# MICRODISC ORIC


## Manuel d'utilisation du MICRODISC et du SEDORIC

SEDORIC a été écrit par Fabrice BROCHE & Denis SEBBAG

# Manuel d'utilisation du MICRODISC ORIC et du SEDORIC

**Conception : Fabrice BROCHE**

Réalisation : Georges EL ANDALOUSSI

# SOMMAIRE

Manuel d'utilisation du MICRODISC ORIC

# Chapitre 1

# Généralités

Manuel d'utilisation du MICRODISC ORIC

# A / POURQUOI UNE UNITE DE DISQUETTE 3"?

Depuis que vous utilisez votre ORIC, vous avez ressenti le besoin de conserver des informations (textes, programmes, fichiers etc..) , celles ci étant perdues dès que l'on éteint l'appareil. Il est donc nécessaire de transférer les informations entre la mémoire vive (RAM) et le support de stockage. De plus, le fait de disposer d'un support où l'on peut aller chercher les informations rapidement, permet d'utiliser des programmes importants qui seront chargés bloc par bloc au fur et à mesure du déroulement du programme, alors que le programme entier ne pourrait jamais tenir dans la mémoire (évidemment limitée) de l'ordinateur.
Vous avez surement déja utilisé la cassette comme support de stockage, mais c'est un support d'accès très lent et peu souple.
La disquette est un moyen beaucoup plus puissant et plus pratique:

- **capacité élevée** (170 à 212 k octets par face).
- **accès direct aux "blocs" d'informations**: comme sur un disque 33 tours sur lequel on peut poser la tête de lecture directement sur le sillon où commence le 3e morceau, on peut aller chercher directement un fichier sur tel ou tel secteur de telle ou telle piste.
- **Simplicité d'utilisation** : le fonctionnement de la disquette ne nécessite aucune manipulation "mécanique" (appuyer sur les touche STOP, PLAY etc..), tout étant géré par le Systéme d'Exploitation du Disque (SED) connu des habitués ou des anglophones sous le nom de DOS (Disk Operating System).
- **Performances : le SEDORIC** est le SED écrit spécialement pour l'ORIC, et qui permet d'obtenir les performances les plus remarquables d'un lecteur de disquettes 3" : **24 kOctets de programmes ou de fichiers chargés en moins de 3 secondes !**
- **Fiabilité** : L'unité de microdisque ORIC a recu de nombreux perfectionnements "HARD et SOFT " (matériel et logiciel) qui la mettent définitivement à l'abri des "Bugs" rencontrés avec les autres modèles.
- **Compacité** : la disquette au standard 3 pouces a été choisie car elle est extrèmement compacte (8 cm ), et qu'elle est entièrement protégée contre les accidents mécaniques: on peut sans crainte la glisser dans une poche de chemise ! Les seules chose qu'elle craint sont la chaleur intense et les rayonnements magnétiques. Le lecteur lui-même est compact, élégant et silencieux. Son alimentation secteur alimente également l'ORIC.

Le système d'exploitation **SEDORIC** fait bien plus que gérer le disque de la façon la plus efficace qui soit . Il contient également un BASIC étendu qui ajoute plus de 60 instructions au BASIC normal de l'ORIC. Ces instructions servent à écrire encore plus facilement (**NUM, RENUM** etc..) des programmes plus performants (**USING, WINDOW, BOX...**).

L'unité de microdisque ORIC donne ainsi à votre ordinateur une dimension et un niveau de performances bien plus élevés. Lisez donc soigneusement ce manuel afin de tirer le meilleur parti de tous ces avantages que vous apporte le Disque ORIC.

# B / INSTALLATION ET BRANCHEMENT

Avant de commencer,un principe à respecter **de façon impérative : dès qu'il s'agit de brancher quoi que ce soit sur le bus,**

**on éteint tout d'abord, et on réfléchit après !**

Le fait de brancher un périphérique sur la prise du Bus, l'appareil étant allumé, revient à enrichir les services après-vente : c'est la panne assurée de l'unité centrale comme du périphérique. On répète donc : **dès qu'il s'agit de brancher quoi que ce soit sur le bus, on éteint tout d'abord, et on réfléchit après !.**

Ceci étant dit, et l'appareil étant éteint, voyons comment brancher la configuration :

Le câble plat gris clair qui sort de l'unité de disquette se branche sur la prise du BUS (marquée **EXPANSION** sur le dessus du boitier de l'ATMOS). Un petit cran sur la prise (appelé détrompeur) oblige à brancher la prise dans le bon sens.

La longueur du cable étant limitée (pour réduire les risques d'interférence magnétique et électrique), on dispose généralement le lecteur à côté de l'ORIC ou derrière. Sur ce cable se trouve une prise type AMPHENOL 34 broches destinée à recevoir les extensions (interface joystick ou série..) que l'on branche habituellement sur le BUS. N'enlevez pas le petit capot plastique qui protège cette prise, le moindre court-circuit sur les broche produit les effets décrits ci-dessus (appelez le SAMU et direction le service après-vente !).

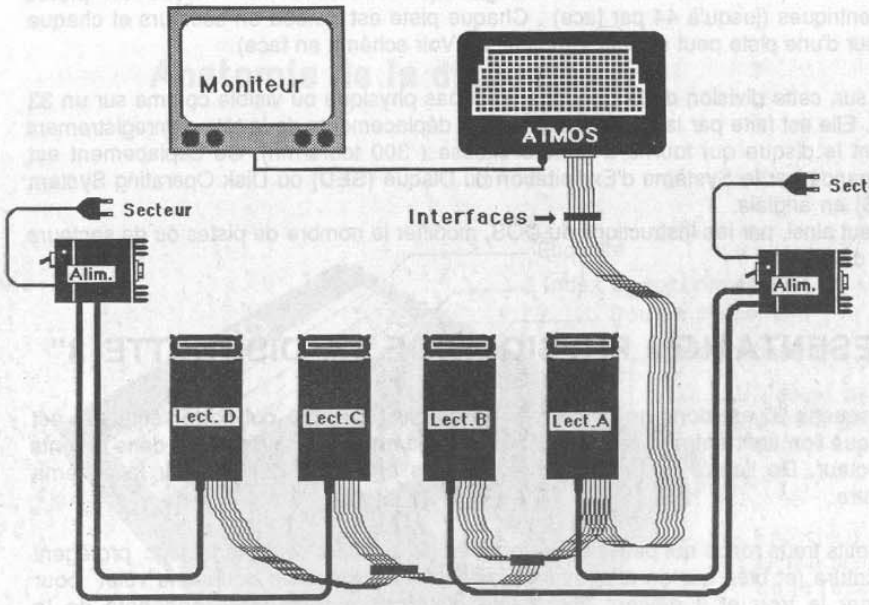
Derrière l'unité de disquette, on trouve la prise pour brancher l'alimentation, le bouton de RESET, et la prise pour brancher une deuxième unité de disquettes esclave (sans contrôleur) .

L'alimentation fournie avec l'unité de disquettes est spéciale : elle fournit les tensions d'alimentation pour 2 unités de disquettes (les deux gros câbles terminés par une prise DIN femelle) et également pour l'ORIC directement par le Bus . L'alimentation d'origine de l'ORIC n'est donc plus nécessaire. Faites donc tous les branchements (alimentation, moniteur, unité de disquette, et éventuellement imprimante, magnétocassette etc..), branchez la prise secteur de l'alimentation **et n'allumez l'interrupteur de l'alimentation secteur qu'en dernier lieu .**

L'écran vous demande alors **"INSERT SYSTEM DISK" :** Vous devrez donc insérer votre disquette. S'il s'agit de la disquette MASTER SEDORIC, l'ordinateur chargera le DOS et vous rendra la main sous BASIC (Curseur et READY) . L'affichage sera en écriture blanche sur fond noir, mais vous pourrez bien sur changer ces parametres (PAPER et INK). Le bouton RESET de l'unité de disquette produisant également un RESET (partiel) de l'ordinateur, vous aurez l'écran "INSERT SYSTEM DISK" chaque fois que vous appuierez sur ce bouton sans avoir une disquette dans le lecteur.

Le lecteur MASTER (A) a un cable plat qui sort de l'arrière du boitier, et qui va sur l'ORIC. Les lecteurs esclaves sont livrés avec un cable plat et une prise femelle sur la face arrière, comme l'unité MASTER, mais celle-ci constitue la prise d'entrée. Le cable plat comporte une prise AMPHENOL à chaque extrémité et une prise mâle sertie sur le cable: Pour brancher le lecteur B, on relie la prise arrière du lecteur B à la prise arrière du lecteur A . Pour brancher les suivants (C et D), on relie l'entrée du lecteur suivant à la prise mâle du cable du lecteur précédent. Le schéma ci-dessous est plus explicite sur ce sujet.

Les alimentations fournies peuvent suffire pour une unité centrale et 2 unités de disquettes. Pour un plus grand nombre de lecteurs, il faudra une alimentation supplémentaire, suivant le schéma ci-dessous.



## C / COMMENT SE PRESENTE LA DISQUETTE 3" ?

Une disquette se présente sous la forme d'une feuille de plastique circulaire, percée d'un trou au centre, et logée dans une pochette plastique destinée à protéger la disquette. La pochette est percée de plusieurs fenêtres destinées à permettre le fonctionnement (rotation, positionnement de la tête d'enregistrement-lecture etc..) sans sortir la disquette de la pochette.

Le standard de disquettes le plus utilisé sur les micro-ordinateurs est le standard 5"1/4 (13 cm). Nous avons retenu pour l'ORIC le nouveau standard de disquettes 3", pour les raisons exposées au paragraphe précédent.

La disquette mesure 71mm de diamètre. Elle est logée dans un boitier **rigide** de 8 x 10 cm qui en facilite la manipulation et le stockage, et améliore sa protection.

Que trouve-t-on sur une disquette ? Pour stocker des informations (transmises entre l'ordinateur et le lecteur sous forme de "trains" d'octets), et surtout pour pouvoir les retrouver rapidement, la surface magnétique circulaire est divisée en pistes concentriques (jusqu'à 44 par face) . Chaque piste est divisée en secteurs et chaque secteur d'une piste peut stocker 256 octets. (Voir schéma en face)

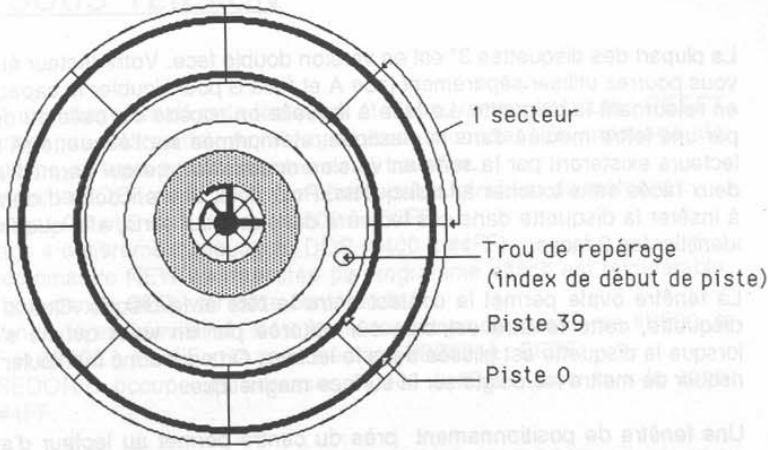
Bien sur, cette division de la disquette n'est pas physique ou visible comme sur un 33 tours. Elle est faite par la programmation des déplacements de la tête d'enregistrement devant le disque qui tourne à grande vitesse ( 300 tours/mn). Ce déplacement est commandé par le Système d'Exploitation du Disque (**SED**) ou Disk Operating System (**DOS**) en anglais.
On peut ainsi, par les instructions du DOS, modifier le nombre de pistes ou de secteurs de la disquette !

## PRESENTATION PHYSIQUE DE LA DISQUETTE 3"

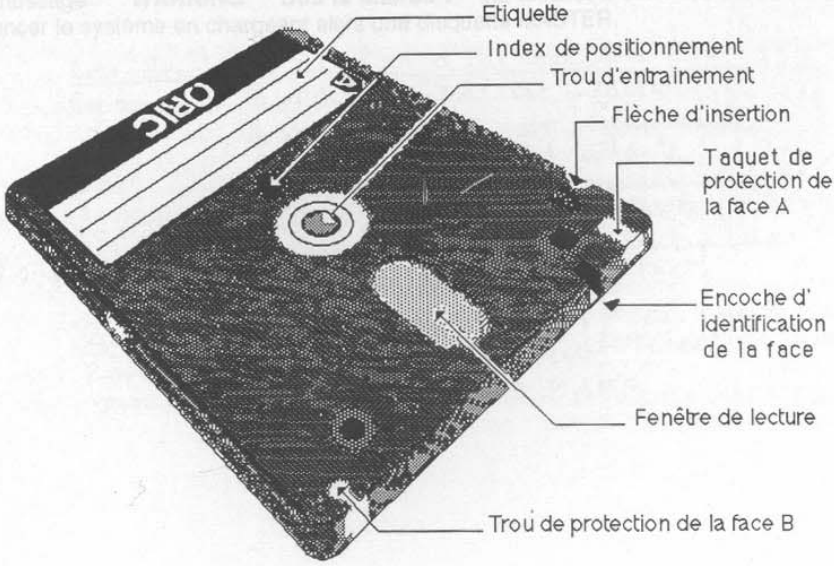
La disquette 3" est donc un boitier plastique rigide. Le côté qui porte l'étiquette est celui que l'on tient entre le pouce et l'index quand on insère la disquette dans la fente du lecteur. De l'autre côté, on trouve diverses encoches décrites sur le schéma ci-contre.

Les petits trous ronds qui peuvent être obturés par un volet coulissant blanc protègent en écriture (et bien sur en effacement) la face concernée. On pousse le volet pour dégager le trou et il devient impossible d'effacer la face correspondante de la disquette ou d'écrire dessus.

## Structure d'enregistrement sur le disque




## Anatomie de la disquette 3"



La plupart des disquettes 3" est en version double face. Votre lecteur étant simple face, vous pourrez utiliser séparément face A et face B pour doubler la capacité, simplement en retournant la disquette. La face à laquelle on accède est celle du dessus, indiquée par une lettre moulée dans le plastique, et imprimée sur l'étiquette (A ou B). Certains lecteurs existeront par la suite en version double-tête, ce qui permettra d'accéder aux deux faces sans toucher à la disquette. Pour cela, une encoche d'identification oblige à insérer la disquette dans ces lecteurs dans un seul sens, afin que le lecteur puisse identifier les 2 faces.

La fenêtre ovale permet le contact entre la tête et le disque. Quand on regarde la disquette, cette fenêtre est bien sûr obturée par un volet qui ne s'escamote que lorsque la disquette est glissée dans le lecteur. On peut donc manipuler le disque sans risquer de mettre les doigts sur la surface magnétique.

Une fenêtre de positionnement près du centre permet au lecteur d'avoir un repère physique de la position du disque, par passage du trou de repérage devant la fenêtre.

Stockez vos disquettes dans leur boite, et mettez-les à l'abri de la chaleur ou de l'humidité excessives, et des rayonnements magnétiques : les transfos d'alimentation, les moteurs (y compris celui du lecteur), les moniteurs et les téléviseurs sont des sources de rayonnements magnétiques importants : évitez d'en approcher vos disquettes, sous peine de les effacer ou de les endommager.

# D / MISE SOUS TENSION

Il suffit d'insérer une disquette dans le lecteur A et d'appuyer sur le bouton **RESET** derrière le lecteur de disquette. Après quelques secondes, le message de présentation s'affiche, et vous pouvez commencer à travailler.
Première remarque, le BOOT du SEDORIC n'affecte qu'un minimum de mémoire :

- Les pages 0,1 et 2 sont vidées (#0000 à #02FF).
- La page 4 entièrement allouée au DOS (#400 à #4FF).
- Une commande **NEW** est exécutée (le programme effacé est récupérable par la commande **OLD**, voir cette commande )
- L'écran texte est réinitialisé, c'est à dire les zones de caractères #B500 à #B7FF et #B900 à #BAFF et la zone d'écran ( #BB80 à BFDF).
- Le SEDORIC occupe les adresses #C000 à #FFFF,ainsi que la zone #400-#4FF.

Lors d'un RESET et avant de rendre la main au BASIC, le SEDORIC exécute une série d'instructions particulières à la disquette, qui peuvent d'ailleurs être modifiées. Il est donc possible d'inclure dans ces instructions d'initialisation un nom de fichier à exécuter, ce qui vous permet de personnaliser l'initialisation de votre disquette. Si aucune instruction d'initialisation n'existe, le DOS saute directement au BASIC.

Il existe deux types de disquettes crées par SEDORIC : les disquettes de type "MASTER", qui contiennent le DOS, et les disquettes de type "SLAVE" qui ne permettent de booter que lorsquele DOS est déjà en mémoire. Si tel n'est pas le cas, le message "**** WARNING ** Dos is altered !** " est affiché. Il est alors conseillé de relancer le système en chargeant alors une disquette MASTER.

Lors du boot ou lors d'un reset à froid
La zone : B400 à B4FF est intacte
B500 à B8EF est reinitialisée
B8F0 à B8FF est intacte
B900 à BAFF est reinitialisée
BB00 à BB7F est intacte
BB80 à BFDF est reinitialisée
BFE0 à BFFF est intacte

Reset à chaud Basic (Call #C003) ne touche pas la
Zone B900 à BB7F mais reset Teletral simple
brouille la zone B400 à BAFF

# E / COMMENT DEMARRER

Vous avez enfin ce lecteur de disquette tant attendu, et vous brulez d'envie de vous en servir, sans pour autant avaler tout de suite tout le manuel. Parcourez donc ce chapitre, il vous donnera envie d'aller plus loin dans la connaissance de votre nouvelle acquisition.

**Première opération** : si vous n'avez qu'un seul lecteur connecté, sautez ce paragraphe . Sinon, il va être nécessaire de signaler au SED qu'un deuxième lecteur est présent . Il faut déprotéger la disquette (descendre la languette en haut à gauche), remettre la disquette dans le lecteur, et frapper (pas trop fort ) **DTRACK A,,42** suivi de "RETURN" bien entendu (voir page 42 pour plus de détails).
Reprotéger la disquette et relancer ensuite le système en appuyant sur le bouton RESET du lecteur et tout est prêt pour la suite.

