POM'S

NUMERO 7
MARS 1983
35 F

ISSN: 0294-6068

Formation continue à la micro-informatique

PHOTO GUNHILD BULT

Tous nos informaticiens viennent de l'informatique traditionnelle, et en maîtrisent totalement les langages classiques : Assembleur, COBOL, FORTRAN... Ils utilisent leur professionnalisme et les méthodes de l'informatique pour réaliser des **applications professionnelles en micro-informatique.** Nous vendons des micro-ordinateurs sans programme. Nous vendons aussi des micro-ordinateurs avec les programmes. Il s'agit de programmes réalisés par la société KA, dont nous **garantissons la qualité** et le bon fonctionnement.

Nos formateurs enseignent l'informatique. L'enseignement de la micro-informatique nécessite des **formateurs professionnels,** suffisamment de **matériel** pour que **chacun puisse pratiquer,** un **support de cours** couvrant non seulement l'enseignement diffusé, mais permettant au participant de **s'auto-former** après le stage. Nous avons déjà accueilli de nombreux stagiaires, d'horizons et de centres d'intérêts divers : chefs d'entreprise, universitaires, professions libérales, informaticiens, musiciens compositeurs, retraités, cadres de grandes entreprises, revendeurs de micro-ordinateurs...

Nous proposons 5 possibilités :

■ Stage de 2 jours bases de données.
Comment utiliser les progiciels :
- bases de données
- manipulateurs de nombres et générateurs de tableaux
- générateurs d'états imprimés

Application pratique (un 48 K + un lecteur de disquettes pour deux participants). Après ce stage, on peut générer, à partir de progiciels, un programme totalement adapté à son application en moins d'une journée de travail.
Ce stage nécessite de connaître la manipulation de l'APPLE II, ou d'avoir suivi au minimum la journée d'initiation.

Dates 27-28 juin
29-30 août
Prix 2192 F h.t.

■ Stage de 1 semaine de programmation BASIC.
Il débute par la journée d'initiation.
Le stage permet d'assimiler la logique de programmation et de l'appliquer (un micro-système 48 K pour 2 participants).
En fin de stage, on sait établir un programme de gestion de fichier avec consultation en temps réel.
Ce stage ne nécessite pas de connaissance de départ en informatique.

Dates
du 18 au 22 avril
du 16 au 20 mai
du 20 au 24 juin
Prix 4245 F h.t.

■ Stage 3 jours disquettes.
Consacré à l'organisation, à la programmation et à l'exploitation de **fichiers sur disquettes magnétiques,** à travers l'étude du Disk Operating System APPLE II.
Travaux pratiques sur micro-systèmes (un 48 K + un lecteur de disquettes pour deux participants).
Ce stage nécessite :
- soit d'avoir suivi le stage de 1 semaine de programmation au préalable ;
- soit d'avoir une bonne connaissance théorique et une sérieuse pratique de BASIC de l'APPLE II.

Dates du 25 au 27 juillet
du 17 au 19 octobre
Prix 3378 F h.t.

■ Journées de sensibilisation et stages de formation à Paris et en Province.
Ils sont organisés à la demande
- d'une instance régionale telle, par exemple, une Chambre de Commerce ;
- d'un organisme de formation dans le cadre d'un cycle plus vaste de formation ;
- d'une entreprise.

La société KA installe le matériel pour la durée de la formation, assure la formation et fournit les supports de cours.

■ Journée d'initiation - Dates : 18 avril, 16 mai, 20 juin - Prix 773 F h.t.

Le nombre de places pour chaque stage est strictement limité, à la fois pour la qualité de l'enseignement et par les contraintes du matériel.
Deux animateurs sont présents pour aider les participants à la réalisation de leurs programmes.
Un support de cours très complet est remis à chaque participant.
Pour la journée d'initiation et pour les stages, les déjeuners sont pris en commun et compris.

l'informatique douce*

Renseignements et inscriptions à KA - Programme détaillé sur demande.
212 rue Lecourbe 75015 Paris - Tél. 533.13.50.
Le calendrier 83 est disponible.

*"L'informatique douce" est une marque déposée de la société KA

pom's n°7

Sommaire

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

** Niveau : D(ébutant) - M(oyen) - P(rofessionnel) - T(ous).
Il s'agit-là du niveau nécessaire pour comprendre le fonctionnement du programme décrit dans l'article, ou pour digérer l'article lui-même. Ceci dit, un programme peut être utilisé sans compréhension de son fonctionnement interne (sinon, qui jouerait aux échecs sur ordinateur ?)

Les annonceurs

Éditions MEV - 49, rue Lamartine - 78000 Versailles

Directeur de la publication : Hervé Thiriez. Imprimerie Sim, 75011 Paris. Imprimé en France. Dépôt légal : 1er trimestre 1983.

dis monsieur, apprends-moi à dessiner un écran.

apple II apple III

Carte MEM/DOS 6502

LE SYSTEME D'EXPLOITATION DU 6502 - MONOPOSTE/MULTIPOSTE

UNE EXTRÊME SIMPLICITÉ DE PROGRAMMATION.

- La division de la longueur des programmes par 20.
- La possibilité réelle de dessiner ses masques de saisie ou d'impression.
- Une indépendance totale de la périphérie choisie par rapport au système.
- L'intégralité du système contenu sur une carte mémoire de 20 K.
- Une gestion de mémoire de 140 K à 120 mégas.
- Des utilitaires déterminants
  - un générateur de programmes de gestion de fichiers permettant même le séquentiel indexé multiclé
  - un générateur d'écrans.
- CALL FN, une nouvelle commande basic, très puissante, intégrée au système permettant l'appel des sous-programmes par noms avec passage de paramètres et variables locales.
- Une version multiposte assurant la mise en commun totale des ressources sans conflit et l'autonomie des postes intelligents disposant de leur propre unité centrale.
- Des programmes compatibles APPLE II et APPLE III automatiquement transférables sur COMMODORE 8096.
- Et pour demain, des logiciels développés aujourd'hui directement compatibles avec le réseau local memnet.

S.E.P. Marketing

micro informatique service

3, rue Meyerbeer - 06000 NICE - Télex 461 916 F

DISTRIBUTEURS AGREES

D.S.A. INFORMATIQUE
5, bd Dubouchage
06000 NICE
Tél. (93) 85.15.96

MICRO ALPHA SOFT
11, impasse du Lacquet
25200 MONTBELIARD
Tél. (81) 97.16.46

S E E M I
61, rue Ch. Rivière - B.P. 0701
44401 REZE CEDEX
Tél. (40) 75.52.80

MICROMEGAS
22, rue des 3 Pierres
69007 LYON
Tél. (7) 861.19.52

G-B
C.I.C.C.
Grove house
the bordage
St Peter Port
GUERNSEY
(0481) 20155

BENELUX
MEGAVOLT S.A.
Rue de Bleurmont
32 B
B 4920 EMBOURG

Éditorial

Après des mois et des mois d'attente, nous avons enfin vu le 19 janvier les nouveaux produits d'Apple, le IIe et le Lisa. Bien entendu, nous vous en parlons dans ce septième numéro de Pom's. Par opposition au dernier numéro, dans lequel nous vous avons offert plusieurs longs articles, nous vous proposons ici de nombreux articles de tous genres. Les amateurs de Pascal peuvent se réjouir, ils trouveront trois programmes écrits dans leur langue !

Trois contributions assez importantes occuperont vos soirées pendant quelque temps : tout d'abord, un générateur de programme dû à **Denis Sureau,** *et accompagné d'une application sous forme de test programmable. Ensuite,* **Olivier Herz** *nous assène un programme Pascal de cryptographie à clef publique — grâce auquel vous pourrez coder et décoder des messages. Enfin,* **Gérard Michel** *concurrence le MEM/DOS en vous donnant un programme de gestion de masques dont l'analyse devrait permettre de grands progrès aux débutants ayant déjà lu ses articles précédents...*

De nombreux autres articles vous apporteront de multiples idées et de non moins nombreux programmes. **Guy Mathieu** *nous montre comment exploiter les possibilités graphiques de l'Epson, ce dont* **Olivier Herz** *s'inspire pour vous aider à effectuer des opérations logiques sur les pages graphiques. Nous apprennons à utiliser les minuscules dans Visicalc sur Apple II, puis nous analysons la conversion bidirectionnelle entre Apple Writer et des fichiers TEXT.*

Michel Marquis *nous propose un programme BASIC de boot pour lier PLE et CRAE (voir le numéro 1 de Pom's) à l'allumage de l'appareil et un programme de dump en Pascal.* **Philippe François** *apprend aux programmeurs en assembleur comment réaliser des programmes relogeables.* **Régis Lardennois** *nous enseigne une sécurité pour l'utilisation du directory Pascal.* **Bruno Lemaire** *et* **Gilles Mauffrey** *ont analysé le moniteur de l'Apple III et nous donnent quelques bonnes adresses. Quand à* **Olivier Herz,** *il propose encore un programme de Hard Copy pour Seikosha et un jeu (les quatre ponts) : avec un autre auteur comme lui, Pom's pourra devenir mensuel !*

Suite aux nombreuses questions des lecteurs relatives aux programmes FID et MUFFIN, **Alexandre Duback** *apporte quelques explications sur leur mode d'emploi. Vous trouverez aussi dans ce numéro un banc d'essai de l'Apple IIe (***Jean-François Duvivier** *est de retour parmi nous), une présentation du Lisa, une analyse du programme Multiplan et du programme d'apprentissage du BASIC d'André Finot.*

Nous rappelons aux nouveaux lecteurs qu'une disquette facultative en DOS 3.3 accompagne chaque numéro de Pom's. Cette disquette reprend tous les programmes de la revue, agrémentés d'explications, d'un générique (souvent) et de programmes de démonstration (parfois). Les disquettes peuvent être commandées séparément.

Le recueil numéro 1 de Pom's regroupant les quatre premiers numéros est en train de « partir » à toute vitesse. Attention, nous n'en avons tiré que 5 000 ! On peut commander les trois disquettes du recueil séparément pour 150 francs (voir le bulletin).

Nous vous remercions de continuer à nous faire confiance et encourageons tous les lecteurs qui nous envoient leurs contributions.

Nous vous donnons rendez-vous du 14 au 18 juin à notre stand (T3) au salon Micro-Expo.

Hervé Thiriez

Ont collaboré à ce numéro : Jean-Jacques Crépy - Alexandre Duback - Jean-François Duvivier - Philippe François - Olivier Herz - Yvan Koenig - Régis Lardennois - Bruno Lemaire - Thierry Le Tallec - Michel Marquis - Guy Mathieu - Gilles Mauffrey - Gérard Michel - Denis Sureau - Hervé Thiriez - Jacques Tran-Van.
Rédacteur : Olivier Herz — *Dessins :* Laurent Bidot.

Directeur de la publication - rédacteur en chef : Hervé Thiriez - *Siège social et abonnements :* Éditions MEV - 49, rue Lamartine - 78000 Versailles - *Rédaction :* 59, bd de Glatigny - 78000 Versailles - Tél. : (3) 918.13.07 - *Courrier des lecteurs (logiciel) :* Olivier Herz - 17, rue du Gros Chêne - 44300 Nantes.
Régie publicitaire : Force 7 - Anne Jourdan - 39, rue de la Grange-aux-Belles - 75483 Paris Cedex 10 - Tél. : (1) 238.66.10.
Diffusion auprès des boutiques et librairies : Éditions du PSI - 41-51, rue Jacquard - B.P. 86 - 77400 Lagny-sur-Marne.
Composition - Impression : SIM - 52, rue Servan - 75011 Paris - Tél. : (1) 357.51.20.

CTRL - ® - RESET
(ou les arcanes du Monitor Apple III)

Bruno Lemaire - Gilles Mauffrey

Tout utilisateur d'Apple *III*, surtout s'il a aussi utilisé un Apple II, s'est sans doute posé au moins une fois les questions suivantes : qu'y a-t-il derrière le rempart mystérieux du SOS ? Peut-on utiliser le langage machine de l'Apple *III* aussi commodément que celui de son petit frère, puisqu'ils utilisent tous deux le même processeur, le 6502 ?

Nous allons donc essayer de donner dans cet article quelques éclaircissements sur l'utilisation et l'implantation mémoire des principales routines moniteur de l'Apple *III*, en réservant à un article ultérieur la présentation de la structure du SOS.

Structure de la mémoire

A. Généralités

La mémoire de l'Apple *III* peut être considérée comme essentiellement composée de zones (banques) de 32 K-octets ; une exception est faite pour la zone s'étendant de $F000 à $FFBF qui contient normalement le moniteur en ROM, mais qui peut être « switchée » pour contenir une zone RAM du SOS.

Le nombre de ces zones est variable, et dépend de la configuration de l'utilisateur. Il y a ainsi 4 zones pour un Apple 128 K, et bien sûr 8 zones pour un Apple 256 K. Il n'y a cependant que 64 K directement adressables par l'utilisateur, décomposés en une banque « système » allant de $0000 à $1FFF, et de $A000 à $FFFF, et en une banque utilisateur allant de $2000 à $9FFF. C'est cette banque qui est « switchable » par une simple modification du contenu de $FFEF. Ce contenu varie de F0 à F2 pour un 128 K, et de F0 à F6 pour un 256 K.

Si l'on veut implanter en mémoire un programme machine, on dispose donc sans problème d'au moins 32 K, plus une partie de la banque système non utilisée par le SOS.

Par contre, si l'on désire utiliser davantage de mémoire en modifiant le numéro de la banque utilisée, il faut veiller à ce que la partie du programme qui veut adresser d'autres banques soit sur la banque système.

B. Zones particulières

1) En dehors de la page zéro traditionnelle (réservée pour différents pointeurs et pour l'adressage indirect du 6502), et de la pile système en page 1 (de $100 à $1FF), d'autres zones font fonction de page zéro (adressage indirect étendu, uniquement lorsque le SOS est actif) et de pile pour le SOS et pour le programme machine chargé au moment du boot.

2) Mémoire écran.

— Pour le texte. En 40 colonnes, elle utilise les positions de $400 à $7FF. En 80 colonnes, elle utilise aussi $800 à $BFF ; les caractères de rang pair sont alors stockés de la page $4 à la page $7, les autres étant placés entre les pages $8 et $B.

— Pour les graphiques. Commence en page $20.

Commandes moniteur

Symbole	Syntaxe	Utilisation
G	nG	Exécute le programme commençant en n
J	nJ	Saute à l'octet n
S	octet < m.pS	Recherche l'existence d'un octet entre m et n
M	n<m.pM	Copie la zone comprise entre m et p à partir de n
V	même syntaxe que M	(= Verify)
U		Jump en 3F8 pour routine utilisateur (User)
R	bloc<buf1.buf2R	Transfère une zone disque en zone mémoire
W	bloc<buf1.buf2W	Fait l'opération inverse. **Attention :** opération potentiellement dangereuse !

m, n et p sont des adresses en hexadécimal. buf1 et buf2 désignent le début et la fin du buffer de données.

Routines moniteur

Les débuts de chaque routine sont données en hexadécimal ; celles-ci sont nommées d'après la correspondance qu'elles ont avec celles de l'Apple II.

F9AE — PRBYTE (print a hexadecimal digit) envoie le contenu de l'accumulateur sur l'organe de sortie

F9B7 — PRHEX envoie le quartet bas de l'accumulateur sur l'organe de sortie

FB7D — HOME

FBAF — CROUT1 (return with clear) : nettoie l'écran à partir de la position courante du curseur, puis appelle CROUT (= retour chariot)

FC06 — COUT 1 : envoie le caractère qui est dans l'accumulateur.

FCD5 — GETLN : collecte une ligne de 80 caractères

FD07 — CROUT : envoie un retour chariot

FD0C — KEYIN : lit le clavier de l'Apple *III*

FD60 — RDKEY : traite le caractère saisi au clavier

Clefs

Le préfixe « e » est mis pour Escape.

e.L : nettoie fin de ligne (e.E en A II)

e.P : nettoie bas de l'écran (e.F en A II) à partir du curseur

e.S : Home

e.flèche : déplacement dans le sens de la flèche

e.8 : passage en 80 colonnes

e.4 : passage en 40 colonnes

(pour sortir du mode Escape, taper deux fois d'affilée sur Escape).

Nous terminons ici cette première présentation, remettant à un autre article une analyse de la structure du SOS. Bien entendu, le titre de cet article vous indique la façon d'obtenir le mode moniteur. Il est difficile de le découvrir par tâtonnements !

Cryptographie à clef publique

Olivier Herz

Le programme ci-joint intéressera sans doute un grand nombre d'industriels ainsi que tous ceux qui sont préoccupés par le secret de leurs communications : il s'agit de l'implémentation sur Apple II en Pascal d'un système de cryptographie à clef publique dont nous pouvons assurer l'inviolabilité même avec l'aide des plus puissants ordinateurs. Pour l'utiliser, il n'est pas nécessaire de connaître le langage Pascal, mais il faut savoir se servir du SYSTEM.EDITOR et du SYSTEM.FILER.

Cryptosystème à clef publique

Dans un cryptosystème conventionnel, l'expéditeur et le destinataire du message possèdent tous deux une clef secrète (c'est-à-dire un ensemble de paramètres, habituellement des grands nombres choisis au hasard) permettant d'exécuter l'algorithme de codage du message par l'expéditeur et celui de décodage par le destinataire. Ce message, une fois codé, peut être transmis par un moyen de communication non secret, comme la radio. Il est clair que la fiabilité d'un tel système repose sur le secret de la clef, qui doit être distribuée au moyen de canaux sûrs, afin de ne pas être interceptée.

Dans un cryptosystème à clef révélée (ou publique), on n'a pas besoin d'un canal secret pour distribuer les clefs. Au contraire, chaque utilisateur possède deux clefs : une clef de codage qu'il rend publique et une clef de décodage privée. L'expéditeur code donc le message avec la clef publique du destinataire, qui le décode avec sa clef privée. L'inviolabilité réside dans le fait qu'on déduit très facilement par le calcul la clef publique à partir de la clef privée alors que l'inverse exigerait des milliers d'années de calcul sur ordinateur, voire plus ! C'est ce qu'on appelle une fonction à sens unique et à gâche : ce calcul inverse ne peut se faire qu'en connaissant une information secrète, la gâche de la fonction.

```
(*SYSTEME COMPLET DE CRYPTOGRAPHIE A CLEF PUBLIQUE*)
(*$C (C) OLIVIER HERZ OCTOBRE 1982 POUR POM'S*)
PROGRAM CRYPTOGRAPHIE;
USES APPLESTUFF;
VAR COMMANDE:CHAR;

PROCEDURE MENU;
CONST TAILLE   =107;
      LONGUEUR =36;
TYPE  ENTIER   =INTEGER[LONGUEUR];
      CHOIDECA =SET OF CHAR;
VAR   BS,CR          :CHAR;
      NOM            :STRING;
      INVFACTEUR,
      BASE,FACTEUR   :ENTIER;
      CLEF           :ARRAY[1..TAILLE] OF ENTIER;
      PUBLIQUE       :TEXT;
      PRIVEE         :FILE OF ENTIER;
      CODE,CLAIR     :TEXT;
      MODTAB         :ARRAY[1..LONGUEUR] OF ENTIER;

(*************************************)
(* PROCEDURES HABITUELLES COURANTES *)
(*     MERCI MICHEL CRIMONT!         *)
(*************************************)

PROCEDURE MESSAGE(I:INTEGER; S:STRING);
BEGIN GOTOXY(0,I); WRITE(S,CHR(11)) END;

FUNCTION PRENCAR(BONSET:CHOIDECA):CHAR;
VAR CH :CHAR;
BEGIN
  REPEAT
    READ(KEYBOARD,CH); IF EOLN(KEYBOARD) THEN CH:=CR;
    IF NOT (CH IN BONSET) THEN WRITE(CHR(7))
      ELSE IF CH IN[' '..'z'] THEN WRITE(CH)
  UNTIL CH IN BONSET;
  PRENCAR:=CH
END;

PROCEDURE OUT(S:STRING);
VAR CH :CHAR;
BEGIN
  CLOSE(PRIVEE); CLOSE(PUBLIQUE); CLOSE(CODE); CLOSE(CLAIR);
  PAGE(OUTPUT); MESSAGE(10,S); WRITE(CHR(7));
  MESSAGE(12,'  APPUYEZ SUR <RETURN>');
  CH:=PRENCAR([CR]); EXIT(MENU)
END;

PROCEDURE SORTIE;
BEGIN OUT('CLEF OU MESSAGE NON VALIDE') END;

PROCEDURE DEHORS;
BEGIN OUT('TITRE OU LECTEUR NON VALIDE') END;

PROCEDURE PRENENTIER(S:STRING; VAR ENT:ENTIER);
VAR I :1..TAILLE;
BEGIN
  ENT:=0;
  IF (LENGTH(S)=0) OR (LENGTH(S)>36) THEN SORTIE;
  FOR I:=1 TO LENGTH(S) DO
  BEGIN
    IF NOT (S[I] IN ['0'..'9']) THEN SORTIE;
    ENT:=ENT*10+(ORD(S[I])-48)
  END
END;

PROCEDURE NOMFICHIER(S:STRING; I:INTEGER);
VAR S1      :STRING[1];
    BONSET :CHOIDECA;
BEGIN
  CLOSE(PRIVEE); CLOSE(PUBLIQUE); CLOSE(CODE); CLOSE(CLAIR);
```

La méthode

Les systèmes cryptographiques actuels tendent de plus en plus à utiliser des problèmes dits de type NP (non polynomiaux) caractérisés par le fait que le nombre d'opérations nécessaires à leur résolution croît comme une fonction exponentielle, et non polynomiale, de leur taille.

Notre programme utilise un cryptosystème basé sur le problème NP de l'empilement : étant donné n jetons de hauteurs respectives A_1 A_n et une hauteur C, comment peut-on obtenir C en empilant un sous-ensemble de ces n jetons ? On ne connaît actuellement pas d'autre solution que la recherche systématique de tous les empilements possibles (au nombre de 2 puissance n). Par contre, ce problème est très facile dans un cas particulier : quand la hauteur d'un jeton est supérieure à la somme des hauteurs de ceux qui le précèdent (i.e. A_i somme de 1 à i-1 des A_k. Il suffit alors pour le résoudre d'essayer de soustraire de C le dernier jeton ; si c'est impossible, on essaye l'avant-dernier, etc. jusqu'à ce que l'on y arrive ; puis l'on recommence le processus avec le résultat de la soustraction, etc... On parvient ainsi à la solution en n opérations au plus. On conçoit donc que l'on utilisera comme clef privée les jetons d'un problème facile et comme clef publique ceux d'un problème difficile.

Explication des calculs

On note $B_1...B_n$ les « hauteurs » du « problème facile » de TAILLE n ; soient M un très grand nombre (la BASE) supérieur à la somme des B_i et F (le FACTEUR) inférieur à M. F doit être premier avec M afin que l'on puisse trouver son inverse modulo M (l'INVFACTEUR E, tel que E.F = 1 modulo M). Ceci constitue la clef privée et la gâche est formée de M et de E. On calcule alors les n « hauteurs » $A_1...A_n$ du « problème difficile » par la formule $A_i = (B_iF)$ moduloM ; les A_i forment la clef publique.

Pour le codage, on décompose le texte en bits valant 0 ou 1 ; nous avons utilisé pour coder un caractère les 6 chiffres de la décomposition binaire de la différence entre son code ASCII et 32 : c'est le tableau BIT de la procédure CODAGE ; les minuscules, de code ASCII supérieur à 96, ont été remises en majus-

```
  PAGE(OUTPUT); MESSAGE(10,S); S1:=' '; NOM:='';
  BONSET:=['0'..'9','A'..'Z','a'..'z','#',':'];
  REPEAT
    IF LENGTH(NOM)=0 THEN S1[1]:=PRENCAR(BONSET)
      ELSE IF LENGTH(NOM)=1 THEN S1[1]:=PRENCAR([CR,BS])
             ELSE S1[1]:=PRENCAR(BONSET+[CR,BS]);
    IF S1[1] IN BONSET THEN NOM:=CONCAT(NOM,S1)
      ELSE IF S1[1]=BS THEN BEGIN
                              WRITE(BS,' ',BS);
                              DELETE(NOM,LENGTH(NOM),1)
                            END
  UNTIL S1[1]=CR;
  IF POS('#',NOM)=0 THEN NOM:=CONCAT('#4:',NOM);
  IF LENGTH(NOM)>I THEN NOM:=COPY(NOM,1,I)
END;

FUNCTION PLUSGRAND(A,B:ENTIER):BOOLEAN;
VAR U,V :STRING;
BEGIN
  STR(A,U); STR(B,V);
  IF LENGTH(U)>LENGTH(V) THEN PLUSGRAND:=TRUE
    ELSE IF LENGTH(U)<LENGTH(V) THEN PLUSGRAND:=FALSE
           ELSE PLUSGRAND:=U>V
END;

FUNCTION EGAL(A,B:ENTIER):BOOLEAN;
VAR U,V :STRING;
BEGIN STR(A,U); STR(B,V); EGAL:=U=V END;

FUNCTION RND:INTEGER;
BEGIN RND:=RANDOM DIV 16384 END;

PROCEDURE ECRIS(I:INTEGER);
BEGIN WRITE('.'); IF I=TAILLE DIV 2 THEN WRITELN END;

PROCEDURE FAITMODTAB(ENT:ENTIER);
VAR I :1..LONGUEUR;
BEGIN
  FOR I:=1 TO LONGUEUR DO
  BEGIN
    MODTAB[I]:=ENT-(ENT DIV BASE)*BASE; ENT:=10*MODTAB[I]
  END
END;

PROCEDURE MODULO(VAR ENT:ENTIER);
VAR I :1..LONGUEUR;
    S :STRING;
BEGIN
  STR(ENT,S); ENT:=0;
  FOR I:= 1 TO LENGTH(S) DO
    ENT:=ENT+(ORD(S[I])-48)*MODTAB[LENGTH(S)+1-I];
  ENT:=ENT-(ENT DIV BASE)*BASE
END;

(**********)
(* CODAGE *)
(**********)

PROCEDURE CODAGE;
VAR I,J,K     :INTEGER;
    CH        :CHAR;
    S         :STRING;
    BIT       :ARRAY[0..63,1..6] OF 0..1;
    ENT       :ENTIER;

BEGIN (*CODAGE*)
  NOMFICHIER('NOM DU CORRESPONDANT: ',13);
  (*$I-*)
  RESET(PUBLIQUE,CONCAT(NOM,'.TEXT'));
  IF IORESULT<>0 THEN DEHORS;
  (*$I+*)
  PAGE(OUTPUT); WRITELN('LECTURE DE LA CLEF PUBLIQUE');
  FOR I:=1 TO TAILLE DO
  BEGIN
    IF EOF(PUBLIQUE) THEN SORTIE; ECRIS(I);
    READLN(PUBLIQUE,S); PRENENTIER(S,CLEF[I])
  END;
  NOMFICHIER('NOM DU MESSAGE: ',11);
  (*$I-*)
```

cules et les caractères de contrôle, de code inférieur à 32, ont été remplacés par des espaces, sauf le retour chariot, de code 13, remplacé par le code 95 qui représente un caractère inaccessible au clavier et donc peu usité. On découpe le message en groupes successifs de n bits $X_1 \ldots X_n$. Si le dernier groupe contient moins de n bits, on le complète par des bits aléatoires. Le codage consiste à calculer pour chaque groupe la hauteur C, somme de 1 à n des $A_i X_i$.

Pour décoder, on commence par calculer D = C.E modulo M, puis l'on résoud le problème d'empilement facile avec les B_i de façon à trouver les n bits $Y_1 \ldots Y_n$ tels que D soit la somme des $B_i Y_i$. Or on voit aisément que les Y_i sont nécessairement égaux aux X_i ; ils permettent donc de reconstituer le texte original (en n'oubliant pas de remplacer le caractère de code ASCII 95 par un retour chariot).

Réalisation pratique

L'obligation d'utiliser de très grands nombres rendait pratique l'emploi du langage Pascal qui possède des entiers ayant déjà 36 chiffres (les grands entiers). De plus, la portabilité du Pascal permet d'utiliser le programme CCP sur d'autres ordinateurs.

Le programme utilise et stocke les clefs et messages sous forme de fichiers. Hormis la clef privée, qui est un fichier de type DATA contenant des grands entiers (afin d'éviter que des indiscrets ne la lisent), la clef publique et les messages codé et clair sont des fichiers TEXT qui peuvent être lus ou écrits avec le SYSTEM.EDITOR (la clef publique et le message codé étant composés de grands entiers transformés en chaînes de caractères).

Le programme est en fait une grande procédure (MENU) destinée à permettre le retour au menu en cas d'erreur par une instruction EXIT (voir les procédures SORTIE, DEHORS et OUT). Les procédures ou fonctions MESSAGES et PRENCAR sont classiques et RND fournit un bit aléatoire. PRENENTIER fait l'inverse de la procédure Pascal STR en transformant une chaîne en un grand entier. NOMFICHIER prend au clavier le nom d'un fichier sans le suffixe qu'elle rajoute automatiquement (.TEXT et .DATA pour les

```
RESET(CLAIR,CONCAT(NOM,'.0.TEXT'));
IF IORESULT=0 THEN REWRITE(CODE,CONCAT(NOM,'.1.TEXT'));
IF IORESULT<>0 THEN DEHORS;
(*$I+*)
FOR I:=0 TO 63 DO
BEGIN
  K:=I; FOR J:=1 TO 6 DO
          BEGIN BIT[I,J]:=K MOD 2; K:=K DIV 2 END
END;
PAGE(OUTPUT); WRITELN('CODAGE'); I:=TAILLE; ENT:=0;
REPEAT
  IF EOLN(CLAIR) THEN BEGIN READLN(CLAIR); WRITELN; K:=63 END
    ELSE BEGIN
           READ(CLAIR,CH); K:=ORD(CH)-32; IF K<0 THEN K:=0;
           IF K>63 THEN K:=K-32; WRITE(CHR(K+32))
         END;
  FOR J:=6 DOWNTO 1 DO
  BEGIN
    ENT:=ENT+CLEF[I]*BIT[K,J];
    I:=I-1; IF I=0 THEN BEGIN
                          STR(ENT,S); WRITELN(CODE,S);
                          I:=TAILLE; ENT:=0
                        END
  END;
  IF (EOF(CLAIR)) AND (I<>TAILLE) THEN
  BEGIN
    REPEAT ENT:=ENT+CLEF[I]*RND; I:=I-1 UNTIL I=0;
```

Suite du listing page 11

Programme VÉRIFICATION

Suite et fin de l'article

Effacement du directory en Pascal

Pages 59-60

```
{$I-}

PROGRAM VERIFICATION;

VAR F:FILE OF INTEGER;
    CL:CHAR;
    J:INTEGER;
    PO:^INTEGER;


BEGIN
  PAGE(OUTPUT);
  WRITE(CHR(12));
  WRITELN;
  WRITELN('PROGRAMME DE DEMONSTRATION DE L''EFFICACITE');
  WRITELN(' DU PROCEDE D''EFFACEMENT DU DIRECTORY EN MEMOIRE');
  WRITELN;
  WRITELN('INSERER UNE DISQUETTE PROGRAMME VIERGE et frapper');
  WRITELN('"O" POUR LIRE LE DIRECTORY DE LA NOUVELLE DISQUETTE PROGRAMME');
  WRITELN('ou une autre touche POUR CONSERVER L''ANCIEN DIRECTORY');
  READ(CL);WRITELN;WRITELN;
  J:=0;
  IF CL='O' THEN MARK(PO); { efface en memoire le directory de la disquette }
                           { PROGRAMME sur laquelle se trouve le programme  }
  REWRITE(F,'TEST');
  F^:=J;PUT(F);
  CLOSE(F,LOCK);
  IF CL='O'
     THEN WRITELN('IL N''Y A PAS D''ERREUR DE DIRECTORY')
     ELSE WRITELN('LE DIRECTORY DE LA VERITABLE DISQUETTE APPLE1 EST ',
                  'MAINTENANT SUR LA FAUSSE');
WRITELN('REMETTRE LA VERITABLE DISQUETTE PROGRAMME et frapper la barre espace');
  READ(CL);
  MARK(PO);
END.
```

clefs publique et privée, .O.TEXT et .1.TEXT pour les messages en clair et codé). S'il n'y a pas de préfixe, elle rajoute #4 : (le préfixe doit indiquer le numéro du drive et non le nom de la disquette). PLUSGRAND et EGAL comparent deux grands entiers (en théorie, rien n'empêche de comparer deux grands entiers comme on le fait avec des entiers ou des réels, mais en pratique cela produit des erreurs et des « plantages » à l'exécution ; cette cause d'erreurs ne fut pas évidente à découvrir et reste mystérieuse !). FAITMODTAB et MODULO permettent de calculer les expressions B_i.F modulo M et C.E modulo M (alors que les poduits B_i.F et C.E excèdent les 36 chiffres) grâce au tableau MODTAB qui contient les valeurs modulo M des produits de F ou E par les puissances successives de 10.

Les procédures CODAGE et DECODAGE comportent deux parties : la demande des noms des fichiers puis le codage ou le décodage proprement dits. Si l'opération de décodage rencontre un problème d'empilement ne pouvant être résolu (mauvaise clef ou mauvais message), elle s'arrête (on peut la laisser continuer si l'on supprime la ligne avant le UNTIL).

La procédure ALEATOIRE fournit un grand entier aléatoire compris entre INF et SUP-1. La méthode utilisée est la dichotomie qui, bien que longue d'exécution, est la plus facile à mettre en œuvre avec des grands entiers. La procédure récursive BEZOUT permet de résoudre par l'algorithme d'EUCLIDE l'équation de BEZOUT $AX-BY=1$ quand A<B. Elle est utilisée ici avec A = F, X = E, B = M et Y = K où EF = KM + 1. La procédure GENERATION commence par calculer la clef privée en se fixant pour chacun de ses entiers des limites supérieure et inférieure confortables (SUP et INF) ; puis elle calcule la clef publique.

Notons que la constante TAILLE vaut 107, ce qui est à peu près le maximum que l'on puisse prendre avec ce programme. Mais on peut parfaitement utiliser ce programme avec une valeur plus faible. Remarquons que le calcul exhaustif de tous les empilements lorsqu'on a 107 jetons (i.e. le décodage de C) prendrait avec un ordinateur étudiant 1000 milliards d'empilements par seconde (ne rêvons pas !) 5000 milliards d'années ! Remarquons encore que 107 n'est pas multiple de 6, le nombre de bits par caractères, ce qui permet d'embrouiller encore plus un éventuel décrypteur en codant certaines lettres à cheval sur deux empilements. On peut prendre une TAILLE multiple de 6 ; chaque empilement correspond alors exactement à TAILLE/6 lettres, ce qui le rend autonome.

Mode d'emploi

Signalons tout d'abord que pour envoyer un fichier TEXT à un correspondant on peut, soit lui envoyer une disquette, soit utiliser un modem, soit encore imprimer et envoyer (par radio ou par courrier) le texte afin qu'il soit édité et sauvé.

Dans un premier temps, chaque utilisateur du système CCP fabrique un système de clefs. M. Dupont par exemple répondra G et DUPONT (ou bien # 5 : DUPONT) au programme CCP et l'Apple fabriquera sur la même disquette les fichiers DUPONT.TEXT (clef publique qu'il devra fournir à ses interlocuteurs) et DUPONT.DATA (clef privée qu'il devra garder sur une disquette dans un coffre à l'abri).

Pour envoyer le message MISSIVE à M. Durand, M. Dubois commencera par éditer son texte (en y glissant si possible de nombreux signes de ponctuation ne gênant pas la compréhension du texte, ce qui rend encore plus difficile tout essai de décryptage) et il le sauvera sous le nom MISSIVE.O.TEXT. Puis, ayant vérifié qu'il possède le fichier DURAND.TEXT, il répondra C, DURAND (ou bien # 5 :DURAND) et MISSIVE (ou bien # 5 :MISSIVE) au programme CCP qui fabriquera le fichier codé MISSIVE.1.TEXT (sur la disquette où se trouve MISSIVE.O.TEXT), qui devra parvenir à M. Durand.

Pour le décodage, ce dernier sortira le fichier DURAND.DATA de son coffre et répondra D, DURAND (ou bien # 5 :DURAND) et MISSIVE (ou bien # 5 :MISSIVE) au programme CCP qui recréera pour lui le fichier MISSIVE.O.TEXT (sur la disquette où se trouve MISSIVE 1.TEXT). Notons que pendant les opérations de codage et de décodage, le message en clair apparaît sur l'écran au fur et à mesure de sa lecture ou de son écriture.

