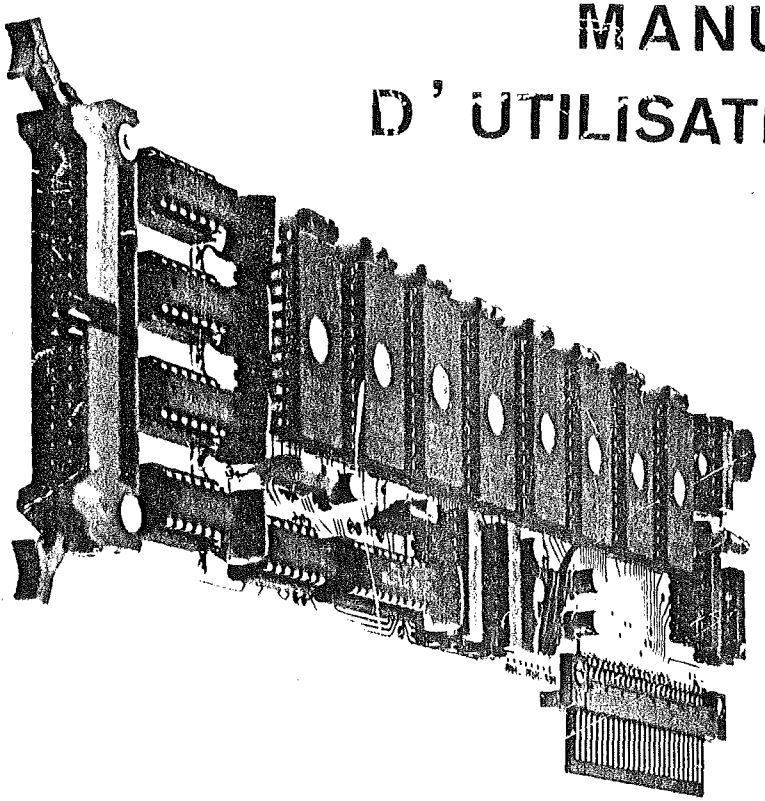
CARTE M/DOS 6502

LE SYSTÈME D'EXPLOITATION DU 6502

MANUEL D'UTILISATION



UN PRODUIT MICRO INFORMATIQUE SERVICE FRANCE

S O M M A I R E

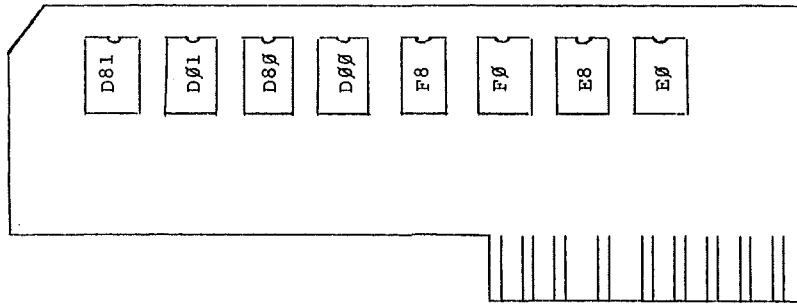
MONTAGE DES REPROMS SUR LA CARTE M/DOS 6502

Chaque reprom possède une petite encoche à l'une de ses extrèmités.

Cette encoche doit toujours être placée en haut de la carte.

Il y a huit reproms. Lorsque vous regardez la carte côté composants, le connecteur pour slot APPLE tourné vers le bas, vous devez les placer de gauche à droite, dans l'ordre suivant :

D81 DØ1 D8Ø DØØ F8 FØ E8 EØ



IMPORTANT : Manipulez les reproms avec précaution.
Après l'insertion des reproms dans leurs supports, vérifiez que toutes les broches de chacune d'elles soient bien en place et bien enfoncée.

MICRO INFORMATIQUE SERVICE

TRES IMPORTANT - TRES IMPORTANT - TRES IMP

CREATION D'UNE DISQUETTE DE DEMARRAGE 140K

La disquette de démarrage qui vous est fournie avec la carte, est, si vous n'avez rien spécifié, en 110K.

MAIS SI VOUS POSSEDEZ UN CONTROLEUR DE DISQUETTE 140K, IL VOUS EST POSSIBLE DE FABRIQUER VOUS-MEME LA MEME DISQUETTE DE DEMARRAGE EN 140K.

Pour cela, après avoir placé la carte M/DOS 6502 comme indiqué dans le manuel, montez deux lecteurs sur votre carte controleur 140K. Placez cette dernière sur le SLOT 6. Introduisez dans le drive Ø (SLOT 6, drive 1) la disquette BASIC fournie avec votre controleur 140K. Démarrez le système.

Le programme vous demande de placer une disquette 13 secteurs. Introduire alors la disquette de démarrage 110K. Vous avez alors démarré le système M/DOS 6502 110K.

Exécutez alors le programme RWTS 3.3 en tapant : RUN "RWTS 3.3"

Il vous est alors demandé : QUEL DRIVE VOULEZ-VOUS EN 140K ?

Répondre : 1

Une fois cette opération terminée, placez une disquette vierge dans le drive 1 (SLOT 6, drive 2) ; exécutez le programme BOOT par : RUN "BOOT"

Ce programme prépare une disquette démarrant seule. Le programme vous demande : DRIVE CHOISI ?

Répondre : 1

Il vous demande alors : FORMATTAGE PHYSIQUE ?

Répondre : O (oui)

Une fois cette opération terminée, faire CTRL/A pour sortir du programme. Votre disquette est prête : elle démarre automatiquement en 140K. Il ne reste plus qu'à placer sur cette disquette les utilitaires. Pour cela, faire : RUN " Ø : AUTO COPIE

Indiquez : ORIGINAL EN Ø, COPIE VERS 1

Il vous est demandé : VOULEZ-VOUS UNE SELECTION ?

Répondre : N (non)

La carte M/DOS 6502 est garantie trois mois, pièces et main d'oeuvre. Port en sus.

CHAPITRE I :

DESCRIPTION GÉNÉRALE DU SYSTÈME

Le M/DOS 6502 est un outil puissant d'aide à la réalisation de logiciels de gestion. Il comprend deux parties principales permettant une programmation rapide, en particulier dans les applioations en temps réel. Cette version sur silicium est entièrement compatible avec les programmes M/DOS 6502 écrits sous la version précédente. Néanmoins, nous attirons l'attention du lecteur sur le fait que cette nouvelle version possède de nouvelles possibilités.
Il faut noter qu'il y a une légère modification quant à l'utilisation de la page 3 de votre APPLE.
ATTENTION : les fonctions SCROLL UP et SCROLL DOWN ont vu leurs adresses modifiées (voir chapitre X : Les utilitaires).

1 - 1 - LE DISK OPERATING SYSTEM :

Il comprend les options suivantes :

- fichiers séquentiels
- fichiers relatifs
- fichiers à accés par clés
- fichiers multiclés

Quelle que soit l'option choisie, aucune modification de variables n'est nécessaire, celle-ci étant faite à la création du fichier, et enregistrée dans un DICTIONNAIRE associé. Le DOS comprend en outre la notion de SOUS-ARTICLES. Les enregistrements sont donc de taille variable.
La gestion du disque est entièrement dynamique et aucun dimensionnement de fichier ou d'enregistrement n'est nécessaire.

1 - 2 - LA GESTION DES MASQUES DE SAISIE ET D'IMPRESSION

Cet utilitaire permet de résoudre de façon rapide et souple tous les problèmes de saisie de données et d'impression. Tous les contrôles de validité sont efectués lors de l'utilisation. Les différents types de variables que l'on utilise sont compatibles avec ceux de la gestion de fichier. Un masque est composé de deux parties :

- un texte dont les caractères répétés sont codés pour gagner de la place, aussi bien sur disque qu'en mémoire centrale au moment de l'utilisation.
- un ensemble de fenêtres de saisie. Chaque fenêtre est associée à une variable BASIC et à un ensemble de contrôles à effectuer.

Pour faciliter la saisie, un certain nombre de touches de fonction ont été créées, facilitant la gestion de l'écran.

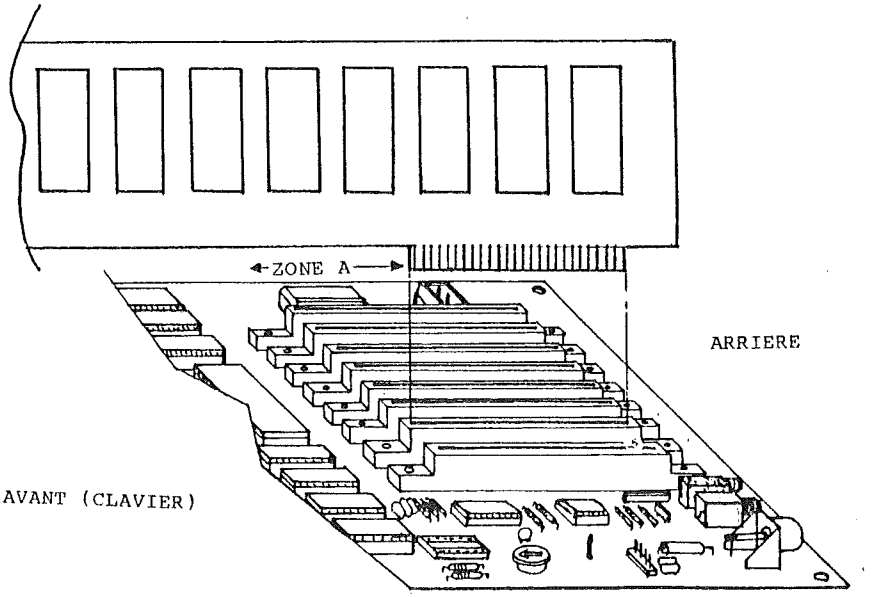
CHAPITRE II

MISE EN ROUTE DU M/DOS 6502

2 - 1 MISE EN PLACE D'UNE CARTE M/DOS

Avant toute manipulation, éteindre votre appareil en basculant votre interrupteur marche - arrêt sur la position arrêt et retirer votre cordon secteur de la prise de courant.

Ouvrez votre appareil. Sur le bac inférieur, vous voyez au fond à droite, un ensemble de huit connecteurs. La carte M/DOS est dissymétrique par rapport à l'axe du connecteur.



La zone A, c'est à dire le coté où la carte est plus longue est toujours dirigé vers l'avant de votre appareil. En regardant la plaque microprocesseur, on voit immédiatement que les connecteurs sur lesquels vous allez fixer votre carte sont numérotés de 0 à 7, le slot 0 étant le plus à gauche, le slot 7 étant le plus à droite. Le n° est imprimé sur le bord de le carte microprocesseur. Ces connecteurs sont aussi appelés slots. Choisissez un des slots entre le numéro 1 et le numéro 7. Fixer la carte dans le slot choisi (Par ex., le 3).

ATTENTION : Ne jamais toucher les surfaces dorées des connecteurs. Ces connecteurs sont de haute qualité et de haute précision. Tout dépôt gras risque d'en compromettre sérieusement le fonctionnement.
Au cas où un connecteur serait encrassé, nettoyer doucement avec un chiffon imbibé d'alcool à bruler ou avec un produit spécial contact.

Ne jamais exercer d'effort susceptible de déformer les contacts.

Ne mettez jamais votre carte en flexion, vous risqueriez de couper les pistes.

Vous vous assurerez ainsi une installation parfaite et fiable.

2 - 2 MISE EN ROUTE DU M/DOS 6502 SUR DISQUETTE APPLE

Pour mettre en oeuvre le M/DOS 6502, introduisez la disquette système dans le lecteur " Ø " (le DRIVE 1 du controleur sur le port 6) et fermez la porte du lecteur.
Si votre appareil est muni d'une ROM AUTOSTART, mettez la tension sur votre micro-ordinateur et le système se chargera automatiquement. Dans le cas contraire, faites :

```
RESET
6 CTRL/P      RETURN
```

Le système se chargera de lui-même et le message suivant apparait :

```
M/DOS 6502 - V4
MICRO INFORMATIQUE SERVICE
P. LAFFITTE
P. NESNIDAL
```

2 - 3 FORMATTAGE D'UNE DISQUETTE

La disquette système qui vous est fournie avec le manuel est une disquette spéciale qui contient les utilitaires du système M/DOS 6502. Si vous avez deux drives, nous vous conseillons tout d'abord de faire une copie de cette disquette. Pour arriver à recréer une nouvelle disquette, il faut prendre une disquette vierge et pré-inscrire dessus les valeurs nécessaires pour que le système puisse retrouver les pistes et les secteurs. Cette opération s'appelle le formattage.

formattage d'une disquette de type système :

Pour cela, utiliser " l'utilitaire BOOT ".
Voir chapitre XI : utilisation de ce programme.

formattage d'une disquette de type fichier :

Cette disquette ne permet pas de démarrer le système. Utiliser l'instruction : LET"#FORMAT,(numéro du drive)".

Cette instruction de formattage est valable quel que soit le disque ou la disquette implantée correctement sous M/DOS 6502. Il n'en va pas de même pour la création de disquettes permettant de démarrer le système. Pour rendre une disquette " autostart " se référer à la notice d'implantation fournie en général par le constructeur de l'unité de disques.

L'utilitaire BOOT permet d'obtenir une disquette qui pourra faire démarrer le M/DOS 6502, mais qui ne contiendra aucun programme . Si vous voulez obtenir une disquette avec les utilitaires MASTER dessus, il vous suffit de faire une copie de la disquette MASTER fournie par M.I.S.

CHAPITRE III

LE DISK OPERATING SYSTEM

3 - 1 - LES FICHIERS

3 - 1 - a) STRUCTURE DES ENREGISTREMENTS

La structure des fichiers du M/DOS 6502 a été conçuede façon à simplifier la programmation, et surtout à limiter le temps d'exécution du programme BASIC.

Un fichier est un ensemble d'enregistrements. Le nombre d'enregistrements par fichier ne peut dépasser 64 000. La taille du fichier est gérée dynamiquement et il est inutile de définir celle-ci à la création d'un fichier.

Chaque enregistrement d'un fichier est un ensemble de variables BASIC.
Ces variables peuvent être de tous types (à quelques restrictions près pour les clés des fichiers à accès par clé - voir fichier séquentiel indexé chapitre 3 - 2 -c -)

Rappelons les différents types de variables du BASIC :

- les flottants (5 octets) notes XY
- les entiers (2 octets) notes XY%
- les chaines de caractères de longueur 0 à 255 notées XY$

avec les conventions suivantes :

- le nom comporte 1 ou2 caractères X, Y
- le premier X est une lettre
- le second Y (facultatif) est une lettre ou un chiffre

De plus, des tableaux peuvent être définis pour chacun de ces types de variables. Les dimensions et le nombre de dimensions de ces tableaux ne sont limités que par la taille de la mémoire centrale de l'ordinateur.

Exemple : TA$(12,2,1)

Les indices des tableaux commencent à 0.

Dans le DOS, 2 nouveaux types de variables ont été créés :

- les dates notées XY$* dans les ordres M/DOS 6502
- les binaires notés XY%/ dans les ordres M/DOS 6502

Afin de réduire la place occupée par un enregistrement sur le disque, les valeurs numériques sont codées en binaire, les chaines de caractères en format variable et les tableaux sous forme de matrices creuses.
Il en résulte que si un tableau ne comporte que3 éléments non nuls, seuls ceux-ci seront sauvés sur le disque.

3 - 1 - b) CONTROLE DES ERREURS

ERREURS RECOUVRABLES

Après chaque ordre du DOS, il est possible de vérifier que l'ordre a bien été exécuté. Pour cela, un octet est réservé (189 sur ITT/APPLE). Cette variable sera appeléeSTATUS dans le reste de cette documentation.

Si PEEK (189) = Ø : tout s'est bien passé.
Sinon, une erreur est apparue lors de l'exécution de l'ordre. Dans ce cas, PEEK (189) est le code de l'erreur.

Une erreur de ce type n'interrompt pas le programme.

En général, il n'y a pour chaque ordre qu'un seul type d'erreur recouvrable. Il suffit donc de vérifier que le STATUS est nul pour s'assurer que l'ordre a été exécuté normalement.

Exemple : Lecture d'un enregistrement

PEEK (189) > Ø l'enregistrement n'existe pas.

CODES ERREURS :	
1	erreur sur masque (abandon...)
10	inexistant
20	priorité trop faible (lecture)
30	existant (en création)
255	fin de fichier

En mode direct, pour éviter au programmeur de tester le status après chaque commande, le message " DIRECT ERROR " apparait, si le STATUS est non nul.

ERREURS NON·RECOUVRABLES

Ces erreurs (qui interrompent le programme) peuvent avoir deux causes :

- Erreur de programmation
  - SYNTAX ERROR
    syntaxe d'un ordre impossible.
  - ILLEGAL QUANTITY ERROR
    erreur sur les numéros logiques (déjà utilisé, non ouvert...)

- BAD SUBSCRIPT ERROR
  tentative de lecture d'un tableau dans un enregistrement avec un dimensionnement inférieur à celui utilisé lors de sa création.

- OUT OF MEMORY ERROR
  plus de buffers libres (pour résoudre ce problème, voir le chapitre 8 POINTEURS DU DOS).

ERREURS SYSTEME

- TA ERROR
  disque plein (Track Allocation Error)

- DATA ERROR
  erreur de lecture ou d'écriture sur le disque. Cette erreur signale un mauvais fonctionnement du support magnétique. Attention, cette erreur peut provenir d'une protection d'écriture du support.

- FUNCTION ERROR
  erreur détectée dans un contrôle dû à une incohérence des paramètres dans un accès disque.

- NG ERROR
  erreur signalant une incohérence des données en mémoire (faire : LET"#C,Ø" par programme pour supprimer l'erreur).

3 - 1 - c) FERMETURE DE FICHIERS

Aucune fermeture n'est nécessaire. A la fin de chaque opération, tous les paramètres nécessaires au fonctionnement du DOS et ayant été modifiés, sont sauvés sur le disque.

La fermeture, qui en général réalise cette opération, mais seulement en fin de traitement, est donc inutile.

En cas d'arrêt imprévu du programme, seule l'opération en cours peut être perdue.

Il est néanmoins possible de récupérer la place occupés en mémoire centrale par l'ouverture d'un fichier. Pour cela, utiliser l'ordre CLEAR.

Il est possible de récupérer la place :

- d'un module particulier (masque ou fichier) désigné par son numéro logique. voir 5 - 1 - b)
- de récupérer toute la place . voir 5 - 1 - c)

Après l'ordre CLEAR d'un fichier, il est nécessaire de le ré-ouvrir pour pouvoir accéder à ses enregistrements.

3 - 2 - LES DIFFERENTS TYPES DE FICHIERS

3 - 2 - a) FICHIERS SEQUENTIELS RELATIFS

Ces fichiers peuvent être manipulés aussi bien en séquentiel qu'en relatif, aussi bien en écriture qu'en lecture.
Il n'y a qu'un seul mode d'ouverture qui permet l'ensemble des opérations.
A la création du fichier, on définit une variable entière BASIC qui sera utilisée comme pointeur vers l'enregistrement.

Un enregistrement d'un fichier séquentiel comprend donc :

- le pointeur XX% permettant d'indiquer son numéro d'ordre.
- la liste des variables.

Remarque : le pointeur est une zone fictive de l'enregistrement et est en réalité le résultat d'un calcul.

La liste des ordres possibles est :

- Opérations " dites " séquentielles

Write = écriture d'un nouvel enregistrement en fin de fichier.
Le numéro d'ordre de l'enregistrement créé est renvoyé dans le pointeur.

Next = lecture d'un enregistrement en séquentiel.
Permet de lire l'enregistrement suivant.

Borne = limite le fichier en imposant un numéro d'ordre maximum.
Une fois que le numéro atteint cette valeur, une fin de fichier sera signalée.

Xindex = renvoie dans le pointeur le numéro du prochain enregistrement qui sera créé par Write.

- Opérations " dites " relatives

Update = modification d'un article préalablement créé par l'ordre Write.

Read = lecture d'un article en fonction de son numéro d'ordre dans le fichier.

Delet = destruction d'un article dont on indique le numéro d'ordre.

Remarque : La tentative de lecture d'un enregistrement détruit n'aboutira pas.
La possibilité de détruire des enregistrements permet de récupérer la place qu'ils occupaient sur le disque.
Néanmoins, la destruction d'un enregistrement ne modifie pas la numérotation des autres.
Les enregistrements détruits ne seront pas lus par l'ordre NEXT.

Tous les ordres peuvent être utilisés dans n'importe quel ordre. En particulier, on pourra accéder directement à un article par son numéro par READ, lire le fichier séquentiellement à partir de cet enregistrement par NEXT. (Par défaut, NEXT après l'ouverture donne le premier enregistrement du fichier)

3 - 2 - b) FICHIERS SEQUENTIELS INDEXES

Contrairement aux fichiers relatifs (voir 3 - 3 -b), le moyen d'accès à un enregistrement n'est plus le numéro d'ordre, mais une CLE plus complexe.

Cette clé est définie comme un ensemble de variables BASIC, et il devient possible d'accèder à un enregistrement directement par la valeur de sa clé.

Prenons l'exemple d'un fichier de personnes.
La clé pourra être :

NO$ = le nom
DN$:= la date de naissance
PR$ = le prénom

C'est l'ensemble de ces variables qui composera la clé.

Contraintes sur les clés d'un fichier séquentiel indexé :

La clé doit être de longueur fixe. Il en résulte que les tableaux ne pourront pas être utilisés. De plus, pour les chaines de caractères, il faudra indiquer une longueur maximum possible.
Dans l'exemple ci-dessus, il faudrait préciser la longueur admise pour les deux variables caractères.
Par exemple, NO$ 13,PR$ 8

Les opérations possibles sont :

En séquentiel

Next = lecture de l'enregistrement suivant dans l'ordre des clés.

Borne = fixe une borne maximum au fichier

Xtract = permet de sélectionner des enregistrements suivant un critère d'égalité sur une des variables clé.

