MANUEL DE REFERENCE CP/M

TABLE DES MATIERES

VUE D'ENSEMBLE SUR LES CARACTERISTIQUES DU CP/M VERSION 2.0

Le CP/M version 2.0 est un système d'exploitation à console unique hautement performant qui utilise des techniques de gestion par table afin de s'adapter très facilement à une grande variété de disques. Les traits caractéristiques du CP/M 2.0 comprennent une spécification de champ de un à seize "drives" logiques, chacun d'eux contenant au plus 8 mégaoctets.
Tout fichier particulier peut atteindre la taille entière du "drive" qui pourra s'aggrandir jusqu'à trente-deux mégaoctets dans les versions futures. La taille de la "directory" peut être configurée en champ pour pouvoir contenir tout nombre raisonable d'entrées et chaque fichier est étiquetté optionnellement avec les attributs de lecture/seulement (R/O) ou système.
Les utilisateurs du CP/M 2.0 sont physiquement séparés par des numéros d'utilisateurs, et les opérations de copie de fichier d'une zone d'utilisateur à une autre sont possibles. Des fonctions performantes d'accès direct à des enregistrements sont présentes dans le CP/M 2.0 ; elles permettent l'accès direct à chacun des 65536 enregistrements d'un fichier de huit mégaoctets.

Toutes les parties du CP/M 2.0 dépendantes du disque sont placées dans un "bloc de paramètres du disque" résident dans le BIOS qui est créé par le constructeur. Celui-ci permet de paramétrer le nombre maximum de disques actifs, les nombre de secteurs, la taille de l'allocation de données, l'extension maximum du disque logique.
Ces informations sur la taille sont exploitées pour générer les tables appropriées et les références des tables pour une utilisation pendant les opérations du CP/M 2.0.
L'utilisation de ces sous-routines, combinées aux algorithmes d'accès aux données gérés par des tables, font du CP/M 2.0 un véritable système universel de gestion de données.

L'agrandissement des fichiers est obtenu en fournissant jusqu'à 512 extensions logiques de fichier, chacune d'entres-elles contenant 16 k octets de données. Le CP/M 2.0 est structuré, cependant, de telle façon qu'une quantité aussi importante que 128 k octets de données soit adressée par une extension physique unique (correspondant à une entrée de "catalogue" unique), ce qui conserve la comptabilité avec les versions antérieures tout en profitant de l'espace du "catalogue".

Les caractéristiques d'accès direct sont présentes dans CP/M 2.Ø et permettent une référence immédiate à tout enregistrement d'un fichier de huit mégaoctets. En utilisant l'organisation des données unique du CP/M, les blocs de données ne sont alloués que lorsqu'ils **sont effectivement demandés** et les déplacements jusqu'à une position d'enregistrement précise requièrent peu de temps de recherche.
L'accès aux fichiers séquentiels est compabible en amont depuis les versions antérieures avec les huit mégaoctets pleins, alors que la comptabilité de l'accès direct s'arrête aux fichiers de 512 K octets. A cause de l'accès direct plus simple et plus rapide du CP/M 2.Ø, les programmeurs d'applications sont encouragés à modifier leurs programmes pour profiter au maximum des caractèristiques du 2.Ø.

Plusieurs modules et utilitaires du CP/M 2.Ø possèdent des améliorations qui correspondent au système des fichiers amélioré. STAT et PIP comptent à la fois pour des attributs de fichier et pour des zones d'utilisateurs, tandis que le CCP procure une fonction "login" qui permet de changer de zone d'utilisateur. Le CCP formatte également les affichages du "catalogue" d'une manière plus pratique à la fois pour l'écran et les périphériques imprimantes avec ses fonctions d'édition de ligne améliorées.

1. GENERALITES

Le CP/M est un logiciel d'exploitation pour micro-ordinateurs. Il est utilisé sur les micro-ordinateurs utilisant le micro-processeur Z80 ou 8080. CP/M donne l'environnement général nécessaire à la construction, au stockage et à l'édition d'un programme aussi bien que des possibilités d'assemblage et de vérification de programmes. Il utilise 60 K octets de mémoire principale avec au plus quatre "drives" de disquettes compatibles IBM.

Le moniteur CP/M donne un accès rapide aux programmes grâce à un logiciel de gestion de fichiers intelligent. Le sous-système fichiers contient une structure de fichiers portant un nom permettant l'allocation dynamique de l'espace fichier ainsi que des accès directs et séquentiels. En utilisant ce système de fichiers, un grand nombre de fichiers distincts peut être stocké à la fois sous forme source ou sous forme exécutable.

Le CP/M contient aussi un éditeur de texte puissant, un assembleur compatible INTEL et des sous-systèmes de détection d'erreurs. Le "software" optionnel inclut un assembleur macro, compatible INTEL, puissant, un détecteur d'erreurs symbolique, ainsi que divers langages de haut niveau. Lorsqu'elles sont couplées avec le processeur de commande console du CP/M, les possibilités proposées sont égales ou supérieures aux possibilités d'ordinateurs similaires plus grands.

Le CP/M est divisé en plusieurs parties logiques :

BIOS - le système d'entrées/sorties, (dépendant du "hardward")
BDOS - le système d'exploitation des disques
CCP - le processeur de commande console
TPA - la zone de programme transitoire

Le BIOS fournit les opérations primitives nécessaires à l'accès aux "drives" de disquettes et à l'interfaçage avec les périphériques standards (télétype, CRT, lecteur/perforateur de bandes et périphériques définis par l'utilisateur).

Le BDOS fournit une gestion du disque en contrôlant un ou plusieurs "drives" contenant des "catalogues" de fichiers indépendants. Le BDOS utilise une stratégie d'allocation du disque qui donne des constructions de fichiers complètement dynamiques tout en minimisant les mouvements de la tête sur le disque lors de l'accès. N'importe quel fichier peut contenir n'importe quel nombre d'enregistrements, dans les limites de la taille d'un disque (240 enregistrements de 128 octets chacun). Dans un système CP/M standard, chaque disque peut contenir jusqu'à 64 fichiers.

Le BDOS a des points d'entrées qui contiennent les opérations primitives suivantes :

SEARCH	recherche d'un fichier particulier sur disque d'après son nom
OPEN	ouverture d'un fichier pour des opérations ultérieures
CLOSE	fermeture d'un fichier après traitement
RENAME	changement du nom d'un fichier
READ	lecture d'un enregistrement d'un fichier donné
WRITE	écriture d'un enregistrement sur le disque
SELECT	sélection d'un "drive" disque pour des opérations ultérieures

Le CCP donne une interface symbolique entre la console utilisateur et le reste du système CP/M. Le CCP reçoit de la console et exécute des commandes telles que l'affichage du "catalogue" des fichiers, l'impression du contenu des fichiers, et le contrôle des opérations des programmes transitoires, tels que les assembleurs, les éditeurs et les outils de mise au points. La liste des commandes standards disponibles dans le CCP est donnée dans les sections suivantes.

Le dernier segment du CP/M est une zone appelée la TPA. La zone de programme transitoire (TPA) contient les programmes qui sont chargés depuis un disque par le CCP après une commande. Pendant l'édition d'un programme, par exemple, la TPA contient le code machine de l'éditeur de texte du CP/M et la zone d'informations. De manière analoge, on peut vérifier des programmes créés sous CP/M en les chargeant et en les exécutant dans la TPA.

Il faut remarquer que n'importe quel sous-système entrant dans la composition du CP/M peut être "recouvert" par un programme en exécution. Autrement dit, une fois qu'un programme utilisateur est chargé dans la TPA, les zones CCP, BDOS et BIOS peuvent être utilisées comme zones de données du programme. Un chargeur du programme d'initialisation est accessible par programme à chaque fois que la portion BIOS n'est pas recouverte ; ainsi le programme utilisateur n'a qu'à se brancher sur le chargeur de réinitialisation, et le moniteur du CP/M en entier sera rechargé depuis le disque.

Remarquez aussi que le système d'exploitation CP/M est divisé en modules distincts, y compris la portion BIOS qui définit la configuration "hardware" dans laquelle le CP/M fonctionne.

2. DESCRIPTION FONCTIONNELLE DU CP/M

L'utilisateur agit avec le CP/M en premier lieu, grâce au CCP qui lit et interprète les commandes entrées sur la console. En général, le CCP adresse l'un des nombreux disques qui sont en utilisation (le système standard peut adresser jusqu'à quatre "drives" disques différents). Ces "drives" portent les étiquettes "A", "B", "C", "D". Un disque est "loggé" si le CCP l'adresse actuellement. Afin d'indiquer clairement le disque qui est actuellement loggé, le CCP signale toujours à l'opérateur le nom du disque, suivi du symbole ">" indiquant que le CCP est prêt pour une autre commande. Après la première initialisation, le système CP/M est chargé depuis le disque A, et le CCP affiche le message de début :

xxk CP/M VER m.m

où xx est la taille de la mémoire (en kilooctets) dont dispose ce CP/M, et m.m est le numéro de la version CP/M. Tous les systèmes CP/M sont initialement établis pour opérer dans un espace de mémoire de 16 K, mais ils peuvent être facilement reconfigurés pour s'adapter à toute taille de mémoire sur le système hôte.
Après le signal de début du système, le CP/M "logge" automatiquement le disque "A", et donne à l'utilisateur le signal "A>" (qui indique que le CP/M adresse actuellement le disque "A") et attend une commande
Les commandes sont implantées à deux niveaux : les commandes fixes et les commandes transitoires.

2.1 STRUCTURE GENERALE D'UNE COMMANDE

Les commandes fixes sont une partie du programme CCP lui même, alors que les commandes transitoires sont chargées dans le TPA à partir du disque, puis exécutées. Les commandes fixes sont :

ERA	efface des fichiers spécifiés
DIR	donne la liste des noms de fichiers du catalogue ("Directory")
REN	change le nom d'un fichier donné
SAVE	sauvegarde le contenu de la mémoire dans un fichier
TYPE	affiche le contenu d'un fichier se trouvant sur le disque loggé

Presque toutes les commandes font référence à un fichier ou à un groupe de fichiers donnés. La forme des références de fichiers est définie ci-dessous.

2.2 REFERENCES DE FICHIERS

Une référence de fichier identifie un fichier ou un groupe de fichiers sur un disque particulier relié au CP/M. Ces références de fichiers peuvent être soit "ambigüe" (rfa), soit "non ambigües" (rfn). Une référence de fichier "non ambigüe" identifie uniquement un seul fichier, alors qu'une référence ambigüe de fichier peut être satisfaite par plusieurs fichiers.

Les références de fichiers se composent de deux parties : le nom primaire et l'extension. Bien que l'extension soit optionnelle, elle est généralement générique ; autrement dit, l'extension "ASM", par exemple,

est utilisée pour indiquer que le fichier est un fichier source d'assem-

bleur, alors que le nom primaire distingue chaque fichier source en particulier. Les deux noms sont séparés par un point :

pppppppp.sss

où pppppppp représente le nom primaire composé de huit caractères ou plus, et sss est l'extension comportant au plus trois caractères. Le nom :

pppppppp

contient un nom secondaire composé de trois blancs. Les caractères utilisés pour spécifier une référence non ambigüe de fichier ne peuvent contenir les caractères spéciaux suivants :

< > . , ; : = ? *
^ blanc

alors que les caractères alphanumériques et les autres caractères spéciaux sont autorisés.

Une référence ambigüe de fichier est utilisée pour les recherches dans le "catalogue" et les égalités de type. La forme d'une référence ambigüe

de fichier est analogue à celle d'une référence non ambigüe, sauf que l'on peut utiliser le symbole "?" dans les noms et extensions. Dans diverses commandes du CP/M, le symbole "?" indique que tout nom de fichier satisfait l'égalité si tous les caractères sont égaux dans les positions correspondantes, le "?" pouvant être égal à n'importe quel caractère. Ainsi, la référence ambigüe :

X?Z.C?M

est satisfaite par les noms de fichiers non ambigüs :

XYZ.COM ET X3Z.CAM

Remarquez que la référence ambigüe :

```
*.*
```

est équivalente à :

```
????????.???
```

alors que :

```
pppppppp.*.
```

et

```
*.sss
```

sont des abbréviations de :

```
pppppppp.sss
```

et

```
????????.sss
```

respectivement. Par exemple,

```
DIR *.*
```

est interprété par le CCP comme une commande donnant la liste des noms de tous les fichiers sur disque de la "directory", alors que :

```
DIR X.Y
```

cherche seulement un fichier de nom X.Y. De manière analogue, la commande :

```
DIR X?Y.C?M
```

entraîne une recherche de tous les noms de fichiers (non ambigüs) du disque qui satisfont cette référence ambigüe. Les noms de fichiers suivants sont des références de fichiers non ambigües correctes :

```
X         XYZ         GAMMA

X.Y       XYZ.COM     GAMMA.1
```

De plus, le programmeur peut également spécifier le nom du "drive" disque avec le nom du fichier. Dans ce cas, le nom du "drive" est donné sous la forme d'une lette de A à Z suivie par deux points (:). Le "drive" spécifié est alors "loggé" avant que l'opération de fichier ne soit réalisée. Ainsi, les noms de fichier et les préfixes de noms de disques suivants sont valides :

```
A:X.Y          B:XYZ          C:GAMMA

Z:XYZ.COM      B:X.A?M        C:*.ASM
```

Il faut aussi noter que toutes les lettres alphabétiques minuscules dans les noms de fichier et de "drive" sont toujours converties en majuscules quand elles sont exécutées par la CCP.

3. CHANGEMENT DE DISQUE

L'opérateur peut changer le disque loggé en tapant le nom du nouveau "drive" disque (A, B, C ou D) suivi de deux points (:) lorsque le CCP attend une entrée de la console. Ainsi, la suite de signaux et de commandes montrée ci-dessous peut apparaître après que l'on ait chargé le système CP/M à partir du disque A :

```
16 K CP/M VER 1.4

A>DIR          donne la liste de tous les fichiers du disque A.

SAMPLE ASM

SAMPLE PRN

A>B:           sélectionne sur le disque B.

B>DIR *.ASM    donne la liste de tous les fichiers "ASM" de B

DUMP  ASM

FILES ASM

B>A:           revient au disque A.
```

4. FORME DES COMMANDES FIXES

Les formes des références des fichiers et périphériques décrites ci-dessus peuvent maintenant être utilisées pour spécifier pleinement la structure des commandes fixes. Voici les abbréviations utilisées dans les descriptions suivantes :

rfn - référence de fichier non ambigüe

rfa - référence de fichier ambigüe

rc - retour chariot

remarquez que le CCP convertit toujours de manière interne les caractères minuscules en caractères majuscules.
Ainsi, les minuscules sont traitées comme si elles étaient des majuscules dans les noms de commandes et dans les références de noms de fichier.

4.1 ERA rfa rc

La commande ERA retire les fichiers du disque actuellement loggé (celui dont le nom est donné avant le signal CP/M ">"). Les fichiers qui sont effacés sont ceux qui satisfont la référence ambigüe rfa. Les exemples suivants illustrent l'utilisation de ERA :

ERA X.Y	Le fichier X.Y sur le disque courant est retiré du "catalogue" du disque et l'espace correspondant est libéré.
ERA X.*	Tous les fichiers dont le nom est X sont enlevés du disque courant.
ERA X?Y.C?M	Tous les fichiers du disque courant qui satisfont la référence ambigüe X?Y.C?M sont effacés.
ERA *.*	Efface tous les fichiers du disque courant (dans ce cas, le CCP affiche sur la console le messag "ALL FILES (Y/N)?" auquel vous devez répondre par un Y pour que les fichiers soient supprimés)
ERA B:*.PRN	tous les fichiers du "drive" B qui correspondent à la référence ambigüe ????????.PRN sont effacés indépendamment du disque actuellement logé.

4.2 DIR rfa rc

La commande DIR donne sur la console la liste de tous les noms de fichiers qui satisfont la référence ambigüe rfa. La commande :

```
DIR
```

affiche une liste des fichiers du disque actuellement loggé (la commande "DIR" est équivalente à la commande "DIR *.*"). Les commandes DIR valides sont citées ci-dessous.

```
DIR X.Y
DIR X?Y.C?M
DIR ??.Y
```

Comme pour d'autres commandes CCP, la rfa peut être précédée par un nom de "drive". Les commandes DIR suivantes s'adressent au "drive" sélectionné avant que la recherche n'ait lieu.

```
DIR B:
DIR B:X.Y
DIR B:*.A?M
```

Si aucun fichier correspondant à la demande ne peut être trouvé dans le catalogue de la disquette sélectionnée, le message "NOT FOUND" (NON TROUVE) apparaît sur la console.

4.3 REN rfn1=rfn2 rc

La commande REN permet à l'utilisateur de changer les noms des fichiers sur disque. Le fichier satisfaisant rfn2 devient rfn1. Le disque courant est supposé contenir le fichier dont il faut changer le nom. Le CCP permet aussi à l'utilisateur de remplacer le signe égal par une flèche orientée à gauche, si sa console contient ce caractère. (←)

Voici quelques exemples de commandes REN :

REN X.Y=Q.R	le fichier Q.R devient X.Y
REN XYZ.COM=XYZ.XXX	le fichier XYZ.XXX devient XYZ.COM

L'opérateur peut faire précéder rfn1 ou rfn2 (ou les deux ensembles) par une adresse de "drive" optionnelle. Si rfn1 est précédé par un nom de "drive", rfn2 est alors supposé existant sur le même "drive" que celui de rfn1. De la même façon, si rfn2 est précédé par un nom de "drive", alors rfn1 est assumé existant sur ce même "drive". Si rfn1 et rfn2 sont tous les deux précédés par un nom de "drive", les "drives spécifiés doivent être identiques. Les commandes REN suivantes illustrent ce format.