En conclusion, nous avons là un système cryptographique très simple d'emploi et parfaitement sûr. Les temps d'exécution sont de 6 minutes pour écrire le système de clefs, d'environ 9 cps (caractères par seconde) pendant le codage et d'environ 5 cps pendant le décodage. Mais ces temps diminuent fortement avec la constante TAILLE et il y a un compromis à trouver entre la vitesse d'exécution et la sécurité du codage. Remarquons que le cryptosystème utilisé ne permet malheureusement pas l'authentification de l'expéditeur, contrairement à un système plus récent, développé au M.I.T., dans lequel les clefs publique et privée ont des rôles symétriques. Mais ce dernier système est pratiquement impossible à mettre en œuvre sur Apple II car il utilise de très grands nombres premiers.

Bibliographie

« La guerre des codes secrets » par David Kahn chez Inter Editions est un excellent ouvrage sur l'histoire du chiffre. Dans « Les progrès des mathématiques », Pour la Science publie un recueil de ses articles de mathématiques dont « Les mathématiques de la cryptographie à clef révélée », écrit par l'un des auteurs du système à empilement.

```
      STR(ENT,S); WRITELN(CODE,S);
    END
  UNTIL EOF(CLAIR);
  CLOSE(CLAIR); CLOSE(CODE,LOCK)
END; (*CODAGE*)

(************)
(* DECODAGE *)
(************)

PROCEDURE DECODAGE;
VAR I,J,K :INTEGER;
    S     :STRING;
    ENT   :ENTIER;

BEGIN (*DECODAGE*)
  NOMFICHIER('VOTRE NOM: ',13);
  (*$I-*)
  RESET(PRIVEE,CONCAT(NOM,'.DATA'));
  IF IORESULT<>0 THEN DEHORS;
  (*$I+*)
  PAGE(OUTPUT); WRITELN('LECTURE DE LA CLEF PRIVEE');
  FOR I:=1 TO TAILLE DO
  BEGIN
    IF EOF(PRIVEE) THEN SORTIE; ECRIS(I);
    CLEF[I]:=PRIVEE^; GET(PRIVEE)
  END;
  IF EOF(PRIVEE) THEN SORTIE; BASE:=PRIVEE^; GET(PRIVEE);
  IF EOF(PRIVEE) THEN SORTIE; INVFACTEUR:=PRIVEE^;
  NOMFICHIER('NOM DU MESSAGE: ',11);
  (*$I-*)
  RESET(CODE,CONCAT(NOM,'.1.TEXT'));
  IF IORESULT=0 THEN REWRITE(CLAIR,CONCAT(NOM,'.0.TEXT'));
  IF IORESULT<>0 THEN DEHORS;
  (*$I+*)
  FAITMODTAB(INVFACTEUR);
  PAGE(OUTPUT); WRITELN('DECODAGE'); J:=0; K:=0;
  REPEAT
    READLN(CODE,S); PRENENTIER(S,ENT); MODULO(ENT);
    FOR I:=TAILLE DOWNTO 1 DO
    BEGIN
      J:=J+1; K:=2*K+1;
      IF PLUSGRAND(CLEF[I],ENT) THEN K:=K-1
        ELSE ENT:=ENT-CLEF[I];
      IF J=6 THEN
      BEGIN
        IF K=63 THEN BEGIN WRITELN; WRITELN(CLAIR) END
          ELSE BEGIN WRITE(CHR(K+32)); WRITE(CLAIR,CHR(K+32)) END;
        J:=0; K:=0
      END
    END;
    IF NOT EGAL(ENT,0) THEN SORTIE
  UNTIL EOF(CODE);
  CLOSE(CODE); CLOSE(CLAIR,LOCK)
END; (*DECODAGE*)

(************************)
(* GENERATION D'UNE CLEF *)
(************************)
```

```
PROCEDURE GENERATION;
VAR I                :1..TAILLE;
    S                :STRING;
    BON              :BOOLEAN;
    INF,SUP,BIDON    :ENTIER;

PROCEDURE ALEATOIRE(VAR ENT:ENTIER);
VAR I    :1..4;
    A,B  :ENTIER;
BEGIN
  A:=INF; B:=SUP;
  REPEAT
    ENT:=(A+B) DIV 2;
    IF NOT EGAL(A+B,2*ENT) THEN ENT:=ENT+RND;
    IF RND=0 THEN B:=ENT ELSE A:=ENT
  UNTIL PLUSGRAND(3,B-A);
  ENT:=A; IF EGAL(2,B-A) THEN ENT:=ENT+RND
END;

PROCEDURE BEZOUT(A,B:ENTIER; VAR X,Y:ENTIER);
VAR Q,R :ENTIER;
BEGIN
  X:=0; Y:=0;
  IF BON THEN
  BEGIN
    Q:=B DIV A;
    IF EGAL(B,Q*A) THEN IF EGAL(A,1) THEN
                          BEGIN Y:=1; X:=B+1 END
                          ELSE BON:=FALSE
      ELSE BEGIN
             B:=B-Q*A; R:=A DIV B;
             IF EGAL(A,R*B) THEN IF EGAL(B,1) THEN
                                   BEGIN Y:=A-1; X:=1+Q*Y END
                                   ELSE BON:=FALSE
               ELSE BEGIN
                      A:=A-R*B; BEZOUT(A,B,X,Y);
                      Y:=Y+R*X; X:=X+Q*Y
                    END
           END
  END
END;

BEGIN (*GENERATION*)
  NOMFICHIER('VOTRE NOM: ',13);
  (*$I-*)
  REWRITE(PRIVEE,CONCAT(NOM,'.DATA'));
  IF IORESULT<>0 THEN DEHORS;
  (*$I+*)
  SUP:=10; INF:=3;
  PAGE(OUTPUT); WRITELN('ECRITURE DE LA CLEF PRIVEE');
  FOR I:=1 TO TAILLE DO
  BEGIN
    ALEATOIRE(CLEF[I]); PRIVEE^:=CLEF[I]; PUT(PRIVEE);
    ECRIS (I); INF:=INF+CLEF[I]; SUP:=2*SUP
  END;
  INF:=SUP; SUP:=2*SUP;
  ALEATOIRE(BASE); PRIVEE^:=BASE; PUT(PRIVEE);
  WRITE('.'); INF:=BASE DIV 2; SUP:=BASE;
```

```
  REPEAT
    ALEATOIRE(FACTEUR); WRITE('.');
    BON:=TRUE; BEZOUT(FACTEUR,BASE,INVFACTEUR,BIDON)
  UNTIL BON;
  PRIVEE^:=INVFACTEUR; PUT(PRIVEE); CLOSE(PRIVEE,LOCK);
  (*$I-*)
  REWRITE(PUBLIQUE,CONCAT(NOM,'.TEXT'));
  IF IORESULT<>0 THEN DEHORS;
  (*$I+*)
  WRITELN; WRITELN; WRITELN('ECRITURE DE LA CLEF PUBLIQUE');
  FAITMODTAB(FACTEUR);
  FOR I:=1 TO TAILLE DO
  BEGIN
    MODULO(CLEF[I]); ECRIS(I);
    STR(CLEF[I],S); WRITELN(PUBLIQUE,S)
  END;
  CLOSE(PUBLIQUE,LOCK)
END; (*GENERATION*)

BEGIN (*MENU*)
  CR:=CHR(13); BS:=CHR(8); PAGE(OUTPUT); RANDOMIZE;
  MESSAGE(2,'       OLIVIER HERZ PRESENTE C.C.P.');
  MESSAGE(5,'            SYSTEME COMPLET DE');
  MESSAGE(7,'      CRYPTOGRAPHIE A CLEF PUBLIQUE.');
  MESSAGE(11,'     G)ENERER VOTRE SYSTEME DE CLEFS');
  MESSAGE(13,'     C)ODER UN MESSAGE');
  MESSAGE(15,'     D)ECODER UN MESSAGE');
  MESSAGE(17,'     Q)UITTER LE SYSTEME C.C.P.');
  MESSAGE(20,'  VOTRE CHOIX:');
  COMMANDE:=PRENCAR['G','C','D','Q','g','c','d','q'];
  CASE COMMANDE OF
    'G','g':GENERATION;
    'C','c':CODAGE;
    'D','d':DECODAGE
  END
END; (*MENU*)

BEGIN (*CRYPTOGRAPHIE*)
  REPEAT MENU UNTIL COMMANDE IN ['Q','q']
END. (*CRYPTOGRAPHIE*)
```

Etudiants, lycéens, lisez

L'ORDINATEUR & L'ECOLE

numéro hors-série de

L'ORDINATEUR INDIVIDUEL

Lycéens, étudiants, enseignants, parents : l'équipe de la revue L'ORDINATEUR INDIVIDUEL publie un numéro spécial hors-série qui fait le point sur l'ordinateur à l'école.

- **Un ordinateur dans une classe ?**
- **Comment l'enseignement peut-il être facilité par un ordinateur individuel ?**
- **A partir de quel âge peut-on découvrir l'informatique ?**
- **Quelle aide un ordinateur familial apporte-t-il sur le plan éducatif ?**
- **Comment créer et gérer un club d'informatique individuelle dans un établissement scolaire ou universitaire ?**

Toutes les réponses à ces questions (et à d'autres !) figurent dans ce dossier indispensable réalisé à partir d'une synthèse des meilleurs articles parus dans L'ORDINATEUR INDIVIDUEL.

En 1983, il n'est plus raisonnable d'ignorer le formidable outil qu'est l'ordinateur. Pour être mieux informé sur ce domaine, lisez L'ORDINATEUR ET L'ÉCOLE.

25 FF chez votre marchand de journaux

Graphique, quand tu nous tiens...

Guy Mathieu

On peut reprocher aujourd'hui à l'Apple II un écran graphique manquant de finesse : la nouvelle génération de micro-ordinateurs exprime souvent ses aptitudes graphiques en centaines de milliers de points, par exemple 800 × 400, plutôt qu'en dizaines de milliers : 280 × 192. N'oublions quand même pas qu'il y a seulement trois ans l'Apple II était le phénix des hôtes de ces vergers, car, à ce prix, les possibilités graphiques étaient rares.

L'Apple II garde une caractéristique particulière attrayante, celle de la double page graphique. Pour peu que l'on possède une imprimante capable d'en tirer le meilleur parti, nous voilà bien armés. C'est le cas avec l'EPSON MX FT munie de l'interface graphique EPSON pour Apple II.

Quelles sont les possibilités offertes ?

D'abord, celles qu'offrent la plupart des imprimantes graphiques modernes :
— noir sur blanc ou blanc sur noir
— simple ou double dimension
et ceci, bien entendu, en utilisant la page graphique 1 ou 2 au choix.

Ensuite, et c'est la grande originalité, des combinaisons en une même impression du contenu des deux pages.

— impression côte à côte du contenu de chacune des deux pages (ex. 1) ;
— opération logique « ET » (produit logique) donnant l'impression des points communs aux graphes des deux pages, et de ceux-là seulement (ex. 2) ;
— opération logique « OU inclusif » (somme logique) donnant l'impression de tous les points figurant sur au moins l'un des deux graphes (ex. 3) ;
— opération logique « OU exclusif » (différence symétrique) donnant l'impression de tous les points qui figurent sur l'un des deux graphes, et pas sur l'autre (ex. 4).

Ces opérations sont bien sûr compatibles avec les précédentes.

Il existe également une possibilité de n'éditer qu'une fraction de graphe, désignée par sa position sur l'écran, en référence aux lignes et colonnes de même emplacement dans le mode texte.

Le programme ci-après permet l'exploitation par un questionnaire en séquence de ces diverses possibilités (à l'exception de la dernière citée). Il permet aussi :
— de visualiser le ou les graphes choisis, de permuter les deux pages, de revenir au menu ;
— de choisir entre l'impression de graphes déjà existants en machine (il peut alors être utilisé en sous-programme d'un programme graphique) et la recherche sur disques de graphes préenregistrés.

Mais tout ceci n'est-il pas un jeu de l'esprit ? Pour nous convaincre du contraire, nous allons examiner quelques cas d'application.

Graphe double dimension

Cette formule est idéale pour créer :
• des pages de titres,
• des transparents pour rétroprojecteur.

Intersection de deux graphes

Permet par exemple de détecter immédiatement les points communs à deux schémas de circulation afin de détecter d'éventuels «points noirs» (ex. 5).

Union des deux graphes

Permet d'utiliser «un fond de plan» et d'y placer, successivement, divers tracés. Par exemple, une carte sur laquelle on trouvera les villes, puis les richesses minières, etc.

Différence de deux graphes :

Permet de repérer immédiatement les modifications apportées à un graphe entre deux versions successives (ex. 6)
L'imagination des «dingues du graphe» étant sans limites, bien d'autres applications peuvent être trouvées. Il est même probable que d'autres imprimantes offrent ou offriront des possibilités du même ordre.

N.B. L'Apple *IIe*, et autres nouveautés Apple, conserveront-ils cette double page graphique ?

Programme et commentaires

1 à 900 : paramètres d'initialisation et tracés - tests sur pages 1 et 2.
200 : affichage des pages graphiques.
5 000 : déroulement du questionnement.
6 000 : mise en route de l'impression.
10 000 : recherche de graphes sur disques.
30 000 : sous-programme d'attente.

Mode d'emploi

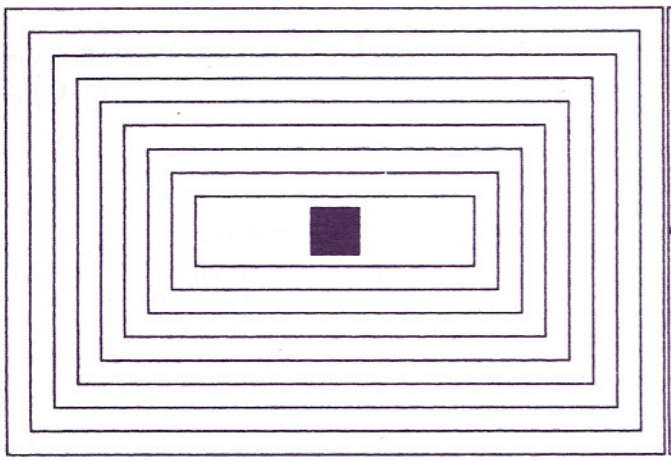
N'oubliez pas que la carte graphique Epson est telle que deux graphes sont toujours tracés côte à côte en format normal. Ces deux graphes, différents dans le cas de notre option COTE A COTE, sont identiques dans tous les autres cas.
Le graphique de la page 18 illustre la séquence des questions posées par le programme à l'utilisateur.
Remarque : si l'on répond 1 (COTE A COTE) à la troisième question, il faudra ultérieurement penser à placer les graphes dans le deux pages pour qu'ils se présentent dans l'ordre convenable à l'édition.

Exemple d'utilisation

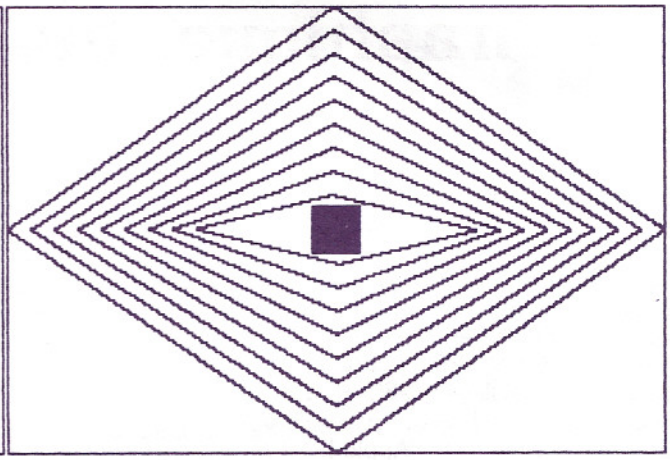
Nous cherchons à imprimer côte à côte deux graphes sauvés sur une disquette sous les nom CHATS 1 et CHATS 2. Il faut exécuter le programme puis placer la disquette dans le lecteur et suivre la séquence suivante :

Question	Réponse	Choix
1	3	Les deux pages
2	1	Normal
3	1	Côte à côte
4	N	Pas en mémoire

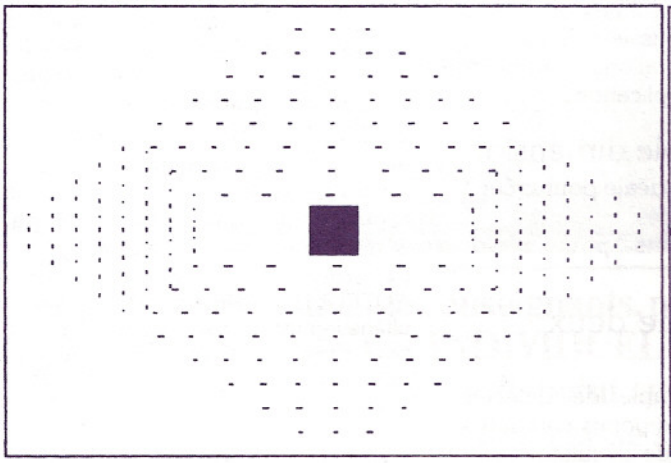
Ensuite, choisir l'option CATALOGUE, amener face à CHATS 1 et faire RETURN. Venir après CHATS 1 et faire RETURN pour le changement. Suivre la même séquence pour CHATS 2. Enfin, dans le menu, choisir 1 (vérification de la page 1), puis 2 (page 2), ESC (retour) et enfin RETURN (impression).



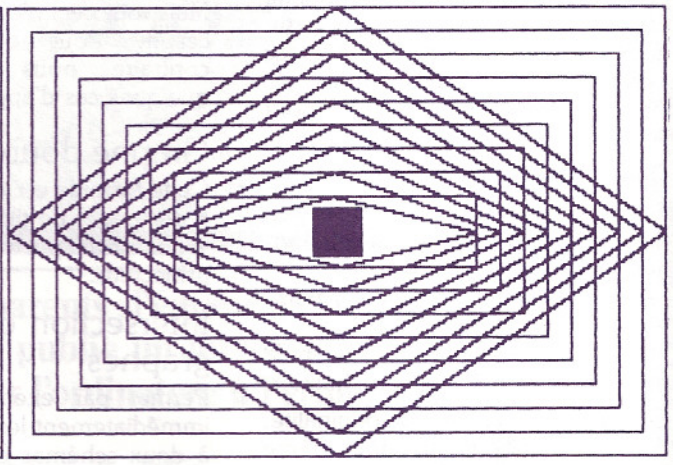
Exemple 1 — Page 1.



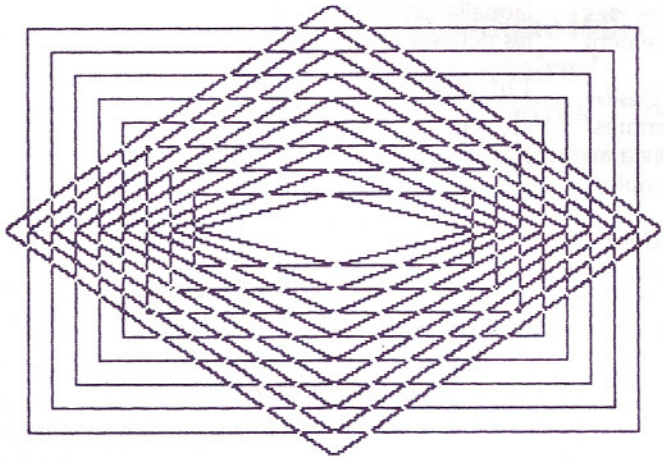
Exemple 1 — Page 2.



Exemple 2 — Page 1 "ET" 2.

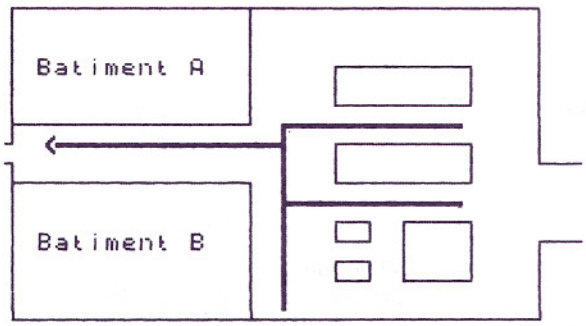


Exemple 3 — Page 1 "OU (inclusif)" page 2.

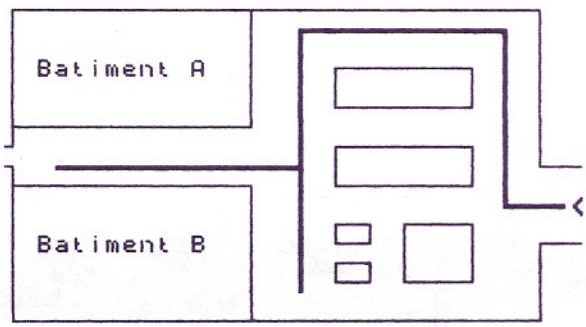


Exemple 4 — Page 1 "OU (exclusif)" page 2.

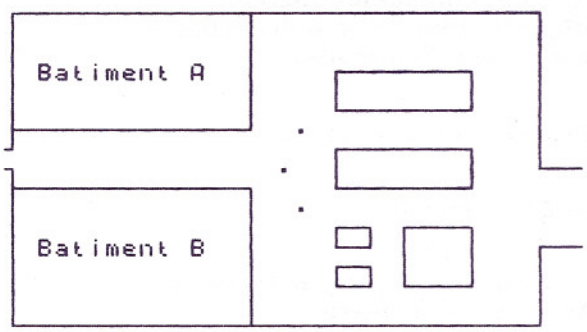
Exemple 5 :
Sécurité Incendie



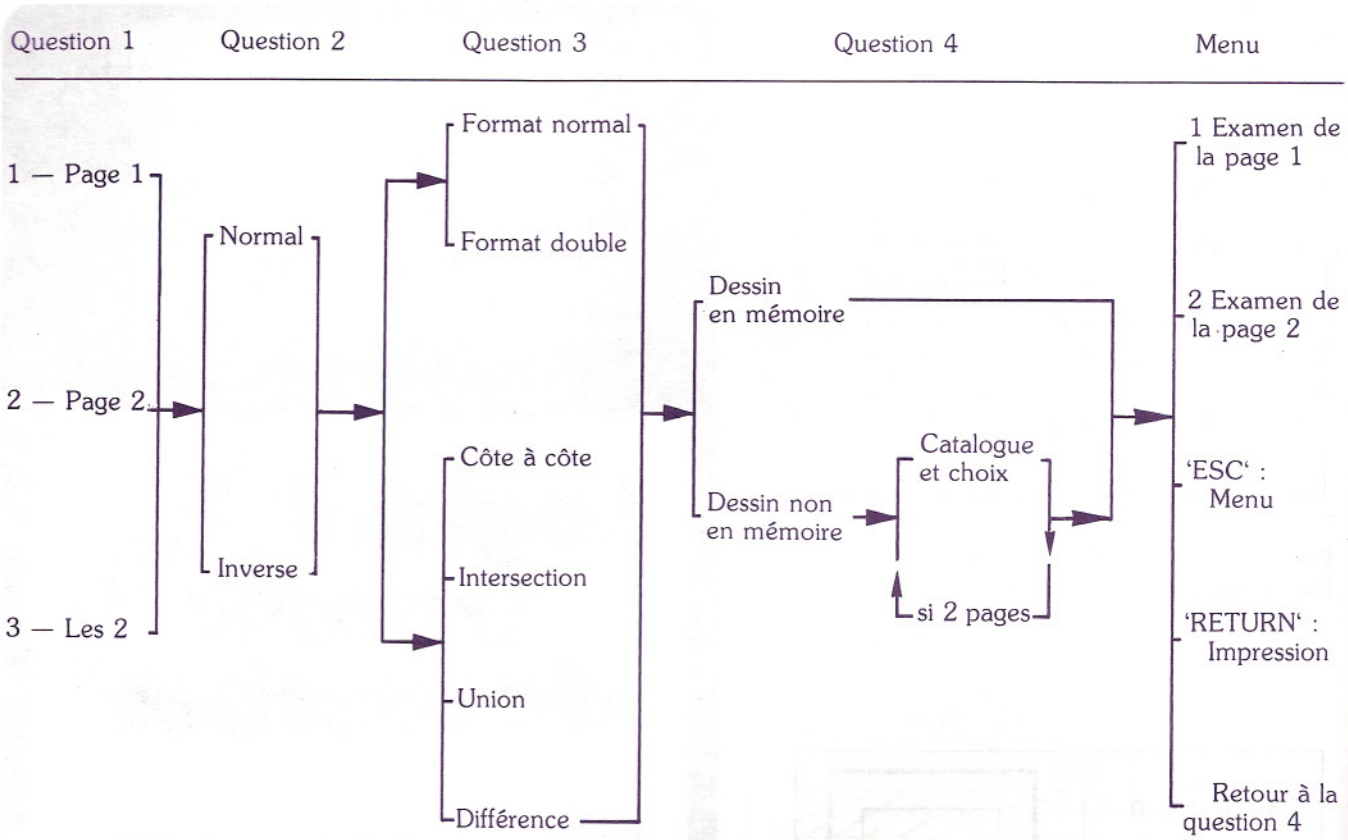
Page 1 — Sécurité Incendie.
Plan d'évacuation des agents.



Page 2 — Sécurité Incendie.
Circulation des véhicules de secours.

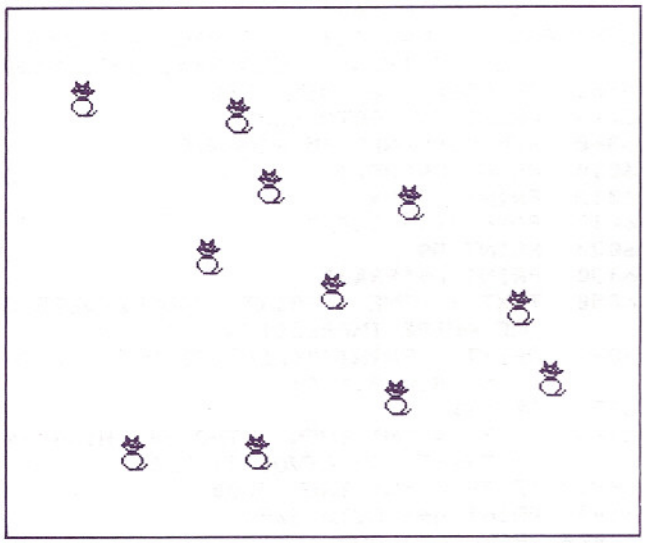


Page 1 "ET" Page 2 — Sécurité Incendie.
Identification des "points noirs".

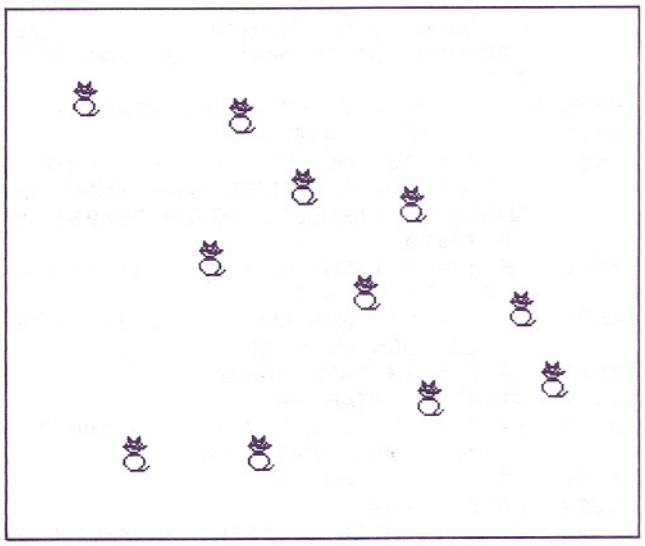


Graphe du mode d'emploi

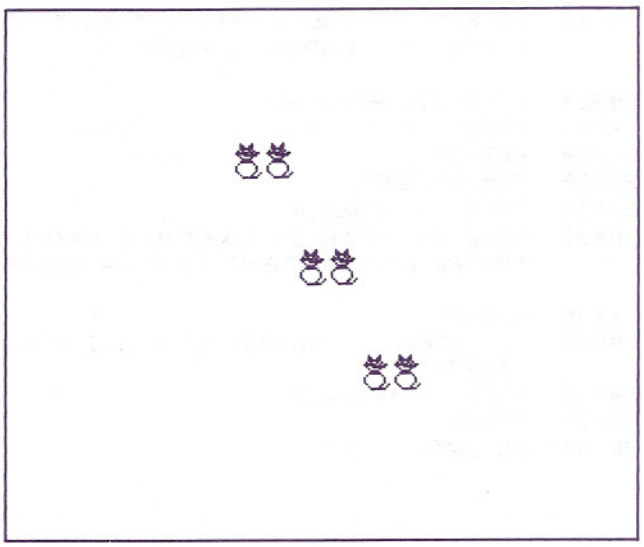
Exemple 6 : les petits chats.



Page 1 — Situation de départ.



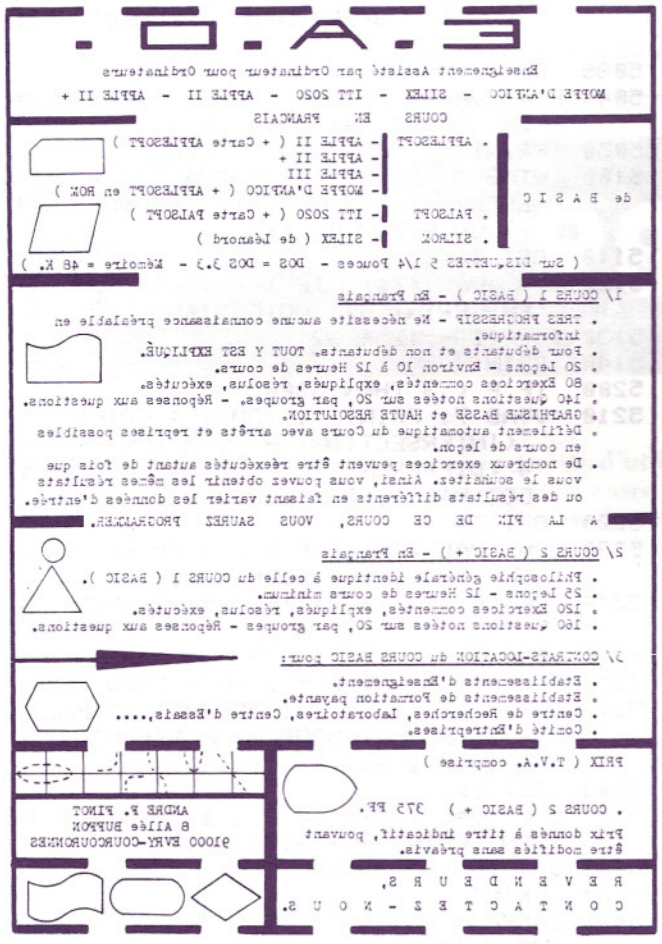
Page 2 — Situation d'arrivée.



SOMME logique : Ceux qui ont bougé entre 1 et 2.

```
1  LOMEM: 25000
2  TEXT
3  HOME
5 DD =  PEEK (40185): IF DD = 0 OR DD >
       2 THEN DD = 1
10 D$ =  CHR$ (4):G$ =  CHR$ (7):Q$ =  C
       HR$ (17):ZERO$ =  CHR$ (0)
20  HGR : HCOLOR= 3
30  FOR I = 0 TO 80 STEP 10
35  HPLOT I,I TO I,190 - I TO 278 - I,19
       0 - I TO 278 - I,I TO I,I
40  NEXT
50  FOR I =  - 10 TO 10: HPLOT 139 + I,8
       5 TO 139 + I,105: NEXT
60  HGR2 : HCOLOR= 3: HPLOT 0,0 TO 0,190
       TO 278,190 TO 278,0 TO 0,0
70  FOR I = 0 TO 80 STEP 10
75  HPLOT 139,I TO I,95 TO 139,190 - I T
       O 278 - I,95 TO 139,I
80  NEXT
85  FOR I =  - 10 TO 10: HPLOT 139 + I,8
       5 TO 139 + I,105: NEXT
90  GOTO 5000
200  REM PERMUT. PAGES
210  POKE  - 16304,0
215  IF PAGE = 1 THEN  POKE  - 16300,0
220  IF PAGE = 2 THEN  POKE  - 16299,0
250  RETURN
5000  REM MENU PRINCIPAL
5010  TEXT : HOME
5020  VTAB 1: PRINT "1(ERE PAGE) - 2(EME
       PAGE) - 3(LES DEUX) REPONDEZ PAR
       1,2 OU 3 S.V.P. ";
5030  GET Z$
5040 P =  VAL (Z$): IF P < 1 OR P > 3 TH
       EN  PRINT G$: GOTO 5010
5050  PRINT
5100  VTAB 4: PRINT "1 (NORMAL) - 2 (INV
       ERSE)                  REPONDEZ PAR
       1 OU 2 S.V.P. ";
5110  GET Z$
5120 I =  VAL (Z$): IF I < 1 OR I > 2 TH
       EN  PRINT G$: GOTO 5100
5130 I = (I - 1) * 32
5140  PRINT
5200  IF P < 3 THEN 5300
5210  VTAB 7: PRINT "1 (COTE A COTE) - 2
       (INTERSECTION) -    3 (UNION) -
       4 (DIFFERENCE)              REPON
       DEZ PAR 1, 2, 3 OU 4 S.V.P. ";
5220  GET Z$
5230 L =  VAL (Z$): IF L < 1 OR L > 4 TH
       EN  PRINT G$: GOTO 5210
5240 L = (L - 1) * 4: IF L = 12 THEN L =
       16
5250  PRINT
5300  IF P = 3 AND L = 0 THEN 5400
5310  VTAB 11: PRINT "1 (FORMAT NORMAL)
       - 2 (FORMAT DOUBLE)   REPONDEZ PA
       R 1 OU 2 S.V.P. ";
5320  GET Z$
5330 D =  VAL (Z$): IF D < 1 OR D > 2 TH
       EN  PRINT G$: GOTO 5310
5340 D = (D - 1) * 64
5350  PRINT
5400 COEFF = P + L + I + D
5500  VTAB 15
5510  IF P < 3 THEN  PRINT "LE DESSIN ES
       T-IL DEJA EN MEMOIRE?"
5512  IF P = 3 THEN  PRINT "LES DESSINS
       SONT-ILS DEJA EN MEMOIRE?"
```

```
5515  PRINT "REPONDEZ PAR  O  OU  N  S.V
       .P. ";
5520  GET Z$
5530  IF Z$ <  > "O" AND Z$ <  > "N" THE
       N  PRINT G$: GOTO 5500
5540  PRINT
5550  IF Z$ = "O" THEN 5600
5560  GOSUB 10000
5570  PRINT D$"BLOAD";ZZ$;",A$2000"
5575  IF P < 3 THEN 5600
5580  GOSUB 10000
5590  PRINT D$"BLOAD";ZZ$;",A$4000"
5600  HOME : PRINT "VOUS DISPOSEZ MAINTE
       NANT DES COMMANDES  SUIVANTES: "
5610  PRINT : PRINT "1  OU  2  POUR FAIR
       E APPARAITRE SUR L'              EC
       RAN UNE DES 2 PAGES"
5620  PRINT : PRINT "'ESCAPE'  POUR PASS
       ER DU MODE GRAPHIQUE             AU
        MODE TEXTE (MENU)"
5630  PRINT : PRINT "'RETURN'  POUR LANC
       ER L'IMPRESSION"
5640  PRINT : PRINT "0             POUR CHAN
       GER EVENTUELLEMENT              LE
        (OU LES) GRAPHE(S)"
5650  PRINT : PRINT : PRINT "VOTRE CHOIX
        ? ";: GET Z$
5660  HOME
5700  PRINT
```



```
5730  IF  ASC (Z$) = 13 THEN 6000
5740  IF  ASC (Z$) = 27 THEN  TEXT : HOM
       E : GOTO 5600
5750 PAGE =  VAL (Z$): IF PAGE = 1 OR PA
       GE = 2 THEN  GOSUB 200: GOTO 5600
5760  IF PAGE = 0 THEN 5560
5770  PRINT G$: GOTO 5600
6000  REM COMMANDE IMPRIMANTE
6010  PRINT D$"PR#1"
6012  PRINT ZERO$
6015  POKE 1913,COEFF
6020  PRINT Q$
6030  PRINT D$"PR#0"
6050  TEXT : HOME : PRINT "VOULEZ-VOUS U
       NE AUTRE IMPRESSION ? "
6060  PRINT : PRINT "(REPONSE PAR  O  OU
         N  S.V.P.) ";
6070  GET Z$
6080  IF Z$ = "N" THEN  VTAB 10: HTAB 10
       : PRINT "AU REVOIR": END
6090  IF Z$ = "O" THEN 5000
6100  PRINT G$: GOTO 6050
10000  REM CATALOG
10040  HOME : PRINT D$"CATALOG,D";DD
10050  VTAB 1: PRINT "FLECHES ->  <- POU
       R DEPLACER LE CURSEUR JUSQU'AU NO
       M VOULU PUIS 'RETURN'          (OU
       'RETURN' SI LE NOM N'EST PAS AFFI
       CHE"
10055 VV = 3:HH = 7: VTAB VV: HTAB HH
10060  GET Z$:Z =  ASC (Z$)
10065  IF Z = 13 AND VV < 4 THEN  HOME :
        PRINT "VOUS POUVEZ MAINTENANT CH
       ANGER DE DISQUE": GOSUB 30000: GO
       TO 10000
10070  IF Z = 8 THEN VV = VV - 1: IF VV
       < 3 THEN VV = 3
10080  IF Z = 21 THEN VV = VV + 1: IF VV
        > 23 THEN VV = 23
10090  IF Z = 13 THEN 10400
10095  VTAB VV: HTAB HH
10100  IF Z <  > 13 AND Z <  > 8 AND Z <
         > 21 THEN  PRINT G$
10150  VTAB VV: HTAB HH
10200  GOTO 10060
10400  VTAB 1: HTAB 1: PRINT "AVANCER AV
       EC LA FLECHE -> JUSQU'A DEPAS-SER
        LE NOM
                                    "
10410  INVERSE : VTAB 3: PRINT "  PUIS 'R
       ETURN' ";: NORMAL : PRINT "
                                    "
10420  VTAB VV: HTAB HH
10500  INPUT ZZ$
10600  RETURN
30000  REM ATTENTE
30010  POKE  - 16368,0
30030  VTAB 24: HTAB 5: INVERSE : PRINT
       "BARRE D'ESPACEMENT POUR LA SUITE
       ";
30040  NORMAL
30050  IF  PEEK ( - 16384) <  = 127 THEN
        30050
30060  POKE  - 16368,0
30070  PRINT
30100  RETURN
```

Graphiques et logique

Olivier Herz

Le programme GRAPHLOG est une illustration de l'article de Guy Mathieu : c'est un petit programme assembleur très simple, relogeable (c'est la mode dans ce numéro de Pom's !) écrit avec l'assembleur LISA 2.5. Il réalise une opération logique bit par bit sur deux pages graphiques et met le résultat dans une troisième. Par page graphique, nous entendons ici les intervalles $2000.3FFF, $4000.5FFF et $6000.7FFF, étant entendu que seuls les deux premiers intervalles correspondent à des pages graphiques vidéo ; pour voir la dernière, il faudra faire 2000 < 6000.7FFFM ou 4000 < 6000.7FFFM en mode moniteur. L'opération logique correspond au type graphique de Pascal, repris aussi par HAIFA dans Pom's 5. Si A et B désignent les points des pages sur lesquelles se réalise l'opération et C le résultat à mettre dans une troisième page (0 = éteint, 1 = allumé), on obtient :

TYPE	VALEUR DE C
0	0
1	A NOR B
2	A AND NOT B
3	NOT B
4	NOT A AND B
5	NOT A
6	A EOR B
7	A NAND B
8	A AND B
9	A = B
10	A
11	A OR NOT B
12	B
13	NOT A OR B
14	A OR B
15	1

Ce TYPE s'applique aux bits des points de l'écran. En ce qui concerne les bits de couleur (le 8e bit des octets de l'écran), on applique un autre type (TYPCOL). Pour se servir de la routine, il suffit de POKEr aux adresses 7 à 9 les numéros des pages graphiques (à savoir 32, 64 ou 96 : il est très dangereux de POKEr autre chose) et en 24 et 25 TYPE et TYPCOL respectivement, puis de faire un CALL à l'adresse où l'on a chargé cette routine relogeable.

