

SOLAR

MACP

Générateur de macros

MACRO PROCESSEUR MACP

MANUEL DE REFERENCE

- △ en haut de page indique le changement complet de la page par rapport à l'IE précédent
- | en marge indique la partie modifiée par rapport à l'IE précédent

SOMMAIRE

	pages
1 - INTRODUCTION	1
1.1 - PRESENTATION DE MACP	1
1.2 - CARACTERISTIQUES DE MACP	1
1.3 - PROCEDURE D'UTILISATION DE MACP	3
2 - REGLES D'ECRITURE	4
2.1 - GENERALITES	4
2.2 - CARACTERES DE CONTROLE	4
2.3 - STRUCTURE DU TEXTE SOURCE	5
2.3.1 - DIRECTIVES	5
2.3.2 - MACRO-DEFINITIONS	6
2.3.3 - MACRO-INSTRUCTIONS	6
2.3.4 - LIGNES A GENERER	7
2.4 - STRUCTURE DU TEXTE OBJET	7
3 - MACRO-DEFINITIONS ET MACRO-INSTRUCTIONS	8
3.1 - PARAMETRES	8
3.2 - MACRO-DEFINITIONS	8
3.3 - MACRO-INSTRUCTIONS	11
3.4 - EXEMPLES D'UTILISATION	12
3.5 - CLASSEMENT DES MACRO-DEFINITIONS	13
3.6 - MODIFICATION D'UN PARAMETRE	15
3.7 - MODIFICATION D'UNE MACRO-DEFINITION	17
4 - VARIABLES	18
4.1 - EXPRESSIONS	18
4.1.1 - EXPRESSIONS ARITHMETIQUES	18
4.1.2 - EXPRESSIONS CHAINES	18
4.2 - VARIABLES	19
4.2.1 - VARIABLES ENTIERES	19
4.2.2 - VARIABLES CHAINES	19
4.3 - UTILISATION DES VARIABLES	20
5 - DIRECTIVE SKIP	22
6 - DIRECTIVE IF	24
6.1 - IF ARITHMETIQUE	24
6.2 - IF LOGIQUE	25
7 - DIRECTIVES DØ ET ENDØ	26
7.1 - DIRECTIVE DØ	26
7.2 - DIRECTIVE ENDØ	26
8 - DIRECTIVE KILL	30

9 - DIRECTIVE C	31
10 - DIRECTIVE EØT	33
11 - DIRECTIVE END	34
12 - MARQUEUR D'ETIQUETTES	35
13 - AIDES A LA MISE AU POINT : DIRECTIVES TRACE ØN ET TRACE ØFF	37
14 - EXEMPLE	39
15 - UTILISATION DU MACRO-PROCESSEUR	42
15.1 - INTRODUCTION	42
15.2 - ACTIVATION DU MACRO-PROCESSEUR	43
15.2.1 - COMMANDE IMAC	43
15.2.2 - COMMANDE CMAC	43
15.2.3 - COMMANDE LMAC	44
15.3 - CORRECTION DES ERREURS	44
15.3.1 - MESSAGES D'ERREUR	44
15.3.2 - CORRECTIONS	46
15.4 - INTERRUPTION D'UN TRAITEMENT EN COURS	47
15.5 - CHARGEMENT DU MACRO-PROCESSEUR MACP	47
15.5.1 - UTILISATION DU CHARGEUR AUTONOME	48
15.5.2 - UTILISATION DU CHARGEUR INTEGRE	49
15.6 - UTILISATION DU MACRO-PROCESSEUR MACP-D	51
15.6.1 - PRINCIPE	51
15.6.2 - MODIFICATION DE LA FU SUPPORT DE L'ESPACE VIRTUEL	51
15.6.3 - MODIFICATION DE LA TAILLE DE L'ESPACE VIRTUEL STANDARD	52
15.6.4 - INTEGRATION SOUS LES SYSTEMES TSM ET TSF	52
15.6.5 - CONTRAINTES	52
15.6.6 - ERREUR AVEC LE FICHER ESPACE VIRTUEL	53
16 - SYNOPTIQUE	54
ANNEXE : LISTE DES NUMEROS D'ERREUR	57

1 - INTRODUCTION

1.1 - PRESENTATION DE MACP

L'objectif fondamental du macro-processeur MACP est d'assister l'utilisateur en lui proposant un outil d'une très grande souplesse offrant, en plus des possibilités classiques, le moyen de réaliser des opérations telles que traduction de langage et configuration de programme.

La technique de réalisation adoptée, en permettant à l'utilisateur de définir la syntaxe de ses macro, l'autorise à écrire ses programmes dans un langage spécialisé qui est reconnu par le macro-processeur.

D'autre part, l'utilisation de MACP n'est pas limitée aux programmes Assembleur. Le code généré peut également être PL 16, FORTRAN, etc...

L'existence de variables et de directives propres au macro-processeur permet de résoudre efficacement tout problème de configuration de système (adaptation à une application donnée).

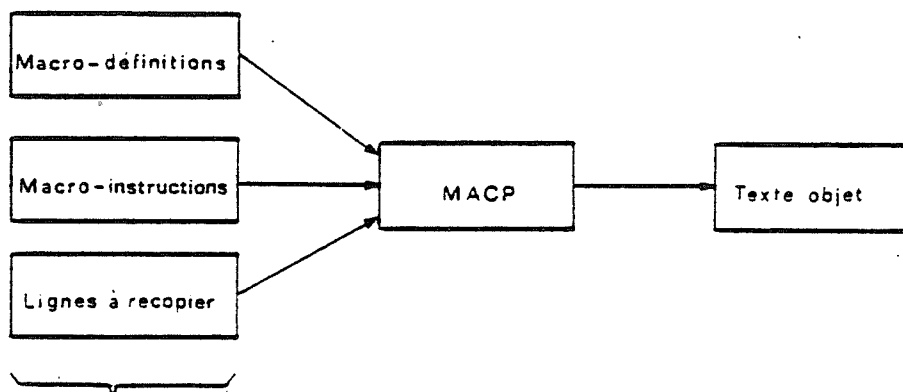
1.2 - CARACTERISTIQUES DE MACP

Le macro-processeur MACP est un programme conversationnel permettant de résumer dans une macro-définition une suite quelconque de lignes et de s'y référer ensuite par une simple macro-instruction :

- MACP identifie chaque macro-définition et conserve la copie de la chaîne qui lui est associée,
- MACP identifie chaque macro-instruction et génère la suite associée à la macro-définition correspondante en substituant les paramètres effectifs de la macro-instruction aux paramètres formels inclus dans la macro-définition.

Le macro-processeur MACP génère un texte objet à partir d'un texte source dont la composition est la suivante :

- macro-définitions qui sont mémorisées et dont le traitement est différé jusqu'au moment de leur appel,
- macro-instructions traitées immédiatement,
- lignes non reconnues comme macro-instructions ou macro-définitions et qui sont à recopier dans le texte objet.



Texte source

Macro-définitions et macro-instructions peuvent être lues par le macro-processeur sur des supports physiques différents, ce qui permet, par exemple, de conserver la bibliothèque des macro-définitions sur disque.

Le macro-processeur MACP possède les particularités suivantes:

- . Fonctionnement en un seul passage:

Le macro-processeur délivre un texte objet en ne faisant qu'une seule lecture du texte source.

- . Conversationnalité :

Les erreurs détectées provoquent l'impression d'un message. L'utilisateur peut alors apporter toute correction utile avant de demander la poursuite de la génération.

- . Langage source varié :

Le macro-processeur utilise une méthode connue sous le nom de "template matching" qui permet à l'utilisateur de définir lui-même la syntaxe des macro-instructions.

- . Génération paramétrée et conditionnelle :

Le langage MACP comporte un certain nombre de directives permettant d'adapter un programme à toute configuration particulière.

La richesse du jeu de directives permet en outre l'utilisation de MACP en tant que macro-générateur pour l'écriture de véritables langages spécialisés.

. Marqueur d'étiquettes :

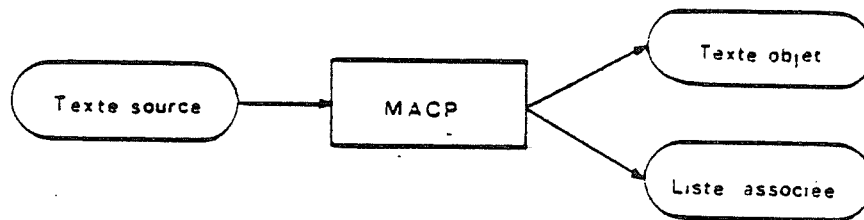
Un marqueur automatique permet l'utilisation d'étiquettes symboliques à l'intérieur d'une macro-définition sans risquer les conflits propres à la définition multiple de symboles ce qui, sans cette facilité, serait le cas lors des appels successifs d'une telle macro.

. Modification des macro-définitions :

On peut à tout moment abandonner la chaîne associée à un nom de macro et associer à ce nom une nouvelle chaîne qui représente la modification de la macro-définition.

1. 3 - PROCEDURE D'UTILISATION DE MACP

A partir d'un texte source, le macro-processeur MACP produit un texte objet ainsi que la liste qui lui est associée.



Deux versions du macro-processeur sont disponibles :

- MACP,
- MACP-D, version disque.

Le macro-processeur MACP s'exécute sous le contrôle des systèmes d'exploitation BOS-A, BOS-B et BOS-C sur des configurations minimales de 8 K mots.

Le macro-processeur MACP-D s'exécute sous le contrôle de BOS-D et BACKM, moniteur permettant une activité Background sous RTES-D.

2 - REGLES D'ECRITURE





2.1 - GENERALITES

Un texte source peut être écrit en utilisant tous les caractères imprimables du code ASCII, les autres caractères étant ignorés par le macro-processeur ou signalés comme erreur lorsqu'ils sont impairs.

Il est structuré en lignes, chacune d'elles comportant 80 caractères (cartes perforées) ou se terminant par un retour-chariot (ruban perforé).

Toute ligne du programme est tronquée lorsqu'elle comporte plus de 72 caractères, de telle sorte que les colonnes 73 à 80 des cartes sont ignorées.

