Société de Micro-informatique et Télécommunications

goupil

goupil

Société de Micro-informatique et Télécommunications

4.
MANUEL D' UTILISATION DU DOS FLEX-GOUPIL 2

1982

NOTE

Les éléments qui suivent, listings et documentations sont produits pour la satisfaction et l'usage personnel par la Société de Micro-informatique et Télécommunications SMT.

Toute reproduction, même partielle, est interdite et constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1975 sur la protection des droits d'auteur.

L'application de cette règle permettra de vendre de plus en plus de programmes à des prix de plus en plus bas.

Son respect est la condition nécessaire pour que la SMT, ou d'autres sociétés, puissent produire, acheter ou adapter toujours plus de logiciels de qualité et que la créativité en ces matières se développe à votre profit.

S O M M A I R E

Chapitre 4

AVERTISSEMENT

Le présent manuel décrit la dernière version G2FLEX du système d'exploitation FLEX à la date du 1er décembre 1981 pour le micro-ordinateur GOUPIL2 équipé de 64 Ko de mémoire.

Dans cette version FLEX est chargé en mémoire centrale de manière à offrir le maximum de confort à l'utilisateur.

L'objet du manuel est de fournir toutes les informations nécessaires pour utiliser votre micro-ordinateur et ses disquettes, qu'il s'agisse de 5 ou de 8 pouces.

Il est conseillé aux amateurs néophytes de lire à plusieurs reprises le mini-glossaire du début afin de se familiariser avec le vocabulaire courant utilisé dans la suite du manuel.

MINI-GLOSSAIRE

moniteur — Logiciel capable de saisir et d'interpréter les messages affichés sur l'écran. Ex. : si vous tapez H, le moniteur-étonné- vous répondra : H? (il ne s'agit pas d'un ordre qu'il comprend...). Dès que la machine est allumée, on se trouve "sous moniteur".

curseur — Signal lumineux apparaissant sur l'écran, (-) par exemple, mais cela peut être un autre signe, et qui indique la position du prochain caractère à imprimer.

disquette — ("floppy disk" en anglais) ; disque souple (8 pouces ou 5 pouces) contenant des programmes et constituant une mémoire de masse externe.

- . disquette vierge : la disquette telle qu'on l'achète dans le commerce.
- . disquette système : le G2FLEX qui permet de faire fonctionner le système. Pour GOUPIL, on place la disquette système dans le lecteur 0 (celui de gauche).
- . disquette travail : placée dans le secteur 1 (celui de droite), elle sert à stocker des informations et à charger des programmes
- . disquette formatée: disquette prête à l'emploi après formatage.

formatage — D'une disquette : opération (NEWDISK) par laquelle vous effectuez un repérage des pistes et des secteurs selon un format donné en les numérotant, de même que lorsque vous avez un paquet de feuilles, vous les numérotez pour vous y retrouver.

lecteur disque — ("drive" en anglais) : unité connectée à l'ordinateur dans laquelle on introduit les disquettes. GOUPIL 2 est équipé d'une unité de deux lecteurs respectivement appelés lecteur 0 (à gauche) et lecteur 1 (à droite).

programme — Ensemble d'instructions ordonnées permettant de traiter les informations et d'exécuter une tâche ou un ensemble de tâches déterminées.

DOS — (sigle de l'anglais "disk operating system") ; système d'exploitation de disques qui permet de gérer les disquettes. Il s'agit ici du G2FLEX.

commande utilitaire — Commande du système G2FLEX qui permet la manipulation d'une fonction précise s'exerçant sur les fichiers.

fichier — Ensemble organisé et structuré de données stockées sur disques. Chaque fichier porte un nom et se voit attribuer une "extension".

extension — Groupe de trois caractères maximum qui définit le type d'informations contenues dans le fichier (fichier texte-TXT), fichier binaire-BIN, fichier de commandes-CMD, etc.).

MINI-GLOSSAIRE (suite)

Lorsque l'on spécifie un fichier, de nombreuses commandes FLEX ne demandent pas l'extension de façon explicite. Si celle-ci est omise, la commande prendra "par défaut" l'extension correspondant au type de fichier sur lequel elle est prévue pour travailler.

écriture — Enregistrement d'informations sur la disquette. La "tête" du lecteur écrit physiquement sur la disquette qui tourne.

lecture — Lecture pour traitement, affichage ou impression de l'information sur la disquette.

DIFFERENTS TYPES DE FICHIERS : INTRODUCTION

. Quatre types principaux d'information peuvent être inscrits dans les fichiers de vos disquettes :

I) Fichiers de commandes :

-ces fichiers contiennent un ordre précis fournissant un service "commandé" à la machine ; ils prennent l'extension.CMD (commande).

II) Fichiers de données -ce sont des fichiers sur lesquels ne sont stockées que des données. Ils sont caractérisés par l'extension .DAT ou .TXT

III) Fichiers système - ces fichiers servent à gérer les matériels. Par exemple: le fichier G2FLEX gère le travail des lecteurs et des disquettes, le fichier .PRINT.SYS (que vous créez en fonction de l'imprimante dont vous disposez - voir p5) gère les imprimantes. Ces fichiers système sont transparents pour l'utilisateur dans la mesure où, après avoir chargé le système, il n'a plus à agir dessus.

IV) Fichiers de programmes d'application ou de jeux

- ce sont des fichiers destinés à une utilisation particulière : programmation en BASIC, déroulement de jeux,etc... L'utilisateur achète ou réalise lui même ces programmes.

I	II	III	IV
BASIC.CMD	.DAT	G2FLEX	.BAS(basic source)
CAT.CMD	.TXT	.PRINT.SYS	.BAC(basic compilé)
.EDIT.CMD		.PØKI.SYS	.TXT
etc.		.PDIABLO.SYS	.BIN

TABLEAU 1 : exemples d'extensions de fichiers tels qu'ils apparaîtront sur l'écran lorsque vous expérimenterez le système.

PREMIERE MISE EN ROUTE DU SYSTEME

Avant d'entrer précisément dans la description de toutes les possibilités offertes par votre système de gestion de disquettes, vous trouverez ci-après quelques conseils liminaires simples vous permettant de manipuler l'appareil et de le mettre en route, ainsi qu'un petit glossaire des termes les plus usuels qui vous aideront à progresser avec votre micro-ordinateur et votre manuel . Le premier acte consiste à mettre votre machine en route.

1. Connexion à GOUPIL

- Connecter le câble plat pour les disquettes 5 pouces sur la face arrière du GOUPIL sur la broche de sortie 26 points, notée "lecteur 5 pouces", le liseret rouge rouge se trouvant orienté vers le haut. Pour les disquettes 8 pouces, connecter le câble sur la broche 50 points - notée "lecteur 8 pouces".

- Mettre sous tension à l'aide de l'interrupteur situé à l'arrière du lecteur (connecter d'abord le câble d'alimentation).

- Mettre sous tension l'écran et le GOUPIL. Le + apparaît sur l'écran : le micro-ordinateur est prêt à fonctionner.

2. Mise en route du lecteur

- Vérifier que les lecteurs sont vides.

- Prendre la disquette G2FLEX et l'introduire dans le lecteur de gauche. (numéro 0; celui de droite portant le numéro 1)

Le sens d'introduction est tel que l'étiquette est orientée vers la gauche, la lunette de lecture est horizontale et située vers l'arrière.

- Fermer la porte et appuyez sur la touche  de votre clavier pour charger le G2FLEX.

Le système vous demande alors la date. Répondez par la date du jour et faites un retour chariot. Exemple:

6800 FLEX V3.0

DATE (JJ, MM, AA) ?

Tapez la date au clavier : par ex. 12,11, 81 pour 12 novembre 1981 suivi d'un retour chariot ↵ .

Vous êtes à présent sous FLEX: le signe +++ apparaît sur l'écran.

. Dans le cas où rien ne se passe : faites RESET (bouton poussoir situé sous le côté droit de GOUPIL) et recommencez la manipulation,

mais...ATTENTION!

- Ne pas faire RESET lorsque la petite lumière rouge sous le lecteur est allumée (fichier en cours d'écriture sur la disquette), sinon le contenu de celle-ci sera endommagé.

- Pour plus de sécurité, en fait, ne pas faire RESET avec des disquettes introduites dans les lecteurs.

3. Première utilisation du DOS G2FLEX : préparation de votre machine

En premier lieu, il est impératif de recopier votre disquette système. Ainsi, toute fausse manoeuvre ne risque que de détruire la copie et non l'original. Les opérations à mener pour ce faire sont les suivantes :

a) Formatage d'une disquette

- voir le glossaire → formatage
- introduire une disquette vierge dans le lecteur 1, la disquette système G2FLEX étant dans le lecteur 0, puis fermer les portes des lecteurs.
- charger G2FLEX comme indiqué précédemment, puis faire la commande :

```
+++ NEWDISK 1
```

Cette commande provoque le formatage de votre disquette vierge une fois que vous avez répondu aux questions posées (Ø pour OUI - N pour NON).

b) Copie de la disquette

- effectuer la comie de la façon suivante :

```
+++ CØPY 0 1   (disquette système toujours dans le lecteur 0, disquette
                                                    copie dans le lecteur 1)
```

- attendre la fin des opérations (les fichiers copies s'affichent les uns après les autres). Ensuite :

```
+++ LINK 1.G2FLEX    (opération spécifique à la disquette système)
```

Cette opération permettra par la suite au système, lorsque vous appuierez sur la touche [RESET] sous moniteur, d'aller chercher au bon endroit le DOS G2FLEX pour le charger en mémoire.

Remarque :

Lorsqu'il y a beaucoup de fichiers sur le disque que vous voulez recopier, il est plus pratique, pour copier une disquette, d'effectuer la commande :

```
+++ BACKUP 0 1
```

Cette commande est plus rapide que la précédente et vous n'avez plus besoin de faire la commande "LINK" lorsque vous recopiez la disquette système. BACKUP recopie toute la disquette et non seulement les fichiers existants comme le fait CØPY.

ATTENTION ! Encore une fois. Ne pas oublier de formater la disquette (commande NEWDISK) avant de faire BACKUP.

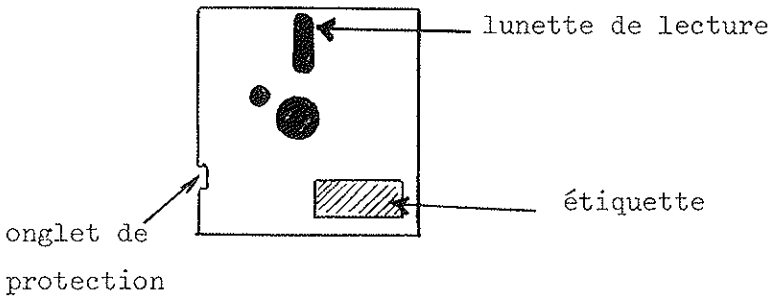
c) Protection de la disquette

Si vous souhaitez protéger votre disquette en écriture (par exemple si vous désirez interdire l'impression de programmes ou données sur la disquette) :

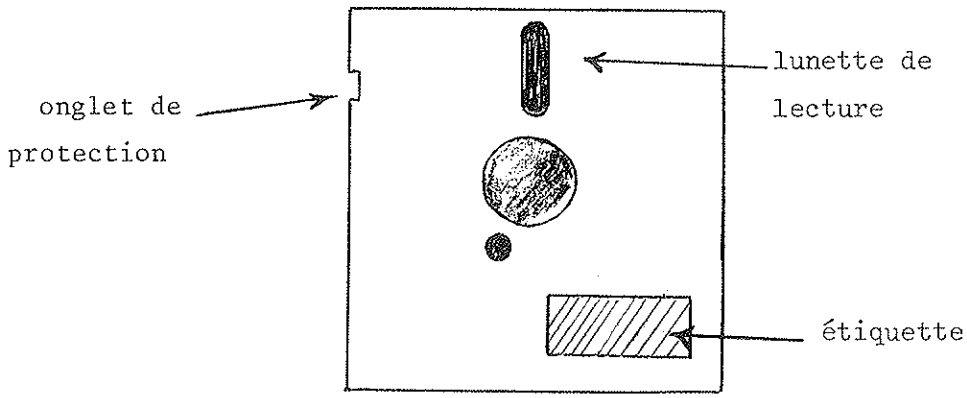
- pour une disquette 5 pouces → coller l'onglet de protection
- pour une disquette 8 pouces → retirer l'onglet de protection (si cet onglet existe, ce qui n'est pas toujours le cas pour les disquettes 8 pouces)

Schémas

.disquette 5 pouces



. disquette 8 pouces



d) Contenu de la disquette

Pour savoir ce qu'il y a sur vos disquettes, utilisez la commande CAT.

Exemple:

+++CAT ⌀ 0 (*) fournit le nom de tous les fichiers existants. sur la disquette introduite dans le drive 0, leur type et l'espace disque occupé.

- voir en page 4 bis la liste des fichiers de la disquette G2FLEX.

* ⌀ veut dire blanc

LISTE DES FICHIERS DE LA DISQUETTE SYSTEME G2FLEX (au 1.12.1981)

- NOM est le nom du fichier
- TYPE est l'extension du fichier telle que présentée dans le mini-glossaire
- TAILLE est l'importance du fichier mesurée en nombre de secteurs (un secteur contient 252 "octets" ou caractères utilisables)

```
CATALOGUE DU DISQUE Ø
DISQUE: NSMT   #Ø

 NOM      TYPE   TAILLE   PRT

GPFLEX   .SYS      25
ERRORS   .SYS       9
PDIABLO  .SYS       1
STARTUP  .TXT       1
ASN      .CMD       1
TTYSET   .CMD       2
CAT      .CMD       3
COPY     .CMD       5
LIST     .CMD       3
LINK     .CMD       1
RENAME   .CMD       1
PRINT    .CMD       2
PROT     .CMD       1
VERIFY   .CMD       1
DELETE   .CMD       2
P        .CMD       1
XOUT     .CMD       2
QCHECK   .CMD       3
APPEND   .CMD       3
O        .CMD       2
VERSION  .CMD       1
BUILD    .CMD       1
SAVE     .CMD       2
DATE     .CMD       2
I        .CMD       1
BASIC    .CMD      66
EXEC     .CMD       2
JUMP     .CMD       1
NEWDISK  .CMD       7
MEMTEST  .CMD       2
MEMTU    .CMD       2
TEST     .CMD       1
RENUMBER.CMD       2
PRINT    .SYS       1
EDIT     .CMD      24
ASMB     .CMD      25
SAVE     .LOW       2
BACKUP   .CMD       3
XBG48    .CMD      86

SECTEURS LIBRES = 40
```


CE QUI APPARAIT SUR VOTRE ECRAN LORSQUE VOUS UTILISEZ LA COMMANDE CAT

APPUYER SUR LA BARRE D'ESPACEMENT POUR FAIRE POURSUIVRE LE DEFILEMENT

→ l'utilisateur de cette disquette possède une imprimante parallèle; il a donc créé PRINT.SYS à partir de PØKI.SYS

→ Il reste 40 secteurs libres sur la disquette

e) Problèmes de mise en page

Si le défilement du texte sur l'écran s'arrête, appuyer sur la barre d'espacement; sinon, faire un retour chariot↵ pour retourner au G2FLEX (+++).

Afin de fournir à l'utilisateur une grande souplesse au niveau de la présentation des textes sur l'écran et sur l'imprimante (longueur de ligne, nombre de lignes par page, gestion du curseur, contrôle de la pagination, etc.), G2FLEX offre les possibilités de la commande TTYSET à laquelle vous pouvez vous référer dès à présent (cf. page 41).

2. Connexion des imprimantes à GOUPIL sous G2FLEX

- Fixer le câble de liaison de l'imprimante sur les sorties notées S1 ou P1 à l'arrière de GOUPIL(série ou parallèle selon l'imprimante - à demander au revendeur).
- La méthode utilisée nécessite la fonction système PRINT.SYS .

Pour cela, il convient d'appeler PRINT.SYS que vous devrez créer sur votre disquette système selon la procédure suivante :

. Cas où l'utilisateur dispose d'une imprimante parallèle :

Vous créez alors PRINT.SYS à partir de PØKY.SYS en tapant :
+++ RENAME 0.PØKY.SYS 0.PRINT.SYS
Cela permet d'obtenir la sortie parallèle au moment de l'appel du FLEX pour l'édition.

. Cas où l'utilisateur dispose d'une imprimante série :

Vous créez PRINT.SYS à partir de PDIABLØ.SYS en tapant :
+++ RENAME 0.PDIABLØ.SYS 0.PRINT.SYS
Cela permet d'obtenir la sortie série.

Attention! La disquette système 5 pouces peut être protégée. Dans ce cas, pour utiliser RENAME, il faut enlever l'onglet. Ne pas oublier de le remettre ensuite pour plus de sécurité.
Pour éditer sur imprimante, on utilise alors la commande P (cf. page 33).
Exemple:
Edition du catalogue de la disquette système sur imprimante. Il suffit de taper :
P ⍽ CAT 0 (catalogue du lecteur 0)

xxx Rappel : pour les possesseurs d'imprimantes parallèles
- pour éditer ligne par ligne simultanément sur l'écran et sur l'imprimante, l'utilisateur dispose sous moniteur de la commande Ø (cf; Tome I,Annexe 5, V.2) Ceci avant tout chargement du FLEX - il ne s'agit pas de la commande Ø du FLEX.

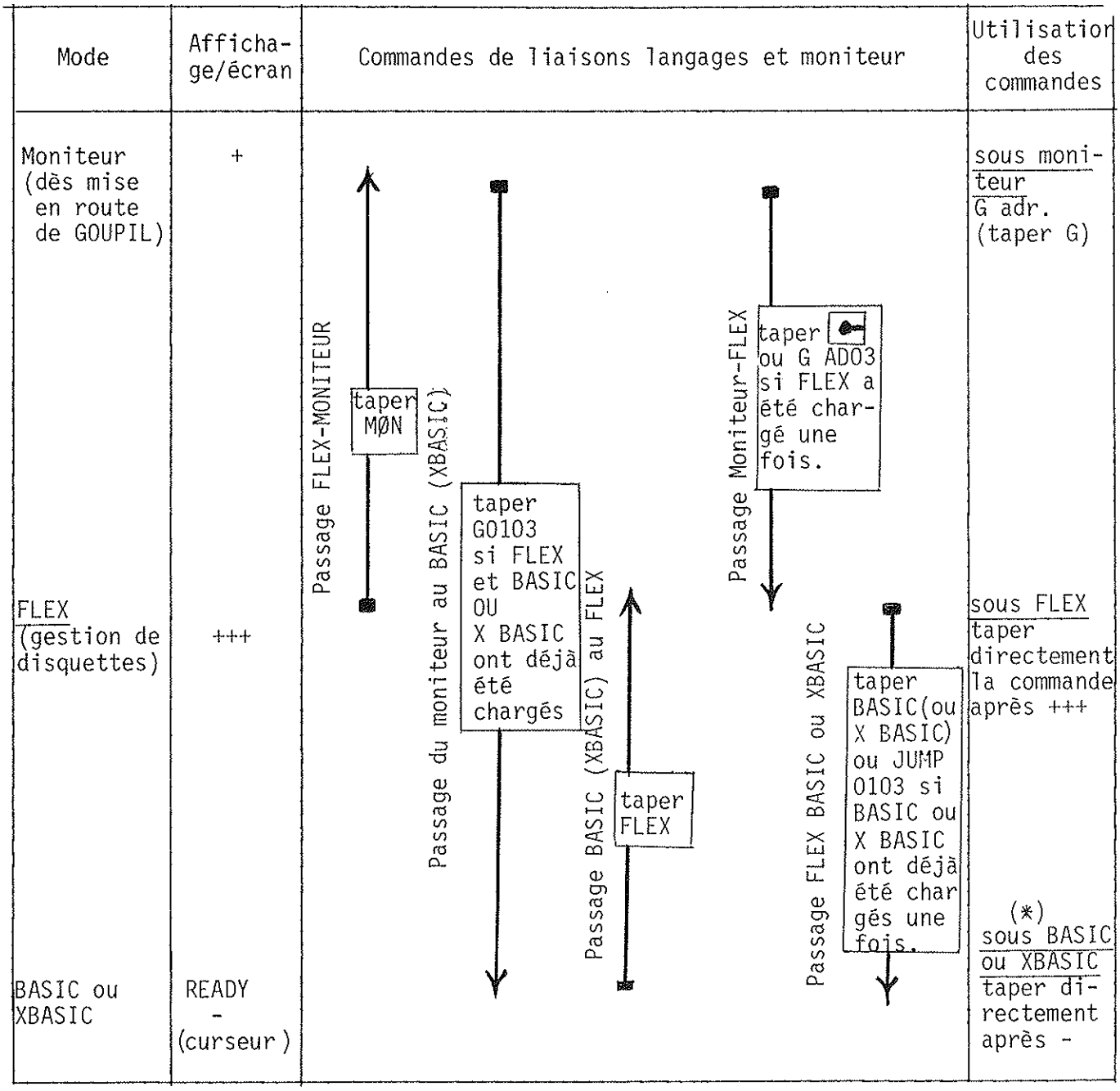
Dès que l'imprimante est sous tension, il suffit de faire:
+ Ø (lettre Ø)
+
Dès cet instant, toute ligne imprimée sur l'écran sera éditée sur l'imprimante.

3. Utilisation du BASIC ou XBASIC (BASIC ETENDU)

Pour utiliser le BASIC, tapez la commande BASIC(ou XBASIC pour le basic étendu), faites LØAD "nom de fichier" pour charger le programme si vous en avez déjà réalisé et stocké sur la disquette de travail (lecteur 1 à l'initialisation), puis faire RUN pour l'exécuter.
Sinon, et pour démonstration, vous pouvez charger le programme DEMØ en tapant :
LØAD "0.DEMØ" (DEMØ est en effet stocké sur le lecteur système numéroté 0 à l'initialisation) - DEMØ veut dire DEMONSTRATION -
Le "nom du fichier" doit être du type BAS (extension par défaut. L'extension .TXT peut être utilisée explicitement - la notion d'"extension" est définie dans le chapitre 1.4 et dans le glossaire).
Si le fichier est du type BAC, faites seulement RUN "nom du fichier".
Les fichiers de données utilisés par des programmes BASIC sont de type DAT.

Rappel: La touche BASIC ne doit pas être utilisée avec le système équipé de lecteurs. Elle ne sert qu'avec la version de base (cassette) pour lancer le BASIC.

SYNOPTIQUE DES COMMANDES ET DES LIAISONS ENTRE LE MONITEUR ET LES LANGAGES



* NOTE IMPORTANTE : Il est possible d'utiliser sous BASIC (ou XBASIC) les commandes FLEX. Il faut alors faire précéder ces commandes du signe +.
Exemple : Sous BASIC, la commande : +CATØ0, affichera les fichiers de la disquette 0.

ATTENTION ! Pour certaines commandes, cette procédure peut être dangereuse. Ces commandes peuvent en effet être chargées au même endroit que le BASIC et écraser celui-ci. Sous BASIC, on peut retenir comme règle de n'utiliser que les commandes ASN, CAT, DAT, DELETE, LIST, P, RENAME, TTYSET, VERSIØN, VERIFY, PRØT, Ø, I, à l'exclusion de toute autre.

Chapitre 1

1.1. Introduction

FLEX est un système d'exploitation de disque particulièrement souple et adaptable. Il fournit à l'utilisateur un ensemble puissant de commandes pour contrôler toutes les opérations sur disque directement à partir de la console écran/clavier. Les programmeurs seront étonnés de la grande variété des accès disque et des routines de gestion de fichiers, disponibles pour leur usage personnel.

FLEX est à ce jour l'un des plus puissants "Systèmes d'exploitation" existants sur le marché de la micro-informatique.

Le FLEX comprend trois parties :

- Le système de gestion de fichiers FMS.
- Le système d'exploitation des disques DOS.
- L'ensemble des Commandes utilitaires UCS.

Une partie de la puissance de FLEX réside dans le fait qu'il peut être étendu par simple addition de commandes utilitaires. L'utilisateur peut donc s'attendre, à l'avenir, à disposer de plus d'utilitaires.

Quelques unes des autres caractéristiques importantes du FLEX sont :

- la gestion dynamique de l'espace alloué aux fichiers.
- la suppression automatique des secteurs défectueux.
- la compression et la reconstitution automatique des fichiers "texte".
- un contrôle complet de l'environnement du système par l'utilisation de la commande utilitaire TTYSET.
- la portabilité uniforme du disque dûe à la gestion dynamique de l'espace disponible.

L'ensemble des commandes utilitaires (UCS) contient de nombreuses commandes très utiles. Ces programmes résident sur le disque "système" et sont chargés en mémoire seulement lorsque l'utilisateur les appelle. L'ensemble des commandes peut être facilement étendu à tout moment, sans changer le reste. Les utilitaires fournis avec FLEX gèrent des taches telles que sauvegarder, charger copier, changer de nom, effacer, fusionner et lister les fichiers sur disque.

Il existe aussi une commande importante, CATalogue, qui permet d'afficher le contenu du disque.
Plusieurs commandes de contrôle de l'environnement du système sont également fournies.

En résumé, FLEX fournit tous les outils nécessaires permettant à l'utilisateur d'agir sur le système disque. Une description plus approfondie de FLEX nous permettra objectivement de nous rendre compte des réelles possibilités de système d'exploitation.

1.2. Contraintes du système

Pour permettre au Flex de fonctionner, il est nécessaire d'avoir au moins 12 Koctets de mémoire RAM (0000 à 2FFF hex.) plus 8 Koctets localisés de C000 à DFFF hex.

Le FLEX permet d'utiliser de 1 à 4 lecteurs disque connectés sur le contrôleur. La configuration la plus courante est composée de 2 lecteurs organisés en lecteur 0 (disque système) et lecteur 1 (disque travail).

1.3. Démarrage du système

Pour charger le FLEX en mémoire il suffit d'appuyer sur la touche de fonction [•] du clavier, après avoir mis une disquette système dans le lecteur 0 et fermé la porte. Les moteurs du disque démarrent, dans le cas d'un lecteur 5" et environ 2 secondes après le message suivant apparaît sur l'écran :

```
>>> FLEX GOUPIL 2 <<<
Système d'exploitation disque
Version 3.0 du 14/8/81

DATE (JJ, MM, AA)?
```

Taper la date avec des virgules ou des blancs et faire un retour chariot (↵).

La réponse de FLEX est alors les trois signes "+++"
Ces signes seront toujours présents lorsque le système sera prêt à accepter une commande de l'opérateur. Le "+++" deviendra donc une vision familière signifiant que FLEX-GOUPIL est prêt à travailler pour vous !

1.4. Les fichiers disque et leurs noms

Les fichiers disque sont formés de "secteurs" disque et dans cette version, chaque secteur contient 256 octets d'information (252 utilisables). Chaque octet peut contenir un caractère de texte ou un octet d'information binaire. Le tableau suivant fournit le nombre de secteurs maximum composant une mini-disquette, suivant le type de disquettes.

NOMBRE DE SECTEURS MAXIMUM PAR DISQUETTE

TYPE DE LECTEUR DE DISQUETTE	SF SD *	DF SD *	SF DD *	DF DD *
5" simple face simple densité (35 PISTES)	390 secteurs ** 10 secteurs/piste			
5" double face double densité (40 PISTES)	390 secteurs 10 secteurs/piste	780 secteurs 10 secteurs/piste		1248 secteurs 16 secteurs/piste
5" double face double densité (80 PISTES)				2528 secteurs 16 secteurs/piste
8" double face double densité (77 PISTES)	1140 secteurs 15 secteurs/piste	2280 secteurs 15 secteurs/piste	1976 secteurs 26 secteurs/piste	3952 secteurs 26 secteurs/piste

* SF SD Simple face simple densité
* DF SD Double face simple densité
* SF DD Simple face double densité
* DF DD Double face double densité

** Pour les lecteurs BASF et DC (les lecteurs Shuggart par exemple ne fournissent que 340 secteurs en 5" simple face simple densité).

ATTENTION ! L'acte de mettre une disquette dans un lecteur est important. Pour certains lecteurs (CDC), il faut veiller, après avoir fermé la porte et chargé la disquette, à ce qu'un petit bruit se produise, manifestant que le système est bien en route.

Un fichier sera toujours d'une longueur au moins égale à un secteur et peut être aussi long que le nombre total de secteurs disponibles. L'utilisateur n'est pas concerné par l'emplacement du fichier sur le disque puisque cela est géré automatiquement par le système. La destruction d'un fichier est aussi géré par le système, et tous les secteurs précédemment utilisés deviennent immédiatement disponibles, pour ranger un autre fichier sur le disque.

Tous les fichiers sur disque ont un nom. On peut trouver par exemple :

PAIE
INVENTAIRE
TEST1234
AVRIL-80

A chaque fois qu'un fichier est créé, appelé ou détruit, son nom doit être utilisé. Un nom de fichier, peut avoir une longueur de 8 caractères maximum et doit commencer obligatoirement par une lettre.

On peut utiliser les caractères alphabétiques de A à Z ou de a à z, les caractères numériques de 0 à 9 ou une combinaison des deux pour créer un nom de fichier. Le tiret (-) ou le souligné (_) peuvent être aussi utilisés.

Les noms de fichiers contiennent également une "extension" L'extension définit plus précisément le fichier et indique le type d'information qu'il contient. Par exemple, on peut citer les extensions suivantes : TXT pour les fichiers texte (ou source), BIN pour les fichiers binaires, CMD pour les fichiers de commande et BAS pour les programmes BASIC. Les extensions peuvent contenir jusqu'à 3 caractères, le premier caractère étant une lettre. De nombreuses commandes FLEX prennent une extension par défaut et l'utilisateur n'a pas à les préciser lors de l'appel de la commande. L'utilisateur peut créer ses propres extensions et les traiter comme une partie du nom du fichier. Voici quelques exemples de noms de fichiers avec leurs extensions :

TEST.CMD
JEU1.BAS
FICHIER.BIN

L'extension doit toujours être séparée du nom du fichier par un point ".". Ce point est appelé "séparateur de champs". Il permet à FLEX de traiter les caractères suivant "le point" comme ayant une signification différente des caractères précédents.

Le nom et l'extension d'un fichier définissent celui-ci uniquement sur un disque mais peuvent exister sur plusieurs disques simultanément.

Pour sélectionner un lecteur en particulier,un "numéro de disque" est ajouté à la dénomination du fichier.

Ce numéro consiste en un simple chiffre (de 0 à 3) et est séparé du nom du fichier par le séparateur de champs ".".

L'ordre dans lequel doivent apparaître les différentes parties du nom d'un fichier n'est pas fixé. Les syntaxes suivantes sont toutes correctes :

```
0.BASIC
LUNDI.2
1.TEST.BIN
LIST.CMD.1
```

En résumé, une dénomination de fichier peut contenir jusqu'à trois champs séparés par des séparateurs de champs. Ces champs sont "le numéro du disque", "le nom", et "l'extension". Les règles de dénomination des fichiers peuvent être établies en utilisant les notations suivantes :

```
[<n° disque>.]  < nom > [.<extension>]
ou      <nom>[.<extension>][.<n° disque>]
```

Les "<>" délimitent un champ et n'apparaissent pas dans l'appellation du fichier. Les "[ ]" entourent les informations optionnelles de la dénomination.

Les exemples suivants sont tous syntaxiquement corrects :

```
0.NØM.EXT
NØM.EXT.0
NØM.EXT
0.NØM
NØM.0
NØM
```

Notons que le seul champ nécessaire est le "nom" lui-même et que les autres valeurs seront habituellement automatiquement prédéterminées. Etudier les exemples ci-dessus clarifiera la notation utilisée. La même notation reviendra régulièrement au long de ce manuel d'utilisation.

1.5. Lancement des commandes

Lorsque FLEX affiche "+++", le système est prêt à accepter une ligne de commande. Une ligne de commande est généralement constituée du nom de la commande choisie, suivi de certains paramètres variant suivant la commande à éxécuter. Il n'y a pas de commande "RUN" dans FLEX. Le programme lié à la commande à exécuter est toujours chargé en mémoire vive avant que l'exécution de la commande ne débute.

Si aucune extension n'est précisée dans le nom du fichier, "CMD" sera celle utilisée par défaut sinon, ce sera l'extension spécifiée qui sera retenue.

Ci-dessous, voici quelques exemples de commande tels qu'ils pourraient apparaître sur l'écran du GOUPIL

```
+++TTYSET
+++TTYSET.CMD
+++LØØKUP.BIN
```

Les deux premières commandes seront exécutées par FLEX de la même manière car, pour la première, une extension "CMD" sera choisie par défaut.

La troisième commande sera interprêtée comme :

- charger le fichier binaire LØØKUP en mémoire
- l'exécuter si l'adresse d'exécution (adresse de transfert) a été sauvegardée en même temps que le fichier sur le disque.

L'adresse de transfert est l'adresse qui indique au FLEX l'endroit où doit commencer l'exécution du programme. Si on essaie de charger et d'exécuter un programme sans adresse de transfert, le message "Adresse d'exécution ?" est affiché sur l'écran.

D'autres messages peuvent apparaître :

- "Comment ?" si la syntaxe d'une commande est incorrecte
- "Inconnu" si le fichier demandé n'existe pas sur le disque.

La fin d'une ligne de commande est marquée par l'utilisation de la touche (↵). Une erreur dans la commande peut être corrigée (si la touche (↵) n'a pas été frappée) en utilisant une touche de correction de caractère, initialisée dans la commande TTYSET. Par défaut, cette touche est CTRL H ou ←. Il est aussi possible d'annuler la ligne de commande en utilisant un second caractère spécial initialisé comme le précédent dans TTYSET. Par défaut, ce caractère est CTRL X. Lorsqu'une ligne de commande est annulée, le message "???" apparaît sur l'écran, il faut alors appuyer sur la touche (↵) pour voir réapparaître le message "+++".

Le premier nom d'une ligne de commande est toujours interprété comme une commande. Une liste optionnelle de noms de fichiers et de paramètres dépendant de la commande choisie, suit celle-ci.

Les différents champs d'une ligne de commande doivent être séparés soit par un espace soit par une virgule. Le format général d'une ligne de commande est le suivant :

<commande>[Ø <liste de noms et de paramètres>]

Une virgule est indiquée ci-dessus, mais un espace peut être utilisé. FLEX permet également de chainer plusieurs commandes sur une seule ligne en les séparant par ":" (celui-ci peut être modifié dans l'utilitaire TTYSET). Ce n'est que lorsque toutes les commandes de la ligne auront été exécutées que le message "+++" apparaîtra sur l'écran. Une erreur dans une des commandes arrêtera l'exécution de la ligne de commande et le système affichera "+++". Voici ci-dessous, quelques exemples de lignes de commandes correctement écrites :

```
+++CATØ1
+++CATØ1:ASNØS=1
+++LISTØBIBLIØ:CATØ1:CATØO
```

Le nombre total de caractères sur une ligne de commandes ne doit pas dépasser 128. Les caractères en excèdent seront ignorés par FLEX.

Un des derniers traits devant être abordé est la notion de disque "système" et disque "travail". Comme nous l'avons indiqué précédemment, si le numéro du lecteur n'est pas spécifié dans la désignation d'un fichier, une valeur par défaut sera automatiquement affectée. Cette valeur sera le numéro soit du disque "système", soit celui du disque "travail" courant.

Le disque "système" sera choisi par défaut pour toutes les commandes, en d'autres termes, pour tout fichier dont le nom apparaîtra en premier dans une ligne de commande.

Le disque "travail" sera choisi par défaut pour tout autre fichier nommé dans la commande.

Après chargement du FLEX, le disque "système" se trouve implicitement sur le lecteur 0 et le disque "travail" sur le lecteur 1. Cela permet de protéger la disquette système et de ne créer des fichiers que sur la disquette de travail préalablement formatée.

Le choix du disque "système" ou "travail" se fait par la commande ASN. (cf. sa description pour plus de détails).

Si le disque "système" est 0, que le disque "travail" est 1, la ligne de commande suivante :

+++LISTØFICHTEXT

sera interprétée par FLEX comme :

- aller chercher la commande LIST sur le disque 0
- l'exécuter sur le fichier FICHTEXT sauvegardé sur le disque 1.

1.6. Description des commandes

Il y a deux types de commandes dans FLEX ; les commandes résidentes en mémoire et les commandes utilitaires (qui résident sur disque et font partie de l'UCS). Il y a seulement deux commandes résidentes en mémoire : GET et MØN. Vous trouverez leur description ci-après. Les commandes utilitaires seront décrites dans les chapitres suivants.

GET

La commande GET est utilisée pour charger un fichier binaire en mémoire. C'est une commande spéciale qui n'est pas souvent utilisée. Sa syntaxe est la suivante :

GET [Ø < liste de noms de fichiers >]

où < liste de noms de fichiers > est :

< spécification du fichier 1 > [Ø < spécification du fichier 2 >] etc...

Comme d'habitude les "[ ] " entourent les items optionnels "spécification du fichier" indique un nom de fichier, décrit comme précédemment. L'action de la commande GET est de charger le fichier, ou les fichiers spécifiés dans la liste, en mémoire pour un usage ultérieur. Si aucune "extension" n'est indiquée dans le nom du fichier, BIN est affecté par défaut.

Ex. : +++GETØTEST
+++GETØ1.TESTØTEST2.0

En réponse à la première ligne de commande, GOUPIL chargera le fichier TEST.BIN du disque "travail" en mémoire, et à la deuxième ligne de commande, GOUPIL chargera en mémoire le fichier TEST.BIN à partir du disque "travail" et le fichier TEST2.BIN à partir du disque "système".

MØN

MØN est utilisée pour retourner sous contrôle du moniteur. La syntaxe de cette commande est simplement MØN suivi par (↵) (retour chariot).

Pour réentrer FLEX après avoir utilisé la commande MØN, vous devez taper la commande :

+G:CDOO↵ (ou G:CD03↵) (point froid, point chaud)

2.1. Ensemble des commandes utilitaires

Les pages suivantes décrivent toutes les commandes utilitaires livrées avec la disquette FLEX.

. Messages d'erreur communs

Plusieurs messages d'erreur sont communs à la plupart des commandes utilitaires. Ce sont :

Disque err ## 4	Ce message indique qu'un fichier référencé dans une commande particulière n'a pas été trouvé sur le disque spécifié. Habituellement, c'est un mauvais disque qui a été spécifié (par commande ou par défaut), ou bien une erreur de frappe s'est produite dans l'écriture.
Disque err ## 21	Cela peut arriver si le nom ou l'extension du fichier demandé ne commence pas par une lettre, ou qu'ils soient trop longs (8 et 3 caractères respectivement) ou tout simplement que le nom du fichier, sur lequel doit s'exécuter la commande, n'a pas été fourni par l'utilisateur.
Disque err ## 3	Ce message apparaitra si vous essayez de créer un fichier avec un nom qui existe déjà sur le même disque. Deux fichiers différents ne peuvent pas avoir le même nom sur la même disquette.
Disque err ## 26	Cela veut dire que la ligne de commande venant d'être tapée ne suit pas les règles de syntaxe établies. Se référer en conséquence aux règles de syntaxe fournies dans la description des commandes.

Note d'information générale

L'impression d'un texte sur le terminal de visualisation du GOUPIL (vidéo, imprimante...), sous contrôle d'une commande utilitaire, peut-être interrompue en tapant sur un caractère spécial (*). Une fois la commande stoppée, le défilement du texte peut reprendre si le caractère spécial est frappé une deuxième fois, ou bien la main est redonnée au FLEX si la touche (↵) est frappée. (Voir la description de la commande TTYSET pour la définition du caractère spécial).

(*) Caractère d'échappement
(espace par défaut)

APPEND

La commande APPEND est utilisée pour rattacher ou fusionner deux ou plusieurs fichiers. Le résultat de cette commande est un nouveau fichier. N'importe quels types de fichiers peuvent être fusionnés mais dans la plupart des cas il s'agit de rattacher ensemble des fichiers de même type. Si on concatène des fichiers binaires dont les adresses de transfert ont été précisées, l'adresse de transfert du fichier résultant sera l'adresse de transfert du dernier fichier de la chaine. Tous les fichiers originaux seront laissés intacts.

. Description

La syntaxe générale de la commande APPEND est :

APPENDØ <fichier spécifié>[Ø <liste de fichiers>]Ø<fichier spécifié>

où <liste de fichiers> peut être une liste optionnelle de spécifications.