REN A:X.ASM=Y.ASM	le fichier Y.ASM est transformé en X.ASM sur le "drive" A.
REN B:ZAP.BAS=ZOT.BAS	le fichier ZOT.BAS est transformé en ZAP.BAS sur le "drive" B.
REN B:A.ASM=B:A.BAK	le fichier A.BAK est renommé A.ASM sur le "drive" B.

Si le fichier rfn1 existe déjà, la commande répondra par "FILE EXISTS" et n'effectuera pas le changement de nom. Si rfn2 n'existe pas sur la disquette spécifiée, le message "NOT FOUND" apparaîtra alors sur la console.

4.4 SAVE n rfn rc

La commande SAVE enregistre n pages (blocs de 256 octets) sur le disque à partir de la TPA et appelle ce fichier rfn. Dans le système CP/M commercialisé, la TPA commence à 100H (héxadécimal), ce qui correspond à la seconde page de mémoire. Ainsi, si le programme de l'utilisateur occupe la zone entre 100H et 2FFH, la commande SAVE doit spécifier 2 pages de mémoire. Le fichier de code machine peut être par la suite chargé et exécuté. Voici quelques exemples :

```
SAVE  3 X.COM      copie de 100H à 3FFH sous le nom de X.COM.

SAVE  40 Q         copie de 100H à 28FFH sous le nom de Q (notez que
                   28 est le numéro de compte de la page dans 28FFH,
                   et que 28H=2*16+8 = 40 entier).

SAVE  4 X.Y        copie de 100H à 4FFH sous le nom de X.Y.
```

La commande SAVE peut aussi spécifier un "drive" disque dans la partie rfa de la commande, comme il est montré ci-dessous.

4.5 TYPE rfn rc

La commande TYPE affiche sur la console le contenu du fichier source en ASCII dont le nom est rfn et qui se trouve sur le disque courant. Les commandes TYPE correctes sont :

```
TYPE  X.Y
TYPE  X.PLM
TYPE  XXX
```

La commande TYPE tient compte des tabulations (controle I), supposant que les positions tabulaires sont placées toutes les huits colonnes. Le rfn peut aussi faire référence à un nom de "drive" comme il est montré ci-dessous.

```
TYPE B:X.PRN       le fichier X.PRN du "drive" B est affiché.
```

5. EDITION DE LIGNES ET CONTROLE DE SORTIE

Le CCP autorise certaines fonctions d'édition de lignes lors de l'enregistrement d'une commande.

RUBOUT efface et renvoie en écho le dernier caractère tapé sur la console.

CTRL-U efface toute la ligne tapée sur la console.

CTRL-X (identique à CTRL-U)

CTRL-R retape la ligne de commande courante corrigée : "nettoyage" suivant la suppression de caractères avec des RUBOUTS.

CTRL-E fin physique de ligne : le chariot est renvoyé, mais la ligne ne sera envoyée que lorsque la touche de retour de chariot aura été

CTRL-C réinitialisation du système CP/M (départ à chaud)

CTRL-Z fin d'entrée console (utilisée par PIP et ED)

Les fonctions de contrôle CTRL-P et CTRL-S affectent la sortie sur console comme il est montré ci-dessous.

CTRL-P produit un écho de toute sortie console sur le périphérique imprimant actuellement assigné (voir la commande STAT). La sortie est envoyée à la fois au périphérique imprimant et à la console jusqu'à ce que CTRL-P soit à nouveau tapé.

CTRL-S stoppe temporairement la sortie console. l'exécution du programme et la sortie continuent quand le prochain caractère est tapé sur la console (e. g., un autre CTRL-S). Cette caractéristique est utilisée pour stopper la sortie sur des consoles à grande vitesse, comme celles d'un écran, afin de pouvoir observer un segment de sortie avant de continuer.

Notez que les séquences de CTRL-lettre présentées ci-dessus sont obtenues en appuyant **simultanément** sur la touche CTRL et la touche de la lettre. Les lignes de commandes CCP peuvent contenir jusqu'à 255 caractères en longeur ; elles ne sont traitées qu'après la frappe de retour-chariot.

6. COMMANDES TRANSITOIRES

Les commandes transitoires sont chargées depuis le disque "loggé" courant et exécutées dans la TPA. Les commandes transitoires définies pour une exécution avec le CCP sont présentées ci-dessous. Des fonctions supplémentaires peuvent être facilement définies par l'utilisateur (voir la définition de la commande LOAD).

STAT	donne le nombre d'octets de stockage restant sur le disque courant, fournit des informations statistiques sur des fichiers particuliers, et affiche ou modifie l'assignement de périphérique.
ASM	charge l'assembleur du CP/M et assemble le programme spécifié
LOAD	charge le fichier en format code machine "hex" d'INTEL et produit un fichier sous forme exécutable qui peut être chargé dans la TPA (ce programme chargé devient une nouvelle commande sous le CCP).
DDT	charge le DDT du CP/M dans la TPA et commence son exécution
PIP	charge le PIP (Périphéral Interchange Program) pour des opérations ultérieures sur fichier de disque et de transfert entre périphériques.
ED	charge et exécute le programme éditeur de texte du CP/M
SYSGEN	crée une nouvelle disquette de système CP/M (Non disponible sur SIL'Z)
SUBMIT	soumet un fichier de commande au traitement par lots (batch)
DUMP	affiche le contenu d'un fichier en héxadécimal
MOVCPM	recrée le système CP/M pour une taille de mémoire particulière. (Non disponible sur SIL'Z)

Les commandes transitoires sont spécifiées de la même manière que les commandes fixes, et des commandes supplémentaires peuvent être facilement créées par l'utilisateur. De plus, la commande transitoire peut être précédée par un nom de "drive", ce qui entraine qu'elle est chargée à partir du "drive" spécifié dans la TPA avant exécution. Ainsi, la commande :

```
B:STAT
```

oblige CP/M à "logger" temporairement le "drive" B pour charger le programme transitoire STAT, et à retourner ensuite au disque "loggé" original pour le traitement suivant. Les commandes transitoires de base sont citées en détail ci-dessous.

6.1 STAT rc

La commande transitoire STAT fournit des informations statistiques générales sur le stockage des fichiers et l'assignation de périphériques.

Elle se présente sous l'une de ces différentes formes :

```
STAT rc
STAT "ligne de commande" rc
```

Des formes spéciales de la "ligne de commande" permettent à l'assignation de périphérique courant d'être examiné et tout aussi bien modifié. Les diverses lignes de commandes qui peuvent être spécifiées sont présentées ci-dessous, avec une explication de chacune des formes.

STAT rc

Si l'utilisateur tape une ligne de commande vide, la transitoire STAT calcule la capacité de stockage disponible sur tous les "drives" actifs, et imprime un message : x: R/W, SPACE: nnnK
ou x: R/O, SPACE: nnnK

pour chaque "drive" x actif, où R/W indique que le "drive" peut être lu ou écrit, et R/O indique que le "drive" est en R/O (lecture seulement) (un "drive" devient R/O en l'établissant explicitement à lecture seulement comme montré ci-dessus, ou en changeant par inadvertance des disquettes sans réinitialiser le système). L'espace restant sur la disquette dans le "drive" x est donné en kilo-octets.

STAT x: rc

Si un nom de "drive" est donné, le "drive" est alors sélectionné avant que le stockage ne soit calculé. Ainsi, la commande "STAT B:" pourrait être envoyée alors que vous êtes loggé dans le "drive" A, ce qui donnerait le message : BYTES REMAINING ON B: nnnK

STAT rfa rc

La ligne de commande peut aussi spécifier une série de fichiers qui devront être analysés par STAT. Les fichiers qui correspondent à la rfa sont rangés par ordre alphabétique ; les besoins de stockage pour chaque fichier sont indiqués sous le titre

```
RECS BYTS EX D:NOMFICHI.TYP
rrrr bbbK ee d:pppppppp.sss
```

où rrr est le nombre d'enregistrement de 128 octets alloués au fichier,
bbb le nombre de kilooctets alloués au fichier (bbb=rrrr*128/1024),
ee le nombre d'extensions de 16K (ee=bbb/16),
d le nom du "drive" contenant le fichier (A...Z),
pppppppp le nom du fichier primaire comprenant jusqu'à huit caractères,
et sss le nom secondaire comprenant au plus trois caractères.
Une fois la liste des fichiers individuels établie, l'emploi du stockage est récapitulé.

STAT x:rfa rc

Pour plus de commodités, le nom du "drive" peut être donné avant la rfa. Dans ce cas, le "drive " spécifié est sélectionné en premier, et la forme "STAT rfa" est exécutée.

STAT x:=R/O rc

Cette forme établit le "drive" donné par un x à "lecture seulement", qui restera effective jusqu'à ce que le prochain "départ à froid" ou "départ à chaud" ait lieu. Quand un disque est en "lecture seulement", le message

BDOS ERR ON x: READ ONLY

apparaît si vous tentez d'écrire sur le disque x établi en "lecture seulement". CP/M attend qu' **une** touche soit frappée pour exécuter un "départ à chaud" automatique (auquel cas le disque revient R/W).

La commande STAT permet également le contrôle des assignations de périphériques aux périphériques logiques. En général, il existe quatre unités logiques périphériques qui sont, à tout instant, assignées chacune à l'une des unités périphériques physiques. Les quatre unités logiques s'appellent :

CON: le périphérique console du système (utilisé par CCP pour communiquer avec l'opérateur)

RDR: le périphérique lecteur de bande

PUN: le périphérique perforateur

LST: le périphérique liste de sortie

Les unités réelles attachées à un système quelconque d'ordinateur sont gérées par des sous-routines dans la portion BIOS du CP/M. Ainsi, l'unité logique RDR:, par exemple, pourrait être un lecteur à grande vitesse, un lecteur télétype, ou un lecteur de cassette. Afin de permettre une certaine liberté dans la dénomination et l'assignation des unités, plusieurs unités physiques sont définies :

TTY: périphérique télétype (console à faible vitesse)

CRT: périphérique à tube cathodique (console à grande vitesse)

BAT: traitement par lots (console est RDR: courant, la sortie se fait sur le périphérique LST: courant)

UCL: console définie-utilisateur

PTR: lecteur de bande perforée(lecteur à grande vitesse)

UR1: lecteur n1 défini-utilisateur

UR2: lecteur n2 défini-utilisateur

PTP: perforateur (perforateur à grande vitesse)

UP1: perforateur n1 défini-utilisateur

UP2: perforateur n2 défini-utilisateur

LPT: imprimante de ligne

UL1: périphérique imprimant n1 défini-utilisateur

Il doit ête signalé que les noms des unités physiques peuvent ou non correspondre aux unités auxquelles les noms s'appliquent. C'est à dire le périphérique PTP: peut être implanté comme une opération d'écriture sur cassette, si l'utilisateur le désire. La sous-routine de correspondance exacte et de "driver" est définie dans la portion BIOS du CP/M.
Les assignations possibles entre unités logiques et physiques peuvent être affichées en tapant : STAT VAL: rc

STAT imprime les valeurs qui peuvent être prises par chacun des périphériques logiques :

```
CON: = TTY: CRT: BAT: UC1:
RDR: = TTY: PTR: UR1: UR2:
PUN: = TTY: PTP: UP1: UP2:
LST: = TTY: CRT: LPT: UL1:
```

Dans chaque cas, le périphérique logique situé sur la gauche peut prendre n'importe laquelle des quatre assignations physiques situées dans la partie droite de chaque ligne. L'affectation courante des périphériques logiques aux périphériques physiques est affichée en tapant la commande : STAT DEV: rc

qui crée un tableau avec sur la gauche chaque périphérique logique et sur la droite le périphérique physique courant qui lui correspond. Par exemple, la liste pourrait apparaître comme suit :

```
CON: = CRT:
RDR: = UR1:
PUN: = PTP:
LST: = TTY:
```

L'assignation courante du périphérique logique au périphérique physique peut être changée en tapant une commande STAT de la forme :

```
STAT ld1 = pd1, ld2 = pd2, ... , ldn = pdn rc
```

où ld1 à ldn sont les noms de périphériques logiques, et pd1 à pdn les noms de périphériques physiques compatibles (i.e., ldi et pdi apparaissent sur la même ligne dans la commande "VAL:" présentée ci-dessus).
Les commandes suivantes sont des commandes STAT valides qui permettent de changer les assignations courantes :

```
STAT CON: CRT: rc
STAT PUN: TTY:, LST: = LPT:, RDR: = TTY: rc
```

6.2 ASM rfn rc

La commande ASM charge et exécute l'assembleur 8080 du CP/M. Le rfn détermine un fichier source contenant des instructions en assembleur, et dont l'extension est supposée être ASM, et peut donc ne pas être spécifiée. Les commandes ASM suivantes sont correctes :

```
ASM X
ASM GAMMA
```

L'assemblage en deux passages est alors automatiquement exécuté. Si des erreurs d'assemblage apparaissent pendant le passage 2, elles sont affichées sur la console.

L'assembleur produit un fichier :

```
X.PRN
```

où X est le nom primaire spécifié dans la commande ASM. Le fichier PRN contient un listing du programme source (avec des caractères de tabulation s'il y en avait dans le fichier source), ainsi que le code machine généré pour chaque instruction et des messages d'erreur et de diagnostic, éventuellement.Le fichier PRN peut être affiché sur console en utilisant la commande TYPE, ou envoyé sur un autre périphérique avec la commande PIP (voir la structure de la commande PIP ci-dessous).

Notez également que le fichier PRN contient le programme source original, augmenté d'informations diverses d'assemblage dans les 16 colonnes les plus à gauche (adresse de programme et code machine hexadécimal, par exemple). Ainsi, le fichier PRN peut servir de copie de sauvegarde pour le fichier source original : si le fichier source est accidentellement enlevé ou détruit, le fichier PRN peut être édité (voir le guide de l'opérateur de ED) en enlevant les 16 caractères les plus à gauche de chaque ligne (cela peut être fait en créant une commande d'éditeur simple "macro"). Le fichier résultant est identique au fichier source original et peut être renommé (REN) de PRN en ASM pour l'édition et l'assemblage suivants.

Le fichier x.HEX

est également produit ; il contient un langage machine 8080 en format "hex" Intel qui convient pour un chargement et une exécution ultérieurs

(voir la commande LOAD). Pour avoir plus de détails sur les programmes en assembleur du CP/M, reportez-vous au "Guide de l'utilisateur de l'assembleur du CP/M (ASM).

Semblable à d'autres commandes transitoires, le fichier source pour assemblage peut être pris d'un autre disque en faisant précéder le nom du fichier en langage assembleur par un nom de "drive" de disque. Ainsi la commande

```
ASM b:ALPHA rc
```

charge l'assembleur à partir du "drive" actuellement loggé et opère sur le programme source ALPHA.ASM sur le "drive" B. Les fichiers HEX et PRN sont aussi placés sur le "drive" B dans ce cas.

6.3 LOAD rfn rc

La commande LOAD lit le fichier rfn (il doit contenir le code machine sous format "hex") et produit un fichier image mémoire qui pourra être ultérieurement exécuté. Le nom de fichier rfn doit être de la forme :

x.HEX

et c'est pourquoi il suffit de spécifier le nom x dans la commande. La commande LOAD crée un fichier appelé :

x.COM

ce qui indique que c'est un fichier de code exécutable. Ce fichier est effectivement chargé en mémoire et exécuté dès que l'utilisateur frappe

x juste après le signal ">" donné par le CCP.
En général, le CCP lit le nom x qui suit le signal et cherche une fonction fixe de ce nom. S'il ne la trouve pas, il cherche dans le "catalogue" du disque système un fichier de nom :

X.COM

S'il le trouve, le code machine est chargé dans la TPA, et le programme est exécuté. Ainsi, l'utilisateur n'a besoin de charger un fichier hex qu'une fois ; il pourra être, par la suite, exécuté n'importe quel nombre de fois en tapant simplement le nom primaire. De cette façon, l'utilisateur peut "inventer" de nouvelles commandes dans le CCP. (En fait, les disques initialisés ont des commandes transitoires sous forme de fichiers COM, qui peuvent ainsi être détruites par l'utilisateur.) L'opération peut se faire sur un autre disque si le nom de fichier est précédé par un nom de "drive".
Ainsi,

LOAD B: BETA

transfère le programme LOAD du disque loggé courant dans la TPA et opère sur le "drive" B après que l'exécution ait commencé.

Il doit être noté que le fichier BETA.HEX doit contenir des enregistrements en code machine hexadécimal de format Intel valides (comme ils sont produits par le programme ASM, par exemple) qui commencent à 100H, le début de la TPA. De plus, les adresses dans les enregistrements hex doivent être dans un ordre croissant ; les trous dans des régions de mémoire non pleines sont remplis par des zéros par la commande LOAD en même temps que les enregistrements hex sont lus. Ainsi, LOAD doit n'être utilisée que pour créer des fichiers "COM" standards du CP/M qui opèrent dans la TPA. Les programmes qui occupent des zones de mémoire autres que la TPA peuvent être chargés sous DDT.

6.4 PIP rc

PIP est le programme de changement de périphérique du CP/M qui implante les opérations de base de conversion entre périphériques, qui sont nécessaires pour changer, imprimer, mettre sur bande, copier et combiner des fichiers sur disque.