**Deuxième opération** : Il est vivement conseillé de faire une copie de votre disquette MASTER, afin d'éviter un accident fatal. Il suffit pour cela de frapper **BACKUP** (ou **BACKUP A TO B** si vous avez deux lecteurs) suivi de "RETURN".
Pressez une deuxième fois "RETURN", et répondez **"Y"** à la question **"Format target disc (Y/N)"** . Il suffit ensuite de se conformer aux instructions qui seront affichées.
**IMPORTANT** : La disquette MASTER doit rester protégée par la languette pour éviter de fausses manipulations.

**Troisième opération** : Prendre contact avec le SED.

Pour connaitre le contenu de la disquette, rien de plus simple : entrer **DIR** et c'est tout ! Si vous possedez un Atmos, appuyer simultanément sur les touches **Funct** et ] .

Pour charger un programme aparaissant au catalogue, il suffit de taper son nom. Par exemple, pour jouer au jeu **MARC.COM** , il suffit de frapper **MARC** suivi comme d'habitude de "RETURN" . Quand vous aurez fini de jouer , appuyer sur le bouton de **RESET** du lecteur de disquettes.

Pour sauver un programme BASIC, il suffit de frapper **SAVEU "ESSAI"** et une seconde après, c'est fini. Pour s'assurer que le programme a été bien sauvé, consulter le catalogue par **DIR** .

Pour rappeler un programme , il suffit de taper son nom. Par exemple, pour récupérer le programme sauvé tout à l'heure, il suffit de frapper **ESSAI** suivi de "RETURN" . Vous pouvez aussi taper **LOAD "ESSAI"** , ce qui conduira au même résultat .

Si vous voulez suprimer un fichier, tout est prévu : l'ordre **DEL vous rendra ce service** . Frapper **DEL "ESSAI"** par exemple .

Et puis après tout, ESSAI est un nom trop commun, trop simple, en un mot vulgaire . Il faut le changer immédiatement . L'ordre **REN** va vous tirer de ce mauvais pas . Taper par exemple **REN "ESSAI" TO "VIVEMOI"** et le tour est joué !
REM : les crises de narcissisme aigü se soignent très bien (un disquette SEDORIC matin et soir) .

Vous avez une disquette vierge qui ne demande qu'à subir les derniers outrages : agissez vite . Frapper **INIT** suivi de deux fois "RETURN" . **Insérez ensuite** la disquette à initialiser et répondez au questions (avec bienveillance ! ) . Une disquette MASTER est une disquette qui contient le SED, ce qui est plus pratique . répondez donc **Y** à la question **Master disc (Y/N)** .

C'est humain, vous n'avez pas encore tout à fait confiance dans ce nouveau jouet (pardon, périphérique) et vous vous dites que faire une copie de votre programme serait plus sécurisant, ce qui est vrai de toutes façons.
Entrez donc **COPY "VIVEMOI"** ( ou **COPY "VIVEMOI" TO B** si vous avez deux lecteurs ) et suivez les instructions.

Vous aurez probablement remarqué le message **"INSERT MASTER DISC IN DRIVE A"** . C'est tout simplement que le SED est très long, et que les concepteurs n'ont pas voulu trop empiéter sur la mémoire utilisateur. Certaines commandes font donc appel à des routines qui se trouvent sur une disquette de type "MASTER". Sauf dans le cas du INIT, si vous appelez plusieurs fois de suite la même commande, il ne sera demandé une disquette MASTER que la toute première fois .

Voilà ! Vous avez fait une brève connaissance avec le MICRODISC associé au SEDORIC . Mais une unité de disquette est bien plus qu'un super magnéto cassette , et nous vous invitons à découvrir le chapitre "GESTION DE FICHIERS". Et puis SEDORIC c'est aussi une soixantaine d'instructions suplémentaires qui enrichissent le BASIC de l'ORIC et pour cela, il est vivement conseillé de parcourir en entier le manuel pour avoir une vision d'ensemble des instructions . Bon courage !

# F / CONTENU DE LA DISQUETTE SEDORIC 1.0

La disquette **SEDORIC 1.0** contient plusieurs programmes ou utilitaires.

- Elle contient en tout premier lieu un système d'exploitation, qui n'apparait pas au catalogue, mais qui est bel et bien présent (la mention 'MASTER" du catalogue rassurera les sceptiques).

**M.A.R.C**

- Elle contient un jeu, un vrai, avec des missiles et tout et tout ,nommé **M.A.R.C.** Pour le charger, il suffit de taper MARC suivi de "RETURN" , et 2 secondes plus tard, à vous les envahisseurs !

**CONVERT**

- Vous qui possédiez auparavant un Oric DOS V1.1 ou un TDOS, tout d'abord bienvenue à SEDORIC ! Ensuite, vous pourrez facilement transférer vos programmes sous ces anciens ( et archaïques ) formats en des tous beaux fichiers SEDORIC.

- Il suffit pour cela de charger le programme **CONVERT.COM** , c'est à dire taper CONVERT suivi de "RETURN".

- CONVERT est un magnifique utilitaire , en ce sens que son usage est d'autant plus simple que CONVERT est puissant. Il suffit de se laisser guider par les différents menus. Dans tous les cas, on passe d'un choix à l'autre (un choix étant matérialisé par une barre en vidéo inverse) en appuyant sur la barre d'espace ou en se déplaçant avec les flèches du curseur. Un choix est validé en appuyant sur "RETURN".

- La touche **ESC** permet à tout instant de revenir au menu , et permet de revenir sous BASIC à partir du menu.

- On peut rappeler le CONVERT simplement par ]] , lorsqu'on est sorti par ESC, sans perdre la configuration courante, sous réserve que le CONVERT lui même ne soit pas altéré . Si tel est le cas, retaper CONVERT .

- Le menu présente 4 options différentes:

**SELECTION DES LECTEURS**

Cette option vous permettra de sélectioner l'unité de disquette qui contient la disquette à l'ancien format, et celle qui contient la disquette au format SEDORIC. Bien entendu, ces 2 lecteurs peuvent être le même, comme pour un simple **COPY.**
La touche **"RETURN"** fait la bascule entre le lecteur source et le lecteur cible, et les lecteurs sont sélectionnés par les **flèches** ou la barre d'**espace.**

**INITIALISER LA DISQUETTE CIBLE**

Cette option permet d'initialiser une disquette au format SEDORIC, au même titre que l'aurait fait le commande INIT. Les disquettes crées ser ont du type **"slave"** .

**SELECTION DU DOS**
Cette option permet de sélectionner le type de la disquette source, soit le format TDOS, soit le format Oric V1.1 ou XL DOS.

**CONVERSION DE FICHIERS**
C'est l'option qui permet le transfert effectif des fichiers.
Il suffit d'insérer la disquette source dans le lecteur concerné.
Pour charger le catalogue de la disquette source, il faut frapper **"DEL"**. Seuls les 80 premiers fichiers de la disquette sont copiables (!) .
La sélection des fichiers à copier est très simple: il suffit de se déplacer avec les flèches haut et bas sur le fichier concerné et d'appuyer sur la barre d'espace. Une étoile apparait alors à côté du nom du fichier, indiquant que le fichier doit être copié. Lorsque la sélection est finie, il suffit de presser "RETURN" pour lancer la conversion . Si une erreur survient, on a le choix entre frapper une touche quelconque, auquel cas le programme reprendra l'exécution au fichier qui a déclenché l'erreur, ou appuyer sur ESC, le programme continuant alors au fichier suivant .

**RESTRICTIONS D'UTILISATION**
- Sous TDOS, les fichiers de type **.SYS, .ARY** et **.DAT** ne sont pas copiés.
- SousTDOS, les caractères dans les noms de fichier se revélant invalides pour SEDORIC (-,* etc...) sont remplacés par des "Z" .
- Les fichiers de données sous XL DOS ou dos V1.1 sont copiés mais non convertis .

**ADRESSE**
La disquette contient aussi un programme de démonstration de fichier à accès direct. Ce programme se nomme **ADRESSE.COM** et il suffit de taper ADRESSE suivi de "RETURN" pour lancer le programme . Ce programme est accompagné des fichiers **ADRESSE.WIN** qui contient le masque de saisie et du fichier **ADRESSE.DAT** qui contient les données des fiches.

**ALPHA**
Le programme **ALPHA.COM** (frapper ALPHA et "RETURN") permet de classer par ordre alphabétique le catalogue d'une disquette. Il suffit de se conformer aux instructions.'

**STAT**
Le programme **STAT.COM** permet de faire des statistiques sur les instructions les plus employées dans un programme BASIC . Pour l'exécuter taper **STAT,J** suivi de **RUN 64000.**

**SECLIBRE**
Le programme **SECLIBRE.COM** permet de visualiser l'occupation physique de la disquette .

**GAMEINIT**
La disquette contient enfin le programme **GAMEINIT.COM** qui permet de créer des disquettes de jeux par exemple, avec un SED réduit ne comprenant que trois instructions: !LOAD , !DIR et chargement direct (nom de fichier précédé du point d'exclamation), le tout avec la syntaxe habituelle.
L'intérêt est que le SED n'occuppe environ que17 secteurs contre une centaine pour une disquette MASTER normale.

La syntaxe de la commande GAMEINIT est très proche de la syntaxe de la commande INIT, à ceci près qu'il faut placer le caractère ": " après la commande.
Par exemple: ***GAMEINIT:A,19,44 .***

**VERSION**
Ce petit programme permet de connaitre la version de votre SEDORIC 1.0, sous la forme SEDORIC 1.0xx . En effet SEDORIC est constament amélioré (tout en assurant bien entendu la totale compatibilité avec les versions précédentes) .

**ROMATMOS ET ROMORIC**
Ces deux routines chargent la ROM V1.1 ou V1.0 en RAM overlay, permetant ainsi de disposer de la ROM Atmos sur un Oric-1, ou de la ROM Oric-1 sur un ATMOS. Bien entendu, il est alors impossible de se servir du DOS.

## G / SEDORIC : UNE TOUCHE DE GENIE !

Le SEDORIC apporte une caractéristique importante à l'ORIC ATMOS (Atmos seulement) : l'utilisateur dispose de 2 claviers supplémentaires accessible en appuyant sur FUNCT et la touche ou FUNCT + SHIFT et la touche. L'assignation des ces touches peut être changée à tout moment grâce aux commandes **KEYUSE** et **KEYDEF** (voir le détail de ces instrucions à la rubrique correspondante). Par exemple, taper *FUNCT* et \ provoque l'exécution de l'instruction LIST. Quelle facilité d'emploi !

De même, vous pouvez obtenir que vos lignes de programme soient automatiquement numérotées en tapant *FUNCT* et *RETURN* , ce qui provoque la numérotation automatique des lignes de programme .

Enfin, appuyer sur *FUNCT* et *DEL* produit le même résultat que DEL, à ceci près que le caractère n'est pas effacé de l'écran, mais seulement de la mémoire-tampon du clavier, ce qui est bien pratique lorsque l'on s'aperçoit d''une erreur quelques caractères avant : en mode éditeur d'écran, cela permet de revenir en arrière sans effacer les caractères corrects de l'écran; on peut ainsi les valider à nouveau avec CTRL A et on n'a pas à les retaper.

Un nouveau caractère de contrôle a été ajouté : **CTRL P** permet de stopper le clignotement du curseur. C'est une bascule ON/OFF : **CTRL P** et le curseur clignote, **CTRL P** et il ne clignote plus. On peut y accéder par programme (Bit 2 de la variable système # 26A ou PRINT CHR$(16);).

Quelques aménagements permettent d'éviter les "plantages" inopportuns :

- Il est possible d'appuyer sur le bouton **RESET** de l'ORIC à tout moment, même pendant les accès disque. Il vaut mieux toutefois éviter les **RESET** pendant les écritures.
- Le SEDORIC gère le Break (interruption soft) et affiche " BREAK ON BYTE #xxxx " si un Break est détecté. Cela évite 99% des "plantages". Si vous n'êtes pas convaincu, essayez donc sans le DOS de taper CALL # 500 ...!
- La touche ESC permet de sortir prématurément des commandes interactives telles que **DIR, DEL, INIT** etc...

Le SEDORIC occupe environ 20 kOctets de mémoire en tout. Pour ne pas empiéter sur le zone utilisateur, certaines fonctions secondaires ou particulières ont été séparées du bloc des commandes principales du SEDORIC : Une zone dans le DOS, appelée **SWAP**, accueille ces commandes au fur et à mesure de leur utilisation. Ces commandes sont regroupées en blocs de 1 kOctets. Le numéro du bloc est donné dans la table alphabetique des commandes en annexe 1. Lors du premier appel de la commande , il sera demandé d'insérer une disquette MASTER pour charger le bloc correspondant.
La commande sera alors accessible comme toutes les autres jusqu'au prochain BOOT, ou jusqu'à l'appel d'une commande figurant dans un autre bloc, ou l'appel de la commande **WINDOW**.

## H / CONVENTIONS ET NOTATIONS

Afin d'en faciliter la lecture , les mêmes notations seront employées tout au long de ce manuel. Toutes les commandes auront donc la même présentation :

**NOM DE COMMANDE** en gros caractères gras et SYNTAXE GENERALE en gros caractères maigres.

- Les descriptifs de la commande et les remarques sont précédés de tirets.

Les options facultatives sont entre parenthèses. Si elles ne sont pas précisées, elles prennent des valeurs par défaut, explicitées dans le texte.

Tous les exemples sont en ***ITALIQUES.***

Quelques abréviations seront fréquemment utilisées :

- Lecteur : Le paramètre est un nom de lecteur ( A ou B ou C ou D) . S'il n'y a rien, c'est le lecteur courant qui est considéré.

Exemple : ***INIT et INIT A*** sont équivalents si A est le lecteur courant.

- NF ou Nom de Fichier : Cette abréviation désigne un nom de fichier quelconque mais valable (voir le paragraphe NOM DE FICHIER). Il ne peut être omis, ni ne comporter de "JOKER"

Exemple : ***LOAD NF***

- NFA ou Nom de Fichier Ambigu : Cette abréviation désigne un nom de fichier (voir ce paragraphe), mais qui peut comporter un ou plusieurs JOKERS. Il peut donc être omis, ou réduit à un simple nom de lecteur.

Exemple : ***DIR NFA***

- EN ou Expression Numérique: Cette abréviation désigne une expression numérique, avec les restrictions que celà implique (voir ce chapitre dans le Manuel de l'ORIC).

Exemple : ***PRINT EN***

- EA ou Expression Alphanumérique: Cette abréviation désigne une expression alphanumérique, avec les restrictions que celà implique (voir ce chapitre dans le Manuel de l'ORIC).

Exemple : ***SEARCH EA***

- VN ou Variable Numérique : Ceci représente un nom de variable numérique, à ne pas confondre avec une expression numérique.

Exemple : ***INPUT (EA;) VN***

- VA ou Variable alphanumerique : même remarque que pour les variables numériques.

Exemple : ***TKEN VA***

- NL ou Numéro Logique : c'est le numéro du tampon utilisé dans la gestion des fichiers

Exemple : ***REWIND NL***

- NC ou Nom de Champ : utilisé en gestion de fichiers.

Exemple : ***LSET NC<EA***

- La touche **ESC** permet de sortir prématurément de toutes les situations où une réponse est attendue de l'utilisateur.

# I / ENVIRONNEMENT GENERAL

Tout a été conçu dans le SEDORIC pour faciliter la tâche de l'utilisateur : les syntaxes sont simples et efficaces, les touches de fonctions donnent accès directement aux commandes, et , surtout, le SEDORIC est un DOS extrêmement rapide : il est impossible de faire plus rapide avec un disque 3". Il faut signaler que les sauvegardes sont aussi rapides que les chargements.

Une telle optimisation a conduit à s'éloigner du format du DOS V1.1 ou du TDOS. Les disquettes créées avec ces DOS ne sont donc pas directement exploitables par le SEDORIC. Le fichier **CONVERT**contenu sur la disquette permet de convertir tous vos anciens fichiers au nouveau format. Les commandes **SEEK** et **CHANGE** faciliteront quelques adaptations syntaxiques.

La syntaxe adoptée est la plus proche des ordres standard du BASIC : par exemple, les ordres **SAVE** et **CSAVE** ont la même syntaxe. Les évaluations de variables sont effectuées par les routines du BASIC, et ne sont donc limitées que par les contraintes du BASIC lui-même.

Tous les mots-clés du SEDORIC peuvent être frappés aussi bien en MAJUSCULES qu'en minuscules, y compris les noms de fichiers, ce qui rend inutile l'emploi de minuscules dans les noms de fichiers.

**ATTENTION : les mots-clés du BASIC doivent toujours être frappés en majuscules.**

Chapitre 2

# Les noms de Fichiers

Manuel d'utilisation du MICRODISC ORIC

# LES NOMS DE FICHIERS

Un nom de fichier se définit comme suit :

- Un nom de fichier est généralement une chaine de caractères. Il doit donc être entouré de guillemets. Il peut tout aussi bien être une expression alphanumérique . Une exception à cette règle : le chargement direct (pour plus de détails, consulter ce chapitre).

- Un nom de fichier comporte au plus 3 parties qui sont :

**1/ Le lecteur concerné.**

Les lecteurs sont désignés par des lettres de A à D. Lorsque le lecteur est précisé, la lettre correspondante doit être suivie d'un tiret qui la sépare de la partie suivante qui est le nom.
Lorsque le lecteur n'est pas précisé, c'est la valeur par défaut (ou valeur courante) fixée par l'utilisateur qui sera retenue.

**2/ Le nom proprement dit.**

Il comporte au plus 9 caractères alphanumériques. C'est la seule partie du nom de fichier qui soit obligatoire. Seuls les caractères alphabétiques et numériques sont acceptés. Dans le cas contraire (symboles, signes de ponctuation etc..), un message "INVALID FILE NAME ERROR" sera affiché.

