ISSN : 0294-6068

pom's

4

Sommaire

NUMÉRO 4 - JUIN 1982 - 35 Francs

58 rue Notre-Dame-de-Lorette - 75009 PARIS
Tél. : 282.19.80 - Télex : 290350 F

JCR Boutique

Essayez! Tous matériels annoncés disponibles à l'essai.

Réfléchissez! Un véritable service avant et après-vente.

- Les conseils du spécialiste pour le matériel et le logiciel.
- Dépannages rapides par notre service après-vente intégré.
- Une garantie 1 an pièces et M.O. sur tous les produits.

Comparez! Des micro-prix sur tous les micros.

Extrait de notre tarif. Prix TTC.

MATERIELS

APPLE II

APPLE II + 48K	[illegible]
DISK II + CONTROLEUR DOS 3.3	[illegible]
DISK II SANS CONTROLEUR	[illegible]
SILDISK 8" 2 X 256K	[illegible]
MODULATEUR N/B EN KIT	[illegible]
CARTE COULEUR SECAM	[illegible]
CARTE COULEUR RVB SONOTEC	[illegible]
CARTE COULEUR RVB	
CHAT MAUVE	1400,00 F
CARTE COULEUR RVB ISTC	1550,00
INTERFACE PARALLELE	[illegible]
INTERFACE SERIE V24 RS 232	[illegible]
CARTE INTEGER	1100,00 F
CARTE APPLESOFT	1100,00 F
CARTE LANGAGE 16K RAM	1150,00 F
SYSTEM PASCAL	2700,00 F
CARTE MICROSOFT Z80 CP/M	2430,00 F
CARTE IEEE 488	2895,00 F
TABLETTE GRAPHIQUE	5500,00 F
CARTE PROTOTYPE	145,00 F
CARTE TIMER CCS	1150,00 F
CARTE 80 COL. BIT3 FULL VIEW	2750,00 F
CARTE 80 COL. SUPER TERM	2950,00 F
CARTE DE COMMUNICATION ILENTYPE II	1150,00 F
PPLE III 128K + SOS	2300,00 F
DISK III ADDITIONNEL	26000,00 F
[illegible]	[illegible]
[illegible]	2500,00 F
[illegible]	26000,00 F
[illegible]	2500,00 F
CARTE PROTO APPLE III	375,00 F
INTERFACE PARALLELE APPLE III	1500,00 F

LOGICIELS

APPLE II

DOS TOOLKIT	440,00 F
APPLE PILOT	950,00 F
VISICALC 16 SECTEURS	1990,00 F
VISITREND + VISIPLOT	2170,00 F
DESKTOP PLAN II	1820,00 F
VISIDEX	1820,00 F
VISIPLOT	1500,00 F
VISITERM	730,00 F
FICHIER PFS	760,00 F
FICHIER PFS REPORT	760,00 F
COMPTA SAAD	3100,00 F
[illegible]	[illegible]
[illegible]	[illegible]
APPLE WRITER II	850,00 F
APPLE PLOT	520,00 F

APPLE III

VISI[illegible]	1700,00 F
APPLE [illegible] III	1500,00 F
ACC[illegible]	1000,00 F
SCRI[illegible]	850,00 F
PASC[illegible]	1700,00 F
BUSI[illegible]C	850,00 F
MAI[illegible]AGER	1000,00 F
COM[illegible]PL/DISQUE DUR	4700,00 F
FICHIER [illegible]	1200,00 F
FICHIER PFS REPORT	1050,00 F

DOCUMENTATION GÉNÉRALE SUR DEMANDE.

Crédit 4-36 mois. Leasing

En raison des fluctuations monétaires ces prix sont susceptibles d'être modifiés sans préavis. Nous consulter pour confirmation.

HARD communication

JCR, l'informatique service compris.

Editorial

Notre environnement d'utilisateurs de l'informatique individuelle évolue chaque jour, comme nous avons pu le constater avec l'arrivée des SORD, IBM et autres SIRIUS... Cette remarque relève presque d'une platitude, tant nous savons notre domaine d'intérêt évolutif. Des changements structurels sont cependant en train de s'effectuer, qui vont beaucoup plus loin que la simple apparition de périphériques ou de logiciels nouveaux. En un mot, les constructeurs « classiques » pénètrent en force, et avec des moyens promotionnels extrêmement puissants le domaine de l'informatique individuelle, comme nous pouvons nous en convaincre avec les informations suivantes.

Selon une étude publiée par Computer Business News en mars dernier, l'évolution prévisible pour les quelques années à venir est importante. En 1981, les parts de marché étaient les suivantes : TRS (28 %), Apple (25 %), Hewlett-Packard (11 %), Commodore (10 %), Xerox + DEC + IBM (10 %), autres (16 %). Selon les prévisions, les chiffres de 1985 seraient les suivants : TRS (15 %), Apple (12 %), H-P (14 %), Commodore (3 %), Xerox (17 %), DEC (11 %), IBM (23 %), autres (5 %). Ceci dit, étant donné l'expansion prévue pour ce marché, la baisse de 50 % de part de marché de TRS et Apple devrait en fait correspondre à une hausse confortable du chiffre d'affaires. Ainsi, les ventes d'Apple au premier trimestre 1982 ont été de 131 millions de $ (66 % de hausse sur 81), avec un bénéfice de 10 % du chiffre d'affaires.

Nous aurons donc l'occasion de pennettre aux utilisateurs de langue française d'échanger leur savoir-faire sur Apple pendant encore longtemps. Dans ce numéro, Michel Crimont tennine sa série sur le programme de catalogue général en Pascal. Christian Colmant vous offre une revue des mémoires de masse. J.-F. Duvivier présente un tableau de référence complet des codes ASCII, afin de vous éviter d'avoir à feuilleter des sources multiples pour rechercher certains renseignements de base. Enfin, Dominique Compère, un lecteur de Pom's, propose des programmes en BASIC, largement commentés à l'attention des débutants, destinés à faciliter la création et l'utilisation de tables de formes, en graphique haute résolution.

Toutes les félicitations de l'équipe de Pom's vont à Dominique Devemay, qui a réussi à résoudre notre problème-concours. Nous avons regroupé ses remarques et celles de l'auteur du problème, J.-F. Duvivier, dans un article explicatif. Nous vous apportons dans ce quatrième numéro de nombreuses infonnations de toutes sortes et une grande variété de programmes. A titre de compensation, les mordus de l'assembleur trouveront leur plaisir dans les articles de Gilles Mauffrey et Jacques Tran-Van. Nous faisons dans ce numéro l'expérience de réduire le texte, ce qui nous pennet de vous proposer l'équivalent de 10 pages supplémentaires.

Nous rappelons aux lecteurs qu'ils peuvent commander les disquettes de Pom's séparément, ainsi que les numéros antérieurs. Nous approchons cependant de la rupture de stock sur le numéro 1, dont plus de 2 000 exemplaires ont été vendus avant la fin du mois d'avril.

Hervé Thiriez
Rédacteur en chef

Directeur de la publication — rédacteur en chef : Hervé Thiriez — *Siège social :* Editions MEV — 49, rue Lamartine — 78000 Versailles — *Rédaction et abonnements :* 59, bd de Glatigny — 78000 Versailles — Tél. (3) 918.13.07 — *Régie publicitaire :* Force 7 — 41, rue de la Grange-aux-Belles — 75483 Paris Cedex 10 — Tél. (1) 238.66.10 — *Diffusion auprès des boutiques informatiques et librairies :* Editions du PSI — 41-51, rue Jacquard — B.P. 86 — Tél. (6) 007.59.31.

Ont colloboré à ce numéro : Christian Colmant, Dominique Compère, Michel Crimont, Xavier Dalloz, Dominique Devernay, Jean-François Duvivier, Bruno Estrangin, Gilles Mauffrey, Jean-Louis Meillaud, Gérard Michel, Hervé Thiriez, Jacques Tran-Van — *Conseillers techniques :* Olivier Herz et Gérard Michel — *Maquette :* Jean Mourot — *Dessins :* Laurent Bidot et Jean Mourot.

Communiquez grâce au format DIF

La tentation est grande, dès que l'on peut amener un gros ordinateur à communiquer avec un ordinateur individuel, de transférer des fichiers entre ces deux matériels, afin par exemple de profiter des nombreux programmes disponibles sur l'ordinateur individuel. Ainsi, n'est-il pas possible d'interroger un fichier géré par un gros ordinateur pour le traiter à l'aide d'un programme de calcul (Visicalc), de gestion de base de données (DB Master) ou d'édition de graphiques (Visiplot), à titre d'exemples ?

En fait, cela est plus facile qu'il n'y parait a priori. Il faut d'abord réussir à établir la communication de façon à pouvoir créer le fichier sur disquette. Ensuite, il suffit de mettre les données sous un format qui les rende accesibles à des programmes tels que ceux cités plus haut. Un format unique suffit pour atteindre cet objectif, le format DIF.

1. Le transfert de l'ordinateur central vers l'Apple

La solution à ce problème est spécifique de l'équipement jouant le rôle d'ordinateur central.

Le cas simple est celui d'un ordinateur muni d'une interface RS 232. Du côté de l'Apple, la communication s'effectue au moyen d'une carte de communication. Un logiciel de communication comme VISITERM permet alors de saisir les données dans un fichier.

Nous avons été confrontés à un autre cas, celui de la communication avec un ordinateur CII-HB 61 DPS. Ce matériel est prévu pour fonctionner seulement en relation avec des terminaux. Il faut alors que l'Apple se comporte à son égard exactement comme un terminal. Nous avons pour cela ajouté à notre Apple une carte de communication spéciale fabriquée par SOFRIG. Deux courts programmes (un pour le 61 DPS et un pour l'Apple) permettent de créer un fichier séquentiel en caractères ASCII.

2. Communiquer sans programmer avec Visicalc et DB Master

La société Personal Software a défini un format de fichier séquentiel, dont le but est de servir de norme de communication entre des programmes différents. Sans aucune programmation, des programmes utilisant le format DIF peuvent échanger des données à loisir.
Ce format est appelé DIF, Data Interchange Format; il est accepté par les programmes de Personal Software ainsi que par ceux de Stoneware, la société qui a produit le DB Master. La description détaillée des normes du format DIF se trouve dans les manuels de Visicalc et DB Master; nous y renvoyons le lecteur. Le programme que nous présentons ci-dessous s'inspire très largement du modèle de programme : CREATING A DIF FILE.
Pour traduire le fichier issu de l'ordinateur, nous le recopions préalablement dans un tableau chargé en mémoire centrale. Cette méthode ne convient qu'au transfert de petits fichiers. Dans le cas de fichiers importants, il faudra mettre dans le format DIF les enregistrements les uns après les autres.
Le tableau obtenu ainsi est ensuite traduit en un fichier de format DIF. Ce format comporte un chapeau (header) qui décrit la structure du fichier, notamment le nombre d'enregistrements (vectors) et de rubriques (tuples) par enregistrement.
Ensuite, chaque début d'enregistrement est signalé par "BOT" et, pour chaque rubrique d'un enregistrement:
a) quand la rubrique est numérique, il faut un code "0", suivi de la valeur numérique et de la lettre "V";
b) quand la rubrique est alphanumérique, on doit avoir un code "1", suivi de la valeur "0" et de la chaîne.

3. Un aspect des relations entre ordinateur individuel et ordinateur central

Les données recueillies sur un ordinateur individuel peuvent être manipulées à volonté sans mobiliser l'ordinateur central. Elles peuvent être traitées et/ou complétées localement puisque des logiciels tels que DB Master savent fusionner des données d'origines variées (en utilisant un logiciel complémentaire, le Utility Pack).
Toutefois, la perspective la plus intéressante me parait être l'utilisation de l'ordinateur individuel pour la manipulation des données et la mise en forme de textes ou de graphiques, le stockage des données étant pris en charge par l'ordinateur principal.
La grande diffusion des logiciels pour micro-ordinateurs les rend accessibles à un coût modéré. Ceci conduira à mon avis à un partage des tâches entre grosses et petites machines: l'ordinateur individuel servira à manipuler les données à l'échelon individuel, selon les besoins spécifiques d'un utilisateur. Il devient ainsi le support d'une expression personnelle.

NDLR - Les utilisateurs de Visicalc 16 secteurs doivent consulter le numéro de leur version Visicalc, qui apparait lors du boot. Si celui-ci est "193B0", il faut essayer de la faire remplacer par une bonne version. En effet, il y a des bogues dans cette version, dont le pire est que la sauvegarde d'un fichier DIF à partir de Visicalc peut rendre la disquette illisible; pour couronner le tout, ce phénomène est aléatoire. Attention !

```
]LIST

10  REM :CETTE MAQUETTE
20  REM :TRADUIT UN FICHIER ASCII
      SEQUENTIEL
30  REM :RECU PAR EX. D'UN 61 DPS

40  REM :EN UN FICHIER -DIF-
50  REM :<DATA INTERCHANGE FORMAT
     >
60  REM :<REGISTRED TRADE MARK>
70  REM :<PERSONNAL SOFTWARE>
80  REM
90  REM :LE PROGRAMME COPIE LE FI
     CHIER DANS UN TABLEAU
100  INPUT "NOM DU FICHIER A LIRE
      : ";F$
110  INPUT "NB D'ENREGISTREMENTS
     : ":N%
120  INPUT "NB DE RUBRIQUES :";R%

130 D$ =  CHR$ (4)
140 OP$ = D$ + "OPEN" + F$
150 RD$ = D$ + "READ" + F$
160 CL$ = D$ + "CLOSE" + F$
170  DIM N$(N%,R%)
180  PRINT OP$: PRINT RD$
190  FOR I = 1 TO N%
200  FOR J = 1 TO R%
210  INPUT N$(I - 1,J - 1)
220  NEXT J
230  NEXT I
240  PRINT CL$
250  REM   ----------------------
     --------
260  REM -TRADUCTION DU TABLEAU
270  REM -EN UN NOUVEAU FICHIER
280  REM -LE PROGRAMME UTILISE
290  REM   LE NOM DU FICHIER
300  REM   LE NB DE VECTEURS
310  REM   (ENREGISTREMENTS)
320  REM   LE NB DE TUPLES
330  REM   (RUBRIQUES)
340  REM   LE TYPE DES DONNEES
350  REM     *TYPE 0 = NUMERIQUE
360  REM     *TYPE 1 = CHAINE
370  REM   ----------------------
     --------
```

Suite ⟶

A TRAVERS LA PRESSE APPLE DECHAINEE

La Pomme d'Or revient ce mois-ci à Henry Roberts qui répond aux questions des lecteurs de CALL-A.P.P.L.E. dans une rubrique intitulée APPLE DOCTOR (numéro d'octobre 1981). Nous traduisons ...

Question : On ne parle que du Pascal un peu partout. De quoi s'agit-il et pourquoi tout cet interêt? Que peut-il faire que le BASIC ne peut pas?

Réponse : Le Pascal tire son nom d'un mathématicien Suisse, nommé Blais Pascal (sic) qui vivait il y a une centaine d'années. On n'a jamais pu me fournir la raison de ce choix, alors que Pascal est mort bien avant que le premier ordinateur ne soit construit!

NE S'AGIT-IL PAS PLUTOT D'UN MOINE BELGE DE LA FIN DU MOYEN AGE QUI, LE PREMIER, A DIT SON CHAPELET SUR UNE MACHINE DE WOZNIAC!

```
380  PRINT "NOM DU FICHIER DIF A
     CREER : "
390  REM -LE PROGRAMME SAISIT LE
     NOM DU FICHIER
400  INPUT F$
410 OP$ = D$ + "OPEN" + F$
420 RD$ = D$ + "READ" + F$
430 CL$ = D$ + "CLOSE" + F$
440 WT$ = D$ + "WRITE" + F$
450  PRINT OP$
460  REM -PUIS SAISIT LE NB D'ENR
     EGISTREMENTS
470  REM   DANS LA VARIABLE NV
480 NV = N%
490  REM -ET LE NB DE TUPLES
500  REM   DANS LA VARIABLE NT
510 NT = R%
520  REM -ET ECRIT LE <DIF HEADER
     >
530  GOSUB 800
540  REM -PUIS SAISIT LE TYPE DE
     CHAQUE DONNEE
550  DIM TY%(NT)
560  FOR I = 0 TO NT - 1
570  PRINT "0=NUM  1=CHAINE"
580  PRINT "LE TYPE DE LA RUBRIQU
     E ";I + 1;" EST :"
590  INPUT TY%(I)
600  NEXT I
610  FOR I = 1 TO NV
620  REM -ECRIT LE DEBUT D'UN TUP
     LE
630 T =  - 1:V = 0:S$ = "BOT"
640  GOSUB 910
650  REM -RECHERCHE CHAQUE DONNEE

660  FOR J = 1 TO NT
670 T = TY%(J - 1)
680  IF T = 0 THEN V =  VAL (N$(I
      - 1,J - 1)):S$ = "V"
690  IF T = 1 THEN V = 0:S$ = N$(
     I - 1,J - 1)
700  REM -ET L'ECRIT
710  GOSUB 910
720  NEXT J
730  NEXT I
740  REM -PUIS ECRIT FIN DES DONN
     EES
750 T =  - 1:V = 0:S$ = "EOD"
760  GOSUB 910
770  PRINT CL$
780  PRINT "FIN DE LA CREATION DU
      FICHIER DIF ";F$
790  STOP
800  REM -PROCEDURE D'ECRITURE DU
      <DIF HEADER>
810  PRINT WT$
820  PRINT "TABLE": PRINT "0,1":
     GOSUB 880
830  PRINT "TUPLES": PRINT "0,";N
     T: GOSUB 880
840  PRINT "VECTORS": PRINT "0,";
     NV: GOSUB 880
850  PRINT "DATA": PRINT "0,0":
     GOSUB 880
860  PRINT D$
870  RETURN
880  REM -PROCEDURE D'ECRITURE DE
      ""(CHAINE VIDE)
890  PRINT  CHR$ (34); CHR$ (34)
900  RETURN
910  REM -PROCEDURE D'ECRITURE DE
     S DONNEES
920  PRINT WT$
930  PRINT T;",";V
940  PRINT S$
950  PRINT D$
960  RETURN
970  END
```

Un programme de TRACE sélective

Ce programme permet d'obtenir la trace sélective d'une série d'instructions d'un programme assembleur. Pour rendre ce programme aussi simple et aussi court que possible, les sous-routines moniteur et DOS ont été utilisées très souvent. Le corps du programme (la sous-routine STEP) est pratiquement identique à celui du moniteur de l'Apple II avec BASIC entier en ROM. Ce programme exige un Apple 48K; il est utilisable en mode moniteur ou en mode direct.

Utilisation du programme

Pour l'appel de ce programme, la commande CHAIN du DOS a été modifiée en une commande TRACE; cette commande, pour ne pas être confondue avec la commande BASIC, doit être suivie du nom du programme à tracer et éventuellement du numéro du drive, de l'adresse de départ du trace et de l'adresse finale du trace. La syntaxe est la suivante :
TRACE nomprogramme(,Dx,Rad,Baf)
chacune des informations entre paranthèses étant facultative. Si le numéro de drive est absent, le drive précédemment utilisé sera pris par défaut. Si l'adresse de départ du trace est omise, l'adresse de début de programme sera prise par défaut. Et enfin, si l'adresse de fin de trace est omise, l'adresse de fin de programme sera prise par défaut. Ces données numériques peuvent être données sous forme décimale ou hexadécimale; dans ce dernier cas, elles doivent être précédées du signe $.

Les modifications du DOS et le chargement du programme de TRACE sont faites par le programme PRTR. Quand vous en avez terminé avec le programme TRACE, je vous conseille de rebooter le DOS, car certains paramètres auront été modifiés, et la commande CHAIN supprimée.

Analyse du programme

Quand on tape "TRACE nomprogramme,Rad,Baf"; nomprogramme est mis en adresse AA75 et suivantes, ad en AA6E et AA6F, af en AA70 et AA71. Les premières instructions sauvent les registres et chargent le programme en mémoire centrale, par appel aux sous-routines moniteur SAVE (FF4A) et DOS BLOAD (A35D).

L'adresse de fin de programme est alors calculée, et les adresses de début et fin de trace sont soit verifiées, si elles ont été fournies, soit initialisées dans le cas contraire.

Le programme de trace proprement dit débute alors. Si une instruction doit être tracée, elle est tout d'abord désassemblée à l'aide de la sous-routine moniteur INSDSP (F8D0), sinon seule la longueur de l'instruction est calculée par la sous-routine INSDS2 (F88C); cette longueur est conservée à l'adresse 2F par le moniteur. Ensuite, les instructions à exécuter, c'est-à-dire les instructions qui ne correspondent pas à des sauts, des branchements ou des retours, sont exécutées après restauration des registres grâce à la sous-routine RESTORE (FF3F), et les différents registres sont édités par REGDSP (FAD7). Le programme donne de plus la décomposition du registre d'état pour que l'utilisateur puisse suivre plus facilement le déroulement du programme.

Les instructions de saut et de retour donnent simplement lieu à un calcul de l'adresse de l'instruction suivante. Les instructions de branchement sont transformées pour se brancher soit sur BREL si le test est positif, soit sur NBREL dans le cas contraire. Dans tous les cas, l'adresse de la prochaine instruction à exécuter est alors calculée, à l'aide des différentes sous-routines moniteur PCADJ, une fois les registres conservés.

Il est possible de suspendre à tout moment l'exécution du programme en appuyant sur la touche ESC. Pour interrompre définitivement le programme, il faut taper CONTROL-C.

Limites du programme

Comme l'utilisation des sous-routines moniteur SAVE et RESTORE modifie les adresses 48-49, cela interdit de tracer des programmes faisant appel au DOS, qui les utilise.

D'autre part, on ne peut tracer que des instructions du programme lui-même et non des instructions en ROM par exemple. Nous laissons le soin au lecteur de modifier le programme pour éliminer la première limitation. Quant à la seconde, dans certains cas, elle ne peut être éliminée. Pour s'en persuader, le lecteur pourra regarder ce qui se passe quand on veut tracer la sous-routine de sortie écran COUT ...

```
10  HOME : PRINT "LA COMMANDE CHA
     IN OU DOS EST CHANGEE EN": PRINT
     " UNE NOUVELLE COMMANDE TRAC
     E": PRINT
20  PRINT "BLOAD B.TR": CALL 2159
```

```
SOURCE FILE: TRACE
0083:              1 CONTRC   EQU  $83        * CONTROL-C
009B:              2 ESC      EQU  $9B        * ESCAPE
03EA:              3 RECON    EQU  $3EA
FF4A:              4 SAVE     EQU  $FF4A      * SAUVEGARDE DES REGISTRES
FF3F:              5 RESTORE  EQU  $FF3F      * RESTAURATION DES REGISTRES
F8D0:              6 INSTDSP  EQU  $FBD0      * DESASSEMBLE 1 INSTRUCTION
002C:              7 RTNL     EQU  $2C
002D:              8 RTNH     EQU  $2D
002F:              9 LENGTH   EQU  $2F        * LONGUEUR DE L'INSTRUCTION
0048:             10 STATUS   EQU  $48
003A:             11 PCL      EQU  $3A        * COMPTEUR DE PROGRAMME
003B:             12 PCH      EQU  $3B        * A TRACER
AA72:             13 DFG      EQU  $AA72
AA60:             14 LFG      EQU  $AA60
AA6E:             15 OTR      EQU  $AA6E      * DEBUT DE TRACE
AA70:             16 FTR      EQU  $AA70      * FIN DE TRACE
FCA8:             17 DELAY    EQU  $FCAB        * ROUTINES
F948:             18 PRBLNK   EQU  $F948        * CLASSIQUES
FDED:             19 COUT     EQU  $FDED        * VOIR REF.
FD8E:             20 CROUT    EQU  $FD8E        * MANUAL
C000:             21 KBD      EQU  $C000
C010:             22 KBDSTRB  EQU  $C010
FF69:             23 MON      EQU  $FF69      * RETOUR AU MONITEUR
F88C:             24 INSDS2   EQU  $F88C
F882:             25 INSDS1   EQU  $F882
FAD7:             26 REGDSP   EQU  $FAD7      * IMPRESSION
FADA:             27 REGDSP1  EQU  $FADA      * DES REGISTRES
F954:             28 PCADJ2   EQU  $F954      * CALCUL D'ADRESSES
F956:             29 PCADJ3   EQU  $F956      * DU PROGRAMME
FC58:             30 HOME     EQU  $FC58
A35D:             31 DLOAD    EQU  $A35D      * ADRESSE SS-ROUTINE BLOAD
AA5F:             32 CDOP     EQU  $AA5F      * ADRESSE CODE INST. DOS
0032:             33 CLOAD    EQU  $32        * 2*CODE DOS BLOAD
------ NEXT OBJECT FILE NAME IS TRACE.OBJ0
9400:             34          ORG  $9400
9400:20 4A FF     35          JSR  SAVE
9403:20 58 FC     36          JSR  HOME
9406:A9 32        37          LDA  #CLDAD
9408:8D 5F AA     38          STA  CDOP
940B:20 5D A3     39          JSR  DLOAD      * CHARGEMENT OU PROGRAMME
940E:AD 60 AA     40          LDA  LFG          * CALCUL
9411:18           41          CLC                * DE
9412:6D 72 AA     42          ADC  DFG          * L'ADRESSE
9415:8D D9 95     43          STA  PFI          * DE
9418:AD 61 AA     44          LDA  LFG+1        * FIN
941B:6D 73 AA     45          ADC  DFG+1        * OU
941E:8D DA 95     46          STA  PFI+1        * PROGRAMME
9421:AD 73 AA     47          LDA  DFG+1      * VERIFICATION
9424:CD 6F AA     48          CMP  DTR+1      * DES ADRESSES
9427:90 16        49          BCC  ADFIO      * DE DEBUT ET
9429:F0 09        50          BEQ  SUIVER     * FIN DE TRACE
942B:8D 6F AA     51          STA  DTR+1      * ET VALEURS
```

```
942E:AD 72 AA   52          LDA  DPG     * PAR DEFAUT
9431:4C 3C 94   53          JMP  MDD2    * EVENTUELLES
9434:AD 72 AA   54 SUIVER   LDA  DPG     * DE 47 A 72
9437:CD 6E AA   55          CMP  DTR
943A:90 03      56          BCC  ADFI0
943C:8D 6E AA   57 MOD2     STA  DTR
943F:4C 51 94   58 ADFI0    JMP  VERFIN
9442:AD DA 95   59 DEF2     LDA  PFI+1
9445:8D 71 AA   60          STA  FTR+1
9448:AD D9 95   61          LDA  PFI
944B:8D 70 AA   62          STA  FTR
944E:4C 67 94   63          JMP  TR0
9451:AD 71 AA   64 VERFIN   LDA  FTR+1
9454:F0 EC      65          BEQ  DEF2
9456:CD DA 95   66          CMP  PFI+1
9459:90 04      67          BCC  SUIFIN
945B:F0 02      68          BEQ  SUIFIN
945D:B0 E3      69          BCS  DEF2
945F:AD 70 AA   70 SUIFIN   LDA  FTR
9462:CD D9 95   71          CMP  PFI
9465:B0 DB      72          BCS  DEF2
9467:AD 72 AA   73 TR0      LDA  DPG
946A:85 3A      74          STA  PCL
946C:AD 73 AA   75          LDA  DPG+1     * INITIALISATION DU PROGRAMME
946F:85 3B      76          STA  PCH
9471:20 94 94   77 TRACE    JSR  WAIT1
9474:20 B6 94   78          JSR  SITR
9477:B0 09      79          BCS  FREEX
9479:20 8F 94   80          JSR  WAIT
947C:20 D0 F8   81          JSR  INSTDSP
947F:4C 89 94   82          JMP  EXEC
9482:A2 00      83 FREEX    LDX  £00
9484:A1 3A      84          LDA  (PCL,X)
9486:20 8C F8   85          JSR  INSDS2
9489:20 E6 94   86 EXEC     JSR  STEP
948C:4C 71 94   87          JMP  TRACE
948F:           88 ** SOUS PROGRAMME DE SAISIE CLAVIER ET DELAY
948F:A9 CB      89 WAIT     LDA  £203
9491:20 A8 FC   90          JSR  DELAY
9494:2C 00 C0   91 WAIT1    BIT  KBD
9497:10 1C      92          BPL  WRTS
9499:AD 00 C0   93          LDA  KBD
949C:C9 9B      94          CMP  £ESC
949E:D0 08      95          BNE  WRTS0
94A0:2C 10 C0   96          BIT  KBDSTRB
94A3:2C 00 C0   97 W2       BIT  KBD
94A6:10 FB      98          BPL  W2
94A8:AD 00 C0   99 WRTS0    LDA  KBD
94AB:2C 10 C0  100          BIT  KBDSTRB
94AE:C9 83     101          CMP  £CONTRC
94B0:D0 03     102          BNE  WRTS
94B2:4C 2F 95  103          JMP  SORTIE
94B5:60        104 WRTS     RTS
94B6:          105 ** SOUS PROGRAMME VERIFICATION A TRACER DU NON
94B6:A5 3B     106 SITR     LDA  PCH
94B8:CD 6F AA  107          CMP  DTR+1
94BB:90 22     108          BCC  NON
94BD:D0 07     109          BNE  SITR2
94BF:A5 3A     110          LDA  PCL
94C1:CD 6E AA  111          CMP  DTR
```

```
94C4:90 19      112          BCC  NON
94C6:A5 3B      113 SITR2    LDA  PCH
94C8:CD 71 AA   114          CMP  FTR+1
94CB:90 0B      115          BCC  OUI
94CD:D0 10      116          BNE  NON
94CF:A5 3A      117          LDA  PCL
94D1:CD 70 AA   118          CMP  FTR
94D4:90 02      119          BCC  OUI
94D6:D0 07      120          BNE  NON
94D8:18         121 OUI      CLC
94D9:A9 00      122          LDA  £00
94DB:8D CF 95   123          STA  INDIC
94DE:60         124          RTS
94DF:38         125 NON      SEC
94E0:A9 01      126          LDA  £01
94E2:8D CF 95   127          STA  INDIC
94E5:60         128          RTS
94E6:           129 ** ANALYSE ET EXECUTION DE L'INSTRUCTION
94E6:68         130 STEP     PLA          * NOUS REPRENONS
94E7:85 2C      131          STA  RTNL    * ICI LE STEP
94E9:68         132          PLA          * DU MONITEUR
94EA:85 2D      133          STA  RTNH    * DE L'APPLE II
94EC:A9 EA      134          LDA  £$EA    * BASIC INTEGER
94EE:8D C6 95   135          STA  INSTR   * SANS AUTOSTART
94F1:8D C7 95   136          STA  INSTR+1
94F4:8D C8 95   137          STA  INSTR+2
94F7:A2 00      138          LDX  £00
94F9:A1 3A      139          LDA  (PCL,X)
94FB:F0 2C      140          BEQ  BREAK
94FD:A4 2F      141          LDY  LENGTH
94FF:C9 20      142          CMP  £$20
9501:F0 50      143          BEQ  JSR
9503:C9 60      144          CMP  £$60
9505:F0 3C      145          BEQ  RETOUR
9507:C9 4C      146          CMP  £$4C
9509:F0 53      147          BEQ  JUMP
950B:C9 6C      148          CMP  £$6C
950D:F0 50      149          BEQ  INJUMP
950F:C9 40      150          CMP  £$40
9511:F0 2C      151          BEQ  XRTI
9513:29 1F      152          AND  £$1F
9515:49 14      153          EOR  £$14
9517:C9 04      154          CMP  £$04
9519:F0 02      155          BEQ  RELAT
951B:B1 3A      156 ECIN     LDA  (PCL),Y
951D:99 C6 95   157 RELAT    STA  INSTR,Y
9520:88         158          DEY
9521:10 F8      159          BPL  ECIN
9523:20 3F FF   160          JSR  RESTORE
9526:4C C6 95   161          JMP  INSTR
9529:20 82 F8   162 BREAK    JSR  INSDS1
952C:20 DA FA   163          JSR  REGDSP1
952F:20 EA 03   164 SORTIE   JSR  RECON
9532:A9 00      165          LDA  £00
9534:A2 03      166          LDX  £03
9536:9D 6E AA   167 SOR0     STA  OTR,X
9539:CA         168          DEX
953A:10 FA      169          BPL  SOR0
953C:4C 69 FF   170          JMP  MON
953F:18         171 XRTI     CLC
9540:68         172          PLA
```

```
]BLOAD CHARGE.OBJ,A$9000
]CALL-151


*9000.922A

9000- A9 00 8D F6 03 A9 9B 8D
9008- F7 03 A9 4C 8D F5 03 A2
9010- 00 BD 28 90 9D 00 9B E8
9018- D0 F7 BD 28 91 9D 00 9C
9020- E8 D0 F7 A9 9A 8D 01 9D
9028- 4C D4 A7 A5 1E BD AC AA
9030- A5 1F 8D AD AA 20 B7 00
9038- C9 22 D0 28 A9 06 85 C1
9040- A0 00 8C EB B7 20 B1 00
9048- C9 22 F0 0C 09 80 99 75
9050- AA C8 C0 1E 90 EF B0 34
9058- A9 EF 85 C1 20 B1 00 C0
9060- 00 D0 4C 60 A5 B9 48 A5
9068- B9 48 20 7B DD 24 11 30
9070- 08 A2 A3 20 F5 9C 4C 12
9078- D4 60 85 B9 68 85 B8 20
9080- E3 DF 85 1E 84 1F A0 00
9088- B1 1E D0 05 A2 0B 4C 0A
9090- 9C 8D AB AA C8 B1 1E 48
9098- C8 B1 1E 85 1F 68 85 1E
90A0- A0 00 B1 1E 09 80 99 75
90A8- AA C8 CC AB AA D0 F3 A9
90B0- A0 99 75 AA C8 C0 1E D0
9088- F8 8D B0 AA 20 F5 9C 20
90C0- B7 00 F0 12 20 BE DE 20
90C8- 67 DD 20 52 E7 A5 51 C9
90D0- 02 90 B9 4E B0 AA A9 01
90D8- 8D F4 B7 A9 10 8D ED B7
90E0- A9 11 8D EC B7 20 E6 9C
90E8- CE ED B7 F0 46 20 CB 9C
90F0- A0 EB 8C AE AA AD AE AA
90F8- 18 69 23 A8 8D AE AA C9
9100- 03 F0 E5 A2 00 B9 BB 83
9108- F0 29 DD 75 AA D0 E6 C8
9110- E8 E0 1E D8 F0 B9 9C B3
```

```
9541:85 48      173          STA  STATUS
9543:68         174 RETOUR   PLA
9544:85 3A      175          STA  PCL
9546:68         176          PLA
9547:85 3B      177 PCINC2   STA  PCH
9549:A5 2F      178 PCINC3   LDA  LENGTH
954B:20 56 F9   179          JSR  PCADJ3
954E:84 38      180          STY  PCH
9550:18         181          CLC
9551:90 14      182          BCC  NEWPCL
9553:18         183 JSP      CLC
9554:20 54 F9   184          JSR  PCADJ2
9557:AA         185          TAX
9558:98         186          TYA
9559:48         187          PHA
955A:8A         188          TXA
955B:48         189          PHA
955C:A0 02      190          LDY  £$02
955E:18         191 JUMP     CLC
955F:B1 3A      192 INJUMP   LDA  (PCL),Y
9561:AA         193          TAX
9562:88         194          DEY
9563:B1 3A      195          LDA  (PCL),Y
9565:86 3B      196          STX  PCH
9567:85 3A      197 NEWPCL   STA  PCL
9569:B0 F3      198          BCS  JUMP
956B:A5 2D      199          LDA  RTNH
956D:48         200          PHA
956E:A5 2C      201          LDA  RTNL
9570:48         202          PHA
9571:AD CF 95   203          LDA  INDIC
9574:D0 06      204          BNE  PASEC
9576:20 D7 FA   205          JSR  REGDSP
9579:20 7D 95   206          JSR  ECRFIN
957C:60         207 PASEC    RTS
957D:           208 ** SOUS PROGRAMME ECITURE COMPLEMENTAIRE
957D:20 8E FD   209 ECRFIN   JSR  CROUT
9580:A5 48      210          LDA  STATUS
9582:8D D0 95   211          STA  TAMPON
9585:A0 08      212          LDY  £$08
9587:B9 D0 95   213 P00      LDA  TAB-1,Y    * ECRITURE
958A:C9 AD      214          CMP  £'-
958C:D0 06      215          BNE  P01
958E:4E D0 95   216          LSR  TAMPON     * DETAILLEE
9591:4C AE 95   217          JMP  LOOP
9594:20 ED FD   218 P01      JSR  COUT       * DES
9597:A9 BD      219          LDA  £'=
9599:20 ED FD   220          JSR  COUT
959C:4E D0 95   221          LSR  TAMPON     * REGISTRES
959F:90 05      222          BCC  P0
95A1:A9 B1      223          LDA  £'1
95A3:4C A8 95   224          JMP  VAL
95A6:A9 B0      225 P0       LDA  £'0
95A8:20 ED FD   226 VAL      JSR  COUT
95AB:20 48 F9   227          JSR  PRBLNK
95AE:88         228 LOOP     DEY
95AF:D0 D6      229          BNE  P00
9581:60         230          RTS
95B2:18         231 BREL     CLC             * BRANCHEMENT
95B3:A0 01      232          LDY  £$01       * RELATIF
95B5:B1 3A      233          LDA  (PCL),Y
```

```
9118- 29 04 D0 04 A2 0D D0 15
9120- AC AE AA B9 B9 B3 BD ED
9128- B7 B9 BB B3 30 C7 8D EC
9130- B7 D0 0E A2 06 BE 5C AA
9138- 20 F5 9C 20 93 FE 4C D5
9140- A6 8D AF AA 20 CB 9C A2
9148- 0C A5 1E A4 1F 20 EE 9C
9150- AD AF AA F0 03 20 EA 9C
9158- 20 BB 9C AD AF AA F0 40
9160- 2C B0 AA 30 0A A5 50 8D
9168- BB B4 A5 51 8D BC B4 38
9170- AD BB B4 E9 04 85 1E 8D
9178- F0 B7 AD BC B4 E9 00 85
9180- 1F 8D F1 B7 A9 04 BD DA
9188- 9C 1B 6D BD B4 8D AB AA
9190- A9 00 BD E2 9C BD AF AA
9198- AD C9 B3 F0 22 20 D9 9C
91A0- AE AE AA EE F1 97 E6 1F
91A8- EB E0 FE B0 1D BD BD B3
91B0- D0 A6 20 EA 9C 20 BB 9C
91B8- A9 00 85 4B BD DA 9C AD
91C0- AB AA 8D E2 9C 20 D9 9C
91C8- F0 56 AD BC B3 F0 E3 20
91D0- BB 9C AD BD B3 BD ED B7
91D8- E6 1F 20 E6 9C AD BC B3
91E0- BD EC B7 4C 19 9C ED BB
91E8- 83 8D EC B7 E8 BD BB B3
91F0- BD ED B7 8E AE AA A9 B7
91F8- A0 EB 20 B5 B7 90 11 A2
9200- 0B 4C 0A 9C A0 00 89 BB
9208- 84 91 1E C8 C0 00 D0 F6
9210- 60 A0 B3 D0 02 A0 B4 A9
9218- BB 8D F0 B7 BC F1 B7 60
9220- AD AC AA 85 1E AD AD AA
9228- 85 1F 60
```

```
95B7:20 56 F9  234          JSR  PCADJ3 * CALCUL DE LA NOUVELLE ADRESSE
95BA:85 3A     235          STA  PCL
95BC:98        236          TYA
95BD:38        237          SEC
95BE:B0 87     238          BCS  PCINC2
95C0:20 4A FF  239 NBREL    JSR  SAVE   * PAS DE PROBLEME
95C3:38        240                      * SAUVEGARDE LES REGISTRES
95C4:B0 83     241          BCS  PCINC3 * ET CALCULE LA PROCHAINE ADRESSE
95C6:EA        242 INSTR    NOP         * PLACE
95C7:EA        243          NOP         * DE L'INSTRUCTION
95C8:EA        244          NOP         * A TRACER
95C9:4C C0 95  245          JMP  NBREL
95CC:4C B2 95  246          JMP  BREL
95CF:00        247 INDIC    DFB  00
95D0:00        248 TAMPON   DFB  00
95D1:CE D6 AD  249 TAB      ASC  'NV-BDIZC
95D4:C2 C4 C9
95D7:DA C3
95D9:00 00     250 PFI      DFB  00,00

*** SUCCESSFUL ASSEMBLY: NO ERRORS

SOURCE FILE: MAJDOS
0073:             1 HIMEM   EQU  $73
9400:             2 DEBUT   EQU  $9400
9D26:             3 APTR    EQU  $9D26
A893:             4 LABE    EQU  $A893
A968:             5 RH      EQU  $A968
A96C:             6 BH      EQU  $A96C
A911:             7 PARTR   EQU  $A911
----- NEXT OBJECT FILE NAME IS MAJDOS.OBJ0
086F:             8         ORG  $86F
086F:A9 00        9         LDA  #>DEBUT
0871:85 73       10         STA  HIMEM
0873:A9 94       11         LDA  #<DEBUT
0875:85 74       12         STA  HIMEM+1
0877:38          13         SEC
0878:A5 73       14         LDA  HIMEM
087A:E9 01       15         SBC  #01
087C:8D 26 9D    16         STA  APTR
087F:A5 74       17         LDA  HIMEM+1
0881:E9 00       18         SBC  #00
0883:8D 27 9D    19         STA  APTR+1
0886:A2 04       20         LDX  #04
0888:BD A4 08    21 LOOP    LDA  TRAC,X
088B:9D 93 A8    22         STA  LABE,X
088E:CA          23         DEX
088F:10 F7       24         BPL  LOOP
0891:A9 FF       25         LDA  #$FF
0893:8D 68 A9    26         STA  RH
0896:8D 6C A9    27         STA  BH
0899:A9 A0       28         LDA  #$A0
089B:8D 11 A9    29         STA  PARTR
089E:A9 76       30         LDA  #$76
08A0:8D 12 A9    31         STA  PARTR+1
08A3:60          32         RTS
08A4:54 52 41    33 TRAC    DCI  'TRACE
08A7:43 C5

*** SUCCESSFUL ASSEMBLY: NO ERRORS
```

