Réseaux
et systèmes
d'information

Bull

MANUEL DE RÉFÉRENCE

S O L A R

B O S 1 6

Système d'exploitation de base sur disque

1 164 325 00 036

Réseaux
et systèmes
d'information

Bull

Manuel de référence

SOLAR

SYSTEME D'EXPLOITATION DE BASE SUR DISQUE

BOS16
ADDENDUM A

Logiciel

SUJET : Système d'exploitation de base sur Disque

OBSERVATION : Première mise à jour du document 1 164 325 00 036 04 de Février 1986. Suivre les directives de mise à jour et placer la présente feuille directement après la couverture de votre manuel pour indiquer que l'addendum A est inclus.

VERSION LOGICIEL :

DATE : DECEMBRE 1986

REF Bull-S-A. : 1 164 325 00 036 05/FR

Réseaux
et systèmes
d'information

Bull

Bull
Sems

Dépôt Légal
3ème trimestre 1986

Imprimé en France

| Vos suggestions sur la forme et le fond de ce manuel seront les bienvenues. Une feuille destinée à recevoir vos remarques se trouve à la fin du présent manuel.

Réseaux
et systèmes
d'information

Bull

A D D E N D U M

TITRE DU DOCUMENT : BOS16 ADDENDUM A

Date : DECEMBRE 1986

INSTRUCTIONS DE MISE A JOUR

- Enlever dans votre manuel le sommaire, les chapitres 6 et 11.
- Remplacer par les nouveaux sommaire et chapitres correspondants ci-joints.

En marge indication de la partie modifiée

Réf Bull S.A. : 1 164 325 00 036 05/FR

Réseaux
et systèmes
d'information

Bull

Manuel de référence

Bull

SOLAR

SYSTEME D'EXPLOITATION DE BASE SUR DISQUE

BOS16

Logiciel

SUJET : Système d'exploitation de base sur Disque

OBSERVATION

VERSION LOGICIEL :

DATE : DECEMBRE 1986

REF Bull-s-A. : 1 164 325 00 036 05/FR

Réseaux
et systèmes
d'information

Bull

Bull

Dépôt Légal
3ème trimestre 1986

Imprimé en France

Vos suggestions sur la forme et le fond de ce manuel seront les bienvenues. Une feuille destinée à recevoir vos remarques se trouve à la fin du présent manuel.

1 PRESENTATION

BOS16 (Basic Operating System/Disk) est un système d'exploitation en multiprogrammation d'une configuration SOLAR 16-35, 16-70 ou 16-90 mono comportant un disque.

Il fournit une activité Background en simultanéité avec le déroulement d'une application temps réel, ce qui permet de rentabiliser au maximum les ressources de l'installation.

Le Background assure la production, la mise au point et l'exploitation de programmes.

Il permet en outre le contrôle et l'évolution des tâches de l'application.

BOS16 se présente sous la forme d'un noyau de base auquel l'utilisateur peut adjoindre un certain nombre de modules regroupant des fonctions qui sont spécifiques du temps réel.

Cette conception modulaire permet l'utilisation de BOS16 en tant que :

- système de production de programmes
- moniteur temps réel.

BOS16 est un système d'exploitation basé sur l'utilisation d'un disque qui permet le stockage :

- des parties non résidentes du système
- des tâches de l'application
- des fichiers de l'utilisateur.

BOS16 est exploitable sur la configuration minimale suivante :

- 64 K mots de mémoire centrale
- téléimprimeur ou console de visualisation appelé périphérique de dialogue
- scheduler microprogrammé MTS16
- disque de moyenne ou grande capacité
  ou deux disques souples double face et double densité
- option protection mémoire DRPS16.

Il supporte l'option horloge temps réel et acquiert une plus grande efficacité avec l'utilisation de périphériques du type :

- lecteur de cartes
- imprimante ligne
  ou imprimante rapide
- dérouleur de bandes magnétiques

ainsi que par l'augmentation de la capacité disque.

2 DESCRIPTION GENERALE DE BOS16

BOS16 est un système pouvant réaliser, simultanément ou non :

- la fonction de production de programmes
- la gestion d'une application temps réel.

Il comporte un noyau de base, orienté vers la production de programmes, ainsi qu'un certain nombre de modules à vocation temps réel.

Le noyau de base assure notamment :

- l'initialisation d'un contexte d'exécution des programmes
- le traitement des interruptions périphériques ainsi que le contrôle et la réalisation des différentes opérations d'entrée-sortie
- le traitement des interruptions système
- la mise en oeuvre des processeurs système ou utilisateur ainsi que celle des tâches de l'application
- la modification, avant exécution d'un programme, des liens entre ce programme et les périphériques sur lesquels il demandera la réalisation d'opérations d'entrée-sortie.

Il permet en particulier :

- d'utiliser simplement tous les processeurs du software de base dont :

  . les compilateurs FORTRAN, PL16
  . le macroprocesseur MACP
  . le macro-assembleur ASM
  . l'éditeur de liens EDILE, le chargeur disque BUILD et LKLOAD
  . l'éditeur de texte EDIT16
  . l'utilitaire de fichiers FUP

- d'exploiter l'installation en mode train de travaux ou conversationnel
- d'utiliser le système d'entrée-sortie IOCS16
- d'utiliser le système de fichiers FMS16
- d'exploiter des programmes dont la taille est supérieure à la taille mémoire disponible (segmentation)
- d'utiliser les bibliothèques système ou utilisateur.

BOS16 s'adapte à la configuration du système et aux besoins de l'utilisateur :

- adaptation du moniteur d'entrée-sortie IOCS16 à toute périphérie particulière rendue possible par l'utilisation de GIO16
- intégration de la version du système de gestion de fichiers FMS16 retenue par l'utilisateur
- gestion par BOS16 d'une zone de mémoire dans laquelle peuvent être alimentés des programmes permanents.

L'adjonction au noyau de base de BOS16 de différents modules apporte à l'utilisateur un ensemble d'outils destinés aux tâches de l'application ou procure des services ou performances supplémentaires à la fonction production de programmes :

- gestion d'une zone allouée aux taches non résidentes
- gestion d'une horloge temps réel
- définition d'un contexte de redémarrage automatique après une disparition du secteur
- gestion d'"événements"
- options de performances.

Ce chapitre a pour but la présentation de la “philosophie” du système BOS16 ; il met l’accent sur certains aspects pratiques de son utilisation.

Il débute par la définition d’un ensemble de termes qui seront repris dans la suite du présent manuel.

2.1 DEFINITIONS

2.1.1 SYSTEME ET APPLICATION

On désigne sous le nom de système le produit software qui, dimensionné aux besoins de l'usager, est prêt à accepter les programmes qu'il doit gérer en temps réel.

L'application est composée d'une part du système, d'autre part de l'ensemble des programmes que l'usager fait gérer à BOS16 pour contrôler son processus.

2.1.2 MULTIPROGRAMMATION ET TACHES

la multiprogrammation est la technique par laquelle un système exécutif gère l'utilisation des ressources d'une installation (unité centrale, périphériques, ...) par plusieurs tâches concurrentes.

L'unité de base de programme dans un tel contexte est en effet la "tâche".

Sous BOS16 elle consiste en un programme ou un ensemble de programmes (link-édités entre eux) écrits en langage FORTRAN, PL16 ou ASM ; pour composer une tâche l'utilisateur utilise les différents langages de la façon la mieux adaptée aux traitements partiels à réaliser.

L'efficacité de la multiprogrammation réside dans la récupération des divers temps d'attente des tâches de l'application (attente de la fin d'une opération d'entrée-sortie, attente d'un événement, ...) au profit de celles qui sont disponibles pour utiliser les ressources de la machine.

2.1.3 PRIORITE ET SCHEDULING

La résolution des conflits d'accès aux ressources du système impose une notion de priorité attachée à chaque tache de l'application.

C'est ainsi que, sous BOS16, tout accès à un dispositif périphérique est géré à travers le critère de priorité relative des différents demandeurs par le "scheduler".

BOS16 utilise la structure multitâche et le scheduler microprogrammé du SOLAR 16 qui font partie intégrante de la machine, ce qui explique les bonnes performances et le faible temps de réponse du système.

2.1.4 FOREGROUND ET BACKGROUND

L'activité des programmes temps réel, ou Foreground, peut consister à contrôler un processus industriel. à effectuer des acquisitions de données ou des expériences de laboratoire; elles sont liées à l'environnement temps réel de l'installation.

Un ensemble de programmes moins critiques et d'intérêt plus général, le Background, utilise alors les ressources laissées disponibles dans un but de production et de mise au point de nouvelles tâches ou d'exploitation de programmes (calculs scientifiques ou traitement de données off-line par exemple).

Le Background est une tache dont le niveau de priorité est l'un des plus faibles de l'application.

2.2 CONVENTIONS

- Les séquences de programmes données en exemple suivent les règles d'écriture de l'assembleur ASM.

- Les nombres précédés du caractère apostrophe ' sont des nombres exprimés en hexadécimal.

- Le caractère e est pris pour "adresse de".

- Un certain nombre de symboles valeurs apparaissant dans les exemples sont supposés prédéfinis.

- Les mnémoniques utilisés correspondent en général aux traductions anglaises des expressions à résumer (la traduction en langue anglaise se trouve entre parenthèses après l'expression en francais).

- Un module de programme est un ensemble exécutable de données et d'instructions, un processeur est constitué de tout ou partie d'un module de programme dont l'exécution complète représente une unité de travail. Il peut donc y avoir plusieurs processeurs dans un programme.

- Les programmes utilisateurs du système lui demandent des services par l'intermédiaire de "requêtes programmées" qui sont des séquences d'instructions terminees par l'instruction SVC (SuperVisory Call).

- L'opérateur peut demander des services au système depuis un périphérique : ce sont les "conmandes" de contrôle.

- Les caractères "retour chariot" et "saut de ligne" lorsqu'ils doivent apparaître dans ce qui est écrit sont symbolisés par :

  "cr" pour Carriage Return

  "lf" pour Line Feed

2.3 CARACTERISTIQUES DE BOS16

2.3.1 FONCTIONS GENERALES DE BOS16

Lors de l'exécution d'un programme la gestion des interruptions. la réalisation des opérations d'entrée-sortie, l'initialisation de l'état calculateur sont des fonctions essentielles au bon déroulement de celui-ci bien qu'elles n'aient parfois qu'un rapport très lointain avec ce programme.

La programmation et la mise au point de ces fonctions sont toujours délicates et entraîneraient un gaspillage considérable de temps et d'efforts si elles devaient être refaites pour chaque programme.

Le système BOS16 destine à l'exploitation des programmes regroupe ces différentes fonctions et facilite ainsi la mise au point et l'exécution d'un programme.

L'utilisateur communique avec le moniteur de deux façons :

- en demandant des services au système par l'intermédiaire d'instructions écrites dans les programmes, ce sont les "requêtes".

- en demandant des services au système par l'intermédiaire d'un périphérique (le périphérique de dialogue), ce sont les "commandes".

Les “requêtes” (incluses dans un programme) sont constituées par une séquence d’instructions et éventuellement une table de paramètres qui spécifient l’opération demandée par le programme.

Certaines de ces requêtes sont relatives aux opérations d’entrée-sortie. Elles spécifient où se trouvent les données et où elles doivent aller. BOS16 réalise l’opération demandée, après avoir vérifié la validité d’un certain nombre de paramètres, et il donne à l’utilisateur un compte-rendu de son transfert.

D’autres requêtes au moniteur permettent de lui transmettre des paramètres relatifs au déroulement du programme, de tester un état du système, de demander des fonctions de positionnement de périphériques ou la synchronisation d’un programme sur la fin d’une opération d’entrée-sortie.

Les “commandes” transmises par un opérateur permettent de réinitialiser le système, de demander le chargement d’un programme. son exécution ou sa réexécution. De plus certaines commandes permettent de préciser les périphériques qui seront lies aux opérations d’entrée-sortie demandées par les programmes, et rendent ainsi ces programmes largement “indépendants” des périphériques utilisés.

Les demandes d’entrée-sortie peuvent porter sur des “unités fonctionnelles” (FU : Fonctional Unit] ou sur des ‘unités symboliques” (SU : Symbolic Unit).

Une unité fonctionnelle désigne un périphérique ou un sous-ensemble de périphérique :

Exemples :
- lecteur de cartes
- clavier du téléimprimeur
- lecteur du téléimprimeur
- partie de disque.

Les unités fonctionnelles sont liées à un périphérique (ou unité physique) lors de la configuration initiale ou des reconfigurations du système.

Les unités symboliques permettent de distinguer les différents flots d’information caractéristiques d'un programme :

Exemples :
- entrée symbolique
- sortie binaire
- liste des erreurs.

Lorsque le système a le contrôle, on peut affecter à tout moment une unité fonctionnelle à une unité symbolique. On peut affecter à plusieurs unités symboliques la même unité fonctionnelle.

Ces unités symboliques permettent de ne préciser les périphériques utilisés par un programme qu’au moment où l’on va demander son exécution. Elles assurent à ces programmes une indépendance totale vis à vis des périphériques.

Sous BOS16 on peut de plus affecter des fichiers aux unités symboliques. Ces fichiers peuvent n’être que temporaires (tués en fin de travail). Ils permettent d’accélérer l’enchaînement des travaux.

Les fichiers sont gérés par le système de fichiers FMS16. Celui-ci permet de gérer l’espace disque au mieux, tout en offrant des accès souples, optimisés et diversifiés :

- accès séquentiel
- accès indexé
- accès direct.

Toutes les opérations d’entrée-sortie sont gérées par interruptions, ce qui permet à plusieurs opérations d’entrée-sortie de se dérouler simultanément, une opération d’entrée-sortie pouvant elle toujours se dérouler simultanément au programme qui l’a demandée.

Enfin BOS16 est “modulaire” c’est-à-dire que l’utilisateur peut aisément

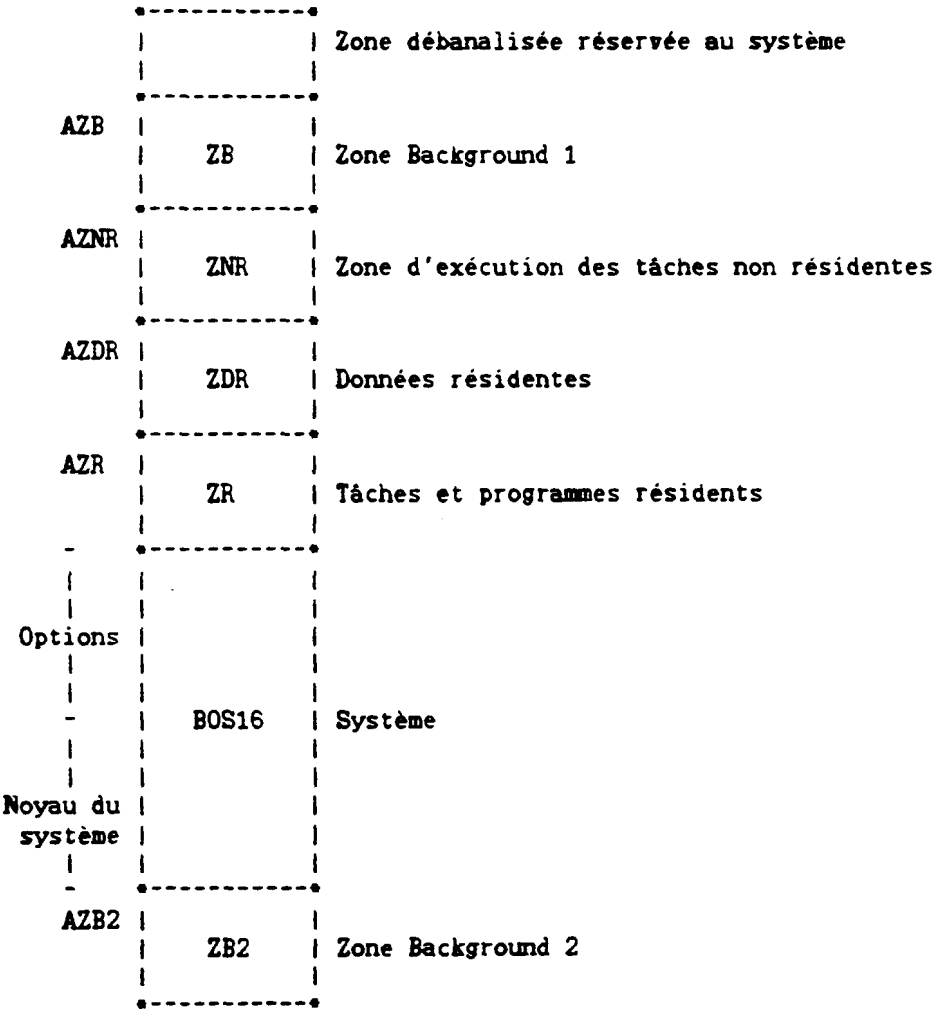
ajouter au superviseur :

- de nouveaux processeurs
- de nouvelles commandes
- de nouvelles tâches
- de nouvelles bibliothèques
- de nouveaux modules optionnels.

Ces adjonctions se font généralement par simple création de fichiers ou d'articles dans la zone disque du système (catalogue système). L'utilisateur peut ainsi adapter son moniteur aux services spécifiques qu'il lui demande.

2.3.2 ORGANISATION DE LA MEMOIRE

L'implantation mémoire du système correspond à la carte suivante :



Ce schéma est général. Il s'applique à BOS16 utilise en tant que système "Foreground-Background".

Six zones sont à considérer :

- zones occupées par le Background (ZB et ZB2)
- zone réservée aux tâches non résidentes de l'application (ZNR)
- zone de données résidentes (ZDR)
- zone résidente (ZR), réservée aux tâches résidentes de l'application et aux programmes permanents
- zone occupée par le système.

La dernière zone comporte le noyau de base de BOS16 et, éventuellement, les différentes options.

La production de programmes nécessite les zones Background.

La deuxième zone Background, appelée ZB2 est implantée après le système.

Cette zone sert à alimenter les programmes utilisateur ou processeurs système fonctionnant en mode esclave.

Elle a une taille de 32 K mots.

Toutes les zones sont configurables. Les zones ZNR, ZDR et ZR peuvent être de longueur nulle.

2.3.3 STRUCTURE DE BOS16

L'ensemble de l'application est divisé en tâches parmi lesquelles on peut distinguer :

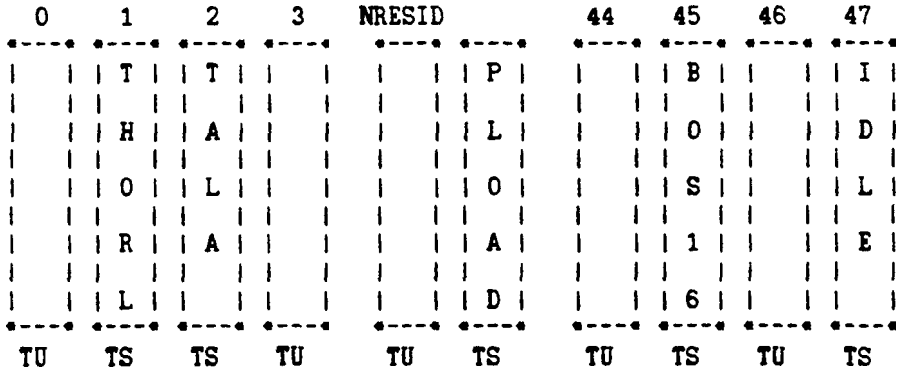
- les tâches utilisateur

  . Soit hardware pour les taches qui, devant répondre rapidement, se déroulent sous niveau hardware.
  C'est alors l'apparition sur le niveau d'une interruption qui déclenche l'exécution de la tâche.
  . Soit software pour les taches dont le traitement se déroule une fois tous les niveaux hardware satisfaits.
  Une priorité software affectée à chaque tâche permet de résoudre les conflits d'activation.

- les tâches système

  . Soit hardware pour traiter les alarmes (niveau O), les opérations d'entrée-sortie et l'horloge temps réel,
  . Soit software, telles les tâches de gestion de l'horloge, des alarmes, du chargement des tâches non résidentes.

48 tâches software sont ainsi gérées, système (TS) ou utilisateur (TU) :



La tâche THORL est activée après une interruption horloge; elle réalise le réveil des tâches après temporisation ainsi que le contrôle de la durée des jobs se déroulant en Background.

La tâche TALA est activée par la tâche hardware 0 ; elle réalise la suspension des tâches en alarme et la préparation des messages d'erreur imprimes par le système.

La tâche IDLE est la tâche butée du scheduler.

La tache BOS16 a deux rôles :

- production, mise au point et exploitation des programmes utilisateur, opérations réalisées en Background
- dialogue système-opérateur permet tant d'exercer un contrôle sur l'application par l'intermédiaire de commandes d'une part, des messages et diagnostics d'erreur d'autre part.

Cette fonction de dialogue, vitale pour l'application, est assortie d'un maximum de sécurités (mot de passe, message d'erreur, ...) ; elle permet en particulier à l'utilisateur :

- l'introduction on-line de nouvelles tâches
- la modification de l'heure
- l'initialisation du système
- la visualisation des états du système
- la sauvegarde sur disque.

Les niveaux de priorité 0 (complément de scheduling), 3 à 44 ainsi que 46 peuvent être attribués aux tâches utilisateur.

Lorsque l'option "Gestion des tâches non résidentes" a été intégrée au système, au moment de la génération, le niveau NRESID (configurable) est assigné aux tâches non résidentes de l'application.

Le système utilise alors le niveau NRESID+1 pour la gestion de ces tâches, opération réalisée par la tâche PLOAD.

2.3.4 TACHES

BOS16 dispose de commandes permettant le chargement en mémoire depuis le disque et la connexion à un niveau de priorité. des tâches résidentes de l'application, ces opérations étant effectuées généralement dans la phase de génération de l'application.

Durant cette même phase l'utilisateur peut préciser la tache de l'application qui devra être activée en cas de "restart" automatique (à la suite d'un défaut secteur) ou d'une réinitialisation du système.

L'activation d'une tâche résidente ainsi que son désarmement sont réalisés au niveau de la machine par le jeu des instructions ARM et QUIT.

Une tache résidente ne peut être structurée en overlay.

2.3.5 TACHES NON RESIDENTES

la zone non résidente (ZNR) est réservée aux tâches non résidentes de l'application.

Le système peut gérer dans cette zone jusqu'à 128 tâches par FU disque reconnue par FMS16.

Le fichier image mémoire support de chaque tâche à pour nom TNRxyz, x, y et z étant trois chiffres décimaux constituant le numéro associé à chaque tâche (000 à 127).

L'appel d'une tache non résidente en mémoire est réalisé à l'aide d'une requête (TCALL ou TCALLW) précisant son numéro et sa FU support.

Le système (tâche PLOAD) mémorise les appels, alimente depuis le disque et active les tâches non résidentes dans l'ordre des appels; il n'existe ainsi aucune notion de priorité entre les différentes tâches non résidentes d'une application. Toutes ces tâches s'exécutent sous le niveau de priorité NRESID.

A un instant donné une seule tâche peut occuper la ZNR, cette zone n'étant libérée qu'en fin d'exécution par la requête EXIT.

Les tâches non résidentes peuvent être structurées en overlay, ce qui n'est pas le cas des taches résidentes.

Il est à signaler que. lorsqu'une tâche non résidente libère la ZNR à la fin de son exécution, elle n'est pas réécrite sur le disque. Lors de sa prochaine activation elle sera alimentée depuis le disque dans sa version initiale. Ceci interdit à une telle tâche de se communiquer des données par des mémoires de travail d'une exécution à une autre.

2.3.6 ZONES DE TRAVAIL DU BACKGROUND

La zone de travail du Background ZB permet le déroulement des processeurs du logiciel de base ou l'exploitation de programmes utilisateur s'exécutant en mode maître.

Cette zone s'étend implicitement jusqu'à la zone non résidente. c'est-à-dire jusqu'à AZNR-1.

Cependant la commande JOB comporte un paramètre permettant de préciser la taille mémoire requise pour qu'un travail puisse être réalisé.

On peut ainsi étendre la zone Background jusqu'à AZDR-1, un mécanisme de swap entrant alors en jeu : lorsque la zone non résidente est requise pour l'exécution d'une tâche. le système sauvegarde sur disque (swap-out) la partie de la ZNR utilisée par le Background, la restitution (swap-in) intervenant au moment de la relance du Background après que la tâche non résidente ait libéré la ZNR.

Ce mécanisme n'est appliqué qu'au Background, toute tâche temps réel restant dans la ZNR jusqu'à la fin de l'exécution en cours, puis étant remplacée par une autre tâche non résidente ou par le Background.

La zone ZB2 permet le déroulement des processeurs système ou utilisateur s'exécutant en mode esclave. Elle est située après le système.

2.3.7 PROTECTION ENTRE TACHES

Le dispositif DRPS16 de protection mémoire est obligatoire avec le système BOS16.

Les processeurs ainsi que les tâches de l'application peuvent se dérouler en mode maître.

La protection mémoire permet cependant aux tâches "vitales" de s'exécuter dans un environnement présentant un maximum de sécurité et de protéger le moniteur des programmes en cours de mise au point.

Lorsqu'une tache opère en mode esclave; un certain nombre d'instructions privilégiées lui sont interdites; elle dispose alors de requêtes programmées adressées au moniteur qui réalise toutes les opérations de contrôle, assurant ainsi à l'application une protection absolue.

2.3.8 COMMUNICATIONS ENTRE TACHES

Certaines tâches résident en mémoire centrale, d'autres sur disque. De plus la protection mémoire impose aux tâches en mémoire des espaces de travail disjoints.

Ces deux constatations imposent l'existence, au niveau du système, de moyens permettant la communication entre les différentes tâches de l'application.

BOS16 présente plusieurs outils pour répondre à cet objectif.

A) Fichiers de l'utilisateur :

FMS16 permet la gestion d'informations de type partageable ou non partageable, temporaire ou permanent.

Le partage de l'accès à l'information est réalisé à l'aide de clés et d'identificateurs.

FMS16 fournit des outils pour gérer automatiquement les conflits d'accès simultanés à l'information.

B) Zone de données résidentes :

La zone commune de données résidentes constitue un moyen complémentaire du précédent pour la communication entre tâches.

En raison de l'encombrement en mémoire centrale pouvant résulter d'une utilisation abusive de cette méthode, il est conseillé de limiter son emploi aux cas suivants :

- Le volume des informations communes n'est pas prohibitif
- La notion de rapidité d'accès à ces informations intervient de façon critique.

L'organisation de la zone de données résidentes est à la charge de l'usager; elle résulte de conventions entre les tâches de l'application.

Aucune protection n'est réalisée par le système : l'usager doit synchroniser les tâches faisant accès à cette zone pour gérer les informations qu'elle contient.

Pour l'accès à ces données résidentes, les tâches disposent de deux requêtes :

- REDGET, pour lire dans un buffer une portion de la zone de données résidentes
- REDPUT, pour transférer un buffer dans une portion de la zone de données résidentes.

C) Ressources de l'usager :

La ressource est un moyen qui peut être utilisé par les tâches ; elle permet la coexistence et l'exécution parallèle des tâches.

On distingue :

- les ressources "hardware" qui appartiennent au système et dont il vérifie l'utilisation afin d'éviter un blocage du système Iles périphériques, la mémoire centrale par exemple)
- les ressources "software" formées par l'ensemble des services de base du moniteur (accessibles par requêtes programmées) ainsi que des ressources créées par l'usager au moyen des instructions RQST et RLSE du SOLAR 16 permettant de demander un accès ou de libérer un accès à une ressource.

D) Paramètre de travail et compte-rendu de traitement :

L'appel d'une tache non résidente peut être effectué en précisant la nature du travail à réaliser à l'aide d'un paramètre de travail.

Ce paramètre est facultatif, la tache appelée en assurant seule la gestion.

Une tache non résidente peut être appelée par les requêtes TCALL et TCALLW, cette dernière provoquant la suspension de la tâche appelante jusqu'à la fin de l'exécution de la tâche appelée qui peut alors envoyer un compte-rendu du traitement effectué.

E) Evénements :

Les événements peuvent être utilisés comme outils de dialogue entre tâches :

- Pour matérialiser l'arrivée d'informations extérieures
- Pour représenter l'accomplissement de fonctions réalisées par l'utilisateur.

Un événement est partageable : plusieurs tâches peuvent attendre l'arrivée d'un même événement.

Un événement a deux états :

- Etat SET caractérisant l'événement arrivé
- Etat RESET caractérisant l'événement non arrivé.

Lors de l'arrivée d'un événement, toutes les tâches en attente de cet événement sont remises à l'état non masqué.

Quatre requêtes programmées permettent l'utilisation des événements :

- WEVENT pour attendre qu'un événement soit dans l'état SET
- SEVENT pour signaler l'arrivée d'un événement, c'est-à-dire le mettre dans l'état SET
- REVENT pour mettre un événement dans l'état RESET
- TEVENT pour tester l'état d'un événement.

Le nombre d'événements gérés par le système est configurable.

2.3.9 TRAITEMENT DE L'HORLOGE

L'option "Horloge temps réel" permet d'intégrer au système la requête WAIT qui permet la suspension de l'exécution de la tâche appelante pendant un délai spécifié, ce délai étant exprimé en nombre de tops de base.

Le système peut gérer simultanément jusqu'à 32 suspensions de tâches.

2.4 COMMANDES DE CONTROLE

Lorsque l'opérateur veut communiquer avec le système, il le fait par l'intermédiaire de "commandes" de contrôle qu'il émet depuis l'un des périphériques d'entrée affecté à ces commandes : clavier du périphérique de dialogue ou lecteur de cartes en général.

Une commande est constituée d'un enregistrement lu sur le périphérique associé à l'unité symbolique "Control Command" (CC).

En plus des commandes de base qui seront introduites dans les chapitres qui suivent, l'utilisateur dispose de macro-commandes des décrites en Annexe 5.

Une commande peut demander, par exemple, l'affectation d'une unité fonctionnelle à une unité symbolique, l'activation d'un processeur qui peut être aussi bien un programme du software de base (l'assembleur ou un utilitaire) qu'un programme utilisateur quelconque.

Le système indique qu'il est en attente d'une commande en écrivant sur le périphérique associé à l'unité symbolique "Listing Log" (LL) le message :

```
"cr" "lf" "."
```

A) Conventions d'écriture des commandes :

- Dans une commande les caractères "saut de ligne" (lf), "null", "perfo on". "perfo off" sont éliminés avant l'analyse.
- Le caractère "flèche arrière" "<-" provoque l'annulation de tous les caractères qui précèdent.
- Le caractère "flèche en l'air" "Î" provoque l'annulation du caractère valide qui précède. Plusieurs caractères "flèche en l'air" annulent autant de caractères.
- Plusieurs espaces consécutifs sont assimilés à un seul.
- Une seule commande peut se trouver par enregistrement. Le nom de cette commande doit figurer au début de l'enregistrement et se termine par un espace ou un "retour chariot".
- Une commande peut comporter des paramètres qui sont généralement séparés par des virgules.

B) Forme générale :

```
nomcom [para 1][,para 2]  ...  "cr"
```

où nomcom est le nom de la commande (seules les quatre premières lettres sont prises en compte si la commande en comporte plus).

Les commandes se divisent en deux classes. La première est constituée des commandes de base de BOS16. la seconde est constituée des commandes spécifiques de chaque processeur. Ces dernières ne sont valides qu'après l'activation du processeur (commandes CALL ou RUN suivant qu'il s'agit d'un processeur du software de base ou non) ; elles sont communiquées au superviseur par la requête CAMO.

Dans le premier cas, les paramètres sont analysés par le superviseur. Dans le deuxième cas ils sont analysés par le processeur lui-même.

En effet, lorsque le superviseur active un processeur, la base W contient l'adresse du buffer où se trouve la commande après suppression des caractères inutiles (conformément au paragraphe ci-dessus) et le registre X précise le rang du premier caractère suivant le nom de la commande (début du premier paramètre).

Le service HELP décrit au paragraphe 3.2.1 permet d'obtenir en ligne des informations sur les différentes commandes du système et des processeurs disponibles sous BOS16.

L'utilisation de macro-commandes permet, par un appel unique, de réaliser l'exécution d'une suite de commandes de BOS16 (cf. Annexe 5).

2.5 REQUETES PROGRAMMEES

Un programme utilisateur peut demander des services au système par l'intermédiaire de requêtes programmées.

Ces requêtes sont des appels à des sous-programmes superviseur ; elles sont assemblées dans le programme utilisateur et interprétées par le superviseur au moment de l'exécution.

Une requête est toujours constituée d'une instruction SVC (SuperVisory Call) ; les paramètres éventuels sont en général transmis par l'intermédiaire des registres.

Exemple :

```
LAI  2
SVC ABOS
```

Cette requête rend le contrôle au programme superviseur; l'accumulateur contient une valeur destinée au système.

Le traitement de chaque requête nécessite la disponibilité d'un certain nombre de mémoires dans la zone pointée par K. Pour BOS16 70 mots sont nécessaires.

Au retour de BOS16 la valeur du registre K est cependant restituée ce qui permet à l'utilisateur la sauvegarde éventuelle d'informations.

Exemple :

```
PSR  X        Sauvegarde de X
SVC  FREEM
PLR  X        Restitution de X
```

En règle générale. lorsqu'après le traitement d'une requête le système rend le contrôle au programme utilisateur :

- les valeurs des registres C, L, U, K, Y et B sont restituées
- la valeur du registre X est modifiée
- le registre A est utilisé pour la transmission d'un compte-rendu ou d'un résultat.

2.6 GESTION DES OPERATIONS D'ENTREE-SORTIE

Toutes les opérations d'entrée-sortie sont réalisées par le moniteur IOCS16 qui, par sa structure et sa modularité, minimise l'occupation de l'unité de traitement.

Sur les calculateurs SOLAR 16, IOCS16 est l'interface software standard entre les programmes et les périphériques de l'installation.

L'utilisateur peut demander des échanges avec retour immédiat, retour en fin d'échange ou avec utilisation du "Pool Buffer".

Il a en outre la possibilité de se synchroniser sur la fin d'un échange, de tuer un échange en cours.

IOCS16 assure l'indépendance des programmes vis-a-vis des périphériques de l'installation par l'utilisation des unités symboliques auxquelles peuvent être affectées à tout moment des unités fonctionnelles.

2.6.1 UNITES FONCTIONNELLES

Les unités fonctionnelles désignent un périphérique ou un sous-ensemble d'un périphérique.

Sous BOS16 elles sont en standard au nombre de 59.

Elles sont référençables de façon externe sous forme de symboles (ou clés) reconnus dans les commandes et de façon interne par des numéros (ou symbole valeur au niveau assembleur) qui sont utilisés comme paramètres des requêtes programmées :

Nom de l'unite fonctionnelle		Mnémonique	Valeur
Zéro	(Zéro)	ZE	'00
Lecteur téléimprimeur	(Teletype Reader)	TR	'01
Feuille téléimprimeur	(Teletype Sheet)	TS	'02
Clavier téléimprimeur	(Teletype Keyboard)	TK	'03
Perforateur téléimprimeur	(Teletype Punch)	TP	'04
Lecteur rapide	(High speed Reader)	HR	'05
Perforateur rapide	(High speed Punch)	HP	'06
Lecteur de cartes	(Card Reader)	CR	'07
Imprimante ligne	(Ligne Printer)	LP	'08
Bande magnétique	(Tape 1-4)	T1àT4	'09à'0C
Disque		D1àD8	'0Dà'14
Unités non affectées		F1àFF	'15à'23
Disque		D9àDF	'24à'2A
Disque		E1àEF	'2Bà'39
Mémoire		ME	'7F

Les unités F1 à FF permettent à l'utilisateur d'introduire de nouveaux périphériques dans la configuration.

Le nombre d'unités fonctionnelles d'une installation peut toutefois être supérieur à 59, la seule limitation étant imposée par GIO16 (125 au maximum) ; les unités de numéro supérieur à '39 ne sont alors pas accessibles par dialogue opérateur au moyen de mnémoniques, exception faite pour HE.

Une installation peut n'inclure qu'un sous-ensemble des unités fonctionnelles introduites dans le tableau précédent; le système impose cependant la présence des unités suivantes : TS, TK, D1, D2 et une FU permettant d'accéder à l'espace initial du disque système (cf. par. 6.4).

BOS16 s'adapte au moniteur IOCS16 qui lui est intégré au moment de la génération du système : le dialogue opérateur reconnaît, en tant qu'unités fonctionnelles, les seules clés correspondant aux unités fonctionnelles configurées dans IOCS16.

Exemple :

Lors de la génération d'IOCS16 à l'aide de GIO16, si le plus grand numéro d'unité fonctionnelle défini est 30 toutes les unités fonctionnelles de numéro supérieur à 30 ('1E) ne seront pas reconnues par le dialogue opérateur de BOS16.

La commande :

BO FA

sera interprétée par le système comme une demande d'association de l'unité symbolique BO au fichier de nom FA.

2.6.2 UNITES SYMBOLIQUES

Les unités symboliques permettent de distinguer les différents flots d'information caractéristiques d'un programme.

Sous BOS16 elles sont en standard au nombre de 25.

Elles sont référençables au niveau des commandes par des clés et au niveau des requêtes par des numéros (symboles valeurs en assembleur).

Nom de l'unite symbolique	Mnémonique	Valeur
Sortie symbolique (Symbolic Output)	SO	'80
Entrée symbolique (Symbolic Input)	SI	'81
Sortie binaire (Binary Output)	BO	'82
Entrée binaire (Binary Input)	BI	'83
Liste des messages (Listing Log)	LL	'84
Commandes de contrôle (Control Command)	CC	'85
Sortie de listes (List Output)	LO	'86
Correction des erreurs (Error Correction)	EC	'87
Liste des erreurs (Error Listing)	EL	'88
Commandes aux processeurs (Processor Command)	PC	'89
Unité non définie No1 (Undefined 1)	U1	'8A
Unité non définie No2 (Undefined 2)	U2	'8B
Unité non définie No3 (Undefined 3)	U3	'8C
Unité non définie No4 (Undefined 4)	U4	'8D
Unité non définie No5 (Undefined 5)	U5	'8E
Unité non définie No6 (Undefined 6)	U6	'8F
Unité non définie No7 (Undefined 7)	U7	'94
Unité non définie No8 (Undefined 8)	U8	'95
Unité non définie No9 (Undefined 9)	U9	'96
Unité non définie NoA (Undefined A)	UA	'97
Unité non définie NoB (Undefined B)	UB	'98
Unité non définie NoC (Undefined C)	UC	'99
Unité non définie NoD (Undefined D)	UD	'9A
Unité non définie NoE (Undefined E)	UE	'9B
Unité non définie NoF (Undefined F)	UF	'9C

Le système considère les unités symboliques U5 à UF comme étant réservées aux tâches de l'application (Foreground) ; elles ne font donc l'objet d'aucune réaffectation standard.

Le nombre d'unités symboliques d'une installation peut être inférieur à 25 ; le système impose cependant la présence des unités SO à U4.

BOS16 s'adapte au moniteur IOCS16 qui lui est intégré au moment de la génération du système : le dialogue opérateur reconnaît, en tant qu'unités symboliques, les seules clés correspondant aux unités symboliques configurées dans IOCS16.

Remarque :

Les numéros '90 à '93 ne sont pas gérés par BOS16; ils correspondent à quatre unités symboliques spécifiques de RTES16.

La configuration de toutes les unités symboliques standard de BOS16 nécessite alors la macro-instruction :

```
%SUMAX = 29
```

2.7 GESTION DU DISQUE

Le système de gestion de fichiers FMS16 est l'outil de stockage et d'accès à l'information sur disque utilisé par BOS16.

Le partage dynamique des fichiers qu'il réalise est particulièrement adapté au temps réel : plusieurs usagers peuvent se partager simultanément l'accès en lecture d'un même fichier permanent sans aucun souci de synchronisation.

A l'opposé tout fichier peut être protégé par l'intermédiaire de clés de verrouillage.

FMS16 gère dynamiquement l'espace disque, alloue et récupère automatiquement la place d'un fichier.

BOS16 propose trois types d'accès aux fichiers :

- le séquentiel
- l'indexé, les fichiers étant constitués d'articles de longueur variable identifies par leur nom
- le direct, les fichiers étant constitués d'articles de longueur fixe identifiés par leur numéro.

La structure modulaire de FMS16 permet de lui adjoindre des méthodes d'accès complémentaires qui correspondent à des types d'applications plus spécifiques.

Un ensemble d'utilitaires, FUP, apporte une aide à la manipulation des fichiers gérés par FMS16.

FUP permet de réaliser un certain nombre d'opérations telles que :

- initialisation, nettoyage, impression du contenu des unités fonctionnelles disque gérées par FMS16
- modification des fichiers permanents
- transfert d'information.

2.8 UTILISATION DE FMS16 PAR BOS16

Le système de fichiers FMS16 est utilisé par BOS16 pour ses propres besoins et pour en donner tous les services à l'utilisateur.

Nous n'introduisons ici que les notions nécessaires à la compréhension de cette notice, la lecture préalable du manuel de référence de FMS16 étant nécessaire.

2.8.1 DEFINITIONS ET NOTATIONS

On appelle catalogue un ensemble de fichiers que l'utilisateur regroupe pour les distinguer de ceux d'un autre utilisateur et éviter ainsi les problèmes d'homonymie entre fichiers sur une unité fonctionnelle disque donnée.

Sur l'unité D2 un catalogue dit catalogue système (:S) est ainsi réservé aux fichiers propres aux différents systèmes (BOS16, RTES16, ...).

On appelle bibliothèque un fichier indexé dont les articles sont des modules objets link-éditables servant à terminer une édition de liens pour la satisfaction des références non définies.

Par la suite on notera :

nomfic-catg
nomart.nomfic-catg

respectivement les noms de fichiers et d'articles nommés dans les commandes.

Ainsi :

TOTO-C4

représente le fichier TOTO du catalogue C4.

2.8.2 UTILISATION DES UNITES FONCTIONNELLES DISQUE

BOS16 impose les deux unités fonctionnelles disque D1 et D2.

L'unité D2 contient tous les fichiers utilisés par le moniteur et est donc gérée par FMS16; elle peut également être employée par l'utilisateur pour ses fichiers.

L'unité D1 par contre n'est pas gérée par FMS16 mais par IOCS16 et ne sert qu'à stocker les versions en cours des parties résidentes des différents systèmes ; elle sert en particulier à rappeler BOS16 au moyen au bootstrap disque. L'unité D1 peut contenir aussi une zone de swap.

2.8.3 ASSOCIATION UNITE SYMBOLIQUE - FICHIER

Les unités symboliques spécifiques de la production de programmes (SO à U4) peuvent être affectées par commandes à des fichiers de type séquentiel ou même à des articles de fichiers indexés.