Le programme PIP est lancé en tapant une des formes suivantes :

(1) PIP rc
(2) PIP "ligne de commande" rc

Dans les deux cas, PIP est chargé dans la TPA et exécuté. Dans le premier cas, PIP lit les lignes de commandes directement sur la console, signalées par un caractère "*" jusqu'à ce qu'une ligne de commande vide soit tapée (un simple retour chariot effectué par l'utilisateur). Chaque ligne de commande a pour effet une conversion selon les règles décrites ci-dessous. La deuxième forme de la commande PIP est équivalente à la première sauf que la ligne de commande unique donnée dans la commande PIP est exécutée automatiquement, et PIP se termine immédiatement sans autre signal sur la console pour d'autres entrées de lignes de commandes. La forme de chaque ligne de commande est :

destination = source 1, source 2, ... source n rc

où "destination" est le fichier ou l'unité périphérique qui doit recevoir l'information, et "source 1, source 2, ..., source n" une série de un ou plusieurs fichiers ou unités périphériques qui sont copiés de gauche à droite vers la destination.

Quand plusieurs fichiers sont donnés dans la ligne de commande (i.e., $n > 1$), les fichiers individuels sont supposés contenir des caractères ASCII, avec un caractère fin-de-fichier (CTRL-Z) apposé par CP/M à la fin de chaque fichier (voir le paramètre O pour outrepasser cette assomption). Le symbole égal (=) peut être remplacé par une flèche orientée vers la gauche si votre console contient ce caractère ASCII, pour améliorer la lecture. Les minuscules ASCII deviennent des majuscules pour être en accord avec les conventions CP/M sur les noms de fichiers et de périphériques. Enfin, la longueur totale de la ligne de commande ne doit pas excéder 255 caractères (le caractères CTRL-E peut être utilisé pour forcer un retour de chariot physique pour les lignes qui excèdent la largeur de la console).

La destination et les éléments sources peuvent être des références ambigües à des fichiers sources du CP/M, avec ou sans nom de "drive" de disque le précédant. Autrement dit, n'importe quel fichier peut être appelé avec un nom de "drive" au début de la référence (A:, B:, C:, ...), ce nom définissant le "drive" dans lequel on doit aller chercher ou stocker le fichier.

Le fichier destination peut aussi apparaître comme un ou plusieurs des fichiers sources, auquel cas, le fichier source ne sera modifié que lorsque toute concaténation sera terminée. Si le fichier de destination existe déjà, il est effacé si la ligne de commande est correctement formée (il ne l'est pas si une condition d'erreur apparaît).

Les lignes de commande suivantes (avec leurs explications à droite) sont correctes pour PIP :

```
X=Y rc      copie Y sur X, où X et Y sont des noms de fichier non
                    ambigüs ; Y reste inchangé.

X=Y,Z rc            concatène les fichiers Y et Z et les copie sur X ;
                    Y et Z restent inchangés.

X.ASM=Y.ASM,Z.ASM,FIN.ASM rc
                    crée le fichier X.ASM à partir de la concaténation des
                    fichiers Y, Z et FIN de type ASM.

B:A.U=B:B.V,A:C.W,D.X rc
                    concatène le fichier B.V du "drive" B avec C.W du
                    "drive" A et D.X du disque loggé, et crée le fichier
                    A.U sur le "drive B.
```

Pour une utilisation plus pratique, PIP accepte les commandes abrégées pour transférer des fichiers entre des "drives" disques. Les formes abrégées sont :

```
PIP x:=rfa rc
PIP x:=y:rfa rc
PIP rfn=y: rc
PIP x:rfn=y: rc
```

La première forme de copie tous les fichiers du disque actuellement loggé qui satisfont à la rfa sur les mêmes noms de fichier sur le "drive" x (x = A...Z). La troisième forme est équivalente à la commande "PIP rfn=y:rfn rc" qui copie le fichier donné par rfn du "drive" y dans le fichier rfn sur le "drive" x. La quatrième forme est équivalente à la troisième, où le disque source est explicitement donné par y.

Notez que les fichiers source et destination doivent être différents dans tous ces cas. Si une rfa est spécifiée, PIP fait une liste de chacune des rfn qui satisfont la rfa, en même temps que la copie est faite. Si un fichier possédant le même nom que le fichier de destination existe, il sera remplacé par le fichier copié si la copie a été effectuée avec succès.

Les commandes PIP suivantes donnent des exemples d'opérations valides de copies de disque sur disque :

```
B:=*.COM rc         copie tous les fichiers du "drive" courant qui ont
                    le nom secondaire "COM" sur le "drive" b.

A:=B:ZAP.* rc       copie tous les fichiers du "drive" B qui ont le nom
                    primaire "ZAP" sur le "drive" A.

ZAP.ASM=B: rc       équivalent à ZAP.ASM=B:ZAP.ASM

B:ZOT.COM=A: rc     équivalent à B:ZOT.COM=A:ZOT.COM

B:GAMMA.BAS rc      identique à B:GAMMA.BAS=A:GAMMA.BAS

B:=A:GAMMA.BAS rc   identique à B:GAMMA.BAS=A:GAMMA.BAS
```

PIP autorise aussi des références à des unités physiques et logiques qui sont reliées au CP/M. Les noms de périphériques sont identiques à ceux qui sont donnés sous la commande STAT, avec un nombre de périphériques spécialement nommés. Les unités logiques données dans la commande STAT sont :

CON: (console), RDR: (lecteur), PUN: (perforateur), et LST: (liste)

alors que les unités physiques sont :

TTY: (console, lecteur, perforateur, ou liste)
CRT: (console, ou liste), UC1: (console)
PTR: (lecteur), UR1: (lecteur) UR2: (lecteur)
PTP: (perforateur), UP1: (perforateur), UP2: (perforateur)
LPT: (liste), UL1: (liste)

(Notez que l'unité physique "BAT:" n'est pas inclue, car cet assignation n'est utilisée que pour indiquer que les unités RDR: et LST: devront être utilisées pour l'entrée/sortie console).

Les unités CON, PUN, LST et RDR sont toutes définies dans la portion BIOS du CP/M et peuvent ainsi être modifiées pour n'importe quel système d'E/S particulier. (Le "mapping" du périphérique physique courant est défini par IOBYTE ; celui-ci n'est pas dans le SIL,Z.
Le périphérique de destination doit être capable de recevoir des (i.e., des données peuvent être envoyées à un perforateur), et les périphériques sources doivent être capables de générer des données (i.e., le périphérique LST: ne peut pas être lu).

Les noms de périphériques supplémentaires qui peuvent être utilisés dans les commandes PIP sont :

NUL: Envoie 40 "nuls" (les 0 ASCII) au périphérique (cela peut être émis à la fin d'une sortie perforateur).

EOF: Envoie un fin-de-fichier CP/M (CTRL-Z ASCII) au périphérique de destination (envoyé automatiquement à la fin de chaque transfert de données ASCII par PIP).

OUT: Destination sortie de PIP spéciale qui peut être insérée dans le programme PIP : PIP appelle avec CALL la localisation 106H avec les données dans le registre C pour chaque caractère à transmettre. Notez que les localisations de 109H à 1FFH de l'image mémoire de PIP ne sont pas utilisées et qu'elles peuvent être remplacées par des "drivers" à but spéciaux en utilisant DDT.

PRN: Identique à LST:, sauf qu'il est tenu compte des tabs positionnés tous les huitièmes caractères, que les lignes sont numérotées, que les sauts de page sont insérés toutes les 60 lignes avec un saut initial (semblable à t8np).

Les noms de périphériques et de fichiers peuvent s'entremêler dans les commandes PIP. Dans tous les cas, le périphérique spécifique est lu jusqu'au fin-de-fichier (CTRL-Z pour les fichiers ASCII, et une réelle fin de fichier **pour les fichiers de disque non-ASCII).** Les données de chaque périphérique ou fichier sont concaténées de la gauche vers la droite jusqu'à ce que la dernière source de données soit lue. Le périphérique ou fichier de destination est écrit en utilisant les données des fichiers sources, et un caractère fin-de-fichier (CTRL-Z) est ajouté au résultat pour les fichiers ASCII. Notez que si la destination est un fichier de disque, un fichier temporaire est alors créé (nom secondaire) qui portera le nom de fichier réel seulement si la copie a réussi. Les fichiers avec l'extension "COM" sont toujours considérés comme étant non-ASCII.

L'opération de copie peut être stoppée à tout moment en frappant une touche quelconque sur le clavier (un RUBOUT suffit). PIP répondra avec le message "ABORTED" pour indiquer que l'opération n'a pas été achevée. Notez que si une quelconque opération est stoppée, ou si une erreur survient durant le traitement, PIP enlève toutes les commandes en attente qui ont été envoyées lors de l'utilisation de la commande SUBMIT.

On notera également que PIP effectue une fonction spéciale si la destination est un fichier sur disque de type "HEX" (fichier de code machine Intel formaté hex), et si la source est une unité périphérique externe, comme un lecteur de bande. Dans ce cas, le programme PIP s'assure que le fichier source contient un fichier sous forme hex, avec des valeurs hexadécimales légales et des enregistrements de vérification.
Quand un enregistrement incorrect est trouvé, PIP envoie un message d'erreur sur la console et attend une action de correction. IL suffit généralement d'ouvrir le lecteur et de faire reculer un morceau de bande (d'environ 60 centimètres). Quand la bande est prête à être relue, tapez un retour de chariot sur la console, et PIP essaiera de faire une seconde lecture. Si ce morceau de bande ne peut être lu correctement, continuez simplement la lecture (en tapant un retour chariot après le message d'erreur), et entrez l'enregistrement manuellement avec le programme ED lorsque le fichier sur disque aura été construit.
Pour plus de commodité, PIP permet au fin-de-fichier d'être entré à partir de la console si le fichier source est un périphérique RDR:. Dans ce cas, le programme PIP lit le périphérique et les moniteurs à partir du clavier.

Si CTRL-Z est tapé au clavier, alors l'opération de lecture est terminée normalement.

Les commandes PIP correctes sont les suivantes :

PIP LST:=X.PRN rc	copie X.PRN sur l'unité LST et termine le programme PIP
PIP rc	début du programme PIP en vue d'une suite de lignes de commandes (PIP donne le signal "*").
*CON:=X.ASM,Y.ASM,Z.ASM rc	concatène trois fichiers ASM et copie le tout sur l'unité CON.
*X.HEX=CON:Y.HEX,PTR: rc	crée un fichier HEX en lisant l'unité CON (jusqu'à ce qu'un CTRL-Z soit tapé), puis avec les PTR jusqu'à ce qu'un CTRL-Z soit rencontré.
*rc	un retour de chariot termine PIP
PIP PUN:=NUL:,X.ASM,EOF:,NUL: rc	envoie 40 nuls au périphérique perforateur ; puis copie le fichier X.ASM sur le perforateur suivi par un fin-de-fichier (CTRL-Z) et par 40 caractères nuls supplémentaires.

L'utilisateur peut ausi spécifier un ou plusieurs paramètres PIP, insérés entre crochets, séparés par aucun blanc ou par plusieurs. Chaque paramètre affecte l'opération de copie, et la liste jointe des paramètres doit suivre immédiatement le fichier ou le périphérique affecté. Généralement, chaque paramètre peut être suivi par une valeur entière décimale optionnelle (les paramètres S et Q sont des exceptions). Les paramètres PIP valides sont cités ci-dessous.

B	mode de transfert de bloc : les données sont "stockées" par PIP jusqu'à ce qu'un caractère "x-off" ASCII (CTRL-S) soit reçu du périphérique source. Cela permet le transfert de données à partir d'un périphérique en lecture continue, tel un lecteur de cassette, dans un fichier sur disque. Après réception du "x-off", PIP vide les "buffers" de disque et s'apprête à recevoir d'autres données d'entrée. La quantité de données qui peut être "bufferisée" dépend de la taille de la mémoire du système hôte (PIP fera paraître un message d'erreur si les "buffers" débordent).
Dn	efface les caractères qui dépassent la colonne n dans le transfert de données à partir de la source du caractère vers la destination. Ce paramètre est utilisé le plus souvent pour tronquer de longues lignes qui sont envoyées à une imprimante ou à un périphérique console (peu large).
E	renvoie en écho toutes les opérations de transfert sur la console en même temps qu'elles sont exécutées.
F	filtre les "form feeds" du fichier. Tous les "form feeds" qui ont été insérés sont enlevés. Le paramètre P peut être utilisé simultanément pour insérer de nouveaux "form feeds"

H transfert de données hex : on vérifie si les données sont dans le format du fichier hex d'Intel. Les caractères non-essentiels se trouvant entre des enregistrements hex sont enlevés pendant l'opération de copie. La console sera sollicitée pour une correction au cas où des erreurs surviendraient.

I ignore les enregistrements ":00" dans le transfert de fichier de format hex d'Intel (le paramètre I établit automatiquement le paramètre H).

L transcrit les lettres majuscules en minuscules.

N ajoute un numéro de ligne à chaque ligne transferrée à la destination commençant à un, et incrémentant de 1. Les zéros de début sont supprimés, et le numéro est suivi par deux points. Si N2 est spécifié, les zéros de début sont alors inclus, et un tab est inséré à la suite du nombre. Il est tenu compte du tab si T est établi.

O transfert de fichier objet (non ASCII) : la fin de fichier normale du CP/M est ignorée.

Pn inclut des sauts de page toutes les n lignes (avec un saut de page initial). Si n = 1 ou est exclu, les sauts de page auront lieu toutes les 60 lignes. Si le paramètre F est utilisé, la suppression des "form feeds" se fait avant que les nouveaux sauts de page soient insérés.

Qs z cesse de copier le périphérique ou fichier source quand la chaîne s (terminée par CTRL-Z) est rencontrée.

Ss z commence à copier le périphérique source quand la chaine s est rencontrée (terminée par CTRL-Z). Les paramètres S et Q peuvent être utilisés pour soustraire une section particulière d'un fichier (telle une sous-routine). Les chaînes de départ et d'arrêt sont toujours présentes dans l'opération de copie.

NOTE - les chaines suivant les paramètres s et q sont transformées en majuscules par le CCP si la forme (2) de la commande PIP est utilisée. La forme (1) par contre n'exécute pas automatiquement la traduction en majuscules.
(1) PIP rc
(2) PIP "ligne de commande" rc

Tn tient compte des tabs (caractères CTRL-I) qui sont placés toutes les huit colonnes, pendant le transfert de caractères de la source vers la destination.

U transforme les minuscules en majuscules pendant l'opération de copie.

V vérifie que les données ont été copiées correctement en les relisant après l'opération d'écriture (la destination doit être un fichier sur disque).

Les commandes suivantes sont des commandes PIP valides qui spécifient des paramètres dans le transfert de fichier :

```
PIP X.ASM=B: v rc
```
copie X.ASM du "drive" B sur le "drive" courant et vérifie que les données ont été copiées correctement.

```
PIP LPT:=X.ASM nt8u rc
```
copie X.ASM sur le périphérique LPT: ; numérotte chaque ligne, tient compte des tabs à chaque huitième colonne, et transforme les minuscules en majuscules.

```
PIP PUN:=X.HEX i,Y.ZOT h rc
```
copie d'abord X.HEX sur le périphérique PUN: en ignorant les enregistrements ":00" de fin dans X.HEX ; puis continue le transfert de données en lisant Y.ZOT, qui contient des enregistrements ":00" qu'il contient.

```
PIP X.LIB = Y.ASM  sSUBR1: z qJMP
```
copie le fichier Y.ASM dans le fichier X.LIB. Commence la copie quand une chaine "SUBR1:" a été trouvée, et s'arrête de copier après que la chaine "JMP L3" ait été rencontrée.

```
PIP PRN:=X.ASM p50
```
envoie X.ASM au périphérique LST:, avec des numéros de lignes, des tabs pris en compte toutes les huitièmes colonnes, et dans des sauts de page toutes les 50ièmes lignes. Notez que nt8p60 est la liste de paramètres supposée pour un fichier PRN ; p50 remplace la valeur par défaut.

6.5 ED rfn rc

Le programme ED est l'éditeur de texte du CP/M ; il permet de créer et de modifier des fichiers ASCII dans la configuration du CP/M. Pour avoir l'ensemble des détails sur ED, reportez-vous au manuel de l'utilisateur d'ED. En général, ED permet de créer et de modifier des fichiers sources organisés en suites de caractères ASCII, séparés par des caractères de fin de ligne (retour de chariot et LINE FEED). Il n'y a pas de restriction pratique sur la longueur des lignes (mais aucune ligne ne doit dépasser la taille de la mémoire) ; la longueur de ligne est définie par le nombre de caractères tapés entre deux retours de chariot. Le programme ED a un certain nombre de commandes pour les recherches , les remplacements et les insertions de chaines de caractères, commandes utiles pour la création et la correction de programmes ou de fichier de texte sous CP/M.

Bien que le CP/M ait une source d'espace de travail en mémoire limitée (environ 5000 caractères dans un système CP/M de 16K), la taille d'un fichier qui peut être édité n'est pas limitée, étant donné que les informations peuvent être facilement "paginées" dans cette zone de travail.

Après son lancement, ED crée le fichier source spécifié s'il n'existe pas, et l'ouvre pour y avoir accès. Le programmeur ajoute ensuite des informations qui se trouvent dans le fichier source dans la zone de travail, si le fichier source existe déjà pour l'édition. Les valeurs ajoutées peuvent être affichées, modifiées, et réécrites sur le disque à partir de la zone de travail (voir commande W). Des points particuliers du programme peuvent être automatiquement mis en page et localisés par l'éditeur (voir commande N), ce qui permet un accès facile à certaines portions d'un grand fichier.