Deux caractères spéciaux facilitent l'écriture :

- la flèche  provoque l'annulation du début de l'enregistrement (ligne ou carte) ;
- la flèche  provoque l'annulation du caractère imprimable significatif qui la précède (exceptions faites pour les caractères ,  et retour-chariot).

Ainsi la ligne suivante :

```
* DEF ← LA ADRESS↑↑↑M
```

est équivalente à la ligne :

```
LA ADREM
```



2.2 - CARACTERES DE CONTROLE

Un certain nombre de caractères sont utilisés par le macro-processeur MACP en tant que caractères de contrôle.

Ils jouent les rôles suivants :

- fin de ligne logique, notée (FL)
- indicateur de directive, noté (ID)
- indicateur de paramètre, noté (IP)
- indicateur de variable, noté (IV)
- indicateur de macro-instruction, noté (IM)
- indicateur de génération, noté (IG)

Implicitement MACP identifie les caractères ASCII suivants :

- retour-chariot (FL)
-  (ID)
-  (IP)

- (IV)
- (IM)
- (IG)

Il est cependant possible de définir d'autres caractères. Ce point sera détaillé au chapitre 9.

La fin physique d'une ligne est toujours le retour-chariot. Cependant, lorsque l'utilisateur définit une fin logique différente, le macro-processeur considère tous les caractères apparaissant après (FL) comme un commentaire.

Exemple :

```
* DEF?:=? ; INSTRUCTION D'AFFECTION
                |
                |----- commentaire
                |
                |----- fin de ligne logique
```

Remarque :

Dans toute ligne source les caractères (IP), (IV) et (IM) peuvent être utilisés sans être considérés comme caractères de contrôle par le macro-processeur. Il suffit de les doubler. Ainsi la ligne source:

```
QUESTION ??
```

donnera lieu à la génération dans le texte objet de :

```
QUESTION ?
```

2. 3 - STRUCTURE DU TEXTE SOURCE

Quatre types d'éléments peuvent apparaître dans un texte source. Ce sont :

- directives
- macro-définitions
- macro-instructions
- lignes à générer

Les macro-définitions sont mémorisées par le macro-processeur alors que les trois autres éléments sont traités immédiatement.

2.3.1 - DIRECTIVES

Une directive est une ligne du texte source dont le premier caractère est (ID).

Elle est reconnue par le macro-processeur comme une consigne qui lui est donnée comme, par exemple, affecter une valeur à une variable :

```
* V01 = 12
```

ou ignorer un certain nombre de lignes :

* SKIP 5

Une directive peut apparaître à l'intérieur d'une macro-définition ou à l'extérieur.

Dans ce dernier cas elle est exécutée immédiatement par le macro-processeur, alors qu'à l'intérieur son exécution n'est effective qu'au moment d'un appel de la macro.

2.3.2 - MACRO-DEFINITIONS

La définition d'une macro consiste à associer à une chaîne de caractères appelée "corps de la macro" un nom appelé "en tête de la macro".

A chaque macro correspond donc un en-tête et un corps.

Une macro-définition a la forme suivante :

```
* DEF en tête (FL)
corps
* ENDEF (FL)
```

Les deux directives du macro-processeur DEF et ENDEF ont pour rôle la délimitation de la définition.

2.3.3 - MACRO-INSTRUCTIONS

Une macro-instruction est une ligne du texte source constituant un appel de macro.

Faisant référence à une macro définie antérieurement, elle s'écrit en utilisant la même structure que l'en-tête correspondant .

MACP analyse toute ligne du texte source. Si une concordance peut être établie avec une structure d'en-tête, la ligne est considérée comme un appel de macro.

Dans le cas contraire, la ligne est reproduite dans le texte objet.

Afin d'éviter que, à la suite d'une erreur d'écriture, une macro-instruction ne soit pas reconnue par MACP, l'utilisateur dispose de l'indicateur de macro (IM).

Lorsque son premier caractère est (IM) toute ligne source à laquelle MACP ne peut faire correspondre aucune structure donne lieu à un message d'erreur.

MACP procède, dans des conditions absolument identiques, à l'expansion de toute macro-instruction, qu'elle soit ou non précédée de (IM).

2.3.4 - LIGNES A GENERER

Toute ligne source qu'il n'a pu reconnaître en tant que directive ou macro-instruction est interprétée par MACP comme une ligne à générer dans le texte objet.

La génération est immédiate lorsqu'une telle ligne apparaît hors d'une macro-définition.

Elle est différée et ne s'effectue qu'au moment de l'appel de la macro dans le cas contraire.

En outre la présence de l'indicateur de génération (IG) au début d'une ligne du texte source impose à MACP la génération de cette ligne après suppression de l'indicateur.

Ainsi la ligne source :

```
\ * END
```

donnera lieu à la génération dans le texte objet de :

```
* END
```

(non reconnaissance par MACP de la directive END).

2 . 4 - STRUCTURE DU TEXTE OBJET

Le texte objet est structuré en lignes.

Le macro-processeur MACP génère un retour - chariot suivi d'un saut de ligne pour chaque fin de ligne objet.

Lorsque l'utilisateur a défini un caractère fin de ligne logique (FL), la partie commentaire des lignes sources comprise entre (FL) et la fin de ligne physique, ne donne lieu à aucune génération.

Le macro-processeur est indépendant du langage généré. Il peut être employé pour générer des textes objets en Assembleur, PLI6 , FORTRAN, etc ...

3 - MACRO-DEFINITIONS ET MACRO-INSTRUCTIONS

3. 1 - PARAMETRES

Une macro peut être employée dans le cas où une séquence est à répéter en plusieurs endroits d'un texte.

Dans la pratique il arrive cependant fréquemment qu'une suite d'instructions se répète, mais avec quelques différences. Les paramètres sont un moyen de transmettre, lors de l'appel, ces variantes qui vont modifier le corps de la macro à expander.

Une définition de macro comporte deux parties :

- l'en-tête qui donne le nom (structure à respecter par toute macro-instruction d'appel) ainsi que l'emplacement des paramètres dans la macro d'appel
- le corps qui représente la description des opérations à effectuer au moment de la génération de la macro.

Les paramètres formels sont des identificateurs (IP) apparaissant dans la description de la macro. Ils seront remplacés au moment de la génération par les paramètres effectifs.

Ainsi la macro-définition suivante :

```
SØM?+?
```

peut être utilisée de différentes manières dans un même texte source.



Son appel par la ligne source :

```
SØMPARAM1 +PARAM2
```

signifie que la génération sera effectuée avec les deux paramètres effectifs PARAM1 et PARAM2 substitués, dans le corps de la macro, à toute référence aux paramètres formels correspondants.

3. 2 - MACRO-DEFINITIONS

Lors d'une macro-définition, l'en tête est la chaîne de caractères apparaissant en opérande de la directive DEF.

Elle peut comporter des identificateurs de paramètres (IP) et des caractères imprimables (autres que  et ), appelés "séparateurs".

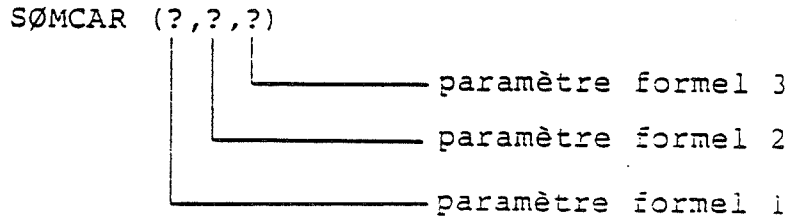
Exemples :

```
?:=?
```

```
IF?THEN?ELSE?
```

L'ordre dans lequel sont placés les paramètres formels est essentiel puisque, lors de leur utilisation, c'est uniquement leur position qui permettra de les distinguer.

Exemple :



A l'intérieur du corps de la macro, une ligne quelconque peut faire référence aux paramètres.

La syntaxe d'une telle référence est la suivante :

(IP) f n

où :

- (IP) est le caractère identificateur de paramètre,
- f représente le type de l'opération à effectuer sur le paramètre,
- n est le numéro du paramètre.

MACP admet, au plus, 9 paramètres par macro.

Les trois opérations permises sont les suivantes :

- le remplacement du paramètre formel par le paramètre effectif

f = P

- le remplacement du paramètre formel par la longueur du paramètre effectif (nombre de caractères composant le paramètre)

f = L

- le remplacement du paramètre formel par le type du paramètre effectif

f = T

Cinq types de paramètres sont reconnus:

- nombre hexadécimal (type = 1)
- nombre décimal (type = 2)
- chaîne de caractères (type = 3)
- symbole (type = 4)
- autre (type = 5)

Aucun contrôle n'est effectué sur le nombre de caractères composant le paramètre.

a) Nombre hexadécimal

Les nombres hexadécimaux sont toujours précédés d'une apostrophe.

Exemples :

'1A
'FFFE

b) Nombre décimal

Les nombres décimaux positifs sont précédés ou non du signe +, les nombres négatifs étant toujours précédés du signe -

Exemples :

+ 512
- 43
1247

c) Chaîne de caractères

Une chaîne de caractères est composée de caractères ASCII imprimables autres que \ominus , \oplus et retour-chariot, encadrés par des doubles apostrophes.

Exemples :

"ABCD"
"&"

d) Symbole

Un symbole est composé de caractères alphanumériques et doit comporter un premier caractère alphabétique.

Exemples :

ALPHA
M301

e) Autre type

Un paramètre ne respectant aucune des règles énoncées ci-dessus est reconnu de type égal à 5 par le macro-processeur.

Exemple :

12 + 4
&ALPHA

Remarques :

- Une macro-définition peut ne comporter aucun paramètre. Dans ce cas les séquences générées, lors des divers appels, seront en tous points identiques.