**3/ L'extension .**

Lorsque celle-ci est spécifiée, elle doit être séparée du nom par un point. Elle comporte au plus 3 caractères alphanumériques. Elle permet généralement de distinguer le type de fichier (PRG, SEQ, DIR etc..) mais l'utilisateur peut choisir toute autre signification.
Lorsque l'extension n'est pas précisée, elle prend une valeur par défaut, définie par l'utilisateur (Voir explication de la commande EXT). A l'initialisation, cette extension est ".COM"

**Une remarque valable pour les extensions comme pour les noms de fichiers** : les minuscules seront automatiquement transformées en majuscules après la première exécution. Dans le catalogue, tous les noms de fichiers sont en fait automatiquement convertis en majuscules.

Exemple de noms valides :
***"ESSAI "*** (équivalent à ESSAI.COM ou essai.com ou A-ESSAI si A est le lecteur par défaut, ou ... eSSai.Com !)
***"ABCD12340.ABC"***
***"ORIC.DOS"***
***"AREU.H"***
***"B-TOTO.COM"***

Exemple de noms invalides :

***"ESSAI+1"*** ( + n'est pas un carctère alphanumérique ).

***"ABCDEFGHIL"*** ( nom de fichier trop long ).

***"ESSAI .TOUT"*** (extension trop longue ).

***"E-BOF"*** (le tiret n'est pas un caractère alphanumérique. Il peut donc identifier E comme un numéro de lecteur, mais E est un nom de lecteur invalide ( 4 lecteurs seulement de A à D) !).

**"H-EXPERIENCE.! BOF"** comporte 4 erreurs ! Quel est l'age du programmeur ?

Une autre caractéristique très intéressante : un nom de fichier peut aussi être ambigu, c'est à dire comporter des caractères spéciaux appelés **JOKERS** qui permettent, dans certaines instructions, de désigner plusieurs noms de fichiers à la fois. La syntaxe de ces instructions précise alors NFA (Nom de Fichier Ambigu)..

Cela est valable pour des instructions telles que **DIR, DEL** etc.. qui peuvent concerner plusieurs fichiers simultanément ( **DEL** " * . BAS " détruit tous les fichiers d'extension .BAS, **DIR** " M * " liste tous les fichiers dont le nom commence par M etc.. ). Les caractères Jokers sont le point d'interrogation **?** et l'astérisque ***** . Le point d'interrogation remplace n'importe quelle lettre, et l'astérisque remplace toutes les lettres d'un nom jusqu'à la fin ; cela équivaut à remplir le nom avec autant de ??? que de lettres à remplacer.

Exemple : soit le catalogue suivant :

| | | | |
|---|---|---|---|
| ***TOTO*** | ***. BAS*** | ***RIRI*** | ***. BAS*** |
| ***PROG1X*** | ***. COM*** | ***DESSIN*** | ***. SCR*** |
| ***DEBUT*** | ***. BAS*** | ***ESSAI*** | ***. COM*** |
| ***PROG2X*** | ***. COM*** | ***PROG3X*** | ***. COM*** |

***DIR " *. BAS"*** listera les fichiers TOTO . BAS, RIRI . BAS et DEBUT . BAS.
***DIR " DE* . * "*** ( ou DIR "DE???????.??? ") listera les fichiers DEBUT . BAS et DESSIN . SCR.
***DIR " PROG?X.COM "*** listera les fichiers PROG1X.COM, PROG2X.COM et PROG3X .COM.

Comme l'astérisque remplit en fait de ??? jusqu'à la fin du nom ou de l'extension, il ne doit pas être suivi de caractères mais d'un point, d'un guillement ou de rien.
D'autre part, lorsqu'un nom de fichier ambigu (NFA) doit être précisé, il est toujours possible de ne pas préciser de nom de fichier, ou de ne préciser que le numéro de lecteur. Dans ce cas, toute la disquette est concernée car le nom pris par défaut est "*.*", donc tous les fichiers.
Exemple : ***DIR, DIR A , DIR"A-*.*"*** et ***DIR "*.*"*** ont exactement la même signification si A est le lecteur courant.

## LECTEUR PAR DEFAUT

(lecteur) -

- Le n° de lecteur est facultatif dans les noms de fichiers. s'il n'est pas précisé, le lecteur courant est pris.

- Pour définir le lecteur courant, il faut taper le nom du lecteur suivi d'un tiret.

- Dans ce manuel, les termes de LECTEUR COURANT et LECTEUR PAR DEFAUT ont été employés indifféremment.

Exemple : ***B- (nom de fichier)***

## **EXT** EA

- Spécification de l'extension par défaut. La chaine doit comporter 3 caractères alphanumériques.

Exemple : ***EXT "BAS"***

## EXT ?

- Affiche l'extension par défaut.

## EXT

- Remet l'extension par défaut à .COM

Chapitre 3

# Les instructions de travail sur le disque

Manuel d'utilisation du MICRODISC ORIC

# A/ CATALOGUE DE LA DISQUETTE

## DIR NFA

- Donne le catalogue de la disquette.

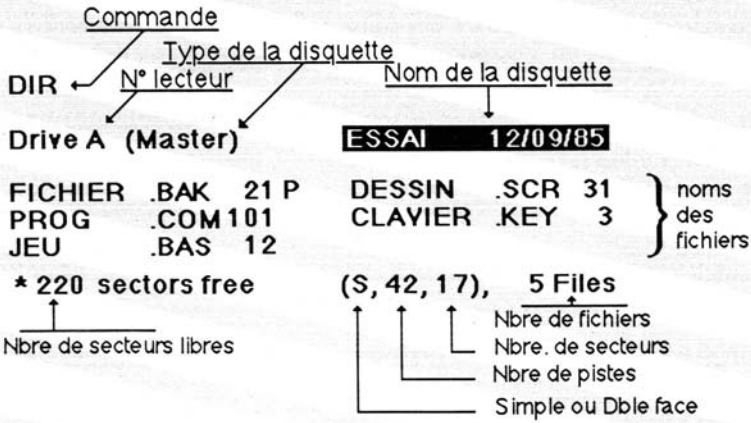
- Le catalogue peut être interrompu par la pression de n'importe quelle touche.

Pour continuer : taper SPACE
Pour sortir definitivement du catalogue : Taper ESC.

- Chaque nom de fichier (ou de programme) est donné avec le nombre de secteurs qu'il occupe. Si ce nombre est suivi de la lettre **"P"**, cela signifie que le fichier en question est protégé en écriture.

- Le nombre de fichiers n'est limité que par le nombre de secteurs d'une disquette ( de 336 à 817 maxi selon formatage).

Exemple :
***DIR***
***DIR " * . BAK "***
***DIR C***




## LDIR NFA

- Même instruction que **DIR** mais avec sortie sur l'imprimante.

## B/ CHARGER UN PROGRAMME

## **LOAD** NF (, A EN) ( , V ) ( , J ) ( , N )

- Charge le fichier spécifié.

  **, A EN** : charge le fichier à l'adresse précisée par l'expression numérique EN. Si c'est un fichier "mergé" (voir SAVEM), seule la première partie du fichier est affectée.

  **, J** : Charge le fichier à la fin du programme BASIC en mémoire. Pour l'exécution, les numéros de lignes doivent être cohérents.

  **, N** : empêche l'exécution d'un programme devant s'exécuter automatiquement.

- Si le fichier n'est pas trouvé, un message "FILE NOT FOUND" apparait.

- Seuls les fichiers sauvegardés par **SAVE, ESAVE** ou **CREATEW** peuvent être chargés par l'instruction LOAD. Pour les autres (fichiers de données créés avec **OPEN** etc...), il est obligatoire d'ajouter l'option **,V** (sous peine de déclencher un message d'erreur "FILE TYPE MISMATCH") :

  **, V** : visualise le "status" du fichier, mais le fichier n'est pas chargé. Cette option peut être utilisée en même temps que **, J** ou **, A** mais pas en même temps que l'option **, N.**

  Le STATUS est un code qui indique la nature et l'etat d'un fichier.

  Si le fichier a été "mergé", plusieurs lignes de status apparaissent, correspondant aux différentes parties composant le fichier. De plus, seul le Status du premier fichier détermine le mode d'exécution de celui-ci. Les variables **ST, ED, FT** et **EX** sont chargées respectivement avec l'adresse de début, l'adresse de fin, le Status du fichier et l'adresse d'exécution.

Le status se présente sous la forme suivante : **DDDD FFFF SS EEEE** (Les valeurs sont indiquées en Hexadécimal)

**DDDD** est l'adresse de début de fichier . Pour un fichier direct, c'est le nombre de fiches.

**FFFF** est l'adresse de fin de fichier . Pour un fichier direct, c'est la longueur d'une fiche.

**SS** est le code d'état du fichier. Il a une signification binaire bien précise, donnée en annexe. Par exemple, **80** correspond à un fichier BASIC, **81** à un fichier BASIC auto, **40** à du langage machine ou à un bloc-mémoire, **41** à du langage machine auto.

**EEEE** : Si le fichier est un programme en assembleur, EEEE est l'adresse d'exécution. Elle est de **0** par défaut.

Exemples : ***LOAD " ESSAI "***
***LOAD " B-ROUTINES ", J , V***
***LOAD A$ + ". TXT ", A # 4000***

## CHARGEMENT DIRECT.

## **NF** (,A EN) (,N) (,J) (,V)

En réalité, l'instruction **LOAD** n'est réellement utile que lorsque le nom de fichier est une variable alphanumérique (***LOAD A$*** par exemple).

- Une des caractéristique intéresantes du DOS est qu'il permet de charger directement un fichier en tapant simplement son nom (voir le chapitre environnement général). Le nom peut être suivi des mêmes options que **LOAD.**

- La syntaxe du nom est un peu particulière : elle n'admet ni guillemet, ni variable, bref, **c'est un mot-clé**! Ceci explique que l'extension soit **.COM** ´pour Commande.
Il convient de se méfier : si le nom de fichier commence par un chiffre, le BASIC l'interprètera comme le début d'une ligne de programme ! De même, si le nom commence par un mot-clé du BASIC ou du DOS, il sera interprété comme tel.
Exemple : un fichier appelé ***LOADER*** sera interprété en chargement direct comme ***LOAD ER*** , donc génèrera un message d'erreur, ER étant interprété comme une variable numérique alors que le DOS attent une chaine comme nom de fichier.

- Pour pallier ces inconvénients, il suffit de placer devant de tels noms le numéro du lecteur qui est normalement facultatif. Une autre solution consiste à insérer dans le nom un espace, ce qui fait qu'il ne pourra être reconnu comme mot-clé.
Exemple : ***TO*** étant un mot-clé du BASIC, ***TOTO*** provoquera un message d'erreur alors que ***T OTO*** sera interprété comme un fichier à charger.

- Ces restrictions mises à part, le nom de fichier a une syntaxe normale : Numéro de lecteur, nom de fichier et extension. En mode **QUIT** (voir ce mot-clé), le ! est obligatoire devant le nom.

- Après l'exécution de l'option " , V ", aussi bien en mode direct que après un **LOAD**, les variables ST, ED, EX et FT contiennent respectivement l' adresse de début, l'adresse de fin, l'adresse d'exécution et le type de fichier chargé..

Exemple : ***ESSAI*** charge le programme "ESSAI"
***B-DESSIN.SCR,A#1000*** charge à l'adresse #1000 l'écran "DESSIN" situé sur la disquette dans le lecteur B.
***ROUTINES.BAK,V*** met à jour 4 variables avec les adresses et visualise le Status correspondant.

## **SEARCH** NFA

Vérifie si un fichier existe sur la disquette et affecte la réponse à une variable EF.

- Si le fichier est trouvé, la variable EF prend la valeur 1, sinon la valeur 0.

- Les vraies valeurs logiques sont **0** (faux) et **-1** (vrai). Il faut donc utiliser -EF comme valeur dans les expressions logiques.

- Des noms de fichiers ambigus peuvent être utilisés. Dans ce cas, EF vaut 1 s'il existe au moins 1 fichier correspondant, et 0 dans le cas contraire.

EXEMPLE :

```
10  SEARCH "ESSAI"
20  IF EF=1 THEN LOAD "ESSAI" ELSE PRINT"Il n'y
a pas de fichier à ce nom"
```

## C/ SAUVEGARDER UN FICHIER

**SAVE** NF (,A EN) (,E EN) ( ,T EN) (, AUTO)
**SAVEO** " " " " "
**SAVEM** " " " " "
**SAVEU** " " " " "

- Sauve un fichier de nom NF sur la disquette.

- Si la disquette est protégée par la languette, la sauvegarde est arrêtée par un message d'erreur "WRITE PROTECTED".

Ces 4 instruction **SAVE** ne diffèrent en fait que par leur comportement si un fichier de nom **NF** existe déjà sur la disquette.

- La syntaxe est la même que pour un CSAVE : Si aucune option n'est précisée, le fichier sauvegardé sera le programme BASIC courant. Si **,A** ou **,E** sont précisés, il s'agira d'un bloc-mémoire commençant à l'adresse A et finissant à l'adresse E.

- Comme pour les cassettes, c'est l'état (Status) donné à la sauvegarde qui déterminera si le fichier doit être ou non exécuté. Cet état peut facilement être modifié (voir la commande **STATUS** ).

- **,T** (EN) permet de spécifier une adresse d'exécution différente du début du programme. Bien sur, cette option ne sert à rien pour un programme BASIC.
incompatibilité entre ,T et ,Auto

- Voici la différence entre ces diverses instructions de sauvegarde : si le fichier n'existe pas, tous ces ordres sont équivalents et le fichier choisi est sauvegardé sous le nom choisi. S'il existe déja un nom de fichier identique sur la disquette, voici ce qui se passe :

  - dans le cas d'un **SAVE**, un message d'erreur **"FILE ALREADY EXISTS"** apparait et la sauvegarde est arrêtée.
  - dans le cas d'un **SAVEO**, le fichier existant est effacé et remplacé par le nouveau, sauf s'il était protégé en écriture, auquel cas la sauvegarde est arrêtée et le message "xxxxxx IS PROTECTED" apparait.
  - dans le cas d'un **SAVEM**, le fichier sauvegardé est **ajouté** à la suite du fichier existant pour n'en former qu'un.
  **ATTENTION** : cette "fusion" n'est qu'une juxtaposition et n'a rien à voir avec l'option ",J" d'un **LOAD**, par exemple : simplement, les 2 fichiers sont stockés sous le même nom et seront chargés ensemble.

Exemple : ***SAVE "ESSAI" , AUTO***
***SAVEM "DESSIN.SCR",A#9800, E#BFDF***

- **SAVEU** : le fichier existant prend une extension ".BAK" (pour Backup) et devient une ancienne version. Attention : si un .BAK existait déja (version antérieure), c'est celle-ci qui sera effacée, de façon à ne pas avoir 2 fichiers de même nom sur la disquette.

- C'est toujours la dernière version et la version précédente (extension .BAK) qui se trouvent sur la disquette après un **SAVEU.** Cette instruction permet donc une mise à jour rapide et en toute sécurité d'un programme en cours d'élaboration.

- Il est interdit de sauver avec **SAVEU** un nom d'extension .BAK.

- **Attention :** Le **SAVEU** commence d'abord par détruire l'éventuel .BAK puis fait le changement de nom, et sauve enfin le fichier. S'il n'y a pas assez de place sur la disquette pour sauvegarder le nouveau fichier, la dernière aura pour extension .BAK;

Exemple : mise au point du fichier Basic "TEST"

| Instruction | Catalogue | N° de la version |
|---|---|---|
| **SAVEU "TEST"** | **TEST.COM** | **1** |
| **SAVEU "TEST"** | **TEST.BAK** | **1** |
| | **TEST.COM** | **2** |
| **SAVEU "TEST"** | **TEST.COM** | **3** |
| | **TEST.BAK** | **2** |

## **ESAVE** NF

- Sauve l'écran dans le mode courant (LORES ou HIRES). Il faut prendre garde de charger l'écran dans le même mode d'écran que lors de la sauvegarde.

- Equivaut à **SAVEU NF, A#BB80, E#BFDF** en mode Text.
et **SAVEU NF, A#A000, E#BF3F** en mode HIRES.

Exemple : ***ESAVE "DESSIN . HGR"***

## D / DETRUIRE UN FICHIER

## **DEL** NFA

- Efface un fichier de nom NF

- Lorsque des caractères "Jokers" sont indiqués, il est demandé une confirmation pour chaque fichier. Taper alors Y pour le détruire, N pour le conserver et ESC pour sortir de la procédure. Si le fichier est protégé en écriture, il ne pourra en aucun cas être effacé.

- Vous constaterez parfois un temps mort après la destruction d'un fichier. En fait, le DOS réorganise le catalogue pour que les fichiers occupent le moins de place possible sur la disquette. Il y a 15 fichiers par secteur de catalogue. Il y aura donc réorganisation si on passe d'un nombre total de fichiers de 16 à 15, de 31 à 30 etc..

Exemple : ***DEL"* . BAS"*** détruit tous les fichiers qui ont pour extension .BAS.
***DEL "ESSAI .COM"*** détruit le fichier ESSAI.
***DEL*** efface (en demandant confirmation) tous les fichiers de la disquette
***DEL B***

## **DESTROY** NFA

**Attention, commande dangereuse** : fonctionne exactement comme **DEL,** mais sans vous demander de confirmation ! Toutefois, les fichiers protégés en écriture seront épargnés.

- Rappel : **DESTROY** équivaut à **DESTROY "*.*",** c'est à dire efface tout sur la disquette !

- Si vous avez fait un geste malencontreux, n'hésitez pas à appuyer sur le bouton "RESET" de l'ORIC, quelques fichiers seront alors épargnés.

Exemple : ***DESTROY "* . BAK"***

## **DELBAK** lecteur

Détruit tous les fichiers d'extension .BAK dans le lecteur indiqué en option.

- équivaut à **DESTROY "* . BAK"**

Exemple : ***DELBAK***
***DELBAK B***

## E/ MODIFICATION DES FICHIERS

### **REN** $NFA_A$ **TO** $NFA_N$

- Changement d'un nom de fichier . Le nouveau nom ($NF_N$) et l'ancien ($NF_A$) sont affichés. Si le nouveau existe déjà, un message **"ALREADY EXISTS"** est affiché.

