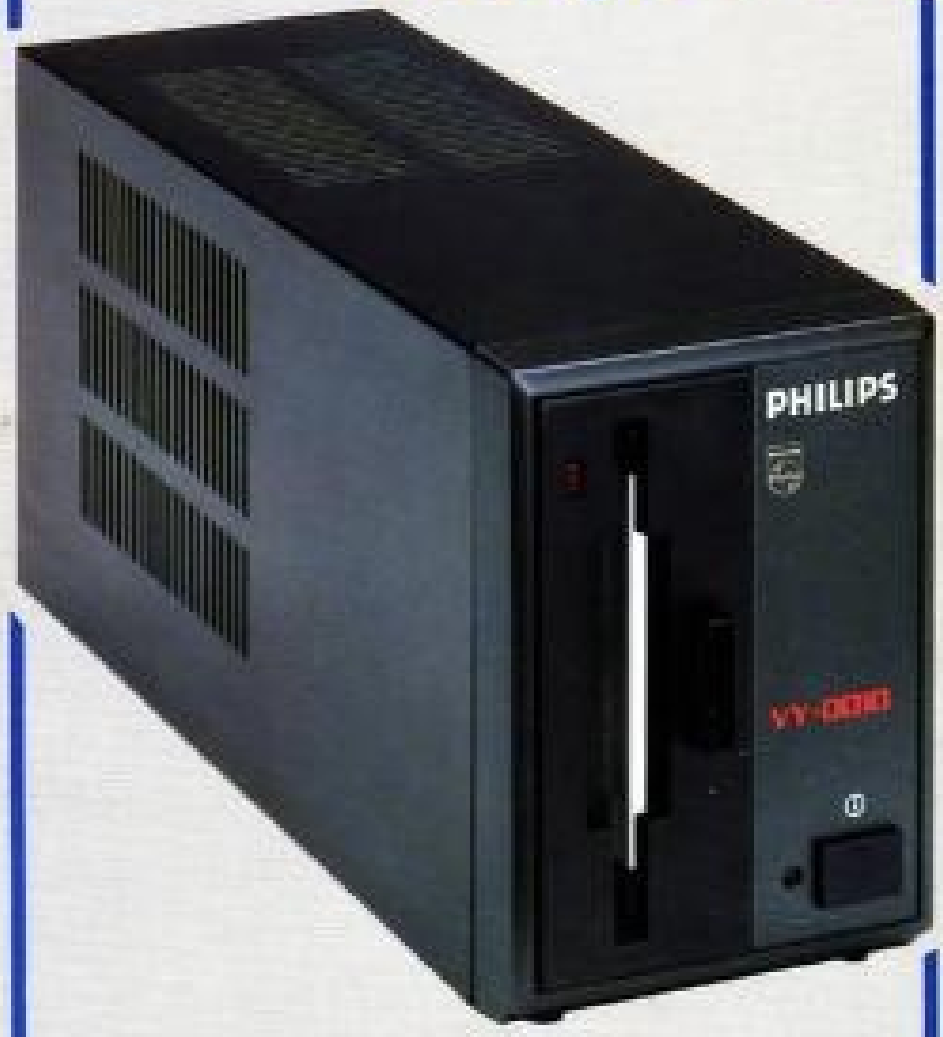
Manu Devos

LE LIVRE DU DISQUE M.S.X.



BCM

1) Chapitre 1 : Les disques du système MSX

1.1 Capacité en unités de disquettes

Un ordinateur MSX peut supporter au maximum 8 lecteurs de disquettes. En fait cela dépend beaucoup de votre appareil, car on ne connecte pas l'unité directement à l'ordinateur mais via une interface, le contrôleur-disque.

Ce contrôleur-disque peut gérer deux lecteurs de disquettes au maximum. Il contient une ROM de 16K qui se charge de ce travail : il doit nécessairement se loger dans un port cartouche ou slot. Le standard MSX a défini un maximum de 4 slots. Le slot 0, qui contient la ROM BIOS et BASIC, est en fait situé à l'intérieur de votre MSX, donc non accessible pour l'utilisateur. Selon la marque de votre ordinateur, vous pouvez trouver jusqu'à 3 slots d'extension (que vous utilisez habituellement pour vos cartouches de jeux). Si vous y glissez des contrôleurs-disques, vous disposerez d'un maximum de 3 contrôleurs-disques externes, plus un éventuel contrôleur interne (sur les ordinateurs avec un lecteur intégré). Cela nous donne donc une capacité théorique de 4 contrôleurs, soit 8 unités de disquettes (voir le chapitre 9).

1.2 Constitution physique d'une disquette

Une disquette est un plateau de forme circulaire recouvert d'une couche d'oxyde de fer sensible aux champs magnétiques. Il ne s'agit pas, comme pour nos disques 33 tours, d'une gravure mécanique d'un disque de plastique ; la disquette ressemble plus à la bande magnétique de nos enregistreurs qu'à un 33 tours.

L'écriture et la lecture des informations sur une disquette s'effectue par rotation de la disquette devant une tête de lecture/écriture tout comme la bande défile devant la tête de votre enregistreur. La vitesse de rotation du lecteur de disquettes est de 300 t/min ce qui correspondrait à une vitesse de défilement de 60 à 100 cm/sec pour un enregistreur qui aurait les mêmes performances que votre unité de disquettes.

Une autre différence est que la tête de lecture/écriture ne suit pas un sillon comme dans un 33 tours, mais doit se positionner sur une piste circulaire grâce à un mécanisme de déplacement. Il y a ainsi une série de pistes concentriques sur votre disquette dont le nombre dépend du type de support. On rencontre des lecteurs de 40 pistes (appelés SD - Single Density), et des lecteurs de 80 pistes (appelés DD – Double Density).

En général, les lecteurs MSX ont 80 pistes. On numérote les pistes de 0 à 79, la piste 0 étant la plus éloignée du centre de la disquette. Le plateau recouvert d'oxyde de fer est enfermé dans un boîtier plastique rigide pour les disquettes 3''1/2 et dans une jaquette souple pour les disquettes 5''1/4.

Une piste est divisée en secteurs. Les secteurs sont des portions de piste de contenance égale en nombre de caractères stockés (512). Le secteur est l'unité de lecture ou d'écriture sur la disquette, autrement dit, c'est la plus petite information que le hardware puisse lire ou écrire sur une disquette

en une seule opération. On numérote aussi les secteurs en donnant le numéro 0 au premier secteur de la piste 0 et le dernier numéro au dernier secteur de la dernière piste. Vous devez encore savoir que certains lecteurs de disquettes sont équipés de deux têtes de lecture/écriture, une sur chaque face de la surface magnétique. On parle dès lors de lecteurs Simple Face ou Double Face (Single Side – Double Side).

Deux méthodes d'enregistrement électronique des informations appelées FM, pour Frequency Modulation, et MFM pour Modified Frequency Modulation sont utilisées. Les contrôleurs MSX emploient la MFM qui permet de stocker deux fois plus d'informations que la méthode FM.

1.3 Les différents types de disquettes

Chaque type de disquette a reçu un code. La norme MSX a défini jusqu'à présent huit types de disquettes dont la liste suit.

CODE	CAPACITE	FORMAT	FACE	PISTE	SECTEUR
F8	362496	3''1/2	Simple	80	9
F9	730112	3''1/2	Double	80	9
FA	322560	3''1/2	Simple	80	8
FB	649216	3''1/2	Double	80	8
FC	179712	5''1/4	Simple	40	9
FD	362496	5''1/4	Double	40	9
FE	160256	5''1/4	Simple	40	8
FF	322560	5''1/4	Double	40	8

Attention ! Tous les contrôleurs-disques ne supportent pas tous les types de disquettes. C'est ainsi que les contrôleurs les plus récents ne supportent plus les disquettes de type FC à FF.

1.4 Le formatage

Une disquette neuve ne peut pas être employée directement pour sauver des programmes ou des fichiers. En effet, bien qu'elle contienne déjà une ou deux faces magnétisables, que les 40 ou 80 pistes soient bien présentes, la disquette neuve n'est pas encore « découpée » en secteurs. C'est intentionnellement que le fabricant l'a laissée dans ce état : ainsi, chaque système pourra découper la disquette suivant ses désirs. On rencontre des formats où chaque piste contient seulement 26 secteurs de 128 octets, ou encore 10 secteurs de 512 octets (format identique aux IBM-PC).

L'opération de formatage est réalisée conjointement par un programme spécial et par le hardware. Elle consiste à écrire au vol sur une piste complète en inscrivant une série de marques sur la piste, pour identifier le début et la fin de chaque secteur, leur ordre, leur longueur, la piste où ils se trouvent ainsi que le numéro de la tête qui les voit passer (voir le chapitre 4).

Si vous êtes novice dans l'emploi des disquettes, sachez que vous pouvez formater une disquette vierge par un CALL FORMAT (ou _FORMAT), sous BASIC ou FORMAT, sous DOS (quand le signe A> apparaît en début de ligne). Attention, cette commande efface complètement le contenu de la disquette si vous en formatez une qui contenait déjà des fichiers.

2) Chapitre 2 : Les noms des unités de disquettes et des fichiers

2.1 Les noms des unités de disquettes

En Disk-BASIC et en MSX-DOS, les huit unités de disquettes connectables à l'ordinateur portent chacune un nom pour facilité d'utilisation.

Ce nom est en fait une simple lettre, de A à H suivie du caractère « : ». L'unité A : est le premier lecteur de votre système, l'unité B :le second, et ainsi de suite jusqu'au lecteur H :. Si vous n'avez qu'une unité, elle portera le nom A:.

Dans ce cas, comment réaliser des copies de disquettes ? Heureusement la norme MSX a prévu une solution à ce problème. En effet, n'ayant pas détecté de seconde unité de disquettes, soit parce quelle n'existe pas, soit parce qu'elle n'était pas allumée au moment du démarrage du système, celui-ci va considérer cependant que vous avez pseudo deuxième lecteur qui portera effectivement le nom B :.

Lorsque vous demanderez une opération avec l'unité de disquettes B :, le système va vous inviter à retirer la disquette présente dans votre unique lecteur et à insérer celle qui serait présente dans votre second lecteur si vous en aviez un. Le message est le suivant :

Insert diskette for drive B : and strike a key when ready

ce qui signifie littéralement : « Insérez la disquette dans le l'unité de disquettes B : et enfoncez une touche lorsque vos êtes prêt ».

A partir de ce moment, votre unique lecteur est devenu le lecteur B :. De même, lorsque vous voudrez revenir au lecteur A :, le système produira le même message mais en vous invitant cette fois à insérer la disquette dans le lecteur A :.

Si vous avez quatre unités, déterminer quel lecteur porte le nom A :, B :, C : ou D : est un peu plus compliqué ; nous devons envisager trois éventualités.

1 – Les unités de disquettes sont toutes sous tension à l'allumage du système MSX :

Si vous allumez l'ordinateur normalement (c'est-à-dire allumer d'abord toutes les unités de disquettes et puis l'ordinateur), les unités A : et B : seront celles connectées au contrôleur-disque se trouvant dans le slot de numéro le plus bas. L'unité A : sera celle dans le sélecteur interne est sur A, l'unité B :, celle dont le sélecteur interne est sur B. Quant aux unités C : et D :, ce seront celles

connectées au contrôleur présent dans l'autre slot. L'unité C : sera celle dont le sélecteur interne est sur A, l'unité D :, celle dont le sélecteur interne est sur B.

Prenons l'exemple du VG 8235 de PHILIPS. Son contrôleur-disque intégré se trouve dans le slot 3. Le lecteur intégré à son sélecteur interne sur A. Connectons un deuxième lecteur avec le sélecteur sur B au connecteur pour second lecteur (à l'arrière de la console). Dans ce cas l'unité A : est le lecteur intégré et l'unité B : le lecteur extérieur.

Ajoutons un contrôleur-disque dans le slot 1 (sur le dessus de l'ordinateur). Connectons deux unités extérieures à ce contrôleur avec leurs sélecteurs respectivement sur A et B. Les lecteurs A : et B : deviennent maintenant les unités A et B du contrôleur que nous venons d'ajouter et les lecteurs C : et D : respectivement l'unité intégrée et la première unité que nous avions ajoutée, parce que le contrôleur que nous avons ajouté se trouve dans un slot de numéro inférieur (1) au contrôleur intégré (3). Tout ce qui vient d'être dit n'est valable que si les lecteurs extérieurs étaient allumés au moment du démarrage de l'ordinateur.

2 – Les unités extérieures connectées ne sont pas mises sous tension :

Dans le cas où les unités extérieures sont éteintes au moment de l'allumage de l'ordinateur, celui-ci ignore combien de lecteurs sont connectés à chaque contrôleur. Il va dès lors supposer qu'il n'y en a qu'un par contrôleur, c'est-à-dire 2 dans notre exemple (1 sur chaque contrôleur). L'unité disque A : sera donc le lecteur A du contrôleur que nous avons ajouté et, suivant la même procédure que nous avons décrite au début de ce chapitre, une unité B : sera allouée à ce même lecteur A. De même , le lecteur intégré aura comme nom C : et D :. L'unité disque B connectée à l'arrière de la console et l'unité B du contrôleur ajouté seront inaccessibles, même si on les allume après le démarrage.

3 – Les unités extérieures connectées ne sont pas mises sous tension ; la touche CTRL est maintenue enfoncée pendant la phase d'initialisation :

La dernière possibilité, c'est d'allumer l'ordinateur avec les lecteurs extérieurs éteints et de maintenir la touche CONTROL enfoncée jusqu'au BIP sonore. Ce cas est identique au cas ci-dessus (2), excepté que le lecteur A du contrôleur ajouté et l'unité intégrée ne seront pas partagés en deux. Dès lors, le lecteur extérieur A sera l'unité A : et le lecteur intégré l'unité B :. Il n'y aura pas de lecteur C : et D : et les lecteurs extérieurs non-assignés ne seront jamais accessibles.

2.2 Les noms de fichiers

2.2.1 Qu'est-ce qu'un fichier ?

Comme son nom l'indique, un fichier est une collection de fiches. Chaque fiche d'un fichier d'adresses renseigne, par exemple, le nom et l'adresse d'une personne. Un agenda est un fichier reprenant, jour après jour, ce que nous avons à faire.

Le fichier-disque a la même origine. Il s'agit de regrouper sur la disquette et sous un même nom toutes les informations faisant partie d'un même ensemble logique et dans un ordre bien déterminé.

Ce peut être un fichier d'adresses classées par ordre alphabétique, un fichier contenant par ordre numérique toutes les instructions d'un programme, etc.

Le Directory (répertoire) est en quelque sorte le fichier des fichiers présents sur la disquette, classés par ordre chronologique.

Il existe en MSX, comme sur la plupart des micros professionnels, deux types de fichiers : le fichier séquentiel et le fichier à accès direct.

2.2.2 Le fichier séquentiel

Le fichier séquentiel est simplement la mise bout à bout d'une série d'informations, tantôt numériques tantôt alphabétiques, avec une marque de séparation entre chaque information. En BASIC, cette marque de séparation est le signe « , »(virgule) et est automatiquement insérée après une information numérique, mais doit être placée par le programmeur après une information de type alphabétique. La paire de caractères CR/LF est aussi une marque de séparation.

Le fichier séquentiel a comme avantage d'être simple à utiliser, d'être plus dense, mais a comme gros inconvénient de devoir être lu en séquence c'est-à-dire en commençant par le début du fichier. Impossible donc d'atteindre, en un seul accès, le nom ZEBU d'un fichier d'animaux classés par ordre alphabétique : il faudra lire tous les enregistrements précédents pour l'atteindre.

2.2.3 Le fichier à accès direct

Le fichier à accès direct (Random File), est plus compliqué à mettre en œuvre mais permet un accès direct à n'importe quel enregistrement (Record) immédiatement. En effet, la lecture ou l'écriture se fait par numéro d'enregistrement. Par contre les informations doivent y être déposées de façon organisée. C'est ainsi qu'il faut déterminer d'avance la longueur de l'enregistrement.

L'enregistrement est l'unité d'accès au fichier. On pourra ainsi lire ou écrire directement l'enregistrement numéro 27 ou 456 sans devoir lire tous les précédents comme c'est le cas avec un fichier séquentiel.

En plus de la longueur de l'enregistrement, il faut aussi prévoir d'avance une répartition de cette longueur entre les différentes zones de l'enregistrement. Par exemple, dans un fichier d'adresses, 20 caractères pour le nom, 15 caractères pour le prénom, 40 caractères pour la rue, 4 caractères pour le numéro, et ainsi de suite pour chaque zone.

Il faudra aussi employer une technique spéciale pour déposer les informations dans chaque zone de l'enregistrement. Ainsi pour les zones alphabétiques, on pourra les aligner à gauche ou à droite et combler les positions inutilisées de la zone avec des espaces grâce à une instruction BASIC spécialisée. De même pour lez zones numériques, il faudra les convertir en chaînes de caractères avec une instruction différente suivant qu'il s'agit d'une variable entière, simple précision ou double précision.

2.2.4 La structure du nom de fichier

Chaque fichier dans la norme MSX porte un nom. Ce nom est divisé en deux parties : d'une part le nom du fichier proprement dit et d'autre part son extension ou son type. Ces deux parties sont séparées par le signe « . ».

Exemples :

PROGRAM.BAS
MSXDOS.SYS
LETTRE3.TXT
PACMAN.ASC
JEU.1
MENU
ADRESSES.MSX

a) Le nom de fichier proprement dit ne peut dépasser 8 caractères ;

b) Le nom de fichier ne doit pas nécessairement être suivi d'une extension. Dans ce cas on ne met pas de point ;

c) L'extension ne peut dépasser 3 caractères ;

d) Si vous incluez des minuscules dans le nom de fichier ou dans son extension, elles seront automatiquement converties en majuscules ;

e) Vous pouvez employer n'importe quel code y compris les chiffres et les codes graphiques mais à l'exclusion des caractères suivant : ; = + . " / [ ] et le code espace.

Les exemples ci-dessous montrent bien l'emploi de l'extension.

PROGRAM2.BAS	L'extension .BAS signifie que le fichier PROGRAM2 est en BASIC.
LETTRE3.TXT	L'extension .TXT signifie que le fichier LETTRE3 est du texte plutôt qu'un programme BASIC.
PACMAN.ASC	L'extension .ASC signifie que le programme PACMAN a été sauvé en ASCII plutôt qu'en binaire.
MSXDOS.SYS	L'extension .SYS signifie que le fichier MSX-DOS fait partie du système.
JEU.1	L'extension .1 signifie probablement qu'il s'agit du jeu numéro 1.
MENU	On voit ici que l'extension n'est pas obligatoire.
ADRESSES.MSX	L'extension peut aussi revêtir un sens pour l'utilisateur seulement.

2.2.5 Les extensions réservées

Vous avez libre choix du nom de l'extension pour vos fichiers, cependant certaines extensions sont réservées pour le MSX-DOS et d'autres pour certains logiciels comme les Assembleurs-Editeurs. Dès lors, nous vous conseillons de n'employer les noms d'extensions ci-dessous qu'en connaissance de cause.

.COM	Fichier en langage machine contenant le programme d'une commande du MSX-DOS. Il se charge en 0100H de la mémoire.
.BAT	Fichier BATCH du MSXDOS. Il s'agit d'une séquence de commandes MSXDOS en ASCII.
.BAS	Fichier programme en BASIC sauvé en binaire compressé.
.FOR	Fichier programme en langage FORTRAN.
.CBL	Fichier programme en langage COBOL.
.PAS	Fichier programme en langage PASCAL.
.PLI	Fichier programme en langage PLI.
.C	Fichier programme en langage C.
.ASM	Fichier programme en langage Assembleur (Z80).
.HEX	Fichier programme en langage hexadécimal Intel.
.MAC	Fichier source de Macro-assembleur.
.REL	Fichier langage machine relogeable.
.BAK	Fichier d'archivage créé par un éditeur.
.SYS	Fichier du système d'exploitation (DOS).
.TXT	Fichier de texte.
.ASC	Fichier programme en BASIC sauvé en ASCII.
.PIC	Fichier contenant une image graphique (PICture).
.LIB	Fichier librairie (contient des routines de programme).
.$$$	Fichier temporaire en CP/M.
.TMP	Fichier temporaire en MS-DOS (IBM).

Certains noms de fichiers sont réservés par le système d'exploitation ou par le Basic. Il ne faut donc pas les utiliser :

MSXDOS.SYS	Fichier d'initialisation du MSX-DOS
COMMAND.COM	Fichier qui contient les commandes résidentes du MSX-DOS.
AUTOEXEC .BAT	Fichier contenant une série de commandes MSX-DOS qui sera auto-exécuté à l'allumage de l'ordinateur. Vous pouvez le créer ou modifier son contenu suivant vos besoins.
AUTOEXEC.BAS	Fichier contenant un programme BASIC qui sera auto-exécuté lors du premier passage en BASIC ou à l'allumage de l'ordinateur si votre disque ne contient pas le fichier MSXDOS.SYS. Vous pouvez le créer ou modifier son contenu suivant vos besoins.

De plus, si vous comptez exploiter vos fichiers sur un PC IBM, nous vous recommandons de ne pas employer les noms de fichiers suivants :

IO.SYS, MSDOS.SYS, ANSI.SYS, VDISK.SYS, CONGIG.SYS

2.2.6 Les fichiers périphériques

Il existe cinq noms de fichiers tout à fait particuliers. Plutôt que de désigner un fichier sur le disque, ils désignent des appareils périphériques qui, grâce à ces noms spéciaux, apparaîtront au système d'exploitation comme de simples fichiers disques. Ces noms de fichiers désignent l'imprimante, le clavier, l'écran et une entrée/sortie auxiliaire telle que l'interface RS-232. Ces noms sont :

LST, PRN, CON, NUL et AUX

LST et PRN désignent tous deux l'imprimante comme destination des données. Ainsi vous pouvez, par le choix du nom de votre fichier, destiner vos données à un vrai fichier disque ou vers l'imprimante.

CON (CONsole) désigne l'écran comme destination de vos données ou le clavier comme source de vos données. Dans ce dernier cas, l'entrée par le clavier se termine par une marque de fin de fichier que vous devrez poser en frappant CTRL-Z suivi de Return.

NUL correspond à un fichier Nul. Les données sont simplement écartées plutôt que d'entrer dans un fichier.

AUX destine vos données à un périphérique externe ou utilise ce périphérique externe comme source de données. Ce périphérique peut être par exemple une interface RS-232 de télécommunication connectée via un Modem à une ligne téléphonique.

2.2.7 Les caractères de substitution

Pour rechercher un nom de programme dont on ne se rappelle que l'extension (.BAS), il serait souhaitable de ne pouvoir afficher que les programmes ayant ce type d'extension, de même qu'il serait intéressant de rechercher tous les fichiers s'appelant « BUDGET », par exemple, quelles que soient leurs extensions.

A cet effet, nous disposons d'un premier caractère de substitution qui est le code astérisque (*). Il remplace une série de caractères quelconques. Son effet se limite à la première partie ou à l'extension du nom de fichier, mais pas aux deux en même temps.

Ainsi, la commande BASIC FILES « *.BAS » visualisera tous les fichiers ayant l'extension .BAS. La commande BASIC FILES « BUDGET.* » visualisera tous les fichiers s'appelant BUDGET, par exemple BUDGET .JAN, BUDGET.FEV, BUDGET.MAR, BUDGET.AVR…

Bien entendu, on peut aussi utiliser l'astérisque des deux côtés du nom de fichier comme c'est le cas dans la commande FILES « *.* » qui visualisera absolument tous les fichiers de votre disquette (la commande FILES sans paramètres produit le même effet).

Le deuxième caractère de substitution est le caractère « point d'interrogation » (?). Contrairement à l'atérisque, il remplace un seul et unique caractère du nom du fichier ou de son extension. Ainsi la

commande FILES « JEU ?.BAS » affichera les fichiers JEU1.BAS, JEU2.BAS, JEU3.BAS, JEUA.BAS si de tels fichiers existent sur la disquette bien entendu.

On peut employer plusieurs codes de substitution dans un nom de fichier comme dans la commade FILES « J?U ?.* » qui affichera les fichiers JEU1.BAS, JEU2.BAS, JEU3.BAS, JEUA.BAS, JEUX.OBJ, JOUE.TXT, JOUR.COM.

3) Chapitre 3 : Le Disk-BASIC

Le Disk-BASIC est une extension de commandes, instructions et fonctions du MSX-BASIC pour couvrir tous les aspects de la manipulation des fichiers sur disquette.

Le Disk-BASIC devient actif dès qu'une interface-disque a été insérée dans un des slots de votre MSX et que le démarrage de l'ordinateur s'effectue SANS enfoncer la touche SHIFT (=majuscules) et en l'absence de disquette dans le lecteur A :. Si une disquette est insérée dans le lecteur A :, le système ne sélectionnera le Disk-BASIC que si cette disquette NE contient PAS le fichier MSXDOS.SYS.

Dans ce chapitre, nous ne décrirons que les mots spécifiques au Disk-BASIC, en supposant que le MSX-BASCI est connu. L'ordre de présentation est uniquement dicté par des considérations d'ordre didactique. Référez-vous à la table des matières pour trouver rapidement la syntaxe d'un mot-clef particulier.

La notation de la syntaxe est standard. En voici un court résumé et, à titre d'exemple, prenons l'instruction INPUT.

INPUT [« <MESSAGE> »;] <VARIABLE> [, VAR2, VAR3, … VARn]

Les crochets signifient que tout ce qui s'y trouve inclus est optionnel. Ainsi dans l'exemple ci-dessus, le message, les guillemets et le point-virgule ne sont pas obligatoires.

Les symboles < > signifient que le texte qu'ils encadrent doit être fourni par vous.

Vous verrez également apparaître les symboles suivants :

<nom fichier > Ce qui se trouve entre les symboles <...> doit être un nom de fichier valide sous la forme d'une chaîne.

<spéc. fic.> Ce qui se trouve entre les symboles <...> doit être une spécification de fichier, c'est-à-dire un nom d'unité de disquette et un nom de fichier, <A:ESSAI.BAS>, par exemple. Le nom de l'unité n'est pas obligatoire. Le système cherchera alors le nom du fichier dans l'unité dernièrement spécifiée. La spécification de fichier peut être donnée par une constante chaîne ou une variable chaîne.

3.1 Les commandes de gestion des fichiers

3.1.1 SAVE

SAVE <spéc. fichier> [,A]

Cette instruction est équivalente à l'instruction CSAVE du BASIC normal. Elle permet de sauver un programme BASIC résident en mémoire sur un fichier disque. L'option A permet de sauver le fichier en ASCII, autrement le fichier est sauvé en format binaire compressé (voir le chapitre 4 pour ces différents formats).

<spéc. fichier> est une chaîne de caractères ou une variable qui spécifie le nom de l'unité et le nom du fichier. Le nom de l'unité, le nom du fichier et l'extension doivent se conformer aux règles décrites dans les chapitres 2.1 et 2.2.4.

Si le fichier existe déjà sur la disquette, son contenu sera remplacé par le programme se trouvant en mémoire. Le format ASCII prend plus de place sur la disquette, mais permettra une relecture de ce fichier par les instructions comme MERGE en BASIC ou TYPE en MSX-DOS qui veulent que ce fichier soit en ASCII.

Attention ! Pour sauver un programme sur cassette, vous utiliserez les instructions CSAVE et SAVE pour sauver le programme respectivement en binaire compressé et en ASCII. Mais pour la disquette, les instructions deviennent SAVE et SAVE…,A pour le même cas.

Le nom de fichier ne peut pas contenir de caractères de substitution, et si le nom de disque n'est pas spécifié, le programme sera sauvé sur le lecteur couramment sélectionné.

SAVE « PROGRAM2.BAS »	Sauve le programme résidant en mémoire sous le nom « PROGRAM2.BAS » sur l'unité couramment sélectionnée.
SAVE « B :JEU.001 »	Sauve le programme résidant en mémoire sous le nom « JEU.001 » sur l'unité B :.
SAVE « A:JEU.002 »,A	Sauve le programme résidant en mémoire sous le nom « JEU.002 » sur l'unité A : en format ASCII.
10 SAVE « JEU.003 »	La commande SAVE est aussi une instruction et peut donc être intégrée à un programme.
10 P$= »JEU.0040 » 20 SAVE P$	On peut également mettre le nom du programme dans une variable.

On peut également sauver un programme sur le fichier périphérique spécial AUX mais cette opération est réservée à ceux qui disposent d'une interface RS-232 (voir chapitre 2.6)

SAVE « AUX : »,A	Sauver le programme dans un autre ordinateur couplé au nôtre par une paire d'interfaces RS-232.

3.1.2 LOAD

LOAD <spéc. fichier> [,R]

Cette instruction permet de charger en mémoire un fichier contenant un programme BASIC. Ce fichier peut avoir été sauvé en format binaire compressé ou en ASCII.

<spéc. fichier> représente le nom de l'unité à partir de laquelle il faut exécuter le chargement et le nom de fichier sous lequel le programme a été sauvé.

L'instruction LOAD détruit toutes les variables et le programme BAIS qui se trouveraient en mémoire avant le chargement du programme spécifié.

L'instruction LOAD va également fermer tous les fichiers laissés ouverts par un programme précédent. Cependant, si l'option [,R] est utilisée, le programme chargé sera lancé dès la fin du chargement, et tous les fichiers préalablement ouverts resteront ouverts.

Dès lors, LOAD avec l'option [,R] peut être utilisé pour enchaîner plusieurs programmes . Des informations pourront être passées d'un programme à l'autre.

Si l'option [,R] n'est pas programmée, le système charge le programme et revient à l'indicatif OK du BASIC. Vous pourrez dès lors LISTer ce programme.

LOAD « PROGRAM2.BAS »	charge le programme PROGRAM2.BAS à partir du lecteur dernièrement sélectionné et revient à l'indicatif du BASIC.
LOAD « A :JEU.001 »,R	charge le programme JEU.001 à partir du lecteur A : et lance l'exécution de ce programme immédiatement.
100 P$= « JEU.001 » 110 LOAD P$,R	On peut également mettre le nom du programme dans une variable.

Les deux programmes suivants illustrent le passage d'une variable d'un programme à un autre grâce à un fichier ouvert dans le premier programme et qui ne sera pas refermé lors du chargement du second programme car l'instruction LOAD a reçu l'option [,R]. Le détail des instructions 20-30-50-60 du premier programme et des instructions 20-30 du second programme vous sera donné plus loin.

```
10 CLS                          : PREMIER PROGRAMME
20 OPEN »VARIABLE » AS #1       :
30 FIELD #1,255 AS Z$           : Ce programme demande votre nom,
40 INPUT « TON NOM »;A$         : le sauve dans un fichier et
50 LSET Z$=A$                   : appelle le programme PROGRAM1.BAS
60 PUT 1,1                      : sans refermer le fichier « VARIABLE »
70 LOAD « PROGRAM1.BAS »,R      :

10 CLS                          : DEUXIÈME PROGRAMME
20 FIELD #1,255 AS T$           : appelé par le premier programme et
30 GET #1,1                     : utilisant le fichier ouvert par
40 PRINT « TON NOM EST »;T$     : le premier programme
```

3.1.3 BSAVE

BSAVE <spéc. fichier> ,<départ>, <fin> [,<exécution>]

Sauve un programme en langage machine résidant en mémoire de l'adresse <départ> (minimum=&H8000) jusqu'à l'adresse <fin> (maximum=&HFFFF) sur le lecteur sous le nom spécifié par <spéc. fichier>.

Si le nom du lecteur n'est pas précisé dans <spéc. fichier>, le programme sera sauvé sur le lecteur dernièrement sélectionné. L'adresse d'<exécution> optionnelle permet de préciser à quelle adresse l'exécution du programme devra démarrer lors d'un chargement futur de ce programme. En son absence, le programme démarrera à l'adresse de <départ>.

BSAVE <spéc. fichier> ,<départ>, <fin> , S

Cette dernière formulation est identique à la première, sauf que le code sauvé sur disque n'est pas un programme en langage machine résidant en mémoire RAM mais plutôt une image stockée dans la mémoire RAM du Vidéo Processor et que dès lors l'adresse minimum est 0000H et l'adresse maximum est 3FFFH pour un MSX1 et FFFFH pour un MSX2.

```
10 KEY1, « PRINT »
20 KEY2, «INPUT»
30 KEY3, «GOTO»
40 KEY4, «LOCATE»
50 KEY5, «DATA»
60 KEY6, «SCREEN»
70 KEY7, «COLOR»
80 KEY8, «CIRCLE»
90 KEY9, «LEFT$(»
100 KEY10, «RIGHT$(»
100 BSAVE « FUNCTION.KEY »,&HF87F, &HF9E1
```

Sauve les positions RAM FBF7 à F9E1 sur le lecteur courant sous le nom « FUNCTION.KEY ». Cette zone de mémoire contient le texte affecté aux 10 touches de fonction. Ainsi, si vous modifiez le contenu du texte des touches de fonction, il vous suffira de recharger ce fichier pour le rétablir comme à l'origine.

```
10 SCREEN 2
20 CIRCLE (128,96),95,15,,,1.4
30A$= « B:CERCLE.PIC »
40 BSAVE A$,&H0000,&H37FF,S
50 SCREEN 0
60 PRINT « DESSIN SAUVE SOUS LE NOM » ; A$
70 END
```

Ce petit programme dessine un cercle blanc et sauve l'écran dans un fichier appelé « CERCLE.PIC ». Constatez que le nom du programme peut aussi être mis dans un variable.

```
10 FOR I=&HC000 TO I+28
20 READ A$
30 POKE I,VAL(« &H »+A$)
40 NEXT I
50 BSAVE « RAPIDE.BIN »,&HC000,&HC01D,&HC01A
60 END
70 DATA D5, 21, 00, 00, 01, C0, 03, 7B, CD, 56
80 DATA 00, D1, 21, 00, 10, CD, 9C, 00, C0, 7D
90 DATA B4, 20, FB, 1C, 20, E6, 1E, 20, 18, E2
```

Ce petit programme installe un programme en langage machine à l'adresse C000H. Il le sauve sur le lecteur par défaut sous le nom « RAPIDE.BIN » en spécifiant qu'il s'étend de l'adresse C000H à l'adresse C01DH et que son exécution doit démarrer à l'adresse C01AH. Vous pourrez voir l'effet spectaculaire de ce programme en essayant les exemples de l'instruction BLOAD.

Chaque paramètre de l'instruction BSAVE peut être mis dans un variable excepté l'option « ,S ».

```
10 P$= »RAPIDE.BIN »
20 B=&HC000:E=&HC01D:J=&HC01A
30 BSAVE P$,B,E,J
```

3.1.4 BLOAD

BLOAD <spéc. fichier> [,R] [,décalage]

Cette instruction charge le fichier précisé dans <spéc. fichier> en mémoire. Ce fichier doit obligatoirement avoir été préalablement sauvé par l'instruction BSAVE.

Si aucun décalage n'est précisé, le programme contenu dans le fichier sera chargé à l'adresse précisée au moment de la sauvegarde du fichier par BSAVE, sinon le décalage sera ajouté à cette adresse.

L'option [,R] permet de lancer automatiquement l'exécution de ce programme à l'adresse précisée au moment du BSAVE.

BLOAD <spéc. fichier> ,S [,décalage]

Cette deuxième formulation sert à charger un fichier contenant une image dans la VidéoRAM (mémoire vidéo). L'image s'implantera à l'adresse à partir de laquelle elle a été sauvée plus un éventuel décalage.

BLOAD « FUNCTION.KEY »	Rappel du texte des touches de fonctions tel qu'il avant été sauvé dans l'exemple de l'instruction BSAVE.

```
10 SCREEN 2                          Charge l'image contenue dans
20 BLOAD « CERCLE.PIC »,S            le fichier CERCLE.PIC en VRAM
30 GOTO 30
```

```
BLOAD « A:RAPIDE.BIN »,R,16          Charge depuis le disque A : le fichier RAPIDE.BIN
                                     avec un décalage de 16 positions par rapport à l'adresse
                                     depuis laquelle il avait été sauvé et lance l'exécution de
                                     ce programme à l'adresse précisée lors du BSAVE plus
                                     16 positions. Ce programme affiche tous les caractères
                                     du MSX dans toutes les positions de l'écran. On peut
                                     l'arrêter en appuyant sur n'importe quelle touche.
```

```
10 SCREEN 0
20 INPUT « DONNEZ LE NOM DU DESSIN A AFFICHER »;D$
30 SCREEN 2
40 BLOAD D$,S
50 GOTO 50
```

Le nom du programme à charger et le décalage peuvent être mis dans un variable. Cela permet, comme dans l'exemple ci-dessus, de choisir à partir du clavier quel fichier doit être chargé et à quelle adresse.

3.1.5 MERGE

MERGE <spéc. fichier>

Ce mot clef est une commande et non une instruction. Cela signifie que, après son exécution, il provoque un retour à l'indicatif OK du BASIC, même s'il est inclus dans un programme.

MERGE permet de fusionner le programme BASIC résidant en mémoire avec un autre programme BASIC résidant sur la disquette dans le fichier précisé par <spéc. fichier>. Le fichier <spéc. fichier> doit avoir été sauvé en ASCII.

Fusionner signifie que tout le programme BASIC chargé se retrouvera en mémoire sans effacer les lignes du programme qui résidant en mémoire. Cependant, quand les lignes portent le même numéro dans les deux programmes, celles qui résidaient en mémoire seont effacées au profit de celles en provenance du fichier.

Exemple : le programme ci-dessous est sauvé en ASCII sous le nom « FUSION.ASC » par l'instruction SAVE « FUSION.ASC »,A

```
25 PRINT « ET LEURS DISQUES »
35 PRINT « DESTINES A L'USAGE FAMILIAL »
40 END
```

Entrons maintenant le programme suivant après avoir pris soin de taper NEW pour effacer la mémoire de l'ordinateur.

```
10 CLS
20 PRINT « LES ORDINATEURS MSX »
30 PRINT « SONT DE BEAUX APPAREILS »
35 END
```

Ce petit programme fonctionne tel qu'il est encodé mais nous allons le fusionner avec le programme FUSION.ASC en posant :

```
MERGE « FUSION.ASC »
```

Un LIST nous fera découvrir le programme résultant :

```
10 CLS
20 PRINT « LES ORDINATEURS MSX »
25 PRINT « ET LEURS DISQUES »
30 PRINT « SONT DE BEAUX APPAREILS »
35 PRINT « DESTINES A L'USAGE FAMILIAL »
40 END
```

On voit que les lignes 25 et 40 ont été ajoutées et que la ligne 35 a été remplacée par celle du programme FUSION.ASC.

3.1.6 NAME

NAME <anc u:nnnnnnnn.eee> AS <nouv u:nnnnnnnn.eee>

Cette instruction change le nom d'un fichier en un nouveau nom. La zone <anc u:nnnnnnnn.eee> peut comporter un nom d'unité de disquette et doit comporter un nom de fichier qui existe déjà sur le lecteur.

Cette instruction ne déplace pas le fichier, elle ne fait qu'en changer le nom. Si le nouveau nom existe déjà sur la disquette, le message « File already exists » (le fichier existe déjà) sera affiché.

Si vous précisez un nom d'unité différent dans <nouv u:nnnnnnnn.eee> par rapport à <anc u:nnnnnnnn.eee>, le message « Rename across disks » (changement de nom à travers deux disques) sera affiché.

Si le fichier <anc u:nnnnnnnn.eee> n'existe pas sur la disquette, le message « File not found » (fichier non trouvé) sera affiché. Dans aucun des trois cas d'erreur ci-dessus, le fichier ne changera de nom.

On peut utiliser les caractères de substitution * et ? avec cette instruction dans la mesure où le résultat reste cohérent.

```
NAME « RAPIDE.BIN » AS « CAR-SHOW.BIN »
```

Rebaptise le fichier RAPIDE.BIN en SHOW-CAR.BIN.

NAME « *.BIN » AS « *.OBJ »

Tous les fichiers se terminant par .BIN porteront le même nom que précédemment mais avec l'extension .OBJ.

NAME « JEU ?.BAS » AS « GAME ?.BAS »

Cette commande ne fonctionne pas car elle est incohérente. En effet, si le fichier JEU1.BAS est trouvé, il va être renommé GAME.BAS et non pas GAME1.BAS car le « ? » n'est pas à la même place dans les deux noms. Dès lors, si un fichier JEU2.BAS est trouvé, il devrait aussi être renommé GAME.BAS or ce nom a déjà été donné à JEU1.BAS. La commande correcte aurait du être la suivante : NAME « JEU ?.BAS » AS « GAM ?.BAS » où le « ? » est exactement à la même place dans chaque nom.

Les deux noms de fichiers peuvent aussi être placés dans des variables et l'instruction peut être placée dans un programme comme dans l'exemple suivant :

```
100 P$= « *.TXT » : N$= « *.TST »
110 NAME P$ AS N$
```

Il est interdit de renommer les fichiers périphériques spéciaux « CON : », « AUX : », « LST : » , « PRN : » ou « NUL : ».

3.1.7 KILL

KILL <spéc. fichier>

Cette commande détruit un fichier du lecteur. Cela signifie que le nom du fichier est retiré du Directory. Le contenu du fichier lui-même n'est pas effacé, mais de toute façon deviendra inaccessible. KILL peut être utilisé pour n'importe quel type de fichier (programme, séquentiel, direct ou MSX-DOS).

On peut également employer les caractères de substitution * et ? Dans le nom du fichier pour effacer plusieurs fichiers.

KILL « ESSAI.TST » enlève le fichier ESSAI.TST du lecteur.

KILL « B:JEU.00 ? » enlève tous les fichiers ayant pour nom « JEU » et pour extension « 00 ? ». Le « ? » valant pour n'importe quel caractère, les fichiers « JEU.001 – JEU.002 – JEU.00A seront enlevés si, bien sûr, de tels fichiers existent sur le lecteur.

KILL « *.* » cette commande est valable mais il ne faut l'utiliser qu'avec circonspection puisqu'elle détruit tous les fichiers de votre disquette.

```
10 INPUT « ENTREZ LE NOM DU PROGRAMME BASIC A EFFACER »;N$
20 IF LEN(N$)>8 THEN PRINT « MAXIMUM 8 CARACTERES »:GOTO10
30 IF INSTR(N$, « . »)<>0 THEN PRINT « LE NOM SEUL SVP »:GOTO 10
40 N$=N$+ « .BAS »
50 KILL N$
60 END
```

Ce petit programme demande le nom du programme à effacer, vérifie qu'il ne dépasse pas 8 caractères et qu'il n'a pas d'extension, sinon il affiche un message approprié. Il ajoute au nom du programme l'extension « .BAS » et efface enfin ce programme.

3.1.8 FILES – LFILES

FILES [<spéc. fichier>]
LFILES [<spéc. Fichier>]

Affiche à l'écran (FILES) ou sur l'imprimante (LFILES) le nom de tous les fichiers du lecteur courant.

Le paramètre <spéc. fichier> peut inclure soit un nom d'unité seul, soit un nom de fichier seul, soit les deux. Les caractères de substitution peuvent être employés dans le nom de fichier. Dans ce cas, les commades FILES et LFILES vont afficher le ou les noms de fichiers trouvés sur la disquette et qui correspondent à <spéc. fichier>.

FILES	Affiche à l'écran les noms de tous les fichiers présents sur le lecteur courant.
FILES « B : »	Affiche à l'écran les noms de tous les fichiers présents sur le lecteur B :.
FILES « JEU.001 »	Affiche JEU.001 si ce fichier existe sur le lecteur courant ou File not found s'il n'existe pas.
FILES « A:RAPIDE.BIN »	Affiche à l'écran RAPIDE.BIN si ce fichier existe sur la disquette A : ou File not found s'il n'existe pas.
FILES « *.BAS »	Affiche à l'écran les noms de tous les fichiers qui portent l'extension « .BAS » et qui sont présents surl l disquette courante.
FILES « B:BUDGET.* »	Affiche à l'écran les noms de tous les fichiers du B : portant le nom BUDGET quelles que soient lecteur

leurs	extensions, par exemple BUDGET.JAN-BUDGET.FEV – BUDGET.AVR – BUDGT.MAI
FILES « BUDGET.?A ? »	Affiche à l'écran le nom de tous les fichiers du lecteur courant portant le nom BUDGET pourvu que la seconde lettre de leur extension soit un 'A'. Par exemple BUDGET.JAN – BUDGET.MAR – BUDGET.MAI
LFILES « B : »	Imprime sur papier le nom de tous les fichiers du lecteur « B : »

Voici un programme « MENU » simplifié qui ne convient que pour les programmes BASIC.

```
10 CLS
20 FILES
30 INPUT « QUEL PROGRAMME VOULEZ-VOUS »;P$
40 LOAD P$
```

3.1.9 COPY

COPY <spéc. fichier source> TO <spéc. fichier dest>

Cette instruction permet de copier un ou plusieurs fichiers d'un disque vers un autre ou vers le même. Vous pouvez aussi donner un nom de fichier de destination différent.

Le fichier <spéc. fichier source> sera copié vers (TO) le fichier <spéc. fichier dest>. Le paramètre <spéc. fichier dest> peut prendre trois formes :

1) S'il est composé du nom de l'unité de disquette seul, le fichier source sera copié sous le même nom vers l'unité de destination.

2) S'il est composé d'un nom de fichier seul, le fichier source sera copié sous le nom désigné sur l'unité courante.

3) S'il est composé d'un nom d'unité de disquette et d'un nom de fichier, le fichier source sera copié vers l'unité désignée, sous le nom désigné.

Si le fichier de destination existe déjà sur l'unité de destination, il sera remplacé par la copie du fichier source.

Rappelez-vous que, si vous n'avez qu'un seul lecteur, le système va le partager en deux unités logiques A : et B :. Vous pouvez donc copier de disqette à disquette, le système vous invitant tantôt à insérer la disquette A :, tantôt la disquette B : dans votre unique lecteur (voir chapitre 2.1).

COPY « RAPIDE.BIN » TO « B : »

Le fichier RAPIDE.BIN de l'unité A : sera copié vers le lecteur B : sous le même nom.

COPY « RAPIDE.BIN » TO « VITE.BIN »

Le fichier RAPIDE.BIN du lecteur courant sera copié sur le même lecteur sous le nom VITE.BIN. Il y a donc maintenant deux fichiers identiques sur la même disquette, mais sous des noms différents.

COPY « A:RAPIDE.BIN » TO « B:VITE.BIN »

Le fichier RAPIDE.BIN de l'unité A : sera copié sur le lecteur B : sous le nom VITE.BIN.

On peut, bien entendu, employer des caractères de substitution dans les deux noms de fichiers pour autant que le résultat reste cohérent.

COPY « A :*.BAS » TO « B : »

Copie tous les fichiers de la disquette A :ayant l'extension .BAS vers le lecteur B : sous le même nom.

COPY « A :*.BAS » TO « B :*. »

Copie tous les fichiers de la disquette A :ayant l'extension .BAS vers le lecteur B : sous le même nom mais sans extension.

COPY « B:BUDGET.?A ? » TO « A:COMPTE.* »

Copie tous les fichiers du lecteur B : qui s'appellent BUDGET et dont l'extension contient un 'A' en seconde position de l'extension vers le lecteur A : sous le nom COMPTE avec la même extension. On peut aussi placer l'instruction COPY dans un programme et placer ses paramètres dans des variables.

```
10 INPUT « NOM DU PROGRAMME A COPIER »;S$
20 INPUT « NOM DU PROGRAMME COPIE »;C$
30 COPY S$ to C$
40 END
```

Il est aussi intéressant de noter que l'on peut employer les fichiers périphériques spéciaux CON, LST, PRN, NUL, AUX (voir chapitre 2.1).

```
COPY « CON » TO « A:ESSAI.TXT »

Mon cher Albert,

Ce petit mot pour te rappeler la réunion du CLUB MSX de mardi à 20H. N'oublie pas de prendre le programme ZIPZAP avec toi.

René

^Z
```

Après avoir entré la commande COPY, faites RETURN et tapez votre texte en appuyant sur RETURN chaque fois que vous désirez passer à la ligne. Pour provoquer la sauvegarde de votre texte dans le fichier ESSAI.TXT vous devez enfoncer CTRL-Z au début d'une ligne et puis faire RETURN. Après la sauvegarde vous reviendrez à l'indicatif OK du BASIC.

COPY « A:ESSAI.TXT » TO « CON »

Cette commande fera apparaître à l'écran le texte précédemment sauvé sur le disque. Il faut bien entendu que le fichier ESSAI.TXT soit en ASCII.

COPY « A:ESSAI.TXT » TO « LST »

Provoque l'impression sur papier du fichier ESSAI.TXT.

3.1.10 RUN

RUN <spéc. fichier> [,R]

Cette instruction charge le programme BASIC spécifié par <spéc. fichier> en mémoire et l'exécute immédiatement. L'instruction RUN détruit toutes les variables et toutes les lignes du programme qui aurait été présent en mémoire avant le RUN et fermera tous les fichiers laissés ouverts par ce programme. Si l'option [,R] est employée, alors les fichiers ouverts par le programme précédent ne seront pas refermés.

RUN « JEU.BAS »	Charge le fichier JEU.BAS et lance l'exécution du programme.

On peut mettre l'instruction RUN dans un programme et le paramètre <spéc. fichier> dans une variable.

```
110 P$= « JEU.BAS »
120 RUN P$
```

L’option [,R], laissant les fichiers du programme précédent ouverts, autorise le chaînage de plusieurs programmes et permet de passer des variables de l’un à l’autre via un fichier.

```
10 CLS
20 INPUT « TON NOM »;N$
30 INPUT « TON AGE »;A
40 OPEN « VARIABLE » AS #1
50 FIELD #1,248 AS Z1$, 8 AS Z2$
60 LSET Z1$=N$
70 LSET Z2$=MKD$(A)
80 PUT #1,1
90 RUN « PROGRAM.2 »,R
```

PROGRAMME « PROGRAM.2 »

```
10 CLS
20 FIELD #1, 248 AS Z1$, 8 AS Z2$
30 GET #1,1
40 A=CVD(Z2$)
50 PRINT A ; « EST L’AGE DE Mr »;Z1$
60 KILL « VARIABLE »
70 END
```

On peut également employer les noms de fichiers périphériques spéciaux CON et AUX. Seul AUX présente de l’intérêt lors d’une connexion entre deux ordinateurs par interfaces RS-232.

3.1.11 MAXFILES

MAXFILES = <expression>

Permet de spécifier le nombre maximum de fichiers qui pourront être manipulés simultanément ou si vous préférez qui pourront être ouverts concurremment.

Si <expression> vaut 0, alors seules les instructions SAVE – LOAD, BSAVE – BLOAD et MERGE pourront fonctionner mais aucun fichier en pourra être ouvert (voir OPEN).

Par défaut, le système fixe MAXFILES à 1 et autorise donc la manipulation d’un seul fichier.

Si MAXFILES est fixé à une valeur supérieure à 6, les six premiers fichiers pourront être des fichiers disques. Cela revient à dire qu’au niveau des disques, le nombre de fichiers manipulés simultanément ne pourra jamais dépasser 6 même si MAXFILES est fixé à 15.

Si vous comptez utiliser 6 fichiers disques avec un fichier imprimante (LPT:), un fichier écran (CRT:), un fichier écran graphique (GRP:) et un fichier cassette (CAS:) alors programmez un MAXFILES=10 et réservez les fichiers 1 à 6 pour les fichiers disques et les numéros 7 à 10 pour les autres périphériques.

L’instruction MAXFILES réserve de l’espace mémoire sous la région de communication des disques à raison de 267 octets par fichier (voir chapitre 5). Ainsi, un MAXFILES=15 va diminuer le nombre d’octets libres de 3738 par rapport à la valeur par défaut.

Attention !! Mettez toujours l’instruction MAXFILES au début de votre programme et avant toute déclaration de variables. Ne mettez pas non plus MAXFILES dans une sous-routine qu’on accède par GOSUB. En effet, la modification de la mémoire réservée provoque l’effacement des variables et de la pile (STACK). Cette remarque vaut aussi pour l’instruction CLEAR. Je la mentionne ici car aucune documentation de fabricant n'en parle.parle.

```
10 CLEAR 1000, &HDFFF
20 MAXFILES=15
30 …
```

3.2 Les Call du Disk-Basic

CALL est une instruction du BASIC normal qui provoque la recherche de l’instruction dont le nom suit immédiatement le mot CALL dans des ROM d’extension du BASIC. Le contrôleur-disque contient une ROM qui comprend deux instructions nouvelles.

3.2.1 CALL FORMAT

Cette commande permet de formater une disquette vierge ou d’effacer complètement une disquette qui contient déjà des fichiers.

Dès que vous aurez entré la commande et enfoncé RETURN, l’ordinateur affichera :

Drive name ? (A, B…)

pour vous demander dans quel lecteur vous comptez insérer la disquette à formater. Ensuite, et dépendant du fabricant de votre contrôleur-disque, un message de choix de format pourra vous être proposé. Par exemple :

1 – Single side
2 – Double side

Répondez-y en consultant la documentation concernant votre lecteur de disquettes. Le message :

Strike a key when ready

sera affiché, ce qui signifie « Enfoncez une touche dès que vous êtes prêt ». Insérez maintenant la disquette à formater dans le lecteur choisi et enfoncez une touche quelconque. Le formatage de la disquette commence immédiatement et le message :

sera affiché dès que l'opération sera terminée. Notez également que l'instruction CALL FORMAT peut être abrégée en _FORMAT.

3.2.2 CALL SYSTEM

CALL SYSTEM permet de quitter le BASIC pour retourner au MSX-DOS. Cette commande n'est valide que si le Disk-BASIC a été invoqué à partir du MSX-DOS (voyez, à ce sujet, le chapitre 5, position RAM &HF346).

Par cette commande, tous les fichiers ouverts en BASIC seront fermés et les données en mémoire effacées. CALL SYSTEMP peut être abrégé en _SYSTEM.

3.3 La manipulation d'un fichier séquentiel

Jusqu'à présent, nous avons vu les commandes de gestion des fichiers et comment sauver et charger un programme sur/d'une disquette, mais le Disk-BASIC permet aussi de créer, de lire ou d'écrire des fichiers qui ne contiendraient pas un programme mais plutôt vos propres données.

On pourrait, par exemple, écrire un programme qui permettrait de consulter un fichier bibliothèque, de lui ajouter ou enlever des fiches ou même d'en modifier.

Un fichier séquentiel est celui où toutes les informations qu'on y dépose sont mises bout à bout avec simplement une marque de séparation entre chaque donnée. La longueur de chaque donnée peut varier à tout moment. Il est facile à employer mais a comme inconvénient de devoir être lu ou écrit en séquence.

Supposons que nos désirons créer un fichier d'adresses de os parents et connaissances. Dans ce cas, le type séquentiel convient très mal, car pour créer le fichier, il faudra introduire la liste ds personnes par ordre alphabétique – ce qui n'est guère facile – et ensuite, lors de la consultation de ce fichier, il faudra lire toutes les informations précédant celles de la personne recherchée avant de pouvoir connaître son adresse.

L'emploi idéal d'un fichier séquentiel est celui ou les informations à mémoriser dans ce fichier sont connues dans un ordre chronologique et où la longueur de chaque information ne peut pas être standardisée.

A titre d'exemple, la tenue d'un journal personnel. En effet, dans un journal, les informations sont notées jour après jour et le contenu de ces informations peu varier énormément quant à leur nombre et à la longueur de chacune d'entre elles. D'autre part, la lecture d'un journal intime n'est significative que si on le lit en séquence. Comment connaître l'évolution de pensée ou l'état d'esprit de quelqu'un en lisant ce qu'il a indiqué dans son journal le 3 mai puis le 8 novembre et enfin le 23 février ?

Nous avons bien dit dans le paragraphe précédent qu'il s'agissait d'un emploi idéal du fichier séquentiel, heureusement il offre beaucoup d'autres ressources. Ainsi, quand le temps d'accès à une information précise n'est pas un élément critique pour le choix du type de fichier, alors le fichier séquentiel est certainement à recommander grâce à sa facilité d'emploi.

