ÉLECTRICITÉ DE FRANCE
DIRECTION DES ÉTUDES ET RECHERCHES

Service Informatique
et Mathématiques Appliquées

Le 6 Avril 1981

2

VERSION PROVISOIRE

DEPARTEMENT METHODES ET MOYENS
DE L'INFORMATIQUE

1, avenue du Général de Gaulle

92141 CLAMART
Tél. 765.43.21

MAGNIER M.

LE LANGAGE DE COMMANDE DU CRAY-1

MANUEL DE REFERENCE

Note CRAY n° 2

HI/ 3560-01

67 pages

Résumé

Les principaux éléments du langage de commande du CRAY : forme des commandes, constitution et traitement d'un travail, création et gestion des fichiers, compilation et exécution d'un programme.

ACCESSIBILITÉ
☐ Libre
☐ EDF-GDF
☐ Restreint
☐ Confidentiel

AVERTISSEMENT

La présentation des diverses possibilités offertes par le langage de contrôle n'est pas exhaustive bien qu'elle puisse paraître prolixe dans l'énumération de certains paramètres. Pour l'exécution de travaux simples, on se reportera à l'annexe 1, qui donne les commandes correspondant aux cas les plus courants, et mentionne un fichier IBM contenant des "squelettes" réutilisables.

De nombreuses personnes nous ont aidés à améliorer cette note ; nous remercions en particulier M. MIGOUX, l'équipe du GETIA (MM. LE BRIS, BONNEVAL, MORICONI, SCHMITT et SCHOONHEERE), ainsi qu'une personne de la Société CRAY-FRANCE (Mme BEAUCHAMP). Le chapitre VI.2 sur les bibliothèques a été rédigé par Mme BEAUCHAMP qui a également apporté son concours à l'annexe 1 (exemples d'utilisation).

Une fiche prête à être remplie est proposée aux utilisateurs en dernière page ; elle pourrait être retournée à l'auteur afin de signaler les compléments souhaités, les imprévisions et/ou les erreurs rencontrées.

Cette note est encore sous forme provisoire et fera l'objet d'éditions ultérieures.

TABLE DES MATIERES

I. QUELQUES NOTIONS SUR LES FICHIERS

I.1. Fichiers locaux
I.2. Fichiers catalogués
I.3. Fichiers résidents en mémoire
I.4. Règles d'écriture des noms de fichiers
I.5. Formats de fichier
1.5.1. Le format bloqué
1.5.2. Le format non bloqué
I.6. Codes disposition des fichiers

II. CONSTITUTION ET TRAITEMENT D'UN TRAVAIL

II.1. Structure du paquet de cartes d'un travail
II.2. Déroulement d'un travail
II.2.1. Prise en compte
II.2.2. Initialisation
II.2.3. Exécution
II.2.4. Achèvement
II.3. Reprise d'un travail
II.4. Historique du travail et informations comptables

III. STRUCTURE DU LANGAGE DE COMMANDE

III.1. Le verbe
III.2. Les séparateurs
III.3. Les paramètres
III.4. La terminaison
III.5. Les commentaires
III.6 Conventions de codage
III.6.1. Blancs
III.6.2. Cartes suites

IV. MODALITES D'EXECUTION D'UN TRAVAIL

IV.1. La carte de travail : JOB
IV.2. Imputation comptable : ACCOUNT

V. COMPILATION FORTRAN : CFT

VI. CHARGEUR ; BIBLIOTHEQUES
VI.1 Chargement d'un programme : LDR
VI.2 Création et gestion de bibliothèques de programmes : BUILD

VII. INSTRUCTIONS DE GESTION DES FICHIERS

VII.1. Création, accès et fin d'accès à un fichier
VII.1.1. Créer un fichier : ASSIGN
VII.1.2. Libérer un fichier : RELEASE
VII.2. Gestion des fichiers catalogués utilisateurs
VII.2.1. Cataloguer un fichier : SAVE
VII.2.2. Utiliser un fichier : ACCESS
VII.2.3. Détruire un fichier : DELETE

VIII. TRANSFERTS DE FICHIER ENTRE CRAY ET FRONTAL
VIII.1. Frontal vers Cray : ACQUIRE
VIII.2. Cray vers frontal : DISPOSE

Annexe 1 - Procédure d'accès du Cray depuis les IBM de Clamart ; exemples de travaux complets.

Annexe 2 - Format des fichiers à transférer entre IBM et Cray.

Annexe 3 - Bibliographie

INDEX

I. QUELQUES NOTIONS SUR LES FICHIERS

Les informations traitées sur le Cray sont organisées en fichiers. Chaque fichier est identifié par un nom symbolique. Un fichier peut être soit local à un travail, soit catalogué et disponible pour le système et d'autres travaux.

I.1. - Fichiers locaux

Un fichier local n'est utilisable que par le travail qui le crée. Les fichiers locaux peuvent être créés selon deux processus :

- soit explicitement par l'instruction **ASSIGN** du langage de commande (cf. VII.1) ;

- soit implicitement, lors d'une première référence à son nom ou à son numéro logique, lors d'une demande d'écriture ou lors d'un **OPEN.**

Un fichier local est vide jusqu'à la première opération d'écriture. Un **REWIND** ou un **BACKSPACE** est nécessaire avant une lecture. Un fichier local peut être catalogué sur le CRAY par l'instruction **SAVE.**
Si le fichier n'est pas catalogué, il est détruit en fin de travail ou par une instruction **RELEASE** et l'espace occupé est rendu au système. (cf. VII.1.2)

I.2. - Fichiers catalogués

Un fichier catalogué est utilisable par le système et par divers travaux et est conservé au cours des démarrages du système. Les fichiers catalogués sont de deux types :

- les fichiers catalogués des utilisateurs, soit créés sur Cray, soit issus d'un calculateur frontal. Ces fichiers catalogués sont gardés aussi longtemps que l'utilisateur ou l'exploitation le souhaite. L'accès à certains fichiers peut être interdit par l'utilisation de mots de contrôle. Pour associer un fichier catalogué sur CRAY à un travail, on utilise l'instruction **ACCESS** ; cependant, le fichier est toujours catalogué et peut être atteint par d'autres utilisateurs s'il n'a pas été déclaré à accès unique.

- les fichiers catalogués du système, soit relatifs à des travaux particuliers, soit contenant l'état des opérations du système. Un fichier d'entrée d'un travail est catalogué quand le travail est reçu par le Cray et est détruit quand le travail est terminé. Les fichiers de sortie locaux à un travail peuvent être retournés au calculateur frontal pendant l'exécution du travail, ou catalogués à la fin du travail puis détruits après avoir été rendus au calculateur frontal. Un exemple de fichier système est l'historique du système ("log").

I.3. - Fichiers résidents en mémoire

Certains fichiers peuvent être spécifiés "résidents en mémoire" par l'utilisateur. Ils n'occuperont en général pas de place sur mémoire de masse.

L'intérêt de déclarer un fichier résident en mémoire est de réduire ou d'annuler les entrées/sorties vers les disques. L'utilisateur doit déclarer une taille de tampon mémoire égale à celle du fichier plus 1 bloc, ceci par le paramètre **BS** de l'instruction **ASSIGN.**

Cela aura comme conséquence que :

- En lecture, le fichier sera chargé dans le tampon mémoire dès le premier ordre READ

- En écriture, le fichier ne sera écrit sur mémoire de masse que lorsque le tampon mémoire sera plein.

L'utilisateur déclarera un fichier résident en mémoire par le paramètre **MR** de l'instruction **ASSIGN** ou par l'appel à la fonction système **F$DNT** avec **DDMR=1.**

En outre, la taille du tampon doit être telle qu'elle contienne le fichier en totalité plus un bloc.

Si une écriture sur le fichier est tentée quand le tampon est plein, le fichier cesse d'être résident ; le tampon est vidé sur mémoire de masse et tous les pointeurs de résidence sont détruits.

I.4. - Règles d'écriture des noms de fichiers

L'utilisateur donne un nom symbolique à chaque fichier ; ce nom est formé de 1 à 7 caractères alphanumériques pour les fichiers locaux, de 1 à 15 caractères pour les fichiers catalogués. Le premier caractère doit être **A** à **Z**, **$**, **@** ou **%**.

Certains langages (autres que Fortran et le langage d'assemblage) peuvent être plus restrictifs quant à l'écriture du nom des fichiers.

Les fichiers définis par le système d'exploitation COS ont des noms de la forme **$dn.** Le $ ne doit donc pas être utilisé comme premier caractère pour les fichiers des utilisateurs.

I.5. - Formats de fichier

Deux formats de fichier sont admis par le Cray : bloqué et non bloqué.

1.5.1. - Le format bloqué

Le format bloqué est obligatoire pour les fichiers externes tels que les fichiers des entrées et des sorties des travaux. Le format bloqué ajoute des mots de contrôle aux données de façon à :

- prendre en compte des enregistrements variables,
- délimiter des niveaux de données à l'intérieur d'un fichier.

Un fichier peut être composé d'un ou plusieurs sous-fichiers qui sont eux-même composés d'un ou plusieurs enregistrements (Fig. 1).