- . Une macro-définition peut ne donner lieu à aucune génération dans le texte objet si elle est vide ou ne comporte que des directives.
- . Si n est le plus grand numéro de paramètre formel figurant dans l'en-tête d'une macro-définition, il ne peut être fait référence dans le corps correspondant à un paramètre de rang plus élevé.
Ainsi la définition :

```
* DEF TEST ?  
PARAMETRE = ?P1  
LONGUEUR = ?L2  
* ENDEF
```

donnera lieu, lors de son appel, à un message d'erreur dû à la référence au paramètre 2.

3. 3 - MACRO-INSTRUCTIONS

Une macro-instruction constitue un appel d'une macro qui a été obligatoirement définie antérieurement.

Elle s'écrit en utilisant la même structure que celle de l'en-tête correspondant.

La différence avec l'en-tête provient du fait que les identificateurs de paramètres sont remplacés par des paramètres effectifs.

Une macro-instruction doit donc comporter les mêmes séparateurs que ceux ayant été introduits dans la partie en-tête de la macro-définition.

Ainsi un appel de la macro d'en-tête:

```
MACRØ ?
```

est réalisé par la ligne source suivante :

```
MACRØ PARAM  
paramètre  
effectif
```

Lors du traitement d'une macro-instruction, MACP remplace les paramètres formels du corps par les paramètres effectifs, leur longueur ou leur type (suivant l'opération demandée) avec une correspondance positionnelle entre paramètres formels et paramètres effectifs, c'est-à-dire que le paramètre formel de rang n est remplacé par le nième paramètre effectif.

On notera cependant qu'un paramètre effectif peut être la chaîne vide dont la longueur et le type sont respectivement 0 et 5.

Une macro-instruction n'est acceptée par le macro-processeur qu'après la définition de la macro correspondante.

Cependant une phrase appartenant au corps d'une macro-définition peut constituer un appel à une macro encore non définie.

Exemple :

```
* DEFMAC1
-----
% MAC2          Appel de la macro MAC2 encore non définie
-----
* ENDEF
* DEFMAC2
-----
* ENDEF
% MAC1          Appel de la macro  MAC1
```

Ceci n'est possible que par le fait que la ligne :

```
% MAC2
```

ne sera analysée que lors de l'appel de MAC1 et qu'à cet instant MAC2 a été définie.

En effet, au cours de la définition d'une macro, les seules lignes du corps de la macro analysées par MACP sont celles dont le premier caractère est (ID).

Dans ce cas, le macro-processeur effectue un contrôle syntaxique et imprime un message d'erreur lorsque la ligne n'est pas reconnue en tant que directive.

3. 4 - EXEMPLES D'UTILISATION

. La macro-définition :

```
* DEFTEST?
PARAMETRE=?P1
LONGUEUR=?L1
TYPE=?T1
* ENDEF
```

appelée par la macro-instruction :

```
TEST124
```

générera le texte objet :

```
PARAMETRE=124 .
LONGUEUR=3
TYPE=2
```


. Avec la macro-définition :

```
* DEFISI ? < ? ALLERA ?  
LA ?P1  
ADRI -?P2,A  
JAL ?P3  
* ENDEF
```

la macro instruction :

```
SI MEM < 4 ALLERA SUITE
```

générera la séquence suivante :

```
LA MEM  
ADRI - 4,A  
JAL SUITE
```

3. 5 - CLASSEMENT DES MACRO-DEFINITIONS

L'appel d'une macro est réalisé en utilisant un en-tête mémorisé par le macro-processeur.

La ligne source comporte alors des séparateurs (caractères imprimables autres que (IP) figurant dans l'en-tête) et des paramètres effectifs.

Afin de définir de manière unique l'en-tête correspondant à une ligne donnée, lorsque plusieurs en-têtes peuvent être choisis, le macro-processeur utilise un certain nombre de règles.

Règle 1 : Le processus de recherche s'effectue de la gauche vers la droite.

Règle 2 : Au moins un séparateur doit s'intercaler entre deux indicateurs de paramètres.

Ceci résulte du fait que doubler le caractère de contrôle (IP) conduit à le considérer comme un caractère ordinaire.

dans l'en-tête :

```
CALCUL???
```

deux parties sont à considérer : les séparateurs CALCUL? et un indicateur de paramètre ?

Règle 3 : Entre un indicateur de paramètre et un séparateur le macro-processeur choisit toujours le second.

Supposons les deux en-têtes :

```
?=?+?
```

```
A=?+?
```

La ligne source :

A=B+C

peut être mise en concordance avec les deux structures,
les paramètres effectifs étant B et C dans le premier cas,
A, B et C dans le second cas.

La règle précise que le séparateur A est choisi de préférence à l'indicateur de paramètre ? et MACP retient la seconde en-tête.

La mise en application de cette règle est facilitée par le fait que, à chaque enregistrement d'une nouvelle macro-définition, le macro-processeur ordonne les en-têtes.

Ainsi l'ordre des deux en-têtes est le suivant (par degré décroissant) :

A=?+?

?=?+?

Pour chaque ligne source MACP analyse les divers en-têtes dans cet ordre.

Dès qu'une correspondance peut être établie, la structure est retenue.

Remarque :

Lorsque la règle précédente ne peut être appliquée pour ordonner deux en-têtes tels :

?=?

?+?

MACP conserve l'ordre de leur apparition dans le texte source.

Avec la ligne source :

B=C+D

il retient ainsi l'en-tête "?=?".

Règle 4 : Lorsqu'il y a ambiguïté dans le choix des paramètres, le macro-processeur effectue le groupement des caractères à droite.

Avec l'en-tête :

A=?+?

la ligne source :

A=X+Y+Z

donnera comme paramètres X et Y+Z et non pas X-Y et Z.

Règle 5 : Dans un en-tête, le nombre d'espaces d'une suite d'espaces n'est pas significatif.

Les deux en-têtes :

SI _ ? _ ALØRS _ ?
SI _ _ ? _ ALØRS _ _ ?

sont identiques (_ est utilisé pour indiquer un espace)

Toute macro-instruction d'appel doit avoir au moins un espace dans les positions correspondantes.

Ainsi la ligne source :

SI _ X=1 _ _ ALØRS _ _ A=B

donnera les paramètres X=1 et A=B

Par contre :

SIX=1 _ ALØRS _ A=B

ne sera pas reconnue comme une macro-instruction.

Remarque :

Les espaces figurant à l'intérieur d'un paramètre effectif sont tous significatifs.

SI _ X= _ 1 _ ALØRS _ _ A _ NE _ _ B

donnera les paramètres X= _ 1 (de longueur 4) et A _ NE _ _ B (de longueur 7).

3. 6 - MODIFICATION D'UN PARAMETRE

Dans le corps d'une macro il est possible de définir ou de redéfinir certains des paramètres.

Cette opération est réalisée par une ligne source dont la syntaxe est la suivante :

(expression chaîne) (IP)An(FL)

n étant le numéro du paramètre ($1 \leq n \leq 9$) et l'expression chaîne (cf. 4.2.1) sa nouvelle valeur (modification du paramètre).

Supposons la macro-définition suivante :

```
* DEF?+?
-----
-----
PARAM3?A3
-----
-----
* ENDEF
```

I

II

Dans la première partie de la macro, seules sont acceptées les références aux paramètres 1 et 2 (l'en-tête ne comportant que deux indicateurs de paramètres).

Dans la seconde partie sont acceptées en plus les références au paramètre 3 qui est alors défini.

Les paramètres ainsi introduits peuvent jouer le rôle de "variables locales", qui ne sont accessibles qu'à l'intérieur de la macro.

Par opposition, les variables qui seront introduites au chapitre suivant peuvent être qualifiées de globales puisqu'elles sont connues en tout point du texte source.

La macro suivante comporte une redéfinition du paramètre 1, de sorte que deux niveaux sont à considérer : avant et après la redéfinition.

```
* DEFPERIF=?
-----
-----
DK?A1
-----
TUP?P1: EQU §
-----
* ENDIF
```

La macro-instruction :

```
PERIF=DISK
```

conduit au paramètre effectif DISK auquel est substitué ensuite DK.

A l'intérieur de la définition la ligne suivante :

```
TUP?P1: EQU §
```

génère dans le texte objet :

```
TUPDK: EQU §
```

3. 7 - MODIFICATION D'UNE MACRO-DEFINITION

On peut à tout moment abandonner le corps associé à un en-tête de macro et lui associer un nouveau corps représentant la modification de la macro-définition.

Il suffit de redéfinir la même en-tête.

Le corps précédent est alors effacé de la mémoire (d'où gain de place) et la nouvelle définition est prise en compte.

Ainsi la macro-définition :

```
* DEFMAC1  
MACRØ - INSTRUCTION  
* ENDEF
```

dont l'appel génère :

```
MACRØ - INSTRUCTION
```

peut être redéfinie :

```
* DEFMAC1  
AUTRE MACRØ  
* ENDEF
```

La macro-instruction MAC1 génère alors :

```
AUTRE MACRØ
```

4 - VARIABLES

4. 1 - EXPRESSIONS4.1.1 - EXPRESSIONS_ARITHMETIQUES

Une expression arithmétique est un assemblage convenable de

- Nombres décimaux signés
- Nombres hexadécimaux (commençant par "H")

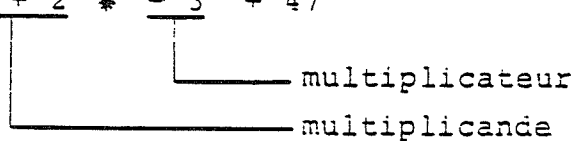
reliés entre eux par des séparateurs arithmétiques.

Les quatre opérateurs arithmétiques :

+ - * /

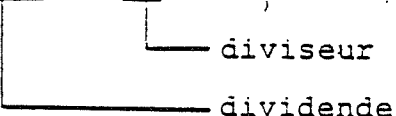
ont même priorité et l'ordre d'exécution des opérations est de la gauche vers la droite.

Le caractère \otimes apparaissant dans une expression indique la multiplication des éléments situés à sa gauche (multiplicande) par l'élément situé à sa droite (multiplicateur).

$$15 + 2 \otimes - 3 + 47$$


_____ multiplicateur
_____ multiplicande

La caractere \oslash apparaissant dans une expression indique la division entière des éléments situés à sa gauche (dividende) par l'élément situé à sa droite (diviseur).

$$16378 \oslash -2 + -4,$$


_____ diviseur
_____ dividende

La valeur d'une expression arithmétique est un nombre décimal signé calculé au moyen de l'expression elle-même.