- Le nom de fichier est ambigu, c'est à dire que les Jokers sont acceptés. Ils doivent alors être à la même place dans l'ancien et dans le nouveau nom, sous peine de déclencher le message d'erreur **"INVALID FILE NAME".**

Exemple : ***REN "TOULON. DAT" TO "MARSEILLE.COM"***
***REN "* . COM" TO "* . BAS"***
changera par exemple ***TOTO.COM*** en ***TOTO.BAS***, ainsi que tous les fichiers d'extension ***.COM.***
***REN "PROG?" TO "TEST?"***

### **STATUS** NF (,A EN) (,T EN) (,AUTO)

Permet de changer le Status d'un fichier sans avoir à le charger puis à le sauvegarder. La syntaxe est assez proche de celle de **SAVE.**

- Si aucune option n'est précisée, le fichier sera forcément en "non AUTO", c'est à dire qu'il ne s'exécutera pas automatiquement.

Les options sont les suivantes :

**,A EN :** change l'adresse de chargement du fichier. La nouvelle adresse de fin sera recalculée par le DOS.

**,T EN :** Le fichier est forcé en AUTO, et doit s'exécuter à l' adresse précisée par EN. Pour les fichiers BASIC cette commande a le même effet que l'option **,AUTO.**

**,AUTO :** Le fichier est forcé en exécution automatique. Si c'est un fichier binaire, l'adresse de lancement sera celle du début du fichier.

- Aucune commande ne permet de changer l'octet d'état (STATUS BYTE) du fichier ( BASIC ⇒ Langage Machine par exemple).

- Dans le cas de fichiers groupés par la commande **SAVEM** ou **COPYM**, la commande n'affecte que le premier fichier. C'est aussi lui seul qui sera pris en compte lors du chargement.

Exemple : ***STATUS "ESSAI"***
***STATUS "TOTO", AUTO***
***STATUS "B-TALKTALJ", A#4000, AUTO***
***STATUS "CRAC.SFT",T#5000***

## **PROT** NFA

- Protège les fichiers contre l'effacement explicite (**DEL, DESTROY**) ou implicite (**SAVEO**). Un **P** apparait à côté du nom dans le catalogue.

Exemple : ***PROT "ESSAI"***

## **UNPROT** NFA

- Déverrouille la protection des fichiers. Le **P** en face du nom de fichier disparait du catalogue.

Exemple : ***UNPROT "*.COM"***
***UNPROT B***

# F / INITIALISATION ET COPIE DE DISQUETTE

## **INIT** lecteur (,nbre de secteurs par piste, nbre de pistes, S ou D)

Permet d'initialiser et de formater des disquettes.

- L'écran affiche les indications à suivre. ESC permet de sortir prématurément de la procédure d'initialisation.

- On peut éventuellement préciser le nombre de secteurs par piste, le nombre de pistes par face et le caractère double ou simple face. Ces parametres prennent les valeurs par défaut suivantes :

Nombre de secteurs : **17 secteurs par piste**
Nombre de pistes : le nombre précisé par **TRACK**
Nombre de faces : le nombre précisé par **TRACK.**

- Les contraintes de formatage sont les suivantes :
Au moins 21 pistes par face, et au plus 99 pistes.
16, 17, 18 ou 19 secteurs par piste.
Le nombre total de secteurs (Nbre de secteurs x Nbre de pistes x éventuellement par 2 si la disquette est double face) ne peut excéder 1920. doit être inférieur.
Si l'une de ces contraintes n'est pas respectée, un message **"ILLEGAL QUANTITY ERROR"** apparaitra à l'affichage.

- Les formatages en 16 et 17 secteurs (format IBM) sont très fiables. Ceux en 18 secteurs le sont moins. Le formattage en 19 secteur augmente la capacité, mais diminue encore la fiabilité, et est très lent en écriture. La lecture est toujours à la vitesse maximale. (15 kOctet par seconde environ).

Plusieurs renseignements seront demandés :
- **Format (Y/N)** : si oui, la disquette sera préalablement formatée.
- **Name** vous demande quel nom vous voulez donner à la disquette.
- **Init Statement** : il faudra rentrer l'instruction ou la suite d'instructions qui sera exécutée lors de l'initialisation. (par exemple , PRINT *"BONJOUR"* etc..). On peut bien entendu ne rien entrer, l'ordinateur rendant alors immédiatement la main au BASIC.
- **Master Disk (Y/N)** : s'informe du type de la disquette : une disquette MASTER peut booter dans tous les cas et contient tous les fichiers externes (RENUM etc..) mais réserve ainsi environ 90 secteurs supplémentaires au système. Une disquette SLAVE permet de booter seulement si le DOS a déja été chargé, mais économise ainsi 90 secteurs environ.

**Attention** : Le HIMEM doit être en #9800 (Faire éventuellement RELEASE)

-**ATTENTION** : une réponse affirmative à la question **"Format"** est irréversible, mais sera nécessaire lors de la première utilisation de la disquette.

La commande **INIT** affecte la zone mémoire #3000-#B0FF. Il est donc recommandé d'effectuer une sauvegarde de son application en cours avant d'initialiser une disquette.

Exemple : ***INIT***
***INIT A***
***INIT B , 18***
***INIT A, 18, 41,D***

## **BACKUP** (lecteur) **TO** (lecteur)

Effectue la copie intégrale d'une disquette (la source) sur une autre disquette (la cible).

- La disquette-cible sera éventuellement formatée, avec évidemment le même type de formatage que la disquette-source.

- Si lon effectue la copie avec un seul lecteur (source identique à cible), il y aura des manipulations de disquettes (4 pour une simple face).

- Le DOS reconnait automatiquement le type de disquette (nombre de faces, de pistes, de secteurs). Il faut veiller à ce que la cible ait le même type de formatage que la source, autrement un message **I/O ERROR** sera généré.

- Il est recommandé de formater la disquette-cible à chaque fois pour avoir un formatage cohérent et pour des raisons de fiabilité.

- Quel que soit le nombre de secteurs de la disquette, le BACKUP sera fait en aussi peu de manipulations que possible.

- Un **BACKUP** affecte la zone mémoire #600 à #B4FF. Attention, tous les programmes seront donc détruits **(NEW)**.

- Pour éviter les fausses manipulations, il est recommandé de protéger la disquette-source contre l'écriture (abaisser le taquet au coin).

Exemple : ***BACKUP***
***BACKUP TO B***
***BACKUP C***

## **COPY** (COPYM, ou COPYO,) (NFA) (TO NFA) (,C) (,N)

Copie des fichiers . **M et O** agissent comme les suffixes de **SAVE :**

- **COPY** vérifie que le nom du fichier-cible (la copie créée) n'existe pas déjà sur la disquette-cible. Dans le cas contraire, le message **"FILE ALREADY EXISTS"** apparait.

- **COPYO** efface sur la disquette-cible le fichier-cible NF s'il existait auparavant. Dans le cas contraire, cette commande se comporte comme **COPY.**

- **COPYM** permet de "merger" (coller bout à bout) des fichiers, grâce notament à l'utilisation des Jokers dans les noms de fichier-source . Si le fichier-cible existait déja sur la disquette-cible, les fichiers désignés par le nom source lui seront "mergés" (ajoutés à la suite). Sinon, le fichier cible sera créé et sera composé de l'ensemble des fichiers désignés par le nom-source.

Exemple : ***COPYM "A-*.CDE" TO "B-PROG"*** copie tous les fichiers de la disquette A , d'extension CDE en un seul fichier de nom PROG sur la disquette B.

**,C** permet , s'il existe plusieurs fichiers de même nom pouvant être copiés, de demander confirmation pour chacun. Avant de commencer, le nom du fichier est affiché, suivi de **Y / N** . Taper **Y** (YES) pour copier le fichier affiché, **N** (no) dans le cas contraire.

**,N** : pour dupliquer sur la même disquette les fichiers désignés. Lorsque les noms de source et de cible désignent le même lecteur, des messages sont automatiquement affichés pour vous indiquer de changer de disquette, et le DOS attend que vous appuyez sur **RETURN** pour passer à l'étape suivante de la procédure. Cependant, avec le **COPYM** par exemple, on peut, grâce à l'option **,N** empêcher l'affichage de ces messages et effectuer la copie sans interruption donc plus vite.

Pour utiliser les Jokers avec **COPY**, 2 cas se présentent en général :

- Avec **COPY** et **COPYO,** le nom source peut comporter des Jokers, ne comporter que le n° de lecteur, ou même être omis. Quand on place un Joker dans le nom source, on doit le remettre à l'emplacement correspondant dans le nom cible. Les différents fichiers ainsi désignés seront copiés avec le même nom que chaque fichier source copié. Seule exception, on peut mettre un Joker dans le nom source, et omettre le nom cible en n'indiquant que le n° du lecteur-cible, ou composer le nom cible que de Jokers ("*.*").

- Avec **COPYM,** le nom source peut comporter des Jokers, mais pas le nom-cible, qui ne doit désigner qu'un seul fichier.

Exemple : ***COPY "A-TOTO" TO "B-RIRI"*** copie le fichier TOTO de la disquette A sur le disquette B et le rebaptise RIRI.

**COPY "B-*.COM" TO "C-*.CDE"** copie sur la disquette B tous les fichiers d'extension ".COM", en leur donnant la nouvelle extension ".CDE".

## G/ CONFIGURATION DE LA DISQUETTE

## SYS

- Permet de visualiser la configuration du DOS qui se trouve en mémoire (et non celle qui est sur la disquette).
- Affiche les lecteurs connectés et les valeurs par défaut (**NUM / RENUM**).

Exemple : ***SYS***

## DSYS (lecteur)

- Permet de visualiser la configuration de la disquette spécifiée. Cette instruction affiche :

Le nombre de pistes de chaque lecteur
les lecteurs non connectés
le nom de la disquette
les instructions d'initialisation
les valeurs par défaut de **NUM** et **RENUM.**
le type de clavier à l'initialisation.

Exemple : ***DSYS B***

## DNAME (Lecteur)

- Modifie le nom de la disquette contenue dans le lecteur désigné. Le nouveau nom est alors demandé, après affichage de l'ancien.

Exemple : ***DNAME***
***DNAME A***

## INIST (lecteur)

- Modifie sur la disquette spécifiée les instructions d'initialisation (voir le paragraphe "MISE SOUS TENSION") .

Exemple : ***INIST B,"CLS:PAPER4:INK6:HELLO*** : la commande HELLO charge le fichier du même nom et l'exécute lors d'une initialisation. Ce fichier peut comporter des messages etc.. Cela permet de personnaliser la présentation de la disquette.

## **DKEY** (lecteur) (,A) (,S)

Permet de définir le type du clavier à l'initialisation:
QWERTY sans accent si aucune option n'est précisée.
AZERTY avec l'option **,A**
Avec les accents avec l'option **,S.**

Exemple : ***DKEY B, A*** donne le lecteur B en AZERTY.
***DKEY , S*** donne le lecteur courant en QWERTY accentué.

## **TRACK** ($EN_A$) (,$EN_B$) (,$EN_C$) (,$EN_D$)

- Modifie , en mémoire seulement la configuration des lecteurs, en déterminant, pour chacun (respectivement A, B, C et D) le nombre de pistes. Une valeur 0 "débranche" le lecteur : si l'on appelle ce lecteur, un message "DRIVE NOT IN LINE ERROR" apparaitra. Si aucune valeur n'est précisée pour un lecteur, sa configuration ne change pas. Pour configurer un lecteur en double tête, ajouter **";S"** à son nombre de pistes.

Exemple : ***TRACK 42*** autorise le lecteur A en configuration 42 pistes. Les lecteurs B, C et D ne changent pas de configuration.
***TRACK 41,,42;D*** autorise le lecteur A en 41 pistes et le lecteur C en 42 pistes double-tête.

## **DTRACK** (lecteur) (,$EN_A$) (,$EN_B$) (,$EN_C$) (,$EN_D$)

- Même commande que ci-dessus, mais la modification est effectuée sur la disquette et non pas sur le DOS qui est en mémoire RAM. C'est cette configuration qui sera placée en mémoire lors de l'initialisation de la disquette en question.

Exemple : ***DTRACK A, 42***

## **DNUM** (lecteur) (,EN) (,EN)

- Modifie sur la disquette les valeurs par défaut (après un RESET avec ladite disquette) respectivement de la première ligne et du pas de la numérotation automatique et de la renumérotation (voir les instructions **NUM** et **RENUM** au chapitre "aides à la programmation).

Exemple : ***DNUM B,100,10***

## **SYSTEM** lecteur

- Spécifie le lecteur système, c'est à dire celui où le DOS ira chercher les blocs des fichiers externes (**COPY, BACKUP** etc..).

- A l'initialisation, c'est le lecteur A qui est choisi par défaut comme lecteur système.

Exemple : ***SYSTEM D*** (il fallait le faire !!!)

Chapitre 4

# Les aides à la programmation

Manuel d'utilisation du MICRODISC ORIC

## **RENUM** ($EN_1$) (,$EN_2$) (,$EN_3$) (,$EN_4$)

Refait toute la numérotation des lignes d'un bloc ou de la totalité d'un programme.

- Les parametres sont dans l'ordre :
  - $EN_1$ est l'origine de la renumérotation (premier nouveau n° de ligne).
  - $EN_2$ est le pas (la valeur dont les n° de lignes seront incré mentés).
  - $EN_3$ et $EN_4$ sont la première et la dernière ligne du bloc de programme à renuméroter.

- Si un ou plusieurs parametres ne sont pas précisés, les valeurs par défaut seront :
  Origine : 100
  Pas : 10
  Première et dernière ligne à renuméroter : la première et la dernière ligne du programme.
Les valeurs d'origine et du pas sont modifiables par la commande **DNUM .**

- Outre les lignes, les instructions suivantes : **GOTO, GOSUB, ON GOTO, ON GOSUB, THEN, ELSE, RUN, RESTORE** (en majuscules) sont automatiquement modifiées en fonction des nouveaux numéros de lignes de branchement.

- Si une ligne n'est pas trouvée, le **RENUM** prendra la ligne suivante, ce qui permet de supprimer les **REM** sans avoir à recalculer certains branchements.

- Lors de la renumérotation des blocs, il convient de faire attention à donner des numéros de ligne cohérents. De plus, la dernière ligne précisée sera également renumérotée.

- Si une expression numérique est rencontrée ( **GOTO 100 * A** par exemple), elle ne sera pas modifiée. Pour cela il est fortement conseillé d'utiliser **ON GOTO** ou **ON GOSUB** qui sont là pour çà !.

- Le **RENUM** est très rapide, de 2 à 5 kOctets par seconde. La zone mémoire comprise entre le début du programme (pointé par #9A) et le **HIMEM** est affectée par le **RENUM.**

- Il ne faut pas faire de **RESET** pendant une renumérotation, car le programme est codé avant d'être renuméroté.

| Exemple : | | | |
|---|---|---|---|
| | ***RENUM*** | équivaut à | ***RENUM 100, 10*** |
| | ***RENUM1000*** | " | ***RENUM 1000, 10*** |
| | ***RENUM,,,10000*** | " | ***RENUM100,10,0,10000*** |
| | ***RENUM1000,,2300*** | " | ***RENUM1000,10,2300*** |
| | ***RENUM100,10,0,10000*** | est l'expression complète. | |

## **MERGE** NF (,L)

- Mélange le programme BASIC en mémoire et le programme BASIC précisé.

- Si le résultat donne un programme trop long, le message "OUT OF MEMORY" est généré.

- Les variables sont conservées, avec les limites habituelles à ce genre d'opération, c'est à dire que toutes les fonctions utilisateurs, définies par **DEF FN** seront perdues.

- En mode programme, il faut faire en sorte qu'aucune ligne ne sera insérée avant la ligne où se trouve la commande **MERGE** sous peine de déclencher des messages d'erreur **"SYNTAX ERROR"** et autres .

- Si des lignes existent à la fois dans le programme en mémoire et dans le programme à "merger", celles résidant en mémoire sont conservées, mais le message **"LINES ALREADY EXISTS"** est affiché.

- Au fur et à mesure de l'insertion, le numéro des lignes du programme à insérer sont affichés. Ceci ralentit notablement l'exécution, et peut être évité en précisant l'option ,L

Exemple: ***MERGE "TOTO"***
***MERGE "ESSAI",L***

## **DELETE** (EN $_1$) - (EN $_2$)

- Détruit un bloc de lignes sans perdre les variables.

- La syntaxe de **DELETE** est la même que celle de **LIST** à ceci près que le **"-"** est obligatoire (sinon **DELETE** siginifierait NEW ! ) .

- Précautions d'emploi : en mode programme, il ne faut pas que **DELETE** se trouve dans ou après le bloc détruit. De même, les fonctions définies par un **DEF FN** se trouvant dans un bloc détruit seront perdues.

Exemple : ***DELETE 100 - 100*** élimine la ligne 100
***DELETE -1000*** élimine toutes les lignes jusqu'à la ligne 1000.
***DELETE 234 - 987*** détruit les lignes 234 à 987

## **SEEK** (EA) (,S) (,M)

- **SEEK** EA recherche dans le programme BASIC en mémoire la chaine EA, et affiche toutes les lignes où la chaine est trouvée. Le listing peut être interrompu comme un vrai listing (ESPACE ,Ctrl C...) .

- La chaine ne doit pas comporter le code ASCII NUL (0). Dans le cas contraire, un message"INVALID STRING ERROR" sera généré.

- Pour chercher des instructions BASIC, il est necéssaire de les coder grâce à **TKEN.**

- La chaine, un peu comme les noms de fichier, peut comporter des caractères 'jokers'. Le caractère joker utilisé ici est le "**£**" (le '?' n'a pas été retenu car il est d'usage trop courant dans le BASIC) .

- La chaine à chercher peut comporter au maximum 78 caractères.

-Si l'option **,S** est précisée, les ligne ne sont pas affichées, mais seulement un message indiquant le nombre de fois où la chaine a été trouvée. De plus la variable SK contient aussi ce nombre.

- Si les options **,S** et **,M** sont précisées, aucun message ni aucune ligne ne sera affichée, seule la variable SK contiendra le nombre d'occurences de la chaine.

- **SEEK** ("tout court") permet de lister une ligne à la fois. S'utilise après un **SEEK** EA. Chaque fois que l'on tape **SEEK**, la ligne où se situe la prochaine occurence est affichée. La chaine à chercher, précisée lors du **SEEK EA**, reste en mémoire jusqu'à un **SEEK** avec une autre chaine ou un **RESET.**

- Si un **SEEK** est effectué alors qu'il n'y a aucune chaine à chercher, le message 'SYNTAX ERROR' sera généré.

