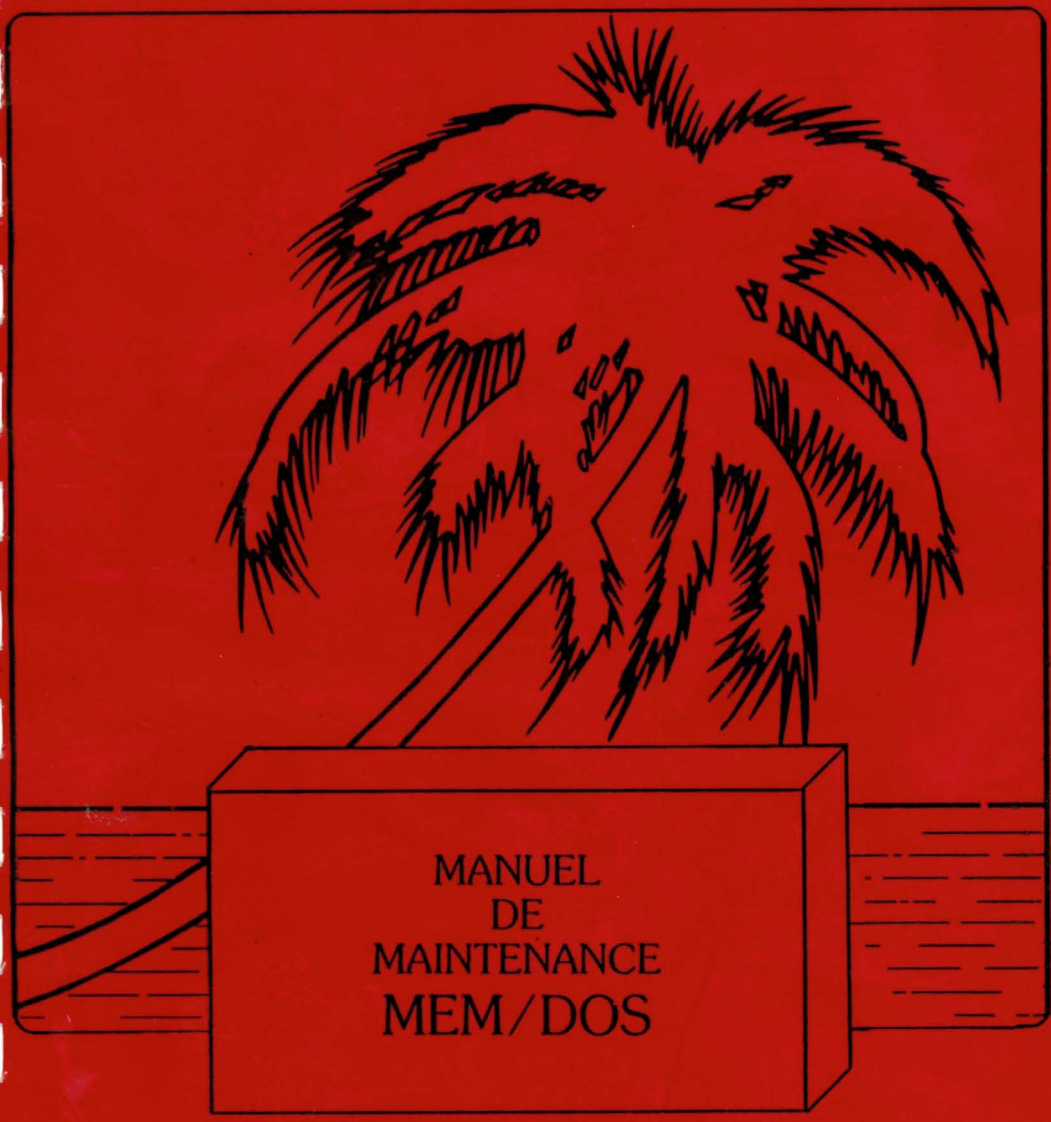


MANUEL DE MAINTENANCE MEM/DOS

mem/dos 6502

MEM/DOS

SYSTEME D'EXPLOITATION

MANUEL DE MAINTENANCE

MEMSOFT

SA AU CAPITAL DE 1.750.000 F . RC NICE B 321 250 490 . APE 7703

NICE - PARIS - LOS ANGELES

SOMMAIRE

...

...

III. FUNCTION ERROR EN LECTURE

IV. LE NG ERROR

CONSOMMATION DES CARTES MEM/DOS 6502

---

①

 BAS

LES MASQUES!

STRUCTURE DES MASQUES MEM/DOS

1) STRUCTURE GENERALE :

Notation : TT = début masque stocké en mémoire.

TT : début du masque
numéro logique : 1 octet

TT + 1 : type d'objet : " M "

TT + 2 : longueur totale du masque (poids faible)

TT + 3 : longueur totale du masque (poids fort)

TT + 4 : longueur jusqu'à la fin de la table des symboles (poids faible)

TT + 5 : longueur jusqu'à la fin de la table des symboles (poids fort)

NOTES :

La longueur indiquée est la longueur de la table des symboles du masque augmenté de la longueur des éléments le précédant (en fait la différence entre la fin de la table et TT)

La table des symboles comporte 7 octets par variable.
Exemple : le masque est composé de 5 variables,
- la longueur de la table est 5 * 7 = 35
- la longueur indiquée ici est donc : 35 + 6 = 41

CAS PARTICULIER :

Les masques 80 colonnes sont repérés par la particularité suivante :
le poids fort de la longueur de la table est augmenté de 128 (80 en héxa).

C'est le test sur ce poids fort qui provoque ILLEGAL QUANTITY ERROR dans une tentative d'ouverture d'un masque 80 colonnes en mode 40 colonnes.

TT + 6 ... TT + longueur table : table elle-même

Les variables sont rangées dans l'ordre des fenêtres de saisie (ou d'affichage) sur l'écran.

Le descriptif de chaque élément de cette table est décrit en 2.

TT + (longueur en TT + 5/TT + 6) ... fin du masque : texte du masque compacté de la façon suivante :

Les caractères non répétés 4 fois sont codés tels quels.

Les caractères répétés de 4 à 255 fois sont codés :
- le caractère en code écran
- < en code écran
- le nombre d'occurences du caractère

Les caractères répétés plus de 255 fois sont codés :

- le caractère en code écran
- > en code écran
- le nombre d'occurences poids faible
- le nombre d'occurences poids fort

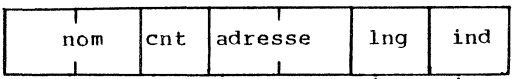
La fin du masque est repérée par :

- le dernier caractère en code écran
- > en code écran
- 0 en indiquant qu'il faut compléter jusqu'à la fin de l'écran

Notant que cette technique est possible car < et > sont des caractères réservés non enregistrés (fenêtres de saisie).

2) STRUCTURE DE CHAQUE VARIABLE DANS LA TABLE :

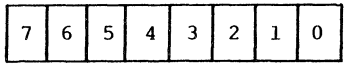
Pour chaque variable du masque, 7 octets sont utilisés :



! ! ! ! !
! ! ! ! pour tableaux seuls
! ! ! de 1 à 255
! ! adresse réelle sur l'écran du début zone
! type et contrôle
nom sur deux caractères (codage similaire à celui du BASIC)

DESCRIPTION PRECISE DE L'OCTET DE CONTROLE :

Chaque bit de cet octet de contrôle a une signification particulière, éventuellement dépendante du type de variable.
Il existe une relation entre cet octet de contrôle et le type de la variable dans BASIC, celle-ci pouvant être flottante, entière ou caractère.




```
!  !  !  !  !  !  !  !
!  !  !  !  !  !  !  0 = tableau,1 = simple
!  !  !  !  !  !  1 = positif (si numérique)
!  !  !  !  !  1 = entier (si numérique)
!  !  !  !  1 = auto skip
!  !  !  en sortie seulement
!  !  0 = date, alpha ou numérique gestion
+--+
 !
 type de variable :

 0 0 = flottant
 0 1 = binaire
 1 0 = entier
 1 1 = alphanumérique
```

NOTE :

binaire = entier positif de 0 à 255
entier = entier signe sur deux octets

EXEMPLE : LE MASQUE UTILM

La table des symboles est la suivante :

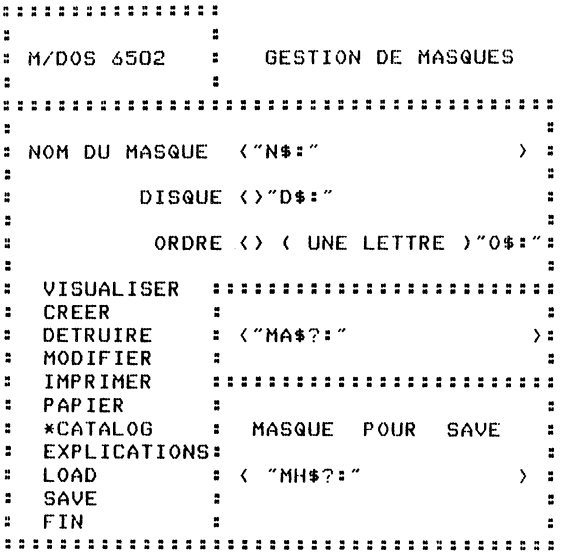
```
4E 80 29 07 11 15 00
44 80 29 04 39 02 00
4F 80 29 05 39 02 00
4D C1 39 07 39 16 00
4D C8 39 06 61 15 00
---- --- ------ -- --
 nom  ct adr   lg in
```

Valeurs " classiques " de l'octet de contrôle :

	non autoskip	autoskip
flottant	E1	E9
entier	A5	AD
binaire simple	67	6F
chaine	21	29
date	01	09

Dans le cas de tableaux, enlever 1.