Si l'opérateur tape :

```
ED X.ASM rc
```

le programme ED crée un fichier de travail intermédiaire ayant pour nom :

```
X.$$$
```

qui contient les informations éditées pendant l'exécution de ED. Lorsque ED est terminé, le fichier X.ASM (de départ) est appelé X.BAK, et le fichier de travail édité est appelé X.ASM. Ainsi, le fichier X.BAK contient le fichier modifié. L'opérateur peut toujours revenir à la version précédente d'un fichier en enlevant la dernière version, et en changeant le nom de la version précédente. Supposons, par exemple, que le fichier X.ASM ait été incorrectement édité ; la suite de commandes CCP décrites ci-dessous rappellerait le fichier de sauvegarde.

```
DIR X.*           vérifie que le fichier BAK est disponible
ERA X.ASM         efface la version la plus récente
REN X.ASM=X.BAK   change le nom du fichier BAK en ASM
```

Remarquez que l'opérateur peut faire stopper l'édition à n'importe quel moment (en réinitialisant, coupant le courant, tapant un CTRL-C ou une commande Q) sans détruire le fichier d'origine. Dans ce cas, le fichier BAK n'est pas créé, et le fichier d'origine est toujours intact.

Le programme ED permet aussi de créer des fichiers de sauvegarde entre deux disques. La forme de la commande ED est dans ce cas :

```
ED rfn d:
```

où rfn est le nom d'un fichier à éditer sur le disque actuellement loggé, et d le nom d'un "drive" alternatif. Le programme ED lit et traite le fichier source, et écrit le nouveau fichier sur le "drive" d en utilisant le nom rfn. A la fin du traitement, le fichier original devient le fichier de sauvegarde. Ainsi, si l'opérateur s'adresse au disque a, la commande suivante est valide :

```
ED X.ASM B:
```

qui édite le fichier X.ASM sur le "drive" A, créant le nouveau fichier X.$$$ sur le "drive" B. A la fin d'une édition réussie, A:X.ASM est renommé A:X.BAK, et B:X.$$$ est renommé B:X.ASM. Pour plus de commodité, le disque actuellement "loggé" devient le "drive" B à la fin de l'édition. Notez que si un fichier portant le nom B:X.ASM existe avant que l'édition ne commence, le message

```
FILE EXIST
```

est imprimé sur la console pour vous empêcher de détruire accidentellement un fichier source. Dans ce cas, l'opérateur doit d'abord effacer (avec ERA) le fichier existant et ensuite recommencer l'opération d'édition.

Semblable à d'autres commandes transitoires, l'édition peut se faire sur un "drive" différent du disque actuellement loggé en faisant précéder le nom du fichier source par un nom de "drive". Voici des exemples de demandes d'édition valides :

ED A:X.ASM	édite le fichier X.ASM sur le "drive" A, avec le nouveau fichier et la sauvegarde sur le "drive" A.
ED B:X.ASM A:	édite le fichier X.ASM sur le "drive" B dans le fichier temporaire X.$$$ sur le "drive" A. A la fin de l'édition, change X.ASM sur le "drive" B en X.BAK, et change X.$$$ sur le "drive" A en X.ASM.

6.6 SYSGEN rc

Cette fonction n'est pas implantée sur le SIL'Z. Nous avons utilisé le programme COPYSYST.

La commande transitoire SYSGEN permet de générer une disquette initialisée contenant le système d'exploitation du CP/M. Le programme SYSGEN donne un signal sur la console pour recevoir des commandes :

SYSGEN rc — lancement du programme SYSGEN

SYSGEN VERSION m.m — message de début de SYSGEN

SOURCE DRIVE NAME (OR RETURN TO SKIP)
Répondez avec le nom du "drive" (une des lettres A,B, C, ou D) du disque contenant un système CP/M généralement A. Si une copie de CP/M existe déjà en mémoire, due à une commande MOVCPM, tapez seulement un rc. Taper un nom de "drive" x donnera la réponse :

SOURCE ON x THEN TYPE RETURN
placez une disquette contenant le système d'exploitation du CP/M sur le "drive" x (x est l'un des A, B, C, ou D). Répondez par un rc quand vous êtes prêt.

FUNCTION COMPLETE — le système est copié en mémoire. SYSGEN vous répond par :

DESTINATION DRIVE NAME (OR RETURN TO REBOOT)
si une disquette est en cours d'initialisation, placez le nouveau disque dans un "drive" et répondez avec le nom du "drive". Autrement, tapez un rc et le système se réinitialisera à partir du "drive" A. Tapez un nom de "drive" x entraine SYSGEN à afficher :

DESTINATION ON x THEN TYPE RETURN
placez la nouvelle disquette dans le "drive" x, et tapez un RETURN quand vous êtes prêt.

FUNCTION COMPLETE — la nouvelle disquette est initialisée dans le "drive" x.

Le message "DESTINATION" sera répété jusqu'à ce qu'un retour chariot simple soit tapé sur la console, afin que plus d'un disque puisse être initialisé.

A la fin d'une génération réussie de système, la nouvelle disquette contient le système d'exploitation, et seules les commandes fixes sont disponibles. Une disquette arrivée directement de l'usine est compatible IBM apparaît au CP/M comme une disquette avec une "directory" vide ; c'est pourquoi, l'opérateur doit copier les fichiers COM appropriés d'une disquette CP/M existante sur une disquette nouvellement fabriquée en utilisant la transitoire PIP.

L'utilisateur peut copier tous les fichiers d'une disquette existante en tapant la commande PIP

```
PIP B:=A:*.*v  rc
```

qui copie tous les fichiers du "drive" de disque A sur le "drive" de disque B, et vérifie que chaque fichier a été copié correctement. Le nom de chaque fichier est affiché sur la console en même temps que l'opération de copie s'effectue.

Il faut remarquer que SYSGEN ne détruit pas les fichiers existant déjà sur une disquette ; SYSGEN a pour seul résultat la construction d'un nouveau système d'exploitation. Par la suite, si une disquette n'est utilisée que sur le "drive" B, et ne sera jamais la source d'une opération de lancement du système sur le "drive" A, SYSGEN n'a pas besoin d'être utilisé, et, en fait, la nouvelle disquette n'a pas besoin d'aucune initialisation pour être utilisée avec le CP/M.

6.7 SUBMIT rfn parm 1 parm 2 ... parm n rc

La commande SUBMIT permet de traiter par lots les commandes CP/M pour des traitements automatiques. La rfn donnée dans la commande SUBMIT doit être le nom d'un fichier qui existe sur le disque loggé courant, avec un type de fichier déterminé "SUM". Le fichier SUB contient des commandes prototypes CP/M, avec une substitution possible de paramètres. Les paramètres réels parm 1 ... parm n sont substitués dans les commandes prototypes, et, si aucune erreur ne survient, le fichier de commandes substituées est exécuté séquentiellement par CP/M.

Le fichier de commandes prototype est créé en utilisant le programme ED, avec des paramètres "$" de la forme :

```
$1 $2 $3 ... $n
```

correspondant au nombre de paramètres réels qui seront inclus quand le fichier sera soumis à exécution. Lorsque le programme transitoire SUBMIT est exécuté, les paramètres parm 1 ... parm n réels sont "accouplés" avec les paramètres formels $1 $n des commandes prototypes. Si les nombres de paramètres formels et réels ne correspondent pas, la fonction SUBMIT est stopée et apparaît un message d'erreur à la console. La fonction SUBMIT crée un fichier de commandes substituées portant le nom :

```
$$$.SUB
```

sur le disque loggé. Quand le système se réinitialise (à la fin de SUBMIT), ce fichier de commande est lu par le CCP comme source d'entrée, à la place de la console. Si la fonction SUBMIT est effectuée sur un autre disque que celui du "drive" A, les commandes ne seront traitées que lorsque le disque aura été inséré dans le "drive" A, et le système réinitialisé.

L'utilisateur peut stopper le traitement dans n'importe quel cas en tapant un "rubout" quand la commande est lue et renvoyée en écho. Dans ce cas, le fichier $$$.SUB est enlevé, et les commandes ultérieures seront prises à partir de la console. Le traitement de commandes est aussi stoppé si le CCP détecte une erreur dans l'une des commandes. Le programme exécuté sous CP/M peut stopper le traitement de fichiers de commandes lorsque des conditions d'erreur apparaissent en effaçant simplement tous les fichiers $$$.SUB.

Pour introduire des symboles dollars dans un fichier SUBMIT, l'utilisateur peut taper un "$$" qui se transforme en un "$" unique à l'intérieur du fichier de comandes. De plus, un symbole flèche orientée vers le haut "↑" peut précéder un caractère alphabétique x, ce qui crée un caractère unique CTRL-X à l'intérieur du fichier.

La dernière commande d'un fichier SUB peut lancer un autre fichier SUB permettant ainsi des traitements par lots de commandes en chaîne.

Supposons que le fichier ASMBL.SUB existe sur disque, et contienne les commandes prototypes :

```
ASM $1
DIR $1.*
ERA *.BAK
PIP $2:=$1.PRN
ERA $1.PRN
```

et que la commande :

```
SUBMIT ASMBL X PRN  rc
```

soit envoyée par l'opérateur. Le programme SUBMIT lit le fichier ASMBL.SUB, et substitue "X" partout où apparaît $1, et "PRN" partout où apparaît $2, ce qui donne un fichier $$$.SUB contenant :

```
ASM X
DIR X.*
ERA *.BAK
PIP PRN:=X.PRN
ERA X.PRN
```

et ces commandes sont exécutées dans l'ordre par le CCP.

La fonction SUBMIT peut accéder à un fichier SUB qui se trouve sur un "drive" alternatif, si un nom de "drive" précède le nom du fichier. Les fichiers soumis ne sont traités que lorsqu'ils apparaissent sur le "drive" A. Ainsi, il est possible de créer un fichier soumis sur le "drive" B qui sera exécuté plus tard quand il sera inséré dans le "drive" A.

6.8 DUMP rfn rc

Le programme DUMP affiche le contenu d'un fichier sur disque déterminé par rfn sur la console sous forme hexadécimale. Le contenu du fichier est affiché par lignes de seize octets, l'adresse absolue étant donnée à gauche de chaque ligne en hexadécimal. Les affichages trop longs peuvent être interrompus en tapant sur la touche "rubout" pendant l'impression. (le listing source du programme DUMP est donné dans le "Guide de l'interface du CP/M", comme exemple de programme écrit pour l'environnement CP/M).

6.9 MOVCPM rc

Ce programme ne figure pas sur la disquette du SIL'Z. En effet il n'existe qu'en version de 64K mémoire

Le programme MOVCPM permet à l'utilisateur de reconfigurer le système CP/M pour toute taille de mémoire particulière. Deux paramètres optionnels peuvent être utilisés pour indiquer (1) la taille désirée du nouveau système et (2) la disposition du nouveau système à la fin d'un programme. Si le premier paramètre est omis ou si un "*" est donné, le programme MOVCPM reconfigurera le système avec sa taille maximum, basée sur les kilo-octets de la mémoire RAM contigüe dans le système hôte (commençant à 0000H). Si le second paramètre est omis, le système exécuté est laissé en mémoire, prêt pour une opération SYSGEN. Le programme MOVCPM relocalise une image mémoire de CP/M et place cette image en mémoire pour préparer une opération de génération de système.
Les formes de la commande sont :

MOVCPM rc	relocalise et exécute CP/M pour gérer la configuration de mémoire courante (la mémoire est examinée pour trouver la mémoire RAM contigüe commençant à 100H). A la fin de la relocalisation, le nouveau système est exécuté mais non enregistré en permanence sur la disquette.
MOVCPM n rc	crée un système CP/M relocalisé pour gérer un système à n kilooctets (n doit être compris entre 16 et 64), et exécute le système, comme décrit ci-dessus.
MOVCPM * * rc	construit une image mémoire relocalisée pour la configuration de mémoire courante, mais laisse l'image mémoire en mémoire, pour préparer une opération SYSGEN.

La commande

MOVCPM * *

par exemple, construit une nouvelle version du système CP/M et la laisse en mémoire, prête pour une opération SYSGEN.
Le message :

READY FOR "SYSGEN" OR
"SAVE 32 CPMxxx.COM"

est affiché sur la console après réalisation, où xx est la taille de mémoire courante en kilo-octets. L'opérateur peut alors taper :

SYSGEN rc — commence la génération du système

SOURCE DRIVE NAME (OR RETURN TO SKIP)
Répondez par un retour chariot pour éviter l'opération de lecture CP/M puisque le système est déjà en mémoire depuis l'opération antérieure MOVCPM.

DESTINATION DRIVE NAME (OR RETURN TO REBBOT)
Répondez par B pour écrire le nouveau système sur la disquette dans le "drive" b. SYSGEN vous répondra par :

DESTINATION ON B, THEN TYPE RETURN
Insérez la nouvelle disquette dans "drive" B et tapez un RETURN quand vous êtes prêt.

Notez que si vous répondez par "A" à la place de "B" à la question précédente, le système sera écrit sur le "drive" A au lieu d'être écrit sur le "drive" B. SYSGEN continuera de taper le message :

DESTINATION DRIVE NAME (OR RETRUN TO REBOOT)

jusqu'à ce que l'opérateur réponde par un retour chariot simple, qui stoppe le programme SYSGEN par une réinitialisation du système.

L'utilisateur peut alors procéder à une réinitialisation avec l'ancienne ou la nouvelle disquette. Au lieu d'exécuter l'opération SYSGEN, l'utilisateur aurait pu taper :

SAVE 32 CPMxx.COM

après achèvement de la fonction MOVCPM, ce qui aurait placé l'image mémoire CP/M sur le disque actuellement loggé dans une forme qui peut être modifiée. Cela est nécessaire quand l'opération a lieu dans un environnement non standard où le BIOS doit être modifié pour une configuration particulière de périphérique, comme il est décrit dans le "Guide de modification du CP/M".

Voici les commandes MOVCPM valides :

MOVCPM 48 rc	construit une version à 48K du CP/M et commence l'exécution.
MOVCPM 48 * rc	construit une version à 48K du CP/M pour préparer l'enregistrement permanent ; la réponse est : READY FOR "SYSGEN" OR "SAVE 32CPM48.COM"
MOVCPM * * rc	construit une version de mémoire maximum du CP/M et commence l'exécution.

Il est important de noter que le système nouvellement créé est publié avec le numéro attaché à la disquette originale et qu'il est soumis aux conditions du Contrat de Licence Soft de Digital Research.

6.10 LA FONCTION XSUB

Un programme utilitaire supplémentaire est fourni avec la version 2.0 du CP/M, appelé XSUB, qui augmente la puissance de la commande SUBMIT en permettant de faire des entrées de lignes dans des programmes comme dans le processeur de commande console. La commande XSUB est inclue comme la première ligne de votre fichier soumis et, lorsqu'elle est exécutée, se relocalise elle-même directement en dessous du CCP. Toutes les lignes de commande soumises suivantes sont traitées par XSUB, afin que les programmes qui lisent des entrées console "bufferisées" (fonction 10 du BDOS) reçoivent leurs entrées directement du fichier soumis. Par exemple, le fichier SAVER.SUB pourrait contenir les lignes soumises :

```
XSUB
DDT
I 1.HEX
R
GO
SAVE 1  2.COM
```

avec une commande SUBMIT suivante :

```
SUBMIT SAVER X Y
```

qui substitue X à 1 et Y à 2 dans le courant de la commande. Le programme XSUB charge, suivi par DDT auquel sont envoyées les lignes de commandes "IX.HEX" "R" et "GO" retournant ainsi au CCP. La commande finale "SAVE 1 Y.COM" est traitée par le CCP.

Le programme XSUB reste en mémoire, et imprime le message :

```
(xsub active)
```

à chaque opération de "départ à chaud" pour indiquer sa présence. Les commandes de soumission suivantes n'exigent pas le XSUB, à moins qu'un "départ à froid" ne soit survenu. Remarquez que XSUB doit être chargé après DESPOOL, s'ils doivent tous les deux être traités simultanément.

7. MESSAGES D'ERREUR BDOS

Il existe trois situations d'erreur que le système d'exploitation du disque de base intercepte pendant le traitement de fichier. Quand l'une de ces conditions est détectée, le BDOS affiche le message :

```
BDOS ERR ON x: error
```

où x est le nom du drive, et "error" l'un des trois messages d'erreur :

```
BAD SECTOR
SELECT
READ ONLY
```

Le message "BAD SECTOR" indique que le contrôleur de disque a détecté une condition d'erreur en lisant ou en écrivant la disquette. Cette condition est généralement dûe à un mauvais fonctionnement du contrôleur disque ou à une disquette extrêmement usagée.
Si vous trouvez que le système rapporte cette erreur plus souvent qu'il convient, vous devez vérifier l'état de votre contrôleur et de votre moyen de communication.
Dans tous les cas, le rétablissement après la condition d'erreur est obtenu en tapant un CTRL-C pour réinitialisation (c'est le plus prudent !), ou un RETURN, qui ignore simplement le mauvais secteur dans l'opération sur fichier. Notez,cependant, que le fait de taper un RETURN peut détruire l'intégrité de votre disquette si l'opération est une écriture de "catalogue" ; aussi assurez-vous que vous possédez des copies de sauvegarde dans ce cas.

L'erreur "SELECT" survient quand on tente de s'adresser à un "drive" entre A et D. Dans ce cas, la valeur de x dans le message d'erreur donne le "drive" sélectionné. Le système s'initialise après toute entrée à partir de la console.