Le dernier nom spécifié ne doit pas exister sur le disque puisqu'il sera le nom du fichier résultant. Si le dernier nom de fichier donné existe néanmoins sur le disque, la question : "le fichier existant doit-il être détruit ?" apparaitra sur l'écran du GOUPIL.

Si votre réponse est Ø (oui) le fichier existant sera effacé et l'opération APPEND sera poursuivie.

Si votre réponse est N (non) l'exécution de la commande est interrompue et la main est redonnée au FLEX (+++).

Tous les autres fichiers spécifiés doivent exister puisque ce sont eux qui devront être rattachés ensemble. Si seulement 2 noms de fichiers sont donnés, le premier fichier est transcrit sur le second. TXT est l'extension par défaut, à moins qu'une extension différente ne soit utilisée sur le premier fichier spécifié, auquel cas l'extension en question devient celle par défaut pour le reste de la ligne de commande.

Voici quelques exemples d'utilisation de la commande APPEND :

1) +++APPENDØ0.CHAP1ØCHAP2ØLIVRE

Sur le disque "travail", le fichier LIVRE.TXT sera crée à partir des fichiers CHAP1.TXT et CHAP2.TXT résidant respectivement sur le disque 0 et le disque "travail".

2) +++APPENDØACH1Ø1.FICH2.BAKØ0.BØNFICH

Cette syntaxe de la commande permet la concaténation du fichier FICH2.BAK, résidant sur le disque 2, au fichier FICH1.TXT, résidant sur le disque "travail". Le résultat sera le fichier BONFICH.TXT crée sur le disque 0.

ASN

La commande ASN est utilisée pour définir les unités disque "système" et "travail" ou pour sélectionner la recherche automatique d'un fichier. Le disque système est utilisé par FLEX pour les commandes ou, en général, pour le premier nom sur une ligne de commande. Le disque "travail" est utilisé par FLEX par défaut pour tous les autres fichiers d'une ligne de commande.

A l'initialisation, FLEX initialise le disque 0 comme disque "système" et le disque 1 comme disque "travail".

L'exemple qui suit montrera comment le système affecte ces valeurs :

APPEND/FICHIER1/FICHIER2/FICHIER3

Si le disque "système" est le disque 0 et le disque "travail" le disque 1, la ligne de commande ci-dessus sera interprétée par FLEX comme :

- Recherche de la commande APPEND sur le disque 0
- Concaténation des FICHIER1 et FICHIER2 résidents sur le disque 1
- Création du fichier résultant, FICHIER3, sur le disque 1.

Le disque "système" (disque 0) a été choisi par défaut pour la commande APPEND.

Le disque "travail" (disque 1) a été choisi par défaut pour l'exécution de la commande APPEND.

La recherche automatique d'un fichier sur les disques est réalisée par la lecture du disque 0, puis du disque 1 si le fichier n'est pas trouvé sur la disquette 0. Un message d'erreur est affiché si le fichier n'existe pas non plus sur la disquette 1.

Dans la version 5", si un lecteur n'est pas prêt (disquette manquante ou porte ouverte), FLEX attend jusqu'à ce que la disquette ait été insérée et la porte fermée.

Il est possible aussi de faire un RESET et de relancer le FLEX en tapant G:CD03

Dans la version 8", FLEX ne va pas explorer les lecteurs qui ne sont pas prêts, au cours d'une recherche automatique de fichier.

DESCRIPTION

La syntaxe générale de ASN est la suivante :

ASN [/W=<disque>][/S=<disque>]

où ⟨disque⟩ est le numéro du disque ou la lettre A.

Si ASN est tapé directement suivi par ↵ (touche retour chariot), le système affichera un message décrivant l'état du système. Par exemple :

```
+++ASN ↵
DISQUE SYSTEME=0
DISQUE TRAVAIL=0
+++
```

On trouve ci-dessous quelques exemples d'utilisation de la commande ASN :

```
ASNØW=1
ASNØS=1ØW=0
```

La première ligne initialise le disque 1 en disque "travail" et laisse le disque "système" tel qu'il était.

La seconde ligne initialise le disque 1 en disque "système" et le disque 0 en disque "travail".

Une utilisation intelligente des spécifications des numéros de disque peut éviter d'avoir à les préciser dans les dénominations des fichiers dans la plupart des cas.

Si vous souhaitez utiliser FLEX en recherche automatique des fichiers, il faut alors taper la lettre A (comme AUTOMATIQUE) à l'endroit prévu pour le numéro du drive.

Exemple :

```
ASNØW=A
ASNØS=AØW=1
ASNØS=AØW=A
```

BACKUP

La commande BACKUP permet des copies entières de disquettes FLEX. Ces copies sont différentes de celles produites par la commande CØPY. Le BACKUP fait une "image miroir" du disque à copier, alors que CØPY réorganise le disque de façon à ce que les secteurs d'un fichier soient tous regroupés. Généralement, CØPY sera utilisée quand le disque contient peu de fichiers et, aussi, pour recopier un seul fichier parmi tous ceux qui sont sur le disque. Le BACKUP dans la plupart des cas sera plus rapide que COPY, et le disque de sortie sera entièrement réécrit. L'expérience aidera à déterminer quelle commande utiliser.

DESCRIPTION

La syntaxe générale de la commande BACKUP est :

+++ BACKUP,⟨N° Lecteur du disque original⟩,⟨N° lecteur du disque à créer⟩

Ex : +++ BACKUP 0,1

Après exécution du BACKUP, la disquette du lecteur n° 1 sera identique à celle du lecteur 0.

Il est à noter que la disquette "copie" (N° 1) devra avoir été formatée préalablement et ne pas comporter de secteurs défectueux.

Si l'on veut faire une copie, par la commande BACKUP, sur une disquette non formatée, un message d'erreur disque sera affiché ainsi que dans le cas d'un secteur défectueux sur l'un des deux disques.

Remarque:

Si le disque original contient le FLEX préalablement chainé par la commande LINK, la copie sera automatiquement chainée.

BACKUP DISQUE DUR D140

La commande BACKUP permet de sauvegarder le contenu du disque fixe sur une cartouche amovible ("backup") ou inversement de recopier le contenu d'une cartouche amovible sur le disque fixe ("chargement").

Cette commande génère une copie "miroir", c'est-à-dire que la copie s'effectue secteur par secteur, et non fichier par fichier, sans aucun transfert en mémoire centrale.

DESCRIPTION :

La syntaxe générale de la commande BACKUP est :

+++BACKUP ⌇[<source>]⌇[<destination>]

où <source> et <destination> désignent les unités Ø (cartouche) ou 1 (disque fixe)

ex : +++BACKUP Ø 1 = chargement
+++BACKUP 1 Ø = backup
+++BACKUP = backup, par défaut

ATTENTION : une erreur peut être fatale ...

Le système affiche l'option retenue et demande confirmation :

taper Ø (oui) si tout est OK,
taper un autre caractère permet de changer d'option ou de retourner sous FLEX.

Le système demande une dernière confirmation, avant de vérifier si le lecteur de destination n'est pas protégé en écriture, et de commencer l'opération de copie.

L'affichage du n° de cylindre en cours de copie permet de suivre le bon fonctionnement de l'opération.

NOTA BENE :

- Cette opération demande plus de 10 minutes
- La cartouche n'a pas besoin d'être formattée au moyen de NEWDISK au préalable, et peut déjà contenir des fichiers qui seront alors effacés par la copie ... si elle n'est pas protégée en écriture.

BUILD

La commande BUILD est utilisée pour créer de petits fichiers TXT sans passer par l'Editeur de texte (par exemple STARTUP - voir STARTUP -).

Ces petits fichiers peuvent être exécutés par la commande EXEC (- voir EXEC -), dans le cas de fichiers de commandes.

DESCRIPTION

La syntaxe générale de la commande BUILD est :

BUILDØ<nom du fichier>

où <nom du fichier> est le nom du fichier que vous souhaitez créer. L'extension donnée par défaut est TXT et le disque affecté par défaut est le disque "travail".

Si le fichier indiqué existe déjà la question suivante apparait : "Le fichier existant doit-il être détruit ?".

Si vous répondez Ø (oui) le fichier déjà existant sera effacé. Si vous répondez N (non) l'exécution de la commande BUILD sera annulée.

Dès que vous êtes sous contrôle de la commande BUILD le signe égal ("=") apparait sur l'écran du GOUPIL. Pour entrer votre texte, tapez simplement sur le clavier les caractères désirés, en gardant bien à l'esprit que si ↵ (retour chariot) est tapé, la ligne est enregistrée dans le fichier et ne pourra plus être modifiée Néanmoins, avant que la touche (↵) ne soit tapée, la touche (back space) peut être utilisée ainsi que CTRLX (annulation de la ligne complète). Dans ce dernier cas le message ??? sera affiché et vous devrez taper retour chariot (↵) pour faire réapparaitre le signe "=" ; la ligne est alors annulée. Il faut bien noter que les seuls caractères alphanumériques (et non les caractères de contrôle) seront sauvegardés dans les fichiers TXT créés par la commande BUILD.

Pour sortir du mode BUILD, immédiatement après le signe "=", vous tapez sur la touche dièse (#) puis sur la touche (↵) retour chariot. La main est alors rendue à FLEX et les trois plus "+++" apparaissent sur l'écran.

CAT

La commande CAT (CATalogue) est utilisée pour afficher sur l'écran du GOUPIL, le nom et les caractéristiques des fichiers sauvegardés sur le disque. L'utilisateur peut demander l'affichage du contenu d'un ou plusieurs disques selon son choix.

Description

La syntaxe générale de la commande est la suivante :

CAT [Ø<n°disque>] [Ø<paramètres>]

Il peut y avoir un ou plusieurs n° de disques séparés par des virgules ou des espaces.

Les paramètres peuvent être un ensemble de caractères distinctifs, un nom, ou une extension. Ces paramètres constitueront le critère dont se servira la commande pour sélectionner le nom des fichiers à afficher.

Par exemple :

+++CAT

Le nom des fichiers contenus sur le disque "travail" sera affiché. Si le système est initialisé en mode recherche automatique, les noms des fichiers contenus sur toutes les disquettes seront affichés.

+++CATØ1ØA.TØDR

Seuls les noms des fichiers du disque 1 commençant par la lettre A et dont l'extension commence par la lettre T, et les noms de fichiers du disque 1 commençant par DR seront affichés.

+++ CAT∅PR

Sur le disque "travail" (ou sur tous les disques si le mode recherche automatique est sélectionné) seuls les noms des fichiers commençant par PR seront affichés.

+++CAT∅O∅1

Tous les noms des fichiers contenus sur les disques 0 et 1 seront affichés.

+++CAT∅O∅1∅.CMD.SYS

Sur les disques 0 et 1 seuls les fichiers dont l'extension est CMD ou SYS seront affichés.

Les informations sont présentées de la façon suivante sur l'écran :

JJ, MM, AA

CATALOGUE DE L'UNITE N° (numéro).

DISQUE (nom) ≠ (n° de volume)

NOM	TYPE	TAILLE	DATE	PRT
(nom du fichier)	(extension)	(nombre de secteurs)	(date de création)	(type de protection)

A la fin de l'affichage des noms de fichiers, FLEX donne à l'utilisateur le nombre de secteurs libres sur la disquette, sous la forme :

Secteurs libres = (nombre de secteurs)

+++

En résumé, si la commande CAT n'est pas paramètrée, tous les noms de fichier situés sur le disque "travail" seront affichés. Si un disque "travail" n'est pas assigné (mode recherche automatique disques), la commande CAT affichera le nom des fichiers contenus sur tous les disques en fonctionnement.

Si la commande CAT est paramètrée par un numéro de disque, tous les noms des fichiers contenus sur ce disque seront affichés.

Si la commande CAT est paramètrée par une extension, seuls les noms des fichiers comportant cette extension seront affichés.

Si un ensemble de caractères est utilisé, seuls les noms des fichiers débutant par ces caractères seront affichés.

Si la commande CAT est paramètrée par un nom et une extension, seuls les fichiers ayant ce nom et cette extension pour racine (sur le disque "travail") seront affichés.

Apprenez à bien utiliser la commande CAT et toutes ses possibilités, et votre travail avec le disque deviendra plus facile.

Les codes de protection pouvant apparaître sont les suivantes :

D Le fichier est protégé contre un effacement ou un changement de nom.

W Le fichier est protégé en écriture (effacement, dénomination ou écriture impossible)

Ø Pas de protection spéciale.

CØPY

La commande CØPY est utilisée pour faire des copies d'un fichier déjà résident sur le disque.

On peut copier des fichiers un à un, ou copier tout un groupe de fichier répondant à un même critère ou recopier des disquettes entières.

Une des qualités de la commande CØPY est aussi de pouvoir rassembler séquentiellment tous les secteurs d'un fichier, dispersés sur le disque lors de sa création ce qui a pour effet d'augmenter la rapidité d'accès au fichier.

DESCRIPTION

CØPY peut être utilisée sous trois formes :

a)+++CØPY <fichier 1> <fichier 2>

b)+++CØPY <fichier 1> <n° disque>

c)+++CØPY <n° disque> <n° disque> <paramètre>

Les règles de syntaxe utilisées sont les mêmes que celles décrites dans la commande CAT.

Si on copie un fichier d'une disquette sur une autre et que la disquette "destination" contient déjà un fichier de même nom le message suivant apparait sur l'écran :

Le fichier existe
doit-on détruire l'original ?

Si vous répondez Ø (oui) le fichier contenu sur le disque "destination" sera détruit.

Si vous tapez N (non) la commande CØPY sera interrompue et la main rendue au FLEX (+++).

Le premier type d'utilisation de la commande CØPY (a) permet de recopier un fichier sur le même disque (si le nom du fichier "destination" est différent de celui du fichier "source" ou sur un autre disque.

L'extension du fichier "source" doit toujours être spécifiée.

L'extension du fichier "destination" sera celle du fichier "source" si elle n'est pas spécifiée.

ex : +++ CØPYØ0.TEST.TXTØ1.TEST25

Le fichier TEST.TXT contenu sur le disque 1 sera recopié sous le nom TEST25.TXT sur le disque 1.

Le deuxième type d'utilisation de la commande CØPY (b) permet de recopier un fichier d'un disque sur un autre, le fichier "destination" prenant automatiquement le nom du fichier "source".

Ex : +++CØPY␣0.LIST.CMD␣1

Le fichier LIST.CMD sera recopié du disque 0 sur le disque 1.

L'extension du fichier "source" doit aussi être toujours précisée.

Le troisième type d'utilisation de la commande CØPY (c) permet de recopier le contenu de tous les fichiers d'une disquette sur une autre, les fichiers "destination" conservant le nom des fichiers "source".

Si une liste de paramètres est ajoutée à la commande seuls les fichiers répondant au cirtère fixé par ces paramètres, seront recopiés.

EX : +++CØPY␣0␣1

Tous les fichiers contenus sur le disque 0 seront recopiés sur le disque 1

+++CØPY␣1␣0␣.CMD␣.SYS

Tous les fichiers possédant les extensions CMD et SYS seront recopiés du disque 1 sur le disque 0

+++CØPY␣0␣1␣A␣B␣CA.T

Seuls les fichiers dont le nom commence par A ou B ou par CA et qui possède une extension commencant par T seront recopiés du disque 0 sur le disque 1.

Au cours de l'exécution de la commande CØPY utilisée sous cette forme, le nom du fichier et le numéro du disque "destination" sont affichés sur l'écran à la fin de la copie du fichier.

DATE

La commande DATE est utilisée pour initialiser, modifier ou afficher la date du jour.

Description :

La syntaxe de la commande est la suivante :

+++DATE Ø[<jour,mois,année>]

Chaque paramètre de la liste comporte 2 chiffres. Pour l'année, il faut préciser les deux derniers chiffres de l'année.

ex : +++DATEØ25,05,80 signifie 25 mai 1980.

En tapant DATE suivie du Retour chariot (↵) la date sera affichée sur l'écran.

ex : +++DATE ↵
25 mai 1980

DELETE

La commande DELETE est utilisée pour effacer un fichier du disque. Le nom de ce fichier sera supprimé du catalogue et les secteurs qu'il utilisait sur le disque deviennent immédiatement disponibles pour un autre fichier.

Description :

La syntaxe générale de la commande DELETE est :

+++DELETE¤<fichier>[<liste de fichiers>]

Un ou plusieurs fichiers peuvent être effacés à la suite. Il est nécessaire de préciser l'extension du ou des fichiers à effacer. Avant d'effacer un fichier, FLEX demandera :

EFFACEMENT DE " nom du fichier "?

Si vous répondez par ¤ (oui) le message :

ETES VOUS SUR ?

apparaîtra sur l'écran.

Si vous répondez par N (non) dans l'un ou l'autre cas, l'exécution de la commande sera interrompue et la main sera rendue au FLEX (+++).

Si par erreur, vous tapez sur une autre touche que ¤ ou N, l'exécution de la commande sera aussi interrompue.

Soyez certain du fichier à effacer, car sa récupération éventuelle nécessite des utilitaires spéciaux...

ex : +++DELETE¤MATHPACK.BIN

Le fichier MATHPACK.BIN sera effacé sur le disque "travail". Si le mode recherche automatique est sélectionné, il sera effacé sur le premier disque qui le contient.

+++DELETE¤1.TEST.TXT¤0.A¤UT.TXT

Le fichier TEST.TXT sera effacé sur le disque 1 et le fichier A¤UT.TXT sera effacé sur le disque 0.

Restrictions :

Un fichier protégé en écriture ou contre l'effacement (voir PR¤T) ne peut être effacé avant que la protection ne soit supprimée.

EXEC

La commande EXEC (exécuter) est utilisée pour exécuter un fichier "texte" constitué d'une liste de commandes.

Cette possibilité de construire des procédures complexes à partir des fichiers de commande confère au système FLEX une grande puissance.

Pour lancer l'exécution d'une telle procédure il suffit de taper au clavier ,EXEC, suivi du nom du fichier "TEXTE" choisi.

Ensuite EXEC exécutera chaque ligne de commande du fichier dans l'ordre d'apparition.

DESCRIPTION

La syntaxe de la commande est :

+++EXEC ␣ <nom du fichier>

où le <nom du fichier> est le nom d'un fichier de commandes. L'extension utilisée par défaut est TXT.

La création du fichier de commande se fait le plus souvent à l'aide de l'utilitaire BUILD.

Exemple d'utilisation :

La création d'une disquette "système" à partir d'une disquette vierge non formattée demande l'utilisation de plusieurs commandes dans un ordre fixé.

La première manipulation consiste à "formater" la disquette.

La seconde manipulation consiste à recopier tous les fichiers CMD, LØW et SYS sur la nouvelle disquette.

La troisième manipulation consiste à lancer la commande LINK afin que le FLEX puisse être chargé à partir du moniteur GPMØN (voir LINK).

On peut aisément concevoir la création d'un fichier MAKEDISK.TXT réalisant cette ensemble de commande.

Ex. :
```
+++BUILD MAKEDISK
   = NEWDISK 1
   = CØPY ØØ1 Ø.CMD Ø.LØW Ø.SYS
   = LINK 1.G2FLEX
   = #

+++
```

Chaque fois qu'une disquette système devra être créée, il suffira de taper :

```
+++ EXEC MAKEDISK
```

EXEC peut aussi être utilisé pour exécuter le fichier STARTUP (voir STARTUP).

I

La commande I permet à une commande utilitaire d'obtenir à partir d'un fichier des caractères qui doivent normalement être frappés par l'utilisateur au clavier.

Description :

La syntaxe de la commande est la suivante :

I⌀ <nom du fichier> ⌀ <commande>

où nom du fichier est le nom du fichier contenant les caractères devant être utilisés et <commande> la commande à exécuter suivant ce mode de fonctionnement. L'extension du fichier de caractères sera par défaut TXT.

Exemple :

Supposons qu'à la mise en route du système vous deviez toujours effacer un fichier appelé par exemple DATA.DAT. Il serait fastidieux d'avoir constamment à lancer la commande DELETE et surtout d'avoir à répondre aux deux questions posées :

EFFACEMENT DE "DATA.DAT"?

ETES-VØUS SUR ?

Aussi est-il plus subtil de construire la procédure suivante :

Premièrement, création d'un fichier de caractères qui répondra affirmativement aux deux questions posées par la commande DELETE.
Soit :

```
+++BUILD⌀ØUIØUI
 = ØØ
 = ##
+++
```

Deuxièmement, création d'un fichier STARTUP (voir STARTUP) qui aura pour fonction de lancer la commande DELETE d'effacement du fichier DATE.DAT
Soit :

```
+++BUILD⌀STARTUP
 = I⌀ØUIØUI⌀DELETE⌀DATA.DAT
 = ##
+++
```

A la mise en route du système, FLEX exécutera le fichier STARTUP donc lancera la commande DELETE et ira chercher dans le fichier ØUIØUI les réponses aux questions posées.

JUMP

La commande JUMP est utilisée pour lancer l'exécution d'un programme résident en mémoire vive, à partir du FLEX.

DESCRIPTION

La syntaxe de la commande est :

JUMP Ø < adresse hexadécimale >

où < adresse hexadécimale > est un nombre de 1 à 4 chiffres hexadécimaux représentant l'adresse d'exécution du programme en mémoire.

Dans la plupart des cas, la commande JUMP sera utilisée pour relancer l'exécution d'un programme assez long, sans passer par le chargement de ce programme en mémoire.

Exemple :
L'interpréteur BASIC est appelé à partir du disque en tapant la commande :

+++ BASIC ↵

FLEX va alors lire le programme sur le disque puis le charger en mémoire.

Si au cours d'une opération quelconque sous BASIC on retourne sous contrôle du FLEX, il suffit alors de taper la commande :

+++ JUMPØ0103 ↵ (0103 : adresse d'entrée du point chaud du BASIC)

pour retourner sous contrôle BASIC.

Attention :

Assurez-vous bien que le programme est effectivement résident en mémoire vive avant de lancer la commande JUMP.

LINK

La commande LINK est utilisée pour créer de nouvelles disquettes "système".

Elle permet le chargement du FLEX du disque en mémoire vive par l'intermédiaire de la commande [⊷] .

DESCRIPTION :

La syntaxe de la commande est :

+++ LINKØ < nom du fichier >

<nom du fichier> est habituellement G2FLEX

L'extension utilisée par défaut est SYS.

Exemples d'utilisation :

+++ LINKØG2FLEX

LINK du fichier G2FLEX.SYS résident sur le disque "travail".

+++ LINKØ1.G2FLEX

LINK du fichier G2FLEX.SYS résident sur le disque 1.

Pour plus d'informations sur l'utilisation de l'ordre LINK, se reporter au manuel de programmation avancée.

LIST

La commande LIST est utilisée pour afficher sur écran le contenu des fichiers "texte" ou "basic". Il est souvent plus agréable d'avoir la possibilité de consulter un fichier sans passer par un Editeur ou un autre programme de ce type.

Il est possible avec LIST de commander l'affichage de tout un fichier ou d'un certain nombre de lignes d'un fichier.

DESCRIPTION :

La syntaxe de la commande est :

+++ LIST␣⟨fichier⟩ ␣ ⟨n° ligne 1-n° ligne 2⟩ ␣ ⟨+ options⟩

⟨fichier⟩ désigne le nom du fichier à éditer. L'extension choisie par défaut est TXT.

⟨n° ligne 1-n° ligne 2⟩ délimitent la portion du fichier à éditer. Si ces deux paramètres ne sont pas précisés, l'ensemble du fichier sera affiché.

⟨+ options⟩ : la commande LIST offre deux options utilisées individuellement ou combinées

1) l'option + N permet la numérotation automatique des lignes à l'affichage

2) l'option + P permet la pagination automatique de l'édition et l'impression d'un titre en haut de chaque nouvelle page, s'il est précisé.

Un titre peut avoir jusqu'à 40 caractères de long.

L'en-tête de chaque page est constitué d'un titre (s'il existe) en haut à droite, de la date et du numéro de page en haut à gauche. Chaque page est constitué de 54 lignes. La fin d'une page est signifiée par l'envoi du caractère hexadécimal $0C.

Ces caractéristiques sont utiles lors de l'utilisation d'une imprimante à des fins de documentation (cf la commande P).

3) l'option +NP sélectionne la numérotation et la pagination automatique.

Exemples d'utilisation :

+++LISTØRECETTES

permet l'édition du fichier RECETTES.TXT contenu sur le disque "travail" sans numérotation de lignes, ni pagination automatique.

Toutes les lignes du fichier seront affichées.

+++LISTØCHAPITREØ30-200Ø+NP

permet d'édition de la 30ème à la 200ème ligne du fichier CHAPITRE.TXT en mode numérotation et pagination automatique.

+++ØLISTØLETTREØ100

Cette syntaxe est spéciale et permet l'affichage des lignes du fichier LETTRE.TXT à partir de la ligne n° 100 jusqu'à la dernière ligne.

Aucun n° de ligne ne sera édité (absence de l'option +N).

MEMTEST

La commande MEMTEST permet de tester la mémoire. Ce test détectera 99 % des problèmes mémoire si on l'exécute pendant suffisamment de temps.

DESCRIPTION

La syntaxe générale de la commande MEMTEST est :

MEMTEST, < addr. début héxa. > , < addr. fin.héxa. >

L'adresse de début (départ) est le nombre hexadécimal où le programme commencera le test et l'adresse de fin est le dernier emplacement mémoire à être testé.

Le test remplit la mémoire de nombres aléatoires, revient et vérifie si les nombres sont corrects, et répète le processus. Le test doit être exécuté approximativement une heure pour chaque bloc mémoire de 4 K.
Chaque passage du test affiche un "!" sur l'écran.

Le programme qui se charge en $C100 permet de tester la mémoire utilisateur de $0000 à $BFFF, mais ne doit pas être utilisé pour des adresses supérieures.

Exemple :

+++ MEMTEST, 0, 3FFF

Ceci aura pour effet de tester la mémoire de l'emplacement 0 à l'emplacement 3FFF.

La commande "RESET" doit être utilisée pour sortir du programme de TEST.

Dans le cas où des mémoires sont défaillantes, le message suivant apparaît :

addr	valeur chargée	valeur lue
↑	↑	↑
adresse de la mémoire défaillante		

MEMTU

La commande MEMTU est similaire à la commande MEMTEST, et elle sert à tester la zone mémoire occupée par FLEX (de 48 à 56 K).

DESCRIPTION

La syntaxe de la commande MEMTU est :

+++ MEMTU ↵

sans aucun paramètre.

Cette commande se charge et s'exécute en 0100

Elle détruit le FLEX qui devra être rechargé.

NEWDISK

NEWDISK sert à initialiser, on dit aussi formater, une nouvelle disquette. Les disquettes vierges ne peuvent être utilisées sans qu'un certain nombre d'informations soient préalablement inscrites dessus. NEWDISK permet d'écrire ces informations et d'éliminer d'éventuels secteurs défectueux (Phénomène extérieur causant des erreurs de données).

DESCRIPTION

La syntaxe générale de NEWDISK est la suivante :

NEWDISKØ <n° du lecteur>

où <n° du lecteur> est un chiffre spécifiant l'unité disque sur laquelle se trouve la disquette à formater. Après avoir tapé la commande, le système vous demande si vous êtes sûr de vouloir utiliser la commande NEWDISK et si le disque à initialiser est un disque qui n'a plus d'utilisation. Pour continuer, il faut alors taper "Ø" (OUI). Il existe quatre commandes NEWDISK que vous utilisez :

- les versions 5" simple densité
- les versions 5" double densité
- les versions 8" double densité
- les versions disque dur

Elles portent toutes les quatre le nom de "NEWDISK".

Les réponses aux différentes questions sur les simple ou double face, les simple ou double densité se font par Ø (oui), N (NØN). NEWDISK vous interroge alors pour un nom et un numéro de volume. (*) Ceci vous permet de référencer votre disquette pour de futures utilisations. La commande NEWDISK prend quelques minutes pour initialiser le disque, en vous assurant qu'il n'y a pas de défauts. Des secteurs défectueux ralentiraient le formatage proportionnellement au nombre de secteurs défectueux. A chaque mauvais secteur , le message suivant est affiché sur le terminal :

SECTEUR DEFECTUEUX A XXYY

où "XX" est le numéro de piste (en héxadécimal) et "YY" est le numéro de secteur (en héxa). NEWDISK retire automatiquement les secteurs inutilisables Si aucun secteur n'est défectueux, le nombre de secteurs formatés utiles par disquette est fourni dans le tableau suivant :

	SIMPLE FACE	DOUBLE FACE
5" simple densité	390	780
5" double densité	663	1326
8" simple densité	1 140	2 280
8" double densité	1 976	3 952
Disque dur		37 536

(*) En fait, n'importe quel nom et numéro.

Quelquefois, pendant un formatage, un secteur peut être défectueux dans une zone du disque utilisé par le système. Dans ce cas le message suivant est affiché :

ERREUR FATALE - FØRMATAGE ABANDØNNE

et le contrôle est redonné au FLEX. Cependant le disque n'est pas obligatoirement défectueux. Pour s'en assurer il suffit de recommencer l'opération en réinsérant le disque et en faisant NEWDISK. Si après plusieurs essais le formatage se termine par un "abandon" le disque est assurément inutilisable.

CREATION DE DISQUETTES SYSTEME

Un disque système contient le FLEX et normalement il contient aussi les utilitaires (Utility Command Set : UCS). La procédure suivante permet de préparer les disques système :

1) Initialisation de la disquette par l'utilisation de la commande NEWDISK décrite précédemment

2) CØPY de tous les fichiers. CMD sur la nouvelle disquette.

3) CØPY de tous les fichiers. SYS sur la nouvelle disquette. On peut noter que les deux étapes 2 et 3 peuvent être réalisées par une seule commande : "CØPYØOØ1Ø.CMDØ.LØWØ.SYS", vous permettant de copier de 0 à 1 tous les fichiers commandes souhaités. LØW copie l'utilitaire "SAVE.LØW".

4) Enfin il est nécessaire d'utiliser l'ordre LINK pour chaîner le fichier G2FLEX au chargeur secondaire et permettre le chargement du système.

Un moyen commode pour obtenir la procédure précédente sans avoir à taper toutes les commandes précédentes est de créer un fichier commande et d'utiliser EXEC. Consulter la documentation sur EXEC pour plus de détails.

Il n'est pas nécessaire d'exécuter cette procédure pour chaque disque. Il est aussi possible de créer des disquettes de travail, c'est-à-dire des disques sur lesquels il n'y a pas le FLEX mais des fichiers textes, "BASIC", ou de données. Pour créer un disque travail il suffit de faire un NEWDISK sur une disquette. Cependant ce disque ne permet pas de "charger" le système car il ne contient pas le "FLEX".

REMARQUE

Les versions GOUPIL 5 pouces contenant une carte contrôleur simple densité ne permettent pas la lecture d'une disquette double densité, alors que les versions munies d'une carte contrôleur double densité autorisent la lecture de disquettes simple et/ou double densité.
L'utilisateur qui dispose d'une carte contrôleur simple densité et qui désire relire les informations d'une disquette double densité sur son matériel peut le faire en procédant de la manière suivante :
- formatage d'une nouvelle disquette en simple densité. Sur les systèmes munis

d'une carte contrôleur double densité, les questions suivantes seront posées :

- disque à initialiser dans l'unité 0 ou 1 ?
- êtes-vous sûr ?
- disque 40 pistes ?(*)
- disque double face ?
- disque double densité ?
- nom du volume ?
- numéro du volume ?

- ensuite copie de la disquette double densité sur disquette simple densité.

Grâce à la copie effectuée, vous pourrez donc lire les informations d'une disquette double densité sur votre GOUPIL simple densité.

(*) Oui pour les lecteurs CONTROL DATA et BASF
Non pour les lecteurs SHUGGART
(D'une manière générale , les lecteurs sont des BASF et CONTROL DATA.)

Ø

La commande Ø est utilisée pour sauvegarder dans un fichier toutes les informations normalement affichées sur l'écran du GOUPIL au cours de l'exécution d'une commande.

La fonction de la commande Ø est similaire à celle de la commande P (voir P) excepté en ce qui concerne la sortie des résultats, qui sont imprimés pour P et stockés sur disque pour Ø.

Description :

La syntaxe de la commande est :

+++Ø ØK<nom du fichier> Ø<commande>

où <nom du fichier> est le nom du fichier dans lequel vont être stockés les résultats de l'exécution de la commande.
où <commande> peut être n'importe qu'elle commande utilitaire.

L'extension utilisée par défaut pour le fichier de sortie est ØUT.

Si Ø précède une série de commandes séparées par ":", son action n'a d'effet que sur la commande qui est juste derrière.

Exemples :

+++OØCATØCAT

permet la création d'un fichier CAT.ØUT contenant la liste des noms de fichier de l'inventaire.

+++OØBASØASMBØBASIC.TXT

permet la création d'un fichier BAS.ØUT contenant le listing source et assemblé du fichier BASIC.TXT

Cette procédure est à utiliser pour la mise en oeuvre du système d'impression différée ("Spooling"). Le fichier CAT.ØUT de l'exemple précédent peut être imprimé au moyen de la commande PRINT, simultanément à l'exécution d'une autre commande.

P

P est une commande système permettant l'impression des résultats de l'exécution d'une commande sur imprimante.

Cette commande est très utile lorsque l'on veut garder une trace écrite d'un fichier (LIST) ou de l'inventaire d'une disquette (CAT).

P initialise simplement une position mémoire qui testée pour le FLEX va permettre, au moment de l'impression des résultats, de faire appel à un fichier PRINT.SYS, sauvegardé sur disque, contenant les sous-programmes nécessaires à la sortie des informations sur l'imprimante utilisée.

Description :

La syntaxe de la commande est :

+++PØ ⟨commande⟩

où ⟨commande⟩ peut être n'importe qu'elle commande standard.

Si P est utilisé devant une liste de commandes séparées par ":", son action n'a d'effet que sur la commande qu'il précède immédiatement.

Exemples :

+++PØCAT

permet l'édition de l'inventaire du disque sur l'imprimante.

+++PØLISTØMØNDAY:CATØ1

permet l'édition du contenu du fichier MØNDAY.TXT sur l'imprimante puis enchaine l'édition de l'inventaire du disque 1 sur l'écran vidéo du GOUPIL.

Pour plus d'informations sur la commande P et le fichier PRINT.SYS, consultez le "guide de programmation avancée".

La commande P cherchera le fichier PRINT.SYS sur le même disque que celui où elle est stockée.

Sur la disquette système G2FLEX fournie par la SMT, plusieurs sous programmes de gestion d'imprimante sont données :

PØKI. SYS	pour une imprimante à interface parallèle de type "Centronics" ou OKIDATA
PDIABLØ.SYS	pour une imprimante à interface série RS232 (vitesse 1.200 bauds).

Selon le type d'imprimante connectée sur GOUPIL il faut alors nommer PRINT.SYS l'un de ces deux fichiers par la commande RENAME.

Si vous souhaitez utiliser une configuration différente, consultez le "Guide de programmation avancée" pour obtenir tous les détails nécessaires à l'écriture de votre propre sous-programme d'interface.

PRINT

Le système FLEX peut sortir sur imprimante des données stockées sur fichiers en même temps qu'il exécute d'autres tâches. Cette possibilité s'avère particulièrement utile lorsqu'il faut imprimer un long listing sans mobiliser les ressources de l'ordinateur.Cette méthode d'impression est appelée "spooling" imprimante.

DESCRIPTION

La syntaxe générale de la commande PRINT est :

PRINT, <nom de fichier>[,+<répéter #>]

<nom de fichier> est le nom du fichier à imprimer. L'extension par défaut est .ØUT .

<répéter> est le nombre de copies supplémentaires que vous voulez faire imprimer.

Admettons par exemple que votre disquette comporte une très grande quantité de fichiers, et que vous désiriez une liste imprimée du catalogue. Il vous faudra d'abord créer un fichier contenant les informations de sortie à l'aide de la commande Ø de la manière suivante :

+++ Ø,CAT.ØUT,CAT.CMD

ou

+++ Ø,CAT,CAT (se reporter à la description de la commande Ø)

Si vous désirez la sortie sur imprimante, il faut taper la commande :

+++PRINT,CAT.ØUT ou +++PRINT,CAT

A ce moment, le fichier CAT.ØUT est rangé dans un tampon appelé "queue d'impression" (file d'attente). Si une autre commande PRINT est émise avant la fin d'exécution de la première, le second fichier se trouvera à l'emplacement disponible suivant dans la queue d'impression. Lorsque le nom du fichier à imprimer a été stocké dans la queue d'impression, le contrôle retourne au système d'exploitation FLEX. Dès cet instant, vous pouvez effectuer n'importe quelle opération sur disque, par exemple supprimer des fichiers, recopier des disquettes, etc. Pendant que vous utilisez le FLEX, la commande PRINT fera sortir le fichier désigné sur l'imprimante. PRINT attendra automatiquement que l'imprimante soit prête (power up), même si le fichier a été rentré dans la queue d'impression.

Après l'impression du premier fichier, le second fichier sur la file

SMT

d'attente sera imprimé (s'il en existe un à imprimer), et ainsi de suite. La queue d'impression peut être examinée ou modifiée à tout moment par l'utilisation de l'utilitaire QCHECK.

Remarque :

L'utilisateur doit prendre certaines précautions lorsqu'il utilise le "spooling" imprimante:

1) Tout fichier figurant dans la queue d'impression ne doit pas être supprimé, renommé ou changé de quelque manière que ce soit avant d'avoir été imprimé ou extrait par l'utilitaire QCHECK qui gère la queue d'impression.
2) Les disques contenant les fichiers dans la queue d'impression ne doivent pas être enlevés tant que les fichiers figurent encore dans la queue.
3) La commande P ne doit pas être utilisée tant que les fichiers sont en attente dans la queue d'impression.
4) Toute opération de chargement à partir d'une bande magnétique ou de ruban papier, ou toute autre opération exigeant que l'ordinateur accepte des données à intervalles de temps précis, ne doit pas être exécutée pendant le processus de "spooling" imprimante.

PRØT

La commande PRØT est utilisée pour modifier le type de protection d'un fichier.

Lorsqu'un fichier est crée et sauvegardé sur disque, il n'est pas protégé et l'utilisateur peut le détruire, le modifier, changer son nom, etc...

En utilisant la commande PRØT, tout fichier peut être protégé en écriture ou en effacement.

DESCRIPTION :

La syntaxe générale de la commande est :

+++PRØTb<fichier> b [<liste d'options>]

où <fichier> désigne le fichier à protéger et <liste d'options> une combinaison des options suivantes :

- D : protection contre l'effacement d'un fichier. . Les ordres RENAME, DELETE n'ont pas d'action sur un fichier protégé contre l'effacement de même que les fonctions de suppression ("le fichier existant doit-il être détruit ?") des commandes SAVE, CØPY, ETC...

- W : protection en écriture d'un fichier. Un fichier protégé en écriture est automatiquement protégé en effacement. Il ne peut être effacé, renommé ou modifié.

- C : protection de catalogue. Le nom du fichier ainsi protégé ne pas pas affiché au cours de l'exécution de la commande CAT.
- X : l'option X permet de supprimer tout type de protection affecté à un fichier.

Exemples :

+++PRØTbCAT.CMD,XW

supprime toutes les protections du fichier CAT.CMD puis le protège en écriture

+++PRØTbCAT.CMD,X

supprime toutes les protections du fichier CAT.CMD

+++PRØTbINFØ.SYS,C

interdit l'édition du nom et des caractéristiques du fichier INFØ.SYS au cours de l'exécution de la commande CAT.

