MOVETO(10,200)
340 PRINT "Appuyez sur une touche pour
   sauvegarder l'image..."
350 WHILE INKEY$="" :WEND :CLS
```

```

360 NI=6766 BSAVE=VARPTR(C(SAV))
370 CALL BSAVE(VARPTR(F$),1,VARPTR(D(0)),NI)
380 IF C(0) THEN PRINT "Erreur !":
    WHILE INKEY$="" WEND END
390 ERASE D:DIM D(3382):CLS
400 CALL MOVETO(10,200)
410 PRINT "Appuyez sur une touche pour recharger
    l'image..."

```

```

420 WHILE INKEY$="" WEND:CLS
430 BLOAD!=VARPTR(C(LOA))
440 CALL BLOAD!(VARPTR(F$),1,VARPTR(D(0)))
450 IF C(0) THEN PRINT "Erreur !":
    WHILE INKEY$="" WEND:END
460 PUT(10,10),D
470 WHILE INKEY$="" WEND:CLS:KILL "XYZ":
    GOTO 250

```

Source des routines "BSAVE" et "BLOAD"

; Segment commun aux deux routines

```

1 0000          ERREUR   DC      0
2              TABLE  DS.L   20
3              NOM      DS.L   15
4 5445 5854     TYPE     DC.L   $54455854

5 3140 FFFE     RETOUR   MOVE    D0,-2(A0)
6 4E5E          UNLK     A6
7 4E75          RTS

8 2248          INITABLE MOVE.A.L A0,A1
9 7013          MOVEQ    #19,D0
10 4299         BINIT    CLR.L   (A1)+
11 51C8 FFFC     DBRA    D0,BINIT
12 4E75          RTS

```

; Routine "BSAVE"

```

13 4E56 0000     BSAVE   LINK    A6,*0
14 41FA FF56     LEA      TABLE,A0
15 61EA          BSR.S   INITABLE

16 45FA FFA0     LEA      NOM,A2
17 214A 0012     MOVE.L   A2,18(A0)
18 226E 0012     MOVE.A.L 18(A6),A1
19 14E9 0001     MOVE.B   1(A1),(A2)+
20 2429 0001     MOVE.L   1(A1),D2
21 0282 00FF FFFF ANDI.L   #$00FFFFFF,D2
22 2642          MOVE.A.L D2,A3
23 723A          MOVEQ    #58,D1
24 14DB          BNOM    MOVE.B   (A3)+,(A2)+
25 51C9 FFFC     DBRA    D1,BNOM
26 316E 0010 0016 MOVE    16(A6),22(A0)
27 A008          DC      $A008
28 3228 0010     MOVE    16(A0),D1
29 66B0          BNE.S   RETOUR
30 A00C          DC      $A00C
31 3228 0010     MOVE    16(A0),D1
32 66A8          BNE.S   RETOUR
33 43FA FFA2     LEA      TYPE,A1
34 2151 0020     MOVE.L   (A1),32(A0)
35 A00D          DC      $A00D
36 3228 0010     MOVE    16(A0),D1
37 66 98         BNE.S   RETOUR
38 42A8 001C     CLR.L   28(A0)
39 A000          DC      $A000

```

```

40 3228 0010     MOVE    16(A0),D1
41 668C          BNE.S   RETOUR
42 216E 000C 0020 MOVE.L   12(A6),32(A0)
43 216E 0008 0024 MOVE.L   8(A6),36(A0)
44 4268 002C     CLR      44(A0)
45 42A8 002E     CLR.L   46(A0)
46 A003          DC      $A003
47 3228 0010     MOVE    16(A0),D1
48 6600 FF70     BNE     RETOUR
49 A001          DC      $A001
50 3228 0010     MOVE    16(A0),D1
51 6600 FF66     BNE     RETOUR
52 A013          DC      $A013
53 4EFA FF60     BRA     RETOUR

```

; Routine "BLOAD"

```

54 4E56 0000     BLOAD   LINK    A6,*0
55 41FA FEC8     LEA      TABLE,A0
56 4EBA FF5C     BSR     INITABLE

57 45FA FF10     LEA      NOM,A2
58 214A 0012     MOVE.L   A2,18(A0)
59 226E 000E     MOVE.A.L 14(A6),A1
60 14E9 0001     MOVE.B   1(A1),(A2)+
61 2429 0001     MOVE.L   1(A1),D2
62 0282 00FF FFFF ANDI.L   #$00FFFFFF,D2
63 2642          MOVE.A.L D2,A3
64 723A          MOVEQ    #58,D1
65 14DB          BNOM2   MOVE.B   (A3)+,(A2)+
66 51C9 FFFC     DBRA    D1,BNOM2
67 316E 000C 0016 MOVE    12(A6),22(A0)
68 42A8 001C     CLR.L   28(A0)
69 A000          DC      $A000
70 3228 0010     MOVE    16(A0),D1
71 6600 FF1C     BNE     RETOUR
72 A00C          DC      $A00C
73 3228 0010     MOVE    16(A0),D1
74 6600 FF12     BNE     RETOUR
75 216E 0008 0020 MOVE.L   8(A6),32(A0)
76 2168 0036 0024 MOVE.L   54(A0),36(A0)
77 4268 002C     CLR      44(A0)
78 42A8 002E     CLR.L   46(A0)
79 A002          DC      $A002
80 3228 0010     MOVE    16(A0),D1
81 6600 FEF4     BNE     RETOUR
82 A001          DC      $A001
83 4EFA FEFE     BRA     RETOUR

```

Adhérez à

TELECHARGEMENT FRANCE PREMIERE



*qui vous propose sa collection de logiciels
français et étrangers à des prix européens.*

et recevez votre interface **APPLE - MINTEL**

Le **Téléchargement**[®] vous permet d'entrer dans le monde fabuleux de la télématique individuelle, grâce à un logiciel de télécommunication et un interface **APPLE - MINTEL** que vous recevez lors de votre adhésion.

La **Télélogithèque**[®] vous propose un catalogue de logiciels de jeux, de jeux éducatifs et de vie pratique pour votre **APPLE**.

EMB SPÉCIAL

COLLECTION HATIER

COLLECTION GOLDEN

COLLECTION HACHETTE-JEUNESSE

Conditions d'adhésion :

- Droit annuel de maintien de compte d'adhérent : **148,50 FF TTC**
- Droit unique de raccordement à **TELECHARGEMENT FRANCE PREMIERE** : **593 FF TTC**
(+ Frais d'envoi de l'interface propre à votre micro-ordinateur)

Nom _____ Prénom _____

Adresse _____

Code postal _____

Je dispose d'un **APPLE IIc/IIe/II+** avec/sans carte super-série **APPLE** (barrez les mentions inutiles).

Je désire recevoir sans engagement de ma part votre proposition d'adhésion à **TELECHARGEMENT FRANCE PREMIERE** et votre catalogue de logiciels.

Date : _____ Signature : _____

TELECHARGEMENT FRANCE PREMIERE - EUROPEAN MEDIA BUSINESS (EMB)

9, place des Terres - 75017 PARIS

Switch vidéo pour carte 80 colonnes

Eric Pascual

Ce petit montage électronique s'adresse aux possesseurs d'une carte 80 colonnes pour Apple II comportant une sortie vidéo propre.

Le problème est le suivant : pour visualiser l'affichage 80 colonnes, le moniteur doit être connecté sur la sortie de la carte, et non plus sur celle de l'Apple. Pour revenir en 40 colonnes ou pour afficher une page graphique, il faut débrancher le moniteur et le remettre sur la sortie normale. Cela devient d'autant plus gênant si on utilise un système comme Pascal, car Pascal reconnaît automatiquement les cartes 80 colonnes et les utilise.

Une première solution consiste à monter un petit interrupteur inverseur qui sert à relier le moniteur à l'une ou l'autre des sorties vidéo, sans avoir à le débrancher. Cela existe dans le commerce, tout monté pour l'Apple, mais c'est relativement cher. De plus, il faut l'actionner à la main et donc introduire dans les programmes, qui utilisent les deux sorties, des messages invitant l'utilisateur à basculer son inverseur avant de continuer. C'est très rapidement exaspérant.

La deuxième solution, objet de cet article, est d'avoir un système qui puisse être commandé soit manuellement, soit directement par programme. Rassurons tout de suite les non habitués du fer à souder, le montage est très simple. De toute manière, on peut le construire et l'utiliser sans en connaître le principe. Enfin le budget à prévoir ne dépasse pas 100 francs dans la version normale.

Principe de fonctionnement

La commutation du moniteur sur l'une ou l'autre des sorties vidéo se fait par l'intermédiaire d'un relais, piloté par un circuit électronique qui permet de le commander soit manuellement, soit en utilisant une des sorties témoins accessibles sur le connecteur de jeux de l'Apple, en l'occurrence la sortie TTL0. Un changement d'état de cette dernière fera basculer le relais dans l'une ou l'autre des deux positions.

Le coeur du montage est un circuit CD4011 qui contient 4 portes NAND en technologie CMOS. Deux de ces quatre portes sont câblées de façon à former une bascule bistable (de type RS pour les connaisseurs). Le fonctionnement de ce type de bascule est le suivant : on dispose de deux entrées et d'une sortie. Si on envoie un état 0 sur l'entrée R (RUN), la sortie de la bascule passera à l'état 1. Si on envoie un état 0 sur l'autre entrée S (STOP), la bascule reviendra à l'état 0 (j'espère ne pas avoir inversé les choses !).

Le schéma complet du montage est donné en figure 1. La bascule est au centre du montage (les deux portes NAND avec les liaisons entrecroisées). L'une de ces sorties sert à commander le relais via le transistor T1, ce dernier amplifiant le courant de sortie du CD4011, insuffisant pour commander un tel composant.

A quoi servent les autres composants ? Détaillons le montage, en commençant par l'entrée. La première des portes du circuit IC1 est montée en inverseur : ses deux entrées sont reliées, ce qui a pour effet de produire un signal de sortie qui est la négation du signal d'entrée (on pourra le vérifier en se reportant à la table de vérité de la fonction NAND). L'utilité de cette porte est d'isoler le montage du reste de l'Apple. En effet, elle présente une très grande résistance d'entrée (il s'agit d'un circuit CMOS). Ainsi les circuits internes de l'ordinateur ne seront pas perturbés par notre montage.

Le signal passe ensuite par le condensateur C1 qui permet de générer une impulsion à partir d'un changement d'état. Cette impulsion est celle qu'il faut appliquer à l'entrée de la bascule pour la faire changer d'état. La résistance R1 permet de forcer l'entrée de la bascule, à l'état 1 en l'absence d'impulsion, et donc de lui faire maintenir son état de sortie constant.

On constate qu'il y a un deuxième chemin pour le signal d'entrée, symétrique par rapport au précédent. C'est tout simplement pour attaquer la deuxième entrée de la bascule et commander le changement d'état inverse du relais. On inverse donc le signal d'entrée pour qu'il produise

l'effet inverse. Les interrupteurs (poussoirs) SW1 et SW2 sont les commandes manuelles du montage. En appuyant sur l'un ou l'autre, on envoie une impulsion appropriée à la bascule qui changera d'état en fonction de l'interrupteur qui aura été utilisé.

Dernier point : les diodes D1 et D2 sont des LEDs (diodes électroluminescentes) qui serviront de voyants témoins de l'état de la bascule. Elles sont facultatives et, si on ne les utilise pas, on peut aussi supprimer les résistances R5 et R6 (limitatrices du courant qui traverse les LEDs), ainsi que le transistor T2 et sa résistance R4 (qui servait, comme plus haut, à obtenir un courant suffisant pour allumer la LED).

Montage pratique

La figure 2 donne un exemple d'implantation des composants sur un circuit imprimé (dont le tracé apparaît en transparence). Si on ne dispose pas du matériel nécessaire à la confection d'un tel circuit, on peut plus simplement monter les composants sur une plaquette "prototype" (circuit imprimé percé de trous métallisés formant un quadrillage) et établir les liaisons avec du fil fin isolé (attention aux court-circuits). L'auteur a utilisé cette méthode et lorsque tout a été bien monté et bien vérifié, tout marche parfaitement.

La figure 3 donne le brochage du connecteur de jeux en indiquant les connexions, ainsi que le brochage du CD 4011 utilisé dans le montage. Attention à la position du repère lors de la mise en place, un circuit monté à l'envers est un circuit mort lors de la mise sous tension !

Pour les tests, il suffit d'alimenter le montage par une pile de 4,5 volts si on ne dispose pas d'alimentation stabilisée. On reliera le fil d'entrée alternativement au plus et au moins de la pile; on vérifiera ainsi que le relais bascule et que les diodes s'allument en conséquence.

Mise en place dans l'Apple

A l'issue d'un test complet, il reste à relier tout cela à l'Apple. Il est très important, pour ne pas dire indispensable, d'éteindre l'ordinateur avant de faire ou de défaire un branchement, sinon ...

Pour relier nos fils au connecteur de jeux, on peut utiliser un connecteur 16 broches pour fils en nappe, que votre marchand de composants se fera un plaisir de sertir sur les fils allant au montage (à Paris, certains magasins font cela gratuitement lorsqu'on leur achète un connecteur). Autre solution, acheter un support de circuit intégré 16 broches à wrapper. Ces supports ont des broches longues et solides qui entrent sans difficulté dans le connecteur de jeux. Il vous suffit alors de souder vos fils dans les logements du support recevant normalement les pattes du CI (solution adoptée par l'auteur). Cette méthode est tout à fait adaptée s'il y a peu de fils à monter (ce qui est le cas ici), sinon on préférera celle du connecteur sertis sur du câble en nappe.

Le circuit et ses composants externes (poussoirs, LEDs témoins) seront montés dans un petit boîtier métallique pour blinder les connexions vidéo. L'ensemble pourra être fixé sur le côté de l'Apple. Les liaisons vidéo seront obligatoirement en câble blindé, et on veillera à relier les blindages des différents fils, sous peine de voir l'écran rempli de parasites divers.

Commande par logiciel

La sortie TTL0 se commande en référant les adresses \$C058 et \$C059 (voir manuel de référence Apple II page 77). Un POKE ou un PEEK en \$C058 mettra la sortie à l'état bas et désactivera le relais. Le moniteur doit alors être relié sur la sortie vidéo 40 colonnes (sinon inverser les liaisons vidéo), de façon à se trouver dans cet état lors de la mise sous tension de l'Apple. Si on utilise l'adresse \$C059, la sortie TTL0 sera mise à l'état 1 et le relais sera alors activé. C'est tout !

Amélioration de la commande logicielle

Ce qui est dit ci-dessus fonctionne parfaitement, mais... Supposons (cas "tordu"), que l'on allume l'Apple, on est donc en 40 colonnes, TTL0 est à l'état bas. Appuyons sur le poussoir qui fait passer en 80 colonnes. Le relais bascule et le moniteur est relié sur notre carte 80 colonnes. Faisons alors un POKE 49240,0 par exemple. Le moniteur ne revient pas en 40 colonnes. Drame ! En fait c'est normal car, pour plus de simplicité, le montage ne répercute pas sur TTL0 le changement d'état manuel que nous avons fait. TTL0 étant resté à l'état 0, le POKE ne fait que

forcer à 0 une sortie qui y était déjà; donc aucune impulsion n'a été générée, ce qui explique l'absence de réaction du montage. Si vous n'avez pas bien suivi, reprenez le paragraphe lentement, c'est simple !

La parade consiste, lorsqu'on veut commander un état (passage en 40 colonnes par exemple), à envoyer en premier lieu la commande inverse, puis la commande voulue. On est ainsi assuré de la génération d'une impulsion de commande puisqu'on fait passer TTL0 dans les deux états successivement. En Basic cela donnera :

```
POKE -16295,0 ($C059)
POKE -16296,0 ($C058)
```

Pour passer en 40 colonnes et inversement, pour passer en 80 colonnes; c'est imparable !

Dernier raffinement

Si on est déjà en 40 colonnes et que l'on exécute les deux instructions ci-dessus, on observera un flash à l'écran dû au passage intermédiaire en 80 colonnes. Ce n'est pas beau et je ne sais pas si la vidéo aime beaucoup. La solution est de faire ces deux accès en assembleur. En effet la lenteur de l'interpréteur Basic (ou P-code si on est en Pascal UCSD) fait que le relais "a le temps" de basculer dans les deux positions. Si on programme la séquence en assembleur, il n'aura plus le temps de "suivre" (du fait de son inertie mécanique) et il n'y aura pas de flash à l'écran. Il y a juste une précaution à prendre : introduire une temporisation entre les deux accès, sinon l'impulsion générée est trop courte pour être enregistrée par le montage. Conclusion, cela se traduit par :