En effectuant une copie d'écran graphique à partir des écrans ainsi engendrés, il vous est maintenant possible de faire tout ce que Guy Mathieu proposait aux possesseurs d'Epson.

Exemples :

7	8	9	24	25	
32	32	32	5	5	..... négatif de la page HGR, 8e bits compris
32	64	64	8	0	..... AND des pages HGR et HGR2 avec résultat en HGR2 et 8e bits mis à 0
64	32	96	14	6	..... OR de HGR2 et HGR : résultat dans la page "invisible" avec EOR des 8e bits

Graphlog — Lisa 2.5

```
 1          LST
 2 ;
 3 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 4 ;X                             X
 5 ;X CE PROGRAMME REALISE UNE    X
 6 ;X OPERATION LOGIQUE SUR LES   X
 7 ;X OCTETS DES DEUX PAGES       X
 8 ;X GRAPHIQUES ET MET LE RE-    X
 9 ;X SULTAT DANS UNE 3EME.       X
10 ;X                             X
11 ;X (C) O. HERZ POUR POM'S 7    X
12 ;X                             X
13 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
14 ;
15 LIGNE    EPZ $6        ;LIGNE DE L'ECRAN
16 PAGE1    EPZ $7        ;PREMIERE PAGE
17 PAGE2    EPZ $8        ;DEUXIEME PAGE
18 PAGE3    EPZ $9        ;PAGE DU RESULTAT
19 TYPE     EPZ $18       ;TYPE POUR LES POINTS
20 TYPCOL   EPZ $19       ;TYPE POUR LE BIT DE COULEUR
21 BASE     EPZ $1E       ;ADRESSE DE LA PREMIERE LIGNE
22 MEM1     EPZ $FD
23 MEM2     EPZ $FE
24 MEM3     EPZ $FF
25 ;
26          ORG $800      ;PROGRAMME RELOGEABLE
27          OBJ $800
28 ;
29          LDA #191
30          STA LIGNE
31 BCLELIG  PHA           ;LIGNES 31 A 47:
32          AND #$C0      ;-VOIR APPLESOFT
33          STA BASE      ;-ROUTINE EN $F417
34          LSR
35          LSR
36          ORA BASE
37          STA BASE
38          PLA
39          STA BASE+1
40          ASL
41          ASL
42          ASL
43          ROL BASE+1
44          ASL
45          ROL BASE+1
46          ASL
```

```
47          ROR BASE
48          LDY #39
49 BCLECOL  LDA BASE+1    ;PREPARATION DES ADRESSES
50          AND #$1F
51          ORA PAGE1
52          STA BASE+1
53          LDA (BASE),Y
54          STA MEM1
55          LDA BASE+1
56          AND #$1F
57          ORA PAGE2
58          STA BASE+1
59          LDA (BASE),Y
60          STA MEM2
61          LDX #8        ;LIGNES 61 A 81:
62 BCLETYP  LDA #1        ;-VOIR HAIFA
63          ASL MEM1      ;-ROUTINE DOPRINT
64          BCC >0
65          ASL
66          ASL
67 ^0       ASL MEM2
68          BCC >1
69          ASL
70 ^1       CPX #8
71          BNE >2
72          AND TYPCOL
73          CLC
74          BCC >3
75 ^2       AND TYPE
76          CLC
77 ^3       BEQ >4
78          SEC
79 ^4       ROL MEM3
80          DEX
81          BNE BCLETYP
82          LDA BASE+1
83          AND #$1F
84          ORA PAGE3
85          STA BASE+1
86          LDA MEM3
87          STA (BASE),Y
88          DEY
89          BPL BCLECOL
90          DEC LIGNE
91          LDA LIGNE
92          CMP #$FF
93          BNE BCLELIG
94          RTS
95 ;
96          DCM "BSAVE GRAPHLOG,A$800,L$80"
97          END
```

Récapitulation

```
]800.871

0800- A9 BF 85 06 48 29 C0 85
0808- 1E 4A 4A 05 1E 85 1E 68
0810- 85 1F 0A 0A 0A 26 1F 0A
0818- 26 1F 0A 66 1E A0 27 A5
0820- 1F 29 1F 05 07 85 1F B1
0828- 1E 85 FD A5 1F 29 1F 05
0830- 08 85 1F B1 1E 85 FE A2
0838- 08 A9 01 06 FD 90 02 0A
0840- 0A 06 FE 90 01 0A E0 08
0848- D0 05 25 19 18 90 03 25
0850- 18 18 F0 01 38 26 FF CA
0858- D0 DF A5 1F 29 1F 05 09
0860- 85 1F A5 FF 91 1E 88 10
0868- B6 C6 06 A5 06 C9 FF D0
0870- 93 60
```

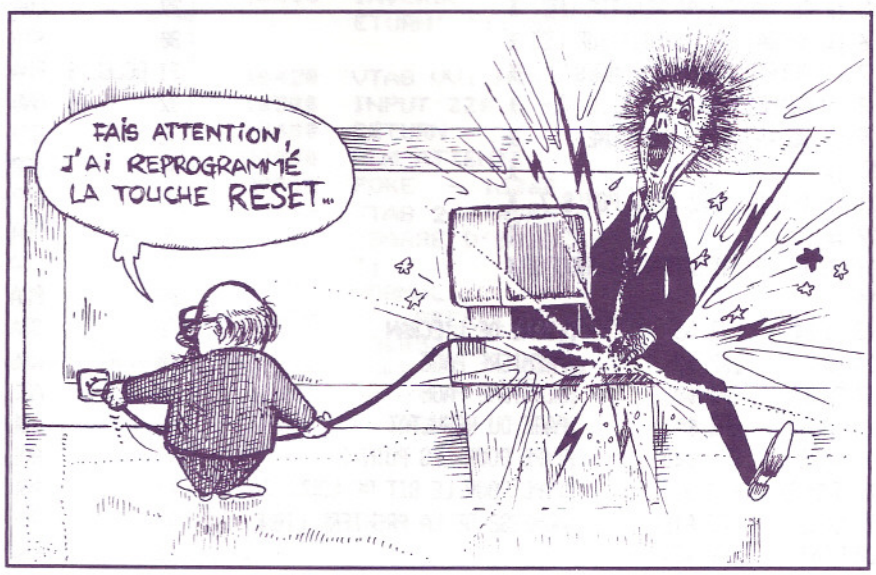
Hard Copy Seikosha

Olivier Herz

Afin de compléter l'article de Guy Mathieu et celui sur « Graphiques et logique », voici un petit programme en langage machine (écrit à mes débuts sur Apple) permettant d'obtenir la copie d'écran graphique sur une imprimante Seikosha.

Ne possédant pas d'assembleur à l'époque, j'avais écrit ce programme directement en langage machine. Il reste toutefois intéressant à publier car il est assez rapide et, d'autre part, les possesseurs de Seikosha n'ont certainement pas tous un tel programme à leur disposition.

La seule routine ROM utilisée est le WAIT $FCA8 ; le programme peut être utilisé par n'importe quel Apple.



Hard Copy Seikosha

```
]LIST

5  REM  ***  HARD COPY SEIKOSHA  ***
6  REM  ***        COPYLEFT       ***
7  REM  *** OLIVIER HERZ - POM'S ***
10  TEXT : HOME : INVERSE : VTAB 6: PRIN
    T "IMPRESSION DES 2 PAGES HAUTE R
    ESOLUTION": NORMAL
20  PRINT : PRINT : PRINT : PRINT "VOULE
    Z-VOUS LA PAGE 1 OU 2 ? ";
30  GET A$: IF A$ <  > "1" AND A$ <  > "
    2" THEN 30
40  PRINT A$: POKE 28,208: POKE 29,31 +
    32 * ( ASC (A$) - 48)
50  PRINT : PRINT "VOULEZ-VOUS LA PAGE E
    NTIERE ? ";
60  GET A$: IF A$ <  > "O" AND A$ <  > "
    N" THEN 60
70  PRINT A$: POKE 24,0: IF A$ = "N" THE
    N  POKE 24,4
80  PRINT : PRINT "VOULEZ-VOUS L'IMPRESS
    ION EN NEGATIF ? ";
90  GET A$: IF A$ <  > "O" AND A$ <  > "
    N" THEN 90
100  PRINT A$: POKE 25,0: IF A$ = "O" TH
    EN  POKE 25,1
110  PRINT : PRINT "VOULEZ-VOUS UN CADRE
     ? ";
120  GET A$: IF A$ <  > "O" AND A$ <  >
    "N" THEN 120
130  PRINT A$: POKE 26,0: IF A$ = "O" TH
    EN  POKE 26,1
140  PRINT : PRINT "VOULEZ-VOUS LE FORMA
    T DOUBLE ? ";
150  GET A$: IF A$ <  > "O" AND A$ <  >
    "N" THEN 150
160  PRINT A$: IF A$ = "O" THEN 200
170  POKE 27,0: POKE 235,192: POKE 236,1
    29: IF  PEEK (24) = 0 THEN 190
180  POKE 237,162: POKE 30,159: GOTO 230
190  POKE 237,194: POKE 30,143: GOTO 230
200  POKE 27,1: POKE 235,224: POKE 236,1
    31: IF  PEEK (24) = 0 THEN 220
210  POKE 237,69: POKE 30,78: GOTO 230
220  POKE 237,133: POKE 30,46
230  PRINT : PRINT : HTAB 3: INVERSE : P
    RINT " APPUYEZ SUR UNE TOUCHE QUE
    LCONQUE ": NORMAL : WAIT  - 16384
    ,128: POKE  - 16368,0
240  IF  PEEK (6148) <  > 1 THEN  PRINT
     CHR$ (4);"BLOADSEIKO.OBJ": POKE
    6148,1
250  POKE  - 16240,8: HOME : CALL 6149
260  CALL  - 1052: CALL  - 1052: CALL  -
     1052
```

SEIKO.OBJ
Code hexadécimal

```
*1800.19DA

1800- EA EA EA EA EA A5 1A F0
1808- 05 A4 EB 20 50 19 A9 00
1810- 85 06 A5 1B 85 0A A5 1E
1818- 85 FB A9 00 85 FC 85 0B
1820- 85 0C A5 1A F0 03 20 90
1828- 19 A9 00 85 07 A9 00 A4
1830- 07 D0 02 A5 18 85 08 A2
1838- 00 20 20 19 A9 00 A4 07
1840- F0 08 18 69 28 88 F0 02
1848- 69 28 85 FD A5 08 4A 85
1850- FE A9 00 6A 18 65 FD 85
1858- FD 8A 0A 0A 18 65 FE 85
1860- FE 38 A5 1C E5 FD 85 F9
1868- A5 1D E5 FE 85 FA A4 06
1870- B1 F9 A4 1B F0 3F A4 0A
1878- 85 FF F0 1D A9 00 46 FF
1880- 90 02 69 02 46 FF 90 02
1888- 69 0B 46 FF 90 02 69 2F
1890- 46 FF 90 02 69 3F 4C B5
1898- 18 A9 00 06 FF 06 FF 90
18A0- 02 69 5F 06 FF 90 02 69
18A8- 17 06 FF 90 02 69 05 06
18B0- FF 90 02 69 00 A4 19 D0
18B8- 02 49 FF 09 80 C9 80 F0
18C0- 07 A8 20 92 19 4C CC 18
18C8- A9 00 85 0B E8 E0 08 F0
18D0- 03 4C 39 18 E6 08 A5 08
18D8- C9 08 F0 03 4C 37 18 E6
18E0- 07 A5 07 C9 03 F0 03 4C
18E8- 2D 18 A5 1A F0 06 20 20
18F0- 19 20 90 19 A9 0A 20 47
18F8- 19 20 31 19 A5 0A F0 05
1900- C6 0A 4C 16 18 E6 06 A5
1908- 06 C9 28 F0 03 4C 12 18
1910- A5 1A F0 05 A4 EC 20 50
1918- 19 60 00 00 00 00 00 00
1920- E6 FB D0 02 E6 FC A5 1B
1928- F0 06 E6 FB D0 02 E6 FC
1930- 60 A5 FC 4A A5 FB 6A 4A
1938- 4A 18 69 0A A8 EA A9 8A
1940- 20 A8 FC 88 D0 F8 60 8D
1948- 90 C0 A9 07 20 A8 FC 60
1950- A9 1B 20 47 19 A9 10 20
1958- 47 19 A9 00 20 47 19 A5
1960- 1E 20 47 19 A9 1C 20 47
1968- 19 A5 1B F0 0E A9 FF 20
1970- 47 19 98 20 47 19 A9 1C
1978- 20 47 19 A5 ED 20 47 19
1980- 98 20 47 19 A9 0A 20 47
1988- 19 A0 40 20 3E 19 60 00
1990- A0 FF A5 0B D0 20 A9 01
1998- 85 0B A9 1B 20 47 19 E6
19A0- 0C A9 10 20 47 19 E6 0C
19A8- A5 FC 20 47 19 E6 0C A5
19B0- FB 20 47 19 E6 0C 98 20
19B8- 47 19 E6 0C A5 1B F0 06
19C0- 98 20 47 19 E6 0C A5 0C
19C8- C9 55 90 0E A9 14 20 47
19D0- 19 A9 00 85 0B 85 0C 20
19D8- 31 19 60
```

La souris de Lisa

Hervé Thiriez

Faut-il encore parler de Lisa ? Nous avons été abreuvés dans les journaux d'articles sur Lisa depuis sa présentation internationale le 19 janvier dernier. D'autre part, nous ne pouvons, dans une revue telle que Pom's, passer sous silence la naissance de Lisa. Voyons donc pourquoi la souris de Lisa veut devenir aussi délèbre que le sourire de Mona...

La première impression que dégage cet ordinateur de bureau — le terme convient ici parfaitement — est celle d'un équipement au design agréable, d'apparence professionnelle. Par opposition, l'Apple II Plus, qui fait notre bonheur depuis si longtemps, suggère un peu la 2CV, avec son aspect rustique et son côté bohème.

Lisa est un système à clavier détaché, incorporant deux lecteurs de disquettes 5 pouces (860K octets par disquette) et un écran à très haute résolution (720 × 364, soit près de cinq fois autant de points que pour l'Apple II). Une surprise désagréable, la seule d'ailleurs : il n'est pas question de travailler en couleur ; tout le monde sait que les souris ne connaissent que le blanc et le gris...

Seul appendice externe à l'appareil, relié par un fil gris lui aussi, la fameuse souris, qui est le mode de communication privilégié de l'utilisateur vers la machine. C'est grâce à cette souris et à la qualité graphique de l'écran qu'il suffit d'une demi-heure à un non-informaticien pour apprendre à utiliser Lisa.

La souris est un petit boîtier que l'on fait glisser sur son bureau dans n'importe quelle direction. Un curseur sur l'écran est associé à la souris et se déplace en totale symbiose avec elle. L'écran présente un certain nombre de petits dessins illustrant les dossiers sur lesquels on travaille, y compris une poubelle pour ceux dont on veut se débarrasser.

Il suffit de pointer le curseur associé à la souris sur le dessin correspondant au dossier que l'on désire « ouvrir », et de confirmer son choix en appuyant sur la petite barre de la souris. Le dossier s'ouvre alors, et on y choisit (toujours grâce à la souris) tel ou tel élément.

Différents dossiers peuvent être ouverts simultanément, comme sur un bureau normal. On peut agrandir ou réduire les documents sur l'écran à volonté, ouvrir ou fermer tout dossier, passer du traitement de texte au Lisacalc ou au programme de réalisation de graphiques. Tout cela se fait très rapidement, et l'on peut ainsi croiser les applications : éditer en traitement de texte les titres de graphiques provenant de données Lisacalc !

A tout moment, une copie de l'écran peut être obtenue, l'écran incorporant à loisir des textes (en plusieurs polices et tailles) et des graphiques.

Le souci de simuler la gestion d'un bureau va très loin : on peut ainsi faire apparaître sur l'écran une calculatrice avec des touches sur lesquelles on « appuie » grâce à la souris.

Avec Lisa sont fournis en standard :

- LisaWrite : traitement de texte (tailles et polices variées).
- Lisacalc : 255 lignes, et 255 colonnes de largeur variable.
- Lisalist, base de données pouvant contenir 6000 enregistrements de 100 caractères chacun.
- LisaGraph, soit Visiplot en beaucoup plus rapide, et avec les possibilités d'édition de texte.
- LisaDraw, outil performant pour la création graphique.
- LisaProject, pour la gestion de projets (PERT, Gantt) jusqu'à 1000 tâches.

Tout cela coexiste sans effort dans le million d'octets de mémoire centrale, le microprocesseur MC68000 assurant une rapidité de traitement inégalée en micro-informatique.

Seul petit problème : tout le monde n'a pas le budget de 120 000 francs TTC nécessaire pour avoir le système complet, avec son imprimante et le disque dur de 5 méga-octets. En outre, ceux qui désirent s'offrir Lisa devront encore patienter jusqu'en septembre. Le prix n'est pas donné, mais le rapport performance/prix est excellent en comparaison avec celui des mini-ordinateurs de bas de gamme.

Nous attendons aussi le discret McIntosh qui, à en croire ce que j'ai lu dans des revues professionnelles, devrait proposer à un prix sensiblement plus abordable les avantages logiciels de Lisa et de sa souris, avec une puissance et une capacité réduites.

Dès que nous aurons eu l'occasion de travailler réellement avec Lisa ou McIntosh, nous vous ferons partager nos impressions plus en détail.

L'Apple *IIe* à l'essai

Jean-François Duvivier

L'Apple *IIe* (e pour «enhanced», c'est-à-dire amélioré) a été annoncé par Apple le 19 janvier dernier. Destiné à remplacer le modèle Apple II + bien connu, il commence à se répandre dans les boutiques micro-informatiques. Grâce à l'amabilité de la société I.E.F., j'ai pu en avoir un exemplaire pendant une soirée, de quoi l'examiner sous toutes ses coutures.

Première constatation : si le nom change, la ligne demeure : le boîtier est celui que nous connaissons tous. Seul le nom a changé pour s'adapter au nouveau logo. Cependant, pour l'Apple-o-mane averti, pas de risque de se tromper pour autant : le *IIe* dispose d'un clavier plus complet qui se remarque dès le premier abord.

Un clavier plus complet

Outre ses 11 touches de plus (63 contre 52), le clavier du *IIe* présente de nombreuses différences avec celui du II. Nous en commentons ci-dessous les plus notables.

La touche RESET est mise à l'écart du reste du clavier afin d'éviter les fausses manipulations. De plus, le RESET n'est pris en compte que lorsque la touche CTRL est enfoncée (il n'y a plus la possibilité de choix offerte sur le II entre RESET tout seul et la double action CTRL-RESET).

Les minuscules sont disponibles au clavier et affichées comme telles à l'écran. En fait, tous les codes Ascii sont générables au clavier.

Une touche «Shift-lock» permet le blocage du clavier en majuscules. C'était nécessaire pour la programmation en Basic (l'Applesoft, comme le DOS, n'accepte que les majuscules). Un bon point : le blocage majuscule/minuscule n'affecte que les touches alphabétiques. Appuyer sur «1» donne toujours le «1», quelle que soit la position de la touche Shift-lock.

Quatre touches de gestion du curseur sont rassemblées en bas à droite du clavier. On y retrouve bien sûr les classiques flèches à droite et à gauche, mais également les flèches vers le haut et vers le bas.

Toutes les touches qui génèrent un code sont à répétition automatique. Le temps entre le premier caractère et le deuxième est relativement long, ce qui évitera, je pense, toute fausse manœuvre. En raison de cette répétition automatique, il n'y a plus de touche REPT (répétition).

Deux touches marquées l'une d'une petite pomme vide (pomme «ouverte»), l'autre d'une pomme pleine (pomme «fermée») sont placées de part et d'autre de la barre d'espacement. Elles ne génèrent aucun code (elles sont en fait équivalentes aux boutons sur les poignées de jeux), mais ont quelques fonctions particulières quand elles sont enfoncées en même temps que CTRL-RESET :

- la pomme ouverte force l'Apple à redémarrer (et donc à rebooter s'il y a un contrôleur) quelle que soit la configuration. Cela devrait permettre d'épargner considérablement le bouton de l'alimentation qui, l'expérience le montre, supporte très mal les opérations de «marche/arrêt» répétitives.
- la pomme fermée provoque l'exécution d'un autotest.

Parmi les nouvelles touches, on trouve également une touche de tabulation «TAB» et une touche «DELETE» qui pourront être utilisées par les logiciels prévus pour le *IIe*.

Le clavier est utilisable en Azerty ou en Qwerty. La sélection est faite par un bouton poussoir judicieusement placé sous la partie inclinée de l'Apple. Ce bouton commande également le changement de caractères à l'écran. Les caractères français sont donc disponibles aussi bien au clavier qu'à l'écran. Malheureusement, la disponibilité d'un double clavier se solde par un double marquage des touches, qui vont donc jusqu'à abriter 4 signes différents sur leur cabochon ! En langage non informatique, cela s'appelle un casse-tête chinois, et taper plus de dix touches sans erreur relève de l'exploit. Il ne semble pas qu'il existe pour le moment la possibilité d'un marquage unique (Qwerty par exemple).

Comme pour le II, un connecteur est toujours prévu pour le clavier numérique séparé.

Certaines publicités ont fait état de touches de fonction programmables. La seule documentation que j'avais en ma possession (Manuel de Référence *IIe*) n'y faisait pas allusion. De même, le programme de démonstration et d'introduction au clavier ne mentionne aucune possibilité de touches de fonctions.

Outre le problème de double notation des touches, j'ai un deuxième regret concernant le clavier : l'absence de buffer. La refonte du clavier était l'occasion rêvée d'introduire un buffer de caractères tel qu'on le trouve sur l'Apple *III* et même sur le II (avec la carte Keyboard Enhancer de Videx par exemple). Le buffer est comme toutes les bonnes choses : une fois qu'on y a goûté, on ne peut plus s'en passer !

Par rapport à l'Apple II, ces connecteurs ont les mêmes signaux disponibles sur leurs broches, ce qui signifie que la majorité des cartes d'extension existant sur le II sont compatibles, notamment les contrôleurs de lecteurs de disquettes, les interfaces imprimantes et la Softcard Z80. Les exceptions sont les cartes qui devaient obligatoirement se mettre dans le slot 0 (ROMcard Applesoft ou Integer par exemple) et toutes les cartes nécessitant d'ôter un circuit de l'Apple pour connecter un câble à sa place (cas de certaines cartes RAM 16,32,64 et 128K ou de cartes 80 colonnes).

La nouveauté vient d'un connecteur supplémentaire situé en plein milieu de la carte et dénommé «Connecteur Auxiliaire» (quelle imagination !). Ce nouveau connecteur comprend 60 contacts (contre 50 aux connecteurs de slot). Son emplacement (dans le prolongement du slot 3) interdit son usage simultané avec le slot 3. Il est prévu spécialement pour les extensions mémoire et pour l'extension 80 colonnes (de nombreux logiciels imposent l'utilisation du slot 3 pour les cartes 80 colonnes pour Apple II, ce qui explique la position du connecteur auxiliaire).

Trois cartes actuellement disponibles en France utilisent ce connecteur :

- une carte 80 colonnes ;
- une carte 80 colonnes doublée d'une extension mémoire de 64K ;
- une carte RVB Chat Mauve permettant l'obtention des couleurs sur un poste téléviseur SECAM, ainsi que l'extension 64K et 80 colonnes.

Le microprocesseur 6502 ne pouvant pas adresser plus de 64K, l'extension de 64K n'est pas une extension de mémoire centrale à proprement parler. Mais elle est comparable aux cartes mémoires disponibles pour le II (Saturn, Ramex, Legend...). Comme elle est proposée par Apple et constitue de ce fait un standard, de nombreux logiciels vont pouvoir en faire usage.

Selon certains renseignements que je n'ai malheureusement pas pu vérifier, l'adjonction de l'extension 64K permettrait de plus de disposer d'un graphisme haute résolution plus complet : 560 × 192.

Sous le capot

La carte mère a été totalement refaite : légèrement plus petite, elle s'arrête avant le clavier, ce qui permet d'accéder à tous les circuits sans démonter autre chose que le capot. Le nombre de circuits intégrés est passé d'environ 90 à 35. Cette réduction permet d'espérer bien entendu une fiabilité accrue. Elle s'accompagne également d'une réduction de la consommation de la carte mère. L'alimentation étant restée la même, on dispose ainsi de plus de puissance sur les slots.

Le gain de circuits est essentiellement dû à l'intégration de plus en plus poussée des composants utilisés. Par exemple, les mémoires vives RAM utilisées sont des 64 Kbits au lieu de 16, ce qui permet de faire 64 Koctets avec seulement huit circuits. De même, les mémoires mortes utilisées sont des 32 Kbits au lieu de 16. Enfin, deux circuits à 40 pattes trônent en plein milieu de la carte. Dénommés MMU (Memory Management Unit) et IOU (Input Output Unit), ils ont été réalisés spécialement pour Apple (il coulera de nombreux bits dans votre Apple avant que vous ne puissiez acheter un *IIe* made in Taiwan !!!)

L'utilisation de mémoires reprogrammables pour le codage clavier et pour la génération vidéo des caractères permet une très grande souplesse. Chaque pays a sa version nationale du *IIe*, disposant en plus du Qwerty, d'un clavier équipé des caractères spécifiques à la langue, et de leur génération à l'écran.

En ce qui concerne la génération video, il faut préciser que les *IIe* vendus en France sont équipés en standard d'un modulateur PAL sélectable par switch (sur la carte mère). Les possesseurs de postes de télévision bi— ou tristandards pourront donc profiter directement des couleurs, sans aucune carte supplémentaire. Les autres devront se contenter du noir et blanc ou attendre l'arrivée imminente d'une carte « Chat Mauvre » adaptée au *IIe* (pour vous consoler, sachez que le standard SECAM est techniquement le meilleur...).

Trucs et astuces

De l'utilité du symbole « : »

] : READ permet de lire les DATA d'un programme (arrêté par CTRL-C, par exemple), alors que] READ donne le message NOT DIRECT COMMAND de la commande READ du DOS.

]:IN#0 : PR#0 permet de déconneter de DOS. Pour le reconnecter : CALL 1002 ou *3D0G ou RESET.

Parfois,] PR#6 ne marche plus pour rebooter (quand par exemple la commande PR # du DOS a été endommagée). Pour s'en sortir sans devoir éteindre l'appareil, faire] : PR#6 ou *6CTRL-P.

La raison de tout cela tient au fait que le DOS intercepte les commandes tapées en début de ligne au clavier, avant de redonner le contrôle à l'Applesoft (ou au moniteur). D'où des problèmes avec les commandes synonymes entre le DOS et Applesoft : PR#, IN # et READ.

Ainsi, si l'on entre en mode direct A=3 : CATALOG, l'instruction CATALOG du DOS n'est pas reconnue. C'est comme cela que]: PR#6 active le « PR# » de l'Applesoft (qui marche toujours, s'il est résident) et non celui du DOS, qui peut avoir été écrasé.

Du côté des slots...

Sept connecteurs (slots) numérotés de 1 à 7 sont disponibles. Il est à noter qu'il n'y a donc plus de slot 0. L'explication en est simple : étant donné la présence sur la carte mère de 64 K de mémoire vive, le slot 0 est considéré comme étant occupé par une carte langage 16 K.

Petit détail intéressant : la présence à côté des slots d'un témoin de tension (plus d'excuses pour les cartes mises ou enlevées avec l'Apple en marche !). On note également le blindage complet de l'arrière de l'Apple, désormais en métal et solidaire de la plaque du fond. Les grandes échancrures ont laissé la place à de multiples ouvertures prévues pour la fixation de connecteurs. Ceci devrait permettre de ne plus avoir à démonter l'Apple pour brancher le câble de l'imprimante ou de toute autre carte interface. Autre amélioration pour les fanas de jeux : un connecteur situé à côté de l'entrée cassette permet de brancher les manettes de jeux sans avoir à démonter le capot et surtout sans devoir se munir d'une loupe pour effectuer le branchement !

Les logiciels

Quant à son utilisation, le *IIe* se comporte comme un II équipé d'une carte langage. Le langage résident est le Basic Applesoft qui n'a pas bougé d'un iota. Les disquettes sont toujours au format DOS 3.3. En fait, il semble que les seules modifications « soft » effectuées concernent le moniteur (Prom F8), qui a été entièrement réécrit. Toutes les routines de gestion écran ont été modifiées pour prendre en compte la présence éventuelle d'une carte 80 colonnes. De même, les routines concernant le clavier autorisent l'utilisation des minuscules. Enfin, un autotest a été ajouté (activé par les touches CTRL-RESET-pomme fermée). Comme ces modifications ont nécessité plus de place mémoire, l'espace $C100-$C7FF auparavant dédié à l'adressage des slots est maintenant sélectable entre les slots et une Prom.

Malgré toutes ces modifications, les principaux points d'entrée du moniteur ont été gardés aux mêmes adresses. Vous pourrez toujours, par exemple, passer en mode moniteur en tapant CALL-151. Ainsi, sur 300 programmes commerciaux testés sur Apple *IIe*, j'ai lu que 95 %

avaient tourné du premier coup et 3 % après quelques légères modifications.

Pour ma part, tous les logiciels que j'ai essayés ont parfaitement tourné, ce qui est rassurant du point de vue compatibilité.

Et le portefeuille...

De même que pour l'Apple II ces derniers mois, le *IIe* sera essentiellement vendu sous forme d'une configuration comprenant un *IIe* 64K, un contrôleur, un drive et un moniteur à écran jaune. Les cartes d'extension spécifiques au *IIe* sont la carte Apple 80 col., la carte 80 col. + 64 K et la carte Chat Mauve. Il est à noter que, si cette dernière était disponibles partout, les deux autres étaient totalement inconnues à deux des cinq boutiques consultées. Les prix TTC que j'indique ci-dessous résultent d'un sondage effectué par téléphone auprès de cinq boutiques parisiennes :

- configuration : de 14 700 à 14 995 F
- Apple *IIe* seul : de 9 900 à 11 950 F
- carte 80 col. : de 900 à S999 F
- carte 80 + 64 K : de 2 300 à 2 400 F
- carte « Chat Mauve » : environ 3 000 F

On remarquera, en dehors du cas particulier de l'Apple *IIe* seul, que les prix varient très peu d'un revendeur à l'autre, résultat de l'uniformisation des prix imposée par Apple-Seedrin.

Conclusion

J'ai rencontré deux problèmes particuliers sur l'exemplaire qui m'a été prêté. D'une part, la commande PR # suivie d'un numéro correspondant à un slot vide sélecte la première carte trouvée dans un slot supérieur. C'est ainsi qu'un PR # 4 fait rebooter le disque, bien qu'il n'y ait aucune carte dans le slot 4 (le contrôleur étant en 6 !). D'autre part, il semble que l'Apple dont je disposais avait quelques difficultés à s'initialiser correctement après des coupures de tension brèves (inférieures à 2 secondes), ce qui m'arrivait lorsque j'utilisais l'interrupteur de l'alimentation pour faire rebooter l'Apple.

En dehors de ces problèmes mineurs, il faut souligner l'excellente compatibilité du *IIe* avec son prédécesseur le II+, tant au niveau du matériel (j'ai essayé sans problème toutes mes cartes interfaces, y compris une extension 128K Saturn) que du logiciel.

Cependant, je dois dire que cette première confrontation avec le *IIe* m'a laissé un sentiment de déception. Attendant avec impatience ce successeur de l'Apple II, je me suis retrouvé en face de la même machine, mais incapable de taper trois touches de suite ! Cette impression d'absence de nouveauté a été certainement accentuée par le fait que je dispose déjà des minuscules sur mon Apple II. D'autre part, je n'ai pas pu emprunter d'une carte extension 80 colonnes pour mon essai, ce qui ne m'a pas permis d'apprécier le confort apporté par les 80 colonnes.

Autre déception : les prix. Compte tenu d'un abaissement vraisemblable des coûts de production (carte plus petite, nombre de circuits en baisse...), on pouvait rêver à des prix inférieurs à ceux du II. Hélas, le *IIe* affiche un bon 2 000 F de plus. Comme, d'autre part, le II ne sera plus vendu, il s'ensuit une augmentation importante du système minimum qui atteint maintenant les 15 000 F. Si cette somme ne constitue pas un problème pour les utilisateurs professionnels (très majoritaires il est vrai), elle devient fort difficile à réunir, en cette période de crise, par un utilisateur individuel qui préfèrera alors se reporter sur des machines plus accessibles. L'évolution dans le domaine de la micro-informatique se traduit d'habitude par « plus puissant et moins cher ». Il semble que ce ne soit plus le cas désormais chez Apple. Dommage !

Ceci dit, l'Apple *IIe* a pour avantages sa compatibilité et sa fiabilité plus grandes. Si l'on pouvait avoir une version Azerty simple (16 cabochons à changer), ce serait beaucoup mieux !

Des programmes relogeables

Philippe François

Les programmes en langage machine dépendent en général de l'adresse à partir de laquelle ils ont été assemblés et ne peuvent de ce fait « fonctionner » que s'ils sont lancés à partir de cette adresse. Si nous voulons qu'un programme se chargeant en haut de la mémoire puisse fonctionner sans problème avec un Apple de 16, 32 ou 48 K, il nous faudrait normalement prévoir trois versions du même programme !

La solution évidente d'un tel programme consiste naturellement à écrire des programmes indépendants de leur origine, c'est-à-dire des programmes relogeables (relocatable programs), exécutables quelle que soit leur origine.