En accès par clé

Write = écriture d'un nouvel article (refuse les homonymes)

Add = écriture d'un nouvel article (accepte les homonymes)

Read = lecture d'un enregistrement

Update = mise à jour d'un enregistrement

Delet = destruction d'un enregistrement

3 - 2 - c) FICHIERS MULTICLES

Les fichiers multiclés du M/DOS sont une extension des fichiers séquentiels indéxés. Un fichier multiclé est un fichier comportant plusieurs groupes de clés, chaque groupe est appelé un MOYEN D'ACCES au fichier. Le nombre maximum de moyens d'accès par fichier est 10. Chaque moyen d'accès est un ensemble de variables formant une clé, identique à la clé unique d'un fichier séquentiel indexé.

Contraintes sur les clés d'un fichier multiclé

Les contraintes sur les clés sont les mêmes que les fichiers séquentiels indexés (3 - 2 - b)

Les opérations possibles sont :

Elles sont identiques à celles des fichiers séquentiels indexés (3 - 2 -b). Cependant, elles nécessitent des paramètres supplémentaires qui sont décrits au paragraphe 3 - 3 - d.

Il est impossible de modifier les clés des autres moyens d'accès.
En lecture, les clés des autres moyens d'accès ne sont pas renvoyées.
Si cette information est nécessaire, répéter dans l'enregistrement les variables clés à connaitre.
La réorganisation du fichier se fera sur l'ensemble des moyens d'accès.

3 - 3 - LA CREATION DE FICHIERS

3 - 3 - a) SYNTAXE GENERALE

La création d'un fichier se fait en deux étapes :

- définition de l'enregistrement
- création du fichier

Pour définir l'enregistrement, utiliser l'ordre >
La syntaxe est la suivante :

LET " > vc1,vc2,...,vcn = ve1,ve2.....,vep

avec vcl..n : variables composant la clé
et vel..p : variables composant l'enregistrement

Dans la clé, les variables sont de longueur fixe. Les possibilités sont :

XX$NNN	..... variable alpha de longueur NNN = 255	(exemple AL$25)
XX	..... flottant	(exemple FL)
XX%	..... entier de -32767 à +32768	(exemple EN%)
XX%/	..... binaire de Ø à 255	(exemple BI%/)
XX$*	..... date format E/S jj/mm/aa, code 2 octets	(exemple DA$*)

En revanche, dans l'enregistrement, les variables sont de longueur non fixe et peuvent être des tableaux BASIC.

Lorsque la variable est un tableau, indiquer le signe " ; " après le nom.

Exemple : AB est un flottant simple
AB ; est un tableau de flottants

Lorsqu'une variable est définie comme tableau, il n'est pas nécessaire d'en indiquer la dimension. En effet, les tableaux sont enregistrés sur disque sous forme de tableaux creux de dimension quelconque (en fait, limitée par la dimension de la mémoire centrale de l'ordinateur). La dimension du tableau sera faite à l'ouverture du fichier. (voir chapitre 4)

Les variables alphanumériques sont, contrairement au cas des variables clé, de longueur variable de Ø (chaine vide) à 255 caractères. Il est donc inutile d'indiquer leur taille.

La liste des variables possibles est donc :

XX	Flottant	XX ;	Tableau de flottants
XX$	Chaine de caractères	XX$;	Tableau de chaines de caractères
XX%	Entier -32767 32768	XX%/ ;	Tableau d'entiers
XX%/	Binaire 0 255	XX%/ ;	Tableau de binaires
XX$*	Date	XX$* ;	Tableau de dates

Lorsque la définition de l'enregistrement a été faite, il ne reste plus qu'à créer le fichier. L'ordre est NEW.

LET " # NEW, [no logique],FICHIER,[no disque] : [nom du fichier]

A la création d'un fichier, le système détermine automatiquement un coefficient de blocage. Celui-ci définit la taille du bloc de base. Le bloc de base sera la plus petite unité de mémoire disque allouée à un enregistrement. Pour savoir comment le système détermine ce coefficient de blocage, reportez-vous au chapitre
Un ordre permet d'imposer un maximum à cette valeur.
Les valeurs possibles sont :

32 = 1/8 secteur
64 = 1/4 secteur
128 = 1/2 secteur
ou 256 = 1 secteur

L'ordre LET"# > n" imposera au coefficient choisi par le système de ne pas dépasser n.

EXEMPLE : LET"# > 64"

même si le système estime la taille idéale à 128 ou 256, il choisira 64. S'il trouve 32, il restera à 32.

3 - 3 - b) FICHIERS RELATIFS

Pour les fichiers relatifs, la clé sera remplacée par le pointeur vers le numéro d'enregistrement? Ce pointeur sera un entier XX%. Pour indiquer que le fichier est relatif, précéder le nom de ce pointeur par : "@ ".

La syntaxe générale est : LET" >@XX% = vel ... vep

EXEMPLE : Création d'un fichier des ventes

Le fichier est un fichier relatif. Le pointeur sera NV%(numéro de la vente). L'enregistrement pourra comprendre :

- la date DA$ de la vente
- le tableau des références pièces vendues :
  - FA% ; numéro de famille des pièces
  - FO% ; numéro du fournisseur
  - NP% ; numéro des pièces
  - PV ; prix de vente

Le programme de création sera :

```
100 LET"#C,$" : S = 189
110 LET" >@NV% = DA$,FA%;,NP%;,PV;
120 LET"#NEW-6,FICHIER,Ø : VENTES" : IF PEEK (S) THEN ? "ERREUR"
130 END
```

3 - 3 - c) FICHIERS SEQUENTIELS INDEXES

La syntaxe est la suivante :

LET " vc1,vc2,...,vcn = ve1,ve2.....,vep

avec vcl ..n : variables composant la clé
et vel ..n : variables composant l'enregistrement

EXEMPLES :

1) Création d'un fichier stock

Le fichier est un fichier à accès par clé.
La clé est composée de 3 éléments.

FA% = numéro de famille de la pièce
FO% = numéro du fournisseur
NP% = numéro de pièce

L'enregistrement comprend :

PA = prix d'achat
PV = prix de vente
TV%/ = taux de TVA
LI$ = libellé de l'article
QS = quantité en stock
QC = quantité en commande
QM = quantité minimum
QX = quantité critique
UN$ = unité

Pour créer ce fichier, le programme à écrire sera :

```
100 LET"#C,$" : S = 189
110 LET" > FA%,FO%,NP% = PA,PV,TV%/,LI$,QS,QC,QM,QX,UN$
120 LET"#NEW,1,FICH,1 : STOCK" : IF PEEK (S) THEN ? "ERREUR"
130 END
```

Pour s'assurer que tout s'est passé normalement, vérifier que le STATUS est nul.

REMARQUE IMPORTANTE : Si une erreur de syntaxe est détectée dans la définition de l'enregistrement en ligne 110, l'erreur indiquera la ligne du NEW.
Dans cette exemple, ? SYNTAX ERROR IN 120

3 - 3 - d) FICHIERS MULTICLES

La syntaxe de création d'un tel fichier diffère légèrement de la syntaxe classique. Elle sera :

LET" vmal1,...,vmaln & vma21,...,vma2n & ... = vel,ve2...vep

On indique dans la liste des variables de chaque moyen d'accés, les différents moyens d'accès étant séparés par " & " au lieu de " , ".

EXEMPLE : Création d'un fichier Sécurité Sociale :

- clé 1 = nom NO$
- clé 2 = numéro de SS NS$

L'enregistrement comprend :

NO$: le nom
NS$: le numéro de sécurité sociale
AD$: l'adresse
DN$: la date de naissance
SM$: la situation de famille

```
100 LET"#C,$" : S = 189
110 LET > NO$15 & NS$ = AD$,NO$,NS$,DN$:,SM$
120 LET"# NEW,1,FICHIER,Ø : SECU": IF PEEK (S) THEN ? "ERREUR"
```

Pour tous les ordres WRITE et ADD, il faut indiquer l'ensemble des clés (de tous les moyens d'accès). Pout tous les ordres READ, NEXT, UPDATE, BORNE, XTRACT, et DELET, il faut indiquer dans l'ordre un paramètre supplémentaire indiquant le moyen d'accès utilisé. Par défaut, ce paramètre vaut 1 (1er moyen d'accès).

EXEMPLE : Avec le fichier que nous venons de créer :

Création :
```
100 NO$ = "LEROI" : NS$ = "1521114254368
110 LET"WRITE-1
```

Lecture d'après le nom :
```
200 NO$ ="LEROI
210 LET"READ-1
```

Lecture d'après le numéro de SS :
```
300 NS$ = "1521114254368
310 LET"READ-1,2
```

Modification :
```
400 AD$ = " 10, RUE JEAN JAURES
410 LET"UPDATE-1,2
```

3 - 4 - L'OUVERTURE D'UN FICHIER

L'ordre à utiliser est OPEN avec la même syntaxe que NEW. Cependant, il est inutile d'indiquer une seconde fois la liste des variables. Au contraire, celle-ci pourra être affichée par les ordres suivants :

```
LET"# Open-[n],FICHIER,[d] : [nom]
LET" ENTER-[n]
LET" Visualise
```

Dans l'exemple 1) (fichier stock), le texte suivant sera imprimé :

```
FA%,FO%,NA%
=
PA,PV,TV%/,LI$,QS,QC,QM,QX,UN$
```

Dans le cas où l'enregistrement contient des tableaux, ceux-ci devront être dimensionnés par l'utilisateur avant l'ouverture du fichier. Sinon, leur dimension par défaut sera 8. Cette valeur par défaut peut être modifiée, indiquer pour cela le nombre d'indices désirés en 770. (Voir chapitre Pointeurs du DOS)

Exemple :

```
]DIM A$(10,3)
]LET"#0,1,F, TEST"
]LET"E,1"
]LET"V"
X,Y
=
A,A$;
```

3 - 5 - COPIE DE FICHIERS

Pour recopier un fichier sur un nouveau, il faut pouvoir créer le deuxième fichier avec la même structure de variables que le premier. Pour cela, l'ordre ENTER décrit ci-dessus peut remplacer l'ordre de description d'enregistrement.

Exemple : Création du fichier BBB de même structure que le fichier AAA

```
LET"#OPEN,1,Fi,Ø : AAA
LET" ENTER,1
LET"#NEW,2,Fi,Ø : BBB
```

L'ordre ENTER lit le dictionnaire du 1er fichier qui servira pour créer le second.

CHAPITRE IV

LES MASQUES DE SAISIE

4 - 1 - STRUCTURE D'UN MASQUE

4 - 1 - a) LE TEXTE

A la création d'un masque, le texte est entré librement sur l'écran (voir création de masque page 16)

Les fenêtres sont limitées par les caractères " < " et " > " qui indiquent l'emplacement ou s'effectuera la saisie.

Les noms des variables à saisir seront indiqués entre à un endroit quelconque du masque.

A la fin de cette étape, le masque est analysé, puis compact..

Le volume occupé par le masque peut être calculé de la façon suivante :

- les caractères isolés occupent 1 octet chacun.
- chaque suite de caractères identiques de longueur comprise entre 3 et 255 occupe 2 octets.
- les fenêtres et les variables sont considérées comme des blancs.

De plus, 7 octets seront alloués pour chaque fenêtre de saisie.

4 - 1 - b) LES ZONES DE SAISIE :

A chaque fenêtre est associée une variable BASIC et un type.

Les types de variables sont les mêmes que dans le cas du DOS.

C'est-à-dire :

- ALPHANUMERIQUE
- FLOTTANT
- ENTIER
- BINAIRE
- DATE

Chaque type pouvant ou non être un tableau à une dimension.

Les contrôles au moment de la saisie sont de 2 types.

- contrôle au niveau du caractère
  (numérique, entier, ...)

- contrôle au niveau de la validation de zone
  (entier < 32768, dates correctes, ...)

Les contrôles que l'on peut demander sont :

- POSITIF ..... FLOTTANT ou ENTIER
- ENTIER ..... FLOTTANT

D'autres contrôles sont implicites :

- ENTIER ..... ENTIER ou BINAIRE
- VECTORISE ou NON . selon le type défini
- DATE ..... si défini comme date
  (la validité des dates est controlée, y compris années bissextiles)

- NUMERIQUE ..... FLOTTANT, ENTIER ou BINAIRE
- 0 <= X < 256 ..... BINAIRE
- -32767 <X < 32768. ENTIER
- moins de 32 chiffres FLOTTANT

Les zones pourront en outre être définies en ENTREE ou en SORTIE sur le masque.

Une zone en sortie ne sera pas demandée au moment d'une saisie.

4 - 1 - c) CARACTERE TRANSPARENT :

Le caractère " @ " sur fond blanc est considéré comme transparent.
Cela signifie que lors du chargement sur l'écran du texte du masque, les caractères déjà présents sur l'écran resteront.

Exemple :
Si vous indiquez en haut et à droite de l'écran la date du jour, et que vous désirez la conserver tout au long du programme, il suffit de mettre dans tous vos masques des caractères @ sur fond blanc à l'emplacement de cette date.

4 - 2 - UTILISATION

Un masque sert à la fois à saisir des données et à les imprimer sur l'écran.

L'impression peut être faite :

- sur l'ensemble des variables
- sur les variables définies en entrée
- sur les variables définies en sortie

En aucun cas, l'impression ne débordera des fenêtres. Si la place définie dans une fenêtre est insuffisante pour écrire la variable (ou une des variables dans le cas d'un vecteur), la zone se terminera par le caractère " %".

De même que dans le cas d'un fichier, l'ensemble des variables contenues dans le masque constitue un DICTIONNAIRE et aucune liste de variables n'est nécessaire.

Dans le cas de la saisie d'informations sur le masque, l'utilisateur aura la possibilité de passer de fenêtre en fenêtre par la touche RETURN.

A ce moment, on contrôle la validité de la saisie. La zone est transférée dans la variable BASIC correspondante et réinscrite de façon standard.

La touche de fonction CTRL/R renvoie le spot dans la fenêtre précédente.

Les fonctions de gestion d'écran (voir page suivante) seront possibles, dans les limites de la fenêtre .

Remarque : Si aucune modification n'est faite dans la fenêtre, RETURN ne réalise que le passage à la zone suivante.

Le fait de faire RETURN sur la dernière fenêtre peut ou non provoquer la fin de la saisie.
Dans ce dernier cas, cette action renvoie à la première fenêtre définie en entrée
De même, le retour à la fenêtre précédente sur la première fenêtre envoie à la dernière.

La touche ESC permet de valider l'ensemble des informations. Son utilisation est obligatoire pour valider la saisie si l'on a pas demandé une sortie automatique en fin de masque.
Dans le cas contraire, son utilisation est facultative mais évite de faire RETURN jusqu'à la fin du masque.

Si l'on désire abandonner, enfoncer la touche CTRL/A.
Dans ce cas, la zone en cours n'est pas transmise au BASIC.

La variable STATUS permet de savoir quel a été le mode de sortie du masque.

0..... ok
1..... abandon

4 - 3 - GESTION DE L'ECRAN

Les commandes sont de deux types :

- traitement d'une zone
- traitement des caractères

4 - 3 - a) VALIDATION D'UNE ZONE

CTRL/A ..... abandon (STATUS rendu à 1)
ESCAPE ..... fin normale

En utilisation de masque :

RETURN ..... passage à la zone suivante
CTRL/R ..... retour zone précédente

En création de masque :

RETURN ..... descente du spot
CTRL/R ..... remontée du spot

4 - 3 - b) TRAITEMENT DU CARACTERE (dans tous les cas)

CTRL/B ..... caractères fond blanc
CTRL/N ..... caractères normaux
CTRL/F ..... caractères clignotants
→ ..... déplacement du spot à droite
← ..... déplacement du spot à gauche
CTRL/Q ..... déplacement du spot vers le haut
CTRL/Z ..... déplacement du spot vers le bas
CTRL/@ ..... effacement de la zone
CTRL/I ..... insertion d'un caractère
CTRL/D ..... destruction d'un caractère
CTRL/T ..... retour au début de la zone
CTRL/V ..... répétition verticale du caractère

4 - 4 - CREATION DE MASQUES

Pour créer un masque utiliser l'ordre :

LET" #NEW,(n°logique), Masque,(n° drive) : (nom)

Cet ordre vous donnera la main pour entrer le texte.

Lorsque celui-ci est tapé, faire ESCAPE pour rendre la main au programme.

La création de masque peut être faite soit en mode direct, soit en mode programme.

Le programme ne détruit pas le contenu actuel de l'écran. Deux possibilités s'offrent alors :

a) Nouveau masque

Il suffit d'effacer l'écran :

Exemple :

```
100 HOME : LET" #NEW,1,MASQUE,1 : TEST-MASK
```

b) Masque issu d'un précédent (éventuellement modification)

Il suffit de charger le masque précédent.

Exemple :

```
100 LET"#OPEN-A,M,1 : TEST-MASK " : REM OUVRE LE MASQUE
110 LET"V,A" : REM AFFICHE LE MASQUE
120 LET"#NEW-B,M,0 : MASQUE 1 : REM CREE LE NOUVEAU MASQUE
```

Pour indiquer les fenêtres, il suffit d'écrire sur l'écran à l'endroit désiré les caractères "< " et " >". Le premier indique le début de la fenêtre, le second la fin. Le nom de la variable sera inscrit entre " ".

Si la fenêtre ne fait qu' 1 caractère, écrire seulement "> ".
Le nom pourra être écrit soit entre les <..>, soit en dehors. Il suffit que l'ordre des variables soit le même que celui des fenêtres.

Les contrôles à effectuer seront indiqués à la suite du nom.

```
XX ..... flottant
XX% ..... entier
XX%/ ..... binaire
XX+ ..... flottant positif
XX%+ ..... entier positif
XX@ ..... flottant entier
XX@: ..... flottant entier cadre à droite
XX :..... flottant format gestion
XX% :..... entier cadré à droite
XX $..... chaine de caractères
XX$: ..... date
XX$+ ..... chaine de caractères considérée comme numérique avec affichage
           format gestion
```

ajouter :

```
? ..... si la zone est en sortie
: ..... si la zone est en enchainement automatique
; (n) ..... si la zone est un indice de tableau
```

(n) = numéro de l'indice (99 pour liste)

Remarque :

Pour une zone en sortie, il est inutile d'indiquer les contrôles.

Les contrôles sont cumulables.

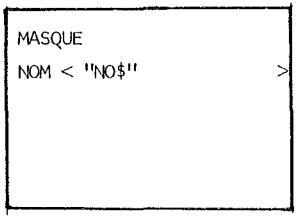
En cas d'erreur sur le masque, (par exemple nombre de variables ≠ du nombre de fenêtres) le système n'acceptera pas votre masque et vous rendra la main pour le corriger, après vous l'avoir signalé par une cloche.

4 - 5 - CREATION D'UN MASQUE DE FACON AUTOMATIQUE

Pour générer un masque par programme, il suffit de précéder le numéro du disque par le caractère /. Dans ces conditions le programme de création ne donnera pas la main à l'utilisateur pour modifier le contenu de l'écran avec validation par ESC. Au contraire, le programme créera un masque dont le contenu sera la reproduction de l'écran au moment de l'exécution de l'ordre.

Par exemple :

```
10 HOME : PRINT"MASQUE.#": PRINT
20 PRINT"NOM <"CHR $(34) NO $"CHR $(34) " >
30 LET" #NEW,3,M,/1 : AUTOMASQUE
40 REM CHR $(34) = "
```



Le masque ci-dessus sera enregistré sur le disque sous le nom : "AUTOMASQUE "

En cas d'erreur sur le masque, le programme rend la main en indiquant une erreur et sans sauver le masque (STATUS = 1)

4 - 6 - EXEMPLE

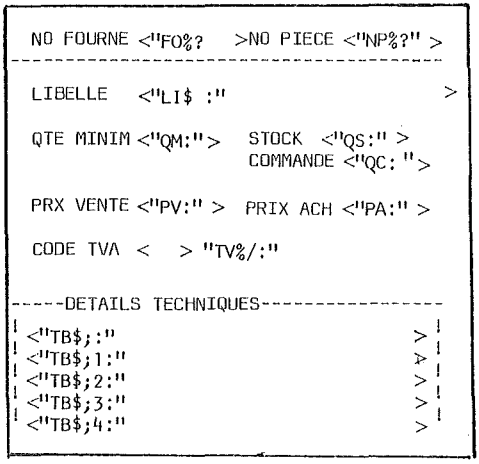
Faire l'ordre :

LET"# CL,$ " : LET "# N,1,M,@@ : ART ← STOCK

(fermeture générale puis création d'un masque, éventuellement en modification)

Entrez sur l'écran le texte et les fenêtres. Par exemple :

MASQUE ART-STOCK



4 - 7 - IMPRESSIONS PARAMETREES

Il est possible de paramètrer des impressions par l'utilisation de masques.
L'ordre permettant cette fonction est LET "?-(c)
(c) étant un caractère de contrôle permettant de répéter la ligne à imprimer sur l'écran.
Les caractères de contrôle doivent être inscrits sur fond blanc, l'un en début de ligne à imprimer, l'autre à la fin de cette ligne.
Ces caractères ne seront bien sûr, pas imprimés.

L'ordre LET"?-(c) imprimera les caractères de l'écran séparés par des caractères (c), puis génèrera un passage à la ligne :

Exemple : l'écran contient :

IMPRESSION [A] TEST [A]

LET"?-A" imprimera :

TEST sur l'imprimante désirée.

Il est nécessaire avant d'exécuter l'ordre LET"?-(c)" de définir l'imprimante sur laquelle on désire l'impression. Pour cela, utiliser les ordres :

```
PR#[n° imprimante]
PRINT CHR$ (9)"80N"
```

(cette dernière commande évite que l'impression se fasse également sur l'écran, évitant ainsi de détruire le contenu de ce dernier.