QCHECK

L'utilitaire QCHECK sert à examiner et à modifier le contenu de la queue d'impression . QCHECK ne s'accompagne d'aucun argument supplémentaire. Il suffit de taper QCHECK;
QCHECK arrêtera l'impression en cours , puis affichera le contenu de la queue d'impression de la manière suivante :

```
+++ QCHECK
      POS       NOM     TYPE      RPT
        1       TEST    .OUT       2
        2       CHAP.   .OUT       0
        3       CHAP2.  .TXT       0
COMMANDE ?
```

Cette sortie de résultats indique que TEST.OUT est le prochain fichier à imprimer(ou qu'il est en cours d'impression) et que 3 copies (1 plus répétition de 2 autres) de ce fichier seront imprimées. Ensuite viendra le tour de CHAP.OUT qui sera imprimé, puis CHAP2.TXT . Le message COMMANDE ? signifie que QCHECK attend l'une des commandes suivantes :

COMMANDE	FONCTION
(retour chariot)	re-lancement de l'impression, retour au système FLEX
Q	une commande Q imprimera à nouveau le contenu de la queue d'impression.
R,#N,X	la commande R répète le fichier à la position #N X fois. Si la valeur de X est omise, le nombre de répétitions ne sera sorti qu'une seule fois. Exemple : R,#3,5
D,#N	la commande D retire le fichier à la position # N. Si N=1, le travail d'impression en cours sera interrompu.
T	la commande T interrompt le travail d'impression en cours et lance le travail d'impression suivant ,sauf si le compteur de répétitions du fichier courant n'est pas nul, auquel cas le système continuera l'impression du fichier courant jusqu'à ce que le compteur soit nul. Pour interrompre totalement le travail en cours, utilisez la commande D,#1.

QCHECK (suite)

N,#N	par cette commande N, le fichier à la position #N sera le prochain fichier à imprimer une fois le travail d'impression en cours achevé. Si vous tapez Q après cette opération, vous obtiendrez le nouvel ordre de la queue d'impression. Exemple : N,#3
S	la commande S permet d'arrêter l'impression. Dès que le travail d'impression en cours est achevé, l'impression s'arrête jusqu'à ce que vous tapiez la commande G.
G	la commande G relance l'impression lorsque la commande S a été utilisée pour l'arrêter.
K	la commande K supprime le processus d'impression en cours. Tous les travaux d'impression mis en file d'attente seront supprimés de la queue d'impression. Les fichiers ne sont pas effacés du disque.

RENAME

La commande RENAME est utilisée pour donner à un fichier déjà existant un nouveau nom dans le catalogue.

Le nom, aussi bien que l'extension d'un fichier peut être modifié.

DESCRIPTION :

La syntaxe de la commande est :

+++RENAMEØ <fichier 1> Ø <fichier 2>

où <fichier 1> est le nom du fichier à modifier et <fichier 2> le nouveau nom à donner.

L'extension du fichier 1 par défaut est TXT.

Le disque choisi par défaut pour rechercher le fichier 1 est le disque "travail".

Si l'extension du fichier 2 n'est pas précisée, celle du fichier 1 est choisie par défaut.

Dans la dénomination du nom du fichier 2, il n'est pas nécessaire de préciser le n° du disque. S'il existe quand même, il ne sera pas pris en compte.

Exemples :

+++RENAMEØTEST1.BINØTEST2

permet de renommer le fichier TEST1.BIN en TEST2.BIN

++RENAMEØL.LETTERØREPLY

permet de renommer sur le disque 1, le fichier LETTER.TXT en REPLY.TXT.

+++RENAMEØ0.FIND.BINØFIND.CMD

permet de renommer le fichier FIND.BIN en FIND.CMD sur le disque 0. Ceci est utilisé pour transformer les fichiers binaires crées en assembleur, en fichiers commandes directement appelables par FLEX.

Si le nouveau nom choisi existe déjà dans le catalogue du disque le message suivant apparaitra :

LE FICHIER SPECIFIE EXISTE DEJA (erreur 3)

Il faut bien retenir que la commande RENAME ne modifie que le nom du fichier et n'a aucun effet sur son contenu.

Les commandes utilitaires étant des fichiers semblables aux autres il est possible de leur affecter d'autres noms, en utilisant simplement la commande RENAME.

SAVE

La commande SAVE est utilisée pour sauvegarder sur disque une zone mémoire. Une des premières applications de cette commande est de sauvegarder sur disque des programmes chargés en mémoire à la main ou à partir de la cassette.

Description :

La syntaxe de la commande est :

+++SAVE␣<nom du fichier>␣<ad.début>␣<ad.fin> [␣<ad.transfert>]

où <nom du fichier> est le nom donné au fichier ainsi créé. L'extension BIN et le disque "travail" sont choisis par défaut.
<ad.début> et <ad.fin> sont les adresses hexadécimales qui définissent la longueur de la zone à sauvegarder.

Le dernier paramètre est optionnel. Il s'agit de l'adresse d'exécution du programme. Cette adresse est précisée lorsque le programme sauvegardé doit être chargé et exécuté par FLEX.

Exemples :

+++SAVE␣DATA␣100␣1FF

permet de sauvegarder la zone mémoire comprise entre 0100 et 01FF (hex) sur disque, dans un fichier appelé DATA.BIN. Ce fichier sera stocké sur le disque "travail". L'adresse d'exécution n'a pas été précisée.

+++SAVE␣1.GAME␣0␣1680␣100

Le contenu de la zone mémoire comprise entre 0 et 1680 (hex) sera sauvegardé sur le disque 1, en un fichier GAME.BIN. 0100 (hex) est l'adresse d'exécution du programme en mémoire.
Ainsi, si à la suite des trois plus (+++), vous tapez GAME.BIN au clavier, FLEX chargera le programme en mémoire et lancera son exécution en 0100.

Si le nom choisi pour ce fichier existe déjà, le message :

Le fichier existant doit-il être détruit ?

apparaîtra sur l'écran.

Si vous répondez Ø (oui) à cette question, l'ancien fichier sera détruit ; si vous répondez N (non) l'exécution de la commande sera abandonnée.

Dans le cas où plusieurs zones mémoires non contigües doivent être sauvegardées, il est nécessaire de créer autant de fichiers que de segments de mémoire à sauver. Ensuite, il suffira d'utiliser la commande APPEND pour les réunir en un seul fichier. Si le dernier fichier à concaténer possède une adresse d'exécution, celle-ci sera automatiquement transmise au fichier résultant.

SAVE.LØW

Il existe une autre forme de la commande SAVE dans l'ensemble des commandes utilitaires. Sa syntaxe est identique à la précédente (ne pas oublier de préciser l'extension). Cette commande est utilisée pour sauvegarder des programmes dans la zone mémoire des commandes utilitaires et permet ainsi de créer ses propres commandes.

Consultez le "Guide de programmation avancée" pour plus de détails.

STARTUP

STARTUP n'est pas une commande utilitaire mais un fichier d'utilisation tout à fait spécial.

Il est souvent très utile d'avoir la possibilité d'initialiser le système directement au chargement par la touche [RESET] par exemple de charger le BASIC et exécuter un programme d'application.

DESCRIPTION

FLEX explore le catalogue du disque "système" à l'initialisation, à la recherche d'un fichier STARTUP.TXT. Si celui-ci n'est pas trouvé, les trois signes "plus" sont affichés (+++) et le système attend une commande de l'utilisateur. Sinon, le fichier trouvé est lu et interprété comme une commande a exécuter immédiatement. En fin de fichier, la main est rendue au FLEX (+++) en attente d'une commande de l'utilisateur.

Exemple :

Supposons que nous souhaitions passer sous contrôle BASIC chaque fois que le système est lancé. Il est nécessaire de créer un fichier STARTUP du type :

```
+++BUILDØSTARTUP
   = BASIC
   = #
+++
```

Le fichier STARTUP ne doit être composé que d'une seule ligne (qui peut comporter plusieurs commandes).

Cette ligne commande à FLEX de charger et d'exécuter le BASIC à chaque mise en marche du système

Il est nécessaire que le disque "système" sur lequel s'applique cette procédure, contienne le G2FLEX,correctement chainé par LINK, et le BASIC.

Un autre exemple d'utilisation du fichier STARTUP est l'initialisation des paramètres controlant l'environnement du système disque. Si le fichier STARTUP est constitué de la ligne de commande suivante,

```
TTYSETØDP=16ØWD=6ØØ: ASN W=1Ø: ASNØ: CAT 0
```

à la mise en route du système les opérations suivantes seront exécutées :

1 - TTYSET initialisera l'écran à 16 lignes de 60 caractères

2 - ASN initialisera le disque "travail" sur le lecteur 1

3 - ASN affichera les n° des disques "système" et "travail" sur l'écran

4 - Le catalogue du disque 0 sera affiché sur l'écran.

Pour avoir plus de détails sur ces différentes commandes, consulter leur description contenue dans ce manuel.

A ce stade, nous avons vu que la longueur du fichier STARTUP est limité à une ligne de commande. Il existe un moyen de contourner cette restriction en utilisant la commande EXEC.

Il suffit de créer un fichier .TXT contenant la liste des commandes à exécuter puis de créer un fichier STARTUP contenant la seule ligne d'instruction :

EXECØ < nom du fichier >

Le fichier STARTUP peut être modifié ou effacé.

Il peut arriver, au cours de l'exécution du STARTUP, de perdre le contrôle du focntionnement du système. Dans ce cas, appuyez sur le bouton RESET et lancez l'exécution du DØS en CD03 (hex.). L'appel au point chaud du FLEX, ne provoque pas la recherche et l'exécution du fichier STARTUP.

TEST

La commande TEST est utilisée pour tester tous les secteurs d'une disquette. Chaque secteur défectueux sera affiché sur l'écran.

DESCRIPTION :

La syntaxe générale de la commande TEST est :

TEST [, <N° du lecteur>]

où le N° du lecteur spécifie quelle disquette doit être testée, par défaut ce sera la disquette de travail.

Tous les secteurs considérés comme mauvais pendant le test sont affichés sur le terminal, sous la forme de deux nombres hexadécimaux, le premier représentant le numéro de la piste, le second le numéro du secteur.

Exemple :

+++ TEST,0

Cette commande testera la disquette du lecteur 0 et affichera les mauvais secteurs. Il est à noter que la commande TEST demande un temps d'exécution assez long puisque tous les secteurs de la disquette sont lus.

TTYSET

La commande TTYSET permet à l'utilisateur de définir les caractéristiques de l'écran ou de l'imprimante utilisés et les caractères de contrôle. Avec cette commande, le format et le contrôle de l'édition peuvent être modifés par l'utilisateur.

Description :

La syntaxe générale de la commande TTYSET est :

+++TTYSET Ø[<Liste de paramètres>]

où liste de paramètres peut être un ou plusieurs des paramètres définis ci-après. Un paramètre est composé de 2 lettres suivies du signe "=" puis par la valeur à lui affecter. Chaque paramètre peut être séparé par une virgule ou un espace.

Si aucun paramètre n'est inscrit, l'exécution de la commande affichera la valeur de tous les paramètres déjà initialisés.

Exemples :

+++TTYSET
permet l'affichage de tous les paramètres.

+++TTYSETØDP=16ØWD=63
initialise la taille de l'écran à 16 lignes de 63 caractères.

+++TTYSETØBS=8ØES=3
initialise le caractère "Backspace" comme étant celui qui répond au code 08 et le caractère "Escape" celui qui répond au code 03.

Description des paramètres :

Dans la description qui suit, "hh" est utilisé pour définir une valeur hexadécimale par défaut, "dd" une valeur décimale choisie par défaut.

Les valeurs des paramètres exprimées en hexadécimal sont affichées sur l'écran précédées du signe $.

BS = hh caractère "Backspace" (hexadécimal)
initialise le caractère "Backspace" à la valeur Ascii : hh
Ce caractère est normalement CRTL H (08 hex.) ou (←)
L'action du caractère "Backspace" est d'annuler le dernier caractère entré au clavier (les derniers en cas de plusieurs BS)

BE = hh caractère écho du "Backspace" (hexadécimal)

Ce caractère est par défaut 08 mais peut prendre n'importe qu'elle valeur ASCII. Son action est de définir le caractère à afficher sur l'écran en écho d'un caractère "Bakspace". (←—)

Lorsque BE est initialisé à Ø8 (hexa), FLEX, en écho du BACKSPACE, enverra un blanc (20) suivi d'un BACKSPACE ce qui efface le caractère. (Le BACKSPACE est la touche ←)

DL = hh caractère d'effacement de ligne (hexadécimal)

Ce caractère est normalement CRTL X (annulation de la ligne). On peut l'initialiser à n'importe quelle valeur ASCII.

EL = hh caractère séparateur (hexadécimal)

Ce caractère est utilisé par FLEX pour séparer plusieurs commandes sur une même ligne. Il est normalement égal à 3A (hex), code AScii de ":". La mise à 0 de EL, interdit l'écriture de plusieurs commandes sur une même ligne.

La valeur de EL doit être celle d'un caractère affichable (les caractères de contrôle ne sont pas autorisés).

DP = dd hauteur de page (décimal)

Ce paramètre initialise le nombre de lignes par page (écran ou imprimante)

Si DP = 0 il n'y a pas de pagination. La valeur initiale de ce paramètre est 24.

Pour plus de renseignements sur la pagination voir EJ et PS.

WD = dd longeur de ligne (décimal)

Ce paramètre initialise le nombre de caractères à afficher par ligne sur l'écran de votre terminal, ou par une imprimante.

Si WD = 0 n'importe quel nombre de caractères est permis sur une ligne. La valeur initiale de ce paramètre est 80.

Si la ligne à afficher comporte un nombre de caractères supérieur à celui initialisé par WD, une nouvelle ligne sera affichée à la suite.

NL = dd nombre de caractères nuls (décimal)

Ce paramètre initialise le nombre de caractères nuls à envoyer en fin de chaque ligne.

Ceci permet de résoudre le problème d'un retour chariot trop lent et d'attendre quelque temps avant d'éditer le caractère suivant sur un terminal imprimant.

La valeur initiale est 0.

TB = hh tabulation (hexadécimal)

Ce caractère n'est pas utilisé par FLEX mais certains logiciels tel l'Editeur de Texte peuvent l'utiliser. Valeur 0 par défaut.

EJ = dd hauteur de saut (décimal)

Ce paramètre initialise le nombre de lignes à sauter en fin de chaque page. Si le paramètre PS est "ØUI", la séquence de saut est envoyé après la fin de la pause.

PS = Ø ou PS = N contrôle de Pause

Ce paramètre rend possible ou non une pause en fin de page.

Si PS = Ø et DP ≠ 0, l'édition est automatiquement suspendue en fin de page. La reprise de l'édition s'effectue en tapant sur la touche définie comme caractère d'échappement.

Si PS = N la pause n'est pas exécutée.

ES = hh caractère d'échappement (hexadécimal)

Ce paramètre force le caractère d'échappement à la valeur AScii "hh" (hexa).

La valeur initiale est $ 20 (barre d'espacement). Le caractère d'échappement est utilisé pour arrêter les éditions en cours et les faire redémarrer. Il est aussi utilisé pour relancer une sortie, arrêtée par une pause.

La valeur initiale correspond à la barre d'espacement.

VERIFY

La commande VERIFY permet de placer le système de gestion de fichier en mode vérification écriture. Si VERIFY est "on", chaque secteur écrit sur le disque est relu pour vérification. Si VERIFY est "off" aucune vérification n'est effectuée.

DESCRIPTION :

La syntaxe générale de VERIFY est

VERIFY [ØN]

ou

VERIFY [ØFF]

Si VERIFY est tapé sans paramètre, l'état de VERIFY est affiché sur le terminal

Exemple :

```
+++VERIFYØN
+++VERIFY
```

Le premier exemple établit le système en mode "vérification d'écriture".

Le deuxième exemple affiche l'état de VERIFY (ØN ou ØFF)

VERSIØN

La commande VERSIØN est utilisée pour afficher le numéro de version d'une commande utilitaire.

DESCRIPTION :

Le syntaxe générale de la commande VERSIØN est

VERSIØNØ < nom du fichier >

où < nom du fichier > est le nom de l'utilitaire que vous souhaitez consulter. Par défaut l'extension choisie est CMD et le disque, le disque "travail"

Exemple :

+++VERSIØNØO.CAT

permet d'afficher le numéro de la version de la commande CAT, stockée sur le disque 0, sur l'écran du GOUPIL.

Pour que le numéro de la version soit valide, l'utilitaire doit être écrit de manière que le troisième octet du programme soit le numéro de la version.

Il faut donc utiliser la syntaxe suivante au début du programme.

```
START    BRA      DEBUT
   FCB   05  <------------  n° de la version
DEBUT       ~
            |
            v
          suite du programme
```

XØUT

XØUT est une forme spéciale de la commande DELETE qui permet la destruction de tous les fichiers ayant l'extension ØUT.

DESCRIPTION :

La syntaxe générale de la commande est :

XØUT [Ø < Numéro du disque >]

ou le < numéro du disque > est le disque choisi. Si aucun n° de disque n'est précisé, tous les fichiers avec extension .ØUT seront supprimés sur le disque "travail" et si la recherche automatique est initialisée tous les fichiers avec extension .ØUT seront supprimés sur tous les disques .XØUT ne supprime pas les fichiers qui sont protégés contre l'effacement ou qui se trouve dans la queue d'impression.

Exemple :

+++XØUT

+++XØUTØ1

3. INFORMATIONS GENERALES

3.1 - CAPACITE DISQUE

Chaque secteur d'un disque FLEX contient 252 caractères ou octets de données utilisables. (chaque secteur est formatté à 256 octets mais les 4 premiers sont utilisés par le système).

Ainsi, une mini-disquette simple face possède 390 secteurs ou 98280 caractères disponibles. La mini-disquette double face en contient exactement le double.

3.2 - PROTECTION D'ECRITURE

Les disques souples peuvent être protégés en écriture pour éviter de procéder à des opérations d'écriture. Dans ce cas une tentative d'écriture provoque l'édition d'un message d'erreur. Il est judicieux en général de mettre en protection d'écriture les disques contenant un nombre important de fichiers à préserver, ou les disquettes "système".

Une mini-disquette peut être protégée en écriture en collant une pièce de papier opaque à l'endroit de la découpe sur un côté de la pochette.

Pour les disques 8" c'est le contraire, c'est-à-dire qu'il faut découvrir la découpe pour protéger le disque (si la découpe existe).

3.3 - LE BOUTON RESET

Le bouton RESET situé à droite, sous votre système, ne doit pas être actionné pendant que le disque fonctionne. Il y a en effet, risque, si le système est en cours d'écriture, que votre disque soit endommagé. Pour arrêter le système FLEX, utiliser le caractère d'échappement (voir la commande TTYSET).

3.4 - NOTES SUR LA COMMANDE P

La commande P charge le sous programme de gestion de l'imprimante (le fichier PRINT.SYS) à partir du disque où P a été trouvée. Pour satisfaire aux caractéristiques de ce fichier et pour écrire votre propre fichier PRINT.SYS, consulter les descriptions précédentes de ce manuel se rapportant à ces ordres ainsi que le guide de programmation avancée.

3.5 - ACCES A DES LECTEURS NE CONTENANT PAS DE DISQUETTES

Si on essaye d'accéder à un lecteur de mini-disque 5" ne contenant pas de disquette, le système se met en attente de lecture jusqu'à ce que l'on insère une disquette et que l'on ferme la porte d'accès. Vous pouvez aussi actionner le RESET et relancer l'exécution à l'adresse $CD03.

Le problème ne se pose pas avec des lecteurs 8", le FLEX connaissant l'état du lecteur (message LECTEUR NØN PRET).

3.6 - ERREURS SYSTEMES

Chaque fois que FLEX détecte une erreur pendant une opération, un message d'erreur est affiché sur l'écran de GOUPIL. De manière interne un code d'erreur est transformé par l'intermédiaire d'une table d'erreur appelée ERRØRS.SYS en un message complet. Si vous avez oublié de copier ce fichier sur le disque système que vous utilisez, FLEX vous donne un code d'erreur par le message suivant :

ERR.DISQUE # XX

où "XX" est un code d'erreur (valeur décimale). La table suivant fournit la liste de ces erreurs et leur signification :

CODE D'ERREUR	SIGNIFICATION
1	CODE DE FONCTION FMS ILLEGAL
2	LE FICHIER REQUIS EST DEJA UTILISE
3	LE FICHIER SPECIFIE EXISTE DEJA
4	LE FICHIER SPECIFIE N'A PAS ETE TROUVE
5	ERREUR CATALOGUE SYSTEME - RECHARGEZ LE SYSTEME
6	PLUS D'ESPACE DISQUE POUR LE CATALOGUE
7	TOUT L'ESPACE DISQUE A ETE UTILISE
8	FIN DE FICHIER RENCONTRE EN LECTURE
9	ERREUR DE LECTURE SUR DISQUE
10	ERREUR D'ECRITURE SUR DISQUE
11	FICHIER OU DISQUE PROTEGE EN ECRITURE
12	FICHIER PROTEGE - FICHIER NON EFFACE
13	BLOC DE CONTROLE DE FICHIER ILLEGAL
14	ADRESSE DISQUE ILLEGALE
15	NUMERO D'UNITE DISQUE ILLEGAL
16	UNITE DISQUE NON PRETE
17	LE FICHIER EST PROTEGE - ACCES REFUSE
18	VIOLATION DES ATTRIBUTS D'ACCES A UN FICHIER
19	POINTEUR D'ACCES DIRECT ERRONE
20	FMS INACTIF - RECHARGER LE SYSTEME
21	SPECIFICATION FICHIER ILLEGALE
22	ERREUR SYSTEME EN FERMETURE FICHIER
23	DEBORDEMENT TABLE D'ALLOCATION - DISQUE TROP SEGMENTE
24	NUMERO D'ENREGISTREMENT INEXISTANT
25	ERREUR NUMERO D'ENREGISTREMENT - FICHIER ENDOMMAGE
26	ERREUR DE SYNTAXE COMMANDE - REDONNER LA COMMANDE
27	COMMANDE NON AUTORISEE PENDANT L'IMPRESSION
28	CONFIGURATION HARDWARE INSUFFISANTE

Pour plus de détails concernant ces erreurs, consultez le "Guide de programmation avancée".

3.7 - GEOGRAPHIE DE LA MEMOIRE

Ce qui suit est une liste de l'espace mémoire "RAM" nécessaire au système d'exploitation FLEX. Toutes les adresses sont en hexadécimal.

0000-BFFF	Ram utilisateur * Note : des parties de cet espace sont utilisées par CØPY et autres utilitaires.
C000-DFFF	Système d'exploitation disque
C04A-C07F	Pile FLEX
C100-C6FF	Espace de chargement des commandes utilitaires.
CD00	Adresse du Point d'entrée froid du FLEX
CD03	Adresse du Point d'entrée chaud du FLEX.

Pour plus d'informations se reporter au manuel de programmation avancée.

3.8 - SOUS PROGRAMMES D'ENTREE/SORTIE DU FLEX

Pour que les fonctions d'E/S du FLEX opèrent correctement et que le contrôle de l'environnement soit réalisé, il faut utiliser dans tout programme, les sous programmes du FLEX plutôt que ceux du moniteur résident en REPROM.

On donne ci-dessous une liste de ces sous programmes avec une brève description (les adresses sont en hexadécimal).

GETCHR $CD15

Ce sous programme lit un caractère au clavier et le range dans l'accumulateur A. Une fois appelé, le sous programme boucle sur lui même jusqu'à ce qu'un caractère soit frappé au clavier. Pour toute entrée, il est donc préférable d'appeler ce sous programme par JSR GETCHR (ou JSR $CD15).

Les registres sont affectés de la façon suivante :

ACC A chargé avec le caractère entré au clavier
ACC B non modifié
REG X non modifié.

PUTCHR $CD18

Ce sous programme est utilisé pour sortir un caractère vers l'écran de contrôle.

Pour utiliser PUTCHR, le caractère codé en ASCII doit être placé dans l'accumulateur A. Pour afficher la lettre "A" sur l'écran, le programme suivant peut être utilisé :

```
LDAA  # $41
JSR     $CD18
```

Les registres B et X du micro processeur ne sont pas modifiés lors de l'appel.

PSTRNG $CD1E

Ce sous programme permet d'afficher une chaine de caractère sur l'écran de contrôle. Quand il est appelé, un retour chariot (CR ou ↵) et un interligne (LF ou ↓) sont automatiquement envoyés vers l'écran puis la chaine est affichée. Le registre X doit contenir l'adresse du premier caractère, et la chaine doit se terminer par le code de contrôle 04. Le contrôle de l'affichage peut se faire selon les règles déjà décrites en utilisant au clavier la touche du caractère d'échappement (Espace ou une autre) et le retour chariot (↵).

Les registres sont modifiés de la façon suivante :

ACC A	modifié
ACC B	inchangé
REG X	contient l'adresse du dernier caractère de la chaine, 04, ou le caractère sur lequel on s'est arrêté en utilisant la touche d'échappement.

Pour plus d'informations sur ces sous programmes et d'autres, consulter le "Guide de programmation avancée".

3.9 - CHARGEMENT DU SYSTEME D'EXPLOITATION FLEX

Le chargement du FLEX, en mémoire RAM, à partir du disque, est réalisé par un court programme en ROM (conférer listing du moniteur GPMON, manuel 1) appelé chargeur (BOOTSTRAP en anglais). Ce programme s'exécute quand vous frapper la touche  .

Si le système ne se charge pas correctement, repositonner le disque dans le lecteur et relancer la commande (en particulier avec des lecteurs 5") après avoir appuyer sur le bouton RESET.

```
+0
    1                           NAM    DISK BOOTSTRAP
    2                           OPT    PAG
    3
    4                   **********************************
    5                   *                                *
    6                   * -S- COMMAND                    *
    7                   *   DISK BOOTSTRAP (5")          *
    8                   *                                *
    9                   **********************************
   10
   11                   * EQUATES FOR WD 1791
   12
   13  E8E0             DRVREG  EQU    $E8E0     DRIVE REGISTER
   14  E8F0             COMREG  EQU    $E8F0     COMMAND REGISTER
   15  E8F2             SECREG  EQU    $E8F2     SECTOR REGISTER
   16  E8F3             DATREG  EQU    $E8F3     DATA REGISTER
   17  A100             LOADER  EQU    $A100
   18
   19                   * PROGRAM STARTS HERE
   20
   21  0000 B6 E8 F0    DKBOOT  LDA A  COMREG    TURN MOTOR ON
   22  0003 7F E8 E0            CLR    DRVREG    SELECT DRIVE £0
   23  0006 CE 00 00            LDX    £0
   24  0009 08          OVR     INX              DELAY FOR MOTOR SPEEDUP
   25  000A 09                  DEX
   26  000B 09                  DEX
   27  000C 26 FB               BNE    OVR
   28  000E C6 F4               LDA B  £$F4      DO RESTORE COMMAND
   29  0010 F7 E8 F0            STA B  COMREG
   30  0013 8D 2F               BSR    DELAY
   31  0015 F6 E8 F0    LOOP1   LDA B  COMREG    CHECK WD STATUS
   32  0018 53                  COM B
   33  0019 C5 01               BIT B  £1        WAIT TILL NOT BUSY
   34  001B 26 F8               BNE    LOOP1
   35  001D 86 FE               LDA A  £$FE      SETUP FOR SECTOR £1
   36  001F B7 E8 F2            STA A  SECREG
   37  0022 8D 20               BSR    DELAY
   38  0024 86 73               LDA A  £$73      SETUP READ COMMAND
   39  0026 B7 E8 F0            STA A  COMREG
   40  0029 8D 19               BSR    DELAY
   41  002B CE A1 00            LDX    £LOADER   ADDRESS OF LOADER
   42  002E C5 02       LOOP2   BIT B  £2        DTA PRESENT ?
   43  0030 27 07               BEQ    LOOP3     SKIP IF NOT
   44  0032 B6 E8 F3            LDA A  DATREG    GET A BYTE
   45  0035 43                  COM A
   46  0036 A7 00               STA A  0,X       PUT IN MEMORY
   47  0038 08                  INX              BUMP POINTER
   48  0039 F6 E8 F0    LOOP3   LDA B  COMREG    CHECK WD STATUS
   49  003C 53                  COM B
   50  003D C5 01               BIT B  £1        IS WD BUSY ?
   51  003F 26 ED               BNE    LOOP2     LOOP IF SO
   52  0041 7E A1 00            JMP    LOADER    JUMP TO FLEX LOADER
   53
   54  0044 8D 00       DELAY   BSR    DEL1
   55  0046 8D 00       DEL1    BSR    RTN
   56  0048 39          RTN     RTS
   57
   58                           END

AUCUNE ERREUR DETECTEE
```

```
DISK BOOTSTRAP                    26-11-81  ASSEMBLEUR TSC 6800   PAGE    1


  TABLE SYMBOLES

COMREG E8F0     DATREG E8F3     DEL1   0046     DELAY  0044     DKBOOT 0000
DRVREG E8E0     LOADER A100     LOOP1  0015     LOOP2  002E     LOOP3  0039
OVR    0009     RTN    0048     SECREG E8F2
```

```
                        NAM    8 OR 5" DISK BOOTSTRAP
                        OPT    PAG


                    **********************************
                    *                                *
                    * -S- COMMAND                    *
                    *   DISK BOOTSTRAP               *
                    *   5 OR 8" DMAF2 - 5" MFB       *
                    *                                *
                    **********************************

                    * EQUATES FOR WD 1795 AND DMA

EB20                STAREG  EQU    $EB20
EB21                TRKREG  EQU    $EB21
EB22                SECREG  EQU    $EB22
EB23                DATREG  EQU    $EB23
EB24                DRVREG  EQU    $EB24
EB00                DMAADD  EQU    $EB00
EB02                DMACON  EQU    $EB02
EB10                DMACOM  EQU    $EB10
EB14                DMAPRI  EQU    $EB14
A100                LOADER  EQU    $A100
FFE6                CONTRL  EQU    $FFE6

                    * EQUATES FOR WD 1791

E8E0                DRVR91  EQU    $E8E0
E8F0                COMR91  EQU    $E8F0
E8F2                SECR91  EQU    $E8F2
E8F3                DATR91  EQU    $E8F3

                    * PROGRAM STARTS HERE

0000 86 01      DKBOOT  LDA A  £$01
0002 B7 EB 22           STA A  SECREG
0005 8D 69              BSR    DEL1
0007 B1 EB 22           CMP A  SECREG    TEST IF DMAF2 PRESENT
000A 26 67              BNE    MFBOOT    IF NOT BOOT ON MFB

000C C6 BF      DMABOT  LDA B  £$BF
000E F7 EB 24           STA B  DRVREG    SELECT DRIVE
0011 8D 4C              BSR    READY     TEST READY
0013 27 0E              BEQ    SBOOT     IF READY CONTINUE
0015 C6 7F              LDA B  £$7F      SELECT 5"
0017 F7 EB 24           STA B  DRVREG
001A 8D 49              BSR    ONESEC    ONE SECOND DELAY
001C 8D 41              BSR    READY     TEST READY
001E 27 03              BEQ    SBOOT     IF READY CONTINUE
0020 7E FF E6           JMP    CONTRL
0023 86 09      SBOOT   LDA A  £9
0025 B7 EB 20           STA A  STAREG    SEND RESTOR COMMAND
0028 8D 44              BSR    DELAY
002A 8D 2D              BSR    TEST      WAIT TILL NOT BUSY
002C CE FE FF           LDX    £$FEFF
002F FF EB 02           STX    DMACON    SET BUFFER LENGTH
0032 CE 5E FF           LDX    £$5EFF
0035 FF EB 00           STX    DMAADD    SET BUFFER ADDRESS
```

```
8 OR 5" DISK BOOTSTRAP              26-11-81  ASSEMBLEUR TSC 6800   PAGE    1

   60  0038 86 FD            LDA A  £$FD
   61  003A B7 EB 10          STA A  DMACOM
   62  003D 86 FE            LDA A  £$FE
   63  003F B7 EB 14          STA A  DMAPRI
   64  0042 86 8C            LDA A  £$8C
   65  0044 B7 EB 20          STA A  STAREG    SEND READ COMMAND
   66  0047 FE EB 02  WAIT    LDX    DMACON
   67  004A 8C FE FF          CPX    £$FEFF
   68  004D 27 F8             BEQ    WAIT      WAIT TIL DMA OP. FINISHED
   69  004F 86 FF             LDA A  £$FF      RESET DMA
   70  0051 B7 EB 14          STA A  DMAPRI
   71  0054 8D 03             BSR    TEST
   72  0056 7E A1 00  GOLOAD  JMP    LOADER    GOTO LOADER
   73  0059 8D 04     TEST    BSR    READY     GET STATUS
   74  005B 47                ASR A
   75  005C 25 FB             BCS    TEST      WAIT TIL NOT BUSY
   76  005E 39                RTS
   77  005F B6 EB 20  READY   LDA A  STAREG    GET WD STATUS
   78  0062 85 80             BIT A  £$80
   79  0064 39                RTS
   80
   81  0065 CE 00 00  ONESEC  LDX    £0        ONE SECOND DELAY
   82  0068 08        D15     INX
   83  0069 09                DEX
   84  006A 09                DEX
   85  006B 26 FB             BNE    D15
   86  006D 39                RTS
   87
   88  006E 8D 00     DELAY   BSR    DEL1
   89  0070 8D 00     DEL1    BSR    RTN
   90  0072 39        RTN     RTS
   91
   92                 * BOOTSTRAP FOR MFB
   93
   94  0073 43        MFBOOT  COM A            SET SECTOR ONE
   95  0074 B7 EB F2          STA A  SECR91     AND START MOTOR
   96  0077 7F EB E0          CLR    DRVR91     SELECT DRIVE 0
   97  007A 8D E9             BSR    ONESEC     DELAY ONE SECOND
   98  007C C6 F4             LDA B  £$F4       SEND RESTOR COMMAND
   99  007E F7 EB F0          STA B  COMR91
  100  0081 8D EB             BSR    DELAY
  101  0083 F6 EB F0  LOOP1   LDA B  COMR91
  102  0086 57                ASR B
  103  0087 24 FA             BCC    LOOP1      WAIT TILL NOT BUSY
  104  0089 86 73             LDA A  £$73       SEND READ COMMAND
  105  008B B7 EB F0          STA A  COMR91
  106  008E 8D DE             BSR    DELAY
  107  0090 CE A1 00          LDX    £LOADER    ADDRESS OF LOADER
  108  0093 F6 EB F0  LOOP2   LDA B  COMR91     GET WD STATUS
  109  0096 57                ASR B             CHECK BUSY
  110  0097 25 BD             BCS    GOLOAD     IF NOT BUSY GO TO LOADER
  111  0099 57                ASR B             CHECK DATA PRESENT
  112  009A 25 F7             BCS    LOOP2      LOOP TO WAIT DATA
  113  009C B6 EB F3          LDA A  DATR91     GET DATA
  114  009F 43                COM A
  115  00A0 A7 00             STA A  0,X        STORE IT IN MEMORY
```

```
8 OR 5" DISK BOOTSTRAP              26-11-81  ASSEMBLEUR TSC 6800   PAGE    2

  116  00A2 08              INX              BUMP POINTER
  117  00A3 20 EE           BRA    LOOP2     CONTINUE
  118
  119                       END

AUCUNE ERREUR DETECTEE
```

```
8 OR 5" DISK BOOTSTRAP              26-11-81  ASSEMBLEUR TSC 6800   PAGE    3

  TABLE SYMBOLES

COMR91 E8F0     CONTRL FFEE     D15    0068     DATR91 E8F3     DATREG EB23
DEL1   0070     DELAY  006E     DKBOOT 0000     DMAADD EB00     DMABOT 000C
DMACOM EB10     DMACON EB02     DMAPRI EB14     DRVR91 E8E0     DRVREG EB24
GOLOAD 0056     LOADER A100     LOOP1  0083     LOOP2  0093     MFBOOT 0073
ONESEC 0065     READY  005F     RTN    0072     SBOOT  0023     SECR91 E8F2
SECREG EB22     STAREG EB20     TEST   0059     TRKREG EB21     WAIT   0047
```

3.10 - INFORMATIONS SUR LE MODULE PRINT.SYS

FLEX est fourni avec deux sous programmes de sortie sur imprimante : PØKI.SYS pour les imprimantes à interface parallèle de type CENTRONICS, PDIABLØ.SYS, pour celles à interface série RS 232.

D'autres sous programmes peuvent être simplement écrit pour gérer d'autres types d'imprimantes, des informations additionnelles sont données dans le Manuel de Programmation Avancée.

3.10.1. - Description des sous-programmes

1) Pour être utilisé avec la commande P, le module d'impression doit résiderdans un fichier nommé PRINT.SYS.

2) Trois sous programmes composent le module :

- PINIT $CCCO initialisation de l'interface
- PCHK $CCD8 test de l'état "prêt" de l'interface
- PØUT $CCE4 sortie d'un caractère.

3) Lors de l'appel du sous programme PØUT, le caractère à imprimer doit être dans le registre A. PINIT peut détruire le contenu des registres. PØUT et PCHK ne doivent pas les modifier (à l'exception de CC).

4) Les sous programmes doivent commencer nécessairement aux adresses données, mais peuvent se prolonger ailleurs en mémoire s'il n'y a pas assez de place. Il faut alors être certain de ne pas provoquer de conflit avec des utilitaires ou des programmes les utilisant.

5) Les trois sous programmes doivent se terminer par une instruction de retour RTS.