Un Apple à la clinique

1. Description générale

Pom's a rendu visite au professeur David Lewin, qui avait été désigné en 1968 par le congrès des gynécologues de langue française pour voir comment on pouvait utiliser un ordinateur dans la gestion de dossiers médicaux en obstétrique et gynécologie. Le professeur Lewin a eu la gentillesse de nous expliquer comment il s'y était pris, et pourquoi il utilise maintenant un Apple à cet effet.

Le problème essentiel soulevé par la gestion informatique de dossiers médicaux est que ceux-ci comportent potentiellement une grande masse et une grande variété d'informations. En outre, pour tout simplifier, ces informations sont de types et de qualité très diverses (par exemple, comment mémorise-t-on une radio ?). Vouloir mémoriser la totalité des informations relatives à un patient est très difficilement envisageable, d'autant qu'un dossier s'enrichit en permanence de données chronologiques. En outre, certaines données sont très longues, par exemple un électro-cardiogramme de 24 heures !

Il faut donc choisir de mémoriser une quantité limitée d'informations, sinon il faudrait compter en méga-octets par patient.

2. L'obstétrique

Dans une première étape, le professeur Lewin choisit de s'attaquer au seul problème de l'obstétrique. L'avantage, dans ce cas particulier, est que l'horizon est limité à une durée inférieure à une année : il n'y a donc pas cumul permanent pour chaque patiente. Un autre avantage de l'obstétrique est qu'il existait depuis déjà fort longtemps une fiche de renseignements quantitatifs qui était remplie habituellement de façon manuelle. Il a donc suffi d'analyser cette fiche en détail, et de concevoir la meilleure façon de la formaliser pour obtenir le dessin de l'enregistrement à prévoir pour chaque patiente.

Le premier programme fut écrit en Fortran sur un CII 10070; il tournait en 5 heures un quart, et comportait 5.500 instructions. Dès la fin de la première année, il fallut en réécrire une bonne partie à cause d'un changement de système d'exploitation ! Le programme, qui marchait parfaitement antérieurement, se mettait à éclater avec un fatidique JOB ABORTED particulièrement gênant dans ce type d'application ...

Une seconde version vit le jour en Cobol sur un IRIS80, suivie par une troisième version en GAP2 sur IBM32. Le professeur Lewin s'était mis à la programmation et a réalisé ces deux dernières versions, la première avec les conseils d'Alain Berdugo. Désirant avoir un matériel sous la main, il se mit à l'Apple, et écrivit la dernière version en Basic et langage machine (sans assembleur, s'il vous plaît).

Le programme tourne maintenant parfaitement et parvient à enregistrer sur une disquette en DOS 3.3 jusqu'à 1000 dossiers, ce qui est une performance. En effet, un dossier comporte le nom et les coordonnées de la patiente, plus 150 informations. Or, la disquette 16 secteurs ne dispose même pas de 150 octets par patiente, si l'on désire en gérer 1000 ! Ceci est devenu possible par une recodification complète des informations.

Une dizaine d'autres obstétriciens sont maintenant équipés de ce système clés-en-main grâce au professeur Lewin. Le programme permet la saisie, l'archivage, la recherche, la mise à jour et certaines analyses statistiques.

Depuis 1969, les fiches d'inscription manuelles sont toutes regroupées et incorporées dans une base de données qui est en fait la plus importante base de données en obstétrique du monde, avec 600.000 dossiers. Pour le moment, les analyses effectuées sur cette base de données ne sont pas encore réalisées sur matériel Apple ! Par contre, les fiches provenant des praticiens déjà équipés par le professeur Lewin sont regroupées en plus dans un fichier particulier, dont on sait qu'il est encore plus fiable, puisque le programme de saisie réalise un certain nombre de tests et filtre les données. Ce sous-fichier pourra être traité et analysé avec un Apple équipé d'un disque.

3. La gynécologie

La gynécologie représente un problème de toute autre nature, car il faut alors, comme en médecine classique, garder le dossier d'une patiente de façon permanente. En outre, contrairement au cas de l'obstétrique, les données intéressantes ne sont pas en

dominance des informations quantitatives : il y a de nombreuses données qualitatives que l'on ne peut se permettre d'ignorer.

La question qui se posait alors était donc la suivante : faut-il mettre en place un fichier mixte avec des composantes fermées (structure fixe) et ouvertes (possibilité de texte libre) ? Faut-il définir comme dans le premier cas un fichier fermé, ou au contraire définir un fichier ouvert ? Dans tous les cas rencontrés par le professeur Lewin, en matière d'informatisation de dossiers médicaux, les fichiers de structure mixte ont abouti à des échecs. Un fichier ouvert a par conséquent été défini, avec pour seule partie fermée la partie signalétique, dont on peut en effet difficilement se passer.

Le programme résultant est donc une sorte de base de données, dans laquelle l'utilisateur définit ses propres rubriques. A l'intérieur de chacune d'entre elles, il peut utiliser des mots-clés qu'il définit lui-même, avec un glossaire de 1000 mots au maximum. Le programme se charge de la recherche des dossiers dans lesquels se trouvent des conjonctions données de mots-clés, ou des mots-clés donnés. A part cela, toutes les fonctions habituelles d'une gestion de fichier sont disponibles. Sur le même principe, un programme de gestion bibliographique a d'ailleurs été conçu, comme variante du programme de gestion de dossiers en gynécologie.

4. Conclusion

Outre les aspects de gestion de dossier, les deux programmes peuvent apporter en temps réel une aide clinique importante. On peut grâce à eux interroger le système quand un problème délicat se pose, et faire des recherches sur les relations entre diverses données physiques et de santé, entre des problèmes, leurs traitements et les résultats statistiques, ...

En conclusion, ce qui est intéressant dans cette expérience, c'est que l'on ait pu arriver à effectuer un certain nombre de travaux sur un petit matériel, après l'avoir fait sur un matériel beaucoup plus puissant. C'est le contraire de l'approche courante, dans laquelle on a tendance à avoir besoin d'un matériel toujours plus puissant. En outre, on constate que le professeur Lewin a dû se mettre à tout programmer pour pouvoir être satisfait du programmeur. Heureusement pour les SSCI que ce n'est pas le cas le plus fréquent !

Enfin, il faut noter que l'approche fichier ouvert est utilisable non seulement dans le domaine gynécologique, mais dans l'ensemble de la profession médicale, et même au-delà.

Les mémoires de masse

Sur tout micro-ordinateur, le problème de la capacité de stockage se pose toujours tôt ou tard, pour une multitude de raisons : nécessité de placer un grand nombre de programmes sur support magnétique, ou de mémoriser des fichiers de grande taille.

Les utilisateurs d'Apple se sont vite rendu compte qu'un espace de 140K octets, sur une disquette 5¼ pouces, était vite saturé; heureusement, il existe plusieurs façons de pallier cette limitation. La moins onéreuse est bien entendu de manipuler tel un jongleur des disquettes 5¼ pouces sur un drive unique. On investit rapidement dans un second lecteur , qui autorise un doublement de la capacité en ligne et, surtout, facilite grandement les copies de disquettes. Au-delà, on passe aux mémoires de masse qui sont en général des disques de plus grande capacité. Nous vous présentons dans cet article les principales mémoires de masse disponibles sur le marché pour Apple ou équivalents Silex et ITT 2020. Notre objectif est ici de présenter les principaux supports de stockage éprouvés sur le marché français, plutôt que de réaliser un dossier complet sur l'ensemble des mémoires de masse compatibles avec un Apple.

On distingue habituellement deux types de supports magnétiques de masse : d'un côté les disques souples (floppies) de grande capacité, de l'autre les disques durs. Analysons les principaux avantages et inconvénients de chaque solution.

Le disque souple (floppy)

Pour commencer, le lecteur de disques souples de 5¼ pouces a l'avantage douteux d'être le plus cher par K-octet, avec ses 140K en simple face, simple densité. En compensation, c'est, après la cassette dont nous ne parlons même pas tant son utilité est réduite, le matériel de lecture/écriture de mémoire de masse (si l'on peut dire) dont le coût fixe est le plus bas.

Les lecteurs de disques souples 8 pouces, du fait de leur diamètre plus important, acceptent une capacité de stockage plus grande, allant de 256K octets à 1M octet (1 méga-octet=1.000K), selon qu'il s'agit de lecteurs simple ou double face, et selon la densité d'enregistrement. Bien que d'un coût relativement élevé, ce type de lecteur offre un compromis valable entre l'investissement et la capacité de stockage. De plus, du fait de leur antériorité sur le marché professionnel, les disques souples 8 pouces sont un standard portable des matériels professionnels normalisés (norme IBM 3740); il devient possible, en utilisant cette norme, d'opérer des transferts de fichiers d'un ordinateur d'une marque donnéée à un ordinateur d'une autre marque. La société française Léanord a d'ailleurs pris une place prépondérante, avec la gamme des SILDISC, sur le marché des disques souples 8 pouces pour Apple. L'inconvénient majeur de ces lecteurs réside dans leur coût, qui les réserve à une clientèle professionnelle.

Depuis peu, des lecteurs de disques souples 5¼ pouces de grande capacité sont proposés par IEF, une autre société française. Chacun de ces disques, travaillant en double face et grande densité, peut stocker 1M octets. Le contrôleur IEF supporte deux lecteurs, comme le contrôleur Apple, et permet donc de disposer de 2M octets en ligne. Ce lecteur représente aujourd'hui le meilleur rapport investissement/capacité, quel que soit le système d'exploitation (SED) utilisé. Il exige cependant, et c'est compréhensible, l'emploi de disquettes spécialement contrôlées.

En conclusion, les disques souples offrent l'avantage de l'interchangeabilité et de la transportabilité, en compensation de leur capacité réduite par rapport à celle des disques durs.

Les disques durs (hard disks)

De conception totalement différente, les disques durs ont une très grande capacité de stockage, allant jusqu'aux 380M octets obtenus chez Micromos avec plusieurs disques durs de type Cynthia (fabrication Honeywell). Un disque dur représente un investissement élevé, mais possède le coût par K-octet le plus bas de tous les supports magnétiques.

Le premier disque dur disponible sur le marché Apple fut le célèbre Corvus, de 10M octets, distribué en France par Micrologie. Il est partagé en plusieurs volumes représentant chacun un équivalent-disquette de 140K sous DOS. Par contre, sous Pascal, il est capable de gérer des fichiers de toutes capacités. Dans sa gamme, Corvus propose aujourd'hui des disques durs de 5, 10 et 20M octets.

Les utilisateurs d'un tel disque, scellé sous vide dans sa boîte, ont tout de suite saisi le risque encouru en cas de destruction inopinée d'un gros fichier. Aussi a-t-on étudié le problème de la sauvegarde des informations sur un autre support. Une possibilité consiste à recopier sur des

disquettes 5¼ pouces, mais il faut 67 disquettes et beaucoup de patience pour effectuer une sauvegarde complète. En outre, si un fichier est de taille supérieure à 140K, il faut le découper pour pouvoir le sauvegarder, ce qui ne simplifie pas l'opération. C'est pourquoi la sauvegarde sur magnétoscope est devenue la solution la plus fréquente; de plus, le magnétoscope passe ainsi au titre des frais professionnels ...

Le disque dur Cynthia, commercialisé par IEF, Microexpansion, Micromos et MIS, est une solution élégante à ce problème de sauvegarde. On dispose dans une même unité d'un disque fixe de 10M octets, et d'un disque amovible de même capacité. La sauvegarde s'effectuant sur une cartouche amovible, plus de problème d'échange de fichiers ni de recopie de sécurité.

Les disques durs disposent en outre d'un système de multiplexage qui permet de relier plusieurs Apples à un disque donné et de réaliser ainsi des applications dans lesquelles plusieurs utilisateurs disposant chacun de leur poste de travail, accèdent aux mêmes fichiers.

Conclusion

L'Apple III n'a rien apporté de neuf en matière de disques souples. Par contre, l'existence du Profile qui apporte 5M octets à un prix abordable prouve l'existence du besoin des utilisateurs en mémoire de masse. On peut simplement regretter que rien n'ait été prévu pour la sauvegarde des informations stockées sur le Profile: qui a envie de faire 35 copies de disquettes consécutives, dans le pire des cas ?

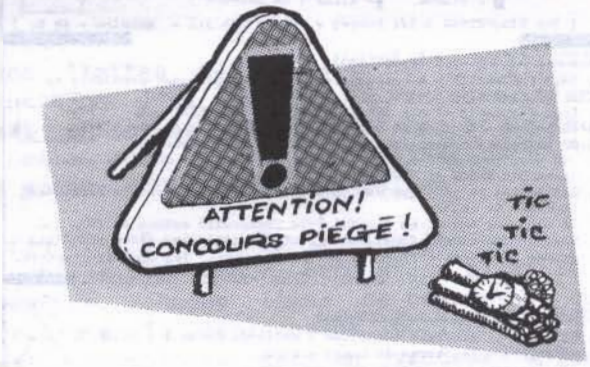
On peut penser aujourd'hui que le bon compromis réside dans la solution mixte souple+dur, avec sauvegarde du disque dur sur le disque souple, sous réserve que celui ait une capacité suffisante comme dans le cas du 1M otets sur 5¼ pouces. D'autant que le disque dur peut être relié à plusieurs Apples ou autres ordinateurs individuels.

TABLEAU COMPARATIF DES PRINCIPALES MEMOIRES DE MASSE POUR APPLE II-SILEX-ITT 2020

Caractéristiques	Sildisc Modèle E	Sildisc Modèle F	Floppy I.E.F.	Corvus	Galaxian 140	Cynthia
Constructeurs ou distributeurs	Léanord	Léanord	I.E.F.	Micrologie	Micro-Exp.+ MIS	Micromos + I.E.F.
Capacité standard	E1:2*256K E2:2*512K	F1:2*512K F2:2* 1M	1M octets 2M octets	5-10-20 M-octets	2*10M	2*10M
Extensions possibles	2è unité	2è unité	2è unité	2è unité	Non	Non
Systèmes d'exploitation						
DOS	Oui	Oui	Oui	Oui	Non	Oui
Pascal UCSD	Oui	Oui	Oui	Oui	Non	Oui
CP/M	Non	Oui	Oui	Oui	Non	NC
M/DOS	Non	Oui	Oui	NC	Oui	Oui
Support	2*8"	2*8"	1M: 1*5"¼ 2M:2*5"¼	Disque scellé	2 disques fixe+amov	2 disques fixe+amov
Compatibilité IBM	Non	Oui	Non	Non	Non	Non
Taux de transfert	250Kbit/s	500Kbit/s	250Kbit/s	NC	920Koct/s	920Koct/s
Sauvergarde	---	---	---	Magnétos. Sy.miroir	Disque amovible	Disque amovible
Dimensions l*L*h cms	47.62.13	32.55.31	cf. Apple	Selon type	68.50.25	68.50.25
Poids	15 kgs	18 kgs	idem	idem	45 kgs	45 kgs
Prix de base (H.T.)	E1:16800F E2 23000F	F1:25500F F2 29500F	1M: 11000F 2M 19000F	36.000 F	59.000 F 10M octets	59.000 F
Prix/K-octets (H.T.)	E1:32,62F E2:22,46F	F1:24,90F F2:14,40F	1M: 10,74F 2M: 9,50F	3,6F	2,88F	2,88F
Nb Apples connectables	1	1	1	1 à 64	1 à 4	1-16+

Réponse au concours de Pom's

```
>LIST
20733 CALL -936: TAB 19: VTAB 13
20734 DEVINETTE= ASC("POMS")* PEEK
      (-1823)/ PEEK (-1595)*POMS ^
      POMS
20735 POKE PEEK (-7782),12: POKE
      PEEK (-4710),184
20736 POKE 2069, PEEK ( RND (4096
      ))
20737 PRINT "POMS"
20738 POKE POMS,POMS
20739 POP
```



Nous avons regroupé ici les explications de Dominique Devernay (initialement condensées en quelques formules lapidaires, jetées sur le papier dans l'ivresse de la victoire), et celles de J.-F. Duvivier, beaucoup plus didactiques (à l'époque de leur rédaction, le pauvre J.F.D. croyait être le seul à détenir la Vérité...).

* Ligne 20733 : pas de mystère.

* Ligne 20734 : elle attribue à la variable DEVINETTE la valeur 215. En effet :
  - ASC(POMS)=ASC(P)=208
  - PEEK(-1823) est une constante, car l'adresse -1823 se trouve dans le moniteur. De fait, PEEK(-1823)=144
  - PEEK(-1595)=139 (cf. ci-dessus)
  - La variable POMS est égale à 0, car elle n'a pas encore été utilisée. En conséquence, POMS^POMS = 0^0 = 1
  - En résumé, DEVINETTE = 208*144/139*1 = 215,48 = 215 (on est en INTEGER).

* Ligne 20735 : $E19A (-7782) et $ED9A (-4710) se trouvent en ROM et contiennent donc des valeurs constantes (respectivement 78 et 79). Cette ligne peut donc s'écrire POKE 78,12:POKE 79,184. Les adresses 78 et 79 correspondent aux registres servant de "racines" pour le générateur de nombres aléatoires. Si RDKEY n'est pas appelé entre temps, RND(4096) donne obligatoirement 2060, en raison des valeurs chargées dans les registres (12 et 184).

* Ligne 20736 : en raison des explications ci-dessus, cette ligne équivaut à POKE 2069,PEEK(2060). Elle modifie la table des variables de l'INTEGER, située à partir de l'adresse 2048 ($800). Plus précisément :

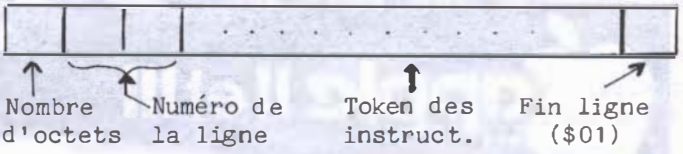
2060 = adresse de l'octet de poids faible de la variable DEVINETTE

2069 = adresse de l'octet de poids faible de la variable POMS.

Ceci revient donc à POMS=DEVINETTE.

* Ligne 20737 : sans commentaires. Mais attention, car nous en arrivons au point le plus délicat.

* Ligne 20738 : depuis les manipulations sur les variables de la ligne 20736, la variable POMS vaut 215. Cette ligne effectue donc un simple POKE 215,215 soit en hexadécimal POKE $D7,$D7. Pour en mesurer la portée, il est nécessaire de connaître un peu la structure d'une ligne de programme INTEGER stockée en mémoire. Chaque ligne est ainsi représentée :



L'interpréteur BASIC INTEGER analyse la mémoire octet par octet et exécute les instructions. A chaque fin de ligne, c'est-à-dire chaque fois qu'il rencontre un $01, il sait que les trois octets suivants ne représentent plus des instructions mais sont des informations sur la ligne suivante (longueur et numéro). Il positionne un "flag" (littéralement "drapeau"), situé en page zéro à l'adresse $D7, lui permettant de savoir dans quelle configuration il se trouve. Si la valeur en $D7 est supérieure à 128, l'interpréteur sait que les octets suivants sont des instructions; si elle est inférieure, il sait que les trois octets suivants sont des "informations de ligne".

Lorsque l'interpréteur a analysé la ligne 20738, il a trouvé un $01 signalant la fin de ligne et a donc positionné $D7 à une valeur inférieure à 128. Il exécute ensuite l'instruction décodée (le POKE 215,215) qui

remet justement $D7 à une valeur supérieure à 128.

* Ligne 20739 - La ligne 20739 est stockée comme suit en mémoire :
05-03-51-77-01, soit :
05 : nombre d'octets de la ligne
03-51 : représente 20739 en Hexa
77 : "token" pour l'instruction POP
01 : fin de ligne.

Quand l'interpréteur en arrive à cette ligne 20739, il trouve trois octets indiquant la longeur de la ligne et son numéro, mais sur la foi de son "flag" $D7, il "croit" que ce sont des instructions. La ligne 20739 est donc analysée comme suit :
05 = "token" pour SAVE (sur cassette)
03 = ":" séparateur d'instructions
51 = "token" de l'instruction END
77 = POP (ne provoque pas d'erreur puisque END a été rencontré).

Ainsi, après le "POKE 215,215", l'Apple exécute un SAVE sur la sortie cassette (responsable du temps d'arrêt observé ainsi que des deux 'bip') et un END. D'ailleurs, comme le fait remarquer D. Devernay, un POKE -27142,3 permet de lister cette dernière ligne bien malmenée :

```
20738 POKE POMS,POMS:SAVE:END:POP
```

Rendez-vous maintenant pour les prochains concours, qui permettront peut-être à D. Devernay d'assurer son approvisionnement en Pom's jusqu'à l'aube du troisième millénaire!

Création de tables de formes

La toute-puissante instruction DRAW du BASIC Applesoft permet, entre autres choses, la création de caractères de formes et de tailles très diverses. Mais cette instruction est liée à l'existence de tables de formes (shape tables) un peu délicates à créer, comme on peut le constater en lisant le BASIC Programming Reference Manual, pages 92 à 95.

Les deux programmes détaillés ci-après sont destinés à faciliter la création de ces tables. Le premier, CONSTRUIT, rend pratique le dessin de formes et le second, ASSEMBLE, assemble les formes construites en une table de formes selon la logique requise par Applesoft. La disquette Pom's contient les fichiers suivants, dont certains ne sont pas listés ici, faute de place :

. MENU : oriente l'utilisateur entre les programmes de création graphique
. TRANSFERT : autorise un transfert rapide de graphismes
. MINUSA, MINUSB, ..., MINUSZ : formes de 26 minuscules
. TAB-MINUS : table regroupant les formes des 26 minuscules
. ESSAYE : programme démontrant l'utilisation de TAB-MINUS pour entrer directement du texte en minuscules à l'écran à partir du clavier.

Le programme CONSTRUIT

Avec ce programme, il est aisé de dessiner ou de corriger à l'écran une forme placée à l'intérieur d'une grille dont les dimensions auront été définies au préalable. Pour réaliser le dessin, 7 touches du clavier sont utilisées come suit :

H ..... déplace le curseur vers le haut
B ..... déplace le curseur vers le bas
→ ..... déplace le curseur vers la droite
← ..... déplace le curseur vers la gauche
* ou : .. enregistre le point à la position du curseur
espace .. efface le point à la position du curseur
RETURN .. indique au programme la fin du dessin

Le signe "*" dans la grille représente un point de la figure haute résolution finale.

La pression sur la touche RETURN provoque l'analyse de chaque point de l'écran (dans les limites de dimension de la grille) et la création d'un tableau de vecteurs. L'analyse est effectuée de haut en bas et de gauche à droite pour les lignes impaires, et de droite à gauche pour les lignes paires.

Afin d'éviter la génération de vecteurs inutiles, et pour économiser ainsi la place mémoire, chaque ligne ne comportant que des vecteurs de déplacement horizontal (1 ou 3) est remplacée par un vecteur unique de déplacement vers le bas (2). Ceci est réalisé tant qu'il n'y a pas eu de ligne comportant

Analyse du programme

190 à 280
Texte de présentation. Utilise la routine de cadrage située en 1950. L2S est la variable de travail pour le texte à cadrer

290 à 350
Demande la largeur LA% et la hauteur HA% du graphisme à traiter (maximum 24 points chacune). La touche RETURN appuyée deux fois consécutivement indique la fin de travail, traitée en 610.

360
Appel de la routine expliquant l'utilisation des touches du clavier en 900.

370 à 380
Définition de la ligne de grille qui sera affichée dans le cas de création d'un graphisme.

390
Débranchement vers la routine en 1480.

400 à 420
Affichage de la grille pour création d'un graphisme. Boucle sur la hauteur HA% obtenue en 290-350.

430 à 590
Routine de saisie des informations passées par l'utilisateur, que ce soit en création ou en mise à jour. Les quatre opérations de déplacement sont commandées par la flèche gauche (480-500), la lettre "B" (510), la lettre "H" (520) et la flèche droite (530). Les signes "*" (540) et ":" (550) marquent le signe "*" à la position du curseur et déplacent d'une case vers la droite. La touche d'espacement (560) met la position du

curseur à blanc (effacement) et avance le curseur. Le contrôle de la position du curseur en fonction des valeurs de HA% et LA% est réalisé dans la section 570-580. La touche RETURN (590) est utilisée pour identifier la fin de la saisie; on appelle alors la routine 1160, qui analyse l'écran, le sauve sur disquette, et revient en 240 pour demander le prochain graphisme.

600
Retour à la lecture du clavier.

610 à 740
Routine de fin de travail, qui propose d'exécuter le programme ASSEMBLE ou de clore le traitement.

750 à 890
Trois routines pour émettre des bruits : note haute, note moyenne et note basse.

900 à 1150
Routine destinée à afficher le texte explicatif sur l'utilisation des touches du clavier. L'affichage est limité à trois fois durant l'exécution du programme; tant pis pour ceux qui n'auront pas encore compris ...

1170 à 1360
La touche RETURN a été enfoncée. Ce sous-programme analyse le contenu de l'écran caractère par caractère, dans les limites de LA% et HA%. En 1180, l'indice K4 du tableau des vecteurs et l'indicateur IN du premier vecteur avec trace sont initialisés.

