Réseaux
et systèmes
d'information

Bull

MANUEL DE RÉFÉRENCE

SOLAR

DRIP16

Utilitaire de mise au point temps réel

Bull

1 164 081 00 036

Réseaux
et systèmes
d'information

Bull

SEMS

DRIP 16

“Debugging Real Time Interactive Program”

MANUEL DE REFERENCE

OCTOBRE 1976

1 164 081 00/- 36 02

AVANT PROPOS

Cette notice suppose connus les Manuels de référence des différents systèmes d'exploitation de la série SOLAR 16.

Elle constitue à la fois un document de référence et un guide d'utilisation des deux versions du produit DRIP16 (Debugging Real time Interactive Program)

- DRIP16 - A, version minimum utilisable en monoprogrammation et multiprogrammation avec seulement une imprimante.

- DRIP16 - B, version complète utilisable en monoprogrammation et multiprogrammation avec un disque et une imprimante et capable de réaliser des espionnages système.

Le lecteur trouvera donc :

- la description des différents services offerts par DRIP16 - B et les restrictions de DRIP16 - A

- un ensemble d'informations concernant l'utilisation pratique du produit : génération, mise en oeuvre et conseils d'utilisation

AVERTISSEMENT

NOTATIONS UTILISEES

[E] les crochets signifient que la présence de E est facultative

Exemple : ONTR [, Pi, Pj . . .]
[MULTI]

{ E1 / E2 } les accolades marquent un choix possible

Exemple : SNAP [d] ({ Variable / RA } ---)

SOMMAIRE

Pages

ANNEXE

SYNOPTIQUE

1 - INTRODUCTION

1.1 - PRESENTATION

DRIP16 est un logiciel d'aide à la mise au point des programmes PL16 ou assembleur exploités sous niveau software et sous les systèmes standard de la série SOLAR 16.

C'est un outil de surveillance et d'extraction efficace aussi bien dans un contexte de monoprogrammation (ou batch) que dans un contexte de multiprogrammation de type temps réel. De plus, hors contexte batch. DRIP16 est interactif.

Dans sa version complète, DRIP16 est prévu pour travailler sur une configuration possédant un disque et une imprimante. II est utilisable quelle que soit la configuration machine.

C'est un processeur indépendant qui doit être intégré sous le système d'exploitation choisi avant le lancement du programme utilisateur contenant les appels au module DRIP16.

II est activé :

- soit par le programme utilisateur lui-même s'il comporte les directives et instructions appropriées
- soit par des commandes système données à la console et offre ainsi des moyens d'analyse aux deux niveaux de la programmation et des essais.

DRIP16 est de conception modulaire de manière à s'adapter aux besoins réels de l'utilisateur en lui assurant un encombrement minimal.

1.2 - LA MISE AU POINT EN TEMPS REEL

La mise au point et l'espionnage d'une application temps réel nécessitent des outils capables de contrôler :

- des paramètres classiques, tout comme les moyens ordinaires de mise au point : ce sont des données "locales", des chemins...
- mais aussi des paramètres aléatoires (événements, informations chaînées...) au caractère :
  * imprévisible, donc difficiles à imaginer ou à reproduire car ils sont dus à des événements externes. Ainsi, en cours de mise au point, des simulations de quelques situations du système ne suffiront pas à valider son bon fonctionnement, de même il sera très difficile de recréer le contexte d'une erreur survenue en cours d'exploitation.
  * fugitif car *leur* période de validité est plus ou moins longue. Ainsi un "dump" à posteriori de la mémoire centrale s'avère en général insuffisant pour trouver les causes d'un incident ou d'une dégradation de l'application.

Donc, un outil de mise au point en temps réel devra cohabiter avec le système à valider afin :

- de contrôler et de surveiller le système dans des situations réelles de fonctionnement
- de réaliser les extractions de paramètres au moment précis où ceux-ci ont un intérêt immédiat et vital pour l'ensemble du système.

Un tel outil servira, ainsi, aussi bien au diagnostic de panne qu'au contrôle du bon fonctionnement dynamique de l'application.

Cet outil devra perturber au minimum l'application, il faudra donc :

- de la part de l'utilisateur. un choix judicieux des points de test vitaux pour le système
- de la part de l'outil lui-même
  * l'utilisation de périphériques rapides pour les extractions
  * un dialogue puissant pour la sélection des extradions. celles-ci n'ayant pas toutes le même degré de priorité (données locales ou système)

1.3 - CARACTERISTIQUES DE DRIP16

Par son mode de fonctionnement et les services qu'il propose, DRIP16 est un outil de mise au point puissant pour réaliser des vidages mémoire. II permet en particulier :

. de connaître le "chemin suivi" par un programme
. l'extraction des données "locales" d'un programme
. l'extraction de données "système" propres à l'environnement du programme.

Ces informations sont :

. soit éditées sous forme claire sur un terminal d'édition
. soit stockées sous forme brute sur fichier disque pour être exploitées et mises en forme en différé par un processeur spécialisé.
. soit éditées et stockées à la fois.

DRIP16 est intégré au système pendant l'exploitation des programmes à tester afin de contrôler, au moment opportun, des situations réelles de fonctionnement.

Sa facilité d'emploi se concrétise par :

- une grande souplesse d'intégration au système d'exploitation utilisé
- sa transportabilité ou indépendance par rapport au langage utilisé
- un interface utilisateur simple aux niveaux interne (programmation) et externe (opérateur)
  . directives et instructions au niveau programmation
  . commandes au niveau mise en oeuvre du programme
- une grande souplesse dans la sélection des tests à effectuer par l'usage de paramètres de contrôle externes (commandes)
- une grande souplesse d'évolution, d'intégration et d'élimination au niveau du programme. On peut vider le programme de ses points de tests en une seule compilation
- sa "transparence", c'est-à-dire une perturbation minimale sur le programme à tester (utilisation du disque pour stocker les extractions)
- sa modularité qui permet de l'adapter aux besoins de l'utilisateur.

De plus, alors que les moyens classiques de mise au point s'avèrent inutilisables dans certaines phases de développement d'un produit logiciel, DRIP16 offre des facilités de mise au point tout au long de la vie d'un programme à l'aide d'une méthodologie de travail unique :

. phase initiale de mise au point indépendante
. phase d'intégration du programme à l'application finale et évaluation de celle-ci
. phase opérationnelle ou d'exploitation
. maintenance

2 - PRINCIPE DE FONCTIONNEMENT

Dans un premier temps, le programmeur définit par un jeu de directives et d'instructions les extractions locales ou "système" qu'il désire effectuer ainsi que le DEGRE d'importance qu'il leur associe. La génération de code correspondante, qui doit réaliser les appels au module DRIP16 est contrôlée par des "clés de compilation". On pourra ainsi facilement vider le programme de ses points de tests.

Au moment des essais le programmeur dispose donc de séquences de test implantées dans son programme ; mais ces séquences ne provoquent pas systématiquement des extractions car le programmeur dispose de commandes pour valider ou invalider les séquences afin de moduler le flot des extractions selon leur DEGRE, leur NATURE ou selon la priorité de l'appelant. Suivant la phase d'avancement de ses essais, l'utilisateur a ainsi la possibilité de sélecter ses points de tests sans avoir à recompiler son programme.

Ces commandes permettent en outre de choisir le périphérique associé aux extractions :

- un terminal d'édition : imprimante rapide
- le disque (dans ce cas les extractions seront exploitées en différé par le processeur de dépouillement spécialise PRODEP)
- ou les deux

Nous proposons en annexe une méthodologie d'emploi de DRIP16.

3 - LA PROGRAMMATION DES APPELS A DRIP16

3.1 - LA NOTION DE DEGRE

Cette notion a été introduite dans un but de sélection dynamique des appels au module DRIP16 en phase d'exploitation de l'application et dans un but de sélection du support de sortie des informations. A chaque appel se trouve associé son degré représenté par un nombre. Au moment des essais, un appel sera

. soit ignoré
. soit édité sur imprimante
. soit stocké sur disque
. soit édité sur imprimante et stocké sur disque selon la valeur de son degré.

Au niveau de l'écriture du programme source le degré est spécifié par une directive ou comme paramètre des instructions spécifiques des appels au module DRIP16. Pour tout appel, le compilateur génère implicitement le degré et lui affecte la valeur spécifiée dans la dernière directive ou spécifiée dans l'instruction d'appel.

3.2 - LES APPELS A DRIP16 EN PL16

35.1 - Les directives

```
* [X] ! DEGRE [d]   (1)
```

Cette directive définit le degré courant d associé aux appels ultérieurs générés par le compilateur. Elle permet à l'usager de découper son programme en zones géographiques consécutives et de leur associer un "degré de mise au point".

Pour tout appel au module DRIP16, le compilateur génère implicitement la valeur d du degré de la zone courante (degré de la dernière directive DEGRE).

En l'absence de cette directive le degré courant est pris égal à 1.

On doit avoir : $1 \leq d \leq 255$

```
* [X] ! TRACE [n]
```

Cette directive permet de rendre effective la génération des appels implicites au module DRIP16

. passage par label
. Entrée et Sortie de procédure

Elle porte sur toutes les définitions de label ou de procédure qui suivent.
n est le degré associé à ces appels

Par défaut de n, le compilateur prend le degré courant défini par la dernière directive DEGRE

```
* [X]  ! NO TRACE
```

Cette directive provoque l'arrêt de la génération des appels implicites.