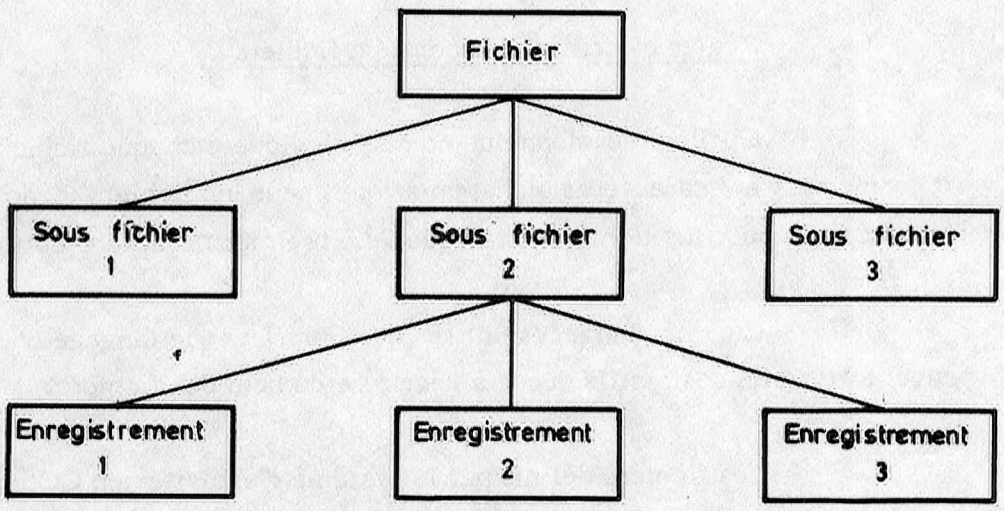


FIGURE 1

Les données dans un fichier bloqué peuvent être du type caractère ou binaire. Chaque bloc est composé de 512 mots.

Deux types de mots de contrôle sont utilisés dans un fichier bloqué :

- les mots de contrôle de blocs (Block Control Word ou **BCW**).
- les mots de contrôle d'enregistrements (Record Control Word ou **RCW**)

Pour les fichiers bloqués de type caractère, les zones contenant 3 à 96 blancs peuvent être comprimées en zones de 2 caractères blancs.

Remarque : Le constructeur utilise le nom **"DATASET"** pour fichier et **"FILE"** pour sous-fichier.

I.5.2. - Le format non bloqué

Un fichier d'entrée-sortie peut ne pas être bloqué. Tout fichier qui n'est pas bloqué selon le format Cray est dit non bloqué. Les fichiers non bloqués ne contiennent pas de mots de contrôle de blocs **(BCW)** ni de mots de contrôle d'enregistrement **(RCW)** de type Cray. Pour un fichier non bloqué, le système n'alloue pas de tampon dans la zone des tampons d'entrées-sorties d'un travail ; l'utilisateur doit spécifier une zone pour le transfert des données. Lors d'une lecture ou d'une écriture sur un fichier non bloqué, les données transitent directement depuis la zone de données utilisateur sans passer par le tampon d'entrées-sorties. Le nombre de mots de données à transférer doit être multiple de 512.

I.6. - Codes disposition des fichiers

Chaque fichier est muni d'un code disposition qui indique au système l'utilisation du fichier quand le travail est terminé. Ce code disposition est un paramètre des instructions de contrôle **DISPOSE** et **ASSIGN.** Il se code avec deux lettres. Par défaut, si le fichier n'est pas de type **$OUT** (voir II.2.2), le code disposition est **SC** (destruction en fin de travail). Par défaut, un fichier **$OUT** a pour code disposition **PR** (impression).

Un fichier **$OUT** n'exige pas l'instruction **DISPOSE** ; il est automatiquement, retrouné vers l'imprimante désignée du calculateur frontal ayant soumis le travail.

II. CONSTITUTION ET TRAITEMENT D'UN TRAVAIL

Un travail forme une entité soumise au traitement du calculateur. Il est constitué d'un ou plusieurs fichiers de cartes ou d'images de cartes. Chaque travail est soumis à plusieurs opérations entre la prise en compte du travail et sa fin de traitement.

II.1. - Structure du paquet de cartes d'un travail

La figure 2 ci-dessous explique la constitution d'un travail.

Le premier (ou seul) fichier d'un travail doit contenir les cartes des instructions du langage de commande qui spécifient le mode d'exécution du travail.

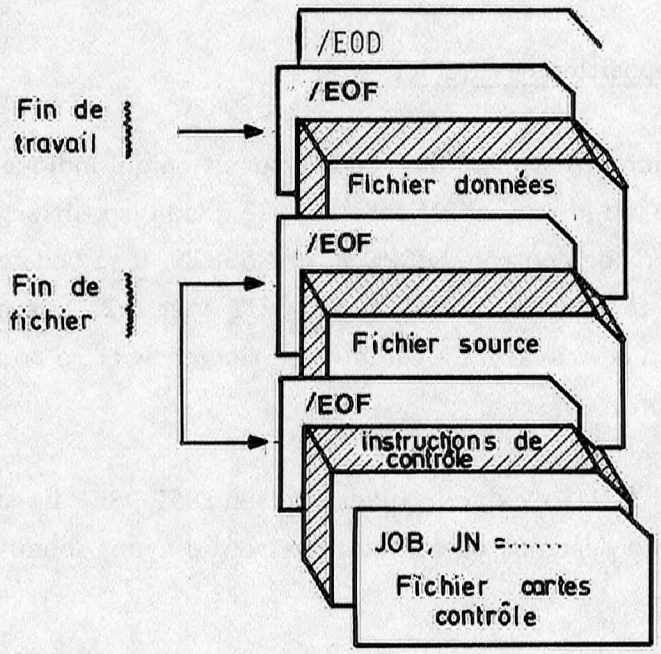


FIGURE 2

Chaque paquet de cartes d'un travail commence par une carte travail **(JOB)** qui permet d'identifier le travail ; les autres ordres du langage de commande viennent ensuite ; la fin du fichier des instructions est désignée par une carte fin de fichier (ou une fin de travail, s'il n'y a pas de fichier source ni de fichier données).

Les instructions du langage de commande donnent les directives d'exécution du fichier source et du fichier de données.

La carte finale du paquet est une fin de travail.

Les cartes fin de fichier et fin de travail ont un format défini par le calculateur frontal (**/EOF** et **/EOD** dans le cas des IBM du Centre de Calcul de Clamart).

II.2. - Déroulement d'un travail

Depuis la lecture sur le calculateur frontal jusqu'à son achèvement, le travail est soumis aux opérations suivantes :

- prise en compte ;
- initialisation ;
- exécution ;
- achèvement.

II.2.1. - Prise en compte

Un travail est soumis au système d'exploitation soit directement par un calculateur frontal, soit par un des terminaux du calculateur frontal.

Ce travail est installé sur mémoire de masse du Cray jusqu'au début de son traitement. Une entrée est créée dans les tables du système d'exploitation pour ce travail.

II.2.2. - Initialisation

Le système d'exploitation examine les paramètres de la carte travail et détermine les ressources nécessaires au travail ; dès que ces ressources sont libres, le travail est programmé pour exécution. L'initialisation du travail comprend :

- la réservation de la zone de mémoire nécessaire à l'exécution, et de la zone nécessaire à l'implantation des tables du travail (**JTA** : Job Table Area) ;

- le positionnement du fichier de travail pour accomplir la première opération du travail ;

- la mise du travail dans la file d'attente d'entrée de l'unité centrale.

Quand le système d'exploitation programme l'exécution du travail, il crée quatre fichiers :

- **$CS** est une copie du fichier des instructions de contrôle utilisée par le système uniquement ; l'utilisateur ne peut pas nommer **$CS.** Ce fichier est détruit en fin de travail (code disposition **SC).**

- **$IN** est le fichier d'entrée du travail. Il est identifié sur le calculateur frontal par un nom ; le travail peut atteindre ce fichier, par son nom **$IN** ou par le numéro logique 5 en Fortran, pour le lire.

- **$OUT** est le fichier de sortie du travail. Le travail peut accéder à ce fichier par son nom ou par le numéro logique 6 en Fortran. Ce fichier sera imprimé en fin de travail.

- **$LOG** est le fichier historique du travail.

II.2.3. - Exécution

Le système commence l'exécution du travail par l'examen des (instructions), qu'il contient et qui spécifient les opérations à effectuer. Ces instructions sont lues, interprétées et exécutées.

Le système d'exploitation COS du Cray 1 fonctionne en multi-programmation : l'unité centrale est partagée entre plusieurs travaux selon des algorithmes choisis par l'exploitation.

II.2.4. - Achèvement

Les sorties d'un travail résident sur mémoire de masse. A la fin du travail, le système copie le fichier **$LOG** sur **$OUT** ; le fichier **$OUT** est catalogué ; les fichiers **$IN**, **$CS** et **$LOG** sont détruits ; **$OUT** est renommé (paramètre **JN** de la carte travail) ; il est dirigé vers la file d'attente de sortie des travaux du calculateur frontal ayant soumis le travail. Quand le calculateur frontal a reçu la totalité du fichier **$OUT**, les entrées des tables du système pour ce fichier sont détruites.

Le calculateur frontal traite le fichier **$OUT** comme prévu dans le code disposition ; d'une manière standard **$OUT** est imprimé sur l'imprimante voulue du calculateur frontal.

S'il n'existe pas de fichier **$OUT**, seul le fichier **$LOG** est produit par le travail.

II.3. - Reprise d'un travail

Un travail peut être repris pour les raisons suivantes :

- commande opérateur ;
- erreur mémoire incorrigible ;
- erreur incorrigible lors du transfert d'un travail depuis la mémoire de masse vers un disque de manœuvre, qui rend exécutable un travail de plus forte priorité ;
- redémarrage du système.

Cependant quelques fonctions rendent un travail impossible à reprendre ; ce sont :

- l'écriture sur un fichier catalogué ;
- la mise au catalogue d'un fichier, la destruction, la modification des paramètres d'un fichier catalogué ;
- l'acquisition d'un fichier à partir d'un calculateur frontal.

L'utilisateur peut annuler l'effet de ces fonctions et forcer la reprise du travail à condition d'être sûr que les résultats du travail seront malgré tout cohérents.

II.4. - Historique du travail et informations comptables

Pour chaque travail, le système fournit un historique qui est listé en fin d'impression des sorties au terminal origine du travail.

Les instructions de contrôle sont reproduites séquentiellement et suivies des messages propres au déroulement du travail. Le temps d'exécution du travail, les messages du système, et des renseignements comptables sont également fournis dans l'historique.

Le tableau I ci-dessous donne un modèle d'historique de travail.