LE MASQUE UTILM



MASQUE UTILM

```
*9124.92C2
```

n° logique "M"
type "M"
longueur totale

9124-	4D	4D	9E	01					
9128-	29	00	4E	80	29	07	11	15	
9130-	00	44	80	29	04	39	02	00	table
9138-	4F	80	29	05	39	02	00	4D	des
9140-	C1	39	07	39	16	00	4D	C8	symboles
9148-	39	06	61	15	00	BA	BC	0F	
9150-	A0	BC	17	BA	A0	BC	0D	BA	
9158-	A0	BC	17	BA	A0	8D	AF	84	
9160-	8F	93	A0	B6	B5	B0	B2	A0	texte
9168-	BC	02	BA	A0	BC	02	87	85	du
9170-	93	94	89	8F	8E	A0	84	85	masque
9178-	A0	8D	81	93	91	95	85	93	
9180-	A0	BC	02	BA	A0	BC	0D	BA	
9188-	A0	BC	17	BA	BC	28	A0	BC	
9190-	25	BA	BA	A0	0E	0F	0D	20	
9198-	04	15	20	0D	01	13	11	15	
91A0-	05	20	A0	A0	A2	A0	BC	13	
91A8-	BA	BA	A0	BC	25	BA	BA	A0	
91B0-	BC	08	04	09	13	11	15	05	
91B8-	A0	BC	02	A2	A0	BC	12	BA	
91C0-	BA	A0	BC	25	BA	BA	A0	BC	
91C8-	08	20	0F	12	04	12	05	A0	
91D0-	BC	03	A8	A0	95	8E	85	A0	
91D8-	8C	85	94	94	92	85	A0	A9	
91E0-	A2	A0	BC	03	BA	BA	A0	BC	
91E8-	25	BA	BA	A0	A0	16	89	93	
91F0-	95	81	8C	89	93	85	92	A0	
91F8-	A0	BA	BC	19	A0	A0	03	92	
9200-	85	85	92	A0	BC	06	BA	A0	
9208-	BC	16	BA	BA	A0	A0	04	85	
9210-	94	92	95	89	92	85	A0	BC	
9218-	03	BA	A0	A0	A2	A0	BC	13	
9220-	BA	BA	A0	A0	0D	8F	84	89	
9228-	86	89	85	92	A0	BC	03	BA	
9230-	A0	BC	16	BA	BA	A0	A0	09	
9238-	8D	90	92	89	8D	85	92	A0	
9240-	BC	03	BA	BC	19	A0	A0	10	
9248-	81	90	89	85	92	A0	BC	05	
9250-	BA	A0	BC	16	BA	BA	A0	A0	
9258-	2A	83	81	94	81	8C	8F	87	
9260-	A0	BC	03	BA	A0	20	0D	01	
9268-	13	11	15	05	20	20	10	0F	
9270-	15	12	20	20	13	01	16	05	
9278-	20	20	A0	BA	BA	A0	A0	05	
9280-	98	90	8C	89	83	81	94	89	
9288-	8F	8E	93	BA	A0	BC	16	BA	
9290-	BA	A0	A0	0C	8F	81	84	A0	
9298-	BC	07	BA	A0	BC	02	A2	A0	
92A0-	BC	12	BA	BA	A0	A0	13	81	
92A8-	96	85	A0	BC	07	BA	A0	BC	
92B0-	16	BA	BA	A0	A0	06	89	8E	
92B8-	A0	BC	08	BA	A0	BC	16	BA	
92C0-	BE	00	8E						

(2)

L'ABC DE LA DCB...

DESCRIPTION DE LA DCB

TT=début DCB

00 000 no logique (1 octet)
01 001 type = "F" (1 octet)
02 002 longueur totale de la DCB (2 octets LOW-HIGHT)
04 004 no du drive (1 octet)
05 005 nom complet (précédé de "F" 21 octets)
1A 026 unused (1 octet)
1B 027 nombre de clés (1 octet)
1C 028 longueur maximum à sauver (2 octets HIGHT-LOW)
1E 030 longueur minimum à sauver (2 octets HIGHT-LOW)
20 032 inutilisé : toujours 0
21 033 adresse disque de la DCB (3 octets)
24 036 position par rapport à TT de la table des symboles enregistrements (HL)
26 038 position par rapport à TT de la table des pistes allouées (PP)
28 040 adresse disque du premier bloc détruit (3 octets)
2B 043 adresse disque du premier bloc libre (jamais utilisé) (3 octets)
2E 046 liste des premiers libres (20 octets : 2 par clé) (si DCB$ (catalog) = 2 octets)

Le début des informations de la première clé (noté MM) est en TT+66 (en HEXADECIMAL : TT + 42)

La longueur de ce bloc d'informations est notée en TT/TT+1 (HIGHT-LOW)

Les informations concernant la clé suivante (si elle existe) sont donc en MM + cette longueur, et ainsi de suite ...

MM = début information pour une clé

00 000 longueur du bloc d'informations (HIGHT-LOW)
02 002 position par rapport à MM de la clé en cours (HIGHT-LOW)
04 004 position par rapport à MM de la clé maximum pour borne (HIGHT-LOW)
06 006 position par rapport à MM des limites de l'XTRACT (H-L)
08 008 position par rapport à MM de la table des symboles de la clé (H-L)
0A 010 type de clé : "I" = indexé, "R" = relatif
0B 011 première piste logique affectée à cette clé dans la liste des pistes
0D 013 dernière piste logique affectée à cette clé dans la liste des pistes
0F 015 numéro de la première clé non classée (2 octets)
11 017 inutilisés (2 octets)
13 019 coefficient de blocage de la clé (1 octet)
14 020 longueur de la clé (avec 5 octets de pointeurs) (1 octet)
15 021 numéro de la clé en cours (2 octets)
17 023 table des symboles de la clé (7 octets * nombre de variables)
.. ... clé recherchée (sans les pointeurs)
.. ... clé maximum pour le borne (sans les pointeurs)
.. ... pour XTRACT : position du premier octet à tester (1 octet)
.. ... pour XTRACT : longueur de la zone de comparaison

Après les informations sur les clés, on trouve les informations sur les pistes utilisées par le fichier.

PP= début des informations pistes (son adresse relative est en TT+38)

00 000 longueur de blocage enregistrement (1 octet=1F, 3F, 7F ou FF en HEXA)
01 001 nombre de pistes pour l'ensemble des clés (2 octets H-L)
03 003 nombre de pistes total-1 (2 octets H-L)
05 005 liste des pistes logiques avec pour chacune : CYLINDRE-TETE

fin de la DCB

NOTES :

- Dans les descriptions ci-dessus, toutes les valeurs sur deux octets sont poids fort / poids faible (HIGHT-LOW) SAUF la longueur totale de la DCB en TT+2

- Le 'numéro' d'une clé est sa position dans la liste des clés : 0,1,2...

- une 'adresse disque' est un ensemble de trois octets décrivant l'emplacement d'un objet sur disque relativement à la liste des pistes de la DCB concernée.

    * 1- numéro de la piste logique LOW
    * 2- secteur
    * 3- les 4 bits de gauche : position du premier octet dans le secteur
      les 4 bits de droite : numéro de piste logique HIGHT

ATTENTION :

Dans le cas de la position de la DCB sur le disque, se référer à la liste des pistes du catalogue (DCB dollar)

CAS PARTICULIER : La DCB dollar du catalog

* il n'y a qu'une seule clé et MM=TT+48

* à la place du nom, on trouve l'indication de la commande LET"%"

PRINCIPE DE FONCTIONNEMENT

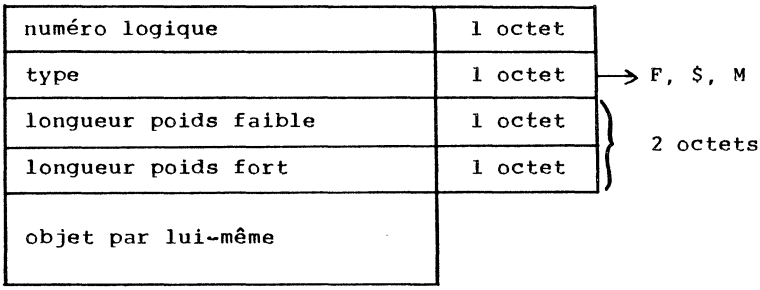
DU MEM/DOS 6502

1 - STRUCTURE DES OBJETS EN MEMOIRE CENTRALE

Les différents types d'objets contenus en mémoire 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'accéder à l'élément suivant.

Et ainsi de suite...

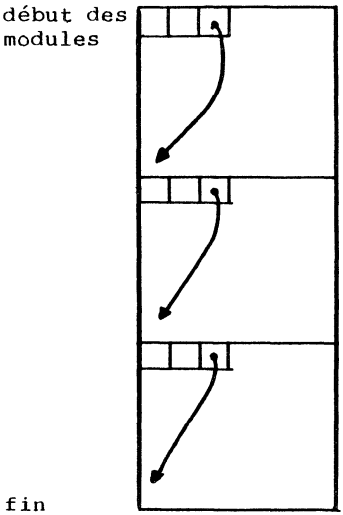
Deux pointeurs permettent de connaître les limites des objets :

- les pointeurs de début des modules :

  CBM 8000 : ($7F0C)+$124
  APPLE : ($300)+$124

- les pointeurs du premier libre :

  CBM 8000 : $7F11
  APPLE : $305



Si le dernier objet ne pointe pas sur le premier libre, l'erreur NG ERROR peut se produire.
(voir détails sur cette erreur : petit guide du dépanneur, chapitre IV)

Destruction d'un objet :

L'ordre LET"▶C,(no logique)
permet de récupérer la place d'un objet.

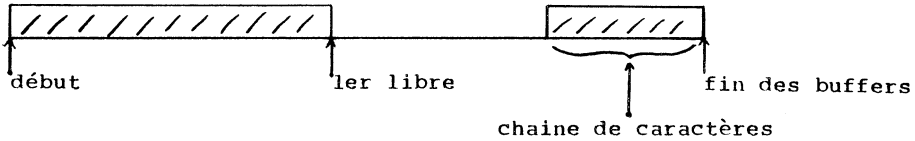
Pour cela, les objets suivants en mémoire sont décalés de la longueur de l'objet détruit et le premier libre est diminué de la même valeur.