Les deux obstacles principaux à de telles pratiques sont les branchements inconditionnels (JMP, abréviation de « jump », l'équivalent en assembleur du GOTO) et les appels à des sous-programmes (JSR, « jump to subroutine », l'équivalent du GOSUB) comme le montre le petit programme ci-dessous :

```
1                JMP DÉBUT
2 DÉBUT          JSR RIEN
3                BRK
4 *
5 RIEN           RTS
6 *
```

Le problème posé par les JMP se résout aisément : il suffit de remplacer ces branchements inconditionnels (dont les adresses sont explicites) par des branchements conditionnels forcés. En effet, les branchements conditionnels du type BEQ, BNE,. étant des branchements relatifs, ils ne dépendent pas de l'adresse d'implantation du programme. On remplacera donc tous les JMP du programme par un des couples d'instructions suivant : CLC et BCC, ou bien SEC et BCS.

```
1                CLC
2                BCC DÉBUT
3 *
```

```
4 DÉBUT          JSR RIEN
5                BRK
6 *
7 RIEN           RTS
8 *
```

Un petit problème encore, à propos de ces branchements conditionnels forcés : les déplacements étant relatifs et s'exprimant sur sept bits (le bit huit indiquant le sens du déplacement, en amont ou en aval de la présente adresse) ne pourront être supérieurs à 126 octets dans un sens ou dans l'autre.

Toute tentative de dépassement de cette limite sera sanctionnée par le fatidique message OUT OF RANGE ERROR de l'assembleur.

La technique employée alors sera celle de l'ascenseur comme le montre l'exemple suivant :

```
          ORG $800
          CLC
          BCC ETAGE
          .
          . ; suite du programme
          .
          SEC
ÉTAGE     BCC DÉBUT
          .
          . ; suite du programme

DÉBUT     JSR RIEN
          BRK
*
RIEN      RTS
*
```

Le problème posé par les JSR se résout moins aisément. Néanmoins, l'examen attentif des programmes machines relogeables du commerce (AMPER MAGIC, pour n'en citer qu'un), ainsi que la lecture d'un petit article dans Apple Assembly Lines (Vol. 2, N° 10, juillet 1982), nous ont permis de comprendre la façon dont il pouvait se résoudre :

A la place de chaque JSR (nom du sous-programme), placez les trois lignes suivantes :

```
CLV
JSR $FF58
BVC (nom du sous-programme)
```

Mystérieux, n'est-ce pas ? Commentons donc cette séquence. L'octet placé à l'adresse $FF58 de la ROM Moniteur est toujours égal à $60, code de l'instruction RTS.

L'appel d'un sous-programme consistant seulement en un RTS peut paraître stupide, mais il a pour effet de placer les deux octets de la présente adresse sur la pile (stack), puis de les ressortir. Nous pourrons alors réutiliser ces deux octets dans le sous-programme concerné :

```
RIEN    TSX ; valeur du pointeur
            ; de pile
        DEX
        DEX
        TXS ; nouvelle valeur
```

Le sous-programme RIEN possède maintenant une adresse de retour sur la pile, exactement comme si nous l'avions appelé par un JSR RIEN normal ! Mais attention, ce n'est pas fini... En effet, si nous essayons de sortir du sous-programme RIEN par un RTS, nous allons nous retrouver à l'instruction BVC RIEN qui sera de nouveau exécutée et nous entraînera dans une boucle infinie ! Il n'y a malheureusement pas d'instruction 6502 du type SEV (« set overflow ») ; aussi allons-nous faire appel de nouveau à notre précieuse adresse $FF58 et terminer notre sous-programme RIEN par le couple d'instructions suivantes :

```
        BIT $FF58 ; place l'over-
                  ; flow
        RTS
```

L'instruction BIT (voir par exemple « Programmation du 6502 » par Rodnay Zaks, page 103) effectue un ET entre l'accumulateur A et la mémoire ; mais, surtout, les bits 6 et 7 de la mémoire sont transférés dans les bits V et N du registre d'état. La mémoire contenant $60 (soit 01100000 en binaire), le bit V du registre sera donc bien positionné à la valeur 1.

Résumons-nous par ce petit programme où sont utilisées toutes les techniques examinées dans cet article :

RELOGEABLE
Big Mac

```
                        1    *
                        2    RETURN   EQU  $FF58
                        3    COUT     EQU  $FDED
                        4    *
8000: 18                5             CLC  *        PSEUDO JMP DEBUT
8001: 90 00             6             BCC  DEBUT
                        7    *
8003: B8                8    DEBUT    CLV  *        EMPILE LA PRESENTE
8004: 20 58 FF          9             JSR  RETURN   ADRESSE
8007: 50 06             10            BVC  BELL     PSEUDO JSR BELL
8009: A9 C1             11            LDA  #"A"
800B: 20 ED FD          12            JSR  COUT
800E: 60                13            RTS
                        14   *
                        15   *
800F: BA                16   BELL     TSX  *        MISE A JOUR
8010: CA                17            DEX  *        DU POINTEUR
8011: CA                18            DEX  *        DE PILE
8012: 9A                19            TXS
8013: A9 87             20            LDA  #$87
8015: 20 ED FD          21            JSR  COUT
8018: 2C 58 FF          22            BIT  RETURN   ARME LE BIT N
                        23   *                      DU REGISTRE D'ETAT
801B: 60                24            RTS
```

Dump Pascal

Michel Marquis

Le programme ci-contre s'adresse à ceux qui souhaitent visualiser les informations stockées sur une disquette formatée en Pascal. Ces informations sont groupées en 280 blocs de 512 octets par disquette.

Description générale

Le programme édite sur l'imprimante un « dump » de la disquette tel que présenté en fin d'article. Chaque ligne indique le numéro d'un groupe de huit octets, puis chacun de ceux-ci en décimal (de 0 à 255), et enfin les caractères ASCII correspondant à ces huit octets.

Vous vous êtes probablement aperçus que le code ASCII est le même pour un octet X (de 0 à 127) et l'octet X + 128. Toutefois, dans le listing ASCII des octets dont la valeur dépasse 127, nous avons précédé ceux-ci du caractère exponentiation « ^ ».

D'autre part, les caractères dont les codes sont compris entre 0 et 32 inclus ne sont pas tous imprimables. C'est pourquoi nous les avons remplacés par un mnémonique relatif à leur rôle respectif (12 = LF, 27 = ESC, 7 = BEL,...). Ces mnémoniques sont imprimés après que leur signification ait été prélevée dans le tableau TABCTRL OF STRING [3]. Ce tableau est rempli en début de programme par la procédure LECTABCTRL à partir du fichier précédemment créé par le programme WRITECTRL sur lequel nous reviendrons ultérieurement.

Le programme

Lors de son exécution, le programme LECTBLOCK demande à l'utilisateur le numéro du canal associé à la disquette à analyser, puis celui du premier bloc, et enfin le nombre de blocs à « dumper ». Pour savoir à partir de quel bloc cette opération doit débuter, il suffit d'examiner le directory de la disquette avec le FILER en sélectionnant la commande E.

Dès cet instant, le « dump » sort sur l'imprimante et occupe 65 lignes sur une page qui en contient normalement 66, ce qui permet d'archiver une page de listing par bloc de 512 octets. Il faut remarquer que, contrairement au BASIC, Pascal enregistre ses fichiers sur disquette de façon continue, si bien qu'il est très simple de lister un fichier connaissant l'adresse de son premier bloc et sa longueur (commande E sous FILER).

Le programme lui-même est simple et s'articule autour d'une instruction UNITREAD (numdrive, fich, 512, numbloc). Cette instruction est capable de lire des informations non structurées, contrairement aux autres instructions d'entrée/sortie. Les paramètres de cette instruction sont : le numéro de drive, fich (un tableau de caractères), 512 (nombre d'octets à lire) et numbloc (le numéro du bloc).

Notons également qu'afin d'alléger au maximum le programme, aucun contrôle de cohérence n'a été programmé dans les saisies, ce qui implique un tant soit peu de vigilance lors de l'entrée des paramètres. On aurait pu effectuer des tests de cohérence, mais il faut bien comprendre que l'utilisation de ce programme est ponctuelle, dans le but de scruter des informations sur disquette, ce qui n'est pas une manipulation courante.

Il n'est toutefois pas inutile de l'exécuter simplement par curiosité, notamment sur les fichiers illisibles par des moyens « légaux » comme SYSTEM.APPLE, SYSTEM.MISCINFO, ... ou sur le texte de votre programme (vous serez surpris !). A titre d'exemple, nous avons joint en page 34 un « dump » partiel du fichier CTRL.DATA créé par le programme WRITECTRL.

Mise en œuvre du programme

1 . Entrer le programme WRITECTRL

2 . L'exécuter en y entrant les mnémoniques ASCII de 0 à 32. S'aider de Pom's 4 ou du « dump » donné en annexe.

3 . Entrer le programme LETBLOCK. Vous pouvez changer le numéro du drive sur lequel se trouve FICHCTRL en remplaçant #5 : CTRL.DATA par #4 : CTRL.DATA

4 . Lancer l'exécution.

Bien entendu, si vous avez acheté la disquette d'accompagnement de ce numéro, vous y trouverez les programmes tout prêts.

Conclusion

Comme nous l'avons remarqué plus haut, ce programme n'est pas d'une utilisation courante, mais il peut apporter beaucoup à ceux qui travaillent « près » du système. De toute façon, il présente un intérêt pédagogique indiscutable. Pour l'instant, nous nous limitons à la publication d'un programme capable uniquement de lire des blocs sur disque, donc sans danger. Il lui manque la possibilité de modifier les octets lus, puis de les réécrire. Nous ne doutons pas que certains d'entre vous l'écriront rapidement et enverront leurs contributions à Pom's. Pour les autres, si la suite les intéresse, nous pourrons la publier ultérieurement.

Programme DUMP

```
PROGRAM LECTBLOCK;
TYPE CARACT=PACKED RECORD
               NOM:STRING[3]
             END;

VAR FICH:PACKED ARRAY[0..511] OF CHAR;
    PAPIER:TEXT;
    TABCTRL:PACKED ARRAY[0..32] OF STRING[3];
    FICHCTRL:FILE OF CARACT;
    NUMBLOCK,NUMDRIVE,NBBLOCK:INTEGER;
    REP:CHAR;

PROCEDURE LECTABCTRL;
VAR I:INTEGER;

BEGIN
  RESET(FICHCTRL,'#5:CTRL.DATA');
  FOR I:=0 TO 32 DO
  BEGIN
    TABCTRL[I]:=FICHCTRL^.NOM;
    GET(FICHCTRL)
  END;
  CLOSE(FICHCTRL);
END;

PROCEDURE LECTURE;
VAR I,J,CH:INTEGER;

BEGIN
  REWRITE(PAPIER,'PRINTER:');
  WRITELN(PAPIER,'BLOCK NUMERO : ':45,NUMBLOCK);
  UNITREAD(NUMDRIVE,FICH,512,NUMBLOCK);
  I:=0;
  REPEAT
    WRITE(PAPIER,I:3,' ');
    FOR J:=1 TO 8 DO
    BEGIN
      CH:=ORD(FICH[I]);
      WRITE(PAPIER,CH:4);
      I:=I+1;
    END;
    WRITE(PAPIER,'    ');
    I:=I-8;
    FOR J:=1 TO 8 DO
    BEGIN
      CH:=ORD(FICH[I]);
      IF CH>127 THEN
        BEGIN
          WRITE(PAPIER,' ^');
          CH:=CH-128
        END
        ELSE WRITE(PAPIER,'  ');
      IF CH<=32 THEN WRITE(PAPIER,TABCTRL[CH])
        ELSE WRITE(PAPIER,CHR(CH):3);
      I:=I+1
```

```
    END;
    WRITELN(PAPIER);
  UNTIL I>511;
  CLOSE(PAPIER);
  END;

BEGIN (*PROGRAMME LECTURE BLOCK(S)*)
  LECTABCTRL;
  REPEAT
    PAGE(OUTPUT);
    GOTOXY(5,5);WRITE('NUMERO DU DRIVE (SANS #) : ');
    READLN(NUMDRIVE);
    GOTOXY(5,10);WRITE('NUMERO DU PREMIER BLOCK : ');
    READLN (NUMBLOCK);
    GOTOXY(5,15);WRITE('NOMBRE DE BLOCK(S) : ');
    READLN(NBBLOCK);
    REPEAT
      LECTURE;
      NUMBLOCK:=NUMBLOCK+1;
      NBBLOCK:=NBBLOCK-1;
    UNTIL NBBLOCK=0;
    GOTOXY(10,22);WRITE('CONTINUER ? (O/N) ');
    READ(REP);
  UNTIL REP='N'
END.
```

Programme WRITECTRL

```
PROGRAM WRITECTRL;
TYPE CARACT=PACKED RECORD
                NOM:STRING[3]
            END;

VAR I:INTEGER;
    FICHCTRL:FILE OF CARACT;


PROCEDURE OUVERTURE;
BEGIN
  (*$I-*)
  RESET(FICHCTRL,'#5:CTRL.DATA');
  (*$I+*)
  IF IORESULT<>0 THEN
    BEGIN
      IF IORESULT=10 THEN REWRITE(FICHCTRL,'#5:CTRL.DATA')
        ELSE BEGIN
               PAGE(OUTPUT);
               WRITE('ERREUR E/S');
               EXIT(WRITECTRL)
             END;
    END;
  CLOSE(FICHCTRL,LOCK)
END;

BEGIN
  OUVERTURE;
  RESET(FICHCTRL,'#5:CTRL.DATA');
  PAGE(OUTPUT);
  FOR I:=0 TO 32 DO
  BEGIN
    WRITE(I:3,' = ');
    READLN(FICHCTRL^.NOM);
    PUT(FICHCTRL);
  END;
  CLOSE(FICHCTRL,LOCK)
END.
```

Exemples de dump en page 34

TOUT SUR LA MICRO INFORMATIQUE

du 14 au 18 juin 1983

L'Exposition :
du 14 au 18 juin 1983

• mardi 14, mercredi 15, vendredi 17, de 9 h 30 à 18 h
• nocturne le jeudi 16, de 9 h 30 à 22 h • samedi 18 de 9 h 30 à 17 h.

Tous les matériels, toutes les applications des micro-ordinateurs. Que votre motivation soit professionnelle ou personnelle, venez évaluer les matériels, comparer leurs coûts et vous renseigner sur les applications disponibles.

Nouveauté 83 : une animation « logiciel »

(organisée par Sybex et Logiciels & Services).

Une banque de données des logiciels présentés par les exposants à MICRO-EXPO vous guidera, selon vos besoins spécifiques, à travers l'exposition.
Un « grand concours » destiné à récompenser les meilleurs logiciels pour micro-ordinateurs. Les lauréats présenteront leurs logiciels pendant toute la durée de MICRO-EXPO (Renseignements : Logiciels & Services, tél. : (1) 226.11.25).

Et pendant toute la durée de l'exposition, une animation-jeux : venez affrontez l'ordinateur et peut-être... le battre.

Le Congrès
du 13 au 18 juin 1983

Du débutant à l'expert, une occasion unique dans l'année de se former et de s'informer.

Séminaires professionnels	Conférences Grand Public
Les micro-ordinateurs : présentation, choix	Choix d'un micro-ordinateur
Les microprocesseurs	Les micro-ordinateurs de poche
Les langages : BASIC, PASCAL, APL	Le traitement de texte
Les nouveaux langages : FORTH, LOGO, PROLOG, C	L'enseignement assisté par ordinateur - E.A.O.
La télématique	Les systèmes d'exploitation pour micro-ordinateurs 16 bits
Le traitement de texte	Journées spécialisées, avec démonstrations : — pour les professions juridiques, — pour la médecine, — pour l'agriculture.
L'entreprise, son système d'information et l'informatique	
La comptabilité et ses logiciels	Journées des constructeurs : COMMODORE, GOUPIL, HEWLETT-PACKARD, TANDY, THOMSON, VICTOR LAMBDA
Les bases de données et leurs logiciels	
La gestion et le Visicalc	
Le système d'exploitation MS-DOS	

INFORMATION

SYBEX

Nom : _____

Adresse : _____

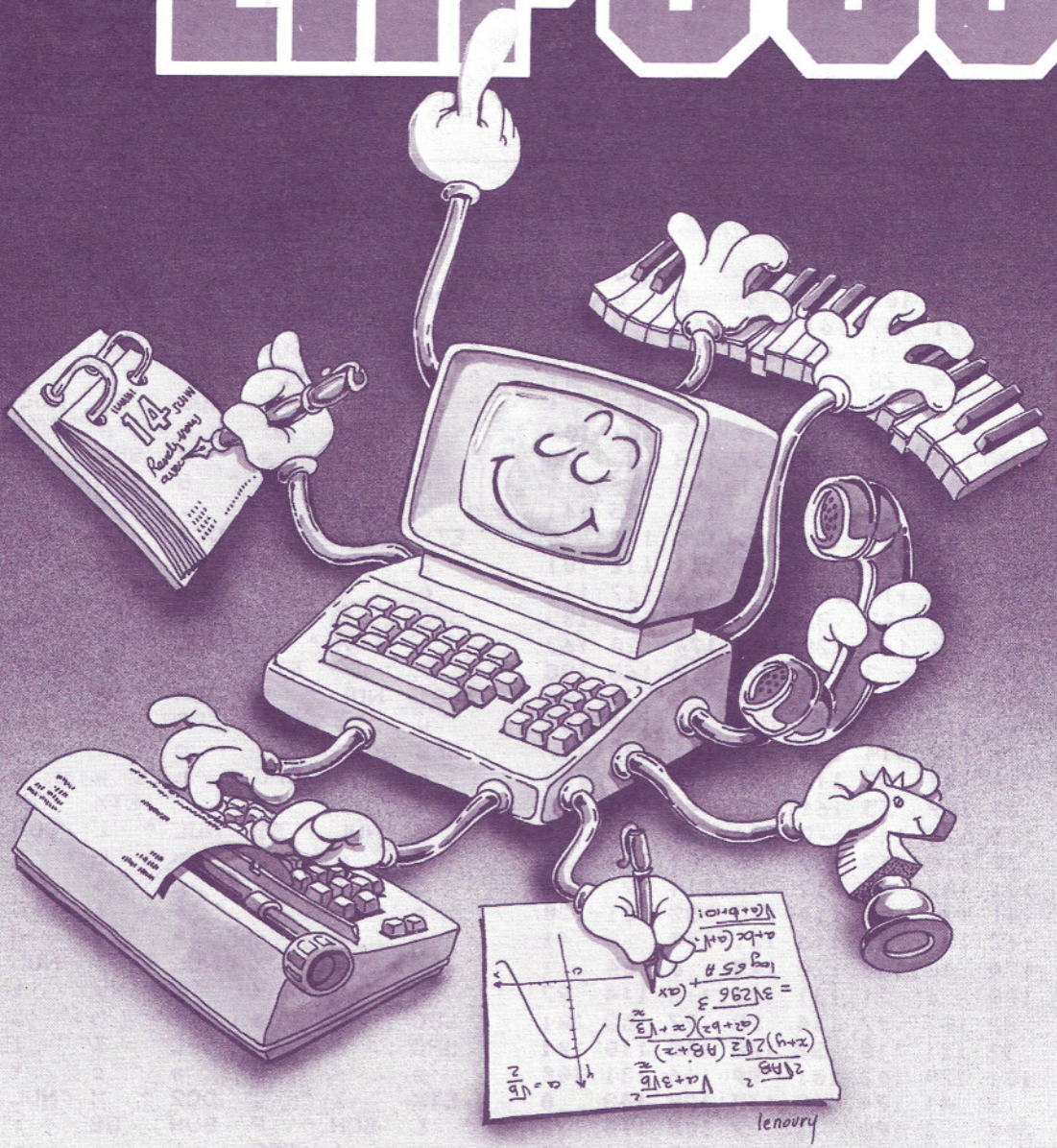
Code postal : |_|_|_|_|_| Ville : _____

☐ ____ entrées(s) à demi-tarif ☐ Programme détaillé du congrès

Pom. 3/83

4, place Félix-Éboué - 75583 PARIS Cedex 12 - Tél. : (1) 347.30.20 - Telex : 211 801 F

MICRO EXPO 83



8e Congrès-Exposition - MICRO-ORDINATEURS

Palais des Congrès - CIP - Porte Maillot - Paris

organisé par :

SYBEX

SYBEX 4, place Félix-Éboué - 75583 PARIS - Tél. : (1) 347.30.20 - Telex : 211 801 F

Fichier CTRL-DATA : dump partiel

BLOCK NUMERO : 21

0	3	78	85	76	3	83	79	72	ETX	N	U	L	ETX	S	O	H
8	3	83	84	88	3	69	84	88	ETX	S	T	X	ETX	E	T	X
16	3	69	79	84	3	69	78	81	ETX	E	O	T	ETX	E	N	Q
24	3	65	67	75	3	66	69	76	ETX	A	C	K	ETX	B	E	L
32	3	32	66	83	3	32	72	84	ETX	SPC	B	S	ETX	SPC	H	T
40	3	32	76	70	3	32	86	84	ETX	SPC	L	F	ETX	SPC	V	T
48	3	32	70	70	3	32	67	82	ETX	SPC	F	F	ETX	SPC	C	R
56	3	32	83	79	3	32	83	73	ETX	SPC	S	O	ETX	SPC	S	I
64	3	68	76	69	3	68	67	49	ETX	D	L	E	ETX	D	C	1
72	3	68	67	50	3	68	67	51	ETX	D	C	2	ETX	D	C	3
80	3	68	67	52	3	78	65	75	ETX	D	C	4	ETX	N	A	K
88	3	83	89	78	3	69	84	66	ETX	S	Y	N	ETX	E	T	B
96	3	67	65	78	3	32	69	77	ETX	C	A	N	ETX	SPC	E	M
104	3	83	85	66	3	69	83	67	ETX	S	U	B	ETX	E	S	C
112	3	32	70	83	3	32	71	83	ETX	SPC	F	S	ETX	SPC	G	S
120	3	32	82	83	3	32	85	83	ETX	SPC	R	S	ETX	SPC	U	S
128	3	83	80	67	0	0	0	0	ETX	S	P	C	NUL	NUL	NUL	NUL

Fichier 6500 ERRORS : dump partiel

BLOCK NUMERO : 151

0	1	32	185	30	0	0	0	0	SOH	SPC	^	9	RS	NUL	NUL	NUL	NUL
8	174	21	147	161	7	165	128	211	^	.	NAK	^DC3	^	!	BEL	^	%	^NUL	^	S
16	190	184	1	94	25	1	165	128	^	>	^	8	SOH		^	EM	SOH	^	%	^NUL
24	185	174	20	167	55	161	3	41	^	9	^	.	DC4	^	'	7	^	!	ETX	)
32	174	18	165	128	143	0	204	1	^	.	DC2	^	%	^NUL	^	SI	NUL	^	L	SOH
40	198	1	15	117	110	100	101	102	^	F	SOH	SI	u	n	d	e	f
48	105	110	101	100	32	108	97	98	i	n	e	d	SPC	l	a	b
56	101	108	190	184	1	94	25	1	e	l	^	>	^	8	SOH	^	EM	SOH
64	165	128	185	174	20	167	55	161	^	%	^NUL	^	9	^	.	DC4	^	'	7	^	!
72	3	41	174	18	165	128	143	0	ETX	)	^	.	DC2	^	%	^NUL	^	SI	NUL
80	204	1	198	1	20	111	112	101	^	L	SOH	^	F	SOH	DC4	o	p	e
88	114	97	110	100	32	111	117	116	r	a	n	d	SPC	o	u	t
96	32	111	102	32	114	97	110	103	SPC	o	f	SPC	r	a	n	g
104	101	1	165	128	185	174	20	167	e	SOH	^	%	^NUL	^	9	^	.	DC4	^	'
112	55	161	3	41	174	18	165	128	7	^	!	ETX	)	^	.	DC2	^	%	^NUL
120	143	0	204	1	198	1	24	109	^	SI	NUL	^	L	SOH	^	F	SOH	CAN	m
128	117	115	116	32	104	97	118	101	u	s	t	SPC	h	a	v	e
136	32	112	114	111	99	101	100	117	SPC	p	r	o	c	e	d	u
144	114	101	32	110	97	109	101	174	r	e	SPC	n	a	m	e	^	.
152	20	167	55	161	3	41	174	18	DC4	^	'	7	^	!	ETX	)	^	.	DC2
160	165	128	143	0	204	1	198	1	^	%	^NUL	^	SI	NUL	^	L	SOH	^	F	SOH
168	29	110	117	109	98	101	114	32	GS	n	u	m	b	e	r	SPC
176	111	102	32	112	97	114	97	109	o	f	SPC	p	a	r	a	m
184	101	116	101	114	115	32	101	120	e	t	e	r	s	SPC	e	x
192	112	101	99	116	101	100	3	41	p	e	c	t	e	d	ETX	)
200	174	18	165	128	143	0	204	1	^	.	DC2	^	%	^NUL	^	SI	NUL	^	L	SOH
208	198	1	21	101	120	116	114	97	^	F	SOH	NAK	e	x	t	r	a
216	32	103	97	114	98	97	103	101	SPC	g	a	r	b	a	g	e
224	32	111	110	32	108	105	110	101	SPC	o	n	SPC	l	i	n	e
232	101	120	112	101	99	116	101	100	e	x	p	e	c	t	e	d
240	3	41	174	18	165	128	143	0	ETX	)	^	.	DC2	^	%	^NUL	^	SI	NUL
248	204	1	198	1	29	105	110	112	^	L	SOH	^	F	SOH	GS	i	n	p
256	117	116	32	108	105	110	101	32	u	t	SPC	l	i	n	e	SPC
264	111	118	101	114	32	56	48	32	o	v	e	r	SPC	8	0	SPC
272	99	104	97	114	97	99	116	101	c	h	a	r	a	c	t	e
280	114	115	3	41	174	18	165	128	r	s	ETX	)	^	.	DC2	^	%	^NUL
288	143	0	204	1	198	1	16	110	^	SI	NUL	^	L	SOH	^	F	SOH	DLE	n
296	111	116	32	101	110	111	117	103	o	t	SPC	e	n	o	u	g
304	104	32	46	73	70	39	115	56	h	SPC	.	I	F	'	s	8
312	48	32	99	104	97	114	97	99	0	SPC	c	h	a	r	a	c
320	116	101	114	115	3	41	174	18	t	e	r	s	ETX	)	^	.	DC2

Un générateur

Denis Sureau

Bien que totalement inédit, ce générateur peut être d'une grande utilité pour un programme scientifique ou éducatif, ainsi que pour tout programme devant conserver et utiliser des données sans créer de fichiers. En fait, il devrait aussi permettre de réaliser des programmes sortant de l'ordinaire...

Le générateur donnera à votre programme la faculté de se développer lui-même, grâce à une simple commande qui, lorqu'elle est rencontrée, a pour effet d'ajouter un numéro de ligne à toute instruction Applesoft et d'insérer la liigne en question dans le programme.

En fait, il a fallu pour l'écrire recourir à des routines de la ROM Applesoft assez peu connues... Ces routines seront indiquées plus loin.

Principe de fonctionnement

Ayant souvent éprouvé l'inconvénient, dans un programme utilisant des fonctions mathématiques, de devoir stopper celui-ci pour y insérer des lignes de définitions ou pour y incorporer des DATA, je me suis dit qu'il serait merveilleux de disposer d'une routine en asssembleur accordant aux programmes écrits en Applesoft une dimension évolutive, la capacité de s'auto-générer !

Les données entrées par INPUT au cours du programme pourraient par exemple lui être incorporées sous forme de lignes d'instructions, avec un numéro qui serait automatiquement incrémenté et placé au début de la ligne générée.

Pour que le générateur puisse opérer sur toute instruction Applesoft, une commande INPUT digérant virgules et deux-points est indispensable ; or, cette commande a été réalisée antérieurement et publié dans le numéro 5 de Pom's (voir « La programmation facilitée »).

Parmi les instructions qui composeront les lignes générées, l'une serait corrélative à l'auto-écriture d'instructions DEF. En effet, lorsque le programme appellera une ligne contenant des instructions DEF, cet appel nécessitera un RETURN, qui devra par conséquent être produit par le générateur.
Demander au programme d'effectuer un appel par GOSUB ou GOTO à des lignes qui n'existent pas encore, et que l'on propose d'ajouter durant le traitement, pose quelques questions ! Une convention s'impose alors : les lignes ajoutées commenceront par exemple à partir du numéro 20 000, avec un incrément de 10 ; on calculera ainsi le numéro d'une ligne créée à partir de son rang. S'il s'agit de la troisième, la première ligne ayant le numéro 20 000, ce sera la ligne 20 020.

On se dit que, puisque le générateur peut ajouter une nouvelle ligne au programme, il serait également nécessaire de pouvoir modifier la dernière ligne existante, lorsqu'il s'agit d'un RETURN notamment. Cette possibilité serait obtenue par une option à laquelle on donnera le nom de POP.

Ces projets me paraissaient prometteurs ; j'imaginais un programme démarrant avec deux lignes, s'autogénérant à partir de quelques indications et se sauvegardant tout seul sur la disquette, une fois la mémoire saturée...

Un écueil de taille s'opposait toutefois à la réalisation : il fallait trouver la procédure Applesoft convertissant les lignes en format programme, dans lequel les instructions sont représentées par un code d'un seul octet. Je ne possédais pas la documentation nécessaire et entrepris de la chercher tout seul dans l'Applesoft, à partir de l'adresse $D000.

C'est à l'adresse $D56D que je l'ai enfin trouvée, rebelle à un emploi immédiat, finissant avec la longueur des données en Y alors qu'elle est initialement en X dans le buffer d'entrée et modifiant en outre TXTPTR, l'ensemble des pointeurs servant à analyser le programme en cours. Cette routine une fois découverte (ainsi que quelques autres à l'occasion), le module était réalisé ; en voici le mode d'emploi.

Mode d'emploi

Une seule commande permet d'obtenir la génération d'une ligne Applesoft de tout type ; son format comporte deux variantes, soit l'emploi d'un mot réservé de l'Applesoft, soit l'emploi d'un caractère quelconque.

Les instructions & DATA A$ et & DEF A$,lorsqu'elles sont rencontrées, ont pour effet d'ajouter au programme les instructions contenues dans A$ sur une ligne qui commencera automatiquement par DATA ou DEF. Plus généralement, avec la syntaxe & X A$, X est choisi par l'utilisateur. Cela peut être un caractère quelconque ou un mot réservé de l'Applesoft, avec un effet différent : le mot réservé sera incorporé en début de ligne, le caractère ne le sera pas.

L'option POP utilise la syntaxe & POP DATA A$ ou enfin & POP X A$, la nouvelle ligne remplaçant la dernière et prenant son numéro.

Les commandes du générateur peuvent s'employer en mode direct si un programme est présent en mémoire et si la LOMEM a été déplacée. Ceci est une précaution dont on s'assurera toujours avant d'utiliser le générateur : la LOMEM doit être placée suffisamment haut pour protéger les variables, puisqu'elles suivent immédiatement, en temps normal, le programme dans la mémoire.

NDLR : ne pas oublier d'abaisser la HIMEM au-dessous du début du générateur avec HIMEM : 38048 si l'origine est en $94A0 comme sur le listing.

Routines Applesoft

Les deux plus importantes sont MOVSTR ($E5E2) pour le transfert des données et en $D56D celle qui « tokenize » le buffer d'entrée, c'est-à-dire qui convertit les mots réservés en codes.

On appelle $D56D avec X mis à 0, en sauvegardant TXTPTR qui sera pointé sur le buffer d'entrée ($200), et en remplaçant par $0 le délimiteur $D (RETURN) se trouvant à la fin du texte lors d'une entrée par INPUT. Convertie en format Applesoft, la ligne se trouve bien sûr réduite ; le registre Y contiendra alors la nou-

velle longueur de la ligne, délimiteur compris.

MOVSTR transfère une donnée pointée par Y (ADL), X (ADH), longueur en A vers l'adresse pointée en $71 et $72 (FRESPC).

MOVINS ($E5D4) fait la même chose que MOVSTR mais copie une chaîne dont le descripteur est pointé en $AB-$AC (STRNG1). Le registre Y revient à 0 et $71-$72 pointent maintenant sur la position qui suit la chaîne transférée. Ces pointeurs sont incrémentés par la valeur de l'accumulateur A par une routine dont l'adresse est $E5F3.

FNDLIN ($D61A) est utilisée par la commande RESTORE. Elle recherche l'adresse de la ligne dont le numéro a été placé en $50-$51 (LINNUM) et la pointe en $9B-$9C (LOWTR). Si la retenue C est mise à 0, la ligne n'a pas été trouvée.

Deux lignes consécutives en Applesoft séparées par 5 octets : le code 0 de fin de ligne, 2 octets qui pointent sur la ligne suivante, 2 octets pour le numéro. L'adresse du numéro de la dernière ligne s'obtient donc en lisant les 2 premiers octets de chaque ligne, jusqu'à ce que deux « 0 » consécutifs signalent la fin de programme.

L'adresse de la ligne sera pointée dans ADRNUM ($1E, $1F), utilisé pour la conserver, et on fera deux adressages indirects indexés par Y pour avoir accès au numéro de ligne. Les deux premiers octets de la ligne générée pointeront sur la fin du programme. Ils seront suivis du numéro de la dernière ligne incrémenté de 10, puis éventuellement du code instruction employé dans le format et finalement, du contenu.

Le listing du programme commente toutes les étapes de cette génération.

Les bons trucs...

Le générateur est court grâce à l'emploi de routines Applesoft ; étant relogeable, il peut être chargé à l'adresse voulue, de préférence au sommet de la mémoire, protégé par l'abaissement de la HIMEM.

A noter dans la rubrique « trucs et astuces » : quand on ne manipule pas de fichiers, mais beaucoup de chaînes de caractères, il est possible de réduire la zone des variables, à l'aide de HIMEM et LOMEM, de 1 K à 3 K. D'où réduction des problèmes dûs au nettoyage mémoire !

Des applications originales peuvent s'imaginer, telles que la conversion de routines en lignes de DATA, la création de programmes à partir de fonctions et d'affectations déterminées par calcul. Imaginons qu'à partir de bases bien définies un programme puisse se construire lui-même pour atteindre un but connu de lui seul...

