SIVEA S.A. Département Micro - Informatique
20, rue de Léningrad 75008 Paris - France

MANUEL D'UTILISATION

(EN FRANCAIS)

DU DISK OPERATING SYSTEM

(SYSTEME D'EXPLOITATION DE DISQUETTE)

POUR TRS 80

D'APRES DOS 2-3 TANDY
" " NEW DOS PLUS APPARAT (INC)

AUTEUR **J.L BERARDO**

SOMMAIRE

SOMMAIRE SUITE

Le présent ouvrage a pour but de vous présenter ce qu'est un DOS ('Disk Opérating System' ou système d'exploitation des disques) et comment se servir des disquettes. Une disquette est une feuille de plastique magnétisée analogue à la bande magnétique des cassettes. Cette feuille de plastique est protégée en permanence par une enveloppe de carton qui a en même temps un rôle de nettoyage. Dans le coin supérieur droit du carton, il y a une petite échancrure. Cette échancrure, si elle est libre, permet au lecteur de disquette de voir qu'il a le droit d'écrire des informations sur la disquette. Si cette échancrure n'est pas libre (par exemple parce qu'on y a placé une étiquette par dessus), alors le lecteur ne pourra pas écrire sur cette disquette (qui est 'write protected' ou protégée à l'écriture). Si vous essayez d'écrire sur une disquette protégée, le système vous affichera un message d'erreur (variable suivant les commandes utilisées). Si vous tenez à écrire sur cette disquette, il faut alors retirer l'étiquette de protection de façon à laisser l'échancrure libre.

Une disquette est très fragile, il faut donc les manipuler avec précautions : ne jamais les plier ou les tordre, ne pas mettre les doigts sur le grand trou ovale, ne jamais forcer en introduisant ou en enlevant une disquette du lecteur, ne jamais essayer d'enlever ou de placer une disquette dans le lecteur en fonctionnement (la petite lampe témoin rouge située sur le devant du lecteur est alors allumée). Bien attendre l'arrêt total du moteur pour ouvrir la porte du lecteur. Sinon vous risquez de détruire la disquette et / ou son contenu !

Un DOS est un ensemble de programmes permettant d'utiliser les disquettes comme support de stockage de la même façon que des cassettes. Les disquettes ont de nombreux avantage sur les cassettes : très grande facilité d'accès à un programme (quelques secondes où que soit situé le programme), très grande fiabilité (1 erreur de lecteur ou d'écriture non détecté sur 1 million), très grande vitesse de transfert disquette. Mémoire centrale (en général de l'ordre de 10 K octets par seconde, contre 62,5 octets par seconde pour les cassettes), possibilité de gérer

des fichiers de données (ce qui est pratiquement infaisable sur une cassettes).

Les disquettes, lorsqu'elles sont vendues, sont vierges ; c'est-à-dire qu'il n'y a rien dessus. Or le DOS ne peut pas écrire sur des disquettes vierges ; il faut qu'elles soient préparées ; si vous n'avez qu'un seul lecteur, toutes vos disquettes doivent comporter le DOS (voir la commande COPY et la commande KILL). Si vous avez plusieurs lecteurs, alors seul le lecteur 0 doit contenir le DOS ; les autres peuvent recevoir des disquettes préparées que l'on nomme disquettes formattées (voir la commande FORMAT).

L'ouvrage se compose de cinq parties :

1) Les commandes du DOS :

   COPY (duplication de disquettes et copies de programmes);
   DIR (repertoire du contenu d'une disquette);
   FREE (place libre sur une disquette);
   AUTO (chargement automatique à allumage);
   ATTRIB & PROT (utilisation des mots-de-passe);
   KILL (destruction d'un programme);
   FORMAT (préparation des disquettes vierges);
   RENAME (changement du nom d'un programme);
   CLOCK, DATE, TIME (utilisation de l'horloge en temps réel),
   APPEND (mélange de deux programmes);
   LIST & PRINT (affichage & impression d'un programme);
   DUMP & LOAD (utilisation des programmes en langage machine);
   VERIFY, TRACE, DEBUB (programme utilitaires);
   BASIC & BASIC2 (accès au Basic).

2) Les commandes du Basic

   SAVE (sauver un programme sur disquette);
   LOAD (charger un programme d'une disquette);
   RUN (charger et exécuter un programme d'une disquette);
   KILL (détruire un programme);
   MERGE (mélanger deux programmes);
   RENUM (renumérotter automatiquement les lignes d'un programme);
   REF (affichage d'une liste de références);
   LIST-EDIT-DELETE (nouvelles formes).

3) Les instructions BASIC supplementaires

&H, &O (constantes en hexadécimal et en octal);
CMD "D" (commande de DEBUG);
CMD "S" (retour au DOS),
CMD "T" et CMD "R" (commande de l'horloge);
DEFFN (définition de fonctions multiarguments);
INSTR (recherche de sous-chaîne dans une chaîne de caractères);
LINE INPUT (entrée de lignes de texte),
MID$ (remplacement d'un morceau de chaîne de caractère);
TIME (accès à l'horloge);
DEFUSR et USRn (appel de sous routines en langage machine).

4) Les fichiers de données

accès séquentiel (OPEN, CLOSE, INPUT#, LINE INPUT#, PRINT#
accès aléatoire (OPEN, CLOSE, FIELD, GET, PUT, LSET, RSET, EOF, LOF, conversions)

5) Les programmes utilitaires :

LMOFFSET (mettre un programme en langage machine d'une cassette sur une disquette);
EDTASM (éditor assembler, version disques);
DISASSEM (désassembleur);
LEVEL1 et LV1DSKSL (utilisation du niveau I sur disques);
SUPERZAP (accès aux disques);
DISKDUMP/BAS (accès direct aux disques du basic).

1ERE PARTIE

COMMANDES DU DOS

1°) COPY

Cette commande permet de faire des copies de disquettes ou de fichiers. Il y a plusieurs façons de l'utiliser :

1- COPY␣DESIGN1␣TO␣DESIGN2
2- COPY␣$DESIGN1␣TO␣DESIGN2
3- COPY␣:D␣DESIGN1␣TO␣DESIGN2
4- COPY␣:D␣$DESIGN1␣TO␣DESIGN2
5- COPY␣:D␣TO␣:E␣MM/JJ/AA

␣ Ce signe signifie qu'il faut taper un espace.
D et E sont des numéros de lecteurs de disquettes (0,1,2,3). Ils peuvent être égaux.

DESIGN1 est la désignation du fichier "source"
DESIGN2 est la désignation du fichier "destination"

Le symbole $ signifie que le fichier "source", le fichier "destination" ou les deux fichiers sont sur une disquette contenant un système étranger ; c'est-à-dire un système autre que le NEWDOS.

Les formats 3 et 4 indiquent que les fichiers "source" et "destination" sont sur des disquettes différentes, mais que seul le lecteur numéro D doit être utilisé pour faire la copie.

Le format 5 permet de faire une duplication d'une disquette. MM/JJ/AA est la date de la création de la disquette (MM est le mois : 01 à 12, JJ est le jour : 01 à 31, AA est l'année : 00 à 99).

Le format : COPY␣ABCD/TTT:0␣TO␣ABCD/TTT:1 peut être abrégé de la façon suivante :

COPY␣ABCD/TTT:0␣TO␣:1

- Vous n'avez qu'un seul lecteur de disquette :

1) Pour faire une duplication de disquette, utilisez la commande.

COPY␣:0␣TO␣:0␣MM/JJ/AA

Le programme 'copy' vous donne alors toutes les indications nécessaires :

'Press "enter" when source diskette mounted on drive 0'
(appuyez sur "enter" lorsque vous aurez placé la disquette source dans le lecteur 0)

'Press "enter" when destination diskette mounted on drive 0'
(appuyez sur "enter" lorsque vous aurez placé la disquette destination dans le lecteur 0)

Après 2 ou 3 manipulations de disquettes, le système affiche :
'Full diskette copy done'

'Press "enter" when ** system ** diskette mounted on drive 0'
(copie complète de la disquette effectuée)

(appuyez sur "enter" lorsque vous aurez placé la disquette ** système ** dans le lecteur 0).

Note: La disquette ** système ** est celle qui contient le NEWDOS.

Il est très fortement conseillé de faire au moins une copie de la disquette contenant le NEWDOS de façon à en avoir une en réserve au cas où la première serait détruite par une fausse manoeuvre.

2) Pour copier un programme nommé ABCD/TTT d'une disquette à une autre, utilisez la commande :

COPY␣:0␣ABCD/TTT␣TO␣ABCD/TTT

Le programme vous donne alors toutes les indications nécessaires de la même façon qu'au 1e.

- Vous avez plusieurs lecteurs de disquettes :

1) Pour faire une duplication de disquette, utilisez la commande :

COPY␣:0␣TO␣:1␣MM/JJ/AA

Le programme affiche les deux messages :

'Press "enter" when source diskette mounted in drive 0'
'Press "enter" when destination diskette mounted in drive 1'

La copie est alors automatique, à la fin, le programme affiche :
' Full diskette copy done'

'Press "enter" when ** system ** diskette mounted in drive o'

2) Pour copier un programme d'une disquette à une autre :

Si la disquette "source" contient le NEWDOS, placez la dans le lecteur 0 et tapez :

COPY␣ABCD/TTT:0␣TO␣:1

Si la disquette source ne contient pas le NEWDOS, mais si la disquette destination le contient, placez la disquette destination dans le lecteur 0 et la disquette source dans le lecteur 1 et tapez :

COPY␣ABCD/TTT:1␣TO␣:0

Si aucune des deux disquettes ne contient le NEWDOS, placez une disquette le contenant dans le lecteur 0 et tapez :

COPY␣:1␣ABCD/TT␣TO␣ABCD/TTT

Dans tous les cas, suivez les indications données par le programme qui vous dit à chaque fois où mettre la disquette source et la disquette destination.

2) DIR

Cette commande permet d'afficher sur l'écran le nom des programmes contenus dans une disquette. (DIR vient de 'Directory' qui veut dire : répertoire). La commande a la forme suivante :

DIR [␣:D] [(param [,param...])]

D est le numéro du lecteur (0,1,2 ou 3). Si vous voulez le répertoire du lecteur 0 vous pouvez tapez seulement DIR.

Param représente l'un des paramètres suivants :

S : affiche les fichiers appartenant au système et les fichiers non invisibles.

I : affiche les fichiers invisibles et ceux n'appartenant pas au système.

A : affiche l'espace allöué à chaque fichier sur la dis quette.

Si aucun paramètre n'est donné, seuls les fichiers non invisibles sont affichés.

L'espace alloué à un fichier sur la disquette est affiché comme suit : LRL (logical record lenght : longueur de l'enregistrement logique). EOF (end of file : fin du fichier) : le plus grand numéro d'enregistrement utilisé. SIZE (taille) : mesuré en GRANules, (où 1 granule représente 1,25 K octects).

Exemples : (Placez la disquette NEWDOS dans le lecteur 0)
tapez :

```
DIR  enter
l'écran affiche alors :

FILE DIRECTORY .......DRIVE Ø          NEWDOS...IØ/Ø3/79

SUPERZAP/COM          DISKDUMP/BAS          LEVEL1/CMD
DIRCHECK/CMD          DISASSEM/CMD          LMOFFSET/CMD
EDT ASM/CMD           LV1 DSKL/CMD          SUPERZAP

DOS READY
-
```

Si vous tapez DIR␣:0␣(S) , vous obtenez :

```
FILE DIRECTORY ..... DRIVE Ø          NEWDOS .... 10/03/79

BOOT/SYS    SIP    DIR/SYS     SIP    SYSØ/SYS    SIP
SYS1/SYS    SIP    SYS2/SYS    SIP    SYS3/SYS    SIP
SYS4/SYS    SIP    SYS5/SYS    SIP    SYS6/SYS    SIP
SYS11/SYS   SIP    SYS12/SYS   SIP    SYS13/SIP   SIP
SUPERZAP/COM       DISKDUMP/BAS       LEVEL1/CMD
DIRCHECK/CMD       DISASSEM/CMD       LMOFFSET/CMD
EDTASM/CMD         LV1DSKSL/CMD       SUPERZAP

DOS READY
```

Si vous tapez DIR␣:0␣(I), vous obtenez

```
FILE DIRECTORY.......DRIVE Ø          NEWDOS .....IØ/Ø3/79

FORMAT/CMD   IP      SUPERZAP/COM           DISKDUMP/BAS
LEVEL1/CMD           DIRCHECK/CMD           BASIC/CMD    IP
DISASSEM/CMD         LMOFFSET/CMD           EDTASM/CMD
LV1 DSKSL/CMD        COPY/CMD     IP        SUPERZAP


DOS READY
-
```

Les lettres S,I et P apparaissant après un programme signifient :

S : le programme fait partie du système
I : le programme est invisible (n'apparait qu'avec le paramètre I)
P : le programme est protégé par un mot-de-passe.

Si vous tapez DIR␣:0␣(A), vous obtenez :

```
FILE DIRECTORY.......DRIVE Ø          NEWDOS....IØ/Ø3/79


SUPERZAP/COM       LRL = 256 / EOF =  26/ SIZE = 6 GRANS
DISKDUMP/BAS       LRL = 256 / EOF =  10/ SIZE = 2 GRANS
LEVEL1/CMD         LRL = 256 / EOF =  19/ SIZE = 4 GRANS
DIRCHECK/CMD       LRL = 256 / EOF =  13/ SIZE = 3 GRANS
DISASSEM/CMD       LRL = 256 / EOF =  20/ SIZE = 4 GRANS
LMOFFSET/CMD       LRL = 256 / EOF =   7/ SIZE = 2 GRANS
EDTASM/CMD         LRL = 256 / EOF =  32/ SIZE = 7 GRANS
LV1 DSKSL/CMD      LRL = 256 / EOF =   3/ SIZE = 1 GRANS
SUPERZAP           LRL = 256 / EOF =  54/ SIZE = 11 GRANS


DOS READY
```

Vous pouvez donner plusieurs paramètres : DIR:0(S,I), DIR:0(S,A),...,DIR:0(S,I,A)

qui combineront les diverses possibilités vues ci-dessus.

3) FREE

Cette commande permet de savoir quelle est la place restante sur une disquette.

La commande est : FREE enter et vous obtenez :

```
DRIVEØ...NEWDOS  IØ/Ø3/79     36 FILES,    5 GRANS
DRIVE1...NEWDOS  ØØ/ØØ/ØØ     38 FILES,    8 GRANS

DOS READY
```

La commande analyse les lecteurs les uns après les autres. L'affichage indique : le numéro du lecteur (DRIVEØ), le nom de la disquette (NEWDOS), la date de création de la disquette (IØ/Ø3/79), le nombre de fichiers possibles restants (36 FILES) et la place disponible en granules (5 GRANS). 1 granule = 1,25 K octets.

Une disquette formatée (voir commande : FORMAT) permet un maximum de 48 fichiers et a une place libre de 67 granules (85760 octects).

Une disquette contenant le NEWDOS, sans les programmes utilitaires permet un maximum de 45 fichiers et a une place libre de 45 granules (57600 octects).

(N'oubliez pas que le DOS doit être présent dans le lecteur 0).

4) AUTO

Cette commande permet le chargement automatique d'un programme executable au moment de la mise en route de l'ordinateur. Le DOS se charge puis va de lui même executer le programme demandé (tel que : passage en BASIC, affichage de l'heure, etc...)

Pour pouvoir utiliser cette commande il faut que la disquette ne soit pas protégée (si elle l'est vous obtenez le message : 'GAT WRITE ERROR').

La commande a la forme suivante :

AUTO␣commande

où commande représente une commande du DOS ou un programme exécutable (avec extension/CMD).

Exemples : AUTO␣BASIC enter : l'ordinateur passera directement en basic à la mise en route ;

AUTO␣CLOCK enter : l'ordinateur affichera l'heure.

Pour supprimer le chargement automatique, il suffit de taper : AUTO enter.

Si vous utilisé cette commande , mais si vous ne voulez pas charger le programme, il suffit de mettre l'ordinateur en route en appuyant sur la touche enter ; seul le DOS se chargera alors.

5) ATTRIB et PROT

Ces deux commandes permettent de proteger des fichiers ou une disquette en leurs attribuant des mots-de-passe. Elles permettent aussi de changer ces mots-de-passe.

Tous les fichiers présents sur une disquette sont protégés par deux motsde passe, un mot de passe d'accès et un mot de passe de mise à jour ('update'). Ces deux mots peuvent être identiques et peuvent être blancs. Le mot de passe de mise à jour permet à l'utilisateur qui le connait de faire tout ce qu'il désire avec les fichier (lecture, écriture, éffacement, etc...)le mot de passe d'accès ne permet à l'utilisateur le connaissant qu'un nombre de choses limité par le degré de protection donné dans la commande ATTRIB.