L'utilisateur doit prévoir en 1ère colonne une clef de compilation X afin de pouvoir facilement vider son programme des appels à DRIP16.

3.2.2 - Les instructions

```
* [X]   SNAP [d] ( { [@ ] nom de variable [+ déplacement ] } , n [, F ] ) ;
                  { RA                                    }
```

Cette instruction permet la visualisation de n mots, dans le format F, à partir de la variable nommée ou à partir de l'adresse contenue dans le registre RA :

Si F = 1 Format hexadécimal

Si F = 2 Format ASCII

Si F = 3 Format décimal

. Par défaut., le format est hexadécimal
. d représente le degré associé à l'appel ; s'il est absent le compilateur prend le degré courant.

Exemples :

```
          ARRAY 40 WORD BUFFER ;
T         CONSTANT ASCII = 2 ;
T         RA := @ TOTO ;              "TOTO ETANT DANS UNE
T         SNAP (RA, 1, ASCII) ;       "SECTION     DUMMY

T         SNAP 240 (@ BUFFER + 20, 20, ASCII) ; " FORMAT
                                                " ASCII

T         SNAP (BUFFER, 40) ;                 "FORMAT HEXA
```

```
* [X]   RSNAP [d] ;
```

permet la visualisation en format hexadécimal des registres A, B, X, Y, C, L, W, K, P, ST, SLØ, SLE et IM (les indicateurs V et C de ST ne sont pas significatifs)

d : idem SNAP

* [X] XPRINT [d] “CHAINE” ;

permet l’impression du message constitué par la chaîne spécifiée

d : idem SNAP

* [X] SILENCE [d] ;

permet d’inhiber dynamiquement les appels implicites

* [X] NO SILENCE [d] ;

permet de valider dynamiquement les appels implicites.

Ces 2 derniers appels ne peuvent être invalidés au niveau des commandes du dialogue car ils sont prévus par le programmeur ; par contre la sortie ou non d’un message signalant le passage sur l’une de ces instructions dépendra du degré d.

* enfin l’instruction :

[X] XSYST [d] (n [, liste de nombres]) ;

permet la visualisation d’informations système propres à RTES-D ou RTES-C.

n = 1 Files du scheduler
2 Etats des événements
3 Phase avancement
4 Occupation des partitions
5 Taux d’attente des partitions
6 Taux d’occupation des zones dynamiques
7 Portion ZDR
8 Etat d’un périphérique
9 Bloc de contrôle d’une tâche
10 Liste des opérations différées et périodiques
11 Liste des tâches en attente d’événement
12 Liste des ressources
13 Liste des fichiers ouverts
14 Dump de zone système

d : idem SNAP

Seuls les appels 1, 8 et 14 sont reconnus sous BOS-D. La "liste de nombres" représente les paramètres de l'extraction système.
Cette liste doit être vide pour les appels 1 à 6 et 10 à 13 par contre pour les autres on doit avoir :

si n = 7 , 2 nombres indiquant respectivement
. le numéro du premier mot de la ZDR à dumper : 1 à N
. le nombre de mots à visualiser

si n = 8 , 1 nombre indiquant le numéro de FU ou de SU

si n = 9 , 1 nombre indiquant le numéro de priorité de la tâche

si n = 14 , 2 nombres indiquant respectivement
. l'adresse absolue de la zone
. le nombre de mots à visualiser

Exemple :

```
T     CONSTANT     ZDR = 7 ;
T     CONSTANT SCHEDU = 1 ;
T     CONSTANT DEBU  = 10 ;
T     CONSTANT NBMOTS = 15 ;

T     XSYST 40 (SCHEDU) ;
T     XSYST 42 (ZDR, DEBU, NBMOTS) ;
```

Remarque :

Les appels explicites formulés par ces instructions sont générés systématiquement par le compilateur PL16.
II faut donc prendre soin de mettre une clé X de compilation optionnelle au début de ces ordres afin de faciliter le vidage du programme de ces instructions.

Exemple récapitulatif :

```
! ! ØN..T
T  !  TRACE  10
T ! DEGRE 200
                                  200 = degré courant des appels explicites
                                   10 = degré courant des appels implicites
T       SNAP     250  (.....);   appel de degré 250
T       SNAP     (.....);        appel de degré 200

     LAB1 : .....                appel de degré 10
T ! NØTRACE
T       RSNAP ;                  appel de degré 200
T       XPRINT 220 “MESSAGE”     appel de degré 220
     L A B 2   :                 pas d’appel
```

En enlevant la 1ère carte, il n’y a plus aucun appel à DRIP16 dans ce programme.

Remarque :

On peut, pour une application simple, se passer des directives TRACE et DEGRE ; dans ce cas il n’y aura aucun appel concernant les labels, les entrées et sorties de procédures ; le degré des appels explicites sera 1 ou celui spécifié dans l’instruction.

3.3 - LES APPELS A DRIP16 EN ASSEMBLEUR ASM

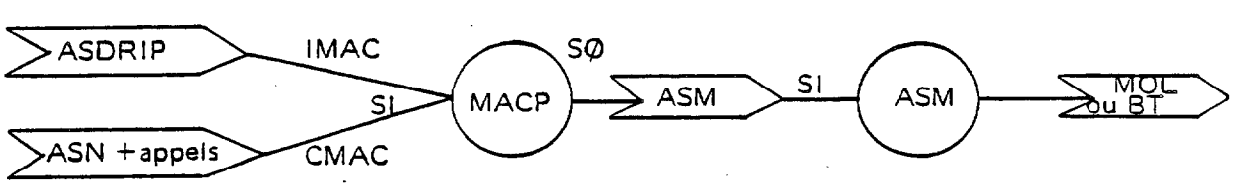
La programmation des appels à DRIP16 en assembleur est facilitée par l’utilisation de ASDRIP.

3.3.1 - ASDRIP

ASDRIP est un ensemble de macros-définitions permettant, par l’emploi de MACP, de générer en langage assembleur, les séquences d’appels à DRIP16.

L’utilisateur définit les extractions qu’il désire effectuer par le jeu de macro-instructions dont la syntaxe est identique à la syntaxe PL16 décrite dans le paragraphe précédent.

3.3.2 - Utilisation



Exemple :

```
CALL MACP
SI HR *                 bibliothèque ASDRIP
IMAC
SI HR                   programme assembleur avec appels

CMAC
CALL ASM
BØ FICHIER
IASM
```

3.3.3 - Les directives

% DRIP ØN

Cette directive valide les appels au module DRIP16.

% DRIP ØFF

Cette directive permet d'ignorer les appels au module DRIP16. C'est ce qui est pris par défaut.

% DEGRE [d]

Cette directive permet de définir le degré courant d associé aux appels ultérieurs générés par MACP.
Pour tout appel au module DRIP16, le macro-générateur génère implicitement la valeur d du degré de la zone courante (degré de la dernière directive DEGRE).
En l'absence de cette directive le degré courant est pris égal à 1.

% ENDRIP

Cette directive permet d'imprimer le nombre d'erreurs détectées par ASDRIP au cours de la génération des appels à DRIP16.

3.3.4 - Les instructions

L'utilisateur dispose seulement des appels explicites au module DRIP16.

% SNAP [d] ({ étiquette [+ déplacement] / RA } , n [,F]) ;

Cette macro-instruction permet la visualisation de n mots, dans le format F, à partir de l'étiquette nommée ou de l'adresse contenue dans RA.
F et d : idem SNAP (paragraphe 3.2.2).

```
% RSNAP [ d ]
```

Idem RSNAP en PL16 (paragraphe 3.2.2).

```
% XPRINT [ d ] "CHAINE"
```

Idem XPRINT en PL16 (paragraphe 3.2.2).
La longueur de la chaîne est limitée à 50 caractères.

```
% XSYST [ d ]   (n [, liste de nombres ] )
```

Idem XSYST en PL16 (paragraphe 3.2.2).

n = 1 ou	SCHEDU	Ces symboles sont prédéclarés par ASDRIP dans le programme utilisateur
2	ETAEVE	
3	PHASAV	
4	ØCPAR	
5	ATPAR	
6	ZØNDYN	
7	ZDR	
8	ETAPER	
9	BCT	
10	ODIPER	
11	TATTEV	
12	RESSØU	
13	FICØUV	
14	ZØNSYS	

3.3.5 - Les erreurs

Elles sont de 2 natures :

Erreurs MACP

- ERM 05 : macro-instruction incorrecte : vérifier la syntaxe de l'appel et si cet appel est autorisé par ASDRIP (erreur non fatale)
- ERM 12 : saturation des tables de MACP : liste de paramètres trop longue, vérifier la syntaxe (erreur fatale : corriger erreur et recommencer).

Erreurs ASDRIP

Il y a un libellé pour chaque erreur détectée par ASDRIP. Ce libellé est généré dans le fichier SO de MACP en dessous du nom de l'appel.

