

CHAINE DE PRODUCTION DE PROGRAMMES

MANUEL DE REFERENCE

SOMMAIRE

1	—	INTRODUCTION	1-1
2	—	PRESENTATION	2-1
2.1	-	IMAGE MEMOIRE -GENERALITES	2-1
2.2	-	IMAGE MEMOIRE 32 K	2-2
2.3	-	IMAGE MEMOIRE 64 K	2-3
2.4	-	CHOIX DE LA GEOGRAPHIE	2-5
2.5	-	CAS PARTICULIER DES DONNEES AU DELA DE 32 K	2-5
3	—	EDITEUR DE LIENS ET CHARGEUR	3-1
3.1	-	EDITEUR DE LIENS	3-2
3.1.1	-	Présentation	3-2
3.1.2	-	Activation de l'Editeur de liens	3-2
3.1.3	-	Bibliothèque utilisateur	3-4
3.1.4	-	Remarque importante	3-5
3.2	-	CHARGEUR DISQUE	3-6
3.2.1	-	Description	3-6
3.2.2	-	Commandes d'activation	3-7
3.2.3	-	Place disque nécessaire	3-9
4	—	EDITEUR • CHARGEUR • LKLOAD	4-1
4.1	-	DESCRIPTION GENERALE	4-1
4.2	-	UTILISATION DE LKLOAD POUR LA PRODUCTION DE PROGRAMMES	4-2
4.2.1	-	Introduction des modules binaires	4-2
4.2.2	-	Utilisation de bibliothèques	4-2
4.2.3	-	Fichier image mémoire résultat	4-2
4.2.4	-	Etat d'avancement de la génération	4-3
4.2.5	-	Définition du contexte de génération	4-3
4.2.6	-	Scrutation automatique de bibliothèque	4-4
4.2.7	-	Utilisation de la CDA	4-5
4.3	-	DESCRIPTION DETAILLEE DES COMMANDES	4-6
4.3.1	-	MODE: définition des paramètres de la génération	4-6
4.3.2	-	PERFORM Option de performance	4-9
4.3.3	-	LINK : édition de liens	4-10
4.3.4	-	BRAN : définition de branche	4-11
4.3.5	-	LOOK : scrutation de bibliothèque	4-12
4.3.6	-	STAT : impression des références non résolues	4-13
4.3.7	-	DEFINE : impression des références résolues	4-14
4.3.8	-	ZDEF : remise à zéro de symboles externes définis	4-14
4.3.9	-	ENDL : fin de traitement	4-14
4.3.10	-	OPTION : spécification d'options	4-15

4.3.11	▪ LIBRARY : scrutation automatique de bibliothèque	4-15
4.3.12	▪ CDA : désignation du fichier de cohérence CDA	4-17
4.3.13	▪ ICDA : initialisation de la géographie CDA	4-17
4.4	- MESSAGES D'ERREUR -CODE DE RETOUR	4-19
4.5	- CONSEILS D'UTILISATION	4-20
4.5.1	▪ Encombrement et performances	4-20
4.5.2	▪ Place disque nécessaire	4-20
4.5.3	▪ Conseils pratiques	4-20

ANNEXES

1.	TABLEAU RECAPITULATIF DES POSSIBILITES ENTRE LES DIFFERENTES COMMANDES ET LES PRINCIPAUX PARAMETRES	A.1
2.	MESSAGES D'ERREUR EMIS PAR LE CHARGEUR DISQUE	A.2
3.	NUMEROS DES ERREURS EMIS PAR LKLOAD ET L'EDITEUR DE LIENS	A.3
4.	NUMEROS DES AVERTISSEMENTS EMIS PAR LKLOAD ET L'EDITEUR DE LIENS	A.4
5.	PROGRAMMATION AVEC DES DONNEES AU DELA DE 32 K	A.5

AVERTISSEMENT

La production de programmes sur matériel SOLAR permet à l'utilisateur d'obtenir des image mémoire exécutables grâce à des outils assurant simplicité et confort.

Le processeur de base de cette chaîne de production de programmes est le processeur LKLOAD utilisé en "MODE 64K".

C'est ce contexte de production de programme qui a été optimisé.

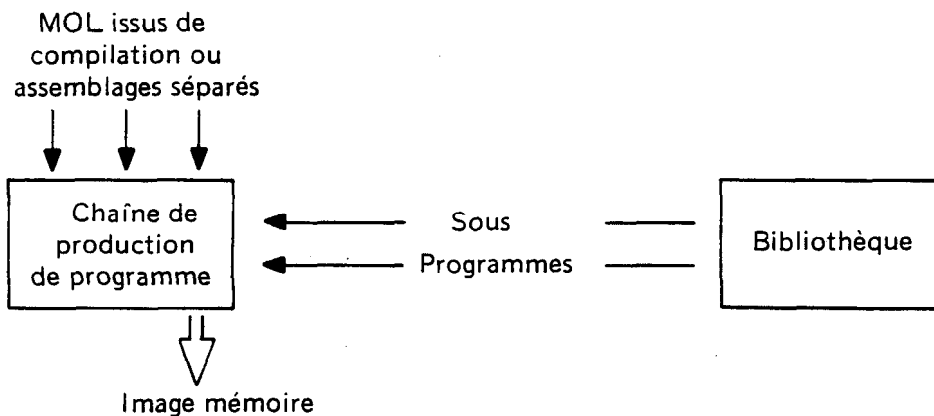
D'autre part, c'est le seul prenant en compte toutes les possibilités offertes par les langages de programmation de haut niveau (FORTRAN77, PASCAL, etc...) dans le domaine du temps réel.

L'utilisateur est donc fortement convié à utiliser le processeur LKLOAD dans le cadre de la géographie 64 K en réservant les processeurs EDILE et BUILD décrits également dans ce document aux applications spécifiques.

1 - INTRODUCTION

Les compilateurs SOLAR (PL16, FORTRAN, PASCAL, etc...) produisent des MOL (Modules Objets "link-éditables"). L'assembleur produit soit des MOL, lorsque le texte source comporte des directives d'assemblage EXT, soit des programmes BT (Binaires Translatables).

La fonction principale de la chaîne de production de programmes est la génération à partir de MOL ou BT, issus de compilations ou assemblages séparés, d'une image mémoire exécutable. Elle permet aussi la scrutation de bibliothèque et l'insertion dans l'image mémoire des sous-programmes référencés par le programme principal.



Une image mémoire peut être produite de deux façons :

- soit par l'éditeur de liens EDILE suivi du générateur d'image mémoire BUILD
 - . EDILE produit un BT à partir de plusieurs MOL ou BT
 - . BUILD génère une image mémoire exécutable à partir d'un BT fourni par EDILE,
- soit par l'éditeur-chargeur LKLOAD qui effectue en un seul passage les deux étapes précédentes.

L'utilisation du processeur LKLOAD est fortement recommandée dans tous les cas :

- soit pour obtenir directement une image mémoire exécutable
- soit pour obtenir un binaire translatable (option "LINK").

La documentation concernant le processeur EDILE ne doit être considérée que pour mémoire (anciennes procédures de génération).





2 - PRESENTATION

2.1 - IMAGE MEMOIRE - GENERALITES

Une image mémoire exécutable est supportée par un fichier FMS de type indexé comportant au minimum trois articles :

- . un article ROOT contenant la racine du programme,
- un article DESC (ou DESC64) contenant les informations nécessaires au chargement et à l'exécution du programme,
- un article LABEL contenant :
 - . la date et l'heure de création de l'image mémoire,
 - . les commentaires d'identification,
 - . éventuellement la liste d'implantation des sections générée par BUILD ou LKLOAD.

Si l'image mémoire a une structure en OVERLAY, le fichier image mémoire comporte en plus un article par branche. Dans ce cas, l'article ROOT comporte également la dernière des branches présentée lors de l'édition de liens.

Il existe deux types d'image mémoire donnant lieu à deux types de géographie une fois l'image chargée en mémoire :

- une image mémoire "32 K" dans laquelle données et instructions sont mélangées et implantées dans l'ordre de présentation des modules à la phase d'édition de liens,
- une image mémoire "64 K" dans laquelle les données sont implantées dans les premiers 32 K suivies par les instructions.

Une image mémoire de type "64 K" peut être de taille quelconque. Par contre, toute image-mémoire de taille supérieure à 32 K doit être de type "64 K".

De toutes façons, ceci est transparent pour l'utilisation final auquel la chaîne de préparation de programme garantit une implantation des données permettant l'exécution correcte de son programme.

Une image mémoire peut comporter 16 tâches au maximum, à raison d'une PST par tâche. On peut donc avoir 16 PST par image mémoire.



2.2 - IMAGE MEMOIRE "32 K"

Comme dit précédemment, ce type d'image mémoire est caractérisé par le fait que données et instructions sont mélangées et destinées à être implantées dans les premiers 32 K de la partition utilisateur.

Une image mémoire "32 K" possède un descripteur de nom "DESC" dont la structure est donnée ci-dessous :

- longueur : 23 mots (1 PST)
- 53 mots (plus d'une PST).

Article DESC

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15																																	
0	Adresse d'implantation																																																
1	Adresse de lancement																																																
2	Adresse des branches																																																
3	Mode								Taille																																								
4	Nombre de PST								Nombre de branches																																								
5	Adresse de la PST																																																
	<div style="text-align: right;">} 18 mots relatifs à la PST 1</div>																																																
22																									Type	Niveau de priorité																							
23																	Adresse de la PST																																
24																									Type	Niveau de priorité																							
																	<div style="text-align: right;">} PST 2</div>																																
51																																	Adresse de la PST																
52																																									Type	Niveau de priorité							
																																	<div style="text-align: right;">} PST 16</div>																

Mode = 1 (esclave) ou 0 (maître)
Type = 0 (logiciel) ou 1 (matériel)

2.3 - IMAGE MEMOIRE "64 K"

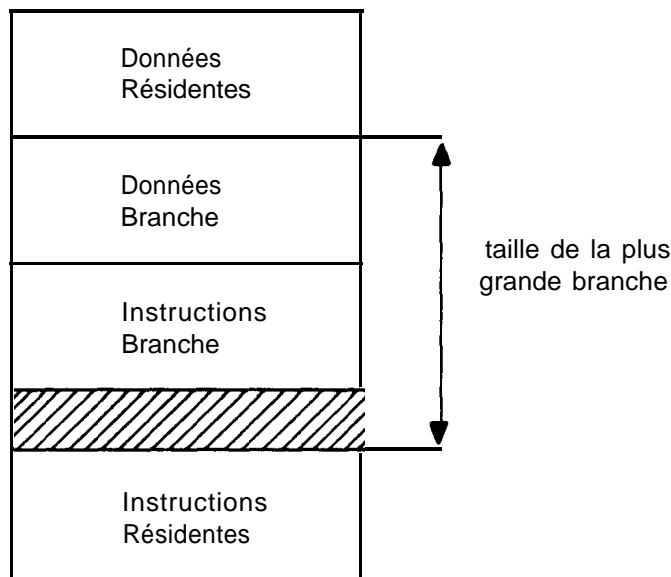
Pour les images mémoires de ce type la chaîne de production réorganise les données et les instructions de manière à implanter les données en tête, dans les 32 premiers K, puis les instructions.

Pour le code généré par l'assembleur ou le compilateur RPGII, la distinction entre données et instructions n'est pas possible, aussi tout le MOL en provenance de l'assembleur et du compilateur RPG II est considéré comme des données.

Dans le cas d'une image mémoire en "overlay", la réorganisation des données et des instructions est également valable au niveau des branches.

Pour ne pas modifier la gestion de l' "overlay" et l'alimentation d'une branche en un seul accès au système de fichiers, l'ensemble données "overlay" - instructions "overlay" forme un seul bloc qui est implanté après les données de la racine.