Le fichier sur disquette, en général, se manipule exactement comme le carnet d'adresses que nous avons tous chez nous. Lorsque nous voulons consulter l'adresse d'une personne, nous commençons par CHOISIR le bon carnet d'adresses (celui des amis et connaissances ou l'annuaire téléphonique ou le carnet d'adresses des relations professionnelles, etc). Ensuite nous OUVRONS ce carnet, nous LISONS l'information qui nous intéresse ou nous ECRIVONS une nouvelle adresse et enfin nous REFERMONS le carnet d'adresses.

De même les fichiers sur disquette, qu'ils soient séquentiels ou à accès direct, seront manipulés suivant ces trois étapes :

FONCTION LOGIQUE	INSTRUCTION BASIC
1 - OUVERTURE	OPEN
2 – LECTURE ECRITURE	INPUT# PRINT#
3 - FERMETURE	CLOSE

INPUT# permet d'introduire dans une variable une donnée en provenance du fichier et PRINT# va permettre d'envoyer une information d'une variable vers le fichier.

3.3.1 L'ouverture d'un fichier séquentiel

OPEN <spéc. fichier> FOR <mode> AS [#] <numéro fichier>

L'instruction OPEN permet d'ouvrir le fichier désigné par <spéc. fichier> dans un des trois modes décrits ci-dessous et de fixer que ce fichier sera dorénavant référencé par son <numéro fichier> plutôt que par son nom.

Il y a trois modes d'ouverture d'un fichier séquentiel :

FOR INPUT	Signifie que le fichier est ouvert en lecture seulement. Comprenez donc bien qu'il est donc interdit d'écrire de nouvelles informations ou de modifier des informations déjà présentes dans ce fichier avec ce mode d'ouverture. D'autre part, le fichier doit déjà exister sur la disquette avant cette ouverture.
FOR OUTPUT	Ce mode permet de créer et d'ouvrir le fichier en écriture seulement. On pourra dès lors écrire des informations à partir du début du fichier. Si le fichier n'existe pas sur le disque au moment de l'ouverture, il sera créé mais ne contiendra encore rien. Si le fichier

existait déjà au moment de l'ouverture, il sera d'abord détruit puis recréé. Donc son contenu sera lui aussi nul.

FOR APPEND — Ce mode permet d'ouvrir un fichier existant en ajout d'écriture. Toute information écrite vers le fichier sera donc ajoutée à celles déjà présentes dans ce fichier. Ce fichier doit donc fatalement exister sur le disque avant l'ouverture.

Le paramètre <numéro fichier> doit être un numéro (ou une variable le contenant), de 1 à 6 (attention que MAXFILES autorise un maximum de 15 fichiers mais seuls les 6 premiers peuvent être utilisés comme fichiers sur disquette), sans pour autant dépasser la valeur fixée par l'instruction MAXFILES. Ce numéro va être associé au fichier dans le but que les autres instructions manipulant des fichiers puissent le référencer par son numéro plutôt que par son nom.

Cette association entre le nom du fichier et <numéro fichier> durera tant que le fichier n'aura pas été refermé.

Plus techniquement, l'instruction OPEN va affecter un des tampons mémoire réservés par l'instruction MAXFILES à ce fichier. Toutes les instructions d'entrée-sortie feront transiter leurs données par ce tampon. Si MAXFILES a réservé 6 tampons mémoire et que l'instruction OPEN référence le fichier sous le numéro 3, ce sera le tampon 3 qui sera affecté à ce fichier.

En plus de cela, OPEN va rechercher le fichier dans le Directory du disque, éventuellement créer le fichier et va installer ce que l'on appelle un FCB (File Control Block, le FCB sera décrit et expliqué dans le chapitre 5).

Un fichier séquentiel peut être ouvert en INPUT sous plusieurs numéros tandis que les autres modes imposent une seule référence par nom de fichier. Les exemples d'emploi de cette instruction vous seront donnés en 3.3.5.

3.3.2 L'écriture dans un fichier séquentiel

PRINT#<numéro fichier>, <liste d'expressions>

PRINT# fonctionne exactement comme l'instruction PRINT bien connue à la différence que les données sont écrites dans un fichier plutôt que sur l'écran. Il faut déduire de cette phrase que les données ne sont pas compressées, sont en ASCII, et qu'elles sont déposées dans le fichier exactement sous la même apparence qu'un PRINT nous les montrerait à l'écran.

PRINT « IL » ; »FAIT » ; »CHAUD » montrerait à l'écran ILFAITCHAUD sans séparation entre les trois mots. Il en sera de même pour le PRINT# sur un fichier. Le but de la sauvegarde de données dans un fichier sur disquette étant la relecture future de ces données, il est impératif que chaque donnée soit séparée de celles qui l'entourent afin de garantir leur bonne compréhension.

<numéro fichier> est une constante, une variable ou une expression numérique qui spécifie quel fichier sera écrit. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro précisé dans l'OPEN du fichier dans lequel on désire écrire. Ce numéro ne

peut être inférieur à 1 ni supérieur au nombre répcisé dans MAXFILES avec de toute façon un maximum de 6.

<liste d'expressions> est une série de constantes, de variables ou de leurs expressions de type numérique ou chaîne de caractères séparées par les délimiteurs habituels de l'instruction PRINT. Ainsi le code « ; » ou l'espace permet la juxtaposition des données et le code « , » l'écriture des données au début de la zone de tabulation suivante. Si la liste des données ne se termine pas par « ; » ou « , » la paire de code CR-LF (Carriage Return – Line Feed) sera écrite dans le fichier.

Les expressions numériques sont écrites dans le fichier telles qu'elles apparaissent à l'écran. Cela signifie que le nombre est précédé d'un espace s'il est positif ou d'un « - » s'il est négatif. De plus, chaque nombre est suivi du code espace comme délimiteur.

Les expressions de type chaîne de caractères n'ont pas de délimiteurs automatiquement insérés. Pour pouvoir distinguer les différentes chaînes lors de la relecture du fichier, il faudra nous-même prévoir des délimiteurs. Le code réservé pour servir de délimiteur est la virgule.

Pour expliquer clairement l'aspect du fichier suivant le contenu des instructions PRINT#, nous allons prendre quelques exemples :

```
100 PRINT#1,10 ;-20;30*2
```

Cette première instruction envoie vers le fichier ouvert sous le numéro 1 les valeurs numériques 10, -20 et 60. L'image de nos données sur le disque sera la suivante :

_10_-20_60__CRLF

Le symbole « _ » représente le code espace, CR représente le code retour chariot et LR le code « à la ligne » (0AH – 10).

Chaque valeur numérique est suivie d'un espace qui sert de délimiteur de telle sorte que nous pourrons plus tard relire ces valeurs du fichier et les attribuer à des variables numériques par l'instruction suivante :

```
320 INPUT#1, A,B,C
```

A contiendra 10, B contiendra -20 et C, 60

Voici un extrait de programme qui écrit un nom et un prénom dans un fichier :

```
100 N$= « DURAND »
110 P$= « ALBERT »
120 PRINT#1,N$ ;P$
```

L'image créée dans le fichier sera exactement identique à ce qu'un PRINT vers l'écran aurait donné à savoir :

DURANDALBERTCRLF

Nous constatons qu'aucun séparateur n'a été placé entre DURAND et ALBERT. Si nous relisons plus tard ce fichier pour attribuer les valeurs lues aux variables N$ et P$ par l'instruction :

```
149 INPUT#1,N$,P$
150 PRINT N$
```

Nous serons probablement surpris de constater que N$ contient DURANDALBERT ; cela est dû au fait qu'aucun séparateur n'a été placé dans le fichier lors du PRINT#. Nous devons donc nous-même installer ce séparateur au moment de l'écriture dans le fichier. Le seul code de séparation valable est la virgule. Modifions donc notre programme comme suit :

```
100 N$= « DURAND »
110 P$= « ALBERT »
120 PRINT#1,N$ ; « , » ;P$
```

Nous obtiendrons alors l'image suivante dans le fichier :

```
DURAND,ALBERTCRLF
```

Et si nous relisons le fichier, cette fois DURAND sera affecté à N$ et ALBERT à P$.

Un dernier problème surgit maintenant. En effet, si notre chaîne de caractères est « CITROEN, 2CV, 1984 », nous voyons que des virgules font partie de la chaîne en elle-même et dès lors, lors de la relecture, cette chapine sera interprétée comme 3 chaînes distinctes.

La parade à ce problème est d'entourer la chaîne de guillemets dans le fichier même. Mais comment faire pour que ces guillemets s'écrivent dans le fichier. Et bien, exactement de la même façon qu'un PRINT normal à l'écran, nous employons la fonction CHR$(34).

Notons aussi qu'il faut entourer la chaîne de guillemets lorsque celle-ci contient des espaces significatifs avant les autres caractères, ou encore si elle contient des virgules, des points-virgules ou des codes CR-LF intégrés à la chaîne.

```
100 A$= « CITROEN,2CV,1984 »
110 PRINT#1,CHR$(34);A$:CHR$(34) ;
120 PRINT#1,CHR$(34) ; « OPEL,KADETT,1982;ENDOMMAGEE ;CHR$(34)
```

L'image suivante sera alors déposée dans le fichier :

```
« CITROEN,2CV,1984 » « OPEL,KADETT,1982;ENDOMMAGEE»CRLF
```

Lors de la relecture du fichier, la présence de guillemets autour des deux chaînes provoquera l'interprétation correcte des données.

```
190 INPUT#1;O1$,O2$                    O1$=CITROEN,2CV,1982
```

O2$=OPEL,KADETT,1982;ENDOMMAGEE

Exemples de syntaxe :

```
PRINT#2, « BONSOIR »
PRINT#3, 27 ;-42
PRINT#1, CHR$(34) ;«      ATTENTION»;CHR$(13);CHR$(10);CHR$(34)
PRINT#3, A$;B$;C$
PRINT#4, A;B;C;D;D$;A$
PRINT#N, A;B$ ; « , »;C$
PRINT#K+2,A;B
```

3.3.3 La lecture d'un fichier séquentiel

INPUT# <numéro fichier>,<liste de variables>

INPUT# lit un ou des éléments d'un fichier séquentiel et les assigne à une ou des variables du programme.

<numéro fichier> est une constante, une variable ou une expression numérique qui spécifie quel fichier sera lu. C'est l'instruction OPEN qui a associé un numéro au nom de fichier ; il faut donc reproduire ici le numéro précisé dans l'OPEN du fichier qu'on désire lire. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre fixé par l'instruction MAXFILES avec de toute façon un maximum de 6.

<liste de variables> représente une ou plusieurs variables séparées par le code virgule dont le type (Entier-Simple-Double-Chaîne) doit correspondre aux éléments lus du fichier sous peine d'erreur (Type mismatch). Il sera lu autant d'éléments du fichier qu'il y a de variables dans <liste de variables>.

Bien qu'il ,'y ait pas de « ? » affiché à l'écran comme dans l'instruction INPUT normale, les éléments présents dans le fichier doivent être exactement sous la même forme qu'ils seraient entrés au clavier dans l'instruction INPUT normale.

Pour les valeurs numériques, les espaces et les codes CR (0DH) et LF (0AH) précédant la valeur sont ignorés. Le premier caractère rencontré qui ne soit pas un espace, un CR ou un LF est considéré comme étant le début d'un nombre. Ce nombre sera considéré comme terminé lors de la rencontre d'un espace, d'un CR, d'un LF ou du code virgule. Si le premier caractère rencontré est différent du signe – ou d'un chiffre, ou si un des autres caractères rencontrés est différent d'un chiffre, le message « Type mismatch » sera affiché.

Pour les valeurs « chaînes de caractères », les espaces, le code CR et le code LF précédant la chaîne seront ignorés. Le premier caractère rencontré qui ne soit pas un espace, un CR ou un LF est considéré comme étant le début de la chaîne. Si le premier caractère de la chaîne est un guillemet ("), la chaîne sera constituée de tous les caractères présents entre le premier et le second guillemet rencontré.

Si le premier caractère n’est pas un guillemet, la chaîne sera considérée comme terminée dès l’apparition d’une virgule, d’un code CR, d’un code LF ou si 255 caractères ont déjà été introduits dans la chaîne.

Si une marque de fin de fichier (1AH) est rencontrée pendant un INPUT#, la variable en cours de constitution sera terminée. Si cette variable n’est pas la dernière de <liste de variables> le message « Input past end » (Input au-delà de la fin du fichier) sera affiché.

Exemples de syntaxe :

INPUT#1,A
INPUT#1,A$
INPUT#C,D$,N$
INPUT#3,A,A$,T(3),T$(B)
INPUT#2,B !,C %,D#,E$;F
INPUT#N+1,A$

3.3.4 La fermeture d’un fichier séquentiel

CLOSE [#]<numéro de fichier>[,[#]<numéro de fichier>][,…]

L’instruction CLOSE (fermer) conclut les opérations d’entrées/sorties sur un fichier.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l’instruction. C’est l’instruction OPEN qui a associé un numéro au nom de fichier ; il faut donc reproduire ici le numéro indiqué dans l’instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Une instruction CLOSE peut fermer plusieurs fichiers à la fois. Indiquez simplement les numéros de fichiers désirés, séparés par une virgule. Si CLOSE ne contient aucun numéro de fichier, tous les fichiers ouverts seront fermés.

L’association entre un nom de fichier et son numéro se termine dès la fin de l’instruction CLOSE. Le fichier peut dès lors être réouvert sous un numéro quelconque. De même, ce numéro pourra être réutilisé pour ouvrir n’importe quel fichier.

L’instruction CLOSE est très importante pour les fichiers ouverts en mode OUTPUT ou APPEND car c’est à ce moment que le code de fin de fichier (1AH) est placé dans le tampon mémoire du fichier et que ce tampon final est écrit sur le disque.

Ensuite, CLOSE va réécrire l’entrée du Directory correspondant à ce fichier afin d’y indiquer la nouvelle longueur de fichier, et la date et l’heure (si MSX2) de la fermeture de ce fichier. La FAT sera aussi réécrite sur le disque.

Les instructions END, CLEAR et NEW de même que toute modification d’un programme provoquent également la fermeture de tous les fichiers laissés ouverts.

Exemples de syntaxe :

```
CLOSE
CLOSE#1
CLOSE1
CLOSE#1,#2,#3
CLOSE 1,3,4
CLOSE A,T
CLOSE N+1
```

3.3.5 Exemples de programmes avec fichier séquentiel

3.3.5.1 Création ou extension d'un fichier d'adresses

```
10 CLEAR500:MAXFILES=1
20 ON ERROR GOTO 190
30 OPEN « ADRESSE.TXT » FOR APPEND AS #1
40 GOTO 60
50 OPEN « ADRESSE.TXT » FOR OUTPUT AS #1
60 CLS
70 INPUT « NOM »;N$
80 INPUT « PRENOM »;P$
90 INPUT « RUE »;R$
100 INPUT « No »;NO$
110 INPUT « CODE POSTAL »;CP
120 INPUT « LOCALITE »;V$
130 PRINT#1,N$ ; « , »;P$ ; « , » ;R$ ; « , » ;NO$ ; « , » ;CP;V$
140 PRINT « RETURN POUR CONTINUER    ESC POUR ARRETER »
150 A$=INKEY$:IF A$= «» GOTO 150
160 IF A$=CHR$(13) GOTO 60
170 IF A$<>CHR$(27)GOTO 140
180 CLOSE#1:CLS:END
190 IF ERL=30 AND ERR=53 THEN RESUME 50
200 IF ERR<>68 GOTO 220
210 PRINT « Disque protégé contre l'écriture »:GOTO 250
220 IF ERR<>70 GOTO 240
230 PRINT « Pas de disque dans le lecteur » : GOTO 250
240 ON ERROR GOTO 0
250 INPUT « Faites Return après correction du problème »;A$
260 RESUME
```

Les lignes 10 et 20 servent à fixer l'espace mémoire réservé aux chaînes, à fixer le nombre maximum de fichiers employés simultanément et à fournir le numéro de ligne à sauter en cas d'erreur.

Si ce programme n'était employé qu'une fois les lignes 30 et 40 seraient superflues. Mais dans le cas contraire, le fichier d'adresses, existant déjà, doit être agrandi d'où la ligne 30 qui ouvre le fichier en mode agrandissement (FOR APPEND). Si le fichier n'existe pas encore, la ligne 30 va provoquer le type d'erreur 53 c'est-à-dire FILE NOT FOUND. Mais grâce à l'instruction ON ERROR GOTO de la ligne 20, l'erreur au lieu d'être affichée va provoquer un saut à la ligne 190.

La ligne 190 va maintenant vérifier si le type d'erreur est bien 53 et qu'elle se produit à la ligne 30 et dans ce cas provoque un RESUME 50 c'est-à-dire une reprise après erreur à la ligne 50 où le fichier sera ouvert en mode OUTPUT autrement dit il sera créé.

Les lignes 60 à 120 permettent d'introduire les données relatives à une personne. La ligne 130 va sauver le contenu des variables N$, P$, R$ et NO$ sur le disque avec une marque de séparation après chacune d'entre elles (« , ») ; elle va aussi sauver le contenu de la variable CP, qui ne nécessite pas de séparateurs, étant numérique, et la variable V$ qui n'a pas besoin non plus de séparateur puisqu'elle est la dernière du PRINT #1 et que dès lors elle est suivie de CR-LF qui fera office de séparateur lors de la lecture.

Les lignes 140 à 170 vous demandent s'il y a encore des adresses à introduire auquel cas on repart à la ligne 60, ou si l'encodage est terminé on saute à la ligne 180. Celle-ci ferme le fichier, efface l'écran et arrête le programme.

Les lignes 200 à 240 offrent une petite touche de « professionnalisme » dans ce programme puisque la ligne 200 affiche un message en cas de disque protégé contre l'écriture et la ligne 210 en cas de lecteur vide et provoque le réessai de l'instruction qui a causé l'erreur après votre feu vert.

3.3.5.2 Consultation d'un fichier d'adresses

```
10 CLEAR500:MAXFILES=1
20 ON ERROR GOTO 170
30 OPEN « ADRESSE.TXT »FOR INPUT AS #1
40 CLS
50 INPUT « DONNEZ LE NOM DE RECHERCHE »;NR$
60 INPUT « DONNEZ LE PRENOM »;PR$
70 INPUT#1,N$,P$,R$,NO$,CP,V$
80 IF N$<>NR$ OR P$<>PR$ GOTO 70
90 PRINT
100 PRINT P$ ; «  »;N$
110 PRINT NO$ ; «  »;R$
120 PRINT CP;SPC(5);V$
130 PRINT:PRINT
140 INPUT « AUTRE RECHERCHE O/N »;A$:A$=CHR$(ASC(A$) AND 95)
150 CLOSE #1
160 IF A$= « O » GOTO 30 ELSE END
170 IF ERR=55 AND ERL=70 THEN PRINT « PAS TROUVE »:RESUME 130
180 IF ERR=53 THEN PRINT « ADRESSE.TXT N'EST PAS DANS LE DISQUE »:GOTO 210
190 IF ERR=70 THEN PRINT « PAS DE DISQUETTE DANS LE LECTEUR »:GOTO 210
200 ON ERROR GOTO 0
210 INPUT « FAITES RETURN LORSQUE LE PROBLEME SERA CORRIGE »;A$
220 RESUME
```

Les lignes 10 et 20 sont les mêmes que celles de l'exemple précédent. La ligne 30 ouvre le fichier ADRESSE.TXT en INPUT sous le numéro 1. les lignes 40 à 60 demandent le nom et le prénom de la personne dont on cherche l'adresse.

La ligne 70 lit du fichier les données de la première personne et la ligne 80 compare le nom et le prénom de la personne recherchée avec ceux lus du fichier. S'ils ne sont pas égaux, on retourne à la ligne 70 et le processus de lecture et de comparaison se répète.

Quand la comparaison est établie, les lignes 90 à 130 affichent les données du fichier pour cette personne. La ligne 140 vous demande alors si vous désirez rechercher l'adresse d'une autre personne et convertir votre réponse en majuscule.

La ligne 150 ferme le fichier de telle sorte qu'on puisse le rouvrir si une autre personne est demandée. Ceci est impératif puisque c'est le seul moyen pour pouvoir relire le fichier depuis le début. Si le fichier était simplement laissé ouvert, une autre recherche ne serait possible qu'à partir de l'endroit du fichier où l'on était arrivé ; or le nom recherché se trouve peut-être avant.

La ligne 160 teste votre réponse et provoque un saut à la ligne 30 si la réponse est « oui »ou termine le programme si la réponse est différente de « oui ».

La ligne 170 est atteinte en cas d'erreur. S'il s'agit de l'erreur 55 (Input past end), c'est que le nom recherché n'a pas pu être trouvé avant la fin du fichier. Un message approprié est alors produit et le programme reprend en ligne 130. Les lignes 180 à 200 affichent un message s'il n'y a pas de disque dans le lecteur ou s'il n'y a pas de fichier ADRESSE.TXT dans la disquette.

3.3.6 LINE INPUT#

LINEINPUT#<numéro de fichier>,<variable chaîne>

Lit à partir du fichier ouvert sous le <numéro fichier> une ligne complète sans tenir compte des délimiteurs conventionnels comme l'espace ou la virgule. Les données lues sont affectées à la variable <variable chaîne>. Celle-ci doit obligatoirement être du type chaîne.

Les fichier doit avoir été ouvert en mode INPUT. Cette instruction lit tous les caractères (y compris l'espace, la virgule et les guillemets) en provenance du fichier jusqu'à ce qu'un code CR (carriage return) soit rencontré ou jusqu'à ce que la variable soit remplie avec 255 caractères. La séquence CR-LF est ensuite sautée permettant ainsi au prochain LINEINPUT# de lire tous les caractères de la ligne suivante du fichier.

Cette instruction permet donc de visualiser le contenu réel d'un fichier séquentiel ou ASCII comme dans le programme qui sert d'exemple ci-dessous.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

```
10 CLEAR 512 :MAXFILES=1:CLS
20 ON ERROR GOTO 60
30 OPEN « ADRESSE.TXT »FOR INPUT AS #1
40 LINEINPUT#1,A$
```

```
50 PRINT A$:GOTO40
60 IF ERR=55 THEN CLOSE#1:END
70 ON ERROR GOTO 0
```

Ce programme affiche le contenu réel du fichier ADRESSE.TXT.

3.4 Les fonctions spécifiques aux fichiers séquentiels

3.4.1 INPUT$

INPUT$ (<nbr caractères>,[#]<numéro fichier>)

Cette fonction permet de lire du fichier portant le <numéro fichier> un certain nombre de caractères, défini par le paramètre <nbr caractères>. Tous ces caractères (y compris les délimiteurs) sont passés tels quel sauf le code 1AH, qui est la marque de fin de fichier.

C'est donc la seule fonction ou instruction qui permet de lire n'importe quel type de donnée qui se trouve dans un fichier séquentiel.

Exemple : Programme affichant le contenu réel d'un fichier séquentiel. Les codes CR et LF sont visualisés à l'écran entre corchets. La marque de fin de fichier sera affichée par [Fin de fichier]. Remarquez qu'il n'y a pas de CLOSE. En effet, l'instruction END inclut un CLOSE de tous les fichiers ouvert.

```
10 MAXFILES=1
20 CLS
30 OPEN « ADRESSE.TXT » FOR INPUT AS #1
40 A$=INPUT$(1,#1)
50 IF A$>CHR$(31) THEN PRINT A$;:GOTO 40
60 IF A$=CHR$(13) THEN PRINT [CR]:GOTO 40
70 IF A$=CHR$(10) THEN PRINT [LF]:GOTO 40
80 IF A$=CHR$(01)THEN PRINT A$;:GOTO 40
90 IF EOF(1)=-1 THEN PRINT « [FIN DE FICHIER] »:END
100 GOTO 40
```

3.4.2 End of file

EOF (<numéro fichier>)

EOF est l'abréviation de End Of File (fin de fichier). Cette fonction retourne -1 (vrai) si la fin d'un fichier séquentiel a été atteinte. Utilisez cette fonction avant chaque INPUT afin de prévenir l'erreur « Input past end ».

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction

OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Exemple : Liste le contenu d'un fichier en hexadécimal

```
10 MAXFILES=1
20 CLS:WIDTH 39
30 OPEN « ADRESSE.TXT » FOR INPUT AS #1
40 IF EOF(1)=-1 GOTO 80
50 A$=INPUT$(1,#1)
60 PRINT RIGHT$(« 0 »+HEX$(ASC(A$)),2) ; «  » ;
70 GOTO 40
80 PRINT
90 END
```

L'instruction de la ligne 40 peut être abrégée en

```
40 IF EOF(1) GOTO 80
```

3.4.3 LOCate

LOC(<numéro fichier>)

Cette fonction retourne la localisation du pointeur du fichier séquentiel. En d'autres mots, elle indique combien de blocs de 256 caractères ont déjà été lus de (ou écrit sur) le fichier <numéro fichier>.

Tous les délimiteurs y compris les codes CR et LF sont compris de même, bien sûr, que vos données. Si aucun bloc n'a encore été lu, la fonction retourne la valeur 1 car, lors de l'OPEN d'un fichier séquentiel en INPUT, le tampon mémoire réservé à un bloc est déjà rempli avec les 256 premiers caractères lus de votre fichier. Par contre en écriture, la fonction retourne bien 0 si vous n'avez pas encore écrit 256 caractères dans le fichier.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Exemple : ce programme va remplir 6 blocs de 256 caractères avec les données spécifiées dans le but de créer un petit fichier séquentiel d'essai.

```
10 MAXFILES=1
20 A$= « FICHIER DE TEST »
30 A%=147
40 A!=123456
50 A#=12345678901234
60 PRINT#1,A$ ; « , »;A%;A!;A#
70 IF LOC(1)<6 GOTO 60
80 CLOSE #1
```

```
90 END
```

3.4.4 LOF Length Of File

LOF (<numéro fichier>)

La fonction LOF retourne la longueur en octets du fichier ouvert sous le <numéro fichier>.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Exemple : ce programme vous donne la longueur du fichier posé en réponse à la question du programme.

```
10 CLS:MAXFILES=1
20 INPUT « DONNEZ LE NOM D'UN FICHIER »;F$
30 OPEN F$ FOR INPUT AS 1
40 PRINT
50 PRINT F$ ; « a une taille de »;LOF(1) ; « octets. »
60 PRINT:END
```

3.5 La manipulation d'un fichier à accès direct

Le fichier à accès direct est moins simple à mettre en œuvre que le fichier séquentiel, comme nous l'avons indiqué au chapitre 2.2.3. Contrairement au fichier séquentiel, les informations ne soient pas mises bout à bout mais organisées en enregistrements de longueur définie d'avance.

La définition de la longueur de l'enregistrement se fait à l'OPEN du fichier : elle ne pourra donc pas être changée durant la manipulation du fichier. De même, lorsqu'un fichier a été créé avec une longueur disons de 200 octets, tout OPEN futur de ce fichier devra se faire en précisant la même longueur.

L'inconvénient principal du fichier à accès direct (Random File) est qu'il faut prévoir une longueur d'enregistrement suffisante pour que toute information qui doit être mémorisée dans l'enregistrement y trouve place. Or, il est parfois difficile de prévoir d'avance quelle grandeur aura la plus grande information. Il en résulte aussi que des informations de petite taille vont être stockées dans des enregistrements de grande taille, gaspillant ainsi de l'espace disque inutilement.

Par contre, l'énorme avantage réside dans la vitesse de recherche d'un enregistrement et par le fait que l'accès se fait par numéro d'enregistrement et qu'il n'y a pas d'obligation qu'un fichier contienne tous les numéros d'enregistrements entre le premier et le dernier. Les classement et tris des données sont aussi grandement facilités par le découpage du fichier.

L’exemple idéal d’emploi d’un fichier à accès direct pourrait être un fichier de stock de marchandises ou de pièces dans un garage ou un magasin. Ainsi, dans un garage, les diverses pièces détachées portent généralement un numéro. On pourrait se servir de ce numéro comme numéro d’enregistrement ; ainsi chaque enregistrement de pièces pourrait nous donner la description de la pièce, le type de voiture auquel elle est destinée, son prix, la quantité de pièces en stock, le stock minimum nécessaire, le fournisseur de la pièce, etc.

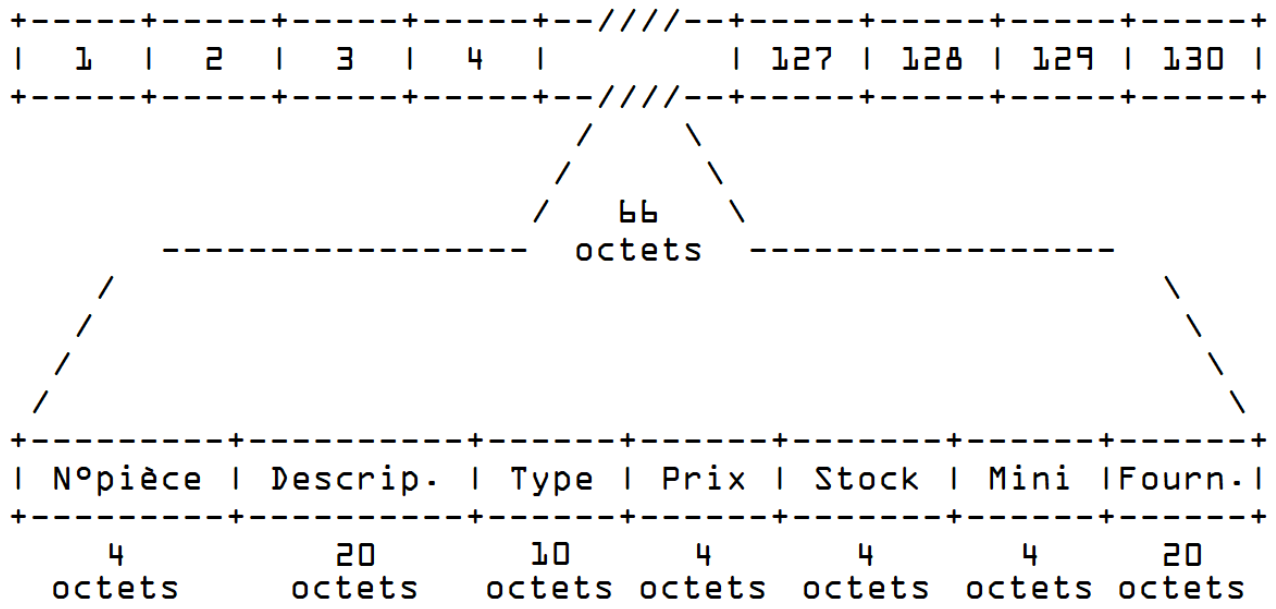
De par cette description, vous constatez qu’il est souhaitable de diviser un enregistrement en diverses zones pour y stocker les informations qui viennent d’être énumérées. C’est d’ailleurs en additionnant la longueur de ces différentes zones que la longueur de l’enregistrement sera déterminée.

Une instruction spéciale du BASIC (FIELD) est destinée à diviser un enregistrement en zones dont on peut fixer la longueur et la place dans l’enregistrement.

D’autre part, dans le but de condenser l’information au maximum dans l’enregistrement, les variables numériques seront sauvegardées dans le même format qu’elles ont lorsqu’elles résident en mémoire (format décimal compacté à virgule flottante – Floating pont packed decimal format). A cet effet, deux fonctions spéciales seront fournies par le BASIC qui permettront, l’une de déposer la variable dans l’enregistrement et l’autre de réattribuer une valeur numérique lue de l’enregistrement à une variable (MKI/S/D et CVI/S/D).

A part cela, l’écriture se fait par une instruction différente (PUT) de même que la lecture (GET). Il est aussi obligatoire de déposer les variables dans le tampon de l’enregistrement par une instruction spéciale (LSET – RSET). Le fichier à accès direct doit aussi être ouvert avant d’être manipulé et être refermé lorsque le travail est terminé.

FICHIER DE 130 ENREGISTREMENTS



3.5.1 L'ouverture d'un fichier à accès direct

OPEN <spéc. fichier> AS [#]<numéro fichier> [LEN=<longueur>]

L'instruction OPEN ouvre le fichier désigné par <spéc. fichier> en mode « accès direct » et fixe que le fichier sera dorénavant référencé par son <numéro fichier> plutôt que par son nom. La longueur de l'enregistrement peut optionnellement être donnée par <longueur>.

Si OPEN ouvre un fichier qui n'existe pas encore sur le disque, ce fichier sera créé mais ne contiendra encore rien.

Un fichier à accès direct peut être ouvert par plusieurs OPEN sous des numéros différents au même moment.

<spéc. fichier> est une chaîne ou une variable chaîne indiquant optionnellement le nom de l'unité de disquettes et impérativement le nom du fichier et son éventuelle extension. On ne peut pas placer de caractère de substitution dans le nom et l'extension du fichier. Si le nom de l'unité n'est pas donné, le fichier sera ouvert sur le lecteur courant.

Le paramètre <numéro fichier> doit être un numéro (ou une variable le contenant) de 1 à la valeur spécifiée dans l'instruction MAXFILES avec de toute façon un maximum de 6 (voir chapitre 3.1.11).

Le paramètre <longueur> est une constante, un variable ou leurs expressions numériques indiquant la longueur des enregistrements de ce fichier. Si ce paramètre n'est pas fourni, la valeur par défaut est de 256 octets. La longueur minimale autorisée est 1.

Un fichier ouvert en mode « accès direct » peut être lu par l'instruction GET et écrit par l'instruction PUT tant qu'il reste ouvert.

Exemples de syntaxe :

10 OPEN « A:STOCK.DIR » AS #1 LEN=66
10 OPEN « STOCK.DIR » AS #1 LEN=66
10 OPEN « BIBILIOTH.RND » AS 1

10 A$= « B:STOCK.DIR »:I=2:E=56
20 OPEN A$ AS I LEN=E
30 OPEN A$ AS I+1 LEN=E

3.5.2 FIELD : création de champs (zones)

FIELD [#]<numéro fichier>,<longueur>AS<variable chaîne>[,…]

FIELD alloue une place et une longueur dans le tampon mémoire réservé aux enregistrements du fichier à accès direct <numéro fichier> pour des variables de type chaîne de caractères. Avant qu'une lecture (GET) ou une écriture (PUT) d'un enregistrement ne puisse être exécutée, il faut découper le tampon mémoire du fichier avec l'instruction FIELD.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

<longueur> est une constante, une variable ou leurs expressions numériques déterminant la longueur de la zone réservée pour la <variable chaîne> dans le tampon mémoire de l'enregistrement.

On peut répéter <longueur>AS<variable chaîne> autant de fois que le permet la longueur maximum de la ligne Basic (255 caractères maximum) afin de partager l'enregistrement en plusieurs zones réservées respectivement à chaque <variable chaîne>. Mais attention que toute instruction FIELD commence le découpage par le début du tampon mémoire.

Exemple :

FIELD #1, 30 AS N$, 15 AS P$, 30 AS R$, 4 AS NO$, 20 AS V$

Cette instruction découpe le tampon mémoire des enregistrements du fichier numéro 1 en :

30 positions réservées pour la variable N$ (N$ = NOM)
15 positions réservées pour la variable P$ (P$ = PRÉNOM)
30 positions réservées pour la variable R$ (R$ = RUE)
4 positions réservées pour la variable NO$ (NO$ = NUMÉRO)
20 positions réservées pour la variable V$ (V$ = VILLE)
–
99 positions pour l'enregistrement.

Une instruction FIELD ne peut allouer un espace total plus grand que la longueur de l'enregistrement définie dans l'instruction OPEN. Dans notre exemple, le total de l'espace alloué

est de 99 ; donc l'OPEN devra avoir été choisi avec une longueur d'enregistrement supérieure ou égale à 99. Sans quoi le message d'erreur 'Field overflow' sera produit (dépassement de capacité dans FIELD).

Attention !! L'instruction FIELD ne place aucune donnée dans le tampon du fichier. Ce sera le rôle des instructions RSET-LSET.

Il faut noter aussi que l'espace réservé pour stocker des valeurs numériques doit être de 2 octets pour les valeurs entières, de 4 octets pour les valeurs simple précision et de 8 octets pour les valeurs double précision. Cet espace NE DOIT PAS être alloué à une variable NUMERIQUE mais au contraire à une variable CHAINE. Des fonctions spéciales sont fournies pour transposer une variable numérique dans la variable chaîne correspondante (MKI MKS-MKD).

On peut découper un même enregistrement suivant plusieurs formats différents par plusieurs instructions FIELD. Ces différents formats seront retenus par l'ordinateur pour autant qu'aucune <variable chaîne> ne porte le même nom qu'une autre et que chaque découpage traite la longueur complète de l'enregistrement.

Exemple :

```
10 OPEN « ADRESSE.TXT » AS 1 LEN=55
20 FIELD #1,11 AS P$, 11 AS N$, 11 AS NC$, 11 AS RN$, 11 AS V$
30 FIELD #1,11 AS X$,22 AS NG$, 11 AS X$, 11 AS X$
40 FIELD #1,33 AS X$, 22 AS AC$
```

P$ = PRENOM
N$ = NOM
NC$ = NOM DU CONJOINT
RN$ = RUE ET No
V$ = VILLE
NG$ = NOM GLOBAL (c'est-à-dire NOM + NOM DU CONJOINT)
AC$ = ADRESSE COMPLETE (c'est-à-dire RUE + No + VILLE)
X$ = Variable bidon

Supposons que le tampon contienne l'enregistrement suivant :

```
JEAN-MARIE DUPOND      DURAND       DE PARIS, 7MARSEILLE
|----------|----------|-----------|----------|-----------|
```

Dans ce cas, les variables suivantes contiendront :

P$ = « JEAN-MARIE »
N$ = « DUPOND »
NC$ = « DURAND »
RN$ = « DE PARIS, 7 »
V$ = « MARSEILLE »
NG$ = « DUPOND DURAND »
AC$ = « DE PARIS, 7MARSEILLE »
X$ = « JEAN-MARIE DUPOND DURAND »

N’utilisez jamais une variable entrant dans la composition d’une instruction FIELD avec un INPUT ou un LET (=). Vous ne pourriez plus obtenir les données correspondantes du tampon dans l’enregistrement.

EXEMPLE 1

```
10 OPEN « ADRESSE.TXT » AS #1 LEN=99
20 FIELD #1,30 AS N$, 15 AS P$, 30 AS R$, 4 AS NO$, 20 AS V$
30 FOR I=2 TO 10
40 GET#1,I
50 PRINT N$:PRINT P$;PRINT R$:PRINT NO$:PRINT V$
60 NEXT
70 END
```

Dans l’exemple 1, un seul découpage de l’enregistrement a été réalisé en ligne 20. La boucle de 30 à 60 lit et affiche à l’écran le contenu des enregistrements 2 à 10 du fichier « ADRESSE.TXT ».

EXEMPLE 2

```
10 OPEN « ADRESSE.TXT » AS #1 LEN=99
20 FIELD #1,2 AS NE$, 97 AS NUL$
30 FIELD #1,30 AS N$, 15 AS P$, 30 AS R$, 4 AS NO$, 20 AS V$
40 GET #1,1
50 T%=CVI(NE$)
60 FOR I =2 TO T %
70 GET#1,I
80 PRINT N$:PRINT P$;PRINT R$:PRINT NO$:PRINT V$
90 NEXT
100 END
```

Dans cet exempls, deux découpages ont été réalisés : le premier (ligne 20) sert à donner le format de l’enregistrement numéro 1 dans lequel on a décidé de placer le numéro du dernier enregistrement du fichier (variable NE$). Les 97 octets réservés à la variable NUL$ font en sorte que l’entièreté de la longueur de l’enregistrement soit réservé par cette instruction FIELD. Le deuxième découpage (ligne 30) sert à fixer le découpage de tous les enregistrements à l’exception du premier.

Attention de bien veiller à l’emploi des variables ! En effet, lors de la lecture du record 1 (GET#1,1), la variable N$ contiendra elle aussi les 15 premiers caractères du record 1 mais ce contenu ne correspondra pas au nom d’une personne comme lors de la lecture des autres records. De même NE$ ne contiendra le numéro de dernier enregistrement que lors de la lecture du premier enregistrement et non lors de la lecture des suivants.

EXEMPLE 3

Supposons que vous désirez sauver une table de 16 noms de maximum 16 caractères chacun dans un enregistrement du fichier « FICHIER.TAB ». Vous auriez pu écrire le programme suivant :

```
10 DIM TB$(15)
20 OPEN « FICHIER.TAB »AS 1
30 FIELD #1,16 AS TB$(0), 16 AS TB$(1), 16 AS TB$(2), 16 AS TB$(3), 16 AS TB$(4), 16 AS
TB$(5), 16 AS TB$(6), 16 AS TB$(7), 16 AS TB$(8), 16 AS TB$(9), 16 AS TB$(10), 16 AS
TB$(11), 16 AS TB$(12), 16 AS TB$(13), 16 AS TB$(14), 16 AS TB$(15)
```

Avouez qu'il est nettement plus simple d'écrire le programme avec une petite boucle plutôt que de devoir taper cette très longue instruction comme vous le montre l'exemple ci-dessous.

```
10 DIM TB$(15)
20 OPEN « FICHIER.TAB »AS #1
30 FOR N=0 TO 15
40 FIELD #1, N*16 AS NUL$, 16 AS TB$(N)
50 NEXT N
```

Remarquez alors que l'expression N*16 AS NUL$ sert à attribuer une longueur à une variable qui ne sert à rien (NUL$) mais qui correspond au décalage de la variable TB$(N) par rapport au début de l'enregistrement. Rappelez-vous, en effet, que l'instruction FIELD commence toujours le découpage au début du tampon mémoire.

3.5.3 LSET (LEFT SET – PLACER A GAUCHE) RSET (RIGHT SET – PLACER A DROITE)

```
LSET <X$>=<Y$>
RSET <X$>=<Y$>
```

Ces deux instructions servent à transférer le contenu de <Y$> vers la variable <X$> en alignant les données à gauche pour LSET ou à droite pour RSET. Habituellement, elles sont employées pour transférer <Y$> vers la partie du tampon d'un fichier à accès direct qu'une instruction FIELD préalable aurait nommé <X$>.

<Y$> est une constante ou une variable, de type chaîne qui est l'information que l'on veut placer dans le tampon mémoire de l'enregistrement dans le but de l'écrire dans le fichier.

<X$> est une variable de type chaîne qui doit avoir été définie par une instruction FIELD préalable.

Exemple :

```
10 OPEN « ADRESSE.TXT »AS #1 LEN=92
20 FIELD #1,30 AS N$, 12 AS P$, 30 AS R$, 20 AS V$
30 LSET N$= « DURAND »
40 PR$= « JEAN »:LSET P$=PR$
50 LSET R$= « AVENUE DE LA LIBERATION,30 »
60 LSET V$= « PARIS »
70 PUT#1,1
```

L’instruction FIELD de la ligne 20 a prévu 30 caractères pour le nom (N$), 12 caractères pour le prénom (P$) , 30 pour la rue (R$) et finalement 20 pour la ville (V$). Cela signifie donc que cette instruction a réservé 30 caractères dans le tampon mémoire du fichier ADRESSE.TXT pour le nom. Que se passe-t-il si le nom à sauver dans le fichier est plus petit ou plus grand que 30 caractères ?

1) Si la longueur de la chaîne est plus petite, le reste de la zone réservée dans le tampon mémoire sera comblé avec des espaces (code 32).

2) Si la longueur de la chaîne à sauver est plus grande, cette chaîne sera tronquée au-delà du 30ème caractère.

Exemple :

zone de 12 caractères réservés à P$:

```
LSET P$="JEAN"                    JEAN--------
RSET P$="JEAN"                    --------JEAN
LSET P$="NABUCHODONOZOR"          NABUCHODONOZ
RSET P$="NABUCHODONOZOR"          NABUCHODONOZ
```

Les tirets représentent les espaces. Remarquez la position de JEAN dans la zone dépendant de LSET ou RSET. Remarques également que le nom trop grand est toujours tronqué à droite tant avec LSET qu’avec RSET.

On peut également employer les instructions LSET et RSET avec des variables normales non allouées à un tampon de fichier à accès direct. Il est impératif dans ce cas de s’assurer que la variable de réception <X$> contienne déjà une chaîne de caractères afin de lui attribuer une longueur.

Exemple :

```
10 A$= « TELEPHONE »
20 B$= « JEAN-MARIE »
30 C$= « RENE »
40 LSET A$ = C$
50 RSET B$ = C$
60 PRINT A$ ; « . »
70 PRINT B$ ; « . »
80 END

RUN
RENE
RENE.
OK
```

3.5.4 MAKE STRINGS

MKI$[<X$>]
MKS$[<X$>]
MKD$[<X$>]

Les valeurs numériques sous formes de constantes ou de variables ne peuvent pas être placées telles quelles dans le tampon mémoire d'un fichier à accès direct car celui-ci n'accepte que des chaînes de caractères. Les fonctions ci-dessus convertissent une valeur numérique entière (MKI$), simple précision (MKS0$) ou double précision (MKD$) en une chaîne de caractères prête à être placée par LSET ou RSET dans le tampon mémoire d'un fichier à accès direct. Cette chaîne est l'exacte représentation de la forme sous laquelle cette valeur numérique reste en mémoire.

Supposons que nous ayons à sauver la valeur de Pi dans un fichier à accès direct. Nous pourrions résoudre le problème comme suit :

```
10 OPEN « TEST » AS #1 LEN=17
20 PI=ATN(1)*4
30 PI$=STR$(PI)
40 FIELD #1,17 AS N$
50 LSET N$=PI$
60 PUT#1,1
70 CLOSE
80 END
```

Le procédé que nous venons d'employer est parfaitement autorisé mais son inconvénient va bientôt nous sauter aux yeux. Il faut, en effet, 17 caractères dans le tampon pour sauver un nombre en double précision (n'oubliez pas le symbole du signe [espace ou] de ce nombre et l'espace final) !

Or nous savons qu'une valeur en double précision est mémorisée en RAM en huit octets (4 octets pour les valeurs simple précision et 2 octets pour les valeurs entières). Ne pourrait-on dès lors sauver la valeur de Pi sous le même format ? C'est exactement ce genre de conversion que réalise la fonction MKD$.

```
10 OPEN « TEST » AS #1 LEN=17
20 PI=ATN(1)*4
30 FIELD #1,8 AS N$
40 LSET N$=MKD$(PI)
50 PUT#1,1
60 CLOSE
70 END
```

Grâce à cette fonction, nous avons épargné l'instruction de la ligne 30 du premier programme mais surtout nous avons réussi à sauver dans le fichier disque une caleur numérique de 17 symboles typographiques en seulement 8 octets. Il devient dès lors possible de placer dans un enregistrement de 256 octets, 32 valeurs numériques double précision.

Pour rappel, voici le format sous lesquels les différentes valeurs numériques sont stockées :

1) Valeur numérique entière

```
2⁷ 2⁶ 2⁵ 2⁴ 2³ 2² 2¹ 2⁰          S 2¹⁴ 2¹³ 2¹² 2¹¹ 2¹⁰ 2⁹ 2⁸
```

1er octet 2ème octet

La notation se fait en binaire sur 15 bits. Le seizième bit est le signe du nombre (S=1 pour négatif, S=0 pour positif). Attention que le premier octet contient les bits les moins significatifs et le deuxième les bits les plus significatifs.

```
-8200 = 0 0 0 0 1 0 0 0       1 0 1 0 0 0 0 0
      =     &H08                  &HA0
      =     1er octet             2ème octet
```

2) Valeur numérique simple précision

Les quatre octets d'une valeur numérique en simple précision forment une série de 32 bits dont chacun permet l'encodage d'un chiffre de 0 à 9. Le chiffre le plus significatif se trouve dans les 4 bits supérieurs de l'octet numéro 2 et le chiffre le moins significatif est présent dans les 4 bits inférieurs du dernier octet. Exemple : -736248 = - 0,776248 * 10

Exemple 1 :

```
1   1  000110 0011 0110 0010 0100 1000
|   |  ------
|   |  exp 6   7    6    2    4    8
|   |
|   +- signe exposant positif
|
+----- signe du nombre négatif
```

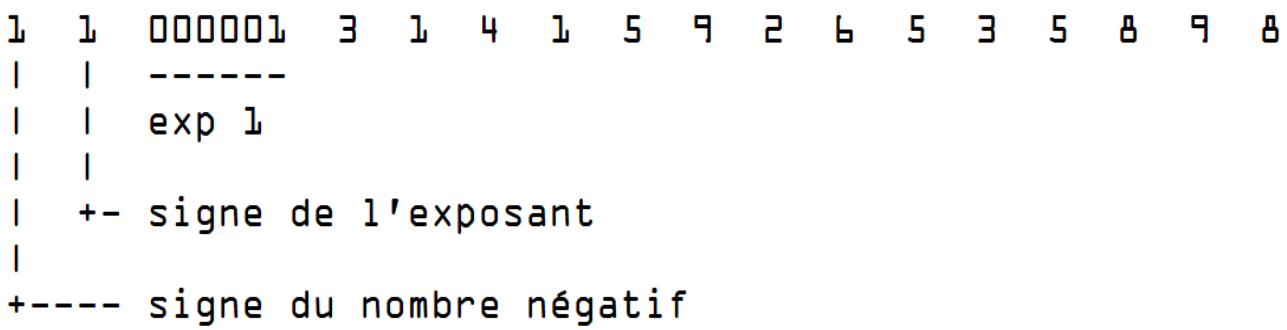
Exemple 2 : +0,004215 = 0,421500 * 10 -2

```
0  0  111110 0100 0010 0001 0101 0000 0000
|  |  ------
|  |  exp 62   4    2    1    5    0    0
|  |
|  +- signe exposant négatif donc l'exposant
|     réel est 64-62=2 : 64-62 =2 et donc -2
|
+---- signe du nombre positif
```

3) Valeur numérique double précision

Le format est identique à celui en simple précision excepté qu'il y a 14 groupe de 4 bits pour exprimer les 14 chiffres maximum que peut contenir une valeur double précision. Le premier octet contient donc les signes et l'exposant et les 7 octets suivants forment les 14 groupes de 4 bits.

Exemple : -3,1415926535898 = -0,31415926535898 * 10



Voici, pour terminer, un petit programme qui vous montrera le contenu d'une variable en double précision sous la forme qu'elle occupe en mémoire ou dans un fichier à accès direct. Les huit octets seront formulés en hexadécimal.

```
10 INPUT « ENTREZ UN NOMBRE EN DOUBLE PRECISION »;A#
20 PRINT « CE NOMBRE RESIDE EN MEMOIRE COMME SUIT »
30 A$=MKD$(A#)
40 FOR I = 1 TO 8
50 S = ASC(MID$(A$,I,1))
60 PRINT RIGHT$(« 0 »+HEX$(S),2); « - » ;
70 NEXT
80 PRINT:PRINT:GOTO 10
```

3.5.5 PUT

PUT [#]<numéro fichier> [,<X>]

L'instruction PUT écrit le tampoin mémoire, réservé au fichier à accès direct et rempli par les instructions LSET-RSET précédentes, dans l'enregistrement numéro <X> du fichier <numéro fichier>.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

<X> est une constante, une variable ou leurs expressions numériques indiquant quel enregistrement du fichier doit être écrit. <X> ne peut être inférieur à 1, ni supérieur à 4 294 967 295. Ce nombre est, bien entendu, théorique car il dépend en fait du nombre d'octets libres sur votre disquette. Appliquez la formule suivante pour connaître le nombre maximum approximatif :

<X>max = Octets libres du disque \ longueur enregistrement

Lorsque <X> est omis, l'écriture se fera sur l'enregistrement qui suit le dernier enregistrement manipulé par GET ou PUT pour ce fichier, ou alors sur le premier enregistrement si aucun GET ou PUT n'a encore été utilisé.

LE tampon mémoire doit avoir été rempli préalablement à l'instruction PUT par les instructions LSET ou RSET sans quoi le contenu du tampon mémoire et partant de l'enregistrement sera indéterminé.

Ne vous effrayez pas si le disque ne tourne pas au premier PUT que vous effectuerez. En effet, le tampon mémoire du fichier est d'abord placé dans le tampon SECTEUR avant d'être écrit sur le disque.

Exemple :

```
5 MAXFILES=1 : I=1
10 OPEN « EXEMPLE.TST » AS #1 LEN=29
20 FIELD #1, 2 AS LR$, 27 AS NUL$
20 FIELD #1, 15 AS Z1$, 2 AS Z2$, 4 AS Z3$, 8 AS Z4$
30 INPUT « NOM »;N$
35 IF N$= « ***** » GOTO 130
40 INPUT « AGE »;A %
50 INPUT « SALAIRE »;S !
60 INPUT « N° TEL »;T#
70 LSET Z1$=N$
80 LSET Z2$=MKI$(A%)
90 LSET Z3$=MKS$(S!)
100 LSET Z4$=MKD$(T#)
110 I = I+1:PUT #1, I
120 GOTO 30
130 LSET LR$=MKI$(I)
140 PUT #1, 1
150 END
```

Ce programme vous permet de créer et de remplir un petit fichier contenant dans chaque enregistrement le nom d'une personne, son âge, son salaire et son numéro de téléphone. Lorsque vous désirez arrêter l'encodage du fichier, tapez comme nom 5 astérisques (*****) ; le fichier sera alors fermé et on reviendra à l'indicatif du BASIC. L'avantage de ce petit exemple est qu'il montre la manipulation de tous les types de variables (chaîne, entier, simple et double précision). Remarquez aussi que le premier enregistrement est réservé pour indiquer combien d'enregistrements ont pris place dans le fichier à partir du record numéro 2. En effet, au moment de l'arrêt de l'encodage, la variable I, indiquant le numéro du dernier enregistrement tapé, est sauvegardée dans le premier record dans la variable LR$ (Last Record).

3.5.6 GET

GET [#]<numéro fichier> [,<X>]

L’instruction GET lit l’enregistrement numéro <X< du fichier ouvert sous <numéro fichier> dans le tampon mémoire réservé à ce fichier. De ce fait, toutes les variables définies par l’instruction FIELD sont remplies avec les données en provenance de l’enregistrement.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l’instruction. C’est l’instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l’instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

<X> est une constante, une variable ou leurs expressions numériques indiquant quel enregistrement du fichier doit être lu. <X> ne peut être inférieur à 1, ni supérieur à 4 294 967 295. Ce nombre est, bien entendu, théorique car il dépend en fait du nombre d’enregistrements qui ont déjà été placés dans le fichier par les instructions PUT.

Lorsque <X> est omis, la lecture se fera sur l’enregistrement qui suit le dernier enregistrement manipulé par GET ou PUT pour ce fichier, ou alors sur le premier enregistrement si aucun GET ou PUT n’a encore été utilisé.

Exemple : relire le fichier créé dans la rubrique PUT

```
 10 MAXFILES=1
20 OPEN « EXEMPLE.TST » AS #1 LEN=29
30 FIELD #1, 2 AS LR$, 27 AS NUL$
40 FIELD #1, 15 AS Z1$, 2 AS Z2$, 4 AS Z3$, 8 AS Z4$
50 GET #1,1 : LR=CVI(Z2$)
60 FOR I=2 TO LR
70 GET #1
80 PRINT « NOM : »;Z1$
90 PRINT « AGE : »;CVI(Z2$)
100 PRINT « SALAIRE : »;CVS(Z3$)
110 PRINT « N° TEL »;CVD(Z4$)
120 NEXT I
130 CLOSE
140 END
```

La ligne 50 lit le premier record pour obtenir dans la variable LR le nombre d’enregistrements de ce fichier. Les lignes 60 à 102 forment une boucle qui lit et affiche les records 2 à LR. Remarquez que le GET de la ligne 70 n’a donc pas besoin de <numéro d’enregistrement> car on lit les records en séquence.

On peut aussi obtenir le contenu du tampon mémoire par les instructions INPUT$ et LINE INPUT$, mais dans ce cas, le contenu du tampon mémoire doit contenir les marques de séparation reconnues par l’instruction INPUT$.