```
SW40 PHA
TXA
PHA
LDA $C059
LDX £#0A
LOOP DEX
BNE LOOP
LDA $C058
PLA
TAX
PLA
RTS
```

Pour la procédure SW80, il suffit d'inverser \$C058 et \$C059. Le nombre de boucles de temporisation a été déterminé par expérimentation : suffisamment grand pour être enregistré par le montage (en fait le mini-

mum est de 5), mais pas trop grand pour que le relais ne suive pas.

Conclusion

L'auteur a introduit ces procédures un peu partout. En Pascal, on peut en faire une UNIT et les mettre en SYSTEM.LIBRARY, ce qui facilite l'appel par les programmes (pas d'édition de liens à faire). En Basic pas de commentaire particulier. Sous CP/M il faut se souvenir du glissement des adresses entre le 6502 et le Z80, et par conséquent \$C058 et \$C059 deviennent respectivement \$E058 et \$E059. En mettant la routine SW80 en AUTORUN, on obtient une disquette CP/M qui commute automatiquement le moniteur en "bootant". La même manipulation peut se faire en Pascal en utilisant le SYSTEM.STARTUP. Sympathique non ?

Comme mentionné en début d'article, le montage revient à environ 100 francs si on utilise des composants courants. Il ne nécessite aucune mise au point si on n'a pas fait d'erreur de câblage.

Remarques diverses

A la longue, pour éviter les problèmes avec les poussoirs, il vaut mieux utiliser des poussoirs de type digitast. Ils sont équipés de contacts anti-rebond en or. Il en existe même, comble de raffinement, un type comportant une LED témoin sur la touche. Le modèle sans LED coûte environ 13 francs, et celui avec, à peu près 25 francs. Il est un plus cher, certes, mais nettement plus fiable.

Le circuit intégré utilisé est un CD 4011 CMOS. Ne pas prendre son équivalent TTL pour deux raisons.

Tout d'abord les valeurs des composants ne sont pas prévues pour ce type de circuit (impédances d'entrée et de sortie beaucoup plus faibles en TTL). D'autre part, le brochage du circuit TTL n'est pas le même que celui du circuit CMOS. Le tracé du circuit imprimé serait donc faux.

Pour ce qui est du relais, il est préférable d'utiliser un relais monté en boîtier CI. Il a l'avantage d'être miniaturisé, et de nécessiter des courants de commande très faibles (ne pas oublier que l'on ne doit pas consommer plus de 200mA sur la sortie 5 volts du connecteur de jeux).

La diode D3 sera une diode de redressement quelconque. Son rôle est d'absorber les courants d'auto-induction qui prennent naissance dans la bobine du relais et qui pourraient endommager le transistor T1.

Figure 1

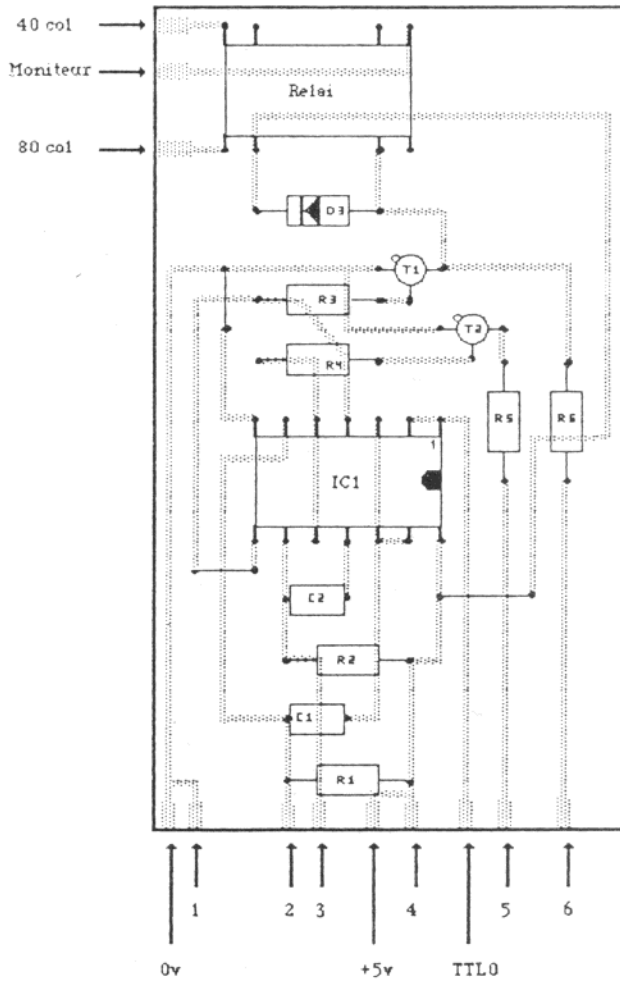
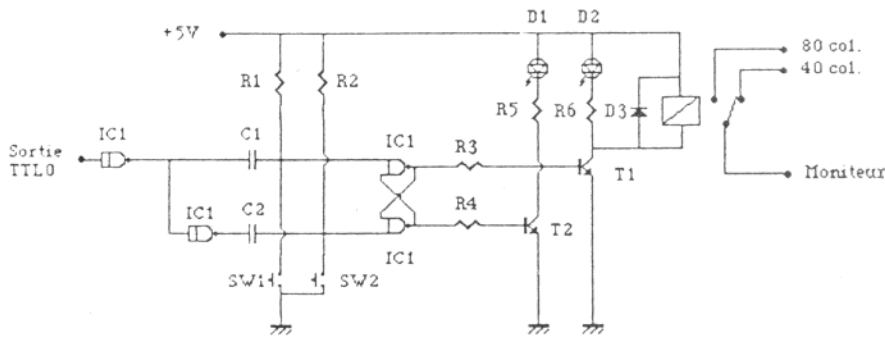


Figure 2

Nomenclature:

Résistances:

- R1, R2: 100 kohm
- R3, R4: 10 kohm
- R5, R6: 470 ohm

Condensateurs:

- C1, C2: 47 nF

Transistors:

- T1, T2: 2N2222 ou équivalent

Diodes:

- D1, D2: LED
- D3: diode redressement

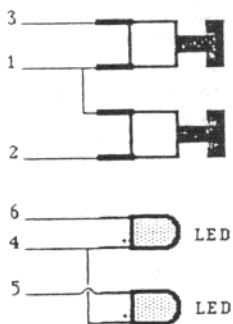
IC:

- IC1: CD4011 (4 portes NAND CMOS 2 entrées)

Divers:

- SW1, SW2: boutons poussoirs
- Relai 5 volts (brochage CI)
- Câble coaxial vidéo, prises CINCH,...
- Connecteur 16 broches CI (pour la sortie GAME de l'APPLE)

Câblage des poussoirs et des LEDs



Trucs et Astuces

Comment éviter le clignotement du curseur sur un Apple //e

Le clignotement est dû au fait que l'on change les données au moment même où on les affiche à l'écran. Il faut lire le signal en \$C019 (49177), et changer l'affichage lorsqu'il est au niveau bas (c'est-à-dire, inférieur à 128).
Essayer le programme ci-dessous avec :
30 GOTO 10

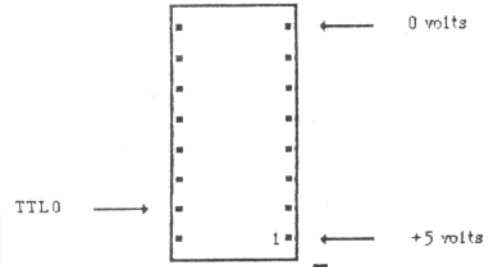
30 GOTO 20

Bien entendu, ceci est particulièrement utile pour le HGR (notamment pour les affichages de shapes).

```
10 IF PEEK (49177) > 127 THEN
10
20 VTAB 10 : HTAB 10 : PRINT "
"
VTAB 10 : HTAB 10 : PRINT
"POM'S"
30 GOTO 10
```

Source : Manuel de référence de l'Apple //e.

Switch vidéo: brochages divers



CLAVIER

Connecteur de jeux APPLE II

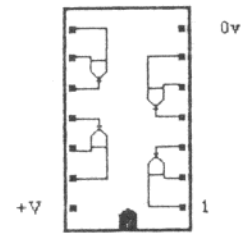


Figure 3

CD4011

Mini - éditeur Basic pour Apple II ou //e

Jean-François Rabasse

Je vous propose un petit programme en langage machine, connecté au niveau de la routine d'accès au clavier (RDCHAR), qui reçoit ses instructions de l'utilisateur par l'intermédiaire du clavier (le vrai) et simule des touches "soft" pour le système. Il n'assure aucune gestion, ni de l'écran, ni du buffer d'entrée (\$200), et fonctionne donc sous toutes les configurations qui utilisent la routine d'entrée KEYIN : moniteur, Applesoft en immédiat, Applesoft en mode programme avec les fonctions INPUT et GET, DOS...

Il donne accès aux fonctions suivantes :

1-Redéfinition sur le clavier QWERTY d'un bloc numérique, suivant la disposition "calculatrice" adoptée sur plusieurs machines (HP 75, Epson HX 20...).

Cette fonction est accessible par programme, en mode immédiat et en Basic (très commode pour un programme de saisie de données qui alterne des saisies alphabétiques et numériques).

2-Programmation des touches pour générer automatiquement une suite de caractères. Le programme possède sa propre table de données et peut donc accepter n'importe quoi (on n'est plus limité aux seuls mots Basic lus dans la ROM Applesoft). On pourra, bien sûr, préprogrammer des mots Applesoft, des commandes DOS courantes (SAVE, LOAD, CATALOG...), mais aussi toute séquence utilisée fréquemment dans un programme (FOR I = 1 TO K, PRINT CHR\$(4) par exemple). La programmation se fait touche par touche, en mode immédiat et on peut la redéfinir à tout moment. On peut préprogrammer un clavier et le stocker sur disque en l'état, et conserver ainsi des claviers différents, que l'on chargera en fonction du type de programme exécuté (clavier Basic courant, clavier spécial commandes graphiques, etc...).

3-Numérotation automatique des lignes en Applesoft. Mise en (ou hors) service en permanence.

4-Le clavier possède des commandes de tabulation (uniquement sur l'Apple //e) qui utilisent les touches DEL et TAB.

Le programme est implanté à partir de l'adresse \$8600, et non plus haut, pour permettre la cohabitation avec un autre utilitaire de programmation, le logiciel RENUMBER - HOLD -

MERGE d'Apple, qui se charge à partir de l'adresse \$8D00. Il modifie HIMEM lors de sa mise en service pour assurer sa protection. Attention, RENUMBER fait de même, il est donc recommandé de charger d'abord RENUMBER, puis le clavier, si l'on désire utiliser les deux.

Utilisation

Mise en service

Si le programme n'est pas en mémoire, il suffit de faire : BRUN CLAVIER2E (ou CLAVIER2+, ou tout autre nom).

Si le programme est déjà chargé, mais a été désactivé, on fera :

- Sous Basic : CALL 34304.

- Sous moniteur : 8600G.

Le système rend la main avec un curseur modifié.

Mise hors service

- Méthode brutale : RESET.

- Méthode délicate : passez sous le moniteur et tapez :

AA55: 1B FD suivi de RETURN.

Sauvegarde

Le programme CLAVIER2E ou CLAVIER2+ ne comporte pas de table de touches prédéfinies. On l'utilisera pour créer une collection de claviers préprogrammés, sauvegardés par :

BSAVE Nom,A\$8600,L\$651.

Commandes

Le fonctionnement est un peu différent selon l'Apple dont vous disposez.

L'Apple //e utilise la touche "Pomme Pleine" (dite PP) pour les commandes, et la touche "Pomme Ouverte" (dite PO) pour les sorties de chaînes programmées (toute ressemblance avec les fonctions glossaires d'Applewriter //e a été très difficile à obtenir !). Les touches "Pommes" sont utilisées, comme les touches SHIFT et CTRL, en maintenant la "Pomme" enfoncée et en appuyant sur une autre touche.

L'Apple II utilise la touche CTRL pour les deux (le pauvre !).

Bloc numérique

La redéfinition est la suivante :

"7", "8", "9", "U", "I", "O", "J", "K", "L", "M", " ", " " correspondent respectivement à 7, 8, 9, 4, 5, 6, 1, 2, 3, 0, E, ". Pour une utilisation facile, il est recommandé de marquer les touches : caractères LE-

TRASET plus vernis mat par exemple, ça marche très bien, venez voir le mien.

- Commutation alpha / numérique
Sur Apple //e : PP-N.
Sur Apple II : CTRL-Z.
- Commutation numérique / alpha
Sur Apple //e : PP-A.
Sur Apple II : CTRL-Z.
- Dans un programme Basic
- Numérique : POKE 255,255
- Alpha : POKE 255,0

Touches programmées

1- Programmation d'une touche :

- Passer en mode programme :

Apple //e : PP-P.

Apple II : CTRL-B.

- Taper les caractères désirés.

- Affecter la séquence à une touche choisie par :

Apple //e : PO-touche.

Apple II : CTRL-touche.

L'affichage écran indique alors la touche d'affectation, annule la ligne, et on sort du mode programme.

ATTENTION :

- En mode programme, les possibilités d'édition sont limitées. On peut corriger la frappe par les flèches gauche et droite, mais pas en édition d'écran. ESC est inhibée, ainsi que CTRL-X.

- Si une touche n'est pas programmable, toute affectation sera ignorée.

- Une touche peut être reprogrammée à volonté.

- Pour déprogrammer, surtout sur Apple II, si l'on souhaite récupérer un caractère de contrôle, il suffit de programmer une chaîne vide :
Apple II : CTRL-B puis CTRL-touche.

Apple //e : PP-P puis PO-touche.

2- Sortie de la programmation d'une touche :

Apple //e : PO-touche.

Apple II : CTRL-touche.

Remarques : une touche non programmée renvoie le caractère qui lui correspond.

La programmation est prioritaire sur la redéfinition numérique du clavier. Une touche K non programmée renverra un K et non un 2, si elle est utilisée avec PO ou CTRL.

3- Capacités de programmation :

Sur **Apple //e**, toutes les "lettres majuscules" acceptent jusqu'à 7 caractères (suffisant pour tous les mots de l'Applesoft ou du DOS) et les touches "chiffres" jusqu'à 23 caractères.

Sur **Apple II**, seules les touches alphabétiques sont programmables,

sauf les caractères de contrôle suivants.

Les fonctions d'édition :

Break : CTRL-C

Cancel : CTRL-X

Return : CTRL-M

Bsp : CTRL-H

Nak : CTRL-U

Les commandes clavier :

CTRL-B

CTRL-Z

On dispose ainsi, sur l'Apple II, de 19 touches acceptant 15 caractères

4- Recommandations

Dans un programme Basic, le clavier fonctionne avec INPUT et GET. Avec GET, il faut éviter de frapper une touche programmée, sinon le GET va récupérer le premier caractère de la séquence et tous les GETs qui lui succèdent prendront les caractères suivants, un par un, sans possibilité d'intervention de l'utilisateur. Résultats folkloriques assurés ! La solution est la suivante : inhiber, pour un accès clavier, la fonction programme par :

POKE 254,0 : GET ...

Générateur de numéros de lignes (mode AUTONOM)

1- Mise en service :

- Passer en mode programme (PP-P ou CTRL-B).
- Taper : le numéro de la ligne et le pas d'incréméntation.
- Return

Le générateur possède des valeurs

par défaut qui sont 10,10. On peut donc utiliser toutes les combinaisons possibles :

Pour 10,10 : Mode Programme "Return".

Pour 100,10 : Mode Programme et "100" "Return".

Pour 10,5 : Mode Programme et "5" "Return".

etc...

Remarques : le générateur sait compter jusqu'à 65535, mais l'Applesoft ne va pas au delà de 63999 (donc faites des programmes courts !). Le générateur sort un nouveau numéro après chaque entrée de ligne et ressort le même numéro après une annulation par CTRL-X.

2- Sortie

La sortie du mode AUTONOM se fait manuellement, en appuyant sur RETURN juste après le numéro de ligne. On sort automatiquement en cas de "collision" lorsque le générateur, en cours de numérotation, rencontre une ligne de programme déjà existante. C'est une sécurité qui permet de réinsérer des lignes, en numérotation automatique, sans risquer de détruire ce qui existe. On ne pourra donc pas utiliser ce mode pour réécrire une série de lignes (prenez vos responsabilités et faites DEL vous même).

Tabulations sur lignes (réservé aux Apple //e).

1- Touche TAB :

Même fonctionnement que la touche

flèche à droite mais déplace le curseur de 10 en 10. Il est ainsi aisé de recopier rapidement une longue ligne Basic (attention aux répétitions, cela va très vite). En maintenant enfoncées les touches PO et TAB le curseur se déplace, vers la gauche, de 10 en 10.

2- Touche DEL :

C'est une touche d'annulation : elle fait reculer le curseur comme la flèche à gauche, qui supprime le dernier caractère frappé, alors que DEL supprime les caractères correspondant à la dernière frappe de touche (supprime le mot si la touche est programmée).

Pour terminer, si vous trouvez des bugs ou si vous avez des suggestions, faites le moi savoir.

Deux conseils :

- Ne pas utiliser le Clavier avec l'INTEGER BASIC.

- En ce qui concerne l'édition sur 80 colonnes : les cartes "incrústées" (carte texte Apple //e) ne supportent pas la routine moniteur KEYIN (et les fonctions Basic qui l'utilisent, INPUT et GET); il est donc impossible d'employer cet éditeur. Par contre, les cartes de type terminal vidéo (Superterm, Videx, etc...), fonctionnent parfaitement.

N.D.L.R. : ce programme tourne sur l'Apple //c. Il faut charger CLAVIER2E. Il est à noter que le curseur n'est pas visualisé.

CLAVIER.LIST

```
10 AD = 30000 : HIMEM : AD
20 INPUT "NOM DE VOTRE CLAVIER ? " : N$
30 PRINT CHR$(4) ; "BLOAD" ; N$ ; " , A" ; AD
35 PRINT CHR$(4) ; "PR#1" : PRINT N$ : PRINT
40 DC = 16 : ON PEEK (AD + 769) / 50 + 1
   GOTO 50,70
50 DC = 24 : CA = 48 : IMAX = 9 : AB = AD + 252
   : GOSUB 1000 : PRINT
```

```
60 DC = 8
70 CA = 65 : IMAX = 25 : AB = AD + 44 : GOSUB
   1000
75 PRINT CHR$(4) ; "PR#0"
80 END
1000 FOR I = 0 TO IMAX : AX = AB + DC * I :
   C$ = CHR$(CA + I) + " : "
1100 IF PEEK (AX) = 0 THEN PRINT C$ : N
   EXT : RETURN
1200 C$ = C$ + CHR$(PEEK (AX) - 128) : A
   X = AX + 1 : GOTO 1100
```

```
1 ;
2 ;*****
3 ;*
4 ;* CLAVIER PROGRAMMABLE *
5 ;*
6 ;* //E (A2E=1) ET II+ (A2E=0) *
7 ;* (ASSEMBLAGE CONDITIONNEL) *
8 ;*
9 ;* IMPLANTATION : $8600 A $8C51 *
10 ;*
11 ;*****
12 ;
13 ;
14 A2E EQU $1 ; OPTI
   ON D'ASSEMBLAGE
15 ;
16 ; ROUTINES SYSTEME
17 ;
18 GIVAYF EQU $E2F2
```