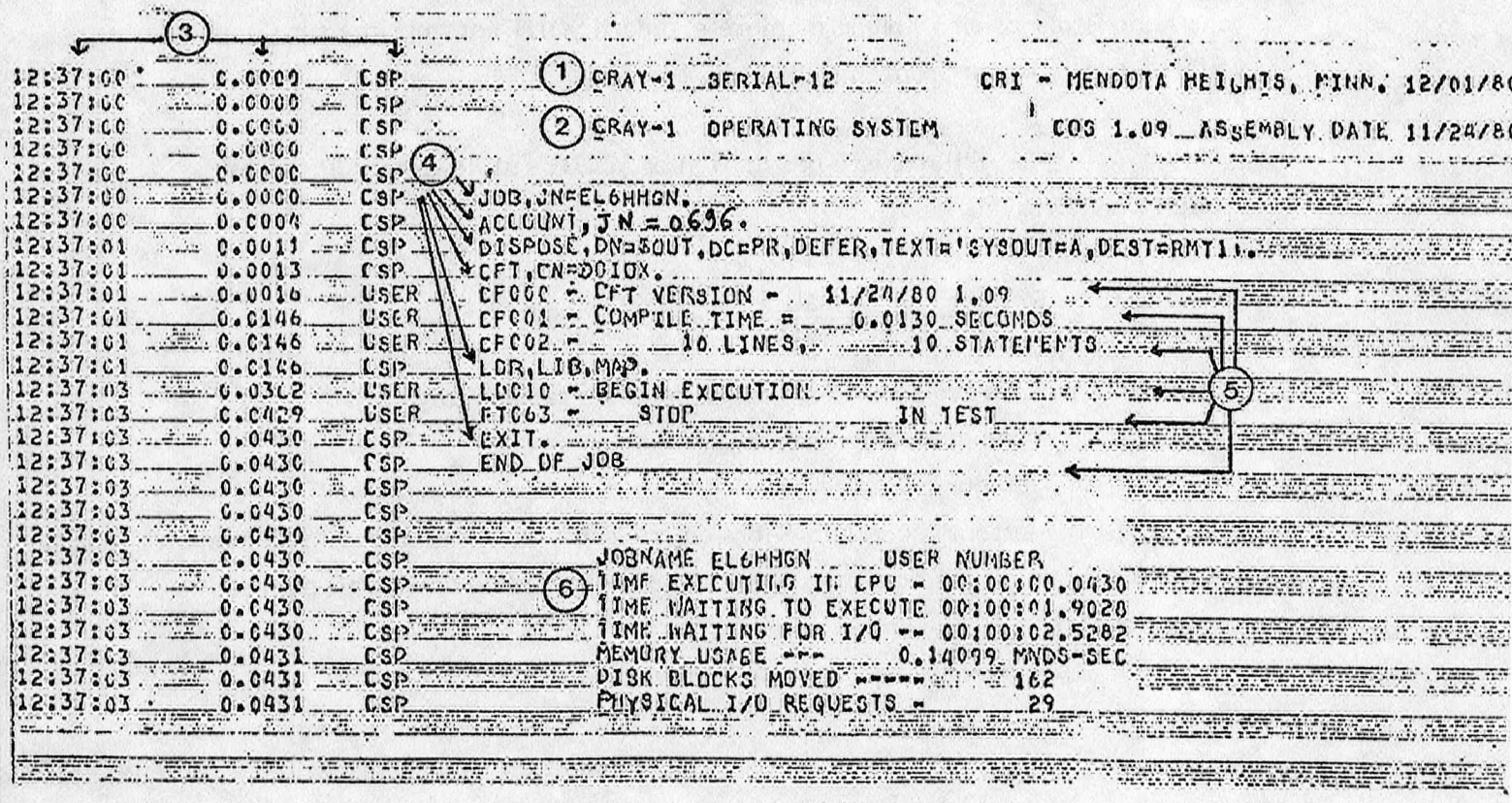


Tableau I

1 Première ligne de restitution : Lieu et date de traitement du travail.

2 Seconde ligne de restitution : Identification de la version du système et date de la dernière révision.

3 La colonne de gauche donne l'heure à laquelle chaque étape s'est déroulée.

La colonne du milieu donne le cumul en décimal du temps UC au cours des étapes.
La colonne de droite identifie l'origine du message :

CSP traitement des instructions du langage de commande
PDM gestion des fichiers catalogués

EXP traitement des échanges entre l'utilisateur et le système

ABORT message de fin anormale

USER messsage lié au traitement demandé par le langage de commande.

4 Liste des instructions du langage de commande dans l'ordre où elles sont traitées ; en cas d'erreur, la dernière instruction traitée est la dernière imprimée.

5 Messages de l'historique : les messages relatifs à une instruction sont listés derrière cette instruction.

6 Informations relatives au déroulement du travail :
- nom et numéro d'identification du travail ;
- temps UC utilisé par le travail en heures, minutes et 1/10.000^{ième} de secondes (décimal) ;
- temps en attente d'exécution, en heures, minutes et 1/10.000^{ième} de secondes (décimal) ;
- temps d'attente pour les entrées et les sorties, en heures, minutes et 1/10.000^{ième} de secondes (décimal) ;
- nombre de secteurs de disques manipulés ;
- nombre d'entrées et de sorties physiques demandées par le travail.

III. REGLES D'ECRITURE DES INSTRUCTIONS DU LANGAGE DE COMMANDE

La syntaxe générale d'une carte du langage de commande est la suivante :

VERBE de définition de la commande	SEPARATEUR	PARAMETRE	TERMINAISON	COMMENTAIRES

Il peut se faire que la commande n'ait pas de paramètre ; dans ce cas elle est réduite au mot-clé, à la terminaison et à d'éventuels commentaires. Le nombre de paramètres dépend du mot-clé. Les commentaires sont optionnels.

III. 1 - Le verbe

Le verbe est la première zone non blanche de la carte. Le verbe doit s'écrire sur une seule carte. Il peut être un nom de commande (gestion de fichiers) ou un nom de programme (appel au compilateur, au chargeur, à un utilitaire). Le verbe est composé de caractères alphanumériques dont le premier est une lettre.

III.2 - Les séparateurs

Un séparateur est constitué d'un caractère. Le caractère virgule (,) sert de séparateur entre le verbe et le premier paramètre, ou entre paramètres.

Le caractère égale (=) est séparateur entre un paramètre à mot-clé et la valeur ou les valeurs associées.

Le caractère deux points (:) est séparateur de concaténation entre plusieurs valeurs d'un paramètre.

Les caractères parenthèse ouverte et fermée sont respectivement un séparateur initial optionnel (entre le verbe et le premier paramètre), et une terminaison optionnelle.

III.3 - Les paramètres

Les paramètres sont positionnels ou à mot-clé. Ils sont définis en fonction des propriétés de chaque commande.

Le format d'un paramètre positionnel est :

valeur

ou **Valeur$_1$: valeur$_2$: ... : valeur$_n$**

Les paramètres à mot-clé sont identifiés par leur forme ; ils sont formés d'une chaîne de 1 à 8 caractères alphanumériques ; ils peuvent être placés dans n'importe quel ordre et peuvent être omis ; un paramètre à mot-clé doit être écrit sur une seule carte.

Le format d'un paramètre à mot clé est le suivant :

paramètre

ou **paramètre = valeur**

ou **paramètre = valeur$_1$: valeur$_2$: ... : valeur$_n$**

III.4 - La terminaison

Le délimiteur de fin de commande est le caractère point (.) ou d'une manière optionnelle la parenthèse fermée.

III.5 - Les commentaires

Un commentaire sert d'annotation pour une carte du langage de commande.

Le commentaire est constitué d'une chaîne de caractères ASCII ; il se place derrière la terminaison d'une instruction. L'interprète des cartes contrôle ignore les commentaires ; ils sont listés dans le fichier de sortie.

III.6 - Conventions de codage

III.6.1 - Blancs

Les blancs sont ignorés sauf s'ils sont contenus dans une chaîne.

Des blancs ne peuvent pas précéder les mots-clés **JOB, DUMPJOB** et **EXIT.**

III.6.2 - Cartes suites

Une instruction peut s'écrire sur plusieurs cartes ; dans ce cas, toute carte suite est précédée par une carte ayant le caractère ^ (caret) en dernière position ou avant le commentaire éventuel.

Exemple :

carte **mot-clé, param$_1$ = valeur, ..., param$_n$ ^**

carte
suite
} **= valeur.**

IV. - MODALITES D'EXECUTION D'UN TRAVAIL

IV.1 - La carte de travail : **JOB**

L'instruction **JOB** doit être la première instruction du fichier instruction de contrôle. Elle définit le travail auprès du système opératoire. L'instruction **JOB** doit s'écrire sur une seule carte et ne pas comporter de blancs.

Format :

JOB, JN = jn, M = fl, T = tl, US = us, OLM = lm, CL = jcn, TIO = tio.

Les paramètres sont à mot clé ; les seuls paramètres obligatoires sont **JN** et **US.**

JN = jn Nom du travail formé de 7 caractères ayant la même signification que les 7 premiers d'une carte travail IBM. **JN** est obligatoire.

M = fl Taille de mémoire ; **fl** spécifie un nombre octal de blocs de 1000_8 mots à fournir pour le travail (1000_8 = 512 décimal).
L'adresse limite est fonction de l'adresse de base et de la taille mémoire spécifiée :

LA = BA + fl * 512

Si le paramètre **M** est omis, la taille de mémoire allouée au programme est de 145000_8 mots (soit environ 400 K mots).

T = tl Temps d'exécution du travail exprimé en secondes et en décimal.
Si **T** est omis, le temps est fixé à 10 secondes.

V. COMPILATION FORTRAN : CFT

L'instruction **CFT** permet l'appel du compilateur Fortran. La syntaxe est la suivante :

CFT (I = idn, L= ldn, B = bdn, C = cdn, E = n, ON = listeon, OFF = listeoff, TRUNC = nn)

Tous les paramètres sont optionnels. Si l'on veut n'en mentionner aucun, il suffit de coder **CFT.** seul.