Exemple :

```
10 OPEN « TEST.TST » FOR OUTPUT AS #1
20 PRINT#1, « BONJOUR, » ;
```

```
30 A%=3:D!=123456 : C#=12345678901234
40 PRINT#1, A %, B !, C#
50 CLOSE
60 OPEN « TEST.TST » AS 1
70 GET#1,1
80 INPUT #1, A$;A%;A!;A#
90 PRINT A$;A%;A!;A#
100 CLOSE
110 END
```

3.5.7 Conversion d'une chaîne numérique en valeurs numériques

CVI(X$)	Convertit en entier (I=Integer)
CVS(X$)	Convertir en simple précision (S=Simple)
CVD(X$)	Convertit en double précision (D=Double)

Les fonctions CVI, CVS et CVD convertissent une variable chaîne extraite du tampon d'un fichier à accès direct en une valeur numérique entière (CVI), simple précision (CVS) ou double précision (CVD).

La chaîne <X$> doit avoir une longueur de 2 octets pour CVI, 4 octets pour CVS et 8 octets pour CVD. Plus généralement, on peut dire que ces fonctions convertissent des données numériques du format sous lequel elles résident en mémoire ou dans le tampon d'un fichier à accès direct en valeurs numériques.

Exemple :

```
10 A$ = MKI$(1024)
20 B$ = MKS$(123456)
30 C$=MKD$(12345678901234)
40 PRINT CVI(A$)
50 PRINT CVS(B$)
60 PRINT CVD(C$)
70 END
```

L'exemple ci-dessus montre bien que les fonctions CVI – CVS – CVD réalisent la fonction inverse de celle de MKI$ – MKS$ – MKD$ (voir le chapitre 3.5.4).

Exemple :

```
10 MAXFILES=1
20 OPEN « EXEMPLE.TST » AS #1 LEN=29
30 FIELD #1,15 AS W1$, 2 AS W2$, 4 AS W3$, 8 AS W4$
40 GET #1,1
50 PRINT « NOM : »;W1$
60 PRINT «AGE: »;CVI(W2$)
70 PRINT «SALAIRE: »;CVS(W3$)
80 PRINT «N° TEL: »;CVD(W4$)
```

90 CLOSE : END

3.5.8 Fermeture d'un fichier à accès direct

CLOSE [#]<numéro fichier> [,…]

L'instruction CLOSE conclut les opérations d'entrée-sortie sur un fichier.

<numéro fichier> est une constante numérique, une variable numérique ou une expression numérique qui spécifie sur quel fichier porte l'instruction. C'est l'instruction OPEN qui a associé un numéro au nom du fichier ; il faut donc reproduire ici le numéro indiqué dans l'instruction OPEN du fichier sur lequel on désire que cette instruction agisse. Ce numéro ne peut être inférieur à 1 ni supérieur au nombre de fichiers fixé par MAXFILES avec de toute façon un maximum de 6.

Une instruction CLOSE peut fermer plusieurs fichiers à la fois. Indiquez simplement les numéros de fichier désirés, séparés par une virgule. Si CLOSE ne contient aucun numéro de fichier, elle fermera tous les fichiers ouverts.

L'association entre un nom de fichier et son numéro se termine dès la fin du CLOSE. Le fichier peut dès lors être réouvert sous le même numéro ou sous un autre numéro. De même, le numéro de fichier pourra dorénavant servir pour ouvrir n'importe quel fichier.

L'instruction CLOSE est vitale si vous avez procédé à des écritures d'enregistrements car le tampon mémoire final ne sera écrit sur la disquette qu'à ce moment. En plus, la FAT et l'entrée du Directory correspondant à ce fichier seront écrites sur la disquette pour refléter un éventuel agrandissement de votre fichier et indique la date et l'heure de cette modification.

Les instructions END , CLEAR et NEW provoquent également la fermeture de tous les fichiers ouverts.

3.6 Programme exemple de fichier à accès direct

3.6.1 Création d'un fichier

Pour illustrer la création d'un fichier à accès direct, nous prendrons l'exemple du fichier d'adresses qui nous a servi à montrer la création d'un fichier séquentiel (voir chapitre 3.3.5.1) de telle sorte qu'une comparaison puisse mieux faire ressortir leurs avantages et inconvénients.

La première opération est de déterminer quelles informations nous voulons placer dans un enregistrement et la longueur de chacune de ces informations.

INFORMATION	LONGUEUR MAXIMUM	TYPE VARIABLE	
Nom de famille	16 caractères	chaîne	16 octets
Prénom	16 caractères	chaîne	16 octets
Rue	25 caractères	chaîne	25 octets

Numéro	6 caractères	chaîne	6 octets
Code postal	6 chiffres	simple	4 octets *
Ville	16 caractères	chaîne	16 octets
Téléphone	10 chiffres	double	8 octets *

(*) Pour stocker des valeurs numériques dans un fichier à accès direct, il faut convertir ces valeurs en chaîne de caractères par les instructions MKI$-MKS$-MKD$. Dès lors, nous devrons préalablement déterminer le type des valeurs numériques. Le code postal étant composé d'un maximum de 6 chiffres, nous choisirons le type simple précision et l'instruction MKS$ convertira la valeur en chaîne de 4 octets. Le numéro de téléphone étant supérieur à 6 chiffres, nous sommes tenus de choisir le type double précision et c'est l'instruction MKD$ qui convertira la valeur en une chaîne de 8 octets. Le numéro de l'adresse a été laissé en chaîne de caractères pour autoriser les numéros de rue avec lettre comme 306A.

Les longueurs de chacune des différentes zones de l'enregistrement étant maintenant déterminées, nous pouvons les additionner pour connaître la longueur totale de l'enregistrement, soit : 16+16+25+6+4+16+8 = 91. Ce total de 91 va servir de taille d'enregistrement lors de l'OPEN du fichier.

```
10 CLEAR 500 : MAXFILES=1 : I=1
20 ON ERROR GOTO 280
30 OPEN « ADRESSE.RND » AS #1 LEN=91
35 FIELD#1, 2 AS LR$, 89 AS NUL$
40 FIELD#1, 16 AS Z1$, 16 AS Z2$, 25 AS Z3$, 6 AS Z4$, 4 AS Z5$ , 16 AS Z6$, 8 AS Z7$
50 CLS
60 INPUT « NOM                »;N$
70 INPUT « PRENOM             »;P$
80 INPUT « RUE                »;R$
90 INPUT « NUMERO             »;NO$
100 INPUT « CODE POSTAL       »;CP
110 INPUT « LOCALITE          »;V$
120 INPUT « TELEPHONE         »;T
130 LSET Z1$ = N$
140 LSET Z2$ = P$
150 LSET Z3$ = R$
160 LSET Z4$ = NO$
170 LSET Z5$ = MKS(CP)
180 LSET Z6$ = V$
190 LSET Z7$ = MKD$(T)
200 I = I+1 : PUT#1, I
210 PRINT « RETURN POUR CONTINUER, ESC POUR ARRETER »
220 A$ = INKEY$ : IF A$= «» GOTO 220
230 IF A$=CHR$(13) GOTO 50
240 IF A$<>CHR$(27) GOTO 220
250 LSET LR$=CVI(I)
260 PUT#1, 1
```

```
270 CLOSE #1:CLS:END
280 IF ERR=68 GOTO 310
290 IF ERR=70 GOTO 320
300 ON ERROR GOTO 0
310 PRINT « Disque protégé contre l'écriture » : GOTO 330
320 PRINT « Pas de sique dans le lecteur »
330 INPUT « Return lorsque le problème sera corrigé »;A$
340 RESUME
```

Guère de particularités dans ce programme sauf que le nombre d'enregistrements est sauvé dans le record numéro 1 et nécessite dès lors un FIELD séparé (ligne 35) et son placement dans le fichier (ligne 250-260). Ce programme est nettement pus long que son équivalent à fichier séquentiel mais offrira par contre une beaucoup plus grande vitesse de recherche en lecture.

3.6.2 Lecture d'un fichier

Le très gros avantage du fichier à accès direct est précisément sa capacité à accéder directement à un enregistrement donné sans avoir à lire tous les précédents. On peut accéder à l'enregistrement 1232 ou 567 aussi rapidement qu'à l'enregistrement 1. Voyons d'abord un programme qui permet de relire au choix n'importe quel enregistrement créé par le programme précédent.

```
10 CLEAR 500:MAXFILES=1
20 ON ERROR GOTO 260
30 OPEN « ADRESSE.RND » AS #1 LEN=91
40 FIELD#1, 2 AS LR$, 89 AS NUL$
50 FIELD#1, 16 AS Z1$, 16 AS Z2$, 25 AS Z3$, 6 AS Z4$, 4 AS Z5$ , 16 AS Z6$, 8 AS Z7$
60 GET#1,1
70 LR=CVI(LR$)
80 CLS
90 PRINT « QUEL RECORD VOULEZ-VOUS (1 - »;LR ; « ) » ;
100 INPUT RN
110 IF RN>RL GOTO 80
120 GET#1,RN : PRINT
130 PRINT Z1$
140 PRINT Z2$
150 PRINT Z3$ ; « , »;Z4$
160 PRINT CVS(Z5$);SPC(5);Z6$
170 PRINT CVD(Z7$)
180 PRINT
190 PRINT « Return pour continuer ESC pour arrêter » ;
200 A$ = INKEY$ : IF A$= «» GOTO 200
210 IF A$=CHR$(13) GOTO 80
220 IF A$<>CHR$(27) GOTO 200
230 CLOSE
240 CLS
250 END
```

```
260 IF ERR=70 GOTO 280
270 ON ERROR GOTO 0
280 PRINT « Pas de disque dans le lecteur»
290 INPUT « Return quand problème corrigé »;A$
300 CLS : RESUME
```

Si vous avez encodé et essayé ce programme, vous trouverez certainement qu'il est beaucoup plus rapide que son équivalent en fichier séquentiel surtout s'il comporte beaucoup d'enregistrements. Mais vous direz aussi qu'il est plus facile de retrouver l'adresse de quelqu'un en donnant son nom plutôt qu'en donnant son numéro d'enregistrement qu'il est difficile de retenir pour chaque nom…

C'est ici qu'intervient ce qu'on appelle l'accès direct indexé. Si vous vouliez vraiment utiliser le programme précédent, vous seriez pratiquement tenu de vous constituer une liste séparée donnant le numéro d'enregistrement pour tous les noms sauvés dans le fichier. Ainsi, si vous voulez connaître rapidement l'adresse de M . Dupont, vous consulteriez d'abord votre liste à « Dupont » et utiliseriez le numéro d'enregistrement écrit en regard pour demander à l'ordinateur le reste des informations.

Mais pourquoi ne pas faire exécuter ce travail par l'ordinateur ? Deux possibilités s'offrent à nous.

On peut créer un tableau résidant en mémoire avec pour chaque nom le numéro d'enregistrement où l'ensemble des données résident. Ce tableau devrait être constitué au lancement du programme en lisant tous les enregistrements pour en extraire uniquement le nom et le sauver dans le tableau.

On pourrait également créer un deuxième fichier contenant le nom de la personne et le numéro de l'enregistrement du fichier principal où réside le reste des informations. Ce fichier étant très court, il ne pénalisera pas trop la recherche de l'information.

Le premier système n'est valable que si la possibilité existe de placer tous les noms en mémoire conjointement avec le programme utilisateur tandis que le second système, un peu plus lent il est vrai, convient pour de très gros fichiers. Nous nous pencherons sur le deuxième système dans le but d'exercer davantage notre apprentissage du Disk-Basic.

La méthode la plus rationnelle consiste à écrire un petit programme séparé qui va créer ou recréer le fichier Index. Ainsi, chaque fois que le fichier principal aura subi des modifications, il suffira de lancer ce petit programme pour réajuster le fichier Index.

```
10 CLEAR 500 : MAXFILES=2
20 OPEN « ADRESSE.RND » AS #1 LEN=91
30 OPEN « ADRESSE.INX » AS #2 LEN=16
40 FIELD #1, 2 AS LR$, 89 AS NUL$
50 FIELD #1, 16 AS Z1$, 75 AS NUL$
60 FIELD #2, 16 AS Y1$
70 GET#1,1
80 LR = CVI (LR$)
90 FOR I = 2 TO LR
100 GET #1, I
110 LSET Y1$=Z1$
120 PUT #2, I
130 NEXT
```

```
140 CLOSE
150 END
```

En fait ce programme ne fait que recopier dans le fichier ADRESSE.INX le nom présent au début de chaque enregistrement du fichier ADRESSE.RND sous le même numéro d'enregistrement.

Voyons maintenant la seconde mouture du programme de consultation qui cette fois-ci va nous permettre de rechercher l'adresse d'une personne par son nom plutôt que par son numéro.

```
10 CLEAR 500 : MAXFILES = 2
20 ON ERROR GOTO 290
30 OPEN « ADRESSE.RND »AS #1 LEN=91
40 OPEN « ADRESSE.INX » AS #2 LEN=16
50 FIELD #1, 2 AS LR$, 89 AS NUL$
60 FIELD #1, 16 AS Z1$, 16 AS Z2$, 25 AS Z3$, 6 AS Z4$, 4 AS Z5$ , 16 AS Z6$, 8 AS Z7$
70 FIELD #2, 16 AS Y1$
80 GET#1, 1
90 LR=CVI(LR$)
100 CLS
110 INPUT « QUEL NOM CHERCHEZ-VOUS »;N$
120 FOR I=2 TO LR
130 GET#2, I
140 IF Y1$=N$ GOTO 180
150 NEXT
160 PRINT « Désolé, ce nom n'existe pas dans le fichier »
170 GOTO 230
180 GET#1, I
190 PRINT : PRINT Z1$ : PRINT Z2$
200 PRINT Z3$ ; « , »;Z4$
210 PRINT CVS(Z5$);SPC(5);Z6$
220 PRINT CVD(Z7$)
230 PRINT
240 PRINT « Return pour continuer – ESC pour arrêter »
250 A$=INKEY$
260 IF A$=CHR$(13) GOTO 100
270 IF A$<>CHR$(27) GOTO 250
280 CLOSE:END
290 IF ERR=70 GOTO 310
300 ON ERROR GOTO 0
310 PRINT « Pas de disque dans le lecteur »
320 INPUT « Return quand problème corrigé »;A$
330 CLS : RESUME
```

En pratique la technique du fichier à accès direct indexé est un peu plus sophistiquée que dans cet exemple. On va, en effet, classer le s,noms du fichier Index par ordre alphabétique et la recherche ne se fera pas séquentiellement comme c'est le cas ici, mais par dichotomie pour réduire le nombre d'accès au minimum.

3.6.3 Mise à jour d'un fichier

La tenue d'un carnet d'adresse ne se limite pas à enregistrer les données dans le fichier et ensuite à consulter ce fichier. Il arrive fréquemment que des ajouts ou des modifications aux données déjà enregistrées soient nécessaires. Le programme suivant vous permet de contourner ce problème.

```
10 CLEAR 500 : MAXFILES = 2
20 ON ERROR GOTO 290
30 OPEN « ADRESSE.RND »AS #1 LEN=91
40 OPEN « ADRESSE.INX » AS #2 LEN=16
50 FIELD #1, 2 AS LR$, 89 AS NUL$
60 FIELD #1, 16 AS Z1$, 16 AS Z2$, 25 AS Z3$, 6 AS Z4$, 4 AS Z5$ , 16 AS Z6$, 8 AS Z7$
70 FIELD #2, 16 AS Y1$
80 GET#1, 1
90 LR=CVI(LR$)
100 CLS
110 INPUT « QUEL NOM CHERCHEZ-VOUS »;N$
120 FOR I=2 TO LR
130 GET#2, I
140 IF Y1$=N$ GOTO 180
150 NEXT
160 PRINT « Ce nom n'existe pas. Voulez-vous l'insérer » ;
161 INPUT A$ : IF LEFT$(A$,1)= «N» GOTO 230
162 IF LEFT$(A$, 1)<> «O» GOTO 160
163 LR=LR+1
164 INPUT « PRENOM          »;P$
165 INPUT « RUE             »;R$
166 INPUT « NUMERO          »;NO$
167 INPUT « CODE POSTAL     »;CP
168 INPUT «VILLE            »;V$
169 INPUT « TELEPHONE       »;T#
170 LSET Z1$ = N$
171 LSET Z2$ = P$
172 LSET Z3$ = R$
173 LSET Z4$ = NO$
174 LSET Z5$ = MKS(CP)
175 LSET Z6$ = V$
176 LSET Z7$ = MKD$(T#)
177 PUT#1, I
178 LSET Y1$=N$
179 PUT#2, I : GOTO 230
180 GET#1, I
190 PRINT : PRINT Z1$ : PRINT Z2$
200 PRINT Z3$ ; « , »;Z4$
210 PRINT CVS(Z5$);SPC(5);Z6$
220 PRINT CVD(Z7$)
221 PRINT : PRINT
222 INPUT « NOM             »;N$
223 GOTO 164
```

```
230 PRINT
240 PRINT « Return pour continuer – ESC pour arrêter »
250 A$=INKEY$:IF A$= «» GOTO 250
260 IF A$=CHR$(13) GOTO 100
270 IF A$<>CHR$(27) GOTO 250
280 CLOSE:END
290 IF ERR=70 GOTO 310
300 ON ERROR GOTO 0
310 PRINT « Pas de disque dans le lecteur »
320 INPUT « Return quand problème corrigé »;A$
330 CLS : RESUME
```

3.7 Les instructions utilitaires

3.7.1 DSKF DiSK Free space

DSKF [<numéro lecteur>]

La fonction DSKF permet d'obtenir le nombre de clusters libres sur la disquette insérée dans le lecteur numéro X. Pour rappel, un cluster est l'unité logique d'affectation d'espace-disquette. Tous les types de disquettes de l'annexe A ont une taille de cluster de 1 K (1024 bytes) sauf les types FC et FE où le cluster a une taille de 512 octets.
<numéro lecteur> est un numéro de 0 à 8 indiquant à quel lecteur vous demandez son espace libre. 0 vaut pour le lecteur courant, 1 pour le lecteur A :, 2 pour le lecteur B :, et ainsi de suite jusqu'à 5 pour le lecteur H :.
Si vous désirez obtenir l'espace libre du lecteur en nombre d'octets, multipliez simplement le nombre obtenu par 1024 ou 512 suivant le type de disquette insérée. Rappelez-vous également que le plus petit programme ou fichier occupe toujours au moins 1 cluster sur la disquette.
Exemple :

```
PRINT DSKF(0)          : Donne le nombre de clusters libre du lecteur courant
PRINT DSKF(1)*1024     : Donne le nombre d'octets libres du lecteur A :
```

```
340 IF DSKF(0)<10 THEN PRINT « PAS ASSEZ D'ESPACE DISQUE »
```

3.7.2 DSKI$ DiSK Input

DSKI$[<numéro lecteur>,<numéro secteur>]

La fonction DSKI$ permet de lire le secteur portant le <numéro secteur> de la disquette insérée dans le lecteur portant le <numéro lecteur> vers le tampon mémoire du Directory dont l'adresse est donnée par le pointeur F351H en 16 bits inversés.

<numéro lecteur> est un numéro de 0 à 8 où 0 vaut pour le lecteur courant, 1 pour le lecteur A :, 2 pour le lecteur B :, et ainsi de suite jusqu'à 5 pour le lecteur H :.

<numéro secteur> est un numéro définissant quel secteur doit être lu. Le premier secteur de la disquette porte toujours le numéro 0 et le dernier dépend du type de disquette (voyez l'annexe A

pour connaître pour chaque type de disquette son nombre exact de secteurs). Ainsi, le type de disquette F8 (3’’1/2 – Simple face – 80 pistes – 9 secteurs/piste) contient 720 secteurs numérotés de 0 à 719. Les 9 premiers secteurs (0 à 8) de cette disquette se trouvent sur la piste 0, les 9 suivants (9 à 17) sur la piste 1 et ainsi de suite. Pour les disques double-face, les 9 premiers secteurs se trouvent sur la piste 0 de la face 1, les 9 suivants sur la piste 0 de la face 2, les 9 suivants sur la piste 1 de la face 1 et ainsi de suite.

Pour connaître l’adresse mémoire du tampon Directory où sera transféré le secteur, appliquez la formule suivante :

ADRESSE – PEEK (&HF351) + 256 * PEEK (&HF352)

ATTENTION ! Il faut exploiter le contenu du tampon mémoire avant qu’une instruction comme LOAD, RUN, MERGE, KILL, SAVE ou OPEN ne le modifie.

Exemple : Le secteur 0 contient en position 3 à 10 ce que l’on appelle l’étiquette du fabricant du contrôleur-disque. C’est l’abréviation du nom du fabricant et le numéro de la version de sa ROM. Ce court programme va vous permettre de connaître quel contrôleur a formaté la disquette présente dans le lecteur A :

```
10 CLS
20 BU = PEEK(&HF351) + 256 * PEEK (&HF352)
30 A$ = DSKI$(1,0)
40 FOR I=BU+3 to BU+10
50 PRINT CHR$(PEEK(I)) ;
60 NEXT
70 END
```

La variable A$ reçoit une chaîne vide. Elle est placée là uniquement parce que DSKI$ est une fonction et non une instruction.

3.7.3 DSKO$ DiSK Output

DSKO$<numéro lecteur>,<numéro secteur>

L’instruction DSKO$ pemet d’écrire le tampon mémoire réservé au Directory et indiqué par le pointeur F351H sur le secteur <numéro secteur> de la disquette insérée dans le lecteur <numéro lecteur>.

<numéro lecteur> est un numéro de 0 à 8 où 0 vaut pour le lecteur courant, 1 pour le lecteur A :, 2 pour le lecteur B :, et ainsi de suite jusqu’à 5 pour le lecteur H :.

<numéro secteur> est un numéro définissant quel secteur doit être écrit. Le premier secteur de la disquette porte toujours le numéro 0 et le dernier dépend du type de disquette (voyez l’annexe A pour connaître pour chaque type de disquette son nombre exact de secteurs). Ainsi, le type de disquette F8 (3’’1/2 – Simple face – 80 pistes – 9 secteurs/piste) contient 720 secteurs numérotés de 0 à 719. Les 9 premiers secteurs (0 à 8) de cette disquette se trouvent sur la piste 0, les 9 suivants (9

à 17) sur la piste 1 et ainsi de suite. Pour les disques double-face, les 9 premiers secteurs se trouvent sur la piste 0 de la face 1, les 9 suivants sur la piste 0 de la face 2, les 9 suivants sur la piste 1 de la face 1 et ainsi de suite.

Pour connaître l'adresse mémoire du tampon Directory où sera transféré le secteur, appliquez la formule suivante :

ADRESSE – PEEK (&HF351) + 256 * PEEK (&HF352)

Attention que cette instruction peut détruire votre disquette puisqu'elle y écrit directement sans aucun contrôle. Choisissez donc une zone de la disquette où vous êtes sûr qu'il n'y a pas de fichier. Pour vos essais, il est d'ailleurs préférable d'employer une disquette nouvellement formatée et qui ne contient aucun fichier. Attention aussi à la zone réservée (Boot secteur – FAT – Directory – voyez le chapitre 4).

Si le secteur 1 indiquant notamment le type de disquette est détruit, cette instruction ne peut pas fonctionner.

Exemple : Programme de copie physique d'une disquette entière. Bien que ce programme puisse tourner sur les configurations à un seul lecteur, il est recommandé de l'utiliser sur des MSX à deux unités, sinon, vous devrez interchanger les disquettes autant de fois qu'il y a de secteurs à copier. Le nombre de secteurs à copier est obtenu par la lecture des octets 19 et 20 du secteur 0 de la disquette à copier (lignes 50-70).

```
10 CLS
20 PRINT « COPIE DU LECTEUR A : SUR LE LECTEUR B : »
30 INPUT « PRET (O/N) »;A$
40 IF LEFT$(A$,1)<> «O» GOTO 30
50 A$ = DSKI$(1,0)
60 BU= PEEK(&HF351) + 256 * PEEK (&HF352)
70 MS = PEEK (BU+19) + 256 * PEEK (BU+20)
80 FOR I = 1 TO MS
90 A$ = DSKI$(1, I)
100 DSKO$ 2, I
110 PRINT I
120 NEXT
130 END
```

3.7.4 VARPTR VARiable PoinTeR

VARPTR (#<numéro fichier>)

la fonction VARPTR(#) retourne l'adresse mémoire où est implanté le File Control Block (FCB) du fichier <numéro fichier>.

Pour une explication détaillée du FCB, voyez le chapitre 5 mais, en résumé, on y trouve le nom du fichier, la date et l'ehure des dernières modifications, la dimension du fichier, le premier cluster du fichier, le dernier cluster accédé, la longueur du record, etc.

Ainsi par exemple, la position FCB+25 indique en quelle position ce fichier se trouve dans le Directory. Le petit programme suivant va nous indiquer, pour un fichier au choix, la position qu'il occupe dans le Directory.

```
10 CLS
20 INPUT « NOM DU FICHIER »;F$
30 OPEN F$ AS #1
40 FCB = VARPTR(#1)
50 P = PEEK(FCB+25)
60 PRINT « POSITION »;P
70 CLOSE
80 END
```

Messages d'erreur du Disk-Basic

Lorsqu'une erreur intervient dans le déroulement des instructions d'un programme, celui-ci s'interrompt et un message approprié est affiché.
Le Disk-BASIC a ajouté une série de nouveaux messages d'erreur dont la liste suit. Grâce à l'instruction ON ERROR GOTO, et à l'instruction IF ERR=XX THEN, on peut éviter l'arrêt du programme et traiter soi-même l'erreur.

50 FIELD OVERFLOW
une instruction FIELD tente d'allouer plus d'espace à ses variables qu'il n'en a été prévu dans l'OPEN du fichier à accès direct concerné.

51 INTERNAL ERROR
Erreur interne au Disk-BASIC qui est tellement rare que MICROSOFT demande qu'on lui rapporte les circonstances dans lesquelles elle se produit.

52 BAD FILE NUMBER
Une instruction ou une fonction fait référence à un fichier qui n'est pas ouvert, ou encore ce numéro de fichier est hors tolérance (plus grand que 6 ou plus grand que le maximum prévu par l'instruction MAXFILES).

53 FILE NOT FOUND
Un LOAD, BLOAD, RUN, MERGE, NAME ou KILL fait référence à un fichier qui n'existe pas sur le disque demandé.

54 FILE ALREADY OPEN
Une instruction OPEN est émise pour un fichier qui est déjà ouvert, ou une instruction KILL essaye de détruire un fichier qui n'est pas encore fermé.

4) Chapitre 4 : l'organisation de la disquette

4.1 Le découpage de la disquettes

Une disquette MSX est découpée en 5 zones :

1) Le boot sector
2) La FAT (File Allocation Table)
3) La copie de la FAT
4) Le Directory
5) La zone des fichiers

Suivant le type de disquette, il y a une ou deux faces, 40 ou 80 pistes, 8 ou 9 secteurs par piste . La face 1 se trouve du côté du moyeu d'entraînement de la disquette, la face 2 du côté de l'étiquette. La piste 0 est située près du bord de la disquette, tandis que la dernière piste est située près du centre.

Le secteur 0 d'une piste est situé le premier après l'index qui est un repère physique indiquant l'endroit du début de la piste. Les autres secteurs sont placés séquentiellement par ordre numérique.

La numérotation des secteurs se fait en donnant le numéro 0 au premier secteur de la première piste de la première face. On attribue ensuite les numéros suivants aux autres secteurs de la piste. Lorsque tous les secteurs d'une piste ont reçu leur numéro, la numérotation continue sur la face 2 pour les disques double face. Pour les disquettes simple face, ou lorsque tous les secteurs de la face 2 – piste 0 ont reçu leur numéro pour les disquettes double face, on passe à la piste suivante et ainsi de suite.

L'emplacement de chacune des 5 zones dépend du type de disquette (voir l'annexe A). Cependant, la plupart des utilisateurs emploient des disquettes de 3''1/2 soit simple face (360K) ou double face (720K).

Voici un tableau indiquant les secteurs réservés à ces 5 zones.

SECTEURS		DESCRIPTION
360	720	
0	0	Boot sector MSX-DOS. Permet l'installation du MSX-DOS si MSXDOS.SYS est présent sur la disquette.
1 2	1 2 3	FILE ALLOCATION TABLE (FAT) : Ces secteurs contiennent une table d'allocation des secteurs pour les différents fichiers de cette disquette.
3 4	4 5 6	Copie secondaire de la FAT. C'est une copie des secteurs précédents.
5 6 7 8 9 10 11	7 8 9 10 11 12 13	Ces secteurs forment le répertoire (Directory). C'eszt la liste des noms de fichiers contenus dans cette disquette avec, pour chacun d'eux, une série de paramètres.

12	14	Secteur à partir duquel vos fichiers s'installent.
719	1439	C'est le dernier secteur de votre disque.

4.2 Le cluster

Littéralement, cluster signifie collection, agglomérat. Dans notre cas, il représente un ensemble de secteurs sur disquette.

La norme MSX a choisi une taille de cluster dépendant de la capacité de la disquette (voir annexe A). Comme la plupart des disquettes MSX ont deux secteurs par cluster et que la taille habituelle d'un secteur est de 512 octets, on dit qu'un cluster vaut 1 K. Mais rappelez-vous que l'on pourrait, théoriquement, voir venir sur le marché d'autres types de lecteurs de disquettes avec une taille-secteur différente et un nombre de secteurs par cluster différent lui aussi. En conséquence, les concepteurs du DOS (Disc Operating System) ont cherché à s'isoler de la notion de secteur en créant le cluster. Cela leur permet d'écrire leur DOS sans aucune contrainte de matériel.

Rappelons ici que le secteur est la plus petite information qui puisse être lue ou écrite sur disque. Impossible, par exemple, de lire ou d'écrire 2 caractères de/sur le disque.

Lorsque le Disk-BASIC sauve un programme sur disquette, il va allouer à ce programme non pas le nombre de secteurs strictement nécessaires mais plutôt autant de clusters que le résultat de la division de la longueur du programme par la longueur du cluster, plus un cluster.

Exemple	Taille	K	Nombre de clusters
Programme1	217	0	1
Programme2	649	0	1
Programme3	1024	1	2
Programme4	1739	1	2
Programme5	5120	5	6

Avantage du système :

Il est réservé plus d'espace que nécessaire sur le disque. Dès lors, si vous modifiez ce programme de telle sorte qu'il s'agrandisse, il y a déjà une place réservée sur le disque pour cela et il n'est pas nécessaire de procéder à des extensions du cluster sur la disquette. Donc tout le programme se trouvera concentré à la même place sur la disquette ; d'où un accès plus rapide aux informations.

Inconvénient du système :

Si vous sauvez beaucoup de petits programmes sur votre disquette, à chaque fois il sera réservé un espace plus grand et dès lors les 360 K de votre disque ne pourront pleinement être utilisés.

4.3 Le record

Le record (enregistrement) est l'unité logique d'information à transférer de ou vers le fichier, aux yeux du programmeur.

En BASIC, cette notion ne s'applique qu'aux fichiers à accès direct et sa taille est programmable de 1 à 256 octets. La limitation de la taille à 256 octets est due au fait qu'il faut réserver des tampons en mémoire pour chaque fichier ouvert sans trop étendre pour autant cette mémoire réservée.

En MSX-DOS, le record a une valeur de 128 octets pour toutes les fonctions compatibles CP/M (fonctions 0F à 24 – voir chapitre 7). Pour les fonctions typiquement MXS-DOS (26H et 27H), la longueur du record est programmable par l'utilisateur de 1 à 65536 octets (cette dernière valeur est théorique : tout dépend de la place mémoire réservée au tampon du record).

4.4 L'extent

Le programmeur BASIC peut ignorer ce terme car il n'a de signification que pour certaines fonctions MSX-DOS compatibles CP/M. En CP/M, le terme cluster n'existe pas. L'unité d'allocation sur le disque est l'extent. Il occupe des secteurs consécutifs sur le disque et sa taille est un des paramètres du CP/M. Supposons qu'il ait une taille de 16 K. Il pourrait alors stocker 128 records de 128 octets.

(128 * 128 = 16384 = 16 K).

Lorsque le programmeur CP/M veut lire ou écrire un ou plusieurs records d'un fichier séquentiel, il peut préciser dans un zone mémoire spéciale appelée FCB (File Control Block) à partir de quel record, de quel extent, il veut manipuler son fichier séquentiel. En effet, il est réservé 16 bits dans la FCB pour donner le numéro d'extent et 7 bits pour le numéro de record dans cet extent. Ce terme est défini ici pour assurer la compréhension du paragraphe concernant le FCB (voir el chapitre 5) ; à part cela, il n'a donc aucun sens en MSX-DOS.

4.5 Le Boot sector

Le terme boot sector vient de l'expression anglaise BOOTSTRAP, que l'on pourrait traduire par passerelle. Tout comme la passerelle d'un bateau nous permet de le charger, le bootstrap sector (en abrégé boot sector) permet de charger le DOS (Disk Operating System = Système d'Eploitation du Disque) dans la mémoire de l'ordinateur. Chaque ordinateur dispose d'un petit programme intégré dans une ROM qui va provoquer, à l'allumage de l'ordinateur, la lecture du secteur 0 de la disquette insérée dans le premier lecteur du système.

Le contenu de ce secteur 0 est lui-même un programme qui va s'occuper de charger le fichier du système d'exploitation proprement dit en mémoire.

En plus de ce programme, le boot sector contient une série de paramètres sur les caractéristiques physiques de la disquette et sur l'organisation des données sur celle-ci. Microsoft, qui a écrit le MSX-DOS pour MSX et le MS-DOS pour IBM ne s'est pas compliqué la tâche, puisque le boot sector des deux systèmes est totalement identique, à l'exception, bien sûr, du programme de chargement qui est écrit en langage machine Z80 pour le MSX-DOS et en langage machine 8088/86 pour le MS-DOS.

Les 256 premiers caractères du boot sector sont visibles en mémoire à l'adresse C000H dès la fin de l'initialisation du système.

En résumé, le boot sector est chargé lorsque vous introduisez la disquette dans votre lecteur immédiatement après le démarrage de l'ordinateur. Les 256 premiers caractères du secteur se chargeront à l'adresse mémoire C000H. La table des caractéristiques du disque est chargée en mémoire, mais le système MSX ne l'utilise pas du tout (il se basera sur le premier caractère de la FAT pour identifier le type de votre disque et en déduira ses caractéristiques lui-même). Si le fichier MSXDOS.SYS est présent sur votre disquette, il sera chargé et installera le MSX-DOS en mémoire, chargera COMMAND.COM et lui passera la main. Vous verrez alors apparaître le prompt distinctif du MSX-DOS : A>

Voici le découpage du secteur 0 :

POS	LONG	Description
0	3	Instruction de saut en langage machine 8088 (IBM). Deux possibilités existent ; EB FE et E9 LL MM. La première empêche l'IBM de charger son DOS à partir de la disquette MSX. La seconde le permet par un saut à l'adresse MMLL.
3	8	Étiquette du fabricant et version software.
11	2	Nombre d'octets par secteur (0200H=512).
13	1	Nombre de secteurs par cluster.
14	2	Numéro du premier secteur de la FAT.
16	1	Nombre de FAT stockées sur le disque.
17	2	Grandeur maximum de le Directory en fichiers.
19	2	Nombre total de secteurs sur le disque.
21	1	Code du type de disque (1-1-1-1-1-P-S-F) F=0 → Simple face F=1 → Double face S=0 → 9 secteurs/piste S=1 → 8 secteurs/piste P=0 → 80 pistes P=1 → 40 pistes
22	2	Nombre de secteurs réservés pour une FAT.
24	2	Nombre de secteurs par piste.
26	2	Nombre de têtes de lecture/écriture.
28	2	Décalage du disque logique en nombre de secteurs.
30	91 103*	Programme en langage machine Z80 qui charge le fichier dont le nom se trouve en position 160 à 170 à l'adresse 100H. Si ce fichier n'existe pas Disk BASIC est installé.
121 133*	38	Message d'erreur de mauvais transfert du fichier à charger . Peut être traduit en français. Le message doit se terminer par le caractère $.
159 171*	1	Numéro du disque pour lire le fichier à charger (0=disque courant. 1 à 8 = disque A : à H:).
160 172*	11	Nom du fichier contenant le DOS. Normalement, on y trouve « MSXDOS.SYS » mais on pourrait y trouver d'autres noms comme « CPM ».

		Ce fichier ne peut dépasser 16128 octets.
171 183*	26	Initialisé à 00H et réservé au FCB du fichier.
197 209*	59 47*	Initialisé à 00H et non réservé. Le texte qui vous y mettriez se retrouvera à l'adresse C0C5H.
256	256	Initialisé à 00H mais n'apparaît pas en mémoire.

(*) Dans les dernières versions le programme a été allongé.

4.6 Le Directory

Le rôle du Directory est d'indiquer pour chaque fichier, l'endroit où ce fichier est situé sur la disquette, sa longueur, son heure de création ou de modification sous forme d'une date et d'une heure.

Il est réservé 32 octets par fichier dans le Directory. On peut habituellement placer 16 noms de fichier par secteur du Directory, puisqu'un secteur fait habituellement 512 octets (512/32=16).

Le Directory ayant une longueur de X secteurs (voir annexe A) contiendra donc un nombre maximum de fichiers bien défini que l'on ne pourra jamais dépasser (112 tant pour les disquettes 3''1/2 de 360K ou 720K).

Le nombre maximum de fichiers que vous mettrez sur votre disquette dépendra aussi de la longueur de chacun de ces fichiers. En effet, il ne suffit pas de trouver place dans le Directory pour y sauver le nom de votre fichier, il faut encore que les données de votre fichier trouvent place dans la partie de la disquette réservée aux fichiers.

Le système MSX a prévu un tampon-mémoire, de la taille du plus grand secteur, pour le Directory. L'adresse de ce tampon est indiquée par le pointeur F351H. En BASIC, pour obtenir l'adresse de ce tampon, tapez :

```
PRINT HEX$(PEEK(&HF351)+256*PEEK(&HF352))
```

Voici la constitution du Directory :

POS	LONG	DESCRIPTION
0	8	Nom du fichier cadré à gauche complété d'espaces.
8	3	Extension du nom de fichier comme ci-dessus.
11	1	Attribut du fichier. Le MSX-DOS utilise les bits D, N, S, C de cet octet pour manipuler ou afficher les fichiers ayant certains de ces 4 bits à 1. 7 6 5 4 3 2 1 0 0 0 A D N S C R Où, pour un IBM-PC : R=1 Seule la lecture de ce fichier est permise R=0 Lecture, écriture et effacement permis C=1 Fichier caché lors de sa recherche

		C=0 Fichier non caché S=1 Fichier système en langage machine IBM S=0 Fichier non-système N=1 Nom de ficher est le nom de la disquette N=0 Fichier normal D=1 Ce fichier est une sous-Directory D=0 Ficher normal A=1 Mis à 1 quand le fichier est modifié A=0 Remis à 0 lorsque le fichier aura été archivé par la commande BACKUP
12	10	Réservés au MSX-DOS (toujours à 00H!).
22	2	Donne en 16 bits inversés, l'heure de création ou de modification du fichier sous la forme : Position 23 : Position 22 H H H H H M M M M M M S S S S S HHHHH donne l'heure (0-23) en binaire MMMMMM donne les minutes (0-59) en binaire SSSSS donne la moitié des secondes (0-29) Si votre MSX n'a pas de dateur électronique, la zone contiendra 0000H.
24	2	Donne en 16 bits inversés, l'heure de création ou de modification du fichier sous la forme : Position 25 : Position 24 A A A A A A A M M M M J J J J J AAAAAA donne le nombre (00-99) à ajouter à 1980 pour obtenir l'année MMMM donne le mois (1-12) en binaire JJJJJ donne le jour (1-31) en binaire En MSX1, cette zone n'est valable que si vous entrez la date à l'allumage de votre système.
26	2	Donne en 16 bits inversés le numéro du premier cluster du fichier. Le premier fichier de le Directory est toujours positionné sur le cluster 2. Cette zone indique donc le secteur ou commence le fichier en appliquant la formule décrite dans le paragraphe suivant qui explique la FAT.
28	4	Donne en 32 bits inversés la taille du fichier.

4.7 La FAT (File Allocation Table)

L'analyse du fonctionnement de la FAT va vous initier à la méthode employée par le MSX (et par l'IBM PC) pour sauver vos fichiers sur disquette.

Supposons, pour faciliter les choses, que vous venez de formater une disquette. Elle ne contient donc encore aucun fichier. Créons un petit programme BASIC et sauvegardons-le par la commande SAVE « PROGRAM1.BAS ». Le système commence par écrire dans la première entrée du Directory le nom du programme, sa longueur, la date et l'heure (si vous avez un MSX2).

Nous avons déjà vu dans le paragraphe du cluster, que les données sont sauvées sur disque par unité d'allocation c'est-à-dire par cluster (2 secteurs pour la plupart des types de disquette). Ainsi une

disquette de 360K – 3’’1/2 en contient 360 clusters. Une disquette de 720K – 3’’1/2 en contient 720. Les clusters sont numérotés de 0 à 359 ou de 0 à 719 suivant le type de disquette.

Vous savez que la disquette contient, outre vos fichiers, un boot secteur, deux copies des secteurs FAT et un Directory de 7 secteurs. De ce fait le nombre de clusters libres utilisables pour nos fichiers est réduit à 354 clusters pour une disquette 360K – 3’’1/2 et à 713 clusters pour une disquette 720K – 3’’1/2. Pour des raisons d’organisation interne de la table, le premier cluster réservé aux fichiers est le 002.

Les secteurs FAT sont toujours lus préalablement à toute opération disque par le système et résident donc en mémoire dans un tampon qui leur est spécialement réservé. Ce tampon est un multiple de la taille du secteur de cette disquette et joue le rôle dynamique dévolu à la FAT.

La FAT est divisée en 360 ou 720 entrées numérotées de 0 à 359/719. L’entrée 0 correspond au cluster 0, l’entrée 1 au cluster 1… et l’entrée 359 au cluster 359.

Chaque entrée de la table contient 12 bits et permet donc de donner un numéro de 0 à 4095 (en hexadécimal de 000H à FFFH). Le numéro de 12 bits indique quel cluster fait suite au cluster correspondant au numéro d’entrée de la table. Ainsi donc, si on trouve 00EH (14 en décimal) dans l’entrée 8, cela signifie que le cluster 14 fait suite au cluster 8.

Revenons maintenant à notre petit programme « PROGRAM1.BAS » dont nous avons commandé la sauvegarde par SAVE « PROGRAM1.BAS ». Analysons la procédure suivie par le MSX.

Le cluster de départ du premier fichier sauvé est toujours 002. Donc la première entrée de le Directory contiendra 002H dans la position 26-27 (Start cluster). Le système va donc sauver le début de votre programme dans le cluster 2 à concurrence de 1023 octets (voir chapitre sur la structure des fichiers). Si votre programme est plus long que 1023 octets, il faut donc rechercher un cluster libre puisqu’il n’y a plus de place dans le cluster 2. Pour ce faire, le MSX balaie la table FAT à partir du cluster 2 jusqu’à ce qu’il trouve une entrée libre. Une entrée est libre quand elle contient 00H.

Puisque notre disque vient d’être formaté, toutes les entrées suivant l’entrée 2 sont libres. Le système va donc isoler l’entrée 3 comme libre. Le système va maintenant écrire le numéro de l’entrée libre dans l’entrée réservée au cluster 2 et écrire les 1024 octets suivants de votre programme dans le cluster 3. Cette technique va ainsi se répéter tant qu’il subsiste une portion de votre programme à sauver.

Quand la dernière portion est arrivée, le cluster final est écrit avec le solde des caractères de votre programme et l’entrée correspondante à ce dernier cluster est marquée avec FFFH, ce qui signifie que ce cluster est le dernier alloué à votre programme.

Donc, si PROGRAM1.BAS a une longueur de 6150 octets par exemple, la FAT se composera de :

E	**Con**	**Description**
0	F F 8	Type de disque (sera vu pus loin)
1	F F F	Cluster réservé (sera vu plus loin)

2 0 0 3 Le cluster 2 est suivi du 3 et contient les 1023 premiers octets du programme.

3 0 0 4 Le cluster 3 est suivi du 4 et contient les 1024 octets suivants du programme.

4 0 0 5 Le cluster 4 est suivi du 5 et contient les 1024 octets suivants du programme.

5 0 0 6 Le cluster 5 est suivi du 6 et contient les 1024 octets suivants du programme.

6 0 0 7 Le cluster 6 est suivi du 7 et contient les 1024 octets suivants du programme.

7 0 0 8 Le cluster 7 est suivi du 8 et contient les 1024 octets suivants du programme.

8 F F F C’est le dernier cluster et contient les 7 derniers octets du programme

9 0 0 0 Ce cluster est libre

10 0 0 0 ‘’ ‘’ ‘’

.. 0 0 0 ‘’ ‘’ ‘’

.. 0 0 0 ‘’ ‘’ ‘’

359 0 0 0 Ce cluster est le dernier de la disquette (360K) et est libre

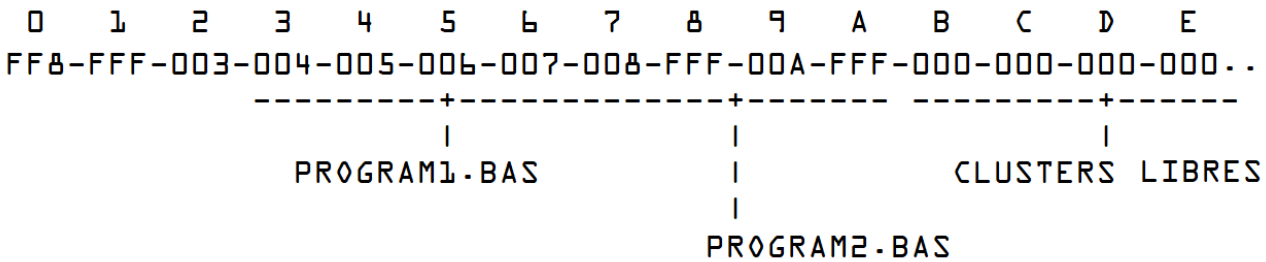
Le programme PROGRAM1.BAS sera donc sauvé sur les clusters 2, 3, 4, 5, 6, 7 et 8. Au niveau des numéros de secteur, cela dépend du type de disquette (voir annexe A). En effet, le cluster 2 correspond au secteur 12 pour les disquettes de 360K et au secteur 14 pour les disquettes de 720K. Notre programme est donc sauvé sur les secteurs 12 à 25 pour une disquette de 360K.

Remarquons au passage, que le cluster 8 (secteurs 24 et 25) ne contient que 7 octets de notre programme. Les 1017 derniers octets de ce cluster contiendront des données qui ne seront pas prises en compte lors d’un chargement futur de ce programme. Elles proviennent du tampon secteur (Sector Buffer) et nous y retrouverons donc les 512 octets du deuxième secteur du cluster précédent écrasés par les 7 derniers octets du programme. Ces données se retrouveront sur les secteurs 24 et 25.

On peut donc affirmer que la FAT est une table de chaînage des clusters alloués à chaque fichier. Le premier cluster d’un fichier est donné par le start cluster du Directory et les autres par la FAT.

Imaginons maintenant la création d’un programme BASIC que nous nommerons PROGRAM2.BAS et sauvons sur la même disquette. L’opération sera identique à celle de la sauvegarde du programme PROGRAM1.BAS excepté que le start cluster de ce programme ne sera plus le cluster 2, mais sera trouvé par recherche d’un cluster libre dans la FAT.

Si notre nouveau programme a une taille de 2000 octets, il sera donc sauvé sur les clusters 9 et 10 et la FAT contiendra les entrées suivantes :



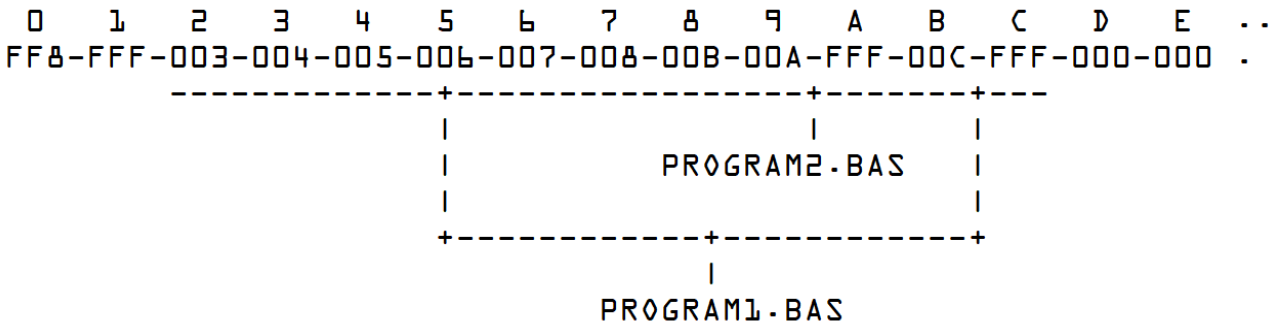
Mais que va-t-il se passer si nous chargeons notre premier programme pour le modifier et que le résultat de cette modification soit un allongement de notre programme ?

Deux possibilités existent : soit la modification a allongé le programme de moins de 1017 octets et alors il reprendra exactement la même place sur le disque en exploitant la portion précédemment inemployée du dernier cluster de ce programme, soit la modification allonge le programme d'une valeur supérieure à la portion inemployée du dernier cluster et on a alors recours à la technique suivante :

Puisque le programme existe déjà sur le disque, il est d'abord détruit. Cette opération consiste à placer un caractère spécial (E5H) en lieu et place du premier caractère de nom de fichier dans le Directory et à remettre à 000H (cluster libre) toutes les entrées de la FAT précédemment allouées à ce fichier.

Ensuite, on procède comme initialement décrit, c'est-à-dire qu'une recherche du premier cluster libre est effectuée et le premier K du programme y est sauvé ; pour chaque K suivant du programme, on procède de même jusqu'à ce que tout le programme soit sauvegardé.

Il faut cependant tenir compte que les clusters 9-10 ne sont pas libres car déjà alloués au programme PROGRAM2.BAS. Si le fichier a été allongé de 2090 octets, par exemple, la FAT contient donc :



Il est à remarquer maintenant qu'il y a une discontinuité dans l'allocation des clusters de PROGRAM1.BAS due au fait que les clusters 9-10 n'étaient pas libres. Ainsi, si vous procédez fréquemment à des agrandissements de vos fichiers, l'ordre d'allocation des clusters d'un fichier peut apparaître complètement désordonné à nos yeux tout en restant parfaitement correct. A titre d'exemple la séquence suivante : 002-009-010-011-051-065-07B.

L'inconvénient d'une telle situation est que le chargement d'un tel fichier prendra plus de temps, car entre chaque cluster il faudra déplacer les têtes de lecture/écriture ce qui est une opération mécanique lente. Voici un moyen efficace de réorganiser votre FAT.

1 – Formater une disquette vierge ou effacer tous les fichiers d'une disquette inemployée.

2 – Copier tous les fichiers du disque à réorganiser sur la disquette nouvellement formatée par la commande COPY « *.* B :» en MSX-DOS.

La disquette formatée contient maintenant les fichiers de la première disquette avec une FAT parfaitement séquentielle. Vous avez tout intérêt à procéder régulièrement à ce type de copie pour

deux raisons : d'abord, l'obtention d'une copie de sécurité et ensuite réorganisation de la FAT accélérant ainsi la vitesse d'accès aux fichiers.

La FAT est une table stockée sur chaque disquette qui est chargée en mémoire lors du premier appel de fichier ou lorsque le disque est susceptible d'avoir été changé, et qui contient le chaînage des clusters alloués aux fichiers de cette disquette. Les entrées sont numérotées suivant le cluster qu'elles représentent et contiennent le numéro du cluster qui fournit la suite des données contenues dans le cluster de cette entrée. Les entrées sont de 12 bits, ce qui autorise un maximum de 4096 clusters par disque. La première entrée de la FAT contient toujours FXX en hexadécimal où XX représente le type de disque.

XX = 1 1 1 1 1 P S F	F=0 Simple face
	F=1 Double face
	S=0 9 secteurs/piste
	S=1 8 secteurs/piste
	P=0 80 pistes/face
	P=1 40 pistes/face

La deuxième entrée contient toujours FFF pour des raisons d'organisation interne de la table.

La troisième entrée correspond à l'entrée du cluster 002 et peut contenir, comme toutes les entrées suivantes d'ailleurs :

- 000H = le cluster est libre.
- FFFH = ce cluster est le dernier associé à ce fichier.
- 002H = le numéro du cluster suivant (pour IBM, la valeur FF7H signifie « Cluster défectueux » et ne doit dès lors pas être employée).
- 001H = cette valeur est réservée et jamais employée.

Pour connaître les numéros de secteurs affectés à un cluster, il faut rechercher le numéro du premier secteur utilisateur suivant le type de disque (voir annexe A) et en soustraire deux fois le nombre de secteurs par cluster, puis y ajouter le numér ode cluster multiplié par le nombre de secteurs par cluster. Soit la formule :

1er SECT. Utilisateur + (Cluster -2) * Nombre Sect. par cluster

Les valeurs employées dans la formule précédente peuvent être trouvées dans le Boot secteur ou dans le DPB (Disk Parameter Block, décrit plus loin) de cette disquette. La localisation de la FAT en mémoire est indiquée dans le DPB associé à cette disquette.

Pratiquement, l'organisation de la FAT n'est pas aussi simple, car la FAT en mémoire et sur disquette mémorise des octets et non des mots de 12 bits. L'exemple précédent donnera en réalité ceci :

```
Entrée: 000       002       004       006       008       00A
        --  -     --  -     --  -     --  -     --  -     --  -
FAT:    F8-FF-FF-03-40-00-05-60-00-07-80-00-0B-A0-00-FF-CF
           -  --     -  --     -  --     -  --     -  --     -
Entrée:    001       003       005       007       009       00B
```

Pour les entrées paires (0, 2, 4, 6…), les 4 bits inférieurs du second octet sont en fait les 4 bits supérieurs du mot de 12 bits et les 8 bits du premier octet sont les 8 bits inférieurs du mot de 12 bits.

Pour les entrées impaires (1, 3, 5…), les 8 bits du second octet sont en fait les 8 bits supérieurs du mot de 12 bits et les 4 bits supérieurs du premier octet sont les 4 bits inférieurs du mot de 12 bits.

Deux problèmes surgissent devant le programmeur Basic :

- Comment trouver la position du cluster X dans la FAT ?
- Comment interpréter le contenu des deux octets de cette position ?

Supposons que nous voulions connaître les secteurs attribués à un fichier quelconque. Il nous suffit d'abord d'ouvrir ce fichier par un OPEN basique, d'extraire le cluster de départ de ce fichier à partir du FCB (qui sera étudié au chapitre suivant), puis de trouver l'emplacement en mémoire de la FAT (chapitre suivant) et enfin d'appliquer deux formules pour trouver l'entrée correcte dans la FAT et en interpréter le contenu.

Les lignes 40 à 80 permettent de trouver l'adresse du FCB, du cluster de départ, l'adresse du DPB utilisé, l'adresse de la FAT et le nombre de secteurs par cluster (voir chapitre suivant).

La ligne 90 convertir le numéro de cluster en numéro de secteur suivant la formule vue dans cve chapitre.

La ligne 100 affiche le numéro de secteur et la ligne 110 affiche le numéro du deuxième secteur du cluster si la taille du cluster est 2 secteurs.

La ligne 120 est la première des deux formules évoquées plus haut. Elle donne un déplacement à ajouter à l'adresse de la FAT pour trouver l'entrée correspondant au cluster désiré. La ligne 130 donne l'adresse de cette entrée.

La ligne 150 est la deuxième formule évoquée. Elle convertit le contenu des 12 bits de l'entrée de la FAT en numéro de cluster pour les entrées impaires.

La ligne 160 fait de même pour les entrées paires.