Double utilisation des buffers

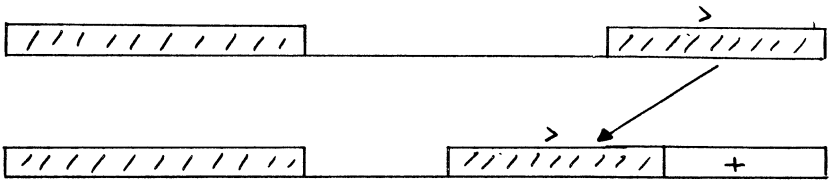
La place disponible dans les buffers entre le pointeur du premier libre et la fin des buffers sert :

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

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



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



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

```
LET">....."
LET"+....."
LET"ENTER~(no logique)
```

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

Dans le cas où 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
```

2 - STRUCTURE DES FICHIERS / DCB

1) Description générale

Un fichier MEM/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 MEM/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 connaître les blocs libérés à la suite de destruction ou non encore utilisés.

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 chaînage sur l'enregistrement. Ce chaînage 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 lire 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 MEM/DOS 6502 de gérer des fichiers de taille variable.

Pour les fichiers séquentiels indexés, les pointeurs vers les enregistrements sont complétés 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 élémine 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 (clé de 5 octets, par exemple un flottant)

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 d'une lecture disque.

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

PROBLEME 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 MEM/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 chaînée à la clé immédiatement inférieure. Pour cela, on utilise le pointeur permettant de retrouver le suivant.

Cela modifie donc l'algorithme de recherche d'un article : après avoir cherché la clé dans la liste triée, il faut vérifier qu'aucun chaînage 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 sûr !)

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 chaînages 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 chaînage 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 chaînés ensemble, le chaînage interne du premier pointant sur le second et ainsi de suite. Le dernier bloc de base ne contient pas de chaînage 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 (contenue dans 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 chaîne 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 chaînage 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 chaîne de caractères	1 + la longueur de chaîne

(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 des 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 :
Le fichier 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 héxadécimal) :

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    00 05      82 00 00 00 00    00 00
--------1-------  ------     --------2--------  ------
                  5 fois 0                       fin
```

B$;

```
04 54 55 54 55    00 00
---------------   ------
longueur 4        fin de
TUTU              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 04 54 55 54 55 00
00
```

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

3

LES OUTILS

!LISTFN

```
 CALCUL MM"(MM=TT,NC/I)
"TYPE MOYEN"(TM$=MM)
"BLOCAGE CLE"(BC=MM)
"CALCUL PP"(PP=TT)
"BLOCAGE RECORD"(BR=TT/PP)
"TOTAL PISTES"(PT=TT/PP)
"PISTES CLE"(PC=TT/PP)
"ADRESSE PISTES"(AD=TT)
"MINMAX PISTES CLE"(MN,MX=NC,TT/MM)
"MINMAX PISTES"(MN,MX=TT/PP,PT)
"EDIT PISTES"(TT/MN,MX,I)
"SEARCH TT"(TT,SS=NL/J)
"EDIT PISTES FILE"(TT/SS/N$/D/NL/SS)
"EDIT PISTES CATALOG"(TT/SS/D)
"ANAL PISTES"
"MAX PARAM"(CM,TM,SM=D)
"HEXA"(A$=A/B,C)
"READ DCB$"(A=D/A$)
"SEARCH KEY/NO"(P,T,S,A,L=TT,NO,NC/N,BC,C,X,MM)
"LONGUEUR CLE"(L=MM)
"EDIT CATALOG KEY/NO"
"READ BLOCK"(AD=D,P,T,S/A$)
"WRITE BLOCK"(A,D,P,T,S/A$)
"TEST FUNCTION ERROR"
"STATUS"(SS)
"MENU"
"CONVERT TRACK"(C,T,S,O=CY,SE,OT,MR,MX)
"RECORD SPACE"(CY,SE,OT,MR,MX,AD,AD$,D,NE)
"VERIFY INIT"(TT,D,MR,MX)
"VERIFY RECORD"(N1=TT,D,NO,AW,AQ)
"VERIFY CATALOG"
"VERIFY CATALOG AUTO"
"FREE SPACE"(D,TT,MR,MX,AD,AD$)
```

```
!LISTEN"CALCUL MM"

1000  DEF  FN "CALCUL MM"(MM = TT,NC / I)
1005  REM  ------------------------------------
1010  REM
1020  REM
1030  REM  NOTATIONS:
1040  REM
1050  REM  TT=ADRESSE DE DEBUT DE LA DCB
1060  REM  MM=ADRESSE DE DEBUT DE DESCRIPTIF POUR UNE CLE
1070  REM
1080  REM  NC=NUMERO DE LA CLE DEMANDEE
1090  REM
1100  REM  I =VARIABLE DE TRAVAIL
1110  CALL  FN "MINMAX PISTES"(MN,MX = TT)
1120  REM
1130  REM  LE CALCUL EST DIFFERENT DANS LE CAS DU CATALOG ET DANS LE CAS
1140  REM  FICHIER.
1150  REM  POUR LE CATALOG, MM EST TOUJOUR EGAL A TT+48 ( UNE SEULE CLE )
1160  REM
1170  REM  POUR UN FICHIER:
1180  REM  = MM 1ERE CLE EST EGAL A TT+66
1190  REM  = MM CLE SUIVANTE SE CALCULE PAR L'ADDITION DE MM ET
1200  REM       L'ADRESSE RELATIVE STOCKEE EN MM, MM+1 (HIG/LOW)
1210  REM
1220  REM
1230  REM  UN CONTROLE PEUT ETRE EFFECTUE, LE NOMBRE TOTAL DE CLES
1240  REM  EST NOTE EN TT+27 ET IL EST DONC FACILE DE VERIFIER NC
1250  REM
1260  REM
1270  REM ***** PROGRAMME *****

1275  IF NC (  = 0 OR NC )  PEEK (TT + 27) THEN  PRINT "NUMERO DE CLE FAUX
     ": STOP
1280  IF  PEEK (TT + 1) = 36 THEN MM = TT + 48: GOTO 1320
1290 MM = TT + 66: IF NC = 1 THEN 1320
1300  FOR I = 2 TO NC
1310 MM = MM +  PEEK (MM) * 256 +  PEEK (MM + 1): NEXT
1320  END  FN
```

```
]LISTFN"TYPE MOYEN

2000  DEF  FN "TYPE MOYEN"(TM$ = MM)
2010  REM  ------------------------
2020  REM
2030  REM
2040  REM  DONNE LE TYPE DE MOYEN D'ACCES D'UNE CLE
2050  REM  ( RELATIF OU INDEXE )
2060  REM
2070  REM  LA REPONSE EST DANS TM$ QUI VAUT "I" OU "R"
2080  REM
2090  REM
2100  REM  LE MOYEN D'ACCES SE TROUVE EN MM+10
2110  REM
2120  REM  MM DOIT DONC AVOIR ETE CALCULE AVANT CETTE OPERATION
2130  REM
2140  REM
2150  REM  ***** PROGRAMME *****



2160 TM$ =  CHR$ ( PEEK (MM + 10))
2170  IF TM$ (  ) "I" AND TM$ (  ) "R" THEN  PRINT "TYPE ERRONE": STOP
2180  END  FN
```

```
]LISTFN"BLOCAGE CLE"

3000   DEF   FN "BLOCAGE CLE"(BC = MM)
3020   REM
3030   REM
3040   REM   EN FONCTION DE MM ( DEBUT PARAMETRES D'UNE CLE ), ON CALCULE
3050   REM   LE COEFFICIENT DE BLOCAGE DE CETTE CLE NOTE BC
3060   REM
3070   REM   RAPPELONS QUE LE COEFFICIENT DE BLOCAGE D'UNE CLE EST LE NOMBRE

3080   REM   DE CLES QUE L'ON PEUT METTRE DANS UN SECTEUR DE 256 OCTETS
3090   REM
3100   REM   DANS LE CAS D'UN FICHIER INDEXE, LA LONGUEUR CLE REELLE DOIT
3110   REM   ETRE AUGMENTEE DE 5 A CAUSE DES CHAINAGES,
3120   REM   DANS LE CAS D'UN FICHIER RELATIF, LE COEFFICIENT DE BLOCAGE
3130   REM    EST FIXE : LA LONGUEUR DU POINTEUR EST 3
3140   REM   ON PEUT DONC EN METTRE 256/3 PAR SECTEUR SOIT 85.
3150   REM
3160   REM   LE COEFFICIENT DE BLOCAGE SE TROUVE EN MM+19
3170   REM
3180   REM
3190   REM   ***** PROGRAMME *****




3200 BC =   PEEK (MM + 19)
3210   END   FN
```

```
]LISTFN"CALCUL PP"

4000  DEF  FN "CALCUL PP"(PP = TT)
4020  REM
4030  REM  PP EST L'ADRESSE DES INFORMATIONS CONCERNANT LES PISTES
4040  REM
4050  REM  EN PP : COEFFICIENT DE BOCAGE ENREGISTREMENTS
4060  REM     PP+1 & PP+2 : NOMBRE DE PISTES DE CLES
4070  REM      PP+3 & PP+4 : NOMBRE DE PISTES TOTAL - 1
4080  REM   A PARTIR DE PP+5, LA LISTE DES PISTES CONCERNEES
4090  REM  AVEC 2 OCTETS:
4100  REM   - LE CYLINDRE
4110  REM   - LA TETE
4120  REM
4130  REM
4140  REM  A NOTER: LES NOMBRES DE PISTES SONT POIDS FORT/FAIBLE
4150  REM           (HIG/LOW)
4160  REM
4170  REM
4180  REM  PP EST LA SOMME DE TT ET DE L'ADRESSE RELATIVE SITUEE
4190  REM  EN TT+38 & TT+39  (HIG/LOW)
4200  REM
4220  REM
4230  REM
4240  REM  ***** PROGRAMME *****



4250 PP = TT +  PEEK (TT + 38) * 256 +  PEEK (TT + 39)
4260  END  FN
```

```
]LISTFN"BLOCAGE RECORD

5000  DEF  FN "BLOCAGE RECORD"(BR = TT / PP)
5020  REM
5030  REM
5040  REM   LE COEFFICIENT DE BLOCAGE DES ENREGISTREMENTS
5050  REM  SE TROUVE EN PP
5060  REM
5070  REM  UTILISONS LA FONCTION "CALCUL PP"
5080  REM
5090  REM
5100  REM
5110  REM  ***** PROGRAMME *****




5120  CALL  FN "CALCULPP"(PP = TT)
5130 BR =  PEEK (PP)
5140  END  FN


]LISTFN"TOTAL PISTES

6000  DEF  FN "TOTAL PISTES"(PT = TT / PP)
6020  REM
6030  REM
6040  REM
6050  REM  LE NOMBRE TOTAL DE PISTES SE TROUVE EN PP+3 & PP+4
6060  REM  (HIG/LOW)
6065  REM  ATTENTION!
6066  REM   CE NOMBRE EST A AUGMENTER DE 1
6067  REM
6070  REM
6080  REM
6090  REM  UTILISONS LA FONCTION "CALCUL PP" DEJA EXISTANTE
6100  REM
6110  REM
6120  REM
6130  REM  ***** PROGRAMME *****




6140  CALL  FN "CALCUL PP"(PP = TT)
6150 PT =  PEEK (PP + 3) * 256 +  PEEK (PP + 4) + 1
6160  END  FN
```

```
]LISTFN"PISTES CLE

7000  DEF  FN "PISTES CLE"(PC = TT / PP)
7020  REM
7030  REM
7040  REM
7050  REM  LE NOMBRE DE PISTES DE CLE EST EN PP+1 & PP+2
7060  REM
7070  REM
7080  REM  UTILISONS LA FONCTION "CALCUL PP"
7090  REM
7100  REM
7110  REM
7120  REM  ***** PROGRAMME *****




7130  CALL  FN "CALCUL PP"(PP = TT)
7140 PC =  PEEK (PP + 1) * 256 +  PEEK (PP + 2)
7150  END  FN


]LISTFN"ADRESSE PISTES

8000  DEF  FN "ADRESSE PISTES"(AD = TT)
8010  REM
8020  REM
8030  REM   INDIQUE DANS AD L'ADRESSE DE LA LISTE DES PISTES UTILISEES
8040  REM  ( TOUS TYPES CONFONDUS )
8050  REM
8060  REM  LES PISTES DE CLE SONT EN PREMIER,
8070  REM  PUIS LES PISTES D'ENREGISTREMENT
8080  REM
8090  REM
8100  REM
8110  REM  POUR ISOLER LES PISTES DE CHAQUE CLE,
8120  REM  ON A POUR CHACUNE D'ELLE
8130  REM  EN MM+11 & +12 LA POSITION DE LA PREMIERE PISTE DANS LA LISTE
8140  REM  ET EN MM+13 & +14 LA POSITION DE LA DERNIERE
8150  REM
8160  REM
8170  REM
8180  REM  ***** PROGRAMME *****




8190  CALL  FN "CALCUL PP"(AD = TT)
8195 AD = AD + 5
```

```
]LISTFN"MINMAX PISTES CLE
9000  DEF  FN "MINMAX PISTES CLE"(MN,MX = NC,TT / MM)
9010  REM
9020  REM
9030  REM   FOURNIT POUR UNE CLE DONNEE
9040  REM  MN=ADRESSE 1ERE PISTE DE LA CLE CONCERNEE
9050  REM  MX=ADRESSE DE LA DERNIERE
9060  REM
9070  REM
9080  REM  UTILISE PP QUI EST CALCULE PAR LA FONCTION
9090  REM  "CALCUL PP"
9100  REM
9110  CALL  FN "CALCUL MM"(MM = TT,NC)
9120  REM  LA VALIDITE DU NUMERO DE PISTE EST FAITE
9130  REM  DANS LA FONCTION "CALCUL MM" QUI EST UTILISEE
9140  REM
9150  REM  POUR CHAQUE CLE ON A:
9160  REM  EN MM+11 MM+12 LA POSITION RELATIVE DE LA 1ERE PISTE
9170  REM  EN MM+13 MM+14 LA POSITION RELATIVE DE LA DERNIERE
9180  REM
9190  REM
9200  REM
9210  REM  ***** PROGRAMME *****
```

```
9220  CALL  FN "ADRESSE PISTES"(MX = TT)
9230  CALL  FN "CALCUL MM"(MM = TT,NC)
9240 MN = MX + ( PEEK (MM + 11) * 256 +  PEEK (MM + 12)) * 2
9250 MX = MX + ( PEEK (MM + 13) * 256 +  PEEK (MM + 14)) * 2
9260  END  FN
```