Exemple :

```
% XSYST (SCHEDU, 3)
```

conduit à la génération de :

```
< !!! XSYST (SCHEDU, 3)
< !!! ERREUR DRIP16 !!! LISTE DE PARAMETRES INCORRECTE !!!
```

Exemple récapitulatif :

```
    TABLE TESSAI
  TABLØ : DZS 20
          ⁞
    % DRIP ØN
      PRØG ESSAI
    % XSYST 1 (SCHEDU)
    % XSYST 8 (ETAPER,8)
    % XSYST 8 (ETAPER,'F)
    % XSYST 14 (ZØNSYS,'002A,24)
    % XPRINT 16 "INFØS LØCALES"
    % DEGRE 30
    % SNAP (TABLØ,10,1)  }
                         }   le degré de ces appels est 30
    % SNAP (RA.20)       }
    % RSNAP 18
    % DRIP ØFF
    % XPRINT 19 "FIN DE GÉNERATION" cet appel n'est pas pris en compte par
                                    ASDRIP
    % ENDRIP
    * END
```

3.4 - LES APPELS RECONNUS PAR LA VERSION MINIMUM DRIP16-A

La version minimum de DRIP16 reconnaît tous les appels cités auparavant, mais ne produit aucune extraction pour les appels XSYST (qui sont ignorés).

4 - LES COMMANDES DE CONTROLE DE DRIP16

4.1 - PRINCIPE

Les commandes de contrôle, émises à travers le dialogue du système. autorisent une sélection des appels à DRIP16 et régulent ainsi le flot des extractions.

Elles permettent :

- d'ignorer certains appels
- de sélecter le support de sortie pour les extractions prises en compte

et de plus :

- de définir le contexte dans lequel DRIP16 est appelé : monoprogrammation ou multiprogrammation
- de cataloguer les fichiers disque support des extractions en vue d'une exploitation en différé.

4 2 - DEFINITION DU CONTEXTE D'APPEL DE DRIP16

Dans son état initial, c'est-à-dire après son chargement en mémoire, DRIP16 est utilisable dans un contexte de monoprogrammation (Batch ou interactif). Pour le rendre utilisable dans un contexte de multiprogrammation, c'est-à-dire pour qu'il puisse être appelé par plusieurs tâches simultanément, l'utilisateur doit émettre la commande

```
MULTI
```

Cette commande est donc nécessaire sous les systèmes RTES-D, RTES-C, MPES et éventuellement sous BOS-D.

4.3 - CRITERES DE SELECTION

Parmi les informations mises en forme par DRIP16, on a distingué :

4.3.1 - Les informations locales au programme testé

. passage par label	(1)
. entrée de procédure	(2)
. sortie de procédure	(3)
. dump de zone	(4)
. dump de registres	(5)
. édition de messages	(6)
. silence	(7)
. nosilence	(8)

Sur ces informations on propose en MONOPROGRAMMATION :

. une validation ou une inhibition complète

la commande

ONTR

valide la prise en compte des appels 1 à 6

la commande

OFTR

inhibe la prise en compte des appels 1 à 6

. une sélection par degré d'appel

en aval du filtrage précédent, on réalise sur les types d'appels 1 à 8 un tri par degré d'appel

la commande

DEGR, L1, L2, D1, D2

définit, sur la plage [1, 255] des degrés d'appel possibles

3 sous-plages :

- plage [L1, L2] : les informations associées à ces degrés d'appel sont sorties sur imprimante
- plage [D1, D2] : les informations associées à ces degrés d'appel sont stockées sur fichier circulaire disque.
- plage complémentaire : les appels associés sont ignorés.

Ces plages peuvent être vides ou avoir une intersection non nulle.

Si L1 et/ou L2 = 0 ou est absent la plage imprimante est vide (aucune extraction), de même pour la plage disque.
Ainsi les commandes suivantes sont équivalentes et conduisent à des plages vides :

DEGR DEGR, 0 DEGR, 0, 0, 0, 0 DEGR, 0, 60

En MULTIPROGRAMMATION les commandes précédentes deviennent :

ONTR [, Pi, Pj, Pk . . . . .]

et

OFTR [, Pi, Pj, Pk . . . . .]

pour respectivement, validation et inhibition des appels effectués par les tâches de priorité Pi, Pj, Pk

Le nombre des priorités que l'on peut indiquer est limité à 25.
De plus ONTR, , 5 est identique à ONTR , 0, .5.

Si aucune priorité n'est indiquée la validation et l'inhibition concernent toutes les priorités.

```
DEGR, L1, L2, D1, D2 [,P]
```

pour définir les plages imprimante et disque pour la tâche de priorité P.

La hiérarchie des degrés est donc différente pour chaque priorité.
Si P est absent, les plages sont les mêmes pour toutes les tâches de l'application.
II ne peut être fait mention d'une priorité dans une commande (ONTR, OFTR, DEGR) tant que la commande MULTI n'a pas été émise,

4.32 - Les informations «système»

Ces informations extraites des tables de RTES-D ou RTES-C sont sélectées selon leur nature (directement liée au type de l'appel) indépendamment de l'appelant. La sélection du support de sortie est également fonction du degré de l'appel.

La commande :

```
ONSYS [ , Ni, Nj, Nk ---- ]
```

valide la prise en compte des appels relatifs aux informations "système" de types Ni, Nj, Nk . . . . . .

La commande :

```
OFSYS [ , Ni. Nj, Nk . . . . . ]
```

inhibe les mêmes appels.

Pour ces deux commandes, les paramètres N sont optionnels : par défaut la validation ou l'inhibition est globale. La numérotation N est identique à celle définie pour l'instruction XSYST.

4.4 - GESTION DU FICHIER CIRCULAIRE DISQUE

Lorsque le degré d'un appel au module DRIP16 appartient à la plage disque les informations associées sont écrites sur un "fichier circulaire" disque dans un format brut. Ces informations seront exploitées en différé par le processeur de dépouillement PRODEP.

Ce "fichier circulaire" se compose en fait de deux fichiers permanents d'accès séquentiel bornés, exploités en bascule : à un instant donné, on distingue le fichier actif et le fichier au repos :

. le fichier actif est ouvert en écriture et incrémenté par DRIP16
. le fichier au repos est provisoirement fermé

Ces deux fichiers ont un nom prédéfini :

: < SS10- SS pour le fichier 1
: < SS20- SS pour le fichier 2

La commande

DFIL, taille

initialise le "fichier circulaire", et par ce fait est obligatoire.

Cette commande
. crée les 2 fichiers sur l'unité U6
. ouvre le fichier actif SS10
. ferme le fichier au repos SS20
. définit la taille maximum de chaque fichier en K mots

Par défaut d'indication de la taille, ce paramètre est pris égal à 30.

L'opérateur est informé des communications de fichiers par le message :

FILE { 1 / 2 } FULL

qui est imprimé sur EL.
II peut alors "déconnecter" le fichier au repos et le renommer.

La commande

DISC, R, FICNAM - PW

opère cette déconnexion et renomme le fichier au repos avec le nom de fichier donné en paramètre.

En fin de session, l'utilisateur peut aussi "déconnecter" le fichier actif et le renommer par la commande :

DISC, A, FICNAM - PW

Cette commande marque la fin des extractions disque. Celles-ci ne seront à nouveau possibles qu'après une nouvelle commande DFIL

Les fichiers "déconnectés" seront dépouillés en différé par le processeur PRODEP.

4.5 - ETAT INSTANTANE DE DRIP 16

La commande

F L T R

provoque l'impression sur l'unité symbolique U5 de l'état des indicateurs de DRIP 16 à un instant donné.

Le format de ce dump se décrit ainsi :

1ère ligne : dump hexadécimal de 8 mots représentant 128 bits

bit i = 0 si la tâche de priorité i a réalisé un NO SILENCE
bit i = 1 si la tâche de priorité i a réalisé un SILENCE

2ème ligne : dump hexadécimal de 8 mots représentant 128 bits

bit i = 0 si la tâche de priorité i est validée par ONTR
bit i = 1 si la tâche de priorité i est invalidée par OFTR

3ème ligne : dump hexadécimal de 1 mot représentant 16 bits

bit i = 0 si appel système i validé par ONSYS
bit i = 1 si appel système i invalidé par OFSYS

Viennent ensuite :

8 lignes de 16 valeurs décimales, soit 128 nombres qui représentent les valeurs L1 (début plage imprimante) pour les 128 priorités.

8 lignes de 16 valeurs décimales soit 128 nombres qui représentent les valeurs L2 (fin plage imprimante) pour les 128 priorités.

8 lignes de 16 nombres décimaux pour les valeurs D1 (début plage disque).

8 lignes de 16 nombres décimaux pour les valeurs D2 (fin plage disque).

4.6 - ETAT INITIAL DU PROCESSEUR DRIP16

Lors de son intégration, le processeur se trouve dans un état initial qui est celui défini par compilation ; cet état se caractérise ainsi :

- DRIP16 est utilisable dans un contexte de monoprogrammation
- tous les appels "locaux" (1 à 8) sont validés pour toute priorité
- tous les appels "système" sont inhibés
- vidage systématique des extractions sur imprimante

[L1, L2] ≡ [1,255]

[D1, D2] ≡ [0 , 0] pour toute priorité

4.7 - LES COMMANDES DANS UN CONTEXTE DE MONOPROGRAMMATION DE TYPE INTERACTIF

Chaque fois qu'il est activé, et avant de redonner le contrôle au programme utilisateur, le module DRIP16 teste "l'appel opérateur" et se met éventuellement en attente de commande.