La ligne 170 teste si le cluster trouvé n'est pas le dernier de ce fichier.

```
10 CLS:MAXFILES=1:I=1
20 INPUT « DONNEZ LE NOM DU FICHIER »;P$
30 OPEN P$ FOR INPUT AS I
40 FCB=PEEK(&HF353)+256*PEEK(&HF354)+I*37
50 CLU=PEEK(FCB+26)+256*PEEK(FCB+27)
60 DPB=PEEK(&HF243)+256*PEEL(&HF244)
```

```
70 FAT=PEEK(DPB+19)+256*PEEK(DPB+20)
80 NSC=PEEK(DPB+7)
90 SECT=PEEK(DPB+12)+256*PEEK(DPB+13)+(CLU-2)*NSC
100 PRINT USING « SECTEUR #### »;SECT
110 IF NSC=2 THEN PRINT USING « SECTEUR #### »;SECT+1
120 DEPLAC=1+FIX((CLU-1)*1.5+.5)
130 ENTFAT=FAT+DEPLAC
140 IF CLU MOD 2=0 GOTO 160
150 CLU=PEEK(ENTFAT+DEPLAC)\16+16*PEEK(ENTFAT+DEPLAC+1)
155 GOTO 170
160 CLU=PEEK(ENTFAT+DEPLAC)+256*(PEEK(ENTFAT+DEPLAC+1)AND 15)
170 IF CLU < &HFFB GOTO 90
180 END
```

4.8 Structure des fichiers

4.8.1 Le fichier-programme BASIC sauvé en binaire compressé

Ce fichier a été créé par l'instruction BASIC SAVE « ... ». Il contient un indicateur de programme BASIC en première position du fichier et est constitué du texte de votre programme BASIC tel qu'il se trouve en mémoire.

Vous savez que le texte de vos lignes d'instructions BASIC est encodé et que ce que vous voyez à l'écran après un LIST n'est que la traduction de ce qui se trouve réellement en mémoire. Imaginez que vous ayez sauvé le petit programme BASIC suivant sous le nom de PROGRAM.TST.

```
10 CLS
20 PRINT « MSX »
30 END
```

Voici ci-dessous le contenu du fichier « PROGRAM.TST ». Le contenu de la première colonne est en hexadécimal.

```
FF      Indicateur de programme BASIC.
07 :    Pointeur vers la ligne suivante
80 :
0A :    Numéro de la ligne (10)
00 :        ‘’      ‘’      ‘’
9F :    Code de l'instruction CLS
00      Code de fin de ligne
12 :    Pointeur vers la ligne suivante
80 :        ‘’      ‘’      ‘’          ‘’
14 :    Numéro de la ligne (20)
00 :        ‘’      ‘’      ‘’
91      Code de l'instruction PRINT
22      «
4D      M
53      S
```

58 X
22 »
00 Code de fin de ligne
18 : Pointeur vers la ligne suivante
80 : ‘’ ‘’ ‘’ ‘’
1E : Numéro de ligne (30)
00 : ‘’ ‘’ ‘’
81 Code de l’instruction END
00 Code de fin de ligne
00 : Code de fin du programme
00 : ‘’ ‘’ ‘’ ‘’

En résumé, le fichier de programme BASIC contient la valeur FF comme indicateur de type de fichier suivi du contenu de la mémoire à partir de l’adresse 8001H et jusqu’à, et y incluse, l’adresse de l’indicatif de fin de programme (00-00). En réalité, l’adresse de départ dépend du pointeur F676.

4.8.2 Le fichier de programme BASIC sauvé en ASCII

Ce fichier a été créé par l’instruction BASIC SAVE «….. », A. Il contient en ASCII le texte de votre programme BASIC tel qu’il apparaît à l’écran après un LIST. Chaque ligne est terminée par la paire de caractères CR-LF (Carriage Return et Line Feed ; 0D et 0A en hexadécimal). Il n’y a donc aucun indicatif de type de fichier ASCII ; par contre, le fichier se termine par une marque de fin de fichier qui est le code hexadécimal 1A. Si vous sauviez le programme de l’exemple précédent en ASCII, le fichier contiendrait :

31-30-20-43-4C-53-0D-0A
1 0 C L S
32-30-20-50-52-49-4E-54-22-4D-53-58-22-0D-0A
2 0 P R I N T ‘’ M S X ‘’
33-30-20-45-4E-44-0D-0A-1A → Marque de fin de fichier
3 0 E N D

4.8.3 Le fichier binaire ou langage machine

Il est créé par l’une des deux instructions BASIC suivantes :
BSAVE « ….. », &Hxxxx , &Hyyyy, &Hssss
BSAVE « ….. », &Hxxxx, &Hyyyy, S

Dans le premier cas, il s’agit de sauver une portion de mémoire comprise entre les adresses &Hxxxx et &Hyyyy qui contient généralement un programme en langage machine. L’adresse &Hssss permet de définir à quel endroit l’exécution doit commencer.

Dans le deuxième cas, il s’agit de sauver une portion de la mémoire réservée au VDP (ViDeo Processor) qui contient généralement une image.

Dans les deux cas, le contenu de cette mémoire est envoyé tel quel vers le fichier précédé d’une entête de 7 octets que voici :

FE : indicatif de fichier binaire
xx : Adresse en 16 bits inversés où stocker le fichier lors de son chargement en mémoire
yy : Adresse en 16 bits inversés où stocker le dernier octet du fichier lors de son chargement en mémoire
ss : Adresse en 16 bits inversés où démarrer l'exécution du programme lorsque le chargement se fait avec l'option ',R'
?? : Ici commence le programme ou l'image

4.8.4 Les fichiers séquentiels

Les fichiers séquentiels n'ont pas d'indicateur de type de fichier, mais par contre, sont terminés par une marque de fin de fichier (code 1AH). Ils sont créés par l'instruction BASIC « OPEN » et remplis par l'instruction « PRINT# ».

A ce sujet, il est bon de rappeler que l'instruction « PRINT#x » insère automatiquement une marque de séparation entre les données numériques (code espace), mais pas entre les chaînes de caractères et qu'elle ajoute CR-LF (code 0D-0A) à votre fichier si elle ne se termine pas par le code point-virgule. Toutes les données sont écrites en ASCII. Ainsi la portion de programme BASIC suivant créera le fichier ci-dessous.

```
10 OPEN « TEST.SEQ » FOR OUTPUT AS #1
20 A=225:A$= « SUISSE »
30 PRINT#1,32 ;-47;A
40 PRINT#1, « FRANCE » ; « BELGIQUE » ; A$
50 PRINT#1, « CANADA » ;
60 CLOSE #1
```

```
32 -47 225 **FRANCEBELGIQUESUISSE**CANADA#
```

La double astérisque représente la paire de caractères CR-LF (0D-0A). Le caractère # représente le code de fin de fichier (1A).

Lorsque le fichier sera relu par l'extrait de programme que voici,

```
200 INPUT#1,A,B,C,D,A$ ,B$,C$,D$
```

A contiendra 32
B contiendra -47
B contiendra 225
A$ contiendra FRANCEBELGIQUESUISSE
B$ contiendra CANADA
C$ et D$ ne contiendront rien

En effet, comme dans tout INPUT, la virgule sert de séparateur des données introduites. Comme l'instruction PRINT# n'insère pas cette virgule pour les chaînes de caractères, la variable A$ reçoit FRANCEBELGIQUESUISSE en un seul mot. Pour éviter ce problème, il faut corriger l'instruction de la ligne 40 en plaçant une virgule entre guillemets.

```
40 PRINT#1, « FRANCE, » ; « BELGIQUE, »;A$
50 PRINT#1, « CANADA »
```

Donc, n'oubliez pas que seuls la virgule, le CR (0D) et le LF (0A) peuvent servir de séparateurs de données de type chaîne.

4.8.5 Les fichiers à accès direct

Le fichier à accès direct ne contient pas non plus d'indicateur de type de fichier ni de marque de fin de fichier. Les données sont sauvées en ASCII sauf pour les constantes et variables numériques qui sont sauvées telles qu'elles se trouvent en mémoire. C'est-à-dire sous le format 2 octets pour les valeurs entières, 4 octets pour les valeurs simple précision et 8 octets pour les valeurs double précision. Voici ce que l'on obtient avec le petit programme suivant :

```
10 OPEN « FICHIER.DIR » AS #1
20 FIELD#1,6 AS Z1$, 2 AS Z2$, 4 AS Z3$, 8 AS Z4$
30 A$= « BASIC »:A%=16384:A!=123456:A#=12345678901234
40 LSET Z1$=A$
50 LSET Z2$=MKI$(A%)
60 LSET Z3$=MKS$(A!)
70 LSET Z4$=MKD$(A#)
80 PUT#1, 1
90 CLOSE #1:END
```

```
42      B
41      A
53      S
49      I
43      C
20      Code espace
00      16384 exprimé en 16 bits inversés (4000H=16384)
40        ‘’      ‘’      ‘’      ‘’      ‘’
46      123456 sous le format simple précision
12
34
56
4E      12345678901234 sous le format double précision
12
34
56
78
90
12
34
```

4.8.6 Les fichiers MSX-DOS

Il existe deux types particuliers de fichier MSX-DOS : les fichiers .COM et .BAT. Ils n(ont aucun des deux un format spécifique. Le fichier .COM est simplement la série de codes qui forment un programme exécutable. Le fichier .BAT est un fichier ASCII (donc terminé par le code 1AH) contenant une liste de commandes MSX-DOS séparées par CR-LF.

Certains assembleurs comme MACRO80 de Microsoft ou certains compilateurs PASCAL, FORTRAN, COBOL ou C peuvent générer des fichiers ayant une structure particulière propre (ex : .REL, .CRF, .HEX pour MACRO80). Reportez-vous à la documentation de ces divers compilateurs pour plus de renseignements.

4.9 Le formatage physique d'une piste

Rappelons pour les non-initiés que le formatage d'une disquette vierge est nécessaire avant toute utilisation de celle-ci…
L'opération de formatage est celle par laquelle le hardware, piloté par un programme spécial, va découper chaque piste de la disquette en 8 ou 9 secteurs suivant le format choisi.
Potentiellement, une piste vierge est capable d'accueillir 6250 octets. Cependant, en MSX (comme dans tout autre système d'ailleurs), l'utilisateur n'aura en fait droit qu'à 9 secteurs de 512 octets, soit 4608 octets par piste. Où sont donc passés les 1642 autres octets ?
Ils vont tout simplement servir à créer un « encadrement » pour les 9 secteurs. Cet encadrement aura pour mission d'annoncer le numéro du secteur qui va bientôt passer sous la tête de lecture et de permettre la vérification de paramètres comme le numéro de piste et de tête.
Il va aussi servir à isoler un secteur du suivant dans le but de tolérer une certaine variation de la vitesse du moteur ou des caractéristiques physiques de la disquette. Voici le format utilisé pour les disquettes de 9 secteurs/piste. Il ne sera pas commenté ici, car réservé aux spécialistes hardware (pour ceux-ci, nous précisons qu'il n'y a pas d'interleave de secteurs).

T	V	Description
80	4E	Zone de l'index (Index Gap)
12	00	Synchronisation du PLO (Phase Lock Oscillator)
3	F6	Marque d'index (Harware Index Mark)
1	FC	Identificateur d'index (Index Id)
26	4E	Zone de fin d'index (Post Index Gap)
12	00	Synchronisation du PLO (Phase Lock Oscillator)
3	A1	Marque de secteur (Hardware Sector mark)
1	FE	Identificateur d'en-tête de secteur (header Id)
1	xx	Numéro de la piste (Track Number)
1	xx	Numéro de la tête (Head Number)
1	xx	Numéro de secteur dans la piste (Physical Sector Number)
2	xx	Caractères de redondance cyclique (CRCC)
24	4E	Zone intra-secteur (Sector Gap)
12	00	Synchronisation du PLO
3	A1	Marque de secteur (Hardware Sector Mark)
1	FB	Identificateur de données (Data Id)
12	xx	Zones des données du secteur (Data)
2	xx	Caractères de redondance cyclique (CRCC)

54 4E Zone inter-secteurs (Inter-Sector Gap)
458 4E Zone d’avant index (Final Gap)

5) Chapitre 5 : la carte mémoire de la zone RAM utilisée par le MSX-DOS et le Disk-BASIC

Lorsque vous faites l’acquisition d’une première unité de disquettes, vous recevez en même temps son interface. Cette interface contient tout le hardware nécessaire au pilotage du disque y compris une ROM de 16K contenant :

1) Le Disk-BASIC programmé par Microsoft
2) Le kernel MSX-DOS (noyau MSX-DOS) programmé par Microsoft
3) Le Disk Driver (pilote du disque) programmé par le fabricant du hardware de cette interface

Le fait que le Disk Driver soit programmé par le fabricant du hardware autorise plus de souplesse dans la conception du hardware. Ainsi certaines disquettes pourraient avoir des secteurs de 128, 256, 512 ou 1024 octets, ou encore être des disques durs de grande capacité ou même des disques à Laser.

Mais d’autre part, certains fabricants ne respectent pas toujours à la lettre la norme imposée par Microsoft d’où certaines différences de fonctionnement ou même certaines erreurs.

Avant de parler de la carte mémoire proprement dite, il faut citer et expliquer les 3 méthodes d’allumage d’un MSX avec disquette(s), car elles influencent la taille et la configuration de la mémoire RAM réservée.

1) Allumer l’ordinateur en laissant le doigt sur la touche Majuscule jusqu’au bip sonore provoque que le Disk-Basic n’est pas installé. Vous aurez ainsi un MSX avec 28815 octets libres mais pas d’accès au(x) disque(s). Cette manipulation est parfois nécessaire avec certaines cassettes ou cartouches de jeu (Eddy2 par exemple) lorsque la mémoire libre est insuffisante avec le Disk-Basic installé.

2) Allumer l’ordinateur normalement fait en sorte que le système réserve de l’espace en mémoire pour deux unités de disquettes par interface détectée, qu’il y ait un ou deux lecteurs connectés à chacune de ces interfaces.

3) Allumer l’ordinateur en laissant le doigt sur la touche CTRL (Control) jusqu’au bip sonore provoque que le système ne réserve de l’espace en mémoire que pour le premier lecteur de chaque interface. Si le second lecteur de certaines de ces interfaces était allumé durant le démarrage de l’ordinateur, alors de l’espace en mémoire est également réservé pour ces unités allumées. Cette technique permet d’ajuste le mémoire réservée au minimum requis pour la configuration dont vous avez besoin. Première différence entre fabricants, la version VG 8235 de Philips ne retient que la première unité de chaque interface même si la seconde était allumé.

Une autre liberté a été prise par JVC (Version JVCKT2) vis-à-vis de la norme de Microsoft et cela a beaucoup plus de conséquence. En effet, la version d’interface JVCKT2 ne réserve de l’espace pour la FAT que pour des disquettes de 360K. Cela a pour avantage de laisser 24456 octets libres au lieu

de 23432, mais par contre provoque que vous ne pourrez pas y connecter un lecteur de 720K ou en tous cas vous ne pourrez en accéder que les 680 premiers K.

5.1 La carte mémoire générale

5.1.1 Taille des zones mémoire pour le DOS et le Disk-BASIC

Vous savez certainement que le BASIC normal de votre MSX s'est déjà réservé la portion de mémoire RAM qui s'étend de F380H à FFFFH. Cette région de communication contient une série de paramètres, de tampons et de constantes que Le livre du MSX de Daniel Martin vous expliquera. Je ne peux que vous en conseiller vivement la lecture.

La région de communication réservée pour le DOS est constituée de deux parties. La première est fixe et s'étend de la position F1C9H à F37FH, c'est-à-dire juste sous la région de communication du BASIC. La seconde partie est de longueur variable et dépend en fait du nombre et du type de disques reconnus par votre MSX lors de l'allumage et du fabricant du contrôleur dont vous êtes équipés.

A titre d'exemple, cette seconde partie de la région de communication disque peut s'étendre de E68FH à F1C8H pour un MSX à un seul contrôleur (version JVCKT2) – un seul lecteur 360K – Touche CTRL enfoncée durant l'allumage, et de DE79H à F1C8H pour un MSX à un contrôleur – un ou deux lecteurs de 720K – démarrage normal.

L'occupation mémoire maximale étant de A98DH à F1C8H pour un MSX de huit unités de 720K – Version d'interface correcte – démarrage normal. Cette configuration ne laisse que 9345 octets libres pour le BASIC et très peu de billets dans votre portefeuille.

5.1.2 Initialisation du système MSX avec disquettes

La procédure d'initialisation des MSX est la suivante : l'ordinateur va d'abord rechercher si une ROM est présente en 4000H ou en 8000H en balayant chaque slot à partir du slot primaire 0 jusqu'au slot primaire 3. Si le slot primaire balayé est un slot étendu, il va aussi balalyer les quatre slots secondaires de ce slot primaire.

Si le système découvre une ROM (identifiable par la présence des lettres AB aux deux premières positions de cette ROM), il exécutera la routine d'initialisation de cette ROM dont l'adresse se trouve dans les deux positions qui suivent les lettres AB.

Si la ROM détectée est une ROM de contrôleur-disque, sa routine d'initialisation va réserver 439 positions RAM en dessous de la région de communication du BASIC (F380H). Ces 439 positions constituent la région de communication fixe des disques (F1C9H à F37FH).

Ensuite cette routine va réserver en dessous des 439 octets, x octets pour ce contrôleur-disque qui est appelé le contrôleur 0 puisqu'il est le premier détecté. Ces x octets forment le CWA0 (Controller Work Area 0, la zone de travail du contrôleur-disque). Pour la plupart des fabricants, la CWA a une longueur de 8 octets.

Dépendant de l'enfoncement de la touche CTRL, la routine d'initialisation va réserver sous la CWA0 un ou deux Disk Parameters Block (DPB) de 21 octets chacun ? Un DPB, ou bloc de paramètres disque, contient, comme nous le verrons plus loin, une série de paramètres de contrôle pour le lecteur dont il a la charge.

La routine d'initialisation de cette ROM étant terminée, il rend la main à la routine d'initialisation du Basic qui va poursuivre sa recherche de ROM dans le prochain slot secondaire ou primaire.

Si un scond contrôleur-disque est trouvé dans un autre solot, le même scenario se répète à l'exception de la réservation des 439 octets qui ont déjà été réservés par le contrôleur 0. Donc un nouveau CWA de x octets sera placé en dessous du ou des DPB du contrôleur précédent et, dépendant de la touche CTRL, un ou deux DPB seront installés pour le ou les disques de ce deuxième contrôleur.

Le même système se répète pour d'éventuels contrôleurs 2 et 3. Une fois cette première initialisation terminée, la routine d'installation du BASIC va rendre la main au contrôleur 0 pour que lui seul achève la procédure de réservation RAM.

Cette routine va ainsi réserver, sous les derniers DPB installés, trois tampons dont la longueur sera déterminée par la longueur du secteur le plus long des disques reconnus (presque toujours 512 octets, mais rappelez-vous que la longueur du secteur physique est fixée par le fabricant de l'interface et que dès lors on pourrait trouver des secteurs de longueur différente de 512 octets).

Ces trois tampons sont :

1	WORK AREA BUFFER	(WAB)	ou TAMPON DE TRAVAIL
1	SECTOR INPUT OUTPUT BUFFER	(SIOB)	ou TAMPON SECTEUR
1	DIRECTORY SECTOR BUFFER	(DSB)	ou TAMPON REPERTOIRE

Ensuite, cette même routine va réserver un TAMPON FAT par DISQUES LOGIQUES dont les longueurs respectives seront égales à la longueur du secteur de chacun de ces disques multipliée par un paramètre propre à chaque type de disques (dimension de la FAT en nombre de secteurs – voir annexe A).

5.1.3 Occupation mémoire commune MSX-DOS/MSX-Basic

Le diagramme suivant représente l'occupation de la mémoire depuis l'adresse FFFFH (Top of Memory) en descendant vers 0000H pour un système à huit disques de 720K (driver du Philips VG 8235). Ce diagramme est commun au Disk-BASIC et au MSX-DOS. Les zones marquées d'un astérisque peuvent être absentes si votre équipement ne contient pas huit disques. Il représente donc l'occupation mémoire maximale d'un MSX pour les disques.

Si, comme c'est habituellement le cas, vous n'avez qu'un lecteur et que vous allumez l'ordinateur normalement voyez le second diagramme quelques pages plus loin. Le sens des flèches indique dans quelle direction le système réserve la mémoire (du haut vers le bas ou du bas vers le haut).

```
DIAGRAMME D'OCCUPATION MAXIMUM.   ADRESSE TAILLE  COMMENTAIRE
+---------------------------------+ FFFF
| BASIC COMMUNICATION AREA        |        3200
+---------------------------------+ F380
| DISK COMMUNICATION FIXED AREA   |         439
+---------------------------------+ F1C9
| CONTROLLER WORK AREA 0          |           8   longueur
|                                 |               courante =8
|                                 |
+---------------------------------+ F1C1
| DISK PARAMETER BLOCK 1    *     |          21
+---------------------------------+ F1AC
| DISK PARAMETER BLOCK 0          |          21
+---------------------------------+ F197
| CONTROLLER WORK AREA 1    *     |           8   Voir CWA0
+---------------------------------+ F18F
| DISK PARAMETER BLOCK 3    *     |          21
+---------------------------------+ F17A
| DISK PARAMETER BLOCK 2    *     |          21
+---------------------------------+ F165
| CONTROLLER WORK AREA 2    *     |           8   Voir CWA0
+---------------------------------+ F15D
| DISK PARAMETER BLOCK 5    *     |          21
+---------------------------------+ F148
| DISK PARAMETER BLOCK 4    *     |          21
+---------------------------------+ F133
| CONTROLLER WORK AREA 3    *     |           8   Voir CWA0
+---------------------------------+ F12B
| DISK PARAMETER BLOCK 7    *     |          21
+---------------------------------+ F116
| DISK PARAMETER BLOCK 6    *     |          21
+---------------------------------+ F101
| SECTOR WORK AREA BUFFER         |           } Longueur 512 octets
+---------------------------------+ EF01
| SECTOR INPUT OUTPUT BUFFER      |
+---------------------------------+ ED01
| DIRECTORY SECTOR BUFFER         |
+---------------------------------+ EB01
| DISK 0 FAT BUFFER               |
+---------------------------------+ E500
| DISK 1 FAT BUFFER               |
+---------------------------------+ DEFF
| DISK 2 FAT BUFFER               |
+---------------------------------+ D8FE
| DISK 3 FAT BUFFER               |
+---------------------------------+ D2FD
| DISK 4 FAT BUFFER               |
+---------------------------------+ CCFC
| DISK 5 FAT BUFFER               |
+---------------------------------+ C6FB
| DISK 6 FAT BUFFER               |
+---------------------------------+ C0FA
| DISK 7 FAT BUFFER               |
+---------------------------------+ BAF9
```

Nous allons ici décrire brièvement ce tableau :

BASIC COMMUNICATION AREA :

Zone de communication du BASIC. Cette zone s'étend de F380H à FFFFH et existe dans tous les MSX avec ou sasn lecteur. Le BASIC normal n'étant pas l'objet de ce livre, je vous renvoie au Livre du MSX de Daniel MARTIN.

DISK COMMUNICATION FIXED AREA :

Zone fixe de communication pour les disques. Elle s'étend de F1C9H à F37FH. Elle sera décrite complètement et en détail à la fin de ce chapitre.

CONTROLLER WORK AREA 0 :

Zone de x octets toujours présente en-dessous de F1C9H qui comporte des informations nécessaires pour le driver (pilote du disque) des disques du contrôleur 0. Dans toutes les versions de driver que j'ai pu examiner, cette zone occupait huit octets et son contenu était identique (sauf dans les premières versions des lecteurs Sony où elle occupait 9 octets). Cependant la norme MSX ne prévoit ni longueur ni contenu.

DISK PARAMETER BLOCK 1 :

Zone de 21 octets toujours présente sauf si vous allumez l'ordinateur avec la touche CTRL enfoncée (et qu'un éventuel disque physique 1 soit éteint). Cette zone contient les paramètres physiques du deuxième disque logique du contrôleur 0 (disque B:) et est décrite en détail dans la suite de ce chapitre.

DISK PARAMETER BLOCK 0 :

Idem que ci-dessus mais toujours présente car il y a toujours au moins un disque connecté ; même s'il est éteint. Il contient les paramètres physiques du premier disque logique du contrôleur 0 (A:).

CONTROLLER WORK AREA 1-2-3 :

Idem que CWA0 mais ne sont présentes que s'il y a respectivement un deuxième, troisième et quatrième contrôleur-disque dans votre système.

DISK PARAMETER BLOCK 2 à 7 :

Idem que pour les DPB 0 et 1 ci-dessus excepté qu'il s'agit des paramètres physuiqyes du premier et du deuxième disques logiques connectés respectivement au deuxième, troisième et quatrième contrôleur-disque de votre système (disques C : D : E : F : G : H:).

SECTOR WORK AREA BUFFER :

Il s'agit d'une zone de travail de la longueur du plus grand secteur de vos disques utilisée lorsque le tampon secteur est occupé ; notamment pour les transferts vers la RAM 4000H.

SECTOR INPUT-OUTPUT BUFFER :

Il s'agit du tampon de lecture/écriture de tout secteur normal du disque en BASIC (secteur normal signifie qu'il n'a pas une vocation spéciale comme les secteurs du Directory ou de la FAT). Il a un longueur identique au WORK AREA BUFFER.

DIRECTORY SECTOR BUFFER :

C’est un tampon de longueur identique au WORK AREA BUFFER réservé pour la lecture/écriture des secteurs du Directory.

FAT BUFFER 0 à 7 :

Il peut y avoir jusqu’à 8 tampons FAT (un par disque logique). Leur longueur dépend de chaque type de disque (voir remarque dans le tableau d’occupation mémoire maximum). Le système y maintient une copie des secteurs FAT du disque et y fait toutes les recherches ou modifications d’allocations de clusters pour les fichiers. Détruire cette zone équivaut à détruire son disque si le travail avec ce disque se poursuit.

VERSION DE CONTROLEUR				DIAGRAMME D'OCCUPATION MEMOIRE
JVCKT2		NORMALE		1 CONTROLEUR-DISQUES 360/720 K
NORM	CTRL	NORM	CTRL	Allumage ordinateur (CTRL tenu)
FFFF F380	FFFF F380	FFFF F380	FFFF F380	REGION DE COMMUNICATION DU BASIC
F37F F1C9	F37F F1C9	F37F F1C9	F37F F1C9	REGION DE COMMUNICATION FIXE DES DISQUES
F1C8 F1C1	F1C8 F1C1	F1C8 F1C1	F1C8 F1C1	ZONE DE TRAVAIL DU CONTROLEUR 0 (CWA0)
F1C0 F1AC		F1C0 F1AC		BLOC DE PARAMETRES DU DISQUE B: (DPB1)
F1AB F197	F1C0 F1AC	F1AB F197	F1C0 F1AC	BLOC DE PARAMETRES DU DISQUE A: (DPB0)
F196 EF97	F1AB EFAC	F196 EF97	F1AB EFAC	TAMPON DE TRAVAIL (SWA)
EF96 ED97	EFAB EDAC	EF96 ED97	EFAB EDAC	TAMPON D'ENTREE/SORTIE SECTEUR (SIOB)
ED96 EB97	EDAB EBAC	ED96 EB97	EDAB EBAC	TAMPON SECTEUR REPERTOIRE (DSB)
EB96 E796	EBAB E7AB	EB96 E596	EBAB E5AB	TAMPON FAT DU DISQUE A: (FAT0)
E795 E395		E595 DF95		TAMPON FAT DU DISQUE B: (FAT1)
24456	25502	23432	24990	NOMBRE D'OCTETS LIBRES BASIC

5.1.4 Différence entre les occupations mémoires MSX-DOS et Disk-BASIC

Les deux premiers tableaux représentaient la carte mémoire commune au Disk-BASIC et au MSX-DOS. Le reste de la réservation mémoire va dépendre du choix que vous ferez de travailler en Disk-BASIC ou en MSX-DOS.

Au cas où la disquette, insérée dans le disque A : lors de l'allumage de l'ordinateur ne contient pas le fichier MSXDOS.SYS, le système ne vous laissera pas le choix et vous imposera de travailler en Disk-BASIC.

Si votre disquette contient le fichier MSXDOS.SYS et le fichier COMMAND.COM, ce sera le MSX-DOS qui sera activé et le message « MSX-DOS version 1.03... » apparaîtra suivi de l'indicatif du mode commande A> ; vous pourrez alors choisir d'appeler un programme MSX-DOS ou taper une commande comme DIR, DATE, TYPE etc ou encore la commande BASIC qui vous renverra vers le Disk-BASIC.

Voici d'abord le tableau reprenant la suite de la carte mémoire. Ce tableau se situe juste au-dessous du dernier tampon FAT.

<table>
<tr><th>Disk-BASIC</th><th>MSX-DOS</th></tr>
<tr><td>Zone de 7*37 octets réservée pour les 7 File Control Block (FCB0 à FCB6)</td><td>Zone de 648 octets (JVCKT2=645) contenant les routines de commutation des slots</td></tr>
<tr><td>Routine en langage machine utilisée par les instructions BLOAD et BSAVE</td><td>Routine destinée aux fonctions du MSX-DOS (Fonctions 00H à 30H)</td></tr>
<tr><td>Zone que vous pouvez réserver pour vos programmes en langage machine par l'instruction CLEAR et qui s'étend vers le bas</td><td>Zone de pile utilisateur (stack) par défaut à l'appel d'un programme MSX-DOS. Elle peut être recouverte par le programme utilisateur</td></tr>
<tr><td>Zone des tampons de fichiers. Le nombre de tampons dépend de MAXFILES. Le tampon réservé au fichier 0 est le plus bas. Cette zone s'agrandit vers le bas.</td><td>Routine des commandes du MSX-DOS (DIR, TYPE, COPY, DEL…). Cette zone peut être recouverte par votre programme si nécessaire.</td></tr>
<tr><td>Zone réservée au stockage de vos variables « chaîne » de 200 octets par défaut mais ajustable par CLEAR et qui s'étend vers le bas.</td><td rowspan="6">ZONE
RESERVEE
AU PROGRAMME
UTILISATEUR</td></tr>
<tr><td>Zone de la pile (Stack) et qui s'étend vers le bas.</td></tr>
<tr><td>Zone libre. Quand cette sone sera nulle, Out of memory sera affiché.</td></tr>
<tr><td>Zone des variables dimensionnées qui s'étend vers le haut.</td></tr>
<tr><td>Zone des variables simples qui s'étend vers le haut.</td></tr>
<tr><td>Zone du texte du programme BASIC qui s'étend de 8000H vers le haut.</td></tr>
<tr><td>ROM de 16K contenant le BASIC de l'adresse 4000H à 7FFFH.</td><td>Page 0 du MSX-DOS (0000H - 0100H)</td></tr>
</table>

Commençons par analyser la partie gauche du tableau réservé au BASIC.

FCB0-FCB6 :

Il s’agit de 7 zones de 37 caractères. Les 7 File Control Blocks (ou blocs de contrôle de fichier) contiennent des informations sur les fichiers ouverts. Le FCB est décrit plus loin dans ce chapitre. Le FCB0 est réservé aux commandes BASIC SAVE/LOAD, BSAVE/BLOAD et MERGE tandis que les FCB 1 à 6 sont réservés pour les fichiers ouverts sous les numéros 1 à 6. Le FCB0 est l’adresse la plus basse et le FCB6 à l’adresse la plus haute.

Routines BLOAD/BSAVE :

Il s’agit d’un petit morceau de code machine utilisé par les commandes BLOAD et BSAVE du BASIC. Ces routines sont indiquées ici parce qu’elles ont la particularité de s’implanter à la plus basse adresse réservée pour les disques en BASIC.

Tout ce qui suit est propre au BASIC seul et non pas au Disk-BASIC. Ces zones ont été notées ici pour compléter la carte mémoire.

Reserved Area :

Zone que le programmeur BASIC peut se réserver pour implanter ses propres routines en langage machine grâce à l’instruction CLEAR 200, &Hxxxx ; xxxx représente la plus haute adresse que le BASIC pourra employer ; donc, &Hxxxx+1 représente la plus basse adresse réservée pour votre programme en langage machine. Votre programme pourra s’étendre jusqu’à l’adresse des routines BLOAD/BSAVE décrites ci-dessus.

Files Area :

L’instruction MAXFILES sert à indiquer combien de fichiers peuvent être ouverts au même moment. Elle réserve un tampon pour chacun d’eux. Ce tampon est constitué d’un entête de 9 octets suivi du tampon fichier proprement dit de 256 octets. L’ensemble de ces tampons est précédé d’une table de pointeurs indiquant l’adresse de départ de l’entête de chacun de ces tampons. Cette table est constituée de deux octets par fichier.

Literal String Pool Area

Zone de stockage du contenu des variables de type chaîne de caractères dont la longueur est fixée par l’instruction CLEAR xxx, où xxx est la longueur de cette zone. Cette zone a une longueur de 200 octets par défaut.

Stack Area :

Zone de mémoire appelée Pile servant au microprocesseur et à sauvegarder les adresses de retour des instructions GOSUB et des boucles FOR-NEXT notamment.

Toutes les zones que nous avons vues depuis l’adresse FFFFH et dont la longueur est variable s’étendent vers le bas de la mémoire.

Par contre, les zones suivantes grandissent vers le haut de la mémoire. Il est donc possible que la zone de la pile finisse par rencontrer la plus haute des zones suivantes.

Dans ce cas, BASIC affichera le message « Out of memory » indiquant par là que votre programme ou divers paramètres comme la réservation de l’espace chaîne (CLEAR) ou du nombre de fichiers

(MAXFILES) ou de l'espace réservé pour les programmes en langage machine (CLEAR, &Hxxxx) sont trop grands.

Basic Program Area :

C'est la zone où s'installe votre programme BASIC. Elle commence généralement en 8001H mais elle peut être changée en modifiant un pointeur de la région de communication du BASIC (F676H).

Simple Variable Table :

La table des variables simples (non dimensionnées) suit immédiatement la fin de votre programme. Elle est constituée uniquement lors de la rencontre de chaque variable au fur et à mesure de leur apparition en cours d'exécution de votre programme.

Donc cette zone s'agrandit (vers le haut) pendant l'exécution de votre programme. Cet agrandissement constitue un facteur de ralentissement de votre programme. En effet, la table des variables dimensionnées qui la suit devra être « déménagée » plus loin en mémoire lors de chaque apparition d'une nouvelle variable simple.

De ce fait, il est vivement conseillé de définir toutes les variables simples avant toute définition de variable dimensionnée.

Dimensioned Variable Table :

C'est la table des variables dimensionnées encore appelées variables tableau ou variables indicées.

Free Space :

C'est l'espace mémoire inemployé lors de l'exécution d'un programme donné. Si cette zone devient nulle, le message « Out of memory » est affiché (voir remarque ci-dessus).

5.1.5 Zone mémoire réservée au MSX-DOS

Voici maintenant une description de la mémoire réservée pour le MSX-DOS. Sous le dernier tampon de la FAT s'installent les zones suivantes en décroissant vers l'adresse 0000H.

COMMUTATION DE SLOT :

il s'agit essentiellement des routines d'interruption, de lecture et d'écriture dans un autre slot, de sélection de slot et de saut dans un autre slot.

FONCTIONS MXS-DOS :

Il s'agit des routines qui permettent de traiter les 42 fonctions offertes par le MSX-DOS et décrites dans le chapitre 8. Cette zone mémoire sert généralement de relais avec la ROM su contrôleur-disque qui contient les routines proprement dites de ces fonctions.

ZONE DE LA PILE (STACK) :

C'est l'emplacement de la pile (stack) réservée au programme qui tourne sous MSX-DOS. On lui a réservé une taille de 256 octets. Si vous comptez utiliser les commandes du MSX-DOS à l'intérieur

même de votre programme et que celui-ci fait un usage important de la pile, il serait préférable de déplacer cette pile sous la zone des commandes du MSX-DOS afin de ne pas écraser le dessus de cette zone par une pile qui s'étendrait trop vers le bas.

COMMANDES DU MSX-DOS :

C'est la zone où se trouvent les commandes du MSX-DOS (DIR, TYPE, DATE…). Cette zone occupe 1050H octets dans la version 1.11 de COMMAND.COM mais est susceptible de changer de longueur dans d'autres versions. Elle a l'avantage de pouvoir être recouverte par votre programme ; en effet, elle se rechargera automatiquement dès la fin de votre programme.

ZONE DU PROGRAMME :

C'est à cet endroit que s'implantent vos programmes MSX-DOS en langage machine. Cette zone commence à l'adresse 0100H et peut s'étendre jusqu'à la zone des fonctions du MSX-DOS si vous avez prévu une zone de pile interne à votre programme ; sinon, elle est limitée par la zone de la pile ou par la zone des commandes du MSX-DOS.

Pour vous donner une idée de la taille de cette zone, nous supposerons que vous avez un seul contrôleur-disque et que vous avez allumé le système normalement. Dans ce cas, vous disposerez de 54221 octets. Justice est ainsi rendue à ceux qui se plaignent des 23432 octets du BASIC puisqu'ils disposent de 2,3 fois plus de mémoire…

Sur un MSX2 avec 128K de RAM utilisateur tel que le VG8235 de Philips, ils disposeront de 65536 octets supplémentaires, mais uniquement par l'emploi du MEMORY MAPPER décrit au chapitre 9, ce qui leur offre 119757 octets réservés à leur programme ; ce qui n'est pas mal du tout pour un ordinateur familial à base de Z80 !

MSX-DOS PAGE 0 :

Les 256 premiers caractères de RAM à l'adresse 0 sont réservés pour le MSX-DOS. Elle sera détaillée plus loin. Sachez qu'on y trouve des points d'entrée, les FCB par défaut et la DTA (Disk Transfer Address) par défaut.

5.2 Table des contrôleurs des unités de disquettes

Cette table est implantée dans la région de communication du BASIC à l'adresse FB21H de telle sorte que même un programme chargé depuis une cassette puisse savoir si le système où il s'est chargé est équipé de disque ou non.

La table est constituée de deux octets par contrôleur soit huit octets. Le premier octet de la paire indique le nombre (0, 1 ou 2) de disques logiques (et non pas physiques) connectés au contrôleur. Le deuxième octet donne le caractère de sélection du slot où est établi ce contrôleur sous la forme :

E 0 0 0 S S P P où PP est le numéro de slot primaire
SS est le numéro de slot decondaire (si E=1)
E=1 si le slot primaire est étendu en secondaires
E=0 si le slot primaire n'est pas étendu

FB21 : Nombre de disques logiques du contrôleur 0

FB22 : Slot du contrôleur 0
FB23 : Nombre de disques logiques du contrôleur 1
FB24 : Slot du contrôleur 1
FB25 : Nombre de disques logiques du contrôleur 2
FB26 : Slot du contrôleur 2
FB27 : Nombre de disques logiques du contrôleur 3
FB28 : Slot du contrôleur 3

L'intérêt de cette table est d'abord qu'elle permet de connaître le nombre de disques implantés par contrôleur mais surtout de vous indiquer dans quel slot se trouve un contrôleur donné.

Vous verrez dans le chapitre 6 qu'il est nécessaire de connaître ce slot si vous voulez exploiter les points d'entrée des ROM des contrôleurs-disques. De plus, le code indiquant le slot du contrôleur est directement assimilable par les routines de manipulation de slots.

5.3 Table des zones de travail réservées par la ROM Disk

Cette table est aussi implantée dans la région de communication du BASIC de l'adresse FD09 à FD88 et a donc une longueur de 128 octets. Elle est constituée de deux octets par ROM qu'il est possible d'installer dans un MSX.

Comme la norme MSX prévoit que la mémoire peut être partagée en 4 slots primaires, eux-même partagés en 4 slots secondaires et contenant chacun 4 blocs de 16K, nous arrivons au total de 64 ROM possibles d'où cette longueur de table de 128 octets.

Les deux octets réservés par ROM sont garantis par la norme MSX comme étant à l'usage exclusif de cette ROM.

Pour trouver l'adresse des deux octets spécifiquement réservés à un contrôleur-disque, il suffit d'employer la formule suivante :

Adresse = FD09 + (32 * SP) + (8 * SS) + (2 * BK)
où
SP est le numéro du slot primaire
SS est le numéro du slot secondaire
BK est le numéro du Bank (0=000H 1=4000H 2=8000H 3=C000H)

Dans le cas d'un contrôleur-disque le numéro du Bank est toujours 1 soit 4000H. À cette adresse se trouvent donc les deux octets exclusivement réservés au contrôleur-disque présent dans le slot secondaire SS du slot primaire SP.

Dans notre cas ces deux octets forment un pointeur d'une zone mémoire RAM où se trouve la Controller Work Area (zone de travail du contrôleur). Le contenu de la CWA dépend du fabricant de l'interface et sera décrit plus loin dans ce chapitre.

5.4 Le Disk Parameter Block

Il y a autant de DPB (bloc de paramètres du disque) qu'il y a de disques reconnus par le système.

Un DPB n’est pas nécessairement attaché à un disque physique ; ainsi lorsque nous n’avez qu’un seul disque physique, vous disposez malgré tout de deux disques logiques appelés A : et B : (voir chapitre 2) et donc il y a 2 DPB.

L’adresse d’un DPB en mémoire est donné par son pointeur qui se trouve dans la table des DPB à l’adresse F355. Pour trouver l’adresse du DPB0 (Disque A:), il suffit de lire les deux octets présents dans TABLE + (2 * N° disque) soit F355 + (2*0) = F355.

Si vous cherchez le DPB3 (disque D:), lisez les deux octets présents à l’adresse F355 + (2*3) = F35B. D’autre part, comme il n’y a jamais qu’un seul disque activé à la fois, il est possible de trouver le DPB du dernier disque activé en prenant le contenu du pointeur F243 (DPB en activité).

Un DPB est constitué de 21 octets donnant des informations sur les caractéristiques physiques ou structurelles de son disque. Voici sa description :

Position	Nom	Description de la zone
0	DR	Drive number : (0 à 7 = A : à H:) auquel est attaché ce DPB
1	MEDIA	Type de disque encodé comme suit : 11111TSF T=0 : 80 pistes S=0 : 9 secteurs/piste F=0 : simple face T=1 : 40 pistes S=1 : 8 secteurs/piste F=1 : double face
2-3	SECSIZ	Longueur physique du secteur en octets
4	DIRMSK	DIRectory MaSK : masque qui permet d’isoler la position d’un fichier dans un secteur Directory d’après sa position globale.
5	DIRSHFT	DIRectory ShiFT : puissance de 2 indiquant le nombre de fichiers dans un secteur Directory. 2^(DIRSHFT
6	CLUSMSK	CLUSter MaSK : masque destiné à connaître le secteur d’un cluster qui est utilisé (vaut CLUSSHFT - 1)
7	CLUSSHFT	Indique le nombre de secteurs d’un cluster
8-9	FIRFAT	FIRst FAT sector : en 16 bits inversés le numéro du premier secteur de la FAT sur le disque
10	FATCNT	FAT CouNT : nombre de FAT du disque
11	MAXENT	MAXimum ENTry : nombre maximum de fichiers que l’ont peut placer dans le Directory
12-13	FIRREC	FIRst RECord : indique en 16 bits inversés le numéro du premier secteur du premier fichier
14-15	MAXCLUS	MAXimum CLUSter : donne en 16 bits inversés la capacité utilisateur maximum du disque en nombre de clusters
16	FATSIZ	FAT SIZe : donne la dimension de la FAT en secteurs
17-18	FIRDIR	FIRst DIRectory sector : donne en 16 bits inversés le numéro du premier secteur du Directory
19-20	FATPTR	FAT PoinTeR : donne en 16 bits inversés l’adresse mémoire du tampon FAT. Il a une longueur de (FATSIZ * SECSIZ)

5.5 Le Controler Work Area

Il y a autant de CWA (zone de travail pour contrôleur-disque) que d'interfaces pour disques. La présence de cette zone n'est pas rendue obligatoire par la norme MSX et est laissée à l'appréciation du fabricant du contrôleur et du disk-driver (c'est-à-dire le programme de pilotage du hardware du disque). Cependant, dans tous les disk-drivers que j'ai pu analyser cette zone était toujours présente avec une longueur de huit octets.

Position	Nom	Description de l'octet
0	MOT	MOTor timer : chaque fois qu'un des deux disques de ce contrôleur est commandé, le moteur des deux disques est activé. Lorsque l'opération de lecture ou d'écriture est terminée, le chronomètre est déclenché pour 1,4 s. Au bout de ce temps, les moteurs sont arrêtés. Si, à la fin d'une opération disque, vous placez la valeur FF dans cette position, alors les moteurs ne s'arrêteront pas sauf si un autre accès re-déclenche le chronomètre pour 1,4s. Le chrono ne fonctionne que si les interruptions sont autorisées et que le hook FD9F n'est pas modifié par votre programme.
1	TD0	Timer Disk 0 : chronomètre du disque physique 0. Ce chrono est déclenché pour 0,7s à chaque opération sur ce disque. Si, lors d'une recherche dans le Directory, ce chronomètre est tombé à 0, le système chargera d'abord les secteurs de la FAT en mémoire et, en fonction du type de disque indiqué dans le premier octet de la FAT, va établir le DPB correspondant à ce type de disque. Cette opération est nécessaire au cas où la disquette a été changée à l'insu du programme. Si le chrono n'a pas atteint zéro, le système est sûr que la disquette n'a pas été changée. Essayez donc de changer une disquette en moins de 0,7s.
2	TD1	Timer Disk 1 : comme ci-dessus pour le disque 1 (B:)
3	SDK	Selected DisK : denier disque sélectionné
4	TK0	TracK disk 0 : mémorise sur quelle piste le disque physique 0 se trouve quand l'on travaille avec le disque physique 1
5	TK1	TracK disk 1 : comme ci-dessus pour le disque 1
6	SLD	Selected Logical Disk : dernier disque logique sélectionné. Cette position permet de savoir à quel disque logique est attribué la disquette présente dans le disque physique dans les systèmes à un disque physique.
7	DCT	Disk CounT : nombre de disques physiques détectés.

5.6 Le File Control Block

Le File Control Block (bloc de contrôle du fichier) est un dispositif capital dans la manipulation des fichiers par le langage machine.

C'est une zone de 37 octets qui contient toutes les informations nécessaires à la manipulation correcte d'un fichier ouvert telles que son nom, l'unité sur laquelle le fichier se trouve, son emplacement sur la disquette, à quel endroit du fichier la prochaine lecture ou écriture doit se faire, etc.

Il existe deux types de FCB supportés par le MSX. Le MSX-DOS a en effet été ainsi conçu qu'il supporte une grande proportion de programmes écrits sous CP/M de même, bien entendu, que les programmes écrits spécifiquement pour le MSX-DOS.

Pour cette raison, nous examinerons d'abord le FCB employé en CP/M et puis celui du MSX-DOS. Le FCB utilisé en Disk-Basic étant identique à celui du MSX-DOS, vous n'en trouverez pas de description séparée.

Lorsqu'un programme en langage machine doit accéder à un fichier, il devrai, comme en BASIC, passer par trois phases :

1 – l'ouverture du fichier
2 – la lecture ou l'écriture du fichier
3 – la fermeture du fichier

En BASIC, c'est à l'ouverture que toutes les informations nécessaires au BASIC seront spécifiées comme dans l'instruction suivante :

OPEN « B:ADRESSE.TXT » AS #1 LEN=128

Cette instruction préciser qu'il faut ouvrir le fichier ADRESSE.TXT sur l'unité B : en 'mode accès direct' avec une taille d'enregistrement de 128 octets sous le numéro de fichier 1. Par la suite, les phases 2 et 3 c'est-à-dire la manipulation du fichier et sa fermeture, se feront par référence au numéro de fichier 1.

En CP/M ou en MSX-DOS, les 3 phases se feront non pas par référence à un numéro de fichier mais par référence à un FCB dont le programmeur en langage machine devra simplement indiquer l'adresse.

Les FCB CP/M et MSX-DOS doivent donc être établis par le programmeur avant l'ouverture du fichier. L'opération d'ouverture du fichier complétera ensuite le FCB établi par le programmeur en y ajoutant des éléments trouvés dans l'entrée du Directory correspondant au fichier.

Il y a deux différences majeures entre CP/M et MSX-DOS :

1) En CP/M, le record a une taille fixe de 128 octets. En MSX-DOS, la taille du record est programmable de 1 à 65535 octets.

2) En CP/M, l'allocation d'espace sur le disque pour un fichier se fait par un Extent (voir chapitre 4.4). Dans le cas du MSX exécutant un programme CP/M, l'extent représente 128 records (128 records de 128 octets soit 84163). La notion d'extent doit être connue si l'on veut programmer en CP/M et se positionner n'importe où dans un fichier de plus de 16K. Par contre en MSX-DOS, la notion d'Extent n'existe pas et des fonctions spéciales et propres au MSX-DOS permettront l'accès direct à n'importe quelle fraction d'un fichier.

5.6.1 Le FCB du CP/M

Le FCB du CP/M est constitué de 36 octets dont voici le découpage. Les 16 premiers octets plus l'octet 32 doivent être remplis par le programmeur avant un OPEN.

Position	Longueur	Nom	Description de la zone
0	1	DR	Drive : le programmeur indique ici le n° du disque sur lequel se trouve le fichier 0 → disque par défaut (1 à 8 pour A à H)
1	8	FNAME	File NAME : le programmeur indique ici la première partie du nom de son fichier. Le nom sera aligné à gauche et la zone sera complétée d'espaces (20H)
9	3	FEXT	File EXTension : extension du nom de fichier. l'extension sera alignée à gauche et complétées d'espaces (20H)
12	1	EX	Contient le n° d'extent actuel
13	1	S1	Cet octet contient, dans le bit 0, un bit d'extension de la zone EX précédente pour que la capacité totale d'un fichier CP/M puisse atteindre 512 extents de 16384 octets soit 8 Megaoctets. Le programmeur doit y mettre 00 avant un OPEN
14	1	S2	Réservé au système. Mettre 0 avant OPEN.
15	1	RC	Record Count : indique après un OPEN le nombre de records qui se trouvent dans l'extent EX (de 1 à 128). Le programmeur doit mettre 00 dans cett ezone avant OPEN
16	4	FILSIZ	FIL SIZe : indique la longueur du fichier
20	2	DATE	Date de création ou de dernière modification du fichier OCTET 21 OCTET 20 AAAAAAAM MMMJJJJJ A=année-1980, M=mois, J=jour
22	2	TIME	Heure de création ou de dernière modification du fichier OCTET 23 OCTET 22 HHHHHMMM MMMSSSSS
24	1	DEVID	DEVice Identification : n° du périphérique sur lequel le fichier se trouve. Forme : 0 E 0 0 0 D D D DDD=N° du disque (0-7) 1 E 1 1 P P P P PPPP=périphérique F = CON (CONsole) E = AUX (AUXiliary)

			D = NUL (NUL device) C = LST (LiST=imprimante) B = PRN (PriNter ‘’ ‘’ ‘’) E = 0 si fichier modifié depuis l’OPEN E= 1 si le fichier n’a pas été modifié
25	1	DIRLOC	DIRectory LOCation : position du fichier dans le Directory sous la forme d’un N° d’entrée (00-6F)
26	2	STRCLS	STaRt CluSter : donne en 16 bits inversés le n° du premier cluster du fichier
28	2	CURCLS	CURrent CluSter : en 16 bits inversés le n° du cluster accédé en dernier lieu
30	2	CLSOFF	CluSter OFFset : en 16 bits inversés le dernier cluster accédé relativement au début du fichier
32	1	CR	Current Record : n° du record (0-127) de l’extent EX à accéder. Cette position est automatiquement incrémentée à chaque accès séquentiel. A 128, elle est remise à 0 et EX est incrémenté
33	3	RN	Record Number : employé pour les fichiers directs. Contient le numéro du record à accéder. L’octet 35 est toujours à 0

Donc lorsqu’un programmeur veut accéder à un fichier, il remplira d’abord les 16 premiers octets du FCB avec le numéro du disque, le nom du fichier, son extension et généralement complétera la zone de 16 octets avec 00H.

Ensuite, il procédera à l’ouverture du fichier.

Pour accéder aux enregistrements d’un fichier séquentiel, il lui suffira d’utiliser les fonctions CP/M READ SEQUENTIAL (14H) et WRITE SEQUENTIAL (15H) sans toucher au FCB, s’il désire accéder aux enregistrements en séquence.

S’il désire accéder aux enregistrements hors séquence, il devra préciser le numéro de l’enregistrement dans l’octet CR et le numéro de l’extent dans l’octet EX avant chaque READ ou WRITE SEQUENTIAL.

Pour accéder aux enregistrements d’un fichier à accès direct, il devra préciser le numéro de l’enregistrement dans les octets RN puis utiliser les fonctions CP/M READ ou WRITE RANDOM (21H et 22H) et cela avant chaque accès. Finalement, le programmeur procédera à la fermeture du fichier.

Les octets 16 à 31 peuvent être consultés par le programmeur mais pas modifiés. Ils ne sont pas identiques au vrai CP/M mais comme ils sont réservés au CP/M pour ses besoins propres et donc jamais employés par les programmes qui tournent sous CP/M, il n’y a pas d’incompatibilité.

5.6.2 Le FCB du MSX-DOS et du Disk-BASIC

Le FCB du MSX-DOS est complètement identique au FCB du MS-DOS qui est le système d'exploitation disque des IBM-PC et des compatibles à la différence près qu'il ne supporte pas les « expanded FCB » ou FCB étendu du MS-DOS.

Il est constitué de 37 octets, soit un de plus que dans le FCB CP/M. Seuls quelques octets diffèrent d'ailleurs entr eles deux types de FCB. Cela est du aux différences entre les systèmes d'allocation d'espace-disque. Seules les zones différentes du CP/M sont indiquées.

Zone RECSIZ : Position 13 : Longueur 2 octets

Pour les fonctions MSX-DOS 26H et 27H, on choisit la longueur des enregistrements en installant avant un accès cette longueur dans RECSIZ. Cette valeur doit être établie en 16 bits inversés et peut couvrir une gamme de 1 à 65535 octets.

Zone RN : Position 33 : Longueur 4 octets

Record number : cette zone n'est employée que par les fonctions MSX-DOS 26H et 27H et par les fonctions 21H et 22H du CP/M. Dans le cas des fonctions MSX-DOS, elle est étendue à 4 octets dont tous sont valides si la longueur de l'enregistrement (RECSIZ) est plus petite que 64 octets, ou dont les 3 premiers sont valides si cette longueur est plus grande que 63 octets. Cette zone permet de préciser le n° de l'enregistrement que l'on veut accéder par les fonctions 26H et 27H. Après une fonction 26H ou 27H, cette zone est automatiquement incrémentée.

Donc, le FCB MSX-DOS ou Disk-BASIC est tout à fait identique au FCB CP/M pour les fonctions MSX-DOS dites compatibles CP/M.

Pour les fonctions disque non compatibles CP/M, le FCB est modifié pour supporter une zone qui fixe la longueur de l'enregistrement (RECSIZ – octets 14 et 15) et pour agrandir la zone qui fixe le numéro d'enregistrement en accès direct (RN – octet 33 à 36).

En effet, si on peut fixer une longueur d'enregistrement de 1 octet, alors les 4 octets RN réservés maintenant en MSX-DOS permettent un accès direct à n'importe quel octet d'un fichier de 4294K soit Megaoctets.

6) Chapitre 6 : les points d'entrée du Disk-ROM

Il existe actuellement toute une série de contrôleurs de disquettes. On peut cependant les classer en 4 grandes catégories :

a) Les contrôleurs de disquettes 360K datant du MSX1
b) Les contrôleurs de disquettes 720K datant du MSX1
c) Les contrôleurs de disquettes 360K datant du MSX2
d) Les contrôleurs de disquettes 720K datant du MSX2

La différence essentielle entre les versions MSX1 et MSX2 se trouve dans le fait que de nouveaux points d'entrée ont été corrigés dans la version MSX2.