Signification des paramètres :

I = idn : nom du fichier source à compiler ; le nom par défaut est **$IN.**

L = ldn : nom du fichier qui recevra la liste de compilation. Le nom par défaut est **$OUT.L = 0** inhibe l'édition de la liste saut en cas d'erreur fatale où les messages sont fournis sur **$OUT.**

B = bdn : désigne le fichier binaire dans lequel est stocké le module objet généré. Le nom par défaut est **$BLD.** Aucune fin de fichier n'est écrite.

C = cdn : nom du fichier contenant la traduction en langage d'assemblage du programme source, généré par le compilateur. La traduction n'a pas lieu si l'option C n'est pas spécifiée.

E = n : supprime les messages du compilateur de niveau inférieur ou égal à **n.** La valeur par défaut est 3. Les messages d'erreurs fatales sont toujours listés. Les niveaux des messages sont décrits dans le tableau II.

ON = listeon : Liste sur 15 caractères des options de compilation activées ; ces options sont précisées dans le tableau III.

OFF = listeoff : Liste sur 15 caractères des options de compilation désactivées.

TRUNC = nn : Indique le nombre de bits tronqués lors de l'évaluation des résultats en virgule flottante. **nn** est compris entre 0 et 47 inclus. La valeur par défaut est 0.
Les résultats en réel double précision, les résultats de fonction, les constantes ne subissent pas la troncature.

Tableau II

Niveau du message d'erreur	Sévérité de l'erreur	Description
1	Commentaire	commentaire sur les parties inefficaces du programme.
2	Note	signale les instructions n'appartenant pas à la norme de Fortran.
3	Avis	erreur possible (exemple : instruction non accessible).
4	Avertissement	erreur probable (exemple : indice manquant dans la référence à un tableau).
5	Erreur	erreur fatale entraînant l'arrêt du travail.

Tableau III - OPTIONS DU COMPILATEUR

Option	DESCRIPTION	Valeur par défaut
A	Le travail se termine après la compilation si l'un des modules du programme contient une erreur fatale.	**OFF**
B	Liste le numéro de séquence des blocs d'instructions du langage généré. L'option **G** implique l'option **B**.	**OFF**
C	Liste des noms de **COMMON** et de leur longueur après chaque module de programme.	**ON**
D	Liste des tables des boucles **DO**	**OFF**
E	Prise en compte des directives du compilateur introduites dans le langage source.	**ON**
F	Prise en compte de l'option **FLOWTRACE** (voir la directive **FLOW/NOFLOW**).	**OFF**
G	Liste du code généré de chaque module de programme (voir la directive **CODE/NOCODE**).	**OFF**
H	Liste de la première instruction de chaque module de programme. Les autres options de liste sont ignorées ou désactivées.	**OFF**
I	Introduction des étiquettes d'instructions générées par le compilateur, dans la table des symboles.	**OFF**
J	Exécution de chaque boucle **DO** au moins une fois (compatibilité avec le Fortran IVH d'IBM).	**OFF**

Option	DESCRIPTION	Valeur par défaut
L	Activation des directives de contrôle de la liste de la compilation.	**ON**
M	Activation du scheduler	**ON**
N	Introduction dans la table des symboles, de symboles définis mais non référencés.	**OFF**
O	Contrôle et impression d'un message d'erreur en cas de débordement d'un indice de tableau. La vectorisation est inhibée.	**OFF**
P	Prise en compte de la double précision. ' Si **OFF** = **p** : - Les déclarations **DOUBLE PRECISION** sont traitées en réel simple précision. - Les fonctions **DOUBLE PRECISION** sont traitées dans leur version réelle simple précision. - Les constantes à convertir en double précision sont converties et tronquées en réel simple précision. - Dans les formats, les spécifications d'édition **D** sont changées en **E**	**ON**
Q	La compilation est abandonnée après 100 messages d'erreurs fatales.	**ON**
R	Arrondi des résultats des multiplications.	**ON**
S	Liste du source Fortran.	**ON**
T	Liste des tables de symboles après chaque module de programme.	**ON**
V	Vectorisation des boucles **DO** internes.	**ON**
W	Remplacement des opérations flottantes câblées par l'appel de fonctions logicielles externes fournies par l'utilisateur (voir tableau IV ci-après).	**OFF**

Option	DESCRIPTION	Valeur par défaut
X	Liste de la table des symboles avec références croisées après chaque module de programme **X** annule **T.**	**OFF**
Y	Arrêt du travail après détection d'une erreur du compilateur. Option de mise au point de programme.	**OFF**
Z	Ecriture sur le fichier **$BLD** de la table des symboles utilisée par l'utilitaire DEBUG.	**OFF**

Tableau IV

Opérations câblées simple précision	Fonctions logicielles équivalentes à fournir par l'utilisateur
s + s	RASS%
s - s	RSSS%
s * s	RMSS%
1 / s	RDSS%
s + v	RASV%
s - v	RSSV%
s * v	RMSV%
1 / v	RDSV%
v + v	RAVV%
v - v	RSVV%
v * v	RMVV%
v - s	RSVS%

v opérande vectoriel

s opérande scalaire

NOTA : Les opérations suivantes sont modifiées comme indiqué ci-dessous avant l'appel aux fonctions :

v + s → s + v

v * s → s * v

v / s → (1/s) * v

- s → 0 - s

- v → 0 - v

s / v → $(\frac{1}{v})$ *s ou s* $(\frac{1}{v})$

$v_1/v_2 \rightarrow 1/v_2 * v_1$

Le passage des arguments, lors de l'appel des sous-programmes ci-dessus est fait par valeur ; normalement, le premier opérande est dans **S1** ou **V1**, le second est dans **S2** ou **V2**, et les résultats sont retournés dans **S1** ou **V1** ; pour la division, les résultats de **S2** et **V2** sont retournés dans **S1** et **V1**.

VI CHARGEUR ; BIBLIOTHEQUES

VI.1. Chargement d'un programme : LDR

L'exécution du chargeur est demandée par l'instruction **LDR :**

LDR, DN = dn, LIB = ldn, NOLIB = ldn, LLD, AB = adn, MAP = op,
T = tra, NX, C, OVL = dir, NA, USA, L = ldn, SET = val,
E = n, I = sdir.

Les paramètres sont du type à mot-clé.

DN = dn Nom du fichier contenant les modules à soumettre au chargeur ; par défaut, le nom de ce fichier est **$BLD.** La reliure et le chargement s'exécutent jusqu'à la rencontre d'une fin de fichier.
Les modules sont chargés selon leur nom défini par une instruction **IDENT** en langage d'assemblage ou par une instruction **PROGRAM, SUBROUTINE, BLOCK DATA** ou **FUNCTION** en Fortran.
Les modules dont le nom est déjà référencé ne sont pas pris en compte et un message informatif est produit. Jusqu'à 8 sous-fichiers d'un même fichier peuvent être chargés à condition de citer le nom du fichier autant de fois qu'il existe de sous-fichiers à charger avec le caractère : comme séparateur.
Les fichiers spécifiés par le paramètre **DN** sont fermés à la fin du chargement ; la fermeture d'un fichier équivaut à un rebobinage et la libération des tables d'entrées-sorties et des tampons.
Les modules à charger peuvent être absolus ou relatifs ; on ne peut pas mélanger les deux types.

Par exemple :

DN = CHARG1 : CHARG2 : SBLD

provoque le chargement de tous les modules du premier sous-fichier de **CHARG1** puis de **CHARG2** puis de **$BLD.**

Normalement, le fichier est rebobiné avant le chargement ; cependant des occurrences consécutives d'un nom de fichier interdisent des opérations de rebobinage successives.
Il en résulte que l'ordre

DN = CHARG3 : CHARG3

provoque le chargement de tous les modules des deux premiers sous-fichiers de **CHARG3.**

Le paramètre **DN** prend une signification particulière quand **OVL** est codé : un seul nom fichier **(dn)** peut être spécifié.

LIB = idn **idn** nomme le fichier où sont recherchés les modules qui correspondent à des références externes non satisfaites. Un maximum de huit fichiers peuvent être nommé ; les noms seront séparés par le caractère **:**. Les fichiers cités ne nécessitent pas de carte **ACCESS** pour être rattachés sauf si leur nom comprend plus de 7 caractères (cas des bibliothèques standard SYSZ%GENERALE, SYSZ%BENSON etc)..
Si **NOLIB** n'est pas codé, les bibliothèques chargées par défaut n'ont pas à être nommées.
Les fichiers spécifiés par le paramètre **LIB** sont fermés à la fin du chargement. La fermeture du fichier a pour effet de le rebobiner, de libérer les tables d'entrées-sorties et les tampons.
N.B. Attention à la nécessité de coder **ACCESS** pour les bibliothèques dont les noms ont plus de 7 caractères.

NOLIB = ldn Ce paramètre permet de nommer les bibliothèques à exclure du chargement. Si **NOLIB** est codé sans paramètre, toutes les bibliothèques normalement chargées par défaut sont exclues du chargement.
Si **NOLIB** n'est pas codé, toutes les bibliothèques chargées par défaut le sont effectivement ainsi que celles citées par **LIB.**

LLD Ce paramètre permet de libérer les bibliothèques utilisées après le chargement ; les fichiers bibliothèques restent ouverts. Si **LLD** n'est pas codé, le chargeur ferme les bibliothèques après le chargement. Les bibliothèques rattachées au travail automatiquement ne sont pas libérées après le chargement **(RELEASE).**

AB = adn Génération du module objet binaire et copie sur le fichier de nom **adn** après la fin du chargement. **AB** n'implique pas **NX** (pas d'exécution). Si **NX** n'est pas spécifié, le programme s'exécute après que le binaire soit recopié. Si **AB** est codé sans adn, le module objet binaire est écrit sur le fichier **$ABD**, fichier réservé par défaut du module objet binaire.
Le fichier **adn** n'est pas rebobiné ni avant ni après l'écriture du module.
Si **AB** n'est pas codé, il n'y a pas génération du binaire.
Si **OVL** est codé, la directive **OVLDN** remplace **AB** ; toute valeur spécifiée pour **AB** est alors ignorée.