Une image mémoire "64 K", une fois chargée en mémoire, présente la géographie suivante :



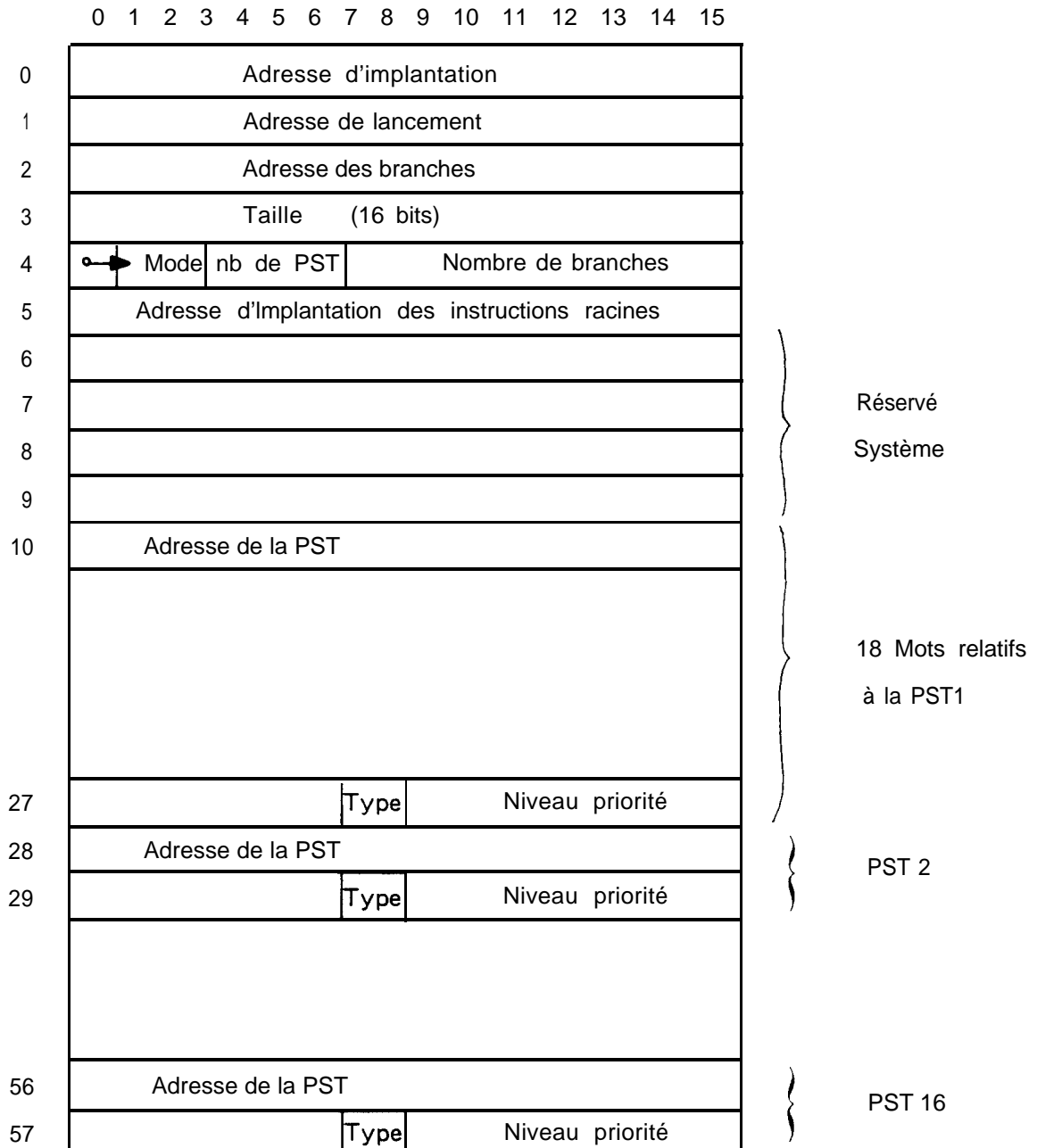
- la taille des données de la racine et de la plus grande branche, ne doit pas dépasser 32 K.
- la taille des instructions peut par contre dépasser 32 K.

Une image mémoire de ce type est caractérisée par un descripteur de nom "DESC64" dont la structure est donnée ci-après :

- longueur : . 28 mots pour 1PST
- . 58 mots pour plus d'une PST

Ce descripteur ne sera reconnu que par un système capable d'exécuter des images mémoires 64 K.

ARTICLE DESC 64





2.4 - CHOIX DE LA GEOGRAPHIE

La géographie "64 K" est reconnue par tous les systèmes disques standards du SOLAR. D'autre part, les langages de haut niveau (COBOL 74, PASCAL, FORTRAN 77) nécessitent ce type de géographie. Donc, du point de vue de l'utilisateur, la géographie "64 K" est la géographie à utiliser en standard, quelle que soit la taille de son programme.

L'utilisation de la géographie "32 K" doit être réservée à la génération d'image-mémoire dans les cas particuliers qui la nécessitent :

- génération de certains processeurs du catalogue SOLAR,
- applications spécifiques (avec des données implantées au delà de 32 K par exemple).

2.5 - CAS PARTICULIER DES DONNEES AU DELA DE 32 K

Il peut arriver que, pour certaines applications, on soit amené à devoir implanter des données au delà de la zone 0-32 K d'une partition. La chaîne de préparation de programme possède des commandes spécialisées pour prendre en compte ce type d'implantation. Ces commandes sont décrites au niveau de chaque processeur dans les pages qui suivent.

Mais il faut savoir que l'écriture de ce genre d'applications nécessite une bonne maîtrise de la part de l'utilisateur de la technique d'adressage SOLAR, ainsi qu'une programmation particulière avec un certain nombre de règles à respecter. Ce style de programmation est décrit dans l'annexe technique n° 5 du présent manuel.

Bien entendu, la possibilité d'utiliser cette programmation particulière n'est effective qu'avec des langages supports proches de la machine, c'est-à-dire essentiellement l'assembleur et le PL16 (pour ce dernier langage, il faut se souvenir que l'on peut l'utiliser de façon tout à fait standard avec des images mémoire "64 K").

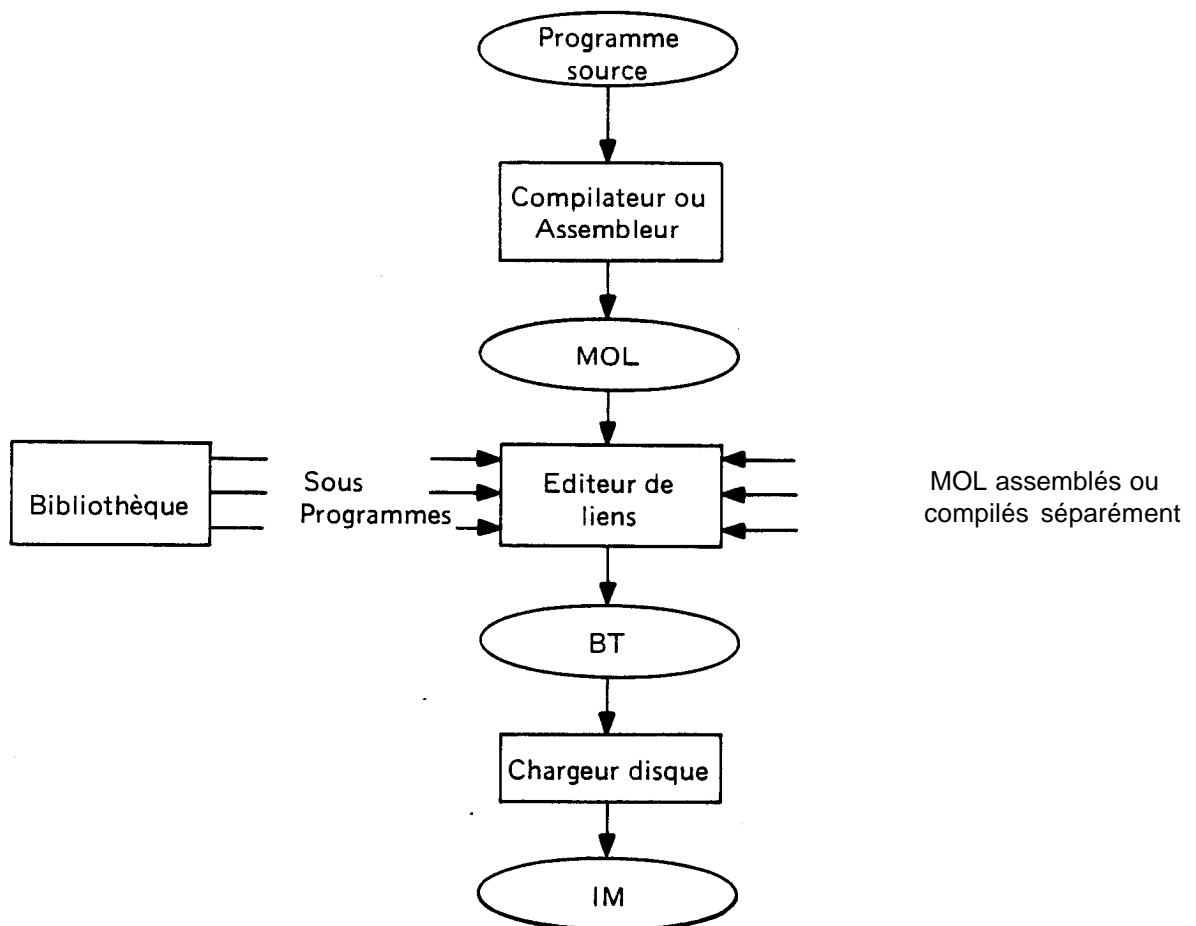


3 - EDITEUR DE LIENS ET CHARGEUR

DESCRIPTION GENERALE

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

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



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



3.1 - EDITEUR DE LIENS

3.1.1 - Présentation

L'éditeur de liens EDILE est appelé en mémoire par la commande :

CALL EDILE (C)

Un certain nombre de commandes vont alors permettre la production d'un programme binaire translatable sur l'unité symbolique BO.

Les modules objets "link-éditables" sont entrés à partir de l'unité symbolique BI.

EDILE permet la scrutation d'une bibliothèque utilisateur et l'insertion dans le programme binaire translatable des programmes référencés.

Tout programme symbolique peut, en effet, préciser les programmes qui lui sont nécessaires et qui seront à rechercher dans une bibliothèque. Il suffit de définir leurs noms en tant que symboles externes (au moyen, par exemple, de la directive assembleur EXT).

3.1.2 - Activation de l'éditeur de liens

L'éditeur de liens EDILE possède 8 points d'entrée. A chacun d'eux correspond une commande qui est reconnue par le superviseur.

A partir d'un certain nombre de MOL l'éditeur de liens génère un programme binaire translatable dont la structure est spécifiée par les commandes d'activation.

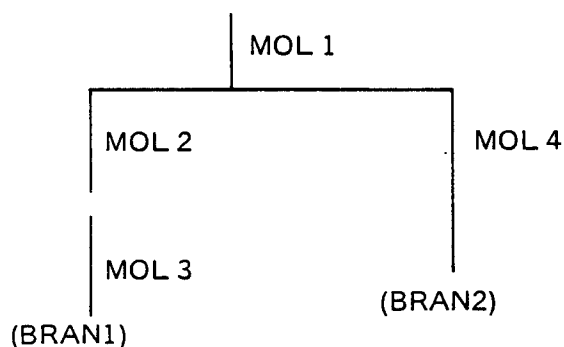
a) Commandes de base : ILNK, CLNK, BLNK

ILNK [,64]	initialisation d'une édition de liens. Origine de la racine. Le paramètre 64 permet la génération d'une IM avec la géographie 64 K.
BLNK	Origine d'une branche (avec définition du nom de la branche)
CLNK	Continuation.

Elles permettent à partir de MOL entrés sur l'unité symbolique BI de définir la structure d'un programme. La commande d'activation de l'éditeur de liens BLNK n'est à utiliser que dans le cadre d'une structure d' "overlay" à chaque origine de branche. Elle comporte le nom de la branche.