]LISTFN" MINMAX PISTES

```
10000  DEF  FN "MINMAX PISTES"(MN,MX = TT / PP,PT)
10010  REM
10020  REM
10030  REM  MN=ADRESSE 1ERE PISTE
10040  REM  MX=ADRESSE DERNIERE PISTE
10050  REM
10060  REM
10070  REM  ON UTILISE "ADRESSE PISTES"
10080  REM  ET "TOTAL PISTES"
10090  REM
10100  REM
10110  REM
10120  REM  ***** PROGRAMME *****
```

```
10130  CALL  FN "ADRESSE PISTES"(MN = TT)
10140  CALL  FN "TOTAL PISTES"(PT = TT)
10150 MX = MN + (PT - 1) * 2
10160  END  FN
```

]LISTFN"EDIT PISTES

```
11000  DEF  FN "EDIT PISTES"(TT / MN,MX,I)
11010  REM
11020  REM
11030  REM  EDITE LA LISTE DES PISTES
11040  REM
11050  REM  UTILISE "MINMAX PISTES"
11060  REM
11070  REM
11080  REM
11090  REM  ***** PROGRAMME *****
```

```
11110  PRINT "TRACK/HEAD LIST"
11120  PRINT "================"
11130  PRINT
11135  CALL  FN "MINMAX PISTES"(MN,MX = TT)
11140  FOR I = MN TO MX STEP 2
11150  PRINT  RIGHT$ ("000" +  STR$ ( PEEK (I)),3)"    ";
11160  PRINT  RIGHT$ ("000" +  STR$ ( PEEK (I + 1)),3)
11170  NEXT : PRINT
11180  END  FN
```

```
]LISTFN"SEARCH TT
12000  DEF  FN "SEARCH TT"(TT,SS = NL / J)
12010  REM
12020  REM  CHERCHE UN MODULE D'APRES SON NUMERO LOGIQUE NL
12030  REM
12040  REM
12050  REM  EN $300,$301 ON A L'ADRESSE DES BUFFERS
12060  REM  A CETTE ADRESSE ON AJOUTE $124 POUR AVOIR LE PREMIER OBJET
12070  REM
12080  REM
12081  REM  SS=STATUS (0=OK)
12090  REM
12100  REM  ***** PROGRAMME *****
12110 TT =  PEEK (768) * 256 +  PEEK (769)
12120 TT = TT + 256 + 36:SS = 0
12130 J =  PEEK (775) * 256 +  PEEK (776)
12140  IF TT )  = J THEN SS = 1: GOTO 12180
12150  IF NL =  PEEK (TT) THEN 12180
12160 TT = TT +  PEEK (TT + 2) +  PEEK (TT + 3) * 256
12170  GOTO 12140
12180  END  FN
```

```
]LISTFN"EDIT PISTES FILE
13000  DEF  FN "EDIT PISTES FILE"(TT / SS / N$ / D / NL / SS)
13010  REM
13020  REM  EDITION DES PISTES UTILISEES
13030  REM  PAR UN FICHIER OUVRABLE
13040  REM
13060  REM
13070  REM
13080  REM  ***** PROGRAMME *****
13090  INPUT "DRIVE ?";D
13100  INPUT "FILE NAME ?";N$
13110  PRINT : PRINT "....OPENING FILE....": PRINT
13120  LET "£C,$"
13130  LET "£0,F,F," +  STR$ (D) + ":" + N$
13140  CALL  FN "STATUS"(SS): IF SS = 0 THEN 13160
13150  PRINT "£££ FILE DON'T EXIST £££": PRINT : GOTO 13100
13160 NL =  ASC ("F")
13170  CALL  FN "SEARCH TT"(TT,SS = NL)
13180  IF SS THEN 13150
13190  CALL  FN "EDIT PISTES"(TT)
13200  END  FN

]LISTFN"EDIT PISTES CATALOG
14000  DEF  FN "EDIT PISTES CATALOG"(TT / SS / D)
14010  REM
14020  REM  EDITION DES PISTES UTILISEES PAR LE CATALOG
14030  REM
14040  REM
14060  REM
14070  REM
14080  REM  ***** PROGRAMME *****
14090  INPUT "DRIVE ?";D
14100  LET "£C,$": CALL  FN "READ DCB$"(TT = D)
14120  IF SS THEN  PRINT "ERROR": STOP
14130  CALL  FN "EDIT PISTES"(TT)
14140  END  FN
```

```
!LISTEN"ANAL PISTES
15000  DEF  FN "ANAL PISTES"
15010  REM
15020  REM
15030  REM  VERIFICATION DE LA VALIDITE
15040  REM  DES PISTES D'UN DRIVE
15050  REM
15060  REM
15070  REM   ***** PROGRAMME *****


15075  PRINT : INPUT "DRIVE ";D: PRINT
15080  LET "£C,$": LET "%" +  STR$ (D)
15100  PRINT "TRACKS ALOCATION TEST ON DRIVE £"D
15110  PRINT : PRINT
15120  CALL  FN "MAX PARAM"(CM,TM,SM = D
15130  DIM TT(CM,TM),TN$(500)
15140  REM
15150  PRINT "**** CATALOG = FILE £1 ****
15155  LET "£D,A,A":NE = 0
15160  CALL  FN "READ DCB$"(TT = D)
15180 NO = 1:TN$(1) = "$$ CATALOG"
15190  CALL  FN "MINMAX PISTES"(MN,MX = TT)
15200  FOR I = MN TO MX STEP 2
15210  PRINT  RIGHT$ ("000" +  STR$ ( PEEK (I)),3)"   ";
15220  PRINT  RIGHT$ ("000" +  STR$ ( PEEK (I + 1)),3)
15230  IF  PEEK (I) )  = CM THEN  PRINT "==FUNCTION ERROR==": LET "G"
15240  IF  PEEK (I + 1) ) TM THEN  PRINT "==FUNCTION ERROR==": LET "G"
15245  IF TT( PEEK (I), PEEK (I + 1)) (  ) 0 THEN  PRINT "££££ ERROR": PRINT
     "  OCCUPED BY FILE £"TT( PEEK (I), PEEK (I + 1))" : "TN$(NO):NE = NE +
     1
15250 TT( PEEK (I), PEEK (I + 1)) = NO
15255  NEXT
15260  LET "*-?":WW$ = WW$ + "": CALL  FN "STATUS"(SS): IF SS THEN 15400
15280  IF WW$ ( "F" THEN 15260
15290  IF WW$ ) "G" THEN 15400
15300 WW$ =  MID$ (WW$,2)
15305  IF  RIGHT$ (WW$,1) = " " THEN WW$ =  MID$ (WW$,1, LEN (WW$) - 1): GOTO
     15305
15306 NO = NO + 1:TN$(NO) = WW$
15310  PRINT : PRINT "FILE £"NO" : "WW$: PRINT
15320  LET "£C,F"
15330  LET "£O,F,F," + WW$
15340  CALL  FN "STATUS"(SS): IF SS THEN  PRINT "....INSUFFISANT PRIORITY
     FOR TEST....": GOTO 15260
15360 NL =  ASC ("F"): CALL  FN "SEARCH TT"(TT,SS = NL)
15370  IF SS THEN  PRINT "....LOGICAL NUMBER ERROR....": GOTO 15260
15390  GOTO 15190
15400  PRINT : PRINT "END."
15410  PRINT : PRINT NE" ERRORS."
15420  PRINT : PRINT "TRACKS LIST"
15430  PRINT "============": PRINT
15440  FOR I = 0 TO CM: FOR J = 0 TO TM
15450  IF TT(I,J) = 0 THEN 15470
15460  PRINT  RIGHT$ ("000" +  STR$ (I),3)"   " RIGHT$ ("000" +  STR$ (J),
     3)"   FILE £"TT(I,J)" : "TN$(TT(I,J))
15470  NEXT : NEXT
15480  PRINT : PRINT NE" ERRORS."
15490  END  FN
```

```
]LISTFN"MAX PARAM

16000  DEF  FN "MAX PARAM"(CM,TM,SM = D)
16010  REM
16020  REM
16030  REM  DEFINITION DES PARAMETRES MAXIMUM D'UN DRIVE
16040  REM
16050  REM
16060  REM
16070  REM  ***** PROGRAMME *****

16080 CM =  PEEK (784 + D)
16090 TM =  PEEK (790 + D) - 1
16100 SM =  PEEK (796 + D) - 1
16110  END  FN