Le message "READ ONLY" apparaît lorsqu'on tente d'écrire sur une disquette désignée comme "lue seulement" dans une commande STAT, ou qui a été établie à "lecture seulement" par le BDOS. En général, l'opérateur devrait réinitiaiser le CP/M soit en utilisant la procédure de "départ à chaud" (CTRL-C) soit en exécutant un "départ à froid" chaque fois que les disquettes sont changées. Si une disquette changée doit être lue mais non écrite, BDOS permet de la changer sans faire de "départ à froid" ou "à chaud", mais marque intérieuremet le "drive" en "lecture seulement". Le statut du "drive" est successsivement changé en écriture/lecture si un "départ à chaud" ou"à froid" survient. Après avoir fait paraître ce message CP/M attend une entrée à partir de la console. Un "départ à chaud" automatique a lieu après toute entrée.

Les disquettes peuvent êtres enlevées de leur "drive" à n'importe quel moment, et le système peut être coupé pendant une opération sans affecter l'intégrité des informations. Remarquez, cependant, que l'utilisateur ne doit pas enlever une disquette et la remplacer par une autre, sans réinitialiser (boot) le système (départ à froid ou à chaud), sauf si la disquette est seulement à lire.

Dans le cas de pannes ou de mauvais fonctionnement du "hardware", le CP/M peut inscrire le message :

BDOS ERR ON x: BAD SECTOR

où x est le nom du disque où se trouve l'erreur permanente.

Cette erreur peut apparaître quand les portes du "drive" sont ouvertes ou fermées à mauvais escient, au moment d'opérations sur le disque ou peut être due à des défauts sur la disquette, le "drive" ou le contrôleur. L'utilisateur peut choisir de ne pas tenir compte de l'erreur en tapant un **RETURN** sur la console. L'erreur peut produire un mauvais enregistrement, nécessitant une réinitialisation des 128 octets

(ou moins) de l'enregistrement. L'opérateur peut réinitialiser le système CP/M et essayer l'opération une nouvelle fois.

Pour arrêter le CP/M, il ne faut rien faire de spécial, bien qu'il soit nécessaire d'enlever les disquettes avant de couper le courant, pour éviter des écritures intempestives qui pourraient survenir.

TABLE DES MATIERES

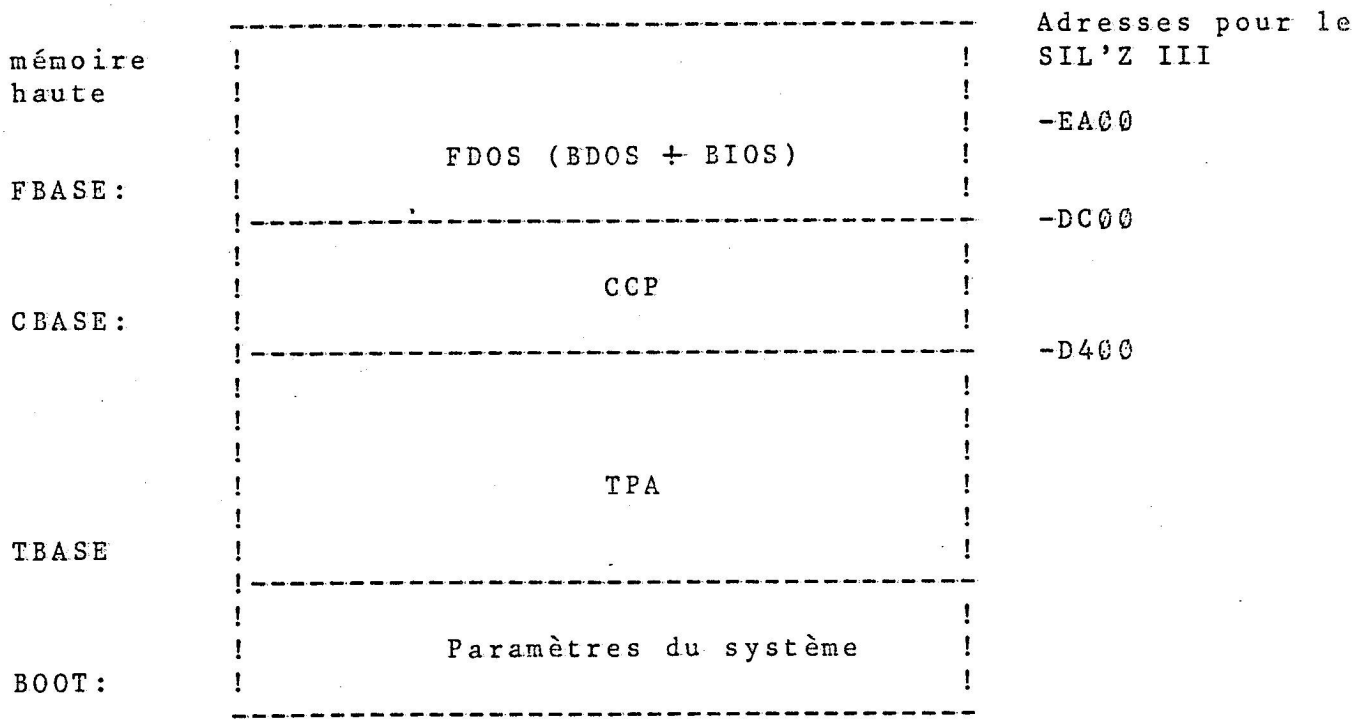
1. INTRODUCTION

Ce manuel décrit l'organisation du système CP/M, la structure de la mémoire et les points d'entrée du système. Le but en est de donner les informations nécessaires à l'écriture des programmes qui opèrent sous CP/M, et qui utilisent les opérations d'E/S avec les disques et les périphériques.

CP/M est logiquement divisé en quatre parties, appelées :

- le système E/S de base BIOS
- le système d'exploitation de base du disque (BDOS)
- le processeur de commande console (CCP)
- la zone de programme transitoire (TPA)

Le BIOS est un module dépendant du "hardware" qui définit l'interface de bas niveau exacte d'un système particulier, nécessaire pour l'E/S d'unités périphériques. Le BIOS et le BDOS sont logiquement combinés dans un module unique avec un point d'entrée commun, et nommé FDOS. Le CCP est un programme distinct qui utilise le FDOS pour fournir une interface orientée-utilisateur aux informations cataloguées sur le périphérique de stockage. La TPA est une zone mémoire (i,e., la portion qui n'est pas utilisées par le FDOS et le CCP) où diverses commandes du système d'exploitation non résident et des programmes utilisateurs sont exécutés. La partie inférieure de la mémoire est réservée aux informations système et est détaillée dans les sections ci-dessous. L'organisation de la mémoire du système CP/M est présentée ci-dessous :



Toutes les versions CP/M standard supposent que BOOT = 0000H, ce qui correspond à la base de la mémoire d'accès direct. Le code machine trouvé à l'adresse BOOT exécute un "départ à chaud" du système qui charge et initialise les programmes et variables nécessaires pour rendre le contrôle au CCP. Ainsi, les programmes transitoires n'ont qu'à se brancher à l'adresse BOOT pour rendre le contrôle au CP/M au niveau de commande. De plus, les versions standard décident que TBASE = BOOT + 0100H qui correspond normalement à l'adresse 0100H. Le point d'entrée principal au FDOS se trouve à l'adresse BOOT + 0005H (normalement 0005H) où un saut à FBASE est trouvé. Le champ d'adresse à BOOT + 0006H (normalement 0006H) contient la valeur de FBASE et peut être utilisé pour déterminer la **taille de la mémoire disponible** en supposant que le CCP est recouvert par un programme transitoire.

Des programmes transitoires sont chargés dans la TPA et exécutés de la manière suivante. L'opérateur communique avec le CCP en tapant des lignes de commandes après chaque caractère de signal. Chaque ligne de commande prend l'une des formes :

```
commande
commande fichier 1
commande fichier 1 fichier 2
```

où "commande" est soit une fonction figée telle que DIR ou TYPE, soit le nom d'une commande ou d'un programme transitoire.
Si la commande est une fonction figée du CP/M, elle est exécutée immédiatement ; sinon le CCP parcourt le disque courant, à la recherche

d'un fichier portant le nom :

```
commande. COM
```

Si le fichier est trouvé, il est supposé être l'image mémoire d'un programme qui s'exécute dans la TPA, et commence ainsi implicitement à TBASE en mémoire. Le CCP charge le fichier COM à partir du disque dans la mémoire en commençant à TBASE, avec la possibilité de l'étendre jusqu'à CBASE.

Si la commande est suivie par une ou deux spécifications de fichier, le CCP prépare un ou deux noms de blocs de contrôle de fichier (FCB) dans la zone des paramètres du système. Ces noms de FCB optionnels ont la forme requise pour accéder à des fichiers par le FDOS, et sont décrits dans la section suivante.

Le programme transitoire reçoit le contrôle du CCP et commence l'exécution, éventuellement en utilisant les possibilités d'E/S du FDOS. Le programme transitoire est "appelé" du CCP, et peut ainsi retourner simplement au CCP après achèvement de son exécution, ou sauter jusqu'à BOOT pour redonner le contrôle au CP/M. Dans le premier cas, le programme transitoire ne doit pas utiliser de la mémoire au dessus de CBASE, alors que dans le dernier cas, la mémoire allant jusqu'à FBASE-1 est libre.

Le programme transitoire peut utiliser les possibilités d'E/S du CP/M pour communiquer avec la console de l'opérateur et les unités périphériques y compris le sous-système disque. On accède au système d'E/S en donnant au CP/M un "numéro de fonction" et une "adresse d'information" au point d'entrée FDOS à BOOT + 0005H. Dans le cas d'une lecture sur disque, par exemple, le programme transitoire envoie au FDOS du CP/M, un nombre correspondant à une lecture sur disque et l'adresse d'un FCB. Le FDOS, à son tour, effectue l'opération, en renvoyant soit une indication de fin de lecture sur disque, soit un numéro d'erreur indiquant que l'opération n'a pas réussi. Les numéros des fonctions et les indicateurs d'erreurs sont donnés ci-dessous.

2. CONVENTIONS SUR LES APPELS DU SYSTEME D'EXPLOITATION

Le but de cette section est de vous fournir des informations détaillées sur l'exécution d'appels directs du système d'exploitation depuis des programmes utilisateurs.

Les caractéristiques du CP/M auxquelles on peut avoir accès par des programmes transitoires sont divisées en deux catégories : les E/S simples et les E/S disque. Les opérations sur périphérique simple incluent :

lecture d'un caractère sur console
écriture d'un caractère sur console
lecture d'un caractère séquentiel d'une bande magnétique
écriture d'un caractère sur bande séquentielle
écriture d'un caractère sur périphérique d'impression
obtention ou mise en place de l'état d' E/S
impression du "buffer" de console
lecture du "buffer" de console
interrogation sur l'état de la console

Les opérations sur FDOS qui exécutent des Entrées/Sorties sur le disque sont :

réinitialisation du système disque
sélection de "drive"
création de fichier
ouverture de fichier
fermeture de fichier
recherche dans le "catalogue"
destruction de fichier
changement de nom de fichier
lecture directe ou séquentielle
écriture directe ou séquentielle
interrogation sur disque choisi
mise en place de l'adresse DMA
indicateurs de mise en place/remise en place de fichier

Comme il est mentionné ci-dessus, l'accès aux fonctions FDOS est effectué en donnant un numéro de fonction et une adresse d'information par le point d'entrée primaire à l'adresse BOOT + 0005H. En général, le numéro de fonction est mis dans le registre C, et l'adresse d'information dans le couple de registres doubles. Les valeurs à octet simple sont retournées dans le registre A, et les valeurs à double octets dans HL (une valeur zéro est retournée quand le numéro de la fonction est en dehors des limites). Pour des raisons de compatibilité, dans tous les cas lors du retour, le registre A = L et le registre B = H.

La liste des numéros des fonctions CP/M est donnée ci-dessous :

	0	réinitialisation du système
	1	entrée sur console
	2	sortie sur console
	3	entrée sur lecteur
	4	sortie sur perforateur
	5	sortie de "listing"
	6	E/S directe sur console
	7	obtention de l'octet d'E/S
	8	mise en place de l'octet d'E/S
	9	impression de chaîne
	10	lecture d'un "buffer" sur console
	11	obtention de l'état de la console
	12	renvoi du numéro de la version
	13	réinitialisation du système disque
	14	sélection de disque
	15	ouverture de fichier
	16	fermeture de fichier
	17	recherche du premier
	18	recherche du suivant
	19	destruction de fichier
	20	lecture séquentielle
	21	écriture séquentielle
	22	création d'un fichier
	23	changement de nom d'un fichier
	24	renvoi du vecteur "login"
	25	renvoi du disque courant
	26	mise en place de l'adresse DMA
	27	obtention de l'adresse (allocation)
*	28	protection du disque contre l'écriture
*	29	obtention du vecteur lecture/seulement (R/O)
	30	mise en place/obtention du code utilisateur
	31	obtention de l'adresse (paramètres disque)
*	32	mise en place/obtention du code utilisateur
	33	lecteur directe
	34	écriture directe
	35	calcul de la taille du fichier
	36	mise en place d'un enregistrement direct

(Les fonctions 28, 29 et 32 doivent être évitées dans les programmes d'application afin de conserver en amont la compatibilité avec MP/M).

Après l'entrée dans un programme transitoire, le CCP laisse le pointeur de pile établi à une zone de pile de huit niveaux, avec l'adresse de retour du lieu. Bien que cette ne soit pas utilisée habituellement par un programme transitoire (i.e., la plupart des transitoires retournant au CCP par un saut à l'adresse 0000H), elle est suffisamment grande pour faire des appels au système CP/M puisque le FDOS se branche sur une pile locale lors de l'initialisation.

Le segment de programme en assembleur suivant, par exemple, lit les caractères continuellement jusqu'à ce qu'il rencontre un astéristique, le contrôle retournant à ce moment au CCP (supposant un système CP/M standard avec BOOT = 0000H):

```
BDOS    EQU     0005H       ;ENTREE CP/M STANDARD
CONIN   EQU     1           ;FONCTION ENTREE SUR CONSOLE
;
        ORG     0100H       ;BASE DE LA TPA
NEXTC:  MVI     C,CONIN     ;LECTURE DU CARACTERE SUIVANT
        CALL    BDOS        ;MET LE CARACTERE DANS <A>
        CPI     '*'         ;FIN D'EXECUTION?
        JNZ     NEXTC       ;SI NON BOUCLE
        RET                 ;RETOUR AU CCP
        END
```

Le CP/M implante la structure d'un fichier ayant un nom sur chaque disque, donnant une organisation logique qui permet à n'importe quel fichier de contenir n'importe quel nombre d'enregistrements ; on peut donc tout avoir, depuis un fichier complètement vide jusu'à un fichier utilisant toute la capacité du "disque".

Chaque disque est logiquement distinct, avec un "catalogue" de disque, et une zone de données pour les fichiers. Les noms des fichiers sur disque comportent trois parties : le code de sélection de "drive", le nom de fichier constitué de un à huit caractères non blancs, et le type de fichier contenant de zéro à trois caractères non blancs. Le type de fichier détermine la catégorie générale d'un fichier particulier, tandis que le nom de fichier détermine des fichiers individuels dans chaque catégorie. Les types de fichier réunis ci-dessous donnent les catégories qui ont été établies, bien qu'elles soient généralement arbitraires :

ASM	source assembleur	PLI	fichier source PL/I
PRN	fichier pour impression	REL	module relogeable
HEX	code machine hexa	TEX	source pour le formatteur TEX
BAS	fichier source basic	BAK	fichier de sauvegarde ED
INT	code intermédiaire	SYM	fichier des symboles SID
COM	fichier de commandes pour CCP	$$$	fichier temporaire

Les fichiers sources sont considérés comme des suites de caractères ASCII, où chaque "ligne" du fichier source est terminée par une séquence retour-chariot "line-feed" (0DH suivi par 0AH). Ainsi, un enregistrement de 128 octets du CP/M peut contenir plusieurs lignes du texte source.

La fin d'un fichier est signifiée par un caractère CTRL-Z (1AH) ou par une fin de fichier réelle, retournée par une opération de lecture CP/M. Les caractères CTRL-Z inclus dans des fichiers en code machine (e.g., fichiers COM) sont ignorés cependant, et la condition de fin de fichier retournée par CP/M est utilisée pour terminer les opérations de lecture.

Les fichiers dans le CP/M peuvent être présentés comme des séries de 65536 enregistrements maximum de 128 octets chacun, numérotés de 0 à 65535, permettant ainsi un maximum de **8 mégaoctets** par fichier.

Notez, cependant, que bien que les enregistrements puissent être logiquement considérés comme contigüs, ils peuvent ne pas l'être logiquement dans la zone des données sur disque. A l'intérieur, tous les fichiers sont fractionnés en segments de 16K octets appellés des extensions logiques, afin que les compteurs soient facilement conservés comme des valeurs 8 bits. Bien que la décomposition en extensions soit présentée dans les paragraphes qui suivent, elle n'a pas de conséquences pour le programmeur puisqu'on accède automatiquement à chaque extension en mode d'accès direct et séquentiel.

Dans les opérations sur fichier commençant avec la fonction numéro 15, DE s'adresse généralement à un bloc de contrôle de fichier (FCB). Les programmes transitoires utilisent souvent par défaut la zone de blocs de contrôle de fichier réservée par CP/M à l'adresse BOOT + 005CH (normalement 005CH) pour des opérations simples sur fichier. L'unité de base des informations de fichier est un enregistrement de 128 octets utilisé pour toutes les opérations sur fichier ; ainsi, CP/M fournit une adresse par défaut pour le disque E/S à l'adresse BOOT + 0080H (normalement 0080H) qui est l'adresse DMA initiale par défaut (voir fonction 26).
Toutes les opérations sur "directory" ont lieu dans une zone réservée qui n'affecte pas l'écriture des "buffers".

