GUIDE DE L'APPLE

LES EXTENSIONS

Benoît de Merly

« Voici le livre que nous attendions : complet, clair et pratique, il devrait rapidement devenir le compagnon familier et indispensable de tous les utilisateurs de l'Apple.
Nous sommes particulièrement heureux de saluer son arrivée au moment où nous sortons notre nouvel Apple II : le Apple II e. »

Jean-Louis Gassée
Directeur Général de Apple France

Ce livre s'adresse à toute personne qui utilise un ordinateur personnel Apple, ou qui désire connaître les possibilités de l'Apple avant de choisir un système.
Chacun y trouvera l'information pratique dont il a besoin, quelle que soit l'application envisagée : gestion, calcul scientifique, jeux, graphiques, acquisition de données, contrôle de processus...

l'auteur

L'auteur, Ingénieur civil des Ponts et Chaussées, titulaire d'une licence en informatique, a eu l'idée de ce livre en utilisant lui-même un Apple : il s'est donc posé les questions que l'utilisateur sera amené à se poser s'il veut tirer de son ordinateur toutes ses ressources, et le maîtriser parfaitement.

GUIDE DE L'APPLE

TOME 1	TOME 2	TOME 3
L'APPLE STANDARD	LES EXTENSIONS	LES APPLICATIONS

GUIDE DE L'APPLE

TOME 2

Edimicro

LES EXTENSIONS DE L'APPLE

Première édition

Tome 2. ISBN : 2-904457-01-1

EDIMICRO
DÉPARTEMENT ÉDITIONS DE F.D.S. SARL
10, rue Henri-Pape, 75013 Paris. Tél. : 588-76-53

LES EXTENSIONS DE L'APPLE

par

Benoît de MERLY

Edimicro

Table des matières

TOME 2 : LES EXTENSIONS

CHAPITRE 2. — **Le système UCSD**

CHAPITRE 1

Le système d'exploitation : le DOS 3.3

1.1 PRÉSENTATION GÉNÉRALE DU DOS 3.3

Le DOS 3.3 est un système de gestion de disquettes (Disisk Operating System) permettant de stocker des informations sur des disisquettes et de les recharger en mémoire quand on le souhaite.

Une disquette est un disque magnétique souple sur lequel on peut stocker des informations comme sur une cassette. La pririncipale différence est qu'il faut dérouler toute la cassette avant d'arriver à la donnée voulue, alors que l'accès à cette donnée est beaucoup plus rapide avec une disquette.

Cette différence se retrouve entre un disque musical et une cassette audio. Vous pouvez accéder directement à la troisième chanson du

disque, alors que vous devez écouter les deux chansons précédentes sur une cassette.

En outre, une disquette permet de stocker plus d'informations qu'une cassette et d'y accéder beaucoup plus rapidement. Le DOS 3.3 permet de stocker 140 K octets de données sur une disquette, soit beaucoup plus que la capacité mémoire de votre APPLE II.

Il existe plusieurs versions du DOS. Le DOS 3.3 est la dernière version sortie pour l'APPLE II à ce jour.

Nous allons étudier dans ce chapitre les commandes du DOS 3.3, les possibilités de gestion de fichier de données, de création de commandes cataloguées, et nous donnerons aussi un aperçu de la structure du DOS et des informations sur la disquette.

Les exemples seront, pour la plupart, écrits en APPLESOFT.

1.2 GÉNÉRALITÉS SUR LA STRUCTURE DES INFORMATIONS SUR UNE DISQUETTE

Les disquettes sont des disques magnétiques souples décomposés en pistes concentriques numérotées de 0 à 34 pour le DOS 3.3. Une piste est aussi décomposée en parties appelées secteurs.

Pour les versions antérieures du DOS, le nombre de secteurs par piste était de treize. Pour le DOS 3.3 ce nombre est devenu seize. Chaque secteur contient dans tous les cas 256 octets.

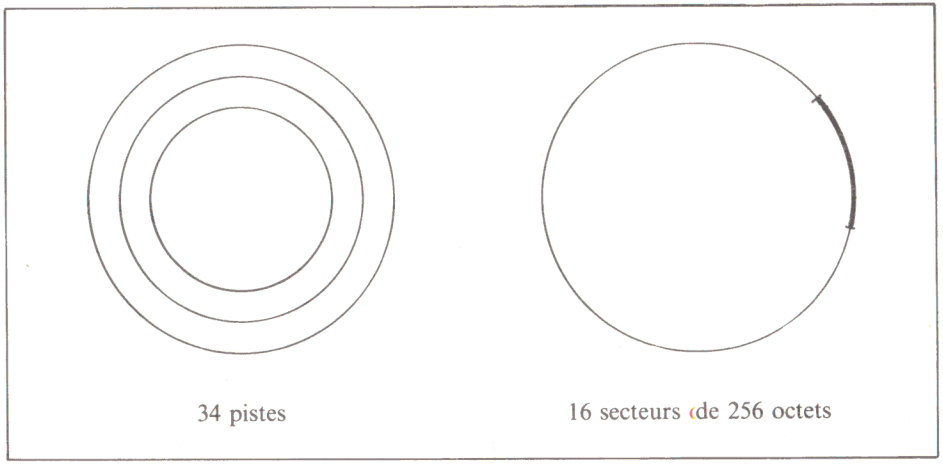


Photo J

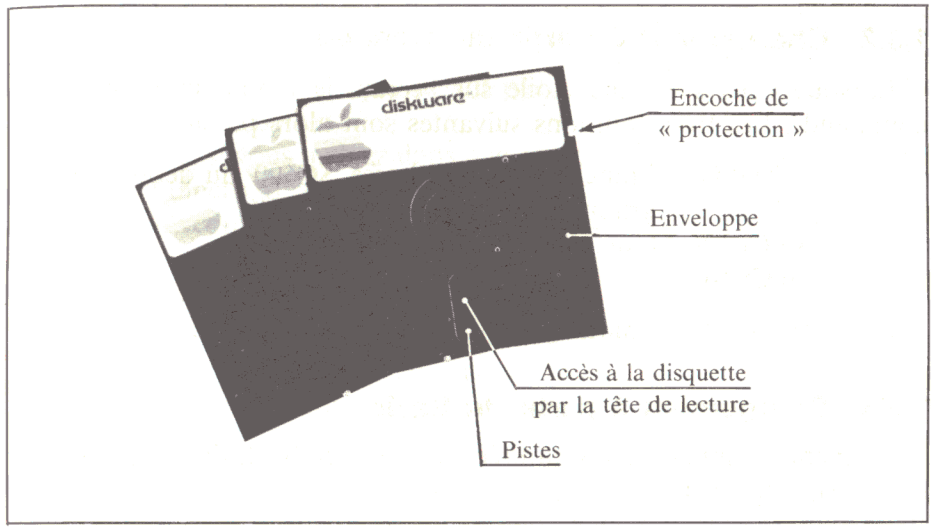


Photo K

L'enveloppe de la disquette permet l'accès par la tête de lecture *via* une encoche radiale. La tête de lecture se déplace pour accéder aux diverses pistes. L'accès aux secteurs s'obtient en faisant tourner la disquette dans son enveloppe.

L'encoche de « protection » permet l'écriture sur la disquette. Si elle est « bouchée », toute écriture sur la disquette est interdite.

1.3 CHARGEMENT DU DOS 3.3

Pour accéder aux commandes du DOS, vous devez l'avoir chargée en mémoire. Cela peut se faire de plusieurs façons, selon la version de l'APPLE II et le langage dans lequel vous êtes situé.

Dans tous les cas, nous supposerons que la disquette « SYSTEM MASTER » est dans le drive 1 du contrôleur situé dans le connecteur 6.

1.3.1 ROM autostart. APPLE II + carte langage

C'est la méthode la plus simple. Il suffit d'allumer votre APPLE II. Lorsque le DOS est chargé, le caractère s'affiche.

1.3.2 Chargement à partir du moniteur

Lorsque vous avez une étoile sur l'écran, le moniteur attend votre commande. Les trois solutions suivantes sont alors possibles :

*C600 G branchement à l'adresse $C600 du début du programme de chargement du DOS
*6 Ctrl-K (voir moniteur de l'APPLE)
*6 Ctrl-P

Le DOS est alors chargé.

1.3.3 Chargement à partir du Basic

La même commande est utilisée pour les deux Basic. Vous pouvez taper l'une des deux commandes suivantes :

PR#6
IN#6

1.3.4 Chargement d'une ancienne version du DOS

Dans ce cas, la disquette Basic doit être mise à la place de la disquette « SYSTEM MASTER ». Lancer alors le chargement du DOS. Lorsque le message :

INSERT BASIC DISK AND PRESS RETURN

ou

INSERT 13 SECTORS DISK AND PRESS RETURN

apparaît, mettez votre ancienne disquette système. Le chargement d'une ancienne version du DOS se fait alors.

1.4 LES COMMANDES GÉNÉRALES DU DOS 3.3

1.4.1 Options générales

La plupart des commandes du DOS 3.3 vous permettent de préciser certaines options : choix d'une unité de disquettes parmi les deux qui sont gérées par un contrôleur, choix du connecteur où se situe le contrôleur à utiliser, ou encore choix du volume de disquette.

Le format de ces trois options est toujours le même. C'est pour cette raison que nous les étudions avant les différentes commandes. Ces options modifient les valeurs prises par défaut par le DOS, lors des commandes ne précisant pas d'option.

Par exemple, si vous incluez dans une commande l'option S5 (choix du connecteur numéro 5), le contrôleur situé dans le connecteur 5 sera utilisé pour toutes les commandes du DOS jusqu'à ce que l'option S soit invoquée à nouveau.

Ces trois options ont le format suivant :

Ss où s désigne le numéro du connecteur (1 à 7),
Dd où d désigne le numéro du disque (1 à 2),
Vv où v désigne le volume de la disquette (1 à 254).

Le numéro de disque correspond à la position occupée par le câble qui relie le lecteur de disquettes au contrôleur.

Le volume d'une disquette est un paramètre qui n'a pas de signification physique sur l'APPLE II. Le DOS 3.3 utilise cette donnée, uniquement dans une commande, pour la comparer avec le volume de la disquette en cours de traitement. Si une différence existe, le message d'erreur VOLUME MISMATCH est affiché et le DOS n'effectue pas la commande.

Si vous ne précisez pas de volume dans votre commande, le DOS ignore le volume de la commande.

Le volume étant indiqué lorsque vous utilisez la commande CATALOG (cf. § 5.4.3), vous pouvez l'utiliser pour numéroter vos disquettes ou pour distinguer les disquettes de sauvegarde de vos programmes.

Le volume vous servira chaque fois que vous voudrez distinguer des disquettes dans un but de protection.

Exemple : CATALOG D2, S5, V1

Si dans une commande, un mauvais numéro de connecteur (ou un numéro de disque) est donné, le DOS affichera :

I/O error

et émettra un « beep ».

Pour récupérer votre programme, vous suivrez le scénario suivant :

1. CATALOG, Ss′ où s′ est le bon numéro du connecteur.
2. Lorsque le système vous rend la main, appuyer sur Reset.
3. Taper 3DOG si nécessaire (cas où vous vous êtes retrouvé dans le moniteur).

1. *Le* DOS *doit toujours être chargé à partir de l'unité de disquette* 1.

2. *Vous ne devez jamais appuyer sur la touche Reset lorsqu'un lecteur de disquettes est en marche (lumière rouge allumée) : la disquette insérée serait perdue.*

1.4.2 Obtention du catalogue d'une disquette

La commande CATALOG du DOS 3.3 permet d'obtenir le catalogue d'une disquette.

Cette commande affiche d'abord le volume de la disquette, puis pour chaque fichier, il indique si le fichier est verrouillé ou non. Le type des données du fichier, le nombre de secteurs de 256 octets occupés par le fichier, et le nom du fichier.

Une disquette DOS 3.3 peut contenir jusqu'à 84 fichiers (limitation de la taille du catalogue).

Un fichier verrouillé est un fichier protégé en écriture. Il ne peut donc ni être écrasé, ni détruit. Les fichiers verrouillés sont indiqués dans le catalogue par un astérisque précédant le type du fichier.

Les 4 types de fichier sont représentés par une lettre :

T fichier texte,
I programme Basic Entier,
A programme APPLESOFT,
B code binaire.

La longueur du fichier est le nombre de secteurs occupés par le fichier. Si le fichier dépasse 255 secteurs, la longueur indiquée par CATALOG est la longueur réelle modulo 256.

Le nom d'un fichier a une longueur comprise entre 1 et 30 caractères. Il doit commencer par une lettre et peut comprendre tous les caractères, sauf la virgule. Vous pouvez mettre dans le nom d'un fichier des caractères de contrôle mais ils ne sont pas visibles à l'écran. C'est un moyen d'empêcher d'autres personnes d'utiliser vos fichiers, mais n'oubliez pas les caractères que vous avez tapés !

Le format de la commande CATALOG est le suivant :

CATALOG Dd, Ss, Vv

Exemple : Si le nombre total de lignes dépasse 20 lignes, le DOS affichera les 21 premières lignes et attend que vous tapiez n'importe quelle touche

(excepté Reset, CTRL, SHIFT) pour afficher la fin du catalogue. Cela vous donne le temps nécessaire pour lire le nom de tous les fichiers.

```
]CATALOG
DISK VOLUME 254
*A 006 HELLO
*I 051 BRIDGE1
*I 007 CLOCK 2
*I 045 YAM'S
*A 009 B. NAVALE
*A 005 LUNARLANDING
*A 013 RADIO AMATEUR
*A 030 DYNASTY
*A 036 BOWLING
*A 029 JEUX MATHS
*I 025 CRYPTOGRAM
*I 015 COWBOY
*B 003 SHOOTOUT
*I 029 LOGIC DESIGNER
*I 025 COLOR PHOTO
*I 029 PHOTO NEG FILE
*A 031 WORLD POWER 6
*I 013 RAID
```

1.4.3 Initialisation d'une disquette

La disquette « System Master » fournie avec le DOS, est une disquette prête à l'emploi. Ce n'est pas le cas pour les disquettes que vous achèterez dans le commerce.

Pour vous en persuader, sortez votre disquette système « System Master » et mettez une disquette vierge. Essayer de recharger votre APPLE.

Par exemple, tapez PR#6. Que se passe-t-il ? L'unité de disquettes se met en route, la lumière rouge s'allume, des bruits douteux se font entendre, puis la disquette continue à tourner et ne s'arrêtera pas.

Vous devez alors appuyer sur RESET. Nous vous avons dit de ne jamais appuyer sur Reset si vous ne voulez pas perdre le contenu de la disquette. Dans ce cas précis, la disquette étant vierge, vous ne perdrez rien.

Que s'est-il passé ? L'APPLE a cherché sur la disquette des informations qu'il n'a pas trouvées. Une disquette vierge est comme une cassette vierge. Elle est vide. Avant de l'utiliser, vous devez l'initialiser.

La commande INIT permet d'initialiser les disquettes « esclaves ». Ces disquettes esclaves sont dépendantes de la taille mémoire du système. La taille de la mémoire utilisée par la disquette est la même que celle du système qui a initialisé la disquette.

Si une disquette est créée avec une mémoire de 16 K octets, seuls les 16 premiers K octets seront utilisés même avec un APPLE II de 32, 48 ou 64 K octets.

Il est possible de transformer ces disquettes esclaves en disquettes maîtres en employant l'utilitaire MASTER CREATE (cf. § 1.8). Le DOS se charge alors automatiquement à l'adresse correspondant à l'utilisation la meilleure de la mémoire.

Le format de la commande INIT est le suivant :

INIT nom de fichier, Ss, Dd, Vv .

Cette commande positionne le volume de la disquette. Si vous ne donnez pas de volume ou si vous donnez v à zéro, le volume est pris, par défaut, égal à 254.

Pourquoi indiquer un nom de fichier dans la commande INIT ? La raison en est simple. Lorsque vous chargez le DOS à partir d'une disquette, celui-ci mettra en mémoire le fichier dont vous aurez indiqué le nom dans la commande INIT et lancera l'exécution du programme correspondant.

1. *Le fichier indiqué dans la commande* INIT *est créé par cette commande en recopiant le programme Basic en mémoire sur la disquette.*

2. *Les informations situées sur la disquette sont écrasées par la commande* INIT. *Faites donc très attention à ne pas initialiser des disquettes non vierges sauf si vous voulez récupérer une ancienne disquette.*

3. *Nous vous conseillons d'utiliser toujours le même nom pour le fichier à lancer au chargement. Par exemple* « BONJOUR ». *De plus il est souvent utile de faire afficher les caractéristiques de la disquette par ce programme.*

Voici un exemple d'utilisation de la commande INIT avec l'APPLESOFT :

1. Enlevez la disquette système « System Master » et remplacez-la par la disquette à initialiser.

2. Entrez le programme à exécuter à chaque chargement du DOS. Voici un exemple de ce type de programme.

3. Tapez les commandes suivantes :

```
]NEW
]10  PRINT "DISQUETTE ESCLAVE 48K"
]20  PRINT "CREEE LE 5 MAI 19822"
]30  END
]INIT HELLO
I/O ERROR                pas de disquettee dans le lecteur 1
]INIT HELLO, D2          initialisation dissquette
]CATALOG, D2
DISK VOLUME 254          } Vérification
A 002 HELLO
]RUN
DISQUETTE ESCLAVE 48K    } Essai dde HELLO
CREEE LE 5 MAI 1982
```

4. Lorsque le système vous rend la main, recharggez le DOS. Vous voyez apparaître le commentaire :

```
APPLE II
DISQUETTE ESCLAVE 48 K
CREEE LE 5 MAI 1982
```

5. Rangez la disquette système « System Mastter ». La disquette que vous venez d'initialiser vous servira pour essayer l les exemples des paragraphes suivants.

Vous avez pu vérifier, en suivant cet exemple, qque le DOS a initialisé la disquette contenue dans le lecteur numéro deuux qui devient à partir de cet instant accessible à l'utilisateur, et qui ccontient le programme HELLO.

1.4.4. Les fichiers verrouillés

Si vous voulez protéger un programme contre unne destruction possible, la commande LOCK du DOS verrouillera le fichnier. Le format de cette commande est le suivant :

LOCK nom fichier, Ss, Dd, Vv

Si vous essayez ensuite de détruire le fichier ou de le renommer, le le DOS refusera votre commande et affichera le message :

FILE LOCKED (fichier verrouillé)

Si vous essayez de sauver sur disquette (cf. § 1.5.1), un programme écrit dans le même langage que le fichier verrouillé, vous obtiendrez le même résultat. Par contre si vous voulez sauvegarder sous le même nom, un fichier de type différent, vous verrez s'afficher le message :

FILE TYPE MISMATCH (type du fichier inapproprié)

Dans un catalogue, les fichiers verrouillés sont distingués des autres par un astérisque (cf. § 1.4.2).

Nous vous recommandons de verrouiller le fichier à exécuter au chargement du DOS (§ 1.4.3).

Il vous est possible de déverrouiller un fichier. Vous devrez utiliser la commande UNLOCK dont le format est le suivant :

UNLOCK nom fichier Ss, Dd, Vv

Le fichier ne sera plus protégé et pourra être détruit ou écrasé par une nouvelle version du programme.

Exemple :

```
]LOCK HELLO
]CATALOG

DISK VOLUME 254

*A 002 HELLO

]UNLOCK HELLO
]CATALOG

DISK VOLUME 254

A 002 HELLO
```

1.4.5 Destruction d'un fichier

La commande DELETE permet de détruire un fichier. Le format de la commande est le suivant :

DELETE nom fichier, Ss, Dd, Vv

Par exemple les commandes :

```
]LOAD HELLO
]SAVE BONJOUR
]DELETE BONJOUR
]CATALOG

DISK VOLUME 254

*A 002 HELLO
```

créeront le fichier BONJOUR puis le supprimeront de la disquette située dans le lecteur actif.

Si le fichier à détruire est verrouillé, le DOS affiche alors le message d'erreur FILE LOCKED et ignore la commande.

Exemple :

```
LOCK HELLO
]DELETE HELLO

FILE LOCKED
```

1.4.6 Changement du nom d'un fichier

La commande RENAME permet de modifier le nom d'un fichier. Son format est le suivant :

RENAME ancien nom, nouveau nom

Comme pour DELETE, le DOS ignore la commande et affiche FILE LOCKED si le fichier est verrouillé.

S'il existe déjà un fichier dont le nom est celui que l'on veut donner au fichier à renommer, le DOS *ne recherche pas cette existence et change dans le catalogue l'ancien nom pour y mettre le nouveau. Vous aurez donc deux fichiers portant le même nom sur la disquette. Ceci pourra entraîner de nombreux quiproquos. Faites donc très attention à ce phénomène.*

1.4.7 Vérification physique d'un fichier

La commande VERIFY permet de vérifier que les secteurs occupés par un fichier ne sont pas abîmés physiquement. Si le DOS trouve une erreur, il affiche le message :

I/O ERROR

Sinon, il rend la main à l'utilisateur sans afficher de message. Le format de cette commande est le suivant :

VERIFY nom de fichier, Ss, Dd, Vv

Vous pouvez utiliser cette commande pour tous les types de fichier et à partir de deux Basic et du moniteur.

1.4.8 Passage d'un Basic à l'autre

Celui-ci est réalisé avec les commandes :

FP (APPLESOFT) du DOS
INT (Basic Entier)

Plusieurs cas se présentent :

— vous êtes en Basic APPLESOFT et vous tapez FP. Ceci détruit votre programme et vos données et vous restez en Basic APPLESOFT ;
— la situation est identique pour INT et le Basic Entier ;
— vous êtes dans un Basic et vous voulez passer à l'autre ;

- tapez FP et vous vous retrouvez en APPLESOFT ;
- tapez INT et vous vous retrouvez en Basic Entier.

Cette possibilité n'est offerte qu'aux possesseurs de la carte-langage, ce qui permet d'avoir un Basic en ROM *et l'autre en* RAM *de la carte-langage (fichiers* INTBASIC *et* FPBASIC).

Pour les autres utilisateurs, la seule solution est de charger l'autre Basic à partir d'une cassette ou d'une disquette en RAM, *mais ceci conduit à une perte de taille mémoire.*

Exemple :

```
LOAD HELLO
]LIST

10 PRINT « DISQUETTE ESCLAVE 48K »

20 PRINT « CREEE LE 5 MAI 1982 »
30 END

]INT
>LIST

>10 PRINT « COUCOU »
>20 END
>LIST
    10 PRINT « COUCOU »
    20 END

>FP

]LIST
```

1.4.9 Commandes de mises au point du DOS, MON, TRACE, NOMON

- **MON, NOMON**

Les commandes MON, NOMON du DOS permettent d'obtenir des traces des ordres et des données relatives aux fichiers manipulés par le DOS. Trois options sont possibles :

Commands (commandes du DOS OPEN, CLOSE...)
Inputs (données lues sur la disquette)
Outputs (données écrites sur la disquette)

Le format des commandes est

MON C, I, O activation des traces
NOMON C, I, O désactivation des traces

Par défaut, l'option NOMON C, I, O (pas de trace) est retenue. L'effet de la commande MON est active jusqu'à ce que l'on rencontre NOMON, INT, FP, Reset, 3DOG.

L'exemple suivant montre l'utilisation des commandes MON et NOMON avec l'option C, I, ou O.

```
]MON C
]PRINT ""CLOSE"      fermeture de tous les fichiers
CLOSE                imprimé par MON
]
]NOMON C, I, O
]PRINT ""CLOSE"
]
```

● **Trace (APPLESOFT)**

La commande Tracce de l'APPLESOFT ne fonctionnant pas avec le DOS, on évitera de s'een servir.

1.5. LES FICHIERS DE PROGRAMMES

1.5.1 Programmattion en Basic

Nous allons vous exxpliquer dans ce paragraphe comment utiliser le DOS pour sauver et reecharger vos programmes Basic. Vous saurez aussi utiliser le DOS dans vcos programmes Basic.

● **Chargement d'un programme Basic en mémoire**

Le chargement d'un programme en mémoire se fait par l'intermédiaire de la commande LOAD dont le format est le suivant :

LOAD nom de fichier, Vv, Dd, Ss

La différence avec le chargement à partir d'une cassette réside dans le fait qu'il faut préciser un nom de fichier et que le chargement est beaucoup plus rapide.

Si vous voulez par exemple charger en mémoire le programme APPLESOFT nommé COLOR DEMOSOFT vous devrez utiliser la commande LOAD COLOR DEMOSOFT.

Le DOS recherche alors le fichier sur la disquette, écrase tout programme Basic en mémoire et charge le fichier. Si vous avez utilisé un nom de fichier inconnu du DOS, celui-ci affichera le message d'erreur FILE NOT FOUND.

Lorsque le DOS vous rend la main, vous pouvez lister, modifier ou exécuter votre programme. Pour exécuter un programme. Il est toutefois inutile de taper les commandes LOAD programme, puis RUN comme sur cassette. Il existe une commande du DOS réunissant ces deux commandes.

● **Exécution d'un programme Basic situé sur disquette**

La commande RUN du DOS permet de réunir en une seule commande les commandes LOAD du DOS et RUN du Basic. Son format est le suivant :

RUN fichier, Dd, Vv, Ss

Exemple :

```
]LOAD HELLO
]LOAD HELLO, D2
]RUN
DISQUETTE ESCLAVE 48K
CREEE LE 5 MAI 1982

]RUN HELLO, D2
DISQUETTE ESCLAVE 48K
CREEE LE 5 MAI 1982
```

Le DOS recherche le fichier sur la disquette, le charge en mémoire et exécute le programme correspondant. S'il ne trouve pas le fichier, il affiche le message d'erreur :

FILE NOT FOUND

et rend la main à l'utilisateur.

● **Sauvegarde d'un programme Basic sur disquette**

Sur la disquette système "SYSTEM MASTER", vous avez d'autres fichiers que le fichier à exécuter HELLO. Ces fichiers ont été sauvés sur la disquette en utilisant la commande SAVE du DOS. Vous pouvez aussi sauvegarder vos programmes sur vos disquettes. Considérons par exemple le programme :

```
 5 HOME
10 PRINT "COUCOU"
20 END
```

Mettez une disquette initialisée dans le drive 1 du contrôleur, connecteur numéro six, et tapez la commande :

SAVE COUCOU, D1, S6

Que se passe-t-il ? Le DOS sauvegarde votre programme Basic en mémoire sur la disquette. Pour en être persuadé, tapez CATALOG lorsque le DOS vous rend la main, le fichier COUCOU figure sur la disquette. Vous pouvez aussi éteindre votre APPLE, revenir au Basic et charger le DOS, et recharger votre programme (LOAD COUCOU). C'est bien le même programme. Listez-le ! Vous obtiendrez le résultat suivant :

```
 5 HOME
10 PRINT "COUCOU"
20 END
```

La commande SAVE a bien sauvegardé votre programme sur la disquette.

Si vous donnez à votre programme un nom de fichier qui existe déjà, le DOS écrasera l'ancien fichier pour le remplacer par la nouvelle version. Si par erreur les deux fichiers sont de types différents, le DOS refusera la commande et affichera le message d'erreur suivant :

FILE TYPE MISMATCH

Si le fichier que vous voulez écraser est verrouillé (cf. § 5.4.4) le DOS refusera aussi la commande et affichera le message d'erreur :

FILE LOCKED

Le format de la commande SAVE est le suivant :

SAVE nom de fichier, Vv, Ss, Dd

La commande SAVE *ne modifiant pas le programme en mémoire, vous pouvez utiliser la commande Verify pour être sûr que les secteurs qu'utilise votre sauvegarde ne sont pas abîmés physiquement.*

Exemple

```
]SAVE BONJOUR
]CATALOG
DISK VOLUME 254
*A 002 HELLO
 A 002 BONJOUR
```


— Utilisation des commandes du DOS dans un programme Basic

Vous pouvez utiliser les commandes du DOS dans vos programmes Basic. Certaines commandes du DOS ne peuvent être utilisées que dans un programme Basic : c'est le cas des commandes de manipulation de données (cf. § 1.6 : Fichiers de données).

Pour utiliser une commande du DOS dans un programme Basic, vous devez utiliser l'instruction PRINT et faire précéder la commande par le code ASCII égal à 4 (CTRL-D). Si vous voulez par exemple lister le catalogue sur l'écran, vous devez utiliser l'instruction :

```
PRINT "CATALOG"
un CTRL-D a été tapé avant le C de la commande CATALOG
```

Le caractère CTRL-D étant invisible sur l'écran, on emploie usuellement la variable D$ pour le CTRL-D.

Avec l'APPLESOFT, vous devrez mettre en tête du programme la ligne :

```
D$=CHR$(4):REM D$=CTRL-D
```

L'instruction que nous avons donnée ci-dessus devient :

```
PRINT D$; "CATALOG"
      CTRL=D pas de CTRL-D devant CATALOG
```

Vos programmes sont alors beaucoup plus lisibles et plus faciles à mettre au point.

1.5.2 Fichiers binaires — programmes assembleurs

Le DOS vous permet de sauver sur votre disquette et de charger en mémoire, à partir d'une disquette, les informations situées dans la mémoire de l'APPLE telles que sous-programmes assembleurs ou formes graphiques prédéfinies. Nous vous avons expliqué les commandes LOAD, SAVE et RUN des programmes Basic ; les commandes BLOAD, BSAVE et BRUN expliquées dans ce paragraphe ont les mêmes fonctions avec des zones mémoires désignées par leurs adresses et longueurs.

Le caractère "B", en-tête de ces trois commandes, signifie fichier binaiire. Ces fichiers sont la représentation exacte d'une zone mémoire.

● **Sauvegarde d'une zone mémoire sur disquette**

La commande du DOS appelée BSAVE permet de recopier dans un fichier binaire une zone mémoire. Le format de cette commande est le suivant :

BSAVE fichier, Aa, L1, Dd, Vv, Ss

Les paramètres A et L sont obligatoires.

Le paramètre A représente l'adresse de début de la zone mémoire à recopier dans le fichier. Cette adresse peut être fournie sous forme décimale ou hexadécimale. Dans le premier cas, elle doit être comprise entre les valeurs 0 et 65535. Les nombres négatifs, utilisés en Basic pour représentter les adresses hautes de la mémoire, ne sont pas acceptés. Dans le second cas, l'adresse doit être précédée du symbole $ et doit être comprise enttre les valeurs $0000 et $FFFF. Si l'adresse que vous fournissez ne se situe pas dans ces valeurs, le DOS affiche le message SYNTAX ERROR et abandonne la commande. Si, par contre, cette adresse ne correspond physiquement à aucun boîtier (à cause de la taille du système), le DOS ne vous dira rien, mais votre commande est nulle.

Le paramètre L représente la Longueur de la zone mémoire concernée. Il peut aussi être fourni sous forme décimale ou hexadécimale. Si la longueur n'est pas comprise entre 0 et 65535, le DOS affichera le message SYNTAX ERROR et abandonnera la commande. En fait, la longueur doit être inférieure à 32 K octets. Si ce n'est pas le cas, le moniteur affichera le message RANGE ERROR et abandonnera la commande. Si vous voulez sauvegarder une zone mémoire supérieure à 32 K octets, il vous faudra utiliser deux fois la commande BSAVE. Comme pour le paramètre adresse, le DOS ne vérifie pas la validité physique de la longueur.

Si vous voulez par exemple sauvegarder une page graphique haute résolution (ici la seconde), vous pourrez taper une des commandes ::

```
BSAVE  FHGR2,  A$4000,  L$2000,  D1,  S6
BSAVE  FHGR2,  A$4000,  L8192,  V254,  D2
BSAVE  FHGR2,  A16384,  L$2000,  S5
BSAVE  FHGR2,  A16384,  L8192
```


● **Chargement d'une zone mémoire à partir d'une disquette**

La commande BLOAD du DOS permet de restaurer une zone mémoire à partir d'un fichier sur disquette. Si vous tapez par exemple :

```
BLOAD FHGR2 (cf. ci-dessus)
POKE – 16304, 0
POKE – 16299, 0
```

Vous retrouvez sur l'écran la page graphique que vous aviez sauvegardée.

Le format de la commande BLOAD est le suivant :

BLOAD nom de fichier Aa, Ss, Dd, Vv

Le paramètre A représente, comme pour BSAVE, une adresse. Il s'agit cette fois de l'adresse de chargement. Ne précisez ce paramètre que si vous voulez charger la zone à une autre adresse que celle fournie dans le BSAVE.

La commande BLOAD écrase seulement la zone sur laquelle elle applique le fichier à la différence de la commande LOAD qui écrase tout programme Basic en mémoire.

1. *Il n'y a pas besoin de préciser une longueur dans la commande* BLOAD *car la longueur de la zone est déterminée par la longueur du fichier.*

2. *Un programme en langage machine ou en assembleur ne peut fonctionner que s'il est chargé à une certaine adresse : l'adresse à partir de laquelle il a été sauvegardé. Vous n'utiliserez généralement que la commande* BLOAD *nom de fichier.*

● **Exécution d'un programme en langage machine à partir d'une disquette**

La commande BRUN réalise une commande BLOAD puis un branchement vers l'adresse de début de la zone.

BRUN nom de fichier, As, Ss, Dd, Vv

Le paramètre A est le même pour la commande BLOAD. Si le fichier contenait un programme en assembleur, BRUN charge et lance l'exécution du programme.

1.6 LES FICHIERS DE DONNÉES

1.6.1 Généralités

Les fichiers de données sont du type texte T. Ils sont indiqués dans le catalogue par la lettre T dans la colonne du type.

Les fichiers de données peuvent être créés par un programme en un certain langage, et relus par un autre langage. Pour créer ou utiliser des fichiers de données, vous devez utiliser la possibilité, décrite au § 1.5.1, d'appeler les commandes du DOS à partir d'un programme Basic.

Les commandes suivantes ne peuvent pas être employées avec des fichiers de données :

LOAD, RUN, SAVE, BLOAD, BRUN, BSAVE

Vous devez écrire des programmes Basic pour stocker et retirer des données dans un fichier de données. Pour cela, les commandes suivantes du DOS ont été créées

OPEN
CLOSE
READ
WRITE
APPEND
POSITION
EXEC

Parmi ces commandes, seules CLOSE et EXEC sont accessibles en mode direct par l'utilisateur.

Pour les fichiers de données, vous pouvez utiliser aussi les commandes :

LOCK, UNLOCK, DELETE, RENAME, VERIFY,
CATALOG, MON et NOMON
(cf. *Commandes générales du DOS*)

Le DOS 3.3 gère deux types de fichier de données :

— les fichiers séquentiels, dans lesquels les données sont enregistrées l'une après l'autre, et qui sont lus de la même manière (cf. *Fichiers sur cassette*) ;

— les fichiers à accès direct, plus compliqués à gérer, mais beaucoup plus performants, dans lesquels on peut lire ou écrire un enregistrement

n'importe où dans le fichier, sans avoir à parcourir les enregistrements qui le précèdent.

Nous allons étudier successivement ces deux types de fichier en expliquant la manière de les utiliser et les différences entre APPLESOFT et Basic Entier.

1.6.2 La commande MAXFILES

Le DOS autorise par défaut des manipulations sur 3 fichiers simultanément. Si ceci ne vous suffit pas, vous pourrez travailler sur 16 fichiers au maximum après avoir utilisé la commande :

MAXFILES n (où n = nombre de fichiers)

Lorsque celle-ci est exécutée, le DOS déplace HIMEM (adresse mémoire maximum utilisable pour le Basic) vers le bas, pour réserver un buffer de 595 octets par fichier pour les entrées-sorties. Ceci peut détruire : en APPLESOFT, la zone mémoire occupée par les chaînes de caractères et, en Basic Entier, le programme utilisateur. Nous vous conseillons, pour éviter ce phénomène :

— de toujours préciser le nombre de fichiers nécessaires avant le chargement du programme,

— de charger et d'exécuter votre programme.

Autre solution :

```
 5 PRINT "CTRL-D MAXFILES N":END
10 REM DEBUT DU PROGRAMME
```

— exécuter le programme (MAXFILES = N),
— le recharger,
— se brancher à la ligne 10 (GOTO 10).

1.6.3 Les fichiers à accès séquentiel

● Relation entre secteurs physiques et enregistrements

Les enregistrements sont constitués de chaînes de caractères de longueur variable séparées par des codes return ( ). Les différentes données d'une chaîne sont séparées par des virgules.

La gestion des articles logiques est indépendante, pour l'utilisateur, de la gestion des secteurs physiques. C'est le DOS qui assure la liaison logique, physique.

Lors des opérations d'entrée-sortie sur un fichier séquentiel, des chaînes de caractères sont échangées entre votre programme et le fichier séquentiel. La séparation des enregistrements est opérée par le caractère Return.

● Ouverture d'un fichier séquentiel

Les fichiers séquentiels doivent être ouverts avant utilisation. Ouvrir un fichier conduit le DOS à rechercher le fichier sur la disquette, à initialiser les données de gestion du fichier et à réserver un buffer de 256 octets pour stocker en mémoire un secteur physique, à éviter de devoir accéder au disque pour chaque article, et gagner ainsi beaucoup de temps.

Le format de la commande d'ouverture d'un fichier séquentiel est le suivant :

OPEN nom du fichier Ss, Vv, Dd

Si le fichier n'existait pas sur la disquette, le DOS le crée et l'ouvre. Considérons le programme suivant :

```
1000 D$=CHR$(4):PRINT D$; "OPEN FSEQ":END
```

Tapez RUN. Le DOS ouvrira le fichier FSEQ. Tapez CATALOG. Vous verrez que le fichier FSEQ existe :

T 001 FSEQ

● Fermeture d'un fichier séquentiel

Lorsque vous avez fini de travailler sur un fichier séquentiel, vous devez le fermer pour reporter sur la disquette les modifications apportées au fichier, et libérer le buffer utilisé pour les entrées-sorties sur le fichier. De plus, si vous changez la disquette avant de fermer les fichiers, vous risquez d'avoir de très graves problèmes sur les disquettes concernées. Pensez toujours à fermer vos fichiers séquentiels à la fin de vos programmes.

La commande CLOSE permet de fermer les fichiers séquentiels. Elle admet deux formats :

CLOSE qui ferme tous les fichiers ouverts sur toutes les disquettes présentes.

CLOSE fichier Vv, Ss, Dd qui ferme le fichier séquentiel indiqué.

● Ecriture dans un fichier séquentiel

L'écriture des données se fait en utilisant l'instruction PRINT, de la même manière que sur l'écran ou l'imprimante. Vous pouvez écrire dans un fichier séquentiel tout ce que vous affichez sur l'écran. Lorsque vous écrivez dans un fichier séquentiel, le DOS met à jour un pointeur vers la première place disponible dans le fichier, tout comme une imprimante avance le papier. Ce pointeur a été mis à zéro lors de l'ouverture du fichier. Il a bien été initialisé.

Vous devez ici vous poser une question. Comment peut-on aiguiller les données à écrire vers le fichier au lieu de l'écran ? Une commande du DOS le permet. C'est la commande WRITE.

Si vous avez dans un programme la ligne PRINT D$; "WRITE FSEQ" avec D$=CTRL-D toutes les instructions PRINT rencontrées dans la suite du programme écriront dans le fichier séquentiel FSEQ.

Exemple :

```
10 D$=CHR$(4)
20 PRINT D$; "OPEN FICHIERSEQ, D2, S6"
30 PRINT D$; "WRITE FICHIERSEQ"
40 PRINT "BONJOUR"
50 PRINT D$; "CLOSE FICHIERSEQ"
60 END
```

Ce programme écrit BONJOUR comme premier article du fichier FICHIERSEQ. Il a pour cela effectué les opérations suivantes :

1. Ouverture du fichier FICHIERSEQ.
2. Aiguillage des données vers le DOS et le fichier.
3. Ecriture de l'article BONJOUR dans le fichier.
4. Fermeture du fichier pour ne pas perdre les données.

Vous pouvez mettre autant de chaînes de caractères que vous voulez dans un article. Si vous remplacez la ligne 40 par :

```
40 PRINT "BONJOUR", "OK"
```

ou

```
40 PRINT "BONJOUR,"; :PRINT "OK"
```

vous écrirez deux données dans dans le fichier. En effet, tant que l'instruction PRINT n'a pas envoyé de Return, le DOS considère que l'enregistrement en cours n'est pas terminé. Vous pouvez aussi remplacer la ligne 40 par :

```
40 PRINT A$; B$; C$
```

Le contenu des trois variables sera enregistré dans le même champ du fichier.

La séparation des données se fait par des virgules. Alors que le point-virgule concatène les données écrites pour n'en créer qu'une seule dans le fichier, la virgule stocke les données sans les concaténer.

Toutefois, une virgule placée à la fin d'une instruction PRINT supprime l'envoi de Return comme un point-virgule.

Si vous remplacez, par contre, la ligne 40 par la ligne

```
40 PRINT A$:PRINT B$:PRINT C$
```

vous aurez trois enregistrements dans votre fichier.

L'écriture de données dans un fichier se fait en utilisant l'instruction PRINT sans changer son format après aiguillage des données. Puis la commande WRITE du DOS. Lorsque vous avez fini d'écrire les données vers le fichier, il vous faut revenir à l'affichage sur l'écran. Vous pouvez le faire de trois façons :

— utilisation d'une commande du DOS,
— envoi au DOS d'une commande vide

```
PRINT D$    avec D$=CTRL-D
```

— utilisation de l'instruction INPUT dans le programme. Dans ce cas, tous les caractères à afficher avant la saisie de données, y compris le point d'interrogation, sont écrits dans le fichier.

Les données ne sont plus écrites dans le fichier, mais le sont sur l'écran ou l'imprimante.