La commande ATTRIB s'utilise de la façon suivante :

ATRIB␣design␣(param,[param...])

où param désigne l'un des paramètres suivants :

I : rend le fichier invisible (c'est-à-dire qu'il n'apparait pas à la commande DIR mais seulement à la commande DIR (I)-)

Attention : rendre un fichier invisible est irréversible ! aucune commande ne pourra le rendre à nouveau visible.

ACC = mot de passe 1 : attribue mot de passe 1 comme nouveau mot de passe d'accès .

UPD = mot de passe 2: attribue mot de passe 2 comme nouveau mot de passe de mise à jour.

PROT = niveau : attribue niveau comme degré de protection offert par le mot de passe d'accès.

Les différents degrés de protection sont les suivants :

KILL : permet de tout faire
RENAME : permet de lire, écrire, exécuter et renommer un fichier.
WRITE : permet de lire, écrire et exécuter un fichier
READ : permet de lire et exécuter un fichier
EXEC : permet l'execution seulement

Exemples : vous voulez protéger un fichier nommé PAYE/TXT. Si vous tapez :

ATTRIB␣PAYE/TXT␣(ACC=SECRET,UPD=MOI,PROT=WRITE)

Alors PAYE/TXT. MOI permet l'accès total au fichier, alors que PAYE/TXT.SECRET permet de le lire, de l'exécuter, et d'écrire dessus.

Si vous tapez : ATTRIB␣PAYE/TXT.MOI (UPD=SECRET) , vous avez simplement changé le mot de passe du fichier et vous lui attribuezle mot de passe d'accès comme mot de passe de mise à jour autrement dit, vous avez supprimé le mot de passe

Pour faire disparaitre un mot de passe il faut taper :

ATTRIB␣PAYE/TXT.SECRET␣(ACC=) qui supprime le mot de passe d'accès c'est-à-dire que le fichier n'est plus protégé.

ATTRIB PAYE/TXT.SECRET (UPD :) agit de même

Un fichier protégé apparait suivi de la lettre P à la commande DIR. Ainsi tous les programmes du système sont protégés par un mot de passe (demeuré jusqu'ici inconnu !)

N'oubliez jamais un mot de passe car il ne peut pas être retrouvé.

Si vous avez un fichier protégé par deux mots de passe et dont le degré de protection est inférieur à RENAME [par exemple après la commande ATTRIB PAYE/TXT (ACC=SECRET,UPD=MOI,PROT= WRITE)] , si vous tapez la commande.

ATTRIB PAYE/TXT.SECRET (ACC=AUTRE)

le programme vous envoie le message : 'illegal accés attempted to protected file'

(accès illégale à un fichier protégé).

Par contre : ATTRIB PAYE/TXT (ACC=SECRET,UPD=MOI,PROT= KILL) autorise l'accès total au fichier PAYE/TXT avec les 2 mots de passe SECRET et MOI.

La commande PROT permet d'utiliser le mot de passe général de la disquette . Ce mot de passe est donné au momment de la création de la disquette soit par la commande FORMAT (voir plus loin) où il vous est demandé , soit par la copie d'une disquette où ce mot de passe est reporté sur la copie. (La disquette NEWDOS possède le mot de passe : PASSWORD).

Cette commande a la forme suivante :

PROT[␣:D][␣(param[,param])]

D est le numéro du lecteur où est la disquette (0,1,2 ou 3)

Param est l'un des paramètres suivants :

PW : permet de changer le mot de passe général
UNLOCK : enlève les mots de passe de tous les fichiers
LOCK : attribue le mot de passe général à tous les fichiers.

(Les deux paramètres LOCK et UNLOCK s'excluent mutuellement)

Pour pouvoir utiliser cette commande, la disquette ne doit pas être protégée (enlever la protection).

Exemples : PROT␣:1(LOCK) , PROT (PW) , PROT (PW,UNLOCK)-

Pour chacune de ces commandes le programme vous demandera le mot de passe général ('master password'); en cas de mot de passe inexact la commande sera rejetée.

6) KILL

Cette commande permet de détruire un fichier contenu dans une disquette. Cette commande à la forme suivante :

KILL Design

design étant la désignation du fichier, incluant le numéro de lecteur, l'extension,_...

Exemple : Si vous n'avez qu'un seul lecteur, vous savez qu'il faut que le DOS soit toujours dans le lecteur 0 (le seul que vous ayez). Or la disquette que l'on vous vous a fournit n'a que 5 granules (6,25 K octects) de libre à cause de tous les programmes utilitaires inclus dans cette disquette.

Avec la commande KILL vous pouvez faire une disquette contenant uniquement le NEWDOS. Il suffit de tapez les commandes (après avoir fait une copie de la disquette fournie):

KILL␣SUPERZAP/COM enter
KILL␣DIRCHECK/CMD enter
KILL␣EDTASM/CMD enter

etc... jusqu'à ce qu'il ne reste plus aucun programme. Par contre n'essayer pas de détruire les programmes du système (SYSØ....SYS13) ou les programmes BASIC/CMD,COPY/CMD,FORMAT/CMD car ils sont protégés par un mot de passe inconnu et inamovible (même à la commande PROT)

Vous pouvez alors placer vos propres programmes sur cette disquette car il y a beaucoup de place libre (le vérifier par la commande FREE).

7) FORMAT

Cette commande permet de préparer des disquette ne contenant pas le DOS. Elles possèdent 67 granules libres (85760 octects). Comme elles ne possèdent pas de DOS, ces disquettes doivent obligatoirement être placées dans les lecteurs 1,2 ou 3 mais

pas dans le lecteur 0 (sauf pour faire une copie).

La commande s'utilise de la façon suivante ;FORMAT enter

Le Programme vous demande alors :

'Wich drive is to be used ?'
(quel lecteur doit être utilisé ?) : répondre par le numéro (0,1,2 ou 3)

'Diskette name ?'
(nom de la disquette)

'Création date (MM/DD/YY) ?'
(date de la création (MM/JJ/AA) ?)

'Master password ?'
(mot de passe général ?)

'Dou you want to lock out any tracks ?'
(voulez vous verouiller des pistes (répondre par "Y" pour OUI et "N" pour NON) ?)

Ce verouillage ne sert que si la disquette doit contenir des systèmes étrangers au NEWDOS.

A ce niveau il y a 2 continuations possibles :

a) 'FORMAT rejected, diskette contains data !'
(formatage rejetté, la disquette contient des données !)

'Diskette to be formated anyway (reply "Y" or "N")?'
(la disquette doit-elle être formatée dans ce cas (répondre par "Y" pour oui et "N" pour non)?)

Si vous répondez oui, le formatage se fait, sinon la commande est rejetée.

b) Si la disquette ne contient rien, le formatage se fait.

Le NEWDOS accepte aussi la commande suivante :

FORMAT d, nom, MM/JJ/AA, mot de passe
ou FORMAT d, nom, MM/JJ/AA, mot de passe, Y
qui donne toutes les réponses aux questions posées par le programme directement à la commande dans le premier cas, si la disquette contient déja des informations, la commande est abandonnée (la Y autorise le formatage d'une disquette contenant des données).

8) RENAME

Cette commande permet à l'utilisateur de changer le nom du fifichier. Les caractéristiques du fichiers telles que le mot de passe, le degré de protection, l'invisibilité,.,, sont conservées.

L'utilisation est la suivante :

RENAME design1␣TO␣design2

design1 étant le nom du fichier qui doit avoir la forme nom/ext.passe:d ;design2 est le nouveau nom qui doit avoir la forme nom/ext.

La commande RENAME verifie aussi que le second nom n'est pas déja employé pour un fichier sur la même disquette (si cela se produit, la commande est abandonnée).

Exemple : vous avez un programme appelé CALCUL/BAS.SECRET sur la disquette située dans le lecteur 1 la commande : RENAME␣CALCUL/BAS.SECRET:1␣TO␣POLYNOME/BAS change le nom du programme en POLYNOME/BASE.SECRET (le mot de passe est conservé).

9) CLOCK-DATE-TIME

Ces trois commandes permettent d'utiliser l'horloge en temps réel incluse dans l'interface d'extension.

La commande CLOCK permet l'affichage de l'heure sur l'écran en haut et à droite (position 53 à 60). Pour mettre en route tapez : CLOCK enter ; pour stopper, tapez :CLOCK␣(OFF) enter. A la mise en route, l'horloge est toujours en position d'arrêt.

La programme n'ayant aucun moyen de deviner la date ou l'heure, au moment de l'allumage de l'ordinateur, la date est mise à 00/00/00 et l'heure à 00:00:00.

La commande DATE permet d'entrer la date dans le programme de la façon suivante :

DATE␣mm/jj/aa

Exemple : pour le 14 juillet 1979 tapez : DATE␣07/14/79 enter

Le NEWDOS ne remet pas la date au jour après 24 h; il faut la mettre manuellement !

La commande TIME permet de mettre l'horloge à l'heure en tapant :

TIME␣hh:mm:ss

Exemple : s'il est 15 h 13 m 20 s tapez : TIME␣15:13:20 enter

L'heure et la date sont accessibles à un programme BASIC grâce à la variable TIME$ (voir la troisième partie).

10) APPEND

Cette commande permet d'ajouter le contenu d'un fichier à la fin du contenu d'un autre fichier. L'utilisation de cette commande est particulierement interessante pour les fichiers de données car il existe d'autres commandes pour les programmes en langage machine (LOAD) et en BASIC (MERGE).

APPEND␣fichier1␣TO␣fichier2 :

La commande permet de copier le contenu du fichier1 à la suite du fichier2. Le fichier1 n'est pas affecté par cette commande, alors que le fichier2 contiendra les contenus des deux fichiers.

11) LIST-PRINT

Ces commandes permettent d'afficher sur l'écran (LIST) ou sur l'imprimante (PRINT) le contenu d'un fichier stocké en format ASCII comme par exemple les programme basic sauvé en mode A, les fichiers de données crées en mode séquentiel (print ≠ n, voir la partie n° 4) etc....

Le texte défilant en continu (surtout génant sur l'écran) pour faire une pausse appuyez sur les touches SHIFT @ ; pour continuer appuyez sur n'importe quelle touche.

12) DUMP-LOAD

La commande DUMP permet de charger le contenu de la mémoire RAM sur disquette. Ce contenu est généralement un programme en langage machine commençant à l'adresse aaaa (4 chiffres hexadécimaux), finissant à l'adresse bbbb (4 chiffres hexadecimaux) et ayant comme point de départ l'adresse cccc (4 chiffres hexadecimaux). La commande est alors :

DUMP␣nom␣(START=X'aaaa',END=X'bbbb',TRA=X'cccc')

L'adresse de fin (bbbb) doit être plus grande que l'adresse de début (aaaa).
L'adresse départ (cccc) n'est obligatoire, si elle est absente le programme la remplace par 402D qui est l'adresse d'entrée du DOS. L'adresse de début (aaaa) doit se situer après 6FFF (sinon le programme vous indique : 'start less than X '7000'' : début inférieur à 7000 H) mais le programme ouvre tout de même un fichier (vide).

La commande LOAD permet de charger un programme en langage machine dans la mémoire RAM et de retouner au DOS. On peut ainsi charger plusieurs programmes en mémoire, ceux-ci pouvant être appelés par un programme directeur en langage machine ou en basic (instruction USR).

Le programme à charger a été crée en général par la commande DUMP, ou par les programmes LMOFFSET, TAPEDISK, etc... pour utiliser cette commande, tapez :

LOAD␣design.[enter] (design est la désignation du programme).

13) VERIFY

Cette commande permet la vérification de toutes les écritures sur disquettes. Pour la mettre en route tapez VERIFY enter ; pour l'arrêter, tapez VERIFY␣(OFF)[enter]. A la mise en route de l'ordinateur, le DOS est en position verify (off).
VERIFY permet de verifier par une relecture immédiate si toutes les informations écrites sur une disquette sont exactes; cependant le temps d'accès à la disquette est doublé (écriture + vérification).

14) TRACE

Cette commande permet l'affichage sur l'écran du "PC" ('program conter' ou compteur de programme, un des registres du Z80). Pour mettre en marche, tapez TRACE enter, pour l'arrêter tapez TRACE (OFF)enter. Malheureusement, le Z80 travaillant très vite (à 1,6 MHz), l'affichage du "PC" est très dificille à suivre ! (il change toutes les 8 millisecondes environ).

15) DEBUG

Cette commande permet la mise en oeuvre d'un programme d'aide à la mise au point ('debugging') en temps réel des programmes écrits en langage machine. Il permet à l'utilisateur d'examiner ou de modifier le contenu des registres du Z80 ou de cases mémoire RAM, de sauter ('jump') à une adresse donnée et de commencer à exécuter avec des points d'arrêts optionnels, d'aller pas à pas dans un programme (ou une instruction CALL).

Toutes les adresses et valeurs d'octets dans DEBUG sont données ou entrées en héxadécimal, qui est la seule forme comprise par ce programme.

Pour pouvoir avoir accès à DEBUG il faut tapez : DEBUG enter. Mais le programme n'est éxécuté que si : 1) vous appuyez sur la touche BREAK 2) après le chargement d'un programme et avant l'exécution de la première instruction 3) après la détection d'une erreur relative aux disques.

Notes : Les programmes appartenant au système du DOS ou les programmes d'exécution seulement ne peuvent pas être altérés par DEBUG.

DEBUG offre deux possibilités d'affichage :

* affichage des registres avec RAM indirecte plus une 'page' de 64 octects de RAM

* affichage d'une 'page' de 256 octects de RAM

Dans l'affichage des registres, DEBUG affiche tous les registres du Z80 par paires (AF,BC,DE,...).

Les registres F et F' sont affichés suivis d'une représentation en ASCII de chaque bit.

Pour voir ceci tapez : DEBUG enter puis break

Les bits des registres F et F' sont affichés de la façon suivante :

bit:	s'il est mis:	sinon:
7 signe	S	-
6 zéro	Z	-
5 inutilisé	1	-
4 demi-retenue	H	-
3 inutilisé	1	-
2 parité/dépassement	P	-
1 négatif	N	-
0 retenue	C	-

Commandes du DEBUG :

A : montre les caractères ASCII ou graphiques correspondant à chaque valeur affichée. Si cette valeur ne correspond à aucun caractère, un point (.) est affiché.

C : exécution du prochain pas de programme, les CALL étant exécutés en entier (l'instruction suivante étant définie par la valeur de PC). Le programme cible ne peut être un programme du système ou un programme à exécution seulement.

Daaaa <espace> : affichage de la mémoire à partir de l'adresse aaaa. dans le mode 1 'page' sur l'écran, affichage de la mémoire de façon à contenir l'adresse aaaa.

Gaaaa [,bbbb [,cccc]] enter : placer aaaa dans le PC et executer avec les points d'arrêt optionnels à bbbb,cccc.

H : affichage de tous les registres et mémoires en héxadécimal.

I : exécution du pas suivant (même reserves que C)

M[aaaa]<espace> : commencer la procédure de modification à l'adresse aaaa. Le dialogue de modification est placé en bas à gauche de l'écran. Si aaaa n'est pas précisé, la dernière adresse de modification est prise pour aaaa. Si aaaa est présent sur l'affichage de l'écran, sa valeur est encadrée par des barres verticales.

R rp␣dddd<espace> : charger la valeur dddd dans la paire de registres rp.

S : affichage d'une page complète de mémoire

U : affichage dynamique.(appuyez sur une touche pendant 2 secondes pour l'arrêter).

X : affichage au format registres. Permet aussi d'abandonner les autres commandes à l'exception de R.

; : passage à la page suivante

- : passage à la page précédente.

Note : Vous ne pouvez pas utiliser la touche ← pour rectifier une erreur de frappe, mais vous devez utiliser la touche X qui permet d'abandonner une commande. Si vous avez fait une erreur en entrant une valeur ou une adresse, corriger seulement après l'erreur, car DEBUG ne regarde que les 4 derniers caractères entrés :

D474080⟨espace⟩ est identique à D4080⟨espace⟩ (affichage de la mémoire contenant l'adresse 4080).

Pour retourner au DOS sans reinitialisation, tapez G402D enter, avec reinitialisation, tapez : G0000 enter. Pour stopper le DEBUG, tapez (quand vous êtes en DOS READY) :

DEBUG␣(OFF) enter

Note : DEBUG est aussi accessible en BASIC (voir CMD"D").

16) BASIC-BASIC2

La commande BASIC2 permet le retour direct au basic level II Tapez simplement BASIC2 enter et vous verrez apparaitre : MEMORY SIZE ?.

En level II si vous voulez revenir au DOS, tapez :

SYSTEM enter , à la réponse *? de l'ordinateur, tapez : /Ø
le DOS se chargera .