Les processeurs de base de BOS16 peuvent ainsi, par le jeu des unités symboliques, utiliser des fichiers ; suivant les processeurs ces fichiers ont la nature suivante :

- entrée symbolique sur fichier séquentiel : c'est le cas des fichiers d'entrée des compilateurs, assembleur et macroprocesseur.

- modules binaires sur fichier séquentiel; c'est le cas des modules objets "link-éditables" (MOL) issus des compilateurs et de l'assembleur. C'est également le cas des modules binaires translatables produits par l'éditeur de liens.

- modules image mémoire sur fichier indexé : c'est le cas des modules produits par le BUILDER qui constitue des images mémoire directement exécutables à raison d'un article pour chaque branche et pour la racine.

2.9 MESSAGES D'ERREUR IMPRIMES PAR LE SYSTEME

BOS16 détecte un certain nombre d'erreurs et les signale à l'utilisateur en imprimant des messages d'erreur du type :

```
ERB n [inf1 [inf2]]
```

n représentant le numéro de l'erreur.

Dans certains cas le système fournit une et parfois deux informations supplémentaires (inf1, inf2).

Ainsi, dans le cas d'un défaut apparu sur un périphérique, le message se compose du numéro d'erreur 13, du numéro de l'unité fonctionnelle en défaut et du mot d'état de cette unité.

De plus, lorsque l'erreur est apparue au cours de l'exécution d'un processeur (système ou utilisateur) ou d'une tâche, BOS16 imprime :

- type (Hard ou Soft) et niveau de priorité
- numéro de la tâche dans le cas d'une erreur détectée au cours de l'exécution d'une tâche non résidente
- valeurs des différents registres constituant la PST.

Exemple :

Arrêt de l'exécution d'un programme s'exécutant en Background par quatre appels opérateur successifs :

```
ERB 10
TN = S '2D
A =  '0005  B = '0056  X = '0008  Y = 'FFCA  } contexte du programme
C =  '01B2  L = '0180  W = 'FFFC  K = '015D  } au moment où il a été
P =  '0180  S = '8000  O = '0800  E = '0FFF  } interrompu
```

On trouvera dans le manuel MEMO16 la liste des numéros des erreurs détectées par le système.

Le service HELP décrit au paragraphe 3.2.1 permet d'obtenir en ligne la signification des différents messages d'erreur émis par le système et les processeurs disponibles sous BOS16.

3 PRODUCTION DE PROGRAMMES OPERATIONNELS

3.1 GENERALITES

3.1.1 NOTIONS DE "JOB" ET DE "STEP"

La commande JOB permet de définir le début d'un travail. Ce travail est normalement indépendant du travail précédent et cette commande en assure l'indépendance en réinitialisant le système, en particulier les affectations standard, les fichiers ouverts,...

La notion de step, elle. est liée à l'évolution du travail et correspond au découpage en phases de celui-ci. Par définition nous dirons qu'un nouveau step commence à chaque fois que l'on réorganise la gestion de la mémoire en alimentant et/ou détruisant la présence d'un processeur. C'est ainsi que l'activation d'un processeur (système ou utilisateur) définit un nouveau step.

3.1.2 GESTION DE LA ZONE BACKGROUND ZB

Le découpage de la mémoire, exception faite pour la zone occupée par le moniteur, est à la charge de l'utilisateur.

Il est réalisé, lors de la configuration du système, par le jeu de la commande FORE qui permet de définir les zones utilisées ainsi que leurs limites.

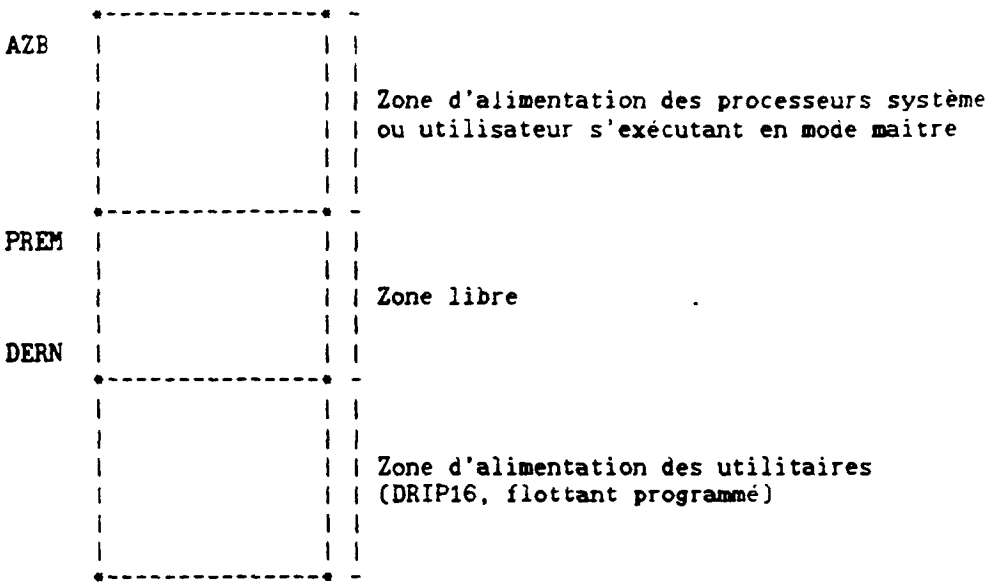
Cinq zones peuvent ainsi être configurées :

- zone Background ZB
- zone non résidente
- zone de données résidentes
- zone résidente
- zone Background ZB2.

La zone Background ZB, autrement dit la zone allouée à la fonction de production de programmes, s'étend implicitement jusqu'à la seconde zone configurée (en général la zone non résidente). Elle est réservée aux programmes s'exécutant en mode maître.

Certains jobs peuvent toutefois nécessiter un espace de travail plus important ; la commande JOB permet alors d'étendre la zone Background jusqu'à la fin de la zone non résidente (si une telle zone a été configurée].

L'utilisation de la zone Background correspond au schéma suivant :



A chaque step correspond une certaine implantation de la mémoire qui laisse une zone libre plus ou moins grande. Cette place est utilisable par les processeurs qui peuvent acquérir les valeurs des premières et dernières mémoires libres (PREM ou DERN) par une requête spéciale (FREEM).

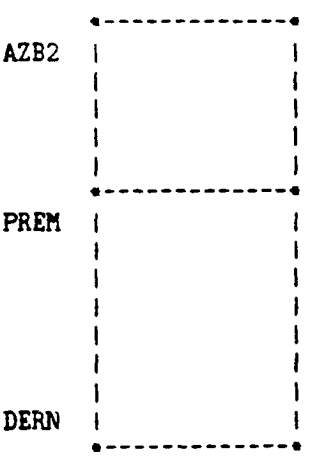
Un seul processeur est implanté en mémoire à un instant donné.

3.1.3 GESTION DE LA ZONE BACKGROUND ZB2

La zone Background ZB2 permet l'exécution des processeurs et des programmes de l'utilisateur en mode esclave.

Elle est implantée derrière la zone occupée par le système et a une taille de 32 K mots.

De même que pour la zone ZB, à chaque step correspond une implantation de la mémoire, les limites de la zone libre (PREM et DERN) pouvant être acquises par la requête FREEM.



Remarque :

La commande DUMP sans paramètre ainsi que la requête FREEM sont relatives à la zone (ZB ou ZB2) où se trouve le dernier programme chargé.

3.2.1 ? : SERVICE HELP

Cette commande permet d'obtenir sur l'unité LL des informations relatives aux c-des du système et des processeurs disponibles sous BOS16, ainsi que la signification des différents messages d'erreur qui sont émis.

A) Mode d'emploi de HELP

Forme :

```
?"cr"
```

B) Liste des commandes de BOS16 et des processeurs disponibles sous BOS16

Forme :

```
?BOS16"cr"
?nomproc"cr"
```

où nomproc est le nom du processeur.

Cl Syntaxe d'une commande

Forme :

```
?nomcde"cr"
```

où nomcde est le nom de la commande (seuls sont pris en compte les 4 premiers caractères).

D) Signification d'un libellé d'erreur

Forme :

```
?libellé d'erreur"cr"
```

Exemples :

```
?
?BOS16
?FUP3
?FDESCRIPT
?ERB 45 '600C
```

E) Définition du support de la base HELP

Forme :

```
FUHELP {SU|FU}"cr"
```

où {SUIFU} désigne l'unité fonctionnelle disque sur laquelle réside la base HELP.

3.2.2.1 Cas d'affectation des unités fonctionnelles

Ces commandes permettent de spécifier l'association d'un périphérique ou plus précisement d'une unité fonctionnelle type à une unité symbolique type. Une telle affectation restera valable jusqu'à la prochaine commande d'affectation portant sur la même unité symbolique. Cependant certaines erreurs peuvent provoquer des réaffectations standard.

A) Forme générale d'une commande d'affectation d'une unité fonctionnelle (FU) à une unité symbolique (SU)I :

Ces commandes sont toujours composées de la juxtaposition du nom d'unité symbolique type et du nom d'unité fonctionnelle type que l'on veut affecter à l'unité symbolique. Ces noms sont ceux définis aux paragraphes 2.6.1 et 2.6.2.

```
SU FU "cr"
```

Si la commande d'affectation est acceptée par le système celui-ci imprime le message :

```
"cr" "lf"
•
```

et passe en attente d'une nouvelle commande provenant du périphérique associé à l'unité symbolique CC.

Si la commande d'affectation n'est pas acceptée par le système, il y a impression du message :

```
ERB 07 "cr" "lf"
•
```

qui indique que l'on a voulu une affectation réputée impossible.

Exemple :

Si l'on suppose qu'a CC et LL sont affectés TK et TS, on peut avoir la suite de commandes et de messages suivante :

•BO HP	Affectation acceptée
•SI CR	Affectation acceptée
•LO HR	Affectation refusée car réputée impossible
ERB 07	Message d'erreur
•LO LP	Affectation acceptée

Remarque :

Lorsque le numéro de l'unité fonctionnelle que l'on désire affecter à une SU est supérieur à '39, aucun symbole ne lui est associé. L'utilisateur peut cependant demander l'affectation de la manière suivante :

```
SU =numfu "cr"
```

numfu étant le numéro hexadécimal de la FU.

Exemple : U1 ='3A

B) Tableau des affectations possibles :

A chaque unité symbolique type, il est possible d'associer uniquement un certain nombre d'unités fonctionnelles :

- seuls des périphériques permettant des sorties peuvent être affectés à des

unités symboliques auxquelles on a associe (par le nom) des opérations de sortie
- certaines unités fonctionnelles sont peu adaptées à la réalisation des opérations particulieres associées à une unité symbolique donnée. Il est ainsi impossible d'utiliser le clavier du téléimprimeur (TK) pour faire des entrées binaires (BI).

Ces affectations possibles et impossibles par des commandes a BOS16 sont résumées dans la matrice suivante. Une croix dans la case indique que l'affectation est réputée possible, un blanc signifie par contre qu'elle est impossible.

TABLEAU DES AFFECTATIONS POSSIBLES (X)

	UNITES SYMBOLIQUES										
UNITES FONCTIONNELLES	SO	SI	BO	BI	LL	CC	LO	EC	EL	PC	U1a UF
ZE	X		X		X		X		X		X
TR		X		X		X		X		X	X
TS	X		X		X		X		X		X
TK		X		X		X		X		X	X
TP	X		X		X		X		X		X
HR		X		X		X		X		X	X
HP	X		X		X		X		X		X
CR		X		X		X		X		X	X
LP	X		X		X		X		X		X
T1 à T4	X	X	X	X	X	X	X	X	X	X	X
D1 à DF											X
F1 à F5	X	X	X	X	X	X	X	X	X	X	X
E1 à EF											X

Remarque :

Si l'on affecte l'unité fonctionnelle ZE à une unité symbolique cela signifie que toutes les demandes d'échange portant sur cette unité symbolique seront sans effet (aucun échange).

C) Affectation standard :

Lorsque l'on initialise le système, celui-ci affecte de façon standard les unités fonctionnelles aux unités symboliques. C'est le cas aussi lors d'une disparition secteur ou d'une erreur fatale en mode train de travaux. Cette affectation est faite en fonction des périphériques disponibles. Elle est résumée dans le tableau ci-joint. Si avec la configuration périphérique dont on dispose plusieurs affectations apparaissent c'est l'affectation correspondant à la colonne la plus à droite qui est adoptée.

Les unités symboliques U5 à UF sont réservées au Foreground (tâches de l'application). En ce qui les concerne le système n'opère donc aucune affectation standard. C'est ainsi qu'une affectation portant sur l'une de ces unités reste valable jusqu'à une nouvelle affectation ou une réinitialisation du système (commande INIT).

		Si l'on a un lecteur de cartes	Si l'on a une imprimante ligne
SO	*		
SI	TS	CR	
BO	*		
BI	ZE		
LL	TS		
CC	TK		
LO	TS		LP
EC	TK		
EL	TS		
PC	TK		
U1	ZE		
U2	ZE		
U3	ZE		
U4	ZE		

- Les unités symboliques SO et BO sont affectées en standard à des fichiers temporaires système (référençables dans les commandes par les clés SO et BO).

Voir aussi le paragraphe 3.7.6 pour la réaffectation des unités BI, BO, SI, SO à chaque step.

Il y a réaffectation standard pour toutes les unités symboliques de SO à U4 après :

- un chargement du système
- un arrêt du système et redémarrage au point initial
- une commande (INIT) de réinitialisation de BOS16 (sont aussi réaffectées les unités symboliques U5 à UF)
- un redémarrage à la suite d'un défaut secteur : il y a alors eu impression du message ERB 08.

Il y a réaffectation standard uniquement pour les unités symboliques CC et LL dans tous les cas d'erreur en mode de fonctionnement conversationnel.

La commande BATCH permet de modifier les affectations standard de la façon suivante dans l'hypothèse où l'on dispose d'un lecteur de cartes et d'une imprimante :

CC est associé à CR
PC est associé à CR
EC est associé à CR
EL est associé à LP

Bans ce mode de fonctionnement la plupart des erreurs sont fatales et provoquent l'abandon du travail en cours. Le système réaffecte CH à CC et ignore les commandes lues jusqu'à la première commande EOJ.

La commande EBATCH permet de revenir aux affectations standards de type conversationnel.

3.2.2.2 Cas d'affectation des fichiers

Aux unités symboliques peuvent également être affectées des fichiers gérés par le système de fichiers FMS16.

La commande a alors la forme suivante :

```
SU [art.]nomfic[-catg][,FU]"cr"
SU {SU'|*|fnum}"cr"
```

où :

- art : désigne le nom d'un article du fichier qui doit alors être de type indexé (par défaut le fichier doit être de type séquentiel)
- nomfic : nom du fichier
- catg : est le nom du catalogue auquel appartient ce fichier (par défaut on utilise le nom de catalogue déclaré dans la commande JOB)
- FU : désigne le support sur lequel se trouve ce fichier (par défaut on utilise l'unité fonctionneile déclarée dans la commande JOB)
- su, su' : sont des noms d'unité symboliques
- fnum : est un numéro d'utilisation de fichier.

A) SU [art.]nomfic[-catg][,FU]

Deux cas se présentent :

- le fichier existe. Il est ouvert.

- le fichier n'existe pas. Il est créé de type permanent (indexé si la commande comporte un nom d'article, séquentiel sinon) lorsque SU représente le nom d'une unité symbolique de sortie. Dans le cas contraire la commande est refusée.

B) su l

Cette commande permet d'affecter à l'unité symbolique SU un fichier temporaire de type séquentiel qui est alors ouvert par le système sur l'unité fonctionnelle disque précisée dans la commande JOB (cf. par. 3.2.5).

C) su SU'

Cette commande permet d'affecter à l'unité SU le fichier ou l'unité fonctionnelle affectée à l'unité SU'. Dans le second cas ne sont acceptées que les affectations possibles (cf. Tableau des affectations possibles : par 3.2.2.1).

D) SU fnum

Cette commande permet d'affecter à l'unité SU le fichier actuellement ouvert avec le numéro d'utilisation fnum.

3.2.2.3 Cas de l'unité CC

Pour cette unité à chaque changement d'affectation. on mémorise l'affectation précédente et la commande :

RETURN "cr"

permet de revenir à l'affectation antérieure. La mémorisation ne se fait qu'a un niveau.

Cette commande a pour but essentiel le retour à l'affectation antérieure de CC après un passage sous le contrôle de l'opérateur (à la suite d'un appel, d'un défaut périphérique ou d'une commande PAUSE) en mode train de travaux.

3.2.3 CALL : ACTIVATION D'UN PROCESSEUR DU SOFTWARE DE BASE

Cette commande permet d'effectuer le chargement et le lancement des processeurs du software de base (assembleur. compilateurs,...).

Forme :

```
CALL nomproc[,FU|SU]"cr"
```

où nomproc est le nom du processeur, FU ou SU désignant le support sur lequel il se trouve. Par défaut, il s'agit de D2.

Exemple : CALL ASM permet d'activer le macro-assembleur

Les processeurs du software de base sont stockés en image mémoire sur le disque à raison d'un fichier indexé du catalogue système par processeur.

On peut par simple adjonction d'un nouveau fichier ajouter un nouveau processeur à la liste des processeurs qui résident déjà sur le disque.

Le lancement de chacun de ces processeurs provoque la communication immédiate d'une liste de commandes (cf. requête CAMO : par. 3.3.3). Ces commandes sont spécifiques de chaque processeur (cf. Manuel de référence associé) et permettent de leur préciser les fonctions désirées ainsi que les valeurs des paramètres éventuels.

La commande CALL définit le début d'un nouveau step et provoque de ce fait la réaffectation automatique des unités symboliques SO et BO aux fichiers système (cf. Enchaînement des steps : par. 3.6.2).

Les commandes spécifiques de chaque processeur ne sont valides que pendant la durée du step et sont donc annulées à chaque nouveau step (CALL, RUN, LOAD et FLOAT).

3.2.4 RUN. LOAD. FLOAT et START : MISE EN OEUVRE D'UN PROCESSEUR UTILISATEUR

Al Commande RUN

Forme 1 :

```
RUN [nomproc][,adlanc]  "cr"
```

Forme 2 :

```
RUN [nomproc],[adlanc],liste d'utilitaires "cr"
```

B) Commande LOAD

Forme 1 :

```
LOAD [nomproc]  "cr"
```

Forme 2 :

```
LOAD [nomproc],liste d'utilitaires "cr"
```

C) Commande FLOAT

Forme :

```
FLOAT "cr"
```

D) Commande START

Forme :

START “cr”

où :

nomproc	désigne le nom du processeur (sous la forme nomfic[-catg])
adlanc	est l’adresse de lancement du processeur (hexadécimale)
liste d’utilitaires	un ou les deux (dans ce cas séparés par une virgule) noms suivants: DRIP16. FLOAT

Le fichier supportant le processeur appartient à la FU spécifiée par la commande JOB.

RUN permet le chargement en mémoire et le lancement d’un processeur utilisateur.

LOAD permet le chargement en mémoire d’un processeur utilisateur.

Lorsque ces deux commandes comportent une liste d’utilitaires, elles permettent la mise en oeuvre des utilitaires spécifies pour la durée du step (jusqu’à la prochaine commande CALL, RUN, LOAD ou FLOAT).

FLOAT permet la mise en oeuvre du flottant programmé pour la durée du job.

START permet de lancer un processeur chargé en mémoire par la commande LOAD.

Ces quatre commandes seront détaillées au chapitre 4 [cf. Exécution d’un programme : par. 4.2).

3.2.5 JOB : DEBUT DE TRAVAIL

Cette commande permet de définir le début d’un nouveau travail et provoque la réinitialisation du système.
Forme :

JOB nom,[catg],[FU],[min][,taille] "cr"

où :

nom	est un symbole alphanumérique de 6 caractères maximum servant à identifier le travail
catg	est le nom du catalogue pris par défaut pour la recherche des fichiers nommés dans les commandes. Par défaut ce nom est “blanc” c’est-a-dire celui d’un catalogue commun.
FU	est le nom de la FU où se trouve le catalogue précédent. Par défaut on prend la FU du système (D2).
min	permet de spécifier la durée maximum (en minutes] d’un travail. Cette disposition n’est effective que si l’option horloge temps réel est présente. Par défaut le système alloue 10 minutes au travail. De plus le contrôle n’est effectif que si la commande BATCH a été émise (le maximum autorisé est 2 heures).
taille	est la taille mémoire requise pour l’exécution du JOB (cf. 3.1.2) exprimée en nombre de K mots. Par défaut le système alloue la totalité de la zone Background. Ce paramètre n’est pris en compte que si l’exécution a lieu dans la zone ZB et si la zone ZNR est de longueur non nulle. Il représente toujours un nombre de K mots dans ZB.

3.2.6 EOJ : FIN DE TRAVAIL

Cette commande détermine la fin d'un travail et provoque la fermeture des fichiers permanents ouverts et la destruction des fichiers temporaires.

Forme :

```
EOJ "cr"
```

3.2.7 CATAL : CATALOGAGE D UN FICHIER TEMPORAIRE

Cette commande permet de rendre permanent un fichier temporaire.

Forme :

```
CATAL {fnum|SU|IM},nomfic[-catg]  "cr"
```

où :

fnum	est le numéro du fichier temporaire que l'on catalogue. Ce numéro peut être désigné symboliquement par le nom de la SU dans le cas des fichiers temporaires système.
nomfic	est le nom que l'on donne au fichier
catg	est le nom du catalogue dans lequel on veut ranger le fichier. Par défaut on utilise le nom de catalogue désigné dans la commande JOB.
In	désigne l'image mémoire créée par le BUILDER dans un fichier temporaire (cf. par. 4.1).

L'opération consistant à cataloguer un fichier ne ferme pas celui-ci qui reste donc utilisable sous le même numéro d'accès.

Lorsqu'un fichier a été associé par le système à une unité symbolique il est possible au niveau des commandes CATAL, CLOSE. DELET, REWIND et RENAM de spécifier en paramètre le nom de cette SU.

Exemple :

CALL ASM	Ouverture par BOS16 d'un fichier temporaire associé à BO
SI FICSYM IASM	Association à SI du fichier FICSYM
CATAL BO,FICBIN	Catalogue du fichier associé à BO
CLOSE SI	Fermeture du fichier associé à SI

BOS16 se charge d'adresser à FMS16 les primitives comportant les numéros d'utilisation de ces fichiers.

3.2.8 CREATE : CREATION D'UN FICHIER PERMANENT

Cette commande permet de créer un fichier permanent de type quelconque et le rend utilisable directement sans déclaration programmée.

Forme :

```
CREATE fnum,nomfic[-catg],type|,{SU|FU}]  "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
nomfic	est le nom du fichier
catg	est le nom du catalogue (par défaut celui de la commande JOB)

type	est le type de fichier : S séquentiel I(i) indexé avec i articles D(i.j) direct avec i articles de j octets
{SU|FU}	désigne l'unité fonctionnelle sur laquelle résidera le fichier (par défaut FU spécifiée par la commande JOB).

3.2.9 OPEN : OUVERTURE D'UN FICHIER

A) Cas d'un fichier permanent :

La commande permet de définir un numéro d'utilisation du fichier.

Forme :

```
OPEN OLD fnum,nomfic[-catg][,{SU|FU}]  "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
nomfic	est le nom du fichier
catg	est le nom du catalogue (par défaut celui de la commande JOB)
{SU|FU}	désigne l'unité fonctionnelle sur laquelle réside le fichier (par défaut FU spécifiée par la commande JOB).

B) Cas d'un fichier temporaire :

La commande permet de créer un fichier temporaire et de lui associer un numéro d'utilisation :

```
OPEN NEW fnum,nomfic,type[,{SU|FU}]  "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
nomfic	est le nom du fichier
type	est le type du fichier : S séquentiel I(i) indexé avec i articles D(i,j) direct avec i articles de j octets
{SU|FU}	désigne l'unité fonctionnelle sur laquelle résidera le fichier (par défaut FU spécifiée par la commande JOB).

3.2.10 CLOSE : FERMETURE D'UN FICHIER

Permet d'effectuer la fermeture de l'accès spécifié. S'il s'agit d'un accès à un fichier temporaire, ce dernier est détruit.

Forme :

```
CLOSE {fnum|SU|IM} "cr"
```

où :

fnum	est le nom d'utilisation du fichier
su	désigne symboliquement le fichier associé à la SU
IM	désigne l'image mémoire créée par le BUILDER dans un fichier temporaire.

3.2.11 DELET : DESTRUCTION D'UN FICHIER

Cette commande permet de détruire un article ou un fichier permanent et l'unité d'accès qu'il utilisait.

Forme :

```
DELET {fnum|SU|[art.]nomfic[-catg][,FU]} "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
SU	désigne symboliquement le fichier associé à la SU
art	est le nom de l'article
nomfic	est le nom du fichier
catg	est le nom du catalogue (par défaut celui de la commande JOB)
FU	désigne l'unité fonctionnelle sur laquelle réside le fichier (par défaut FU spécifiée par la commande JOB).

Exemple

```
CALL   FUP3
FDUP,FICH-PW,D3,,D4
DELET   FICH-PW,D3
```

Transfert du fichier FICH de catalogue PW de D3 dans D4.

3.2.12 REWIND : SAUT AU DEBUT D'UN ARTICLE

Cette commande permet de positionner le pointeur courant d'un fichier au début de l'article précédemment sélectionné.

Forme :

```
REWIND {fnum|SU} "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
su	désigne symboliquement le fichier associé à la SU.

Pour FMS16 un fichier séquentiel est mono-article. La commande REWIND s'appliquant à un tel fichier positionne donc le pointeur courant au début du fichier.

3.2.13 RENAM : CHANGEMENT DU NOM D'UN FICHIER

Cette commande (RENAM) permet de changer le nom d'un fichier permanent.

Forme :

```
RENAM {fnum|SU},nomfic[-catg]  "cr"
```

où :

fnum	est le numéro d'utilisation du fichier
su	désigne symboliquement le fichier associé à la SU
nomfic	est le nom du fichier
catg	est le nom du catalogue (par défaut celui de la commande JOB).

Exemple :

```
/SI FICH-PW,D3
/RENA SI,FICH1-RW
/CLOS SI
```

3.2.14 BATCH : PASSAGE EN MODE TRAIN DE TRAVAUX

Forme :

BATCH "cr"

Cette commande fait passer le superviseur en mode train de travaux et provoque :

- la réaffectation suivante si la configuration comporte un lecteur de cartes et une imprimante :

  CC est affecté à CR
  PC est affecté à CR
  EC est affecté à CR
  EL est affecté à LP

- le contrôle des durées de travaux si l'option horloge temps réel est présente.
- impose le caractère "/" au début de chaque commande.

Remarque : En mode train de travaux, les commandes ne sont plus recopiées sur l'unité symbolique LL, mais sur EL.

3.2.15 EBATCH : FIN DU MODE TRAIN DE TRAVAUX

Forme :

EBATCH "cr"

Cette commande annule les effets de la commande BATCH.

On revient donc aux affectations standard ; le contrôle de durée est supprimé et le caractère "/" devient optionnel.

3.2.16 MSG : MESSAGE A L'OPERATEUR

Forme :

MSG texte "cr"

Cette commande permet de provoquer l'impression du message "texte" sur l'unité symbolique LL.

3.2.17 PAUSE : MESSAGE A L'OPERATEUR ET ATTENTE

Forme :

PAUSE texte "cr"

Cette commande permet également d'imprimer un message sur l'unité symbolique LL mais après réaffectation standard des unités CC et LL.

Seul l'opérateur pourra donc à l'aide de la commande RETURN relancer un éventuel train de travaux après avoir effectué les opérations demandées par le ou les messages.

Par la commande RETURN l'opérateur peut en effet redonner à l'unité symbolique CC son affectation précédente ce gui permet donc de continuer le traitement du travail.

3.2.18 IF : ENCHAINEMENT CONDITIONNEL

A) Code de retour d'un processeur :

Lors de la fin d'exécution d'un processeur il y a un retour au superviseur par une requête SVC ABOS. Le contenu de l'accumulateur à cet instant est appelé code de retour (R). Il est mémorisé par le système et peut être édité au moyen de la commande de PRINT (cf. par. 3.2.20).

Le code de retour permet d'indiquer la façon dont s'est exécutée la dernière phase du travail en cours.

La signification du code est définie par chaque processeur. Ainsi avec l'assembleur deux possibilités sont rencontrées :

R = 0 Assemblage correct
R = n Nombre d'erreurs

B) Utilisation du code de retour (commande IF) :

La valeur du code de retour transmis au système par le processeur précédent peut être utilisée dans la commande d'enchaînement conditionnel IF.

Forme :

IF R{<I=I>}i,n "cr"

Le code de retour est comparé au nombre décimal i.

Si la condition demandée est satisfaite les n commandes qui suivent sur l'unité CC sont ignorées (si n = 0 il y a abandon du JOB en cours).

Dans le cas contraire le système passe à l'analyse de la phrase de commande suivante, sans traitement du nombre n.

E x e m p l e :

```
/CALL ASM                 |
/SI FIC                   | Assemblage
/IASM                     |
/IF R>0,0                 Abandon du JOB si erreur
/CALL EDILE               |
/ILNK                     | Edition de liens
.....
.....                     |
```

Remarque :

Pour comptabiliser les commandes à ignorer le système reconnaît une commande lors de la présence du caractère "/" en début d'enregistrement.

Ceci permet notamment à BOS16 de ne pas considérer comme commande une carte destinée à un processeur.

3.2.19 DUMP : IMPRESSION DU CONTENU D'UNE ZONE DE MEMOIRE

Forme :

DUMP [début,fin] "cr"

Cette commande permet d'imprimer le contenu de la zone mémoire comprise entre les adresses "début" et "fin" incluses, sur l'unité LO.

Les adresses doivent être données en hexadécimal.

La commande DUMP sans aucun paramètre liste la zone de mémoire ZB ou ZB2 où se trouve le dernier programme chargé.

3.2.20 PRINT : EDITION DE LA MEMOIRE

Forme :

PRINT adresse[,nombre] "cr"

Cette commande permet l'édition sur le périphérique associé à l'unité symbolique EL du contenu d'une mémoire (éventuellement de plusieurs, le nombre étant alors le second paramètre de la commande).

L'adresse mémoire et le nombre sont à exprimer respectivement en hexadécimal et en décimal.

Remarque :

La commande PRINT sans aucun paramètre permet d'éditer le code de retour communiqué au système par le dernier processeur activé.

3.2.21 ABORT : ABANDON D'UN TRAVAIL

Forme :

ABORT "cr"

En mode train de travaux un appel quatre coups ou un défaut périphérique détecté par le système provoquent l'impression d'un message d'erreur et la réaffectation à l'unité symbolique CC du clavier du téléimprimeur.

La décision prise alors par l'opérateur peut être l'abandon du travail en cours.

Ceci est réalisé par l'émission de la commande ABORT qui affecte à CC le lecteur de cartes.

Les commandes suivantes sont alors lues et ignorées jusqu'à la prochaine commande EOJ.

3.2.22 EBOS : FIN D'UTILISATION DU MONITEUR

Forme :

EBOS "cr"

Cette commande provoque la mise du moniteur sur une boucle d'inaction. Le téléimprimeur est mis hors tension et seul un appel opérateur permettra de relancer le moniteur.

Après une commande EBOS tout défaut périphérique ou toute erreur survenant dans une tâche de l'application sont signalées par le système qui redevient ensuite inactif.

3.2.23 POFF : ARRET DU LECTEUR DE CARTES

Forme :

POFF "cr"

Cette commande provoque la mise hors tension du lecteur de cartes.

3.2.24 TIME : DEMANDE ET MISE A JOUR DE L'HEURE ET DE LA DATE

A) Demande de l'heure

Forme :

TIME "cr"

Cette commande provoque l'impression sur EL de la date et de l'heure.

Exemple :

16/01/74 07H 35M 55S

B) Mise à jour de l'heure et de la date

Forme :

```
TIME j,m,a [h,m[,s]]"cr"
```

Cette commande permet de réinitialiser la date et éventuellement l'heure gérées par le système.

3.2.25 PAGE : SAUT DE PAGE

Forme :

PAGE [n] "cr"

Cette commande permet d'effectuer n sauts de page sur LO et d'imprimer sur chacune des pages un en-tête avec nom éventuel du job, date et numéro de page.

Par défaut un seul saut est effectué.

Cette commande est ineffective si TS est affecté à LO.

3.2.26 GSYS : CONFIGURATION D'UN SYSTEME SOUS BOS16

La commande GSYS permet d'implanter dans une FU bootstrap un système qui sera ensuite rappelable par RAPD.

Forme :

```
GSYS n,nomfic,adk[,{SU1|FU1},{SU2|FU2}]  "cr"
```

Le fichier image mémoire du système est un fichier de nom nomfic. de catalogue :s, implanté dans l'unité fonctionnelle désignée par SU1 ou FU1 (D2 par défaut).

Le système est à implanter dans l'unité fonctionnelle désignée par SU2 ou FU2 (D1 par défaut), à l'adresse adk exprimée en secteurs. Son numéro est n.

Exemple :

```
GSYS 1,RTE16,300
```

Remarque :

Lorsque la commande GSYS entraîne l'écrasement du système de numéro n déjà implanté dans la FU bootstrap, BOS16 imprime :

ERB 38 No du système

L'utilisateur peut alors, s'il le désire, visualiser les limites des systèmes bootstrapables (commande MAP), éventuellement supprimer un OU plusieurs systèmes (commande de DSYS), avant de réémettre la commande GSYS.

3.2.27 MAP : IMPRESSION DE LA LISTE DES VACATIONS GEREES PAR RAPD

Forme :

```
MAP [{SU|FU}]  "cr"
```

sous BOS16 l'émission de la commande MAP permet d'obtenir sur le périphérique associé à l'unité symbolique EL la liste des numéros des systèmes gérés par RAPD. les bornes de la zone qu'ils occupent sur le disque ainsi que les noms des fichiers supportant les systèmes.

Les adresses données par la commande de MAP sont relatives au début du disque.

Le disque est identifié par l'unité fonctionnelle désignée par SU ou FU (D1 par défaut).

3.2.28 INIT : CHANGEMENT DE VACATION

A partir du moment où un système de numéro n a été configuré dans une FU bootstrap par une commande GSYS ou CONF (cf. par. 5.1.11) il est possible. depuis BOS16, de lui donner le contrôle. Il suffit d'émettre la commande INIT dont la forme est la suivante :

```
INIT [n][,FU]  "cr"
```

La commande INIT sans paramètre provoque la réinitialisation complète du moniteur en cours par relecture de la version sauvegardée sur le disque dans DI. Au préalable tous les fichiers ouverts sont fermés et les opérations d'entrée-sortie en cours abandonnées.

Le paramètre FU désigne la FU bootstrap (DI par défaut).

Remarque :

Si le paramètre n est supérieur au plus grand numéro de système connu de RAPD, le système de numéro 0 est rappelé. Il en est de même lorsque aucun système de numéro n n'a été configuré dans la FU bootstrap.

3.2.29 DSYS : SUPPRESSION D'UNE VACATION GEREE PAR RAPD

Forme :

```
DSYS n[,{SU|FU}]  "cr"
```

La commande DSYS permet la suppression dans la FU bootstrap désignée par SU ou FU (DI par défaut) du système de numéro n appelable par RAPD. Son action est limitée aux informations gérées par le bootstrap. Il n'y a donc pas de destruction du fichier support du système.

3.2.30 TPIO : POSITIONNEMENT DES DEROULEURS DE BANDES MAGNETIQUES

La commande TPIO réalise toutes les fonctions de positionnement des dérouleurs de bandes magnétiques.

Forme :

```
TPIO posit[,{SU|FU}]  "cr"
posit=REWI|RWUL|BSFI|BSRE|FSFI|FSRE|WRMA|WTMA|WGAP|SKIP [n]
```

Description des paramètres :

- le paramètre {SUIFU} doit correspondre à un dérouleur T1, T2, T3 ou T4 ; il peut être omis ; dans ce cas, la commande s'adresse à T1,
- REWI exécute un rebobinage à grande vitesse et un positionnement sur la marque de début de bande,
- RWUL exécute un rebobinage à grande vitesse, un positionnement sur la marque de début de bande et un passage en local du dérouleur; cette commande n'est effective qu'avec certains modèles de dérouleurs,
- BSFI exécute un saut arrière d'un fichier et un positionnement devant la marque EOF,
- BSRE exécute un saut arrière d'un bloc,
- FSFI exécute un saut avant d'un fichier et un positionnement derrière la marque EOF,
- FSRE exécute un saut axant d'un bloc,
- WRMA écrit un EOF.
- WTMA écrit la marque fin de bande (2 EOF),
- WGAP efface 10 cm de bande,
- SKIP exécute un saut avant de n fichiers ; si n n'est pas spécifié, un seul fichier est sauté.

Exemples :

```
•TPIO  REWI        }
•TPIO  REWI,T1     } rebobinage de la bande montée sur T1

•TPIO  FSFI,T2     }
•TPIO  SKIP,T2     } saut de 1 fichier sur T2

•L0    T4          }

•TPIO  SKIP 6,L0   } saut de 6 fichiers sur T4.
```

3.2.31 L'APPEL OPERATEUR

L'opérateur peut communiquer avec le système même si celui-ci n'est pas en attente d'une commande. Il suffit pour cela qu'il appuie sur la touche "BREAK" du périphérique de dialogue. Le programme qui s'exécute à ce moment-là, s'il teste les appels opérateur par une requête (SVC TAPS), sera arrêté et après impression du message :

```
ERB 14 "cr" "lf"
•
```

le système passera en attente d'une commande sur le périphérique associé de façon standard à l'unité symbolique "Control Command" (CC).

Cependant si le programme ne teste pas les appels opérateur. il existe quand même une possibilité de l'interrompre en faisant "quatre appels opérateur" consécutifs. Le système imprimera alors le message :

```
ERB 10 "cr" "lf"
•
```

et, après réaffectation standard des unité symboliques CC et LL. BOS16 se mettra en attente d'une commande provenant de l'unité symbolique "Control Command" (CC].

Nous noterons ici qu'un programme dont l'exécution a été interrompue Par un appel opérateur unique (impression du message ERB 14) pourra être réactive par toutes les commandes associées à ce programme alors qu'après "quatre appels opérateur" consécutifs (impression du message ERB 10). le programme interrompu ne pourra être réactivé que par une commande particulière, si celle-ci a été prévue.

L'appel opérateur simple suffit en général à interrompre le déroulement de programmes déjà éprouvés tels que ceux du software de base : assembleur, chargeur, éditeur de liens etc... Cependant, lorsqu'un programme que l'on met au point "boucle", le seul moyen de l'interrompre sera de faire quatre appels opérateur consécutifs.

Seuls les processeurs ayant en entrée des programmes symboliques (assembleur, éditeur de texte, macro-processeur) peuvent être relancés, après un appel opérateur, à l'aide d'une commande appropriée.

Remarque :

Si le périphérique de dialogue est un téléimprimeur, et que celui-ci reste en entrée de commande pendant plus d'une vingtaine de secondes sans qu'aucun caractère soit frappé, il y a un défaut "time-out". Un message ERB05 est alors imprimé. Pour relancer le dialogue il suffit de faire un "appel opérateur" (touche BREAK).

3.2.32 CHANGEMENT DE PERIPHERIQUE DE DIALOGUE

En standard l'utilisateur communique avec le système par l'intermédiaire du périphérique de dialogue dont l'adresse coupleur est '17F8.

Il est cependant possible de dialoguer à partir d'un autre périphérique, dont la FU associée est spécifiée à BOS16 par son nom ou son numéro :

```
FUCC {FU|=numfu} "cr"
```

La commutation entre le périphérique standard et celui désigné par FUCC est réalisée par la commande :

```
AVOU "cr"
```

3.3 REQUETES PROGRAMMEES

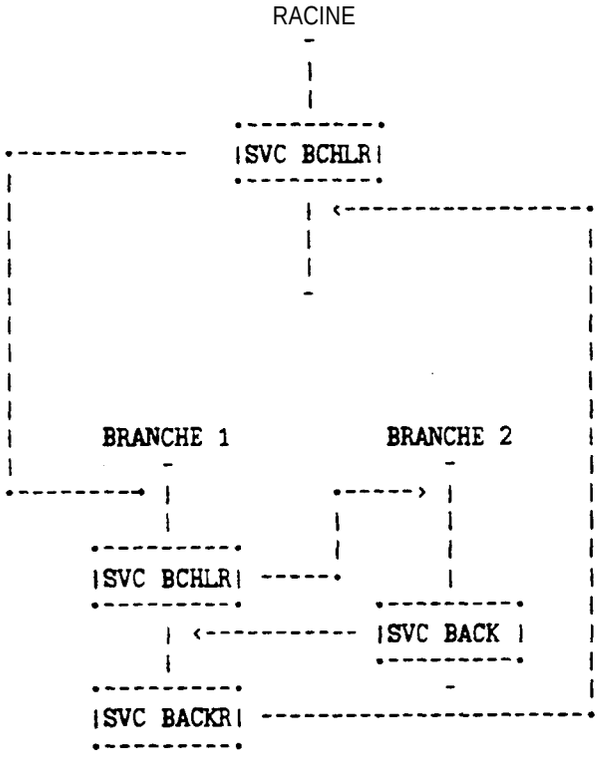
3.3.1 REQUETES DE SEGMENTATION : BCHLR, BACK, BACKR

3.3.1.1 Généralités

La segmentation permet d'exécuter des programmes dont la taille excède la taille disponible en mémoire vive.

On découpe alors le programme en parties dont l'une appelée racine est résidente et les autres appelées branches sont stockées sur disque et chargées à la suite de la racine chaque fois qu'il est nécessaire.

Sous BOS16 il ne peut y avoir qu'un niveau de branche conformément au schéma suivant :



Une branche peut être appelée depuis la racine ou depuis une autre branche

Une branche peut effectuer un retour à l'appelant ou à la racine.

Le retour à l'appelant n'est fait qu'à un niveau.

Tout appel de branche recouvre la précédente sans sauvegarde de celle-ci.

3.3.1.2 Fichier de programme segmenté

La racine et les différentes branches d'un programme segmenté sont rangées sur disque dans le format image mémoire et constituent chacune un article d'un fichier indexé géré par FMS.

Le nom de l'article doit être le nom de la branche.

Le nom du fichier est mémorisé par le superviseur au chargement du programme à exécuter. Il en est de même pour l'adresse d'implantation des branches.

3.3.1.3 Requêtes

A) BCHLR (Branch load and run)

Cette requête provoque le chargement de la branche spécifiée et son lancement à l'adresse contenue dans son premier mot. L'adresse de la requête est mémorisée pour un retour éventuel en séquence.

- Séquence d'appel en assembleur :

L'adresse de la table précisant le nom de la branche est transmise dans A. La forme générale d'une requête BCHLR est donc la suivante :

