ProCommand

de Glen Bredon

Traduction du manuel

Éditions MEV — Revue Pom's
12, rue d'Anjou
78000 Versailles
© (1) 39 51 24 43
Serveur Minitel (1) 39 53 04 40

Introduction

Le disque PROCMD contient un certain nombre d'utilitaires conçus sous forme de "modules" ajoutés à l'interpréteur BASIC de ProDOS.

Vous pouvez ajouter en mémoire plusieurs ou bien tous les modules. Vous aurez ainsi de nouvelles commandes ProDOS accessibles à partir du clavier ou d'un programme. Par exemple, la commande `TYPE` affichera un fichier texte, disons MONTEXT, en tapant simplement `TYPE MONTEXT` sur le clavier. À partir d'un programme, la syntaxe sera :

```
PRINT CHR$(4)"TYPE MONTEXT"
```

Vous pouvez savoir quelles sont les commandes actuellement actives en tapant "`&`". Ces utilitaires étant dans des modules séparés, vous pouvez ne charger que ceux qui vous sont utiles et donc ne pas occuper inutilement de la mémoire.

La méthode la plus simple pour charger les modules est d'utiliser le programme STARTUP ou bien une variante du programme COMMANDER. Toutefois vous pouvez charger un module, par exemple COPY, à partir du clavier en tapant :

```
CLEAR (ou NEW)
BLOAD COMMANDS/COPY,TCMD,A$4000
CALL 4*4096
```

La première ligne de commande permet d'éviter que la commande BLOAD n'écrase la mémoire utilisée, la deuxième charge le module en mémoire, et la troisième lance une routine de relocation en mémoire qui met le module à sa place finale et le chaîne aux modules déjà chargés.

Le programme STARTUP

Le programme STARTUP est un programme conçu avec des menus qui vous permet de charger une ou bien toutes les commandes annexes disponibles sur le disque. Il vous donnera également la liste des syntaxes des commandes pour les commandes déjà chargées.
Les utilisateurs de disque dur devront changer le prefixe dans la ligne 30 de STARTUP.

Le programme COMMANDER

Le programme COMMANDER est un petit programme BASIC qui charge une liste de modules de commande ProDOS. Les noms des modules sont dans les lignes de DATAs. Vous pouvez supprimer ou ajouter des lignes de DATAs si vous voulez une nouvelle liste de modules.

Vous pouvez également renommer COMMANDER et avoir plusieurs ainsi plusieurs programmes 'chargeurs'. Quand le programme COMMANDER est lancé, la commande `& RETURN` donne la liste de toutes les commandes externes de ProDOS qui sont actuellement disponibles et indique la syntaxe pour les utiliser. Les crochets signifient que la syntaxe est optionnelle. Voici des exemples de syntaxe :

```
EDIT [start, end]["repl"]
AUTO [start, incr]
FIND chaîne, chemin d'accès [,A décalage]
FIND$ chaîne hexa, chemin d'accès [,A décalage]
LYST [entête, E ligne/page,L carac./ligne,à marge]
XREF [£][entête, E ligne/page,L carac./ligne]
```

Les modules double haute résolution donnent des commandes "&" supplémentaires.

À la différence du programme STARTUP, COMMANDER est conçu pour être utilisé sur votre disque pour charger des commandes dès le démarrage. Par exemple, vous avez un disque appelé MONDISK qui contient ProDOS et BASIC.SYSTEM, vous voulez que votre programme STARTUP charge les commandes POINTERS, TYPE et SORT. Utilisez alors le programme PROCMD STARTUP pour charger la commande COPY, et exécutez les commandes :

```
COPY /PROCMD/COMMANDER, /MONDISK/STARTUP
COPY /PROCMD/COMMANDS/POINTERS, /MONDISK/POINTERS
COPY /PROCMD/COMMANDS/TYPE, /MONDISK/TYPE
COPY /PROCMD/COMMANDS/SORT, /MONDISK/SORT
```

Puis changez le programme STARTUP sur votre disque pour qu'il ne contienne que les lignes de DATA pour ces commandes et supprimez le sous-dossier "COMMANDS" de la ligne qui fait le chargement. D'autres modifications simples à mettre en place permettent qu'un programme STARTUP lance un autre programme.

Le programme FITR

FITR est un utilitaire de gestion de fichiers bâti autour de la commande COPY. Il permet d'automatiser la copie, le verrouillage, le déverrouillage et la suppression de fichiers sélectionnés. Il est important de se rappeler que pour copier, FITR a les mêmes avantages et les mêmes inconvénients que la commande COPY, c'est-à-dire que les dates ne seront pas perturbées, mais les fichiers 'sparse' ne seront pas gérés. (En fait, il travaillera sur les fichiers sparses mais la copie ne sera pas sparse).

Note : Les blocs ne comportant que des zéros, dans un fichier 'sparse', ne sont pas physiquement enregistrés sur le disque.

Utilisez les flèches pour mettre l'option voulue en vidéo inverse, puis appuyez sur RETURN. Il vous sera demandé de mettre un préfixe et vous devrez taper le chemin d'accès entier du DIRECTORY (dossier) dans lequel vous voulez entrer. Si cette option n'est pas la première choisie, le dernier préfixe sera alors affiché par défaut, vous n'aurez qu'à appuyer sur RETURN pour valider (dans ce cas vous n'avez pas besoin de retaper le nom). Si vous avez sélectionné l'option COPY FILES, vous devrez préciser le préfixe destination. (Il doit être différent du préfixe source. Pour copier sur un disque qui porte le même nom, il faudra d'abord renommer le disque avec FITR ou bien faire la copie sur /RAM, puis sur l'autre disque).

Ensuite, les dossiers indiqués seront lus sur le disque, et les noms de fichiers seront affichés à l'écran. Seuls les fichiers relatifs à la commande sont affichés. (Cependant, pour les options COPY et DELETE, les fichiers du dossiers ne sont pas affichés).

Utilisez les flèches pour mettre le fichier choisi en vidéo inverse et appuyez sur la barre d'espace pour sélectionner ou désélectionner les fichiers. Ensuite appuyez sur RETURN et les fichiers sélectionnés seront traités. (Les fichiers sélectionnés sont marqués d'une flèche.)

Pour l'option COPY, si le fichier que vous voulez copier existe déjà dans le dossier destinataire, il vous sera alors demandé si vous voulez vraiment le remplacer. Si tel est le cas, et s'il est verrouillé, il vous sera demandé si vous voulez toujours faire cette opération.

Description détaillée des commandes

POINTERS et RESET

Ces deux commandes figurent dans le même fichier Pointers. La commande RESET étant extrêmement pratique, il est recommandé d'inclure systématiquement ce fichier.

La commande POINTERS imprime les valeurs en hexadécimal des sept pointeurs Applesoft suivants :

`TXTTAB` : début du programme BASIC courant.
`VARTAB` : début du stockage des variables (LOMEM).
`ARYTAB` : début des tableaux de variables.
`STREND` : début de la mémoire libre (disponible pour des chaînes).
`FRETOP` : fin de la mémoire libre.
`MEMTOP` : haut du stockage des chaînes (HIMEM).
`PRGEND` : adresse de fin du programme BASIC.

La commande RESET retire toutes les commandes externes ProDOS de la mémoire et remet à zéro la table d'occupation de la mémoire, restituant de la place pour le BASIC. Cette commande peut être nécessaire si vous trouvez que les modules occupent trop de place pour lancer un programme particulier.

ONLINE

Cette commande recherche tous les lecteurs qui sont connectés et affiche les noms de tous les volumes trouvés.

DATE

Cette commande imprime la date et l'heure de ProDOS. Pour en avoir l'impression n'importe où à l'écran ou sur papier, aucun retour chariot n'est généré avant ou après l'impression de la date. En mode immédiat, il est préférable de taper un espace après la commande DATE, car la date sera imprimée sur la même ligne.

Le module DATESTR

Ce module contient une autre version du module DATE. Cette version a le même identificateur que DATE, si bien qu'une seule commande peut être utilisée à la fois. Cette version est un peu plus générale mais utilise deux pages mémoire au lieu d'une. La date est donnée sous forme de chaîne et n'est pas directement imprimée. Elle ne peut être utilisée que dans un programme et non en mode immédiat.
Cette commande peut être utilisée de trois manières différentes :

```
10 PRINT CHR$(4)"DATES":DT$
```

met la date dans DT$ sous la forme : `12-FEB-86 14:05`

```
10 PRINT CHR$(4)"DATE£":DT$
```

met la date dans DT$ sous la forme : `12-FEB-86 2:05 PM`

```
10 PRINT CHR$(4)"DATE%":DT$
```

envoie le jour sous forme de codes Ascii (par exemple : `Wednesday`)

USING

Il s'agit d'un PRINT USING, il ne peut être utilisé que dans un programme. La commande :

```
PRINT CHR$(4)"USING £,£££,£££.££':V
```

imprimera la variable V sous le format £,£££,£££.££, le signe £ correspondant aux positions de chaque chiffre. Si par exemple, V a la valeur 1 234,56, alors cette commande imprimera la chaîne : `1,234.56`

Comme il est indiqué, les premiers zéros sont supprimés et le nombre est justifié à droite. De même, la commande

```
PRINT CHR$(4)"USING £££/£££" :A
```

avec A=12345 imprimera la chaîne : `12/345`

Si le format de la chaîne commence par le signe ^ le caractère suivant sera utilisé pour remplir (au lieu d'utiliser des espaces). D'où :

```
PRINT CHR$(4)"USING ^.£,£££,£££.££":V
```

où V = 1 234,56 imprimera `.....1,234.56`

Si, après l'éventuel caractère de remplissage, le format de la chaîne contient le signe "+", alors la chaîne retournée commencera par "+" ou "-" correspondant au signe du nombre. (Ce signe est justifié à gauche). Par exemple, la commande

```
PRINT CHR(4)"USING +££,£££.££":V
```

pour V = 1 234,56, -1 234,56 , 12,34, -12,34 imprimera les chaînes :

```
+ 1,234.56
- 1,234.56
+    12.34
-    12.34
```

Si une chaîne contient le signe "$", alors la chaîne imprimée sera précédée du signe $. Ce caractère est un caractère flottant dans la chaîne, cela signifie donc qu'il est accolé au premier chiffre. La commande :

```
PRINT CHR$(4)"USING $££,£££.££":V
```

où V = 12.34 sera imrimé : `$12.34`

Si l'option "+" n'est pas sélectionnée, alors le nombre négatif sera imprimé avec le signe - flottant. Toutes les chaînes sont imprimées en laissant un espace pour ce signe, qu'il soit utilisé ou non. D'où :

```
PRINT CHR$(4)"USING £££.££":V
```

pour V = 123,45, -123,45, -12,34 donnera :

```
 123.45
-123.45
 -12.34
```

Pour être pleinement utilisable, la routine ne génère pas de retour chariot avant ou après l'impression de la chaîne formatée. Par exemple :

```
PRINT "......." ;:PRINT CHR$(4)"USING ^.£££££":V
```

avec V = 123, sera imprimée :

```
.........123
```

Des variables chaînes, telles que F$, peuvent être utilisées pour préciser le format d'une chaîne avec la syntaxe :

```
PRINT CHR$(4)"USING";F$:V
```

Quand le format de la chaîne ne laisse pas assez de place (à gauche du point pour les décimales) pour un nombre, ce nombre sera imprimé dans le format Applesoft.
Contrairement à d'autres, cette routine PRINT USING n'est pas limitée à 9 chiffres, mais il est important de noter que seuls 9-10 chiffres seront significatifs. D'où :

```
PRINT CHR$(4)"USING .£££,£££,£££,£££":1/3    donne 0.333,333,333,500
PRINT CHR$(4)"USING .£££,£££,£££,£££":1/300  donne 0.003,333,333,333
PRINT CHR$(4)"USING £££,£££,£££.££":1E9/9    donne 111,111,111.16
```

VARLST

Cette commande imprime la liste de toutes les variables actives. (Elle recherche la véritable table des variables, il ne s'agit pas d'une référence croisée). Utilisez-la après avoir lancé un programme pour voir quelles ont été les variables utilisées par le programme. Vous pouvez également utiliser la commande ProDOS `RESTORE` pour charger une table de variables précédemment sauvegardée avec la commande `STORE`, et utiliser VARLST pour voir ce que contient cette table. Faites d'abord `PR£1` si vous voulez que la liste soit imprimée.

La liste donne la valeur de toutes les variables simples (qui ne sont pas dans un tableau), ainsi que des chaînes. Pour les variables figurant dans un tableau, la liste donne la dimension du tableau mais n'imprime pas les valeurs du tableau. (Cette opération est donc inutile quand vous connaissez les dimensions du tableau). Pour une variable FN (définie avec l'instruction Basic `DEF FN`) la variable factice utilisée dans la définition sera indiquée sous la forme F(X). Si la table a été chargée avac `RESTORE`, alors de telles variables FN sont généralement invalides et ne doivent pas être utilisées. Ce sera indiqué en mettant la variable FN sous la forme F().

VARTRC

C'est une commande de debuggage. La commande `VARTRC` déconnecte le mode VARTRC s'il est actif. (Cette commande est équivalente à RESET). La commande :

```
VARTRC variable
```

(comme par exemple `VARTRC I`) mettra la variable indiquée en mode trace. Si vous lancez un programme Basic, la variable indiquée sera calculée en permanence et affichée en haut de l'écran. (Cela ne fonctionne que sur l'écran TEXT). Le programme n'est pas perturbé – sauf quand il lit directement le haut de l'écran – mais il est ralenti.