La commande Basic permet de passer au DISK BASIC, beaucoup plus performant que le BASIC LEVEL II et qui est décrit dans les deuxieme et troisième parties.

Cette commande peut prendre de très nombreuse forme :

BASIC enter : passage au basic-disque.

BASIC␣* enter : passage au basic-disque avec restitution du programme basic contenu dans la mémoire de l'ordinateur. Cette commande, si elle est exécutable est obligatoirement suivie d'un LIST ; si elle n'est pas exécutable elle est rejetée et le DOS est rechargé (pour passer en Basic tapez alors BASIC enter).

BASIC␣N enter : passage au basic-disque avec reservation de N (0 à 15) zones d'entrées-sorties (voir la quatrième partie). Si N n'est pas donné, il est pris égal à 3.

BASIC␣M enter : passage au basic-disque avec protection de la zone mémoire suivant M (analogue au 'Memory Size' du basic level II). Si M n'est pas donné, toute la mémoire disponible est allouée au basic.

BASIC␣COMMANDE enter : passage au basic-disque puis exécution de la commande basic donnée (voir la partie 2 pour les commandes basic).

On peut melanger dans la même commande plusieurs spécifications :

BASIC␣N,M,CMD enter, BASIC␣M,N,CMD enter , BASIC␣N,M enter, BASIC␣M,N enter, BASIC␣N,CMS enter , BASIC␣M,CMD enter

Du cas où une erreur quelconque se produit avec l'une de ces commandes, il y a un retour automatique au DOS READY.

Note : la commande BASIC␣* ne fonctionne pas après la commande DIR , mais fonctionne après les commandes : TRACE;CLOCK,

(la seconde partie traite des commandes du BASIC, la troisième partie des instructions du disk-basic et la quatrième partie des fichiers basic.).

2ieme PARTIE : COMMANDES DU BASIC

Après être passé au disk-basic grâce à la commande BASIC vous pouvez exécuter les commandes suivantes :

1) SAVE

Cette commande permet de sauver des programmes basic sur disquette. Deux cas peuvent se produire.

Vous venez de taper le programme, pour le placer sur disquette, tapez :

SAVE expression$ enter ou SAVE expression$, A enter.

Exemple : SAVE"JEU1/BAS" : le programme est alors stocké en format condensé qui tient moins de place, mais ne peut pas être LISTé ou PRINTé ni servir à étendre un autre programme (voir la commande MERGE ou APPEND.

"JEU2/TXT",A : le programme est stocké en format ASCII, plus long mais qui permet d'utiliser les commandes LIST,PRINT, MERGE, APPEND,...

Les extensions employées sont en général : /BAS pour un programme basic en format condensé et /TXT pour un programme Basic en format ASCII.

Note : Pour pouvoir utiliser la commande SAVE, la disquette ne doit pas être protégée à la lecture (ôter la protection).

Votre programme basic est sur une cassette. Pour le placer sur disquette, tapez :

CMD"T" enter puis à la réponse READY> , tapez CLOAD enter les opérations se déroulant alors comme en basic level II. Dès que votre programmes est chargé (vérifiez qu'il est correct grâce à un LIST) tapez SAVE"...." puis CMD"R" enter. Le programme est alors sur la disquette.

2) LOAD et RUN

La commande LOAD permet de charger un programme BASIC en mémoire. Elle est analogue à la commande CLOAD du BASIC level II. Cette commande s'utilise de la façon suivante :

LOAD expres$ où expres$ est le nom d'un programme.

Exemple : pour charger en mémoire le programme JEU1/BAS tapez : LOAD"JEU1/BAS" enter.

La commande LOAD efface tous les programmes BASIC contenus en mémoire, met à zéro toutes les variables et ferme tous les fichiers ouverts.

Vous pouvez aussi utiliser : LOAD"JEU1/BAS",R enter ; le programme basic JEU1/BAS est alors chargé en mémoire puis exécuté automatiquement. Cette commande ne ferme pas les fichiers ouverts.

Si vous utilisez la commande LOAD pour charger un programme qui n'est pas écrit en BASIC vous obtenez le message . 'DIRECT STATEMENT IN FILE' (instruction direct dans le fichier) ou le message 'LOAD FORMAT ERROR' (erreur de format au chargement).

La commande RUN permet de charger et d'exécuter un programme BASIC stocké sur disque.

Cette commande présente deux formes :

RUN"...." ou RUN".....",R

RUN"JEU/BAS" charge et execute le programme JEU/BAS, en fermant au préalable tous les fichiers ouverts RUN"JEU/BAS",R charge et execute le programme JEU/BAS, mais en laissant ouverts les fichiers.

Les commandes LOAD"JEU/BAS",R et RUN"JEU/BAS",R sont totalement identiques.

Les commandes LOAD et RUN peuvent aussi être utilisées comme instruction dans un programme écrit en basic. Ceci permet d'avoir un programme maitre appelant au fur et à mesure des besoins divers programmes qui retournent au programme maitre.

Exemple : le premier programme nommé PROG1/BAS est :

```
10 PRINT"EXECUTION DU PROGRAMME 1"
20 RUN"PROG2/BAS
```

Le second programme nommé PROG2/BAS est :

```
10 PRINT "EXECUTION DU PROGRAMME 2"
20 RUN : PROG1/BAS
```

Tapez alors : RUN"PROG1/BAS" et observez le résultat. Pour stopper l'exécution appuyer sur break.

3) KILL

Cette commande du DOS est aussi valable en basic. Elle permet de détruire un fichier. Le nom du fichier doit seulement être mis entre guillemets.

Exemple : vous voulez détruire le programme JEU3/BAS.SECRET
tapez alors :
KILL"JEU3/BAS.SECRET" enter.

Ne détruisez jamais un fichier ouvert (voir partie n°4) sous peine de voir le contenu de la disquette entier détruit. (tapez d'abord : CLOSE enter si vous n'êtes pas sûr que tous les fichiers sont fermés).

4) MERGE

Cette commande est pratiquement identique à la commande DOS APPEND. Elle permet de mélanger deux programmes écrits en BASIC, l'un dans la mémoire de l'ordinateur , l'autre sauvé sur disque avec l'option A (c'est-à-dire sauvé en format ASCII).

Exemple : Dans la mémoire RAM, vous avez le programme :

```
10 REM... Programme principal
20 GOSUB 1000
30 REM ... RESTE DU PROGRAMME
999 END
1000 REM ...VOUS DEVEZ AJOUTER LA SOUS ROUTINE ICI
1010 REM ...AUSSI UTILISEZ LA COMMANDE MERGE
1020 PRINT "LA SOUS ROUTINE N'EST PAS VALABLE" : RETURN
```

Si nous supposons que vous avez sauvé en format ASCII le programme suivant, nommé SOUSROUT/TXT

```
1000 REM .... DEBUT DE LA SOUS ROUTINE
1010 PRINT "EXECUTION DE LA SOUS ROUTINE"
1020 REM .... RESTE DE LA SOUS ROUTINE ICI
1100 RETURN
```

Alors la commande MERGE "SOUSROUT/TXT" donnera le resultat suivant, en RAM :

```
10 REM .... PROGRAMME PRINCIPAL
20 GOSUB 1000
30 REM .... RESTE DU PROGRAMME
999 END
1000 REM .... DEBUT DE LA SOUS ROUTINE
1010 PRINT "EXECUTION DE LA SOUS ROUTINE"
1020 REM .... RESTE DE LA SOUS ROUTINE ICI
1100 RETURN
```

Ainsi que vous le voyez les numéros de lignes du second programme n'ont pas besoin de suivre celle du premier programme. Les lignes sont simplement remplacées. Cette commande permet de stocker sur disques de nombreuses sous-routines et de les incorporer dans un programme lorsque l'on en a besoin sans avoir à les retaper. (Particulierement interessant pour les calculs faisant intervenir les fonctions trigonométriques en double précision).

5) RENUM

Cette commande permet de vérifier que la numérotation des lignes d'un programme basic ne comporte pas d'erreur et de renuméroter automatiquement ces lignes.

I) RENUM␣U

Cette commande permet de rechercher dans un programme BASIC contenu dans la mémoire, les erreurs liées à la numérotation des lignes ainsi que quelques autres erreurs ; les erreurs sont indiquées de la façon suivante :

SSSSS/U : la ligne numéro SSSSS n'existe pas

SSSSS/X : erreur de syntaxe à la ligne numéro SSSSS (mais toutes les erreurs de syntaxe ne sont pas détectées).

SSSSS/S : erreur de numérotation à la ligne SSSSS

Exemple : dans la mémoire, se trouve le programme suivant :

```
10 PRINT
20 GOTO 100
30 GOTO 40
35 GOSUB 3456789
50 END
```

La commande RENUM␣U enter donne :

```
ERROR LINES
100/U  40/U  35/S DONE
```

ce qui signifie :

Les lignes numéros 100 et 40 n'existent pas

A la ligne 35, il y a une erreur de numérotation (numéro trop grand).

II) RENUM␣SSSSS,IIIII,PPPPP,QQQQQ

Cette commande permet de renuméroter les lignes d'un programme BASIC situé dans la mémoire de l'ordinateur. Elle renumérote les lignes dont les numéros sont compris entre PPPPP et QQQQQ, en prenant SSSSS comme nouveau premier numéro et IIIII comme intervalle entre les numéros. SSSSS et IIIII doivent être compris entre 1 et 65529 ; la valeur par défaut est 10. PPPPP doit être compris entre 1 et 65529 et être inferieur ou égal à SSSSS ; la valeur par défaut est 0 (le début de programme). QQQQQ doit être compris entre 1 et 65529 et être superieur ou égal à SSSSS ; la valeur par défaut est 65529 (la fin du programme).

Pour utiliser les valeurs par défaut, tapez les commandes :

RENUM␣, RENUM␣SSSSS RENUM␣,IIIII
RENUM␣,,PPPPP RENUM␣,,,QQQQQ RENUM␣SSSSS,IIIII
RENUM␣SSSSS,,PPPPP RENUM␣IIIII,PPPPP RENUM␣SSSSS,,,QQQQQ
RENUM␣IIIII,QQQQQ RENUM␣,,PPPPP,QQQQQ RENUM␣SSSSS,IIIII,PPPPP
RENUM␣SSSSS,IIIII,,QQQQQ RENUM␣SSSSS,,PPPPP,QQQQQ RENUM␣IIIII,PPPPP,QQQQQ

Si une erreur est détectéedans le programme avant le changement du texte du programme, la commande passe automatiquement en 'RENUM␣U' et affiche les erreurs qu'il trouve .

Si une erreur est détectéeaprès que la commencement du changement du texte alors le message 'FATAL ERROR. TEXT NOW BAD' (erreur fatale. Le texte est maintenant mauvais) est affiché et l'ordinateur revient au DOS READY ou en DEBUG (si celui-ci est activé). Le programme Basic ne peut être retrouvé (même en utilisant la commande Basic␣*).

Exemples : pour renuméroter toutes les lignes d'un programme en partant de 10 avec un pas de 5 tapez : RENUM␣10,5 enter;

en partant de 100 avec un pas de 20, tapez :
RENUM␣100,20 enter

Pour renuméroter les lignes 11000 à 17000 d'un programme Basic, en partant de 10000 avec un pas de 100 tapez :

RENUM␣10000,100,11000,17000 enter.

Pour renuméroter les lignes 5000 à 15000 d'un programme, en partant de 10 avec un pas de 10, tapez :

RENUM␣,,5000,15000 enter.

Pour renumeroter les lignes 7000 à la fin d'un programme, en partant de 10 avec un pas de 5, tapez :

RENUM␣,5,7000 etc....

Avant d'utiliser la commande RENUM, sauvez d'abord le programme par SAVE au cas où il y aurait des erreurs fatales.

6) REF

Cette commande permet de trouver les réferences d'un programme Basic situé en mémoire de l'ordinateur.

I) REF *

affiche la liste complète de toutes les références.

II) REF $

imprime la liste complète de toutes les références.

III) REF NN

affiche la liste des références à la variable NN ; si NN n'a qu'un seul caractère, un blanc y est ajouté comme second caractère ; NN ne peut avoir plus de deux caractères et ne doit pas avoir de type (!,≠,$...)

IV) REF SSSSS

affiche la liste des références à l'entier SSSSS formé de 1 à 5 chiffres décimaux (de 0 à 999 9) les références hexadécimales ou octales ne sont pas listées.

V) REF*NN

affiche la liste des références à partir de la variable NN.

VI) REF$NN

imprime la liste des références à partir de la variable NN.

VII) REF*SSSSS

affiche la liste des références à partir de l'entier SSSSS

VIII) REF $SSSSS

imprime la liste des références à partir de l'entier SSSSS

IX) REF

affiche la prochaine ligne du texte contenant au moins une référence à la variable ou à l'entier spécifié dans la dernière commande REF NN ou REF SSSSS exécutée ; s'il n'y a plus d'autre références le message 'TEXT END' (fin du texte) est affiché.

Pour faire une pause dans l'affichage , appuyez sur BREAK ; pour continuer, appuyez sur enter.

Exemple : en mémoire centrale se trouve le programme :

```
5 A$ = "PGCD"
10 PRINT A$
2Ø PRINT
3Ø PRINT "ENTREZ 0,0 POUR STOPPER LE PROGRAMME"
40 INPUT "ENTREZ DEUX NOMBRES";A,B
50 IF A<>0 THEN 90
60 IF B<>0 THEN 90
80 GOTO 190
90 A=ABS(A): B=ABS(B)
110 R= A-B * INT (A/B)
120 IF R=0 THEN 160
130 A=B : B=R
140 GOTO 110
160 PRINT A$ ;" = " ; B
180 GOTO 40
190 END
```

Si vous tapez : REF* enter vous obtenez :

```
  0      50    60    120
 40      180
 90      50    60
110      150
160      120
190      80
  A      5/$   10/$   40   50   90/2   110/2   130   160/$
  B      40    60    110/2   130/2    160
  R      110   120   130
```

Par exemple : 40 apparait une fois à la ligne 180 ; 90 apparait une fois à la ligne 50 et une fois à la ligne 60. A apparait aux lignes 5,10 et 160 sous la forme A$, apparait une fois aux lignes 40,50 et 130 et deux fois aux lignes 90 et 110.

Si vous tapez : REF A enter, vous obtenez :

A S/$ 10/$ 40 50 90/2 110/2 130 160/$

Si vous tapez : REF* A enter, vous obtenez :

A 5/$ 10/$ 40 50 90/2 110/2 130 160/$
B 40 60 90/2 110/2 130/2 160
R 110 120 130

Cette commande peut être très utile dans un très long programme pour savoir quelles sont les variables déja utilisées ou dans un programme quelconque pour changer le nom d'une variable, ou le numéro d'une ligne.

7) LIST - EDIT - DELETE

Le Basic - disque permet de remplacer les commandes LIST, EDIT et DELETE par L,E et D lorsqu'elles sont suivies d'un point ou d'un chiffre décimal. L, E et D doivent être le premier caractère d'un ligne qui ne doit pas contenir de signe '=' -

Exemple : pour lister la ligne 120, tapez L120 enter.

Les abréviations suivantes sont aussi acceptées :

[.] : lister la ligne de texte courrante (remplace LIST. du basic level II).

[E.] : éditer la ligne de texte courrante (remplace EDIT. du Basic level II).

 : lister la ligne de texte suivante ; s'il n'y a pas de ligne suivante, la dernière ligne est répétée

 : lister la ligne de texte précédente ; s'il n'y a pas de ligne précédente, la première ligne est repétée.

Shift  : lister la derniere ligne du programme.

Shift  : lister la première ligne du programme.

3eme PARTIE : INSTRUCTIONS BASIC

Cette partie va décrire les instructions Basic que le DOS permet d'utiliser en plus des instructions basic Level II.

1) Constantes hexadécimales et octales

Le Basic-Disque permet l'utilisation dans un programme de constantes écrites en hexadécimal (base 16) ou en octal (base 8). Il suffit pour cela de faire précéder la constante par &H pour les constantes hexadécimales et par &O ou & pour les constantes octales.

Les constantes hexadécimales ont donc la forme &Hdddd, où dddd représentent 4 chiffres hexadécimaux (0,1,2,...,9,A,...,F).

Les constantes octales ont la forme &Odddddd ou &dddddd, ou dddddd représentent 6 chiffres octaux (0,1,2,...,6,7) dont la valeur doit être inférieur à 1777777.

Ces constantes représentent toujours des entiers signés :

&H1 = 1, &H2 = 2,..., &H7FFF = 32767, &H8000 = -32768,...,
&HFFFF = -1 &1 = 1, &2 = 2,... &77777 = 32767,& I0000 = -32768,...,
& 177777 = -1

Ces constantes sont particulièrement intéressantes pour programmer une routine en langage machine à partir d'un programme basic ; en utilisant les constantes hexadécimales, il n'y a plus besoin de traduire le programme assemblé en décimal. Mais malheureusement, ces constantes ne peuvent pas figurer dans les instructions DATA, ni être données en réponse à une question posée par un programme.