Les deux expressions ci-dessus ont pour valeurs respectives
- 4 et - 8193.

4.1.2 - EXPRESSIONS_CHAINES

Toute chaîne de caractères (autres que \ominus et \oslash qui sont des caractères d'annulation) est une expression chaîne.

Une expression arithmétique peut donc être considérée comme une expression chaîne, la réciproque n'étant évidemment pas vraie.

Exemples :

```
SYMBOL + 2  
'1234  
CHAINE
```

La valeur d'une expression chaîne est la chaîne elle-même.

4. 2 - VARIABLES

L'utilisateur a la possibilité de définir des variables qui seront reconnues par le macro-processeur. Elles pourront donc être utilisées dans la suite du texte source.

Ces variables sont identifiées par leur nom qui comporte trois caractères : la lettre V suivie du numéro de la variable sur deux chiffres.

Le nombre maximum de variables admises par MACP étant de 100 les variables sont numérotées de 00 à 99 : V00, V01, ..., V99.

Il existe deux types de variables :

- les variables arithmétiques,
- les variables chaînes.

4.2.1 - VARIABLES_ENTIERES

Elles sont définies par les directives dont la syntaxe est la suivante :

```
(ID) V n1n2 EQ (expression arithmétique) (FL)
```

n1 et n2 étant les deux chiffres composant le numéro de la variable.

La valeur de la variable est celle de l'expression arithmétique.

Le signe n'apparaît qu'au cas où la valeur est négative.

Exemples :

```
* V01 = + 45          (valeur = 45)  
* V02 = - 33/4       (valeur = -8)
```

4.2.2 - VARIABLES_CHAINES

Elles sont définies par les directives dont la syntaxe est la suivante :

```
(ID) Vn1n2 EQ (expression chaîne) (FL)
```

n1 et n2 étant les deux chiffres composant le numéro de la variable.

La valeur de la variable est celle de l'expression chaîne.

Exemples :

```
* V03 EQ CHAINE
* V04 EQ '124
```

Le nombre de caractères de l'expression chaîne est limité à 8.

Remarque :

Une même variable peut être définie plusieurs fois dans un même programme ("entière" ou "chaîne"), chaque redéfinition annulant la définition précédente.

Exemple :

```
-----
* V05 = 15
-----
* V05 EQ CHAINE
-----
      ↑
      | V05 "entière"
      |
      ↓
      | V05 "chaîne"
      |
      ↓
```

4 . 3 - UTILISATION DES VARIABLES

Toute variable doit être définie avant son usage dans le texte source.

Toute référence à une variable à la syntaxe suivante :

(IV) Vn1n2

n1 et n2 étant les deux chiffres composant le numéro de la variable.

Après avoir ainsi défini les variables V05 et V06 :

```
* V05 EQ SYMBOL
* V06 = 12
```

la phase source :

```
*V07 EQ @V05+@V06
```


est reconnue comme la définition de la variable chaîne V07 dont la valeur est SYMBØL+12

Les références aux variables ne sont reconnues par le macro-processeur que dans toute directive identifiée par (ID) et dans toute ligne appartenant au corps d'une macro.

Toute référence à une variable est remplacée par la valeur de la variable.

Supposons la macro-définition :

```
* DEFMACRØ
* V00 EQ MACRØ
* V01 EQ INSTRUC
* V02 EQ -15
* V03 = -4
* V04 = 2 V02/ 2 V03
* V05 EQ TIØN
CECI EST LA 2 V00- 2 V01 2 V05 DE LA NO 2 V04
* ENDEF
```

Le texte suivant sera alors généré lors de chaque appel :

```
CECI EST LA MACRØ - INSTRUCTION DE NO 3
```

5 - DIRECTIVE SKIP

Cette directive a la syntaxe suivante :

```
(ID) SKIP (expression arithmétique) (FL)
```

SKIP est une directive de rupture inconditionnelle. Elle permet d'interrompre un traitement séquentiel.

La valeur de l'expression arithmétique est interprétée par le macro-processeur comme un déplacement relatif à la ligne traitée.

Ainsi dans la séquence :

```
* SKIP 3  
1ERE LIGNE IGNOREE  
2EME LIGNE IGNOREE  
REPRISE DU TRAITEMENT
```

le traitement est interrompu jusqu'à la troisième ligne suivant la directive SKIP.

Cette ligne est traitée alors que toutes celles qui la séparent du SKIP sont ignorées.

La valeur de l'expression arithmétique peut être positive ou négative.

Dans le second cas, elle donne lieu à une reprise du traitement en arrière de la directive SKIP, à la condition que ceci soit réalisé dans le corps d'une macro.

Un déplacement négatif apparaissant hors de ce contexte donne lieu à l'impression d'un message d'erreur.

Exemple :

```
* DEFMACRØ  
-----  
* SKIP 2  
-----  
* SKIP -4  
-----  
* ENDEF
```

La valeur de l'expression arithmétique peut être nulle.

L'effet produit est différent selon le contexte d'utilisation:

- exécutée à l'intérieur d'une macro la directive SKIP 0 provoque une sortie de la macro ; le traitement reprend au point d'où a été appelée la macro.
- exécutée hors d'une macro une telle directive indique la fin du traitement, de la même manière que la directive END (retour sous le contrôle du système).

Le texte source suivant :

```
* DEFTYPE?
* SKIP 2 * ?T1-1
HEXADECIMAL
* SKIP 0
DECIMAL
* ENDEF
TYPE127
* SKIP 0
```

Macro-définition

Macro-instruction

provoque la génération du texte objet :

DECIMAL

et un retour sous le contrôle du superviseur.

6 - DIRECTIVE IF

IF est une directive de rupture conditionnelle.

Elle permet la poursuite du traitement en séquence ou à une ligne qu'on lui indique, suivant qu'une relation est "vraie" ou "fausse".

La syntaxe de cette directive est la suivante :

```
(ID) IF (relation) SKIP (expression arithmétique) (FL)
```

Lorsque la relation est vérifiée, la directive IF se résume à une directive SKIP.

Dans le cas contraire, le traitement continue à la ligne suivante.

Deux types de relations peuvent intervenir. Elles conduisent à deux formes de la directive.

6.1 - IF ARITHMETIQUE

On dispose de 3 opérateurs de relation qui sont :

< = >

Une relation est alors formée de l'assemblage :

```
(expression arithmétique) { < = > } (expression arithmétique)
```

Exemple :

La macro suivante donne la valeur entière de la solution de l'équation $AX+B=0$, lorsqu'elle existe (A et B en sont les paramètres).

```
* DEF?X+?=0  
* IF ?P1=0 SKIP 4  
* V00 =-?P2/?P1  
SOLUTION=- V00  
* SKIP 0  
IMPOSSIBLE  
* ENDEF
```

Appelée par la macro-instruction :

```
4X+22=0
```

le texte généré sera :

```
SOLUTION=-5
```

6. 2 - IF LOGIQUE

On dispose de deux opérateurs logiques qui sont :

EQ NE

Une relation est alors formée de l'assemblage :

(expression chaîne) \sqcup $\left\{ \begin{array}{l} \text{EQ} \\ \text{NE} \end{array} \right\} \sqcup$ (expression chaîne)
--

Les opérateurs EQ et NE ont les significations suivantes :

EQ : équivalent
NE : non équivalent

Les deux expressions sont comparées entre elles, caractère par caractère.

Deux expressions chaîne sont équivalentes si elles ont la même longueur et sont composées de la même séquence de caractères.

Ainsi la relation :

3+4 EQ 7

est "fausse".

Exemple :

La macro suivante assigne au paramètre 1 la somme des paramètres 2 et 3. Le calcul est réalisé par l'intermédiaire du registre A.

```
* DEF?:=?+?  
* IF ?P2 EQ A SKIP 2  
LA ?P2  
ADRI ?P3,A  
* IF ?P1 EQ A SKIP 0  
STA ?P1  
* ENDEF
```

Avec la macro-instruction :

```
MEM1:=MEM2+3
```

le texte généré sera :

```
LA MEM2  
ADRI 3,A  
STA MEM1
```

alors que :

```
A:=A+5
```

provoque la génération de :

```
ADRI 5,A
```

7 - DIRECTIVES DØ ET ENDØ

7. 1 - DIRECTIVE DØ

La directive DØ est utilisée dans le corps d'une macro lorsqu'une séquence doit être répétée un certain nombre de fois.

Elle a la forme suivante :

(ID)DØ \sqsubset (expression arithmétique positive ou nulle) (FL)

La séquence est à traiter autant de fois que la valeur de l'expression arithmétique.

Le nombre maximum de boucles que l'on peut effectuer est de 252.

Lorsque l'expression arithmétique est nulle la séquence est ignorée.

7. 2 - DIRECTIVE ENDØ

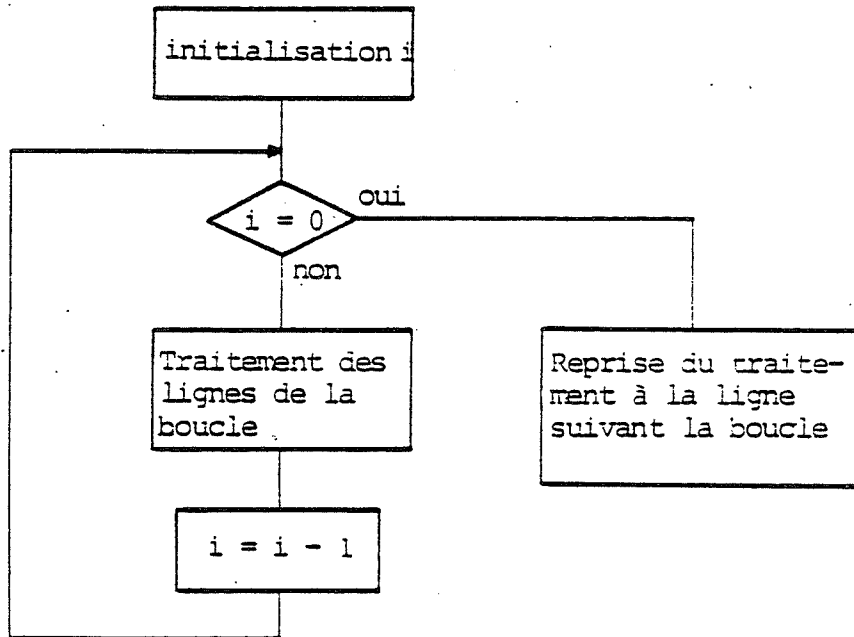
A chaque directive DØ est associée une directive ENDØ dont la syntaxe est la suivante :

(ID)ENDØ (FL)

La portée de la directive DØ est définie comme étant l'ensemble de toutes les lignes comprises entre les directives DØ et ENDØ.