3.10.2. Listing du module PRINT.SYS pour imprimante parallèle.

```
 1                          NAM   POKI
 2                          OPT   PAG
 3                  *
 4                  * PRINT.SYS POUR IMPRIMANTE PARALLELE
 5                  * INTERFACE TYPE CENTRONIC
 6                  *
 7
 8                  * INITIALISATION INTERFACE
 9  CCC0                    ORG   $CCC0
10  CCC0 86 FF      PINIT   LDA A £$FF
11  CCC2 B7 E8 42           STA A VIADRB
12  CCC5 86 C3              LDA A £$C3
13  CCC7 B7 E8 4C           STA A VIAPCR
14  CCCA 86 E3              LDA A £$E3
15  CCCC B7 E8 4C           STA A VIAPCR
16  CCCF 39                 RTS
17
18
19
20                  * TEST ETAT
21  CCD8                    ORG   $CCD8
22  CCD8 37         PCHK    PSH B
23  CCD9 F6 E8 4D           LDA B VIAIFR
24  CCDC 59                 ROL B
25  CCDD 59                 ROL B
26  CCDE 59                 ROL B
27  CCDF 33                 PUL B
28  CCE0 39                 RTS
29
30
31
32                  * SOUS PROGRAMME DE SORTIE CARACTERE
33  CCE4                    ORG   $CCE4
34  CCE4 36         POUT    PSH A
35  CCE5 8D F1      POUT1   BSR   PCHK
36  CCE7 2A FC              BPL   POUT1
37  CCE9 B7 E8 40           STA A VIAES
38  CCEC 86 C3              LDA A £$C3
39  CCEE B7 E8 4C           STA A VIAPCR
40  CCF1 86 E3              LDA A £$E3
41  CCF3 B7 E8 4C           STA A VIAPCR
42  CCF6 32                 PUL A
43  CCF7 39                 RTS
44
45                  * ADRESSES DU VIA CARTE E/S
46  E840            VIAES   EQU   $E840
47  E842            VIADRB  EQU   $E842
48  E84C            VIAPCR  EQU   $E84C
49  E84D            VIAIFR  EQU   $E84D
50                          END

AUCUNE ERREUR DETECTEE

POKI                            26-11-81  ASSEMBLEUR TSC 6800   PAGE    1


  TABLE SYMBOLES

PCHK   CCD8    PINIT  CCC0    POUT   CCE4    POUT1  CCE5    VIADRB E842
VIAES  E840    VIAIFR E84D    VIAPCR E84C
```

3.10.3 - Listing du module PRINT.SYS série

Le programme donné en exemple permet de sortir sur une imprimante série au moyen d'une interface de type ACIA, à la vitesse de 1200 BDS, sans contrôle de parité, et avec une procédure de contrôle de type DC1/DC3.

```
    1                       NAM    PDIABLO
    2                       OPT    PAG
PDIABLO                             26-11-81  ASSEMBLEUR TSC 6800   PAGE    1

    4                 *
    5                 * PRINT.SYS DRIVER FOR SERIAL PRINTER DIABLO
    6                 * WITH DC1/DC3 BUFFER PROTECTTION
    7                 *
    8                 * SMT JUIN 80
    9                 *
   10                 * MISE A JOUR POUR 56 K LE 29/7/81
   11                 *
   12                 *
   13  E80C           ACIA    EQU    $E80C     ACIA ADDRESS
   14  E81B           VIA1    EQU    $E81B
   15  E814           VIA2    EQU    $E814     VIA ADDRESS
   16                 *
   17                 * PRINTER INITIALISATION
   18                 *
   19  CCC0                       ORG    $CCC0     MUST RESIDE AT $CCC0
   20  CCC0 86 03     PINIT   LDA A  £3        RESET ACIA
   21  CCC2 B7 E8 0C          STA A  ACIA
   22  CCC5 86 11             LDA A  £$11      SET 8 BITS AND 2 STOPS
   23  CCC7 B7 E8 0C          STA A  ACIA
   24  CCCA 86 C0             LDA A  £$C0      CLOCK FOR 1200 BDS
   25  CCCC CE 18 00          LDX    £$1800
   26  CCCF B7 E8 1B          STA A  VIA1
   27  CCD2 FF E8 14          STX    VIA2
   28  CCD5 39                RTS               RETURN
   29                 *
   30                 * CHECK FOR PRINTER READY
   31                 *
   32  CCD8                       ORG    $CCD8     PRINT TEST AT $CCD8
   33  CCD8 37        PCHK    PSH B             SAVE B
   34  CCD9 F6 E8 0C          LDA B  ACIA      GET STATUS
   35  CCDC 56                ROR B             GET TDR BIT INTO
   36  CCDD 56                ROR B             SIGN POSITION
   37  CCDE 56                ROR B
   38  CCDF 33                PUL B             RESTORE B
   39  CCE0 39                RTS               RETURN
   40                 *
   41                 * PRINTER OUTPUT CHARACTER ROUTINE
   42                 *
   43  CCE4                       ORG    $CCE4     MUST RESIDE AT $CCE4
   44  CCE4 37        POUT    PSH B             SAVE B
   45  CCE5 F6 E8 0D  POUT1   LDA B  ACIA+1    GET RECEIVED CHAR IF ANY
   46  CCE8 C1 13             CMP B  £$13       TEST IF DC3
   47  CCEA 27 F9             BEQ    POUT1     IF YES WAIT FOR DC1
   48  CCEC 8D EA     POUT2   BSR    PCHK      CHECK READY
   49  CCEE 2A FC             BPL    POUT2     LOOP IF NOT
   50  CCF0 33                PUL B             RESTORE B
   51  CCF1 B7 E8 0D          STA A  ACIA+1    WRITE OUT CHARACTER
   52  CCF4 39                RTS               RETURN
   53                         END

AUCUNE ERREUR DETECTEE

TABLE SYMBOLES

ACIA   E80C    PCHK   CCD8    PINIT  CCC0    POUT   CCE4    POUT1  CCE5
POUT2  CCEC    VIA1   E81B    VIA2   E814
```

4.1 - RESUME DES COMMANDES

APPENDØ <nom du fichier>[Ø <liste de fichiers>] Ø <nom du fichier>
Extension par défaut :.TXT
Page de description 15

ASN [ØW= <disque>][ØS= <disque>]
Page de description 16

BACKUPØ <n° disque> Ø <n° disque>
Page de description 17 bis

BUILDØ <nom du fichier>
Extension par défaut :.TXT
Page de description 18

CAT[Ø <n° de disque>][Ø <paramètres>]
Page de description 19

CØPYØ <nom du fichier> Ø <nom du fichier>
CØPYØ <nom du fichier> Ø <disque>
CØPYØ <disque> Ø <disque> [Ø <paramètres>]
Page de description 21

DATE [Ø <jourØmoisØannée>]
Page de description 23

DELETEØ <nom du fichier> [Ø <liste de fichiers>]
Page de description 24

EXECØ <nom du fichier>
Extension par défaut :.TXT
Page de description 25

GETØ <nom de fichier> [Ø <liste de fichiers>]
Extension par défaut : .BIN
Page de description 13

IØ <nom du fichier> Ø <commande>
Extension par défaut : .TXT
Page de description 27

JUMPØ <adresse hexadécimale>
Page de description 28

LINKØ <nom du fichier>
Extension par défaut : .SYS
Page de description 29

LISTØ <nom du fichier>[Ø<n° ligne 1-n° ligne 2> Ø <+ options>
Extension par défaut : .TXT
Page de description 30

MEMTESTØ <adresse début> Ø <adresse fin>
Page de description 31bis

MEMTU
Page de description 31ter

MØN
Page de description 13

NEWDISK <disque>
Page de description 31quarter

ØØ <nom de fichier> Ø <commande>
Extension par défaut : .ØUT
Page de description 32bis

PØ <commande>
Page de description 33

PRINTØ <nom de fichier>
Page de description 33bis

PRØTØ <nom de fichier>[Ø <liste d'options>
Page de description 34

QCHECK
Page de description 34bis

RENAMEØ <nom de fichier 1> Ø <nom de fichier 2>
Extension par défaut : .TXT
Page de description 35

SAVEØ <nom de fichier> Ø <adr de début> Ø <adr de fin>[Ø<adr de transfert>]
Extension par défaut : .TXT
Page de description 37

STARTUP
Page de description 39

TESTØ <n° disque>
Page de description 40bis

TTYSET [Ø <liste de paramètres>]
Page de description 41

VERIFY [Ø <ØN or ØFF>]
Page de description 44

VERSIØNØ <nom de fichier>
Extension par défaut : .CMD
Page de description 45

XØUT[Ø <numéro du disque>]
Page de description 46

Imprimé en France. — Imprimerie JOUVE, 18, rue Saint-Denis, 75001 PARIS

goupil

Société de Micro-informatique et Télécommunications

5.
MANUEL DE PROGRAMMATION AVANCEE DU DOS FLEX-GOUPIL 2

1982

NOTE

Les éléments qui suivent, listings et documentations sont produits pour la satisfaction et l'usage personnel par la Société de Micro-informatique et Télécommunications SMT.

Toute reproduction, même partielle, est interdite et constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1975 sur la protection des droits d'auteur.

L'application de cette règle permettra de vendre de plus en plus de programmes à des prix de plus en plus bas.

Son respect est la condition nécessaire pour que la SMT, ou d'autres sociétés, puissent produire, acheter ou adapter toujours plus de logiciels de qualité et que la créativité en ces matières se développe à votre profit.

PREFACE

Le propos de ce manuel est de fournir à l'utilisateur toutes les informations nécessaires à l'utilisation des sous programmes et des fonctions offertes par le système d'exploitation FLEX. Ce manuel peut être utilisé pour toutes les versions du FLEX (versions 5", 8"...). Il est déconseillé de l'ouvrir avant d'être parfaitement familiarisé avec l'utilisation du système GOUPIL et de ses disques.

Ce manuel est destiné aux programmeurs ayant acquis une bonne expérience des techniques de programmation en langage assembleur.

Il s'applique à la dernière version G2FLEX de l'adaptation du système FLEX au micro-ordinateur GOUPIL.

SOMMAIRE

AVERTISSEMENT

Le logiciel doit être utilisé selon les indications données dans ce document et le manuel d'utilisation. La SMT ne sera pas responsable d'un usage incorrect et destructif des fonctions et des paramètres, du système d'exploitation. Le lecteur est invité à porter son attention sur les diverses mises en garde qu'il trouvera dans ce document.

En aucun cas, la SMT ne sera tenue pour responsable de destructions provoquées par des modifications du produit effectuées sans son accord.

1 - INTRODUCTION

FLEX peut être divisé en trois grandes parties qui sont :

- le DØS ("Disk Operating System") qui exécute les commandes,

- le FMS ("File Management System") qui réalise la gestion des fichiers sur les disquettes,

- l'UCS ("Utility Command Set") qui regroupe l'ensemble des commandes utilitaires dont dispose l'utilisateur.

Les commandes sont décrites dans le manuel d'utilisation du FLEX. Nous nous intéresserons ici aux fonctions du DOS et du FMS afin de permettre au programmeur d'écrire ses propres commandes et de mettre en oeuvre une application utilisant des fichiers disques.

Lors de la mise au point des programmes qui traitent des fichiers sur disque avec l'aide de FMS, le programmeur doit prendre les précautions suivantes :

(i) protéger en écriture la disquette système en couvrant ou découvrant la découpe de protection de la disquette (voir le manuel d'utilisation pour plus de détails sur cette opération). Ceci afin d'éviter une éventuelle destruction du système en cas d'erreur de programmation, l'exécution d'un programme erroné pouvant avoir des effets inattendus et dévastateurs.

(ii) utiliser une disquette de travail pour la création ou le traitement de fichiers par un programme. Si des données doivent être préservées, en établir une copie pour ne pas les perdre en cas de fausse manoeuvre et de destruction du contenu de la disquette.

(iii) tester le programme dans les "cas anormaux" où une donnnée lue peut ne pas correspondre à ce que le programme attend. Un programme bien écrit doit pouvoir détecter les erreurs et contrôler sa bonne exécution. C'est à dire que lorsqu'une erreur se produit au cours d'un traitement, ou d'une opération d'entrée/sortie par exemple, il faut prévoir un traitement particulier de reprise ou carrément d'abandon du programme.

2 - LE MONITEUR D'EXPLOITATION DISQUE

2.1. PRESENTATION

Le DØS, système d'exploitation disque, assure le lien entre l'utilisateur, via le clavier et l'écran, et le système de gestion des fichiers disque. Les commandes sont lues et traitées par le DOS. Les fonctions telles que l'analyse syntaxique des noms de fichiers (nom + extension + numéro d'unité disque), ou des arguments des commandes, la gestion des entrées/sorties du clavier et de l'écran, les rapports d'erreur sont aussi toutes assurés par le DØS.

Les paragraphes suivants donnent la définition des variables globales du DØS et leur adresse en mémoire, et décrivent les sous programmes appelables par l'utilisateur.

2.2. VARIABLES GLOBALES

Toutes les variables intéressant le programmeur sont rangées dans deux zones mémoires du DØS, $C080 - $COFF et $CC00 - CCFF. Certains emplacements de ces zones sont réservés à l'usage de celui-ci (constante, mémoire de travail,...) et ne doivent en aucun cas être utilisés par un programme de l'utilisateur qui risquerait de provoquer des destructions.

2.3. CARTE MEMOIRE DES VARIABLES ET PARAMETRES

2.3.1. TAMPON LIGNE

$C080 - $COFF : tampon ligne.

Bloc de 128 octets où le sous programme INBUF range les caractères frappés au clavier. Tous les caractères, à l'exception des caractères de contrôle sont placés dans le tampon. Les caractères effacés par la touche "retour arrière" ne sont pas mémorisés dans le tampon, ainsi que la touche "retour arrière" elle-même. Le retour chariot signalant la fin de l'entrée d'une ligne est rangé dans le tampon. Le tampon est aussi utilisé pour charger le fichier STARTUP lors d'un démarrage à froid du système disque (commande [↵]).

2.3.2. VARIABLES DE CONTROLE DE L'ENVIRONNEMENT DU SYSTEME

$CC00 : caractère de retour arrière TTYSET BS

C'est le caractère interprété par le sous programme INBUF comme retour arrière (backspace). Il peut être défini par l'utilisateur au moyen de la commande utilitaire TTYSET. Sa valeur par défaut est le caractère ASCII BS, soit $08 en hexadécimal (ou aussi CTRL - H).

$CC01 : caractère d'effacement TTYSET DL

C'est le caractère interprété par le sous programme INBUF comme caractère d'effacement ou d'annulation de ligne. Il peut être défini par l'utilisateur au moyen de la commande TTYSET. C'est par défaut le caractère ASCII CAN (ou CTRL - X) soit $18.

$CC02 : caractère séparateur (EOL) TTYSET EL

C'est le caractère séparant plusieurs commandes dans une ligne (commande multiple). Il peut être défini par l'utilisateur au moyen de la commande TTYSET. Valeur par défaut $3A, caractère ASCII deux points ":".

$CC03 : nombre de lignes par page TTYSET DP

Cet octet détermine le nombre de lignes que FLEX imprime sur une page avant de se mettre en pause ou de commander un saut à la page suivante. Il peut être défini par l'utilisateur au moyen de la commande TTYSET. Sa valeur par défaut est 24.

$CC04 : nombre de caractères par ligne TTYSET WD

Cet octet fixe le nombre de caractères par ligne. S'il est nul, il n'y a pas de limite à la longueur de la ligne. Sa valeur est 80 par défaut et peut être redéfinie au moyen de la commande TTYSET.

$CC05 : temporisation retour chariot

Cet octet donne le nombre de caractères nuls ou de remplissage à envoyer après la sortie d'un retour chariot de façon à tenir compte du temps de retour chariot non négligeable de certains terminaux imprimants. Sa valeur par défaut est 0 et peut être redéfinie par la commande TTYSET.

$CC06 : caractère de tabulation TTYSET TB

Cet octet définit un caractère de tabulation qui peut être utilisé par des programmes tel que l'éditeur. Le FLEX ne l'utilise pas. Sa valeur par défaut est 0, il n'est pas défini et peut être fixé par la commande TTYSET.

$CC07 : caractère écho du retour arrière TTYSET BE

C'est le caractère que le sous programme INBUF envoie comme écho à la réception du caractère retour arrière. Si il a la même valeur $08 que le caractère retour arrière, FLEX envoie un espace $20 avant l'écho ce qui permet d'effacer sur un écran le caractère affecté par le retour arrière. Sa valeur par défaut est 08.

$CC08 : nombre de lignes sautées entre deux pages TTYSET EJ

Cet octet indique à FLEX le nombre de lignes à sauter entre deux pages, une page ayant le nombre de lignes défini en $CC03. La valeur par défaut est 0, il n'y a pas de mise en page, elle peut être redéfinie par la commande TTYSET.

$CC09 : drapeau de pause entre pages TTYSET PS

Ce drapeau permet d'arrêter la sortie entre chaque page. Si l'octet est nul, le FLEX arrête la sortie (sur écran ou imprimante) à la fin de chaque page, s'il est non nul il n'y pas de pause. Le caractère d'échappement permet de reprendre l'édition. Sa valeur par défaut est 0 et peut être modifiée par la commande TTYSET.

$CC0A : caractère d'échappement TTYSET ES

La frappe de ce caractère provoque l'arrêt de l'édition en fin de ligne. Sa valeur par défaut est $20, le caractère ASCII espace. Pour reprendre la sortie, frapper à nouveau ce caractère.

2.3.3. VARIABLES SYSTEMES

$CC0B : numéro de l'unité système

C'est le numéro de l'unité de disquette à partir de laquelle les commandes sont chargées. Si la valeur de cet octet est $FF, la recherche se fera sur toute les unités prêtes. L'unité par défaut est l'unité 0. Peut être assigné par la commande ASN.

$CC0C : numéro de l'unité de travail

C'est le numéro choisi par défaut de l'unité de disquette utilisé pour la recherche ou la création d'un fichier. Si la valeur de cet octet est $FF, la recherche se fera sur toutes les unités prêtes. Sa valeur par défaut est 1. Peut être assigné par la commande ASN.

$CC0D : réservée au système

$CC0E - $CC10 : date

Ces trois octets sont utilisés pour ranger la date donnée lors du chargement du moniteur. Elle est mémorisée en binaire, le jour dans le second octet, le mois dans le premier et l'année dans le troisième (chiffre des unités et dizaines seulement).

$CC11 : caractère terminal

Cet octet contient le dernier caractère non alphanumérique rencontré dans la lecture séquentielle du tampon de ligne. Voir les commentaires sur les sous programmes NXTCH et CLASS dans le paragraphe sous programmes système appelables par l'utilisateur.

$CC12 - $CC13 : adresse de la table de commande utilisateur.

Le programmeur peut ranger dans ces deux octets l'adresse de sa table de commande. Se référer au chapitre concernant les commandes écrites par l'utilisateur pour les détails. La valeur de cette adresse est 0 par défaut, c'est à dire qu'il n'y a pas de table de commande utilisateur.

$CC14 - $CC15 : pointeur du tampon de ligne

Ces deux octets contiennent l'adresse du prochain caractère à traiter dans le tampon de ligne. Voir les sous programmes INBUFF, NXTCH, GETFIL, GETCHR et DØCMND dans le paragraphe traitant des sous programmes système.

$CC16 - $CC17 : adresse de retour sur échappement

Ces deux octets contiennent l'adresse de retour si la frappe d'un retour chariot suit celle d'un caractère d'échappement pour arrêter une sortie. Se referer à la description de la commande TTYSET dans le manuel d'utilisation du FLEX pour plus d'informations sur la procédure d'échappement. Voir aussi la documentation du sous programme PCRLF.

$CC18 : caractère courant

Cet octet contient le dernier caractère traité par le sous programme NXTCH dans le tampon ligne. Voir le sous programme pour plus de détails.

$CC19 : caractère précédent

Cet octet contient le précédent caractère traité par le sous programme NXTCH dans le tampon ligne. Voir le sous programme pour plus de détails.

$CC1A : numéro de la ligne courante

Cet octet contient le nombre de lignes sorties dans une page. Comparé au nombre de lignes par page, il permet de déterminer la fin de la page.

$CC1B - $CC1C : déplacement de l'adresse de chargement

Ces deux octets contiennent le déplacement à ajouter à l'adresse de chargement d'un programme à partir du disque. Voir le sous programme LØAD pour plus de détails. Cet emplacement est aussi utilisé comme mémoire de travail par certains modules du FLEX.

$CC1D : drapeau de transfert

Après le chargement en mémoire d'un programme à partir du disque, cet octet est non nul si une adresse de transfert a été trouvée au cours du chargement. Il est aussi utilisé par certains modules du FLEX.

$CC1E - $CC1F : adresse de transfert

Si le drapeau de transfert est non nul après le chargement d'un programme, ces deux octets contiennent la dernière adresse de transfert rencontrée (c'est à dire l'adresse d'exécution du programme). Si le drapeau est à zéro, le contenu est indéterminé.

$CC20 : code d'erreur

Cet octet contient le numéro d'erreur retourné en cas d'erreur par plusieurs des fonctions du système de gestion des fichiers (FMS).

$CC21 : drapeau d'entrée/sortie spéciale

Si cet octet est non nul, le sous_programme PØTCHR ignore le nombre de caractères par ligne (TTYSET WD) et le caractère d'échappement (TTYSET ES). Le sous-programme RSTRIØ remet le drapeau à 0. La valeur par défaut est 0.

$CC22 : aiguillage de la sortie

Si cet octet est nul, le sous-programme de sortie PUTCHR utilise le sous programme ØUTCH, sinon le sous-programme ØUTCH2. Voir la description de ces sous-programmes pour plus de détails.

$CC23 : aiguillage de l'entrée

Si cet octet est nul, le sous_programme d'entrée GETCHR utilise le sous programme INCH, sinon le sous_programme INCH2. Voir la description de ces sous_programmes pour plus de détails.

$CC24 - $CC25 : adresse du FCB de sortie

S'ils ne sont pas nuls, ces deux octets contiennent l'adresse d'un FCB (bloc de contrôle de fichier) utilisé pour la sortie dans un fichier. S'ils sont nuls, la sortie n'est pas exécutée. Voir le sous programme ØUTCHR pour plus de détails. Les deux octets sont remis à 0 par le sous programme RSTRIØ.

$CC26 - $CC27 : adresse du FCB d'entrée

S'ils ne sont pas nuls, ces deux octets contiennent l'adresse d'un FCB (bloc de contrôle de fichier) utilisée pour l'entrée à partir d'un fichier. S'ils sont nuls. l'entrée n'est pas exécutée. Voir le sous programme GETCHR pour plus de détails. Le sous programme RSTRIØ remet à 0 ces deux octets.

$GC28 : drapeau commande

Cet octet est non nul si le FLEX est appelé par un programme utilisateur via le point d'entrée DØCMND. Voir la documentation de DØCMND pour plus de détails.

$CC29 : compte des caractères sortis

Cet octet contient le compte des caractères sortis dans une ligne. Il est comparé au nombre de caractères par ligne TTYSET WD pour déterminer la fin de la ligne. La sortie d'un caractère de contrôle remet ce compte à 0..

$CC2A : réservé au système

$CC2B - $CC2C : adresse de fin de la mémoire utilisateur

Ces deux octets contiennent l'adresse de la dernière cellule mémoire de l'utilisateur. Le contenu est initialisé lors du chargement du système et peut être lu par le programme ayant besoin de cette information.

$CC2D - $CC2E : aiguillage fichier d'erreurs

Si ces deux octets sont nuls, le sous programme RPTERR utilise ERRØRS.SYS comme fichier contenant les messages d'erreur. Sinon ils contiennent l'adresse du nom du fichier à utiliser. Voir le sous programme RPTERR pour plus de détails.

$CC2F : drapeau d'écho de lecture fichier

Si cet octet est non nul (valeur par défaut) et que l'entrée est faite à partir d'un fichier, le caractère lu est envoyé sur le canal de sortie. Si l'octet est nul il n'y a pas d'écho.

$CC30 - $CC4D : réservés au système

$CC4E - $CCBF : constantes du système

$CCC0 - $CCD7 : initialisation imprimante

Cet espace est réservé pour le chargement du sous programme d'initialisation de l'imprimante.

$CCD8 - $CCE3 : test de l'état de l'imprimante

Cet espace est réservé pour le chargement du sous programme de test de l'imprimante.

$CCE4 - $CCF7 : sortie imprimante

Cet espace est réservé pour le chargement du sous programme de sortie d'un caractère vers l'imprimante.

$CCF8 - $CCFF : réservés au système.

2.4. SOUS-PROGRAMMES DU MONITEUR

A défaut d'indications contraires, les sous programmes doivent être appelés par une instruction JSR et le contenu de tous les registres doit être présumé détruit par leur exécution.

2.4.1. POINTS D'ENTREE DU MONITEUR

$CD00 CØLDS : point d'entrée à froid

Le chargeur disque se branche à cette adresse pour initialiser le système chargé en mémoire. Le DØS et le FMS sont initialisés, ensuite FLEX s'identifie par le message habituel et demande la date. Si un fichier STARTUP existe il est directement chargé et exécuté. Le point d'entrée est réservé au chargeur et ne doit pas être utilisé par le programmeur qui pourrait risquer de détruire le contenu de la disquette système.

$CD03 WARMS : point d'entrée à chaud

Pour les programmes utilisateurs c'est le point d'entrée principal dans le DØS. Le branchement doit se faire par une instruction JMP. La pile système et le registre de retour sur échappement ($CC16 - $CC17) sont initialisés, et le moniteur est prêt à traiter de nouvelles commandes. Le moniteur vérifie le dernier caractère terminal ($CC11), s'il trouve le caractère TTYSET EL (fin de ligne, séparateur de commandes) il traitera la commande en attente dans le tampon de ligne (commande multiple). Sinon il ira lire une nouvelle ligne au clavier. Si le DØS a été appelé par un programme utilisateur, via le point d'entrée DØCMND, le contrôle sera rendu au programme à la fin du traitement des commandes rangées dans le tampon de ligne.

$CD06 RENTER : point d'entrée directe

C'est un point d'entrée directe dans la bouche principale du DØS, aucune initialisation n'est effectuée. Le branchement se fait par une instruction JMP. Normalement ce point d'entrée est utilisé de manière interne par le système et les programmes utilisateurs n'en ont pas besoin.

2.4.2. ENTREE D'UN CARACTERE

$CD09 INCH entrée caractère
$CDOC INCH2 entrée caractère

Ces deux sous programmes lisent un caractère au clavier et le retournent au programme appelant dans le registre A. Ils peuvent être remplacés par des sous programmes adaptés à la configuration de l'utilisateur. Le sous programme GETCHR utilise normalement INCH, mais si le drapeau "d'aiguillage de l'entrée ($CC23) est non nul, il utilisera INCH2. Un programme utilisateur peut changer le vecteur de branchement à l'adresse INCH pour utiliser un autre sous programme gérant par exemple, un lecteur de ruban papier. Le vecteur de branchement à l'adresse INCH2 ne doit jamais être modifié. Lors d'une entrée à chaud dans le DØS par le point WARMS, le drapeau d'aiguillage de l'entrée est remis à 0 et le vecteur de branchement INCH est réinitialisé pour pointer sur le même sous programme que le vecteur INCH2. Le sous programme RSTRIØ effectue lui aussi cette réinitialisation. Ces sous programmes ne tiennent pas compte des paramètres TTY SET.

$CD15 GETCHR : acquisition d'un caractère au clavier

Ce sous programme permet l'acquisition d'un caractère au clavier, et le retourne au programme appelant dans le registre A. Son appel provoque la remise à zéro du numéro de ligne courante ($CC1A). Il prend en compte les paramètres TTY SET, son utilisation est donc préférable à celle du INCH. Si le drapeau aiguillage de l'entrée est non nul, le sous programme INCH2 est utilisé pour l'entrée. S'il est nul, les deux octets $CC26 - $CC27, adresse FCB d'entrée, sont testés. S'ils sont nuls, le caractère est lu par le sous programme INCH, sinon ils contiennent l'adresse d'un FCB pour lire le caractère dans un fichier précédemment ouvert. Les registres B et X sont sauvegardés.

2.4.3. SORTIE D'UN CARACTERE

$CDOF ØUTCH sortie caractère
$CD12 ØUTCH2 sortie caractère

A l'appel de ces sous programmes le registre A doit contenir le caractère à sortir vers le périphérique désiré, normalement l'écran. Les vecteurs ØUTCH et ØUTCH2 pointent de façon standard vers le même sous programme de sortie sur écran ; cependant ØUTCH peut être modifié par l'utilisateur pour exécuter un autre sous programme de sortie vers un périphérique différent, par exemple une imprimante.

Le sous-programme PUTCHR, décrit ci-dessous, utilise ØUTCH ou ØUTCH2 selon le contenu du drapeau "aiguillage de sortie" ($CC22). Lors d'une entrée à chaud dans le moniteur (WARMS) l'aiguillage de sortie est remis à zéro, et ØUTCH est réinitialisé pour pointer le même sous programme que ØUTCH2. Le sous programme RSTRIØ effectue la même réinitialisation. Ces sous programmes ne tiennent pas compte des paramètres TTY SET.

$CD18 PUTCHR : écriture caractère

Ce sous programme sort le caractère contenu dans le registre A vers le périphérique choisi, normalement l'écran, en prenant en compte les paramètres TTY SET. Si le drapeau d'E/S spéciale ($CC21) est nul, le compte des caractères est testé ($CC29) et si la ligne courante est pleine, on passe alors à la suivante. Si un caractère d'échappement TTY SET a été frappé au clavier, la sortie s'arrête à la fin de la ligne courante. Le sous programme ØUTCH2 est utilisé pour envoyer le caractère si l'aiguillage de sortie ($CC22) est non nul, sinon l'adresse FCB de sortie est testée ($CC24 - $CC25). Si elle n'est pas nulle, c'est l'adresse du FCB d'un fichier de sortie, précédemment ouvert, et le caractère est écrit dans le fichier. Si elle est nulle, le sous programme ØUTCH est utilisé pour envoyer le caractère, normalement vers l'écran, mais l'utilisateur peut modifier le vecteur ØUTCH pour exécuter un autre sous programme de sortie. Les registres B et X sont préservés.

2.4.4. LECTURE D'UNE LIGNE

$CD18 INBUFF : entrée dans le tampon ligne

Ce sous programme lit une ligne au clavier et la range dans le tampon ligne. Les caractères de retour arrière et d'effacement TTYSET BS et TTYSET DL sont traités. Tous les autres caractères de contrôle sont ignorés, à l'exception du retour chariot et du saut de ligne. Le retour chariot est placé dans le tampon à la fin de la ligne, le saut à la ligne crée un blanc dans le tampon et provoque un retour chariot sur l'écran pour continuer l'entrée sur une nouvelle ligne. 128 caractères peuvent être entrés dans le tampon y compris le retour chariot final. Au cas où plus de caractères seraient frappés, la ligne est tronquée au 127ième, un retour chariot étant placé dans le 128ième. A la sortie du sous programme, le pointeur du tampon de ligne ($CC14 - $CC15) est initialisé sur le premier caractère.

Attention : les commandes du système entrées au clavier sont gardées dans le tampon de ligne. L'utilisation de INBUFF par un programme utilisateur peut donc détruire des commandes qui n'ont pas encore été traitées par le moniteur. Il est donc prudent de réserver l'usage du tampon de ligne à l'entrée de commandes du moniteur, pour ne pas risquer des effets imprévisibles.

2.4.5. SORTIE D'UNE CHAINE DE CARACTERES

$CD1E PSTRNG : sortie d'une chaine de caractère

Ce sous programme envoie un retour chariot et un saut de ligne avant de sortir la chaine de caractères. A l'appel, le registre X doit contenir l'adresse du premier caractère de la chaine, qui doit se terminer par le caractère ASCII EØT ($04). Les paramètres TTYSET sont pris en compte. Le registre B est préservé.

2.4.6. CLASSEMENT D'UN CARACTERE

$CD21 CLASS : classement d'un caractère

Le sous programme teste si un caractère est alphanumérique ou non. En entrée le caractère doit se trouver dans le registre A, en sortie, si le caractère est alphanumérique (une lettre ou un chiffre) le bit de retenue (carry) du code de condition est mis à zéro ; sinon la retenue est mise à un et le caractère rangé dans le "dernier caractère terminal" ($CC11). Tous les registres sont préservés.

2.4.7. TRAITEMENT DU RETOUR CHARIOT/SAUT DE LIGNE

$CD24 PCRLF : sortie retour chariot/saut de ligne

Ce sous programme envoie un retour chariot et un saut de ligne (caractères ASCII CR et LF) en prenant en compte un certain nombre de paramètres TTYSET. Avant toute chose, le sous programme teste si un caractère d'échappement (TTYSET ES) a été frappé au clavier pendant la sortie de la ligne précédente. Si oui, il attend l'entrée d'un autre caractère d'échappement pour continuer ou d'un retour chariot qui provoquera l'abandon du travail en cours. Sur un retour chariot, le sous programme met à zéro le dernier caractère terminal ($CC11) pour supprimer les éventuelles commandes restant dans le tampon de ligne et se branche à l'adresse contenue dans le registre de retour sur échappement. Si elle n'est pas modifiée par l'utilisateur, cette adresse est celle du point d'entrée à chaud du moniteur, WARMS. Sur un caractère d'échappement (si nécessaire), le

numéro de la ligne courante est testé : si la page est pleine et qu'une pause est demandée entre chaque page (TTYSET PS), le sous programme attend la frappe d'un autre caractère d'échappement ou d'un retour chariot comme ci-dessus. Toutes les pauses sont faites avant la sortie des deux caractères CR et LF, suivi du nombre de caractères nuls spécifiés par la temporisation retour chariot (TTYSET NL). Quand la page courante est pleine, le sous programme sort le nombre de saut de ligne spécifié en $CC08 (TTYSET EJ). Le registre X est préservé.

2.4.8. LECTURE DU TAMPON LIGNE

$CD27 NXTCH : lecture du caractère suivant

Ce sous programme permet de lire séquentiellement le tampon de ligne. Le caractère courant remplace le caractère précédent ($CC19 : = ($CC18), le caractère suivant désigné par le pointeur du tampon ligne devient le caractère courant ($CC18). Les espaces multiples sont ignorés pour ne donner qu'un seul espace. Le pointeur du tampon ligne est augmenté de un pour désigner le caractère suivant, sauf si le caractère courant est un retour chariot ou un séparateur (TTYSET EL). Dans ce dernier cas, tout nouvel appel a NXTCH continuera à donner le retour chariot ou le séparateur, sans passer à la nouvelle commande éventuellement présente dans le tampon. NXTCH sort par le sous programme CLASS, c'est à dire que le caractère se trouve dans le registre A et que la retenue est à zéro s'il est alphanumérique. Les registres B et X sont préservés.

2.4.9. RESTAURATION DES VECTEURS D'E/S

$CD2A RSTRIØ : restauration des vecteurs d'E/S

Ce sous programme restaure les valeurs initiales des vecteurs ØUTCH et INCH pour qu'ils pointent vers les mêmes sous programmes d'E/S que, respectivement, ØUTCH2 et INCH2. Les aiguillages d'entrée et de sortie ($CC23 et $CC22) sont remis à zéro, ainsi que les adresses des FCB d'entrée et de sortie. Les registres A et B sont préservés.

2.4.10. TRANSFERT DES SPECIFICATIONS D'UN FICHIER

$CD2D GETFIL : lecture des spécifications d'un fichier

Ce sous programme est destiné au rangement dans un FCB (conférer le chapitre sur le gestionnaire de fichier) des spécifications d'un fichier (disque, nom, extension) se trouvant dans le tampon de la ligne. A l'entrée le registre X doit contenir

l'adresse du FCB, et le pointeur du tampon ligne doit désigner le premier caractère des spécifications du fichier. Si le disque n'est pas précisé, le numéro de l'unité de travail sera rangé dans le FCB. En sortie la retenue du code de condition est à zéro s'il n'y a pas d'erreur de format, à un sinon. Si l'extension n'est pas précisée, le programme appelant doit utiliser le sous programme SETEXT pour spécifier l'option par défaut. Le pointeur du tampon ligne est laissé sur le caractère suivant le séparateur, sauf si c'est un retour chariot ou une fin de ligne (:). Quand une erreur est détectée, le code d'erreur 21 est rangé dans le FCB. Le registre X est préservé.

2.4.11. CHARGEUR

$CD30 LØAD : chargement d'un fichier

Ce sous programme est destiné au chargement en mémoire d'un fichier binaire résident sur disque. En entrée le FCB du système (en $C840) doit contenir le nom du fichier préalablement ouvert en lecture binaire. Le fichier est chargé en mémoire à l'adresse normalement spécifiée dans le fichier lui-même, mais il est possible de la charger dans une zone mémoire différente en utilisant le déplacement donné en $CC1B - $CC1C. Le déplacement est ajouté à l'adresse de chargement lue dans le fichier ; attention au débordement de l'addition. Si une adresse de transfert est trouvée, elle n'est pas modifiée par le déplacement. Il faut noter aussi que le contenu du fichier n'est pas modifié par le déplacement et qu'il ne s'agit pas d'un procédure de relogement de programme. Si l'on ne veut pas de déplacement il faut le mettre à zéro avant l'appel du sous programme. En sortie, le drapeau de transfert ($CC1D) est à zéro, si aucune adresse de transfert n'a été trouvée, sinon il est non nul et la dernière adresse trouvée est rangée en $CC1E - $CC1F. Le fichier est fermé par le sous programme et si une erreur se produit, un message d'erreur est sorti et le contrôle est transféré au FLEX, au point d'entrée à chaud WARMS.

2.4.12. RANGEMENT DE L'EXTENSION DANS UN FCB

$CD33 SETEXT : Définition de l'extension

Le sous programme permet de définir l'extension d'un fichier si elle n'est pas déjà précisée dans le FCB. En entrée, le registre X doit contenir l'adresse du FCB, le registre A le code numérique de l'extension :

0	BIN	Binaire
1	TXT	Texte
2	CMD	Commande
3	BAS	Basic source
4	SYS	Système
5	BAK	Backup éditeur
6	SCR	

7	DAT	Données
8	BAC	Basic compilé
9	DIR	
10	PRT	
11	ØUT	Fichier imprimante (Spooler).

Si le FCB contient déjà une extension, à la suite d'un appel à GETFIL par exemple, le sous programme ne fait rien. Tout code différent de ceux donnés ci-dessus est ignoré. Le registre X est préservé.

2.4.13. ADDITION DU REGISTRE B AU REGISTRE X

$CD36 ADDBX : addition de B à X

Ce sous programme ajoute le contenu du registre B au contenu du registre X. Le contenu de B est détruit à la sortie.

2.4.14. SORTIE D'UN NOMBRE DECIMAL

$CD39 ØUTDEC : sortie d'un décimal

En entrée, le registre X doit contenir l'adresse de l'octet de poids fort d'un nombre binaire non signé sur 16 bits. Le registre B doit contenir un drapeau de suppression d'espace, s'il est non nul les zéros de tête sont remplacés par des blancs, s'il est nul l'impression commence avec le premier chiffre non nul.

2.4.15. SORTIE D'UN NOMBRE HEXADECIMAL

$CD3C ØUTHEX : sortie d'un hexadécimal

En entrée le registre X doit contenir l'adresse d'un octet qui sera imprimé sous la forme de deux chiffres hexadécimaux. Les registres B et X sont préservés.

2.4.16. RAPPORT D'ERREUR

$CD3F RPTERR : message d'erreur

Ce sous programme assure la sortie d'un message en cas de retour en erreur d'une fonction du gestionnaire de fichier FMS. Les messages d'erreur système sont stockés dans le fichier disque ERRØRS.SYS. mais le programmeur peut utiliser son propre fichier de messages. En entrée, le registre X doit contenir l'adresse d'un FCB dont le code d'erreur est non nul. Ce code est tranféré en $CC20, un appel au sous programme RSTRIØ est effectué et l'aiguillage fichier d'erreurs est testé ($CC2D - $CC2E). Si l'aiguillage est nul, le fichier ERRØR.SYS est ouvert en accès direct pour la recherche du message, sinon c'est l'adresse d'une chaine ASCII donnant les spécifications du fichier à utiliser (non + extension sur 11 caractères). Le fichier utilisateur est alors ouvert en accès direct. Le code d'erreur est utilisé pour calculer le numéro de l'enregistrement dans lequel se trouve le message et son emplacement dans l'enregistrement. Sachant que chaque message

est rangé dans une zone de 63 caractères, à raison de quatres zones par enregistrement (un enregistrement correspondant à un secteur disque). Si le message est trouvé il est sorti sur l'écran, sinon (le code d'erreur n'étant pas répertorié ou le fichier n'ayant pas été trouvé) le code d'erreur nnn est affiché sur l'écran avec le message :

ERR.DISQUE ## nnn

La description des codes d'erreur système est donnée dans la suite du manuel.

2.4.17. CONVERSION D'UN NOMBRE HEXADECIMAL

$CD42 GETHEX : entrée d'un nombre hexadécimal

Ce sous programme permet de convertir en binaire un nombre hexadécimal entré dans le tampon de ligne. En entrée, le pointeur tampon de ligne doit désigner le premier caractère du nombre ; en sortie la retenue est à zéro si le sous programme a trouvé un nombre valide, le registre B est non nul, et le registre X contient le nombre en binaire. Le pointeur tampon de ligne est laissé sur le premier caractère suivant le séparateur sauf si ce séparateur est un retour chariot ou une fin de ligne.

La valeur retournée dans le registre X est nulle si le premier caractère examiné dans le tampon est un séparateur (tel qu'une virgule), la retenue est mise à zéro ainsi que le registre B pour indiquer que le nombre n'a pas été trouvé (ou bien qu'il a été omis intentionnellement, sachant qu'il prendra la valeur 0 par défaut). Si un caractère non hexadécimal est trouvé, les suivants sont sautés jusqu'au prochain séparateur, et la retenue est mise à 1. La longueur du nombre dans le tampon n'est pas limitée mais sa valeur binaire est tronquée à $FFFF inclue.

2.4.18. SORTIE D'UNE ADRESSE EN HEXADECIMAL

$CD45 ØUTADR : sortie d'une adresse hexadécimale

En entrée, le registre X contient l'adresse de l'octet de poids fort d'une adresse binaire codée sur 2 octets. Cette valeur est sortie codée en hexadécimal avec quatre chiffres.

2.4.19. CONVERSION D'UN NOMBRE DECIMAL

$CD48 INDEC : entrée d'un nombre décimal

Ce sous programme permet de convertir en binaire un nombre décimal non signé entré dans le tampon ligne.

En entrée, le pointeur du tampon de ligne doit désigner le premier caractère du nombre. En sortie, la retenue est mise à zéro si un nombre valide a été trouvé, le registre B est non nul et le registre X contient la valeur binaire de ce nombre. Le pointeur du tampon est laissé sur le premier caractère suivant le séparateur, sauf si ce séparateur est un retour chariot ou une fin de ligne. Si le premier caractère examiné dans le tampon est un séparateur (une virgule, par exemple), la retenue est mise à zéro, ainsi que le registre B pour indiquer l'absence du nombre, et la valeur retournée dans X est nulle. La longueur du nombre dans le tampon n'est pas limité mais le résultat est tronqué à la précision des 16 bits.