Souvent, pour éviter des erreurs de syntaxe, il faut mettre ces constantes entre parenthèses.

Exemples : POKE &H3COO,42
```
100 FOR I= (&H3COO)TO(&H3FFF) STEP(&H40)
200 IF A=(&H37E8)THEN A=A+1
300 POKE A%, (X% AND &HFF)...
```

2) CMD"D"

Cette instruction permet d'utiliser le programme DEBUG (déja décrit dans la 1ere partie) à partir d'un programme BASIC. Le programme BASIC n'est en aucune façon affecté par le programme DEBUG car celui-ci se charge avant le Basic Disque.

Pour revenir au Basic, a partir de DEBUG, sans reinialisation, il faut taper : G enter et le message habituel READY apparait. Le programme basic est toujours en mémoire.

Pour revenir au Basic avec reinitialisation, tapez : G5200 enter ; vous perdez alors les programmes Basic ou la valeur des variables.

Lorsque vous avez exécuté la commande CMD"D", chaque fois que vous appuyez sur la touche break vous entrez dans le programme DEBUG. Tapez G enter pour revenir au Basic et CONT enter

4) CMD "R" et CMD "T"

La commande CMD "T" permet de stopper l'horloge et d'interdire toutes interruptions . La commande CMD "R" permet de remettre en route l'horloge et de permettre à nouveau les interruptions.

(Les interruptions sont : horloge en temps réel ; affichage de l'heure ; TRACE;...)

Les commandes suivantes doivent être précédées de l'instruction CMD "T" :

CLOAD, CLOAD?, CSAVE, SYSTEM, PRINT #-1, PRINT #-2, INPUT #-1, INPUT #-2

(c'est à dire toutes les commandes commandant l'utilisation du magnétophone).

Lorsque ces commandes sont effectuées, alors utilisez l'instruction CMD "R".

Pour stopper l'entrée dans le programme DEBUG après avoir tapé la commande CMD "D", vous pouvez utiliser la commande CMD "T".

5) DEF FN

Cette instruction permet de définir des fonctions à plusieurs arguments et de les utiliser de la même façon que les fonctions déjà existantes telles que SIN, COS,....,STRING$,...

La syntaxe est la suivante : DEFFN var1 (var2,var,...) = expression où var 1 est le nom donné à la fonction, var 2,var,... est la liste des variables intervenant dans le calcul de la fonction donné par expression. var1,var2,.... suivent les règles des noms de variables (2 caratères dont le premier est obligatoirement une lettre, suivi d'un type de variable facultatif).

Par exemples :

DEFFNA(B,C,D)= SQR(B*B + C*C +D*D) définit la fonction FNA que vous pouvez alors utiliser avec n'importe quelles variables de même type que B,C, et D :

Z=FNA(K,P,3) ou A5=FNA(A1,A2,7.5) ou P=FNA(3.2,5.1,2.9) etc...

Les noms que vous donnez aux variables situées dans la liste d'arguments de la fonction n'ont aucune importance et n'empèchent pas d'utiliser ces noms dans le programme. Ainsi :

DEFFNZ(A,B)=A+B et DEFFNZ(K,X)=K+X définissent exactement la même fonction et les variables A,B et K,X peuvent être reutilisées dans le programme sans aucun inconvenient.

Exemples :

Définir deux fonctions trouvant respectivement le plus grand et le plus petit des deux nombres :

```
DEF FNMIN (A,B)= (A+B - ABS(A-B))/2
DEF FNMAX (A,B)= (A+B + ABS(A-B))/2
```

Ainsi pour trouver le plus grand et le plus petit des 3 nombres X,Y et Z, on pourra écrire :

```
PG =FNMAX (FNMAX(X,Y),Z) ou PG =FNMAX(X,FNMAX(Y,Z))
PP =FNMIN (FNMIN(X,Y),Z) ou PP =FNMIN(X,FNMIN(Y,Z))
```

(PG est le plus grand et PP le plus petit des nombres X,Y et Z).

DEFFNRA(A,B)=A + INT(B*RND(0)) permet grâce à l'appel de la fonction FNRA de retourner un nombre aléatoire compris entre A et B.

ainsi Z=FNRA(10,20) permet de placer dans Z un nombre aléatoire compris entre 10 et 20.

DEF FNA$(B$,C$,L7)=LEFT$(B$,L7)+" "+RIGHT$(C$,L7+1)
est un exemple de fonction utilisant des variables caractères.
en choisissant R1$="ABCDEFGH" , R2$="IJKLMNOPQR";

T$=FNA$(R1$,R2$,5); alors T$="ABCDE MNOPQR".

Le nombre d'arguments dans la liste n'est pas limité, mais rappelez-vous que l'instruction de définition doit tenir en 256 caractères au maximum.

6) INSTR

Cette instruction permet de rechercher si une chaine de caractères est présente à l'interieur d'une autre chaine de caractères ; si la réponse est oui, INSTR retourne la position de départ, sinon elle retourne zéro.

```
Exemples  : si A$="ABCDEFG"
            Z =INSTR(A$,"BCD")  donnera Z=2
            Y =INSTR(A$,"BXY")  donnera Y=0
            X =INSTR(A$,"ABCDEFGH") donnera X=0
```

vous pouvez aussi imposer la position du début de recherche:

T = INSTR("123123123","12") donnera T = 1
T = INSTR(2,"123123123","12") donnera T = 4

indique que l'on cherche 12 à partir de 2eme caractère.

T = INSTR(5," 123123123","12") donnera T= 7

7) LINE INPUT

L'instruction LINE INPUT est pratiquement identique à l'instruction INPUT. Les différences entre les deux instructions sont les suivantes : l'instruction LINE INPUT ne peut être utilisée que pour les variables chaînes de caractères ; lorsque cette instruction est executée, l'ordinateur n'affiche pas de point d'interrogation ; une instruction LINE INPUT ne peut assigner une valeur qu'à une seule variable (ainsi LINE INPUT A$ est autorisée, mais LINE INPUT A$,B$ est interdit ; l'ordinateur vous donnera alors une erreur de syntaxe.) tous les caractères même les virgules et les guillemets sont autorisés dans les chaines de caractères ; les caractères blancs de tête ne sont pas ignorés, mais feront partie de la variable ; la seule façon de terminer l'entrée d'une chaine de caractères est d'appuyer sur enter.

Il est aussi possible de faire afficher une chaine de caractères avant une entrée :

LINE INPUT "NOM,PRENOM"; A$

Exemple d'utilisation :

```
10 LINE INPUT A$
20 PRINT : PRINT A$ : GOTO 10
RUN enter
```

l'ordinateur affiche seulement : -

Vous pouvez alors tapez ce que vous voulez y compris les caractères tels que : , ; " ' : . les passages à la ligne (touche [↓]) et les tabulations (touche [→]). L'ordinateur affichera alors tout ce que vous avez écrit.

Si lors de l'utilisation des instructions LINE INPUT, l'ordinateur vous affiche le message 'OUT OF STRING SPACE' (dépassement de l'espace alloué aux chaines de caractères) ajoutez simplement à votre programme une instruction CLEAR n.

8) MID$

L'instruction MID$ permet de remplacer une portion de chaine de caractères par une autre chaine de caractèresde même longueur. L'utilisation de cette instruction est la suivante :

MID$ (var $, n1,n2)=exp$

où var$ est le nom de la variable contenant la chaine de caractères ; exp$ est une chaine de caractères; n1 donne la position de départ pour le remplacement; n2 donne le nombre de caractères à remplacer dans var$ (si n2 n'est pas donné, l'ordianteur prendra le plus petit des deux nombres : LEN(exp $) et LEN(VAR$) - n1+1).

La longueur de la chaine de caractères contenu dans var$ ne peut pas être changée par une instruction MID$; au cas où exp$ est trop longue, les caractères supplémentaires sont simplement ignorés; au contraire si vous indiquez que n2 caractères doivent être remplacés, alors que exp$ contient n3, avec n3 plus petit que n2, alors seuls n3 caractères seront remplacés.

Exemples :

```
A$ ="ABCDEFGHIJ"
MID$ (A$,3,4)="1234"  donnera A$ ="AB1234GHIJ"
MID$ (A$,1,6)="12345" donnera A$ ="12345FGHIJ"
MID$ (A$,5)="999"  donnera A$="ABCD999HIJ"
MID$ (A$,10)="12345"  donnera A$="ABCDEFGHI1"
MID$ (A$,5,2)="123456"  donnera A$="ABCD12GHIJ"
```

9) TIME$

Cette instruction basic complète les commandes DOS CLOCK,TIME, DATE. Elle permet d'accéderà l'horloge en temps réel à partir d'un programme BASIC. TIME$ ramène une chaine de caractères contenant la date et l'heure, sous la forme :

MM/JJ/AA␣HH:MM:SS

En conjonction avec les instructions LEFT$, RIGHT$ et MID$, la variable TIME$ permet d'avoir toutes les variables de temps.

Exemples :

```
A$ = TIME$
B$ = RIGHT$(A$,2)   donnera les secondes
C$ = MID$(A$,13,2)  donnera les minutes
D$ = MID$(A$,10,2)  donnera les heures ....
```

Pour savoir combien de temps réclame l'exécution d'un programme, on peut utiliser la variable TIME$:

```
10 A$ = TIME$
20 A$ = RIGHT$(A$,2) : B$ = MID$(A$,13,2)
30 A = VAL(A$) : C = VAL(B$)
40 FOR I = 0 TO 10000
50 NEXT I
```

```
60 A$ = TIME$
70 A$ = RIGHT$(A$,2) : B$ = MID$(A$,13,2)
80 B = VAL(A$) : D = VAL(B$)
90 TS = B-A : TM = D-C : IF TS<0 THENTS=TS+60:TM=TM-1
100 PRINT "TEMPS D'EXECUTION ="; TM; "MN";TS;"S"
110 END
> RUN
TEMPS D'EXECUTION = 1 MN 34 S
READY
```

10) DEFUSR et USRn

Le basic niveau II permet l'accès à un seul programme en langage machine à partir d'un programme BASIC par l'instruction USR. Le BASIC Disque permet l'accès à 10 programmes en langage machine grâce aux instructions DEFUSRn et USRn. (n étant un chiffre décimal 0,1,2,....,8,9).

L'instruction DEFUSRn définit l'adresse d'entrée du programme en langage machine dont l'accès est permis par USRn

Exemple : DEFUSR3=27100
DEFUSR5=&HFC00

Exemple de programme (il multiplie un entier N par 2 et par 4)

```
10 DEF USR5  = &H7D00
20 DEF USR7  = &H7E00
30 FORX = 32000 TO 32006
40 READ A
50 POKE X,A
60 NEXT X
70 FOR X= 32256 TO 32263
80 READ A
```

```
90 POKE X,A
100 NEXT X
110 CLS : PRINT "PROGRAMME DE DEMONSTRATION"
120 PRINT : INPUT "DONNEZ UN ENTIER (-8000 < N < 8000)"; N
130 PRINT N; "X 2 ="; USR5(N); TAB(25); N; "X 4 =" ; USR7(N)
140 GOTO 120
150 DATA 205,127,10,41,195,154,10
160 DATA 205,127,10,41,41,195,154,10
RUN
PROGRAMME DE DEMONSTRATION

DONNEZ UN ENTIER (-8000 < N < 8000) ? 50
  50 X 2 = 100        50 X 4 = 200

DONNEZ UN ENTIER (-8000 < N < 8000) ? - 112
  - 112 X 2 = -224     -112 X 4 = -448

DONNEZ UN ENTIER (-8000 < N < 8000) ? break
BREAK IN 120
READY
```

Les programmes en langage machine peuvent aussi être crées par DEBUG, sauvés sur disques par DUMP, chargés en mémoire par LOAD ;

L'argument de USRn(A), l'entier A, se trouve à l'adresse OA7F (appelée par un CALL) .

Si une valeur est retournée par la routine machine on peut l'avoir en OA9A. Ainsi un sous programme en langage machine doit commencer par CALL OA7FH s'il y a un argument à prendre et se terminer par JP OA9AH s'il y a une valeur à retourner.

Dans les deux cas le nombre doit se trouver dans le registre HL.

Voici le listing de deux sous programmes de l'exemple :

```
USR5 :      CALL   OA7FH
            ADD    HL,HL
            JP     OA9AH

USR7 :      CALL   OA7FH
            ADD    HL,HL
            ADD    HL,HL
            JP     OA9AH
```

Si aucun argument n'est retourné par la sous routine, terminez le programme par RET

Exemple : programme blanchissant l'écran :

```
10 DATA 33,0,60,17,1,60,1,255,3,54,191,237,176,201
20 DEF USR2 = &H7D00
30 FOR I = 32000 TO 32013
40 READ A : POKE I,A : NEXT I
50 X=USR2(X)
60 GOTO 60
```

Tapez RUN et observez ! (pour stoppez le programme appuyez sur break).

Listing de sous programme machine :

```
LD  HL, 3CØØH
LD  DE, 3CØ1H
LD  B , 3FFH
LD (HL) ØBFH
LDIR
RET
```

4eme PARTIE : LES FICHIERS DE DONNEES

La gestion des fichiers de données est l'une des plus intéressantes utilisations des disques magnétiques. Un fichier de données est un fichier contenant des données (c'est-à dire des nombres ou des chaînes de caractères) servant d'informations (en général non permanentes) à un ou plusieurs programmes (écrits en BASIC le plus souvent). Il y a deux types de fichiers : les fichiers à accès séquentiel et les fichiers à accès aléatoire. Dans les fichiers à accès séquentiel, les données sont enregistrées en séquences, de la même façon que les fichiers sur cassettes ; le programmeur ne s'occupe pas de la façon dont sont enregistrées les données (ou très peu) ; les données ne peuvent être relues que dans l'ordre ou elles ont été enregistrées (ainsi pour accéder à la 7e donnée, il faut d'abord lire les 6 prémieres puis la 7e). Dans les fichiers à accès aléatoire, les données peuvent être écrites et lues dans n'importe quel ordre (ainsi on peut très bien commencer à écrire la 3e donnée, les deux premières restant vides !); le programmeur doit choisir lui-même comment seront enregistrées les données sur le disque.

Pour les deux types de fichiers, il y a quelques instructions communes. D'abord pour pouvoir lire ou écrire dans un fichier il faut que celui-ci soit ouvert, ce qui se fait par l'instruction OPEN. La transmission des données de la disquette à la mémoire ou de la mémoire à la disquette n'est pas directe, mais passe par une zone d'entrée-sortie, appelée 'Buffer' ; lorsque vous ouvrez un fichier par l'instruction OPEN,vous lui attribuez en même temps un buffer parmi les 15 disponibles (en passant en BASIC par BASIC [enter], vous n'avez que 3 buffers seulement ; si vous devez en utiliser plus de 3,7 par exemple, alors tapez BASIC␣7[enter]). Un buffer ne peut être attribué qu'à un seul fichier, mais en revanche, on peut attribuer plusieurs buffers au même fichier (utilisé assez rarement). Lorsque l'utilisation du fichier est terminée,il faut le fermer ainsi que le buffer associé, afin de protéger le contenu de ce fichier (et même de la disquette entière) contre une des-

Cela peut se faire avec l'instruction CLOSE. Vous avez aussi la possibilité de visualiser facilement le contenu d'un fichier car il est écrit en format ASCII. Il suffit d'utiliser les commandes du DOS : LIST ou PRINT (voir 1er partie).

1°) Les fichiers à accès séquentiel :

L'utilisation de ce mode d'accès se fait grâce aux instructions BASIC : OPEN (ouverture du fichier), CLOSE (fermeture du fichier), PRINT #n (écriture séquentielle), INPUT #n ou LINEINPUT #n (lecture séquentielle).

Un fichier de données, comme tous les fichiers, doit avoir un nom respectant le format imposé par le DOS (voir l'introduction).

Si vous désirez ouvrir un fichier nommé DONNEES/TXT (l'extension pour un fichier de données généralement /TXT), en lui attribuant le buffer n° 1, pour pouvoir y écrire des données, il faut utiliser l'instruction :

```
OPEN "O",1,"DONNEES/TXT"
```
(le O signifie 'output' ou sortie).

Dans le cas où le fichier DONNES/TXT existe déjà, cette instruction détruit son contenu pour pouvoir écrire au début du fichier. Si vous voulez garder les données contenues dans le fichier et ajouter de nouvelle données, il faut utiliser l'instruction.

```
OPEN"E",1,"DONNEES/TXT"
```
(le E signifie étendre)

Si le fichier DONNEES/TXT n'existe pas,OPEN"E",...et OPEN"O",.. agissent exactement de la même façon, en le créant, et en positionnant le pointeur au début du fichier.