Le déroulement d'une boucle DØ peut être schématisé de la manière suivante :

(i est un indice de contrôle initialisé à la valeur de l'expression arithmétique)



Exemple :

La boucle suivante :

```

* VO1 = 0
* DØ 3
PHRASE ØVO1
* VO1 = ØVO1+1
* ENDØ
  
```

conduit à la génération de texte objet :

```

PHRASE 0
PHRASE 1
PHRASE 2
  
```

Deux boucles DØ peuvent être imbriquées : la boucle associée à une directive DØ comporte alors elle-même une boucle DØ.

Exemple :

```

* DØ 5
-----
* DØ 4
-----
* ENDØ
-----
* ENDØ
  
```

MACP autorise jusqu'à 10 boucles DØ imbriquées les unes dans les autres.

Dans ce cas les règles suivantes sont adoptées :

- A l'intérieur d'une macro-définition le nombre de DØ doit être égal au nombre de ENDØ.
 - En cours de définition d'une macro le nombre de DØ doit toujours être supérieur ou égal au nombre de ENDØ.
- Ainsi l'enchaînement :

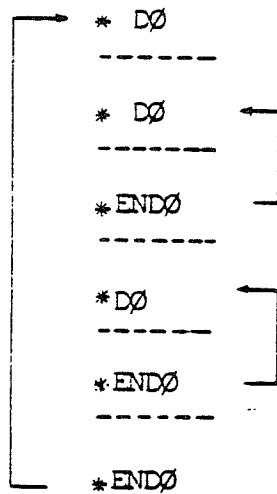
```
* DØ 5
-----
* ENDØ
-----
* ENDØ
-----
* DØ
```

conduit, lorsque MACP rencontre la seconde directive ENDØ, à l'impression d'un message d'erreur.

- L'association DØ - ENDØ est réalisée lorsque le macro-processeur enregistre la macro-définition. Elle est donc indépendante du déroulement de la génération de la macro.

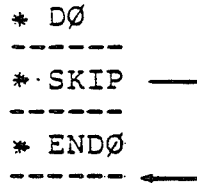
MACP associe à chaque directive ENDØ la dernière directive DØ rencontrée dont l'effet n'a pas encore été annulé (par une directive ENDØ).

Exemple :

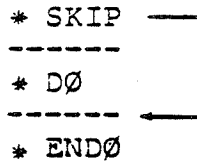


Remarque :

Au moment de la génération d'une macro, il est possible d'effectuer, par une directive IF ou SKIP, un branchement de l'intérieur de la portée d'un DØ vers l'extérieur (abandon de la boucle DØ).

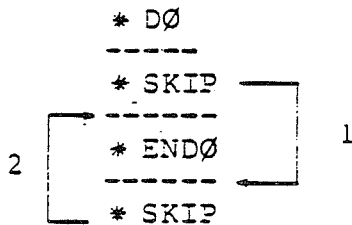


Il est également possible d'effectuer un branchement de l'extérieur de la portée d'un DØ vers l'intérieur :



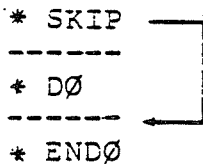
Dans ce cas deux situations peuvent apparaître :

- le branchement à l'intérieur de la portée (2) a été précédé d'un abandon de la boucle DØ (1).



Lorsque la directive ENDØ sera traitée, il y aura renvoi à la directive DØ correspondant (reprise de la boucle DØ).

- le branchement à l'intérieur de la portée n'a pas été précédé d'un abandon de la boucle DØ.



Lorsque la directive ENDØ sera traitée il y aura passage en séquence.

8 - DIRECTIVE KILL

La directive KILL a la syntaxe suivante :

```
(ID)KILL(expression chaîne) (FL)
```

Elle ne peut être rencontrée que dans une macro-définition.

Elle a pour effet :

- d'abandonner la génération de la macro-instruction qui a été lue dans le texte source,
- d'imprimer l'expression chaîne qui joue ainsi le rôle d'un message d'erreur dont le texte est fourni par l'utilisateur,
- de commuter la lecture sur le fichier correction.

Avec un texte source comportant la séquence suivante :

```
* DEFPARAM?  
% VERIF?P1  
-----  
* ENDEF  
* DEFVERIF?  
* IF ?T1 = 2 SKIP 2  
* KILL..TYPE INCORRECT ..  
* ENDEF
```

la macro-instruction :

```
% PARAM'12
```

provoque l'appel de la macro d'en-tête VERIF? dont le but est la vérification du type du paramètre effectif introduit .

Lorsque ce type n'est pas conforme à ce qui a été prévu (dans l'exemple, type non décimal) il y a exécution de la directive KILL, ce qui a pour effet l'impression de :

```
.. TYPE INCORRECT ..
```

ainsi que la demande d'une nouvelle phrase sur le fichier correction.

9 - DIRECTIVE C

La directive C permet la redéfinition des caractères de contrôle qui sont reconnus par le macro-processeur MACP.

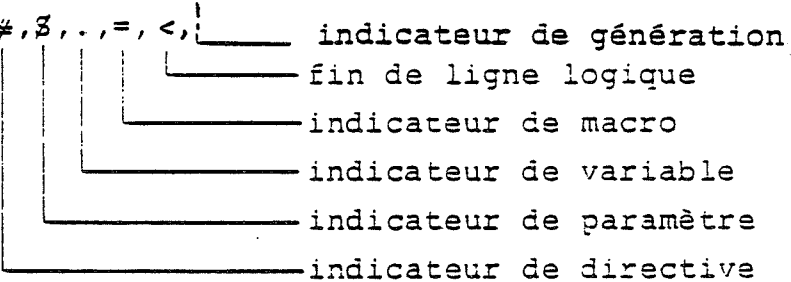
Elle a la syntaxe suivante :

```
(ID)C(ID'),(IP'),(IV'),(IM'),(FL'),(IG') (FL)
```

La zone opérande comporte, séparés par des virgules, les caractères de contrôle utilisés dans la suite du texte source.

Exemple :

```
* C #, $, ., =, <, !
```



- indicateur de génération
- fin de ligne logique
- indicateur de macro
- indicateur de variable
- indicateur de paramètre
- indicateur de directive

Deux des caractères ne peuvent être égaux, ce qui provoquerait l'impression d'un message d'erreur.

Lorsque certains des caractères de contrôle sont à redéfinir, il n'est pas indispensable de spécifier ceux qui demeurent inchangés.

En effet, le macro-processeur interprète toute absence de caractères de contrôle entre deux virgules comme une demande de non redéfinition.

Ainsi la directive :

```
* C #, , , , ;
```

redéfinit seulement l'indicateur de directive (#) et la fin de ligne logique (;).

De même, lorsque le macro-processeur rencontre la fin de ligne (FL) tout caractère non redéfini à ce moment est conservé.

Exemples :

```
* C, , .      redéfinition de l'indicateur de variable  
* C #        redéfinition de l'indicateur de directive
```

Remarques :

- Lorsqu'un texte source ne comporte aucune directive C, les caractères de contrôle introduits en 2.2 sont pris par défaut.
- Toute directive C est prise en compte immédiatement par le macro-processeur, quelque soit le contexte où elle apparaît (en particulier dans une macro-définition). Il est donc impossible de différer son traitement jusqu'au moment de l'appel d'une macro.

10 - DIRECTIVE EØT

La syntaxe de la directive EØT est la suivante :

`(ID)EØT(FL)`

Lorsque le support utilisé pour le texte source est le ruban de papier, il est intéressant de pouvoir le découper en plusieurs bandes. Les corrections sont ainsi facilitées.

Chaque bande intermédiaire doit se terminer par la directive EØT qui indique une fin de bande physique mais non la fin logique du programme. La dernière se termine par la directive END.

Exemple :

```
Bande 1  |-----  
          |-----  
          | *EØT  
          |-----  
Bande 2  |-----  
          |-----  
          | *EØT  
          |-----  
Bande 3  |-----  
          |-----  
          | *END
```

La directive EØT permet également d'avoir macro-définitions et macro-instructions sur des supports physiques différents : cartes, rubans perforés, fichiers disque, ...

Lorsque le macro-processeur rencontre une telle directive, il redonne le contrôle au superviseur.

L'utilisateur peut alors monter la suite du programme source sur l'organe de lecture, réaliser éventuellement une nouvelle association "unité symbolique-unité fonctionnelle" ou "unité symbolique-fichier" avant de demander la poursuite du traitement.

Remarques :

- Lorsque la directive EØT figure dans une séquence du texte source ignorée à la suite d'une directive SKIP elle n'est pas reconnue par MACP.
- La directive EØT ne peut être rencontrée à l'intérieur d'une macro-définition.
- Il est donc interdit d'avoir le début et la fin d'une même définition sur des supports physiques différents.

11 - DIRECTIVE END

La syntaxe de la directive END est la suivante :

`(ID)END(FL)`

Elle indique la fin logique et physique du texte source.

Remarques :

- Lorsque la directive END figure dans une séquence du texte source ignorée à la suite d'une directive SKIP, elle n'est pas reconnue par MACP.
- La directive
 - * SKIP 0figurant dans le texte source hors d'une définition de macro a le même effet que END.
- La directive END ne peut être rencontrée à l'intérieur d'une macro-définition.
Il est donc interdit d'avoir le début et la fin d'une même définition sur des supports physiques différents.