Exemple: Soit le programme suivant :

```
10 REM TEST SEEK SEDORIC 0.0
20 PRINT "BONJOUR SEDORIC 1.0!"
30 END
```

***SEEK "BONJOUR"*** listera la ligne 20
***SEEK "BONJOUR",S*** affichera 1 Found(s)
***SEEK "ORIC £.0"*** listera les lignes 10 et 20
ensuite,***SEEK*** listera la ligne 10, et un 2ème ***SEEK*** listera la ligne 20
***SEEK CHR$(#BA)*** listera la ligne 20 car #BA est le code (ou token ) de la commande BASIC **PRINT**

## CHANGE $EA_1$ TO $EA_2$

- Remplace toutes les occurences de la chaine $EA_1$ par la chaine $EA_2$ dans le texte BASIC.

- La chaine suit les mêmes conventions que pour SEEK (pas de caractère nul, longueur, joker) .

- Des caractères jokers peuvent être précisés dans la chaine source. S'ils sont précisés dans la chaine cible, ils seront laissés tels quels.

- L'usage du caractère 'joker' permet de trouver plusieurs chaines à remplacer par une seule, mais ne permet pas de faire un changement "sélectif" comme le fait la commande REN pour les noms de fichier .

- Si le programme BASIC devient trop long pour tenir en mémoire, le message 'OUT OF MEMORY ERROR' est généré et le remplacement est arrété.

- De même, si une ligne devient trop longue , le remplacement est arrété et le message **"LINE ; xxxx**
**?STRING TOO LONG ERROR"** est affiché.

Exemple: ***CHANGE "ORIC" TO "EUREKA"***

***CHANGE CHR$(#BA) TO CHR$(#8F)***
change tous les **PRINT** en **LPRINT** car #BA est le code de PRINT et #8F est le code de LPRINT .

Dans le petit programme de l'exemple du SEEK, la commande:
***CHANGE "ORIC £.0" TO "BOF !"*** donnera les lignes:
***10 REM TEST SEEK SEDBOF !***
***20 PRINT "BONJOUR SEDBOF !"***

## **NUM** ($EN_1$) (,$EN_2$)

- Spécifie les parametres de la numérotation automatique. Le premier parametre est l'origine (numéro de la première ligne) et le deuxième le pas de la numérotation. Chaque fois que l'on appuie sur FUNCT et RETURN, le numéro courant est affiché et le nouveau est calculé .

- Les valeurs prises par défaut sont les mêmes que pour **RENUM**, c'est à dire 100 pour l'origine et 10 pour le pas. Elles sont modifiables par **DNUM** .

Exemple :

***NUM 1000*** donne une numérotation automatique de 10 en 10 à partir de la ligne 1000
***NUM ,5*** donne une numérotation automatique de 5 en 5 à partir de la ligne 100.
***NUM 3300 , 20*** donne une numérotation automatique de 20 en 20 commençant à partir de la ligne 3300.
***NUM END***

## NUM END

- Si **NUM END** est précisé, le DOS recherche automatiquement la dernière ligne du programme, et démarre la numérotation à la ligne suivante (Dernière + argument par défaut).

Exemple : ***NUM END***

# Chapitre 5

# Le BASIC étendu

Manuel d'utilisation du MICRODISC ORIC

# A / GESTION DU CLAVIER

## KEY SET ou KEY OFF

- Inhibe (OFF) ou autorise (SET) le clavier. Un **KEY OFF** augmente de 20% la rapidité de l'ORIC. Attention : en mode direct, vous perdez la main si vous tapez **KEY OFF.** Pour la reprendre, appuyez sur le RESET de l'ORIC.

Exemple :

```
10 KEY OFF
20 .....
.....
820 KEY SET
830 INPUT .....
```

## KEYIF EN GOTO ou KEYIF EN THEN

- Détecte la pression sur une touche précisée par l'argument EN. La pression est détectée même si le clavier est inhibé ou si l'on appuie sur plusieurs touches en même temps.

- La syntaxe est tout à fait similaire à celle d'un **IF THEN ELSE .**

- EN est un code un peu particulier donné dans un tableau en annexe 7. Cette codification a été préférée au code ASCII par exemple pour accélérer l'exécution.

Exemple :

```
100 KEYIF #84 THEN SHOOT
...
200 KEYIF AB THEN 100 ELSE PING
```

## QWERTY

- Configure le clavier normalement (généralement pour annuler une commande **AZERTY**) et exécute un **ACCENT OFF** . Cette commande est inopérante en mode **HIRES.**

Exemple :

```
100 QWERTY
```

# AZERTY

- Transforme le clavier QWERTY d'origine en clavier AZERTY (clavier français semblable à celui des machines à écrire), c'est à dire que

| | | |
|---|---|---|
| **Q** | devient | **A** |
| **W** | " | **Z** |
| **A** | " | **Q** |
| **Z** | " | **W** |
| **M** | " | **;** |
| **;** | " | **M** |

Pour une plus grande commodité, les chiffres restent accessibles sans SHIFT.
**Cette commande ne fonctionne pas sur l'ORIC 1.**

- Effectue automatiquement un **ACCENT SET** (voir cette commande).

- **Attention :** - L'instruction **KEY $** n'est pas affectée par **AZERTY** : le clavier correspondant à la touche et non au code reste en QWERTY.
- Les fonctions CTRL ... suivent la touche considérée en AZERTY car elles se réfèrent au code ASCII de la lettre et non à la touche du clavier.
- Les touches de fonctions elles, **ne sont pas affectées** par la commande AZERTY car elles se réfèrent à la touche et non au code.

- La commande AZERTY est interdite en mode HIRES.

# ACCENT SET / ACCENT OFF

- Cette commande redéfinit un certain nombre de caractères, permettant ainsi d'avoir les caractères accentués à l'écran. Les caractères redéfinis ont été choisis pour avoir des codes ASCII correspondant à ceux des caractères français dans les imprimantes ainsi configurées. Les caractères suivants sont redéfinis :

| ASCII | NORMAL | ACCENTUE |
|---|---|---|
| **40** | @ | **à** |
| **5C** | \ | **ç** |
| **7B** | { | **é** |
| **7C** | \| | **ù** |
| **7D** | } | **è** |
| **7E** | ▒ | **ê** |

L'accent circonflexe ne peut être obtenu seul, on dispose pour cela du e accentué.
**Attention : ACCENT OFF** rétablit tous les caractères normaux, alors que **ACCENT SET** ne touche qu'aux caractères redéfinis. Cette commande est interdite en mode HIRES.

Exemple :

```
ACCENT SET
ACCENT OFF
```

## B / GESTION DES TOUCHES DE FONCTION

L'affectation des touches de fonction se fait en 2 étapes : à une chaine de caractères , un mot-clé du DOS ou un ordre BASIC correspond un code de 0 à 255 que l'on affecte à une touche (**KEYDEF**) . On peut aussi modifier certaines commandes utilisateurs grace à la commande **KEYUSE** .

## **KEYDEF** EN

- Cette instruction permet de définir à volonté des touches de fonctions : apppuyer sur la touche définie donne la fonction correspondante.

- Cette commande a été voulue pratique avant tout. Elle est assez particulière : le nombre en argument est le code de fonction (indiqué ci-après). Après avoir entré la commande, il suffit de taper la touche que l'on veut assigner .

- Si l'on tape simplement la touche, on definira une fonction accessible par FUNCT. + la touche en question. Si l'on tape la touche + SHIFT, on définira une fonction accessible par FUNCT. + SHIFT . + la touche tapée.

- Les codes de fonctions sont les suivants :
  - Codes de 0 à 15 pour les commandes définissables par l'utilisateur grâce à la commande **KEYUSE** (voir plus bas).
  - Codes 16 à 31 pour les quelques mots-clés spéciaux (voir leur liste en annexe 6).
  - Codes 32 à 127 pour les mots-clés du DOS (voir leur liste en annexe 6).
  -Codes 128 à 246 pour les mots-clés du BASIC (voir leur liste dans la table du Manuel de l'ORIC).
  - Code 254 pour FUNCT + DEL (effacer la mémoire-tampon).
  - Code 255 pour FUNCT + RETURN (numérotation automatique des lignes de programme.

Exemple :

***KEYDEF 0*** affecte à la touche qu'on va presser la commande définie par **KEYUSE 0, "xxxx"**
***KEYDEF #80*** Affecte à la touche que l'on va presser le mot-clé END (#80 étant en hexadécimal le code de END).

## **KEYUSE** EN, EA

- Commande fonctionnant sur l'ATMOS uniquement et pas sur l'ORIC 1. Elle permet d'assigner une définition (une ou plusieurs commandes) aux 16 commandes utilisateur.

- EN est le code de la commande (de 0 à 15), EA la chaine de définition des commandes (16 caractères alphanumériques maximum) .

- La chaine de définition peut comporter tous les caractères ASCII (sauf 0 et les codes >127), particulièrement CHR$(13) qui simule un **RETURN.**

Exemple : ***KEYUSE 0 , "PAPER 0:INK 7" + CHR$ ( 16 ) + CHR $ ( 13)***

## **KEYSAVE** NF

- Sauvegarde la configuration courante des touches de fonctions sur la disquette.

- Fonction équivalant à **SAVE NF, A#C800,E#C97F.**

- Pour rappeler une configuration sauvegardée, il suffit de charger le fichier concerné.

Exemple :

***KEYSAVE " ESSAI.KEY ".***
( l'extension KEY permet d'identifier une configuration de clavier dans le catalogue mais n'est pas obligatoire) .

## **VUSER**

- Visualise les chaines de définition des 16 commandes utilisateurs.

- Si ces chaines comportent des caractères de controle, ils aparaissent en vidéo inverse ( M en vidéo inverse signifie **Return** par exemple, car **Ctrl M** est équivalent à **Return**).

Exemple: ***VUSER***

## C/ TRAVAIL SUR LES CHAINES

### TKEN VA

- Instruction inutilisable en mode direct.

- Code la variable comme une suite d'instructions BASIC (de Tokens), qui pourront être exécutées par un **STRUN** (voir ci-dessous) .

- La variable doit contenir la chaine à coder. Après exécution, elle contiendra la chaine codée.

-La chaine doit avoir au plus 79 caractères. En général, elle est raccourcie après codage (les mots-clés du BASIC sont en effet codés sur un seul octet) .

Exemple : ***TKEN A $*** ) c'est du mode direct, il faut
***TKEN "PING "*** ) un n° de ligne

de plus TKEN demande une variable alphanumerique et non une constante
voir page ci-contre

### UNTKEN VA

- Commande inverse de la précédente **TKEN.** La commande **UNTKEN** interprète tout les codes supérieurs à 128 comme des numéros de mots-clés BASIC, qui seront donc transformés en leur équivalent en "clair" .

- La chaine peut avoir une longueur quelconque mais la chaine décodée étant plus longue que la chaine codée, il peut apparaitre une erreur de type **"String too long ERROR".**

Exemple :
```
10  A$="PRINT"
20 UNTKEN  A$
30 PRINT A$
```
*affiche"PRINT" car #BA est le code de PRINT*

## **STRUN** VA

- Exécute une chaine codée par **TKEN** comme une ligne de programme BASIC.

- Permet d'exécuter facilement l'entrée de formules. (voir exemples ), ou de créer un fichier de commande.

- Tous les mots du BASIC ou du DOS sont utilisables sauf ceux qui sont normalement inutilisables en mode direct. Si une erreur est générée, tout se passe comme si l'instruction avait été exécutée en mode direct.

- La chaine doit être codée avant exécution (voir **TKEN).** Mais si une même chaine doit être exécutée plusieurs fois, il n'estt necessaire de la coder qu'une fois bien entendu .

- La commande STRUN est très rapide. En revanche, la commande TKEN est, elle, lente . D'où l'intérêt de stocker les chaines déjà codées si possible.

Exemples :

```
10 A$ = "PING"
20 TKEN A$: STRUN A$   (Va éxécuter le PING)
30 INPUT "EXPRESSION"; A$
40 A$ ="EP = " + A$
50 TKEN A$ : STRUN A$
60 PRINT EP
```

## **INSTR** $EA_1$, $EA_2$, EN

- Recherche la première occurence de la chaine $EA_2$ dans la chaine $EA_1$, à partir du caractère dont la position est spécifiée par l'argument EN.

- La position est renvoyée par la variable IN. Si aucune occurence n'est trouvée (si $EA_2$ne se trouve pas dans $EA_1$), IN retourne la valeur 0.

- Si la chaine à examiner ($EA_1$) est vide, IN retourne 0 par convention.

- Si la chaine recherchée ($EA_2$) est vide, une d'erreur "ILLEGAL QUANTITY ERROR" est générée .

Exemple : ***10 INSTR "BEAU TEMPS" , "TEMPS" , 1***
***20 PRINT IN***
donnera ***6.***

# D/ GESTION DES ERREURS

## ERR SET OU ERR OFF

- Interdit (**ERR SET**) ou autorise (**ERR OFF**) l'interruption d'un programme par une erreur du DOS. **Seules les erreurs du DOS seront prises en considération** (celles du BASIC seront ignorées). Voir la table des messages en annexe 2.

- A l'initialisation, les interruptions sont autorisées et les erreurs sont affichées comme des erreurs classiques du BASIC.

- Lorsque les interruptions par les erreurs sont inhibées (**ERR SET**), et qu'une erreur est déclenchée, le programme continue en exécutant la ligne spécifiée par **ON ERR GOTO,** ou s'arrête si aucune instruction de gestion d'erreur n'est prévue dans le programme.

- **ERR** réinitialise la ligne de traitement des erreurs sur une ligne vide. Ill faut donc faire le **ERRGOTO** après le **ERRSET.**

- Quel que soit le mode, si une erreur est déclenchée, la variable EN sera chargée avec le numéro de l'erreur (que l'on retrouve à l'adresse #4FD), et la variable EL sera chargée avec le numéro de la ligne où s'est produite l'erreur (également stockée à l'adresse #4FE - #4FF).

- Par convention, si l'erreur se produit en mode direct, le numéro de ligne sera pris comme égal à 65635.

## ERRGOTO EN

- Spécifie le numéro de la ligne où seront traitées les erreurs déclenchées par le DOS, dans le cas ou une **ERR SET** a été exécuté auparavant.

- RAPPEL : un **ERR SET** annule un **ERRGOTO.**

Exemple :

```
1000 ERR GOTO 10000
...
10000 PRINT "Tapez une variable numérique !"
10010 GOTO 990
```

## RESUME (NEXT)

- Reprend l'exécution où elle avait été interrompue (**RESUME**) , ou à la ligne suivante (**RESUME NEXT**) . Le contexte est inchangé, hormis ce qui a été modifié lors du traitement de l'erreur.

## ERROR EN

- Déclenche une erreur de numéro spécifié. Cela permet de centraliser les erreurs, y compris celles générées par l'utilisateur, dutype "Texte trop long " etc..

- Les variables EN permises sont de 50 à 255, les autres étant réservées aux erreurs du DOS.

- Si les interruptions par les erreurs sont autorisées (ERR OFF), un "USER xxx ERROR" est généré, où xxx est le numéro de l'erreur.

exemple : ***ERROR 50***

## E/ SAISIE FORMATEE :

**LINPUT** (@EN,EN,) (EA,) EN ; VA (,E) (,S) (,C) (,J) (,K)

**LINPUT** (@X,Y,) (Caractère,) Longueur; VA . . .

Instruction destinée à saisir des textes (n'accepte pas les variables numériques).
- Entrée de texte formatée, de longueur spécifiée. Le texte entré est assigné à la variable alphanumérique spécifiée VA.
Cette fonction est très puissante, il faut donc l'aborder progressivement . Dans sa forme la plus simple (**LINPUT longueur,VA**), elle agit à peu près comme un **INPUT** dont on choisirait le nombre de caractères à frapper.

- En fait, cette comande crée une fenêtre à l'écran à l'intérieur de laquelle l'éditeur est du type pleine page. Selon les options précisées, cette fenêtre sera préalablement remplie avec un caractère désigné éventuellement, et l'on pourra ou non sortir de la fenêtre par les flèches de curseur.

- A l'intérieur de la fenêtre, seuls quelques caractères de contrôle sont valides : **CTRL T, CTRL N, CTRL Z** (CTRL Z est équivalent à **ESC** pour générer des attributs vidéo) et **DEL.** Le curseur est automatiquement autorisé.

- La longueur de la fenêtre peut varier de 1 à 255.

- Quel que soit le mode, on peut sortir en appuyant sur ESC ou sur RETURN. La variable alpha sera alors chargée avec les caractères entrés. La variable OM voit son contenu varier avec son mode de sortie :

0 = sortie par RETURN
1 = ESC
2 = ⇐
3 = ⇒
4 = ⇓
5 = ⇑
6 = sortie automatique (fin de fenêtre)

**,E** permet de ne pas effacer la fenêtre avant l'entrée du texte.

**,S** permet de sortir avec les flèches de déplacement.

**,C** autorise la sortie à la fin de la fenêtre, sans frapper RETURN, dès que le dernier caractère de la fenêtre est tapé. Sinon, le curseur revient au début de la fenêtre.

**,K** justifie le texte entré à l'écran en ajoutant des espaces (seul l'affichage est justifié, les espaces n'apparaissant pas dans la variable alpha).

**,J** idem mais justifie la variable alphanumérique en ajoutant des espaces, l'écran n'est pas affecté.

- Le premier caractère de EA servira à remplir la fenêtre si l'effacement est demandé. Si la chaine vide est précisée, le caractère par défaut sera ".". A l'initialisation, c'est aussi le point qui est pris comme caractère . Le caractère de remplissage est gardé d'un LINPUT à l'autre, tant qu'il n'est pas modifié explicitement .

- **@X, Y** permet d'effectuer l'entrée à une position sur l'écran d'abscisse X et d'ordonnée Y. Même convention que pour l'instruction **PLOT** (décalage d'un caractère vers la droite pour l'ORIC 1).