```
19 FNDLIN EQU $D61A
20 SAVE EQU $FF4A
21 DOSPTR EQU $03EA
22 RESTORE EQU $FF3F
23 MOVAF EQU $EB63
24 SNGFLT EQU $E301
25 FMULTT EQU $E982
26 FADDT EQU $E7C1
27 FOUT EQU $ED34
28 FRMNUM EQU $DD67
29 KSWDOS EQU $AA55
30 GETADR EQU $E752
31 LINPTR EQU $ED24
32 ;
33 ; ADRESSES SYSTEME
34 ;
35 TXTPTR EPZ $B8
36 MEMSIZ EPZ $73
37 LINNUM EPZ $50
```


148	STX KB				
149	.IF A2E				
150	CPX ##FF		;EST CE		
	UNE TOUCHE DEL ?				
151	BNE >3				
152	DEC CPTC				
153	BNE >4				
154	INC CPTC				
155 ^4	JMP DEL				
156	.FI				
157 ^3	LDA ##0				
158	STA CPTC				
159	.IF A2E				
160	LDA POM				
161	CPX ##89		;EST CE		
	LA TOUCHE HTAB ?				
162	BNE >2				
163	JMP TAB				
164 ;					
165 ;	TRANSFORMATION D'UNE FRAPPE DE CLAVIER APPLE2+				
166 ;					
167	.EL		;TRI DE		
	S TOUCHES CTRL				
168	STA POM		; NON		
	AUTORISEES.				
169	CPX ##9B				
170	BCS SUITE				
171	CPX ##80				
172	BEQ SUITE				
173	CPX ##83				
174	BEQ SUITE				
175	CPX ##88				
176	BEQ SUITE				
177	CPX ##8D				
178	BEQ SUITE				
179	CPX ##95				
180	BEQ SUITE				
181	CPX ##98				
182	BEQ SUITE				
183	CPX ##9A		;SI CTR		
	L Z ,ON SIMULE				
184	BNE >7		; POM P		
	LEINE +A OU N POUR				
185	LDA ##CE		; LA C		
	COMMUTATION CLAVIER				
186	STA KB				
187	BIT MODE				
188	BPL >8				
189	LDA ##C1				
190	STA KB				
191 ^8	LDA ##01				
192	STA POM				
193	JMP SUITE				
194 ^7	CPX ##82		;SI CTR		
	L B ,ON SIMULE POM				
195	BNE >9		;PLEINE		
	+P POUR L'APPEL DU				
196	LDA ##D0		; MODE		
	PROGRAMME				
197	STA KB				
198	JMP <8				
199 ^9	LDA ##80		;SI AUT		
	RE TOUCHE CONTROLE,				
200	STA POM		; POM 0		
	UV + TOUCHE ALPHA				
201	LSR				
202	CLC				
203	ADC KB				
204	STA KB				
205 SUITE	LDA POM				
206	LDX KB				
207 ;					
208	.FI				
209 ^2	BIT FLPR				;EST ON
	EN COURS DE PROG?				
210	BPL >1				
211	JMP PROGMT				
212 ^1	ASL				
213	BCS PRG1				;SI POM
	OUV ,SORTIE CHAINE				
214	LSR				
215	LSR				
216	BCS FONC1				;SI POM
	PLEINE ,COMMANDE				
217	BIT MODE				
218	BPL >2				
219	JSR REDEF				;SI CLA
	U NUM ,CHANG. CARACT.				
220 ^2	LDA KB				
221	CMP ##8D				;SI ON
	VIENT DE TAPER RET,				
222	BNE >1				; ALORS
	QU'ON SORTAIT				
223	BIT AUTOSOR				; UN NO
	DE LIGNE,				
224	BPL >1				
225	LDA ##0				
226	STA AUTOFL				;ON ANN
	ULE LE MODE AUTONUM				
227 ^1	LDA ##0				
228	STA AUTOSOR				;ON ANN
	ULE LE FLAG DE SORTIE				
229 SORT	INC CPTC				;ON COM
	PTE LES CARAC. SORTIS				
230	ASL STATUS				
231	JSR RESTORE				; ON RE
	MET TOUT DANS L'ETAT				
232	STA (\$28),Y				; OU ON
	L'A TROUVE ,				
233	LDA KB				
234	BIT KBSTROBE				; ET ON
	SE RETIRE SUR LA				
235	RTS				; POIN
	TE DES PIEDS				
236 FONC1	JMP FONC				
237 ;					
238 ;	ROUTINE D'INITIALISATION				
239 ;					
240 INIT	LDA ##0				;INITIA
	LISATION DES FLAGS				
241	STA TYPE				
242	STA PROG				
243	STA MODE				
244	STA FLPR				
245	STA STATUS				
246	STA AUTOFL				
247	LDA #ACCES				;POSITI
	ONNEMENT DE HIMEM				
248	STA MEMSIZ				; DEV
	ANT LE CODE.				
249	LDA /ACCES				
250	STA MEMSIZ+1				
251	NOP				
252	LDA #ENT				;MODIF
	DU POINTEUR KSW				
253	STA KSWDOS				
254	LDA /ENT				
255	STA KSWDOS+1				
256	RTS				
257 ;					
258 ;	REDEFINITION DU CLAVIER				
259 ;					
260 REDEF	LDA KB				;LECTUR
	E DE LA TABLE DE				
261	CMP ##FB				;REDEFI
	NITION INDEXEE PAR				
262	BCS FIND				;LE COD

```

E ASCII DU CARAC.
263      CMP #A0          ;
      -A0
264      BCC FIND
265      SBC #A0
266      TAY
267      LDA #ATR
268      STA ALP
269      LDA /ATR
270      STA ALP+1
271      LDA (ALP),Y
272      STA KB
273 FIND  RTS
274 ;
275 ;TOUCHES PROGRAMMEES
276 ;
277 PRG1  LDA #0          ;ANNULA
      TION DU FLAG DE SOR-
278      STA AUTOSOR      ;TIE DE
      NO DE LIGNE
279      JSR CALCA        ;CALCUL
      ADRESSE CHAINE
280      LDA TYPE         ;CONTRO
      LE
281      BEQ SORT2
282 ;
283 ;ROUTINE DE SORTIE D'UNE CHAINE POIN
      TEE PAR ALP ,
284 ;          TERMINEE PAR #0
      0
285 ;
286 PMES  ASL PROG        ;FLAG S
      ORTIE CHAINE
287      SEC
288      ROR PROG
289      LDA #0           ;POINTE
      UR EN DEBUT DE CHAINE
290      STA PTTB
291      TAY
292      LDA (ALP),Y      ;LECTUR
      E IER CARACTERE
293      BNE PRG2         ;SI #00
      TOUCHE "VIDE"
294      .IF A2E
295      JMP SORT
296      .EL
297      SEC
298      LDA KB
299      SBC #A40
300      STA KB
301      JMP SORT
302      .FI
303 PRG2  LDY PTTB        ;LECTUR
      E CHAINE ET SORTIE
304      LDA (ALP),Y      ; DU C
      ARACTEREE
305      BEQ FINP
306      STA KB
307      INC PTTB
308 SORT2 JMP SORT
309 FINP  ASL PROG        ;SI #00
      CHAINE TERMINEE ,
310      LSR PROG         ;ANNULE
      LE FLAG ET RETOUR
311      JMP LEC          ;EN LEC
      TURE CLAVIER
312 ;
313 ; ROUTINE DE CALCUL DE L'ADRESSE D'
      UNE CHAINE
314 ;
315 CALCA  LDA #0
316      STA TYPE
317      LDA KB           ;LA TOU
      CHE FRAPPEE EST ALPHA

```

```

318      CMP #DB          ;NUMERI
      QUE, OU NON
319      BCS SORT1       ;PROGRA
      MMEBLE
320      CMP #C1
321      BCS ALFA
322      .IF A2E
323      CMP #BA
324      BCS SORT1
325      CMP #B0
326      NOP
327      BCS NUM
328      .FI
329 SORT1  RTS
330 ALFA  SBC #C1         ;CALCUL
      INDEX CHAINE DANS
331      ASL              ;LA TAB
      LE ALPHA, ET RANGEM.
332      ASL              ;DU RES
      ULTAT DANS LE
333      ASL              ;POINTE
      UR ALP
334      .IF A2E
335      ADC #ATP
336      STA ALP
337      LDA /ATP
338      .EL
339      ASL
340      PHA
341      LDA #0
342      ADC /ATP
343      STA ALP+1
344      CLC
345      PLA
346      ADC #ATP
347      STA ALP
348      LDA ALP+1
349      .FI
350      ADC #0
351      STA ALP+1
352      ASL TYPE
353      SEC
354      ROR TYPE
355      RTS
356      .IF A2E
357 NUM   SBC #B0         ;IDEM A
      VEC TABLE TOUCHES
358      ASL              ;NUMERI
      QUES
359      ASL
360      ASL
361      STA CODE
362      ASL
363      ADC CODE
364      ADC #ATN
365      STA ALP
366      LDA /ATN
367      ADC #0
368      STA ALP+1
369      LSR TYPE
370      SEC
371      ROL TYPE
372      RTS
373      .FI
374 ;
375 ; FONCTIONS APPELLEES PAR POM PLEIN
      E
376 ;
377 FONC  LDA KB
378      CMP #D0          ;TRI P
      ?
379      BEQ TPR
380      CMP #CE
381      BEQ TNU          ; N ?

```

382	CMP #C1	; OU A	437	BEQ LEC1	
383	BEG TAL		438	CMP #F98	
384	JMP SORT		439	BEQ LEC1	
385	TNU LDA #FF	;PASSAG	440	CMP #B8D	;SI RET
	E EN MODE CLAV NUM.		441	URN,APPEL DU MODE NUM	
386	STA MODE			BNE >7	; DE
387	LDA #MSG2	; ET SO		LIGNES.	
	RTIE DU MESSAGE		442	JMP AUTOLIN	
388	STA ALP		443	BIT POM	
389	LDA /MSG2		444	BMI >3	
390	STA ALP+1		445	CMP #B88	;LES CA
391	JMP PMES		446	RACTERES FRAPPES SONT	
392	TAL LDA #00	;MODE A		BNE >1	;RANGES
	LPHA ET MESSAGE		447	DANS LE BUFFER AZT	
393	STA MODE			LDX LPR	;LE POI
394	LDA #MSG3		448	NTEUR X PREND EN	
395	STA ALP			BEQ LEC1	;COMPTE
396	LDA /MSG3		449	LES TOUCHES FLECHES	
397	STA ALP+1			DEC LPR	; AVANT
398	JMP PMES			ET ARRIERE.	
399			450	JMP SORT	
400	;ROUTINE DE PROGRAMMATION: VALIDATIO		451	^1 CMP #F95	;LE POI
	N			NTEUR CONTROLE LA	
401			452	BEQ >2	;LONG.
402	RANG1 LDA #0	; UN \$0		MAX DE LA CHAINE	
	0 DANS LE BUFFER,		453	LDX LPR	
403	STA AZT,X		454	STA AZT,X	
	EN FIN DE CHAINE		455	^2 INC LPR	;ON SOR
404	INX			T APRES FRAPPE	
405	TXA		456	LDX #F18	; TOUC
406	TAY			HE POMME OUVERTE.	
407	RECOP DEX		457	CPX LPR	
408	DEY		458	BCC <6	
409	BMI FINPR		459	JMP SORT	
410	LDA AZT,X	;ON REC	460	^3 JSR CALCA	
	OPIE LE BUFFER DANS		461	LDX LPR	
411	STA (ALP),Y	; LA TA	462	LDA TYPE	
	BLE DE RANGEMENT		463	BEQ LEC1	
412	JMP RECOP		464	BIT TYPE	
413	LEC1 JMP LEC		465	BPL >5	
414	FINPR LDA KB		466	.IF A2E	
415	STA CODE		467	CPX #B08	
416	ASL FLPR	;ON POS	468	.EL	
	ITIONNE LE FLAG DE		469	CPX #F10	
417	LDA #MSG1	;SORTIE	470	.FI	
	, ON AFFICHE LA		471	BMI >4	
418	STA ALP	;TOUCHE	472	.IF A2E	
	D'AFFECTATION		473	LDX #B07	
419	LDA /MSG1	;ET LE	474	.EL	
	MESSAGE CORRESPONDANT		475	LDX #B0F	
420	STA ALP+1		476	.FI	
421	JMP PMES	;SORTIE	477	^4 JMP RANG1	
422			478	^5 CPX #F18	
423	;ROUTINE DE PROGRAMMATION: SAISIE,ED		479	BMI RA1	
	ITION		480	LDX #F17	
424			481	RA1 JMP RANG1	
425	TPR LDA #MSG0	;PASSAG	482		
	E EN MODE PROGRAMME		483	;FONCTION TABULATION	
426	STA ALP	;SORTIE	484		
	DU MESSAGE.		485	TAB BIT POM	;TEST D
427	LDA /MSG0			E LA POMME OUV.	
428	STA ALP+1		486	BMI >1	;POUR L
429	LDA #B80			E SENS DE DEPLAC-	
430	STA FLPR		487	LDA #FTAB	;EMENT
431	LDA #0			CURSEUR.	
432	STA LPR		488	STA ALP	
433	JMP PMES		489	LDA /FTAB	;ON SOR
434				T UNE CHAINE DE	
435	PROGMT LDA KB	;INTERC	490	RL H OU CTRL U.	;10 CT
	EPTION DES FRAPPES		491	JMP PMES	
436	CMP #F9B	; REF	492	^1 LDA #BTAB	
	US DE ESC ET CTRL X.		493	STA ALP	
			494	LDA /BTAB	

```

495          STA ALP+1
496          JMP PMES
497 ;
498 ;FONCTION DEL
499 ;
500 DEL      SEC                      ;ON SOR
           T UNE CHAINE DE CPTC
501          LDA #ZDEL                ; FO
           IS CTRL U.
502          SBC CPTC
503          STA ALP
504          LDA /ZDEL
505          SBC #0
506          STA ALP+1
507          SEC
508          LDA #0
509          SBC CPTC
510          STA CPTC
511          JMP PMES
512 ;
513 ;NUMEROTATION LIGNES: SAISIE DES PA
           RAMETRES
514 ;
515 AUTOLIN  LDX LPR                  ; $00 E
           N FIN DU BUFFER AZT.
516          LDA #0
517          STA AZT,X
518          INX
519          STA AZT,X
520          DEX
521          STA NLIN+1
522          STA INCL+1
523          TAY
524          DEX
525          BMI >5
526 ^4      LDA AZT,X                  ;MISE E
           N FORME DU BUFFER.
527          AND #$7F
528 ^3      STA AZT,X
529          DEX
530          BPL <4
531 ^5      ASL FLPR
532          BIT KBSTROBE              ;DEBLOQ
           UER LE CLAVIER SI LES
533          LDA #AZT                  ; ROUTI
           NES FRMNUM,GETADR,
534          STA TXTPTR                ; "PLAN
           TENT" ET FONT SORTIR
535          LDA /AZT                  ; PAR H
           NDLERR.
536          STA TXTPTR+1
537          LDA #$0A                  ;ON POS
           ITIONNE LE TXTPTR,
538          STA NLIN                  ;ON RAN
           GE LES VALEURS PAR
539          STA INCL                  ; DEFAU
           T.
540          LDA (TXTPTR),Y
541          CMP #0
542          BEQ AUTOFIN
543          CMP #$2C
544          BEQ CALINC
545          JSR FRMNUM                ;CALCUL
           ET RANGEMENT DU NO
546          JSR GETADR                ; DE LI
           GNE
547          LDA LINNUM
548          STA NLIN
549          LDA LINNUM+1
550          STA NLIN+1
551 CALINC  LDY #0
552          INC TXTPTR
553          BNE >2
554          INC TXTPTR+1

```

```

555 ^2      LDA (TXTPTR),Y
556          CMP #0
557          BEQ AUTOFIN
558          CMP #$2C
559          BEQ AUTOFIN
560          JSR FRMNUM                ;CALCUL
           DE L'INCREMENT.
561          JSR GETADR
562          LDA LINNUM
563          ORA LINNUM+1
564          BEQ AUTOFIN
565          LDA LINNUM
566          STA INCL
567          LDA LINNUM+1
568          STA INCL+1
569 AUTOFIN LDA #$FF                  ;PREPA
           DES FLAGS ET SORTIE
570          STA AUTOFL                ; DU ME
           SSAGE.
571          ASL FLPR
572          LDA #NUL
573          STA ALP
574          LDA /NUL
575          STA ALP+1
576          JMP PMES
577 ;
578 ;NUMEROTATION LIGNES: SORTIE NUMERO
           S
579 ;
580 NUM1    CLC
581          LDA NLIN                  ;CALCUL
           NOUVEAU NO DE LIGNE
582          ADC INCL                  ; EN E
           NTIER 2 OCTETS.
583          STA NLIN
584          LDA NLIN+1
585          ADC INCL+1
586          STA NLIN+1
587 NUM2    LDA #$01                  ;CALCUL
           EN FLOTTANT,
588          LDY #$00
589          JSR GIVAYF
590          JSR MOVAF
591          LDY NLIN+1
592          STY LINNUM+1
593          JSR SNGFLT
594          JSR FMULTT
595          JSR MOVAF
596          LDY NLIN
597          STY LINNUM
598          JSR SNGFLT
599          JSR FADDT
600          JSR FOUT                  ;POUR U
           TILISER LA ROUTINE
601          STA ALP                    ;DE CON
           VERSION EN CHAINE.
602          STY ALP+1
603          LDA #$FF
604          STA AUTOSOR
605          LDY #0
606 ^1      LDA (ALP),Y
607          BEQ >2
608          ORA #$80                  ;TRANSF
           0. DES CHIFFRES EN
609          STA (ALP),Y                ; ASCII
           NEGATIF.
610          INY
611          JMP <1
612 ^2      LDA #$A0                  ;ON TER
           MINE PAR UN BLANC.
613          STA (ALP),Y                ; (C'ES
           T PLUS JOLI!!!!).
614          INY
615          TYA

```

```

616 PHA
617 JSR FNDLIN ;LA LIG
NE EXISTE ELLE ?
618 BCC >3
619 LDA #*0 ;OUI ,0
N ANNULE TOUT,
620 STA AUTOFL
621 STA AUTOSOR
622 PLA
623 TAY
624 LDA #*87 ; ON F
AIT COUINER
625 STA (ALP),Y ;ET ON
TERMINE PAR CTRL X.
626 INY
627 STA (ALP),Y
628 INY
629 LDA #*98
630 STA (ALP),Y

```

```

631 INY
632 TYA
633 PHA
634 PLA
635 TAY
636 LDA #*0
637 STA (ALP),Y
638 STA CPTC
639 JMP PMES ;ON SOR
T LE NUMERO PAR LA
640 ; VOIE
HABITUELLE.
641 ;
642 ;
643 ;AU REVOIR LES PETITS AMIS,ET A BIEN
TOT
644 ;
645 LONGUEUR = *-ACCES
646 END

```

CLAVIER 2 E

8600- 4C 78 89 00 00 00 00 00
87F6.8C50

```

87F6- 00 A0
87F8- A1 A2 A3 A4 A5 A6 A7 A8
8800- A9 AA AB AC AD AE AF B0
8808- B1 B2 B3 B4 B5 B6 B7 B8
8810- B9 BA BB BC BD BE BF C0
8818- C1 C2 C3 C4 C5 C6 C7 C8
8820- B5 B1 B2 B3 B0 CE B6 D0
8828- D1 D2 D3 D4 B4 D6 D7 D8
8830- D9 DA DB DC DD DE DF E0
8838- E1 E2 E3 E4 E5 E6 E7 E8
8840- B5 B1 B2 B3 B0 EE B4 F0
8848- F1 F2 F3 F4 B4 F6 F7 F8
8850- F9 FA D0 D2 CF C7 D2 C1
8858- CD CD C5 A0 BA A0 00 A0
8860- C1 C6 C6 C5 C3 D4 C5 A0
8868- C1 A0 00 A0 A0 98 00 A0
8870- C1 D5 D4 CF CE D5 CD A0
8878- 98 00 C2 CC CF C3 A0 CE
8880- D5 CD C5 D2 C9 D1 D5 C5
8888- A0 A0 98 00 C3 CC C1 D6
8890- C9 C5 D2 A0 C1 CC D0 C8
8898- C1 C2 C5 D4 C9 D1 D5 C5
88A0- 98 00 95 95 95 95 95 95
88A8- 95 95 95 95 00 88 88 88
88B0- 88 88 88 88 88 88 88 88
88B8- 88 88 88 88 88 88 88 88
88C0- 88 88 88 88 88 88 00 20
88C8- 4A FF 24 FE 10 03 4C D9
88D0- 89 2C 0C 86 10 03 20 EA
88D8- 03 0E 0C 86 38 6E 0C 86
88E0- EE 00 86 2C 10 C0 2C 07
88E8- 86 10 11 AD 0A 86 C9 98
88F0- 00 03 4C E3 88 C9 8D D0
88F8- 03 4C D0 88 2C 00 C0 10
8900- FB 2C 62 C0 08 AD 61 C0
8908- 29 80 AA 68 0A 8A 69 00
8910- 8D 10 86 AE 00 C0 8E 0A
8918- 86 E0 FF D0 0B CE 0D 86
8920- D0 03 EE 0D 86 4C 29 88
8928- A9 00 8D 0D 86 AD 10 86
8930- E0 89 D0 03 4C 0E 88 2C
8938- 0E 86 10 03 4C AF 8A 0A
8940- B0 78 4A 4A 80 2F 24 FF
8948- 10 03 20 9E 89 AD 0A 36
8950- C9 8D 00 0A 2C 08 86 10
8958- 05 A9 00 8D 07 86 A9 00
8960- 8D 08 86 EE 00 86 0E 0C
8968- 86 20 3F FF 91 28 AD 0A
8970- 86 2C 10 C0 60 4C 3F 8A
8978- A9 00 8D 08 86 85 FE 85
8980- FF 8D 0E 86 8D 0C 86 8D
8988- 07 86 A9 00 85 73 A9 86
8990- 85 74 EA A9 C7 8D 55 AA
8998- A9 88 8D 56 AA 60 AD 0A
89A0- 86 C9 FB 80 14 C9 A0 90
89A8- 10 E9 A0 A8 A9 F7 85 FC

```