```
LAD BCB
SVC BCHLR (61)
```

- Description de la table des paramètres :

Nom de la branche : 6 caractères alphanumériques (ASCII) complétés éventuellement par des octets nuls.

- Compte-rendu :

1 Requête acceptée

2 Nom de branche inexistant

4 Requête interdite aux tâches hardware

6 Adresse du BCB incorrecte

Ce compte-rendu est donné dans l'accumulateur.

B) BACK (Branch back)

Cette requête signale la fin de l'exécution de la branche en cours et demande le retour à l'appelant. Cet appelant peut être la racine ou une branche qui sera alors rechargée au préalable.

- Séquence d'appel en assembleur :

Cette requête est toujours composée d'une seule instruction :

```
SVC BACK (62)
```

- Compte-rendu :

1 Requête acceptée

3 Requête exécutée dans la racine

4 Requête interdite aux taches hardware

Ce compte-rendu est donné dans l'accumulateur.

C) BACKR (Back to root)

Cette requête signale la fin d'exécution de la branche en cours et demande le retour direct à la racine à la suite de la dernière demande de branche.

- Séquence d'appel en assembleur :

Cette requête est toujours composée d'une seule Instruction :

```
SVC BACKR (63)
```

- Compte-rendu :

Le compte-rendu, identique à celui de la requête BACK, est donné dans l'accumulateur.

3.3.2 REQUETES DE PAGINATION DE DONNEES : DPAG. SPAG. LPAG

BOS16 permet de manipuler un tableau de mots de taille supérieure à la taille disponible en mémoire vive en segmentant ce tableau en pages qui sont rangées sur le disque et gérées par le superviseur.

La taille des pages est définie par la taille du buffer fourni par le processeur.

Le fichier utilisé pour le rangement est temporaire et inaccessible à l'utilisateur. Il est ouvert dans la FU disque spécifiée par la commande JOB (par défaut il s'agit de D2).

Trois requêtes permettent l'utilisation de cette pagination.

A) DPAG : définition des paramètres

La requête DPAG permet de communiquer au système :

- La taille du tableau qui est supérieure à la taille disponible en mémoire vive, ceci étant réalisé par l'intermédiaire de l'indice de son dernier mot (taille-l),
- L'adresse du mot où seront rangés le compte-rendu de la requête elle-même, mais également ceux des requêtes LPAG et SPAG ultérieures,
  L'adresse et la taille du buffer utilisable par le système.

- Séquence d'appel en assembleur :

L'adresse de la table des paramètres de la pagination est transmise dans A. La forme générale d'une requête DPAG est donc la suivante :

```
LAD RCB
SVC DPAG (48)
```

- Description de la table des paramètres :

0 Indice du dernier mot

1 Adresse du compte-rendu

2 Adresse du buffer

3 Taille du buffer

- Compte-rendu :

1 Requête acceptée

5 Disque saturé

6 Un paramètre au moins est incorrect

Le compte-rendu est fourni simultanément dans l'accumulateur.

B) SPAG : rangement de l'accumulateur

Cette requête provoque le rangement de l'accumulateur dans le mot dont l'indice est fourni dans le registre Y.

- Séquence d'appel en assembleur :

```
LY  INDICE
LA  VALEUR
SVC SPAG (49)
```

- Compte-rendu :

1 Requête acceptée

2 Adresse de tableau incorrecte

Le compte-rendu est fourni dans le mot défini par la requête DPAG.

C) LPAG : chargement de l'accumulateur

Cette requête provoque le chargement de l'accumulateur par la valeur du mot dont l'indice est fourni dans le registre Y.

- Séquence d'appel en assembleur :

```
LY  INDICE
SVC LPAG (50)
```

- Compte-rendu :

1 Requête acceptée

2 Adresse de tableau incorrecte

Le compte-rendu est fourni dans le mot défini par la requête DPAG.

3.3.3 REQUETE DE COMMUNICATION DES COMMANDES D'UN PROCESSEUR : CAMO

Un programme peut transmettre au superviseur les commandes associées aux différents modules qui le constituent ainsi que leurs adresses, ceci par une requête programmée faite à une partie du superviseur appelée CAMO (Chargement des Adresses des Modules de programme).

Tous les programmes du software de base ont une telle requête à la fin de la séquence d'instructions correspondant au lancement automatique par le système Ainsi ces programmes, une fois chargés en mémoire, peuvent être connus du superviseur et activés par une commande.

Cette requête, disponible pour tous les programmes des utilisateurs, leur fournit aussi la possibilité d'activer par une simple commande les différents modules qui constituent un programme.

La création de nouveaux modules de programme exploités sous le contrôle du système BOS16 est rendue extrêmement simple par l'utilisation de cette requête

A) Séquence d'appel en assembleur :

L'adresse de la table des commandes est transmise dans A. La forme générale d'une requête CAMO est donc la suivante :

```
LAD TBCMOD
SVC CAMO (10)
```

Le superviseur enregistre l'adresse de la table de noms de modules qui lui est communiquée et se met en attente d'une commande de l'utilisateur. Le superviseur ne rend jamais directement le contrôle au programme ayant provoqué cet appel.

Si BOS16 ne peut plus enregistrer la demande, il y a impression sur l'unité symbolique Listing Log (LL) du message ERB 17 (le système peut enregistrer 5 tables de commandes au maximum).

Il est à noter que la table ainsi communiquée fait toujours partie au programme utilisateur et qu'il peut donc en modifier le contenu en cours d'exécution de façon à valider ou invalider certaines commandes suivant ses besoins (une commande ne pouvant jamais contenir un espace, pour invalider une commande il suffira par exemple d'y placer le code "espace" dans le premier octet de son nom).

B) Description de la table des commandes associées au module de programme :

not (0) :		Nombre de commandes associées à ce programme
- - - - - - - - - - - - - - - - - - - -		
Mot (1) :		Nom de la première commande écrite en ASCII sur 2
Mot (2) :		mots
Mot (3) :		Adresse de lancement du module associé à cette commande
- - - - - - - - - - - - - - - - - - - -		
Mot 3n+1 :		Nom de la (n+1)e commande écrite en ASCII sur 2
Mot 3n+2 :		mots
Mot 3n+3 :		Adresse de lancement du module associé à cette commande

Nota :

Lorsque BOS16 donne le contrôle à un processeur, suite à une commande de l'opérateur, il le fait sans initialiser les valeurs des bases (ce sont celles du système) et le programme utilisateur est en mode maître ou en mode esclave selon l'état du processeur au moment de l'exécution de la requête CAMO.

Le buffer d'entrée de la commande est alors pointé par la base W et il ne sera plus modifié jusqu'à ce que le système reprenne le contrôle. Une commande peut donc éventuellement être composée de 4 caractères qui sont analysés par BOS16 et de paramètres qui pourront être analysés par les processeurs, le buffer d'entrée ayant été préalablement entièrement traité par BOS16 [élimination des blancs, line feed. flèche haute et arrière). De plus le registre X précise le rang du premier caractère du premier paramètre suivant le nom de la commande.

CI Exemple de programmation : soit un Programme composé de 3 modules dont les adresses de lancement sont PRDEB1, PRDEB2, PRDEB3. Pour associer les commandes "ALAN", "BLAN", "CLAN" à ces trois modules on créera la table suivante :

```
TBCMD1 :    WORD 3 < IL Y A 3 MODULES
            ASCI "ALAN" < COMMANDE ASSOCIEE AU PREMIER MODULE
            WORD PRDEB1 < e DE LANCEMENT DU MODULE
            ASCI "BLAN" < COMMANDE ASSOCIEE AU DEUXIEME MODULE
            WORD PRDEB2 < e DE LANCEMENT DU MODULE
            ASCI "CLAN" < COMMANDE ASSOCIEE AU TROISIEME MODULE
            WORD PRDEB3 < e DE LANCEMENT DU MODULE
```

La séquence de programme permettant de passer ces commandes au superviseur est alors :

```
LAD TBCMD1
SVC CAMO
```

Remarques :

- En général les programmes que l'on veut exploiter sous le contrôle de BOS16 terminent par une SVC CAMO la séquence activée lors du lancement automatique par le système. Cette séquence commence à l'adresse indiquée dans la directive END.

- Généralement tout module activable par une commande de lancement se termine par une requête SVC ABOS redonnant le contrôle au système.

3.3.4 REQUETE DE RETOUR AU SUPERVISEUR : ABOS

Lorsqu'un processeur se termine, il rend le contrôle au système en lui communiquant dans le registre accumulateur un compte-rendu appelé "état de processeur".

Ce compte-rendu pourra être éventuellement réclamé ultérieurement pour analyse par un autre processeur ou par une commande IF.

La forme générale de la requête est la suivante :

```
LAI STATPR
SVC ABOS (12)
```

Le système prend le contrôle, imprime un message et passe en attente d'une commande sur l'unité symbolique Control Command (CC).

Si la requête ABOS est émise par une tâche, elle sera transformée en instruction QUIT (si la tâche est résidente) ou en requête EXIT (si la tâche est non-résidente).

3.3.5 REQUETE DE DEMANDE DE L'ETAT PROCESSEUR : RBOS

Le dernier état processeur transmis au système par une requête SVC ABOS peut être communiqué à un processeur par une simple requête de celui-ci.

Cette requête est toujours composée d'une seule instruction :

SVC RBOS (14)

Cette requête est toujours satisfaite. Le retour est fait en séquence et le registre A contient alors le dernier état processeur transmis au système.

Nota :

Cet état peut être testé par la commande IF pour contrôler le job en cours ou édité par la commande de PRINT.

3.3.6 REQUETE DE TEST DES APPELS ET DES DEFAUTS : TAPS

L'opérateur ne peut en général arrêter l'exécution d'un programme qu'en des points bien particuliers de ce programme qui sont seuls connus de ce dernier.

Par l'intermédiaire de la requête SVC TAPS. BOS16 permet donc à un programme de demander au système si celui-ci veut reprendre le contrôle pour une cause qu'il a enregistrée (constatation d'un défaut périphérique ou d'un appel opérateur).

A) Séquence d'appel en assembleur :

Cette requête ne comporte aucun paramètre. Sa forme est donc toujours réduite à une seule instruction :

SVC TAPS (11)

Si aucun appel opérateur ou défaut périphérique n'est enregistré au moment de la requête, le système rend alors le contrôle en séquence au programme ayant fait la requête. Sinon le système arrête l'exécution du programme et, en prenant le contrôle, imprime sur l'unité symbolique Listing Los (LL) le message :

ERB 13	s'il y a arrêt de l'exécution du programme sur constatation d'un défaut périphérique
ERB 14	s'il y a arrêt de l'exécution du programme sur constatation d'un appel opérateur.

Puis le système passe en attente d'une commande.

L'appel opérateur peut être provoqué à tout moment en appuyant une fois sur la touche "BREAK" du périphérique de dialogue (téléimprimeur ou console de visualisation).

Avant chaque lecture de commande (sur l'unité symbolique CC) le système contrôle l'ensemble des périphériques de l'application et signale tous les défauts enregistrés. Par contre lorsque la requête TAPS est émise par un processeur système ou un programme utilisateur seuls sont contrôlés les périphériques associés, au moment de la requête, à des unités symboliques de production de programmes (SO à U4).

Le but recherché est d'éviter l'arrêt de l'exécution d'un programme à la suite d'un défaut survenant sur un périphérique non utilisé par ce programme.

B) Conseils d'utilisation :

A l'utilisation d'une requête SVC TAPS sera en général associé un point d'entrée dans le programme qui permettra de demander par une simple commande la continuation du programme ainsi arrêté.

Cette commande et ce point d'entrée sont préalablement communiqués au superviseur par l'intermédiaire d'une requête SVC CAMO.

Le programme utilisateur ne fera une requête SVC TAPS qu'en des endroits où son état est stable et après avoir éventuellement sauvegardé un certain nombre d'informations qui lui sont propres et qu'il pourra tester lorsqu'il sera réactivé par une c-de de l'opérateur. On peut par exemple invalider certaines commandes relatives au programme avant de faire une requête SVC TAPS, si l'on ne doit pas pouvoir activer certains processeurs suite à l'arrêt du programme au point où l'on fait la requête.

Tous les programmes du software de base font des requêtes SVC TAPS. Cependant si un programme ne fait pas de SVC TAPS ou bien s'il boucle l'opérateur peut quand même arrêter son exécution en faisant 4 appels opérateurs successifs. Il devra alors prendre certaines précautions (éventuellement un rechargement du programme) avant de demander la réactivation du programme ainsi interrompu.

3.3.7 REQUETE DE DEMANDE DU MOT D'ETAT D' PERIPHERIQUE : RMDEF

La requête RMDEF permet à un processeur ou à une tâche :

- de faire l'acquisition du mot d'état d'une unité physique combiné à deux indicateurs (défaut périphérique et appel opérateur)
- d'effacer les défauts ou appels relatifs à cette unité qui ont été enregistrés par le système mais pas encore traités, par l'intermédiaire d'une requête SVC TAPS. par exemple.

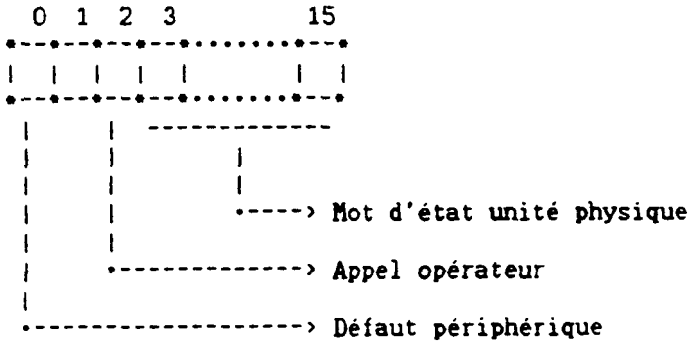
Séquence d'appel en assembleur :

Le numéro d'une unité fonctionnelle ou symbolique est transmis dans Y.
La séquence d'appel est alors la suivante :

```
LY NUMFU ou LY NUMSU
SVC RMDEF (25)
```

Le retour se fait toujours en séquence et l'on trouve dans A le mot d'état de l'unité physique pointée par l'unité fonctionnelle ou symbolique.

Le mot d'état d'une unité physique a la structure suivante :



On trouvera dans le manuel MEMO16 la description des mots d'état des périphériques conventionnels.

3.3.8 REQUETE DE TRANSMISSION D'UN SOUS-PROGRAMME SUPERVISEUR : NEWS

De la même façon qu'un utilisateur peut par une requête SVC CAMO augmenter le nombre de processeurs activables par une simple commande de l'opérateur on peut, par une simple opération de chargement suivie d'une requête spécialisée, ajouter au système un nouveau programme système appelable par une instruction SVC.

L'utilisation de cette requête permettra notamment d'ajouter des bibliothèques de sous-programmes au système sans aucune reconfiguration du système.

La modification apportée au système par une requête NEWS est :

- valable uniquement pour le step en cours si elle est émise par un processeur alimenté dans la zone Background (commandes CALL, RUN et FLOAT)
- permanente si elle est émise par un programme alimenté dans la zone résidente (cf. Commande PRUN : par. 5.1.8 et commande OPTION : par. 5.1.4). Dans ce cas, la requête sera accessible à toutes les taches de l'application.

A) Séquence d'appel en assembleur :

On transmet dans le registre accumulateur A l'adresse du sous-programme et dans Y le numéro de sous-programme superviseur que l'on veut affecter à ce sous-programme. La séquence d'appel est alors la suivante :

```
LA ASPSVC
LYI NOSVC
SVC NEWS (15)
```

Remarque importante : le numéro de la requête transmis dans Y doit être différent des numéros gérés par BOS16 (cf. MEMO16).

Le système enregistre l'adresse du sous-programme superviseur si et uniquement si :

- la requête est faite par un programme exécuté en mode maître,
- le numéro de requête est compris entre 0 et 75, bornes comprises,
- le numéro n'est pas déjà accessible aux tâches de l'application.

Si la requête est acceptée, le superviseur rend le contrôle au programme appelant en séquence. Si elle est refusée, le système ne rend pas le contrôle mais imprime le message :

```
    ERB 20 numéro de la requête
ou  ERB 21 numéro de la requête
```

et passe en attente d'une commande.

La gestion de la zone mémoire dans laquelle est implanté ce nouveau sous-programme superviseur reste entièrement à la charge de l'utilisateur, le système ne pouvant pas la protéger.

B) Comment rédiger un sous-programme superviseur :

Un sous-programme superviseur est un sous-programme qui se déroule toujours en mode maître.

Ce sous-programme est activé à la suite d'une instruction SVC faite par le programme requérant le service et après un traitement préalable standard réalisé par le système qui fait que les conditions d'entrée dans le sous-programme sont les suivantes :

- l'indicateur Carry est positionné à 1 si l'appel vient d'un programme exécuté en mode maître, à 0 si l'appel vient d'un programme exécuté en mode esclave
- le registre X contient le numéro de sous-programme superviseur active
- les bases C et L ont été chargées par des valeurs propres au système et qu'il est inutile de préserver ou restituer
- la base W est chargée par le contenu de l'accumulateur.

Les valeurs des registres C, L, W lors de l'instruction SVC par le programme appelant ont été préservées dans la zone mémoire pointée par le registre K. Elles seront restituées par le système avant le retour effectif à l'utilisateur. La zone pointée par le registre K a donc la signification suivante :

Adresse de retour au programme utilisateur (K-4)
Valeur de la base C du programme utilisateur (K-3)
Valeur de la base L du programme utilisateur (K-2)
Valeur de la base W du programme utilisateur (K-1)
Adresse de retour au programme système (K)

Les sous-programmes superviseur ainsi gérés par le système doivent être des sous-programmes se terminant par une instruction RSR qui rend le contrôle au système. Celui-ci restitue alors à l'utilisateur les valeurs des registres C, L, W et lui rend le contrôle en mode maître ou esclave suivant le mode dans lequel le programme se trouvait lors de l'instruction SVC.

Le système qui sert ainsi d'interface entre un programme utilisateur et un sous-programme superviseur ne modifie jamais (ni dans un sens, ni dans l'autre) les valeurs des registres :

A, B, Y

Le registre X contient toujours le numéro de la requête.

Les conventions de communication entre le programme utilisateur et le sous-programme superviseur sont particulières à chacun des sous-programmes superviseur.

3.3.9 BEQUETE D'ACQUISITION D'INFORMATIONS SYSTEME : TEST

Cette requête est sans paramètre et ne comporte donc que l'instruction suivante :

SVC TEST (16)

Le retour se fait toujours en séquence et l'on trouve dans A la valeur de la base C de BOS16. dans Y la valeur de la base C de IOCS16. dans X la valeur 3 caractérisant BOS16.

Cette requête devra toujours être exécutée par des programmes écrits et mis au point en mode maître.

3.3.10 REQUETE D'ACQUISITION DE LA DATE ET DE L'HEURE : TIME

La date et l'heure sont mises à jour par l'opérateur au moyen de la commande TIME.

- Séquence d'appel en assembleur :

L'adresse d'une table de paramètres est transmise dans A.
La séquence d'appel est alors la suivante :

```
LAD RCB
SVC TIME (53)
```

- Table des paramètres :

0 Adresse de tableau

1 Adresse de compte-rendu

Le tableau donné en paramètre doit comporter 7 mots où seront rangés dans l'ordre : l'année, le mois, le jour, l'heure, minute, seconde et milliseconde.

- Compte-rendu :

Le compte-rendu est fourni simultanément dans l'accumulateur et vaut :

1 Requête acceptée

6 Un paramètre au moins est incorrect

3.3.11 REQUETE D'ACQUISITION DES LIMITES DE LA ZONE DISPONIBLE : FREEM

Cette requête permet d'obtenir les adresses début et fin de la mémoire libre dans les registres A et B.

Ces adresses sont initialisées par BOS16 et mises à jour à chaque chargement de programme en mémoire.

- Séquence d'appel en assembleur :

Cette requête est toujours composée d'une seule instruction :

```
SVC FREEM (51)
```

- Résultats :

En mode maître :

A := adresse première mémoire libre (PREM)
B := adresse dernière mémoire libre (DERN)

En mode esclave :

A := adresse première mémoire libre - SLO
B := SLE - SLO

Remarque :

Cette requête peut être émise par une tâche non résidente. Elle donne alors les limites de la zone disponible en fond de la partition non résidente.

3.3.12 REQUETE DE DEMANDE DE SAUT DE PAGE : PAGE

Cette requête permet d'effectuer un saut de page sur l'unité symbolique LO et d'imprimer un en-tête avec date, nom de job et numéro de page.

La numérotation est réinitialisée à chaque début de "job".

Séquence d'appel en assembleur :

Cette requête est toujours composée d'une seule instruction :

SVC PAGE (52)

Cette requête est ineffective si TS ou ZE est affectée à LO.

3.3.13 REQUETES S'ADRESSANT A IOCS16 : IOCS, WEIO

Pour les requêtes IOCS (8) et WEIO (9), le lecteur se reportera à la notice:

MANUEL DE REFERENCE D'IOCS16

3.3.14 REQUETES S'ADRESSANT A FMS16 : FMS, FMSS, FMSI, FMSD

Pour les requêtes FMS (56), FMSS (57), FMSI (58), FMSD (59), relatives à FMS16, le lecteur se reportera à la notice :

MANUEL DE REFERENCE DE FMS16

Avertissement :

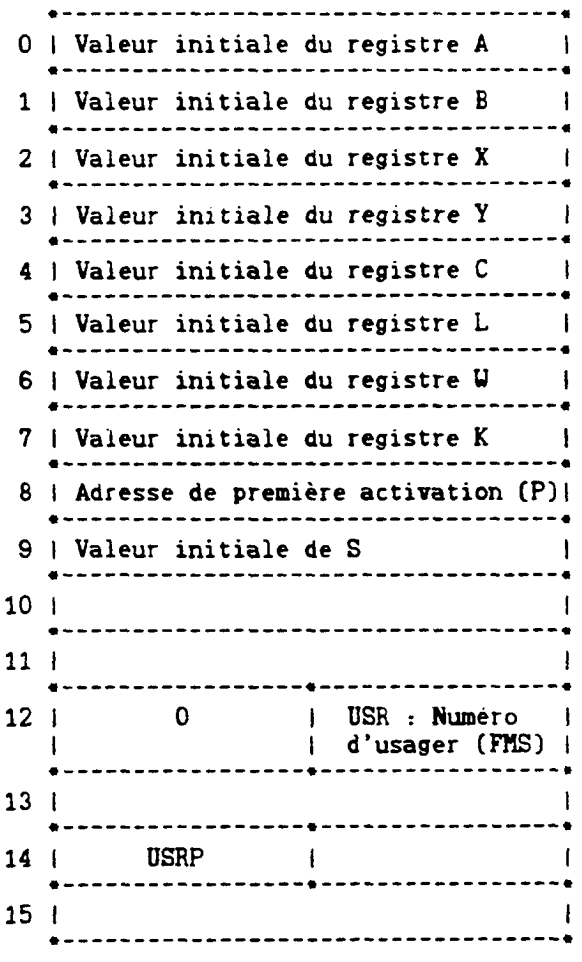
Cette notice de FMS16 ne décrit pas exactement l'interface tel que l'utilisateur le voit mais tel que le système le voit.

En effet le système analyse et complète éventuellement chaque requête à FMS avant de la lui faire exécuter.

En particulier BOS16 transmet la valeur du paramètre USR (numéro d'usager).

Chaque tâche de l'application accédant au système de fichier doit préciser, au moment de son intégration sous BOS16, la valeur du paramètre USR qu'elle utilise; ceci est réalisé par l'intermédiaire du mot (12) de la PST associée à la tâche.

Les 16 mots qui suivent la directive assembleur PSTS. la déclaration de fa tâche PL16 ou FORTRAN sont en effet interprétés de la manière suivante :



Dans le cas d'une PST Hardware seules sont prises en compte les valeurs initiales C, K, P et S.

Le système utilise deux numéros d'usager :

0 fichiers gérés par le système en Background,
1 fichiers supportant les taches non résidentes de l'application.

et le numéro d'usager public (USRP) égal à 0.

Au retour de FMS16. le paramètre de retour de FMS16 est transmis dans l'accumulateur et dans le mot (3) du FCB.

Dans le cas où le numéro d'usager est supérieur ou égal à 128, le contrôle est toujours rendu à l'appelant.

Si le numéro d'usager est inférieur à 128 (c'est en particulier le cas du Background). le contrôle n'est pas rendu à l'utilisateur si :

- un défaut disque a été détecté,
- le compte-rendu est supérieur ou égal à '6020 ce qui indique une erreur logique grave. Il y a alors impression d'un message d'erreur ERB 41 compte-rendu FMS16.

3.3.15 REQUETE D'ACQUISITION DE L'AFFECTATION D'UNE SU : AFSU

Cette requête permet de connaître l'affectation d'une unité symbolique.

- Séquence d'appel en assembleur :

Le numéro de SU est transmis dans A.
Y doit contenir 21.
La séquence d'appel est alors la suivante :

```
LYI 21
LAI NUMSU
SVC AFSU (20)
```

- Résultat :

Le registre A peut contenir les valeurs suivantes :
. '8000 si le numéro de SU est incorrect
. Numéro de FU si une FU est affectée à la SU
. '80+fnum du fichier si un fichier est affecté à la SU.

3.4 DETECTION DES ERREURS

BOS16 détecte un certain nombre d'erreurs et les signale à l'utilisateur en imprimant des messages.

En mode conversationnel toute erreur entraîne la réaffectation des unités symboliques CC et LL respectivement au clavier et à la feuille du téléimprimeur de service ou au clavier et à l'écran de la console de service.

En mode train de travaux (BATCH) toute erreur est fatale et provoque l'abandon du travail en cours (les commandes lues sur l'unité symbolique CC réaffectée à CR sont ignorées jusqu'à la première commande EOJ).

Cette règle comporte cependant deux exceptions : les défauts apparus sur les périphériques et les appels opérateur (un coup ou quatre coups).

Le système réaffecte alors les unités symboliques CC et LL en sauvegardant les affectations antérieures.

Ainsi, au cours d'une compilation PL, après un défaut donnant lieu au message :

```
ERB 13 '0007 '8400
```

l'opérateur peut émettre les commandes suivantes :

```
CPLC
RETURN
```

qui permettent de poursuivre la compilation et de reprendre ensuite le travail en cours.

Il peut également émettre la commande ABORT qui provoque l'abandon du travail en cours.

3.5 UTILISATION DE LA ZONE RESIDENTE

Les commandes OPTION et PRUN qui seront détaillées au chapitre 5 permettent le chargement dans la zone résidente et le lancement :

- d'utilitaires système (flottant programmé par exemple)
- de processeurs système ou utilisateur.

Les modifications apportées au système par les requêtes NEWS émises éventuellement par ces programmes sont alors permanentes et les numéros des sous-programmes superviseur ainsi introduits sont protégés : le système refuse toute requête NEWS émise par le Background lorsqu'elle transmet un numéro géré dans la zone résidente.

De même toutes les commandes communiquées au système, au moyen de requêtes CAMO émises depuis la zone résidente. sont permanentes; elles seront reconnues quel que soit le contexte d'exécution du Background.

Le système est capable d'enregistrer jusqu'à 5 tables de clés, y compris celles qui sont communiquées par le Background.

3.6 CONTEXTE D'EXECUTION DE BOS16

3.6.1 ENCHAINEMENT DES "JOBS"

Les commandes JOB et EOJ permettent de définir le début et la fin d'un travail.

JOB provoque ainsi l'initialisation du contexte lié au travail :

- mise à zéro de toute la zone Background (ZB et ZB2).
- prise en compte des paramètres de la commande (FU, catalogue, temps limite).

Elle ne provoque cependant pas la réaffectation standard des unités symboliques.

La commande EOJ indique la fin d'un travail, ce qui a pour conséquences :

- la fermeture de tous les fichiers permanents, la destruction de tous les fichiers temporaires utilisés par le Background (paramètre USR = 0),
- la réaffectation standard de toutes les unités symboliques, à l'exception des unités US à UF réservées au Foreground.
- la désactivation des sous-programmes superviseur introduits dans la zone Background ainsi que du processeur précédemment en mémoire.

3.6.2 ENCHAINEMENT DES "STEPS"

Un travail peut être décomposé en un certain nombre de steps, chacun d'eux étant caractérisé par une réorganisation de la mémoire.

Les deux commandes d'activation de processeur, CALL et RUN, précisent ainsi un début de step (il en est de même pour les commandes LOAD et FLOAT).

Les opérations que réalise alors le système sont les suivantes :

- la désactivation du processeur (software de base ou utilisateur) appelé en mémoire au cours du step précédent (annulation des clés transmises à BOS16 par requête CAMO. des requêtes transmises par NEWS).
- la réaffectation de certaines unités symboliques lorsque le step débute par la commande CALL.

La réaffectation porte uniquement sur les unités symboliques BI, BO, SI et SO :

- lorsqu'au cours du step précédent un fichier (temporaire ou permanent) était associé à l'unité symbolique BO, ce même fichier est associé à l'unité symbolique BI ; dans le cas contraire l'affectation standard est réalisée pour BI,
- un fichier temporaire système est associé à l'unité symbolique BO.

Le processus précédent est identique en ce qui concerne les unités symboliques SI et SO. Il permet un enchaînement automatique des processeurs travaillant sur des données de même type (binaire ou symbolique). la sortie d'un processeur servant d'entrée au processeur suivant.

Ce point sera repris et détaillé au paragraphe 3.7.6.

Exemples :

- Enchaînement Assembleur, Editeur de liens, Chargeur disque (données de type binaire)
- Enchaînement Editeur de texte, Macro-processeur, Assembleur (données de type symbolique).

Remarque :

Toute affectation parmi les quatre que réalise le système à chaque step reste valable jusqu'à la prochaine commande d'affectation portant sur la même unité symbolique. Il est ainsi possible à tout moment de modifier l'affectation implicite.

3.7 REMARQUES SUR L'UTILISATION DE FMS16

Ce paragraphe suppose acquises les différentes notions introduites dans le manuel de référence de FMS16.

3.7.1 EMPLOI D'UNE COMMANDE COMPORTANT UN FNUM

L'ouverture et la création d'un fichier sont réalisées par le jeu de primitives adressées à FMS16 et comportant, entre autres paramètres, le nom du fichier ainsi que le numéro d'utilisation qui lui est associé. Il s'agit du fnum.

Par la suite toute primitive relative à ce fichier sera référencée à l'aide de ce numéro. Elle ne comportera plus le nom du fichier.

Le numéro d'utilisation d'un fichier reste valable tant que l'accès au fichier est autorisé par FMS16. c'est-a-dire jusqu'à la première des primitives CLOSE, DELET, EOJ (les deux premières primitives devant alors comporter ce même numéro d'utilisation).

L'ouverture ou la création d'un fichier, et par conséquent la définition du numéro d'utilisation du fichier, peuvent être réalisées par requête adressée directement à FMS16 ou par l'une des commandes CREATE, OPEN OLD et OPEN NEW. Dans ces derniers cas les fnum spécifiés par commande doivent être inférieurs à 64.

Par la suite ce fichier sera accessible indifféremment par requête ou par commande à la condition que son numéro d'utilisation soit rappelé. C'est le cas des commandes CLOSE, DELET. CATAL. RENAM et REWIND.

Exemple :

```
/JOB DUPONT,C3,D4
/OPEN OLD 2,FIC
/RENAM 2,FIC1
/CLOS 2
/EOJ
```

(changement du nom du fichier FIC-C3 supporté par D4).

Les commandes ci-dessus permettent la réalisation d'un certain nombre d'opérations sur fichiers telles que création, ouverture, destruction, catalogage, changement de nom, etc...

Elles permettent, en particulier, de remédier à un certain nombre d'erreurs de programmation lors de la mise au point de processeurs utilisateur.

3.7.2 CARACTERISTIQUES DES FICHIERS OUVERTS OU CREES PAR BOS16

Tous les fichiers permanents créés par BOS16 (au moyen de la commande CREATE ou d'une commande "SU fichier inexistant") sont partageables simultanément avec autorisation d'écriture.

Lorsque l'utilisateur demande au système l'ouverture d'un fichier (au moyen de la commande OPEN OLD ou d'une commande "SU fichier existant") une autorisation d'écriture est toujours demandée ce qui a pour effet d'interdire tout autre accès à ce fichier.

Ainsi après une commande :

```
OPEN OLD 4,FIC-PW,D3
```

la commande :

SI FIC-PW,D3

provoque l'émission du message d'erreur

ERB 07 '601E

(fichier occupé)

Ces restrictions ne concernent bien entendu que les commandes citées. L'utilisateur est libre, au niveau de ses programmes, de manipuler par requêtes adressées à FMS16 n'importe quel type de fichier et d'en fixer lui-même les conditions d'exploitation.

3.7.3 NOMBRE DE FICHIERS OUVERTS SIMULTANEMENT SOUS BOS16

Les tables de FMS16 dans le système BOS16 sont configurées de manière à permettre à un instant donné, l'accès à un maximum de 20 fichiers (dont le fichier support de BOS16). Ce nombre peut être modifié au moment de la configuration du système (cf. Commande NPAV : par. 5.1.3).

L'utilisateur peut demander une unité d'accès par une commande CREATE, OPEN OLD. OPEN NEW ou par une commande d'association "SU fichier".

De son coté le système est amené à utiliser un certain nombre d'accès dans les cas suivants :

- lors de chaque step débutant par une commande CALL deux fichiers temporaires séquentiels sont associés respectivement aux unités symboliques BO et SO
- le chargeur disque nécessite. pour élaborer une image mémoire, deux ou même trois (dans le cas d'une adresse aval) fichiers temporaires qui sont détruits en fin de step
- l'assembleur gère la table des symboles relative au programme traité au moyen d'un fichier temporaire direct
- les compilateurs nécessitent l'accès à deux fichiers temporaires.

Lorsque, ne disposant plus d'aucune ressource, FMS16 est incapable d'honorer une demande d'unité d'accès il donne le compte-rendu '6020 ce qui conduit à l'émission, par le système, d'un message d'erreur comportant ce numéro d'erreur logique en information. L'action à entreprendre par l'utilisateur est alors la suppression d'un certain nombre d'unités d'accès par le jeu de commandes CLOSE, DELET ou EOJ, cette dernière provoquant la destruction de tous les fichiers temporaires ouverts à ce moment-la.

3.7.4 COMMUTATION D'UN APPEL A IOCS16 EN UN APPEL A FMS16

Les processeurs système réalisent leurs opérations d'entrée-sortie au moyen de requêtes adressées à IOCS16. Ces requêtes portent sur des unités symboliques.

Ainsi l'assembleur lit le texte symbolique source sur l'unité symbolique SI et génère le programme binaire objet sur l'unité symbolique BO.

A chaque step débutant par une commande CALL le système associe deux fichiers temporaires respectivement aux unités symboliques BO et SO.

De même l'utilisateur peut associer par commande un fichier Permanent ou temporaire à une unité symbolique.

Toute requête adressée à IOCS16 portant sur une unité symbolique à laquelle est associé un fichier est transformé par BOS16 en une requête adressés à FMS16 (il s'agit d'une requête FMSS).

Dans ce cas toute erreur (hardware ou logique) détectée par FMS16 provoque l'émission du message d'erreur

ERB 40 paramètre de retour transmis par FMS16

Remarque :

Lorsque dans un processeur utilisateur une requête adressée à IOCS16 porte sur une SU à laquelle est susceptible d'être associé un fichier. la profondeur nécessaire de la zone pointée par K doit être de 70 mots (30 mots suffisent lorsqu'une FU est associée à la SU).

3.7.5 GESTION DES FICHIERS PAR LE SYSTEME BOS16

Les deux commandes "SU fichier" et "SU *" (associant respectivement à la SU un fichier permanent et un fichier temporaire), ainsi que la commande CALL en ce qui concerne les unités symboliques BO et SO, accordent au système la gestion complète du fichier associé à l'unité symbolique.

Ainsi la fermeture de ce fichier (ou sa destruction dans le cas d'un temporaire non catalogué par l'utilisateur] sont à la charge de BOS16 qui a seul connaissance de son numéro d'utilisation (fnum).

Par contre la commande "SU fnum" n'accorde à BOS16 que l'accès au fichier par l'intermédiaire de l'unité symbolique spécifiée. L'ouverture, la fermeture ou la destruction éventuelle sont la charge de l'utilisateur.

Un certain nombre de règles sont alors appliquées. Elles sont relatives à la commande "SU fichier".

Règle 1 : Le système interdit d'associer à une unité symbolique d'entrée un fichier inexistant, quel que soit le type de ce dernier.

Règle 2 : Lorsque la commande comporte un nom d'article le fichier est de type indexé (séquentiel sinon).

Exemple :

SO ART.FICH

En reprenant l'exemple précédent, si le fichier FICH n'existe pas. il est créé par le système ainsi que l'article de nom ART (SO étant une unité symbolique de sortie).

Si le fichier FICH existe et ne comporte aucun article de nom ART, un tel article est créé par BOS16.

Règle 3 : La commande "SU fichier" positionne le pointeur courant au début du fichier (ou de l'article si la commande comporte un nom d'article).

Remarque :

La commande "SU fnum" ne déplace pas le pointeur courant associé au fichier. Il en est de même de la commande "SU SU'" lorsqu'un fichier est associé à SU'.

Une quatrième règle plus générale peut être énoncée.

Règle 4 : Toute demande d'association à une SU d'un fichier ou d'une FU ferme (ou détruit, dans le cas d'un temporaire) le fichier qui était éventuellement associé à cette SU lorsque les deux conditions suivantes sont réalisées simultanément :

. La gestion du fichier *est* réalisée *par* BOS16 qui en a donc effectué l'ouverture ou la création

. L'unité symbolique spécifiée par la commande est seule utilisatrice du fichier.

Contre exemple 1 :

```
/ OPEN OLD 3,FIC1
/ SO 3
-------
```

/ SO FIC2 ne ferme pas le fichier FIC1 non géré par BOS16.

Contre exemple 2 :

```
/CALL ASM
/LO FIC3
/EL  LO
---------
```

/LO ZE ne ferme pas le fichier FIC3 utilisé également par EL.

3.7.6 OUVERTURE IMPLICITE DE FICHIERS TEMPORAIRES

Les processeurs du software de base, tels que l'assembleur et le chargeur disque, peuvent ouvrir des fichiers temporaires de manière interne. Lorsque le processeur rend le contrôle au système ceux-ci sont alors détruits.

De même la commande CALL peut être amenée à créer des fichiers temporaires. Elle réalise en outre les basculements BO->BI, SO->SI.

Trois cas peuvent alors se présenter, le premier étant le plus fréquent.

Cas 1 : l'unité symbolique BO est associée à un fichier non vide : après la commande CALL le pointeur courant est positionné au début du fichier, ce dernier étant associé à l'unité symbolique BI. Un fichier temporaire est créé. Il est associé à l'unité symbolique BO.

cas 2 : l'unité symbolique BO est associée à un fichier vide : après la commande CALL l'unité symbolique BI prend l'affectation standard. Aucun changement en ce qui concerne l'unité symbolique BO : le même fichier lui reste associé.

Cas 3 : l'unité symbolique BO est associée à une unité fonctionnelle : après la commande CALL l'unité symbolique BI prend l'affectation standard. Un fichier temporaire est créé. Il est associé à l'unité symbolique BO.

Dans les trois cas si un fichier est associé à BI et qu'il a été ouvert par l'utilisateur (au moyen d'une commande OPEN OLD, OPEN NEW ou CREATE), le système ne réalise aucune opération sur ce fichier. S'il s'agit par contre d'un fichier géré par le système, une primitive CLOSE est émise : le fichier est fermé s'il s'agissait d'un permanent, détruit s'il s'agissait d'un temporaire.

Le processus est identique en ce qui concerne les unités symboliques SI et SO.

Remarque importante :

Les commandes :

CALL BUILD et CALL LKLOAD

n'associent aucun fichier aux unités symboliques BO et SO.

Exemple :

/JOB DUPONT,C2,D3,10
/CALL MACP

. BI ZE
. SI CR (ou TS)
. BO l Un fichier temporaire est créé pour BO
. SO l Un fichier temporaire est créé pour SO

/SO SYMOUT

. Le fichier temporaire associé à SO est détruit
. SO est associé au fichier SYMOUT gui est créé, s'il n'existe pas encore

. SO reçoit le symbolique objet réalisé par MACP

/CALL ASM

. BI ZE car le fichier temporaire associé à BO est resté vide (il est détruit)
. SI est associé à SYMOUT
. BO reste associé au même fichier temporaire (vide)
. SO l Un fichier temporaire est créé pour SO

/OPEN OLD 1,BINOUT
/BO 1

. Le fichier temporaire associé à BO est détruit
. BO est associé à BINOUT

. BO reçoit le binaire objet link-éditable émis par ASM

/CALL EDILE

. Le fichier SYMOUT est fermé
. BI est associé à BINOUT
. SI CR (ou ZE) car le fichier temporaire associé à SO est resté vide (il est détruit)
. BO l Un fichier temporaire est créé pour BO
. SO reste associé au même fichier temporaire (vide)

. BO reçoit le binaire objet translatable émis par EDILE

/CALL BUILD

. BI est associé au fichier précédemment associé à BO
. SI CR (ou ZE) car le fichier temporaire associé à SO est resté vide [il est détruit)
. BO ZE
. SO ZE
. Le fichier BINOUT est à la charge de l'utilisateur

/EOJ

- Tous les fichiers sont fermés (Permanents) ou détruits (temporaires).

3.7.7 SATURATION D'UNE FU DISQUE GEREE PAR FMS16

Il existe des cas normaux de saturation d'une FU disque gérée par FMS16. Tous les granules sont alors occupés par des fichiers permanents et, éventuellement, par des fichiers temporaires ayant été créés dans le job en cours.

Deux cas peuvent alors de présenter :

- Un certain nombre de fichiers permanents peuvent être détruits ; il redevient donc possible de travailler sur cette FU.

- La FU disque a été sous-dimensionnée; la génération doit être reprise complètement. Le montage de volume étant obligatoire sous BOS16 une restructuration du disque est suffisante dans la plupart des cas.

Cependant toute fausse manipulation qui conduit à l'impossibilité dans un job d'émettre la commande EOJ entraîne la perte des granules occupés à ce moment-la par des fichiers temporaires.

Ces granules seront récupérables par l'intermédiaire de la commande FUCLEAN du processeur FUP4.

Un certain nombre de conseils peuvent être donnés à l'utilisateur :

. Eviter de saturer la FU disque D2 en travaillant dans une autre FU (par l'intermédiaire de la commande JOB); celle-ci pourra plus facilement être remise à zéro que s'il s'agissait de D2 (à l'aide de la commande FUINI du processeur FUP4).
. Répartir production de programme (nécessitant un certain nombre de fichiers temporaires dont certains ne sont détruits que par la commande EOJ) et mise au point en des jobs distincts.

Exemple :

```
/JOB PRODUC              |
-----------              |  job de
/CATAL IM,PROG           |  production
/EOJ                     |

/JOB ESSAI               |
/RUN PROG                |  job d'essai
/EOJ                     |
```

Dans ces conditions, il n'y aura perte d'aucun granule. La commande RUN n'entraine en effet la création d'aucun fichier temporaire (il en est de même pour la commande LOAD).

3.7.8 CREATION D'UNE BIBLIOTHEQUE

Une bibliothèque est un fichier indexé dont chaque article est un programme binaire link-éditable.

Le nom d'un article est celui d'un symbole défini en tant qu'externe dans le programme constituant l'article (au moyen, par exemple, de la directive ENT en assembleur, du symbole DEF en PL16).

L'utilisateur peut alors commander à EDILE la scrutation d'une telle bibliothèque pour compléter un programme principal. Tout symbole externe auquel il est fait référence est recherché en tant que nom d'article dans la bibliothèque. L'article, considéré comme un MOL, est link-édité au programme principal et apporte la définition du symbole référencé.

Un programme bibliothèque peut faire référence à des symboles externes. Ceux-ci seront recherchés automatiquement par l'éditeur de liens dans la bibliothèque.

Exemple de création d'une bibliothèque :

Soient les trois sous-programmes suivants :

```
I           ENT SINUS
            ---------
            SINUS:PSR L
            ---------
            END

II          SEGMENT PROCEDURE COSIN
            ---------
            END.