Vous pouvez tracer des variables mais aussi toute expression Applesoft valide de 28 caractères au maximum (sous forme tokenisée). Par exemple, vous pouvez avoir envie de voir si la variable I est toujours inférieure à la variable J. Tapez `VARTRC I<J` et la valeur de cette expression logique sera affichée en permanence en haut de l'écran pendant l'exécution du programme. Vous pouvez également tracer les valeurs des variables chaîne telles que `VARTRC A$`. Dans ce cas, la longueur de la chaîne sera tronquée à 40 caractères.

Souvenez-vous qu'il faut taper `VARTRC <RTN>` pour sortir de cet utilitaire. Correctement utilisée, la commande VARTRC peut se révéler très utile pour retrouver des bugs dans un programme Applesoft.

TYPE

La commande `TYPE chemin d'accès` affichera le fichier TXT défini par le chemin d'accès. Le fichier est imprimé 'brutalement' sans formatage.

La commande `TYPE chemin d'accès, E60, L80` imprimera le fichier en indiquant le numéro de page et un en-tête. Le nombre de lignes par page est donné par le paramètre E. Le paramètre L donne le nombre de caractères par ligne, il est optionnel et vaut 80 par défaut. (Il est considéré que l'imprimante fait elle-même les retours-chariot après ce nombre de caractères et qu'elle sait quand se produit le changement de page). Dans ce mode, le caractère tab (control I) sera chargé de tabuler pour se positionner sur la colonne la plus proche qui est un multiple de 8. Le paramètre 'à' donne l'interligne (0 = interlinge simple, 1 = interligne double, etc.) et sa valeur est de 0 par défaut (interligne simple).
La barre d'espacement peut être utilisée pour interrompre le listing et CTRL-C interrompera la lecture du fichier.

DUMP

Il s'agit d'une commande de listage hexa/Ascii. Il y a deux modes : Dump mémoire et Dump fichier. Pour le Dump mémoire, la syntaxe est la suivante :

```
DUMP, A$1000,E$2000
```

Notez que DUMP doit obligatoirement être suivi d'une virgule. Avec cette commande, le contenu de la mémoire de $1000 à $2000 sera listé par groupe de 16 octets en hexadécimal qui seront affichés sur une ligne et suivis de leurs équivalents en codes Ascii. Cette présentation convient pour un affichage 80 colonnes. Le paramètre de longueur (,L8 par exemple) peut être ajouté pour limiter ou étendre la suppression à un nombre précis d'octets par ligne. L8 convient pour un écran 40 colonnes.

Le Dump d'un fichier peut être utilisé avec n'importe quel fichier ProDOS, y compris avec des dossiers. La syntaxe standard est :

```
DUMP /MONVOL/MONFICHIER
```

Le fichier indiqué sera lu et affiché à l'écran (ou imprimé si vous le demandez). Là encore, un paramètre de longueur peut être ajouté à la commande pour préciser le nombre d'octets par ligne, la valeur par défaut étant de 16 .

SORT

C'est l'une des routines de tri les plus générales pour l'Apple //. Ses caractéristiques sont les suivantes :

1 Sa grande rapidité, elle se base sur l'algorithme "Quicksort".
2 Elle trie un tableau de chaînes par ordre alphabétique ou numérique.
3 Elle trie des tableaux de chaînes, des tableaux de réels ou d'entiers.
4 Elle gère des tableaux en plusieurs dimensions, triant sur n'importe quel champ.
5 Elle trie plusieurs tableaux à la fois (en trie un, les autres restant en ligne).
6 Elle trie n'importe quelle zone d'un tableau.
7 Majuscules et minuscules sont traitées indifféremment.
8 Les enregistrements de chaîne vide sont mis à la fin.

Le tableau lui-même n'est pas modifié. Mais les modifications sont faites sur un tableau "index" parrallèle. C'est ce qui rend possible simplement les points 4 et 5.

Utilisation de Sort

La commande SORT est une nouvelle commande ProDOS pouvant être utilisée à partir d'un programme Basic. La syntaxe de la commande est :

```
PRINT CHR$(4)"SORT":A$(0,X),I%(0),F,L
```

Dans ce cas, le tableau à trier est A$ (tableau à deux dimensions dans cet exemple). I% est le tableau d'index associé (le tableau effectivement trié). F est le premier enregistrement de la zone à trier et L est le dernier. X est le champ que vous voulez trier. Si A$ est à une dimension, omettez le "X". S'il est à plusieurs dimensions, X veut dire toutes).

Par exemple, si vous voulez trier un tableau d'entiers à trois dimensions A%(x,y,z) pour x ayant une valeur de 5 à 100 dans la zone, y=2 et z=9, la commande est :

```
PRINT CHR$(4)"SORT" :A%(0,2,9),I%(0),5,100
```

Attention : Le premier 0 de A% et de I% est obligatoire, mais le programme ne le contrôle pas. N'essayez pas de substituer d'autres chiffres pour cette première dimension.

Trier un tableau de chaînes par ordre numérique

Le tri par ordre numérique d'un tableau de chaînes est beaucoup plus lent que le tri par ordre alphabétique, et doit être évité si un autre type de tri peut être utilisé pour obtenir le même résultat. Le processus est relativement lent car les chaînes doivent être converties en valeurs numériques.

Pour trier un tableau de chaînes par ordre numérique, ajoutez ",1" (ou ",V" où V est une variable dont la valeur est 1) à la commande de tri, par exemple :

```
PRINT CHR$(4)"SORT":A$(0),I%(0),0,100,1
```

Vous pouvez également ajouter ce dernier paramètre en lui donnant une valeur différente de 1 pour trier par ordre alphabétique (il est simplement ignoré). Le même type de syntaxe devient ainsi possible pour les deux types de tri.

Tri sur champs

Pour les tableaux à deux dimensions, par exemple A$(x,y), la deuxième dimension y doit être considérée comme précisant un champ. D'où le premier champ est A$(0,0), A$(1,0), A$(2,0)... le second champ est A$(0,1), A$(1,1), A$(2,1)... le troisième champ est A$(0,2), A$(1,2), A$(2,2)...,etc. La première dimension du tableau est l'index de chaque champ et est celle qui sera triée.

Lister un tableau trié

Si le tableau A$(x) a été trié avec le tableau d'index I%(x) grâce à cette commande par exemple :

```
PRINT CHR$(4)"SORT":A$(0),I%(0),0,S
```

alors le tableau peut être listé trié ainsi :

```
100 FOR X = 0 TO S : PRINT A$(I%(X)) : NEXT
```

Dimensionner les tableaux

Le tableau index I% doit être dimensionné de la même façon que le premier tableau à trier, ou au moins comme l'index le plus haut de la série à trier. Par exemple, si le tableau à trier est A$(x,y) et le tableau d'index I%(x), la première dimension de A$ étant 1000, la seconde étant 2, pour dimensionner faites :

```
10 DIM A$(1000,2), I%(1000)
```

Initialiser le tableau d'index

Vous n'avez pas besoin d'initialiser le tableau d'index. La routine regarde si les deux premiers éléments (I%(0) et I%(I)) du tableau d'index sont à zéro, et dans ce cas initialise très rapidement tout le tableau. Donc si vous voulez forcer la réinitialisation, il vous suffit de mettre les deux premiers éléments à zéro. L'initialisation doit être faite si le tableau peut contenir des valeurs autres que des entiers 0, 1, 2, 3, 4... et si l'un des deux premiers éléments n'est pas un zéro. (Si le tableau n'est utilisé que pour le tri, l'initialisation n'est jamais une nécessité).

Tri secondaire

La raison pour laquelle la routine n'initialise pas le tableau quand ceci a déjà été fait, est qu'il est possible de faire un deuxième tri sur un champ (ou un autre tableau), sur une partie d'un tableau déjà trié. Par exemple, supposez que vous ayez une liste de 1 000 noms dans le tableau A$(x,y), le premier champ A$(x,0) contient les noms et le deuxième A$(x,l) les prénoms. Supposez que vous triez les noms avec :

```
PRINT CHR$(4)"SORT" :A$(0,0),I%(0),0,1000
```

Si, en lisant les enregistrements triés, vous trouvez que les numéros 5 à 13 ont le même nom, vous pouvez faire un deuxième tri de ceux-ci par prénoms avec la commande :

```
PRINT CHR$(4)"SORT":A$(0,1),I%(0),5,13
```

Les lignes 570 à 630 du programme DEMO/SORT illustrent cette technique.

Trier plusieurs tableaux

Grâce au système d'indexation, le tri de plusieurs tableaux est simple. Supposez que le tableau L$(x) est un tableau de noms et F(x) un tableau de prénoms associés. Pour trier les enregistrements de 1 à 500 par noms, faites :

```
PRINT CHR$(4)"SORT":L$(0),I%(0),1,500
```

Le programme :

```
100 FOR X = 1 TO 500 : PRINT L$(I%(X));", ";F$(I%(X)) : NEXT
```
lira les enregistrements triés par noms. Si vous le souhaitez, un nouveau tri peut être fait sur les prénoms pour les homonymes.

Noms de tableau

Nos exemples ont tous utilisé I% comme nom de tableau d'index. Ce n'est pas une obligation et n'importe quel nom peut convenir. Le tableau d'indexation doit, toutefois, être un tableau d'entiers. De même, les tableaux à trier peuvent porter n'importe quel nom.

Interrompre un tri

Vous ne devez jamais utiliser la touche RESET pour interrompre le processus de tri. Des paramètres Applesoft se trouveraient modifiés dans la page zéro. La routine de tri vérifie régulièrement que vous n'avez pas appuyé sur CTRL-C et grâce à cette commande, l'interruption se fera correctement. Si la touche RESET est pressée par erreur, faites :
```
NORMAL : NOTRACE : SPEED = 255
```
ce qui devrait tout remettre en ordre. Si vous listez votre programme avant de faire la commande NORMAL, il ne ressemblera pas à ce qu'il devrait être, mais en fait il n'aura pas été endommagé.

Le programme DEMO/SORT

Ce programme illustre la plupart des possibilités de la routine de tri. Il fait en particulier un tri de tableau de chaînes à deux dimensions sur quatre champs. Un champ est trié par ordre numérique. Il lui est également associé un tableau de réels et un tableau d'entiers.

La ligne 50 donne la taille standard S de la première dimension de tous ces tableaux. S est mis à 17 si bien que cette nouvelle présentation permet de tout voir en une seule fois sur un écran 80 colonnes. Si vous mettez S = 500 à la ligne 50, vous pouvez vérifier l'efficacité sur un grand nombre de tableaux, jusqu'à ce que la mémoire soit quasiment pleine. Les données de vos programmes ne sont sûrement pas de cette longueur et le tri s'en trouvera donc beaucoup plus court pour des applications standard. Même sur ce très grand nombre de tableaux, le tri ne prend que quelques secondes.

COPY

La commande :
```
COPY CHEMIN1, CHEMIN2
```
copie le fichier CHEMIN1 sur le fichier CHEMIN2 (en général sur un autre disque). Il sera réécrit par dessus le deuxième fichier si celui-ci existe déjà, sauf s'il est verrouillé.

Cet utilitaire a été écrit pour deux raisons. La première est qu'il évite d'aller dans un programme de copie et de revenir, et il ne détruit pas le programme que vous pouvez avoir en mémoire. Deuxièmement, il donne une copie littérale, à la différence de FILER et des programmes équivalents, c'est-à-dire qu'il conserve la date de création et la date de modification du fichier original.
Il ne fonctionnera pas sur les fichiers "sparse" ni sur les dossiers. (Créer d'abord le dossier destinataire et y copier les fichiers du dossier source).

La routine COPY utilise la place libre existant entre la table des variables et les chaînes. Donc plus cet espace est grand, plus elle sera à l'aise. Faites CLEAR avant d'effectuer une copie, ou, si vous n'avez pas besoin de conserver le programme en cours en mémoire, faites NEW.
Puisque les paramètres des lecteurs ne sont pas acceptés par la commande, les noms des volumes doivent obligatoirement être différents. Pour effectuer une copie entre deux disques qui portent le même nom, vous pouvez renommer l'un des volumes, faire la copie, puis ensuite le renommer. Vous pouvez également copier le fichier sur le /RAM, changer de disque, et recopier sur le nouveau disque (sur un Apple //c ou un Apple //e 128Ko).

Renumber

La commande RENUMBER accepte quatre paramètres — S, I, F, L — dans n'importe quel ordre. Ils sont tous optionnels avec les valeurs par défaut suivantes :

```
S10, I10, F0, L65535
```

Voici la signification de ces paramètres :

- S Numéro de la ligne de début du programme renuméroté.
- I Incrémentation des numéros de lignes.
- F Numéro de la première ligne à renuméroter.
- L Dernier numéro de ligne à renuméroter.

Par exemple, la commande :

```
RENUMBER S100, I5O0, F2000, L2999
```

renumérotera les lignes 2000 à 2999 en commençant par le nouveau numéro de ligne 100 avec un incrément de 50 entre chaque ligne renumérotée.

Vous pouvez ne pas taper les lettres pour les paramètres S, I, F, L si les paramètres sont donnés dans l'ordre indiqué. Si les lettres sont tapées, les paramètres peuvent être entrés dans n'importe quel ordre.

Cette routine de renumérotation permettra de numéroter au-delà de la limite habituelle de 63999. Ceci permet la création de remarques qui seront difficiles à supprimer, (mais il faut se rappeler que les GOSUB et les GOTO sont illégaux dans de telles lignes). Par exemple, si vous écrivez des lignes telles que :

```
10 REM ******************
20 REM *                  *
30 REM * COPYRIGHT 1985 *
40 REM *                  *
50 REM ******************
```

...et que vous faites `RENUMBER S65531,I1`, le résultat sera

```
65531 REM ******************
66632 REM *                  *
65533 REM * COPYRIGHT 1985 *
65534 REM *                  *
65535 REM ******************
```

Ces lignes ne peuvent pas être facilement supprimées ou modifiées par les utilisateurs.