Pour revenir au mode normal d'impression, faire :

```
PR#Ø
```

Cas particulier : (HARD COPY)

Les caractères de contrôle que l'on doit utiliser sont les lettres de l'alphabet A, B, C,... Z
Nénamoins, le caractère de contrôle ^ joue un rôle particulier. L'ordre]LET"?-^" réalise la recopie complète de l'écran sur l'imprimante.
De même que dans le cas de l'impression d'une ligne, il faut définir l'imprimante choisie et protèger l'écran.
L'ensemble des ordres réalisant le HARD COPY sera donc :

```
]PR#S : PRINT CHR$ (9)"80N" : LET"?-^" : PR#O
```

Utilisation de masque :

Définir par exemple le masque EDIT suivant :

[A]ARTICLE < "AR" > ACHAT <"PA" > VENTE < "PV" > [A]

Pour lister un catalogue de prix. L'enregistrement du fichier stock contient les variables AR (numéro d'articles), PA (prix d'achat) et PV (prix de vente).
L'impression d'une ligne sera :

```
100 REM LA PIECE A ETE LUE SUR LE FICHIER
110 LET"OUTPUT-M" : REM AFFICHAGE SUR LE MASQUE
120 LET"?-A" : REM EDITION SUR IMPRIMANTE
```

Il peut y avoir plusieurs textes différents sur le même écran, qui peuvent être utilisés simultanément si leurs séparateurs sont différents. (par exemple, un titre et la ligne courante)

Remarque : La ligne d'impression peut représenter plusieurs lignes de l'écran permettant ainsi de s'adapter à n'importe quelle imprimante (80 ou 132 colonnes)

De plus, cet ordre gère le nombre de lignes imprimées.
Le nombre de lignes s'obtient par PEEK (947). Cette valeur augmente de 1 à chaque exécution de l'ordre LET"?-(c). Lorsqu'elle atteint 66, elle repasse à 0. L'ordre LET"?" réalise la fonction " FORM FEED " (haut de page) en utilisant le compteur de ligne PEEK (947).

4 - 8 - TABLEAUX ET MASQUES

Il y a deux possibilités pour saisir (ou afficher) des tableaux dans les masques.

- saisie de l'ensemble du tableau dans une seule fenêtre.
- saisie d'un élément du tableau dons une fenêtre.

4 - 8 - a) LA SAISIE DU TABLEAU COMPLET

Cette saisie ne peut se faire que pour les tableaux à une dimension dont les indices 0 à 8 seront traités. (Si le tableau est de dimension supérieure, les valeurs correspondant aux indices supérieurs ne seront pas affectées par une saisie, si la longueur est inférieure, l'erreur BAD SUBSCRIPT ERROR IN 0 apparaitra.)

Pour saisir ou afficher un tableau de cette manière, indiquer dans le masque XX ; 99, XX étant le nom de la variable et les contrôles.
le nombre de variables du tableau saisies et affichées est 9.
Cette valeur peut être modifiée ; pour cela indiquer la valeur souhaitée en 770.

Exemple :

<"AC ; 99" >

A la saisie, les différents éléments du tableau doivent être séparés par " ; ".

Par exemple :

La saisie de : ;;12;13;;15 réalise :

```
AC (0) = 0                    AC (6) = 0
AC (1) = 0                    AC (7) = 0
AC (2) = 12                   AC (8) = 0
AC (3) = 13                   AC (9) = 0
AC (4) = 0                    AC (10) = 0
AC (5) = 15
```

A l'affichage, le tableau sera réécrit sous la même forme.

4 - 8 - b) LA SAISIE D'UNE SEULE ZONE D'UN TABLEAU

Pour cela, indiquer après le nom " ; " puis le numéro de l'indice.

Par exemple :

" AC ; 8 "

Dans ce cas l'indice peut varier de 0 à 98.(par défaut, la valeur de l'indice est 0 : " AC ; " équivaut à " AC ; 0 ")

Dans l'exemple ci-dessus, la saisie de : 145,réalise : AC(8) = 145 sans affecter les autres indices.

4 - 9 - PRINT USING

Les masques permettent, tant pour l'affichage sur l'écran que pour l'impression sur papier,un formattage des zones numériques.

Il y a trois possibilités d'impression des variables numériques :

- cadrage à gauche, format flottant
- cadrage à droite, format flottant
- cadrage à droite, format gestion

a) Cadrage gauche, format flottant

C'est le formattage pris par défaut dans les masques. Il correspond au format du BASIC.

123	123
10.0	10
12.25	12.25
10000	10000
99999999999	1E + 11
0	0

b) Cadrage droite, format flottant

Les valeurs sont cadrées à droite de la fenêtre de saisie, (permet de colonner des entiers).

123	123
10.0	10
12.2500	12.25
10000	10000
99999999999	1E+11

c) Cadrage droite format gestion

Ce format permet de colonner des valeurs décimales (par exemple des montants).

```
123             !          123.000!
10.0            !           10.000!
12.2500         !           12.250!
10000           !        10000.000!
99999999999     !1E+11             !
```

Lorsque le formattage gestion est impossible, le cadrage de la zone se fait à gauche, ce qui permet de la repérer immédiatement sur un LISTING.
Ce cas peut se produire lorsque :

- la valeur est sous forme exponentielle (ex 12E-12)
- le nombre de chiffres après le " . " est supérieur au nombre désiré.
- lorsque le nombre de chiffre est supérieur à la dimension de la fenêtre.

Une variable nulle ne sera pas imprimée, sauf s'il s'agit de l'indice 0 d'un tableau. Cela permet d'aérer le résultat de tableaux numériques.

Pour fixer le nombre de chiffres imprimés après le point, utiliser l'ordre :

LET"@- nombre de chiffres

Par défaut, au chargement du système, la valeur est 2.

d) Indication du cadrage dans un masque

La zone sera cadrée à droite si le nom de la variable est suivie de " % ".

Exemples :

```
"AC% "
"PO%% "
"EN%/% "
```

Si la zone est un entier ou un binaire, seul le cadrage à droite sera réalisé.

Si la zone est un flottant, le formattage gestion sera réalisé.

Pour cadrer à droite un flottant sans colonner les décimales, le déclarer entier par " @ ".

```
"FL% "   :  format gestion
"FL@%"   :  cadrage à droite
```

4 - 10 - LES MASQUES GLOBAUX

Ce type d'objet " G " représente un ensemble de masques sauvés sur le disque sous un seul nom. Les avantages de cette méthode sont multiples :

1) gain de place

Un masque isolé occupera sur le disque

- la place d'un nom dans le catalogue
- un nombre entier de secteurs du disque

Réunis, la place est optimisée.

2) gain de vitesse

Le chargement se fait en un seul accès au catalogue et avec un chargement minimum de blocs en mémoire.

3) raccourcissement des programmes

Un seul objet est à ouvrir en lieu et place d'une liste plus ou moins longue.

En revanche, la mise à jour des masques ainsi réunis est plus délicate. Il est donc conseillé de n'effectuer ce regroupement que lorsque votre logiciel est au point.

MISE EN OEUVRE :

Pour sauver sur le disque ces masques globaux, la méthode à employer est la suivante :

1) Ouvrir tous les masques concernés. Par exemple :

```
LET"# 0, 1, M, MASK 1
LET"# 0, 2, M, MASK 2
LET"# 0, 3, M, MASK 3
LET"# 0, 4, M, MASK 4
```

2) Exécuter ensuite l'ordre :

```
LET"# N, G, G, GLOBALMASK1234
```

Ceci aura pour effet :

- de supprimer de la mémoire tous les objets non concernés
  - drives
  - fichiers
- de sauver l'ensemble obtenu

Pour utiliser ultérieurement ces masques regroupés, ouvrez l'ensemble en une seule opération par :

```
LET"# 0, G, G, GLOBALMASK1234
```

Tous les masques sont alors accessibles avec le numéro logique qui leur a été donné à la création.

NOTE :

Vous pouvez charger en mémoire plusieurs globaux. Si ceux-ci contiennent des masques ayant été sauvés dans un même numéro logique, seul le premier chargé sera accessible tant qu'il n'aura pas été fermé par l'ordre :

```
LET"# C, numéro logique
```

EXEMPLE :

```
GLOBAL 1 regroupe 1, 2, 3
GLOBAL 2 regroupe 1, 5

LET"# 0, G, G, GLOBAL 1
LET"# à, G, G, GLOBAL 2
LET"C, 1"                concerne le " 1 " de GLOBAL 1
LET"# C, 1"
LET"C, 1"                concerne le " 1 " de GLOBAL 2
```

IMPORTANT : Le numéro logique indiqué dans la création ou ouverture de globaux n'a pas d'importance. Dans nos exemples, nous avons toujours choisi G. Cependant, le système n'acceptera pas le numéro logique déjà utilisé. Dans ce cas, utilisez un autre caractère.

Exemple : `LET"# 0, 2, G, GLOBAL`

4 - 11 - CREATION D'UN MASQUE EN MEMOIRE

LET"# Image, n° logique, M, d : nom

Création d'un masque sans que celui-ci ne soit sauvé sur le disque.
Cet ordre a deux intérêts :

- création de masques temporaires construits automatiquement
- utilisation de la gestion de masques en DOS 3.2 ou DOS 3.3
- fabrication directe de masques globaux

CHAPITRE V

LISTE DES ORDRES ET SYNTAXE

Les ordres sont envoyés dans une chaine de caractères par LET.

Il y a deux niveaux d'ordres :

- niveau global : ordres gérant des entités complètes (fichiers, masques)
- niveau action : ordres gérant un élément (enregistrement)

5 - 1 - NIVEAU GLOBAL

Les ordres sont précédés d'un " # ".

Let"# Open- n° logique ;[type d'ouv.] ; [n° disque] ; [nom]

Ouverture d'un fichier :

- n° logique : 1 caractère quelconque
  C'est ce numéro logique qui sera utilisé dans tous les ordres au " niveau action" concernant ce fichier.
- type d'ouverture
  - M : masque
  - F : fichiers
- n° du disque : 0, 1,....A, B, ...
- nom : 21 caractères significatifs.

LET"# Clear-[n° logique]

Récupère en mémoire centrale la place réservée à un fichier ou un masque.

LET"# Réorganise- [n° logique]

Réorganise la table d'accès d'un fichier à accès par clé. (Améliore le temps d'accès.)
Lorsque de nouvelles pièces sont créées dans le fichier, elles ne sont pas insérées directement à leur place exacte (dans l'ordre croissant des places) En effet, cette opération ralentirait énormément la création (surtout pour les gros fichiers i.e 10000 pièces.)
L'opération de réorganisation reclasse ces clés.

Les fichiers relatifs ne doivent pas être réorganisés.

LET"# Reorganise-$[n° de disque]

Réorganise le catalogue d'un disque. L'exécution de cet ordre permet d'accélérer les ordres LOAD, SAVE, OPEN, NEW.
En effet, le DOS permettant de gérer de nombreux objets sur disque, l'accès à un module se fait en séquentiel indexé (beaucoup plus rapide qu'une recherche séquentielle)
Il demande donc, comme un fichier de ce type, des réorganisations lorsque de nombreuses créations ont été faites. (SAVE, NEW)

LET"#New-[n° logique],[type],[drive] : [nom]

Création de masque ou de fichier.

MASQUES : Le type est Masque

Si le numéro du disque est précédé de @, le masque écrasera un éventuel masque déjà existant et portant le même nom.

Si le numéro du disque est précédé de /, le programme de création ne donne pas la main à l'utilisateur. (Le masque correspond alors au contenu de l'écran au moment de l'exécution de l'ordre.)

En cas d'erreur dans le masque, le programme rend la main pour correction (sauf en création automatique)

FICHIERS : Le type est Fichier

L'ordre NEW doit être précédé de l'ordre "> " ou de l'ordre ENTER.
" > " indique une description d'enregistrement.
ENTER reprend la description d'enregistrement d'un fichier déjà existant.
(voir création de fichiers)

LET"#Delet-[type] , [n° disque] : [nom]

Cet ordre provoque la destruction d'un objet sur le disque.

Indiquer le type de l'objet :

- programme
- fichier
- masque

LET"#Format-[n° disque]

Pour formatter une disquette; l'introduire dans l'un des lecteurs :

Ø : port numéro 6, drive 1
1 : port numéro 6, drive 2
2 : port numéro 5, drive 1

L'ordre de formattage est : LET "#FORMAT,Ø "

LET"# FORMAT, Ø " s'adressera à la disquette située dans le lecteur Ø
LET"# FORMAT,n " avec n = numéro du lecteur

Ne pas s'étonner de la durée relativement importante du formattage, ceci est normal.

En cas d'erreur, le système affiche le message : ? DATA ERROR
signalant qu'il lui est impossible d'utiliser la disquette.

Il peut y avoir plusieurs causes :

1) la porte du lecteur n'est pas fermée ou il n'y a pas de disquette dans le lecteur concerné.

2) La disquette est protégée en écriture et il est impossible d'écrire.

3) La disquette est usagée et la surface magnétique est endommagée, ce qui interdit son utilisation.

Si aucune erreur n'est détectée, le formattage se déroule normalement. (le voyant lumineux du lecteur s'allume)

A la fin du formattage, le système vous rend la main.

Cette disquette est alors utilisable.

5 - 2 - NIVEAU ACTION MASQUE

Lorsqu'un masque à été ouvert, les opérations suivantes sont possibles :

LET"Charge-[n° logique]

Charge sur l'écran le texte du masque.
Cette opération efface le contenu de l'écran, sauf aux endroits où le masque a été déclaré comme transparent. (voir le chapitre Création de masques)

LET"Visualise-[n° logique]

Charge sur l'écran le texte du masque et affiche les fenêtres de saisie. Les fenêtres sont réécrites avec la même syntaxe que celle exigée pour la création. Il est donc possible de créer un nouveau masque ou d'en modifier un en utilisant successivement les ordres visualise et NEW.
Les zones transparentes apparaissent sous la forme de caractères "@" sur fond blanc.

LET"Output-[n° logique][Ø/O]

Affiche sur l'écran les variables du masque.
Le paramètre (optionnel) Ø/O permet d'indiquer quelles sont les variables à afficher.

- Ø : affichage des variables définies en entrée (celles où se déplace le spot en lecture)
- O : affichage des variables en sortie

  Si le paramètre est omis, toutes les variables sont affichées.

Si une variable est trop longue pour entrer dans une fenêtre, elle est tronquée, et un caractère " ¤ " apparait en fin de fenêtre.

LET"Print-[n° logique][Ø/O]

Cet ordre est le cumul des deux ordres Charge et Output.
La syntaxe est la même que celle de Output.

LET"Input-[n° logique][,n° de zone]

Donne la main à l'utilisateur pour saisir les zones.
Seules les fenêtres déclarées en entrée sont concernées par cet ordre.
Le parametre (optionnel) permet d'indiquer à quelle fenêtre doit commencer la saisie. Attention ! seules les fenêtres en entrée comptent.
La première fenêtre porte le numéro 1, et ainsi de suite.
La valeur par défaut est 1.

Si le numéro indiqué est supérieur au nombre de fenêtres en entrée, le programme commencera par balayer le masque autant de fois qu'il est nécessaire pour atteindre ce numéro. Si la dernière fenêtre est du type " enchainement automatique ", le programme ressortira alors sans donner la main.

LET"Take, n° logique"

Reprend le contenu de toutes les variables en entrées du masque indiqué telles qu'elles sont actuellement sur l'écran.

EXEMPLE 1 : Remise à zéro

```
LET"C, M"
LET"T, M"
```

Remet toutes les variables du masque à vide ou nulles, puisque l'ordre chargé n'affiche aucune variable.

EXEMPLE 2 : Paramétrage

On saisit des paramètres sur un masque M, puis on sauve l'ensemble :

```
LET"# O, M, M, Ø : QUESTIONS
LET"P, M"
LET"I, M"
LET"# N, P, M, Ø : PARAMETRES
```

pour relire les paramètres, faire :

```
LET"# O, M, M, Ø : QUESTIONS
LET"# O, P, M, Ø : PARAMETRES
LET"C, P"
LET"T, M"
```

Toutes les variables du masque M reprennent les valeurs qu'elles avaient au moment du sauvetage.

5 - 3 - NIVEAU ENREGISTREMENT FICHIER

Ces ordres sont possibles sur un fichier ouvert.

LET"Write-[n° logique]

Création d'un nouvel enregistrement dans un fichier.

- Si le fichier est un fichier séquentiel indexé, la création se fait en fonction de la valeur de la clé au moment de l'ordre.

Exemple :

Fichier de clé AA$et AA 10 et d'enregistrement XX$

```
100 LET"#OPEN-1, FICHIER,Ø : FICO
110 AA$ = "MIS": AA = 1980
120 XX$ = "ENREGISTREMENT"
130 LET"WRITE,1
140 IF PEEK(189) THEN PRINT"EXISTANT"
```

L'ordre WRITE n'accepte pas les homonymes. Si la clé de l'enregistrement à créer existe déjà dans le fichier, le status renvoyé sera non nul et l'enregistrement ne sera pas créé.

- Si le fichier est un fichier relatif, le DOS ne tiendra pas compte de la valeur du pointeur. L'enregistrement sera créé à la fin du fichier, avec la première valeur libre pour le pointeur. Cette valeur du pointeur sera renvoyée à l'utilisateur.

LET"Add-[n° logique]

L'ordre ADD est identique à l'ordre WRITE, mais il accepte les homonymes dans le cas des fichiers séquentiels indexés.
Lorsque plusieurs enregistrements de même clé sont créés dans un fichier, l'ordre de rangement est l'ordre inverse de l'ordre de création.
L'enregistrement le plus récent sera donc lu en premier lors d'une lecture séquentielle.

LET"Read-[n° logique]

Lecture d'un enregistrement en fonction de la valeur de la clé (ou du pointeur).

Exemple :

```
200 XX$ = " " : AA$ = "MIS" : AA = 1980
210 LET"READ-1
220 ?XX$
```

LET"Next-[n° logique]

Lecture de l'enregistrement suivant.

Employé après l'ouverture du fichier, l'ordre NEXT permet de lire le fichier séquentiellement, chaque appel renvoyant à l'utilisateur la clé (ou le pointeur) et l'enregistrement suivants.

Si une lecture a été faite par READ, la lecture séquentielle par NEXT commence à la première clé (ou pointeur) suivant celle (celui) du READ.

REMARQUE : L'ordre NEXT ne tient pas compte de la valeur de la clé indiquée dans les variables BASIC. C'est la valeur obtenue à la dernière lecture qui compte.
Pour lire l'enregistrement suivant pour une valeur donnée de la clé, enchainer READ et NEXT.

Exemple :

Dans un fichier dont la clé est NO$ de longueur de 10 caractères, cherchons le premier enregistrement commençant par " T ".

```
100 LET"#C,$"
110 LET"#0,1,F,NOMS"
120 NO$ = " T "
130 LET"R,1" : IF PEEK (189) = 0 THEN 150
140 LET"N,1"
150 REM ICI LE PREMIER ENREGISTREMENT
160 REM COMMENCANT PAR " T " A ETE LU
```

LET"Update-[n° logique]

Mise à jour d'un enregistrement.
Indiquer les nouvelles valeurs des variables de l'enregistrement, puis exécuter UPDATE.
Il n'est pas nécessaire d'avoir lu l'enregistrement auparavant. (mais dans ce cas la mise à jour est plus rapide)
En cas d'homonymie, si l'un des homonymes vient d'être lu, c'est celui-là qui est modifié. Sinon, c'est toujours le premier (c'est-à-dire le plus récent) qui l'est.

Exemple :

```
300 AA = 1980 : AA$ = "MIS": LET" READ-1
310 XX$ = XX$+" VERSION 2
320 LET"UPDATE-1
```

LET"Delet-[n° logique]

Destruction d'un enregistrement dont on indique la clé (ou le pointeur).
Cette opération est possible aussi bien sur les fichiers séquentiels indexés que sur les fichiers relatifs.
Dans le cas d'un fichier séquentiel indexé, il est possible de recréer l'enregistrement détruit, par l'ordre WRITE ou ADD. Ceci est impossible pour un fichier relatif et une valeur du pointeur détruite ne pourra être réutilisée. (Dans le cas des fichiers relatifs, l'intérêt principal de cet ordre est la récupération de la place de l'enregistrement sur le disque.) Dans le cas d'un fichier multiclé, la place de l'enregistrement ne sera récupérée qu'à la prochaine ré-organisation.

LET"Borne-[n° logique]

Fixe une borne maximum au fichier pour les lectures séquentielles.
L'ordre NEXT déclenche une erreur " fin de fichier " (status = 255) si cette borne est atteinte ou dépassée.
Pour utiliser l'ordre BORNE, positionner les clés (ou le pointeur) à la valeur maximum désirée, puis exécuter l'ordre.
La valeur exacte de la borne, si elle correspond à une clé existante, n'est pas lue.
REMARQUE : l'ordre BORNE ne provoque pas de lecture disque, et ne trouble pas le fonctionnement de l'ordre NEXT (qui, rappelons-le, travaille par rapport à la dernière lecture et non par rapport aux valeurs des variables.)
Pour ne lire qu'une partie d'un fichier, il suffit donc de fixer une borne minimum par READ, une borne maximum par BORNE. Une boucle de lecture par NEXT permet d'obtenir cet extrait du fichier comme s'il s'agissait d'un fichier entier.

LET"Xtract-[n° logique],[n° variable]

Cette commande permet d'extraire des éléments d'un fichier en conditionnant l'une des variables composant la clé.

Par exemple, si la clé se compose de :

A1 % numéro de famille
A2 % numéro de fournisseur
A3 % numéro interne de l'article

Le numéro logique de ce fichier est " F ".

Pour lister les éléments du fichier appartenant au fournisseur numéro 2, il faut tout d'abord désigner au système le numéro d'ordre de la variable testée et sa valeur.

```
] LET " XTRACT, F, 2 "
] A1 % = Ø : A2 % = 2 : A3 % = Ø
] LET " READ, F "
? DIRECT ERROR
```

La commande XTRACT positionne le numéro d'ordre de la variable.
La commande READ, qui ici n'aboutit pas, fournit au système une valeur de référence pour la variable A2 %, en l'occurence 2.
De plus, il ne faut pas oublier que ce READ positionne le haut de fichier pour l'instruction LET " NEXT, F " qui extrait les éléments recherchés.

Il est possible de redonner une nouvelle valeur de recherche à la commande NEXT.
Pour cela, il faut exécuter un nouveau READ en positionnant les nouvelles valeurs pour les variables composant la clé.

ATTENTION : ne pas oublier qu'en faisant cela, le haut de fichier est repositionne pour le prochain NEXT.

LET"Xindex-[n° logique]

Pour les fichiers relatifs, cet ordre renvoie dans le pointeur le numéro d'ordre du prochain enregistrement qui sera créé.
Cet ordre permet, entre autre, de créer des chainages entre fichiers avant d'avoir créé tous les enregistrements.

5 - 4 - PROGRAMMES

Les ordres sont :

LOAD"[d] : [nom]

Chargement d'un programme.