III         SEGMENT PROCEDURE TANG
            ---------
            REF PROCEDURE SINUS,COSIN;
            ---------
            END.
```

La création de la bibliothèque BIBLI (fichier BIBLI-BB supporté par D3) peut donner lieu à la suite de commandes suivantes :

```
/JOB BISTOR,BB,D3
/CALL ASM
/SI CR
/BO SINUS.BIBLI
/IASM
/CALL PL
/SI CR
/BO COSIN.BIBLI
/IPLC
/CALL PL
/SI CR
/BO TANG.BIBLI
/IPLC
/EOJ
```

Exemple d'utilisation d'une bibliothèque :

```
/JOB DUPONT,C2,D2
/CALL ASM
/SI CR
/IASM
/CALL EDILE
/ILNK
   ERL 09                Message émis par l'éditeur de liens
/RLNK
  SINUS                  Symboles référencés par le programme principal
  TANG
/OPEN OLD 1,BIBLI-BB,D3
/LLNK 1

                         EDILE va link-éditer les deux articles SINUS et
                         TANG ainsi que l'article COSIN référencé par TANG

/RLNK
                         Tous les symboles sont définis
/ELNK
/CALL BUILD
/SLOD
/CATAL IM,PROG
/EOJ
```

4 UTILISATION DES PROCESSEURS SYSTEME

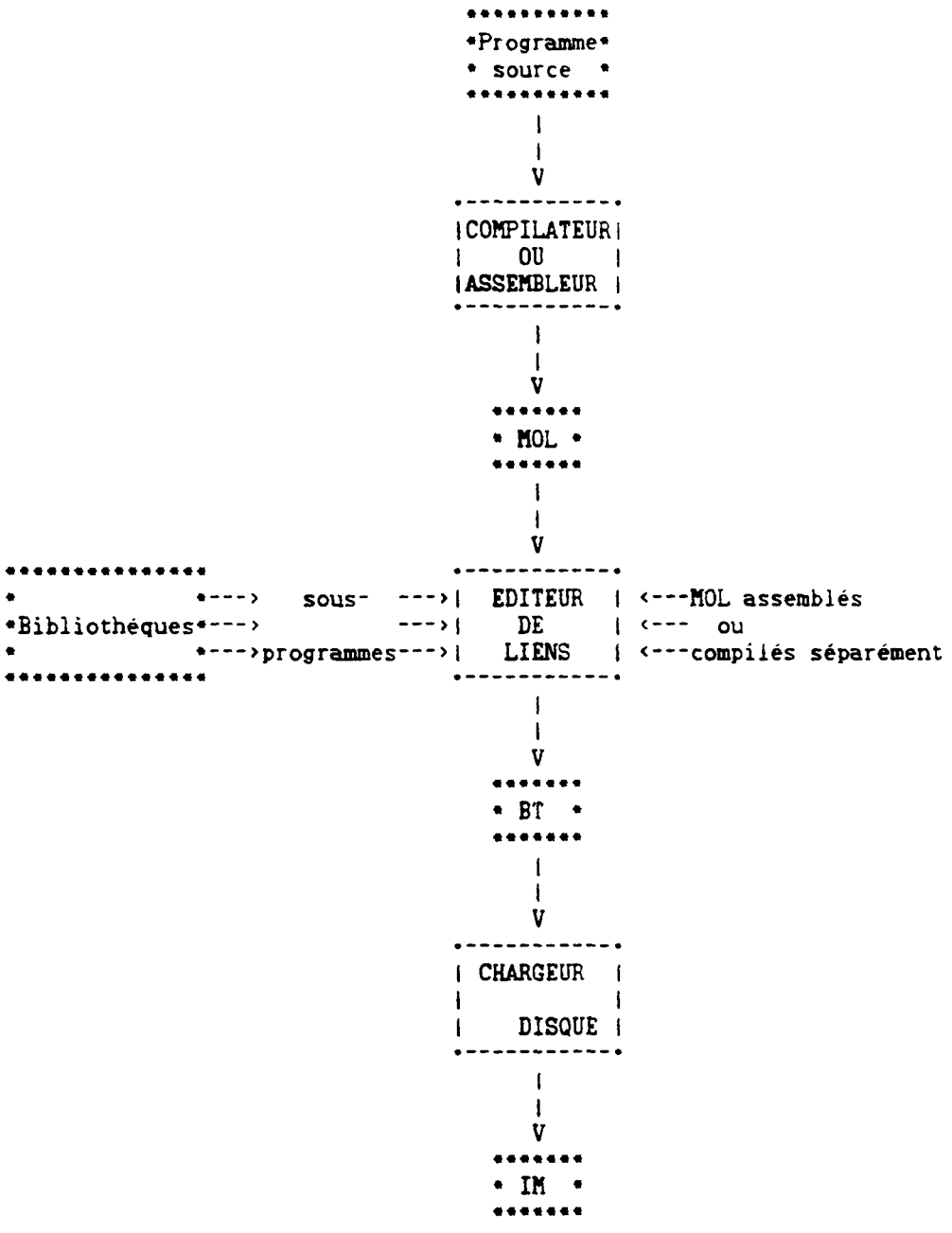
4.1 DESCRIPTION GENERALE

Les compilateurs PL16 et FORTRAN produisent des modules objets link-éditables (MOL).

Le macro-assembleur produit soit des MOL, lorsque le texte source comporte des directives d'assemblage EXT, soit des programmes binaires translatables [BT).

Le format binaire translatable est le seul accepté par le chargeur disque.

La fonction essentielle de l'éditeur de liens EDILE est la génération, à partir de MOL issus d'assemblages ou de compilations séparées, de BT transformes ensuite par le chargeur disque en images mémoires (IM).



La commande RUN permet alors le chargement en mémoire d'une IM et son activation.

Remarques :

- Le processeur LKLOAD permet de réaliser en un seul passage les opérations d'édition de liens et de création d'image mémoire. avec de grandes performances.

- Pour plus de détails concernant EDILE, BUILD et LKLOAD se reporter au manuel de référence "Chaîne de production de programme".

Trois commandes permettent le chargement en mémoire d'un processeur utilisateur (ayant pour support un fichier indexé obtenu par l'intermédiaire du chargeur disque). Ce sont RUN. LOAD) et START.

Le fichier réside sur la FU disque spécifiée par la commande JOB.

Forme de la commande RUN :

RUN [nomproc][,adresse de lancement] "cr"

Pour un programme exécutable en mode maître l'adresse d'implantation est précisée dans le mot (0) du descripteur (article DESC).

Pour un programme exécutable en mode esclave l'implantation est effectuée en début de la zone Background à la première adresse multiple de 16 (zone ZB2).

L'exécution démarre à l'adresse spécifiée par la commande ou par défaut à l'adresse contenue dans le mot (11 du descripteur.

En l'absence simultanée d'adresse de lancement dans la commande et dans le descripteur, le processeur n'est pas activé.

Forme de la commande LOAD :

LOAD [nomproc] "cr"

En ce qui concerne l'adresse d'implantation du programme, les règles sont identiques à celles de la commande RUN.

De la même manière que RUN, LOAD permet le chargement en mémoire du processeur spécifié par la commande ou par défaut de l'image mémoire produite au step précédent par le chargeur disque.

La différence avec RUN provient du fait qu'une fois implanté en mémoire, le programme n'est jamais activé.

Les deux commandes RUN et LOAD possèdent d'autre part une forme étendue permettant la mise en oeuvre d'un ou des deux processeurs utilitaires suivants :

- Module DRIP16 associé au compilateur PL16,
- Flottant programmé.

Forme étendue des commandes :

RUN [nomproc],[adresse de lancement],liste d'utilitaires "cr"
LOAD [nomproc],liste d'utilitaires "cr"

La liste des utilitaires à mettre en oeuvre est constituée d'un ou, séparés par une virgule, des deux noms suivants :

DRIP16, FLOAT

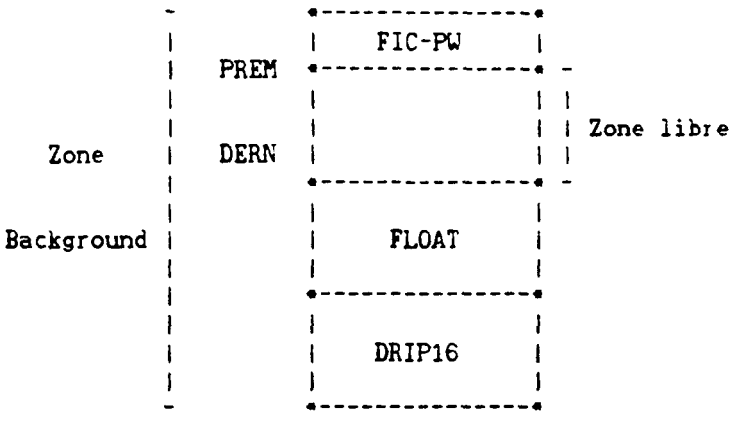
Les processeurs utilitaires s'implantent en fond de la zone Background ZB.

Ainsi pour un programme maître nécessitant pour se dérouler la présence du module DRIP16 et du flottant programmé, la mise en oeuvre peut être la suivante :

RUN FIC-PW,'1000,FLOAT,DRIP16

L'image mémoire est alors supportée par le fichier FIC ayant pour catalogue PW.

L'exécution démarre en '1000. une fois chargés en mémoire, conformément au schéma suivant, le flottant programmé et le module DRIP16.



Le flottant programmé, mis en oeuvre par une commande LOAD ou RUN, n'est résident en mémoire que pour la durée d'un step, c'est-a-dire jusqu'à la première des quatre commandes CALL. LOAD, RUN et EOJ.

Il est cependant possible d'intégrer le flottant au moniteur BOS16 pour la durée d'un job par l'intermédiaire de la commande FLOAT dont la forme est la suivante :

FLOAT "cr"

Ainsi lorsqu'un même job comporte plusieurs appels du flottant programmé, il est plus commode de les remplacer par un appel unique en début de job.

D'autre part cette commande doit être utilisée lorsqu'un compilateur nécessite en mémoire la présence du flottant.

Exemple :

```
FLOAT              Alimentation du flottant
CALL FORTRA,D3 Mise en oeuvre du compilateur
                   FORTRAN (qui se trouve dans
                   la FU disque D3)
```

De plus, lors de la configuration de BOS16. la commande OPTION (cf. par. 5.1.4) permet d'alimenter le flottant dans la zone résidente et de l'intégrer de manière permanente au moniteur.

La commande

START "cr"

permet de lancer un programme préalablement chargé en mémoire par la commande LOAD. On pourra en particulier initialiser l'utilitaire DRIP16 entre le chargement et le lancement du programme.

Exemple :

```
• LOAD FIC-PW,DRIP16
• ONTR          Commandes d'initialisation de DRIP16
• DEGRE,5
• START
```

4.3 CONTEXTE DE LANCEMENT DES PROCESSEURS

Lors de leur activation par BOS16. les processeurs disposent du contexte suivant :

- Registre A :
  - Le bit 0 est à 1 en mode train de travaux (BATCH)
  - Le bit 1 est à 1 si l'unité symbolique LO n'est pas associée à LP
  - L'octet droit indique le fnum utilisé par BOS16 pour charger le processeur.

- Le registre B contient le numéro de FU disque sur laquelle on travaille (paramètre donné dans la commande JOB).

- Le registre W donne l'adresse du buffer qui contient la commande d'activation du processeur.

- Le registre X est l'indice du premier caractère du premier paramètre suivant la clé d'activation dans ce buffer.

- Le registre Y indique si une FU ou un fichier est affecté aux unités symboliques BO (octet gauche) et BI (octet droit). Chaque octet a la signification suivante :
  0 <= octet < '7E : numéro de FU
  '80 <= octet <= 'FF : '80+fnum de fichier disque (FMS16).

- Les registres C, L, K sont propres au système et doivent être réinitialisés.

- Les registres SLO et SLE ne doivent pas être modifiés.

5 EXPLOITATION TEMPS REEL

5.1 COMMANDES DE CONTROLE

5.1.1 GENERALITES

Les différentes commandes de contrôle reconnues par BOS16 peuvent être réparties en 3 catégories suivant l'opération demandée :

- configuration du système
- production et aide à la mise au point de programmes
- contrôle de l'application temps rée1.

Les commandes appartenant à la première catégorie réalisent les fonctions suivantes :

- configuration des tables du système
- intégration à BOS16 de modules optionnels.

Le contrôle de l'application est effectué par des commandes d'intervention on-line permettant en particulier :

- l'intégration de tâches
- l'activation de tâches
- l'initialisation de l'horloge temps réel (cf. TIME : par. 3.2.24)
- l'impression de la carte mémoire
- la visualisation d'une zone mémoire (cf. PRINT : par. 3.2.20)
- la modification de la mémoire
- l'affectation de périphériques aux unités symboliques réservées au Foreground (cf. Commandes d'affectation : par. 3.2.2).

5.1.2 PASS : DEFINITION D'UN MOT DE PASSE

Un certain nombre de commandes sont “contrôlées” par le système car elles peuvent présenter un danger pour l'application.

C'est le cas des commandes INIT et GSYS, ainsi que de toutes les commandes introduites dans ce chapitre, exception faite pour la commande SYST.

La commande PASS permet alors de définir un mot de passe qu'il sera ensuite nécessaire de fournir pour être habilité par le système à émettre les commandes contrôlées.

Sa forme est la suivante :

PASS nom “cr"

Le système n'assure aucune protection après l'émission d'une commande PASS jusqu'à la fin du travail en cours signalée par EOJ.

A partir de ce moment les commandes contrôlées ne sont autorisées que lorsqu'elles apparaissent à l'intérieur de jobs dont le nom est le mot de passe.

Il est toujours possible de redéfinir un mot de passe à condition de se placer dans un contexte en accord avec le mot de passe précédent.

Remarque :

Lorsqu'aucun mot de passe n'est défini au cours de la configuration, le système n'assure aucun contrôle des commandes.

5.1.3 NPAV : DEFINITION DE L'ESPACE DE TRAVAIL DE FMS16

L'espace de travail de FMS16 est constitué de 4 zones :
ZUEP, ZIOCB, ZWCB et ZDF.

Chaque zone est un ensemble de pavés de tailles égales, consécutifs en mémoire et gérés dynamiquement.

ZUEP : Zone d'unités d'Enregistrements Physiques
Chaque pavé a la taille d'un secteur disque (128 mots) et sert de buffer à FMS16 pour la réalisation de certaines primitives.

ZIOCB : Zone des Input-Output Control Blocks
Chaque pavé a une taille de 10 mots.
Un pavé reçoit les informations constituant un IOCB au moyen duquel FMS16 demande la réalisation d'une opération d'entrée-sortie à IOCS16.

ZUCB : Zone des Working Control Blocks
Chaque pavé a une taille de 65 mots et sert de zone de travail à FMS16 pour réaliser une primitive.

ZDF : Zone des Descripteurs de Fichiers, des File Access Unit, des Tables des Ligatures des Granules et des postes systèmes des fichiers Séquentiel Indexé.
Chaque pavé a une taille de 30 mots et sert à mémoriser le descripteur d'un fichier ouvert, le descripteur d'une unité d'accès ouverte sur un fichier, la table des ligatures des granules d'un fichier à organisation physique direct (OFI=1) ouvert en accès direct rapide ou le poste système d'un fichier séquentiel indexé.

A un instant donné, chaque fichier ouvert nécessite N+1 pavés de la zone DF, N étant le nombre d'accès demandés simultanément au fichier (au moyen de paramètres USR ou fnum différents).

Un fichier SIX nécessite un pavé de la zone DF de plus pour stocker son poste système.
Lorsqu'un fichier à organisation physique direct est en accès direct rapide un pavé de la zone DF permet de mémoriser les ligatures de 14 granules pour une Grande FU à organisation Grand DIsque (GDI) et 28 granules pour les autres FU (les plus grands fichiers à organisation physique direct sont de 128 granules, leur TLG en mémoire centrale occupe donc 10 pavés de la zone DF pour une Grande FU et 5 pavés pour les autres).

Les nombres de pavés UEP (NUEP), IOCB (NIOCB) et WCB (NWCB) doivent être tels que :

1 <= NWCB <= nombre de tâches utilisant simultanément FMS16
1 <= NIOCB <= nombre de PU disque supportant des fichiers gérés par FMS16
2 <= NUEP <= 2 x nombre de PU disque supportant des fichiers gérés par FMS16.

La saturation de la zone DF est, pour FMS16, une erreur fatale conduisant au paramètre de retour '6020.

Par contre, l'accès aux pavés UEP, IOCB et UCB est géré par des sémaphores d'exclusion. Un sous-dimensionnement de ces trois zones provoque simplement une diminution du degré de réentrance de FMS16. c'est-&-dire un accroissement de son temps de réponse.

En standard, l'espace de travail de FMS16 est configuré de la manière suivante :

. 2 pavés UEP (NUEP = 2)
. 1 pavé IOCB (NIOCB = 1)
. 1 pavé WCB (NWCB = 1)

. 40 pavés DF (NDF = 40)

Cette configuration est suffisante pour la production de programmes.

L'utilisateur peut cependant dimensionner les diverses zones au moyen de la commande NPAV dont la forme est la suivante :

NPAV NUEP,NIOCB,NWCB,NDF "cr"

Cette commande ne peut être émise qu'avant toute autre commande de configuration du système et ses paramètres doivent respecter les règles suivantes :

1 <= NIOCB, NWCB <= 16
2 <= NUEP <= 16
12 <= NDF <= 55

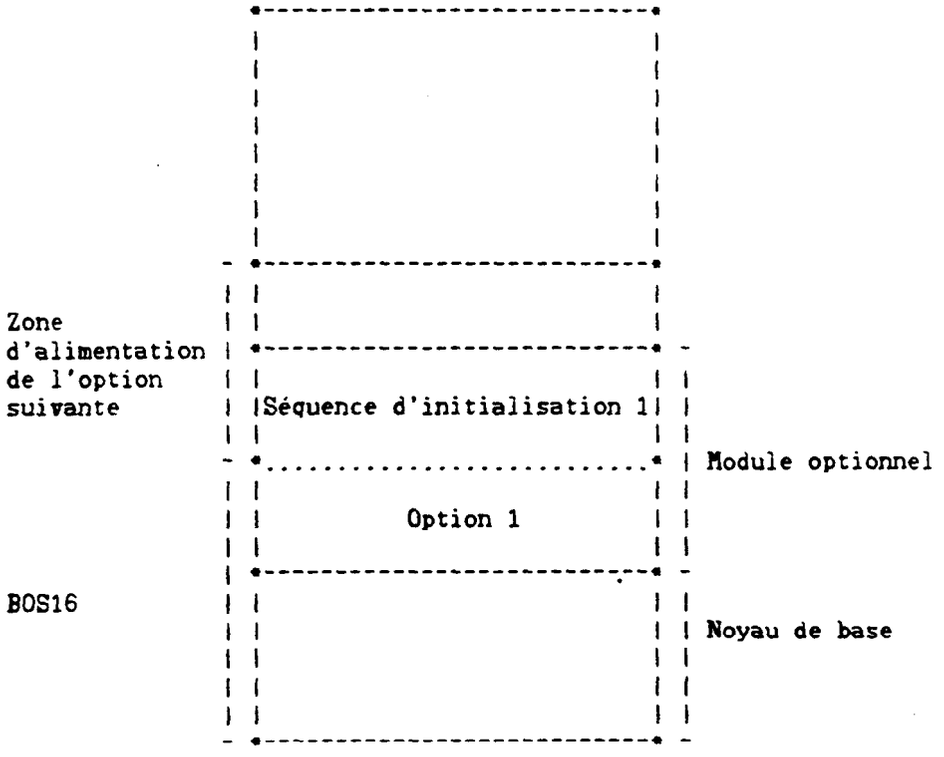
5.1.4 OPTION : INTEGRATION AU SYSTEME DES MODULES OPTIONNELS

La commande OPTION permet d'intégrer au système des modules implantés en format binaire translatable dans la bibliothèque système BSYS16 qui contient en standard les modules suivants :

- Gestion des tâches non résidentes (NRESID)
- Gestion d'événements (EVENT')
- Horloge temps réel (HORL)
- Requêtes Foreground (REQUET)
- Restart automatique (START)
- Flottant programmé (FLOAT et DFLOAT)
- Modification de l'en-tête des listes (TITLE)
- Bufferisation des entrées-sorties (BUFFER)
- Outil de mise au point (DRIP16)
- module d'overlay rapide (FASTOV)
- Lancement automatique d'un fichier de commande (FCDE)
- Affichage au pupitre de la charge de l'unité centrale (VISU).

Chaque module est alimenté en fond de mémoire, immédiatement devant le système; il comporte généralement 2 parties :

- Séquence d'initialisation dont le but est la mise à jour du système (actualisation des tables, communication des requêtes introduites par le module,...)
- Option proprement dite qui. après l'initialisation, fait partie intégrante du système (la place de la séquence d'initialisation est récupérée par le système).



A) Intégration de l'option "Gestion des tâches non résidentes"

Forme :

OPTION NRESID,niv,catg,n,adk "cr"

où :

niv — est le niveau de priorité assigné aux tâches non résidentes de l'application.

Ce niveau peut être compris entre 3 et 43. bornes comprises.
Le système utilise le niveau niv+1, assigné à la tâche PLOAD assurant la gestion des tâches non résidentes.

catg est le catalogue caractérisant les tâches non résidentes de la vacation (2 caractères alphanumériques).
Le système est capable de gérer 128 tâches non résidentes par FU connue de FMS : chaque tâche est supportée par un fichier de nom TNRxyz avec xyz = 000 à 127.
D'autre part une même installation peut supporter plusieurs vacations BOS16, chacune d'elles comportant ses propres tâches non résidentes.

Chaque vacation est alors caractérisée par le catalogue commun à tous ses fichiers TNR xyz.
n est le nombre d'appels que peut cumuler le système :

1 <= n <= 127

Lorsque la zone d'exécution des tâches non résidentes est occupée par une tâche ou que d'autres taches sont en attente de traitement, le système mémorise tout nouvel appel jusqu'à concurrence de n appels.
La mémorisation d'un appel nécessite une zone mémoire de 5 octets.
La saturation des tables du système au moyen d'une requête TCALL ou TCALLW que ne peut honorer BOS16 donne lieu au compte-rendu de requête 7 (système sous-dimensionné).

adk est l'adresse de la zone de swap.
Lorsque la zone non résidente est requise (totalement ou en partie) pour la réalisation d'un travail par le Background, le système met en place un mécanisme de swap.
La zone de sauvegarde sur disque du Background est alors définie par le paramètre adk qui est interprété comme une adresse disque relative à l'unité fonctionnelle DI.
Il est recommandé de prévoir dans DI une zone de swap dont la taille est celle de la zone non résidente.
Lorsqu'une installation supporte plusieurs vacations BOS16, celles-ci peuvent utiliser la même zone de swap.

Les paramètres niv, n et adk doivent être exprimés en décimal.

L'intégration de 1 'option "Gestion des tâches non résidentes" rend accessibles aux tâches de l'application les requêtes TCALL, TCALLW, EXIT, FREEM ainsi pue BCHLR, BACK, BACKR. les tâches non résidentes pouvant avoir une structure d'overlay.

Remarque :

Lorsqu'un processeur utilise la ZNR et que cette zone est requise pour l'exécution d'une tâche non résidente le système effectue la sauvegarde sur disque du Background.

Le swap-out ne peut toutefois être réalisé qu'en dehors des traitements des requêtes effectués en Background.

Toute requête adressée à IOCS16 en provenance du Background est alors analysée par le système, éventuellement transformée, l'opération d'entrée-sortie étant toujours demandée avec retour en fin d'échange.

Ce processus, induit par le mécanisme de swap, n'implique qu'une seule

contrainte par l'utilisateur : tout IOCB utilisé par les processeurs doit avoir une longueur d'au moins 5 mots, le mot (4) étant utilisé par IOCS16 pour positionner un événement fin d'échange.

B) Intégration de l'option "Gestion d'événements"

Forme :

OPTION EVENT,n "cr"

où n représente le nombre d'événements que doit gérer le système (à exprimer en décimal) :

1 <= n <= 128

Une zone de travail de 3 mots par événement ainsi qu'une file de bits, à raison d'un bit par événement, sont réservées pour la gestion des événements.

L'intégration de l'option "Gestion d'événements" rend accessibles aux tâches de l'application les requêtes SEVENT, WEVENT, REVENT et TEVENT.

C) Intégration de l'option "Horloge temps réel"

Forme :

OPTION HORL,n "cr"

L'intégration de cette option rend accessible aux tâches de l'application la requête WAIT dont le but est la suspension de la tâche appelante pendant un délai spécifié.

Le paramètre n indique le nombre maximum de suspensions qu'aura à gérer le système à un instant donné (à exprimer en décimal) :

1 <= n <= 32

Une zone de travail de 3 octets par suspension est réservée pour la gestion de l'horloge.

D) Intégration des requêtes Foreground

Forme :

OPTION REQUET "cr"

Le noyau de base de BOS16 accorde aux tâches de l'application l'utilisation des seules requêtes :

IOCS, WEIO
FMS. FMSS, FMSI, FMSD
RMDEF
EXIT. ABOS

Certaines options intègrent au système de nouveaux sous-programmes superviseur qui sont alors accessibles aux tâches au moyen de requêtes.

L'option REQUET pour sa part introduit :

- TIME. TEST, requêtes introduites au chapitre 3.
- RACT, RUAIT, REDGET, REDPUT, qui seront détaillées au paragraphe 5.2.

E) Intégration de l'option "Restart automatique

Forme :

OPTION START,n,{R|N}"cr"

Lors d'une disparition suivie d'une réapparition du secteur, après une commande INIT ou un redémarrage aux clés de l'application (action sur les clés STOP - INITIALIZE - RUN), la mémoire centrale est réinitialisée au moyen du dernier état de l'application sauvegardé par la commande CON?' (cf.

par. 5.1.11).

Le système peut alors activer une tâche spécialisée dont le but est de relancer les diverses tâches de l'application après avoir effectue un certain nombre d'actualisations spécifiques de chaque application.

Cette tâche est précisée lors de l'intégration de l'option "Restart automatique".

Elle peut être résidente (R) ou non résidente (N).

Dans le premier cas le paramètre n est interprété comme le niveau de priorité software d'une tâche utilisateur résidente de l'application :

3 <= n <= 44
n <> niveau de priorité des tâches non résidentes
n <> niveau de priorité de la tâche système PLOAD

Dans le second cas n est interprété comme le numéro d'une tâche non résidente supportée par un fichier appartenant à l'unité fonctionnelle disque D2 :

0 <= n <= 127

Le paramètre n doit être exprimé en décimal.

F) Alimentation du flottant programmé dans la zone résidente

Forme :

```
OPTION {FLOAT|DFLOAT} "cr"
```

Cette commande permet d'alimenter le flottant programmé simple précision (FLOAT) ou double précision (DFLOAT) dans la zone résidente et de l'intégrer de manière permanente au moniteur.

Après l'alimentation du flottant simple précision, les commandes RUN et LOAD comportant le nom de l'utilitaire FLOAT ainsi que la commande FLOAT. émises en Background, sont rejetées par le système et donnent lieu à l'impression du message d'erreur

ERB 21 '0000

(numéro de requête déjà géré dans la zone résidente).
Le flottant programmé utilise en effet les requêtes de numéros 0 et 1.

G) Modification de l'en-tête des listes

OPTION TITLE,"en-tête" "cr"

La commande PAGE (cf. par. 3.2.25) ainsi que la requête PAGE (cf. par. 3.3.13) provoquent sur le périphérique associé à l'unité symbolique LO un saut de page ainsi que l'impression du nom du job. de la date, du numéro de page et de l'en-tête :

SOLAR 16 : BOS16

L'utilisateur a cependant le moyen, par l'intermédiaire de la commande OPTION TITLE de lui substituer l'en-tête de son choix (40 caractères maximum).

La commande :

OPTION TITLE, " "cr"

provoque la suppression de l'en-tête.

H) Bufferisation des entrées-sorties

OPTION BUFFER.nbuf,nsect "cr"

Cette commande réserve “nbuf” buffers de “nsect” secteurs qui seront alloués à chaque ouverture ou création de fichier exécutée par le système (commandes OPEN, CREATE, CALL, SU fichier).

Cette commande n’est utile que si FMS16 comporte l’option bufferisation (cf. Manuel de Référence FMS16), ce qui augmente grandement les performances du Background.

Les buffers sont alloués dans une zone mémoire située au delà de 64K, la seule contrainte à respecter étant :

```
nbuf* (nsect + 1) < 512
```

I) Intégration de DRIP16

OPTION DRIP16 “cr”

Cette commande alimente de manière permanente l’outil de mise au point DRIP16, qui définit la requête de numéro 7, et un ensemble de commandes.

J) Intégration de l’option “Overlay Rapide”

L’option “Overlay Rapide” réserve des zones mémoire destinées à rendre résidents tout ou partie des tables d’index des fichiers indexés supportant les programmes segmentés. Elle permet ainsi de réduire le nombre d’accès disque nécessaires à chaque chargement de branche. Cette option peut porter sélectivement sur les 3 overlays gérés par BOS16 :

. Overlay système
. Overlay processeur Background
. Overlay des tâches non résidentes.

Forme :

OPTION FASTOV,nbos,nproc,ntnr “cr”

nbos : nombre de secteurs résidents de la table d’index du fichier support de BOS16.
nproc : nombre de secteurs résidents de la TIX des fichiers processeurs.
ntnr : nombre de secteurs résidents de la TIX des fichiers support des tâches non résidentes

nbos = 0 ou 3
0 <= nproc, ntnr <= 4

Remarques :

1 - Les performances d’overlay optimales seront obtenues si, de plus. lors de la création de l’image mémoire, l’utilisateur a émis la commande de FAST du chargeur disque BUILD ou la commande PERFORM de LKLOAD.

2 - Pour l’utilisation des processeurs standard, il est inutile de définir “nproc” différent de 0.

K) Intégration de l'option “Lancement automatique d'un fichier de commande”

Cette option permet de lancer automatiquement un fichier de commande au lancement du système.

OPTION FCDE,nom du fichier de commande,catg “cr”

Contraintes :

- Le fichier de commande devra se trouver sur D2.
- A chaque commande CONF, il faudra refaire l’option FCDE car à la fin de l’exécution du fichier de commande on restaure l’adresse du dialogue.
- Si le fichier de commande fait des “CC Fichier” en cascade ne pas oublier de faire “U1 CC” afin que BOSD ne ferme pas le fichier de commande précédent.

L) Intégration de l’option “Affichage au pupitre de la charge de l’unité centrale”

OPTION VISU “cr”

5.1.5 CONFIGURATION DE LA CARTE DE LA MEMOIRE

Forme :

FORE AZB,AZNR,AZDR,AZR “cr”

où :

AZB est l’adresse d’implantation de la zone Background
AZNR est l’adresse d’implantation, de la zone d’exécution des tâches non résidentes
AZDR est l’adresse d’implantation de la zone de données résidentes
AZR est l’adresse d’implantation de la zone résidente.

La commande FORE permet la configuration initiale de la carte de la mémoire.

Dans ce cas les 4 paramètres exprimés en hexadécimal sont obligatoires et doivent respecter les règles suivantes :

'33 <= AZB <= AZNR <= AZDR <= AZR <= adresse d’implantation de BOS16
' 170 <= AZDR.

Deux adresses identiques impliquent la non-configuration de la zone correspondante.

Ainsi la commande :

FORE ‘33,‘1000,‘1000.‘1800

conduit à une ZNR non configurée.

Par la suite la commande FORE permet de réajuster la carte mémoire.

Dans ce cas certains paramètres peuvent être omis, seules étant précisées les adresses des zones à reconfigurer le rôle de chaque paramètre est alors fonction du nombre de virgules qui le précèdent.

Une reconfiguration partielle de la carte de la mémoire est acceptée à condition de respecter la règle énoncée plus haut et ne pas conduire à une diminution de la taille de la ZR.

La commande de reconfiguration :

FORE ,,‘1800,‘2000

est ainsi rejetée par le système car elle conduit à réduction de la zone résidente qui ne peut être réalisée que dans le cadre d'une reconfiguration totale du système (cf. Commande ZCONF : par. 5.1.12).

Il est impératif de ne réaliser une reconfiguration de la carte mémoire qu'avec une application inerte.

En standard AZB = '33. AZB2 = '8000 et FINZB2 = 'FFFF (adresses début et fin de la zone Background 2). les autres zones n'étant pas configurées.

Remarques :

. La commande DUMP sans paramètre imprime la zone (ZB ou ZB2) où se trouve le dernier programme chargé.

. Dans la commande JOB, le paramètre "taille" n'est pris en compte que si la zone ZNR est de longueur non nulle, et représente toujours un nombre de K mots dans ZB (et non ZB2).

. La commande SYST imprime les limites de ZB2.

. La commande JOB remet à 0 les zones ZB et ZB2.

5.1.6 SYST : IMPRESSION DE LA CARTE DE LA MEMOIRE

Forme :

SYST "cr"

Cette commande permet d'obtenir, sur le périphérique associé à l'unité symbolique EL, les bornes des différentes zones configurées.

Exemple :

ZB : '0033

BOS16 : '1CFA, '7FEF

ZB2 : '8000, 'FFFF

Chaque zone est représentée par son adresse initiale et son adresse finale.

La commande SYST permet, en particulier, de connaître l'encombrement exact du système après l'intégration des options choisies par l'utilisateur.

5.l.7 TLOAD : INTRODUCTION DE TACHES RESIDENTES SOUS BOS16

Forme :

TLOAD nomfic[-catg],FU[,adm] "cr"

où :

nomfic[-catg] est le nom du fichier image mémoire (créé par le chargeur disque) supportant une ou plusieurs tâches utilisateur résidentes

FU désigne l'unité fonctionnelle sur laquelle réside le fichier

adm est l'adresse d'implantation (arrondie par BOS16 au multiple de 16 égal ou supérieur) dans la zone résidente lorsque le programme est exécutable en mode esclave (cette adresse a été précisée dans la commande d'activation du BUILDER pour les programmes exécutables en mode maître); elle doit être exprimée en hexadécimal.

La commande TLOAD charge en mémoire le programme et prend en compte les informations contenues dans le descripteur du fichier image mémoire :

- les PST des diverses tâches qui constituent le programme sont reliées aux tables PSTS et PSTH utilisées par le scheduler microprogrammé
- les files "système" sont mises à jour, chaque tâche software de niveau de priorité i étant déclarée "prête" et "non masquée" :

  RSTFi = ESTFi = 1
  ASTFi = 0

L'activation d'une tâche résidente est réalisée par un programme de l'utilisateur au moyen de l'instruction ARM ou par la commande IRUN de BOS16 qui sera introduite en 5.1.9.

5.1.8 PRUN : ALIMENTATION D'UN PROCESSEUR DANS LA ZONE RESIDENTE

Forme :

```
PRUN nomfic[-catg],FU "cr"
```

où :

nomfic[-catg] est le nom du fichier image mémoire (créé par le chargeur disque) supportant le processeur

FU désigne l'unité fonctionnelle sur laquelle réside le fichier.

Cette commande permet le chargement en mémoire et le lancement d'un processeur utilisateur s'exécutant en mode maître.

L'utilisateur peut introduire ainsi sous BOS16 de nouvelles commandes (à l'aide de la requête CAMO) ou de nouveaux sous-programmes superviseur (à l'aide de la requête NEWS). ces modifications étant permanentes et accessibles par le Background aussi bien que par les tâches de l'application.

5.1.9 IRUN : ACTIVATION D'UNE TACHE

Forme :

```
IRUN R,niveau "cr"
IRUN N,numéro,FU[,param]  "cr"
```

Cette commande permet à l'opérateur d'activer une tâche résidente (R) ou non résidente (N).

Dans le premier cas le niveau spécifié doit correspondre à une tâche que l'utilisateur aura introduite auparavant sous BOS16 au moyen de la commande TLOAD; l'activation de la tache résidente par le système est réalisée au moyen de l'instruction ARR.

Dans le second cas le système commence par vérifier la validité des paramètres de la commande :

numéro est le numéro de la tâche non résidente
0 <= numéro <= 127

FU désigne l'unité fonctionnelle disque sur laquelle réside le fichier supportant la tâche

param est le paramètre d'appel éventuel de la tâche (il est donné sous une forme hexadécimale).

BOS16 vérifie en particulier l'existence d'un fichier image mémoire correspondant à la FU et au numéro spécifiés.

L'activation est alors réalisée au moyen de la requête TCALL (cf. par. 5.2.3).

La tâche n'est cependant pas prioritaire par rapport aux autres taches de l'application; elle est appelée en mémoire et mise en oeuvre après le traitement par le système des appels déjà en attente.

S.I.10 MEMORY : MODIFICATION DE LA MEMOIRE

Forme :

```
MEMORY adm,mask[,{AND|OR|EOR}[,n]]  "cr"
```

Cette commande permet de modifier le contenu d'une mémoire (d'adresse adm) ou de n mémoires consécutives (à partir de l'adresse adm) par intersection (AND), union (OR), disjonction (EOR) de leur contenu avec un masque (mask) ou par le chargement direct du masque en mémoire.

L'adresse et le masque doivent être sous une forme hexadécimale.

Il s'agit d'une commande d'aide à la mise au point des taches de l'application.

La commande MEMORY est une commande qui peut être "contrôlée" : l'utilisateur n'est alors habilité à l'émettre qu'après avoir fourni le mot de passe.

Le système n'effectue aucune protection : toute la mémoire vive du calculateur est accessible.

Exemple :

```
*PRINT  '100,8
'001B
'1F02
'817C
'1000
'9A00
'1E08
'F00C
'2AB2
*MEMO   '100,'0,,2
*MEMO   '102,'7FFF,AND
*MEMO   '103,'7,OR,3
*MEMO   '106,'FFFF,EOR,2
*PRINT  '100,8
'0000
'0000
'017C
'1007
'9A07
'1E0F
'0FF3
'D54D
```

5.1.11 CONF : VALIDATION DE LA CONFIGURATION DU SYSTEME

Forme :

```
CONF [n,adk][,{SU|FU}]  "cr"
```

où :

n est un numéro de vacation
0 <= n <= 7

adk est une adresse de sauvegarde dans l'unité fonctionnelle disque désignée par SU ou FU (D1 par défaut].

Les deux paramètres n et adk doivent être exprimés en décimal.

Cette commande sauvegarde sur disque la portion de mémoire vive débutant à l'adresse AZR et s'étendant jusqu'au fond de la mémoire, c'est-a-dire la zone résidente et le système. Toutes les commandes communiquées au système par SVC CAMO ainsi que toutes les requêtes communiquées par SVC NEWS sont rendues permanentes.

La sauvegarde est réalisée dans l'unité fonctionnelle disque à partir de l'adresse adk. Par la suite la commande INIT avec n en paramètre permettra le rappel en mémoire de la vacation.

En l'absence de paramètres le système prend comme numéro et comme adresse disque ceux de la vacation en cours.

La commande CONF ne doit être utilisée que lorsque l'application est dans un état inerte, en fin de configuration, par exemple, après l'intégration des modules optionnels, des tâches résidentes et des processeurs permanents, ainsi que l'affectation des unités symboliques réservées aux tâches de l'application (U5 à UF).

Remarque :

Lorsque la commande CONF entraîne l'écrasement d'un système déjà implanté dans la FU bootstrap, BOS16 imprime :

ERB 38 No du système

L'utilisateur peut alors, s'il le désire, visualiser les limites des systèmes bootstrapables (commande MAP). éventuellement supprimer un ou plusieurs systèmes (commande DSYS), avant de réémettre la commande CONE.

5.1.12 ZCONF : ANNULATION DE LA CONFIGURATION

Forme :

ZCONF “cr”

Cette commande permet d'alimenter en mémoire une version “vierge” du système et d'annuler les effets des commandes de configuration introduites jusque-là.

Elle affecte également la version du système sauvegardée sur disque dans l'unité fonctionnelle DI.

La commande ZCONF peut être utilisée en cas d'erreur au cours de la configuration ou pour reconfigurer totalement le système.

5.2 REQUETES PROGRAMMEES

5.2.1 GENERALITES

En standard les requêtes suivantes sont accessibles aux tâches de l'application :

IOCS, WEIO, requêtes s'adressant à IOCS16
les requêtes s'adressant à FMS16
RMDEF
EXIT et ABOS (fin de tâche résidente)

L'intégration au système de certaines options leur rend accessibles en plus (ainsi qu'au Background) :

- option "Gestion des tâches non résidentes"
  TCALL, TCALLW
  EXIT (pour les tâches non résidentes)
  BCHLR, BACK, BACKR

- option "Gestion d'événements"
  SEVENT, WEVENT, REVENT, TEVENT

- option "Horloge temps réel"
  WAIT

- option "Requêtes Foreground"
  TIME
  RACT, RWAIT
  REDGET, REDPUT
  TEST

Ces requêtes, de type temps réel, sont présentées dans le langage ASM.
L'utilisateur trouvera au chapitre 6 la syntaxe des appels dans les langages FORTRAN et PL16 (cf. Bibliothèque temps réel : par. 6.2).

5.2.2 COMPTE-RENDU D'UNE REQUETE

La forme générale d'appel d'une requête est la suivante :