GÉNÉRATEUR
Big Mac

```
 1 *XXXXXXXXXXXXXXXXXXXXXXX VERS.
 2 *XXXXX GENERATEUR XXXXX 2.0
 3 *XXXXXXXXXXXXXXXXXXXXXXX
 4 *XX PAR DENIS SUREAU XX
 5 *XXXXXXXXXXXXXXXXXXXXXXX
 6 *X COPYRIGHT (C) 1982 X
 7 *XXXXXXXXXXXXXXXXXXXXXXX
 8
 9
10 FNDLIN   =    $D61A
11 FRENUM   =    $DD67
12 PTRGET   =    $DFE3
13 MOVINS   =    $E5D4
14 MOVSTR   =    $E5E2
15 GETADR   =    $E752
16
17 LINNUM   =    $0050
18 VARPNT   =    $0083
19 FORPNT   =    $0085
20 CHRGET   =    $00B1
21 TXTPTR   =    $00B8
22
23 LNGTEXT  =    $0018
24 ADRTEXT  =    $0019     ;$19-$1A
25 INSTRUC  =    $001B
26 SEPARATR =    $001C
27 INCR     =    $001D
28 ADRNUM   =    $001E     ;$1E-$1F
29
30          ORG  $94A0
31
32 *< UN PROGRAMME RELOCATABLE >*
33
34 ADRPROG  JSR  $FF58     ;SUR RTS
35          TSX
36          DEX
37          LDA  $100,X
38          CLC
39          ADC  #DEPART-ADRPROG-2
40          STA  $3F6
41          INX
42          LDA  $100,X
43          ADC  #0
44          STA  $3F7
45          LDA  #$4C      ;JMP
46          STA  $3F5
47          RTS
48
49 DEPART   STA  INSTRUC
50          CMP  #174      ;RESTORE?
51          BEQ  RESTORE
52          TAY
53          JSR  CHRGET
54          CPY  #161      ;POP?
55          BNE  GENER
56          STA  INSTRUC   ;SI OUI,
57          JSR  CHRGET    ;INCR MIS
58          LDA  #0        ;A 0
59          BEQ  GENER2
60 RESTORE  JSR  CHRGET    ;PLACE
61          JSR  FRENUM    ;NUM LGNE
62          JSR  GETADR    ;EN $50-51
63          LDA  LINNUM    ;LE PLACE
64          STA  $7B       ;EN NUM
65          LDA  LINNUM+1  ;DE LIGNE
66          STA  $7C       ;DE DATA
67          JSR  FNDLIN    ;CHERCHE
68          LDA  $9B       ;LA LIGNE
```

```
 69            ADC  #$03       ;ADR EN
 70            STA  $7D        ;$9B-9C.
 71            LDA  $9C        ;MET ADR
 72            ADC  #0         ;LIGNE EN
 73            STA  $7E        ;POINTEURS
 74            RTS             ;DE DATA
 75
 76 GENER      LDA  #$0A       ;MET 10
 77 GENER2     STA  INCR       ;EN INCR.
 78
 79            LDA  #$04
 80            STA  SEPARATR
 81
 82 * LOCALISE LE DESCRIPTEUR DE LA
 83 * VARIABLE ET PLACE LA LONGUEUR
 84 * EN LNGTEXT,L'ADRESSE EN ADRTEXT
 85
 86            JSR  PTRGET
 87            STA  FORPNT     ;SAUVE
 88            STY  FORPNT+1   ;ADR DESCR
 89            STA  $AB        ;POINTEURS
 90            STY  $AC        ;PR MOVINS
 91            LDY  #$02
 92 DSCLOOP    LDA  (VARPNT),Y
 93            STA  LNGTEXT,Y
 94            DEY
 95            BPL  DSCLOOP
 96
 97            LDA  INSTRUC
 98            CMP  #132       ;INPUT?
 99            BNE  ADRDRN
100
101 * $FD6F EST L'INPUT DU MONITEUR
102 * $D539 CONVERTIT LA CHAINE EN
103 *    FORMAT APPLESOFT,BIT 7 A 0
104 * $E3E9 & $DA9A COPIENT LA DONNEE
105 * DU BUFFER D'ENTREE EN MEMOIRE,
106 * DESCRIPTEUR POINTE EN FORPNT.
107
108           JSR  $FD6F
109           JSR  $D539
110           LDA  #0
111           LDY  #2
112           TAX             ;DELIMITR
113           JSR  $E3E9
114           JMP  $DA9A
115
116 * TROUVE L'ADRESSE DE LA DERNIERE
117 * LIGNE
118
119 ADRDRN     LDY  #1
120           LDX  103        ;DEBUT DE
121           LDA  104        ;PROGR.
122           BNE  ADRTRNS
123 LOOP      STX  ADRNUM
124           LDA  $9C
125           STA  ADRNUM+1
126           DEY             ;LIT ADR
127           LDA  ($9B),Y    ;LIGNE
128           TAX             ;SUIVANTE
129           INY             ;LA MET EN
130           LDA  ($9B),Y    ;$9B-$9C
131 ADRTRNS   STX  $9B
132           STA  $9C
133           LDA  ($9B),Y
134           BNE  LOOP
135
136 * TESTE SI POP OPTION
137
138           LDA  INCR
139           BNE  BRANCH
140
141           LDA  ADRNUM
142           STA  $9B
143           LDA  ADRNUM+1
144           STA  $9C
145
146 BRANCH    LDA  INSTRUC
147           BPL  TOKEN      ;CARACT.
148           INC  SEPARATR
149           CMP  #131       ;DATA?
150           BEQ  COPY
151
152 * 'TOKENIZE' UNE INSTRUCTION
153 * DEF OU UNE LIGNE D'INSTRUCTIONS
154
155 TOKEN     LDA  TXTPTR     ;SAUVE
156           PHA             ;TXTPTR
157           LDA  TXTPTR+1
158           PHA
159           LDA  #0         ;COPIE
160           STA  $71        ;LA CHAINE
161           LDA  #$02       ;EN MEM
162           STA  $72        ;DANS LE
163           JSR  MOVINS     ;BUFFER
164
165 * ELLE N'ACCEPTE PAS UN CODE SEUL
166 * AUSSI ON AJOUTE UN CODE NUL,':'
167 * DANS LE BUFFER,IL SERA SUPPRIME
168 * ENSUITE
169
170           LDX  LNGTEXT
171           LDA  #':'       ;MET ':'
172           STA  $200,X
173           LDA  #0         ;MET LE
174           STA  $201,X     ;DELIMITR
175           TAX             ;ZERO
176
177 * $D56D EST LA ROUTINE DE
178 * CONVERSION DES MOTS-CLES
179 * EN CODES, Y PASSERA LA NOUVELLE
180 * LONGUEUR
181
182           STA  $13
183           JSR  $D56D
184           DEY
185           DEY
186           STY  LNGTEXT    ;UPDATE
187           LDA  #$FC       ;LNG & ADR
188           STA  ADRTEXT    ;POUR
189           LDA  #$01       ;MOVINS
190           STA  ADRTEXT+1
191           PLA             ;RESTAURE
192           STA  TXTPTR+1   ;TXTPTR
193           PLA
194           STA  TXTPTR
195
196 COPY      LDA  $9B        ;COPIE
197           CLC             ;LA CHAINE
198           ADC  SEPARATR   ;DE LA VAR
```

```
199             STA  $71        ;OU DU
200             LDA  $9C        ;BUFFER
201             ADC  #0         ;A LA FIN
202             STA  $72        ;DU PROGR
203             LDY  LNGTEXT
204             LDA  #0         ;MET LE
205             STA  ($71),Y    ;DELIMITR
206             LDA  LNGTEXT
207             LDX  ADRTEXT
208             LDY  ADRTEXT+1
209             JSR  MOVSTR
210             LDA  #1         ;INCREM.
211             JSR  $E5F3      ;$71
212
213 * INSERTION DE L'ADRESSE DE LIGNE
214 * SUIVANTE DANS LES 2 PREMIERS
215 * OCTETS DE LA LIGNE, AJUSTEMENT
216 * DES POINTEURS DE FIN DE
217 * PROGRAMME
218
219             LDA  $71
220             STA  ($9B),Y
221             CLC
222             ADC  #2
223             STA  175
224             INY
225             LDA  $72
226             STA  ($9B),Y
227             ADC  #0
228             STA  176
229
230 * INSERTION DU NUMERO DE LIGNE
231 * DANS LES 2 OCTETS SUIVANTS
232
233             INY
234             LDA  (ADRNUM),Y
235             CLC             ;MET LE
236             ADC  INCR       ;NUMERO
237             STA  ($9B),Y    ;DE LIGNE
238             INY
239             LDA  (ADRNUM),Y
240             ADC  #0
241             STA  ($9B),Y
242             LDA  INSTRUC    ;MET
243             BPL  FIN        ;'DATA' OU
244             INY             ;'DEF'
245             STA  ($9B),Y
246
247 * DEUX ZEROS LA OU DEVRAIT SE
248 * TROUVER LE DEBUT D'UNE AUTRE
249 * LIGNE SIGNALENT LA FIN DE
250 * PROGRAMME
251
252 FIN         LDA  #0
253             TAY
254             STA  ($71),Y
255             INY
256             STA  ($71),Y
257             RTS
```

GÉNÉRATEUR

Récapitulation

```
*94A0.95E2

94A0- 20 58 FF BA CA BD 00 01
94A8- 18 69 1B 8D F6 03 E8 BD
94B0- 00 01 69 00 8D F7 03 A9
94B8- 4C 8D F5 03 60 85 1B C9
94C0- AE F0 11 A8 20 B1 00 C0
94C8- A1 D0 2A 85 1B 20 B1 00
94D0- A9 00 F0 23 20 B1 00 20
94D8- 67 DD 20 52 E7 A5 50 85
94E0- 7B A5 51 85 7C 20 1A D6
94E8- A5 9B 69 03 85 7D A5 9C
94F0- 69 00 85 7E 60 A9 0A 85
94F8- 1D A9 04 85 1C 20 E3 DF
9500- 85 85 84 86 85 AB 84 AC
9508- A0 02 B1 83 99 18 00 88
9510- 10 F8 A5 1B C9 84 D0 11
9518- 20 6F FD 20 39 D5 A9 00
9520- A0 02 AA 20 E9 E3 4C 9A
9528- DA A0 01 A6 67 A5 68 D0
9530- 0D 86 1E A5 9C 85 1F 88
9538- B1 9B AA C8 B1 9B 86 9B
9540- 85 9C B1 9B D0 EB A5 1D
9548- D0 08 A5 1E 85 9B A5 1F
9550- 85 9C A5 1B 10 06 E6 1C
9558- C9 83 F0 35 A5 B8 48 A5
9560- B9 48 A9 00 85 71 A9 02
9568- 85 72 20 D4 E5 A6 18 A9
9570- 3A 9D 00 02 A9 00 9D 01
9578- 02 AA 85 13 20 6D D5 88
9580- 88 84 18 A9 FC 85 19 A9
9588- 01 85 1A 68 85 B9 68 85
9590- B8 A5 9B 18 65 1C 85 71
9598- A5 9C 69 00 85 72 A4 18
95A0- A9 00 91 71 A5 18 A6 19
95A8- A4 1A 20 E2 E5 A9 01 20
95B0- F3 E5 A5 71 91 9B 18 69
95B8- 02 85 AF C8 A5 72 91 9B
95C0- 69 00 85 B0 C8 B1 1E 18
95C8- 65 1D 91 9B C8 B1 1E 69
95D0- 00 91 9B A5 1B 10 03 C8
95D8- 91 9B A9 00 A8 91 71 C8
95E0- 91 71 60
```

Un programme de test universel

Denis Sureau

En écrivant le programme TEST, j'avais en vue un programme de test général qui utiliserait des données interchangeables, et serait de préférence très court, puisque ce programme serait la partie commune à tous les programmes de test que l'on voudrait réaliser, et serait donc stocké en de multiples exemplaires.

Il permet de créer des tests dans tout domaine à votre convenance ; il est en outre général au niveau du système, car il ne nécessite aucune configuration spécifique, grâce au générateur de lignes qui « auto-écrira » les données sous forme de DATA.

Un examinateur patient

TEST est un programme simple qui vous pose des questions sur un sujet donné, de façon exhaustive ou aléatoire, et vous fournit la bonne réponse. Quand vous avez sélectionné un chapitre et un mode d'interrogation, il présente la question tandis que la réponse se trouve sous un cache. Puis, lorsque vous avez donné votre réponse, ou simplement appuyé sur RETURN, le cache disparaît pour laisser apparaître la bonne réponse.

Le CURSEUR INVERSE, programme en langage machine qui se trouve sous forme de lignes de DATA, permet de réaliser le cache. Ainsi d'ailleurs que la présentation du programme, notamment le cadre du sommaire et la page de titre. On peut faire énormément de choses avec le CURSEUR INVERSE ; c'est pourquoi je l'ai écrit sous forme relogeable et compatible avec d'autres routines binaires. Comme il utilise l'ampersand, il y aurait eu appropriation de celui-ci par la deuxième routine chargée sur la première si elle l'avait également utilisé ! Ce problème a une solution, que je vous propose plus loin.

Les données que vous avez fournies seront rajoutées automatiquement à la fin du programme de test, à partir de la ligne 20000, sur des lignes de DATA.

Pour créer un test différent, il suffit de supprimer les lignes 20010 et suivantes, et de placer DATA END à la ligne 20000. Les titres des chapitres sont la seule chose que l'on ajoute « à la main » sur le programme. Les lignes 100 et suivantes sont réservées à cet effet et affectées aux variables B$(1), B$(2)... A la ligne 100, le nom du programme affecté à la variable F$ peut aussi être changé pour prendre le nom du test créé.

Un programme cybernétique

Plutôt que de recourir au programme hiérarchisé classique avec un menu principal auquel on revient pour avoir accès à chaque section, on utilise une approche systémique permettant d'appeler le sommaire à partir de n'importe quel endroit. Lorsque le numéro du chapitre est demandé, appuyer sur « S » fait apparaître le sommaire ; la question est ensuite posée de nouveau et on indique alors le numéro de l'option souhaitée. Les options « cherche » et « numéro ligne » ne figurent pas au menu. On les appelle de n'importe quel point du programme, ainsi que chacune des autres fonctions. La description de chacune de ces procédures suit.

PROCEDURE DE TEST. Présentation en 3000, option test complet en 6000, test aléatoire en 4000. La ligne 4097 stocke les numéros de ligne de chaque question posée et teste si le numéro généré aléatoirement figure dans la liste ; ainsi, lors du test aléatoire, une question n'est jamais posée deux fois.

AJOUTE commence en 5000. Les données sont stockées sur deux lignes de DATA, la première contenant le numéro de chapitre et la question, la seconde la réponse.

ARRET débute en ligne 8000. On teste à la ligne 8000 le drapeau DR positionné à 1 pour indiquer que des données ont été entrées et sauvegarder, le cas échéant, le programme.

CHERCHE et NUMERO DE LIGNE se partagent le même sous-programme à partir de 2000, en fonction de A, la variable réservée aux numéros d'options.

Problèmes d'interfaçage

Comment utiliser simultanément plusieurs routines ayant pour vecteur l'ampersand ? La solution que j'ai adoptée dans la routine INVERSE consiste à sauter à l'adresse indiquée en 6-7 si l'ampersand est suivi d'un code autre que INVERSE, par exemple ici RESTORE, DATA, DEF, etc... On place donc en 6-7 l'adresse du début de la seconde routine, augmentée du nombre des octets (inutilisés dans ce cas) qui précèdent les tests de syntaxe dans celle-ci, soit, dans le cas du générateur, 29 octets pour la partie initialisation.

USR pourrait encore être utilisé pour une troisième routine, les positions $A-$B-$C n'étant pas utilisées.

Conclusion

J'espère que ce programme vous aidera à contrôler vos connaissances dans un domaine, ou à les vérifier avant un examen, de façon facile et rapide. Il y a bien sûr la nécessité de rentrer en différé les questions et réponses, mais cet exercice peut s'avérer utile pour mieux comprendre un texte.

En fait, il appelle (je crois) un programme complémentaire, qui apprendrait au lieu de tester, et fonctionnerait selon le principe de l'enseignement programmé... J'ignore si un tel programme existe. J'essaierai d'en écrire une version à l'occasion.

Liste des variables réservées

A = numéro d'option
A$() = libellé des fonctions
B = hauteur sonore ou divers
B$() = titres des chapitres
C = durée du son ou divers
DR = signale si des données ont été ajoutées
F$ = nom du programme
N% = nombre de questions
R% = indice du tableau enregistrant les lignes lors du test aléatoire.

```
0  PRINT "BRUN GENERATEUR"
1  HIMEM: 38047: REM SOUSTRAIRE 16384 PO
       UR 32K
3  LOMEM:  PEEK (115) +  PEEK (116) * 25
       6 - 1000
4  REM PLACE EN 6-7 L'ADRESSE DE LA PROC
       EDURE OU DOIT POINTER L'AMPERSAND
        LORSQU'IL EST SUIVI D'UN CODE DI
       FFERENT DE CELUI D'INVERSE
5 N = 38048 + 29: POKE 7,N / 256: POKE 6
       ,N -  PEEK (7) * 256
6 B = 0:C = N:I = N:J = N
7  GOSUB 13000: GOSUB 100: GOTO 9000
9  REM ROUTINES FREQUENTES
11  POKE  - 16368,0: RETURN
12  PRINT  TAB( 10)"OPTION "A$(A): RETUR
       N
13 B =  PEEK ( - 16384): RETURN
14  FOR J = 1 TO 500: NEXT : RETURN
15 N =  PEEK (123) +  PEEK (124) * 256:
       RETURN
16  POKE 773,B: POKE 774,C: CALL 778: RE
       TURN
17 N = 0: &  RESTORE 20000: PRINT "ATTEN
       DEZ..."
18  READ A$: IF A$ <  > C$ AND A$ <  > "
       FIN" THEN N = N + 1: GOTO 18
19  RETURN
20  PRINT "CHAPITRE #";: CALL  - 868: GO
       SUB 23
21  IF C$ = "S" THEN  GOSUB 1000: VTAB 2
       3: GOTO 20
22  RETURN
23  &  INPUT C$: IF C$ = "" THEN  POP
24  RETURN
25  GOSUB 13: IF B = 155 THEN  POP
26  RETURN
27  GOSUB 11: WAIT  - 16384,128: GOSUB 1
       3: IF B = 141 OR B = 160 THEN  GO
       SUB 11: RETURN
28  POP : GOTO 10450
30  POKE 32,4: POKE 33,30: POKE 34,5: PR
       INT : RETURN
40  &  INVERSE 5,30,6,3: RETURN
50  INVERSE
52  VTAB 5: HTAB 6: PRINT "QUESTION"
56  VTAB 11: HTAB 7: PRINT "REPONSE EXAC
       TE"
57  NORMAL : RETURN
60 A =  PEEK (36): PRINT A$(I): &  INVER
       SE A + 1,1, PEEK (37),1: PRINT :
       RETURN
80  VTAB 1: HTAB 10: GOTO 20
90  VTAB 4: CALL  - 958: GOSUB 50: VTAB
       19: PRINT "VOTRE REPONSE":B = 60:
       C = 30: GOSUB 16
91  GOSUB 30: VTAB 6: PRINT B$: TEXT : G
       OSUB 40: &  INVERSE 1,40,12,5
92  VTAB 20: &  INPUT E$
94  &  INVERSE 1,40,12,5: VTAB 12: PRINT
        A$
95  GOTO 27
96  HOME
97  PRINT "FIN DE DONNEES": GOTO 27
99  REM VOS VARIABLES
100 F$ = "TEST"
110 B$(1) = "PROGRAMMATION"
120 B$(2) = "FICHIERS"
121 B$(3) = "TABLES ET TABLEAUX"
122 B$(4) = "STRUCTURE DE DONNEES"
123 B$(5) = "PROGRAMMATION STRUCTUREE"
124 B$(6) = "MODELES:FONCTIONS COURANTES
       "
125 B$(7) = "MODELES:METHODOLOGIE"
126 B$(8) = "ALGORITHMIQUE"
130  RETURN
990  REM SOMMAIRE
1000 I = 0
1040  TEXT : HOME
1045  PRINT : PRINT A$(3),F$
1047  VTAB 5
1050 I = I + 1
1060  IF B$(I) = "" THEN 1110
1070  IF  PEEK (37) > 20 THEN  PRINT "AP
       PUYEZ SUR UNE TOUCHE";: GOSUB 111
       0:I = I - 1: GOTO 1040
1080  PRINT  TAB( 4) MID$ (R$,I * 5 - 4,
       5);
1090  PRINT ".  "B$(I): PRINT
1100  GOTO 1050
1110  &  INVERSE 1,9,1,3: &  INVERSE 10,
       31,4,20
1115  &  INVERSE 11,29,5,18: &  INVERSE
       2,7,5,18
1120 B = 80:C = 40: GOSUB 16
1130  GOTO 27
1990  REM RECHERCHE/NUM LIGNE
2000  HOME : GOSUB 12: VTAB 4: PRINT "QU
       ESTION:";: GOSUB 23
2010  GOSUB 17: IF A$ = "FIN" THEN  GOSU
       B 97: GOTO 2000
2015  IF A = 7 THEN  PRINT "LIGNE: ": GO
       SUB 15: PRINT N: GOTO 27
2020  READ A$: GOSUB 90: GOTO 2000
2990  REM QUESTION
3000  VTAB 21
3010  PRINT "1. SUR TOUT"
3020  PRINT "2. AU HASARD"
3030  GET A$:N =  VAL (A$)
3040  GOSUB 20: TEXT : HOME
3045  GOSUB 12: HTAB 10: PRINT "CHAPITRE
        ";C$
3047  &  INVERSE 9,22,1,3
3060  VTAB 1: HTAB 1: PRINT "ESC POUR":
       PRINT "QUITTER"
3100  ON N GOTO 6000,4000
3490  RETURN
3990  REM TEST ALEATOIRE
4000  IF  NOT N% THEN  RETURN
4010 R% = 0
4040  GOSUB 11
4050 N =  INT ( RND (1) * N% * 2) + 1
4060  &  RESTORE N * 10 + 19990
4070  READ A$: IF A$ = "FIN" THEN 4050
4080  GOSUB 25
4090  IF A$ <  > C$ THEN 4070
4092  GOSUB 15
4095  FOR I = 0 TO R%: IF N = R%(I) THEN
       4050
4097  NEXT I:R%(I) = N:R% = (I < 51) * I
4100  READ B$: READ A$
4120  GOSUB 90: GOTO 4050
4990  REM AJOUTE
5000  IF  PEEK (175) +  PEEK (176) * 256
       > ( PEEK (105) +  PEEK (106) * 2
       56 - 239) THEN  PRINT "MEMOIRE SA
       TUREE": GOTO 10
5050  HOME : GOSUB 12: PRINT " "N%" QUES
       TIONS"
5055  GOSUB 50
5060  VTAB 12
5064  &  INPUT B$: IF B$ = "" THEN  RETU
```

```
      RN
5065  GOSUB 30
5070  VTAB 6: &  INPUT A$
5080  TEXT : GOSUB 80
5090 C$ = C$ + "," +  CHR$ (34) + A$ +
      CHR$ (34)
5100  &  POP  DATA  C$
5102 B$ =  CHR$ (34) + B$ +  CHR$ (34)
5105  &  DATA B$
5107 DR = 1:A$ = "FIN"
5110  &  DATA A$
5210 N% = N% + 1
5490  GOTO 5050
5990  REM TEST EXHAUSTIF
6000  &  RESTORE 20000
6010  READ A$: IF A$ = "FIN" THEN 96
6015  GOSUB 25
6020  IF A$ <  > C$ THEN 6010
6030  READ B$: READ A$
6040  GOSUB 90: GOTO 6010
6990 INSTR.
7000  HOME
7010  GOSUB 12: VTAB 5
7020  POKE 32,8: PRINT
7030  PRINT "R - RECHERCHE UNE QUESTION"
      : PRINT
7040  PRINT "S - QUAND LE CHAPITRE EST "
      ,"DEMANDE,AFFICHE LE SOMMAIRE": P
      RINT
7045  PRINT "L - NUMERO DE LIGNE D'UNE",
      "QUESTION": PRINT
7047  PRINT "ESC - ARRETE LA PROCEDURE",
      "EN COURS": PRINT
7050  PRINT " LE NOM DU PROGRAMME ET LES
      ","CHAPITRES SONT MIS AUX LIGNES"
      ,"100 ET SUIVANTES"
7490  TEXT : GOTO 27
8000  IF  NOT DR THEN 8200
8005  PRINT "NOM DU PROGRAMME: ";F$;: PR
      INT " OK?";: GET A$: IF A$ = "N"
      THEN  HTAB 19: CALL  - 868: INPUT
       F$
8010 D$ =  CHR$ (4)
8030  PRINT D$"SAVE"F$
8200  TEXT : HOME : PRINT "AU REVOIR!":
      END
9000 N$ = "TASIQRL": REM INITIALES
9010 R$ = "    I   II  III   IV    V   V
      I  VII VIII   IX    X   XI  XII X
      III  XIV   XV  XVI XVIIXVIII  XIX
         XX  XXI XXIIXXIII XXIV  XXV XX
      VIXXVIXVIII XXIX  XXX"
9020  DIM R%(50)
9050  GOSUB 17
9060 N% = N / 3
9070  HOME
9990  REM MENU
10000  TEXT : HOME
10005  PRINT ,,,,F$,,,"PAR D. SUREAU"
10010  VTAB 10
10020 A$(1) = "TEST"
10030 A$(2) = "AJOUTE"
10040 A$(3) = "SOMMAIRE"
10050 A$(4) = "INSTRUCTIONS"
10060 A$(5) = "QUITTE & SAUVEGARDE"
10070 A$(6) = "RECHERCHE"
10080 A$(7) = "LIGNE"
10100  FOR I = 1 TO 5: HTAB 22 - I * 2:
      GOSUB 60: NEXT
10200  GOSUB 12000
10450  GET A$
10500  FOR A = 1 TO  LEN (N$)
10510  IF  MID$ (N$,A,1) <  > A$ THEN  N
      EXT : GOTO 10000
10520  IF A <  > 1 THEN  TEXT : HOME
10530  ON A GOSUB 3000,5000,1000,7000,80
      00,2000,2000
10600  GOTO 10000
11990  REM LOGO
12000  GOSUB 11
12005  FOR I = 1 TO 20
12010 J =  INT (I / 2)
12020  &  INVERSE 21 - I,I * 2,11 - J,J
      * 2
12030  NEXT
12040  GOSUB 14: GOSUB 14
12065  GOSUB 13
12066  IF B > 128 THEN  RETURN
12075 B = 30:C = 10: GOSUB 16
12080  GOTO 12040
13000  REM SON & INVERSE
13030  FOR I = 778 TO 903: READ B: POKE
      I,B: NEXT
13040  CALL 799: REM INIT. INVERSE
13050  DATA 173,48,192,136,208,5,206
13060  DATA 6,3,240,9,202,208,245,174
13070  DATA 5,3,76,10,3,96
13080  DATA  32,88,255,186,202,189,0,1,2
      4,105,27,141,246,3,232,189,0,1,10
      5,0,141,247,3,169,76,141,245,3,96
      ,201,158,240,3,108,6,0,160,0,152,
      72,32,245,230,104,168,202,150,24,
      200,192,4,144,241,165,26,133,37,5
      6,101,27,133
13090  DATA  27,165,24,56,101,25,133,25,
      198,37,230,37,32,34,252,165,37,19
      7,27,144,1,96,164,24,177,40,48,4,
      9,128,208,2,41,63,145,40,200,196,
      25,144,239,176,223,255,0
13100  RETURN
19000  REM     ** T E S T **
         COPYRIGHT (C) 1982
       DENIS SUREAU
20000  DATA FIN
```

Message aux créateurs de logiciel

Nous allons bientôt commercialiser des programmes en plus de la revue Pom's, la rémunération des auteurs s'effectuant selon le principe des droits d'auteur.

Bien entendu, les programmes destinés à ce mode de distribution doivent être des produits finis, autant par la qualité des programmes que par celle de la documentation et par l'environnement visuel et sonore.

Nous sommes prêts à analyser vos propositions et à vous aider à transformer des idées en produits logiciels. Envoyez-nous vos contributions ; nous vous aiderons à diffuser ces programmes.

Datalife

BY Verbatim

DISQUETTES ET MINI DISQUETTES TOUTES CONFIGURATIONS

- Certification unitaire 100 % sans erreur.
- Durée de vie : 30 millions de révolutions (standard de l'Industrie 3,5 millions de révolutions).
- Anneau de renforcement en standard sur le 5 1/4 ".
- 5 1/4 " en 48 et 96 TPI, simple et double face.

Importateur exclusif : BFI ELECTRONIQUE - 9 RUE YVART - 75015 PARIS.
Tél. 533-01-37.

Visicalc et traitement de texte

Hervé Thiriez

Je tiens à remercier ici Judith Kertesz, une de nos fidèles lectrices, qui nous a donné l'idée de cet article en nous montrant comment mettre des minuscules dans un tableau Visicalc.

Visicalc avec des minuscules

Ceux d'entre nous qui travaillent avec un Apple II Plus sans la ROM minuscules regrettent parfois de ne pouvoir les utiliser dans Visicalc. En fait, cela peut se faire sans difficulté, quoique l'effort ne soit pas justifié pour un tableau d'usage peu fréquent. Par contre, un tableau à usage multiples, servant à l'élaboration de factures par exemple, est plus satisfaisant s'il peut exploiter les minuscules de l'imprimante.

Le principe est relativement simple. Le fichier Visicalc est un fichier TEXT que l'on peut modifier à l'aide d'un programme de traitement de texte. La procédure est par conséquent la suivante :

1) Créer le tableau souhaité avec Visicalc
2) Sauvegarder le fichier correspondant avec [/SSFACTURE] (ou tout autre nom que « FACTURE »)
Comme dans le livre « Visicalc sur Apple », nous mettons entre crochets les textes à entrer au clavier pour Visicalc.
3) Lire en traitement de texte le fichier FACTURE
4) Remplacer en mode édition les majuscules par les minuscules souhaitées
5) Sauvegarde le fichier sous un nom qui nous rappelle la présence des minuscules, par exemple FACT.MIN

Quand on souhaite imprimer une facture, il suffit de faire [/SLFACT.MIN] en Visicalc, d'y ajouter les informations propres à la facture considérée, puis de lancer son impression. Quand Visicalc lit des minuscules, il fait apparaître à l'écran les majuscules correspondantes. C'est uniquement à l'impression (ou avec les lettres accentuées) que l'on voit qu'il y a effectivement des minuscules aux endroits nécessaires.

Les utilisateurs d'Apple *IIe* pourront aussi constater que le programme Visicalc fait apparaître des majuscules à l'écran, bien que les minuscules existent avec ce matériel.

Bien entendu, si l'on veut éviter de perdre du temps en pratiquant ainsi le va-et-vient entre Visicalc et le programme de traitement de texte, il faut se contenter de majuscules pour les informations relatives à chaque facture spécifique.

Cette façon de travailler suppose que votre programme de traitement de texte sache lire des fichiers texte. Il n'y aura pas de problème si vous travaillez avec Applewriter II, Magic Window ou tout autre système utilisant des fichiers TEXT.

Si par contre vous possédez le célèbre Applewriter I, vous pouvez tout de même vous en tirer en utilisant les programmes TEXT to Applewriter I et Appelwriter I to TEXT présentés ci-dessous.

Nous fournissons en réduction un exemple de facture réalisée pour Pom's. Le fichier FACT.MIN se trouve sur la disquette d'accompagnement. Le lecteur pourra constater sans effort que la solution proposée dans cet article fonctionne sans moindre problème.

Passons maintenant aux programmes de conversion entre un fichier TEXT et le programme Applewriter I.

TEXT to Applewriter I

Afin de convertir un fichier TEXT en format Applewriter I, il suffit d'exécuter le programme suivant. Outre son intérêt pour l'utilisation liée à Visicalc, ce programme permet d'exploiter les possibilités d'Applewriter avec tout fichier TEXT, en particulier pour l'édition de fichiers EXEC.

Une autre application de ce programme de converssion consiste, après avoir effectué avec [/PF-E36 *RETURN* FACTURE] la sauvegarde du tableau à imprimer (et non plus de la description du tableau), à le convertir en format Applewriter. Le tableau Visicalc peut alors être intégré avec CTRL-I dans un rapport édité en Applewriter.

Notre but se limite ici à vous fournir un bon outil de travail. Nous ne commenterons donc pas le mode de fonctionnement de ce programme. Nous laissons le lecteur exercer sa sagacité dans cette analyse.

Applewriter I to TEXT

Ce programme, à l'opposé, transforme un fichier Applewriter quelconque en fichier TEXT. On peut ainsi créer un fichier EXEC en traitement de texte, et le convertir grâce à ce programme en fichier TEXT.

•••/•••

pom's

Editions MEV
49, rue Lamartine
78000 Versailles

STE. MICROFANA & CIE
13, RUE DE L'APPLE 4
75016 PARIS CEDEX 89

Facture No. 1234

Date 10/01/83

Références	Quantité	Prix Un.	Total HT
Recueil de Pom's	1	112.15	112.15
Pom's numéros 7 à 10	1	112.15	112.15
		Total H.T.	224.30
		TVA A 7%	15.70
		TOTAL TTC	240.00
Disquettes recueil	1	126.48	126.48
Disquettes NOS 7-10	1	147.55	147.55
Disquette Visicalc	1	63.24	63.24
Boites 10 Verbatim	2	200.00	400.00
T-Shirts	4	42.16	168.63
		Total H.T.	905.90
		TVA 18,6 %	168.50
		Total TTC	1074.40

Total à payer 1314.40

En votre aimable règlement à réception de facture

Editions MEV Bureaux : 59, Bd de Glatigny 78000 - Versailles ☎ (3) 918-13-07

SARL au capital de 20 000 F — RCS Versailles B 322 815 317 — SIRET 322 815 317 00019

TEXT to Applewriter I

```
10  REM  *** TEXT TO APPLEWRITER ***
20  TEXT : HOME :D$ =  CHR$ (4): ONERR
      GOTO 100
30  DATA  169,1,133,30,169,25,133,31,32,
      12,253,160
40  DATA  0,9,128,201,141,240,12,201,160
      ,144,241,201
50  DATA  224,144,2,233,97,105,64,145,3
      0,230,30,208
60  DATA  227,230,31,208,223: FOR I = 7
      68 TO 808: READ J: POKE I,J: NEXT

70  HTAB 5: INVERSE : PRINT "CONVERSION
      TEXT ==> APPLEWRITER": NORMAL : V
      TAB 6: INPUT "FICHIER A TRANSFERE
      R : ";N$: PRINT
80 A$ = "DU DRIVE : ": GOSUB 130:D1 = D:
       PRINT :A$ = "AU DRIVE : ": GOSUB
       130:D2 = D
90  PRINT D$"OPEN"N$",D"D1: PRINT D$"REA
      D"N$: CALL 768
100  PRINT D$"CLOSE": POKE 216,0: POKE 6
      400,131
110 D =  PEEK (30) +  PEEK (31) * 256: P
      OKE D,96: PRINT : PRINT D$"BSAVET
      EXT."N$",A$1900,L"D - 6399",D"D2
120  PRINT : PRINT "TRANSFORMATION TERMI
      NEE ...": END
130  PRINT A$;: GET A$: IF A$ < "1" OR A
      $ > "2" THEN 130
140 D =  VAL (A$): PRINT D: RETURN
```

Applewriter I to TEXT

```
10  REM  *** APPLEWRITER TO TEXT ***
20  TEXT : HOME :D$ =  CHR$ (4): ONERR
      GOTO 140
30  PRINT "   ";: INVERSE : PRINT "CONVE
    RSION APPLEWRITER ==> TEXT": NORM
    AL
40  VTAB 4: INPUT "FICHIER APPLEWRITER :
      TEXT.";N$: PRINT : PRINT "METTRE
      LA DISQUETTE DANS LE DRIVE 1"
50  VTAB 8: PRINT "APPUYER SUR UNE TOUCH
    E POUR CONTINUER": GET A$: PRINT
    : PRINT D$"BLOADTEXT."N$: VTAB 8:
      CALL  - 868: PRINT "FICHIER "N$"
      CHARGE ..."
60  VTAB 10: INPUT "A TRANSFERER SUR QUE
    L DRIVE ? ";D: IF D < 1 OR D > 2
    THEN 60
70  PRINT : PRINT "ATTENDEZ LE MESSAGE D
    E BONNE CONCLUSION,LE PROCESSUS E
    ST ASSEZ LONG ...":ADR = 6401: PR
    INT D$"OPEN"N$",D"D: PRINT D$"WRI
    TE"N$
80 A =  PEEK (ADR):ADR = ADR + 1
90  IF A > 223 THEN A = A - 64
100  IF A > 191 THEN A = A + 32
110  IF A < 32 THEN A = A + 192
120  IF A < 64 THEN A = A + 128
130  IF A = 96 THEN 150
140  PRINT  CHR$ (A);: GOTO 80
150  PRINT : PRINT D$"CLOSE": PRINT : PR
    INT "TRANSFORMATION TERMINEE ..."
```