LOAD"#[d] : [nom]

Chargement d'un programme à la suite de celui déjà existant en mémoire centrale.
IMPORTANT : Les numéros de lignes du second programme doivent être > à ceux du 1er. Dans le cas contraire, modifiez-les au préalable par l'utilitaire RENUMEROTE.

LOAD"/[d] : [nom]

Chargement, puis exécution d'un programme sans détruire les variables.
Attention ! le programme doit être d'une taille inférieure à celle du dernier programme chargé normalement.

SAVE"[d] : [nom]

Sauve le programme en cours.
S'il existe une erreur celle-ci sera signalée dans les status (PEEK (189)).

SAVE"@[d] : [nom]

Sauve le programme en cours en écrasant éventuellement une version précédemment enregistrée.

RUN"[d] : [nom]

Charge le programme et lance son exécution.

RUN"#(d) : [nom]

Charge le programme à la suite de celui déjà en mémoire, puis lance l'exécution de l'ensemble.

LET"%-[d]

Imprime le catalogue du disque d.

LET"%-[d],[P/M/F]

Imprime seulement la liste des Programmes, Masques ou Fichiers.

```
]NEW

]10PRINT"TEST"

]SAVE"1:TEST1"

]NEW

]20FORI=0TO10:?"#";:NEXT

]SAVE"1:TEST2"

]LET"*,1"

CATALOG DISK 1

P  TEST1
P  TEST2

]NEW

]LOAD"1:TEST1"

]LOAD"#1:TEST2"

]LIST

10  PRINT "TEST"
20  FOR I = 0 TO 10: PRINT "#";: NEXT


]SAVE"1:TEST1+2"

]RUN
TEST
###########
]LET"*,1"

CATALOG DISK 1

P  TEST1
P  TEST1+2
P  TEST2
```

REMARQUE : Dans le CATALOG, les objets sont imprimés dans l'ordre alphabétique. Les fichiers d'abord, puis les masques et enfin les programmes.

Notes concernant la syntaxe

1) Caractères significatifs

Les blancs ne sont pas significatifs, sauf dans le nom d'un objet.
Seul le premier caractère suivant un séparateur (, ; - :) compte (sauf dans le cas du nom d'un objet).
Les autres caractères sont ignorés.

Exemple :

```
LET"#OPEN-1, FICHIER,Ø : FIC TEST
```

est équivalant à :

```
LET"#0 - 1 , F , Ø : FIC TEST
```

2) Numéro logique

On peut choisir comme numéro logique n'importe quel caractére ASCII à l'exception du caractére " $ ".
De plus, dans tous les ordres, les caractères suivant le numéro logique sont ignorés.

On pourra écrire :

```
LET"#OP--STOCK-F,1 : STOCK
LET"READ-STOCK
Le numéro logique étant " S ".
```

3) Nom d'un objet

Le nom d'un objet est une suite de caractères quelconques. Seuls les 21 premiers caractères sont pris en compte. S'il y en a moins de 21, le nom est complété par des blancs (code ASCII 32).

4) Valeur par défaut du disque

Si le premier caractère du nom est supérieur à " 9 ", on pourra omettre le numéro du disque, qui gardera la valeur précédente.

Exemple :

```
LOAD"0 : PRG TEST
SAVE"PRG TEST
SAVE"@PRG TEST
```

5) Paramétrage des ordres

Les ordres ont été montrés sous la forme :

```
  LET"XXXXXX
ou LOAD"XXXXX....
```

Mais l'ordre peut être une chaine de caractères quelconque, construite ou non, envoyée par LET, LOAD, SAVE ou RUN.

Exemple :

```
100 INPUT"NOM DU MASQUE" ; NM$
110 LET"#OPEN-2,MASQ,1 : "+NM$
```

6) Séparateurs

Dans tous les ordres, les séparateurs sont interchangeables. (, : ; -)

7) Cas particuliers

Tout ordre du M/DOS 6502 suivant THEN doit être précédé de " : ".

Exemple : IF A = B THEN : LET G

Si l'on omet les " : ", le système répondra :

SYNTAX ERROR

5 - 5 - BINAIRES ET PROGRAMMES ASSEMBLEURS

Les modules binaires se comportent d'une façon voisine de celle des programmes BASIC. Les seules différences sont :

1) Les adresses délimitant la partie de mémoire à sauver ont été définies par l'utilisateur et non imposé comme dans le cas d'un programme BASIC.

2) Ils sont notés dans le catalogue sous le code " B " au lieu du code " P "

La syntaxe est la suivante :

Sauvetage :

SAVE "$XXXX, $YYYY, d : nom"

avec : XXXX adresse minimale en héxadécimal

Exemple : SAVE"$1ØØØ,$11ØØ,Ø : BIN"

sauve la partie de la mémoire comprise entre :
$1ØØØ et $1ØFF

Relecture :

LOAD"$XXXX,$YYYY,d : nom"

avec XXXX adresse de départ choisie
YYYY adresse maximale autorisée

Cette deuxième adresse permet de mettre en " chien de garde " évitant d'écraser une autre partie de la mémoire. Même si YYYY est supérieur à l'adresse de fin de chargement, le comportement du LOAD ne sera pas mofifié.

NOTE : Il peut être interessant de pouvoir connaitre l'adresse de fin d'un module à l'issue d'un chargement.
Cela est possible, l'adresse de fin du module +1 se trouve placé en ADBLC à l'issue du LOAD avec,

ADBLC = $3ØE poids fort de l'adresse
ADBLC + 1 = $3ØF poids faible de l'adresse

5 - 6 - ACCES DIRECT

Cette fonction permet de lire ou écrire un ou plusieurs secteurs, directement sur le disque ou la disquette. La syntaxe est :

```
LET"$d,[C=]c ,[N=]n ,[P=]p ,[T=]t ,[S=]s ,$XXXX
```

Avec XXXX adresse du chargement
- d numéro du drive (Ø 5)
- n nombre de secteurs à charger ou sauver
- p piste
- t tête
- s secteur
- c commande (1 = lire ; 2 = écrire ; 4 = formatter)

Les caractères notés en [ ] sont facultatifs, mais contrôlés s'ils sont présents.

Deux particularités interessantes de cet ordre sont :

1) dans le cas où le nombre n est trop important et que cela conduirait à dépasser la fin du disque, l'ordre est exécuté normalement tant que cela est possible, puis la main est rendue avec un status non nul pour signaler le fait.

2) après son exécution, le programme positionne en :

$3ØA $3ØB $3ØC

Les paramètres piste, tête, secteurs du PROCHAIN secteur à lire.

Une conséquence de ces caractéristiques est la simplicité de réalisation de certains utilitaires.

EXEMPLE : Programme copiant des drives de même type.

```
ready.

 50 himem:16*256
 100 input  " copie depuis ?";d1$
 110 input  " vers         ?";d2$
 120 let"#c,$"
 200 x=7*256*16+15*256+16+5
 210 p=0:s=0:t=0
 215 vtab10:print"lecture  "p" "t" "s
 220 let"$"+d1$+",1,16,"+str$(p)+","+str$(t)+","+str$(s)+"$1000"
 225 vtab10:print"ecriture "p" "t" "s
 230 let"$"+d2$+",2,16,"+str$(p)+","+str$(t)+","+str$(s)+"$1000"
 240 ifpeek(189)then500
 260 p=peek(x+1):t=peek(x+2):s=peek(x+3):goto215
 500 print"@@@fin
ready.
```

5 - 7 - EXEMPLES

```
 100 REM ************************************************
 101 REM * EXEMPLE 1                                    *
 102 REM *                                              *
 103 REM * ---------------------                        *
 104 REM * CLEAR-$    >     NEW                         *
 105 REM * ---------------------                        *
 110 REM *                                              *
 120 REM *                                              *
 130 REM * CREATION D'UN FICHIER STOCK                   *
 131 REM * ---------------------------                  *
 135 REM *                                              *
 140 REM * LE FICHIER EST UN FICHIER SEQUENTIEL INDEXE   *
 150 REM *                                              *
 160 REM * LA CLE :                                     *
 170 REM *  FO%=NUMERO DE FOURNISSEUR                   *
 180 REM *  NP%=NUMERO DE PIECE                         *
 190 REM *                                              *
 200 REM * L'ENREGISTREMENT                             *
 210 REM *  PA=PRIX D'ACHAT                             *
 220 REM *  PV=PRIX DE VENTE                            *
 230 REM *  TV%/=TYPE TVA                               *
 240 REM *  QS=QUANTITE EN STOCK                        *
 250 REM *  QC=QUANTITE EN COMMANDE                     *
 260 REM *  QM=QUANTITE MINIMALE                        *
 270 REM *  LI$=LIBELLE DE LA PIECE                     *
 280 REM *                                              *
 290 REM *                                              *
 300 REM * LE PROGRAMME INDIQUE LA DEFINITION DE        *
 310 REM *  L'ENREGISTREMENT ET CREE LE FICHIER         *
 320 REM ************************************************
 400 LET"#CLEAR-$":S=189
 410 LET">FO%,NP%=PA,PV,TV%,QS,QC,QM,LI$
 420 LET"#NEW,1,FICHIER,0:STOCK
 430 IF PEEK(S)THENPRINT"ERREUR "
READY.
```

```
100 REM ************************************************
105 REM * EXEMPLE 2                                     *
110 REM *                                               *
120 REM *                                               *
130 REM * MISE A JOUR D'UN FICHIER                      *
140 REM *                                               *
150 REM * LE PROGRAMME EST INDEPENDANT                  *
160 REM * DU FICHIER                                    *
170 REM *                                               *
180 REM * SEULS LES MASQUES UTILISES SONT A CHANGER     *
190 REM *                                               *
200 REM *                                               *
300 REM * LE PROGRAMME PERMET LES ORDRES SUIVANTS :      *
310 REM *  LECTURE D'UNE PIECE                          *
320 REM *  CREATION D'UNE PIECE                         *
330 REM *  MODIFICATION D'UNE PIECE                     *
340 REM *  DESTRUCTION D'UNE PIECE                      *
350 REM *  SUIVANT (LECTURE DANS L'ORDRE DES CLES)      *
360 REM *                                               *
370 REM * LE PROGRAMME UTILISE 2 MASQUES                *
380 REM * - CLE-XXXXX = DEMANDE DE LA CLE ET DE L'ORDRE *
390 REM * - ART-XXXXX = DEMANDE OU AFFICHAGE DE LA PIECE*
400 REM *                                               *
410 REM ************************************************
500 LET"#CLEAR-$":S=189:INPUT"NOM DU FICHIER ";N$
510 LET"#OPEN,F,FICHIER,0:"+N$
520 LET"#OPEN,C,MASQUE,0:CLE-"+N$
530 LET"#OPEN,A,MASQUE,0:ART-"+N$
540 LET"CHARGE,C
550 LET"OUTPUT,C":MC$=""
560 LET"INPUT,C":IFPEEK(S)THENMC$="REPRENEZ":GOTO550
570 IFO$="F"THENEND
580 IFO$="L"THEN700
590 IFO$="S"THEN800
600 IFO$="M"THEN900
610 IFO$="C"THEN1000
620 IFO$="D"THEN1100
630 MC$="ORDRE ERRONE":GOTO550
700 LET"READ-F":IFPEEK(S)THEN790
710 MA$="LECTURE:FAITE RETURN":LET"PRINT,A
720 GETA$:IFA$<>CHR$(13)THEN720
730 GOTO540
790 MC$="ARTICLE INEXISTANT":GOTO550
800 LET"READ-F":IFPEEK(S)THEN790
810 LET"NEXT-F":IFPEEK(S)THENMC$="PLUS D'ARTICLE":GOTO550
820 GOTO710
900 LET"READ-F":IFPEEK(S)THEN790
910 MA$="MODIFICATION":LET"PRINT-A":LET"INPUT-A":IFPEEK(S)THEN990
920 LET"UPDATE-F":GOTO540
990 MC$="ABANDON":GOTO540
1000 LET"READ-F":IFPEEK(S)=0THENMC$="ARTICLE EXISTANT":GOTO550
1010 MA$="CREATION":LET"PRINT-A":LET"INPUT-A":IFPEEK(S)THEN990
1020 LET"WRITE-F":GOTO540
1100 LET"READ-F":IFPEEK(S)THEN790
1110 MA$="OK POUR DETRUIRE (O/N)":LET"PRINT-A"
1120 GETA$:IFA$="N"THEN990
1130 IFA$<>"O"THEN1120
1140 LET"DELET-F":GOTO540
READY.
```

```
 100 REM ************************************************
 101 REM * EXEMPLE 3                                    *
 102 REM *                                              *
 103 REM * ---------------------                        *
 104 REM * CLEAR-$  OPEN  NEXT                          *
 105 REM * ---------------------                        *
 110 REM *                                              *
 120 REM *                                              *
 130 REM * LECTURE SEQUENTIELLE D'UN FICHIER             *
 131 REM *--------------------------------              *
 135 REM *                                              *
 140 REM * LE FICHIER EST UN FICHIER SEQUENTIEL INDEXE   *
 150 REM *                                              *
 160 REM * LA CLE :                                     *
 170 REM *  FO%=NUMERO DE FOURNISSEUR                   *
 180 REM *  NP%=NUMERO DE PIECE                         *
 190 REM *                                              *
 200 REM * L'ENREGISTREMENT                             *
 210 REM *  PA=PRIX D'ACHAT                             *
 220 REM *  PV=PRIX DE VENTE                            *
 230 REM *  TV%/=TYPE TVA                               *
 240 REM *  QS=QUANTITE EN STOCK                        *
 250 REM *  QC=QUANTITE EN COMMANDE                     *
 260 REM *  QM=QUANTITE MINIMALE                        *
 270 REM *  LI$=LIBELLE DE LA PIECE                     *
 280 REM *                                              *
 290 REM *                                              *
 300 REM * LE PROGRAMME LISTE  LE                       *
 310 REM *    FICHIER COMPLET                           *
 320 REM ************************************************
 400 LET"#CLEAR-$":S=189
 410 LET"#OPEN,1,FICHIER,0:STOCK
 450 LET"NEXT-1": IF PEEK(S) THEN END
 460 PRINT FO%,NP%,PA,PV,TV%,QS,QC,QM,LI$:GOTO450
READY.
```

```
 100 REM ************************************************
 101 REM * EXEMPLE 4                                    *
 102 REM *                                              *
 103 REM * ---------------------------------            *
 104 REM * CLEAR-$  OPEN  NEXT  READ  BORNE             *
 105 REM * ---------------------------------            *
 110 REM *                                              *
 120 REM *                                              *
 130 REM * LECTURE D'UNE PARTIE D'UN FICHIER             *
 131 REM * ---------------------------------            *
 135 REM *                                              *
 140 REM * LE FICHIER EST UN FICHIER SEQUENTIEL INDEXE   *
 150 REM *                                              *
 160 REM * LA CLE :                                     *
 170 REM *  FO%=NUMERO DE FOURNISSEUR                   *
 180 REM *  NP%=NUMERO DE PIECE                         *
 190 REM *                                              *
 200 REM * L'ENREGISTREMENT                             *
 210 REM *  PA=PRIX D'ACHAT                             *
 220 REM *  PV=PRIX DE VENTE                            *
 230 REM *  TV%/=TYPE TVA                               *
 240 REM *  QS=QUANTITE EN STOCK                        *
 250 REM *  QC=QUANTITE EN COMMANDE                     *
 260 REM *  QM=QUANTITE MINIMALE                        *
 270 REM *  LI$=LIBELLE DE LA PIECE                     *
 280 REM *                                              *
 290 REM *                                              *
 300 REM * LE PROGRAMME LISTE  LES PIECES D'UN FOURNISSEUR *
 310 REM *                                              *
 320 REM ************************************************
 400 LET"#CLEAR-$":S=189
 410 LET"#OPEN,1,FICHIER,0:STOCK
 420 INPUT"NOM DU FOURNISSEUR CONCERNE ";FO%
 430 NP%=0:LET"READ-1
 440 NP%=32700:LET"BORNE-1
 450 LET"NEXT-1": IF PEEK(S) THEN END
 460 PRINT FO%,NP%,PA,PV,TV%,QS,QC,QM,LI$:GOTO450
READY.
```

```
 100 REM ************************************************
 101 REM * EXEMPLE 5                                    *
 102 REM *                                              *
 103 REM * -------------------------------              *
 104 REM * CLEAR-$  OPEN  NEXT  XTRACT                  *
 105 REM * -------------------------------              *
 110 REM *                                              *
 120 REM *                                              *
 130 REM * EXTRACTION D'UN SOUS FICHIER                  *
 131 REM * -------------------------------              *
 135 REM *                                              *
 140 REM * LE FICHIER EST UN FICHIER SEQUENTIEL INDEXE   *
 150 REM *                                              *
 160 REM * LA CLE :                                     *
 170 REM *  FO%=NUMERO DE FOURNISSEUR                   *
 180 REM *  NP%=NUMERO DE PIECE                         *
 190 REM *                                              *
 200 REM * L'ENREGISTREMENT                             *
 210 REM *  PA=PRIX D'ACHAT                             *
 220 REM *  PV=PRIX DE VENTE                            *
 230 REM *  TV%/=TYPE TVA                               *
 240 REM *  QS=QUANTITE EN STOCK                        *
 250 REM *  QC=QUANTITE EN COMMANDE                     *
 260 REM *  QM=QUANTITE MINIMALE                        *
 270 REM *  LI$=LIBELLE DE LA PIECE                     *
 280 REM *                                              *
 290 REM *                                              *
 300 REM * LE PROGRAMME LISTE  LES PIECES DONT          *
 310 REM * LE NUMERO (NP%) EST EGAL  A 1000             *
 320 REM ************************************************
 400 LET"#CLEAR-$":S=189
 410 LET"#OPEN,1,FICHIER,0:STOCK
 440 NP%=1000:LET"XTRACT-1
 450 LET"NEXT-1": IF PEEK(S) THEN END
 460 PRINT FO%,NP%,PA,PV,TV%,QS,QC,QM,LI$:GOTO450
READY.
```

```
 100 REM ************************************************
 101 REM * EXEMPLE 6                                    *
 102 REM *                                              *
 103 REM * ----------------------                       *
 104 REM * CLEAR-$      >     NEW                       *
 105 REM * ----------------------                       *
 110 REM *                                              *
 120 REM *                                              *
 130 REM * CREATION D'UN FICHIER RELATIF                 *
 131 REM * ----------------------------                 *
 135 REM *                                              *
 137 REM *  PAR EXEMPLE : FICHIER D'ECRITURES COMPTABLES *
 138 REM *                                              *
 140 REM * SA DESCRIPTION EST :                         *
 150 REM *                                              *
 160 REM * LE POINTEUR                                  *
 170 REM *  NEX=NUMERO D'ECRITURE                       *
 190 REM *                                              *
 200 REM * L'ENREGISTREMENT                             *
 210 REM *  DA$*=DATE DE L'ECRITURE                     *
 220 REM *  LI$=LIBELLE                                 *
 230 REM *  CP$=COMPTE CONCERNE                         *
 240 REM *  DE=DEBIT                                    *
 250 REM *  CR=CREDIT                                   *
 260 REM *  NF$=REFERENCE FACTURE                       *
 270 REM *  DP$*=DATE DE L'OPERATION                    *
 280 REM *                                              *
 290 REM *                                              *
 300 REM * LE PROGRAMME INDIQUE LA DESCRIPTION          *
 310 REM *  DE L'ENREGISTREMENT ET CREE LE FICHIER      *
 320 REM ************************************************
 400 LET"#CLEAR-$":S=189
 410 LET">@NEX=DA$*,LI$,CP$,DE,CR,NF$,DP$*
 420 LET"#NEW,2,FICHIER,1:ECRITURES
 430 IF PEEK(S) THEN PRINT"ERREUR "
READY.
```

```
100 REM ************************************************
101 REM * EXEMPLE 7                                     *
102 REM *                                               *
103 REM * ----------------------                        *
104 REM * CLEAR-$      >     NEW                        *
105 REM * ----------------------                        *
110 REM *                                               *
120 REM *                                               *
130 REM * LISTE D'UN FICHIER RELATIF                     *
131 REM * --------------------------                    *
135 REM *                                               *
137 REM *  PAR EXEMPLE : FICHIER D'ECRITURES COMPTABLES  *
138 REM *                                               *
140 REM * SA DESCRIPTION EST :                          *
150 REM *                                               *
160 REM * LE POINTEUR                                   *
170 REM *  NE%=NUMERO D'ECRITURE                        *
190 REM *                                               *
200 REM * L'ENREGISTREMENT                              *
210 REM *  DA$*=DATE DE L'ECRITURE                      *
220 REM *  LI$=LIBELLE                                  *
230 REM *  CP$=COMPTE CONCERNE                          *
240 REM *  DE=DEBIT                                     *
250 REM *  CR=CREDIT                                    *
260 REM *  NF$=REFERENCE FACTURE                        *
270 REM *  DP$*=DATE DE L'OPERATION                     *
280 REM *                                               *
290 REM *                                               *
300 REM * LE PROGRAMME LISTE LE FICHIER COMPLET          *
310 REM *  ( JOURNAL )                                  *
320 REM ************************************************
400 LET"#CLEAR-$":S=189
420 LET"#OPEN,3,FICHIER,1:ECRITURES
430 LET"NEXT-3":IF PEEK(S) THEN END
440 PRINTNE%,DA$,LI$,DE,CR,NF$,DP$
450 GOTO430
READY.
```

```
100 REM ************************************************
101 REM * EXEMPLE 8                                     *
102 REM *                                               *
103 REM * ----------------------                        *
104 REM * CLEAR-$      >     NEW                        *
105 REM * ----------------------                        *
110 REM *                                               *
120 REM *                                               *
130 REM * CREATION D'UN FICHIER MULTICLE                *
131 REM * ------------------------------                *
135 REM *                                               *
137 REM *  PAR EXEMPLE : FICHIER DES COMPTES COMPTABLES  *
138 REM *                                               *
140 REM * SA DESCRIPTION EST :                          *
150 REM *                                               *
160 REM * 1ERE CLE                                      *
170 REM *  CO$10=LE NUMERO DE COMPTE                    *
190 REM *                                               *
200 REM * 2EME CLE                                      *
210 REM *  AL$12=CODE ALPHABETIQUE EQUIVALENT           *
220 REM *                                               *
230 REM * L'ENREGISTREMENT                              *
240 REM *  LI$=LIBELLE DU COMPTE                        *
250 REM *  CR=CREDIT                                    *
260 REM *  DE=DEBIT                                     *
270 REM *  SD=SOLDE                                     *
280 REM *  AL$=CODE ALPHA (REPETE POUR INFORMATION)     *
290 REM *  CO$=COMPTE     (REPETE POUR INFORMATION)     *
295 REM *                                               *
300 REM * LE PROGRAMME INDIQUE LA DESCRIPTION           *
310 REM *  DE L'ENREGISTREMENT ET CREE LE FICHIER       *
320 REM ************************************************
400 LET"#CLEAR-$":S=189
410 LET">CO$10 & AL$12 =LI$,CR,DE,SD,AL$,CO$
420 LET"#NEW,4,FICHIER,1:COMPTES
430 IF PEEK(S) THEN PRINT"ERREUR
READY.
```

```
100 REM ************************************************
101 REM * EXEMPLE 9                                     *
102 REM *                                               *
103 REM * --------------------------                    *
104 REM * CLEAR-$  OPEN  NEXT  UPDATE                   *
105 REM * --------------------------                    *
110 REM *                                               *
120 REM *                                               *
130 REM * MISE A JOUR D'UN FICHIER  (MULTICLE)          *
131 REM * ----------------------------                  *
135 REM *                                               *
137 REM *  PAR EXEMPLE : FICHIER DES COMPTES COMPTABLES  *
138 REM *                                               *
140 REM * SA DESCRIPTION EST :                          *
150 REM *                                               *
160 REM * 1ERE CLE                                      *
170 REM *  CO$10=LE NUMERO DE COMPTE                    *
190 REM *                                               *
200 REM * 2EME CLE                                      *
210 REM *  AL$12=CODE ALPHABETIQUE EQUIVALENT           *
220 REM *                                               *
230 REM * L'ENREGISTREMENT                              *
240 REM *  LI$=LIBELLE DU COMPTE                        *
250 REM *  CR=CREDIT                                    *
260 REM *  DE=DEBIT                                     *
270 REM *  SD=SOLDE                                     *
280 REM *  AL$=CODE ALPHA (REPETE POUR INFORMATION)     *
290 REM *  CO$=COMPTE     (REPETE POUR INFORMATION)     *
295 REM *                                               *
300 REM * LE PROGRAMME EFFECTUE LE CALCUL SUIVANT       *
310 REM *  SOLDE (SD) = DEBIT (DE) - CREDIT (CR)        *
315 REM *  ET REECRIT L'ENREGISTREMENT MODIFIE          *
320 REM *                                               *
330 REM *  ICI ,LE FICHIER EST MULTICLE ,LA LECTURE SE   *
340 REM *  FAIT SUR LE PREMIER MOYEN D'ACCES            *
350 REM *  QUI CORRESPOND A LA VALEUR PAR DEFAUT .       *
360 REM *                                               *
370 REM *  LE PROGRAMME EST DONC LE MEME POUR UN FICHIER *
375 REM *  SIMPLE OU UN FICHIER MULTICLE .              *
380 REM ************************************************
 400 LET"#CLEAR-$":S=189
 420 LET"#OPEN,6,FICHIER,1:COMPTES
 430 LET"NEXT-6":IF PEEK(S) THEN END
 440 SD=DE-CR: LET"UPDATE-6":GOTO430
READY.
```

```
100 REM ************************************************
105 REM * EXEMPLE 10                                    *
110 REM *                                              *
120 REM *                                              *
130 REM * MISE A JOUR D'UN FICHIER AVEC HOMONYMES       *
140 REM *                                              *
150 REM * LE PROGRAMME EST INDEPENDANT                  *
160 REM * DU FICHIER                                    *
190 REM *                                              *
280 REM *                                              *
300 REM * LE PROGRAMME PERMET LES ORDRES SUIVANTS :     *
310 REM *  LECTURE ,CREATION ,DESTRUCTION,MODIFICATION  *
320 REM *  ET LECTURE SEQUENTIELLE (SUIVANT)            *
360 REM *                                              *
370 REM * LE PROGRAMME UTILISE 2 MASQUES                *
380 REM * - CLE-XXXXX = DEMANDE DE LA CLE ET DE L'ORDRE *
390 REM * - ART-XXXXX = DEMANDE OU AFFICHAGE DE LA PIECE*
400 REM *                                              *
410 REM *  LES DIFFERENCES AVEC LE PROGRAMME N'ACCEPTANT*
420 REM *  PAS LES HOMONYMES SONT:                     *
430 REM *    - POUR CREER UN ARTICLE ,ADD REMPLACE WRITE *
440 REM *    - POUR METTRE A JOUR ,NE PAS FAIRE READ     *
450 REM *      AVANT NEXT ,CELA PERMET DE LIRE PAR NEXT  *
460 REM *      LES DIFFERENTS ARTICLES DE MEME CLE .     *
490 REM ************************************************
500 LET"#CLEAR-$":S=189:INPUT"NOM DU FICHIER ";N$
510 LET"#OPEN,F,FICHIER,0:"+N$
520 LET"#OPEN,C,MASQUE,0:CLE-"+N$
530 LET"#OPEN,A,MASQUE,0:ART-"+N$
540 LET"CHARGE,C
550 LET"OUTPUT,C":MC$=""
560 LET"INPUT,C":IFPEEK(S)THENMC$="REPRENEZ":GOTO550
570 IFO$="F"THENEND
580 IFO$="L"THEN700
590 IFO$="S"THEN800
600 IFO$="M"THEN900
610 IFO$="C"THEN1000
620 IFO$="D"THEN1100
630 MC$="ORDRE ERRONE":GOTO550
700 LET"READ-F":IFPEEK(S)THEN790
710 MA$="LECTURE:FAITE RETURN":LET"PRINT,A
720 GETA$:IFA$<>CHR$(13)THEN720
730 GOTO540
790 MC$="ARTICLE INEXISTANT":GOTO550
800 LET"NEXT-F":IFPEEK(S)THENMC$="PLUS D'ARTICLE":GOTO550
820 GOTO710
900 LET"READ-F":IFPEEK(S)THEN790
910 MA$="MODIFICATION":LET"PRINT-A":LET"INPUT-A":IFPEEK(S)THEN990
920 LET"UPDATE-F":GOTO540
990 MC$="ABANDON":GOTO540
1000 LET"READ-F":IFPEEK(S)=0THENMC$="ARTICLE EXISTANT":GOTO550
1010 MA$="CREATION":LET"PRINT-A":LET"INPUT-A":IFPEEK(S)THEN990
1020 LET"ADD-F":GOTO540
1100 LET"READ-F":IFPEEK(S)THEN790
1110 MA$="OK POUR DETRUIRE (O/N)":LET"PRINT-A"
1120 GETA$:IFA$="N"THEN990
1130 IFA$<>"O"THEN1120
1140 LET"DELET-F":GOTO540
READY.
```

```
100 REM ********************************************************
110 REM * EXEMPLE 11                                            *
120 REM *                                                       *
130 REM * TRI DE FICHIER                                        *
140 REM * --------------                                        *
150 REM *                                                       *
160 REM * ----------------------------------------------------- *
170 REM * CLEAR-$  NEXT  REORGANISE  ADD  NEW                   *
180 REM * ----------------------------------------------------- *
190 REM *                                                       *
200 REM * FICHIER D'ORIGINE : FICH ORIG                         *
210 REM *                                                       *
230 REM *  CLE : A$10 ,AA                                       *
240 REM *  ENREGISTREMENT : BB ,CC ,DD                          *
250 REM *                                                       *
260 REM * FICHIER D'ARRIVEE : FICH ARR                          *
270 REM *                                                       *
280 REM *  CLE : CC ,DD                                         *
290 REM *  ENREGISTREMENT : A$ ,BB ,AA                          *
300 REM *                                                       *
310 REM * PRINCIPE DU TRI :                                     *
320 REM *  CREATION DANS UN NOUVEAU FICHIER SEQUENTIEL          *
330 REM *  INDEXE .LE PROGRAMME REORGANISE LE NOUVEAU FICHIER   *
340 REM *  DES QUE LE NOMBRE D'ENREG.  CREES ATTEINT LE NOMBRE  *
350 REM *  D'ENREGISTREMENT DANS LE FICHIER (SAUF AU DEBUT OU   *
360 REM *  LES INSERTIONS SONT RAPIDES )                        *
370 REM *                                                       *
380 REM ********************************************************
400 LET"#CLEAR-$":S=189
410 LET"#OPEN,ORI,FICHIER,0:FICH ORIG
420 LET">CC,DD=A$,BB,AA
430 LET"#NEW,ARR,FICHIER,0:FICH ARR
440 NRANGES=0:NTAS=0
450 LET"NEXT-ORI":IF PEEK(S) THEN PRINT"TRI TERMINE":END
460 LET"ADD-ARR":NTAS=NTAS+1:IF NTAS < NRANGES THEN 450
470 IF NRANGES=0 AND NTAS<30 THEN 450
480 LET"#REORG-ARR":NRANGES=NRANGES+NTAS:NTAS=0:GOTO450
READY.
```