```
LAD RPB
SVC  RNUM
```

où RPB (Request Parameters Block) est la table des paramètres de la requête et RNUM est un symbole valeur représentant la requête sollicitée.

Après avoir réalisé le traitement demandé par une requête le système a deux actions possibles :

a) la requête est acceptée ; le travail demandé a été exécuté.

Le compte-rendu de requête dont l'adresse se trouve dans le mot 1 du RPB est égal à 1.

L'appelant reprend le contrôle en séquence après l'appel sauf pour les 4 requêtes BCHLR, BACK, BACKR et EXIT.

b) la requête est refusée par le système; le travail demandé n'a pas été exécuté.

La tâche appelante est alors désarmée, le Background éventuellement interrompu et un message est imprimé sur la feuille du téléimprimeur de service afin d'informer l'opérateur de cette anomalie.

Après la prise en compte de ce message l'opérateur pourra, s'il le désire, relancer la tâche erronée qui se poursuivra en séquence après l'appel. exploitant alors le compte-rendu de requête qui lui est renvoyé.

Remarques :

La relance d'une tâche après une requête refusée ne peut se faire s'il s'agit d'une requête de type temps réel (message d'erreur ERB 83).

Un programme Background ou une tâche non résidente sont définitivement interrompus après une erreur sur requête.

Dans le cas d'un refus de requête le compte-rendu a une valeur supérieure à 1 et, de manière générale, sa signification est la suivante :

2 L'entité spécifiée n'existe pas

3 La requête est illogique

4 L'appelant n'a pas accès à cette requête

5 La requête est incompatible avec l'état actuel de l'entité sur laquelle elle travaille

6 Il existe au moins un paramètre incorrect

7 On ne sait pas enregistrer la requête : le système est sous-dimensionné.

Ce compte-rendu est fourni simultanément dans le mot 1 du RPB et dans l'accumulateur, en cas de relance par l'opérateur au moyen de la commande IRUN après que la requête ait été refusée par le système.

5.2.3 REQUETES D'ACTIVATION D'UNE TACHE NON RESIDENTE : TCALL. TCALLW

Les deux requêtes TCALL et TCALLW permettent d'activer une tâche non résidente de l'application et de lui transmettre un paramètre de travail ; le système utilise l'accumulateur pour fournir à la tâche appelée la valeur de ce paramètre.

Sauf dans les cas d'erreurs sanctionnés par un refus de la requête, à chaque demande d'activation d'une tâche correspond son exécution effective : le système gère le cumul des appels.

Al TCALL : appel sans attente

L'appel réalisé par TCALL est un appel sans attente : BOS16 ne suspend pas l'appelant après l'enregistrement de l'appel.

Au retour chez l'appelant, après la prise en compte de la requête, la table des paramètres est disponible.

- Séquence d'appel en assembleur

La forme générale de la requête TCALL est la suivante :

```
LAD RPB
SVC TCALL (741
```

- Description de la table de paramètres

0 Numéro de la tâche appelée

1 Adresse du compte-rendu

2 Valeur du paramètre de travail

3 Numéro de la FU support

4

à Réservé au système

7

- Compte-rendu

1 Requête acceptée

6 Il existe au moins un paramètre incorrect

7 Système sous-dimensionné : le nombre maximum d'appels stockables est atteint.

B) TCALLW : appel avec attente

La tâche appelante est alors suspendue jusqu'à la fin de l'exécution de la tâche appelée, celle-ci étant signalée au moyen d'une requête EXIT (un compte-rendu du travail demandé peut alors être transmis).

La requête TCALLW est interdite, bien entendu, aux tâches non résidentes de l'application.

- Séquence d'appel en assembleur

La forme générale de la requête TCALLW est la suivante :

```
LAD RBP
SVC TCALLW (75)
```

- Description de la table de paramètres

0 Numéro de la tâche appelée

1 Adresse du compte-rendu

2 Valeur du paramètre de travail

3 Numéro de la FU support

4

à Réservé au système

7

8 Valeur du compte-rendu de traitement

- Compte-rendu

1 Requête acceptée

4 Requête inaccessible aux tâches hardware et aux tâches non résidentes

6 Il existe au moins un paramètre incorrect

7 Système sous-dimensionné : le nombre maximum d'appels stockables est atteint

Remarque :

Pour les requêtes TCALL et TCALLW le système interprète le mot (0) du RPB comme un numéro qu'il transcrit sous la forme de 3 chiffres décimaux : xyz.

Le fichier supportant la tâche non résidente appelée a alors pour nom TNRxyz et pour catalogue celui qui a été défini lors de l'émission de la commande d'intégration au système de l'option "Gestion des tâches non résidentes".

Il réside sur l'unité fonctionnelle disque précisée dans le mot (3) du RPB.

Le système vérifie que le fichier est le support d'une image mémoire (présence des articles ROOT et DESC) comportant une PST.

5.2.4 REQUETE DE FIN D EXECUTION D'UNE TACHE NON RESIDENTE : EXIT

Cette requête permet d'indiquer au système la fin de l'exécution d'une tâche non résidente; celui-ci désarme alors la tache et libère la zone occupée qui devient donc disponible pour une autre tache non résidente ou pour le Background.

De plus lorsque la tâche non résidente a été appelée au moyen de la requête TCALLW, la tâche appelante est réactivée avec transmission d'un compte-rendu de traitement.

Séquence d'appel en assembleur

Le compte-rendu de traitement est transmis dans A :

```
LAI ICRDU
SVC EXIT (23)
```

Dans le cas d'un appel au moyen de la requête TCALL le contenu de l'accumulateur n'est pas exploité par la requête.

Lorsqu'une tâche non résidente est mise au point en Background la requête EXIT est transformée par le système en une requête ABOS.

Inversement une requête ABOS émise par une tâche non résidente est transformée par le système en SVC EXIT.

5.2.5 REQUETES SUR EVENEMENTS : SEVENT, WEVENT, REVENT, TEVENT

Al SEVENT : positionnement d'unévénement

La requête SEVENT permet de mettre un événement dans l'état "SET".

Elle est ineffective si l'événement est déjà positionné.

Cette requête est acceptée sous niveau hardware.

Toutes les tâches en attente de cet événement sont réveillées par le système qui les met dans l'état éligible; elles seront donc traitées même si l'une d'elles (la plus prioritaire ou non) efface l'événement.

- Séquence d'appel en assembleur

La forme générale de la requête SEVENT est la suivante :

```
LAD RPB
SVC SEVENT (27)
```

- Description de la table des paramètres

0 Numéro d'événement

1 Adresse du compte-rendu

- Compte-rendu

1 Requête acceptée

6 Il existe au moins un paramètre incorrect

B) WEVENT : attente sur événement

La requête WEVENT permet d'attendre l'arrivée d'un événement.

L'appelant n'est pas suspendu si l'événement est déjà positionné.

Plusieurs tâches peuvent attendre le même événement.

La requête WEVENT est refusée à une tâche hardware même si l'événement est positionné.

- Séquence d'appel en assembleur

La forme générale de la requête WEVENT est la suivante :

```
LAD RPB
SVC WEVENT (28)
```

- Description de la table de paramètres

0 Numéro d'événement

1 Adresse du compte-rendu

2

à Réservé au système

4

- Compte-rendu

1 Requête acceptée

4 Requête inaccessible aux tâches hardware

6 Il existe au moins un paramètre incorrect

C) REVENT : effacement d'un événement

La requête REVENT permet de mettre un événement dans l'état "RESET".

Elle est ineffective si l'événement est déjà effacé.

Les tâches hardware ont accès à cette requête.

- Séquence d'appel en assembleur

La forme générale de la requête REVENT est la suivante :

```
LAD RPB
SVC REVENT (29)
```

- Description de la table de paramètres

0 Numéro d'événement

1 Adresse du compte-rendu

- Compte-rendu

1 Requête acceptée

6 Il existe au moins un paramètre incorrect

D) TEVENT : test d'un événement

La requête TEVENT permet à une tâche de savoir si un événement est dans l'état "SET" ou RESET", sans être suspendue.

Cette requête est accessible aux tâches hardware.

- Séquence d'appel en assembleur

La forme générale de la requête TEVENT est la suivante :

LAD RPB
SVC TEVENT (30)

- Description de la table de paramètres

  0 Numéro d'événement

  1 Adresse du compte-rendu de requête

  2 Adresse du compte-rendu de traitement (état de l'événement)

- Compte-rendu de requête

  1 Requête acceptée

  6 Il existe au moins un paramètre incorrect

- Compte-rendu de traitement

  1 Etat SET

  2 Etat RESET

5.2.6 REQUETE DE DEMANDE DE SUSPENSION : WAIT

Cette requête permet à une tâche de demander sa suspension pendant un délai exprimé en tops de base.

Le délai maximum est de 65535 tops de base; il peut donc atteindre une durée voisine de 21'50". avec une horloge temps réel de période de 20 millisecondes.

- Séquence d'appel en assembleur

  La forme générale de la requête WAIT est la suivante :

  LAD RPB
  SVC WAIT (31)

- Description de la table de paramètres

  0 Réservé

  1 Adresse du compte-rendu

  2 Délai

  3 Inutilisé

- Compte-rendu

  1 Requête acceptée

  4 Requête inaccessible aux tâches hardware

  6 Il existe au moins un paramètre incorrect

  7 Système sous-dimensionné : le nombre maximum de tâches suspendues est atteint

5.2.7 REQUETES D'UTILISATION DES SEMAPHORES PRIVES : RWAIT, RACT

Afin de résoudre le problème de cumul des appels ainsi que la synchronisation entre tâches, les deux requêtes RWAIT et RACT rendent accessibles aux tâches en mode esclave les instructions sur sémaphore WAIT et ACT (cf. Manuel de présentation du SOLAR 16).

Les sémaphores prives ainsi utilisés doivent être situés dans la zone de données résidentes et peuvent être avec ou sans file de paramètres.

L'initialisation du sémaphore est à la charge de l'usager.

A) RUAIT : mise en attente d'une tâche sur un sémaphore privé

- Séquence d'appel en assembleur

La forme générale de la requête RUAIT est la suivante :

```
LAD RPB
WC  RUAIT  (33)
```

- Description de la table de paramètres

0 Déplacement depuis le début de la ZDR

1 Adresse du compte-rendu

- Compte-rendu

1 Requête acceptée

2 Débordement de la ZDR

4 Requête inaccessible aux tâches hardware

6 Il existe au moins un paramètre incorrect

B) RACT : réactivation d'une tâche en attente sur un sémaphore privé

En cas d'utilisation d'un sémaphore privé paramétré lors de la requête RACT, le registre Y permet de personnaliser l'exécution de la tâche appelée, par le passage d'un paramètre,

- Séquence d'appel en assembleur

Le paramètre éventuel est transmis dans Y.
La forme générale de la requête RACT est donc la suivante :

```
LY  IPARAM
LAD RPB
SVC RACT (32)
```

- Description de la table de paramètres

0 Déplacement depuis le début de la ZDR

1 Adresse du compte-rendu

- Compte-rendu

1 Requête acceptée

2 Débordement de la ZDR

5 Premier mot du sémaphore incorrect

6 Il existe au moins un paramètre incorrect

5.2.8 REQUETES A LA ZONE D'ACCES DE DONNES RESIDENTES : REDGET, REDPUT

La requête REDGET permet de lire une portion de la zone de données résidentes et de la transférer dans un buffer de l'appelant.

La requête REDPUT permet l'opération inverse, c'est-à-dire le transfert d'un buffer propre à la tâche appelante dans une portion de la zone de données résidentes.

Au retour de ces deux requêtes le registre X contient l'adresse de la zone de données résidentes.

La gestion de la ZDR est laissée à la charge de l'usager (protection,

exclusion d'accès,...).

Les deux requêtes REDGET et REDPUT sont acceptées sous niveau hardware.

A) REDGET : lecture de la zone de données résidentes

- Séquence d'appel en assembleur

La forme générale de la requête REDGET est la suivante :

```
LAD RPB
SVC REDGET (36)
```

- Description de la table de paramètres

0 Déplacement depuis le début de la ZDR

1 Adresse du compte-rendu

2 Adresse du buffer

3 Nombre de mots à transmettre

- Compte-rendu

1 Requête acceptée

6 Il existe au moins un paramètre incorrect

B) REDPUT : écriture dans la zone de données résidentes

- Séquence d'appel en assembleur

La forme générale de la requête REDPUT est la suivante :

```
LAD RPB
SVC REDPUT (37)
```

- Description de la table de paramètres

0 Déplacement depuis le début de la ZDR

1 Adresse du compte-rendu

2 Adresse du buffer

3 Nombre de mots à transmettre

- Compte-rendu

1 Requête acceptée

6 Il existe au moins un paramètre incorrect

5.3 DETECTION DES ERREURS

BOS16 détecte un certain nombre d'erreurs et d'alarmes en provenance du Foreground :

a) erreurs sur requêtes

Chaque requête émise par une tâche de l'application donne lieu a deux traitements au niveau du système :

- Un traitement général effectué par le SVC Handler qui contrôle le numéro de la requête, l'adresse éventuelle du RPB ainsi que la valeur du registre K

- Un traitement spécifique de chaque requête pouvant entraîner un refus d'exécution avec les compte-rendus signalés dans les descriptifs des diverses requêtes.

b) erreurs de programmation

Ce sont les 9 erreurs qui provoquent l'activation de la tâche hardware de niveau de priorité 0, et l'impression d'un message d'erreur ERB 00 n

n = 0 Adresse mémoire inexistante
n = l Alarme protection mémoire
n = 2 Imparité mémoire
n = 3 Instruction optionnelle inexistante
n = 4 Instruction privilégiée
n = 5 Instruction RQST ou WAIT sous niveau hardware
n = 6 Instruction IPI
n = 7 Instruction STEP
n = 8 Breakpoint

cl alarmes système

On regroupe sous cette rubrique :

- les défauts des périphériques
- les appels de l'opérateur
- disparition et réapparition du secteur.

5.3.1 ERREUR SUR REQUETES

Les erreurs sur requêtes sont fatales à l'exception de toutes les erreurs détectées par IOCS16 et des erreurs logiques détectées par FMS16 avec un numéro inférieur à '6020 ; elles provoquent :

- la suspension de la tache appelante :

  . ASTFi = 0, i étant le niveau de priorité, dans le cas d'une tâche software résidente

  . émission par le système d'une requête EXIT dans le cas d'une tâche software non résidente

  . masquage dans le cas d'une tâche hardware

- l'interruption du Background, immédiatement ou à la fin de l'opération d'entrée-sortie en cours (effectuée par IOCS16 ou par FMS16)
- la réaffectation standard des unités symboliques CC et LL
- l'impression sur le périphérique associé à l'unité symbolique LL d'un message comportant :

  . le numéro de l'erreur
  . le type et le niveau de priorité de la tâche ainsi que son numéro dans le cas d'une tâche non résidente

. la PST de la tâche au moment de la détection de l'erreur. (Dans le cas d'une alarme provoquée par une tâche hardware, seuls les registres C, K, P, S sont significatifs).

- la remise du moniteur sur une boucle d'inaction ou la relance du dialogue opérateur selon le contexte d'exécution du Background au moment de la détection de l'erreur :

  . après l'émission d'une commande EBOS (cf. par. 3.2.22) toute erreur est signalée par le système qui redevient ensuite inactif (le dialogue sera relancé par un appel opérateur)
  . interrompu en cours de traitement, le dialogue se poursuit à partir du clavier du téléimprimeur (en mode conversationnel) ou du lecteur de cartes (en mode batch, avec abandon du travail en cours).

Après la prise en compte du message d'erreur l'opérateur peut, s'il le désire, relancer une tâche résidente erronée au moyen de la commande IRUN.

Le compte-rendu de requête est transmis par le système à la tâche (dans le mot prévu à cet effet ainsi que dans l'accumulateur) qui doit alors l'exploiter.

5.3.2 ERREURS DE PROGRAMMATION

Toute erreur de programmation détectée par le système au cours de l'exécution d'une tâche de l'application donne lieu au même traitement que dans le cas d'une erreur sur requête (au numéro d'erreur près).

5.3.3 DEFAUTS PERIPHERIQUES ET APPELS OPERATEUR

Le système répartit les différents périphériques d'une installation en deux sous-ensembles, en fonction des unités symboliques qui leur sont associées :

- les périphériques de production de programmes; ils sont associés aux unités SO à U4
- les périphériques du temps réel; ils sont associés aux unités U5 à UF.

Le système étant inactif (à la suite de l'émission d'une commande EBOS), tout défaut survenant sur un périphérique du temps réel est signalé immédiatement, le système étant remis ensuite sur une boucle d'inaction.

En cours de production' de programme, par contre, le système ne contrôle l'ensemble des périphériques de l'application qu'au moment d'une lecture de commande.

Lorsqu'un défaut survient sur un périphérique du. temps réel, au cours de l'exécution d'un processeur, ce dernier n'est donc pas interrompu; le défaut n'est signalé que lorsque le processeur se termine et rend le contrôle au système.

La requête RMDEF permet cependant à une tâche de prendre connaissance et d'effacer tout défaut ou appel relatif à une unité physique; aucun message ne sera alors imprimé par le système.

5.3.4 DEFAUT SECTEUR

Sur un tel défaut, lors de la réapparition du secteur, le système entreprend les actions suivantes :

- envoi à FMS16 d'une primitive EOJ provoquant la fermeture de tous les fichiers permanents et la destruction de tous les fichiers temporaires
- rappel depuis le disque de la version du système sauvegardée sur l'unité fonctionnelle D1
- restitution de l'état des événements dans l'état où se trouvaient ces informations au moment du défaut secteur
- éventuellement activation d'une tâche spécialisée (spécifiée lors de la configuration du système) qui dispose des événements et de la zone de données résidentes pour entreprendre la réinitialisation de l'application
- impression du message :

  ERB 08
- remise du moniteur sur une boucle d'inaction ou relance du dialogue opérateur selon le contexte d'exécution du Background au moment du défaut secteur.

6 UTILISATION DE BOS16

6.1 GENERALITES SUR LES DISQUES

6.1.1 LES ECHANGES

Les échanges au niveau coupleur se font une fois que le bras a été positionné sur un seul cylindre (pas de changement de cylindre), à partir d'une frontière de secteur, sur un nombre quelconque de mots.

Au niveau d'un coupleur on peut multiplexer les déplacements de bras mais, en cours d'échange, il est impossible de faire un échange ou un déplacement sur une autre unité.

6.1.2 RAPPEL DE LA NOTION D'UNITE FONCTIONNELLE DISQUE

On appelle unité fonctionnelle disque, une portion de disque adressable *et* qui est définie par sa taille et l'adresse réelle du premier secteur. Les FU disque constituent autant de disques indépendants.

Ces disques virtuels sont organisés en secteurs de 128 mots et sont adressables par secteur.

6.1.3 RAPPEL CONCERNANT L'UTILISATION D'IOCS16

IOCS16 assure la gestion des FU disque :

- lecture ou écriture d'une zone à partir d'une adresse de secteur,
- vérification des écritures par la lecture de contrôle,
- vérification de la validité des demandes par rapport aux paramètres des FU,
- protection en écriture,
- positionnement du bras sur un cylindre (disque à tête mobile).

De plus, pour les disques à tête mobile :

- toute demande d'échange peut donner lieu à un mouvement,
- un échange peut donner lieu à un mouvement intermédiaire.

Les échanges sont généralement limités à 8K mots (6K mots pour les disques 10 et 20 Moctets).

6.1.4 RAPPEL CONCERNANT L'UTILISATION DE FMS16

FMS16 est un utilisateur de IOCS16 comme les autres. Il travaille sur des FU.

Des fichiers appartenant à des utilisateurs différents peuvent être implantés dans la même FU.

FMS16 réalise, pour un utilisateur, la création, l'utilisation et la gestion *des* fichiers.

De plus, il assure la protection d'un fichier vis à vis des autres

utilisateurs travaillant sur la même FU.

Un utilisateur de FMS16 ne peut donc avoir accès qu'aux FU gérées par FMS16.

Mais en utilisant directement IOCS16, on a accès à toutes les FU. Aucune vérification sur la validité d'accès à une FU par un utilisateur de IOCS16 n'est réalisée.

Il est donc recommandé de n'utiliser que FMS16 pour traiter des informations sur disque (sûreté et facilité d'utilisation).

Remarque :

Le système BOS16 utilise nécessairement une FU disque particulière contenant la version courante du système. Il s'agit de D1. Cette FU n'est gérée que par IOCS16, et commence au secteur 0 du disque support de D1 et D2.

6.1.5 CHARGEMENT D'UNE VACATION PAR LE PROGRAMME DE RAPPEL RAPD

RAPD est un programme implanté dans la FU disque D1. Il permet de rappeler à partir du disque qui le supporte, une parmi 8 vacations possibles.

RAPD doit donc être configuré en fonction des vacations (ou systèmes) à rappeler.

La commande GSYS de BOS16 et la commande SAVE de RTESl6 permettent de communiquer à RAPD les informations concernant les systèmes bootstrapables.

Principe de fonctionnement :

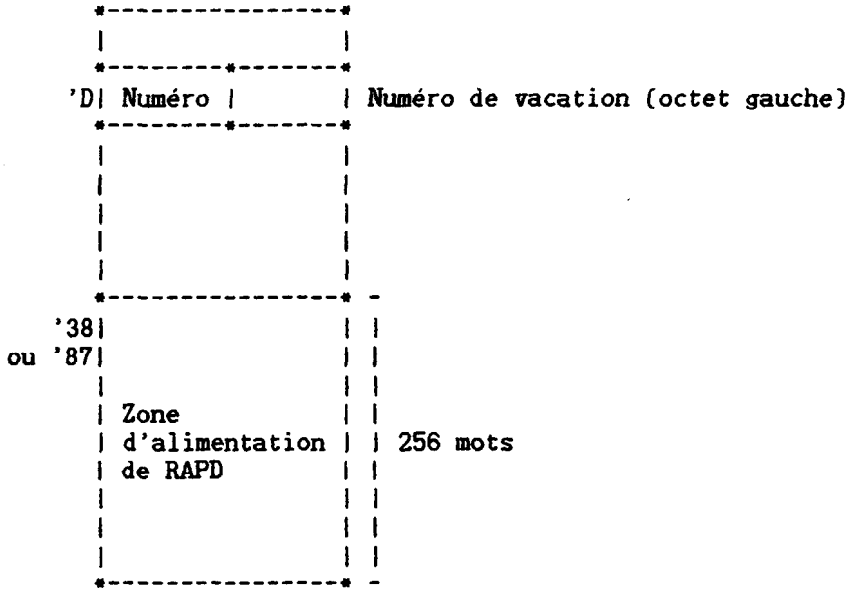
Un redémarrage à froid, après mise hors tension du calculateur, nécessite le rappel en mémoire d'une vacation. Cette opération est réalisée au moyen de la clé LOAD du pupitre, le sélecteur BOOTSTRAP DEVICE étant sur la position MHD, FHD ou FLD.

Le microprogramme charge RAPD en mémoire.

La clé RUN du pupitre permet de lancer RAPD qui appelle en mémoire et lance la vacation de numéro 0 (en général BOS16).

On peut alors éventuellement, par la commande INIT, lancer la vacation de numéro désiré.

Schéma mémoire à la suite du rappel d'un système au moyen du bootstrap ou d'une commande INIT :



'87 est l'adresse de chargement du bootstrap disque souple.

6.1.6 IMPLANTATION DE RAPD SUR DISQUE

Cas de disques souples double face et double densité :

RAPD est implanté sur les secteurs 0 à 3 du disque, par le processeur INFDD.

Cas de disques à têtes mobiles :

RAPD est implanté sur le plateau fixe et sur toutes les cartouches par le programme de formatage sur les secteurs 0 à 3.

Cas des disques à interface SMD :

RAPD est implanté sur les secteurs 0 à 3 par le processeur INSMD (cf. Annexe 3).

Cas des disques Winchester :

RAPD est implanté sur les secteurs 0 à 3 par le processeur INSAS (cf. Annexe 4).

N.B. : En aucun cas on ne devra écraser les secteurs 0 à 4, ni des car touches, ni des plateaux fixes. Il est donc recommandé de ne définir aucune unité fonctionnelle disque, autre que DI et la FU permettant l'accès à l'espace initial, commençant en début de plateau (le secteur 4 contient la description de la structure du disque).

BOS-R est un utilitaire destiné à assurer la sauvegarde et la restitution des disques.

Le support utilisable pour cette opération est soit la bande magnétique, soit la cartouche, soit la cassette, soit la disquette (5"1/4 haute densité) en fonction des configurations.

BOS-R assure ainsi :

- pour tout type de disque, sa sauvegarde/restitution en utilisant une ou plusieurs bandes magnétiques,
- pour les disques comportant deux ensembles (disque fixe et cartouche) la sauvegarde/restitution des parties fixes en utilisant une ou plusieurs cartouches,
- pour les disques Winchester avec streamer (DWB), la sauvegarde/restitution en utilisant une ou deux cassettes,
- pour les disques Winchester DWF20 la sauvegarde/restitution en utilisant une ou plusieurs disquettes.

Le traitement d'un disque, lors d'une sauvegarde ou d'une restitution n'utilisant pas le support cassette, est optimisé. En effet BOS-R n'effectue aucune opération d'entrée-sortie sur les parties des espaces gérés par FMS16 ne comportant pas de données. De plus l'utilisateur peut préciser quels sont les espaces non significatifs du disque, ce qui permet d'exclure de la sauvegarde les espaces vides ou contenant des données facilement reconstituables.

BOS-R permet de sauvegarder tout disque, y compris le disque système, du fait qu'il est entièrement résident.

Dans le cas du disque système, les supports utilisés pour la sauvegarde sont bootstrapables. Ils comportent en effet un système BOS-R, chargé en mémoire et activé par un bootstrap, sous le contrôle duquel s'effectue la restitution. Ceci impose pour les disques durs DWF20 une limitation de la taille de D1 à 32 cylindres, soit 4160 secteurs.

A l'exception des espaces non significatifs les disques sont sauvegardés et restitués en totalité. Cependant les secteurs 20 à 66 de la piste 0 des disques à interface SMD (CDD 27/55/83 et RDD 300) ne sont pas traités. Ils contiennent en effet la zone de gestion des statistiques d'erreurs de ces types de disques.

Sauvegarde et restitution des disques Winchester sur cassette :

BOS-R utilise les cassettes en "streaming mode", ce qui permet d'obtenir de bonnes performances en sauvegarde/restitution.

Ce mode de fonctionnement interdit à BOS-R d'optimiser la sauvegarde en ne tenant pas compte des parties inutilisées des espaces traités.

Dans le cas des disques Winchester BOS-R assure :

- la sauvegarde/restitution d'un espace appartenant à un disque de données, cet espace étant, ou non, géré par FMS16

- la sauvegarde du disque supportant le système en utilisant deux cassettes : les espaces systèmes d'une part (D1 et D2), les espaces données, d'autre part. La partie système est restituée par le bootstrap (fonction KD), la partie données étant traitée par BOS-R.

Remarque concernant les disquettes :

Ce support de sauvegarde doit avoir été initialisé (commande INVOL du processeur INSAS).

6.2.1 MISE EN OEUVRE DE BOS-R

BOS-R est un utilitaire autonome auquel on donne le contrôle depuis un système disque par la commande INIT de changement de vacation.

Remarque :

BOS-R est construit à partir du moniteur BOS-B qui offre à l'utilisateur la possibilité d'interrompre un traitement en cours par action sur la touche "BREAK" du périphérique de dialogue.

6.2.2 SAUVEGARDE D'UN DISQUE

La sauvegarde d'un disque peut être demandée par l'utilisateur au moyen de la commande SAVE.

Lorsque le disque comporte des espaces non significatifs, l'utilisateur peut les communiquer à BOS-R au moyen de la commande NOSP. qui doit alors précéder la commande SAVE (la commande NOSP n'est pas prise en compte pour la sauvegarde des disques Winchester sur cassette).

SAVE permet de sauvegarder tout type de disque sur un support bande magnétique, les plateaux fixes des disques comportant deux ensembles sur un support cartouche. les disques Winchester sur cassette (DWB) ou sur disquette (DWF20).

Le nombre de bandes, de cassettes, de cartouches ou de disquettes à utiliser dépend de la capacité du disque à sauvegarder et de la quantité d'information supportée par le disque. En effet :

- la sauvegarde ne doit concerner que les espaces significatifs du disque,
- tout espace significatif non géré par FMS16 est sauvegardé intégralement,
- tout espace géré par FMS16 comporte une table d'allocation de granules qui est exploitée par BOS-R pour ne sauvegarder que les granules occupés [exception faite pour la sauvegarde des espaces des disques Winchester sur cassette qui est intégrale).

Lorsque la sauvegarde nécessite plusieurs supports --bandes magnétiques, cassettes, cartouches ou disquettes- ceux-ci sont numérotés. Dans le cas de la sauvegarde du disque système le premier support obtenu contient un utilitaire BOS-R bootstrapable sous le contrôle duquel se déroulera la restitution.

Les bandes magnétiques sont contrôlées systématiquement après écriture. Cette opération est réalisée par une lecture arrière de la bande et une comparaison avec le contenu du disque.

Forme :

```
[NOSP,liste d'espaces "cr"]

SAVE,"texte"[,[fu1][,fu2]]  "cr"
```

Signification des paramètres :

liste d'espaces	numéros des espaces non significatifs du disque (séparés par des virgules) Une plage de numéros consécutifs s'exprime sous la forme n1-n2 (tous les numéros compris entre n1 et n2, bornes incluses). Par défaut de la commande NOSP tous les espaces qui ont été définis sur le disque sont traités.
texte	texte qui sera écrit sur chacun des supports utilisés pour la sauvegarde et édité lors de la restitution Il est recommandé que ce texte comporte la date de la sauvegarde. Le nombre de caractères est limité à 38.
fu1	FU initiale du disque à sauvegarder

Par défaut, il s'agit du disque système.
Pour la sauvegarde d'un disque Winchester sur cassette, fu1 représente l'espace à traiter. Lorsque ce paramètre est absent, les espaces système (D1 et D2) sont traités.

f u 2 — support de la sauvegarde :
- T1, T2, T3 ou T4 pour la bande magnétique
- FU initiale de la cartouche

Par défaut la sauvegarde sera réalisée avec la cassette pour les disques Winchester DWB, avec la disquette pour les disques DWF20. avec T1 pour les autres disques.

Exemples :

- Sauvegarde du disque système sur bande magnétique, les espaces 5, 12, 13, 14 et 15 n'étant pas significatifs :

```
NOSP,5,12-15
SAVE. "DISQUE SYSTEME 10/3/82"
```

- Sauvegarde d'un disque de données (FU initiale El) sur bande magnétique (T2) :

```
SAVE, "DISQUE DATA3 13/4/82", El, T2
```

- Sauvegarde du plateau fixe d'un disque de données (FU initiale E2) sur cartouche (FU initiale E3) :

```
SAVE, "DISQUE DATA APPLI4 15/4/82" , E2, E3.
```

- Sauvegarde d'un espace (D8) d'un disque Winchester DWB sur cassette :

```
SAVE , "ESPACE 2 DISQUE DATA 8/6/84", D8
```

- Sauvegarde des deux espaces système (D1 et D2) d'un disque Winchester DWB sur cassette :

```
SAVE, "SYSTEME 8/6/84"
```

- Sauvegarde d'un disque Winchester DWF20 de FU initiale E3 sur une disquette 5"1/4, les espaces 2, 21 et 22 étant les seuls significatifs :

```
NOSP,.l,3-20,23-27
SAVE, "DISQUE DATA APPLIS 28/11/86", E3
```

Remarques :

- Après chaque sauvegarde, BOS-R invalide la liste d'espaces qui lui a été communiquée par une commande NOSP.

- Lorsque, en cours de sauvegarde, le support est complet (cartouche, bande, cassette ou disquette) et que la sauvegarde n'est pas terminée, BOS-R imprime le message :

```
MONTER LE VOLUME SUIVANT ET FRAPPER LA COMMANDE CSAV
```

et se met en attente de la commande CSAV.

- Les espaces initiaux du disque à sauvegarder et des cartouches de sauvegarde doivent avoir été définis de manière à permettre l'accès à l'ensemble du support.

6.2.3 RESTITUTION D'UN DISQUE

La restitution d'un disque Winchester à partir du support cassette s'effectue de façon différente selon qu'il s'agit des espaces système (D1 et D2) ou d'un espace de données. Elle est réalisée par le bootstrap (fonction KD) pour le système et par BOS-R pour les données.

La restitution des autres disques s'effectue sous le contrôle d'une version de BOS-R bootstrapée à partir du support de sauvegarde (restitution du disque système) ou à partir du disque système (restitution d'un disque de données).

Elle est demandée par l'utilisateur au moyen de la commande REST.

Lors de la restitution les bandes, les cassettes, les cartouches ou les disquettes sont traitées dans un ordre identique à celui de la sauvegarde.

Forme :

```
REST,[fu1],fu2 "cr"
```

Signification des paramètres :

fu1	support de la sauvegarde : - T1, T2, T3 ou T4 pour la bande magnétique - FU initiale de la cartouche ou de la disquette. Par défaut la restitution sera effectuée avec la cassette pour les disques Winchester DWB, avec la disquette pour les disques DWF20-0, avec T1 pour les autres disques.
fu2	FU initiale du disque à restituer

Exemples :

- Restitution d'un disque SMD (FU initiale El) à partir d'un support bande magnétique (T1) :

```
REST, ,E1
```

- Restitution d'un plateau fixe (FU initiale E3) à partir d'un support cartouche (FU initiale E4) :

```
REST,E4,E3
```

- Restitution d'un disque Winchester DWF20 (FU initiale E3) à partir d'un support disquette [FU initiale E4) :

```
REST,E4,E3
```

- Restitution d'un espace (FU D8) d'un disque Winchester sur cassette (DWB) :

```
REST, ,D8
```

Remarque :

Lorsqu'en cours de restitution BOS-R détecte une fin de volume, il imprime le message :

```
MONTER LE VOLUME SUIVANT ET FRAPPER LA COMMANDE CRES
```

et se met en attente de la commande CRES.

A) En sauvegarde :

- Lecture d'un disque Winchester DWB sauvegardé sur cassette : BOS-R imprime le message :

  DEFAUT ENTREE-SORTIE

- Lecture d'un autre type de disque : BOS-R imprime l'adresse du bloc qu'il n'a pas pu lire et met en oeuvre une procédure de récupération. Cette procédure consiste à lire secteur par secteur en effectuant 10 tentatives pour chaque secteur. Si la récupération échoue il y a impression des adresses des secteurs non sauvegardés, sinon BOS-R imprime le message :

  DEFAUT RECUPERE

- Ecriture d'une bande magnétique, d'une cassette, d'une cartouche ou d'une disquette : le volume doit être abandonné. Après le montage d'un autre volume BOS-R reprend la sauvegarde. Le point de reprise correspond au début du traitement du volume défectueux.

B) En restitution :

- Lecture d'une bande magnétique ou d'une cassette : l'exécution est abandonnée.

- Ecriture d'un disque Winchester (à partir d'une cassette) : BOS-R imprime le message :

  DEFAUT ENTREE-SORTIE

- Lecture d'une cartouche ou d'une disquette ou écriture d'un disque : BOS-R imprime l'adresse du bloc qu'il n'a pas pu lire ou écrire et met en oeuvre une procédure de récupération. Cette procédure consiste à lire ou écrire secteur par secteur en effectuant 10 tentatives pour chaque secteur. Si la récupération échoue, il y a impression des adresses des secteurs non restitués, sinon BOS-R imprime le message :

  DEFAUT RECUPERE

6.2.5 GENERATION DE BOS-R

La génération d'un utilitaire BOS-R adapté aux besoins de l'utilisateur est réalisée automatiquement.

Elle peut s'effectuer simultanément avec celle du système BOS16. Dans ce cas il suffit d'ajouter une nouvelle macro-instruction à celles de GBOS16 gui sont nécessaires pour BOS16 (cf. par. 7.4.3) :

```
%BOSR
```

Le moniteur d'entrée-sortie intégré dans BOS-R est alors celui de BOS16, ce gui permet à l'utilisateur d'utiliser les mêmes FU sous les deux systèmes.

Une génération séparée de BOS-R nécessite la préparation de deux jeux de macro-instructions :

- macro-instructions de GIO16 décrivant le moniteur d'entrée-sortie intégré dans l'utilitaire,

- macro-instructions de GBOS16 décrivant les phases de la génération.

Exemple de jeu de macro-instructions de GBOS16 :

```
%FUGENE E2
%BOSR  NOBOS16
%MACRO IOCS ON MACIO-UT,D2
%FIN
*END
```

La génération de BOS-R étant séparée de celle de BOS16, la macro-instruction BOSR comporte le paramètre NOBOS16.

GBOS16 permet d'obtenir le fichier image mémoire de l'utilitaire BOS-R généré, BOSR-:S. Il faut ensuite implanter ce système dans la FU bootstrap D1, afin qu'il puisse être appelé par une commande INIT. Ceci est réalisé par l'intermédiaire de la commande GSYS.

Remarque :

Pour les macro-instructions de GIO16 se reporter au manuel d'utilisation d'IOCS16.

6.2.6 MESSAGES D'ERREURS EMIS PAR BOS-R

ERREUR SYNTAXE COMMANDE Commande incorrecte. Cette erreur peut concerner soit la forme de la commande, soit les FU utilisées.

ESPACE INITIAL INCORRECT L'espace initial ne permet pas l'accès à l'ensemble du support. Le message comporte le nom de l'espace.

ERREUR MONTAGE VOLUME Erreur détectée par IOCS16 lors du montage du volume à sauvegarder. Le message comporte le compte-rendu d'IOCS16.

LISTE D'ESPACES INCORRECTE
La liste d'espaces n'est pas adaptée à la structure du volume.

ERREUR ENCHAINEMENT VOLUME — Erreur dans l'ordre de traitement des volumes.

DEFAUT EN VERIFICATION — Différence entre les informations sur bande et sur disque détectée par la lecture de contrôle.

DEFAUT LECTURE DISQUE
DEFAUT ECRITURE DISQUE
DEFAUT LECTURE BANDE
DEFAUT ECRITURE BANDE — Erreur détectée lors d'une entrée-sortie. Le message comporte le mot d'état de l'unité physique

DEFAUT ENTREE-SORTIE — Erreur fatale détectée lors d'une entrée-sortie disque/cassette Winchester.
Le message comporte l'octet poids fort du premier mot d'état complémentaire contrôleur : XY, X représentant le type d'erreur et Y le numéro du code d'erreur (cf. Manuel d'exploitation des disques durs 8" ou 5" 1/4).

LABELS DIFFERENTS — Les labels du disque sauvegardé et du disque à restaurer sont différents.'
Il est alors demandé à l'utilisateur de confirmer sa demande.

SAUVEGARDE D1 IMPOSSIBLE La taille de Dl est trop importante pour permettre sa sauvegarde sur un seul volume avec le support utilisé (disquette).

AUCUNE FU DISQUETTE ! — La commande émise nécessite la disquette comme support par défaut, mais la configuration ne comporte aucune FU disquette.

6.3 BIBLIOTHEQUE TEMPS REEL

6.3.1 INTRODUCTION

Les sous-programmes de la bibliothèque BIBRTE-RT ont pour but d'établir l'interface entre un programme utilisateur écrit en langage FORTRAN ou PL16 et les requêtes programmées de type temps réel de BOS16.

Une requête programmée est constituée d'une séquence d'instructions et, en règle générale, d'une table de paramètres qui spécifient l'opération demandée : le sous-programme récupère les paramètres passés lors de l'appel (CALL), les met en forme pour que ceux-ci soient assimilables par la requête.

6.3.2 SYNTAXE FORTRAN

Elle se présente sous la forme :

```
CALL RNAME (P1, P2, ..., Pn)
```

où RNAME est un nom de sous-programme externe faisant partie de la bibliothèque système et intégré au programme utilisateur par édition de liens.

P1, P2, ..., Pn sont des adresses de variables ou de tableaux entiers.

6.3.3 SYNTAXE PL16

Selon le niveau de programmation adopté la syntaxe d'une requête peut dériver de la syntaxe FORTRAN ou de la syntaxe assembleur.

A) SYNTAXE FORTRAN

Une tâche écrite en PL16 peut appeler une requête avec la syntaxe FORTRAN à condition que les divers paramètres de cette requête soient des pointeurs (les paramètres passés en FORTRAN étant des adresses).

La liste des paramètres doit être suivie d'une instruction de chargement du registre accumulateur avec le nombre de paramètres effectivement passés.

Exemple :

```
CALL SEVENT (ENUMEV, IERR, RA := 2);
CALL CEXIT (RA := O);
```

Remarque :

WAIT et EXIT sont des mots réservés du langage PL16.

Il est alors indispensable d'utiliser les deux sous-programmes sous des noms différents : CWAIT et CEXIT.

B) Syntaxe assembleur

Une tâche écrite en PL16 peut appeler une requête avec une syntaxe voisine de la syntaxe assembleur à condition d'avoir auparavant déclaré l'instruction SVC.

Le contenu de la table des paramètres est alors le même qu'en assembleur.

Exemple :

```
INSTRUCTION SVC = (3, '1C00);
CONSTANT FREEM q '33;

--------------------

SVC (FREEM);
```

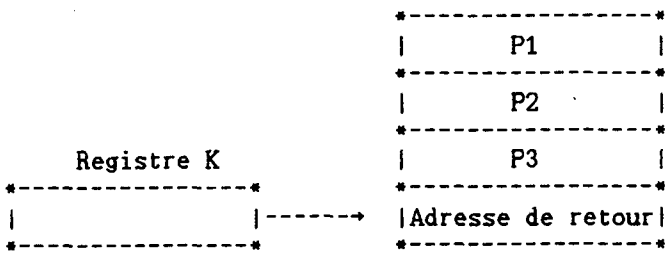
6.3.4 DEROULEMENT D'UNE REQUETE DANS LA SYNTAXE FORTRAN

A) Passage des paramètres

En langage FORTRAN comme en langage PL16. le passage des paramètres s'effectue dans la KSTORE; cette fonction est réalisée par les compilateurs.

Exemple :

CALL SPi(P1, P2, P3)



Dans tous les cas, les paramètres passés sont des adresses de variables ou tableaux.

B) Fonctions réalisées par un sous-programme

- Récupération des paramètres dans la KSTORE, élaboration d'une table de paramètres (RPB) pour la requête (si celle-ci en comporte).
- Pré-détection d'erreurs dans la liste des paramètres, dans certains cas.
- Tassage de la KSTORE.
- Réalisation du branchement vers le sous-programme superviseur par l'intermédiaire de l'instruction SVC de numéro approprié.
- Retour à l'appelant.

Nota :

La table des paramètres d'une requête est interne au sous-programme.

C) Détection des erreurs dans un sous-programme

La plupart des requêtes comportent un nombre imposé de paramètres.

Lorsque le nombre de paramètres transmis par une requête est inférieur à ce nombre la requête n'est pas effectuée, l'appelant n'est pas suspendu et le compte-rendu de traitement n'est pas rangé.

Dans un programme FORTRAN il est alors conseillé avant d'effectuer l'appel, d'initialiser la variable chargée de recevoir le compte-rendu avec une valeur non utilisée par la requête (-1 par exemple); si après un appel cette valeur n'est pas modifiée une erreur a été détectée par le sous-programme.

Dans un programme PL16 le registre accumulateur au retour du sous-programme a pour valeur 6.

Remarque :

Dans le cas où le nombre de paramètres est supérieur au nombre imposé la requête est acceptée; les paramètres sont alors exploités dans l'ordre où ils sont déclarés.

D) Compte-rendu de requête

Le compte-rendu est rangé par la requête à l'adresse précisée lors de l'appel.

Dans le cas du PL16 la valeur du compte-rendu est également contenue dans l'accumulateur.

6.3.5 COMPOSITION DE LA BIBLIOTHEQUE

La bibliothèque se compose des sous-programmes des requêtes du système RTES16, à l'exception de :

IOCS, WEIO
FMS. FMSS, FMSI, FMSD
BCHLR, BACK, BACKR
RMDEF

Elle comporte par conséquent les sous-programmes de toutes les requêtes de type temps réel de BOS16 (avec les restrictions ci-dessus).

Outre ces sous-programmes, la bibliothèque comporte plusieurs sous-programmes communs qui rendent plus rationnelle l'occupation mémoire puisqu'ils ne sont pas dupliqués lors de leur utilisation.

Une telle organisation interdit l'emploi dans les programmes écrits en langage PL16 ou FORTRAN des symboles RTESDi, 0 <= i <= 6 (RTESDi : nom de ces sous-programmes).

6.3.6 EXEMPLE D'UTILISATION

Dans l'exemple suivant la tâche est écrite en PL16 ; il s'agit de la tâche software de niveau de priorité 15.

Elle réalise les fonctions suivantes :

- attente de l'événement de numéro 3
- lecture du mot 23 de la zone de données résidentes.