Lorsque le fichier DONNES/TXT existe et contient des données pour pouvoir lire celles-ci il faut utiliser l'instruction.

OPEN"I",1,"DONNEES/TXT".
(le I signifie 'Input" ou entrée).

Pour écrire des données dans le fichier 'DONNEES/TXT , que l'on vient de créer, on utilise l'instruction PRINT#n (n est le numéro du buffer attribué au fichier). Cette instruction produit des enregistrements sur disquette exactement semblables à ceux affichés par l'instruction PRINT du basic level II.

Les données sont numériques :
A=12.05 B=0.91 C=710.442

PRINT#1,A;B;C donne l'enregistrement : ␣12.05␣␣.91␣␣710.442␣ <CR>
(␣ représente un caractère blanc) (<CR> représente le caractère 'retour_chariot' ou enter).

PRINT#1,A,B,C donne l'enregistrement :␣12.05␣␣␣␣␣␣␣␣.91␣␣␣␣␣␣␣␣
710.442␣ <CR>
qui comporte beaucoup de blancs et perd inutilement de la place

On peut aussi utiliser PRINT#1,USING"...",... mais il faut faire très attention au format proposé si l'on veut que l'enregistrement soit lisible !

PRINT#1,USING"###.#";A,B,C donne l'enregistrement :
␣12.1␣␣0.9710.4␣ <CR>

De même PRINT#1,USING"###.#";A,B,C donne l'enregistrement :

Qui ne sera pas relu correctement comme trois variables puisqu'il manque un séparateur.

et PRINT#1,USING"##.##";A,B,C donne l'enregistrement :
12.05ØØ0.91%710.44Ø <CR>

qui ne pourra pas être relu correctement, l'un des caractères n'étant pas un chiffre. (le % indique seulement que 710. 442 ne peut être écrit avec le format ##.##).

Les données sont des chaines de caractères :

Il faut faire très attention à la syntaxe, si l'on veut pouvoir relire correctement les enregistrements.

A$ = "ABCD" B$ = "EFGH" C$ = "IJKL"

PRINT#1,A$;B$;C$ donne l'enregistrement : ABCDEFGHIJKL <CR>
qui évidemment ne pourra pas être relu comme 3 variables, mais comme une seule variable (ce qui peut toutefois présenter un intérêt !).

PRINT#1,A$,B$,C$ donne l'enregistrement : ABCDØØØØØØØØØØØØEFGH
ØØØØØØØØØØØØIJKL <CR>qui ne pourra être relu comme 3 variables, mais comme une seule, puisqu'il n'y a pas de séparateurs.

Pour obtenir un enregistrement correct des 3 variables, comme 3 variables distinctes il faut utiliser :

PRINT#1,A$;",";B$;",";C$ qui donne l'enregistrement:
ABCD,EFGH,IJKL <CR>qui effectivement représente trois variables.

Si l'une des variables présente un caractère séparateur, il faut la mettre entre " "

Exemple : E$ = "ABCD,EF" F$ = "GHIJ"

PRINT#1,E$;",";FS donne l'enregistrement ABCD,EF,GHIJ <CR>
qui représente 3 variables, il faut donc utiliser

PRINT#1,CHR$(34);E$;CHR$(34);F$ qui donne l'enregistrement:
"ABCD,EF"GHIJ qui pourra bien être relu comme 2 variables.

Vous pouvez aussi utiliser comme pour les variables numériques, PRINT#n,USING"...", avec les mêmes précautions d'usage.

Pour lire des données dans un fichier, on utilise pour les données numériques : INPUT#n, et pour les données chaînes de caractères : INPUT#n ou LINE INPUT#n.

Les données sont numériques :

L'instruction INPUT#N permet de relire les variables enregistrées par une instruction PRINT#n (sous reserve que celles-ci aient été convenablement enregistrées).

Exemple : Voici un programme montrant l'utilisation de INPUT#n

```
10 OPEN "O",1 "DATA/TXT"
20 A = 12.05 : B =.91 : C = 710.442
30 PRINT # 1,A;B;C
40 CLOSE
```

Ce programme crée un fichier nommé DATA/TXT et y inscrit l'enregistrement :
␣12.05␣␣ .91␣␣710.442␣ <CR>

Pour relire les variables voici un programme :

```
10 OPEN "I",2, "DATA/TXT"
20 INPUT # 2,K,L,M
30 CLOSE
```

```
40 PRINT K;L;M
```

Ce programme lira la valeur des variables K,L et M dans le fichier DATA/TXT et les imprimera sur l'écran.

Si vous essayez de lire plus de variables qu'il n'y a de données dans un fichier (par exemple : 20 INPUT #2,K,L,M,N) vous obtiendrez le message d'erreur :

'INPUT PAST END IN 20' (les entrées passent la fin de l'enregistrement en 20).

Par contre vous pouvez toujours lire moins de données qu'il n'y en a d'enregistrées.

Pour les variables de chaînes de caractères , il y a deux instruction permettant de relire des données dans un fichier : INPUT #n et LINE INPUT #n (elles fonctionnent de la même façon que les instructions INPUT et LINE INPUT).

Le tableau suivant montre l'effet de INPUT #1,A$,B$ suivant l'enregistrement :

Enregistrement	Valeurs assignées :
1- ABCD,EFGH,IJKL (eof)	A$=ABCD B$=EFGH
2- ØØØ ABCD,EFGH (eof)	A$=ABCD B$=EFGH
3- "ABCD",EF"GHIJ (eof)	A$=ABCD,EF B$=GHIJ
4- ABCD"EF",GHIJ (eof)	A$=ABCD"EF" B$=GHIJ
5- ABCD <LF><CR> EFGH (eof)	A$=ABCD<LF><CR>EFGH B$=vide
(<LF>= line feed : passage à la ligne: ↓)	"erreur" input past end in...)

Note : (eof) signifie end of file ou fin de fichier).

L'instruction LINE INPUT ≠ 1, A$ lit tout ce que contient l'enregistrement depuis le premier caractère jusqu'à :

1) un caractère CR non précédé d'un caractère LF
2) la fin du fichier
3) la lecture du 255e caractère inclu

Tous les caractères rencontrés, gillemets, virgules, blancs, LF - CR sont inclus dans A$.
Vous pouvez ainsi lire le contenu d'un fichier de données ou d'un programme basic sauvé en format ASCII (option A) avec le programme :

```
5 CLEAR 5000
10 OPEN "I", 1,"....." : ' (remplacer les pointillés par le nom d'un
                           fichier).
20 IFEOF (1) THEN 60
30 LINE INPUT
40 PRINT A$
50 GOTO  20
60 CLOSE  : END
```

- La ligne 20 est là pour rechercher la fin du fichier. Si vous la supprimez vous obtenez le message INPUT PAST END IN 30

EOF(n) est une variable logique ne pouvant prendre que deux valeurs vraies ou fausses (-1 ou 0). Elle est très utile pour éviter les erreurs.

2°) Les fichiers à accès aléatoire :

L'utilisation de ce mode d'accès se fait grâce à de très nombreuses instructions basic : OPEN (ouverture du fichier), CLOSE (fermeture du fichier), FIELD (découpage du champ), GET (lecture d'un enregistrement), PUT (écriture d'un enregistrement), LSET et RSET (placement des données dans le champs du buffer), MKD$,MKI$ et MKS$ (conversion des données numériquesen chaînes de caractères), CVD, CVI et CVS (conversion des chaînes de caractères sous forme numérique), EOF (détecteur de fin de fichier), LOF (donne le nombre d'enregistrements du fichier).

L'ouverture d'un fichier nommé DONNEES/TXT en mode aléatoire se fait par l'instruction OPEN "R",1,"DONNEES/TXT (R signifie 'RANDOM' : aléatoire), qui est valable aussi bien en entrée qu'en sortie. La fermeture du fichier se fait par l'instruction CLOSE, de façon identique à celle des fichiers à accès séquentiel.

Dans le mode d'accès aléatoire vous travaillez directement sur les 255 octets du buffer (dont vous avez donné le numéro dans l'instruction OPEN), la longueur d'un enregistrement étant de 255 octets. L'instruction FIELD permet de découper les 255 octets du buffer en plusieurs variables (qui sont obligatoirement des variables de chaînes de caractères).

Exemple : FIELD 1, 255 AS A$ attribue les 255 octets à la variable A$

FIELD 1, 100 AS A$, 100 AS B$, 55 AS C$ attribue 100 octets aux variables A$ et B$ et 55 octets à la variables C$.

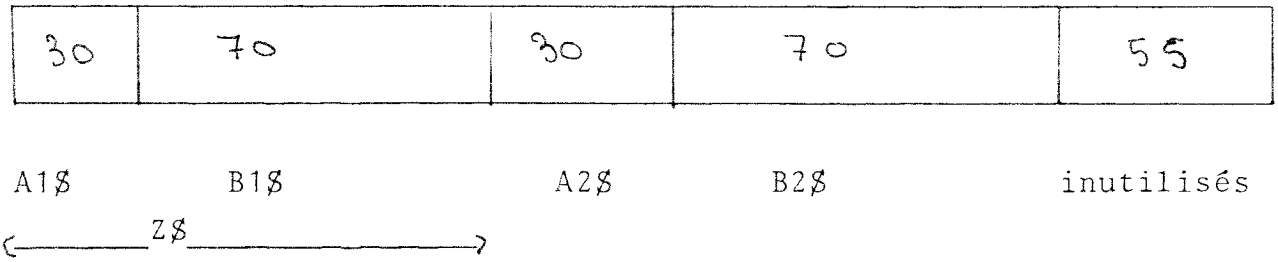
Vous pouvez utilisez l'instruction FIELD autant de fois que vous le désirez dans un même programme pour redécouper à votre convenance le champ du buffer, cette instruction n'effaçant pas le contenu du buffer.

Exemple : vous n'avez besoin que de 100 octets pour enregistrer deux variables A1$ et B1$ et de 100 octets pour A2$ et B2$ alors vous pouvez utiliser :

```
FIELD 1, 30 AS A1$, 70 AS B1$
.
.
.
.
FIELD 1, 100 AS Z$, 30 AS A2$, 70 AS B2$
```

Cela permet ainsi de placer deux séries de variables dans le même enregistrement, Z$ est une variable n'ayant aucune signification mais permettant simplement de se positionner correctement pour la deuxième série de variables.

Le buffer se présente ainsi



Note : 55 octets restent inutilisés en fin de buffer).

L'instruction GET permet de placer un enregistrement d'un fichier aléatoire sur disque dans le buffer.

Exemple : GET 1,1 : placer le 1e enregistrement dans le buffer 1

GET 2,7 : placer le 7e enregistrement dans le buffer 2

Exemple de programme (supposant qu'un fichier aléatoire DATA/TXT a déja été crée) :

```
10 OPEN "R",1, "DATA/TXT"
20 FIELD 1, 30 AS A$, 70 AS B$, 120 AS C$, 35 AS D$
30 GET 1,1
40 PRINT A$, B$, C$, D$
50 GET 1,3
60 PRINT C$
70 CLOSE
```

Si vous utilisez l'instruction GET 1,30, le fichier ne contenant que 20 enregistrements,il n'y aura aucun message d'erreur, l'ordinateur remplira simplement le buffer de blancs. Pour connaitre le numéro du dernier enregistrement, il faut utiliser la fonction LOF (lengh of file : longueur du fichier) qui détermine ce numéro.

Exemple :

```
10 OPEN "R",1, "DATA/TXT"
20 FIELD 1, 100 AS A$, 100 AS B$, 55 AS C$
30 FOR I% = 0 TO LOF(1)
40 GET 1, I%
50 PRINT A$ : PRINT B$ : PRINT C$ : PRINT
60 NEXT I%
70 CLOSE : END
```

L'instruction PUT permet de placer le contenu du buffer dans le fichier aléatoire :

Exemple : PUT 1,1 : le contenu du buffet 1 devient le 1er enregistrement

PUT 2,7 : le contenu du buffer 2 devient le 7e enregistrement

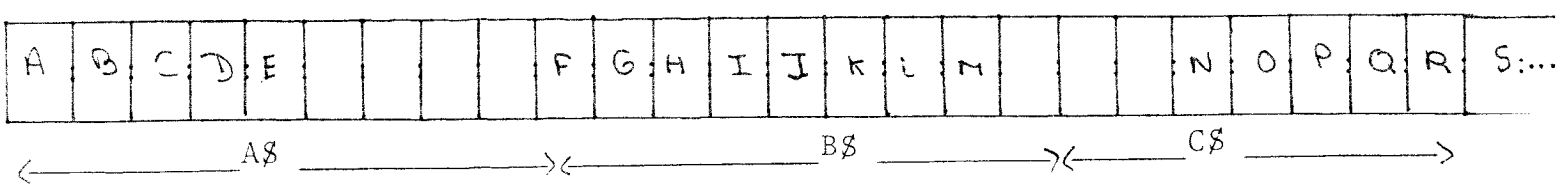
Attention : il ne peut pas y avoir plus de 335 enregistrements sur la même disquette . Si vous utilisez PUT 1,337 l'ordinateur vous répond : DISK FULL (disque plein) car il a créé (ou essayé de créer) les 336 enregistrements numérotés de 1 à 336 pour pouvoir enregistrer le 337é ; au 335e (avec une disquette formatté vide) l'espace utile de la disquette est utilisé et l'ordinateur ne peut plus rien enregistrer.

Pour pouvoir remplir le buffer avec des données, on utilise les instruction LSET et RSET, qui permettent de placer les données (chaînes de caractères obligatoirement) dans les espaces attribués aux variables pas l'instruction FIELD, en cadrant à gauche (LSET) ou à droite (RSET).

Exemple :

```
10 OPEN "R",1, "DATA/TXT"
20 FIELD 1, 10 AS A$, 10 AS B$, 10 AS C$
30 LSET A$ ="ABCDE"
40 LSET B$ ="FGHIJKLM"
50 RSET C$ = "NOPQRS"
60 PUT 1,1
70 ....
```

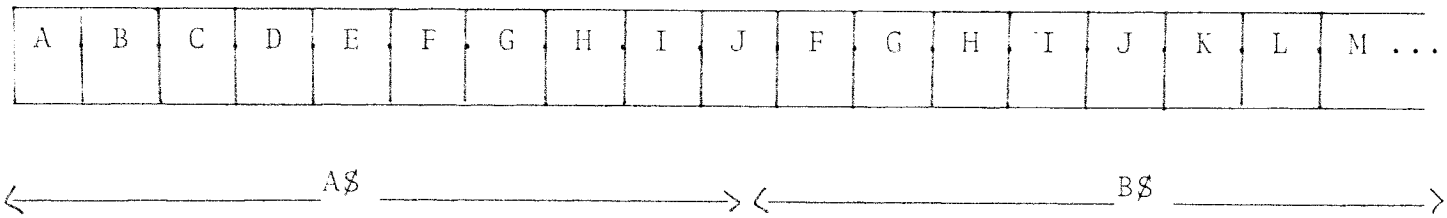
Le 1e enregistrement sera ainsi :



A$ et B$ étant cadrés à gauche et C$ cadré à droite, les octets non utilisés étant remplis de blancs. (ainsi que les 225 octets restants à droite de C$!).

Si la chaine de caractères est trop longue pour la place allouée, elle est tronquée à droite :

Si on remplace la ligne 30 par LSET A$ = "ABCDEFGHIJKLMNO"
alors le 1 e enregistrement sera :



Les données doivent être sous forme de chaînes de caractères. Si certaines données sont des nombres, il faut les convertir grâce aux instructions MKD$, MKI$ et MKS$.

MKI$ est utilisée pour les nombres entiers et retourne une chaîne de deux octets ;

MKS$ est utilisée pour les nombres en simple précision et retourne une chaîne de quatre octets;

MKD$ est utilisée pour les nombres en double précision et retourne une chaîne en huit octets

Pour reconvertir les chaînes de caractères en nombres, il faut utiliser les instructions CVD, CVI, CVS.

CVI convertit une chaine de 2 caratères en un nombre entier ; CVS convertit une chaine de 4 caractères en un nombre simple précision ; CVD convertit une chaine de 8 caractères en un nombre double précision. Si les chaînes ont moins de 2,4 ou 8 caractères, l'ordinateur affiche le message d'erreur 'ILLEGAL FONCTION CALL' (appel illegal à une fonction). Si les chaines ont plus de 2,4 ou 8 caractères, seuls les 2,4 ou 8 premiers caractères sont pris en compte.

3°) Exemple de programme avec un fichier à accès sequentiel :

Voici un ensemble de programmes permettant la gestion (extrêmement simplifiée) d'un stock d'articles. Pour chaque catégorie d'articles, le fichier a son identification (codée sur 5 caratères), le nombre d'articles, le prix d'achat, le prix de vente. Le fichier se nomme STOCK/TXT.