Exemple :

EDILE effectue la liaison entre quatre MOL lus sur le lecteur rapide. La structure de l'arbre d' "overlay" à réaliser est la suivante :





BI	HR	Affectation à B1 du lecteur rapide
ILNK		Origine de la racine (MOL 1)
BLNK	BRAN1	Origine de la branche BRAN1 (MOL 2)
CLNK		Continuation de BRAN1 (MOL 3)
BLNK	BRAN2	Origine de la branche BRAN2 (MOL 4)

Dans l'exemple ci-dessus l'éditeur de liens redonne le contrôle au superviseur après chaque traitement élémentaire. Cela permet à l'utilisateur de monter le module suivant sur le périphérique associé à BI avant de réactiver EDILE.

b) Commande RLNK

Après chaque traitement l'éditeur de liens signale s'il existe des références non résolues avant de rendre le contrôle au superviseur.

Lorsque c'est le cas, la commande RLNK permet de demander l'édition de leur liste.

c) Commande ELNK

La commande ELNK est utilisée pour indiquer la fin de l'édition de liens.

Elle donne lieu à la génération dans le programme objet d'un certain nombre d'informations utiles au chargeur ainsi que du "checksum" du programme qui sera recalculé et vérifié au moment d'une exploitation ultérieure.

La commande ELNK est donc obligatoire et doit terminer toute édition de liens.

Exemple :

```
ILNK }   Traitement du programme 1
----- }
CLNK }   Traitement du programme 2
----- }
ELNK    Fin de l'édition de liens
```

Remarques :

- Lorsqu'il subsiste encore des références qui n'ont pas été résolues, l'éditeur de liens force à la valeur zéro les adresses qui correspondent aux symboles externes non définis.
- La commande CLNK est refusée par l'éditeur de liens lorsqu'elle intervient après une commande ELNK. Il convient en effet de réinitialiser EDILE (commande ILNK) avant d'effectuer un nouveau traitement.

d) Commande TLNK

Lorsque le support utilisé pour le programme objet est le ruban papier, il peut être intéressant d'avoir le programme en plusieurs bandes.

La commande TLNK est alors utilisée pour indiquer la fin de chaque bande intermédiaire, la commande ELNK devant indiquer la fin de la dernière bande.

Exemple :

```
ILNK }   Traitement du programme 1
..... }
CLNK }   Traitement du programme 2
..... }
TLNK    Fin de la bande objet 1
..... }
CLNK }   Traitement du programme 3
..... }
ELNK    Fin de la bande objet 2 (fin de l'édition de liens).
```

e) Commande ZLNK

La commande ZLNK permet à l'utilisateur de rendre non définis les symboles externes déjà définis.

Cette commande donne ainsi la possibilité de "link-éditer" plusieurs fois la même portion de programme relative à un externe.

3.1.3 - Bibliothèque utilisateur

La commande LLNK permet de préciser le numéro d'utilisation d'une bibliothèque utilisateur. Sa forme est la suivante :

```
LLNK fnum (cr)
```

où fnum est le numéro d'utilisation de la bibliothèque.

Une bibliothèque est en fait un fichier indexé qui a été ouvert par une commande OPEN. Elle sera refermée par une commande CLOSE ou EOJ.

Exemple :

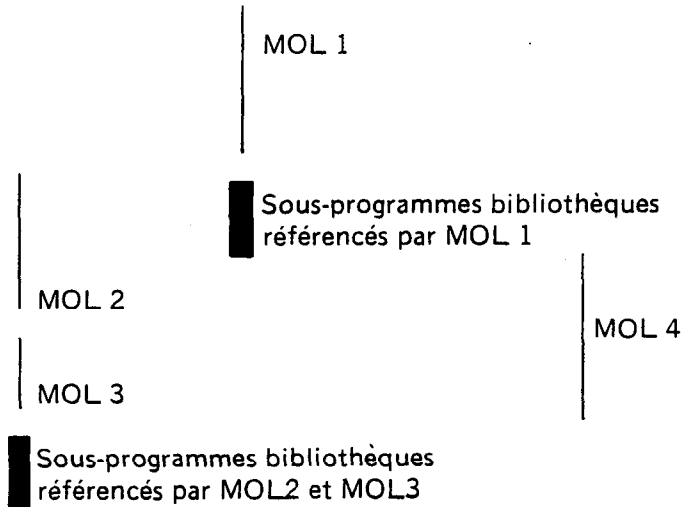
```
OPEN OLD 1, BIBLIO, D3
LLNK 1
CLOSE 1
```

LLNK commande à l'éditeur de liens la scrutation de la bibliothèque utilisateur spécifiée afin de satisfaire les références externes non résolues :

- tout symbole externe non défini est recherché en tant que nom d'article dans la bibliothèque
- si l'article existe, il est considéré comme un MOL qui est alors traité par l'éditeur de liens
- si le MOL entrant comporte des références externes non résolues la bibliothèque est scrutée de nouveau.

Tout article d'une bibliothèque doit comporter dans le MOL la définition du symbole externe de même nom que lui.

Exemple :



La segmentation interdit qu'une branche fasse référence à un symbole défini dans une autre branche. Par conséquent, lorsqu'en fin de branche ii existe des références apparues dans la branche et non résolues après scrutation des diverses bibliothèques, l'éditeur de liens force à la valeur zéro les adresses correspondantes.

3.1.4 - Remarque importante

Le processeur EDILE n'accepte pas en entrée les binaires linkétables issus des compilateurs FORTRAN 77 et PASCAL version Temps Réel.

Il est nécessaire d'utiliser alors le processeur LKLOAD avec l'option "LINK".

3.2 - CHARGEUR DISQUE

3.2.1 - Description

L'appel en mémoire du chargeur disque est réalisé au moyen de la commande :

CALL BUILD 

L'opération réalisée est la création sur disque, dans un fichier temporaire, d'une image mémoire, à partir d'un programme binaire translatable issu de l'assembleur ou de l'éditeur de liens et entré sur l'unité symbolique BI. Ce fichier temporaire est créé dans l'espace disque connu sous le nom de "FU du JOB en cours" (sous TIME-SHARING, FU de l'utilisateur). Une image mémoire est un fichier indexé qu'il est possible de cataloguer.

Exemple :

CALL BUILD	Activation du chargeur disque
BI HR	Association à BI du lecteur rapide
MLOD '80	Création de l'image mémoire
CATAL IM, PROG-PW	Catalogue de l'image mémoire (fichier permanent PROG du catalogue PW)

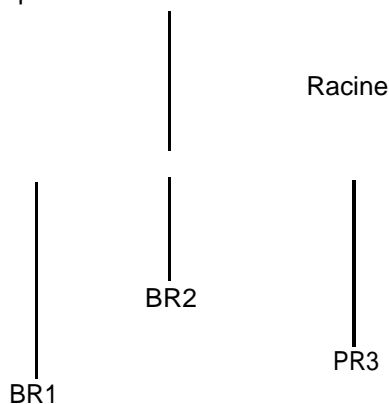
Le chargeur disque accepte qu'une image mémoire soit le support de plusieurs tâches (16 au maximum).

Cela permet à l'utilisateur d'effectuer l'édition de liens de ses tâches et de résoudre les problèmes de communication entre elles, à condition que l'image mémoire soit exploitée par BOS-D (pour RTES-D une seule tâche par IM).

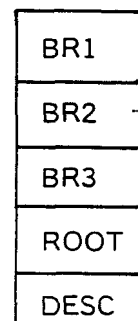
A raison d'une PST par tâche, une image mémoire peut donc contenir jusqu'à 16 PST.

Un fichier image mémoire comporte au moins trois articles, le descripteur DESC, le LABEL et la racine ROOT et, dans le cas d'un programme segmenté, un article par branche.

Exemple :



donne lieu au fichier indexé :



L'article LABEL regroupe :

- la date et l'heure de génération de l'image mémoire
- et, dans le cas où, dans sa forme symbolique, le programme comporte des directives assembleur IDP ; les diverses identifications du programme.

Cet article peut être listé par EDIT16 ou le processeur FTPD (clé LIDP).

Lors de la constitution d'une image mémoire un certain nombre d'informations sont reconnues par le chargeur disque et sont regroupés dans le descripteur DESC (ou DESC 64).

Ces informations seront exploitées ultérieurement lors du chargement de l'IM sous le système superviseur.


3.2.2 - Commandes d'activation

Les six commandes du chargeur disque sont les suivantes :

MLOD	Programme exécutable en mode maître
SLOD	Programme exécutable en mode esclave, implanté en 0 de la partition
ALOD	Programme absolu en mode maître
TLOD	Programme en mode esclave implanté à une adresse non nulle par rapport au début de la partition
CLOD	Continuation
FAST	Option de performance.

a) Commande MLOD

$$\text{MLOD } [[/] ' \text{adr1}] \left[, [\text{nvprio}] \left[, [\text{nbran}] \left[, [\text{taille}] [, [/] ' \text{adr2}] \right] \right] \right] \text{ (cr)}$$


adr1 : adresse d'implantation du programme (adresse amont) le caractère  permet de spécifier au chargeur que l'adresse qui suit est l'adresse de fin d'implantation du programme (adresse aval).

$0 \leq \text{adr1} = \text{nombre hexadécimal} \leq \text{'FFFE}$

nvprio : niveau de priorité associé à la première PST contenue dans le programme. Ce niveau remplace alors celui donné dans la forme symbolique du programme.

nbran : nombre de branches dans le cas d'un programme segmenté
 $2 \leq \text{nbran} = \text{nombre décimal} \leq 255$

taille : taille du programme exprimée en K mots
 $1 \leq \text{taille} = \text{nombre décimal} \leq 64$
 dans le cas où le paramètre taille est supérieur à 32, l'image mémoire générée a une géographie 64 K.

adr2 : adresse d'implantation des instructions de la racine (adresse amont)
 le caractère  permet de spécifier une adresse de fin d'implantation des instructions de la racine (adresse aval)

Ce paramètre n'est accepté que si le paramètre taille est supérieur à 32
 $0 \leq \text{adr2} = \text{nombre hexadécimal} \leq \text{'7FFE}$

L'usage de ce paramètre est réservé à la génération et à l'implantation des produits du catalogue SOLAR.

Par défaut la commande MLOD est :

MLOD '40, x, 28, 4

b) Commande SLOD

Forme :

SLOD *nvprio,nbran, taille* $\text{\textcircled{CR}}$

où les paramètres sont ceux de la commande MLOD.

Aucun des paramètres n'est obligatoire, toutefois le rôle de chacun d'eux est fonction du nombre de virgules qui le précèdent.

Remarque :

Dans le cas des commandes MLOD et SLOD. lorsque l'image mémoire a été sous-dimensionnée, il y a impression par le chargeur disque des messages suivants :

ERU 15	La taille effective est supérieure à celle qui a été spécifiée par la commande d'activation.
ERU 22	Le nombre de branches effectif est supérieur à celui qui a été spécifié par la commande d'activation.

c) Commande ALOD

ALOD [/] 'adr1 [,[*nvprio*][,[*nbran*] [,*taille*]]]

où les paramètres ont la même signification que pour la commande MLOD.

Le paramètre *taille* ne doit pas être supérieur à 32.

La création d'une image mémoire de type absolu en mode maître nécessite certaines contraintes au niveau de la programmation :

Exemple : ne pas avoir un adressage au-delà de 32 K avec des relais indexés (cf. ANNEXE n° 5).

d) Commande TLOD

TLOD [/] 'adr1 [,[*nvprio*][,[*nbran*] [, *taille*]]]

où les paramètres ont la même signification que pour la commande MLOD.

Le paramètre *taille* doit être supérieur à 32 K.

Remarques :

- La connaissance à priori des paramètres *taille* et nombre de branches permet au chargeur disque une gestion optimisée de l'occupation sur disque.

- Le paramètre *taille* doit être supérieur ou égal à la taille réelle du programme.

Dans le cas d'une structure d' "overlay" la taille réelle est obtenue par addition de la taille de la racine et de la taille de la plus longue des branches.

Le paramètre *taille* définit à la fois la taille de l'image mémoire et le type de géographie (*taille* > 32 K donne une géographie 64 K). Donc, même pour une image mémoire de faible taille, si l'on veut une géographie 64 K, on précisera un paramètre *taille* supérieur à 32 (par exemple : 33).