```

89B0- A9 87 85 FD B1 FC 8D 0A
89B8- 86 60 A9 00 8D 08 86 20
89C0- F0 89 AD 08 86 F0 1F 06
89C8- FE 38 66 FE A9 00 8D 09
89D0- 86 A8 B1 FC D0 03 4C 63
89D8- 89 AC 09 86 B1 FC F0 09
89E0- 8D 0A 86 EE 09 86 4C 63
89E8- 89 06 FE 46 FE 4C D9 88
89F0- A9 00 8D 08 86 AD 0A 86
89F8- C9 DB 8D 0D C9 C1 B0 0A
8A00- C9 BA 80 05 C9 B0 EA 80
8A08- 18 60 E9 C1 0A 0A 69
8A10- 2F 85 FC A9 86 69 00 85
8A18- FD 0E 08 86 38 6E 0B 86
8A20- 60 E9 B0 0A 0A 8D 6A
8A28- 88 0A 6D 6A 88 69 FF 85
8A30- FC A9 86 69 00 85 FD 4E
8A38- 08 86 38 2E 08 86 60 AD
8A40- 0A 86 C9 D0 F0 54 C9 CE
8A48- F0 07 C9 C1 F0 12 4C 63
8A50- 89 A9 FF 85 FF A9 7A 85
8A58- FC A9 88 85 FD 4C C7 89
8A60- A9 00 85 FF A9 8C 85 FC
8A68- A9 88 85 FD 4C C7 89 A9
8A70- 00 9D 11 86 E8 8A 8A CA
8A78- 88 30 0B B0 11 86 91 FC
8A80- 4C 77 8A 4C D9 88 AD 0A
8A88- 86 8D 6A 88 0E 0E 86 A9
8A90- 5F 85 FC A9 88 85 FD 4C
8A98- C7 89 A9 52 85 FC A9 88
8AA0- 85 FD A9 80 8D 0E 86 A9
8AA8- 00 8D 0F 86 4C C7 89 AD
8AB0- 0A 86 C9 98 F0 CD C9 98
8AB8- 0F C9 C9 8D 00 03 4C 43
8AC0- 88 2C 10 86 30 26 C9 88
8AC8- D0 08 AE 0F 86 F0 B4 CE
8AD0- 0F 86 4C 63 89 C9 95 F0
8AD8- 06 AE 0F 86 9D 11 86 EE
8AE0- 0F 86 A2 18 EC 0F 86 90
8AE8- E6 4C 63 89 20 F0 89 AE
8AF0- 0F 86 AD 08 86 F0 8C 2C
8AF8- 08 86 10 09 E0 08 30 02
8800- A2 07 4C 6F 8A E0 18 30
8808- 02 A2 17 4C 6F 8A 2C 10
8810- 86 30 0B A9 A2 85 FC A9
8818- 88 85 FD 4C C7 89 A9 BC
8820- 85 FC A9 88 85 FD 4C C7
8828- 89 38 A9 C6 ED 0D 86 85
8830- FC A9 88 E9 00 85 FD 38
8838- A9 00 ED 0D 86 8D 00 86
8840- 4C C7 89 AE 0F 86 A9 00
8848- 9D 11 86 E3 9D 11 86 CA
8850- 8D 04 86 8D 06 86 A8 CA
8858- 30 08 B0 11 86 29 7F 9D
8860- 11 86 CA 10 F5 0E 0E 86
8868- 2C 10 C0 A9 11 85 88 A9
8870- 86 85 B9 A9 0A 8D 03 86
8878- 8D 05 86 B1 88 C9 00 F0
8880- 3C C9 2C F0 10 20 67 DD
8888- 20 52 E7 A5 50 8D 03 86
8890- A5 51 8D 04 86 A0 00 E6

```

```

8898- 88 00 02 E6 B9 B1 88 C9
88A0- 00 F0 1A C9 2C F0 16 20
88A8- 67 D0 20 52 E7 A5 50 05
88B0- 51 F0 0A A5 50 8D 05 86
88B8- A5 51 8D 06 86 A9 FF 8D
88C0- 07 86 0E 0E 86 A9 6F 85
88C8- FC A9 88 85 FD 4C C7 89
88D0- 18 AD 03 86 6D 05 86 8D
88D8- 03 86 AD 04 86 6D 06 86
88E0- 8D 04 86 A9 01 A0 00 20
88E8- F2 E2 20 63 EB AC 04 86
88F0- 84 51 20 01 E3 20 82 E9
88F8- 20 63 EB AC 03 86 84 50
8C00- 20 01 E3 20 C1 E7 20 34
8C08- ED 85 FC 84 FD A9 FF 8D
8C10- 08 86 A0 00 B1 FC F0 08
8C18- 09 80 91 FC C8 4C 14 8C
8C20- A9 A0 91 FC C8 98 48 20
8C28- 1A D6 90 19 A9 00 8D 07
8C30- 86 8D 08 86 68 A8 A9 87
8C38- 91 FC C8 91 FC C8 A9 98
8C40- 91 FC C8 98 48 68 A8 A9
8C48- 00 91 FC 8D 0D 86 4C C7
8C50- 89

```

CLAVIER 2 +

8600- 4C 8C 89 00 00 00 00 00
87DD.8C50

```

87DD- 00 A0 A1
87E0- A2 A3 A4 A5 A6 A7 A8 A9
87E8- AA AB AC AD AE AF B0 B1
87F0- B2 B3 B4 B5 B6 B7 B8 B9
87F8- BA BB BC BD BE BF C0 C1
8800- C2 C3 C4 C5 C6 C7 C8 B5
8808- B1 B2 B3 B0 CE B6 D0 D1
8810- D2 D3 D4 B4 D6 D7 D8 D9
8818- DA DB DC DD DE DF E0 E1
8820- E2 E3 E4 E5 E6 E7 E8 B5
8828- B1 B2 B3 B0 EE B4 F0 F1
8830- F2 F3 F4 B4 F6 F7 F8 F9
8838- FA D0 D2 CF C7 D2 C1 CD
8840- CD C5 A0 BA A0 00 A0 C1
8848- C6 C6 C5 C3 D4 C5 A0 C1
8850- A0 00 A0 A0 98 00 A0 C1
8858- D5 D4 CF CE D5 CD A0 98
8860- 00 C2 CC CF C3 A0 CE D5
8868- CD C5 D2 C9 D1 D5 C5 A0
8870- A0 98 00 C3 CC C1 D6 C9
8878- C5 D2 A0 C1 CC D0 C8 C1
8880- C2 C5 D4 C9 D1 D5 C5 98
8888- 00 95 95 95 95 95 95 95
8890- 95 95 95 00 88 88 88 88
8898- 88 88 88 88 88 88 88 88
88A0- 88 88 88 88 88 88 88 88
88A8- 88 88 88 88 88 00 20 4A
88B0- FF 24 FE 10 03 4C F6 89
88B8- 2C 0C 86 10 03 20 EA 03
88C0- 0E 0C 86 38 6E 0C 86 EE

```

```

88C8- 00 86 2C 10 C0 2C 07 86
88D0- 10 11 AD 0A 86 C9 98 D0
88D8- 03 4C E3 88 C9 8D D0 03
88E0- 4C 00 88 2C 00 C0 10 FB
88E8- AE 00 C0 8E 0A 86 A9 00
88F0- 8D 0D 86 8D 10 86 E0 98
88F8- B0 4B E0 80 F0 47 E0 83
8900- F0 43 E0 88 F0 3F E0 8D
8908- F0 3B E0 95 F0 37 E0 98
8910- F0 33 E0 9A D0 16 A9 CE
8918- 8D 0A 86 24 FF 10 05 A9
8920- C1 8D 0A 86 A9 01 8D 10
8928- 86 4C 45 89 E0 82 D0 08
8930- A9 D0 8D 0A 86 4C 24 89
8938- A9 8D 8D 10 86 4A 18 6D
8940- 0A 86 8D 0A 86 AD 10 86
8948- AE 0A 86 2C 0E 86 10 03
8950- 4C AF 8A 0A B0 78 4A 4A
8958- B0 2F 24 FF 10 03 20 B2
8960- 89 AD 0A 86 C9 8D D0 0A
8968- 2C 08 86 10 05 A9 00 8D
8970- 07 86 A9 00 8D 08 86 EE
8978- 0D 86 0E 0C 86 20 3F FF
8980- 91 28 AD 0A 86 2C 10 C0
8988- 60 4C 3F 8A A9 00 8D 08
8990- 86 85 FE 85 FF 8D 0E 86
8998- 8D 0C 86 8D 07 86 A9 00
89A0- 85 73 A9 86 85 74 EA A9
89A8- AE 8D 55 AA A9 88 8D 56
89B0- AA 6D AD 0A 86 C9 FB B0
89B8- 14 C9 A0 90 10 E9 A0 A8
89C0- A9 FE 85 FC A9 87 85 FD
89C8- B1 FC 8D 0A 86 60 A9 00
89D0- 8D 08 86 20 0D 8A AD 08
89D8- 86 F0 28 06 FE 38 66 FE
89E0- A9 00 8D 09 86 A8 B1 FC
89E8- D0 0C 38 AD 0A 86 E9 40
89F0- 8D 0A 86 4C 77 89 AC 09
89F8- 86 B1 FC F0 09 8D 0A 86
8A00- EE 09 86 4C 77 89 06 FE
8A08- 46 FE 4C C0 88 A9 00 8D
8A10- 0B 86 AD 0A 86 C9 DB B0
8A18- 04 C9 C1 B0 01 60 E9 C1
8A20- 0A 0A 0A 0A 48 A9 00 69
8A28- 86 85 FD 18 68 69 2F 85
8A30- FC A5 FD 69 00 85 FD 0E
8A38- 0B 86 38 6E 08 86 60 AD
8A40- 0A 86 C9 D0 F0 54 C9 CE
8A48- F0 07 C9 C1 F0 12 4C 77
8A50- 89 A9 FF 85 FF A9 61 85
8A58- FC A9 88 85 FD 4C DB 89
8A60- A9 00 85 FF A9 73 85 FC
8A68- A9 88 85 FD 4C DB 89 A9
8A70- 00 9D 11 86 E8 8A A8 CA
8A78- 88 30 0B B0 11 86 91 FC
8A80- 4C 77 8A 4C C0 88 AD 0A
8A88- 86 8D 51 88 0E 0E 86 A9
8A90- 46 85 FC A9 88 85 FD 4C
8A98- DB 89 A9 39 85 FC A9 88
8AA0- 85 FD A9 8D 8D 0E 86 A9
8AA8- 00 8D 0F 86 4C DB 89 AD
8AB0- 0A 86 C9 98 F0 C0 C9 98
8AB8- F0 C9 C9 8D 00 03 4C 43
8AC0- 8B 2C 10 86 30 26 C9 88
8AC8- D0 0E AE 0F 86 F0 B4 CE
8AD0- 0F 86 4C 77 89 C9 95 F0
8AD8- 06 AE 0F 86 9D 11 86 EE
8AE0- 0F 86 A2 18 EC 0F 86 90
8AE8- E6 4C 77 89 20 0D 8A AE
8AF0- 0F 86 AD 08 86 F0 8C 2C
8AF8- 0B 86 10 09 E0 10 30 02
8B00- A2 0F 4C 6F 8A E0 18 30
8B08- 02 A2 17 4C 6F 8A 2C 10
8B10- 86 30 0B A9 89 85 FC A9
8B18- 88 85 FD 4C DB 89 A9 A3
8B20- 85 FC A9 88 85 FD 4C DB
8B28- 89 38 A9 AD ED 0D 86 85
8B30- FC A9 88 E9 00 85 FD 38
8B38- A9 00 ED 0D 86 8D 0D 86
8B40- 4C DB 89 AE 0F 86 A9 00
8B48- 9D 11 86 E8 9D 11 86 CA
8B50- 8D 04 86 8D 06 86 A8 CA
8B58- 30 0B B0 11 86 29 7F 9D
8B60- 11 86 CA 10 F5 0E 0E 86
8B68- 2C 10 C0 A9 11 85 B8 A9
8B70- 86 85 B9 A9 0A 8D 03 86
8B78- 8D 05 86 B1 B8 C9 00 F0
8B80- 3C C9 2C F0 10 20 67 D0
8B88- 20 52 E7 A5 50 8D 03 86
8B90- A5 51 8D 04 86 A0 00 E6
8B98- B8 D0 02 E6 B9 B1 B8 C9
8BA0- 00 F0 1A C9 2C F0 16 20
8BA8- 67 D0 20 52 E7 A5 50 05
8BB0- 51 F0 0A A5 50 8D 05 86
8BB8- A5 51 8D 06 86 A9 FF 8D
8BC0- 07 86 0E 0E 86 A9 56 85
8BC8- FC A9 88 85 FD 4C DB 89
8BD0- 18 AD 03 86 6D 05 86 8D
8BD8- 03 86 AD 04 86 6D 06 86
8BE0- 8D 04 86 A9 01 A0 00 20
8BE8- F2 E2 20 63 EB AC 04 86
8BF0- 84 51 20 01 E3 20 82 E9
8BF8- 20 63 EB AC 03 86 84 50
8C00- 20 01 E3 20 C1 E7 20 34
8C08- ED 85 FC 84 FD A9 FF 8D
8C10- 08 86 A0 00 B1 FC F0 08
8C18- 09 80 91 FC C8 4C 14 8C
8C20- A9 A0 91 FC C8 98 48 20
8C28- 1A D6 90 19 A9 00 8D 07
8C30- 86 8D 08 86 68 A8 A9 87
8C38- 91 FC C8 91 FC C8 A9 98
8C40- 91 FC C8 98 48 68 A8 A9
8C48- 00 91 FC 8D 0D 86 4C DB
8C50- 89

```

Visualisation d'une Distribution Normale

Daniel Hirst

Dans de très nombreuses situations, on peut observer que la valeur d'une variable quelconque est fonction d'un grand nombre d'évènements élémentaires dont les valeurs individuelles sont incertaines.

L'exemple le plus couramment utilisé est celui de la longueur d'une queue de file d'attente à un guichet. En supposant que le temps de traitement de chaque client par l'employé est constant, la longueur de la queue dépend du rythme d'arrivée des nouveaux clients; or, le temps écoulé entre deux arrivées est une variable aléatoire. On démontre que, pour un nombre infini d'évènements élémentaires, la fonction globale suit une distribution dite "normale" dont la représentation graphique est appelée

courbe de Gauss et suit l'équation :

$$Y = \frac{1}{s\sqrt{2\pi}} \times e^{-\frac{(x-m)^2}{2s^2}}$$

avec m la moyenne arithmétique, s l'écart-type (c'est à dire l'écart moyen par rapport à la moyenne), et pi=3,14159.. bien sûr.

Le programme NORM-DISTR présente une animation où un grand nombre de petites variations aléatoires et indépendantes peuvent s'accumuler et s'annuler mutuellement pour aboutir à une distribution de ce type.

Chaque point commence en haut et au milieu de l'écran et tombe jusqu'à la ligne horizontale. En tombant il se

déplace de façon aléatoire soit à gauche soit à droite, un peu comme un flocon de neige qui tombe. Une fois arrivés à la ligne horizontale, les points tombent jusqu'en bas de l'écran où ils s'entassent progressivement. Un petit trait horizontal indique à chaque instant la valeur moyenne de la distribution.

Pour obtenir la valeur chiffrée de la moyenne et de l'écart-type, il suffit à tout moment d'appuyer sur la touche 'S' (statistiques). On peut ensuite soit revenir à la visualisation, soit demander l'affichage de la courbe normale. La touche 'Q' permet d'arrêter la visualisation. Le programme est assez lent et gagne beaucoup à être compilé, par Expediter II, par exemple. ■

```

1 REM VISUALISATION D'UNE
2 REM DISTRIBUTIION NORMALE
3 REM ***D.J.HIRST 1984***
5 REM =====
6 REM
7 REM ----REVISE 7-JUN-84 ----
8 REM
10 REM -----CONSTANTES
20 NP = 7
30 NT = 500
40 NY = 140
50 NC = NT * NY / NP
100 REM -----TABLEAUX
110 DIM HI(279)
120 DIM X(NP)
130 DIM Y(NP)
199 GOTO 1010: REM ----->DEBUT
200 REM -----S/R INCR
210 IA = INT ( RND (1) + .5) * 4 - 2
220 X2 = X1 + IA
230 Y2 = Y1 + 1
299 RETURN
300 REM -----S/R PLOT
310 HCOLOR= 0: HPLOT X1,Y1

```