L'utilisateur a ainsi la possibilité d'arrêter son programme afin :

- de donner, par les commandes déjà citées, de nouveaux critères de sélection
- de visualiser sur le périphérique de dialogue l'état des registres de son programme par la commande

REGS

Ainsi en cas de bouclage l'utilisateur peut connaître la valeur de ses bases et de son pointeur P.

II relancera l'exécution de son programme par la clé :

CONT

4.8 - LES COMMANDES RECONNUES PAR LA VERSION MINIMUM DRIP16-A

La version minimum de DRIP16 ne reconnaît que les commandes suivantes :

MULTI

ONTR — sans paramètre

OFTR — sans paramètre

pour, respectivement validation et inhibition des appels 1 à 6

DEGR, L1 , L2

qui définit sur la plage [1 à 255] des degrés d'appel possibles
2 sous plages :
- plage [L1, L2] : les informations associées à ces degrés sont sortis sur imprimante
- plage complémentaire : les appels associés sont ignorés.

REGS — cf § 4.7

CONT

5 — L'EDITION DES EXTRACTIONS

Les appels pris en compte par le module DRIP16 et dont le degré appartient à la plage [L1,L2] sont édités d'une manière claire sur le dispositif associé à l'unité symbolique U5.

L'utilisateur doit donc affecter U5 à l'unité fonctionnelle sur laquelle il désire, voir imprimer ses extractions : imprimante ou visualisation alphanumérique.

Remarque : l'impression des extractions nécessite un POOL de Buffers de longueur au moins égale à 120 octets.

Description des enregistrements

Pour chaque appel, il y aura :

* en partie droite
  - le mnémonique identifiant l'appel
  - le numéro de ligne de l'appel ; il représente le numéro de ligne PL16 pour un programme écrit en PL16 (sa valeur est donnée en décimal). II représente le compteur d'assemblage pour un programme écrit en assembleur dont les appels au module DRIP16 ont été générés par ASDRIP (sa valeur est donnée en hexadécimal).
  - le degré de l'appel
  - la priorité de l'appelant
  - l'heure de l'appel (pour DRIP16-B seulement).

Exemple : - - EN - - - - - - - - - - L0787 D001 P045 - 14H 10M 55S-

ligne Degré Priorité heure

dans cet exemple le mnémonique est EN pour ENtrée de procédure.

* en partie gauche, l'extraction proprement dite que nous détaillons ici :

(1) Passage par label

Mnémonique : LA -> DEPART Nom du label

(2) Entrée de procédure

Mnémonique : EN -> PROCC Nom de la procédure

(3) Sortie de procédure

Mnémonique : SO -> PROCC Nom de la procédure

(4) Dump de zone

Mnémonique : SN == FCBDWT
0198 : 0210 0248 0002 0000 0003

Nom de zone et dump

(5) Dump de registres

Mnémonique : RG — Dump des 13 Registres

```
= =
A = '0001  B = '0002  X = '0003   ---------------  P = '01DC
ST = '0000  O = '088F  E = '09FF   IM = 'FF78
```

(6) Edition de message

Mnémonique : MG == BUFFER NOM DU FICHIER

(7) Inhibition de 1, 2 et 3

Mnémonique : INHI — Pas d'extraction

(8) Validation de 1, 2 et 3

Mnémonique : VALI — Pas d'extraction

(9) Files du scheduler microprogrammé

Mnémonique : FS

Cet appel provoque le dump, en format hexadécimal, des 3 files ASTF, ESTF, RSTF (3 lignes de 8 mots soit 3 lignes de 128 bits).

(10) Etat des événements

Mnémonique : EV

Cet appel provoque le dump, en format hexadécimal, de la table des événements si ceux-ci sont gérés par le système d'exploitation.

	Classe 0		Classe 1		Classe 2		Classe 3	
0174 :	0410	0000	0000	0000	C553	5341	C953	4141
017C :	908B	0000	0002	0006	0000	0000	0000	0000
	Classe 4		Classe 5		Classe 6		Classe 7	

Ce dump consiste en 2 lignes de 8 mots, soit 256 bits, pour représenter 256 événements (1 bit par événement).
L'événement i est SET si le bit i est à 1
L'événement i est RESET si le bit i est à 0

(11) Phases d'avancement des tâches

Mnémonique : PH

Cet appel provoque le dump, en format décimal, des 128 octets qui représentent l'état d'avancement des 128 tâches de l'application, sous la condition que cette notion soit gérée par le système. Ce dump consiste en 8 lignes de 16 octets.

(12) Occupation des partitions

Mnémonique : OP

Cet appel provoque le dump, en format décimal, des n octets qui définissent les numéros des tâches occupant chacune des n partitions de l'application.

La valeur maximum de n est 16 sous RTES-C, 48 sous RTES-D.

(13) Taux d'attente des partitions non résidentes

Mnémonique : AP

Cet appel provoque le dump en format décimal, des n octets qui définissent le nombre de tâches en attente sur chacune des n partitions de l'application.

La valeur maximum de n est 16 sous RTES-C, 48 sous RTES-D.
Ces informations, permettant de savoir si l'application a des risques de se trouver bloquée par manque de la ressource mémoire, sont importantes en phase d'évaluation d'une application ; au vu de ces données, l'utilisateur peut reconsidérer ses choix sur l'affectation des partitions aux différentes tâches de l'application.

(14) Taux d'occupation des zones dynamiques du système

Mnémonique : DY

Cet appel provoque le dump, en format décimal, de 6 octets qui représentent respectivement le nombre maximum moins un de pavés occupés depuis le lancement de l'application dans les zones dynamiques suivantes :

ZUEP, ZIOCB, ZFAU, ZDF, ZGIN, ZBCT

Se reporter aux manuels d'utilisation de RTES-D et RTES-C pour tout renseignement concernant ces zones.

Ces informations, comme pour l'appel précédent, permettent à l'utilisateur d'évaluer son application et éventuellement de reconsidérer ses choix.

(15) Portion de ZDR

Mnémonique : ZR

Cet appel provoque le dump, en format hexadécimal, de la portion de zone des Données Résidentes choisie par l'utilisateur.

16 Etat d'un périphérique

Mnémonique : EP

Cet appel provoque le dump, en format hexadécimal, du mot d'état de l'unité physique attachée à la SU ou FU concernée. Devant le mot d'état est imprime le numéro de la SU ou FU.

Se reporter aux différents manuels des drivers pour l'interprétation de ce mot d'état.

17 Bloc de contrôle d'une tâche (BCT)

Mnémonique : PS

Cet appel provoque le dump, en format hexadécimal, des 28 mots constituant le bloc de contrôle de la tâche concernée (PST et son extension).

Se reporter au manuel d'utilisation de RTES-D pour l'interprétation de ce bloc d'informations.

18 Liste des opérations différées et périodiques : liste horloge

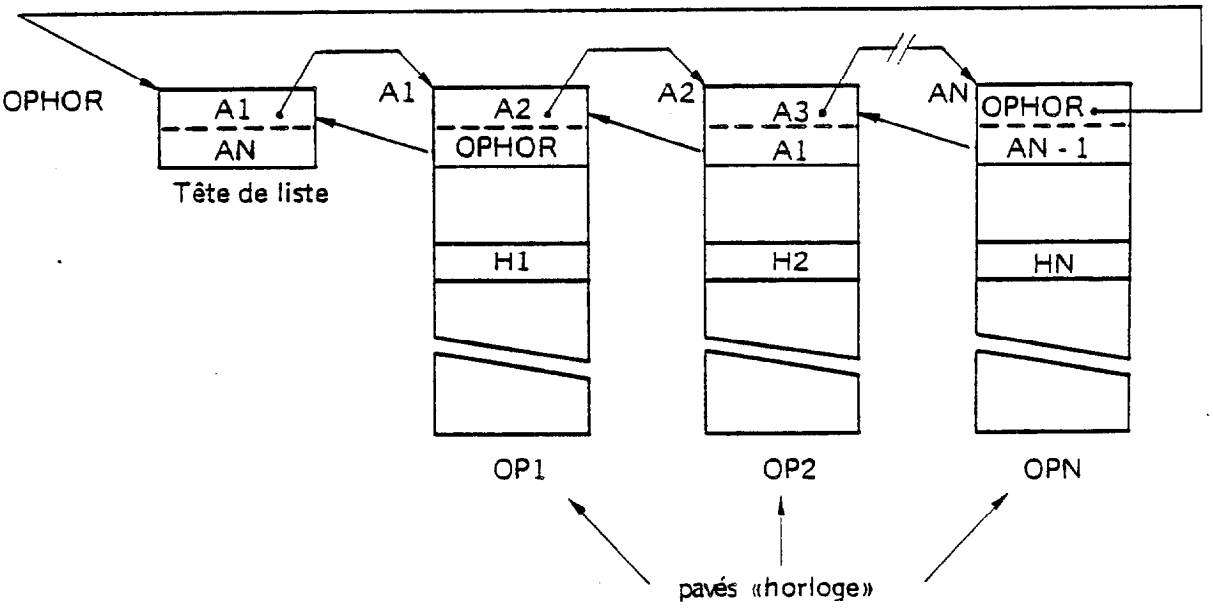
Mnémonique : LD

Cet appel conduit au dump, en format hexadécimal, des blocs d'informations chaînés (nommés pavés) correspondant à toutes les opérations sur horloge demandées au système (RTES-D ou RTES-C) à un instant donné.