1220 : si la position de l'écran comporte un "*", stockage du vecteur 5, indicateur du premier vecteur avec trace trouvé et saut vers la fin de boucle.

1230 : dans le cas contraire, stockage du vecteur 1 et position de l'écran à blanc.

1250 : s'il n'y a pas de vecteur trace dans la ligne (IN=0), retour arrière de LA%-1 dans le tableau des vecteurs, et mise à 2 du vecteur du 1er caractère de la ligne.

1260 : dans le cas contraire, on ajoute 1 au dernier vecteur pour forcer le déplacement vers le bas, et 1 à l'indice de la boucle sur les lignes afin de traiter la ligne suivante (ligne paire).

1270 : si la fin de boucle est atteinte, attendre avant l'analyse.

1280-1330 : comme 1200-1270, mais pour une ligne paire, donc de droite à gauche.

1350 : fin de la boucle des lignes, donc de l'analyse de l'écran.

1355 : ajout de deux vecteurs à zéro dans le tableau.

1360 : wait avant sauvegarde. L'utilisateur doit appuyer sur une touche pour en sortir.

1370 à 1380
Demande le nom du fichier disque qui recevra le tableau des vecteurs.

1390 à 1470
Sauvegarde du tableau des vecteurs, dont la structure est la suivante : un entier

donne le nombre de vecteurs du tableau; il y a ensuite un entier par vecteur. Retour au début de programme en 240.

1490 à 1630
Demande s'il s'agit d'une création ou d'une mise à jour d'un tableau (fichier) de vecteurs existant. Retour en 400 pour une création; la mise à jour débute en 1650.

1640
Instructions de sortie du programme.

1650 à 1850
Routine de lecture des vecteurs et trace sur l'écran en mode texte selon leur contenu.

1860 à 1930
Création du tableau LI% donnant les adresses de début de chaque ligne de la page 1 en mode texte (c'est vrai, j'aurais pu les calculer ...)

Le programme ASSEMBLE

Ce programme réalise l'assemblage des formes qui dessinées, puis sauvées par CONSTRUIT.

Après avoir reçu l'adresse mémoire où sera écrite la table ($6000 est un bon choix; de toute façon, une table de formes est "relocatable"), et les noms des fichiers à prendre en compte, le programme ASSEMBLE lit ces derniers et crée une table de formes. A la fin de chaque fichier, la forme correspondante est affichée en haute résolution.

En fin de travail, la table de formes peut être sauvée sur disquette. Son contenu est ensuite affiché à l'aide de l'instruction DRAW. Le programme pousse enfin la gentillesse jusqu'à fournir à l'utilisateur les deux POKEs à introduire dans le programme appelé à utiliser la table de formes.

Remarque : il suffit de modifier les instructions du programme CONSTRUIT entre les numéros 480 et 530 pour remplacer les codes de déplacement selon les quatre directions cardinales par les codes traditionnels I, J, K et M.

Tel qu'il est conçu, ASSEMBLE est adapté à l'assemblage de formes ayant été créées sous un nom ayant une racine commune, la dernière lettre seule identifiant les formes à assembler les unes aux autres. Quand on crée avec CONSTRUIT des formes destinées à être assemblées, il faut alors prévoir de leur donner des noms adéquats. Si l'on souhaite donner aux formes initiales des noms variés, il faut indiquer lors de l'assemblage une racine vide (on répond par un RETURN à blanc).

Analyse du programme

100 à 160
Création du tableau destiné à la conversion de l'adresse de chargement de la table.

190
Fonction modulo pour la conversion décimale-hexadécimale.

200 à 260
Entrée du nombre de figures à assembler et des noms des fichiers source (SUB 1470).

270 à 290
Entrée de l'adresse mémoire (hexa) où doit être chargée la table.

300 à 330
Construction du début de la table en mémoire : nombre de définitions et index de la première définition.

340 à 380
Calcul, mise en place et affichage de l'adresse de la table dans les positions connues d'Applesoft (SE8 et SE9).

400
M = adresse de la première définition, et L = adresse où sera écrite la prochaine définition.

430 à 530
Boucle sur le nombre de figures : assemblage de la définition, calcul de l'adresse de chargement de la prochaine et affichage de la forme construite.

550 à 605
Sauvegarde (facultative) de la table sur disquette. L'adresse et la longueur sont affichées après la sauvegarde.

610 à 740
Affichage du contenu de la table.

770 à 860
Conversion hexadécimal à décimal de l'adresse entrée au clavier en 270.

870 à 900
Sous-programme de conversion.

1230 à 1420
Lecture et chargement d'un fichier source.

1470 à 1580
Entrée des noms des fichiers sources à traiter.

5000 à 5500
Routine de création des octets de la table à partir du tableau de valeurs entières créé par CONSTRUIT. Toutes les valeurs lues, comprises entre 0 et 7 inclus, sont combinées pour former les octets de la table selon la logique de l'Applesoft.

7000 à 7090
Traitement de fin de programme.

```
]LOAD CONSTRUIT
]LIST

100  REM **********************
          *                    *
          * CREATION GRAPHISME *
          *  HAUTE RESOLUTION  *
          *                    *
          *    08 FEVRIER 82   *
          *                    *
          **********************
110  REM
120  DIM TB%(600)
130  REM   DEF NUMEROS DE LIGNE
140  GOSUB 1860
150  SPEED= 255
160  TEXT : HOME
170  GOSUB 2080
180  REM
190  REM ---------------------
          P R O L O G U E
          ---------------------
200 L2$ = "VOUS DESIREZ CREER UN
     DESSIN EN"
210 VT = 2: GOSUB 1950
220 L2$ = "H A U T E   R E S O L
     U T I O N"
230 VT = 4: INVERSE : GOSUB 1950:
     NORMAL
240 L2$ = "ENTREZ LA LARGEUR ET L
     A HAUTEUR"
250 VT = 8: GOSUB 1950
260 L2$ = "SOUS LA FORME :  LL,HH
     "
270 VT = 10: GOSUB 1950
280 L2$ = "(VALEURS MAXI RESPECTI
     VES = 24 ET 24)"
290 VT = 12: GOSUB 1950
295  VTAB 21: PRINT "     (2 FOIS
     RETURN POUR TERMINER)
     "
300  VTAB VT + 3: HTAB 16
310  INPUT "=>";LARGE$,HAUT$
320  IF  LEN (LARGE$) = 0 THEN
     GOTO 610
330  IF  VAL (LARGE$) < 1 OR  VAL
     (LARGE$) > 24 THEN  GOSUB 75
     0: GOSUB 800: GOTO 160
340  IF  VAL (HAUT$) < 1 OR  VAL
     (HAUT$) > 24 THEN  GOSUB 750
     : GOSUB 800: GOTO 160
350 LA% =  VAL (LARGE$):HA% =  VAL
     (HAUT$)
360  GOSUB 900: REM EXPLIC.
370  REM
380 LI$ =  LEFT$ (".............
     .........",LA%)
390  HOME : GOTO 1480
400  FOR NL = 1 TO HA%
410  VTAB NL: HTAB 1: PRINT LI$;
420  NEXT NL
430  REM
440  REM **********************
          *                    *
          *    REMPLISSAGE     *
          *                    *
          **********************
450  LET VT = 1:HT = 1
460  VTAB VT: HTAB HT
470  GET RP$
480  IF  ASC (RP$) = 8 AND HT > 1
      THEN HT = HT - 1: GOTO 460
490  IF  ASC (RP$) = 8 AND HT = 1
      AND VT = 1 THEN  GOTO 460
500  IF  ASC (RP$) = 8 AND HT = 1
      THEN VT = VT - 1: GOTO 460
510  IF  ASC (RP$) = 66 THEN VT =
     VT + 1
520  IF  ASC (RP$) = 72 AND VT >
     1 THEN VT = VT - 1
530  IF  ASC (RP$) = 21 THEN HT =
     HT + 1
540  IF RP$ = "*" THEN  PRINT RP$
     ;:HT = HT + 1
550  IF RP$ = ":" THEN  PRINT "*"
     ;:HT = HT + 1
560  IF RP$ = " " OR RP$ = "."
     THEN PRINT RP$;:HT = HT + 1
570  IF HT > LA% THEN HT = 1:VT =
     VT + 1
580  IF VT > HA% THEN VT = 1:HT =
     1: GOTO 460
590  IF  ASC (RP$) = 13 THEN
     GOSUB 1160: HOME : GOTO 240
600  GOTO 460
610  REM **********************
          *                    *
          *  FIN DE TRAVAIL    *
          *                    *
          **********************
620  HOME
630 VT = 10
640  VTAB 7: PRINT "   ENTREZ : "

650 L2$ = "1   SI VOUS VOULEZ ESS
     AYER LES DESSINS"
660  GOSUB 1950
670 VT = 12:L2$ = "2   SI VOUS VO
     ULEZ TOUT ARRETER          "
680  GOSUB 1950
690  VTAB 16: GET ZZ$
700  IF ZZ$ = "2" THEN  GOTO 1640

710  IF ZZ$ <  > "1" THEN 620
720 D$ =  CHR$ (4): PRINT
730  PRINT D$;"RUN ASSEMBLE"
740  END
750  REM <<<<<NOTE HAUTE>>>>>
760  POKE 769,30
770  POKE 768,80
```

```
780  CALL 770
790  RETURN
800  REM <<<<<NOTE MOYENNE>>>>>
810  POKE 769,30
820  POKE 768,130
830  CALL 770
840  RETURN
850  REM <<<<<NOTE BASSE>>>>>
860  POKE 769,30
870  POKE 768,180
880  CALL 770
890  RETURN
900  REM
910  REM **********************
          *                    *
          * TOUCHES FONCTIONS *
          *                    *
          **********************
920  REM
930 SX = SX + 1: IF SX > 3 THEN
    RETURN  940  HOME
950  FLASH : VTAB 2: PRINT "ATTEN
    TION !"
960  NORMAL :VT = 5:L2$ = "NOTEZ
    BIEN L'UTILISATION DES TOUCH
    ES :"
970  GOSUB 1950
980  VTAB 8
990  PRINT "->  (FL.DROITE)   =
       CURSEUR A DROITE";
1000  VTAB 10
1010  PRINT "<-  (FL.GAUCHE)   =
       CURSEUR A GAUCHE";
1020  VTAB 12
1030  PRINT "H   (LETTRE  H)   =
       CURSEUR EN HAUT ";
1040  VTAB 14
1050  PRINT "B   (LETTRE  B)   =
       CURSEUR EN BAS  ";
1060  VTAB 16
1070  PRINT ":   (2  POINTS)   =
       POSITION REMPLIE";
1080  VTAB 18
1090  PRINT "ESPACE OU '*'   =
      POSITION BLANCHE";
1100 VT = 20
1110 L2$ = "<RETURN> QUAND C'EST
    FINI.": GOSUB 1950
1120  VTAB 24
1130  HTAB 32: PRINT "<RETURN>";
1140  GET Z$
1150  RETURN
1160  REM
1170  REM **********************
          *                    *
          * ANALYSE  D'ECRAN *
          *                    *
          **********************
1180  LET K4 = 0:IN = 0
1190  FOR A = 1  TO HA%
1200  FOR CA = 0 TO (LA% - 1)
1210  LET K4 = K4 + 1
1220  IF  PEEK (LI%(A) + CA) = 17
    0 THEN TB%(K4) = 5:IN = 1:
     GOTO 1240
1230 TB%(K4) = 1: POKE (LI%(A) +
    CA),160
1240  NEXT CA
1250  IF IN = 0 THEN K4 = K4 - (L
    A% - 1):TB%(K4) = 2: GOTO 13
    50
1260 TB%(K4) = TB%(K4) + 1:A = A +
    1
1270  IF A > HA% THEN 1355
1280  FOR CA = (LA% - 1) TO 0 STEP
    - 1
1290  LET K4 = K4 + 1
1300  IF  PEEK (LI%(A) + CA) = 17
    0 THEN TB%(K4) = 7:IN = 1:
     GOTO   1320
1310 TB%(K4) = 3: POKE (LI%(A) +
    CA),160
1320  NEXT CA
1330 TB%(K4) = TB%(K4) - 1
1350  NEXT A
1355 K4 = K4 + 1:TB%(K4) = 0:K4 =
    K4 + 1:TB%(K4) = 0
1360  GET RP$
1370  HOME :VT = 12:L2$ = "NOM DU
     FICHIER :": GOSUB 1950
1380  VTAB 14: HTAB 15: INPUT "";
    NM$
1385  IF NM$ = "" THEN 240
1390 D$ =  CHR$ (4)
1400  PRINT D$;"OPEN ";NM$
1410  PRINT D$;"WRITE ";NM$
1420  LET K4% = K4: PRINT K4%
1430  FOR A = 1 TO K4%
1440  PRINT TB%(A)
1450  NEXT A
1460  PRINT D$;"CLOSE ";NM$
1470  HOME : GOTO 240
1480  REM
1490  REM --------------------
           CREE OU MET A JOUR
          --------------------
1500  HOME :VT = 12
1510 VT = 12
1520 L2$ = "CREATION OU MISE A JO
    UR ?"
1530  GOSUB 1950: VTAB 14: HTAB 1
    7
1540  GET Z$
1550  IF Z$ = "C" THEN  HOME :
     GOTO  400
1560  IF Z$ <  > "M" THEN 1500
1570  HOME :VT = 12
1580 L2$ = "NOM DU FICHIER ?":
     GOSUB  1950
1590  VTAB 14: HTAB 16: INPUT "";
    NM$
```

```
1595  IF NM$ = "" THEN 1500
1600  HOME : VTAB 23: HTAB 38:
      PRINT  "";
1610 D$ =  CHR$ (4)
1620  GOTO 1650
1630  REM  =========F=I=N========
      ==
1640  HOME :L2$ = "A BIENTOT ..."
      :VT = 12: GOSUB 1950: VTAB 2
      4: END
1650  REM ------------------
          MISE A JOUR DE GRAF
          ------------------


1660  PRINT D$;"OPEN ";NM$
1670  PRINT D$;"READ ";NM$
1680  INPUT K4%
1690  LET CA = 0:LI = 1
1700  FOR X = 1 TO K4%
1710  INPUT TB%: IF TB% = 0 THEN
      1770
1720  IF TB% > 3 THEN  POKE (LI%(
      LI) + CA),170: GOTO 1740
1730  POKE (LI%(LI) + CA),174
1740  ON (TB% + 1) GOTO 1760,1800
      ,1820,1840,1760,1800,1820,18
      40
1750  STOP
1760 LI = LI - 1: IF LI < 1 THEN
       GOSUB 750: GOSUB 750:LI = 1


1770  NEXT X
1780  PRINT D$;"CLOSE ";NM$
1790  GOTO 430
1800 CA = CA + 1: IF CA > LA% THEN
       GOSUB 800: GOSUB 800:CA = L
      A% - 1
1810  GOTO 1770
1820 LI = LI + 1: IF LI > HA% + 1
       THEN  GOSUB 850: GOSUB 850:
      LI = HA% + 1
1830  GOTO 1770
1840 CA = CA - 1: IF CA < 0 THEN
       CALL  - 1052: CALL  - 1052:
      CA = 0
1850  GOTO 1770
1860  REM
1870  REM NUMEROS DES LIGNES
1880  REM
1890  DATA 1024,1152,1280,1408,15
      36,1664,1792,1920,1064,1192,
      1320,1448,1576,1704,1832,196
      0,1104,1232,1360,1488,1616,1
      744,1872,2000
1900  DIM LI%(24)
1910  FOR LI = 1 TO 24
1920  READ LI%(LI)
1930  NEXT LI
1940  RETURN
1950  REM ********************
          *                  *
          * CENTRAGE DE L2$ *
          *                  *
          ********************
1960  LET X =  FRE (0): REM FREE
      MEM
1970  IF  LEN (L2$) > 39 THEN
       PRINT  "TROP LONG ..."
1980 HT% = (40 -  LEN (L2$)) / 2
1990  REM  FIN CENTRAGE TITRE
2000  VTAB VT: REM  FOURNI
2010  HTAB HT%: REM CALCULE
2020  PRINT L2$: NORMAL
2030  RETURN
2040  FLASH : PRINT "LONGUEUR L2$
      > 40"
2050  PRINT L2$: NORMAL : STOP
2060  FLASH : PRINT "HTAB = ";HT%

2070  GOTO 2050
2080  REM
2083  REM ROUTINE SON
2085  REM
2090  POKE 770,173: POKE 771,48: POKE
      772,192: POKE 773,136: POKE
      774,208: POKE 775,5: POKE 77
      6,206: POKE 777,1: POKE 778,
      3: POKE 779,240: POKE 780,9:
       POKE 781,202
2100  POKE 782,208: POKE 783,245:
       POKE 784,174: POKE 785,0: POKE
      786,3: POKE 787,76: POKE 788
      ,2: POKE 789,3: POKE 790,96:
       POKE 791,0: POKE 792,0
2110  RETURN
]LOAD ASSEMBLE
]LIST

10  LOMEM: 16384
12  REM  **********************
         *                    *
         *  TABLE DE FORMES   *
         *                    *
         **********************


19  SCALE= 1
20  DIM TP%(600)
40  TEXT : POKE 34,0
50  HOME : VTAB 1: HTAB 12
55  INVERSE
60  PRINT "  ASSEMBLAGE  ": NORMAL

70  POKE 34,2
90  REM
100  REM
110  DIM HEX$(16,2)
120  FOR I = 1 TO 16
130  FOR J = 1 TO 2
```

```
140  READ HEX$(I,J)
150  NEXT J
160  NEXT I
190  DEF  FN MOD(A) =  INT ((A /
     256 -  INT (A / 256)) * 256 +
     .05) *  SGN (A / 256)
200  REM
210  REM **********************
          *                    *
          * DEBUT DU PROGRAMME *
          *                    *
          **********************
220  VTAB 4: CALL  - 958
230  INPUT "COMBIEN DE FIGURES A
     ASSEMBLER ? ";L$
240  LET NN =  VAL (L$)
245  IF NN < 1 THEN  GOTO 7000
250  GOSUB 1470
260  PRINT
270  INPUT "ADRESSE HEXA = ";L$
280  GOSUB 770: PRINT
290  LET L =  VAL (L$): REM ADDR
300  POKE L,NN: POKE (L + 1),0
310  LET LL = 4 + (NN - 1) * 2
320  GOSUB 870
330  POKE (L + 2),L2: POKE (L + 3
     ),L1
340  LET LL = L: GOSUB 870
350  POKE 232,L2: POKE 233,L1
355  HOME : PRINT : PRINT : HTAB
     5: PRINT "AJOUTEZ DANS VOTRE
      PROGRAMME :": PRINT
360  PRINT : PRINT  TAB( 08):
     INVERSE
370  PRINT "POKE 232,";L2;": POKE
      233,";L1
380  NORMAL : PRINT :M =  FRE (0)

390  GET A$: PRINT : HGR
400  LET M = L:L = L + 2 + NN * 2

410  VTAB 21
420  REM **********************
          *                    *
          * DEBUT DE LA BOUCLE *
          *                    *
          **********************
430  FOR N1 = 1 TO NN
440  GOSUB 1230
470  GOSUB 5000
500  PRINT "FIN DE LA FIGURE ";N1
     : POKE L,0
510  IF N1 <  > NN THEN MM = M +
     2 + N1 * 2:LL = L - M + 1:
     GOSUB 870: POKE MM,L2: POKE
     (MM +  1),L1
520  LET L = L + 1
523  HGR
524  DRAW N1 AT 140,80
530  NEXT N1
540  HOME : VTAB 21
550  REM
560  REM **********************
          *                    *
          * SAUVEGARDE TABLE   *
          *                    *
          **********************
570  PRINT "VOULEZ-VOUS SAUVER LA
      TABLE ? "
580  GET A$
590  IF A$ <  > "O" AND A$ <  > "
     N" THEN 540
600  IF A$ = "O" THEN  INPUT "QUE
     L NOM ? ";N$:N$ = "BSAVE" +
     N$ + ",A" +  STR$ (M) + ",L"
      +  STR$ (L - M): PRINT : PRINT
      CHR$ (4);N$
605  PRINT N$: GET A$
610  REM
620  REM **********************
          *                    *
          *  ESSAI  FIGURES    *
          *                    *
          **********************
630  HOME : VTAB 21: PRINT "ESSAI
      OES FIGURES.";
660  HGR : HCOLOR= 3
670  ROT= 0:X2 = 0:Y2 = 20
680  INPUT " LARGEUR = ";LA
690  FOR N1 = 1 TO NN
700  DRAW N1 AT X2,Y2
720 X2 = X2 + LA: IF X2 > 260 THEN
     X2 = 0:Y2 = Y2 + 14
730  NEXT N1
732  POKE 34,0
740  GOTO 7000
760  REM
770  REM --------------------
          CONVERSION  ADRESSE
          --------------------
780  FOR K = 1 TO  LEN (L$)
790  LET J$ =  MID$ (L$, LEN (L$)
      - K + 1,1)
800  FOR I = 1 TO 16
810  IF J$ = HEX$(I,1) THEN J =  VAL
     (HEX$(I,2)): GOTO 830
820  NEXT I
830  LET M = M + 16 ^ (K - 1) * J

840  NEXT K
850  LET L$ =  STR$ (M)
860  RETURN
870  REM
880  REM --------------------
          CONV DEC-->HEX (POKE)
          --------------------

890  LET L1 =  INT (LL / 256):L2 =
      FN MOD(LL)
900  RETURN
```

```
1230  REM **********************
      *                      *
      * LECTURE AUTOMAT. *
      *                      *
      **********************
1310  LET NM$ = NM$(N1)
1320  LET D$ =  CHR$ (4)
1330  PRINT D$;"OPEN ";NM$
1340  PRINT D$;"READ ";NM$
1350  INPUT C2%
1360  FOR C1 = 1 TO C2%
1370  INPUT TP%(C1)
1380  NEXT C1
1390  PRINT D$;"CLOSE ";NM$
1400  LET TP%(C1) = 0
1405  LET TP%(C1 + 1) = 0
1406  LET TP%(C1 + 2) = 0
1407  LET TP%(C1 + 3) = 0
1410 C1 = 1
1420  RETURN
1470  REM
1480  REM **********************
      *                      *
      * LECTURE DES NOMS *
      * DES FICH.SOURCES *
      *                      *
      **********************
1490  DIM NM$(NN)
1500  HOME : INPUT "PREFIXE : ";P
      F$
1505  IF  LEN (PF$) = 0 THEN LE =
      1: VTAB 24: HTAB 31: PRINT "
      RETURN": VTAB 5: HTAB 1: GOTO
      1520
1510  LET LE = 2
1520  FOR NM = 1 TO NN
1530  PRINT "NOM ";NM;" ";PF$;
1540  IF LE = 1 THEN  INPUT "";NM
      $
1550  IF LE = 2 THEN  GET NM$: PRINT
      NM$
1560 NM$(NM) = PF$ + NM$
1570  NEXT NM
1580  RETURN
5000  REM **********************
      *                      *
      *  TABLE DE FORMES  *
      *                      *
      **********************
5010 NB = 1: REM NUMERO VECTEUR
5020  REM

        LECTURE/CODIFICATION

5025 VE$ =  STR$ (TP%(C1)):C1 = C
      1 + 1
5040  IF VE$ < "0" OR VE$ > "7" THEN
      5020
5050 VE(NB) =  VAL (VE$)
5060  ON NB GOTO 5080,5140,5210
5070  REM
5080  REM 1ER VECTEUR
5090  REM
5100 VE%(1) = VE(1)
5110 NB = NB + 1:VE(1) = 0
5120  GOTO 5020
5130  REM
5140  REM 2EME VECTEUR
5150  REM
5160 VE%(2) = VE%(1) + (VE(2) * 8
      )
5170  IF VE%(2) = 0 THEN 5330
5180 NB = NB + 1:VE(1) = 0:VE%(1)
      = 0
5190  GOTO 5020
5200  REM
5210  REM 3EME VECTEUR
5220  REM
5230  IF VE(3) <  > 0 AND VE(3) <
      4 THEN 5280
5240 VE(1) = VE(3):VE(3) = 0
5250  POKE L,VE%(2):L = L + 1
5260 NB = 1
5270  GOTO 5060
5280 VE%(3) = VE%(2) + (VE(3) * 6
      4)
5290  IF VE%(3) = 0 THEN 5330
5300  POKE L,VE%(3):L = L + 1
5310 NB = 1:VE%(3) = 0:VE%(2) = 0
      :VE%(1) = 0:VE(1) = 0:VE(2) =
      0:VE(3) = 0: GOTO 5020
5320  REM
5330  REM FIN DE FORME
5340  REM
5360  REM VE%(3) CONTIENT VECTEUR

5370  POKE L,VE%(3):L = L + 1
5375  POKE L,0: POKE L + 1,0: POKE
      L + 2,0: POKE L + 3,0:L = L +
      1
5380  RETURN
5500  REM *****
6000  DATA   0,0,1,1,2,2,3,3,4,4,
      5,5,6,6,7,7,8,8,9,9,A,10,B,1
      1,C,12,D,13,E,14,F,15
7000  REM
7005  REM
7010  HOME : VTAB 21
7020  PRINT "  1  POUR CONSTRUIRE
      "
7030  VTAB 23
7040  PRINT "  2  POUR ARRETER  "
      ;
7050  GET Z$
7060  IF Z$ = "2" THEN  HOME : END
7070  HOME : PRINT
7075  IF Z$ <  > "1" THEN 7010
7080 D$ =  CHR$ (4)
7090  PRINT D$;"RUN CONSTRUIT"
```

```
]LOAD ESSAYE
]LIST

300  HOME : HGR2 : HCOLOR= 3
320  PRINT  CHR$ (4);"BLOAD TAB-M
     INUS,A$6000"
330  POKE 232,0: POKE 233,96: ROT=
     0: SCALE= 1
340 L2$ = "QUAND CE TEXTE SERA EF
     FACE VOUS POURREZ ENTRER": GOSUB
     2000
350 L2$ = "DES CARACTERES ALPHABE
     TIQUES AU CLAVIER": GOSUB 20
     00
360 L2$ =  CHR$ (13): GOSUB 2000
370 L2$ =  CHR$ (13): GOSUB 2000
380 L2$ = " JE NE DISPOSE PAS ENC
     ORE  DES CHIFFRES NI DES": GOSUB
     2000
390 L2$ = "CARACTERES  MINUSCULES
      ACCENTUES  MAIS IL PEUT": GOSUB
     2000
400 L2$ = "ETRE INTERESSANT DE LE
     S CREER AVEC  CONSTRUIT ": GOSUB
     2000
410  FOR WAI = 1 TO 5000: NEXT WA
     I
420  HGR2 :X = 0:Y = 0
430 L2$ = ""
440  GET L2$
450  GOSUB 2000
460  GOTO 440
1999  END
2000  REM
2020  FOR A = 1 TO  LEN (L2$)
2030 N = ( ASC ( MID$ (L2$,A,1)))

2040  IF N = 13 THEN 2230
2050  IF N = 32 THEN X = X + 6: GOTO
     2200
2060  IF N - 64 > 26 OR N - 64 <
     1 THEN  CALL  - 1052: CALL  -
     1052: RETURN
2070  DRAW N - 64 AT X,Y
2080 X = X + 6
2200  IF X > 273 THEN X = 0:Y = Y
      + 9
2210  IF Y > 183 THEN  TEXT : HOME
     : END
2220  NEXT A
2225  IF  LEN (L2$) = 1 THEN  RETURN

2230 X = 0:Y = Y + 9: IF Y > 183 THEN
      TEXT : HOME : END
2250  RETURN

]LOAD MENU
]LIST

10  HOME : VTAB 3:D$ =  CHR$ (4)
20  INVERSE : PRINT "PHEBUS";:
    NORMAL: PRINT " VOUS PROPOSE :"
30  VTAB 8: GOSUB 200: VTAB 8:
    CALL - 958
40  PRINT "1  CONSTRUIT   POUR CR
     EER VOS GRAPHISMES";
50  VTAB 11
60  PRINT "2  ASSEMBLE    POUR LE
     S ASSEMBLER (DRAW)";
70  VTAB 20
90  PRINT "   (ENTREZ LE NUMERO D
     E VOTRE CHOIX)  ";
100  GET Z$: PRINT
110  IF Z$ = "1" THEN  PRINT D$;"
     RUN CONSTRUIT"
120  IF Z$ = "2" THEN  PRINT D$;"
     RUN ASSEMBLE"
130  PRINT D$;"CATALOG "
140  GET Z$
150  GOTO 10
200  REM
240  PRINT "    4 PROGRAMMES :
                    "
250  VTAB 11
260  PRINT " CONSTRUIT  POUR SAIS
     IR DES GRAPHISMES "
270  PRINT
280  PRINT " ASSEMBLE   POUR CREE
     R UNE SHAPE TABLE "
290  PRINT
300  PRINT " ESSAYE     POUR ESSA
     YER UNE TABLE      "
310  PRINT
320  PRINT " TRANSFERT  POUR COPI
     ER DES SOURCES SUR"
325  PRINT
330  PRINT "                    U
     NE AUTRE DISQUETTE"
350  VTAB 23
360  PRINT "
              <RETURN>";
370  GET Z$: PRINT
400  RETURN
60000  REM
60002  REM CREE LE 20/02/81
60004  REM
60006  REM DATES DE MISE A JOUR
60010  REM
60012  REM 15/03/82
60014  REM  18/03/82
]LOAD TRANSFERT
]LIST

5  REM **********************
      *                    *
      * TRANSFERT   SOURCES *
      *                    *
      *      CONSTRUIT     *
      *******          *******
10  SPEED= 255: TEXT
```

```
20  POKE 33,40
90  REM ====================
100  HOME : VTAB 12: INVERSE
110 L2$ = "TRANSFERT GRAPHISMES"
115  HTAB (40 -  LEN (L2$)) / 2: PRINT
    L2$
130  FOR WAI = 1 TO 1300: NEXT WA
    I
180  NORMAL : VTAB 15: HTAB 10: INPUT
    "COMBIEN ? ";CX%: DIM NM$(CX
    %),NB(CX%)
289  REM
290  REM ====================
291  REM
300  HOME
310  INPUT "LONGUEUR,HAUTEUR (MAX
    I) ";LO%,HA%
315  PRINT
320 NN = LO% * HA%:NN = NN * CX%:
     PRINT "DIM = ";NN: GET A$
330  DIM TB%(NN)
332  PRINT
335 D$ =  CHR$ (4)
340  PRINT CX%;" NOMS : ": FOR T =
    1 TO CX%: INPUT NM$(T): NEXT T
343  PRINT
345 RL = 1: REM ENTREE DANS TB%
350  FOR IN = 1 TO CX%
370  PRINT D$;"OPEN ";NM$(IN)
380  PRINT D$;"READ ";NM$(IN)
390  INPUT NB(IN)
400  FOR Z = 1 TO NB(IN)
410  INPUT TB%(RL):RL = RL + 1
420  NEXT Z
430  PRINT D$;"CLOSE "NM$(IN)
440  NEXT IN
445  PRINT
450  FOR T = 1 TO CX%
460  PRINT NM$(T);" A ";NB(T);" E
    NTREES."
470  NEXT T
480  PRINT
490  PRINT "CHANGEZ LA DISQUETTE
    PUIS <RETURN> ";: GET A$
650  REM
660  REM **********************
          *                    *
          * SAUVE LE TABLEAU   *
          *                    *
          **********************
 668 RL = 1:D$ =  CHR$ (4): PRINT
 670  FOR IN = 1 TO CX%
 690  PRINT D$;"OPEN ";NM$(IN)
 700  PRINT D$;"WRITE ";NM$(IN)
 710  PRINT NB(IN)
 720  FOR I = 1 TO NB(IN)
 790  PRINT TB%(RL):RL = RL + 1
 800  NEXT I
 810  PRINT D$;"CLOSE ";NM$(IN)
 840  NEXT IN
 860  END
```

TAB-MINUS

```
6000- 1A 00 36 00 46 00 59 00
6008- 69 00 7C 00 8D 00 9F 00
6010- AF 00 C1 00 D3 00 E4 00
6018- F6 00 09 01 19 01 29 01
6020- 39 01 49 01 59 01 68 01
6028- 78 01 8A 01 9A 01 A9 01
6030- BA 01 CA 01 DA 01 52 2D
6038- 15 DF 53 2D 35 DF 73 2D
6040- F5 DB 02 00 00 00 4D D1
6048- DB 2E 2D 15 DF 33 4D 31
6050- DF 33 2D AD DB 13 00 00
6058- 00 52 2D 15 DF 33 4D 11
6060- DF 73 2D D5 DB 02 00 00
6068- 00 49 31 DF 53 2D 35 DF
6070- 33 4D 31 DF 73 2D F5 DB
6078- 02 00 00 00 52 2D 15 DF
6080- 33 2D 2D DE 1B 0E 2D D5
6088- DB 02 00 00 00 29 8D FB
6090- 33 4D D1 3B 37 4D D1 DB
6098- 6E 89 DB 13 00 00 00 52
60A0- 2D 35 DF 33 4D 31 3F BF
60A8- 49 F1 3F 17 00 00 00 4D
60B0- D1 DB 6E 89 3B 3F 6E 09
6068- FE 1B 6E 09 DE 9B 00 00
60C0- 00 09 8D DB 53 6D DA 9F
60C8- 09 8D 1B 9F 29 AD DB 13
60D0- 00 00 00 49 D5 DB 4A AD
60D8- FB 53 09 D5 DF 6A A9 1B
60E0- BF 00 00 00 69 D1 1B 57
60E8- 4D 1A 1F 57 6D 1A 1F 57
60F0- 4D DE 9B 00 00 00 29 8D
60F8- 1B 9F 09 8D 1B 9F 09 8D
6100- 1B 9F 29 AD DB 13 00 00
6108- 00 52 0D 15 1F 1F 6E 0D
6110- FE 1F 6E 0D DE 9B 00 00
6118- 00 12 0D AD DF 37 4D 31
6120- DF 33 4D F1 DB 02 00 00
6128- 00 52 2D 15 DF 33 4D 31
6130- DF 73 2D D5 DB 02 00 00
6138- 00 12 2D AD DF 33 4D F1
6140- 3F 37 4D D1 DB 06 00 00
6148- 00 52 2D 35 DF 33 4D 31
6150- 3F BF 49 31 DF 13 00 00
6158- 00 12 0D AD DF 37 4D D1
6160- DB 6E 89 DB 13 00 00 00
6168- 52 2D F5 DB 0E 2D 15 DF
6170- 13 2D AD DB 13 00 00 00
6178- 69 D1 1B 17 2D AD DB 57
6180- 4D 1A 1F 57 69 DA 9B 00
6188- 00 00 12 4D 31 DF 33 4D
6190- 31 DF 73 2D F5 DB 02 00
6198- 00 00 12 4D 31 DF 33 4D
61A0- F1 1F 57 69 DA 9B 00 00
61A8- 00 12 4D 31 DF 33 0D 0D
61B0- FE 1F 0E 00 D5 DB 02 00
61B8- 00 00 12 4D F1 1F 57 69
61C0- 1A 1F 17 4D F1 DB 02 00
61C8- 00 00 12 4D 31 DF 73 2D
61D0- 35 DF 53 49 1E 3F 17 00
61D8- 00 00 12 2D 2D 1E DF 4A
61E0- 8D DB 17 2D 2D DE 9B 00
61E8- 00 00 FF FF
```

La carte M/DOS 6502 à l'essai

Nous en avions l'intention depuis longtemps, à force d'en entendre parler; nous avons enfin pu essayer la carte M/DOS 6502 fabriquée par Micro Informatique Service (MIS), à Nice.