L'affichage d'un message d'erreur arrête aussi l'effet de l'instruction WRITE *après que tout le message d'erreur ait été écrit.*

Considérons le programme suivant :

```
10 REM CREATION D'UN FICHIER SEQUENTIEL
20 D$ = CHR$ (4)
30 HOME
40 PRINT "   CE PROGRAMME VOUS PERMET DE CREER" ;
50 PRINT "UN FICHIER SEQUENTIEL"
60 PRINT "POUR CELA VOUS ENTREREZ UNE CHAINE DE " ;
70 PRINT "CARACTERES DE LONGUEUR COMPRISE ENTRE 1" ;
80 PRINT " ET 239 CARACTERES"
90 PRINT
100 PRINT "LORSQUE VOUS VOUDREZ ARRETER, VOUS TAPEREZ RETURN"
110 PRINT D$ ; "OPEN FSEQ"
120 HOME : I = 1
130 PRINT "ENTREZ LA CHAINE NO "; I ; " : " ;
140 INPUT ""; A$
150 IF A$ = "" GOTO 210
160 PRINT D$ ; "WRITE FSEQ"
170 PRINT A$
180 PRINT D$
190 I = I + 1
200 GOTO 130
210 PRINT D$ ; "CLOSE"
220 END
```

Essayez-le ! Ce programme ouvre le fichier séquentiel FSEQ et écrit les chaînes de caractères saisies au clavier à partir du début du fichier.

Supposons que vous tapiez la première fois les chaînes BONJOUR et OK, vous obtiendrez le fichier :

```
BONJOUR)OK)
```

Relancez le programme et tapez par exemple la chaîne OK seulement. Vous obtiendrez le fichier :

```
OK)JOUR)OK)
```

Qu'observez-vous ? Le début du fichier a été écrasé et remplacé par OK. Mais les informations suivantes n'ont pas été effacées. En effet, lorsque vous ouvrez un fichier, le DOS met le pointeur sur le début du fichier, mais ne détruit pas les informations précédemment écrites. Il est donc normal de conserver, dans certains cas, une partie des anciennes données.

Que faire pour éviter ce phénomène ? Deux positions sont possibles ::

— soit vous désirez conserver les anciennes données et il vous faut positionner le pointeur à la fin du fichier (cf. commandes APPEND et POSITION ci-après) ;

— soit vous ne voulez pas conserver les anciennes données et la meilleure solution est de détruire le fichier en utilisant la commande DELETE. Un nouveau problème se pose alors : si le fichier n'existe pas lors de l'exécution de la commande DELETE, le DOS créera une erreur et le programme s'arrêtera. Une solution très simple consiste à utiliser la commamde OPEN avec la commande DELETE. De cette manière, soit le fichier existait déjà et il est ouvert, soit le fichier n'existait pas et il est créé. La commande DELETE trouvera donc toujours le fichier à détruire.

Pour rajouter cette solution au programme précédent, il vous suffit de rajouter les deux lignes suivantes :

```
108  PRINT D$ ; "OPEN FSEQ"
109  PRINT D$ ; "DELETE FSEQ"
```

De cette manière, vous n'aurez jamais de superposition de données entre deux versions d'un fichier séquentiel.

● Lecture des données d'un fichier séquentiel

De la même manière que des données peuvent être aiguillées vers un fichier, des données peuvent être lues à partir d'un fichier. La commamde READ fichier avertit le DOS que les données ne seront plus saisies au clavier, mais simplement lues dans le fichier indiqué.

Si vous avez dans votre programme la ligne :

```
PRINT D$ ; "READ FSEQ"
```

Les données, demandées par le programme, ne seront plus saisies au clavier, mais lues dans le fichier.

Les données sont lues en utilisant l'instruction Basic Input.

Par exemple, si vous avez utilisé l'instruction suivante pour créer le fichier séquentiel :

```
PRINT A$, ", ", B$, ", ", C$, ", ", E$
```


vous devez utiliser l'instruction

```
INPUT G$, F$, S$, O$
```

pour relire le fichier (même nombre que paramètres dans le PRINT et l'INPUT).

Si l'instruction Input a moins de paramètres que l'instruction PRINT, il reste une donnée dans l'enregistrement qui serait perdue car le prochain Input utiliserait l'enregistrement suivant. Il s'ensuit le message d'erreur

```
? EXTRA IGNORED.
```

Si, au contraire, il y a plus de paramètres dans l'instruction Input qu'il n'y en avait dans l'instruction PRINT, les données présentes sont affectées aux premiers paramètres ; mais certains paramètres n'ont pas reçu de valeur ou de donnée : le DOS lit alors l'enregistrement suivant. Si celui-ci n'existe pas, le DOS déclenche alors l'erreur END of DATA. Si, dans le nouvel enregistrement, le nombre de données est inférieur au nombre de paramètres auxquels le DOS doit affecter une valeur, il recommence la même opération avec l'enregistrement suivant. Si le nombre de données correspond au nombre de paramètres auxquels le DOS doit affecter une valeur, il n'y a pas d'erreur provoquée. Par contre s'il y a trop de données dans le dernier enregistrement lu, le DOS provoque l'erreur

```
? EXTRA IGNORED.
```

Vous avez donc intérêt à mettre le même nombre de données dans l'instruction PRINT correspondante.

Considérons les programmes suivants :

Programme 1 : création du fichier séquentiel

```
 10 D$ = CHR$ (4)
 20 PRINT D$;"OPEN FSEQ"
 30 PRINT D$;"DELETE FSEQ"
 40 PRINT D$;"OPEN FSEQ"
 50 PRINT D$;"WRITE FSEQ"
 60 PRINT "CHAT,CHIEN "
 70 PRINT "FRUIT,APPLE  ,AIGLE "
 80 PRINT "CHEVAL"
 90 PRINT "D$;"CLOSE FSEQ"
100 END
```

Programme 2 : lecture du fichier

```
 10 D$ = CHR$ (4)
 20 PRINT D$;"OPEN FSEQ"
 30 PRINT D$;"READ FSEQ"
 40 INPUT A$,B$
 50 INPUT C$,D1$,E$
 60 INPUT F$
 70 PRINT D$;"CLOSE"
 80 HOME
 90 PRINT A$,B$,C$,D1$,E$,F$
100 END
```

Programme 3 : idem programme 2 sauf

```
 50 INPUT C$,D1$,E$,F$,G$
```

Programme 4 : idem programme 2 sauf

```
 40 INPUT A$
 50 INPUT B$,C$,D1$,E$
```

Faites exécuter le programme 1, puis chacun des trois autres programmes. Que se passe-t-il ? Le programme 2 se déroule bien, mais les programmes 3 et 4 se terminent par l'erreur ? EXTRA IGNORED.

Essayez de comprendre pourquoi en relisant les explications ci-dessus.

La commande READ du DOS est annulée de la même façon que la commande WRITE, c'est-à-dire par une erreur ou par une nouvelle commande au DOS.

1. *L'utilisation de plusieurs données n'est possible qu'en* APPLESOFT, *car les Basic de l'*APPLE *n'utilisent pas la virgule de la même manière comme séparateur dans les instructions* PRINT *et* INPUT. *La virgule n'est pas considérée par l'instruction Input du Basic Entier comme séparateur de chaînes de caractères.*

2. *Les virgules utilisées dans les instructions* PRINT (*Basic Entier*) *ne sont pas prises en compte lorsque l'on écrit dans un fichier. Les virgules jouent alors le même rôle que des points-virgules.*

L'instruction PRINT "HELLO", "APPLE" *stocke dans le fichier la donnée* HELLOAPPLE *en Basic Entier.*

Les deux chaînes sont concaténées.

● Ajout de données dans un fichier séquentiel

Lorsque vous utilisez la commande OPEN du DOS, celui-ci ouvre le fichier et positionne le pointeur vers le premier enregistrement. Que faire pour ajouter des données à la fin du fichier ?

— Lisez tous les enregistrements, vous éviterez l'erreur END OF DATA et vous pourrez alors écrire de nouvelles données dans le fichier ;

ou bien :

— Utilisez la commande APPEND qui remplace la commande OPEN (celle-ci devient inutile). Le DOS ouvre le fichier et positionne le pointeur après le dernier enregistrement du fichier. Vous pourrez alors écrire de nouvelles données dans le fichier.

Considérons les programmes suivants :

Programme 1 :

```
10 D$ = CHR$(4)
20 PRINT D$ ; "OPEN FSEQ"
30 PRINT D$ ; "WRITE FSEQ"
40 PRINT "APPLE   "
50 PRINT D$ ; "CLOSE"
60 END
```

Programme 2 :

```
10 D$ = CHR$(4)
20 PRINT D$ ; "APPEND FSEQ"
30 PRINT D$ ; "WRITE FSEQ"
40 PRINT "FDS"
50 PRINT D$ ; "CLOSE"
60 END
```

Programme 3 :

```
10 D$ = CHR$(4)
20 PRINT D$ ; "OPEN FSEQ"
30 PRINT D$ ; "READ FSEQ"
40 INPUT A$
50 INPUT B$
60 PRINT D$ ; "CLOSE"
70 PRINT A$, B$
80 END
```

Faites exécuter les programmes 1, 2 et 3 dans l'ordre 1, 2, 3. Que se passe-t-il ? Le programme 1 crée le fichier séquentiel et y écrit la donnée "APPLE". Le programme 2 ouvre le fichier, se positionne en fin de fichier (APPEND) et y écrit la donnée "FDS". Le fichier contient les données APPLE et FDS ; ce qui est bien vérifié par le programme 3. Essayez de comprendre pourquoi en relisant les explications ci-dessus.

1. *Si la commande* APPEND *ne trouve pas le fichier, le* DOS *provoque l'erreur* FILE NOT FOUND *et arrête l'exécution du programme.*
2. *Les problèmes suivants se produisent avec la commande* APPEND :

   — *Si votre fichier se termine à la frontière d'un secteur* (128 *octets*), *cette commande positionnera le pointeur au début du fichier et vous « l'écraserez ». Pour éviter ce risque, il faut écrire une marque de fin de fichier à la fin de celui-ci avant sa fermeture.*

   En effet, le DOS *ne rajoute pas de marque de fin de fichier à la fermeture, car tous les secteurs qu'il attribue au fichier sont remplis au départ de ces marques. Lorsque vous écrivez les* 128 *octets, toutes les marques sont écrasées.*

   — *Si votre fichier dépasse* 128 *secteurs, la commande* APPEND *positionnera le pointeur sur le premier qui suit la marque de fin de fichier qui n'est donc pas effacée. A la prochaine occurrence de* APPEND, *les nouvelles données introduites seront perdues.*

● Positionnement du prochain enregistrement à lire ou à écrire

La commande POSITION du DOS permet de déplacer le pointeur d'un certain nombre d'enregistrements dans le fichier. Si l'enregistrement courant est l'enregistrement numéro 1 et que vous voulez lire par exemple l'enregistrement numéro 5, vous devrez insérer l'instruction :

PRINT D$; "POSITION FSEQ, R4"

Le format exact de la commande POSITION est le suivant :

POSITION nom de fichier, Rp où p est la position relative de l'enregistrement sur laquelle se positionner par rapport à l'enregistrement courant ($p^{ème}$ enregistrement après le courant). Si p = 0, le pointeur n'est pas modifié. Si p = 1, le pointeur passe sur l'enregistrement suivant ; etc.

Si l'enregistrement sur lequel on doit se positionner n'existe pas (fin de fichier), le DOS provoque l'erreur END OF DATA et stoppe l'exécution du programme. La commande POSITION examine en fait le fichier octet pour déterminer les enregistrements. L'erreur précédente se produit aussi si un caractère interdit est trouvé (par exemple un Nul, code ASCII 0).

Considérons l'exemple suivant qui va de pair avec les programmes 1, 2, 3 du paragraphe précédent.

Programme 4 :

```
10 D$ = CHR$ (4)
20 PRINT D$;"OPEN FSEQ"
30 PRINT D$;"POSITION FSEQ,R1"
40 PRINT D$;"READ FSEQ"
50 INPUT A$
60 PRINT D$;"CLOSE"
70 HOME
80 PRINT A$
90 END
```

Une fois le fichier FSEQ créé par les programmes 1 et 2, lancez l'exécution du programme 4. Vous voyez alors s'afficher FDS sur l'écran.

La commande utilise des déplacements relatifs par rapport à l'enregistrement. Toutefois, si la commande POSITION *est utilisée avant toute opération de lecture ou d'écriture, le déplacement devient absolu* (*par rapport au début du fichier*).

● Accès à un fichier à partir d'un octet donné

Les commandes du DOS : WRITE et READ peuvent être utilisées avec un paramètre qui désigne l'octet du fichier sur lequel doit être positionné le pointeur. La lecture (ou l'écriture) des données commencent alors à partir de cet octet.

L'utilisation de cette commande vous oblige à connaître la structure exacte des données dans le fichier (return, ; et autres caractères). La

numération des octets dans le fichier commence à 0. L'option B30 revient à se positionner sur le 31e octet du fichier ;

```
WRITE FSEQ, B30
READ
```

Cette possibilité peut être utilisée avec la commande POSITION. L'octet indiqué est alors situé par rapport au début de l'enregistrement.

Cette possibilité est à éviter sauf si vos données suivent un format fixe ou si vous êtes expérimenté.

● Exemple

L'emploi des fichiers séquentiels étant limité, nous vous conseillons de passer directement aux fichiers à accès direct. Pour cette raison, nous ne donnons pas d'exemple d'applications à base de fichiers séquentiels. Seul l'exemple du paragraphe 1.7.5, traitant la commande EXEC, utilisera les fichiers à accès séquentiel.

1.6.4 Les fichiers à accès direct

● Relation entre secteurs physiques et enregistrements

Les fichiers à accès direct sont des fichiers de données de type texte, qui se distinguent des fichiers séquentiels par le fait que les enregistrements sont de taille fixe. Si vous écrivez un enregistrement plus petit que la taille fixe, cet enregistrement est complété par des NULS (code ASCII égal à 0).

La structure des informations sur la disquette est la suivante :

B O N J O U R , 0 K	A P P L E . . . . .
10 octets	10 octets

C'est à vous de décider la taille des enregistrements. Le DOS assure alors la gestion physique des secteurs de la disquette. Lorsque la taille d'un fichier nécessite l'utilisation d'un nouveau secteur physique, le DOS initialise le secteur avec des NULS. Vos données écraseront ensuite certains NULS.

Le principal intérêt des fichiers à accès direct réside dans le fait que l'on peut accéder à un enregistrement sans avoir lu ou écrit tous les enregistrements précédents. La taille des enregistrements étant fixe, le DOS peut alors calculer la position de l'enregistrement sur la disquette et y accéder.

● Ouverture d'un fichier à accès direct

L'ouverture d'un fichier à accès direct se fait à l'aide de la même commande OPEN que pour un fichier séquentiel, en ajoutant toutefois le paramètre L qui caractérise la longueur de chaque enregistrement.

Si vous utilisez la commande :

```
OPEN FDIRECT, L10
```

vous ordonnez au DOS d'ouvrir le fichier FDIRECT avec une longueur d'enregistrement égale à 10 octets et, si le fichier FDIRECT n'existe pas, de le créer.

Le format de la commande OPEN est le suivant :

```
OPEN fichier, L1, Ss, Vv, Dd
```

Le paramètre 1 doit être compris entre les valeurs 1 et 32767.

Si le fichier que l'on ouvre a été créé avec une longueur différente de celle que l'on a précisé dans la commande OPEN, *c'est cette dernière longueur qui est prise en compte par le* DOS. *Il n'y a aucune manière de connaître la longueur qui a été utilisée lors de la création du fichier. Vous devez conserver dans votre documentation la longueur des enregistrements du fichier. Vous pouvez par exemple inclure cette donnée dans le nom du fichier (par exemple* FDIRECT:L10).

Autre solution : conservez les données de gestion du fichier dans le premier enregistrement (numéro 0). *C'est cette deuxième solution que nous retiendrons de manière standard.*

● Fermeture d'un fichier à accès direct

La fermeture d'un fichier se fait de la même manière que pour un fichier séquentiel. La commande CLOSE est utilisée avec les mêmes paramètres selon le format suivant :

```
CLOSE fichier, Dd, Ss, Vv
```

● Ecriture dans un fichier à accès direct

L'écriture de données dans un fichier à accès direct se fait en utilisant l'instruction PRINT. Comme pour un fichier séquentiel, il faut modifier l'aiguillage des données vers le DOS, en utilisant la commande WRITE du DOS.

Pour écrire dans un fichier à accès direct, la commande WRITE du DOS a un paramètre supplémentaire : le numéro de l'enregistrement que vous voulez modifier. Un fichier à accès direct peut en effet être utilisé sans lire tous les enregistrements du DOS. Vous devez pour cela donner le numéro de l'enregistrement sur lequel doit se positionner le pointeur enregistrement.

Les numéros d'enregistrement commencent à la valeur zéro pour le premier enregistrement du fichier.

Si vous utilisez l'instruction :

PRINT D$; "WRITE FDIRECT, R4"

le DOS positionnera le pointeur sur le 5e enregistrement du fichier et les prochaines instructions PRINT écriront des données dans ce 5e enregistrement..

Si la longueur des données écrites dans un enregistrement est supérieure à la longueur de l'enregistrement déterminée par la commande OPEN, vous aurez un écrasement des données des enregistrements suivants. Vous aurez donc intérêt à éviter ce genre de problème.

Si vous ne précisez pas le numéro d'enregistrement dans la commande WRITE, le pointeur n'est pas modifié et reste sur l'enregistrement courant.

Le format exact de la commande WRITE est le suivant :

WRITE fichier Rr, Ss, Dd, Vv

Un enregistrement de fichier à accès direct peut comporter un ou plusieurs champs (données) en APPLESOFT. Il revient au même d'utiliser les instructions

PRINT "BONJOUR, COMMENT ALLEZ-VOUS" ;
PRINT "APPLE"
et
PRINT "BONJOUR," ; "COMMENT ALLEZ-VOUS", "APPLE"

Dans les deux cas un enregistrement contenant trois données est créé.

Par contre, un enregistrement de fichier à accès direct ne contient qu'une seule donnée en Basic entier. Dans le cas ci-dessus, la donnée suivante est créée :

BONJOUR, COMMENT ALLEZ-VOUS, APPLE

Une question peut vous venir à l'esprit après avoir lu ce paragraphe.

Comment rajouter des données à la fin d'un fichier à accès direct ?

La commande APPEND n'existe pas pour les fichiers à accès direct, et n'est pas applicable car elle considère avoir atteint la fin du fichier lorsqu'elle rencontre un NULL (code ASCII nul). Or ce même caractère est utilisé pour remplir les trous entre les données de deux enregistrements successifs. Vous devez donc assurer la gestion du numéro du prochain enregistrement à lire. La meilleure solution consiste à stocker ce numéro dans le premier enregistrement (numéro 0) du fichier et à aller le relire lorsque vous voudrez ajouter un enregistrement à la fin du fichier.

Il est par contre beaucoup plus facile de modifier un enregistrement dans un fichier à accès direct que dans un fichier séquentiel. En effet vous pouvez vous positionner au début de tout enregistrement et la longueur d'un enregistrement étant donnée, vous avez beaucoup moins de mal à ne pas écraser les enregistrements suivants.

Si votre application consiste à changer assez souvent des données (exemple : carnet d'adresses) vous aurez beaucoup plus intérêt à utiliser les fichiers à accès direct.

● Lecture de données dans un fichier à accès direct

La lecture de données dans un fichier à accès direct se fait en utilisant l'instruction INPUT. Elle lit les données à partir d'un fichier à accès direct lorsque la commande READ du DOS est active. Son format est le suivant :

READ fichier, Rr, Ss, Dd, Vv

Comme pour un fichier séquentiel, le nombre de données de l'instruction INPUT doit être le même que l'instruction PRINT correspondante.

De plus, vous devez avoir toujours le même nombre de données quel que soit l'enregistrement concerné. Par exemple, le DOS 3.3 n'accepte pas d'écrire 0 dans l'enregistrement 0, 0 et 1 dans l'enregistrement 1, etc. ; mais se plantera à la lecture même si les instructions PRINT et INPUT concordent.

Le mode d'utilisation de l'instruction GET est identique.

● Accès à un octet donné

Comme pour un fichier séquentiel, il est possible de positionner le pointeur du fichier vers un octet donné du fichier (cf. § 5.6.3). Cette possibilité s'applique aussi pour un octet d'enregistrement donné. Par exemple, si vous utilisez la commande du DOS suivante :

READ FDIRECT, R5, B4

le DOS positionnera le pointeur sur l'octet numéro 4 (5e octet de l'enregistrement) de l'enregistrement numéro 5 (6e enregistrement du fichier).

Comme pour les fichiers séquentiels, nous déconseillons l'emploi de cette possibilité au débutant car il faut connaître la structure du fichier ou de l'enregistrement octet par octet. Toutefois, elle est utile si le format des enregistrements est fixe.

● Exemple

Programme de gestion d'un carnet d'adresses écrit en Basic APPLESOFT, reprenant les sous-programmes présentés au tome 1, chapitre 2, et ajoutant la gestion du fichier :

Lignes 10-200 : initialisations

Lignes 210-320 : création du fichier si nécessaire

Lignes 330-420 : lecture du nombre d'enregistrements du fichier

Lignes 430-580 : menu

Lignes 590-640 : début des sous-programmes

Lignes 650-1 380 : recherche dans le fichier :
- — définition du mode de recherche
- — parcours du fichier et test de concordance
- — appel du sous-programme de modification/suppression d'un enregistrement

Lignes 1 390-1 790 : ajout d'inscriptions :
- — saisie des données
- — écriture en fin de fichier

Lignes 1 800-1 990 : listing du fichier

Lignes 2 000-2 070 : fin du programme

Lignes 2 080-2 130 : début des sous-programmes du second niveau

Lignes 2 140-2 600 : modification/suppression d'un enregistrement :
- lignes 2 140-2 260 menu
- lignes 2 270-2 520 modification
- lignes 2 530-2 600 suppression

Lignes 2 610-2 700 : lecture du fichier

Lignes 2 710-2 760 : écriture dans le fichier :
- — saisie du nom
- — saisie du prénom
- — saisie du genre de voie
- — saisie du nom de la voie
- — saisie du numéro
- — saisie de la ville
- — saisie du code postal
- — saisie de l'indicatif et du numéro de téléphone

Lignes 3 630-3 720 : affichage du message d'erreur « PAS PLUS DE N CARACTÈRES »

Lignes 3 730-3 830 : affichage des données saisies

Lignes 3 840-3 930 : cadrage d'une donnée parmi N blancs.

```
10 ' CARNET D'ADRESSES
20 '
30 ' COPYRIGHT B.DE MERLY
40 '
50 ' 11/7/82
60 '
70 ' CREATION DU FICHIER ?
80 '
90 HOME:INVERSE
100 PRINT "      CARNET D'ADRESSES  ":PRINT
110 NORMAL
120 INPUT "EST-CE LA CREATION DU FICHIER (O/N)",A$
130 IF A$<>"N" AND A$<>"O" THEN 120
140 IF A$="N" THEN 320
150 '
160 ' CREATION DU FICHIER
170 '
180 PRINT"EN CREATION LE FICHIER PRECEDENT":PRINT" EST SUPPRIME"
190 INPUT"CONTINUEZ-VOUS QUAND MEME (O/N)",A$
200 IF A$<>"N" AND A$<>"O" THEN 190
210 IF A$="N" THEN 320
220 GOSUB 2920
```

```
230 CLOSE
240 KILL "FICHAD"
250 GOSUB 2930 'OUVERTURE FICHIER
260 LSET NB$=MKI$(1)
270 LSET SV$=MKI$(0)
280 LSET PV$=MKI$(2)
290 PUT#1,1
300 CLOSE
310 '
320 'LECTURE DU NOMBRE D'ENREGISTREMENTS
330 ' DANS LE FICHIER
340 '
350 GOSUB 2930 'OUVERTURE FICHIER
360 GET#1,1
370 NB=CVI(NB$):SV=CVI(SV$):PV=CVI(PV$)
380 CLOSE
390 '
400 ' DEFINITION DU FICHIER
410 '
420 GOSUB 2970
430 '
440 '   PROPOSITION DU MENU
450 '   ********************
460 '
470 HOME:INVERSE
480 PRINT"    *** CARNET D'ADRESSES ***":PRINT:PRINT:PRINT
490 PRINT"1- CONSULTATION AVEC/SANS MODIFICATION"
500 PRINT"2- INSCRIPTION (NOUVELLE ADRESSE)"
510 PRINT"3- LISTE DU FICHIER"
520 PRINT"4- FIN DU PROGRAMME"
530 PRINT
540 INPUT"ENTREZ VOTRE CHOIX (1,2,3,4)";A
550 IF A<1 OR A>4 THEN 470
560 NORMAL
570 ON A GOSUB 640,1490,1960,2180
580 GOTO 470
590 '
600 '
610 '     DEBUT DES SOUS-PROGRAMMES
620 '     *************************
630 '

650  REM  ***********************************
660  REM   RECHERCHE DANS LE FICHIER (1)
670  REM  ***********************************
680  REM
690  HOME : INVERSE : PRINT "   *** CONSULTATION ***": NORMAL
700  PRINT : PRINT : PRINT
710  PRINT "A PARTIR DE QUOI VA SE FAIRE LA RECHERCHE"
720  PRINT
730  PRINT "1-LE NOM"
740  PRINT "2-LE PRENOM"
750  PRINT "3-L'ADRESSE"
760  PRINT "4-LA VILLE"
770  PRINT "5-LE CODE POSTAL"
```



```
780  PRINT "6-LE NUMERO DE TELEPHONE"
790  PRINT : PRINT : INPUT "VOTRE CHOIX (ACCOLER LES DIVERS CHIFFRES) ";B$
800 RECH$(1,2) = "NOM A RECHERCHER"
810 RECH$(2,2) = "PRENOM A RECHERCHER"
820 RECH$(3,2) = "ADRESSE A RECHERCHER"
830 RECH$(4,2) = "VILLE A RECHERCHER"
840 RECH$(5,2) = "CODE POSTAL A RECHERCHER"
850 RECH$(6,2) = "NUMERO DE TELEPHONE A RECHERCHER"
860 RECH(1,2) = 15:RECH(2,2) = 15:RECH(3,2) = 15:RECH(4,2) = 15:RECH(5,2) =
    5:RECH(6,2) = 7
870 B =  LEN (B$)
880  FOR I = 1 TO 6
890 RECH(I,1) = 0
900  FOR J = 1 TO B
910  IF  VAL ( MID$ (B$,J,1)) <  > I THEN 950
920 RECH(I,1) = 1
930  PRINT RECH$(I,2): INPUT " ";A$
940 N = RECH(I,2): GOSUB 3840:RECH$(I,1) = A$
950  NEXT J
960  NEXT I
970  REM
980  REM  RECHERCHE PROPREMENT DITE
990  REM  ************************
1000  REM
1010  IF NB > 1 THEN 1030
1020  PRINT "FICHIER VIDE": INPUT "TAPEZ RETURN POUR CONTINUER";A: RETURN
1030  FOR I = 1 TO NB - 1
1040 OK = 1
1050  PRINT RD$ +  STR$ (I)
1060  GOSUB 2650
1070  PRINT D$
1080  REM
1090  REM  TEST DE CONCORDANCE
1100  REM  ********************
1110  REM
1120  REM  ARTICLE LU-DONNEES RECHERCHEES
1130  REM  ******************************
1140  REM
1150 OK = 1
1160  IF RECH(1,1) = 0 THEN 1180
1170 OK = OK AND (NOM$ = RECH$(1,1))
1180  IF RECH(2,1) = 0 THEN 1200
1190 OK = OK AND (PRENOM$ = RECH$(2,1))
1200  IF RECH(3,1) = 0 THEN 1230
1210 A$ = NUMERO$ + " " + GENREVOIE$ + " " + NVOIE$
1220 OK = OK AND A$ = RECH$(3,1))
1230  IF RECH(4,1) = 0 THEN 1250
1240 OK = OK AND (VILLE$ = RECH$(4,1))
1250  IF RECH(5,1) = 0 THEN 1270
1260 OK = OK AND (CDPST$ = RECH$(5,1))
1270  IF RECH(6,1) = 0 THEN 1290
1280 OK = OK AND (TEL$ = RECH$(6,1))
1290  IF OK = 0 THEN 1370
1300  PRINT "1375"
1310  HOME : INVERSE : PRINT "DONNEES POSSIBLES": NORMAL
```

```
1320  PRINT : PRINT : PRINT : GOSUB 3770: PRINT
1330  PRINT "DESIREZ-VOUS MODIFIER OU SUPPRIMER"
1340  INPUT "CET ENREGISTREMENT (O/N) ";A$
1350  IF A$ <  > "N" AND A$ <  > "O" THEN 1330
1360  IF A$ = "O" THEN  GOSUB 2140
1370  NEXT I
1380  RETURN
1390  REM
1400  REM  ************************************
1410  REM      AJOUT D'INSCRIPTIONS (2)
1420  REM  ************************************
1430  REM
1440  INVERSE
1450  HOME : PRINT "NOUVELLE ADRESSE": PRINT : PRINT
1460  NORMAL
1470  PRINT "SI UNE REPONSE NE PEUT ETRE FOURNIE"
1480  PRINT "TAPEZ SEULEMENT RETURN"
1490  PRINT
1500 CODE$ = "1"
1510  GOSUB 2770: REM   SAISIE DU NOM
1520 B$ = NOM$
1530  GOSUB 2870: REM  SAISIE DU PRENOM
1540  GOSUB 2970: REM  SAISIE DU GENRE DE VOIE
1550  GOSUB 3070: REM  SAISIE DU NOM DE LA VOIE
1560  GOSUB 3170: REM  SAISIE DU NUMERO
1570  GOSUB 3270: REM  SAISIE DE LA VILLE
1580  GOSUB 3370: REM  SAISIE DU CODE POSTAL
1590  GOSUB 3470: REM  SAISIE DU TEL
1600  HOME
1610  INVERSE
1620  PRINT "INFORMATIONS SAISIES"
1630  NORMAL : PRINT : PRINT
1640  GOSUB 3770: REM  AFFICHAGE BUFFER
1650  PRINT : INVERSE
1660  INPUT "VALIDEZ-VOUS CES INFORMATIONS (O/N)";A$
1670  IF A$ <  > "N" AND A$ <  > "O" THEN 1660
1680  IF A$ = "N" THEN 1410
1690  REM
1700  REM  AJOUT DE L'ARTICLE DANS LE FICHIER
1710  REM
1720  PRINT WR$;NB
1730  GOSUB 2750
1740 NB = NB + 1
1750  PRINT WR$ + "O": PRINT  STR$ (NB)
1760  PRINT D$
1770  INPUT "TAPEZ RETURN POUR CONTINUER";A$
1780  RETURN
1790  REM
1800  REM  ******************
1810  REM  LISTING DU FICHIER
1820  REM  ******************
1830  REM
1840  IF NB = 1 THEN 1970
1850  FOR I = 1 TO NB - 1
1860  PRINT RD$;I
```



```
1870  GOSUB 2650
1880  PRINT D$
1890  IF CODE$ <  > "1" THEN 1960
1900  HOME : INVERSE
1910  PRINT "ARTICLE NUMERO ";I
1920  NORMAL
1930  PRINT
1940  GOSUB 3770
1950  INPUT "TAPEZ RETURN POUR CONTINUER ";A$
1960  NEXT
1970  HOME : INVERSE : PRINT "FIN DE FICHIER": NORMAL
1980  INPUT "TAPEZ RETURN POUR CONTINUER ";A$
1990  RETURN
2000  REM
2010  REM  ************************************
2020  REM      FIN DU PROGRAMME (4)
2030  REM  ************************************
2040  REM
2050  PRINT WR$ + "O": PRINT  STR$ (NB)
2060  PRINT D$ + "CLOSE"
2070  NORMAL
2080  END
2090  REM
2100  REM  DEBUT DES SOUS-PROGRAMMES
2110  REM  DU SECOND NIVEAU
2120  REM  *************************
2130  REM
2140  REM
2150  REM  MODIFICATION/SUPPRESSION
2160  REM  ************************
2170  REM
2180  REM  D'UN ENREGISTREMENT
2190  REM  ******************
2200  REM
2210  PRINT : PRINT "SOUHAITEZ-VOUS"
2220  PRINT "MODIFIER L'ARTICLE  (1)"
2230  PRINT "SUPPRIMER L'ARTICLE (2)"
2240  INPUT "VOTRE CHOIX";A
2250  IF A < 1 OR A > 2 THEN 2210
2260  ON A GOTO 2270,2530
2270  REM
2280  REM  MODIFIER UN ARTICLE
2290  REM  ******************
2300  REM
2310  PRINT : PRINT "QUE SOUHAITEZ-VOUS MODIFIER"
2320  PRINT : PRINT "1-LE NOM": PRINT "2-LE PRENOM": PRINT "3-L'ADRESSE":
      PRINT "4-LA VILLE": PR
INT "5-LE CODE POSTAL": PRINT "6-LE
NUMERO DE TELEPHONE"
2330  PRINT : INPUT "VOTRE CHOIX (1,2,3,4,5,6) ";A
2340  IF A <  > 3 THEN 2390
2350  GOSUB 2970: REM  GENREVOIE
2360  GOSUB 3070: REM  NOM DE LA VOIE
2370  GOSUB 3170: REM  NUMERO
2380  GOTO 2400
```

```
2390  ON A GOSUB 2770,2870,,3240,3340,3440
2400  HOME : INVERSE : PRINT "INFORMATIONS SAISIES": NORMAL
2410  GOSUB 3830
2420  PRINT : PRINT "SOUHAITEZ-VOUS"
2430  PRINT "1-MODIFIER UNE AUTRE DONNEE DE L'ARTICLE"
2440  PRINT "2-ENREGISTRER LES MODIFICATIONS"
2450  PRINT "3-ABANDONNER LES MODIFICATIONS"
2460  INPUT "VOTRE CHOIX (1,2,3)";A
2470  ON A GOTO 2270,2480,2520
2480  REM
2490  REM  ENREGISTREMENT DES MODIFICATIONS
2500  REM
2510  PRINT WR$;I: GOSUB 2750: PRINT D$
2520  RETURN
2530  REM
2540  REM  SUPPRESSION D'UN ARTICLE
2550  REM  ************************
2560  REM
2570  PRINT WR$;I
2580 CODE$ = "0"
2590  GOSUB 2750
2600  PRINT D$:RETURN
2610  REM
2620  REM   LECTURE FICHIER
2630  REM    ***************
2640  REM
2650  INPUT B$
2660 CODE$ =  MID$ (B$,1,1):NOM$ =  MID$ (B$,2,15):PRENOM$ =  MID$ (B$,17,15)
2670 GENREVOIE$ =  MID$ (B$,32,3):NVOIE$ =  MID$ (B$,35,15):NUMERO$ =
     MID$ (B$,50,3)
2680 VILLE$ =  MID$ (B$,53,15):CDPST$ =  MID$ (B$,68,5)
2690 INDTEL$ =  MID$ (B$,73,2):TEL$ =  MID$ (B$,75,7)
2700  RETURN
2710  REM
2720  REM  ECRITURE DANS LE FICHIER
2730  REM  ************************
2740  REM
2750  PRINT CODE$;NOM$;PRENOM$;GENREVOIE$;NVOIE$;NUMERO$;VILLE$;CDPST$;
     INDTEL$;TEL$
2760  RETURN
2770  REM
2780  REM  SAISIE DU NOM
2790  REM  *************
2800  REM
2810  INPUT "NOM ";A$
2820 N = 15: IF  LEN (A$) <  = N THEN 2850
2830  GOSUB 3690: REM  AFFICHAGE MESSAGE D'ERREUR
2840  GOTO 2810
2850  GOSUB 3840:NOM$ = A$
2860  RETURN
2870  REM
2880  REM  SAISIE DU PRENOM
2890  REM  ****************
2900  REM
2910  INPUT "PRENOM ";A$
```



```
2920 N = 15: IF  LEN (A$) <  = N THEN 2950
2930  GOSUB 3690: REM  AFFICHAGE MESSAGE D'ERREUR
2940  GOTO 2910
2950  GOSUB 3840:PRENOM$ = A$
2960  RETURN
2970  REM
2980  REM  SAISIE DU GENRE DE VOIE
2990  REM  ************************
3000  REM
3010  INPUT "GENRE DE VOIE(RUE,AV,BLD,...) ";A$
3020 N = 3: IF  LEN (A$) <  = N THEN 3050
3030  GOSUB 3690: REM  AFFICHAGE MESSAGE D'ERREUR
3040  GOTO 3010
3050  GOSUB 3840:GENREVOIE$ = A$
3060  RETURN
3070  REM
3080  REM  SAISIE DU NOM DE LA RUE
3090  REM  ************************
3100  REM
3110  INPUT "NOM DE LA VOIE ";A$
3120 N = 15: IF  LEN (A$) <  = N THEN 3150
3130  GOSUB 3690: REM  AFFICHAGE MESSAGE D'ERREUR
3140  GOTO 3110
3150  GOSUB 3840:NVOIE$ = A$
3160  RETURN
3170  REM
3180  REM  SAISIE DU NUMERO
3190  REM  ****************
3200  REM
3210  INPUT "NUMERO ";A$
3220 N = 3: GOSUB 3840:NUMERO$ = A$
3230  RETURN
3240  REM
3250  REM  SAISIE DE LA VILLE
3260  REM  ******************
3270  REM
3280  INPUT "VILLE ";A$
3290 N = 15: IF  LEN (A$) <  = N THEN 3320
3300  GOSUB 3640: REM  AFFICHAGE MESSAGE D'ERREUR
3310  GOTO 3280
3320  GOSUB 3840:VILLE$ = A$
3330  RETURN
3340  REM
3350  REM  SAISIE DU CODE POSTAL
3360  REM  **********************
3370  REM
3380  INPUT "CODE POSTAL ";A$
3390 N = 5: IF  LEN (A$) <  = N THEN 3420
3400  PRINT "PAS PLUS DE CINQ CHIFFRES SVP"
3410  GOTO 3380
3420  GOSUB 3840:CDPST$ = A$
3430  RETURN
3440  REM
3450  REM  SAISIE DE L'INDICATIF TELEPHONIQUE
3460  REM  ********************************
```

```
3470  REM
3480  INPUT "INDICATIF TELEPHONIQUE ";A$
3490 N = 2: IF  LEN (A$) <  = N THEN 3520
3500  PRINT "PAS PLUS DE DEUX CHIFFRES SVP"
3510  GOTO 3480
3520 N = 2: GOSUB 3840:INDTEL$ = A$
3530  REM
3540  REM  SAISIE DU NUMERO DE TELEPHONE
3550  REM  ****************************
3560  REM
3570  INPUT "NUMERO DE TELEPHONE ";A$
3580 N = 7: IF  LEN (A$) <  = N THEN 3610
3590  PRINT "PAS PLUS DE SEPT CHIFFRES SVP"
3600  GOTO 3570
3610  GOSUB 3840:TEL$ = A$
3620  RETURN
3630  REM
3640  REM
3650  REM  AFFICHAGE MESSAGE D'ERREUR
3660  REM  CHAINE TROP LONGUE
3670  REM  ************************
3680  REM
3690  PRINT : INVERSE
3700  PRINT "PAS PLUS DE ";N;" CARACTERES, SVP"
3710  NORMAL
3720  RETURN
3730  REM
3740  REM  AFFICHAGE DES DONNEES SAISIES
3750  REM  ****************************
3760  REM
3770  PRINT
3780  PRINT "NOM"; TAB( 20);NOM$
3790  PRINT "PRENOM"; TAB( 20);PRENOM$
3800  PRINT "ADRESSE"; TAB( 20);NUMERO$;" ";GENREVOIE$;" ";NVOIE$
3810  PRINT "VILLE"; TAB( 20);CDPST$;" ";VILLE$
3820  PRINT "TELEPHONE"; TAB( 20);"(";INDTEL$;")";TEL$
3830  RETURN
3840  REM
3850  REM  CADRAGE DES DONNEES SAISIES
3860  REM  ****************************
3870  REM
3880  REM  DANS UNE CHAINE DE N CARACTERES BLANCS
3890  REM  ********************************
3900 N = N -  LEN (A$)
3910  IF N <  = 0 THEN  RETURN
3920 A$ = A$ +  LEFT$ (C$,N)
3930  RETURN
```


1.7 LES COMMANDES CATALOGUÉES SUR L'APPLE

1.7.1 Présentation

Vous pouvez créer des commandes cataloguées sur votre APPLE. Pour cela, vous devez créer un fichier séquentiel (cf. § 1.6.3) contenant les commandes à exécuter.

Supposons par exemple que le fichier FCOM contienne les enregistrements suivants :

```
 5 HOME
10 PRINT "COUCOU"
20 END
   LIST
   RUN
   HOME
   CATALOG
```

Comment lancer maintenant la commande cataloguée FCOM ? Pour cela il suffit de taper la commande suivante :

```
EXEC FCOM
```

Que se passe-t-il ?

Les lignes Basic 5, 10, 20 sont chargées en mémoire. Le programme est alors listé puis exécuté. L'écran s'efface ensuite et le catalogue de la disquette est affiché.

1.7.2 Exemple : changement du type d'un fichier Basic

Lorsque dans le fichier séquentiel de la commande cataloguée se trouvent des lignes Basic numérotées, la commande EXEC les charge en mémoire. C'est un moyen de restaurer un programme Basic à partir d'un fichier de type texte.

Considérons le programme suivant :

```
10 D$=CHR$ (4)
20 PRINT D$ ; "OPEN LISTING"
30 PRINT D$ ; "WRITE LISTING"
35 POKE 33, 33 : REM LIGNE DE 33 CARACTÈRES
```

```
40 LIST
50 PRINT D$ ; "CLOSE"
60 END
```

Ce programme crée le fichier LISTING qui contient le listing du programme dont on veut changer le type. Changeons de Basic et tapons la commande EXEC LISTING. Le fichier est alors rechargé en mémoire. Il ne reste plus qu'à sauvegarder le programme en mémoire dans un fichier de l'autre type.