Hello corrigé

Thierry Le Tallec et Jacques Tran-Van

Programme HELLO du Pom's n° 6 (page 34)
Récapitulation corrigée

```
¥800D.CFD

800D- 20 2F FB
0810- 20 58 FC 20 89 FE 20 93
0818- FE AD D5 03 C9 9D F0 06
0820- 20 2D FF 4C D0 03 20 51
0828- A8 A9 D3 8D 00 9D A9 9C
0830- 8D 01 9D A2 00 86 19 BD
0838- 04 0C 9D A6 9A E8 D0 F7
0840- A0 9A A9 9B 20 8C 9B A2
0848- 02 86 22 20 58 FC CA 20
0850- 62 0B A0 8C B9 37 1F F0
0858- 0F 0A 90 FB 08 EE 67 9B
0860- D0 03 EE 68 9B 28 D0 F1
0868- 88 D0 E9 2C 83 C0 AD 00
0870- E0 AC 01 E0 2C 81 C0 C9
0878- 4C D0 04 C0 28 F0 08 C9
0880- 20 D0 09 C0 00 D0 05 4C
0888- 03 09 E6 02 A0 EB 84 FB
0890- A5 FB 18 69 23 A8 85 FB
0898- C9 03 F0 EE A2 0B B1 01
08A0- F0 52 DD D8 0B D0 E9 C8
08A8- CA D0 F3 A4 FB 88 88 B1
08B0- 01 8D ED B7 88 B1 01 30
08B8- D7 8D EC B7 BD E4 0B 9D
08C0- AF 05 E8 E0 19 D0 F5 20
08C8- 69 9B 20 19 0B A0 FC A9
08D0- 2F 20 71 9B 20 3E 9B 84
08D8- 0E 84 10 A2 30 86 0F A9
08E0- D0 85 11 B1 0E 91 10 C8
08E8- D0 F9 E6 0F E6 11 CA D0
08F0- F2 20 58 FC A9 10 85 02
08F8- D0 09 A5 06 18 69 23 90
0900- 04 E6 02 A9 0B A8 84 06
0908- A5 00 C9 28 90 03 4C 8F
0910- 09 C9 14 D0 0B 85 20 85
0918- 21 20 58 FC A4 06 A5 00
0920- 0A 0A AA B1 01 30 D3 F0
0928- 66 9D 00 96 E6 00 C8 A5
0930- 02 0A 0A 0A 0A 11 01 9D
0938- 01 96 C8 98 9D 03 96 A9
0940- FF 85 03 B1 01 30 04 46
0948- 03 46 03 29 7F 9D 02 96
0950- A2 07 0A 0A B0 03 CA D0
0958- FA BD A5 0B 25 03 20 87
0960- 09 A5 00 C9 1B B0 02 69
0968- 2C 69 94 20 87 09 A2 0F
0970- C8 B1 01 C9 A0 B0 04 C9
0978- 80 B0 F5 20 ED FD CA D0
0980- EF 20 8E FD 4C FA 08 20
0988- ED FD A9 AD 4C ED FD 20
0990- 2F FB A0 0F 84 24 A9 00
0998- 85 07 20 5B FB B9 AC 0B
09A0- 99 FF 03 88 D0 F7 98 AE
09A8- F6 B7 20 24 ED 20 9C FC
09B0- A9 15 85 24 AE 67 9B AD
09B8- 68 9B 20 24 ED A0 0F B9
09C0- BC 0B 99 18 04 88 10 F7
09C8- 20 2F FB A9 0D 85 24 20
09D0- 9C FC A2 6A AD 89 9B C9
09D8- BF D0 02 A2 60 8E DD 07
09E0- A0 0D B9 CB 0B 99 CF 07
09E8- 88 D0 F7 46 04 46 03 20
09F0- 0C FD C9 A0 D0 03 4C 9A
09F8- 9B C9 AA D0 0C A9 69 A0
0A00- FF 8D 89 9B 8C 8B 9B D0
0A08- 86 C9 DD D0 06 A9 BF A0
0A10- 9D D0 EE C9 95 D0 0B A9
0A18- 02 8D EA B7 8D 68 AA 4C
0A20- 0D 08 C9 88 D0 04 A9 01
0A28- D0 EF C9 9B D0 09 85 07
0A30- A9 3F 8D DD 07 D0 B4 C9
0A38- 8C D0 09 85 03 29 7F 8D
0A40- DD 07 D0 AB C9 84 D0 06
0A48- 85 04 46 03 10 EF C9 AF
0A50- F0 58 C9 B0 90 99 C9 DB
0A58- B0 95 C9 C1 90 02 E9 2C
0A60- E9 95 C5 00 B0 89 0A 0A
0A68- AA BC 00 96 BD 01 96 48
0A70- 29 0F 20 FD 0B 68 4A 4A
0A78- 4A 4A 09 10 85 02 BD 02
0A80- 96 85 0C BD 03 96 85 01
0A88- A0 1E B1 01 99 DC 07 99
0A90- 74 AA 88 D0 F5 24 03 10
0A98- 26 20 B7 0A B1 01 49 80
0AA0- 91 01 A2 02 20 62 0B 4C
0AA8- 0D 08 20 58 FC A2 06 20
0AB0- 94 0B 20 0C FD 30 F0 24
0AB8- 1E 10 6A A2 04 D0 3F 24
0AC0- 04 10 1D 20 B7 0A B1 01
0AC8- 10 04 A2 0A D0 30 A9 75
0AD0- 8D C3 B5 A9 AA 8D C4 B5
0AD8- A2 05 20 94 0B 4C 0D 08
0AE0- A5 0C D0 03 4C C6 A5 C9
0AE8- 04 F0 3B 90 04 A2 0D D0
0AF0- 0D C9 02 F0 31 A9 20 20
0AF8- B2 A5 F0 2A A2 01 E0 04
0B00- 08 20 DD FB 20 80 FE 20
0B08- 02 A7 20 84 FE 20 9C FC
0B10- 20 0C FD 28 F0 91 4C 8F
0B18- 09 20 E3 03 20 B5 B7 90
0B20- 04 A2 08 D0 D9 60 20 69
0B28- 9B 20 19 0B AD 01 97 8D
0B30- 00 98 A8 F0 17 AD 02 97
0B38- 20 FD 0B EE F1 B7 20 19
0B40- 0B A0 00 B9 0C 98 99 00
0B48- 98 C8 D0 F7 AC 0C 97 AD
0B50- 0D 97 20 FD 0B A0 00 A9
0B58- 96 20 71 9B 20 19 0B 4C
0B60- A6 9A 8E F4 B7 86 1E A0
0B68- 00 8C EB B7 84 00 84 01
0B70- A9 10 85 02 20 71 9B A9
0B78- 0F A0 11 20 FD 0B 20 19
0B80- 0B EE F1 B7 CE ED B7 10
0B88- F5 AD ED C0 AD EE C0 10
0B90- 02 85 1E 60 8E BB B5 A9
0B98- 06 8D C1 B5 AD 68 AA 8D
0BA0- C0 B5 4C D6 03 AE BE DD
0BA8- AA D3 D2 DD AA 2D 2D 10
0BB0- 0F 0D 27 13 2D 2D A0 A0
0BB8- D6 CF CC AE A0 D3 C5 C3
0BC0- D4 C5 D5 D2 D3 A0 CC C9
0BC8- C2 D2 C5 D3 16 0F 14 12
0BD0- 05 20 03 00 0F 09 18 A0
0BD8- BA A0 A0 A0 C3 C9 D3 C1
0BE0- C2 D4 CE C9 A8 C3 C8 C1
0BE8- D2 C7 C5 CD C5 CE D4 A0
0BF0- C4 C5 A0 CC A7 C9 CE D4
0BF8- C5 C7 C5 D2 A9 8C EC B7
0C00- 8D ED B7 60 A5 0C 4A B0
0C08- 5B 4A B0 30 AD 00 96 85
0C10- 04 E9 03 A8 AD 01 96 85
0C18- 05 E9 00 20 71 9B 20 3E
0C20- 9B 20 78 9B 10 13 A6 04
0C28- A4 05 20 96 FD AE 02 96
0C30- AD 03 96 20 41 F9 4C BF
0C38- 9D 6C 04 00 6D 00 96 85
0C40- AF B5 69 AD 01 96 69 08
0C48- 85 B0 85 6A A9 FF A8 A9
0C50- 07 20 71 9B 20 3E 9B 85
0C58- D8 8D 00 08 20 78 9B 30
0C60- D5 4C 66 D5 A9 60 8D E7
0C68- 9D 20 9E A5 A9 6C 8D E7
0C70- 9D 20 00 F0 20 D4 A7 38
0C78- ED 00 96 85 CA A9 96 ED
0C80- 01 96 85 CB 38 A5 CA E9
0C88- 02 A8 A5 CB E9 00 20 71
0C90- 9B 20 3E 9B 20 7B 9B 30
0C98- 9D 4C EC EF A0 0C B1 01
0CA0- F0 20 8D EC B7 C8 B1 01
0CA8- 8D ED B7 84 06 A0 E8 A9
0CB0- B7 20 B5 B7 B0 42 A4 06
0CB8- C8 D0 02 E6 02 EE F1 B7
0CC0- D0 DC 4C 5B A7 00 00 A0
0CC8- 00 84 01 A9 97 85 02 8C
0CD0- F0 B7 8D F1 B7 60 20 D4
0CD8- A7 85 48 A9 16 20 5A FC
0CE0- 20 1A FC 20 8E FD A0 BF
0CE8- A9 9D 8C F2 03 8D F3 03
0CF0- 49 A5 8D F4 03 24 07 60
0CF8- 20 78 9B 4C BF 9D

¥
```

Gestion de masques en BASIC

Gérard Michel

Pour commencer, rendons à César ce qui lui appartient ! Les routines présentées ci-après s'inspirent largement des principes développés pour la gestion des masques d'entrée/sortie de la carte MEM/DOS 6 502. Elles ne prétendent pas toutefois, c'est évident, atteindre le même niveau de performance, tant en ce qui concerne la facilité d'utilisation (gestion conjointe des masques et des variables), que le stockage sur disque (recodage des informations) ou la rapidité d'accès (stockage des masques en mémoire RAM après leur ouverture...).

Notre ambition se borne à vous offrir un utilitaire qui, combiné avec la routine d'INPUT publiée dans le numéro 6 de Pom's, vous permette de réaliser une gestion d'écran plus souple et plus complète que par les seules instructions du BASIC.

Routine de gestion de masques

Le principe consiste à « dessiner » le masque directement sur l'écran, puis à sauver la page TEXT correspondante pour un chargement ultérieur à partir de vos programmes. Le programme Applesoft GESMASK vous permet, dans les traitements CRÉATION et MODIFICATION, de réaliser ces « mise en pages ». Quelques touches de fonction ont été prévues pour faciliter ce travail :

- Déplacement vers la droite ou la gauche sans affecter le contenu de l'écran par les touches « → » et « ← ».
- Déplacement vers le bas par RETURN.
- Déplacement vers le haut par CTRL-R.
- Passage en mode INVERSE par CTRL-I.
- Passage en mode NORMAL par CTRL-N.
- Décalage de tout l'écran vers la droite à partir d'un point donné par CTRL-D.
- Décalage de tout l'écran vers la gauche à partir d'un point donné par CTRL-G.
- Répétition verticale d'un caractère par CTRL-V (il faut se placer sur le caractère concerné, puis faire CTRL-V).
- La touche REPT conserve sa fonction habituelle.
- La touche ESC permet de valider l'écran et de le sauver sur disquette (ou de revenir au menu en cas de consultation).

Vous constaterez rapidement qu'il est plus agréable de réaliser par ce moyen des écrans complexes ou « esthétiques » que d'utiliser une suite d'instructions BASIC telles que VTAB, HTAB et autres PRINT...

De plus, si vous perdez de la place sur disquette (chaque écran occupe 6 secteurs), vous en économiserez en mémoire centrale, l'encombrement de vos programmes étant réduit.

Hard-Copy TEXT

Cette routine « standard » vous permettra :

- d'imprimer vos masques afin de repérer l'endroit où seront positionnées les variables (paramètres H et V qu'il faut fournir à la routine d'INPUT).
- d'imprimer des écrans à partir de vos programmes après consultation (voir le programme Applesoft MASK.DEMO).

Impression paramétrée

Pour éditer sur imprimante des données sous une forme quelque peu présentable, il faut bien souvent commencer par définir un tableau à la main. Il faut ensuite jongler avec les SPC, LEFT$... à l'intérieur du programme BASIC pour retrouver le tableau voulu sur l'imprimante.

La solution proposée ici consiste, une fois encore, à dessiner le tableau sur l'écran pour le reproduire ensuite sur papier. Les masques correspondants peuvent être créés par GESMASK.

Chaque ligne d'impression est repérée par deux caractères de « contrôle » identiques entre lesquels elle doit se trouver. Ces caractères peuvent être n'importe quelle lettre de A à Z affichée en INVERSE sur l'écran (ceci impose de ne pas mettre de lettres en INVERSE dans les lignes à imprimer). Deux lignes différentes doivent en outre comporter des

caractères de contrôle différents. Le programme MASK.DEMO fournit une illustration de ce type de traitement (si votre interface/imprimante réalise une impression simultanée à l'écran, rajoutez PRINT CHR$ (9) "80 N" après chaque PR#1 de ce programme).

Mode d'emploi

Le lecteur est prié de se reporter au mode d'emploi décrit dans la routine d'INPUT présentée dans le numéro 6 de Pom's. Il est difficile de la répéter ici, d'autant que le mode d'emploi peut aussi être découvert par tâtonnements ou, dans le pire des cas, par analyse du programme !

Programme GESMASK

```
10  HIMEM: 37340
20 D$ =  CHR$ (4):D1$ =  CHR$ (13) + D$:
       PRINT D$"BLOAD INPUT.OBJ": PRINT
       D$"BLOAD MASKIN.OBJ": PRINT D$"B
       LOAD HCT.OBJ"
30  ONERR  GOTO 9000
40  GOTO 300
70  VTAB 22: HTAB 1: INVERSE : PRINT ZM$
      ;: NORMAL : INPUT " ? ";Z$: VTAB
      22: CALL  - 868:Z$ =  LEFT$ (Z$,1
      ): IF Z$ = "O" OR Z$ = "N" THEN
      RETURN
71  GOTO 70
80  REM
90  REM  ROUTINE D'INPUT
100  REM
149  VTAB V: HTAB H: PRINT  LEFT$ (PO$,L
      O)"  ";: VTAB V: HTAB H: IF ME =
      2 OR X = 1 THEN  PRINT ZY$(LL): I
      F B% = 8 THEN 168
150  VTAB V: POKE 8,TY: POKE 36,H - 1: P
      OKE 7,LO: CALL 37350
154 LC =  PEEK (9):E =  PEEK (6): IF E =
       9 THEN  RETURN
156 ZZ$ = "": PRINT  SPC( LO - LC): IF L
      C = 0 THEN 168
158  FOR ZZ = 1 TO LC:Z =  PEEK (784 + Z
      Z) - 128:ZZ$ = ZZ$ +  CHR$ (Z): N
      EXT
162  IF TY < 3 THEN  RETURN
163  IF  MID$ (ZZ$,2,1) = "/" THEN ZZ$ =
      "0" + ZZ$
164  IF  MID$ (ZZ$,5,1) = "/" THEN ZZ$ =
       LEFT$ (ZZ$,3) + "0" +  RIGHT$ (
      ZZ$,4)
165  IF  LEN (ZZ$) <  > 8 THEN Z$ = ME$(
      3): GOSUB 190: GOTO 149
166 Z4 =  VAL ( MID$ (ZZ$,4,2)): IF  VAL
       ( LEFT$ (ZZ$,2)) > 31 OR Z4 > 12
       OR Z4 <  VAL ( RIGHT$ (ZZ$,2)) <
       = 0 THEN Z$ = ME$(3): GOSUB 190
      : GOTO 149
167  RETURN
168  IF ME = 2 OR X = 1 THEN  VTAB V: HT
      AB H:ZZ$ = ZY$(LL): PRINT ZZ$ SPC
      ( LO -  LEN (ZZ$))
169  RETURN
180  REM
190  VTAB 21: HTAB 1: CALL  - 868: INVER
      SE : FOR Z = 1 TO 150:Z1 =  PEEK
      ( - 16336): NEXT : PRINT Z$;: NOR
      MAL : FOR Z = 1 TO 2500: NEXT : H
      TAB 1: CALL  - 868: RETURN
300  DATA  9,28,1,2,17,36,1,1
310  FOR I = 1 TO 2: READ ZV%(I),ZH%(I),
      ZL%(I),TY%(I): NEXT
320 PO$ = "......................."
400  TEXT : HOME : PRINT D1$"BLOAD GESMA
      SK1,D1": REM  PREMIER MASQUE DU
      PROGRAMME
405 V = 14:H = 28:LO = 1:TY = 2:X = 0:ME
      = 0:B% = 0
410 E = 0: GOSUB 149: IF E = 9 THEN 410
420 Z% =  VAL (ZZ$): IF Z% < 1 OR Z% > 6
      THEN 410
430  IF Z% = 6 THEN  HOME : END
440 V = 17:H = 24:LO = 9:TY = 1:LL = 1
450 E = 0: GOSUB 149: IF E = 9 THEN 405
460 ZY$(1) = ZZ$:M$ = ZZ$:ZM$ = "ENREGIS
      TREMENT CONFIRME": GOSUB 70: IF Z
      $ = "N" THEN X = 1: GOTO 450
470  HOME : PRINT D$"BLOAD GESMASK2":X =
      0:ME = 0:B% = 0: REM  DEUXIEME M
      ASQUE DU PROGRAMME
480 V = 7:H = 29:TY = 2:LO = 1:E = 0: GO
      SUB 149: IF E = 9 THEN 400
490 ZY$(1) = ZZ$:D =  VAL (ZZ$): IF D <
      1 OR D > 2 THEN X = 1: GOTO 480
500  IF Z% <  > 4 THEN 560
510 ME = 0:D% = 2:X = 0
520  FOR LL = D% TO 3:L1 = LL - 1:V = ZV
      %(L1):H = ZH%(L1):LO = ZL%(L1):TY
      = TY%(L1)
525 E = 0: GOSUB 149: IF E = 9 AND LL =
      2 THEN LL = 3: NEXT :X = 1: GOTO
      480
530  IF E = 9 THEN LL = 1:X = 1: GOTO 55
      0
540 ZY$(LL) = ZZ$
550  NEXT :S =  VAL (ZY$(2)):E$ = ZY$(3)
      : IF S < 1 OR S > 5 OR (E$ <  > "
      O" AND E$ <  > "N") THEN D% = 3:X
       = 1: GOTO 520
560  GOSUB 70: IF Z$ = "N" THEN X = 1: G
      OTO 480
570 Z = 0: HOME : PRINT D1$"BLOAD"M$",D"
      D: ON Z% GOTO 600,610,620,630,650
600  IF Z = 0 THEN 9010
602  CALL 37500: REM  ROUTINE DE GESTION
       DE MASQUES
605  POKE 34,24: PRINT D1$"BSAVE"M$",A10
      24,L1024": GOTO 400
610  CALL 37503: GOTO 605: REM  ON SAUTE
       LE "HOME" DE LA ROUTINE
620  POKE 36,0: POKE 37,23: CALL 37537:
      GOTO 400: REM  ON SAUTE DIRECTEME
      NT AU "JSR VTAB" DE LA ROUTINE
630  PRINT D$"PR#"S: IF E$ = "O" THEN  P
      RINT  CHR$ (9)"80N"
640  CALL 37989: PRINT D1$"PR#0": GOTO 4
      00: REM  ROUTINE DE HARD-COPY TEX
      T
650 ZM$ = "ANNULATION CONFIRMEE": GOSUB
      70: IF Z$ = "N" THEN 400
660  PRINT D1$"DELETE"M$: GOTO 400
9000 Z =  PEEK (222): IF Z = 6 AND Z% =
      1 THEN 600
9010  IF Z% = 1 THEN Z$ = "CE MASQUE EXI
      STE DEJA": GOSUB 190: GOTO 400
9020 Z$ = "CE MASQUE N'EXISTE PAS": GOSU
      B 190: GOTO 400
```

Programme MASK.DEMO

```
10  HIMEM: 37340
20 D$ =  CHR$ (4):D1$ =  CHR$ (13) + D$:
       PRINT D$"BLOAD INPUT.OBJ": PRINT
       D$"BLOAD PARA.OBJ": PRINT D$"BLO
       AD HCT.OBJ"
40  GOTO 300
45  REM
46  REM  PLACE A L'ADRESSE $6 LE CARACTE
       RE DE CONTROLE EN INVERSE
50 Z =  ASC (Z$) - 64: POKE 6,Z: CALL 38
       039: RETURN
55  REM
70  VTAB 22: HTAB 1: INVERSE : PRINT ZM$
       ;: NORMAL : INPUT " ? ";Z$: VTAB
       22: CALL  - 868:Z$ =  LEFT$ (Z$,1
       ): IF Z$ = "O" OR Z$ = "N" THEN
       RETURN
71  GOTO 70
130  VTAB 22: HTAB 1: INVERSE : PRINT "'
       RETURN' OU '?' POUR IMPRESSION";:
        NORMAL : GET Z$: IF  ASC (Z$) <
        > 13 AND Z$ <  > "?" THEN 130
140  VTAB 22: HTAB 1: CALL  - 868
142  IF  ASC (Z$) = 13 THEN  RETURN
145  POKE 34,24: PRINT D1$"PR#1": CALL 3
       7989: PRINT D1$"PR#0": RETURN
149  VTAB V: HTAB H: PRINT  LEFT$ (PO$,L
       O);: VTAB V: HTAB H: IF ME = 2 OR
        X = 1 THEN  PRINT ZY$(LL): IF B%
        = 8 THEN 168
150  VTAB V: POKE 8,TY: POKE 36,H - 1: P
       OKE 7,LO: CALL 37350
154 LC =  PEEK (9):E =  PEEK (6): IF E =
        9 THEN  RETURN
156 ZZ$ = "": PRINT  SPC( LO - LC): IF L
       C = 0 THEN 168
158  FOR ZZ = 1 TO LC:Z =  PEEK (784 + Z
       Z) - 128:ZZ$ = ZZ$ +  CHR$ (Z): N
       EXT
162  IF TY < 3 THEN  RETURN
163  IF  MID$ (ZZ$,2,1) = "/" THEN ZZ$ =
       "0" + ZZ$
164  IF  MID$ (ZZ$,5,1) = "/" THEN ZZ$ =
        LEFT$ (ZZ$,3) + "0" +  RIGHT$ (
       ZZ$,4)
165  IF  LEN (ZZ$) <  > 8 THEN Z$ = ME$(
       3): GOSUB 190: GOTO 149
166 Z4 =  VAL ( MID$ (ZZ$,4,2)): IF  VAL
        ( LEFT$ (ZZ$,2)) > 31 OR Z4 > 12
        OR Z4 *  VAL ( RIGHT$ (ZZ$,2)) <
        = 0 THEN Z$ = ME$(3): GOSUB 190
       : GOTO 149
167  RETURN
168  IF ME = 2 OR X = 1 THEN  VTAB V: HT
       AB H:ZZ$ = ZY$(LL): PRINT ZZ$ SPC
       ( LO -  LEN (ZZ$))
169  RETURN
190  VTAB 21: HTAB 1: CALL  - 868: INVER
       SE : FOR Z = 1 TO 150:Z1 =  PEEK
       ( - 16336): NEXT : PRINT Z$;: NOR
       MAL : FOR Z = 1 TO 2500: NEXT : H
       TAB 1: CALL  - 868: RETURN
300 PO$ = "....................."
310  DATA  CREATION,MODIFICATION,CONSULT
       ATION
320  FOR I = 1 TO 3: READ LI$(I): NEXT
400  TEXT : HOME : PRINT D1$"BLOADMASK1"
       :L% = 0
410 V = 22:H = 38:LO = 1:TY = 2:X = 0:ME
       = 0: GOSUB 149: IF E = 9 THEN 41
       0
420 Z% =  VAL (ZZ$): IF Z% < 1 OR Z% > 5
        THEN 410
430  IF Z% = 5 THEN  HOME : END
440  IF Z% = 4 THEN 600
450  HOME : PRINT D1$"BLOAD MASK2": VTAB
        4: HTAB 1: INVERSE : PRINT LI$(Z
       %);: NORMAL
460 B% = 8 * (Z% > 1):X = 1 * (B% = 8):M
       E = 0:D% = 1:TY = 2
470  FOR LL = D% TO 10:L1 = LL / 2:L =
       INT (L1):DH = 1 * (L = L1):V = 9
       + 2 * (L - DH):H = 16 + 15 * DH:L
       O = 9 - 2 * DH
480 E = 0: GOSUB 149: IF E = 9 AND LL =
       1 THEN LL = 10: NEXT : GOTO 400
490  IF E = 9 AND ME = 0 THEN L% = LL
500  IF E = 9 THEN LL = LL - 2:ME = 2: G
       OTO 520
510 ZY$(LL) = ZZ$:ME = 2 * (LL < L% - 1)
520  NEXT :B% = 0:L% = 0: IF Z% = 3 THEN
        GOSUB 130: GOTO 400
530 ZM$ = "ENREGISTREMENT CONFIRME": GOS
       UB 70: IF Z$ = "N" THEN X = 1:D%
       = 10: GOTO 470
540  GOTO 400
600  HOME : PRINT D1$"BLOADMASK3": POKE
       34,24: PRINT D1$"PR#1":Z$ = "A":
       GOSUB 50:Z$ = "B": GOSUB 50:Z$ =
       "A": GOSUB 50:Z$ = "D": GOSUB 50:
        PRINT "";: PRINT D$"PR#0"
610  FOR I = 1 TO 5: VTAB 5: HTAB 13: PR
       INT "RUBRIQUE "I: VTAB 6: HTAB 7:
        PRINT ZY$((2 * I) - 1);: HTAB 27
       : PRINT ZY$(2 * I);
620  PRINT D1$"PR#1":Z$ = "C": GOSUB 50:
        PRINT "";: PRINT D$"PR#0": NEXT
       : PRINT D1$"PR#1":Z$ = "D": GOSUB
        50:Z$ = "A": GOSUB 50: PRINT D1$
       "PR#0": GOTO 400
```

Exemple : masque de MASK.DEMO

```
*************************************************************************
*            INTITULE                *     VALEUR NO 1     *     VALEUR NO 2  *
*************************************************************************
*                                    *                     *                  *
*          RUBRIQUE 1                *       1234.56       *      987.65      *
*          RUBRIQUE 2                *       2233.44       *      999.88      *
*          RUBRIQUE 3                *       1010.20       *      234.56      *
*          RUBRIQUE 4                *       9753.10       *      963.08      *
*          RUBRIQUE 5                *       2323.23       *      111.22      *
*                                    *                     *                  *
*************************************************************************
```

MASKIN.SCE — Lisa 1.5

```
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
;X                               X
;X ROUTINE DE GESTION DE MASQUES X
;X                               X
;X      CODE = MASKIN.OBJ        X
;X                               X
;X                               X
;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
;
        ORG $927C
        OBJ $800
V2      EQU $18
H2      EQU $19
V1      EPZ $8
H1      EPZ $9
M       EPZ $7          ;DRAPEAU POUR INVERSE
C       EPZ $6
ADR     EPZ $28         ;ADRESSE 1ERE COLONNE
H       EPZ $24         ;POSITION HOR. CURSEUR
V       EPZ $25         ;POSITION VER. CURSEUR
VTAB    EQU $FC22       ;DEPLACE CURSEUR EN V
GET     EQU $FD0C       ;ENTREE DE CARACTERE
HOME    EQU $FC58
        JSR HOME
        LDA #0
        STA M
DEB     LDA V
        CMP #$18        ;DERNIERE LIGNE ?
        BNE S0
        LDA #0          ; RETOUR 1ERE COL/1ERE LIGNE
        STA V
        STA H
        JMP S1
S0      LDA H
        CMP #$28        ;DERNIERE COL. DE LA LIGNE ?
        BNE S1
        LDA #0          ;PASSER A 1ERE COL. LIGNE SUIVANTE
        STA H
        INC V
        JMP DEB
S1      JSR VTAB        ;POSITIONNE CURSEUR
        JSR GET         ;ENTREE CARACTERE
        CMP #$9B        ;ESC ?
        BNE S2
        RTS
S2      CMP #$8D        ;RETURN ?
        BNE S3
        INC V           ;OUI => LIGNE SUIVANTE
        JMP DEB
S3      CMP #$95        ; "->" ?
        BNE S4
        INC H           ;OUI => CARACTERE SUIVANT
        JMP S0
S4      CMP #$88        ; "<-" ?
        BNE S5
        LDA H           ;OUI => CARACTERE PRECEDENT
        CMP #0          ; "<-" SUR 1ER CARAC. DE LA LIGNE
        BEQ S6
        DEC H
        JMP S1
S6      LDA V
        CMP #0          ;SI 1ER CARAC./1ERE LIGNE : ON RESTE LA
        BEQ S1
        LDA #$27        ;DERNIER CARAC./LIGNE PRECEDENTE
        STA H
        DEC V
        JMP S1
S5      CMP #$92        ; CTRL-R ?
        BNE S51
        DEC V           ;OUI => LIGNE PRECEDENTE
        BPL S1          ;SI V)=0 ON POSITIONNE EN V
        LDA #$17        ;ON SAUTE A LA DERNIERE LIGNE
        STA V
        JMP S1
S51     CMP #$89        ; CTRL-I ?
        BNE S52
        LDA #1
S53     STA M           ;SIGNALER LE "INVERSE"
        JMP S1
S52     CMP #$8E        ; CTRL-N ?
        BNE S7
        LDA #0          ;SIGNALER LE "NORMAL"
        JMP S53
S7      CMP #$96        ; CTRL-V ?
        BNE S8
        LDA V
        CMP #$17        ;DERNIERE LIGNE => ON IGNORE
        BEQ S1
        JSR $FBC1       ;CALCUL ADRESSE BASE LIGNE V
        LDY H
        LDA (ADR),Y     ;PREND CARAC. H DE LIGNE V
        INC V           ;PASSE A LA LIGNE SUIVANTE
        JSR D1          ;AFFICHE LE CARACTERE
        DEC H           ; REMET LE CURSEUR DANS LA MEME COLONNE
        JMP S0
S8      CMP #$84        ; CTRL-D ?
        BEQ D0
        JMP S9
D0      LDA V
        CMP #$17
        BNE D1
        LDA H
        CMP #$27
        BNE D1
        JMP S1          ;DERNIERE COL/DERNIERE LIGNE : ON IGNORE
D1      LDA V           ;SAUVEGARDE POSITION CURSEUR
        STA V2
        LDA H
        STA H2
        CMP #$27        ;SI DERNIER CARAC. DE LA LIGNE => PRENDRE
;LE PREMIER DE LA SUIVANTE
        BNE D6
        LDA #0
        STA H1
        INC V
        LDA V
        STA V1
        JMP D7
D6      INC H           ;DECALAGE A PARTIR DU CARAC. SUIVANT
        LDA H
        STA H1
        LDA V
        STA V1
D7      LDA #$17        ;ON DECALE LES CARACTERES DE
; L'AVANT DERNIER DE L'ECRAN JUSQU'A H1/V1
        STA V
D3      LDY #$26
D5      JSR $FBC1
        LDA (ADR),Y
        INY
        STA (ADR),Y
        LDA V
        CMP V1
        BNE D2
        CPY H1
        BNE D2
D11     LDA V2          ;ON EST A H1/V1 => RESTITUER H/V ET
; METTRE UN "BLANC" EN H/V
        STA V
        LDA H2
        STA H
        LDA #$A0
        JSR D1
        DEC H
        JMP S1
;
; PROCEDURE EN D2 => COMMENT COPIER LE DERNIER CARACTERE
; D'UNE LIGNE SUR LE PREMIER DE LA SUIVANTE ET REVENIR
; A CETTE LIGNE
;
```

```
151 D2      CPY #1
152         BNE D4
153         DEC V
154         LDA V
155         JSR $FBC1
156         LDY #$27
157         LDA (ADR),Y
158         STA C
159         INC V
160         LDA V
161         JSR $FBC1
162         LDY #0
163         LDA C
164         STA (ADR),Y
165         CPY H1
166         BNE D10
167         LDA V
168         CMP V1
169         BNE D10
170         JMP D11
171 D10     DEC V
172         LDA V
173         JMP D3
174 D4      DEY                 ;DECALE CARACTERE PRECEDENT
175         DEY
176         JMP D5
177 S9      CMP #$87            ; CTRL-G ?
178 ;
179 ; CTRL-G => DECALAGE A GAUCHE
180 ; (MEMES PRINCIPES GENERAUX QUE POUR CTRL-D)
181 ;
182         BEQ G0
183         JMP S10
184 G0      LDA V
185         CMP #$17
186         BNE G1
187         LDA H
188         CMP #$27
189         BNE G1
190         JMP S1
191 G1      LDA V
192         STA V2
193         LDA H
194         STA H2
195 G7      CMP #$27
196         BNE G4
197         INC V
198         LDA #0
199         STA H
200         JMP G2
201 G4      INC H
202 G2      LDA V
203         LDY H
204         JSR $FBC1
205         LDA (ADR),Y
206         STA C
207         CPY #0
208         BNE G3
209         DEC V
210         LDA V
211         JSR $FBC1
212         LDY #$27
213         LDA C
214         STA (ADR),Y
215         INC V
216         LDA #0
217         STA H
218         JMP G5
219 G3      LDA V
220         JSR $FBC1
221         DEY
222         LDA C
223         STA (ADR),Y
224 G5      LDA V
225         CMP #$17
226         BEQ G6
227         LDA H
228         JMP G7
229 G6      LDA H
230         CMP #$27
231         BEQ G8
232         JMP G4
233 G8      LDA #$A0
234         JSR O1
235         LDA V2
236         STA V
237         LDA H2
238         STA H
239         JMP S1
240 S10     JSR OUT
241         JMP S0
242 OUT     LDX M
243 ; M=1 => TRANSFORMER LES CODES-ECRAN DES CARACTERES
244 ; POUR AVOIR LES "INVERSE"
245         CPX #1
246         BNE O1
247         CMP #$C0
248         BCS O2
249         SEC
250         SBC #$80
251         JMP O1
252 O2      SEC
253         SBC #$C0
254 ; PROCEDURE O1 : AFFICHE LE CARACTERE ET AVANCE CURSEUR
255 O1      STA C
256         LDA V
257         JSR $FBC1
258         LDY H
259         LDA C
260         STA (ADR),Y
261         INC H
262         RTS
263         DCM "INT"
264         END
```

HCT.SCE — Lisa 1.5

```
 1 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2 ;X                               X
 3 ;X      HARD-COPY ECRAN TEXT     X
 4 ;X                               X
 5 ;X        CODE = HCT.OBJ         X
 6 ;X                               X
 7 ;X                               X
 8 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 9 ;
10         ORG $9465
11         OBJ $800
12 V       EPZ $25              ; CF IMPRESSION PARAMETREE
13 ADR     EPZ $28
14 PR      EQU $FDED
15         LDA #0
16         STA V
17 HC0     LDY #0
18         JSR $FBC1
19 HC1     LDA (ADR),Y
20 ; CODE-ECRAN >= 40 : CARAC. "NORMAL"
21         CMP #$40
22         BCS HC2
23 ; CODE-ECRAN COMPRIS ENTRE 20 ET 39: CARAC. AUTRE QUE A-Z EN "INVERSE"
24         CMP #$20
25         BCS HC3
26         CLC
27 ; TRANSFORME CARAC. DE "A" A "Z" EN "NORMAL"
28         ADC #$C0
29         JMP HC2
30 HC3     CLC
31 ; TRANSFORME CARAC. DE "SPACE" A "9" EN "NORMAL"
32         ADC #$80
```

```
33 HC2   JSR PR            ; IMPRIMANTE
34       INY               ; CARACTERE SUIVANT
35       CPY #$28          ;DERNIERE COLONNE ?
36       BNE HC1
37       LDA #$8D          ;FIN DE LIGNE D'IMPRESSION
38       JSR PR
39       INC V             ;LIGNE SUIVANTE
40       LDA V
41       CMP #$18          ;DERNIERE LIGNE ?
42       BNE HC0
43       RTS
44       DCM "INT"
45       END
```

PARA.SCE — Lisa 1.5

```
 1 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 2 ;X                               X
 3 ;X    IMPRESSION PARAMETREE      X
 4 ;X                               X
 5 ;X        CODE = PARA.OBJ        X
 6 ;X                               X
 7 ;XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 8 ;
 9       ORG $9497
10       OBJ $800
11 V     EPZ $25           ;POSITION VERTICALE CURSEUR
12 ADR   EPZ $28           ;ADRESSE PREMIERE COLONNE
13 C     EPZ $6            ;CARAC. DE CONTROLE EN "INVERSE"
14 PR    EQU $FDED         ;SORTIE DE CARACTERE
15       LDA #0
16       STA V
17       LDX #0
18 HP4   LDY #0            ; Y=NO DE COLONNE
19       JSR $FBC1         ;CALCUL ADRESSE BASE LIGNE COURANTE
20 HP3   LDA (ADR),Y
21       CMP C             ; CARACTERE DE CONTROLE ?
22       BNE HP1
23       CPX #1
24       BNE HP0
25       LDA #$8D          ;FIN DE LIGNE D'IMPRESSION
26       JSR PR
27       RTS
28 HP0   LDX #1            ;ON A TROUVE LE 1ER CARAC. CONTROLE
29       JMP HP2
30 HP1   CPX #1            ;1ER CARAC. CONTROLE DEJA TROUVE ?
31       BNE HP2
32       JSR PR
33 HP2   INY               ; CARACTERE SUIVANT
34       CPY #$28          ;DERNIERE COLONNE ?
35       BNE HP3
36       INC V             ;LIGNE SUIVANTE
37       LDA V
38       CMP #$18          ;DERNIERE LIGNE ?
39       BNE HP4
40       RTS
41       DCM "INT"
42       END
```

Exemple : masque de GESMASK

```
*****************************************
*                                       *
*          GESTION DE MASQUES           *
*                                       *
*---------------------------------------*
*                                       *
* 1 - CREATION        2 - MODIFICATION  *
*                                       *
* 3 - CONSULTATION    4 - IMPRESSION    *
*                                       *
* 5 - ANNULATION      6 - FIN           *
*                                       *
*                                       *
*           VOTRE CHOIX :               *
*                                       *
*                                       *
*      NOM DU MASQUE : .........        *
*                                       *
*                                       *
*****************************************
```

MASKIN.OBJ

¥927C.9464

```
927C- 20 58 FC A9
9280- 00 85 07 A5 25 C9 18 D0
9288- 09 A9 00 85 25 85 24 4C
9290- A1 92 A5 24 C9 28 D0 09
9298- A9 00 85 24 E6 25 4C 83
92A0- 92 20 22 FC 20 0C FD C9
92A8- 9B D0 01 60 C9 8D D0 05
92B0- E6 25 4C 83 92 C9 95 D0
92B8- 05 E6 24 4C 92 92 C9 88
92C0- D0 1A A5 24 C9 00 F0 05
92C8- C6 24 4C A1 92 A5 25 C9
92D0- 00 F0 CE A9 27 85 24 C6
92D8- 25 4C A1 92 C9 92 D0 0B
92E0- C6 25 10 BD A9 17 85 25
92E8- 4C A1 92 C9 89 D0 07 A9
92F0- 01 85 07 4C A1 92 C9 8E
92F8- D0 05 A9 00 4C F1 92 C9
9300- 96 D0 17 A5 25 C9 17 F0
9308- 98 20 C1 FB A4 24 B1 28
9310- E6 25 20 55 94 C6 24 4C
9318- 92 92 C9 84 F0 03 4C B8
9320- 93 A5 25 C9 17 D0 09 A5
9328- 24 C9 27 D0 03 4C A1 92
9330- A5 25 8D 18 00 A5 24 8D
9338- 19 00 C9 27 D0 0D A9 00
9340- 85 09 E6 25 A5 25 85 08
9348- 4C 55 93 E6 24 A5 24 85
9350- 09 A5 25 85 08 A9 17 85
9358- 25 A0 26 20 C1 FB B1 28
9360- C8 91 28 A5 25 C5 08 D0
9368- 18 C4 09 D0 14 AD 18 00
9370- 85 25 AD 19 00 85 24 A9
9378- A0 20 55 94 C6 24 4C A1
9380- 92 C0 01 D0 2E C6 25 A5
9388- 25 20 C1 FB A0 27 B1 28
9390- 85 06 E6 25 A5 25 20 C1
9398- FB A0 00 A5 06 91 28 C4
93A0- 09 D0 09 A5 25 C5 08 D0
93A8- 03 4C 6D 93 C6 25 A5 25
93B0- 4C 59 93 88 88 4C 5B 93
93B8- C9 87 F0 03 4C 3C 94 A5
93C0- 25 C9 17 D0 09 A5 24 C9
93C8- 27 D0 03 4C A1 92 A5 25
93D0- 8D 18 00 A5 24 8D 19 00
93D8- C9 27 D0 09 E6 25 A9 00
93E0- 85 24 4C E7 93 E6 24 A5
93E8- 25 A4 24 20 C1 FB B1 28
93F0- 85 06 C0 00 D0 16 C6 25
93F8- A5 25 20 C1 FB A0 27 A5
9400- 06 91 28 E6 25 A9 00 85
9408- 24 4C 16 94 A5 25 20 C1
9410- FB 88 A5 06 91 28 A5 25
9418- C9 17 F0 05 A5 24 4C D8
9420- 93 A5 24 C9 27 F0 03 4C
9428- E5 93 A9 A0 20 55 94 AD
9430- 18 00 85 25 AD 19 00 85
9438- 24 4C A1 92 20 42 94 4C
9440- 92 92 A6 07 E0 01 D0 0D
9448- C9 C0 B0 06 38 E9 80 4C
9450- 55 94 38 E9 C0 85 06 A5
9458- 25 20 C1 FB A4 24 A5 06
9460- 91 28 E6 24 60
```

HCT.OBJ

¥9465.9496

```
9465- A9 00 85
9468- 25 A0 00 20 C1 FB B1 28
9470- C9 40 B0 0D C9 20 B0 06
9478- 18 69 C0 4C 81 94 18 69
9480- 80 20 ED FD C8 C0 28 D0
9488- E5 A9 8D 20 ED FD E6 25
9490- A5 25 C9 18 D0 D3 60
```

PARA. OBJ

¥9497.94CB

```
9497- A9
9498- 00 85 25 A2 00 A0 00 20
94A0- C1 FB B1 28 C5 06 D0 0F
94A8- E0 01 D0 06 A9 8D 20 ED
94B0- FD 60 A2 01 4C BE 94 E0
94B8- 01 D0 03 20 ED FD C8 C0
94C0- 28 D0 DF E6 25 A5 25 C9
94C8- 18 D0 D2 60
```

INPUT. OBJ

(Voir Pom's 6 pages 19-22)

¥91E6.926F

```
91E6- A9 80
91E8- A6 07 9D 11 03 CA 10 FA
91F0- A2 00 86 06 20 0C FD C9
91F8- 8D F0 72 E4 07 90 0A C9
9200- 88 F0 28 20 DD FB 4C F4
9208- 91 C9 9B D0 09 E0 00 D0
9210- 05 A9 09 85 06 60 C9 AC
9218- D0 05 A9 AE 4C 63 92 C9
9220- AE F0 40 C9 88 D0 0A E0
9228- 00 F0 06 CA C6 24 4C F4
9230- 91 C9 95 D0 0C A5 25 20
9238- C1 FB A4 24 B1 28 4C 63
9240- 92 C9 A0 90 AF C9 DB B0
9248- AB A4 08 C0 01 F0 0C C9
9250- AF B0 04 C9 AD D0 9D C9
9258- BA B0 99 E0 00 D0 04 C9
9260- A2 F0 91 20 ED FD 9D 11
9268- 03 E8 4C F4 91 86 09 60
```

L'INDISPENSABLE :

Clefs pour l'Apple II
par Nicole Bréaud-Pouliquen
Ce livre de référence est destiné à se trouver en permanence à côté de votre Apple II. Son but est de vous faire accéder rapidement à l'information dont vous avez besoin : syntaxes des commandes, codes machine, adresses utiles. Il comprend également un recueil de 25 "trucs" utiles, les "Comment... ?"
Format : 14,5 × 21
144 pages
92,00 FF / 710,00 FB

Et 11 livres de référence pour votre Apple...