MAP = op Contrôle de l'impression de la table de chargement.

MAP	peut prendre les valeurs suivantes :
ON	Liste des blocs et des points d'entrée avec les références croisées pour chaque entrée.
FULL	Equivalent à ON
OFF	Pas de table fournie ; par défaut **MAPP = OFF.**
PART	Liste des blocs ; équivalent à **MAP** codé sans valeur.

T = tra Nom du module principal. Le paramètre **T** permet de spécifier un nom de module principal où le chargeur donnera le contrôle à la fin du chargement ; ce paramètre spécifie également le module principal qui sera inclus dans l'absolu généré.
Si le paramètre **T** n'est pas spécifié le chargeur prendra le premier module principal rencontré sur le fichier.
Un module principal est défini par la seconde instruction **START** en assembleur. Il est défini en Fortran, par l'instruction **PROGRAM** ou l'absence d'instruction **FUNCTION, SUBROUTINE** ou **BLOCK DATA.**

NOTE : Un fichier relatif peut contenir plusieur modules principaux mais on ne peut en exécuter qu'un seul à la fois d'où l'intérêt du paramètre T pour sélectionner un module principal dans un fichier donné.

NX Si **NX** est codé, l'exécution du programme chargé est inhibée.

C Chargement avec compression : le chargement de chaque module se fait d'une manière contiguë. Si **C** n'est pas codé, le chargement des modules se fait à la prochaine adresse multiple de 20_8 mots.

OVL = dir Chargement en overlay ; les directives de l'overlay sont fournies sur le fichier de nom dir.
Si **OVL** est codé, sans valeur de dir, les directives de l'overlay sont données sur **$IN.**
Par défaut, il n'y a pas d'overlay. **OVL** implique **NX** (pas d'exécution).

NA Si **NA** est codé, pas de fin anormale possible. Si ce paramètre n'est pas codé, une erreur du chargement provoque la fin du travail.

USA Fin de travail en cas de référence externe non satisfaite. Une liste des références externes non satisfaites est produite, si demandée.

L = ldn Nom du fichier sur lequel est écrite la table de chargement. Par défaut, ce fichier est **$OUT.**

SET = VAL Initialisation de la mémoire. Les variables, les COMMON à label ou blancs, les zones mémoires retenues par la déclaration DIMENSION sont initialisées pendant le chargement.

SET = ZERO Ces objets sont affectés de zéros binaires.

SET = ONES Ces objets sont affectés de - 1 (tous les bits du mot à 1).

SET = INDEF Ces objets sont affectés à une valeur flottante hors limite afin de provoquer une erreur fatale lors de leur utilisation.

E = n Contrôle de la liste des messages d'erreur ; il existe 5 niveaux de messages d'erreurs (tableau V) : n indique le niveau de messages le plus haut à ne pas lister. Par défaut **E = 2,** (les erreurs de niveaux 3, 4, 5 seront listées).

Tableau V

Niveau du message d'erreur	Sévérité de l'erreur	Description
1	Commentaire	L'erreur n'empêche pas l'exécution du travail.
2	Note	L'erreur empêche probablement l'exécution du travail.
3	Avis	Le travail se termine après le chargement à moins que NA soit codé.
4	Avertissement	Le travail se termine après le chargement si NX est codé. L'exécution du travail est impossible.
5	Erreur fatale	Le travail se termine immédiatement.

I = sdir Chargement sélectif. Des modules d'un autre fichier peuvent être chargés selon certaines directives données dans le fichier de nom **sdir.** Si I est codé sans valeur, les directives sont données dans **$IN** . Les directives de chargement sélectif sont décrites dans le tableau VI.

Tableau VI

Directives du chargement **sélectif : INCLUDE, EXCLUDE**

INCLUDE, SDN = sdn, FN = fn, MOD = md_1 : md_2 : ... : md_{50}.
EXCLUDE, SDN = sdn, FN = fn, MOD = md_1 : md_2 : ... : md_{50}.
Les paramètres à mot-clé sont : **SDN = sdn** Nom du fichier contenant les modules à charger sélectivement. Si **SDN** est codé sans valeur, le premier fichier spécifié par **DN** (dans carte **LDR**) est choisi, par défaut. Si **SDN** est omis, un message d'erreur est écrit, la directive n'est pas prise en compte, le chargement est effectué. **SDN** et **FN** doivent nommer le même fichier. **FN = fn** Numéro compris entre 0 et 7 du positionnement du sous-fichier dans le fichier spécifié par **SDN** ou **DN** de la carte **LDR.** Si **FN** est codé sans valeur, par défaut **fn = 0.** Si **FN** n'est pas codé, les modules à charger sont recherchés dans tout le fichier nommé par **SDN** ; un message est écrit. **SDN** et **FN** doivent nommer le même fichier. **MOD = md_1** Nom d'un module ou d'un point d'entrée d'un module à inclure ou exclure au moment du chargement ; on peut spécifier jusquà 50 modules ; (les noms des modules sont séparés par le signe :). Si **MOD** n'est pas codé, un message d'erreur est écrit et la directive est ignorée.

Exemple de directive :

Soit l'appel au chargeur :

LDR, DN = D1 : D1 : D2, ...

La directive suivante placée dans **$IN :**

INCLUDE, SDN = D1, FN = 1, MOD = ...

permet de charger un module du second sous-fichier du fichier **D1.**

Les messages de chargement sélectif ne peuvent pas être supprimés.

VI.2. - Création et gestion de bibliothèques de programmes : **BUILD**

Généralités

BUILD est un utilitaire permettant de créer et de mettre à jour des bibliothèques de programmes.

Une bibliothèque est un fichier composé d'un ou plusieurs modules objets et d'un répertoire contenant une entrée pour chaque module objet. Chaque entrée du répertoire comprend le nom du module objet ainsi que son adresse relative au début du fichier. Le nom d'un module objet est celui précisé par les cartes **PROGRAM, SUBROUTINE, FUNCTION** et **BLOCKDATA** en Fortran. (IDENT en langage d'assembleur).

L'accès à une ou plusieurs bibliothèques se fait par l'instruction **LDR.** (Voir paramètre **LIB** page ...). Le chargeur charge en mémoire les références externes qu'il recherche dans la ou les bibliothèques précisées.

Pour créer une nouvelle bibliothèque, **BUILD** peut utiliser une ou plusieurs directives que l'utilisateur fournira soit dans **$IN** soit dans un fichier annexe.

Format de l'instruction **BUILD**

BUILD,I=ddn,L=ldn,OBL=odn,B=bdn,NBL=ndn,SØRT,NØDIR.

I = ddn — **ddn** est le nom local du fichier contenant les directives. Si ce paramètre apparaît seul ou est omis, les directives sont lues dans le fichier **$IN.**

I = 0 → aucune directive n'est fournie par l'utilisateur

L = ldn — **ldn** est le nom local du fichier qui recevra la liste de sortie. Si ce paramètre apparaît seul ou est omis, la liste apparaît sur **$OUT.**

L=0 → aucune liste n'est écrite

OBL = odn — **odn** est le nom local de la bibliothèque précédemment créée et que l'utilisateur veut mettre à jour.
Si ce paramètre apparaît seul ou est omis, le nom du fichier pris par défaut est **$OBL.**
OBL = 0 → Il n'y a pas d'ancienne bibliothèque car il s'agit de la création initiale.

B = bdn — **bdn** est le nom local du fichier contenant les modules objets à ajouter ou à remplacer. Ce fichier résulte d'une compilation ou d'un assemblage. Si le paramètre **B** apparaît seul ou est omis, le nom du fichier pris par défaut est **$BLD.**

B = 0 → le programme fait une édition de l'ancienne bibliothèque sans en faire de mise à jour.

NBL = ndn — **ndn** est le nom local de la nouvelle bibliothèque. Si ce paramètre apparaît seul ou est omis, le nom du fichier pris par défaut est **$NBL**

NBL = 0 → aucune bibliothèque n'est créée. Ce cas est limité à l'usage de certaines directives de **BUILD.**

SORT — Ce paramètre doit être précisé lorsque l'utilisateur veut une liste des modules selon un ordre alphabétique. Par défaut, cette liste apparaît dans l'ordre où les modules sont stockés dans le fichier.

NODIR — Cette directive indique que le répertoire ne doit pas être ajouté à la fin de la nouvelle bibliothèque créée. Celle-ci aura donc la même structure qu'un fichier séquentiel tel que **$BLD.** Par défaut, le répertoire est ajouté au fichier créé.

Les directives de **BUILD**

Ces directives sont contenues dans le fichier précisé par le paramètre I de l'instruction **BUILD.**

Une directive consiste en un mot-clé suivi éventuellement par un blanc et par une liste de noms de modules ou de noms de fichiers, séparés les uns des autres par des virgules.

La directive **FROM**

Cette directive peut servir à préciser le nom du fichier d'entrée utilisable par les directives **COPY, OMIT** et **LIST.**

Elle peut également servir à préciser les noms des fichiers d'entrée (séquentiels ou sous forme de bibliothèques) qui seront regroupés dans la nouvelle bibliothèque créée. Le dernier nom de cette liste sert de fichier d'entrée aux directives (si elles existent) **COPY, OMIT** et **LIST.**

Cette directive s'écrit :

FROM dn1, dn2, ... dn_n

Exemple de concaténation de plusieurs bibliothèques

```
ACCESS, DN = LIB1, PDN = MONFICH1.
ACCESS, DN = LIB2, PDN = MONFICH2.
ACCESS, DN = LIB3, PDN = MALIB.
BUILD, I, OBL = 0, B = 0.
SAVE, DN = $NBL, PDN = MALIB.
/EOF
FROM    LIB1, LIB2, LIB3
```

Remarque

Si deux modules portent le même nom dans 2 fichiers différents, le module pris en compte est celui appartenant au fichier qui apparaît le dernier dans la directive **FROM.**

La directive **OMIT**

Cette directive sert à supprimer un ou plusieurs modules de l'ancienne bibliothèque, ou d'un quelconque fichier d'entrée précisé par une précédente directive **FROM.** Cette directive n'est pas affectée par les directives **FROM** qui la suivent.