]LISTFN"HEXA

17000  DEF  FN "HEXA"(A$ = A / B,C)
17010  REM  CONVERTIT A DECIMAL XXXXX
17020  REM  EN A$ HEXA $YYYY
17030  REM
17040  REM
17050  REM
17060  REM   ***** PROGRAMME *****

17070 B =  INT (A / 256):A$ = "$": GOSUB 17090
17080 B = A - B * 256: GOSUB 17090: GOTO 17130
17090 C =  INT (B / 16): GOSUB 17110
17100 C = B - 16 * C
17110  IF C ( 10 THEN A$ = A$ +  CHR$ (C + 48): RETURN
17120 A$ = A$ +  CHR$ (C + 55): RETURN
17130  END  FN
```

```
]LIST FN"READ DCB$
18000  DEF  FN "READ DCB$"(A = D / A$)
18010  REM
18020  REM  LIT LA DCB$ DU DRIVE D
18030  REM   ET LA PLACE AU DEBUT DES BUFFERS
18040  REM   ( ADRESSE INDIQUEE EN $300 $301 [HIG,LOW] )
18050  REM
18060  REM
18070  REM  REND DANS A L'ADRESSE DE LA DCB$
18080  REM
18090  REM
18100  REM
18110  REM  ***** PROGRAMME *****
```

```
18120 A =  PEEK (768) * 256 +  PEEK (769)
18130  CALL  FN "HEXA"(A$ = A)
18140  LET "$" +  STR$ (D) + ",1,1," +  STR$ ( PEEK (D + 784)) + ",0,0," +
    A$
18150  END  FN
```

```
]LISTFN"SEARCH KEY/NO
19000  DEF  FN "SEARCH KEY/NO"(P,T,S,A,L = TT,NO,NC / N,BC,C,X,MM)
19010  REM
19020  REM  TT=ADRESSE DE LA DCB
19030  REM   NC=NUMERO CLE ( MOYEN D'ACCES )
19035  REM  NO=NUMERO D'ORDRE DE LA CLE
19040  REM
19050  REM  P=PISTE REELLE
19055  REM  T=TETE REELLE
19060  REM  S=SECTEUR
19065  REM  A=POSITION DANS SECTEUR
19070  REM  L=LONGEUR CLE
19075  REM
19080  REM
19090  REM  ****** PROGRAMME *****
```

```
19100  CALL  FN "MINMAX PISTES CLE"(MN,MX = NC,TT)
19120  CALL  FN "CALCUL MM"(MM = TT,NC)
19130  CALL  FN "BLOCAGE CLE"(BC = MM)
19140 N =  INT (NO / BC): REM  NUMERO DU SECTEUR
19150  CALL  FN "MAX PARAM"(CM,TM,SM = D)
19160 X =  INT (N / (CM + 1)): REM  NUMERO D'ORDRE DE LA PISTE
19170 S = N - X * (CM + 1): REM  SECTEUR
19180  IF MN + X * 2 ) MX THEN  PRINT "TRACK OUT OF LIMIT ERROR": STOP
19190 P =  PEEK (MN + X * 2):T =  PEEK (MN + X * 2 + 1)
19210  CALL  FN "LONGUEUR CLE"(L = MM)
19220 A = L * (NO - N * BC)
19230  END  FN
```

```
]LISTFN"LONGUEUR CLE
```

```
20000  DEF  FN "LONGUEUR CLE"(L = MM)
20010  REM
20020  REM
20030  REM  DONNE DANS L LA LONGUEUR COMPLETE DE LA CLE
20040  REM  ( AVEC LES CHAINAGES !)
20050  REM
20060  REM
20070  REM  ***** PROGRAMME *****
```

```
20080 L =  PEEK (MM + 20)
20090  END  FN
```

```
]LISTFN"EDIT CATALOG KEY/NO

21000  DEF  FN "EDIT CATALOG KEY/NO"
21010  REM
21020  REM  PERMET DE RETROUVER UNE CLE
21030  REM  ET LES POINTEURS ASSOCIES
21040  REM  POUR POUVOIR FLAGER L'OBJET
21050  REM
21060  REM
21070  REM  ***** PROGRAMME *****




21080  PRINT : INPUT "DRIVE ? ";D
21090  PRINT : INPUT "KEY NO ? ";NO
21100  LET "£C,$"
21110  CALL  FN "READ DCB$"(TT = D)
21120 NC = 1
21130  CALL  FN "SEARCH KEY/NO"(P,T,S,A,L = TT,NO,NC)
21140  CALL  FN "READ BLOCK"(AD = D,P,T,S)
21145  INVERSE
21150  PRINT " KEY £"NO" : ";: FOR I = 0 TO 20
21160  PRINT  CHR$ ( PEEK (AD + A + 5 + I));: NEXT
21162  IF  PEEK (AD + A + 1) = 255 THEN  PRINT : PRINT "(((((( DELETED ))))
    )"
21164  PRINT : PRINT : PRINT "NUMERO CLE SUIVANTE : " PEEK (AD + A + 3) *
     256 +  PEEK (AD + A + 4)
21165  PRINT : NORMAL
21170  PRINT : PRINT "RETURN:NEXT KEY"
21171  PRINT "S:STOP"
21172  INPUT "R:RESTART    .....";0$
21180  PRINT : PRINT : IF 0$ = "R" THEN 21090
21185  IF 0$ = "" THEN 21280
21190  PRINT
21200  PRINT "TRACK = "P
21210  PRINT "HEAD  = "T
21220  PRINT "SECTOR= "S
21230  PRINT "RELATIVE ADRESS=";A
21240  PRINT "ACTUAL ADRESS OF SECTOR="AD
21250  GOTO 21300
21280 NO = NO + 1:A = A + 26: IF A ( 230 THEN 21145
21290  GOTO 21110
21300  PRINT : PRINT "S : STOP
21310  PRINT "F : FLAG AS DELETED"
21320  INPUT "..............";0$
21330  IF 0$ = "S" THEN 21370
21340  POKE AD + A + 1,255
21350  CALL  FN "WRITE BLOCK"(AD,D,P,T,S)
21360  REM
21370  END  FN
```

```
JLISTFN"READ BLOCK
22000  DEF  FN "READ BLOCK"(AD = D,P,T,S / A$)
22010  REM
22020  REM  LIT UN SECTEUR DONNE
22030  REM  DANS LES PREMIERS OCTETS DES BUFFERS
22040  REM
22050  REM
22060  REM  ***** PROGRAMME *****

22070 AD =  PEEK (768) * 256 +  PEEK (769)
22080  CALL  FN "HEXA"(A$ = AD)
22090  LET "$" +  STR$ (D) + ",1,1," +  STR$ (P) + "," +  STR$ (T) + "," +
     STR$ (S) + "," + A$
22100  CALL  FN "TEST FUNCTION ERROR"
22110  END  FN

JLISTFN"WRITE BLOCK
23000  DEF  FN "WRITE BLOCK"(A,D,P,T,S / A$)
23010  REM
23020  REM  ECRIT UN SECTEUR SUR LE DISQUE
23030  REM
23040  REM  AD=ADRESSE EN MEMOIRE
23050  REM
23060  REM  P,T,S = ADRESSE SUR LE DISQUE
23070  REM
23080  REM
23090  REM  ***** PROGRAMME *****

23100  CALL  FN "HEXA"(A$ = A)
23110  LET "$" +  STR$ (D) + ",2,1," +  STR$ (P) + "," +  STR$ (T) + "," +
     STR$ (S) + "," + A$
23120  CALL  FN "TEST FUNCTION ERROR"
23130  END  FN
```

```
]LISTFN"TEST FUNCTION ERROR

24000  DEF  FN "TEST FUNCTION ERROR"
24010  REM
24020  REM  APRE" UN READ OU WRITE BLOC
24030  REM  TEST  U STATUS POUR SAVOIR SI LES PARAMETRES ETAIENT
24040  REM  HORS LIMITES
24050  REM
24060  REM
24070  REM   ***** PROGRAMME *****



24080  IF  PEEK (189) THEN  PRINT "FUNCTION ERROR IN R/W OPERATION": STOP

24090  END  FN



]LISTFN"STATUS

25000  DEF  FN "STATUS"(SS)
25010  REM
25020  REM  RENVOIE LE STATUS
25030  REM
25040  REM
25050  REM  ***** PROGRAMME *****



25060 SS =  PEEK (189)
25070  END  FN
```