Cette liste de pavés, nommée liste horloge, est dumpée pavé après pavé, soit opération après opération, ces opérations étant prises dans l'ordre de la liste, c'est-à-dire ordonnées dans le temps.

Nous décrivons ici la gestion des opérations sur horloge réalisée par RTES-D ou RTES-C ainsi que le contenu des blocs d'informations associées aux opérations.

. Description de la liste horloge



OP1, OP2, OPN sont des opérations différées, ordonnées dans le temps. OP1 est l'opération la plus "proche" c'est-à-dire celle qui est à réaliser la première (heure d'échéance H1), puis viennent OP2, OP3 . . . .

Remarque :

Il peut exister en fin de liste horloge des opérations qui ne sont pas ordonnées par rapport aux précédentes : ce sont les opérations sur horloge demandées au système depuis le dernier "top" d'horloge. Elles se distinguent par le bit 15 positionné à 1 dans le mot 2 du pavé. Ces opérations seront ordonnées au prochain "top" d'horloge.

. Description des pavés horloge

Il existe 3 types d'opérations sur horloge :

a) suspension de tâche (requête WAIT)

b) activation différée de tâche (requêtes START, STARTW, TRNON, TRNONW)

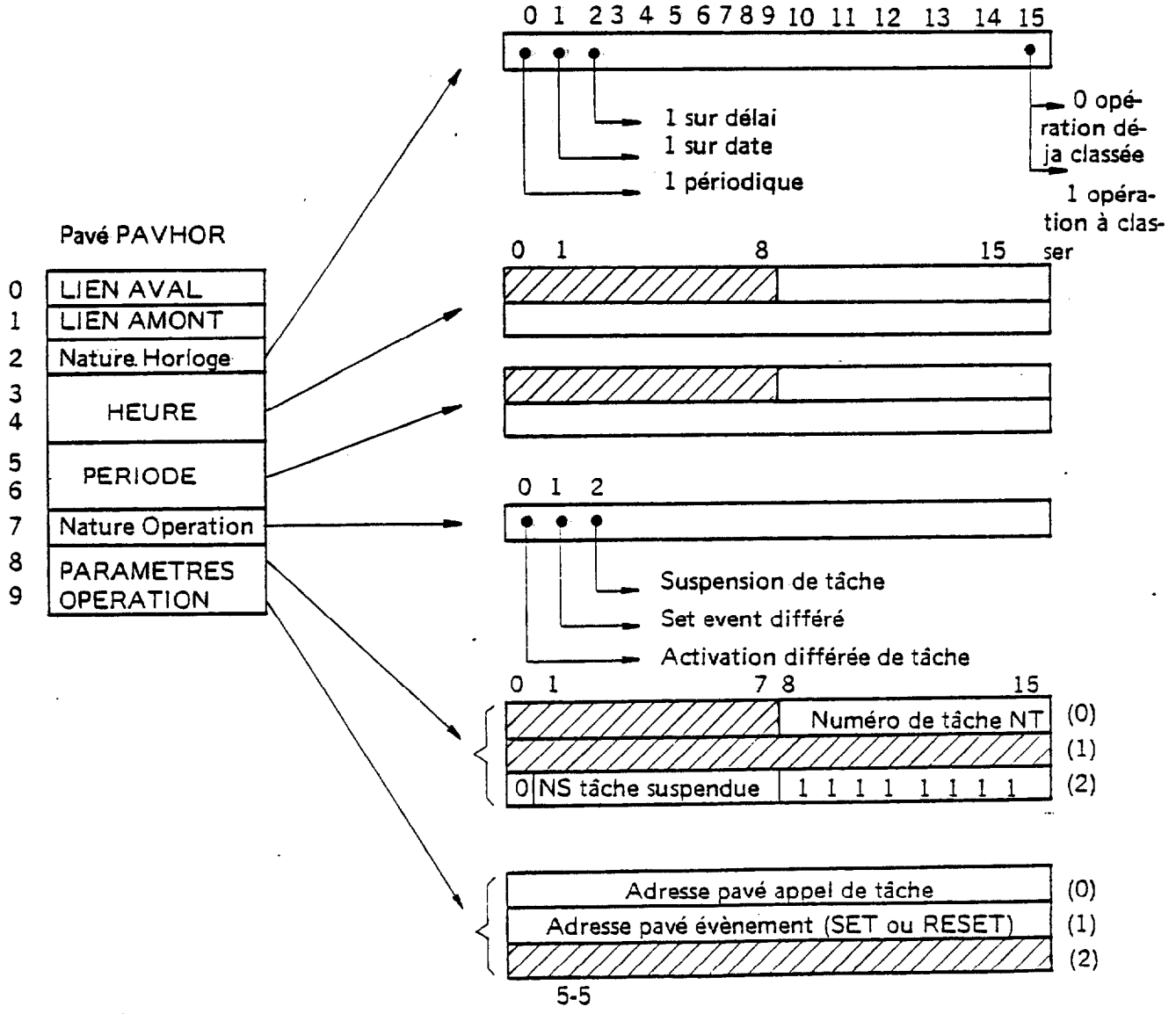
c) set event différé (requête SEVDEL)

de plus 2 types d'inhibitions de ces opérations

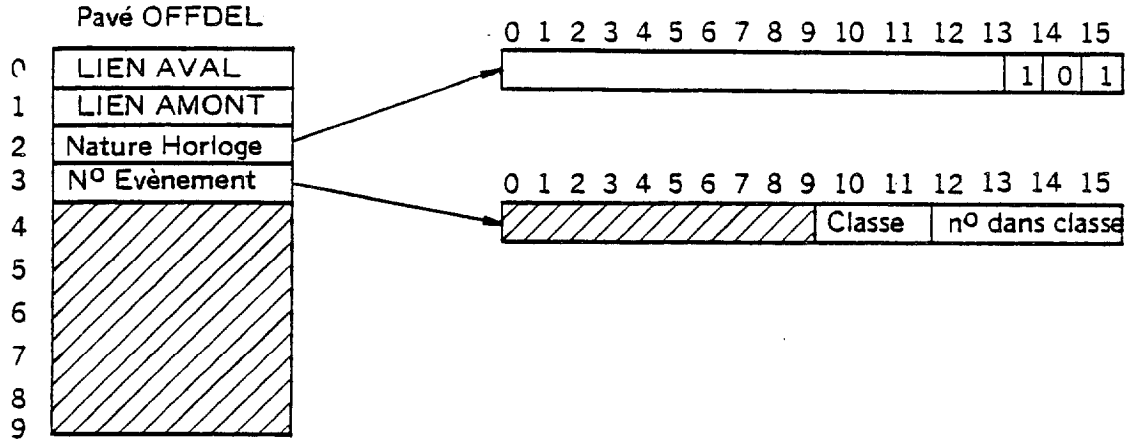
d) effacement de la requête d'arrivée différée d'un événement (requête OFFDEL)

e) suppression des demandes d'activation différées et périodiques d'une tâche faite avec un paramètre de travail spécifique (requête OFF)

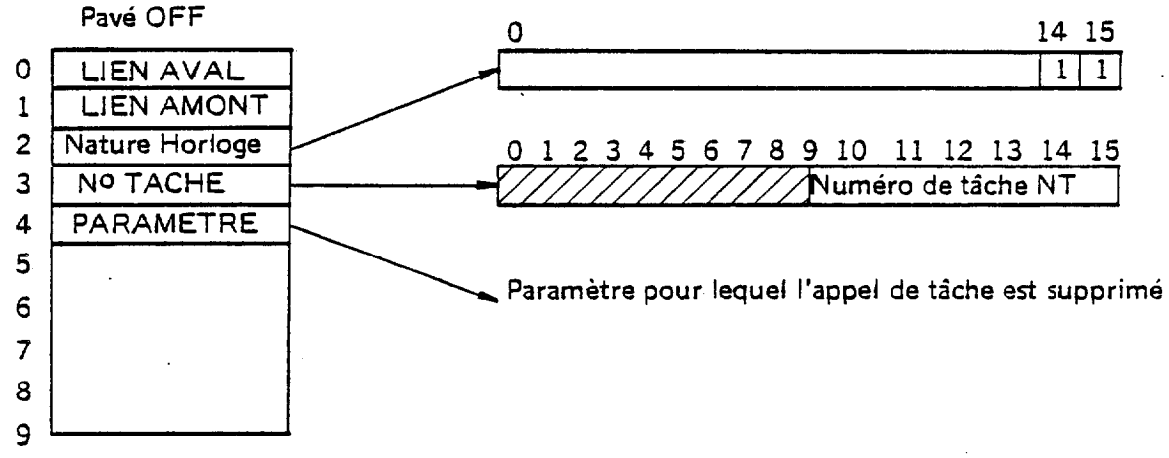
Pour les types a, b, c les pavés ont la forme suivante :



Pour le type d le pavé a la forme suivante :



Pour le type e le pavé a la forme suivante :



Ces différents blocs se trouvent dans la ZGIN.