2.4.20. UTILISATION DU DØS COMME SOUS PROGRAMME

$CD4B DØCMND : appel du DØS

Ce point d'entrée dans le moniteur permet à un programme de faire exécuter une commande par le moniteur et de récupérer le contrôle à la fin de la commande (qui peut être multiple). La commande doit être placée dans le tampon de ligne par le programme, et le pointeur doit être initialisé sur le premier caractère de la commande. Il faut noter que cette opération détruit le contenu du tampon et que l'on doit veiller à ne pas rompre l'enchainement de commandes multiples. La commande doit être terminée par un retour chariot ($D). Le DØS rend le contrôle au programme utilisateur après l'exécution de la commande avec le registre B à zéro si aucune erreur n'a été détectée, avec le code d'erreur dans B en cas d'erreur (du FMS s'il est utilisé).

Attention ! ne pas utiliser cette procédure pour charger un programme qui pourrait recouvrir le programme appelant en mémoire et le détruire.

Un exemple d'utilisation pourrait être la sauvegarde sur disque d'une partie de la mémoire dans un fichier binaire, en construisant une commande SAVE dans le tampon avec le nom du fichier et les paramètres et en appelant le moniteur par ce point d'entrée.

3 - ECRITURE D'UTILITAIRES PAR L'UTILISATEUR

La grande souplesse de FLEX, est que l'utilisateur peut écrire ses propres commandes utilitaires. Elles peuvent être stockées sur disque ou résidentes en mémoire vive (RAM) ou morte (RØM-EPRØM).

3.1 - Commandes résidentes en mémoire

De telles commandes sont des programmes, toujours résidents en mémoire, auquels le DØS fait appel lorsque le nom associé à la commande est entré au clavier.

Les noms des commandes et les points d'entrée (adresse d'exécution) des programmes associés sont stockés dans une table créée par l'utilisateur en mémoire. Le format de cette table, est défini ci-après :

```
FCC "nom de la commande"
FCB 0
FDB "adresse d'exécution de la commande".
```

La table des commandes utilisateur doit toujours se terminer par un octet nul.

Par exemple :

```
FCC "DEBUG"        1ère commande utilisateur
FCB 0
FDB $3000          adresse d'exécution du pgm DEBUG
FCC "PUNT"         2ème commande utilisateur
FCB 0
FDB $3200          adresse d'exécution du pgm PUNT
FCB 0              fin de la table.
```

L'adresse de la table des commandes utilisateur doit être fournie au DØS en $CC12-$CC13 (conférer carte mémoire).

Lorsqu'une commande est entrée au clavier, le DØS explore successivement sa propre table des commandes, la table des commandes utilisateur (si elle existe) puis le catalogue du disque.

La table du DØS contient uniquement les commandes GET et MØN. Comme elle est explorée avant celle de l'utilisateur, celui-ci ne pourra pas créer des commandes répondant aux noms GET et MØN. Par contre, il pourra créer des commandes aux noms identiques à ceux des commandes répertoriées dans le catalogue du disque, tout en sachant que ces dernières ne pourront plus être exécutées tant que l'accès à la table des Commandes Utilisateur n'est pas inhibé, en remettant à zéro le pointeur $CC12-$CC13.

3.2 - Commandes résidentes sur disque

Les commandes utilisateurs sauvegardées sur disque doivent obligatoirement posséder une adresse de transfert et une extension CMD.

En général, ces commandes sont créées en assembleur, et la manière d'affecter une adresse de transfert à un programme en cours d'assemblage, est décrite dans le manuel de présentation de l'assembleur (ASMB).

Une commande utilisateur peut être exécutée en n'importe quel endroit de la mémoire RAM utilisateur. Son adresse de chargement est précisée à l'assemblage par la directive ØRG .

Mais dans la plupart des cas, il est utilé de pouvoir les charger dans l'espace des commandes réservé par le FLEX ($C100-$C6FF).

Pratiquement toutes les commandes fournies avec FLEX s'exécutent dans cette zone mémoire. La commande SAVE ne fait pas exception à la règle, aussi il n'est pas possible de l'utiliser pour sauvegarder des informations contenues dans cette zone. C'est pour cette raison que la commande SAVE.LØW a été créée. Elle est identique à la commande SAVE mais s'exécute en $0100.

La table FCB est utilisée pour le chargement des commandes disque, les commandes écrites pour s'exécuter dans l'espace des commandes utilitaires ne doivent donc pas empiéter dessus.

Par contre, une fois chargées, elles peuvent l'utiliser comme zone tampon ou FCB pour leurs propres accès disque.

Pour plus de détails se référer aux exemples donnés dans le chapitre descriptif du FMS.

3.3 - Remarques d'ordre général

Les commandes écrites par l'utilisateur sont appelées par l'instruction de branchement JMP. En retour, elles doivent redonner la main au DØS en se branchant (instruction JMP) au point d'entrée à chaud du FLEX $CD03.

3.3.1. - Acquisition des arguments

Les commandes écrites par l'utilisateur peuvent être amenées à utiliser des arguments entrés au clavier. Le nom de la commande et les arguments sont stockés dans le buffer de ligne ($C080 - $C0FF). A l'entrée de la commande, le pointeur du buffer de ligne adresse la position du 1er caractère du 1er argument s'il existe, sinon il désigne soit la position du caractère de fin de ligne de commande (:), soit la position du retour chariot (↵).

Les sous-programmes NXTCH, GETFIL et GETHEX peuvent être utilisés pour faire l'acquisition de ces arguments.

3.3.2. - Traitement des erreurs

Si en cours d'exécution une erreur est détectée par le DØS, le FMS ou par la commande elle-même, il faut prévoir un traitement de reprise ou l'abandon de l'exécution de la commande en redonnant le contrôle au FLEX par le point d'entrée à chaud ($CD03). Après avoir affiché un message approprié (sous programme RPTERR par exemple pour une erreur FMS).

3.4 - Exemple d'utilisation

1 - L'écriture du nom d'un fichier dans le FCB peut être réalisée de la manière suivante :

On suppose que le pointeur du buffer de ligne désigne le 1er caractère du nom du fichier et que son extension sera TXT par défaut.

```
LDX     # FCB      (X) pointe sur le FCB
JSR     GETFIL     écriture du nom du fichier dans le FCB
BCS     ERRØR      traitement d'erreur
LDAA    # 1        code de l'extension TXT
JSR     SETEXT     écriture de l'extension par défaut
```

Il ne reste plus qu'à ouvrir le fichier afin de procéder à l'opération souhaitée (voir les exemples du FMS pour l'ouverture d'un fichier).

2 - Les 2 exemples suivants illustrent la simplicité de l'utilisation des sous-programmes d'E/S fournis par le DØS.

21)
```
LDAA    # 'A      caractère ASCII A
JSR     PUTCHR   sortie du caractère
```

22)
```
LDX     # STRING  (x) pointe sur une chaîne de caractères
JSR     PSTRNG   affiche Retour-chariot + Saut ligne + Chaîne
```

4 - LE SYSTEME DE GESTION DES FICHIERS (FMS)

Le système de gestion de fichier (FMS) assure l'interface entre le DØS (système d'exploitation disque) et la mémoire de masse que constituent les disques.

Le FMS gère la création et la destruction de tous les fichiers sur le disque. Tout l'espace est alloué dynamiquement. La zone du disque utilisée par un fichier peut-être réutilisée aussitôt après la destruction de ce fichier.

La communication avec le FMS se fait par l'intermédiaire des FCB (Blocs de contrôle du fichier). Ces blocs contiennent les caractéristiques du fichier comme son nom ou le numéro du disque sur lequel il existe.

La lecture et l'écriture des données d'un fichier (E/S disque) seront faites caractère par caractère, le disque n'étant considéré que comme un terminal du micro-ordinateur.

Les fichiers peuvent être ouverts en lecture ou en écriture. Plusieurs fichiers peuvent être ouverts en même temps du moment qu'ils ont chacun leur FCB.

Le FMS est un langage de commande dans lequel chaque commande est représentée par un nombre appelé : Code Fonction. A chaque Code Fonction correspond une tâche bien déterminée, comme l'ouverture en lecture ou la destruction d'un fichier.

En général, l'utilisation du FMS est relativement simple. Il suffit de pointer le registre d'index sur l'adresse du FCB, de ranger le Code Fonction choisi dans le 1er octet du FCB puis d'appeler le FMS par un JSR ("jump to subroutine").

A aucun moment l'utilisateur n'a à se préoccuper de la place du fichier sur le disque, de sa longueur ou de sa référence dans le catalogue. Le FMS traite ces problèmes automatiquement.

Du fait de la gestion dynamique de l'espace disponible sur le disque par une structure de liste linéaire chaînée, il est possible que des fichiers occupent sur le disque d'une suite des secteurs non contigus.

Normalement, quand une disquette vient d'être formattée, les fichiers qui y seront rangés occuperont des secteurs contigus.
Au fur et à mesure des créations et destructions de fichiers, toutefois, le disque peut se "fragmenter" ; c'est-à-dire que les secteurs occupés par un fichier ne se suivent plus physiquement sur le disque : pour le logiciel, ils restent évidemment contigus.

Ceci est une caractéristique des structures de listes linéaires chainées et des méthodes d'allocation dynamique de l'espace disponible.

L'utilisateur du FLEX n'a pas à se préoccuper de cette fragmentation possible des fichiers ; il doit cependant garder à l'esprit que les fichiers créés peuvent être constitués de secteurs disséminés sur le disque.

La seule conséquence de la "fragmentation" est l'augmentation de la durée de lecture d'un fichier, dûe à un nombre de déplacements de la tête de lecture plus important.

4.1 - Le bloc de contrôle du fichier

Le bloc de contrôle de fichier (FCB) est le coeur du FMS. Un FCB est un bloc de 320 octets de RAM, dans l'espace mémoire utilisateur, qui est utilisé par les programmes pour communiquer avec le FMS.

A chaque fichier ouvert correspond un FCB particulier. Après la fermeture du fichier, le FCB peut être réutilisé pour ouvrir un autre fichier ou pour exécuter d'autres fonctions comme la destruction (DELETE) ou le changement de nom (RENAME) d'un fichier.

Un FCB peut être localisé n'importe où dans la mémoire utilisateur sauf dans la page 0.

La zone mémoire réservée au FCB n'a pas besoin d'être initialisée de quelque manière que ce soit. Seuls les paramètres utilisés pour l'exécution de la fonction demandée devront être chargés dans le FCB.

Le FMS initialisera lui-même les zones du FCB dont il pourrait avoir besoin.

Dans la description qui suit, le numéro des octets sont désignés par rapport au début du FCB i.e : Octet 0 est le premier octet du FCB.

DESCRIPTION D'UN BLOC DE CONTROLE

- Octet 0 Code fonction

Le code de la fonction choisie doit être stocké dans cet octet, par l'utilisateur, avant d'appeler le FMS. Voir le chapitre de descriptions des codes-fonction.

- Octet 1 Code Erreur

Si une erreur est détectée pendant l'exécution de la fonction, le FMS charge un code erreur dans cet octet et rend la main à l'utilisateur après avoir mis à zéro le bit z (condition zéro) du registre condition du CPU i.e. une condition non-nul existe.

Ceci peut être testé par une instruction BEQ ou BNE.

- Octet 2 Code d'activité

Cet octet est mis à "1" par le FMS si le fichier est ouvert pour lecture et mis à "2" s'il est ouvert pour écriture. Cet octet est testé par plusieurs modules du FMS pour déterminer la légalité de l'opération demandée. Un code erreur est donné en cas d'opération illégale.

Les douze octets suivants (3-14), renferment la "spécification de fichier" qui consiste en un numéro de lecteur, nom de fichier et extension de fichier. Certaines des fonctions du FMS ne requièrent pas le nom du fichier ni son extension.

- Octet 3 Numéro de lecteur

C'est le numéro du lecteur contenant la disquette sur laquelle le fichier se trouve. Ce numéro peut être 0, 1, 2 ou 3.

Les 24 octets suivants (4-27) contiennent les informations nécessaires à la dénomination du fichier. Ce sont les mêmes informations que celles contenues dans l'inventaire du disque.

- Octets 4 à 11 Nom du fichier

C'est le nom du fichier choisi. Ce nom doit obligatoirement commencer par une lettre et doit contenir uniquement des lettres, des chiffres, des traits d'union (-) et/ou des soulignés (_).

Si le nom du fichier contient moins de 8 caractères, les octets restants seront à zéro. Le nom du fichier doit être calé à gauche dans le champ qui lui est assigné.

- Octets 12-14 Extension du fichier

C'est l'extension du fichier choisi. Elle doit commencer par une lettre et ne doit contenir que des lettres, des chiffres, des traits d'union (-) et/ou des soulignés (_).

Si la longueur de l'extension est inférieure à 3 caractères, les octets restants seront mis à zéro.

L'extension doit être calée à gauche dans le champ qui lui est assigné. Les fichiers sans extension ne pourront être créés.

- Octet 15 Attributs du fichier

A présent, seuls les 4 bits les plus significatifs sont définis. Ces bits sont utilisés pour définir les types de protection nécessaires. Ils sont définis comme :

bit 7 : protection en écriture
bit 6 : protection contre la destruction
bit 5 : protection en lecture
bit 4 : protection de catalogue

Mettre à "1" l'un de ces bits, met en place la protection qui lui est associée.

Tous les bits sans fonction doivent être à zéro !

- Octet 16 Réservé pour une utilisation future

- Octet 17-18 Adresse de début du fichier sur le disque

Ces deux octets contiennent respectivement le numéro de la piste et du secteur, du premier secteur du fichier.

- Octet 19-20 Adresse de fin du fichier sur le disque

Ces deux octets contiennent respectivement le numéro de la piste et du secteur, du dernier secteur du fichier.

- Octets 21-22 Taille du fichier

C'est un nombre sur 16 bits qui indique le nombre de secteurs occupés par le fichier.

- Octet 23 Indicateur d'une table des secteurs du fichier

Si cet octet est non nul (habituellement $02), le fichier est un fichier à accès direct et contient une table des secteurs du fichier. Voir la description des fichiers à accès direct pour plus de détails.

- Octet 24 Réservé pour une utilisation future

- Octet 25-27 Date de création du fichier

Ces 3 octets contiennent la date, en binaire, de la création du fichier. Le premier octet est le mois, le second le jour, et le troisième l'année (uniquement les dizaines et les unités).

- Octet 28-29 Pointeur de liste des FCB

Tous les FCB qui sont ouverts pour une lecture ou une écriture sont chainés ensemble. Ces deux octets contiennent l'adresse mémoire du pointeur de liste des FCB du bloc suivant dans la chaine. Le 1er FCB dans la chaine est adressé par le pointeur de base des FCB (cf. les variables globales). Ces deux octets sont à zéro si ce FCB est le dernier de la chaine.

- Octets 30-31 Position courante

Ces octets contiennent respectivement les numéros de la piste et du secteur, du secteur alors contenu dans la zone tampon du FCB. Si le fichier est à écrire, le secteur adressé par ces octets n'a pas encore été écrit sur la disquette ; il est encore dans le buffer.

- Octets 32-33 Numéro d'enregistrement courant

Ces octets contiennent le numéro logique de l'enregistrement du secteur contenu dans le FCB.

- Octet 34 Indice des données

Cet octet contient l'adresse du prochain octet de données à aller chercher (pour lecture) ou à ranger (pour écriture) dans le tampon. Cette adresse est relative au début du tampon; elle est automatiquement avancée par la fonction "Lecture/Ecriture du Prochain Octet". Le programme utilisateur n'a pas besoin d'utiliser cet octet.

- Octet 35 Indice d'accès direct

Cet octet est utilisé conjointement avec la fonction d'accès direct à un octet à l'intérieur d'un secteur, pour lire un octet dans le tampon sans avoir à lire tous les autres octets le précédant.

L'adresse de l'octet choisi, calculée relativement au début du secteur, est rangée par l'utilisateur dans l'octet d'indice et la fonction d'Accès Direct à un octet est lancée par le FMS. L'octet de données désigné est retourné dans l'accumulateur A.

Si la valeur rangée dans l'octet d'indice d'accès direct est inférieure à 4, l'utilisateur a accès à l'un des octets de chainage du secteur. Les données proprement dites commencent à une adresse relative supérieure à 4.

- Octets 36-46 Tampon

Ces octets sont utilisés par le FMS pour ranger temporairement le nom d'un fichier. Ces positions ne sont pas utilisées par un programme utilisateur.

- Octets 47-49 Adresse courante du catalogue

Si le FCB est utilisé pour traiter des informations du catalogue. à l'aide de la fonction "Acquisition/Sauvegarde des informations d'un enregistrement", ces trois octets contiennent les numéros de la piste, du secteur et l'indice des données du catalogue contenues dans la zone dénominative du fichier (octets 4 à 27). Les valeurs de ces trois octets sont remis à jour par la fonction "Acquisition des informations d'un enregistrement".

- Octets 50-52 Pointeur de recherche du 1er fichier effacé dans le catalogue

Ces octets sont utilisés par le FMS lorsqu'il cherche une entrée libre dans le catalogue pour y ranger le nom d'un nouveau fichier.

- Octets 53-63 Zone de travail

Ces octets sont utilisés pour ranger le nouveau nom et l'extension d'un fichier dont on change le nom. Ce nouveau nom répond aux mêmes caractéristiques que celles décrites auparavant (octets 4 à 14).

- Octet 59 Indicateur de compression de l'espace

Si un fichier est ouvert pour lecture ou pour écriture, cet octet indique si une compression d'espace a été faite.

S'il est à zéro, la compression sera faite lors de la lecture ou de l'écriture des données. Cette valeur y est rangée par les opérations "Ouverture pour Lecture" et "Ouverture pour Ecriture" d'un fichier.

S'il est à $FF, il n'y a pas de compression à faire. Cette valeur doit être rangée par l'utilisateur, après l'ouverture d'un fichier, s'il ne désire pas une compression d'espace (par exemple pour un fichier binaire).

Une valeur positive non nulle indique que la compression est en cours ; cette valeur représente le nombre d'espaces traités jusque là. (Notons que bien que cet octet soit inclus dans la zone de travail décrite ci-dessus, cela n'entraîne aucun conflit dans la mesure où la compression ne peut se faire que si le fichier est ouvert et que la zone de travail n'est utilisée que pour un changement de nom (RENAME), qui ne se fait que fichier fermé).

En règle générale cet octet sera à 0 pour tout travail sur des fichiers TXT et à $FF pour tout travail sur des fichiers binaires.

- Octets 63-319 Tampon secteur

Ces octets contiennent les données du secteur lu ou écrit. Les 4 premiers octets du secteur sont utilisés par le système, les 252 autres sont utilisés pour y ranger les données.

4.1 bis - Points d'entrée du système de gestion de fichier

1 - Initialisation du système : $D400

Ce point d'entrée est utilisé par le système d'exploitation disque pour initialiser le FMS après un départ à froid. L'utilisateur ne devrait pas avoir à utiliser ce point d'entrée. Une initialisation au mauvais moment peut détruire les fichiers de données, nécessitant alors une ré-initialisation des disquettes.

2 - Fermeture du système : $D403

Ce point d'entrée est utilisé par le système d'exploitation disque à la fin de chaque ligne de commande pour fermer tout fichier laissé ouvert par le processeur. Les programmes-utilisateur peuvent aussi utiliser ce point d'entrée pour fermer tous les fichiers ouverts ; toutefois, si une erreur est détectée en essayant de fermer un fichier, tous les fichiers restants resteront dans leur état. Le programmeur est ainsi prévenu contre l'usage de cette routine au lieu de la bonne méthode qui consiste à fermer les fichiers individuellement. Il n'y a pas d'arguments dans cette routine. Elle est entrée par une instruction JSR comme une sous-routine. En sortie, le bit Z du registre de conditions du CPU est à 0 si une erreur est détectée, à 1 sinon ; dans le cas d'une erreur, le registre d'index contient l'adresse du FCB qui cause l'erreur.

3 - Appel du système : $D406

Ce point d'entrée est utilisé pour tous les autres appels au système de gestion des fichiers.

Un code fonction est rangé dans l'octet n° 0 du FCB, l'adresse du FCB est mis dans le registre d'index du CPU, et ce point d'entrée est appelé par un JSR.

Les différentes fonctions sont décrites ci-dessous.

En sortie, le bit Z du registre de conditions du CPU est mis à "1" si aucune erreur n'a été détectée au cours de l'exécution de la fonction choisie. Ce bit pourra être testé par une instruction BEQ ou BNE. Si une erreur est détectée, le bit Z est mis à zéro et l'octet "code erreur" du FCB est chargéavec le numéro de l'erreur détectée.

Dans tous les cas, le registre d'index du CPU contient l'adresse du FCB.

Quelques fonctions ont besoin de certains paramètres, chargés dans les accumulateurs A et/ou B du CPU.

Pour plus de détails, se référer à la description des différentes fonctions possibles.

Dans tous les cas, le registre d'index X et le registre B ne sont pas modifiés par un appel au FMS.

4.2 - Variables globales

Ce paragraphe décrit les différentes variables, internes au FMS, susceptibles d'intéresser l'utilisateur. Toutes les autres positions mémoire dans la zone du FMS ne doivent pas être utilisées par le programme utilisateur pour y ranger des données.

- $D409 $D40A Pointeur de base du FCB

Ces deux octets contiennent l'adresse du pointeur de la liste des FCB dans le premier bloc de la chaîne des fichiers ouverts.

Le contenu de ces octets est géré par le FMS, et le programmeur ne doit y ranger aucune valeur.

Si pour une raison ou une autre, l'utilisateur désire chaîner les blocs de contrôle des fichiers à ouvrir, il doit se souvenir qu'à cette adresse, se trouve le point de départ de la chaîne.

Attention ! il ne faut pas oublier que cette adresse désigne les emplacements dans le FCB du pointeur de liste FCB et non pas le premier mot du FCB.

Si ces deux octets sont nuls, cela signifie qu'il n'y a pas de fichiers à ouvrir.

- $D40B $D40C Adresse du bloc de contrôle courant

Ces deux octets contiennent l'adresse du dernier bloc de contrôle traité par le FMS. Cette adresse est celle du premier mot du FCB.

- $D435 Indicateur de vérification

Si cet octet est non **nul**, le FMS testera chaque secteur juste après l'écriture afin de détecter d'éventuelles erreurs. Si cet octet est nul, aucun test de vérification à l'écriture ne sera exécuté.

La valeur donnée par défaut à cet octet est "nul".

4.3 - Codes fonction du FMS

L'utilisation du système de gestion des fichiers du FLEX passe par l'utilisation d'un certain nombre de codes fonction.

Le code fonction choisi, est placé par l'utilisateur dans l'octet de code-fonction (octet 0) du bloc de contrôle du fichier (FCB) avant l'appel du FMS.

Le FMS sera appelé par un JSR au point d'appel du FMS ($D406). Le registre d'index du CPU devra contenir l'adresse du bloc de contrôle.

En sortie du FMS, le bit Z du registre de conditionsdu CPU sera mis à zéro si une erreur a été détectée en cours d'exécution de la fonction choisie. Ce bit pourra être testé par les instructions BNE et BEQ.

N.B : Dans tous les exemples qui suivent, la ligne "JSR FMS" fait référence au point d'appel du FMS : $D406.

- Fonction 0 : Lecture/écriture du prochain octet/caractère

Si le fichier est ouvert pour lecture, le prochain octet est lu dans le fichier et transmis au programme appelant dans l'accumulateur A.

Si le fichier est ouvert pour écriture, le contenu de l'accumulateur A est placé dans le buffer afin d'être écrit dans le prochain octet du fichier.

L'indicateur de compression d'espace doit contenir les valeurs convenables pour réaliserladite compression, si nécessaire (voir dans la description d'un FCB, le paragraphe traitant de l'indicateur de compression d'espace).

En sortie, l'octet de code fonction demeurera inchangé.

Ainsi des lectures/écritures consécutives pourront être exécutées sans avoir à recharger le code fonction.

En lecture, une erreur "Fin de Fichier" est envoyée quand toutes les données du fichier ont été lues.

Quand le secteur courant à lire est vide, le prochain dans le fichier est automatiquement préparé pour l'exécution de la fonction, sans aucune intervention de l'utilisateur.

De même, en écriture, des secteurs pleins sont automatiquement écrits sur le disque sans intervention de la part de l'utilisateur.

Exemple :

En lecture :

```
LDX   ## FCB    Pointe sur l'adresse du FCB
JSR   FMS       Appel du FMS
BNE   ERRØR     Test d'erreur
```
Le caractère lu est maintenant dans l'accu. A.

En écriture :

```
LDAA  CHAR      Chargement du caractère
LDX   ## FCB    Pointe sur l'adresse du FCB
JSR   FMS       Appel du FMS
BNE   ERRØR     Test d'erreur
```
Le caractère dans A a été écrit.

Fonction 1 Ouverture pour lecture

Le fichier désigné dans le FCB est ouvert pour lecture seulement. Si le fichier ne peut être trouvé, un code erreur est renvoyé. Les zones du FCB qui doivent être initialisées avant l'exécution de la fonction sont le numéro du lecteur, le nom du fichier, l'extension du fichier et le code fonction. Les autres zones du FCB seront initialisées par l'exécution de l'ouverture du fichier.

L'ouverture du fichier met l'indicateur de compression d'espace à zéro, désignant par là un fichier source (TXT). Si le fichier est un fichier binaire, le programmeur devra positionner l'indicateur de compression à $FF après avoir ouvert le fichier, pour ne pas valider le mode de compression de place.

En sortie du FMS, après l'ouverture d'un fichier, le code fonction dans le FCB du fichier est mis automatiquement à zéro (Lecture/Ecriture du prochain caractère) en anticipation d'une E/S sur le fichier.

Exemple :

```
LDX   ## FCB    Pointe sur l'adresse du FCB
(initialisation des attributs du fichier dans le FCB)
LDAA  ## 1      Code Fonction Ouverture en Lecture
STAA  0,X       Sauvegarde dans le FCB
JSR   FMS       Appel du FMS
BNE   ERRØR     Test d'erreur
```
Le fichier est maintenant ouvert pour lecture.
Dans le cas du fichier binaire, il faut continuer avec :
```
LDAA  $FF       Suppression compression de place
STAA  59,X      sauvegarde dans le FCB
```

Fonction 2 Ouverture pour écriture

C'est le même type de fonction que la fonction 1 sauf que le fichier ne doit pas déjà exister sur la disquette, et qu'il est ouvert en écriture seulement.

Un fichier ouvert en écriture ne peut être lu tant qu'il n'a pas été fermé puis rouvert en lecture.

L'indicateur de compression de place pourra être manipulé de la même manière qu'il est décrit plus haut dans l'ouverture d'un fichier pour lecture.

Un fichier est normalement ouvert comme un fichier à accès séquentiel mais peut l'être comme un fichier à accès direct en chargeant une valeur non nulle dans l'octet du FCB indicateur d'une table des secteurs, juste après l'ouverture du fichier pour écriture.

Pour plus de détail, se référer aux chapitre traitant des fichiers à accès direct.

Le fichier sera créé sur le disque désigné à moins que le numéro de lecteur soit $FF. Dans ce cas, le fichier sera créé sur le premier disque prêt (mode recherche automatique).

Exemple :

```
LDX  . ## FCB    pointe sur l'adresse du FCB
(initialisation des attributs du fichier dans le FCB)
LDAA  ## 2       Code fonction Ouverture en Ecriture
STAA  0,X        Sauvegarde dans le FCB
JSR   FMS        Appel du FMS
BNE   ERR$R      Test d'erreur
Le fichier est maintenant ouvert pour écriture.
```

Dans le cas d'un fichier binaire, il faut continuer avec :

```
LDAA  $FF        Suppression compression de place
STAA  59,X
```

- Fonction 3 : Ouverture pour mise à jour

Cette fonction permet d'ouvrir un fichier aussi bien pour lecture qu'écriture. Le fichier ne doit pas être déjà ouvert et doit exister sur le disque spécifié. Si la spécification de lecteur est $FF, tous les lecteurs seront recherchés.

Une fois que le fichier est ouvert pour être remis à jour, quatre opérations peuvent être exécutées :

1 - Lecture en accès séquentiel
2 - Lecture en accès direct
3 - Ecriture en accès direct
4 - Fermeture du fichier

Il n'est pas possible d'écrire séquentiellement sur un fichier ouvert pour une remise à jour. Cela implique, qu'il n'est pas possible d'augmenter la taille d'un fichier à l'aide de la fonction 3.

Fonction 4 : Fermeture d'un fichier

Si le fichier est ouvert pour lecture, sa fermeture supprime simplement son bloc de contrôle de la chaine des FCB. Si le fichier est ouvert pour écriture, les données restantes dans le tampon secteur sont écrites sur le disque, en y ajoutant des 0 si besoin est pour remplir le secteur.

Si un fichier ouvert pour écriture n'est jamais écrit, son nom ne sera pas inscrit dans le catalogue du disque puisqu'il ne contient pas de données.

Exemple :

```
LDX   ## FCB    Pointe sur l'adresse du FCB
LDAA  ## 4      Code fermeture
STAA  0,X       Sauvegarde dans le FCB
JSR   FMS       Appel du FMS
BNE   ERRØR     Test d'erreur
Le fichier est maintenant fermé.
```

Fonction 5 : Repositionnement au début d'un fichier

Seuls les fichiers ouverts pour lecture peuvent être repositionnés. En sortie du FMS, le code fonction dans le FCB est mis à Ø, anticipant un ordre de lecture du fichier. Si le programmeur désire repositionner au début un fichier ouvert pour écriture, de telle sorte qu'on puisse maintenant le lire, il doit d'abord le fermer puis le rouvrir pour lecture.

Exemple :

```
Le fichier est déjà ouvert pour lecture
LDX   ## FCB    Pointe sur l'adresse du FCB
LDAA  ## 5      Code de repositionnement
STAA  0,X       sauvegarde dans le FCB
JSR   FMS       Appel du FMS
BNE   ERRØR     Test d'erreur
Le fichier est maintenant repositionné au début et prêt en lecture.
```

Fonction 6 : Ouverture du catalogue

Cette fonction permet l'ouverture du catalogue afin qu'un programme puisse y accèder. Le FCB utilisé pour réaliser cette fonction ne doit pas l'être déjà pour l'utilisation d'un fichier.

En entrée, la seule information à stocker dans le FCB est le numéro du lecteur. Aucun nom de fichier n'est nécessaire. Les informations dans le catalogue sont lues en utilisant la fonction "acquisition des données d'un enregistrement".

La fonction "écriture des données d'un enregistrement" est utilisée pour écrire dans le catalogue.

La fonction "Lecture/Ecriture du prochain caractère" ne fonctionne pas sur un FCB ouvert pour accéder au catalogue.

Il n'est pas nécessaire de fermer un FCB ouvert par la fonction 6 après avoir terminé la manipulation sur le catalogue. Normalement, l'utilisateur ne doit pas avoir besoin d'utiliser cette fonction.

Fonction 7 : Acquisition des données d'un enregistrement

Cette fonction doit être utilisée, uniquement, avec un FCB ouvert pour "accès catalogue" (fonction 6). Chaque fois qu'elle est terminée, le prochain enregistrement dans le catalogue est chargé dans la zone des "informations du catalogue" du FCB (octets 4 à 27). Tous les enregistrements du catalogue, y compris ceux qui ne sont pas utilisés, ou ceux qui sont effacés, sont lus lorsque cette fonction est utilisée.

Après la lecture d'un enregistrement, le FCB "pointe" sur l'enregistrement qui vient d'être lu ;

L'adresse courante du catalogue dans le FCB (octets 47-49) est celle de l'enregistrement qui vient d'être lu.

Une erreur "Fin de Fichier" est retournée lorsque la fin du catalogue est atteinte.

Exemple :

Acquisition du 3ème enregistrement dans le catalogue.

```
        LDX   #FCB     pointe sur l'adresse du FCB
        LDAA  DISQUE   (DISQUE) = n° du lecteur choisi
        STAA  3,X      Sauvegarde dans le FCB
        LDAA  #6       Code ouverture du catalogue
        STAA  0,X      Sauvegarde dans le FCB
        JSR   FMS      Appel du FMS
        BNE   ERRØR    Test d'erreur
        LDAB  #3       Init. compteur à 3
LØØP    LDAA  #7       Code acquisition d'un enregistrement
        STAA  0,X      Sauvegarde dans le FCB
        JSR   FMS      Appel du FMS
        BNE   ERRØR    Test d'erreur
        DECB           Décrémente compteur
        BNE   LØØP     répété jusqu'à la fin
```

Le 3ème enregistrement du catalogue est maintenant dans le FCB.

Fonction 8 : Ecriture des données d'un enregistrement

Cette fonction doit être utilisée uniquement avec un FCB ouvert pour "accès catalogue". L'enregistrement à écrire dans le catalogue est recopié de la "zone des informations du catalogue" du FCB dans le catalogue, à l'endroit désigné par le FCB. Le secteur du catalogue venant d'être mis à jour est alors automatiquement récrit sur la disquette, pour être certain que le catalogue a bien été remis à jour.

Un programme utilisateur, ne doit, normalement, pas avoir à écrire dans le catalogue.

Il faut utiliser, cette fonction avec précaution. Une erreur de manipulation peut détruire les fichiers sur la disquette.

Fonction 9 : Lecture d'un secteur

Cette fonction permet la lecture d'un secteur sur le disque. Ce secteur est adressé par les octets 30-31 du FCB (position courante). Les données lues sont chargées dans le tampon secteur du FCB.

Cette fonction est normalement exécutée par le FLEX et l'utilisateur ne devrait pas avoir besoin de la programmer. Il est préférable d'utiliser la fonction "Lecture/Ecriture du prochain caractère" chaque fois que possible.

Cette fonction doit être manipulée avec prudence, car sa procédure ne ressemble pas à celle des autres fonctions du FLEX.

Exemple :

```
LDX   #FCB    pointe sur l'adresse du FCB
LDAA  PISTE   (PISTE) = n° de la piste choisie
STAA  30,X    init. FCB position courante
LDAA  SECT    (SECT) = n° du secteur choisi
STAA  31,X    init. FCB position courante
LDAA  #9      Code Lecture d'un secteur
STAA  0,X     Sauvegarde dans le FCB
JSR   FMS     Appel du FMS
BNE   ERRØR   Test d'erreur
```

Le contenu du secteur est maintenant dans le FCB.

Fonction 10 : Ecriture d'un secteur ($0A hex.)

Cette fonction permet d'écrire directement un secteur sur le disque. Il faut faire très attention en l'utilisant. Cette fonction est normalement exécutée par FLEX et l'utilisateur ne devrait pas avoir à la programmer. Il est préférable d'utiliser la fonction "Lecture/Ecriture du prochain caractère" chaque fois que possible.

Une erreur dans l'utilisation de cette fonction peut provoquer la destruction de la disquette nécessitant sa ré-initialisation.

L'adresse du secteur à écrire doit être contenue dans les octets 30-31 du FCB (position courante) et les données à écrire doivent être présentes au tampon secteur du FCB. Cette fonction teste l'indicateur de vérification (voir le chapitre sur les variables globales) et vérifiera le contenu du secteur sur le disque après l'écriture si le mode de vérification est programmé.

Fonction 11 : Réservé pour une utilisation future ($0B hex.)

Fonction 12 : Destruction d'un fichier ($0C hex.)

Cette fonction permet d'effacer le fichier dont le nom est contenu dans le FCB (n° du lecteur, nom du fichier, extension). Les secteurs utilisés par ce fichier peuvent être immédiatement réutilisés.

Le fichier ne doit pas être ouvert lorsque cette fonction est demandée.

Le nom du fichier dans le FCB est altéré pendant l'exécution de cette fonction.

Exemple :

```
LDX   # FCB    pointe sur l'adresse du FCB
(mise en place du nom du fichier dans le FCB)
LDAA  # 12     code destruction
STAA  0,X      Sauvegarde dans le FCB
JSR   FMS      Appel du FMS
BNE   ERRØR    Test d'erreur
Le fichier est maintenant effacé.
```

Fonction 13 : Changement du nom d'un fichier ($0D hex.)

En entrée, le fichier ne doit pas être ouvert. L'ancien nom doit être dans la zone d'identification du fichier du FCB (octets 4 à 27) et le nouveau nom dans la zone de travail du FCB (octets 53 à 63).

Le fichier dont le nom est contenu dans la zone d'identification du FCB porte maintenant le nom contenu dans la zone de travail.

Le nouveau nom et la nouvelle extension du fichier doivent être obligatoirement spécifiés.

Exemple :

```
LDX   # FCB    pointe sur l'adresse du FCB
(mise en place du nom du fichier dans la zone d'identification)

LDAA  # 13     code ré-appelation
STAA  0,X      sauvegarde dans le FCB
JSR   FMS      appel du FMS
BNE   ERRØR    Test d'erreur
```

Le fichier est maintenant renommé.

Fonction 14 : Réservé pour une utilisation future ($0E hex.)

Fonction 15 : Accès séquentiel au prochain secteur ($0F hex.)

En entrée, le fichier peut être ouvert pour lecture ou pour écriture (pas pour une mise à jour).

Si le fichier est ouvert pour lecture, lors de l'exécution de cette fonction, tous les octets de données restants (i.e. pas encore lus) sont ignorés et le pointeur de données sera positionné sur le premier octet de données du secteur suivant dans le fichier. Si le fichier est ouvert pour écriture, au cours de l'exécution de cette fonction la partie non écrite du secteur courant sera initialisée à ZERO, puis le secteur entier sera écrit sur le disque. Le caractère suivant, à écrire dans ce fichier, sera placé dans la première position de la zone des données du secteur suivant. Il est à noter que tous les appels à cette fonction seront ignorés tant qu'il n'y aura pas un octet d'écrit ou de lu dans le secteur courant.

Fonction 16 : Ouverture de l'enregistrement des informations système ($10 hex.)

En entrée, seul le n° du lecteur doit être spécifié dans le FCB ; il n'y a pas de nom de fichier associé.

Le FCB choisi ne doit pas être déjà ouvert pour l'utilisation d'un fichier.

Cette fonction permet l'accès à l'enregistrement des informations système de la disquette désignée dans le FCB.

Il n'y a pas de fonctions différentes pour la lecture ou la modification de ce secteur. Toutes les manipulations sur les données contenues dans l'enregistrement des informations système, sont faites directement dans le tampon secteur du FCB. Il n'est pas nécessaire de fermer le FCB après exécution de cette fonction.

Cette fonction est utilisée de manière interne par le FLEX ; elle ne devrait normalement pas être appelée par un programme-utilisateur.

Une mauvaise utilisation peut avoir pour résultat, la destruction du contenu de la disquette.

Fonction 17 : Acquisition d'un octet du secteur par accès direct ($11 hex.)

En entrée, le fichier doit être ouvert pour lecture ou pour mise à jour. Le numéro de l'octet désiré doit être rangé dans l'octet "indice d'accès direct" du FCB. (octet 35) Ce numéro est relatif au début du tampon secteur.

En sortie, l'octet choisi est retourné au programme appelant dans l'accumulateur A.

L'"indice d'accès direct" ne doit pas être inférieur à 4 car il n'y a pas de données utilisateur dans les 4 premiers octets du secteur.

exemple :

Lecture du 54ème octet de donnée

```
LDX   ## FCB        pointe sur l'adresse du FCB
LDAA  ## 54 + 4     init. numéro de l'octet choisi
STAA  35,X          sauvegarde dans le FCB
LDAA  ## 17         code fonction
STAA  0,X           sauvegarde dans le FCB
JSR   FMS           appel du FMS
BNE   ERRØR         test d'erreur
```

Maintenant, le caractère choisi est dans l'accu. A

Fonction 18 : Ecriture d'un octet dans le secteur par accès direct ($12 hex.)

Le fichier doit être ouvert pour mise à jour. Cette fonction est semblable, à la précédente, mis à part que le caractère chargé dans l'accumulateur A est écrit dans le secteur à la position indiquée par l'octet d'"indice d'accès direct" du FCB.

L'"indice d'accès direct" ne doit pas être plus petit que 4 car les 4 premiers octets de chaque secteur sont réservés à des données système.

Exemple :

Ecriture d'une donnée dans le 54ème octet du secteur courant

```
LDX   ## FCB        pointe sur l'adresse du FCB
LDAA  ## 54 + 5     init. numéro de l'octet choisi
STAA  35,X          sauvegarde dans le FCB
LDAA  ## 18         code fonction
STAA  0,X           sauvegarde dans le FCB
LDAA  CARACT.       acquisition du caractère à écrire
JSR   FMS           appel du FMS
BNE   ERRØR         test d'erreur
```

Le caractère a été écrit dans le secteur.