]LISTFN"MENU

```
26000  DEF  FN "MENU"
26010  REM
26020  REM  MENU GENERAL
26030  REM
26040  REM
26050  HOME
26060  PRINT "   MEM/DOS 6502
26070  PRINT "   ------------
26080  PRINT : PRINT : PRINT "       DIAGNOSTICS DISK"
26090  PRINT : PRINT : PRINT
26100  PRINT "1/ ANALYSE TRACKS"
26110  PRINT
26120  PRINT "2/ FLAG OBJECT IN CATALOG"
26130  PRINT
26140  PRINT "3/ EDIT TRACKS ALLOCATION FOR A FILE"
26150  PRINT
26160  PRINT "4/ EDIT TRACKS ALLOCATION FOR CATALOG"
26170  PRINT : PRINT "5/ VERIFY CATALOG RECORD (MANUEL)"
26180  PRINT : PRINT "6/ VERIFY CATALOG RECORD (AUTO)"
26500  PRINT
26505  INPUT "?";A
26506  HOME
26510  ON A GOTO 26600,26610,26620,26630,26640,26650,26660,26670,26680,266
     90
26600  CALL  FN "ANAL PISTES": GOTO 29000
26610  CALL  FN "EDIT CATALOG KEY/NO": GOTO 29000
26620  CALL  FN "EDIT PISTES FILE"(TT)
26630  CALL  FN "EDIT PISTES CATALOG"(TT): GOTO 29000
26640  CALL  FN "VERIFY CATALOG": GOTO 29000
26650  CALL  FN "VERIFY CATALOG AUTO": GOTO 29000
29000  END  FN
```

]LISTFN"CONVERT TRACK

```
40000  DEF  FN "CONVERT TRACK"(C,T,S,O = CY,SE,OT,MR,MX)
40010  REM
40020  REM  CONVERTION ADRESSE DISQUE EN CYL TETE SECT ORDRE PHYSIQUE
40030  REM
40040  REM
40050  REM
40060  REM  PISTE DE MR A MX
40070 S = SE
40080 O =  INT (OT / 16) * 16
40090 C = OT - O:C = C * 256 + CY
40100  IF C ) (MX - MR) / 2 THEN  PRINT " TRACK OUT OF RANGE ": STOP
40110 T =  PEEK (MR + 2 * C + 1)
40120 C =  PEEK (MR + 2 * C)
40130  END  FN
```

```
]LISTFN"RECORD SPACE

41000  DEF  FN "RECORD SPACE"(CY,SE,OT,MR,MX,AD,AD$,D,NE)
41010  REM
41020  REM  LIT LA SUITE DES BLOCS OCCUPES
41030  REM  ET MET A JOUR LE ; BLEAU MAX
41040  REM
41050  REM
41060  IF SE = 255 THEN 41210
41070  CALL  FN "CONVERT TRACK"(C,T,S,0 = CY,SE,OT,MR,MX)
41080  LET "$" +  STR$ (D) + ",1,1," +  STR$ (C) + "," +  STR$ (T) + "," +
      STR$ (S) + "," + AD$
41090  IF  PEEK (189) THEN  PRINT "  FONCTION ERROR": STOP
41100  IF  PEEK (AD + 0 + 4) = 0 THEN NE =  - 1: REM  DETRUIT
41110  PRINT C" "S" "T" " INT (0 / 16)
41120  IF MA%((OT - 16 *  INT (OT / 16)) * 256 + CY,S, INT (0 / 16)) THEN
      PRINT " ERROR : RECORDS "NE" & "MA%(256 * (OT - 16 *  INT (OT / 16))
      + CY,S, INT (0 / 16))" SAME SPACE":ER = ER + 1
41130 MA%((OT - 16 *  INT (OT / 16)) * 256 + CY,S, INT (0 / 16)) = NE
41140  IF  PEEK (AD + 0 + 3) ( 255 GOTO 41210
41150  IF  PEEK (AD + 0) = CY AND  PEEK (AD + 0 + 1) = SE AND  PEEK (AD +
     0 + 2) - 16 *  INT ( PEEK (AD + 0 + 2) / 16) = OT - 16 *  INT (OT / 1
     6) THEN 0 =  INT ( PEEK (AD + 0 + 2) / 16) * 16: GOTO 41110
41160 CY =  PEEK (AD + 0):SE =  PEEK (AD + 0 + 1):OT =  PEEK (AD + 0 + 2)
41170  CALL  FN "CONVERT TRACK"(C,T,S,0 = CY,SE,OT,MR,MX)
41180  LET "$" +  STR$ (D) + ",1,1," +  STR$ (C) + "," +  STR$ (T) + "," +
      STR$ (S) + "," + AD$
41190  IF  PEEK (189) THEN  PRINT " FUNCTION ERROR": STOP
41200  GOTO 41110
41210  END  FN


]LISTFN"VERIFY INIT

42000  DEF  FN "VERIFY INIT"(TT,D,MR,MX)
42010  REM
42020  REM  LIT POUR UN ARTICLE
42030  REM   LA CLE ET LA SUITE DE SES BLOCS
42040  REM
42050  CALL  FN "MAX PARAM"(CM,TM,SM = D)
42060  CALL  FN "MINMAX PISTES"(MN,MX = TT)
42070  CALL  FN "CALCUL PP"(PP = TT)
42080 MR = MN + 2 * ( PEEK (PP + 1) * 256 +  PEEK (PP + 2))
42090 X = (MX - MR) / 2
42095  DIM MA%(X,SM,16)
42096  END  FN
```

JLISTFN"VERIFY RECORD

```
42100  DEF  FN "VERIFY RECORD"(N1 = TT,D,NO,AW,AQ)
42120 NC = 1
42130  PRINT
42140  CALL  FN "SEARCH KEY/NO"(P,T,S,A,L = TT,NO,NC)
42150  CALL  FN "READ BLOCK"(AD = D,P,T,S)
42155 N1 =  PEEK (AD + A + 3) * 256 +  PEEK (AD + A + 4)
42160  CALL  FN "HEXA"(AD$ = AD)
42170 CY =  PEEK (AD + A):SE =  PEEK (AD + A + 1):OT =  PEEK (AD + A + 2)
42175  PRINT : PRINT : FOR I = 0 TO 20: PRINT  CHR$ ( PEEK (AD + A + 5 + I
     ));: NEXT : PRINT : PRINT
42180  CALL  FN "RECORD SPACE"(CY,SE,OT,AW,AQ,AD,AD$,D,NO)
42200  END  FN
```

JLISTFN"VERIFY CATALOG

```
43000  DEF  FN "VERIFY CATALOG"
43010  REM
43020  REM  OPERE LE MAPPING DU MAGMA DCB$
43030  REM
43040 AD =  PEEK (773) * 256 +  PEEK (774)
43050  CALL  FN "HEXA"(AD$ = AD)
43060  INPUT "DRIVE ?";D
43070  CALL  FN "MAX PARAM"(CM,TM,SM = D)
43080  LET "$" +  STR$ (D) + ",1,1," +  STR$ (CM) + ",0,0," + AD$
43082  CALL  FN "VERIFY INIT"(AD,D)
43085  PRINT : INPUT "RECORD NUMBER ?";NO
43090  CALL  FN "VERIFY RECORD"(N1 = AD,D,NO)
43095  GOTO 43085
43100  END  FN
```

]LISTFN"VERIFY CATALOG AUTO

```
44000  DEF  FN "VERIFY CATALOG AUTO"
44010  REM
44020 AD =  PEEK (773) * 256 +  PEEK (774)
44030  CALL  FN "HEXA"(AD$ = AD)
44040  INPUT "DRIVE ?";D
44045  CALL  FN "MAX PARAM"(CM,TM,SM = D)
44050  LET "$" +  STR$ (D) + ",1,1," +  STR$ (CM) + ",0,0," + AD$
44060  CALL  FN "VERIFY INIT"(AD,D,AW,AQ)
44070 NO = 0:NC = 1:AZ = AD
44080  CALL  FN "CALCUL MM"(MM = AD,NC)
44090 NX =  PEEK (MM + 15) * 256 +  PEEK (MM + 16)
44100  CALL  FN "SEARCH KEY/NO"(P,T,S,A,L = AD,NO,NC)
44110  CALL  FN "READ BLOCK"(AD = D,P,T,S)
44120 NO =  PEEK (AD + A + 3) * 256 +  PEEK (AD + A + 4)
44130  IF NO = NX THEN 44500
44140  CALL  FN "VERIFY RECORD"(N1 = AZ,D,NO,AW,AQ)
44150 NO = N1: GOTO 44130
44500  PRINT : PRINT "ERRORS:"ER
44510  PRINT
44600  PRINT "FREE BLOCKS CONTROL": PRINT
44610  CALL  FN "FREE SPACE"(D,AZ,AW,AQ,AD,AD$)
44620  PRINT : PRINT : PRINT "ERRORS:"ER
44900  END  FN
```