Vendue au prix public de 2.800 F HT, la carte M/DOS 6502 est compatible Apple, ITT 2020, Silex et systèmes intégrés IEF. Ses caractéristiques principales sont les suivantes :

. 16K supplémentaires de mémoire centrale;
. un système d'exploitation de disquettes (SED) puissant;
. une gestion par masques des entrées et des sorties;
. précision numérique de 48 chiffres significatifs en traitement particulier;
. adaptabilité à toutes les cartes 80 colonnes compatibles Apple/ITT 2020;
. compatibilité des programmes de 110K à 40 méga-octets;
. jusqu'à 16 méga-octets peuvent être gérés d'un seul tenant.

La conception originale du SED et la gestion par masques des entrées et des sorties permettent de gagner beaucoup de temps dans la réalisation d'un programme. Nous l'avons constaté par nous-mêmes en mettant des programmes au point sur M/DOS 6502; cela nous a été confirmé par des SSCI qui utilisent aussi la carte.

La gestion de fichier

La gestion de fichier est très différente de celle du DOS, tout d'abord par la nature même des fichiers utilisés, dont on distingue trois types :

. les fichiers séquentiels relatifs;
. les fichiers séquentiels indexés;
. les fichiers multiclés.

Les fichiers séquentiels relatifs (FSR) sont semblables aux fichiers séquentiels du DOS.

Les fichiers séquentiels indexés (FSI) possèdent une clé d'accès unique formée d'un ensemble de variables BASIC. Quand un enregistrement est recherché, ou bien il est trouvé et tout va bien, ou bien il ne l'est pas et un drapeau signale l'échec de la recherche.

Les fichiers multiclés (FM) possèdent au maximum 10 clés, chacune de celles-ci étant soit un ensemble de variables BASIC, soit un pointeur de fichier relatif.

Une grande originalité du M/DOS 6502 est la gestion dynamique des fichiers. Il n'est pas besoin de définir une longueur d'enregistrement précise. Quand un fichier de type FSI ou FM est créé, le système propose une longueur indicative de référence qui sert à sa gestion du fichier, et que l'utilisateur peut modifier à loisir. Le SED gère par lui-même la longueur d'enregistrement.

Quel que soit le type de fichier, lorsqu'un enregistrement recherché a été trouvé, cet enregistrement est lu automatiquement. Les opérations possibles à partir de là correspondent aux ordres suivants :

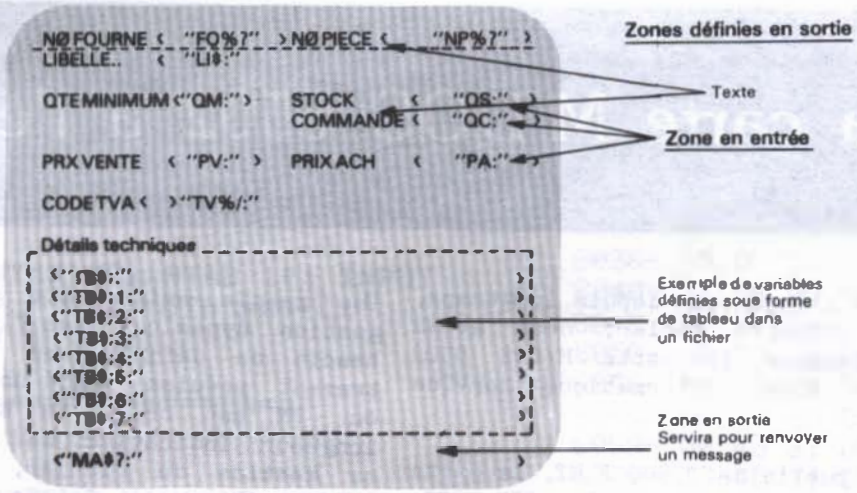
NEXT - lit l'enregistrement suivant selon les clés ou le pointeur (s'il s'agit d'un fichier multiclés, l'utilisateur sélectionne la clé)
BORNE - fixe une limite à la clé du fichier (FSR, FSI, FM)
XTRACT - sélectionne un sous-fichier par sélection d'une sous-zone dans la clé (FSI, FM)
XINDEX - met dans le pointeur d'un FSR le numéro du prochain enregistrement qui sera créé par un ordre WRITE (FSR)
WRITE - écriture d'un nouvel enregistrement (tous fichiers); refuse les homonymes
ADD - écriture d'un nouvel enregistrement (FSI, FM); accepte les homonymes
UPDATE - mise à jour d'un enregistrement (tous fichiers)
READ - lecture d'un enregistrement (tous fichiers)
DELET - destruction d'un enregistrement (tous fichiers)

Le M/DOS 6502 étant résident sur la carte, la place que le DOS occupe sur les disquettes (si l'on désire booter sans jongler avec les disquettes) est libérée.

La gestion par masque des entrées/sorties

Il s'agit là d'une idée très intéressante; seuls ceux qui n'ont jamais travaillé à l'aide de masques peuvent ne pas réaliser l'apport de ceux-ci. Un des premiers programmes que j'ai écrits quand j'ai commencé à travailler sur Apple est un programme de gestion des menus et des entrées par masque sur l'écran.

Pour définir un masque de saisie en M/DOS 6502, il suffit (une fois l'éditeur de masques chargé) de rentrer à l'écran les libellés exactement où on désire les voir apparaître. Pour cela, il suffit d'utiliser



EXEMPLE DE MASQUE E/S

les instructions de déplacement du curseur à l'écran (flèches, barre d'espacement, RETURN). Ensuite, les endroits où doivent être rentrées les données sont délimités avec les signes "<" et ">". Le nom de la donnée correspondante est inscrit entre les deux, quand il y a suffisamment de place, à côté autrement.

L'utilisation de mémoires de masse

M/DOS 6502 a été conçu de façon que l'adaptation d'un programme à l'utilisation de mémoires de masse (disquette grande capacité, disque lourd) ne nécessite aucune modification. Il est possible d'écrire le programme de manière à le rendre immédiatement opérationnel sur divers supports de ce genre. C'est pourquoi MIS, ainsi que d'autres fabricants ou distributeurs, propose des systèmes M/DOS 6502 multipostes. Christian Colmant en parle dans son article sur les mémoires de masse.

Commentaires généraux

Quelles sont les critiques que l'on peut émettre à l'égard de la carte M/DOS 6502 ?

Inconvénient 1
Le tout premier est que, et il s'agit là d'un défaut hélas bien français, la documentation est franchement mauvaise; quand au texte, il pourrait être plus clair. Les fautes de français courent après celles d'orthographe. Alors que le M/DOS 6502 s'adresse aux utilisateurs Apple, aucune référence n'est faite, lors de la présentation des divers types de fichiers, au vocabulaire du DOS ou aux notions courantes pour les Appleomanes.

Inconvénient 2
La place que l'on est censé gagner sur la disquette en n'y chargeant pas de SED est reperdue par les fichiers que M/DOS 6502 doit mettre en place pour effectuer sa gestion dynamique de la mémoire. En effet, la gestion de fichier est prévue pour des fichiers de 16 M-octets et 64K enregistrements.

Inconvénient 3
Les masques d'entrées/sorties gèrent divers formats et/ou modes de représentation : alphanumérique, entier, réel, gestion (x décimales). Il est regrettable qu'il n'y ait pas pour l'utilisateur la possibilité de définir deux ou trois formats complémentaires dont il a souvent besoin. Par exemple, il est utile en gestion d'avoir le format xxx.xxx.xxx,xx dans lequel les grandes sommes sont rendues lisibles par le découpage en tranches de trois chiffres. Note : un contrôle interzone conditionnel est annoncé pour la release XI.

Inconvénient 4
La carte M/DOS 6502 n'est pas compatible avec les cartes langage possédant la ROM F8, ce qui est gênant pour les utilisateurs de Pascal ou ceux qui ne peuvent se satisfaire des maigres 18K disponibles avec Visicalc 3.3. MIS a, parait-il, l'intention de modifier légèrement sa carte de façon à rendre la carte langage prioritaire, ce qui éliminera ce problème.

Inconvénient 5
La carte actuelle consomme trop et, lorsque nous avons voulu l'utiliser avec le lecteur 1M d'IEF (voir l'article sur les mémoires de masse), nous n'arrivions plus à booter. Il a fallu changer des ROMs de l'Apple et y installer de plus puissantes pour enfin pouvoir travailler. Il est vrai que, si l'Apple comporte huit slots, cela ne veut pas dire que l'appareil fonctionne si on les utilise tous ...

Après ces critiques, quels sont les aspects positifs que nous pouvons souligner ? Nous reprenons ici ce qui nous a particulièrement intéressés sans énumérer tous les avantages annoncés au début de cet article. Nous n'avons pas eu l'occasion d'utiliser M/DOS 6502 en multi-poste et ne pouvons donc faire aucun commentaire sur ses performances à cet égard.

Avantage 1
Hors le problème soulevé ci-dessus, la gestion par masque des entrées/sorties gagne beaucoup de temps lors de la création d'un programme. Les fonctions READ, UPDATE et WRITE, en gérant les informations un écran à la fois, évitent de pénibles séries d'instructions de lecture et écriture, tout en compactant énormément les programmes.

Avantage 2
Les fichiers multiclés sont particulièrement utiles et épargnent l'écriture de routines de recherche d'enregistrements par clés, tout en fonctionnant de façon très performante.

Avantage 3
L'équipe qui a mis le M/DOS 6502 au point est une équipe sérieuse. Nous avons eu l'occasion de discuter avec des personnes qui avaient eu des difficultés avec la carte intiale; il ont bénéficié d'un excellent service après-vente, et les problèmes ont été résolus à la satisfaction générale.

Conclusion

En gérant la structure des données et des E/S écran et imprimante indépendamment de la structure du programme BASIC, ce système permet une plus grande adaptabilité des applications sans modification du code en BASIC.
Notre conclusion générale est tout à fait favorable. La principale critique que l'on pouvait adresser sur le fond était le manque de transportabilité : un programme écrit en M/DOS 6502 ne tourne que sur Apple et devra donc être réécrit si l'on change de matériel. Cet argument risque de perdre de son poids, puisque MIS étudie actuellement la réalisation de M/DOS 6502 sur 8088, 8086, sous MS/DOS, CP/M86 et sur 68000!

Remarques de la société MIS

1. Une nouvelle documentation est en cours de réalisation.
2. Un contrôle interzone conditionnel est annoncé pour la release XI.
3. Des efforts sont consacrés actuellement à l'amélioration de la carte, afin qu'il n'y ait plus de problèmes de puissance.

Un catalogue général en Pascal

Dans les précédents articles, nous avons placé dans un seul fichier les répertoires de tous nos disques Pascal. Chaque ligne contient le nom du fichier et sa date, le genre et le type de programme, le nom du disque et le numéro de la boîte où il est rangé.

Nous allons profiter de ces éléments pour effectuer, à l'écran ou sur l'imprimante, des sélections permettant de retrouver un fichier par son nom bien sûr, mais aussi de lister les fichiers sélectionnés sur un type et sur un genre déterminés, ou encore d'éditer un catalogue par numéro de boîte ou par disque.

Le programme précédemment écrit (cf. Pom's 2 et 3) reste valable; seules les procédures TRI et LIRE, qui étaient indiquées à améliorer, sont à remplacer par celles qui paraissent aujourd'hui.

La procédure LIRE propose maintenant différentes options ;

(0) : édition complète du catalogue comme précédemment
(1) : édition du catalogue sélectionné en fonction d'un type et d'un genre
(2) : édition du catalogue par nom de disque ou de fichier
(3) : édition par numéro de boîte ou de fichier.
Le choix étant fait, et le programme ayant demandé s'il faut une impression ou non, (le booléen IMP prenant la valeur de la fonction OUI), le CASE distribue la tâche en fonction de l'option demandée. Si IMP est vrai, édition d'un trait sur l'imprimante, sortie de quelques lignes de papier et demande pour une autre recherche, le booléen QUIT (qui prend la valeur de NOT OUI) permettant l'interruption de la boucle globale WHILE NOT QUIT.

Les différentes procédures de sélection sont très simples :

- demande de précision dans le critère
- choix du critère
- sortie de la procédure par un EXIT si le critère est nul
- sinon, affichage à l'écran ou sur l'imprimante des facteurs de sélection et lecture ligne par ligne du fichier par une boucle FOR I := 1 TO NUMCAT-1 DO.

La boucle effectue une comparaison avec les valeurs de sélection; quand la fiche est bonne, elle est imprimée si IMP est vrai, sinon affichée à l'écran. Le paramètre I sert à indiquer le numéro du fichier.

Quelques remarques

- dans TRIGT : le type et le genre sont des lettres de A à Y entrées lors de la création de la fiche; en effet la valeur Z sert ici à supprimer le facteur de sélection, c'est à dire que genre A type Z signifie n'importe quel type de genre A.

- dans TRINOM : il est possible, si l'on utilise des noms de fichiers très structurés (par exemple, quand tous les programmes graphiques comportent les lettres GR et les programmes utilitaires UT) de remplacer l'égalité de la boucle FOR par l'utilisation de la fonction POS du Pascal. Dans ce cas,

(GARDE^.NOMFILE=ST)

devient (POS(ST,GARDE^.NOMFILE)<>0). On peut trouver de cette façon tous les noms ayant une syllabe valide au lieu d'une égalité stricte.


```
(U/G) EXTRPOM3.TEXT        (C)  1982 M.CRIMONT                    31/03/82
(* =====POUR POM'S=====PROGRAMME DE CATALOGUE GENERAL  *)
-------------------------------------------------------------------------

(*****************************************)
(* L I R E  &  T R I                     *)
(*****************************************)

PROCEDURE TRI;
VAR I     :INTEGER;
BEGIN
  MESSAGE(3,'      EDITION DU CATALOGUE GENERAL');
  IF IMP THEN
    BEGIN
      MESSAGE(10,'      ***** SUR L''IMPRIMANTE *****');
      WRITELN(PAPIER,'EDITION DU CATALOGUE GENERAL':55);
     WRITELN(PAPIER)     END;
   TITRE;
  FOR I:=1 TO NUMCAT-1 DO
  BEGIN
    GET(GARDE);
    IF IMP THEN IMPRIME(I,GARDE^) ELSE AFFICHE(I,GARDE^)
  END
END;

PROCEDURE TRIGT;
VAR I       :INTEGER;
    G,T,NUL :CHAR;
BEGIN
  NUL:='Z';
  MESSAGE(5,'CODE DE ''A'' A ''Y'' OU ''Z'' POUR TOUS');
  MESSAGE(7,'CODE GENRE:');G:=PRENCAR(['A'..'Z',' ']);
  IF G=' ' THEN EXIT(TRIGT);
  MESSAGE(9,'CODE TYPE :');T:=PRENCAR(['A'..'Z',' ']);
  IF T=' ' THEN EXIT(TRIGT);
```

```
GOTOXY(0,3);WRITE(EFB);
MESSAGE(3,'CATALOGUE TRIE SUR ');
IF G<>NUL THEN WRITE('GENRE:',G);
IF (G<>NUL) AND (T<>NUL) THEN WRITE(' & ');
IF T<>NUL THEN WRITE('TYPE :',T);
IF IMP THEN
  BEGIN
    MESSAGE(10,'      ***** SUR L''IMPRIMANTE *****');
    WRITE(PAPIER,' ':15,'EDITION DU CATALOGUE TRIE SUR ');
    IF G<>NUL THEN WRITE(PAPIER,'GENRE:',G);
    IF (G<>NUL) AND (T<>'Z') THEN WRITE(PAPIER,' & ');
    IF T<>NUL THEN WRITE(PAPIER,'TYPE:',T);WRITELN(PAPIER);
  END;
TITRE;
FOR I:=1 TO NUMCAT-1 DO
BEGIN
  GET(GARDE);
  IF ((GARDE^.GENRE[1]=G) OR (G=NUL))AND((GARDE^.GENRE[2]=T)OR(T=NUL))
 THEN IF IMP THEN IMPRIME(I,GARDE^) ELSE AFFICHE(I,GARDE^)
  END
END;

PROCEDURE TRINOM;
VAR I   :INTEGER;
    OP  :CHAR;
    ST  :STRING;
BEGIN
  ST:='';
  MESSAGE(5,'<1> NOM DE FICHIER');
  MESSAGE(6,'<2> NOM DE DISQUE');
  MESSAGE(7,'OPTION ?');OP:=PRENCAR['1','2'];
  MESSAGE(9,'NOM DU ');IF OP='1' THEN WRITE('FICHIER:') ELSE WRITE
 ('DISQUE:');  IF OP='1' THEN I:=15 ELSE I:=7;
 (*$V-*)
 PRENCHAINE(I,['A'..'Z','0'..'9','.'],ST);
 (*$V+*)
 IF ST='' THEN EXIT(TRINOM);
 GOTOXY(0,3);WRITE(EFB);
 MESSAGE(3,'CATALOGUE DU ');
 IF OP='1' THEN WRITE('FICHIER:',ST)
   ELSE WRITE('DISQUE:',ST);
 IF IMP THEN
   BEGIN
     MESSAGE(10,'      ***** SUR L''IMPRIMANTE *****');
     WRITE(PAPIER,' ':15,'CATALOGUE DU  ');
     IF OP='1' THEN WRITE(PAPIER,'FICHIER:',ST)
       ELSE WRITE(PAPIER,'DISQUE:',ST);WRITELN(PAPIER);
   END;
 TITRE;
 FOR I:=1 TO NUMCAT-1 DO
 BEGIN
   GET(GARDE);
   IF ((OP='1') AND (GARDE^.NOMFILE=ST))
     OR ((OP='2') AND (GARDE^.NOMDISK=ST)) THEN
       IF IMP THEN IMPRIME(I,GARDE^) ELSE AFFICHE(I,GARDE^)
 END
END;
PROCEDURE TRINUM;
VAR I,N :INTEGER;
    OP  :CHAR;
BEGIN
```

```
  N:=0;
  MESSAGE(5,'<1> NUMERO DU FICHIER');
  MESSAGE(6,'<2> NUMERO DE BOITE');
  MESSAGE(7,'OPTION ?');OP:=FRENCAR(['1','2']);
  REPEAT
    MESSAGE(9,'NUMERO ');
    IF OP='1' THEN WRITE('DU FICHIER:') ELSE WRITE('DE BOITE:');
    ENTIER(4,N)
  UNTIL (N<NUMCAT);
  IF N=0 THEN EXIT(TRINUM);
  GOTOXY(0,3);WRITE(EFB);
  IF OP='1' THEN
  BEGIN
    TITRE;
    SEEK(GARDE,N);GET(GARDE);AFFICHE(N,GARDE^)
  END ELSE
  BEGIN
    MESSAGE(3,'CONTENU DE LA BOITE NUMERO ');WRITE(N);
    IF IMP THEN
      BEGIN
        MESSAGE(10,'      ***** SUR L''IMPRIMANTE *****');
        WRITE(PAPIER,' ':15,'CONTENU DE LA BOITE NUMERO ',N);
        WRITELN(PAPIER);
      END;
    TITRE;
    FOR I:=1 TO NUMCAT-1 DO
    BEGIN
      GET(GARDE);
      IF GARDE^.NUMBOITE=N THEN
        IF IMP THEN IMPRIME(I,GARDE^) ELSE AFFICHE(I,GARDE^)
    END
  END
END;
PROCEDURE LIRE;
VAR CHOIX  :CHAR;
BEGIN
  QUIT:=FALSE;
  GOTOXY(0,2);WRITE(EFB);
  MESSAGE(3,'LE CATALOGUE DOIT ETRE DANS LE DRIVE 2');
  WHILE NOT QUIT DO
  BEGIN
    GOTOXY(0,4);WRITE(EFB);NUMLIGNE:=6;
    MESSAGE(4,'<0>:EDITION COMPLETE');
    MESSAGE(5,'<1>:EDITE PAR GENRE & TYPE');
    MESSAGE(6,'<2>:EDITE PAR NOM (DISQUE OU FICHIER)');
    MESSAGE(7,'<3>:EDITE PAR NUMERO (BOITE OU FICHIER)');
    WRITE(INV);IMP:=FALSE;
    MESSAGE(9,'OPTION ?');CHOIX:=FRENCAR['0'..'3'];
    MESSAGE(9,'VOULEZ-VOUS IMPRIMER ? ');IMP:=OUI;
    WRITE(NORM);
    IF IMP THEN TRAIT;
    GOTOXY(0,3);WRITE(EFB);
    SEEK(GARDE,1);
    CASE CHOIX OF
      '0':TRI;
      '1':TRIGT;
      '2':TRINOM;
      '3':TRINUM
    END;
    IF IMP THEN TRAITFIN;
    MESSAGE(22,'UNE AUTRE RECHERCHE (O/N) ? ');QUIT:=NOT OUI
  END
END;
```

Chargez vite vos fichiers binaires

Ce programme, baptisé &LOAD, effectue un chargement rapide de tout programme en binaire, y compris les images graphiques: ceci est rendu possible par l'appel direct de la RWTS et le chargement direct de chaque secteur à son adresse définitive (sauf le premier et le dernier, pour ne pas "abîmer" l'environnement). Grâce à ce programme, une image graphique haute résolution est chargée en trois secondes, et même moins si le moteur est déjà lancé.

Cette routine requiert 48K de mémoire, le DOS 3.3 à son adresse habituelle et l'Applesoft en ROM. Elle peut être appelée sous Applesoft, à partir d'un programme ou en mode direct. Elle ne fonctionne ni en Integer, ni en mode moniteur, car la commande "&" n'est pas interprétée dans ces deux cas. Enfin, nous rappellons qu'elle ne s'applique qu'aux fichiers de type binaire.

Elle ne détruit aucune adresse mémoire de la page 0, sauf bien sûr celles utilisées par RWTS. Elle nécessite un seul pointeur page 0, qui est d'ailleurs restauré à la fin, et des adresses inutilisées dans le DOS.

La syntaxe est la suivante :
&nomfichier[,adresse de chargement]
ou bien :
&variablealpha[,adresse de chargement]

L'adresse de chargement est optionnelle; elle peut être un nombre ou une variable numérique, mais elle doit être décimale. Sa valeur doit être inférieure à 512, pour ne pas écraser la pile ni la page zéro. Si le nom du fichier est donné entre guillemets, les blancs sont significatifs, sauf s'il sont placés à droite.

Dans cette version, la routine se dissimule entre le DOS et ses buffers. Ces buffers sont descendus de deux pages, ce qui peut être ennuyeux si on utilise &LOAD dans un programme qui positionne HIMEM très haut en mémoire.

Une autre solution consiste à la dissimuler dans le premier buffer du DOS en $9AA6, ce qui permet, si l'on ne touche pas à MAXFILES, un fonctionnement normal de la plupart des programmes. Il est en effet rare que l'on ait plus de deux fichiers ouverts simultanément. La procédure à suivre est alors la suivante : remplacer les lignes 68 à 70 par RTS, et changer l'ORG de la ligne 74 de $9B00 à $9AA6.

Le programme a été écrit à l'aide de l'assembleur BIG MAC, qui est à mon avis intrinsèquement supérieur au LISA 2.5. Il autorise en particulier l'assemblage conditionnel et l'utilisation de minuscules dans les commentaires.