Cet exemple peut servir à ajouter des sous-programmes standards à vos programmes, à modifier les numéros de lignes d'une partie d'un programme ou pour récupérer une partie de programme qui a eu des problèmes (disquettte abîmée...).

1.7.3 Exemple : transformation d'un programme en langage machine en programme Basic

Pour cela, il suffit de lire le contenu d'une adresse et d'écrire des lignes du type POKE a, b dans un fichier séquentiel. Ces lignes seront ensuite chargées en mémoire par la commande EXEC.

1.7.4 Format de la commande EXEC

Le format habituel de la commande EXEC est le suivant :

EXEC nom fichier

Le fichier est un fichier séquentiel contenant des commandes Basic ou des lignes de programmes.

Les enregistrements sont lus et sont soit exécutés s'il s'agit de commandes Basic, soit chargés en mémoire s'il s'agit de lignes de programmes. La commande cataloguée se termine lorsqu'il n'y a plus d'enregistrements à traiter.

Lorsqu'un enregistrement ne peut être interprété, le DOS affiche alors le message SYNTAX ERROR et enchaîne sur le traitement de l'enregistrement suivant.

Si vous voulez arrêter l'exécution de la commande cataloguée active, vous devrez appuyer sur la touche RESET car CTRL-C est sans effet.

Il vous est possible de mettre la commande RUN *dans une commande cataloguée. Toutefois vous devrez être prudent car toute instruction* INPUT *(ou* GET*) se référera au fichier des commandes cataloguées (comme si la commande* READ *était active). Lorsque l'instruction* INPUT *est rencontrée, la prochaine commande est lue et même exécutée. Le résultat s'avérera rapidement très compliqué.*

Si vous voulez arrêter l'exécution d'un programme Basic, vous pourrez le faire avec CTRL-C*, mais l'exécution de la commande cataloguée le sera aussi.*

Deux commandes cataloguées ne peuvent être actives à la fois. Si durant l'exécution d'une commande cataloguée, le DOS *trouve la commande* EXEC *fichier* 2*, la première commande est arrêtée et la seconde est activée.*

L'exécution d'une commande cataloguée peut commencer à la p^{ème} commande. Pour cela, tapez :

EXEC fichier, Rp

Les commandes sont numérotées à partir de zéro. Utiliser R0 est inutile car 0ème commande est située au début du fichier. Si vous utilisez R1, vous sauterez la première commande (n = 0), et l'exécution de la commande cataloguée commencera à la seconde commande (n = 1).

La position p *indiquée est toujours absolue (c'est-à-dire par rapport au début du fichier).*

1.7.5 Exemple

Nous reprenons ci-dessous l'exemple de tracé de courbes Y = f(X) en coordonnées cartésiennes présenté au tome 1, paragraphe 3.3. La modification réside dans le fait que la ligne 1280 est placée automatiquement en mémoire et que l'équation tapée peut être réinvoquée par la commande EXEC EQUATION lorsque le programme est en mémoire.

La structure obtenue pour le programme est la suivante :

Lignes 10-240	création de la ligne 1 280 et de la commande cataloguée EQUATION qui est exécutée immédiatement
Lignes 300-390	: initialisations des paramètres de la courbe
Lignes 400-610	: calcul des échelles et des axes

Lignes 620-680 : affichage de l'équation
Lignes 690-740 : tracé des axes
Lignes 750-810 : affichage du point de départ
Lignes 820-890 : tracé de la courbe
Lignes 900-1 010 : tracé de l'échelle de l'axe des X
Lignes 1 020-1 120 : tracé de l'échelle de l'axe des Y
Lignes 1 120-1 300 : sous-programme de détermination des échelles
Lignes 1 310-1 500 : traitement des erreurs.

```
10  REM  PROGRAMME DE TRACE DE COURBES
20  REM
30  REM  COPYRIGHT DE MERLY
40  REM
50  TEXT : HOME
60  PRINT "CE PROGRAMME TRACE TOUTE COURBE"
70  PRINT "ACCESSIBLE AVEC LES FONCTIONS"
80  PRINT "MATHEMATIQUES DE L'APPLESOFT."
90  PRINT "POUR CELA, TAPEZ VOTRE EXPRESSION"
100  PRINT "SELON LE FORMAT:"
110  PRINT " Y=F(X)"
120  PRINT
130  PRINT " EXEMPLE: Y=EXP(X)"
140  INPUT "";E$:F$ = "1280 " + E$:D$ =  CHR$ (4)
150  PRINT D$;"OPEN EQUATION"
160  PRINT D$;"DELETE EQUATION"
170  PRINT D$;"OPEN EQUATION"
180  PRINT D$;"WRITE EQUATION"
190  PRINT F$
200  PRINT "RUN 270"
210  PRINT E$
220  PRINT D$;"CLOSE"
230  PRINT D$;"EXEC EQUATION"
240  END
250  REM
260  REM  DEBUT DU PROGRAMME REEL
270  REM
280  INPUT E$
290  DEF  FN I(X) =  INT (X + (1 -  SGN (X)) / 2.00000001#)
300  REM
310  REM  PARAMETRES DE LA COURBE
320  REM
330  HOME
340  PRINT E$
350  PRINT : PRINT ""
360  INPUT "X INITIAL -->";XD
370  PRINT
380  INPUT "X FINAL   -->";XF
390  ONERR  GOTO 1340
400  REM
```



```
410  REM  CALCUL DES ECHELLES ET DES AXES
420  REM
430  IF XF < XD THEN T = XF:XF = XD:XD = T
440  IF XD <  = 0 AND XF > 0 THEN XL = XF - XD:XC =  - XD
450  IF XD < 0 AND XF <  = 0 THEN XL =  - XD:XC =  - XD
460  IF XD > 0 AND XF > 0 THEN XL = XF:XC = 0
470 XI = XL / 279
480 XZ = XC / XI
490  PRINT : PRINT "TRAITEMENT EN COURS"
500 YD = 0:YF = 0
510  FOR X = XD TO XF STEP XI
520  GOSUB 1280
530  IF Y > YF THEN YF = Y
540  IF Y < YD THEN YD = Y
550  NEXT X
560  IF YD <  = 0 AND YF > 0 THEN YL = YF - YD:YC =  - YD
570  IF YD < 0 AND YF <  = 0 THEN YL =  - YD:YC =  - YD
580  IF YD > 0 AND YF > 0 THEN YL = YF:YC = 0
590 YI = YL / 159
600 YZ = YC / YI
610 YZ = 159 -  INT (YZ)
620  REM
630  REM  AFFICHAGE DE L'EQUATION
640  REM
650  HGR
660  VTAB 22
670  IF  LEN (E$) < 40 THEN  HTAB (40 -  LEN (E$)) / 2
680  PRINT E$
690  REM
700  REM  TRACE DES AXES
710  REM
720  HCOLOR= 3
730  HPLOT 0,YZ TO 279,YZ
740  HPLOT XZ,0 TO XZ,159
750  REM
760  REM  POINT DE DEPART
770  REM
780 X = XD
790  GOSUB 1280
800 Y = 159 - (Y + YC) / YI
810  HPLOT (X + XC) / XI,Y
820  REM
830  REM  TRACE DE LA COURBE
840  REM
850  FOR X = XD + XI TO XF STEP XI
860  GOSUB 1280
870 Y = 159 - (YC + Y) / YI
880  HPLOT  TO (X + XC) / XI,Y
890  NEXT X
900  REM
910  REM  TRACE DE L'ECHELLE DE L'AXE DES X
920  REM
930 R = XL
940  GOSUB 1150
950  HCOLOR= 0
```

```
960  IF XD < 0 AND XF < 0 THENN XF = 0
970  IF XD > 0 AND XF > 0 THENN XD = 0
980  FOR X =  FN I(XD / DI) * DI TO  FN I(XF / DI) * DI STEP DI
990  HPLOT XZ + X / XI,YZ
1000  NEXT X
1010  VTAB 24: PRINT DI;"/ DIWISION SUR X";
1020  REM
1030  REM  TRACE DE L'ECHELLE  DE L'AXE DES Y
1040  REM
1050 R = YL
1060  GOSUB 1150
1070  FOR Y =  FN I(YD / DI) ** DI TO  FN I(YF / DI) * DI STEP DI
1080  HPLOT XZ,YZ - Y / YI
1090  NEXT Y
1100  PRINT  SPC( 2);DI;"/DIVIISION SUR Y";
1110  GOTO 1490
1120  REM
1130  REM  DETERMINATION DES ECHELLES
1140  REM
1150 K = 0
1160  IF R < 1 THEN 1220
1170  IF R < 10 THEN 1250
1180 K = K + 1
1190 R = R / 10
1200  GOTO 1170
1210  IF R > 1 THEN 1250
1220 K = K - 1
1230 R = R * 10
1240  GOTO 1210
1250 DI =  INT (R) * 10 ^ (K — 1)
1260  RETURN
1270  REM ************************
1280 Y = 4
1290  REM ************************
1300  RETURN
1310  REM
1320  REM  TRAITEMENT DES ERREURS
1330  REM
1340  POKE 216,0: REM  ANNULATTION ONERR
1350 EA =  PEEK (218) +  PEEK  (219) * 256
1360 E2$ = "ERREUR"
1370 N$ = "":E1$ = ""
1380  IF EA <  > 1010 THEN N$  = "N'EST PAS ":E1$ = " A LA LIGNE " +  STR$ (EA)
1390  HOME : VTAB 21
1400 ER =  PEEK (222)
1410  IF E$ = "" THEN  PRINT "ERREUR PAS D'EQUATION.": END
1420  PRINT "ERREUR DE SYNTAXE ";N$;"DANS L'EQUATION."
1430  PRINT E$
1440  IF EA <  > 1010 THEN 14660
1450  IF ER = 16 OR ER = 163 OR ER = 224 THEN E2$ = "EQUATION FAUSSE"
1460  IF ER = 53 OR ER = 69 OR ER = 133 THEN E2$ = "DONEES FAUSSES"
1470  IF ER = 255 THEN E2$ = "CONTROL-C"
1480  PRINT E2$ + E1$
1490  VTAB (15): PRINT  CHR$ (7): GET N$: TEXT
1500  END
```


1.8 LES UTILITAIRES DU DOS 3.3

1.8.1 Copie d'une disquette

Il est préférable de garder toujours une copie de vos disquettes car il peut toujours se produire un accident.

Un programme vous permet de recopier entièrement une disquette : les programmes COPY (Basic Entier) ou COPYA (APPLESOFT), qui se situent sur la disquette SYSTEM MASTER.

Lorsque vous lancez la copie d'une disquette, le programme vous demande alors la situation des disquettes (connecteur, unité de disquette) selon le format suivant :

```
APPLE DISKETTE DUPLICATION PROGRAM
ORIGINAL SLOT:DEFAULT=6
DRIVE:DEFAULT=1
DUPLICATE SLOT:DEFAULT=6
DRIVE:DEFAULT=2
-- PRESS "RETURN" KEY TO BEGIN COPY
```

A chacune des questions concernant les connecteurs et les unités de disquettes, vous devez taper : Return si l'option par défaut vous convient, ou le numéro voulu puis Return.

Lorsque le programme vous demande d'appuyer sur Return pour commencer la copie, mettez la disquette à copier dans la première unité de disquettes précisée, et une disquette vierge ou à écraser dans la seconde unité.

La copie commence. La disquette vierge est initialisée puis tous les fichiers sont recopiés. Lorsque la copie est terminée, le programme vous demande si vous voulez faire une autre copie.

Dans le cas d'un système à une seule unité de disquettes, le programme vous demandera de mettre successivement la disquette à copier et la disquette sur laquelle on fait la copie.

Pour éviter une fausse manipulation, nous vous recommandons de protéger en écriture la disquette à copier. (cf. § 1.2).

1.8.2 Recopie de fichiers. Le programme FID

● **Présentation**

Le programme FID vous permet de copier des fichiers entre plusieurs disquettes, d'obtenir le catalogue, de détruire un fichier, de verrouiller un fichier et de connaître la place disponible sur une disquette.

Pour utiliser ce programme, la disquette SYSTEM MASTER doit être dans le drive actif. Vous devez alors taper la commande BRUN FID.

S'affiche alors sur l'écran une liste de commandes précédées d'un numéro entre chevrons < >. Pour exécuter une des commandes, tapez le numéro de la commande puis Return.

● **Désignation des fichiers**

Certaines des commandes de FID vous demanderont un nom de fichier, qui aura le même format que pour les commandes du DOS, mais vous pouvez remplacer une partie du nom par le symbole =. Tous les fichiers dont le nom peut tenir dans le format indiqué seront alors considérés.

Si vous tapez par exemple :

COLOR=

tous les fichiers dont le nom commence par COLOR seront alors pris en compte.

Si le symbole = est utilisé, la question suivante est affichée :

DO YOU WANT PROMPTING ?

Si vous répondez Y (Yes, oui), pour chaque fichier correspondant au nom donné, le programme vous demandera si vous voulez exécuter ou non la commande. Vous devrez répondre par Y (oui) ou N (non), ou Q (quitter). Si vous tapez Y, le fichier sera traité par la commande. Si vous tapez Non, le fichier ne sera pas traité. Si vous tapez Q, la commande est abandonnée.

Si vous répondez N (non) à la question DO YOU WANT PROMPTING ? tous les fichiers dont le nom correspond seront traités par la commande active.

● **Copie d'un fichier**

Si vous avez répondu 1 au menu principal de FID, la commande de copie de fichier est sélectionnée. Le programme vous demande alors les

caractéristiques des unités de disquettes (même principe que 1.8.1), puis le nom du fichier (cf. *Désignation des fichiers*). La commande vous demande alors de mettre les disquettes en place et de taper soit ESC si vous voulez arrêter la copie, soit toute touche excepté RESET, CTRL, SHIFT, ESC, pour continuer.

Si les fichiers dont vous avez demandé la copie existent et qu'il y a assez de place sur la nouvelle disquette, vous verrez s'afficher sur l'écran le message DONE. Par contre, si aucun fichier ne correspond sur la disquette origine à ce que vous avez demandé, le message suivant s'affiche : NO FILES SELECTED.

Si la disquette destination contient un fichier de même nom que le fichier à copier, le message suivant est affiché :

FILE ALREADY EXISTS
TYPE IN NEW FILE NAME FOR THE COPY OR
<RETURN> TO REPLACE PRESENT FILE OR
<CTRL-C> <RETURN> TO CANCEL COPY

Trois solutions s'offrent à vous :

— donner un nouveau nom de fichier,
— taper Return pour remplacer l'ancien fichier par le nouveau,
— taper CTRL-C Return pour ne pas recopier le fichier.

Si dans la seconde solution l'ancien fichier est verrouillé, le message suivant s'affiche sur l'écran :

FILE LOCKED
DO YOU WISH TO REPLACE IT ANYWAY ?

Si vous tapez N (non), le message ci-dessus est réaffiché. Si vous tapez Y (oui), l'ancien fichier est tout de même remplacé.

Si votre APPLE II a une seule disquette, il est possible de copier un fichier entre deux disquettes. Vous devrez seulement suivre les ordres affichés pour mettre les disquettes en place.

● **Catalogue d'une disquette**

Si vous avez répondu 2 au menu général, le catalogue de la disquette courante est affiché sur l'écran. Si la disquette courante n'a pas été définie, la commande vous demandera le contrôleur et le numéro du drive avant d'afficher le catalogue.

- **Place disponible sur une disquette**

Si vous avez répondu 3 au menu général, le nombre de secteurs utilisés et de secteurs disponibles est affiché sur l'écran.

- **Déverrouillage d'un fichier**

Pour déverrouiller un fichier, vous devez répondre au menu général de FID. La commande vous demande le nom du fichier, vous indique de mettre la disquette en place et vous pose la question : voulez-vous vraiment déverrouiller le fichier ?

Si le fichier n'est pas sur la disquette, le message :

NO FILE SELECTED

est affiché.

- **Verrouillage d'un fichier**

Le principe est le même que précédemment. La seule différence consiste dans le fait que vous devez répondre 5 (au lieu de 4) au menu principal de FID.

- **Destruction d'un fichier**

Cette commande permet de détruire un ou plusieurs fichiers. Pour l'utiliser, il suffit de répondre 6 au menu principal de FID et de donner le nom du fichier. Si tout se passe bien, le message < DONE > est affiché. Si aucun fichier n'est trouvé, le message NO FILES SELECTED. Si un des fichiers est verrouillé, le message FILE LOCKED est affiché et le fichier n'est pas détruit.

Lorsque le message PRESS ANY KEY TO CONTINUE est affiché, vous devrez taper sur une touche excepté Reset, CTRL, SHIFT, RETURN.

- **Reset des caractéristiques de la disquette active**

Si vous tapez 7 lors du menu général de FID, les numéros de connecteur et de drive de l'unité de disquette active vous seront redemandés lors de la prochaine exécution d'une commande qui les nécessite.

- **Vérification physique d'un fichier**

La commande 8 de FID est identique à la commande VERIFY du DOS. Elle vous demande un nom de fichier et vérifie les secteurs physiques occupés par les fichiers correspondant au nom fourni.

Lorsqu'un fichier a été vérifié, la commande affiche son nom et le message DONE. Si par contre le fichier ne peut être vérifié, il en résulte l'erreur I/O ERROR. Le programme attend alors que vous tapiez sur une touche pour retourner au menu principal de FID.

1.8.3 Création d'une disquette système maître

Les disquettes créées par la commande d'initialisation INIT du DOS dépendent de la taille mémoire du système dans lequel on a créé la disquette. Le programme MASTER CREATE présent sur la disquette SYSTEM MASTER vous permet de créer des disquettes maîtres adaptées à la taille mémoire du système utilisé.

Pour utiliser ce programme, tapez la commande :

```
BRUN MASTER CREATE
```

Le message suivant vient à l'écran :

```
DOS 3.3 MASTER-CREATE UTILITY
COPYRIGHT 1980 BY APPLE COMPUTER INC
ALL RIGHTS RESERVED
(NOW LOADING DOS IMAGE)
```

puis le message

```
PLEASE INPUT THE GREETING PROGRAM'S
FILE NAME
```

Tapez ici le nom du programme qui sera exécuté au chargement du DOS. A la différence de la commande INIT, ce programme doit exister sur la disquette.

Vous voyez ensuite s'afficher le message :

```
REMEMBER THAT MASTER DOES NOT CREATE
THE GREETING PROGRAM, OR PLACE IT IN
THE DISK DIRECTORY
THIS IS THE FILE NAME THAT WILL BE
PLACED WHITHIN THE IMAGE:
HELLO
PLACE THE DISKETTE TO BE MASTERED IN
THE DISK DRIVE
```

PRESS <RETURN> WHEN READY
NOTE:IF YOU WANT A DIFFERENT FILE NAME,
PRESS <ESC>

Tapez Return si vous êtes prêt.

Lorsque le programme vous rend la main, rechargez toujours le DOS avant de faire tout autre travail. Votre disquette est devenue une disquette maître.

1.8.4 Passages de fichiers DOS ancien en DOS 3.3 : MUFFIN

Le programme MUFFIN, fourni sur la disquette SYSTEM MASTER, permet de transférer vos programmes sur une disquette DOS 3.3. Pour le lancer, envoyez :

BRUN MUFFIN

Vous voyez apparaître sur l'écran le message

```
****************************************
*  APPLE ][ DOS 3.2 TO 3.3 CONVERTER   *
*                                      *
*           MUFFIN VERSION D           *
*                                      *
*  COPYRIGHT 1979 APPLE COMPUTER INC.  *
****************************************
CHOOSE ONE OF THE FOLLOWING OPTIONS
       <1>   CONVERT FILES
       <2>   QUIT
WHICH WOULD YOU LIKE?
```

Pour lancer la conversion, tapez 1. Les questions suivantes s'affichent sur l'écran :

SOURCE SLOT ?	connecteur de départ
DRIVE ?	lecteur (1, 2) de départ
DESTINATION SLOT ?	connecteur d'arrivée
DRIVE ?	lecteur d'arrivée
FILENAME ?	nom du fichier à transférer

Il est possible de transférer plusieurs fichiers à la fois, en utilisant le caractère = qui permet de remplacer un ou plusieurs caractères.

Par exemple, si vous avez les fichiers FICH 11, FICH 12, FICH 21, le nom FICH=1 permettra de désigner à la fois FICH 11 et FICH 21.

DO YOU WANT PROMPTING ?

Le DOS vous demande si vous voulez vérifier le transfert (ou non). Si vous tapez Y (oui), à chaque fichier possible le DOS vous demandera si le transfert est réellement à faire. Si vous tapez N (non), tous les fichiers possibles seront transférés.

Le DOS affiche ensuite :

INSERT DISKS THEN PRESS <ESC> TO RETURN
TO MAIN MENU OR ANOTHER KEY TO BEGIN

Insérez vos disquettes et tapez Return. Si vous désirez arrêter, tapez ESC et vous reviendrez au premier menu.

Vous pouvez utiliser MUFFIN *avec une seule unité de disquettes, en mettant les mêmes valeurs pour les plots et disques de départ et d'arrivée. Le* DOS *vous indique alors les disquettes à insérer (source et arrivée).*

1.9 STRUCTURE INTERNE DU DOS ; STRUCTURE DES INFORMATIONS SUR DISQUETTES ET SOUS-PROGRAMMES UTILISABLES

1.9.1 Introduction

Nous allons décrire comment le DOS stocke les fichiers sur disquette, gère le catalogue, etc.

Sur une disquette, les informations sont enregistrées en 35 pistes numérotées de $00 (extérieur) à $22 (intérieur). Ces pistes sont découpées en 16 secteurs de 256 octets. Pour accéder à un secteur donné d'une piste donnée, le DOS met en marche le moteur faisant tourner la disquette, et envoie le signal nécessaire pour positionner la tête de lecture en face de la piste désirée. Les divers secteurs passent sous la tête jusqu'au secteur voulu.

Pour écrire sur un secteur, le DOS rassemble les informations dans un buffer de 256 octets, accède au secteur sur la disquette et y stocke les données.

En général, le DOS commence à stocker un programme ou un fichier en début d'un secteur libre. Comment fait-il pour gérer ces secteurs libres ?

Il existe pour cela, par piste, une table indiquant l'utilisation ou non des secteurs de la piste (Track bit map). Nous la décrirons dans la suite de ce paragraphe.

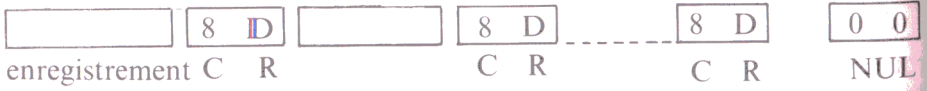
Lorsqu'un secteur est plein et qu'il reste des données à stocker, le DOS garde les renseignements du secteur (nº de piste 0 à 34, nº de secteur dans la piste 0-15) dans une table appelée « Track Sector List » (liste des secteurs des pistes). Il utilise un autre secteur libre pour continuer à stocker le fichier. Lorsque celui-ci est entièrement stocké, le DOS sauvegarde la table des secteurs du fichier dans un nouveau secteur dont il conserve la position au catalogue de la disquette.

Le catalogue de la disquette est stocké dans un certain nombre de secteurs « chaînés » entre eux. L'adresse (piste, secteur) du catalogue est stockée dans la table du contenu de la disquette placée dans le secteur 0 de la piste numéro 17 ($11). Cette table (VTOC) est décrite en détail dans la suite de ce paragraphe.

1.9.2 Format des fichiers

* BASIC (Entier ou APPLESOFT)

* TEXTE (séquentiel ou à accès direct)



Chaque enregistrement contient les codes ASCII des textes.

* BINAIRE

A B poids faible de l'adresse de chargement
A H poids fort de l'adresse de chargement
L B poids faible de la longueur
L H poids fort de la longueur

1.9.3 Table des secteurs d'un fichier (Track/Sector List)

Elle contient les adresses (nº de piste, nº de secteur) des secteurs d'un fichier. Son format est le suivant :

Octet		Utilisation
	0	vide
	1	piste } du prochain secteur de stockage
	2	secteur (0-F) } de la T/SL
	3, 4	vide
	5, 6	nombre de secteurs utilisés pour stocker cette table T/SL
	7 à B	vide
122 emplacements	C, D	nº de piste, nº de secteur (0-F)
	E, F	nº de piste, nº de secteur (0-F)
	10, 11	nº de piste, nº de secteur (0-F)
	⋮	
	FE, FF	

Tous les secteurs de la T/S sont identiques. L'adresse du premier est contenue dans le catalogue.

Pour les fichiers de type texte, à accès direct, qui peuvent contenir de nombreux trous, seule la place effectivement nécessaire est occupée. Il n'y a pas de réservation. Si par exemple, la longueur d'un enregistrement est de 64 *octets* (4 *enregistrements par secteur*) *et que vous voulez écrire l'enregistrement numéro* 20000, *les secteurs suivants seront réservés :*

— 1 *secteur pour contenir l'enregistrement* 2000,0,
— 5 *secteurs pour la* T/S, *c'est-à-dire*

$$\text{INT}\left(\frac{20000}{122 \times 4} + 1\right)$$

soit au total 6 *secteurs.*

Les emplacements correspondant aux enregistrements non encore existants seront mis à 0/0 (piste/secteur) et mis à jour si le secteur correspondant est à créer.

Par contre, le DOS ne libérera pas les secteurs devenant inoccupés si la taille d'un fichier diminue.

1.9.4 Format du catalogue d'une disquette

Le DOS conserve les informations relatives aux différents fichiers dans un ensemble de secteurs appelés catalogue. Vous obtenez leur contenu en utilisant la commande CATALOG. Sur toute disquette initialisée, la piste numéro 17 ($11) est réservée à cet effet. Le catalogue commence au secteur $F de la piste $11. Les secteurs suivants ($E à $1 de la piste $11) lui sont progressivement alloués au fur et à mesure des besoins. Le catalogue peut donc contenir au maximum 105 fichiers.

Chaque secteur du catalogue d'une disquette a le format suivant :

Octet	Fonction
0	vide
1	piste } de la suite du catalogue
2	secteur } (0/0 si fin)
3 à A	vide
B à 2D	entrée du catalogue pour le fichier 1
2E à 50	entrée du catalogue pour le fichier 2
51 à 73	entrée du catalogue pour le fichier 3
74 à 96	entrée du catalogue pour le fichier 4
97 à B9	entrée du catalogue pour le fichier 5
BA à DC	entrée du catalogue pour le fichier 6
DD à FF	entrée du catalogue pour le fichier 7

L'entrée d'un fichier est rendue libre quand celui-ci est détruit en mettant le premier octet de l'entrée à $FF.

Chaque entrée de fichier dans le catalogue a le format suivant :

Octet	Fonction
0	piste au début de la table des secteurs du fichier (T/S List)
1	secteur
2	type du fichier
	Bit 7 * fichier verrouillé si bit à 1
	6
	5 extensions futures
	4
	3
	2 B fichier binaire si ce bit est à 1
	1 A fichier APPLESOFT si ce bit est à 1
	0 I fichier Basic Entier si ce bit est à 1
	type texte si les bits 0 à 6 sont à zéro
3 à 20	nom du fichier
21	nombre de secteurs du fichier modulo 256

Si ce nombre est supérieur à 256, *la gestion du fichier par le* DOS *ne pose pas de problèmes, sauf pour la place libre indiquée à l'utilisateur.*

1.9.5 Table du contenu d'une disquette

Elle est située dans le secteur 0 de la piste 17 ($11). Elle contient les informations suivantes :

Octet nº hexa	Valeur nº hexa	Description
0	02	vide
1	11	piste du premier secteur du catalogue
2	0F	secteur
3	04	version du DOS
4	00	vide

Octet nº hexa	Valeur nº hexa	Description
5	00	vide
6	1 à $FE	volume de la disquette
7 à 26	00	vide
30	$FF	masques pour les tables de secteurs libres (bit à 1 = secteur libre)
31	$FF	
32	$00	
33	$00	
34	$23	nombre de pistes
35	$0F	nombre de secteurs
36	00	nombre d'octets par secteurs
37	01	
38 à 3B	00	table des secteurs libres des pistes 0 à 2 (DOS)
3C à 3F	00	
40 à 43	00	
44, 45	?	table des secteurs libres piste $3
46, 47	0	
48, 49	?	table des secteurs libres piste $4
4A, 4B	0	
4C, 4D	?	table des secteurs libres piste $5
4E, EF	0	
...		
C0, C1	?	table des secteurs libres piste $22
C2, C3	0	
C4 à FF	0	vide

Les tables de secteurs libres des pistes sont formées de 4 octets. Les deux derniers sont à zéro pour le DOS 3.3.

Chaque bit des deux premiers octets indique si le secteur correspondant est libre ou non.

— Bit à 1 secteur libre.
— Bit à 0 secteur occupé.

Un exemple de table est le suivant :

bit	0 0 0 0 1 1 1 1	1 1 1 1 1 1 1 1	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0
secteur	F E D C B A 9 8	7 6 5 4 3 2 1 0		

Les pistes sont utilisées par le DOS dans l'ordre suivant :

18 à 34 ($12 à $22) vers l'intérieur

puis

16 à 3 ($10 à $03) vers l'extérieur

Les secteurs d'une piste sont attribués dans l'ordre $OF→$00.

1.9.6 Adresses mémoire du DOS sur un Apple II 48 K

La table suivante donne les adresses où sont chargées les pistes 0 à 2 correspondant au DOS sur la disquette. Lors du chargement d'une disquette-maître, le DOS est chargé à une adresse qui dépend de la taille mémoire. Pour une disquette esclave, le DOS est chargé à l'adresse où il était lorsque l'on avait initialisé la disquette.

Disquette (maître ou esclave)		Mémoire	
piste	secteur	adresse chargement	adresse après déplacement
00	00	$3600	$B600
00	01	$3700	$B700
00	02	$3800	$B800
00	03	$3900	$B900
00	04	$3A00	$BA00
00	05	$3B00	$BB00
00	06	$3C00	$BC00

Disquette (maître ou esclave)		Mémoire	
piste	secteur	adresse chargement	adresse après déplacement
00	07	$3D00	$BD00
00	08	$3E00	$BE00
00	09	$3F00	$BF00
00	0A	$1B00	
00	0B	$1C00	
00	0C	$1D00	$9D00
00	0D	$1E00	$9E00
00	0E	$1F00	$9F00
00	0F	$2000	$A000
01	00	$2100	$A100
01	01	$2200	$A200
01	02	$2300	$A300
01	03	$2400	$A400
01	04	$2500	$A500
01	05	$2600	$A600
01	06	$2700	$A700
01	07	$2800	$A800
01	08	$2900	$A900
01	09	$2A00	$AA00
01	0A	$2B00	$AB00
01	0B	$2C00	$AC00
01	0C	$2D00	$AD00
01	0D	$2E00	$AE00
01	0E	$2F00	$AF00
01	0F	$3000	$B000
02	00	$3100	$B100
02	01	$3200	$B200
02	02	$3300	$B300
02	03	$3400	$B400
02	04	$3500	$B500

Sur un APPLE II 48 K, le DOS occupe les adresses $9600 à $BFFF (fin de la RAM), soit 38400 à 49151, c'est-à-dire environ 10 K octets.

Il occupe également certains octets en page trois de la mémoire.

Vecteurs du DOS			
3D0	JMP	$9DBF	rechargement « à chaud » du DOS
3D3	JMP	$9D84	rechargement complet du DOS
3D6	JMP	$AAFD	branchement au gestionnaire de fichiers
3D9	JMP	$B7B5	branchement à RWTS
3DC	LDA	$9D0F	restauration des paramètres pour le gestionnaire de fichiers
	LDY	$9D0E	(adresse de la table FMPL)
	RTS		
3E3	LDA	$AAC2	restauration des paramètres pour RWTS
	LDY	$AAC1	(adresse de la table IOB)
	RTS		
3EA	JMP	$A851	remise à jour du DOS pour les buffers d'entrée-sortie
	NOP		
	NOP		

Vecteurs du moniteur et de l'APPLESOFT.

Ce paragraphe décrit le sous-programme RWTS (accès direct aux disquettes sans passer par le DOS) et le gestionnaire de fichiers.

Nous avons indiqué à l'adresse $3EA un branchement vers un sous-programme de remise à jour du DOS pour les buffers d'entrée-sortie. Lorsque le DOS est actif, qu'il n'est pas en cours d'accès à un fichier (pas de commande READ ou WRITE active) et que vous tapez PRINT "COUCOU", que se passe-t-il ?

Les registres du moniteur CSWL,H et KSWL,H (cf. tome 1) contiennent l'adresse du DOS, ce qui permet au moniteur de lui transmettre la commande d'entrée-sortie. Le DOS ayant conservé les adresses des contrôleurs d'entrée-sortie dans des registres internes, lance l'entrée-sortie. A la fin de celle-ci, il rend le contrôle au moniteur en remettant son adresse dans CSWL,H et KSWL,H. C'est le rôle du sous-programme dont le vecteur se situe en $03EA.

1.9.7 Accès direct aux secteurs physiques de la disquette

Le sous-programme RWTS (Read Write Track Sector = lire écrire piste secteur) est situé pour un APPLE II 48 K à l'adresse $B7B5. Il est plus simple d'appeler toujours le vecteur RWTS en $3D9 qui est indépendant de la taille mémoire et qui contient un branchement à RWTS.

Les paramètres d'appel de RWTS sont deux tableaux appelés IOB (bloc d'entrée-sortie) et DCT (caractéristiques du drive). Le sous-programme $3E3 fournit dans les registres A (poids fort) et Y (poids faible) l'adresse du tableau IOB en mémoire.

Examinons plus précisément ces deux tableaux.

● **DCT**

Il a le format suivant :

Type	Phase	Moteur ON
$00	$01	$EFD8

Ce sont les caractéristiques du lecteur de disquettes. Elles seraient différentes avec un disque dur par exemple.

● **IOB**

Octet	Nom	Fonction
1	IBTYPE	
2	IBSLOT	numéro du plot du contrôleur *16
3	IBDRVN	numéro du lecteur de disquettes ($1 ou $2)
4	IBVOL	volume de la disquette à laquelle on va accéder
5	IBTRK	piste } auquel accéder
6	IBSEC	secteur } auquel accéder

Octet	Nom	Fonction
7 et 8	IBDCTP	adresse de la table DCT
9 et 10	IBBUFP	adresse du buffer de 256 octets où lire (ou écrire) les données à écrire (ou lire) par RWTS
11 et 12	vide	
13	IBCMD	Code de la commande pour RWTS $00 POSITIONNEMENT DE LA TETE $01 LECTURE $02 ECRITURE $04 FORMATAGE DE LA DISQUETTE
14	IBSTAT	Code d'erreur si le bit de retenue (cf. § 1.2) est à un $08 erreur de formatage $10 protection en écriture $20 mauvais volume $40 erreur de disque $50 erreur de lecture
15	IBSMOD	volume lu sur la disquette
16	IOBPSN	dernier plot
17	IOBPDN	dernier disque auquel on a accédé

1.9.8 Gestionnaire de fichiers

Outre le programme RWTS d'accès aux disquettes, le DOS est constitué d'un interpréteur de commandes et du gestionnaire de fichiers. L'interpréteur de commandes travaille sur les données saisies au clavier en mode direct ou communiquées par le Basic (CTRL-D) en mode programme. Pour toute commande de travail sur fichiers, il prépare une table de données appelée FMPL dont l'adresse peut être obtenue en appelant le sous-programme d'adresse $3DC (cf. § 1.9.6). Celui-ci fournit dans l'accumulateur l'octet de poids fort et dans le registre d'index Y l'octet de poids faible de cette adresse.

La table peut contenir jusqu'à 18 octets dont la signification est dépendante de la commande à réaliser, suivant le format de la page.

Les codes de retour possibles sont (+0A) :

Code	Signification
00	pas d'erreur
04	protection en écriture
05	fin de fichier
06	fichier inexistant
07	erreur de volume
08	erreur physique
0A	fichiers verrouillés

L'appel du gestionnaire de fichiers est réalisé par les instructions :

```
LDX  #0 ;     création de fichier autorisée
JSR  $3D6
BCS  erreur ; branchement sur erreur
```

1.9.9 Adresses utiles du DOS 3.3

● **Fichier binaire**

Après avoir tapé la commande BLOAD FICH1, il est utile de connaître l'adresse de chargement et la longueur du fichier.

Pour un APPLE II 48K, l'adresse de chargement est contenue en $AA72, AA73 et la longueur en $AA60.

● **Table des commandes du DOS**

Le DOS reconnaît les commandes grâce à une table située à l'adresse $AA88 qui contient les codes ASCII des noms des commandes. Chacune des 28 commandes est séparée de la suivante grâce au bit de poids fort de son dernier caractère qui est à 1. La longueur totale de la table est au maximum de 133 caractères et elle ne peut contenir d'espaces.

Ordre des commandes du DOS dans la table :

INIT, LOAD, SAVE, RUN, CHAIN, DELETE, LOCK, UNLOCK, CLOSE, READ, EXEC, WRITE, POSITION,

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11
OPEN	01		longueur d'un enregistrement ou 0000		V	D	S	type du fichier		adresse du buffer du fichier	code de compte-rendu (cf. ci-dessus)		adresse de la zone de travail du gestionnaire de fichiers (buffer de 45 octets)		adresse de la Track/Sector List			adresse du buffer des données
CLOSE	02								adresse relative des données dans le buffer									
READ	03		numéro de l'enregistrement		numéro d'octet			longueur des données										
WRITE	04																	
DELETE	05				V	D	S			adresse du buffer du fichier					adresse de la Track/Sector List			
CATALOG	06				V	D	S											
LOCK	07				V	D	S			adresse du buffer du DOS relatif au (nom du) fichier								
UNLOCK	08				V	D	S											
RENAME	09		adresse du buffer du nom nouveau		V	D	S			idem ci-dessus								
POSITION	0A		nº de l'enregistrement		nº d'octet													
INIT	0B				V	D	S											
VERIFY	0C				V	D	S											

Table FMPL des données du gestionnaire de fichier

V volume, D drive, S slot

ABTGF adresse du buffer de travail du gestionnaire de fichiers
ATSL adresse de la table des secteurs libres
ADATA adresse du buffer des données du secteur (256 octets)
ABS adresse nom de fichier suivant

Nom de fichier	ABTGF
ATSL	
ADATA	
ABS	

Format du buffer d'un fichier du DOS

OPEN, APPEND, RENAME, CATALOG, MON, NOMON, PR, IN, MAXFILES, FP, INT, BSAVE, BLOAD, BRUN, VERIFY

Vous pouvez ainsi personnaliser votre DOS et par exemple remplacer la commande CATALOG par CAT plus facile à taper rapidement. Une fois votre table fixée, vous devrez réenregistrer votre DOS sur disquette en utilisant par exemple le programme fourni au paragraphe 1.9.10.

● **Remplacement de l'affichage DISK VOLUME**

Si vous étudiez le contenu des adresses $B3B0 à $B3BA vous obtiendrez :

```
B3B0  C5  CD  D5  CC  CF  D6  A0  CB
B3B8  D3  C9  C4
```

Ces codes correspondent aux codes ASCII APPLE suivants :

```
B3B0  E  M  U  L  O  V  K
B3B8  S  [I]  D
```

Vous pouvez très bien remplacer ce message DISK VOLUME par un autre pour personnaliser vos disquettes. Mettez votre nom par exemple et utilisez le volume pour numéroter vos disquettes.

De plus, il est possible d'utiliser les divers modes vidéo de l'écran pour les caractères du message.

Si L=ASC(L$)

et L<64 le code écran du caractère est :

C+128 mode normal
C+ 64 mode clignotant
C mode inverse

et L>64 le code écran du caractère est :

C+128 mode normal
C mode clignotant
C− 64 mode inverse

La longueur de votre message est limitée à 11 *caractères et vous devez l'inverser en mémoire.*

Vous devrez pour conserver la modification restocker votre DOS *sur disquette* (cf. § 1.9.10).

● **Paramètres du système**

Si vous voulez connaître, à un instant donné, les caractéristiques du système, vous pourrez examiner les contenus des adresses suivantes :

$AA6A	−21910	plot actuel
$AA68	−21912	lecteur de disquette (1 ou 2) actif
$AA66	−21914	volume actuel
$AA6C	−21908	longueur d'enregistrement
$AA6E	−21906	numéro d'enregistrement
$AA70	−21904	numéro de l'octet actuel

1.9.10 Recopie d'un DOS modifié sur disquette

Si vous tapez le programme suivant :

```
300 : A909     LDA  $09
      8D5DAA   STA  $AA5D
      A900     LDA  $00
      8D5FAA   STA  $AA5F
      A99D     LDA  $9D
      8DBCB5   STA  $B5BC
      20C2B7   JSR  $B7C2
      204AB7   JSR  $B74A
      60       RTS
```

puis si vous frappez 300G, le DOS chargé en mémoire en RAM sera recopié sur les pistes 0, 1 et 2 de votre disquette.