12 - MARQUEUR D'ETIQUETTES

Il se peut qu'une macro utilise des étiquettes pour définir, par exemple, une mémoire de manœuvre locale.

Si aucune précaution n'est prise, la répétition de ces étiquettes sera interprétée par l'assembleur ou un compilateur comme des définitions multiples qui entraîneront un diagnostic d'erreur.

Un moyen pour éviter cet inconvénient consiste à déclarer les étiquettes comme paramètres de la macro et à les modifier lors de chaque appel.

Mais on peut aussi utiliser le marqueur automatique d'étiquettes symbolisé par :

`(I# 0)`

qui est remplacé, lors de l'expansion de la macro, par un nombre (de 3 chiffres au plus).

Le nombre garde la même valeur à l'intérieur d'une macro-instruction, mais est incrémenté de 1 lors de chaque appel de macro.

Exemple :

Avec les macro-définitions :

```
* DEF?=?  
DEB?O:PSR A  
?P2  
STA ?P1  
END?O:PLR A  
* ENDEF  
Macro 1  
  
* DEF?+?  
ADD?O:LA ?P1  
AD ?P2  
* ENDEF  
Macro 2
```

la macro-instruction :

```
MEM1=MEM2+MEM3
```

génère le texte objet :

```
DEB1:PSR A      ] Macro 1 (marqueur = 1)
                ]
ADD2:LA MEM2    ] Macro 2 (marqueur = 2)
AD MEM3         ]
                ]
STA MEM1        ] Macro 1 (marqueur = 1)
END1:PLR A      ]
```

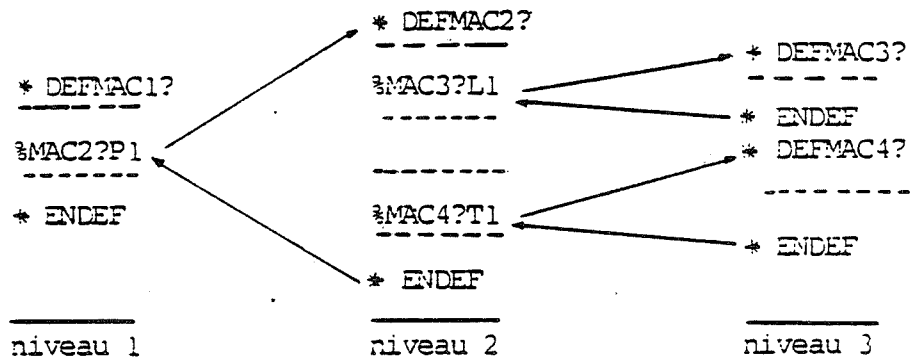
Au prochain appel de macro, le marqueur d'étiquettes sera incrémenté et vaudra 3.

13 - AIDES A LA MISE AU POINT : DIRECTIVES TRACE ON ET TRACE OFF

Dans une macro-définition, il arrive fréquemment que certaines lignes soient des macro-instructions.

On peut alors considérer plusieurs niveaux dans les appels de macro dans une phase de génération.

Avec la structure suivante :



la macro-instruction :

```
%MAC1PARAM
```

est un appel de MAC1 au niveau 1, de MAC2 au niveau 2, de MAC3 et MAC4 au niveau 3.

Pour résoudre les problèmes de sauvegarde de paramètres, MACP met en oeuvre une pile de génération permettant d'aller jusqu'au niveau d'imbrication 10.

Les deux directives TRACE ON et TRACE OFF fournissent une aide à la mise au point des programmes comportant de telles imbrications en fournissant, au cours de la génération, un certain nombre d'informations.

Elles ont les syntaxes respectives suivantes :

```
(ID) TRACE  $\sqsubset$  ON (FL)
```

```
(ID) TRACE  $\sqsubset$  OFF (FL)
```

La directive TRACE ON met le module de trace à l'état "actif".

- A chaque appel de macro, il y a impression d'un message comportant :
 - n° ligne source : n° d'imbrication de la macro macro-ins-
 - truction d'appel
- A chaque sortie de macro, il y a impression d'un message comportant :
 - n° ligne source : n° d'imbrication au point de retour

Avec l'exemple plus haut, la succession des messages est la suivante :

```
40: 1  MAC1PARAM
40: 2  MAC2PARAM
40: 3  MAC35
40: 2
40: 3  MAC44
40: 2
40: 1
40: 0
```

La directive TRACE OFF met le module de trace à l'état "passif".

Aucun message n'est alors imprimé lors des appels et ces sorties des macro.

TEXTE SOURCE (listé par l'éditeur EDITEX)a) Macro-définitions :

```
1 *C,.,.,;
2
3 *DEF?=? ;AFFECTATION
4 *V00 =0
5 ?P2
6 STA ?P1
7 *V02 EQ ?P1
8 *ENDEF
9
10 *DEFADD(?) ;ADDITION
11 *V01 EQ AD
12 ?P1, ;APPEL MACRO "??"
13 *ENDEF
14
15 *DEFSUB(?) ;SOUSTRACTION
16 *V01 EQ SB
17 ?P1, ;APPEL MACRO "??"
18 *ENDEF
19
20 *DEF?,,? ;RECHERCHE OPERANDES ET CALCUL
21 *IF @V00 NE 0 SKIP 4
22 LA ?P1
23 *V00 =1
24 *SKIP 2
25 @V01 ?P1 ;OPERATION
26 *IF ?L2=0 SKIP 0
27 ?P2 ;PAS TERMINE
28 *ENDEF
29
30 *DEFIF ?<? THEN ?
31 CP(?P1-?P2-?P3-JGE)
32 *ENDEF
33
34 *DEFIF ?=? THEN ?
35 CP(?P1-?P2-?P3-JNE)
36 *ENDEF
37
38 *DEFIF ?>? THEN ?
39 CP(?P1-?P2-?P3-JLE)
40 *ENDEF
41
42 *DEFCP(?-?-?-?)
43 *IF @V02 EQ ?P1 SKIP 2
44 LA ?P1
45 CP ?P2 ;COMPARAISON
46 ?P4 ET?0 ;RUPTURE SI FAUX
47 ?P3
48 ET?0 EQU S
49 *FNDEF
```

b) Macro-instructions :

```
50  
51 *TRACE ON ; DEMANDE DE TRACE  
52  
53 A=ADD(B, C, D)  
54 IF A<A1 THEN IF B>B1 THEN G=SUB(H, I)
```

TEXTE OBJET

```
LA B  
AD C  
AD D  
STA A  
CP A1  
JGE ET6  
LA B  
CP B1  
JLE ET8  
LA H  
SB I  
STA G  
ET8: EQU $  
ET6: EQU $
```

TRACE

```
53: 1 A=ADD(B, C, D)  
53: 2 ADD(B, C, D)  
53: 3 B, C, D,  
53: 4 C, D,  
53: 5 D,  
53: 4  
53: 3  
53: 2  
53: 1  
53: 0  
54: 1 IF A<A1 THEN IF B>B1 THEN G=SUB(H, I)  
54: 2 CP(A-A1-IF B>B1 THEN G=SUB(H, I)-JLE)  
54: 3 IF B>B1 THEN G=SUB(H, I)  
54: 4 CP(B-B1-G=SUB(H, I)-JLE)  
54: 5 G=SUB(H, I)  
54: 6 SUB(H, I)  
54: 7 H, I,  
54: 8 I,  
54: 7  
54: 6  
54: 5  
54: 4  
54: 3  
54: 2  
54: 1  
54: 0
```

Remarque :

53: 1	}	Appels de	=?
54: 5			
53: 2		Appel de	ADD(?)
53: 6		Appel de	SUB(?)
53: 3	}	Appels de	?,?
53: 4			
53: 5			
54: 7			
54: 8			
54: 1		Appel de IF	?<? THEN ?
54: 3		Appel de IF	?>? THEN ?
54: 2	}	Appels de	CP(?-?-?-?)
54: 4			

15 - UTILISATION DU MACRO-PROCESSEUR

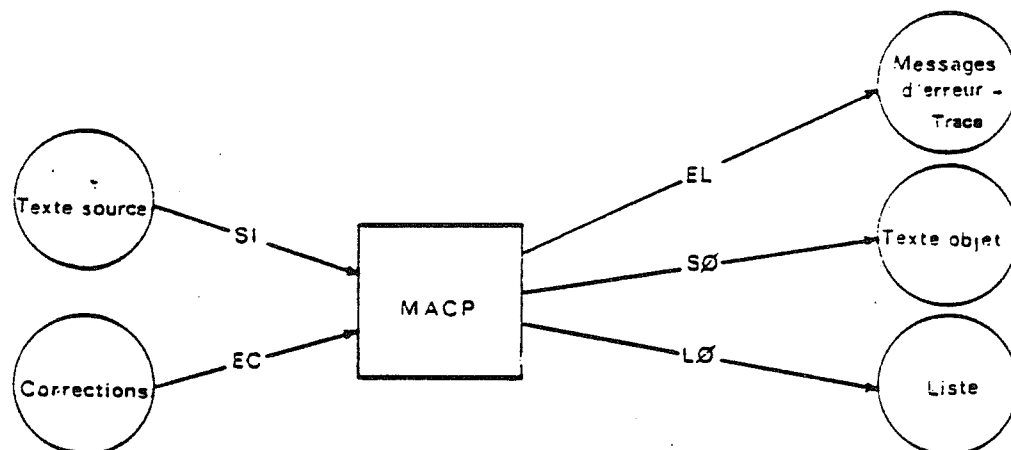
15.1 - INTRODUCTION

MACP est un processeur géré par les systèmes d'exploitation BOS-A, BOS-B et BOS-C. MACP-D est géré par BOD-D et BACKM.

Ils utilisent un certain nombre d'unités symboliques pour leurs demandes d'opérations d'entrée-sortie.