La renumérotation fonctionnera sur des programmes qui remplissent presque toute la mémoire. De nombreux programmes de renumérotation ne fonctionnent pas toujours correctement avec certaines instructions qui utilisent des numéros de ligne, telles que LIST, DELETE et RUN. Ce RENUMBER fonctionne même avec des syntaxes ésotériques telles que `LIST 10-20`.

Si une ligne référencée n'existe pas dans le programme, RENUMBER utilisera la ligne suivante qu'il trouvera.

Plusieurs messages d'erreur peuvent être générés par le programme :

`OVERFLOW` : un nouveau numéro de ligne dépasserait le chiffre limite de 65535.

`LINE £ CONFLICT` : un nouveau numéro de ligne serait un doublon d'un numéro déjà existant (non modifié), ou un enchevêtrement de numéros de ligne serait causé par la renumérotation.

`OUT OF MEMORY` : le programme est trop grand. (Ceci est très rare car RENUMBER nécessite fort peu de place pour fonctionner).

`PROGRAM INVALID` : soit une erreur a été découverte dans le programme, soit du fait de la renumérotation une ligne dépasse la limite d'une page.

Ces deux dernières occurences sont très rares.

HOLD (dans le fichier RENUMBER)

Cette commande protège en mémoire le programme courant dans le but de le fusionner avec un autre grâce à la commande MERGE. Cela fonctionne en déplaçant le pointeur TXTTAB au-dessus du programme. Si la commande POINTERS est active, vous pouvez l'utiliser pour savoir s'il y a ou non un fichier actuellement conservé en mémoire. (En général, TXTTAB = $801). Si vous faites la commande `HOLD` pendant qu'un autre fichier est utilisé, le fichier actuel sera fusionné au programme conservé en mémoire et le résultat sera un nouveau fichier conservé en mémoire. Cela ne peut fonctionner que si les numéros de ligne d'un programme sont tous supérieurs à ceux de l'autre. Si non, vous obtenez le message `LINE £ CONFLICT, OK ?` (problème de numéros de ligne, d'accord ?) et vous pouvez appuyez sur `Y` pour obliger la fusion dans le nouveau fichier protégé.

MERGE (dans le fichier RENUMBER)

Cette commande fusionne le programme conservé en mémoire (par HOLD), s'il y en a un, au programme en cours. Comme avec HOLD, l'enchevêtrement des numéros de lignes ne sera possible que si vous le demandez. Si des numéros de ligne se retrouvent en double, ils seront fusionnés dans le fichier sous le même numéro de ligne. Ils peuvent être séparés avec la commande `RENUMBER`.

Puisque les lignes du programme doivent être triées par MERGE, l'exécution de cette commande peut prendre un certain temps.

Important : Il ne faut jamais appuyer sur RESET quand les commandes RENUMBER HOLD et MERGE sont en cours. Votre programe serait perdu !

XREF

La commande XREF imprimera une table de références croisées du programme Applesoft en mémoire. Il admet le paramètre E pour le nombre de lignes par page. Cette option est conçue pour l'impression papier de la table XREF. XREF admet également le paramètre L qui donne le nombre de caractères par ligne. La valeur par défaut est 80. Pour un écran en 40 colonnes, utilisez :

```
XREF,L40
```

Pour imprimer la table avec une interface dans le port 1 :

```
PR£1
XREF,E60
```

La table est dans l'ordre alphabétique. Chaque nom de variable de un ou de deux caractères peut en fait définir sept types différents de variantes : réelle, chaîne, entière, fonction, tableau de réels, tableau de chaînes et tableau d'entiers. La liste sera dans cet ordre. Pour une définition de fonction (comme par exemple `DEF FN A(X)`), la variable de la fonction sera imprimée sous la forme A[. La variable A peut donc prendre les formes suivantes :

```
A  A$  A%  A[  A(  A$(  A%(
```

Quand la variable est définie ou modifiée, une astérisque apparaît sur le numéro de ligne et ce y compris dans les instructions DIM, DEF FN, READ, INPUT et GET comme pour les = simples.

La commande `XREF£` fera référence aux numéros de ligne du programme courant. Le listing sera constitué des numéros de lignes référencés suivis de ":" ou de "?" et des lignes auxquelles ils renvoient. Le point d'interrogation signifie que la ligne n'existe pas dans le programme. Des renvois suivis de "s" sont des appels à des sous-programmes.
Avec les deux modes de XREF, vous pouvez préciser un en-tête de page comme dans :

```
XREF£ MONPROGRAMME,E60
```

Ceci ne peut être utilisé si vous omettez le paramètre E.
La barre d'espace peut être utilisée pour interrompre et reprendre l'affichage, CTRL-C annulera l'exécution de la référence croisée.

FIND

Cette commande est conçue pour rechercher une chaîne (Ascii ou hexa) dans des fichiers de tous types sur disque. Pour rechercher la chaîne CHAINE dans le fichier MONFICHIER, faites

```
FIND CHAINE, MONFICHIER
```

L'espace après FIND est obligatoire et le début de la chaîne recherchée est le signe qui vient immédiatement après. Si vous tapez deux espaces après FIND, le premier signe de la chaîne recherchée sera un espace. De même, la chaîne à rechercer doit obligatoirement être suivie d'une virgule.

Vous pouvez essayer par exemple la commande `FIND HUSTON,???? ????` étant mis pour n'importe quel dossier.

Majuscules et minuscules sont trouvées indifféremment et le signe ^ peut être utilisé comme joker pour remplacer n'importe quel caractère. D'où la commande :

```
FIND GS^H,MONFICHIER
```

qui permettra de trouver `GOSH,GISH, gush` et `Gash`.

Quand la chaîne est retrouvée, la position par rapport au début de fichier est affiché sous forme de 3 octets hexa suivi de la chaîne effectivement trouvée.

Pour rechercher un groupe d'octets en hexadécimal, par exemple 20 ED FD , on fera :

```
FIND$20 ED FD,MONFICHIER
```

Dans ce cas les espaces sont optionnels, mais il ne doit pas y avoir d'espace avant la virgule. Dans ce mode, l'affichage n'est constitué que du décalage sous la forme de 3 octets.

Si vous mettez le paramètre A, alors l'adresse précisée sera ajoutée sur le décalage dans le fichier afin que le résultat imprimé soit plus significatif. Si par exemple vous voulez rechercher la chaîne en hexadécimal 4C 60 E0 dans un fichier binaire dont l'adresse de chargement est $2000, vous faites :

```
FIND$ 4C 00 E0,MONFICHIER,A$2000
```

alors, la véritable adresse en mémoire de la chaîne trouvée sera affichée.

Dans un programme, la syntaxe à employer pour cette commande est :

```
PRINT CHR$(4)"FIND STRING,MONFICHIER"
```

ou bien :

```
PRINT CHR$(4)"FIND STRING,";F$
```

où F$ est le chemin d'accès, ou bien encore :

```
PRINT CHR$(4)"FIND ";S$;",";F$
```

où S$ est la chaîne recherchée et F$ est le chemin d'accès, etc.

Comme dans les autres modules, la barre d'espacement interrompt le listing tant qu'une autre touche n'est pas pressée. Pour faire défiler le listing pas à pas, appuyer plusieurs fois sur la barre d'espacement. CTRL-C annulera la lecture du fichier.

FORMAT

La commande FORMAT formate les disquettes 5,25" ou 3,5". La syntaxe est :

```
FORMAT nom.de.volume, S£,D£
```

Les paramètres de port (S) et lecteur (D) sont optionnels (6,1 par défaut). Cette commande s'exécute dès qu'elle est entrée. Aucun message ne s'affiche à l'acran pour que l'affichage des programmes ne soit pas perturbé. Les messages d'erreur standard No device connected, Write protected, I/O error (pour tout autre chose) peuvent être générés et peuvent être alors gérés avec l'ONERR de l'Applesoft.

EDIT et AUTO

Edit

La commande EDIT offre un éditeur complet de programme BASIC Applesoft. Bien qu'il ait la puissance des autres utilitaires de ce type, il n'occupe que 1,5Ko de mémoire. La commande :

```
EDIT
```

affiche tout le programme, une ligne à la fois, pour édition. La commande :

```
EDIT 100,1000
```

affichera de la ligne 100 à la ligne 1000. De même :

```
EDIT 100
```

n'affichera que la ligne 100 pour qu'elle soit éditée. La commande :

```
EDIT "CHAINE"
```

(ou `EDIT 100,1000"CHAINE"` etc.)
n'affichera que les lignes qui contiennent CHAINE. Quand une chaîne est indiquée et qu'un seul chiffre est donné, l'édition se fera de ce numéro de ligne jusqu'à la fin du programme.

Pour éditer une ligne, les commandes suivantes sont disponibles :

CTRL-I mode insertion à l'emplacement du curseur
CTRL-F cherche les caractères tapés après cette commande
DELETE supprime le caractère qui précède le curseur
CTRL-D supprime le caractère sous le curseur
RETURN accepte une ligne entière telle qu'elle apparaît à ce moment
← → déplacent le curseur
CTRL-B positionne le curseur au début de la ligne
CTRL-N positionne le curseur à la fin de la ligne
CTRL-X annule l'édition, la ligne courante reste inchangée
CTRL-C identique à CTRL-X pour ceux qui n'auraient pas de mémoire !
CTRL-Q tronque la ligne à l'endroit du curseur
CTRL-Z supprime entre le curseur et le caractère tapé ensuite
CTRL-R récupère la forme originale de la ligne
CTRL-O insère un caractère tapé après la commande - utilisé pour saisir des caractères de contrôle
CTRL-P compacte, supprime les espaces excédentaires dans une ligne
CTRL-W positionne le curseur sur l'espace précédant le mot suivant
CTRL-S positionne le curseur sur les deux points précédant l'instruction suivante
CTRL-L transforme le caractère sous le curseur en majuscule/minuscule

La commande EDIT peut être également utilisée pour changer complètement une chaîne par une autre :

```
EDIT "MAL","BIEN"
```

affichera toutes les lignes qui contiennent la chaîne MAL, fera le remplacement et BIEN apparaîtra en vidéo inverse. Il vous faudra taper sur la touche `Y`, `N` ou `CTRL-X` (cette dernière annule tout le processus). La ligne modifiée est ensuite réaffichée, avec l'éventuelle modification. Puis le processus reprend pour la prochaine occurence de MAL qui peut figurer à nouveau sur la même ligne. Vous pouvez aussi utiliser cette commande dans une série de lignes, comme dans :

```
EDIT 100,2000"MAL","BIEN"
```

N'importe quel caractère peut être utilisé comme délimiteur de chaîne pour remplacer le guillemet, à condition qu'il ne soit pas contenu dans les chaînes. La syntaxe `EDIT "MAL"BIEN"` est également acceptée.

EDIT est compatible avec toutes les cartes 80 colonnes qui permettent l'affichage en vidéo inverse ainsi qu'avec tous les modes de la carte Ultraterm.
L'éditeur n'ajoute pas d'espaces sur les lignes de REM ou de DATA comme le font certains éditeurs. En fait, les espaces à la fin de ces lignes sont automatiquement retirés quand vous validez une ligne en appuyant sur RETURN. (Il y aura donc une différence uniquement sur les lignes REM et DATA). Il est rare de vouloir conserver des espaces excédentaires dans les lignes de REM, mais cela reste possible en positionnant le curseur à l'endroit désiré et en utilisant CTRL-Q (et non sur RETURN) pour achever l'édition.

Si une ligne est trop longue pour entrer dans le buffer de l'éditeur, le mode EDIT essaie de la raccourcir en supprimant les espaces de formatage et en changeant les "PRINT" en "?". Si la ligne est encore trop longue, le message d'erreur OUT OF MEMORY s'affichera. Si une ligne devient trop longue pendant l'édition, alors les caractères que vous ajoutez dans le buffer ne seront pas acceptés. Dans ce cas, vous devrez supprimer les espaces de formatage avec la commande CTRL-P. Si cela ne suffit toujours pas, vous pouvez sortir du mode EDIT et revenir, ce qui aura pour effet de transformer tous les "PRINT" par des "?".

Auto

La commande :

```
AUTO 1000,100
```

vous mettra en mode numérotation automatique de ligne avec toutes les possibilités d'édition, en commençant à la ligne 1000 et en incrémentant de 100. L'incrémentation par défaut est de 10, le numéro de ligne par défaut est supérieur au dernier numéro de ligne du programme (arrondi à un multiple de 10). Toutes les touches de contrôle de l'éditeur sont acceptées sauf CTRL-R.

Si vous faites la commande AUTO alors que le numéro de ligne précisé existe déjà dans le programme, la commande sera simplement ignorée. De même, si la numérotation automatique atteint ou dépasse un numéro de ligne existant, vous sortirez simplement du mode AUTO. Cela évitera qu'une ligne existante ne soit recouverte à cause du mode AUTO et également que des enchevêtrements peu judicieux ne se produisent avec les lignes déjà existantes.

Pour sortir du mode AUTO, utilisez `CTRL-X` ou `CTRL-C`.