La zone des données des blocs de contrôle de fichier (FCB) consiste en une série de 33 octets pour un accès séquentiel et en une série de 36 octets dans le cas d'un accès direct à un fichier. Le bloc de contrôle de fichier par défaut normalement localisé à 005CH peut être utilisé pour des fichiers d'accès direct, puisque les trois octets commençant à BOOT + 007DH sont disponibles pour ce propos. Le format FCB est exposé avec les champs suivants :

dr	f1	f2	/	/	f8	t1	t2	t3	ex	s1	s2	rc	d0	/	/	dn	cr	r0	r1	r2
00	01	02	...		08	09	10	11	12	13	14	15	16	...		31	32	33	34	35

où

dr — code du "drive" (0 - 16)
0 => utilisez le "drive" par défaut pour le fichier
1 => sélection automatique du "drive" A
2 => sélection automatique du "drive" B
...
16 => sélection automatique du "drive" P

f1...f8 — contient le nom de fichier en ASCII majuscules, avec le bit haut = 0

t1,t2,t3 — contient le type de fichier en ASCII majuscules, avec le bit haut = 0
t1',t2', et t3' dénotent le bit de ces positions
t1' = 1 => fichier Lecture/Seulement
t2' = 1 => fichier SYS, pas de "listing" sous DIR

ex contient le numéro de l'extension courante, établi normalement à 00 par l'utilisateur, mais dans les limites 0 - 31 pendant l'E/S de fichier

s1 réservé pour une utilisation interne du système

s2 réservé pour une utilisation interne du système, établi à zéro sur les appels de OPEN, MAKE, SEARCH

rc compte d'enregistremet pour l'extension "ex," prend les valeurs de 0 - 128

d0...dn rempli par CP/M, réservé pour l'utilisation du système

cr enregistrement courant à lire ou à écrire dans une opération sur fichier séquentiel, établi normalement à zéro par l'utilisateur

r0,r1,r2 numéro d'enregistrement direct optionnel dans les limites 0 - 65535, avec le recouvrement de r2,r0,r1 constituent une valeur 16 bits avec l'octet bas r0, et l'octet haut r1

Chaque fichier auquel on a eu accès par le CP/M doit avoir un FCB correspondant qui fournit des informations sur le nom et l'emplacement pour toutes les opérations suivantes sur le fichier. Lors d'un accès à des fichiers, c'est au programmeur de remplir les 16 octets les plus bas du FCB et d'initialiser le champ "cr". Normalement, les octets 1 à 11 sont mis aux valeurs des caractères ASCII pour le nom de fichier et le type de fichier, alors que tous les autres champs sont **nuls** .

Les fichiers du FCB sont stockés dans la zone de la "directory" du disque, et sont amenés en mémoire centrale avant le traitement des opérations sur les fichiers (voir les fonctions OPEN et MAKE).
La copie mémoire du FCB est mise à jour au cours du traitement des opérations sur fichiers, et enregistrée plus tard en permanence sur le disque à la fin du traitement de fichier (voir la commande CLOSE).

Le CCP construit les 16 premiers octets de deux fichiers optionnels de FCB pour un programme transitoire en analysant le restant de la ligne suivant le nom du programme transitoire, signifié par "fichier1" et "fichier2" dans la ligne de commandes prototype décrite ci-dessus, avec des champs non spécifiés considérés comme des blancs ASCII. Le premier FCB est construit à l'adresse BOOT+005CH, et peut être utilisé comme tel pour les opérations sur fichier suivantes. Le second FCB occupe la portion d0 ... dn du premier FCB, et doit être déplacé jusqu'à une autre zone de mémoire pour être utilisé. Si, par exemple, l'opérateur tape :

```
PROGNOM B:X.ZOT Y.ZAT
```

Le fichier PROGNOM.COM est chargé dans la TPA, et le FCB par défaut situé à BOOT+005CH est initialisé au code de "drive" 2, au nom "X" et au type "ZOT". Le second code de "drive" prend la valeur par défaut 0, qui est placée à BOOT+006CH, avec le nom de fichier "Y" placé dans l'adresse BOOT+0075H. Tous les champs restants jusqu'à "cr" sont fixés à zéro. Notez encore que c'est au programmeur de déplacer ce second nom de fichier et de le taper dans une autre zone, généralement un bloc de contrôle de fichier différent, avant d'ouvrir le fichier qui commence à BOOT+005CH, cela à cause du fait que l'opération d'ouverture recouvrirait le second nom et le second type.

Si aucun nom de fichier n'est spécifié dans la première commande, les champs commençant à BOOT+005DH et à BOOT+006DH contiennent des blancs. Dans tous les cas, le CCP passe des minuscules aux majuscules, afin de rester conforme aux conventions d'appellation des fichiers CP/M.

Outre les caractéristiques de programmation, la zone du "buffer" par défaut située à l'adresse BOOT+0080H est initialisé à l'extrémité de la ligne de commande tapée par l'opérateur, à la suite du nom de programme. La première position contient le nombre de caractères, et les caractères eux-mêmes suivent ce nombre. Etant donné la commande ci-dessus, la zone qui commence à BOOT+0080H est initialisée ainsi :

BOOT + 0080H :

+00	+01	+02	+03	+04	+05	+06	+07	+08	+09	+10	+11	+12	+13	+14
14	"	"	"B"	":"	"X"	"."	"Z"	"O"	"T"	"	"	"Y"	"."	"Z"	"A"	"P"

où les caractères sont transcrit en majuscules ASCII avec une mémoire non initialisée suivant le dernier caractère valide. C'est à nouveau au programmeur d'extraire les informations de ce "buffer" avant toute opération, à moins que l'adresse DMA par défaut soit explicitement chargée.

Les fonctions individuelles sont décrites en détail dans les pages qui suivent.

FONCTION 0 : REINITIALISATION DU SYSTEME

Paramètres d'entrée :
Registre C : 00H

La fonction de réinitialisation du système retourne le contrôle au système d'exploitation du CP/M au niveau du CCP. Le CCP réinitialise le sous-système disque en sélectionnant et en "loggant" le "drive" A du disque. Cette fonction a exactement le même effet qu'un saut jusqu'à l'adresse BOOT.

FONCTION 1 : ENTREE SUR CONSOLE

Paramètres d'entrée :
Registre C : 01H

Valeur retournée :
Registre A : caractère ASCII

La fonction d'entrée sur console lit le prochain caractère console dans le registre A. Les caractères graphiques, ainsi que les retour-chariot, LINE-FEED, et BACKSPACE (CTRL-H) sont renvoyés en écho sur la console. Les caractères de tabulation (CTRL-I) sont situés tous les huit colonnes. Une vérification est faite pour savoir si un contrôle-S sur le terminal ou un contrôle-P sur l'imprimante a été frappé.
Le FDOS ne retourne pas au programme d'appel avant qu'un caractère ne soit tapé ; il suspend ainsi l'exécution si un caractère n'est pas prêt.

FONCTION 2 : SORTIE SUR CONSOLE

Paramètres d'entrée :
Registre C: 02H
Registre E : caractère ASCII

Le caractère ASCII du registre E est envoyé au périphérique console. De la même façon que pour la fonction 1, les tabs sont pris en compte, et des tests sont faits pour savoir si un CTRL-P est tapé.

FONCTION 3 : ENTREE SUR LECTEUR

Paramètres d'entrée :
Registre C : 03H

Valeur retournée :
Registre A : caractère ASCII

La fonction d'entrée sur lecteur lit le prochain caractère à partir du lecteur logique dans le registre A.
Le contrôle n'est pas redonné au système d'exploitation avantque le le caractère ne soit lu.

Attention ! Cette fonction fait intervenir l'IOBYTE qui n'est pas implanté dans le SIL'Z.

FONCTION 4 : SORTIE SUR PERFORATEUR

Paramètres d'entrée :
Registre C : 04H
Registre E : caractère ASCII

La fonction de sortie sur perforateur envoie le caractère du registre E au périphérique perforateur logique.

FONCTION 5 : SORTIE DE LISTING

Paramètres d'entrée :
Registre C : 05H
Registre E : caractère ASCII

La fonction de sortie de listing envoie le caractère ASCII du registre E au périphérique d'impression logique.

FONCTION 6 : E/S CONSOLE DIRECTE

Paramètres d'entrée :
Registre C : 06H
Registre E : 0FFH (entrée) ou
car (sortie)

Valeur retournée :
Registre A : car ou état
(aucune valeur)

L'E/S console directe est acceptée sous le CP/M pour ces applications spéciales dans lesquelles l'entrée et la sortie console sont exigées. L'utilisation de cette fonction devrait être évitée en général puisqu'elle ne se soumet pas à toutes les autres fonctions normales de caractères de contrôle du CP/M (par exemple, le CTRL-S et le CTRL-P).

Après l'entrée de la fonction 6, le registre E contient soit des FF hexadécimaux, ce qui dénote une demande d'entrée console, soit un caractère ASCII. Si la valeur d'entrée est FF, la fonction 6 retourne alors A = 00 si aucun caractère n'est prêt ; sinon A contient le prochain caractère d'entrée console.

Si la valeur d'entrée dans E n'est pas FF, la fonction 6 décide alors que E contient un caractère ASCII valide qui est envoyé à la console.

FONCTION 7 : OBTENTION DE L'OCTET D'E/S

Paramètres d'entrée :
Registre C : 07H

Valeur retournée :
Registre A : valeur de l'octet d'E/S

La fonction d'obtention de l'octet d'E/S retourne la valeur courante de l'IOBYTE dans le registre A. Cette valeur est toujours nulle dans le SIL'Z.

FONCTION 8 : MISE EN PLACE DE L'OCTET D'E/S

Paramètres d'entrée :
Registre C : 08H
Registre E : valeur de l'octet d'E/S

La fonction de mise en place de l'octet d'E/S transforme la valeur de l'IOBYTE du système en celle qui est donnée dans le registre E.

FONCTION 9 : IMPRESSION DE CHAINE

Paramètres d'entrée :
Registre C : 09H
Registre DE : adresse de la chaîne

La fonction d'impression de chaîne envoie la chaîne de caractères stockée en mémoire à l'adresse donnée par DE, à la console, jusqu'à ce qu'un "$" soit rencontré dans la chaîne. Les tabs sont pris en considération comme dans la fonction 2, et les tests sont égalements faits.

FONCTION 10 : LECTURE D'UN "BUFFER" SUR CONSOLE

Paramètres d'entrée :
Registre C : 0AH
Registres DE : adresse du "buffer"

Valeur retournée :
caractères console dans le "buffer"

La fonction de lecture de "buffer" lit une ligne d'entrée console éditée dans un "buffer" adressé par les registres DE. L'entrée console est terminée quand l'un ou l'autre des "buffers" d'entrée déborde. La lecture de "buffer" prend la forme :

DE: +0	+1	+2	+3	+4	+5	+6	+7	+8	. . .	+n
mx	nc	c1	c2	c3	c4	c5	c6	c7	. . .	??

où "mx" est le nombre maximum de caractères que le "buffer" portera (1 à 255), "nc" le nombre de caractères lus (établi par FDOS après RETURN), suivi par les caractères lus à partir de la console.
Si nc < mx, des positions non initialisées suivent le dernier caractère, signifié par "??" dans la figure du dessus.

Un certain nombre de fonctions de contrôle sont reconnues pendant l'édition de lignes :

rub/del	enlève et renvoie en écho le dernier caractère
CTRL-C	réinitialise au début de la ligne
CTRL-E	crée la fin physique de la ligne
CTRL-H	va en arrière d'une position de caractère
CTRL-J	("line-feed") termine l'entrée de la ligne
CTRL-M	(RETURN) termine l'entrée de la ligne
CTRL-R	retape la ligne courante après la nouvelle ligne
CTRL-U	enlève la ligne courante après la nouvelle ligne
CTRL-X	va en arrière jusqu'au début de la ligne courante

Notez également que certaines fonctions qui renvoient le chariot à la position la plus à gauche (par exemple, le CTR-X) ne le renvoient qu'à la position où le signe d'attente s'est arrêté (dans les versions antérieures, le chariot revenait à la marge gauche extrême). Cette convention rend l'entrée de données et la correction de ligne plus facile.

FONCTION 11 : OBTENTION DE L'ETAT DE LA CONSOLE

Paramètres d'entrée :
Registre C : 0BH

Valeur retournée :
Registre A : état de la console

La fonction d'obtention de l'état de la console vérifie qu'un caractère a été tapé à la console. Si un caractère est prêt, la valeur 0FFH est retournée dans le registre A. Sinon, une valeur 00H est retournée.

FONCTION 12 : RENVOI DU NUMERO DE LA VERSION

Paramètres d'entrée :
Registre C : 0CH

Valeur retournée :
Registres HL : numéro de la version

La fonction 12 fournit des informations qui permettent une programmation indépendante de la version. Une valeur de deux octets est retournée avec H =00 désignant la version du CP/M (H = 01 pour MP/M) et L = 00 pour toutes les versions antérieures à 2.0. La version 2.0 du CP/M retourne un 20 hexadécimal dans le registre L ; les versions suivantes retournent une valeur allant de 21 à 2F hexadécimal.
En utilisant la fonction 12, vous pouvez par exemple, écrire des programmes d'application qui fournissent à la fois des fonctions d'accès séquentiel et direct, avec l'accès direct impossible lors d'une opération sous des versions antérieres du CP/M.

FONCTIONS 13 : REINITIALISATION DU SYSTEME DISQUE

Paramètres d'entrée :
Registre C : 0DH

La fonction de réinitialisation de disque est utilisée pour rétablir par un programme le système de fichiers à un état de réinitialisation où tous les disques sont établis à lecture/écriture (voir les fonctions 28 et 29), où seul le "drive" A de disque est sélectionné, et où l'adresse par défaut DMA est refixée à BOOT+0089H. Cette fonction peut être utilisée, par exemple, par un programme d'application qui exige un changement de disque sans réinitialisation du système.

FONCTION 14 : SELECTION DE DISQUE

Paramètres d'entrée :
Registre C : 0EH
Registre E : disque sélectionné

La fonction de sélection de disque désigne le "drive" disque qui sera considéré dans le registre E comme le disque par défaut pour les opérations sur fichiers suivantes, avec E = 0 pour le "drive" A, 1 pour le "drive" B, et ainsi de suite jusqu'à 15 qui correspond au "drive" P dans un système à seize "drives" complet. Le "drive" est placé dans un état "on-line" qui rend active en particulier sa "directory" jusqu'au nouveau "départ à froid", "départ à chaud", ou jusqu'à la prochaine opération de réinitialisation du système disque. Si le moyen de communication avec le disque est changé alors qu'il est "on-line", le "drive" prend automatiquement l'état lecture/seulement (R/O) dans une configuration CP/M standard (voir la fonction 28). Les FCB qui spécifient le code de "drive" zéro (dr = 00H) font référence automatiquement au "drive" par défaut courant. Les valeurs entre 1 et 16 des codes de "drive" ignorent cependant le "drive" par défaut et font référence directement aux "drives" A à P.

FONCTION 15 : OUVERTURE DE FICHIER

Paramètres d'entrée :
Registre C : 0FH
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

L'opération d'ouverture de fichier est utilisée pour rendre actif un fichier qui existe dans la "directory" du disque pour le numéro d'utilisateur actif courant. Le FDOS analyse la "directory" du disque référencé par DE (l'octet s1 est automatiquement annulé), où un point d'interrogation ASCII (3FH) s'identifie à tout caractère de la "directory" dans n'importe laquelle de ces positions. Normalement, aucun point d'interrogation n'est inclu, et les "ex" et "s2" du FCB sont nuls.

Si un élément de "directory" correspond à celui qui est recherché, les informations de la "directory" qui s'y rapportent sont copiées dans les octets d0 à dn du FCB, permettant ainsi l'accès aux fichiers par des opérations de lecture et d'écriture successives. Notez qu'un fichier ne doit pas être accedé avant qu'une ouverture réussie ne soit terminée. Après un RETURN, la fonction d'ouverture renvoie un "code de directory" avec une valeur allant de 1 à 3 si l'ouverture a réussi ou 0FFH (255 décimal) si le fichier n'a pas été trouvé. Si des points d'interrogation apparaissent dans le FCB, le premier FCB correspondant est alors rendu actif. Notez que l'enregistrement courant ("cr") doit être mis à zéro par le programme si l'accès au fichier va être séquentiel à partir du premier enregistrement.

FONCTION 16 : FERMETURE DE FICHIER

Paramètres d'entrée :
Registre C : 10H
Registre DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

La fonction de fermeture de fichier est la fonction opposée de la fonction d'ouverture de fichier. Si le FCB adressé par DE a été rendu actif auparavant par une fonction d'ouverture ou de création (voir les fonctions 15 et 22), la fonction de fermeture enregistre en permanence le nouveau FCB pour la fermeture est identique à celui de la fonction d'ouverture. Le code de "directory" retourné après une opération de fermeture réussie est 0, 1, 2, ou 3, alors qu'un 0FFh (255 décimal) est renvoyé si le nom de fichier ne peut être trouvé. Un fichier n'a pas besoin d'être fermé si seules des opérations de lecture y ont été faites. Si des opérations de lecture ont été réalisées, l'opération de fermeture est cependant nécessaire pour enregistrer en permanence les informations de la nouvelle "directory".