```
<<
<< TACHE SOFTWARE DE NIVEAU DE PRIORITE 15
<<
MAIN PROCEDURE TASK15
.KSTORE SECTION KT15
     RES 30;
.COMMON SECTION CT15
     EXT PROCEDURE TASK15;
     REF PROCEDURE WEVENT,
                   REDGET;
     WORD NEV=(3),
          NUMBER R=(0),
          NBRE=(1),
          NDEP=(23),
          MOTLU;
.LOCAL SECTION LT15
     SOFT TASK 15 (
          RC=CT15,
          RL=LT15,
          RK=KT15,
          START=TASK15);
.USING RC IS CT15,
       RL IS LT15,
       RK IS KT15;
DEPART:
     CALL WEVENT(@NEV, @NUMERR, RA:=2);
     CALL REDGET(@NBRE, @NUMERR, @NDEP, @MOTLU, RA:=4);
     QUIT;
     GOTO DEPART;
END.


*JOB T15,UT,D3
*CALL PL
*IPLC
*CALL LKLOAD
*MODE S
*LINK *.
*STAT

   WEVENT
   REDGET

*LOOK BIBRTE-RT,D2;
*CATAL IM,TASK15
```

6.4 LA GESTION DE VOLUMES

6.4.1 PRINCIPES GENERAUX

A) Généralités

La gestion de volume est un service obligatoire de BOS16.

Elle est opérationnelle pour tous les disques.

BOS16 impose donc que tous les supports des disques soient utilisés avec la gestion d'espace.

Les commandes IDEF, SDEF (LTAG) de FUP4 permettent de rendre exploitables les supports disque.

Les objectifs généraux de la gestion de volume sont :

- Sécurité de fonctionnement
- Portabilité des supports
- Portabilité des programmes
- Souplesse de génération.

La gestion de volume comprend 2 niveaux de service :

- La gestion des supports c'est-à-dire la gestion des labels et des tables de cylindres en défaut des supports disque.
- La gestion d'espace :

    1) la possibilité d'inscrire sur le support, sa structure en espace IOCS16 et FMS16
    2) la reconfiguration dynamique des tables de IOCS16 et FMS16 en fonction de la structure du support disque.

B) Quelques définitions

- On appelle Unité Physique le périphérique dans lequel se trouve un support disque ou dans lequel on place un support disque amovible. Certains périphériques sont décomposés en plusieurs parties appelées emplacement d'unité physique. Exemple : les disques à têtes mobiles 10 ou 20 Moctets pour lesquels on distingue 2 emplacements. un pour le disque fixe, un pour les cartouches. Par commodité, le terme Unité Physique, dans la documentation traitant de la gestion d'espace, sera utilisé pour désigner également un emplacement d'unité physique.

- On appelle volume un support disque identifié par un label, que ce support soit fixe ou qu'il soit démontable.

- On appelle espace disque une partie d'un disque définie par son adresse et sa longueur (exprimées en cylindres).

- Les espaces d'un disque, sont ordonnés et identifiés par un numéro de zéro à 27 dans une table d'espace décrivant la structure du disque, et inscrite dans le disque.

- On appelle espace initial l'espace de rang zéro dans la table d'espace. L'espace initial commence obligatoirement à l'adresse zéro du disque. Le programme d'initialisation le définit avec une longueur égale au nombre de cylindres formatés.

- Du point de vue de la programmation, une Unité Physique est adressable par les Unités Fonctionnelles (FU] qui lui ont été affectées à la génération du système (GIO16). Les FU d'une Unité Physique sont générées dans un ordre précis, de 0 à 27, dans les tables de IOCS16, à l'aide de

macro-instructions de GIO16.

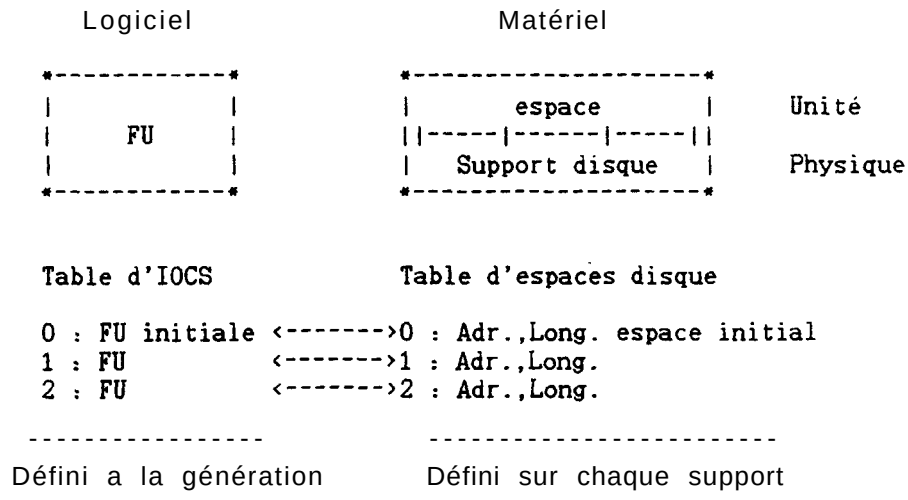
- On appelle FU initiale la première FU générée (GIO16).

- On appelle Symbolic Unit (SU) le Nom Logique qui sera affecté à une FU par les commandes du système d'exploitation et qui permet de réaliser des programmes portables.

- On appelle FU disque mnémonique l'une des 29 FU disques qui possèdent un nom, appartenant aux plages D1 à DF, E1 à EF (sauf EC).

- On appelle FU disque numérique une FU disque sans nom : 68 FU de 58 à 125 (en décimal).

C) Principe général : niveau IOCS16



Le ième espace du disque est accessible par la ième FU générée pour cette unité physique.

- La reconfiguration dynamique des tables de IOCS16 exécutée à l'occasion d'une commande de montage de volume consiste à recharger les tables de IOCS16 avec les valeurs adresse et longueur inscrites sur le support en respectant la règle d'affectation par rang.

  Il est vivement conseillé de configurer l'espace initial d'un support disque avec une longueur qui couvre la totalité du disque. Autrement dit, ne pas modifier l'option qui est implicitement choisie par le programme de formatage des disques, à savoir :
  Adr = 0 (non modifiable)
  Long = Nombre total de cylindres formatés (modifiables si nécessaire avec Long. # O).

- La table d'espaces d'un disque est située dans le secteur d'adresse 4 du cylindre zéro du disque. Elle est initialisée par le programme de formatage, dans le cas des disques 10, 20 et 50 Moctets, ou par les utilitaires INFDD, INSMD, INSAS, dans le cas des autres disques, et mise à jour à l'aide de la commande SDEF de FUP4.

- Une FU IOCS16 est dite ouverte lorsqu'il lui correspond dans les tables de IOCS16 un couple (adresse, longueur) définissant un espace disque. Dans le cas contraire, une FU IOCS16 est dite fermée ; en particulier, après l'exécution d'une commande de démontage de volume.

- On appelle structurer un support disque. l'opération (FUP4 : SDEF) qui consiste à créer ou modifier la table d'espaces d'un disque.

Règles :

- Les supports disques doivent tous être formatés et structurés.

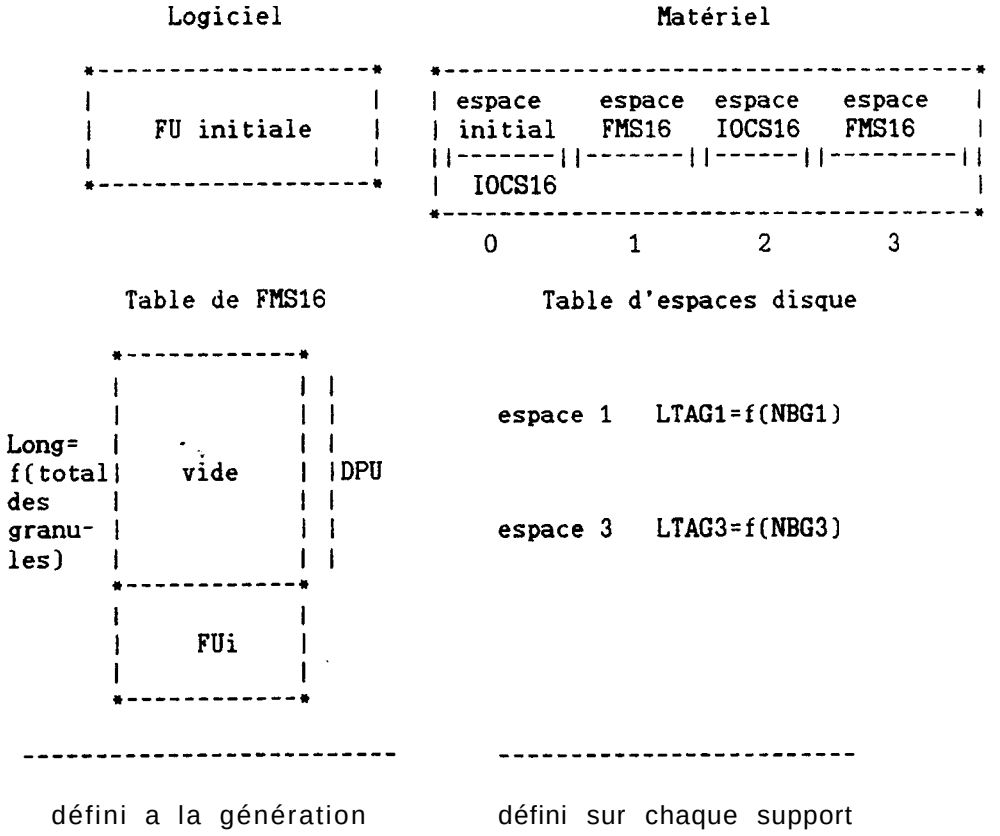
- Les unités physiques correspondantes doivent être générées de façon dite dynamique standard. C'est-a-dire, qu'il faut générer l'ensemble des Unités Fonctionnelles (FU) utilisables sur cette Unité Physique. En se

rappelant que l'ordre de génération des FU définit l'ordre d'affectation des espaces et que la FU initiale sert à identifier l'unité physique.

Une génération IOCS16 dynamique standard est dite vide, c'est-a-dire que les paramètres adresse = 0 et longueur = 0 définissent une FU "fermée".

Elle doit être utilisée pour toutes les unités physiques sur lesquelles il y aura reconfiguration dynamique, suite à une commande de montage de volume ou automatiquement au lancement du système.

D) Principe général : niveau FMS16



- LTAG indique la taille en mots de la Table d'Allocations des Granules d'un espace disque FMS16, NBG est le nombre de granules d'un espace FMS16.

- La table d'espaces indique pour chacun s'il est géré par FMS16 (LTAG <> 0).

- La table d'espaces est automatiquement mise à jour par la commande FUINI de FUP4, lorsque l'utilisateur initialise un espace pour FMS16.

- La reconfiguration dynamique de la table d'une Unité Physique FMS16 consiste à créer les descripteurs des FU gérés par FMS16 sur un support; par exemple lors de l'exécution d'une commande de montage de volume.

Règles :

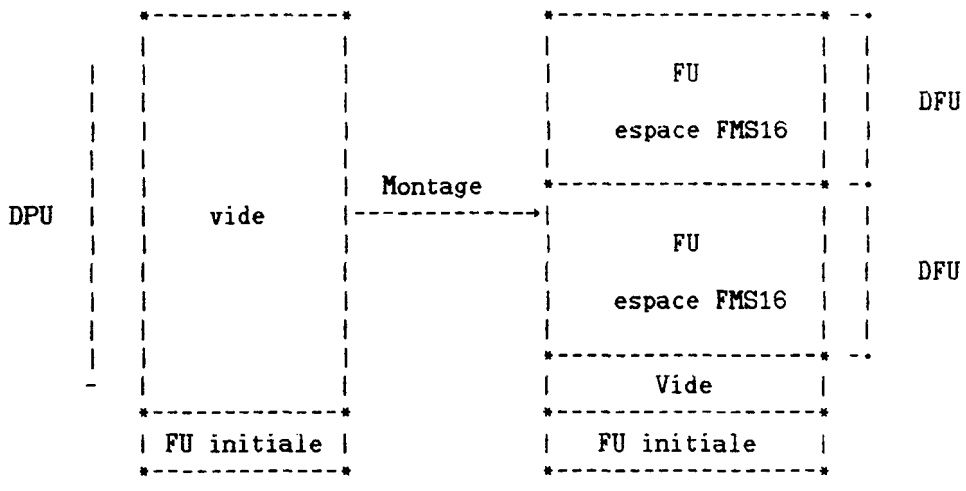
- Une même Unité Physique doit être identifiée par le même numéro de FU initiale dans les tables de IOCS16 et de FMS16.

- La génération d'une Unité Physique FMS16, est dite statique lorsqu'elle ne permet pas de reconfigurer dynamiquement les tables. Elle est utilisable pour les disques sans gestion d'espace c'est-à-dire les disques à têtes fixes. Elle est réalisée à l'aide des macro %CONFMS situées en tête de la génération GFMS16.

- La génération d'une Unité Physique FMS16 est dite dynamique lorsqu'elle permet la reconfiguration dynamique en citant à l'aide d'une macro %PUFMS, la FU initiale correspondante, et la place que l'on doit réserver pour tous les descripteurs de FU FMS16 de l'unité physique (DFU). Un DFU contiendra la Table d'Allocation de Granules (TAG) de l'espace FMS16 correspondant.

- Une unité physique FMS16 générée dynamiquement est normalement vide; c'est-à-dire que les FU correspondantes sont inexistantes pour FMS16.

Schéma de génération dynamique vide d'une unité physique FMS16 (DPU)



Génération dynamique après démontage

Après reconfiguration dynamique

E) Règles d'utilisation sous BOS16

Sécurité

La commande MONT permet de faire connaître au système les caractéristiques d'un volume donné :

- Label
- Table des cylindres en défaut (éventuellement)
- Adresse, longueur des espaces IOCS16
- Taille des descripteurs de FU gérées par FMS16.

La commande DMONT permet de vérifier que l'on a bien le droit de démonter un volume (fichiers fermés). Ensuite, les FU IOCS16 et FMS16 sont respectivement fermées et inexistantes.

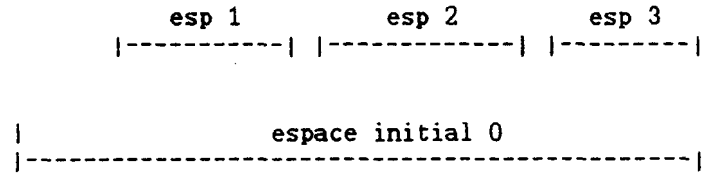
Attention :

On peut mettre ou enlever physiquement un support amovible sans que le système en soit averti. La sécurité n'est obtenue que lorsqu'on utilise les commandes de gestion de volume.

Portabilité des supports

La portabilité des supports amovibles est assurée par la reconfiguration dynamique des tables de IOCS16 et FMS16, suite à un montage de volume.

Exemple : Rechargement de trois espaces FMS16 dans un disque sous BOS16



```
CALL FUP4
DMONT          fermeture de l'unité physique (sauf FU initiale)

SDEF 1      )
SDEF 2      )  définition de 3 espaces
SDEF 3      )

MONT           reconfiguration IOCS16
FUINI       )
FUINI       )  initialisation FMS16 des 3 espaces
FUINI

MONT           reconfiguration IOCS16 et FMS16

CALL FUP2

FUST        )
FUST        )  contrôle de l'initialisation des espaces IOCS16 et FMS16
FUST

CALL FUP3

REST ou DUPL )
REST        )  chargement des fichiers
REST        )
```

Règles

- Structurer tous les supports des disques, y compris le disque système.

- Générer un système maximal, c'est-à-dire, pour chaque unité physique générer un nombre de FU égal au plus grand nombre d'espaces définis sur un support.

- Générer une table de FMS16 (DPU) qui puisse contenir tous les descripteurs de FU (DFU) du support.

- Programmer sur SU aussi bien au niveau des programmes (SVC) qu'au niveau des fichiers de commandes.

- Réaliser l'affectation d'une SU au ième espace disque par la commande SUSPACE (par. 6.4.2).

Remarque :

Pour assurer la comptabilité des applications programmées sur FU, il est possible de figer la correspondance FU <-> (Adr. Long.) en utilisant la possibilité de laisser des trous dans une table d'espace. Voir notices IOCS16, FMS16.

Souplesse de génération

- Une partie de la génération est réalisée au niveau de chaque support disque, d'où une grande variété de découpes possibles

  jème FU <--------> jème espace (adr., long.)

- Associé à une programmation sur SU, ce type de génération, permet une économie en nombre de FU IOCS16 et sur la taille des tables de FMS16.

  Le nombre de FU IOCS16 à générer n'est plus lié au nombre de couples (adr.,long.) différents, mais au nombre d'espaces d'un support disque.

  La taille des tables de FMS16 n'est plus fonction de tous les espaces FMS16 possibles (adr.,long.), mais fonction du support disque qui possède le plus grand nombre de granules (en tenant compte de tous ses espaces FMS).

- Il est possible d'étendre les FU disques utilisables par commande au-delà

de l'ensemble des 29 FU disques mnémoniques.

Règle :

A condition de programmer sur SU et en accord avec la commande SUSPACE, il est possible de générer au niveau IOCS16 des FU disques numériques (au niveau FMS16 réaliser une génération dynamique standard, c'est-à-dire vide en utilisant seulement la macro %PUFMS).

Attention :

Il est nécessaire que la FU initiale soit mnémonique. Il est vivement conseillé que la FU de rang 1 le soit également. De toute façon, il ne faut utiliser les FU numériques que lorsque la configuration nécessite plus de 29 FU disques.

Après une reconfiguration dynamique, les disques avec gestion d'espace se programment au niveau IOCS16 et FMS16 de la même façon que les disques avec génération statique. Le seul problème étant le traitement des compte-rendus d'erreur suite aux anomalies liées au montage de volume : disque mal structuré, FU fermées (IOCS16) ou inexistantes (FMS16), etc...

6.4.2 COMMANDES DE GESTION DE VOLUME SOUS BOS16

A) Commande de démontage

DMONT {SUIFU} "cr"

où :

FU	nom de l'unité fonctionnelle initiale qui identifie l'unité physique.
su	nom de l'unité symbolique à laquelle est affectée la FU initiale.

But : Vérifier qu'aucun fichier n'est ouvert sur le volume avant d'enlever le disque de l'unité physique.
Invalider les FU pointant sur le volume, c'est-a-dire les fermer au sens IOCS16 et les rendre inexistantes au sens FMS16.

Remarque :

Si aucune erreur n'est détectée, le volume peut être déchargé de l'unité physique.

B) Commande de montage

MONT [{SUIFU}] [label] "cr"

où :

FU	nom de la FU initiale qui identifie l'unité physique.
su	nom de l'unité symbolique à laquelle est affectée la FU initiale.
label	label du volume à monter (1 à 7 caractères alphanumériques non blancs).

But : Prendre en compte, au niveau IOCS16 et FMS16, la structure d'un volume.

- Démonter implicitement le volume précédent (s'il reste des fichiers ouverts sur le volume précédent, le montage est refusé).
- Si "label" est présent dans la commande, vérifier la concordance avec le label inscrit sur le volume. (Si "label" est absent aucune vérification de label n'est effectuée).
- Prendre en compte les éventuels cylindres en défaut du volume.
- Prendre en compte les espaces définis sur le volume, et les affecter aux

FU appartenant à l'unité physique.
- Imprimer sur EL le label au volume, ainsi que pour chaque espace monté, son numéro et sa FU associée.
- Le disque système est monté implicitement au lancement du système. L'utilisateur peut cependant connaître les espaces pris en compte sur ce disque, ainsi que les FU auxquelles ils sont affectés, par la commande MONT sans paramètre.

Remarques :

- En général lorsque le volume à monter est non structuré, le montage est refusé. Cependant dans le cas des disques souples, le processus est différent puisque la commande MONT permet l'adaptation d'IOCS16 aux caractéristiques physiques de la disquette à monter (nombre de faces, densité, taille des secteurs, nombre de secteurs par piste).
  Après l'impression par le système du message :
  ERB 44 '6001
  l'utilisateur peut accéder par IOCS16 à ses informations en utilisant la FU initiale.

- Dans les cas d'erreurs :
  ERB 44 '6005 et ERB 45 '6003
  le volume a été monté mais tous les espaces n'ont pu être pris en compte.
  Dans les autres cas d'erreur. le volume n'a pas été monté.

C) Commande d'affectation SU-espace

```
SUSPACE SU,nesp[,FU]  "CR"
```

où :

su	nom d'unité symbolique de U1 à UF
nesp	numéro d'espace (de 0 à 27) dans un volume monté par la commande MONT
FU	nom d'une unité fonctionnelle initiale identifiant l'unité physique (par défaut le dernier disque monté).

But : Affecter à la SU la FU disque pointant sur l'espace "nesp" d'un volume monté et identifié par sa FU initiale.

La commande SUSPACE permet de rendre portables des programmes (ou fichiers de commandes) utilisant des unités symboliques pour accéder aux disques, ces programmes étant :

- dépendants seulement de la structure du volume utilisé,
- totalement indépendants du système utilisé et de l'unité physique sur laquelle est monté le volume.

Exemple de programmation portable en batch :

. JOB sur cartes

```
/JOB NOM,CT,UI
/PAUSE MONTER LE VOLUME TOTO
/SUSP U1.3
/SUSP U2,4
/CC FICOM-FC
/EOJ
```

. Commandes de l'opérateur après PAUSE
* DMONT E5
- enlever le volume du job précédent
- mettre le volume TOTO sur l'unité physique désignée par la FU initiale ES.
* MONT E5,TOTO
* RETURN

. Fichier de commande FICOM-FC,Ul
/CALL PL
/SI SYMBOL
/BO BINAIR-BO,U2
/IPLC
etc

/RUN PROG
/RETURN

Remarques :

. JOB sur cartes

- Le fichier de commande FICOM-FC est situé sur la SU implicite Ul affectée au 3e espace du volume TOTO.

- Il est possible de forcer un contrôle de label en rajoutant après la carte PAUSE
/SUSP U1.0
/MONT U1.TOTO

. Commandes de l'opérateur

- La commande DMONT permet de contrôler que l'on a bien le droit d'enlever le volume utilisé par le job précédent (fichiers fermés) et donc se protéger d'un mauvais fonctionnement de ce job.

- L'opérateur ne connaît pas les numéros de SU utilisés par le job. Il déclare seulement avoir monté le volume TOTO sur E5.

. Fichier de commande

- L'affectation SI SYMBOL désigne implicitement le catalogue CT et la SU U1 de la carte JOB.

- L'affectation BO BINAIR-BO,U2 est totalement explicite. et le fichier est situé sur l'espace 4 du volume TOTO.

Remarque :

En conversationnel, il suffit de remplacer la commande PAUSE par DMONT et MONT.

Si la SU U1 qui pointe sur l'espace 3 peut être utilisée par plusieurs jobs, elle doit être choisie parmi U5 à UF (U1 à U4 sont réaffectés sur EOJ).

D) Commande d'impression de la liste des FU initiales

ISPACE "CR"

L'émission de la commande ISPACE permet d'obtenir sur le périphérique associé à l'unité symbolique EL la liste des FU initiales des disques gérés par IOCS16.

E) Messages d'erreur de montage

Les commandes de montage (DMONT, MONT, SUSP) peuvent conduire aux messages d'erreur suivants :

ERB 44 : compte-rendu IOCS16 (erreur détectée par IOCS16)

ERB 45 : compte-rendu FMS16 (erreur détectée par FMS16).

Compte-rendus IOCS16

ERB 44	'6000	: erreur de génération (cf. par. 2.2.5)
	'6001	: pas de gestion de volume sur l'unité physique ou sur le support à monter Le second cas concerne les disques souples. L'information qu'ils supportent est toutefois accessible à un programme utilisateur à la condition que celui-ci utilise la FU initiale, dont la taille a été adaptée par IOCS16 à La capacité du disque souple.
	'6002	: aucun volume n'est monté sur l'unité physique
	'6003	: erreur de label
	'6004	: erreur de génération (pas de SU enveloppe)
	'6005	: IOCS ne peut prendre en compte tous les espaces Les espaces pris en compte sont ceux imprimés par la commande MONT.
	'6006	: volume non structuré
	'6007	: aucun cylindre valide parmi les 8 premiers

Compte-rendus FMS16

ERB 45	'6003	: sous-dimensionnement de la zone réservée à l'unité physique par GFMS16 (un ou plusieurs espaces peuvent avoir été pris en compte)
	'600C	: erreur de label
	'600E	: erreur ne pouvant apparaître qu'au lancement du système BOS16. IOCS16 ne peut prendre en compte tous les espaces du disque système. L'utilisateur peut connaître ceux qui ont été pris en compte en émettant une commande MONT sans paramètre.
	'6018	: volume mal structuré
	'601A	: type de disque sans gestion de volume
	'601E	: il reste des fichiers ouverts sur le volume
	'602A	: SU ou FU non gérée par FMS16
	'6035	: erreur de génération de BOS16 (cf. par. 7.4)

7 GENERATION DE BOS16 SUR DISQUE A TETE MOBILE

7.1 PRINCIPE

Un système d'exploitation disque BOS-G prêt à fonctionner implanté sur les unités fonctionnelles E1 et E2, permet de définir et d'initialiser les unités fonctionnelles D1 et D2 nécessaires à BOS16.

La génération du système BOS16 adapté aux besoins de l'utilisateur sera faite automatiquement.

7.2 MISE EN OEUVRE DE BOS-G

BOS-G, ainsi que tous les fichiers nécessaires à la génération, sont implantés sur la cartouche de génération.

Mise en oeuvre

- Monter la cartouche de génération sur l'unité de disque de numéro 0, démarrer le disque, attendre que la lampe READY s'allume
- Mettre le sélecteur de bootstrap sur MHD
- Actionner les touches : STOP-INITIALIZE-LOAD-RUN
- Impression sur le périphérique de dialogue de "*"

  BOS-G est alors en attente de commande

7.3 UTILISATION DE BOS-G

7.3.1 INTRODUCTION

BOS-G est un sous-ensemble de BOS16 ne reconnaissant que les commandes nécessaires à la production de programmes.

Lors de son premier lancement, seules les unités fonctionnelles disque E1 et E2 sont définies.

Le moniteur d'entrée-sortie intégré à BOS-G est configuré avec les périphériques suivants :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP	FU
Périphérique de dialogue	Indifférent		'17F8	Programmé Simple			TK,TS
Disque à tête mobile	14	1	'30	HDC ou MDC	1	0 ou 1 0	D1 à DA DF,E1,E2,E4
Disques souples	14	Paramètres non définis					Un.0:E6,E7 Un.1:E8,E9
Lecteur de cartes	14	Paramètres non définis					CR
Imprimante	14	Paramètres non définis					LP
Bande magnétique	14	Paramètres non définis					T1

Le processeur CONFIG permet de définir les caractéristiques des périphériques suivants : disques souples, lecteur de cartes, imprimante et dérouleur de bande magnétique [cf. par. 7.3.3).

Restriction : l'imprimante ne peut être utilisée sous BOS-G que si elle est gérée par le driver imprimante standard.

Remarques :

- Pour le disque, la mise à jour du mode de fonctionnement (HDC ou MDC) et du processeur (IOP = 0 ou 1) est effectuée automatiquement par BOS-G.

- La définition des FU D1 et D2 sera faite par le processeur FUP4 (cf.par. 7.3.4).

7.3.2 PARTICULARITES DE BOS-G

BOS-G fonctionne indifféremment avec comme périphérique de dialogue un téléimprimeur, un terminal imprimante 30 caractères/seconde (TER 30), une console de visualisation. Cela amène quelques particularités de fonctionnement.

- le défaut "time out" sur téléimprimeur qui se traduit sous BOS16 par le message

  ERB 05

  n'apparaîtra jamais sous BOS-G

- l'appel opérateur (touche BREAK) n'est effectif sous BOS-G que pendant un échange sur le périphérique de dialogue

- après une commande EBOS de BOS-G, on ne peut pas relancer le dialogue de BOS-G par l'appel opérateur. Il faut rappeler BOS-G par le bootstrap

  STOP - INITIALIZE - LOAD - RUN

- la commande INIT est refusée sous BOS-G avec disque à têtes mobiles.

7.3.3 PROCESSEUR CONFIG

Ce processeur est intégré dans la FU E2. Il n'est utile que dans la phase de mise à jour du système. Cette phase se terminera après l'émission de la commande CONF qui rendra permanentes les modifications apportées.

- Dans le cas d'un disque à têtes mobiles, enlever la protection en écriture de la cartouche et du plateau fixe.

- Appel de CONFIG : émettre les commandes

```
LO ZE
JOB  CONFIG,,E2
RUN  CONFIG-:S
```

- Lorsque le périphérique de dialogue est TER 30 émettre la commande : TER30

- Suppression d'un périphérique absent de la configuration ou non géré par le driver standard : émettre la commande NO {HRIHPICRILP}

- Définition des caractéristiques d'un périphérique : émettre la commande DEF {CRILPIMTIFD}

CONFIG pose les questions :

ADRESSE ? réponse "cr" (adresse du périphérique désigné dans la commande - adresse ON ou OFF RACK)

SOUS-NIVEAU EXCEPTION ? réponse "cr" (0 à 8)

NUMERO D'IT NORMALE : réponse "cr" (0 à 8)

MODE CANAL ? réponse "cr" (LDC. MDC. HDC)

NUMERO DE PROCESSEUR E/S ? réponse "cr" (0 à 3)

et seulement pour MT :

DENSITE (800 ou 1600) ? réponse "cr" (800 ou 1600)

- Validation de toutes les commandes précédentes. Emettre la commande

```
CONF "Cr"
```

Après cette commande, les suppressions et définitions de caractéristiques de périphériques sont mémorisées définitivement par BOS-G. En particulier, lors d'un lancement ultérieur de BOS-G, il ne sera plus nécessaire d'appeler CONFIG.

- Structurer éventuellement la partie du disque (FU D1, D2) à l'aide des commandes SDEF et FUINI de FUP4 (cette opération n'est pas nécessaire dans le cas d'une mise à jour, la structure du fixe étant conservée).

- Emettre la commande MONT DA.

BOS-G est opérationnel. La génération de BOS16 consistera à créer, dans la FU D2 actuellement vide, les fichiers suivants :

- processeurs en image mémoire
- bibliothèques
- fichier système supportant BOS16.

Exemple d'utilisation de CONFIG :

```
*LO ZE
*JOB CONFIG.,E2
*RUN CONFIG-:S Appel de CONFIG
*NO CR          On ne dispose pas de lecteur de cartes
* DEF LP
. . . . . .
*CONF
```

Remarque :

Pour utiliser la cartouche de génération sur une autre configuration, d'abord revenir à la configuration standard (commande ZCONF) puis recommencer la configuration.

7.3.4 INITIALISATION DU DISQUE SYSTEME

BOS-G sur cartouche est utilisable sur les cartouches utilisées sur les périphériques 10 Méga et 20 méga.

Ce système connait la partie fixe du périphérique comme un support volume [cf. Notices FMS16 et IOCS16).

Il connaît 10 FU (D1 à DA), DA étant la FU initiale.

Les FU gérées par FMS16 ne peuvent pas occuper au total plus de 3600 granules.

La génération d'un système BOS16 adapté à la configuration de l'utilisateur nécessite l'initialisation du disque-le plateau fixe-qui sera utilisé par la suite en tant que disque système.

Cette initialisation est réalisée par le processeur FUP4.

```
*DMONT DA
*LO ZE
*RUN FUP4 - : S
*IDEF,DA,label,X,128,Y.400      Disque 20 Méga : X = 4 Y = 144
                                 Disque 10 Méga : X = 1 Y = 48

*SDEF,DA,label,1,0,longueur      Dl
*SDEF,DA,label,2,début   D2,longueur   D2
*SDEF,DA,label,3,début   D3,longueur   D3
*MONT DA
```

Il faut initialiser D2 qui sera gérée par FMS16 ainsi que les autres FU au même type.

```
*FUINI,FU[,tg]
*MONT  DA
```

Pour plus de précision consulter la notice FUP.

7.3.5 INTEGRATION DES PROCESSEURS

La cartouche de génération supporte les processeurs, bibliothèques et générateurs associés à BOS16.

Leur intégration dans l'unité fonctionnelle D2 s'effectue au moyen des commandes :

```
LO ZE
RUN  FUP3-:S
DUPL,PROCl6-RP,E2,D2
```

Les processeurs LKLOAD, ASM, FUP3, MACP, BUILD sont indispensables pour la génération de BOS16.

7.3.6 INTEGRATION DES BIBLIOTHEQUES

Diverses bibliothèques peuvent être nécessaires pour l'utilisation de BOS16.

- bibliothèque temps réel BIBRTE-RT
- bibliothèque FORTRAN
- etc...

Ces bibliothèques se trouvent sur la cartouche de livraison.

Emettre les commandes :

```
*CALL FUP3
*FDUP,nomfic-catg,E2,nomfic-catg,D2
```

7.4 GENERATION DE BOS16

La génération de BOS16 nécessite la préparation de 3 jeux de macro-instructions dans des fichiers disque :

- macro-instructions de GIO16 décrivant le moniteur d'entrées-sorties IOCS16 du système à générer
- macro-instructions de GFMS16 décrivant le système de gestion de fichiers FMS16
- macro-instructions de GBOS16 précisant les différentes phases de la génération.

Quand ces 3 jeux de macro-instructions sont prêts, la génération peut se dérouler automatiquement.

7.4.1 MACRO-INSTRUCTIONS DE GIO16

Les macro-instructions de GIO16 décrivent le moniteur d'entrée-sorties IOCS16, c'est-à-dire :

- les périphériques de l'installation
- le découpage des disques en unités fonctionnelles disque.

Il faut donc écrire un jeu de macros approprié à la configuration (cf. Manuel d'utilisation IOCS16), qui sera traité automatiquement lors de la génération.

Exemple :

Pour une installation comportant :

- un téléimprimeur
- deux unités de disques souples accessibles par les unités fonctionnelles
  E1 (FU initiale), D1 et D2 pour l'unité 0
  E2 (FU initiale), D3, D4 et D5 pour l'unité 1
- un lecteur de cartes et une imprimante
- une horloge temps réel

tous ces périphériques étant aux adresses débanalisées. les macros à écrire seront :

```
%BOS16
%NIVEAU 14 KSTOR =
%PUFDD SNIV=O MODE=LDC ADR='28 ITN=O IOP=0
%FUIFDD 43 VOIE=0
%FUESPFDD 13
%FUESPFDD 14
%FUIFDD 44 VOIE=1
%FUESPFDD 15
%FUESPFDD 16
%FUESPFDD 17
%CR
%LP
```

```
%NIVEAU 15 KSTOR =
%TTY
%NIVEAU 13 KSTOR =
%HTR SNIV=O FREQ=50
%ENDGEN
* END
```

Remarque :

Plusieurs contraintes doivent être respectées pour la génération de IOCS16 pour BOS16 :

- la première macro doit être %BOS16

- les unités fonctionnelles TS, TK, D1 et D2 (numéros 2, 3, 13 et 14) doivent obligatoirement être définies ainsi qu'une unité fonctionnelie disque au choix de l'utilisateur devant permettre l'accès à l'espace initial du disque système,

- cette dernière FU disque doit être définie par l'intermédiaire d'une macro %FUIxx. en fonction du type de disque,

- les FU disque Dl et D2 doivent être définies par l'intermédiaire de macros %FUESPxx respectivement en seconde et troisième position.

7.4.2 MACRO-INSTRUCTIONS DE GFMS16

A) Les macro-instructions de GFMS16 décrivent le moniteur de gestion de fichiers FMS16 :

- définition des FU disque gérées par FMS16, et des zones mémoires nécessaires à leur gestion,

- génération des commandes nécessaires à l'édition des liens de FMS16.

Les macro-instructions de GFMS16 sont décrites dans le manuel d'utilisation FMS16.

B) Configuration d'une FU disque gérée par FMS16

- Pour que FMS16 puisse gérer une FU disque, deux étapes sont nécessaires :

  . réservation d'une zone mémoire permettant la gestion de la FU
  . initialisation du support physique de la FU.

- Première étape :

La première étape est réalisée à la génération du système :

  . au moyen de GIO16 définir la FU initiale ainsi que les FU qui permettront d'accéder au support,
  . au moyen de FMS16 définir pour la FU initiale un nombre maximum de granules gérables sur le support pour l'ensemble des FU.

Cette étape conduit à l'élaboration d'informations logiques résidentes en mémoire.

- Deuxième étape :

La deuxième étape sera réalisée sous le contrôle de BOS16, après la génération.

Elle consiste, au moyen de FUP4, à :

  . structurer le support, c'est-a-dire à définir les espaces qui seront reconnus sur le support,
  . initialiser chaque espace destiné a être géré par FMS16.

L'initialisation d'un espace consiste à écrire, dans une zone située au début de l'espace, un certain nombre de renseignements tels que la taille

des granules, l'emplacement des granules libres.

Cette initialisation sera réalisée par la commande FUINI (cf. Manuel de Référence de FUP).

Remarque :

Tout espace disque, autre que celui qui correspond à D2, pourra être réinitialisé. en particulier pour redéfinir la taille des granules en vue d'une meilleure organisation du disque.

Toute réinitialisation entraîne cependant la destruction des fichiers de l'espace.

C) Exemple

L'installation donnée en exemple (par. 7.4.1) comporte deux unités de disques souples.

Les FU disque qui ont été définies sont :

- E1, D1, D2 pour l'unité 0 (support du disque système),

- E2, D3, D4, D5 pour l'unité 1.

Avec un disque système structuré de la manière suivante :

- espace initial (0),
- espace 1 de 5 cylindres,
- espace 2 de 70 cylindres (géré par FMS16 avec une taille de granule de 13 secteurs) et en supposant que les disquettes montées sur l'unité 1 n'auront pas plus de deux espaces gérés par FMSl6 et un total de 400 granules. Les macro-instructions de GFMS16 à fournir sont :

```
%PUFMS FUI=E1 OFU=O NBFMSMAX=1 PUNBGRAN=280
%PUFMS FUI=E2 OFU=O NBFMSMAX=2 PUNBGRAN=400
* END
```

7.4.3 MACRO-INSTRUCTIONS DE GBOS16

A) GBOS16 permet de générer l'ensemble des commandes nécessaires à la génération de BOS16.

Le jeu de macro-instructions de GBOS16 à écrire doit décrire les différentes phases de la génération :

Phase 1	Configuration de IOCS16 : création du fichier IOCS-LK
Phase 2	Configuration de FMS16 : création du fichier TABFMS-:S
Phase 3	Edition de liens des modules composant le système : IOCS16, FMS16, BOS16, drivers et éventuellement des modules spécifiques de l'utilisateur (drivers, fonctions spéciales de IOCS16, etc...). puis création de l'image mémoire du système.

B) Description des macro-instructions de GBOS16

Les différentes macro-instructions de GBOS16 précisent sur quel support sont pris les éléments nécessaires à la génération de BOS16.

%FUGENE unité fonctionnelle disque

But : Préciser la FU disque supportant les éléments de génération livrés.

Paramètre : nom de FU disque connu du système sous lequel se déroule la génération (si cette macro est omise la FU de génération est E2).

%BUILD nomsys [FAST] [BRESID]

But : Création du fichier image mémoire de BOS16.

Premier paramètre : nom du fichier image mémoire de BOS16. Le fichier aura le catalogue :S et sera implanté dans la FU spécifiée par la commande JOB.

Deuxième paramètre : Ce paramètre optionnel permet de générer un système BOS16 qui aura de grandes performances, au prix d'un encombrement disque supplémentaire de l'ordre de 4K mots. Pour des performances optimales, BOS16 devra recevoir l'option FASTOV lors de sa configuration (cf. par. 5.1.4 J).

Troisième paramètre : Ce paramètre précise que certaines branches utilitaires de BOS16 sont résidentes, cela conduisant à une amélioration du temps de réponse du système. Ce paramètre est vivement conseillé dans le cas des disques souples car il entraîne également une moindre usure au support du disque système.

%MACRO IOCS ON nomfic-catg[,FU] FMS ON nomfic-catg[,FU]

But : Création des tables de IOCS16 et de FMS16.

Les macro-définitions de GIO16 et de GFMS16 sont supportées par les fichiers GIO16-SY et GFMS16-SY de FUGENE.

Premier paramètre : support des macro-instructions de GIO16. Ce jeu de macros a été préparé au paragraphe 7.4.1.

Deuxième paramètre : support des macro-instructions de GFMS16. Ce jeu de macros a été préparé au paragraphe 7.4.2.

%BIBDRV ON nomfic-catg[,FU]

But : Edition de liens des drivers nécessaires se trouvant dans une bibliothèque de drivers.

Paramètre : nom de la bibliothèque.

On peut utiliser plusieurs bibliothèques de drivers. On mettra alors plusieurs macros %BIBDRV.

%DRV ON nomfic-catg[.FU]

But : Edition de liens d'un module link-éditable quelconque : driver, fonction spéciale de IOCS16 spécifique de l'utilisateur. etc...

Paramètre : fichier support du module.

Il peut y avoir plusieurs macros %DRV.

Remarque :

Toutes les macros %BIBDRV et %DRV doivent se trouver entre les macros %BUILD et %FIN.

%BOSR [NOBOS16]

But : Génération de l'utilitaire de sauvegarde BOS-R (cf. par. 6.2).

Cette macro, lorsqu'elle est utilisée, doit obligatoirement suivre la macro FUGENE

Le paramètre NOBOS16 doit être utilisé pour une génération de BOS-R séparée de celle de BOS16.

%FIN

But : Création de l'image mémoire du système et génération de la commande EOJ qui rendra le contrôle au périphérique de dialogue, à la fin de la génération.

Cette macro est obligatoire à la fin de tout jeu de macros de GBOS16.

%STANDARD [FAST] [BRESID]

But : Cette macro-instruction peut remplacer toutes les précédentes, dans le cas où on dispose d'une cartouche àe génération.

Les macro-instructions de GIO16 doivent se trouver dans le fichier MACIOC-SY et les macros-instructions de GFMS16 dans le fichier MACFMS-SY de la FU de génération (macros %FUGENE ou %MACRO).

Tous les drivers nécessaires sont dans la bibliothèque des drivers standard.

Le système généré aura pour nom BOS16.

Remarques importantes :

- Ne pas oublier que tout jeu de macro-instructions doit se terminer par la directive *EN adressée à MACP.

- La création des fichiers MACIOC-SY et MACFMS-SY se fait au moyen d'EDIT16.

Cl Exemples de jeu de macro-instructions de GBOS16

- Configuration avec disques souples