MACRO

La commande `MACRO` appellent une série de macro-commandes à partir du clavier. Elles peuvent être utilisées avec le module EDIT s'il est en mémoire. La table des macros peut être éditée avec le programme `MACRO.EDIT`. La commande MACRO attachée à une touche donnée est appelée en utilisant simultanément l'une des pommes (ou l'un des boutons de la poignée de jeux) et la touche en question. Les macros actuellement définies sont :

```
-       Bip, retour arrière, "-", 15 flèches de droite et retour-chariot
0       CATALOG
1       CATALOG,D1
2       CATALOG,D2
3       CATALOG/RAM
L       LIST
4       CHR$(◊)
9       PRINT SPC(◊)
CTRL-D  PRINT CHR$(4)"◊"
P       PREFIX
=       PREFIX retour-chariot (suivi par les huit premiers caractères du préfixe lu sur disque)
'       PRINT "◊"
G       GOSUB
I       IF ◊ THEN
/       INPUT "◊"
F       FOR ◊ = 1 TO
D       DATA "◊"
X       XDRAW
K       PEEK(49152) (adresse clavier)
S       POKE49168,0 (Remise à 0 du clavier)
CTRL-U  Huit flèches de droite
CTRL-H  Huit flèches de gauche
R       RENUMBER
H       HOLD
```

```
M       MERGE
E       EDIT
A       AUTO
```

Dans cette liste le caractère ◊ est mise pour signifier où s'arrêtra la macro si elle est appelée en mode immédiat. Dans le mode édition (commande EDIT), toute la chaîne sera produite et vous resterez à cet endroit, en mode insertion. Le ◊ ne fait pas partie de la macro-commande.

La macro "-" est conçue pour que vous mettiez le curseur sur le premier caractère d'un fichier du catalogue et que vous puissiez ainsi lancer ce programme . Le caractère Bip est mis ici pour des raisons techniques et ne doit absolument pas être retiré.

Les commandes utilisant la flèche de droite ne fonctionneront pas avec certaines cartes 80 colonnes, parce que la touche flèche de droite n'est pas gérée de la même façon par toutes les cartes. Vous pouvez redéfinir ces macro-commandes. La seule carte 80 colonnes qui est à présent compatible avec ces macro-commandes est celle de l'Apple //c. Toutefois, la macro "-" fonctionnera sur le //c (même en mode 40 colonnes), seulement si vous positionnez le curseur au-dessus de la ligne choisie, que vous tapez CTRL-X, puis la macro-commande pomme-.

Les macro-commandes peuvent être déconnectées, en mode immédiat, avec les commandes `PR£` ou `IN£` ou bien en pressant sur Reset. Retapez `MACRO` pour les reconnecter. Toutefois, quand elles sont connectées avec le mode EDIT, elles ne sont jamais déconnectées.

Dans le programme `MACRO.EDIT`, pour produire la marque ◊, utilisez la touche `DELETE` ou `CTRL-à` à l'endroit voulu (vous obtiendrez un damier type curseur). Soyez prudent quand vous fabriquez une macro-commande avec ce programme parce que vous ne pouvez pas utilisez la flèche de gauche pour corriger une erreur. La flèche de gauche met un CTRL-H dans la chaîne et de même RETURN mettra un CTRL-M. Vous devez sortir en appuyant deux fois sur ESCAPE. (Un seul ESCAPE suivi d'un autre caractère met ce caractère dans la chaîne. Vous ne voyez pas ESCAPE tant que vous n'avez pas appuyé sur une autre touche).

Le fichier MACRO.2

Le fichier MACRO.2 dans le dossier COMMANDS est une autre version du module MACRO qui est conçue pour être utilisée sur les anciens ordinateurs Apple qui n'ont pas les touches "Pomme ouverte" et "Pomme fermée". (bien que vous puissiez utiliser la poignée de jeux, mais ce n'est pas très commode). Cette version utilise une autre touche d'accès, CTRL-Z, qui devra être suivie de la touche de la macro-commande désirée. Nous avons utilisé CTRL-Z et non ESCAPE parce que de nombreuses cartes 80 colonnes interceptent les caractères ESCAPE et en interdisent donc son usage. Toutefois, vous pourrez facilement utiliser une autre touche d'accès si CTRL-Z ne vous convient pas. Cette touche est à la position relative $124 dans le fichier MACRO.2. (Elle est donc en $4124 quand le fichier est chargé à l'adresse $4000). Pour utiliser le fichier MACRO.2 à la place de MACRO, renommer les deux fichiers pour que MACRO.2 devienne MACRO.

LYST

La commande LYST affiche le listing formaté du programme en mémoire. Caractéristiques du formatage :

1 Les numéros de ligne sont justifiés à droite.
2 Les caractères de contrôle sont affichés, CTRL-G devient ainsi ^G.
3 Les instructions du programme sont imprimées sur des lignes séparées.
4 Les boucles FOR-NEXT sont indentées de trois espaces.
5 IF génère une indentation.
6 Il est possible de numéroter les pages et d'avoir une marge à gauche.
7 Les lignes de REM sont mises entre des lignes de tirets (optionnellement).

La commande LYST accepte un nom de fichier comme paramètre (utilisé pour l'en-tête de page) et les paramètres `E`, `L`, à et `R`. La syntaxe habituelle est :

```
LYST MONPROG,E60,L70,à5
```

Le paramètre E donne le nombre de lignes par page. La pagination n'est faite que si ce paramètre est utilisé.
Le paramètre L donne la longueur de la ligne, 80 par défaut.
Le paramètre à donne la marge de gauche, sa valeur par défaut est de 0.
Si le paramètre R est utilisé, les lignes de REM ne sont plus affichées entre des lignes de tirets. La valeur donnée à ce paramètre est sans importance.
La barre d'espacement fait interrompre et reprendre le défilement du listing. CTRL-C annule l'affichage du listing.

RND

Le module RND est un générateur de nombre aléatoire très rapide et efficace. Dans un programme la commande :

```
PRINT CHR$(4)"RND"
```

lance le générateur. Le générateur s'utilise alors avec l'instruction Basic `USR(X)` de la même façon que RND (X). Le générateur emploie la même racine que le RND habituel, et une racine `RND(X)` peut être invoquée avec un X négatif avant d'utiliser `USR(1)`. La valeur de l'argument X dans l'instruction `USR(X)` est sans importance. `R = USR (X)` donne un nombre aléatoire R dans la limite 0 <= R < 1.

Ce générateur ne redonnera les mêmes chiffres qu'après 4 millions d'utilisations, alors qu'avec le RND standard, ce sera après 40 000 appels avec la plupart des racines et après 202 appels avec d'autres. En outre il est environ 15 fois plus rapide que le RND standard.

SCI

Il s'agit d'un module de PRINT USING qui permet la notation scientifique. La syntaxe est assez semblable à celle du module USING, nous n'entrerons donc pas trop dans les détails ici. Il faut absolument la place d'un chiffre et un seul avant le point décimal, dans la chaîne de formatage, sinon un message d'erreur de syntaxe s'affichera. Vous pouvez utiliser n'importe quelle chaîne pour remplacer le "E" habituel. Lors de l'impression, le nombre est précédé soit d'un espace, soit du signe moins selon que le nombre est positif ou négatif. Il n'est pas envoyé de retour-chariot avant ou après le nombre, du texte pouvant ainsi être imprimé sur la ligne. La syntaxe à utiliser (toujours dans un programme) est :

```
PRINT CHR$(4)"SCI £.£££,£££ E":N ou
PRINT CHR$(4)"SCI";FT$:N
```

FT$ est dans ce dernier exemple la chaîne de formatage et N l'expression numérique Applesoft à imprimer. (L'espace après SCI dans le premier exemple est facultatif et n'est là que pour faciliter la lecture du programme. Voici des formats acceptés :

```
"£.£££££ EXP "  (l'espace après "EXP" sera mis avant l'exposant)
"£.£££,££ x10^"
"£.££ E"
```

RDLINE

Cette commande ne s'utilise que dans un programme avec la syntaxe :

```
PRINT CHR$(4)"RDLINE format":A$ ou
PRINT CHR$(4)"RDLINE";F$:A$  où F$ est la chaîne de formatage, ou bien encore :
PRINT CHR$(4)"RDLINE";F$:A$,DF$
```

où F$ est la chaîne de formatage et DF$ une valeur par défaut qui sera affichée à l'écran pour que l'utilisateur y apporte les modifications ou la ressaisisse avec les flèches. (Des soulignés seront imprimés pour remplir les espaces laissés libres quand la valeur par défaut est plus courte que le format).

Il s'agit d'un "input anything" (un Input qui accepte presque tous les caractères) qui mettra ce qui est saisi dans la chaîne A$ (toutes les variables chaîne sont acceptées). Le format est une chaîne composée de caractères de commande

et de caractères de formatage. Les caractères de commande sont :

```
^ / - _ % £ A a B b C c
```

et tous les autres caractères sont des caractères de formatage sauf ceux qui permettraient à des caractères de commande d'être interprétés comme des caractères de formatage ';' et '!' (seulement pour le premier caractère) qui dit au programme d'autoriser les flêches haut et bas pour valider une entrée.
Les caractères de commande disent quels sont les caractères d'entrée autorisés. Les caractères de formatage sont simplement affichés à l'écran et le curseur se déplace de l'un à l'autre. L'entrée se termine toujours par RTN, ESC ou par la flêche haut ou bas (si autorisée). La chaîne finale (A$ dans les exemples) est constituée de ce qui est affiché à l'écran (les soulignés s'effaceront lors de la validation).
Le dernier caractère tapé (l'un des caractères autorisés pour quitter) reste disponible à l'adresse clavier $C000. Le programme devra l'effacer par un GET ou par toute autre méthode de remise à zéro du clavier.
Cela a été conçu ainsi de sorte que la touche utilisée pour quitter soit facilement récupérable dans le programme où elle sera employée au gré du programmeur. RTN, retour arrière, DEL et les autres touches d'édition ne sont pas acceptés dans la chaîne saisie.
Les caractères de commande ont les significations suivantes :

^	accepte les caractères alpha-numériques, le point et RTN, la saisie est convertie en majuscule.
/	identique à ^ et accepte en outre /.
-	accepte tous les caractères sauf les caractères de contrôle, valide avec RTN.
_	identique à - mais RTN n'est pas accepté.
%	n'accepte que les chiffres ou l'espace, et le résultat final justifié à gauche.
£	n'accepte que les chiffres et l'espace.
*	n'accepte que les chiffres.
A et a	n'acceptent que les caractères alphabétiques.
B et b	acceptent les caractères alphabétiques et RTN.
C et c	acceptent les caractères alphabétiques, l'espace et RTN.
A B et C	convertissent les caractères saisis en majusules.

Quelques exemples :

```
PRINT CHR$(4)"RDLINE %%,%%%,%% *.**":N$
```

attendra un nombre sous le format ___,___,___.__, les chiffres et l'espace seront acceptés sauf dans la zone _.__ qui n'acceptera que des chiffres. RTN ne sera accepté que lorsque tous les blancs seront emplis. Si des espaces suivis de 1,234.56 sont entrés, la chaîne obtenue sera 1, 234.56 car % indique que les espaces de départ sont à retirer de la chaîne finale.

```
PRINT CHR$(4)"RDLINE £*¨-AAA¨-**":DA$
```

introduira les caractères dans le format __-___-__ , la chaîne obtenue aura un format de type date, le premier caractère peut être un espace, les trois du milieu doivent être des caractères alphabétiques et ils seront transformés en majuscules, les autres seront des caractères numériques. Notez que le caractère ¨ est utilisé pour indiquer que ce qui suit est un caractère de formatage et non un caractère de commande.

```
PRINT CHR$(4)"RDLINE A----------":N$
```

permettra de saisir une chaîne de 20 caractères maximum, le premier doit être alphabétique et transformé en majuscule. Le premier caractère excepté, l'entrée peut s'achever à tout moment par RTN. La chaîne saisie ne sera jamais vide et pourra contenir une virgule, des espaces, des points, etc.

```
PRINT CHR$(4)"RDLINE ¨/A////////////////// ":P$
```

donnera à une entrée la syntaxe ProDOS des chemins d'accès, le / étant accolé devant la chaîne qui en résulte.

Lors de la saisie, les commandes d'éditions suivantes sont disponibles :

← →	déplacent le curseur.
TAB	passe en mode insertion.
CTRL-I	passe en mode insertion.
DEL	supprime le caractère à gauche du curseur.
CTRL-D	supprime le caractère sous le curseur.

CTRL-B début de la ligne.
CTRL-N met le curseur à la fin de la chaîne par défaut.
CTRL-W "mot" suivant.
CTRL-R remplace par la valeur par défaut d'origine.
CTRL-Q efface du curseur à la fin ou bien jusqu'au prochain caractère de formatage.

Le mode insertion se signale par un curseur sous la forme d'une ligne verticale. L'insertion ou la suppression ne concernent que le segment compris entre les caractères de formatage.

La commande &INPUT est également disponible dans ce module. Il s'agit d'un "input anything" plus traditionnel qui a pour syntaxe :

```
&INPUT A$
```

Il est principalement conçu pour lire sur un fichier ouvert sur disque.

CLOCK

Le module CLOCK offre sur l'Apple //c une horloge pilotée par des interruptions. Vous devez saisir la date ainsi :

```
CLOCK 24/MAR/85
```

L'écran s'effacera alors et une zone d'affichage de l'heure apparaîtra dans l'angle supérieur droit. Il faudra utiliser les flêches et les chiffres pour saisir l'heure. L'horloge se mettra à tourner dès que vous aurez appuyé sur RTN. L'horloge affichera l'heure en permanence dans l'angle supérieur droit, elle comptera deux fois 12 heures par 24 heures. L'affichage permanent vous permet de vous assurer que l'horloge est bien toujours en service. RESET arrêtera l'horloge.

Les opérations sur disque, ou tout ce qui inhibe les interruptions retardera l'horloge. Il y a cependant moyen de modifier l'heure facilement. Si vous appuyez sur le bouton de la souris, l'heure affichée avancera d'une minute, si en même temps vous maintenez appuyée la touche pomme fermée, l'heure tournera plus rapidement. Notez que l'horloge va jusqu'à 12, si bien que vous devez passer deux fois le 12 pour remettre l'heure à zéro avec cette méthode. (La date ne sera pas modifiée).