La découverte de l'Applesoft
Tome 1 - par Dominique Schraen et Frédéric Levy
Série verte - *Format : 14 × 21*
128 pages
72,00 FF/ 555,00 FB

La découverte de l'Applesoft
Tome 2 - par Frédéric Levy
Série verte - *Format : 14,5 × 21*
120 pages
72,00 FF/ 555,00 FB

La pratique de l'Apple II
Tome 1
par Nicole Bréaud-Pouliquen
Série bleue - *Format : 14,5 × 21*
128 pages
72,00 FF/ 555,00 FB

La pratique de l'Apple II
Tome 2
par Nicole Bréaud-Pouliquen
Série rouge - *Format : 14,5 × 21*
120 pages
72,00 FF/ 555,00 FB

La pratique de l'Apple II
Tome 3
par Nicole Bréaud-Pouliquen
Série noire - *Format : 14,5 × 21*
176 pages
82,00 FF/ 635,00 FB

Lisp sur Apple II
par Nicole Bréaud-Pouliquen
Série bleue - *Format : 17 × 25*
112 pages
72,00 FF/ 555,00 FB

Visicalc sur Apple
par Hervé Thiriez
Série verte - *Format : 17 × 25*
176 pages
82,00 FF/ 635,00 FB

Mathématiques et statistiques
par Hervé Haut
Série rouge - *Format : 17 × 25*
272 pages
92,00 FF / 710,00 FB

La comptabilité sur Apple II
par Gérard et Serge Lilio
Format : 21 × 28,5
160 pages
102,00 FF / 785,00 FB

32 Basic Programs for Apple II computer
Format 13,5 × 21
280 pages
150,00 FF / 1155,00 FB

Recueil Pom's n° 1
200 pages
120,00 FF / 925,00 FB

AGRAPH

P.S.I.

P.S.I. DIFFUSION
41-51, rue Jacquard
BP 86
77402 Lagny-s/Marne Cedex
FRANCE
Téléphone (6) 007.59.31

P.S.I. BENELUX
5, avenue de la Ferme Rose
1180 Bruxelles
BELGIQUE
Téléphone (2) 345.08.50

Envoyer ce bon accompagné de votre règlement à P.S.I. DIFFUSION ou, pour la Belgique et le Luxembourg, à P.S.I. BENELUX

DESIGNATION	NOMBRE	PRIX
	TOTAL	

par avion ajouter 8 FF (75 FB) par livre.

M. P. 3

NOM _____ PRENOM _____

rue _____ N° _____

Code post. ⌊_⌊_⌊_⌊_⌊_⌋ Ville _____

FID, MUFFIN et DEMUFFIN

Hervé Thiriez

Les programmes FID et MUFFIN se trouvent sur la disquette MASTER du DOS 3.3. Beaucoup de lecteurs nous ont posé des questions à leur sujet et ne semblent donc pas bien maîtriser leur utilisation. Nous allons par conséquent apporter quelques éclaircissements à cet égard. Nous signalons à tout hasard que la brochure du DOS en français explique l'utilisation de ces deux programmes ; le lecteur pourra aussi s'y reporter.

Les fonctions du programme FID sont multiples ; il vous permet de :

- copier des fichiers (1 ou 2 lecteurs) de toute nature, même binaire ou texte
- demander le catalogue
- obtenir le nombre de secteurs libres
- verrouiller (LOCK) ou déverrouiller tout fichier
- effacer (DELETE) tout fichier

Le programme MUFFIN, quand à lui, permet de transformer des fichiers du DOS 3.2 vers le DOS 3.3, y compris pour les fichiers Visicalc.

Ces deux programmes sont agréables sur le plan ergonomique. Ils savent, grâce à l'utilisation de valeurs par défaut, vous éviter des répétitions inutiles. Par exemple, lorsqu'on demande une seconde fois d'affilée la copie de fichiers, le programme ne demande plus la spécification des numéros de slot et drive.

Si l'on possède un seul lecteur, le transfert se fait du slot 6, drive 1 vers lui-même ; FID et MUFFIN se chargent de vous indiquer quand il faut mettre la disquette initiale ou la disquette finale dans le lecteur.

Utilisation du signe « = »

Pour les deux programmes, la même technique d'identification abrégée des fichiers est possible. Quand le nom du fichier à copier ou à adapter est demandé, l'utilisateur peut répondre par le signe « = ».

Ce signe signifie « tous les fichiers » ; le programme demande alors « Do you want prompting ? », c'est-à-dire « Voulez-vous une demande de confirmation à chaque fichier ? ».

Si vous répondez par l'affirmative, le programme vous demande à chaque fichier s'il est concerné ou non par l'opération.

Si vous répondez de façon négative, tous les fichiers de la disquette sont concernés.

Le signe « = » peut aussi être utilisé dans le nom. Ceci permet par exemple de sélectionner seulement les fichiers de texte en Applewriter avec la réponse « TEXT= ». Le reste se passe comme dans le cas où on a répondu par « = ».

Ainsi, en spécifiant un nom tel que « =AL=X= », l'utilisateur sélectionne tous les fichiers comportant dans leur nom « AL » puis « X » dans cet ordre.

Retour au menu principal

Lorsque l'on désire revenir au menu principal, il suffit d'entrer la réponse « Q » (pour QUIT). C'est particulièrement utile pour copier ou transformer un fichier dont on ne se rappelle plus bien le nom.

On utilise alors l'option « = » et le « prompting » pour identifier le fichier ; une fois la copie (FID) ou la transformation (MUFFIN) assurée, il est superflu de répondre « Y » ou « N » à la proposition de chaque fichier ultérieur. Il suffit de répondre « Q » au lieu de « Y » ou « N ».

Rappel du programme

Les deux programmes résident en mémoire jusqu'au moment où un RESET a eu lieu, à moins que des programmes utilisés avant celui-ci n'aient occupé les mêmes emplacements mémoire. On peut par conséquent rappeler les programmes, s'ils n'ont pas été écrasés, par un simple CALL.

CALL 2051 réactive FID ou MUFFIN ; le dernier chargé en mémoire, car les deux utilisent la même zone et ne peuvent par conséquent coexister.

Création de DEMUFFIN

Ceux qui ne possèdent pas le programme SUPER COPY, qui permet de transférer des fichiers dans les deux sens entre le DOS 3.2 et le 3.3, peuvent transformer leur MUFFIN en DEMUFFIN afin de pouvoir adapter leurs fichiers en DOS 3.2. Il y a en effet encore des DOS 3.2 non transformés dans la nature.

La procédure la plus simple est celle proposée par Rob Moore dans la lettre du SNAC (Southern New Hampshire Apple Core) et publiée par CALL A.P.P.L.E. en février 81 :

```
1. BLOAD MUFFIN
2. CALL -151
15AF : B3
15B6 : B2
15F7 : C4 C5
20AA : A9 1E 8D B9 B7 20
20B0 : FD AA 48 A9 BD 8D
20B6 : B9 B7 68 60
1155 : 00 1E
115B : D9 03
1197 : AA 20
```

Cette procédure crée un programme DEMUFFIN DOS 3.3, c'est-à-dire qu'il permet, quand on est en DOS 3.3, de transporter un fichier sur une disquette en DOS 3.2.

Il ne reste plus qu'à revenir en Applesoft et à faire : BSAVE DEMUFFIN 3.3, A$803, L$1900.

Le cours de BASIC Applesoft d'André Finot

Jean-Jacques Crépy

Ayant récemment acquis un Apple dans le but de permettre à mes enfants de s'initier à l'informatique, je me suis renseigné pour savoir quelles étaient les possibilités d'apprendre le BASIC à domicile. J'ai fait cette expérience avec le « Cours de BASIC Applesoft » d'André Finot.

Ce cours se présente sous la forme de trois disquettes accompagnées de deux feuilles de commentaires sur le cours, ses objectifs et son mode d'utilisation. Il ne reste plus qu'à mettre la première disquette dans le lecteur et à lire les écrans successifs, en répondant aux questions éventuelles.

Mes enfants et moi-même avons travaillé avec ce cours de BASIC. Seule l'augmentation de mes activités professionnelles m'a empêché de dépasser la leçon 8 (sur les 26 que compte ce cours en trois disquettes). Voici donc mes premières impressions, complétées par celles de mes fils qui ont été jusqu'au bout du cours.

Ma première impression est favorable : la méthode proposée est simple et permet à quelqu'un n'ayant aucune notion d'informatique de comprendre les premiers rudiments de ce langage.

Ma seconde impression générale est un peu moins bonne, et concerne principalement l'impossibilité d'effectuer des retours en arrière. Il devrait être possible de bloquer le déroulement du programme d'une leçon, *pour mieux en comprendre telle ou* telle partie.

Je regrette personnellement de ne pouvoir me référer à des explications antérieures pour mieux comprendre le programme étudié. Ceci est particulièrement valable lorsqu'il est fait état de courbes et de dessins.

Je pense que, si cette possibilité était offerte, le cours serait très facilement assimilé, en particulier dans ce cas précis.

En ce qui concerne les révisions, l'étudiant a la possibilité de soumettre trois réponses. Si, au terme de la troisième, il n'a pas réussi, la bonne réponse lui est donnée.

Personnellement, j'aimerais que la réponse soit, dans certains cas, mieux explicitée, et qu'il soit éventuellement possible de connaître l'erreur que l'on a commise. En effet, puisqu'un retour en arrière est impossible, on ne peut comparer la mauvaise réponse avec la bonne. Il est possible de tourner cette limitation en inscrivant ses réponses au fur et à mesure sur un morceau de papier, mais cela me paraît peu « informatique ».

En conclusion, je peux dire que je suis très content de ce cours, et qu'il en est de même pour mes enfants, dont l'opinion synthétisée suit.

Jean-Jacques Crépy

Points positifs
- les instructions sont clairement expliquées
- le langage est simple et accessible à tous
- n'importe qui, même très jeune, peut suivre ce cours
- le révisions sont bien faites

Points négatifs
- on n'avance pas très vite
- trop grande « sophistication » de la présentation, qui ralentit le processus
- on est esclave du programme : il faut suivre une leçon dans sa totalité avant de revenir en arrière
- le programme s'adresse à nous comme si l'on avait 8 ans.

Conclusion

Ce programme devrait être enseigné en 6ème ou en 5ème. L'élève peut le suivre seul, le professeur circulant entre les élèves. C'est une initiation parfaite pour un débutant. Il aurait fallu que le programme comporte un tableau récapitulatif des mots réservés du BASIC que l'on puisse sortir sur imprimante.

David, 18 ans

Points positifs
- utilisation aisée
- texte clair, agréable et précis
- présentation agréable
- nombreuses questions, posées 3 fois (en cas d'erreur)
- précision des questions
- temps de réflexion pour les réponses.

Points négatifs
- impossibilité d'interrompre le programme pour faire un retour en arrière
- nombreuses questions trop faciles, qui se répètent
- pas d'explication (avec référence au cours) en cas d'erreur

Suggestions
- réaliser un fichier des termes, de leur signification et de leur mode d'emploi
- fournir un sommaire plus détaillé.

Philippe, 16 ans

Remarques d'André Finot

Je tiens à vous remercier de votre analyse de mon cours de BASIC. La prochaine version, déjà bien avancée, aura les possibilités suivantes :

- arrêts et reprises possibles du cours par page ou demi-page d'écran (existe déjà) ;
- pour un positionnement rapide dans le cours :
  — retour en arrière possibles ;
  — déroulement en avant à grande vitesse ;
- fractionnement des leçons en parties avec indication sur l'écran de l'entrée et de la sortie de telle par*tie ou de telle leçon ;*
- la plupart des programmes, sinon tous, pourront être listés sur imprimante.

D'autre part, je vais tenir compte des points suivants que vous signalez :

- affichage des réponses aux questions ;
- tableau récapitulatif des mots réservés ;
- sommaire plus détaillé ;
- fichier des termes, de leur signification et de leur mode d'emploi.

Multiplan à l'essai

Hervé Thiriez

Etant un utilisateur quasi quotidien de Visicalc, la disponibilité de Multiplan sur Apple II Plus m'a tout de suite excité. Il fallait que je puisse mettre la main sur ce programme afin de l'évaluer. C'est maintenant chose faite, et je dois reconnaître que je ne regrette pas l'expérience !

Initialement disponible seulement sur CP/M, Multiplan fonctionne maintenant sur Apple II Plus 64 K et Apple *IIe*. La version française du programme doit sortir sur le marché en mars/avril 1983.

Le programme fonctionne en 40 colonnes, mais il est conçu initialement pour 80 colonnes. La compatibilité avec les cartes suivantes est prévue : Videx Videotherm, M&R Sup'R'Therm, ALS Smarterm, Vista Vision-80 et Wesper Wizard-80. Ceci dit, ce banc d'essai a été réalisé en 40 colonnes sans difficulté.

Présentation générale

Produit par la célèbre compagnie Microsoft, à laquelle nous devons les meilleurs BASICs sur micro-ordinateur et d'autres réalisations de tous genres (Olympic Decathlon, pour n'en citer qu'une), Multiplan se propose de remplacer Visicalc. Dans ce but, il vise à assurer toutes les fonctions de ce best-seller, et à en ajouter quelques-unes de son cru.

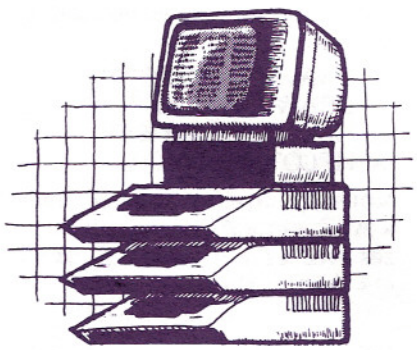
Le lancement est quelque peu différent : le programme est fourni avec une disquette de BOOT protégée et une disquette programme copiable à volonté.

Nous retrouvons au lancement du programme un tableau de 63 lignes et 255 colonnes, à deux différences près :

- l'identification des lignes et des colonnes est entièrement numérique (je ne vois pas ce que cela apporte) ;
- les commandes apparaissent toutes en bas de l'écran, et peuvent être sélectionnées soit par leur première lettre, soit par validation d'un curseur des commandes que l'on déplace aisément.

Il existe aussi une différence non visible au premier coup d'œil, mais importante pour le débutant : l'existence d'une fonction HELP permettant à l'utilisateur de se renseigner à tout moment sur le mode d'emploi de la commande en cours. Bien entendu, il faut pour cela que la disquette programme soit dans le drive 1 ; ce n'est pas vraiment gênant car il y reste assez de place pour mémoriser un très grand tableau.

Comme son grand frère Visicalc, Multiplan laisse l'utilisateur définir le contenu de chaque case, sous forme de libellé, de valeur numérique ou de formule. Comme lui, il recalcule tout le tableau automatiquement ou à la demande. On retrouve dans Multiplan toutes les possibilités de Visicalc ; il y a même (symbole de l'esprit pratique des américains) un programme de conversion des fichiers Visicalc...



Les nouveautés

Les différences sont cependant nombreuses, Multiplan répondant préventivement aux critiques qui avaient été formulées antérieurement à l'égard de Visicalc. Les améliorations sont multiples :

- la largeur des colonnes est variable
- les formules peuvent être globalement verrouillées
- toute case peut être verrouillée
- il est possible de baptiser une case, ou une série de cases : on peut utiliser les noms dans des formules, ce qui est fort agréable
- il y a au maximum huit fenêtres (c'est même trop)
- de nombreux formats sont disponibles
- il existe une commande de tri (SORT)
- les modes d'impression sont beaucoup plus riches
- le recalcul automatique s'effectue de façon réellement automatique (pas besoin d'appuyer n fois sur la touche « ! »).

La fonction de reproduction est assez différente : il n'est plus question de définir, pour les coordonnées des formules, si elles doivent être prises en absolu ou en relatif. C'est au moment de la définition d'une formule que l'on indique la nature des coordonnées.

Ainsi, la formule « +R2C3 +(1.5xRC[−2]) » signifie que l'on utilise le contenu de la ligne 2 et colonne 3 (en absolu), et celui de la case placée deux colonnes à gauche dans la même ligne (en relatif). Si cette formule est reproduite, elle sera reproduite telle quelle, la notation rendant immédiatement relatif ce qui doit l'être.

Enfin, et là se trouve la justification du nom de Multiplan, on peut travailler sur plusieurs tableaux et les lier les uns aux autres, ce qui évite de devoir transférer des informations avec le format DIF. Ainsi, par exemple :

- le tableau principal peut être la consolidation de trois activités ;
- chaque activité possède son tableau principal propre ;
- des tableaux annexes effectuent les calculs secondaires.

Il ne reste plus alors qu'à définir les liens entre les tableaux.

Aspects ergonomiques

Pour un habitué de Visicalc, le premier contact avec Multiplan est frustrant. Pour commencer, le scrolling (déplacement « rapide » dans le tableau) est plus lent ; surtout, il est moins pratique : il faut utiliser les touches CTRL-E, CTRL-S, CTRL-X et CTRL-D (je ne suis pas gaucher, et préfère utiliser un doigt plutôt que deux).

Ce problème n'apparaît pas sur l'Apple *IIe*, qui dispose de quatre touches de déplacement.

Heureusement, on peut aussi se déplacer écran par écran, ou directe-

ment en haut à gauche ou en bas à droite de la partie du tableau utilisée. Enfin, on peut donner un nom (baptiser) une case et s'y positionner directement grâce à ce nom.

Dans un souci de protection du contenu des cases, on ne peut rentrer une valeur ou une formule dans une case sans préciser d'abord « A » (alpha), « V » (value) ou utiliser un symbole mathématique (value). L'expert ès Visicalc court au-devant de surprises désagréables en entrant trop vite des libellés : « ALLOCATIONS » est compris comme le texte (« A » pour alpha) « LLOCATIONS ».

Pour la définition de cases en série, le programme maintient par défaut l'option Alpha/Value si l'on sort d'une case en mode déplacement et non par RETURN.

Enfin, et c'est là le plus gros problème pour le possesseur d'Apple II Plus, il y a deux fois moins de place disponible avec Multiplan qu'avec son grand frère. J'ai converti deux tableaux à partir de Visicalc ; voici les résultats :

	Mémoire occupée	
	Visicalc	Multiplan
Tableau 1	27 %	54 %
Tableau 2	48 %	98 %

En outre, le second tableau se recalculait en 5 secondes avec Visicalc, et 20 secondes avec Multiplan (4 fois plus lent).

Ceci nous montre qu'avec la carte langage et Multiplan, la capacité est similaire à celle de Visicalc sans carte langage. Bien entendu, il me reste avec Multiplan la possibilité de déporter une partie du tableau dans des tableaux annexes que je relierai ensuite au tableau principal : dans ce cas, seules les informations des tableaux annexes devant être transférées à d'autres tableaux actifs « occupent » de la place en mémoire.

Trucs et astuces

Le travail au noir des blancs...

Les blancs entrés au clavier, contrairement à ce que disent les manuels, ont une importance. Essayez d'entrer, puis de lister le programme suivant :

```
10 DRAW 1 AT N,M
20 DRAW 2 ATN,M
30 HPLOT B,A TO C,D
40 HPLOT B,ATO C,D
```

Vous pourrez constater que « ATO » et « ATN » sont tokénizés avec les mots réservés « TO » et « ATN », alors que « AT O » et « AT N » le sont avec « AT ».

Les fanatiques de l'assembleur pourront trouver la raison de tout cela en épluchant la routine Applesoft débutant en $D56D.

Dernière critique : il est obligatoire d'avoir l'imprimante en slot 1. Je trouve ce type de contrainte regrettable : que faire si, pour utiliser un progiciel de création graphique, mon imprimante thermique doit aussi se trouver sur ce slot ? Je ne peux changer mes cartes sans arrêt, les Apples ne sont pas prévus pour cela, on s'en aperçoit vite...

Conclusion

La phase d'adaptation à partir de Visicalc une fois passée, Multiplan est un excellent programme, sous réserve que l'on ne soit pas bloqué par la mémoire. Je n'hésiterai pas à l'acheter avec un Apple *IIe* ou avec un Apple II Plus équipé d'une extension mémoire 32 K ou plus et d'un programme de boot Multiplan sachant exploiter la mémoire supplémentaire.

Si je ne dispose que d'un Apple 64 K, j'utiliserai Multiplan si la nature des tableaux que je suis amené à manier s'accommode du découpage en plusieurs tableaux. Possédant déjà Visicalc, j'estime que Multiplan m'apporte suffisamment pour l'acquérir quand même. Dans ce cas, selon l'application, je sélectionne le programme le mieux adapté, en n'oubliant pas que la possibilité de conversion n'existe aujourd'hui que dans un sens...

Nous venons, dans une société, de développer sur Sirius un plan stratégique à huit ans avec Multiplan. Ce plan concernait la société française et sa filiale américaine, plusieurs activités et des analyses détaillées de la production, des ressources humaines,... Au total, plus de 20 tableaux reliés les uns aux autres. Impressionnant !

Effacement de directory en Pascal

Régis Lardennois

Après avoir passé trois numéros de Pom's à faire l'analyse du directory Pascal avec Michel Crimont, nous allons voir maintenant comment effacer de la mémoire de l'Apple le directory de la dernière disquette lue... A quoi cela sert-il ?

Chaque fois qu'une opération de lecture ou d'écriture est effectuée sur une disquette, son directory est placé en haut de la pile réservé aux variables dynamiques (heap). Il reste en fantôme à cet emplacement, c'est-à-dire invisible (les pointeurs de pile accessibles aux utilisateurs n'en tiennent pas compte) tant qu'il n'y a pas utilisation de cette pile (et sans doute aussi tant qu'il n'y a pas de collision avec le sommet du stack).

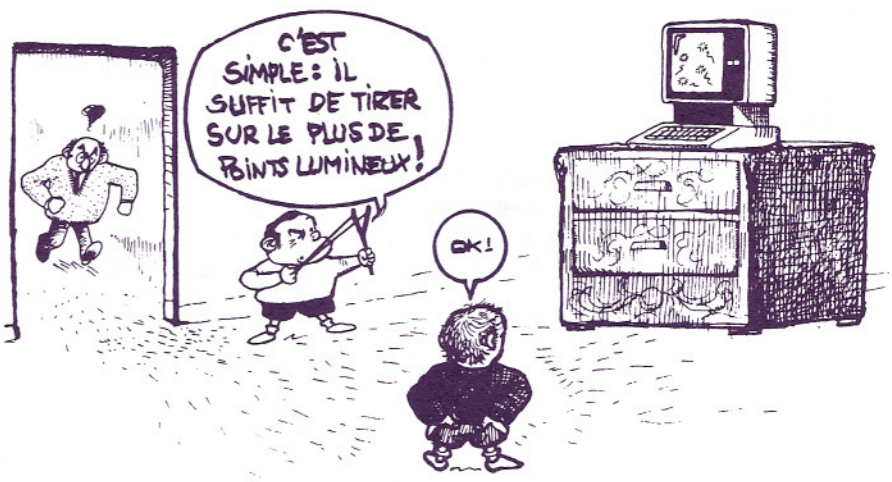
Lors des accès ultérieurs aux disquettes, ce directory est utilisé.

Si un fichier est appelé par un nom de volume (nom de disquette), et que ce nom correspond à celui ainsi placé en mémoire, l'Operating System fait l'hypothèse que la disquette n'a pas été changée, et donc que le directory en mémoire est le bon.

Si un fichier est appelé par un numéro d'unité (#4 pour le drive 1), il lit le nom de volume et compare celui-ci à celui du directory en mémoire. S'il y a identité de nom, il suppose là aussi que l'ensemble du directory est le même.

On comprendra aisément les catastrophes qui peuvent se produire si deux disquettes différentes portant le même nom de volume sont utilisées successivement.

Bien entendu, cela est clairement expliqué dans les manuels Apple II, qui soulignent la nécessité d'éviter l'utilisation de disquettes portant le même nom de volume. Malheureusement, en particulier pour les programmes de gestion, la prudence requiert le recours à des disquettes de sauvegarde qui, étant copiées globalement, portent le même nom !



Le programme introduit ici montre qu'il suffit d'une seule instruction faisant référence au heap pour que l'Apple II n'utilise plus le directory en mémoire.

Mise en œuvre

Tout d'abord, exécuter ce programme. On peut vérifier que, si l'on essaie d'accéder deux fois de suite à un fichier sur la même disquette, l'Apple II ne lit qu'une fois le directory (mise en rotation du drive) tandis que, si la procédure d'effacement a été exécutée, l'Apple II doit aller relire le directory.

Remarques

Pour que la démonstration fonctionne correctement, les disquettes APPLE1 et APPLE2 doivent être réellement présentes ; autrement, il ne peut y avoir de lecture du directory.

Dans le cas de l'Apple *III*, il semble qu'il y ait toujours relecture du directory complet d'une disquette avant lecture ou écriture, et je n'ai pas réussi à provoquer de destruction de disquette (NDLR : à quoi mène l'informatique !).

Par contre, j'ai déjà eu des problèmes en lecture avec des disquettes formatées sur Apple *III*. Je n'ai pas eu le temps d'en approfondir les circonstances, mais ils ont disparu une fois le procédé décrit ci-dessus appliqué.

Il s'agissait d'une non-relecture du directory en lecture avec pourtant des disquettes portant des noms différents, ce qui semble être un bug réel dans l'Opérating System de l'Apple *III*.

Application pratique

Ce programme de test doit être placé sur une disquette dite disquette PROGRAMME durant son exécution et, à la demande, remplacée par une disquette vierge de même nom. Si vous ne demandez pas de lecture du directory de la nouvelle disquette, vous retrouverez en plus du fichier TEST tout le directory de la première disquette PROGRAMME sur la seconde.

Attention : la seconde disquette PROGRAMME sera détruite et la vraie risque de l'être en cas d'erreur de manipulation. Il est donc conseillé de protéger contre l'écriture la vraie disquette PROGRAMME.

```
PROGRAM CLEARDIRECTORY;

VAR  CL     : CHAR;
(*$I-*)
PROCEDURE TESTER1;VAR F : FILE OF INTEGER;
  BEGIN RESET(F,'APPLE1:TEST');CLOSE(F);     { le fichier TEST ne doit  }
                                             { pas exister car autrement}
                                             { il y a toujours relecture}
                                             { du directory             }
  END;

PROCEDURE TESTER2;VAR F : FILE OF INTEGER;
  BEGIN RESET(F,'APPLE2:TEST');CLOSE(F);
  END;

PROCEDURE EFFACER;VAR P0 : ^INTEGER;
  BEGIN MARK(P0);
  END;

BEGIN REPEAT WRITE(CHR(12)); { APPLE II  }
             WRITE(CHR(28)); { APPLE /// }
             WRITELN('VOULEZ-VOUS ? ');
             WRITELN('1 TESTER LA PRESENCE D''UN FICHIER DRIVE 1');
             WRITELN('2 TESTER LA PRESENCE D''UN FICHIER DRIVE 2');
             WRITELN('3 EFFACER LE DIRECTORY EN MEMOIRE');
             WRITELN('esc FIN');
             READ(CL);WRITELN;
             CASE CL OF '1':TESTER1;
                        '2':TESTER2;
                        '3':EFFACER;
             END;
      UNTIL CL=CHR(27);
END.
```

Suite et fin page 9

Création de fichiers EXEC

Yvan Koenig

Le petit programme en BASIC ci-contre a été inspiré par le programme de création de fichier EXEC présenté par Bruno Rives (Pom's 3, pages 55-56 ; Recueil, pages 142-143).

L'inconvénient du programme, tel qu'il était présenté, résidait dans son incapacité à accepter des instructions *comportant des séparateurs*, par exemple POKE A,B.

Le programme ci-contre tourne la difficulté en réalisant un « INPUT ANYTHING », c'est-à-dire qu'il autorise la rentrée de n'importe quelle chaîne. On pourra noter l'instruction 230 dans laquelle l'utilisation du « ON... GOTO » fournit l'équivalent d'un « IF... THEN... ELSE ».

```
1 D$ =  CHR$ (4):R$ =  CHR$ (13):G$ =  C
      HR$ (7):FG$ =  CHR$ (8): TEXT : H
      OME : PRINT "DONNEZ LE NOM DU FIC
      HIER EXEC A CREER": INPUT N$: PRI
      NT D$"OPEN"N$: PRINT : PRINT "DON
      NEZ LES COMMANDES A EXECUTER": PR
      INT "<RETURN> POUR TERMINER"
80  PRINT :C$ = ""
90  GET X$: IF X$ = R$ THEN  ON  LEN (C$
      ) > 0 GOTO 190: GOTO 200
92  IF X$ = FG$ AND  LEN (C$) > 0 THEN C
      $ =  MID$ (C$,1, LEN (C$) - 1): P
      RINT X$" "X$;: GOTO 90
95  IF  ASC (X$) < 32 THEN  PRINT G$;: G
      OTO 90
100 C$ = C$ + X$: PRINT X$;: GOTO 90
190  PRINT : PRINT D$"WRITE"N$: PRINT C$
      : PRINT D$: GOTO 80
200  PRINT : PRINT D$"CLOSE"N$: PRINT :
      PRINT "VOULEZ-VOUS TESTER CE FICH
      IER EXEC (O/N)";
230  GET A$: ON A$ < "N" OR A$ > "O" GOT
      O 230: PRINT : ON A$ = "N" GOTO 2
      60: PRINT : PRINT D$"EXEC"N$
260  END
```

Boot PLE + CRAE

Michel Marquis

Suite à l'article d'Hervé Thiriez dans le Pom's 1, dans lequel il présente les possibilités du PLE (Program Line Editor) et du CRAE (Co-Resident Applesoft Editor), d'aucuns ont pu regretter que l'on doive charger manuellement CRAE en mémoire après avoir « booté » PLE, ce dernier pouvant être chargé automatiquement à partir du HELLO.

Je ne reviendrai pas sur l'intérêt de ces deux programmes ni sur leurs propres commandes, ces dernières étant très largement expliquées dans les articles précités, mais plutôt sur le moyen de les charger en mémoire sans aucune intervention naturelle, si ce n'est la mise sous tension de votre Apple préféré !

Je me suis donc confectionné une disquette qui « boote » directement ces deux utilitaires d'aide à la programmation. Etant enseignant, j'ai pu constater que les stagiaires n'acceptent plus de travailler autrement que sous ce nouvel éditeur PLE + CRAE, surtout s'ils ont peiné quelque peu sur les corrections par ESC I, J...

L'obstacle majeur pour la réalisation de cette association est de supprimer l'effet de l'instruction PRINT CHR$(4)"FP" contenue dans le programme PLE. En effet, il n'est pas question de traiter une quelconque instruction de chargement de CRAE après cela. La méthode que j'ai employée nécessite plusieurs étapes, que je détaille ci-dessous.

Première étape

Initialiser une disquette avec le programme HELLO PLE + CRAE suivant :

```
1  REM  -- PROGRAMME HELLO PLE+CRAE --
10  TEXT : HOME : INVERSE
20  VTAB 5: HTAB 13: PRINT "SUPER EDITEU
    R"
30  VTAB 8: HTAB 14: PRINT "PLE ET CRAE"
40  PRINT  CHR$ (4)"RUN BOOT PLE+CRAE":
    NORMAL
```

```
1  REM   -- PROGRAMME BOOT PLE+CRAE --
10 D$ =  CHR$ (4): PRINT D$;"BRUN PLE":
    REM  LANCEMENT DE PLE
20  CALL  - 22572: REM   INITIALISATION
    ET CONSTRUCTION DES BUFFERS DU DO
    S MODIFIES PAR PLE
30  POKE 1015,135: POKE 1014,0: POKE 101
    3,76: REM  MODIFICATION DU VECTEU
    R & A $8700, ADRESSE DE DEBUT DU
    PLE APRES QU'IL SE SOIT DEPLACE P
    AR LUI-MEME
40  PRINT D$;"BLOAD CRAE": REM  CHARGEME
    NT DE CRAE
50  POKE 111,240: POKE 112,115: REM  MIS
    E A JOUR DES POINTEURS
60  POKE 115,240: POKE 116,115: REM  FRE
    TOP ET HIMEM A LA VALEUR $73F0
70  NEW : REM  NETTOYAGE DE LA PARTIE DE
     LA MEMOIRE NON PROTEGEE
```

Deuxième étape

Charger PLE avec LOAD PLE.FP. Ce programme comporte une partie en BASIC, suivie d'une partie en assembleur.