]LISTFN"FREE SPACE

```
45000  DEF  FN "FREE SPACE"(D,TT,MR,MX,AD,AD$)
45010 CY =  PEEK (TT + 40):SE =  PEEK (TT + 41):OT =  PEEK (TT + 42)
45015 NE =  - 2
45020  IF CY =  PEEK (TT + 43) AND SE =  PEEK (TT + 44) AND OT =  PEEK (TT
     + 45) THEN 45900: REM  FIN
45030  CALL  FN "CONVERT TRACK"(C,T,S,0 = CY,SE,OT,MR,MX)
45040  LET "$" +  STR$ (D) + ",1,1," +  STR$ (C) + "," +  STR$ (T) + "," +
    STR$ (S) + "," + AD$
45050  IF  PEEK (189) THEN  PRINT " FUNCTION ERROR": STOP
45055  IF CY =  PEEK (TT + 43) AND SE =  PEEK (TT + 44) AND OT =  PEEK (TT
     + 45) THEN 45900: REM  FIN
45060  PRINT C" "S" "T" " INT (0 / 16)
45070  IF MA%((OT - 16 *  INT (OT / 16)) * 256 + CY,S, INT (0 / 16)) THEN
     PRINT " ERROR : RECORDS "NE" & "MA%(256 * (OT - 16 *  INT (OT / 16))
     + CY,S, INT (0 / 16))" SAME SPACE":ER = ER + 1
45080 MA%((OT - 16 *  INT (OT / 16)) * 256 + CY,S, INT (0 / 16)) = NE
45090  IF  PEEK (AD + 0) = CY AND  PEEK (AD + 0 + 1) = SE AND  PEEK (AD +
    0 + 2) - 16 *  INT ( PEEK (AD + 0 + 2) / 16) = OT - 16 *  INT (OT / 1
    6) THEN 0 =  INT ( PEEK (AD + 0 + 2) / 16) * 16: GOTO 45055
45100 CY =  PEEK (AD + 0):SE =  PEEK (AD + 0 + 1):OT =  PEEK (AD + 0 + 2)
45110  GOTO 45020
45900  END  FN
```

④

PETIT GUIDE

DU DEPANNEUR

I - LES PISTES LIBRES

SYMPTOME :

FUNCTION ERROR

en : - création de fichier
- réorganisation

éventuellement en :
- SAVE
- WRITE
- UPDATE

DIAGNOSTIC PROBABLE :

Les pistes récupérées lors de destruction de fichier ou de réorganisation sont mal chainées.

CONTROLE :

DEBUG-ANAL PISTE doit détecter une erreur.

REMEDE :

1) lère méthode : détruire les pistes récupérées
Pour cela charger en mémoire la DCB$ du drive (par exemple le 0) en faisant :

```
] HIMEM : 5 * 4096
] LET"$0,1,1,34,0,0,$5000
] CALL-151 passage au moniteur
* 50 FC-50FF
  aa bb cc dd
* 50FC : cc dd
CTRL/C
] LET"$0,2,1,34,0,0,$5000
```

indiquer ici le numéro du dernier cylindre. exemple :
floppy 5' → 34
CII D140 → 195

2) 2ème méthode : utiliser les modules existants

```
50000   DEF FN " FLAG TRACK " (D)
50010   CALL FN " READ DCB$ " (TT = D)
50020   POKE TT + 252, PEEK (TT + 254)
50030   POKE TT + 253, PEEK (TT + 255)
50040   CALL FN " MAX PARAM " (CM, TM, SM = D)
50050   CALL FN " HEXA " (A$ = TT)
50060   LET " $ " + STR$ (D) + ",2, 1, " + STR$ (CM) +
          " ,0,0," + A$
50070   END FN

]  D = ??
]  CALL FN " FLAG TRACK "(D)
```

3) 3ème, dernière et meilleure méthode
Reconstruire les chainages. Pour cela :

- lancer DEBUG-ANAL PISTE pour connaitre la liste des pistes utilisées

- lire les deux derniers octets de la DCB$ pour connaitre le " cylindre-tête " de la première piste logique jamais utilisée

- chainer ensemble les pistes libres (le chainage est à mettre dans les deux premiers octets du 1er secteur de la piste.

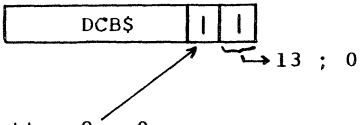
  Note :
  Dans DEBUG-ANAL PISTE, les pistes logiques à récupérer sont les " trous ".

- Mettre à jour la DCB$

EXEMPLE : Floppy 140K

```
piste     0  0      - catalog
          1  0      - catalog
          2  0      - catalog
          3  0      - file 1
          4  0      - file 1
          5  0      - catalog
          6  0      - file 2
          7  0      - catalog
          8  0        <-----+
          9  0      - file 2  | trous
          10 0      - catalog |
          11 0        <-----+
          12 0      - catalog
```

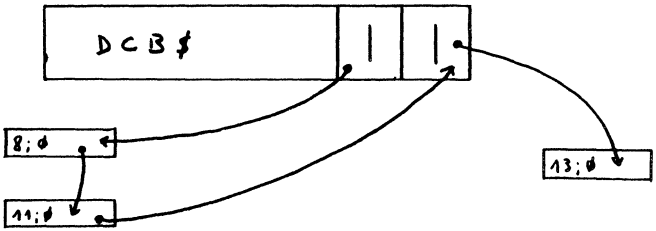
La première piste n'ayant jamais servie est 13 ; 0



Mettre 8 ; 0
dans 8 ; 0 mettre 11 ; 0
dans 11 ; 0 mettre 13 ; 0

C'est tout !!!

SCHEMA D'UN BON CHAINAGE :



RIEN N'EST ECRIT
DANS LA 13 ; 0

II - OBJET ABIME DANS LE CATALOG

SYMPTOMES :

- Après un LOAD de programme, le résultat est abérrant.

Exemple :

```
10 FOR I = 0 TO 100
20 HGR SPEED = GOTO XUHCOLOR
59431 HPLOTLIST GOTO
...
...
...
```

- Un masque ne peut plus s'ouvrir (ILLEGAL QUANTITY ERROR à l'utilisation alors qu'il est bien en 40 colonnes si l'on ne travaille pas en 80 colonnes !).

- L'ouverture d'un fichier provoque SYNTAX ERROR IN 0.

DIAGNOSTIC PROBABLE :

Des objets ont été abimés dans le catalog et sont écrasés par autre chose.

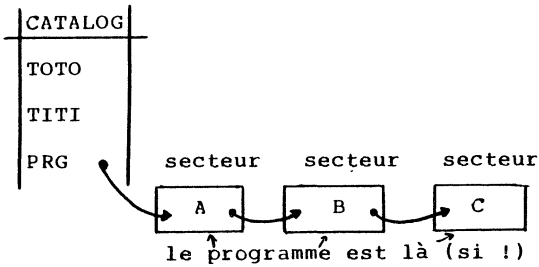
DERNIERE CHOSE A FAIRE ...

DETRUIRE LES OBJETS ABIMES !!!

POURQUOI ? (me direz-vous)

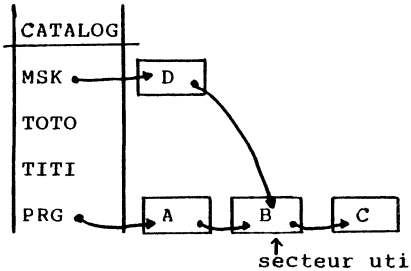
Prenons un exemple simple (simple, mais néanmoins à la page suivante)

Soit le programme PRG abimé :



Pourquoi est-il abimé ?
Parce qu'un autre objet utilise les mêmes secteurs, par exemple le masque MSK.

Complétons notre schéma :



Le masque a été créé <u>après</u> le programme et a pris sa place, mais lui est <u>bon</u>.

Si on détruit le programme PRG, les blocs A, B, C sont libérés et chainés aux autres blocs libres, donc réutilisables par un nouvel objet.

DONC

- on <u>détruit</u> le masque <u>MSK</u> qui est à son tour abimé

- on n'a <u>absolument pas</u> résolu le problème qui n'est que déplacé de PRG vers MSK.

E N C O R E P L U S G R A V E ! ! !
(mais oui, c'est possible !)

SI L'ON DETRUIT UN FICHIER ABIME :

Si un fichier est abimé par écrasement de pistes (DEBUG ANAL PISTE est là pour le confirmer) ou par détérioration de la DCB, le détruire est un véritable SUICIDE.
(et si le budget " photocopie " était plus important, je l'écrirai sur toute la page).

En effet, dans ce cas, on libère des pistes qui sont allouées à d'autres objets et on détruit alors les objets à la pelle (même si ce n'est pas le 18 juin).

LE BON CHOIX :

La solution à employer dépend de l'incident. Son choix demande donc un minimum d'astuce.

1) Les cas bénins :
- un programme, masque, binaire ou globala été détérioré
- DEBUG-ANAL PISTE ne signale pas d'erreurs
- DEBUG - VERIFY non plus

La méthode est alors la suivante et ne nécessite pas d'anesthésie générale :

a) Chercher la clé dans les pistes de clés de la DCB ; la trouver est relativement simple.
- Si la DCB a été réorganisée depuis la création de cet objet, elle est à sa place dans l'ordre alphabétique.

  - compter le nombre d'objets précédents (sauf les plus récents créés après la dernière réorganisation)

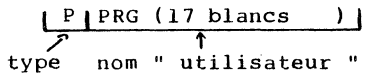
  - ajouter les objets détruits depuis la dernière réorganisation

  Soit n ce nombre, pour trouver la position de cette clé, utiliser le programme " SEARCH KEY/NO "

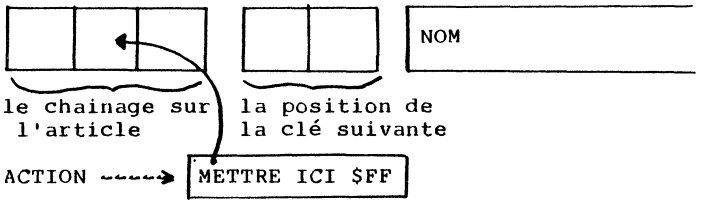
- Sinon, chercher à partir de la fin dans le TAS

b) La position de la clé étant déterminée lire le secteur qui la contient. Trouver le nom (21 caractères)

Dans notre exemple :



Les cinq octets précédents représentent :



En mettant à $FF le deuxième octet qui représente habituellement le secteur dans le chainage vers l'enregistrement l'article sera considéré comme DETRUIT.