Dans le dossier `PROCMD.DOC/COMMANDS` se trouvent trois autres versions du module horloge : une version avec un affichage sous 24 heures et deux versions "CLOCKE.XX" fonctionnant sur Apple //e avec une extension mémoire et carte souris. Il faudra simplement renommer et remplacer le module CLOCK par l'un de ceux-ci dans le dossier `PROCMD/COMMANDS` si vous le désirez. Les versions CLOCKE ont une longueur de quatre pages alors que les versions //c n'en comptent que trois. Elles ont toutes le même IDentificateur. Les modules CLOCK et MOUSE sont incompatibles entre eux : l'utilisation de l'un déconnectera l'autre.

Attention : il faut être prudent avec les routines qui sont pilotées par des interruptions et s'assurer que rien n'écrase la routine elle-même. La routine s'arrête et se déconnecte si elle conclut que la situation est dangereuse, mais vous ne devez absolument pas vous y fiez. En cas de doute, appuyez sur RESET ou bien faites la commande RESET avant d'utiliser certains fichiers système. N'utilisez pas CLOCK avec les versions de ProDOS antérieures à 1.1.1.

ONEKEY

Le module ONEKEY associera aux touches A, R, L, E et M, les ordres AUTO, RENUMBER, LIST, EDIT et CALL-151.
Pour que A, R et E soient opérationnelles, le module ONEKEY doit impérativement être chargé après les modules EDIT et RENUMBER.

HEXDEC

Ce module exécute des conversions hexadécimal/décimal. Pour convertir un nombre décimal en hexadécimal, tapez :

```
D (nombre décimal)
```

Dans l'autre sens, tapez :

```
H (nombre hexa)
```

Par exemple :

```
D1000 donne $03E8
D-151 donne $FF69
H1000 donne 4096
HFF69 donne 65385
```

Ceci peut fonctionner en mode programme. Par exemple, si vous voulez que votre programme donne la valeur en hexadécimal de la variable AD, vous pourriez écrire :

```
1000 PRINT "L'adresse est";
1010 PRINT CHR$(4)"D";AD
1020 PRINT ", en hexadécimal."
```

(Il n'y a pas de retour chariot après le nombre pour qu'il puisse être inséré dans une ligne de texte). Si AD = 4100, ces lignes de programme afficheront :

```
L'adresse est $1004, en hexadécimal.
```

POP

La commande POP vous permet de remonter facilement d'un niveau de préfixe. POP2 remontera de deux niveaux, etc. POP et POP1 sont équivalents. Si par exemple le préfixe courant est :

```
/DISQUE/DOSSIER/D1/D2
```

et que vous tapez POP2, alors le préfixe devient instantanément :

```
/DISQUE/DOSSIER/
```

Vous pouvez le programmer comme suit :

```
100 PRINT CHR$(4)"PREFIX"
110 INPUT PFX$
120 IF LEN(PFX$) > 7 THEN IF RIGHT$(PFX$,7) <>"/DOSSIER/" THEN PRINT
    CHR$(4)"POP":GOTO 100
```

EJECT

Le module EJECT s'utilise avec les lecteurs Unidisk 3,5" et donne la possibilité d'éjecter des disquettes à partir d'un programme Applesoft. Si vous le souhaitez, vous pouvez également l'utiliser en mode immédiat.

Le disque à éjecter est déterminé de trois façons : par défaut (dernier utilisé), par chemin d'accès, par numéros de port et de lecteur. En voici les syntaxes :

```
EJECT
EJECT chemin d'accès (entier ou partiel)
EJECT,S5,D1
```

`EJECT chemin d'accès, S5,D1` est aussi accepté.
Depuis un programme, faites : `PRINT CHR$(4)"EJECT"`

Si le contrôleur de disque se trouve bien dans le port indiqué mais qu'aucun lecteur ne lui est connecté, le message

d'erreur "NO DEVICE CONNECTED" apparaîtra. Dans d'autres cas (par exemple, le lecteur est un 5,25", aucun message d'erreur ne sera généré (et rien ne se produira). Cela permet d'utiliser la commande par programme ou avec un lecteur inconnu.

Vous pouvez également commander l'éjection de tous les lecteurs Unidisk en incluant le paramètre "à" comme dans `EJECT à0`. Vous devez mettre un chiffre après le caractère à mais sa valeur sera sans effet. Ce mode ne génère jamais de message d'erreur.
Si vous utiisez un chemin d'accès partiel, le lecteur concerné sera celui que BASIC.SYSTEM utilise par défaut.

Le module CMDS

N'avez-vous jamais travaillé avec PROCMD en vous rendant compte que vous vouliez travailler avec l'un des modules que vous n'aviez pas chargé en mémoire ? Bien que vous puissiez charger un autre module sans grande difficulté, il arrive parfois que cela ne soit pas pratique.
Ce module CMDS est la réponse. Pour utiliser CMDS, vous devez le charger en premier. (Ce n'est pas absolument indispensable, mais il fonctionnera mieux ainsi). Pour procéder aisément quand vous vous trouvez en Basic et que vous n'avez pas pensé à charger des modules dont vous avez besoin, CMDS a été mis sous la forme d'un fichier BIN, il peut être ainsi exécuté en tapant simplement `-CMDS` (ou `-COMMANDS/CMDS` s'il se trouve dans le sous-dossier COMMANDS, etc.). Une fois en mémoire, tout autre commande peut être appelée en tapant le nom du fichier de commande. Par exemple, si vous voulez copier un fichier, tapez :

```
COPY PATHNAME1,PATHNAME2
```

ainsi la commande COPY se chargera en mémoire et cette commande sera utilisable.

De fait ce n'est pas aussi simple. Si vous appelez la commande & pour voir la syntaxe CMDS, vous aurez :

```
CMDS /PROCMD/COMMANDS/
```

et cela signifie que les autres commandes sont supposées être dans le dossier /PROCMD/COMMANDS. Ce dossier peut bien sûr ne pas être en ligne directement si vous travaillez sur disque dur ou Unidisk.
Aucune importance, vous pouvez modifier le dossier de stockage CMDS en tapant la commande `CMDS` suivie du dossier contenant vos modules de commande PROCMD. Il est conseillé d'entrer le chemin d'accès complet de sorte que vous puissiez modifier les dossiers sans avoir à retaper cette commande. Par exemple, si vos commandes sont dans `/HARD1/PROCMD/COMMANDS`, tapez alors

```
CMDS /HARD1/PROCMD/COMMANDS
```

Les commandes recherchées ne sont que celles dont le nom est identique au nom du module. Cela fonctionne avec EDIT, par exemple, mais pas avec AUTO ; de même RESET n'est pas un nom de fichier mais figure dans le fichier POINTERS. Vous pouvez, bien sûr, copier le fichier POINTERS et l'appeler RESET.
Autre exemple, ONEKEY est le nom d'un fichier de commande, mais n'est pas une commande. Pour charger les commandes en mémoire, vous tapez ONEKEY. Comme ce n'est pas un nom de commande, un message d'erreur s'affichera (Syntax Error), ce qui est normal ; mais le fichier ONEKEY aura tout de même été chargé et les commandes associées seront disponibles.
Les commandes chargées ainsi résident en mémoire comme si elles avaient été chargées avec l'ancienne méthode.

Les fichiers de commande qui portent le même nom qu'une commande BASIC ne seront pas chargés avec CMDS. Cela ne concerne que les modules HGR et PRINT, mais en fait ils ne sont pas conçus pour être des commandes immédiates, c'est donc sans grande importance.
Vous pouvez, et sans doute devrez, changer le dossier de recherche par défaut `/PROCMD/COMMANDS/` et donner un nom d'accès complet pour ne pas avoir besoin de toujours taper la commande CMDS pour le modifier. Il est logé à l'adresse relative $2D4 dans le fichier CMDS. Il doit être en "ascii négatif", le bit de poids fort de l'octet à 1. Le nom doit s'achever par un "/" puis par un retour chariot $8D et enfin par un zéro. Sa longueur (y compris le slash "/", mais pas le retour chariot) doit être mise à l'adresse relative $2FE (à présent $11, longueur de "/PROCMD/COMMANDS/").
L'utilisation de ce fichier de commande n'est pas sans danger car un problème peut se produire en mémoire lors d'une tentative d'exécution d'autres routines. Le système pourrait se trouver écrasé.
Ce risque est inévitable à cause du mode de fonctionnement de BASIC.SYSTEM, il a été réduit au minimum mais il subsiste encore. Pour cette raison il vous faudra sauvegarder régulièrement le programme sur lequel vous

travaillez ; c'est d'ailleurs une précaution toujours valable même sans ce problème spécifique. Si un message PROGRAM TOO LARGE s'affiche lorsque vous demandez une commande qui n'est pas en mémoire, cela signifie alors qu'il n'y a pas assez de place, et que la commande n'a pas été chargée. Cela se produit, par exemple quand le LOMEM est placé trop haut. Si le module RENUMBER est en mémoire, la commande MERGE met automatiquement le LOMEM le plus bas possible, ainsi vous pourrez peut-être utiliser la commande refusée. (Ceci reste vrai même s'il n'y a rien à jumeler).

SETINFO

La syntaxe de cette commande est :

```
SETINFO chemin d'accès [,T type, A type auxilliaire, à accès]
```

Les trois paramètres sont facultatifs. Cela permet de modifier le type de fichier (sauf pour les fichiers DIR), le type auxilliaire, et les bits d'accès. L'octet d'accès est constitué de 5 bits significatifs (vous ne pouvez pas modifier les autres qui sont tous à zéro). Ils sont notés `DNB...WR` du bit de poids fort au bit de poids faible. Lorsqu'ils sont à 1, ils signifient :

- D destrucuction possible, bit 8 = 128
- N possibilité de renommer, bit 7 = 64
- B sauvegarde nécessaire, bit 6 = 32
- W écriture possible, bit 2 = 2
- R lecture possible, bit 1 = 1

Un fichier ordinaire non verrouillé a $C3 (en effet 128 + 64 + 2 + 1 = 195, soit $C3). Si vous faites la commande `SETINFO monfichier,à0`, vous n'aurez même pas la possibilité de lire le fichier de la façon traditionnelle (s'il s'agissait d'un fichier BAS, par exemple, vous ne pourriez pas le charger). L'utilisation de `SETINFO monfichier,à$C3` permettra à nouveau la lecture. (Dans ce cas particulier, la commande de base UNLOCK produirait le même résultat.)

PATH

La syntaxe de la commande PATH est la suivante :

```
PATH nom.de.fichier [,T type, S port, D lecteur]
```

Les trois paramètres sont facultatifs. Si les numéros de port et de lecteur ne sont pas donnés, le volume par défaut sera alors le dernier volume utilisé. Cette commande recherche le fichier spécifié dans tout le volume. S'il n'est pas trouvé, le message d'erreur "file not found" apparaît à l'écran. S'il est trouvé, le préfixe est alors mis pour le dossier qui contient le fichier et aucun message ne s'affiche. (Utilisez la commande `PREFIX` ou `CATALOG` pour voir où il a été trouvé). Si le fichier est un dossier, c'est le préfixe de ce dossier qui s'affiche et non le précédent préfixe qui pointe sur celui-ci.
Si le paramètre T est présent, il ne recherche que les fichiers du type demandé (au cas où il y aurait plusieurs fichiers portant le même nom).

COMPARE et PASTE

Ces deux commandes figurent dans le fichier de commande COMPARE.

COMPARE

La syntaxe en est la suivante :

```
COMPARE chemin1, chemin2
```

La même syntaxe s'applique pour PASTE. La commande COMPARE lira les deux fichiers et les comparera octet à octet. Si aucune différence n'est trouvée, elle quitte et le prompt Applesoft apparaît de nouveau. S'il y a des

différences, alors s'affichent les 128 octets de chaque fichier à partir de l'endroit où se trouvent la première des différences. Les octets qui diffèrent apparaissent en vidéo inverse. À ce niveau vous pouvez appuyer sur ESC pour sortir de la routine, sur RETURN pour continuer la recherche de différences, sur les flèches gauche/droite pour déplacer d'une ligne (16 octets) l'affichage du fichier, les flèches haut/bas pour vous déplacer de huit lignes.
(Notez que les flèches ne modifient pas la longueur de fichier affiché même si aucune erreur ne figure à l'intérieur de ces 128 octets, tel ne serait pas le cas avec la touche RTN).
Vous pouvez également passer outre un décalage de position entre les deux fichiers en appuyant sur la pomme ouverte ou fermée ainsi que sur une touche numérique. Cela permet de continuer à rechercher les autres différences dans les fichiers dans lesquels une insertion ou une suppression auront été opérées.

La touche CTRL-P envoie ce qui est à l'écran sur l'imprimante en port 1. Dans ce cas les octets qui diffèrent sont suivis du signe "<".

Si vous utilisez cette commande en mode 40 colonnes, l'affichage passera en 80 colonnes. Des caractères souris sont utilisés, aussi l'affichage paraîtra curieux sur des appareils autres que l'Apple //e 65C02, le //c ou le IIGS.

PASTE

La commande PASTE ajoutera le deuxième fichier à la fin du premier. Même syntaxe que COMPARE.

Attention : Le deuxième fichier restera inchangé mais le premier se trouvera modifié. Vous pouvez préférer copier le premier fichier sous un autre nom avant d'effectuer cette opération et éviter ainsi les risques, ou bien encore si vous voulez changer le premier fichier et le coller à nouveau à un autre.

Graphiques en double haute résolution

Les modules HGR, DHGR, FILL, PRINT, MOUSE, RGB et HLOAD contiennent des routines graphiques en double haute résolution. Elles ne peuvent être utilisées que sur un Apple //c, un GS ou un //e dont la carte mère est "révision B" et dont la carte 80 colonnes est dotée d'un strap.

Ces modules sont légèrement différents des autres parce que (HLOAD et HSAVE exceptés), ils fonctionnent avec des commandes ampersand et non avec des commandes externes ProDOS. Ces commandes ne figurent pas sur la liste "&" des commandes additionnelles.