La fin est identifiée par le pointeur de fin de Program Text PRGEND situé en $AF-$B0 en page Zéro (on trouve normalement $1001).

On peut trouver l'adresse de début en passant en mode moniteur et en examinant la mémoire à partir de $ 800 jusqu'au début des instructions en assembleur :

```
NOP
NOP
NOP
NOP
TXA
PHA
JSR $FF5B
```

Cela se situe normalement, si l'on n'a pas modifié les lignes en BASIC du PLE, à l'adresse $8E0.

Troisième étape

Calculer la longueur du programme en assembleur en effectuant la soustraction (adresse de fin +1) - (adresse de début) = $1001 - $8E0 = $721.

Quatrième étape

Déplacer ce bloc d'instruction en assembleur à l'adresse $3000, en mode (adresse de destination) < (adresse de début). (adr. de fin) M, soit ici : 3000 < 8E0.1001M.

Cinquième étape

Effectuer la sauvegarde sur la disquette initialisée du bloc ainsi déplacé par la commande : BSAVE PLE, A$3000, L$721.

Sixième étape

Entrer le programme ci-dessous, et l'enregistrer sur la disquette sous le nom BOOT PLE + CRAE.

Septième étape

Copier sur la même disquette le programme binaire livré avec CRAE (sur ma disquette, il s'appelait PLE. EDIT.PROD), effectuer la sauvegarde de ce programme avec BSAVE CRAE,A$7400,L$1C00.

Nous rappelons que les adresses AA60/61 et AA72/73 donnent respectivement la longueur et l'adresse de chargement du dernier programme binaire manipulé.

NDLR : il est aussi possible de transférer PLE.EDIT.PROD avec le programme FID fourni sur la disquette MASTER, à condition de le rebaptiser CRAE, puisque c'est sous ce nom qu'il est appelé par le programme décrit dans la sixième étape.

Huitième étape

Il ne vous reste plus qu'à « booter » pour vous trouver après quelques instants de patience en SUPER EDITEUR. N'oubliez pas d'appuyer sur & pour bénéficier des possibilités de CRAE.

Remarque : tout ceci s'applique à la version 48 K de l'Apple II, mais vous n'aurez aucun mal à l'adapter à la capacité mémoire de votre Apple...

Un programme de fondu enchaîné

Denis Sureau

Le programme suivant permet d'effectuer un fondu enchaîné par échange très rapide de deux pages graphiques haute résolution. Il faut tout d'abord charger en mémoire le programme par BLOAD HR TRANS, et les images souhaitées dans les trois pages graphiques débutant en $2000, $4000 et $6000.

Il suffit ensuite de faire un POKE 7,32*M où M est le numéro (1, 2 ou 3) de la première page graphique désirée. Un POKE 9,32*N où N est le numéro de la seconde page graphique désirée complète le travail. CALL 768 lance l'échange des deux pages.

NDLR : une démonstration de ce programme est proposée sur le HELLO de la disquette d'accompagnement de ce numéro de Pom's.

Le programme en BASIC ci-dessous vous permet de créer un effet de fondu enchaîné entre trois images dont vous définissez les noms dans les instructions 100-120.

HR TRANS — Big Mac

```
 1 ********************
 2 *                  *
 3 *  FONDU ENCHAINE  *
 4 *                  *
 5 * AVEC 3 PAGES HR  *
 6 *                  *
 7 *  - D. SUREAU -   *
 8 *                  *
 9 ********************
10
11 MAXI      =     $000C
12 * UTILISE $6,$7,$8,$9
13
14           ORG   $300
15
16           LDA   #$00
17           STA   $6
18           STA   $8
19
20
21 TRANS     LDY   #$00
22           LDA   $7          ;SOMMET
23           CLC               ;DE
24           ADC   #$20        ;PAGE
25           STA   MAXI
26
27 LOOP      LDA   ($6),Y      ;TRANSFERT
28           PHA
29           LDA   ($8),Y
30           STA   ($6),Y
31           PLA
32           STA   ($8),Y
33
34           INY
35           BNE   LOOP
36           INC   $7
37           INC   $9
38           LDA   $7
39           CMP   MAXI
40           BNE   LOOP
41
42           RTS
```

HR TRANS DEMO

```
5  IF  PEEK (10) = 1 THEN 10
7  PRINT  CHR$ (4)"BLOAD HR TRANS"
10  TEXT : HOME
20  COLOR= 10
30  HLIN 10,30 AT 20
40  HLIN 10,30 AT 28
50  VLIN 20,28 AT 10
60  VLIN 20,28 AT 30
70  VTAB 13: HTAB 15: PRINT "DEMO TRANS
      HR"
75  VTAB 20: PRINT "   CHARGEMENT DES LO
      GOS...PATIENCE."
80 D$ =  CHR$ (4) + "BLOAD"
90  IF  PEEK (10) = 1 THEN 150
95  POKE 10,1
100  PRINT D$"Image no. 1,A8192"
110  PRINT D$"Image no. 2,A16384"
120  PRINT D$"Image no. 3,A24576"
150  POKE  - 16304,0: POKE  - 16297,0: P
       OKE  - 16300,0
151  POKE  - 16368,0
200  POKE 7,32: POKE 9,64: CALL 768
250  GOSUB 1000
300  POKE 7,32: POKE 9,96: CALL 768
350  GOSUB 1000
400  POKE 7,32: POKE 9,64: CALL 768
450  GOSUB 1000
500  POKE 7,64: POKE 9,96: CALL 768
550  GOSUB 1000
600  IF  PEEK ( - 16384) > 128 THEN 700
650  GOTO 200
700  TEXT : END
1000  FOR I = 1 TO 50: NEXT : RETURN
```

Récapitulation

```
]CALL -151

*300.326

0300- A9 00 85 06 85 08 A0 00
0308- A5 07 18 69 20 85 0C B1
0310- 06 48 B1 08 91 06 68 91
0318- 08 C8 D0 F3 E6 07 E6 09
0320- A5 07 C5 0C D0 E9 60
*3D0G
```

Les quatre ponts

Olivier Herz

Le petit jeu en Applesoft que voici m'a été inspiré par un de ces nombreux jeux électroniques, Game and Watch en tous genres. Ayant vu ce jeu, je me suis demandé s'il était difficile d'en programmer un : la réponse est « non ».

Le but du jeu est de réussir à empêcher les gens de se noyer. Comme vous pouvez le voir, tout le jeu est écrit en Applesoft, la seule particularité résidant dans des octets constituant une table de formes.

Ceux qui désirent se renseigner sur la création de tabes de formes peuvent lire l'article de Dominique Compère (Pom's 4 pages 19-28, ou Recueil pages 146 et suivantes).

Les quatre ponts

```
]LIST:LES 4 PONTS

10  TEXT : HOME : VTAB 5: INVERSE : PRIN
    T " OLIVIER HERZ PRESENTE: "
20  PRINT : PRINT : NORMAL : PRINT "   L
    E S   Q U A T R E S   P O N T S
     "
30 I = 1:E = 0:F = 0:G1 = 0:G2 = 0:G3 =
    0:G4 = 0:G = 0:A = 0:B = 0:C = 0:
    X = 0:Y = 0:N = 0:K = 0:D = 0:S =
     0:H = 0
40  DIM A(19),B(9),C(9): PRINT : PRINT :
     PRINT "VOUS DEVEZ PLACER VOTRE P
    ONT A L'UN DES QUATRE EMPLACEMENT
    S POSSIBLES"
50  PRINT : PRINT "  -> GRACE AUX TOUCHE
    S   A     K
                Z     M"
60  PRINT "  -> OU GRACE A VOTRE JOYSTIC
    K"
70  PRINT : PRINT "CECI AFIN DE PERMETTR
    E AUX BONSHOMMES   DE PASSER SANS
     ENCOMBRE..."
80  PRINT : PRINT : PRINT "VERSION CLAVI
    ER OU JOYSTICK ? ";
85  GET A$: IF A$ <  > "J" AND A$ <  > "
    C" THEN 85
90  PRINT A$:JO = 0: IF A$ = "J" THEN JO
     = 1
95  IF  PEEK (768) <  > 4 THEN 2000
100  POKE 232,0: POKE 233,3: SCALE= 1: R
    OT= 0
110  HOME : HGR : HCOLOR= 3: HPLOT 0,0:
    CALL 62454
120  HCOLOR= 1: FOR J = 48 TO 80: HPLOT
    0,J TO 279,J: NEXT : FOR J = 128
    TO 160: HPLOT 0,J TO 279,J: NEXT
130  HCOLOR= 3: FOR J = 48 TO 65: GOSUB
    135: NEXT : FOR J = 128 TO 145: G
    OSUB 135: NEXT : HCOLOR= 2: FOR J
     = 66 TO 80: GOSUB 135: NEXT : FO
    R J = 146 TO 160: GOSUB 135: NEXT
     : GOTO 140
135  HPLOT 86,J TO 113,J: HPLOT 168,J TO
     195,J: RETURN
140  REM
160  VTAB 21: HTAB 1: PRINT "SCORE","HI-
    SC","NOYES";: VTAB 22: HTAB 1: PR
    INT S,H,K;
170 G3 =  INT ( RND (1) * 4):G3 = 65 * (
    G3 = 0) + 90 * (G3 = 1) + 75 * (G
    3 = 2) + 77 * (G3 = 3): GOSUB 920
200 I = 3 - I:B = 0:C = 0:D = 0: GOSUB 9
    00: FOR J = 0 TO 19
210  IF A(J) <  > I THEN 250
220 A(J) = 0:C(C) = J:C = C + 1: IF J =
    0 OR J = 10 THEN A = A - 1: GOTO
    250
230 A(J - 1) = I: IF J = 4 OR J = 7 OR J
     = 14 OR J = 17 THEN D = J - 1: G
    OTO 250
240 B(B) = J - 1:B = B + 1
250  NEXT
260 N =  INT ( SQR (S) / 2 + 1): IF A =
    N OR  RND (1) > .5 THEN 300
270 J =  INT (2 *  RND (1)) * 10 + 9: IF
     A(J - 1) = 3 - I OR A(J) = 3 - I
     OR A(6) = I OR A(16) = I THEN 30
    0
280 B(B) = J:B = B + 1:A = A + 1:A(J) =
    I
300  GOSUB 900: IF C = 0 THEN 320
310  HCOLOR= 3: FOR J = 0 TO C - 1:E =
    INT (C(J) / 10):F = C(J) - 10 * E
    : GOSUB 800: DRAW 2 + E AT X,Y: N
    EXT
320  IF B = 0 THEN 340
330  HCOLOR= 0: FOR J = 0 TO B - 1:E =
    INT (B(J) / 10):F = B(J) - 10 * E
    : GOSUB 800: HCOLOR= 0: DRAW 2 +
    E AT X,Y: NEXT
340  IF D = 0 THEN  FOR J = 1 TO 5:E =
    PEEK ( - 16336): NEXT : GOTO 200
350 E =  INT (D / 10):F = D - 10 * E: GO
    SUB 800: IF G = D THEN  HCOLOR= 0
    : DRAW 2 + E AT X,Y:S = S + 1: VT
    AB 22: HTAB 1: PRINT S;: FOR J =
    1 TO 10:E =  PEEK ( - 16336): NEX
    T : GOTO 200
360  GOSUB 800:Y = Y + 12 + (E = 1):X =
    X + 2 - 16 * (E = 1): HCOLOR= 0:
    DRAW 1 AT X,Y: FOR J = 1 TO 100:Q
     =  PEEK ( - 16336): NEXT :A(D) =
     0:A = A - 1
370 K = K + 1: VTAB 22: HTAB 1: PRINT ,,
    K;: IF K = 3 THEN 400
380  HCOLOR= 3: DRAW 1 AT X,Y: FOR J = 1
     TO 250 - 150 * JO: GOSUB 900: NE
    XT : GOTO 200
400  VTAB 23: HTAB 1: PRINT "VOULEZ-VOUS
     REJOUER ? ";
410  POKE  - 16368,0: GET A$: PRINT A$:
    IF A$ = "N" THEN  END
420  FOR J = 0 TO 19:A(J) = 0: NEXT :I =
```

```
        1:A = 0:K = 0: IF S > H THEN H =
        S
430 S = 0: GOTO 110
800  IF E = 1 THEN 820
810 X = 258 - 28 * F:Y = 46: RETURN
820 X = 21 + 28 * F:Y = 125: RETURN
900  IF JO = 1 THEN 1000
905  IF  PEEK ( - 16384) < 128 THEN  RET
      URN
910 G3 =  PEEK ( - 16384) - 128: POKE  -
      16368,0
920  IF G3 = 90 THEN G = 13:G3 = 86:G4 =
      133: GOTO 990
930  IF G3 = 75 THEN G = 3:G3 = 168:G4 =
      53: GOTO 990
940  IF G3 = 77 THEN G = 16:G3 = 168:G4
      = 133: GOTO 990
950  IF G3 = 65 THEN G = 6:G3 = 86:G4 =
      53: GOTO 990
960  RETURN
990  HCOLOR= 3: DRAW 4 AT G1,G2: HCOLOR=
      0: DRAW 4 AT G3,G4:G1 = G3:G2 =
      G4: RETURN
1000 G3 =  PDL (0): IF G3 > 127 THEN 104
      0
1010 G4 =  PDL (1): IF G4 > 127 THEN 103
      0
1020 G = 6:G3 = 86:G4 = 53: GOTO 1070
1030 G = 13:G3 = 86:G4 = 133: GOTO 1070
1040 G4 =  PDL (1): IF G4 > 127 THEN 106
      0
1050 G = 3:G3 = 168:G4 = 53: GOTO 1070
1060 G = 16:G3 = 168:G4 = 133
1070  IF G1 = G3 AND G2 = G4 THEN  RETUR
      N
1080  GOTO 990
2000  GOTO 2020
2010 HEX$ = HEX$ + " ND823G": FOR I = 1
      TO  LEN (HEX$): POKE 511 + I, ASC
       ( MID$ (HEX$,I,1)) + 128: NEXT :
       POKE 72,0: RETURN
2020 HEX$ = "0300: 04 00 0A 00 26 00 5A
      00 8E 00 36 0E 76 76 4D 29 38 3F
      28 0D 2D 38 1F 1F 67 2D 2D 1C 3F
      4C 49 49 30 F6 F6 F6 06 00 24 2D
      2D 2D 25 24 24 E4 5B 38 38 20 0C
      0C 0C 25 24 5F 28 2D 38 3F 27 2D
      2D 1C": GOSUB 2010: CALL  - 144
2030 HEX$ = "0340: 3F 67 2D 4D 89 92 92
      12 17 17 3F 38 38 36 36 36 16 2A
      2D 35 36 36 36 2D 05 00 24 3F 3F
      3F 27 24 24 24 4D 28 28 20 1C 1C
      1C 3F 24 2C 38 3F 28 2D 25 3F 3F
      67 2D E5 39 FF DB 92 92 12 AA 15
      2D 28": GOSUB 2010: CALL  - 144
2040 HEX$ = "0380: 28 36 36 36 36 36 3F
      3F 36 36 36 3E 3F 00 2C 38 2C 6D
      38 3F 27 2D 2D 2D 2D 2D 2D 2D 2D
      2D 2D 2D 2D 2D 35 3F 3F 4E 2D 3E
      0E 36 00": GOSUB 2010: CALL  - 14
      4
2050  GOTO 100
```

Courrier des lecteurs

Un lecteur nous ayant envoyé la petite annonce suivante «Cause double emploi, vends logiciels américains originaux pour Apple II», nous avons répondu comme suit :

Cher lecteur,

Il ne nous est malheureusement pas possible, bien que les annonces soient gratuites dans Pom's, de publier celle que vous nous proposez.

En effet, des esprits mal intentionnés pourraient croire qu'après avoir acquis ces logiciels américains, et les avoir intelligement dupliqués, vous cherchiez à en amortir le coût.

Notre politique éditoriale est très stricte sur le plan éthique ; nous préférons, dans le doute, un excès de sévérité plutôt que de prêter le flanc à d'éventuelles critiques.

Vous demandez à vos lecteurs et à vos abonnés de vous écrire. Personnellement, j'ai longtemps hésité car, il est dur d'avouer, je suis bloqué. S.O.S., Pom's, que dois-je faire ?

Et pourtant, je ne vois que quatre solutions :

1. Me résigner à des programmes rudimentaires.
2. Acheter des logiciels de «confection» sans vraiment comprendre ce qui s'y passe.
3. Etudier, étudier encore, mais dans quels livres trouverai-je ce qui m'intéresse ?
4. Vous écrire, car je dois être le type même du néophyte qui démarre et a besoin d'une revue bienveillante.

J'aimerais trouver une solution à mes problèmes actuels de programmation :

- *dans un affichage long, comment bloquer l'écran à certaines lignes ?*
- *comment trier un tableau qui comporte des données alphanumériques et des valeurs ?*

M. Albert Bazin - 75020 Paris.

Pour bloquer une partie de l'écran, il suffit d'utiliser les POKE's. Ainsi, POKE 34,n bloque les n premières lignes de l'écran, comme vous pouvez le constater facilement. Dans ce cas, HOME ne nettoie plus la partie bloquée. Le haut de l'écran reste bloqué tant que vous n'utilisez pas TEXT ou un autre POKE 34,x (avec x<n).

Un tableau ne peut contenir à la fois des caractères alphanumériques et des valeurs, sauf si celles-ci sont définies comme des alphas. En effet, le tableau porte un nom de type NOM$( ), ou de type NOM( ). Si le tableau est alphanumérique, les valeurs numériques seront classées selon l'ordre alphabétique (celui du dictionnaire) : si par exemple, vous devez classer «2» et «111», 111 passe avant 2 car son premier caractère est «plus petit».

Voici le patch à apporter à HAIFA (Pom's numéro 5) pour obtenir : 1.234.567,89F
avec l'instruction :
& PRINT USR « ---. ---. ---,-- F » ; 1234567.89.

Pour un 48K, il faut opérer en deux étapes successives :

BLOAD HAIFA. CODE1,A$6800.
En moniteur :
7281:46
7285:2C
BSAVE HAIFA.CODE1,A$6800,

L$1000.

BLOAD HAIFA.CODE2,A$7800.

En moniteur :

7964:2E
7979:46
7990:46
7993:2E

BSAVE. HAIFA.CODE2,A$7800, L$1000.

M. Chauvière - 21300 Chenove.

Si, après avoir utilisé le programme HELLO de Thierry Le Tallec et Jacques Tran-Van (Pom's numéro 6), on utilise un programme en assembleur implanté en $300, cela efface la page 3 du DOS. On est alors obligé de rebooter pour parvenir à utiliser le DOS. Y a-t-il une parade ?

André Babeanu - 78350 Jouy-en-Josas.

Nous transmettons la demande aux auteurs.

Il existe aux Pays-Bas une émission sur la chaîne nationale Hilversum 2, le dimanche de 19 h 10 à 19 h 45 sur 747 Khz qui se propose d'émettre en code des programmes d'informatique compatibles avec tous les micros, y compris l'Apple II.

Le système BASICODE est conçu pour 1 200 bauds. Des expériences ont lieu pour le même protocole à 300 bauds, qui doit donner une réception plus fiable.

Pour tous renseignements, contacter BASICODE, Administrative Algemeen Secretariaat, NOS, P.O. Box 10, 1200 JB Hilversum. Pays-Bas.

Thierry Lombry - Rue Tienne-aux-Pierres 94 - B. 5150 Wepion - Belgique.

---

1. *Le patch de la ROM LowerCase est déconnecté par le HELLO de Pom's 6. Je suppose que c'est le module destiné à effacer PLE qui est coupable. Pourriez-vous me dire comment modifier HELLO pour éviter ce désagrément ?*
2. *Je voudrais faire quelques remarques au sujet de SNTX COMPILE (Pom's 6). Contrairement à ce qu'écrit Herz, TASC ne provoque pas l'effacement du programme étudié, mais seulement celui de ses deux premiers octets. Il suffit de mémoriser ces octets, puis de les rétablir. D'ailleurs, il n'y a pas lieu de sauvegarder 104, 104, 175 et 176 puisque la mise en place de COMPILE ne les perturbe pas.*

Si l'on fait tourner le programme ci-joint, on crée un fichier KO : il suffit alors de faire EXEC KO pour lancer SNTX COMPILE et récupérer ensuite le programme étudié. Il semble donc possible de créer un programme compilé selon les indications précédentes.

Il serait souhaitable de compiler de façon à replacer sous la HIMEM de PLE. On pourrait prévoir de placer entre cette HIMEM et SNTX COMPILE les différents octets à mémoriser, à savoir :

- *les deux premiers octets du programme étudié ;*
- *l'indication du slot I de l'imprimante, défini par le fichier EXEC et lu par SNTX ;*
- *ce serait préférable à l'implantation actuelle en 1912, 1913,... qui interfèrent avec l'imprimante.*

3. *Je joins les listings annotés de SNTX et FAIT SYNTAXE permettant d'imprimer sur papier les messages d'erreurs.*
4. *Je vous signale que le programme considère comme une erreur une ligne de DATA telle que : 2000 DATA "Montant : ...". Apparemment, ce sont les « : » qui le perturbent.*

Yvan Koenig - Mosaïque Gerbino Rue du Stade - 06220 Vallauris

---

1. Patch de la ROM LC

Signalons tout d'abord que le menu ne détruit pas en fait PLE : PLE se contente d'abaisser les buffers du DOS et de se loger ainsi entre le DOS et ses buffers, de façon à ne pas être affecté entre autres par la commande FP puisqu'il se trouve de cette façon au-dessus de la HIMEM. Le menu se contente de remettre ces buffers à leur place. Or, le patch d'entrée de caractères de la ROM LowerCase abaisse de $100 ces buffers et la HIMEM par la même occasion. Un autre problème avec ce menu était le JSR SETKBD du début qui fait un IN#0 de façon à déconnecter la routine d'entrée de caractères, ce qui est nécessaire si PLE est présent, mais doit être évité dans le cas présent. Finalement, je propose le patch suivant, qui évite de réassembler le programme de menu, en utilisant une « déviation » par quelques instructions assembleur ajoutées à la fin du code du programme.

```
LOAD MENU
CALL -151
810 :4C FE 0C
CFE : 20 58 FC AE 55 AA E0 28
D06 :D0 0C AE 56 AA E0 9C D0 05
D0F :A9 9B 4C 30 08 4C 13 08
AF  : 17 0D
SAVE MENU LC
```

2. SNTX compilé

Remarquons tout d'abord que le programme que j'ai compilé est légèrement différent du programme SNTX, car il n'y avait pas lieu de retenir les pointeurs 103, 104, 175 et 176. La compilation n'a pas été très facile : j'ai implanté la bibliothèque RUNTIME à partir de $6500. J'ai ensuite mis le programme objet derrière (en décimal 29869). L'ensemble a été calculé de façon à ne pas dépasser l'adresse $9000, afin d'être compatible avec PLE.

Quand j'écrivais que TASC efface le programme, je pensais bien sûr à un effacement au sens des commandes FP ou NEW et non à un effacement physique. En effet, TASC termine son exécution en mettant à zéro les mémoires 800-801-802, à $801 le pointeur 103-104, et à $803 le pointeur 175-176, ainsi que les pointeurs 105-106, 107-108 et 109-110 : il s'agit visiblement d'un NEW (ce n'est pas un FP car, si l'on change la HIMEM, elle n'est pas rétablie). Pour éviter cet ennui, il existe des petits programmes en assembleur permettant de récupérer un programme effacé par NEW ou FP. On peut aussi utiliser le fichier EXEC KO créé par le programme FAIT KO de la page 66 (provenant de votre lettre, après une petite modification).

3. Version imprimante de SNTX

Nous n'avons pas la place de mettre ici la liste complète de SNTX modifié que vous nous envoyez. Voir en page 66 la liste des lignes à modifier dans SNTX pour pouvoir sortir l'analyse sur imprimante.

4. Un bug dans SNTX

En ce qui concerne les instructions DATA mal comprises par SNTX, cela fait partie des 0,1 % à 1 % de cas que je pense avoir oubliés. Vous trouverez en page 66 la liste des modifications à apporter pour que ce problème soit résolu.

Olivier Herz

J'utilise HAIFA en version carte langage. Pouvez-vous me dire comment charger un autre programme de musique en cours de programme

en $D000 ? Faut-il refaire A=PEEK(-16247), une fois, deux fois ?

P.Y. Gouriou 110, avenue Lartigue - 08600 Givet

Pour charger un autre air de musique dans la version 64K de HAIFA (en $D000 dans le BANK 1 de la carte langage), il faut procéder comme HAIFA.EXEC et faire (depuis un programme ou bien au clavier) deux fois A=PEEK(-16247) avant de faire le BLOAD ; il est en outre conseillé de faire A=PEEK (-16254) après ce BLOAD pour remettre la carte langage dans l'état normal en utilisation de l'Applesoft. De même, pour charger une autre table de caractères (en $D000, $D600, $D900 ou $DC00 sur le BANK 2 de la carte langage) il faut faire deux fois A=PEEK(-16255) avant le BLOAD et A=PEEK (-16254) après.

Votre question m'a d'ailleurs fait découvrir une erreur dans le début de l'annexe 1 de l'article sur HAIFA (numéro 5 de Pom's) : j'ai inversé les rôles des BANKS 1 et 2 de la carte langage.

Fait KO

```
]LIST

10 D$ =  CHR$ (13) +  CHR$ (4):N$ = "KO"
20  PRINT D$"OPEN"N$D$"DELETE"N$D$"OPEN"
      N$D$"WRITE"N$
30  PRINT "POKE1912,PEEK(103):POKE1913,P
      EEK(104)"
40  PRINT "POKE1914,PEEK(175):POKE1915,P
      EEK(176)"
50  PRINT "POKE1916,PEEK(PEEK(103)+PEEK(
      104)*256):POKE1917,PEEK(1+PEEK(10
      3)+PEEK(104)*256)"
60  PRINT "BLOAD SNTX COMPILE"
70  PRINT "CALL 29869"
71  PRINT "0"
75  PRINT "POKE103,PEEK(1912):POKE104,PE
      EK(1913):POKE175,PEEK(1914):POKE1
      76,PEEK(1915)
80  PRINT "POKEPEEK(1912)+PEEK(1913)*256
      ,PEEK(1916)"
90  PRINT "POKE1+PEEK(1912)+PEEK(1913)*2
      56,PEEK(1917)"
95  PRINT "DEL1,0"
100  PRINT D$"CLOSE"N$D$"LOCK"N$
```

Fait syntaxe IMP

```
]LIST

30  PRINT "POKE 1914,PEEK(103):POKE 1915
      ,PEEK(104)"
40  PRINT "POKE 1916,PEEK(175):POKE 1917
      ,PEEK(176)"
```

SNTX IMP

```
]LIST

97  PRINT  CHR$ (4)"PR#1" CHR$ (13): POK
      E 1657,80:YK = 1: RETURN
100  IF  NOT ERR GOTO 130
102  IF ERR AND YK = 0 THEN  GOSUB 97
104  PRINT "LIGNE "L1" INSTR "IST" > ";:
      IF ERR = 2 OR ERR = 3 THEN  PRIN
     T B$;" ";
110  IF ERR < 8 AND ERR <  > 0 THEN  PRI
      NT "*****";
5000 ERR = 11: ON YK = 0 GOSUB 97: PRINT
      "LIGNE "L1" INSTR "IST" > "ER$(E
     RR):PTR = NL: GOTO 10000
10003  GOSUB 10:LN = A: GOSUB 10:LN = A
      * 256 + LN:L1 = LN
10006  REM
10010 IST = IST + 1
12000  PRINT : POKE 1657,40: PRINT  CHR$
       (4)"PR#0": PRINT "VOULEZ-VOUS RE
      VENIR AU PROGRAMME INITIAL?";
12020  PRINT A$: IF A$ = "O" THEN  POKE
      103, PEEK (1914): POKE 104, PEEK
      (1915): POKE 175, PEEK (1916): PO
      KE 176, PEEK (1917)
20200 PTR =  PEEK (1914) +  PEEK (1915)
      * 256
```

SNTX BUG

```
]LIST

3300 ERR = 12
3310  GOSUB 10: IF A = 0 OR A = 58 THEN
       100
3320  IF A <  > 34 THEN 3310
3330  GOSUB 10: IF A = 0 THEN 100
3340  IF A = 34 THEN 3310
3350  GOTO 3330
```

Courrier des clubs

L'Association Floppymathique anime un club de micro-informatique sur matériels Apple et Sharp. Ce club s'adresse à des jeunes (à partir de la classe de 4e) et à des adultes.

Niveau : initiation ou perfectionnement.

Pour tous renseignements, téléphoner le jeudi de 16 h à 18 h ou le samedi de 14 h à 18 h au (20) 73.94.80. Sinon, le soir vers 20 h au (20) 89.90.76.

R. Courouble - Association Floppymathique - 54, rue de Lille - 59100 Roubaix.

Le 12 janvier 1983, une vingtaine d'architectes, utilisant professionnellement des Apples II et *III*, se sont réunis à Paris, à la suite de quoi des axes d'intérêt majeur et des actions communes ont été définis.

En outre, la fondation d'un club d'utilisateurs a été décidée. Les architectes équipés d'Apples sont invités à y participer et priés de se faire connaître à l'adresse ci-dessous, afin d'être informés des développements ultérieurs.

Ceux qui ne possèdent pas encore de matériel et qui souhaitent avoir des renseignements peuvent nous demander la liste des participants, afin de contacter le plus proche.

Les producteurs de matériels et de logiciels susceptibles d'être utilisés dans notre profession sont aussi priés de se faire connaître.

Jean-Michel Guillaume - 110, avenue du Général-Leclerc - 75014 Paris.

Robotwar

Appel à tous les utilisateurs de Robotwar. Un concours va être organisé. Prendre contact d'urgence avec Jean-Marc Servat. Tél. : 651.66.61 (de préférence après 19 heures).

L'ALTERNATIVE EUROPÉENNE :

Compatible apple II* MEM/D.O.S

BASIS 108

BASIS 108, 128 K RAM	14.985 F H.T.
Pseudo disque 64 Koctets	incorporé
Z 80 C.P.U. (compatible CP/M*)	incorporé
80 colonnes	incorporé
Minuscules	incorporé
Touches de fonction (15)	incorporé
Clavier numérique	incorporé
Sortie parallèle	incorporé
Entrée/sortie série	incorporé
Support Drive	incorporé
Sortie Vidéo composite couleur	incorporé
Sortie couleur R.V.B.	incorporé
Bloc de mouvements curseur	incorporé
TOTAL	14.985 F H.T.

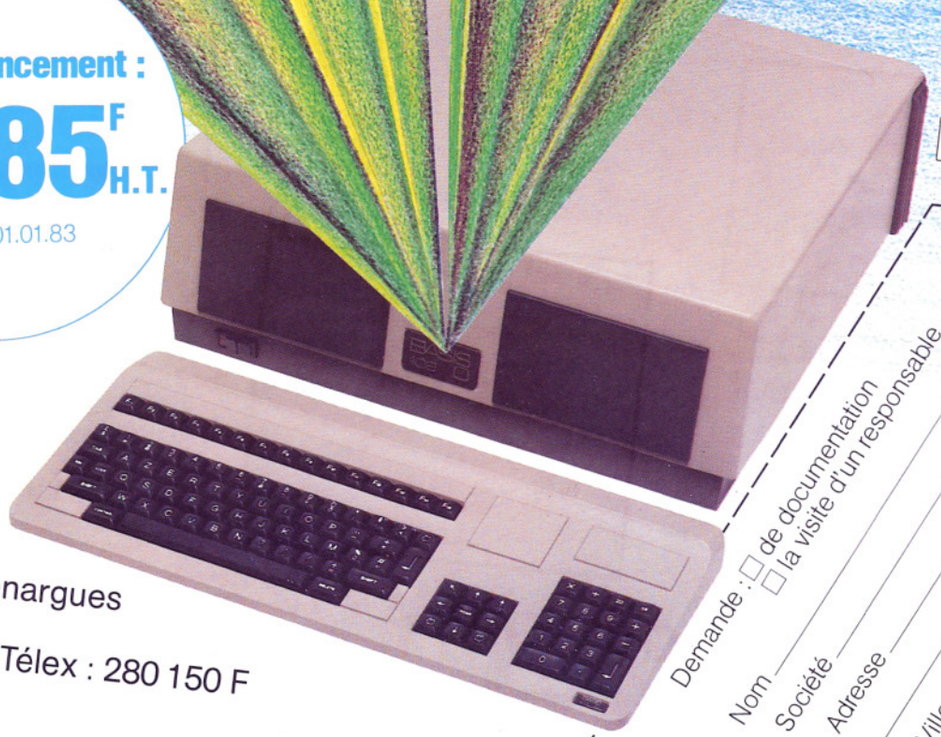
BMI

BOROMÉE MULTISYSTÈME INFORMATIQUE

Offre de lancement :

14985 F H.T.

Tarif au 01.01.83



IMPORTATEUR EXCLUSIF

Demande : ☐ de documentation ☐ la visite d'un responsable

Nom

Société

Adresse

Code postal

Ville

POM'S 7

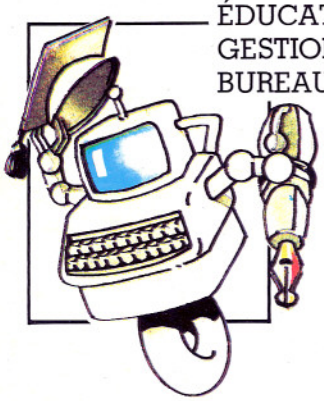
* APPLE marque déposée APPLE computer INC
* CP/M marque déposée digital research INC

BMI
BOROMÉE MULTISYSTÈME INFORMATIQUE

17 bis, rue Vauvenargues
75018 Paris
Tél. : 229 19.74 + Télex : 280 150 F

JCR, DES MICRO-ORDINATEURS
PROFESSIONNEL ET GRAND PUBLIC.

JEUX
ÉDUCATION
GESTION
BUREAUTIQUE



APPLE II
PROMOTION Nous consulter.

Le plus populaire des micro-ordinateurs. 48 K RAM. Basic Applesoft. Une gamme incomparable de logiciels et d'accessoires.

Apple II + 48 K + Disk avec Contrôleur + Moniteur 12".

SHARP PC 1500
Ordinateur de poche de 1,85 Ko de mémoire vive extensible avec module de 8 K CE 155.

CE 150
Mini table traçante 4 couleurs directement connectable sur PC 1500, Interface K 7 incorporé.
PC 1500 + CE 150. **4100F**

CE 158

EPSON HX 20
Un système compact clavier écran LCD avec imprimante. **5900F**

Micro K 7. **1200F**

Extension 16 K. **1300F**

EPSON
Imprimantes de haute qualité d'impression. Interface parallèle type Centronics.

MX 80 FT : 80 cps. ou 132 compressés. **5 800F**

MX 100 : 100 cps. 132 caractères ou 233 compressés. **8200F**

APPLE III
L'outil professionnel par excellence. 128 Ko ou 256 Ko. Unité de disque incorporée. Sortie RS 232. Nombreux interfaces disponibles. Adjonction possible d'un disque dur de 5 méga. Profilé. Écran vert haute résolution antireflets. Clavier Azerty - Qwerty.
Nous consulter.

ENCORE MOINS CHER

COMMODORE VIC 20
Un vrai micro-ordinateur puissant et évolutif idéal pour l'initiation comme pour la pratique de la programmation. 16 couleurs RAM 3,5 K.
Version en PAL. **2350F**

NOUVEAU CHEZ JCR

- ATARI 400 et 800
- APPLE II E
- CASIO PB 100
- SHARP PC 1251
- SHARP PC 1212
- INTERF. RS 232/PC 1500
- VICTOR II 48 K HR

TO 7 THOMSON
Un ordinateur 100% français 8 Ko extensible à 32 Ko. Fourni avec un lecteur optique. Sortie couleur Péritel. Clavier Azerty accentué.
Idéal pour apprendre en famille. **3650F**

Vente par correspondance
Catalogue gratuit sur demande
Crédit 4-36 mois
Leasing 36-48 mois

BOUTIQUE

58, rue Notre-Dame-de-Lorette
75009 PARIS
Tél. (1) 282.19.80 – Télex : 290350 F

59, rue du Docteur Escat
13006 MARSEILLE
Tél. (91) 37.62.33

Horaires d'ouverture du magasin - du mardi au samedi : 10 h - 12 h 45 / 14 h - 19 h.

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