```
%FUGENE D5
%BUILD BOS16 BRESID
%MACRO IOCS ON MACIOC-SY,D3 FMS ON MACFMS-SY,D3
%FIN
* END
```

- Configuration avec cartouche de génération, utilisation de drivers situés dans la bibliothèque standard, dans une bibliothèque BIBUTI-SP, et sur un fichier (génération de BOS16 et de BOS-R. l'utilitaire de sauvegarde) :

```
%BOSR
%BUILD BOS16 FAST
%MACRO IOCS ON MACIO-UT,D2 FMS ON MACFM-UT,D2
%BIBDRV ON BIBUTI-SP
%DRV ON DRVSP-DV,D2
%FIN
l END
```

- Configuration avec cartouche de génération :

```
%STANDARD
```

D) Choix du numéro du système

GBOS16 permet d'obtenir le fichier image mémoire du système généré. Il faut ensuite implanter ce système dans la FU bootstrap Dl, afin qu'il puisse être appelé par le bootstrap ou une commande INIT. Ceci est obtenu par l'intermédiaire de la commande GSYS comportant en paramètres :

- le numéro du système,
- le nom du fichier système,
- le numéro du secteur de D1 où sera implanté le système; ce numéro doit être supérieur ou égal à 5.

Choix du numéro de système :

Il doit obligatoirement exister un système de numéro 0 dans D1 (cf. par. 6.1.6) ; la première génération doit donc être réalisée avec le numéro 0.

7.4.4 GENERATION DE BOS16

A) La génération de BOS16 nécessite :

- sur la FU D2
  . Les processeurs MACP, ASM, LKLOAD, BUILD, EDIT16

- sur la cartouche de génération
  . GBOS16-CC
  . GIO16-SY
  . IOCS16-SY
  . GFMS16-SY
  . Les modules link-éditables (MOL) de BOS16
  . Tous les drivers nécessaires, isolés ou dans des bibliothèques
  . La bibliothèque FMS16-:S

B) Lancement de la génération

- Emettre la commande :

  EOJ “cr”

- Lancement de GBOS16. Emettre les commandes :

  SI nom du fichier support des macros de GBOS16 “cr”

  CC GBOS16-CC,E2 “cr”

- Lecture des macros de GBOS16 création et lancement du fichier de commandes.

  La génération se fait automatiquement.

- Impression du message :

  END OF GENERATION

- Implantation du système dans la FU bootstrap [DI).
  Emettre la commande :

  GSYS n,nomsys,adk “cr”

- Lancement du système qui vient d’être généré.
  Emettre la commande :

  INIT n,D1

C) Erreurs détectées au cours de la génération

- Erreurs détectées par BOS-G. Elles sont de la forme :

  ERB XX 'XXXX

  Se reporter au manuel MEMO16 (paragraphe 1.2).

- Phase de lecture des macros de GBOS16

. Erreurs détectées par MACP

Elles sont de la forme :

ERM n
Macro-instruction erronée

Se reporter au manuel MEMO16

. Erreurs détectées par GBOS16

Impression des messages :

ILLEGAL PARAMETER *

TOO MUCH DRIVERS
TOO MUCH LIBRARIES

Corriger les macros de GBOS16 et recommencer la génération.

- Phase de création de IOCS16

. Erreurs détectées par MACP (cf. ci-dessus)

. Erreurs détectées par GIO16 (cf. Manuel utilisation IOCS16)

Corriger les macros de GIO16 et recommencer la génération.

- Erreurs détectées lors de l'assemblage de IOCS16

Elles sont de la forme :

********ERA n Phrase erronée

Ces messages sanctionnent une erreur, dans les macros de GIO16, qui n'a pas pu être détectée par GIO16.

Corriger les macros de GIO16 et recommencer la génération.

- Phase de lecture des macros de GFMS16

. Erreurs détectées par MACP (cf. ci-dessus)

. Erreurs détectées par GFMS16 (cf. Manuel d'utilisation FMS16)

Corriger les macros de GFMS16 et recommencer la génération.

- Phase de création de l'image mémoire

Les messages WARN 15 émis par LKLOAD sont normaux jusqu'à la fin de l'édition de liens.

Par contre, à la fin de l'édition de liens, il ne doit plus y avoir de symbole non défini. GBOS16 imprime sur le périphérique de dialogue la liste des symboles non définis.

S'il en existe, la génération doit être recommencée en rajoutant une macro %BIBDRV ou %DRV dans GBOS16.

Exemple : les symboles non définis imprimés sont :

DRVCR
LOCCR

Le driver lecteur de cartes n'a pas été link-édité.

Le processeur LKLOAD ne doit détecter aucune erreur. Sinon, il faut recommencer la génération.

Les erreurs détectées sont de la forme :

ERK XX

7.4.5 CONFIGURATION DE L'APPLICATION

Après la génération et le passage sous le contrôle de BOS16, on dispose du noyau de base de BOS16 sans module optionnel.

La configuration de l'application consiste à intégrer ces modules optionnels. définir la carte mémoire, intégrer les tâches résidentes, et sauvegarder l'ensemble dans Dl.

- Si nécessaire, redéfinir l'espace de travail de FMS16 par la commande NPAV (cf. par. 5.1.3).

- Intégrer au système les modules optionnels nécessaires, au moyen de la

commande OPTION (cf. par. 5.1.4).

Remarques :

. Si la gestion des tâches non résidentes doit être intégrée (OPTION NRESID), elle le sera en premier.

. La commande SYST (cf. par. 5.1.6) permet de connaître l'encombrement exact du système.

- Définir la carte mémoire par la commande FORE (cf. par. 5.1.5).

- Alimenter dans la zone résidente les tâches de l'application (cf. TLOAD par. 5.1.7) et les processeurs permanents (cf. PRUN par. 5.1.8).

- Définir les unités symboliques U5 à UF éventuellement utilisées par les tâches.

- Sauvegarder l'application sur Dl au moyen de la commande CONF (cf. par. 5.1.11).

- L'application est configurée

  Si on a intégré l'option "Restart automatique" (commande OPTION START), démarrer l'application par la commande INIT (cf. par. 3.2.28).

  Eventuellement lancer les autres tâches par la commande IRUN (cf. par. 5.1.9).

7.5 SYNOPTIQUE DE BOS-G

7.5.1 LISTE DES COMMANDES DE BOS-G

Ce sont les mêmes que celles de BOS16 sauf :

- FLOAT
- TLOAD
- IRUN
- PRUN
- MEMORY
- LOAD
- START
- POFF
- PASS
- MAC
- FUCC
- AVOU

7.5.2 LISTE DES MACROS DE GB0S16

```
%FUGENE?
%BUILD ? [FAST] [BRESID]
%MACRO IOCS ON ? FMS ON ?
%BIBDRV ON ?
%DRV ON ?
%FIN
%STANDARD [FAST] [BRESID]
%BOSR [NOBOS16]
```

7.5.3 CONTENU DE LA CARTOUCHE DE GENERATION

- Fichiers nécessaires à BOS-G

BOSG-:S	support du système BOS-C
BUILD-:S	processeur BUILD
CONFIG-:S	processeur CONFIG
GBOS16-CC	

- Fichiers image-mémoire des processeurs (catalogue :S)

BUILD-:S
LKLOAD
EDILE-:S
ASM-:S
MACP-:S
FUP2-:S
etc...

- Fichiers supports des bibliothèques

BSYS16-:S bibliothèque système de BOS16
etc...

- Fichiers nécessaires à la création de IOCS16

GIO16-SY
IOCS16-SY noyau IOCS16

- Fichiers nécessaires à la création de FMS16

GFMS16-SY
FMS16- / S

- Fichiers nécessaires à l'édition de liens de BOS16

BDRV16- :S bibliothèque des drivers standards
BVOL16- :S bibliothèque relative à la gestion d'espaces
MOLB16-MO modules link-éditables de BOS16
autres bibliothèques ou drivers non standards.

- Fichiers nécessaires à la génération d'autres systèmes.

8 ANNEXE 1 : DISQUES MOYENNE CAPACITE

8.1 CARACTERISTIQUES TECHNIQUES DES DISQUES DE MOYENNE CAPACITE

Un coupleur peut contrôler jusqu'à 4 unités de disque.

Capacité formatée d'un disque :

- 50 Méga-octets répartis sur 20 surfaces de disque
- 406 cylindres de 20 pistes
- 24 secteurs par piste (480 secteurs par cylindre)
- 128 mots par secteur (plus deux mots de contrôle)

Cadence d'échange : 156 Kmots/seconde

Temps d'accès :
. Cylindre à cylindre 10 ms
. Moyen 35 ms
. Maximum 70 ms

Temps d'accès moyen a un secteur : 47,5 ms

Le constructeur assure 400 cylindres valides sur 406.

Au niveau du coupleur on peut multiplexer les déplacements de bras sur les 4 unités, mais en cours d'échange, il est interdit de lancer un échange ou un déplacement de bras sur une autre unité.

8.2 FORMATAGE DES DISQUES DE MOYENNE CAPACITE

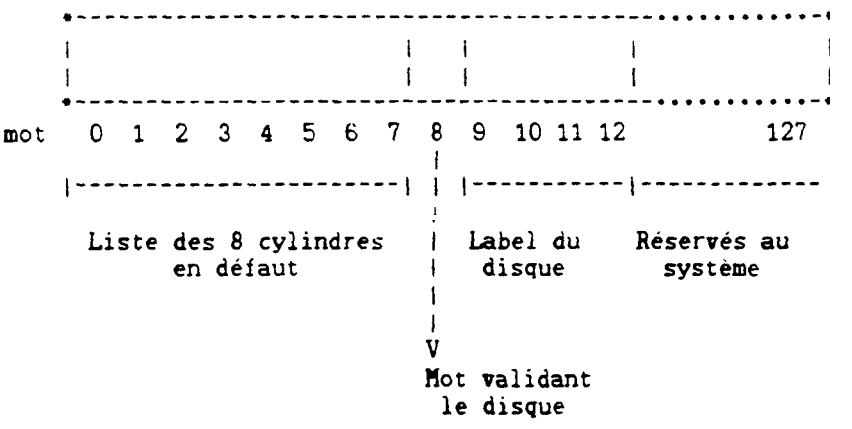
Le programme de formatage des disques de moyenne capacité initialise chaque secteur du disque.

Il permet :

- de découvrir les cylindres défectueux (un cylindre est déclaré défectueux si l'un au moins de ses 480 secteurs est en défaut).
- de donner un nom (label) au disque. Le label est constitué de 7 caractères au plus suivis de "retour chariot".

Ces informations sont rangées dans le secteur 3 du disque.

Structure du secteur 3 :



Remarque : Le cylindre 0 (contenant le secteur 3) d'un disque de moyenne capacité ne peut pas appartenir à une FU gérée par FMS16.

8.3 GENERATION DE BOS16 SUR DISQUE DE MOYENNE CAPACITE

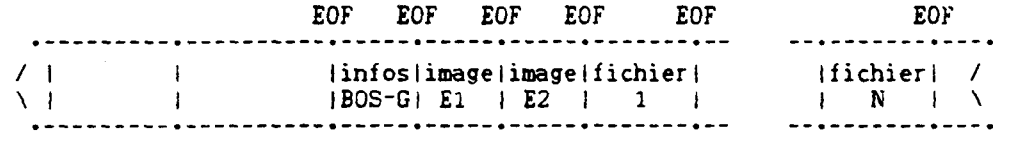
La livraison du logiciel s'effectue sur bande magnétique.

8.3.1 MISE EN OEUVRE DE BOS-G

Une bande magnétique (800 ou 1600 BPI) contient :

- un utilitaire bootstrapable
- l'image des unités fonctionnelles E1 (1 cylindre) et E2 (1 cylindre) supportant BOS-G
- l'ensemble des fichiers nécessaires à la génération de systèmes, dans un format utilisable par l'utilitaire FUP3.

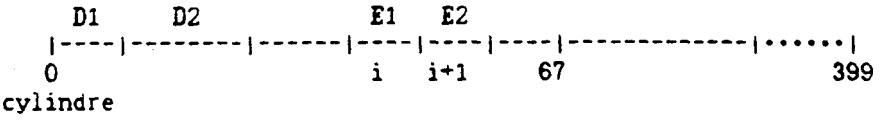
Structure de la bande magnétique BOS-G :



L'utilitaire bootstrapable permet l'implantation des FU E1 et E2 sur les 2 cylindres choisis par l'utilisateur.

Remarque : E1 et E2 doivent être implantés avant le cylindre 68, et après la zone disque qui sera réservée à D1 et D2 par CONFIG.

Structure du disque de moyenne capacité supportant le système :



Mise en oeuvre :

- Mettre le sélecteur de bootstrap sur MTU

- Monter la bande magnétique BOS-G, sur l'unité 0

- Actionner les touches

STOP - INITIALIZE - LOAD - RUN

Chargement de l'utilitaire de transfert et impression de •. L'utilitaire de transfert est en attente de commandes.

On dispose alors sous l'utilitaire des commandes :

- DEFU pour configurer le disque et la bande magnétique
- TRANS pour initialiser les FU E1 et E2 supports de BOS-G.

8.3.2 MISE A JOUR DE L'UTILITAIRE DE TRANSFERT

En plus de la console de service (FU TK et TS), l'utilitaire reconnaît les périphériques suivants :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP
Disque de moyenne capacité	14	1	'30	HDC	1	0
Bande magnétique 800 bpi	14	2	'18	HDC	2	0

La commande DEFU permet de mettre à jour l'utilitaire lorsque les caractéristiques du disque et de la bande magnétique ne correspondent pas au tableau ci -dessus.

Emettre la commande :

DEFU {DI I MT} "cr"

L'utilitaire pose alors les questions :

ADRESSE ? réponse "cr" (adresse du périphérique désigné dans la commande - adresse ON ou OFF RACK)

SOUS NIVEAU EXCEPTION ? réponse "cr" (0 à 8)

NUMERO D'IT NORMALE ? réponse "cr" (0 à 8)

MODE CANAL ? réponse "cr" (LDC, MDC, HDC)

NUMERO DE PROCESSEUR E/S ? réponse "Cr" (0 à 3)

et seulement pour MT :

DENSITE (800 ou 1600) ? réponse "cr" (800 ou 1600)

8.3.3 INITIALISATION DES UNITES FONCTIONNELLES E1 ET E2 SUPPORTANT BOS-G

L'initialisation de ces unités fonctionnelles s'opère de la manière suivante :

- Emettre la commande :

TRANS "cr"

Impression de :

NUMERO DU CYLINDRE D'IMPLANTATION DE BOS-G ? réponse "cr'

Ce numéro correspond au numéro i de début de El. Il doit être inférieur à 67.

Impression de :

FIN E2 : CYL xx

- Transfert de la bande magnétique sur E1 et E2

- Mise à jour de RAPD, qui connait désormais BOS-G sous les numéros de système 0 et 7

- Mettre le sélecteur de bootstrap sur MHD

- Actionner les touches

STOP - INITIALIZE - LOAD - RUN

- Impression de •

BOS-G est en attente de commande

On dispose alors sur E2 des processeurs :

- CONFIG pour la mise à jour de IOCS16,
- FUP3 pour charger sur disque les fichiers nécessaires à la génération,
- FUP4 pour structurer le disque et initialiser la FU D2.

8.3.4 MISE A JOUR DE BOS-G

En plus des unités fonctionnelles TK, TS, CR, LP décrits au paragraphe 7.3, BOS-G disque de moyenne capacité reconnaît les périphériques suivants :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP	FU
Disque de moyenne capacité	14	1	'30	HDC ou MDC	1	0 ou 1	D1,D2 E1,E2
Unité 0 de bande magnétique	14	2	'18	HDC	2		T1

Le processeur CONFIG permet de mettre à jour IOCS16. Son activation s'effectue de la façon suivante :

```
*LO ZE
*JOB CONFIG,,E2
*RUN CONFIG- : S
```

La modification des caractéristiques de la bande magnétique peut être effectuée par la commande de CONFIG :

DEF MT

CONFIG utilise le même dialogue que la commande DEFU vue précédemment.

La définition des unités fonctionnelles D1 et D2 s'opère de la manière suivante :

Emettre la commande :

FUDISK

CONFIG pose les questions :

TAILLE D1 ? réponse "cr"

La réponse représente un nombre de cylindres

TAILLE D2 ? réponse "cr"

La FU D1 commence en début de disque. La FU D2 est consécutive à D1. Sa taille sera telle que D2 ne chevauche pas E1 et E2.

Validation de toutes les commandes précédentes. Emettre la commande :

CONF "Cr"

Après cette commande, les suppressions et modifications de périphériques, ainsi que les adresses de D1 et D2 sont mémorisées définitivement par BOS-G. En particulier, lors d'un lancement ultérieur de BOS-G, il ne sera plus nécessaire d'appeler CONFIG.

BOS-G est opérationnel. La génération de BOS16 consistera à créer, dans la FU D2 actuellement vide, les fichiers suivants :

- processeurs en image mémoire,
- bibliothèques,
- fichier système supportant BOS16.

8.3.5 INITIALISATION DU DISQUE SYSTEME

La génération d'un système BOS16 nécessite l'initialisation du disque qui sera utilisé par la suite en tant que disque système.

Cette initialisation est réalisée au moyen des commandes IDEF et SDEF de FUP4 :

```
*RUN FUP4-:S
*IDEF,E3,[label],2
*SDEF,E3,[label],1,0,taille          D1
*SDEF,E3,[label],2,début      D2,taille      D2
*FUINI,D2,[tg]
```

avec tg = taille du granule (16 secteurs par défaut).

Note :

Lorsque la FU D2 contient déjà des informations qu'il ne faut pas détruire, les commandes de FUP4 seront :

```
*  IDEF,E3,[label].2
*  SDEF,E3,[label],1,O,taille  D1
*  SDEF,E3,[labe1],2,début  D2,taille  D2,ltag
```

avec ltag = (nombre de granules + 15)/16.

On ne fera pas de commande FUINI.

8.3.6 INTEGRATION DES PROCESSEURS ET BIBLIOTHEQUES

La bande magnétique supportant BOS-G supporte également les processeurs, bibliothèques et générateurs associés à BOS16.

Leur intégration dans l'unité fonctionnelle D2 s'effectue au moyen des commandes :

```
*LO ZE
*RUN FUP3-:S
*REST,T1,D2
```

La liste des fichiers supportés par la bande magnétique est fournie avec la bande.

8.3.7 GENERATION DE BOS16

La préparation des jeux de macro-instructions nécessaires à la génération de BOS16 est décrite au paragraphe 7.4.

9 ANNEXE 2 : DISQUES SOUPLES

9.1 CARACTERISTIQUES TECHNIQUES DES DISQUES SOUPLES

Un coupleur peut contrôler deux unités de disques souples.

Une disquette peut être simple ou double face, simple ou double densité.

La taille du secteur peut être 128 ou 256 octets.

BOS16 assure la gestion de fichiers sur deux types de disquettes dont les capacités sont les suivantes :

- 243,75 Koctets par disquette simple face, simple densité avec secteurs de 128 octets (gestion par FUP13),

  . 75 cylindres par disquette,
  . 1 piste par cylindre,
  . 26 secteurs par piste,
  . 128 octets par secteur,

- 971,75 Koctets par disquette double face, double densité avec secteurs de 256 octets (gestion par FMS et FUP),

  . 75 cylindres par disquette,
  . 2 pistes par cylindre,
  . 26 secteurs par piste,
  . 256 octets par secteur (à l'exception de la piste 0, face 0 dont les secteurs comportent 128 octets par secteur).

Les disquettes comportent deux cylindres de réserve qui sont utilises lorsque le programme de formatage a détecté une ou deux pistes en défaut.

Cadence de transfert : 31 250 mots/s.

Temps d'accès à une piste :

. déplacement d'une piste : 3 ms + 15 ms de stabilisation
. déplacement moyen : 111 ms
. déplacement maximum : 228 ms.

Temps d'accès moyen en rotation : 83,5 ms.

9.2 FORMATAGE DES DISQUES SOUPLES

L'opération de formatage d'un disque souple consiste à :

- écrire en tête de chaque secteur du disque le "header" du secteur,
- vérifier la validité de chaque cylindre en opérant un décalage du cylindre si celui-ci est en défaut,
- donner un nom appelé "label" au disque souple.

Le programme de formatage est lancé par la commande :

CALL FORFDD "cr"

Il est conversationnel. Les questions posées et les réponses à fournir sont décrites dans le Manuel d'exploitation du disque souple.

9.3 INITIALISATION DES DISQUES SOUPLES GERES PAR FMS16

Les disques souples double face, double densité avec secteurs de 256 octets sont utilisables en tant que disque système ou en tant que support de données gérées par FMS16.

Dans le premier cas les secteurs 0, 1 et 2 des disques doivent supporter le programme de chargement en mémoire du système.

Dans les deux cas les secteurs 3 et 4 doivent contenir les informations nécessaires à la gestion de volume.

Le processeur INFDD permet de rendre exploitable ces disques souples en les initialisant.

L'appel en mémoire de INFDD est réalisé au moyen de la commande :

CALL INFDD "cr"

- La commande INVOL permet alors d'initialiser les cinq premiers secteurs du disque souple.

Forme :

INVOL,{SUIFU},label "cr"

où :

{SUIFU}	désigne l'unité fonctionnelle sur laquelle est monté le disque souple
label	est une suite de 1 à 7 caractères alphanumériques représentant le nom du disque souple.

- La commande CHRAPF permet de changer le bootstrap et éventuellement le label sans modification des autres informations supportées par le disque souple.

Forme :

CHRAPF,{SUIFU}[,label] "cr"

- Messages d'erreur émis par INFDD :

COMMANDE INCORRECTE	Erreur de syntaxe
CE N'EST PAS UN FLOPPY 2D. 2F. SECTEURS DE 128 MOTS	Le support ne correspond pas au standard

LES SECTEURS NE SONT PAS DE 128 MOTS	Le disque souple n'a pas été configuré avec des secteurs de 128 mots (FINDIC = 1 dans la macro-instruction %PUFDD correspondant à l'unité)
LA FU NE COMMENCE PAS AU SECTEUR 0	INFDD initialise les 5 premiers secteurs de la disquette. SU ou FU ne désigne pas un espace initial
ERREUR IOCS 'XXXX	Erreur détectée par IOCS16 en cours d'échange. L'information 'XXXX représente le mot d'état de l'unité physique

9.4 UTILISATION DES DISQUES SOUPLES

Le driver disque souple DRVFDD est capable de gérer plusieurs formats de disquettes :

- simple face et simple densité avec secteurs de 128, 256 ou 512 octets.
- double face et simple densité avec secteurs de 128, 256 ou 512 octets,
- double face et double densité avec secteurs de 256, 512 ou 1024 octets.

BOS16 peut assurer la gestion de fichiers sur les disquettes double face et double densité avec secteurs de 256 octets. les fichiers étant pris en compte par FMS16 (ce type de disque est utilisable en tant que disque système par BOS16).

Les autres types de disques souples -non formatés par la SEMS- ne sont gérés que par IOCS16.

L'information qu'ils supportent peut être exploitée par un programme utilisateur approprié.

9.5 GENERATION DE BOS16 SUR DISQUE SOUPLE

9.5.1 PRINCIPE

Un système d'exploitation BOSGFD est livré, prêt à fonctionner sur disque souple. La génération d'un système BOS16 adapté aux besoins de l'utilisateur se fait sur une configuration avec disque dur.

9.5.2 BOSGFD

Le disque souple BOSG comporte :

- le système BOSGFD,
- les fichiers image mémoire supportant les processeurs du logiciel de base indispensables pour la génération : EDIT16, MACP, ASM, LKLOAD, FUP2, FUP3. FUP4, INFDD, FORFDD et CONFIG.

9.5.2.1 Caractéristiques de BOSGFD

Le moniteur d'entrée-sortie intégré à BOSGFD est configuré pour les périphériques correspondant au tableau :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP	RACK	FU
Périphérique de dialogue	Indifférent		'17F8	programmé simple				TK,TS
		0		LDC	0	0	0,2,4,6	
		1			1	0	1,3,5,7	E1,D1
Disques	14	3	'28	MDC	3	0 à 3	0,2,4,6	D2,E3
Souples		4			4	0 à 3	1,3,5,7	D3,D4
		3		HDC	3	0	0,2,4,6	E2
		4			4	0	1,3,5,7	
Imprimante	14	Paramètres non définis						LP

Les unités fonctionnelles gérées par BOSGFD sur les disques souples sont :

- E1, D1, D2 sur l'unité 0

E1	FU initiale
D1	FU bootstrap
D2	FU système

- E3, D3, D4, E2 sur l'unité 1

E3	FU initiale
D3,D4,E2	FU utilisateur

Lors de son lancement BOSGFD s'adapte aux caractéristiques du coupleur disques souples.

Le processeur CONFIG permet de définir les caractéristiques de l'imprimante, les opérations possibles étant les suivantes :

- suppression si le périphérique n'appartient pas à la configuration,
- définition du numéro de processeur (IOP 16) sur lequel est connecté le périphérique,
- définition de l'adresse coupleur, du sous-niveau exception, du numéro d'IT normale et du mode canal.

L'imprimante ne peut toutefois être utilisée sous BOSGFD que si elle est gérée par le driver imprimante standard.

D'autre part, lorsque le périphérique de dialogue est un terminal imprimante 30 caractères/seconde (TER30) il est indispensable de l'indiquer au système au moyen de la commande TER30 du processeur CONFIG.

9.5.2.2 Particularités de BOSGFD

BOSGFD fonctionne indifféremment avec comme périphérique de dialogue un téléimprimeur, un terminal imprimante 30 caractères/seconde (TER30), une console de visualisation. Cela amène quelques particularités de fonctionnement.

- Le défaut "time out" sur téléimprimeur qui se traduit sous BOS16 par le message :

  ERB 05

  n'apparaîtra jamais.

- L'appel opérateur (touche BREAK) n'est effectif que pendant un échange sur le périphérique de dialogue.

- Après une commande EBOS. on ne peut pas relancer le dialogue par l'appel opérateur. Il faut rappeler BOS16 par le bootstrap :

STOP - INITIALIZE - LOAD - RUN

9.5.2.3 Mise en oeuvre de BOSGFD

Pour donner le contrôle à BOSGFD il faut :

- monter le disque souple BOSGFD sur l'unité 0,
- positionner le commutateur DEVICE sur FLD et actionner les clés :

STOP - INITIALIZE - LOAD - RUN

BOSGFD imprime :

```
SYSTEM 0
.
```

sur le périphérique de dialogue et passe en attente de commande.

9.5.2.4 Processeur CONFIG

Ce processeur est intégré dans la FU D2. Il n'est utile que dans la phase de mise a jour du système.

Cette phase se terminera après l'émission de la commande CONF qui rendra permanentes les modifications apportées.

Appel de CONFIG. Emettre les commandes :

```
.LO ZE
*RUN  CONFIG-:S
```

Lorsque le périphérique de dialogue est TER30 émettre la commande :

```
TER30
```

Suppression de l'imprimante absente de la configuration ou non gérée par le driver standard. Emettre la commande :

```
NO LP
```

Définition des caractéristiques de l'imprimante. Emettre la commande

```
DEF LP
```

CONFIG pose les questions :

ADRESSE ? réponse "cr" (adresse périphérique désigné dans la commande - adresse ON ou OFF RACK)

SOUS-NIVEAU EXCEPTION ? réponse "cr" (0 à 8)

NUMERO D'IT NORMALE ? réponse "cr" (0 à 8)

MODE CANAL ? réponse "cr" (LDC. MDC, HDC)

NUMERO DE PROCESSEUR E/S ? réponse "cr" (0 à 3)

Validation de toutes les commandes précédentes. Emettre la commande :

```
CONF
```

Après cette commande, les modifications sont mémorisées définitivement par BOSGFD. En particulier, lors d'un lancement ultérieur au système il ne sera plus nécessaire d'appeler CONFIG.

Le système est opérationnel.

9.5.3 GENERATION DE BOS16

La génération d'un BOS16 se fera sous un BOS16 disque dur sur une configuration disposant de disques souples.

Cette génération impose la préparation dans des fichiers disque de trois jeux de macro-instructions :

- macro-instructions de GIO16 décrivant le moniteur d'entrées-sorties IOCS16 du système à générer,
- macro-instructions de GFMS16 décrivant le système de gestion de fichiers FMS16.
- macro-instructions de GBOS16 précisant les différentes phases de la génération.

Remarque :

La préparation des jeux de macro-instructions nécessaires à la génération de BOS16 est décrite aux paragraphes 7.4.1, 7.4.2 et 7.4.3. La génération d'un système BOS16 adapté à la configuration du client nécessite l'initialisation du disque qui sera utilisé par la suite en tant que disque système.

Dans le cas du disque souple cette initialisation est réalisée par les processeurs INFDD et FUP4.

Le disque supportant le système BOS16 généré par le client doit comporter au moins trois espaces numérotés 0, 1 et 2, avec le dernier espace géré par FMS16.

Lorsque le système sera généré, le disque système du client sera monté sur l'unité 0. Les espaces devront alors être accessibles par l'intermédiaire d'une FU initiale, de D1 (espace 1) et D2 (espace 2).

9.6 DUPLICATION DES DISQUES SOUPLES

9.6.1 PRESENTATION

BOSRFL est un système autonome qui permet de dupliquer tout type de disque souple. Il est supporté par le disque souple BOSG.

La duplication nécessite que la disquette supportant la copie soit identique à la disquette dupliquée : même nombre de faces, même densité et taille au secteur identique.

Lors de la duplication tout secteur est dupliqué, y compris les secteurs délaissés. A la fin de l'opération les deux disquettes sont donc strictement identiques.

9.6.2 MISE EN OEUVRE

BOSRFL est implanté dans l'espace bootstrap (espace 1) de la disquette BOSG en tant que système 1.

Sa mise en oeuvre s'effectue de la façon suivante :

- Emettre sous BOS16 Standard la commande

  INIT 1 "cr"

  BOSRFL imprime alors "*" et passe en attente de commande.

- Monter le disque souple à dupliquer sur l'unité 0.

- Monter un disque souple de même type sur l'unité 1.

- Emettre la commande :

  DUPL "cr"

  L'ensemble du disque souple monté sur l'unité 0 est alors dupliqué.

  A la fin de la duplication BOSRFL imprime "*".

- Pour repasser sous le contrôle de BOS16 émettre la commande :

  EBOS "cr"

  puis monter une disquette BOS16 ou BOSG et lancer au pupitre le bootstrap disque souple.

10 ANNEXE 3 : DISQUES A INTERFACE SMD

10.1 CARACTERISTIQUES TECHNIQUES DES DISQUES A INTERFACE SMD

Un coupleur peut contrôler jusqu'à 4 unités de disques à interface SMD :

- disques CDD 27/55/83 comportant un support fixe et un support amovible (cartouche),
- disques RDD 300 ne comportant qu'un support amovible.

Les capacités de ces différents disques sont les suivantes :

- 13,86 Méga-octets pour les cartouches CDD.
- 13,86 Méga-octets, 41,58 Méga-octets et 69,30 Méga-octets respectivement pour les supports fixes CDD 27, CDD 55 et CDD 83.
- 263,34 Méga-octets pour les disques RDD 300.

répartis ainsi :

. 823 cylindres par support dont 15 cylindres de réserve,
. 1 piste par cylindre pour les cartouches CDD.
. 1, 3 et 5 pistes respectivement pour les supports fixes CDD 27. CDD 55 et CDD 83.
. 19 pistes par cylindre pour les disques RDD 300,
. 67 secteurs de 128 mots par piste.

La première piste de chaque support est réservé à un usage statistique. Elle est donc hors de l'espace accessible par l'utilisateur.

Les autres caractéristiques des disques sont les suivantes :

- cadence d'échange : 257,28 Kmots/seconde.
- temps d'accès :

. Cylindre à cylindre : 6 ms
. Moyen : 30 ms
. Maximum : 55 ms

- temps d'accès moyen à un secteur : 40 ms.

Le couplage des disques à interface SMD offre les possibilités suivantes :

- gestion de disques de capacités différentes,
- multiplexage des déplacements de bras sur les différentes unités,
- correction automatique d'erreurs (mécanisme ECC).
- récupération par positionnements paramétrés des erreurs non récupérables par ECC.

10.2 INITIALISATION DES DISQUES A INTERFACE SMD

Le programme de formatage des disques initialise chaque secteur ou disque. cette opération consistant à :

- écrire en tête de chaque secteur le “header” du secteur,
- vérifier la validité de chaque piste en substituant à toute piste invalide une piste de réserve.

De manière à pouvoir être pris en compte par les systèmes d’exploitation, les disques doivent comporter un bootstrap et contenir les informations nécessaires à la gestion de volume (structure initiale). Le processeur INSMD permet de rendre exploitables les disques à interface SMD.

Dans le cas des disques CDD le support fixe et la cartouche doivent être initialisés. L’utilisateur peut préciser si la partie qu’il initialise est destinée à supporter le système.

L’appel en mémoire de INSMD est réalisé au moyen de la commande :

```
CALL INSMD "cr"
```

- La commande INVOL permet d’initialiser le volume.

Forme :

```
INVOL,{SUIFU},label[,C]  “cr”
```

où :

(SUIFU} désigne la FU initiale associée au volume à initialiser.

INVOL vérifie qu’elle commence en début du support.

label est une suite de 1 à 7 caractères alphanumériques représentant le nom du disque.

C précise, lors du traitement des disques CDD, que le système sera supporté par la partie mobile du disque (cartouche).
Par défaut, le système sera bootstrapé sur la partie fixe du disque.

- La commande CHRAP permet de changer le bootstrap et éventuellement le label sans modification des autres informations supportées par le disque.

Forme :

```
CHRAP,{SUIFU}[,label]"cr"
```

où les paramètres ont la même signification que pour la commande INVOL.

- Messages d’erreur émis par INSMD :

ERREUR SYNTAXE COMMANDE
LABEL > 7 CARACTERES

SUPPORT PRECISE INCORRECT — Le volume à initialiser n’est pas un disque à interface SMD

SU/FU INCORRECTE — La SU/FU spécifiée n’est pas une FU disque ou ne commence pas au cylindre 0

DEFAUT LECTURE — Défaut détecté par IOCS16 lors d’une lecture du disque (le message comporte le mot d’état de l’unité physique)

DEFAUT ECRITURE	Défaut détecté par IOCS16 lors d'une écriture sur disque (le message comporte le mot d'état de l'unité physique)

10.3 GENERATION DE BOS16 SUR DISQUE A INTERFACE SMD

La livraison du logiciel s'effectue sur bande magnétique dans le cas du disque RDD 300.

Elle peut s'effectuer sur bande magnétique ou sur cartouche dans le cas des disques CDD.

10.3.1 LIVRAISON SUR CARTOUCHE

Un système d'exploitation disque, BOS-G, est livré prêt à fonctionner sur cartouche.

La mise en oeuvre de BOS-G, son utilisation ainsi que la génération d'un système BOS16 adapté à la configuration du client sont identiques à la description du chapitre 7 (Génération de BOS16 sur disque à tête mobile).

Il existe cependant quelques différences par rapport à la génération décrite :

- pour mettre en oeuvre BOS-G sur disque à interface SMD il est nécessaire de mettre le sélecteur de bootstrap sur FHD (et non MHD comme cela est spécifié au paragraphe 7.2).

- les périphériques gérés par BOS-G sont configurés conformément au tableau du paragraphe 7.3.1 en substituant au disque à tête mobile le disque CDD correspondant au tableau suivant :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP	FU
Disque CDD	14	0	'38	HDC	0	0 ou 1	D1 à DA DF,E1,E2

- l'initialisation du disque système décrite au paragraphe 7.3.4 nécessite avant l'utilisation de FUP4 celle du processeur INSMD :

```
LO ZE
RUN  INSMD-:S
INVOL, DA, label
DMONT DA
RUN  FUP4-:S
SDEF,DA,,1,0,taille  D1            Définition des espaces
SDEF,DA,,2,débutD2,longueurD2      correspondant à D1 et D2
MONT DA
FUINI,DA,tg,nbg                    Initialisation de D2
MONT DA
```

10.3.2 LIVRAISON SUR BANDE MAGNETIOUE

10.3.2.1 Mise en oeuvre de BOS-G

Une bande magnétique (800 ou 1600 BPI) contient :

- un utilitaire bootstrapable.
- l'image des unités fonctionnelles E1 et E2 supportant BOS-G,
- l'ensemble des fichiers nécessaires à la génération de systèmes, dans un format utilisable par l'utilitaire FUP3.

Sa structure est identique à celle utilisée pour les disques de moyenne capacité (cf. par. 8.3.1).

L'utilitaire bootstrapable permet l'implantation des FU E1 et E2 sur les cylindres choisis par l'utilisateur.

Mise en oeuvre :

- Mettre le sélecteur de bootstrap sur MTU

- Monter la bande magnétique BOS-G sur l'unité 0

- Actionner les touches

STOP - INITIALIZE - LOAD - RUN

Chargement de l'utilitaire de transfert

Impression de •

L'utilitaire de transfert est en attente de commande

On dispose alors sous l'utilitaire des commandes :

- DEFU pour configurer le disque et la bande magnétique,
- INVOL pour initialiser le disque.
- TRANS pour initialiser les FU E1 et E2 supports de BOS-G.

10.3.2.2 Mise à jour de l'utilitaire de transfert

En plus de la console de service (FU TK, TS). l'utilitaire reconnaît les périphériques suivants :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP
Disque SMD	14	0	'38	HDC	0	0
Bande magnétique 800bpi	14	2	'18	HDC	2	0

La commande DEFU permet de mettre à jour l'utilitaire lorsque les caractéristiques du disque et de la bande magnétique ne correspondent pas au tableau ci-dessus.

- Emettre la commande :

DEFU {DI I MT) "cr"

- L'utilitaire pose les questions :

ADRESSE ? réponse "cr" (adresse du périphérique désigné dans la commande - adresse ON ou OFF RACK)

SOUS NIVEAU EXCEPTION ? réponse "cr" (0 à 8)

NUMERO D'IT NORMALE ? réponse "cr" (0 à 8)

MODE CANAL ? réponse "cr" (LDC, MDC, MDC)

NUMERO DE PROCESSEUR E/S ? réponse "cr" (0 à 3)

et seulement pour MT

DENSITE (800 ou 1600) ? réponse "cr" (800 OU 1600)

- Impression de •

10.3.2.3 Initialisation des unités fonctionnelles E1 et E2 supportant BOS-G

La commande INVOL permet d'initialiser le disque qui supportera E1 et E2 (disque RDD 300 ou cartouche des disques CDD).

INVOL,D1,label,C "CR"

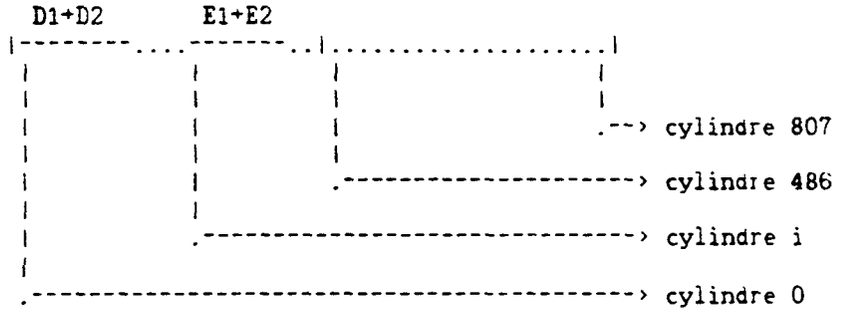
label étant le nom du disque (1 à 7 caractères alphanumériques).

La commande TRANS permet ensuite d'implanter BOS-G dans les unités fonctionnelles E1 et E2.

Dans le cas du disque RDD 300, E1 occupe un cylindre, et doit être implantée avant le cylindre 25 et après la zone disque réservée à D1 et D2.

Dans le cas des disques CDD l'utilisateur peut générer BOS16 soit sur la cartouche, soit sur la partie fixe.

E1 occupe quatre cylindres, doit être implantée avant le cylindre 486 et après la zone disque réservée à D1 et D2 lorsque l'utilisateur génère BOS16 sur la cartouche.



L'initialisation des unités fonctionnelles E1 et E2 s'opère de la manière suivante :

- Emettre la commande :

TRANS "cr"

Impression de

NUMERO DU CYLINDRE D'IMPLANTATION DE BOS-G ? réponse "cr"

Ce numéro correspond au numéro i de début de E1. Il doit être inférieur à 25 (RDD 300) ou 486 (CDD).

- Impression de

FIN E2 : CYL xx

- Transfert de la bande magnétique sur E1 et E2

Mise à jour de RAPD, qui connait désormais BOS-G sous les numéros de système 0 et 7.

- Mettre le sélecteur de bootstrap sur FHD

- Actionner les touches

STOP - INITIALIZE - LOAD - RUN

Impression de •

BOS-G est en attente de commande

On dispose alors sur E2 des processeurs :

- CONFIG pour la mise à jour de IOCS16,
- FUP3 pour charger sur disque les fichiers nécessaires à la génération.
- FUP4 pour structurer le disque et initialiser la FU D2.
- INSMD pour initialiser la partie fixe des disques CDD.

10.3.2.4 Mise à jour de BOS-G

Le moniteur d'entrée-sortie IOCS16 intégré à BOS-G est configuré avec les périphériques décrits au paragraphe 10.3.1.

Le processeur CONFIG permet de configurer la bande magnétique, les disques souples et l'imprimante. Son activation s'effectue de la façon suivante :

```
LO ZE
RUN CONFIG- : S
```

La définition des caractéristiques d'un périphérique peut être effectuée par la commande de CONFIG :

- DEF MT pour la bande magnétique
- DEF LP pour l'imprimante
- DEF FD pour les disques souples

CONFIG utilise le même dialogue que la commande DEFU vue précédemment en 10.3.2.2. La validation des modifications précédentes est réalisée au moyen de la commande :

```
CONF
```

10.3.2.5 Initialisation du disque système

La génération d'un BOS16 adapté à la configuration du client nécessite l'initialisation des FU système D1 et D2.

Cette initialisation est différente selon que le disque système est le plateau fixe d'un disque CDD. la cartouche d'un disque CDD ou un disk-pack RDD 300.

A) Plateau fixe CDD

La commande FIXE du processeur CONFIG permet d'indiquer à BOS-G que les FU D1 et D2 sont supportées par le plateau fixe :

```
LO ZE
RUN CONFIG-:S
FIXE
```

BOS-G connait alors trois FU sur le plateau fixe : DA (FU initiale), D1 et D2.

L'initialisation du disque est réalisée par les processeurs INSMD et FUP4 :

```
RUN INSMD-:S
INVOL,DA,label
DMON DA
RUN FUP4-:S
SDEF,DA,,1,0,taille D1          Definition des espaces
SDEF,DA,,2,début D2,taille D2   correspondant à D1 et D2
MONT DA
FUINI,D2,tg,___                 Initialisation de D2
MONT DA
```

Pour plus de précision consulter la notice FUP.

B) Cartouche CDD ou disque RDD 300

BOS-G connait cinq FU : DF (FU initiale), D1, D2, E1 et E2.

L'initialisation de D1 et D2 est réalisée par le processeur FUP4.