. Principe de traitement des requêtes sur horloge

Sur toutes les requêtes vues plus haut, le système chaîne un pavé horloge en fond de liste horloge, ce pavé n'est alors pas ordonné dans le temps.

A chaque "top" d'horloge, le driver horloge est activé pour :

. assurer la gestion de la date et heure internes du système
. activer la tâche software RLOGE si :

a) il est minuit
b) une ou plusieurs demandes d'opérations sur horloge sont intervenues entre ce top et le précédent : il y a un ou plusieurs pavés en fin de liste qui doivent être classés (pavés de types a, b, c) ou qu'il faut traiter car ils concernent des suppressions d'opérations (pavés OFFDEL et OFF)
c) une opération est arrivée à échéance : l'heure indiquée dans le premier pavé est égale à l'heure interne du système.

(19) Listes des tâches en attente d'événement

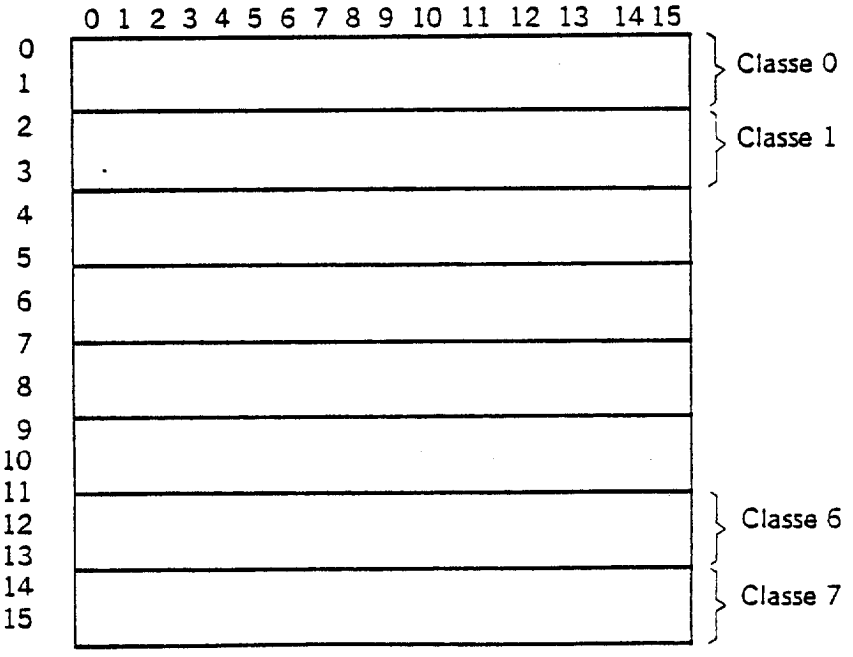
Mnémonique : LV

Cet appel conduit au dump, en format hexadécimal, des blocs d'informations chaînés (pavés) correspondant aux tâches en attente d'événement à un instant donné (tâches qui ont effectué les requêtes WEVENT, WEVAND, WEVOR). Le dump est effectué, liste après liste (il y a une liste par classe d'événements) et pavé après pavé.

Nous décrivons ici la gestion des événements telle qu'elle est réalisée sous RTES-D (les événements ne sont pas gérés sous RTES-C).

. Principe

Il existe 256 événements dont les états sont mémorisés dans une table de 16 mots. Ces 256 événements sont partagés en 8 classes de 32 événements :



Une requête porte donc sur :

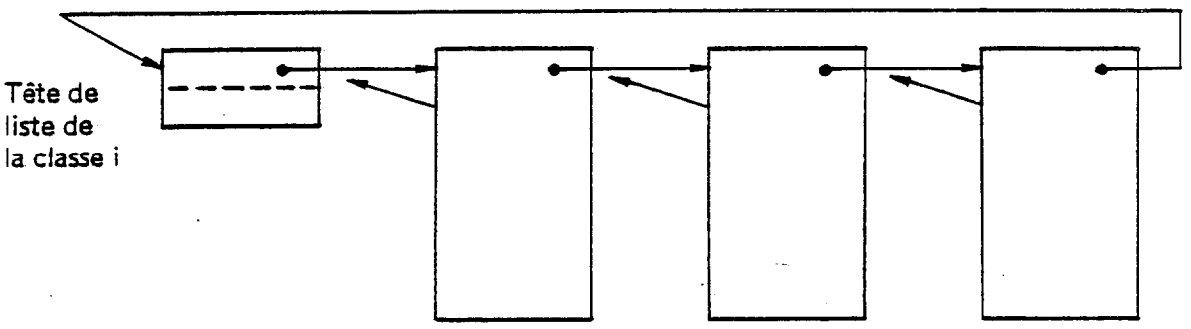
. un numéro d'événement (0 à 255)
ou
. la liste (2 mots) des événements d'une classe
(cf. requêtes WEVENT, WEVAND, WEVCR dans manuel de référence de RTES-D).

On rappelle qu'avec le numéro d'événement est implicitement précise :

. le numéro de classe
. le numéro d'événement dans la classe

. Fonctionnement

A chacune des requêtes de mise en attente sur événement, lorsque la condition n'est pas remplie, on chaîne la tâche à la liste des événements de la classe correspondante.

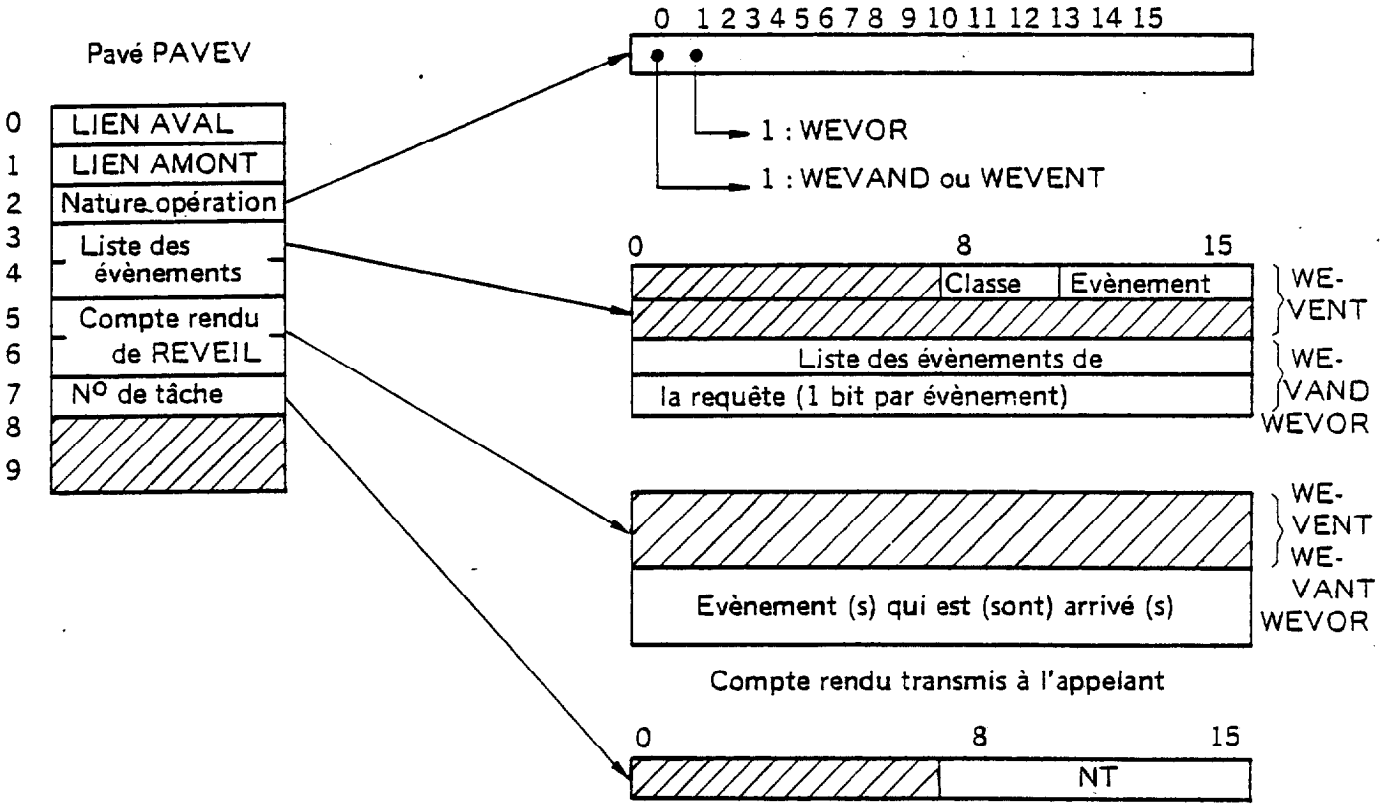


Liste des tâches en attente d'événement sur la classe i

Lors d'une requête de positionnement d'état (SEVENT, REVENT), la liste de la classe à laquelle appartient l'événement est scrutée et les tâches qui étaient en attente sur l'arrivée de cet événement sont réveillées.

Les différentes listes se trouvent dans la ZGIN.