Ce sont les suivants :

SI	Entrée du texte source
SØ	Sortie du texte objet
LØ	Sortie de la liste du texte objet
EL	Sortie des messages d'erreur Sortie de la trace
EC	Entrée des corrections symboliques



Avant d'activer le macro-processeur, l'utilisateur peut associer à chaque unité symbolique une unité fonctionnelle dont il dispose dans sa configuration.

Les commandes d'association sont décrites dans le manuel de référence du système d'exploitation sous lequel s'exécute le macro-processeur.

15.2 - ACTIVATION DU MACRO-PROCESSEUR

Le macro-processeur MACP possède trois points d'entrée.

A chacun d'eux correspond une commande qui est reconnue par le superviseur.

15.2.1 - COMMANDE IMAC

La commande IMAC donne le contrôle au macro-processeur qui, après s'être réinitialisé (annulation de toutes les macro-définitions, etc...), démarre le traitement à partir de l'unité symbolique SI.

Le texte objet est produit sur l'unité symbolique SØ, la liste qui lui est associée sur l'unité symbolique LØ et la trace (optionnelle) sur l'unité symbolique LL.

Lorsque le macro-processeur trouve l'une des deux directives EØT et END, il rend le contrôle au superviseur.

Exemple :

* SI	CR	Affectation à SI du lecteur de cartes
* SØ	HP	Affectation à SØ du perforateur rapide
* LØ	ZE	Suppression de la liste du texte objet .
* IMAC		Activation de MAC avec demande de réinitialisation
*		Retour au superviseur en fin de traitement

Les caractères soulignés sont émis par le système.

15.2.2 - COMMANDE CMAC

Lorsque le support utilisé est le ruban papier, la directive EØT permet d'avoir un texte source en plusieurs bandes.

De même, cette directive permet d'avoir macro-définitions et macro-instructions sur des supports physiques différents.

En cours de traitement, lorsque EØT est reconnue par MACP, le contrôle est donné au superviseur .

L'utilisateur peut alors placer la suite du texte source sur l'unité de lecture appropriée avant de demander la poursuite du traitement.

Ceci est réalisé par l'intermédiaire de la commande CMAC.

Exemple :

```
* SI CR
* SØ HP      Entrée symbolique réalisée
* LØ ZE      sur le lecteur de cartes
* IMAC      Initialisation de MAC
```


```
* SI HR      Entrée symbolique réalisée
              sur le lecteur rapide de
              bandes
              Continuation du traitement
```

*

15.2.3 - COMMANDE_LMAC

La commande LMAC permet d'obtenir, sur l'unité symbolique LØ, la liste de toutes les en-têtes des macros.

L'ordre de leur impression est celui établi par le macro-processeur en fonction des règles introduites en 3.5.

Ainsi, avec les macro-définitions du chapitre 14, la liste suivante est obtenue :

```
ADD (?)
SUB (?)
IF ?<? THEN ?
IF ?=? THEN ?
IF ?>? THEN ?
CP (?-?-?-?-)
??
?,?
```

15.3 - CORRECTION DES ERREURS

15.3.1 - MESSAGES D'ERREUR

Lorsqu'au cours du traitement le macro-processeur détecte une erreur, il imprime un message sur l'unité symbolique EL et commute l'entrée symbolique de SI sur EC.

La composition du message est différente suivant que le macro-processeur se trouve dans une phase d'expansion ou non.

Dans le premier cas, la dernière phrase lue dans le texte source est une macro-instruction.

L'erreur est détectée alors que MACP est en cours d'expansion de la ligne de rang r dans une macro dont le numéro d'imbrication est n

La génération est abandonnée quelque soit le niveau d'imbrication en cours.

Le message comporte alors :

- numéro de la ligne source
- niveau d'imbrication, soit n
- numéro de l'erreur
- numéro de la ligne dans la macro, soit r
- éventuellement, l'instruction constituant l'appel de la macro.

Cette dernière indication ne peut être fournie que si la macro de niveau n n'a fait appel à aucune macro dont le niveau serait alors n+1.

Supposons le texte source suivant :

```
* DEFUN?  
* VOL = +?P1  
3 DEUX?T1  
* VO2 = 4/?P1  
* ENDEF
```

La macro-instruction (ligne 50 du texte source, par exemple) :

```
UN+12
```

Conduit au message d'erreur :

```
50 : 1 UN+12  
ERM 02  
LIGNE 1  
(expression ++12 interdite)
```

Par contre, la macro-instruction :

```
UNO
```

conduit au message d'erreur :

```
SO : 1
ERM 02
LIGNE 3
(division par 0 interdite)
puis que la macro d'en-tête UN a, au moment de
la détection de l'erreur, fait appel à la macro
d'en-tête DEUX.
```

Lorsque la macro-processeur ne se trouve pas dans une phase d'expansion, le message d'erreur comporte seulement :

- numéro de la ligne source
- ligne source
- numéro de l'erreur

Avec la ligne suivante dans le texte source :

```
* SKIP -4
```

MACP peut imprimer le message :

```
51: * SKIP -4
ERM 03
(saut négatif interdit)
```

15.3.2 - CORRECTIONS

Après l'impression du message d'erreur, il y a commutation sur l'unité symbolique EC.

La correction peut être constituée d'une ou plusieurs lignes.

Elle se termine par une ligne vide qui déclenche la poursuite du traitement sur l'unité symbolique SI.

Lorsqu'une erreur a été détectée dans une phase d'expansion il y a abandon de la génération en cours.

Il est alors possible de modifier la macro-définition incriminée et de réintroduire la macro-instruction d'appel.

15.4 - INTERRUPTION D'UN TRAITEMENT EN COURS

L'opérateur dispose sur le périphérique de dialogue (télé-imprimeur ou console de visualisation) d'un bouton d'appel du superviseur ("Break") qui peut être utilisé pour interrompre un traitement en cours.

Un appel unique est pris en compte immédiatement mais le contrôle n'est rendu au système qu'à la fin de la fonction en cours.

Il est ainsi impossible d'arrêter une expansion de macro en cours.

Bien entendu, le contrôle est rendu au superviseur après quatre appels consécutifs.

15.5 - CHARGEMENT DU MACRO-PROCESSEUR MACP

Le macro-processeur MACP s'exécute sous le contrôle des superviseurs BOS-A, BOS-B et BOS-C.

Destiné à l'un des deux premiers systèmes il se présente sous la forme d'une bande binaire translatable.

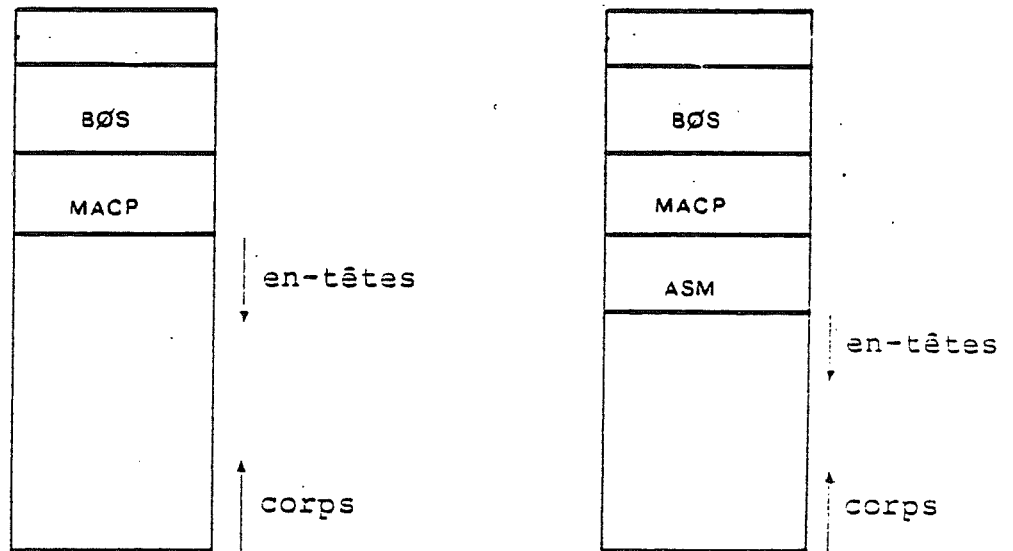
La suite de ce paragraphe ne concerne donc que BOS-A et BOS-B.

La séquence de programme qui est activée lors du lancement automatique par le chargeur se termine par une requête SVC CA=0 transmettant au système les commandes introduites en 15.2.

Il est donc nécessaire, au moment du chargement de MACP, que le superviseur soit déjà en mémoire.

D'autre part, la zone allouée aux tables d'en têtes et de corps de macro est déterminée par le contenu de la mémoire FIRSTM ('B) au moment de l'activation de MACP par la commande IMAC.

Cette mémoire, gérée par le chargeur translateur, pointe sur la première mémoire libre, ce qui interdit aux tables de venir recouvrir un processeur chargé postérieurement à MACP.