```
LO ZE
RUN  FUP4-:S
SDEF,DF,,1,0,taille  D1
SDEF,DF,,2,debut  D2,taille  D2
INIT
LO ZE
RUN  FUP4-:S
FUINI,D2[,tg]
INIT
```

10.3.2.6 Intégration des processeurs et bibliothèques

La bande magnétique supportant BOS-G supporte également les processeurs, bibliothèques et générateurs associés à BOS16.

Leur intégration dans l'unité fonctionnelle D2 s'effectue au moyen des commandes :

```
LO ZE
RUN  FUP3-:S
REST,T1,D2
```

La liste des fichiers supportés par la bande magnétique est fournie avec la bande.

10.3.2.7 Génération de BOS16

La préparation des jeux de macro-instructions nécessaires à la génération de BOS16 est décrite au paragraphe 7.4.

11 ANNEXE 4 : DISQUES WINCHESTER

11.1 CARACTERISTIQUES TECHNIQUES DES DISQUES WINCHESTER

Un coupleur permet la connexion de 1 à 3 unités de disques par l'intermédiaire d'un contrôleur.

Cet ensemble est toujours accompagné d'un streamer (dérouleur de cassettes) ou d'un disque souple permettant la sauvegarde/restitution des informations supportées par les disques connectés au même contrôleur.

Les disques ont, suivant le modèle, une capacité formatée de 16,8 (DWB20), 33,6 (DWB40), 58,3 (DWB50-0 ou DWB50), 56,5 (DWB50-1), 20.1 (disque dur DWF20) ou 1,06 (disque souple DWF20) Méga-octets répartis ainsi :

- disques 8 pouces DWB20 et DWB40 :

    - 500 cylindres utiles par support, plus un cylindre (de numéro 500) réservé aux programmes de tests et de formatage et 12 cylindres (de numéros 501 à 512) de remplacement,
    - 4 (DWB20) ou 8 (DWB40) pistes par cylindre,
    - 32 secteurs de 128 mots par piste,

- disques 5 pouces 1/4 DWB50-0 ou DWB50 :

    - 976 cylindres utiles par support, plus un cylindre (de numéro 976) réservé aux programmes de formatage et de tests et 10 cylindres (de numéro 977 à 986) de remplacement,
    - 7 pistes par cylindre,
    - 33 secteurs de 128 mots par piste,

- disques 5 pouces 1/4 DWB50-1 :

    - 976 cylindres utiles par support, plus un cylindre (de numéro 976) réservé aux programmes de formatage et de tests et 10 cylindres (de numéro 977 à 986) de remplacement,
    - 7 pistes par cylindre,
    - 32 secteurs de 128 mots par piste,

- disques durs 5 pouces 1/4 DWF20 :

    - 604 cylindres utiles par support, plus un cylindre (de numéro 604) réservé aux programmes de formatage et de tests et 10 cylindres (de numéro 605 à 614) de remplacement,
    - 4 pistes par cylindre,
    - 32 secteurs de 128 mots par piste,

- disques souples 5 pouces 1/4 haute densité DWF20 :

    - 80 cylindres utiles, pas de cylindre de remplacement,
    - 2 pistes par cylindre,
    - 26 secteurs de 128 mots par piste.

Remarques : les disques dur et souple DWF20 forment un ensemble où la disquette 5 pouces 1/4 peut être utilisée soit en tant que support de sauvegarde/restitution du disque dur, soit en tant que support de données. De plus, le terme "disque dur" ne sera employé que pour l'ensemble DWF20, ce dernier étant le seul à comporter également une unité disque souple.

Les autres caractéristiques des disques sont les suivantes :

- disques DWB20 et DWB40 :

  . Vitesse de transfert des données : 136,5 Koctets/seconde

  . Temps d'accès :

    . Piste à piste : 15 ms
    . Maximum : 100 ms
    . Moyen : 50 ms

  . Temps d'accès aux données : 10 ms

- disques DWB50-0 ou DWB50 :

  . Vitesse de transfert des données : 169 Koctets/seconde

  . Temps d'accès :

    . Piste à piste : 5ms
    . Maximum : 60ms
    . Moyen : 30ms

  . Temps d'accès aux données : 8,3ms.

- disques DWB50-1 :

  . Vitesse de transfert des données : 490 Koctets/seconde

  . Temps d'accès :

    . Piste à piste : 5 ms
    . Maximum : 60 ms
    . M o y e n : 30 ms

  . Temps d'accès aux données : 8,3 ms

- disques durs DWF20 :

  . Vitesse de transfert des données : 490 Koctets/seconde

  . Temps d'accès :

    . Piste à piste : 20 ms
    . Maximum : 150 ms
    . Moyen : 65 ms

  . Temps d'accès aux données : 8,3 ms

- disques souples DWF20 :

  . Vitesse de transfert des données : 40 Koctets/seconde

  . Temps d'accès piste à piste : 4 ms

  . Temps d'accès aux données : 83,3 ms

11.2 UTILISATION PAR LE LOGICIEL DES DISQUES WINCHESTER

- UTILISATION DES DISQUES 8" DWB20 ET DWB40

De façon à pouvoir s'adapter facilement au modèle de disque (DWB20 ou DWB40), le logiciel gère un cylindre logique de 4 pistes, soit 128 secteurs.

Par la suite, le terme "cylindre" devra être interprété comme "cylindre logique", ceci quel que soit le modèle de disque.

Le nombre de cylindres logiques accessibles est ainsi de 500 pour les disques DWB20 et 1000 pour les disques DWB40.

Pour être pris en compte par le logiciel. les disques doivent être initialisés et structurés. Ces opérations seront décrites dans les paragraphes qui suivent.

REMARQUE IMPORTANTE :

> Les espaces qui seront définis par l'utilisateur sur les disques DWB40 doivent avoir des adresses paires (exprimées en cylindres logiques). A cette seule condition, il sera possible d'utiliser BOS-R pour réaliser la sauvegarde d'un disque et la restitution des informations sur un disque de capacité différente.

- UTILISATION DE TOUS LES DISQUES (8" ET 5" 1/4)

Pour être pris en compte par le logiciel, les disques doivent être initialisés et structurés. Ces opérations seront décrites dans les paragraphes qui suivent.

11.3 INITIALISATION DES DISQUES WINCHESTER

Le programme de formatage des disques initialise chaque secteur du disque, cette opération consistant à :

- écrire en tête de chaque secteur le “header” du secteur
- vérifier la validité de chaque piste en substituant à toute piste invalide une piste de remplacement.

De manière à pouvoir être pris en compte par les systèmes d’exploitation, les disques doivent comporter un bootstrap et contenir les informations nécessaires à la gestion de volume (structure initiale).

Le processeur INSAS permet de rendre exploitable les disques Winchester.

L’appel en mémoire de INSAS est réalisé au moyen de la commande :

```
CALL INSAS "cr"
```

- la commande INVOL permet d’initialiser le volume.

Forme :

```
INVOL,{SU|FU},label[,num]"cr"
```

où :

{SUIFU} désigne la FU initiale associée au volume à initialiser

label est une suite de 1 à 7 caractères alphanumériques représentant le nom du disque

num permet d’indiquer la position du commutateur DEVICE utilisée pour lire le bootstrap du disque :
6 : FLD (adresse coupleur ‘28)
7 : MHD (adresse coupleur ‘30)
8 : FHD (adresse coupleur ‘38)
Par défaut le numéro retenu est 6.

- la commande CHRAP permet de remplacer le bootstrap et éventuellement le label sans modification des autres informations supportées par le disque.

Forme :

```
CHRAP,{SU|FU}[,label][,num]"cr"
```

où les paramètres ont la même signification que pour la commande INVOL.

- messages d’erreur émis par le processeur :

ERREUR SYNTAXE COMMANDE
La syntaxe de la commande n’a pas été respectée.

DEFAUT LECTURE CRDU='XXXX
Lors d’une lecture disque, un défaut a été détecté par IOCS16, ‘XXXX est le compte-rendu fourni par IOCS16.

DEFAUT ECRITURE CRDU='XXXX
Lors d’une écriture disque, un défaut a été détecté par IOCS16, ‘XXXX est le compte-rendu fourni par IOCS16.

SUPPORT PRECISE INCORRECT
Le volume à initialiser ne correspond pas à un disque à interface SASI.

SUIFU INCORRECTE FU='00XX
La FU spécifiée n'est pas une FU disque ou ne commence pas en zéro.

LABEL > 7 CARACTERES

NUMERO SELECTION INCORRECT
Le numéro de sélection du bootstrap est différent de 6, 7 et 8.

11.4 GENERATION DE BOS16 SUR DISQUE WINCHESTER

11.4.1 PRINCIPE

Un système d'exploitation, BOS-G, est livré, prêt à fonctionner : sur une cassette pour les disques DWB20, DWB40, DWB50-0 et DWB50-1, sur une disquette pour les DWF20.
Le bootstrap permet, de transférer sur le disque 0 le contenu de la cassette (disques DWB20/40/50-0/50-1) et ensuite de lancer BOS-G ou de reconstituer le disque 0 à partir de la disquette (DWF20).

L'utilisateur peut alors générer un système BOS16 adapté à ses besoins.

11.4.2 MISE EN OEUVRE DE BOS-G DWB20, DWB40, DWB50

- Insérer la cassette BOS-G dans le streamer
- Mettre le sélecteur de bootstrap sur le numéro correspondant à l'adresse coupleur (FLD, MHD ou FHD).
- Actionner les touches :

  STOP-INITIALIZE-LOAD-RUN

- Après l'édition du message :

  FM-DM-KD-KM-MK ?

répondre KD

- Après l'édition de l'identification de la cassette et du message :

  OK ?

répondre Y

- En fin de transfert, après l'édition du message :

  ?

répondre G pour lancer BOS-G

- Edition du message :

  SYSTEM 0
  .

Le système BOS-G est en attente de commande.

- Insérer la disquette BOS-G dans l'unité disque souple

- Mettre le sélecteur de bootstrap sur le numéro correspondant à l'adresse coupleur (FLD, MHD ou FHD)

- Actionner les touches :

  STOP-INITIALIZE-LOAD-RUN

- Après l'édition du message :

  FM-DM-KD-KM-MK ?

répondre FM

- Après reconstitution automatique du BOS-G disque dur DWF20

  édition du message :

  /MSG * ** SI LE COUPLEUR N'EST PAS A L'ADRESSE STANDARD ('28)
  /MSG -----> EMETTRE : CHRAP,DA,,NUM (NUM = 7 OU 8)
  /PAUSE * SINON/PUIS : RETURN

- Effectuer la commande CHRAP demandée si nécessaire, puis frapper RETURN

- Editiondu message :

  SYSTEME 0
  *

Le système BOS-G disque dur DWF20 est en attente de commande.

11.4.4 MISE A JOUR DE BOS-G

Le moniteur d'entrée-sortie IOCS16 intégré à BOS-G est configuré avec les périphériques suivants :

PERIPHERIQUE	NIVEAU	SOUS-NIVEAU	ADRESSE COUPLEUR	MODE	ITN	IOP	RACK	FU
Périphérique de dialogue	Indifférent	'17F8	Programmé simple					TK, TS
Disque Winchester	14	3	'28	HDC ou MDC	3	0 à 3	0,2,4,6	Unité 0 : DF,D1,
		4			4		1,3,5,7	D2,D3, D4,D5, E1,E2, E3
								Unité 2 : E4,D6, D7
Bande magnétique 800bpi	14	2	'18	HDC	2	0	0	T1

Si le coupleur est à une adresse bootstrapable autre que '28 (position 6 du bootstrap), émettre les commandes (sauf pour les disques DWF20) :

LO ZE
RUN INSAS-:S
CHRAP,DF,,num (num = 7 ou 8)

Le processeur CONFIG permet de définir les caractéristiques de l'imprimante ou de la bande magnétique.

- Activation de CONFIG. Emettre les commandes :

```
LO ZE
RUN CONFIG-:S
```

- Suppression de l'imprimante absente de la configuration ou non gérée par le driver standard. Emettre la commande :

```
NO LP
```

- Définition des caractéristiques de l'imprimante ou de la bande magnétique. Emettre la commande :

```
DEF{LPIMT}
```

CONFIG pose alors les questions :

```
ADRESSE ? Réponse "cr" (adresse périphérique)
SOUS-NIVEAU EXCEPTION ? Réponse "cr" (0 à 8)
NUMERO D'IT NORMALE ? Réponse "cr" (0 à 8)
MODE CANAL ? Réponse "cr" (LDC, HDC, HDC)
NUMERO DE PROCESSEUR D'E/S ? Réponse "cr" (0 à 3)
```

et seulement pour HT :

```
DENSITE (800 OU 1600) ? Réponse "cr" (800 ou 1600)
```

- Validation. Emettre la commande :

```
CONF
```

Après cette commande les modifications sont mémorisées par BOS-G.

BOS-G est alors opérationnel.

11.4.5 INITIALISATION DU DISQUE SYSTEME

Le disque système est configuré avec 4 espaces accessibles par les FU DF, E1, E2 et E3 :

- DF (espace initial) occupant les cylindres 0 à 499 pour les DWB20 et DWB40, 0 à 975 pour le DWB50-0 et DWB50-1, 0 à 603 pour le DWF20.

- E1 et E2 (espaces 6 et 7) occupant les cylindres 250 à 263 sur DWB20, DWB40. DWF20, 139 à 146 sur DWB50-0 ou 143 à 150 sur DWB50-1.

- E3 (espace 8) occupant les cylindres 264 à 413 (150 cylindres) sur DWB20, DWB40, DWF20, 147 à 229 (83 cylindres) sur DWB50-0 ou 151 à 236 (86 cylindres) sur DWB50-1 et permettant à l'utilisateur la récupération du logiciel SEMS.

Lorsque BOS-G est supporté par un disque DUB40 émettre les commandes :

```
LO ZE
RUN FUP4-:S
SDEF,DF,,0,0,1000
```

Les espaces 1 à 5, accessibles par les FU D1 à D5, doivent être configurés par l'utilisateur en tenant compte des contraintes suivantes :

- D1 (espace 1) doit débuter au cylindre 0
- D2 (espace 2) doit être consécutif à D1
- la taille de D1+D2 doit être inférieure ou égale à l'adresse de début de El (250 pour DWB20, DWB40, DWF20, 139 pour DWB50-0 ou 143 pour DWB50-1). La taille de D1 est limitée à 32 cylindres pour les disques DWF20 qui seront sauvegardés par BOSR sur disquettes.

L'initialisation des espaces est réalisée par le processeur FUP4 :

- Définition des espaces correspondant à D1 et D2

```
LO ZE
RUN FUP4-:S
SDEF,DF,,1,0,taille  D1
SDEF,DF,,2,début  D2,taille  D2
INIT
```

- Initialisation de D2 (tg=taille des granules)

```
LO ZE
RUN FUP4-:S
FUINI,D2[,tg]
INIT
```

Rappel :

Les adresses et les longueurs des espaces sont exprimées en cylindres logiques.

11.4.6 INTEGRATION DES PROCESSEURS ET BIBLIOTHEQUES POUR DISQUES DWB20, DWB40, DWB50-0, DWB50-1

Le logiciel est livré sur cassette(s); le contenu d'une cassette est l'image d'une portion de disque de 150 cylindres sur DWB20, DWB40, 83 cylindres sur DWB50-0 ou 86 cylindres sur DWB50-1, gérée par FMS16 avec respectivement des granules de 16 secteurs (LTAG = 751, 33 secteurs (LTAG = 37) ou 16 secteurs (LTAG = 76).

Le transfert du contenu d'une cassette dans l'espace de livraison (E3) est réalisé de la manière suivante :

```
INIT 7
REST,, E3
INIT 0 (après la fin du transfert signalée par le message :
FONCTION OK)
```

L'intégration des processeurs et bibliothèques associés à BOS16 s'effectue au moyen des commandes :

```
JOB TRANS,,E3
LO ZE
RUN FUP3-:S
DUPL,PROC16-RP,E3,D2
EOJ
```

11.4.7 INTEGRATION DES PROCESSEURS ET BIBLIOTHEQUES POUR DISQUES DURS DWF20

Le logiciel est livré sur disquette(s). Chaque disquette comporte un espace initial et un espace 1 de 79 cylindres géré par FMS16 avec des granules de 26 secteurs (LTAG = 10).

Pour générer l'utilisateur aura le choix entre deux solutions : soit utiliser directement l'espace 1 de la disquette comme FU dite de génération (D6), soit transférer les fichiers des disquettes de livraison sur la FU E3 qui servira alors de FU de génération. Pour BOS-G DWF20 c'est une FU de 150 cylindres gérée par FMS16 avec des granules de 16 secteurs (soit un LTAG = 753.
Le transfert d'une disquette s'opère de la façon suivante :

```
JOB TRANS,,E2
LO ZE
MONT E4
RUN FUP3-:,S
DUPL,,D6,E3
EOJ
```

L'intégration des processeurs et bibliothèques associés à BOS16 s'effectue au moyen des commandes :

```
JOB  TRANS,,D6
LO ZE
MONT E4
RUN  FUP3-:,S
DUPL,PROC16-RP,D6,D2
EOJ
```

11.4.8 GENERATION DE BOS16

La préparation des jeux de macro-instructions, nécessaires à la génération de BOS16 est décrite au paragraphe 7.4.

11.4.9 INITIALISATION DES DISQUES DE DONNEES

Les disques de données doivent être initialisés par l'utilitaire INSAS et structurés par l'utilitaire FUP4.

11.5 RELIVRAISON DU LOGICIEL

11.5.1 DISQUES DWB20, DWB40, DWB50-0, DWB50-1

Toute relivraison de logiciel (mise à jour ou diffusion de nouveaux produits) s'effectue sur cassette.

Cette cassette est en fait gérée par BOS-R (cf. par. 6.2); elle supporte la sauvegarde d'un espace FMS16 de 150 cylindres (DWB20 et DWB40), de 83 cylindres (DWB50-0) ou 86 cylindres (DWB50-1).

Elle doit donc être restituée dans un espace désigné par l'utilisateur, de taille suffisante au moins 150 cylindres, avec des granules de 16 secteurs (LTAG>75 mots) pour les DWB20, DWB40, au moins 83 cylindres avec des granules de 33 secteurs (LTAG>37 mots) pour les DWB50-0. au moins 86 cylindres avec des granules de 16 secteurs (LTAG >= 76 mots) pour les DWB50-1.

Cet espace, appartenant au disque système ou à un disque de données, peut être utilisé par l'application. Il est cependant demandé à l'utilisateur, avant la réception d'un nouveau logiciel, d'en assurer la sauvegarde éventuelle au moyen de BOS-R.

11.5.2 DISQUES DURS DWF20

Toute relivraison de logiciel (mise à jour ou diffusion de nouveaux produits) s'effectue sur disquette de même format qu'une disquette de livraison (cf. par. 11.4.7). Le disque souple DWF20 devra donc être connu du système BOS16 de l'utilisateur en tant qu'unité 2 avec au moins deux FU [espace initial et espace 1), la seconde FU devra pouvoir gérer par FMS16 un espace de 79 cylindres avec des granules de 26 secteurs (nombre de granules NBG >= 158 donc LTAG >= 10 mots).

12 ANNEXE 5 : MACRO-COMMANDES

12.1 GENERALITES

L'utilisation de macro-commandes peut être envisagée quand on veut par un appel unique entraîner l'exécution d'une suite de commandes de BOS16. Cette suite peut être exécutée :

- soit séquentiellement.
- soit avec un enchaînement et une logique déterminés par la valeur de paramètres d'appel.

Après la définition d'une macro-commande, l'utilisateur peut en demander l'exécution en fournissant des valeurs pour paramétrer cette exécution.

Une macro-definition est une suite de lignes constituant le corps de la macro-commande. Elle peut comporter :

- des chaînes de caractères qui seront ultérieurement traitées comme des commandes,
- des directives,
- des appels de macro-commandes.

L'appel d'une macro-commande constitue une demande de génération (ou d'expansion) du contenu d'une macro-définition de même nom. Pendant cette génération, l'exécution de directives permet de définir des variables, de calculer des expressions, de réaliser des branchements, de dialoguer avec l'utilisateur.

La génération est réalisée dans un fichier temporaire auquel BOS16 associe implicitement l'unité symbolique CC. Toute macro doit donc se terminer par une commande qui provoque le retour au dialogue initial : RETURN ou EOJ.

12.1.1 DEFINITION DE MACRO-COMMANDES

La définition d'une macro-commande consiste à associer une suite de lignes appelées "corps" de la macro à un nom appelé "en-tête" de la macro.

Une macro-définition a la forme suivante :

• DEF en-tête
Corps
• ENDEF

L'en-tête contient la structure qu'il faudra respecter lors de l'appel. Cet appel utilisera simplement un nom ou associera des caractères à des paramètres formels, qui seront référencés dans le corps de la macro et remplacés au moment de l'expansion par des paramètres effectifs.

La directive • DEF réalise la mémorisation de l'en-tête, c'est-a-dire se la forme possible de l'appel, et du corps de la macro limite par la directive • ENDEF.

12.1.2 UTILISATION DE MACRO-COMMANDES

L'appel sera réalisé ultérieurement par utilisation de cet en-tête derrière le caractère "%" qui constitue un indicateur de macro.

Exemples :

1) Après la définition :

```
• DEFLISTE
/DUMP '1000,'2000
/RETURN
• ENDEF
```

l'appel %LISTE entraînera l'exécution de :

```
/DUMP '1000,'2000
/RETURN
```

2) Pour compiler tout programme PL, on peut créer la définition :

```
• DEFPL?
/CALL PL
/SI ?P1
/IPLC
/RETURN
• ENDEF
```

L'appel %PLSOURCE provoquera la substitution de la chaîne SOURCE à ?P1 :

```
/CALL PL
/SI SOURCE
/IPLC
/RETURN
```

12.2 DEFINITION DE MACRO-COMMANDES

12.2.1 DIRECTIVE DE DEFINITION

Syntaxe :

• DEFen-tête

L'en-tête est la chaîne de caractères apparaissant à la suite du nom de la directive.

Il peut comporter jusqu'à 9 identificateurs de paramètres représentés par le caractère "?" et des caractères imprimables, qui permettront, lors de l'appel, l'identification de la macro. L'ordre dans lequel ces éléments sont placés est essentiel, puisque c'est uniquement leurs positions qui permettent à 1 'exécution de les distinguer. Ils seront utilisés ultérieurement sous la forme ?Pn avec n évoluant de 1 à 9.

Exemple :

• DEFTRANSFERT?

peut être appelé respectivement par :

%TRANSFERTFIC1 ou %TRANSFERTART.FIC,U7

Remarque :

Une macro-définition peut ne comporter aucun "?", afin qu'à la génération aucun paramètre ne soit fourni. Dans ce cas, les séquences générées lors des différents appels sont invariantes ou peuvent être déterminées par la réponse a une directive de dialogue.

Redéfinition :

Une macro avec un même en-tête qu'une macro déjà définie remplace la définition précédente et l'annule donc.

12.2.2 DIRECTIVE DE FIN DE DEFINITION

Syntaxe :

• ENDEF

Cette directive finit la définition ouverte par •DEF.

12.2.3 CORPS DE MACRO-COMMANDE

Les éléments compris entre l'en-tête •DEF et la fin de définition •ENDEF sont des lignes qui peuvent être considérées comme /

- des directives si la ligne commence par le caractère "•",
- des lignes 0 mémoriser. A l'appel de la macro, elles seront considérées comme des commandes de BOS16.

La directive % d'appel de macro-commande peut apparaître dans le corps d'une macro-commande.

L'appel récursif à la macro en cours est possible.

Si elle commence par le caractère "\" (antislash), toute ligne est générée sans être interprétée avec suppression du "\".

Ainsi la ligne :

\•ENDEF

donne lieu à la génération de la ligne

•ENDEF

et n'est pas reconnue comme une fin de macro-définition.

12.2.4 DIRECTIVES DE MACRO-COMMANDE

Elles sont identifiées par le caractère "•". Elles peuvent se classer en 3 types :

- Affectation de variables :

•V

- Rupture de séquence :

•SKIP
•IF SKIP
•KILL

- Dialogue :

•PRINT
•LPRINT
•REPLY

Elles sont décrites au cours des paragraphes suivants.

12.2.5 UTILISATION DES PARAMETRES

A l'intérieur du corps de la macro, une ligne quelconque peut faire référence aux paramètres. Le nombre maximum de paramètres par macro est de 9. Ils sont numérotés de 1 à 9 dans leur ordre d'apparition.

Les trois opérations permises sur les paramètres sont les suivantes :

- ?Pn est le remplacement du paramètre formel par le paramètre effectif de rang n

- ?Ln est le remplacement par la longueur du paramètre exprimée en nombre de caractères

- ?Tn est le remplacement par le type du paramètre effectif de rang n.

Les différents types considérés sont :

type = 1 :	nombre hexadécimal (chiffres de 0 à F précédés de "'")
type = 2 :	nombre décimal précédé ou non d'un signe + ou -
type = 3 :	chaîne de caractère (caractères encadrés par des doubles apostrophes)
type = 4 :	symbole (chaîne de caractères alphabétiques et numériques, le premier étant alphabétique)
type = 5 :	autre.

Remarque : Le paramètre vide a comme longueur 0 et comme type 5.

12.2.6 DEFINITION DE VARIABLES : DIRECTIVE •V

Une fois définies dans le corps d'une macro-commande, les variables sont utilisables dans toute autre macro.

Les variables sont identifiées par un nom de la forme Vn1n2 dans la plage de V01 a V99.

La directive de définition •V permet d'attribuer à une variable un type et une valeur :

- type ENTIER si définie par :

```
•Vnln2  expression arithmétique
```

La valeur attribuée à la variable est le résultat du calcul de l'expression arithmétique entière.

Exemple :

```
•V01  = 45     donne 45
•V02  = -33/4          -8
```

- type CHAINE si définie par :

```
•Vnln2  EQ  expression  chaîne
```

Le nombre maximum de caractères est 78.

Exemple :

```
•VO3 EQ CHAINE
•VO4 EQ '124
```

Une même variable peut être définie plusieurs fois. Chaque redéfinition annule la définition précédente, en valeur et en type.

•V05 = 15 (entière)
•V05 EQ DOUZE (chaîne)

32.2.7 UTILISATION DES VARIABLES

La référence à une variable se fait par la syntaxe :

eVn1n2

qui représente la valeur actuelle de la variable.

Exemple :

•V06 = 20
PeV06

génère P20

12.2.8 EXPRESSIONS ARITHMETIQUES

Une expression arithmétique est un assemblage de variables entières et de nombre décimaux signes, relies entre eux par des opérateurs arithmétiques.

Les quatre opérateurs arithmétiques :

+ - • /

ont même priorité et l'ordre d'exécution des opérations est de la gauche vers la droite.

Le caractère "*" apparaissant dans une expression indique la multiplication des éléments situés à sa gauche (multiplicande) par l'élément situé à sa droite (multiplicateur).

Exemple :

15+2*-3+47

Multiplicande = 15 + 2

Multiplicateur = - 3

Résultat = - 4

Le caractère "/" apparaissant dans une expression indique la division entière des éléments situés à sa gauche (dividende) par l'élément situé à sa droite (diviseur).

Exemple :

16378/-2 -4

Dividende = 16378

Diviseur = - 2

Résultat = - 8193

La valeur d'une expression arithmétique est un nombre décimal signé, calculé au moyen de l'expression elle-même.

Exemple :

*V06 = eV05+12

Si V05 a la valeur 15, la valeur de VO6 est 27.

12.2.9 EXPRESSIONS CHAINES

Toute chaîne de caractères, assemblage de variables chaînes et caractères, est une expression chaîne.

Une expression arithmétique peut être considérée comme une expression chaîne. la réciproque n'étant évidemment pas vraie.

La valeur d'une expression chaîne est la chaîne elle-même.

Exemple :

```
•V06 EQ eV03 eV05
```

Si les valeurs respectives de V03 et V05 sont CHAINE et DOUZE, V06 a pour valeur CHAINE DOUZE.

12.2.10 RUPTURE DE SEQUENCE

12.2.10.1 Directive •SKIP

Syntaxe :

```
•SKIP expression arithmétique
```

Directive de rupture inconditionnelle, elle interrompt la séquence du traitement.

Le résultat du calcul de l'expression ou la simple valeur n fournie est un déplacement relatif a la ligne traitée.

Exempie :

```
•SKIP 3
....    Ligne ignorée
....    Ligne ignorée
....    Reprise du traitement
```

Cette séquence provoque un saut à la 3e ligne suivant la directive •SKIP.

Une valeur négative provoque un saut en arrière.

Une valeur aboutissant hors du corps d'une macro provoque une erreur (cf. par. 11.3.4).

La valeur 0 provoque la sortie de la macro ; le traitement reprend au point d'où a été appelée la macro.

12.2.10.2 Directive •IF SKIP

Syntaxe :

```
•IF relation SKIP expression arithmétique
```

Directive de rupture conditionnelle, elle poursuit le traitement, en séquence ou à une ligne indiquée, suivant qu'une relation est vraie ou fausse.

Lorsque la relation est vérifiée, •IF se résume à un •SKIP; dans le cas contraire, le traitement continue à la ligne suivante.

Deux types de relations peuvent intervenir. Elles conduisent à deux formes.

12.2.10.3 Relations arithmétiques

Trois opérateurs de relation sont utilisés :

<	inférieur
=	égal
>	supérieur

Une relation est alors formée de l'assemblage :

"expression arithmétique" {<I=I>} "expression arithmétique"

Exemples :

•IF ?P1 = 4 SKIP 3
•IF ?L2 = 0 SKIP -12

12.2.10.4 Relations logiques

Deux opérateurs sont utilisés :

EQ	équivalent
NE	non équivalent

Une relation est alors formée de l'assemblage :

"expression chaîne" {EQINE} "expression chaîne"

Les 2 expressions sont comparées entre elles, caractère par caractère. Deux chaînes sont équivalentes si elles ont la même longueur et sont composées de la même séquence de caractères.

Exemple :

•IF ?P2 EQ A SKIP 0
•IF eV10 EQ eV09 SKIP 5

Note :

La relation suivante est fausse : 3+4 EQ 7

12.2.10.5 Directive •KILL

Syntaxe :

*KILL expression chaîne

But :

- Abandon de la génération de la macro-commande,
- Impression de l'expression chaîne, qui peut représenter un message d'erreur.

Note :

La différence avec un •SKIP 0 réside dans le fait que ce dernier ne fait pas abandonner la génération, mais revient au point d'où a été appelée la macro.

12.2.11 DIRECTIVES DE DIALOGUE

12.2.11.1 Directive •PRINT

Syntaxe :

•PRINT texte

But :

Impression sur l'unité affectée à l'unité symbolique EL du texte précise.

La chaîne de caractères qui constitue le texte ne doit pas dépasser 71 caractères.

Le texte est complété par les caractères "retour-chariot" et "line-feed", ce qui provoque un passage à la ligne suivante.

12.2.11.2 Directive •LPRINT

Syntaxe :

•LPRINT texte

But :

De même que •PRINT, impression du texte sur l'unité symbolique EL mais sans que soient ajoutés les caractères de positionnement. On reste ainsi positionné en fin du texte imprimé.

12.2.11.3 Directive •REPLY

Syntaxe :

•REPLY [n]

But :

Lecture de caractères sur l'unité associée à l'unité symbolique EC. Ces caractères sont disponibles dans ?PO.

Comme pour les paramètres d'appel, le contenu de la chaîne peut être utilise (?PO), aussi bien que sa longueur (?LO) et son type (?TO).

En l'absence de paramètre, une ligne entière est lue, jusqu'à un retour chariot.

Si le nombre de caractères "n" est précisé, la lecture s'arrête quand ce nombre est atteint, quels que soient les caractères entres.

12.2.12 UTILISATION DES CARACTERES SPECIAUX

Les caractères "?", "e" et "%" jouent le rôle d'indicateurs. Ils repèrent en effet respectivement les paramètres, les variables et les macro-commandes. Il est toutefois possible de les utiliser dans les expressions chaînes et les textes imprimés par les directives •PRINT et •LPRINT. Il suffit alors de les doubler :

Ainsi :

•PRINT NON DU FICHIER??

imprime le texte :

NOM DU FICHIER?

12.2.13 CLASSEMENT DES MACRO-DEFINITIONS

Afin de définir de manière unique l'en-tête correspondant à une ligne donnée, lorsque plusieurs en-têtes peuvent être choisis, un certain nombre de règles sont utilisées :

Règle 1 : Le processus de recherche s'effectue de la gauche vers la droite.

Règle 2 : Au moins un séparateur doit s'intercaler entre deux indicateurs de paramètres (?).

Règle 3 : Entre un indicateur de paramètre et un séparateur le second est toujours choisi.

Règle 4 : Lorsqu'il y a ambiguïté dans le choix des paramètres, un groupement des caractères a droite est effectué.

Règle 5 : Dans un en-tête, le nombre d'espaces d'une suite d'espaces n'est pas significatif.

Il en résulte que :

- avec les deux en-têtes ? = ? + ? Et A = ? + ? la ligne source A = B + C correspond au second en-tête (règle 3)
- avec l'en-tête A = ? + ? la ligne source A = X + Y + Z donne comme paramètres X et Y + Z (règle 4).

12.3 UTILISATION DES MACRO-COMMANDES

12.3.1 GESTION DES MACRO-COMMANDES

Sous BOS16, la gestion des macro-commandes est supportée par l'option MACRO dont l'intégration au système est réalisée par la commande :

OPTION MACRO. taille "cr"

Le paramètre "taille" indique la dimension exprimée en mots de la zone de travail qui pourra être utilisée pour la gestion des macro-définitions.

12.3.2 DEFINITION DES MACRO-COMMANDES

L'éditeur de texte EDIT16 permet d'enregistrer les macro-définitions de l'utilisateur dans un fichier séquentiel ou dans un article d'un fichier indexé.

Ces définitions se terminent par la directive •ENDEF relative à la dernière macro.

La communication à BOS16 de ces macro-définitions est réalisée par la commande :

MAC [art.]nomfic[-catg][,[FU][,C]] "cr"

Le paramètre C permet de demander au système de ne pas réinitialiser ses tables. Les nouvelles définitions viennent alors s'ajouter à celles déjà gérées.

FU désigne l'unité fonctionnelle sur laquelle réside le fichier (par défaut FU spécifiée par la commande JOB).

12.3.3 EXECUTION DES MACRO-COMMANDES

L'appel d'une macro-commande est réalisée par une commande :

%macro-commande "cr"

La génération du contenu de la macro-définition correspondant est réalisée dans un fichier temporaire auquel est associé automatiquement l'unité symbolique CC.

Toute macro doit donc se terminer par une commande provoquant le retour au dialogue initial (RETURN ou EOJ).

12.3.4 MESSAGES D'ERREUR

Des messages ERMnn peuvent être émis en cas d'erreur. Leur signification est donnée ci-dessous :

ERM 00	Erreur de parité
ERM 01	Erreur d'écriture - Erreur dans l'écriture d'une instruction de macro - Numéro de variable supérieur à 99
ERM 02	Expression arithmétique incorrecte - Opérande non décimal - Enchaînement d'opérateurs interdit - Division par zéro - Valeur d'un opérande non représentable sur 16 bits - Valeur d'une expression non représentable sur 16 bits - etc...
ERM 03	Contexte incorrect ENDEF, KILL, SKIP, IF... hors d'une macro-définition
ERM 04	Elément non défini : variable ou paramètre
ERM 05	Appel de macro incorrect ou définition non intégrée
ERM 06	Définitions imbriquées
ERM 09	Saut à l'extérieur d'une macro
ERM 11	Niveau d'imbrication d'une macro supérieur à 10
ERM 12	Saturation des tables. Espace alloué dans la commande OPTION MACRO insuffisant
ERM 14	Ligne générée trop longue
ERM 15	Saturation de l'espace alloué aux variables

Lorsque le fichier (ou l'article) de macro-définitions est incomplet, la commande MAC est sanctionnée par le message : ERB94.

12.4 EXEMPLES

12.4.1 COMPILATION ET GENERATION D'IMAGE MEMOIRE

12.4.1.1 Macro-définition

```
•DEFPL
•LPRINT NOM DU FICHIER SOURCE ??
•REPLY
•V01 EQ ?PO
•LPRINT FIN DE TRAVAIL ??
•REPLY
•V02 EQ ?PO
•V03 EQ ZE
•LPRINT VOULEZ-VOUS UN LISTING (O/N) ??
•REPLY
•IF ?PO EQ N SKIP 2
•VO3 EQ LP
•LPRINT PROGRAMME MAITRE(M) OU ESCLAVE(S) ??
•REPLY
•VO4 EQ ?PO
•IF eV04 EQ S SKIP 4
•LPRINT ADRESSE D'IMPLANTATION ??
• REPLY
•VO5 EQ ?PO
•LPRINT NOM DU FICHER IMAGE MEMOIRE ??
• REPLY
•VO6 EQ ?PO
/JOB COMPIL,,eV02
/LO eV03
/CALL PL
/S1 eV01
/IPLC
/CALL LKLOAD
•IF eV04 EQ S SKIP 3
/MODE M,eV05
•SKIP 2
/MODE S
/LINK • ;
/CATA IM,eVO6
/EOJ
•ENDEF
```

12.4.1.2 Utilisation

La macro-commande :

```
%PL
```

initialise un dialogue qui a pour but de définir :

- le nom au fichier symbolique source à compiler,
- la FU disque de travail,
- la génération ou non d'une liste.
- le mode d'exécution du programme,
- en mode maître, l'adresse d'implantation,
- le nom du fichier image mémoire.

12.4.2 ARCHIVAGE D'UN FICHIER SUR BANDE MAGNETIQUE

12.4.2.1 Macro-définition

```
•DEFARCH ?,?
CALL FUP3
PAUSE POSITIONNER LA BANDE ET FRAPPER RETURN
FDUM,?P1,?P2,T1,0
TPIO REWI
SEAR,?P1,T1
FVER,T1,?P2
RETURN
•ENDEF
```

12.4.2.2 Utilisation

La macro-commande ARCH comporte deux paramètres qui permettent de définir le nom du fichier à archiver sur bande magnétique, ainsi que la FU qui le supporte.

Exemple :

```
%ARCH SOURCE-SY,D3
```

Réseaux
et systèmes
d'information

Bull

VOS REMARQUES SUR CE DOCUMENT

Bull Sems

TITRE

BOS16 ADDENDUM A

N° DE REFERENCE

Bull S.A.: 1 164 325 00 036 05/FR

DATE

DECEMBRE 1986

ERREURS DETECTEES

AMELIORATIONS SUGGEREES

*→ Vos remarques et suggestions seront attentivement examinées.
si vous désirez une réponse écrite, veuillez indiquer ci-après
votre adresse postale complète.

NOM : ..... DATE .....

SOCIETE : .....

ADRESSE : .....

*→ Remettez cet imprimé à un responsable Bull-S.A. ou envoyez le
directement à

Bull CEDOC-CELOG
Boite Postale 110
Parc Industriel d'INCARVILLE
27100 - VAL DE REUIL - FRANCE

TYPE DE DOCUMENT : Manuel de référence

Bull Sems

NOM : BOS16 ADDENDUM A CLASSIFICATION : SSL SYST SUPV

REFERENCE : 1 164 325 00 036 05/FR VERSION : 10

REVISIONS SUCCESSIVES

INDICE	DATE	AUTEUR	OBJET
00	23/05/84	HISLEUR	Prise en compte des nouvelles UC SOLAR
01	09/08/84	HISLEUR	Prise en compte des disques Winchester
02	19/04/85	HISLEUR	Amélioration Winchester
03	08/01/86	COTTE-BARROT	Addendum A - Amélioration des performances
04	17/02/86	COTTE-BARROT	Intégration de l'addendum A, Ajout exploitation des disques VERTEX (DWB50)
05	05/12/86	COTTE-BARROT	Addendum A - Exploitation des disques Winchester DWF20, DWB50-1

VISAS		APPROBATION
AUTEUR	HIERARCHIQUE	RESPONSABLE DE PROJET
COTTE-BARROT		HISLEUR