CHAPITRE VI

FONCTIONS COMPLÉMENTAIRES

6 - 1 - CONTENU D'UN DISQUE

Le M/DOS 6502 permet de connaitre la place libre sur une disquette.

Utilisez pour celà l'ordre :

```
LET " % d "      d = numéro de drive
```

Par exemple :

```
LET " % Ø "
```

Pour connaitre la place disponible, lire l'octet situé à l'adresse 189 (habituellement utilisé comme status)
Cet octet contient le nombre de pistes disponibles sur la disquette.

Remarque : En mode direct si le nombre de pistes libres est non nul, le message ? DIRECT ERROR apparait, mais cela n'a pas d'importance.

Le nombre total de pistes de la disquette est 34.

Si par exemple vous obtenez le résultat

```
? PEEK (189)
17
```

La disquette est occupée à 50 %.

6 - 2 - CHANGEMENT DE DISQUETTES

Pour accélérer le fonctionnement du M/DOS 6502, certains pointeurs décrivent la disquette logiquement. Si vous changez la disquette sans que le système le sache, vous risquez alors de détruire le contenu de celle-ci.
Pour signaler le changement de disquette, utiliser l'ordre :

```
LET "# C," + CHR$(d)      d = numéro de drive
```

De plus, tous les fichiers ouverts sur le drive devront être fermés.

REMARQUE : En mode direct, l'ordre LET "# C," + CHR$(d) est effectué automatiquement, avant chaque ordre.

Par exemple : En mode direct

```
LOAD " Ø : TEST "
```

Changement de disquette

```
SAVE " Ø : TEST "
```

ne détruit pas la disquette

En mode programme

```
10 LOAD " Ø : TEST "
20 INPUT " CHANGEZ LA DISQUETTE ET FAITES RETURN " ; A$
30 SAVE " Ø : TEST
```

détruit la disquette.

Il faut faire :

```
10 LOAD " Ø : TEST "
20 INPUT " CHANGEZ LA DISQUETTE ET FAITES RETURN " ; A$
30 LET " #C, " + CHR$ (Ø)
40 SAVE " Ø : TEST "
```

NOTE : Après LET " #C,$ qui nettoie tous les buffers du DOS, il est possible de changer de disquette sans risque.

CAS PARTICULIER : LET"#C,$$" ferme tous les drives.

L'ordre LET ")M " supprime le CLEAR en mode direct (ce qui permet d'utiliser les fichiers), mais le contrôle n'est plus effectué.
LET ") " rend le CLEAR.

6 - 3 - FONCTIONS DE CALCUL SUR 48 CHIFFRES

Le M/DOS 6502 contient des fonctions de calcul qui sont une aide précieuse pour les applications de gestion comme la comptabilité.

Il y a trois fonctions :

- addition
- soustraction
- arrondi

Ces fonctions manipulent des chaines de caractères.

La précision des calculs est de 48 chiffres

Exemple :

```
LET " = 325.1 + 3.21 "
PRINT WW$
328.31
```

La syntaxe générale est :

```
LET " = op1 ↯ op2 "
Résultat dans WW$
```

ou

```
LET " = " + 01$ + " + " +02$  (addition)
LET " = " + 01$ + " - " +02$  (soustraction)
LET " = " + 01$               (arrondi)
```

6 - 4 - PRECISIONS DES CALCULS

Les calculs se font avec un nombre de décimales fixé par l'ordre :

```
LET "Ω - n "     n = nombre de décimales (1 à 9)
```

Exemple

```
LET "Ω - 2 "
```

Par défaut, au démarrage du système, le nombre de décimales est 2.
Les arrondis sont faits à la valeur inférieure.

Exemples :

```
 5 REM PROGRAMME DE CALCUL
10 INPUT " OPERANDE 1 ? " ; 01$
20 INPUT " OPERANDE 2 ? " ; 02$
30 INPUT " OPERANDE (+/-) " ; OP$
40 LET " = "+  01$ + OP$ + OP$
50 PRINT " RESULTAT = ";WW$

] LET "└3 + 2.2 " : PRINT WW$
5.2
] LET "Ω - 2 "
] LET " = 3.234 + 1.001 " : PRINT WW$
4.23
] LET " = 3.1234 " : PRINT WW$
3.12
] LET " = 1234567890123456789.00000 "
1234567890123456789
```

Le résultat dans WW% est toujours présenté sous la forme la moins couteuse en place.

```
    0  :  vide
 1.20  :  1.2
```

6 - 5 - SAISIE ET AFFICHAGE

Cet outil de calcul ne serait pas utilisable s'il n'était complété par les fonctions complémentaires de saisie et d'affichage.
Ces fonctions existent dans le M/DOS 6502.

Dans un masque, définissez une zone comme :

```
" A% + "
```

Elle sera considérée comme un numérique au niveau de la saisie, et comme alphanumérique pour le stockage.

L'affichage se fait en format gestion, cadré à droite en décimale fixe. Le nombre de décimales est fixé par le même ordre.

```
LET "Ω - n "
```

Note : pour l'affichage gestion, se reporter à ce chapitre dans la documentation.

6 - 6 - AZERTY

De nombreux utilisateurs, habitués aux machines à écrire, demandent un clavier du même type sur leur micro-ordinateur.
Le M/DOS 6502 permet de résoudre ce problème pour l'utilisation de l'ordre :

```
LET " Z "
```

qui échange la signification de certaines touches pour obtenir l'ordre classique AZERTY.

Pour cela, il est nécessaire d'échanger sur le clavier les capuchons des touches :

```
A       Q
W       Z
M       ;
```

et d'ajouter l'ordre LET " Z " dans le programme HELLO qui est executé à la mise en route de votre système.

Remarque : Si vous enfoncez accidentellement la touche RESET, faites

```
CTRL/C RETURN
LET " Z "
```

pour reprendre la main.

Attention ! pour taper LET " Z " faites W au lieu de Z, dans ce cas.

6 - 7 - EXECUTE

L'execute est une fonction issue du langage APL qui permet de construire et d'executer par programme un certain nombre d'instructions.

Le M/DOS 6502 vous offre cette possibilité.

L'opération se fait en deux temps :

1) Enregistrement des instructions par :

```
LET " >          "    pour entrer le texte
LET " +          "    pour en ajouter
```

Note : Il faut remplacer les cotes (") par CHRØ (34) et finir les lignes par CHRØ (13) sauf la dernière.

Exemple :

```
LET " > HOME : VTAB10 " + CHRØ(13)
LET " + PRINT " + CHRØ (34) + " BONJOUR "
```

2) Execution : pour executer ces instructions, faites:

```
LET " ! "  : END
```

Après l'execution, le buffer est vidé.

Vérification :
L'ordre LET " V " permet de relire le texte des ordres enregistrés par LET " >..." et LET " +..." sans les détruire.

Dans notre exemple après le premier temps,

```
LET " V "
HOME : VTAB10
PRINT " BONJOUR
```

Remarque : Les ordres sont executés comme s'ils étaient tapés en mode direct. En particulier les erreurs s'afficheront même si un ordre ON ERR GOTO a été inseré dans le programme.

Exemple : Programme chargeant et listant un certain nombre de programmes donnés par des DATA, (# indique la fin de la liste.)

```
 5 LET " > " : REM VIDE LE BUFFER
10 READ N$ : IF N$ = "#" THEN 60
20 LET " + LOAD " + CHR$(34) + N$ + CHR$(13)
30 LET " + PRINT " + CHR$(34) + N$ + CHR$(13)
40 LET " + LIST " + CHR$(13)
50 GOTO 10
60 LET " + PR#Ø : NEW
70 PR#2 : LET " ! "
90 DATA UTIL, AUTO COPIE, DEMO, HGR,#
```

3) L'exécute et ses réalisations

L'ordre LET " ! " du M/DOS permet de modifier l'entrée clavier. Les caractères ne sont plus lus au clavier mais dans un buffer du DOS.

EXEMPLE :

```
10 LET " > XYZ "
20 LET " + ABC "
30 LET " ! "
40 INPUT A$
50 ? A$
RUN
XYZABC
```

Les caractères peuvent être relus par GET ou INPUT.

APPLICATIONS

1) Passage de paramètres d'un programme à un autre :

Vous désirez passer les variables A$ et B$. Faire :

```
20 LET " > " + A$ + CHR$(13) + B$
30 RUN " TOTO "
- - - - - - -
TOTO LET " ! "
INPUT A$,B$
-
-
- etc ...
```

Lorsque tous le buffer a été relu, le GET redevient normal.

2) Lecture des dictionnaires

```
10 LET " #0,F,F,FILE "
20 LET " E,F "
30 LET " ! "
40 GET A$ : IF A$ = " 2 " THEN
   ? " FICHIER RELATIF "
```

Dans cet exemple, on teste le premier caractère du dictionnaire, pour savoir si le fichier est relatif.

3) Exécution d'ordre BASIC

On désire exécuter une formule de calcul saisie au clavier.

```
 10 INPUT " Y = ? " ; A$
 20 LET " > Y = " + A$ + " :GOTO 100 " +CHR$(13)
 30 LET " + GOTO 200 "
 40 END
100 ? " RESULTAT " Y : GOTO 10
200 ? " ERREUR " : GOTO 10
```

Le END rend la main au BASIC qui exécute l'ordre.
Les lignes tapées sont équivalentes à :

```
] Y = (.....) : GOTO 100
] GOTO 200
```

En cas d'erreur, la première ligne n'est pas exécuter entièrement, et le GOTO 100 ne se fait pas.

4) Empêcher les demandes clavier pendant le CATALOG

LET " * " demande d'appuyer sur une touche s'il y a plus de 20 lignes. Pour empêcher ce phénomène, faire :

```
10 LET "  > AAAAAAAA
20 LET " ! "
30 LET " * "
40 IN#Ø
```

Le IN Ø interroge le EXECUTE car on ne sait pas s'il ne reste pas de " A " à prendre.

CHAPITRE VII

PARAMÉTRAGE DU M/DOS 6502

La carte M/DOS 6502 présente un certain nombre de particularités au niveau de son implantation et de son adaptabilité.
Elle est théoriquement adaptable :

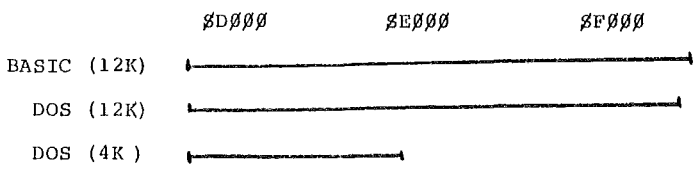
- à toute carte 80 colonnes
- à tout terminal connecté en lieu et place de la vidéo et du clavier d'origine ayant l'écran projeté en mémoire
- à toute mémoire de masse qu'elle peut gérer par tranches allant jusqu'à 16 méga octets.

7 - 1 - IMPLANTATION

La mémoire de l'APPLE II se décompose de la façon suivante :

$0000 - $BFFF	RAM
$C000 - $CFFF	Entrées / sorties
$D000 - $F7FF	BASIC (ROM)
$F800 - $FFFF	Moniteur

Une particularité essentielle de l'APPLE II est la possibilité de déconnecter les ROM Basic et MONITEUR pour les remplacer par d'autres mémoires externes.
C'est cette possibilité qu'utilise la carte M/DOS 6502. Les programmes assembleurs du DOS sont donc en parallèle avec le Basic et le Moniteur. Le Basic et le Moniteur occupent ensemble une place mémoire égale à 12K. Le M/DOS 6502 occupe lui 16K. Il a donc fallu décomposer cet espace de 16K en deux parties elles-mêmes superposées.



Les commutations entre le BASIC et les différentes pages du M/DOS 6502 se font par un logiciel intégré dans la carte. Cette opération est transparente pour l'utilisateur, ce qui signifie que pour lui, le Basic et le DOS fonctionnent sans qu'il ne s'aperçoive de rien.

7 - 2 - PRINCIPE DE LA COMMUTATION

La commutation se fait de la façon suivante :

Notation PAGE Ø = 1ère page de 4K de $DØØØ à $DFFF du M/DOS 6502

PAGE 1 = 2ème page de 4K de $DØØØ à $DFFF du M/DOS 6502

RACINE = reste du DOS (8K) de $EØØØ à FFFF

BASIC = 12K de $DØØØ à $FFFF

Ces quatre parties contiennent bien sûr des informations différentes bien qu'elles soient aux mêmes adresses.

Soit X le numéro du slot où la carte a été installée.
Des commutations se font en écrivant à l'adresse $CXFF

```
LDA # $80
STA$CXFF ..... BASIC

LDA # $00
STA$CXFF ..... PAGE 0 + RACINE

LDA # $01
STA$CXFF ..... PAGE 1 + RACINE
```

7 - 3 - RESET ET INTERRUPTIONS

La mise en route ou le RESET se font dans les ROM de la carte M/DOS 6502. Un programme s'y trouve donc pour gérer les interruptions.

RESET : le programme ramène au RESET de la ROM MONITEUR de l'APPLE. La présence de la carte est donc tranparente.