FONCTION 17 : RECHERCHE DU PREMIER

Paramètres d'entrée :
Recherche C : 11H
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

La fonction recherche du premier recherche dans la "directory" un fichier correspondant à celui qui est donné par le FCB adressé par DE. La valeur 255 (FF hexadécimal) est retournée si le fichier n'est pas trouvé ; sinon, le fichier est trouvé, l'adresse courante DMA est remplie avec l'enregistrement contenant l'entrée de "directory", et la position de départ relative est A * 32 (fait tourner le registre A de cinq octets vers la gauche, ou ajoute (ADD) A cinq fois). Bien qu'elles ne soient pas normalement exigées pour des programmes d'application, les informations de la "directory" peuvent être extraites du "buffer" à cette position.

Un point d'interrogation ASCII (63 décimal, 3F hexadécimal) dans toute position de "f1" à "ex" remplace le champ correspondant de toute entrée de "directory" sur le "drive" de disque par défaut ou sur le "drive" sélectionné automatiquement. Si le champ "dr" contient un point d'interrogation ASCII, alors la fonction de sélection automatique de disque devient ineffective, le disque par défaut est recherché, la fonction de recherche retournant toute entrée correspondante, allouée ou libre, appartenant à tout numéro d'utilisateur. Cette dernière fonction n'est pas normalement utilisée par des programmes d'application, mais permet d'analyser toutes les valeurs de la "directory" courante. Si le champ "dr" n'est pas un point d'interrogation, l'octet "s2" est automatiquement mis à zéro.

FONCTION 18 : RECHERCHE DU SUIVANT

Paramètres d'entrée :
Registre C : 12H

Valeur retournée :
Registre A : code de "directory"

La fonction de recherche du suivant est identique à la fonction de recherche du premier sauf que l'analyse de "directory" continue à partir de la dernière entrée recherchée. Semblable à la fonction 17, la fonction 18 retourne la valeur décimale 255 dans A quand la recherche devient vaine.

FONCTION 19 : DESTRUCTION DE FICHIER

Paramètres d'entrée :
Registre C : 13H
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

La fonction de destruction de fichier enlève des fichiers qui correspondent au FCB adressé par DE. Le nom de fichier et le type peuvent contenir des références ambigües (i.e., des points d'interrogation dans diverses positions), mais le code de sélection du "drive", lui, ne peut être ambigü, comme dans les fonctions de recherche.

La fonction 19 retourne un décimal 255 si le fichier ou les fichiers référencé(s) ne peuvent être trouvés ; sinon, une valeur comprise entre 1 et 3 est renvoyée.

FONCTION 20 : LECTURE SEQUENTIELLE

Paramètres d'entrée :
Registre C : 14H
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

Etant donné que le FCB adressé par DE a été rendu actif par une fonction d'ouverture ou de création (numéros 15 et 22), la fonction de lecture séquentielle lit le prochain enregistrement de 128 octets à partir du fichier dans la mémoire à l'adresse courante DMA. L'enregistrement est lu à partir de la position "cr" de l'extension, et le champ "cr" est automatiquement incrémenté jusqu'à la position de l'enregistrement suivant. Si le champ "cr" déborde, l'extension logique suivante est automatiquement ouverte et le champ "cr" est rétabli à zéro pour préparer la prochaine lecture. La valeur 00H est retournée dans le registre A si l'opération de lecture a réussi, alors qu'une valeur non nulle est renvoyée si aucune donnée n'existe à la position de l'enregistrement suivant (par exemple, quand la fin du fihier apparaît).

FONCTION 21 : ECRITURE SEQUENTIELLE

Paramètres d'entrée :
Registre C : 15H
Registres DE : Adresse du FCB

Valeur retournée :
Registre A : code de "directory"

Etant donné que le FCB adressé par DE a été rendu actif par une fonction d'ouverture ou de création (numéros 15 et 22), la fonction d'écriture séquentielle écrit l'enregistrement de données de 128 octets à l'adresse courante DMA sur le fichier nommé par le FCB. L'enregistrement est placé à la position "cr" du fichier, et le champ "cr" est automatiquement incrémenté jusqu'à la position de l'enregistrement suivant. Si le champ "cr" déborde, l'extension logique suivante est automatiquement ouverte et le champ "cr" est rétabli à zéro en vue de l'opération d'écriture suivante. Celles-ci peuvent se faire dans un fichier existant, auquel cas, les enregistrements dernièrement écrits recouvrent ceux existant déjà dans le fichier. Le registre A = 00H à la fin d'une opération d'écriture réussie, alors qu'une valeur non nulle indique une écriture non exécutée due au fait que le disque était plein.

FONCTION 22 : CREATION DE FICHIER

Paramètres d'entrée :
Registre C : 16H
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

L'opération de création de fichier est identique à l'opération d'ouverture de fichier sauf que le FCB doit nommer un fichier qui n'exite pas dans la "directory" du disque concerné courant (i.e., celui qui est dénommé explicitement par un code "dr" non nul, ou le disque par défaut si "dr" est zéro). Le FDOS crée le fichier et transforme à la fois la "catalogue" et la valeur de la mémoire principale, en un fichier vide. Le programmeur doit s'assurer qu'il n'a pas été produit deux noms de fichiers semblables ; une opération de destruction antérieure est suffisante s'il est possible de trouver deux mêmes noms. Lors d'un retour, le registre A = 0,1,2, ou 3 si l'opération a réussi et 0FFH (255 décimal) si plus aucun espace n'est disponible dans la "catalogue". La fonction de création a l'effet secondaire de rendre actif le FCB et de faire qu'ainsi une ouverture suivante n'est pas nécessaire.

FONCTION 23 : CHANGEMENT DU NOM D'UN FICHIER

Paramètres d'entrée :
Registre C : 17H
Registre DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

La fonction de changement de nom utilise le FCB adressé par DE pour changer le nom composé des 16 premiers octets en un nom composé par les 16 "deuxième" octets. Le code de "drive" "dr" à la position 0 est utilisé pour sélectionner le "drive", alors que le code de "drive" pour le nouveau nom de registre A est mis à une valeur comprise entre 0 et 3 si le changement de nom a été fait avec succès, et à 0FFH (255 décimal) si le premier nom de fichier n'a pas pu être trouvé dans l'analyse de la "directory".

FONCTION 24 : RENVOI DU VECTEUR "LOGIN"

Paramètres d'entrée : Registre C : 18H

Valeur retournée : Registres HL : vecteur "login"

La valeur "login" retournée par CP/M est une valeur 16 bits dans HL, où le bit le moins significatif de L correspond au premier "drive" A, et le bit de grand ordre de H correspond au seizième "drive", étiquetté P. Un bit "0" indique que le "drive" n'est pas "on-line" à cause d'une sélection de "drive" de disque explicite, ou d'une sélection de "drive" implicite causée par une opération sur fichier qui spécifiait un champ "dr" non nul. Notez que la compatibilité est conservée avec les versions précédentes, puisque les registres A et L contiennent, lors du retour, les mêmes valeurs.

FONCTION 25 : RENVOI DU DISQUE COURANT

Paramètres d'entrée : Registre C : 19H

Valeur retournée : Registre A : disque courant

La fonction 25 retourne le numéro du disque par défaut sélectionné actuellement dans le registre A. Les numéros de disque s'échelonnent de 0 à 15 correspondant aux "drives" A à P.

FONCTION 26 : MISE EN PLACE DE L'ADRESSE DMA

Paramètres d'entrée : Registre C : 1AH
Registre DE : adresse DMA

"DMA" correspond aux premières lettres de "Direct Memory Address"; adresse de mémoire directe, qui est souvent utilisée avec des controleurs de disque qui accèdent directement à la mémoire d'un gros ordinateur pour transférer des données de et vers le sous-système disque. Bien que plusieurs systèmes d'ordinateurs utilisent un accès non-DMA (i.e., les données sont transférées par des opérations d'E/S programmées), l'adresse DMA dans le CP/M est devenue l'adresse à laquelle réside l'enregistrement de données de 128 octets avant une écriture sur disque et après une lecture sur disque. Après un "départ à froid", "départ à chaud" ou réinitialisation du système disque, l'adresse DMA est automatiquement fixée à BOOT+0080H. La fonction de mise en place de la DMA peut être utilisée cependant pour changer cette valeur par défaut pour adresser une autre zone de mémoire où les enregistrements de données résident. Ainsi, l'adresse DMA devient la valeur spécifiée par DE jusqu'à ce qu'elle soit changée par une fonction de mise en place de DMA, un "départ à froid", "départ à chaud", ou une réinitialisation du sytème disque.

FONCTION 27 : OBTENTION DE L'ADRESSE (ALLOCATION)

Paramètres d'entrée :
Registre C : 18H

Valeur retournée :
Registres HL : adresse de l'allocation

Un "vecteur d'allocation" est conservé dans la mémoire principale pour chaque "drive" de disque "on-line". Divers programmes du système utilisent les informations fournies par le vecteur d'allocation pour déterminer la quantité de stockage restant (voir le programme STAT). La fonction 27 retourne l'adresse de base du vecteur d'allocation pour le "drive" de disque sélectionné courant. Les informations d'allocation peuvent être cependant invalides si le disque sélectionné a été marqué lecture/seulement (R/O). Bien que cette fonction ne soit pas normalement utilisée par des programmes d'application, des détails supplémentaires sur le vecteur d'allocation sont fournis dans le "Guide de modification du CP/M."

FONCTION 28 : PROTECTION DU DISQUE CONTRE L'ECRITURE

Paramètres d'entrée :
Registre C : 1CH

La fonction de protection du disque contre l'écriture protège temporairement le disque sélectionné actuellement contre l'écriture. Toute tentative d'écriture sur le disque avant le prochain "départ à chaud ou à froid" fera apparaître le message :

BDOS ERR ON d: R/O

FONCTION 29 : OBTENTION DU VECTEUR LECTURE/SEULEMENT (R/O)

Paramètres d'entrée :
Registre C : 1DH

Valeur retournée :
Registre HL : valeur du vecteur R/O

La fonction 29 retourne un vecteur de bit dans la paire de registres HL qui indique des "drives" qui ont le bit temporaire lecture/seulement établi. Semblable à la fonction 24, le bit le moins significatif correspond au "drive" A, alors que le bit le plus significatif correspond au "drive" P. Le bit R/O est fixé soit par un appel explicite à la fonction 28, soit par un progrmme automatique du CP/M qui détecte les disques changés.

FONCTION 30 : MISE EN PLACE DES ATTRIBUTS DE FICHIER

Paramètres d'entrée :
Registre C : 1EH
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de "directory"

La fonction de mise en place des attributs de fichier permet la manipulation par des programmes d'indicateurs permanents attachés aux fichiers. En particulier, les attributs système (t1' et t2') et R/O peuvent être établis ou changés. La paire DE s'adresse à un nom de fichier ambigü avec les attributs appropiés mis en place ou changés. La fonction 30 recherche le fichier désiré, et change l'entrée de "directory" correspondante pour qu'elle contienne les indicateurs sélectionnés. Les indicateurs f1' à f4' ne sont pas utilisés à présent, mais peuvent être utiles pour des programmes d'application, puisqu'ils ne sont pas compris dans le traitement d'égalisation pendant les opérations d'ouverture et de fermeture de fichier. Les indicateurs f5' à f8' et t3' sont réservés pour un développement futur du système.

FONCTION 31 : OBTENTION DE L'ADRESSE (PARAMETRES DISQUE)

Paramètres d'entrée :
Registre C : 1FH

Valeur retournée :
Registres HL : adresse du DPB

L'adresse du bloc des paramètres du disque (DPB) résident du BIOS est retournée dans HL après l'appel de cette fonction. Cette adresse peut être utilisée pour l'un ou l'autre des propos suivants. Premièrement, les valeurs des paramètres du disque peuvent être extraites pour affichage ou calcul d'espace ; deuxièmement des programmes transitoires peuvent changer dynamiquement les valeurs des paramètres du disque courant quand la configuration du disque change, si nécessaire. Normalement, les programmes d'application n'exigeront pas cette facilité.

FONCTION 32 : MISE EN PLACE/OBTENTION DU CODE UTILISATEUR

Paramètres d'entrée :
Registre C : 20H
Registre E : 0FFH (obtention) ou
code utilisateur (mise en place)

Valeur retournée :
Registre A : code courant ou
(aucune valeur)

Un programme d'application peut changer ou demander quel est le numéro d'utilisateur actif courant en appelant la fonction 32. Si le registre E = 0FFH, la valeur du numéro d'utilisateur courant est retournée dans le registre A, où elle sera comprise entre 0 et 31. Si le registre E n'est pas 0FFH, le numéro d'utilisateur courant est alors changé en la valeur de E (modulo 32).

FONCTION 33 : LECTURE DIRECTE

Paramètres d'entrée :
Registre C : 21H
Registre DE : Adresse du FCB

Valeur retournée :
Registre A : code de retour

La fonction de lecture directe est identique à l'opération de lecture de fichier séquentiel des versions précédentes, sauf que l'opération de lecture a lieu à un numéro d'enregistrement particulier, sélectionné par la valeur 24 bits construite à partir du champ de trois octets qui suit le FCB (positions d'octet r0 à 33, r1 à 34, et r2 à 35). Notez que la série de 24 bits est stockée avec d'abord l'octet le moins significatif (r0), puis l'octet moyen (r1), et en dernier l'octet haut (r2). Le CP/M ne fait pas référence à l'octet r2 qui doit être égal à zéro, puisqu'une valeur non nulle indique un débordement au delà de la fin du fichier.

Ainsi, la paire d'octets r0, r1 est considérée comme un octet double ou valeur "mot" qui contient l'enregistrement à lire. Cette valeur est comprise entre 0 et 65535, donnant accès à tout enregistrement particulier du fichier de 8 mégaoctets. Afin de pouvoir traiter un fichier en utilisant un accès direct, l'extension de base (extension 0) doit être ouverte en premier. Bien que l'extension de base puisse ou non contenir des données allouées, cela assure que le fichier est correctement enregistré dans la "directory", et est visible dans les demandes de DIR. Le numéro de l'enregistrement sélectionné est ensuite conservé dans le champ de l'enregistrement. A la fin de l'appel, le registre A contient soit un code d'erreur, l'un de ceux qui sont exposés ci-dessous, soit la valeur 00 indiquant que l'opération a été réussie. Dans le dernier cas, l'adresse courante DMA contient l'enregistrement auquel on accède directement.
Notez que contrairement à ce qui ce passe dans une opération de lecture séquentielle, le numéro de l'enregistrement n'est pas avancé. Ainsi, les lectures directes suivantes continuent à lire le même enregistrement.

Après chaque opération de lecture directe, l'extension logique et les valeurs de l'enregistrement courant sont établies automatiquement. Ainsi, le fichier peut être lu ou écrit séquentiellement, en commençant à partir de la position courante à laquelle on accède directement. Notez cependant, que dans ce cas, le dernier enregistrement lu directement sera re-lu si vous passez du mode direct à la lecture séquentielle, et le dernier enregistrement sera ré-écrit si vous passez à une opération d'écriture séquentielle. Vous pouvez, bien sûr, avancer simplement la position de l'enregistrement direct après chaque lecture ou écriture directe pour obtenir l'effet d'une opération d'E/S séquentielle.

Voici une liste des codes d'erreur retournés dans le registre A après une lecture directe.

01	lecture de données non écrites
02	(non retourné en mode direct)
03	ne peut pas fermer l'extenion courante
04	recherche d'une extension non écrite
05	(non retourné en mode lecture)
06	recherche après la fin physique du disque

Les codes d'erreur 01 et 04 apparaissent quand une opération de lecture directe accède à un bloc de données qui n'a pas été écrit auparavant, ou à une extension qui n'a pas été créée, qui sont des conditions équivalentes. L'erreur 3 ne survient pas normalement sous des opérations sur le système, propre, mais elle peut être éliminée en relisant simplement, ou en reouvrant l'extension zéro jusqu'à ce que le disque soit physiquement protégé contre l'écriture. Le code d'erreur 06 apparaît à chaque fois que l'octet r2 est non nul sous la version actuelle 2.0. Normalement, les codes de retour non nuls peuvent être considérés comme des données manquantes, et des codes nuls indiquent la fin d'une opération.

FONCTION 34 : ECRITURE DIRECTE

Paramètres d'entrée :
Registre C : 22H
Registres DE : adresse du FCB

Valeur retournée :
Registre A : code de retour

L'opération d'écriture directe est lancée de la même façon qu'un appel à une lecture directe, sauf que les données sont écrites sur le disque à partir de l'adresse courante DMA. De plus, si l'extension ou le bloc de données du disque qui doit être écrit n'a pas encore été alloué, l'allocation est exécutée avant que l'opération d'écriture ne continue. Comme dans l'opération de lecture directe, le numéro de l'enregistrement direct n'est pas changé à la suite de l'écriture. Le numéro de l'extension logique et les positions de l'enregistrement courant du FCB sont établies de telle façon qu'ils correspondent à l'enregistrement direct qui est en cours d'écriture. Les opérations de lecture ou d'écriture séquentielle peuvent commencer après une écriture directe, mais l'enregistrement auquel on vient d'accèder est soit relu soit réécrit, quand l'opération séquentielle commence. Vous pouvez aussi avancer simplement la position de l'enregistrement direct après chaque écriture pour obtenir l'effet d'une opération d'écriture séquentielle. Remarquez, en particulier que le fait de lire ou d'écrire le dernier enregistrement d'une extension en mode direct n'entraîne pas le passage automatique à une autre extension, comme cela se produit en mode séquentiel.

Les codes d'erreur retournés par une écriture directe sont identiques à ceux d'une opération de lecture directe avec l'addition du code d'erreur 05, qui indique qu'une nouvelle extension ne peut être créée à cause du débordement de la "directory".