Fonction 19 : Réservée pour une utilisation future ($13 hex.)

Fonction 20 : Recherche du lecteur suivant ($14 hex.)

Cette fonction est utilisée pour trouver le prochain lecteur prêt. FLEX exécute cette fonction de manière différente suivant les versions disques 5" ou 8". Cela est dû aux limites du matériel.

Le déroulement de cette fonction est le suivant pour la version 8" :

Si le numéro de lecteur dans le FCB est $FF (hex), la recherche des lecteurs prêts démarre sur le lecteur 0. Si le numéro du lecteur dans le FCB est 0, 1, ou 2, la recherche démarre par le lecteur 1, 2 ou 3 respectivement. Si un lecteur est prêt, son numéro est chargé dans l'octet "numéro de lecteur" du FCB et le bit de condition 0 (zéro) dans le registre de condition du CPU est mis à "1". Si aucun des lecteurs n'est prêt, le bit Z du registre de conditions du CPU est mis à "0" et l'erreur # 16 (lecteur pas prêt) est positionné.

Dans la version 5", le déroulement de la fonction est le suivant :

Si le numéro de lecteur dans le FCB est $FF (hex), au retour le numéro de lecteur dans le FCB sera 0. Si cette fonction est appelée alors que le numéro de lecteur dans le FCB est 0, au retour celui-ci sera 1. Dans les deux cas, le bit de condition 0 dans le registre de condition du CPU sera mis à "1".

Si elle est appelée alors que le numéro de lecteur dans le FCB est 1 ou plus grand que 1, le numéro de lecteur reste inchangé, le bit Z du registre de conditions du CPU est mis à "0" et l'erreur # 16 (lecteur non prêt) est positionnée.

Fonction 21 : Accès direct à un enregistrement ($15 hex.)

C'est l'une des fonctions permettant l'accès direct à un secteur d'un fichier. Le numéro de l'enregistrement désiré est chargé dans le FCB, dans le "numéro d'enregistrement courant" (octets 32-33).

Avant d'utiliser cette fonction, le fichier doit être ouvert pour lecture ou pour mise à jour.

Le premier enregistrement des données d'un fichier est l'enregistrement n°1. Se positionner sur l'enregistrement 0, déclenchera la lecture du premier secteur de la Table des secteurs du fichier. Après s'être correctement positionné sur un enregistrement, le premier caractère lu (accès séquentiel) est le premier octet de données de l'enregistrement spécifié. Essayer d'accèder à un enregistrement qui n'existe pas, déclenche une erreur.

Pour plus d'informations sur les fichiers à accès direct, lire le chapitre intitulé "Fichiers à accès direct".

Exemple :

Positionnement sur l'enregistrement n°6

```
LDX   # FCB        pointe sur l'adresse du FCB
LDAA  # 6          init. position
STAA  93,X         sauvegarde dans le FCB
CLR   32,X         init M.S.B. position à 0
LDAA  # 21         code fonction
STAA  0,X          sauvegarde dans le FCB
JSR   FMS          appel du FMS
BNE   ERRØR        test d'erreur
```

L'enregistrement est prêt à être lu.

Fonction 22 : Retour arrière d'un enregistrement ($16 hex.)

Cette fonction est aussi utilisée pour les fichiers à accès direct. Elle prend le numéro d'enregistrement courant dans le FCB et le décrémente de 1. Un re-positionnement sur le nouvel enregistrement est exécuté. Cela a pour effet de se décaler en arrière d'un enregistrement en entier.

Par exemple, si le numéro d'enregistrement courant est #16 et que la fonction "retour arrière d'un enregistremeent" est exécutée, le fichier sera alors positionné pour permettre la lecture du premier octet de l'enregistrement # 15.

Le fichier doit être ouvert pour lecture ou pour mise à jour avant l'utilisation de cette fonction.

Voir le chapitre "fichiers à accès direct" pour plus de détails.

5 - FICHIERS A ACCES DIRECT

Le Système d'exploitation FLEX permet la gestion des fichiers en accès direct.

La technique de l'accès direct permet d'atteindre n'importe quel secteur d'un fichier en précisant simplement la position correspondant à cet enregistrement. La longueur du fichier n'a pas d'importance, et le temps d'accès au secteur choisi est celui de deux lectures, au maximum.

En utilisant le même mécanisme, l'utilisateur peut aussi aisément se positionner sur le nième caractère d'un secteur. Seul un petit calcul utilisant le nombre d'octets de données dans un secteur (252) est nécessaire.

Tous les fichiers ne peuvent être manipulés en accès direct. Il est nécessaire qu'ils aient été auparavant créés en tant que fichiers à accès direct.

Le mode de création d'un fichier est par défaut le mode séquentiel. Tous les utilitaires standard du FLEX fonctionnent avec des fichiers à accès séquentiel.

Le seul fichier à accès direct utilisé par FLEX est le fichier ERRØRS.SYS.

Un fichier créé comme un fichier à accès direct peut être lu de manière directe ou séquentielle.

Un fichier créé comme un fichier à accès séquentiel ne peut être lu que de manière séquentielle.

Pour créer un fichier à accès direct, la procédure normale d'ouverture pour écriture doit être utilisée. Puis, tout de suite après l'ouverture (si elle est réussie...), il faut initialiser l'indicateur de table des secteurs du fichier dans le FCB, à une valeur non nulle, ensuite la création du fichier peut commencer.

Seule la création de fichiers texte en accès direct, présente un intérêt.

Lors de la création du fichier le système crée une table des secteurs du fichier.

La table des secteurs du fichier (FSM) est une table ou catalogue qui permet au système de repérer chaque enregistrement (secteur) du fichier sur le disque.

La FSM est constituée de 2 secteurs et le numéro d'enregistrement 0 lui est affecté.

Ceci implique qu'un fichier de données, de 5 secteurs de long aura finalement une longueur de 7 secteurs s'il est créé en accès direct.

L'utilisateur n'a pas besoin de manipuler les secteurs de la FSM, aussi lorsqu'un fichier est ouvert en lecture, ils sont automatiquement sautés. Par contre, le FMS les utilise pour exécuter les fonctions 21 et 22.

La zone d'identification d'un fichier, contient l'information précisant si un fichier a été créé en accès direct ou non.

Si l'octet indicateur de table des secteurs d'un fichier est non nul, le fichier est un fichier à accès direct sinon, il est à accès séquentiel.

Il est à noter qu'un fichier à accès direct peut être recopié sur une autre disquette sans perdre ses propriétés, mais qu'il ne peut pas être fusionné à un autre fichier.

6 - ERREURS FLEX

1 - CODE DE FONCTION FMS ILLEGAL

La valeur du code de fonction dans le FCB est trop grande ou interdite. L'erreur se déclenche à l'appel du FMS.

2 - LE FICHIER SPECIFIE EST DEJA OUVERT

Une fonction d'ouverture pour lecture, mise à jour, ou écriture a été programmée dans un bloc de contrôle pour un fichier déjà ouvert.

3 - LE FICHIER SPECIFIE EXISTE DEJA

a) - Une ouverture pour écriture est exécutée avec un FCB contenant les spécifications d'un fichier déjà existant sur la disquette.

b) - La fonction "changement de nom d'un fichier" est demandée alors que le nom existe déjà sur la disquette.

4 - LE FICHIER SPECIFIE N'A PAS ETE TROUVE

L'ouverture d'un fichier pour lecture ou pour remise à jour ou pour ré-appelation ou pour destruction a été demandée alors que le FCB correspondant contient les spécifications d'un fichier n'existant pas sur la disquette.

5 - ERREUR CATALOGUE SYSTEME - RECHARGEZ LE SYSTEME

Réservé pour une utilisation future.

6 - PLUS D'ESPACE DISQUE POUR LE CATALOGUE

Cela signifie que le disque est plein, l'allocation d'espace pour le catalogue étant dynamique.

7 - TOUT L'ESPACE DISQUE A ETE UTILISE

Tout l'espace utile de la disquette a été utilisé par les fichiers. Si cette erreur est renvoyée par le FMS, le dernier caractère qui devait être écrit dans le fichier ne l'a pas été.

8 - FIN DE FICHIER RENCONTRE EN LECTURE

Une opération de lecture d'un fichier s'est terminée par la rencontre d'une Fin de Fichier (E.Ø.F.). Toutes les données du fichier ont été lues. Cette erreur peut être renvoyée au cours de la lecture du catalogue du disque, à l'aide de la fonction : "Acquisition des données d'un enregistrement" (fonction 7), quand la fin du catalogue est atteinte.

9 - ERREUR DE LECTURE SUR DISQUE

Une erreur de parité (ou autre...) a été détectée par le contrôleur au cours d'un essai de lecture d'un secteur.

Le FLEX aura automatiquement essayé de relire le secteur défectueux plusieurs fois sans succès avant de rapporter l'erreur.

Cette erreur peut aussi être due au chargement de numéros de piste et de secteurs illégaux dans le FCB.

10 - ERREUR D'ECRITURE SUR DISQUE

Une erreur de parité (ou autre...) a été détectée par le contrôleur au cours d'un essai d'écriture d'un secteur. Le FLEX aura automatiquement essayé de récrire plusieurs fois le secteur défectueux, sans succès, avant de rapporter l'erreur.

Cette erreur peut aussi être due au chargement de numéros de piste et de secteur illégaux, dans le FCB. Une erreur d'écriture sera aussi renvoyée si une erreur de lecture est détectée par le FLEX au cours de la mise à jour du catalogue de la disquette.

11 - FICHIER OU DISQUE PROTEGE EN ECRITURE

Un essai d'écriture a été tenté sur une disquette protégée en écriture (découpe d'autorisation d'écriture sur l'enveloppe du disque) ou sur un fichier alors que le bit de protection d'écriture est positionné à "1".

12 - FICHIER PROTEGE - NON EFFACE

Le bit de protection contre la destruction du fichier spécifié est mis à "1" et celui-ci ne peut être détruit.

13 - BLOC CONTROLE DE FICHIER ILLEGAL

Une tentative d'accès à un bloc de contrôle dans la chaine des FCB a été faite, mais celui-ci n'y existe pas.

14 - ADRESSE DISQUE ILLEGALE

Réservé pour une utilisation future.

15 - NUMERO D'UNITE DISQUE ILLEGAL

Réservé pour une utilisation future.

16 - UNITE DISQUE NON PRETE

Le lecteur n'a pas de disquette, ou sa porte n'est pas fermée. Ce message ne peut être sorti s'il n'y a aucun moyen de détecter cet état sur les unités (certains 5" par exemple).

17 - LE FICHIER EST PROTEGE - ACCES REFUSE

Réservé pour une utilisation future.

18 - VIOLATION DES ATTRIBUTS D'ACCES A UN FICHIER

a) - Une lecture ou un repositionnement ont été tentés sur un fichier fermé, ou ouvert pour écriture.

b) - Une écriture a été tentée sur un fichier fermé, ou ouvert pour lecture.

19 - POINTEUR D'ACCES DIRECT ERRONE

La fonction accès direct d'un octet du secteur est appelé avec un pointeur supérieur à 255.

20 - FMS INACTIF - RECHARGEZ LE SYSTEME

Réservé pour une utilisation future.

21 - SPECIFICATION FICHIER ILLEGALE

Une erreur de format a été détectée dans l'appelation d'un fichier.

Le nom du fichier doit commencer par une lettre, et ne doit contenir que des lettres, des chiffres, des traits d'union et/ou des soulignés.

Idem pour l'extension d'un fichier.

Le nom d'un fichier est limité à 8 caractères, l'extension à 3.

22 - ERREUR SYSTEME EN FERMETURE FICHIER

Réservé pour une utilisation future.

23 - DEBORDEMENT TABLE D'ALLOCATION - DISQUE TROP SEGMENTE

Une tentative de création d'un très grand fichier à accès direct, a été faite sur un disque trop segmenté. Toutes les informations sur l'emplacement des enregistrements n'ont pû être rangées dans les 2 secteurs de la table des secteurs du fichier (FSM).

Pour résoudre ce problème, il suffit de recréer ce fichier sur une nouvelle disquette préalablement formatée.

24 - NUMERO D'ENREGISTREMENT INEXISTANT

Au cours de l'exécution de la fonction "Accès direct à un enregistrement" (fonction 21), le numéro de l'enregistrement choisi est plus grand que le numéro du dernier enregistrement du fichier.

25 - ERREUR NUMERO D'ENREGISTREMENT - FICHIER ENDOMMAGE

L'enregistrement localisé par le FMS au cours d'un accès direct au fichier, n'est pas le bon. Le fichier est probablement endommagé.

26 - ERREUR DE SYNTAXE - REDONNEZ LA COMMANDE

Une erreur de syntaxe a été détectée dans la ligne de commande venant d'être tapée.

27 - COMMANDE INTERDITE PENDANT L'IMPRESSION

La commande qui vient d'être entrée ne peut pas être exécutée pendant une impression simultanée ("spooling" actif).

28 - CONFIGURATION HARDWARE INSUFFISANTE

Cette erreur est affichée généralement, si la configuration mémoire installée ne suffit pas à l'exécution d'une fonction donnée.

7. INFORMATIONS PARTICULIERES SUR LES DISQUES ET FICHIERS

7.1 - Modules d'interface du disque

Les informations suivantes sont pour les utilisateurs qui souhaitent écrire leurs propres modules d'interface. Aucune garantie ne peut être réclamée en cas de disfonctionnement du FLEX, si les modules d'interface du disque ont été modifiés.

Les modules d'interface du disque, sont des programmes d'interface entre le FLEX et le contrôleur du système disque. Les modules fournis avec FLEX sur GOUPIL sont prévus pour les contrôleurs formateurs Western Digital (série 1771 ou 179X).

Les modules d'interface du disque sont localisés en RAM entre $DE80 (hex) et $DFFF (hex). Toutes les fonctions du disque sont des sauts vectorisés au début de cette zone. En cas d'erreur, les modules d'interface du disque n'ont pas à tenter de reprises ; FLEX les gère automatiquement. Il suffit que le code erreur fourni par le contrôleur soit chargé dans le registre B du CPU et que le bit Z du registre de condition du CPU soit mis à "0". Les codes d'erreurs doivent être conformes aux codes Western Digital.

Ces codes doivent donc être reproduits par logiciel si un autre type de contrôleur est utilisé.

Tous les modules d'interface du disque ne modifie pas le contenu du registre d'index du CPU (X).

Tous les modules d'interface du disque sont appelés par une instruction JSR.

Les modules d'interface décrit ci-après sont communs à toutes les versions du FLEX.

$DE80 - LECTURE

Entrée :

(X) = adresse du buffer secteur dans le FCB
(A) = n° de piste
(B) = n° de secteur.

Le secteur désigné par le numéro de piste et de secteur sur le disque est lu et chargé dans le tampon secteur du FCB indiqué.

$DE83 - ECRITURE

Entrée :

(X) = adresse du buffer secteur dans le FCB
(A) = n° de piste
(B) = n° de secteur.

Le contenu du tampon secteur du FCB indiqué est écrit sur le disque, dans le secteur désigné par les numéros de piste et de secteur.

$DE86 - VERIFICATION

Entrée : pas de paramètres.

Le secteur qui vient d'être écrit est vérifié pour repérer d'éventuelles erreurs CRC.

$DE89 - REPOSITIONNEMENT SUR LA PISTE 00

Entrée : (X) = adresse du FCB.

Sortie : bit C = 0, bit Z = 1 et (B) = $0B si le disque est protégé en écriture

bit C = 1, bit Z = 1 et (B) = $0F s'il ny a pas de disque.

Une opération de repositionnement sur la piste 00 est exécutée sur le disque dont le numéro est contenu dans le bloc de contrôle du fichier (FCB) désigné.

$DE8C - SELECTION DU DISQUE

Entrée : (X) = adresse du FCB.

Le disque dont le numéro est contenu dans le bloc de contrôle du fichier sera sélectionné.

$ E8F - TEST DISQUE PRET

Entrée : (X) = adresse du FCB

Sortie : bit Z = 1 et bit C = 1 si le disque n'est pas prêt

bit Z = 0 et bit C = 0 si le disque est prêt.

Ce module existe s'il est possible de tester l'état du disque dont le numéro est précisé dans le FCB, après avoir sélectionné ce disque et attendu le temps nécessaire à la stabilisation de la vitesse des moteurs (1 seconde environ dans le cas des 5"). Dans les configurations qui ne permettent pas ce test, ce module doit simplement renvoyer l'état prêt si le numéro du disque est 0 ou 1 dans le FCB, et un état pas prêt pour tout autre numéro ;

$DE92 - TEST RAPIDE DISQUE PRET

7.2 - Initialisation des disquettes

La commande NEWDISK est utilisée pour "initialiser" une disquette suivant un format reconnu par FLEX.

Le processus d'initialisation consite en l'écriture des numéros de piste et de secteurs sur chaque secteur d'une disquette, suivant le format utilisé par FLEX. En plus, ce processus relie ensemble tous les secteurs de la disquette pour former une chaine de secteurs disponibles.

La première piste de la disquette, piste 00, est spéciale.

Aucun des secteur de cette piste n'est autorisée à recevoir des fichiers de données. Ils sont réservés à l'usage du système FLEX.

Les deux premiers secteurs contiennent un "chargeur binaire" qui est appelé par la commande [↵] du moniteur GPMØN. Ce "chargeur binaire", une fois résident en mémoire centrale charge FLEX de la disquette vers le système.

Le secteur 3 sur cette piste contient l'enregistrement des informations système. Ce secteur contient les adresses physiques sur le disque, de début et de fin des secteurs de la chaine des secteurs libres, réservés aux fichiers de données.

Tout le reste des secteurs à partir du 5 de la piste 00, est réservé au catalogue des noms de fichiers.

Après l'initialisation, toutes les pistes disponibles ont un format commun.

Les deux premiers octets de chaque secteur contiennent le numéro de la piste et du secteur du prochain secteur de la chaine. Les deux octets suivants seront utilisés pour y charger le numéro logique du secteur dans le fichier. Les 252 octets restants sont à 0.

A l'originetous les octets représentent les numéros d'enregistrement sont à 0.

Quand des données sont stockées dans un fichier, les deux octets de lien au début de chaque secteur sont modifiés pour pointer sur le secteur suivant dans le fichier et non plus dans la chaîne des secteurs disponibles.

Les secteurs de l'inventaire du disque sur la piste 00 ont aussi des octets de lien de même type que ceux décrits dans la chaine des secteurs disponibles et dans les fichiers de données.

Une disquette FLEX n'est pas initialisée dans un format IBM standard. En effet, dans un format standard, tous les secteurs se suivent physiquement dans l'ordre défini par la suite logique de leur numéro.

i.e. Le secteur 2 suit le secteur 1, le secteur 3 suit le secteur 2 etc...

Sur une disquette FLEX les secteurs sont entrelacés. Si les secteurs se suivaient physiquement, à la lecture d'un enregistrement, après l'acquisition des données, le secteur suivant directement celui-ci serait déjà passé sous la tête de lecture avant que le deuxième ordre de lecture ne soit donné. Aussi l'entrelaçage des secteurs accorde-t-il plus de temps pour manipuler les données.

Le phénomène de "perte" d'un secteur du aux temps de traitement relativement élevés est appelé "révolutions perdues", et a pour résultat de ralentir considérablement l'exécution des programmes utilisant des fichiers.

Le format du FLEX permet de réduire le nombre de révolutions perdues et donc de diminuer le temps d'exécution d'un programme.

7.3 - Description d'un secteur du catalogue

Chaque secteur dans la partie de la disquette FLEX réservée au catalogue, est composé de 10 "enregistrements catalogue". Chaque enregistrement correspond à un fichier sur la disquette. Dans chaque secteur, les quatre premiers octets contiennent les informations de chainage, et les 12 suivants ne sont pas utilisés.

Si on utilise la fonction "Acquisition des données d'un enregistrement" pour lire les informations contenues dans le catalogue, ces 16 octets sont automatiquement sautés.

Chaque enregistrement du catalogue contient exactement les mêmes informations que celles stockées dans la zone d'identification du fichier dans les FCB (octets 4-27).

Pour plus de détails, se référer à la description du bloc de contrôle du fichier (FCB).

Un enregistrement du catalogue qui n'a jamais été utilisé a un 0 dans le premier octet du nom du fichier.

Un enregistrement du catalogue qui a été effacé a le bit de poids le plus fort du premier octet du nom du fichier mis à "1" (i.e. le premier octet du nom du fichier est négatif).

DESCRIPTION D'UN SECTEUR DE DONNEES

Chaque secteur sur une disquette FLEX (sauf les deux secteurs du "chargeur binaire") a le format suivant :

- Octets 0-1 adresse du prochain secteur
- Octets 2-3 numéro logique du secteur dans le fichier
- Octets 4-255 données

Si un fichier est composé de plus d'un secteur, les octets "d'adresse du prochain secteur", contiennent les numéros de la piste et du secteur, respectivement, du prochain secteur dans le fichier. Ces octets sont à zéro, si c'est le dernier secteur du fichier (Condition E.Ø.F. : fin de fichier).

L'utilisateur ne doit jamais changer les octets de chainage d'un secteur, ceux ci étant automatiquement gérés par le FMS.

7.4 - Description d'un fichier binaire

Sous FLEX, un fichier binaire peut contenir n'importe quel type de données ; tous les caractères ASCII sont autorisés. Chaque fichier binaire est composé d'un ou de plus d'un enregistrement binaire. Il peut y avoir plus d'un enregistrement binaire dans un secteur.

Un fichier binaire a le format suivant :

- Octet 0 Indicateur de début d'enregistrement ($02, STX)
- Octet 1 MSB de l'adresse de chargement
- Octet 2 LSB de l'adresse de chargement
- Octet 3 Nombre d'octets de données dans l'enregistrement
- Octet 4-n Données binaires de l'enregistrement.

L'adresse de chargement représente l'adresse de la zone mémoire où les données étaient stockées avant d'être écrites sur disque (commande : SAVE).

Quand un fichier binaire est chargé en mémoire à partir du disque, il sera rangé dans la même zone mémoire que celle d'où il est issu.

Un fichier binaire peut aussi contenir une adresse de transfert (adresse d'exécution) facultative dans un enregistrement. Cet enregistrement donne alors l'adresse mémoire du point d'entrée du fichier binaire.

Le format d'un enregistrement contenant une adresse de transfert est le suivant :

- Octet 0 Indicateur d'adresse de transfert ($16.AC4)
- Octet 1 MSB de l'adresse de transfert
- Octet 2 LSB de l'adresse de transfert.

Si un fichier binaire contient plus d'un enregistrement possédant une adresse de transfert (concaténation de fichiers binaires), le dernier rencontré par la procédure de chargement, sera celui qui sera utilisé, les autres seront ignorés.

Pour lire ou écrire un fichier binaire à travers le FMS, directement à partir d'un programme utilisateur, le programme appelant doit lui même gérer les octets d'informations en en-tête de l'enregistrement ; FLEX ne fournira pas ces informations à l'utilisateur.

7.5 - Description d'un fichier source

Un fichier source (appelé aussi "fichier ASCII", "fichier codé", ou "fichier texte") ne contient que des caractères ASCII imprimables, plus quelques caractères de contrôle spéciaux.

Il n'y a pas d'adresse de chargement pour un fichier source. C'est au programme qui lit le fichier texte, qu'il appartient de mettre les données du fichier à leur place.

Les seuls caractères de contrôle reconnus ou manipulés par FLEX sont :

- $0D (ASCII : CR ou RETOUR CHARIOT)

Ce caractère est utilisé pour marquer la fin d'une ligne ou d'un enregistrement dans le fichier.

- $00 (ASCII : NULL)

Ce caractère est ignoré par FLEX. S'il est rencontré dans un fichier, il ne sera pas renvoyé au programme appelant.

- $18 (ACSII : ÀNNULER)

Ce caractère est ignoré par FLEX. S'il est rencontré dans un fichier, il ne sera pas renvoyé au programme appelant.

- $09 (ASCII : HT ou TABULATION HORIZONTALE)

C'est un caractère drapeau qui indique qu'une chaine d'espace a été retirée du fichier. L'octet suivant ce caractère contient le nombre d'espaces supprimés (2-127).

Le programme appelant ne voit ni le caractère drapeau, ni le compteur de caractères. Le nombre exact d'espaces est retourné au programme utilisateur caractère par caractère à chaque exécution de la fonction "lecture du prochain caractère" (fonction 0).

A l'écriture d'un fichier, les espaces sont automatiquement effacés au fur et à mesure qu'ils sont envoyés au FMS par la fonction "Ecriture du prochain caractère". La compression des données est donc totalement transparente pour le programme appelant.

(Cette description n'est valable que pour des fichiers sources. Si le fichier ouvert est un fichier binaire, le caractère drapeau et le compteur seront transmis identiques, à ce qu'ils sont dans le fichier).

7.6 - Création de commandes utilitaires

Les commandes utilitaires peuvent être réalisées en assembleur. FLEX réserve un espace mémoire dans lequel des utilitaires de taille moyenne peuvent être chargés.

Cette zone mémoire commence en $ C100 (hex) et peut s'étendre jusqu'en $ C6FF. Le FCB système localisé en $ C840 peut être aussi utilisé par les commandes utilitaires créées par le programmeur, comme zone de FCB ou de stockage temporaire. Il ne doit pas y avoir de données résidant dans cette zone de FCB tant qu'il peut y avoir des interférences avec le chargement des programmes utilitaires (FLEX utilise cette zone pour charger les commandes utilitaires en mémoire).

Un exemple sera donné plus loin en démonstration des conventions et des techniques à utiliser pour créer des commandes utilitaires.

Cet exemple dont on trouvera le listing dans les pages suivantes est simplement un utilitaire permettant de lister un fichier texte sur l'écran.

La syntaxe de la commande est :

LIST, [<identification du fichier>]

L'extension spécifiée par défaut est TXT. Le contenu du fichier sera affiché sur le terminal ligne par ligne.

DESCRIPTION DU LISTING SOURCE DE L'UTILITAIRE LIST

1) EQUIVALENCES FLEX

C'est un ensemble de paramètres indiquants à l'assembleur l'emplacement mémoire des différents sous-programmes du FLEX utilisés.

Ces paramètres font partie des adresses données dans ce manuel, dans le chapître "sous-programmes système appelables par l'Utilisateur".

2) EQUIVALENCES FMS

Ce sont les points d'entrée du FMS, utilisés au cours de l'exécution du programme.

3) EQUIVALENCES SYSTEME

Cette section contient l'adresse de début du bloc de contrôle du fichier (FCB) utilisé.

L'adresse de début de stockage du programme est donnée par l'instruction ORG. La zone mémoire réservée aux commandes utilitaires est utilisée,aussi l'origine du programme est en $ C100.

4) PROGRAMME

Une des premières conventions à respecter à l'écriture d'une commande utilitaire est de toujours commencer le programme par une instruction BRA suivie d'une ligne de type "VN FCB 1" qui définira le numéro de version de l'utilitaire.

Dans cet exemple, le numéro de version est 1.

Cette convention permet l'utilisation correcte de la commande utilitaire VERSIØN.

En "LIST 2", le programme fait l'acquisition des paramètres d'identification du fichier et les stocke dans le FCB. Le sous-programme GETFIL est utilisé en plus de l'acquisition et de la sauvegarde des informations, il teste d'éventuelles erreurs. Si tout se passe bien, le bit de retenue (c) dans le registre de condition du CPU est mis à "0', autrement, une ou plusieurs erreurs ont été détectées dans le nom du fichier (C=1) et le contrôle est passé à la ligne d'étiquette "LIST 9". A cet endroit, un message d'erreur est affiché et la main est redonnée au système FLEX.

Si le nom du fichier est correct et que le bit de retenue du registre de condition est à zéro, une extension TXT est donnée par défaut au nom du fichier. Il suffit de mettre le code de l'extension TXT dans l'accumulateur A (TXT=1) et d'appeler le sous-programme SETEXT résidant dans le DØS. Le registre d'index doit pointer sur l'adresse du FCB choisi. Le code "1" est aussi chargé dans l'octet de code fonction du FCB en prévision de l'opération à effectuer sur le fichier (ouverture en lecture). Il n'y a pas d'erreurs générées par un appel à SETEXT.

Maintenant, le fichier peut être ouvert en lecture, le code fonction correspondant ayant déjà été chargé dans le FCB et le registre d'index du CPU pointant toujours sur l'adresse de début du FCB, il suffit d'appeler le FMS.

Si une erreur est détectée, l'instruction "BNE LIST 9" passera le contrôle au module de gestion de l'erreur.

La première chose que celui-ci fait, c'est d'appeler le sous-programme du DØS, RPTERR qui affichera sur le terminal le message correspondant au code de l'erreur détectée. Ensuite tous les fichiers ouverts seront fermés ; ceci est facilement réalisé en appelant le point d'entrée du FMS : "fermeture des fichiers" (FMSCLS).

Finalement, le contrôle est redonné au DØS au point d'entrée à chaud.

Si l'ouverture du fichier s'est déroulée avec succès, le programme continue à la ligne d'étiquette LIST4". A ce moment, il est possible de faire l'acquisition des caractères un par un à l'intérieur du fichier, et de les afficher sur le terminal. Les "Retour Chariot" ne sont pas stockés dans un fichier texte. (Ils marquent la fin d'une ligne, mais la ligne suivante suit immédiatement après), aussi tous les "Retour Chariot" reçus par le système ne sont pas affichés. Pour passer à la ligne à la réception du caractère "Retour Chariot", il suffit de faire appel au sous-programme du DØS - "PCRLF".

Chaque fois qu'un caractère venant du fichier est reçu par le système (appel du FMS en LIST4), un test d'erreur est exécuté. Si une erreur est détectée, le contrôle est passé à ligne d'étiquette "LIST6". Comme le FLEX ne sauvegarde pas un caractère de "Fin de Fichier" dans le fichier, la seule façon de déterminer la fin du fichier est de tester l'erreur "Fin de Fichier" générée par le FMS (code erreur 8). Si le code erreur envoyé n'est pas égal à 8, le contrôle est passé au module de gestion de l'erreur décrit plus haut. S'il est égal à 8, l'édition du fichier est terminée, aussi doit-il être fermé. Le code fonction FMS de fermeture de fichier est 4. Il est chargé dans l'accumulateur A et stocké dans le FCB, un appel au FMS suffira alors pour fermer le fichier. En retour, les erreurs sont testées et s'il n'y en a pas, la main est redonnée au DØS en appelant le point d'entrée à chaud du FLEX.

Cet exemple illustre une des méthodes utilisées pour créer un utilitaire. De nombreuses routines du DØS et du FMS sont appelées. Le concept de base de l'ouverture ou de la fermeture d'un fichier est démontré, ainsi que celui des Entrées/Sorties. La méthode de traitement des erreurs est aussi exposée.

L'étude approfondie de cet exemple permettra à l'utilisateur de comprendre parfaitement les techniques de création des commande utilitaires.

CØMMANDE UTILITAIRE = LIST


```
                * EQUIVALENCES DØS

CD03            WARMS   EQU     $CD03       POINT D'ENTREE A CHAUD
CD2D            GETFIL  EQU     $CD2D       ACQUI. SPECIF. FICHIER
CD18            PUTCHR  EQU     $CD18       SORTIE DU CARACTERE
CD24            PCRLF   EQU     $CD24       IMPRESSION SAUT A LA LIGNE
CD33            SETEXT  EQU     $CD33       INIT. EXTENSION PAR DEFAUT
CD3F            RPTERR  EQU     $CD3F       AFFICHAGE ERREURS DISQUE

                * EQUIVALENCES FMS

D406            FMS     EQU     $D406
D403            FMSCLS  EQU     $D403

                * EQUIVALENCES SYSTEME

C840            FCB     EQU     $C840       FCB SYSTEME

                * DEBUT DU PROGRAMME

C100                    ØRG     $C100
C100 20 01      LIST    BRA     LIST2       DEBUT
C102 01         VN      FCB     1           NUMERO DE LA VERSION
C103 CE C8 40   LIST2   LDX     #FCB        POINTE SUR L'ADRESSE DU FCB
C106 BD CD 20           JSR     GETFIL      ACQUIS. SPECIF. FICHIER
C109 25 34              BCS     LIST9       ERREURS ???
C10B 86 01              LDA A   #1          CODE TXT OU LECTURE
C10D A7 00              STA A   0,X A       SAUVEGARDE DANS LE FCB
C10F BD CD 33           JSR     SETEXT      EXTENSION PAR DEFAUT
C112 BD B4 06           JSR     FMS         APPEL DU FMS. OUVERTURE DU FICHIER
C115 26 28              BNE     LIST9       TEST D'ERREUR
C117 CE C8 40 LIST4     LDX     #FCB        POINTE SUR L'ADRESSE DU FCB
C11A BD D4 06           JSR     FMS         APPEL DU FMS. ACQUIS. D'UN CARACT.
C11D 26 0E              BNE     LIST6       ERREURS ???
C11F 31 0D              CMP A   #$D         RETOUR CHARIOT ???
C121 26 05              BNE     LIST5
C123 BD CD 24           JSR     PCHLF       SORTIE SAUT A LA LIGNE
C126 20 EF              BRA     LIST4       REPETITION
C128 BD CD 18 LIST5     JSR     PUTCHR      SORTIE DU CARACTERE
C12B 20 EA              BRA     LIST4       REPETITION
C12D A6 01      LIST6   LDA A   1,X         ACQUIS. CODE ERREUR
C12F 81 08              CMP A   #8          ERREUR FIN DE FICHIER ???
C131 26 0C              BNE     LIST9
C133 86 04              LDA A   #4          CODE FERMETURE DU FICHIER
C135 A7 00              STA A   0,X         SAUVEGARDE DANS LE FCB
C137 BD D4 06           JSR     FMS         APPEL DU FMS. FERMETURE
C13A 26 03              BNE     LIST9       ERREURS ???
C13C 7E CD 03           JMP     WARMS       RETOUR SOUS FLEX
C13F BD CD 3F LIST9     JSR     RPTERR      AFFICHAGE DE L'ERREUR
C142 BD D4 03           JSR     FMSCLS      FERMETURE DE TOUS LES FICHIERS
C145 7E CD 03           JMP     WARMS       RETOUR SOUS FLEX
                        END     LIST
```

7.7 - Commande utilitaire : LINK

L'utilitaire LINK fourni sur la disquette FLEX est une commande spéciale. Sa seule fonction est de préciser au "chargeur binaire", qui se trouve sur la piste 00, où se trouve le système FLEX sur le disque.

LeFLEX pouvant résider n'importe où sur le disque, LINK prend son adresse physique sur le disque, et la stocke dans un des pointeurs des secteurs du chargeur.

Quand le programme s'exécute, le chargeur lit simplement cette adresse et charge en mémoire le fichier binaire qui se trouve à cet endroit. Le processus de chargement s'arrête à la réception d'une adresse de transfert. A ce moment, le contrôle est passé au programme venant d'être chargé par une instruction de saut (JMP) à l'adresse de transfert reçue. Si jamais le programme est déplacé sur le disque après avoir été "relié" au chargeur binaire, il est nécessaire de relancer la commande LINK.

LINK peut être utilisée dans plusieurs applications. L'une d'elles est le développement d'un système d'exploitation propre à l'utilisateur. L'utilisateur peut parfaitement utiliser son propre système d'exploitation, le relier au chargeur binaire et l'utiliser à la place du FLEX.

Il peut être aussi intéressant dans des configurations spéciales de pouvoir charger quelques modules spécialisés avant le FLEX. Si cela est le cas, il faut se rappeler que tant que le FLEX n'a pas été chargé en mémoire, il n'y a ni module d'exploitation des disques résidents en MEV, ni système de gestion de fichiers.

7.8 - Modules d'impression

Deux programmes d'impression sont fournis avec FLEX. Ils ne peuvent se concevoir l'un sans l'autre. Le premier est la commande utilitaire "P" et l'autre est le fichier "PRINT.SYS" (initialisation du terminal imprimant et sortie du caractère).

Le listing source de la commande "P" est fourni dans les pages suivantes.

Le fichier PRINT.SYS peut être modifié afin de s'adapter au terminal choisi par l'utilisateur. On trouvera ci-après les impératifs à suivre à la création du fichier PRINT.SYS.

Le listing source de celui fourni avec la disquette FLEX (interface parallèle de type CENTRONICS) se trouve dans le "Manuel d'utilisation FLEX."

CARACTERISTIQUES DU FICHIER "PRINT.SYS"

Le fichier doit fournir au système au moins trois sous-programmes de base.

- un sous-programme d'initialisation de l'imprimante
- un sous-programme de test de l'imprimante
- un sous-programme de sortie du caractère

La commande "P" et le handler physique d'impression utilisent ces trois sous-programmes pour communiquer avec l'imprimante. Les trois modules et leurs caractéristiques sont décrits ici :

PINIT : ($CCCO-CCD7) : ce module sert à initialiser le port de l'imprimante. Aucun des registres du CPU n'a besoin d'être sauvegardé.

PCHK : ($CCD8-CCE3) : ce module teste l'état de l'imprimante pour voir si elle est prête à accepter un autre caractère. Un code condition négatif est renvoyée si elle n'est pas libre, positif dans l'autre cas. Les registres A, B et X ne doivent pas être modifiés en sortie.

PØUT : ($CCE4-CCF7) : ce module sort sur l'imprimante le caractère contenu dans l'accu. A après avoir testé si l'imprimante peut le recevoir (PCHK). Les registres B et X ne doivent pas être modifiés.

MODULE D'IMPRESSION SIMULTANEE

FLEX contient un module d'impression simultanée ("Spooling") Essentiellement, ce module est un système multi-tâche, la fonction d'impression étant une tâche à priorité de bas niveau.