Note de la rédaction : nous avons grâce &LOAD chargé en 9 secondes au lieu de 30 un fichier de 144 secteurs. Question : les adresses AA60 (longueur de fichier) et AA72 (début) ne sont plus valables. Où les trouve-t-on ?

```
                1               ORG  $9000
                2       *
                3       *
                4       ************************************
                5       *                                  *
                6       *              &BLOAD              *
                7       *                                  *
                8       *          Jacques Tran-Van        *
                9       *                                  *
                10      ************************************
                11      *
                12      *
                13      *          18/04/82
                14      *
                15      *
                16      PTR         EQU   $1E
                17      START       EQU   $1E
                18      VALTYPE     EQU   $11
                19      TRAP        EQU   $48
                20      LINUM       EQU   $50
                21      CHARGET     EQU   $B1
                22      CHARGOT     EQU   $B7
                23      TXTPTR      EQU   $B8
                24      AMPERVT     EQU   $3F5
                25      ERHANDL     EQU   $A6D5
                26      CLNBUFF     EQU   $A7D4
                27      ERRCODE     EQU   $AA5C
                28      NAME        EQU   $AA75
                29      LENGTH      EQU   $AAAB
                30      SAV1        EQU   $AAAC
                31      SAV2        EQU   $AAAD
                32      TEMP        EQU   $AAAE
                33      FLAG        EQU   $AAAF
                34      ADRFLAG     EQU   $AAB0
                35      BUFFER      EQU   $B3BB
                36      TAMPON      EQU   $B4BB
                37      RWTS        EQU   $B7B5
                38      IOB         EQU   $B7E8
                39      VOLUME      EQU   IOB+3
                40      TRACK       EQU   IOB+4
                41      SECTOR      EQU   IOB+5
                42      BUFADR      EQU   IOB+8
                43      COMMAND     EQU   IOB+12
                44      BASCERR     EQU   $D412
                45      FRMNUM      EQU   $DD67
                46      FRMEVL      EQU   $DD7B
                47      CHKCOM      EQU   $DEBE
                48      PTRGET      EQU   $DFE3
                49      GETADR      EQU   $E752
                50      SETVID      EQU   $FE93
                51      *
                52      *
9000: A9 00     53                  LDA   #<ENTREE    ;positionne le vecteur
9002: 8D F6 03  54                  STA   AMPERVT+1   ;d'ampersand
9005: A9 9B     55                  LDA   #>ENTREE
9007: 8D F7 03  56                  STA   AMPERVT+2
900A: A9 4C     57                  LDA   #$4C        ;'JMP'
900C: 8D F5 03  58                  STA   AMPERVT
900F: A2 00     59                  LDX   #0
9011: BD 2B 90  60      MOVEPGM     LDA   MOVEADR,X   ;cache le pgm entre
9014: 9D 00 9B  61                  STA   ENTREE,X    ;le dos et ses buffers
9017: E8        62                  INX
9018: D0 F7     63                  BNE   MOVEPGM
901A: 8D 2B 91  64      MOVEPGM1    LDA   MOVEADR+256,X
901D: 9D 00 9C  65                  STA   ENTREE+256,X
9020: E8        66                  INX
```

```
]BLOAD CHARGE.OBJ,A$9000
]CALL-151

*9000.922A

9000- A9 00 8D F6 03 A9 9B 8D
9008- F7 03 A9 4C 8D F5 03 A2
9010- 00 BD 2B 90 9D 00 9B E8
9018- D0 F7 BD 2B 91 9D 00 9C
9020- E8 D0 F7 A9 9A 8D 01 9D
9028- 4C D4 A7 A5 1E 8D AC AA
9030- A5 1F 8D AD AA 20 B7 00
9038- C9 22 D0 28 A9 0A 85 C1
9040- A8 00 BC EB B7 20 B1 00
9048- C9 22 F0 0C 09 00 99 75
9050- AA C8 C8 1E 90 EF B0 34
9058- A9 EF 85 C1 20 B1 00 C0
9060- 03 D0 4C 69 A3 88 48 A5
9068- B9 48 20 7B DD 24 11 30
9070- 08 A2 A3 20 F5 9C 4C 12
9078- D4 60 85 09 68 85 88 20
9080- E3 0F 85 1E 84 1F A0 00
9088- B1 1E D0 05 A2 08 4C 0A
9090- 9C 8D AB AA C8 B1 1E 48
9098- C8 B1 1E 85 1F 68 85 1E
90A0- A0 00 B1 1E B9 B6 99 75
90A8- AA C8 CC A8 AA D0 F3 A9
90B0- A0 99 75 AA C8 C4 1E D0
90B8- F8 8D B0 AA 20 F5 9C 20
90C0- B7 00 F0 12 20 BE DE 20
90C8- A7 DD 20 52 E7 A5 51 C9
90D0- 02 90 B9 4E 88 AA A9 01
90D8- 8D F4 B7 A9 10 8D ED B7
90E0- A9 11 8D EC B7 20 E4 9C
90E8- CE ED B7 F0 4A 20 CB 9C
90F0- A8 EB 8C AE AA AD AE AA
90F8- 10 A9 23 A8 8D AE AA C9
9100- 03 F0 E5 A2 00 B9 BB B3
9108- F0 29 DD 75 AA D0 EA C8
9110- E8 E0 1E D0 F0 B9 9C B3
9118- 29 04 D0 04 A2 0D D0 15
9120- AC AE AA B9 B7 B3 8D ED
9128- B7 B9 B8 B3 38 C7 8D EC
9130- B7 D0 8E A2 04 0E 5C AA
9138- 20 F5 9C 20 93 FE 4C 05
9140- A6 8D AF AA 29 E0 9C A3
9148- 0C A5 1E AA 1F 20 EE 9C
9150- A0 AF AA F0 02 20 EA 9C
9158- 29 BB 9C A0 AF AA F0 10
9160- ZC B6 AA 38 9A A5 50 8D
9168- BB B4 A5 51 8D BC B4 38
9170- AD BB B4 E9 04 85 1E 8D
9178- F0 B7 A0 BC B4 E9 00 85
9180- 1F 8D F1 B7 A9 04 8D DA
9188- 9C 18 6D 8D B4 8D AB AA
9190- A9 00 8D E2 9C 8D AF AA
9198- AD C9 B3 F0 22 20 D9 9C
91A0- AE AE AA EE F1 B7 E4 1F
91A8- E8 E0 FE B0 10 8D BD B3
91B0- D0 A6 20 EA 9C 20 BB 9C
91B8- A9 00 85 48 8D DA 9C AD
91C0- A0 AA 8D E2 9C 20 D9 9C
91C8- F0 56 AD BC B3 F0 E3 20
91D0- B8 9C A0 8D B3 8D ED B7
91D8- E6 1F 20 E6 9C A0 BC B3
91E0- 8D EC B7 4C 19 9C 8D 88
91E8- B3 8D EC B7 E8 8D 8D B3
91F0- 8D ED B7 8E AE AA A9 8
91F8- A0 E8 20 85 B7 98 11 A2
9200- 0B 4C 0A 9C A0 16 B9 0B
9208- B9 91 1E C8 C8 B1 B4 F6
9210- A0 A0 B3 D8 82 A8 B4 A0
9218- BB BD F8 B7 8C F1 B7 A0
9220- D AC AA 85 1E AD A0 9A
9228- 85 1F 68
```

```
9021: D0 F7     67              BNE   MOVEPGM1
9023: A9 9A     68              LDA   #$9A
9025: 8D 01 9D  69              STA   $9D01     ;decale les buffers des 2 pages
9028: 4C D4 A7  70              JMP   CLNBUFF   ;et les reconstruit
                71      *
                72      MOVEADR EQU   *
                73      *
                74              ORG   $9B00
                75      *
9B00: A5 1E     76      ENTREE  LDA   PTR       ;sauve les memoires page 0
9B02: 8D AC AA  77              STA   SAV1      ;utilisees par le pgm
9B05: A5 1F     78              LDA   PTR+1
9B07: 8D AD AA  79              STA   SAV2
9B0A: 20 B7 00  80              JSR   CHARGOT
9B0D: C9 22     81              CMP   #'"'      ;titre entre apostrophes ?
9B0F: D0 28     82              BNE   NOCONST
9B11: A9 06     83              LDA   #6        ;modifie CHARGET pour qu'elle
9B13: 85 C1     84              STA   $C1       ;accepte les espaces
9B15: A0 00     85              LDY   #0
9B17: 8C EB B7  86              STY   VOLUME    ;volume prevu
9B1A: 20 B1 00  87      INPLOOP JSR   CHARGET   ;lit le caractere suivant
9B1D: C9 22     88              CMP   #'"'
9B1F: F0 0C     89              BEQ   ENDNAME   ;fin du titre
9B21: 09 80     90              ORA   #$80      ;leve le bit de poids fort
9B23: 99 75 AA  91              STA   NAME,Y    ;sauve le nom dans NAME
9B26: C8        92              INY
9B27: C0 1E     93              CPY   #30       ;pas plus de 30 caracteres
9B29: 90 EF     94              BCC   INPLOOP
9B2B: B0 34     95              BCS   SYNTAX    ;si trop long, 'SYNTAX ERROR'
9B2D: A9 EF     96      ENDNAME LDA   #$EF
9B2F: 85 C1     97              STA   $C1       ;restaure CHARGET
9B31: 20 B1 00  98              JSR   CHARGET
9B34: C0 00     99              CPY   #0
9B36: D0 4C     100             BNE   ENDSTOR   ;termine par des espaces
9B38: 60        101             RTS             ;retour au BASIC
                102     *
9B39: A5 B8     103     NOCONST LDA   TXTPTR
9B3B: 48        104             PHA             ;sauve le pointeur de CHARGET
9B3C: A5 B9     105             LDA   TXTPTR+1
9B3E: 48        106             PHA
9B3F: 20 7B DD  107             JSR   FRMEVL    ;evalue l'expression
9B42: 24 11     108             BIT   VALTYPE   ;controle le type de la variable
9B44: 30 08     109             BMI   STRING    ;si < 0, alphanumerique
9B46: A2 A3     110             LDX   #$A3      ;code de 'TYPE MISMATCH'
9B48: 20 F5 9C  111             JSR   RESTORE
9B48: 4C 12 D4  112             JMP   BASCERR   ;affichage du message d'erreur
                113     *
9B4E: 68        114     STRING  PLA             ;recupere le pointeur initial
9B4F: 85 B9     115             STA   TXTPTR+1
9B51: 68        116             PLA
9B52: 85 B8     117             STA   TXTPTR
9B54: 20 E3 DF  118             JSR   PTRGET    ;trouve le pointeur de variable
9B57: 85 1E     119             STA   PTR       ;le range dans PTR
9B59: 84 1F     120             STY   PTR+1
9B5B: A0 00     121             LDY   #0
9B5D: B1 1E     122             LDA   (PTR),Y   ;longueur du nom
9B5F: D0 05     123             BNE   NONUL     ;le string existe
9B61: A2 0B     124     SYNTAX  LDX   #11
9B63: 4C 0A 9C  125             JMP   ERROR     ;retour au BASIC, et 'SYNTAX ERROR'
9B66: 8D AB AA  126     NONUL   STA   LENGTH
9B69: C8        127             INY
9B6A: B1 1E     128             LDA   (PTR),Y   ;pointeur vers les valeurs
9B6C: 48        129             PHA             ;dans PTR
9B6D: C8        130             INY
```

```
9B6E: B1 1E     131           LDA  (PTR),Y
9B70: 85 1F     132           STA  PTR+1
9B72: 68        133           PLA
9B73: 85 1E     134           STA  PTR
9B75: A0 00     135           LDY  #0
9B77: B1 1E     136 STORE     LDA  (PTR),Y    ;sauve le titre dans NAME
9B79: 09 80     137           ORA  #$80
9B7B: 99 75 AA  138           STA  NAME,Y
9B7E: C8        139           INY
9B7F: CC AB AA  140           CPY  LENGTH
9B82: D0 F3     141           BNE  STORE
9B84: A9 A0     142 ENDSTOR   LDA  #" "        ;complete le nom par des espaces
9B86: 99 75 AA  143 SPACE     STA  NAME,Y
9B89: C8        144           INY
9B8A: C0 1E     145           CPY  #30
9B8C: D0 F8     146           BNE  SPACE
9B8E: 8D B0 AA  147           STA  ADRFLAG    ;flag d'adresse < 0
                148 *
9B91: 20 F5 9C  149           JSR  RESTORE    ;restaure le pointeur(si erreur)
9B94: 20 B7 00  150           JSR  CHARGOT    ;y a-t-il autre chose derriere ?
9B97: F0 12     151           BEQ  SEARCH     ;non
9B99: 20 BE DE  152           JSR  CHKCOM     ;alors il faut une virgule !
9B9C: 20 67 DD  153           JSR  FRMNUM     ;evalue la formule qui suit
9B9F: 20 52 E7  154           JSR  GETADR     ;range l'adresse dans LINUM
9BA2: A5 51     155           LDA  LINUM+1    ;si adresse de chargement < $200
9BA4: C9 02     156           CMP  #2         ;alors erreur !!
9BA6: 90 B9     157           BCC  SYNTAX
9BA8: 4E B0 AA  158           LSR  ADRFLAG    ;flag d'adresse > 0
                159 *
9BAB: A9 01     160 SEARCH    LDA  #1         ;commande lecture pour la RWTS
9BAD: 8D F4 B7  161           STA  COMMAND
9BB0: A9 10     162           LDA  #$10
9BB2: 8D ED B7  163           STA  SECTOR
9BB5: A9 11     164           LDA  #$11
9BB7: 8D EC B7  165           STA  TRACK
9BBA: 20 E6 9C  166           JSR  SETBUF     ;BUFFER pris comme buffer
9BBD: CE ED B7  167 NEXTSCT   DEC  SECTOR     ;secteur suivant
9BC0: F0 46     168           BEQ  NOFILE     ;fin de la directory
9BC2: 20 CB 9C  169           JSR  READSCT    ;lecture du secteur
9BC5: A0 EB     170           LDY  #$EB
9BC7: 8C AE AA  171           STY  TEMP
9BCA: AD AE AA  172 NEXTFIL   LDA  TEMP
9BCD: 18        173           CLC
9BCE: 69 23     174           ADC  #35
9BD0: A8        175           TAY             ;pointe vers le prochain titre
9BD1: 8D AE AA  176           STA  TEMP
9BD4: C9 03     177           CMP  #3
9BD6: F0 E5     178           BEQ  NEXTSCT    ;fin du secteur
9BD8: A2 00     179           LDX  #0
9BDA: B9 BB B3  180 TESTNOM   LDA  BUFFER,Y
9BDD: F0 29     181           BEQ  NOFILE     ;fin de la directory
9BDF: DD 75 AA  182           CMP  NAME,X     ;compare le nom a NAME
9BE2: D0 E6     183           BNE  NEXTFIL
9BE4: C8        184           INY
9BE5: E8        185           INX
9BE6: E0 1E     186           CPX  #30
9BE8: D0 F0     187           BNE  TESTNOM    ;fin du nom ?
                188 *
9BEA: B9 9C B3  189           LDA  BUFFER-31,Y ;quel est son type ?
9BED: 29 04     190           AND  #4         ;binaire(4 ou 84) ?
9BEF: D0 04     191           BNE  FINDFIL    ;oui bon type
9BF1: A2 0D     192           LDX  #13        ;code de 'FILE TYPE MISMATCH'
9BF3: D0 15     193           BNE  ERROR
9BF5: AC AE AA  194 FINOFIL   LDY  TEMP       ;recupere le pointeur de nom
9BF8: B9 B9 B3  195           LDA  BUFFER-2,Y
```

```
9BFB: 8D ED B7 196          STA  SECTOR      ;# du secteur de la TSL
9BFE: B9 B8 B3 197          LDA  BUFFER-3,Y ;# de la piste de la TSL
9C01: 30 C7    198          BMI  NEXTFIL     ;si efface, le saute
9C03: 8D EC B7 199          STA  TRACK
9C06: D0 0E    200          BNE  LOADPGM     ;charger le programme
9C08: A2 06    201 NOFILE   LDX  #6          ;code de 'FILE NOT FOUND'
9C0A: 8E 5C AA 202 ERROR    STX  ERRCODE     ;range le code de l'erreur
9C0D: 20 F5 9C 203          JSR  RESTORE     ;restaure les memoires page 0
9C10: 20 93 FE 204          JSR  SETVID      ;PR#0
9C13: 4C D5 A6 205          JMP  ERRHANDL    ;saut a la routine d'erreur du DOS
              206 *
9C16: 8D AF AA 207 LOADPGM  STA  FLAG        ;initialisation de FLAG(<> 0)
9C19: 20 CB 9C 208 NEXTSL   JSR  READSCT     ;lit la TSL dans BUFFER
9C1C: A2 0C    209          LDX  #$0C        ;pointeur dans la TSL
9C1E: A5 1E    210          LDA  START       ;recupere l'adresse du buffer,
9C20: A4 1F    211          LDY  START+1     ;si c'est la 2eme TSL
9C22: 20 EE 9C 212          JSR  LOCBUF
9C25: AD AF AA 213          LDA  FLAG
9C28: F0 03    214          BEQ  INPSECT     ;si = 0, lit le secteur
9C2A: 20 EA 9C 215          JSR  SETTAMP     ;TAMPON est pris comme buffer
9C2D: 20 BB 9C 216 INPSECT  JSR  READPGM     ;lecture du secteur du pgm
9C30: AD AF AA 217          LDA  FLAG
9C33: F0 40    218          BEQ  NEXTONE
9C35: 2C B0 AA 219          BIT  ADRFLAG     ;chargement a une autre adresse ?
9C38: 30 0A    220          BMI  NOADR       ;non
9C3A: A5 50    221          LDA  LINUM
9C3C: 8D BB B4 222          STA  TAMPON
9C3F: A5 51    223          LDA  LINUM+1
9C41: 8D BC B4 224          STA  TAMPON+1
9C44: 38       225 NOADR    SEC
9C45: AD BB B4 226          LDA  TAMPON
9C48: E9 04    227          SBC  #4          ;calcule les adresses
9C4A: 85 1E    228          STA  START       ;de debut de programme,
9C4C: 8D F0 B7 229          STA  BUFADR      ;et de chargement
9C4F: A0 BC B4 230          LDA  TAMPON+1
9C52: E9 00    231          SBC  #0
9C54: 85 1F    232          STA  START+1
9C56: 8D F1 B7 233          STA  BUFADR+1
9C59: A9 04    234          LDA  #4          ;depart du 'move' du 1er
9C5B: 8D DA 9C 235          STA  DEPLACE+1   ;secteur du pgm
9C5E: 18       236          CLC
9C5F: 6D BD B4 237          ADC  TAMPON+2    ;augmente la longueur de 4
9C62: 8D AB AA 238          STA  LENGTH      ;longueur du dernier secteur
9C65: A9 00    239          LDA  #0          ;fin du 'move'
9C67: 8D E2 9C 240          STA  ENDMOVE+1
9C6A: 8D AF AA 241          STA  FLAG        ;annule le flag
9C6D: AD C9 B3 242          LDA  BUFFER+14   ;un autre secteur a lire ?
9C70: F0 22    243          BEQ  LAST        ;c'est le dernier
              244 *
9C72: 20 D9 9C 245          JSR  DEPLACE     ;deplace le secteur
9C75: AE AE AA 246 NEXTONE  LDX  TEMP        ;recupere le pointeur dans la TSL
9C78: EE F1 B7 247          INC  BUFADR+1    ;buffer suivant
9C78: E6 1F    248          INC  START+1     ;incremente l'adresse destination
9C7D: E8       249          INX              ;secteur suivant
9C7E: E0 FE    250          CPX  #$FE        ;dernier de cette TSL ?
9C80: B0 1D    251          BCS  ENDTSL      ;oui
9C82: BD BD B3 252          LDA  BUFFER+2,X ;y a-t-il une suite ?
9C85: D0 A6    253          BNE  INPSECT
              254 *
9C87: 20 EA 9C 255 LASTSCT  JSR  SETTAMP     ;TAMPON comme buffer
9C8A: 20 BB 9C 256          JSR  READPGM     ;lit le dernier secteur du pgm
9C8D: A9 00    257          LDA  #0
9C8F: 85 48    258          STA  TRAP        ;Voir Call Apple janvier 82
9C91: 8D DA 9C 259          STA  DEPLACE+1   ;deplace a partir du debut
9C94: AD AB AA 260 LAST     LDA  LENGTH
```

```
9C97: 8D E2 9C 261            STA  ENDMOVE+1  ;'longueur' du dernier secteur
9C9A: 20 D9 9C 262            JSR  DEPLACE    ;deplacement
9C9D: F0 56    263            BEQ  RESTORE    ;retour au BASIC
              264  *
9C9F: AD BC B3 265  ENDTSL    LDA  BUFFER+1   ;# de secteur pour la TSL suivante
9CA2: F0 E3    266            BEQ  LASTSCT    ;pas de 2nde TSL
9CA4: 20 BB 9C 267            JSR  READPGM    ;lit le dernier secteur de la TSL
9CA7: AD BD B3 268            LDA  BUFFER+2   ;# de secteur de la 2nde TSL
9CAA: 8D ED B7 269            STA  SECTOR
9CAD: E6 1F    270            INC  START+1    ;buffer suivant(pour la suite)
9CAF: 20 E6 9C 271            JSR  SETBUF     ;la 2nde TSL est lue dans BUFFER
9CB2: AD BC B3 272            LDA  BUFFER+1   ;# de piste de la 2nde TSL
9CB5: 8D EC B7 273            STA  TRACK
9CB8: 4C 19 9C 274            JMP  NEXTSL
              275  *
9CBB: BD BB B3 276  READPGM   LDA  BUFFER,X   ;piste
9CBE: 8D EC B7 277            STA  TRACK
9CC1: E8       278            INX
9CC2: BD BB B3 279            LDA  BUFFER,X   ;secteur
9CC5: 8D ED B7 280            STA  SECTOR
9CC8: 8E AE AA 281            STX  TEMP       ;sauve le pointeur
9CCB: A9 B7    282  READSCT   LDA  #>IOB      ;adresse de l'IOB
9CCD: A0 E8    283            LDY  #<IOB      ;( pour un 48K )
9CCF: 20 B5 B7 284            JSR  RWTS
9CD2: 90 11    285            BCC  NOERROR    ;pas d'erreur a la lecture
9CD4: A2 08    286            LDX  #8         ;code de 'I/O ERROR'
9CD6: 4C 0A 9C 287            JMP  ERROR      ;retour au BASIC
              288  *
9CD9: A0 00    289  DEPLACE   LDY  #0         ;'poke' les valeurs
9CDB: B9 BB B4 290  MOVE      LDA  TAMPON,Y   ;de depart et de fin
9CDE: 91 1E    291            STA  (START),Y  ;de cette routine
9CE0: C8       292            INY
9CE1: C0 00    293  ENDMOVE   CPY  #00
9CE3: D0 F6    294            BNE  MOVE
9CE5: 60       295  NOERROR   RTS
              296  *
9CE6: A0 B3    297  SETBUF    LDY  #>BUFFER   ;BUFFER pris comme buffer
9CE8: D0 02    298            BNE  COMMUN
9CEA: A0 B4    299  SETTAMP   LDY  #>TAMPON   ;TAMPON pris comme buffer
              300  *
9CEC: A9 BB    301  COMMUN    LDA  #<BUFFER   ;BUFFER et TAMPON consecutifs
              302  *
9CEE: 8D F0 B7 303  LDCBUF    STA  BUFADR     ;positionne le pointeur de buffer
9CF1: 8C F1 B7 304            STY  BUFADR+1
9CF4: 60       305            RTS
              306  *
9CF5: AD AC AA 307  RESTORE   LDA  SAV1
9CF8: 85 1E    308            STA  PTR        ;restaure les memoires page 0
9CFA: AD AD AA 309            LDA  SAV2       ;utilisees par le pgm
9CFD: 85 1F    310            STA  PTR+1
9CFF: 60       311            RTS
              312  *

--END ASSEMBLY--

ERRORS: 0

555 BYTES
```

Les codes ASCII épluchés

Certains de nos lecteurs nous ont reproché de copier des articles déjà parus dans des revues étrangères. La critique est aisée, mais l'art est difficile ! De par leur nombre, leur périodicité et leur antériorité, les revues américaines ont accumulé un nombre impressionnant d'articles. Il serait assurément possible à un collectionneur assidu des Softalk, Call-Apple, Nibble, Compute et autres revues de trouver pour la majorité des articles parus dans Pom's un article déjà publié traitant du même sujet ou ayant des ressemblances.

Devrions-nous nous abstenir de publier des articles sur le DOS, des comparaisons entre les systèmes de traitements de texte, une initiation à l'assembleur, etc... sous prétexte que des choses semblables ont été vues dans des revues étrangères ? Outre que nos lecteurs ne lisent pas tous l'anglais couramment, on n'a pas toujours le loisir d'avaler (et de digérer ...) une dizaine de revues par mois. Nous tenons à vous offrir une revue intéressante, vivante et utile, avec le maximum d'articles originaux. Mais, lorsque nous trouvons des choses intéressantes chez les autres, pourquoi ne pas vous en faire profiter ? C'est le cas ici, puisque les tableaux 1 à 4 nous ont été inspirés par un article de la revue Micro, the 6502 and 6809 Journal, du mois d'octobre 1981.

Nous espérons que ces tableaux vous aideront et qu'ils seront en bonne place dans votre aide-mémoire. Afin que leur utilisation vous soit plus facile, voici ci-dessous les explications concernant chaque colonne :

* HEXA : contient tous les nombres hexadécimaux de $00 à $FF.
* DECIMAL : traduction décimale du nombre hexadécimal.
* BINAIRE : octet de 8 bits représentatif du nombre en hexa.
* DEC*256 : valeur décimale de l'octet haut d'un nombre ou d'une adresse 16 bits.
* ASCII : signe ASCII attribué au nombre hexa correspondant. Ce code étant exprimé en 7 bits, les signes sur les valeurs de $00 à $7F sont identiques à ceux des valeurs de $80 à $FF.
* ECRAN : le manuel de référence donne en page 15 la correspondance exacte entre la valeur hexa et sa représentation sur l'écran. Les lettres I, F et N représentent les modes Inverse, Flash et Normal. Les valeur non affichées à l'écran sont remplacés par des tirets.

* TOUCHE : indique la touche à appuyer afin d'obtenir la valeur correspondante. A noter que lesvaleurs $00 à $7F ne peuvent être obtenues à partir du clavier. Les lettres C, S et CS représentent les touches Control, Shift et la combinaison des deux.
* OPCODES : représente en mnémoniques 6502 la signification des différentes valeurs hexa, dans le cas du langage machine. Les notations suivantes ont été utilisées :

A	absolu	Im	immédiat
A,X	absolu indexé par X	(I)	indirect
A,Y	absolu indexé par Y	ZP	page zéro
(I,X)	indirect indexé par X		
(I,Y)	indirect indexé par Y		
ZP,X	page zéro indexé par X		
ZP,Y	page zéro indexé par Y		

* TOKEN : le BASIC Integer et l'Applesoft utilisent, afin de diminuer l'encombrement mémoire, un codage des mots réservés du BASIC. Les codes $00 à $7F sont réservés à l'Integer, ceux entre $80 et $FF à l'Applesoft. Dans le cas de l'Integer, où un même signe peut avoir plusieurs codes selon le contexte, un court exemple est donné à droite de la colonne.

TABLEAU 1 ($00 a $3F)

HEXA	DEC	BINAIRE	DEC*256	ASCII	ECRAN	TOUCH	OPCODE-6502	TOKEN INTEGER
00	0	00000000	0	nul	I:@	-	BRK	Debut de ligne
01	1	00000001	256	soh	I:A	-	ORA-(I,X)	Fin de ligne
02	2	00000010	512	stx	I:B	-	---	Non utilise
03	3	00000011	768	etx	I:C	-	---	: separateur
04	4	00000100	1024	eot	I:D	-	---	LOAD K7
05	5	00000101	1280	enq	I:E	-	ORA-ZP	SAVE K7
06	6	00000110	1536	ack	I:F	-	ASL-ZP	CON
07	7	00000111	1792	bel	I:G	-	---	RUN ligne n
08	8	00001000	2048	bs	I:H	-	PHP	RUN debut
09	9	00001001	2304	ht	I:I	-	ORA-Im	DEL
0A	10	00001010	2560	lf	I:J	-	ASL	, DEL 0,5
0B	11	00001011	2816	vt	I:K	-	---	NEW
0C	12	00001100	3072	ff	I:L	-	---	CLR
0D	13	00001101	3328	cr	I:M	-	ORA-A	AUTO
0E	14	00001110	3584	so	I:N	-	ASL-A	, AUTO 0,5
0F	15	00001111	3840	si	I:O	-	---	MAN
10	16	00010000	4096	dle	I:P	-	BPL	HIMEM:
11	17	00010001	4352	dc1	I:Q	-	ORA-(I),Y	LOMEM:
12	18	00010010	4608	dc2	I:R	-	---	+ operateurs
13	19	00010011	4864	dc3	I:S	-	---	- numeriques
14	20	00010100	5120	dc4	I:T	-	---	* par exemple:
15	21	00010101	5376	nak	I:U	-	ORA-ZP,X	/ A=14*(27+15)
16	22	00010110	5632	syn	I:V	-	ASL-ZP,X	^
17	23	00010111	5888	etb	I:W	-	---	# operateurs
18	24	00011000	6144	can	I:X	-	CLC	>= logiques
19	25	00011001	6400	em	I:Y	-	ORA-A,Y	> pour
1A	26	00011010	6656	sub	I:Z	-	---	<= variables
1B	27	00011011	6912	esc	I:[	-	---	<> numeriques
1C	28	00011100	7168	fs	I:\	-	---	< par exemple
1D	29	00011101	7424	gs	I:]	-	ORA-A,X	AND IF X>=12
1E	30	00011110	7680	rs	I:^	-	ASL-A,X	OR THEN...
1F	31	00011111	7936	us	I:_	-	---	MOD
20	32	00100000	8192		I:	-	JSR	^
21	33	00100001	8448	!	I:!	-	AND-(I,X)	Non utilise
22	34	00100010	8704	"	I:"	-	---	(DIM A$(3)
23	35	00100011	8960	#	I:#	-	---	, A$(3,3)
24	36	00100100	9216	$	I:$	-	BIT-ZP	THEN IFX=3THEN10
25	37	00100101	9472	%	I:%	-	AND-ZP	THEN IFX=3THENA=2
26	38	00100110	9728	&	I:&	-	ROL-ZP	, INPUT"ST",A$
27	39	00100111	9984	'	I:'	-	---	, INPUT"ST",A
28	40	00101000	10240	(	I:(	-	PLP	" D(but
29	41	00101001	10496	)	I:)	-	AND-Im	" Fin
2A	42	00101010	10752	*	I:*	-	ROL	(A$(3)
2B	43	00101011	11008	+	I:+	-	---	Non utilise
2C	44	00101100	11264	,	I:,	-	BIT-A	Non utilise
2D	45	00101101	11520	-	I:-	-	AND-A	(A(3)
2E	46	00101110	11776	.	I:.	-	ROL-A	PEEK
2F	47	00101111	12032	/	I:/	-	---	RND
30	48	00110000	12288	0	I:0	-	BMI	SGN
31	49	00110001	12544	1	I:1	-	AND-(I),Y	ABS
32	50	00110010	12800	2	I:2	-	---	PDL
33	51	00110011	13056	3	I:3	-	---	Non utilise
34	52	00110100	13312	4	I:4	-	---	(DIM A(3)
35	53	00110101	13568	5	I:5	-	AND-ZP,X	+ A=+3
36	54	00110110	13824	6	I:6	-	ROL-ZP,X	- A=-3
37	55	00110111	14080	7	I:7	-	---	NOT
38	56	00111000	14336	8	I:8	-	SEC	(
39	57	00111001	14592	9	I:9	-	AND-A,Y	= IFA$="C"THEN..
3A	58	00111010	14848	:	I::	-	---	# IFA$#"C"THEN..
3B	59	00111011	15104	;	I:;	-	---	LEN(
3C	60	00111100	15360	<	I:<	-	---	ASC(
3D	61	00111101	15616	=	I:=	-	AND-A,X	SCRN(
3E	62	00111110	15872	>	I:>	-	ROL-A,X	, SCRN(3,5)
3F	63	00111111	16128	?	I:?	-	---	(

TABLEAU 2 ($40 a $7F)

HEXA	DEC	BINAIRE	DEC$256	ASCII	ECRAN	TOUCH	OPCODE-6502	TOKEN INTEGER
40	64	01000000	16384	@	F:@	-	RTI	$ A$
41	65	01000001	16640	A	F:A	-	EOR-(I,X)	Non utilise
42	66	01000010	16896	B	F:B	-	---	(
43	67	01000011	17152	C	F:C	-	---	,
44	68	01000100	17408	D	F:D	-	---	,
45	69	01000101	17664	E	F:E	-	EOR-ZP	;
46	70	01000110	17920	F	F:F	-	LSR-ZP	;
47	71	01000111	18176	G	F:G	-	---	;
48	72	01001000	18432	H	F:H	-	PHA	,
49	73	01001001	18688	I	F:I	-	EOR-Im	,
4A	74	01001010	18944	J	F:J	-	LSR	,
4B	75	01001011	19200	K	F:K	-	---	TEXT
4C	76	01001100	19456	L	F:L	-	JMP-A	GR
4D	77	01001101	19712	M	F:M	-	EOR-A	CALL
4E	78	01001110	19968	N	F:N	-	LSR-A	DIM DIM A$(3)
4F	79	01001111	20224	O	F:O	-	---	DIM DIM A(3)
50	80	01010000	20480	P	F:P	-	BVC	TAB
51	81	01010001	20736	Q	F:Q	-	EOR-(I),Y	END
52	82	01010010	20992	R	F:R	-	---	INPUT INPUT A$
53	83	01010011	21248	S	F:S	-	---	INPUT INPUT"T",A$
54	84	01010100	21504	T	F:T	-	---	INPUT INPUT A
55	85	01010101	21760	U	F:U	-	EOR-ZP,X	FOR
56	86	01010110	22016	V	F:V	-	LSR-ZP,X	= FOR I=1 TO 3
57	87	01010111	22272	W	F:W	-	---	TO
58	88	01011000	22528	X	F:X	-	CLI	STEP
59	89	01011001	22784	Y	F:Y	-	EOR-A,Y	NEXT
5A	90	01011010	23040	Z	F:Z	-	---	, NEXT I,J
5B	91	01011011	23296	[	F:[	-	---	RETURN
5C	92	01011100	23552	\	F:\	-	---	GOSUB
5D	93	01011101	23808	]	F:]	-	EOR-A,X	REM
5E	94	01011110	24064	^	F:^	-	LSR-A,X	LET
5F	95	01011111	24320	_	F:_	-	---	GOTO
60	96	01100000	24576	`	F:	-	RTS	IF
61	97	01100001	24832	a	F:!	-	ADC-(I,X)	PRINT PRINT A$
62	98	01100010	25088	b	F:"	-	---	PRINT PRINT A
63	99	01100011	25344	c	F:#	-	---	PRINT PRINT
64	100	01100100	25600	d	F:$	-	---	POKE
65	101	01100101	25856	e	F:%	-	ADC-ZP	, POKE 3,3
66	102	01100110	26112	f	F:&	-	ROR-ZP	COLOR=
67	103	01100111	26368	g	F:'	-	---	PLOT
68	104	01101000	26624	h	F:(	-	PLA	, PLOT 3,3
69	105	01101001	26880	i	F:)	-	ADC-Im	HLIN
6A	106	01101010	27136	j	F:*	-	ROR	, HLIN 3,5 AT
6B	107	01101011	27392	k	F:+	-	---	AT HLIN 3,5 AT
6C	108	01101100	27648	l	F:,	-	JMP-(I)	VLIN
6D	109	01101101	27904	m	F:-	-	ADC-A	, VLIN 3,5 AT
6E	110	01101110	28160	n	F:.	-	ROR-A	AT VLIN 3,5 AT
6F	111	01101111	28416	o	F:/	-	---	VTAB
70	112	01110000	28672	p	F:0	-	BVS	= A$="HELLO"
71	113	01110001	28928	q	F:1	-	ADC-(I),Y	= A=3
72	114	01110010	29184	r	F:2	-	---	)
73	115	01110011	29440	s	F:3	-	---	Non utilise
74	116	01110100	29696	t	F:4	-	---	LIST LIST 3,5
75	117	01110101	29952	u	F:5	-	ADC-ZP,X	, LIST 3,5
76	118	01110110	30208	v	F:6	-	ROR-ZP,X	LIST LIST
77	119	01110111	30464	w	F:7	-	---	POP
78	120	01111000	30720	x	F:8	-	SEI	NODSP NODSP A$
79	121	01111001	30976	y	F:9	-	ADC-A,Y	NODSP NODSP A
7A	122	01111010	31232	z	F::	-	---	NOTRACE
7B	123	01111011	31488	{	F:;	-	---	DSP DSP A$
7C	124	01111100	31744	|	F:<	-	---	DSP DSP A
7D	125	01111101	32000	}	F:=	-	ADC-A,X	TRACE
7E	126	01111110	32256	~	F:>	-	ROR-A,X	PR#
7F	127	01111111	32512	rub	F:?	-	---	IN#

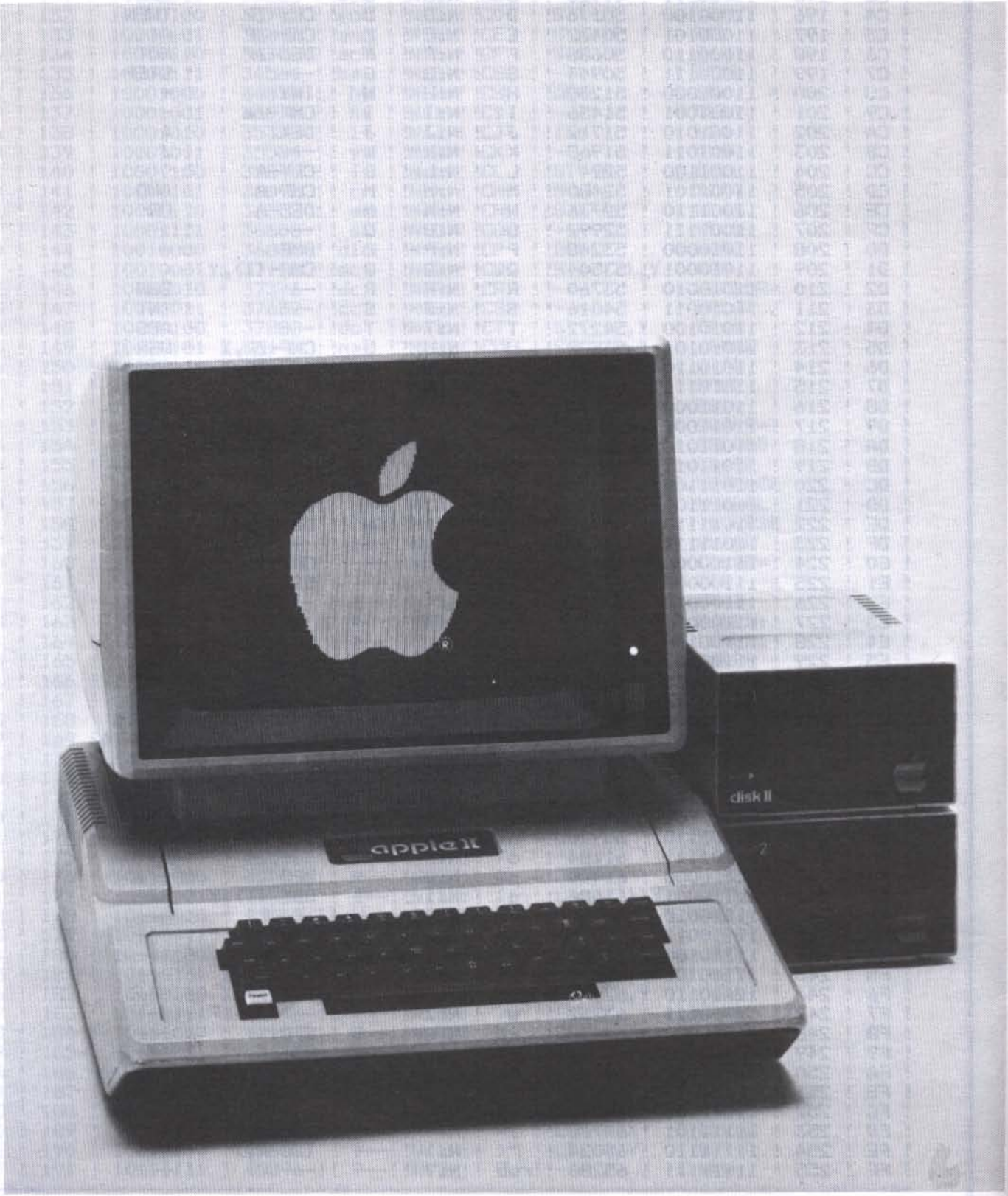
TABLEAU 3 ($80 a $BF)