1.10 TRAITEMENT DES ERREURS

Il est réalisé de la même façon qu'avec l'APPLESOFT. De nouveaux codes d'erreur sont utilisés :

Code	Message	Signification
1	LANGAGE NOT AVAILABLE	langage non disponible
2, 3	RANGE ERROR	le paramètre d'une commande est en dehors de l'intervalle autorisé
4	WRITE PROTECTED	disquette protégée en écriture (cf. § 1.2)
5	END OF DATA	fin de fichier (toutes les données ont été déjà lues)
6	FILE NOT FOUND	fichier non trouvé
7	VOLUME MISMATCH	erreur de volume
8	I/O ERROR	erreur physique (disquette non initialisée, porte non fermée, ...)
9	DISK FULL	disque plein
10	FILE LOCKED	fichier verrouillé (protégé en écriture)
11	SYNTAX ERROR	erreur de syntaxe
12	NO BUFFERS AVAILABLE	trop de fichiers ouverts de type TEXT (cf. § 1.6.2 MAXFILES)
13	FILE TYPE MISMATCH	mauvais type de fichier
14	PROGRAM TOO LARGE	taille d'un programme trop importante
15	NOT DIRECT COMMAND	commande non utilisable en mode direct

CHAPITRE **2**

Le système U.C.S.D.

2.1 PRÉSENTATION

2.1.1 Généralités

Le système UCSD permet d'utiliser sur l'APPLE II des langages évolués tels que Pascal, Fortran, ... et fournit avec ces langages un assembleur évolué.

Sur l'APPLE II e, il suffit d'avoir les disquettes correspondantes pour disposer du système UCSD, alors qu'avec l'APPLE II +, une carte d'extension, la carte langage, est nécessaire.

2.1.2 Structure mémoire de la carte langage

La carte langage vous offre 16 K octets supplémentaires de mémoire RAM, soit au total 64 K octets de RAM à la capacité mémoire adressable du microprocesseur 6502. La RAM offerte peut être en partie protégée en écriture ; vous y stockerez vos données les plus importantes.

La carte langage inclut de plus la ROM Autostart. Elle peut fonctionner avec les diffférentes versions du DOS 3.2, 3.3...

La carte mémoire de l'APPLE II + devient la suivante :

$FFFF $F800	CARTE LANGAGE		ROM Autostart carte langage
$F7FF $E000	8 K RAM supplémentaires		ROM APPLESOFT ou Basic
$DFFF $D000	1er bloc 4 K RAM	2nd bloc 4 K RAM	Entier
$CFFF $C000	Entrées/sorties cartes d'extension		
  $0000	RAM 48 K octets		

Pour avoir vraiment 64 K *octets de* RAM *accessibles à l'utilisateur, il faut remplacer la place mémoire réservée aux entrées/sorties par de la* RAM. *Ceci étant difficile,* APPLE *a préféré offrir dans la carte langage deux blocs de* 4 K RAM *sélectionnables séparément aux mêmes adresses.*

Tableau des possibilités de sélection, d'accès en lecture, en écriture et de protection en écriture de la mémoire

RAM $D000-FFFF		Sélection RAM/ROM et protection en écriture RAM
1er bloc 4 K inclus	2e bloc 4 K inclus	
$C080 – 16256	$C088 – 16248	Sélection en lecture RAM Protection en écriture RAM
$C081 – 16255	$C089 – 16247	Sélection en lecture ROM Non sélection en lecture RAM Sélection en écriture RAM après deux lectures de cette adresse
$C082 – 16254	$C08A – 16246	Sélection en lecture ROM Non sélection en lecture RAM Protection en écriture RAM
$C083 – 16253	$C08B – 16245	Sélection en lecture RAM Sélection en écriture RAM après deux lectures de cette adresse

Les adresses

{ $C080-C083 et $C084-$C087
{ $C088-C08B et $C08C-$C08F

sont équivalentes car on ne tient pas compte du bit 2 *de l'adresse.*

2.1.3 Utilisation avec le DOS 3.3

La carte langage peut être utilisée selon une des manières suivantes :

{ RAM Basic Entier
{ ROM APPLESOFT

ou

{ RAM APPLESOFT
{ ROM Basic entier

Les commandes FP, INT sont alors faciles d'emploi (cf. chap. 1).

2.2 MISE EN ŒUVRE DU SYSTÈME

2.2.1 Introduction

Nous allons réaliser dans ce paragraphe un exemple de programme Pascal, en expliquant ce qui est nécessaire pour mettre en œuvre le système Pascal, le système de gestion de fichiers, l'éditeur de textes, le Compilateur Pascal.

2.2.2 Démarrage du P système

● **APPLE II monodisquette**

Le chargement du système se fait en deux étapes :

— mettez d'abord la disquette marquée APPLE 3 dans votre lecteur, allumez votre APPLE et fermez la porte du lecteur ;

— insérez alors la disquette APPLE 0 et appuyez sur Reset. Le message suivant s'affiche :

```
WELCOME APPLE0, TO
U.C.S.D. PASCAL SYSTEM II.I
CURRENT DATE IS 28-FEB-82
```

Au sommet de l'écran, vous lisez :

```
COMMAND:E(DIT, R(UN, F(ILE, C(OMP, L(IN
```

Vous pouvez à cet instant éditer un fichier, compiler un programme, etc...

● **APPLE II pluridisquettes**

Une seule étape est suffisante. Placez la disquette APPLE 1 dans le lecteur de disquette du connecteur 6, en position 1. Mettez alors en route votre APPLE II. Vous lisez

```
WELCOME APPLE1, TO
U.S.C.D. PASCAL SYSTEM II.I
CURRENT DATE IS 30-JAN-82
```

Vous pouvez à cet instant éditer un fichier, ..., mais pas compiler un programme. Pour cela, mettez la disquette APPLE 2 dans un autre lecteur de disquettes. En effet, la disquette APPLE 1 ne contient pas le compilateur Pascal.

2.2.3 Modification de la date

Prenez la bonne habitude de changer la date au chargement du système. Pour cela tapez la commande F(ILE puis D, vous verrez alors apparaître le message :

```
DATE SET:<1..31>-<JAN..DEC>-<0000..99>
TODAY IS 28-FEB-82
NEW DATE ?
```

Entrez alors la nouvelle date (jour, mois, année), puis tapez Q (retour au niveau commande). La ligne

```
COMMAND:E(DIT, R(UN, F(ILE, C(OMP, L(IN
```

réapparaît en haut de l'écran.

2.2.4 Formatage de nouvelles disquettes

Pour pouvoir commencer à travailler sur votre système, il vous faut d'autres disquettes qu'APPLE 0:, 1:, 2:, 3: qui sont protégées en écriture. Pour utiliser des disquettes vierges, vous devrez les formater (initialiser). Le fichier APPLE3:FORMATTER doit être placé dans un lecteur de disquette. Ensuite vous taperez la commande X (exécution d'un programme). Le système répond :

```
EXECUTE WHAT FILE ?
```

indiquez alors :

```
APPLE3:FORMATTER
```

Le message suivant s'affiche alors sur l'écran :

```
APPLE DISK FORMATTER PROGRAM
FORMAT WHICH DISK (4, 5, 9, 12) ?
```

Enlevez des lecteurs de disquettes toutes les disquettes non vierges, pour ne pas risquer de perdre leur contenu. Préparez par ailleurs une pile de vos disquettes vierges. Insérez alors une disquette vierge à la place de la disquette de chargement (APPLE0: ou 1:) et tapez 4. Le lecteur de disquettes se met à fonctionner et le message suivant est affiché :

```
NOW FORMATTING DISKETTE IN DRIVE 4
```

Lorsque l'opération est terminée, le programme repose la question :

FORMAT WHICH DISK (4, 5, 9..12) ?

Si vous désirez formater une nouvelle disquette, placez-la dans le même lecteur et retapez 4. Sinon insérez votre disquette système et tapez Return. Le système revient au niveau commande (COMMAND:).

Si le programme reconnaît une disquette Pascal déjà formatée, il vous demandera confirmation en affichant le message :

DESTROY DIRECTORY OF nom de disquette ?

Tapez N *et votre disquette sera préservée,* Y *et elle sera réinitialisée.*

2.2.5 Recopie de disquettes

Pour éviter des catastrophes et la perte du système Pascal, nous vous conseillons fortement de recopier les quatre disquettes APPLE (0:, 1:, 2:, 3:) et de n'utiliser que les copies.

Pour copier une disquette, placez-vous au niveau Filer en tapant la commande F(ILE). La ligne suivante s'affiche :

FILER: G, S, N, L, R, C, T, D, Q

Insérez votre disquette à copier dans un lecteur de disquettes et si vous avez un APPLE II pluridisquette, la disquette destination dans un autre lecteur. Tapez ensuite la commande T.

La question suivante s'affiche :

TRANSFER ?

Introduisez maintenant le nom de votre disquette à copier et Return. Le système vérifie que le nom correspond à une disquette en ligne (ici dans un lecteur) puis affiche :

TO WHERE ?

Indiquez le nom de la disquettte destination. Le système vous demande confirmation en affichant :

TRANSFER 280 BLOCKS ? (Y/N)

Si vous vous étiez trompé, tapez N et la commande sera abandonnée, Y (oui) et la recopie est lancée.

Exemple :

TRANSFER ? APPLE0:
TO WHERE ? BLANK: (nom d'une disquette vierge formatée)
TRANSFERT 280 BLOCKS ? (Y/N) Y

Les deux points (:) *dans le nom d'une disquette sont très importants car ils distinguent les noms des fichiers des noms des disquettes.*

Le système recherche ensuite la disquette BLANK:. S'il la trouve, il pose la question :

DESTROY BLANK: ?

Répondez par Y (oui) et BLANK: deviendra la copie parfaite de la disquette source.

Si le Filer ne trouve pas BLANK:, il demande :

PUT IN BLANK:
TYPE <SPACE> TO CONTINUE

Insérez alors BLANK: dans un lecteur (en enlevant APPLE0: par exemple) et tapez un blanc.

Dans le cas d'un système monodisquette, le système vous indiquera quelle disquette est à insérer dans votre lecteur. Une fois la copie terminée, le FILER affichera :

APPLE0: → BLANK:

Tapez ensuite Q(uit pour sortir du Filer et vous retrouver au niveau commande.

2.2.6 Création d'un programme

Ecrivons maintenant un programme. Au niveau commande, sélectionnons l'éditeur de textes en tapant E. Le message suivant apparaît :

>EDIT:
NO WORKFILE IS PRESENT. FILE ?(<RET> FOR NO FILE <ESC-RET> TO EXIT)

En fait vous ne voyez que les 40 premiers caractères du message. Pour lire les autres, tapez CTRL-A, et pour revenir, tapez à nouveau CTRL-A

Ceci est dû au fait que le Pascal gère un écran de 80 colonnes alors que l'écran de l'APPLE n'a que 40 colonnes.

Le mot EDIT signifie que l'éditeur de texte est actif. Ici nous devons créer un nouveau fichier. Tapons RETURN(<RET> FOR NO FILE). La ligne des commandes de l'éditeur s'affiche en haut de l'écran.

>EDIT: A(DJST C(PY D(LTE F(IND I(NSRT J(MP R(PLACE Q(UIT X(CHNG Z(AP

La description détaillée de ces commandes figure au paragraphe 2.5. Nous ne décrivons ici qu'en partie les commandes I (insertion de textes), D (suppression de texte) et Q (sortie de l'éditeur).

Tapons I pour saisir notre programme :

>INSERT: TEXT <BS> A CHAR, <DEL> A LINE <ETX> ACCEPTS, <ESC> ESCAPS

A partir de cet instant, tout ce que vous saisirez apparaîtra sur l'écran. Lorsque vous avez fini, tapez CTRL-C(ETX) et votre insertion sera définitive. Si vous vous trompez, vous pourrez revenir en arrière avec ←, ou supprimer tous les caractères saisis avec ESC.

Pour supprimer des caractères erronés après avoir tapé ETX, vous devrez utiliser une autre commande D. Pour cela, positionnez le curseur sur le premier caractère à supprimer (←, →, CTRL-O, CTRL-L) puis tapez D et déplacez-vous à l'aide de la flèche vers la droite (→) sur les caractères à supprimer.

Lorsque vous avez fini, tapez CTRL-C et la nouvelle ligne corrigée apparaît.

Si vous avez oublié un caractère, vous devrez à nouveau utiliser la commande I. Positionnez d'abord le curseur sur le caractère devant lequel devra être inséré du texte.

Exemple :

PROGAM

placez-vous sur le A, tapez I R CTRL-C et la ligne deviendra

PROGRAM

Le programme ARAROM, écrit en Pascal, convertit un nombre arabe en un nombre romain de même valeur.


```
PROGRAM ARAROM;
CONST
        LGMAX = 18;

TYPE
        CHAINE = RECORD
                        CARACTERES : ARRAY[1..LGMAX] OF CHAR;
                        LONGUEUR   : 0..LGMAX

                END ;
        NOMBREARA = 0..3999;

VAR
        X : NOMBREARA;
        NBROM : CHAINE;
        I :1..LGMAX;

FUNCTION CONVAR (X:NOMBREARA):CHAR;

BEGIN (* DE CONVAR *)

CASE X OF
              1:CONVAR:='I';
              5:CONVAR:='V';
             10:CONVAR:='X';
             50:CONVAR:='L';
            100:CONVAR:='C';
            500:CONVAR:='D'

END  (* DU CASE *);

IF X=1000 THEN CONVAR:='M'

END; (* DE CONVAR *)

FUNCTION PLANCHER(X:NOMBREARA):NOMBREARA;

(* CETTE FONCTION RECHERCHE LE NOMBRE MAXIMUM
   CORRESPONDANT A UN CHIFFRE ROMAIN INFERIEUR
   A X *)

BEGIN (* DE PLANCHER *)

IF X>=1000 THEN PLANCHER:=1000
ELSE IF X>=500 THEN PLANCHER:=500
     ELSE IF X>=100 THEN PLANCHER:=100
          ELSE IF X>=50 THEN PLANCHER:=50
               ELSE IF X>=10 THEN PLANCHER:=10
                    ELSE IF X>=5 THEN PLANCHER:=5
                         ELSE PLANCHER:=1

END; (* DE PLANCHER *)

PROCEDURE CONCAT (C:CHAINE;VAR CH:CHAINE);

(* CETTE PROCEDURE CONCATENE UN CARACTERE
   EN QUEUE D'UNE CHAINE DE CARACTERES *)
```

```
BEGIN (* DE CONCAT *)

CH.LONGUEUR := CH.LONGUEUR + C.LONGUEUR ;
FOR I:= LGMAX DOWNTO 1 DO
BEGIN
IF I> C.LONGUEUR  THEN
                      CH.CARACTERES[I] := CH.CARACTERES[I-C.LONGUEUR]
                  ELSE
                      CH.CARACTERES[I] := C.CARACTERES[I]
END;

END; (* DE CONCAT *)

PROCEDURE TRANSFOR (X:NOMBREARA;VAR CH:CHAINE);

VAR
       C : CHAINE;

BEGIN (* DE TRANSFORM *)

C.LONGUEUR := 1;
CH.LONGUEUR :=0;
WHILE X<>0 DO
           BEGIN
           C.CARACTERES[C.LONGUEUR] :=CONVAR(PLANCHER(X));
           CONCAT(C,CH);
           X := X-PLANCHER(X);
           END;

END; (* DE TRANSFORM *)

BEGIN (* DE ARAROM *)
WHILE NOT EOF DO BEGIN
    FOR I:=1 TO LGMAX DO NBROM.CARACTERES*I§:=' ';
    READLN(X);
    WHILE (X>=4000) DO READLN(X);
    TRANSFOR(X,NBROM);
    WRITE('LE NOMBRE ROMAIN CORRESPONDANT AU NOMBRE ',X,' EST LE SUIVANT ');
    FOR I:=LGMAX DOWNTO 1
        DO WRITE (NBROM.CARACTERES[I]);
    WRITELN;
END;
END. (* DE ARAROM *)
```

Maintenant, sauvegardez ce programme. Lorsque vous êtes au niveau commande de l'éditeur, tapez Q. Le message suivant est alors affiché

>QUIT:

U(PDATE THE WORKFILE AND LEAVE
E(XIT WITHOUT UPDATING
R(ETURN TO THE EDITOR WITHOUT UPDATING
W(RITE TO A FILE NAME AND RETURN
S(AVE WITH SAME NAME AND RETURN

Tapez V pour créer un « fichier de travail » SYSTEM.WRK.TEXT. Le système affiche :

WRITING...
YOUR FILE IS × × × BYTES LONG

Avant d'exécuter le programme, il faut le compiler. La commande R lance la compilation, l'édition de liens si nécessaire et l'exécution. Vous lisez alors :

RUNNING...

Si une erreur a été détectée à la compilation, le système affichera un message semblable à celui-ci :

PROGRAM <<<<
LINE 1, ERROR 18: <SP>(CONTINUE), <ESC>(TERMINATE), E(DIT

Tapez alors E et vous reviendrez automatiquement dans l'éditeur.

2.3 LE NIVEAU COMMANDE

2.3.1 Généralités

Lorsque vous avez chargé le système Pascal sur votre APPLE, vous avez vu s'afficher la ligne suivante :

COMMAND : E(DIT, R(UN, F(ILE, C(OMP, L(INK, X(ECUTE, A(SSEM, D(EBUG, ?

ou la moitié de la ligne si l'écran n'a que 40 caractères. L'autre moitié est obtenue en tapant CTRL-A (cf. § 2.1).

Cette ligne est standard en Pascal UCSD (P système). Toutefois, certaines fonctions ne sont pas implantées sur l'APPLE (D(EBUG) ; si vous tapez D la commande sera ignorée.

Nous allons décrire dans ce paragraphe les diverses commandes brièvement et les fichiers nécessaires pour les mettre en œuvre.

2.3.2 Sémantique succincte des différentes commandes

La ligne COMMAND ci-dessus ne fournit pas la liste des commandes en entier. En effet, la liste complète ne pourrait pas tenir dans une ligne de 80 caractères. C'est la raison pour laquelle s'affiche un point d'interrogation ?. La fin de la ligne est obtenue en tapant un point d'interrogation sur le clavier.

COMMAND : U(SER RESTART, I(NITIALIZE, H(ALT

- **F(ILE**

Taper F(ILE lorsque l'on est au niveau commande place l'utilisateur dans le système de gestion de fichiers. Vous pouvez alors sauvegarder, détruire, lire, transférer des fichiers et le fichier de travail, consulter le catalogue des disquettes, vérifier les disquettes physiquement, fixer définitivement les mauvais secteurs, ... (cf. § 2.4 *Le système de gestion de fichier*).

- **E(DIT**

Vous avez vu que taper E(DITa lorsque l'on est au niveau commande place l'utilisateur dans l'éditeur. Vous pouvez alors insérer du texte dans un fichier, en supprimer, en modifier, etc... (cf. § 2.5 *L'éditeur de textes*).

- **C(OMPILE**

Taper C, lorsque l'on est au niveau commande, lance le compilateur. Celui-ci recherche s'il existe un fichier de travail. Si oui il compile celui-ci, sinon il vous demande le nom du fichier à compiler. Lorsqu'il trouve une erreur, le compilateur l'affiche et demande à l'utilisateur s'il désire modifier le fichier-source et revenir sous l'éditeur, ou continuer la compilation. Quand on parvient à une compilation sans erreur, un fichier-objet en code P est généré.

- **A(SSEMBLE**

Le système Pascal offre à l'utilisateur la possibilité de créer des sous-programmes assembleurs 6502 appelés à partir du Pascal. Pour lancer l'assemblage vous devez taper A lorsque vous êtes au niveau commande. De la même manière que le compilateur, l'assembleur vous demande, lorsqu'il trouve une erreur, si vous voulez retourner sous l'éditeur ou continuer l'assemblage (cf. § 2.6 *Assembleur*).

- **L(INK**

Cette option permet à l'utilisateur de créer un programme exécutable à partir de son fichier compilé, de sous-programmes assembleurs ou Pascal compilés séparément et figurant dans une bibliothèque (cf. § 2.7 *Editeur de liens et exécution d'un programme*).

- **X(ECUTE**

Pour exécuter un programme qui a été compilé, tapez X lorsque vous êtes au niveau commande. Le système vous demande alors quel programme vous voulez exécuter.

EXECUTE WHAT FILE ?

Vous devez fournir le nom d'un fichier. Le système ajoute .CODE à la suite du nom et recherche le fichier sur la disquette. Si vous désirez exécuter un programme dont le nom ne se termine pas par .CODE, donnez comme nom de fichier le nom de votre programme suivi d'un point.

Par exemple il est identique de donner comme nom de fichier :

ROMARA

et

ROMARA.CODE.

Le système lance l'exécution de votre programme. Il existe cependant les cas d'impossibilité suivants :

— le programme n'a pas été « linké » (il reste des étiquettes non définies). Le système affiche :

MUST L(INK FIRT

et retourne au niveau commande. Vous pouvez taper L(INK (cf. ci-dessus) ;

— le fichier fourni ne contient pas du P code (fichier texte, fichier de données...). Vous lisez :

nom du fichier NOT CODE

— la bibliothèque de sous-programmes SYSTEM.LIBRARY n'est pas présente sur la disquette. Vous verrez alors s'afficher le message :

REQUIRED INSTRINSIC(S) NOT AVAILABLE

Ceci indique qu'il manque des sous-programmes qui se trouvent normalement dans la bibliothèèque (graphisme, gestion des poignées de jeu, entrées/sorties...) ;

— le programme utilise les sous-programmes WCHAR ou WSTRING (cf. *TURTLEGRAPHICS*, Toome 3 Pascal UCSD) et le fichier SYSTEM.CHARSET n'est pas préésent sur la disquette. Le programme est exécuté mais rien ne s'affiche suur l'écran.

La commande X(ECUTE poermet d'exécuter des programmes qui ont été mis au point, sans avoir à lees recompiler à chaque exécution.

● **R(UN**

La commande R(UN lance laa compilation du fichier de travail, l'édition de liens si nécessaire, et l'exécutition si elle est possible.

La disquette système de chargement doit rester dans le lecteur de disquettes pendant la commande R(UN *car il y a retour à une partie du niveau commande, entre la compilation, l'édition de liens et l'exécution.*

La commande R(UN est trèès utile lors de la mise au point d'un programme. La mise au point se faitit en utilisant par alternance les commandes E(DIT et R(UN.

● **D(EBUG**

Cette commande fait partie du niveau commande standard en Pascal mais n'est pas implémentée suur l'APPLE II. Si toutefois, vous tapez D au niveau commande, le compilateur est lancé.

● **U(SER RESTART**

Taper la commande U relannce la dernière commande utilisée. Si par exemple, vous tapez les commanndes :

X(ECUTE

et

U(SER RESTART

le programme sera exécuté deuxx fois.

● **I(NITIALIZE**

Taper I réinitialise le systèmee, mais ne recharge pas l'interpréteur P-code en mémoire. La disquette de chargement (Boot) doit être présente. C'est

lla seconde phase de chargement du P-système qui est refaite (cf. *Chargement avec un seul drive*).

Cette commande s'exécute automatiquement lorsqu'il y a une erreur dans le déroulement d'un programme (division par zéro, dépassement de capacité...).

● **H(ALT**

Taper H(ALT revient à éteindre et à rallumer votre APPLE. L'interpréteur P-code est rechargé en mémoire, et vous avez à répéter toute la procédure de mise en route du P-système.

2.3.3 Disquettes fournies

● **Fichiers nécessaires aux commandes du système**

Le système Pascal est un programme beaucoup trop gros pour pouvoir être conservé constamment en mémoire. Vous n'utilisez à un instant donné qu'une petite partie du système. Celui-ci est décomposé en fichiers sur les disquettes et, selon la commande désirée, tel ou tel fichier sera chargé en mémoire. Les fichiers correspondant à une commande peuvent être situés sur n'importe quelle disquette (le système parcourt toutes les disquettes pour les trouver) à l'exception du fichier de travail (Workfile SYSTEM.WRK.TEXT et SYSTEM.SYNTAX).

1. *Il est préférable de toujours vérifier que les fichiers nécessaires à une commande soient présents sur les disquettes utilisées. Le système « s'arrêtera » sinon et vous devrez le recharger.*

2. *L'accès aux fichiers* SYSTEM.PASCAL *et* SYSTEM.LIBRARY *se fait différemment des autres fichiers. Au chargement, le système relève leurs positions. Lorsqu'il devra utiliser ces fichiers, il utilisera ces positions, sans reconsulter le contenu des disquettes. Si vous avez bougé vos disquettes, le système sera perdu et s'arrêtera. Il vous faudra alors le recharger. Pour éviter ce problème il est utile de taper la commande* I(NITIALIZE *lorsque vous changez les disquettes de place.*

Nous donnons ci-dessous la liste des fichiers nécessaires aux différentes parties du système.

Partie du système	Fichiers nécessaires	Disquettes où l'on doit trouver le fichier
Niveau commande	SYSTEM.PASCAL	Ce fichier doit être placé au même endroit qu'au chargement du système. Si la disquette de chargement est présente, et si le fichier SYSTEM.PASCAL ne se trouve plus à la même place, le système « se plante » et on doit le recharger. Si la disquette de chargement est absente, le système demande de la remettre en place
F(ILE	SYSTEM.FILER Fichiers utilisés	Présence sur une disquette quelconque (n° contrôleur, n° disque indifférents) Fichier de travail par défaut (situé sur la disquette de chargement)
E(DIT	SYSTEM.EDITOR Fichier source à éditer	Présence sur une disquette quelconque (n° contrôleur, n° disque indifférents) Fichier de travail par défaut (situé sur la disquette de chargement)
C(OMPILE	SYSTEM.COMPILER SYSTEM.LIBRARY SYSTEM.EDITOR SYSTEM.SYNTAX Fichier-source à compiler	Disquettes quelconques sauf LIBRARY Utilisé si nécessaire Utilisé pour corriger les erreurs Messages d'erreur lorsque l'on corrige les erreurs Disquette quelconque fichier de travail par défaut (sur la disquette de chargement)
A(SSEMBLE	SYSTEM.ASSEMBLER 6500.OP CODES 6500.ERRORS Fichier-source à assembler SYSTEM.EDITOR	Disquettes quelconques Codes 6502. Fichier toujours nécessaire Messages d'erreurs d'assemblage Disquette quelconque fichier de travail par défaut sur la disquette de chargement Pour corriger les erreurs
L(INK	SYSTEM.LINKER Fichier à « linker » Fichier-librairie	 Par défaut SYSTEM.WRK.CODE situé sur la disquette de chargement Par défaut SYSTEM.LIBRARY

Partie du système	Fichiers nécessaires	Disquettes où l'on doit trouver le fichier
X(ECUTE	Fichier à exécuter SYSTEM.LIBRARY SYSTEM.CHARSET	Nécessaire au chargement Nécessaire si le programme utilise les sous-programmes de la bibliothèque Nécessaire si WCHAR, WSTRING utilisés
R(UN	Fichier-source en objet SYSTEM.COMPILER SYSTEM.EDITOR SYSTEM.SYNTAX SYSTEM.LINKER SYSTEM.LIBRARY SYSTEM.PASCAL SYSTEM.CHARSET	Disquette quelconque. Par défaut fichier de travail WRK sur disquette de chargement Nécessaire si fichier texte Correction des erreurs de compilation Messages d'erreur de compilation Nécessaire si le programme utilise des sous-programmes externes Nécessaire avec l'édition de liens (E/S fichiers, graphisme, jeux, ...) Retour au niveau commande entre C, L, X Nécessaire si le programme utilise WCHAR ou WSTRING (cf. § 2.6)
U(SER RESTART	Mêmes fichiers que la commande à répéter	Mêmes positions que la commande à répéter
INITIALIZE	SYSTEM.PASCAL SYSTEM.APPLE SYSTEM.MISCINFO	2e étape 1re étape 2e étape

La disquette de chargement (BOOT) *se trouve dans l'unité de disquette* 1 *du contrôleur du plot* 6 (cf. § 2.2).

● Contenu des disquettes systèmes fournies avec le Pascal

Les fichiers correspondant aux commandes sont disposés comme suit sur les disquettes APPLE :

```
APPLE0:
SYSTEM.PASCAL      41 22-Sep-80    6 Code
SYSTEM.MISCINFO     1  4-May-79   47 Data
SYSTEM.COMPILER    75 19-Sep-80   48 Code
SYSTEM.EDITOR      47 24-Sep-80  123 Code
SYSTEM.FILER       28 18-Sep-80  170 Code
```

```
SYSTEM.LIBRARY    34 19-Sep-80  198 Data
SYSTEM.CHARSET     2 14-Jun-79  232 Data
SYSTEM.SYNTAX     14  1-Aug-80  234 Data
< UNUSED >        32             248
8/8 files, 32 unused, 32 in largest

APPLE1:
SYSTEM.APPLE      32  9-Nov-80    6 Data
SYSTEM.PASCAL     41 22-Sep-80   38 Code
SYSTEM.MISCINFO    1  4-May-79   79 Data
SYSTEM.EDITOR     47 24-Sep-80   80 Code
SYSTEM.FILER      28 18-Sep-80  127 Code
SYSTEM.LIBRARY    34 19-Sep-80  155 Data
SYSTEM.CHARSET     2 14-Jun-79  189 Data
SYSTEM.SYNTAX     14  1-Aug-80  191 Data
< UNUSED >        75             205
8/8 files, 75 unused, 75 in largest

APPLE2:
SYSTEM.COMPILER   75 19-Sep-80    6 Code
SYSTEM.LINKER     24 16-Sep-80   81 Code
SYSTEM.ASSMBLER   54 19-Sep-80  105 Code
6500.OPCODES       2 20-Dec-78  159 Data
6500.ERRORS        7 28-Mar-79  161 Data
< UNUSED >       112            168
5/5 files, 112 unused, 112 in largest

APPLE3:
SYSTEM.APPLE      32  9-Nov-80    6 Data
FORMATTER.CODE     4 14-Aug-80   38 Code
FORMATTER.DATA     6 14-Aug-80   42 Data
LIBRARY.CODE       8 15-Sep-80   48 Code
LIBMAP.CODE        9  1-Aug-80   56 Code
SETUP.CODE        33  7-Feb-79   65 Code
BINDER.CODE        5  1-Aug-80   98 Code
CALC.CODE          8 28-Dec-78  103 Code
LINEFEED.TEXT      4  1-Aug-80  111 Text
LINEFEED.CODE      2  1-Aug-80  115 Code
SOROCGOTO.TEXT     4 29-Mar-79  117 Text
SOROCGOTO.CODE     2 14-Aug-80  121 Code
SOROC.MISCINFO     1 13-Mar-79  123 Data
HAZELGOTO.TEXT     4 29-Mar-79  124 Text
HAZELGOTO.CODE     2 14-Aug-80  128 Code
HAZEL.MISCINFO     1 19-Mar-79  130 Data
CROSSREF.TEXT      8  4-Mar-80  131 Text
CROSSREF.CODE      3  1-Aug-80  139 Code
SPIRODEMO.TEXT     6  4-May-79  142 Text
SPIRODEMO.CODE     2 14-Aug-80  148 Code
HILBERT.TEXT       6  4-May-79  150 Text
HILBERT.CODE       2 14-Aug-80  156 Code
GRAFDEMO.TEXT     28  4-May-79  158 Text
GRAFDEMO.CODE     12 14-Aug-80  186 Code
GRAFCHARS.CODE     3 14-Aug-80  198 Code
```



```
GRAFCHARS.TEXT     6 22-Jun-79  201 Text
TREE.TEXT          8 22-Jun-79  207 Text
TREE.CODE          3 14-Aug-80  215 Code
BALANCED.TEXT     12 22-Jun-79  218 Text
BALANCED.CODE      4 14-Aug-80  230 Code
DISKIO.TEXT       22 14-Aug-80  234 Text
DISKIO.CODE        7 14-Aug-80  256 Code
< UNUSED >        17             263
32/32 files, 17 unused, 17 in largest
```

La disquette APPLE 0 : contient les fichiers nécessaires pour éditer et exécuter un programme.

La disquette APPLE 1 : contient les fichiers nécessaires pour charger le système et éditer un programme.

La disquette APPLE 2 : contient les fichiers nécessaires pour compiler ou assembler un programme.

APPLE 0 : est utilisée avec un système monodisquette.

APPLE 1 et APPLE 2 : sont utilisées avec un système multidisquettes (cf. § 2.2).

La disquette APPLE 3 : contient des programmes utilitaires (formatage disquette...).

2.3.4 Commandes utilisables à tous les niveaux

- **Gestion de l'écran**

L'écran standard de l'APPLE est de 40 caractères par ligne. Celui du système Pascal de 80 caractères par ligne. Il y a donc deux pages d'écran. Vous pouvez passer d'une page à l'autre en tapant CTRL-A. On revient à la première page en tapant à nouveau CTRL-A.

La commande CTRL-Z permet d'avoir une page continue sur l'écran. Le déplacement se fait en suivant le curseur. Cette commande est annulée par CTRL-A.

● **Interruption et relance d'un programme**

● *Arrêt du programme*

La commande CTRL-A provoque l'arrêt du programme en cours et l'affichage du message :

PROGRAM INTERRUPTED BY USER

Vous devez alors taper un blanc et le système se réinitialise.

● *Arrêt des affichages et impressions*

Taper CTRL-F stoppe les affichages et les impressions créés par le programme. Celui-ci n'est pas arrêté, et la sortie des résultats et des données reprend avec le prochain CTRL-F.

● *Arrêt et reprise d'un programme*

La commande CTRL-S permet d'arrêter momentanément l'exécution d'un programme (par exemple pour étudier des données intermédiaires). Retaper CTRL-S relance le programme.

2.4 LE SYSTÈME DE GESTION DE FICHIER

2.4.1 Généralités

Le système de gestion de fichiers est la partie du système Pascal qui gère les disquettes, les périphériques, et le transfert de données entre périphériques. Créer un fichier, le détruire, changer les noms des fichiers, lister un fichier sur un périphérique est du ressort du FILER (système de gestion de fichiers).

Le système de gestion de fichiers est invoqué en tapant la commande F(ILER. Le fichier SYSTEM.FILER est alors chargé en mémoire et vous pouvez enlever la disquette qui le contient :

Vous obtenez sur l'écran :

FILER : G, S, N, L, R, C, T, D, Q

ou

(FILER : G(ET, S(AVE, W(HAT, N(EW, L(DIR, R(EM, C(HNG, T(RANS, D(ATE, Q(UIT)

Tapez ? Il s'affiche alors d'autres commandes du « FILER »

FILER : W, B, E, K, M, P, V, X, Z

ou

(FILER : B(LAD-BLDS, E(XT-DIR, K(RNCH, M(AKE, P(REFIX, V(OLS, X(AMINE, Z(ERO)

Nous verrons au § 2.4.5 la signification de ces commandes.

La seule commande qui vous soit nécessaire actuellement est la commande Q(UIT qui conduit à quitter le système de gestion de fichier et à revenir au niveau commande. Pour cela, le fichier SYSTEM.PASCAL doit être présent sur la disquette de chargement dans le disque 1 du contrôleur du Plot 6 (cf. § 2.3).

2.4.2 Les volumes

Dans la description des commandes du FILER nous appellerons volume un périphérique d'entrée/sortie.

L'écran, le clavier ou une disquette seront considérés par exemple comme des volumes. Un volume peut être référencé par son nom ou par son numéro. Nous vous donnons ci-dessous la correspondance entre numéro de volume et périphérique.

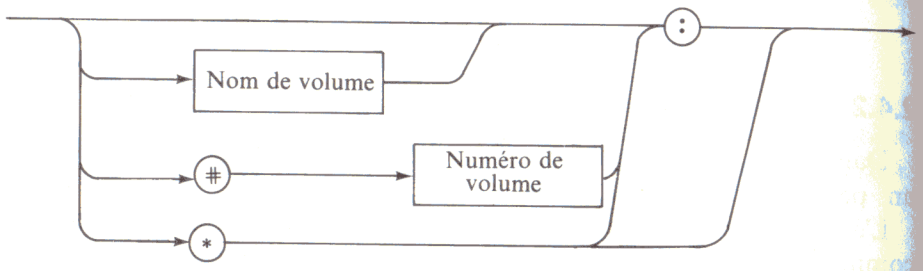
Numéro de volume	Nom du volume	Description du périphérique
0:		(Nom utilisé)
1:	CONSOLE	Ecran/clavier avec écho du clavier
2:	SYSTEM	Ecran/clavier sans écho du clavier
3:		(Nom utilisé)
4:	<nom de disquette>:	Contrôleur plot 6 unité de disquettes 1
5:	<nom de disquette>:	Contrôleur plot 6 unité de disquettes 2
6:	PRINTER:	Imprimante (carte d'interface plot 1)
7:	REMIN	Entrée carte d'interface
8:	REMOUT:	Sortie Plot 2
9:	<nom de disquette>:	Contrôleur plot 5 unité 1
10:	<nom de disquette>:	Contrôleur plot 5 unité 2
11:	<nom de disquette>:	Contrôleur plot 4 unité 1
12:	<nom de disquette>:	Contrôleur plot 4 unité 2

De nombreuses commandes du système de gestion de fichiers nécessitent un volume au moins. La spécification d'un volume consiste à fournir un numéro ou un nom de volume suivi de :. Ces deux points (:) différencient le nom d'un périphérique (disquette) du nom d'un fichier. Ne pas fournir le nom de fichier après les deux points (:) indique que la commande doit être appliquée à toute la disquette.

Un nom de volume doit avoir moins de 7 caractères de longueur et ne doit pas comporter les 4 caractères suivants :

$ = ? ,

La classification d'un volume suit le diagramme suivant :



1. *Un astérisque* * *remplace le nom de la disquette de chargement.*

2. *Il est possible* (*commande* FILER PREFIX) *de fixer le nom d'une disquette par défaut. Cette disquette sera invoquée en tapant seulement* : (cf. § 2.4.5).

2.4.3 Les fichiers

● **Les différents types de fichiers**

Un fichier est un ensemble d'informations stockées sur disquette et désignées par un nom de fichier. Chaque disquette renferme un catalogue qui contient les noms de fichiers et les informations nécessaires à la gestion des fichiers.

L'utilisation d'un fichier est déterminée par son type. Les types suivants sont acceptés par le système Pascal.

Suffixe	Type du fichier	Nom du type
.TEXT	Fichier source éditable	TEXTFILE
.CODE	Fichier de P CODE	CODEFILE
.DATA	Données	DATAFILE
.BAD	Secteurs physiques abîmés	BADFILE
.INFO	(non utilisé)	INFOFILE
.GRAF	(non utilisé)	GRAFFILE
.FOTO	(non utilisé)	FOTOFILE

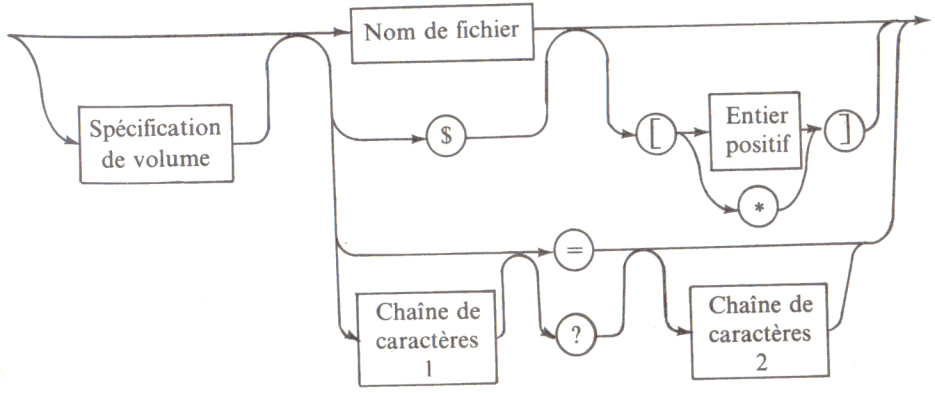
● **Le fichier de travail « WORKFILE »**

Un fichier est utilisé uniquement comme fichier de travail. C'est le SYSTEM.WRK.TEXT (ou .CODE). Lorsque ce fichier est présent sur une disquette, il est pris par défaut lors des commandes R(UN, E(DIT, C(OMPILE, A(SSEMBLE et L(INK.

Le système de gestion de fichiers (FILER) peut détruire le fichier de travail, le sauvegarder et désigner le nouveau fichier à charger dans le fichier de travail.

● **Spécification d'un fichier**

De nombreuses commandes du système Pascal vous amènent à préciser le nom d'un fichier ou sa spécification. Celle-ci est constituée du nom ou du numéro de volume, suivi du nom du fichier. Le diagramme suivant définit la syntaxe exacte :



Exemple :

```
4 : nom fichier 5
APPLES: 5:$*
BLA=BLA
BLA?NC
```

- ***Nom de fichier***

Le nom d'un fichier est une chaîne de quinze caractères au maximum. Les cinq derniers doivent être .TEXT (fichier source) ou .CODE (fichier de P CODE).

Tous les caractères sont autorisés dans un nom de fichier. Toutefois, vous ne devez pas entrer de nom de fichier au clavier contenant les caractères. ( ) ouverture de crochets, ($) symbole du dollar, (=) caractère égal, (?) point d'interrogation, (RETURN) retour chariot. N'entrez pas non plus les caractères de contrôle CTRL-S,-M,-F,-C,-U,-A.

Les caractères sont interprétés différemment par le système de gestion de fichier Pascal de l'APPLE II. Celui-ci n'est pas capable de traiter des fichiers dont le nom contient un des caractères suivants $, = ?

- ***Taille d'un fichier***

Il est possible de spécifier la taille d'un fichier en donnant celle-ci emtre crochets.

Par exemple la spécification BLABLA [4] désigne le fichier BLABLA de taille égale à 4 blocs de 5 102 octets (cf. § 2.4.5).

Il existe des possibilités spéciales pour le nombre de blocs. Si vous metttez BLABLA [0] le système de gestion de fichiers FILER prend, comme taille et place, celles de la zone libre maximum sur la disquette. C'est cette option qui est prise par défaut.

Si vous indiquez comme nombre de blocs une étoile (par exemple BLABLA [*]) le système affecte au fichier la zone la plus grande choisie entre la moitié de la zone libre maximum sur la disquette et la seconde zone libre maximum sur la disquette.