- Tout binaire translatable produit avec la commande ILNK,64 de l'éditeur de liens doit être "buildé" avec un paramètre *taille* supérieure à 32 K.

e) Commande CLOD

Elle permet le traitement de programmes en plusieurs bandes (cf commande TLNK de l'éditeur de liens).

Le traitement de la première bande s'effectue après une commande MLOD ou SLOD. Il y a ensuite retour sous le contrôle du superviseur ce qui permet à l'utilisateur de monter la bande suivante sur le périphérique associé à l'unité symbolique BI avant de relancer le traitement par une commande CLOD.

Le processus peut être répété autant de fois qu'il est nécessaire.

Exemple :

MLOD '80	Traitement de la bande 1
CLOD	Traitement de la bande 2
.....	

f) Commande FAST

La commande FAST a pour objectif de permettre, lors de l'exécution d'un programme segmenté, l'alimentation des branches en un minimum d'accès disque.

Dans ce but :

- . la taille des articles constituant les branches est arrondie à un multiple de secteurs
- . un ensemble d'informations système est créé sur disque définissant pour le fichier une organisation physique directe (traitement équivalent à celui réalisé par la commande FAST de l'utilitaire FUP 10).

Ces dernières informations seront exploitées par le système de fichiers FMS-E lors de l'ouverture du fichier avec demande de l'option ADR.

La forme de la commande est la suivante :

FAST nomfic [-catg] **CR**

nomfic : nom du fichier image mémoire.

catg : nom du catalogue du fichier (par défaut le catalogue est nul).

Remarque :

Le fichier image mémoire généré par le chargeur disque est temporaire. En fin de traitement, si celui-ci n'a donné lieu à aucune erreur fatale, le fichier est transformé en un permanent dont le nom et le catalogue sont ceux qui ont été précisés par la commande FAST. Il n'y a donc plus lieu de cataloguer le fichier.

3.2.3 - Place disque nécessaire à la construction d'une image mémoire

Lors de la génération d'une image mémoire, le chargeur disque nécessite sur disque une place suffisante pour :

- . un fichier temporaire de stockage des commentaires,
- . un fichier de pagination de la taille précisée dans le paramètre taille,
- . un fichier indexé support de l'image mémoire à générer.



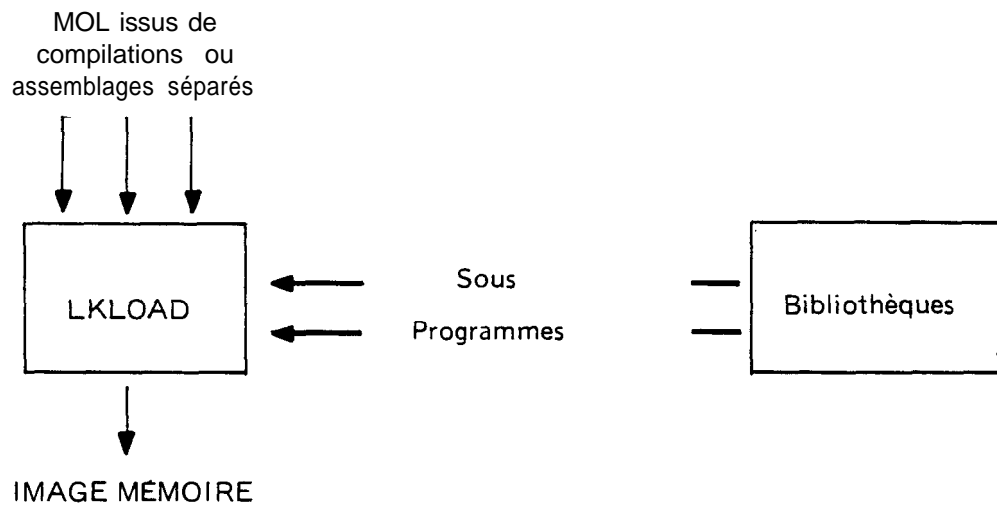
4 - EDITEUR - CHARGEUR - LKLOAD

4.1 - DESCRIPTION GENERALE

L'opération réalisée par l'éditeur-chargeur LKLOAD est la génération d'une image mémoire exécutable en une seule opération.

Cette image mémoire peut résulter de l'édition de liens d'un certain nombre de modules désignés par l'utilisateur ou recherchés en bibliothèque.

LKLOAD permet en effet la scrutation de bibliothèques et l'insertion dans l'image mémoire des sous-programmes référencés par le programme principal.



LKLOAD réalise donc en une seule fois les phases d'édition de liens (EDILE) et de génération d'une image mémoire.

Il permet également d'obtenir du binaire translatable assimilable par le processeur BUILD. Il doit donc être préféré dans ce contexte au processeur EDILE précédemment décrit.

Les fichiers utilisés en entrée ou sortie par LKLOAD sont supposés se trouver dans la FU "du job en cours" (FU banale de l'utilisateur sous les systèmes temps partagé) à moins qu'une autre FU (ou SU) ne soit explicitement spécifiée dans la commande.



4.2 - UTILISATION DE LKLOAD POUR LA PRODUCTION DE PROGRAMMES

Dans ce paragraphe sont décrites les grandes fonctions de LKLOAD avec l'utilisation des commandes disponibles.

La syntaxe détaillée de chaque commande est décrite au paragraphe suivant.

4.2.1 - Introduction des modules binaires

Le processeur LKLOAD reçoit en entrée des binaires compilés qu'il concatène en résolvant les références externes. Dans le mode "64 K", il implante en premier les données des modules puis les fait suivre par les instructions.

Un module est introduit dans le processeur LKLOAD par une commande LINK ou une commande BRAN.

Un module introduit par une commande LINK alors qu'aucune commande BRAN n'a été émise est implanté dans la RACINE au début ou à la suite selon qu'il s'agit du premier LINK ou d'une commande suivante.

Un module introduit par une commande BRAN définit une branche d'overlay dont il constitue le premier module.

Un module introduit par une commande LINK après qu'une commande BRAN ait été émise est implanté dans la branche en cours à la suite.

La fin de la génération d'une image mémoire est signifiée par une commande ENDL ou par la présence d'un ";" en fin d'une commande d'introduction de module.

4.2.2 - Utilisation de bibliothèques

Une bibliothèque est un ensemble de binaires compilés supportés par un fichier indexé au sens FMS à raison d'un module par article.

Deux utilisations de ces fichiers bibliothèques sont possibles :

- support plus compact de modules binaires : un module est introduit par une commande LINK ou BRAN portant sur le nom de l'article contenant le module.
- scrutation avec sélection de modules binaires :
les modules de la bibliothèque impliqués dans la génération en cours sont introduits par un ordre LOOK portant sur le nom du fichier bibliothèque.
Pour ce faire, il est nécessaire que les modules soient supportés par des articles dont le nom est celui sous lequel ils sont référencés en tant qu'externes dans le programme appelant. Un article peut contenir un module définissant plusieurs externes, dont un doit avoir le même nom que celui de l'article. Lorsqu'une commande LOOK est émise, le processeur LKLOAD extrait dans sa table d'externes ceux qui ne sont pas encore définis et cherche s'il n'y a pas d'articles de nom correspondant dans le fichier bibliothèque spécifié.
La commande LOOK est donc équivalente à une série de commande LINK portant sur des articles de bibliothèque.

4.2.3 - Fichier image mémoire résultat

Le résultat de la production de programme par LKLOAD est un fichier image mémoire dont la structure et les caractéristiques sont décrites au chapitre n° 2.

Ce fichier résultat peut être :

- soit un fichier temporaire créé à chaque activation de LKLOAD et connu par les systèmes sous le mnémonique IM. En fin de phase de génération, il est donc nécessaire de cataloguer ce fichier temporaire sous le nom désiré,

- soit un fichier permanent. Dans ce cas, il est nécessaire de spécifier les paramètres de ce fichier (nom, support) dans une commande PERFORM émise en phase d'initialisation de la génération. De plus, dans le cas d'image mémoire en overlay, cette commande réalise l'alignement des branches à frontière de secteur afin de minimiser le temps d'alimentation de ces branches lors de l'exécution du programme.

4.2.4 - Etat d'avancement de la génération

En fin de chaque commande d'introduction de module binaire (LINK, BRAN, LOOK), LKLOAD indique quel est l'état d'avancement de la génération du point de vue de la résolution des externes. Si tous les externes sont définis, aucun message n'est émis sur l'organe de dialogue. S'il existe des références non résolues dans la racine, le message

WARN 15

est émis.

S'il existe des références non résolues dans la branche en cours, le message

WARN 16

est émis.

Pour avoir une image mémoire s'exécutant correctement, il est impératif qu'en fin de branche toutes les références relatives à la branche soient résolues (plus de WARN 16) et qu'en fin de génération toutes les références relatives à la racine soient résolues également (pas de message).

On peut obtenir à tout moment la liste des références non résolues par la commande STAT.

On peut également obtenir la liste des externes définis avec leur adresse par la commande DEFINE :

- émise avant ENDL dans le cas du mode "32 K",
- émise après ENDL dans le cas du mode "64 K".

4.2.5 - Définition du contexte de génération

Outre la commande PERFORM signalée auparavant, qui permet de définir le fichier image mémoire résultat, il existe deux autres commandes définissant le contexte de la génération. Il s'agit des commandes OPTION et MODE.

La commande MODE permet de spécifier :

- le type de géographie : 32 K ou 64 K
- la nature de l'image mémoire générée : Maître ou Esclave
- éventuellement les adresses d'implantation
- le stockage de la liste de génération dans l'article LABEL
- la taille de l'image mémoire à générer (même paramètre que la géographie). En géographie 64 K cette taille est ignorée,.
- le nombre de branches de l'image mémoire (ignoré en géographie 64 K),
- la priorité de l'image mémoire si celle-ci est une tâche,
- la taille de la table des externes à utiliser (ignoré en géographie 64 K).

En l'absence de commande MODE, LKLOAD prend par défaut :

- géographie 64 K,
- mode esclave
- recopie de la liste de génération dans l'article LABEL.

Les autres paramètres sont implicites.



La commande OPTION permet de définir un contexte de génération différent de celui existant par défaut.

Rappelons en effet :

- que la liste de génération est produite sur l'unité symbolique EL à raison d'une section par ligne (monocolonne),
- que les COMMON FORTRAN ne sont pas considérés comme des externes,
- que l'on obtient en sortie un fichier image mémoire.

La commande OPTION permet d'activer les options suivantes :

- COMMON : dans ce cas, les COMMON FORTRAN sont considérés comme des externes au niveau de la génération,
- LINK : le processeur LKLOAD fonctionne comme un éditeur de liens et génère du binaire translatable au lieu de générer un fichier image mémoire,
- LIST : la liste de génération est produite sur l'unité symbolique LO en format pleine page (huit colonnes),.
- NOLIST : aucune impression relative à la liste de génération n'a lieu. Le message :

"NO LIST REQUIRED"

est émis.

- . Ces options ne sont prises en compte qu'en géographie "64 K". Elles sont ignorées sinon.

4.2.6 - Scrutation automatique de bibliothèques

Le processeur LKLOAD possède un mécanisme de scrutation automatique de bibliothèques, évitant à l'utilisateur d'émettre des commandes LOOK lors des générations.

Ce mécanisme :

- est toujours actif lorsque les binaires introduits sont issus des compilateurs FORTRAN 66, FORTRAN 77 et PASCAL,
- est rendu actif par une commande LIBRARY lorsque les bibliothèques à scruter sont des bibliothèques autres que les bibliothèques standards des langages précédents.

Ce mécanisme fonctionne comme suit :

- Lors de l'émission d'une commande STAT, BRAN ou ENDL, le processeur LKLOAD, avant d'exécuter la commande elle-même, si le mécanisme est actif (voir ci-dessus), entreprend la scrutation :
 - . des bibliothèques standards relatives aux trois langages concernés dans l'ordre adéquat,
 - . des bibliothèques utilisateurs spécifiées dans la commande LIBRARY. Ces bibliothèques sont scrutées avant les bibliothèques standards des langages.
- Lorsque le mécanisme est activé il y a impression du message :

"SEARCHING IN :" nom de bibliothèque

chaque fois qu'une bibliothèque est scrutée.