HEXA	DEC	BINAIRE	DEC*256	ASCII	ECRAN	TOUCH	OPCODE-6502	TOKEN APPLESOFT
80	128	10000000	32768	nul	N:@	---	---	END
81	129	10000001	33024	soh	N:A	CA	STA-(I,X)	FOR
82	130	10000010	33280	stx	N:B	CB	---	NEXT
83	131	10000011	33536	etx	N:C	CC	---	DATA
84	132	10000100	33792	eot	N:D	CD	STY-ZP	INPUT
85	133	10000101	34048	enq	N:E	CE	STA-ZP	DEL
86	134	10000110	34304	ack	N:F	CF	STY-ZP	DIM
87	135	10000111	34560	bel	N:G	CG	---	READ
88	136	10001000	34816	bs	N:H	CH	DEY	GR
89	137	10001001	35072	ht	N:I	CI	---	TEXT
8A	138	10001010	35328	lf	N:J	CJ	TXA	PR#
8B	139	10001011	35584	vt	N:K	CK	---	IN#
8C	140	10001100	35840	ff	N:L	CL	STY-A	CALL
8D	141	10001101	36096	cr	N:M	CM	STA-A	PLOT
8E	142	10001110	36352	so	N:N	CN	STX-A	HLIN
8F	143	10001111	36608	si	N:O	CO	---	VLIN
90	144	10010000	36864	dle	N:P	CP	BCC	HGR2
91	145	10010001	37120	dc1	N:Q	CQ	STA-(I),Y	HGR
92	146	10010010	37376	dc2	N:R	CR	---	HCOLOR=
93	147	10010011	37632	dc3	N:S	CS	---	HPLOT
94	148	10010100	37888	dc4	N:T	CT	STY-ZP,X	DRAW
95	149	10010101	38144	nak	N:U	CU	STA-ZP,X	XDRAW
96	150	10010110	38400	syn	N:V	CV	STX-ZP,Y	HTAB
97	151	10010111	38656	etb	N:W	CW	---	HOME
98	152	10011000	38912	can	N:X	CX	TYA	ROT=
99	153	10011001	39168	em	N:Y	CY	STA-A,Y	SCALE=
9A	154	10011010	39424	sub	N:Z	CZ	TXS	SHLOAD
9B	155	10011011	39680	esc	N:[	esc	---	TRACE
9C	156	10011100	39936	fs	N:\	---	---	NOTRACE
9D	157	10011101	40192	gs	N:]	CSM	STA-A,X	NORMAL
9E	158	10011110	40448	rs	N:^	CSN	---	INVERSE
9F	159	10011111	40704	us	N:_	---	---	FLASH
A0	160	10100000	40960		N:	esp	LDY-Im	COLOR=
A1	161	10100001	41216	!	N:!	S1	LDA-(I,X)	POP
A2	162	10100010	41472	"	N:"	S2	LDX-Im	VTAB
A3	163	10100011	41728	#	N:#	S3	---	HIMEM:
A4	164	10100100	41984	$	N:$	S4	LDY-ZP	LOMEM:
A5	165	10100101	42240	%	N:%	S5	LDA-ZP	ONERR
A6	166	10100110	42496	&	N:&	S6	LDX-ZP	RESUME
A7	167	10100111	42752	'	N:'	S7	---	RECALL
A8	168	10101000	43008	(	N:(	S8	TAY	STORE
A9	169	10101001	43264	)	N:)	S9	LDA-Im	SPEED=
AA	170	10101010	43520	*	N:*	S:	TAX	LET
AB	171	10101011	43776	+	N:+	S;	---	GOTO
AC	172	10101100	44032	,	N:,	,	LDY-A	RUN
AD	173	10101101	44288	-	N:-	-	LDA-A	IF
AE	174	10101110	44544	.	N:.	.	LDX-A	RESTORE
AF	175	10101111	44800	/	N:/	/	---	&
B0	176	10110000	4505	0	N:0	0	BCS	GOSUB
B1	177	10110001	45312	1	N:1	1	LDA-(I),Y	RETURN
B2	178	10110010	45568	2	N:2	2	---	REM
B3	179	10110011	45824	3	N:3	3	---	STOP
B4	180	10110100	46080	4	N:4	4	LDY-ZP,X	ON
B5	181	10110101	46336	5	N:5	5	LDA-ZP,X	WAIT
B6	182	10110110	46592	6	N:6	6	LDX-ZP,Y	LOAD
B7	183	10110111	46848	7	N:7	7	---	SAVE
B8	184	10111000	47104	8	N:8	8	CLV	DEF
B9	185	10111001	47360	9	N:9	9	LDA-A,Y	POKE
BA	186	10111010	47616	:	N::	:	TSX	PRINT
BB	187	10111011	47872	;	N:;	;	---	CONT
BC	188	10111100	48128	<	N:<	<	LDY-A,X	LIST
BD	189	10111101	48384	=	N:=	=	LDA-A,X	CLEAR
BE	190	10111110	48640	>	N:>	>	LDX-A,Y	GET
BF	191	10111111	48896	?	N:?	?	---	NEW

TABLEAU 4 ($C0 a $FF)

HEXA	DEC	BINAIRE	DEC*256	ASCII	ECRAN	TOUCH	OPCODE-6502	TOKEN APPLESOFT
C0	192	11000000	49152	@	N:@	SP	CPY-Im	TAB(
C1	193	11000001	49408	A	N:A	A	CMP-(I,X)	TO
C2	194	11000010	49664	B	N:B	B	---	FN
C3	195	11000011	49920	C	N:C	C	---	SPC(
C4	196	11000100	50176	D	N:D	D	CPY-ZP	THEN
C5	197	11000101	50432	E	N:E	E	CMP-ZP	AT
C6	198	11000110	50688	F	N:F	F	DEC-ZP	NOT
C7	199	11000111	50944	G	N:G	G	---	STEP
C8	200	11001000	51200	H	N:H	H	INY	+
C9	201	11001001	51456	I	N:I	I	CMP-Im	-
CA	202	11001010	51712	J	N:J	J	DEX	*
CB	203	11001011	51968	K	N:K	K	---	/
CC	204	11001100	5224	L	N:L	L	CPY-A	^
CD	205	11001101	52480	M	N:M	M	CMP-A	AND
CE	206	11001110	52736	N	N:N	N	DEC-A	OR
CF	207	11001111	52992	O	N:O	O	---	>
D0	208	11010000	53248	P	N:P	P	BNE	=
D1	209	11010001	53504	Q	N:Q	Q	CMP-(I),Y	<
D2	210	11010010	53760	R	N:R	R	---	SGN
D3	211	11010011	54016	S	N:S	S	---	INT
D4	212	11010100	54272	T	N:T	T	---	ABS
D5	213	11010101	54528	U	N:U	U	CMP-ZP,X	USR
D6	214	11010110	54784	V	N:V	V	DEC-ZP,X	FRE
D7	215	11010111	55040	W	N:W	W	---	SCRN(
D8	216	11011000	55296	X	N:X	X	CLD	PDL
D9	217	11011001	55552	Y	N:Y	Y	CMP-A,Y	POS
DA	218	11011010	55808	Z	N:Z	Z	---	SQR
DB	219	11011011	56064	[	N:[	---	---	RND
DC	220	11011100	56320	\	N:\	---	---	LOG
DD	221	11011101	56576	]	N:]	SM	CMP-A,X	EXP
DE	222	11011110	56832	^	N:^	SN	DEC-A,X	COS
DF	223	11011111	57088		N:	---	---	SIN
E0	224	11100000	57344	_	N:_	---	CPX-Im	TAN
E1	225	11100001	57600	a	N:!	---	SBC-(I,X)	ATN
E2	226	11100010	57856	b	N:"	---	---	PEEK
E3	227	11100011	58112	c	N:#	---	---	LEN
E4	228	11100100	58368	d	N:$	---	CPX-ZP	STR$
E5	229	11100101	58624	e	N:%	---	SBC-ZP	VAL
E6	230	11100110	58880	f	N:&	---	INC-ZP	ASC
E7	231	11100111	59136	g	N:'	---	---	CHR$
E8	232	11101000	59392	h	N:(	---	INX	LEFT$
E9	233	11101001	59648	i	N:)	---	SBC-Im	RIGHT$
EA	234	11101010	59904	j	N:*	---	NOP	MID$
EB	235	11101011	60160	k	N:+	---	---	---
EC	236	11101100	60416	l	N:,	---	CPX-A	---
ED	237	11101101	60672	m	N:-	---	SBC-A	---
EE	238	11101110	60928	n	N:.	---	INC-A	---
EF	239	11101111	61184	o	N:/	---	---	---
F0	240	11110000	61440	p	N:0	---	BEQ	---
F1	241	11110001	61696	q	N:1	---	SBX-(I),Y	---
F2	242	11110010	61952	r	N:2	---	---	---
F3	243	11110011	62208	s	N:3	---	---	---
F4	244	11110100	62464	t	N:4	---	---	---
F5	245	11110101	62720	u	N:5	---	SBC-ZP,X	---
F6	246	11110110	62976	v	N:6	---	INC-ZP,X	---
F7	247	11110111	63232	w	N:7	---	---	---
F8	248	11111000	63488	x	N:8	---	SED	---
F9	249	11111001	63744	y	N:9	---	SBC-A,Y	---
FA	250	11111010	64000	z	N::	---	---	---
FB	251	11111011	64256	{	N:;	---	---	---
FC	252	11111100	64512	¦	N:<	---	---	---
FD	253	11111101	64768	}	N:=	---	SBC-A,X	---
FE	254	11111110	65024	~	N:>	---	INC-A,X	---
FF	255	11111111	65280	rub	N:?	---	---	---

"Croque moi



et tu inventeras!!

L'homme invente de plus en plus. Et de mieux en mieux
Pour repousser les limites du possible, il s'est fabriqué des outils à sa mesure.
L'ordinateur personnel Apple en est un.

Rappelez-vous. Il n'y a pas si longtemps, l'ordinateur personnel c'était un rêve. Aussi fou que de vouloir posséder son propre vaisseau spatial.

Et puis il y eut Apple.

L'informatique indépendante, abordable (un Apple coûte moins qu'un simple photocopieur) et accessible (on apprend à s'en servir en quelques heures).

Un Apple ne vient jamais seul. Avec lui, vous disposez d'une bibliothèque de programmes avec lesquels vous pouvez vous mettre tout de suite au travail. Sans avoir à apprendre le langage informatique.

Un Apple, c'est le meilleur moyen d'aller plus vite et plus loin. D'être créatif sans aucune contrainte. De regagner le temps perdu en tâches répétitives, en routine. D'aller jusqu'au bout de chaque nouvelle idée. De redevenir inventif à 100 %.

Vous faut-il d'autres bonnes raisons?

Alors, examinez une de vos journées de travail, vous en trouverez. Mais si vous savez déjà qu'un ordinateur personnel peut vous faire du bien, documentez-vous (voyez le bon à croquer Apple au bas de cette page).

Et gardez bien en tête que votre ordinateur personnel doit disposer de programmes pour vos travaux habituels. Et qu'il doit être capable de grandir en fonction de vos besoins (Apple dispose de plus d'accessoires que n'importe quel autre ordinateur personnel).

Choisissez aussi un ordinateur célèbre et qui a fait ses preuves : 400.000 Apple fonctionnent chaque jour dans le monde. C'est la meilleure preuve de leurs hautes performances et la certitude d'un service disponible sur le champ. On ne devient pas célèbre par hasard.

Un Apple, c'est vrai, change les façons de travailler, de penser, de décider.
A vous de décider.

BON A CROQUER

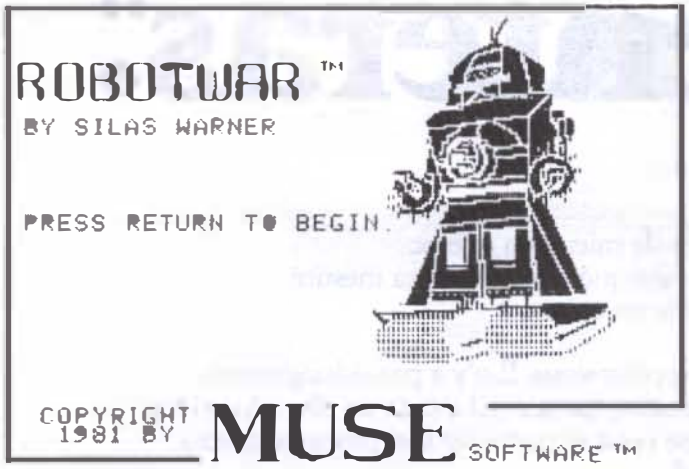
SEEDRIN
Avenue de l'Océanie Z.I. de Courtabeuf
91944 LES ULIS CEDEX

EXS 1

☐ Que lire? Si vous voulez vous familiariser avec le monde de l'ordinateur personnel à travers la littérature Apple et les revues spécialisées, cochez cette case.
☐ En français ☐ En anglais
☐ Si vous ne pouvez plus attendre, cochez cette case pour recevoir la liste des revendeurs agréés Apple.

Nom _____
Société _____
Adresse _____
_____ Code postal _____

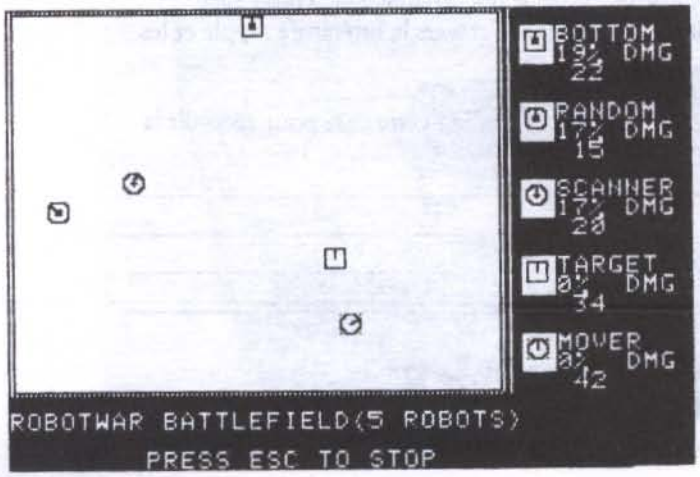
apple®
l'ordinateur personnel



ROBOTWAR - De Silas Warner - MUSE SOFTWARE
Prix indicatif : 350 FF TTC

Lorsque j'ai acheté Robotwar (la guerre des robots), il y a un peu moins d'un an, je ne me doutais pas de ce qui m'attendait ! Sous des apparences de jeu innocent, se cache un programme diabolique qui peut ruiner votre santé si vous y passez vos nuits !

De quoi s'agit-il au juste ? De concevoir un programme pour votre robot afin de le faire combattre avec d'autres robots (et de gagner, bien sûr !). Le combat se déroule dans un lieu clos de 250m sur 250m, dans lequel les robots ont tout loisir de se déplacer. Vous contemplez la scène de votre poste d'observation, situé au dessus du lieu de bataille et séparé de lui par une vitre épaisse afin d'être à l'abri des projectiles tirés par les robots ! La partie droite de l'écran est un tableau de contrôle vous donnant la liste des robots engagés, leur représentation graphique, le pourcentage des dommages (suivant les projectiles reçus) et leur score.



Comment réaliser un programme ? Tout d'abord, il faut définir la stratégie du robot. Comment va-t-il se déplacer ? Que faire lorsqu'il est touché ? Où tirer ? Ensuite, il convient de bien étudier le langage et les possibilités du robot. Vous disposez en effet d'instructions simples vous permettant d'effectuer des opérations, de tester, de faire des boucles. Votre robot dispose de registres permettant de commander sa vitesse en X et en Y, de savoir quelle est sa position, de commander le radar dans une direction, de tirer un projectile (tir réglable en direction et en longueur) et également de connaître sont taux d'endommagement (êtes-vous la cible d'un robot concurrent ?).

Le programme est entré et modifié grâce à un éditeur incorporé. Il doit ensuite être assemblé pour être compris par le robot. Vous pouvez alors évaluer sa performance grâce à un banc de test ultra perfectionné (une excellente idée). Lorsque tout est prêt, choisissez les adversaires de votre robot et enfermez-les tous ensemble. Le robot ayant la meilleure tactique et le programme le plus évolué l'emportera. A noter que vous pouvez effectuer plusieurs combats de suite afin d'avoir une évaluation plus complète de votre robot. Ces combats peuvent se dérouler de façon entièrement automatique, ce qui permet par exemple d'en faire effectuer 40 durant la nuit. Et vous découvrez au matin les scores respectifs !

Ce jeu réjouira tous les fervents de stratégie et de programmation optimisée. Une idée originale, une réalisation sans faille, cela donne un "best-seller". Compte tenu de la concurrence, Robotwar a obtenu une très honorable 20ème place dans le classement de tous les logiciels sur Apple parus depuis novembre 1980, classement effectué par Softalk auprès de ses lecteurs.

Il paraît même qu'un championnat du monde a eu lieu aux Etats-Unis. Malheureusement, je l'ai su trop tard, sinon leur compte était bon, à ces Yankees ... Mais pourquoi pas un championnat de France ? Si vous êtes intéressé, écrivez-moi et surtout programmez un robot redoutable. Nous en reparlerons dans un prochain Pom's et tâcherons d'organiser quelque chose d'ici la fin de l'année.

A vos robots !

Jean-François Duvivier, 1 rue du Sergent Blandan, 92130 Issy-les-Moulineaux.

Un PRINT USING d'intérêt général

L'une des instructions BASIC qui manque le plus à l'Applesoft est le PRINTUSING. Le programme que nous présentons permet de pallier son absence. Ecrit en Applesoft, il occupe les lignes 40000 à 40044. Afin d'avoir le moins d'interaction possible avec le programme principal, il n'utilise que des variables de deux lettres commençant par Z.

La valeur à écrire doit être placée dans la variable ZZ. Le masque de sortie est défini par ZZ$. L'appel se fait par un GOSUB 40000. Le sous-programme effectue la sortie formattée sur le périphérique en service au moment de l'appel (écran, imprimante...) et rend la main sans générer de retour à la ligne.

Le masque peut être composé de quatre caractères différents : l'espace, le #, le . et ^^^^ . Les espaces ne peuvent être incorporés qu'avant ou après les autres caractères, mais ne peuvent pas être incorporés à l'intérieur de ces caractères. Ces espaces seront automatiquement avant ou après le nombre formaté.

définit un chiffre;
. donne l'emplacement de la virgule;
^^^^ indique une sortie avec notation scientifique de la forme E +ou- et deux chiffres.

Les exemples donnés dans le tableau vous permettront de mieux comprendre le fonctionnement de ce programme. A noter que :
- aucun retour à la ligne n'est généré;
- les zéros en tête ne sont pas inscrits;
- en cas de nécessité, des zéros seront ajoutés à la fin afin de remplir le masque;
- en cas de nécessité (précision du chiffre supérieure au masque), le nombre sera arrondi au plus proche;
- si le nombre est négatif, l'une des positions # est prise pour le signe;
- si il y a plus ou moins de quatre signes ^ , le masque sera agrandi ou réduit automatiquement afin d'en laisser quatre;
- si le nombre ne tient pas dans le masque défini, ou si le masque est erroné, la sortie est formatée avec des points d'interrogation.

Utilisation

Les lignes 40000 à 40044 doivent être ajoutées au programme déjà contenu en mémoire. La solution la plus aisée consiste à faire un fichier Exec qui contienne les lignes 40000 à 40044 (voir Pom's, numéro 3). Pour cela, après avoir tapé le sous-programme de PRINTUSING, ajoutez-lui les lignes suivantes :

```
10 D$=CHR$(4):F$="PRINTUSING"
20 PRINT D$"OPEN"F$
30 PRINT D$"WRITE"F$
40 POKE 33,33
50 LIST 40000,40044
60 PRINT D$"CLOSE"
70 END
```

et faites RUN. Le fichier PRINTUSING est créé (pour ceux qui ont la disquette POM'S, ce fichier existe déjà). Pour ajouter les lignes du sous-programme PRINTUSING à votre programme, il suffit alors de le charger en mémoire par un LOAD et de taper EXEC PRINTUSING. Lorsque la disquette s'est arrêtée, vous pouvez alors faire un SAVE de votre programme complet.

Attention : vérifiez bien avant que les lignes 40000 à 40044 de votre programme sont inutilisées et que vous n'employez pas de variables commençant par Z.

```
]RUN
          PRINTUSING DEMONSTRATION

          ##.##  ###  #.###^^^^  ####.####
-10       ??.??  -10  -.100E+02   -10.0000
1.25       1.25    1  1.250E+00     1.2500
12.5      12.50   13  1.250E+01    12.5000
23.75     23.75   24  2.375E+01    23.7500
35        35.00   35  3.500E+01    35.0000
46.25     46.25   46  4.625E+01    46.2500
57.5      57.50   58  5.750E+01    57.5000
68.75     68.75   69  6.875E+01    68.7500
80        80.00   80  8.000E+01    80.0000
91.25     91.25   91  9.125E+01    91.2500
102.5     ??.??  103  1.025E+02   102.5000
```

```
JLIST

20 D$ =  CHR$ (4):F$ = "PRINTUSING
   "
30  PRINT D$;"OPEN";F$
40  PRINT D$;"WRITE";F$
50  LIST 40000,40044
60  PRINT D$;"CLOSE";F$
70  END
40000  REM    PRINT USING
40001 ZX = 0:ZW = 0:ZY = 0:ZV = 0:
    ZT = 0:ZU = 0:ZS = 0:ZR = 0:Z
    Q = ZZ:ZY$ =  STR$ (ZQ)
40002  IF  LEN (ZY$) >  = 5 THEN  IF
     MID$ (ZY$, LEN (ZY$) - 3,1) =
    "E" THEN ZV =  VAL ( RIGHT$ (
    ZY$,3)):ZY$ =  LEFT$ (ZY$, LEN
    (ZY$) - 4)
40003  FOR ZW = 1 TO  LEN (ZY$): IF
     MID$ (ZY$,ZW,1) <  > "." THEN
     NEXT ZW
40004  IF  LEN (ZY$) > 3 AND  LEFT$
    (ZY$,3) = "-.0" THEN ZY$ = "-
    ." +  RIGHT$ (ZY$, LEN (ZY$ -
    3):ZV = ZV - 1: GOTO 40007
40005  IF ZW = 1 THEN ZY$ =  RIGHT$
    (ZY$, LEN (ZY$) - 1): IF  LEFT$
    (ZY$,1) = "0" THEN ZY$ =  RIGHT$
    (ZY$, LEN (ZY$) - 1):ZV = ZV -
    1
40006  IF ZW = 1 THEN 40009
40007  IF ZW <  LEN (ZY$) THEN ZY$
     =  LEFT$ (ZY$,ZW - 1) +  RIGHT$
    (ZY$, LEN (ZY$) - ZW)
40008 ZV = ZV + ZW - 1
40009  FOR ZI = 1 TO  LEN (ZZ$): IF
     MID$ (ZZ$,ZI,1) = " " THEN  PRINT
    " ";: NEXT ZI: GOTO 40039
40010  FOR ZJ = ZI TO  LEN (ZZ$): IF
     MID$ (ZZ$,ZJ,1) = "$" THEN Z
    T = ZT + 1: NEXT ZJ: GOTO 400
    15
40011  IF  MID$ (ZZ$,ZJ,1) = "." THEN
    ZS = 1:ZJ = ZJ + 1: IF ZJ >  LEN
    (ZZ$) THEN  GOTO 40015
40012  FOR ZK = ZJ TO  LEN (ZZ$): IF
     MID$ (ZZ$,ZK,1) = "^" THEN Z
    R = 1: GOTO 40015
40013  IF  MID$ (ZZ$,ZK,1) = "$" THEN
    ZU = ZU + 1: NEXT ZK: GOTO 40
    015
40014  IF  MID$ (ZZ$,ZK,1) <  > "
    " THEN 40039
40015  IF  LEN (ZY$) < ZU + ZT + 1
     THEN  FOR ZI =  LEN (ZY$) TO
    ZU + ZT + 1:ZY$ = ZY$ + "0": NEXT
    ZI
40016  IF  VAL (ZY$) = 0 THEN ZV =
    1: GOTO 40021
40017 ZP = ZV + ZU + 1: IF ZR = 1 THEN
    ZP = ZT + ZU + 1
40018  IF ZP <  = 0 THEN 40021
40019 ZY$ =  STR$ ( VAL ( LEFT$ (Z
    Y$,ZP)) + 5 *  SGN (ZZ))
40020  IF  LEN (ZY$) > ZP THEN ZV =
    ZV + 1
40021  IF ZR <  > 0 THEN  GOTO 400
    30
40022  IF ZV > ZT THEN 40039
40023 ZX$ = "": IF ZZ < 0 THEN ZY$
     =  RIGHT$ (ZY$, LEN (ZY$) -
    1):ZX$ = "-"
40024  IF ZV <  = 0 THEN ZY$ = "0"
    + ZY$:ZV = ZV + 1: GOTO 4002
    4
40025 ZY$ = ZX$ + ZY$
40026  IF ZT - ZV > 0 THEN  FOR ZI
     = 1 TO ZT - ZV: PRINT " ";:Z
    T = ZT - 1: NEXT ZI
40027  IF ZT > 0 THEN  PRINT  LEFT$
    (ZY$,ZT);
40028  IF ZS <  > 0 THEN  PRINT ".
    ";: IF ZU > 0 THEN  FOR ZI =
    ZT + 1 TO ZT + ZU: PRINT  MID$
    (ZY$,ZI,1);: NEXT ZI
40029  GOTO 40037
40030  PRINT  LEFT$ (ZY$,ZT);:ZV =
    ZV - ZT
40031  IF ZS <  > 0 THEN  PRINT ".
    ";: IF ZU > 0 THEN  FOR ZI =
    ZT + 1 TO ZT + ZU: PRINT  MID$
    (ZY$,ZI,1);: NEXT ZI
40032  IF  VAL (ZY$) = 0 THEN ZV =
    0
40033  PRINT "E";: IF ZV < 0 THEN
     PRINT "-"
40034  IF ZV >  = 0 THEN  PRINT "+
    ";
40035  IF  ABS (ZV) < 10 THEN  PRINT
    "0"; ABS (ZV);
40036  IF  ABS (ZV) > 9 THEN  PRINT
     ABS (ZV);
40037  FOR ZI =  LEN (ZZ$) TO 1 STEP
    - 1: IF  MID$ (ZZ$,ZI,1) = "
    " THEN  PRINT " ";: NEXT ZI
40038  RETURN
40039  FOR ZI = 1 TO  LEN (ZZ$)
40040  IF  MID$ (ZZ$,ZI,1) = "V" THEN
     PRINT " ";: GOTO 40044
40041  IF  MID$ (ZZ$,ZI,1) = "." THEN
     PRINT ".";: GOTO 40044
40042  IF  MID$ (ZZ$,ZI,1) = "$" THEN
     PRINT "?";: GOTO 40044
40043  IF  MID$ (ZZ$,ZI,4) = "^^^^
    " THEN  PRINT "E???";:ZI = ZI
     + 3: GOTO 40044
40044  NEXT ZI: RETURN
```

Notions de base : les fichiers

En version standard, le DOS de l'Apple II n'offre pas d'utilitaires de gestion de fichier. Il vous appartient donc de définir par vous-même l'organisation de vos fichiers, les modes d'accès, les méthodes de recherche, etc. Les réponses à ce type de problème sont extrêmement variées. Cet article n'en présentera que quelques-unes, relativement simples à mettre en oeuvre et ne faisant appel qu'aux seules ressources du BASIC Applesoft, sans s'attarder sur les structures purement séquentielles dont l'intérêt et la difficulté semblent assez réduits.

1. Fichiers séquentiels à accès direct avec table de référence

a) Table de référence complète en mémoire

Supposons que l'on veuille écrire un programme permettant de gérer un stock d'articles quelconques. Indépendamment des divers traitements à réaliser, ce qui nous intéresse ici est de pouvoir stocker et récupérer rapidement les données concernant un article précis (stock, prix unitaire ...).

Poussons donc l'audace jusqu'à doter chacun de nos articles d'un code, long par exemple de 6 caractères alphanumériques, et qui nous servira de clé d'accès. Le principe de gestion des fichiers se décompose alors en trois étapes :

. Etablir une table des codes-articles, classée par ordre alphabétique.
. Associer à chaque code-article une adresse renvoyant au fichier qui mémorise les données.
. Ecrire ou lire les données à l'aide de ce fichier.

Seule la table de référence, comportant les codes et les adresses, réside en mémoire centrale. Par ailleurs, elle sera stockée sur disquette dans un fichier séquentiel à accès séquentiel.

Le programme suivant donne un exemple d'utilisation de cette méthode.

```
]LIST

1  GOTO 1000
30  REM
40  REM  **********************
         *      RECHERCHE     *
         *   DICHOTOMIQUE     *
         **********************

50 K1 = 1:K2 = NZ: IF NZ = 0 THEN
     K = 1: RETURN
51 K =  INT ((K1 + K2) / 2):A$ =
     LEFT$ (N$(K),6): IF S$ = A$
     THEN K1 =  - 1: RETURN
52  IF S$ < A$ AND K2 > K1 + 1 THEN
     K2 = K: GOTO 51
53  IF S$ < A$ THEN  RETURN
54  IF K2 > K1 + 1 THEN K1 = K: GOTO
     51
55 K = K + 1: IF S$ >  LEFT$ (N$(
    K2),6) THEN K = K2 + 1
56  IF S$ =  LEFT$ (N$(K2),6) THEN
     K1 =  - 1:K = K2: RETURN
57  RETURN
60  REM
200  DATA  CREATION,CONSULTATION,
    ANNULATION,FIN
210  DATA  DONNEE 1,DONNEE 2,DONN
    EE 3,DONNEE 4,DONNEE 5,DONNE
    E 6
220  FOR I = 1 TO 4: READ ME$(I):
     NEXT : FOR I = 1 TO 6: READ
    LI$(I): NEXT : RETURN
300  VTAB 22: HTAB 1: CALL  - 868
    : VTAB 22: HTAB 1: INVERSE :
     PRINT Z$: NORMAL : FOR Z =
    1 TO 2000: NEXT : RETURN
980  REM
990  REM  **********************
         *    DEBUT DU        *
         *    PROGRAMME       *
         **********************

1000 D$ =  CHR$ (4):B$ = "      "
    : ONERR  GOTO 9900
1010  DIM N$(100),L%(50)
1015  GOSUB 200
1020  TEXT : HOME : FOR I = 1 TO
    4: VTAB 2 + 2 * I: HTAB 1: PRINT
    I" - "ME$(I): NEXT : VTAB 22
    : HTAB 1: INPUT "VOTRE CHOIX
     ? ";CH: IF CH < 1 OR CH > 4
     THEN 1020
```

```
1030  IF CH = 4 THEN 8000
1040  IF NZ <  > 0 THEN 1070
1045  REM
1046  REM  ********************
           * LECTURE DE LA   *
           *   TABLE DE      *
           *   REFERENCE     *
           ********************

1050 E% = 1:Z = 0: PRINT D$"OPEN
     TABLE": PRINT D$"READ TABLE"
     : INPUT NZ: FOR I = 1 TO NZ:
      INPUT N$(I): NEXT : INPUT N
     L: IF NL = 0 THEN 1055
1052  FOR I = 1 TO NL: INPUT L%(I
     ): NEXT
1055  PRINT D$"CLOSE":E% = 0: IF
     Z = 5 THEN NZ = 0:NL = 0
1060  PRINT D$"PR#0"
1070  TEXT : HOME :Z$ = ME$(CH): HTAB
     20 -  LEN (Z$) / 2: INVERSE
     : PRINT Z$: NORMAL
1075  VTAB 4: HTAB 1: PRINT "CODE
      ARTICLE : ";: VTAB 4: HTAB
     16: INPUT "";CA$: IF  LEN (C
     A$) > 6 THEN 1075
1080 S$ =  LEFT$ (CA$ + B$,6): GOSUB
     50: ON CH GOTO 1100,2000,300
     0
1085  REM
1086  REM  ********************
           *    CREATION     *
           ********************

1100  IF K1 =  - 1 THEN Z$ = "EXI
     STE": GOSUB 300: GOTO 1020
1110  FOR I = 1 TO 6: VTAB 5 + I:
      HTAB 1: PRINT LI$(I)" : ": NEXT
     : FOR I = 1 TO 6: VTAB 5 + I
     : HTAB  LEN (LI$(I)) + 4: INPUT
     "";DD$(I): NEXT : VTAB 22: HTAB
     1: INPUT "OK ? ";Z$: IF Z$ =
     "N" THEN 1110
1120 E% = 9:R = NZ + 1: IF NL >
     0 THEN R = L%(NL):NL = NL -
     1
1125  IF K <  = NZ THEN  FOR I =
     NZ TO K STEP  - 1:N$(I + 1) =
     N$(I): NEXT
1130 NZ = NZ + 1:N$(K) = S$ +  STR$
     (R): PRINT D$"OPEN ARTICLES,
     L100": PRINT D$"WRITE ARTICL
     ES,R"R: PRINT S$: FOR I = 1 TO
     6: PRINT DD$(I): NEXT : PRINT
     D$"CLOSE": PRINT D$"PR#0": GOTO
     1020
1180  REM
1190  REM  ********************
           *  CONSULTATION   *
           ********************

2000  IF K1 > 0 THEN Z$ = "N'EXIS
     TE PAS": GOSUB 300: GOTO 102
     0
2005 E% = 9 * (E% = 9)
2010  GOSUB 2020: GOTO 1020
2020 R =  VAL ( MID$ (N$(K),7)): PRINT
     D$"OPEN ARTICLES,L100": PRINT

     D$"READ ARTICLES,R"R: INPUT
     CA$: FOR I = 1 TO 6: INPUT D
     0$(I): NEXT : PRINT D$"CLOSE
     ": PRINT D$"PR#0"
2030  FOR I = 1 TO 6: VTAB 5 + I:
      HTAB 1: PRINT LI$(I)" : "DD
     $(I): NEXT : VTAB 22: HTAB 1
     : PRINT "FRAPPEZ UNE TOUCHE"
     ;: GET Z$: RETURN
2980  REM
2990  REM  ********************
           *   ANNULATION    *
           ********************

3000  IF K1 > 0 THEN 2000
3010  GOSUB 2020: VTAB 22: HTAB 1
     : CALL  - 868: VTAB 22: HTAB
     1: INPUT "ANNULATION CONFIRM
     EE ? ";Z$: IF Z$ <  > "D" THEN
     1020
3020 E% = 9:NZ = NZ - 1: IF K <
      = NZ THEN  FOR I = K TO NZ:
     N$(I) = N$(I + 1): NEXT
3030  PRINT D$"OPEN ARTICLES,L100
     ": PRINT D$"WRITE ARTICLES,R
     "R: PRINT "ANNULE": PRINT D$
     "CLOSE": PRINT D$"PR#0":NL =
     NL + 1:L%(NL) = R: GOTO 1020



7980  REM
7990  REM  ********************
           *  SAUVEGARDE DE  *
           *  LA TABLE DE    *
           *   REFERENCE     *
           ********************

8000  IF E% <  > 9 THEN  END
8010  PRINT O$"OPEN TABLE": PRINT
     D$"WRITE TABLE": PRINT NZ: FOR
     I = 1 TO NZ: PRINT N$(I): NEXT
     : PRINT NL: FOR I = 1 TO NL:
      PRINT L%(I): NEXT : PRINT D
     $"CLOSE": PRINT D$"PR#0": END

9900 Z =  PEEK (222): IF Z = 5 AND
     E% = 1 THEN 1055
9910  PRINT D$"CLOSE": IF Z = 5 OR
     Z = 6 THEN  PRINT "CETTE DIS
     QUETTE N'EST PAS LA BONNE": FOR
     Z = 1 TO 1500: NEXT : GOTO 1
     020
9920  PRINT "ERREUR NO "Z" OANS L
     A LIGNE " PEEK (218) + 256 *
      PEEK (219): PRINT : PRINT "
     FRAPPEZ UNE TOUCHE";: GET Z$
     : GOTO 1020
```

Il ne s'agit là que d'un programme plutôt rudimentaire, mais seules nous importent vraiment les méthodes de gestion des fichiers, qui méritent quelques commentaires:

-- TABLE est le fichier séquentiel où se trouve stockée la table de référence. En mémoire, la table est chargée dans le tableau N$.