```

320 IF X2 < 0 THEN X2 = 0
330 IF X2 > 279 THEN X2 = 279
340 HCOLOR= 3: HPLLOT X2,Y2
399 RETURN
400 REM -----S/R HISTO
410 HI(X1) = HI(X1) + 1
420 Y2 = 191 - HI(X1)
430 X2 = X1:NV = NV + 1
440 SX = SX + X1
450 SC = SC + X1 * X1
460 X(K) = 140:Y(K) = 0
470 MX = SX / NV
480 HCOLOR= 0: HPLLOT 0,191 TO 279,191
490 HCOLOR= 3: HPLLOT MX - 1,191 TO MX +
    1,191
499 RETURN
500 REM -----S/R HGR P1
510 POKE 49232,0: POKE 49234,0: POKE 492
    36,0: POKE 49239,0
520 RETURN
600 REM -----S/R COURBE NORMALE
610 C1 = 0.398942 / ET
620 C2 = - 1 * ET * ET
630 HCOLOR= 3
640 FOR X2 = 0 TO 279
650 Y2 = C1 * EXP ((X2 - MX) * (X2 - MX)
    / C2)
660 Y2 = Y2 * NV * 5
670 HPLLOT X2,190 - Y2
680 NEXT X2
699 RETURN
700 REM -----S/R TOUCHE CLAVIER
710 IF TC = 209 THEN GOTO 2010
720 IF TC = 211 THEN GOSUB 810: REM <-
    --> STATISTIQUES
799 RETURN
800 REM ----- S/R STATISTIQUES
810 IF NV < 2 THEN RETURN
815 TEXT
820 MX = SX / NV
825 MA = INT (MX * 1000 + .5) / 1000 - 1
    40
830 ET = SQR ((SC - SX * SX / NV) / (NV
    - 1))
835 EA = INT (ET * 1000 + .5) / 1000
840 PRINT : PRINT "          N = ";NV: PR
    INT " MOYENNE = ";MA: PRINT "ECA
    RT TYPE = ";EA: PRINT : PRINT
850 INPUT "VOULEZ-VOUS LA COURBE NORMALE
    ? ";A#
360 GOSUB 510: REM <---> HGR P1
870 IF A# = "0" OR "A#" = "OUI" THEN GO
    SUB 610: REM <--->
    COURBE NORMALE
899 RETURN
1000 REM -----PRESENTATION
1010 HOME : VTAB 12: INVERSE
1020 HTAB 7: PRINT "VISUALISATION D'UNE
    "
1030 HTAB 7: PRINT "DISTRIBUTION NORMALE
    " : NORMAL
1040 PRINT : PRINT : PRINT : PRINT " T
    APEZ 'S' POUR LES STATISTIQUES"
1050 PRINT " 'Q' POUR S'ARRETER
    "
1060 PRINT : PRINT : PRINT : HTAB 10: PR
    INT "FAIRE RETURN >";: GET A#
1100 REM -----AFFICHE CADRE
1110 HGR : POKE 49234,0
1120 HCOLOR= 3: HPLLOT 140,0 TO 140,NY
1130 HPLLOT 0,NY TO 279,NY
1200 REM -----INITIALISATIONS
1210 FOR I = 1 TO NP:X(I) = 140:Y(I) =
    - 1: NEXT I
1220 FOR I = 1 TO 279:HI(I) = 0: NEXT
1230 NV = 0
1240 SX = 0:SC = 0
1300 REM -----PROGRAMME PRINCIPAL
1310 FOR I = 1 TO NC
1315 IF I < = NP THEN Y(I) = 0
1320 FOR J = 1 TO NY / NP
1325 FOR K = 1 TO NP
1330 TC = PEEK ( - 16384)
1335 IF TC > 127 THEN POKE - 16368,0:
    GOSUB 710: REM <----> TOUCHE
    CLAVIER
1340 IF Y(K) < 0 THEN 1385
1345 X1 = X(K):Y1 = Y(K)
1350 GOSUB 210: REM --->INCR
1355 X(K) = X2:Y(K) = Y2
1360 IF Y2 > NY THEN GOSUB 410: REM -
    ---> HISTO
1370 GOSUB 310: REM ---->PLOT
1380 NEXT K
1385 NEXT J
1390 NEXT I
2000 REM -----FIN
2010 GOSUB 810: REM ---> STATISTIQUES
2020 GET A#
2030 TEXT

```

Max : Le moniteur étendu

Jacques Supernant

Apple II+, IIe, IIc

Ce moniteur autorise un contrôle de l'exécution des routines en langage machine.

- *Un mode Trace et Pas à Pas très évolués et sélectifs sont complétés par un accès direct aux registres du 6502 (ou 65C02). Il est ainsi possible de les initialiser à son gré ou à l'aide de la routine de hasard.*
- *La gestion des fenêtres d'écran simplifie le mode Trace.*
- *Une routine permet la recherche de suites d'octets.*
- *Un mini-assembleur très souple fait partie de MAX.*
- *Une ligne de commande peut devenir une boucle avec l'ordre JUMP.*

Les fichiers source sont sur la disquette.

Disquette et documentation : 150,00 F TTC franco (bon de commande page 74).

La version B4 du ProDOS, bien qu'ancienne et non commercialisée par Apple, est une version que certains d'entre vous possèdent. C'est pourquoi, nous apportons les modifications nécessaires au bon déroulement des programmes, publiés dans le précédent numéro, de l'article Pot-Pourri ProDOS.

Dans le programme Basic, il faut remplacer la ligne 117 par :

IF PEEK (48896) <> 76 THEN.....

Dans le programme assembleur, la ligne 47 devient :

JSR \$9C9D

De ce fait, il faut modifier le code objet correspondant :

032A-20 9D 9C

D'autre part, pour ceux qui ont la disquette d'accompagnement, un fichier étant mal nommé, il faut remplacer la ligne 6000 du programme Basic : BLOAD CQFD.DOS.16K

Transformez votre Apple //e en Apple II.

François Sermier

Comment tourner l'incompatibilité entre II et //e.

L'Apple //e diffère de son prédécesseur par des modifications matérielles (carte langage intégrée et RESET différent), mais il comporte également des modifications du système d'exploitation de base, principalement en ce qui concerne les routines d'entrées / sorties d'un caractère. Si l'on étudie attentivement le listing de la ROM moniteur, fourni dans le manuel de référence de l'Apple //e, on remarque que l'assemblage conditionnel se trouve en 9 endroits du moniteur, allant de \$FA75 à \$FD83, et donne un total de 137 octets de différence entre le moniteur //e et celui du II.

Les logiciels écrits pour l'Apple II tournent sur le //e s'ils utilisent les sous-programmes d'entrées / sorties standards (COUT, RDKEY, KEYIN, GETLN). Malheureusement, il semblerait que certains d'entre eux ne puissent tourner sur le //e, c'est le cas de l'APP-L-ISP. On peut envisager de modifier à la main, dans l'interpréteur LISP, les appels aux fonctions d'entrées / sorties, mais il faut alors analyser quelques 6 Ko de langage machine pour un résultat qui n'est ni garanti, ni général.

La solution retenue consiste à utiliser la mémoire à bancs commutés (ex-carte langage) pour y installer un moniteur d'Apple II et à commuter la dite mémoire pour fonctionner en émulation d'un II.

Quelques remarques

Ceci est donc valable pour tout programme n'utilisant pas la carte langage, par contre, si l'on souhaite que le RESET ne détruise pas le mode émulation, il faut se passer de l'affichage 80 colonnes.

Si vous n'avez pas la disquette d'accompagnement, il faut introduire les modifications du moniteur II, puis sauvegarder le nouveau moniteur ainsi obtenu pour un appel ultérieur dans un programme.

Modification du moniteur

La séquence des opérations est la suivante :

- CALL-151 : pour passer en moniteur.
- D000<D000.FFFFM : copie le moniteur en RAM (à l'initialisation,

l'Apple lit en ROM et écrit dans le second banc de RAM).

- FA75: AD 5D C0 ... : introduire les 137 octets modifiés d'après la liste ci-dessous.

FA75: AD 5D C0 AD 5F C0

FB51: A9 28 85 21

FBA3: 0C

FBB3: EA EA EA EA EA EA EA EA

FBB8: EA EA EA EA EA

FC42: A4 24 A5 25 48 20

FC48: 24 FC 20 9E FC A0 00 68

FC50: 69 00 C5 23 90 F0 B0 CA

FC58: A5 22 85 25 A0 00 84 24

FC60: F0 E4

FC70: A5 22 48 20 24 FC A5 28

FC78: 85 2A A5 29 85 2B A4 21

FC80: 88 68 69 01 C5 23 B0 0D

FC88: 48 20 24 FC B1 28 91 2A

FC90: 88 10 F9 30 E1 A0 00 20

FC98: 9E FC B0 86 A4 24 A9 A0

FCA0: 91 28 C8 C4 21 90 F9 60

FD1B: E6 4E D0 02 E6

FD20: 4F 2C 00 C0 10 F5 91 28

FD28: AD 00 C0 2C 10 C0 60 20

FD31: 0C

FD42: 85 32

FD83: DF

- C080 puis RETURN : la lecture de cette adresse commute le second banc de RAM en lecture; ainsi, vous êtes en présence d'un II reconnaissable à son curseur plein.
- BSAVE APPLE
2+.OBJ,A\$FA75,L\$30F : pour conserver sur disque ces modifications.
- ou BSAVE APPLE
2+.OBJ,A\$FA75,L\$2CF : pour garder l'usage des minuscules du //e
- 3D0G si vous voulez rester en II, ou bien CTRL-RESET si vous voulez retrouver votre //e.

Il ne reste plus qu'à répéter la même chose dans un programme pour retrouver, à volonté, un Apple II.

Programme de transformation

Pour ce faire, nous utilisons le sous-programme, bien connu, de

S.H.LAM, cité dans Pom's 2, entre autres, pour passer une commande au moniteur à partir d'un programme Basic.

- Ligne 40 : on reconnaît la copie du moniteur en RAM.

- Ligne 50 : les modifications sont introduites à partir du disque. N'oubliez pas que vous devez être dans la même configuration qu'à l'initialisation : lecture en ROM, écriture en RAM. En cas de doute, faire auparavant en mode immédiat : PRINT PEEK (49281) qui commute en mode normal.

- Ligne 60 : modification du RESET. L'Apple //e commute automatiquement en mode normal (lecture ROM, écriture deuxième banc RAM) lors d'un RESET, il faut donc le réexpédier en lecture / écriture sur la RAM si l'on veut rester en II. C'est le rôle du petit sous-programme en \$9DC5 :

LDA \$C083

LDA \$C083

RTS

Lors d'un RESET, il y a saut au point d'entrée à chaud du DOS en \$9DBF. Nous l'avons modifié (normalement, il teste quel Basic est actif et où il est situé) pour qu'il exécute le sous-programme en \$9DC5, puis reprenne le déroulement normal du Warmstart Basic Applesoft en \$9DD0.

\$9DBF: JSR \$9DC5

\$9DC2: JMP \$9DD0

Enfin, toujours en ligne 60, 9DC5G, passé par la routine S.H.LAM, commute la mémoire vive en lecture / écriture et l'on est de nouveau en présence d'un Apple II. Pour retourner sur le //e il suffit de faire :

X = PEEK (49281).

De la même manière, pour retrouver le II on fera :

X = PEEK (-16256)

Au lecteur de substituer au END de la ligne 70 toute instruction à sa convenance, permettant le chaînage ou le chargement du programme si longtemps convoité, afin de disposer du programme de "boot" lançant un //e sur un logiciel jusqu'alors réservé à son petit frère.

A titre d'exemple, pour l'APP-L-ISP, le HELLO comprendra les deux instructions suivantes : BLOAD SLIST et BRUN LISP-CODE.

```

10 PRINT CHR$(21): HOME
20 PRINT : PRINT " PATIENCE, JE ME TRANS
   FORME EN"
30 PRINT : HTAB 15: INVERSE : PRINT "APP
   LE 2+": NORMAL
40 Y$ = "D000<D000.FFFFF": GOSUB 150
50 PRINT CHR$(4);"BLOAD APPLE 2+.OBJ"
60 Y$ = "9DBF:20 C5 9D 4C D0 9D AD 83 C0
   AD 83 C0 60 N 9DC5G": GOSUB 150
    
```

```

70 END
150 REM ROUTINE S.H.LAM (POM'S [2]
160 REM
170 Y$ = Y$ + " N D9C6G"
180 FOR I = 1 TO LEN(Y$)
190 POKE 511 + I, ASC ( MID$(Y$,I,1)) +
   128
200 NEXT : POKE 72,0: CALL - 144
210 RETURN
JPR#0
    
```

Conversion chiffres-lettres

François Fleury

NUM-ALPHA est un programme de conversion de chiffres en lettres de toutes sommes supérieures à 1 centime et inférieures à mille milliards de francs.

Ce programme pourra donc être utile à tous ceux d'entre vous qui établissent des factures, pour les collectivités publiques, ou des chèques à partir d'une imprimante. Il est constitué de deux parties :

- Les lignes 9000 à 9265 contiennent la routine de saisie de la somme en chiffres.
- Les lignes 10000 à 10850 effectuent le formatage en lettres, en tenant compte des diverses particularités de la langue française, qu'il n'est peut-être pas inutile de rappeler :

- Cent n'est jamais précédé de "un".
- Cent et quatre-vingt s'accordent s'ils ne sont suivis d'aucun chiffre.
- Mille est invariable et n'est précédé de rien si le nombre de milliers est inférieur à deux.
- On place "de" (millions ou milliards DE francs) si aucun chiffre ne suit.
- etc..

La structure est volontairement modulaire (seules des variables Applesoft commençant par Y et Z sont utilisées) afin de faciliter la fusion de ce programme avec votre chef-d'oeuvre préféré !

N.D.L.R. : un lecteur nous a pris en flagrant délit ! En effet, dans Pom's 16 (Trucs et astuces p. 70), nous avons oublié de publier le source de la routine HSCREEN que nous évoluons. Sans plus tarder nous réparons notre erreur.

0300-	20 58 FF	JSR	\$\$F58
0303-	BA	TSX	
0304-	CA	DEX	
0305-	BD 00 01	LDA	\$0100,X
0308-	18	CLC	
0309-	49 17	ADC	£\$17
030B-	85 08	STA	\$08
030D-	BD 01 01	LDA	\$0101,X
0310-	69 00	ADC	£\$00
0312-	85 0C	STA	\$0C
0314-	A9 4C	LDA	£\$4C
0316-	85 0A	STA	\$0A
0318-	60	RTS	
0319-	20 0C E1	JSR	\$E10C
031C-	A6 A1	LDX	\$A1
031E-	A4 A0	LDY	\$A0
0320-	A5 E2	LDA	\$E2
0322-	20 13 F4	JSR	\$F413
0325-	A4 E5	LDY	\$E5
0327-	B1 26	LDA	(\$26),Y
0329-	25 30	AND	\$30
032B-	A8	TAY	
032C-	4C 01 E3	JMP	\$E301

```

9000 REM *****
9005 REM MODULE DE SAISIE D'UNE SOMME E
   N CHIFFRES AU CLAVIER
9010 REM TOUTES LES VARIABLES COMMENCEN
   T PAR Y
9015 REM *****
9020 CLEAR : HOME : VTAB 3: HTAB 20
9025 INVERSE : PRINT " " ;: NORMAL : PR
   INT " " ;
9030 INVERSE : PRINT " " ;: NORMAL : PR
   INT " " ;
9035 INVERSE : PRINT " " ;: NORMAL : PR
   INT " " ;
9040 INVERSE : PRINT " " ;: NORMAL : PR
   INT " " ;
9045 INVERSE : PRINT " " ;: NORMAL
9050 DIM YA$(50)
9055 I = 34:YN = 0
9060 VTAB 3: PRINT "TOTAL FACTURE....."
9065 YB$ = YB$ + YA$(I)
9066 YN = YN + 1
9070 IF YN = 13 THEN YA$(I) = CHR$(13)
   : GOTO 9105
9075 VTAB 3: HTAB I
9080 GET YA$(I): PRINT YA$(I);
9085 IF YA$(I) = "0" OR YA$(I) = "1" OR
   YA$(I) = "2" OR YA$(I) = "3" OR YA
   $(I) = "4" OR YA$(I) = "5" THEN G
   OTO 9105
9090 IF YA$(I) = "6" OR YA$(I) = "7" OR
   YA$(I) = "8" OR YA$(I) = "9" THEN
   GOTO 9105
    
```

```

9095 IF YA$(I) = CHR$(13) THEN GOTO 9
   235
9100 GOTO 9020
9105 IF YA$(I) = CHR$(13) THEN GOTO 9
   235
9110 IF YN = 1 THEN GOTO 9220
9115 HTAB (I - 1): PRINT YA$(I - 1);: IF
   YN = 2 THEN GOTO 9215
9120 HTAB (I - 2): PRINT YA$(I - 2);: IF
   YN = 3 THEN GOTO 9210
9125 HTAB (I - 4): PRINT YA$(I - 4);: IF
   YN = 4 THEN GOTO 9205
9130 HTAB (I - 5): PRINT YA$(I - 5);: IF
   YN = 5 THEN GOTO 9200
9135 HTAB (I - 6): PRINT YA$(I - 6);: IF
   YN = 6 THEN GOTO 9195
9140 HTAB (I - 8): PRINT YA$(I - 8);: IF
   YN = 7 THEN GOTO 9190
9145 HTAB (I - 9): PRINT YA$(I - 9);: IF
   YN = 8 THEN GOTO 9185
9150 HTAB (I - 10): PRINT YA$(I - 10);:
   IF YN = 9 THEN GOTO 9180
9155 HTAB (I - 12): PRINT YA$(I - 12);:
   IF YN = 10 THEN GOTO 9175
9160 HTAB (I - 13): PRINT YA$(I - 13);:
   IF YN = 11 THEN GOTO 9170
9165 HTAB (I - 14): PRINT YA$(I - 14);:
   IF YN = 12 THEN GOTO 9230
9170 YA$(I - 14) = YA$(I - 13)
9175 YA$(I - 13) = YA$(I - 12)
9180 YA$(I - 12) = YA$(I - 10)
9185 YA$(I - 10) = YA$(I - 9)
9190 YA$(I - 9) = YA$(I - 8)
9195 YA$(I - 8) = YA$(I - 6)
    