La spécification de la taille d'un fichier est utilisée surtout pour les commandes du FILER T(RANSFER et M(AKE (transfert et création du fichier).

- ***Spécification de plusieurs fichiers à la fois***

Il est possible de désigner plusieurs fichiers à la fois en utilisant les caractères égal (=) et point d'interrogation (?) comme « jokers ». Cette utilisation est limitée à une occurrence par nom de fichier.

Par exemple si vous donnez comme nom de fichier

```
BLA?BLA
```

ou

```
BLA=BLA
```

tous les fichiers dont le nom commence par BLA et qui se terminent par BLA seront traités.

Si vous avez sur votre disquette les fichiers suivants :

```
BLABLA
BLBLA
BLASERBLA
LASERBLAST
BLABLA1
```

et que vous tapez BLA=BLA, les fichiers BLABLA et BLASERBLA correspondront à votre spécification.

2.4.4 Obtention d'informations sur les volumes et les disquettes

- **Liste des volumes**

Si vous tapez V(OLS lorsque le système de gestion de fichier attend une commande (cf. § 2.4.1 *Généralités*), le système listera sur l'écran les volumes actifs (cf. § 2.4.2 *Volumes*).

Avec un APPLE III monodisquette, vous verrez s'afficher sur l'écran :

```
VOLS          ON-LINE
  1           CONSOLE:
  2           SYSTERM:
  4           APPLE0:
ROOT VOL      IS-APPLE0:
PREFIX        IS-APPLE0:
```

Le volume appelé « ROOT VOL » est le volume de la disquette de chargement. Celui appelé « PREFIX » est le volume courant actif pour la disquette (cf. § 2.4.9).

Avec un APPLE II et plusieurs minidisquettes, vous obtiendrez par exemple sur l'écran :

```
VOLS  ON-LINE
  1   CONSOLE:
  2   SYSTERM:
  4   #APPLE1:
  5   #PROJETS:
  6   PRINTER
```

● **Catalogue d'une disquette**

La commande L(IST du système de gestion de fichiers permet d'avoir le catalogue d'une disquette, tout entier ou en partie. Lorsque vous tapez L au niveau commande du FILER (cf. § 2.4.1), le système demande alors

DIR LISTING OF ? (ou DIR LISTING OF WHAT VOL ?)

Vous devez alors préciser un volume et si vous le souhaitez des informations restrictives.

Supposons que la disquette APPLE0: soit présente dans le lecteur de minidisquette qui a servi au chargement du système (#4). Lorsque vous avez le message :

DIR LISTING OF ?

il est équivalent de répondre *, #4, APPLE0:. Le catalogue de APPLE0: s'affiche en entier sous le format suivant :

Nom du fichier	Nombre de blocs de 5 120 utilisés	Date de dernière modification
APPLE0 :		
SYSTEM.PASCAL	36	4-MAY-79
SYSTEM.MISCINF	1	4-MAY-79
SYSTEM.COMPILE	71	30-MAY-79
SYSTEM.EDITOR	45	29-JAN-79
SYSTEM.FILER	28	24-MAY-79
SYSTEM.LIBRARY	36	22-JUN-79
SYSTEM.CHARSET	2	14-JUN-79
SYSTEM.SYNTAX	14	18-APR-79
SYSTEM.WRK.TEXT	10	20-FEB-82

9/9 FILES, 31 blocks UNUSED, 23 blocks IN LARGEST

Vous pouvez ne demander qu'une partie d'un catalogue. Si vous répondez par exemple :

DIR LISTING OF ? SYSTEM.C=X

vous obtiendrez le résultat suivant sur l'écran

```
APPLE0:LINE
SYSTEM.COMPILER     71     30-MAY-79
SYSTEM.CARSET        2     14-JUN-79
2/9 FILES, 45 UNUSED, 45 IN LARGEST
```

Les chiffres indiqués par le système, lors d'un catalogue partiel, sont faux en ce qui concerne la place disponible sur une disquette. Ceci est dû au fait que le dernier fichier listé est considéré comme dernier fichier de la disquette, et que la suite de la disquette est considérée comme libre.

Pour avoir la place disponible sur une disquette, on demandera toujours le catalogue complet.

Il est possible de faire sortir le catalogue vers un autre volume par exemple une imprimante ou un fichier.

Si vous utilisez la commande

DIR LISTING OF ? #4:S=R,PRINTER:

vous obtiendrez le message

```
APPLE0:
SYSTEM.COMPILER     71      4-MAY-79
SYSTEM.EDITOR       45     29-JAN-79
SYSTEM.FILER        28     24-MAY-79
3/11 FILES, 93 UNUSED, 93 IN LARGEST
```

Si vous utilisez la commande

DIR LISTING OF ?#4:,#4:DIR.TEXT

le catalogue du volume #4 est copié dans le fichier DIR.TEXT et le message suivant s'affiche

WRITING ...

● **Catalogue étendu d'une disquette**

La commande E du système de gestion de fichiers de la carte Pascal (cf. § 2.4.1) permet d'obtenir un catalogue de disquette plus complet que celui qui est fourni par la commande L (cf. § 2.4.4).

La syntaxe des échanges avec le système pour la commande E est la même que pour L.

Exemple :

DIR LISTING OF ?#4:

Nom de fichier	Nom de blocs de 256 octets pris	Date	N° 1er bloc	Type
APPLE0:				
SYSTEM.PASCAL	36	4-MAY-79	6	DATA
SYSTEM.MISCINFO	1	4-MAY-79	42	DATA
SYSTEM.COMPILER	71	30-MAY-79	43	CODE
SYSTEM.EDITOR	45	29-JAN-79	114	CODE
SYSTEM.FILER	28	24-MAY-79	159	CODE
SYSTEM.LIBRARY	36	22-JUN-79	187	DATA
SYSTEM.CHARSET	2	14-JUN-79	223	DATA
SYSTEM.SYNTAX	14	18-APR-79	225	TEXT
<UNUSED>	12		239	
SYSTEM.WRK.TEXT	10	2-JUL-81	251	TEXT
<UNUSED>	19		261	
9/9 FILES, 27 UNUSED, 19 IN LARGEST				

Si votre écran admet 80 caractères (carte d'extension), vous aurez en plus le nombre d'octets occupés dans le dernier bloc de chaque fichier.

2.4.5 Structure des informations sur une disquette gérée par le FILER

Nous allons évoquer dans ce paragraphe l'organisation des données sur une disquette et expliquer certains points laissés en suspens dans les paragraphes précédents.

Une disquette est divisée en 35 pistes, de 16 secteurs de 256 octets de données. Elle contient 140 K octets de données utiles. Chaque secteur contient une partie adresse qui contient le numéro de la piste et le numéro du secteur dans la piste et le numéro du secteur dans la piste et les données.

Le système Pascal utilise des blocs de deux secteurs, soit de 512 octets (1/2 K octet). On peut y accéder en utilisant les fonctions UNITWRITE et UNITREAD (cf. § 2.6).

Les 280 blocs d'une disquette ne sont pas tous utilisables pour y stocker vos fichiers.

Les blocs 0 et 1 contiennent le programme d'initialisation de l'APPLE II qui charge en mémoire le système Pascal.

Les blocs 2 à 5 (2 K octets = 4 blocs) contiennent le catalogue de la disquette et toutes les informations nécessaires à la gestion de vos fichiers. Il est possible de gérer jusqu'à 77 fichiers par disquette.

Pour simplifier le système de gestion de fichier FILER, vos fichiers sont stockés sur des blocs continus de pistes voisines.

Ce choix a une conséquence importante pour l'utilisateur. Pour créer, modifier, etc... des fichiers, vous devrez toujours vérifier les zones libres sur la disquette. Une disquette sera une alternance de fichiers et de zones de trous qui sont catalogués lorsque l'on utilise la commande E (cf. § 2.4.4). Plusieurs zones de trous ne permettront pas de stocker un gros fichier.

Lorsque vous avez un fichier à modifier à l'éditeur, vous devez préalablement vérifier qu'il existe sur la disquette une zone libre supérieure à la taille de votre fichier. En effet, que se passe-t-il quand vous quittez l'éditeur (cf. § 2.2 ou 2.4) par l'une des commandes W(RITE ou U(DATE ?. Le système de gestion de fichier sauvegarde la nouvelle version du fichier, et détruit ensuite l'ancienne version. La disquette doit pouvoir contenir les deux versions. Dans le cas contraire, vous ne pourrez jamais sauvegarder vos modifications. Nous vous conseillons d'avoir une zone libre égale environ au double de la taille de votre fichier à éditer.

Il existe une commande du système de gestion de fichiers permettant de déplacer des fichiers et de rassembler les zones libres pour en créer une plus grande.

2.4.6 Commandes générales du système de gestion de fichiers

- **Regroupement des zones libres K(RUNCH**

Cette commande regroupe les diverses zones libres en fin de disquette en déplaçant les fichiers nécessaires. Elle permet de créer des zones libres plus importantes.

La commande K nécessite seulement le nom d'un volume de disquette ou d'un numéro de volume et affiche les noms des fichiers déplacés sur la disquette.

1. *Il est utile de vérifier l'existence, et de fixer les mauvais blocs avant de taper cette commande, pour ne pas risquer d'abîmer un fichier* (cf. 2.4.8).

2. *Ne jamais ouvrir la porte du drive, appuyer sur* Reset, *etc. avant que la commande* K *soit finie. Votre disquette risquerait sinon d'être à jamais détruite. Réinitialiser le système si le fichier* SYSTEM.PASCAL *est déplacé.*

3. *Il est possible de créer la zone libre autour d'un autre bloc que le bloc* 280. *Vous devez alors donner le numéro du bloc, et les fichiers seront rassemblés au début et en fin de la disquette pour laisser la place autour de votre bloc.*

Exemple :

```
K
CRUNCH ? *
ou #4:
FROM END OF DISK, BLOCK 280? (Y/N)
```

taper Yes *lancera l'opération normale en fin de disquette,*
taper No *provoquera l'affichage du message*

```
STARTING AT BLOCK ?
```

vous devez entrer alors un numéro de bloc.

● Création d'un fichier

La commande M(ake réserve une entrée dans le catalogue et une zone sur la disquette sous le nom de fichier donné. Elle est surtout utilisée pour réserver de la place sur une disquette pour un usage futur ou pour récupérer les erreurs de destruction de fichier.

Exemple :

```
M
MAKE WHAT FILE ? #4: PLACE.TEXT [32]
```

Le fichier PLACE.TEX de 32 blocs est alors créé dans la première place libre supérieure à 32 blocs.

Un format est associé à chaque type de fichier. Consultez-les pour connaître les places à réserver (cf. § 2.4.10).

● Changement du nom d'un fichier, d'une disquette

La commande C(HANGE permet de changer le nom d'un fichier ou de changer le nom d'un volume de disquette. Elle nécessite que vous donniez soit deux spécifications de fichiers, soit deux noms de disquettes. Les deux spécifications sont séparées par une virgule ou en tapant RETURN. L'ancienne spécification est donnée en premier. S'il s'agit de la spécification d'un fichier, tout numéro ou nom de volume donné dans la seconde spécification seront ignorés (elles doivent être identiques).

En utilisant les caractères jokers = et ? vous pouvez modifier le nom de plusieurs fichiers en une seule commande.

La commande C(HANGE n'est pas exécutée si le nouveau nom obtenu dépasse en longueur 15 caractères. Le message NOT PROCESSED s'affichera alors.

Exemples :

1. CHANGE? APPLE5:, APPLE4:

vous obtiendrez la réponse suivante si la disquette APPLE5: existe dans un drive

```
APPLE5:      → APPLE4:
```

La disquette APPLE5: a été renommée APPLE4:

2. CHANGE ? APPLE0:EXEMPLE, APPLE1:EXEMPLE1

Le fichier APPLE0:EXEMPLE va être renommé en APPLE0: EXEMPLE1 et non en APPLE1:EXEMPLE1. Le deuxième volume est ignoré.

3. Supposons que vous ayez la disquette suivante :

```
LIVRE:
CH1.TEXT
CH3.TEXT
CH7.TEXT
INTRO.TEXT
```

La commande

CHANGE ? CH = TEXT, CH = OLD.TEXT

créera le catalogue suivant

LIVRE:
CH1.OLD.TEXT
CH3.OLD.TEXT
CH7.OLD.TEXT
INTRO.TEXT

Si vous tapez maintenant la commande

CHANGE ? CH =
TO WHAT ? CHAPITRE =

La modification ne sera possible pour aucun fichier et vous obtiendrez le message

BAD DEST FOR FILES FOUND

Les signes (=),(?) représentent pour la commande C la même chaîne de caractères au départ et à l'arrivée (recopie d'une partie du nom des fichiers).

Si l'on change le nom de la disquette de chargement ou de la disquette courante (*cf.* Commande Prefix), *la commande* C(hange *modifie automatiquement ces données.*

— Destruction d'un fichier

La commande R(emove permet de détruire un ou plusieurs fichiers d'une disquette en restituant la place occupée par les fichiers, qui est alors considérée comme libre, et en libérant les places dans le catalogue.

La commande R(emove nécessite une spécification de fichiers pouvant inclure les caractères jokers = et ? . Les informations relatives à la taille du fichier sont ignorées.

Exemple :

Supposons que nous voulions détruire le fichier créé au paragraphe 2.2 :

APPLE0:ARAROM.TEXT

Tapons R
Le FILER répond

REMOVE ? (ou REMOVE WHAT FILE ?)

tapons alors APPLE0:ARAROM =
Le FILER cherche le(s) fichier(s) et vous pose la question suivante à titre de vérification

UPDATE DIRECTORY ?

Si vous répondez Y (oui), le(s) fichier(s) est(sont) détruit(s). Sinon la commande est abandonnée.

Ne jamais utiliser la commande Remove *pour supprimer le fichier de travail. Utiliser les commandes spéciales* New *et* Get *pour remplacer le fichier de travail. Vous iriez sinon au-devant de catastrophes.*

● Remise à zéro d'une disquette

La commande Z(éro écrase le catalogue d'une disquette en y mettant des zéros. Le contenu de la disquette est perdu, tous les blocs sont considérés comme libres. Mais cette commande ne formate pas la disquette. Elle est équivalente à la séquence suivante

REMOVE ? =

Exemple :

Z
ZERO DIR OF ? (ou ZERO DIR OF WHAT VOL ?)
4:
DESTROY APPLE5: ? Y (abandon de la commande sinon)
DUPLICATE DIR ? N (le système APPLE ne l'accepte pas)
ARE THERE 280 BLKS ON THE DISK ? (Y/N) Y

Dans cette question, si le nombre est différent de 280, il est préférable de reformater la disquette.

NEW VOL NAME ? APPLE4:
APPLE4: CORRECT ? Y (sinon le processus est repris)
APPLE4: ZEROED

2.4.7 Les commandes de manipulation du « Workfile »

● **Remise à zéro du fichier de travail**

La commande N(ew du FILER purge le fichier de travail, de telle sorte qu'aucun fichier ne puisse être trouvé par l'éditeur de textes, le compilateur Pascal, ou l'Assembleur. Le précédent contenu du fichier de travail est perdu et il n'y a plus de fichiers SYSTEM.WRK = jusqu'à une prochaine création du fichier de travail (Editeur de textes ou commande G(et).

Lorsque vous tapez N(ew alors qu'il existe un fichier de travail non vide sur la disquette de chargement, le système vous demande alors

THROW AWAY CURRENT WORKFILE ?

Si vous répondez Yes, le(s) fichier(s) de travail de la disquette de chargement sont détruits.

● **Etat du fichier de travail**

La commande W(hat permet d'obtenir l'identification (nom de fichier) et l'état (sauvegardé/ou non) du fichier de travail.

Si le fichier de travail a été sauvegardé sur une autre disquette que la disquette de chargement, son état sera « non sauvegardé » car il continue à exister sur la disquette de chargement.

● **Remplacement du fichier de travail**

La commande G(et du système de gestion de fichiers permet d'identifier un fichier comme fichier de travail. A la prochaine utilisation de l'éditeur de textes, du Compilateur ou à la prochaine exécution, ce fichier devra être présent sur un volume. Si à ce moment-là le fichier n'est pas présent, le message suivant paraît :

ERROR:WORKFILE LOST

Ce n'est pas au moment du G(et *que l'on a besoin du fichier, mais à sa première utilisation comme fichier.*

S'il existe déjà un fichier de travail SYSTEM.WRK *sur la disquette de chargement lorsque vous faites un* G(et, *le message suivant s'affiche*

THROW AWAY CURRENT WORKFILE ? (cf. N(ew)

Il est inutile de préciser un type (.TEXT ou .CODE). Ils seront tous chargés pour le fichier de travail et vous obtiendrez l'un des messages suivants

TEXT FILE LOADED
ou TEXT CODE FILE LOADED
ou CODE FILE LOADED

● **Sauvegarde du fichier de travail**

La commande S(ave sauvegarde toutes les versions du fichier de travail de la disquette de chargement sous le nom spécifié lors du G(et (cf. § 2.4.7) ou sous un nouveau nom.

Si un fichier existe sous le nom précisé, le système vous demande si vous confirmez votre opération. Dans ce cas, l'ancienne version est supprimée et la nouvelle version du fichier est créée.

Si vous sauvegardez le fichier de travail sur votre disquette de chargement, le ou les fichiers composant le fichier de travail sont nommés. Sinon il y a transfert effectif et le fichier de travail est inchangé.

Ne jamais donner de type (.TEXT *ou* .CODE) *pour le fichier où l'on sauvegarde le fichier de travail. Ils sont ajoutés automatiquement pour chaque type.*

2.4.8 Gestion de la validité physique des disquettes

La gestion de la validité physique des disquettes se fait bloc par bloc. Les mauvais blocs sont mis dans des fichiers de suffixe .BAD et ne sont pas déplaçables (fixage des mauvais blocs).

● **Examen d'une zone physique de disquette**

La commande B(ad blocks recherche les blocs défectueux d'un volume en comparant le code somme du bloc (CRC ou équivalent) avec celui que l'on va calculer pour le bloc. Cette commande nécessite que l'on donne un nom ou un numéro de volume.

Exemple :

```
B
BAD BLOCK SCAN OF ? APPLE0:   (nom de la disquette)
SCAN FOR 280 BLOCKS ? (Y/N) Y   (toute la disquette ?)
```

Si le nombre de blocs indiqué ici est différent de 280, votre disquette est probablement abîmée. Si vous répondez Y, la recherche de mauvais blocs se fait sur toute la disquette. Dans le cas contraire, le système vous demandera l'intervalle à examiner.

Dans 90 % des cas, vous n'aurez pas d'erreur et le message suivant s'affichera

```
0 BAD BLOCKS
```

Si un bloc est abîmé, vous entendrez le lecteur de disquettes claquer et vous verrez s'afficher un message ressemblant à celui-ci :

```
BLOCK 45 IS BAD
BLOCK 140 IS BAD
2 BAD BLOCKS
FILE(S) ENDANGERED:
SYSTEM.COMPILER     43   113
SYSTEM.EDITOR      114   158
```

Vous obtenez la liste des blocs abîmés et celle des fichiers en danger. Rien n'est définitif, les blocs ne sont pas marqués comme mauvais, vous pouvez essayer de récupérer vos fichiers. Toutefois si vous avez une sauvegarde de ces fichiers sur une autre disquette, n'essayez pas de récupérer les mauvais blocs et marquez-les en utilisant la commande X(amine (cf. ci-après).

● **Fixation des blocs endommagés d'une disquette**

La commande X(amine fixe les mauvais blocs d'une disquette. Cette commande nécessite que vous donniez le nom ou le numéro de volume d'une disquette.

Exemple :

```
X
EXAMINE BLOCKS ON ? APPLE0:
(ou EXAMINE BLOCKS ON WHAT VOLUME ?)
BLOCK-RANGE ?
```

A cette étape, vous aurez utilisé la commande B(ad blocks qui vous a donné les mauvais blocs. Vous devez alors indiquer ici ces mauvais blocs.

Dans notre exemple, nous répondrons 45-140.

Tous les blocs compris dans l'intervalle vont alors être lus sur la disquette, réécrits à la même place, puis relus. Si la comparaison entre les deux lectures révèle une erreur, le bloc est marqué et sera considéré comme mauvais.

La commande B(ad Blocks *est redondante avec la commande* eXamine. *Il est toutefois utile de vous en servir car elle est beaucoup plus rapide et peut détecter d'autres erreurs.*

A la fin de la commande X, *les blocs examinés sont diagnostiqués :*

```
BLOCK 45 IS BAD
BLOCK 46 MAY BE OK
        :
        :
BLOCK 139 MAY BE OK
BLOCK 140 IS BAD
FILE (S) ENDANGERED:
SYSTEM.COMPILER  43 113
SYSTEM.EDITOR    114 158
MARK BAD BLOCKS ? (FILES WILL BE REMOVED !)
(Y/N) Y
```

Le système vous offre la possibilité de retirer les fichiers en danger du catalogue et d'isoler les mauvais blocs dans les fichiers indéplaçables. Dans notre exemple, les fichiers SYSTEM.COMPILER et SYSTEM.EDITOR sont détruits et les fichiers BAD.00045.BAD et BAD.00140.BAD sont créés.

Un bloc déclaré MAY BE OK *signale qu'il n'y a pas d'erreur physique (ou probablement pas) sur les secteurs du bloc. Il peut y avoir des erreurs logiques.*

Si le champ adresse d'un secteur d'une disquette devient illisible, le bloc sera détecté par la commande B(ad Blocks, *mais ne pourra pas être fixé, ni atteint. La seule solution sera de reformater la disquette.*

2.4.9 Autres commandes

● **Désignation d'une disquette courante**

La commande P(refix permet de désigner une disquette comme disquette courante, c'est-à-dire comme disquette où l'on ira chercher les fichiers dont on n'aura pas spécifié le volume. Cette commande nécessite que vous

indiquiez un nom ou un numéro de volume. Il n'est pas nécessaire que ce volume soit actif à ce moment-là.

Pour utiliser la disquette courante, il vous suffit d'indiquer le symbole (:).

La disquette courante est initialisée avec le nom de la disquette de chargement. Si vous changez le nom de la disquette courante par la commande C(hange la commande P est lancée automatiquement.

● **Utilisation et mise à jour de la date**

La commande D(ate permet de modifier la date utilisée par le système. Il est possible de fixer le jour, le mois et/ou l'année.

Exemple :

```
DATE SET: <1..31>-<JAN..DEC>-<00..99>
TODAY IS 25-JUN-81
NEW DATE ? 26          (change le jour)
           -JUL        (change le mois)
           --82        (change l'année)
           1/JAN/83    (change les 3 paramètres)
```

Le tiret et le slash (/) peuvent servir de séparateurs. Vous changez la partie de la date que vous désirez. Après avoir tapé Return, la nouvelle date est affichée sur l'écran.

2.4.10 Format des fichiers sur les disquettes

● **Fichiers de type TEXT**

Au début d'un fichier de type TEXT se trouvent deux blocs, dont l'usage est interne à l'éditeur de textes et qui sont respectés par toutes les parties du système.

Quand un programme utilisateur ouvre un fichier dont le titre se termine par .TEXT et se place au début par REWRITE ou par RESET, le système passe au-dessus de cette zone pour se positionner juste derrière.

De plus cet en-tête ne sera pas transmis lors d'un affichage à l'écran ou d'une impression sur l'imprimante. Il le sera par contre en cas de transfert de disquette à disquette.

A la suite de l'en-tête se trouve le texte proprement dit, découpé en pages de 1 024 octets selon le format suivant :

DLE marge texte CR DLE marge text CR ... nuls

Les nuls en fin de page sont présents pour que le nombre de lignes par page soit entier (pas de ligne à cheval entre deux pages de 1 K octet imposé par le compilateur).

DLE marge permet de préciser une largeur de marge. S'il n'y a pas de marge à laisser, il est inutile. Sinon on met comme marge 32, plus la largeur réelle de la marge.

● **Fichiers de type DATA**

Les formats des fichiers de type DATA sont à la libre volonté de leur utilisateur.

● **Fichiers de type CODE**

Un fichier de type CODE est constitué de segments. Leur nombre peut aller jusqu'à 16. Le bloc zéro d'un fichier de type CODE contient les informations nécessaires pour gérer les 16 segments de code. Ces informations sont représentées comme suit :

```
RECORD
DISKINGO  : ARRAY 0..15 OF RECORD
              CODE LENG,    (*LONGUEUR DU SEGMENT*)
              CODEADDR      (*ADRESSE DU SEGMENT*)
                : INTEGER END;
SEGNAME   : ARRAY 0..15 OF PACKED ARRAY 0..7 OF CHAR;
            (*TABLEAU DES NOMS DU PROGRAMME PRIN-
            CIPAL ET DES PROCEDURES COMPILEES*)
SEGKIND   : ARRAY 0..15 OF (LINKED, HOSTSEG, SEGPROC,
            UNITSEG, SEPRTSEG, UNLINKED-INTRINS,
            LINKED-INTRINS, DATASEG);
            (*TABLEAU DES TYPES DES SEGMENTS :
              LINKED        segment totalement exécutable
              HOSTSEG       programme hôte
              SEGPROC       une procédure PASCAL dans un
                            segment (non utilisé)
```

```
                UNITSEG              unité compilée régulièrement
                SEPRTSEG             unité compilée séparément
                UNLINKED-INTRINS     unité contenant des appels à
                                     des fonctions ou des procé-
                                     dures externes
                LINKED-INTRINS       unité intrinsèque complète
                DATASEG              segment de données
                (cf. § 2.6 Pascal pour plus de précisions)*)
TEXTADDR : ARRAY 0..15 OF INTEGER;
           (*pour une unité, valeur du nº de bloc où commence
           l'interface avec l'unité, sinon 0*)
SEGINFO  : PACKED ARRAY 0..15 OF PACKED RECORD
           SEGNUM : 0..255 ; (*Bits 0 à 7 numéros de segment pour
                    le numéro de segment de code)
           MTYPE  : 0..15 ; (*TYPE DE P CODE
                             0 ancien
                             1 P code msb en 1er (remplissage→)
                                          msb en 2e (dans le mot
                                          (←sens du remplissage)
           UNUSED : 0..1 ;
           VERSION : 0..7 (*numéro de version)
           END ;
INTRINS-SEGS : SET OF 0..31 ;
               (*UNITES INTRINSEQUES DE LA BIBLIOTHE-
               QUE NECESSAIRES*)
COMMENT : STRING
END ;
```

2.5 L'ÉDITEUR DE TEXTES

2.5.1 Généralités

L'éditeur de textes est assez puissant. Nous avons vu au paragraphe 1.2 un exemple d'utilisation de l'éditeur. Nous allons reprendre les diverses commandes et expliquer leurs possibilités. Rappelons tout d'abord comment mettre en œuvre l'éditeur de textes.

● **Mise en œuvre sur un système monodisquette**

D'une manière générale, le processus à mettre en œuvre pour éditer un fichier qui n'est pas sur la disquette de chargement est le suivant :

— purger le fichier de travail (commande N(ew du FILER),
— transférer le fichier à éditer sur la disquette de chargement (commande T(ransfer du Filer),
— prendre ce fichier comme fichier de travail (commande G(et du Filer),
— éditer le fichier et faire les modifications,
— quitter l'éditeur en mettant à jour le fichier de travail (commandes Quit et Update de l'éditeur),
— sauvegarde du fichier de travail (commande S(ave du Filer),
— transfert en sens inverse des fichiers corrigés.

● **Mise en œuvre sur un système multidisquettes**

La manière de procéder est la même, sinon que le transfert est effectué directement par les commandes G(et et S(ave du Filer (cf. § 2.4.8). Les manipulations de disquettes sont grandement allégées.

● **Commandes de l'éditeur**

Pour rentrer dans l'éditeur, il vous faut taper E(DIT lorsque vous êtes au niveau commande du système. L'éditeur est chargé. Il regarde s'il existe un fichier de travail. Si oui, il le charge en mémoire, affiche sa première page de texte et se met en attente d'une commande. Si non, il affiche le message suivant :

```
>EDIT :
NO WORKFILE IS PRESENT. FILE? (<RET> FOR NO
FILE <ESC-RET>> TO EXIT)
```

Vous pouvez alors introduire un nom de fichier. La commande G(et est faite implicitement. Si vous ne mettez aucun nom, un nouveau fichier est créé. La ligne des commandes de l'éditeur est la suivante :

```
>EDIT : A(DJST C(PY D(LETE F(IND I(NSRT J(MP
R(PLACE Q(UIT X(CHNG Z(AP
```

Dans ce paragraphe, nous allons étudier les commandes les unes après les autres.

● **Gestion de l'écran et déplacements du curseur**

Lorsque vous travaillez sous l'éditeur, vous voyez affichée devant vous une page d'écran sur 40 caractères. Or l'éditeur de textes travaille sur des lignes pouvant aller jusqu'à 80 caractères. On utilisera, comme pour tout le système, les caractères CTRL-A et CTRL-Z pour suivre le curseur sur l'écran.

Sur l'écran une page de texte s'affiche à un instant donné. Pour pouvoir accéder à une phrase, un mot ou un caractère situé au milieu de l'écran, pour faire une insertion, une suppression ou un remplacement de caractères, vous pouvez déplacer le curseur à l'aide des 4 caractères suivants :

CTRL-O : montée du curseur d'une ligne,
CTRL-L : descente du curseur d'une ligne,
→ (HT) : déplacement du curseur vers la droite,
← (BS) : déplacement du curseur en arrière vers la gauche.

Taper un numéro avant ces caractères conduit à une répétition du caractère voulu. Taper un slash (/) conduit à une répétition infinie. Vous pouvez ainsi voir tous les textes que vous allez modifier.

Il existe une direction (et un sens de déplacement) par défaut pour certains caractères (vers l'avant ou en arrière). A l'entrée dans l'éditeur, la direction est prise par défaut vers l'avant.

Mais taper un des caractères , < — changera la direction vers l'arrière, et taper un des caractères . > + changera la direction vers l'avant.

Certaines commandes ou caractères sont affectés par cette direction. Ce sont les suivants :

blanc	avance ou recule d'un cran selon la direction,
CTRL-I	se positionne à la plus proche tabulation selon la direction,
Tab	avant ou après le curseur actuel,
Return	remonte ou descend d'une ligne selon la direction.

Taper sur le signe égal (=) positionnera le curseur sur le dernier texte ajouté, remplacé, ou recherché.

2.5.2 Commandes de modification du texte

● **Insertion de texte**

● *Généralités*

Pour insérer du texte dans un fichier, vous devez procéder en deux étapes :

— positionnement du curseur à l'emplacement voulu pour l'insertion,
— taper I(nert au niveau commande de l'éditeur.

Vous voyez alors s'afficher la ligne suivante :

>INSERT:TEXT <BS> A CHAR, <DEL> A LINE
<ETX> ACCEPTS, <ESC>, ESCAPES

Le texte que vous tapez sera alors inséré entre le caractère placé juste à gauche du curseur et le caractère sur lequel est situé le curseur.

Pour aller plus vite, l'éditeur de textes ne remet pas le texte à jour sur l'écran tout de suite et crée un trou sur l'écran pour votre insertion. Le texte reparaîtra dans sa nouvelle version lorsque vous aurez tapé CTRL-C (<<ETX>) pour accepter l'insertion, ou dans son ancienne version si vous tapez ESC (abandon de l'insertion).

Si vous faites une erreur durant l'insertion de votre texte, plusieurs solutions sont possibles :

— taper <—(<BS>) qui annule le dernier caractère,
— taper CTRL-X(<DEL>) qui annule la dernière ligne,
— taper ESC qui supprime le texte inséré, puis I (entrée à nouveau de texxte),
— taper CTRL-C qui accepte le texte, revenir sur l'erreur et la corriger.

Lorsque vous avez fini votre insertion par CTRL-C, celle-ci est disponible dans le buffer utilisé pour copier des textes. Il vous est alors possible de la reproduire à divers endroits du fichier à l'aide de la commande COPY de l'éditeur. Cette possibilité n'existe pas si vous finissez en annulant l'insertion (en terminant par <ESC>).

La taille maximum d'un fichier est de 38 blocs, soit environ 18 400 caractères. Si vous approchez de la taille maximum de votre fichier, vous verrez s'afficher le message suivant sur l'écran :

ERROR : PLEASE FINISH UP THE INSERTION. PLEASE PRESS <SPACEBAR> TO CONTINUE

Vous devez alors taper un blanc pour revenir en mode insertion. Vous pouvez finir votre phrase avant de quitter le mode insertion. Si vous avez encore beaucoup de texte à ajouter, vous devez couper votre fichier en deux ou plusieurs morceaux.

Si vous continuez à entrer du texte après le message d'erreur précédent, votre buffer se sature et vous recevez un message plus grave :

ERROR : BUFFER OVERFLOW!!!! PLEASE PRESS <SPACEBAR> TO CONTINUE

Vous devez alors taper un blanc, ce qui vous sort du mode insertion, et toute tentative d'y retourner conduira au message d'erreur suivant :

ERROR : NO ROOM TO INSERT. PLEASE PRESS <SPACEBAR> TO CONTINUE

- ***Caractères spéciaux***

Les caractères ([) et (]) sont utilisés pour les tableaux en Pascal. Car ils ne sont pas à première vue accessibles au clavier. Pour les atteindre, vous devrez taper CTRL-K pour [, SHIFT-M pour].

- ***Formatage de texte***

Le schéma de formatage est fixé en utilisant les commandes S(ET et E(nvironnement de l'éditeur (cf. ci-dessous). Il y a deux choix à faire :

— ajustement automatique de la marge gauche d'une ligne sous le premier caractère non blanc de la ligne précédente. Nous appellerons ceci ajustement automatique de marge,

— utilisation de marges et limitation du texte entre les marges.

Ces choix sont deux paramètres indépendants, mais nous allons maintenant examiner les quatre cas possibles :

* AJUSTEMENT AUTOMATIQUE SANS UTILISATION DE MARGES

C'est l'option prise par défaut par l'éditeur et c'est l'option utile pour écrire des programmes Pascal ou Fortran.

Vous devez terminer chaque ligne en tapant Return. Le curseur se positionne alors sous le premier caractère non blanc de la ligne du dessus. Chaque ligne commencera à la même marge que la précédente.

Pour changer la marge, il vous suffit de taper un (ou plusieurs) blancs pour augmenter la marge, une ou plusieurs flèches vers la gauche pour diminuer la marge ou CTRL-Q pour annuler la marge.

Lorsque vous commencez une insertion, la marge retenue est celle de la ligne où commence l'insertion.

Exemple : Tapez les caractères suivants :

ONE <return> <space> <space> <space> DEUX <return> QUATRE <return> <←> CINQ <return>

vous obtenez le résultat suivant sur l'écran

ONE
DEUX
QUATRE
CINQ

* UTILISATION DE MARGES SANS AJUSTEMENT AUTOMATIQUE D'UNE LIGNE SUR LA PRÉCÉDENTE

C'est le mode utilisé pour écrire des textes tels que des lettres ou d'autres documents. Le texte est cadré par une marge de paragraphe, une marge gauche et une marge droite.

Si un mot dépasse une marge droite, un caractère RETURN est inséré automatiquement avant le mot et ce qui suit ce mot. Pour l'éditeur, un mot est défini comme une chaîne de caractères compris entre deux délimiteurs. Un délimiteur de mot est l'un des caractères suivants : blanc, return, début et fin (avant CTRL-C) de l'insertion en cours. Cependant, le tiret n'est pas reconnu comme séparateur de mots.

Les paragraphes sont séparés par des délimiteurs de paragraphes qui sont les suivants : une ligne blanche (plus d'un Return), début ou fin de fichier et une ligne commençant avec la commande C(ommande character (cf. *Environnement*).

* AJUSTEMENT AUTOMATIQUE ET UTILISATION DE MARGES

La marge gauche est gérée par ajustement automatique sous le premier caractère non blanc de la ligne du dessus. La marge droite est gérée pour

l'insertion réalisée. Il n'y a pas de contrôle pour les caractères situés à droite du curseur excepté un point d'exclamation en colonne 80 si la ligne déborde de 80 caractères.

Changer la marge se fait comme suit : blanc pour augmenter la marge, ← pour la diminuer, CTRL-Q pour l'annuler. Ces caractères ne servent que si vous êtes placé sur le premier caractère d'une ligne.

* AUCUN FORMATAGE DE TEXTE

C'est à vous de gérer les marges à gauche et à droite, les paragraphes, etc. Les caractères peuvent être ajoutés partout sur l'écran. Si vous dépassez la colonne 71, vous entendrez un bip sonore et si vous dépassez la colonne 80, vous verrez un point d'exclamation dans celle-ci. Ce sont les seuls contrôles effectués.

● Suppression de texte dans un fichier

La commande D(elete de l'éditeur nous fait entrer dans le mode suppression de texte. Vous voyez alors apparaître la ligne suivante sur l'écran :

>DELETE: < > <MOVING COMMANDS> <ETX>
TO DELETE, <ESC> TO ABORT

Le curseur doit avoir été placé préalablement sur la position où doit commencer la suppression. Si vous voulez supprimer des caractères en allant vers la droite, le curseur doit avoir été placé sur le premier caractère à détruire. Si vous vous déplacez vers la gauche, le curseur doit être situé juste à droite du premier caractère à détruire.

Au fur et à mesure de la destruction, les caractères sont supprimés et effacés sur l'écran. Toutefois, en faisant la manœuvre inverse, vous verrez réapparaître les caractères détruits jusqu'à la position initiale du curseur qui a été conservée en mémoire.

Lorsque vous aurez fini votre manipulation, vous devrez taper CTRL-C pour l'enregistrer définitivement. Si par contre vous désirez annuler les modifications, vous devrez taper ESC. Dans les deux cas, vous sortirez du mode suppression de texte et le curseur sera positionné à la place initiale.

Tous les caractères compris entre le curseur et sa position initiale sont stockés pour pouvoir être recopiés, que vous sortiez du mode suppression de texte en annulant ou non la commande (<ESC> ou <ETX>). Vous

récupérez ainsi les caractères pour les remettre en place si vous avez fait une erreur. De même vous dupliquez les caractères en plusieurs endroits du texte, ou enfin vous déplacez une zone de texte. Si vous essayez de supprimer plus de texte que le buffer récepteur n'en peut contenir, vous obtiendrez le message suivant, lorsque vous taperez CTRL-C pour enregistrer la suppression :

THERE IS NO ROOM TO COPY THE DELETION.
DO YOU WISH TO DELETE ANY WAY ? (Y/N)

Dans les deux cas, le buffer reste dans l'état où il était avant la commande D(elete. Si vous répondez Y(oui), le texte est détruit tout de même. Cette question n'est pas posée si vous essayez de copier le texte placé dans le buffer de copie. Vous recevrez le message :

ERROR : NO ROOM. PLEASE PRESS
<SPACEBAR> TO CONTINUE.

● Remplacement de caractères

Si vous frappez X au clavier lorsque vous êtes au niveau commande de l'éditeur, vous serez placé au niveau remplacement de caractères et la ligne suivante apparaîtra :

>X(CHANGE: TEXT <BS> A CHAR <ESC>
ESCAPES; <ETX> ACCEPTS

Vous pouvez alors :

— soit remplacer le caractère sur lequel est positionné le curseur en tapant un nouveau caractère ; le curseur se déplace alors d'un cran vers la droite ;

— soit taper un caractère < BS > (flèche vers la gauche), ce qui fait réapparaître le caractère placé là avant modification ;

— soit frapper ESC, ce qui annule les modifications effectuées et fait reparaître les caractères modifiés depuis que vous êtes passé en mode remplacement ;

— soit frapper CTRL-C qui entérine définitivement les modifications réalisées.

Exemple : Supposons que vous vouliez remplacer la ligne

WRITE("SALUT");

par la ligne suivante :

```
WRITE("HELLO");
```

Pour faire cette modification, suivez le processus :

— placer le curseur sur le S de "SALUT",
— taper XHELLO,
— taper CTRL-C pour rendre cette modification définitive, ESC pour annuler la modification.

● **Suppression du texte compris entre la position courante du curseur et le début du dernier texte inséré, remplacé ou trouvé (ZAP)**

Cette commande est activée en tapant Z(AP au niveau commande de l'éditeur. Elle est utile par exemple dans ce cas :

```
WRITELN('POMME');
```

— positionnement du curseur à la fin de POMME
— commande I APPLE CTRL-C

```
WRITELN('POMMEAPPLE');
```

— positionnement du curseur sur le P de POMME
— commande Z(AP

```
WRITELN('APPLE');
```

Si vous essayez de supprimer plus de 80 caractères à la fois, le message d'erreur suivant s'affichera :

```
WARNING! YOU ARE ABOUT TO ZAP MORE THAN 80
CHARS, DO YOU WISH TO ZAP? (Y/N)
```

Tapez non et la commande n'est pas effectuée.
Tapez oui(Y) et le nouveau message d'erreur s'affiche :

```
THERE IS NO ROOM TO COPY THE DELETION.
DO YOU WISH TO DELETE ANYWAY? (Y/N)
```

Tapez oui(Y) et la commande est traitée sans que les caractères détruits soient recopiés dans le buffer de recopie.

● **Copie d'un texte dans un fichier**

Lorsque vous êtes au niveau commande de l'éditeur, et que vous tapez la commande C, vous entrez dans le mode copie de l'éditeur et la ligne suivante s'affiche sur l'écran :

```
>COPY: B(UFFER F(ROM FILE <ESC>
```

Vous pouvez alors soit insérer le texte situé dans le buffer de copie avant le curseur, soit insérer un fichier externe au même endroit.

● ***Copie du buffer***

En tapant B, le texte compris dans le buffer interne est immédiatement inséré avant le curseur. A la fin de la copie, le curseur est placé sur le premier caractère du texte inséré à partir du buffer. A la différence du mode insertion, la pagination du texte inséré n'est pas modifiée par les marges du contexte.