Chaque demande d'accès au disque interrompt l'impression tant que le disque est utilisé. Il faut noter que les vecteurs d'interruption SWI et LRQ sont utilisés par le "spooler". La commande PRINT est utilisée pour mettre en route ce module qui imprimera tour à tour les fichiers contenus dans la queue d'impression.

```
* COMMANDE UTILITAIRE "P"
* LA COMMANDE "P" INITIALISE UN PORT D'E/S ET MODIFIE LE VECTEUR DE SAUT
  A ØUTCH DANS FLEX.
    1                       OPT    PAG,NOG
    2                       TTL    P COMMAND
P COMMAND                          26-11-81  ASSEMBLEUR TSC 6800   PAGE    1


    4
    5                * COPYRIGHT 1978 BY TSC
    6
    7                * EQUATES
    8
    9  0010          INDEX   EQU    $0010
   10  C840          FCB     EQU    $C840
   11  CD30          LOAD    EQU    $CD30
   12  D406          FMS     EQU    $D406
   13  D403          FMSCLS  EQU    $D403
   14  CD06          RENTER  EQU    $CD06
   15  0004          NFER    EQU    $4
   16  CC09          PAUSE   EQU    $CC09
   17  CD1E          PSTRNG  EQU    $CD1E
   18  CD3F          RPTERR  EQU    $CD3F
   19  CD03          WARMS   EQU    $CD03
   20  CC11          LSTTRM  EQU    $CC11
   21  CC02          EOL     EQU    $CC02
   22  CCC0          PINIT   EQU    $CCC0
   23  CCE4          POUT    EQU    $CCE4
   24  CD0F          OUTCH   EQU    $CD0F
   25  CCFC          PR1     EQU    $CCFC
   26
   27  C100                  ORG    $C100
   28
   29  C100 20 01    P       BRA    P1
   30
   31  C102 01       VN      FCB    1
   32
   33  C103 B6 CC FC P1      LDA A  PR1
   34  C106 27 09            BEQ    P12
   35  C108 CE C8 40         LDX    £FCB
   36  C10B C6 1B            LDA B  £27
   37  C10D E7 01            STA B  1,X
   38  C10F 20 43            BRA    P3
   39  C111 B6 CC 11 P12     LDA A  LSTTRM
   40  C114 81 0D            CMP A  £$D
   41  C116 27 45            BEQ    P8
   42  C118 B1 CC 02         CMP A  EOL
   43  C11B 27 40            BEQ    P8
   44  C11D 7F CC 09         CLR    PAUSE
   45  C120 B6 CC E4         LDA A  POUT
   46  C123 81 39            CMP A  £$39
   47  C125 26 13            BNE    P15
   48  C127 CE C8 40         LDX    £FCB
   49  C12A 86 01            LDA A  £1
   50  C12C A7 00            STA A  0,X
   51  C12E BD D4 06         JSR    FMS
   52  C131 26 13            BNE    P2
   53  C133 86 FF            LDA A  £$FF
   54  C135 A7 3B            STA A  $3B,X
   55  C137 BD CD 30         JSR    LOAD
   56  C13A BD CC C0 P15     JSR    PINIT
   57  C13D CE CC E4         LDX    £POUT
   58  C140 FF CD 10         STX    OUTCH+1
```

SMT

```
P COMMAND                          26-11-81  ASSEMBLEUR TSC 6800   PAGE    2

   59  C143 7E CD 06          JMP    RENTER
   60  C146 A6 01     P2      LDA A  1,X
   61  C148 81 04             CMP A  #NFER
   62  C14A 26 08             BNE    P3
   63  C14C CE C1 62          LDX    #NOPST
   64  C14F BD CD 1E  P25     JSR    PSTRNG
   65  C152 20 03             BRA    P4
   66  C154 BD CD 3F  P3      JSR    RPTERR
   67  C157 BD D4 03  P4      JSR    FMSCLS
   68  C15A 7E CD 03          JMP    WARMS
   69  C15D CE C1 79  P8      LDX    #ERSTR
   70  C160 20 ED             BRA    P25
   71
   72  C162 22        NOPST   FCC    '"PRINT.SYS" non trouvé'
   73  C178 04                FCB    4
   74  C179 22        ERSTR   FCC    /"P" doit etre suivie d'une commande/
   75  C19C 04                FCB    4
   76
   77  C843                   ORG    $C843
   78
   79  C843 FF                FCB    $FF
   80  C844 50                FCC    'PRINT'
   81  C849 00                FCB    0,0,0
   82  C84C 53                FCC    'SYS'
   83
   84                         END    P

AUCUNE ERREUR DETECTEE
```

```
P COMMAND                          26-11-81  ASSEMBLEUR TSC 6800   PAGE    3

  TABLE SYMBOLES

EOL    CC02    ERSTR  C179    FCB    C840    FMS    D406    FMSCLS D403
INDEX  0010    LOAD   CD30    LSTTRM CC11    NFER   0004    NOPST  C162
OUTCH  CD0F    P      C100    P1     C103    P12    C111    P15    C13A
P2     C146    P25    C14F    P3     C154    P4     C157    P8     C15D
PAUSE  CC09    PINIT  CCC0    POUT   CCE4    PR1    CCFC    PSTRNG CD1E
RENTER CD06    RPTERR CD3F    VN     C102    WARMS  CD03
```

```
* COPYRIGHT (C) 1978 BY
* TECHNICAL SYSTEM CONSULTANTS,INC.
```

7.9 - Les interruptions dans FLEX

FLEX fait une utilisation intensive des interruptions pendant l'impression simultanée.

Tous les utilitaires du FLEX, l'Editeur, l'Assembleur, le Processeur de texte et le Basic sont interruptibles.

Les programmes utilisateurs peuvent eux-mêmes être interrompus. Mais à aucun moment, les vecteurs d'IRQ ou SWI ne doivent être modifiés dans un utilitaire utilisant les modules d'impression.

CARTES MEMOIRE OU SYSTEME

0000 B FFF	RAM utilisateur
C000 C07F	Pile système
C080 C0FF	Buffer d'entrée
C100 C6FF	Zone des commandes utilitaires
C700 C83F	Superviseur et modules d'impression
C840 C97F	FCB du système
C980 CBFF	Zone des fichiers systèmes
CC00 D3FF	DØS
D400 DE7F	FMS
DE80 DFFF	Modules d'exploitation du disque

Imprimé en France. — Imprimerie JOUVE, 18, rue Saint-Denis, 75001 PARIS

goupil

Société de Micro-informatique et Télécommunications

6. EDITEUR DE TEXTE GOUPIL 2

1982

CHAPITRE I

INTRODUCTION

I.1 Préambule

Le but de ce chapitre est de présenter brièvement au lecteur l'éditeur de texte. Pour ce faire, nous illustrerons son fonctionnement par plusieurs exemples . la convention suivante est utilisée : ce qui est souligné est frappé au clavier, ce qui ne l'est pas est affiché par l'éditeur.

L'éditeur appelé répond (1) (NEW FILE par exemple comme ci-dessous). Dès lors nous créerons notre fichier simplement en frappant toutes les lignes qui le composent, chaque ligne se terminant par un "retour chariot".

NEW FILE :

```
  1.00 = VØICI UN EXEMPLE D'UTILSATIØN DE
  2.00 = L'EDITEUR DE TEXTE. BEAUCØUP
  3.00 = D'EXEMPLES NØUS PERMETTRØNT D'APPRENDRE
  4.00 = RAPIDEMENT ET FACILEMENT SES CARACTERISTIQUES
  5.00 = LES LIGNES SUIVANTES N'ØNT AUCUN SENS :
  6.00 = ABCDEFGHIJKL
  7.00 = AAAAAAAA
  8.00 = TEST 1234
  9.00 = L'EDITEUR EST AMUSANT A UTILISER !
 10.00 = BBBBBBBB
 11.00 =
 12.00 = NØUS VØICI A LA FIN DE CE FICHIER
 13.00 = AU MØINS PØUR L'INSTANT
 14.00 = ##
 13.00 = AU MØINS PØUR L'INSTANT
##
```

(1) cf Ch.III

> Attention ! Ø représente la lettre O.
> 0 représente le chiffre zéro.

Vous remarquerez qu'il a fallu frapper un dièse (##) en première colonne pour sortir du mode saisie. A ce moment le système a récrit la dernière ligne et affiché le dièse. L'éditeur est prêt à accepter des commandes. A tout instant , quand on frappe des caractères dans l'éditeur, deux caractères ont un sens spécial :

BS (BACKSPACEou CØNTRØL H)	Efface le dernier caractère (retour arrière)
CAN (CØNTRØL X)	Efface toute la ligne courante

Ces commandes sont utiles, en cas de fautes de frappe, et permettent la correction immédiate. (sur GOUPIL 2 utiliser les touches ← ou CTRL H, et CTRL X du clavier.).

I.2 Identification des lignes

Un numéro est donné à chaque ligne de texte de l'éditeur et permet d'identifier la ligne. Chaque numéro de ligne est de la forme m.nn'où m est un nombre entier et n et n' des chiffres de 0 à 9. 73, 73., 73.0 et 73.00 désignent la même ligne ; de même pour 259.6 et 259.60.

Le plus grand numéro utilisable est : 9999.99

Nous noterons le repérage d'une ligne par le symbole "⟨ligne⟩". Nous utiliserons cette convention tout au long de ce document.

Un ordre indique à l'éditeur l'action à accomplir et, si besoin est , la ligne ou le bloc de lignes affecté par cet ordre.

I.3 Les commandes de l'éditeur

L'éditeur peut supprimer ou insérer des lignes de texte dans un fichier,imprimer ces mêmes lignes de texte et ainsi de suite. A chaque possibilité correspond une instruction : par exemple une instruction de suppression, une instruction d'insertion, une instruction d'impression, etc.Si nous définissons le symbole "⟨instruction⟩" pour signifier "instruction" à l'éditeur, la forme de base d'une instruction d'édition est :

"⟨ligne⟩" "⟨instruction⟩"

Par exemple, l'ordre pour afficher la ligne 12.00 est :

```
#12 P
12.00 = NØUS VØICI A LA FIN DE CE FICHIER
```

ou 12 est la spécification de la ligne et P est l'instruction dans cet ordre.

I.4 L'insertion

Dans l'emploi normal du mot "insérer" nous entendons par exemple "insérer cette carte après cette autre carte". Pour utiliser l'instruction d'insertion nous précisons le N° de la ligne après laquelle la nouvelle ligne doit être insérée, suivi du Symbole I

⟨ligne⟩ I

Après la frappe de l'instruction suivie d'un "retour chariot" (↵), l'éditeur va choisir un numéro de ligne approprié et inviter à frapper la ligne en affichant ce numéro de ligne suivi d'un signe égal (=). Après que chaque ligne et le retour chariot ont été tapés, l'éditeur affiche un nouveau n° de ligne suivi d'un égal (=). Pour sortir du mode "insertion" taper un dièse (#). Un nouvel ordre peut alors être donné à l'éditeur.

Voici quelques exemples d'insertion :

```
#8I
 8.10 = VØICI UNE LIGNE INSEREE
 8.20 = AINSI QUE CELLE-CI
 8.30 = #11 I
11.10 = ENCØRE UNE LIGNE INSEREE
11.20 = # 6P
 6.00 = ABCDEFGHIJKL
```

Note :

L'éditeur peut renuméroter des lignes qui suivent le texte inséré ; ceci quand assez de lignes sont insérées et que leurs numéros recoupent ceux des lignes du texte original.

I .5 La suppression

Avec cette instruction on peut supprimer une ligne ou un bloc de lignes.

Pour supprimer une ligne indiquons la ⟨ligne⟩ à supprimer suivie de D

⟨ligne⟩ D

Lorsque vous faites le retour chariot, la ligne disparaît.

Pour supprimer plusieurs lignes, il nous faut indiquer, non seulement la première ligne à supprimer mais aussi la dernière que nous appellerons ligne cible et que nous indiquerons par ⟨cible⟩ Bien que l'éditeur permette plusieurs moyens de spécifier la ligne cible, voyons d'abord les deux plus simples :

1°) ⟨cible⟩ peut être le nombre de lignes à supprimer (en incluant la première et la dernière ligne du bloc)

2°) ⟨cible⟩ peut être un dièse (#) suivi du n° de la dernière ligne du bloc à supprimer.

Quelques exemples :
```
      3 (supprimer 3 lignes)
     26 (supprimer 26 lignes)
# 26 (supprimer jusqu'à la ligne 26.00)
```

La syntaxe de suppression d'un bloc est :

⟨ligne⟩ D ⟨cible⟩

ou ⟨ligne⟩ indique le N° de la première ligne
et ⟨cible⟩ l'étendue de la suppression.

Pour illustrer l'emploi de l'instruction de suppression supposons que nous avons un fichier comportant cinquante trois lignes numérotées de 1 à 53.

Avec les instructions :

```
# 15 D
# 24 D # 31
# 52 D 2
BØTTØM ØF FILE REACHED
#
```

Nous nous trouvons maintenant avec un fichier comportant les lignes 1 à 14, 16 à 23, 32 à 51. La première instruction a supprimé la ligne 15, la seconde les lignes 24 à 31, la troisième 2 lignes à partir de la ligne 52. Comme la dernière ligne du fichier a été supprimée, l'éditeur affiche :

FIN DU FICHIER ATTEINTE

Avant d'examiner de nouvelles instructions, nous devons étendre les définitions de ⟨ligne⟩ et⟨cible⟩.

I.6 La notion de ligne

Durant les opérations d'édition, l'éditeur garde trace de la ligne courante qui est généralement la dernière ligne affectée par une instruction.

Lors de l'appel du programme d'édition la ligne courante est la première ligne du fichier.

Si par exemple nous venons d'insérer trois lignes, entre les lignes 12.00 et 13.00, la ligne courante sera 12.30.

Après la suppression d'une ou plusieurs lignes, la ligne suivant immédiatement la dernière ligne supprimée devient la ligne courante (si la dernière ligne du fichier a été supprimée, c'est la nouvelle dernière ligne du fichier qui devient ligne courante).

Dans nos considérations précédentes, nous avons signalé qu'il fallait explicitement indiquer une ligne pour chaque instruction en spécifiant le numéro de la ligne intéressée.

Néanmoins, si aucune ligne particulière n'est mentionnée la ligne courante est prise par défaut.

Par exemple, en réponse à l'ordre

```
## D2
#
```

l'éditeur supprimera deux lignes à partir de la ligne courante : puisque dans notre exemple nous en étions à la ligne 6.00, les lignes 6.00 et 7.00 seront supprimées. Comme vous vous en rendrez compte à l'usage, la considération de la ligne courante en cas d'absence de ⟨ligne⟩ (par défaut) est extrêmement commode.

Après toutes ces manipulations notre fichier est devenu :

```
 1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE
 2.00 = L'EDITEUR DE TEXTE. BEAUCØUP
 3.00 = D'EXEMPLES NØUS PERMETTRØNT D'APPRENDRE
 4.00 = RAPIDEMENT ET FACILEMENT SES CARACTERISTIQUES
 5.00 = LES LIGNES SUIVANTES N'ØNT AUCUN SENS
 8.00 = TEST 1234
 8.10 = VØICI UNE LIGNE INSEREE
 8.20 = AINSI QUE CELLE-CI
 9.00 = L'EDITEUR EST AMUSANT A UTILISER !
10.00 = BBBBBBBB
11.00 =
11.10 = ENCØRE UNE LIGNE INSEREE
12.00 = NØUS VØICI A LA FIN DE CE FICHIER
13.00 = AU MØINS PØUR L'INSTANT
```

Nous avons vu que l'on peut spécifier une ligne par son numéro ; Par défaut, nous tombons sur la ligne courante. Il existe aussi plusieurs autres moyens de spécifier une ligne ou, en d'autres termes, de déplacer le "pointeur" vers une ligne concernée avant l'éxécution de l'instruction. On peut aussi spécifier la ligne par "+n" ou "-n" (n entier) pour dire nième ligne suivante ou nième ligne précédente. Deux autres indicateurs utiles sont "^" et "!" qui spécifient directement la première ligne ou début du fichier et la dernière ligne ou fin du fichier. Tous ces moyens d'indiquer une ligne sont illustrés dans l'exemple ci-dessus en conjugaison avec l'instruction PRINT.

```
## ∧P
      1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE
## + 3P
      4.00 = RAPIDEMENT ET FACILEMENT SES CARACTERISTIQUES
## ! P
      13.00 = AU MØINS PØUR L'INSTANT
##-? P
      11.10 = ENCØRE UNE LIGNE INSEREE
##
```

Parfois,une partie du contenu de la ligne est connu mais pas son numéro, ni sa position par rapport à la ligne courante. Dans ce cas le caractère "analyse du contenu"de l'éditeur permet de réagir.

La syntaxe devient alors :

/ ⟨chaîne de caractères⟩/

Les // délimitent la chaîne de caractèresdont on connait l'existence dans la ligne recherchée.

Dans ce cas l'éditeur recherche, à partir de la ligne courante la prochaine ligne qui contient cette chaîne.

Exemples :

/RENARD/

L'éditeur recherche la prochaine ligne qui contient la chaîne de caractère RENARD. Si la chaîne existe la ligne qui la contient devient la ligne courante sur laquelle l'opération demandée sera exécutée. Si la chaîne n'est pas trouvée la ligne courante ne change pas et le message "NØ SUCH LINE" apparait. Notons que le séparateur n'est pas forcément un "/", ce peut être une apostrophe (') ou un point virgule (;). Par exemple :

'A/B' envoie à la prochaine ligne contenant A/B.

On peut faire précéder la chaîne désignée par le signe moins "-" pour indiquer une ligne précédente contenant la chaîne recherchée. Quelques exemples sur notre fichier vont illustrer cette méhode de recherche d'une ligne.

```
# -/RAPIDEMENT/P

  4.00 = RAPIDEMENT ET FACILEMENT SES CARACTERISTIQUES

#;123;P

  8.00 = TEST 1234

# +"FIN"P

 12.00 = NØUS VØICI A LA FIN DE CE FICHIER

#
```

En résumé, une ligne peut être spécifiée par :

1) la ligne courante par défaut
2) le numéro de la ligne
3) "+N" soit la nième ligne suivant la ligne courante
4) "-N" soit la nième ligne précédant la ligne courante
5) / chaîne de caractères/ soit la première ligne contenant cette chaîne de caractèresà partir de la ligne courante
6)-/ chaîne de caractères: soit la première ligne contenant cette chaîne de caractères , rencontrée en remontant les lignes, à partir de la ligne courante.
7) "^" pour désigner la première ligne du fichier
8) "!" pour désigner la dernière ligne du fichier

I.6 bis - La notion de cible

Vous vous rappelez que ⟨cible⟩est utilisée avec certaines instructions pour indiquer le nombre de lignes affectées par l'opération en question. Nous avons déjà vu qu'une cible peut être spécifiée par :

1) Un entier n (au besoin négatif) pour indiquer le nombre de lignes qui sont affectées à partir de la ligne désignée.

Par exemple :

P3 pour imprimer trois lignes (à partir de la ligne courante)

2) ## numéro de ligne, pour indiquer le numéro de la dernière ligne affectée par la commande

Par exemple :

P##6 (impression depuis la ligne courante jusqu'à la ligne n° 6 comprise) Mais nous pouvons aussi utiliser :

3) par défaut, l'ordre est effectué sur la seule ligne désignée.

4) / chaîne de caractères / pour indiquer que la commande sera exécutée depuis la ligne désignée jusqu'à la prochaine ligne contenant cette chaîne de caractère.

5) -/ chaîne de caractère / même opération que (4) mais en remontant les lignes.

6) "^" pour désigner la première ligne du fichier

7) "!" pour désigner la dernière ligne du fichier

Avec cette meilleure compréhension de ⟨ligne⟩et ⟨cible⟩ nous pouvons maintenant examiner des instructions supplémentaires. L'instruction P (RINT) affiche une ou plusieurs lignes. Sa syntaxe est :

⟨ligne⟩ P ⟨cible⟩

Mais pour afficher une seule ligne, la spécification de ladite ligne suffit ; en d'autres termes, ⟨ligne⟩ et ⟨ligne⟩ P ont le même effet.

Revenons à notre fichier :

```
##2P
      2.00 = L'EDITEUR DE TEXTE. BEAUCØUP
##-1
      1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE
##P/APPRENDRE/
      1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE
      2.00 = L'EDITEUR DE TEXTE. BEAUCØUP
      3.00 = D'EXEMPLES NØUS PERMETTRØNT D'APPRENDRE
##! P-3
     13.00 = AU MØINS PØUR L'INSTANT
     12.00 = NØUS VØICI A LA FIN DE CE FICHIER
     11.10 = ENCØRE UNE LIGNE INSEREE
```

```
#-/BBB/P -/123/

      10.00 = BBBBBB
       9.00 = L'EDITEUR EST AMUSANT A UTILISER
       8.20 = AINSI QUE CELLE-CI
       8.10 = C'EST UNE LIGNE INSEREE
       8.00 = TEST 1234

## 12P !

      12.00 = NØUS VØICI A LA FIN DE CE FICHIER
      13.00 = AU MØINS PØUR L'INSTANT

##
```

I.7 L'instruction NEXT

Cette instruction sert principalement à déplacer le pointeur de ligne. Bien qu'elle puisse être utilisée autrement, c'est d'habitude avec la ligne par défaut.

La syntaxe est `N⟨cible⟩`

Cette instruction trouve la ligne indiquée par ⟨cible⟩, l'affiche,et en fait la ligne courante.

Voyons quelques exemples :

```
## ΛP

      1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE

##N

      2.00 = L'EDITEUR DE TEXTE. BEAUCØUP

##N6

      8.20 = AINSI QUE CELLE-CI

##N-2

      8.00 = TEST 1234

##
```

I .8 Remplacement ou insertion d'une seule ligne

La syntaxe de cette instruction est la suivante :

⟨ligne⟩ = ⟨texte⟩

-ligne est spécifiée par son numéro ; il s'agit de la ligne à remplacer ou à insérer ; par défaut c'est la ligne courante.

-texte est le texte de la ligne.

Reprenons notre exemple

```
## = REMPLACE LA LIGNE CØURANTE ICI
##5.25 CETTE LIGNE EST CREEE AVEC =
##
```

La première instruction remplace le premier texte de la ligne 8.00, ligne courante.

La deuxième instruction insère une nouvelle ligne de numéro 5.25.

I.9 Modification d'une chaîne de caractères

L'instruction C est utilisée pour remplacer une chaîne de caractères par une autre.

La syntaxe est la suivante :

⟨ligne⟩ C/ ⟨chaîne⟩1 / ⟨chaîne⟩2 / ⟨cible⟩ ⟨occurrence⟩

"/" est le séparateur pour séparer les deux chaînes de caractères : ⟨chaîne⟩1 est la chaîne à modifier , ⟨chaîne⟩2 la nouvelle chaîne.

cible détermine l'étendue de la modification.

⟨occurrence⟩ est l'occurrence de la chaîne 1 dans une ligne.
Si ⟨occurrence⟩ est 1 ou n'est pas spécifiée, seule la première occurrence de chaîne 1 dans toutes les lignes du bloc sera changée.
Si ⟨occurrence⟩ est 2 seule la deuxième occurrence de la chaîne 1 sera modifiée dans les lignes du bloc.
Si ⟨occurrence⟩ est * toutes les occurrences de la chaîne 1 seront modifiées dans toutes les lignes du bloc.

Exemple :

```
#4 C/ FACILEMENT / AISEMENT/

  4.00 = RAPIDEMENT ET AISEMENT SES CARACTERISTIQUES
#8.1 C/VØICI//
  8.10 = UNE LIGNE INSEREE

#-5C ; A ; $ ; SENS ; *
  3.00 = D'EXEMPLES NØUS PERMETTRØNT D'$PPRENDRE
  4.00 = R$PIDEMENT ET $ISEMENT SES C$R$TERISTIQUES
  5.00 = LES LIGNES SUIV$NTES N'ØNT AUCUN SENS

#12C/E/?3/-2.3

 12.00 = NØUS VØICI A LA FIN DE CE FICH?ER
 11.10 = ENCØRE UNE LIGN? INSEREE
```

Dans le premier exemple le mot "FACILEMENT" est remplacé par le mot "AISEMENT" .

Dans le deuxième exemple la locution "VØICI" est supprimée dans la ligne 8.10.

Dans le troisième exemple le pointeur de ligne remonte cinq lignes, donc jusqu'à la ligne 3.00 et change chaque occurrence de "A" en "$" dans toutes les lignes jusqu'à la ligne 5.00 qui contient la chaîne de caractères "SENS".

Le dernier exemple montre le changement de la troisième occurrence de "E" en "?" dans la ligne 12.00 et dans la ligne 11.00

I.10 Comment sortir de l'éditeur

Pour sortir de l'éditeur, plusieurs formes sont possibles : STØP, S, ou LØG. Toutes ramènent au FLEX.

I.11 Récapitulatif

Retournons à notre fichier pour illustrer l'effet de ces dernières instructions.

```
## ^ P !

   1.00 = VØICI UN EXEMPLE D'UTILISATIØN DE
   2.00 = L'EDITEUR DE TEXTE. BEAUCØUP
   3.00 = EXEMPLES NØUS PERMETTRØNT D'ØPPRENDRE
   4.00 = RØPIDEMENT ET ØISEMENT SES CØRØCTERISTIQUES
   5.00 = LES LIGNES SUIVØNTES N'ØNT ØUCUN SENS
   5.25 = CETTE LIGNE EST CREEE AVEC =
   8.00 = REMPLACE LA LIGNE CØURANTE ICI
   8.10 = UNE LIGNE INSEREE
   8.20 = AINSI QUE CELLE-CI
   9.00 = L'EDITEUR EST AMUSANT A UTILISER !
  10.00 = BBBB BBBB
  11.00 =
  11.10 = ENCØRE UNE LIGN? INSEREE
  12.00 = NØUS VØICI A LA FIN DE CE FICHI?R
  13.00 = AU MØINS POUR L'INSTANT

## 2C/E /E 6800 / 1 5

   2.00 = L'EDITEUR DE TEXTE 6800. BEAUCØUP

## / BBB /

  10.00 = BBBBBBBB
```

```
## - ; VØICI ; C  'E' XX ' !

      1.00 = VØICI UN XXXEMPLE D'UTILISATIØN DE
      2.00 = L'XXDITEUR DE TEXTE 6800. BEAUCØUP
      3.00 = D'XXXEMPLES NOUS PERMETTRØNT D'$PPRENDRE
      4.00 = R$PIDXXMENT ET $ISEMENT SES C$R$CTERISTIQUES
      5.00 = LXXS LIGNES SUIV$NTES N'ØNT $UCUN SENS
      5.25 = CXXTE LIGNE EST CREEE AVEC =
      8.00 = RXXMPLACER ICI LA LIGNE PØINTEE
      8.10 = UNXX LIGNE INSEREE
      9.00 = L'XXDITEUR EST AMUSANT A UTILISER
     11.10 = XXCØRE UNE LIGNE INSEREE
     12.00 = NØUS VØICI A LA FIN DXX CE FICHI?R

## N-3
     10.00 = BBBBBBBB

## -1 I
      9.10 = TEST-TEST-TEST
      9.20 = 1234567890
      9.30 = ## D !
      BØTTØM ØF FILE REACHED

## I D !
      BØTTØM ØF FILE REACHED

## I P !

## S
```

Ceci n'était qu'une introduction à l'éditeur de texte. Le chapitre suivant décrit chaque instruction en détail avec des exemples. Il est important de lire et étudier le manuel complètement pour parfaitement comprendre l'éditeur et en utiliser pleinement toutes les possibilités.

CHAPITRE II

LES COMMANDES DE L'EDITEUR

Ce chapitre décrit plus précisément tous les ordres d'édition, l'utilisation des aspects particuliers.

La lecture de l'introduction vous a donné une vue générale sur les possibilités de l'éditeur rendant ainsi plus compréhensibles les descriptions détaillées.

Avant d'examiner la description complète des pseudo-instructions de l'éditeur, nous aborderons quelques points généraux.

II.1 Considérations générales

II.1.1 L'utilisation des chaînes

Plusieurs pseudo-instructions de l'éditeur utilisent des chaînes de caractères comme arguments. Ces chaînes remplacent ou se lient à des chaînes du texte sur lequel l'utilisateur travaille.

Une chaîne argument constituée de caractères commence et finit par un séparateur.

Les séparateurs ne sont pas considérés comme faisant partie de la chaîne pour pouvoir être utilisés dans les opérations de remplacement ou de liaison.

Bien que les séparateurs dans les descriptions suivantes soient représentés fréquemment par des slashs, "/", tout caractère non alphanumérique autre que le blanc peut être utilisé comme séparateur :

* / () ß =, . : etc...

Notez que les caractères suivants ne peuvent pas être utilisés pour entourer une chaîne à moins d'être précédés par un signe plus "+" ou moins "-" qui signale que la cible est au-dessus de la ligne courante :

- "^" qui marque la première ligne du fichier.
- "!" qui marque la dernière ligne du fichier.
- le caractère LINØ,##, utilisé pour signaler les numéros de lignes.

Le caractère séparateur est redéfini à chaque nouvelle demande par sa présence avant une chaîne. Si deux chaînes existent dans une instruction (comme dans l'instruction CHANGE) le même caractère séparateur doit être utilisé pour chaque chaîne.

Toutes les instructions de l'éditeur utilisent l'indication de ligne précédant l'instruction pour positionner le pointeur avant toute exécution de l'instruction.

Le paramètre ⟨ligne⟩ peut bien sûr être absent laissant le pointeur à sa position actuelle.

Tous les caractères suivants peuvent être utilisés comme référence de ligne.

1) Tout entier	Référence à un numéro de ligne donné
2) +N	Signifie la nième ligne suivante
3) -N	Signifie la nième ligne précédente
4) / chaine de caractères /	Renvoie à la prochaine ligne du fichier qui contient la chaine de caractères indiquée
5) -/ chaine de caractères /	Renvoie à la dernière ligne précédente qui contient la chaine de caractères indiquée
6) ^	Marque la première ligne du fichier
7) !	Marque la dernière ligne du fichier
8) absence	par défaut la ligne pointée.

La plupart des instructions de l'éditeur requièrent une spécification de cible : ⟨cible⟩. Ce pour indiquer à l'éditeur de traiter toutes les lignes, de la courante à celle spécifiée par ⟨cible⟩
Par défaut, ⟨cible⟩ est pris à 1 ; en conséquence seule la ligne courante sera concernée par l'instruction.
Tous les caractères suivants peuvent être utilisés comme référence de cible :

1)un entier N	indique que n lignes sont concernées par l'opération (à partir de la courante).
2)## N	indique le N° de la dernière ligne concernée.
3)/ chaîne /	repère la première ligne (après la courante bien sûr) contenant la chaîne en question.
4)- / chaîne /	repère la première ligne au-dessus de la ligne courante contenant la chaîne en question.
5) ^	toutes les lignes sont concernées jusqu'à la première.
6)!	toutes les lignes sont concernées jusqu'à la dernière.
7)+ ou - N	indique que n lignes sont concernées par l'opération et dans quelle direction par rapport à la ligne courante.
8)défaut	la cible est 1 et seule la ligne courante est affectée.

Comme nous l'avons vu, la forme ⟨cible⟩ est utilisée pour indiquer la gamme de lignes auxquelles l'instruction s'applique ; l'instruction s'appliquera à chaque ligne à partir de la ligne spécifiée par ⟨ligne⟩ et continuera jusqu'à atteindre la cible.

Si une chaîne ⟨cible⟩ est spécifiée, l'instruction s'appliquera à toutes les lignes de texte jusqu'à atteindre une ligne contenant la chaîne. La recherche se fait en descendant le fichier sauf si la cible est précédée d'un signe moins, auquel cas la recherche (et le traitement) se font en remontant le fichier.

Les cibles peuvent être également précédées d'un signe + (recherche descendante). Si un numéro de ligne cible est spécifié, le traitement commence à ⟨ligne⟩ et s'achève à la ligne cible. Voici quelques exemples de ⟨cible⟩:

2

+ 10

- 3

/ CHAINE /

+ / CIBLE CHAINE /

- / DEPLACEMENT ARRIERE A UNE CHAINE /

+ * TØUT SEPARATEUR CØNVIENT PØUR UNE CHAINE *

+ + MEME LE SIGNE PLUS CØNVIENT +

23.00

II.1.2 Spécification d'un numéro de colonne

Toute indication "/ chaîne /" peut être suivie d'un numéro de colonne immédiatement après le second séparateur pour indiquer que la chaîne en question doit commencer dans la colonne spécifiée pour être trouvée.

Si la colonne spécifiée n'est pas dans l'ensemble de la zone affectée, la demande sera ignorée.

Voici quelques exemples :

/ IDENTIFICATEUR/11
/ PRØGRAMME /77
* ETIQUETTE * 2
$ CØMMENTAIRE $ 30

II.1.3 Caractère "ordre répété"

Un caractère "ordre répété" a été introduit dans l'éditeur pour permettre de répéter exactement le dernier ordre dans le tampon d'entrée.

Si un ordre entraîne une erreur ou change le contenu du tampon d'entrée, le commentaire ILLEGAL CØMMAND (ordre illégal) sera affiché à chaque utilisation du caractère de répétition jusqu'à ce qu'un autre ordre répétable soit donné.

Le caractère de répétition est CTRL R (12_{hex} ASCII).

Quelques exemples d'ordres qu'il peut être utile de répéter :

PRINT 15	pour imprimer d'un coup un écran de lignes.
NEXT	pour dérouler pas à pas un fichier avec une seule touche.
ΛCØ !!	pour remplir rapidement l'espace de travail.
FIND/Chaîne de caractères/	si la première chaîne trouvée n'est pas la bonne.

II.1.4 Utilisation du caractère de fin de ligne (EØL)

L'éditeur comprend un caractère EØL (End of Line) pour permettre des ordres multiples dans une seule ligne.

Les ordres INSERT et ØVERLAY sont des exceptions car ils ne peuvent pas être suivis par d'autres ordres.

Le caractère EØL peut être changé de façon interactive par l'ordre SET.

Exemple de l'utilisation de EØL (avec "$" comme caractère EØL):

```
^ D 2 $ P 10 $ T
```

La séquence supprimera les deux premières lignes du fichier, imprimera les dix lignes suivantes et finalement renverra le pointeur au début du fichier.

II.1.5 Utilisation du tabulateur

L'utilisateur peut spécifier interactivement un caractère de tabulation, et ceci jusqu'à 20 arrêts de tabulation par ligne.

Le caractère de tabulation peut être inséré dans une ligne où il sera pris en considération après envoi du caractère de fin de ligne.

Si les arrêts ou le caractère de tabulation n'ont pas été spécifiés auparavant, mais qu'un caractère a servi de tabulateur dans tout le fichier, il peut encore être pris en considération en l'affectant à TAB, plaçant les arrêts et utilisant l'ordre EXPAND sur le fichier.

Notons que si un caractère de tabulation a été spécifié il sera automatiquement pris en considération par les ordres d'insertion INSERT ou de remplacement REMPLACE.

Si toutefois un caractère de tabulation est ajouté à l'aide des ordres CHANGE, APPEND ou ØVERLAY il ne sera pas pris en considération jusqu'à ce qu'un ordre EXPAND soit appliqué à la ligne contenant ce caractère de tabulation.

II.2. Instructions de l'éditeur

Il existe quatre groupes d'instructions :

- les instructions d'environnement
- les instructions systèmes
- les déplacements du pointeur de ligne
- les instructions d'édition

Une description complète de toutes les instructions de chaque groupe est donnée ci-après.

Dans les descriptions suivantes, les caractères compris entre parenthèses ( ) sont optionnels et peuvent être omis.

Un slash (/) est utilisé pour séparer les options.

II.2.1 Les instructions d'environnement

H (EADER) (<NØMBRE>)

Signification :

Cette instruction permet d'afficher la ligne suivante :

12345678901234567890.....

sur 80 colonnes ou le nombre de colonnes précisé, pour faciliter la mise en page du texte en indiquant les colonnes dix par dix.

Les colonnes qui contiennent un arrêt de tabulation ont un caractère "-" à la place du chiffre.

Si le nombre de colonnes est précisé, il sera ensuite gardé comme valeur par défaut, lors de la frappe de H.

Exemples :

HEADER 72	affiche une ligne de 72 colonnes
H 30	affiche une ligne de 30 colonnes.

NU(MBERS) (ØFF/ØN)

Signification :

Le drapeau numéro de ligne est arboré ou amené.

Si le drapeau est amené les numéros de ligne ne seront jamais imprimés.

Si ni ØFF ni ØN sont spécifiés, le drapeau sera inversé par rapport à l'état actuel.

Exemples :

NUMBERS ØFF	n'imprimera pas les numéros de ligne
NU ØN	les numéros de ligne seront imprimés
NU	bascule de ØN sur ØFF et de ØFF sur ØN

REN(UMBER)

Signification :

La pseudo-instruction renumérote toutes les lignes dans le fichier en cours d'édition.

La première ligne du fichier renuméroté sera 1.00 ; le pas est de 1.

La ligne courante avant l'ordre sera encore la ligne courante après l'ordre (mais son numéro sera probablement différent).

Exemples :

RENUMBER	renumérote les lignes dans le fichier de travail en cours
REN	" "

SET⟨nom⟩ = '⟨caractère⟩'

Signification:

SET est utilisé pour définir certains caractères ou symboles spéciaux. Les noms peuvent être :

TAB Caractère de tabulation

FILL Caractère de remplissage de la tabulation

EØL Caractère de fin de ligne qui peut être utilisé pour séparer plusieurs ordres dans une seule ligne

LINØ Caractère de drapeau de numéro de ligne qui est utilisé pour indiquer qu'une cible est un numéro de ligne spécifique.

Les valeurs implicites sont :

TAB et EØL	absence
FILL	'espace'
LINØ	' ## '

Exemples:

SET TAB='/'	Le caractère de tabulation est un slash
SET TAB=	Invalide la tabulation
SET FILL = ' '	Le caractère de remplissage de la tabulation est un espace
SET EØL = '$'	Le caractère de fin de ligne est $
SET LINØ= ' *'	Le caractère de drapeau de numéro de ligne est *

TAB (<colonnes>)

Signification :

TAB est utilisé pour placer les arrêts de tabulation : les précédents arrêts sont effacés.

Si aucune colonne n'est spécifiée, la seule action est d'effacer tous les arrêts de tabulation.

Tous les caractères de tabulation qui sont au-delà du dernier arrêt de tabulation sont laissés dans le texte.

Le nombre maximum d'arrêtsde tabulation est vingt.

Les arrêts de tabulation doivent être saisis en ordre ascendant.

Exemples :

TAB 11, 18, 30	met des arrêts de tabulation aux colonnes 11, 18, 30
TAB 7, 72	met des arrêts de tabulation pour un programme FORTRAN
TAB	nettoie tous les arrêts de tabulation

V (ERIFY) (ØN/ØFF)

Signification :

Le drapeau de vérification est arboré ou amené.

Ce drapeau est utilisé dans les instructions CHANGE et NEXT (et plusieurs autres) pour visualiser leurs résultats.

Si ni "ØN" ni "ØFF" ne sont spécifiés, le drapeau sera inversé par rapport à son état initial.

On contrôle l'état de VERIFY en constatant si dans les instructions concernées les résultats s'affichent ou non.

Exemples :

VERIFY ØFF	ote la vérification
V ØN	met la vérification
V	inverse ØN et ØFF

X

Signification :

"X" est l'ordre commande du curseur.

A chaque entrée de cet ordre, l'éditeur renvoie la chaîne de six caractères spéciaux précédemment définie.

Ceci est utile pour des applications spéciales.

Exemple :

X sort la chaîne de commande du curseur

Z (ØNE) (C1,C2)

Signification :

ZØNE est utilisé pour restreindre toutes les recherches de sous-chaines (FIND, CHANGE, cible , etc..) seulement entre les colonnes C1 et C2 (comprises).

Toutes les sous-chaînes commençant en dehors de ces colonnes ne seront pas détectées.

Si C1 et C2 ne sont pas spécifiés, les zones seront prises par défaut (colonnes 1 et 136).

Exemple :

ZØNE 11, 29	restreint les recherches entre les colonnes 11 à 29.
ZØNE	recherche entre les colonnes 1 et 136.

II.2.2 Les pseudo-instructions "système"

LØG

Signification :

sortie de l'Editeur

S (TØP)

Signification :

même que LØG

II.2.3 Déplacements du pointeur de ligne

B(ØTTØM)

Signification :

BØTTØM déplace le pointeur jusqu'à la dernière ligne du fichier, qui devient la ligne courante.

Exemple:

BØTTØM	La dernière ligne du fichier devient ligne pointée
B	" "

F(IND) ⟨cible⟩ (occurrence)

Signification :

FIND déplace le pointeur de la ligne courante jusqu'à la ligne spécifiée dans la cible, qui devient alors la ligne courante.

Si le drapeau de vérification VERIFY est arboré (voir VERIFY), la ligne sera imprimée. Si l' occurrence est spécifiée (un entier ou un astérisque) la commande sera répétée le nombre de fois indiqué. Si occurrence est un entier, frapper un blanc entre le second séparateur de la chaine et l' occurrence (cf exemple 2) pour éviter la confusion avec un suffixe de chaine (cf exemple 3).

Un astérisque entraine la recherche jusqu'à la dernière ou la première ligne du fichier (cf exemple 4-5).

Si la cible n'est pas trouvée, le pointeur ne sera pas déplacé.

Exemple :

FIND/RENARD/	Recherche la prochaine ligne qui contient la chaine "RENARD"
F/GØUPIL/ 3	Recherche les trois prochaines lignes qui contiennent la chaine /GØUPIL/
F/GØUPIL/30	Recherche dans la ligne pointée la chaine GØUPIL commençant en colonne 30
F/GØUPIL/ *	Recherche toutes les occurrences de la chaine "GØUPIL" jusqu'à la fin du fichier.
F/GØUPIL/7 *	Recherche toutes les lignes précédentes qui ont le mot "GØUPIL" commençant en colonne 7 jusqu'au début du fichier.

N(EXT) (⟨cible⟩ (⟨occurence⟩))

Signification :

La ligne spécifiée par la cible devient la ligne courante .
Si le drapeau de vérification VERIFY est arboré,cette ligne sera imprimée.
Si l' occurrence est spécifiée (nombre entier),elle indique laquelle parmi les lignes qui contiennent la cible devient la ligne courante.

Si la cible n'est pas trouvée, le pointeur de ligne sera déplacé à la fin du fichier (ou au début si le déplacement est ascendant).
Si aucune cible n'est mentionnée la ligne suivante devient la ligne courante.