Il faut bien avoir à l'esprit que ce mécanisme a lieu AVANT l'exécution de la commande proprement dite qui l'a activé.

D'autre part, la commande LIBRARY permet de spécifier l'espace disque support des bibliothèques standards lorsque celui-ci n'est pas la FU classique : D2.

4.2.7 - Utilisation de la CDA

Les langages FORTRAN 77 et PASCAL autorisent l'implantation de variables dans la CDA. Cette allocation est effectuée par la chaîne de préparation de programme. Cette allocation étant commune à plusieurs images mémoire, il est nécessaire que la géographie de la CDA soit conservée entre les générations des diverses images-mémoire concernées.

Pour assurer cette cohérence, l'utilisateur doit donc fournir un fichier de stockage de cette géographie et indiquer si cette géographie existe déjà dans ce fichier ou si on doit l'initialiser.

Deux commandes sont disponibles pour assurer ces fonctions :

- la commande "CDA" : qui permet de spécifier le nom du fichier de cohérence CDA. Cette commande est obligatoire à partir du moment où l'on traite des modules binaires comportant des accès à la zone CDA.
- la commande "ICDA" : qui, émise avant la commande CDA, spécifie qu'il s'agit d'une phase d'initialisation de la géographie CDA. Cette commande peut être suivie d'une liste de noms de "blocs CDA" forgeant cette géographie à la configuration voulue par l'utilisateur. En l'absence de cette liste, les blocs CDA sont implantés selon leur ordre d'arrivée.

4.3 - DESCRIPTION DETAILLEE DES COMMANDES

4.3.1 - Mode : définition des paramètres de la génération

But : La commande MODE permet à l'utilisateur de définir les paramètres de génération du programme concerné :

- mode d'exécution (maître ou esclave),
- adresse d'implantation si le programme doit s'exécuter en mode maître,
- caractéristiques du programme (taille, nombre de branches si le programme est segmenté),
- type de géographie.

Elle lui permet en plus de demander la création dans le fichier image mémoire de l'article LABEL mémorisant la cartographie du programme.

Syntaxe :

$$\text{MODE } \left\{ \begin{array}{l} M, [/] \text{ adr1} \\ S \\ T, [/] \text{ adr1} \\ A, [/] \text{ adr1} \end{array} \right\} \left[, \text{nbran} \left[, [\text{taille}] \left[, \begin{array}{l} M[\text{AP}] \\ N[\text{OMAP}] \end{array} \right] \left[, [/] \text{ adr2} [, n] \right] \right] \right] \text{ (CR)}$$

où :

M : Programme exécutable en mode maître
s : Programme exécutable en mode esclave (implanté en 0 de la partition)
T : Programme exécutable en mode esclave (implanté à une adresse $\neq 0$ dans la partition)
A : Programme absolu exécutable en mode maître.

adr1 Adresse d'implantation du programme.
Dans le cas où une adresse adr2 est fournie (image mémoire avec un trou entre la fin de l'overlay et les instructions de la racine en géographie 64 K) il s'agit de l'adresse de la première partie de l'image mémoire (données racine + overlay).
Le caractère / permet de spécifier l'adresse de fin d'implantation (adresse aval). Ce caractère n'est valide qu'en mode "64 K".

nbran Nombre de branches dans le cas d'un programme segmenté
 $2 \leq \text{nbran} = \text{nombre décimal} \leq 255$

taille Taille du programme exprimée en K mots
 $1 \leq \text{taille} = \text{nombre décimal} \leq 64$
dans le cas où ce paramètre est supérieur à 32 l'image mémoire générée à une géographie 64 K.

MAP Génération de la cartographie de l'image mémoire, de la date de génération et de l'IDP, s'il existe.

NOMAP Génération de la date de génération et de l'IDP seulement.

adr2 Adresse d'implantation des instructions racine (adresse amont) le caractère $\textcircled{/}$ permet de spécifier l'adresse de fin d'implantation des instructions racine (adresse aval). L'emploi de cette adresse est réservé à la génération et à l'implantation des produits du catalogue SOLAR.

Taille de la table des externes exprimée en K mots.

Remarques :

- 1) Pour le mode A le paramètre adr 2 n'est pas autorisé (usage réservé).
- 2) Pour les modes A ou T, lorsque les adresses sont supérieures à 32 K, le message WARN 22 est imprimé sur l'unité symbolique EL, mais la génération de l'image mémoire est correcte.
- 3) Une connaissance à priori du nombre d'externes du programme permet une déclaration suffisante pour la taille de la table des externes à l'aide du paramètre n, et évite ainsi une saturation des tables de l'éditeur de liens provoquant l'erreur ERK 04. (l'éditeur de liens a besoin de 6 mots par externe).
- 4) Les paramètres nbran et taille permettent de réaliser une gestion optimisée de l'occupation sur disque par leur connaissance à priori par le chargeur-éditeur. Un sous-dimensionnement de l'image mémoire est considéré comme une erreur fatale par LKLOAD qui imprime :

ERK 09

ou

ERK 10

Lorsque la taille effective ou le nombre de branches effectif sont supérieurs aux paramètres spécifiés par la commande MODE.

- 5) Le paramètre taille sert à préciser :

- le type de géographie (32 K ou 64 K selon que taille > 32 ou non),
- la taille de l'image mémoire à générer.

Dans le cas de la géographie 64 K, les paramètres taille, nombre de branches, taille de la table des externes sont ignorés, car le processeur optimise en calculant les valeurs adéquates. Dans le cas de la géographie 32 K, les valeurs spécifiées sont prises en compte et peuvent donc donner lieu à des erreurs :

- ERK 09 ou
ERK 10

en cas de sous dimensionnement de l'image mémoire

- ERK 04 en cas de saturation de la table des externes

La géographie 64 K est donc à utiliser systématiquement pour éviter ce type d'erreur.

- 6) Lors du traitement l'éditeur-chargeur imprime sur l'unité symbolique EL (ou sur LO si option LIST) :

- La date et l'heure de génération de l'image mémoire.
- L'identification du programme obtenue par le jeu de la directive assembleur IDP.
- Les noms des différentes sections constituant le programme ainsi que leurs adresses d'implantation.
- Les adresses limites ainsi que l'adresse de lancement du programme (BEGIN - END - RUN).
- Type des tâches (Matériel ou Logiciel) et leur niveau de priorité lorsque le programme est structuré en tâches.

Ces informations constituent la cartographie de l'image mémoire.

Elles sont sauvegardées dans l'article LABEL du fichier image mémoire lorsque la commande MODE comporte le paramètre MAP.

Leur liste pourra être, par la suite, obtenue par l'intermédiaire d'EDIT 16 ou du processeur' FTPD.



7) La commande MODE n'est pas obligatoire. Par défaut LKLOAD prend en effet les décisions suivantes :

- S (programme exécutable en mode esclave),
- Géographie 64 K
- MAP (demande de la cartographie de l'image mémoire).
- Les autres paramètres sont calculés à leurs valeurs optimales.

8) La commande MODE doit être émise avant la première commande LINK de la génération. Aucun des paramètres n'est obligatoire. Toutefois le rôle de chacun d'eux est fonction du nombre de virgules qui le précèdent.

Dans le cas d'un programme exécutable en mode maître, l'adresse d'implantation par défaut est '40.

En ce qui concerne les autres paramètres les décisions par défaut sont spécifiées dans la remarque précédente.

Exemples :

MODE S,,16	Mode esclave, 28 branches au plus, taille inférieure ou égale à 16 K, MAP.
MODE M,,,, NOMAP	Mode maître, implantation en '40, 28 branches au plus, taille inférieure ou égale à 8 K, NOMAP.
MODE, M,'1000, 45, 16	Mode maître, implantation en '1000, 45 branches, taille inférieure ou égale à 16 K, MAP.
MODE M,/'7000,,64,M,'9000	Mode maître, fin d'implantation des "données + overlay" '7000, taille inférieure ou égale à 64 K. MAP, implantation des instructions en '9000

4.3.2 - PERFORM Option de performance

- But :
- La commande PERFORM a pour objectif de permettre, lors de l'exécution d'un programme segmenté, l'alimentation des branches en un minimum d'accès disque.
- Dans ce but :
- La taille des articles constituant les branches est arrondie à un multiple de secteur.
 - Un ensemble d'informations système est créé sur disque définissant pour le fichier une organisation physique directe (ce traitement est équivalent à celui réalisé par la commande FAST de l'utilitaire FUP 10).

Syntaxe :

```
PERFORM nom fic [ - catg ], [FU] (SU) (CR)
```

nomfic : nom du fichier image mémoire
catg : catalogue du fichier image mémoire (par défaut le catalogue est nul).
FU : nom de l'unité fonctionnelle disque (par défaut FU du Job en cours).
SU : nom de l'unité symbolique connectée à l'unité fonctionnelle disque.

Remarques

- 1) La commande PERFORM doit être émise avant la première commande LINK.
- 2) Le fichier image mémoire généré par LKLOAD est temporaire. En fin de traitement, si celui-ci n'a donné lieu à aucune erreur fatale, le fichier est transformé en un permanent dont le nom et le catalogue ont été précisés par la commande PERFORM.

Exemple :

```
CALL LKLOAD  
MODE M, '1300,45  
PERFORM PROC - : S
```

4.3.3 - LINK : édition de liens

But : La commande LINK permet de lancer le traitement du ou des MOL désignés.

Syntaxe :

LINK	mol	[/mol...]	[;]	(cr)
avec :				
mol:: =	{	[nomart.]	nomfic	[- catg]
	*			{FU}
				{SU}
				}

Si mol ::= "*", le binaire est lu sur l'unité symbolique BI.

Cette facilité n'est admise que pour le 1er mol de la première commande LINK.

nomart : nom d'un article du fichier qui doit alors être indexé (par défaut le fichier est de type séquentiel),

nomfic : nom du fichier,

catg : catalogue du fichier (par défaut le catalogue est nul),

FU : support sur lequel se trouve le fichier (par défaut on utilise l'unité fonctionnelle déclarée dans la commande JOB).

s u : unité fonctionnelle connectée à une FU disque.

Remarques :

- 1) En utilisant LINK l'utilisateur peut commander l'édition de plusieurs MOL. Dans ce cas les paramètres correspondants sont séparés par des caractères / (max 4 paramètres).
- 2) Le caractère ; en fin de commande LINK permet d'indiquer la fin du traitement.

Exemples :

```
1) CALL LKLOAD
   BI HR
   MODE M, '100
   LINK * ;
```

```
2) CALL LKLOAD
   LINK ART1. NOM1 - CT, D3
   LINK NOM2
   LINK ART3. NOM3 ;
   CATA IM, NOM4 - C4
```

```
3) CALL LKLOAD
   LINK ART1, NOM1 - CT, D3/NOM2/ART3. NOM3;
   CATA IM, NOM4 - C4
```

Les exemples 2 et 3 sont équivalents.

4.3.4 - BRAN définition de branche

But : La commande BRAN permet de définir la structure d'un programme segmenté. Elle doit être utilisée à chaque origine de branche. Elle comporte en paramètre le nom de la branche.