A part cela, les versions 360K appellent également quelques remarques. On les appelle versions 360K parce qu'lles sont fournies avec une unité de disquette de 360K, mais certaines d'entre elles

supportent également la connexion de disquettes 720K alors que d'autres ne les supportent pas. Il est donc primordial de pouvoir distinguer ces deux versions et d'en connaître les avantages et inconvénients.

Les versions 360K qui affichent 24546 ou 24455 octets libres lors d'un PRINT FRE(0) sont des versions qui ne supportent pas les disquettes de 720K. Par contre les versions qui affichent 23432 octets libres après un PRINT FRE (0) supportent les disquettes de 720K.

Les avantages et inconvénients des deux versions sont exactement opposées : l'un ne supporte pas les disquettes de 720K mais offre 104 octets de plus et vice-versa pour l'autre.

En dehors de cela, il est un aspect plus difficile à identifier : c'est la vitesse effective de lecture et d'écriture sur la disquette. Elle ne dépend pas du classement dans l'une des quatre catégories précédentes mais du fabricant de ce contrôleur.

En effet, rappelons ici que la ROM interne d'un contrôleur contient trois modules :

- Le Disk-BASIC ;

- Le kernel MSX-DOS programmé par Microsoft ;

- Le driver (pilote) du lecteur, programmé par le fabricant de ce contrôleur.

Or la vitesse dépend essentiellement de ce driver et donc varie d'un fabricant à l'autre ; reconnaissons cependant que les versions MSX2 sont, généralement, plus rapides que les versions MSX1.

Pour pouvoir utiliser les points d'entrée du Disk-ROM qui est situé en page 1 c'est-à-dire à l'adresse 4000H, il faut fatalement sélectionner le slot où est située cette ROM. D'autre part, si nous avons plusieurs contrôleurs, ils seront nécessairement disposés dans des slots différents et demanderont donc une sélection spécifique pour pouvoir manipuler les unités de disquettes de ces contrôleurs.

Comment donc sélectionner le Disk-ROM qui nous intéresse ? Heureusement pour nous, une zone de mémoire RAM a été initialisée à l'allumage du système pour nous indiquer dans quel slot se trouve chacun des quatre contrôleurs possibles et combien de disquettes logiques ils manipulent chacun.

Cette zone de mémoire est située de l'adresse FB21H à FB28H. Bien entendu, cette zone n'est valide que si votre système contient au moins un contrôleur. Pour rendre vous programmes compatibles sur tout système, il suffit de vérifier que le hook FFA7H ne contienne pas la valeur C9H : dans ce cas, aucun contrôleur n'est installé.

6.1 4010H DSKIO DiSK Input Output

Ce point d'entrée permet de lire et écrire de 1 à 255 secteurs en un appel de la routine à partir du secteur spécifié sur une unité de disquette de type et de numéro déterminé.

En entrée :

CF Le carry flag doit être à 0 pour lire

Le carry flag doit être à 1 pour écrire

A Le numéro de l'unité logique choisie (0 ou 1) du contrôleur dont la ROM est actuellement sélectionnée. Un contrôleur ne supporte que deux unités qui sont toujours numérotées 0 ou 1.

B Nombre de secteurs (1 à 255) à transférer de/vers la mémoire. Ce nombre doit tenir compte de la taille de la mémoire réservée pour stocker les secteurs. En effet, si vous demandez le transfert de 100 secteurs de 512 octets, il faudra 51200 positions de mémoire pour les stocker et vous devrez donc avoir prévu la place nécessaire.

C Code du type de disquette (F8 à FF voir annexe A). Attention ! Certains contrôleurs ne supportent plus les types FC à FF réservés pour les disquettes de 5''1/4.

DE Numéro du premier secteur à transférer (voir annexe A pour les numéros de secteurs maximum suivant les types de disques – pour un disque de 360K, la gamme s'étend de 0 à 2CFH, pour un disque de 720K, la gamme s'étend de 0 à 59FH).

HL Contient l'adresse de la zone de mémoire de réception/émission du/des secteurs. Les secteurs lus se trouveront à l'adresse HL ou le contenu de l'adresse HL sera écrit sur le disque suivant l'état du Carry flag.

En sortie :

CF 0 si pas d'erreur
1 si lecture/écriture erronée.

A Si le Carry flag est à 1, A contient un des codes d'erreur suivants :
0 : la disquette est protégée contre l'écriture lors d'une tentative d'écriture (write protect).
2 : pas de disquette dans le lecteur ou lecteur non allumé ou lecteur inexistant (disk offline).
4 : mauvaise lecture des données (CRC error)
6 : erreur de positionnement de la tête de lecture (seek error)
8 : secteur non trouvé (header bot found : nécessite un reformatage)
A : mauvaise écriture (Write current ou RAW error)
C : autres types d'erreurs non spécifiées (ex : DMA Time out)

Registres affectés : AF, BC, DE, HL, IX, IY

Attention ! Il s'agit ici de lecture ou d'écriture physique. L'adressage du disque se fait par numéro de secteur et non par numéro d'enregistrement dans un fichier. D'autre part, ce point d'entrée est avantageusement remplacé par le CALL BIOS 144H de la ROM BASIC où tous les paramètres sont identiques sauf A qui représente les unités logiques 0 à 7. En effet, la routine détermine elle-même quel contrôleur choisir d'après le numéro de disque 0 à 7.

Exemple :

Lire les secteurs 5 à 11 (Directory) du disque C : (360K) en C100H de la mémoire. Nous intégrerons cette routine dans un programme BASIC pour afficher tous les fichiers du Directory.

```
ORG     C000
```

```
C000  3E 2          LD   A, 2          ; A = 2 = Disque C :
C002  21 21 FB      LD   HL, 0FB21     ; table contrôleurs
C005  96      RPT : SUB  (HL)          ;
C006  3B 4          JR   C, CONT       ;
C008  23            INC  HL            ;
C009  23            INC  HL            ; Sélectionne
C00A  A8 F9         JR   RPT           ; le slot du contrôleur
C00C  86      CONT: ADD  A, (HL)       ; du disque C :
C00D  F5            PUSH AF            ; A indique si C est
C00E  23            INC  HL            ; le premier (0) ou
C00F  7E            LD   A, (HL)       ; le second disque
C010  26 40         LD   H, 40         ;
C012  CD 24 00      CALL 24            ;
C015  FB            EI                 ;
C016  F1            POP  AF            ;
C017  6 7     READ: LD   B, 7          ; Lecture de 7 secteurs
C019  0E F8         LD   C, 0FB        ; à partir du secteur 5
C01B  11 5 00       LD   DE, 5         ; en C100H
C01E  21 00 C1      LD   HL, 0C100     ;
C021  A7            AND  A             ;
C022  CD 10 40      CALL 4010          ;
C025  21 FF 00      LD   HL, 0FF       ; pas d'erreur = 255
C028  30 1          JR   NC, RDEND     ; variable BASIC
C02A  6F            LD   L, A          ; code d'erreur
C02B  22 F8 F7 RDEND: LD (0F7FB), HL   ; → variable BASIC
C02E  3E 2          LD   A, 2          ;
C030  32 63 F6      LD   (0F663), HL   ;
C033  3A C1 FC      LD   A, (0FCC1)    ; re-sélectionne ROM
C036  26 40         LD   H, 40         ; BASIC et retour au
C038  CD 24 00      CALL 24            ; BASIC
C03B  FB            EI                 ;
C03C  C9            RET                ;
```

Pour exploiter cette petite routine dans un programme BASIC, nous procéderons comme suit (si vous avez un MSX à un seul disque, modifiez l'instruction à l'adresse C000 par 3E 00 pour lire le Directory du lecteur A : plutôt que celui du lecteur C :. Dans le programme BASIC, remplacez le 3E 02 de la ligne DATA numéro 50 par 3E 00).

```
10 CLEAR200,&HBFFFF
20 CLS:AD=&HC000:DEFUSR=AD
30 FOR I=AD TO I+60 : READ A$
40 POKE I, VAL(« &H »+A$):NEXT
50 DATA 3E, 02, 21, 21, FB, 96, 38, 04, 23, 23, 18, F9
60 DATA 86, F5, 23, 7E, 26, 40, CD, 24, 00, FB, F1, 06
70 DATA 07, 0E, F8, 11, 05, 00, 21, 00, C1, A7, CD, 10
80 DATA 40, 21, FF, 00, 30, 01, 6F, 22, F8, F7, 3E, 02
90 DATA 32, 63, F6, 3A, C1, F6, 26, 40, CD, 24, 00, FB, C9
100 A%=USR(0)
110 IF A%<255 GOTO 160
120 FOR I = &HC100 TO I+7*512-1 STEP 32
130 FOR J = 0 TO 10 : C=PEEK(I+J) : IF C = 0 THEN END ELSE IF C=&HE5 GOTO 150
140 PRINT CHR$(C);:NEXT J : PRINT
150 NEXT I : END
160 ON A/2 GOTO 170, 180, 190, 200, 210, 220:END
170 PRINT « DISK OFFLINE »: END
180 PRINT « DISK READ ERROR » : END
190 PRINT « DISK SEEK ERROR » : END
200 PRINT « SECTOR NOT FOUND »:END
210 REM
220 PRINT « DMA TIME OUT »:END
```

Une autre méthode consiste à employer un point d'entrée de la ROM-BIOS du BASIC plutôt que le point d'entrée 4010H du contrôleur. Le programme en langage machine sera nettement plus court parce que la routine du BIOS va elle-même sélectionner le contrôleur adéquat et rendre la main à votre programme en ayant déjà re-sélectionné la ROM-BIOS. L'emploi des registres est identique excepté que le registre A peut varier de 0 à 7 au lieu de 0 à 1 (unité A : à H:).

```
                                ORG     C000

C000  3E   2                    LD      A, 2        ; disque logique C :
C002   6   7                    LD      B,7         ; nbr secteur à lire
C004  0E  F8                    LD      C , 0F8     ; type de disque
C006  11   5   00               LD      DE, 0005    ; secteur départ = 5
C009  21  00   C1               LD      HL, 0C100   ; adresse tampon
C00C  A7                        AND     A           ; carry = 0 = lire
C00D  CD  44    1               CALL    144         ; appel ROM-BIOS 144
C010  21  FF   00               LD      HL, 00FF    ; si pas erreur, 255
C013  30   1                    JR      NC, RDEND   ; variable BASIC
C015  6F                        LD      L, A        ; code d'erreur dans
C016  22  F8   F7   RDEND :     LD      (0F7F8), HL ; variable BASIC
C019  3E   2                    LD      A, 2        ;
```

```
C01B  32  63  F6          LD    (0F663), A  ;
C01E  C9                  RET               ; retour au BASIC
```

Quand vous intégrerez cette routine dans le programme Basic ci-dessus, modifiez les lignes DATA par le contenu du langage machine de la nouvelle routine et n'oubliez pas non plus de changer la ligne 30 par :

```
30 FOR I=AD TO I+30 : DEFUSR=AD
```

6.2 4031H DSKCHG DiSK ChanGe query

Cette routine vérifie si la disquette a été changée. Il est important, lorsqu'un programme lit ou écrit sur une disquette, de vérifier si cette disquette n'a pas été remplacée par une autre à l'insu du programme. La réponse de la routine peut être l'un des 3 états suivants :

a) La disquette n'a pas été changée ;
b) Impossibilité de vérifier si la disquette a été changée ;
c) La disquette a été changée.

Dans les cas b et c, la routine va déterminer le type de disquette (F8 à FF) placée dans le lecteur et va établir un DPB (Disk Parameter Block – voir chapitre 5.4) correspondant au type de l'éventuelle nouvelle disquette. Ce type sera déterminé en lisant le premier caractère du secteur 1 (FAT) ou la position 22 du secteur 0 (boot sector), suivant les fabricants. Ce caractère est appelé le « media descriptor » et a déjà été décrit dans les chapitres 5.4 (DPB) et 4.5.

En entrée :

A	Numéro du disque choisi (0 ou 1) de ce contrôleur
B	0
C	Type de disque que l'on s'attend à trouver dans le lecteur (F8 à FF).
HL	Adresse du DPB du disque choisi (voir chapitre 5.4)

En sortie :

CF = 0	L'opération s'est déroulée sans erreur
B = 1	La disquette n'a pas été changée
B = FF	La disquette a été changée et le type de disque est différent
B = 0	La disquette a peut-être été changée

Dans le cas où la disquette a ou a peut-être été changée, le DPB pointé par HL a été mis à jour avec les paramètres correspondants au type de la disquette présente dans le lecteur.

CF = 1	Une erreur s'est produite pendant l'opération
A = 2	Pas de disquette dans le lecteur
A = 4	Mauvaise lecture du secteur 0 ou 1
A = 6	Mauvais positionnement sur la piste 0

A = 8 Pas trouvé le secteur 0 ou 1
A = 0A Le type de la disquette n'est pas supporté par ce lecteur
A = 0C Autres types d'erreur (DMA Time out)

Registres affectés : AF, BC, DE, HL, IX, IY

Exemple :

Vous avez un lecteur de disquettes double faces. Vous insérez une disquette de type inconnu. Avant de lire ou d'écrire sur cette disquette, il faut charger le DPB avec les paramètres appropriés au type de disquette. D'autre part, il est important de connaître le type de disquette afin de savoir où se trouve le Directory, par exemple.

Voici un petit programme qui vous renseignera le type de disquette insérée dans le lecteur A : et si cette disquette a été changée oui ou non.

```
                                ORG     C000

C000  3A  22  FB  START :  LD    A, (0FB22)   ; sélectionne la ROM
C003  26  40              LD    H, 40        ; du contrôleur 0
C005  CD  24  00          CALL  0024         ;
C008  3E  00              LD    A, 0         ; disque A:
C00A  1   F8  00          LD    BC, 00F8     ; type attendu 360K
C00D  2A  55  F3          LD    HL, (0F355)  ; pointeur DPB0
C010  CD  13  40          CALL  4013         ; Call DSKCHG
C013  26  00              LD    H, 0         ;
C015  6F                  LD    L, A         ; code erreur → HL
C016  3B  3               JR    C, REND      ; si erreur → REND
C018  5                   DEC   B            ;
C019  5                   DEC   B            ; si OK, B-2 → HL
C01A  68                  LD    L, B         ;
C01B  22  F8  F7  RDEND :  LD   (0F7F8), HL ; HL → variable
C01E  3E  2               LD    A, 2         ; BASIC
C020  32  63  F6          LD    (0F663), A   ; de type entier
C023  3A  C1  FC          LD    A, (0FCC1)   ; rétablit ROM BASIC
C026  26  40              LD    H, 40        ;
C028  CD  24  00          CALL  0024         ; et
C02B  FB                  EI                 ; Exit vers BASIC
C02C  C9                  RET                ;
```

Incorporons cette routine dans le programme BASIC :

```
10 CLEAR200, &HBFFF
20 CLS : AD=&HC000 : DEFUSR=AD
30 FOR I = AD TO I+44 : READ A$
40 POKE I, VAL (« &H »+A$): NEXT
50 DATA 3A, 22, FB, 26, 40, CD, 24, 00, 3E, 00, 01, F8, 00
60 DATA 2A, 55, F3, CD, 13, 40, 26, 00, 6F, 3B, 03, 05, 05
70 DATA 68, 22, FB, F7, 3E, 02, 32, 63, F6, 3A, C1, F1, 26
80 DATA 40, CD, 24, 00, FB, C9
90 A% = USR(0)
95 DPB = PEEK(&HF355)+256*PEEK(&HF356) : T$ = HEX$(PEEK(DPB+1))
100 IF A%<253 GOTO 150
110 ON A %-252 GOSUB 120, 130, 140, GOTO 90
120 PRINT « Disque changé. Nouveau type = »;T$ : RETURN
130 PRINT « Disque peut-être changé. Type = »;T$ : RETURN
140 PRINT « Disque non changé. Type = »;T$ : RETURN
150 ON A %/2 GOSUB 160, 170, 180, 190, 200, 210 : GOTO 90
160 PRINT « DISK OFFLINE » : RETURN
170 PRINT « SEEK ERROR » : RETURN
180 PRINT « SECTOR 0/1 READ ERROR » : RETURN
190 PRINT « SECTOR 0/1 NOT FOUND » : RETURN
200 PRINT « DISK TYPE NOT SUPPORTED» : RETURN
210 PRINT « DMA TIME OUT » : RETURN
```

Il est à noter que le message « Disque changé... » n'apparaît que si le type de la disquette est différent de celui de la disquette précédemment installée dans le lecteur. Il faut aussi noter que les routines du « disk driver » varient parfois d'un fabricant à l'autre ou d'une version de « disk driver » à l'autre chez le même fabricant.

Le principe généralement utilisé par la majorité des fabricants pour déterminer si un disque a été changé est de lancer un chronomètre à la fin de chaque opération disque. Si le chronomètre n'a pas atteint le temps de 0,7 secondes au moment où la routine DSKCHG s'exécute, la disquette ne peut pas avoir été changée (faites donc l'essai de changer une disquette en moins de 0,7 secondes!!!).

Si par contre le chronomètre a dépassé le temps de 0,7 secondes, alors le secteur 0 ou 1 est lu par la routine et si le type de disque ainsi détecté est différent de celui que vous avez renseigné dans le registre C, c'est que la disquette a été changée, sinon la routine déclare simplement que la disquette pourrait avoir été changée.

6.3 4016H GETDPB GET Disk Parameter Block

Cette routine permet d'installer les paramètres d'un type de disquette dans le DPB (Disk Parameter Block), spécifié par HL.

Vous avez, par exemple, un MSX2 VG8235 de Philips avec un disque de 360K intégré (type F8). Si vous y connectez un disque 5''1/4 de 320K (type FA) au connecteur pour second disque à l'arrière de votre console, il faudra, avant de pouvoir utiliser ce second lecteur, installer les paramètres de ce type de disquette. C'est précisément ce dont s'occupe cette routine.

En entrée :

A Numéro de l'unité physique choisie (0 ou 1) de ce contrôleur
B Code du type de disquette (F8 à FF) dont on souhaite installer les paramètres dans le DPB
HL Adresse du DPB de l'unité choisie

En sortie :

Le DPB est mis à jour avec les paramètres du type de disquette spécifié par B. Le DPB commence à l'adresse précisée par HL, mais seules les positions HL+1 à HL + 18 sont mises à jour.

Registres affectés : AF, BC, DE, HL, IX, IY

Cette routine est implicitement appelée par la routine DSKCHG-4013H décrite ci-avant lorsque le disque a ou pourrait avoir été changé. En conséquence il ne sera pas proposé d'exemple pour ce point d'entrée.

6.4 4019H CHOICE CHOICE of type of format

Cette routine est appelée par les routines FORMTM (4025H) et FORMTK (4026H) qui sont respectivement la routine de formatage de la disquette en MSX-DOS et une routine de formatage générale. Toutes deux peuvent proposer des choix de format à l'écran.

Ce point d'entrée fournit simplement dans HL l'adresse du message de choix de format. En effet, les fabricants de contrôleurs-disque n'offrent pas nécessairement tous les types de format. Ainsi, par exemple, le contrôleur intégré du MSX2 VG8235 de Philips n'autorise aucun choix lorsqu'il s'agit de formater une disquette sur le lecteur intégré mais en offre un lorsqu'il s'agit de formater une disquette sur le lecteur externe (360K ou 720K). Dans le cas de cet ordinateur, le message de choix :

1 – Single Side …
2 – Double Side…

En entrée Rien

En sortie

HL 0000H s'il n'y a pas de message d choix
HL Adresse du message de choix de formatage

Le message de choix indique toujours un numéro. En effet, ce numéro servira dans la routine de formatage à sélectionner le format désiré. Le message de choix se termine toujours par 00H.

Registres affectés :

Tous les registres peuvent être affectés bien que la majorité des fabricants les préservent tous excepté HL bien entendu.

Ce point d'entrée est automatiquement appelé par les deux routines décrites ci-après. Son utilité pour nous est donc minime sauf dans le but d'analyser le message par programme pour connaître les différents formats supportés.

6.5 DSKFMT DiSK ForMaT

C'est le point d'entrée de la routine de formatage proprement dite. Toutes les pistes vont être formatées, le secteur 0 (boot sector) va être installé, les deux copies de la FAT initialisés à 00H, sauf les trois premiers octets qui contiendront le type de disque (F8 à FF) et deux octets FFH. Les secteurs réservés au Directory vont être initialisés à 00H et tous les autres secteurs à E5H.

En entrée :

A Code du type de format (1 à 8). Si la routine n'offre pas de choix de format, A n'a pas d'effet. Sinon, le code de choix doit correspondre au numéro de choix présent dans le message sont l'adresse est donnée par le point d'entrée précédent (CHOICE 4019H).
D Numéro de l'unité physique à formater (0 ou 1)
HL Adresse d'une zone de travail que la routine peut utiliser dans risquer d'écraser votre programme.
BC Longueur de cette zone de travail

En sortie :

CF Si l'opération se déroule sans erreur, CF = 0
Si une erreur se produit, CF = 1 et A donne le code d'erreur dont la signification est reprise ci-dessous :
A = 00 La disquette est protégée contre l'écriture
A = 02 Il n'y a pas de disquette dans le lecteur
A = 04 Erreur de lecture
A = 06 Erreur de positionnement des têtes
A = 08 Secteur pas trouvé
A = 0A Erreur d'écriture
A = 0C Mauvais paramètre d'entrée (A ou D)
A = 0E Zone de travail trop petite
A = 10 Autre type d'erreur (DMA time-out)

Registres affectés : AF, BC, DE, HL, IX, IY

Ce point d'entrée n'est intéressant que si vous connaissez d'avance le numéro du disque où sera insérée la disquette à formater et le code de type de format désiré. L'avantage de cette routine par rapport aux deux suivantes est qu'elle laisse l'écran intact.

Exemple :

Dans un VG8235 de Philips, le second lecteur du contrôleur intégré peut être formaté suivant le choix 1 (360K) ou suivant le choix 2 (720K). Voici une routine qui va formater la disquette du

disque B : au format 720K, pour autant que le lecteur B : soit du type « double-face » bien entendu. Nous l'intégrerons dans un programme BASIC.

```
                                ORG     C000

C000  3A  48  F3  START :  LD    A, (0F34B)  ; sélectionne la ROM disque
C003  26  40              LD    H, 40       ;
C005  CD  24  00          CALL  0024        ;
C008  3E  02              LD    A, 2        ; Format 2 (720K)
C00A  16  1               LD    D, 1        ; Disque B :
C00C  21  00  90          LD    HL, 9000    ; Zone de travail
C00F  1   00  30          LD    DC, 3000    ; Longueur zone
C012  CD  1C  40          CALL  401C        ; Call format
C015  21  FF  00          LD    HL, 00FF    ; 255 → HL
C018  30  1               JR    NC, FIN     ; pas erreur → FIN
C01A  6F                  LD    L, A        ; Code erreur → HL
C01B  22  F8  F7  FIN :   LD    (0F7F8), HL ; HL → variable
C01E  3E  3               LD    A, 3        ; type entier
C020  32  63  F6          LD    (0F663), A  ;
C023  3A  C1  FC          LD    A, (0FCC1)  ; Sélection ROM BASIC
C026  26  40              LD    H, 40       ;
C028  C3  24  00          JP    0024        ; et retour au BASIC
```

```
10 CLEAR200, &H8FFF
20 CLS : PRINT « FORMATAGE DISQUE B: EN 720 K »
30 AD = &HC000 : DEFUSR = AD : FOR I = AD TO I+42
40 READ A$ : POKE I, VAL (« &H »+A$) : NEXT
50 A = USR(0)
60 IF A = 255 THEN END
70 ON A/2+1 GOTO 80, 90, 100, 110, 120, 130, 140, 150, 160
80 PRINT « Disque protégé» : END
90 PRINT « Lecteur vide» : END
100 PRINT « Erreur de lecture» : END
110 PRINT « Erreur de positionnement» : END
120 PRINT « Secteur pas trouvé» : END
130 PRINT « Erreur d'écriture» : END
140 PRINT « Mauvais paramètre» : END
150 PRINT « Zone de travail trop petite» : END
160 PRINT « Autre type d'erreur» : END
170 DATA 3A, 48, F3, 26, 40, CD, 24, 00, 3E, 02, 16, 01, 21, 00, 90, 01, 00, 30
```

180 DATA CD, 1C, 40, 21, FF, 00, 30, 01, 6F, 22, F8, F7, 3E, 03, 32, 63, F6, 3A
190 DATA C1, FC, 26, 40, C3, 24, 00

6.6 FORMTM FORMaT Msx-dos

Ce point d'entrée va simplement appeler le suivant (4026H – FORMTK) en ayant préalablement activé le CARRY FLAG.

6.7 FORMTK FORMaT with Keyboard choice

Ce point d'entrée sert au formatage d'une disquette tant en BASIC qu'en MSX-DOS. Le bon déroulement d'une opération de formatage nécessite une certaine quantité de mémoire RAM. Il faut donc que la routine FORMTK connaisse l'emplacement et la dimension de cette mémoire RAM de travail. Il y a deux possibilités :

1) En MSX-DOS, il suffit d'indiquer dans HL l'adresse de la zone de travail et dans BC la dimension de cette zone et de placer le CARRY FLAG à l'état 1 (le CF est automatiquement mis à 1 par le point d'entrée précédent – FORMTM).

2) En Disk-BASIC, il n'est plus nécessaire de remplir HL et BC car ils vont être automatiquement trouvés par la routine. En effet, l'adresse de la zone de travail proviendra du pointeur F6C6H qui indique en BASIC le début de la zone de la pile (STACK) moins 256 octets. La taille de la zone sera donnée par la soustraction de ces deux adresses. Pour que la routine agisse de la sorte, le CARRY FLAG doit être à l'état 0 lors de l'appel de la routine.

Que fait encore cette routine ? Elle va afficher à l'écran un message vous demandant quelle unité vous désirez employer pour formater votre disquette.

Si vous avez 4 unités logiques sur votre système, le message sera : « DRIVE NAME (A, B, C, D) ». Les noms offerts en option dans les parenthèses sont automatiquement ajustés suivant la configuration disque dont vous êtes équipés.

Lorsque vous aurez répondu au choix proposé, un éventuel message de choix de format vous sera proposé si le contrôleur du disque choisi supporte plusieurs formats (voir point d'entrée 4 – 4019 – CHOICE).

Par exemple :

1 – SINGLE SIDE 2 – DOUBLE SIDE

Après avoir répondu à ce choix, le message suivant sera affiché :

STRIKE A KEY WHEN READY

Dès l'enfoncement de n'importe quelle touche le point d'entrée DSKFMT – 401C sera appelé et le formatage commencera. Par contre, si vous enfoncez CTRL-STOP ou CTRL-C, le message « ABORTED » sera produit et l'on quitte la routine sans autre action.

En entrée :

CARRY = 1 Formatage sous environnement BASIC

CARRY = 0 Formatage sous environnement MSX-DOS
où HL = adresse de la zone de mémoire libre
BC = longueur de la zone de mémoire libre

En sortie :

Voir DSKFMT 401C ci-dessus.

Registres affectés : AF, BC, DE, HL, IX, IY

Exemple :

Si le programme est destiné à être intégré à un programme BASIC, il est de loin plus facile de poser directement CALL FORMAT dans le programme BASIC plutôt que de passer par le langage machine. Par contre, pour un programme en langage machine sous MSX-DOS, vous pouvez procéder comme suit pour formater n'importe quelle disquette. Remarquez également la méthode employée pour appeler la routine en 4025H du contrôleur. Si cette méthode ne vous est pas familière, voyez le chapitre 9.

```
                                ORG     0100

0100  2A  06  00      START :   LD      HL, (0006)   ; adresse bas du DOS
0103  11  1A  01                LD      DE, PRGEND   ; adresse fin de PRG
0106  B7                        OR      A            ; CF=0
0107  ED  52                     SBC     HL, DE       ; HL=longueur zone libre
0109  44                       LD      B, H         ; longueur → BC
010A  4D                        LD      C, L         ;
010B  EB                        EX      DE, HL       ; HL=adresse zone libre
010C  DD  21  25  40            LD      IX, 4025     ; routine format
0110  FD  21  21  FB            LD      IY, (0FB21)  ; Slot contrôleur 0
0114  CD  1C  00                CALL    CALSLT       ; Appel IX slot IY
0117  C3  00  00                JP      0000         ; Retour au DOS
011A  00              PGMEND :  DEFB    0            ; Adresse fin PRG
```

6.8 401FH DSKSTP DiSK StoP (dernières versions)

Ce point d'entrée n'est présent que sur les dernières versions de contrôleurs (celles apparues avec le MSX2). Il permet de dé-sélectionner les deux unités de disquettes de ce contrôleur et d'arrêter immédiatement leurs moteurs.

Pour rendre votre programme compatible avec tous les types de versions, testez d'abord si la position 401FH contient 00H. Si tel est le cas, n'appelez pas ce point d'entrée car il n'existe pas. Si vous persistez à l'utiliser, alors l'effet produit sera celui du point d'entrée 4022H-BASIC, à savoir el retour au BASIC.

En entrée : Rien

En sortie : Rien

Registres affectés : AF, BC, DE, HL, IX, IY

Exemple :

Cet appel est utile lorsque votre programme compte travailler avec les « interruptions interdites » ou lorsqu'il supprime les « hooks » réservés aux interruptions. Car, dans ce cas, le moteur du lecteur à partir duquel votre programme a été chargé ne s'arrêterait pas du fait que le « driver » du disque emploie un délai créé à partir des « interruptions » pour arrêter le moteur. Ce programme montre comment tester si le point d'entrée existe et, dans ce cas, arrête immédiatement les moteurs des disques A : et B :; autrement le programme provoque un délai suffisant pour que le moteur du disque s'arrête de lui-même.

```
                                        ORG     0100

0100  DD  21  1F  40   START :   LD      IX, 401F    ; Test si 401F = 0
0104  3E  00                    LD      A, 0        ; Si oui, délai
0106  DD  BE  00                VP      (IX+0)      ; pour que le moteur
0109  28  09                    JR      Z, DELAI    ; s'arrête
010B  FD  2A  21  FB            LD      IY, (0FB21) ; Si non, arrêt
010F  CD  1C  00                CALL    1C          ; par appel 401F
0112  18  0C                    JR      FIN         ;
0114  06  80          DELAI :   LD      C, 80       ; 128 fois
0116  21  00  00      DELAI1 :  LD      HL, 0       ; 65536 pour arrêt
0119  2B              REPEAT :  DEC     HL          ; du moteur
011A  7C                        LD      A, H        ; de lui-même
011B  B5                        OR      L           ;
011C  20  FB                    JR      NZ, REPEAT
011E  10  F6                    DJNZ    DELAI1
0120  F3                FIN     DI
0121  ...                                         Suite du programme
```

6.9 4029H ALLSTP All disks StoP (dernières versions)

Ce point d'entrée n'est présent que sur les dernières versions de contrôleurs apparues avec le MSX2. Il permet de dé-sélectionner et d'arrêter le moteur de TOUS les disques de votre MSX indépendamment du contrôleur auquel ils sont connectés.

Pour savoir quelle version tourne sur votre programme, testez simplement le premier octet de ce point d'entrée. S'il s'y trouve C9H, il s'agit d'une ancienne version non équipée du point d'entrée.

En entrée : Rien

En sortie : Rien

Registres affectés : AF, BC, DE, HL, IX, IY

Exemple :

Voir ci-dessus mais remplacez 401FH par 4029H et 00H par C9H aux endroits appropriés.

6.10 4022H BASIC return to BASIC

Cet appel permet de quitter un programme en langage machine tournant sous MSX-DOS pour retourner à la routine d'initialisation du BASIC. Il agit donc comme si vous allumiez votre ordinateur sans y introduire de disquette.

Trois options se présentent, selon le contenu de la position F340H.

1) F340H = 00H

Si la position F340 contient 00H au moment de l'appel, le programme AUTOEXEC.BAS sera recherché sur le disque couramment sélectionné par le MSX-DOS.

- s'il existe, il sera chargé et exécuté

- dans le cas contraire, après avoir éventuellement demandé la date (si vous n'avez pas de dateur électronique, cas du MSX1), un RUN automatique sera effectué

Ce RUN permet le lancement d'un programme BASIC qui résiderait à l'adresse donnée par le pointeur F676H (TXTTAB). Le programme doit être en binaire compressé et non en ASCII et aura été déposé en mémoire par le programme en langage machine.

Si les trois premières positions mémoire données par le pointeur F676H sont à 00H, alors les messages de copyright du Disk-BASIC s'afficheront à l'écran et aucun programme ne sera exécuté.

2) F340H <> 0H – 0000H <> C3H

Si la position F340H contient autre chose que 00H et que la position RAM 0000H est différente de C3H alors seul un « RUN » automatique sera exécuté comme dans le paragraphe précédent.

3) F340H <> 0H – 0000H = C3H

Si la position F340H contient autre chose que 00H et que la position RAM 0000H est égale à C3H (environnement MSX-DOS) alors le programme BASIC dont le nom est indiqué à partir de 0081H de la RAM (et dont la longueur du nom est indiquée en 0080H) sera recherché et exécuté.

En résumé :

1) F340 = 0 : AUTOEXEC.BAS existe → exécution AUTOEXEC.BAS
2) F340 = 0 : AUTOEXEC.BAS n'existe pas → demande la date
RUN
a) si programme BASIC en mémoire → exécution
b) si pas de programme BASIC → message copyright
3) F340 <> 0 : RAM 000H <> C3H → voir a) et b) ci-dessus
4) F340 <> 0 : RAM 000H = C3H → recherche et exécute le programme BASIC dont le nom se trouve à partir de 0081H. Si la longueur du nom de programme indiquée par 0080H est 0 voir a) et b) ci-dessus.

En entrée :

F380H = 0 pour exécuter AUTOEXEC.BAS s'il existe. Sinon pour demander la date (MSX1) et exécuter un éventuel programme BASIC se trouvant en mémoire. Si la mémoire pointée par F676H vaut 00H-00H-00H, alors affichage des messages de copyright du Disk-BASIC.

F380H <> 0 1) Si on est dans l'environnement Disk-BASIC, exécution d'un éventuel programme BASIC se trouvant en mémoire ou affichage des messages de copyright du Disk-BASIC.

2) Si on est dans l'environnement MSX-DOS, exécution du programme BASIC dont le nom est donné en mémoire à l'adresse de la DTA (Disk Transfer Address, par défaut 0080H). En cas d'absence de nom de programme dans la DTA, exécution d'un éventuel programme BASIC se trouvant en mémoire ou retour au message de copyright du Disk-BASIC.

0080H Dans l'environnement MSX-DOS, placez en 0080H la longueur du nom de programme BASIC que vous placez à partir de 0081H.

0081H Dans l'environnement MSX-DOS, placez en 0081H le nom du programme BASIC à exécuter sous la forme habituelle « nnnnnnnn.eee ». le nom du programme peut être précédé d'un certain nombre d'espaces qui seront ignorés par la routine et qui ne doivent pas être comptabilisés dans la longueur du nom de programme placée en 0080H.

Ce point d'entrée est donc essentiellement utile dans l'environnement MSX-DOS pour chaîner un programme en BASIC avec un programme MSX-DOS. Revenez ici après avoir assimilé les chapitres 7 et 8...

Exemple 1 :

Nous allons montrer ici comment appeler le programme BASIC « EXEMPLE.BAS » à partir d'un programme MSX-DOS.

```
                                        ORG         0100
0100                                    ...                     ; Corps programme
  ...                                   ...                     ;
  ...                                   ...                     ;       MSX-DOS
  ...                                   ...                     ;
04FB  3E  01          Basic :   LD          A,1         ; F340 = 1 = pas de
04FD  32  40  F3                LD          HL, NOM     ; autoexec
0500  21  16  05                LD          DE, 0080    ; Déplacement nom
0503  11  80  00                LD          BC, 0D      ; programme et
0506  01  0D  00                LD                      ; longueur dans 81
0509  ED  B0                    LDIR                    ; et 0080.
050B  FD  2A  21  FB            LD          IY, (0FB21) ; IY = slot ctrl 0
050F  DD  21  22  40            LD          IX, 4022    ; IX = adresse call
0513  C3  1C  00                JP          001C        ; point d'entrée
0516  0C              NOM :     DEFB        0C          ; Longueur nom
0517  45  58  45                DEFM        'EXEMPLE.BAS' ; Nom du programme
051B  50  4C  45  4D                                    ;
051F  42  41  53  2E                                    ;
```

Exemple 2 :
Pour quitter un programme MSX-DOS et revenir au message de copyright du Disk-BASIC, vous pouvez utiliser l'exemple suivant.

```
                                        ORG         0100
0100                                    ...                     ; Corps programme
  ...                                   ...                     ;
  ...                                   ...                     ;       MSX-DOS
  ...                                   ...                     ;
1000  3E  01                    LD          A,1         ;
1002  32  40  F3                LD          (0F340), A  ;
1005  3E  00                    LD          A, 0        ; Longueur nom = 0
1007  32  80  00                LD          (0080), A   ;
100A  2A  76  F6                LD          HL, (0F676) ; Efface 3 premiers
100D  06  03          REPEAT :  LD          B,3         ; octets de la zone
100F  36  00                    LD          (HL), 0     ; réservée au
```

```
1011  10  FA                      DJNZ    REPEAT      ; programme BASIC
1013  FD  2A  21  FB  Basic :    LD      IY, (0FB21) ; IY = slot ctrl 0
1017  DD  21  22  40             LD      IX, 4022    ; IX = adresse call
101B  C3  1C  00                JP      001C        ; Appel du BASIC
```

Exemple 3 :
Cet exemple vous montre comment quitter un programme MSX-DOS et provoquer le chargement et l'exécution du programme AUTOEXEC.BAS, s'il exsite.

```
                                ORG     0100
0100                            ...                 ; Corps programme
  ...                           ...                 ;
  ...                           ...                 ;       MSX-DOS
  ...                           ...                 ;
1000  3E  00                    LD      A, 0        ; AUTOEXEC.BAS
1002  32  40  F3                LD      (0F340), A  ;
1005  2A  76  F6                LD      HL, (0F676) ; Efface 3 premiers
1008  06  03          REPEAT :  LD      B, 3        ; octets de la zone
100A  36  00                    LD      (HL), 0     ; réservée au
100C  10  FA                    DJNZ    REPEAT      ; programme BASIC
100E  FD  2A  21  FB  Basic :   LD      IY, (0FB21) ; IY = slot ctrl 0
1012  DD  21  22  40             LD      IX, 4022    ; IX = adresse call
1016  C3  1C  00                JP      001C        ; Appel du BASIC
```

Exemple 4 :
Cet exemple vous montre comment installer un programme BASIC sommaire (10 PRINT « BONJOUR » : END) en mémoire et lancer son exécution à partir d'un programme MSX-DOS.

```
                                ORG     0100
0100                            ...                 ; Corps programme
  ...                           ...                 ;
  ...                           ...                 ;       MSX-DOS
  ...                           ...                 ;
1000  3E  01                    LD      A,1         ; AUTOEXEC.BAS
1002  32  40  F3                LD      (0F340), A  ;
1005  3E  00                    LD      A, 0        ; 80 = 0 Pas de nom
1007  32  80  00                LD      (0080), A   ;
```

```
100A  ED  5B  76  F6          LD      DE, (0F676)     ; Copie programme
100E  21  21  10              LD      HL, PBASIC      ; en position
1011  01  13  00              LD      BC, LONG        ; mémoire correcte
1014  ED  B0                  LDIR                    ;
1016  FD  2A  21  FB  Basic :  LD      IY, (0FB21)     ; IY = slot crl 0
101A  DD  21  22  40          LD      IX, 4022        ; IX = adresse call
101E  C3  1C  00              JP      001C            ; Appel du BASIC
1021  0C  80  0A  00  PBASIC :DEFB    0C, 80, 0A, 00  ; Pointeur ligne + n° ligne
1024  91                      DEFB    91              ; Code PRINT
1025  22  42  4F  4E          DEFM    ‘ « BONJOUR »’  ; « BONJOUR »
1029  4A  4F  55  52
102D  22
102E  3A  81                  DEFB    3A, 81          ; Code : code END
1030  00  00  00              DEFB    00, 00, 00      ; Code fin prg
1033  13  00          LONG :  DEFW    0013            ; Longueur prg
```

6.11 402DH GETSLOT GET the controller SLOT number

Cette routine est pratiquement inutile pour le programmeur MSX-DOS. Elle fournit dans l’accumulateur un octet qu indique dans quel slot primaire et secondaire ce contrôleur réside sous la forme classique E000SSPP. Pour atteindre ce point d’entrée, il faut d’abord sélectionner cette ROM, donc connaître le slot primaire et secondaire où il réside…

En entrée : Rien

En sortie : A = N° du slot de ce contrôleur E000SSPP

Registres affectés : AF , BC, DE, HL, IX, IY

6.12 0000 GETTOP GET TOP of user memory

Ce point d’entrée fournit dans HL la plus haute adresse RAM que l’utilisateur peut employer dans l’environnement MSX-DOS (voir pointeur F34B dans le chapitre 5.7).

En entrée : Rien

En sortie : HL = adresse mémoire maximum pour l’utilisateur

7) Chapitre 7 : BASIC-DOS et MSX-DOS en langage machine

Le MSX-DOS et le Disk-BASIC ne vous offrent pas seulement tout ce qui a été vu dans les chapitres 3 et 4, mais également une série de routines garanties par la norme MSX qui vont vous permettre de manipuler le hardware de votre MSX à partir de programmes en langage machine.

Manipuler le hardware signifie disposer de routines qui permettront, par exemple, de saisir des caractères qu clavier, d'afficher ou d'imprimer des zones mémoire, d'ouvrir, de fermer, de lire et d'écrire des fichiers, etc.

Ce chapitre va nous décrire complètement toutes les possibilités offertes et sera compété par de nombreux exemples.

7.1 Pourquoi un BASIC-DOS et un MSX-DOS ?

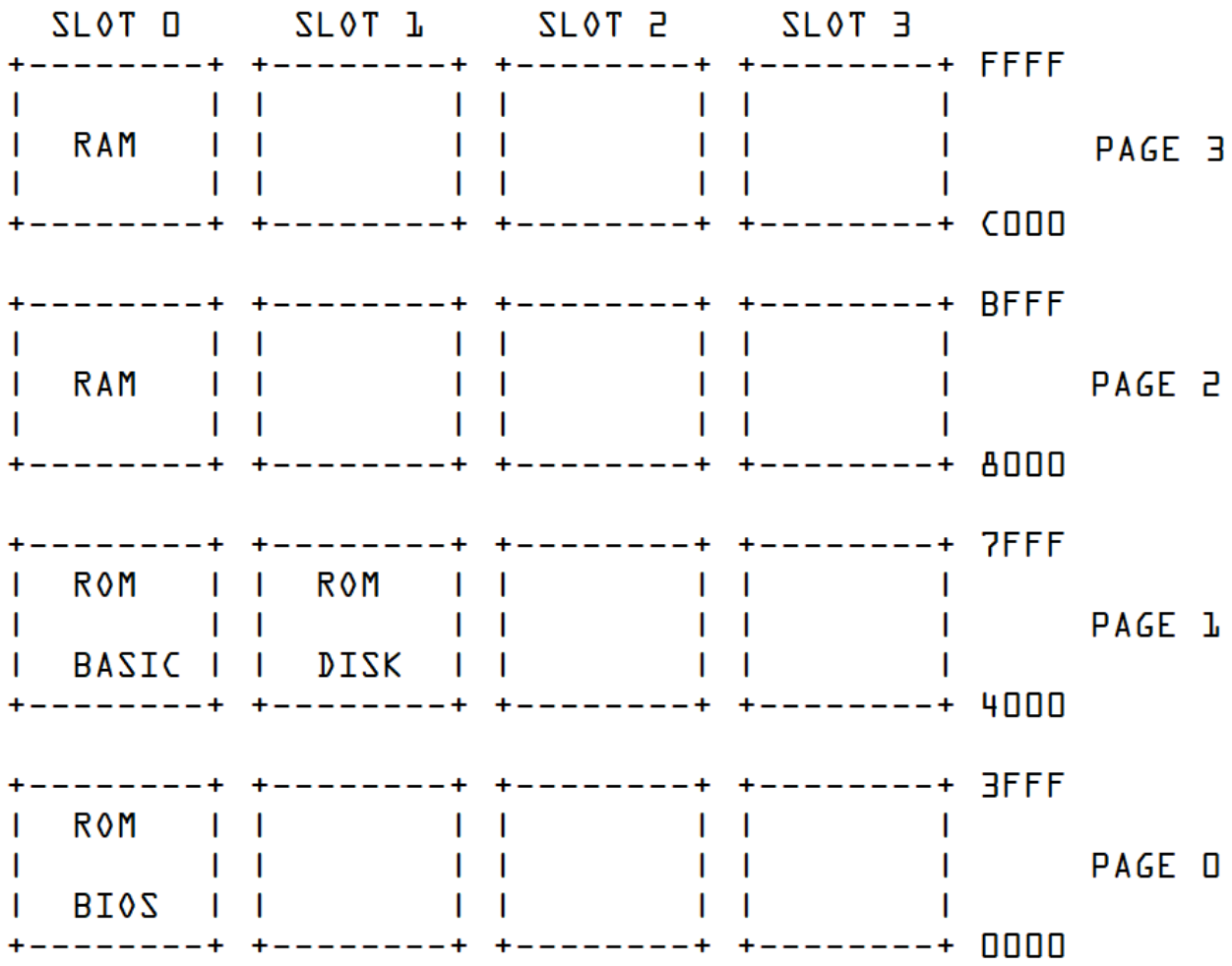
Vous savez déjà que pour utiliser le MSX-DOS, il faut disposer d'un MSX équipé de 64K de mémoire RAM et des deux fichiers MSXDOS.SYS et COMMAND.COM. Par contre, il suffit de 32K pour déjà pouvoir utiliser le Disk-Basic sans autre logiciel que celui fourni par la ROM du contrôleur-disque.

Pour permettre aux possesseurs de MSX 32K d'utiliser aussi les facilités offertes par les routines décrites dans la suite de ce chapitre, Microsoft a pourvu la ROM du contrôleur-disque de tout ce qu'il fallait pour simuler le travail d'un MSX de 64K mais sous l'environnement du BASIC c'est-à-dire avec les deux premières pages contenant la ROM BIOS et la ROM BASIC.

Cette facilité permet aux maisons de software de développer des logiciels travaillant avec disquette sans devoir développer ces logiciels sous les deux environnements MSX-DOS et Disk-BASIC.

7.2 Le BASIC-DOS

Le BASIC-DOS va offrir toutes les possibilités du MSX-DOS à l'exception bien entendu des commandes vues dans le chapitre précédent. Le travail sous BASIC-DOS implique de bien connaître son environnement mémoire. Nous allons donc replacer ci-dessous le diagramme mémoire du BASIC, tant au point de vue des ROM que des RAM. Bien entendu, les slots représentés dans ce dessin peuvent changer d'une machine à l'autre en ce qui concerne la place des RAM et même d'une configuration à l'autre en ce qui concerne l'emplacement de la ROM du contrôleur-disque. Par contre, les deux ROM du BIOS et du BASIC sont fixés par la norme. Il n'a pas non plus été tenu compte que certains MSX1 et la majorité des MSX2 travaillent également avec des slots étendus ou slots secondaires car l'important dans ce tableau est de considérer verticalement de quoi se composent les 64K en Disk-BASIC.



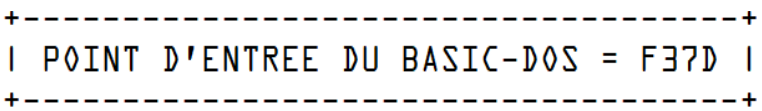
Que voyons-nous dans ce tableau ? Essentiellement que la page 0 (0000-3FFF) est constamment occupée par la ROM BIOS, que la page 1 (4000-7FFF) est occupée soit par la ROM BASIC, soit par la ROM DISK suivant que l'instruction en cours est une instruction du BASIC normal ou du Disk-BASIC.

Cette même ROM DISK sera également choisie lorsque votre programme BASIC aura lancé soit par BLOAD, soit par la fonction USR, un programme en langage machine utilisant les routines du BASIC-DOS.

Les pages 2 (8000-BFFF) et 3 (C000-FFFF) sont occupées par des RAM qui peuvent se trouver dans n'importe quel slot primaire ou secondaire dépendant de la marque et du modèle de votre MSX.

De ce tableau, il découle qu'un programme en langage machine ne peut fatalement être implanté que dans les pages 2 et 3, car elles sont les seules à contenir de la RAM.

D'autre part, le chapitre 5 nous a expliqué qu'une partie de la page 3 était réservée pour le BASIC et pour les disques. C'est donc dans cette page 3 que Microsoft a implanté le point d'entrée principal du BASIC-DOS.

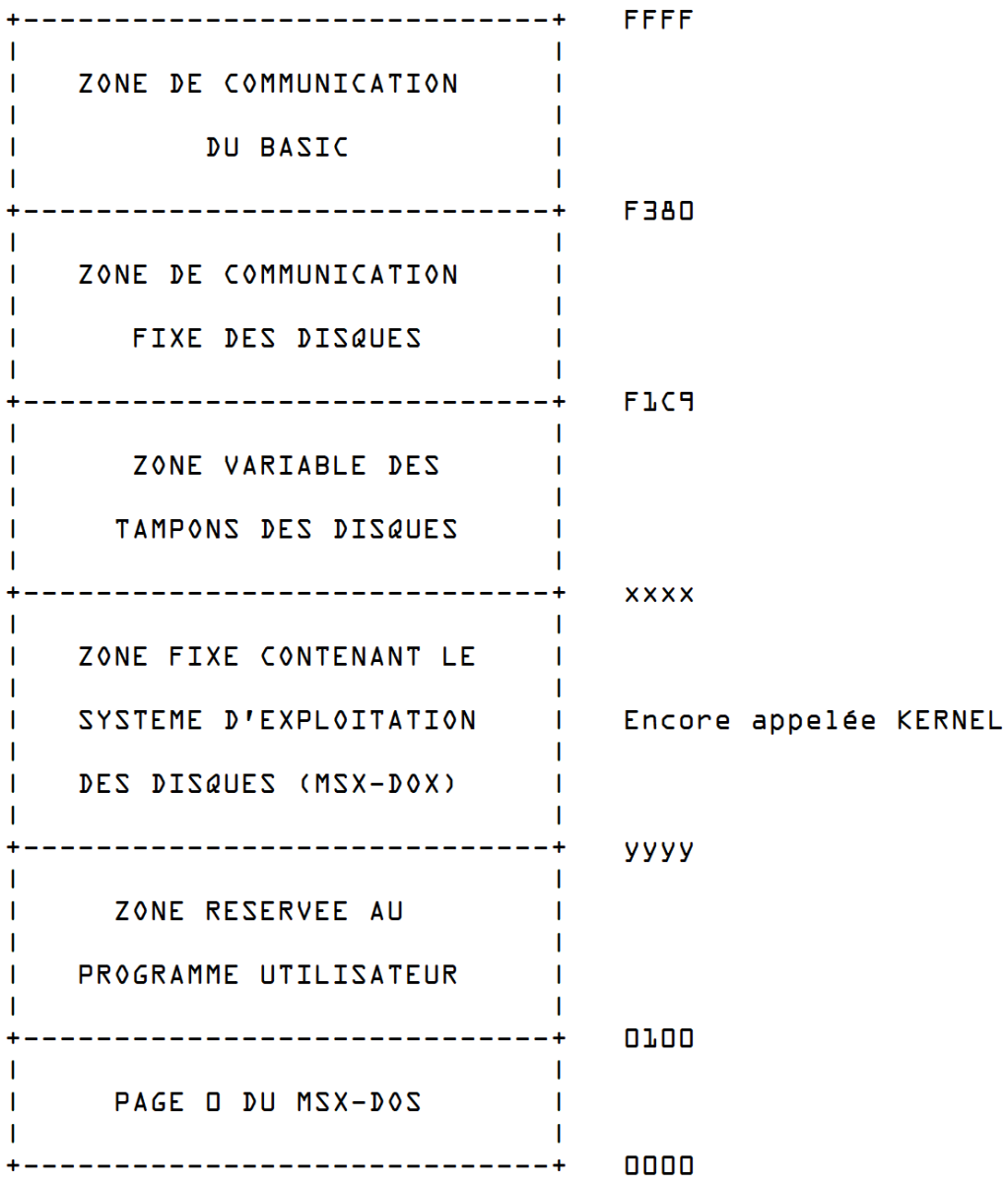


7.3 Le MSX-DOS

L'environnement mémoire en MSX-DOS s'apparente à celui des machines professionnelles. Dans celles-ci, l'intégralité de l'espace d'adressage du microprocesseur (64K) est occupé par les RAM. Il n'y a donc plus de ROM BIOS ou BASIC installés à demeure dans cet espace d'adressage.

De ce fait, le système d'exploitation disque doit être chargé en mémoire RAM à l'allumage du système à partir de fichiers (MSXDOS.SYS et COMMAND.COM – revoyez le chapitre 5 pour le détail de cette installation).

Voici donc la carte mémoire dans l'environnement MSX-DOS.



Tout le problème vient, dans l'environnement MSX-DOS, de ce que la zone des tampons des disques (sector buffer, Directory buffer, workarea buffer et les FATs) est de taille variable. En effet, cette taille dépend du nombre de contrôleurs connectés et de la méthode d'allumage de votre MSX (avec la touche CTRL enfoncée ou pas).

De toute façon, cette zone s'implante sous la zone de communication fixe des disques et s'agrandit vers le bas au plus il y a de contrôleurs-disques connectés. L'adresse F349 (zone de communication fixe des disques) contient un pointeur donnant la plus basse adresse de la zone des tampons disques (voir chapitre 5). A titre d'exemple, pour un seul disque connecté tel que dans le MSX2 VG8235 de Philips, l'adresse xxxx est DF95.

Sous la zone des tampons disques s'implante le système d'exploitation du MSX-DOS proprement dit. Cette zone est découpée en plusieurs parties distinctes qui ont été détaillées dans le chapitre 5.

La première de ces parties a été installée directement par la ROM du contrôleur-disque et contient les routines de commutations des slots. L'adresse F34B de la zone de communication fixe des disques fournit la plus basse adresse utilisée par ces routines de commutations.

Sous cette première partie est implanté le KERNEL (noyau) du MSX-DOS. C'est lui qui contient toutes les routines destinées aux fonctions du MSX-DOS qui vont être décrites dans la suite du chapitre, la description de la page 0 dans laquelle est indiquée la plus basse adresse utilisée par le KERNEL et qui servira de limite extrême pour notre programme utilisateur.

Enfin, sous le KERNEL, s'implante un stack de 256 octets qui est le stack par défaut du programmeur et le code machine des commandes internes du DOS c'est-à-dire DIR, DATE, TYPE, etc vues au chapitre précédent. Ce stack et ces commandes internes occupent 1400H positions qu'il faut donc soustraire de la plus basse adresse du KERNEL arrondie à un multiple de 256 octets pour obtenir sa plus basse adresse qui est C200H dans un MSX2 VG8235 ou VG8255.

Ce stack et les commandes internes peuvent être recouverts par le programme de l'utilisateur pourvu qu'il veille à réserver un autre espace pour le stack du système et qu'il n'emploie pas ces commandes internes.

En dehors de la page 0 (0000-0100H), tout le reste de la mémoire est disponible pour l'utilisateur.

7.4 La page 0 du MSX-DOS

La page 0 (0-0100H) de la mémoire est réservée au MSX-DOS et contient une série de points d'entrée ou de zones qui vont nous servir à exploiter toutes les facilités offertes par les fonctions du MSX-DOS et également à interconnecter notre programme avec des routines présentes dans les ROM BIOS et DISK. Cette page 00 est pratiquement compatible avec celle du CP/M. Nous indiquerons dans le tableau les différences entre CP/M et MSX-DOS en marquant d'une '*' les adresses non compatibles CP/M.