```

```

9200 YA$(I - 6) = YA$(I - 5)
9205 YA$(I - 5) = YA$(I - 4)
9210 YA$(I - 4) = YA$(I - 2)
9215 YA$(I - 2) = YA$(I - 1)
9220 YA$(I - 1) = YA$(I)
9225 Y = POS (1)
9230 GOTO 9065
9235 VTAB 3: HTAB (I + 2): GET YA$(I + 2)
      ): IF YA$(I + 2) = CHR$(13) THEN
      PRINT "00": GOTO 9255
9240 PRINT YA$(I + 2);:YB$ = YB$ + "." +
      YA$(I + 2)
9245 HTAB (I + 3): GET YA$(I + 3): IF YA
      $(I + 3) = CHR$(13) THEN PRINT
      "0": GOTO 9255
9250 PRINT YA$(I + 3);:YB$ = YB$ + YA$(I
      + 3)
9255 IF YN = 12 THEN GOTO 9263
9256 IF YN = 13 THEN GOTO 9265
9260 FOR YT = 0 TO Y - 21: VTAB 3: HTAB
      (20 + YT): PRINT " ": NEXT YT: GOT
      O 9265
9263 VTAB 3: HTAB 20: PRINT " "
9265 GOTO 10000
10000 REM *****
10005 REM *****
10010 REM MODULE DE TRANSFORMATION DE S
      OMME EN CHIFFRES A SOMME EN LETTRE
      S
10015 REM TOUTES LES VARIABLES COMMENCE
      NT PAR Z
10020 REM *****
10025 VTAB 10: HTAB 1: PRINT "CERTIFIEE
      CONFORME ET VERITABLE,LA PRES-ENTE
      FACTURE ARRETEE A LA SOMME DE....
      .."
10030 ZK$ = "C.C.P BORDEAUX NO 123456 "
10035 ZT$ = "CENT"
10040 ZW$ = "CENTIME"
10045 ZE$ = "MILLE"
10050 ZF$ = "FRANC"
10055 ZN$ = "MILLION"
10060 ZD$ = "MILLIARD"
10065 REM ENTREE DE LA CHAINE DE CARA
      CTERE QUI CONSTITUE LE NOMBRE A ET
      UDIER
10070 ZH$ = YB$
10075 REM NOMBRE ENTIER OU AVEC DECIM
      ALE?
10080 REM CAS PARTICULIER DES NOMBRES D
      E LONGUEUR Z1<=2 AVEC OU SANS DECI
      MALE
10085 REM A LA SUITE TOUS LES NOMBRES
      SONT AVEC 2 DECIMALES
10090 Z1 = LEN (ZH$)
10095 IF Z1 = 2 AND VAL (ZH$) > = 10 T
      HEN ZH$ = ZH$ + ".00":Z1 = LEN (Z
      H$): GOTO 10110
10100 IF Z1 = 1 AND VAL (ZH$) > 0 THEN
      ZH$ = ZH$ + ".00":Z1 = LEN (ZH$):
      GOTO 10110
10105 IF Z1 = 2 AND VAL (ZH$) > = .10
      THEN ZH$ = ZH$ + "0":Z1 = LEN (ZH
      $): GOTO 10110
10110 REM
10115 IF MID$(ZH$,Z1 - 2,1) = "." THEN
      GOTO 10130
10120 IF MID$(ZH$,Z1 - 1,1) = "." THEN
      ZH$ = ZH$ + "0":Z1 = Z1 + 1: GOTO
      10130
10125 ZH$ = ZH$ + ".00":Z1 = Z1 + 3
10130 REM
10135 REM MILLE MILLION MILLIARD
10140 REM Z3=PARTIE ENTIERE

```

```

10145 Z3 = Z1 - 3
10150 IF Z3 = 0 THEN GOTO 10375
10155 IF Z3 > 0 THEN Z4 = 1
10160 IF Z3 > 3 THEN Z4 = 2
10165 IF Z3 > 6 THEN Z4 = 3
10170 IF Z3 > 9 THEN Z4 = 4
10175 REM Z5=NOMBRE DE CHIFFRES DU GROU
      PE DE 3 LE PLUS A GAUCHE
10180 Z5 = Z3 - ((Z4 - 1) * 3):Z2$ = LEF
      T$(ZH$,Z5)
10185 IF Z4 = 2 AND Z2$ = "1" THEN PRIN
      T "MILLE ";: GOTO 10260
10190 Z2 = VAL (Z2$): GOSUB 10425
10195 IF Z4 = 4 AND MID$(ZH$,Z5 + 1,9)
      = "000000000" AND Z2 = 1 THEN PR
      INT " ";ZD$;" "; "DE FRANCS ";: GOT
      O 10375
10200 IF Z4 = 4 AND MID$(ZH$,Z5 + 1,9)
      = "000000000" AND Z2 > 1 THEN PR
      INT " ";ZD$ + "S";" "; "DE FRANCS "
      ;: GOTO 10375
10205 IF Z4 = 3 AND MID$(ZH$,Z5 + 1,6)
      = "000000" AND Z2 = 1 THEN PRINT
      " ";ZN$;" "; "DE FRANCS ";: GOTO 1
      0375
10210 IF Z4 = 3 AND MID$(ZH$,Z5 + 1,6)
      = "000000" AND Z2 > 1 THEN PRINT
      " ";ZN$ + "S";" "; "DE FRANCS ";:
      GOTO 10375
10215 IF Z4 = 4 AND Z2 > 1 THEN PRINT "
      ";ZD$ + "S";" ";
10220 IF Z4 = 4 AND Z2 = 1 THEN PRINT "
      ";ZD$;" ";
10225 IF Z4 = 3 AND Z2 > 1 THEN PRINT "
      ";ZN$ + "S";" ";
10230 IF Z4 = 3 AND Z2 = 1 THEN PRINT "
      ";ZN$;" ";
10235 IF Z4 = 2 AND Z2 > 1 THEN PRINT "
      ";ZE$;" ";
10240 IF Z4 = 1 AND Z2 > 1 THEN PRINT "
      ";ZF$ + "S";" ";
10245 IF Z4 = 1 AND Z2 < = 1 THEN PRIN
      T " ";ZF$;" ";
10250 IF Z4 = 1 THEN GOTO 10375
10255 IF Z4 = 3 AND MID$(ZH$,Z5 + 1,3)
      = "001" THEN PRINT " ";ZE$;" ";:
      GOTO 10305
10260 Z2$ = MID$(ZH$,Z5 + 1,3)
10265 Z2 = VAL (Z2$): GOSUB 10425
10270 IF Z4 = 4 AND Z2 > 1 THEN PRINT "
      ";ZN$ + "S";" ";
10275 IF Z4 = 4 AND Z2 = 1 THEN PRINT "
      ";ZN$;" ";
10280 IF Z4 = 3 AND Z2 > 0 THEN PRINT "
      ";ZE$;" ";
10285 IF Z4 = 2 THEN PRINT " ";ZF$ + "S
      ";" ";
10290 IF Z4 = 2 THEN GOTO 10375
10295 IF Z4 = 1 THEN GOTO 10375
10300 IF Z4 = 4 AND MID$(ZH$,Z5 + 4,3)
      = "001" THEN PRINT " ";ZE$;" ";:
      GOTO 10340
10305 Z2$ = MID$(ZH$,Z5 + 4,3)
10310 Z2 = VAL (Z2$): GOSUB 10425
10315 IF Z4 = 4 AND Z2 > 0 THEN PRINT "
      ";ZE$;" ";
10320 IF Z4 = 3 THEN PRINT " ";ZF$ + "S
      ";" ";
10325 IF Z4 = 3 THEN GOTO 10375
10330 IF Z4 = 2 THEN GOTO 10375
10335 IF Z4 = 1 THEN GOTO 10375
10340 Z2$ = MID$(ZH$,Z5 + 7,3)
10345 Z2 = VAL (Z2$): GOSUB 10425
10350 IF Z4 = 4 THEN PRINT " ";ZF$ + "S

```

```

";" ";
10355 IF Z4 = 4 THEN GOTO 10375
10360 IF Z4 = 3 THEN GOTO 10375
10365 IF Z4 = 2 THEN GOTO 10375
10370 IF Z4 = 1 THEN GOTO 10375
10375 REM CENTIMES
10380 Z2$ = MID$(Z2$,Z3 + 2,2)
10385 Z2 = VAL(Z2$); GOSUB 10425
10390 IF Z2 > 1 THEN PRINT " ";ZW$ + "S
";
10395 IF Z2 = 1 THEN PRINT " ";ZW$;
10400 PRINT : PRINT : PRINT "EN VOTRE AI
MABLE REGLEMENT " : PRINT ZK$
10405 PRINT : PRINT : FLASH : PRINT "NOU
VELLE VALEUR..(Q/N)..?"; GET Z$
10410 IF Z$ = "0" THEN HOME : GOTO 9000
10415 IF Z$ = "N" THEN GOTO 10850
10420 IF Z$ < > "N" THEN GOTO 10405
10425 REM MODULE PRINCIPAL-GROUPE DE
3 CHIFFRES-
10430 IF Z2 = 0 THEN GOTO 10845
10435 IF Z2 < = 9 THEN ZX$ = " " + ST
R$(Z2); GOTO 10450
10440 IF Z2 < = 99 THEN ZX$ = " " + ST
R$(Z2); GOTO 10450
10445 IF Z2 < = 999 THEN ZX$ = STR$(Z
2); GOTO 10450
10450 REM
10455 ZA$ = LEFT$(ZX$,1)
10460 ZV$ = ZA$; GOSUB 10510
10465 ZB$ = ZV$
10470 ZB$ = MID$(ZX$,2,1)
10475 ZV$ = ZB$; GOSUB 10570
10480 ZB$ = ZV$
10485 ZC$ = MID$(ZX$,3,1)
10490 ZV$ = ZC$; GOSUB 10510
10495 ZC$ = ZV$
10500 GOSUB 10635
10505 RETURN
10510 REM CALCUL DES UNITES ET DES CE
NTAINES
10515 IF ZV$ = "1" THEN ZV$ = "UN"
10520 IF ZV$ = "2" THEN ZV$ = "DEUX"
10525 IF ZV$ = "3" THEN ZV$ = "TROIS"
10530 IF ZV$ = "4" THEN ZV$ = "QUATRE"
10535 IF ZV$ = "5" THEN ZV$ = "CINQ"
10540 IF ZV$ = "6" THEN ZV$ = "SIX"
10545 IF ZV$ = "7" THEN ZV$ = "SEPT"
10550 IF ZV$ = "8" THEN ZV$ = "HUIT"
10555 IF ZV$ = "9" THEN ZV$ = "NEUF"
10560 IF ZV$ = "0" THEN ZV$ = " "
10565 RETURN
10765 REM *****
10770 IF ZC$ = "UN" THEN ZB$ = "QUATRE-V
INGT";ZC$ = "ONZE"; RETURN
10775 IF ZC$ = "DEUX" THEN ZB$ = "QUATRE
-VINGT";ZC$ = "DOUZE"; RETURN
10780 IF ZC$ = "TROIS" THEN ZB$ = "QUATR
E-VINGT";ZC$ = "TREIZE"; RETURN
10785 IF ZC$ = "QUATRE" THEN ZB$ = "QUAT
RE-VINGT";ZC$ = "QUATORZE"; RETURN
10790 IF ZC$ = "CINQ" THEN ZB$ = "QUATRE
-VINGT";ZC$ = "QUINZE"; RETURN
10795 IF ZC$ = "SIX" THEN ZB$ = "QUATRE-
VINGT";ZC$ = "SEIZE"; RETURN
10800 RETURN
10805 REM *****
10810 PRINT ZA$;" ";ZT$;; GOTO 10845
10815 PRINT ZA$;" ";ZT$;" ";ZB$;; GOTO 1
0845
10820 IF ZA$ = " " AND ZB$ = " " THEN P
RINT ZC$;; GOTO 10845
10825 IF ZA$ = " " THEN PRINT ZB$;" ";Z

```