Exemples :

NEXT 5	Le pointeur de ligne est déplacé à la cinquième ligne suivante
N	La ligne suivante devient ligne courante.
N-10	La dixième ligne précédente devient ligne courante.
N/GØUPIL/	La prochaine ligne qui contient la chaine "GØUPIL" devient ligne courante.
N/GØUPIL/ 3	La troisième ligne contenant la chaine "GØUPIL" devient ligne pointée.

T(ØP)

Signification :

La première ligne du fichier devient la ligne courante.

Exemple :

TØP	Met le pointeur de ligne à la première ligne du fichier.

II.2.4. Les instructions d'édition

A (PPEND)/ ⟨chaîne⟩ /(⟨cible⟩)

Signification :

APPEND lie la chaine spécifiée au dernier caractère de la ligne courante et des lignes suivantes jusqu'à ce que la cible soit atteinte. Si la chaine de caractères est postfixée par un numéro de colonne, le début de la chaine est inséré à la colonne spécifiée et non à la fin de la ligne. Tous les caractères qui suivaient la colonne spécifiée dans l'ancienne ligne sont effacés.

Exemples :

APPEND/./	Ajoute un point à la fin de la ligne courante
A ¥HELLØ*2	Ajoute le mot "HELLØ" à la fin de la ligne courante et à la fin de la ligne suivante.
A/GØUPIL/73*FIN*7	Ajoute le mot "GØUPIL" à partir de la ligne courante et des lignes suivantes jusqu'à trouver une ligne contenant la chaine FIN commençant à la 7ème colonne.

C(HANGE)/⟨chaine 1⟩/⟨chaine 2⟩/(⟨cible⟩ (⟨occurrence⟩))

Signification :

Remplace la chaine de caractères1 par la chaine de caractères2

Si aucune cible n'est spécifiée, seule la ligne courante est modifiée. Les slashes "/" représentent tout caractère séparateur autre qu'un blanc.

⟨occurrence⟩ spécifie quelle occurrence de la chaine 1 doit être remplacée dans chaque ligne : c'est soit un entier soit un astérisque ('*') qui signifie que toutes les occurrences de la chaine 1 doivent être remplacées par la chaine 2 . Par défaut seule la première occurrence sera modifiée.

NOTE :

Si l'occurrence est spécifiée et si la modification ne doit avoir lieu que sur la ligne courante alors la cible doit être un 1 (un).

Exemples:

CHANGE/CELUI/CELLE/	Remplace le premier "CELUI" de la ligne courante par "CELLE".
C/A/B/1*	Change tous les "A" par "B" dans la ligne courante.

C/PREMIER/DERNIER/10	Change le premier "PREMIER" par "DERNIER" dans la ligne courante et dans les neuf lignes suivantes.
C/NØUVEAU/ANCIEN/UN ØBJET/	Change le premier "NØUVEAU" par "ANCIEN" dans chaque ligne jusqu'à ce qu'une ligne contenant la chaine "UN OBJET" soit rencontrée.
C,A,, -10*	Enlève tous les "A" dans la ligne pointée et dans les neuf lignes précédentes.
C*HELLØ*	Efface la chaine de caractères "HELLØ" dans la ligne courante.

CØ(PY) (⟨cible-destination⟩ (⟨cible-gamme⟩))

Signification :

La ligne courante et les lignes suivantes jusqu'à la cible-gamme sont copiées après la cible-destination.

La destination par défaut est 1 : la ligne courante est alors copiée après la ligne suivante.

La gamme par défaut est 1 : dans ce cas seule la ligne courante est copiée.

Après l'éxécution de la pseudo-instruction, le pointeur de ligne se place à la nouvelle position de la dernière ligne copiée. Il peut y avoir renumérotation de certaines lignes sans qu'un message de renumérotation apparaisse.

Exemples :

CØ//18	Copie de la ligne courante après la ligne 18.
CØ//34	Copie de la ligne suivante et les trois lignes suivantes après la ligne 3.
CØ/ØRDRE/+/RANG	Copie de la ligne courante et des lignes suivantes jusqu'à la ligne contenant la chaine de caractère "RANG" après la prochaine ligne qui contient la chaine de caractère "RANG".

D(ELETE) (<cible>)

Signification :

La ligne courante (et les lignes suivantes jusqu'à la ligne de cible sont détruites). Après l'éxécution de l'instruction. la ligne courante sera la ligne suivant la dernière ligne détruite)

Exemple :

DELETE 5	Suppression de cinq lignes (la ligne courante etles quatre lignes suivantes)
D	Suppression de la ligne courante
D/CHAINE/	Suppression de la ligne courantejusqu'à la prochaine ligne qui contient la chaîne de caractères "CHAINE".

EXP (AND) (< cible >)

Signification :

Le caractère de tabulation spécifié est étendu à toutes les lignes depuis la ligne courante jusqu'à la ligne cible.

Les caractères de tabulation sont normalement étendus lorsque des lignes sont insérées dans le fichier mais cette commande est d'une grande utilité lorsqu'on a oublié de définir un caractère de tabulation.

Exemple :

EXPAND 100	extension à 100 lignes à partir de la ligne courante.
EXP	extension à la ligne courante.

(I)NSERT

Signification :

L'éditeur entre les données écrites dans la mémoire tampon, en indiquant les numéros de lignes et insère les lignes après la ligne courante (si ces numéros ne sont plus valables voir la commande NUMBERS).

La saisie dans le tampon continue jusqu'à ce que le caractère d'arrêt (le dièse "#") soit rencontré au début d'une ligne (dans la première colonne).

Les caractères qui suivent ce caractère d'arrêt sont traités comme une commande d'édition.

L'éditeur essaiera de choisir le pas pour pouvoir insérer 10 lignes au moins. Si ce n'est pas possible, il prendra le plus petit pas possible.

Le pointeur de ligne sera positionné à la dernière ligne insérée.

L'éditeur peut renuméroter les lignes de textes qui suivent le texte inséré si les numéros des lignes insérées chevauchent les numéros de lignes existantes dans le fichier.

Exemple :

INSERT	Accepte la saisie de nouvelle ligne
ou I	après la ligne pointée.

I(NSERT) ⟨ texte ⟩

Signification :

Le texte qui suit immédiatement le caractère de séparation (un blanc) après la commande (I) sera écrit sur une ligne après la ligne courante,cette nouvelle ligne devient alors la ligne courante.

L'éditeur peut renuméroter les lignes de textes qui suivent le texte inséré si le numéro de la ligne insérée coïncide avec le numéro de lignes existantes dans le fichier.

Exemple :

```
2 = FIN
3 =
I   RAJØUTER
```

MØ(VE) (<destination>(<cible-gamme>))

Signification :

La ligne courante et les lignes suivantes jusqu'à la ligne cible (gamme)sont récrites. après la ligne destination.

La destination implicite est 1, c'est-à-dire que la ligne courante est récrite au-dessous de la ligne suivante du fichier. La gamme implicite est 1 : une seule ligne est récrite.

Après exécution de la commande le pointeur de ligne sera placé à la dernière ligne récrite. Quelques lignes peuvent être renumérotées après un mouvement sans qu'un message de renumérotation soit exprimé.

Exemple :

MØVE 3	Récrit la ligne courante trois lignes plus bas
MØVE-/PRØGRAMME/5	Récrit la ligne courante et les quatre suivantes après la première ligne précédente du fichier qui contient la chaine de caractère "PRØGRAMME"
MØ 10 - 5	Récrit la ligne courante et les quatre lignes précédentes au-dessous de la ligne 10 du fichier.

Ø(VERLAY) (< séparateur >)

Signification :

La ligne courante est imprimée et une ligne de saisie est acceptée ensuite sur le terminal : la ligne de recouvrement.

Cette ligne est positionnée directement au-dessous de la ligne courante.

Chaque caractère de recouvrement qui est différent du caractère séparateur (un blanc par défaut) remplacera le caractère correspondant dans la ligne courante.

La ligne modifiée sera imprimée si VERIFY est ØN.

Exemple :

```
## ØVERLAY
                    =   BØICI LA PØNNE VIGNE
   ØVERLAY              V       B     L
                    =   VØICI LA BØNNE LIGNE
```

Ø(VERLAY)〈 d 〉〈texte〉

Signification :

Cette instruction est semblable à la commande de recouvrement précédente ØVERLAY mais la ligne courante n'est pas imprimée, le texte qui suit le caractère délimiteur est considéré comme recouvrement supplémentaire.

Exemple :

```
ØVERLAY-------LA------------ ECRITE

25.00 = VØILA LA BØNNE LIGNE ECRITE
```

P(RINT) (<cible>)

Signification :

Impression de la ligne courante et des lignes suivantes jusqu'à la ligne cible. Par défaut, seule la ligne courante est imprimée.

Exemples :

P	Imprime la ligne pointée
PRINT 5	Imprime la ligne pointée et les quatre suivantes
P-10	Imprime la ligne pointée et les neuf précédentes
100P	Imprime la ligne 100
PRINT*CHAINE*	Imprime toutes lignes précédentes jusqu'à la première ligne contenant la chaine de caractère CHAINE.

NOTE : Un retour chariot imprime la ligne courante.

R(EPLACE) (< cible>)

Signification :

Suppression de la ligne courante jusqu'à la ligne cible.

L'éditeur se met en mode saisie dans mémoire tampon ; les nouvelles lignes sont mises dans l'espace libéré. Il n'est pas obligatoire de rentrer autant de lignes que de lignes supprimées. Les numéros deslignes insérées peuvent être différents de ceux des lignes supprimées. Le pointeur de ligne sera positionné à la dernière ligne saisie.

Par défaut seule la ligne courante est supprimée.

Exemple :

R	Remplace la ligne courante.
REPLACE 10	Remplace 10 lignes (ligne courante comprise)
R/GØUPIL/	Remplace toutes les lignes à partir de la ligne courante jusqu'à la ligne qui contient la chaîne de caractères"GØUPIL".

= ⟨texte⟩

Signification :

L'instruction"=" remplace la ligne pointée par le texte exprimé.

Le texte de remplacement commence avec le premier caractère qui suit le signe =.

Le pointeur de ligne ne bouge pas.

Exemple :

```
=   C'EST UN NØUVEAU TEXTE
```

CHAPITRE III

MANIPULATION DE FICHIERS

Ce chapitre décrit comment créer et accéder à des fichiers de textes, comment manipuler ces fichiers et comment sortir de l'éditeur.

III-1 - Description générale

La syntaxe générale de la commande d'exécution de l'éditeur est :

EDIT, fichier 1 (, fichier 2)

L'extension implicite est .TXT et le lecteur de disque implicite est le lecteur du disque de travail.

Si on spécifie seulement 'fichier 1' et que le nom de ce fichier n'existe pas sur le disque, un nouveau fichier est créé à ce nom.

Lors de la création de nouveaux fichiers, l'éditeur envoie le message suivant :

NEW FILE

1.00 =

Si la ligne de commande EDIT ne spécifie que 'fichier 1' et que ce nom existe déjà sur le disque, ce fichier sera chargé dans le tampon de saisie et l'éditeur imprimera un '*' sur le terminal, signifiant qu'il est prêt à accepter des ordres de l'utilisateur.

Quand le processus d'édition est terminé, le fichier original garde son nom mais il a maintenant comme extension BAK pour "back up".

Si un fichier .BAK du même nom existe déjà l'éditeur demande :

DELETE BACK UP FILE ? (Y - N)

Détruire le fichier secondaire (ØUI ou NØN).

Une réponse positive (Y) (ØUI) détruit l'ancien fichier .BAK est en crée un nouveau. Une autre réponse renvoie à FLEX.
Le nouveau fichier aura le même nom que le précédent y compris son extension.

La dernier forme de EDIT permet d'assigner au nouveau fichier un nom spécifique différent de celui du fichier qui conserve alors le sien.

Notons que lorsque on édite un fichier existant, un nouveau fichier est toujours créé, l'original restant intact même si son nom peut être changé.

Plusieurs exemples nous aideront à clarifier cette syntaxe :

1) Si vous voulez créer un fichier appelé TEST.TXT (aucun fichier de ce nom n'existe encore sur le disque). Il faut taper la commande suivante : EDIT, TEST

L'éditeur répond :
NEW FILE
1.00=

Il est prêt à accepter des lignes de texte.

2) Vous avez créé un fichier TEST.TXT et vous voulez l'éditer pour le modifier.

Il faut taper la commande suivante :

EDIT, TEST

Le fichier TEST.TXT est alors chargé en mémoire et l'éditeur est prêt à accepter des ordres.

A la fin de la modification,la main renvoyée au FLEX, le premier fichier TEST.TXT s'appelle maintenant (mais son contenu reste inchangé) TEST.BAK et le fichier modifié s'appelle TEST.TXT.

Si de nouvelles modifications doivent être faites la même procédure peut être employée :

EDIT, TEST

L'éditeur vous demande alors si vous voulez détruire le fichier "TEST.BAK" ; si oui, la même procédure s'applique, un nouveau fichier .BAK sera créé prenant la version en cours avant les nouvelles modifications et ainsi de suite...

3) Si vous voulez éditer un fichier mais lui donner un nouveau nom vous devez taper la commande suivante :

EDIT, TEST,TEST2

Le fichier TEST.TXT sera chargé en mémoire mais le nouveau fichier s'appelera TEST 2.TXT. Cette forme de commande est aussi utilisée quand il s'agit d'éditer un fichier à partir d'un lecteur et charger le nouveau fichier sur un autre lecteur, par exemple :

EDIT, 0.TEST, 1.TEST

Le fichier TEST.TXT du lecteur 0 sera édité et le nouveau fichier TEST.TXT sera copié sur le lecteur 1 (mais il ne faut pas que TEST.TXT existe déjà sur le lecteur 1)

Une fois dans l'éditeur, toutes les commandes d'édition s'appliquent à la version FLEX de l'éditeur à part quelques exceptions expliquées ci-dessous.

III-2 - Sortie de l'éditeur

Il faut utiliser la commande STØP ou "S". On peut aussi utiliser la commande LØG. Après que LØG, STØP ou S ont été frappés, l'éditeur terminera automatiquement le transfert de l'ancien vers le nouveau fichier disque. Pour l'édition d'un fichier long : la manipulation doit avoir le temps de s'effectuer et les +++ du FLEX n'apparaitront qu'à la fin de l'opération.

III-3 - L'ordre "NEW"

C'est une commande additionnelle de la version FLEX de l'éditeur. Cette commande aide l'usager dans la manipulation de fichiers de textes plus importants que ceux que peut contenir la mémoire centrale. Lors de l'édition d'un tel fichier l'éditeur ne chargera en mémoire que ce qu'il peut.

La commande NEW dit à l'éditeur que vous avez fini l'édition de cette partie et que voulez en charger le morceau suivant.

Elle agit de la façon suivante : après avoir frappé NEW, toute la partie du fichier contenu dans la mémoire tampon jusqu'à la dernière ligne pointée est rangée sur le fichier disque .BAK sauf la dernière ligne pointée. Ensuite la plus grande partie possible du fichier original qui n'a pas encore été lue est copiée dans la mémoire tampon. La main sera rendue à l'éditeur et la dernière ligne pointée deviendra la première ligne pointée de cette nouvelle partie.

La commande NEW peut être utilisée aussi souvent que nécessaire mais il faut se souvenir que lorsqu'une portion de fichier a été rechargée sur disque elle n'est plus accessible par l'éditeur. Car il ne peut travailler que sur le texte chargé dans la mémoire tampon ; en conséquence des commandes globales comme CHANGE ou FIND ne peuvent agir que dans le cadre du texte chargé dans la mémoire tampon et non sur le fichier complet, à moins, bien sûr qu'il ne soit compris tout entier dans la mémoire tampon.

L'ordre NEW peut aussi servir lors de la création d'un fichier ; il est en effet possible d'emplir complètement le tampon (l'éditeur délivre alors le message NØT ENØUGH RØØM - pas assez de place).

Frapper alors NEW et tout le début du fichier jusqu'à la ligne courante sera rangé sur disque, libérant ainsi l'espace dans la mémoire tampon. Il n'y aura aucune lecture à partir du disque puisque le fichier vient d'être créé.

III-4 - Taille de la mémoire tampon

Le volume disponible en mémoire tampon est directement proportionnel au volume de mémoire total de l'ordinateur.

Plus la mémoire totale est grande plus le tampon de l'éditeur est important.

L'éditeur s'adapte automatiquement à la taille de la mémoire.

Par exemple dans un GOUPIL 2 en configuration 48Ko, il reste environ 34Ko pour la mémoire tampon (pour le texte donc sur lequel l'utilisateur travaille).

De même, il reste 42Ko dans un GOUPIL 2 en configuration 56Ko.

ANNEXE

1. Les caractères 'système'

Le caractère prêt (prompt) est rangé à l'adresse 0528_{16} ; c'est un dièse # (23_{16} ASCII).

Le caractère de suppression de ligne est celui du FLEX (touche CTRLX) de même que le caractère de suppression du dernier caractère affiché (touche CTRL H ou ←).

La cloche retentit quand le tampon d'entrée est saturé (plus de 136 caractères frappés par ligne).

Le caractère de répétition d'ordre est rangé à l'adresse 0530_{16} ; c'est un CTRL R (12_{16} ASCII) (à ne pas confondre avec la touche REPEAT)

2. Caractéristiques du système

Le plus grand numéro de ligne est 9999.99; si plus de 9999 sont frappées, le compteur de lignes revient à 0 ; en conséquence l'éditeur ne peut être utilisé avec des fichiers de 10000 lignes ou plus (ceci n'est pas vraiment une limitation puisque 10000 lignes vierges occupent 40 Ko de mémoire !).

Quand un numéro de ligne est inférieur à 1, il est nécessaire de faire précéder le point d'un 0 : 0.10 et non .10 par exemple. Ceci pour que le point ne soit pas pris pour un séparateur.

Le tampon d'entrée accepte 136 caractères ; au-delà les caractères sont ignorés ; pour sortir de la ligne il faut revenir (CTRL H ou ←) avant de faire un "retour chariot".

Il y a plusieurs façons d'insérer une ligne au début d'un fichier qui contient déjà une ligne 0.01. L'une est de renuméroter le fichier, 0.01 devenant 1.00 et permettant l'insertion d'une ligne 0.10. Une autre façon de faire est de frapper "INSERT - <x> " où x est le nombre de ligne qui renvoie une ligne au dessus du fichier. Si par exemple la ligne courante est 0.01, la commande "INSERT-1" permet d'insérer avant cette ligne.

3. Erreurs

Les erreurs, qui ne peuvent se produire que rarement, entraînent l'affichage de messages d'erreur très simples imprimés en clair en langue anglaise.

LISTE DES COMMANDES ET ORDRES DE L'EDITEUR DE TEXTE

Spécification	Signification	Page
A(PPEND)	Ajoute des chaines en fin des lignes	36
B(ØTTØM)	Déplace le pointeur jusqu'à la dernière ligne du fichier	32
C(HANGE)	Remplace une chaine par une autre	37
CØ(PY)	Copie des lignes à un endroit du texte	39
CTRL X	Efface la ligne courante	2
CTRL H ou ←	Efface le dernier caractère	2
CTRL R	Caractère de répétition	19
D(ELETE)	Supprime des lignes	40
EDIT	Edition des fichiers	51
EØL	Caractère de fin de ligne	20
EXP(AND)	Etend le caractère de tabulation à plusieurs lignes	41
F(IND)	Recherche une ligne dans le fichier	33
H(EADER)	Affiche une ligne d'en tête de numérotation de colonnes	23
I(NSERT)	Insère des lignes ou un texte dans le texte	42-43
LØG	Sort de l'Editeur	31-53
MØ(VE)	Transfère des lignes à un autre endroit	44
N(EXT)	Place le pointeur à une ligne suivante	34
NEW	Place un nouveau bloc de fichier dans la mémoire tampon	54
NU(MBERS)	Affiche ou supprime les numéros de lignes	24
Ø(VERLAY)	Imprime une ligne de recouvrement sur la ligne courante	45-46
P(RINT)	Affich la ligne courante et es suivantes	47
REN(UMBER)	Renumérote les lignes du fichier de travail	25
R(EPLACE)	Supprime des lignes	48
SET	Définit des caractères spéciaux de tabulation, fin de ligne...	26
S(TØP)	Sort de l'Editeur	31-33
TAB	Place les arrêts de tabulation	27
T(ØP)	Déplace le pointeur jusqu'à la première ligne du fichier	35

LISTE DES COMMANDES ET ORDRES DE L'EDITEUR DE TEXTE (suite)

Spécification	Signification	Page
V(ERIFY)	Affiche ou supprime le drapeau de vérification	28
X	Ordre de commande du curseur	29
Z(ØNE)	Restreint les recherches entre certaines colonnes	30
=	Remplace la ligne pointée par un nouveau texte	49
!	Spécifie la dernière ligne du fichier	6
^	Spécifie la première ligne du fichier	6
##	Permet de sortir du mode où l'on se trouve	1-2-4-5-42

Imprimé en France. — Imprimerie JOUVE, 18, rue Saint-Denis, 75001 PARIS

goupil

Société de Micro-informatique et Télécommunications

7. ASSEMBLEUR FLEX GOUPIL 2

1982

NOTE

Les éléments qui suivent, listings et documentations sont produits pour la satisfaction et l'usage personnel par la Société de Micro-informatique et Télécommunications SMT.

Toute reproduction, même partielle, est interdite et constitue une contrefaçon passible des peines prévues par la loi du 11 mars 1975 sur la protection des droits d'auteur.

L'application de cette règle permettra de vendre de plus en plus de programmes à des prix de plus en plus bas.

Son respect est la condition nécessaire pour que la SMT, ou d'autres sociétés, puissent produire, acheter ou adapter toujours plus de logiciels de qualité et que la créativité en ces matières se développe à votre profit.

L'Assembleur mnémonique TSC pour le système d'exploitation FLEX est un assembleur très rapide et puissant.
Il permet l'assemblage de fichiers-source très longs et crée directement sur disque des fichiers binaires lisibles par la machine.

Plusieurs options d'assemblage sont aussi disponibles.

SOMMAIRE

Attention ! Ø représente la lettre O.
O représente le chiffre zéro.

I - PRELIMINAIRES

1. Appel du programme

Le format général de l'ordre ASMB est le suivant :

ASMB, ⟨fichier 1⟩ (,⟨fichier 2⟩) (,+ ⟨liste d'options⟩)

⟨fichier 1⟩ est le fichier à assembler ; l'extension .TXT et le disque de travail sont pris par défaut.

⟨fichier 2⟩, s'il est mentionné, est le nom du fichier binaire qui sera chargé sur le disque.

Si ⟨fichier 2⟩ n'est pas spécifié le fichier binaire aura le même nom que ⟨fichier 1⟩ mais l'extension deviendra .BIN.

Au cas où un fichier binaire de même nom existerait déjà sur le disque, l'assembleur demande s'il faut le détruire.

Si oui répondre par "Y".

La liste d'options permet 8 possibilités. Elle est séparée de la ligne de commande par le signe + et chaque option est représentée par une seule lettre.

Ces options sont les suivantes :

B ne crée pas de fichier binaire sur le disque. Avec cette option, aucun fichier binaire n'est crée même si <fichier 2> est spécifié. C'est utile pour l'exécution de l'assembleur et le contrôle des erreurs avant de figer le programme final.

G édite la chaîne de caractères contenue dans le FCC (chapitre III) et n'édite pas sur le listing les valeurs hexadécimales des caractères formant cette chaîne.

L déconnecte le LIST assembleur. Normalement l'assembleur produit un listing du programme assemblé. L'option L supprime ce listing et seules sont éditées les lignes erronées (s'il y en a).

S supprime la sortie de la table des symboles. Normalement l'assembleur édite une table des symboles à la fin de chaque assemblage. L'option S supprime la sortie de cette table.

N imprime les numéros de ligne du listing assemblé. Normalement l'assembleur n'imprime pas les numéros de lignes.
Pendant la mise au point du programme il est souvent utile que chaque fonction soit numérotée.
Ces numéros sont les mêmes que ceux fournis par le système éditeur de texte. De plus les lignes erronées sont toujours imprimées avec leur numéro de ligne même si l'option N n'a pas été spécifiée.

Y cette option demande à l'assembleur de détruire un éventuel ancien fichier binaire du même nom que celui du fichier source si un fichier binaire doit être crée. Si cette option n'est pas spécifiée l'assembleur vous avertit quand même au cas où un fichier binaire de même nom existe déjà.

D supprime la date dans la ligne de titre de chaque page Normalement si l'option PAG est spécifiée, la date sera imprimée dans la ligne de titre.
Si le mode PAG n'est pas demandé, aucune date ne sera imprimée.

2. Exemples

Voici quelques exemples d'appel de l'assembleur :

1) ASMB,TEST
Le fichier TEST.TXT du disque de travail est assemblé dans un fichier TEST.BIN sur ce même disque.

2) ASMB,TEST,+LS
Même effet, mais de plus, listing et sortie de la table des symboles sont annulés.

3) ASMB,0.TEST,1.TEST.CMD,+S
Assemblage du fichier TEST.TXT du disque 0 dans un fichier binaire TEST.CMD sur le disque 1, sans édition de la table des symboles.

4) ASMB,TEST,+BN
Assemblage du fichier TEST.TXT et listing avec les numéros de lignes, sans, toutefois création d'un fichier binaire.

3. Sortie des listings assemblés sur imprimante

Pour sortir les listings sur imprimante, faire précéder la commande de la lettre P (cf la commande P dans la description de l'UCS FLEX).

Ceci suppose bien sûr que le fichier PRINT.SYS approprié existe dans le système d'exploitation.

Exemple :

```
+++ P,ASMB,TEST
```

Le listing assemblé du fichier source TEST.TXT sortira sur l'imprimante.

4. Taille de la table des symboles

Le nombre de symboles permis par l'assembleur dépend de la taille de la mémoire de l'ordinateur.

Grosso modo,12 K permettent 200 symboles, et 24 K 1.000 symboles. Entre ces deux valeurs, il y a proportionnalité (approximativement).

L'assembleur détermine automatiquement la taille de la mémoire.

5. Zones d'instruction

Sans tenir compte de l'espacement des zones dans le fichier-source, la sortie tabule les zones en colonne.

II - DESCRIPTION DE L'ASSEMBLEUR

1. Préambule

L'assembleur mnémonique TSC 6800 a été écrit pour un maximum de souplesse, le rendant utilisable aussi bien sur simple système RAM que sur système avec unité de disques.

Comme toujours, l'exigence de souplesse accroit la complexité, et il est conseillé de lire attentivement les notes suivantes avant de se servir de l'assembleur.

L'utilisateur est supposé connaître le langage assembleur et en particulier le code mnémotechnique des instructions du microprocesseur EF 6800.

Sinon, il peut se reporter au "manuel de programmation du microprocesseur EF 6800" disponible chez les distributeurs EFCIS.

Le langage source (entrée) de l'assembleur mnémonique TSC 6800 est un sous-ensemble de l'ensemble des caractères 7 bit ASCII (American Standard Code for Information Interchange, 1968).

Une signification spéciale est attachée à certains de ces caractères, comme nous le verrons ci-après.

Dans tous les cas, le bit de parité (bit le plus significatif) doit être 0. Cette restriction ne s'applique pas bien sûr aux numéros de ligne s'ils existent.

Tous les fichiers sources peuvent contenir des mnémoniques en caractère majuscule ou minuscule et des caractères hexadécimaux.

Chaque ligne de source pour l'assembleur comprend un nombre quelconque d'octets (peut-être aucun) précédant le premier caractère de l'instruction source, suivi par l'instruction source, suivie d'un retour chariot (Hex. OD).

L'instruction source comprend au plus quatre "zones" qui sont en format libre.

De gauche à droite les quatre zones sont l'étiquette, le code opération (mnémonique), l'opérande et le commentaire.

Les zones doivent être séparées par au moins un espace (un blanc).

2. Zone étiquette

Les autres restrictions et options pour chacune de ces zones sont :

a) l'étiquette doit commencer en colonne 1 et être unique.

b) les étiquettes se composent de lettre (A-Z,a-z) ou de chiffres (0-9)

c) chaque étiquette doit commencer avec une lettre (A-Z, a-z)

d) seuls les six premiers caractères de l'étiquette sont significatifs. Le reste est ignoré.

e) la zone étiquette peut être la seule zone existante.

3. La zone code

1) le code est un mot de trois lettres (A-Z, a-z) obligatoirement suivi d'un espace

2) des mnémoniques comme LDA A et AND B peuvent être écrit LDAA et ANDB. Dans ce cas la quatrième lettre doit être suivie obligatoirement d'un espace ( ).

4. La zone opérande

1) La zone opérande peut être un indicateur de mode d'adressage accompagné d'une expression ou simplement une expression.

2) l'indicateur de mode d'adressage est soit un dièse (#) suivi d'une expression pour un adressage immédiat ou une expression suivie de, X pour un adressage indexé.

3) cet opérande peut exister ou non selon le mode d'adressage.

5. La zone commentaire

1) la zone commentaire est facultative.

2) les commentaires peuvent comprendre tout caractère depuis l'espace ($ 20) jusqu'à DEL ($ 7F).

6. Les expressions

Une expression consiste en une combinaison de nombres et de symboles séparés par un des quatre opérateurs arithmétiques : +, -, *, /.

Les opérations arithmétiques se font sur des opérandes entiers de 16 bits, tronqués si nécessaire.
Les résultats sur 8 bits sont pris sur les 8 bits les moins significatifs. Les expressions ne doivent pas contenir d'espaces.

7. Les nombres

Les nombres sont composés des chiffres 0-9 et de lettres préfixées ou postfixées par un indicateur de base décrit ci-dessous.

La base ASCII permet l'emploi d'un seul caractère ASCII ($ 20 - $ 5F) comme opérande s'il est précédé d'une apostrophe.

BASE	PREFIXE	POST FIXE	COMMENTAIRE
Décimale	aucun	aucun	nombre décimal
Binaire	%	B	0,1 autorisés
Octale	@	Ø ou Q	0 à 7 autorisés
Hexadécimale	$	H	0-9, A-F autorisé
ASCII	'	non autorisé	équivalent ASCII

8. Les symboles

Les symboles sont des groupes de lettres et de chiffres ; seuls les six premiers caractères du symbole sont significatifs et le premier caractère doit être une lettre.

L'astérique * est un symbole spécial dont la valeur est la valeur courante du compteur de programme (CP).

Les lettres majuscules ne sont pas équivalents aux lettres minuscules, si bien que l'étiquette "START" est différente de l'étiquette "start".

9. L'évaluation des symboles et des expressions

L'assembleur est à deux passages et tous les symboles doivent être résolus en deux passages. En conséquence,un seul niveau de référence en avant est autorisé.

III - LES DIRECTIVES DE L'ASSEMBLEUR

Outre les expressions mnémoniques des soixante douze instructions exécutables du microprocesseur EF 6800, cet assembleur offre quatorze directives qui entraînent les opérations d'assemblage proprement dites, dont les expressions mnémoniques sont :

FCC	(Form Constant Characters)	permettent l'attribution de valeurs à des données destinées à être rangées en RAM ; ces données sont respectivement des caractères ASCII, des données de 8 bits, des données de 16 bits.
FCB	(Form Constant Byte)	
FDB	(Form Double constant Byte)	

SPC permet d'insérer des lignes blanches

ØPT permet de retenir ou d'annuler diverses possibilités

PAG permet de sauter à la page suivante

ØRG permet l'initialisation du compteur ordinal

EQU permet l'attribution d'une valeur à un symbole

END,MØN permettent de signaler à l'assembleur la fin du programme (source)

NAM,TTL permettent de donner un nom au programme (source)

RMB (Reserve Memory Bytes), permet de définir le nombre d'octets de mémoire à réserver

LIB permet à FLEX d'abandonner momentanément le fichier source en cours et d'ouvrir un fichier spécifié.

1. FCC

Crée des chaînes de caractères pour des messages ou des tables.

La chaîne de caractères "texte" est transformée en ASCII, un caractère par octet.

Les deux formats autorisés sont :

étiquette FCC nombre, texte

étiquette FCC séparateur texte séparateur (identique)

où nombre est tout nombre décimal.

Si un nombre est utilisé en tant que séparateur le premier caractère du texte ne doit pas être une virgule (,).

Le nombre de caractères maximal d'une instruction FCC est 255.

L'étiquette est facultative.

2. FCB

Evalue une expression et place les huits bits résultant en mémoire : la syntaxe est la suivante :

étiquette FCB expression 1,

expression 2, FCB , expression n

Chaque expression est séparée de la suivante par une virgule, avec 255 expressions au maximum par instruction FCB.

L'étiquette est facultative.

3. FDB

même instruction que FCB, mais génère 2 octets par expression soit 16 bits. Le format est le suivant :

étiquette FDB expression 1, expression 2, ..., expression n

Le nombre maximum d'expressions est 127 par instruction FDB.

L'étiquette est facultative.

4. SPC

insère le nombre spécifié de lignes blanches dans le listing de sortie.

Le format est le suivant :

SPC n

Aucune étiquette n'est autorisée.

Si n est absent, un seul espace est inséré.

L'opérateur SPC n'apparaît pas lui-même dans le listing de sortie.

Si le mode PAG est requis l'instruction agit dans la page donnée et non dans les pages suivantes.

5. ØPT

sert à retenir ou non diverses possibilités de l'assembleur.

Le format est le suivant :

ØPT option 1 , option 2 , ... , option n

Les étiquettes ne sont pas autorisées et aucun code n'est généré.

Si des options contradictoires apparaissent , la dernière est prioritaire.

Toutes les options sont traitées au début du deuxième passage.

Les options implicites sont prises en compte à moins de spécification particulière de l'utilisateur.

Seuls les trois premiers caractères d'une option sont significatifs et les options doivent être séparées par une virgule.

6. PAG

Si l'option PAG est retenue, l'opérateur PAG provoque un saut de page et par la suite imprime le titre (s'il existe), la date (à condition que D ne soit pas retenue) et le numéro de page en haut de la page suivante. Aucune étiquette n'est autorisée et aucun code n'est généré. La première page de listing est la page 0 et cette page n'a pas de titre.

L'opérateur PAG n'apparait pas lui-même dans le listing.

La procédure habituelle est d'écrire toutes les déclarations d'options, le titre, l'option PAG, au début du programme avant les premières instructions.

7. ØRG

l'opérateur ØRG dont le format est défini ci-dessous, définit une nouvelle adresse comme origine (PC) pour les instructions suivantes :

ØRG expression

Aucune étiquette n'est autorisée et aucun code n'est généré.

Si ØRG n'est pas mentionné l'origine implicite est 0000.

8. EQU

permet l'assignation d'une valeur à un symbole.

Une étiquette est demandée mais aucun code n'est généré.

Un seul niveau de référence en avant est permis et l'assignation ne doit pas être récursive.

étiquette EQU expression

9. END ou MØN

ces opérateurs indiquent à l'assembleur la fin du programme source. Aucune étiquette n'est autorisée et aucun code n'est généré.

Une seconde version de l'instruction END permet l'affectation d'une adresse de transfert au fichier binaire créé. Il faut alors écrire une étiquette ou une valeur dans la zone opérande de l'instruction END.

Supposons par exemple que le programme à assembler commence son exécution à la localisation $ 100, et que dans le fichier source l'étiquette START accompagne l'instruction placée par ØRG à $ 100.

L'instruction END (fin) doit maintenant inclure cette étiquette dans la zone opérande pour affecter $ 100 comme adresse de transfert.

```
          ØRG             $ 100
START     LDX             $ FFFF
          programme ici
          END             START
```

10. NAM ou TTL

Ces opérateurs définissent un titre qui sera édité au début de toutes les pages (autre que la page 0) si l'option PAG est retenue . Sinon cet ordre n'a aucun effet. Le format permet un titre de 32 caractères au plus. Aucune étiquette n'est autorisée et aucun code généré.

TTL texte du titre

Si plusieurs opérateurs TTL ou NAM apparaissent le dernier exécuté sera édité sur la page suivante.

11. RMB

cet opérateur demande à l'assembleur de réserver de la mémoire pour y ranger des données.

Aucun code n'est généré et donc le contenu de ces cases de mémoire est indéfini au moment de l'exécution.

L'étiquette est facultative :
étiquette RMB expression
expression occupe 16 bits.

12. LIB

cet opérateur demande au FLEX d'abandonner momentanément le fichier source en cours et d'ouvrir le fichier spécifié (sur le disque travail et avec une extension .TXT par défaut) pour le traitement de la source.

La forme est :
LIB fichier spécifié

Aucune étiquette n'est autorisée.

LIB n'apparaitra pas dans l'édition de sortie mais la source assemblée à partir du fichier spécifié le sera.

L'opérateur LIB ressemble à un appel MACRO, mais la source est trouvée dans un fichier externe et aucun passage de paramètres n'est permis .

Si une instruction END est trouvée dans le fichier LIB elle est ignorée. Quand le fichier est épuisé, l'assembleur retourne au fichier original et repart là où il l'avait laissé.

Les opérateurs LIB ne peuvent pas être emboîtés, ou en d'autres termes, un fichier LIB ne peut pas appeler un autre fichier LIB.

IV - DESCRIPTION D'UNE OPERATION D'ASSEMBLAGE

PASS 1 : PASØNE ($ 0293)

Le premier passage sert à construire la table des symboles qui est utilisée pour résoudre les références croissantes. Il doit exécuteravant PASS 2.

PASS 2 : PASTWØ ($ 02C3)

Plusieurs choses peuvent se produire durant ce passage. Si l'option est retenue, le listing du programme source assemblé est édité avec les messages d'erreur.

Si l'option n'est pas retenue, seules les lignes sources erronées sont éditées avec les messages d'erreur correspondants.

Si l'option est retenue, une table des symboles sera éditée après le listing assemblé (s'il existe) PASS 1 doit être lancé avant PASS 2.

Initialisation

Il existe dans l'assembleur une routine d'initialisation pour chacun des passages qui doit être lancée avant l'exécution de ce passage. Elles sont appelées : P1 INIT et P2 INIT.

V - ANNEXES

V.1 - Contrôle de page

Saut de page

Les quatre octets en $ 1166 permettent à l'utilisateur d'insérer les caractères de contrôle nécessaires pour formatter l'imprimante, c'est-à-dire le saut au début de la page suivante.

Si vous n'avez besoin que d'un caractère, placer simplement le 04 après ce caractère dans la chaîne.

Le caractère de contrôle est généralement affecté à $ 0C (présentation de feuille).

Contrôle de la marge de début de page

L'octet en $ 10C4 contrôle le nombre de lignes entre le début de la page et la ligne du titre et du numéro de page (peut être nul).

Contrôle de longueur de page

L'octet en $ 06A2 contrôle le nombre de lignes à imprimer avant l'envoi d'un saut de page. Ce nombre comprend la marge du début de page et la ligne de titre. Il doit donc être supérieur à (n + 1) lignes.

Quelques utilisateurs peuvent vouloir modifier d'autres caractères tel le nombre des colonnes imprimées dans la table des symboles, etc.

La plupart des modifications de ce type n'intéresse que peu de personnes et il ne saurait être question de les développer ici. Il est alors conseillé de consulter le code.

V.2 - Messages d'erreur

Cet assembleur comprend 12 messages d'erreur qui sont édités après les lignes erronées. Les messages d'erreur annoncent les violations de toutes les restrictions exprimées dans ce manuel et s'expliquent donc d'eux mêmes.

De plus l'octet "ERRØRS" (remis à zéro par P1INIT) est incrémenté, si des erreurs se sont produites lors d'un passage.

NOTE :

Les caractères ASCII 00 à $0C, $ à $1F et $80 à $8F ne doivent pas se trouver dans une zone de programme source, à l'exception toutefois des octets qui sont sautés par l'assembleur (octets de numéro de ligne). Leur présence donne des résultats indéfinis. D'une manière générale, il vaut mieux ne pas taper au clavier les caractères entre $00 et $1F hexadécimal (idem de $80 à $FF).

V.3 - Procédures additionnelles

L'assembleur fournit deux autres mnémoniques appelés BHS et BLØ qui sont les équivalents logiques de BCC et BCS.

Cependant BHS - Brancher si supérieur ou équivalent (BRANCH IF Higher or same) et BLØW brancher si inférieur (BRANCH IF LØWER) sont plus faciles à utiliser et à se rappeler.

Imprimé en France. — Imprimerie JOUVE, 18, rue Saint-Denis, 75001 PARIS