ADRESSE	LONGUEUR	NOM	DESCRIPTION
0000	3	WBOOT	L'adresse 0 contient une instruction de saut vers une routine qui va réinitialiser le MSX-DOS. En d'autres mots, si vous exécutez un jump vers cette adresse, votre programme s'arrêtera et rendra la main à l'indicatif du MSX-DOS. Le contenu des adresses 1 et 2 forment ensemble un pointeur auquel on pourra ajouter un offset pour atteindre des routines qui seront décrites plus loin.

0003*	1	IOBYT	Le IO-BYTE n'est pas supporté en MSX-DOS et nous trouvons donc 00 à cette adresse.
0004*	1	DDISK	Le numéro du disque par défaut est indiqué ici en CP/M. Ce n'est pas le cas en MSX-DOS. Employez la fonction 19 décrite plus loin pour obtenir le numéro du disques par défaut.
0005	3	BDOS	L'adresse 5 contient une instruction de saut vers le point d'entrée principal du MSX-DOS pour manipuler toutes les fonctions qui vont être décrites dans la suite de ce chapitre. Les octets des adresses 6 et 7 fournissent également la plus basse adresse du KERNEL du MSX-DOS. On peut donc utiliser le mot présent en 0006-0007 pour connaître la plus haute adresse de la TPA que vous pouvez utiliser pour votre programme en déduisant simplement 1 du contenu de 0006-0007.
000C*	3	RDSLT	L'adresse 0C contient une instruction de saut vers une routine qui permet de lire dans l'accumulateur, l'adresse mémoire HL du slot dont le caractère de sélection se trouve dans l'accumulateur.
0014*	3	WRSLT	L'adresse 14 contient une instruction de saut vers une routine qui permet d'écrire le contenu du registre E vers l'adresse de mémoire HL du slot dont le caractère de sélection se trouve dans l'accumulateur.
001C*	3	CALSLT	L'adresse 1C contient une instruction de saut vers une routine qui permet d'appeler une routine à l'adresse IX du slot dont le caractère de sélection se trouve dans IY.
0024*	3	ENASLT	L'adresse 24 contient une instruction de saut vers une routine qui permet d'installer de façon permanente le slot dont le caractère de sélection se trouve dans l'accumulateur et dont l'adresse figure dans le registre HL (voir chapitre 9).
0028*	3	RST28	Les programmes Debugger comme ZSID-DDT emploient normalement un RST 38 comme Break Point. Cependant, en MSX-DOS cette position 38 est utilisée par la routine Interrupt. Ces programmes sont donc patchés pour utiliser le RST 28.
0030*	3	CALLF	L'adresse 30 contient une instruction de saut vers une routine qui permet d'appeler une routine suivant la procédure suivante : (voir chapitre 9) RST 30 DEFB Slot de destination DEFW Adresse de la routine
0038*	3	INTRPT	L'adresse 38 contient une instruction de saut vers la routine d'interruption standard de la ROM BIOS en MSX-DOS. En CP/M, cette position contient en général une

			instruction de saut utilisée par les programmes Debugger comme DDT ou ZSID. Ces programmes DDT ou ZSID devront donc être modifiés pour tourner en MSX-DOS.
003B*	11		Les 11 octets présents à partir de l'adresse 3B forment une routine utilisée par le MSX-DOS pour commuter les slots secondaires de la page 3 (C000-FFFF) suivant les sntrées suivantes : A : Code des slots primaires à sélectionner B : Code des slots primaires à sélectionner en retour de routine. H : Masque des slots secondaires à conserver. D : Code du slot secondaire à sélectionner. En sortie, le registre L fournit le contenu précédent du registre de sélection des slots secondaires.
0046*	5		Les 5 octets de l'adresse 46 contiennent une routine permettant de forcer le registre des slots secondaires. Le registre A doit contenir le code de sélection à mettre dans le registre de sélection des slots primaires. Le registre L doit contenir le code à forcer dans le registre de sélection des slots secondaires. Le registre B doit contenir le code de sélection à mettre dans le registre de sélection des slots primaires au retour de la routine.
004B*	10		Les 10 octets de cette routine ont la même mission que celle de l'adresse 0046 à l'exception que le code à forcer dans le registr ede sélection des slots secondaires doit être placé dans le registre E.
005B	16	FCB1	Les 16 octets de l'adresse 5B contiennent le premier argument d'une commande ou d'un programme MSX-DOS. On appelle premier argument le texte continu qui suit le nom du programme après un espace au moins de séparation. Le second argument doit être séparé du premier par au moins un espace. Ainsi, si vous tapez COPY B:TEXT1.ASC A:TEXT1.COP, vous retrouverez le numéro du disque A : du fichier à copier en 5B, suivi du nom du fichier cadré à gauche en 8 caractères et de l'extension du ficher cadrée à gfauche en 3 caractères. Ce format convient pour le FCB de ce fichier.
006C	16	FCB2	Les 16 octets de l'adresse 6C sont réservés au second argument d'une commande ou d'un programme MSX-DOS suivant les règles décrites ci-dessus pour le premier argument. TEXT1.COP est le seconda argument dans l'exemple ci-dessus.
0080	128	DTA DMA	Les 128 octets à partir de l'adresse 0080 forment le tampon DTA (Disk Transfer Address) ou DMA (Direct memory Access) par défaut du système. Lorsqu'on lira un

			enregistrement d'un fichier, le contenu de cet enregistrement se place en mémoire à l'adresse DMA. Par défaut le système place ce tampon à l'adresse 80.
0100	xxx	TPA	A partir de cette adresse et jusqu'à 1 octet avant l'adresse donnée par les positions 6 et 7, se trouve ce que l'on appelle la TPA ou TRANSIENT PROGRAM AREA, ce qui signifie littéralement la ZONE DES PROGRAMMES TRANSITOIRES. C'est en fait l'espace mémoire réservé pour installer vos propres programmes. Cet espace dépend du nombre de contrôleurs-disque de votre système et la méthode d'allumage de votre MSX. A titre d'exemple, elle s'étend de 0100 à DD05 pour les machines de la gamme MSX2 de Philips soit DC05 octets ou 56325 octets libres pour le programmeur MSX-DOS. Ce nombre correspond donc à peu près au double de ce qu'offre une MSX sans disque (28815).

Commentons ce tableau. Le premier élément à en retenir est que, pour sortir d'un programme MSX-DOS de votre cru, il suffit de programmer un saut à l'adresse 0. Vous reviendrez ainsi de façon propre vers l'indicatif A> du MSX-DOS.

ADRESSE POUR SORTIR DU MSX-DOS = 0

Deuxième élément très important, c'est l'adresse du point d'entrée principal du MSX-DOS. Avouez que cette adresse 0005 est vraiment facile à retenir.

POINT D'ENTREE MSX-DOS = 0005

Troisième élément : l'adresse mémoire la plus haute que vous puissiez utiliser par votre programme est donnée indirectement par le point d'entrée principal du MSX-DOS puisqu'il suffit de soustraire 1 du mot présent en 6 et 7 pour connaître cette plus haute adresse. Cependant, pensez à réserver un espace suffisant pour la pile du système (STACK). Il est d'ailleurs recommandé de placer cette pile juste sous le KERNEL MSX-DOS donc à l'adresse fournie par le contenu de l'adresse 6 et 7. Pour placer et réserver une pile de 256 octets et fixer le « top » de votre mémoire, procédez comme suit :

FIXATION STACK ET TOP DE LA MEMOIRE

```
LD      HL, (0006H)
LD      SP, HL
DEC   H
LD     (TOP), HL
```

Quatrième élément de facilité de programmation, toutes les routines de commutations de slot se trouvent dans la page 0 aux mêmes adresses que dans la ROM BIOS du BASIC. Leur présence autorise le MSX-DOS à accéder à tous les ROM du système et donc à toutes les routines du BIOS ou de la ROM d'extension BASIC du MSX2 étendant donc le MSX-DOS à toutes les richesses graphiques du MSX2. Ces routines sont décrites dans le chapitre 9.

Les routines de commutation des slots secondaires présentes à partir de l'adresse 3B ne sont normalement pas exploitées par les programmes utilisateur mais appelées automatiquement par les routines standards de commutations de slot (RDSLT – WRSLT – CALSLT – ENASLT – CALLF…). Je les ai mentionnées uniquement pour que le tableau de la page 0 soit complet et pour que vous évitiez d'utiliser ces zones réservées.

Autr élément, très important quand on programme en MSX-DOS, est la possibilité d'appeler un programme en y adjoignant jusqu'à deux arguments. Si par exemple vous réalisez un programme en langage machine qui convertisse un fichier de texte écrit avec le code MSX en un fichier de texte en code ISO-7 bits où les caractères accentués en français sont différents, vous pouvez directement à l'appel du programme spécifier quel fichier est à traduire et en quel autre nom de fichier vous voulez que le résultat soit sauvegardé. Si CONVERT.COM est le nom de votre programme, vous pourriez l'appeler en posant par exemple :

CONVERT B:TEXT.MSX TEXT.ISO

Dans cet exemple B:TEXT.MSX est le premier « argument » et TEXT.ISO est le second. Le système d'exploitation MSX-DOS va ici nous aider considérablement en plaçant lui-même le premier argument à l'adresse 5C et le second à l'adresse 6C. Il ne se contente pas seulement de les y placer, mais il va aussi les formater en FCB c'est-à-dire convertir le nom du fichier et son extension suivant le format du FCB. Ainsi, pour l'exemple ci-dessus nous trouverons en 5C et en 6C :

```
     +-- Numéro du disque
     |
5C   02-54-45-5B-54-20-20-20-20-4D-53-5B-00-00-00-00
        T  E  X  T              M  S  X

     +-- Disque par défaut
     |
6C   00-54-45-5B-54-20-20-20-20-49-53-4F-00-00-00-00
        T  E  X  T              I  S  O
```

En 5C, le code 02 signifie que le fichier TEXT.MSX est à recherche sur le disque B : et en 6C le code 00 signifie qu'il faut générer le fichier TEXT.ISO sur le disque par défaut. Pour ouvrir ces fichiers, il nous suffira de copier les 16 octets de 5C vers l'adresse où nous comptons établir le FCB

définitif de ce fichier et de même pour le second argument. N'oubliez pas, en effet, que le FCB occupe 37 octets et que dès lors il ne dispose pas d'assez de place en 5C ou 6C.

En BASIC, chaque lecture de fichier se fait dans un variable fixée par INPUT# pour les fichiers séquentiels ou par FIELD# pour les fichiers à accès direct. En MSX-DOS, les lectures et écritures se font par référence à une zone de mémoire que l'on appelle DMA BUFFER en CP/M ou DTA (Disk Transfer Address) en MSX-DOS. L'adresse de ce tampon pourra être fixé au gré du programmeur par une fonction spéciale que nous verrons plus loin, cependant, si le programmeur omet de fixer cette adresse, le système pourvoit une zone par dafaut où les lectures et écritures d'un fichier seront transférées : c'est le Defaul Disk Transfer Address (DDTA) présent à l'adresse 80H de la page 0.

Finalement, quelques mots à propos de la TPA ou zone où s'installe vos propres programmes. Sachez d'abord que le simple fait de poser le nom d'un programme MSX-DOS après l'indicatif A> du MSX-DOS provoque le chargement de ce programme à partir de l'adresse 0100H. Il n'y a donc pas d'entête dans un programme MSX-DOS spécifiant qu'il doit être chargé de telle adresse à telle autre adresse et démarrer à une troisième adresse comme dans les programmes binaires du BASIC chargés par BLOAD. Le chargement se fera toujours en 0100H. A propos du nom d'un programme MSX-DOS, rappelons ici que le nom du programme doit contenir l'extension « .COM » mais qu'il n'est pas nécessaire de taper cette extension pour appeler le programme.

7.5 Les fonctions du BASIC-DOS et du MSX-DOS

Les fonctions qui vont être décrites dans cette section sont valables aussi bien en BASIC-DOS qu'en MSX-DOS. Le terme employé pour ces fonctions en Anglais est « System Call » ou appel système mais pour la facilité, nous continuerons de les appeler fonctions.

L'emploi des fonctions avec le langage machine est très simple et correspond à celui du CP/M ou d'autres DOS comme le MS-DOS par exemple.

Chaque fonction a reçu un numéro. Pour appeler une fonction, il suffira de mettre dans le registre C du microprocesseur le numéro de la fonction désirée et appeler le point d'entre principal du MSX-DOS (0005H) ou le point d'entrée principal du BASIC-DOS (F37DH) dépendant de l'environnement sous lequel on travaille.

Dans le reste de cette section, les exemples d'emploi feront tantôt référence à l'environnement BASIC-DOS ou à l'environnement MSX-DOS sachez, cependant, que le principe s'applique aussi bien aux deux environnements mais en changeant bien sûr le point d'entrée (0005 ou F37D) et bien entendu l'implantation à l'adresse mémoire correcte.

Bien entendu, pour certaines fonctions, il ne suffira pas de mettre le numéro de la fonction dans le registre C et d'appeler le point d'entrée du MSX-DOS. Il faudra aussi lui fournir des paramètres comme par exemple quel caractère afficher à l'écran pour une fonction d'affichage ou quel fichier ouvrir pour la fonction OPEN d'un fichier. Ce sont les paramètres d'entrée. De même, certaines fonctions vont nous rendre un résultat qui sera stocké dans des registres du microprocesseur ou en mémoire. Ce sont les paramètres de sortie.

Dans la liste des fonctions qui va suivre, vous trouverez également une indication de compatibilité avec le CP/M. En effet, le MSX-DOS est presque complètement compatible avec le CP/M ; chaque fonction sera donc renseignée comme compatible ou non avec le CP/M. D'autre part, la liste des registres préservés par la fonction sera aussi donnée.

7.5.1 Fonction 00 SYSTEM RESET

Dans l'environnement MSX-DOS, cette fonction joue le même rôle qu'un saut à l'adresse 0000H. Cela signifie que le programme en cours s'arrête et que l'on revient à l'indicatif A> du MSX-DOS.

Dans l'environnement BASIC-DOS, cette fonction provoque l'arrêt du programme et le retour à l'indicatif OK du BASIC.

En entrée : Rien
En sortie : Rien
Compatible : Oui

Exemple : Pour terminer un programme, vous avez le choix d'utiliser cette fonction ou, en MSX-DOS, un saut à l'adresse 0000H.

```
LD    C, 0        ou        JP    0000H
JP    0F37DH
```

7.5.2 Fonction 01 CONSOLE INPUT

Cette fonction permet de saisir un caractère posé au clavier. Le caractère tapé s'affiche à l'écran et la routine vérifie également si le code posé est Control-C, auquel cas la fonction 00 est automatiquement exécutée, ou Control-P et alors tout ce qui sera affiché sera aussi dirigé vers l'imprimante, ou Control-N et alors l'écho vers l'imprimante sera supprimé. Attention qux caractères graphiques à deux codes (01-XX), car un seul code est saisi.

En entrée : Rien
En sortie : Les registres A et L contiennent le code du caractère posé au clavier.
Préserve : Les registres C, D et E sont préservés
Compatible : Oui

Exemple : Voici notre premier programme DOS. Il ne fait qu'autoriser l'opérateur à poser un caractère à la fois. Si ce caractère est le code * ou Control-C, alors le programme s'arrête et l'on revient à l'indicatif OK ou A> du DOS.

```
                        ORG     B000

B000    0E01    START : LD      C, 1        ; A$=INPUT$(1)
B002    CD7DF3          CALL    0F37D       ;
B005    FE3A            CP      '*'         ; IF A$<> « * »
B007    20F7            JR      NZ, START   ; GOTO START
```

```
B009    0E00          LD      C, 0        ; Retour au
B00B    C37DF3        JP      0F37D       ; BASIC-DOS
```

Comme il s'agit de notre premier exemple, nous allons montrer comment introduire ce petit programme dans un fichier '.BIN' qui pourra être rechargé plus tard par la commande BLOAD « INPUT.BIN ».

```
10 CLEAR 200, &HAFFF
20 FOR I = &HB000 to &HB00D
30 READ A$
40 POKE I, VAL(« &H »+A$)
50 NEXT
60 BSAVE « INPUT.BIN », &HB000, &HB00D, &HB000
70 END
80 DATA 0E, 01, CD, 7D, F3, FE, 3A
90 DATA 20, F7, 0E, 00, C3, 7D, F3
```

Si vous désirez travailler dans l'environnement MSX-DOS et que vous ne disposez pas d'un macro-assembleur comme DEVPAC ou MACRO-80, voici le moyen de créer le fichier 'INPUT.COM' en BASIC.

```
10 OPEN « INPUT.COM » AS #1 LEN = 1
20 FIELD #1, 1 AS A$
30 I=1
40 READ W$
50 IF W$ = « * » THEN CLOSE : END
60 LSET A$=CHR$(VAL(« &H »+W$))
70 PUT #1, I
80 I = I + 1 : GOTO 40
90 DATA 0E, 01, CD, 7D, F3, FE, 3A
95 DATA 20, F7, 0E, 00, C3, 7D, F3, *
```

Voilà, il vous suffira d'appeler votre programme 'INPUT' lorsque vous aurez rechargé votre système en MSX-DOS.

7.5.3 Fonction 02 CONSOLE OUTPUT

Cette fonction permet d'envoyer un caractère vers l'écran à l'emplacement courant du curseur. Cette fonction teste également le clavier pour vérifier si un des codes CTRL-C, CTRL-P, CTRL-N ou CTRL-S a été généré, auquel cas la fonction qui y est associée est exécutée.

En entrée :	Le registre E doit contenir le caractère à afficher. Ce peut être un caractère typographique ou un des codes de fonction comme 07 (Beep), 0C (CLS), etc.
En sortie :	Le registre E d'origine est copié vers les registres A et L.
Préserve :	Les registres C, D et E sont préservés.

Compatible : Oui

Exemple : Ce petit programme va simplement effacer l'écran en envoyant le code 0C. Si vous l'appelez 'CLS.COM', vous pourrez effacer l'écran dans l'environnement MSX-DOS comme vous avez l'habitude de el faire en BASIC c'est-à-dire en posant simplement CLS.

```
                        ORG     0100

0100    1E0C    START : LD      E, 0C       ; Code 0C = CLS
0102    0E02            LD      C, 02       ; Envoi vers l'écran
0104    CD0500          CALL    5           ;
0107    C30000          JP      0000        ; Retour au DOS
```

7.5.4 Fonction 03 AUXILIARY INPUT

Cette fonction permet de saisir un octet en provenance de l'entrée auxiliaire. Cette entrée peu connue est en général réservée pour l'interface RS-232-C qui permet de mettre en liaison deux ordinateurs MSX via un câble ou encore via modems et le réseau téléphonique. Cette fonction teste également les codes CTRL-C, CTRL-P et CTRL-N en provenance du clavier et le cas échéant exécute la fonction qui y est associée.

En entrée : Rien
En sortie : Les registres A et L contiennent le caractère en provenance de l'entrée auxiliaire. Si cette entrée n'existe pas sur votre MSX, le système retourne le code 1A qui est le code de fin de fichier.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui

Exemple : Ce programme affiche à l'écran tous les codes reçus de l'entrée auxiliaire jusqu'à ce qu'un code de fin de fichier (1A) arrive, auquel cas, le programme revient à l'indicatif A> du DOS.

```
                        ORG     0100

0100    0E03    START : LD      C, 03       ; Lire PORT AUX.
0102    CD0500          CALL    005         ;
0104    FE1A            CP      1A          ; Si fin fichier
0107    CA0000          JP      Z, 0000     ; retour au DOS
010A    5F              LD      E, A        ; Envoi du code
010B    0E02            LD      C, 02       ; à l'écran
010D    CD0500          CALL    0005        ;
```

```
0110  1BED          JR    START     ; recommence
```

7.5.5 Fonction 04 AUXILIARY INPUT

La fonction 04 envoie le contenu du registre E vers la sortie auxiliaire qui est généralement employée avec l'interface RS-232-C. Si cette interface n'est pas installée, cette fonction est sans effet. Cette fonction teste également les codes CTRL-C, CTRL-P et CTRL-N en provenance du clavier et le cas échéant exécute la fonction qui y est associée.

En entrée : Le registre E doit contenir le code à envoyer à la sortie auxiliaire.
En sortie : Le registre E d'origine est copié vers les registres A et L.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui

Exemple : Ce programme envoie le message « Bien reçu votre message » suivi d'une marque de fin de fichier vers la porte de sortie auxiliaire.

7.5.6 Fonction 05 LIST OUTPUT

La fonction 05 envoie le contenu du registre E vers l'imprimante. Si l'imprimante n'est pas connectée ou si elle est OFF-LINE, la routine attend que vous la placiez ON-LINE. Cependant, il est possible d'arrêter le programme et donc l'impression si vous enfoncez CTRL-C ou CTRL-STOP. N'oubliez pas d'éventuellement convertir le code MSX dans le code de l'imprimante si vous avez une imprimante non-MSX.

En entrée : Le registre E doit contenir le code à imprimer.
En sortie : Le registre E sera copié vers les registres A et L.
Préserve : Les registres C, D et E seront préservés.
Compatible : Oui

Exemple : Nous allons imprimer l'argument qui suit le nom du programme. Rappelons qu'en MSX-DOS on peut faire suivre le nom d'un programme de deux arguments et qu'ils seront ainsi placés en mémoire aux adresses 5C et 6C. Ainsi, si nous appelons ce petit programme LPRINT.COM, vous obtiendrez l'impression de BONJOUR si vous tapez après l'indicatif A> LPRINT bonjour. Attention que l'argument doit suivre les règles propres aux noms de fichiers.

```
                      ORG   0100

0100  215D00  START : LD    HL, 05D   ; Pointe argument
0103  0608          LD    B, 8      ; Boucle B fois
0105  C5      RPT :  PUSH  BC        ; Sauve compteur boucle
0106  E5            PUSH  HL        ; Sauve pointeur
0107  5E            LD    E, (HL)   ; Lit un code
0108  0E05          LD    C, 05     ; Envoi vers l'imprimante
010A  CD0500        CALL  0005      ;
```

```
010D   E1          POP   HL      ; Rappel pointeur
010E   23          INC   HL      ; Incrémente le
010F   C1          POP   BC      ; Rappel compteur boucle
0110   10F3        DJNZ  RPT     ; Répète boucle
0112   C30000      JP    0000    ; Retour au DOS
```

7.5.7 Fonction 06 DIRECT CONSOLE INPUT/OUTPUT

- Si le registre E contient FF, alors il s'agit de la fonction Direct Input. Cette fonction permet d'obtenir le code qui serait prêt dans le tampon du clavier et le retourne dans le registre A. Si aucun code n'a été posé, alors le registre A contiendra 00. Attention qu'il s'agit d'une seule scrutation de clavier sans attente de frappe d'un caractère et dès lors le code aura du être frappé avant l'appel de cette fonction.

- Si le registre E contient un code différent de FF, alors il s'agit de la fonction Direct Output. Elle permet d'afficher à l'écran le code contenu dans le registre E.

- On parle de « Direct » Input ou Output en ce sens que cette fonction évite les contrôles et dispositifs habituels du CP/M à savoir qu'il n'y a pas de test des codes CTRL-P, CTRL-N, CTRL-C ou CTRL-S ; ni d'affichage du caractère tapé en input, ni de formatage des caractères envoyés à l'écran en output.

En entrée : Le registre E doit contenir FF pour un Direct Input et le code à afficher en Direct Output.
En sortie : En Direct Input, les registres A et L contiennent 00 si aucun code n'est prêt dans le tampon du clavier ou le code de la touche dans le cas contraire.
En Direct Output, les registres A et L contiendront le code envoyé à l'écran.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui

Exemple : L'emploi de cette fonction n'est pas conseillé par Digital Research qui est le concepteur du CP/M parce qu'elle court-circuite les tests et dispositifs habituels des inputs/outputs. Mais c'était la seule fonction qui pouvait saisir des caractères sans les afficher à l'écran. Le MSX-DOS a comblé cette lacune par les deux fonctions 07 et 08. Cependant, elle conserve son utilité, car c'est la seule fonction qui scrute le clavier sans attendre la frappe d'une touche. Nous allons montrer un programme qui affiche la touche enfoncée tant qu'elle le reste. La touche ESCape permet de sortir du programme.

```
                      ORG   0100

0100   1EFF    START : LD    E, 0FF    ; Fonction INPUT
0102   0E06          LD    C, 06     ;
0104   CD0500        CALL  0005      ;
0107   5F            LD    E, A      ; Fonction OUTPUT
```

```
0108    CD0500          CALL    5           ;
010B    FE1B            CP      1B          ; Check ESCAPE
010D    20F1            JR      NZ, START   ; Non → START
010F    C30000          JP      0000        ; Oui → Retout DOS
```

7.5.8 Fonction 07 MSX-DOS DIRECT INPUT

La fonction 07 permet d'attendre la frappe d'un caractère au clavier sans afficher le caractère frappé et sans tester les touches CTRL-C, CTRL-P, CTRL-N ou CTRL-S.

En entrée : Rien
En sortie : Le registre A contiendra le code de la touche frappée.
Préserve : Les registres C, D et E sont préservés.
Compatible : Non. La fonction 07 en CP/M s'appelle GET IO BYTE. Cette fonction CP/M n'est pas implémentée en MSX-DOS. Cependant, vu le fait que cette fonction CP/M est employée vraiment très rarement, il est peu probable qu'un programme CP/M soit incompatible avec le MSX du à cela.

Exemple : Lorsqu'on désire protéger un programme contre son emploi par une personne non autorisée, il est usuel d'employer la technique du mot de passe. Cependant, il est important lors de la frappe de ce mot de passe que des yeux indiscrets ne puissent voir à l'écran ce qui vient d'être tapé. Voici un programme qui demande un mot de passe (MSX) avant d'afficher l'écran.

```
                        ORG     0100

0100    0E09    START : LD      C, 9        ; Affichage de
0102    113701          LD      DE, MES     ;  « MOT DE PASSE : »
0105    CD0500          CALL    0005        ;
0108    0603            LD      B, 3        ; Boucle pour lire
010A    213101          LD      HL, BUF     ; 3 caractères
010D    C5      RPT :   PUSH    BC          ; Sauve compteur boucle
010E    E5              PUSH    HL          ; Sauve pointeur
010F    0E07            LD      C, 07       ; Lire un code sans
0111    CD0500          CALL    0005        ; Echo à l'écran
0114    E1              POP     HL          ; Rappel pointeur
0115    77              LD      (HL), A     ; Sauve code
0116    23              INC     HL          ; Incrémente pointeur
0117    C1              POP     BC          ; Rappel compteur boucle
0118    10F3            DJNZ    RPT         ; Répète 3 fois
```

```
011A  0603          LD    B, 3        ; Compare mot de
011C  113401        LD    DE, PASS    ; Passe posé avec
011F  2B      CHK : DEC   HL          ; « MSX »
0120  1A            LD    A, (DE)     ;
0121  BE            CP    (HL)        ;
0122  20DB          JR    NZ, START   ; Si différent → START
0124  13            INC   DE          ;
0125  10F8          DJNZ  CHK         ;
0127  1E0C          LD    E, 0C       ; Si égal = envoi CLS
0129  0E02          LD    C, 02       ;
012B  CD0500        CALL  0005        ;
012E  C30000        JP    0000        ; Retour au DOS
0131  000000  BUF : DEFS  3           ; Tampon lecture
0134  58534D  PASS: DEFB  'XSM'       ; Password inversé
0137  0D0A    MES : DEFB  0D, 0A      ; Message initial
0139  4D4F5420      DEFB  'MOT '      ;
013D  444520        DEFB  'DE '       ;
0140  50415353      DEFB  'PASSE :'   ;
0143  453A
```


7.5.9 Fonction 08 DIRECT INPUT WITH TEST

La fonction 08 attend la frappe d'un caractère au clavier sans afficher le caractère frappé et vérifie si un des codes CTRL-C, CTRL-P ou CTRL-N a été généré auquel cas la fonction qui y est associée est exécutée.

En entrée : Rien
En sortie : Le registre A reçoit le code du caractère posé au clavier.
Préserve : Les registres C, D et E sont préservés.
Compatible : Non. La fonction 08 en CP/M s'appelle SET IO BYTE. Cette fonction CP/M n'est pas implémentée en MSX-DOS. Cependant, vu le fait que cette fonction CP/M est employée vraiment très rarement, il est peu probable qu'un programme CP/M soit incompatible avec le MSX pour cette raison.

Exemple à : Vous pouvez modifier l'exemple de la fonction précédente en remplaçant le code 07 l'adresse 110H par 08. La seule différence de fonctionnalité de ce nouveau programme sera qu'il autorise de stopper le programme par CTRL-C au moment où le mot de passe est demandé.

7.5.10 Fonction 09 STRING OUTPUT

La fonction 09 permet d'afficher à l'écran une chaîne de caractères. Cette chaîne peut être constituée de n'importe quel caractère du code MSX, y compris les codes de fonction ou les séquences ESCape. Cependant, elle doit se terminer par le caractère $ (code 24H) qui marque donc la fin de la chaîne et de ce fait ne sera pas affiché. Si vous devez absolument afficher ce code $, alors utilisez la fonction 02 . Il n'y a pas non plus de limite de longueur à la chaîne comme en BASIC. Cette fonction teste également si un des codes CTRL-C, CTRL-P, CTRL-N ou CTRL-S a été généré par le clavier et le cas échéant exécute la fonction qui y est associé.

En entrée : Le registre de 16 bits DE doit contenir l'adresse mémoire où commence la chaîne. Ne pas oublier de terminer la chaîne par le code $.
En sortie : Les registres A et L contiendront le code $ (24H). Le registre DE contiendra l'adresse du code $ plus un.
Préserve : Seul le registre C est préservé.
Compatible : Oui.

Exemple : Nous allons montrer comment effacer l'écran, afficher un titre au milieu de la ligne, le souligner et mettre le curseur en mode souligné sur la dixième ligne et toute cela en un seul appel de la fonction. Voici son équivalent BASIC.

```
10 PRINT CHR$(12) ;              'CLS
20 A = PEEK(&HF3B0)              'A = largeur de l'écran
30 A = A\2 – 2                   'A = 1/2 largeur -2
40 LOCATE A, 0                   '
50 PRINT « TITRE »               '
60 LOCATE A, 1                   '
70 PRINT « ----- »               '
80 LOCATE 0,10                   '
90 PRINT CHR$(27) ;'y4' ;        ' Curseur mode souligné
```

```
                        ORG     0100

0100    3AB0F3  START : LD      A, (0F3B0)      ; A = largeur écran
0103    0F              RRCA                    ; A = A\2
0104    C620            ADD     A, 1E           ; A = A+1E
0106    321B01          LD      (COL), A        ; Sauve X (milieu) ->
0109    322401          LD      (COL1), A       ; COL et COL1
010C    111101          LD      DE, CLS         ; Affichage titre
010F    0E09            LD      C, 09           ;       complet
0111    CD0500          CALL    0005            ;
0114    C30000          JP      0000            ; Retour au DOS
```

```
0117  0C        CLS :   DEFB  0C          ; Code CLS
0118  1B59      ESCY :  DEFB  1B, 59      ;     ESC-Y
011A  20        LIN :   DEFB  20          ;     Ligne      0
011B  20        COL :   DEFB  20          ;     Milieu ligne
011C  544954    TITRE : DEFB  'TITRE'     ;     TITRE
011F  5245                                ;
0121  1B59      ESCY1 : DEFB  1B, 59      ;     ESC-Y
0123  21        LIN1 :  DEFB  21          ;     Ligne      1
0124  20        COL1 :  DEFB  20          ;     Milieu lign1
0125  2D2D2D    TIRET : DEFB  '-----'     ;
0128  2D2D                                ;
012A  1B59      ESCY2 : DEFB  1B, 59      ;      ESC-Y
012C  2A        LIN2 :  DEFB  2A          ;     Ligne      10
012D  20        COL2 :  DEFB  20          ;     Position    0
012E  1B7934    CUR_ :  DEFB  1B, 79, 34  ; ESC-y-4 = cur. fin
0130  24        FIN :   DEFB  '$'         ; $ = fin de chaîne
```

Les instructions de 0100 à 0109 sont destinées à diviser le nombre de positions dans la ligne (F3B0) par deux et sauver le résultat dans les deux séquences ESC-Y chargées de positionner le titre et son soulignement au milieu de la ligne.

Les instructions de 010C à 0114 servent à réellement afficher la chaîne et à stopper le programme. La chaîne commence à l'adresse 0117. Elle est composée d'abord du code 0C qui efface l'écran suivi d'une séquence ESCape-Y qui permet de localiser le curseur sur la ligne LIN, en colonne COL. Ensuite vient le titre proprement dit avec le curseur sous le titre pour le souligner. Enfin une troisième séquence ESCape-Y positionne le curseur en ligne 10, position 0 suivie de la séquence ESCape-y-4 qui donne au curseur la même forme qu'en mode insertion en BASIC. Finalement, le code $ termine la chaîne.

7.5.11 Fonction 0A STRING INPUT

La fonction 0A permet d'introduire en mémoire une chaîne complète jusqu'à ce que la touche Return soit enfoncée. Les caractères posés au clavier seront placés en mémoire à l'adresse indiquée par le registre DE+2. Le nombre maximum de caractères qui pourront être posés doit être placé en mémoire à l'adresse DE et ne peut dépasser 255. Dès qu'on tente de poser plus de caractères que le nombre spécifié par l'adresse DE, les caractères en surplus seront refusés et un BIP sonore sera émis. Seul le code Return sera accepté alors pour terminer l'entrée. Chaque caractère posé sera affiché à l'écran. Un test des codes CTRL-C, CTRL-P ou CTRL-N sera utilisé et les fonctions associées exécutées le cas échéant.

En entrée : Le registre DE doit pointer sur une adresse mémoire dont le contenu doit fixer le nombre maximum de caractères à lire.

En sortie : - La position mémoire (DE+1) indiquera combien de caractères ont effectivement été posés à l'exclusion du Return final. Attention que certains caractères graphiques utilisent deux codes (01-xx).
- La position mémoire (DE+2) contient le premier caractère posé de la chaîne. C'est donc l'endroit du début du tampon mémoire de réception. Une copie de la chaîne est également implantée à l'adresse F459.
- Les registres A et L contiendront le nombre de caractères lus si on a posé le nombre maximum de caractères, sinon ils contiendront le code Return (0D).
- Les registres B, C et H retournent la valeur 00.
- Le registre IX contiendra l'adresse du tampon 'copie' plus le nombre de caractères posés (F459 + nb).
- Le flag Z sera actif si le nombre de caractères posés a atteint le maximum sinon Carry sera actif.
- Le tampon mémoire pointé par (DE+2) se terminera par le code 0D si le nombre de caractères posés n'a pas atteint le maximum sinon il se terminera par le dernier caractère posé.

Préserve : Le registre DE seul est préservé.

Compatible : Oui.

Exemple : Prenons comme exemple le simple programme BASIC suivant mais nous limiterons le nombre de caractères posés à 40.

```
10 INPUT A$
20 PRINT A$
30 GOTO 10
```

```
                        ORG     C000

C000    1129C0  START : LD      DE, MAX     ; DE = adresse de MAX
C003    3E28            LD      A, 28       ; Mettre 40 dans MAX
C005    12              LD      (DE), A     ;
C006    0E0A            LD      C, 0A       ; INPUT A$
C008    CD7DF3          CALL    0F37D       ;
C00B    13              INC     DE          ;
C00B    1A              LD      A, (DE)     ; A = nb de caractères posés
C00D    13              INC     DE          ;
C00E    83              ADD     A, E        ; A est ajouté à l'
C00F    5F              LD      E, A        ; adresse des données
C010    1001            JR      NC, NEXT    ; = fin du tampon
C012    14              INC     D           ;
C013    3E24    NEXT :  LD      A, '$'      ; Mettre $ à la fin
```

```
C015  12              LD     (DE), A     ; de la chaîne
C016  112BC0          LD     DE, BUF     ; PRINT A$
C019  0E09            LD     C, 09       ;
C01B  CD7DF3          CALL   0F37D       ;
C01E  112A01          LD     DE, CLRF    ; Affiche CR-LF
C021  00E9            LD     C, 09       ;
C023  CD7DF3          CALL   0F37D       ;
C026  C300C0          JP     START       ; Recommence
C029  00        MAX : DEFB   0           ; nombre de caractères à lire
C02A  00        CNT : DEFB   0           ; nombre de caractères lus
C02B            BUF : DEFS   OFF         ; tampon de 255 positions
C12A  0D0A24    CRLF: DEFB   0D, 0A, 24  ; codes CR-LF-$
```

Les instructions en C003 et C005 permettent de fixer le nombre de caractères maximum demandés. Les instructions de C00B à C015 servent à placer le code $ dans le buffer juste après le dernier code posé en se servant de (DE+1) qui indique le nombre de caractères qui ont été posés. C016 à C01B affichent la chaîne et C021 à C023 provoquent le retour à la ligne.

7.5.12 Fonction 0B GET CONSOLE STATUS

La fonction 0B examine l'état du clavier. Si un caractère a été posé, alors le registre A contient FF sinon il contient 00. Pour obtenir le code du caractère posé, employez la fonction 01. Cette fonction teste aussi les codes CTRL-S, CTRL-P, CTRL-N et CTRL-C.

En entrée : Rien.
En sortie : Les registres A et L seront à 00 et le flag Z à 1 si aucun code n'a été posé. Les registres A et L contiendront FF et le flag Z sera à 0 dans le cas contraire.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui.

Exemple : Le programme suivant est équivalent à la routine BASIC suivant (vous pouvez l'arrêter par CTRL-C).

```
10 A$ = INKEY$
20 IF A$ = « »  GOTO 10
30 PRINT A$
40 GOTO 10
```

```
                      ORG    0100

0100  0E0B      TEST: LD     C, 0B       ; Test clavier
0102  CD0500          CALL   0005        ;
```

```
0105    B7              OR      A           ; A = 00 ?
0106    28F8            JR      Z, TEST     ; Oui → TEST>
0108    0E01    INPUT : LD      C, 01       ; Non → lit le
010A    CD0500          CALL    0005        ; code
010D    5F      PRINT : LD      E, A        ; Affiche le
010E    0E02            LD      C, 02       ; code
0110    CD0500          CALL    0005        ;
0113    C30001  GOTO :  JP      TEST        ; Recommence
```

7.5.13 Fonction 0C GET VERSION NUMBER

Cette fonction n'apparaît dans le MSX-DOS que par compatibilité avec le CP/M. Elle permet, en CP/M, de connaître la version de software utilisée. Comme le MSX-DOS est compatible avec la version 2.2 du CP/M, cette fonction place 00 dans le registre H et 22H dans le registre L. Cependant, elle n'est pas prévue pour refléter des versions différentes du MSX-DOS.

En entrée : Rien
En sortie : Le registre H indique 00.
Le registre L indique 22H.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui.

7.5.14 Fonction 0D DISK RESET

La fonction 0D sélectionne le disque A : comme disque par défaut, place la DTA ou DMA à l'adresse par défaut (0080H), écrit le tampon secteur sur le disque s'il s'avère modifié et ré-écrit les FATs de tous les disques si elles apparaissent modifiées. Attention pour le travail en BASIC-DOS, la valeur par défaut du DMA buffer (0080H) ne convient pas et doit donc être modifiée par la fonction 1A à partir de 8000H.

En entrée : Rien.
En sortie : Les registres A et L contiennent 00. Les registres DE et IX contiennent l'adresse du DPB (Disk Parameter Block) du dernier disque de votre système.
Préserve : Seul le registre C n'est pas modifié.
Compatible : Oui.

Exemple : Il est recommandé d'employer cette fonction lors de s programmes qui nécessitent des permutations de disquettes afin d'éviter d'écrire des secteurs sur le mauvais disque. De même, lorsqu'un programme a sélectionné un autre disque par défaut, il est souhaitable d'employer cette fonction avant de quitter le programme pour rétablir le système comme à l'origine.

```
                        ORG     300

0300    0E0D    START : LD      C, 0D           ; Reset des disques
0302    CD0500          CALL    5               ;
0305    0E09            LD      C, 09           ; Affiche message d'
0307    111003          LD      DE, MES         ; échange disques
0000    CD0500          CALL    5               ;
0000    C32403          JP      SUITE           ; Suite programme
0310    45434841  MES : DEFB    'ECHANGEZ'      ;
0314    4E47455A                                ;
0318    4C455320        DEFB    'LES '          ;
0000    44495351        DEFB    'DISQUES'       ;
0320    5545532E                                ;
0324    …..     SUITE : ….                      ;
```

7.5.15 Fonction 0E SELECT DEFAULT DISK

Cette fonction permet de désigner le disque par défaut. En CP/M et en MSX-DOS, les fonctions manipulant les disques font référence à un numéro de disque plutôt qu'au nom du disque (A : à H:). Ces numéros vont de 0 à 8, soit neuf possibilités. Le disque A : est référencé par 1, le B : par 2 et ainsi de suite jusqu'à 8 pour le disque H :. Le numéro 0 référence le disque par défaut. Cette fonction permet donc de déterminer quel sera ce disque par défaut. Cette fonction permet aussi de déterminer le nombre de disques logiques dont vous disposez sur le système.

En entrée : Le registre E doit indiquer quel disque sera dorénavant le disque par défaut suivant la table que voici : A:=0 B:=1 C:=2 D:=3 E:=4 F:=5 G:=6 H:=7. Attention qu'il s'agit ici d'une numérotation propre à cette fonction où 0 vaut pour le disque A : et non pour le disque par défaut.

En sortie : Le registre A indique le nombre de disques logiques connectés au système. Ainsi, si vous avez un seul disque physique et que vous avez démarré l'ordinateur sans avoir appuyé sur CTRL, le registre A contiendra 02 car dans ce cas votre unique disque sera accessible tant par le disque logique A : que par le B : en permutant les disquettes. Si vous tentez de sélectionner un disque logique qui n'existe pas sur votre système, alors le flag Carry sera mis à 1, sinon il sera à 0. La position mémoire F247 reflétera le disque par défaut.

Préserve : Les registres C, D et E sont préservés.

Compatible : Oui, excepté qu'en CP/M le nombre de disques n'est pas indiqué dans le registre A en sortie.

Exemple : Si vous employez ce court programme qui sélectionne le disque B : comme disque par défaut et qui retourne ensuite à l'indicatif du MSX-DOS vous verrez que cet indicatif ne sera plus A> mais B> et ainsi tout programme appelé sera recherché sur le disque B:.

```
                        ORG     0100

0100    10      START : LD      E, 01       ; Disque B :
0102    0E0E            LD      C, 0E       ; Fonction 0E
0104    CD0500          CALL    0005        ; Appel DOS
0107    C30000          JP      0000        ; Retour au DOS
```

7.5.16 Fonction 0F OPEN FILE

La fonction 0F permet d'ouvrir un fichier existant déjà sur le disque. Pour créer le fichier voyez la fonction 16 MAKE. L'ouverture d'un fichier en CP/M ou en MSX-DOS se fait par la référence au FCB (File Control Block) de ce fichier. Revoyez le FCB dans le chapitre 5. Le FCB est une zone de 37 caractères implantée en mémoire à une adresse choisie par le programmeur. On peut ouvrir des fichiers tant en mode séquentiel qu'en mode direct. Avant d'ouvrir un fichier, le programmeur doit remplir les 16 premiers octets du FCB qui représentent les zones suivantes du FCB :

DR	Numéro de disque avec ou sans bit 8.
FNAME	Nom du fichier en 8 octets.
FEXT	Extension du fichier en 3 octets.
EX	Numéro de l'extent en mode séquentiel. En général, on y met toujours 0000, mais on peut ouvrir un fichier directement sur un extent donné (1 extent = 128 records = 16K).
RECSIZ	Doit toujours être mis à 00.

Il est possible d'ouvrir un fichier caché (voir fonction CREATE 16) en positionnant le bit 8 dans l'octet DR du FCB. Le reste du FCB sera automatiquement mis à jour par la fonction OPEN. Il s'agit des zones suivantes :

FILSIZ	Indiquera la longueur du fichier en nombre d'octets.
DATE	Indiquera la date de création ou de dernière modification du fichier.
TIME	Indiquera l'heure de création ou de dernière modification du fichier sur les MSX2.
DEVID	Donnera l'identificateur du type de périphérique.
DIRLOC	Donnera la position du fichier dans le Directory.
STRCLS	Donnera le Start Cluster du fichier.
CURCLS	Donnera le Current Cluster.
CLSOFF	Donnera le décalage en clusters par rapport au début du fichier.

Les zones CR (current record) et RN (record number) sont laissées telles quelles c'est-à-dire non initialisées par la fonction OPEN.

En entrée : Le registre DE doit contenir l'adresse du FCB.

En sortie : Les registres A et L contiendront 00 et le CARRY flag sera à 0 si l'ouverture s'est déroulée sans erreur. Les registres A et L contiendront FF et le CARRY flag sera à 1 si l'ouverture s'est soldée par une erreur telle que :

- disque inexistant
- disquette absente
- fichier inexistant

Le FCB sera mis à jour comme décrit plus haut et en plus l'octet RC indiquera combien de records sont disponibles dans l'extent EX ouvert. Ce nombre ca de 1 à 80H au maximum. Si l'extent ouvert ne contient pas encore de record, alors RC vaudra 00. Si RC donne un nombre entre 01 et 7F, c'est que cet extent est le dernier du fichier. Ce renseignement n'est utile qu'en mode séquentiel lorsqu'on désire ouvrir un fichier directement sur l'extent EX pour pouvoir lire ou écrire cet extent en séquence sans devoir lire tous les enregistrements des premiers extents.
Le registre DE contiendra l'adresse du dernier octet mis à jour dans le FCB.
Le registre IX contiendra l'adresse du DPB (Disk Parameter Block) du disque contenant le fichier.

Préserve : Seul le registre C est préservé.
Compatible : Oui

Exemple : Voir la fonction suivante.

7.5.17 Fonction 10 CLOSE

La fonction 10 permet de fermer un fichier. Le fichier sera référencé par son FCB dont l'adresse sera indiquée par le registre DE. La fonction Close est particulièrement importante lorsqu'on a procédé à des écritures dans le fichier. En effet, c'est à ce moment que les données du dernier enregistrement sont écrites dans le fichier et que les données du FCB sont réécrites dans le Directory. Les secteurs FAT sont également mis à jour sur le disque à partir des tampons en mémoire. Si le fichier a seulement été lu, vous pouvez omettre le « close » de ce fichier.

En entrée : Le registre DE doit contenir l'adresse du FCB. Le FCB a été composé au moment de l'Open du fichier et mis à jour par les fonctions de lecture et écriture du fichier. Il ne faut normalement pas y toucher.
En sortie : Les registres A et L contiendront 00H et le CARRY flag sera à 0 si la fermeture du fichier se fait sans erreur sinon A et L contiendront FF et le CARRY flag sera à 1.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui.

Exemple : Ce petit programme va ouvrir un fichier appelé TEXT.ASC et afficher à partir des données du FCB, sa longueur et sa date de création en hexadécimal. Il sera ensuite refermé. Si le fichier n'est pas trouvé, alors on reviendra immédiatement à l'indicatif du MSX-DOS.

```
                        ORG     0100

0100    116B01  OPEN :  LD      DE, FCB         ; Ouverture fichier
```

```
0103    0E0F            LD      C, 0F           ;
0105    CD0500          CALL    0005            ; Si erreur
0108    DA0000          JP      C, 0000         ;           → 0000
0000    115601  LONG :  LD      DE, LOMES       ; Affiche « LONGUEUR »
0000    0E09            LD      C, 09           ;
0110    CD0500          CALL    0005            ;
0113    2A7A01          LD      HL, (FCB+12)    ; Affiche en hexa
0116    CD3B01          CALL    HEXPR           ; la longueur
0119    2A7A01          LD      HL, (FCB+10)    ; donnée par FCB+12
0000    CD3B01          CALL    HEXPR           ; et FCB+10
0000    116101  DATE :  LD      DE, DAMES       ; Affiche « DATE »
0122    0E09            LD      C, 09           ;
0124    CD0500          CALL    0005            ;
0127    2A7C01          LD      HL, (FCB+14)    ; Affiche la date
0000    CD3B01          CALL    HEXPR           ; présente en DCB+14
0000    116B01  CLOSE : LD      DE, FCB         ; Fermeture fichier
0130    0E10            LD      C, 10           ;
0132    CD0500          CALL    005             ;
0135    C30000          JP      0000            ; Et retour au DOS
0138    7C      HEXPR : LD      A, H            ; Routine d'affichage
0139    CD3D01          CALL    HEX1            ; en hexadécimal
0000    7D              LD      A, L            ; du contenu de HL
0000    F5      HEX1 :  PUSH    AF              ;
0000    0F              RRCA                    ;
0000    0F              RRCA                    ;
0140    0F              RRCA                    ;
0141    0F              RRCA                    ;
0142    CD4601          CALL    HEX2            ;
0145    F1              POP     AF              ;
0146    E60F    HEX2 :  AND     0F              ;
0148    C630            ADD     A, 30           ;
0000    FE3A            CP      3A              ;
0000    3802            JR      C, HEX3         ;
0000    C607            ADD     A, 7            ;
0150    5F      HEX3 :  LD      E, A            ;
0151    0E02            LD      C, 02           ;
```

```
0153    C30500              JP      0005            ;
0156    0D0A4C4F  LOMES :   DEFB    0D, 0A, 'LONGUEUR : ', 24
0000    4E475545                                    ;
0000    555224                                      ;
0161    0D0A4441  DAMES :   DEFB    0D, 0A, 'DATE : ', 24
0165    555224                                      ;
0168    544558    FCB :     DEFB    0, 'TEXT    ASC'
0000    54202020                                    ;
0170    20415343                                    ;
0174    00000000            DEFW    0, 0, 0, 0, 0, 0    ;
0178    00000000                                    ;
0000    00000000                                    ;
0180    00000000            DEFW    0, 0, 0, 0, 0, 0    ;
0184    00000000                                    ;
0188    00000000                                    ;
```

Commentaire : Pour ne pas trop allonger le programme, l'affichage de la longueur et de la date se fait en hexadécimal par la routine HEXPR qui affiche le contenu du registre HL à l'écran. Les positions réservées à la longueur du fichier sont FCB+10 à FCB+13 (voir le FCB dans le chapitre 4). Celles réservées à la date occupent les positions FCB+14 et FCB+15. Remarquez l'initialisation du FCB en 168 : toutes les positions non-utilisées ont été initialisées à 0 par les instructions DEFW. Pour essayer ce programme, n'oubliez pas de créer un petit fichier TEXT.ASC.

7.5.18 Fonction 11 SEARCH FIRST

La fonction 11 permet de rechercher dans le Directory un fichier dont le FCB est précisé par le registre DE. Si ce fichier existe dans le Directory, les 32 octets du Directory relatifs à ce fichier sont copiés dans le DMA buffer (Défaut = 0080H) et les registres A et L sont positionnés à 00 et le CARRY flag est mis à zéro.

Si le fichier n'existe pas, les registres A et L retournent la valeur FFH et le CARRY flag est mis à 1.

On peut utiliser les caractères de substitution dans le nom du fichier présent dans le FCB. Dans ce cas, cette fonction recherche le premier fichier, à partir du début du Directory, coïncidant avec le nom présent dans le FCB.

En entrée : Le registre DE doit pointer sur le FCB du fichier. Le FCB doit être composé avant l'appel de cette fonction et peut contenir des caractères de substitution.

En sortie : Les registres A et L contiendront 00 si le fichier est trouvé dans le Directory. Les registres A et L contiendront FF si le fichier n'est pas trouvé dans le

Directory. Le DTA ou DMA buffer contiendra les 32 octets du Directory relatifs à ce fichier avec en plus les indications suivantes :

DTA+12 = numéro d'extent tel que précisé dans la position FCB+12
DTA+13 = attribut du fichier tel qu'indiqué dans l'entrée Directory + 12. Cet attribut est aussi placé dans le registre B.
DTA+14 = 0
DTA+15 = nombre de records (128 octets) présents dans l'extent DTA+12. Cette valeur est aussi placée dans le registre C.

Préserve : Aucun registre n'est préservé. Par contre, l'adresse du FCB est sauvegardée à la position F3074H en RAM pour être réutilisée par la fonction suivante SEARCH NEXT code 12.

Compatible : Oui.

Exemple : Voir fonction suivante.

7.5.19 Fonction 12 SEARCH FOR NEXT

La fonction 12 est complémentaire à la fonction 11 précédente. Elle permet de rechercher la prochaine occurrence dans le Directory du fichier dont le FCB a été précisé lors de la fonction précédente SEARCH FIRST 11. Comme l'on peut utiliser des caractères de substitution, les fonctions 11 et 12 forment une paire permettant entre autres de visualiser le contenu du Directory. A fonction 11 recherchant la première occurrence du nom de fichier et des fonctions 12 successives recherchant les occurrences suivantes.

En entrée : Aucun paramètre n'est nécessaire, mais cette fonction doit avoir été précédée de la fonction 11 qui fixe le FCB du/des fichiers recherchés.

En sortie : Voyez la fonction 11.

Préserve : Rien.

Exemple : Ce petit programme va vous montrer comment réaliser la fonction DIR ou FILES en langage machine. La fonction 1A au début du programme permet de fixer l'adresse de la DTA et sera vue plus loin. Tous les fichiers du Directory seront affichés y compris les fichiers dits cachés.

```
                        ORG     C000

C000    0E1A    DMA :   LD      C, 1A       ; Place le tampon DMA
C002    1100B0          LD      DE, 0B000   ; en B000H
C005    CD7DF3          CALL    0F37D       ;
C008    0E11    SFIRST: LD      C, 11       ; Search first file
C00A    1143C0          LD      DE, FCB     ;
C00D    CD7DF3          CALL    0F37D       ;
C010    380F            JR      C, END      ; Pas trouvé → END
C012    CD26C0          CALL    PRINT       ; Affiche nom fichier
C015    0E12    SNEXT : LD      C, 12       ; Search Next File
```

```
C017  C37DF3          CALL  0F37D        ;
C01A  3B05            JR    C, END       ; Pas trouvé → END
C01C  CD26C0          CALL  PRINT        ; Affiche nom fichier
C01F  18F4            JR    SNEXT        ; Répète la recherche
C021  0E00    END :   LD    C, 0         ; Retour au BASIC
C023  C37DF3          JP    0F37D        ;
C026  2101B0  PRINT : LD    HL, 0B001    ; Affiche nom fichier
C029  060C            LD    B, 0C        ; Boucle = 12 caractères
C02B  5E      PRINT1: LD    E, (HL)      ; Lire caractère nom → E
C02C  23              INC   HL           ; Pointe sur suivant
C02D  C5              PUSH  BC           ; Sauve compteur boucle
C02E  E5              PUSH  HL           ; Sauve pointeur
C02F  0E02            LD    C, 2         ; Affiche registre E
C031  C37DF3          CALL  0F37D        ;
C034  E1              POP   HL           ; Rétablit HL
C035  C1              POP   BC           ; Rétablit BC
C036  10F3            DJNZ  PRINT1       ; Répète 12 fois
C038  0E09    CRLF :  LD    C, 09        ; Affiche CR-LF
C03A  1140C0          LD    DE, CR       ;
C03D  C37DF3          JP    0F37D        ;
C040  0D0A24  CR :    DEFB  0D, 0A, 24   ; Codes CR-LF-$
C043  00      FCB :   DEFB  00           ; disque défaut
C044  3F3F3F3F        DEFB  ‘ ????’      ; nom fichier =
C048  3F3F3F3F        DEFB  ‘ ????’      ;
C04C  3F3F3F          DEFB  ‘ ???’       ;
C04F  00000000        DEFB  0, 0, 0, 0   ; ????????. ???
C053  00000000        DEFB  0, 0, 0, 0   ; Efface reste du
C057  00000000        DEFB  0, 0, 0, 0   ; FCB.
C05B  00000000        DEFB  0, 0, 0, 0   ;
C05F  00000000        DEFB  0, 0, 0, 0   ;
C063  00000000        DEFB  0, 0, 0, 0   ;
```

7.5.20 Fonction 13 DELETE FILE

La fonction 13 permet d’enlever un fichier du Directory de la disquette. Cette opération n’efface pas le contenu réel du fichier mais supprime tout accès à ce fichier en remplaçant le premier caractère du nom de fichier dans le Directory par le code E5H et en libérant dans la FAT tous les clusters précédemment alloués à ce fichier. On peut effacer plusieurs fichiers à la fois en utilisant

les caractères de substitution. Le fichier à effacer sera référencé par son FCB dont l'adresse doit être fournie dans le registre DE.