Le premier programme sauvé sous le nom CREATION/BAS permet la création du fichier.

```
10 REM PROGRAMME CREANT LE FICHIER
20 OPEN "O",1, "STOCK/TXT"
30 INPUT "CODE DE L'ARTICLE"; C$
40 C$ = LEFT$ (C$,5)
50 INPUT "NOMBRE D'ARTICLES"; N%
60 INPUT "PRIX D'ACHAT"; PA
70 INPUT "PRIX DE VENTE"; PV
80 PRINT #1, C$; ","; N%; PA;PV
90 INPUT "AUTRE ARTICLE";Z$
100 IF LEFT$ (Z$,1) = "O" THEN 30
110 CLOSE : END
```

Le second programme sauvé sous le nom MODIFIC/BAS permet la modification du fichier :

```
10 REM PROGRAMME MODIFIANT LE FICHIER
20 Z = 0
30 OPEN "I",1,"STOCK/TXT"
40 IF EOF(1) THEN 60
50 INPUT #1, C$,N%,PA,PV : Z = Z + 1 : GOTO 40
60 CLOSE
69 REM LES LIGNES 20 A 60 COMPTENT LE NOMBRE DE CATEGORIES
70 OPEN "I",1, "STOCK/TXT"
80 DIM C$(Z), N%(Z), PA(Z), PV(Z)
90 FOR I% = 1 TO Z : INPUT #1, C$(I%), N%(I%), PA(I%), PV(I%)
100 NEXT I% : CLOSE
102 REM LES LIGNES 70 A 100 LISENT LE FICHIER ET LE RANGENT EN MEMOIRE
105 REM LES LIGNES 110 A 230 MODIFIENT LE FICHIER
110 FOR I% = 1 TO Z
120 CLS : PRINT "ENREGISTREMENT NO.";I%
130 PRINT
140 PRINT "CODE =";  C$ (I%)
150 PRINT "NOMBRE =";  N% (I%)
160 PRINT "PRIX D'ACHAT =";  PA(I%)
170 PRINT "PRIX DE VENTE =";  PV(I%)
180 PRINT : PRINT "APPUYEZ SUR<ENTER>SI VOUS NE MODIFIEZ PAS."
190 K$ = "" : INPUT "NOUVEAU CODE";K$ : IF K$()"" THEN C$(I%)=LEFT$(K$,5)
200 K$ = "" : INPUT "NOMBRE ACTUEL";K$ : IF K$()""THEN N%(I%)=VAL(K$)
210 K$ = "" : INPUT "NOUVEAU PRIX D'ACHAT"K$ : IF K$()""THEN PA(I%)=VAL(K$)
220 K$ = "" : INPUT "NOUVEAU PRIX DE VENTE" ;K$ : IF()""THEN PV(I%)=VAL(K$)
230 NEXT I %
239 REM LES LIGNES 240 A 270 RECRIVENT LE FICHIER
240 OPEN "O", 1 "STOCK/TXT"
250 FOR 1% = 1 TO Z
260 PRINT #1, C$ (1%) ; ","; N% (I%) ; PA (I%) ; PV (I%) ;
270 NEXT I%  : CLOSE : END
```

Le troisième programme sauvé sous le nom EXTENSI/BAS permet d'étendre le fichier :

```
10 REM PROGRAMME AUGMENTANT LE FICHIER
20 OPEN "E",1,"STOCK/TXT"
30 INPUT "CODE DE L'ARTICLE"; C$
40 C$ = LEFT$(C$,5)
```

```
50 INPUT "NOMBRE D'ARTICLES";N%
60 INPUT "PRIX DE VENTE"; PV
70 INPUT "PRIX D'ACHAT"; PA
80 PRINT #1,C$;",";N%;PA;PV
90 INPUT "AUTRE ARTICLE";Z$
100 IF LEFT$(Z$,1) ="O "THEN 30
110 CLOSE : END
```

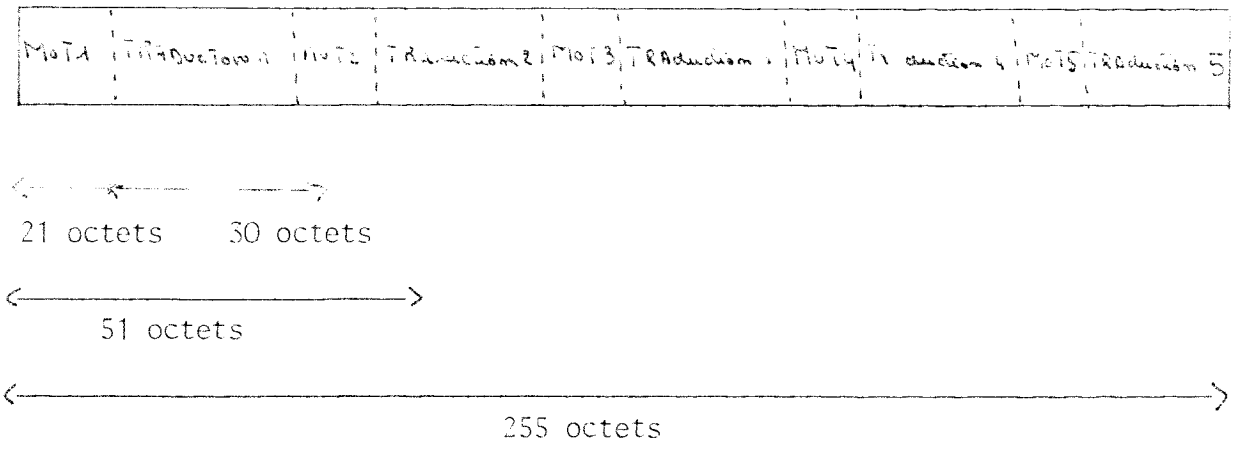
Le quatrième programme sauvé sous le nom EXPLOIT/BAS permet d'exploiter le fichier :

```
10 REM PROGRAMME EXPLOITANT LE FICHIER
20 OPEN "I",1,"STOCK/PXT"
30 CLS : X=0 :Y=0
40 PRINT "ARTICLE #"; TAB(13);"NOMBRE"; TAB(25);"PRIX";TAB(36);"PRIX";
   TAB(46);"VALEUR"
50 PRINT TAB(23); "D'ACHAT";"TAB(34);"DE VENTE"
60 IF EOF(1) THEN 140
70 INPUT #1, C$, N%, PA, PV
80 V = PV* N% = V2 = PA* N%
90 PRINT C$; : PRINT USING "ØØØØØØØØ###";N%;
100 PRINT USING "ØØØØØ#####.##";PA;
110 PRINT USING "ØØØØØ#####.##";PV;
120 PRINT USING "ØØØØØØ#####.##";V;
130 X = X + V  : Y = Y +V2  GOTO 60
140 PRINT : B = X-Y
150 PRINT "VALEUR A L'ACHAT DU STOCK =";Y
160 PRINT "VALEUR A LA VENTE DU STOCK =";X
170 PRINT "BENEFICE ESPERE = ";B
180 CLOSE : END
```

4°) Exemple de programme avec fichier a accès aléatoire

Nous voulons créer un programme permettant de traduire en français un certain nombre de mots anglais utilisés en pro

Nous avons vu que dans ce mode d'accès, on utilise comme unité d'enregistrement un secteur complet de 255 octets. Les mots anglais et leur traduction ne remplissant pas les 255 octets, nous pouvons décider de diviser chaque enregistrement physique de 255 octets en 5 sous-enregistrement de 51 octets Ces 51 octets sont divisés en 21 octets pour le mot anglais et 30 octets pour sa traduction en français . Le secteur se présente alors de la façon suivante :



Pour délimiter les variables dans le champ, on utilise l'instruction suivante :

FIELD 1, ((I%-1)*51) AS Z$, 21 AS M$,30 AS T$
ainsi lorsque I % = 1 , cette instruction est équivalente à :

FIELD 1, 0 AS Z$, 21 AS M$, 30 AS T$
et M$ et T$ pointent le 1er mot et sa traduction.

Lorsque I% = 3, l'instruction est équivalente à :
FIELD 1, 102 AS Z$, 21 AS MS, 30 AS T$
et M$ et T$ pointent le 3e mot et sa traduction.(Z$ n'ayant aucune signification). Cette technique permet d'économiser de la place sur la disquette.

Pour connaitre les positions respectives de l'enregistrement et du sous-enregistrement, on utilise les formules suivantes :

P désigne le numéro de l'enregistrement
S désigne le numéro d'entrée du sous-enregistrement
N désigne le nombre de sous-enregistrement par enregistrement
R désigne le numéro du sous-enregistrement dans l'enregistrement

$$P = INT((S-1)/N) + 1$$
$$R = S-N*(P-1)$$
$$S = (P-1)*N + R$$

Dans notre programme où chaque enregistrement est divisé en 5 sous-enregistrements, on prendra N=5. Si nous voulons savoir où est enregistré le 29e mot du fichier :

$P = INT((29-1)/5 + 1 = 6$
$R = 29-5*(6-1)=4$

Le 29e mot est donc le 4e mot du 6e enregistrement

Si nous voulons connaitre le numéro d'entrée du 3e mot du 4e enregistrement :

$S = (4-1)*5+3=18$

Ce mot est donc le 18e mot du fichier

Pour des raisons de simplification le programme proposé sera découpé en plusieurs programmes ayant chacun une seule fonction.

Le premier programme va créer un fichier nommé DICT/TXT où seront enregistrés les mots anglais et leur traduction :

```
10 OPEN "R",1,"DICT/TXT"
20 I%=1 : J%=1 : REM I% EST LE NUMERO D'ENREGISTREMENT ET J% CELUI DU SOUS-ENR.
30 FIELD 1, ((J%-1) 51) AS Z$, 21 AS M$, 30 AS T$
```

```
40 PRINT "ENREGISTREMENT NO."; I%;" SOUS ENREGISTREMENT NO.; J%
50 PRINT "NUMERO D'ENTREE :"; (I%-1)*5+J%
60 INPUT "MOT ANGLAIS:";M2$
70 IF LEN(M2$)>21 THEN PRINT"MOT TROP LONG" : GOTO 60
80 LINE INPUT "TRADUCTION : ?";T2$
90 IF LEN(T2$)>30 THEN PRINT"TRADUCTION TROP LONGUE" : GOTO 80
100 LSET M$=M2$ : LSET T$=T2$
110 INPUT "AUTRE ENTREE";ZZ$
120 IF LEFT$ (ZZ$,1)="N" THEN 150
130 IF J%=5 THEN PUT 1,I% : I%:I%+1 : J%=1 : GOTO 30
140 J%=J%+1 : GOTO 30
150 PUT 1,I% : CLOSE 1 : END
```

Exécutez ce programme et entrez tous les mots anglais dont vous avez besoin.

Exemple : ARRAY : tableau; BYTE : octet ; DATA : données ;FILE : fichier ; INPUT : entrée ; INTEGER : entier ; LOAD : charger ; MERGE : fusionner ; OUTPUT : sortie ; OVERFLOM : depassement de capacité ; READ : lire ; REAL : réel ; RECORD : enregistrement ; RUN : éxecuter ; SAVE : sauver ; SORT : trier ; STORE : ranger ; STRING : chaîne (de caractères) ; SUBROUTINE : sous programme ; UPDATE : mettre à jour ; etc .....)

Le second programme va permettre d'utiliser ce fichier DICT/TXT pour rechercher la traduction de mot anglais.

La ligne 60 demande à être expliquée : sur la disquette, nous avons attribué 21 octets pour le mot anglais, aussi T$(J%) seront de chaînes de 21 caractères, les caractères les plus à droite étant des blancs (code 32 décimal); aussi il faut rajouter au mot N$ 21 blancs à droite , puis prendre les 21 premiers caractères, afin d'en faire une chaîne de 21 caractères.

```
10 CLEAR 2000
20 DIM M$(5), T$(5)
```

```
30 OPEN "R", 1 "DICT/TXT"
40 FIELD 1,21 AS M$(1), 30 AS T$(1), 21 AS M$(2), 30 AS T$(2)
   21 AS M$(3), 30 AS T$(3), 21 AS M$(4), 30 AS T$(4)
   21 AS M$(5), 30 AS T$(5)
50 INPUT "QUEL EST LE MOT A RECHERCHER"; N$
60 M$ = N$ + STRING $(21,32) : M$=LEFT$(M$,21)
70 FOR I%=1 TO LOF(1)
80 GET 1,I%
90 FOR J%=1 TO 5
100 IF M$=M$(J%) THEN 140
110 NEXT J%,I%
120 PRINT "CE MOT N'EST PAS DANS LE DICTIONNAIRE"
130 GOTO 150
140 PRINT " LA TRADUCTION DE"; N$;" EST "; T$(J%)
150 INPUT "AVEZ VOUS UN AUTRE MOT A RECHERCHER";Z$
160 IF LEFT$(Z$,1)="O" THEN 50
170 CLOSE : END
```

5°) Comparaison des deux modes de fichiers :

Les fichiers à accès séquentiel sont plus simples a utiliser que les fichiers à accès aléatoire. Mais les fichiers à accès aléatoire présentent les avantages suivants : 1) au lieu de commencer la lecture au début du fichier, on peut lire n'importe quel enregistrement donné, 2) l'accès aléatoire est plus rapide, la lecture ou l'écriture prennent moins de temps, 3) un fichier ouvert en mode aléatoire permet la lecture et l'écriture, alors qu'un fichier ouvert en mode séquentiel permet soit la lecture, soit l'écriture.

Pour mettre à jour un fichier :

- S'il est en mode séquentiel :
deux façon de faire :

Ouvrir le fichier en mode séquentiel entrée (OPEN"I",...) ; entrer la totalité du fichier (par INPUT n ou LINE INPUT n) et la stocker dans un ou plusieurs tableaux ; fermer le fichier (par CLOSE); corriger, additionner, retirer toutes les données

que vous voulez ; ouvrir à nouveau le fichier en mode séquentiel sortie (OPEN"O",...) ; sortir la totalité du fichier (par PRINT ≠ n) ; fermer le fichier (par CLOSE).

Cette façon de faire n'est possible qu'avec de petits fichiers pouvant tenir en totalité dans la mémoire centrale.

Si au contraire, le fichier est assez important :

Ouvrir le fichier en mode séquentiel entrée; ouvrir un autre fichier en mode séquentiel-sortie ; entrer une donnée ; la corriger (ou non!) ; sortir la donnée dans le nouveau fichier; recommencer les 3 précédents points jusqu'à la fin du fichier ; ajouter au besoin de nouvelles données ; fermer les deux fichiers; détruire l'ancien fichier ; renommer le nouveau fichier avec le nom de l'ancien fichier (par la commande DOS: RENAME).

S'il est en mode aléatoire :

Ouvrir le fichier (par OPEN"R",...); découper le buffer (par FIELD...); entrer les données à modifier (par GET...) afficher le contenu de l'enregistrement (ne pas oublier d'utiliser CVD,CVI, CVS avant d'afficher les données numériques) placer les nouvelles valeurs des données dans le buffer (par LSET ou RSET) (ne pas oublier d'utiliser MKD$, MKI$, MKS$ avant de placer des données numériques); sortir les données (par PUT...); recommencer jusqu'à avoir corrigé toutes les données en ayant besoin, à partir du 3e point (GET...) ; fermer le fichier (par CLOSE).

6°) Messages d'erreurs :

FIELD OVERFLOW : ('dépassement du champ') : plus de 255 octets alloués à un buffer en mode aléatoire.

INTERNAL ERROR : ('erreur interne') : erreur dans le DOS lui même, ou défaut du disque.

BAD FILE NUMBER : ('numéro de fichier erroné') : un numéro de buffer a été utilise incorrectement : le numéro n'est pas celui assigné au fichier lors de son ouverture par OPEN.

FILE NOT FOUND : ('fichier non trouvé') : essai de lecture d'un fichier non contenu dans la disquette.

BAD FILE MODE : ('mode de fichier érroné') : essai d'entrée ou de sortie en contradiction avec le mode d'ouverture du fichier.

DISK I/o ERROR : ('erreur d'entrée/sortie du disque') : une erreur est intervenue durant le transfert d'une donnée entre l'ordinateur et le disque.

DISL FULL : ('disque plein') : tout l'espace du disque à été utilisé .

INPUT PAST END : ('les entrées dépassent la fin') : durant une entrée séquentielle, lecture de plus de données qu'il n'y en a sur le disque.

BAD RECORD NUMBER : ('numéro d'enregistrement erroné') : dans une instruction PUT, le numéro d'enregistrement n'est pas compris entre 1 et 340.