Cette directive s'écrit :

OMIT fn1, fn2, ..., fn_n

Chaque nom fn_i représente soit le nom d'un module simple, soit un ensemble de modules ayant une chaîne de caractères commune dans leur nom

F$-** → représente tous les modules dont le nom commence par **F$

AB* → représente tous les modules dont les noms de 5 caractères possèdent les caractères **AB** en 2ème et 3ème position.

Exemple de suppression d'un module d'une bibliothèque

```
ACCESS,DN = $OBL, PDN = MALIB.
BUILD, B = 0.
SAVE, DN = $NBL, PDN = MALIB.
```



```
/EOF
OMIT      SUB1
```

Le module **SUB1** n'apparaîtra plus dans la nouvelle bibliothèque.

La directive **COPY**

Cette directive permet de sélectionner certains modules de la bibliothèque d'entrée et de les recopier sur le fichier de sortie créé (bibliothèque ou fichier séquentiel créé avec option **NODIR**).

Cette directive s'écrit :

COPY $fn_1, fn_2, \ldots fn_n$

Chaque nom fn_i, comme pour la directive **OMIT**, représente soit le nom d'un module simple, soit un ensemble de modules ayant une chaine de caractère commune dans le nom. (voir directive **OMIT**).

La directive **LIST**

Cette directive liste les caractéristiques des différents modules contenus dans une bibliothèque.

Cette directive s'écrit :

LIST

VII. - INSTRUCTIONS DE GESTION DES FICHIERS

VII.1. - Création, accès et fin d'accès à un fichier

Les fonctions de création et d'accès aux fichiers sur le Cray sont réalisées par les deux instructions suivantes :

- **ASSIGN** permet de créer un fichier et de lui attribuer ses caractéristiques.

- **RELEASE** libère un fichier.

VII.1.1. Créer un fichier : **ASSIGN**

ASSIGN est utilisée pour créer un fichier et définir ses caractéristiques.

Si l'instruction **ASSIGN** est utilisée pour la création d'un fichier, elle doit apparaître avant toute référence à ce fichier ; sinon les caractéristiques du fichier seront définies à la première référence.

Format :

ASSIGN, DN = dn, S = size, A = adn, BS = blk, RDM, U.

Les paramètres sont à mot-clé. Le seul obligatoire est **DN.**

DN = dn Nom du fichier local. DN est obligatoire.

S = size Taille du fichier. Nombre en octal de secteurs (blocs de 1000_8 = 512 mots) à réserver pour le fichier ; si ce paramètre n'est pas fourni, l'espace sur disque est alloué dynamiquement au fichier selon les besoins. La taille maximale possible est de 60 millions de caractères.

A = adn — **adn** sert à établir la relation avec la description logique du fichier. En Fortran, adn sera **FTnn** ou **00 <nn<** 99.

BS = blk — Taille du buffer ; nombre octal de blocs de 512 = 1000_8 mots à réserver pour le buffer. Le nombre de blocs pris par défaut par l'installation est 4.
Si le paramètre U (format non bloqué) est spécifié, **BS** provoque une erreur.

RDM — Si **RDM** est codé, le fichier est à accès direct ; **si RDM** est absent, le fichier est séquentiel.

U — Si **U** est codé, le fichier est de format non bloqué.
Si **U** n'est pas codé, le fichier est de format bloqué.

VII.1.2. - Libérer un fichier : **RELEASE**

L'instruction **RELEASE** libère les fichiers qui sont nommés. Si le fichier n'est pas catalogué la place allouée au fichier est rendue au système. Si le code disposition est **IN**, le fichier est mis dans la file d'attente d'entrée du Cray. Pour les autres codes dispositions, le fichier est placé dans la file d'attente de sortie vers le calculateur frontal origine. Si le fichier est catalogué, il n'est plus rattaché au travail.

Format :

RELEASE, DN = dn_1 : dn_2 : ... dn_8.

DN = dn_1, nom du fichier à libérer ; on peut spécifier jusqu'à un maximum de huit fichiers.

VII.2. - Gestion des fichiers catalogués utilisateurs

Le Permanent Data Set Manager permet aux utilisateurs de gérer leur fichiers catalogués par l'intermédiaire des instructions de contrôle suivantes :

SAVE permet de cataloguer un fichier.

ACCESS permet l'attribution d'un fichier catalogué à un travail.

DELETE Supprime un fichier du catalogue ou le rend local.

VII.2.1. - Cataloguer un fichier : **SAVE**

permet de catagoguer un fichier local en lui donnant une entrée au catalogue ; un fichier catalogué est uniquement identifié par un nom, l'identification de l'utilisateur et le numéro de version.

SAVE joue deux rôles :

- création de la version initiale d'un fichier catalogué ;
- création d'une version additionnelle d'un fichier catalogué.

Le mot de contrôle de maintenance gère la création des versions additionnelles d'un fichier catalogué existant ; en conséquence, pour créer une nouvelle version d'un fichier catalogué existant, l'utilisateur doit mettre à jour le mot de contrôle de maintenance de la version précédente. Les mots de contrôle pour **READ** et **WRITE** spécifiés à la dernière version d'un fichier catalogué s'appliquent à toutes les versions suivantes.

Pour des impératifs d'exploitation, un fichier catalogué ne peut pas être conservé plus de 7 jours. Il est donc recommandé de conserver les fichiers permanents sur les frontaux IBM **(DISPOSE)** et de les transférer sur Cray au début de chaque travail **(ACQUIRE)**.

Format :

SAVE, DN = dn, PDN = pdn, ID = uid, ED = ed, R = rd, W = wt, UQ, NA.

Les paramètres sont du type à mot-clé ; les paramètres obligatoires sont **DN, PDN,** et **ID.**

DN = dn Nom du fichier local au travail. Le fichier doit être fermé avant d'être catalogué. Ce nom s'écrit avec 1 à 7 caractères.

PDN = pdn Nom de catalogue du fichier, écrit avec 1 à 15 caractères, fourni par l'utilisateur qui crée le fichier ; les 2 premiers caractères doivent être le code du terminal, les 13 suivants sont libres.

ID = uid Nom de l'utilisateur, sur 1 à 8 caractères,

ED = ed Numéro de version du fichier, de 1 à 4095, fourni par l'utilisateur qui crée le fichier. La valeur par défaut est :

- 1 s'il n'existe aucun fichier de même **PDN** et **ID**
- le plus grand numéro courant de version 1, s'il existe un fichier permanent de même **PDN** et **ID.**

NOTA : Pour des impératifs d'exploitation, il ne sera pas gardé plus de 10 versions d'un même fichier, sauf accord préalable

R = rd Le fichier reçoit un mot de passe, **rd** (1 à 8 caractères) qui sera nécessaire pour tout accès en lecture et/ou écriture Défaut : pas de mot de passe.

W = wt Le fichier reçoit un mot de passe, **wt** (1 à 8 caractères) qui sera nécessaire pour tout accès en écriture (mais non en lecture). Défaut : accès multiple.

UQ Accès exclusif : à tout instant, un programme au plus pourra accéder au fichier. Défaut : accès multiple.

NA Le travail continue en cas d'erreur. Si ce paramètre n'est pas codé, une erreur provoquera la fin du travail.

VII.2.2. - Utiliser un fichier : **ACCESS**

L'instruction **ACCESS** associe un fichier catalogué à un travail. Après avoir formulé **ACCESS,** les références au fichier catalogué se font sous son nom local spécifié par le paramètre **DN.** L'instruction **ACCESS** doit précéder l'instruction **ASSIGN** ou l'ouverture du fichier.

Les fichiers catalogués contenant des utilitaires du système (compilateur CFT, chargeur, assembleur **CAL,** utilitaires tels **UPDATE** et **BUILD,** copies et dumps et autres fichiers choisis par l'exploitation n'exigent pas un **ACCESS.**

ACCESS, DN = dn, PDN = pdn, ID = uid, ED = ed, R = rd, W = wt, UQ, NA.

Les paramètres sont à mot-clé ; les paramètres obligatoires sont **DN, PDN** et **ID.**

DN = dn Nom local par lequel le fichier catalogué doit être nommé. Ce paramètre est obligatoire.

PDN = dn Nom de catalogue du fichier auquel on veut accéder et qui existe déjà. (voir VII.2.1 : instruction **SAVE,** paramètre **PDN**)

ID = uid Nom de l'utilisateur sur 1 à 8 caractères qui a été spécifié dans l'instruction **SAVE ; ID** doit être spécifié dans **ACCESS.**

ED = ed Numéro de version d'un fichier permanent à laquelle on accède ; valeur de 1 à 4095 qui a été assignée à la création du fichier. Si **ED** n'est pas spécifié, par défaut, la version prise en compte est

celle de plus haut rang connue par le système. (Voir VII.2.1, NOTA, instruction **SAVE**, paramètre **ED**).

R = rd Mot de passe en lecture, nécessaire s'il a été spécifié au **SAVE**.

W = wt Mot de passe en écriture, nécessaire s'il a été spécifié au **SAVE** et si le programme doit modifier le fichier. Dans ce cas, **UQ** doit être également codé.

UQ Spécifie l'accès exclusif au fichier.

NA Le travail continue en cas d'erreur. Si **NA** n'est pas codé, une erreur provoquera l'arrêt du travail.