En entrée : Le registre DE doit contenir l'adresse du FCB du fichier à effacer.

En sortie : Les registres A et L contiendront 00H si l'opération se déroule sans erreur. Les registres A et L contiendront FFH si l'opération se termine par une erreur telle que :
- la disquette est absente.
- le fichier n'existe pas.

Préserve : Aucun registre n'est préservé.

Compatible : Oui

Exemple : Ce petit programme efface tous les fichiers portant l'extension .TMP

```
                        ORG   0100

0100  0E13      START : LD    C, 13          ; Fonction Delete
0102  210B01            LD    DE, FCB        ; DE = FCB
0105  CD0500            CALL  0005           ; Appel fonction
0108  C30000            JP    0000           ; Retour au DOS
0000  00        FCB :   DEFB  0              ; Disque courant
010C  3F3F3F3F          DEFB  ' ????????'    ; NOM = *
0110  3F3F3F3F                                ;
0114  544D50            DEFB  'TMP'          ; extension = TMP
0117  00000000          DEFB  0, 0, 0, 0     ; Efface reste FCB
011B  00000000          DEFB  0, 0, 0, 0     ;
011F  00000000          DEFB  0, 0, 0, 0     ;
0123  00000000          DEFB  0, 0, 0, 0     ;
0127  00000000          DEFB  0, 0, 0, 0     ;
```

7.5.21 Fonction 14 SEQUENTIAL READ

La fonction 14 permet de lire un enregistrement de 128 octets (standard CP/M) à partir d'un fichier et de transférer cet enregistrement en mémoire dans la DTA. Le fichier à lire doit avoir été préalablement ouvert par la fonction OPEN (0F). Le registre DE doit préciser l'adresse de ce FCB.

Les enregistrements seront lus séquentiellement à partir de l'enregistrement 0. Donc à chaque appel de cette fonction l'enregistrement suivant sera transféré vers le DTA. Cet accè_s séquentiel est obtenu par le fait que la routine va automatiquement incrémenter les positions CR (Current Record) et EX (Extent) du FCB.

Cependant, il y a moyen de lire directement un enregistrement X en plaçant soi-même les zones CR et EX comme décrit ci-dessous après l'ouverture du fichier ou à chaque fois que l'on désire rompre la séquence de lecture.

L’enregistrement à lire sera défini par les zones EX et CR du FCB. EX définit dans quel extent (1 extent = 16K = 128 enregistrements de 128 octets) se trouve l’enregistrement et CR définit le numéro de l’enregistrement dans cet extent (de 0 à 127).

En entrée : Le registre DE doit pointer sur le fichier ouvert dont on désire lire un enregistrement.

En sortie : Les registres A et L contiendront 00H si l’opération s’est déroulée sans erreur. Les registres A et L contiendront 01H si l’opération s’est terminée par une erreur telle que Fin de fichier ou tentative de lecture au-delà de la Fin de fichier par exemple. Le registre IY contiendra l’adresse du FCB et le registre IX l’adresse du DCB concerné.

Préserve : Cette fonction ne préserve aucun registre.

Compatible : Oui, d’ailleurs cette fonction n’a été placée en MSX-DOS que pour assurer la compatibilité avec le CP/M. En effet, la fonction MSX-DOS Random Block Read (fonction 27) offre beaucoup plus d’avantages et s’exécute bien plus rapidement que celle-ci.

Exemple : Ce petit exemple va vous montrer comment afficher un fichier ASCII de votre choix à l’écran. Appelez ce petit programme LIST.COM et pour afficher un fichier ASCII, posez simplement LIST suivi du nom de fichier après l’indicatif du DOS A>. Cependant, si ce fichier n’a pas exactement une longueur égale à un multiple de 128 octets, le dernier enregistrement sera suivi de caractères fantaisistes.

```
                        ORG     0100

0100    21C500  START : LD      HL, 5C          ; Copie argument dans
0103    114001          LD      DE, FCB         ; le FCB
0106    010C00          LD      BC, 0C          ;
0109    EDB0            LDIR                    ;
010B    0E0F    OPEN :  LD      C, 0F           ; Ouverture fichier
010D    114001          LD      DE, FCB         ;
0110    CD0500          CALL    0005            ; Si fichier n’existe
0113    DA0000          JP      C, 0000         ; pas, retour au DOS
0116    0E14    READ :  LD      C, 14           ; Lire un record
0118    00114001        LD      DE, FCB         ;
011B    CD0500          CALL    0005            ;
011E    B7              OR      A               ; Si erreur, aller à
011F    2014            JR      NZ, CLOSE       ; CLOSE
0121    2A3DF2  PRINT : LD      HL, (0F23D)     ; HL = adresse DTA
0124    0680            LD      B, 80           ; Boucle 128 fois
0126    E5      PRINT1 : PUSH   HL              ; Sauve pointeur DTA
0127    C5              PUSH    BC              ; Sauve boucle
```

```
0128  5E          PRINT2 :  LD    E, (HL)        ; Affiche un octet
0129  0E02                  LD    C, 02          ; du record
012B  CD0500                CALL  0005           ;
012E  C1                    POP   BC             ; Rappel boucle
012F  E1                    POP   HL             ; Rappel pointeur DTA
0130  23                    INC   HL             ; Incrémente HL
0131  10F5                  DJNZ  PRINT1         ; Répète la boucle
0133  1BE1                  JR    READ           ; Retour à la lecture
0135  0E10        CLOSE :   LD    C, 10          ; Fermeture fichier
0137  114001                LD    DE, FCB        ;
013A  CD0500                CALL  0005           ;
113D  C30000                JP    0000           ; Retour au DOS
0140  00000000    FCB :     DEFB  0,0,0,0,0,0,0,0 ; Le FCB sera rempli
0144  00000000              DEFB  0,0,0,0,0,0,0,0 ; par le premier
014B  00000000              DEFB  0,0,0,0,0,0,0,0 ; argument
014C  00000000              DEFB  0,0,0,0,0,0,0,0 ;
0150  00000000              DEFB  0,0,0,0,0,0,0,0 ;
0154  00000000              DEFB  0,0,0,0,0,0,0,0 ;
0158  00000000              DEFB  0,0,0,0,0,0,0,0 ;
015C  00000000              DEFB  0,0,0,0,0,0,0,0 ;
0160  00000000              DEFB  0,0,0,0,0,0,0,0 ;
```

7.5.22 Fonction 15 SEQUENTIAL WRITE

Cette fonction transfère le contenu de la DTA vers un enregistrement du fichier dont le FCB est précisé par le registre DE. Le fichier doit avoir été préalablement ouvert par la fonction OPEN (0F) ou créé par la fonction CREATE (16). Les enregistrements seront écrits séquentiellement cas la routine incrémente automatiquement les octets CR (Current Record) et EX (Extent) du FCB.

Il est possible d'écrire directement un enregistrement X en indiquant soi-même la valeur des octets CR et EX avant l'appel de cette fonction. Rappelons qu'il y a toujours 128 octets dans un enregistrement CP/M et qu'il y a 128 enregistrements dans un extent.

En entrée : Le registre DE doit pointer sur le FCB du fichier. Le DTA doit contenir les 128 octets de l'enregistrement. Les octets CR et EX du FCB doivent être positionnés sur le numéro d'enregistrement que vous désirez écrire si l'écriture doit se faire hors séquence.

En sortie : Les registres A et L contiendront 00H si l'opération s'est déroulée dans erreur. Les registres A et L contiendront 01H si l'opération s'est terminée par une erreur telle que disque plein, fichier non ouvert, etc. Le registre IY contiendra l'adresse du FCB. Le registre IX contiendra l'adresse du DCB.

Préserve : Cette routine ne préserve aucun registre.
Compatible : Oui, d'ailleurs cette fonction n'a été placée en MSX-DOS que pour assurer la compatibilité avec le CP/M. En effet, la fonction MSX-DOS Random Block Write (fonction 26) offre beaucoup plus d'avantages et s'exécute bien plus rapidement que celle-ci.

Exemple : Voyez l'exemple de la fonction CREATE suivante.

7.5.23 Fonction 16 CREATE FILE

La fonction CREATE FILE permet de créer un fichier qui n'existe pas encore dans le Directory du disque. Elle crée donc un fichier dont la longueur est de 0 octets et laisse ce fichier à l'état ouvert ce qui permet d'utiliser directement les fonctions 15 (Sequential Write), 22 (Random write), 26 (Random block write) et 28 (Random Write with zero fill).

Si le fichier existe déjà et que la position FCB+12 est placée à 00H, il est recréé ; on perd ainsi toutes les informations qu'il contenait.

Si le fichier existe déjà et que la position FCB+12 est différente de 00H, le contenu du fichier est conservé et le FCB est complété avec les informations du Directory. Donc, la date et l'heure de création, la longueur du fichier et son cluster de départ sont conservés. Cependant, si vous désirez agrandir ce fichier, n'oubliez pas de positionner les octets EX et CR à la valeur souhaitée, car cette opération les laisse dans l'état initial.

Cette fonction convient pour créer tous les types de fichier qu'ils soient de type séquentiel ou direct. Les paramètres de taille (RECSIZ) et de numéro d'enregistrement (CR et EX ou RN) seront placés dans le FCB après l'exécution de cette fonction.

Si le bit 8 du numéro de disque dans le FCB (DR) est à 1, la fonction va créer un fichier caché. L'octet Attribut du Directory (DIR+11) sera égal à 06H et dès lors le fichier ne sera pas trouvé lors des recherches dans le Directory (voir Directory dans le chapitre 4).

En entrée : Le registre DE doit contenir l'adresse du FCB du fichier. Le FCB doit avoir été complété avant l'appel de cette fonction.
En sortie : Les registres A et L contiendront 00H si l'opération se déroule dans erreur. Les registres A et L contiendront FFH si l'opération s'est terminée par une erreur telle que plus de place dans le Directory. Le registre IX donnera l'adresse du DCB.
Préserve : Aucun registre n'est préservé.
Compatible : Oui, excepté que la technique des fichiers cachés n'existe pas en CP/M.

Exemple : Voici un programme de copie du fichier placé comme premier argument vers un fichier dont le nom figure en deuxième argument. Si vous sauvez ce programme sous le nom COPIE,
A>COPIE B:LETTRE.ASC A:ARCHIVE.ASC
copiera le fichier LETTRE.ASC présent sur le disque B : vers un fichier ARCHIVE.ASC qui sera créé sur le disque A :.

```
ORG     0100
```

```
0100  21C500    START :  LD    HL, 5C      ; Copie argument 1
0103  114E01             LD    DE, FCB1    ; vers FCB1
0106  011000             LD    BC, 10      ;
0109  EDB0               LDIR              ;
010B  216C00             LD    HL, 6C      ; Copie argument 2
010E  117301             LD    DE, FCB2    ; vers FCB2
0111  011000             LD    BC, 10      ;
0114  EDB0               LDIR              ;
0116  0E0F     OPEN :    LD    C, 0F       ; Open fichier 1
0118  114E01             LD    DE, FCB1    ;
011B  CD0500             CALL  0005        ; Si erreur,
011E  DA0000             JP    C, 0000     ;          → 0000
0121  0E16     CREATE :  LD    C, 16       ; Create fichier 2
0123  117301             LD    DE, FCB2    ;
0126  CD0500             CALL  0005        ;
0129  B7                 OR    A           ; Si erreur,
012A  C20000             JP    NZ, 0000    ;          → 0000
012D  0E14     READ :    LD    C, 14       ; Lire un record du
012F  114E01             LD    DE, FCB1    ; fichier 1
0132  CD0500             CALL  0005        ;
0135  B7                 OR    A           ; Si erreur
0136  200B               JP    NZ, CLOSE   ;          → Close
0138  0E15     WRITE :   LD    C, 15       ; Ecrit record dans
013A  117301             LD    DE, FCB2    ; fichier 2
013D  CD0500             CALL  0005        ;
0140  B7                 OR    A           ; Si erreur
0141  28EA               JR    Z, READ     ;          → Close
0143  0E10     CLOSE :   LD    C, 10       ; Fermeture fichier 2
0145  117301             LD    DE, FCB2    ;
0148  CD0500             CALL  0005        ;
014B  C30000             JP    0000        ; Retour au DOS
014E  00       FCB1 :    DEFB  00          ; FCB copié du 1er
014F  00000000           DEFB  0, 0, 0, 0  ; argument
0153  00000000           DEFB  0, 0, 0, 0  ;
```

```
0157    00000000            DEFB  0, 0, 0, 0        ;
015B    00000000            DEFB  0, 0, 0, 0        ;
015F    00000000            DEFB  0, 0, 0, 0        ;
0163    00000000            DEFB  0, 0, 0, 0        ;
0167    00000000            DEFB  0, 0, 0, 0        ;
016B    00000000            DEFB  0, 0, 0, 0        ;
016F    00000000            DEFB  0, 0, 0, 0        ;
0173    00          FCB2 :  DEFB  00                ; FCB copié du 2ème
0174    00000000            DEFB  0, 0, 0, 0        ; argument
à       00000000            DEFB  0, 0, 0, 0        ;
0194    00000000            DEFB  0, 0, 0, 0        ;
```

7.5.24 Fonction 17 RENAME FILE

La fonction 17 permet de changer le nom d'un fichier spécifié par le FCB pointé par le registre DE dans le nom donné par ce même FCB mais en position FCB+16. La structure de ce FCB spécial devient donc :

```
Long: 1     8     3       4       1     8     3       4
      -- -------- --- ----------- -- -------- --- -----------
Nom: DR-FILENAME-EXT-00-00-00-00-DR-FILENAME-EXT-00-00-00-00
      --------------- ----------- --------------- -----------
Type:   Ancien nom    doit être    Nouveau nom    doit être
        du fichier      à 00H      du fichier       à 00H
```

L'octet DR (Drive) doit obligatoirement être identique dans les deux portions du FCB. FILENAME.EXT peut contenir des caractères de substitution tant dans l'ancien nom que dans le nouveau.

Dans l'ancien nom, les caractères de substitution permettent de trouver tous les fichiers du Directory coïncidant avec le nom fourni tandis que dans le nouveau nom, les caractères de substitution indiquent qu'il faut utiliser les caractères de l'ancien nom à chaque position où on rencontre un ' ?'.

En entrée : Le registre DE doit pointer sur le FCB dont les 16 premières positions contiennent l'ancien nom et les 16 suivantes le nouveau nom de fichier.
En sortie : Les registres A et L contiendront 00H et le Carry flag sera mis à 0 si l'opération se déroule sans erreur. Les registres A et L contiendront FFH en cas d'erreur. Si l'erreur est que le fichier à renommer n'est pas trouvé, alors le Carry flag sera également mis à 1.
Préserve : Aucun registre n'est préservé.
Compatible : Oui.

Exemple : Ce programme est l'équivalent de la commande BASIC
NAME « JEU ?.BAS » AS « ????.OLD »

```
                        ORG     0100

0100    0E17    START : LD      C, 17           ; Fonction 17
0102    11….            LD      DE, FCBOLD      ; De = FCB
0105    CD0500          CALL    0005            ; Appel DOS
0108    B7              OR      A               ; Si pas erreur,
0109    CA0000          JP      Z, 0000         ; retour au DOS
010C    0E09            LD      C, 09           ; Si erreur, affiche
010E    113701          LD      DE, ERRMES      ; message d'erreur
0111    CD0500          CALL    0005            ; et
0114    C30000          JP      0000            ; retour au DOS
0117    00      FCBOLD :DEFB    00              ; Disque par défaut
0118    4A45553F        DEFB    'JEU ?  '       ; Ancien nom
011C    20202020                                ;
0120    424153          DEFB    'BAS'           ;
0123    00000000        DEFB    0, 0, 0, 0      ;
0127    00      FCBNEW :DEFB    00              ; Disque par défaut
0128    3F3F3F3F        DEFB    ' ????  '       ; Nouveau nom
012C    20202020                                ;
0130    4F4C44          DEFB    'OLD'           ;
0133    00000000        DEFB    0, 0, 0, 0      ;
0137    0D0A    ERRMES :DEFB    0D, 0A          ; CR-LF
0139    45727275        DEFB    'Erreur'        ; Message d'erreur
013D    7572                                    ;
013F    0D0A0A24        DEFB    0D, 0A, 0A, 24  ; CR-LF-LF-$
```

7.5.25 Fonction 18 LOGGIN VECTOR

Cette fonction permet de connaître les disques présents sur votre système. Elle retourne dans le registre HL un code où chaque bit à l'état 1 indique la présence d'un disque on-line. Le bit 1 de L est réservé au disque A :, le bit 2 au disque B : et ainsi de suite jusqu'au bit 8 pour le disque H :.

En entrée : Rien
En sortie : Le registre H est à 00H et le registre L contient la table des disques on-line.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui, mais en MSX-DOS, tous les disques présents sur le système sont toujours on-line tandis qu'en CP/M, ils peuvent être off-line.

Exemple : Ce programme affiche les noms des disques présents sur votre système.

```
                        ORG     0100

0100    0E18    START : LD      C, 18
0102    CD0500          CALL    0005    ; HL = table disque
0105    0608            LD      B, 08   ; BC = 8 boucles
0107    7D              LD      A, L    ; A = table disque
0108    C5      START1 : PUSH   BC      ; Sauve compteur boucle
0109    F5              PUSH    AF      ; Sauve table disque
010A    E601            AND     01      ; isole 1 bit
010C    2808            JR      Z, START2 ; si 0, → start2
010E    0E09    PRINT : LD      C, 9    ; si 1, affiche
0110    112201          LD      DE, MES ; Disc X : présent
0113    CD0500          CALL    0005    ;
0116    212901  START2 : LD      HL, DISC ; Incrémente Nom de
0119    34              INC     (HL)    ; disque.
011A    F1              POP     AF      ; rappel table disque
011B    0F              RRCA            ; au suivant...
011C    C1              POP     BC      ; rappel compteur bouble
011D    10E9            DJNZ    START1  ; répète 8 fois
011F    C30000          JP      0000    ; Retour au DOS
0122    44697371 MES :  DEFB    'Disque '  ; Message
0126    756520                          ;
0129    41      DISC :  DEFB    41      ; « Disque X : présent »
012A    3A207073        DEFB    ' : présent'  ;
012E    8273656E                        ;
0132    74                              ;
0133    0D0A24          DEFB    0D, 0A, 24  ;
```

7.5.26 Fonction 19 GET DEFAULT DRIVE NAME

Cette fonction permet d'obtenir, dans le registre A, le numéro du disque par défaut. C'est donc la fonction inverse de Select Disk (0E) qui permet de défiir le disque par défaut. La valeur 0 est attribuée au disque A :, 1 au disque B : et ainsi de suite jusqu'à 7 pour le disque H :.

En entrée : Rien
En sortie : Les registres A et L indiquent le numéro du disque par défaut.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui.

Exemple : Ce programme affiche le nom du disque par défaut.

```
                        ORG   0100

0100  0E19      START : LD    C, 19          ; A = disque par
0102  CD0500          CALL  0005           ;       défaut
0105  C641            ADD   A, 41          ; Convertit en nom
0107  321E01          LD    (DISC), A      ; de disque
010A  0E09            LD    C, 9           ; Affiche message
010C  111501          LD    DE, MES        ;
010F  CD0500          CALL  0005           ;
0112  C30000          JP    0000           ; Retour au DOS
0115  4C652064  MES : DEFB  'Le disque '   ; Message
0119  69737175                             ;
011D  65                                   ;
011E  003A20    DISC: DEFB  00, ' : '      ;
011F  65737420        DEFB  'est le '      ;
0123  6C6520                               ;
0126  64697371        DEFB  'disque '      ;
012A  756520                               ;
012D  70617220        DEFB  'par défaut'   ;
0131  64826661                             ;
0135  00007574                             ;
0137  0D0A0A24        DEFB  0D, 0A, 0A, 24 ;
```

7.5.27 Fonction 1A SET DMA (DTA) ADDRESS

La fonction 1A permet de fixer l'adresse mémoire où sera situé le tampon de l'enregistrement appelé DMA BUFFER en CP/M ou DTA BUFFER en MSX-DOS. Rappelons que ce tampon est situé par défaut à l'adresse 0080H tant en MSX-DOS qu'en BASIC-DOS et que la fonction DISK RESET (0D) replace ce tampon à cette adresse. Si vous déplacez le tampon DTA à l'adresse C000H par exemple, toutes les fonctions lisant ou écrivant sur le disque se serviront de cette adresse comme tampon. Elle est particulièrement nécessaire en BASIC-DOS puisque la valeur par défaut (0080H) ne contient pas de RAM dans l'environnement BASIC-DOS.

En entrée : Le registre DE doit contenir l'adresse où l'on désire implanter le tampon DTA.
En sortie : Rien.
Préserve : Les registres C, D et E sont préservés.
Compatible : Oui.

Exemple : Voir l'exemple de la fonction 11 (SEARCH FIRST).

7.5.28 Fonction 1B GET ALLOCATION

La fonction 1B permet d'obtenir une série de paramètres concernant le disque dont le numéto est précisé dans le registre E où 0 vaut pour le disque par défaut, 1 pour le disque A :, 2 pour le disque B : etc.

En entrée : Le registre E doit contenir le numéro de disque dont on veut obtenir les paramètres.
En sortie : A donne le nombre de secteurs par cluster.
BC donne la taille du secteur en nombre d'octets.
DE donne le nombre de clusters réservés à l'utilisateur.
HL donne le nombre de clusters qui restent libres.
IX donne l'adresse du DCB de ce disque.
IY donne l'adresse du premier octet de la FAT réservée à ce disque.
Si le numéro de disque en entrée est invalide, le registre A contiendra FFH et le Carry flag sera à 1.
Préserve : Rien.
Compatible : Non. Cette fonction fournit les mêmes types de renseignements en CP/M mais par le biais d'une adresse donnée dans HL. Étant donné les différences de structure du système d'exploitation disque, il était impossible de simuler la fonction CP/M en MSX-DOS. Bien que rarement employée en CP/M, la non-compatibilité de cette fonction peut être source de gros problèmes.

Exemple : Voici un programme donnant les paramètres du disque choisi en hexadécimal.

```
                        ORG   0100

0100  0E09    START :   LD    C, 09      ; Affiche question
0102  118A01            LD    DE, QUEST  ; « Quel disque ? »
0105  CD0500            CALL  0005       ;
0108  0E01    INPUT :   LD    C, 01      ; Obtient la
010A  CD0500            CALL  0005       ; réponse
010D  5F      GETALL :  LD    E, A       ; Appel fonction
010E  0E1B    GETALL :  LD    C, 1B      ; Get Allocation
0110  CD0500            CALL  0005       ;
0113  3C      CHECK :   INC   A          ; Test si erreur
0114  200A              JR    NZ, START1 ; non → START1
0116  0E09    ERROR :   LD    C,9        ; Affiche message
0118  119B01            LD    DE,ERR     ; d'erreur
011B  CD0500            CALL  0005       ;
011E  18E0              JR    START      ;
0120  C631    START1 :  ADD   A, 31      ; Affichage des
0122  FDE5              PUSH  IY         ; différents
```

```
0124  DDE5            PUSH  IX          ; paramètres
0126  E5              PUSH  HL          ; précédés d'un
0127  D5              PUSH  DE          ; message.
0128  C5              PUSH  BC          ;
0129  32BF01          LD    (SEC), A    ;
012C  11AB01          LD    DE, NSEC    ;
012F  CD6701          CALL  PRINT       ;
0132  11C101          LD    DE, SECS    ;
0135  CD6701          CALL  PRINT       ;
0138  C1              POP   BC          ;
0139  CD6C01          CALL  HEXA        ;
013C  11D601          LD    DE, NCLUS   ;
013F  CD6701          CALL  PRINT       ;
0142  C1              POP   BC          ;
0143  CD6C01          CALL  HEXA        ;
0146  11EB01          LD    DE, NFCLU   ;
0149  CD6701          CALL  PRINT       ;
014C  C1              POP   BC          ;
014D  CD6C01          CALL  HEXA        ;
0150  110002          LD    DE, DPB     ;
0153  CD6701          CALL  PRINT       ;
0156  C1              POP   BC          ;
0157  CD6C01          CALL  HEXA        ;
015A  111502          LD    DE, FAT     ;
015D  CD6701          CALL  PRINT       ;
0160  C1              POP   BC          ;
0161  CD6C01          CALL  HEXA        ;
0164  C30000          JP    0000        ; Retour au DOS
0167  0E09    PRINT : LD    C, 9        ; Sous-routine
0169  C30500          JP    0005        ; d'impression
016C  78      HEXA :  LD    A, B        ; Sous-routine
016D  CD7101          CALL  HEX1        ; d'affichage en
0170  79              LD    A, C        ; hexadécimal
0171  F5      HEX1 :  PUSH  AF          ; du registre BC
0172  0F              RRCA              ;
0173  0F              RRCA              ;
```

```
0174    0F                  RRCA                    ;
0175    0F                  RRCA                    ;
0176    CD7A01              CALL    HEX2            ;
0179    F1                  POP     AF              ;
017A    E60F        HEX2 :  AND     0F              ;
017C    C630                AND     A, 30           ;
017E    FE3A                CP      3A              ;
0180    3802                JR      C, HEX3         ;
0182    C607                ADD     A, 7            ;
0184    5F          HEX3 :  LD      E, A            ;
0185    0E02                LD      C, 02           ;
0187    C30500              JP      0005            ;
018A    0D0A        QUEST : DEFB    0D, 0A
018C    5075656C            DEFB    'Quel disque ? $'
0190    20646973
0194    71756520
0198    3F2024
019B    0D0A        ERR :   DEFB    0D, 0A
019D    4E756D82            DEFB    'Numéro erroné$'
01A1    726F2065
01A5    72726F6E
01A9    B224
01AB    0D0A        NSEC :  DEFB    0D, 0A
01AD    53456374            DEFB    'Sect./cluster :'
01B1    2E2F636C
01B5    75737465
01B9    72202020
01BD    2A20
01BF    0024        SEC :   DEFB    0,24
01C1    0D0A5461    SECS :  DEFB    0D, 0A, 'Taille secteur : $'
01C5    696C6C65
01C9    20207365
01CD    63746575
01D1    72203A20
01D5    24
01D6    0D0A436C    NCLUS : DEFB    0D, 0A, 'Clusters/disque :
```

```
                                                $’
01DA    75737465
01DE    72732F64
01E2    69737175
01E6    65202A20
01EA    24
01EB    0D0A436C    NFCLU :     DEFB    0D, 0A, ‘Clusters libres : $’
01EF    75737465
01F3    7273206C
01F7    69627265
01FB    73202A20
01FF    24
0200    0D0A4164    DPB :       DEFB    0D, 0A, ‘Adresse DPB : $’
0204    72657373
0208    65204450
020C    42202020
0210    20202A20
0214    24
0215    0D0A4164    FAT :       DEFB    0D, 0A, ‘Adresse FAT : $’
0219    72657373
021D    65204641
0221    54202020
0225    20202A20
0229    24
```

7.5.29 Fonction 1C non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s’agit de la fonction SET WRITE PROTECT VECTOR.

7.5.30 Fonction 1D non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s’agit de la fonction GET WRITE PROTECT VECTOR.

7.5.31 Fonction 1E non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s’agit de la fonction SET FILE ATTRIBUTE.

7.5.32 Fonction 1F non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction GET DISK PARAMETER ADDRESS.

7.5.33 Fonction 20 non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction SET/GET USER CODE.

7.5.34 Fonction 21 RANDOM READ

Cette fonction permet la lecture d'un enregistrement d'un fichier à accès direct dont le contenu sera transféré vers le tampon DTA. Le fichier devra avoir été préalablement ouvert par la fonction OPEN (0F). Le fichier est référencé par son FCB dont l'adresse doit être placée dans le registre DE. L'enregistrement qui sera lu doit être déterminé par les octets RN (FCB+33 à FCB+35). Ces octets ne seront pas incrémentés par cette fonction et doivent donc toujours être placés par le programmeur avant chaque appel de cette fonction. Le record ou enregistrement a toujours une longueur fixe de 128 octets en CP/M. Cette fonction a été implémentée en MSX-DOS uniquement pour le rendre compatible avec le CP/M mais l'usage de la fonction MSX-DOS RANDOM BLOCK READ (27) est beaucoup plus rapide et avantageux.

En entrée : Le registre DE doit pointer sur le FCB du fichier.
Les octets RN du FCB déterminent le numéro de l'enregistrement qui sera lu.
En sortie : Les registres A et L retournent la valeur 00 si l'opération s'est déroulée sans erreur.
Les registres A et L retournent la valeur 01 si l'opération s'est terminée par une erreur (lecture au-delà de la fin de fichier).
Préserve : Aucun registre n'est préservé.
Compatible : Oui, mais utilisez de préférence la fonction 27 RANDOM BLOCK READ.

Exemple : Voir fonction suivante.

7.5.35 Fonction 22 RANDOM WRITE

Cette fonction permet l'écriture d'un enregistrement dans un fichier à accès direct. Le fichier est référencé par son FCB dont l'adresse doit être placée dans le registre DE. Le fichier devra avoir été ouvert par la fonction OPEN (0F) et le tampon DTA devra être rempli avec les données de l'enregistrement préalablement à l'appel de cette fonction. L'enregistrement qui sera écrit sera déterminé par les octets RN (FCB+33 à FCB+35). Ces octets ne seront pas incrémentés par cette fonction et doivent toujours être placés par le programmeur avant chaque appel de cette fonction. Le record ou enregistrement a toujours une longueur fixe de 128 octets en CP/M. Cette fonction a été implémentée en MSX-DOS uniquement pour le rendre compatible avec le CP/M mais l'usage de la fonction MSX-DOS RANDOM BLOCK WRITE (26) est beaucoup plus rapide et avantageux.

En entrée : Le registre DE doit pointer sur le FCB du fichier.
Le tampon DTA doit contenir les données de l'enregistrement.
Les octets RN du FCB déterminent le numéro de l'enregistrement qui sera écrit.
En sortie : Les registres A et L retournent la valeur 00 si l'opération s'est déroulée sans erreur.
Les registres A et L retournent la valeur 01 si l'opération s'est terminée par une erreur.

Préserve : Aucun registre n’est préservé.
Compatible : Oui, mais utilisez de préférence la fonction 26 RANDOM BLOCK WRITE.

Exemple : Ce programme extrait les enregistrements 8, 3 et 5 du fichier TEST1.TST pour les sauvegarder dans les enregistrements 9, 1 et 7 du fichier TEST2.TST.

```
                        ORG   C000

C000  0E1A     SETDMA : LD    C, 1A          ; Ajuste tampon DTA
C002  1100B0            LD    DE, 0B000      ; en B000 pour
C005  CD0500            CALL  0005           ; travail en BASIC.
C008  0E0F     OPEN1 :  LD    C, 0F          Ouvre TEST1.TST
C00A  1171C0            LD    DE, FCB1       ;
C00D  CD0500            CALL  0005           ;
C010  B7                OR    A              ; si erreur,
C011  C25EC0            JP    NZ, END        ;       → END
C014  0E0F     OPEN2 :  LD    C, 0F          ; Ouvre TEST2.TST
C016  1196C0            LD    DE, FCB2       ;
C019  CD0500            CALL  0005           ;
C01C  B7                OR    A              ; si erreur,
C01D  C25EC0            JP    NZ, END        ;       → END
C020  2163C0            LD    HL, TABLE      ; Transfère numéro
C023  4E       REC1 :   LD    C, (HL)        ; de record vers
C024  23                INC   HL             ; FCB1+21 à partir
C025  46                LD    B, (HL)        ; d’une table de
C026  23                INC   HL             ; records à copier
C027  3                 INC   BC             ;
C028  78                LD    A, B           ;
C029  A1                OR    C              ;
C02A  CA56C0            JP    Z, CLOSE       ;
C02D  08                DEC   BC             ;
C02E  ED4392C0          LD    (FCB1+21), BC  ;
C032  E5                PUSH  HL             ;
C033  0E21     READ :   LD    C, 21          ; Lit le record
C035  1171C0            LD    DE, FCB1       ;
C038  CD0500            CALL  0005           ;
C03B  B7                OR    A              ; si erreur,
```

```
C03C  C256C0          JP    NZ, CLOSE        ;       → CLOSE
C03F  E1       REC2 : POP   HL               ; Transfère record
C040  4E              LD    C, (HL)          ; de destination de
C041  23              INC   HL               : la table vers
C042  46              LD    B, (HL)          ; FCB2+21
C043  23              INC   HL               ;
C044  ED43B7C0        LD    (FCB2+21), BC    ;
C048  E5              PUSH  HL               ;
C049  0E22     WRITE: LD    C, 22            ; Écrit record lu
C04B  1196C0          LD    DE, FCB2         ; dans le fichier 2
C04E  CD0500          CALL  0005             ;
C051  E1              POP   HL               ;
C052  B7              OR    A                ; si pas d'erreur,
C053  CA23C0          JP    Z, REC1          ; → REC1
C056  0E10     CLOSE: LD    C, 10            ; Ferme fichier 2
C058  1196C0          LD    DE, FCB2         ;
C05B  CD0500          CALL  0005             ;
C05E  0E00     END  : LD    C, 0             ; Retour au BASIC
C060  C30500          JP    0005             ;
C063  0B000900 TABLE: DEFW  8,9              ; Table des records
C067  03000100        DEFW  3,1              ; à copier.
C06B  05000700        DEFW  5,7              ;
C06F  FFFF            DEFW  0FFFF            ; FFFF = fin de table
C071  00       FCB1 : DB    0                ; Numéro de disque
C072  54455344        DB    'TEST1   TST'    ; Nom de fichier
C076  31202020                               ;
C07A  545354                                 ;
C07D  00000000        DB    0, 0, 0, 0       ;
C081  00000000        DB    0, 0, 0, 0       ;
C085  00000000        DB    0, 0, 0, 0       ;
C089  00000000        DB    0, 0, 0, 0       ;
C08D  00000000        DB    0, 0, 0, 0       ;
C091  00000000        DB    0, 0, 0, 0       ;
C095  00              DB    0                ;
C096  00       FCB2 : DB    0                ; Numéro de disque
C097  54455354        DB    'TEST2   TST'    ; Nom de fichier
```

```
C09B    32202020
C09F    545354
C0A2    00000000            DB      0, 0, 0, 0          ;
C0A6    00000000            DB      0, 0, 0, 0          ;
C0AA    00000000            DB      0, 0, 0, 0          ;
C0AE    00000000            DB      0, 0, 0, 0          ;
C0B2    00000000            DB      0, 0, 0, 0          ;
C0B6    00000000            DB      0, 0, 0, 0          ;
C0B7    00                  DB      0                   ;
```

7.5.36 Fonction 23 GET FILE SIZE

La fonction 23 permet de connaître le nombre d'enregistrements de 128 octets présents dans un fichier dont le FCB est pointé par le registre DE. Le résultat est placé dans les octets RN du FCB c'est-à-dire en position FCB+33, FCB+34 et FCB+35. Cette fonction se sert de la taille du fichier en nombre d'octets pour obtenir le nombre de records. Cette fonction est particulièrement utile pour ajouter des enregistrements à la suite du dernier enregistrement d'un fichier à accès direct existant. Il suffit en effet d'ouvrir d'abord le fichier, puis d'utiliser cette fonction 23 pour installer le numéro du prochain enregistrement à écrire dans le FCB et enfin d'écrire le ou les enregistrements suivants. Attention qu'il s'agit en fait d'un nombre virtuel d'enregistrements ; il est en effet possible de créer un fichier direct avec seulement deux enregistrements portant respectivement les numéros 5 et 123 et des « trous » entre eux. Dans ce cas, la fonction retournera dans les octets RN la valeur 124 comme s'il y avait déjà 124 enregistrements réellement écrits dans le fichier alors qu'il n'y en a que deux.

En entrée : Le registre DE doit contenir l'adresse du FCB du fichier. Ce fichier peut être ouvert ou fermé. Le FCB doit être installé avant l'appel de cette fonction.

En sortie : Les positions FCB+33, FCB+34 et FCB+35 donneront le numéro du record qui suit le dernier record du fichier. Si ce numéro est inférieur à 255, il peut aussi être trouvé dans le registre C. Le registre IX donnera l'adresse du FCB. Les registres A et L seront placés à 00 si l'opération se déroule sans erreur ou à FFH si l'opération se termine par une erreur telle que fichier non trouvé.

Préserve : Aucun registre n'est préservé.

Compatible : Oui.

Exemple : Ce programme affiche le nombre de records de 128 octets du fichier placé comme premier argument de ce programme. Si vous sauvez ce programme sous le nom RECSIZ.COM, posez simplement ceci :

```
A>RECSIZ B:FICHIER.TST
```

pour connaître la taille du fichier FICHIER1.TST présent sur le disque B : en nombre de records exprimé en hexadécimal.

```
        ORG     0100
```

```
0100  215C00    START :  LD    HL, 5C      ; Copie l'argument
0103  115A01             LD    DE, FCB     ; dans le FCB
0106  011000             LD    BC, 10      ;
0109  EDB0               LDIR              ;
010B  0E23      SIZE :   LD    C, 23       ; Appel fonction 23
010D  115A01             LD    DE, FCB     ;
0110  CD0500             CALL  0005        ;
0113  B7                 OR    A           ; Si erreur,
0114  c20000             JP    NZ, 0000    ;       → DOS
0117  3A7D01    PRINT :  LD    A, (RN2)    ; A = RN2
011A  CD3401             CALL  HEXA        ; affiche A en hexa
011D  3A7C01             LD    A, (RN1)    ; A = RN1
0120  CD3401             CALL  HEXA        ; affiche A en hexa
0123  3A7B01             LD    A, (RN0)    ; A = RN0
0126  CD3401             CALL  HEXA        ; affiche A en hexa
0129  0E09      END :    LD    C, 09       ; Affiche message
012B  114D01             LD    DE, CRLF    ;         et
012E  CD0500             CALL  0005        ; retour au DOS
0131  C30000             JP    0000        ;
0134  F5        HEXA :   PUSH  AF          ; Affichage de A en
0135  0F                RRCA              ;     hexadécimal
0136  0F                RRCA              ;
0137  0F                RRCA              ;
0138  0F                RRCA              ;
0139  CD3D01             CALL  HEX1        ;
013C  F1                POP   AF          ;
013D  E60F      HEX1 :   AND   0F          ;
013F  C630               ADD   A, 30       ;
0141  FE3A               CP    3A          ;
0143  3802               JR    C, HEX2     ;
0145  C607               ADD   A, 7        ;
0147  5F        HEX2 :   LD    E, A        ;
0148  0E02               LD    C, 02       ;
014A  C30500             JP    0005        ;
014D  48207265  CRLF :   DEFB  'H records' ; Message final
```

```
0151    636F7264                                    ;
0155    73                                          ;
0156    0D0A0A24            DEFB    0D, 0A, 0A, 24  ;
015A    00          FCB :   DEFB    0               ; FCB dont les 16
015B    00000000            DEFB    0, 0, 0, 0      ; premiers octets
015F    00000000            DEFB    0, 0, 0, 0      : sont copiés du
0163    00000000            DEFB    0, 0, 0, 0      ; premier argument
0167    00000000            DEFB    0, 0, 0, 0      ;
016B    00000000            DEFB    0, 0, 0, 0      ;
016F    00000000            DEFB    0, 0, 0, 0      ;
0173    00000000            DEFB    0, 0, 0, 0      ;
0177    00000000            DEFB    0, 0, 0, 0      ;
017B    00          RN0 :   DEFB    0               ; Octets RN du FCB
017C    00          RN1 :   DEFB    0               ;
017D    00          RN2 :   DEFB    0               ;
017E    00          RN3 :   DEFB    0               ;
```

7.5.37 Fonction 24 SET RANDOM RECORD

La fonction 24 est une fonction du CP/M qui est officiellement supportée par le MSX-DOS. Cependant, dans les différentes versions de contrôleur-disque que j'ai pu examiner une erreur (ou bug) s'est glissée dans la portion de code destinée à cette fonction par Microsoft. Il en résulte qu'elle ne peut pas fonctionner. Cependant, nous allons quand même expliquer son fonctionnement pour ceux qui auraient une version de ROM correcte et nous fournirons une petite routine qui réalise la même fonction pour les autres.

Le but de cette fonction est de traduire le numéro de record séquentiel en un numéro de record à accès direct. En effet, lorsqu'on manipule un fichier séquentiel, le programme ne garde pas toujours la trace du numéro de record où l'on est arrivé. Ainsi, cette fonction traduit les octets S2-EX-CR (FCB+14, FCB+12, FCB+32) déterminant le numéro de record en mode séquentiel en leur équivalent dans les octets RN (FCB+35, FCB+ 34 et FCB+33) utilisés en mode direct.

En entrée : Le registre DE doit pointer sur le FCB du fichier ouvert qui a déjà été manipulé par des fonctions SEQUENTIAL READ (14) et SEQUENTIAL WRITE (15).
En sortie : Sans bug dans la ROM, les octets RN (FCB+33, FCB+34 et FCB+35) devraient contenir le numéro de record. Le regustre IY contient l'adresse du FCB.
Préserve : Rien.
Compatible : Oui, elle devrait être compatible sans le bug de la ROM. Actuellement les octets FCB+33 et FCB+34 retournent respectivement 01-00 et l'octet FCB+35 retourne la valeur correcte.

Exemple : Nous fournissons ici une routine qui peut être appelée à la place de cette fonction et fournit le résultat correct. Cette routine est entièrement relogeable. Cela signifie que vous pouvez l'implanter n'importe où en mémoire.

```
                        ORG   XXXX

XXXX  D5        RNDREC :  PUSH  DE          ; Sauve le FCB dans
      FDE1                POP   IY          ; le registre IY
      FD4E20              LD    C, (IY+20)  ; C = FCB+32 CR
      FD460C              LD    B, (IY+0C)  ; B = FCB+12 EX
      FD5E0E              LD    E, (IY+0E)  ; E = FCB+14 S2
      1600                LD    D, 0        ; D = 0
      CB21                SLA   C           ; C = C * 2
      CB3B                SRL   E           ; }
      CB18                RR    B           ; } EBC \ 2
      CB19                RR    C           ; }
      FD7521              LD    (IY+21), C  ; FCB + 33 = C
      FD7422              LD    (IY+22), B  ; FCB + 34 = B
      FD7323              LD    (IY+23), E  ; FCB + 35 = E
      C9                  RET               ;
```

7.5.38 Fonction 25 non utilisée en MSX-DOS

Retourne 00 dans les registres A et L et préserve les registres C, D et E. En CP/M, il s'agit de la fonction RESET DISK DRIVE.

7.5.39 Fonction 26 RANDOM BLOCK WRITE (MSX-DOS)

Voici une fonction exclusivement MSX-DOS qui remplace à elle seule les fonctions CP/M SEQUENTIAL WRITE (15) et RANDOM WRITE (22). Elle est beaucoup plus rapide et autorise de fixer la longueur et le nombre des enregistrements qui vont être écrits à votre guise.

Cette fonction écrit dans le fichier spécifié par le FCB pointé par le registre DE un nombre de records spécifié par le registre HL dont les données se trouvent dans le tampon DTA.

Le numéro du premier record à écrire doit être placé dans les octets RN du FCB (FCB+33, FCB+34, FCB+35 et FCB+36). Ces octets RN seront automatiquement incrémentés du nombre de records écrits si l'opération se déroule sans erreur. Cela permet de quitter l'opération avec le FCB prêt pour une écriture suivante (mode séquentiel). Si vous désirez travailler en mode direct, il faudra placer le numéro du record à écrire dans les octets RN avant chaque appel de cette fonction.

La taille des records à écrire n'est plus figée à 128 octets comme en CP/M mais est programmable. Il suffit pour cela de placer la taille désirée (de 1 à 65535 octets) dans les positions RECSIZ du FCB (FCB+14 et FCB+15).

En entrée : Le registre DE doit indiquer l'adresse du FCB.
Le registre HL doit indiquer le nombre de records à écrire.
Les octets RECSIZ du FCB doivent indiquer la taille du record.
Les octets RN du FCB doivent indiquer le numéro du record de départ.
Le tampon DTA doit contenir les données du/des records à écrire.

En sortie : Le registre A indiquera 00 si l'opération se déroule sans erreur ou 01 dans le cas contraire.
Les octets RN du FCB seront incrémentés du nombre de records écrits.
Le registre IX donnera l'adresse du DCB et le registre IY l'adresse du FCB.

Préserve : Aucun registre n'est préservé.

Compatible : Non. Cette fonction n'existe pas en CP/M.

Exemple : Ce programme permet de sauver sur un fichier le contenu complet de la mémoire réservée au DOS.

```
                        ORG   0100

0100  0E1A   SETDTA :  LD    C, 1A       ; Tampon DTA = 0000
0102  110000           LD    DE, 0000    ;
0105  CD0500           CALL  0005        ;
0108  0E16   CREATE :  LD    C, 16       ; Création fichier
010A  113701           LD    DE, FCB     ; MEMORY.DMP
010D  CD0500           CALL  0005        ;
0110  B7               OR    A           ; Si erreur,
0111  C20000           JP    NZ, 0000    ;         → DOS
0114  210000   WRITE :  LD    HL, 0       ; Numéro record = 0
0117  225B01           LD    (FCB+21), HL ;
011A  225A01           LD    (FCB+23), HL ;
011D  23               INC   HL          ; Longueur record = 1
011E  224501           LD    (FCB+0E), HL ;
0121  2A0600           LD    HL, (0006)  ; Nombre records = TOP
0124  113701           LD    DE, FCB     ;
0127  0E26             LD    C, 26       ; Écriture TOP records
0129  CD0500           CALL  0005        ;
012C  0E10   CLOSE :   LD    C, 10       ; Fermeture fichier
012E  113701           LD    DE, FCB     ;
0131  CD0500           CALL  0005        ;
```

```
0134  C30000          JP    0000          ; Retour au DOS
0137  00       FCB :  DB    'MEMORY  '    ; FCB
0138  4D454D4F        DB                  ;
013C  5259
013E  444D50          DB    'DMP'         ;
0151  00000000        DB    0, 0, 0, 0    ;
0155  00000000        DB    0, 0, 0, 0    ;
0159  00000000        DB    0, 0, 0, 0    ;
015D  00000000        DB    0, 0, 0, 0    ;
0161  00000000        DB    0, 0, 0, 0    ;
0165  00000000        DB    0, 0, 0, 0    ;
0169  00              DB    0             ;
```

7.5.40 Fonction 27 RANDOM BLOCK READ (MSX-DOS)

Voici encore une fonction exclusivement MSX-DOS qui remplace à elle seule les fonctions CP/M SEQUENTIAL READ (14) et RANDOM READ (21). Elle est beaucoup plus rapide et autorise de fixer la longueur et le nombre des enregistrements qui vont être lus à votre guise.

Cette fonction lit dans le fichier spécifié par le FCB pointé par le registre DE un nombre de records spécifié par le registre HL et transfère les donnés dans le tampon DTA.

Le numéro du premier record à lire doit être placé dans les octets RN du FCB (FCB+33, FCB+34, FCB+35 et FCB+36). Ces octets RN seront automatiquement incrémentés du nombre de records lus si l'opération se déroule sans erreur. Cela permet de quitter l'opération avec le FCB prêt pour une lecture suivante (mode séquentiel). Si vous désirez travailler en mode direct, il faudra placer le numéro du record à écrire dans les octets RN avant chaque appel de cette fonction.

La taille des records à lire n'est plus figée à 128 octets comme en CP/M mais est programmable. Il suffit pour cela de placer la taille désirée (de 1 à 65535 octets) dans les positions RECSIZ du FCB (FCB+14 et FCB+15).

En entrée : Le registre DE doit indiquer l'adresse du FCB.
Le registre HL doit indiquer le nombre de records à lire.
Les octets RECSIZ du FCB doivent indiquer la taille du record.
Les octets RN du FCB doivent indiquer le numéro du record de départ.

En sortie : Le registre A indiquera 00 si l'opération se déroule sans erreur ou 01 dans le cas contraire.
Les octets RN du FCB seront incrémentés du nombre de records lus.
Les registres HL et BC indiqueront le nombre de records effectivement lus.
Le registre IX donnera l'adresse du DCB et le registre IY l'adresse du FCB.

Préserve : Aucun registre n'est préservé.

Compatible : Non. Cette fonction n'existe pas en CP/M.

Exemple : Vous savez que tout programme MSX-DOS se charge en posant simplement le nom du programme après l'indicatif du DOS A>. Par contre, ces programmes s'implantent toujours à l'adresse 0100H au début de la TPA. Cet exemple va vous montrer comment charger un programme à une adresse différente. Nous appellerons ce programme LOAD.COM et pour l'utiliser nous poserons après l'indicatif du DOS ceci :

A> LOAD NOM-DU-PROGRAMME 23FF

Le premier argument sera le nom du programme ou du fichier à charger et le second argument l'adresse mémoire où vous désirez l'implanter. La taille du fichier sera extraite de la FCB pour connaître la taille du record à charger.

```
                        ORG   0100

0100  215C00  START :   LD    HL, 005C    ; Copie du premier
0103  115A01            LD    DE, FCB     ; argument dans le
0106  011000            LD    BC, 10      ; FCB
0109  EDB0              LDIR              ;
010B  215901  LDADD :   LD    HL, RESULT+1; Convertir second
010E  116D00            LD    DE, 006D    ; argument en
0111  CD4801            CALL  BIN         ; binaire dans
0114  2B                DEC   HL          ; RESULT.
0115  CD4801            CALL  BIN         ;
0118  ED5B5801 SETDTA : LD    DE, (RESULT); Place le tampon
011C  0E1A              LD    C, 1A       : DTA à l'adresse
011E  CD0500            CALL  0005        ; RESULT.
0121  115A01    OPEN :  LD    DE, FCB     ; Ouvre le fichier
0124  0E0F              LD    C, 0F       ;
0126  CD0500            CALL  0005        ;
0129  B7                OR    A           ; Si erreur,
012A  C20000            JP    NZ, 0000    ;       → DOS
012D  2A6A01    LOAD :  LD    HL, (FCB+10); Taille fichier ->
0130  226801            LD    (FCB+0E), HL;         taille record.
0133  210000            LD    HL, 0       ; Mettre 0 dans
0136  227B01            LD    (FCB+21), HL; numéro de record
0139  227D01            LD    (FCB+23), HL; (octets RN).
013C  23                INC   HL          ; HL = nombre records
013D  115A01            LD    DE, FCB     ;       à lire (1)
```

```
0140  0E27            LD    C, 27       ; Appel fonction
0142  CD0500          CALL  0005        ;
0145  C30000          JP    0000        ; Retour au DOS.
0148  CD4B01  BIN :   CALL  BIN1        ; Routine qui
014B  1A      BIN1 :  LD    A, (DE)     ; convertit (DE) et
014C  D630            SUB   30          ; (DE+1) en binaire
014E  FE3A            CP    3A          ; dans (HL)
0150  3802            JR    C, BIN2     ;
0152  D607            SUB   7           ;
0154  13      BIN2 :  INC   DE          ;
0155  ED6F            RLD               ;
0157  C9              RET               ;
0158  0000    RESULT: DEFW  0           ; Adresse chargement
015A  00      FCB :   DEFB  0           ; FCB dont les 16
015B  00000000        DEFB  0, 0, 0, 0  ; premiers octets
015F  00000000        DEFB  0, 0, 0, 0  ; sont copiés du
0163  00000000        DEFB  0, 0, 0, 0  ; premier argument
0167  00000000        DEFB  0, 0, 0, 0  ;
016B  00000000        DEFB  0, 0, 0, 0  ; Les autres octets
016F  00000000        DEFB  0, 0, 0, 0  ; sont fournis pas
0173  00000000        DEFB  0, 0, 0, 0  ; la routine LOAD.
0177  00000000        DEFB  0, 0, 0, 0  ;
017B  00000000        DEFB  0, 0, 0, 0  ;
```

7.5.41 Fonction 28 RANDOM WRITE WITH ZERO FILL

Cette fonction est similaire à la fonction RANDOM WRITE (22) à l'exception que, lors d'une extension du fichier, tous les records non encore écrits sont remplis de codes 00.

L'emploi de cette fonction à la place de la fonction 22 fait que le fichier ne contiendra aucun « trou » c'est-à-dire des recors non écrits qui pourraient tromper un programme de lecture si ces records non écrits étaient lus à tort.

Pour que ce fichier soit réellement sans trous, il faut absolument utiliser la fonction 28 pour chaque écriture dans ce fichier.

Exemple : Si vous créer un fichier et que vous le remplissez avec cette fonction en écrivant les records dans l'ordre suivant 1, 2, 5, 9, 21, alors les records de numéro 0, 3, 4, 6, 7, 8 et 10 à 20 contiendront 128 octets 00 chacun.

Mode d'emploi : Voir fonction 22.

Compatible : Oui.

7.5.42 Fonction 29 NO FUNCTION (MSX-DOS)

Cette fonction est réservée pour de futures extensions du MSX-DOS. Elle retourne la valeur 0 dans le registre A et préserve les registres C, D et E.

7.5.43 Fonction 2A GET DATE AND TIME (MSX-DOS)

Cette fonction retourne la date établie à l'allumage de votre système pour les MSX sans horloge électronique (MSX1). Pour les MSX2, elle retourne la date et le temps (heure et minute seulement) mémorisés dans le dateur électronique protégé par batterie (horodateur permanent).

En entrée : Rien.
En sortie : HL retourne l'année sous la forme 19xx en binaire. Exemple : 1987 = 07C3, H = 07, L = C3.
D retourne le mois en binaire (1-12).
B retourne l'heure en binaire (0-23 MSX2).
C retourne les minutes en binaire (0-59 MSX2).
E retourne le jour du mois en binaire (1-31).
A retourne le jour de la semaine (0 pour dimanche, 1 pour lundi … 6 pour samedi).
Préserve : Aucun registre n'est préservé.
Compatible : Non.