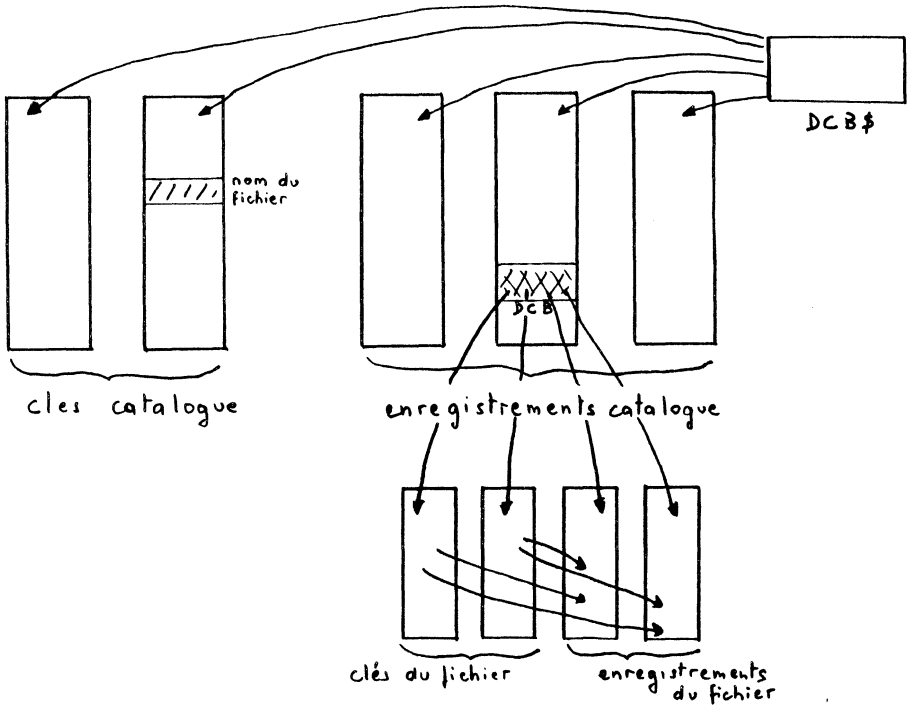
En revanche, les secteurs pointés par cette clé seront désormais ignorés, donc le phénomène destructif ne se propagera plus.

NB : Penser à réécrire le secteur de la clé.

LES CAS GRAVES :

Un fichier a été touché. Rappelons la structure des pistes logiques concernées par un fichier.

VOIR DESSIN PAGE SUIVANTE



TROIS CAS DE DESTRUCTION SONT POSSIBLES :

- le fichier a écrasé des objets du catalog
- des objets du catalog ont écrasé un fichier
- un fichier peut en cacher un autre

Dans tous les cas DEBUG-ANAL PISTE doit détecter l'erreur (ou ne pas aboutir et stopper en erreur)

A) UN FICHIER ECRASE LE CATALOG :

A - 1 - il écrase les enregistrements :

Dans ce cas LET" * "donne un catalog I M P E C C A B L E

- repérer les objets détruits et les " flager " avec l'algorithme.

Dans le cas d'une DCB de fichier abimée se reporter au cas des pistes détruites incorrectes pour récupérer les pistes que celui-ci occupait (à faire en fin de travail).
Les objets ainsi " flagés " sont hélas perdus pour la science.

- Normalement, l'écraseur, lui, est intact. Le sauver par FILE COPY sur une AUTRE SURFACE si les pistes libres sont douteuses.
Si l'utilisation de FILE COPY vous répugne, il existe une méthode astucieuse pour débloquer la situation (astuce ---> risque d'erreur).

- s'allouer une piste libre en mettant bien à jour la DCB$ (voir allocation de piste)

- copier la piste " écraseuse " sur cette nouvelle piste

- mettre à jour la liste des pistes occupées par le fichier

- ne pas oublier de resauver la DCB du fichier (par exemple en le réorganisant)

A - 2 - Il écrase les clés du catalog :

LET"*" ne fonctionne plus.

Solution : sauve qui peut

- copier tout ce qui est accessible sur une nouvelle surface

B - DES OBJETS DU CATALOG ECRASENT UN FICHIER :

- essayer de récupérer les articles en état dans le fichier et les copier sur un autre

- " flager " le fichier par la méthode décrite auparavant

- éventuellement récupérer les " trous " dans les pistes en refaisant les chainages.

C - UN FICHIER EN ECRASE UN AUTRE :

Même méthode qu'en B.

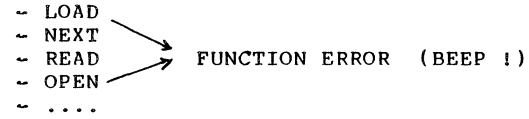
Si un fichier est intact :
- sauver les morceaux récupérables de l'autre
- le " flager "
- rechainer les pistes restantes

Sinon :
- sauver les restes des deux fichiers
- " flager " les 2 fichiers
- rechainer les pistes restantes

III - FUNCTION ERROR EN LECTURE

SYMPTOMES :

Divers :



DIAGNOSTIC :

D'une façon générale cette erreur signale une erreur dans les paramètres cylindre, tête, secteur qui dépassent les limites autorisées.
(Rappelons pour mémoire que le DOS 3.3 se contente dans ce cas d'un discret I/O ERROR.)

Les causes du problème peuvent être multiples et il convient de faire une étude dans chaque cas.

Les problèmes du type II étudiés précédemment peuvent relever de ce cas.

AUTRES POSSIBILITES D'ERREURS :

Même s'il n'y a pas de recouvrement avec un autre objet, détruire ce dernier est illusoire, car le FUNCTION ERROR arrêtera le DELET avant que les paramètres prennant en compte la destruction soient sauvegardés.

EXCEPTION :

Les pistes d'un fichier :

Elles seront récupérées jusqu'à la piste erronée dans la DCB mais l'opération est extrémement dangereuse car si une piste est complétement fausse, les autres sont-elles justes ???

METHODE A SUIVRE :

Il faut d'abord déterminer si l'erreur vient :

- d'un chainage article
- d'une piste fausse dans la DCB

DEBUG-ANAL PISTE vous dira immédiatement si les pistes du fichier (ou catalog) sont bonnes ou hors limites.

1) S'il s'agit d'un chainage d'un article :

* du catalog

Appliquer la méthode de " flageage " de l'erreur II.

* d'un fichier

Appliquercette méthode sur les clés du fichier. Si la méthode est trop difficile à appliquer (exemple : clé flottante difficile à repérer) copier le fichier par FILE COPY et le détruire.

2) S'il s'agit d'une piste fausse dans la DCB :

* copier les articles intacts
* " flager " le fichier (méthode II)
* récupérer les pistes perdues en refaisant les chainages.

IV - LE NG ERROR

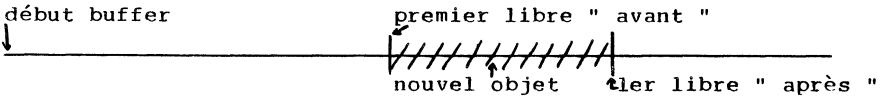
SYMPTOME :

- ouverture d'un masque, d'un fichier
- LOAD (plus rare)
- LET"#C,..."

DIAGNOSTIC :

L'erreur provient généralement de l'objet qui a été ouvert précédemment.
En effet, le processus d'ouverture d'un objet est le suivant :

- recherche de l'attribution éventuelle du numéro logique
- chargement de l'objet à partir du premier octet libre
- mise à jour du premier libre



Le premier libre est mis à jour d'après la longueur chargée depuis le disque.

MAIS !

La longueur de l'objet a été sauvée avec le module (octet TT+2/TT+3)

S'il n'y a pas concordance, l'erreur n'apparaitra qu'à la prochaine recherche d'un numéro logique non attribué, donc en général, au prochain chargement ou clear.

NOTE :

En mode direct, sans l'ordre LET")M" un clear des drives est effectué avant chaque commande, donc, un NG ERROR vous empèchera d'exécuter toute commande MEM/DOS 6502.
Pour sortir de cette situation, faire un LET"#C,$" par programme.

Exemple :

```
] 1 LET"#C,$": STOP
] RUN
BREAK IN 1
```

CONCLUSION :

Repérer avec précision l'objet abimé et appliquer les méthodes précédentes pour corriger.

CONSOMMATION DES CARTES MEM/DOS 6502

MODELE 16K (avec 2716) APPLE][

typique 180mA sur 5V
pointe 350mA sur 5V

MODELE 20K (avec 2732) APPLE][- APPLE ///

typique 140mA sur 5V
pointe 270mA sur 5V

" Au repos " 20mA sur 5V

NOTE :

Les valeurs maximum des pointes sont en réalité plus faibles car absorbées par les condensateurs.

IMPORTANT :

Pour le bon fonctionnement de votre système assurez-vous que la consommation totale des cartes ne dépasse pas 500mA sur le 5V.

MEMSOFT

3, rue Meyerbeer
06000 Nice
FRANCE
Tel : (93) 87 74 67
Tx : 461 916

1801 avenue of the Stars
Suite 300
Los Angeles CA 90067
U.S.A.
Tel : (213) 553 9033
Tx : 691 600

FRANCE
MEMSOFT FRANCE
62, bd Davout
75020 Paris
Tel : (1) 636 22 07
Tx : 215 825

SWISS
A.P.J. ELECTRONICS
Place Pepinet 2
Case Postale 2051
1002 LAUSANNE
Tel : (021) 232 164
Tx : 24 620

Dealer stamp – Cachet du revendeur

UNITED KINGDOM
DYNATECH MICROSOFTWARE
Microcomputing Design Centre
Rue du Commerce, Bouet
St. Peter Port
Guernsey CHANNEL ISLANDS
Tel : 0481 20155
Tx : 419 1130

BELGIUM
MICROTRAITEMENT SA NV
32 bd. Tirou
6000 Charleroi
Tel : (071) 31 74 75
Tx : 51 741