BAD FILENAME : ('nom de fichier erroné') : erreur dans le nom du fichier.

DIRECT STATEMENT IN FILE : ('instruction directe dans le fichier') essai de LOAD, RUN, MERGE,... un fichier qui n'est pas en BASIC

TOO MANY FILES : ('trop de fichier') : essai de placer plus de 48 fichiers sur une disquette.

DISK WRITE PROTECTED ('disque protégé à la lecture') :

FILE ACCES DENIED : ('accès au fichier interdit') : le fichier

5° PARTIE : LES UTILITAIRES

1°) EDTASM/CMD

Ce programme éditeur/assembleur est une version modifiée de l'éditeur/assembleur sur cassette de TANDY RADIO-SHACK Les modifications sont les suivantes :

- Possibilité de stocker et de lire les programmes sources et objets sur disquette aussi bien que sur cassette.

Pour écrire un programme source :
sur disquette : W D = NNNNNNNN/TTT.PPPPPPPP : L
sur cassette : W T = NNNNNN

Pour lire un programme source :
d'une disquette : L D = NNNNNNNN/TTT.PPPPPPPP
d'une cassette : LT = NNNNNN

NNNNNN est un nom de programme de 6 lettres.
NNNNNNNN/TTT.PPPPPPPP : L est une désignation de fichier sur disquette (N...N = nom, TTT = extention, P...P = mot de passe, L = n° de lecteur)

Avant la lecture d'un programme source, si le buffer de texte contient déjà un texte, le programme pose la question : 'Text in buffer- Are you concatenating ?' (texte présent dans le buffer- Voulez-vous concatener ?). Si vous répondez N (Non), le texte du buffer est effacé et le programme source est chargé. Si vous répondez Y(oui), le texte du programme source est ajouté à la suite du texte du buffer. Mais les chevauchements de numérotation ne sont pas testés, aussi pour que les deux textes soient compatibles, vous pouvez d'abord utiliser la command'N' de l'EDTASM avant la concatenation, pour assurer une bonne suite de numéros de lignes.

Pour écrire un programme objet :
sur cassette : utilisez : la commande A NNNNNN sans option
sur disquette : utilisez la commande A sans option.

Le programme vous demande alors : 'object file to disk or tape ? reply Dor T' (fichier objet sur disque ou cassette. Répondre D ou T)
Si vous répondez T (programme objet sur cassette), l'assemblage se fait et l'écriture de programme se fait sous le nom NNNNNN donné avec la commande A
Si vous répondez D (programme objet sur disquette), le programme vous demande : 'object filespec ?' (désignation du fichier objet?).

Répondez par une désignation de fichier au format habituel (NNNNNNNN TTT.PPPPPPPP : L), l'extention TTT est dans ce cas CMD.Le fichier est alors ouvert, mais l'écriture du programme objet ne se fera qu'une fois l'assemblage terminé.
A l'ouverture d'un fichier (qu'il soit programme source ou objet)sur disquette le programme a deux continuations :

- 'file already exists- Use-it ????' (le fichier existe déjà l'utilisez-vous ?) répondre par Y pour oui et par breack pour non.
- '*** file non-existent - Reply "C" to create it' (le fichier n'existe pas. Répondez par "C" pour le créer). Répondre par C pour le créer et par breack sinon.

La liste des symboles employés dans le programme source est donnée en ordre alphabétique, avec une liste de références.
Pour les instructions 'DEFM', seul le premier octet du code objet associé est listé.
Le programme accepte les minuscules, mais les convertit en majuscules.
Le programme permet l'emploi de [↓] pour lister les 15 lignes suivantes du texte.

2°)DISASSEM/CMD

Ce programme est un désassembleur qui désassemble les codes Z80, et peut fonctionner sur des modules chargés en mémoire ou sur disquette.
Ce désassembleur est écrit en code Z80.

Utilisation :
Tapez la commande DOS : DISASSEM.
Le programme pose alors la question : 'object from main memory or disk ?' (programme objet est en mémoire-centrale ou sur disque ?)
Les réponses sont : D ou [enter] si le programme est sur disquette
M si le programme est en mémoire centrale.
Si vous avez répondu M, le programme pose la question : 'object virtual base address (hex) ?' (adresse de base virtuelle du programme objet). Répondre en donnant 1 à 4 chiffres hexadécimaux dont la valeur indique la position où est exécuté le programme objet. Dans le listing, cette valeur est la valeur de la location de la première instruction imprimée.

La question suivante est 'Object real base address (hex) ?' (adresse de base réelle du programme objet). Il y a deux réponses possibles : [enter] si la location réelle est la même que la location virtuelle. 1 à 4 chiffres hexadécimaux indiquant la location mémoire où le désassembleur trouvera actuellement le code objet.
si vous avez répondu D, le programme pose la question 'filespec ?' (désignation du fichier ?) répondre en donnant la désignation complète (NNNNNNNN/TTT.PPPPPPPP : L). Puis le programme questionne : 'offset object virtual addresses by (hex) ?' (déplacement des adresses virtuelles de (hex) ?). Deux réponses possibles : [enter] signifie offset = 0. 1 à 4 chiffres hexadécimaux dont la valeur ajoutée aux adresses du programme donnera les adressesoù est réellement exécuté le programme..Ce paramètre est obligatoire quand un programme est stocké en mémoire centrale à une place puis déplacé à une autre place pour être exécuté (exemple : un programme est chargé en C000-FFFF, mais est exécuté en 7000-AFFF. Appliquant un offset de B000, le désassembleur désassemble ce programme comme s'il était chargé en 7000-AFFF).
Le programme demande ensuite : 'Virtual restart location (hex) ?' (location virtuelle de redémarrage (hex) ?). Deux réponses : [enter] signifie : aucune. 1 à 4 chiffres hexadécimaux indiquant la location virtuelle de redépart. Ceci permet lors d'un désassemblage très long ayant été interrompu, de redémarrer à la location donnée (en général celle de la fin du listing précédent).
Ensuite pour les deux possibilité, le programme demande : 'output to printer ?' (sortie sur imprimante ?) répondre Y pour oui et N pour Non (ou [enter]).
Si vous répondez oui : 'reply 'enter' when printer at top of page' (répondez 'enter' quand l'imprimante est positionnée en haut de page).
Le programme suppose qu'il y a 66 lignes/page et saute les 3 premières lignes et les 3 dernières lignes de chaque page.
Si le programme est situé en mémoire centrale : 'byte count (hex) ?' (nombre d'octets (hex) ?). Répondre par 1 à 4 chiffres hexadécimaux indiquant le nombre d'octets à désassembler.
Si vous avez répondu non (le listing va alors sur l'écran) : si le programme objet est sur disquette , la table de location n'est pas listée.

'Normal display pauses ?' (pauses normales dans l'affichage ?).
Les réponses sont : Y ou enter : pendant le listing des instructions désassemblées, le programme s'arrête à la fin de chaque page affichée; pour obtenir la page suivante , appuyez sur 'enter'; pour terminer le désassemblage, appuyez sur 'X'.
N : les instructions désassemblées s'affichent aussi vite que le programme le peut.
Pour l'arrêter appuyez sur 'P' ; pour repartir, appuyez sur 'enter' ; pour terminer le désassemblage, appuyez sur 'X'.

Si le programme objet est sur disquette, le programme ajoute la question : 'any options ?' (options ?). Les différentes réponses sont :

'NIP' : ne pas imprimer ou afficher les instrutions désassemblées. Ceci n'affecte pas l'affichage du premier passage ('location cross-reference pass').
'RTD' : La table des références doit être stockée sur disquette. Après le premier passage le programme demande : 'reference table filespec ?' (désignation du fichier de la table de référence ?). Répondre en donnant la désignation du fichier au format habituel (NNNNNNNN/TTT.PPPP PPPP ; L). Les fichiers de tables de références sur disquettes peuvent être utilisés (par un programme créé par l'utilisateur) pour fusionner les tables de références de deux ou plusieurs programmes.
'REA' : permet le listing de tous les types de références (c'est la condition de défaut).
'RE&' : permet le listing du type de références spécifié, ou & est l'un de : L,P,R,S,T,U,V,W,X. Les types de références sont définis au début de chaque listing de table de location.
'RIA' : interdit le listing de tous les types de références.
'RI&' : interdit le listing du type de références spécifiés où & est l'un de : L,P,R,S,T,U,V,W,X.

Pour avoir plus de renseignements sur les tables de références voir la documentation du NEWDOS page 27.

Si le désassembleur trouve quelque chose de mauvais dans le programme objet, il affiche le message : 'Disk object file format not as expected' (le format du fichier objet sur disque n'est pas celui attendu).
Si le DOS retourne une erreur, '&& Disk Error' est affiché et le désassemblage est terminé.
&& peut-être : 1C ou 1D : EOF (l'objet n'est un programme machine ???)
25 : le programme objet est protégé à la lecture

1B : plus d'espace sur la disquette
19 : le programme objet a un mot-de-passe
18 : le fichier objet n'existe pas.

Dans le listing des instructions désassemblées :
La colonne 1 indique le nombre de références faites aux octets de cette instruction, les valeurs sont en hexadécimal (0 et F indiquent 15 références ouplus).
La colonne 2 indique quel octet de l'intruction a été référencié. Si la colonne 2 est blanche et la colonne 1 non-blanche , seul le premier octet est référencié. Sinon le chiffre hexadécimal est un masque de 4 bits indiquant la position de l'octet référence, depuis la gauche.

3°) DIRCHECK/CMD

Ce programme permet de tester et de lister le directory d'une disquette. Pour l'exécuter, tapez la commande DOS : DIRCHECK.
Le programmedemande 'Output to printer ?' (sortie sur imprimante ?). Répondre par Y pour oui et par Npour non.
Le programme demande alors : 'wich drive contains target diskette ?' (quel lecteur contient la disquette cible ?). Répondre par le numéro du lecteur, 0,1,2,ou 3.
Si le directory ne contient aucune erreur, le programme affiche alors (début du directory de la disquette du NEWDOS) :

```
NEWDOS    11/15/79

BASIC/CMD       IP = 6       EOF = 19/149        2 EXTS    20 SECTORS
BODT/SYS        SIP = 6      EOF =  5/0          1 EXTS     5 SECTORS
COPY/CMD        IP = 6       EOF =  4/253        1 EXTS     5 SECTORS
DIR/SYS         SIP = 5      EOF = 10/0          1 EXTS    10 SECTORS
DIRCHECK/CME                 EOF = 12/236        3 EXTS    15 SECTORS
DISASSEM/CMD                 EOF = 19/134        1 EXTS    20 SECTORS
    .
    .
    .
    .
    .
SYS5/SYS        SIP = 7      EOF =  4/203        1 EXTS     5 SECTORS
SYS6/SYS        SIP = 7      EOF = 13/33         1 EXTS    15 SECTORS
```

```
5 FREE GRANULES.      0 LOCKED-OUT GRANULES.
NEWDOS                DIRECTORY CHECK & LIST COMPLETED
FUNCTION COMPLETED.

START PROGRAM ANEW ?
(le programme doit-il recommencer ?)
```

La première ligne donne le nom de la disquette, et la date de sa création.
Ensuite, tous les programme sont listé par ordre alphanumérique avec les renseignements suivants :

S = le fichier appartient au système (DOS)

I = le fichier est invisible

P=N = le fichier est protégé par des mots de passe, N indiquant le niveau de protection (7 = pas d'accès ; 6 = accés pour exécution seulement ; 5 = accès pour exécution et lecture ; 4 = accès pour éxécution, lecture et écriture ; 2 = accès pour exécution, lecture,écriture, changement de nom ; 1 = accès pour exécution, lecture, écriture, changement de nom, effacement).

EOF = AAA/BBB = indicateur de fin de fichier. AAA est le numéro de secteur relatif dans le fichier. BBB est le numéro d'octet relatif dans le secteur.

N EXTS = N est le nombre d'éléments d'extention (= nombre d'aires de stockage)

NN SECTORS = nombre de secteurs alloués au fichier.

Puis est affiché : la place libre restante (XX FREE GRANULES) en granules (5 secteurs) ; le nombre de granules locked out (xx LOCKED-OUT GRANULES).

Si le directory contient des erreurs, celles-ci sont affichées en premier ; tous les nombres mentionnés sont alors en hexadécimal afin de permettre les corrections par le programme 'SUPERZAP'. Ne pas essayer de corriger un mauvais directory si vous ne savez pas ce que vous faites !!!*** Vous pourriez en effet détruire la totalité du contenu de la disquette. Si vous ne savez pas faire les corrections, recopiez les programmes corrects sur une autre disquette et reformattez la mauvaise disquette.
Si l'erreur a lieu dans une entrée du directory pour un fichier, (entrée primaire ou entrée étendue) le code hexadécimal donne le DEC (directory entry code) pour le FPDE (file primory entry) du fichier.
Si l'erreur a lieu dans un octet du secteur HIT, le code est l'adresse relative de cet octet dans le secteur HIT.

4°) LMOFFSET/CMD

Le programme LMOFFSET permet de charger en mémoire centrale des programmes en langage machine venantd'une cassette ou disquette, puis de stocker ce programme sur disquette, en changeant au besoin la position de stockage si des conflits avec les programmes du systèmes interviennent.

Le programme pose les questions suivantes :
'Source from disk or tape ? reply "D" or "T" ?
(Le programme source vient-il d'une disquette ou d'une cassette ? répondre par "D"ou "T")

Si le programme source vient d'une cassette répondre par T enter ; le programme déclenche alors le magnétophone et lit le contenu de la cassette. Si le programme n'est pas en langage machine (ou si le volume du magnétophone est mal règlé) le message : 'source illegal load to-memory address' '"enter" to restart program anew ?' est affiché. (Le programme source contient des adresses de chargement en mémoire illégales. Appuyez sur "enter" pour redémmarer le programme ?)
Si le programme source vient d'une disquette répondre par D enter ; le programme demande : 'Source filespec ?' (désignation du fichier source ?) répondre par cette désignation dans le format habituel (NNNNNNNN/TTT.PPPPPPPP : L). Si le programme désigné n'existe pas le message 'Illegal filenamé' est affiché (nom de fichier illégal.
Si le programme désigné n'est pas en langage machine, le message 'record number out of range'est affiché (numéro d'enregistrement hors des limites).
Si tout ce passe correctement, le programme LMOFFSET affiche alors : 'Module loads to....' (le module se charge en ....) suivi de l'adresse du début et de l'adresse de fin du module en hexadécimal. Le programme affiche ensuite les conflits possibles avec les modules du système, par exemple :
'Module load overlaps DOS RAM (4000-51FF)' (le chargement du module recouvre les RAM du DOS) ou 'Module load will overlap "CMD" program area (5200-6FFF)' (le chargement du module recouvrira l'aire des programmes "CMD"). Le programme donne enfin le point d'entrée du programme en hexadécimal :
'Entry point = ...' . La question 'New load base adress (hex)?' (nouvelle adresse de base de chargement ?).

Si vous êtes satifait de l'actuelle base (pas de conflit avec les RAM-DOS, ...) appuyez simplement sur "enter" ; sinon tapez la nouvelle base en hexadécimal (de préférence, après 7000 H). Le programme redonne alors les mêmes renseignements qu'auparavant, mais en tenant compte de la nouvelle base. IL ajoute en même temps un court programme de 15 octets à la fin du module à charger dont le rôle sera, à l'exécution du module, de replacer celui-ci à sa véritable base d'exécution (sauf pour les modules se chargeant dans la zone d'affichage 3COO. 3FFF dont les adresse ne sont pas modifiées). Ce petit programest appelé en anglais 'appendage'.

Le programme pose la question 'shall appendage be suppressed (Y or N)' (l'appendage doit-il être supprimé ? (Y ou (N)). Si vous répondez N (non) le nouveau point d'entrée sera le premier octet de l'appendage. Si vous répondez Y(oui) alors le nouveau point d'entrée sera ØØØØH.
Interrupts to be disabled (Yor N) ?' (les interruptions doivent-elles être interdites ?) si vous répondez (Y) (oui), le programme ajoute un "DI" (disable interrupts) dans le petit programme appendage.
'Destination filespec ?' (désignation du fichier destination ?).
Répondre par une désignation de fichier au format habituel.
Le programme écrit alors le module sur la disquette puis affiche :
'Module write completed'
'"enter" to restart program anewo'
(écriture du module terminée)
(appuyez sur "enter" pour recommencer le programme).

Durant la lecture d'une cassette, un simple "*" est affiché lorsque le programme est prêt à lire. Rembobiner si nécessaire, et appuyez sur 'play'. "* * *" apparait lorsque la lecture des octets de synchronisation a été faite. Pendant la lecture, le "*" le plus à droite clignote. Si un mauvais 'checksum' est rencontré, un "C" apparait. Un "P" apparait lorsque le programme rencontre les octets supplémentaires en début de programme (avant le "V". Un I apparait lorsque le programme rencontre les octets de données supplémentaires en fait de programme (avant l'enregistrement suivant ou avant le point d'entrée).