Exemple : ***LINPUT 20;A$,S***
***LINPUT "-",100;A$***
***LINPUT @10,3, 100;A$,C,J***
***LINPUT @10,3,"-",2;A$***

## **CREATEW** (NF)

- Crée un masque d'écran exploitable par **WINDOW.** Le masque consiste en un écran texte de 25 lignes de 40 caractères, qui commence à la ligne 2 de l'écran.

- Commande interdite en mode HIRES. Tous les caractères peuvent être écrits, l'éditeur étant du type pleine page.

- A tout moment, **CTRL S** peut sauver l'écran, et **CTRL C** peut en sortir.

- Les champs de données sont définis par **CTRL W**, apparaissant comme un pavé plein à l'écran.

- Le nombre de champs n'est pas limité, la longueur d'un champ est en revanche limitée à 255 caractères.

Exemple ***CREATEW , "MASQUE. SCR"***

# **WINDOW** (NF)

- Permet la saisie d'un masque d'écran.

- Commande interdite en mode HIRES.

- Si le nom du fichier NF contenant le masque (sauvé par **CREATEW**) n'est pas précisé, le masque courant sera pris, si c'est possible.

- Avant d'appeler la commande **WINDOW** , il faut dimensionner correctement le tableau **WI$** avec le nombre de champs du masque. Sinon, le tableau sera dimensionné implicitement à 11 éléments, comme d'habitude.

- La commande **WINDOW** affiche le masque de saisie à l'écran, et remplit les champs de données avec les valeurs trouvées dans le tableau **WI$**, complétées par des espaces si nécessaire.

- Pendant la saisie des données, il n'est permis à l'utilisateur que d'écrire dans les champs (bonjour les écolos !), ou de se déplacer d'un champ à l'autre par les flèches de curseur ou la touche **RETURN** . La plupart des caractères de controlle sont filtrés, pour assurer une mise en page correcte (Ctrl L interdit par exemple) .

- Lorsque l'utilisateur tape **CTRL C**, tous les champs sont lus, les uns après les autres, et sont chargés dans l'ordre (de gauche à droite et de haut en bas) dans le tableau WI$. Le nombre de champs n'est pas limité, mais l'utilisateur doit dimensionner correctement le tableau WI$.

- Un champ doit avoir au maximum 255 caractères de longueur, car c'est la longueur maximum d'une chaine.

- Si l'Oric est en mode 40 colonnes, des champs peuvent aparaitres disjoints alors qu'ils le sont en mode normal 38 colonnes. Le problème se pose aussi sur l'Oric-1. Il faut prendre garde à gérer correctement les champs à cheval sur plusieurs lignes .

Exemple

***WINDOW " ESSAI "***
*appelle le masque **ESSAI** dans le tampon et exécute la saisie*
***WINDOW***
*exécute la saisie du masque se trouvant dans le tampon .*

# F/ IMPRESSION FORMATEE

## **USING** EN, EA (,VA)

Formatage de nombres . Le nombre est représenté par la variable numérique EN, et son format par la variable alpha EA.

- Si une variable alphanumérique est précisée (,**VA**), le nombre formaté ne sera pas affiché mais affecté à la variable précisée.

- Le format d'impression EA est donné ci-dessous. Tous les autres caractères seront reproduits tels quels.

**+** affiche le signe , + ou -
**-** affiche un espace, ou le signe - (pas de signe devant un nombre positif).
**^** affiche l'exposant en notation scientifique (+12 par exemple).
**&a** : a est le caractère qui remplacera les 0 devant la partie entière d'un nombre. Par défaut, il s'agira d'espaces.
**%x** : x vaut de 0 à 9, affiche x caractères de la partie entière de EN.
**#x** : x vaut de 0 à 9, affiche x caractères de la partie décimale de EN.
**! x** : x vaut de 0 à 9, arrondit au $x^{ième}$ caractère de la partie entière.
**@x** : x vaut de 0 à 9, arrondit au $x^{ième}$ caractère de la partie décimale.

- Si l'on veut faire entrer le nombre dans une partie entière trop courte, un message "ILLEGAL QUANTITY ERROR" est généré.

Exemple : ***10 A= 10000 / 6***
***20 USING A,"&0Nombre:%9.#4 E^"***
affiche ***Nombre : 000001666.6666 E+00***

***30 USING A,"@2+%5. Francs et #2 centimes."***
affiche ***1666 Francs et 67 centimes.***

***40 USING PI,"&*%2.#5",A$***
***50 PRINT A$***
affiche ****3.14159***

## **LUSING** EN, EA (,VA)

Agit comme la commande **USING** mais avec sortie sur l'imprimante au lieu de l'affichage à l'écran.

Exemple : ***LUSING 12, "+ -^#3"***

## **WIDTH** (EN) ou **WIDTH LPRINT** (EN)

Précise la largeur utile de l'affichage à l'écran (**WIDTH**) ou de l'impression (**WIDTH LPRINT**). EN représente le nombre de caractères au bout duquel un retour chariot (CR) et un saut de ligne (LF) seront exécutés automatiquement.

- Sur l'ORIC 1, la gestion de l'écran et de l'imprimante ne sont pas différenciées, c'est à dire qu'un **WIDTH** formate également l'écran et la sortie impression.

- Les valeurs par défaut sont 40 pour l'écran et 80 pour l'imprimante (ATMOS) et respectivement 53 et 93 (ORIC 1).

Exemple ***WIDTH 20*** provoquera un affichage de 20 caractères de large.
***WIDTH LPRINT 132*** provoquera une impression en 132 colonnes.

## G/ GESTION DE L'IMPRIMANTE

### OUT EN

Envoie un code ASCII vers l'imprimante sans gérer aucunement les retours chariots etc.. Cela remplace donc, en plus rapide (et à la saisie, et à l'exécution) un LPRINT CHR $ (EN) .

- Cette instruction ne souffre donc pas des bogues connues dans la gestion d'imprimante de l'ORIC 1.

Exemple :

***OUT 27:OUT 33:OUT 30*** provoque une impression en caractères gras sur une EPSON FX, équivalent à LPRINT CHR (27) ; "!" ; CHR$ ( 30 ).

### PR SET OU PR OFF

Met l'imprimante en service (**PR SET**) ou hors service (**PR OFF**).

- après un **PRSET**, et jusqu'à ce que l'utilisateur reprenne la main (BREAK, INPUT...), tout ce qui devait être affiché à l'écran sera envoyé à l'imprimante.
- **ATTENTION:** un **PRSET** interprète tous les **PRINT** en **LPRINT**, mais un **PR OFF** ne transforme pas les **LPRINT** en **PRINT** !

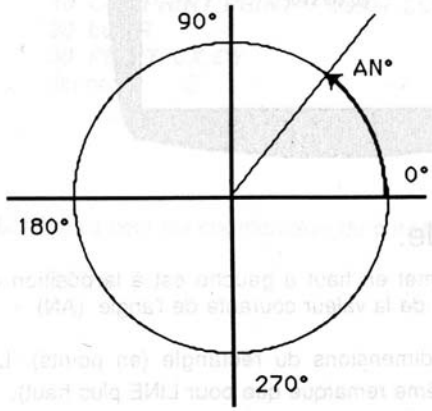
- Dès que le message "READY" est affiché, l'imprimante est déconnectée. En mode direct, il faudra donc mettre plusieurs instruction par ligne.

- Il faut remettre explicitement l'imprimante hors service avant un STOP explicite (**STOP, END**), ou implicite (**CTRL C**, fin de programme..), car une bogue de l'ATMOS remet incorrectement la longueur de la ligne d'écran (il suffit sinon de taper un **WIDTH**).

Exemple : ***PR SET***

# H/ LES INSTRUCTIONS GRAPHIQUES

Ces commandes sont construites autour de la variable **AN** , qui contient l'angle courant. La variable AN est une variable tout à fait normale : l'angle est exprimé en degrés selon la convention trigonometrique :




## **LINE** EN, FB code

Trace EN points dans la direction courante (précisée précédemment par une variable AN) à partir de la position courante du curseur.

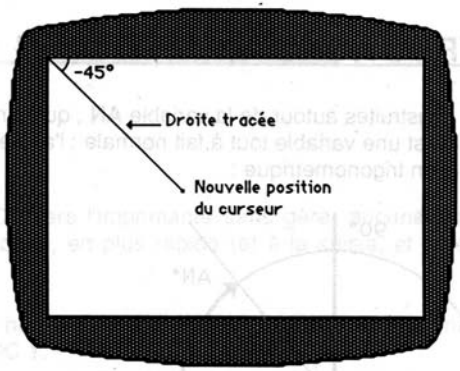
- Le FB CODE est le même que dans les instructions graphiques du BASIC de l'ORIC et donne la couleur de la figure par rapport aux autres couleurs utilisées (FB=0 --> même couleur que PAPER, FB=1 --> même couleur que INK , FB=2 --> inversion etc..)

- Les coordonnées du curseur sont remises à jour.

EXEMPLE :

```
10 HIRES
20 CURSET 0,0
30 AN = -45
40  LINE 100, 0
```

Trace une ligne inclinée à 45°, partant du coin haut gauche (position 0,0) et longue de 100 points (voir figure de la page suivante) et inclinée de 45° dans le sens négatif..




## **BOX** $EN_1$, $EN_2$, FB code.

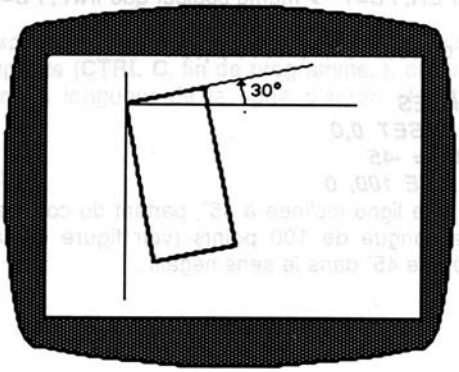
Trace un rectangle dont le sommet en haut à gauche est à la position courante du curseur. Le rectangle sera incliné de la valeur courante de l'angle. (AN)

- $EN_1$ et $EN_2$ représentent les dimensions du rectangle (en points). Le FB code représente le code graphique (même remarque que pour LINE plus haut).

Exemple :

```
10 HIRES:PAPER7:INK0
20 CURSET 40,40,0: AN = 20
30 BOX 50, 120, 1
40 CURSET 40,20,0:AN=-90
50 LINE 160,1
60 CURSET 40,40,0: AN = 0
70 LINE 150,1
```

trace la figure suivante :



## LCUR

Retourne dans les variables CX et CY les coordonnées horizontales et verticales du curseur en mode **TEXT.**

Exemple:
```
10 CLS:PRINT:PRINT  "essai LCUR";
20 LCUR
30 PRINT CX,CY
```
donne:12  2

## HCUR

Idem que **LCUR** mais pour les coordonnées du curseur HIRES.

Exemple:
```
HCUR
```

## I / COMMANDES UTILISATEUR

**USER** x **, DEF** adresse (,O)
**USER** x (,A $EN_1$) (,X $EN_2$) ( ,Y $EN_3$) ( ,P $EN_4$)

Permet l'appel de sous-routines en assembleur, avec passage et retour des parametres.

- **x** est le numéro de la routine utilisateur (de 0 à 3, mais peut être une expression numérique).

- **USER x,DEF** permet de définir l'adresse d'exécution de la routine. Si l'option **,O** est est précisée, la routine s'exécutera dans la RAM en overlay (quelle que soit son adresse).

- **USER x** exécute cette routine à l'adresse définie. Les registres du 6502 sont éventuellement chargés avec les parametres précisés.

- Au retour, les variables RA, RX, RY et RP contiennent les valeurs des registres A, X, Y et P.

- **USER x** est une commande ouverte (comme le **CALL** du Basic), c'est à dire que l'on peut lui rajouter autant de parametres que l'on veut. L'analyse syntaxique s'arrête dès que A, X, Y ou P n'est pas rencontré.

Exemple : ***USER 2,DEF #CCB0***

***USER 1,DEF #4000, 0***

***USER 2, X 12, Y 20***

***USER A, A A00, X 200, P #C1***

## ] (bracket droit)

Même usage que le "!". Il suffit à l'utilisateur de détourner l'adresse #2F9-#2FA sur sa propre routine.

Attention : Après un **QUIT**, le point d'exclamation est obligatoire.

## J/ DIVERS

## QUIT

Replace les pointeurs détournés par le DOS (replace les IRQ et les NMI, rend obligatoire le ! et inhibe les touches de fonctions).

- Cette instruction peut être nécessaire pour exécuter des programmes qui altèrent l'amorce du DOS logée à partir de l'adresse #400.

## RESET

Simule par programme la pression sur le bouton RESET de l'unité de disquettes.

## RESTORE (N° de ligne )

Place le pointeur de **DATA** au début du programme ou de la ligne précisée, qui n'existe pas nécessairement.

- Seul le **RESTORE** en majuscules sera renuméroté par le **RENUM**.

Exemple : ***RESTORE***
***!RESTORE 100*** (pour différencier le **RESTORE** du DOS de celui du BASIC.)
***restore 1000***

## MOVE EA, EA, EA
## ou MOVE début, fin, adresse cible

Permet de déplacer un bloc mémoire,en précisant l'adresse de début (incluse),l'adresse de fin(exclue),et la nouvelle adresse de début du bloc.
**Attention** : Les adresse au dessus de #C000 sont situées en RAM overlay, c'est à dire là où se trouve le DOS: gare aux décalages hasardeux !

Exemple: ***MOVE#A000,#BF40,#1000*** sauve l'écran HIRES en #1000
***MOVE#1000,#2F40,#A000*** le rapelle

## OLD

Récupère un programme BASIC détruit par un **BOOT** ou par un **NEW.**

- Un **OLD** entraine la perte de toutes les variables, comme après un **CLEAR** ou un **NEW.** Parfois, quand un programme a été complètement détruit, un **OLD** ne rend pas la main. Il suffit dans ce cas d'appuyer sur le RESET de l'ORIC pour sortir.

- **ATTENTION** : s'il y a eu affectation de variable entre le **NEW** et le **OLD,** le programme n'est pas récupérable.

## RANDOM (EN)

Initialise le générateur de nombres aléatoires, au hasard si aucun argument n'est précisé, ou avec la racine précisée (permet d'obtenir toujours la même séquence).

- La valeur aléatoire est obtenue, notament grâce aux timers du VIA.

Exemple : ***RANDOM 15***

## SWAP Variable, Variable

Echange le contenu de 2 variables. Ces deux variables doivent bien sûr être de même type.

Exemple : ***SWAP A, B***

# Chapitre 6

# Gestion de Fichiers

Manuel d'utilisation du MICRODISC ORIC

# A/ GENERALITES

Nous n'avons manipulé jusqu'à présent entre la disquette et l'unité centrale que des fichiers de programmes BASIC ou assembleur, des écrans ou des zones mémoire quelconques.

Ce chapitre aborde un autre type de fichiers : les fichiers de données. Cette fois, ce seront des valeurs de variables, numériques ou alphanumériques qui seront stockées dans le fichier.
Bien entendu, pour retrouver les informations stockées, il faut un minimum d'instructions et d'organisation. On regroupe sous le terme un peu général de **gestion de fichiers** cette organisation, ces commandes ainsi que les programmes qui exploitent ces commandes.

Il y a 2 manières d'accéder aux données, assez différentes : **l'accès direct** et **l'accès séquentiel.** Ces deux techniques sont assez différentes dans leur usage comme dans leurs performances.
L'accès séquentiel est d'utilisation simple mais peu performant. L'accès direct est d'accès plus ardu, mais il est très performant.

Pour simplifier les syntaxes, à chaque fichier est associé un numéro appelé **numéro logique** (NL en abrégé).
Le numéro logique est un nombre de 0 à 63 : on peut donc ouvrir jusqu'à 64 fichiers simultanément, s'il y a assez de place en mémoire évidemment.

Il faut dans un premier temps **ouvrir** un fichier, c'est à dire lui réserver un numéro logique et lui associer un nom de fichier sur la disquette.. Le DOS charge à ce moment toutes les informations nécessaires et crée éventuellement le fichier sur la disquette s'il n'existait pas. Le fichier étant ouvert, toutes les opérations de lecture ou d'écriture sont possibles.
A la fin de "l'utilisation" du fichier, il convient de **fermer** le fichier, c'est à dire de libérer le numéro logique. Si cette opération est oubliée, aucune perte d'information n'est à craindre.

**ATTENTION** : il est recommandé de fermer puis de réouvrir un fichier après des erreurs du type "WRITE PROTECTED" ou "DISK FULL ERROR" car il risque d'y avoir incohérence entre le fichier en mémoire et celui sur la disquette.

## B / ACCES SEQUENTIEL

Dans les fichiers séquentiels, on accède aux données dans l'ordre dans lequel elles ont été écrites.
On peut imaginer le travail en séquentiel comme un pointeur qui se déplace tout au long du fichier , toujours en avançant d'une donnée à la suivante. Les lectures ou écritures de variables se feront à la position du pointeur.
Pour faciliter le travail, des instructions permettant d'agir directement sur la position du pointeur ont été écrites.

Toutes les écritures sont directement reportées sur la disquette, il n'y a donc rien à craindre en cas de coupure de courant, ou d'oubli de fermer un fichier.

### OPEN S, NF, NL

Ouvre un fichier séquentiel.

- S'il ce fichier n'existe pas encore sur la disquette, il est automatiquement créé.

- Réserve un tampon dans la mémoire et place le pointeur au début du fichier.

Exemple : ***OPEN S, "DISQUES", 2*** ouvre un fichier séquentiel appelé DISQUES avec le numéro logique 2.

### CLOSE (NL, NL, ...)

Libère le tampon réservé par **OPEN en fermant le ou les fichiers ouverts**.

Exemple : ***CLOSE 34, 1*** ferme les fichiers 1 et 34
***CLOSE*** ferme tous les fichiers ouverts.

## **PUT** NL, liste de variables

Ecrit des variables dans un fichier.

- Les variables peuvent être du type réelles, entières ou chaines.

- L'écriture en fin de fichier se fait sans aucune contrainte (sauf celle de la place disponible sur la disquette).

- Lors de l'écriture au milieu d'un fichier, il faut impérativement que les variables écrites soient de même type que celles déja écrites à la position du pointeur. Les chaines seront complétées par des espaces à droite si elles sont plus courtes ou tronquées à droite si elles sont plus longues.