. Description des pavés attente d'événement

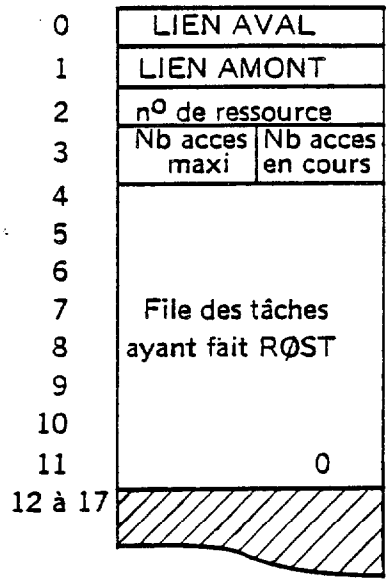


(20) Liste des ressources

Mnémonique : LR

Cet appel conduit au dump. en format hexadécimal, des blocs d'informations chaînés (pavés) décrivant les ressources définies par l'utilisateur (requête RESDEF). Le dump est effectué, ressource après ressource.

Description des pavés ressource



Ces pavés sont rangés dans la ZDF.

(21) Liste des fichiers ouverts

Mnémonique : LF

Cet appel provoque le dump, en format hexadécimal de blocs d'informations de FMS constituant les zones système suivantes :

* ZUEP : blocs de 128 mots
* ZIOCB : blocs de 6 mots
* ZFAU : blocs de 20 mots
* ZDF : blocs de 18 mots

Se reporter au manuel d'utilisation de FMS/E pour tout renseignement concernant ces informations.

(22) Dump de zone système

Mnémonique : ZS

Cet appel provoque le dump, en format hexadécimal, de la *zone* dont l'utilisateur a fourni l'adresse en valeur absolue.

Remarque :
On ajoute que pour les appels implicites (appels correspondant au "chemin suivi") les deux premiers caractères de l'extraction sont -> tandis que pour tous les appels explicites les deux premiers caractères sont = = (pour les appels locaux) ou * * (pour les appels système).
Cette distinction permet un repérage plus facile des différents types d'informations.

6 - LE STOCKAGE DES EXTRACTIONS SUR DISQUE

Les appels pris en compte par le module DRIP16 et dont le degré appartient à la plage [D1, D2] sont stockés d'une manière brute sur fichier circulaire disque par l'intermédiaire de l'unité symbolique U6. L'utilisateur doit donc affecter U6 à l'unité fonctionnelle disque sur laquelle se trouve le fichier circulaire :

U6 Di

Nous rappelons que l'utilisateur doit aussi, dans ce cas de stockage sur disque, envoyer les commandes suivantes :

[MULTI]	pour un contexte de travail de multiprogrammation (RTES-D, RTES-C, MPES)
DFIL, n	pour créer et ouvrir le fichier circulaire

avant de lancer son application.

A chaque extraction stockée sur disque correspond un enregistrement qui se décompose en :

. un identificateur de 7 mots ainsi constitué

Degré de l'appel	Type de l'appel
n° de ligne de l'appel	
Priorité de l'appelant	
heure appel	
minute	
seconde	
milleseconde	

. un bloc d'informations, sous forme brute, fonction du type de l'appel.

Afin d'être le plus "transparent" possible, le module DRIP16 minimise ses échanges avec le disque en employant l'option "bufférisation" du système de fichier FMS/E. Dans la version standard de DRIP16 le buffer intermédiaire a une longueur de 256 mots, il est défini par le link edit au module DRIP16 de la séquence suivante :

```
          ENT     BUFCD
          TABLE BUFDIS
LGBUCD : VAL     256
          WORD LGBUCD          < TAILLE DU BUFFER
BUFCD : DZS     LGBUCD         < RESERVATION BUFFER
          END
```

L'utilisateur peut ainsi redéfinir à son gré la taille de ce buffer (pour l'annuler, la diminuer ou l'agrandir) afin d'établir selon ses besoins un compromis entre la place occupée et le temps (cf. chapitre Génération).

En cas d'erreur intervenant lors de l'écriture dans le fichier circulaire, la plage disque est invalidée afin de ne pas arrêter le déroulement de l'application.

7 - LES EXTRACTIONS SYSTEME

Les extractions système prises en compte par le module DRIP16 (validées par la commande ONSYS) sont, suivant leur degré, éditées sur l'unité symbolique U5 dans le format décrit au chapitre 5 ou bien stockées sur le fichier circulaire ou encore ignorées.

Ces informations sont extraites des différentes tables de RTES-D ou RTES-C et amenées, toutes interruptions masquées, dans une zone tampon à partir de laquelle elles sont mises en forme ou stockées sur disque (les informations de types 13 et 14, tables de FMS et dump de zone absolue, ne passent pas dans la zone tampon).

Dans la version standard de DRIP16, la zone tampon a une longueur de 256 mots, elle est définie par link edit au module DRIP16 de la séquence suivante :

```
            ENT     BUFSYS
            TABLE BUFSY
LGBUSY    : VAL     256
            WORD LGBUSY            < TAILLE DE LA ZONE
BUFSYS    : DZS     LGBUSY         < RESERVATION ZONE
            END
```

La taille ainsi choisie permet d'obtenir toutes les informations système de taille fixe et de récupérer des informations chaînées ayant une taille ne dépassant pas 25 pavés du type horloge ou événement (pavés de 10 mots) ou 14 pavés ressource (pavés de 18 mots).

Lorsque la zone tampon est insuffisante pour ranger des informations chaînées, l'utilisateur est averti par un message d'erreur que ces informations sont tronquées. Ce dernier a alors la possibilité de redéfinir la taille de la zone en changeant la valeur de LGBUSY ; il établit ainsi, lui-même, le compromis entre la place occupée par le module DRIP16 et le service rendu.

8 - GENERATION D'UN MODULE DRIP16

Le module DRIP16 dans sa version réduite, DRIP16-A, offre un service minimum (pour une taille de 950 mots) et un service maximum dans sa version complète, DRIP16-B (pour une taille de 3500 mots).

La conception modulaire de DRIP16 permet à l'utilisateur de générer à partir de la version complète DRIP16-B la version qui leur offre les services réellement adaptés à ses besoins. II peut, en effet :

. par le choix des modules
. par le choix des tailles des buffers BUFCD et BUFSYS

définir respectivement les services et la qualité des services qu'il demande à son outil de mise au point, ce qui lui permet d'établir, à son gré, le compromis entre l'encombrement mémoire et les facilités offertes.

L'utilisateur n'a recours à la procédure de génération d'un module DRIP16 que lorsqu'il veut réduire le nombre de fonctions de DRIP16-B ou lorsque les tailles des buffers BUFCD et BUFSYS ne lui conviennent pas.

8.1 - STRUCTURE DU MODULE DRIP16

La version complète DRIP16-8 est composée de 6 modules :

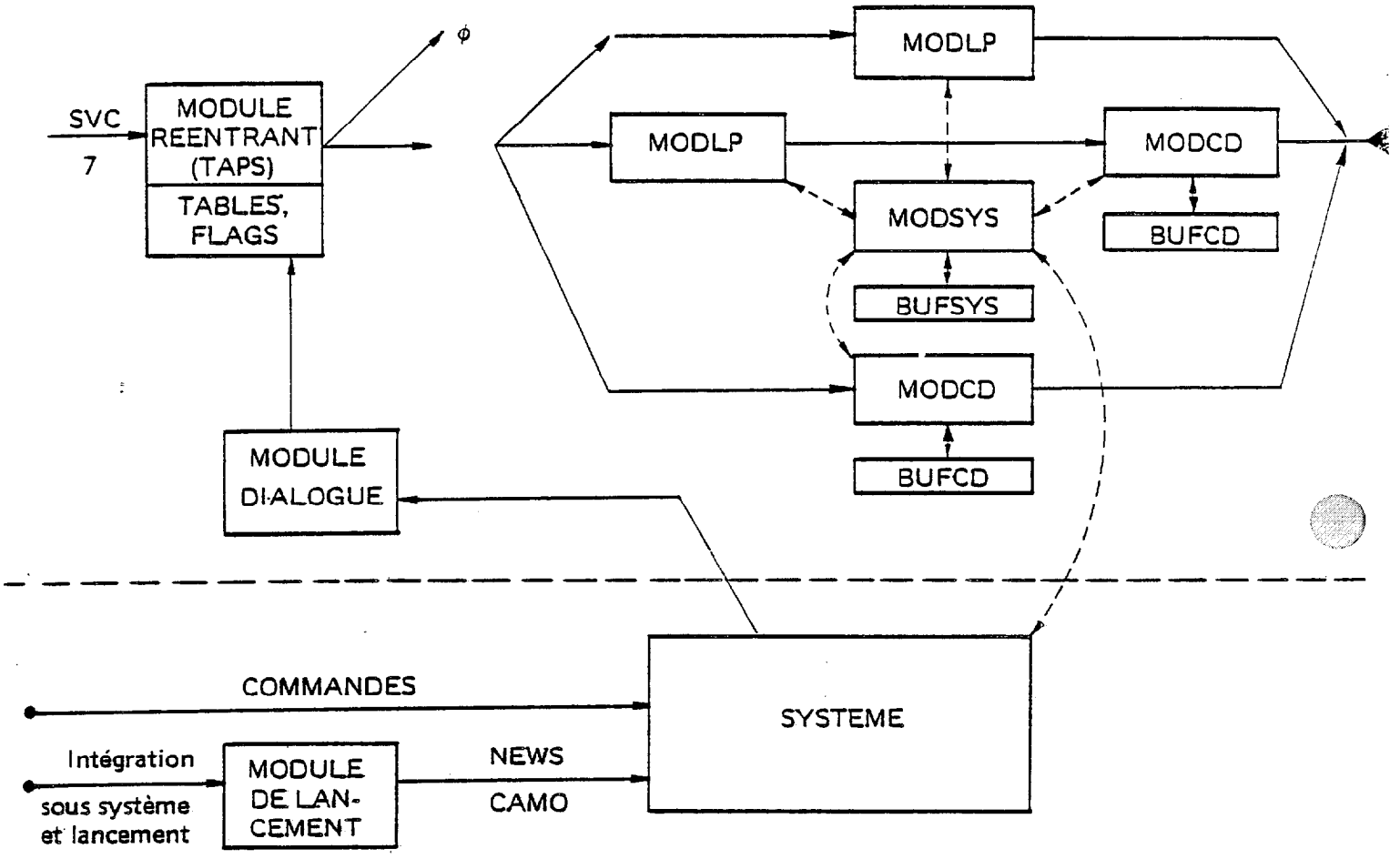
. module de lancement (appelé LANTRA)
. module de dialogue (appelé DIATRA)
. module réentrant (appelé CSTRAS)
. module d'édition (appelé MODLP)
. module de stockage sur disque (appelé MODCD)