La commande C(opy du Buffer est utilisée pour recopier un texte que l'on vient d'insérer ou pour recopier un texte que l'on déplace dans le fichier (commandes D(elete et C(opy). Le buffer interne est modifié par les commandes de l'éditeur I(nsert, D(elete et Z(ap.

● ***Copie d'un fichier externe***

En tapant F(ROM FILE lorsque vous êtes au niveau copie de l'éditeur, vous voyez s'afficher le message suivant :

```
>COPY: FROM WHAT FILE MARKER, MARKER ?
```

Vous pouvez faire insérer avant le curseur, le contenu d'un autre fichier situé sur une disquette de votre fichier. Pour cela, vous devez donner le nom d'un fichier présent sur une disquette « en ligne » ; le suffixe .TEXT est inséré automatiquement à la fin du nom du fichier, vous n'avez pas à la préciser obligatoirement. Si vous voulez donner un nom d'un fichier qui n'a pas de suffixe, il vous suffira de taper un point (.) après la spécification complète du fichier. Si vous tapez Return, le fichier indiqué sera inséré en entier dans votre fichier. Il existe la possibilité de n'introduire qu'une partie du fichier dans votre texte. Pour cela vous aurez défini des marques de référence dans le fichier inclus (cf. *Commande Set*). Précisez les marques encadrant la portion de fichier que vous voulez inclure.

Si le nom de fichier n'est suivi d'aucun nom de marque, tout le fichier est inclus.

Si vous faites suivre le nom du fichier par "nom de marque", le fichier est inclus jusqu'à la première occurrence de la marque. Si le nom de fichier est suivi par "nom de marque", la partie du fichier située après la marque est incluse.

Si vous précisez deux marques, seule la partie du fichier située entre les deux marques est incluse dans votre fichier.

Lorsque l'insertion de tout ou partie du fichier indiqué a été réalisée, le curseur est placé sur le premier caractère copié et le message suivant s'affiche sur l'écran :

BE SURE ORIGINAL SYSTEM.EDITOR DISK IS IN SAME DRIVE: RETURN TO CONTINUE

Ce message est surtout destiné à prévenir les utilisateurs d'APPLE II monodisquette de bien remettre la disquette de chargement en place.

Si votre fichier ne peut pas contenir tout le texte à inclure, l'éditeur de textes en conserve le maximum possible et affiche le message :

ERROR: BUFFER OVERFLOW. PLEASE PRESS <SPACEBAR> TO CONTINUE.

Lorsque vous tapez un blanc, la copie sera faite au maximum, mais non en entier. Prenez vos précautions pour la rendre propre et découpez votre fichier en deux.

Exemple : Supposons qu'il existe un programme P1.TEXT et que vous vouliez le modifier pour en faire le programme P2.TEXT tout en conservant l'original. Vous pouvez suivre le processus suivant :

— éditer P2.TEXT,
— taper les commandes C(opy, F(ROM FILE et APPLE4:P1,
— le fichier P2.TEXT contient alors P1 que vous pouvez alors modifier comme bon vous semble.

2.5.3 Commandes de modification de la position du curseur

● **Positionnement du curseur**

La commande J(ump permet de positionner le curseur :

— soit au début du fichier,
— soit en fin du fichier,
— soit sur une marque (cf. § 2.5.5).

Cette commande est activée en tapant J lorsque l'on est au niveau commande de l'éditeur. Il apparaît alors sur l'écran la ligne suivante :

>JUMP: B(EGINNING E(ND M(ARKER <ESC>

Taper B place le curseur en tête du fichier et la première page de texte est alors affichée.

Taper E place le curseur en fin du fichier et la dernière page de texte est alors affichée.

Taper M provoque l'affichage de la ligne :

JUMP TO WHAT MARKER ?

Vous devez alors indiquer une marque. Si l'éditeur trouve la marque dans le fichier, le curseur est placé à la position de la marque et il affiche la page de texte correspondante. Sinon, vous obtiendrez le message d'erreur suivant :

ERROR: NOT THERE. PLEASE PRESS <SPACEBAR> TO CONTINUE.

et le curseur ne sera pas déplacé.

● **Changement de page de texte affiché**

La commande P(age de l'éditeur déplace le curseur d'une page de texte dans la direction de déplacement active (cf. § 2.5.1). Le curseur est alors placé en début d'une ligne. Un facteur de répétition peut être utilisé pour se déplacer de plusieurs pages à la fois.

2.5.4 Commandes de travail sur des chaînes de caractères

● **Introduction**

Il existe deux possibilités de travail sur les chaînes de caractères :

— travail sur les caractères qui se suivent,
— travail sur les caractères réunis en mots.

Selon la première possibilité, la chaîne « son » par exemple sera trouvée dans le mot maison. Selon la seconde possibilité, le mot « son » ne sera pas trouvé. Le choix du mode de travail se fait dans l'environnement.

● **Recherche d'une chaîne de caractères**

La commande F(ind permet de rechercher les occurrences d'une chaîne de caractères dans un fichier. Pour y accéder, il vous suffira de taper F au niveau commande de l'éditeur. Vous voyez alors s'afficher le message :

>FIND 1:L(IT <TARGET> – > (si vous travaillez par mot)

ou

>FIND 1:T(OK <TARGET> – > (si vous travaillez par caractères)

Vous pouvez passer d'un mode de travail à l'autre en tapant T(OKEN pour travailler par mot, ou L(ITTERAL pour travailler par caractères.

Lorsque l'un des messages ci-dessus s'affiche, vous pouvez taper :

— L ou T pour changer de mode de travail,
— la chaîne de caractères à rechercher, encadrée par deux caractères délimiteurs identiques.

Exemple :

```
'MOT'
/WRITE/
```

Dès que vous avez tapé le deuxième caractère délimiteur, le curseur se positionne sur la fin de la première occurrence de la chaîne demandée.

Il vous est possible de rechercher la neuvième occurrence d'une chaîne de caractères dans un fichier. Il vous faudra préciser un facteur de répétition (nombre entier compris entre 0 et 3999) lorsque vous taperez la commande F.

Exemple : Supposons que vous ayez le fichier suivant :

```
BEGIN
WRITE X;
PAGE;
WRITE ('LE RESULTAT OBTENU EST', RES(X))
END.
```

Plaçons-nous au début du fichier. Tapons pour cela la commande JB. (Jump Beginning.)

Précisons ensuite la direction de déplacement + et tapons :

JB+2F/WRITE/

ou

JB+F/WRITE/

Dans le premier cas vous verrez apparaître le message :

<FIND 2:L(IT <TARGET> – > /WRITE/

et le curseur se positionnera sur le premier caractère suivant le E de la deuxième occurrence de WRITE.

Dans le second cas vous verrez apparaître le message :

<FIND 1:L(IT <TARGET> – > /WRITE/

et le curseur se positionnera sur le banc compris entre X et E de la fin de la première occurrence de WRITE.

1. *Si l'éditeur ne trouve rien dans la direction de déplacement du mode de travail désiré, la n*^{ème} *occurrence de la chaîne, il affiche le message d'erreur suivant :*

ERROR: PATTERN NOT IN THE FILE. PLEASE PRESS <SPACEBAR> TO CONTINUE

Il ne recherchera pas la chaîne dans l'autre direction.

2. *Tous les caractères sauf les lettres et les chiffres peuvent être utilisés comme caractères délimiteurs. Si vous oubliez d'en préciser, vous obtiendrez le message suivant :*

ERROR: INVALID DELIMITER. PLEASE PRESS <SPACEBAR> TO CONTINUE.

Essayez à nouveau avec un délimiteur correct.

3. *Taper* <ESC> *arrête toujours la commande* F(IND.

4. *Taper* S *comme chaîne à rechercher* (Target) *indique à l'éditeur de rechercher la prochaine occurrence de la dernière chaîne recherchée.*

Il est par exemple équivalent de taper :

JB+2F/WRITE/

et

JB+F/WRITE/FS

Dans les deux cas, le curseur sera positionné sur le caractère qui suit le deuxième WRITE *du fichier.*

● **Remplacement d'une chaîne de caractères**

La commande R(eplace permet de remplacer les occurrences d'une chaîne de caractères par une autre chaîne de caractères. Pour entrer dans le mode remplacement de chaînes de caractères, il vous suffit de taper la lettre R lorsque vous êtes au niveau commande de l'éditeur. Deux modes de fonctionnement sont possibles :

— par caractère (ou littéral)
— par mot (ou T(oken).

Le mode de fonctionnement est positionné avec l'environnement (cf. § 2.5.5). Il peut être modifié par la commande R(eplace. Selon le mode de travail, vous obtiendrez le message suivant sur l'écran :

>REPLACE n:L(IT V(FY <TARG> <SUB> – >
(mode mot)

ou

>REPLACE n:T(OK V(FY <TARG> <SUB> – >
(mode caractère)

n est le facteur de répétition de la commande.

Vous pouvez alors choisir entre :

— changer de mode de travail pour la commande,
— indiquer si vous voulez préciser votre choix à chaque occurrence de remplacement (option V(FY),
— préciser la chaîne origine et la chaîne de remplacement. Ces chaînes doivent être indiquées entre deux délimiteurs chacune, (exemple /WRITE/ /WRITELN/).

Dès que vous avez frappé le deuxième délimiteur de la chaîne de remplacement, l'éditeur cherche dans la direction de déplacement (cf. § 2.5.1) les n premières occurrences de la chaîne à remplacer et leur substitue la nouvelle chaîne, après confirmation si vous avez utilisé l'option Verify. Le curseur est placé à la fin de la $n^{ème}$ occurrence de la chaîne à remplacer.

Si l'éditeur n'arrive pas à trouver les n occurrences demandées dans la direction de déplacement, il ne continue pas à partir de l'autre extrémité du fichier et affiche le message d'erreur :

ERROR: PATTERN NOT IN THE FILE.PLEASE PRESS <SPACEBAR> TO CONTINUE.

La confirmation de la commande de remplacement peut être demandée par l'éditeur à chaque occurrence de la chaîne. Pour cela, il suffit de taper V avant la chaîne à remplacer. Rien n'apparaît à ce moment-là, mais l'option est activée. A chaque occurrence de la chaîne, vous verrez apparaître le message :

>REPLACE: <ESC> ABORTS, 'R' REPLACES, ' ' DOESN'T

A cet instant, taper :

R provoquera le remplacement de l'ancienne chaîne par la nouvelle.

Un blanc, le passage à la prochaine occurrence de la chaîne ou au niveau commande de l'éditeur.

ESC l'abandon de la commande.

Entre deux appels de la commande R(eplace, vous pouvez conserver les chaînes indiquées :

S /nouvelle chaîne/	même commande avec la même ancienne chaîne
/ancienne chaîne/ S	même commande avec la même nouvelle chaîne
SS	même commande.

2.5.5 Commandes de positionnement de l'environnement (marges, marqueurs, formatage...)

● **Déplacement des caractères sur une ligne A(djust**

Taper A au niveau commande de l'éditeur conduit à l'affichage du message :

>ADJUST: L(JUST R(JUST C(ENTER <LEFT, RIGHT, UP, DOWN-ARROWS> <ETX> TO LEAVE

Cette commande permet de déplacer une ligne déjà tapée d'une position vers la gauche (←), vers la droite (→) ou de la recentrer. La ligne peut être déplacée jusqu'à la première colonne vers la gauche ; dans l'autre sens, les caractères non affichables (dépassant ici les 80 colonnes) sont indiqués par un point d'exclamation.

Pour enregistrer définitivement la position de la ligne, vous devrez taper CTRL-C. Esc annulera la modification.

Il est possible de déplacer tout un ensemble de ligne en une seule fois. Pour cela, il suffit d'ajuster une des lignes et de se positionner sur les autres lignes en utilisant CTRL-O (↑) ou CTRL-L (↓). Ces dernières seront alors alignées sur la première.

Si vous avez par exemple le programme suivant :

```
BEGIN;
    IF  A>B
    THEN  WRITE(A)
    ELSE  WRITE(B);
    C=B;
END;
```

et si vous tapez les commandes A←CTRL-L CTRL-L CTRL-C vous obtiendrez le nouveau programme suivant :

```
BEGIN
    IF  A>B
    THEN  WRITE(A)
    ELSE  WRITE(B);
    C:=B;
END;
```

Les trois premières lignes ont été déplacées d'une position vers la gauche.

● **Positionnement de l'environnement**

En utilisant successivement les commandes S(et puis E(NVIRONMENT, vous verrez s'afficher par exemple sur l'écran le message suivant :

```
>ENVIRONMENT: OPTIONS <ETX> OR <SP> TO LEAVE
  A(UTO) INDENT                ajustement automatique (cf. § 2.5.2)
  F(ILLING          FALSE      utilisation de marges (cf. § 2.5.2)
  L(EFT MARGIN      0          marge gauche
  R(IGHT MARGIN     79         marge droite
  P(ARA MARGIN      5          marge paragraphe
  C(OMMAND CH                  marque de début de paragraphe
  T(OKEN DEF        TRUE       recherche des chaînes par mot
                               (cf. § 2.5.4)
  7440 BYTES USED, 12016 AVAILABLE
```

Pour changer une des caractéristiques, il suffit de taper la lettre indiquée suivie de la valeur voulue.

Exemples :

— suppression de l'ajustement automatique des lignes

(SE) AF

— utilisation de marge :

(SE) FT

— marge gauche :

(SE) L 10

● **Positionnement des marges d'un paragraphe**

Quand l'option « utilisation de marges » est vraie (F.True) et l'option « ajustement automatique » fausse (A False), la commande M(argin permet de recadrer le paragraphe où est situé le curseur dans de nouvelles marges.

Un paragraphe est un ensemble de lignes séparé des autres par une ligne blanche, le caractère de commande ou la fin de fichier.

● **Positionnement de marqueurs**

Vous avez la faculté d'introduire jusqu'à dix marqueurs dans un fichier éditable (.TEXT). Ceux-ci pourront être utilisés par les commandes J(UMP et C(OPY.

Pour positionner un marqueur, tapez les commandes S puis M. Le message suivant s'affiche alors :

SET WHAT MARKER?

Cette question vous demande de préciser le nom du marqueur à donner à la position actuelle du curseur. Huit caractères maximum composent ce dernier. Si le nom indiqué correspond à un ancien marqueur, la position précédente est perdue. Si vous essayez d'ajouter un onzième marqueur, le message suivant s'affichera :

```
MARKER OVFLW.WHICH ONE TO REPLACE?
0) nom 0
1) nom 1
:
:
9) nom 9
```

Tapez un numéro et le marqueur correspondant sera remplacé. C'est la seule solution pour éliminer un marqueur.

2.5.6 Sortie de l'éditeur

La commande Q(UIT permet de sortir de l'éditeur. Le message suivant s'affiche alors sur l'écran :

```
>Q(UIT :
  U(PDATE THE WORKFILE AND LEAVE
                    (sauvegarde du texte dans le workfile)
  E(XIT WITHOUT UPDATING
                    (sortie de l'éditeur sans sauvegarde)
  R(ETURN TO THE EDITOR WITHOUT UPDATING
                    (retour à l'éditeur)
  W(RITE TO A FILE NAME AND RETURN
                    (sauvegarde du texte dans un nouveau fichier)
  S(AVE WITH SAME NAME AND RETURN
                    (sauvegarde du texte dans le même fichier)
```

Pour exécuter l'opinion voulue tapez une des lettres V, E, R, W, S.

2.6 L'ASSEMBLEUR 6502

L'assembleur 6502 fourni avec le langage Pascal est un macro-assembleur permettant d'écrire des procédures et des fonctions interfaçables avec un programme Pascal ou Fortran. Il permet en outre de faire de l'assemblage conditionnel.

Les pseudo-instructions utilisables sont les suivantes :

• MACRO : • ENDM	définition de macro-instructions
• IF • ELSE • ENDC	assemblage conditionnel

• PROC • FUNC • END	définition d'une procédure ou d'une fonction	
• ASCII	chaîne de caractères	réservations de zones mémoires et définition d'étiquettes
• BYTE	octet	
• WORD	mot	
• BLOCK	bloc de données	
• ORG	adresse de départ	
• EQU	étiquette	
• ABSOLUTE	assemblage absolu	
• INTERP	assemblage relatif	
• CONST	définition de constantes	
• PUBLIC	variable globale (Pascal et assembleur)	
• PRIVATE	variable locale (assembleur)	
• DEF	variable globale (deux s. p. assembleurs)	
• REF	variable externe (deux s. p. assembleurs)	
• LIST • NOLIST • MACROLIST • NOMACROLIST • PAGE • TITLE • PATCHLIST • NOPATCHLIST	directives de présentation du listing	

Cet assembleur est puissant. Si vous êtes débutant, il vaut mieux que vous commenciez par l'assembleur LISA2.5™ ou par le miniassembleur.

L'emploi de l'assembleur avec le Pascal est utile pour combler les lacunes de ce langage quant à l'accès à la mémoire.

2.7 L'ÉDITEUR DE LIENS

Il permet de regrouper des programmes Pascal, Fortran, et assembleur pour créer des programmes exécutables. De plus, il est possible de créer et d'utiliser des bibliothèques de sous-programmes (APPLESTUFF, TURTLEGRAPHICS, ...).

L'éditeur de liens permet de résoudre les références non définies (PROC, FUNC, GLOBAL, PUBLIC, REF, DEF...) entre les diverses parties du programme. Il crée un fichier en code P interprétable par le système UCSD.

2.8 LANGAGES DISPONIBLES AVEC LA CARTE LANGAGE

2.8.1 Pascal

Le Pascal de l'APPLE II est le Pascal UCSD modifié, pour tenir compte des possibilités graphiques haute résolution. Il est compilé vers du code P accepté par le système UCSD et permet d'exécuter des programmes plus rapidement qu'ils ne le seraient en Basic. Par ailleurs, les possibilités de programmation structurée, de structure de données, en font un langage puissant et agréable à pratiquer.

La bibliothèque de sous-programmes TURTLEGRAPHICS permet un accès facile aux programmes de graphiques couleur haute résolution de l'APPLE II. Les fonctions et les procédures permettent à l'utilisateur de choisir une couleur, de déplacer et faire tourner le curseur, de spécifier les limites des fenêtres de vision, de colorer ces fenêtres, de copier sur l'écran un tableau de données situé en mémoire et d'écrire des caractères en zone graphique. Il ajoute les procédures et fonctions suivantes :

INITTURTLE : passage en mode graphique haute-résolution de tout l'écran avec effacement de l'écran et positionnement de la couleur transparente et du curseur au centre de l'écran

GRAFMODE : passage en mode graphique haute-résolution sans effacement, ni initialisation

TEXTMODE : retour en mode texte

VIEWPORT : définition d'une fenêtre graphique

PENCOLOR : positionnement de la couleur du curseur

FILLSCREEN : positionnement de la couleur de la fenêtre graphique

MOVE TO : déplacement du curseur vers le point indiqué

TURTLEX, TURTLEY : obtention des coordonnées du curseur

TURNTO : rotation du curseur jusqu'à l'angle précisé (absolu)

TURN : rotation du curseur de l'angle précisé (relatif)

TURTLEANG : angle avec l'horizontale « du curseur »

MOVE TO : déplacement du curseur de la longueur indiquée dans la direction fixée par TURN (ou TURN TO)

SCREENBIT (X,Y) : fonction booléenne vraie si le point de coordonnées n'est pas de la couleur noire

DRAWBLOK : tracé d'une figure définie point par point

WCHAR : affichage d'un caractère dans la zone graphique

WSTRING : affichage d'une chaîne de caractères dans la zone graphique

Les possibilités de gestion des entrées-sorties du moniteur sont incluses dans la bibliothèque de sous-programmes appelée APPLESTUFF. Les procédures et fonctions offertes sont les suivantes :

NOTE (fréquence, durée) : émission de sons

KEYPRESS : test de la frappe d'un caractère au clavier

PADDLE (n°) : lecture de la valeur indiquée par une manette de jeux

BUTTON(n°) : test pour savoir si le bouton-poussoir de la manette est pressé

RANDOM, RANDOMIZE : génération de nombres aléatoires

Le Pascal de l'APPLE II permet de découper les programmes en plusieurs fichiers. Vous pouvez, par exemple, séparer en tâches distinctes le développement d'un logiciel évolué, entre plusieurs personnes. Plus la taille des programmes est petite, plus la réalisation et les tests sont aisés et moins vous risquez de « catastrophes » (gestion de fichiers...).

Remarquons que les langages Fortran, Pilot, Logo présentés ci-après ont été écrits en Pascal, ce qui prouve son intérêt.

2.8.2 Fortran

Le Fortran est un langage conçu spécialement pour les travaux de mathématiques, d'engineering et les travaux scientifiques. De nombreuses

bibliothèques de sous-programmes FORTRAN existent et sont adaptables sur l'APPLE II. Le Fortran de l'APPLE est le sous-ensemble ANSIX3-9-1978 du Fortran 77 avec quelques améliorations comme l'accès aux possibilités graphiques haute résolution, aux possibilités sonores et aux manettes de jeux.

Une application sur l'APPLE II peut, de plus, être réalisée en partie en Fortran et en partie en Pascal. On optimise ainsi la vitesse des calculs numériques tout en programmant le reste de l'application de façon structurée.

Le compilateur Fortran traduit les programmes en code P. Vous les exécuterez sur toute machine munie du système UCSD.

Grâce à sa portabilité et à son caractère évolué, le Fortran sera pour vous un facteur important dans la réduction des « coûts » de programmation.

2.8.3 Logo

Logo est un langage évolué, développé par le professeur Seymour Papert du M.I.T., pour apprendre aux enfants la géométrie dans un premier temps et la programmation ensuite, uniquement par amusement. Il offre de nombreuses possibilités graphiques haute résolution semblables à celle de la bibliothèque TURTLEGRAPHICS présentée ci-avant. Le curseur se présente sous forme d'une flèche (→) orientable à volonté dans toutes les directions et pouvant se déplacer entre deux points en traçant ou non la droite suivie. Plusieurs couleurs sont offertes et vous pouvez ainsi dessiner des figures sur l'écran. Les notions de boucles de programmes, puis de sous-programmes apparaissent ensuite pour éviter de saisir toujours les mêmes instructions.

2.8.4 Pilot

Apple Pilot est un système de développement de programmes d'enseignement assisté par ordinateur (EAO). En plus des possibilités du langage « COMMON PILOT », ce langage offre le graphique couleur, les effets sonores et un éditeur de jeux de caractères pour présenter les leçons en mots, dessin et sons.

Le logiciel Apple Pilot a deux modes : Auteur et Leçon. Dans le premier mode, le concepteur crée les leçons et les enregistre sur disquette ; dans le second mode, l'élève suit la leçon du professeur et le système peut enregistrer son niveau, ses temps de réponse, etc...

Citons les commandes « Auteur » suivantes, comme exemple :

créer/éditer des leçons
créer/éditer des graphiques
créer/éditer des bruitages
créer/éditer des caractères définis
sélectionner une diisquette
copier une disquette

Le langage Apple Pilot a été écrit en Pascal ; il ne nécessite pas la carte-langage.

CHAPITRE **3**

La Softcard Z80

3.1 PRÉSENTATION DE LA SOFTCARD

La Softcard Z80 est une carte d'extension développée par la firme américaine Microsoft. Cette carte, contenant un microprocesseur Z80 de Zilog, permet d'étendre les possibilités de l'APPLE II aux logiciels écrits pour le Z80 et notamment au système d'exploitation CP/M et à l'interpréteur MBASIC de Microsoft.

Après une description matérielle de la Softcard, ce chapitre explique la structure du CP/M ainsi que les possibilités du MBASIC et des autres langages offerts.

L'utilisation de la Softcard nécessite 48 K octets de mémoire RAM et au moins un lecteur de minidisquettes. Elle est insérable dans les plots d'extension numéro 1 à 7, elle est compatible avec les cartes d'extension acceptées par la carte langage.

3.2 STRUCTURE MATÉRIELLE DE LA SOFTCARD

La Softcard installée, l'APPLE II peut fonctionner en deux modes distincts :

— le mode 6502 de fonctionnement normal de l'APPLE où la Softcard est déconnectée de l'APPLE (microprocesseur Z80 arrêté par le signal HALT et Softcard séparée du bus de l'APPLE par des buffers trois états en haute indépendance) ;

— le mode Z80 de fonctionnement de la Softcard où le microprocesseur 6502 assure seulement le rafraîchissement des mémoires dynamiques (accès à la mémoire du 6502 et du Z80 alternés).

Le passage d'un mode de fonctionnement à l'autre se fait en adressant la Softcard (adresses correspondant au plot dans lequel est installée la Softcard). Ce passage sert notamment à faire assurer par le 6502 les entrées-sorties alors que le Z80 assure les opérations internes (cf. § 3.3.4 *Appel de sous-programmes* 6502).

La Softcard comprend :

— des buffers trois états permettant de faire communiquer les bus 6502 et Z80,

— la logique nécessaire pour obtenir une horloge pour le Z80 à partir des phases de l'horloge 6502. La fréquence obtenue est de 2 MHz,

— un décaleur d'adresse permettant d'adapter la mémoire de l'APPLE à la structure classique CP/M.

La carte mémoire obtenue est la suivante :

Adresses 6502	Adresses Z80	Contenu
$0800 – $0FFF	0F800 – 0FFFF	contrôleurs d'entrée-sortie
$0400 – $07FF	0F400 – 0F7FF	mémoire d'écran
$0200 – $03FF	0F200 – 0F3FF	buffer E/S et adresses spéciales
$0000 – $01FF	0F000 – 0F1FF	page 0 et page 1 (pile 6502)
$C000 – $CFFF	0E000 – 0EFFF	mapping des entrées-sorties ROM cartes d'extension
$FFFA – $FFF	0DFFA – 0DFF	vecteurs 6502 RESET, BREAK, NMI
$F800 – $FFFF	0D800 – 0DFF9	ROM Autostart
$D000 – $F7FF	0B000 – 0D7FF	ROM APPLESOFT, Basic Entier
$1000 – $BFFF	0000 – 0AFFF	RAM carte langage RAM continue

— La Softcard comprend également la logique nécessaire pour assurer le rafraîchissement des mémoires RAM dynamiques et du microprocesseur 6502. Elle met le 6502 en état d'attente (signal RDY bas) et lui donne le contrôle du bus lorsque le Z80 est en train de décoder l'instruction qu'il va exécuter,

— quatre « switchs » jouant les rôles suivants :

- S1 positionné — pas de décalage d'adresses.
  S1 non positionné — décalage d'adresses.
- S2 positionné — accès direct à la mémoire Z80 autorisé.
- S3 positionné — interruptions non masquables prises en compte.
- S4 positionné — interruptions masquables prises en compte.

Le lecteur intéressé par le fonctionnement de la Softcard se reportera aux manuels techniques des microprocesseurs Rockwell 6502 et Zilog Z80.

3.3 LE SYSTÈME D'EXPLOITATION CP/M

3.3.1 Introduction au CP/M

Le CP/M est un système d'exploitation écrit pour les microprocesseurs Intel 8080 et Zilog Z80. Etant très diffusé, il donne accès à un large éventail de logiciels :

— Logiciel de base :

- éditeur de textes,
- langages évolués : Pascal, Basic, Cobol, Fortran, PL/1, LISP,
- outils d'aide à la mise au point debbuger, désassembleurs...

— Logiciel d'application orienté traitement de texte, ...

Grâce à sa structure interne, les programmes écrits sur une machine sont adaptables sur une autre machine fonctionnant sous CP/M.

3.3.2 Environnement du CP/M sur l'APPLE

Le CP/M est compatible avec toutes les cartes acceptées par la carte langage.

La place est imposée toutefois pour les autres cartes d'extension.

Connecteur 0 : cartes langage, APPLESOFT, Basic Entier.

Connecteur 1 : carte d'interface imprimante (parallèle ou série) (périphérique logique LST:).

Connecteur 2 : carte d'interface d'entrées-sorties généralisées (périphériques logiques RDR: et PUN:) :

— carte d'interface cassette,
— carte d'interface lecteur/perforateur de ruban,

ou toute autre carte d'entrées/sorties.

Connecteur 3 : carte d'interface entrée/clavier (périphérique logique CON:).

Au cas où une carte est présente dans ce Plot, l'écran standard de l'APPLE *n'est plus utilisé.*

Connecteurs 4, 5, 6 : contrôleurs de disques.

4 : disques E et F,
5 : disques C et D,
6 : disques A et B.

Connecteur 7 : carte d'extension de n'importe quel type.

Exemple : carte d'interface pour une télévision couleur.

La carte Z80 peut être mise dans n'importe quel connecteur inoccupé.

Grâce à la structure interne du CP/M, l'utilisateur de la Softcard peut adapter celui-ci à n'importe quelle carte d'extension.

3.3.3 Utilisation du CP/M

- **Généralités**

Le CP/M (Control Program/Microprocessors) est composé de nombreux petits programmes dont la fonction est de gérer les entrées-sorties avec l'utilisateur, d'assurer le fonctionnement des disquettes. Le CP/M offre à l'utilisateur l'accès à de nombreux sous-programmes d'entrées-sorties et facilite ainsi l'écriture de logiciels en langage d'assemblage 8080.

La plupart des commandes du CP/M sont des programmes sur disquettes, accessibles et modifiables par l'utilisateur.

Nous présentons ci-dessous succinctement les commandes du CP/M. Le lecteur intéressé pourra lire, pour en savoir plus, le manuel de référence CP/M.

● **Commandes résidentes en mémoire**

Elles sont au nombre de six. Ce sont les suivantes :

DIR : affichage du contenu d'une disquette.
ERA : destruction d'un fichier.
REN : modification du nom d'un fichier.
SAVE : sauvegarde du contenu d'une zone mémoire dans un fichier.
TYPE : affichage du contenu d'un fichier sur l'écran.
USER : changement du contexte selon le numéro de l'utilisateur.

● **Commandes résidentes sur fichier**

Sur les disquettes fournies avec la Softcard se trouvent les commandes suivantes :

STAT :
— modification des attributs d'un fichier,
— affichage du contenu d'une disquette avec les attributs des fichiers et la place occupée,
— modification des assignations entre périphériques logiques et physiques (cf. § 3.3.2).

ASM : assemblage d'un programme.
LOAD : création d'une commande exécutable.
DDT : mise au point d'un programme.
PIP : manipulations de fichier (copie, concaténation, impression du contenu).
ED : édition de textes.
SUBMIT : exécution d'une commande cataloguée.
DUMP : impression du contenu binaire d'un fichier.

3.3.4 Structure interne du CP/M et adaptation à l'APPLE II

Le CP/M est composé de trois modules :

— le BIOS (Basic I/O System) qui fournit les sous-programmes nécessaires pour gérer physiquement les disquettes et interfacer les périphériques standards (écran, clavier, imprimante, modem, ...),

— le BDOS (Basic Disk Operating System) qui gère logiquement les disquettes et les fichiers,
— le CCP (Console Command Processor) qui est l'interface entre l'utilisateur et le CP/M. Il utilise pour cela le BIOS et le BDOS,
— le TPA (Transcient Program Area) qui est la zone réservée pour les programmes utilisateurs.

Nous donnons ci-dessous les adresses de ces modules dans les deux cas suivants :

— APPLE II 48 K,
— APPLE II 48 K avec carte langage — APPLE II e

APPLE II 48 K		APPLE II 64 K (carte langage)	
(ou CP/M 44 K)		(ou CP/M 56 K)	
0C000H 0A000H	BIOS	0E000H 0D000H	BIOS
	BDOS		BDOS
9C00H 9400H	CCP	0CC00H 0C400H	CCP
100H	TPA (utilisateur)	100H	TPA (utilisateur)

Nous allons étudier maintenant la structure des trois modules BIOS, BDOS et CCP, leur adaptation à l'APPLE et les fonctions CP/M offertes à l'utilisateur.

Les exemples donnés étant en assembleur 8080, il est nécessaire de bien connaître les commandes du CP/M et la programmation assembleur.

● Le BIOS

● *Présentation*

Le BIOS gère les entrées-sorties avec les cartes d'extension classiques de l'APPLE II, avec l'écran et le clavier.

C'est le BIOS qui a été adapté à l'APPLE pour offrir le CP/M à l'utilisateur. Il peut être adapté par l'utilisateur à chaque environnement matériel spécifique et à toutes les cartes d'extension APPLE.

Le BIOS est composé de sous-programmes assembleurs Z80 et 6502. Au chargement du système, il reconnaît les cartes d'extension existantes.

Le BIOS accepte les mêmes cartes d'extension que la carte langage. Ces cartes doivent être dans un connecteur précis :

Connecteur 1	imprimante
Connecteur 2	entrée-sortie série
Connecteur 3	écran-clavier, télétype, carte 80 colonnes
Connecteurs 4, 5, 6	contrôleur de disquette.

● *Adaptation du* BIOS *à un environnement matériel donné*

Le BIOS est adaptable à tout environnement matériel donné en modifiant le bloc de configuration des entrées-sorties (I/O configuration block).

Celui-ci contient les informations suivantes :

— configuration des fonctions graphiques de l'écran,
— reconfiguration de caractères du clavier,
— table des adresses des sous-programmes d'entrée-sortie et sous-programmes considérés,
— table des indicateurs de présence de cartes d'entrées-sorties.

Chaque disquette système contient son propre bloc de configuration des entrées-sorties, qui est chargé et initialisé (indicateurs de présence quand le système est lancé).

Nous allons étudier maintenant les différentes fonctions du bloc de configuration des entrées-sorties.

* FONCTIONS GRAPHIQUES ET ADRESSAGE DU CURSEUR

PRÉSENTATION

La plupart des terminaux vidéo (dont l'écran standard de l'APPLE II) peuvent supporter des fonctions spéciales telles que l'adressage direct du curseur, l'effacement de l'écran, de la ligne courante ou le passage en mode inverse de l'écran (lettres noires sur fond blanc). Ces fonctions sont réalisées en envoyant une certaine séquence de caractères sur l'écran.

Les problèmes de compatibilité entre les différents types de terminaux ont conduit à créer la table de transcodage suivante :

Software (écran émulé)/Hardware (écran de l'APPLE)
(terminal Soroc/terminal Datamedia généralement).

Les neuf fonctions suivantes sont acceptées par le BIOS :

1. effacement de l'écran,
2. effacement jusqu'à la fin de l'écran,
3. effacement jusqu'à la fin de la ligne,
4. passage en mode normal,
5. passage en mode inverse,
6. positionnement du curseur en haut et à gauche,
7. adressage du curseur,
8. remontée du curseur d'une ligne,
9. décalage du curseur d'une position vers la droite.
10. descente du curseur d'une ligne,
11. décalage du curseur d'une position vers la gauche.

Ces deux dernières fonctions sont assurées de façon standard en envoyant sur l'écran les caractères suivants :

— Line feed (LF code ASCII 10) pour la fonction 10
— Backspace (BS code ASCII 8) pour la fonction 11.

Pour réaliser ces fonctions graphiques, on dispose de deux types de séquences à envoyer sur l'écran :

— un caractère unique,
— un caractère ASCII, précédé par un caractère d'introduction.

Le format de la table de transcodage Software/Hardware est le suivant :

N° de fonction	Adresse du code dans la table relative au Logiciel	Matériel	Description
	0F396H	0F3A1H	Valeur à rajouter aux coordonnées X, Y pour positionner le curseur (intervalle 0-127). Si le premier bit de gauche est 0, les coordonnées doivent être envoyées dans l'ordre Y, X. Si le premier bit est 1 l'ordre est X,Y
	0F397H	0F3A2H	Caractère d'introduction (0 si aucune fonction ne le nécessite)
1	0F398H	0F3A3H	Effacement de l'écran
2	0F399H	0F3A4H	Effacement de la fin de l'écran
3	0F39AH	0F3A5H	Effacement de la fin de la ligne
4	0F39BH	0F3A6H	Passage de l'écran en mode normal
5	0F39CH	0F3A7H	Passage de l'écran en mode inverse
6	0F39DH	0F3A8H	Positionnement du curseur en haut et à gauche de l'écran
7	0F39EH	0F3A9H	Adressage du curseur
8	0F39FH	0F3AAH	Remontée d'une ligne
9	0F3A0H	0F3ABH	Déplacement du curseur d'une position vers la droite

Si une des cases de la table est à zéro, la fonction correspondante n'est plus disponible. Le caractère d'introduction est à envoyer sur l'écran si le bit de poids fort de la case est à 1.

L'écran standard d'APPLE apporte ces neuf fonctions.

EXEMPLE D'UTILISATION

Utiliser la table des fonctions graphiques permet d'obtenir des programmes indépendants du matriel sur lequel on travaille.

Deux positions sont adoptables :

— Soit on décide d'utiliser un écran donné et on n'emploie la partie logiciel de la table que pour interfacer les programmes de l'APPLE.

La plupart des logiciels écrits sous CP/M sur d'autres machines utilisent un terminal Soroc. En remplisant la partie logiciel de la table avec les caractéristiques du terminal Soroc et la partie matériel avec les caractéristiques du terminal utilisé ou de l'écran APPLE, l'adaptation des logiciels CP/M sur APPLE est grandement facilitée.

— Soit on décide de ne pas connaître l'écran considéré et on utilise les fonctions graphiques en lisant le contenu de la table pour savoir quels caractères envoyer sur l'écran.

MODIFICATION DE LA TABLE DES FONCTIONS GRAPHIQUES

La modification de la table de transcodage est faite par le programme CONFIGIO.BAS, en répondant 1 au menu proposé. Le contenu de la table s'affiche alors, suivi d'un nouveau menu.

```
+ TERMINAL SCREEN FUNCTION DEFINITION +

FUNCTION            SOFTWARE        HARDWARE

CLEAR SCREEN        ESC *           FF
CLR TO EOS          ESC Y           VT
CLR TO EOL          ESC T           GS
LO-LITE TEXT        ESC )           SO
HI-LITE TEXT        ESC (           SI
HOME CURSOR         RS              EM
ADDRESS CURSOR      ESC =           RS
XY COORD OFFST      32              32
XY XMIT ORDER       YX              YX
CURSOR UP           VT              US
CURSOR FORWARD      FF              FS

      1. SOROC IQ 120/IQ 140
      2. HAZELTINE 1500/1510
      3. DATAMEDIA
      4. OTHER
      Q. QUIT

      SELECT –
```

Trois types de terminaux (Soroc, Hazeltine, Datamédia) sont proposés. Vous pouvez les choisir en répondant respectivement 1, 2, 3. Une quatrième option (4) est possible dans laquelle vous pouvez définir séparément chaque fonction.

```
      + + SCREEN FUNCTION DEFINITION + +

1 - LEAD-IN CHARACTER
2 - CLEAR SCREEN
3 - CLR TO EOS
4 - CLR TO EOL
5 - LO-LITE TEXT
6 - HI-LITE TEXT
7 - HOME CURSOR
8 - ADDRESS CURSOR          -19
9 - CURSOR UP
10 - CURSOR FORWARD
Q - QUIT

SELECT –
```

Une fois la fonction choisie, le programme vous demande le caractère à entrer dans la table et diverses réponses relatives au numéro de la fonction.

Lorsque vous avez redéfini votre table, il faut la stocker sur la disquette en tapant Q pour retourner au menu principal, puis 4 pour entériner vos modifications.

* REDÉFINITION DE CARACTÈRES DU CLAVIER

PRÉSENTATION

Certains logiciels CP/M nécessitent des caractères spéciaux, généralement accessibles sur les claviers.

Le clavier de l'APPLE II est très déficient dans ce domaine. Ce problème est résolu par la table de reconfiguration des caractères.

La reconfiguration des caractères est faite par la fonction CP/M n° 1 (cf. § 3.3.4).

La table est située à l'adresse 0F3ACH et permet de reconfigurer 6 caractères. Elle est structurée comme suit :

code ASCII du caractère	code ASCII du caractère reconfiguré

Tous les caractères de la table ont leur bit de poids fort à 0 excepté le dernier d'entre eux.

MODIFICATION DE LA TABLE

La modification de la table se fait en utilisant le programme CONFIGIO.BAS et en répondant 2 au menu proposé. Sur l'écran s'affichent alors les lignes suivantes :

```
++ KEYBOARD CHARACTER DEFINITION ++
Ctrl-K→
Ctrl-A→RUB
Ctrl-B→
Ctrl-U→CTRL-I
ADD/DELETE/QUIT(A/D/Q)
```

Pour ajouter un caractère, taper A.
Pour supprimer un caractère, taper D.
Pour retourner au menu général, taper Q.

Si l'on a tapé A, le programme affiche :

```
CHAR
```

Il faut taper le caractère à modifier. C'est possible aux formats suivants :

— caractère,
— nom d'un caractère ASCII (exemple NUL, BEL, LF, CR),
— CTRL-caractère,
— LC-caractère (LC signifie lower case, c'est-à-dire minuscule),
— code ASCII hexadécimal (précédé de BH).

Le lecteur intéressé se reportera en annexe pour connaître les noms des caractères et les codes.

Lorsque le caractère à modifier a été reconnu par le programme, celui-ci affiche une flèche → et l'utilisateur donne le nouveau caractère selon un des formats ci-dessus.

Le principe est le même pour la suppression de la redéfinition d'un caractère.

Il est nécessaire, les modifications terminées, de les recopier sur la disquette. Pour cela, on répondra 4 *au menu principal et le bloc de configuration des entrées-sorties sera recopié sur la disquette système* CP/M.

* ADAPTATION DES CARTES NON STANDARD D'EXTENSION

PRÉSENTATION

L'utilisateur peut adapter le CP/M à des cartes d'extension non standard et à des logiciels d'entrée-sortie. Tous les sous-programmes d'entrée-sortie sont vectorisés dans une table contenue dans le bloc de configuration des entrées-sorties. Cette table donne normalement les adresses des sous-programmes d'entrée-sortie standard du BIOS, mais l'utilisateur peut la changer pour la faire pointer sur ses programmes.