```

C$;; GOTO 10845
10830 IF ZB$ = " " AND ZC$ = " " THEN P
RINT ZA$;" ";ZT$;; GOTO 10845
10835 IF ZC$ = " " THEN PRINT ZA$;" ";Z
T$;" ";ZB$;; GOTO 10845
10840 PRINT ZA$;" ";ZT$;" ";ZB$;" ";ZC$;
10845 RETURN
10850 HOME : HTAB 17; VTAB 10; FLASH : P
RINT "AU REVOIR": NORMAL : END
10570 REM CALCUL DES DIZAINES
10575 IF ZV$ = "1" THEN ZV$ = "DIX"
10580 IF ZV$ = "2" THEN ZV$ = "VINGT"
10585 IF ZV$ = "3" THEN ZV$ = "TRENTA"
10590 IF ZV$ = "4" THEN ZV$ = "QUARANTE"
10595 IF ZV$ = "5" THEN ZV$ = "CINQUANTE
"
10600 IF ZV$ = "6" THEN ZV$ = "SOIXANTE"
10605 IF ZV$ = "7" THEN ZV$ = "SOIXANTE-
DIX"
10610 IF ZV$ = "8" THEN ZV$ = "QUATRE-VI
NGT"
10615 IF ZV$ = "9" THEN ZV$ = "QUATRE-VI
NGT-DIX"
10620 IF ZV$ = "0" THEN ZV$ = " "
10625 RETURN
10630 REM *****
10635 IF ZA$ = "UN" THEN ZA$ = "":ZT$ =
"CENT"; GOTO 10655
10640 IF ZA$ = " " THEN ZT$ = "": GOTO 1
0655
10645 IF ZA$ < > " " THEN ZT$ = "CENT"
10650 IF ZB$ = " " AND ZC$ = " " THEN ZT
$ = "CENTS"; GOTO 10810
10655 REM *****
10660 IF ZB$ = "DIX" THEN GOSUB 10690:
GOTO 10820
10665 IF ZB$ = "SOIXANTE-DIX" THEN GOSU
B 10730; GOTO 10820
10670 IF ZB$ = "QUATRE-VINGT" AND ZC$ =
" " THEN ZB$ = "QUATRE-VINGTS"; GO
TO 10815
10675 IF ZB$ = "QUATRE-VINGT-DIX" THEN
GOSUB 10770; GOTO 10820
10676 IF ZB$ < > " " AND ZC$ = "UN" THE
N ZC$ = "ET UN"
10680 GOTO 10820
10685 REM *****
10690 IF ZC$ = "UN" THEN ZB$ = "":ZC$ =
"ONZE"; RETURN
10695 IF ZC$ = "DEUX" THEN ZB$ = "":ZC$
= "DOUZE"; RETURN
10700 IF ZC$ = "TROIS" THEN ZB$ = "":ZC$
= "TREIZE"; RETURN
10705 IF ZC$ = "QUATRE" THEN ZB$ = "":ZC
$ = "QUATORZE"; RETURN
10710 IF ZC$ = "CINQ" THEN ZB$ = "":ZC$
= "QUINZE"; RETURN
10715 IF ZC$ = "SIX" THEN ZB$ = "":ZC$ =
"SEIZE"; RETURN
10720 RETURN
10725 REM *****
10730 IF ZC$ = "UN" THEN ZB$ = "SOIXANTE
":ZC$ = "ET ONZE"; RETURN
10735 IF ZC$ = "DEUX" THEN ZB$ = "SOIXAN
TE";ZC$ = "DOUZE"; RETURN
10740 IF ZC$ = "TROIS" THEN ZB$ = "SOIXA
NTE";ZC$ = "TREIZE"; RETURN
10745 IF ZC$ = "QUATRE" THEN ZB$ = "SOIX
ANTE";ZC$ = "QUATORZE"; RETURN
10750 IF ZC$ = "CINQ" THEN ZB$ = "SOIXAN
TE";ZC$ = "QUINZE"; RETURN
10755 IF ZC$ = "SIX" THEN ZB$ = "SOIXANT
E";ZC$ = "SEIZE"; RETURN
10760 RETURN

```

"The rumour mill" (le moulin à rumeurs), c'est ainsi qu'on commence à désigner Apple dans le petit monde des fanatiques américains de la micro. Il faut dire, qu'à Cupertino, le siège d'Apple n'a jamais autant résonné de ces fameuses rumeurs. A la fin de l'année 1984, John Sculley annonçait la sortie d'un Apple //x (disquettes de 3 pouces 1/2, nouveau processeur, et une capacité mémoire importante). Au début de l'année 1985, Steve Wozniak, le créateur de l'Apple // démentait le PDG, "ce projet est mort" disait-il lors d'interview donnée en Nouvelle-Zélande, et reprise dans le magazine britannique Apple User. Cette déclaration a causé quelques bruits à Cupertino; et fin janvier Wozniak quittait Apple, pour se lancer dans la vidéo. Un départ qui n'a pas manqué d'inquiéter tous les développeurs de programmes pour l'Apple //. Cet ordinateur serait-il bientôt délaissé par Apple ?

Difficile à imaginer, quand on sait que le // s'est déjà vendu à plus de deux millions et demi d'exemplaires. Dans sa dernière interview, Steve Wozniak annonçait quand même plusieurs choses intéressantes.

Le rôle assigné au //x serait en fait rempli par l'Apple //, et ce grâce à une carte. Une de ces cartes d'extension qui font les beaux jours du //. Celle-ci contiendrait le nouveau biprocesseur 65816. Quant à sa disponibilité, le problème semble être de surmonter les défauts de jeunesse du 65816 qui le rendent actuellement peu fiable, donc très onéreux. Mais, tous les processeurs ont connu ces défauts là...

D'autre part, une augmentation de la mémoire qui porterait la capacité de l'Apple // à 256 Ko. En attendant, deux innovations semblent être bien plus proches.

Il s'agit d'abord d'une modification du clavier de l'Apple //e, qui deviendrait comparable à celui du //. Avec notamment, la possibilité d'obtenir des chiffres, dès que la touche de verrouillage majuscule est enfoncée. La seconde nouveauté concerne les disquettes. L'Apple // abandonnerait (progressivement) le format 5 pouces 1/4, pour adopter celui du Macintosh (3 pouces 1/2). On parle pour ces disquettes d'une capacité pouvant atteindre 1,6 mégaoctets. Il s'agirait alors vraisemblablement de lecteurs de disquettes double face, double densité. Eh bien, même si le //e est toujours très demandé, le //c n'est pas pour autant délaissé.

Du nouveau pour l'Apple //c

Qui l'eu cru, l'Apple //c a aujourd'hui un écran plat à cristaux liquides ! Il

Micro-informations

Jean-Michel Gourévitch

sera disponible dès le mois d'avril 1985 pour un prix d'environ 6000 Frs HT.

Il ne fait qu'accroître la transportabilité de la machine. En effet, il pèse 1,1 kg et mesure 13,8 cm de large sur 29 cm de long par 4 cm d'épaisseur. Il affiche 24 lignes sur 80 colonnes (en mode texte) et 560 points sur 192 lignes (en mode graphique).

La technologie à cristaux liquides permet une consommation réduite et une alimentation par le connecteur DB15 du //c.

Pour plus de confort d'utilisation et un angle de vision optimum, l'écran plat de l'Apple //c est inclinable. Il doit être utilisé dans un environnement lumineux d'intensité suffisante.

Les précautions d'utilisation :

Il est conseillé d'éviter de l'exposer aux rayons solaires, de le démonter sous peine d'un endommagement qui serait peut-être irrémédiable.

L'Apple //c connaît le DOS 3.3, le ProDOS, et pourra disposer de CP/M. Une carte est prévue à cet effet et permettra de développer et d'exécuter tout programme standard pour Apple //e sous CP/M, sans modification préalable, et ce 30% plus vite que la plupart des cartes CP/M pour Apple //e.

Le module est complètement transparent pour l'utilisateur. Cette carte comporte un coprocesseur Z80 et permet donc l'exécution directe de code Z80 ou 8080. De ce fait, elle est compatible à 100% avec CP/M version 2.23.

Les programmes

En France

Contrôle X, qui a réussi l'exploit de vendre aux américains de Hayden son programme intégré CX MacBase pour le Macintosh, met la dernière main à un programme pour le // dont le nom de code est encore Pro-Base. Il s'agirait d'une gestion de fichiers doublée d'un tableur intégré, permettant le transfert des données des fiches vers le tableur. Utilisation de la souris, de 128 Ko, d'un écran totalement graphique, on en entendra encore parler. Les aficionados de CX Base 200, qui réclamaient 80 colonnes, doivent déjà se réjouir.

Le nouveau traitement de texte **Gri-bouille** est le premier, à notre connaissance pour Apple //, à couper automatiquement les mots, conformément à la grammaire française. Il a été développé, pour la société Berlingot, par Madeleine Hodé. Aide mémoire en 10 tableaux intégrés au

programme, glossaire de 1000 à 22000 caractères, dispositif évitant les sauts de lignes ou de pages inopportuns, impression de graphiques intégrés au texte, possibilité de définir ses propres caractères. On nous promet la facilité d'utilisation et la rapidité pour un prix de 1700 Frs. C'est bien alléchant.

De l'autre côté de l'Atlantique

Les programmeurs s'assoupiraient-ils ? Il serait temps qu'un nouvel Apple les réveille. On peut remarquer un programme de jardinage, **Ortho**, qui aide à organiser un jardin.

Un programme d'apprentissage de la musique que l'on dit aisé : **Magic piano** d'EduSoft pour environ 50 dollars.

Deux nouveaux programmes de Beagle Bros. L'un (I.O. Silver) est un jeu de stratégie : il faut construire un super ordinateur en évitant le gang des "bugs" (30 dollars); l'autre est un utilitaire D.Code permettant de debugger et de gagner de la place dans les programmes (40 dollars).

Pour exploiter au mieux la tablette Koala Pad, Koala propose des outils pour programmeurs, et un **Graphic Exhibitor** permettant de présenter les dessins réalisés dans une séquence (40 dollars chacun).

Fabriquer des cartes d'identité à l'américaine, c'est possible avec le **ID Maker** de Scarlet Software. On peut même y incorporer son propre logo, choisir les caractères, il en coûtera 50 dollars.

Quelques accessoires

Tout d'abord un petit gadget qui convertit la sortie lecteur de l'Apple //c aux normes des anciens lecteurs. Ainsi, on peut utiliser n'importe quel lecteur prévu pour les II+ et //e sur le //c, grâce à **Adapt a disc** de Computer Accents (30 dollars).

Ensuite, un capot de protection pour le clavier du //e, permettant de le protéger contre divers liquides, il s'agit du Hard Cover de Diversified Manufacturing (disponible aussi pour le Macintosh).

Et surtout des lecteurs de disquettes 3 pouces 1/2. Ils existent déjà ? Eh oui, c'est l'oeuvre de Haba. Ce lecteur est livré avec des programmes de communication, des formats prédéfinis pour Appleworks; et un utilitaire permettant de transférer AppleWorks sur une disquette 3 pouces 1/2, pour un prix d'environ 450 dollars.

Un nouveau confrère

RS 232 est l'hebdomadaire informatique consacré aux petites annonces (gratuites). Toutefois, on y trouve de nombreuses rubriques (clubs informatiques, contact, jeux,...). Il est né, depuis plus de six mois, pour faciliter l'expression et les contacts entre particuliers. Ce magazine est indépendant et compte le rester. Distribué dans plus de 60 grandes villes, vous le trouverez dans les kiosques au prix de 3F. Un prix dérisoire qui lui permettra de vous aider et de se développer.

Vive le Mac

Le Macintosh devient l'enfant chéri des programmeurs. Même si aux Etats-Unis, les utilisateurs sont furieux d'avoir dû déboursier 1000 dollars pour se payer le kit d'extension 512 Ko (là-bas, compte tenu des baisses de prix, les nouveaux utilisateurs achetant un 512 Ko le paient moins cher que le 128 Ko plus la transformation). Leurs vociférations couvrent même parfois les bruyantes campagnes publicitaires d'Apple...

Voici donc un Mac qui est allé se rhabiller. Transformé en portable, il s'appelle maintenant le Mac Colby, et peut intégrer toute une série de périphériques : deuxième lecteur, modem, disque dur, lecteur de code barre,...

Voici surtout un disque dur interne révolutionnaire, il s'intègre à l'intérieur du coffret de Macintosh. C'est l'hyperdrive de General Computer. Il est d'une capacité de 10 mégaoctets, sur un disque dur de 3 pouces 1/2. Ainsi, les temps d'accès seraient 20 fois plus rapides qu'avec une disquette et 7 fois plus rapides qu'avec un disque dur externe. Son prix est de \$2800 avec l'extension 512 Ko et \$2200 sans extension.

Et puis, pour ceux qui ont une bonne vue, une rallonge (4 mètres) pour le câble du clavier. Elle est vendue par Yas et coûte 10 dollars.

Des produits comme s'il en pleuvait

Tout d'abord, un processeur d'équations, c'est le célèbre **TK!Solver**, mis au point par les inventeurs de Visicalc. Il remodèle des équations jusqu'à trouver les inconnues. Avec TK!Solver, les inconnues, connaît plus ! Ce programme existait déjà pour d'autres ordinateurs, il prend sa pleine valeur sur Mac. D'une étonnante rapidité de calculs, il va faire le désespoir (mais aussi le bonheur) des professeurs de mathématiques. Ce programme permettant de résoudre quasi instantanément et de façon magique des problèmes (pour les-

quels il fallait des journées entières de travail) est promis à un bel avenir. Il pourrait même en inciter certains à acheter un Macintosh. Il est vendu 3500 Frs par Software Arts.

MacFortran est une implémentation sur Macintosh du Fortran ANSI 77. Il permet de développer des applications utilisant le système de gestion de bureau. Il contient un debugger symbolique au niveau source et une routine VIRTUAL permettant aux tableaux plus grands que la mémoire d'être gérés en mémoire virtuelle sur disque.

Il autorise la programmation structurée avec sous-programmes, fonctions, procédures,... (très proche du Pascal). Son prix est d'environ 5800 Frs HT.

MacPlot est un logiciel conçu pour travailler avec MacDraw, mais accepte les graphiques de Chart, MacProject, CX MacBase,... Il analyse l'image et la trace avec plusieurs couleurs et/ou épaisseurs de traits. L'échelle du document original est respectée mais peut être modifiée, la vitesse est paramétrable selon le type de support (papier, film,...). Il se connecte sur le port modem du Macintosh. Les deux logiciels cités ci-dessus sont distribués par Alpha Systèmes.

LaserWriter est l'imprimante à laser haute résolution qu'annonce Apple. Elle est très différente des autres imprimantes à laser. En effet, elle se compose de trois éléments principaux :

- un ordinateur équipé d'un processeur MC68000 avec un mégaoctet et demi de mémoire centre et 512 Ko de ROM (c'est le plus gros ordinateur qu'Apple ait jamais construit).
- l'imprimante à laser Canon LB-CX210
- le langage de télécomposition PostScript.

LaserWriter sera disponible en France en Juin 1985 et son prix sera d'environ 70 000 F.

Appletalk est un réseau personnel, d'une grande simplicité d'installation et d'utilisation. Il permet d'interconnecter des Macintosh (32 au maximum sur une distance de 300 mètres), par le biais de simples câbles bi-fils, afin de faciliter les échanges d'informations et les transferts de documents. Il offre la possibilité aux utilisateurs de partager des ressources communes, tel un disque de grande capacité ou l'imprimante LaserWriter. Il est prévu pour communiquer avec d'autres réseaux ou ordinateurs centraux. Il est entièrement géré par l'intelligence de chaque élément. Ceux-ci pouvant être des Macintosh 128K - 512K - XL (Lisa), des périphériques...

Appletalk sera disponible en France vers le mois de Juin pour un prix inférieur à 500 Frs par connexion.

MacBooster est un nouveau logiciel qui apporte un plus quant au confort d'utilisation du Mac 512 Ko. Il permet de diminuer les temps d'accès aux documents les plus fréquemment employés. Il offre à l'utilisateur la possibilité de choisir la taille mémoire allouée et les lecteurs de disquettes à desservir. MacBooster reste en mémoire; ainsi, les portions de fichiers les plus référencées sont recopiées en mémoire de façon à obtenir un accès presque immédiat.

MacBooster est compatible avec toutes les applications qui accèdent aux disquettes au moyen du système de fichiers standard du Macintosh. Il fonctionne selon les principes des anté-mémoires et des systèmes de migration de fichiers. L'utilisation de MacBooster ne modifie pas le contenu de la disquette. Ce logiciel a été développé par Jean-Luc Delatre.

MacPublisher, distribué par Sonotec permet de réaliser une lettre ou un journal, de façon spectaculaire.

Un nouveau traitement de texte britannique **Macauthor** d'Icon Technology permet de choisir facilement le style d'écriture d'un document, de réaliser des documents de 698 pages, d'insérer dans le texte des caractères inhabituels, musicaux,...

A propos de traitement de texte, et en attendant Word (disponible au printemps), MacWrite version 3.2 devrait être bientôt disponible. A noter, la possibilité de stocker et surtout d'imprimer des documents de 65 pages. Le numéro de la page en cours indiqué sur l'ascenseur, un choix de menu permettant d'aller directement à une page donnée, la possibilité de centrer ou de justifier individuellement un élément du texte,...

La version améliorée de **Think Tank** pour 512 Ko, est là. Elle possède, enfin, toutes les qualités de la version de l'Apple II, et notamment les paragraphes, la possibilité d'obtenir des documents de 35 pages, d'y incorporer des graphiques ou des dessins,... Une version complètement différente de celle pour le 128 Ko.

Désormais le Lisa s'appellera Macintosh XL. Ce nouveau nom symbolise la parfaite adaptation de Lisa à l'environnement de Macintosh. Il utilise tous les logiciels du Macintosh et est totalement compatible avec les nouveaux produits annoncés par Apple (Appletalk, LaserWriter...).

Et, puis, pour l'automne un match au sommet s'annonce passionnant. L'équipe dirigée par Paul Lutus, qui a créé Appleworks est passée chez Microsoft et y prépare un programme

intégré pour le Macintosh, qui pourrait commencer le célèbre Jazz de Lotus, déjà créateur de 1, 2, 3 pour l'IBM PC. L'année Mac s'annonce décidément bien.

Les livres

Les produits d'Enseignement Assisté par Ordinateur nous envahissent. Chez Magnard :

– Sésame (230 F) est destiné à apprendre les rudiments du Basic aux enfants.

– Le Basic (990 F), Racines verbales (490 F), Homophones (490 F), Logique et Math (560 F). Ces produits, d'origine canadienne, nous semblent dans l'ensemble bien faits, si ce n'est pour le dernier qui aurait mérité d'être traduit du français canadien en franco-français.

– Macintosh, modes d'emploi, de William Skyvington, Seuil / La Recherche, 150 F. Traduction. C'est le énième livre d'initiation au Macintosh, à croire que les pauvres acheteurs de ce matériel sont totalement incapables de lire les notices pourtant bien faites d'Apple. A part les classiques MacWrite, Paint, Multiplan et Chart, on y trouve une présentation de MacDraw et MacProject. Un petit plus par rapport aux livres qui n'en parlent pas.

– Apple, modes d'emploi, de William Skyvington, Seuil / La Recherche,

110 F. Traduction. On retrouve le même principe de description de logiciels standards mais, ce qui est moins fréquent, appliqué à l'Apple //. Il s'agit principalement de modes d'emploi fort résumés (mais sans fiche synthétique ni glossaire) d'Apple Writer, Quick File, Multiplan et Business Graphics, avec une touche de Calvados.

– 250 questions sur la micro-informatique de Ilya Virgatchik, Marabout service. Les questions sont classées par thèmes : généralités, achat et maintenance, questions techniques, systèmes d'exploitation, périphériques, langages, programmation, programmes d'application, jeux / enseignement. Heureusement, un index de quatre pages permet un peu de s'y retrouver. J'aurais préféré un sommaire détaillé avec la liste des questions, afin de pouvoir mieux organiser la consultation et la lecture. Tel qu'il est, ce livre est malheureusement peu pratique à lire pour répondre aux questions qu'on se pose, puisqu'on ne peut les connaître qu'en le lisant de bout en bout ...

Adresses

Contrôle X - Tour Maine Montparnasse - 33 ave du Maine - 75755 Paris Cedex 15

Berlingot - 18 rue Elile Duclaux - 75015 Paris

Ortho Information Services - 575 Market St - San Francisco - CA 94105

EduSoft - PO Box 2560Q - Berkeley - CA 94702

Beagle Bros - 3990 Old Town Ave - suite 1026 - San Diego CA 92110

Koala - 3100 Patrick Henry Drive - Santa Clara CA 95052

Scarlet Software - PO Box 11166 Milwaukee - WI 53211