FONCTION 35 : CALCUL DE LA TAILLE DU FICHIER

Paramètres d'entrée :
Registre C : 23H
Registres DE : adresse du FCB

Valeur retournée :
Mise en place du champ de l'enregistrement direct

Lors du calcul de la taille d'un fichier, la paire de registres DE s'adresse à un FCB dans le format du mode direct (les octets r0, r1, et r2 sont présents). Le FCB contient un nom de fichier non ambigü qui est utilisé dans l'analyse de la "directory". Lors du retour, les octets de l'enregistrement direct contiennent la taille "virtuelle" du fichier qui est, en réalité, l'adresse de l'enregistrement suivant la fin du fichier. Si, après un appel de la fonction 35, l'octet haut r2 de l'enregistrement est 01, le fichier contient alors le nombre maximum d'enregistrement 65536. Autrement, les octets r0 et r1 constituent une valeur 16 bits (r0 est l'octet le moins significatif, comme avant) qui est la taille du fichier.

Des données peuvent être ajoutées à la fin d'un fichier existant en appelant simplement la fonction 35 pour qu'elle établisse la position de l'enregistrement direct à la fin du fichier, puis en exécutant une séquence d'écritures directes commençant à l'adresse pré-établie de l'enregistrement.

La taille virtuelle d'un fichier correspond à la taille physique quand le fichier est écrit séquentiellement. Si, au lieu de cela, le fichier a été créé en mode direct et que des "trous" existent dans l'allocation, le fichier peut en fait contenir moins d'enregistrement que la taille n'en indique. Si, par exemple, seul le dernier enregistrement d'un fichier de huit mégaoctets est écrit en mode direct (i.e., le numéro d'enregistrement 65535), la taille virtuelle est alors de 65536 enregistrements, bien qu'un seul bloc de données ne soit réellement alloué.

FONCTION 36 : MISE EN PLACE D'UN ENREGISTREMENT DIRECT

Paramètres d'entrée :
Registre C : 24H
Registres DE : adresse du FCE

Valeur retournée :
Champ de l'enregistrement direct établi

La fonction de mise en place d'un enregistrement direct oblige le BDOS à produire automatiquement la position d'un enregistrement direct à partir d'un fichier qui a été lu ou écrit séquentiellement, jusqu'à un endroit particulier. La fonction peut être utile de deux manières.

D'abord, il est souvent nécessaire de lire et d'analyser initialement un fichier séquentiel pour extraire les positions des divers champs de "clé". A chaque fois qu'une clé est rencontrée, la fonction 36 est appelée pour calculer la position de l'enregistrement direct pour les données correspondant à cette clé. Si la taille de l'unité des données est 128 octets, la position résultante de l'enregistrement est placée dans une table avec la clé pour les prochains appels. Après avoir analysé le fichier entier et tabulé les clés et leurs numéros d'enregistrement, vous pouvez vous rendre tout de suite à un enregistrement avec clé particulière, en exécutant une lecture directe avec le numéro de l'enregistrement direct correspondant, qui a été sauvegardé auparavant. Ce schéma peut être facilement généralisé avec des enregistrements de longueurs variables, puisque le programme n'a besoin que de stocker la position de l'octet relatif au "buffer" ainsi que la clé et le numéro d'enregistrement afin de pouvoir trouver plus tard la position de départ exacte de la donnée clé.

La deuxième utilité de la fonction 36 apparaît lors du passage d'une lecture ou écriture séquentielle à une lecture ou écriture directe. Quand on accède séquentiellement à un fichier, jusqu'à un endroit particulier, la fonction 36 est appelée pour fixer le numéro d'enregistrement ; les opérations de lecture et d'écriture directe suivantes commencent à partir de l'endroit sélectionné dans le fichier.

3. UN EXEMPLE DE PROGRAMME DE COPIE DE FICHIER SUR LUI-MEME

Le programme présenté ci-dessous est un exemple relativement simple d'opérations sur des fichiers. Le fichier source du programme est créé sous le nom COPY.ASM en utilisant le programme ED du CP/M, puis assemblé en utilisant ASM ou MAC, ce qui donne un fichier "HEX". Le programme LOAD est celui qui est utilisé pour produire un fichier COPY.COM qui s'exécute directement sous le CP/M. Le programme commence par fixer le pointeur de pile à une zone locale, puis ontinue en enlevant le second nom de la zone par défaut à 006CH et en le mettant dans un bloc de contrôle de fichier de 33 octets appelé DFCB. Celui-ci est alors préparé pour des opérations sur fichier en vidant le champ de l'enregistrement courant. A cet endroit, la source et la destination du FCB sont prêtes pour le traitement puisque le SFCB à 005CH est établi par le CP/M après entrée dans le programme COPY. Ainsi, le premier nom est placé dans le FCB par défaut, avec les propres champs annulés, incluant le champ de l'enregistrement courant à 007CH. Le programme continue en ouvrant le fichier source, en détruisant tout fichier de destination existant, puis en créant le fichier de destination.
Si toutes ces opérations réussissent, le programme retourne à l'étiquette COPY jusqu'à ce que chaque enregistrement soit lu à partir du fichier source et placé dans le fichier de destination. Après achèvement du transfert de données, le fichier de destination est refermé et le programme retourne au niveau de commande du CCP en sautant jusqu'à BOOT.

```
        ;  programme exemple de copie d'un fichier sur lui-même
        ;
        ;  au niveau du CCP, la commande
        ;
        ;       copy a:x.y b:u.v
        ;
        ;  copie le fichier nommé x.y du "drive" a dans un fichier
        ;   nommé u.v sur le drive b.
        ;
0000 =  boot    equ     0000h   ; réinitialisation du système
0005 =  bdos    equ     0005h   ; entrée du bdos
005c =  fcb1    equ     005ch   ; premier nom de fichier
005c =  sfcb    equ     fcb1    ; fcb source
006c =  fcb2    equ     005ch   ; second nom de fichier
0080 =  dbuff   equ     0080h   ; "buffer" par défaut
0100 =  tpa     equ     0100h   ; début de la tpa
        ;
0009 =  printf  equ     9       ; fonction d'impression de "buffer"
000f =  openf   equ     15      ; fonction d'ouverture de fichier
0010 =  closef  equ     16      ; fonction de fermeture de fichier
0013 =  deletef equ     19      ; fonction de destruction de fichier
0014 =  readf   equ     20      ; lecture séquentielle
0015 =  writef  equ     21      ; écriture séquentielle
        makef   equ     22      ; fonction de création de fichier
        ;
```

```
0100                    org     tpa     ; début de la tpa
0100 311b02             lxi     sp,stack; pile locale
                        ;
                        ;       place le second nom de fichier dans dfcb
0103 0e10               mvi     c,16    ; un demi fcb
0105 116c00             lxi     d,fcb2  ; source de déplacement
0108 21da01             lxi     h,dfcb  ; fcb de destination
010b 1a mfcb:           ldax    d       ; fcb source
010c 13                 inx     d       ; prêt pour le suivant
010d 77                 mov     m,a     ; fcb de destination
010e 23                 inx     h       ; prêt pour le suivant
010f 0d                 dcr     c       ; numéro 16 ... 0
0110 c20b01             jnz     mfcb    ; boucle 16 fois
                        ;
                        ;       le nom a été déplacé, zéro retour chariot
0113 af                 xra     a       ; a = 00h
0114 32fa01             sta     dfcbcr  ; enregistrement courant = 0
                        ;
                        ;       les fcb source et de destination sont prêts
                        ;
0117 115c00             lxi     d,sfcb  ; fichier source
011a cd6901             call    open    ; erreur si 255
011d 118701             lxi     d,nofile; message prêt
0120 3c                 inr     a       ; 255 devient 0
0121 cc6101             cz      finis   ; effectif si aucun fichier
                        ;
```

```
                        ;       fichier source ouvert, préparation de la destination
0124 11da01             lxi     d,dfcb  ; destination
0127 cd7301             call    delete  ; enlève si présent
                        ;
012a 11da01             lxi     d,dfcb  ; destination
012d cd8201             call    make    ; crée le fichier
0130 119601             lxi     d,nodir ; message prêt
0133 3c                 inr     a       ; 255 devient 0
0134 cc6101             cz      finis   ; effectif si pas de place dans
                        ;       "directory", fichier source ouvert, fichier de
                        ;       destination ouvert
                        ;       copie jusqu'à fin de fichier sur source
                        ;
                        ;
0137 115c00 copy:       lxi     d,sfcb  ; source
013a cd7780             call    read    ; lecture du prochain enregistrement
013d b7                 ora     a       ; fin de ficier?
013e c25101             jnz     eofile  ; si oui saute l'écriture
                        ;
                        ;       pas la fin du fichier, écriture de l'nregistrement
0141 11da01             lxi     d,dfcb  ; destination
0144 cd7d01             call    write   ; écriture de l'enregistrement
0147 11a901             lxi     d,space ; message prêt
014a b7                 ora     a       ; 00 si écriture O.K.
014b c46101             cnz     finis   ; si ainsi, fin
014e c33701             jnp     copy    ; boucle jusqu'à eof
                        ;
```

```
        eofile: ; fin de fichier, fermeture de la destination
0151 11da01     lxi     d,dfcb    ; destination
0154 cd6e01     call    close     ; 255 si erreur
0157 21bb01     lxi     h,wrprot  ; message prêt
015a 3c         inr     a         ; 255 devient 00
015b cc6101     cz      finis     ; ne devrait pas arriver
        ;
        ;       opération de copie terminée, fin
015e 11cc01     lxi     d,normal  ; message prêt
        ;
        finis:  ; message d'écriture donné par de, réinitialisation
0161 0e09       mvi     c,printf
0163 cd0500     call    bdos      ; message d'écriture
0166 c30000     jmp     boot      ; réinitialisation du système
        ;
        ;       sous-routines de l'interface du système
        ;       (toutes retournent directement à partir de bdos)
        ;
0169 0e0f open: mvi     c,openf
016b c30500     jmp     bdos
        ;
016e 0e10 close: mvi    c,closef
0170 c30500     jmp     bdos
        ;
0173 0e13 delete:mvi    c,deletef
0175 c30500     jmp     bdos



        ;
0178 0e14 read: miv     c,readf
017a c30500     jmp     bdos
        ;
017d 0e15 write:miv     c,writef
017f c30500     jmp     bdos
        ;
0182 0e16 make: mvi     c,makef
0184 c30500     jmp     bdos
        ;
        ;       messages consoles
0187 6e6f20f nofile: db  'no source file$'
0196 6e6f209 nodir:  db  'no directory spaces$'
01a9 6f7574f space:  db  'out of data spaces$'
01bb 7772695 wrprot: db  'write protected?$'
01cc 636f700 normal: db  'copy complete$'
        ;
        ;       zones de données
01da    dfcb:   ds      33        ; fcb de destination
01fa =  dfcbcr  equ     dfcb+32   ; enregistrement courant
        ;
01fb            ds      32        ; pile de 16 niveaux
        stack:
021b            end
```

Notez que de nombreuses simplifications ont été faitres dans ce programme particulier. D'abord, il n'y a aucune recherche de noms de fichier invalides, qui pourrait contenir, par exemple, des références ambigües. Cette situation pourrait être détectée en analysant la zone par défaut de 32 octets commençant à l'adresse 005CH et en y recherchant les points d'interrogation ASCII. Une vérification devrait être faite pour s'assurer que les noms de fichier ont bien été inclus (recherche dans les adresses 005DH et 006DH des caractères non-blancs ASCII). Enfin, il serait nécessaire de s'assurer que les noms des fichiers source et de destination sont différents. Une augmentation de la rapidité d'exécution pourrait être obtenue en "bufferisant" plus de données dans chaque opération de lecture. On pourrait déterminer, par exemple, la taille de la mémoire en allant chercher FBASE à partir de l'adresse 0006H et utiliser toute la partie restante pour un "buffer" de données. Dans ce cas, le programmeur refixe simplement l'adresse DMA à la zone de 128 octets suivante avant chaque lecture. Après écriture dans le fichier de destination, l'adresse DMA est réétablie au début du "buffer" et incrémentée de 128 octets jusqu'à la fin, en même temps que chaque enregistrement est transféré dans le fichier de destination.

4. UN EXEMPLE D'UTILITAIRE DUMP D'UN FICHIER

Le programme de DUMP d'un fichier présenté ci-dessous est légérement plus complexe que le programme de copie de la section précédente. Le programme de DUMP lit un fichier d'entrée, spécifié dans la ligne de commande du CCP, et affiche le contenu de chaque enregistrement dans le format hexadécimal sur la console. Notez que le programme de DUMP sauvegarde la pile du CCP après entrée, rétablit la pile à une zone locale, et remet à zéro la pile du CCP avant de retouner directement au CCP. Ainsi, le programme de DUMP n'exécutre pas de "départ à chaud" à la fin de son traitement.

```
        ; programme de DUMP lit le fichier d'entrée et affiche les
        ;  données hexa
        ;
0100            org     100h
0005 =  bdos    equ     0005h   ; entrée du dos
0001 =  cons    equ     1       ; lecture sur console
0002 =  typef   equ     2       ; fonction type
0009 =  printf  equ     9       ; entrée d'impression du "buffer"
000b =  brkf    equ     11      ; fonction"break key"(vrai si car prêt)
000f =  openf   equ     15      ; ouverture du fichier
0014 =  readf   equ     20      ; fonction de lecture
        ;
005c =  fcb     equ     5ch     ; adresse du bloc de contrôle
0080 =  buff    equ     80h     ; entre l'adresse du "bufer" de disque
        ;
        ;       caractères non graphiques
000d =  cr      equ     0dh     ; retour chariot
000a =  lf      equ     0ah     ; "line-feed"
        ;
        ;       définitions du bloc de contrôle
005c =  fcbdn   equ     fcb+0   ; nom du disque
005d =  fcbfn   equ     fcb+1   ; nom du fichier
0065 =  fcbft   equ     fcb+9   ; type - fichier (3 caractères)
0068 =  fcbrl   equ     fcb+12  ; numéro d'extension
006b =  fcbrc   equ     fcb+15  ; compteur d'enregistrements (0 à 128)
007c =  fcbcr   equ     fcb+32  ; numéro d'enregistr.courant(prochain)
007d =  fcbln   equ     fcb+33  ; longueur de fcb
        ;
        ;       mise en place de la pile
0100 210000     lxi     h,0
0103 39         dad     sp
        ;       entrée du pointeur de pile dans hl à partir du CCP
0104 221502     shld    oldsp
        ;       mise en place du pp dans la zone de pile locale (remis
                à zéro à la fin)
0107 315702     lxi     sp,stktop
        ;       lit et imprime sucessivement les "buffers"
010a cdc101     call    setup   ; mise en place du fichier d'entrée
010d feff       cpi     255     ; 255 si fichier non présent
010f c21b01     jnz     openok  ; saute si ouverture est O.K.
        ;
        ;       fichier non ici, donne message d'erreur et retourne

0112 11f301     lxi     d,opnmsg
0115 cd9c01     call    err
0118 c35101     jmp     finis   ; à RETURN
        ;
        openok: ;ouverture réussie, mise en place de l'index de
                 "buffer" à la fin
011b 3e80       mvi     a,80h
011d
```

SOMMAIRE DES FONCTIONS DU SYSTEME

FONC	NOM DE LA FONCTION	PARAMETRES D'ENTREE	VALEURS DE SORTIE
0	réinitialisation du système	aucun	aucun
1	entrée sur console	aucun	A = car
2	sortie sur console	E = car	aucun
3	entrée sur lecteur	aucun	A = car
4	sortie sur perforateur	E = car	aucun
5	sortie de listing	E = car	aucun
6	E/S console directe	voir def	voir def
7	obtention de l'octet d'E/S	aucun	A = IOBYTE
8	mise ne place de l'octet d'E/S	E = IOBYTE	aucun
9	impression de chaîne	DE = .Buffer	aucun
10	lecture d'un buffer sur la console	DE = . Buffer	voir def
11	obtention de l'état de la console	aucun	A = 00/FF
12	renvoi du numéro de la version	aucun	HL= Version*
13	réinitialisation du système disque	aucun	voir def
14	sélection de disque	E = No de disque	voir def
15	ouverture de fichier	DE =.FCB	A= code de "dir"
16	fermeture de fichier	DE =.FCB	A= code de "dir"
17	recherche du premier	DE =.FCB	A= code de "dir"
18	recherche du suivant	aucun	A= code de "dir"
19	destruction de fichier	DE =.FCB	A= code de "dir"
20	lecture séquentielle	DE =.FCB	A= code de "dir"
21	écriture séquentielle	DE =.FCB	A= code d'erreur
22	création de fichier	DE =.FCB	A= code d'erreur
23	changement du nom d'un fichier	DE =.FCB	A= code de "dir"
24	renvoi du vecteur "login"	aucun	HL=vecteur "login"*
25	renvoi du disque courant	aucun	A= disque courant
26	mise en place de l'adresse DMA	DE =.DMA	aucun
27	obtention de l'adresse (allocation)	aucun	HL= .Allocation
28	protection du disque contre l'écriture	aucun	voir def
29	obtention du vecteur lecture/Seulement (R/O)	aucun	HL= Vect* R/O
30	mise en place des attributs de fichier	DE =.FCB	voir déf
31	obtention de l'adresse (paramètres disque)	aucun	HL= .DPB
32	mise en place/Obtention du code utilisateur	voir déf	voir déf
33	lecture directe	DE =.FCB	A= code d'erreur
34	écriture directe	DE =.FCB	A= code d'erreur
35	calcul de la taille du fichier	DE =.FCB	r0, r1, r2
36	mise en place d'un enregistrement direct	DE =.FCB	r0, r1, r2

* Notez que A= L, et B= H lors du retour