Pour la réalisation d'une fonction graphique, le registre B *contient le numéro de la fonction lors de l'appel du sous-programme d'affichage sur console* (4, 5).

La structure de la table est la suivante :

N° vecteurs	Adresse du vecteur	Nom	Description	Valeur standard
1	0F380H	Etat de la console	Renvoie 0FFH dans le registre A si un caractère a été tapé, 00H sinon	0F34AH
2 3	0F382H 0F384H	Lecture de la console 1 2	Lecture d'un caractère de la console et rangement dans le registre A avec le bit de poids fort à zéro	0F358H 0AB12H
4 5	0F386H 0F388H	Affichage sur la console 1 2	Affichage du caractère dont le code ASCII est contenu dans le registre C	0F35EH 0AC3EH
6 7	0F38AH 0F38CH	Lecture d'une cassette (périphérique logique RDR:) 1 2	Lecture d'un caractère sur cassette (entrée généralisée), et rangement dans le registre A	AD45H AD45H
8 9	0F38EH 0F390H	Ecriture sur cassette (PUN:) 1 2	Envoi du caractère contenu dans C vers la cassette (sortie généralisée)	AD3FH AD3FH
10 11	0F392H 0F394H	Impression d'un caractère (LST:) 1 2	Envoi du caractère contenu dans C vers l'imprimante	AD2BH

Trois zones mémoires de 128 octets sont réservées pour les drivers (sous-programmes gérant les entrées-sorties) de l'utilisateur.

0F200H – 0F27FH Plot 1 périphérique logique LST:
0F280H – 0F2FFH Plot 2 périphériques logiques PUN:
RDR:
0F300H – 0F37FH Plot 3 périphériques logiques CRT:
TTY:

La plupart des cartes d'entrées-sorties ont un driver 6502. L'interfaçage de ces cartes se fera en appelant ces drivers (cf. *infra*) et en inscrivant l'adresse du sous-programme appelant le driver 6502 dans la table décrite ci-dessus.

* APPEL DE SOUS-PROGRAMMES 6502

Il est possible d'appeler des sous-programmes assembleurs 6502. En effet il est facile de passer du mode de fonctionnement Z80 au mode de fonctionnement 6502 et réciproquement.

Pour appeler un sous-programme 6502, l'utilisateur doit procéder ainsi :

— positionner l'adresse du sous-programme à appeler sur l'adresse 0F3D0H selon l'ordre du Zilog (c'est-à-dire : octet de poids faible, puis octet de poids fort) ;

— positionner les arguments à passer aux adresses suivantes :

Adresses Z80	Adresses 6502	
0F045H	$45	registre A 6502
0F046H	$46	registre X 6502
0F047H	$47	registre Y 6502
0F048H	$48	registre P 6502 (états)
0F049H	$49	registre S 6502

Cette zone de passage des paramètres est celle utilisée par les sous-programmes IOSAVE *et* IOREST *du moniteur de l'*APPLE.

— *Ecriture à l'adresse de la Softcard.*

L'adresse de la carte est 0EN00H, *où* N *est le numéro du plot dans lequel est placée la Softcard.*

Cette adresse est reconnue par le CP/M *au chargement du système et elle est recopiée à l'adresse* 0F3DEH.

L'appel se fera par la séquence suivante :

```
LHLD  0F3DEH
MOV   M, A ; écriture à l'adresse pointée par 0F3DEH
```

Cette possibilité d'appeler des sous-programmes 6502 sert surtout à interfacer de nouvelles cartes d'extension en appelant les sous-programmes en ROM des cartes.

L'utilisation de la mémoire présente sur la carte langage est différente selon le mode de fonctionnement de la carte Z80.

En mode Z80 (cf. § 3.2) la RAM de la carte langage est accessible en lecture et en écriture et correspond aux adresses $D000-$FFFF du 6502. En mode 6502, la ROM de la carte centrale de l'APPLE est sélectionnée en lecture et la RAM de la carte langage est inaccessible en lecture. Par contre, toute écriture aux adresses citées sera réalisée dans la RAM.

* INDICATION DE PRÉSENCE DES CARTES D'EXTENSION

Au chargement, le CP/M teste chaque connecteur pour voir s'il contient une carte d'extension. Le résultat obtenu est le suivant :

0	pas de carte
1	carte inconnue
2	contrôleur de disquettes
3	APPLE Communications Interface CCS 7710A Serial Interface (Interface série)
4	APPLE High Speed Serial Interface Videx-Videotherm APPLE Silentype Printer Interface
5	APPLE Parallel Printer Interface (imprimante interface parallèle)

Ces résultats sont réunis dans une table, à l'adresse 0F3B9H. La valeur obtenue pour un plot S est placée à l'adresse 0F3B8H + S.

Le nombre de contrôleurs de disquettes est, de plus, placé à l'adresse 0F3B8H.

● Le BDOS

Le BDOS est, nous l'avons dit, la partie du CP/M chargée de la gestion logique des fichiers et des disquettes.

Un fichier est géré par des f.c.b. (file control block ou bloc de contrôle des fichiers). Chaque f.c.b. décrit 16 K octets d'un fichier, c'est-à-dire 128 secteurs. Un fichier peut utiliser 512 f.c.b. soit une taille maximale logique de 8 mégaoctets (la taille maximale physique est celle d'une disquette).

Un fichier est constitué d'une séquence de caractères ASCII. Chaque ligne est terminée par la séquence retour chariot 0DH et line feed 0AH. Un secteur de 128 octets peut donc contenir plusieurs lignes de textes. La fin du fichier est marquée par un CTRL-Z (1AH) ou une fin de fichier physique.

Lors de l'ouverture d'un fichier, l'utilisateur doit fournir l'adresse d'un f.c.b. qui aura été partiellement rempli (nom du fichier). Le CP/M consultera alors le catalogue de la disquette et initialisera le reste des données du f.c.b. A la fermeture du fichier, le f.c.b. en mémoire sera recopié dans le catalogue.

Une zone est réservée en mémoire par déffaut pour deux f.c.b. à l'adresse 005CH.

005CH nom du premier fichier
006CH nom du second fichier .

Il appartient à l'utilisateur de recopier le nom d'un deuxième fichier dans un autre f.c.b. avant l'ouverture d'un des fichiers.

La structure d'un f.c.b. est la suivante :

dr	f1 f2 f8	t1 t2 t3	ex	s1 s2	rc	d0 d15	cr r0 r1 r2
00	01 02...08	09 10 11	12	13 14	15	16... 31	32 33 34 35

avec

dr = numéro du drive
0 : drive actif par défaut
1 : A
2 : B
:
16 : P

f1...f8	nom du fichier (avec bit de poids fort à 0 0 des fi)
t1 t2 t3	type du fichier t1 = 1 fichier accessible en lecture seulement avec bit de t2 = 1 fichier SYS n'est pas listé par DIR poids fort
ex	numéro du bloc de 16 K actif
s1 s2	paramètres internes
rc	numéro du secteur dans le bloc ex
d0...d15	paramètres internes
cr	prochain secteur à lire ou écrire dans un fichier séquentiel initialisé par l'utilisateur
r0, r1, r3	numéro du secteur auquel on va accéder en accès direct.

Une zone de 128 octets est réservée en mémoire pour les entrées-sorties sur disque. L'adresse de cette zone est 0080H par défaut. Elle est modifiable par l'utilisateur en utilisant une fonction du BDOS (cf. § 3.3.4 Fonctions CP/M).

Le BDOS utilisant le BIOS pour les entrées-sorties physiques sur disquette n'a pas été modifié pour implanter le CP/M sur l'APPLE II.

● Le CCP

Le CCP est la partie du CP/M qui dialogue avec l'usager. Lorsqu'une commande est tapée sur le clavier, le CCP analyse la ligne tapée :

— il recopie la partie de la ligne à analyser par la commande à l'adresse 080H,
— il crée les f.c.b. (cf. BDOS) relatifs aux fichiers à manipuler par la commande à l'adresse 05CH,
— il analyse la commande tapée, teste s'il s'agit d'une commande interne (DIR, SAVE, TYPE, USER, REN, ERA) ou d'une commande externe à charger. Dans le premier cas, il appelle le sous-programme correspondant, dans le second cas, il consulte le « directory » (catalogue) de la disquette et charge la commande en mémoire à l'adresse 100H. Il rend la main à l'utilisateur à la fin de la commande.

Par exemple, si l'on tape la commande suivante :

```
A>MBASIC A:JEU.BAS
```

le CCP va faire le traitement suivant :

1) Initialisations

— à l'adresse 005CH :

```
01  4A  45  55    42  41  53
    J   E   U     B   A   S
N° drive (A:)
```

— à l'adresse 0080H :

```
A:J E U.B A S
```

2) Chargement de la commande MBASIC.COM et lancement de celle-ci.

Le CCP est composé globalement des parties suivantes :

- sous-programmes standard d'accès aux fonctions CP/M,
- sous-programmes d'analyse des caractères dans un buffer,

- sous-programmes d'analyse des commandes tapées au clavier,
- sous-programmes de chargement et de lancement des commandes externes,
- sous-programmes correspondant aux six commandes internes du CP/M,
- programme d'enchaînement des différentes parties du CCP.

Le CCP utilisant les commandes du BIOS et du BDOS (fonctions CP/M) est indépendant de la configuration matérielle. Il n'a pas été modifié pour adapter le CP/M sur l'APPLE II.

● Les fonctions CP/M offertes à l'utilisateur

De nombreux sous-programmes du CP/M sont accessibles par les programmes en langage d'assemblage de l'utilisateur. Ils se classent en deux catégories :

— ceux du BIOS,
— ceux du BDOS.

Dans le BIOS sont accessibles notamment les sous-programmes suivants :

— affichage d'un caractère,
— saisie d'un caractère,
— lecture/écriture d'une cassette,
— impression d'un caractère sur imprimante,
— affichage/saisie d'un buffer,
— lecture de l'état de la console (écran/clavier).

Sous-programmes accessibles dans le BDOS :

— création, ouverture, fermeture, destruction d'un fichier,
— lecture/écriture des fichiers séquentiels et à accès direct,
— modification des caractéristiques d'un fichier.

L'accès à ces sous-programmes se fait de manière standardisée :

— chargement dans le registre C du code de la fonction,
— appel du sous-programme situé à l'adresse 005H.

Voici la liste des fonctions accessibles avec leur numéro d'ordre, les paramètres d'appel et de sortie, et une sémantique succincte.

Numéro de la fonction	Paramètres d'appel	Paramètres de retour	Sémantique
0	C=00H		Relance du système
1	C=01H	A=code ASCII	Saisie d'un caractère (reste bloquée tant qu'un caractère n'est pas tapé sur le clavier) et reconnaissance des caractères de contrôle suivants : CTRL-I(-Tab), CTRL-S, CTRL-P, CTRL-H (Backspace), LF (line feed), CR (carriage return) (périphérique CON:) cf. § 3.3.2
2	C=02H E=Code ASCII du caractère à afficher		Affichage d'un caractère (comme ci-dessus CTRL-I (Tab), CTRL-P, CTRL-S... sont traités) (périphérique CON:) cf. § 3.3.2
3	C=03H	A=code ASCII du caractère lu sur la cassette ou le ruban	Lecture d'un caractère sur cassette ou ruban. Reste bloquée tant qu'un caractère n'a pas été lu (périphérique RDR:) cf. § 3.3.2
4	C=04H E=Code ASCII du caractère à écrire		Ecriture d'un caractère sur cassette ou ruban (périphérique PUN:) cf. § 3.3.2
5	C=05H E=Code ASCII du caractère à imprimer		Impression d'un caractère sur imprimante (périphérique LST:) cf. § 3.3.2
6	C=06H E=0FFH (entrée) ou Code ASCII (sortie)	A=code ASCII ou status	Entrée-sortie directe (les contrôles classiques CP/M ne sont plus faits)
7	C=07H	A=octet d'E/S	Lecture de l'octet d'entrée-sortie (cf. § 3.3.4) (IOBYTE)
8	C=08H E=octet d'E/S		Ecriture de l'octet d'entrée-sortie (cf. § 3.3.4) (IOBYTE)
9	C=09H DE=adresse chaîne de caractères		Affichage d'une chaîne de caractères terminée par $ (les contrôles CP/M de la fonction 2 sont réalisés)
10	C=0AH DE=adresse du buffer	Caractères saisis	Lecture d'une ligne de la console Au retour de la fonction, le buffer du CP/M présente le format suivant : mx nc cl... cn... mx : nombre maximum de saisies nc : nombre de caractères saisis

Numéro de la fonction	Paramètres d'appel	Paramètres de retour	Sémantique
			les fonctions suivantes sont reconnues : rub/del : supprime l'écho du dernier caractère CTRL-C : rechargement du système si début de ligne CTRL-E : fin de ligne physique CTRL-H : backspace CTRL-J : line feed } fin de saisie CTRL-M : return } CTRL-R : recopie de la ligne courante CTRL-X : retour au début de la ligne
11	C=0BH	A=état de la console	A=0FFH si un caractère a été tapé 00H sinon
12	C=0CH	HL=numéro de version H=CP/M ou MP/M, L=version	Lecture du numéro de version
13	C=0DH		Réinitialisation des disques : — placer tous les disques sans protection écriture, — repositionner l'adresse DMA à 80H
14	C=0EH E=numéro du disque 0 A: 1 B: . . . 15 P:		Sélection du disque actif
15	C=0FH DE=adresse du f.c.b.	A=code du directory	Ouverture d'un fichier : le BDOS recherche un fichier dont le nom correspond à celui donné dans le f.c.b. Si aucun fichier n'a été trouvé, A=255, si un fichier a été trouvé, A=0 à 3 et le f.c.b. est chargé en mémoire
16	C=10H DE=adresse du f.c.b.	A=code du directory	Fermeture d'un fichier : Le BDOS recherche un fichier dont le nom correspond avec celui donné dans le f.c.b. Si aucun fichier n'a été trouvé A=255 Si un fichier a été trouvé le f.c.b. est recopié dans le catalogue et A = 0, 1, 2, 3. Les modifications du fichier sont enregistrées définitivement sur la disquette

Numéro de la fonction	Paramètres d'appel	Paramètres de retour	Sémantique
17	C = 11H DE = adresse du f.c.b.	A = code du directory	Recherche de la première occurrence d'un fichier dans le directory. Si aucun fichier n'a été trouvé A = 255. Si une occurrence du fichier a été trouvée, le secteur dans lequel est situé le fichier dans le directory est recopié dans le buffer commençant à l'adresse DMA et l'entrée du fichier est située à l'adresse relative A*32 où A = 0, 1, 2, 3
18	C = 12H	A = code du directory	Recherche de la prochaine occurrence d'un fichier dans le directory (cf. Fonction 17)
19	C = 13H DE = adresse du f.c.b.	A = code du directory	Destruction d'un fichier • = 255 si aucun fichier n'a été détruit Sinon, A = 0, 1, 2, 3
20	C = 14H DE = adresse du f.c.b.	A = code du directory A = 00H OK < > 00H fin de secteur	Lecture d'un fichier séquentiel : lecture du secteur numéro "cr" du bloc du fichier. Le numéro du secteur doit avoir été initialisé par l'utilisateur après l'ouverture du fichier. Le numéro de secteur cr est incrémenté d'une unité et le passage du bloc "ex" courant au bloc suivant est fait automatiquement. Le secteur lu est recopié dans le buffer d'adresse DMA
21	C = 15H DE = adresse du f.c.b.	A = code du directory A = 00H OK A< >00H écriture impossible (disquette pleine)	Ecriture d'un fichier séquentiel : Le buffer situé à l'adresse DMA est recopié dans le secteur "cr". La gestion de "cr" est la même que pour la fonction 20
22	C = 16H DE = adresse du f.c.b.	A = code du directory A = 0 1 2 3 0K = 255 directory plein	Création d'un fichier : Le BDOS crée le fichier caractérisé par le f.c.b. C'est à l'utilisateur de vérifier que deux fichiers ne porteront pas le même nom
23	C = 17H DE = adresse du f.c.b.	A = code du directory = 0 1 2 3 0K = 255 fichier non trouvé	Changement du nom d'un fichier
24	C = 18H	HL = vecteur logique des disques A = L (compatibilité avec les anciennes versions)	Lecture du vecteur logique des disques H L 15 8 | 7 0 le bit n° i est à 1 si le drive correspondant est actif

Numéro de la fonction	Paramètres d'appel	Paramètres de retour	Sémantique
25	C = 19H	A = n disque actif 0 → A : . . . 15 → P :	Lecture du numéro du disque actif
26	C = 1AH DE = adresse DMA		Positionnement de l'adresse DMA : adresse de base d'un buffer servant aux entrées-sorties. Elle est initialisée par défaut à 80H
27	C = 1BH	HL = adresse du vecteur d'allocation	Un vecteur d'allocation existe en mémoire pour chaque disquette en ligne. Cette fonction fournit l'adresse du vecteur d'allocation de la disquette active
28	C = 1CH		Passage de la disquette en protection écriture jusqu'au prochain rechargement (CTRL-C). Un essai d'écriture sur la disquette produira le message Bdos Err on d : R/O
29	C = 1DH	HL = vecteur lu	Acquisition du vecteur indiquant les disquettes accessibles seulement en lecture (protection en écriture). Le bit i est à 1 si le drive correspondant est protégé en écriture HL 15 . . . 0 P : A :
30	C = 1EH DE = adresse du f.c.b.	A = code du catalogue	Positionnement des attributs d'un fichier (i.e. bits t'_1, t'_2 du f.c.b. — cf. § 3.3.4 Le BDØS)
31	C = 1FH	HL = adresse du bloc des paramètres de la disquette	Lecture de l'adresse du bloc des paramètres de la disquette (place libre, ...)
32	C = 20H E = 0FFH (lecture) ou E = numéro de l'utilisateur	A = valeur lue	Si E = 0FFH, lecture du numéro d'utilisateur courant (cf. Fonction USER), sinon positionnement du numéro
33	C = 21H DE = adresse du f.c.b.	A = code de retour 00 ØK 01 pas de données écrites 03, 04, 06 } autres erreurs	Lecture d'un fichier à accès direct. Le CP/M lit l'article indiqué par r_0, r_1, le recopie dans le buffer situé à l'adresse DMA et positionne le code retour

Numéro de la fonction	Paramètres d'appel	Paramètres de retour	Sémantique
34	C = 22H DE = adresse f.c.b.	A = code de retour	Ecriture d'un fichier à accès direct. Le CP/M recopie les données à l'adresse DMA dans l'article indiqué par r_0 r_1 et positionne le code retour
35	C = 23H DE = adresse f.c.b.	r_0, r_1 = longueur	Calcul de la longueur du fichier
36	C = 24H DE = adresse f.c.b.	r_0, r_1 = prochain article à accéder	Positionnement du numéro du prochain article à lire/écrire (utile après des accès séquentiels par exemple)

Nous avons indiqué des fonctions qui sont utilisées de façon interne par le CP/M (*n°* 27, 31 *par exemple*). *Le lecteur voulant en savoir plus pourra se reporter au manuel d'adaptation du* CP/M (« CP/M *altération guide* »).

3.4 LES BASIC MICROSOFT

3.4.1 Présentation générale

Le Basic Microsoft est un Basic très puissant et très répandu. La version livrée avec la Softcard est la version 5.0. Il offre aux utilisateurs de l'APPLE des instructions telles que PRINT USING, CHAIN, WHILE/WEND, l'accès à des chiffres en double précision et à des manipulations de fichiers très commodes. La version APPLE incorpore les possibilités graphiques et sonores de l'APPLE.

A cause des tailles mémoires nécessaires pour le graphisme en haute résolution, deux versions sont situées sur les disquettes fournies :

— MBASIC qui est la version basse résolution et qui occupe 24 K octets de mémoire.

— GBASIC qui est la version haute résolution et qui occupe 30 K octets de mémoire.

Nous allons étudier dans ce paragraphe les différences entre le Basic Microsoft et l'APPLESOFT en approfondissant la gestion des fichiers.

3.4.2 Comparaison APPLESOFT-Basic Microsoft

● **Possibilités du Basic Microsoft, non offertes par APPLESOFT**

Le Basic Microsoft offre les possibilités suivantes :

— Instructions supplémentaires :

CHAIN/COMMON	Passage de variables à un autre programme Basic
CALL	Appel de sous-programmes assembleurs Z80 et 6502
PRINT USING	Sortie de résultats selon un format déterminé
PUT#, GET#, FIELD#, WRITE#, INPUT# PRINT#, PRINT# USING	gestion de fichiers (cf. 3.4.3).
SWAP	Echange le contenu de deux variables
WHILE/WEND	Exécution d'une boucle tant qu'une condition est vraie
IF..THEN..ELSE	Possibilité de traitement si la condition n'est pas vérifiée
ERASE	Suppression en mémoire du tableau indiqué.

— Commandes du Basic :

AUTO et RENUM	Génération de numéros de lignes et renumérotation des lignes
EDIT	Editeur de lignes

— Types de variables :

* Entier, réels simple précision (7 chiffres).
* Réels double précision (16 chiffres).
* Noms des variables sur 40 caractères au lieu de deux.

— Fonctions de chaînes de caractères :

Assignation de MID$, INSTR, STRING$, HEX$ OCT$

— Opérateurs logiques :

XOR, IMP, EQV, division entière, MOD (reste)

— Définition de fonctions à multiples arguments :

— Fonctions graphiques et sonores supplémentaires :

BUTTON(i)	(i n° manette de jeu). Permet de savoir si le bouton de la manette numéro i est appuyé
BEEP(i, j)	Génère un son de tonalité i et de durée j
HSCRN(x, y)	Permet de savoir si le point (x, y) est alllumé sur l'écran
VPOS(o)	Retourne la position verticale du cursseur.

● **Instructions utilisées différemment en Basic Microsoft et en APPLESOFT**

Ce sont les suivantes :

	* FOR... NEXT
Format APPLESOFT	On passe toujours une fois dans la boucle. Le test est fait en fin de boucle
Format MBASIC	Le test est fait en début de boucle. Il est possible de ne pas passer dans la boucle
	* INPUT
Format APPLESOFT	INPUT string ; <liste de variables>
Format MBASIC	INPUT ; string <liste de variables> : le premier ; supprime le passage à la ligne après l'INPUT, la virgule supprime le point d'interrogation
	* ON ERROR GOTO
	Le passage du code de l'erreur est différent
Format APPLESOFT	Passage à l'adresse 222
Format MBASIC	Utilisation des variables ERR (erreur) et ERL (ligne)
	* RESUME
Format APPLESOFT	L'exécution du programme reprend à l'instruction qui a déclenché l'erreur
Format MBASIC	L'exécution du programme reprend à la ligne précisée dans l'instruction RESUME ou à celle qui a déclenché l'erreur

	* TEXT
Format MBASIC	Efface l'écran si l'on revient du mode graphique basse résolution
	* GR
Format APPLESOFT	Passage en mode graphique basse résolution avec quatre lignes de textes, effacement de l'écran et positionnement de la couleur à 0
Format MBASIC	Passage en mode graphique basse résolution avec ou non 4 lignes de textes, remplissage de l'écran avec la couleur désirée
	* HGR, HGR2
Format APPLESOFT	Passage en mode graphique haute résolution avec 4 lignes de textes
Format MBASIC	Passage en mode graphique haute résolution avec choix du format de l'écran de la couleur et de l'effacement ou non de l'écran, en page 1
	* IF THEN ELSE
	Ajout du ELSE
	* CALL
Format APPLESOFT	Appel sans paramètres de S.p. 6502.
Format MBASIC	Appel de S.p.Z80 avec passage de paramètres 6502 (cf. § 3.4.5)

● **Possibilités de l'APPLESOFT non offertes par le Basic Microsoft**

Ce sont :

DRAW	Affichage du dessin sur l'écran à partir d'un point.
XDRAW	Même chose avec la couleur inverse.
SCALE	Modification de l'échelle d'un dessin.
ROT	Rotation d'un dessin.
FLASH	Passage de l'écran en mode clignotant.
SHLOAD	Chargement d'un dessin en mémoire.

C'est-à-dire qu'il manque surtout la possibilité d'utiliser des dessins prédéfinis.

3.4.3 Fichiers en MBASIC (ou GBASIC)

● **Introduction**

La gestion de fichiers est le meilleur argument en faveur du MBASIC par rapport à l'APPLESOFT et au DOS. Nous pouvons dire que le MBASIC est plus compliqué à écrire comme logiciel de base, mais nettement plus puissant alors que la formule APPLESOFT-DOS est plus facile à écrire mais nettement moins puissante. Nous recommandons fortement aux utilisateurs voulant réaliser des applications de gestion professionnelles de s'offrir la Softcard. Ils économiseront du temps de travail et leur investissement sera très rapidement amorti.

Il existe deux types de fichiers de données avec le MBASIC :

— les fichiers séquentiels,
— les fichiers à accès direct.

La société américaine KISS propose de plus une version de l'interpréteur MBASIC permettant la gestion de fichiers à accès séquentiel indexé.

Nous allons étudier dans ce paragraphe les deux types de fichiers MBASIC.

● **Les fichiers à Accès Séquentiel**

Les instructions et fonctions offertes sont :

OPEN	"O", N° de fichier, "nom de fichier" "I" O output = écriture de données I input = lecture de données
CLOSE	N° de fichier
INPUT	N° de fichier, liste de variables Si un chiffre doit être lu, MBASIC ignore les blancs, CR, LF, et considère que le nombre commence au premier caractère, différent de ceux indiqués ci-dessus, et finit au premier blanc suivant Si une chaîne est à lire, MBASIC ignore les mêmes caractères que pour les chiffres. Il considère le premier caractère comme début de la chaîne de caractères

Deux cas se présentent :

— si le caractère est ", tous les caractères lus, jusqu'à la prochaine occurrence de " sont rangés dans la chaîne,
— sinon la fin de la chaîne sera marquée par une virgule CR ou LF.

WRITE	Ecriture de données dans un fichier séquentiel. Les données sont séparées physiquement par des virgules dans le fichier et les chaînes de caractères sont encadrées par "
PRINT	Ecriture de données dans un fichier séquentiel. C'est à l'utilisateur d'insérer les séparateurs voulus
PRINT USING	Même instruction que PRINT avec formatage des données
LINE INPUT	Lecture d'une ligne à la fois
EOF	Fonction indiquant si l'on est en fin de fichier ou non
LOC	Fonction indiquant le numéro du prochain article à lire ou écrire

Pour écrire des données dans un fichier séquentiel, vous devrez suivre le scénario suivant :

1) OPEN "O", N° de fichier, "nom de fichier"
2) WRITE
PRINT
PRINT USING — données
3) CLOSE, N° de fichier

Pour lire des données, vous devrez faire :

1) OPEN "I", N° de fichier, "nom de fichier"
2) INPUT
LINE INPUT — données
3) CLOSE, N° de fichier

Pour ajouter des données dans un fichier séquentiel, il n'existe pas de commande équivalente à APPEND du DOS, mais celle-ci n'étant pas d'une fialibilité à toute épreuve, il est préférable dans les deux cas de recopier le fichier pour lui ajouter des données en queue.

● **Les fichiers à Accès Direct**

Les fichiers à Accès Direct occupent moins de place sur disquette car les données sont stockées avec un code binaire condensé. Il est possible d'accéder aux données des enregistrements, sans avoir à parcourir tout le fichier.

Les instructions et fonctions offertes à l'utilisateur sont les suivantes :

OPEN	"R", n° de fichier, "nom de fichier", longueur articles
CLOSE	N° de fichier
FIELD	N° de fichier, largeur AS chaîne... Cette instruction définit la décomposition du buffer d'entrée/sortie en variables. Vous devrez remplir ce buffer avant d'écrire dans le fichier et lire ce buffer après une lecture dans le fichier
LSET/RSET	Ces instructions permettent de remplir le buffer d'entrée/sortie en cadrant les variables à gauche (LSET) ou à droite (RSET)

Exemple :

```
10  FIELD 1, 3 AS A$
20  LSET A$="AA"   donne  AA
                          A$
30  RSET A$="AA"   donne  AA
                          A$
```

PUT	N° fichier, N° d'enregistrement Cette instruction recopie le buffer d'entrée/sortie dans l'enregistrement indiqué
GET	N° fichier, N° d'enregistrement Cette instruction recopie l'enregistrement indiqué dans le buffer d'entrée/sortie
MKI$ (entier) MKS$ (réel) MKD$ (double précision)	Conversion de nombres en chaînes de caractères codées en binaire condensé
CVI CVS CVD	Fonction inverse
LOC	N° du prochain enregistrement à lire/écrire

Voici un exemple de programme en MBASIC gérant un fichier à Accès Direct contenant un carnet d'adresses. Vous pourrez comparer cet exemple à celui du paragraphe 1.6.4.

● **Exemple comparatif**

Nous reprenons l'application du carnet d'adresses proposée au paragraphe 1.4.3 pour la transposer sur MBASIC en lui ajoutant la gestion des trous (enregistrements vides) et pour éviter de faire croître inutilement la taille du fichier.

Ceci est obtenu facilement grâce à l'instruction FIELD qui permet de redéfinir les champs de données des enregistrements.

La structure du programme est la suivante :

Lignes 120-420 : * ouverture et création du fichier si nécessaire
* lecture du nombre d'enregistrements
* définition des champs de données

Lignes 430-580 : menu et appel des sous-programmes correspondant

Lignes 590-630 : début des sous-programmes

Lignes 640-1 460 : recherche d'un enregistrement :
— choix du mode de recherche
— lecture du fichier
— test de concordance

Lignes 1 470-1 960 : ajout d'un enregistrement :
— saisies de données
— recherche du premier enregistrement libre
— écriture dans le fichier
— modification des paramètres de gestion des trous

Lignes 1 970-2 180 : listing du fichier

Lignes 2 190-2 250 : fin du programme

Lignes 2 260-2 300 : début des sous-programmes du second niveau

Lignes 2 310-2 840 : modification/suppression d'un article
— menu
— modification
— suppression

Lignes 2 850-2 990 : ouverture du fichier et définition des champs

Lignes 3 000-3 090 : saisie du nom
Lignes 3 100-3 190 : saisie du prénom
Lignes 3 200-3 290 : saisie du genre de voie
Lignes 3 300-3 390 : saisie du nom de la voie
Lignes 3 400-3 460 : saisie du numéro
Lignes 3 470-3 560 : saisie de la ville
Lignes 3 570-3 660 : saisie du code postal
Lignes 3 670-3 850 : saisie de l'indicatif téléphonique et du numéro de téléphone
Lignes 3 860-3 950 : affichage du message d'erreur « PAS PLUS DE N CARACTERES »
Lignes 3 960-4 060 : affichage des données

```
100 ' CARNET D'ADRESSES
110 '
120 ' COPYRIGHT B.DE MERLY
130 '
140 ' 11/7/82
150 '
160 ' CREATION DU FICHIER ?
170 '
180 HOME:INVERSE
190 PRINT "      CARNET D'ADRESSES  ":PRINT
200 NORMAL
210 INPUT "EST-CE LA CREATION DU FICHIER (O/N)",A$
220 IF A$<>"N" AND A$<>"O" THEN 210
230 IF A$="N" THEN 320
240 '
250 ' CREATION DU FICHIER
260 '
270 PRINT"EN CREATION LE FICHIER PRECEDENT":PRINT" EST SUPPRIME"
280 INPUT"CONTINUEZ-VOUS QUAND MEME (O/N)",A$
290 IF A$<>"N" AND A$<>"O" THEN 280
300 IF A$="N" THEN 320
310 GOSUB 2920
320 CLOSE
330 KILL "FICHAD"
340 GOSUB 2930 'OUVERTURE FICHIER
350 LSET NB$=MKI$(1)
360 LSET SV$=MKI$(0)
370 LSET PV$=MKI$(2)
380 PUT#1,1
390 CLOSE
400 '
```



```
410 'LECTURE DU NOMBRE D'ENREGISTREMENTS
420 ' DANS LE FICHIER
430 '
440 GOSUB 2930 'OUVERTURE FICHIER
450 GET#1,1
460 NB=CVI(NB$):SV=CVI(SV$):PV=CVI(PV$)
470 CLOSE
480 '
490 ' DEFINITION DU FICHIER
500 '
510 GOSUB 2970
520 '
530 '   PROPOSITION DU MENU
540 '   ******************
550 '
560 HOME:INVERSE
570 PRINT"     *** CARNET D'ADRESSES ***":PRINT:PRINT:PRINT
580 PRINT"1- CONSULTATION AVEC/SANS MODIFICATION"
590 PRINT"2- INSCRIPTION (NOUVELLE ADRESSE)"
600 PRINT"3- LISTE DU FICHIER"
610 PRINT"4- FIN DU PROGRAMME"
620 PRINT
630 INPUT"ENTREZ VOTRE CHOIX (1,2,3,4)";A
640 IF A<1 OR A>4 THEN 560
650 NORMAL
660 ON A GOSUB 640,1490,1960,2180
670 GOTO 560
680 '
690 '
700 '     DEBUT DES SOUS-PROGRAMMES
710 '     *************************
720 '
730
740 '**********************************
750 ' RECHERCHE DANS LE FICHIER (1)
760 '**********************************
770 '
780 HOME:INVERSE:PRINT"    *** CONSULTATION ***":NORMAL
790 PRINT:PRINT:PRINT
800 PRINT"A PARTIR DE QUOI VA SE FAIRE LA RECHERCHE"
810 PRINT
820 PRINT"1-LE NOM"
830 PRINT"2-LE PRENOM"
840 PRINT"3-L'ADRESSE"
850 PRINT"4-LA VILLE"
860 PRINT"5-LE CODE POSTAL"
870 PRINT"6-LE NUMERO DE TELEPHONE"
880 PRINT:INPUT"VOTRE CHOIX (ACCOLER LES DIVERS CHIFFRES) ",A$
890 DIM RECH$(6,2),RECH(6)
900 RECH$(1,1)=SPACE$(15)
910 RECH$(1,2)="NOM A RECHERCHER"
920 RECH$(2,1)=SPACE$(15)
930 RECH$(2,2)="PRENOM A RECHERCHER"
940 RECH$(3,1)=SPACE$(21)
950 RECH$(3,2)="ADRESSE A RECHERCHER"
```

```
870 RECH$(4,1)=SPACE$(15)
880 RECH$(4,2)="VILLE A RECHERCHER"
890 RECH$(5,1)=SPACE$(5)
900 RECH$(5,2)="CODE POSTAL A RECHERCHER"
910 RECH$(6,1)=SPACE$(7)
920 RECH$(6,2)="NUMERO DE TELEPHONE A RECHERCHER"
930 A=LEN(A$)
940 FOR I=1 TO 6
950 RECH(I)=0
960 FOR J=1 TO A
970 IF VAL(MID$(A$,J,1))<>I THEN 1010
980 RECH(I)=1
990 PRINT RECH$(I,2):INPUT" ",B$
1000 LSET RECH$(I,1)=B$
1010 NEXT J
1020 NEXT I
1030 '
1040 ' RECHERCHE PROPREMENT DITE
1050 ' *************************
1060 '
1070 GOSUB 2920
1080 GET#1,1
1090 NB=CVI(NB$)
1100 IF NB>1 THEN 1120
1110 PRINT"FICHIER VIDE":INPUT"TAPEZ RETURN POUR CONTINUER",A:RETURN
1120 GOSUB 2960
1130 FOR I=1 TO NB-1
1140 J=I+1 'NUMERO D'ARTICLE
1150 OK=1
1160 GET#1,J
1170 '
1180 '   TEST DE CONCORDANCE
1190 '   *******************
1200 '
1210 '   ARTICLE LU-DONNEES RECHERCHEES
1220 '   ******************************
1230 '
1240 IF RECH(1)=0 THEN 1260
1250 IF NOM$=RECH$(1,1) THEN OK=OK AND -1 ELSE OK=0
1260 IF RECH(2)=0 THEN 1280
1270 OK = OK AND (PRENOM$=RECH$(2,1))
1280 IF RECH(3)=0 THEN 1310
1290 A$=NUMERO$+" "+GENREVOIE$+" "+NOMVOIE$
1300 OK = OK AND (A$=RECH$(3,1))
1310 IF RECH(4)=0 THEN 1330
1320 OK = OK AND (VILLE$=RECH$(4,1))
1330 IF RECH(5)=0 THEN 1350
1340 OK = OK AND (CDPOST$=RECH$(5,1))
1350 IF RECH(6)=0 THEN 1370
1360 OK = OK AND (TEL$=RECH$(6,1))
1370 IF OK=0 THEN 1440
1380 HOME:INVERSE:PRINT"DONNEES POSSIBLES":NORMAL
1390 PRINT:PRINT:PRINT:GOSUB 3960:PRINT
1400 PRINT"DESIREZ-VOUS MODIFIER OU SUPPRIMER"
1410 INPUT"CET ENREGISTREMENT (O/N)",A$
```



```
1420 IF A$<>"O" AND A$<>"N" THEN 1400
1430 IF A$="O" THEN GOSUB 2310
1440 NEXT I
1450 ERASE RECH$,RECH
1460 RETURN
1470 '
1480 '*********************************
1490 ' AJOUT D'INSCRIPTIONS (2)
1500 '*********************************
1510 '
1520 INVERSE
1530 HOME:PRINT"NOUVELLE ADRESSE":PRINT:PRINT
1540 NORMAL
1550 PRINT"SI UNE REPONSE NE PEUT ETRE FOURNIE"
1560 PRINT "TAPEZ SEULEMENT RETURN"
1570 PRINT
1580 LSET CODE$="1"
1590 GOSUB 3000 'SAISIE DU NOM
1600 GOSUB 3100 'SAISIE DU PRENOM
1610 GOSUB 3200 'SAISIE DU GENRE DE VOIE
1620 GOSUB 3300 'SAISIE DU NOM DE LA VOIE
1630 GOSUB 3400 'SAISIE DU NUMERO
1640 GOSUB 3470 'SAISIE DE LA VILLE
1650 GOSUB 3570 'SAISIE DU CODE POSTAL
1660 GOSUB 3670 'SAISIE DE L'INDICATIF ET DU NUMERO DE TELEPHONE
1670 HOME
1680 INVERSE
1690 PRINT"INFORMATIONS SAISIES "
1700 NORMAL:PRINT:PRINT
1710 GOSUB 3960 'AFFICHAGE BUFFER
1720 PRINT:INVERSE
1730 INPUT "VALIDEZ-VOUS CES INFORMATIONS (O/N) ",A$
1740 IF A$<>"N" AND A$<>"O" THEN 1730 ELSE IF A$="N" THEN 1490
1750 '
1760 '  AJOUT DE L'ARTICLE DANS LE FICHIER
1770 '
1780 PUT#1,PV
1790 GOSUB 2940
1800 IF SV<>0 THEN 1830
1810 ' PAS D'ARTICLES VIDES AVANT LA FIN DU FICHIER
1820 PV=PV+1 :GOTO 1860
1830 PV=SV
1840 GET#1,SV
1850 SV=CVI(SV$)
1860 ' STOCKAGE DES NOUVELLES DONNEES
1870 ' DANS LE 1ER ARTICLE
1880 GET#1,1
1890 NB=CVI(NB$)+1
1900 LSET NB$=MKI$(NB)
1910 LSET PV$=MKI$(PV)
1920 LSET SV$=MKI$(SV)
1930 PUT#1,1
1940 GOSUB 2980
1950 INPUT A:RETURN
1960 '
```

```
1970 '*********************************
1980 '  LISTING DU FICHIER (3)
1990 '*********************************
2000 '
2010 GOSUB 2920
2020 GET#1,1
2030 NB=CVI(NB$):PV=CVI(PV$):SV=CVI(SV$)
2040 GOSUB 2960
2050 FOR I=1 TO NB-1
2060 GET#1,I+1
2070 IF CODE$<>"1" THEN 2140
2080 HOME:INVERSE
2090 PRINT"ARTICLE NUMERO ";I
2100 NORMAL
2110 PRINT
2120 GOSUB 3960
2130 INPUT "TAPEZ RETURN POUR CONTINUER",A
2140 NEXT
2150 HOME:INVERSE:PRINT"FIN DE FICHIER":NORMAL
2160 INPUT"TAPEZ RETURN POUR CONTINUER",A
2170 RETURN
2180 '
2190 '*********************************
2200 '   FIN DU PROGRAMME (4)
2210 '*********************************
2220 '
2230 CLOSE
2240 NORMAL
2250 END
2260 '
2270 '  DEBUT DES SOUS-PROGRAMMES
2280 '  DU SECOND NIVEAU
2290 '  *************************
2300 '
2310 '
2320 ' MODIFICATION/SUPPRESSION
2330 ' ************************
2340 '
2350 ' D'UN ENREGISTREMENT
2360 ' ********************
2370 '
2380 PRINT:PRINT"SOUHAITEZ-VOUS"
2390 PRINT TAB(20);"MODIFIER L'ARTICLE (1)"
2400 PRINT TAB(20);"SUPPRIMER L'ARTICLE (2)"
2410 INPUT"VOTRE CHOIX",A
2420 IF A<1 OR A>2 THEN 2380
2430 ON A GOTO 2440,2700
2440 '
2450 '  MODIFIER UN ARTICLE
2460 '  *******************
2470 '
2480 PRINT:PRINT"QUE SOUHAITEZ-VOUS MODIFIER"
2490 PRINT:PRINT"1-LE NOM":PRINT"2-LE PRENOM":PRINT"3-L'ADRESSE":PRINT"4-
LA VILLE":PRINT"5-LE CODE POSTAL":PRINT"6-LE NUMERO DE TELE
PHONE"
```



```
2500 PRINT:INPUT"VRE CHOIX (1,2,3,4,5,6)",A
2510 IF A<>3 THEN 60
2520 GOSUB 3200  'NREVOIE
2530 GOSUB 3300  'MVOIE
2540 GOSUB 3400  'MERO
2550 GOTO 2570
2560 ON A GOSUB 30,3100,,3470,3570,3670
2570 HOME:INVERSE:INT"INFORMATIONS SAISIES":NORMAL
2580 GOSUB 3960
2590 PRINT:PRINT"SHAITEZ-VOUS"
2600 PRINT "1-MODIER UNE AUTRE DONNEE DE L'ARTICLE"
2610 PRINT "2-ENRESTRER LES MODIFICATIONS EFFECTUEES"
2620 PRINT "3-ABANNER LES MODIFICATIONS"
2630 INPUT"VOTRE IX (1,2,3)",A
2640 ON A GOTO 2442650,2690
2650 '
2660 '  ENREGISTRENT DES MODIFICATIONS
2670 '
2680 PUT#1,J
2690 RETURN
2700 '
2710 '  SUPPRESSIOD'UN ARTICLE
2720 '  **********************
2730 '
2740 GOSUB 2940 'MIFICATION FIELD
2750 GET#1,1
2760 PV=CVI(PV$):SVI(SV$)
2770 LSET PV$=MKI$)
2780 LSET SV$=MKI$/)
2790 PUT#1,1
2800 LSET CODE$="O'
2810 LSET PV$=MKI$)
2820 LSET SV$=MKI$/)
2830 PUT#1,J
2840 RETURN
2850 '
2860 '  OUVERTURE HIER
2870 '  **************
2880 '
2890 '  DEFINITIONS FIELD(S)
2900 '  ********************
2910 '
2920 CLOSE
2930 OPEN"R",#1,"FIAD.FIC",101
2940 FIELD#1,1 AS DE$,2 AS NB$,2 AS PV$,2 AS SV$
2950 RETURN
2960 CLOSE
2970 OPEN"R",#1,"FIAD.FIC",101
2980 FIELD#1,1 AS DE$,15 AS NOM$,15 AS PRENOM$,3 AS GENREVOIE$,15
AS NOMVOIE$,3 AS NERO$,20
AS COMP$,15 AS VILL,5 AS CDPOST$,2
 AS INDTEL$,7 AS TE
2990 RETURN
3000 '
```

```
3010 '  SAISIE DU NOM
3020 '  *************
3030 '
3040 INPUT "NOM ",A$
3050 N=15:IF LEN(A$)<=N THEN 3080
3060 GOSUB 3870 'AFFICHAGE MESSAGE D'ERREUR
3070 GOTO 3040
3080 LSET NOM$=A$
3090 RETURN
3100 '
3110 '  SAISIE DU PRENOM
3120 '  ****************
3130 '
3140 INPUT"PRENOM",A$
3150 N=15:IF LEN(A$)<=N THEN 3180
3160 GOSUB 3870 'AFFICHAGE MESSAGE D'ERREUR
3170 GOTO 3140
3180 LSET PRENOM$=A$
3190 RETURN
3200 '
3210 '  SAISIE DU GENRE DE VOIE
3220 '  ***********************
3230 '
3240 INPUT "GENRE DE VOIE(RUE,AV,BLD,..) ",A$
3250 N=3:IF LEN(A$)<=N THEN 3290
3260 GOSUB 3870 'AFFICHAGE MESSAGE D'ERREUR
3270 GOTO 3250
3280 LSET GENREVOIE$=A$
3290 RETURN
3300 '
3310 '  SAISIE DU NOM DE LA RUE
3320 '  ***********************
3330 '
3340 INPUT"NOM DE LA VOIE ",A$
3350 N=15:IF LEN(A$)<=N THEN 3380
3360 GOSUB 3870 'AFFICHAGE MESSAGE D'ERREUR
3370 GOTO 3340
3380 LSET NOMVOIE$=A$
3390 RETURN
3400 '
3410 '  SAISIE DU NUMERO
3420 '  ****************
3430 '
3440 INPUT"NUMERO ",A$
3450 LSET NUMERO$=A$
3460 RETURN
3470 '
3480 '  SAISIE DE LA VILLE
3490 '  ******************
3500 '
3510 INPUT"VILLE ",A$
3520 N=15:IF LEN(A$)<=N THEN 3550
3530 GOSUB 3870 'AFFICHAGE MESSAGE D'ERREUR
3540 GOTO 3510
3550 LSET VILLE$=A$
```



```
3560 RETURN
3570 '
3580 '  SAISIE DU CODE POSTAL
3590 '  *********************
3600 '
3610 INPUT "CODE POSTAL ",A$
3620 IF LEN(A$)<6 THEN 3650
3630 PRINT"PAS PLUS DE CINQ CHIFFRES SVP"
3640 GOTO 3610
3650 LSET CDPOST$=A$
3660 RETURN
3670 '
3680 '  SAISIE DE L'INDICATIF TELEPHONIQUE
3690 '  **********************************
3700 '
3710 INPUT "INDICATIF TELEPHONIQUE ",A$
3720 IF LEN(A$)<=2 THEN 3750
3730 PRINT "PAS PLUS DE DEUX CHIFFRES SVP"
3740 GOTO 3710
3750 LSET INDTEL$=A$
3760 '
3770 '  SAISIE DU NUMERO DE TELEPHONE
3780 '  *****************************
3790 '
3800 INPUT"NUMERO DE TELEPHONE ",A$
3810 IF LEN(A$)<8 THEN 3840
3820 PRINT"PAS PLUS DE SEPT CHIFFRES SVP"
3830 GOTO 3800
3840 LSET TEL$=A$
3850 RETURN
3860 '
3870 '
3880 '  AFFICHAGE MESSAGE D'ERREUR
3890 '  CHAINE TROP LONGUE
3900 '  **************************
3910 '
3920 PRINT:INVERSE
3930 PRINT "PAS PLUS DE ";N;" CARACTERES, SVP"
3940 NORMAL
3950 RETURN
3960 '
3970 '  AFFICHAGE DES DONNEES SAISIES
3980 '  *****************************
3990 '
4000 PRINT
4010 PRINT"NOM";TAB(20);NOM$
4020 PRINT"PRENOM";TAB(20);PRENOM$
4030 PRINT"ADRESSE";TAB(20);NUMERO$;" ";GENREVOIE$;" ";TAB(20);NOMVOIE$
4040 PRINT "VILLE";TAB(20);CDPOST$;" ";VILLE$
4050 PRINT "TELEPHONE";TAB(20);"(";INDTEL$;") ";TEL$
4060 RETURN
```