Le module HGR

Le module HGR contient des routines graphiques pour dessiner des points et tracer des lignes en couleur double haute résolution. Ce mode double résolution a une résolution de 140 par 192. (Chaque point en couleur comprend quatre pixels pour un total de 560 pixels). Le module contient aussi des routines de dessin de formes dont la résolution est de 560 par 192.

Le module HGR se charge comme les autres, par :

```
BLOAD COMMANDS/HGR,TCMD,A$4000
CALL 4* 4096
```

Les fonctions disponibles sont les suivantes :

- `&HGR` (initialise les routines et efface l'écran)
- `&HGR2` (identique à &HGR mais tout l'écran est utilisé)
- `&HCOLOR=` (sélectionne les couleurs de 0 à 15)
- `&HPLOT X,Y (TO X2, Y2 etc.)` (dessine les points)

```
&PLOT X,Y (TO X2, 72 etc.) (fait un "XPLOT")
&DRAW S AT X,Y (identique au DRAW de base)
&XDRAW S AT X,Y (identique au XDRAW de base)
```

Les coordonnées horizontales des points doivent être les coordonnées habituellement utilisées pour la double haute résolution, soit 0 à 279. Elles sont converties par les routines en résolution 0-139 qui sera effectivement utilisée. Ainsi il sera très facile d'adapter des programmes en haute résolution à ces routines. (&DRAW et &XDRAW sont des routines de 560x192 mais les coordonnées de départ doivent être comprises dans les mêmes limites que pour les autres commandes. Cela permet d'augmenter la compatibilité ainsi que d'assurer l'intégrité des couleurs des formes. Cette limite ne s'applique pas pour une utilisation de &DRAW, sans précision de coordonnées.

Attention : Lorsque &DRAW est utilisé pour la première fois, il faut donner des coordonnées X,Y.

Les 15 couleurs disponibles pour la commande &HCOLOR sont :

```
 0 - noir          1 - magenta
 2 - bleu foncé    3 - violet
 4 - vert foncé    5 - gris 1
 6 - bleu moyen    7 - bleu clair
 8 - marron        9 - orange
10 - gris 2       11 - rose
12 - vert clair   13 - jaune
14 - aiguemarine  15 - blanc
```

Ce sont les même couleurs que celles de l'Apple en basse résolution.

&DRAW et &XDRAW fonctionnent avec les tables de forme standard mais dessinent en résolution 560 x 192. La commande &HCOLOR n'a pas d'effet sur les routines &DRAW. La coordonnée X de début doit se trouver entre 0 et 279 comme pour les autres routines. (Une forme commence donc toujours sur un pixel multiple de quatre en partant de la gauche, mais les déplacements vers la gauche et la droite se font d'un pixel à la fois). Avant de les utiliser, définissez ROT, SCALE et le pointeur de table de forme comme vous le feriez pour des graphiques en haute résolution. Mettez HCOLOR en blanc (3,7) ou en noir (0,4). Notez qu'il s'agit du HCOLOR standard qui est complètement indépendant de &HCOLOR. Le blanc et le noir sont disponibles pour qu'une image puisse être effacée en étant dessinée en noir. Les autres couleurs ne produiront pas de bons résultats.

Attention : Sous ProDOS il est important de protéger votre table de forme en plaçant le LOMEM au-dessus de celle-ci.

Note : Du fait de la conversion interne des valeurs horizontales 0 à 279 utilisées par la syntaxe de commande en valeurs comprises entre 0 et 139 utilisées par la machine, la commande &HPLOT 11,0 donnera le même résultat que &HPLOT 10,0 et il en sera de même pour toutes les coordonnées impaires de X. Cela peut être une source de confusion.

Le programme `DEMO/D.BRIAN` est une version double haute résolution du programme bien connu "Brian's theme" et propose une illustration de l'utilisation de ces commandes.
Le programme `DEMO/DRAW` est une adaptation d'une ancienne démonstration de formes créés avec la routine XDRAW en double haute résolution. `DEMO/DRAW2` en est une autre version.
Le programme `DEMO/COLOR` est du même type que `DEMO/DRAW` mais il utilise &HPLOT et &PLOT et non les routines DRAW.
Grâce au grand nombre de couleurs offertes par la haute résolution, ces démonstrations sont assez impressionnantes.

L'accès aux routines à partir du langage machine

Sauf pour &DRAW et &XDRAW, il est simple d'utiliser ces routines à partir de programmes en langage machine. L'accès peut se faire par un JSR à l'adresse relative :

```
$1E --- HPLOT et PLOT
$21 --- HGLIN
$24 --- HCOLOR
$27 --- HGR2 et HGR
```

Si le module HGR est le seul chargé ou bien s'il est chargé en premier, ces routines se trouvent toutes sur la page $97 (HGR2 par exemple se trouvera donc en $9727).
La coordonnée horizontale des entrées en langage machine doit se situer entre 0 et 139 ($00-$8B), soit la moitié des valeurs possibles avec les commandes ampersand en BASIC.

Description de l'utilisation de ces routines :

`HPLOT` Coordonées passéesX,Y. Tous les registres sont touchés. Coordonnées sauvegardées en $E0 et $E2.

`HGLIN` trace une ligne du point dont les coordonnées se trouvent en $E0, $E2 vers le point dont les coordonnées sont dans les registres A, Y. En sortie, les dernières coordonnées sont mises en $E0, $E2. Tous les registres sont touchés. Il devrait en principe n'être appelé qu'après un premier appel à HPLOT.
[Les routines HPLOT et HGLIN font un "PLOT" ou un "XPLOT" selon le signe de l'adresse $EB (235) : positif (<$80) ou négatif (=>$80).]

`HCOLOR` Couleur dans le registre A. Tous les registres sont touchés.

`HGR et HGR2` Effacent l'écran, et procèdent à toutes les initialisations nécessaires. L'appel se fait avec le registre X à 0 pour HGR2, à 1 pour HGR. Tous les registres sont touchés.

Le programme `DEMO/SWISH` est un exemple d'art cinétique et montre comment utiliser ces routines à partir du langage machine. Notez que la partie en Basic du programme utilise la commande RESET pour effacer tous les modules avant de charger le module HGR. Il en est ainsi parce que la partie de programme en langage machine sait qu'elle doit trouver les routines haute résolution aux adresses absolues indiquées ci-dessus.
`DEMO/SWISH.S` est le fichier source en Merlin-pro de la partie principale du programme SWISH. Il explique comment utiliser ces routines de traçage de ligne à partir de l'assembleur.

Le module DHGR

Le module DHGR comporte des routines en double haute résolution 'noir et blanc'. Ces routines de traçage de lignes utilisent la résolution 560x192 du mode double haute résolution, mais ne disposent pas de la couleur. Les commandes sont :

```
&&HGR   (identique à &HGR du module &HGR)
&&HGR2  (identique à &HGR2 du module HGR)
&&HPLOT X,Y (TO X1,Y1 etc.)
&&PLOT X,Y (TO X1,Y1 etc.)
```

Elles fonctionnent comme les commandes du module HGR auxquelles elles ressemblent, mais X peut avoir une valeur comprise entre 0 et 559. Les commandes `&&HGR` et `&&HGR2` mettent également HCOLOR à 7. Vous pouvez mettre HCOLOR à 0 pour dessiner une ligne en noir, sinon vous devez conserver HCOLOR à 7 pour utiliser ces routines. Notez qu'il s'agit de "HCOLOR" et non de "&HCOLOR".

Le langage machine et le module HGR ont les mêmes adresses relatives d'entrée, Mais les conditions d'entrée diffèrent. La routine HPLOT doit être appelée avec A = coordonnée y, X = coordonnée x poids faible, Y = coordonnée x poids fort, HGLIN doit être appelée avec A = coordonnée x poids faible, X = coordonnée x poids fort, Y = coordonnée y. Notez que ce module n'a que deux pages de long alors que HGR en a trois, si bien que les adresses absolues des routines s'en trouveront modifiées.

Le module FILL

Le module FILL comporte une routine de remplissage de couleur en double haute résolution. Les commandes, appelées par l'& sont les suivantes :

```
&COLOR= C  (sélectionne une couleur unie pour la commande FILL)
&FILL AT X,Y (remplit en commençant au point X,Y avec X < 280)
```

`&FILL W AT X,Y` (identique mais utilise le motif de couleur de 32 octets à l'adresse W)

(Notez que la commande &COLOR diffère de &HCOLOR).
Utiliser une couleur de base (1-15) pour remplir ne nécessite pas d'autre explication. La routine de remplissage utilise de la 15ème à la 56ème couleur et dispose ainsi de tons différents. Une table de couleur a 32 ($20) octets de long. Les bits de poids fort des octets de la table ne sont pas utilisés. La table a un format de 4 octets par huit. En supprimant les bits de poids fort, un point de couleur se lit sur quatre bits à la fois. (Note : Un bit remplace chaque segment de 4 bits quand ils sont tous à zéro, sinon la routine de remplissage planterait l'ordinateur).

Chaque couleur est représentée par quatre bits au format indiqué dans la table ci-dessous, bit le moins significatif en premier :

Noir	0000	Magenta	0001
Bleu foncé	1000	Violet	1001
Vert foncé	0100	Gris1	0101
Bleu moyen	1100	Bleu clair	1101
Marron	0010	Orange	0011
Gris2	1010	Rose	1011
Vert clair	0110	Jaune	0111
Aigue marine	1110	Blanc	1111

Le programme `DEMO/FILL` présente le mode remplissage couleur unie.
Le programme `DEMO/FILL2` illustre l'utilisation d'une table de couleurs. La table est générée de façon aléatoire et est positionnée en $300. Elle utilise le module DHGR et non HGR pour tracer le polygone.
Le programme `DEMO/COLOR.CHART` utilise l'ensemble des couleurs de la table `DEMO/MIXTURES` pour présenter un graphique des 225 motifs les plus simples avec seulement deux couleurs différentes. La table elle-même a été générée par le source en assembleur Merlin `DEMO/MIXTURES` ; l'étude de ce fichier source aidera à la compréhension de la structure d'une table de couleur en général.
Le programme `DEMO/BAR` propose un graphique sous forme de barres et en utilisant FILL.

Le nombre de tables de couleurs différentes est de 15 puissance 56, nombre si important que dans la pratique il peut être considéré comme infini.

La routine de remplissage est accessible avec des programmes en langage machine. Le point d'entrée se situe à l'octet $1E dans le module FILL. Pour entrer, il faut indiquer les coordonnées X,Y du point de départ du remplissage dans les registres X,Y (sous le format 140x192). En outre, l'adresse de la table de remplissage doit se trouver aux adresses $EE, $EF.

Pour utiliser une couleur unie à partir du langage machine, vous pouvez faire un JSR dans la routine COLOR= à l'adresse relative $21 avec la couleur désirée dans le registre A. Cette routine met le pointeur de la table afin qu'après cet appel vous n'ayiez qu'à effectuer un JSR dans la routine FILL.

Si le module HGR est chargé en premier et le module FILL en second, ces points d'entrée dans les routines FILL seront alors sur la page $95.

Le module PRINT

PRINT est un générateur de caractères en double haute résolution. Il est en fait à mi-chemin entre un générateur de caractères et une routine de dessin bit map. Ce qui le rend très souple. La syntaxe de PRINT est la suivante :

```
&PRINT W AT X,Y:"chaîne"
```

Ici W est l'adresse de la police de caractères qui sera utilisée. Elle ne doit être précisée que la première fois où PRINT est utilisé. X et Y sont les coordonnées de l'angle supérieur gauche de début de l'impression sous le format 280x192.

Si les adresses des coordonnées et de la police de caractères ne sont pas données, l'affichage commencera à partir de la dernière position utilisée.

PRINT a trois modes qui dépendent de la valeur de HCOLOR :

`HCOLOR = 0 à 3` impression en OU exclusif.
`HCOLOR = 4 ou 5` le fond.est effacé.
`HCOLOR = 6 ou 7` impression en ou, le fond est respecté

Chaque entrée dans la table de la police de caractères a sa propre définition horizontale et verticale. Ainsi il devient possible d'utiliser l'espacement proportionnel et le module PRINT a plus de possibilités en bit map.

Chaque entrée dans la police de caractères a le format suivant :

Octet 1 = Ascii (positif) du caractère défini.
Octet 2 = nombre d'octets dans l'entrée (y compris l'en-tête)
Octet 3 = nombre de pixels horizontaux de couleur sur 4 bits.
Octet 4 = nombre de lignes verticales.

Ils sont suivis par la définition du caractère. Le premier octet contient les deux premiers pixels, le premier étant le quartet de poids faible, le deuxième le quartet de poids fort et ainsi de suite jusqu'à ce que la première ligne soit utilisée. (Si le nombre de pixels est impair, le dernier demi-octet n'est pas utilisé). La table doit s'achever par un octet à zéro.
Le programme `FONT.EDIT` vous permet de créer ou d'éditer une police de caractères. Il se limite aux polices dont le nombre de pixels n'excède pas 8 (32 bits) en largeur et 16 en hauteur, mais les tables de police dépassent rarement ces dimensions. Il y a sur le disque une police qui s'appelle `DEMO/FONT`. Elle est utilisée par les programmes `DEMO/BAR` et `DEMO/ANIMATION`. Tous les caractères d'impression Ascii sont définis. Une autre petite "police" appelée `DEMO/ANI.FONT` est utilisée par `DEMO/ANIMATION`, elle contient des caractères graphiques.

En langage machine, vous pouvez appeler la routine PRINT à l'octet relatif $1E. L'appel doit se faire avec le caractère qui sera imprimé dans le registre A en Ascii positif. La police de caractères a nécessairement pour adresse l'octet relatif $1B,$1C. Sous le format 140x192, le point de départ doit se situer aux adresses X0 = $E0 et Y0 = $E2.