-- Le tableau L% contient les adresses libérées par l'annulation d'articles qui sont donc disponibles pour de nouveaux enregistrements. Sur disquette, ces adresses sont stockées à la suite de la table de référence dans le même fichier TABLE.

-- ARTICLES est le fichier à accès direct où sont stockées les données concernant les différents articles. La longueur des enregistrements (ici fixée à 100 de façon non significative), dépend bien sûr du nombre des données et du nombre de caractères que comporte chacune d'elles.

-- Dans la procédure de recherche dichotomique (lignes 50 à 57), K indique la position d'un code donné dans la table (position virtuelle si l'article en question n'a pas encore été créé). K1 est fixé à -1 si le code-article existe; il est toujours positif dans le cas contraire. La longueur d'un code dans la table est forcée à 6 caractères par ajout de blancs si nécessaire. L'adresse corres- pondante dans ARTICLES se trouve simplement à la suite du code; ceci évite d'avoir à gérer deux tableaux pour les codes et les adresses (Cf lignes 1080, 1130). Pour retrouver l'adresse, il suffit de prendre la partie de la chaîne qui se trouve au-delà du sixième caractère.

-- En cas de création d'un article dont le code n'est pas le dernier de la liste dans l'ordre alphabétique, il faut décaler la table en conséquence (ligne 1125). De même si l'on annule un code qui n'est pas le dernier de la liste (ligne 3020).

-- En fin de programme, si la table de référence a été modifiée (ajouts ou supressions), il faut la sauver sur disquette (lignes 8000-8010). La variable EE% signale ces éventuelles modifications.

b) Table de référence "tronquée" en mémoire

Cette méthode repose sur les mêmes principes que la précédente, mais peut s'appliquer à des situations dans lesquelles le nombre d'éléments de la table de référence et la longueur des codes rendent problématique le stockage des deux données "code + adresse" en mémoire centrale.

La solution envisagée ici consiste alors à ne garder en mémoire que les codes (ce qui permet d'effectuer des recherches rapides) et à utiliser un fichier à accès direct, dont les enregistrements sont classés dans l'ordre des codes, contenant les adresses.

En reprenant le programme ci-dessus, la récupération des données s'opère comme suit (ADRES étant le fichier des adresses) :

```
2020 PRINT D$"OPEN ADRES,L5" : PRINT D$"READ
ADRES,R"K : INPUT R : PRINT D$"OPEN ARTICLES,
L100" : ...
```

Dans la procédure de recherche dichotomique, il n'est plus nécessaire de préciser LEFT$(...,6), puisque N$ ne contient plus que les codes.

En ce qui concerne la mise à jour du fichier des adresses, deux solutions sont envisageables :

** Mise à jour immédiate : en création par exemple, on aurait alors :

```
1125 PRINT D$"OPEN ADRES,L5":IF K>NZ THEN
GOTO 1128
1126 FOR L=NZ TO K STEP-1 : R1=L : R2=L+1 :
PRINT D$"READ ADRES,R"R1 : INPUT Z : PRINT
D$"WRITE ADRES,R"R2 : PRINT Z : NEXT
1127 FOR L=NZ TO K STEP-1:N$(L+1)=N$(L):NEXT
1128 PRINT D$"WRITE ADRES,R"K : PRINT R
1130 NZ=NZ+1 : N$(K)=S$ : ...
```

C'est la solution la plus simple à mettre en oeuvre, mais elle risque parfois d'obliger l'utilisateur à patienter un certain temps devant son écran si l'article créé ou supprimé se trouve en début d'une liste longue.

** Mise à jour en fin de traitement : les décalages du fichier des adresses sont alors effectués à l'issue d'une phase de création ou de suppression.

On peut, par exemple, stocker les codes et les adresses des articles créés dans un tableau annexe puis, en fin de traitement, reprendre chaque code, localiser sa position K dans la table et mettre à jour le fichier des adresses.

2. Fichiers non séquentiels à accès direct

Si nous rebaptisons nos articles "composants", on peut admettre qu'ils entrent dans la nomenclature de produits plus complexes, et que l'on veuille également gérer ces nomenclatures sur Apple. Une de ces nomenclatures se présenterait sous la forme d'une liste de codes-articles, avec pour chacun d'eux la quantité du composant correspondant qui entre dans la composition du produit concerné. Cette liste étant susceptible d'évoluer, il faudrait en outre pouvoir modifier rapidement une quantité ou un code donnés.

Là encore, bien des solutions sont envisageables. Celle que nous examinons ici suppose que l'on sache quel est le nombre

maximum des composants par nomenclature et que l'utilisation non optimale de la capacité des disquettes ne soit pas pour nous un problème fondamental. Supposons donc que l'on mette 10 nomenclatures (désignées par un code à 2 caractères) par disquette et que chacune comporte au maximum 100 composants.

On affecte à chaque nomenclature un chiffre compris entre 0 et 9 (C%) et l'on utilise un fichier à accès direct dans lequel les enregistrements 0 à 100 (1 à 100 pour les composants) correspondent à la première nomenclature, les enregistrements 101 à 201 (102 à 201 pour les composants) à la seconde, et ainsi de suite.

Pour chaque nomenclature, le premier enregistrement se trouve à l'adresse RO=101*C% ; on peut y stocker le code et le nombre de composants (NC) dont les données sont mémorisées dans les enregistrements RO+1 à RO+NC. En outre, on conservera en mémoire la liste des codes-nomenclatures avec les chiffres C% correspondants (qui sera stockée dans un fichier séquentiel).

Voici quelques procédures liées à ce type d'organisation :

** Affectation d'un C% à une nomenclature nouvelle, après avoir vérifié que NN est inférieur à 10 et que CN$ n'existe pas déjà (NN=nombre de nomenclatures, NO$=table des nomenclatures, CN$=code de la nomenclature) :

```
10 CN$=LEFT (CN$+"  ",2)
20  FOR  I=0  TO  9 : Z=0 : FOR J=1 TO NN : IF
VAL(MID$(NO$(J),3))=I THEN Z=1 : J=NN
30 NEXT : IF Z=0 THEN C%=I : I=9 : RO=101*C%
40 NEXT : NN=NN+1:NO$(NN)=CN$+STR$(C%): ...
```

** Création d'un composant (NOMENC=fichier des nomenclatures) :

```
10  NC=NC+1  : R=RO+NC : PRINT D$"OPEN NOMEMC,
L100"  : PRINT D$"WRITE NOMENC,R"R : PRINT ...
(données)
```

** Récupération des données d'un composant situé en Ième position dans la liste :

```
10  R=RO+I  :  PRINT  D$"OPEN  NOMENC,L100"  :
PRINT D$"READ NOMENC,R"R : INPUT ...
```

3. Fichiers à accès direct avec pointeurs

Gardons notre problème de nomenclatures à titre d'exemple. Une autre façon de le résoudre pourrait se décomposer ainsi :

a) Création d'une table de référence, avec codes et adresses (cf. supra)

b) Création d'un fichier à accès direct (cf. supra) mémorisant, pour chaque nomenclature, le code, le libellé,..., le nombre de composants et les adresses des premier et dernier de la nomenclature dans le fichier des composants.

c) Création d'un fichier des composants (séquentiel à accès direct) mémorisant pour chacun d'entre eux le code-article, la quantité et l'adresse du composant suivant pour la même nomenclature. Un caractère spécial à la place de cette adresse signale que l'on est arrivé au dernier composant d'une nomenclature donnée.

d) Il faut en outre stocker dans un fichier le nombre total de composants enregistrés (toutes nomenclatures confondues), afin de savoir à quelle adresse écrire les données d'un composant nouvellement créé.

L'utilisation de ce type de structure ne pose guère de problèmes. Soient:

-- NT : nombre total de composants
-- A1 : adresse du premier composant pour une nomenclature donnée
-- A2 : adresse du dernier composant pour une nomenclature donnée
-- AS : adresse du composant suivant pour la même nomenclature (0 si dernier composant)
-- COMPO : fichier des composants
-- NOMENC : fichier des nomenclatures

** Création d'un composant :

. enregistrement des données dans COMPO à l'adresse NT+1, avec AS=0
. enregistrement de AS=NT+1 pour le composant situé à l'adresse A2
. enregistrement de A2=NT+1 dans NOMENC

** Recherche d'un composant :

. lecture de l'enregistrement situé à l'adresse A1 (on connait donc AS). Si le code-article est celui que l'on recherche, tant mieux !
. sinon, lecture de l'enregistrement AS ..., et ainsi de suite, jusqu'à ce que l'on trouve ce que l'on cherche, ou bien AS=0.

Si l'on doit accéder rapidement à un composant donné, cette méthode est certainement moins efficace que la précédente qui, à partir d'une liste imprimée des différents composants de la nomenclature avec leur numéro d'ordre, permet d'accomplir ce genre de recherche sans perte de temps.

Mais la méthode des "pointeurs" peut trouver à s'appliquer dans de nombreuses situations :

. ignorance du nombre maximum d'éléments par chapitre (nos nomenclatures)
. souci de ne pas "gaspiller" de place sur la disquette
. mémorisation d'un nombre limité d'éléments par chapitre, avec roulement (on mémorise par exemple 10 évènements, l'enregistrement d'un onzième provoque l'écrasement du premier). Il suffit en effet de modifier les différentes adresses (dont A1 qui devient l'adresse du deuxième) et d'enregistrer le nouvel élément à l'ancienne adresse du premier d'entre eux.

La 7e W.C.C.F. (San Francisco)

La septième West Coast Computer Faire (WCCF) à San Francisco vient de confirmer, avec ses 100 conférenciers, 600 exposants et 40.000 visiteurs en trois jours, le dynamisme de l'industrie américaine de l'informatique individuelle. N'oublions pas que 80% des ordinateurs individuels dans le monde se trouvent aux Etats-Unis; par ailleurs, 80% des ordinateurs individuels vendus dans le monde en 1981 ont été fabriqués aux Etats-Unis. Le marché américain devrait dépasser un million d'unités en 1985, contre 300.000 en 1981. A la 7ème WCCF, le dynamisme le plus marquant était celui de tous les développements réalisés autour de l'Apple, que ce soit pour ses périphériques, ses logiciels d'application ou ses logiciels de base.

Les périphériques

Il serait difficile de dresser une liste exhaustive des périphériques qu'il est possible de relier à un Apple. Toutefois, parmi les derniers apparus sur le marché, il est intéressant de noter le succès de quelques-uns d'entre eux :

- la reconnaissance vocale, avec 64 mots pour 300 dollars;
- la synthèse vocale à partir d'un texte rentré au clavier (375 dollars);
- la musique synthétique d'une très grande qualité;
- les périphériques de communication et l'annonce de réseaux locaux;
- les disques durs 5 pouces et 8 pouces;
- les imprimantes graphiques.

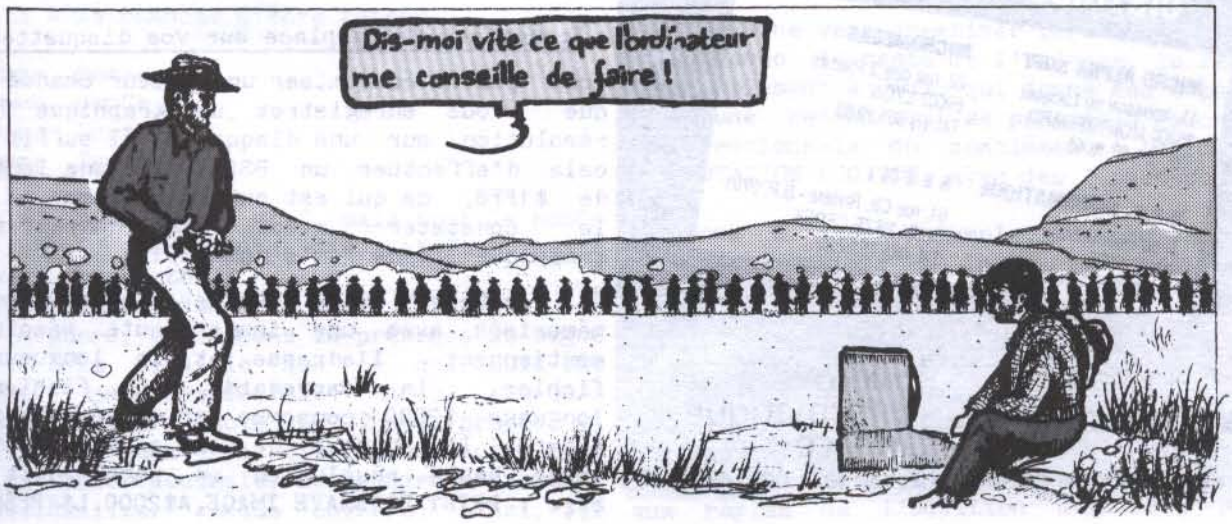
Logiciel d'application pour DOS 3.2 ou 3.3

Environ 5.000 logiciels d'application pour l'Apple II sont actuellement commercialisés sur le marché américain. Ces logiciels peuvent être classés en trois grands domaines : les logiciels professionnels, les logiciels éducatifs, et les jeux.

Quatre types de logiciels professionnels connaissent un vif succès :

1) Les logiciels de traitement de texte, dont les performances deviennent comparables à celles de machines dédiées : pagination automatique, possibilité de souligner, d'inverser des paragraphes, de disposer (en anglais) d'un dictionnaire de 50.000 mots avec avertissement des fautes d'ortographe éventuelles, aide à la mise au point de formes grammaticales, à la mise en page automatique de lettres, ... Une vingtaine de logiciels de traitement de texte pour Apple étaient en démonstration, dont les principaux étaient SuperScribe II, Magic Window, Apple Writer II, Executive Secretary, Easy Writer, Letter Perfect, Wordstar, Write-on et Zardax.

2) Les logiciels d'aide à la décision dont les publicités vantent le temps considérable qu'ils font gagner à leurs utilisateurs pour la mise au point d'un budget, l'analyse d'une



enquête ou la simulation des effets d'un changement de paramètre. Parmi ces logiciels, une dizaine se sont imposés dont Visicalc, Personal Finance Manager, Tax Planner, Micro Planner et Micro Finesse. Ce type de logiciel connait un succès considérable outre-Atlantique.

3) Les logiciels de gestion de base de données, qui permettent à un non-informaticien de travailler sur un fichier en bénéficiant d'énormes possibilités. Parmi les logiciels les plus connus, on retrouve DB Master, PFS, CCA DMS, Data Factory et File Fax.

4) Les logiciels de gestion qui sont des versions adaptées à l'Apple des logiciels déjà rôdés sur mini-ordinateur (comptabilité, facturation, paye, ...). Plusieurs sociétés se sont spécialisés dans la conception de ce type de logiciel, en particulier Micro Lab, Continental Software, Broderbund et Peachtree.

Parmi les logiciels éducatifs, ce sont les logiciels réalisés pour l'apprentissage des disciplines scientifiques qui connaissent le plus grand succès, tels les produits de North Strategy ou de Northware pour les mathématiques. Les didacticiels de lecture et/ou écriture comme Magic Spell marchent aussi très bien. Enfin, il faut souligner l'intérêt de logiciels d'apprentissage de disciplines non scientifiques : Apple Music Theory en musique, Supermap en géographie, ou Stellar Astronomy pour apprendre à repérer les étoiles. Et la liste n'est en rien exhaustive ! Dans le domaine de l'éducation, remarquons aussi la percée du langage Logo à la 7è WCCF.

Quand aux logiciels de jeu, ils connaissent une floraison exceptionnelle et atteignent des niveaux de performance remarquables.

En complément de cette brève revue, n'oublions pas le développement de la carte Z 80, deux cartes 8088 avec MS DOS rendant l'Apple compatible avec l'ordinateur individuel IBM, les extensions mémoire 128K et la diffusion de plus en plus grande du langage Forth.

Economisez de la place sur vos disquettes

Vous pouvez économiser un secteur chaque fois que vous enregistrez un graphique haute résolution sur une disquette. Il suffit pour cela d'effectuer un BSAVE avec une longueur de $1FF8, ce qui est suffisant comme on peut le constater avec "Les adresses du graphique", dans Pom's numéro 1.

En effet, les quatre premiers octets mémorisés avec une image haute résolution contiennent l'adresse et la longueur du fichier. La sauvegarde d'un fichier de longueur $2000 consomme par conséquent $2004 octets. La syntaxe de la sauvegarde d'une image haute résolution en page 1 doit donc être : PRINT D$"BSAVE IMAGE,A$2000,L$1FF8".

Apprentissage de l'assembleur (II)

Notre propos n'est pas ici d'écrire un manuel sur l'assembleur : d'une part, je ne m'en sens pas la compétence; d'autre part, il faudrait que Pom's puisse paraître à une fréquence plus grande. Mais surtout, comme vous avez pu le constater par la bibliographie de mon premier article (Pom's 3), nous avons l'embarras du choix en matière de bons manuels; je vous cite pour mémoire le tome III de "La pratique de l'Apple II", aux Editions du PSI.

Cependant, un bon manuel n'est pas tout; il faut la pratique, et la meilleure manière de l'acquérir consiste à décortiquer des programmes déjà écrits. C'est précisément ce que je vous propose, du moins dans le précédent article et le prochain, en attirant votre attention sur les pierres d'achoppement. Les difficultés, je les connais, je les ai rencontrées !

Dans cette deuxième partie, je m'adresse plus particulièrement aux programmeurs moins expérimentés, en exposant à ma manière les points qui m'ont fait souffrir. Puissent les plus expérimentés parmi les lecteurs, qui auront le courage de poursuivre cette lecture, y découvrir certaines vérités qui leur avaient peut-être échappé !

Pensez hexa !

L'hexadécimal, ce n'est pas diffile, et ça peut rapporter gros ...

Pour ceux qui partagent ma mauvaise mémoire, un procédé mnémotechnique peut rendre de fiers services. Plus le truc semble ridicule, plus il a de chances d'être retenu :

B comme Bonze
C comme douCe
D comme Dreize
E comme Equateurze

Je n'ai rien pour A et F, mais tout le monde sait que A représente dix, et F est seize moins un, ou plutôt :

$F=$10-$1 ("Un-Zéro moins Un")

Tout d'abord, soulignons la présence du signe "$" : cela signifie que ce qui suit est exprimé en hexadécimal. Enfin, pourquoi énoncer "Un-Zéro moins Un" ? A ce propos, il est utile, quand on utilise une autre base que la base décimale, de prendre l'habitude d'individualiser chaque chiffre: ainsi, $16 n'est pas Seize, mais Un-Six. C'est un pli à prendre, qui évite la confusion avec les nombres décimaux, surtout quand le nombre hexa concerné ne comporte pas de lettre. Rassurez-vous, au niveau de l'Apple, il n'y a jamais plus de quatre chiffres hexa ensemble!

Pensez binaire

Chaque fois que j'ai un problème, je me reporte à la représentation binaire des valeurs que je dois traiter. C'est toujours plus simple de construire quelque chose à partir de bonnes définitions, et de revenir à celles-ci quand on est en panne.

Par exemple, tout le monde sait que, dans la représentation négative d'un nombre, il y a un bit dans la position la plus significative du mot (en l'occurence de l'octet, quand on travaille en simple précision). Mais comment trouve-t-on l'opposé de 00100101 ? Dans tous les bons manuels, on parle de complément à Deux, avec une démonstration dont le cheminement m'échappe régulièrement. Or, l'addition en binaire étant très facile, il me suffit de me rappeler que l'addition d'un nombre et de son opposé donne 0. Je n'ai plus qu'à trouver un nombre qui, additionné au nombre de départ, donne des 0 partout :

```
  00100101
+ 11011010
----------
1 00000000
```

Il y a un 1 qui apparait dans la position la plus à gauche, mais comme il disparait de l'octet, je n'ai pas à m'en préoccuper, du moins tant que je n'ai pas à tester les indicateurs du MOT D'ETAT.

REM - Je ne veux chagriner personne et reconnais, avec le reste de l'univers, la réalité du complément à Deux, qui donne des 0 partout avec une retenue qui se propage à gauche, et différencions-la du complément à Un, ou COMPLEMENTATION LOGIQUE, avec des 1 partout.

Revenons à notre exemple; la même opération en hexa consiste à faire le complément à $100:

```
  25
+ DB
----
1 00
```

D'ailleurs, la représentation négative de l'octet a été choisie précisément pour obéir aux règles de l'addition utilisées par le

microprocesseur. Ce dernier ignore complètement les nombres négatifs. Il se préoccupe simplement de savoir si le bit le plus significatif de l'octet est positionné (bit 7). Si tel est le cas, il positionnera de la même manière le bit le plus significatif du mot d'etat, bit 7, que nous autres humains, nous empressons d'appeler N, ou indicateur de signe négatif (ce que se refuse à faire Lance Leventhal dans son manuel - cf. Biblio. - 2)

Il ne vous a pas échappé, à la lecture du précédent article que cet indicateur servait à bien d'autres choses que de déterminer si un octet est négatif. Le moniteur l'utilise par exemple savoir si un caractère est frappé sur le clavier, l'octet de l'accumulateur prenant immédiatement un bit "Un" en position la plus significative, alors que, pour permettre la lecture il avait été mis à zéro précédemment par "STROBE" (Octet $C010 de la mémoire). Il n'y a rien de bien négatif dans tout celà !

Les limitations du "huit-bits"

La mémoire de l'Apple II fait au maximum 64K. Cela se représente en binaire par un "Un" suivi de seize zéros. Comme on utilise l'adresse zéro, on voit qu'il faut seize bits pour adresser toute la mémoire. On ne s'étonnera pas de savoir que le registre qui contient l'adresse de l'instruction suivante -le "Registre Contrôleur de Séquence", ou "Compteur Ordinal" (PC pour Program Counter)- est un registre de deux octets. Il est d'ailleurs le seul de ce genre. L'accumulateur et tous les autres registres sont des huit-bits.

On conçoit la difficulté qu'il y a à manipuler des adresses de seize bits à l'aide de registres et de mémoires qui n'ont que huit bits: il faut à chaque fois doubler les instructions de transfert du type LDA et STA (quand on passe par l'accumulateur). Il faut donc s'habituer, quand on lit ou écrit un programme, à saisir ces instructions par groupes de quatre et à réserver en mémoire deux adresses contiguës pour stocker les adresses. Pour tout simplifier, la partie la plus significative de l'adresse se place généralement dans le second octet (il parait que cela permet une exécution plus rapide!). Remarquez que votre grande pratique des programmes en BASIC vous a familiarisés avec les instructions du genre:

```
40 LET FIN = PEEK(30)+PEEK(31)*256
50 POKE FIN,96
```

ce qui n'est en fait que du langage machine transposé en Applesoft.

Pour fixer les idées, supposons que nous soyons en train de fabriquer un fichier binaire en mémoire et que nous voulions mettre un caractère "fin de fichier" quand celui-ci sera rempli (91 en décimal, ou $60 en hexa, est effectivement le symbole de fin de fichier pour le programme AppleWriter). Nous sommes obligés d'avoir un compteur, qui pointe sur la fin de notre fichier (et qui sera incrémenté à chaque fois que l'on stocke un nouveau caractère en mémoire). Comme ce pointeur doit pouvoir adresser sur seize bits, il nous faut lui réserver deux mémoires contiguës - dans notre exemple 30 (partie basse) et 31 (partie haute)-.

Essayons de traduire ces instructions en langage machine (transposé en Applesoft) en bon assembleur, en n'oubliant pas que si Applesoft affecte et réserve une place mémoire aux variables quand elles se présentent, le programmeur en langage machine doit les réserver à l'avance et, de préférence, les déclarer en début de programme.

```
FIN     EQU $1E
        LDA £$60
        LDX £$00
        STA (FIN,X)
```

(30 et 31 décimal donneront $1E et $1F)

On remarquera que l'instruction LDA, qui comporte bien des modes d'adressage (huit, voir article précédent pour leur signification), n'a pas l'adressage indirect sans un registre associé. Il a fallu utiliser l'adressage indirect préindexé par X. Nous avons pris le soin de mettre le registre-X à zéro avant de l'utiliser, sinon nous serions allés quérir l'adresse ailleurs qu'en $1E-$1F.

Si nous avions choisi l'adressage indirect post-indexé par Y [à savoir STA (FIN),Y] , il aurait fallu également s'assurer préalablement que le registre-Y contienne bien zéro, faute de quoi on serait bien passé par $1E-$1F, mais l'adresse trouvée dans notre pointeur se serait ajoutée au reliquat contenu dans le registre-Y.

On notera avec satisfaction que l'instruction va chercher toute seule, dans les deux octets contigus qui constituent le pointeur ($1E et $1E+1), la totalité de l'adresse où ranger notre constante. Mais il faut parfois gérer soit-même les deux parties de l'adresse. Par exemple quand on souhaite faire progresser le pointeur FIN, après avoir mis un nouveau caractère dans le fichier:

```
AJOUT   INC FIN
        BNE BOUCLE
        INC FIN+1
        BNE BOUCLE
```

La première instruction actualise la partie basse de l'adresse ($1E),jusqu'à la valeur £$FF et se rebranche sur la boucle de traitement. Au tour suivant, cette partie basse prend la valeur £$00, car $100 dépasse la capacité d'un octet, l'indicateur Z(éro) se positionne vrai (1), et le test BNE n'est pas vérifié. Il faut traiter soi-même le report, ce que je fais dans la suite du code en actualisant la partie haute ($1F) par INC FIN+1.

N.B. Il faut espérer que le deuxième BNE est toujours vérifié, faute de quoi je serais parvenu à écrire dans la mémoire morte! Mais rassurez-vous: auparavant, le DOS ne se serait pas laissé passer sur le corps sans réagir!

La page zéro est arrivée !

Pour que mes pointeurs fonctionnent avec l'adressage indirect, il faut qu'ils soient choisis dans la page zéro, c'est-à-dire dans les 256 premières positions de mémoire, $00 à $FF, les seules qui soient adressables en un seul octet. Mais attention, il n'y en a que 256 et, si vous ouvrez votre Manuel de Référence à aux pages 74 et 75, vous constaterez que la fameuse PAGE ZERO est plus encombrée que la place de l'Etoile à six heures du soir. L'utilisateur ne devra pas empiéter sur les systèmes auxquels il risque d'avoir recours (DOS, bien sûr, mais aussi le moniteur dont on va utiliser abondamment les sous-programmes et également le BASIC de service, si on veut parvenir à un bon mixage...).

La pile (STACK) peut apporter une solution de rechange, pour certains stockages temporaires. Mais c'est une solution à utiliser avec prudence (on n'est pas en FORTH !) et qui demande une saine gestion (de pile).

Une solution de rechange : les compilateurs

On assiste actuellement à un certain engouement pour les compilateurs. En effet le BASIC interprété se montre trop lent dans certaines applications. Je citerai à ce sujet une étude parue dans "Cider Press" et qui comparait les temps d'exécution d'un petit programme de jeu écrit en différents langages. Il s'agissait d'un jeu de Marienbad, l'Apple jouant contre lui-même. Les résultats (étonnants) furent les suivants:

BASIC Integer	8 minutes
Pascal	2 minutes
Forth	1 minute
Langage Machine	1 seconde

Certes, le BASIC compilé fait gagner du temps en exécution ,mais ce temps est payé au prix d'un temps de chargement plus long des programmes. En effet, la compilation est prolixe et le programme résultant est beaucoup plus long que le programme BASIC initial. Le temps d'exécution sera raccourci par rapport à celui obtenu en BASIC interprété, sans atteindre - tant s'en faut - la vitesse d'un programme écrit en langage assembleur. Par contre, si l'on part d'un programme existant en Applesoft, la compilation sera beaucoup plus rapide qu'une compilation manuelle; celle-ci consiste à traduire chaque instruction BASIC en une série d'instructions assembleur (pour plus de détails cf."Converting Integer BASIC to Assembly Language" - Biblio 1).

Mais la meilleure solution sera de programmer directement en langage-machine ou, si l'on part d'un programme BASIC, de reprogrammer les parties du programme qui nécessitent une exécution plus rapide. Si limitée soit-elle, l'expérimentation que mes amis et moi-même avons faite des compilateurs disponibles s'avère plutôt décevante . Il semble que les compilateurs actuels tolèrent assez mal les programmes BASIC qui n'ont pas été écrit spécialement en vue de la compilation. Cette allergie se traduit par un certain nombre de messages d'erreur qui interrompent la compilation. S'il faut ré-écrire le programme pour pouvoir le compiler, où se trouve le bénéfice? Tout donne à penser que nous sommes en présence de compilateurs de première génération qui auront des descendants plus civilisés.

Une optimisation facile

S'il s'agit avant tout d'améliorer la vitesse d'éxécution, tout en réduisant la taille du code, le programme AOPTIMISER apporte une solution à considérer. Il s'agit d'un petit logiciel de "Sensible Software" qui est extrêmement performant. En deux passages, il réduit la taille des variables à une seule lettre et condense en une ligne toute série d'instructions susceptibles logiquement de figurer sur une même ligne. Il fait donc, en mieux, ce que fait le CRUNCHER de DAKIN (voir Pom's 2).

Un espoir déçu

Dans la suite logique de son article sur la compilation manuelle, Randy HYDE -toujours lui- nous a donné, deux ans plus tard, un outil automatisé avec "SPEED/ASM" (chez "On Line"). J'avoue que l'annonce de ce logiciel m'avait mis l'eau à la bouche (N'est-ce pas le propre de toute bonne publicité !).

Ce logiciel se veut de nature à satisfaire les réfractaires aux langage machine qui veulent néanmoins profiter de l'efficacité d'un assembleur. Il s'agit d'une extension de votre assembleur lui donnant les capacités d'un macro-assembleur. SPEED/ASM permet de traduire ("à la main") un programme BASIC en programme assembleur, pratiquement instruction par instruction ou, pour être plus précis, à raison de deux instructions SPEED/ASM (au minimum) pour une instruction BASIC; en effet chaque opérande fait l'objet d'une instruction séparée.

L'idée est séduisante, la réalisation moins satisfaisante : on y introduit une phase supplémentaire, car on n'est pas dispensé de l'assembleur et, en outre, même s'il est limité à une vingtaine d'instructions, il faut apprendre un nouveau langage, une sorte d'assembleur hybride. Peu satisaisant en première instance, SPEED/ASM est susceptible d'être réhabilité en appel.

Conclusion

Tout cela nous laisse peu d'espoir d'échapper au code machine et à l'assembleur. Mais le mal n'est pas bien terrible. La connaissance du langage machine nous permet de mieux comprendre l'Apple et de mieux utiliser le BASIC (cf. "All about Applesoft", Biblio - 4).

La solution, pour être efficace, consiste à mélanger langage machine et Applesoft. Dans le prochain article, nous étudierons un utilitaire qui vous permettra d'utiliser votre système de traitement de texte pour éditer vos listes-source d'assemblage, même si votre système de traitement de texte favori n'accepte que des fichiers en binaire. Le programme annoncé est un parfait exemple de l'utilisation simultanée de BASIC et code-machine, en prenant le meilleur de chaque monde.

BIBLIOGRAPHIE

1 - Nicole Bréaud-Pouliquen et Daniel-Jean David : La pratique de l'Apple II - Vol III. Editions du P.S.I.

2 - Lance A. Leventhal : 6502 Assembly Language Programming

3 - Randall Hyde : Converting Integer BASIC to Assembly Language. Article paru dans Apple Orchard, Vol I, Number I, Mars-Avril 1980.

4 - All About Applesoft : Call-APPLE In Depth. Numéro 1

Courrier des lecteurs

Abonné depuis peu à votre revue, j'ai pu en apprécier la qualité des articles. Néanmoins, j'aimerais faire une remarque. Je souhaite que la partie "initiation" à l'utilisation de l'Apple et de ses langages n'empiète pas sur les articles de fond ou les programmes élaborés destinés aux amateurs "avertis"... Mais pour ne léser personne, je pense qu'il vous sera nécessaire d'augmenter le nombre de pages de chaque numéro, ainsi tout le monde sera satisfait. A moins qu'une parution bimestrielle ou mensuelle vous semble plus avantageuse.