NMI : L'interruption NMI ramène à l'indirection habituelle en page $3

IRQ : Cette interruption sert à la fois aux interruptions extérieures masquables ainsi qu'au BREAK logiciel.
Lorsque la carte est connectée et sélectée (en page 0 et 1) le BREAK est impossible. L'IRQ ramène donc toujours à l'indirection prévue à cet effet en page $3.

IMPORTANT : Si la carte est sélectée, ce sont les vecteurs d'interruption de la carte qui servent; si elle est désélectée (mode BASIC), ce sont les vecteurs d'origine (ROM MONITEUR). Cependant, au moment d'une interruption, (sauf RESET), l'état reste inchangé et vous pouvez aussi bien être en mode BASIC qu'en mode DOS. Si vous désirez dans vos programmes d'interruptions, utiliser des fonctions MONITEUR vous devez :

1) chercher en quel état vous êtes. Pour cela, lisez un octet dans la page $DXXX à un endroit où les 3 versions sont différentes.

2) commuter vers BASIC par :

```
LDA   $8Ø
STA $CXFF
```

3) avant de finir l'interruption; reconnectez, si besoin est, la PAGE précédente par :

```
LDA   $ØØ ou $Ø1
STA $CXFF
```

7 - 4 - L'OCCUPATION RAM DU M/DOS 6502

Pour fonctionner, le M/DOS 6502 utilise les zones RAM suivantes :

a) .$Ø3ØØ - $Ø3CF

b) .$Ø250 - $Ø2FF

La zone a) en page 3 ne doit pas être modifiée. Par contre, la zone en page 2 est composée de variables de travail et peut être détruite entre deux ordres. (Elle l'est d'ailleurs car cette zone sert également de BUFFER pour INPUT.)
D'autre part, le DOS a besoin d'autres places en RAM pour :

1) Créer des objets en mémoire :

- descriptifs de disques
- descriptifs de fichiers
- masques
- ...

Cette zone sera appelée BUFFERS du DOS.

2) Accéder aux périphériques

Il faut prévoir en RAM les programmes permettant de lire, écrire ou formatter un disque. Dans le cas des lecteurs 5 pouces, il s'agit du programme RWTS fourni avec les disquettes.

Ces deux zones RAM ne sont pas en place fixe comme celle en page 2 et 3. L'utilisateur choisit lui-même suivant la version désirée leur implantation. En page 3, on trouve un certain nombre de pointeurs indiquant au DOS l'endroit où se trouvent ces zones.

7 - 5 - POSITION DES BUFFERS DU DOS

Note : Les pointeurs sont toujours dans l'ordre poids fort, poids faible.

$3ØØ - $3Ø1 = début des buffers du DOS

$3Ø3 - $3Ø4 = fin de la première partie

$3Ø5 - $3Ø6 = pointeur courant (indique le début de la place non encore occupée dans les buffers)

$3Ø7 - $3Ø8 = fin des buffers du DOS

Pour mettre en place ces pointeurs, choisissez les valeurs mini et maxi que vous réservez à ces adresses et placez-les en :

$3ØØ - $3Ø1
$3Ø7 - $3Ø8

placez en $3Ø3 - $3Ø4 les mêmes valeurs qu'en $3Ø7 - $3Ø8

placez en $3Ø5 - $3Ø6 l'adresse correspondant à celle indiquée en $3ØØ - $3Ø1 plus $124.

EXEMPLE :
$3ØØ : 9Ø
$3Ø2 : ØØ } $9ØØØ
$3Ø2 :
$3Ø3 : B6
$3Ø4 : ØØ } $B6ØØ
$3Ø5 : 91
$3Ø6 : 24 } $9124 = $9ØØØ + $124
$3Ø7 : B6
$3Ø8 : ØØ } $B6ØØ

Dans cet exemple, les parties réservées aux BUFFERS du DOS s'étendent de :

$9ØØØ à $B6ØØ (inclus)

7 - 6 - LES ACCES AUX PERIPHERIQUES

Le M/DOS 6502 permet de gérer simultanément des périphériques de types différents. Pour décrire ces périphériques, un certain nombre de tables sont prévues. Ces tables indiquent pour chaque périphérique :

- le nombre de pistes
- le nombre de têtes
- le nombre de secteurs
- l'adresse du programme permettant d'y accéder

M/DOS 6502 permet d'utiliser dans une même configuration 6 périphériques.
Par exemple : - Un disque dur à 3 drives
- 4 lecteurs 5 pouces

Pour chaque périphérique (numéroté de Ø à 5), dont nous appellerons n le numéro, indiquez :

en $31Ø + n le nombre de pistes - 1

en $316 + n le nombre de têtes

en $31C + n le nombre de secteurs

en $322 + 2 n l'adresse (poids fort, poids faible) du programme d'accès (HANDLER)

Si l'adresse correspondant au HANDLER est remplacée par $ØØØ, $ØØØ, le périphérique sera considéré comme inexistant.

7 - 7 - CARACTERISTIQUES DU PROGRAMME D'ACCES AU PERIPHERIQUE

Le programme gérant le périphérique doit être capable de lire, écrire ou formatter un disque d'après les paramètres suivants :

$3Ø9 numéro du périphérique (permet de différencier les actions pour plusieurs périphériques pour lesquels l'adresse du HANDLER est la même)

$3ØA piste choisie

$3ØB tête choisie

$3ØC secteur choisi

$3ØD commande : 1 - lecture
2 - écriture
3 - formattage

$3ØE poids fort de l'adresse du bloc

$3ØF poids faible de l'adresse du bloc

Il est inutile d'effectuer des contrôles de validité, ceux-ci sont faits par le M/DOS 6502 avant de faire appel à ces fonctions. Vous trouverez ci-après les versions permettant de faire fonctionner les disquettes 5 pouces APPLE 110K ou 140K en utilisant le programme RWTS.

IMPORTANT : En ce qui concerne le formattage, il est nécessaire que tous les secteurs puissent être lus, même s'ils n'ont pas été remplis. En conséquence, vous constaterez que dans la version 110K, le formattage est suivi de l'écriture de tous les secteurs de la disquette, car, en formattage 110K, la lecture est impossible sans écriture préalable.
Le cas se présente également pour les cartouches du disque CII D140.

```
rwts.....page 0001

line# loc   code          line


0001  0000                      * = $b700
0002  b700                ;
0003  b700                drive  = $309
0004  b700                cylind= $30a
0005  b700                tete   = $30b
0006  b700                secteu= $30c
0007  b700                comman= $30d           ; 1 2 ou 4
0008  b700                adblc  = $30e          ;adresse du bloc poids fort / poids faible
0009  b700                ;
0010  b700                ;
0011  b700                ; acces au periferique ( version 110k )
0012  b700                ;
0013  b700                diob=$b7e8
0014  b700                ;
0015  b700  a2 60             ldx #$60          ;calcul du slot
0016  b702  ad 09 03          lda drive
0017  b705  c9 02             cmp #2
0018  b707  90 02             bcc vrwts1
0019  b709  a2 50             ldx #$50
0020  b70b  8e e9 b7    vrwts1 stx diob+1
0021  b70e  ae 0d 03          ldx comman
0022  b711  8e f4 b7          stx diob+12
0023  b714  ae 0a 03          ldx cylind
0024  b717  8e ec b7          stx diob+4
0025  b71a  a2 01             ldx #1
0026  b71c  4a                .byte $4a         ; lsr a
0027  b71d  90 01             bcc rwts07
0028  b71f  e8                inx
0029  b720  8e ea b7    rwts07 stx diob+2
0030  b723  ad 0c 03          lda secteu
0031  b726  8d ed b7          sta diob+5
0032  b729  a9 00             lda #0
0033  b72b  8d eb b7          sta diob+3        ;volume
0034  b72e  ad 0e 03          lda adblc
0035  b731  8d f1 b7          sta diob+9
0036  b734  ad 0f 03          lda adblc+1
0037  b737  8d f0 b7          sta diob+8
0038  b73a  a9 b7             lda #>diob
0039  b73c  a0 e8             ldy #<diob
0040  b73e  20 b5 b7          jsr $b7b5
0041  b741  b0 08             bcs vrw005
0042  b743  ad 0d 03          lda comman
0043  b746  c9 04             cmp #4            ; format ?
0044  b748  f0 02             beq vrw001
0045  b74a  18                clc
0046  b74b  60          vrw005 rts
0047  b74c                ;
0048  b74c  a9 00         vrw001 lda #0
0049  b74e  8d 8f b7          sta cylcyl
0050  b751  a9 00         vrw004 lda #0
0051  b753  8d 8e b7          sta secsec
0052  b756  a9 02         vrw003 lda #2
0053  b758  8d f4 b7          sta diob+12
0054  b75b  a9 08              lda #8
0055  b75d  8d f0 b7          sta diob+8
```

```
rwts......page 0002

line# loc    code            line


0056  b760  8d f1 b7          sta diob+9
0057  b763  ad 8f b7          lda cylcyl
0058  b766  8d ec b7          sta diob+4
0059  b769  ad 8e b7          lda secsec
0060  b76c  8d ed b7          sta diob+5
0061  b76f  a9 b7             lda #>diob
0062  b771  a0 e8             ldy #<diob
0063  b773  20 b5 b7          jsr $b7b5
0064  b776  b0 15             bcs vrw002
0065  b778  ee 8e b7          inc secsec
0066  b77b  ad 8e b7          lda secsec
0067  b77e  c9 0d             cmp #13
0068  b780  90 d4             bcc vrw003
0069  b782  ee 8f b7          inc cylcyl
0070  b785  ad 8f b7          lda cylcyl
0071  b788  c9 23             cmp #35
0072  b78a  90 c5             bcc vrw004
0073  b78c  18                clc
0074  b78d  60          vrw002 rts
0075  b78e                   ;
0076  b78e  00          secsec .byte 0
0077  b78f  00          cylcyl .byte 0
0078  b790                   ;
0079  b790                   .end


errors = 0000


symbol table

symbol  value
 adblc   030e   comman  030d   cylcyl  b78f   cylind  030a
 diob    b7e8   drive   0309   rwts07  b720   secsec  b78e
 secteu  030c   tete    030b   vrw001  b74c   vrw002  b78d
 vrw003  b756   vrw004  b751   vrw005  b74b   vrwts1  b70b


end of assembly
```

7 - 8 - DESCRIPTION DE LA GESTION D'ECRAN

La carte M/DOS est prévue pour pouvoir adapter la gestion de masques à n'importe quel type de carte 80 colonnes ou terminal intelligent. Pour cela, un certain nombre de paramètres sont à mettre en place pour décrire l'écran.

$3 68	adresse de l'écran (poids fort, poids faible)
$3 6A	longueur de l'écran (poids fort, poids faible)
$3 6C	nombre de caractères par lignes

Un certain nombre de fonctions sont également à créer pour gérer l'écran. Elles utilisent des pointeurs en page Ø notés :

VV ($DB,$DC)
WW ($DD,$DE)

Le premier (VV) indique la position du curseur (poids faible, poids fort).
Le second (WW) indique le nombre de caractères séparant le curseur du premier caractère de l'écran (poids faible, poids fort).
WW évolue donc entre Ø et la largeur de l'écran - 1.
La position du curseur peut être fictive ou réelle. Dans la version 40 colonnes fournie en standard, VV correspond à l'adresse réelle en mémoire de l'octet se rapportant au caractère de l'écran.
La valeur initiale de VV (caractère en haut à gauche de l'écran) doit correspondre à la valeur indiquée dans le paramètre " adresse de l'écran ".
A chaque opération concernant les masques, le M/DOS 6502 initialisera VV à cette valeur et WW à zéro.
D'autre part, le M/DOS 6502 doit savoir se déplacer sur cet écran qui lui est inconnu. Il doit pouvoir aussi lire ou écrire sur cet écran. Il doit finalement pouvoir convertir les caractères lus en ASCII et inversement.

Nous noterons :

CODE ECRAN : la valeur qu'il faut écrire sur l'écran un caractère dont on connait le code ASCII.

Il faut donc fournir au M/DOS les fonctions suivantes :

AVANCE : calcule les nouvelles valeurs de VV et WW lorsque l'on avance de 1 caractère sur l'écran. Le carry sera égal à 1 en sortie si l'on dépasse la fin de l'écran. (WW = longueur d'écran) (la longueur de la zone est en NC = Ø26A sur deux octets poids fort, poids faible).

RECULE : calcule les nouvelles valeurs de VV et WW lorsque l'on recule de 1 caractère sur l'écran. Le carry sera égal à 1 en sortie si l'on dépasse le début de l'écran. (Il s'agit du cas où (WW) = Ø en entrant dans le programme RECULE.)

INCAR : renvoie dans l'accumulateur le code écran du caractère situé en VV.

OUTCAR : écrit le caractère dont le code écran est dans l'accumulateur à l'endroit du curseur pointé par VV.

E A : convertit dans l'accumulateur un code écran en code ASCII.

A E : convertit dans l'accumulateur un code ASCII en code écran.

INVERS : convertit dans l'accumulateur un code écran en code écran du caractère sur fond blanc correspondant.

FLASH : convertit dans l'accumulateur un code écran en code écran du caractère clignotant correspondant. (Si cette fonction n'existe pas, remplacez-la, par exemple, par INVERS.)

Vous devez écrire les programmes assembleurs réalisant ces fonctions et les implanter en mémoire. Pour indiquer au M/DOS 6502 leurs positions remplissez l'ensemble des JMP prévues à cet effet.

```
ø36F      JMP AVANCE
ø372      JMP RECULE
ø384      JMP INCAR
ø387      JMP OUTCAR
ø378      JMP EA
ø375      JMP AE
ø37E      JMP INVERS
ø381      JMP FLASH
```

NOTE : Pour simplifier votre programmation, sachez que lorsque le M/DOS 6502 se branche à l'INCAR ou OUTCAR, le registre y est nul. Donc, si VV est l'adresse réelle du caractère, on pourra choisir :

```
INCAR  LDA (VV), y
       RTS

OUTCAR STA (VV), y
       RTS
```

que l'on pourra placer directement à la place des JMP correspondants pour accélérer, car dans les deux cas, la longueur est 3 octets. Ci-joint la version 40 colonnes de base contenu dans la ROM.

IMPORTANT : Lors de l'appel de ces fonctions, le M/DOS 6502 ne reconnecte pas le BASIC (pour gagner du temps), donc vous ne pouvez pas :

- utiliser les fonctions moniteurs
- tester vos programmes en utilisant l'instruction BRK (code øøø).

Essayez d'écrire des programmes les plus performants possibles car dans l'exemple d'une insertion de caractères sur tout l'écran en 80 colonnes, on aura :

- 2 000 appels à AVANCE
- 2 000 appels à RECULE
- 2 000 appels à INCAR
- 2 000 appels à OUTCAR

```
ready.

 1000 ; version de base en 40 colonnes
 1010 ;
 1020 ; ae40c
 1030 ; inv40c
 1040 ; fla40c
 1050 ; set40c
 1060 ; ea40c
 1070 ; av40c
 1080 ; rec40c
 1090 ;
 1100 ;
 1110 ;ascii - ecran
 1120 ;
 1130 ae40c  and #63
 1140        ora #128
 1150        rts
 1160 ;
 1170 ; inverse video
 1180 ;
 1190 inv40c and #63
 1200        rts
 1210 ;
 1220 ; flash video
 1230 ;
 1240 fla40c and #63
 1250        ora #64
 1260 rts
 1270 ;
 1280 ;set un caractere avec spot en (vv)
 1290 ;
 1300 set40c ldy #0
 1310        jsr incar
 1320        sta n4
 1330        jsr flash
 1340        jsr outcar
 1350 sel
 1360 etq5   bit 49152
 1370        bpl sel
 1380        lda 49152
 1390        bit 49168
 1410 seqwar and #$7f
 1420        tax
 1430        lda n4
 1440        jsr outcar
 1450        txa
 1460        rts
 1470 ;
 1480 ;conversion e a
 1490 ;
 1500 ea40c  and #127
 1510        cmp #32
 1520        bcs etq36
 1530        ora #64
 1540 etq36  rts
 1550 ;
 1560 ;
 1570 ;
 1580 ;avance /recule de un caractere
 1590 ;
 1600 av40c  lda ww+1
```

```
1610          cmp nc
1620          bcc avok
1630          lda ww
1640          cmp nc+1
1650          bcc avok
1660          rts
1670 avok     lda vv+1
1680          cmp #7
1690          bne av1
1700          lda vv
1710          cmp #$a7
1720          beq av2
1730          cmp #$cf
1740          bne av1
1750 av2      sec
1760          sbc #216
1770          sta vv
1780          lda vv+1
1790          sbc #3
1800          sta vv+1
1810          av1 lda vv
1820          and #127
1830          cmp #$27
1840          beq av3
1850          cmp #$4f
1860          beq av3
1870          cmp #$77
1880          bne av4
1890 av3      lda vv
1900          clc
1910          adc #88
1920          sta vv
1930          bcc av4
1940          inc vv+1
1950 av4      inc ww
1960          bne etq21
1970          inc ww+1
1980 etq21    inc vv
1990          bne etq7
2000          inc vv+1
2010 etq7     clc
2020          rts
2030 ;
2040 ;recule de 1 caractere"
2050 ;
2060 rec40c ldx ww+1
2070          bne reok
2080          ldx ww
2090          cpx #1
2100          bne reok
2110          sec
2120          rts
2130 reok     lda vv+1
2140          cmp #4
2150          bne re1
2160          lda vv
2170          cmp #$28
2180          beq re2
2190          cmp #$50
2200          bne re1
2210 re2      clc
2220          adc #128
```

```
2230          sta vv
2240          lda vv+1
2250          adc #3
2260          sta vv+1
2270          jmp re4
2280 re1      lda vv
2290          and #127
2300          beq re3
2310          cmp #$28
2320          beq re3
2330          cmp #$50
2340          bne re4
2350 re3      lda vv
2360          sec
2370          sbc #88
2380          sta vv
2390          bcs re4
2400          dec vv+1
2410 re4      dec ww
2420          ldx ww
2430          cpx #255
2440          bne etq12
2450          dec ww+1
2460 etq12    dec vv
2470          ldx vv
2480          cpx #255
2490          bne etq9
2500          dec vv+1
2510 etq9     clc
2520          rts
2530 ;
2540 .end

ready.
```

7 - 9 - MODIFICATIONS DE L'ENTREE CLAVIER

L'accès clavier dans les saisies par masques se fait par l'intermédiaire d'une indirection en page 3. Pour modifier cette fonction, placez dans l'indirection l'adresse de votre programme (poids faible, poids fort)

```
Ø37B     JMP  GET
```

Le programme doit renvoyer dans l'accumulateur soit le code ASCII, soit le code ASCII + 128.

NOTE : La fonction AZERTY pourra être utilisée quelque soit la fonction GEØT que vous avez créée.

7 - 10 - AJOUT DE NOUVELLES COMMANDES AU M/DOS 6502

Vous pouvez, si vous le désirez, ajouter vous-même des fonctions au DOS. Pour cela, il existe une indirection sur deux octets (poids faible, poids fort) indiquant l'adresse du programme analysant les commandes. Si vous voulez ajouter une commande :

1) sauvez l'adresse précédemment placée dans cette indirection.

2) mettez l'adresse de votre programme.

3) à la fin de l'éxécution de votre programme, si la commande ne vous concernait pas, rebranchez-vous à l'adresse que vous avez sauvée si elle n'est pas nulle.

Cette méthode permet de faire fonctionner ensemble plusieurs programmes ajoutant de nouvelles commandes.

NOTE : Dans la version de base, l'indirection contient Ø,Ø. Dans ce cas, le M/DOS 6502 ignore l'indirection.

L'adresse de l'indirection est :

- Ø38A poids faible
- Ø380 poids fort

Lorsque vous prenez la main dans votre programme :

- l'accumulateur contient le premier caractère de la commande
- AA(Ø272) contient la longueur de l'instruction
- yy(ØE1,ØE2) contient l'adresse de l'instruction
- le registre y pointe vers le caractère en cours dans l'instruction par rapport à yy.

4 fonctions peuvent vous être utiles :

JSR CARAC : - renvoie dans l'accumulateur le prochain caractère non blanc de la commande et met à jour y.

- si le carry est 1, on a atteint la fin de la commande.

JSR SEPARA : - renvoie le prochain caractère non blanc de la commande suivant un séparateur. (, : -) dans les mêmes conditions.

JSR VALEUR : - renvoie une valeur sur 1 octet lu dans la commande en décimal à partir de l'octet en cours.

JSR GETHX1 : - idem pour une valeur sur deux octets saisis sous la forme ØXXXX.
En sortie X = poids faible.
A = poids fort.

EXEMPLE : La commande est : LET"H,valeur,une lettre,héxadécimal"
LET"H,12,A,Ø25FØ"

```
cmp   'H
bne   NON
jsr   SEPARA
bcs   ERR
sta   VAL
jst   SEPARA
bcs   ERR
                        sta   LETTRE
                        jsr   SEPARA
                        bcs   ERR
                        jsr   GETHX1
                        stx   HEXA
                        sta   HEXA + 1
                        jmp   OK
                  NON   rts
```

```
ready.

 1000 carac   cpy aa
 1010         bcc ee72
 1020         rts
 1030 ee72    lda (yy),y
 1040         iny
 1050         cmp #'
 1060         beq carac
 1070         clc
 1080         rts
 1090 ;
 1100 ;
 1110 separa  jsr carac
 1120         bcc ee80
 1130         rts
 1140 ee80    cmp #'          ;
 1150         beq carac
 1160         cmp #'-
 1170         beq carac
 1180         cmp #58
 1190         beq carac
 1200         cmp #44
 1210         beq carac
 1220         bne separa
 1230 ;
 1240 ;
 1250 valeur  cmp #58
 1260         bcs val1
 1270         cmp #48
 1280         bcc val1
 1290         sbc #48
 1300         sta nuacc
 1310 va333   jsr carac
 1320         bcs val2
 1330         cmp #58
 1340         bcs val2
 1350         cmp #48
 1360         bcc val2
 1370         sbc #48
 1380         sta aa+1
 1390         lda nuacc
 1400         cmp #26
 1410         bcs val1
 1420         asl a
 1430         asl a
 1440         asl a
 1450         adc nuacc
 1460         adc nuacc
 1470         adc aa+1
 1480         bcs val1
 1490         sta nuacc
 1500         jmp va333
 1510 val1    sec
 1520         rts
 1530 val2    clc
 1540         rts

ready.
```

Dans le cas où l'instruction n'est pas pour vous :

- retourner à l'indirection précédente si celle-ci existait.