Syntaxe :

```
BRAN nombran : mol [/mol...] [ ; ] (cr)
avec :
mol ::= { [nomart.] nomfic [-catg] [ {FU} ] }
      { [SU] }
```

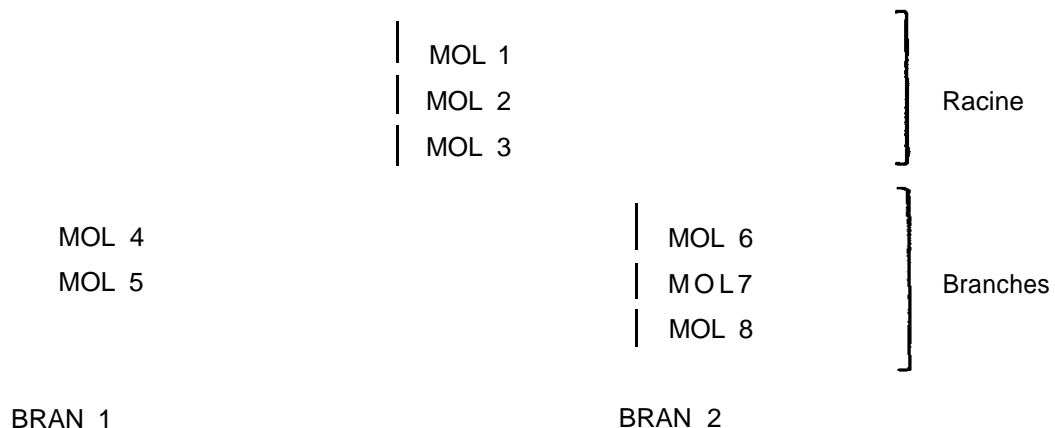
nombran : nom de la branche (6 caractères au plus),
 nomart : nom d'un article du fichier qui doit alors être de type indexé (par défaut le fichier est de type séquentiel),
 nomfic : nom du fichier,
 catg : catalogue du fichier (par défaut le catalogue est nul),
 FU : support sur lequel se trouve le fichier (par défaut on utilise l'unité fonctionnelle déclarée dans la commande JOB).
 su : unité symbolique connectée à une FU disque.

Remarque :

Les caractères $(/)$ et $(;)$ ont le même rôle que pour la commande LINK. Lorsque la commande comporte $(;)$ elle indique la fin du traitement. Aucune commande LINK ou BRAN ne peut être émise par la suite. Cette commande peut activer le mécanisme de scrutation automatique de bibliothèques (voir § 4.2.6).

```
Exemple CALL LKLOAD
MODE M,,,16
LINK MOL1/MOL2/MOL3
BRAN BRAN1 : MOL4/MOL5
BRAN BRAN2 : MOL6
LINK MOL7/MOL8 ;
```

Les commandes précédentes donnent lieu à la structure suivante :



4.3.5 - LOOK scrutation de bibliothèque

But :

La commande LOOK permet de scruter une ou plusieurs bibliothèques afin de satisfaire les références externes non résolues.

Syntaxe :

```
LOOK  nombib [/nombib ... ] [ ; ] (cr)
avec
nombib :: = nomfic [ - catg ] [ {FU}
                               {SU} ]
```

où

nomfic : nom du fichier indexé constituant la bibliothèque,

catg : catalogue du fichier (par défaut le catalogue est nul),

FU : support sur lequel se trouve le fichier (par défaut on utilise l'unité fonctionnelle déclarée dans la commande JOB).

s u : unité fonctionnelle connectée à une FU disque.

Remarques :

- 1) Les caractères (/) et (;) jouent le même rôle que pour la commande LINK. Il est ainsi possible par une seule commande de commander la scrutation de plusieurs bibliothèques. Le caractère (;), lorsqu'il apparaît en fin de commande, indique la fin du traitement.
- 2) La scrutation d'une bibliothèque est demandée afin que LKLOAD satisfasse les références externes non résolues :
 - . Tout symbole externe non défini est recherché en tant que nom d'article dans la bibliothèque.
 - . Si l'article existe, il est considéré comme un MOL qui est alors traité par l'éditeur-chargeur.
 - . Si le MOL entrant comporte des références externes non résolues la bibliothèque est scrutée de nouveau.
 Une précaution doit être prise : Faire comporter dans le MOL la définition du symbole externe de même nom que l'article.
- 3) La segmentation interdit qu'une branche fasse référence à un symbole défini dans une autre branche. Par conséquent, lorsqu'en fin de branche il existe des références apparues dans la branche et non résolues après la scrutation des diverses bibliothèques, l'éditeur-chargeur force à la valeur zéro les adresses correspondantes.

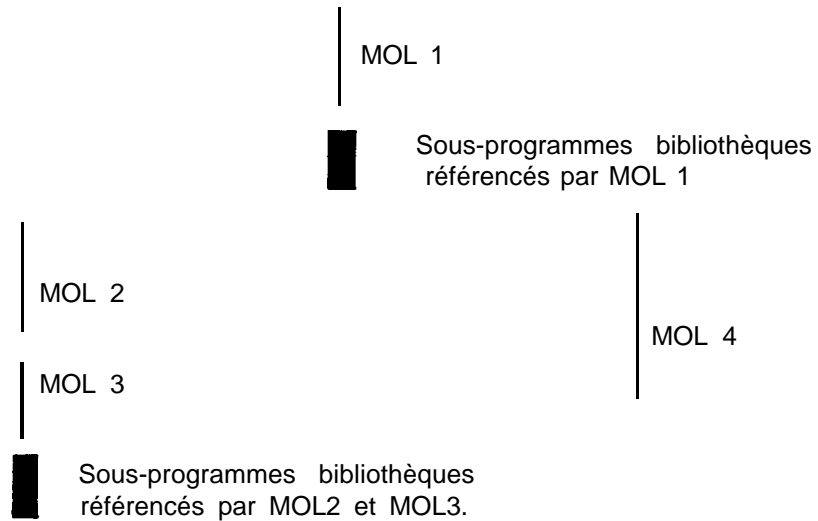
Exemples :

```
1) CALL    L K L O A D
   LINK    MOL1/MOL2
   LOOK    BIB1/BIB2/BIB3 ;
```



```
2) CALL    LKLOAD
   LINK    MOL1
   LOOK    BIB1
   BRAN    BRAN 1 : MOL2/MOL3
   LOOK    BIB1/BIB2, D3
   BRAN    BRAN2 : MOL4 ;
```

Les commandes précédentes donnent lieu à la structure suivante :



4.3.6 - STAT impression des références non résolues

But :

La commande STAT permet à tout moment d'obtenir l'édition sur l'unité symbolique EL des références non résolues.

Syntaxe :

```
STAT (cr)
```

Remarque : cette commande peut activer le mécanisme de scrutation automatique de bibliothèques (voir § 4.2.6).

4.3.7 - DEFINE : impression des références résolues

But :

La commande DEFINE permet d'obtenir l'édition des références résolues ainsi que des adresses qui leur sont associées.

Syntaxe

DEFINE 

Remarque :


En mode "32 K" la commande DEFINE doit être frappée avant la commande ENDL. En mode "64 K", elle ne peut être utilisée qu'après ENDL.

4.3.8 - ZDEF : remise à zéro de symboles externes définis

But :

La commande ZDEF permet la remise à zéro de tous les symboles externes déjà définis, donnant ainsi la possibilité de "linkéditer" plusieurs fois la séquence relative à un symbole externe.

Syntaxe


ZDEF 

4.3.9 - ENDL : Fin de traitement

But :


La commande ENDL est utilisée pour indiquer la fin du traitement.

Syntaxe

ENDL 

Remarques :

1) Il est indispensable d'indiquer à LKLOAD la fin du traitement.

Cette opération peut être réalisée par l'émission de la commande ENDL ou par celle du caractère  en fin de commande LINK, BRAN ou LOOK.

Elle a pour effet la fermeture des fichiers temporaires de travail utilisés par LKLOAD ainsi que la génération dans le fichier image mémoire des informations système contenues dans l'article DESC. De plus, lorsque l'option PERFORM a été demandée, LKLOAD transforme le fichier image mémoire en un fichier permanent dont le nom et le catalogue ont été donnés en paramètres de la commande.

2) Lorsqu'il subsiste en fin de traitement des références qui n'ont pas été résolues LKLOAD force à la valeur zéro les adresses qui correspondent aux symboles externes non définis.

3) Cette commande peut activer le mécanisme de scrutation automatique de bibliothèques (voir § 4.2.6)

4.3.10 - OPTION : spécification d'options

But :

La commande OPTION permet de définir des options modifiant le mode de fonctionnement standard de LKLOAD.

Syntaxe

```
OPTION option1 [ , option2, . . . ]
```

Quatre options sont disponibles :

- LIST : cette option permet d'avoir une liste de génération en format pleine page sur l'unité symbolique LO.
- NOLIST : cette option permet de supprimer la sortie de la liste de génération. Dans ce cas, le message :

NO LIST REQUIRED

est émis.
- COMMON : cette option permet de définir les COMMON FORTRAN en tant qu'externes. Attention : ceci peut être dangereux en cas d'homonymie entre des noms de COMMON et des noms de sous-programmes (utilisateurs ou systèmes). Tout COMMON utilisé en tant qu'externe donne lieu à l'émission du message :

WARN 23 Nom de COMMON

afin que l'utilisateur s'assure qu'il n'y a pas de problèmes de conflit de nom.
- LINK : cette option permet de faire fonctionner le processeur LKLOAD comme un éditeur de liens. Dans ce cas, on n'a pas d'image-mémoire de générée : on obtient un binaire translatable généré dans l'unité symbolique BO.
Le binaire est généré selon la géographie 64 K.
Cette option annule l'effet de la commande MODE si celle-ci a été émise auparavant.

Remarque :

Ces options ne sont prises en compte que dans le cas de la géographie "64 K". En mode "32 K", elles sont ineffectives.

La commande OPTION doit être émise avant la première commande LINK de la génération.

4.3.11 - LIBRARY : scrutation automatique de bibliothèques

But :

La commande LIBRARY permet :

- de spécifier des bibliothèques utilisateurs à scruter lors de l'activation du mécanisme de scrutation automatique de bibliothèques,
- de modifier les paramètres des bibliothèques standards qui sont scrutées automatiquement en FORTRAN 66, FORTRAN 77 et PASCAL :
 - . FU support
 - . catalogue (-FN ou - VS) dans le cas du FORTRAN 66.



Syntaxe LIBRARY nombib [/ nombib]

avec $\text{nombib} ::= \left\{ \begin{array}{l} \text{LIBSTD} \left[\begin{array}{l} \{- \text{FN} \} \\ \{- \text{VS} \} \end{array} \right] \left[, \begin{array}{l} \{ \text{SU} \} \\ \{ \text{FU} \} \end{array} \right] \\ \text{nomfic} \left[- \text{catg} \right] \left[, \begin{array}{l} \{ \text{SU} \} \\ \{ \text{FU} \} \end{array} \right] \end{array} \right\}$

LIBSTD : désigne l'ensemble des bibliothèques standards associées aux langages FORTRAN 66, FORTRAN 77, PASCAL :

- en FORTRAN 66 :

BIBFOR-FN	BIBIC-FN	BIBDBL-FN
BIBFOR-VS	BIBFIC-VS	BIBDBL-VS
BIBUTI-FN	BIBSTD-FN	
- en FORTRAN 77

BIBFOR-F7	BIBFIC-F7	BIBDBL-F7
BIBUTI-F7	BIBSTD-F7	
- en PASCAL

BIBPAS-RT

Dans le cas du FORTRAN 66, on peut préciser le catalogue souhaité :

- FN cas standard
 - VS utilisation de l'option VS.
- SU } désigne le support disque de la ou
FU } des bibliothèques

Les bibliothèques utilisateurs sont scrutées avant les bibliothèques standards.

En l'absence des paramètres SU, FU c'est la FU banale de l'utilisateur (ou FU du JOB en cours) qui est prise par défaut.

Le mécanisme de scrutation des bibliothèques est appelé avant l'exécution des commandes suivantes :

- STAT
- BRAN
- ENDL

lorsqu'il est actif, c'est-à-dire :

- si la génération comporte des modules FORTRAN 66, FORTRAN 77 ou PASCAL,
- si une commande LIBRARY spécifiant une bibliothèque utilisateur a été émise.

Remarques :

- La commande LIBRARY accepte au plus 4 paramètres (LIBSTD inclus).
- La commande LIBRARY doit être émise avant la première commande LINK de la génération.
- Le message :

SEARCHING IN nombib

indique que la bibliothèque spécifiée est en cours de scrutation.

4.3.12 - CDA : désignation du fichier de cohérence CDA

But :

La commande CDA permet de spécifier le nom du fichier de cohérence CDA (voir § 4.2.7).