Exemple : Ce programme va vous donner le jour de la semaine et la date dans l'ordre correct, que votre clavier soit AZERTY ou QWERTY.

```
                        ORG   0100

0100  0E2A    START :   LD    C, 2A        ; Appel fonction
0102  CD0500            CALL  0005         ;
0105  E5                PUSH  HL           ; Sauve l'année
0106  D5                PUSH  DE           ; Sauve mois et jour
0107  217F01  NOMJR :   LD    HL, JOUR     ; HL = table des jours
010A  87                ADD   A, A         ; Multiplie code du
010B  87                ADD   A, A         ; jour de la semaine
010C  87                ADD   A, A         ; par 8.
010D  85                ADD   A, L         ; Additionne résultat
010E  6F                LD    L, A         ; à l'adresse de la
010F  3001              JR    NC, NOMJ1    ; table.
0111  24                INC   H            ;
0112  0608    NOMJ1 :   LD    B, 8         ; Boucle 8 fois pour
```

```
0114  C5        NOMJ2 :  PUSH  BC        ; afficher le nom du
0115  5E                 LD    E, (HL)   ; jour de la semaine
0116  23                 INC   HL        ;
0117  E5                 PUSH  HL        ;
0118  0E02               LD    C, 02     ;
011A  CD0500             CALL  0005      ;
011D  E1                 POP   HL        ;
011E  C1                 POP   BC        ;
011F  10F3               DJNZ  NOMJ2     ;
0121  0E09       NOMJ3 :  LD    C, 09     ; Affiche virgule et
0123  117801             LD    DE, MES1  ; un espace.
0126  CD0500             CALL  0005      ;
0129  D1         DATJ :   POP   DE        ; Rappel Mois / Jour
012A  D5                 PUSH  DE        ; Sauve Mois
012B  7B                 LD    A, E      ; Jour → A
012C  CD5401             CALL  PRDU      ; Affiche jour -
012F  F1         DATM :   POP   AF        ; Rappel Mois → A
0130  CD5401             CALL  PRDU      ; Affiche Mois -
0133  3E13       DATA:    LD    A, 13     ; Affiche 19
0135  CD5F01            CALL  BINDEC    ;
0138  CD6D01            CALL  PRINT     ;
013B  E1                 POP   HL        ; Rappel Année
013C  016C07             LD    BC, 076C  ; BC = 1900
013F  B7                 OR    A         ; clear Carry
0140  ED42               SBC   HL, BC    ; L = année - 1900
0142  7D                 LD    A, L      ; Affiche Année
0143  CD5F01            CALL  BINDEC    ;
0146  CD6D01            CALL  PRINT     ;
0149  0E09               LD    C, 09     ; Affiche CR-LF
014B  117B01             LD    DE, CRLF  ;
014E  CD0500             CALL  0005      ;
0151  C30000             JP    0000      ; Retour au DOS
0154  CD5F01     PRDU :  CALL  BINDEC    ; Affiche dizaines
0157  CD6D01             CALL  PRINT     ; et unités.
015A  1E2D               LD    E, ' -'   ; Affiche « - »
015C  C30500             JP    0005      ; et retour.
```

```
015F  1E30      BINDEC :  LD    E, 30          ; Convertit binaire
0161  D60A      BINDE1 :  SUB   0A             ; en 2 chiffres
0163  00003803           JR    C, BINDE2      ; décimaux.
0165  1C                 INC   E              ;
0166  18F9               JR    BINDE1         ; E = (A\B) OR 30
0168  C60A      BINDE2 :  ADD   A, 0A          ;
016A  C630               ADD   A, 30          ; A = (A MOD B) OR 30
016C  C9                 RET                  ;
016D  F5        PRINT :   PUSH  AF             ; Routine d'affichage.
016E  0E02               LD    C, 02          ;
0170  CD0500             CALL  0005           ; Affiche E.
0173  F1                 POP   AF             ;
0174  5F                 LD    E, A           ; Affiche A.
0175  C30500             JP    0005           ;
0178  2C2024    MES1 :    DEFB  ', $'          ;
017B  0D0A0A24  CRLF :    DEFB  0D, 0A, 0A     ;
017F  44696D61  JOUR :    DEFB  'Dimanche'
0183  6E636865
0187  4C756E64            DEFB  'Lundi', 0, 0, 0
018B  69000000
018F  4D617264            DEFB  'Mardi', 0, 0, 0
0193  69000000
0197  4D657263            DEFB  'Mercredi'
019B  72656469
019F  4A657564            DEFB  'Jeudi', 0, 0, 0
01A3  69000000
01A7  56656E64            DEFB  'Vendredi'
01AB  72656469
01AF  53616D65            DEFB  'Samedi', 0, 0
01B3  64690000
```

7.5.44 Fonction 2B SET DATE (MSX-DOS)

Cette fonction installe la date posée dans les registres définis ci-dessous dans les positions mémoires réservées à cet effet à savoir le jour en F248, le mois en F249, l'année en F24A. La routine calcule elle-même le numéro du jour de la semaine et l'installe en F24E et le nombre de jours écoulés depuis le 01/01/1980 et l'installe en F24C et F24D. Elle modifie également la table

du nombre de jours par mois suivant que l'année es bissextile ou non (F22C = février). Si vous disposez d'un horodateur électronique (MSX2), la date est aussi sauvée dans la mémoire de ce composant qui est protégé par batterie. Bien entendu, cette fonction va vérifier la validité de la date que vous posez dans les registres et signalera toute erreur en plaçant FF dans le registre A.

En entrée : HL doit contenir l'année. Les années entre 1980 et 2099 sont seules considérées comme valides et ne peuvent être abrégées en années du siècle.
D doit contenir le mois.
E doit contenir le jour.
En sortie : A indiquera 00 si la date est valide.
A indiquera FF si la date n'est pas valide.
Les positions mémoire de la date seront mises à jour de même que les tables indiquées ci-dessus.
Préserve : Aucun registre n'est préservé.
Compatible : Non.

Exemple : Voici comment installer la date du 30/09/97

```
                        ORG   0100

0100  21C307  START :   LD    HL, 07C3    ; 07C3H = 1987
0103  1609             LD    D, 09       ; 9 = MOIS
0105  1E1E             LD    E, 1E       ; 1EH = 30
0107  0E2B             LD    C, 2B       ;
0109  CD0500           CALL  0005        ; Appel fonction
010C  C30000           JP    0000        ; Retour au DOS
```

7.5.45 Fonction 2C GET TIME (MSX-DOS)

Cette fonction retourne l'heure qu'il est en heures, minutes et secondes dans les registres définis ci-dessous. Bien entendu, il est nécessaire d'avoir un horodateur électronique sinon cette fonction retourne uniquement des zéros. Cette fonction n'est intéressante que si vous avez besoin des secondes, sinon la fonction 2A GET DATE est plus pratique puisqu'elle fournit la date et l'heure.

En entrée : Rien.
En sortie : H retourne les heures de 0 à 23.
L retourne les minutes de 0 à 59.
D retourne les secondes de 0 à 59.
E est prévu pour retourner les centièmes de secondes lorsqu'une horloge aussi précise aura vu le jour en MSX.
Préserve : Aucun registre n'est préservé.
Compatible : Non.

Exemple : Ce petit programme affiche l'heure en permanence dans le coin de l'écran.

```
                        ORG   0100

0100  114901  START :   LD    DE, CLS     ; CLS + curseur off
0103  0E09              LD    C, 09       ;
0105  CD0500  START1 :  CALL  0005        ; HOME
0108  0E2C    TIME :    LD    C, 2C       ; Appel fonction
010A  CD0500            CALL  0005        ;
010D  D5                PUSH  DE          ; Sauve secondes
010E  E5                PUSH  HL          ; Sauve minutes
010F  7C      HEURE :   LD    A, H        ; A = heures
0110  CD3101            CALL  BINDEC      ; Binaire → Décimal
0113  3E48              LD    A, 'H'      ; Affiche « H »
0115  CD4301            CALL  PRINT       ;
0118  E1      MINUT :   POP   HL          ; Rappel minutes
0119  7D                LD    A, L        ;
011A  CD3101            CALL  BINDEC      ; Binaire → Décimal
011D  3E27              LD    A, " ' "    ; Affiche '
011F  CD4301            CALL  PRINT       ;
0122  D1      SECON :   POP   DE          ; Rappel secondes
0123  7A                LD    A, D        ;
0124  CD3101            CALL  BINDEC      ; Binaire → Décimal
0127  3E22              LD    A, ' " '    ; Affiche "
0129  CD4301            CALL  PRINT       ;
012C  1E0B              LD    E, 0B       ; Curseur Home
012E  C30501            JP    START1      ; et recommence
0131  1E30    BINDEC :  LD    E, 30       ; Convertit A binaire
0133  D60A    BINDE1 :  SUB   0A          ; en E = dizaines
0135  3803              JR    C, BINDE2   ;      A = unités
0137  1C                INC   E           ;
0138  18F9              JR    BINDE1      ;
013A  C63A    BINDE2 :  ADD   A, 3A       ;
013C  F5                PUSH  AF          ;
013D  0E02              LD    C, 02       ; Affiche dizaines
013F  CD0500            CALL  0005        ;
0142  F1                POP   AF          ; Affiche unités
0143  5F      PRINT :   LD    E, A        ; Affiche A
```

```
0144  0E02            LD    C, 02          ;
0146  C30500          JP    0005           ;
0149  0C1B7835  CLS : DEFB  0C, 1B, 'x5$'  ; CLS + ESC-x-5
014D  24
```

7.5.46 Fonction 2D SET TIME (MSX-DOS)

Cette fonction permet d'ajuster l'horloge interne de votre MSX2 en plaçant l'heure donnée dans les registres décrits ci-dessous dans l'horloge électronique. Bien entendu un contrôle de validité des heures, minutes et secondes est effectué.

En entrée : H dit contenir les heures de 0 à 23.
L doit contenir les minutes de 0 à 59.
D doit contenir les secondes de 0 à 59.
E pourra contenir les centièmes de secondes lorsqu'une horloge électronique de cette précision aura vu le jour en MSX.
En sortie : A contiendra 00 si l'heure est valable.
A contiendra FF si l'heure n'est pas valable.
Préserve : Aucun registre n'est préservé.
Compatible : Non.

Exemple : Voici comment placer 12H00'00" dans l'horloge électronique.

```
                        ORG   0100

0100  260C      START : LD    H, 0C          ; H = 12 (heure)
0102  2E00            LD    L, 0           ; L = 0 (minute)
0104  1600            LD    D, 0           ; D = 0 (seconde)
0106  0E2D            LD    C, 2D          ; Appel fonction
0108  CD0500          CALL  0005           ;
010B  C30000          JP    0000           ; Retour au DOS
```

7.5.47 Fonction 2E SET/RESET VERIFY FLAG (MSX-DOS)

Cette fonction permet d'établir ou de remettre à zéro le sémaphore de vérification (Verify Flag). Lorsqu'il est établi, ce sémaphore provoque que toutes les écritures sur disque seront relues de façon automatique par le MSX-DOS pour s'assurer que les données ont été écrites correctement sur le disque. Cette méthode de travail est beaucoup plus sûre puisque vous avez alors la garantie que les données écrites sur le disque sont re lisibles mais a comme inconvénient de ralentir un peu l'opération d'écriture. La mise à zéro de ce sémaphore supprime cette relecture.

En entrée : E doit contenir 00 pour mettre le sémaphore à zéro.
E doit contenir autre chose que 00 pour établir le sémaphore.

En sortie : Le code placé dans le registre E sera transféré vers la position RAM F30D. Cette position peut être consultée pour connaître l'état du sémaphore.
Préserve : Les registres C, D et E sont préservés.
Compatible : Non.

7.5.48 Fonction 2F ABSOLUTE DISK READ (MSX-DOS)

Cette fonction permet de lire le disque par secteurs plutôt que par enregistrements. Il s'agit donc d'une lecture en mode « physique » et non pas « logique ». La fonction en elle-même lit sur le disque donné par le registre L un nombre H de secteurs à partir du secteur numéro DE vers le tampon DTA.

En entrée : H doit contenir le nombre de secteurs que vous désirez lire (de 1 à 255).
L doit contenir le numéro de disque sur lequel se trouvent les secteurs à lire (0=A :, 1=B :, 2=C :… 7 =H:).
DE doit contenir le numéro du premier secteur à lire. La numérotation des secteurs sur le disque commence à 0. le numéro maximum dépend du type de disque et peut être trouvé dans l'annexe A.
En sortie : C indiquera le nombre de secteurs lus.
DE indiquera le premier secteur lu.
IX indiquera l'adresse du DCB.
En cas d'erreur, vous recevrez le message d'erreur habituel du MSX-DOS ou du BASIC suivant l'environnement dans lequel vous travaillez, par exemple :
En BASIC-DOS : Disk Offline
En MSX-DOS : Drive not ready error…
Aabort, Retry, Ignore
Il n'y a pas d'indication dans un registre ou un flag de cette erreur.
Préserve : Seul le registre DE est préservé s'il n'y a pas d'erreur.
Compatible : Non.

Exemple : Voici comment lire les 7 secteurs du Directory d'un disque de 360K en 0200H de la mémoire. Voyez l'annexe A pour changer les paramètres pour un autre type de disque.

```
                        ORG   0100

0100  0E1A    START :   LD    C, 1A      ; Place le tampon
0102  110002            LD    DE, 0200   ; DTA en 0200H
0105  CD0500            CALL  0005       ;
0108  2E00    READ :    LD    L, 0       ; Disque A :
010A  2607              LD    H, 7       ; 7 secteurs à lire
010C  110500            LD    DE, 5      ; premier secteur = 5
010F  0E2F              LD    C, 2F      ; Appel fonction
0111  CD0500            CALL  0005       ;
0114  C30000            JP    0000       ; Retour au DOS
```

Ce programme ne fait que lire les 7 secteurs du Directory et les place de 0200H à 11FFH de la mémoire. À vous de placer une routine de visualisation en hexadécimal par exemple.

7.5.49 Fonction 30 ABSOLUTE DISK WRITE (MSX-DOS)

Cette fonction permet d'écrire sur le disque en mode physique c'est-à-dire en donnant le numéro de secteur. La fonction écrit H secteurs sur le disque logique L à partir du secteur DE. Le contenu des secteurs à écrire doit avoir été préalablement installé dans le tampon DTA.

En entrée : H doit contenir le nombre de secteurs que vous désirez écrire (de 1 à 255).
L doit contenir le numéro de disque sur lequel se trouvent les secteurs à écrire
(0=A :, 1=B :… 7=H:).
DE doit contenir le numéro du premier secteur à écrire. La numérotation des secteurs sur le disque commence à 0. Le numéro maximum dépend du type de disque et peut être trouvé dans l'annexe A.

En sortie : C indiquera le nombre de secteurs écrits.
DE indiquera le premier secteur écrit.
IX indiquera l'adresse du DCB.
En cas d'erreur, vous recevrez le message d'erreur habituel du MSX-DOS ou du BASIC suivant l'environnement dans lequel vous travaillez, par exemple :
En BASIC-DOS : Disk Offline
En MSX-DOS : Drive not ready error…
Aabort, Retry, Ignore
Il n'y a pas d'indication dans un registre ou un flag de cette erreur.

Préserve : Seul le registre DE est préservé s'il n'y a pas d'erreur.

Compatible : Non.

Exemple : Attention à cette fonction. Comme elle écrit sur le disque physique, il est très facile d'écraser le contenu de celui-ci. Pour expérimenter cette fonction utilisez un disque ne contenant aucun fichier utile ! Soyez particulièrement attentif à ne pas écraser le secteur 0, les tables FAT et le Directory (voyez l'annexe A pour connaître le numéro de ces secteurs).
Ce petit programme va lire le secteur 0 du disque A :, vous demander votre nom et réécrire ce secteur en plaçant votre nom en position 3 à 8.

```
                        ORG   0100

0100  0E1A    START :   LD    C, 1A       ; Place DTA en 200H
0102  210002            LD    DE, 200     ;
0105  CD0500            CALL  0005        ;
0108  2E00    READ :    LD    L, 0        ; lecture disque A :
010A  2601              LD    H, 1        ; 1 secteur
010C  110000            LD    DE, 0       ; Numéro 0
010F  0E2F              LD    C, 2F       ;
```

```
0111  CD0500          CALL  0005          ;
0114  0E09    NOM :   LD    C, 09         ; Affiche message
0117  114201          LD    DE, QUEST     ; demandant votre
011A  CD0500          CALL  0005          ; nom
011D  0E0A    INPUT : LD    C, 0A         ; lecture de votre
011F  115001          LD    DE, BUF       ; réponse en 8 caractères
0122  3E08            LD    A, 8          ; maximum dans BUF
0124  0012            LD    (DE), A       ;
0125  CD0500          CALL  0005          ;
0128  215201  COPY :  LD    HL, BUF+2     ; Copie de votre réponse
012B  110302          LD    DE, 203       ; dans tampon DTA+3
012E  010800          LD    BC, 08        ;
0131  EDB0            LDIR                ;
0133  2E00    WRITE : LD    L, 0          ; Écrit sur le disque A :
0135  2601            LD    H, 1          ; 1 secteur
0137  110000          LD    DE, 0         ; numéro 0
013A  0E30            LD    C, 30         ;
013C  CD0500          CALL  0005          ;
013F  C30000          JP    0000          ; Retour au DOS
0142  0C      QUEST : DEFB  0C            ; CLS
0143  564F5452        DEFB  'VOTRE NOM ? $'
0147  45204E4F
014B  4D203F20
014F  24
0150  0000    BUF :   DEFB  0             ;
0152                  DEFS  8             ;
```

7.6 Direct BIOS access

Le CP/M permet des accès directs à certaines routines du BIOS en ajoutant un décalage à l'adresse d'implantation du CP/M en mémoire qui est indiquée en position 1 et 2 de la mémoire.

Le MSX-DOS a réalisé aussi ce genre d'accès à certaines routines mais du aux différences de manipulation des fichiers, une partie limitée seulement de ces routines peut être accédée en MSX-DOS.

Voici le tableau comparatif entre les accès du CP/M et du MSX-DOS.

OFFSET	MSX-DOS	CP/M	FONCTION

00H	oui	oui	Retour à l'indicatif du DOS
03H	oui	oui	Retour à l'indicatif du DOS
06H	oui	oui	Fonction console status (0B)
09H	oui	oui	Fonction console input (01)
0CH	oui	oui	Fonction console output (02)
0FH	non	oui	Fonction list output (05)
12H	non	oui	Fonction Aux. Output (04)
15H	non	oui	Fonction Aux. Input (03)
18H	non	oui	Move to track 00
1BH	non	oui	Select disk drive
1EH	non	oui	Set track number
21H	non	oui	Set sector number
24H	non	oui	Set DMA address
27H	non	oui	Read selected sector
2AH	non	oui	Write selected sector
2DH	non	oui	Return list status
30H	non	oui	Sector translate subroutine

Exemple : Pour retourner à l'indicatif du DOS, il y a maintenant 3 possibilités :

1) JP 0000 C'est l'utilisation du point de sortie de la page 0.

2) LD C, 00 C'est l'utilisation de la fonction 0.
CALL 0005

3) LD HL, (1) C'est l'utilisation du Direct BIOS call
JP (HL)

Exemple : Pour lire un code du clavier, il y a deux possibilités :

1) LD C, 02 C'est l'utilisation de la fonction 2.
CALL 0005

2) Utiliser le Direct BIOS call. Il faut, pour cela, mettre dans BC la valeur de l'offset du tableau ci-dessus, appeler une routine qui va charger dans HL l'adresse d'implantation du DOS (données par les positions 1 et 2) et ajouter BC à HL pour ensuite sauter à (HL). Dans l'exemple ci-dessous, chaque caractère tapé au clavier sera affiché deux fois à l'écran.

```
                        ORG   0100

0100   010300   RKBD :   LD    BC, 09          ; BC = Offset table
```

```
0103    CD0E01          CALL    DBIOS           ; pour Console Input
0106    010C00  WSCR :  LD      BC, 0C          ; BC = Offset table
0109    CD0E01          CALL    DBIOS           ; pour Console Output
010C    18F2            JR      START           ; Recommence.
010E    2A0100  DBIOS : LD      HL, (1)         ; HL = adresse du DOS
0111    09              ADD     HL, BC          ; Ajoute Offset à HL
0112    E9              JP      (HL)            ; Saute à (HL)
```

Vous aurez certainement remarqué qu'il est beaucoup plus simple d'employer les fonctions standard décrites plus avant dans ce chapitre.

8) Chapitre 8 : L'éditeur MSX-DOS et les fichiers BATCH

8.1 L'éditeur du MSX-DOS

8.1.1 Généralités

Si vous avez déjà travaillé en MSX-BASIC, vous aurez certainement apprécié le confort de son éditeur plein-écran. En effet, la correction d'une faute de frappe est aussi simple que possible, il suffit de déplacer le curseur jusqu'à l'endroit de l'erreur grâce aux touches de déplacement du curseur et puis de taper le caractère correct.

En MSX-DOS par contre, l'éditeur n'utilise pas la technique du plein-écran mais un éditeur-ligne par compatibilité avec le MS-DOS qui est le système d'exploitation des PC compatibles.

Vous aurez déjà remarqué que pour valider une commande du DOS, il suffit d'enfoncer la touche RETURN comme en BASIC. De même, pour corriger une commande tant que l'on n'a pas fait RETURN, la touche BS (BackSpace) et la touche ← (curseur à gauche) ont toutes deux pour effet d'effacer le caractère précédent le curseur et puis de positionner le curseur à l'emplacement effacé.

Par contre, les touches suivantes ne fonctionnent pas comme en BASIC :

AZERTY	QWERTY	
SUP	(DEL)	Suppression caractère
INS		Mode insertion
EFF/DEP	(CLR/HOME)	Effacement écran / curseur en haut à gauche

et les trois touches de déplacement du curseur vers le haut, le bas et à droite avec lesquelles vous avez sans doute déjà remarqué qu'il n'est pas possible de déplacer le curseur sur une ligne supérieure ou inférieure ou encore à droite.

L’éditeur-ligne du MS-DOS et du MSX-DOS a la particularité de ne traiter qu’une ligne à la fois et de disposer d’une mémoire de la dernière ligne posée. Cette mémoire porte le nom de « TEMPLATE » dans le jargon anglais du MS-DOS. Pour la facilité d’expression, nous emploierons le terme « Tampon d’entrée » dans la suite de ce chapitre.

Ce tampon d’entrée se remplit automatiquement lorsque vous posez un texte quelconque après l’indicatif A> du DOS. Ainsi, si vous posez le texte ci-dessous suivi de RETURN, nous allons pouvoir expérimenter quelques possibilités :

```
A>REM Petit texte pour essayer l’éditeur
A>
```

L’indicatif A> s’est affiché sur la ligne suivante dès que vous avez enfoncé la touche RETURN. Enfoncez maintenant la touche « Curseur vers le bas » et vous verrez que le contenu du tampon d’entrée c’est-à-dire la ligne que nous avons tapée précédemment sera immédiatement affichée et le curseur restera en fin de ligne pour vous permettre de la modifier ou de la valider par RETURN.

Essayez maintenant la touche « Curseur vers le haut » et vous voyez que la ligne qui vient d’être rappelée disparaît immédiatement à l’exception de A>.

Si vous enfoncez maintenant de façon répétitive la touche « Curseur à droite », vous voyez réapparaître la ligne caractère par caractère de même que la touche « Curseur à gauche » le fait disparaître caractère par caractère :

```
A> REM Petit texte pour essayer l’éditeur
A> REM Petit tex
```

Enfoncez « Curseur vers le bas » pour ré-afficher la ligne complète et faitre RETURN.

La touche SELECT suivie d’un caractère fait apparaître le contenu du tampon d’entrée jusqu’au caractère spécifié après SELECT mais le tampon d’entrée ne sera pas modifié.

```
A>REM Petit texte pour essayer l’éditeur
A>     (Enfoncez SELCT + p)
A>REM Petit texte
```

Enfoncez SELECT + é et la ligne précédente se modifiera en

```
A>REM Petit texte pour essayer l’
```

Nous allons maintenant étudier des fonctions qui jouent uniquement sur le contenu tampon d’entrée sans avoir d’effet visible à l’écran.

D’abord la touche SUP qui saute dans le tampon d’entrée le caractère sur lequel le curseur se trouve. Pour visualiser le résultat, appuyez sur « Curseur vers le bas » :

```
A>REM Petit texte pour essayer l’éditeur
```

A>(SUP + « Curseur vers le bas »)
A>EM Petit texte pour essayer l'éditeur

Si vous enfoncez plusieurs fois de suite la touche SUP, vous sauterez autant de caractères. Le tampon d'entrée n'est pas modifié.

La touche EFF saute dans le tampon d'entrée les caractères depuis l'endroit du curseur jusqu'au caractère spécifié après EFF mais non compris celui-ci. Pour visualiser le résultat, appuyez sur « Curseur vers le bas ». Ici aussi, le tampon d'entrée ne sera pas modifié.

A>REM Petit texte pour essayer l'éditeur
A>(EFF + p et puis « Curseur vers le bas »)
A>pour essayer l'éditeur

La touche DEP permet d'envoyer la ligne telle qu'elle est affichée à l'écran dans le tampon d'entrée. Un caractère @ d'affiche en fin de cette ligne et le curseur apparaît sous le premier caractère de la ligne sans indicatif A>. Cette touche DEP est donc identique à la touche RETURN à l'exception que la commande n'est pas exécutée mais simplement rangée dans le tampon d'entrée. RETURN range ET exécute la commande affichée.

Finalement, il reste la touche INS qui sert à insérer du texte dans une ligne déjà posée.

A>REM Petit texte pour essayer l'éditeur
(SELECT + P)
A>REM
(INS + Voici un p)
(SUP = supprime le « P » de Petit)
(Curseur vers le bas)
A>REM Voici un petit texte pour essayer l'éditeur

TABLEAU RÉSUMÉ DES TOUCHES DE FONCTION

Dans le texte explicatif précédent, nous n'avons employé qu'une seule touche pour chacune des fonctions analysées. Dans le tableau ci-dessous, nous allons indiquer les autres touchez ou combinaisons de touches qui provoquent le même résultat. Le code placé entre parenthèses est le libellé de cette touche sur les claviers QWERTY.

TOUCHE	FONCTION
Curseur vers bas CTRL-_	Affiche le contenu du tampon d'entrée.
Cuseru vers haut CTRL-U CTRL-[ESC	Efface de l'écran la ligne affichée sans modifier le tampon d'entrée.
Curseur à droite	Affiche le prochain caractère du tampon d'entrée.

CTRL-\	
Curseur à gauche CTRL-H CTRL-J BKSP	Efface le dernier caractère affiché de l'écran sans modifier le tampon d'entrée.
DEP (HOME) CTRL-K	Range la ligne affichée dans le tampon d'entrée
SELECT car. CTRL-X car.	Affiche le tampon d'entrée jusqu'au caractère spécifié.
EFE (CLR) car. CTRL-L car.	Pointe dans le tampon d'entrée sur le caractère spécifié.
SUP (DEL)	Avance le pointeur d'un caractère dans le tampon d'entrée
INS CTRL-R	Place le système en mode insertion. Arrêt du mode insertion par une autre commande.

8.1.2 Les caractères de contrôle

Il existe cinq codes qui fonctionnent non seulement au cours de l'édition, mais aussi durant l'exécution de n'importe quel programme MSX-DOS.

Le plus important de ces codes est certainement CTRL-C qui est l'équivalent de CTRL-STOP du BASIC. En effet, CTRL-C permet de forcer l'arrêt d'un programme MSX-DOS et le retour à l'indicatif A> du DOS.

Le second en importance est certainement CTRL-S. Son but est d'arrêter le défilement de l'écran lors de l'affichage de texte. En effet, lors d'un TYPE d'un fichier très long ou lors d'un DIR d'un Directory fort remplie, l'écran se remplit très rapidement et les premières lignes disparaissent pour laisser la place aux suivantes. Il devient dès lors impossible de lire ces lignes. CTRL-S permet de stopper ce défilement et n'importe quelle autre touche, excepté CTRL-C, le redémarre. La technique la plus souvent employée consiste à laisser le majeur de la main gauche continuellement enfoncé sur la touche CTRL et avec l'index sur la touche S d'arrêter le défilement puis de le reprendre par un nouvel appui de l'index sur la touche S. Le premier code CTRL-S stoppe le défilement et le second le redémarre. Si vous appuyez sur CTRL-C lorsque le défilement est arrêté, le programme est stoppé et un retour à l'indicatif A> du DOS est exécuté.

Le code CTRL-P provoque que toute affichage à l'écran sera aussi envoyé vers l'imprimante. Si, par exemple, vous désirez imprimer le Directory d'une disquette, vous pouvez employer une des deux méthodes suivante : soit enfoncer d'abord CTRL-S puis poser DIR ou l'inverse. La différence sera que dans le premier cas la commande DIR sera elle-même imprimée suivie du Directory tandis que dans le second cas seul le Directory sera imprimé.

Le code CTRL-N désactive l'imprimante et annule ainsi le code CTRLP.

Finalement, le code CTRL-J provoque un retour au début de la ligne suivante sans exécution de la commande posée. Cela peut être utile pour couper en plusieurs lignes une commande très longue.

Mais attention, le code CTRL-J ne sert pas de marque de séparation et ne modifie pas le tampon d'entrée.

8.2 Les fichiers de commandes ou BATCH files

8.2.1 Généralités

Il existe deux types particuliers de fichiers en MSX-DOS.

Le premier est celui dont l'extension se termine par .COM et est composé d'un programme en langage machine qui s'implante toujours à l'adresse 100H de la mémoire RAM. Il peut être exécuté simplement en entrant le nom du fichier sans l'extension .COM après l'indicatif du DOS A> .

Le second type de fichier porte l'extension .BAT et ne contient pas un programme en langage machine mais plutôt une liste de noms de programmes ou de commandes du DOS. Il peut être exécuté aussi en entrant simplement le nom du fichier sans son extension .BAT après l'indicatif du DOS A>.

Le terme BAT provient du nom anglais BATCH qui signifie « LOT » et est reconnu par le MSX-DOS pour spécifier que son contenu est une liste 'un « lot ») de programmes plutôt qu'un seul programme en langage machine.

Lorsque vous tapez le nom d'un fichier .BAT, le système examine le Directory du disque pour trouver ce nom soit avec l'extension .COM soit avec l'extension .BAT. De ce fait, il faut toujours éviter de donner un même nom à un fichier .COM et .BAT car ces extensions ne suffisent plus à différencier les deux fichiers.

Si le nom est trouvé dans le Directory du disque, le système va analyser si l'extension porte l'extension .COM ou .BAT. Si l'extension est en .COM, le fichier sera chargé en 0100H de la mémoire dans la zone appelée TPA et le contrôle sera passé à ce programme par un JUMP 0100H.

Si l'extension porte le nom .BAT, alors le fonctionnement du système sera différent. Le fichier en question sera lu ligne par ligne et chacune de ces lignes doit être vue dans la première partir de ce chapitre (DIR-DATE-TYPE-etc).

Le programme ou la commande indiquée par la première ligne du fichier .BAT sera chargé et exécuté. Quand ce premier programme sera terminé, la seconde ligne du fichier .BAT sera lue et de nouveau le programme ou la commande indiquée sera chargé et exécuté. Ce processus va ainsi se poursuivre pour chaque ligne du fichier .BAT et lorsque le système lira la marque de fin de fichier (CTRL-Z ou 1AH), on reviendra à l'indicatif du MSX-DOS.

Voici d'abord un exemple de création d'un petit fichier de commandes que nous appellerons CONTENU.BAT. Le rôle de ce fichier de commandes sera de nous donner la date, l'heure (si vous avez un MSX2) et le contenu de la disquette du disque par défaut.

Pour créer ce fichier, nous pourrions utiliser un éditeur, mais comme nous n'en possédons pas tous, nous allons utiliser la commande COPY CON CONTENU.BAT qui permet d'envoyer vers le fichier CONTENU.BAT ce que nous taperons au clavier. Après la frappe de chaque ligne, nous enfoncerons la touche RETURN et quand nous aurons introduit toutes les lignes, nous poserons

CTRL-Z (fin de fichier) sur la ligne vierge suivante suivi d'un code RETURN. À ce moment, tout ce que nous aurons tapé sera sauvé sur le disque dans le fichier CONTENU.BAT.

```
A>COPY CON CONTENU.BAT
DATE
TIME
DIR/W
^Z
A>
```

Première constatation, le code CTRL-Z (fin de fichier) s'affiche par « ^Z ». C'est la méthode employée par le MSX-DOS (comme le CP/M ou le MS-DOS d'ailleurs) pour représenter les codes CONTROL.

Deuxième constatation, si nous avons commis une faute de frappe sur une ligne précédente, il est impossible de remonter à cette ligne car l'éditeur que nous employons (COPY) est vraiment un éditeur très sommaire (voir chapitre 7.3). Il faudra reprendre toute l'opération depuis le début.

Troisième constatation, nous pouvons maintenant afficher le contenu de notre fichier en procédant comme suit :

```
A>TYPE CONTENU.BAT200DATE
TIME
DIR/W
A>
```

Et nous voyons que tout ce que nous avons tapé s'affiche à l'exception du CTRL-Z qui sert de marque de fin de fichier. Notre fichier étant maintenant créé, nous allons pouvoir l'utiliser en posant tout simplement son nom sans l'extension .BAT après l'indicatif du DOS.

```
A>CONTENU                          ← Vous posez ceci et RETURN
A>DATE                             ← Automatique
Current date is Sun 21-06-87       ← Automatique
Enter new date :                   ← Si date OK posez RETURN
A>TIME                             ← Automatique
Enter new time :                   ← Si heure OK posez RETURN
A>DIR/W                            ← Automatique
MSXDOS    SYS   COMMAND  COM  CONTENU.BAT    ← Automatique
3 files         35132 bytes freee            ← Automatique
A>
```

Nous pouvons constater que chaque commande s'affiche d'abord automatiquement derrière l'indicatif du DOS et puis qu'elle s'exécute. Si nous sommes satisfaites de la date te de l'heure nous n'avons qu'à taper RETURN et le système poursuit l'exécution du fichier de commandes.

Voici un autre exemple surtout utile pour ceux qui ont un MSX à deux lecteurs tels que les PHILIPS NMS 8255 ou NMS 8280. Il fonctionne également sur les machines à un seul disque, mais nécessitera alors des changements de disquettes. Il sert à créer une disquette contenant le MSX-DOS à partir d'une disquette vierge. Voici d'abord comment le créer :

```
A>COPY CON DOSDIK.BAT

REM CREATION D'UNE DISQUETTE MSX-DOS
REM ------------------------------------------------------

REM ATTENTION REPONDEZ B : A LA QUESTION SUIVANTE

VERIFY ON
FORMAT
COPY A:MSXDOS.SYS B :
COPY A:COMMAND.COM B :
^Z
A>
```

Dans cet exemple vous pouvez constater l'utilité de la commande REM qui sert à afficher des remarques qui seront visualisées lors de l'exécution du fichier. Remarquez également qu'on peut introduire une ligne blanche pour aérer la présentation lors de l'exécution. Pour exécuter ce fichier, placez d'abord la disquette à formater dans le lecteur B : et posez DOSDISK après l'indicatif du DOS.

```
A>DOSDISK
A>
A>REM CREATION D'UNE DISQUETTE MSX-DOS
A>REM ------------------------------------------------------
A>
A>VERIFY ON
A>FORMAT
Drive name ? (A, B) B
1 – Single side…
2 – Double side…
2
Strike a key when ready…
Format complete
A>COPY A:MSXDOS.SYS B :
1 Files copied
A>COPY A:COMMAND.COM B :
1 Files copied
A>
```

Si vous enfoncez CTRL-C durant l'exécution d'un fichier de commandes, le système produira le message suivant :

Terminate batch file (Y/N) ?

Si vous tapez Y, le système arrêtera l'exécution du fichier de commandes et l'indicatif du DOS réapparaîtra. Si vous tapez N, la commande qui a été interrompue par le CTRL-C restera inachevée mais le reste des commandes du fichier s'exécutera. Il n'est donc pas conseillé d'interrompre un

fichier de commandes par CTRL-C ; la commande PAUSE a été spécialement créée à cet effet comme vous le montre l'exemple suivant :

```
A>COPY CON DELETE.BAT

REM PROGRAMME D'EFFACEMENT DES FICHIERS .TST
REM ---------------------------------------------------------------------

REM TAPEZ RETURN POUR CONTINUER LE BATCH FILE
PAUSE TAPEZ CTRL-C POUR ARRETER LE BATCH FILE

DEL *.TST
^Z
```

La commande PAUSE agit un peu comme REM en ce sens qu'elle s'affiche à l'écran et que le texte qui la suit permet d'indiquer ce qu'il y a lieu de faire mais elle diffère aussi de REM par le fait que le système va permettre une PAUSE c'est-à-dire attendre l'enfoncement de CTRL-C pour arrêter ou n'importe quelle autre touche pour continuer le fichier de commandes. Voici l'exécution de ce batch file :

```
A>DELETE
A>
A>REM PROGRAMME D'EFFACEMENT DES FICHIERS .TST
A>REM ---------------------------------------------------------------------
A>
A>REM TAPEZ RETURN POUR CONTINUER LE BATCH FILE
A>PAUSE TAPEZ CTRL-C POUR ARRETER LE BATCH FILE
Strike a key when ready...
A>
A>DEL *.TST
A>                    Tapez CTRL-C pour arrêter le fichier ou n'importe quoi pour continuer.
```

La technique de la commande PAUSE permet donc d'arrêter un batch file entre deux commandes de ce fichier sans tomber dans le travers décrit plus haut.

Dans les manipulations de disquettes lors du travail avec un seul lecteur, il est possible de se tromper. Lorsque le système a besoin de la disquette contenant le fichier de commandes, il affichera le message suivant si vous placez une disquette ne contenant pas ce fichier de commandes :

```
Insert disk with batch file
and strike a key when ready
```

Placez à ce moment la bonne disquette et enfoncez n'importe quelle touche. Le fichier de commandes se poursuivra alors avec la commande suivante.

Sachez également que l'on peut introduire dans un fichier de commandes des programmes .COM quelle que soit leur origine. Ainsi, si le programme JOUR.COM permet d'afficher le jour en

français et la date, on peut l'introduire dans un fichier de commandes en posant le nom du programme sans l'extension .COM.

On peut également appeler un autre fichier de commandes, pour autant qu'il soit la dernière commande du premier fichier de commandes. Voici un exemple de ces deux dernières possibilités :

```
A>COPY CON TEST.BAT
DATE        → Commande MSX-DOS DATE
JOUR        → Appelle le programme JOUR.COM
DELETE      → Appelle le fichier de commandes  DELETE.BAT décrit plus haut
^Z
```

En voici l'exécution :

```
A>TEST
A>DATE
Current date is Sun 21-06-1987
Enter new date :
A>JOUR                  ← Appel JOUR.COM
Dimanche 21-06-1987     ← Exécution JOUR.COM
A>DELETE                ← Appel DELETE.BAT
A>
A>REM PROGRAMME D'EFFACEMENT DES FICHIERS .TST
A>REM ----------------------------------------------------------------
A>
A>REM TAPEZ RETURN POUR CONTINUER LE BATCH FILE
A>PAUSE TAPEZ CTRL-C POUR ARRETER LE BATCH FILE
Strike a key when ready...
A>
A>DEL *.TST
A>
```

8.2.2 Le fichier AUTOEXEC.BAT

Il est possible de créer un fichier de commandes spécial qui doit porter obligatoirement le nom AUTOEXEC.BAT. Ce fichier a la particularité d'être appelé et exécuté automatiquement quand vous allumez votre ordinateur avec une disquette contenant les fichiers MSXDOS.SYS et COMMAND.COM.

Effectivement, quand le MSX-DOS est démarré, il vérifie dans le Directory si AUTOEXEC.BAT existe, et si tel est le cas, il sera exécuté en lieu et place de la demande de la date (MSX1).

Grâce à ce type de fichier de commandes, il est par exemple possible d'afficher la date et l'heure, de se placer en mode Vérification et d'afficher le Directory chaque fois que vous démarrez le système avec cette disquette comme dans l'exemple ci-dessous ou encore d'appeler un programme important se trouvant dans cette disquette.

```
A>COPY CON AUTOEXEC.BAT
DATE
```

```
TIME
VERIFY ON
DIR
^Z
A>
```

8.2.3 Les arguments d'un fichier de commandes

Tel que nous l'avons vu jusqu'à présent, l'inconvénient du fichier de commandes est qu'il exécute une tâche bien précise sans aucune possibilité de variante. Ainsi, dans l'exemple suivant, le fichier de commandes recopie tous les fichiers .TST en .TXT et efface les fichiers .TST d'origine.

```
A>COPY CON TSTTXT.BAT
COPY *.TST *.TXT
DEL *.TST
^Z
```

Si nous voulions copier les fichiers .RND vers .TST et effacer les fichiers .RND d'origine, il aurait fallu créer un autre fichier de commandes.

Grâce aux arguments du fichier de commandes, nous allons voir qu'il est possible de créer un seul fichier de commandes et lui faire exécuter des tâches différentes bien que similaires. Regardons d'abord l'exemple de création suivant :

```
A>COPY CON COPDEL.BAT

COPY %1 B :
DEL %1
^Z
```

Nous reconnaissons immédiatement la commande COPY vers le disque B : mais il ne s'agit pas ici de copier le fichier qui porte le nom de %1. Le symbole %1 représente en quelque sorte une variable dont le contenu va être déterminé non pas à la création du fichier mais bien au moment de son appel. Ainsi si nous posons :

```
A>COPDEL TEST.BAS
```

COPDEL est le nom du fichier de commandes et TEST.BAS est appelé le premier argument de la commande. Ce premier argument va être introduit dans la variable %1 et dès lors :

```
COPY %1 B :        sera interprété en      COPY TEST.BAS B :
DEL %1             sera interprété en     DEL TEST.BAS
```

ce qui provoque la copie du fichier TEST.BAS vers le disque B : et l'effacement du fichier TEST.BAS sur le disque par défaut.

A>COPDEL *.TST

provoquera la copie de tous les fichiers .TST vers le disque B : et l'effacement de ces mêmes fichiers du disque par défaut.

Le symbole %1 représente donc une variable qui sera remplie par le premier argument de la commande. Mais le système est bien plus riche, car il existe un symbole pour 9 variables dont le contenu proviendra des 9 arguments maximum que l'on peut ajouter au nom du batch file :

A>COPY CON DATEHEUR.BAT

DATE %1-%2-%3
TIME %4-%5-%6
^Z

Nous voyons ici que 6 variables ont été utilisées (%1 à %6). %1 doit recevoir le jour du mois, %2 le mois, %3 l'année, %4 les heures, %5 les minutes et %6 les secondes. Le premier argument qui suivra l'appel DATEHEUR.BAT sera affecté à la variable%1, le deuxième argument à la variable %2, et ainsi de suite jusqu'à %6.

AWDATEHEUR 21 9 1987 14 15 10
A>
A>DATE 21-09-1987
A>TIME 14:15:10
A>

Par l'exemple ci-dessus, on voit que les arguments doivent être séparés par un espace, qu'ils sont affectés aux variables %1 à %6 dans l'ordre de positions de ces arguments dans la ligne de commande et que les commandes DATE et TIME sont affichées au moment de leur exécution avec le CONTENU des variables plutôt qu'avec le nom de ces variables.

Autre exemple :

A>COPY CON SUPERCOP.BAT
TYPE %1
REM EST-CE BIEN CE FICHIER QUE VOUS VOULEZ COPIER ?
PAUSE OUI→ RETURN, NON→ CTRL-C
REM SOUS LE NOM : %2 ?
PAUSE OUI → RETURN, NON → CTRL-C
COPY %1 %2
REM FAUT-IL EFFACER %1 ?
PAUSE OUI → RETURN, NON → CTRL-C
DEL %1
^Z

Exécution :

```
A>SUPERCOP A:TOTO.TST B:TEXTE.TXT
A>TYPE A:TOTO.TST
Petit fichier d'essai de deux lignes.
Ceci est la deuxième ligne de TOTO.TST
A>REM EST-CE BIEN CE FICHIER QUE VOUS VOULEZ COPIER ?
A>PAUSE OUI→ RETURN, NON→ CTRL-C
Strike a key when ready...
A>REM SOUS LE NOM : B:TEXTE.TXT ?
A>PAUSE OUI → RETURN, NON → CTRL-C
Strike a key when ready...
A>COPY A:TOTO.TST B:TEXTE.TXT
A>REM FAUT-IL EFFACER A:TOTO.TST?
A>PAUSE OUI → RETURN, NON → CTRL-C
Strike a key when ready...
A>DEL A:TOTO.TST
```

Il existe enfin une dixième variable appelée %0 dont l'emploi est facultatif et qui reçoit toujours le même contenu, à savoir le nom même de votre fichier de commandes. Ainsi dans l'exemple précédent, on aurait pu ajouter au début du fichier de commandes les deux lignes de remarques suivantes :

```
REM %0 : PROGRAMME DE COPIE CONVIVIAL
REM ----------------------------------------------------
```

et à l'exécution, cela aurait donné :

```
A>REM SUPERCOP : PROGRAMME DE COPIE CONVIVIAL
A>REM ----------------------------------------------------
etc.
```

Une dernière remarque. Si vous employez la variable%3, par exemple, la ligne de commande que vous devez poser pour appeler le fichier de commandes doit contenir 3 arguments obligatoirement. Sinon, le système ne trouvant pas de troisième argument produira le message « File not found » :

```
A>COPY CON TEST.BAT
TYPE %3
^Z

A>TEST TEXTE.TXT
A>TYPE
File not found
A>
```

Par contre la méthode suivante marchera car « AA » servira de premier argument, « BB » de deuxième et « TEXTE.TXT » sera donc bien le troisième argument :

```
A>TEST AA BB TEXTE.TXT
A>TYPE TEXTE.TXT
Contenu du fichier TEXTE.TXT
A>
```

9) ANNEXE : Les différents types de disques

9.1 Les disques de 3”1/2

	TYPES DE DISQUE			
Media descriptor	F8	F9	FA	FB
Nombre de faces	1	2	1	2
Nombre de pistes	80	80	80	80
Nombre de secteurs par piste	9	9	8	8
Taille du secteur	512	512	512	512
1er secteur FAT	1	1	1	1
Taille de la FAT	1024	1024	1024	1024
Nombre de FAT	2	2	2	2
1er secteur du Directory	5	7	3	5
Nombre de secteurs dans le Directory	7	7	7	7
Nombre d’entrées dans le Directory	16	16	16	16
Nombre maximum d’entrées Directory	112	112	112	112
1er secteur pour fichier	12	14	10	12
Taille du cluster	1024	1024	1024	1024
Nombre de clusters pour fichiers	354	713	315	634
Nombre de secteurs total	720	1440	640	1280
Capacité utilisateur	362496	730112	322560	649216
Capacité formatée	368640	737280	327680	655360

9.2 Les disques de 5”1/4

	TYPES DE DISQUE			
Media descriptor	FC	FD	FE	FF
Nombre de faces	1	2	1	2
Nombre de pistes	40	40	40	40
Nombre de secteurs par piste	9	9	8	8
Taille du secteur	512	512	512	512

1er secteur FAT	1	1	1	1
Taille de la FAT	1024	1024	512	512
Nombre de FAT	2	2	2	2
1er secteur du Directory	5	5	3	3
Nombre de secteurs dans le Directory	4	7	4	7
Nombre d'entrées par secteur Directory	16	16	16	16
Nombre maximum d'entrées Directory	64	112	64	112
1er secteur pour fichier	9	12	7	10
Taille du cluster	512	1024	512	1024
Nombre de clusters pour fichiers	351	354	313	315
Nombre de secteurs total	360	720	320	640
Capacité utilisateur	179712	362496	160256	322560
Capacité formatée	184320	368640	163840	327680

Table des matières

1) Chapitre 1 : Les disques du système MSX....2
1.1 Capacité en unités de disquettes....2
1.2 Constitution physique d'une disquette....2
1.3 Les différents types de disquettes....3
1.4 Le formatage....3
2) Chapitre 2 : Les noms des unités de disquettes et des fichiers....4
2.1 Les noms des unités de disquettes....4
2.2 Les noms de fichiers....5
2.2.1 Qu'est-ce qu'un fichier ?....5
2.2.2 Le fichier séquentiel....6
2.2.3 Le fichier à accès direct....6
2.2.4 La structure du nom de fichier....7
2.2.5 Les extensions réservées....8
2.2.6 Les fichiers périphériques....9
2.2.7 Les caractères de substitution....9
3) Chapitre 3 : Le Disk-BASIC....10
3.1 Les commandes de gestion des fichiers....11
3.1.1 SAVE....11
3.1.2 LOAD....12
3.1.3 BSAVE....13
3.1.4 BLOAD....14
3.1.5 MERGE....15
3.1.6 NAME....16

3.1.7 KILL....17
3.1.8 FILES – LFILES....18
3.1.9 COPY....19
3.1.10 RUN....21
3.1.11 MAXFILES....22
3.2 Les Call du Disk-Basic....23
3.2.1 CALL FORMAT....23
3.2.2 CALL SYSTEM....24
3.3 La manipulation d'un fichier séquentiel....24
3.3.1 L'ouverture d'un fichier séquentiel....25
3.3.2 L'écriture dans un fichier séquentiel....26
3.3.3 La lecture d'un fichier séquentiel....29
3.3.4 La fermeture d'un fichier séquentiel....30
3.3.5 Exemples de programmes avec fichier séquentiel....31
3.3.5.1 Création ou extension d'un fichier d'adresses....31
3.3.5.2 Consultation d'un fichier d'adresses....32
3.3.6 LINE INPUT#....33
3.4 Les fonctions spécifiques aux fichiers séquentiels....34
3.4.1 INPUT$....34
3.4.2 End of file....34
3.4.3 LOCate....35
3.4.4 LOF Length Of File....36
3.5 La manipulation d'un fichier à accès direct....36
3.5.1 L'ouverture d'un fichier à accès direct....38
3.5.2 FIELD : création de champs (zones)....39
3.5.3 LSET (LEFT SET – PLACER A GAUCHE) RSET (RIGHT SET – PLACER A DROITE)....42
3.5.4 MAKE STRINGS....44
3.5.5 PUT....46
3.5.6 GET....47
3.5.7 Conversion d'une chaîne numérique en valeurs numériques....49
3.5.8 Fermeture d'un fichier à accès direct....50
3.6 Programme exemple de fichier à accès direct....50
3.6.1 Création d'un fichier....50
3.6.2 Lecture d'un fichier....52
3.6.3 Mise à jour d'un fichier....55
3.7 Les instructions utilitaires....56
3.7.1 DSKF DiSK Free space....56
3.7.2 DSKI$ DiSK Input....56
3.7.3 DSKO$ DiSK Output....57
3.7.4 VARPTR VARiable PoinTeR....58
4) Chapitre 4 : l'organisation de la disquette....59
4.1 Le découpage de la disquettes....59
4.2 Le cluster....61
4.3 Le record....61
4.4 L'extent....62
4.5 Le Boot sector....62
4.6 Le Directory....64
4.7 La FAT (File Allocation Table)....65
4.8 Structure des fichiers....71

4.8.1 Le fichier-programme BASIC sauvé en binaire compressé....71
4.8.2 Le fichier de programme BASIC sauvé en ASCII....72
4.8.3 Le fichier binaire ou langage machine....72
4.8.4 Les fichiers séquentiels....73
4.8.5 Les fichiers à accès direct....74
4.8.6 Les fichiers MSX-DOS....75
4.9 Le formatage physique d'une piste....75
5) Chapitre 5 : la carte mémoire de la zone RAM utilisée par le MSX-DOS et le Disk-BASIC..76
5.1 La carte mémoire générale....77
5.1.1 Taille des zones mémoire pour le DOS et le Disk-BASIC....77
5.1.2 Initialisation du système MSX avec disquettes....77
5.1.3 Occupation mémoire commune MSX-DOS/MSX-Basic....78
5.1.4 Différence entre les occupations mémoires MSX-DOS et Disk-BASIC....82
5.1.5 Zone mémoire réservée au MSX-DOS....85
5.2 Table des contrôleurs des unités de disquettes....86
5.3 Table des zones de travail réservées par la ROM Disk....87
5.4 Le Disk Parameter Block....87
5.5 Le Controler Work Area....89
5.6 Le File Control Block....90
5.6.1 Le FCB du CP/M....91
5.6.2 Le FCB du MSX-DOS et du Disk-BASIC....93
6) Chapitre 6 : les points d'entrée du Disk-ROM....93
6.1 4010H DSKIO DiSK Input Output....94
6.2 4031H DSKCHG DiSK ChanGe query....98
6.3 4016H GETDPB GET Disk Parameter Block....100
6.4 4019H CHOICE CHOICE of type of format....101
6.5 DSKFMT DiSK ForMaT....102
6.6 FORMTM FORMaT Msx-dos....104
6.7 FORMTK FORMaT with Keyboard choice....104
6.8 401FH DSKSTP DiSK StoP (dernières versions)....105
6.9 4029H ALLSTP All disks StoP (dernières versions)....107
6.10 4022H BASIC return to BASIC....107
6.11 402DH GETSLOT GET the controller SLOT number....111
6.12 0000 GETTOP GET TOP of user memory....111
7) Chapitre 7 : BASIC-DOS et MSX-DOS en langage machine....111
7.1 Pourquoi un BASIC-DOS et un MSX-DOS ?....112
7.2 Le BASIC-DOS....112
7.3 Le MSX-DOS....114
7.4 La page 0 du MSX-DOS....115
7.5 Les fonctions du BASIC-DOS et du MSX-DOS....120
7.5.1 Fonction 00 SYSTEM RESET....121
7.5.2 Fonction 01 CONSOLE INPUT....121
7.5.3 Fonction 02 CONSOLE OUTPUT....122
7.5.4 Fonction 03 AUXILIARY INPUT....123
7.5.5 Fonction 04 AUXILIARY INPUT....124
7.5.6 Fonction 05 LIST OUTPUT....124
7.5.7 Fonction 06 DIRECT CONSOLE INPUT/OUTPUT....125
7.5.8 Fonction 07 MSX-DOS DIRECT INPUT....126
7.5.9 Fonction 08 DIRECT INPUT WITH TEST....127
7.5.10 Fonction 09 STRING OUTPUT....128

7.5.11 Fonction 0A STRING INPUT....129
7.5.12 Fonction 0B GET CONSOLE STATUS....131
7.5.13 Fonction 0C GET VERSION NUMBER....132
7.5.14 Fonction 0D DISK RESET....132
7.5.15 Fonction 0E SELECT DEFAULT DISK....133
7.5.16 Fonction 0F OPEN FILE....134
7.5.17 Fonction 10 CLOSE....135
7.5.18 Fonction 11 SEARCH FIRST....137
7.5.19 Fonction 12 SEARCH FOR NEXT....138
7.5.20 Fonction 13 DELETE FILE....139
7.5.21 Fonction 14 SEQUENTIAL READ....140
7.5.22 Fonction 15 SEQUENTIAL WRITE....142
7.5.23 Fonction 16 CREATE FILE....143
7.5.24 Fonction 17 RENAME FILE....145
7.5.25 Fonction 18 LOGGIN VECTOR....146
7.5.26 Fonction 19 GET DEFAULT DRIVE NAME....147
7.5.27 Fonction 1A SET DMA (DTA) ADDRESS....148
7.5.28 Fonction 1B GET ALLOCATION....149
7.5.29 Fonction 1C non utilisée en MSX-DOS....152
7.5.30 Fonction 1D non utilisée en MSX-DOS....152
7.5.31 Fonction 1E non utilisée en MSX-DOS....152
7.5.32 Fonction 1F non utilisée en MSX-DOS....153
7.5.33 Fonction 20 non utilisée en MSX-DOS....153
7.5.34 Fonction 21 RANDOM READ....153
7.5.35 Fonction 22 RANDOM WRITE....153
7.5.36 Fonction 23 GET FILE SIZE....156
7.5.37 Fonction 24 SET RANDOM RECORD....158
7.5.38 Fonction 25 non utilisée en MSX-DOS....159
7.5.39 Fonction 26 RANDOM BLOCK WRITE (MSX-DOS)....159
7.5.40 Fonction 27 RANDOM BLOCK READ (MSX-DOS)....161
7.5.41 Fonction 28 RANDOM WRITE WITH ZERO FILL....163
7.5.42 Fonction 29 NO FUNCTION (MSX-DOS)....164
7.5.43 Fonction 2A GET DATE AND TIME (MSX-DOS)....164
7.5.44 Fonction 2B SET DATE (MSX-DOS)....166
7.5.45 Fonction 2C GET TIME (MSX-DOS)....167
7.5.46 Fonction 2D SET TIME (MSX-DOS)....169
7.5.47 Fonction 2E SET/RESET VERIFY FLAG (MSX-DOS)....169
7.5.48 Fonction 2F ABSOLUTE DISK READ (MSX-DOS)....170
7.5.49 Fonction 30 ABSOLUTE DISK WRITE (MSX-DOS)....171
7.6 Direct BIOS access....172
8) Chapitre 8 : L’éditeur MSX-DOS et les fichiers BATCH....174
8.1 L’éditeur du MSX-DOS....174
8.1.1 Généralités....174
8.1.2 Les caractères de contrôle....177
8.2 Les fichiers de commandes ou BATCH files....178
8.2.1 Généralités....178
8.2.2 Le fichier AUTOEXEC.BAT....182
8.2.3 Les arguments d’un fichier de commandes....183
9) ANNEXE : Les différents types de disques....186
9.1 Les disques de 3’’1/2....186

9.2 Les disques de 5’’1/4.....186