- sinon, faites simplement RTS

Dans le cas où l'ordre est pour vous :

- exécutez celui-ci

- puis faites : - pla
  - pla
  - RTS

  pour terminer l'ordre correctement.

A N N E X E

```
1000 ;
1010 ; adresses des parametres
1020 ;
1030 zwdeb  = $300
1040 dimdef = $302
1050 zt     = $303
1060 lm     = $305
1070 zf     = $307
1080 drive  = $309
1090 cylind = $30a
1100 tete   = $30b
1110 secteu = $30c
1120 comman = $30d
1130 adblc  = $30e
1140 ;
1150 ;tables des drives
1160 ;
1170 tcylm  = $310          ;012345 ......
1180 ttetem = $316          ;6789ab ......
1190 tsectm = $31c          ;cdef01 ......
1200 trwts  = $322          ;234567 89abcd
1210 ttype  = $32e          ;ef0123 ......
1220 ;
1230 jsrmon = $334          ; jsr combas
1240 ;              jsr $0000
1250 ;              lda #0
1260 ;        comb1 sta $c200
1270 ;              rts
1280 ;       combas lda #$80
1290 ;              bne comb1
1300 ;
1310 slotcd=jsrmon+10
1320 pasecd=jsrmon+7
1330 comb1 =jsrmon+8
1340 combas=jsrmon+12
1350 ;
1360 aplexe = $344
1370 ;
1380 ;
1390 ; indirection retour systeme
1400 ;
1410 jmpsy  = $34a
1420 ;
1430 ;
1440 ;
1450 ;caracteres ecran utilises
1460 ;
1470 zblanc = $350          ; 160
1480 zapost = $351          ; 162;''
1490 zetoil = $352          ; 170;'*
1500 zzero  = $353          ;  48;'0
1510 zneuf  = $354          ; 185;'9
1520 zplus  = $355          ; 171;'+
1530 zadres = $356          ; 128;'@
1540 zdollr = $357          ; 164;'$
1550 zpourc = $358          ; 165;'%
1560 zslash = $359          ; 175;'/
1570 zinfer = $35a          ; 188;'<
1580 zsuper = $35b          ; 190;'>
```

```
1590 zdeuxp = $35c              ; 186;'';
1600 zpvirg = $35d              ; 187;';
1610 zinter = $35e              ; 191;'?
1620 zpoint = $35f              ; 174;'.
1630 ztrans = $360              ;   0; transparent
1640 manuel = $361              ;   0
1650 ;
1660 call   = $362              ; !!!!!!!!!!!!!
1670 ;call 941
1680 ;
1690 ; call .byte 0,140;-< sys >-
1700 ; .byte '941'
1710 ; .byte 0,0,0
1720 ;
1730 ;
1740 ;caracteristiques ecran
1750 ;
1760 adrecr = $368              ; 4,0
1770 lngecr = $36a              ; 3,192
1780 nlscrn = $36c              ; 40
1790 cinp   = $36d              ; $1b,$fd
1800 ;
1810 ; fonction indirectes
1820 ;
1830 ;
1840 avance = $36f              ; jmp av40c
1850 recule = $372              ; jmp rec40c
1860 ae     = $375              ; jmp ea40c
1870 ea     = $378              ; jmp ae40c
1880 getind = $37b              ; jmp get40c
1890 invers = $37e              ; jmp inv40c
1900 flash  = $381              ; jmp fla40c
1910 incar  = $384 lda (vv),y
1920 ;                rts
1930 outcar = $387              ; sta (vv),y
1940 ;                rts
1950 indcom = $38a
1960 ;              0
1970 ;              0
1980 ;
1990 ;
2000 ;indirection
2010 ;
2020 jmpent=$3ad;vers deb prg ass
2030 ;
2040 ;
2050 ;
2060 ; impressions ( compteur de lignes )
2070 ;
2080 nblgn  = $3b3
2090 ;
2100 ;
2110 nbdeci = $3b4
2120 ;
2130 evamsk = $3b5
2140 ;
2150 azerty = $3b6
2160 ;
2170 ; prg acces azerty
2180 ;
2190 aplaze = $3b7
2200 ;
2210 ; 1er libre en $3be
```

CHAPITRE VIII

LES POINTEURS DU DOS

8 - 1 - UTILISATION DE LA MEMOIRE

L'espace mémoire de votre micro-ordinateur est utilisé de la façon suivante :

ADRESSE (HEXADECIMALE)	
\$0000 – \$00FF	page zéro - pointeurs divers
\$0100 – \$Ø1FF	pointeurs internes
\$Ø200 – \$Ø2FF	buffers
\$Ø300 – \$Ø3FF	pointeurs
\$Ø400 – \$Ø7FF	mémoire écran
\$Ø800 – fin programme	texte du programme BASIC
début variable – fin variable	variables du BASIC
début buffers – fin de buffers	buffers du DOS
début des handlers – fin des handlers	HANDLERS
\$C000 – \$CFFF	I/O
\$D000 – SFFFF	BASIC et MONITOR OU DOS

Pour plus de détails sur les pointeurs, utilisés par le BASIC, se reporter au manuel d'utilisation de celui-ci.
Deux pointeurs sont particulièrement intéressants pour l'utilisateur puisqu'ils peuvent être modifiés.

- le pointeur de fin de programme BASIC
- le pointeur de début des buffers du DOS

Normalement, ces deux pointeurs doivent être les-mêmes, les buffers du DOS commençant là où finit le BASIC.

Adresse des pointeurs (en décimal)

115 : fin du BASIC poids faible
116 : fin du BASIC poids fort

768 : début des buffers du DOS poids fort
769 : début des buffers du DOS poids faible

Pour modifier ces valeurs, faites par exemple :

```
POKE 115, Ø : POKE 116,50 : CLEAR
POKE 769, Ø : POKE 768,50 : LET"#C,$"
```

Les deux ordres CLEAR et LET"#C,$" sont indispensables pour que les modifications soient prises en compte.

NOTE : Il est possible de laisser une place libre entre la fin du BASIC et le début des buffers du DOS.

8 - 2 - DIMENSION PAR DEFAUT DES TABLEAUX

Lors de l'ouverture ou de la création d'un fichier, les tableaux qui n'ont pas été dimensionnés au préalable le sont automatiquement.

Ce sont alors des tableaux à 1 dimension dont l'indice maximum est 8.

Cette valeur par défaut peut être changée . Faire :

```
POKE 770,[nombre d'indice désiré]
```

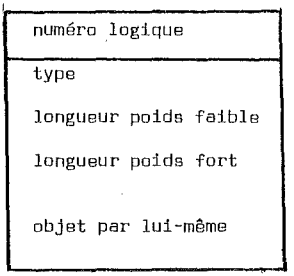
CHAPITRE IX: PRINCIPE DE FONCTIONNEMENT DU M/DOS 6502

9 - 1 - a STRUCTURE DES OBJETS EN MEMOIRE CENTRALE

Les différents types d'objets contenus en memoire sont :

1) les fichiers : type = " F "
2) les masques : type = " M "
3) les descriptions de disque : type = " $ "

Chaque objet est rangé sous la forme :



Les objets sont placés consécutivement en mémoire.
Pour retrouver un objet, la méthode est la suivante :

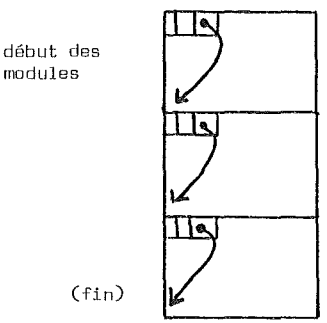
- accès au premier élément
- comparaison du numéro logique

  Si celui cherché : fin
  Sinon, ajouter à l'adresse de départ la longueur totale qui permet d'accèder à l'élément suivant.

Et ainsi de suite...

Deux pointeurs permettent de connaitre les limites des objets :

- les pointeurs de début des modules ($300,$301) + 292 ($124)
- les pointeurs du premier libre



Le premier module commence à l'adresse :

PEEK (768)*256 + PEEK (769) + 1028

Destruction d'un objet :

L'ordre LET"#C,[n° logique]

permet de récupérer la place d'un objet.

Pour cela, les objets suivants en mémoire sont décalés de la longueur de l'objet détruit et le premier libre est diminué de la même valeur.

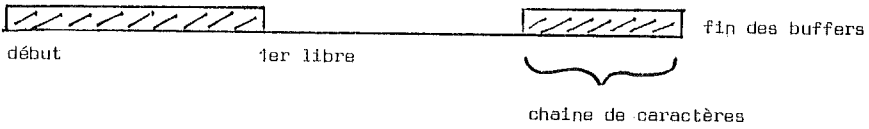
Double utilisation des buffers

La place disponible dans les buffers entre le pointeur du premier libre et la fin des buffers est également :

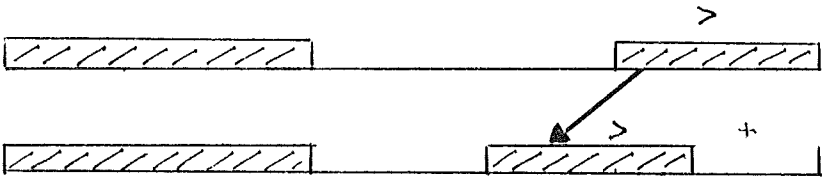
Elle sert :

- à enregistrer les descriptions de fichiers
- à enregistrer des ordres pour l'EXECUTE

Par exemple, l'ordre LET" > ..... "
va transférer la chaine de caractères suivant le " > " dans cet espace, cadrée à droite.



L'ordre LET" +.... " permet de compléter cette chaine. Pour cela, on décale vers la gauche la chaine déjà enregistrée, puis on insère la nouvelle.



Trois ordres permettent d'inscrire des caractères dans ce buffer :

LET" > ....."
LET" + ..."
LET"ENTER-[n° logique]

Ce dernier ordre décode le dictionnaire d'un fichier déjà enregistré et le place dans le buffer.

Dans le cas ou la place disponible entre le premier libre et la fin des buffers est insuffisante, le programme le signale par le message :

? OUT OF MEMORY ERROR

9 - 1 - b STRUCTURE DES FICHIERS / DCB

1) Description générale

Un fichier M/DOS est découpé en un certain nombre d'unités de base que nous appelerons PISTE LOGIQUE.
Ces pistes peuvent ou non correspondre à des pistes physiques du disque.

CREATION DU FICHIER

A la création du fichier, le système réserve deux pistes logiques :
- la première est réservée aux clés ou pointeurs
- la seconde est réservée aux enregistrements

Au fur et à mesure du grossissement du fichier, un certain nombre de pistes logiques pourront être allouées dynamiquement au fichier.

Les fichiers M/DOS ne sont pas tous semblables puisque chacun a une définition d'enregistrement qui lui est propre.
Pour cela, un descriptif est attaché à chaque fichier.
Ce descriptif est appelé D.C.B. (Data Control Bloc). Il comporte un certain nombre d'éléments dont les principaux sont :

- La description de l'enregistrement
  Ces paramètres définissent les différentes clés, leur type,...
  Pour chaque variable de l'enregistrement, 7 octets sont réservés pour sa description.

- La liste des pistes logiques occupées par le fichier.
  Au fur et à mesure que des pistes sont allouées, leurs numéros sont ajoutés à cette liste, classés dans un ordre fictivement croissant pour permettre les recherches par clés.

- Les différents pointeurs permettant dans ce fichier de connaitre les blocs libérés à la suite de destruction ou non encore utilisés.

Une description de cette DCB est donnée en fin de chapître.

LES CLES OU POINTEURS

Les fichiers peuvent être de deux types :

- fichiers relatifs
- fichiers séquentiels indexés

Dans le cas des fichiers relatifs, la structure est la suivante :

- Les pistes de pointeurs contiennent un chainage sur l'enregistrement.
  Ce chainage est une suite de trois octets, ce qui permet de prévoir des extensions importantes.

Dans chaque bloc réservé aux pointeurs, on peut donc mettre 85 pointeurs.

Chaque pointeur renvoie sur l'enregistrement de la façon suivante :

- 1er paramètre : la piste logique où se trouve le début de l'enregistrement.
- 2ème paramètre : le numéro du bloc dans cette piste logique.

Lorsque vous demandez la lecture de l'enregistrement N, le système va calculer tout d'abord la piste logique où se trouve ce pointeur, puis lira le bloc correspondant. Le pointeur permettra alors de trouver l'enregistrement.

Cette méthode impose de lire deux blocs au moins pour accéder à un enregistrement, mais en revanche, elle n'impose aucune contrainte sur la longueur de l'enregistrement. C'est cet avantage qui permet au M/DOS 6502 de gérer des fichiers de taille variable.

Pour les fichiers séquentiels indexés, les pointeurs vers les enregistrements sont complètés par la clé. Cette clé peut être éventuellement une suite de sous-clés.

Le COEFFICIENT DE BLOCAGE des clés d'un fichier séquentiel indexé sera donc calculé de la façon suivante :

Soit L le nombre d'octets de la clé,
pour chaque article, on enregistre L + 3 + 2 octets
avec :
L octets pour la clé
3 octets pour pointer sur l'enregistrement
2 octets pour pointer vers le suivant

Le nombre de clés que l'on pourra mettre sur un bloc de 256 octets sera donc la partie entière de 256 / (L + 5)

FICHIERS MULTICLES

Dans le cas d'un fichier comportant plusieurs clés, chaque clé demandera un ensemble de pistes logiques.

PRINCIPE DE RECHERCHE PAR CLE

Les clés sont réparties en deux types :
- les clés triées
- les clés non triées (appelées TAS)

Les clés triées sont rangées par ordre croissant sur les pistes logiques.
Le principe de recherche sur ces clés est la recherche DICHOTOMIQUE.
Il consiste à comparer la clé recherchée avec une clé médiane, ce qui permet d'éliminer la moitié des clés.

Exemple :

La liste des clés enregistrées est la suivante :

A B G K P Q U V W Y Z

recherche de la clé I :
La clé médiane est Q : I est plus petit que Q : les clés supérieures à Q sont éliminées.

La clé médiane est G : I est plus grand que G : on élimine A.....G

La clé médiane est K : I est plus petit que K

Il n'y a plus de clés entre G et K, la clé recherchée n'existe pas.

Intérêt de la méthode :

A chaque essai, (qui demande au plus une lecture disque) on élimine la moitié du fichier. Le temps de recherche est donc de l'ordre de :

LOG (nombre de clés) x constante

Exemple : lecture sur disque dur (temps d'accès moyen 50 millisecondes) le coefficient de blocage est de 25 (clés de 5 octets, par exemple un flottant)

parmi	25 clés,	1 lecture	50 ms
	50 clés,	2 lectures	100 ms
	100 clés,	3 lectures	150 ms
	200 clés,	4 lectures	200 ms
	400 clés,	5 lectures	250 ms
	800 clés,	6 lectures	300 ms
	1600 clés,	7 lectures	350 ms
	3200 clés,	8 lectures	400 ms
	6400 clés,	9 lectures	450 ms
	12800 clés,	10 lectures	500 ms

Chaque fois que le fichier double de taille, le temps de recherche n'augmente que de une lecture disque.

Dans l'exemple (cas assez fréquent) le temps d'accès à un article parmi 10 000 est environ 1/2 seconde.

Problème du TAS : création d'articles

Lors d'une création d'article, le but est d'insérer une nouvelle clé dans la liste. Cependant, si le fichier comporte un nombre important d'articles, cette opération risque de demander un grand nombre de décalages. Par exemple, dans le cas décrit ci-dessus, avec 10 000 articles, pour insérer une nouvelle clé en début de liste il faudrait décaler 10 000 clés, soit 400 blocs de 256 octets ! Le temps d'insertion serait alors de 20 secondes, ce qui n'est pas admissible en utilisation normale.

De ce fait, la technique employée dans le M/DOS 6502 pour créer de nouvelles clés est légèrement différente.

Une nouvelle clé n'est pas insérée directement à sa place, mais ajoutée à la fin de la liste dans ce que nous appelerons le TAS.
Ces nouvelles clés ne sont pas triées, mais CHAINEES aux autres clés.

A la fin de la recherche dichotomique, le système connait la clé immédiatement inférieure et la clé immédiatement supérieure. La nouvelle clé du TAS est chainée à la clé immédiatement inférieure. Pour cela, on utilise le pointeur permettant de retrouver le suivant.

Cela modifie dont l'algorithme de recherche d'un article : après avoir cherché la clé dans la liste triée, il faut vérifier qu'aucun chainage ne conduit à la clé recherchée dans le tas.

Le temps d'accès à un article dont la clé est dans le tas est bien entendu moins performant que dans le cas de la liste triée.

Supposons par exemple que le fichier comporte 10 000 articles.
Nous en insérons 10 000 autres.
Pour les 10 000 premiers qui sont dans la liste triée, le temps d'accès est de 500 ms.
Les nouveaux articles, eux, sont dans le tas.

Si la répartition des clés est aléatoire, on peut espérer que chaque nouvelle clé se situera entre deux des clés triées.
Dans ce cas, la détérioration du temps d'accès sera en moyenne de un accès supplémentaire pour la lecture d'une clé du tas.
Le temps d'accès est alors :

500 ms (10 accès) pour une clé de la liste triée
550 ms (11 accès) pour une clé du tas.

Le système comporte un ordre : REORGANISATION qui permet de ranger les clés du tas. Pour cela, toutes les clés sont relues dans l'ordre croissant grâce aux pointeurs vers le suivant et sont réécrites sur de nouvelles pistes de clés.
Lorsque l'opération est terminée, la nouvelle liste de clés utilisées remplace la précédente dans la DCB et les pistes précédentes sont libérées.

REMARQUE : Le problème du tas est particulièrement sensible dans le cas d'un fichier initialement vide. En effet, dans ce cas, aucune clé n'est triée et le temps d'accès est alors en moyenne la moitié du temps d'une recherche séquentielle (le temps de recherche est alors proportionnel au nombre de clés !).

Le temps de réorganisation est facile à calculer : c'est le temps de lecture et d'écriture de toutes les clés. Pour 10 000 articles, il faut manipuler 400 blocs en lecture et écriture soit : 400 x 2 x 50 ms = 40 000 ms

En quarante secondes, le fichier est trié !

NOTE : en multiclés, l'opération de réorganisation est répétée sur chacune des clés.

Problème de la place disponible pour réorganiser

Pour que la réorganisation soit possible, il est nécessaire que le nombre de pistes libres soit suffisant pour recopier la liste des clés.
Au besoin, on peut récupérer des pistes libres en détruisant d'autres fichiers.

Il est important de noter que les différentes réorganisations d'un fichier font que les pistes utilisées pour les clés se déplacent sur le disque et ne restent pas figées. Or ces pistes sont fréquemment utilisées lorsque l'on fait des recherches par clé. Ce principe permet de répartir l'usure de la surface sur tout le disque (dans le cas des disques souples bien sur !)

Le CATALOGUE du disque étant également un fichier à accès par clé, le même phénomène se produit également dans le cas de chargements de programmes ou masques.

STRUCTURE DE L'ENREGISTREMENT

L'enregistrement a la même structure que le fichier soit relatif, ou séquentiel indexé.

Une caractéristique de l'enregistrement est le coefficient de blocage. Celui-ci représente le nombre d'octets contenus dans un bloc de base.

Les enregistrements sont de tailles variables, il n'est donc pas possible de définir exactement la taille du bloc de base (élément le plus petit d'un enregistrement).

Pour des raisons d'optimisation, la taille du bloc de base pourra être :

- 32 octets
- 64 octets
- 128 octets
- 256 octets

Sur ces octets définissant le bloc de base, trois sont réservés pour les chainages internes.

Le fonctionnement est le suivant :

1) L'article est plus petit que le bloc de base.
   Il ne demandera alors qu'un seul bloc et le chainage interne indiquera fin d'article.

2) L'article est supérieur au bloc de base.
   Il faudra alors plusieurs blocs de base pour le contenir et ceux-ci seront chainés ensemble. Le chainage interne du premier pointant sur le second et ainsi de suite. Le dernier bloc de base ne contient pas de chainage significatif, comme dans le cas d'un bloc de base unique.

Lors d'une mise à jour d'un article, celui-ci pouvant changer de dimension, tous les blocs de base sont désalloués puis repris en fonction des besoins.

La taille du bloc de base est une constante du fichier (contenuedans la DCB). Celle-ci est déterminée automatiquement lors de la création.

L'algorithme de calcul est le suivant :

En ne tenant compte que des variables de l'enregistrement,

chaque flottant vaut 5 octets
chaque chaine de caractères est approximée à 12 octets
chaque binaire simple vaut 1 octet
chaque binaire double vaut 2 octets
chaque date vaut 2 octets

Les tableaux sont supposés contenir en moyenne trois éléments.

Le coefficient de blocage choisi est celui immédiatement supérieur à la valeur obtenue dans la liste des possibles (32, 64, 128, 256) en tenant compte des 3 octets de chainage par bloc de base.

Intérêt de l'optimisation du bloc de base

Plus le bloc de base est petit, plus l'utilisation de l'espace disque sera optimisée.(moins de place perdue)

Plus le bloc de base est grand, plus le temps d'accès sera petit. (en effet, les différents blocs de base ne se trouvent pas forcément dans le même bloc physique, dans ce cas, il faudra plusieurs lectures disques. De plus, en cas d'écriture, il faut lire les blocs avant de les réécrire.)

CODAGE DES INFORMATIONS

Dans le but de minimiser la place perdue sur le disque, les informations sont compactées.
Pour lire ou écrire un article, le système se fie au dictionnaire (qui décrit l'enregistrement)
Les variables simples sont écrites directement sans séparateur.

Exemple :

un flottant A ..... 5 octets
un binaire simple ..... 1 octet
une date ..... 2 octets
un binaire double ..... 2 octets
une chaine de caractères ... 1 + la longueur de chaine

(l'octet supplémentaire indique la longueur)
les blancs non significatifs sont supprimés (aux extrémités)

Dans le cas des tableaux le codage est le suivant :
Les variables du tableau sont lues dans l'ordre.

- si n indices du tableau correspondent à des variables non nulles ou non vides consécutives, ces variables sont codées comme une suite de variables simples.