VII.2.3. - Détruire un fichier : **DELETE**

L'instruction **DELETE** permet de rendre local un fichier catalogué ; elle exige un **ACCESS** préalable. Si le fichier a été catalogué par un **SAVE** incluant respectivement un mot de passe en écriture **W = wt** et/ou la spécification d'accès exclusif **UQ**, **l'ACCESS** doit comprendre les mêmes paramètres. A la suite d'un **DELETE**, le fichier se comporte comme un fichier local. Il peut être lu et écrit. Il sera détruit soit en fin de travail, soit sur une insruction **RELEASE.**

Format :

DELETE, DN = dn, NA.

DN = dn Nom local d'un fichier catalogué pour lequel on a réalisé un **ACCESS.**

NA Le travail continue en cas d'erreur. Si ce paramètre est omis, une erreur provoquera la fin anormale du travail.

VIII. - TRANSFERTS DE FICHIERS ENTRE CRAY ET FRONTAL

Deux instructions de contrôle permettent la gestion des transferts des fichiers entre le Cray et les calculateurs frontaux : **ACQUIRE** et **DISPOSE.**

ACQUIRE atteint un fichier installé sur un système frontal, le catalogue et le rend accesible au travail concerné. Si le fichier est déjà catalogué sur le Cray, **ACQUIRE** permet l'accès à ce fichier.

DISPOSE dirige le fichier vers la file d'attente d'entrée en CRAY ou vers la file d'attente de sortie des calculateurs frontaux. **DISPOSE** peut ainsi être utilisé pour libérer la place d'un fichier ou pour changer ses caractéristiques.

VIII.1 - Frontal vers Cray : **ACQUIRE**

- L'instruction **ACQUIRE** permet de cataloguer un fichier et le rend accessible par le travail concerné. Lors d'un **ACQUIRE**, le système opératoire Cray détermine si le fichier est résident en mémoire de masse du Cray ou s'il appartient aux systèmes frontaux.

Si le système du Cray détermine que le fichier est déjà catalogué sur Cray, il le rend accessible au travail demandeur.

Si le fichier n'appartient pas au Cray, il est demandé pour transfert depuis le système frontal, puis catalogué sur Cray et rendu accessible au travail demandeur. L'exécution du travail est retardée tant que le fichier n'est pas catalogué.

ACQUIRE, DN = dn, PDN = pdn, ID = uid, ED = ed, TEXT = text, DF = df.

Les paramètres sont à mot-clé. Les paramètres obligatoires sont **DN, PDN, ID.**

DN = dn Nom local du fichier pour lequel le fichier est connu.
Ce paramètre est obligatoire.

PDN = pdn Nom du fichier catalogué auquel on doit accéder, ou que l'on doit chercher sur un calculateur frontal, avant de le cataloguer et d'y accéder ; le nom, est formé de 1 à 15 caractères ; les 2 permiers caractères représentent le code du terminal ; les 13 suivants sont libres.

ID = uid Nom de l'utilisateur, formé de 1 à 8 caractères alphanumériques.

ED = ed Numéro de version ayant pour valeur 1 à 4095, assigné à la création du fichier ; par défaut :
- 1, s'il n'existe pas de fichier ayant le même **PDN** ou **ID**
- le plus grand numéro de version courant existant, s'il existe un fichier avec ce **PDN** et cet **ID.**

NOTA : Pour des impératifs d'exploitation, il ne sera pas gardé plus de 10 versions d'un même fichier sauf accord préalable exceptionnel.

TEXT = text Texte contenant les paramètres de la carte **DD** décrivant sur IBM le fichier traité. Text ne doit pas comporter plus de 240 caractères.

DF = df Format du fichier, codé sur deux caractères alphanumériques ; ce paramètre indique si le fichier à présenter au Cray est en format bloqué Cray ou si le système frontal doit effectuer la conversion en ASCII. Par défaut DF = CB.

CB format caractères, bloqué ; le système bloque le fichier avant de le transférer au Cray et si nécessaire effectue la conversion en ASCII (8 bits).

BB format binaire bloqué ; le calculateur frontal bloque le fichier avant de le transférer au Cray mais n'effectue aucune conversion.

TR format "transparent" ; aucun blocage ni conversion n'est effectué.

VIII.2. - Cray vers frontal = **DISPOSE**

L'instruction **DISPOSE** dirige un fichier dans la file d'attente de sortie du Cray (ou d'entrée si DC = IN) pour réaliser le transfert vers (vu depuis) le calculateur frontal. **DISPOSE** peut aussi être utilisé pour modifier les caractéristiques d'un fichier ou le détruire.

L'instruction **DISPOSE** ne retarde pas l'exécution d'un travail à moins qu'on ait codé **WAIT** (cf. ci-après).

Format :

DISPOSE, DN = dn, SDN = sdn, DC = dc, DF = df,
ID = uid, ED = ed,
TEXT = text, NOWAIT, DEFER, NRLS.

Les paramètres sont de type à mot-clé ; le seul paramètre obligatoire est **DN.**

DN = dn Nom par lequel le fichier est connu sur le Cray 1.
Ce paramètre est obligatoire.

SDN = sdn Nom de transfert du fichier, formé de 1 à 15 caractères, les deux premiers caractères représentent le code du terminal ; les 13 suivants sont libres.

DC = dc Code disposition attribué au fichier en fin de traail ; la valeur par défaut du code est **PR** (le fichier est déclaré de type **OUT PUT**).

DC est codé avec deux caractères alphabétiques :

IN le fichier est placé dans la file d'attente d'entrée du Cray. Ceci correspond à une génération de travail Cray.

ST le fichier est transféré vers le calculateur frontal ayant soumis le travail.

SC le fichier est détruit.

PR le fichier sera imprimé sur l'imprimante connectée à l'unité centrale d'origine.

PU le fichier sera perforé sur la perforatrice connectée à l'unité centrale d'origine.

PT le fichier sera tracé sur le matériel de dessin connecté à l'unité centrale d'origine.

MT le fichier sera écrit sur bande magnétique, sur l'unité frontale ayant soumis le travail. Seules les bandes standard label IBM sont prises en compte. Un seul dérouleur est disponible pour la station Cray.

DF = df Format du fichier, codé sur deux caractères alphanumériques. Ce paramètre indique si le fichier à présenter au Cray est en format bloqué Cray ou si le système frontal doit effectuer la conversion en ASCII. Par défaut,
DF = CB pour **DC = PR** ou **PU** ou **PT.**

CB format caractère, bloqué,
. le système frontal bloque le fichier avant de le transférer au Cray et si nécessaire effectue la conversion en ASCII (8 bits).

BB format binaire bloqué ; le calculateur frontal n'effectue aucune conversion. Le CRAY n'effectue pas de déblocage avant le transfert. Pour DISPOSE, BB est équivalent à TR.

TR format "transparent" ; aucun blocage ni conversion n'est effectué.
D'autres codes peuvent être écrits ; dans ce cas le fichier est traité en **TR.**

ID = uid Nom de l'utilisateur, codé sur 1 à 8 caractères alphanumériques.

ED = ed Numéro de version, validé seulement si DC = ST, dont la valeur est de 1 à 4095. La valeur par défaut dépend du calculateur frontal destinataire. (Maximum actuellement fixé par l'exploitation : 10).

TEXT = text Texte image des paramètres de la carte DD décrivant le fichier sur IBM. Le format de **text** dépend de la gestion des fichiers sur calculateur frontal, **text** ne doit pas comporte plus de 240 caractères.

NOWAIT Pas de mise en attente du travail. Si ce paramètre est codé le travail est exécuté sans attendre que le fichier soit transféré vers le calculateur frontal ; si le calculateur frontal annule le transfert du fichier, le travail se déroule sans fin anormale ; d'une manière standard **NOWAIT** est pris en compte. L'utilisateur n'est pas prévenu du déroulement de **DISPOSE.**
L'emploi de **NOWAIT** est conseillé.

DEFER Signifie que le fichier sera transféré, soit après un **RELEASE** ou en fin de travail.

NRLS Pas de libération du fichier. Quand ce paramètre est codé le fichier reste local au travail après l'exécution du **DISPOSE.** Quand **NRLS** a été codé, on ne peut que libérer le fichier ou faire un autre **DISPOSE.**

ANNEXE 1

QUELQUES EXEMPLES DE TRAVAUX COURANTS

Les exemples suivants sont des petits tests illustrant les travaux les plus couramment soumis :

- compilation
- execution
- sauvegarde de fichier sur le CRAY ou le frontal IBM
- Creation et utilisation de bibliotheques.

Les exemples de programmes Fortran donnes ne sont pas des modeles de programmation et ne servent qu'a lire ou a ecrire sur des fichiers.

Dans les exemples suivants:

nomdejo :nom de job-cray sur 7 caracteres.On repete les 7 premiers caracteres du nom de job-ibm

xxxx :numero d etude sur 4 caracteres(le meme que pour ibm)

****
EXEMPLE 1
****

Lecture de donnees sur le fichier d'entree standard $IN
Creation en ecriture et remplissage du fichier de nom local FT01 (correspondant en Fortran a l'unite logique 1) et de nom permanent MONFICHIER.