Le programme `DEMO/ANIMATION` utilise les possibilités graphiques du module PRINT pour présenter une petite animation. Il utilise également les modules DHGR et FILL, ainsi que le fichier normal `DEMO/FONT` comme la petite police `DEMO/ANI.FONT` qui contient les formes de blocs. Cette démonstration est conçue pour illustrer la plupart des possibilités de ces modules. C'est une démonstration "honnête" ne contenant rien qui soit hors de portée d'un programmeur moyen en Applesoft.

Le module MOUSE

Ce module ne fonctionne que sur un Apple //c, un GS ou //e avec nouvelle ROM et carte souris. Il comporte des routines gérées par les interruptions qui permettent à l'utilisateur de déplacer sur l'écran des objets double haute résolution grâce à la souris. Elle est transparente pour un programme BASIC.

Les objets qui peuvent être déplacés sont des caractères qui appartiennent à un fichier FONT qui a le même format que pour le module PRINT. À l'origine ils sont placés sur l'écran par certaines commandes ampersand. Le module accède aux caractères dans les polices grâce à des "numéros logiques d'objet". Cela permet la localisation d'un même caractère dans la police par différents numéros logiques et donc de différencier les objets à l'écran. La valeur des numéros logiques doit se situer entre 0 et 127, le nombre 0 désigne le curseur de la souris. (Ainsi, chaque caractère de la police peut être désigné comme curseur). Le "point chaud" est toujours l'angle supérieur gauche d'un objet.

Les commandes du module MOUSE sont :

`&INT ou &INT W`	initialise les routines d'interruption de la souris. W est l'adresse de la police, $4000 par défaut
`&LET N,A$`	affecte A$ à l'objet logique numéro N
`&STORE N AT X,Y`	met l'objet logique N sur l'écran en X,Y. 0<X<280. Il peut alors être déplacé avec la souris indépendament du programme

`&STORE N`	retire l'objet N de l'écran
`&POS N,X,Y,S`	met la position de l'objet N dans les variables X,Y et l'état de la souris dans S
`&WAIT 1`	inhibe les interruptions, désactive la souris
`&WAIT 0`	réactive les interruptions
`&END`	déconnecte les routines, désactive la souris

L'état de la souris n'indique que les trois bits de poids fort correspondant à l'adresse de l'état ($778 + port). Bit 7 = bouton appuyé, bit 6 = le bouton a été appuyé, bit 5 = la position a changé. (Reportez-vous au mode d'emploi de votre souris).

L'affichage de pixels se fait comme avec le module PRINT mais il est toujours du type XOR et n'affiche qu'un seul caractère à la fois (pas de chaînes complètes), les coordonnées doivent être données pour chaque caractère. Ainsi A$ ci-dessus doit être une chaîne d'un seul caractère.

Le programme `DEMO/MOUSE` est un programme très simple qui montre l'utilisation du module MOUSE. Il emploie le module HGR pour tracer un plan de maison, met quelques "meubles" au bas de l'écran et entame une boucle infinie. Vous pouvez utiliser la souris pour déplacer les meubles. Vous stopperez le programme par CTRL-C. Le programme pourrait aussi bien regardez indéfiniment si une touche n'a pas été appuyée et ensuite procéder à l'opération correspondante, par exemple, il pourrait placer sur l'écran d'autres objets correspondant à la touche appuyée.

L'accès aux routines de la souris peut se faire à partir du langage machine par une table de saut à l'octet relatif $1E. L'accès peut se faire par un JSR à l'octet relatif :

```
$1E --- INIT
$21 --- INITA
$24 --- LETCHR
$27 --- ERASE
$2A --- STORE
```

Ces routines fonctionnent comme suit :

`INIT`	initialise la souris (comme avec &INT).
`INITA`	initialise et prend comme adresse de la table de police la valeur des registres A,Y (poids fort, poids faible).
`LETCHR`	affecte un caractère Ascii dans A au numéro logique dans X. A doit être en Ascii positif et X doit impérativement être compris entre 0 et $7F. Le nombre logique 0 correspond au curseur.
`ERASE`	efface le caractère dont le numéro logique est dans X (<$80).
`STORE`	Met le caractère dont le nombre logique est dans le registre X sur l'écran à la position indiquée dans les registres A et Y. Le registre A contient la coordonnée X comprise entre 0 et 139 et Y contient la coordonnée Y comprise entre 0 et 191. Le registre X (numéro logique) doit être positif.

Les fonctions WAIT peuvent être exécutées simplement par les instructions SEI et CLI.

Attention : Aucun contôle n'est effectué pour vérifier les valeurs données aux registres lors des appels en langage machine. Vous devez vous assurer qu'elles se situent bien dans les limites autorisées sinon vous pourriez détruire les routines. C'est plus dangereux avec les routines gérées par les interruptions, la plus grande vigilance est donc recommandée.
Les routines gérées par les interruptions peuvent être dangereuses si elles sont partiellement recouvertes par erreur. Ces routines sont écrites de sorte qu'elles fassent leurs propres contrôles et elles se déconnectent si un problème est décelé. Pour plus de sécurité, toutefois, vous devez penser à envoyer l'instruction &END pour déconnecter les routines quand vous les avez utilisées ; ou simplement faites Control Reset. Les routines d'interruption du module MOUSE ne doivent pas être utilisées avec les versions de ProDOS antérieures à 1.1.1.

Le module RGB

C'est un nouveau module conçu pour être chargé APRÈS les modules HGR ou DHGR (ou les deux). Il intercepte certaines commandes ampersand pour manipuler les interrupteurs logiques de la façon qui convient à mes cartes RGB. Il ajoute également un mode RGB mixte.

Les commandes qui sont interceptées sont :

`&HGR` et `&HGR2`	(elles sélectionnent le mode couleur RGB et fonctionnent de façon traditionnelle)
`&&HGR` et `&&HGR2`	(sélectionnent le mode monochrome RGB, etc.)
`&HCOLOR`	(interceptée pour des raisons techniques)

Les nouvelles commandes sont :

`&TEXT`	(déconnecte la double haute résolution et sélectionne le mode texte 40 colonnes)
`&GR`	(sélectionne le mode RGB "mixte couleur/monochrome")
`&CLEAR`	(efface l'écran graphique quelque soit le mode)

Utilisation de ces extensions

Il est absolument indispensable de charger le module RGB après le module HGR ou le module DHGR (ou bien les deux), sinon les commandes intecerptées seront prises en compte par les anciennes routines et non par les nouvelles. L'ordre de chargement des autres modules tels que FILL ou PRINT est sans conséquence.

Après le chargement du module RGB, certaines modifications devront être apportées dans les programmes pour que ces commandes externes soient encore plus efficaces. Les programmes qui utilisent les routines de dessin de formes (en HGR) doivent connecter le mode graphique avec une commande &&HGR ou &&HGR2 et non avec les commandes &HGR, car ces routines sont uniquement monochromes. Par contre, la commande &GR peut être utilisée pour le mode mixte. Ceci est valable également pour les programmes qui utilisent le module PRINT.

Pour les mêmes raisons, les programmes qui emploient la double haute résolution couleur doivent utiliser &HGR, &HGR2 ou &GR pour connecter le mode graphique.

De nombreux programmes utiliseront la double haute résolution en couleur et en monochrome. Par exemple, vous pouvez avoir des écrans en couleur contenant du texte grâce au module PRINT. Avec le mode monochrome, la définition est nettement supérieure. Dans ce cas, vous avez donc avantage à utiliser le mode "mixte". C'est-à-dire qu'il faudra utiliser la commande &GR pour connecter le mode graphique et effacer l'écran.

Pour déconnecter de façon fiable les modes graphiques, utilisez `&TEXT`.

HLOAD et HSAVE

Ce sont des commandes externes ProDOS. Elles permettent de charger et de sauvegarder des écrans en double haute résolution. La commande de sauvegarde ne peut être utilisée que si le fichier existe déjà (utiliser l'ordre `Create`).

Ces commandes acceptent les deux types de fichier, BIN et $F7, elles fonctionnent différemment en fonction de chacun.

Pour les fichiers de type BIN, le fichier est sauvegardé (et chargé) selon la méthode classique. C'est-à-dire qu'il n'y a pas de compression de fichier ni de décompression. Ces fichiers ont une longueur de 33 blocs. D'autres programmes double haute résolution utilisent ce format. Ces fichiers ont une longueur de $4000 octets, les $2000 premiers octets constituent la partie mémoire auxilliaire de l'image et le reste, la partie mémoire principale.

Si le fichier est du type $F7, l'écran double haute résolution sera compressé pendant la sauvegarde et décompressé

pendant le chargement. L'écran ne s'en trouve pas modifié. Cela économise de la place sur le disque mais la sauvegarde et le chargement demandent davantage de temps. Le fichier peut occuper 3 à 5 fois moins de place. Des dessins très compliqués et très détaillés ne seront pas compressés efficacementet la place occupée sur le disque peut même être plus importante que les 33 blocs d'un fichier standard. Dans ce cas il est préférable d'utiliser le format BIN.

Pour créer un fichier du type $F7, faites :

```
CREATE MONFICHIER,T$F7
```

L'écran se met en mode graphique double haute résolution quand les commandes HLOAD et HSAVE sont invoquées mais il revient à son état original quand le chargement ou la sauvegarde est terminé. Ainsi, si vous voulez qu'un écran graphique reste affiché après la fin du chargement, il vous faudra appeler le mode graphique double haute résolution avant le HLOAD.

Le programme CONVGR est un utilitaire très court et très rapide qui fonctionne sur la page 3 et dont le but est de convertir une image standard en image double haute résolution. Pour l'utiliser, chargez d'abord l'image standard sur l'écran haute résolution (A$2000) et tapez BRUN CONVGR. Comme avec la commande HLOAD, l'écran se mettra en double haute résolution et puis reviendra à l'état initial après l'exécution du processus (qui est bref). Vous pouvez même utiliser la commande HSAVE pour sauvegarder l'image double haute résolution ainsi obtenue.
Rappelez-vous que vous devez d'abord créer le fichier destination avant d'utiliser la commande HSAVE.

Résumé des commandes haute résolution

Module HGR

&HGR	initialise l'écran mixte
&HGR2	initialise le plein écran
&HCOLOR= (0-15)	sélectionne la couleur
&HPLOT X,Y (TO X1,Y1...)	plot
&PLOT X,Y (TO X1,Y1...)	xplot
&DRAW S (AT X,Y)	trace la forme S
&XDRAW S (AT X,Y)	xdraw la forme S
HCOLOR = 0 ou 4	choisit le noir pour &DRAW
HCOLOR= 3 ou 7	choisit le blanc pour &DRAW

Module DHGR

&&HGR	initialise l'écran mixte
&&HGR2	initialise le plein écran
&&HPLOT X,Y (TO X1,Y1...)	plot
&&PLOT X,Y (TO X1,Y1...)	xplot
HCOLOR= 0	sélectionne le noir
HCOLOR= 7	sélectionne le blanc (valeur par défaut)

Le module FILL

&COLOR= (0-15)	remplit en couleur unie
&FILL AT X,Y	remplit en utilisant la dernière table
&FILL W AT X,Y	remplit en utilisant la table à W

Module PRINT

&PRINT W AT X,Y:chaîne	imprime en utilisant la police à l'adresse W
&PRINT AT X,Y:chaîne	imprime en utilisant la dernière police
&PRINT:chaîne	imprime à partir de la dernière position
HCOLOR= (0-3)	affiche en XOR
HCOLOR= (4-5)	affiche en 'copy', efface le fond
HCOLOR= (6-7)	mode ou

Module MOUSE

```
&INT ou &INT W          initialise les routines souris
&LET N,A$               affecte A$ au numéro logique N
&STORE N AT X,Y         met l'objet N sur l'écran
&STORE N                retire l'objet N
&POS N,X,Y,S            X,Y = position de N, S = statut
&WAIT 0                 interruptions activées
&WAIT 1                 interruptions inhibées
&END                    déconnecte les routines, module hors service
```

Module RGB

```
&TEXT                   retourne au mode texte
&GR                     monochrome et couleur mélangés
&CLEAR                  efface l'écran quel que soit le mode
```

Si vous voulez avoir un écran mixte, dans votre programme vous devez faire :

```
PRINT CHR$ (4) "PR£3"
```

avant la commande `&HGR`.

Pour sortir proprement du mode texte avec ces routines et si le mode PR£3 n'est pas utilisé, vous devez faire :

```
POKE 49164,0:TEXT
```

Si vous avez fait `PR£3`, TEXT tout seul suffit sauf si vous voulez revenir en 40 colonnes.

Attention : Si vous n'avez pas fait PR£3 et que vous tapez `TEXT` sans ajouter `POKE 49164,0` votre programme peut être écrasé et détruit.

Si vous utilisez le module RGB (même sans écran RGB), il est plus facile d'utiliser la commande `&TEXT` à la place de ce POKE, pour déconnecter le mode double haute résolution.