- si n variables successives sont nulles ou vides :

  - si n est plus petit que 128
    les variables sont codées sur 2 octets : 0, n

  - si n est plus grand que 127
    les variables sont codées sur 3 octets : 0, (poids fort de n+128)
    (poids faible de n)

- fin de tableau :
  La fin de tableau est repérée par deux 0 consécutifs (configuration impossible dans le codage de valeurs nulles ou vides)

CODAGE DES DATES

Les dates sont codées sur 2 octets.

A partir de la date standard 21/02/81 la transformation est :

AAAAAAAM	MMMJJJJJ
01234567	01234567

L'année est codée sur 7 bits (0 à 128)
Le mois est codé sur 4 bits (0 à 16)
Le jour est codé sur 5 bits (0 à 32)

Le codage de l'année se fait en ajoutant 50 aux années inférieures à 50 et en enlevant 50 aux années supérieures à 50.

De ce fait, les années 00 à 49 sont considérées comme 2000 à 2049
les années 50 à 99 sont considérées comme 1950 à 1999

Les années 2000 et plus sont donc bien supérieures aux années 80-90.

Importance de l'ordre

Si vous créez un fichier séquentiel indexé dont une clé (ou une partie d'une clé) est une date, les articles seront automatiquement classés.

Exemple concret de codage d'enregistrement

a) Définition

L'enregistrement est défini par :

```
LET" A = A$,B,C%,A ; , B$ ;
```

Ecrivons sur le fichier l'enregistrement suivant :

```
B = 1
A$ = " TEST "
C% = 10
A(0) = 1, A(6) = 2, les autres éléments sont nuls.
B$ (0) = " TUTU ", les autres éléments sont vides.
```

b) Codage (en hexa décimal)

A$

04 54 45 53 54 — longueur 4 "TEST "

B

81 00 00 00 00 — flottant 1 sur 5 octets

C%

00 0A — entier

A ;

81 00 00 00 00 (1) | 00 05 (5 fois 0) | 82 00 00 00 00 (2) | 00 00 (fin) | 1,5fois0,2

B$

04 54 55 54 55 | 00 00 — longueur 4 " TUTU ", fin de tableau

L'enregistrement complet est donc codé :

: 04 54 45 53 54 81 00 00
: 00 00 00 0A 81 00 00 00
: 00 00 05 82 00 00 00 00
: 00 00

Notons que cet enregistrement pourrait avoir plusieurs milliers d'octets de longueur si les tableaux étaient bien remplis.

DATA CONTROL BLOCK

TT	n° logique	1	Ø
	Type de l'objet en face Ø (ou M)	1	1
	Longueur de l'objet Pf-PF	2	2
	n° DRIVE	1	4
	Entête nom sur DRIVE = " F "	1	5
	NOM LØ = 20 -Nombre de pistes libres pour DCBØ	21	6
	Nombre de moyens d'accès	1	27
	Longueur sur le DRIVE PF-Pf	2	28
	Longueur minimum pour le save sur DRIVE PF-Pf (Ø ; 6Ø)	2	30
	Etat du fichier Ø normal ; octet de contrôle	1	32
	dans Magma DEBØ de l'objet	3	33
	Rel/TT de la T.S. Magma DCB PF-Pf	2	36
	Rel/TT de la table des pistes alloués	2	38
	LISTE DES 1ER LIBRES (PF-Pf) rel/T.T.	20	40
	MAT POINTEUR 1ER LIBRE MAM	6	66
MM	Rel prochain moyen d'accès PF-Pf	2	2
	Rel clé recherchée/MM PF-Pf	2	2
	Rel de MaX/MM PF-Pf	2	4
	Rel Xtract/MM PF-Pf	2	6
	Rel T.S. clé/MM PF-Pf	2	8
	Type de moyen d'accès R ou I	1	10
	Rel 1ère piste logique/PP PF-Pf	2	11
	Rel Dère piste logique/PP PF-Pf	2	13
	FIN DE TABLE TRIEE PF-Pf	2	15
	Inused	2	17
	Coefficient de blocage	1	19
	Longueur de l'enregistrement en octet	1	20
	n° enregistrement en MC PF-Pf	2	21
	T.S. pour les clés descriptif 7 octets/variables		23
	Clé recherchée LØ = 21		
	Clé maximum LØ = 21		
	Xtract : Rel de la zone 1	1	
	Longueur de la zone 1	1	
	AUTRES MOYENS D'ACCES		
	T.S. Magma descriptif 7 octets/variables		
PP	LISTE DES PISTES UTILISEES en deux octets		
	Coefficient de blocage enregistrement		
	LISTE DES PISTES / TETES	5	

9 - 1 - c STRUCTURE DES MASQUES

Les masques du M/DOS 6502 comprennent deux parties :

1) le texte du masque

Il s'agit du texte qui apparait par l'ordre LET"C,[n° logique]
Ce texte ne contient donc pas les fenêtres de saisie.

Le texte est compacté pour gagner de la place.

La méthode est la suivante :

Les caractères < et > étant interdits dans les masques car étant réservés aux positions des zones de saisie, ils seront utilisés comme caractères de contrôle.

Si un caractère de code écran X est répété n fois, il sera codé :

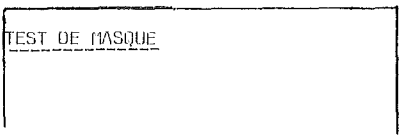
X, CODE("< "), n si n < 256
X, CODE("> "), poids faible de n, poids fort de n dans le cas contraire

Si n est inférieur à 4, il n'y aura pas de codage, car celui-ci n'apporterait pas de gain de place.

La fin de l'écran sera repérée par un nombre nul de caractères identiques (configuration normalement impossible)

NOTE : CODE(" < ") = 60
CODE(" > ") = 62

Exemple : Codage du masque suivant :



Le texte du masque de cet exemple sera codé :

212	T	148
197	E	133
211	S	147
212	T	148
160	blanc	160
196	D	132
197	E	133
160	blanc	160
205	M	141
193	A	129
211	S	147
209	Q	145
213	U	149
197	E	133

```
160      blanc  }
168      <      }    26 blancs consécutifs
26       26     }
173      -      }
168       <     }    14 tirets consécutifs
14       14     }
160      blancs }
190       >     }    blanc jusqu'à la fin de l'écran
Ø        Ø      }
Ø        Ø      }
```

2) les zones de saisie

Pour chaque variable du masque à saisir, on enregistre 7 octets :

- adresse écran du début de la zone — 2 octets

- longueur de la zone — 1 octet

- nom de la variable (codé comme dans le BASIC) — 2 octets

- contrôles à effectuer — 1 octet

- indice dans le cas d'un tableau (99 dans le cas des listes) (Dans le cas des variables simples, cette dernière valeur est inutilisée) — 1 octet

Le contrôle sur 1 octet se décompose de la façon suivante :

- 2 bits pour indiquer le type
  - ØØ flottant
  - Ø1 binaire
  - 1Ø entier
  - 11 alphanumérique

- 1 bits pour contrôle (*)
  - Ø si date et alphanumérique ou numérique gestion
  - 1 dans les autres cas

- 1 bit pour entrée/sortie (?)
  - 1 : en sortie

- 1 bit pour positif (+)
  - 1 : numérique positif ou alpha numérique considéré comme numérique
  - Ø : dans le cas contraire

- 1 bit pour autoskip (!)
  - 1 : autoskip

- 1 bit pour entier (@)
  - 1 : entier ou binaire ou flottant entier (C)

- 1 bit pour tableaux
  1 : tableau
  Ø : variable simple

L'ensemble du masque est structuré de la façon suivante :

contenu	position
- numéro logique	Ø
- type : " M "	1
- longueur totale poids faible poids fort	2 3
- longueur de la table poids faible poids fort (7 fois le nombre de variables NV)	4 5
- table (7 * NV octets) — — — —	6
- texte du masque — — — — —	6 + 7* NV

CHAPITRE X

UTILITAIRES STANDARDS

AZERTY (M) :

Ce masque vous montre les touches à inverser, si vous voulez travailler en AZERTY (commande LET " Z ").

BINARY (M) :

Ce masque indique pour les modules binaires leur longueur. Cela vous permettra de les recopier sans problèmes.

Exemple : HIMEM : 16 * 256
LOAD " $1000,$1300,1 : BOOT1 "
SAVE " $1000,$1300,$: BOOT2 "

AUTO COPIE :

Recopie d'un drive sur un autre les masques, globaux et programmes. Le drive d'arrivée n'est pas détruit. S'il s'agit d'une disquette vierge, il faut donc la formatter par l'ordre LET " #F, n° drive ", avant de lancer le programme AUTO COPIE
IMPORTANT : Les drives peuvent être de types différents (5', 8', disque dur).

AUTO LIST :

Edition de tous les masques et programmes contenus sur une disquette.

AUTOSTART :

Simule les fonctions clavier de la ROM autostart dans le cas où votre micro-ordinateur n'en est pas équipé.

CTRL/S : bloque un list
ESC/I : déplacement du spot vers le haut
ESC/J : déplacement du spot vers la gauche
ESC/K : déplacement du spot vers la droite
ESC/L : déplacement du spot vers le bas

BOOT :

Formattage d'un disque en laissant la piste Ø pour le BOOT. Si celui-ci est sur la disquette, il sera recopié en piste Ø. Les disquettes ainsi obtenues peuvent mettre en route le M/DOS 6502.

CHECK ROM :

Affiche les valeurs trouvées et les valeurs réelles des 8 ROMS du DOS, pour contrôler leur validité.

COPIE :

Copie intégralement un drive sur un autre. Les drives doivent être de même type.
IMPORTANT :

- Ce programme fonctionne quelque soit le périphérique.
- Le drive d'arrivée est effacé.

Si le disque d'arrivée a déjà été formatté, indiquer " N " à la question concernant le formattage.
Le cas se produit en particulier si le disque :

- a été déjà formatté (par LET " # F,n ")
- est déjà une copie dont on ne se sert plus.

COPIE SI :

Copie un fichier séquentiel indexé d'un disque sur un autre, avec possibilité de changer le nom du fichier.

DEMO :

Exemple de programme gérant un fichier en accès par clé. Vous constaterez que les deux masques utilisés ont été regroupés sous forme de GLOBAL, ce qui présente deux avantages :

- temps de chargement accéléré
- place restreinte sur disque

FILE COPY :

Copie un fichier d'un disque sur un autre, après affichage du catalogue.

HELLO :

Ce programme est celui que est exécuté à la mise en route. Vous pouvez le remplacer par un autre de votre choix. Exemple : pour que l'un de vos logiciels démarre dès la mise en route, il vous suffit de rajouter dans le programme HELLO un RUN " nom du premier programme à exécuter.

INTERRO :

Ce programme permet de mettre à jour un fichier quelconque. Pour cela, deux masques sont créés provisoirement (il ne sont pas enregistrés sur disque).
Le premier demande l'opération désirée :

- la clé de recherche définie dans le fichier
- la commande : L = lire
  M = modifier
  C = créer
  D = détruire
  S = lire le suivant

Le deuxième sert à décrire l'article aussi bien en affichage qu'en saisie.

Dans le cas de tableaux, ceux-ci apparaissent en une ligne. Les différents indices sont séparés par " ; ".

MAJ PAGE 3 :

Mise à jour des paramètres de la page 3 (voir chapitre VII)

RENUMEROTE :

Renumérotation des lignes d'un programme en fonction de certaines valeurs.

Exemple : &F300, I 5 , S 200, E 700

Cet exemple renumérote de 5 en 5 la partie du programme comprise entre les lignes 200 et 700, en commençant par la ligne 300. Les valeurs par défaut sont :

F1Ø, I1Ø, S1Ø, E63999

RWTS 3.2 / 3.3 :

Deux versions des lecteurs 5 pouces existent :

- version 3.2 ..... 110K
- version 3.3 ..... 140K

Ce programme permet de faire résider en mémoire les deux versions 110K et 140K et d'effectuer des conversions entre ces deux versions .

Exemple : Vous avez démarré sur une version 140K (3.3). Vous désirez lire et recopier un programme TEST écrit en M/DOS 6502 version 110K. Faites RUN "RWTS 3.2". Le programme affiche l'état des lecteurs actuels : si vous avez deux lecteurs en slot 6 avec interface 140K (drive Ø et 1) et deux en slot 5 avec interface 110K (drives 2 et 3), le programme indique les valeurs par défaut :

```
Ø ..... 140K
1 ..... 140K
2 ..... 140K
3 ..... 140K
4 ..... non connecté
5 ..... non connecté
```

Le programme demande le drive à passer en 110K.
Répondre 2.
Il suffit ensuite, après avoir repris la main, de faire :

```
LOAD " 2 : TEST "
SAVE " Ø : TEST "
```

SCROLL :

Affiche les adresses du SCROLL UP et du SCROLL DOWN.
L'adresse du SCOLL DOWN est calculé par :

```
PEEK(116) * 256 + PEEK(115)
```

L'adresse du SCROLL UP est fixe :

```
CALL 64624
```

SUPER CONTROLE :

Les caractères de contrôles apparaissent sur fond blanc dans le listing du programme, mais l'exécution s'effectue normalement.

UTIL :

Tous les ordres concernant les masques de saisie peuvent être exécutés directement sans affecter la mémoire, en particulier le programme. Cependant, cet utilitaire permet une mise à jour rapide et simple de votre catalogue de masques.
Les commandes possibles sont :

C | Création d'un masque :

L'écran s'éfface, saisissez votre masque puis faites Escape. (CTRL/A pour abandonner). Une cloche signale une erreur dans les fenêtres. Corrigez-la ou abandonnez.

M | Modification ;

Le contenu du masque apparait sur l'écran. Vous pouvez le modifier de la même façon qu'en création.

D | Destruction :

Le masque est affiché pour une dernière validation. Répondez " O " pour détruire.

* | Contenu du disque :

Permet de consulter la liste des masques avant de les manipuler.

L | Ouverture d'un masque :

Son nom s'affiche en bas à droite de l'écran. Il pourra servir de base pour la création d'un deuxième masque par la commande S

S | Création d'un masque à partir d'un autre :

Le module est le masque chargé par " L ".

V | Visualisation :

Simple affichage du masque.

F | Sortie du programme

I | Impression d'un masque :

Imprime un masque sur une imprimante placée en slot 2.

CHAPITRE XI

UTILISATION D'UNE PARTIE DES FONCTIONS DU M/DOS 6502 EN DOS. 3.2 DOS.3.3

Un certain nombre d'ordres du M/DOS 6502 pourront être utilisés en DOS 3.2. DOS 3.3 aux conditions suivantes :

- la page 3 est conforme aux spécifications (voir chapitre paramétrage du M/DOS 6502)
- une place a été réservée pour les buffers M/DOS 6502 et leurs adresses mises à jour en page 3.

Les ordres suivants sont alors utilisables :

LET"#I,n ,M, d : nom" ..... création d'un masque en mémoire
LET"C, n" ..... chargement du texte
LET"P, n" ..... affichage texte + variables
LET"O, n" ..... affichage variables
LET"V, n" ..... visualisation
LET"I, n" ..... saisie
LET"T, n" ..... saisie globale
LET"#C,n" ..... clear
LET"=xxxx±yyyy" ..... addition/soustraction sur 48 chiffres
LET" > xxxx" ..... remplissage buffers
LET" + xxxx" ..... ajout des buffers
LET" ! " ..... execute
LET"Z" ..... azerty
LET"- P" ..... arrondis
LET"y" ..... inversion vidéo écran
LET"G" ..... cloche
LET"?-:" ..... " hard copy "
LET"?-c" ..... masques d'édition
LET"V" ..... visualisation buffers
LET"#C,$" ..... clear de tous les masques

Deux fonctions vont manquer puisqu'elles son liées à la gestion de fichier, il s'agit :

- du sauvetage des masques sauvés par LET"# I"
- de leur relecture

Pour cela, il faudra faire appel aux ordres :

```
BLOAD
BSAVE
```

Les limites du module à sauver sont :

```
début = PEEK(768)*256 + PEEK(769) + 292
fin + 1 = PEEK(773)*256 + PEEK(774)
```

Le sauvetage de l'ensemble des masques en mémoire se fera par :

```
X = PEEK(768)*256 + PEEK(769) + 292
L = PEEK(773)*256 + PEEK(774) - X
?CHR$(4) " BSAVE MASQUES, A " X ", L " L
```

La relecture se fera par :

```
X = PEEK(773)* 256 + PEEK(774)
?CHR$(4) " BLOAD MASQUES , A " X
Y = X + PEEK (43616) + PEEK(43617) * 256   sur 48K
POKE 773 , INT(Y/256)
POKE 774 , Y - 256 * INT(y/256)
```

RÉSUMÉ DES ORDRES DU DOS

MASQUES

```
LET"#NEW-(n), MASQUE,(d) : (NOM) ..... Création d'un masque (donne la main)
LET"#NEW-(n), MASQUE,/(d) : (NOM) ..... Création de masques (prend l'écran)
LET"#NEW-(n), MASQUE,  (d) : (NOM) ..... Remplacement d'un masque
LET"#DELET-MASQUE,(d) : (NOM) ..... Destruction d'un masque
LET"#OPEN-(n), MASQUE,(d) : (NOM) ..... Ouverture d'un masque
LET"CHARGE-(n) ..... Affichage du texte du masque
LET"VISUALISE-(n) ..... Affichage du texte et des fenêtres
LET"OUTPUT-(n),(t) ..... Affichage des variables du masque
LET"PRINT-(n) ..... Affichage du texte et des variables
LET"INPUT-(n),(z) ..... Saisie des variables du masque
LET"T,n" ..... Reprise de variables d'un masque
```

FICHIERS

```
LET" > vci...vcn = vei...vep ..... Description d'un dictionnaire
LET"+  ..... Ajout au dictionnaire
LET"#NEW-(n),FICHIER,(d) = (NOM) ..... Création d'un fichier
LET"#OPEN-(n),FICHIER,(d) : (NOM) ..... Ouverture d'un fichier
LET"#DELET-FICHIER,(d) : (NOM)..... Destruction d'un fichier
LET"#REORG-(n) ..... Réorganisation d'un fichier
LET"READ-(n) ..... Lecture d'un article
LET"NEXT-(n) ..... Lecture de l'article suivant
LET"UPDATE-(n) ..... Mise à jour d'un enregistrement
LET"DELET-(n) ..... Destruction d'un enregistrement
LET"BORNE-(n) ..... Fixe une borne maximum au fichier
LET"XTRACT-(n),(v) ..... Extraction par criteres sur la clé
LET"XINDEX-(n) ..... Envoie en relatif le prochain numéro
LET"WRITE-(n) ..... Création d'un enregistrement
LET"ADD-(n) ..... Ajout (homonymes autorisés)
LET"ENTER-(n) ..... Traduit en clair le dictionnaire
LET"VISUALISE ..... Imprime ce dictionnaire
LET"# > n" ..... Fixe un maximum au coefficient de blocage
```

PROGRAMMES

```
SAVE"(d) : (NOM) ..... Sauve le programme
SAVE"@(d) : (NOM) ..... Sauve en écrasant
LOAD "(d) : (NOM) ..... Charge un programme
LOAD"#(d) : (NOM) ..... Charge le programme à la suite
LOAD"/(d) : (NOM) ..... Exécute le programme sans CLEAR
RUN"(d) : (NOM) ..... Charge et exécute un programme
LET"#DELET,PROGRAMME,(d) : (NOM) ..... Destruction d'un programme
```

DIRECTORY

```
LET" %,(d) ..... Directory complète du disque d
LET" %,(d),(M/F/P/) ..... Liste des masques/fichiers/programmes
LET"#REORG-$(d) ..... Réorganise le catalogue du disque d
```

ORDRES GENERAUX

```
LET"#CLEAR,(n) ..... Récupère la place RAM d'un objet
LET"#CLEAR,"+CHR$(d) ..... Fermeture d'un disque
LET"#CLEAR,$ ..... Ferme tous les fichiers, masques, ...
LET"Y ..... Inversion vidéo de tout l'écran
LET"! ..... Exécute d'une chaine entrée par
LET"%(d) ..... Indique le nombre de pistes libres
LET" = oper 1 + - oper 2 ..... Addition et soustraction de chaines
LET"G ..... Emission d'un signal sonore
LET"Z ..... Passage en AZERTY
LET"?-(x) ..... Impression de la ligne repérée par x
LET"?-% ..... Hard Copy
LET"#C,$d " ..... Fermeture du drive d pour chargement disque
LET"#C,$$" ..... Ferme tous les drives
```

FICHIERS MULTICLES

```
LET" > vai,...,vai & vbi,...vbj & .....
LET"+ = vei,...., ven ..... Définition d'un fichier multiclé
LET"#NEW-(n),FICHIER,(d) : (NOM) ..... Création d'un fichier (après ....)
LET"#DELET-FICHIER,(d) : (NOM) ..... Destruction d'un fichier
LET"OPEN-(n),FICHIER,(d) : (NOM) ..... Ouverture d'un fichier
LET"#REORG-(n) ..... Réorganisation d'un fichier
LET"READ-(n),(my) ..... Lecture d'un enregistrement
LET"NEXT-(n),(my) ..... Lecture séquentielle
LET"UPDATE-(n),(my) ..... Mise à jour d'un enregistrement
LET"DELET-(n),(my) ..... Destruction d'un enregistrement
LET"BORNE-(n),(my) ..... Fixe une borne maximum au fichier
LET"XTRACT-(n),(v),(my) ..... Extraction sur la variable v
LET"WRITE-(n) ..... Création d'un enregistrement
LET"ADD-(n) ..... Ajout (homonymes autorisés)
LET"ENTER-(n) ..... Traduit en clair le dictionnaire
LET"VISUALISE ..... Imprime ce dictionnaire
```

BINAIRES

```
LOAD"$XXXX,$YYYY,d: NOM ..... LOAD de module binaire
SAVE"$XXXX,$YYYY,d: NOM ..... SAVE de module binaire
```

GLOBAUX

```
LET"#N,G,G, d : NOM ..................... Création d'un global
LET"#O,G,G, d : NOM ..................... Ouverture d'un global
```

ACCES DIRECT

```
LET"$ d,c,n,p,t,s,$XXXX ................. Accès direct
```