- La sauvegarde sur la disquette est faite à la fin de chaque **PUT** , cela pour limiter au maximum les risques de perte d'informations. En revanche, les opérations sont plus lentes.

Exemple : ***PUT 2, A$, "ESSAI", 12, A%*** écrit dans le fichier de NL 2, les variable A$, ESSAI, 12, et la variable numérique entière A.

## **TAKE** NL , liste de variables

Lit dans le fichier portant le numéro NL les variables précisées dans la liste.

- Bien entendu, il convient de lire des variables de même type qu'à l'écriture.

- Si la fin du fichier est atteinte pendant l'ordre TAKE, un message "END OF FILE ERROR" sera afiché.

## **APPEND** NL

Place le pointeur de fichier à la fin du fichier auquel on a affecté le n° logique NL

- Cette opération est assez lente.

Exemple : ***APPEND 2*** place le pointeur à la fin du fichier de NL 2.

## **REWIND** NL

Place le pointeur de fichier au début du fichier précisé.

Exemple : ***REWIND 3***

## **JUMP** NL, nombre d'enregistrements.

Déplace le pointeur de fichier du nombre d'enregistrements précisé.

- Si ce nombre est trop grand, le pointeur est placé à la fin du fichier, comme c'est le cas pour **APPEND.**

- Cette instruction est en fait assez lente, puisqu'elle agit de manière séquentielle.

Exemple : ***JUMP 2, 200***
***JUMP 12, A*12***

## **BUILD** NL

Permet de construire des fichiers séquentiels à partir du clavier.

-Les caractères entrés seront assemblés sous la forme de chaines de 200 caractères, et créeront ainsi un fichier séquentiel.

- Tous les 200 caractères tapés, le **SED** effectue automatiquement la sauvegarde sur le disque.

- On peut sortir prématurément en appuyant sur **Ctrl C** (la sauvegarde des derniers caractères tapés sera tout de même effectuée) .

- Le code ASCII 13 (**CR** ou **Return**) est automatiquement remplacé par la séquence 13 /10 (soit aller à la ligne) pour améliorer la lisibilité.

- **TOUS** les caractères entrés au clavier seront fidèlement reproduits dans le fichier, ce qui permet de faire de véritables animations (en mode texte) .

Exemple: ***BUILD 2***

NB ajoute les caractères tapés à la fin d'un fichier non vide, même si on a fait rewind.

## **TYPE** NL

Permet d'afficher le contenu d'un fichier séquentiel

- Les chaines sont affichées ... comme des chaines, les nombres étant quant à eux affichés en décimal.

- L'affichage commence à la position courante du pointeur de fichier, et ce jusqu'à la fin du fichier.

- A tout moment, la pression d'une touche stoppe l'affichage, appuyer alors sur espace pour recommencer, ou sur **Ctrl C** pour sortir.

- L'affichage est très rapide et permet, en association avec la commande **BUILD** de faire de véritables séquences animées.

Exemple: ***TYPE 2***

## LTYPE NL

Même commande que TYPE, mais avec sortie sur l'imprimante au lieu de l'écran.

Exemple: ***LTYPE 2***

## & (NL)

Retourne 0 (FALSE) si la fin de fichier est atteinte, et -1 (TRUE) dans le cas contraire.
**Attention** : les parenthèses ici font partie de la syntaxe des instructions &, et n'indiquent pas une option facultative.

Exemple : ***1000 A=&(2)***
***1010 PRINT A***

## & (-NL)

Retourne le type du prochain enregistrement à lire.

- Le code retourné est le suivant :
  0 si c'est une variable réelle.
  128 si c'est une chaine
  -1 si la fin du fichier est atteinte

Exemple : ***A = & (-2)***

## C/ ACCES DIRECT

L'accès direct est très différent de l'accès séquentiel : cette fois, les données sont regroupées en **FICHES** . Chaque fiche contient toutes les informations sur un article donné par exemple. Toutes les fiches contiennent le même type de renseignements. Une fiche est donc divisée en plusieurs zones de données appelées **CHAMPS.**

Pourquoi accès direct ? Parce que l'accès à une fiche se fait directement, sans lire ou sans faire défiler toutes les fiches précédentes.
Une fiche est considérée comme une entité qui est écrite ou lue d'un bloc sur la disquette.
Pour faciliter le travail sur une fiche, On a recours au travail dans une zone de mémoire-tampon ou BUFFER, qui contient la totalité de la fiche considérée.

Les instructions seront donc de 2 types :

- Les transferts entre disquette et Buffer (Disquette ⇒ buffer en lecture et buffer ⇒ disquette en écriture)

- Les transferts de variables dans le buffer (Variables ⇒ buffer en écriture et buffer ⇒ variable en lecture.

L'accès à une fiche ne nécessite qu'un accès disque en lecture ou 2 accès disques en écriture. L'accès à une fiche ne demande que 0.2 à 1 seconde maximum en lecture.

## OPEN R, NF, NL (, longueur de la fiche , nombre de fiches à réserver)

Ouvre un fichier en accès direct.

- Lors de la première ouverture, il est indispensable de préciser la longueur de la fiche et le nombre de fiches à réserver. Il est interdit par contre de préciser ces options lors des ouvertures futures.

- Si le nombre de fiches dépasse celui initialement prévu, la place nécessaire sera automatiquement créée.

- La place globale occupée par une fiche est indiquée en nombres de secteurs par (Nbre de fiches x Longueur) / 256 , soit environ 600 secteurs pour un fichier de 3000 fiches de 50 caractères.

Exemple : ***OPEN R,"DIRECT", 2,183,100*** Ouvre un fichier appelé "DIRECT", où chaque fiche fait 183 caractères maximum, et réserve la place pour 100 fiches .

## CLOSE (NL ,NL..... )

- Ferme le ou les fichier de numéro logique NL et libère le buffer.

Exemple : ***CLOSE***

## TRANSFERTS BUFFER <---> DISQUETTE

## **TAKE** NL, N° de fiche

Charge dans le tampon la fiche considérée.

- L'accès à une fiche nécessite 0.2 secondes à 1 seconde maximum.

- Si la fiche n'existe pas, un message "BAD RECORD NUMBER " sera affiché.

Exemple : ***TAKE 2, 120***

***TAKE F, JP*12***

## **PUT** NL, N° de fiche

Transfère le contenu de la fiche considérée du tampon dans le fichier NL sur la disquette.

- Si la fiche n'existe pas, elle sera créée.

- Créér une fiche n° 1000 revient à créer les fiches 0, 1, 2 ... 999 et 1000. Il est recommandé de créer plusieurs fiches d'un coup pour avoir un fichier homogène, qui sera plus rapide à gérer.

- La création d'une ou plusieurs nouvelles fiches en fin de fichier est une opération assez longue car elle demande une recherche dans le catalogue etc..

Exemple : ***PUT 2, 120***

## TRAVAIL SUR LE BUFFER

Pour faciliter le transfert des variables dans le buffer, la fiche est divisée en un certain nombre de zones appelées **CHAMPS**, qui sont définis par leur nom.

Les champs permettent de définir en l'identifiant à quel endroit de la fiche telle ou telle variable va être sauvée.

Les champs sont un peu comme des variables : on peut leur affecter une valeur, et les lire, de la même manière ou presque que l'on peut le faire avec une variable. Attention, malgré une grande ressemblance de syntaxe, **les champs ne sont pas des variables !**

Voici les conventions adoptées pour les noms de champs (en abrégé NC) :

- 5 caractères significatifs.
- Peut être un pseudo-tableau (index de 0 à 255), mais ce n'est qu'une simple notation.
- Nom avec index (0) = nom sans index.
- Comme une variable, un nom de champ ne doit pas comporter de mot-clé du BASIC, et ne doit pas commencer par un mot-clé du DOS
- Le nombre total de champs n'est limité que par la place mémoire.

Exemples : ***NOM***
***INDEX (I)***
***PRENOM (0)*** identique à ***PRENOM*** et même à ***PRENO*** car le M n'est pas significatif

Nom de Champ

## **FIELD** NL, NC **TO** type (, ...)

Permet de définir les champs à l'intérieur du tampon. Le type est défini par :

**$ EN** pour un champ alphabétique (EN définit sa longueur)
**%** pour un champ entier
**!** pour un champ réel.
**O** pour un champ octet.

- Si l'instruction se termine par une virgule, la prochaine définition de champ commencera à la position courante (s'il s'agit du même fichier naturellement).

- La longueur du champ ne peut pas excéder la longueur totale de la fiche.

- La longueur d'un champ réel est de 5 octets, celle d'un champ entier 2 octets, celle d'un champ octet un octet et celle d'un champ alphanumérique la longueur précisée. Il faut de plus rajouter 2 octets par champ pour les information de gestion interne (sauf pour les fichiers de type DISQUE , voir chapitre suivant).

Exemple : ***FIELD 2, NOM TO $12, PRENOM TO $8, AGE TO %, SALAIRE TO R, SEXE TO O*** ouvre dans le fichier de numéro logique NL un champ pour le nom de 12 caractères alphanumériques, un de 8 caractères alpha pour le prénom, un champ entier pour l'âge, un champ réel pour le salaire et un champ octet pour le sexe.
Longueur totale : 2+12 + 2+8 + 2+2 + 2+5 + 2+1 = 38 octets par fiche.

## F/ TRANSFERTS CHAMPS <--> VARIABLES

### **RSET** NC < Expression

Ecrit la valeur donnée dans le champ.

- Le champ et la valeur doivent être de même type.

- Les variables alphanumériques seront justifiées à droite et éventuellement tronquées à gauche si la chaine est plus longue que le champ, ou complétées par des espaces si la chaine est plus courte que le champ.

Exemple : ***RSET NOM (2) < "EUREKA"*** (NOM (2) est un champ alphanumerique)
***RSET ESSAI < 123*** ( ESSAI est un champ de type réel).

Si le champ chaine a 5 caractères par exemple,

| Instruction | Valeur stockée |
|---|---|
| ***RSET CHAINE < "123456"*** | ***"23456"*** |
| ***RSET CHAINE < "TOTO"*** | ***" TOTO"*** |

### **LSET** NC < Expression (, ...)

Identique à **RSET** pour les champs réels ou entiers.

- Les variables alphanumériques seront justifiées à gauche et tronquées à droite si elles sont trop longues, ou complétées à droite par des espaces si elles sont plus courtes que le champ.

Exemple : ***LSET PAPA < 2000 * AS***

Si le champ chaine a 5 caractères :

***LSET CHAINE < "TOTO"*** stockera ***"TOTO "***
***LSET CHAINE < "123456"*** stockera ***"12345"***

## LECTURE DE CHAMPS

### NC > Variable

Lit le champ de nom NC et affecte la valeur lue à la variable en question.

- Bien entendu, le champ et la variable concernés doivent être de même type.

Exemple : ***NOM (I) > A$***
***ESSAI > A***

**Attention :** ***PRNOM > A*** **est invalide car PR est un mot-clé du DOS.**

### & (NL)

Affiche le nombre de fiches d'un fichier de numéro logique NL

Exemple : ***& (12)***

# D / ACCES DISQUE

Il s'agit d'un type de fichier assez particulier, qui permet aux utilisateurs chevronnés (ou curieux !), de modifier directement la disquette, secteur par secteur, et ce directement en BASIC.
Ce type de fichier s'apparente aux fichiers directs pour ce qui est du travail dans le tampon (voir les instructions qui permettent de travailler sur le tampon, FIELD, LSET, RSET et >). Attention, contrairement à l'instruction FIELD, les champs sont ici définis sans la séparation de 2 octets entre eux.

Cette fois, ce n'est plus une fiche qui est dans le tampon, mais le contenu d'un secteur de la disquette. Le tampon réservé fait donc une longueur de 256 octets.

Pour faciliter le travail, des instructions permettant de rechercher des secteurs libres sur la disquette ont été rajoutées.

## OPEN D , NL (, Lecteur)

Reserve un tampon.

- La longueur du Buffer réservé est de 256 octets.

Exemple : ***OPEN D, 1***

***OPEN D, 1, B***

**ATTENTION:** l'espace entre **OPEN** et le D est nécessaire pour éviter la présence du mot-clé BASIC **END**

## CLOSE (NL (,NL . . . ))

Libère le numéro logique

## **TAKE** NL, Piste, Secteur (, lecteur)

Prend dans le tampon le secteur de coordonnées précisées.

- Aucun test d'existence du secteur n'est effectué. Si le secteur n'existe pas, une I/O ERROR de type 10 est générée.

Exemple : ***TAKE 2, 14,3, A***
***TAKE 0, 20,1***

## **PUT** NL, PISTE, SECTEUR, (,lecteur)

Instruction semblable à **TAKE,** mais écrit le secteur au lieu de le lire.

**Quand on dispose d'un lecteur double face, pour accéder à une piste de la 2' face avec ces 2 instructions, ajouter 128 au n' de piste.**

Exemple : ***PUT 1,131,2*** écrit le tampon dans le secteur 2 de la piste 3 de la face B.

## RESERVATION DE SECTEURS

Il est possible de chercher des secteurs libres sur une disquette, ainsi que de libérer certains secteurs.
Il faut savoir que toutes ces informations sont contenues dans un secteur spécial (Piste 20, secteur 2), appelé **BIT MAP**.
Pour réserver un secteur, il faut donc lire le BIT MAP, allouer le secteur, et réécrire le Bit Map pour mettre à jour la disquette.

### **PMAP** Lecteur

Lit en mémoire le BIT MAP du lecteur considéré.

Exemple : ***PMAP***
***PMAP A***

### **SMAP** Lecteur

Ecrit sur la disquette le BIT MAP en mémoire.

- Entre un **PMAP** et un **SMAP**, il ne doit pas avoir eu de **LOAD, SAVE, DIR,** d'accès séquentiel ou direct, sous peine de rendre le BIT MAP incohérent.

Exemple : ***SMAP A***

### **CRESEC**

Retourne dans les variables FP et FS l'adresse piste et secteur d'un secteur libre. S'il n'y a plus de place, affiche "DISK FULL"

- Il faut être sur qu'un PMAP a été exécuté.

- décrémente automatiquement le nombre de secteurs libres.

### **FRSEC** EN $_{piste}$ , EN $_{secteur}$

Libère le secteur désigné et incrémente automatiquement le nombre de secteurs libres. S'il était déja libre, n'a aucun effet.

- Le BIT MAP doit être à jour...

Exemple : ***FRSEC 20, 10***

# Chapitre 7

# Annexes

Manuel d'utilisation du MICRODISC ORIC

# Annexe 1 : INDEX DES MOTS CLES

# Annexe 2 : MESSAGES D'ERREURS

nº ds variable EN voir p 59

## 01 - FILE NOT FOUND

Le fichier n'a pas été trouvé sur la disquette (n'existe pas, ou NF mal orthographié, ou joker mal placé ..)

## 02 - DRIVE NOT IN LINE

Il a été fait appel à un lecteur non connecté. Vérifier le n° de lecteur. Pour changer celui-ci, utiliser la commande SYS.

## 03 - INVALID FILE NAME

Le nom du fichier comportait des caractères alphanumériques, ou était trop long.

## 04 - DISK I/O

L a disquette a été endommagée.
Le DOS affiche aussi le secteur (Sector :) et la piste (Track :)ainsi que le code de l'erreur, pour permettre aux utilisateurs chevronnés de récupérer une disquette endommagée.
N.B. : le numéro du secteur n'étant pas significatif lors d'un formatage, il n'est pas affiché si une erreur se produit pendant cette opération.

## 05 - WRITE PROTECTED

Tentative d'écriture sur une disquette dont la languette de protection a été tirée. Il est d'ailleurs conseillé d'observer cette précaution sur les disquettes de sauvegarde.

## 06 - WILDCARD(S) NOT ALLOWED

Des JOKERS ont été employés dans un nom de fichier utilisé avec une commande n'acceptant pas les jokers (**LOAD** etc..)

## 07 - FILE ALREADY EXISTS

On a tenté de sauvegarder un fichier sur une disquette où un fichier portant le même nom existe déja.

## 08 - DISK FULL

Le fichier est trop long compte tenu de la place disponible sur la disquette.

## 09 - ILLEGAL QUANTITY

Un parametre trop grand ou trop petit a été précisé.

## 10 - SYNTAX ERROR

En un mot comme en 100, erreur de syntaxe !

## 11 - UNKNOWN FORMAT
Tentative d'opération sur une disquette qui n'a pas été initialisée par SEDORIC.

## 12 - TYPE MISMATCH ERROR
Dans une entrée, une variable alpha a été tapée là ou une variable numérique était prévue et vice-versa.

## 13 - FILE TYPE MISMATCH
On a appelé un fichier d'un certain type avec un nom associé à un fichier d'un autre type (Séquentiel au lieu de direct, essayer de charger un fichier de données avec un **LOAD** etc..)

## 14 - FILE NOT OPEN
Tentative d'opération ( autre que **OPEN**) sur un fichier qui n'a pas encore été ouvert.

## 15 - FILE ALREADY OPEN
Tentative d'allouer (par **OPEN**) un numéro logique déja utilisé.

## 16 - END OF FILE
On a essayé de lire une donnée alors que la fin du fichier est atteinte.

## 17 - BAD RECORD NUMBER
Le numéro d'enregistrement de la fiche dépasse la capacité autorisée du fichier.

## 18 - FIELD OVERFLOW
Lors de la définition des champs, la longueur totale de la fiche a été dépassée.

## 19 - STRING TOO LONG
La chaine entrée est trop longue (voir syntaxe de **TKEN** etc..)

## 20 - UNKNOWN FIELD NAME
Tentative d'affectation ou de lecture d'un champ non encore défini par **FIELD.**

Valeurs utilisateur de 50 à 255 voir page 59 et 60

# Annexe 3 : LES VARIABLES RESERVEES

Ce sont les variables dont la valeur est définie par système, pour connaitre un Status, la localisation d'une erreur etc.. On en connait la valeur en affichant la variable. Il est donc recommandé de ne pas utiliser ces noms pour définir des variables dans vos programmes.

Exemple: ***100 ERRGOTO 1000***
*****
***1000 PRINT "Erreur"; EN,"à la ligne";ER***

## Mises à jour après une erreur :

**EN** (Error Number) affiche le numéro de l'erreur (voir dans la table annexe )
**EL** (Line number) affiche le numéro de ligne où s'est produite l'erreur.