Si le module doit se charger en 400-51FF, en mémoire DOS, lui ou/et le DOS pourront mal fonctionner. Si le module doit se charger en 5200-6FFF, en mémoire des programmes CMD, le DOS pourra l'exécuter comme une commande (le module doit avoir l'extention/CMD), mais l'instruction LOAD rejettera le module. Si le module doit se charger en 7000-FFFF, aucun problème.

ATTENTION : chaque fois qu'une nouvelle base de chargement est spécifiée pour un module, le programme ajoute un appendage, car il ne connaît pas et ne vérifie pas les 'appendages' précédents.

5°) LEVEL 1 (ou BASIC 1) et LV1DSKSL

Ces deux programme permettent de programmer en basic level I sur un TRS80, level II, en bénéficiant de l'usage des lecteurs de disquettes

Le premier programme nommé LEVEL 1 ou BASIC 1 suivant les disquettes (vérifiez dans le directory) est une version RAM modifiée de la ROM level I. Pour l'exécuter, tapez la commande DOS : LEVEL 1 enter ou BASIC 1 enter. Le programme affiche simplement READY. Vous êtes alors en basic level I.

Plusieurs commandes on été ajoutées :

1 - CSAVE*: le texte du programme basic est déplacé en mémoire centrale à l'adresse 8000H et au delà, au lieu d'être envoyé vers la cassette (instruction CSAVE) un 'RESET' suivi de la commande DOS 'LV1DSKSL' avec l'option 'S' stockera le programme level I situé en 8000H et au delà sur une disquette.

2 - CLOAD*: Tout d'abord, exécutez la commande DOS 'LV1DSKSL' avec l'option 'L'. Le programme en level I est chargé depuis une disquette en mémoire centrale en 8000H et au delà. Ensuite la commande DOS'BASIC 1' ou 'LEVEL 1' sera exécutée automatiquement, vous ramenant au basic level 1. En tapant 'CLOAD*', le texte du programme basic level 1 sera placé dans le buffer de texte et sera prêt à être exécuté.

3 - CSAVE $ $ $: un saut à l'adresse ØH est exécuté pour amener un redémarrage du level II, mais sans perte du contenu de la mémoire. Ainsi on peut revenir au level I en tapant : BASIC␣28160 puis SYSTEM,*?/28160

Le programmme LV1DSKSL s'occupe de stocker les programmes basic level 1 situés en mémoire centrale en 8000H et au-delà sur disquette, et de charger les programmes level 1 sur disquette en mémoire centrale toujours en 8000H et au-delà.
Le programme pose les deux questions :
'reply "S" for save, "L" for load' et 'file-spec ?'
(répondre "S" pour sauver, "L" pour charger) (désignation du fichier).
Si le message 'Bad Text Prefix' est affiché, soit le texte du disque n'est pas un programme écrit en level I , soit la mémoire centrale n'est pas bonne, soit vous avez oublié d'effectuer un 'CSAVE*'

6°) SUPERZAP

Ce programme écrit en Basic est en deux parties : SUPERZAP/COM et SUPERZAP. La première partie étant des commentaire sur l'utilisation de la seconde partie. Ce programme permet d'afficher sur l'écran, d'imprimer ou de modifier des secteurs de disquette. Pour l'exécuter tapez la commande DOS BASIC RUN "SUPERZAP" ou la commande basic RUN "SUPERZAP". Le programme affiche (après quelques instants) :
'For program documentation, run basic program superzap/com'
inuput on of the following functions :

'DD' or NULL - display disk sector
'PD' - print disk sector
'DM' - display main mémory
'PM' - print main mémory
'Verify disk sectors'
'Zéro disk sectors'
'Copy disk sectors'
'Disk Backup'
'Copy disk data'
?

(Pour une documentation sur le programme, exécutez le programme basic superzap/com.
Entrez l'une des fonctions suivantes :
' DD' ou rien - affichage d'un secteur de disquette
'PD ' - imprimer un secteur de disquette
'DM ' - affichage de la mémoire centrale
'PM ' - imprimer la mémoire centrale
'Vérifier les secteurs d'une disquette
' 'Mettre à zéro des secteurs d'une disquette
'Copier des secteurs d'une disquette
'Copier une disquette

'Copier des données d'une disquette).

ATTENTION : le programme peut mal fonctionner si vous essayez d'écrire sur une disquette protégée à l'écriture.

Les fonctions 'DD' et 'PD' permettent d'afficher ou d'imprimer un secteur de disquette (un secteur vaut 256 octets). Le programme demande :

Relative disk (0-3) ? (numéro relatif du lecteur de disquette (0-3))
Track (hex) (0-22) ? (numéro de piste (hexadécimal) (0-22))
Sector (0-9) ? (numéro de secteur (0-9))
Puis la fonction 'PD' : Sector count ? (nombre de secteurs)

L'affichage se fait de la façon suivante : colonne 1 : numéro relatif du disque (1,2,3) ; colonnes 2 et 3 : numéro relatif de la piste (en hexadécimal, de 00 à 22) ; colonne 4 : numéro relatif du secteur sur la piste (de 0 à 9) ; colonnes 5 et 6 position relative (en hexadécimale) dans le secteur(de 00 à FF) ; colonne 7 dernière ligne : si un 6 apparait, il veut dire que la lecture s'est effectuée correctement mais que le secteur est normalement protégé à la lecture. (c'est-à-dire, non lisible par les opérations normales, mais lu par superzap).

Les fonctions'DM' et PM' permettent d'afficher ou d'imprimer 256 octets de mémoire centrale.

Le programme demande :
'Start addresse (1 to 4 hex digits)' ?' (adresse de départ (1 à 4 chiffres hexadécimaux)).

Le programme affiche alors 256 octets de mémoire centrale, contenant l'adresse de départ spécifiée dans la réponse. (Par exemple, si vous donnez 1234 H comme adresse de départ, le programme affiche les 256 octets de 1200 H à 12FFH).

Pour stoper l'imprimante, appuyez sur la touche H jusqu'à l'arrêt de l'imprimante.

Lors de l'affichage sur l'écran d'un secteur (fonction 'DD') ou d'un bloc de mémoire (fonction 'DM', le programme teste périodiquement le clavier pour voir si l'une des touches suivantes a été

enfoncée :

[X] : termine la fonction et retourne au menu
[R] : affiche à nouveau le même secteur/bloc
[+] : affiche le secteur suivant ou bloc suivant
[-] : affiche le secteur/bloc précédent
[J] : recommence la même fonction
[K] : (seulement pour 'DD') : recommence la même fonction, mais sur le même disque
[MOD &&] : modifie le secteur/bloc courant. && est la position relative de l'octet à modifier (00 à FF) à l'intérieur du secteur/bloc.
Le programme répond en plaçant un M en colonne 7 de la ligne à modifier. Précédent le groupe de 4 chiffre à modifier, un des indicateurs suivants est affiché : + (1e chiffre), - (2° chiffre), (3° chiffre), / (4° chiffre).
Si maintenant vous tapez :
Un chiffre hexadécimal : 0-F : remplace le chiffre courant et passe au chiffre suivant (pour 'DM' les remplacements sont réels. Les remplacements pour 'DD' sont placés dans un buffer jusqu'à "enter" ou un "Q".) Quand le dernier chiffre affiché est remplacé, l'affichage du M se positionne au premier chiffre, mais plus aucun remplacement n'est accepté avant qu'une touche autre que 0-F ne soit pressée.
Un espace : le chiffre courant n'est pas modifié, et la position est avancée de 1 chiffre.
[→] : identique à un espace.
[Shift][→] : le chiffre actuel n'est pas changé, et la position est avancée de 4 chiffres
[←] : le chiffre actuel n'est pas changé, et la position est reculée de 1 chiffre
[Shift][←] : le chiffre actuel n'est pas changé, et la position est reculée de 4 chiffres.

 : le chiffre actuel n'est pas changé, et la position est remontée d'une ligne. La ligne suivant la première est la dernière.

 : le chiffre actuel n'est pas changé, et la position est descendu d'une ligne. La ligne suivant la dernière est la première.

Q : (seulement pour 'DD') : anulle l'écriture du secteur mis à jour. Termine le 'mode modification'.

enter : termine le 'mode modification'. Pour 'DD', le secteur mis-à-jour est réécrit sur la disquette.

SCOPY : (seulement pour 'DD') : déplace le secteur affiché vers une destination à spécifier = le programme demande :

DRIVE X,TRACK Y, SECTOR Z (Lecteur N° X, piste n°Y, secteur n° Z
IS TO BE COPIED TO est à copier en :
Relative disk # (0-3) ? n° relatif du disque (0-3) ?
Track ## (hex) (0-22) ? n° piste (hex) (0-22) ?
Sector # (0-9) ? n° secteur (0-9) ?)

puis : (1) Drive A, Track B, Sector C to be written is not read-protected - Shall it be read protected ?
(lecteur n° A, piste n°B, secteur n° C qui vient d'être écrit n'est pas protégé à la lecture. Doit-il être protégé à la lecture ?)

ou : (2) Drive A, Track B, Sector C to be written is read-protected - Shall it remain so ?
(lecteur n°A, piste n°B, secteur n°C qui vient d'être écrit est protégé à la lecture. Doit-il le rester ?)

COPY DONE (copie faite)
Reply "enter" to see result ? (appuyez sur "enter" pour voir le résultat)

La fonction 'Zéro Disk Sectors' (mettre à zéro des secteurs) fait exatement cela !

Le programme affiche : You are asking to zero out disk sectors
Be careful !!!!!!!!
(vous demandez de mettre à zéro des secteurs de disque. Soyez très prudent !!!!!!!)
Relative disk ## (0-3) ?
Track ## (hex) (0-22) ?
Sector ## (0-9) ?
Sector count ? (nombre de secteurs)

Si le secteur à mettre à zéro est protégé à la lecture, le programme demande si l'utilisateur désire que ce secteur reste protégé.

La fonction 'Copy Disk Sectors' copie les secteur spécifiés à une nouvelle location du disque, recopiant le statut de protection à la lecture. Normalement, la copie se fait en ordre de n° de pistes/secteurs croissants, mais si secteur de destination le plus bas est situé dans les secteurs à copier, la copie se fait en ordre décroissant de n° de pistes/secteurs.

Le programme affiche : Provide source base information (donnez les informations sur la source)
Relative disk ## (0-3) ?
Track ## (hex) (0-22) ?
Sector ## (0-9) ?
Provide destination base in information
(donnez les informations sur la destination)
Relative disk ## (0-3) ?
Track ## (hex) (0-22) ?
Sector ## (0-9) ?
Sector count ?

La fonction 'Vérify disk sectors' (vérifier les secteurs du disque) lit les secteurs spécifiée. Une pause intervient à chaque secteur non lisible.

Le programme demande : Pause when sector read-protected (reply Y or N)
(pause quand un secteur est protégé à la lecture (répondre Y ou N)).
Relative disk ## (D-3) ?
Track ## (hex) (D-22) ?
Sector ## (D-9) ?
Sector count ?

Le programme affiche alors les secteurs protégés qu'il rencontre :
Drive X, Track Y, Sector 2 read-Protected
Press 'enter' to continue (si vous avez répondu oui (Y) à la première question).
(lecteur n°X, piste n°Y, secteur n°Z protégé à la lecture. Appuyez sur "enter" pour continuer).

La fonction 'Copy Disk Data' (copier des données du disque) permet de copier une chaîne de moins de 65536 octets de longueur d'une position sur le disque à une autre position.

Le programme demande : Provide source base information
Relative disk ## (0-3) ?
Track ## (Hex) (0-22) ?
Sector ## (0-9) ?
Relative byte ## in sector (hex, 00-FF) ?
(numéro relatif de l'octet dans le secteur)
Provide destination base information
Relative disk ## (0-3) ?
Track ## (hex) (0-22) ?

Sector # (0-9) ?
Relative bite# in sector (hex, OO-FF) ?
Byte Count (hex) ? (nombre d'octet, hex)

Le statut de protection à la lecture des secteurs de destination ainsi mis à jour par la copie n'est pas changé.
La fonction 'Disk Backup' permet de faire une copie complète d'une disquette, sous réserve que la disquette de destination : (1) ne soit pas dans le même lecteur que la disquette source ; (2) soit formattée. Cette disquette n'est pas testée du point de vue contenu avant l'écriture, mais vérifiée après l'écriture.

Le programme affiche : ***Caution !!!!! destination disquette not tested before backup writes and cannot be non-formattée.
(Attention ! la disquette de destination n'est pas testée avant l'écriture du backup et ne peut pas être non-formattée.)
Source Drive # (0-3) ? (n°du lecteur source (0-3))
Destination Drive # (0-3) ? (n° du lecteur destination (0-3))

Si une erreur intervient, le programme affiche :
Sector read error (erreur de lecture d'un secteur)
Drive X, Track Y , Sector Z (lecteur X, piste Y, secteur Z)
Système error code A (code erreur du système A)
x x x x x (énoncé de l'erreur : Bad parity,...)
Reply 'R' for retry, 'S' for skip error sector, (répondre 'R' pour
ou 'X' to cancel function ? réssayer, 'S' pour sauter le secteur erreur,
ou 'X' pour abandonner la fonction)

Ce mode de copie se faisant par secteur de 256 octets est très long (plusieurs minutes) mais est utile pour des disquettes qui ne peuvent être copiées avec la commande DOS COPY.

7°) DISKDUMP/BAS

Ce programme écrit en basic permet de lire et d'afficher sur l'écran ou d'imprimer le contenu d'un fichier sur une disquette. Pour l'exécuter, tapez la commande DOS BASIC RUN "DISKDUMP/BAS" ou la commande basic RUN "DISKDUMP/BAS"

Le programme affiche :
Pressing 'P' will cause pause in dump display/list. To continu, press "enter"
Pressing 'X' will restart processing for same file
(Appuyez sur 'P' fait faire une pause dans l'affichage/listing du dump.Pour continuer, appuyez sur 'enter'. Appuyez sur 'X' fait repartir le processus pour le même fichier).
Filespec : (désignation du fichier)
Output to display or printer (reply 'D' or 'P')
(sortie sur écran ou imprimante (répondre 'D' ou 'P'))
Dump by sector or by logical record (reply "S" or "R")
(dump par secteur ou par enregistrement logique (répondre par S ou R))

Si la réponse est S : sector number (or 'enter'for next sector) :
(numéro du secteur (ou 'enter' pour le secteur suivant)

Si la réponse est R : logical records are displayed/printed in character format only
(les enregistrement logique sont affichés/imprimés en format caractères seulement)
Starting line number (>= 1) ?
(numéro de la ligne de départ (>= 1) .

Le programme affiche ou imprime alors le dump effectué, ou un message d'erreur.
(Comme EOF encountered (fin de fichier rencontré)...)

6ème PARTIE : ADDITIF

Le Basic-disque du NEWDOS permet d'utiliser les commandes du DOS au niveau du Basic. Ainsi, si vous êtes dans le basic-disque, la commande basic CMD"DIR : 1" vous permet d'obtenir un directory de la disquette située dans le lecteur n° 1, puis de revenir au basic. Le format de cette commande est : CMD "XXXX", ou XXXX remplace n'importe quelle commande du DOS (1ere partie). Y compris celle de programme créée par EDTASM (ainsi CMD "EDTASM" est parfaitement valable). XXXX peut aussi remplacer une variable chaîne de caractères. Ainsi si A$ = "FORMAT", la commande CMD "A$" est équivalente à CMD "FORMAT". La commande CMD "XXXX" produira le message d'erreur 'out of memory' s'il n'y a pas assez de mémoire libre entre le bas de l'aire de chaînes de caractères et le haut de l'aire des tableaux (approximativement 8 k sont réclamés). Avant d'aller au dos, un checksum est effectué sur toute l'aire basic en mémoire centrale qui n'est pas suceptible de changer par une fonction DOS. Ceci prend à peu près 2 secondes. Si la fonction DOS revient au basic, ce check sum est à nouveau effectué (encore 2 autres secondes). Si le check sum est faux, le basic ne peut pas continuer et une sortie est faite vers 'DOS READY'.

La plupart des commandes dos ainsi effectuées n'affectent pas le programme basic en mémoire. On peut le vérifier par un LIST.

Le NEWDOS permet le recopie fr l'écran sur imprimante, en appuyant en même temps sur les trois touches.JKL. Ceci ne peut être fait que lorsque l'ordinateur attend une entrée : soit avec le DOS READY, soit avec le READY du BASIC, soit dans un programme BASIC avec un INPUT ou un LINEINPUT.

Cela ne marche pas avec un INKEY$.