Computer Accents - PO Box 5307 Kingwood - TX 77325

Diversified Manufacturings - 4722 8th Street - Wichita KS 67208

Haba Systems - 15154 Stagg Street - Van Nuys CA 91405 1025

Colby Computer - 849 Independence Ave MountainView - CA 94043

General Computer - 2155 First street - Cambridge MA 02142

YAS - 1525 N Elston Chicago - IL 60622

Jean-Luc Delatre - 7, rue de Rambouillet - 78120 Clairefontaine - Tél (3) 484.50.51

Apple Seedrin - ZA de Courta-boeuf - Ave de l'Océanie, BP 131 - 91944 Les Ulis - Tél (6) 928.01.39

Alpha Systèmes - 16 rue de Sausure - 75017 Paris - Tél 763.59.81

RS 232 - 109 rue G. Lauriau - 93100 Montreuil - Tél 859.71.01

Courrier des lecteurs

Olivier Herz

En ce qui concerne l'article sur le "Disque Virtuel 64K", publié dans Pom's 14, je me demande s'il est possible d'utiliser une carte d'extension 128 Ko comme disque virtuel, et de faire tourner un programme comme CX BASE 200 ou PFS.

Mr François Fossat - Immeuble Le Tilleul - 74 bd Loius Roux - 06700 Saint Laurent du Var

Le disque virtuel est fait pour fonctionner avec le DOS 3.3 et les programmes "tournant" sous ce système. Par conséquent, les programmes CX BASE 200 et PFS, ayant leur propre système d'exploitation, ne peuvent utiliser le disque virtuel proposé dans Pom's.

Est-il possible de copier des disquettes formatées sous Pascal UCSD ou CP/M à l'aide du programme

COPYA se trouvant sur la disquette Master DOS 3.3 ?

Mr Stéphane Lévy - 142 rue Gambetta - 45120 Chalette

Le programme COPYA vous permet, effectivement, de copier toutes disquettes formatées sous Pascal UCSD, CP/M, ProDOS...

A propos de "Disk Manager 2", j'aimerais savoir quel est le niveau de son utilitaire "Ultra Copie" par rapport à Locksmith 5.0 ? Est-il en français ?

Mr Gilbert Roche - Quartier Jumel, Saint Julien du Serre - 07200 Aubenas

Ultra Copie et Locksmith ne sont pas comparables.

Locksmith 5.0 permet, entre autres, de copier certaines disquettes protégées, de copier très rapidement des

disquettes formatées en 16 ou 8 secteurs (DOS 3.3, ProDOS, CP/M, Pascal).

Ultra Copie, quant à lui, permet de copier des disquettes formatées en 16 secteurs : soit normalement, soit sélectivement en ne copiant que les secteurs qui sont effectivement utilisés. Ce procédé est d'autant plus rapide que la disquette originale est moins remplie.

Le Disk Manager est, à notre connaissance, un utilitaire sans précédent. Il permet d'accéder directement aux secteurs de la disquette pour écrire des programmes en Basic agissant sur la disquette (gestion de catalogues, utilitaires...); et ce, depuis le Basic. De telles choses n'étaient réalisables qu'avec des routines en assembleur.

Le Disk Manager est l'oeuvre d'un lecteur français de Pom's, il est donc en français, ainsi que sa documentation.

Par quel biais peut-on réaliser le déplacement du DOS vers la carte langage 16 Ko ?

Mr Pradayrol - 21 rue Guenegaud - 75006 Paris

De nombreux logiciels, le plus souvent vendus avec une carte d'extension 64 ou 128 Ko, permettent de placer le DOS sur la carte langage. Le programme MEMORY MASTER, par exemple, qui est fourni avec la carte LEGEND (testée dans Pom's 5). DIVERSI DOS (DOS 3.3 amélioré) possède un utilitaire, DDMOVER, qui le reloge sur la carte langage. Et enfin, plusieurs revues ont publié des programmes déplaçant le DOS sur la carte 16 Ko. La plus célèbre est CALL A.P.P.L.E (CALL A.P.P.L.E in Depth : All about DOS), avec son programme DOS MOVER.

Je souhaiterais avoir quelques éclaircissements sur les entrées / sorties de l'Apple //e avec la carte 80 colonnes. J'essaie vainement de faire une copie d'écran 80 colonnes sur une imprimante DMP.

Maurice Gaston - Boussieyral - Saint André Allas - 24200 Sarlat

La méthode suivante permet de réaliser une copie d'écran en 80 colonnes. Toutefois la place mémoire nécessaire est importante.

1-Transférer la page texte dans la mémoire à l'endroit souhaité (\$2000 - \$23FF par exemple). Pour ce faire, utiliser la routine MOVE du moniteur (\$FE2C), qui déplace la zone délimitée par \$3C-\$3D et \$3E-\$3F vers la zone commençant en \$42-\$43. Ceci permet d'éviter d'éventuels problèmes liés au DOS lors du transfert vers l'imprimante.

2-Transférer la page texte de la mémoire auxiliaire dans la mémoire principale à l'endroit souhaité (\$2400 - \$27FF par exemple). Pour ce faire, on utilise la routine AUXMOVE du moniteur (\$C311), qui déplace la zone délimitée par \$3C-\$3D et \$3E-\$3F vers la zone commençant en \$42-\$43. De plus, il faut forcer le bit de retenue du 6502 à 0 (CLC) pour indiquer que le transfert se fait de la mémoire auxiliaire vers la mémoire principale.

3-Il ne reste plus qu'à tout envoyer à l'imprimante en faisant attention à la disposition des octets de la page texte, et en sachant qu'un octet sur deux vient de la mémoire principale, tandis que l'autre provient de la mémoire auxiliaire.

Vous pouvez vous inspirer de l'article de Christian Guerin, publié dans

Pom's 3, en ajoutant respectivement \$1C00 et \$2000 aux adresses de début de ligne de la page provenant de la mémoire principale (qui commence en \$2000) et de celles provenant de la mémoire secondaire (qui commence en \$2400)

Une deuxième solution, plus économique en temps et en place mémoire, serait d'utiliser le Basicium. Il vous permettrait de réaliser une copie d'écran texte grâce à la commande]H.

DIF.OBJ, paru dans Pom's 11: Pour éviter un OUT OF MEMORY avec des programmes Basic comprenant plus de 160 lignes, il faut taper, depuis le moniteur :

93D1 : B8 B4 B4

93D4 : 68 68

93D6 : 4C 54 93

Ainsi, la pile sera mise à jour et les appels à LIST ne poseront plus de problèmes

Yvan Koenig - Céramiques Gerbino - rue du Stade - 06220 Vallauris

J'ai lu dans Pom's 15 que l'astrologie intéressait un de vos lecteurs, Joël Moreau, et je suis ravi de pouvoir lui répondre. J'ai développé un logiciel sur le sujet; il s'agit de "Astrologie Interprétation Rationnelle". Cette version se compose d'une disquette contenant, sur une face les fichiers des diverses influences astrologiques (environ 100 Ko) et des modules de calcul d'interprétation sur l'autre. Ceux-ci sont principalement écrits en Basic compilé (très peu d'assembleur). Le prix est très raisonnable. Pour plus de renseignements n'hésitez pas à me contacter

Paul Franceschi - 5 rue de la Forêt Sainte - 67500 Haguenau

Grâce à un programme paru dans Pom's 7, j'ai pu réaliser un programme de calculs financiers relativement complet : taux d'intérêts, rendement d'investissement à flux de montants non égaux et non réguliers (portefeuille de boursier, par exemple), achat d'obligations, effet de déduction fiscale... Compte tenu de la spécificité du problème, je ne m'étendrais pas. Que les lecteurs intéressés me contactent.

Jean Michel Quetin - 8 allée de la Boissière - 92350 Le Plessis Robinson

Vends un Apple II, extension 16K, deux drives, un moniteur noir et blanc, joystick et paddles au prix de

10000 Frs et une imprimante OKI 84. 200 cps, 132 colonnes au prix de 7000 Frs (valeur 11000 Frs).

Philippe Zitoun - 11 allée de la Pavillone - La Celle St Cloud

Ayant programmé un jeu d'aventures, j'ai eu la folle idée de le faire éditer (à 16 ans, on a encore des illusions). D'autres l'avaient fait avant moi... Je prenais donc de nombreux contacts. Toujours la même réponse : envoyez votre disquette. Ediciel, qui favorise la création française en traduisant les plus mauvais programmes américains, m'a même répondu que les jeux d'aventures ne l'intéressaient pas... Et puis un beau jour, miracle, arrive chez moi une lettre, de l'agence Octet, m'invitant aux "Journées du Logiciel", ministère de la culture à l'appui. Emportant mes rêves de célébrité et mes disquettes, je m'y rendais. A l'entrée, on me remettait un superbe badge avec "François Coulon - Créateur". Je commençais à m'y croire...

J'ai erré parmi les stands, hélas rien d'intéressant pour moi, sauf peut-être une proposition d'Exelvision m'offrant la possibilité d'adapter mon jeu à leur matériel. Je découvrais, avec l'arrivée de certains journalistes, un des buts de l'organisation : s'écouter parler entre gens biens, l'informatique est à la mode, n'est-ce-pas ?

Après la remise des prix, nous partions, mon père et moi. J'avais perdu une demi-journée et ramassé des adresses que je possédais auparavant.

J'ai donc décidé de ne jamais faire éditer quoi que ce soit, et de prouver que je n'ai pas besoin d'éditeur pour faire connaître mes produits. C'est ce pari que j'aimerais pouvoir tenir, grâce à vous. Vous Pom's, en publiant cette lettre, vous montrerez aux éditeurs qu'ils ne sont pas indispensables. Vous lecteurs, en achetant mon programme, vous prouverez que je n'ai pas besoin d'eux pour gagner de l'argent avec le fruit de mon travail. Je n'ai pas besoin d'eux, mais j'ai besoin de vous.

Pour tout autre renseignement sur ce jeu ("1928") et sur le prochain n'hésitez pas à me contacter.

Les opinions que j'exprime n'engagent, bien entendu, que moi.

François Coulon - 4 route de Ham - 80190 Nesle

Votre lettre est un témoignage prouvant, une fois de plus, qu'il est difficile de s'imposer comme créateur à 16 ans; d'autant plus que la commercialisation d'un produit (quel qu'il soit) ne peut se faire que dans le cadre d'une structure adaptée.

Vous...

Toujours dans l'optique de satisfaire nos lecteurs, voici un petit questionnaire qui nous permettra de savoir ce que vous pensez de Pom's. Ainsi, pour la réalisation des prochains numéros, nous prendrons en considération toutes les réponses et remarques que vous nous ferez parvenir.

Nous procéderons à un tirage au sort, afin de déterminer les 10 bulletins gagnants. Ceux-ci bénéficieront d'un abonnement avec disquettes. Pour participer à ce tirage, il vous suffit de nous renvoyer ce petit questionnaire avant la fin du mois d'avril.

Ce bulletin pourra, bien sûr, rester anonyme.

Votre âge : moins de 15 ans de 15 à 19 ans
de 20 à 24 ans de 25 à 34 ans
de 35 à 45 ans plus de 45 ans

Sexe : féminin masculin

Vous utilisez : un Apple II+ un Apple //e
un Apple //c un Apple ///
un Mac 128 K un Mac 512 K
un Lisa

Vous l'utilisez : chez vous à votre travail
autre

Date d'achat

Vous disposez de :
..... lecteurs

une imprimante marque modèle

une carte 80 c. marque modèle

une carte couleur une carte CP/M

une table traçante un disque dur

autre

Quel matériel allez-vous acheter dans les 6 mois :
.....

Vous utilisez votre matériel pour :
votre travail une gestion personnelle
jouer le plaisir de programmer
autre

En programmation, votre niveau :
débutant intermédiaire
amateur averti haut niveau

Quels langages pratiquez-vous aisément :
Basic Pascal Forth Logo
Assembleur 6502 65C02 68000
Lisp autres

Vous passez chaque semaine devant l'écran :
moins de 2 heures entre 3 et 8 heures
entre 9 et 16 heures plus de 16 heures

... et pom's

Combien de numéros de Pom's avez vous lu

Comment avez-vous connu Pom's :
publicité parue dans une revue kiosque
Sicob, Micro-Expo, Apple Expo boutique
par un autre amateur

Combien de personnes lisent votre Pom's

votre opinion, de 5 (très bon) à 0 (très mauvais), sur

le niveau général de la revue

sa présentation

l'intérêt des programmes

l'originalité des programmes

la clarté des articles joints

l'intérêt pédagogique de ces articles

les bancs d'essais

les micro-informations

les disquettes d'accompagnement

Ce que vous préférez dans chaque numéro :
.....
.....

Ce que vous aimez le moins :
.....
.....

Ce que vous souhaitez voir prochainement dans Pom's :
.....
.....

Ce qu'il faut développer ou changer :
.....
.....

Votre opinion sur le "Cahier Mac" :
.....
.....

Vos remarques particulières :
.....
.....

Quelles autres revues informatiques lisez-vous ?
.....
.....

Nom :

Prénom :

Adresse :

Ville :

Code Postal :

pom's

DISQUETTES (sauf précision, toutes les disquettes fonctionnent sous DOS 3.3 sur Apple II + , //e ou //c)

HAIFA	(cf. Pom's n° 5)	à	55,00 F
H-BASIC	(cf. Pom's n° 8)	à	150,00 F
MUSIC	(cf. Pom's n° 10)	à	80,00 F
DISK-MANAGER	(cf. Pom's n° 11)	à	450,00 F
DBSTAG (CP/M)	(cf. Pom's n° 11)	à	450,00 F
JEUX A	(cf. Pom's n° 12)	à	80,00 F
JEUX B	(cf. Pom's n° 12)	à	80,00 F
BASICUM	(cf. Pom's n° 13)	à	150,00 F
E.P.E.	(cf. Pom's n° 15)	à	150,00 F
MACINTOSH	(cf. Pom's n° 14-15-16)	à	150,00 F
MACINTOSH	(cf. Pom's n° 17)	à	80,00 F
PASCAL	(cf. Pom's n° 15)	à	80,00 F
DEMO MAX THE GOLBE TROTTER ..	(cf. Editorial)	à	55,00 F
MAX (moniteur étendu)	(voir p. 64)	à	150,00 F

RECUEILS

N° 1, recueil des revues 1 à 4	à	140,00 F
Disquettes d'accompagnement des numéros 1 à 4	à	150,00 F
N° 2, recueil des revues 5 à 8	à	140,00 F
Disquettes d'accompagnement des numéros 5 à 8	à	190,00 F

ANCIENS NUMEROS

REVUES	<input type="checkbox"/> 4	<input type="checkbox"/> 7	<input type="checkbox"/> 8	à	35,00 F
REVUES	<input type="checkbox"/> 9	<input type="checkbox"/> 10	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	
	<input type="checkbox"/> 14	<input type="checkbox"/> 15	<input type="checkbox"/> 16	<input type="checkbox"/> 17	à	40,00 F
DISQUETTES	<input type="checkbox"/> 1	<input type="checkbox"/> 2	<input type="checkbox"/> 3	<input type="checkbox"/> 4	<input type="checkbox"/> 5	
	<input type="checkbox"/> 6	<input type="checkbox"/> 7	<input type="checkbox"/> 8	<input type="checkbox"/> 9	<input type="checkbox"/> 10	
	<input type="checkbox"/> 11	<input type="checkbox"/> 12	<input type="checkbox"/> 13	<input type="checkbox"/> 14	<input type="checkbox"/> 15	à 55,00 F
	<input type="checkbox"/> 16	<input type="checkbox"/> 17				

ABONNEMENTS

POUR 6 NUMEROS à partir du n°

ABONNEMENT SANS DISQUETTES	à	200,00 F
ABONNEMENT AVEC DISQUETTES (Apple II + , //e, //c)	à	480,00 F

TOTAL TTC

Supplément expédition
par avion à l'étranger

MONTANT
DU REGLEMENT

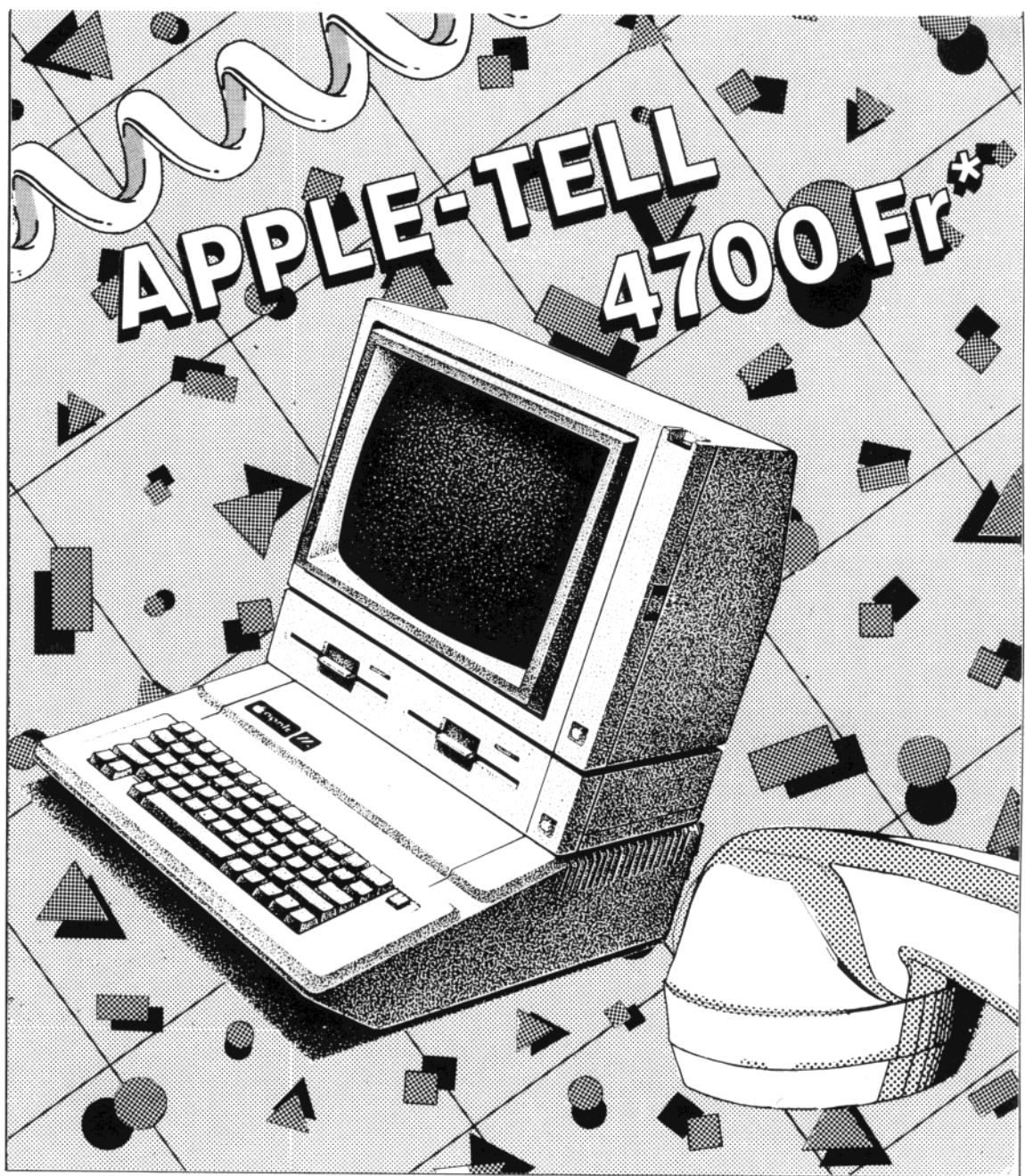
**Ces tarifs comprennent l'envoi postal en France métropolitaine, CEE et Suisse
Supplément avion hors CEE : 10 F par numéro et/ou disquette**

Envoyez ce bon et votre règlement à : (Abonnés : n'oubliez pas de joindre l'étiquette-adresse).

Editions MEV - 64-70, rue des Chantiers - 78000 VERSAILLES

Nom :

Adresse :



*Prix utilisateur H.T. conseillé au 1^{er} février 1985.

C'est pour pouvoir faire face à une demande plus de dix fois supérieure à ce qui avait été prévu qu'Hello a décidé de lancer une nouvelle fabrication de 2 000 cartes Apple-Tell. Avantage de la grande série : de très sensibles réductions de coût sur le prix des composants LSI et VLSI, permettant de faire chuter de plus de 20 % le prix de la carte Apple-Tell.

Le modem vedette de votre Apple, déjà couronné Pomme d'Or 1983, a suscité en 1984 trois nouvelles Pommes d'Or, récompensant trois des (très nombreux) logiciels que ses utilisateurs ont déjà dédiés à Apple-Tell :

- MICRO-KIDS : serveur monovoie pour les établissements d'enseignement.
- VASA : outil de composition de pages vidéotex incluant un serveur arborescent complet.
- TÉLEBASIC : Basic télécom et vidéotex, destiné à la création de serveurs et de terminaux automatiques.

Parmi les autres logiciels créés pour Apple-Tell :

- TELEPOM : enrichissement du Basic (60 instructions nouvelles) permettant de créer soi-même des serveurs (Ascii et vidéotex), des messageries ou toutes applications télématiques.
- ASCII EXPRESS : outil général de télécom destiné à permettre l'impression et le stockage des données reçues, incluant un émulateur universel de terminaux.
- PROTEXT : éditeur-souris, permettant la composition (texte et dessins) de pages Télétel destinées à des serveurs.
- FAKIR : serveur Ascii (messagerie électronique intégrée, incluant panneau d'annonces, annuaire des abonnés, téléchargement, horloge temps réel, mots de passe, console opérateur, etc.).
- DISCOBOLE : copie de disquettes par téléphone (Dos, MemDos, Pascal, CP/M).
- TRANSTEXT : convertisseur d'écrans (Haute Résolution Télétel).
- BASI : module-système permettant la commande du modem depuis le Basic, sous ProDos.
- PROSPECTOR : générateur d'étiquettes-adresses par consultation automatique de l'annuaire électronique selon des choix socioprofessionnels et géographiques.
- CALVA-KIT : consultation

automatisée du centre serveur "Calvados".

- PELAGIE : serveur spécialisé, permettant la consultation d'un Apple à partir d'un réseau de minitel. (Deux versions : Basic/Dos 3.3 et MemDos.)
- NESTOR4 : serveur à QUATRE ACCÈS SIMULTANÉS, disque dur 10 Mo, gestion automatique des touches de fonctions du minitel, arborescence et mots clés intégrés.
- NESTORI : serveur monovoie (même caractéristiques que Nestor4).

... Les serveurs sont servis !

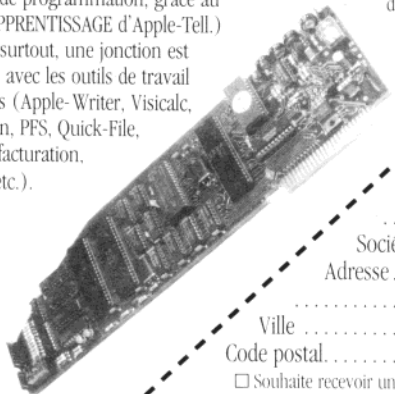
APPLE-TELL COMPREND :

- une carte Modem (avec terminal Ascii intégré) incluant un décodeur Télétel.
- Un logiciel d'émulation de terminal Minitel enrichi des fameuses trois fonctions dont l'absence est une tragédie quotidienne pour les utilisateurs de Télétel : IMPRESSION (sur l'imprimante de l'Apple),

STOCKAGE sur disquettes des pages consultées (formats Télétel ou Ascii), AUTOMATISME : interrogation automatique des serveurs (appel téléphonique, orientation Transpac, identification, choix successifs), enregistrement des données consultées, puis traitement et incorporation des données dans l'application. (Ces procédures d'interrogation sont créées par l'utilisateur, sans aucun langage de programmation, grâce au mode APPRENTISSAGE d'Apple-Tell.) Enfin et surtout, une jonction est possible avec les outils de travail habituels (Apple-Writer, Visicalc, Multiplan, PFS, Quick-File, compta/facturation, fichier, etc.).

CARACTÉRISTIQUES GÉNÉRALES :

- modem 1 200 75 (Ccitt) et 300 full (standards Ccitt et Bell), à numérotation automatique.
- auto-connexion, permettant la création de serveurs 300 ou 1 200 bauds.
- sorties : vidéo composite (N & B) et Pétitel couleurs.
- enfichable dans n importe quel slot libre de votre Apple 2^e ou 2+ (+8 K. une disquette).
- transparence totale vis-à-vis du système.



HELLO Informatique
1, rue de Metz
75010 PARIS
Tél. : (1) 523.30.34
Télex : FLASH 210 500 F

Nom

Société

Adresse

Ville

Code postal Tél.

Souhaite recevoir une documentation sur le système Apple-Tell.

ORDINATEUR INDIVIDUEL

LA RÉFÉRENCE EN MICRO-INFORMATIQUE

Belgique : 186 FB - Suisse : 7,5 FS - Canada : 2,95 \$C

DOSSIER
PLUS DE 200 ORDINATEURS
TABLEAUX COMPARATIFS

**PANORAMA
PRINTEMPS 85**

TOUS LES DERNIERS-NÉS

LAS VEGAS :
LE CHOC AT

NOUVELLE PRESENTATION

A L'ESSAI :
HP 150, ALICE 90



Pragum

23 F EN VENTE CHEZ TOUS
LES MARCHANDS DE JOURNAUX

Mars 1985 n° 68 23 F