## Mises à jour par SEARCH

**EF** (Existing File) : Retourne EF = 1 si le fichier existe et 0 s'il n'existe pas.

## Mises à jour par LOAD ou par chargement direct

**ST** (STart adress) : indique l'adresse de début d'un fichier.
**ED** (EnD adress) : Indique l'adresse de fin d'un fichier.
**FT** ( File Type) : Type du fichier chargé.
**EX** (EXecution address) : Adresse d'exécution du fichier chargé.

## Mises à jour par SEEK

**SK** (SeeK) : affiche le nombre d'occurences de la chaine. appelée par **SEEK.**

## Mises à jour par INSTR

**IN** (INstr) : affiche la position de la sous-chaine dans la chaine.

## Mises à jour par USER

**RA** : affiche la valeur du registre A du microprocesseur
**RX** : affiche la valeur du registre X
**RY** : affiche la valeur du registre Y
**RP** : affiche la valeur du registre d'état.

## Mises à jour par HCUR et LCUR

**CX** : abscisse du curseur texte ou graphique
**CY** : ordonnee du curseur texte ou graphique

## Mises à jour par PRESEC

**FP** : numero de la piste où le secteur a été libéré
**FS** : numéro du secteur libéré.

# Annexe 4 : CODAGE D'UN FICHIER

L'octet d'état (STATUS Byte) d'un fichier est en fait un code binaire établi selon les règles ci-dessous :

L'Octet est représenté sous sa forme classique :

b7 b6 b5 b4 b3 b2 b1 b0

Le bit est actif s'il est à 1.

**b0 : exécution automatique**

**b1** : inutilisé

**b2** : inutilisé

**b3 : direct**

**b4 : séquentiel**

**b5 : window** (b6 = 1 aussi)

**b6 : bloc de données**

**b7 : Basic**

# Annexe 5 : STRUCTURE DE LA DISQUETTE

Les premiers secteurs de la disquette (en partant du secteur 1 de la piste 0 ) sont occupés par les routines de Boot et par le DOS (pour une disquette MASTER).

Les secteurs suivants ont une place fixe et sont réservés lors d'une opération d'initialisation :

| Piste | Secteur | Utilisation |
|---|---|---|
| **20** | **01** | **Entête** (nom de la disquette) |
| **20** | **02** | **Bit Map** |
| **20** | **03** | **Réservé** |
| **20** | **04** | **Catalogue 1** |
| **20** | **07** | **Catalogue 2** |
| **20** | **10** | **Catalogue 3** |
| **20** | **13** | **Catalogue 4** |
| **20** | **16** | **Catalogue 5** |

Les autres secteurs seront alloués au fur et à mesure des besoins.

# Annexe 6 : CODES DES TOUCHES DE FONCTIONS

En association avec la commande **KEYDEF,** des codes permettent d'associer n'importe quelle commande à n'importe quelle touche du clavier. Le chiffre qui précède chaque définition est celui précisé en argument (EN) derrière **KEYDEF.**

**000 à 015** **Définitions utilisateur**

| | |
|---|---|
| **016** | **HIRES + RETURN** |
| **017** | **TEXT + RETURN** |
| **018** | **LIST + RETURN** |
| **019** | **RUN + RETURN** |
| **020** | **LPRINT + RETURN** |
| **021** | **FOR I = 1 TO** |
| **022** | **CURSET 120, 100, 1 + RETURN** |
| **023** | **CTRL A 6 fois** |
| **024** | **EXT + RETURN** |
| **025** | **NUM END + RETURN** |
| **026** | **SEEK + RETURN** |
| **027** | **RENUM + RETURN** |
| **028** | **OLD + RETURN** |
| **029** | **DIR + RETURN** |
| **030** | **CTRL T ou CHR$ (20)** |
| **031** | **© ou CHR$ (96)** |
| **032 à 127** | **Mots-clés du DOS** (Voir page suivante) |
| **128 à 253** | **Mots-clés du BASIC** (Voir manuel de l'ATMOS). |
| **254** | **DEL** |
| **255** | **Génération des numéros de lignes.** |

| Commande | Code | Commande | Code | Commande | Code |
|---|---|---|---|---|---|
| ACCENT | 32 | INIT | 66 | RENUM | 102 |
| APPEND | 33 | INSTR | 67 | RESET | 100 |
| AZERTY | 34 | JUMP | 69 | RESTORE | 106 |
| BACKUP | 37 | KEY | 76 | RESUME | 99 |
| BOX | 36 | KEYDEF | 73 | REWIND | 101 |
| BUILD | 38 | KEYIF | 71 | RSET | 100 |
| CHANGE | 39 | KEYSAVE | 75 | SAVE | 117 |
| CLOSE | 40 | KEYUSE | 72 | SAVEM | 115 |
| COPY | 41 | LCUR | 86 | SAVEO | 116 |
| CREATEW | 42 | LDIR | 84 | SAVEU | 114 |
| CRESEC | 43 | LINE | 77 | SEARCH | 118 |
| DEL | 48 | LINPUT | 82 | SEEK | 109 |
| DELBAK | 47 | LOAD | 83 | SMAP | 120 |
| DELETE | 45 | LSET | 78 | STATUS | 113 |
| DESTROY | 46 | LTYPE | 85 | STRUN | 111 |
| DIR | 49 | LUSING | 80 | SWAP | 108 |
| DKEY | 53 | MERGE | 88 | SYS | 119 |
| DNAME | 52 | MOVE | 87 | SYSTEM | 112 |
| DNUM | 51 | NUM | 89 | TAKE | 122 |
| DSYS | 54 | OLD | 91 | TKEN | 121 |
| DTRACK | 55 | OPEN | 92 | TRACK | 124 |
| ERR | 60 | OUT | 90 | TYPE | 123 |
| ERRGOT | 57 | PMAP | 96 | UNPROT | |
| ERROR | 59 | PR | 95 | UNTKEN | 126 |
| ESAVE | 61 | PROT | 94 | USER | 125 |
| EXT | 62 | PUT | 93 | USING | |
| FIELD | 63 | QUIT | 97 | USR | |
| FRSEC | 64 | QWERTY | 98 | VUSER | |
| HCUR | 65 | RANDOM | 105 | WIDTH | |
| INIST | 68 | REN | 103 | WINDOW | |

## Annexe 7

# CODAGE DES TOUCHES DU CLAVIER

## POUR L'INSTRUCTION KEYIF

## Annexe 8 : PASSAGES RAM ⇔ ROM

Le DOS étant en RAM Overlay, il peut être intéresant de savoir comment passer de la ROM à la RAM et vice-versa.

Il suffit, lorsqu'on est sur la ROM, de faire **JSR #04F2** pour passer sur la RAM, un autre **JSR #04F2** reviendra sur la ROM.

Il faut utiliser ce sous-programme sous peine de "plantage" lors du retour au BASIC. Notons que l'appel de cette routine n'affecte aucun registre.

# Annexe 9 : EXTENSION DE LA TABLE DES MOTS-CLES

Il a été prévu de rallonger à l'infini le vocabulaire du DOS, et ce sans le point d'exclamation.
Vous devrez simplement faire un programme qui doit reconnaitre votre ou vos mots-clés. Si aucun n'est reconnu, il faut sortir par **RTS, TXTPTR (#E9-#EA)** étant inchangé, et sinon par **PLA : CLC : ADC #$03 :PHA, TXT PTR** pointant un séparateur ( " : " ou " 0 " ).
Pour activer votre sous-programme, il suffit de modifier le vecteur **#4E9.**
Exemple : Le mot-clé "S" doit effectuer un "**SHOOT**".

```
            ORG   ??????

INTERP      LDY   #$00
            LDA   ($E9),Y
            CMP   #"S"
            BNE   PASHOT
            INY
            LDA   ($E9),Y
            BEQ   SHOOT
            CMP   #":"
            BNE   PASHOT
SHOOT       JSR   $FAB5     (ORIC-1 $FA9B)
            PLA
            CLC
            ADC   #$03
            PHA
            JMP   $00E2

PASHOT      RTS

            ORG   $04E9

            JMP   INTERP
```

# ANNEXE 10 / QUELQUES VARIABLES SYSTEMES

| ADRESSE | NOM | SIGNIFICATION |
|---|---|---|
| #04FD | ERROR | numéro de l'erreur |
| #04FE-F | NOLIGN | numéro de la ligne de l'erreur |
| #00-#0C | | zone de travail (RENUM, fichiers) |
| #F2-#F9 | TRAV0-TRAV7 | zone de travail (travail DOS) |
| #C000 | DRIVE | numéro de lecteur |
| #C001 | PISTE | numéro de piste (b7=1 si face B) |
| #C002 | SECTEUR | numéro du secteur |
| #C003-4 | RWBUF | adresse de chargement du secteur |
| #C009 | DRVDEF | numéro du lecteur par défaut |
| #C00A | DRVSYS | numéro du lecteur système |
| #C00D-E | EXTER | adresse messages d'erreur externe |
| #C00F-10 | EXTMS | adresse messages externes |
| #C015 | EXTNB | numéro du bloc externe |
| #C01D-1E | ERRVEC | adresse de traitement des erreurs |
| #C023 | SAUVES | sauvegarde du pointeur de pile ( si sortie par erreur) |
| #C024 | ATMORI | 0 (ROM V1.0) ou #80 (ROM V1.1) |
| #C025 | POSNMP | piste du nom cherché dans le catalogue |
| #C026 | POSNMS | secteur idem |
| #C027 | POSNMX | position dans le secteur idem |
| #C028 | BUFNOM | nom du fichier à chercher |
| #C039-3C | TABDRV | table d'activation des lecteurs |
| #C03D | MODCLA | mode clavier (b6=accent, b7=AZERTY ) |
| #C03E-3F | DEFNUM | origine par défaut (NUM,RENUM) |
| #C040-41 | DEFPAS | pas par défaut (NUM,RENUM) |
| #C04C | DEFAFF | code ASCII devant les nombres décimaux |
| #C04D | VSALO0 | code pour LOAD/SAVE |
| #C04E | VSALO1 | code pour LOAD/SAVE |
| #C051 | FTYPE | type du fichier chargé |
| #C052-53 | DESALO | adresse de début du fichier |
| #C054-55 | FISALO | adresse de fin du fichier |
| #C056-57 | EXSALO | adresse d'exécution du fichier |

C075 Sauvegarde du caractère pour LINPUT

# ANNEXE 11 / QUELQUES VECTEURS DU DOS

Les lettres A, X, Y, Z, V, S correspondent aux registres ou indicateurs du 6502 .

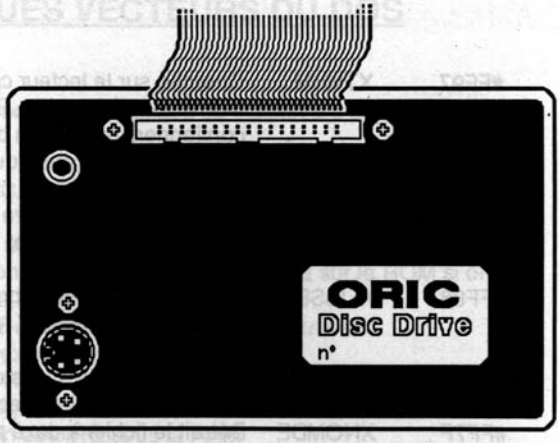
| ADRESSE | NOM | SIGNIFICATION |
|---|---|---|
| #04E9 | DETE2C | Vecteur utilisateur (rajout mots clés) |
| #04EC | EXERAM | Exécution d'une routine sur la RAM, ou sur la ROM si on est sur la RAM<br>EXEVEC+1 contient l'adresse de la routine |
| #04EF | EXEVEC | vecteur exécution, voir EXERAM |
| #04F2 | RAMROM | bascule RAM/ROM<br>aucun registre n'est affecté |
| #04F5 | IRQRAM | exécution des IRQ sous la RAM |
| #04F8 | NMIRAM | exécution des NMI sur la RAM |
| #FFC4 | XRWTS | accès à la routine de gestion des lecteurs<br>X contient la commande (conformément à la notice du 1793). En sortie, Z=1 si pas d'erreur,Z=0 sinon. V=1 si la disquette est protégée en écriture.<br>DRIVE, PISTE, SECTEUR et RWBUF doivent être à jour . |
| #FFC1 | XPMAP | prend le secteur de bit map, vérifie le format |
| #FFBE | XPBUF1 | Charge dans le BUFFER1 le secteur dont la piste est contenue dans A et le secteur dans Y . |
| #FFB8 | XPRSEC | Lis un secteur, conformément aux variables systèmes DRIVE, PISTE, SECTEUR, RWBUF |
| #FFB5 | XSMAP | sauve le secteur de bitmap sur la disquette |
| #FFB2 | XSCAT | sauve le secteur de catalogue contenu dans BUF3, selon POSNMP et POSNMS . |
| #FFAF | XSBUF1 | Sauve BUF1 à la piste A et le secteur Y . |
| #FFAC | XSAY | sauve le secteur indiqué par RWBUF à la piste A et le secteur Y . |
| #FFA9 | XSVSEC | Ecrit un secteur, selon DRIVE, PISTE, SECTEUR, RWBUF |
| #FFA6 | XVBUF1 | Remplit de 0 le BUFFER1 |
| #FFA3 | XBUCA | Transfère le nom de fichier contenu dans BUFNOM dans le secteur de catalogue contenu dans BUF3, à la position POSNMX . |
| #FF9D | XCABU | Tranfère dans BUFNOM le nom de fichier contenu dans le secteur de catalogue placé dans BUF3 , à la position POSNMX. |

| | | |
|---|---|---|
| #FF97 | XTVNM | Cherche sur le lecteur courant le nom contenu dans BUFNOM . A la sortie, POSNMX, POSNMP et POSNMS contiennent la position du nom dans le catalogue, et Z=1 si le fichier n'est pas trouvé |
| #FF8E | XTRVCA | Cherche une place libre dans le catalogue. A la sortie, POSNMX, POSNMP et POSNMS indiquent la position de la place réservée |
| #FF88 | XLIBSE | Cherche un secteur libre sur le bitmap courant. en sortie, A contient la piste et Y le numéro de secteur du secteur réservé . |
| #FF85 | XDETSE | Libére le secteur Y, piste A sur le bitmap courant. |
| #FF82 | XCREAY | Crée une table piste secteur de AY secteurs . |
| #FF7F | XNOMDE | Détruit le fichier indexé par POSNMX, dont le secteur de catalogue est dans BUF3 . (en fait, tout est positionné comme après un XTVCAT . |
| #FF7C | XSAVEB | Sauve le fichier de nom contenu dans BUFNOM, selon VSALO0, VSALO1, DESALO, FISALO, EXSALO . |
| #FF79 | XDEFLO | Positionne les valeurs par défaut pour XLOADA |
| #FF76 | XDEFSA | Positionne les valeurs par défaut pour XSAVEB ( en fait, positionne pour sauver le programme BASIC) . |
| #FF73 | XLOADA | Charge le programme dont le nom est dans BUFNOM , selon VSALO0, VSALO1, DESALO . |
| #FF6D | XROM | Permet d'exécuter à partir de la RAM une routine ROM . Le JSR XROM doit être suivi dans l'ordre de l'adresse de la routine pour la V1.0, puis de l'adresse pour la V1.1 . |
| #FF6A | XAFSTR | Affiche la chaine terminée par un 0 et dont l'adresse est donnée par AY . |
| #FF67 | XAFHEX | Affiche en hexadécimal le contenu de A . |
| #FF64 | XAFCAR | Affiche comme un caravtère ASCII le contenu de A . |
| #FF52 | XAFSC | Affiche le X ème message terminé par un caractère codé "+128" . EXTMS doit contenir l'adresse -1 du premier message . |
| #FF4F | XNFA | Saisit dans le texte indexé par TXTPTR un nom de fichier ambigu . Le résultat est dans BUFNOM . |
| #FF4C | XNF | Idem mais pour un nom de fichier non ambigu . |
| #FF49 | XCRGOT | Saisit dans A le caractère courant (identique à CHRGOT du BASIC, mais en plus convertit automatiquement les minuscules . |
| #FF46 | XCRGET | Saisit dans A le caractère suivant . |
| #FF43 | XLINPU | Appel à la routine LINPUT . TRAV0 contient la longueur de la chaine à saisir, au retour TRAV2 contient le mode de sortie et #D0-#D1 donne l'adresse de début de la chaine . |

# ANNEXE 12 / SCHEMAS DE BROCHAGE

## LECTEUR MASTER :

Le câble qui sert à raccorder le lecteur-maitre à l'unité centrale est celui qui sort de l'arrière du boitier du lecteur (Cable en nappe gris).
La prise AMPHENOL 34 broches sur la face arrière sert à raccorder les lecteurs esclaves: **c'est une sortie** après controleur.
Le bouton rouge est le bouton de RESET qui réinitialise le système et "boote" la disquette MASTER lors de l'allumage.




## LECTEUR ESCLAVE :

Sur le lecteur esclave, la prise AMPHENOL 34 broches sert à raccorder le lecteur esclave au lecteur maitre (sur la prise chassis) ou au lecteur esclave précédent (sur la prise sertie sur le câble) : **c'est une entrée.**

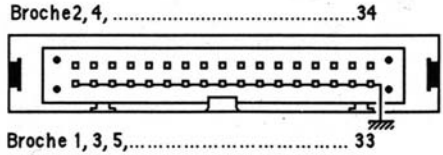

## BROCHAGE DE LA PRISE D'ALIMENTATION

Brochage de la prise vue côté extérieur:

1 : Masse
2 : non connecté
3 : +12 Volts
4 : + 5 Volts.

## Schéma du connecteur d'extension du lecteur

Vue côté extérieur :




Les broches impaires sont reliées à la masse.

6 : Drive select 3
8 : Index
10 : Drive select 0
12 : Drice select 1
14 : Drive select 2
16 : Motor ON
18 : Direction
20 : STEP
22 : Write DATA
24 : Write Gate
28 : Write protect detect.
30 : Read DATA
32 : Side select.

Achevé d'imprimer
par Collet Associés Imprimeurs
en décembre 1985


ORIC
INTERNATIONAL