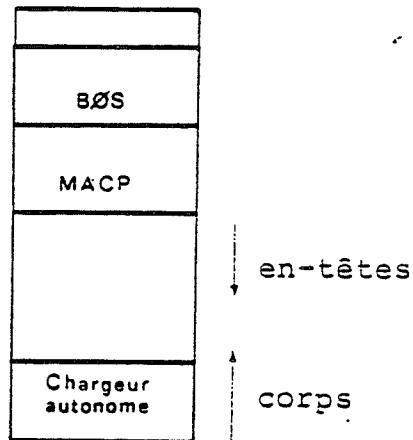


Remarque :

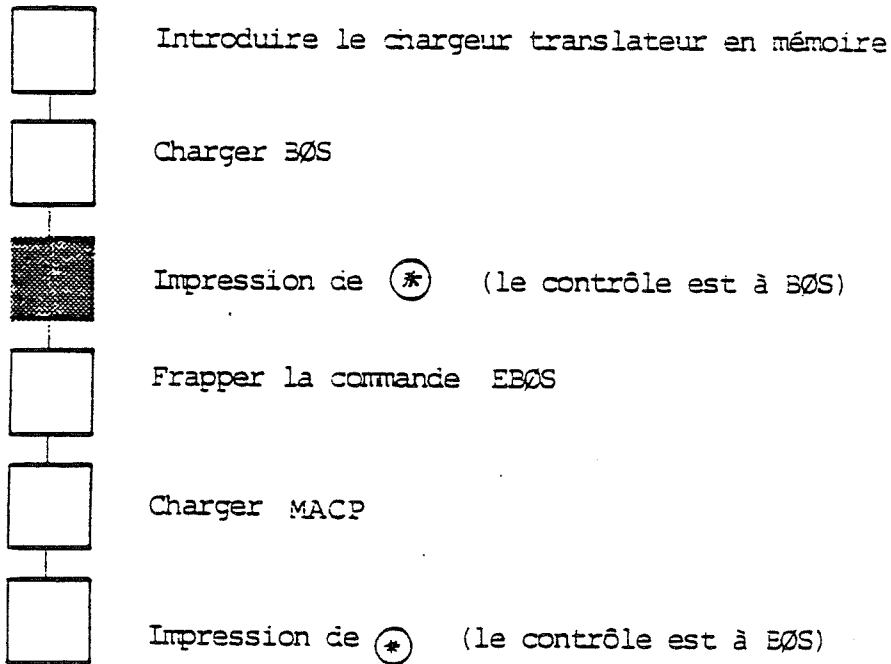
Les implantations des paragraphes suivants ne sont données qu'à titre d'exemples. Pour plus de précisions, il est recommandé de se reporter au manuel de référence de BOS-B.

15.5.1 - UTILISATION DU CHARGEUR AUTONOME

Une implantation peut être la suivante :



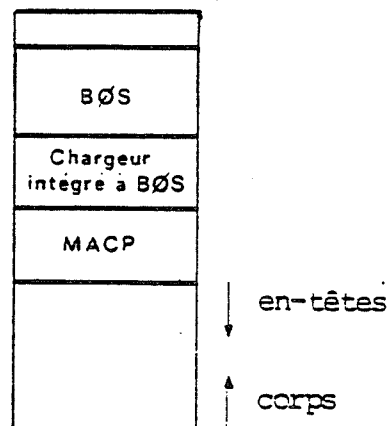
En correspondance, la suite des opérations à effectuer est



Le système est alors prêt à accepter les commandes du macro-processeur.


15.5.2 - UTILISATION DU CHARGEUR INTEGRE

Lorsque le superviseur ainsi que le chargeur translat qui lui est intégré sont en mémoire, une implantation peut être la suivante :



La suite des opérations à effectuer devient :



Impression de  (le contrôle est à BØS)




Frapper la commande IRLD (le contrôle est donné au chargeur)



Charger MACP



Impression de  (le contrôle est à BØS)

Bull 15.6 - UTILISATION DU MACRO-PROCESSEUR MACP-D

MACP-D est un processeur fonctionnant en mode maître ou esclave sous tous les systèmes d'exploitation disque du SOLAR16. Il nécessite une partition mémoire de 10K mots.

15.6.1 - Principe

Tout comme le processeur EDIT16, MACP-D utilise pour stocker le corps des macro définitions un fichier espace virtuel sur disque. Cet espace virtuel est ouvert à la commande IMAC et fermé sur la directive XEND.

Pour pouvoir garder les macros contenues dans l'espace, il est donc nécessaire de revenir sous système par la directive XEOT, puis de réutiliser l'espace virtuel par la commande CMAC.

La taille disque nécessaire est fonction de la taille de l'espace virtuel.

En standard, MACP-D peut gérer un fichier de 5460 lignes ce qui correspond à un espace virtuel de l'ordre de 260 Kmots, implanté sur l'unité fonctionnel D2.

Le fichier espace virtuel (qui est le même que celui d'EDIT16) se nomme << VI >>, catalogue AB pour un système mono utilisateur ou catalogue = Public Word (PUBW = N°) pour un système multiservice.

La taille du fichier espace virtuel et l'unité fonctionnelle disque, qui le supporte peuvent être modifiées lors de l'intégration de MACP-D.

Les 2 premiers mots du processeur contiennent en effet respectivement :

- le numéro d'unité fonctionnelle disque support de l'espace virtuel standard.
- un indicateur caractérisant la taille de l'espace virtuel standard

0	pour 1364 lignes
1	pour 2729 lignes
2	pour 5460 lignes
3	pour 10921 lignes

15.6.2 - Modification de la FU support de l'espace virtuel

Emettre les commandes :

X CALL FUP7

X PATCH,MACP-:S,D2

X ADRE,'O,, ROOT

X VISU

'OOOE

(en standard D2)

X STOR, 'numéro de FU (par exemple STOR, '14 pour D8)

Les caractères soulignés sont émis par le système.

15.6.3 - Modification de la taille de l'espace virtuel standard

Emettre les commandes :

⊗ CALL FUP7

⊗ PATCH,MACP-:S,D2

⊗ ADRE, '1 , , ROOT

'0002 (en standard 5 460 lignes)

⊗ STOR, 'i (par exemple STOR, '1 pour 2 729 lignes)

15.6.4 - Intégration sous les systèmes TSM et TSF

MACP-D est intégré sous les systèmes TSM et TSF en tant qu'utilitaire. Les fichiers espaces virtuels devront obligatoirement se trouver dans des unités fonctionnelles disque banales.

Il est conseillé par ailleurs de ne pas utiliser l'unité fonctionnelle disque D2 comme support des espaces virtuels (encombrement; D2 réservé aux systèmes).

Il convient donc de modifier le numéro de la FU support des espaces virtuels.

15.6.5 - Contraintes

La taille des granules de l'unité fonctionnelle disque support d'un espace virtuel doit respecter le tableau suivant :

Valeur de i	Nombre de lignes	Nombre d'articles de 1 K mots dans l'espace virtuel	Taille des granules minimum
0	1 364	65	8 secteurs
1	2 729	129	16 secteurs
2	5 460	257	32 secteurs
3	10 921	513	48 secteurs

Bull  5.6.6 - Erreur avec le fichier espace virtuel

ERV 1 : le fichier espace virtuel n'a pas été ouvert

ERV 2 : l'espace est saturé

ERV 3 : erreur FMS (impression compte rendu du FMS)

ERV 4 : erreur lors de l'ouverture du fichier espace virtuel

16 - SYNOPTIQUE

CARACTERES DE CONTROLE

Notation	Rôle	Caractères standards
(ID)	Indicateur de directive	*
(IP)	Indicateur de paramètre	?
(IV)	Indicateur de variable	∂
(IM)	Indicateur de macro-instruction	‡
(FL)	Fin de ligne logique	Retour chariot
(IG)	Indicateur de génération	\

REFERENCES AUX PARAMETRES

avec :

(IP) fn

- f = type de l'opération à effectuer
- n = numéro du paramètre

Opérations permises :

- remplacement par le paramètre effectif (f = P)
- remplacement par la longueur du paramètre effectif (f = L)
- remplacement par le type du paramètre effectif (f = T)

Types reconnus par MACP :

- nombre hexadécimal (type = 1)
- nombre décimal (type = 2)
- chaîne de caractères (type = 3)
- symbole (type = 4)
- autre (type = 5)

REFERENCE A UNE VARIABLE(IV) V n₁ n₂n₁ et n₂ étant deux chiffres composant le numéro de la variableMODIFICATION D'UN PARAMETRE

(expression chaîne) (IP) An (FL)

n étant le numéro du paramètre.

DIRECTIVES

(ID) DEF (en-tête) (FL)
(ID) ENDEF (FL)
n ₁ et n ₂ étant deux chiffres composant le numéro de la variable :
(ID) V n ₁ n ₂ ⌊ = (expression arithmétique) (FL)
(ID) V n ₁ n ₂ ⌊ EQ ⌊ (expression chaîne) (FL)
(ID) SKIP ⌊ (expression arithmétique) (FL)
(ID) IF ⌊ (relation) ⌊ SKIP ⌊ (expression arithmétique) (FL)
Une relation étant :
(expression arithmétique) { < = > } (expression arithmétique)
ou :
(expression chaîne) ⌊ { EQ NE } ⌊ (expression chaîne)
(ID) DØ ⌊ (expression arithmétique position nulle) (FL)
(ID) ENDØ (FL)
(ID) KILL (expression chaîne) (FL)
(ID) C (ID'), (IP'), (IV'), (IM'), (FL'), (IG') (FL)
(ID) EØT (FL)
(ID) END (FL)
(ID) TRACE ⌊ ØN (FL)
(ID) TRACE ⌊ ØFF (FL)

⌊ est utilisé pour indiquer un espacement obligatoire (un ou plusieurs espaces)..

MARQUEUR D'ETIQUETTES

(IP) O

ANNEXE : LISTE DES NUMEROS D'ERREUR

- 00 Erreur de parité
- 01 Erreur d'écriture
 - . Erreur dans l'écriture d'une directive
 - . Numéro de variable supérieur à 49
 - . Deux caractères de contrôle identiques dans une directive C
 -
- 02 Expression arithmétique incorrecte
 - . Opérande non décimal
 - . Enchaînement d'opérateurs interdit
 - . Division par zéro
 - . Valeur d'un opérande non représentable sur 16 bits
 - . Valeur de l'expression non représentable sur 16 bits
 -
- 03 Contexte incorrect
 - . ENDEF, DØ, ENDØ, KILL hors d'une macro-définition
 - . Saut négatif à la suite d'une directive SKIP ou IF hors d'une macro-définition
 - . END, EØT dans une macro-définition
- 04 Élément non défini
 - . Variable non définie
 - . Paramètre non défini
- 05 Macro-instruction spécifiée par (IM) incorrecte
- 06 Définitions imbriquées
- 07 Nombre de DØ différent du nombre de ENDØ dans une macro-définition
- 08 Directive ENDØ interdite avant la directive DØ correspondante
- 09 Saut à l'extérieur de la macro
- 10 Nombre de boucles DØ imbriquées supérieur à 10
- 11 Niveau d'imbrication d'une macro supérieur à 10
- 12 Saturation des tables du macro-processeur
- 13 Erreur avec fichier espace virtuel