Syntaxe

```
CDA nomfic [-catg-] [, {SU }  
FU } ]
```

nomfic - catg : fichier séquentiel FMS

su }
FU } : support disque du fichier

La commande CDA doit être émise chaque fois qu'une génération implique des modules faisant appel à la CDA en FORTRAN 77 ou PASCAL.

Le fichier spécifié :

- doit exister si l'on n'est pas en phase de création de la géographie CDA (voir commande ICDA),
- peut exister (dans ce cas il sera écrasé) ou sera créé s'il n'existe pas lorsqu'on est en phase de création de la géographie CDA.

Dans le cas où le fichier de cohérence est utilisé en lecture (hors phase initialisation), ce fichier doit comporter la description COMPLETE de la CDA. En effet, un module ne peut rajouter de zone dans la CDA en dehors de la phase d'initialisation (commande ICDA émise).

Si l'on omet la commande CDA ou si l'on essaie d'agrandir la CDA hors initialisation, le message

ERK 17

est émis.

Remarques :

- La commande CDA doit être émise avant la première commande LINK de la génération.
- En fin de génération, la géographie de la CDA est imprimée.

4.3.13 - ICDA : initialisation de la géographie CDA

But

Créer la géographie de la CDA en décrivant l'ordre d'implantation des blocs CDA dans la CDA.

Syntaxe

```
ICDA nom CDA, . . . . , nom CDA
```

nom CDA : nom du COMMON FORTRAN ou de la variable PASCAL implanté dans la CDA

Les variables CDA sont implantées dans la CDA selon leur ordre de description dans la commande. On peut émettre plusieurs commandes ICDA consécutives lorsque le nombre de variables CDA est important.

Si la liste de variables CDA est vide, dans ce cas l'ordre d'implantation dans la CDA est l'ordre d'arrivée des variables CDA dans la phase de génération.

Si l'ordre d'implantation des variables dans la CDA n'a pas d'importance, il suffit donc d'émettre la commande ICDA seule.

Remarques :

- La commande ICDA doit être émise avant la commande CDA.

4.4 - MESSAGES D'ERREUR -CODE DE RETOUR

Messages d'erreur

LKLOAD détecte un certain nombre d'erreurs et les signale à l'utilisateur en imprimant des messages sur l'unité symbolique EL.

Certaines sont fatales. D'autres ne sont en fait que de simples avertissements.

Les erreurs fatales donnent lieu à des messages du type :

ERK n [inf]

n représente le numéro de l'erreur.

Dans le cas de l'erreur 12 (erreur logique ou matériel détectée par FMS) LKLOAD fournit en plus le numéro de l'erreur logique (inf)

ERK 12 '6001

Les avertissements donnent lieu à des messages du type :

WARN n

n représentant le numéro de l'avertissement.

Les numéros des erreurs et avertissements sont fournis en annexe.

Un certain nombre de commandes peuvent comporter plusieurs paramètres séparés alors par des caractères (/).

Dans ce cas les messages comportent en plus la spécification du module en cours de traitement lors de la détection de l'erreur.

Ainsi après la commande:

LINK FIC1,D3/ART2.FIC2/FIC3

le message :

MODULE 3 : ERK 09

précise la détection de l'erreur au cours du traitement du troisième module, c'est-à-dire de FIC3.

Après une erreur fatale, toutes les commandes qui suivent sont rejetées avec impression du message :

ERK 13

CODE DE RETOUR

Lorsque LKLOAD rend le contrôle au système après chaque traitement de commande, il lui communique dans le registre accumulateur un code de retour. Celui-ci est mémorisé car il permet à l'utilisateur de connaître la façon dont s'est exécutée la dernière phase de travail. Le code de retour peut être testé par la commande IF du système.

Les valeurs possibles pour le code de retour transmis par LKLOAD sont les suivantes :

R = 0	Traitement correct
R = 8	Il existe des références non résolues
R = 8	Erreur fatale.

4.5 - CONSEILS D'UTILISATION

4.4.1 - Encombrement et performances

LKLOAD occupe environ 10 Kmots et utilise la zone libre restant à sa disposition pour y implanter la table des externes et les buffers du système de pagination. Les performances sont d'autant meilleures que la taille de la zone libre est grande. LKLOAD sait utiliser la zone libre jusqu'à 64 K.

LKLOAD utilise d'autre part les options de performance permises par FMS-E. Il effectue en particulier une primitive IRTIX afin d'accélérer la scrutation des bibliothèques.

4.4.2 - Place disque nécessaire

LKLOAD doit disposer sur disque de la place nécessaire pour créer les fichiers suivants :

- fichier temporaire de stockage des commentaires,
- fichier de pagination de la taille précisée dans la commande MODE,
- fichier indexé support de l'image mémoire produite.

En géographie 64 K, LKLOAD stocke également tous les MOL référencés dans un fichier temporaire.

4.4.3 - Conseils pratiques

- 1) Ne pas oublier que le fichier image mémoire est généré dans l'unité fonctionnelle disque déclarée par la commande JOB, si la commande PERFORM ne contient pas de paramètre FU ou SU.
- 2) Lorsque les MOL ainsi que les bibliothèques à traiter n'appartiennent pas à cette FU, préciser par commande (LOOK, LINK, BRAN) la FU support du fichier correspondant.
- 3) LKLOAD effectue ses sorties de messages (avertissements, cartographie, liste des symboles externes) sur l'unité symbolique EL (ou sur LO si option LIST).
Si la configuration comporte une imprimante rapide, affecter LP aux unités concernées de manière à améliorer les performances de l'éditeur-chargeur.
- 4) En l'absence de l'option PERFORM ne pas oublier de cataloguer le fichier temporaire constituant l'image mémoire en fin de traitement.
- 5) Dans les cas standards de production de programme en langage évolué, la procédure-type à utiliser est la suivante :

```
CALL LKLOAD  
LINK MOL-BT  
STAT  
ENDL
```

LKLOAD prenant à sa charge la scrutation des bibliothèques systèmes.

ANNEXE 1

TABLEAU RECAPITULATIF DES POSSIBILITES ENTRE LES DIFFERENTES COMMANDES
ET LES PRINCIPAUX PARAMETRES.

	MLOD	SLOD	ALOD	TLOD
$0 < \text{paramètre taille} < 32$	OUI	OUI	OUI	NON
$32 < \text{paramètre taille} \leq 64$	OUI	OUI	NON	OUI
Adresse d'implantation $\leq 32 \text{ K}$	OUI	ZERO	OUI	OUI
Adresse d'implantation $\geq 32 \text{ K}$	NON	NON ZERO	OUI	OUI

ANNEXE 2

MESSAGES D'ERREUR EMIS PAR LE CHARGEUR DISQUE

Message	Signification
ERU 11	* Erreur de parité
ERU 12	* Erreur dans le programme source (non binaire translatable)
ERU 13	* "Overlay" interdit
ERU 14	* Adresse d'implantation de donnée supérieure à 32 K (MLOD et SLOD)
ERU 15	* Sous-dimensionnement de l'image mémoire (taille)
ERU 16	* Erreur de "checksum"
ERU 17	* Adresse calculée supérieure à 32 K
ERU 18	* Erreur logique détectée par FMS
ERU 19	* Tâche matériel ayant un niveau de priorité introduit par commande supérieur à 15
ERU 20	Commande d'activation erronée
ERU 21	PST incorrecte : mode d'exécution de la tâche non compatible avec la commande d'activation
ERU 22	* Sous-dimensionnement de l'image mémoire (nombre de-branches)
ERU 23	* Incompatibilité entre le type de BT et le type d'IM (paramètre taille)
ERU 24	Zone libre insuffisante
ERU 25	Adresse supérieure à 64 K

* Erreurs fatales : les autres erreurs signalées par le chargeur disque sont de simples avertissements.

Dans le cas de l'erreur de numéro 21 le mode d'exécution précisé par la commande d'activation est forcé dans la PST (valeur initiale du registre S).

ANNEXE 3

NUMEROS DES ERREURS EMIS PAR LKLOAD ET L'EDITEUR DE LIENS

Message	Signification
ERK 01	Erreur de parité.
ERK 02	Erreur dans le programme source (non binaire "link-éditable" ou translatable).
ERK 03	Elément de bibliothèque incorrect (l'article "link-édité" ne comporte pas la définition du symbole externe de même nom que celui de l'article).
ERK 04	Saturation des tables de l'éditeur de liens.
ERK 05	Passage du compteur au-delà de 32 K (taille des données > 32 K)
ERK 06	Constante adresse supérieure à 32 K.
ERK 07	Enchaînement table des sections - programme incorrect dans le cas de PL 16.
ERK 08	Enchaînement des sections incorrect dans le cas de FORTRAN.
ERK 09	Sous-dimensionnement de l'image mémoire (taille).
ERK 10	Sous-dimensionnement de l'image mémoire (nombre de branches).
ERK 11	Erreur dans le programme source (comporte plus de 16 PST).
ERK 12	Erreur détectée par FMS.
ERK 13	Erreur de syntaxe dans la commande d'activation ou commande illogique.
ERK 14	Erreur sur le chargement des branches.
ERK 15	Adresse supérieure à 64 K
ERK 16	Zone libre (FREEM) insuffisante
ERK 17	Erreur relative à la gestion de la CDA
ERK 18	Incompatibilité entre les adresses d'implantation et la taille du programme

ANNEXE 4

NUMEROS DES AVERTISSEMENTS EMIS PAR LKLOAD ET L'EDITEUR DE LIENS

Message	Signification
WARN 15	Existence de références non résolues dans la racine.
WARN 16	Existence de références non résolues dans la branche en cours.
WARN 17	Multiplés adresses de lancement (la dernière est prise en compte)
WARN 18	Symbole externe déjà défini (la première définition est retenue).
WARN 19	PST incorrecte (mode d'exécution de la tâche indiqué dans la PST non compatible avec la commande d'activation) : le mode précisé par la commande est forcé dans la PST (valeur initiale du registre S).
WARN 20	Erreur de "checksum".
WARN 21	Commande d'activation erronée.
WARN 22	Données débordant de 32 K.
WARN 23	COMMON FORTRAN utilisé en tant qu'externe
WARN 24	Image mémoire esclave supérieure à 32 K implantée en 0 de la partition

ANNEXE 5

Programmation avec des données implantées au delà de 32 K.

5.1 - AVERTISSEMENT

L'utilisation des règles qui suivent est délicate pour un utilisateur non averti du système d'adressage sur SOLAR.

Aussi, chaque fois que cela est possible, il est préférable d'utiliser les possibilités standards de la chaîne de préparation de programme.

La programmation qui est décrite ci-après est réservée aux langages Assembleur et PL 16 qui permettent à l'utilisateur d'être maître de l'implantation et du mode d'adressage de ses données. Elle est sans objet dans le cas des langages de haut niveau (FORTRAN, PASCAL, etc...) qui utilisent la géographie 64 K autorisée par la chaîne de production de programme.

5.2 - RAPPELS

5.2.1 - L'adressage sur Solar

A l'intérieur d'une partition de 64 K, on dispose de 3 modes d'adressage :

- direct,
- indirect,
- indirect post-indexé.

C'est un adressage basé, c'est-à-dire que le mot mémoire que l'on veut adresser, ou le relais dans le cas d'un adressage indirect, est désigné à l'aide d'un déplacement par rapport à un registre dit de base. Le SOLAR possède 3 registres de base C, L, W qui sont des registres de 16 bits.

Avec 16 bits, on peut adresse 64 K emplacements-mémoire. Autrement dit, on peut adresser DIRECTEMENT sans problème tous les emplacements mémoire d'une partition de 64 K.

5.2.2 - Cas de l'adressage indirect

Lorsqu'une instruction référence mémoire est une instruction indirecte (bit 2 positionné à 1), ceci veut dire que l'emplacement mémoire référencé directement par la base contient l'adresse de l'emplacement que l'on veut atteindre. Or, cette adresse est une adresse 15 bits car le bit 0 du relais d'indirection a une signification particulière. Si ce bit est à 1, cela signifie que l'on veut utiliser un adressage indirect post-indexé, c'est-à-dire que l'adresse réelle de l'emplacement-mémoire spécifié est obtenue en ajoutant aux 15 bits restants du relais, la valeur contenue dans le registre index RX.