Table de couleurs pour les commandes &HCOLOR et &COLOR

```
 0 noir                  1 magenta
 2 bleu foncé            3 violet
 4 vert foncé            5 gris 1
 6 bleu moyen            7 bleu clair
 8 marron                9 orange
10 gris 2               11 rose
12 vert clair           13 jaune
14 aigue marine         15 blanc
```

Informations techniques

Chaque module occupe en mémoire un multiple de 256 octets et tous les efforts ont été fait pour utiliser un minimum de mémoire. De plus chaque module contient un octet IDentificateur qui peut être utilisé par un programme pour déterminer si un module en particulier se trouve en mémoire. L'octet ID se trouve à l'adresse relative $1D dans le module relogé (adresse relative $11D dans le fichier CMD lui-même). Le nombre de pages (1 page = 256 octets) occupées par chaque module et l'ID sont donnés dans la table ci-dessous :

SCI	1 page	ID = 53
RND	1 page	ID = 34
FORMAT	8 pages	ID = 33
RDLINE	4 pages	ID = 32
COMPARE et PASTE	5 pages	ID = 31
PATH	2 pages	ID = 30

SETINFO	1 page	ID = 29
CMDS	2 pages	ID = 28
EJECT	1 page	ID = 27
RGB	1 page	ID = 26
MOUSE	6 pages	ID = 25
POP	1 page	ID = 24
ONEKEY	1 page	ID = 23
HEXDEC	1 page	ID = 22
CLOCK	3 pages	ID = 21
HLOAD et HSAVE	3 pages	ID = 20
PRINT	2 pages	ID = 19
DHGR	2 pages	ID = 18
FILL	2 pages	ID = 17
FIND et FIND$	2 pages	ID = 16
LYST	3 pages	ID = 15
HGR	3 pages	ID = 14
MACRO	3 pages	ID = 13
EDIT et AUTO	6 pages	ID = 12
XREF	4 pages	ID = 11
RENUMBER, HOLD et MERGE	6 pages	ID = 10
COPY	2 pages	ID = 9
USING	2 pages	ID = 8
DATESTR	2 pages	ID = 7
DATE	1 page	ID = 7
VARTRC	2 pages	ID = 6
VARLST	2 pages	ID = 5
SORT	4 pages	ID = 4
DUMP	2 pages	ID = 3
POINTERS et RESET	1 page	ID = 2
TYPE	2 pages	ID = 1
ONLINE	1 page	ID = 0

Comme les modules doivent être chargés à l'adresse $4000 pour que la routine de relocation fonctionne, ils ne peuvent être chargés avec un long programme Basic qui dépasserait cette adresse. La meilleure solution sera d'employer un programme du type de COMMANDER qui ne charge que les modules désirés puis qui lance le programme principal.

Quand vous chargez un module et que vous tapez `CALL 4 * 4096`, l'octet IDentificateur du module chargé est vérifié. Si ce module est déjà installé, rien ne se passe. Cela permet d'éviter de charger plusieurs fois un même module et de perdre de la place. Voici un morceau de programme qui peut être utilisé dans un programme Basic pour rechercher un module particulier par son octet ID :

```
10 ID = ? : REM IDentificateur du module désiré
20 FD = 0
30 IF PEEK (116) = 150 THEN 100 : REM aucun module n'est chargé
40 PT = 48646
50 IF PEEK (PT + 1) THEN 100 : REM pas d'autre module
60 PT = 256 * PEEK (PT + 2) + 3
70 IF PT > 39420 THEN 100 : REM ne pointe pas sur un module
80 IF PEEK (PT + 26) <> ID THEN 50 : REM ce n'est pas celui-ci
90 FD = 1 : REM trouvé
100 IF NOT FD THEN PRINT CHR$(4) "BLOAD ???,TCMD,A$4000": CALL 4 * 4096
```

Attention : Comme ProDOS a un système particulier de nettoyage mémoire (garbage collection), il peut arriver qu'un module soit écrasé lors de l'opération. Le risque est limité, mais vous pouvez le réduire encore en chargeant les modules que vous voulez avant qu'un grand nombre de chaînes aient été définies. En mode immédiat, le risque

peut être évité en tapant systématiquement CLEAR ou NEW avant la commande BLOAD.

Quand un module est chargé en $4000 et que vous faites `CALL 4 * 4096`, un contrôle est fait pour voir si la table de variables n'a pas été écrasée. Si elle l'a été, alors la tentative de relogement avorte et le message d'erreur NO BUFFERS AVAILABLE (pas de buffers disponibles) apparaît à l'écran. Cela peut être une source d'erreur car cela signifie que le LOMEM n'a pas pu être mis en $4000 ou au-dessus avant ce CALL. Donc chargez et faites le CALL avant de placer le LOMEM. À partir du clavier, si ce message d'erreur apparaît, entrez une ligne 'inutile' (par exemple, `0 <RTN>` (en considérant qu'il n'y a pas de ligne 0 dans le programme courant), qui aura pour effet de réinitialiser le LOMEM, puis faites les BLOAD et CALL.

Ces modules sont conçus pour être compatibles avec d'autres programmes qui se logent au-dessus des buffers de fichier. Toutefois, pour fonctionner, de tels programmes doivent se conformer aux règles de programmation sous ProDOS. (Précisément, le programme APA de chez Apple ne peut pas être utilisé, mais pourquoi voudriez-vous l'utiliser ?) La compatibilité avec les autres utilitaires de ProDOS n'est pas toujours garantie. Notez également que la commande RESET supprimera tout ce qui est au-dessus des buffers de fichier, et pas seulement ces modules. D'autres utilitaires peuvent contenir des commandes pour supprimer ce qui se trouve au-dessus des buffers de fichier, mais elles ne doivent pas être utilisées à la place de la commande RESET car avec certaines commandes, RESET procède à d'autres nettoyages de détail.

Écrivez vos propres modules

Trois éléments sont nécessaires pour écrire facilement des modules compatibles avec l'ensemble PROCMD. L'utilisateur doit être à l'aise en langage machine et bien connaître ProDOS. Le module doit être écrit en assembleur Merlin-pro avec les options REL et DSK (ainsi le code objet sera écrit sur le disque sous la forme d'un fichier LNK (type $F8)). Vous devez alors lancer le programme `RELOC` sur ce disque avec `BRUN` (opération devant être faite à partir de Merlin-pro) et préciser le nom du fichier LNK qui a été généré avec l'assembleur. Un en-tête de relogement d'une page de long sera ajouté au fichier LNK et les pointeurs de sauvegarde de Merlin-Pro seront mis à jour afin que le fichier ainsi obtenu soit sauvé sous la forme d'un fichier CMD.

Le programme lui-même doit obéir à certaines règles, il doit commencer par trois sauts :

```
DOSENTRY   JMP PARSE
DOSEXIT    JMP XRETURN  (= $BE9E)
AMPEXIT    JMP MONRTS   (= $FFCB)
```

(Ces adresses seront modifiées par le relogeur.) Ces sauts doivent également être suivis de la routine qui analyse la commande "&". La routine PARSE est la routine de votre programme qui analyse les commandes externes de ProDOS. (Si votre programme n'utilise que les commandes "&", le premier JMP peut être un JMP DOSEXIT). Si vous utilisez les commandes externes de ProDOS, votre analyseur de "&" doit alors imprimer la syntaxe de la commande quand l'appel se fait avec le bit Z positionné (BEQ).

De plus l'ID de la routine doit être mis à l'adresse $1D. Voici un exemple de début d'un tel fichier (sans commandes "&") :

```
DOSENTRY   JMP PARSE
DOSEXIT    JMP XRETURN
AMPEXIT    JMP MONRTS
AMPENTRY   BNE AMPEXIT              ;passe les autres commandes &
           LDY £0
:LOOP      LDA CMDNAME,Y
           JSR COUT                 ;montre la syntaxe de la commande
           INY
           CPY £AMPEND-CMDNAME+1
           BLT :LOOP
           JSR CHRGOT               ;(=$B7) Retrouve A et le statut
           BDQ AMPEXIT              ;branche toujours
```

```
          DFB ID                    ;votre ID
CMDNAME   ASC "COMMAND"
CMDEND    ASC " sa syntaxe"
AMPEND    HEX 8D
PARSE     ...
```

La routine "PARSE" doit aller à DOSEXIT avec la retenue à 1 si vous n'acceptez pas la commande. (La commande sera dans le buffer d'entrée avec le bit de poids fort à 1).

Pour éviter la duplication des nombres ID, nous avons établi les règles suivantes :

1. L' ID $FF est réservé aux routines au-delà de la 255ème.
2. Les ID $F0-$FE doivent être utilisés pour des routines "temporaires", nombre provisoire en attendant un prochain "vrai" ID. Ils doivent être utilisés pour des routines envoyées à un éditeur.
3. Les ID $E0-$EF sont réservées à l'usage "personnel". Utilisez-les pour vos propres routines qui ne seront distribuées qu'à quelques amis.
4. Aucun autre ID ne doit être utilisé sauf si c'est l'éditeur qui l'a attribué à votre routine.

Attention : Lancer RELOC à partir de Merlin-pro supprime le fichier source en mémoire. Assurez-vous d'abord qu'il a bien été sauvegardé).

Sommaire
Index alphabétique des Commandes

Introduction	2
Le programme STARTUP	2
Le programme COMMANDER	2
Le programme FITR	3
Le programme DEMO/SORT	9
Graphiques en DHGR	22
Le module DHGR	24
Le module FILL	24
Le module PRINT	25
Le module MOUSE	26
Le module RGB	28
Informations techniques	30
A.R.L.E.M.	18
Auto	13, 14
&Clear	28
Clock	18
Cmds	20
&Color=	24
Compare	21
Copy	9
Date	4
Dates	4
Date£	4
Date%	4
&Draw	23
Dump	6
Edit	13
Eject	19
&End	27
&Fill	24
Find	12
Format	12
&Gr	28
&Hcolor	22
Hexdec	19
&Hgr	22
&&Hgr	24
&Hgr2	22
&&Hgr2	24
Hload	28
Hold	11
&Hplot	22
&&Hplot	24
Hsave	28
&Int	26
&Let	26
Lyst	15
Macro	14
Merge	11
Mouse	18

Onekey 18
Online 4
Paste 21, 22
Path 21
&Plot 23
&&Plot 24
Pointers 4
Pop 19
&Pos 27
&Print 25
Rdline 16
Renumber 10
Reset 4
Rnd 16
Sci 16
Setinfo 21
Sort 7
&Store 26
&Text 28
Type 6
Using 5
Varlst 6
Vartrc 6
&Wait 27
&Xdraw 23
Xref 11

Loi du 3 juillet 1985 : Protection des auteurs de logiciels de Micro-Informatique

Sept articles de la loi du 3 juillet 1985, applicable à compter du 1er janvier 1986, sont consacrés à la lutte contre le piratage des logiciels. Cette loi définit la notion de contrefaçon et prescrit les sanctions applicables aux contre-facteurs.

Définition de la contrefaçon au sens de la loi du 3 juillet 1985

L'article 47 de la loi du 3 juillet 1985 dispose :
"Par dérogation au 2 de l'article 41 de la loi N° 57–298 du 11 mars 1957 précitée, toute reproduction autre que l'établissement d'une copie de sauvegarde par l'utilisateur ainsi que toute utilisation d'un logiciel non expressément autorisée par l'auteur ou ses ayants–droit est passible des sanctions revues par ladite loi".

La reproduction d'un logiciel

Constitue le délit de contrefaçon de logiciels, la reproduction d'un logiciel faite pour l'usage privé de celui à qui est fourni le logiciel. La seule reproduction licite en vertu de la nouvelle loi vise à l'établissement d'une copie de sauvegarde pour l'usage de celui à qui est fourni le logiciel.

Utilisation en dehors des conditions prévues par la licence concédée au concessionnaire

Constitue également un délit de contrefaçon une utilisation d'un logiciel qui n'a pas été expressément autorisée par l'auteur ou ses ayants–droit. Ainsi, toute personne utilisant sans autorisation le logiciel qui lui a été fourni ou l'utilisant pour un autre usage que celui qui lui a été concédé par le propriétaire des droits, peut-être poursuivie pénalement et civilement par l'auteur, même si cette utilisation ne constitue pas une violation du monopole de la reproduction.

La mise en œuvre des poursuites

Elle se fait par le biais d'une saisie-contrefaçon suivie ensuite d'une assignation délivrée au contrefacteur soit devant le tribunal civil, soit devant le tribunal correctionnel.

La saisie-contrefaçon peut-être uniquement descriptive. Dans ce cas, l'huissier ou le commissaire se contente de saisir deux exemplaires des œuvres contrefaites pour obtenir un moyen de preuve. Elle peut être réelle et porter sur l'intégralité des exemplaires saisis. Toutefois dans ce cas, l'autorisation préalable du juge, saisi par voie de requête, (dont bien sûr la partie adverse n'a pas connaissance) est nécessaire ; de plus, la saisie-contrefaçon réelle de tous les exemplaires a lieu aux risques et périls du saisissant.

La saisie est faite soit par un huissier de justice, soit par un commissaire de police. L'un et l'autre peuvent se faire assister par un homme de l'art de leur choix.

Sanctions applicables au délit de contrefaçon au sens de la loi du 3 juillet 1985

L'article 47 de la nouvelle loi prévoit que le piratage de logiciels est passible des sanctions prévues par la loi du 11 mars 1957, qui donne à l'auteur du logiciel la possibilité d'exercer l'action en contrefaçon afin de faire condamner le contrefacteur, tant devant le tribunal civil que le tribunal correctionnel, puisque la contrefaçon en matière de droits d'auteur, constitue un délit pénal. La loi du 11 mars 1957 renvoie elle-même aux articles 425 à 429 du Code Pénal. Les deux textes principaux sont les articles 425 et 426 du Code Pénal qui prescrivent les sanctions suivantes :

Article 425 — Alinéa 1 et 2

Toute édition d'écrits, de composition musicale, de dessin, de peinture ou de toute autre reproduction, imprimée ou gravée en entier ou en partie, au mépris des lois et règlements relatifs à la propriété des auteurs, est une contrefaçon, et toute contrefaçon est un délit.
La contrefaçon, sur le territoire français, est punie d'un emprisonnement de trois mois à deux ans et d'une amende de 6 000 francs ou de l'une de ces deux peines seulement.

Article 426

Est également un délit de contrefaçon toute reproduction, représentation ou diffusion par quelque moyen que ce soit d'une œuvre de l'esprit en violation des droits de l'auteur, tels qu'ils sont définis et réglementés par la loi.