En attendant, je vous envoie une disquette contenant deux programmes que j'ai écrits en collaboration avec un de mes amis, Thierry Le Tallec ...

*** Jacques Tran-Van, Marseille ***

Réponse

Nos lecteurs, étant tous des utlisateurs Apple, représentent par la force des choses un marché somme toute encore assez restreint; nous ne pouvons pas particulariser la revue en la dédiant exclusivement aux "pros"! C'est pourquoi nous nous efforçons de publier dans chaque numéro des articles de niveaux variés. Il est possible que nous passions dans l'avenir à une fréquence de publication plus grande, ou que nous augmentions le nombre de pages. Nous devons, avant de le faire, nous assurer de rendre Pom's économiquement viable. Nous avons fait l'expérience, dans le présent numéro, de réduire le texte, ce qui revient à l'augmenter de dix pages (par rapport au format normal).

L'augmentation de la fréquence ou du nombre de pages prend plus de temps que prévu initialement, car la Commission Paritaire des revues de presse a refusé de nous reconnaître en tant que publication. Nous sommes considérés comme un organe publicitaire de la société Apple et, à ce titre, privé du régime postal et du régime de TVA des périodiques. Cette blague augmente nos coûts d'au moins 20%, sans compensation de recette. Nous devons donc croître avec plus de prudence ...

Ceci dit, nous sommes heureux de pouvoir publier dans ce numéro de Pom's un de vos deux programmes, en espérant publier le reste par la suite.

J'ai acheté les numéros 2 et 3 de votre revue et la trouve très intéressante. Je me permets de vous envoyer ci-joint quelques remarques :

1) A propos de l'article de J-F Duvivier (Pom's numéro 2)
*Correction en page 19 : il ne faut pas repasser au clavier entre le moment où l'on remplit les mémoires $AA5D et AA5F, et celui où l'on lance le petit programme, car le DOS les modifie quand il rend le contrôle à l'utilisateur. Il faut donc rentrer : *300 : ..., *AA75: ..., *AA5D: 9, *AA5F: 0, *300G.*

2) A propos de l'article de J-L Meillaud (Pom's numéro 3)
M. Meillaud regrette que les gens n'ayant qu'un simple Apple II Plus ne puissent utiliser le mini-assembler. C'est pourtant possible (cf. 5). Correction en page 11, ligne 3 : lire $14 et non $28.

3) A propos de "La pratique de l'Apple II, volume III
Il y a quelques erreurs quant aux mémoires libres pour l'utilisateur: $1A, $1B et $1C sont utilisés en haute résolution; $F9 est la valeur de ROT; $D6 est utilisé par l'Applesoft. En effet, selon E. Augier (Pom's 3, page 66), faire POKE 214,255 empêche de lister; en fait, POKE 214,x avec x supérieur à 127 fait que toute commande (hors DOS) tapée à partir du clavier équivaut à un RUN. Pourquoi ?

4) Attention au FLASH !
Il m'est arrivé, alors que je tapais un programme tout en l'essayant, la mésaventure suivante : relisant une ligne de programme avec la flèche à droite, certains caractères changeaient au passage du curseur. Le RESET n'y faisait rien; il fallait, c'est le cas de le dire, se retaper la ligne. Je viens récemment de trouver le pourquoi de cette bogue : j'avais pressé RESET alors que j'étais en mode FLASH ! Voyons donc les détails ...

Le manuel de référence Apple nous apprend que COUT, la routine de sortie de caractères du moniteur, fait avant l'affichage un AND logique entre l'accumulateur, qui contient le caractère à imprimer, et la mémoire $32, qui contient $FF en mode normal (l'AND ne change rien), $3F en mode inverse (on ne garde que les 2 bits de poids faible) et $7F en mode flash (on garde les 4 bits faibles). Et le tableau des caractères ASCII affichés à l'écran nous montre que certains signes en mode flash sont représentés en inverse : c'est effectivement ce qui se passe dans le moniteur et en Integer. Mais l'Applesoft dispose d'un second masque : pour représenter tous les caractères clignotants, il fait un OR logique entre l'accumulateur et la mémoire $F3 qui vaut $0 en mode normal ou inverse, et $40 en mode flash (on force à 1 le 3ème bit). C'est ce que montrent les routines Flash, Inverse et Normal ($F273 ...), et la routine OUTDO ($DB5C à $DB66).

Or, le RESET exécute un programme du moniteur qui remet la mémoire $32 à $FF, mais qui ne touche pas à la mémoire $F3, chasse gardée de l'Applesoft. Ainsi, appuyer sur RESET quand on est en mode flash entraîne un mode pseudo-normal qui fait que les caractères ASCII affichés (dans un LIST ou un PRINT) de codes 32 à 63 sont transformés dans les caractères de 96 à 127 : ils paraissent normaux, mais repasser dessus avec le curseur les change.

Pour sortir de cette situation, on peut taper POKE 243,0 - ou taper NORMAL - ou encore FP (NEW et RESET ne marchent pas). Et comme il faut bien tirer parti de ses ennuis, cela nous amène à une astuce qui permet d'imprimer des minuscules sur l'imprimante, tout en tapant des majuscules dans le listing, ceci de la façon la plus simple du monde : il suffit avant le PRINT de faire POKE 232,32. Et l'on imagine bien que l'ORA$F3 et le AND$32 permettent de varier à sa guise de telles combinaisons.

5) Le mini-assembler à portée de tous
Dans la disquette SYSTEM MASTER du DOS 3.3, le fichier INTBASIC permet de charger sur la carte langage le BASIC Integer, le MINI-ASSEMBLER, SWEET16 et le Programmer's Aid No.1. Si l'on n'a pas la carte langage, on peut quand même récupérer le mini-assembler en chargeant INTBASIC plus bas dans la mémoire et en modifiant dans le programme les adresses de branchement absolu. Les opérations à effectuer sont les suivantes :

```
BLOAD INTBASIC,A$6000          85DD:86
CALL -151                      85E7:86
8537:8                         8633:85
855B:85                        8668:85
85BF:86      BSAVE MINI-ASSEMBLER,A$8500,L$170
```

Le mini-assembler proprement dit est compris entre $8500 et $863C. Mais le fichier va jusqu'en $865F, car il y a à l'adresse mnémotechnique $8666 un JMP $8592 vers l'adresse d'entrée de l'assembleur. Pour utiliser l'assembleur, il suffit donc de le BLOADer et de faire CALL-151, suivi de 8666G. Si l'on veut faire cela à partir d'un programme Applesoft, il faut faire HIMEM:34048. On peut sans doute appliquer le même traitement à l'Integer, mais cela devient titanesque. Pour ceux qui n'ont pas le DOS 3.3 ni l'Integer, il suffit de recopier le listing désassemblé ou non du mini-assembleur, qui ne fait que 317 octets, plus le JMP $8592 en $8666.

*** Olivier Herz - 78 Chatou ***
N.D.L.R.

Merci pour tout ce travail. Malheureusement, le manque de place ne nous permet pas de reproduire ici le code du mini-assembleur. Ce ne devrait pas être dur à trouver, puisque les DOS 3.3 sont largement diffusés !

Comme nous le fait remarquer un autre lecteur, Monsieur Philippe Jaffré, le numéro 1 de Nibble 1982 publie une lettre dans laquelle on montre comment le mini-assembleur peut être ensuite déplacé en mémoire. Ceci dit, vous pouvez aussi le faire en utilisant le programme de déplacement en assembleur du numéro 1 de Pom's.

Suite à la lecture des trois premiers numéros de Pom's, voici quelques remarques qui, je pense, pourront aider d'autres lecteurs.

1) Le livre What's Where in Apple
Je viens de trouver dans MICRO de janvier 82 la signification des signes utilisés. La syntaxe en est la suivante : /XY/ avec :

Première lettre	*Seconde lettre*
S : subroutine	*E : entry*
P : parameter	*B : block*
H : hardware	*n : n bytes long*

B : buffer
L : label
F : flag

2) Personnalisation du DOS
*Afin d'avoir un maximum de place sur chaque disquette, il est intéressant quand on possède un Apple II muni de la carte langage de n'avoir que sur une disquette de boot le langage complémentaire (INTBASIC pour Apple II Plus, FPBASIC pour l'ancien). Or, à chaque fois que l'on reboote une disquette après la mise sous tension, le DOS force le chargement du langage dans la carte en écrivant 00 à la première adresse de celle-ci. Pour éviter cette opération, il faut passer en mode moniteur et taper *BFD3 : EA EA EA (ces NOP vont remplacer le code 8D 00 E0), puis initialiser la nouvelle disquette.*

3) Personnalisez vos disquettes, de P. Boutreux
Tout d'abord, une petite rectification : le DOS réserve 12 caractères pour l'information "DISK VOLUME ", et non pas 11. Remarque : si, après avoir booté une disquette ainsi personnalisée, on change de disquette et que l'on tape CATALOG, on obtient le nouveau catalogue avec l'ancien en-tête, en effet, le DOS ne relit pas cette information. Est-il possible de l'y amener ? Dans le même ordre d'idées, le DOS ne relit pas le numéro de volume quand il liste un catalogue; que faut-il faire ?

4) Comment court-circuiter les numéros de volume
*Si vous voulez empêcher les ordres DOS provenant d'un programme d'utiliser le contrôle par numéro de volume, il suffit de vous mettre en moniteur et de taper : *BE1E:EA EA D0 (qui remplace la séquence C5 2F, F0 04, c'est-à-dire CMP $2F, BEQ $BE26).*

5) Petite annonce
Je vends un moniteur portable couleur Thomson (08/81) à 2500 FF, l'interface RVB pour Apple (revision 0 ou 1, 500 FF), une imprimante Seikosha GP80M (09/81) avec logiciel hard copy d'écran, 500 feuilles et 2 rubans (2200 FF le lot).

*** Alain Sorin - 80 rue Rouget de Lisle - 92000 Nanterre - Tel (1) 721 04 10.***

N.D.L.R.

Merci pour tous ces éléments, et bonne chance pour la petite annonce, à laquelle nous produisons une compagne ci-dessous.

Je vends un moniteur N/B (10/80) 700 FF, et une imprimante à aiguille OKI 40 colonnes (friction) avec interface Apple (les deux pour 1000 F TTC).

*** Jean-François Duvivier - 1, rue du Sergent Blandan - 92130 Issy les Moulineaux - Tel 558 0578 (le soir) ***

Les clubs ont la parole

Groupe Apple de Genève (Suisse)

Le groupe Apple de Genève (GAG) est un club regroupant environ une centaine d'utilisateurs enthousiastes d'Apple de la région Genevoise et de la France voisine. Ce club a de nombreuses activités, dont la publication d'un journal, GAG'S LETTER, et huit groupes d'intérêt sur les sujets suivants : langage assembleur 6502, Pascal, matériel, jeux et graphiques, introduction au BASIC, télécommunications, applications commerciales et enfin enseignement. Ce club recherche l'échange d'idées, d'informations et d'expérience.

Contacter Alan J. Ehrlich, Groupe Apple de Genève, Ch. Ami-Argand 9, Case Postale 20, CH-1290 Versoix, SUISSE.

Association Hespérides (69)

L'association Hespérides a été créée récemment. Elle est constituée essentiellement d'experts-comptables équipés de matériels de marque Apple et a pour objet de favoriser entre ses membres la connaissance, la communication et l'échange d'informations et d'expériences sur matériel micro-informatique.

Le siège social est situé au 63 rue de Créqui, 69006 Lyon.
Président : M. Jacques Calop (79) 07.03.65
Trésorier : M. Marcel Gorlier (79) 33.25.49
Secrétaire : M. Victor Bérard (7) 893.42.63
... tous trois experts-comptables.

Club Microtel CESA (78)

Le club Microtel CESA a récemment ouvert ses portes à Jouy-en-Josas. Il dispose actuellement de matériels de marque Apple, Goupil et IBM. Un intérêt tout particulier est accordé à l'utilisation de l'informatique individuelle en gestion.

Contacter Christophe Peslerbes, HEC chambre G 33, 1 rue de la Libération, 78350 Jouy-en-Josas. Tel (3) 956 8000.

NOUVEAU

DES LIVRES POUR VOTRE APPLE II

MATÉRIELS

La découverte de l'Applesoft

Tome 1
par Dominique Schraen et Frédéric Lévy

Cet ouvrage d'initiation s'adresse aussi bien aux futurs utilisateurs de l'Apple voulant apprendre la programmation en Basic Applesoft, qu'à l'Applelophile chevronné sollicité par ses proches curieux de "voir un peu comment ça marche". D'approche progressive, il est illustré de nombreux exemples et exercices.

Série verte - 128 pages - 65,00 FF

La découverte de l'Applesoft

Tome 2
par Frédéric Lévy

Recueil d'exercices, destiné à tous ceux qui connaissent les instructions Basic de l'Applesoft et ne maîtrisent pas encore la programmation, c'est une invitation à l'analyse et à la programmation de problèmes simples et fréquemment rencontrés. L'énoncé de chaque exercice est suivi de son analyse; une ou deux solutions commentées sont proposées.

Série verte - 120 pages - 65,00 FF

La pratique de l'Apple II

Volume 1
par Nicole Bréaud-Pouliquen

Cet ouvrage présente les spécificités du Basic Applesoft à partir d'une description du matériel et du logiciel du système Apple. Les techniques de programmation, de composition et d'animation de dessins et graphiques colorés y sont expliqués à l'aide d'exemples illustratifs et d'exercices résolus.

Série bleue - 128 pages - 65,00 FF

La pratique de l'Apple II

Volume 2
par Nicole Bréaud-Pouliquen

Ce second volume de la pratique de l'Apple II est consacré au système d'exploitation disque, à la gestion des fichiers, à l'impression et aux imprimantes, à la carte horloge Appleclock. De nombreux exemples de programmes illustrent les fonctions et les commandes décrites.

Série rouge - 120 pages - 65,00 FF

La pratique de l'Apple II

Volume 3
par Nicole Bréaud-Pouliquen et Daniel-Jean David

Ce volume est une initiation à la programmation en langage machine 6502, dont le jeu d'instruction est expliqué et utilisé. L'assembleur symbolique et ses logiciels connexes y sont décrits. L'interaction avec le Basic et avec le système y sont étudiés.

Série noire - 176 pages - 75,00 FF

LANGAGES

Programmer en Basic

par Michel Plouin

Ce livre a été écrit pour les utilisateurs d'ordinateurs individuels en particulier d'Apple II, TRS-80 et PET/CBM. Un répertoire Basic rend son utilisation très pratique et facilité la transposition d'un programme écrit pour un P.S.I. sur un autre.

Série verte - 132 pages - 65,00 FF

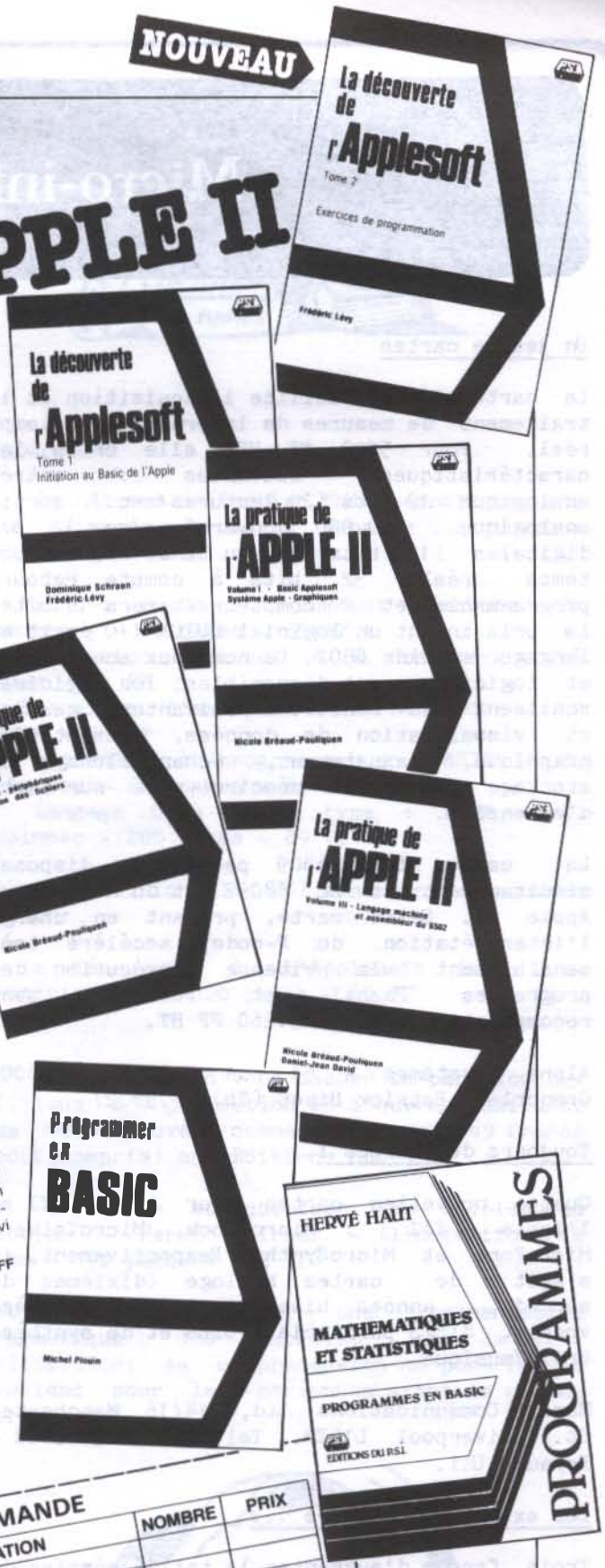
PROGRAMMES

Mathématiques et statistiques

Par Hervé Haut

Cet ouvrage est un recueil de 16 logiciels de base (niveau supérieur) tant en mathématiques qu'en stastistiques. Chaque problème traité comporte une introduction numérique, un exposé de la technique de programmation utilisée, un organigramme détaillé et un programme complet en Basic suivi d'un exemple d'utilisation.

Série rouge - 272 pages - 85,00 FF



P.S.I.

Editions du P.S.I.
41-51, rue Jacquard
BP 86 - 77400 Lagny-s/Marne
Téléphone (6) 007.59.31

BON DE COMMANDE

DESIGNATION	NOMBRE	PRIX
	TOTAL	

Envoyer ce bon accompagné de votre règlement à
EDITIONS DU P.S.I.
41-51, rue Jacquard
BP 86 77400 Lagny-s/Marne
Tél. (6) 007.59.31.

Les prix sont : taxes, emballage et port compris.
(par avion : ajouter 5 FF par livre)

NOM _____ PRENOM _____ N° _____

rue _____ Ville _____

Code post. |_|_|_|_|_|

Date et signature :

AP.A

Micro-informations

Un jeu de cartes

La carte ADALAB facilite l'acquisition et le traitement de mesures de laboratoire en temps réel. Pour 5200 FF HT, elle offre les caractéristiques suivantes : entrée analogique 12 bits (20 lectures/sec.), sortie analogique (50.000 conversions/sec.), E/S digitales (16 bits E/S ou 8E et 8S, horloge temps réel 32 bits à compte rebours programmable et 2 compteurs/timers 16 bits. Le prix inclut un logiciel QUICK I/O écrit en langage machine 6502. De nombreux accessoires et logiciels sont disponibles; les logiciels réalisent les fonctions suivantes : gestion et visualisation de données, présentation graphique, ajustement, échantillonage et stockage rapide en mémoire vive sur carte d'extension.

La carte ALPHA 6809 permet de disposer simultanément du MC 6809E et du MC6502 sur Apple II. Cette carte, prenant en charge l'interprétation du P-code, accélère très sensiblement la vitesse d'exécution des programmes Pascal et FORTRAN, sans recompilation. Prix : 3.260 FF HT.

Alpha systèmes - 51 rue Thiers - 38000 Grenoble - Patrick Binet (76) 87.98.27

Toujours des cartes !

Quatre nouvelles cartes pour l'Apple II et l'Apple III : MicroClock, MicroTalker, MicroPort et MicroSynth. Respectivement, il s'agit de cartes horloge (dixièmes de seconde, années bisextiles), de synthèse vocale, d'E/S parallèle 8 bits et de synthèse bruit/musique.

March Communications Ltd, 14/16 Manchester St., Liverpool L16ER. Tel (51) 236 2000 - Royaume Uni.

Des extensions mémoire ...

Trois façons d'augmenter la taille mémoire de votre Apple :

1. La carte RAM 64KC de Legend rajoute 64K à vos 48K, en occupant un seul connecteur et en consommant nettement moins. Elle est compatible avec Pascal (en remplaçant la carte langage si nécessaire) et CP/M. Le DOS peut s'y charger, libérant ainsi la mémoire centrale. Prix public : 3.300 FF TTC.

2. L'émulateur de disquette 2*64KDE de Legend met à votre disposition un "lecteur de disquette" à accès très rapide, et bénéficie des mêmes comptabilités que la carte précédente. Des utilitaires spéciaux sont fournis avce la carte, dont l'un effectue la copie complète ou la sauvegarde d'une disquette en 16 secondes. Prix public : 6.500 FF TTC.

Ces deux cartes sont livrées avec toute une série de programmes utilitaires et de démonstration. Vous pouvez étendre votre Visicalc à plus de 80K avec la carte 128KDE, et à plus de 100K avec deux cartes 64K. Les documentations en français doivent être terminées à la date de parution de Pom's. Contacter pour tout renseignement BIP, 3 rue de Duras, 75008 Paris. Tel (1) 264 0232.

Un bras robotisé pour l'Apple !

Pour 200 livres, vous vous offrez un troisième bras, en kit, commandé par moteurs pas à pas, et comportant cinq axes ou articulations. Il pèse huit livres, mesure 17 pouces et possède trois doigts d'origine. Il lui faut du 12 volts à 3 ampères, et celui lui donne la force de soulever 10 onces (environ 280 g.); la précision de positionnement est de 0,1 pouce soit 0,285 ongle. D'autres extensions que les trois doigts peuvent être fixées à loisir; certaines sont déjà réalisées et opérationnelles.

Pour la conversion des unités anglaises en unités plus courantes, ainsi que pour tout renseignement, contacter Colne Robotics Ltd, 1 Station Road, Twickenham, Middlesex. Tel (1) 892 7044. Royaume Uni.

Mettez un tigre dans votre moteur !

1. La carte 88

Selon sa publicité, la carte qu'IBM craint de vous voir acheter transforme votre Apple en ordinateur 16 bits compatible avec le MS-DOS du micro IBM. En outre, elle étend votre mémoire de 64K. Il vous suffit de débloquer 899 dollars et de contacter Coprocessors, 50 West Borkaw Road, Suite 64, San Jose, CA 95110 (USA).

2. La MetaCard

La MetaCard, outre ces possibilités, permet le fonctionnement simultané du 6502 (Apple) et du 8088 (IBM), mais exige par contre une alimentation séparée (fournie avec la carte). Metamorphic Systems, dont nous n'avons pas l'adresse, vend sa carte 980 dollars, avec l'extension 64K.

3. Informations complémentaires

Les deux cartes peuvent être étendues à 128K. La MetaCard possède en standard le CP/M-86 de Digital Research, le MS-DOS et le Pascal UCSD étant en option. La carte 88 offre l'un ou l'autre des deux premiers SED en standard, l'autre et le Pascal pouvant être acquis en option.

4. Le microprocesseur Motorola 6809

La société IEF commercialise depuis peu une carte comportant 64K de MEV et un microprocesseur 6809 fonctionnant en 8/16 bits. Les 64K peuvent être utilisés en simulation de disquette à accès rapide. L'ensemble est vendu 4900 FF HT, programmes d'application compris.

IEF- 228/230 rue Lecourbe - 75015 Paris - Tel (1) 828 0601.

LOGICIEL

Un générateur de programmes

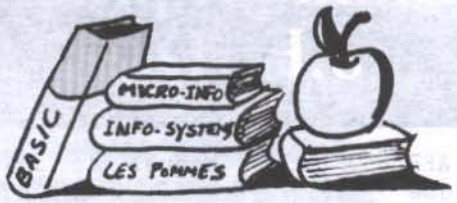
Encore un générateur de programmes pour l'Apple ! Avec C.O.R.P., on crée entre autres ses masques de saisie et de sortie sans savoir programmer. Les programmes sont générés en Basic Applesoft et peuvent ensuite être modifiés facilement. Il faut avoir un Apple 48K et deux lecteurs, ainsi que 235 dollars US pour le Basic System. La version 2, qui comporte une disquette d'utilitaires, revient à 425 dollars.

Dynatech Microsoftware Division, 3 New England Executive Park, Burlington, MA 01083 U.S.A.

Jonglez avec les disquettes ...

Avec l'utilitaire LinkDisc, vous pouvez, à partir du Pascal, réaliser les opérations suivantes : obtenir un catalogue de disquette en DOS 3.3, convertir en Pascal des fichiers binaires ou texte DOS 3.3, comparer bit à bit deux fichiers ou volumes Pascal, examiner une disquette au niveau de l'octet ou même du bit et lister un fichier texte Pascal avec numérotation automatique de lignes et de pages, dates et titres.

LinkDisc est réalisé par Link Systems, 1655 26th Street, Santa Monica (USA) et venu 70 dollars.



Publications

1. La récente collection "Micro-ordinateurs", aux Editions Eyrolles, propose les nouveaux titres suivants :

. Micro-ordinateurs : comment ça marche - R. Schomberg - 96 pages - 55 FF
. Le Basic universel - R. Schomberg - 128 pages - 55 FF
. Pascal par l'exemple - J.A. Hernandez - 156 pages - 55 FF
. Apprendre à programmes en BASIC - C. Delannoy - 272 pages - 80 FF
. Langage d'un autre type : LISP - C. Queinnec - 200 pages - 89 FF

2. Les éditions du PSI annoncent un nouveau titre :

. La découverte de l'Applesoft (tome 2) - Frédéric Lévy - 120 pages - 65 FF

3. Peut-on enfin vous cacher la parution aux Editions d'Organisation de l'ouvrage suivant, que vous pouvez commander pour 149 francs (port compris) aux Editions MEV ?

. Comprendre et utiliser les modèles en gestion - Hervé Thiriez - Illustrations de Piem - 192 pages - 149 FF

Cet ouvrage offre une présentation pragmatique des modèles et de leur utilisation; sa compréhension ne pose pas de problème pour les personnes ayant le niveau Terminale C.

Bibliographie J.F. D.

BASIC APPLESOFT - Manuel de référence alphabétique
Paul MERRY - Editions MNEMODYNE
Prix indicatif: 120 F chez SIDEG

Voici un livre qui pourra faire bonne figure dans votre bibliothèque, ou plutôt à côté de votre Apple, car il risque de vous servir souvent ! D'autant plus souvent qu'il est écrit en français. Nul besoin donc d'être familiarisé avec la langue de Shakespeare pour l'utiliser...

A vrai dire, il s'agit plus d'un répertoire que d'un livre. Le sous-titre le définit d'ailleurs fort bien : "Définition, utilisation et exemples pour les instructions et concepts de programmation, éditions de textes et traitement de fichiers (DOS)".

Les différentes instructions de l'Applesoft et du DOS y sont classées par ordre alphabétique et une ou plusieurs pages est consacrée à chacune. On y retrouve, avec une présentation très claire, leur syntaxe sous forme de diagramme, leur utilisation avec conseils, modes d'emploi, restrictions...et un petit programme utilisant cette instruction.

Outre les instructions du Basic et du DOS, sont rassemblés d'autres mots-clés (erreurs, reset, moniteur, instructions, DOS, curseur...). Par exemple, si vous avez une erreur dans votre programme, nul besoin de fouiller dans le manuel de l'Applesoft ou celui du DOS. Cherchez donc à "ERREURS". Tout est là : le code, les causes possibles, les remèdes...

Par contre, des erreurs non référencées émaillent le texte : j'ai pu en relever jusqu'à six dans une seule page! Il est regrettable que l'on puisse publier un livre sans apparemment l'avoir relu une seule fois. Avis aux éditeurs.

Tout d'abord sceptique quant à l'utilisation que je pourrais faire de cet ouvrage, je me suis vite aperçu de son utilité et de la facilité d'emploi pour retrouver une information (facilité dûe essentiellement à une présentation exellente).

C'est dire que je le conseille non seulement aux débutants en Applesoft, pour qui il est un instrument remarquable, mais également aux utilisateurs plus avançés, qui peuvent avoir de temps en temps des doutes ou des lacunes. Courrez vite chez votre distributeur Apple pour le découvrir !

Imprimerie Laballery, 58500 Clamecy, imprimé en France. Directeur de la publication Hervé Thiriez
Dépôt légal : 2e trimestre 1982

ILLEL
ESPACE ELECTRONIQUE

ILLEL CENTER PARIS 15e : 143, av. Félix-Faure - 75015 Paris, Tél. 554.97.48, Métro : Balard.
ILLEL CENTER PARIS 10e : 86 bd Magenta 75010 Paris, Tél. 201.94.68, Métro : Gare de l'Est, Parking : Magenta.

apple®

LES MICRO-ORDINATEURS

APPLE II

Un des micro-ordinateurs les plus fiables de sa génération, Apple II est utilisé dans de nombreux domaines : gestion, comptabilité, enseignement, utilisations scientifiques et industrielles, applications domestiques.

D'une très grande robustesse (garantie totale 1 an) Apple II n'excède pas 5 kg et sa facilité de transport renforce encore sa souplesse d'utilisation.

Son extensibilité est remarquable : Apple II étant compatible avec la plupart des périphériques actuels, il bénéficie d'un large éventail de possibilités.

PROMOTION
1 APPLE II 48 K
+ 1 FLOPPY
+ 1 MONITEUR
+ 10 DISQUETTES
12 900 F TTC

CONFIGURATION DE DÉVELOPPEMENT

Matériel	Langage BASIC	PILOT	PASCAL	FORTRAN
Système	II Plus	II Plus	II Plus	II Plus
Mémoire utilisateur (RAM)	32K	48 K	48 K	48 K
Micro-programmation	Cartes BASIC	Cartes BASIC	Carte Langage	Carte Langage
Unités Disk II	1	1 ou 2	1	1

APPLE III

L'Apple III est un système d'ordinateur de bureau puissant, faisant partie d'ensembles étudiés sur mesure et conçus pour résoudre vos besoins complexes en application. Pour les managers, les financiers, les analystes et tous ceux qui ont besoin d'organiser des faits et des chiffres, il existe le système d'Analyse de l'information Apple III.

Option A : **33.330 F TTC** visicale 3 - S.O.S. buissness Basic - Moniteur3 12"

Option B : **38.100 F TTC.**
Idem A + Floppy supplémentaire

Option C :
41.100 F TTC
Idem B + Imprimante thermique graphique.

LES LOGICIELS

Pour APPLE II

PHANTOMS FIVE 48 K (DOS 3.3)	260,00 F TTC
SPACE EGGS 48 K (DOS 3.2 ou 3.3)	260,00 F TTC
RASTER BLASTER 48 K	295,00 F TTC
APPLE PANIC 48 K	380,00 F TTC
COMPUTER BISMARCK 48 K	395,00 F TTC
COMPUTER NAPOLEONICS 48 K	450,00 F TTC
COMPUTER AIR COMBAT 48 K	495,00 F TTC
VISICALC (DOS 3.3) 16 secteurs	1 764,00 F TTC
VISITREND + VISIPLOT	2363,76 F TTC
VISIDEX	1 764,00 F TTC
VISIPLOT	1 640,52 F TTC
VISITERM	1 375,92 F TTC
DESK TOP PLAN II	1 764,00 F TTC
CCA/DMS (Gestion de Fichier)	1 105,44 F TTC
APPLE WRITER	576,24 F TTC
APPLE POST	352,80 F TTC
APPLE PILOT	1 293,60 F TTC
MINI-ASSEMBLEUR APPLE SOFT	235,20 F TTC
PROGRAMME COMPTABILITÉ GÉNÉRALE (SAARI)	3 410,40 F TTC
PROGRAMME PAYE (GIPSI)	2587,20 F TTC

diagonale

BON DE COMMANDE EXPRESS ILLEL

A retourner à : ILLEL Center Informatique :
service vente par correspondance
143, avenue Félix Faure 75015 Paris.

MODE DE RÈGLEMENT CHOISI

☐ à la commande paiement comptant
☐ à crédit* à partir de 2000 F.

Je verse 20 % du montant total de mon achat : ____ , ____ F
ci-joint : Chèque bancaire ☐ C.C.P. ☐ Mandat carte ☐

* Conditions de crédit CREG : • Être salarié, • 20 % minimum au comptant, solde arrondi à la centaine supérieure.

Je soussigné : Nom _____ Prénom _____

N° _____ Rue _____

Codepostal _____ Ville _____ Tél. _____

commande ferme et désire recevoir d'urgence	Quantité	Prix unitaire	Prix total

Signature :

Montant net

Frais de port pour envoi postal 30,00

TOTAL A PAYER

Pour mieux choisir "votre" ordinateur et pour mieux l'utiliser.

Lisez

L'ORDINATEUR INDIVIDUEL

Vous y trouverez :

L'actualité et les tendances de l'informatique individuelle • des galops et des bancs d'essai des principaux matériels • des panoramas et des tests comparatifs • le point des grandes manifestations internationales • des articles d'initiation • des synthèses • des programmes • des interviews "exemplaires" • des conseils • des idées • des astuces.

L'ORDINATEUR INDIVIDUEL

chez votre marchand de journaux

41 rue de la Grange aux Belles - 75010 Paris