Autrement dit, en adressage indirect on ne sait désigner de façon simple qu'une partition de 32 K, car l'adresse contenue dans le relais d'indirection est une adresse 15 bits.

5.3 - COMMENT ADRESSER DES DONNEES AU DELA DE 32 K

Lorsque l'application ou la configuration matérielle nécessitent le déport au-delà de 32 K des données et des instructions, il faut prendre certaines précautions quant à l'adressage des données si l'on ne veut pas avoir de problèmes.

LA REGLE DE BASE A RESPECTER est la suivante :

“NE JAMAIS UTILISER D'INDIRECTION”

Pourquoi est-on amenés dans un programme à utiliser l'indirection ?

Il y a deux raisons principales :

- on veut accéder à des données sous forme de tableau :
 - . soit par pointeur que l'on incrémente,
 - . soit par pointeur avec utilisation du registre d'index Rx,
- on veut accéder à une donnée externe dont on n'aura l'adresse qu'après édition de liens avec le module dans lequel elle est définie.

La seule façon simple d'accéder à un élément de tableau ou à un externe dans le cadre d'une programmation au-delà de 32 K est d'utiliser un ADRESSAGE DIRECT BASE.

Par exemple, supposons que l'on ait dans un mot ADATA l'adresse d'un externe DATA mis par édition de liens en langage assembleur :

```
EXT DATA  
ADATA : WORD DATA
```

L'adresse de DATA étant sur 16 bits, on ne pourra accéder à ce mot de la façon classique par un

```
LA &ADATA
```

mais par une séquence du type :

```
LA    ADATA  
PSR  W  
LR   A, W  
LA   O, W  
PLR  W
```

Cette séquence a pour but de faire pointer la base W qui contient une adresse 16 bits sur le mot que l'on veut atteindre.

Si DATA est un tableau, dont on veut atteindre un élément dont l'indice est précisé dans Rx, il ne faut pas utiliser la méthode classique :

```
EXT DATA  
ADATA : WORD DATA, X
```

```
LX INDICE  
LA &ADATA
```


mais utiliser une séquence analogue à celle citée plus haut :

```
EXT DATA

ADATA : WORD  DATA
      PSR     W
      LA      ADATA
      LR      A, W
      LX      INDICE
      ADR     X, W      Ajout de l'index
      LA      O, W
      PLR     W
```

Les deux exemples que nous avons vu montrent qu'il faut remplacer une instruction simple par une séquence. Comme ce phénomène est, à priori, répétitif, on a intérêt à utiliser un sous-programme spécifique par opération à effectuer sur des données implantées au-delà de 32 K.

Par exemple, un sous-programme de base consiste à simuler le LOADA d'une variable au-delà de 32 K.

Ce sous-programme pourrait s'écrire :

```
LOADA : PSR  W
      LR   Y, W
      LA  O, W
      PLR W
      RSR
```

en supposant que l'adresse est contenue dans le registre RY, donc que la séquence d'appel est du type :

```
EXT. DATA
ADATA : WORD DATA
ALOAD : WORD LOADA

LY  ADATA
BSR ALOADA
```

On notera au passage que l'utilisation d'un relais ALOADA vers le sous-programme LOADA ne pose pas de problème car on s'y branche par une instruction (BSR) DIRECTE.

5.4 - PROGRAMMATION EN PL16

5.4.1 - Généralités

L'utilisation du langage PL16 dans un tel cadre est marginale et il faut toujours se souvenir que si l'utilisation d'une image mémoire de type 64 K convient, cette solution est de loin la meilleure car on pourra écrire son application de manière standard.

Dans les lignes qui suivent, on va donc attirer l'attention sur l'utilisation de ce langage dans le contexte particulier de cet annexe.

5.4.2 - Cas des références externes

Lorsqu'on veut référencer un mot défini en externe dans un autre module, on déclare :

```
REF WORD DATA
```

Si l'on veut accéder à ce mot, pour mettre le contenu dans le registre RA par exemple, on écrit :

```
RA : = DATA
```

Or, bien que le caractère "&" n'apparaisse pas explicitement dans le source PL 16, le compilateur génère une instruction LOADA indirecte, ce qui ne convient pas dans le cadre de la programmation au-delà de 32 K. On doit dans ce cas utiliser la séquence suivante :

```
EXT    WORD DATA ;  
WORD  ADATA = (@ DATA) ;
```

et on utilise un sous-programme simulant le LOADA (en entrée dans RY l'adresse du mot référencé)

```
RY : = ADATA ;  
CALL LOADA ;
```

Dans le cas où l'externe est un sous-programme, on peut toujours utiliser la séquence "standard" :

```
REF PROCEDURE SPEXT ;
```

avec un

```
CALL SPEXT ;
```

car, sur le "CALL SPEXT", le compilateur génère un BSR direct sur le relais implicite généré dans la section où apparaît le "REF PROCEDURE".

5.4.3 - Cas des tableaux

En PL 16, la manipulation de tableaux est rendue très facile par le langage. Or, qui dit manipulation de tableaux, dit adressage indirect post indexé, ce qui n'est pas utilisable au-delà de 32 K.

Lorsqu'on déclare un tableau en PL 16 par :

```
ARRAY N WORD TOTO ;
```

le compilateur réserve N mots dans une section dite de tables, qui est non basée, et génère dans la section de déclaration un relais vers le début des N mots AVEC LE BIT 0 pour l'adressage post indexé.

Toutes les instructions faisant référence au tableau sont alors des instructions indirectes, et on ne peut donc les utiliser.

Par contre, la réservation de tableaux et leur initialisation à la compilation sont des facilités que l'on souhaite souvent pouvoir continuer à utiliser.

De toute façon, il faut, lorsqu'on veut manipuler un tableau implanté au-delà de 32 K, utiliser des sous-programmes de "simulation" d'un adressage post-indexé.



On peut par exemple convenir d'un sous-programme simulant un LOADA indexé auquel on fournit :

- dans RY l'adresse du début du tableau,
- dans RX l'indice de l'élément.

Le tableau est réservé par la déclaration classique :

```
ARRAY N WORD TABLO ;
```

qui permet également de l'initialiser à la compilation :

```
ARRAY N WORD TABLO = (* N (0) ) ;
```

On s'affranchit du bit indexation généré par PL16 en déclarant un pointeur explicite sur le tableau par :

```
WORD ADTABLO = (@ TABLO AND '7FFF) ;
```

On peut alors accéder à un élément du tableau par le sous-programme LOADX comme précédemment :

```
RX := INDICE ;  
RY := ADTABLO ;  
CALL LOADX ;
```

En résumé, lorsqu'on veut déclarer en PL 16 un tableau implanté par la suite au-delà de 32 K. on est obligé de déclarer un 2ème mot initialisé avec l'adresse du tableau moins le bit index ; on a donc 2 relais vers chaque tableau :

- le relais implicite avec le bit index généré sur la déclaration "ARRAY . . .",
- le relais explicite sans le bit index.

5.4.4 - Cas des ruptures de séquence

Dans le cas des ruptures de séquence, c'est toujours la même règle : ne pas utiliser d'instruction de déroutement indirecte. En PL 16, ceci signifie :

- on ne peut avoir des tableaux de procédures ou de labels,
- on ne peut utiliser de "CASE OF",
- par contre, on peut utiliser des "INDIRECT LABEL".

5.5 - CONSEILS PRATIQUES - EXEMPLES DE PROCEDURES DE SERVICE

5.5.1 - Création de l'image mémoire

Pour créer l'image mémoire à partir des modules de l'application, on doit utiliser la chaîne de production de programme.

Si la taille de l'image mémoire est inférieure à 32 K, mais si cette image mémoire doit être implantée au-delà de 32 K, il faut utiliser la commande ALOD du BUILDER (MODE A dans LKLOAD).

Exemple :

```
CALL BUILD  
BI APPLI-BT  
ALOD '8100
```

L'utilisation de la commande ALOD est d'ailleurs restrictive : il faut que le binaire translatable corresponde à des modules respectant les règles énoncées ci-dessus ; sans cela les réactions du BUILDER peuvent être anormales (bouclage, ou ERU 15).

5.5.2 - Exemples de procédures de service

Dans ce paragraphe, on va donner des exemples d'écriture en PL 16 de sous-programmes simulant une instruction indirecte ou indirecte post indexée dans le cadre d'une programmation au-delà de 32 K. Toutes les instructions possibles ne sont pas citées, mais le principe reste le même et l'on pourra adapter le mécanisme du sous-programme au cas de figure spécifique.

Dans tous les exemples qui suivent, l'interface d'entrée et de sortie est supposé le même :

- dans RA la donnée à écrire ou la donnée lue,
- dans RY l'adresse 16 bits du relais vers la donnée,
- dans RX éventuellement l'indice de l'élément du tableau.



- Simulation d'un LOADA indirect :

PROCEDURE LDAI

```
. DUMMY WORKING SECTION DUM
  RES 128 ;
  WORD ACCESS ;
  INSTRUCTION XR (2, '2B80) ;

. USING RW IS DUM ;
  XR (RY, RW) ;
  RA := ACCESS ;
  XR (RY, RW) ;

END ;
```

- Simulation d'un STORE A indirect

PROCEDURE STAI

```
. DUMMY WORKING SECTION DUM
  RES 128 ;
  WORD ACCESS ; (2, '2B80) ;
  INSTRUCTION XR (2, '2B80) ;

. USING RW IS DUM ;
  XR (RY, RW) ;
  ACCESS := RA ;
  XR (RY, RW) ;

END ;
```

- Simulation d'un LOAD A indirect post indexé

PROCEDURE LDAX

```
. DUMMY WORKING SECTION DUM
  RES 128 ;
  WORD ACCESS :
  INSTRUCTION LR (2, '2BC0) ,
                ADR (2, '2C00) ;

. USING RW IS DUM ;
  SAVE (RW) ;
  LR (RY, RW) ;
  ADR (RX, RW) ;
  RA := ACCESS ;
  RESTORE (RW) ;

END ;
```

- Simulation d'un STORE A indirect post indexé

PROCEDURE identique à LDAX avec

ACCESS := RA ;

au lieu de

RA := ACCESS ;

- Simulation d'un LOAD BYTE indirect post indexé

PROCEDURE LDBYX

. DUMMY WORKING SECTION DUM

RES 128;

WORD ACCESS1 ;

BYTE ACCESS2 SYN ACCESS1 ;

INSTRUCTION LR (2, '2BC0) ;

ADR (2, '2C00) ;

. USING RW IS DUM ;

SAVE (RW) ;

LR (RY, RW) ;

RA := RX SARS1 ;

IF (NOT CARRY) THEN

ADR (RA, RW) ;

RA := ACCESS2 ;

ELSE

ADR (RA, RW) ;

RA := ACCESS1 AND'FF ;

END ;

RESTORE (RW) ;

END ;

- Simulation d'un STORE BYTE INDIRECT INDEXE

PROCEDURE STBYX

. DUMMY WORKING SECTION DUM

RES 128 ;

WORD ACCESS1 ;

BYTE ACCESS2 SYN ACCESS1 ;

INSTRUCTION LR (2, '2BC0) ;

ADR(2, '2C00) ;

DIT (0, '1E13) ;

EIT (0, '1E14) ;



```
USING RW IS DUM ;
SAVE (RX, RW) ;
LR (RY, RW) ;
RA := : RX ;
RA := RA SARS 1 ;
IF (NOT CARRY) THEN
    ADR (RA, RW) ;
    RA := RX ;
    ACCESS2 := RA ;

ELSE

    ADR (RA, RW) ;
    DIT ;
    RA := ACCESS2
    ACCESS1 := RX ;
    ACCESS2 := RA ;
    EIT ;
    RA:= RX ;

END ;

RESTORE (RX, RW) ;

END ;
```

Pour cette dernière procédure, on notera la présence des instructions DIT, EIT destinées à se protéger des interruptions éventuelles pendant lesquelles des tâches plus prioritaires viendraient modifier l'octet gauche que l'on ne traite pas mais qu'on recopie.