****

```
JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
LDR.
SAVE,IN=FT01,PDN=MONFICHIER,ID=NOMUTILI.
/EOF
      PROGRAM ESSAI1
      DIMENSION IA(10,10),N(10)
      READ 200,N
      DO 50 J=1,10
            DO 10 I=1,10
10                IA(I,J)=I*N(J)
50          CONTINUE
      DO 70 J=1,10
70          WRITE(1,100) (IA(I,J),I=1,10)
200   FORMAT(10I5)
100   FORMAT(10I8)
      STOP
      END
/EOF
```

****
EXEMPLE 2
****

- Lecture du fichier de nom local FT04 (Fortran : unite logique 4), de nom permanent MONFICHIER (sauvegarde dans l'exemple 1) ;
- Creation du fichier de nom local FT10 (u.l. 10) ;
- Sauvegarde de ce fichier sur IBM sous le nom permanent

SYSTEME.XXESSAI1.DATA

```
**********
.JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
ACCESS,DN=FT04,PDN=MONFICHIER,ID=NOMUTILI.
LDR.
DISPOSE,DN=FT10,DC=ST,DF=BB,TEXT='DSN=SYSTEME.XXESSAI1.DATA,'^
'DISP=(NEW,CATLG),UNIT=3350,VOL=SER=TEST19,SPACE=(TRK,(5,1)),'^
'DCB=(RECFM=FB,LRECL=80,BLKSIZE=16000)',WAIT.
/EOF
      PROGRAM ESSAI2
      DIMENSION IA(10,10),IB(10,10)
      DO 10 I=1,10
10          READ (4,100) (IA(K,I),K=1,10)
      DO 20 J=1,10
          DO 15 I=1,10
15            IB(I,J)=IA(I,J)*5
20        CONTINUE
      DO 30 I=1,10
30        WRITE(10,100) (IB(J,I),J=1,10)
100   FORMAT(10I8)
      STOP
      END
/EOF
```

****

EXEMPLE 3

****

```
      Lecture du fichier FT09 de nom permanent SYSTEME.ESSAI1.DATA
      sauvegarde precedemment sur IBM.
      impression des resultats.
```

****

```
JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
ACQUIRE,DN=FT09,DF=BB,TEXT='DSN=SYSTEME.XXESSAI1.DATA,DISP=SHR'.
LDR.
/EOF
      PROGRAM ESSAI3
      DIMENSION IA(10,10)
      DO 10 I=1,10
10    READ(9,100)(IA(J,I),J=1,10)
      DO 20 J=1,10
20    PRINT 200,(IA(I,J),I=1,10)
      STOP
100   FORMAT(10I8)
200   FORMAT(10X,10(I8,2X))
      END
/EOF
```

```
**********************************************
                EXEMPLE 4
**********************************************
```

Creation et utilisation d'une bibliotheque de sous-programmes.

1. Creation d'une bibliotheque contenant les sous-programmes SUB1,SUB2,SUB3.
   Sauvegarde de cette bibliotheque sous le nom permanent LIBBEN2.
2. Utilisation de la bibliotheque precedemment creee.

```
**********************************************
```

1.Creation de la bibliotheque

```
JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
BUILD,I=0,OBL=0.
SAVE,DN=$NEL,PDN=LIBBEN2.
/EOF
      SUBROUTINE SUB1(A)
      A=8.
      RETURN
      END
      SUBROUTINE SUB2(B)
      B=5.7
      RETURN
      END
      SUBROUTINE SUB3
      RETURN
      END
/EOF
```

2. Utilisation de la bibliotheque

```
JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
LDR,LIB=LIBBEN2.
/EOF
      PROGRAM TEST
      CALL SUB1(X)
      CALL SUB2 (Y)
      PRINT 100,X,Y
      STOP
      END
/EOF
```

```
**********************************************
                EXEMPLE  5
**********************************************
Execution d'un travail avec sortie graphique
sur cathographe Benson.
LIB1 nom local de la bibliotheque Benson
SYSZZBENSON nom de catalogue de la biblio-
theque Benson.
FT10 reference au numero logique du fichier
des informations graphiques.
**********************************************
.JOB,JN=nomde jo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
ACCESS,DN=LIB1,FDN=SYSZZBENSON.
LDR,LIB=LIB1.
REWIND,DN=FT10.
DISPOSE,DN=FT10,DC=PT,DF=BB,TEXT='SYSOUT=E,'^
'DCB=(RECFM=F,LRECL=936,BLKSIZE=936)',WAIT.
/EOF
          cartes fortran du programme
/EOF
```

```
**********************************************
                EXEMPLE 6
**********************************************
Execution d'un travail avec sortie graphique
sur traceur mecanique Benson "on line" sur
un terminal Retina.
LIB1 nom Local de la bibliotheque Benson


SYSZXBENSON nom de catalogue de la bibliothe-
que Benson.
FT10 reference au numero logique du fichier
des informations graphiques
cter identificateur du terminal
**********************************************
.JOB,JN=nomdejo,US=xxxx.
ACCOUNT,AC=xxxx.
CFT.
ACCESS,DN=LIB1,FDN=SYSZXBENSON.
LDR,LIB=LIB1.
REWIND,DN=FT10.
DISPOSE,DN=FT10,DC=PT,DF=BB,TEXT='SYSOUT=I,DEST=cter,'^
'DCB=(RECFM=F,LRECL=240,BLKSIZE=240),WAIT.
/EOF
              cartes fortran du programme
/EOF
```

ANNEXE 2

Format des fichiers à transférer au Cray.

Correspondance entre les formats des fichiers Cray et les **DCB** de ces fichiers sur IBM.
Paramètres **DF** et **TEXT** des cartes **ACQUIRE** et **DISPOSE.**

Fichier sur Cray. Data Format (DF)	Fichier sur IBM. DCB	
TR	**F**	4096 seulement
CB ou **BB**	**FB (A)** **VB (A)** **VBS**	N'importe quelle valeur pour le BLKSIZE taille de piste IBM 3350
CB ou **BB**	**F**	N'importe quelle valeur sauf 4096 taille de piste IBM

Ceci s'applique pour **DC = ST**

NOTA. Le format **TR** n'est pas exploitable sur IBM ; mais c'est ce format qui doit être utilisé pour les fichiers qui sont stockés sur IBM en attente de réutilisation sur le Cray.
Le format **CB** est le seul exploitable sur IBM.

ANNEXE 3
BIBLIOGRAPHIE

NOTES EDF CONCERNANT LE CRAY

CR1 AL24 B. MEYER Un calculateur vectoriel : Le Cray 1 et sa programmation

Juin 1980

CR2 M. MÅGNIER Langage de contrôle du calculateur Cray

Avril 1981

CR3 B. MEYER Soumission des travaux sur le CRAY.

Novembre 1980

NOTES CRAY

La note de base sur le langage de commande est :

CRAY-OS Version 1 Référence Manual, ST-0011.

Elle peut être consultée à l'assistance et aux terminaux.

********** INDEX **********

CET INDEX A ETE CONSTITUE A L AIDE DE LA PROCEDURE INDEX DE L ATELIER LOGICIEL. VOUS TROUVEREZ LA DESCRIPTION DE CELLE-CI DANS LA NOTE ATELIER LOGICIEL NUMERO 25, QUE VOUS POURREZ VOUS PROCURER AUPRES DE MME FUMADELLES BUREAU C121 POSTE 3746 A CLAMART.

LE SIGNE * PLACE DEVANT UN NUMERO DE PAGE INDIQUE QUE LE TERME EST DEFINI A CETTE PAGE.

$

$ABD 29
$BLD 36
$CS 12
$IN 12, 30
, 35

$LOG 12
$NBL 36
$OBL 36
$OUT 9, 12
, 31-32
, 35

A

A 40
AB 27
AC 20
ACCESS 5, 28
, *40
, 42
, *44

ACCOUNT *20
ACQUIRE *46
appel au compilateur:voirCFT
ASSIGN 6, 9

B

B 21, 35

BB 48, 50

BCW 8
BS 6, 40

BUILD 35
directive 37

C

C 21, 27

carte ACCOUNT:voir ACCOUNT
carte TRAVAIL:voirJOB
CB 48, 50

CFT *21
chargement d un programme:voirLDR
chargeur:voir LDR
CL 20
compilateur
options 23
compilateur:voir CFT
compilation:voir CFT
concatenation de bibliotheques 37
COPY 39

D

DC 48-49
DEFER 48
DELETE 42
DF 46, 48

directive:voir BUILD
DISPOSE 9, 46
, *48

DN 27, 40-41

, 43-44
, 46
, 48

E

E 21, 27

ED 43-44, 46, 48

EOD 10
EOF 10
EXCLUDE 33

F

Fichier *5
acces 40
catalogue 5, 42

code disposition 9
creation 40
fin d acces 40
format bloque 7
format non bloque 9
local 5
nom 7
resident en memoire 6
FN 33-34
FROM 37

I

I 21, 27, 35

ID 43-44, 46, 48

IN 49
INCLUDE 33-34

J

JN 19
JOB 10, *19

L

L 21, 27
, 35

LDR *27
chargement selectif 33-34
LIB 27
LIST 39
LLD 27

M

M 19
MAP 27
MOD 33-34
MR 6
MT 50

N

NA 27, 43-44

NBL 35
NODIR 35
NOLIB 27
NOWAIT 48
NRLS 48
NX 27

O

OBL 35
OFF 21
OLM 20
OMIT 38
ON 21
OVL 27

P

PDN 43-44, 46

PR 49
PT 50
PU 50

R

R 43-44
RCW 8
RDM 40
regle d ecriture carte jcl:voir syntaxe carte jcl
RELEASE 5, 29
, 40
, *41

S

S 40
SAVE 5, 42
, *43
SC 9, 49
SDN 33-34, 48
SET 27
SORT 35
ST 49
syntaxe carte jcl *16
carte suite 18
commentaire 17
parametre 16
separateur 16
terminaison 17
verbe 16
SYSZXBENSON 28
SYSZXGENERALE 28

T

T 19, 27
TEXT 46, 48
TIO 20
TR 48, 50
travail *10
achevement 13
deroulement 11
execution 12
historique 13
initialisation 11
prise en compte 11
reprise 13
structure 10
TRUNC 21

U

U 40
UQ 43–44
US 20
USA 27

W

W 43–44

Division APCOL

VOS COMMENTAIRES S'IL VOUS PLAIT

Votre opinion nous est précieuse. Adressez-nous vos remarques sur cette note et le produit qu'elle décrit. Merci !

Note : Langage de commande du Cray-1, note Cray n° 2, HI/3560-01

Version : 1

Feuille à adresser à : M^me M. MAGNIER
EDF/DER, Service IMA
1 avenue du Général de Gaulle 92141 CLAMART
Bureau C 101, tél. (1) 765 37 37

VOS REMARQUES :