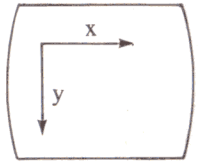
3.4.4 Possibilités graphiques et sonores

Le Basic Microsoft standard a été étendu sur l'APPLE II pour permettre au possesseur d'une carte Z80 de conserver les possibilités graphiques du moniteur et de l'APPLESOFT. Les fonctions offertes sont les suivantes :

POS	Fonction indiquant la position horizontale du curseur sur l'écran
VPOS	Fonction indiquant la position verticale du curseur sur l'écran
BUTTON (nº manette)	Fonction indiquant si le bouton de la manette de jeux indiquée est enfoncé (valeur −1) ou non (valeur 0)
BEEP I, J	Emission d'un son de tonalité I et de durée J (I entre 0 — tonalité la plus élevée et 255 — tonalité la plus basse ; J entre 0 et 255 i.e. 1 s)
GR nº écran, nº couleur	Passage en mode graphique basse résolution sur l'écran 0-1 (4 ou 0 lignes de textes) avec la couleur voulue

COLOR = numéro de couleur (0 à 15) sur l'écran

PLOT x, y (x entre 0 et 39, y entre 0 et 47)



VLIN y1, y2 AT x :	En mode basse résolution, tracement d'une ligne verticale du point (x, y1) au point (x, y2)
HLIN x1, x2 AT y :	En mode basse résolution, tracement d'une ligne horizontale du point (x1, y) au point (x2, y)
SCRN (x, y)	Fonction retournant la valeur de la couleur du point (x, y)
TEXT	Retour en mode texte avec effacement de l'écran si retour de basse résolution Sans effacement de l'écran si retour de haute résolution et positionnement à la ligne 24

HTAB	nº de colonne	Déplacement du curseur vers la colonne indiquée modulo 40
VTAB	nº de ligne	Déplacement du curseur vers la ligne indiquée modulo 24
INVERSE		Passage en mode inverse de l'écran (caractères noirs sur fond blanc)
NORMAL		Retour en mode normal de l'écran (caractères blancs sur fond noir)
PDL	(nº de manette)	Lecture de la valeur indiquée par la manette de jeux (0-255)

*Comme pour le moniteur de l'*APPLE II, *vous devez attendre entre deux appels de la fonction. Il vous suffit pour cela d'insérer la ligne*

```
FOR Z=1 TO 10:NEXT Z
```

Possibilités graphiques haute résolution offertes par GBASIC :

HGR < nº écran >, < nº couleur >

où numéro écran est compris entre 0 et 3
numéro couleur est compris entre 0 et 12.

Initialisation du mode graphique haute résolution

Les numéros 0-3 d'écran correspondent aux formats suivants :

Nº écran	Effacement de l'écran	Mode de l'écran
0	OUI	280 × 160, 4 lignes de texte
1	OUI	280 × 192, pas de texte
2	identique au zéro sans effacement de l'écran	
3	identique au un sans effacement de l'écran	

HCOLOR = < nº couleur >

Positionnement de la couleur désirée (0-12) avec comme couleurs correspondantes :

0 noir 0	4 noir 1	8 noir 2	à utiliser avec vert et violet
1 vert	5 orange	9 blanc 2	
2 violet	6 bleu	10 noir 3	
3 blanc 0	7 blanc 1	11 blanc 3	avec orange et bleu
		12 négatif	

Les différentes couleurs noires ou blanches correspondent à diverses finesses de trait.

HPLOT <x1, y1> TO <x2, y2>---

affichage d'un point ou tracé d'une ligne en mode haute résolution.

HSCRN (x, y) indique s'il existe sur l'écran un point de coordonnées (x, y) (−1 s'il n'y a pas de point, 0 sinon).

Ces fonctions étant de maniement semblable à celles du moniteur et de l'APPLESOFT, le lecteur intéressé se reportera au chapitre 3.

Il est à notre avis préférable, pour programmer des jeux, de pouvoir disposer des formes graphiques préenregistrées (cf. Tome 1) et simples à déplacer.

La présence de fonctions graphiques dans le Basic Microsoft peut permettre une meilleure mise en forme des résultats sur l'écran. Il est utile de les connaître à cet effet.

3.4.5 Appel de sous-programmes assembleurs et adresses mémoire

Il existe plusieurs possibilités d'appel de sous-programmes assembleur à partir du MBASIC :

— fonction mono-argument USR équivalente aux fonctions FN du Basic,

— appel de sous-programmes assembleur Z80 par l'instruction CALL,

— appel de sous-programmes assembleur 6502 par l'instruction CALL %.

● **Fonctions mono-argument USR**

L'instruction

DEF USR<digit> = <expression entière>
0 à 9

permet de définir 10 fonctions assembleur 8080 en précisant leur adresse

Exemple : DEF USR3 = 2048

L'appel des diverses fonctions se fera alors de la manière suivante :

USR<digit> (argument)
0 à 9

Exemple : B = USR4(2000)

Lors de l'appel d'une fonction USR, le registre A (accumulateur) indique le type d'argument. La valeur de A peut être :

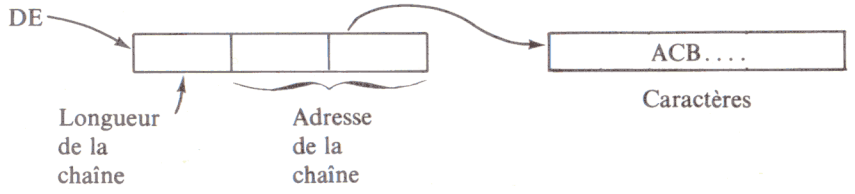
2 entier (sur 2 octets)
3 chaîne de caractères
4 nombre réel simple précision
8 nombre réel double précision

Au retour de la fonction, l'accumulateur doit contenir le type du paramètre de retour.

* Si le paramètre d'appel est une chaîne de caractères, les registres DE pointent vers 3 octets appelés descripteur de chaîne

DE		ACB...
longueur de la chaîne	adresse de la chaîne	caractères

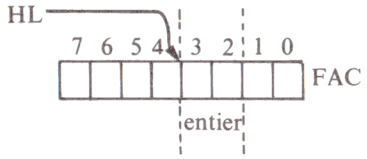
L'exemple suivant inverse les deux premiers caractères d'une chaîne de caractères :



L'exemple suivant inverse les deux premiers caractères d'une chaîne de caractères :

```
10 POKE &H8000,&HEB      'XCHG
20 POKE &H8001,&H23      'INX    H
30 POKE &H8002,&H5E      'MOV    E,M
40 POKE &H8003,&H23      'INX    H
50 POKE &H8004,&H56      'MOV    D,M
60 POKE &H8005,&H2B      'DEX    H
70 POKE &H8006,&H2B      'DEX    H
80 POKE &H8007,&HEB      'XCHG
90 POKE &H8008,&H7E      'MOV    A,M
100 POKE &H8009,&H23     'INX    H
110 POKE &H800A,&H46     'MOV    B,M
120 POKE &H800B,&H77     'MOV    M,A
130 POKE &H800C,&H2B     'DEX    H
140 POKE &H800D,&H70     'MOV    M,B
150 POKE &H800E,&H3E     'MVI    A,3H
160 POKE &H800F,&H3
170 POKE &H8010,&HC9     'RET
180 DEF USR=&H8000
190 INPUT A$
200 A$=USR(A$)
210 PRINT A$
220 GOTO 190
```

* Si le paramètre d'appel est un entier sur 2 octets, les registres HL pointent sur une partie d'un buffer appelé « Floating point accumulator » de 8 octets.



FAC3 = poids faible (reste de la division par 256)
FAC2 = poids fort (quotient de la division par 256)

L'exemple suivant montre un exemple de division par 256 d'un entier :

```
10 POKE &H8000,&HE5      'PUSH   H
20 POKE &H8001,&H23      'INX    H
30 POKE &H8002,&HAF      'XRA    A
40 POKE &H8003,&H77      'MOV    M,A
```



```
50 POKE &H8004,&HE1      'POP    H
60 POKE &H8005,&H3E      'MVI    A,2H
70 POKE &H8006,&H2
80 POKE &H8007,&HC9      'RET
90 DEF USR=&H8000
100 INPUT A%
110 A%=USR(A%)
120 PRINT A%
130 GOTO 100
140,END
```

* Pour un nombre réel, HL pointent sur FAC3 avec

FAC3, FAC2, FAC1 } mantisse { poids faible / poids fort
FAC0 exposant

* Pour un nombre réel double précision, FAC7 à FAC4 représentent 4 octets de mantisse en plus. HL pointent sur FAC7.

Le paramètre de retour d'USR doit être dans le même format qu'un paramètre de même type à l'appel.

● Sous-programmes assembleur Z80

Ils permettent d'avoir plus de paramètres d'appel que les fonctions USR.

L'instruction permettant d'appeler un sous-programme est :

CALL < nom de variable > < liste d'arguments >

La variable indiquée contient l'adresse à laquelle le contrôle sera passé.

Le nombre de paramètres doit être connu par le sous-programme appelé.

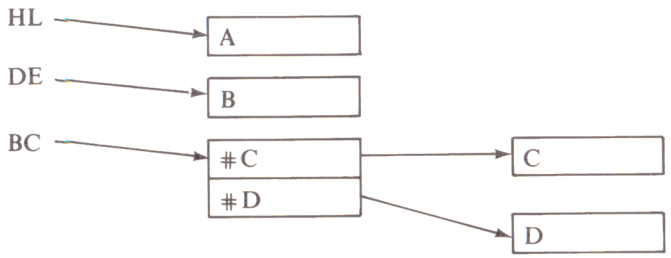
Le passage de ceux-ci est réalisé de la manière suivante :

HL adresse de l'argument numéro 1
DE adresse de l'argument numéro 2
BC adresse d'une zone mémoire contenant les adresses des autres paramètres

Par exemple si le MBASIC rencontre l'instruction :

CALL SPROG(A,B,C,D)

il créera le contexte suivant :



C'est au sous-programme assembleur de récupérer les paramètres que lui passe le MBASIC.

● **Sous-programmes assembleur 6502**

Le format de l'instruction d'appel de sous-programmes 6502 est le suivant :

CALL% nom s.p (liste de paramètres)

Trois parammètres au plus peuvent être passés entre le Basic et l'assembleur. Ils sont transmis respectivement dans les registres :

6502 A(1)
X(2)
Y(3)

● **Fonction VARPTR**

Elle permet de connaître l'adresse d'une variable en mémoire. Son format est le suivant :

PTX = VARPTR(X)

où X est une variable de type quelconque initialisée.

Cette fonction peut être utile pour appeler des sous-programmes assembleur Z80 ou 6502.

3.5 LANGAGES DISPONIBLES

3.5.1 Introduction

De nombreux langages et logiciels de base sont adaptables très facilement au CP/M de l'APPLE II *via* une carte d'interface V24, et à un programme de transfert adéquat. La firme Microsoft a ainsi transféré ses langages sur la Softcard et vous pouvez disposer du Fortran, du Cobol, d'un Compilateur Basic, du Pascal, du LISP, ou de programmes évolués de traitement de texte tels que Wordstar.

3.5.2 Système de développement assembleur ALDS

Le produit A.L.D.S. développé spécialement par Microsoft pour l'APPLE II vous permettra de programmer dans les trois langages d'assemblage des microprocesseurs Zilog Z80, Intel 8080 et Rockwell 6502. En sus de ces possibilités, le système A.L.D.S. est le seul assembleur (avec celui de la carte langage) permettant l'emploi de macro-instructions, l'assemblage conditionnel de différentes zones de programmes et générant du code binaire relogeable (c'est-à-dire implantable n'importe où en mémoire, après édition de liens « LINK »).

Le système A.L.D.S. contient les logiciels suivants :

a) M80 : macroassembleur du système,

b) L80 : éditeur de liens permettant de rassembler des programmes assembleurs, Basic compilé, Fortran, Cobol,... générant un programme chargeable en mémoire,

c) CREF80 : générateur de références croisées (c'est-à-dire fournissant la liste des variables et leurs emplois),

d) DDT65 : programme d'aide à la mise au point équivalant au logiciel DDT de CP/M, pour le microprocesseur 6502 (examen et modification des registres, de la mémoire ; désassemblage de zones mémoire ; etc...),

e) CPMXFER : logiciel de transfert de fichiers binaires contenant des programmes 6502 prêts à être chargés, du système CP/M vers le DOS APPLE.

3.5.3 Compilateur Basic

Accroître la vitesse d'exécution d'un programme n'est pas le seul intérêt du Compilateur Basic de Microsoft. Selon les programmes compilés, la vitesse d'exécution est en moyenne de 10 à 20 fois plus rapide, et peut même devenir jusqu'à 30 fois supérieure pour des logiciels utilisant au maximum les variables entières.

De plus, vous pourrez regrouper des programmes Basic avec du Fortran ou du Cobol. Le code généré par le compilateur est fortement optimisé. Il peut pratiquement n'être interprétable que par un spécialiste et facilite ainsi la protection de vos logiciels.

Il faut noter toutefois qu'un certain nombre d'instructions du Basic interprété ne sont pas utlisables en Basic compilé.

3.5.4 Le Compilateur Fortran

Le Fortran Microsoft est conforme à la norme américaine ANSI 1966, excepté l'absence des nombres complexes. Créé spécifiquement pour résoudre les problèmes numériques, le langage Fortran est devenu le langage le plus utilisé pour les applications scientifiques. De nombreuses bibliothèques de sous-programmes Fortran 66 existent qui sont facilement adaptables à l'APPLE II.

Comme le Fortran existe avec la carte langage, nous ne le développerons pas

*Les logiciels de contrôle des entrées/sorties non standards de l'*APPLE II *sont adaptables facilement au Fortran.*

3.5.5 Le Compilateur Cobol

Le langage Cobol domine les applications de gestion informatisées depuis de nombreuses années. Il est spécialement adéquat pour manipuler les importants volumes de données gérées. Ceux-ci sont ordonnés de manière hiérarchique dans des structures logiques.

Le Cobol Microsoft a été développé spécialement pour un environnement de micro-ordinateurs avec un comportement interactif, un formatage d'écran et une mise au point interactive des programmes. Il admet des nombres représentés sur 18 bits, nécessaires à la précision des variables et réalise les opérations internes sur 38 bits.

Il permet de gérer 4 types de fichier. Le logiciel de tri « Microsoft Sort » est fourni avec le Cobol et permet de réorganiser des fichiers d'une taille allant jusqu'à 2 milliards d'octets.

3.5.6 Le LISP

Le LISP est un langage de très haut niveau pour manipuler des symboles et de traiter des expressions symboliques. Il a été créé par et pour des chercheurs en intelligence artificielle, mais il est aussi utilisé par des scientifiques ou par des mathématiciens.

Les structures et objets du LISP de Microsoft conviennent à ces usages, de par sa nature récursive et son habileté à manier des listes de données. Les programmes et les données sont représentés par des listes chaînées. Ceci rend plus aisé la réorganisation de données pour diverses applications, et élimine le besoin d'allouer des zones mémoire aux différentes parties du programme. Faciles à apprendre et à utiliser, les objets du LISP permettent de développer des expressions très complètes. Avec les 24 fonctions de base du LISP de Microsoft, vous pourrez créer votre propre langage.

Le produit muSTAR livré avec muLISPP est un éditeur de textes facilitant la mise au point des programmes.

3.5.7 Système muSIMP/muMATH

Ce système propose la manipulation de symboles mathématiques pour faire de l'algèbre, des calculs trigonométriques, des dérivations formelles, des intégrations formelles ($\int X\,dx = X^2/2$ par exemple).

Il est divisé en 2 parties intitulées muMATH et muSIMP (langage dans lequel est écrit muMATH).

muMATH est un ensemble de programmes qui procurent des facilités de traitement algébrique et analytique. Les expressions sont simplifiées au fur et à mesure de leur saisie. Vous pouvez créer votre propre logiciel, créer des leçons selon vos désirs, etc...

muMATH est écrit dans le langage muSIMP, sous-ensemble de LISP, spécialement conçu pour implémenter des fonctions algébriques. Il sert aussi à développer des applications d'intelligence artificielle.

3.5.8 Pascal

Le Pascal existe sur la Softcard. Nous n'en parlerons pas, la carte-langage et le Pascal UCSD étant largement répandus.

CHAPITRE 4

Cartes d'extension pour APPLE II

4.1 INTRODUCTION

De nombreuses cartes d'extensions sont disponibles pour l'APPLE II, qui est actuellement le micro-ordinateur le plus riche en extensions.

Ces cartes se répartissent en 4 catégories principales :

— Cartes permettant la connexion de périphériques « standard ».
— Cartes d'extension mémoire.
— Cartes permettant de disposer d'un logiciel plus complet.
— Cartes permettant l'utilisation d'un autre microprocesseur et en général d'un autre système d'exploitation.

Ce chapitre présente, sans rentrer dans les détails, les principales cartes.

4.1.1 Cartes de connexion pour périphériques standards

Il s'agit essentiellement de cartes d'interface permettant la connexion de périphériques, non conçus exclusivement pour l'APPLE II (disques durs, terminaux, imprimantes, modem...).

Excepté les disques, qui nécessitent une interface spéciale, les périphériques classiques sont connectés grâce à trois principaux types de cartes :

— cartes d'interface parallèle, les 8 bits d'un octet sont transmis en parallèle,
— cartes d'interface série asynchrone qui transmettent successivement les 8 bits d'un octet sur un même fil sans que les partenaires (APPLE, périphérique) ne se soient synchronisés auparavant,
— cartes d'interface série synchrone pour laquelle les partenaires sont en permanence synchronisés entre eux par un fil d'horloge.

APPLE propose les trois cartes d'interface suivantes :

— interface de communication fonctionnant en boucle de courant 20 mA à la vitesse de 110 ou 300 bits/seconde, adaptée aux terminaux écran-clavier et aux modems acoustiques ;
— interface série RS232 (A à E) asynchrone fonctionnant à une vitesse comprise dans l'intervalle 75-19 200 bits/seconde et d'emploi plus souple ;
— interface parallèle pour imprimantes.

Les cartes d'interface de la firme California Computer System étant de très bonnes cartes, nous les citons ci-dessous :

CCS 7712 : carte d'interface série synchrone.
CCS 7710 : carte d'interface série asynchrone,
CCS 7720 : carte d'interface parallèle,

Entre dans cette catégorie, la carte EPSON présentée au tome 1, paragraphe 2.3.

4.1.2 Cartes d'extension mémoire

Parmi ces cartes citons :

— La RAMCARD dont l'origine est Microsoft (capacité 16 K octets, ce qui constitue un bon complément pour la Softcard).
— La carte de SATURN SYSTEM qui permet des extensions de 16, 32, 64, ou même 128 K octets. Cette carte est livrée avec un complément de logiciel permettant de la mettre en œuvre. En version 128 K octets, elle

permet de simuler également des fichiers disque en mémoire vive, ce qui réduit considérablement les temps d'accès. Un logiciel complémentaire permet d'exploiter VISICALC en utilisant le supplément de mémoire disponible, ce qui réduit également le nombre d'accès disque et améliore les performances à l'exécution.

— La carte LEGEND comporte 128 K octets de mémoire vive, découpée en 8 blocs de 16 K octets. Elle peut être également livrée avec un complément de logiciel compatible avec le DOS 3.3, ce qui permet de simuler des fichiers disque.

Dans aucun cas il ne faut pas oublier, en fin de travail, de copier les fichiers modifiés sur disquette pour ne pas les perdre.

4.1.3 Cartes permettant de disposer d'un logiciel de base plus complet

Citons les deux plus connues :

— La carte langage qui permet en particulier de disposer du Pascal UCSD et qui a fait l'objet d'un chapitre spécial compte tenu de son importance.

— La carte M/DOS, développée par l'entreprise française MIS, qui est essentiellement destinée à des applications de gestion. La carte M/DOS apporte un complément important au DOS 3.3, notamment pour la gestion de fichiers (fichiers séquentiels indexés à une ou plusieurs clés), pour la saisie et l'édition (utilisation très simple de masques de saisie et d'impression), possibilité de réseaux, etc...

4.1.4 Cartes permettant l'utilisation d'un autre processeur

La carte la plus connue est la Softcard qui permet de disposer du système d'exploitation CP/M. Il existe cependant d'autres cartes moins connues :

— Cartes 8088 permettant de simuler l'Ordinateur Personnel d'IBM. Ces cartes comportent un microprocesseur 8088 et 64 ou 128 K octets.

— Carte 6809 permettant de disposer du système d'exploitation OS9 et d'une plus grande rapidité d'exécution en Basic, Fortran et Pascal. Cette carte comporte un 6809 à la fréquence de 1 MHz alors que sa vitesse nominale est de 2 MHz.

4.1.5 Cartes permettant des applications d'acquisition de mesures

Ces cartes sont très nombreuses, citons :

— La carte A/D et D/A commercialisée par APPLE pour l'acquisition de données analogiques et la sortie d'informations sous forme analogique.

— La carte ADALAB, commercialisée par ALPHA SYSTEMES, qui permet non seulement les entrées/sorties analogiques, mais également des entrées/sorties sur 8 ou 16 bits en parallèle, une horloge pour rythmer l'acquisition. Elle est livrée avec un peu de logiciel, afin de faciliter l'apprentissage de l'utilisateur.

4.1.6 Cartes 80 colonnes

La carte Vidéotherm permet l'affichage de 24 lignes de 80 caractères de dimensions 7 × 9 points en lettres majuscules et minuscules. Le jeu de caractères est programmable par logiciel dans des dimensions de 8 × 2 à 16 points. Il est possible d'avoir les jeux de caractères anglais, français, allemand, japonais, etc. Le constructeur de la carte Vidéotherm propose en outre une carte de reconfiguration de clavier, et une carte de sélection des modes vidéo 40 ou 80 colonnes.

Fournisseur : VIDEX
897 N.W. Grant Ave,
Corvallis
Oregon 97330
USA

Autre carte 80 colonnes : la Sup'R terminal compatible avec la carte M/DOS décrite au paragraphe 4.2.

4.1.7 Cartes à digitaliser

Ces cartes transforment une information du mode extérieur (analogique) en une information compréhensible par l'APPLE II. En plus des cartes CA/D citées au paragraphe précédent, qui travaillent sur une dimension, nous pouvons citer des systèmes à deux dimensions (tablette graphique, carte Digisector...) ou à trois dimensions. Nous reviendrons sur ces cartes au paragraphe 4.7.

4.1.8 Synthèse et reconnaissance sonore

Parmi les cartes qui permettent de manipuler les sons sur l'Apple II :

SUPERTALKER : digitaliseur et synthétiseur de voix humaine (2 min par disquette)

MUSIC SYSTEM : synthétiseur de musique avec composition à l'aide des manettes, du clavier ou d'un stylet magnétique

VOICE INPUT MODULE : reconnaissance d'un vocabulaire limité, enregistré précédemment

VOICE BOX : synthèse d'un texte tapé au clavier à l'aide d'un dictionnaire de phonèmes enregistrés sur disquette.

4.2 CARTE M-DOS (APPELÉE MAINTENANT MEM/DOS)

Cette carte est surtout destinée aux applications de gestion :

— gestion des fichiers à classer,
— gestion d'écran (masque de saisie),
— gestion d'édition (masque d'édition).

4.2.1 Gestion des Fichiers

Cette carte permet l'utilisation de fichiers séquentiels, indexés, multiclé, ainsi que la gestion de fichiers relatifs et de fichiers séquentiels classiques. Quel que soit le type de fichier, la gestion des disques est dynamique. La taille de fichier ne doit pas excéder 60 000 articles. Les disques gérés peuvent être des disquettes 5 pouces, 8 pouces ou des disques durs.

4.2.2 Gestion d'Ecran

Pour la gestion de l'écran, avec des masques de saisie. Le contrôle lors de la saisie est effectué par le logiciel contenu dans la carte.

La saisie est extrêmement rapide, avec un excellent contrôle de vraisemblance de données. Le même masque peut être utilisé par plusieurs programmes.

4.2.3 Gestion d'Edition

Les éditions, en particulier vers l'imprimante, peuvent se faire avec des masques, ce qui donne accès à l'instruction PRINT USING, notamment, qui n'est normalement pas disponible avec le Basic de l'APPLE.

Cette carte M/DOS 6502 contient essentiellement des programmes écrits en mémoire EPROM, ce qui libère complètement la mémoire vive de l'APPLE.

Il existe une version monoposte et une version multipostes. En version multipostes, elle permet à plusieurs APPLE de protéger des périphériques (mémoire de masse et imprimante) lorsque des disques sont mis en commun :

— partage des fichiers,
— accès simultané à un même fichier par plusieurs utilisateurs avec réservation d'articles. Il est également possible pour un utilisateur de se réserver des programmes et des fichiers stockés sur un disque commun.

Fournisseur : MICRO INFORMATIQUE SERVICE
3, rue Meyerber
06000 NICE
Tél. (93) 87.74.67

4.3 CARTES MEM/PLOT

Cette carte, fournie par MIS, sert aux applications graphiques. Elle contient 20 K octets de mémoire EPROM qui gèrent de 1 à 3 périphériques graphiques, tels que :

— crayon optique,
— table traçante 1 à 16 couleurs,
— imprimante de graphiques,
— tablette de digitalisation,
— écran graphique.

En général, pour des applications graphiques, la configuration du système est composée d'un écran graphique, d'une table traçante et d'une tablette de digitalisation.

Le mode d'exploitation consiste à ajouter, au Basic de l'APPLE, un module graphique qui est un sous-ensemble très proche du module graphique de la future norme Basic. En outre, ce système trace les axes

— ett les graduations sur les axes — au moyen d'instructions de type GR"XAXIS" pour l'axe des abscisses et GR"YAXIS" pour l'axe des ordonnées. Comme dans le module graphique, cette carte permet la gestion de sous-programmes graphiques (définition, tracé, transformation de l'image : rotation, translation, effet de zoom).

4.4 CARTES PERMETTANT L'UTILISATION D'UN AUTRE PROCESSEUR

4.4.1 Cartes 6809

Cette carte comporte un 6809 fonctionnant à 1 MHz. Le 6809 est plus rapide que le 6502 car il dispose de registres de 16 bits. Elle dispose du système d'exploitation OS/9 et du Basic sous-OS/9 qui accepte des variables locales, des sous-programmes avec transmission de paramètres, et des fonctions mathématiques plus nombreuses que celles du Basic de l'APPLE (par exemple ARCSIN et ARCCOS).

Cette carte exécute les programmes écrits en Pascal ou en Fortran, avec de meilleures performances.

4.4.2 Cartes Z80

Nous avons déjà vu la Softcard. Il existe des cartes concurrentes :

La CP/M Card qui permet de disposer du CP/M version 3, à condition d'y adjoindre la carte 80 colonnes.

— L'APPLI-Card qui comporte un Z80 64 K octets de mémoire vive, la gestion de l'écran avec 70 colonnes et les logiciels CP/M, SB80, WORDSTARD. Une version de cette carte fonctionne à 6 MHz (meilleures performances à l'exécution).

— La Z-Card.

— La CP/M-Card, plus récente, qui est livrée avec CP/M + (version 3), C-Basic et quelques autres programmes. Cette carte comporte un Z80 B à 6 MHz et 64 K de mémoire vive.

Fournisseur : APPLI-CARD
PCPI
16776 Bernardo Center Drive
San Diego, Cal 92128
USA

4.4.3 Cartes 8088

Plusieurs cartes 8088 comportent des compléments de mémoire vive (24 à 128 K en général) pour pouvoir utiliser le système d'exploitation MS-DOS, disponible sur l'ordinateur personnel d'IBM.

Dans la pratique, ces cartes ne présentent d'intérêt que pour les configurations disposant d'au moins deux unités de disquettes et d'une imprimante.

4.5 CARTES POUR APPLICATIONS DANS LABORATOIRES

Ces cartes sont très nombreuses. Nous ne pouvons en donner ici qu'un aperçu.

Il existe des cartes de conversion analogique, numérique, des cartes pour entrées/sorties TTL, des cartes permettant l'interface avec le BUS IEEE-488 très utilisé pour les applications d'acquisition de mesures en laboratoire (HEWLETT PACKARD est à l'origine de ce bus). Une carte originale est la 85aSCOPE qui transforme l'APPLE en oscilloscope. Elle permet, avec un APPLE II, de faire de l'acquisition de mesures sur fréquence pouvant atteindre 50 MHz. La conversion analogique digitale a une précision de 8 bits.

Fournisseur : NORTHWEST INSTRUMENT SYSTEMS
P.O. Box 1309
Beaverton, Oregon 97075 Tél. (503) 297.143
USA

4.6 CARTE P.A.R. (CARTE PROCESSEURS ARITHMÉTIQUES RAPIDES)

Il s'agit d'une carte particulière pour effectuer des calculs rapides en virgule flottante. Une option « mémoire morte » de cette carte permet de calculer la transformée des FOURIER rapide. Une autre carte permet d'effectuer des opérations sur 16 et 32 bits avec de meilleures performances que celles obtenues avec l'APPLESOFT. Cette carte permet également le

calcul de fonctions mathématiques de type SINUS, LOGARITHME, etc...

Fournisseurs : 1) IEF
193, rue de Javel
75015 PARIS
Tél. (1) 828.06.01
2) Carte CCS 7811B (circuit AMD9511)
California Computer Systems

4.7 CARTES A DIGITALISER

4.7.1 Tablette graphique

La tablette graphique APPLE est un système à deux dimensions sur lequel vous réalisez des dessins à l'aide d'un stylet magnétique. Le dessin tracé est alors digitalisé pour être traitable par le système APPLE II et pour pouvoir être affiché sur l'écran.

Ceci permet, par exemple, de faire de la création artistique sur l'APPLE II.

Fournisseur : APPLE SEEDRIN
av. de l'Océanie
Z.A. de Courtabœuf
91... Les Ulis

4.7.2 Cartes Vision

Il s'agit d'un ensemble comportant : une caméra vidéo spécialisée donnant une image de définition 128 × 256 (inférieure à l'écran de l'APPLE II), un pied, la carte d'extension, le câble, et le logiciel de mise en œuvre.

Ce matériel existe en version APPLE II, IBM-PC, TRS80, Commodore ou Sinclair ZX81.

Fournisseur : MICRON TECHNOLOGY INC
2805 East Columbia Road
Boise, Idaho 83706
USA

Une autre carte, la Digisector, offre une résolution du même ordre avec traitement possible toutes les secondes.

Ces cartes sont utiles dans des applications de reconnaissance de formes en reprenant par exemple l'image digitalisée dans la page graphique haute résolution de l'APPLE II.

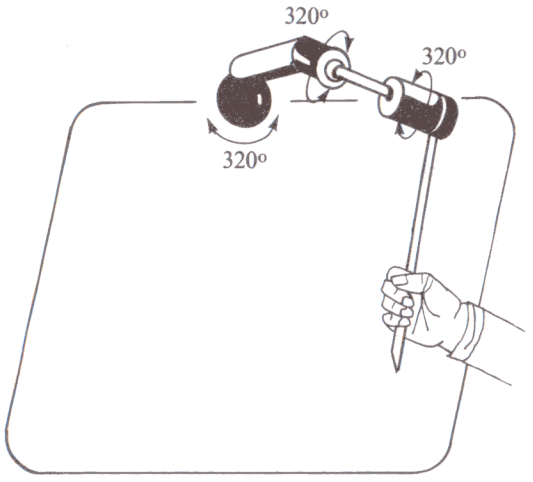
4.7.3 Carte à digitaliser en trois dimensions

Système composé d'une tablette graphique à deux dimensions sur laquelle a été adaptée un bras articulé. Celui-ci permet de suivre le contour d'un objet, qui est alors reproduit sur l'écran en perspective. Le bras est composé de trois segments ayant une possibilité de rotation de 320°, et la tablette a une surface de 13,5 × 16 pouces.

44,5 × 52,8 centimètres

Le modèle standard est destiné à l'APPLE II. Il existe un modèle dit « professionnel » connectable à l'APPLE II et à l'ordinateur personnel IBM-PC, comportant un quatrième axe de rotation.

Fournisseur : MICRO CONTROL SYSTEM INC
143 Tunnel Road
Vernon, Connecticut 06066
USA



4.8 CARTES DE TRAITEMENT DE LA PAROLE OU DE LA MUSIQUE

4.8.1 Traitement par échantillonnage. Carte Super Talker

Cette carte permet d'échantillonner la voix humaine par l'intermédiaire d'un microphone, de la traiter (des programmes de comparaison ou de reconnaissance vocale sont facilement réalisables) et de la restituer *via* un haut-parleur ou un amplificateur Hifi.

L'ensemble Super-Talker contient une carte d'extension, un microphone, un haut-parleur et les logiciels d'échantillonnage et de restitution de la parole, de stockage sur disquette de fichier de parole digitalisée.

Les quatre vitesses d'échantillonnage : 500 octets/seconde, 1 000 o/s, 2 000 o/s ou 4 000 o/s sont utilisables, ce qui correspond à des fréquences respectives de 4 000 Hz, 8 000 Hz, 16 000 Hz ou 32 000 Hz. A la vitesse de 2 K octets/seconde (16 kHz), il est possible de stocker 50 s de parole sur une disquette.

Cette carte peut être utilisée pour d'autres sons que la voix humaine à condition que la fréquence soit dans l'intervalle 300-3 000 Hz (intervalle de fréquence de la voix).

Fournisseur : MOUNTAIN HARDWARE INC
300 Harvey West Blvd
Santa Cruz, CA95060
USA

4.8.2 Carte « Music system »

Cette carte musicale met la musique à portée de tous. Elle offre un synthétiseur digital 16 notes avec sortie en stéréophonie. Vous pouvez ainsi, à l'aide d'un crayon lumineux, composer vos partitions...

Fournisseur : MOUNTAIN HARDWARE
300 Harvey West Blvd
Santa Cruz, CA95060
USA

4.8.3 Carte « Voice Input » Module de reconnaissance de la parole

Cette carte permet de faire de la reconnaissance de parole pré-enregistrée. En fait la reconnaissance n'est possible que sur des mots isolés.

Exemple : carte VMC 2020.

Sur la carte, sont présents 8 K octets de mémoire vive (RAM) et 4 K octets de mémoire morte ROM. Sa capacité de reconnaissance est limitée à 80 mots ; le temps nécessaire pour la reconnaissance est inférieur à 150 ms, ce qui impose un intervalle minimum entre deux mots successifs de 160 ms. Elle est livrée avec un microphone et son interface.

Fournisseur : VOICE MACHINE COMMUNICATIONS INC
10522 Covington Circle
Villa Park
California 92667
USA

4.8.4 Carte Voice Box

Ce système permet de synthétiser un texte saisi au clavier en le décomposant en phonèmes (sons de bases - be, te, ...) et en synthétisant ces derniers. Un dictionnaire de 64 phonèmes adapté à la langue anglaise est en mémoire morte sur la carte d'extension. Le passage à un autre langage se fait en changeant la ROM ; seule existe actuellement la version anglaise.

Fournisseur : THE ALIEN GROUP
Department PT-2
27W 23 St
New York 10010
USA

INDEX TOME 2

EDIMICRO
10, rue Henri-Pape 75013 PARIS

Imprimé en France. — JOUVE, 18, rue Saint-Denis, 75001 PARIS
N° d'éditeur : 11001 Dépôt légal : Avril 1983