auquel est lié le buffer BUFCD de 256 mots

. module d'extraction système (appelé MODSYS)

auquel est lié le buffer BUFSYS de 256 mots

qui s'organisent selon le schéma suivant :



Les modules : LANTRA (lancement)
DIATRA (dialogue)
CSTRAS (réentrant) sont obligatoires

car ils sont, respectivement, nécessaires à l'intégration sous système
à la réception des commandes
à la prise en compte de la SVC 7

par contre les modules :

MODLP
MODCD
MODSYS sont facultatifs

car l'utilisateur peut désirer, selon son contexte de travail, ne réaliser des extractions que sur imprimante, que sur disque, ou enfin faire ou ne pas faire des vidages système.

De plus, selon la place dont il dispose, l'utilisateur peut rendre son système plus ou moins performant en agissant sur la taille du buffer BUFCD (qui est en standard de 256 mots).

Enfin, l'utilisateur peut adapter la taille du buffer BUFSYS à la quantité d'informations chaînées susceptibles d'être extraites lors d'un appel système. Cette taille est fonction de la complexité (nombre de tâches) de l'application à espionner.

8.2 - LES FOURNITURES ASSOCIEES A DRIP16

Le produit DRIP16 est fourni à l'utilisateur sous forme de 5 bandes :

Bande 1 :	1	164	081	01/01	01	63	nommée DRIP16-A
Bande 2 :	1	164	081	02/01	11	63	nommée DRIP16-B
Bande 3 :	1	164	081	03/01	01	64	nommée BIDRIP
Bande 4 :	1	164	081	04/01	01	64	nommée GEDRIP
Bande 5 :	1	164	081	05/01	01	63	nommée PRODEP
Bande 6 :	1	164	081	06/01	01	64	nommée ASDRIP

La bande 1 est fournie en standard avec les autres logiciels de la gamme SOLAR.

Les bandes 2, 3, 4, 5 sont fournies avec RTES-D OU RTES-C.

Les bandes 1 et 2 constituent respectivement les binaires translatables de la version minimum et de la version complète de DRIP16.

La bande 3 constitue un fichier indexé dont les articles représentent les binaires linkeditables de tous les modules constitutifs de DRIP16-B.

La bande 4 constitue un fichier indexé dont les articles représentent les jeux de commandes servant à la génération d'un module DRIP16 "à la carte" à partir de la bande 3 (cf § suivant).

La bande 5 constitue le binaire translatable du processeur de dépouillement PRODEP (cf chapitre 11).

8.3 - PRINCIPE DE LA GENERATION

Pour réaliser la génération d'un module DRIP16 l'utilisateur dispose des 2 bandes nommées BIDRIP et GEDRIP et d'un disque.

La bande BIDRIP représente le dump (par FDUMP de FUP3) d'une bibliothèque dont les articles sont :

LANTRA	. BIDRIP - : S	binaire	link-editable de LANTRA
DIATRA	. BIDRIP - : S	binaire	link-editable de DIATRA
CSTRAS	. BIDRIP - : S	binaire	link-editable de CSTRAS
MODLP	. BIDRIP - : S	binaire	link-editable de MODLP
MODCD	. BIDRIP - : S	binaire	link-editable de MODCD
BUFCD	. BIDRIP - : S	binaire	link-editable du buffer BUFCD
MODSYS	. BIDRIP - : S	binaire	link-editable de MODSYS
BUFSYS	. BIDRIP - : S	binaire	link-editable du buffer BUFSYS
MLPOUT	. BIDRIP - : S	binaire	link-editable de MODLP
MCDOUT	. BIDRIP - : S	binaire	link-editable de MODCD
MSYOUT	. BIDRIP - : S	binaire	link-editable de MODSYS

Binaires constitutifs de la version DRIP16-B

Binaires des modules complémentaires de MODLP, MODCD et MODSYS

Les différentes versions possibles de DRIP16 sont représentées par le tableau suivant :

Modules	LANTRA	CSTRAS	DIATRA	MODLP	MODCD + BUFCD	MODSYS + BUFSYS	RESULTAT	TAILLE
Taille	96	147	1044	723	583 + 256	389+256		
0 si absents 1 si présent	1	1	1	0	0	0	pas d'intérêt	
	1	1	1	1	0	0	DRLP	2010
	1	1	1	0	1	0	DRCD	2126
	1	1	1	1	1	0	DRLPCD	2849
	1	1	1	0	0	1	pas d'intérêt	
	1	1	1	1	0	1	DRLPSY	2655
	1	1	1	0	1	1	DRCDSY	2771
	1	1	1	1	1	1	DRIP16-B	3494

Présents Obligatoirement — Facultatifs

Les tailles fournies ci-dessus sont obtenues avec des buffers standard BUFCD et BUFSYS de 256 mots, le module MODCD impliquant la présence obligatoire de BUFCD et le module MODSYS celle de BUFSYS.
Ces tailles sont celles de l'IE/00, elles ne seront pas remises à jour.

La génération d'une version du module DRIP16 consiste à produire un binaire translatable par le link-edit des modules obligatoires, des modules choisis ou de leur complémentaire, et des buffers, à l'aide des commandes fournies dans GEDRIP.

La bande GEDRIP représente le dump (par FDUMP de FUP3) d'une bibliothèque dont les articles sont :

DRLP .	GEDRIP - : S	jeu de commandes pour la fabrication de DRLP
DRCD .	GEDRIP - : S	jeu de commandes pour la fabrication de DRCD
DRLPCD .	GEDRIP - : S	jeu de commandes pour la fabrication de DRLPCD
DRLPSY .	GEDRIP - : S	jeu de commandes pour la fabrication de DRLPSY
DRCDSY .	GEDRIP - : S	jeu de commandes pour la fabrication de DRCDSY
DRIP16 .	GEDRIP - : S	jeu de commandes pour la fabrication de DRIP16-B

Ces différents jeux de commandes sont décrits en ANNEXE III.
Ils permettent d'obtenir des versions de DRIP16 utilisables en monoprogrammation et multiprogrammation dont les services sont fonction des modules effectivement intégrés. Un service demandé à DRIP16, alors que le module qui le traite est absent, est refusé ; ainsi la définition d'une place de degré [L1, L2] alors que MODLP est remplacé par MLPOUT conduit à une erreur.
Les clefs ONSY et OFSY n'existent pas si MODSYS est remplacé par MSYOUT.
Les clefs DFIL et DISC n'existent pas si MODCD est remplacé par MCDOUT.
La clef FLTR n'existe pas si MODLP est absent.

8.4 - LA GENERATION

8.4.1 - Intégration des bibliothèques BIDRIP et GEDRIP

Celle-ci s'effectue par le processeur FUP3.

```
/JOB INTEG, : S, D2
/CALL FUP3              << MONTER BIDRIP SUR HR
/FREST, HR, D2
/FREST, HR, D2          << MONTER GEDRIP SUR HR
/EOJ
```

8.4.2 - Définition des tailles des buffers BUFCD et BUFSYS

Les tailles des buffers définies en standard dans BIDRIP sont de 256 mots. Si ces tailles ne conviennent pas à l'utilisateur ce dernier doit les redéfinir avant de lancer la génération proprement dite.

a) définition de BUFCD

II suffit d'écraser dans BIDRIP le binaire de BUFCD standard par l'assemblage de la séquence suivante :

```
          ENT BUFCD
          TABLE BUFDIS
LGBUCD : VAL n
          WORD LGBUCD
BUFCD  : DZS LGBUCD
          END
```

où n peut être 0 (si on ne veut pas de bufferisation)
ou un multiple de 128 (taille d'un secteur)

Le "job" d'assemblage étant :

```
/JOB  BUFCD , : S, D2
/CALL ASM
/SI   CR
/BO   BUFCD . BIDRIP
/IASM
/EØJ
```

b) définition de BUFSYS

Il suffit d'écraser dans BIDRIP le binaire de BUFSYS standard par l'assemblage de la séquence suivante :

```
            ENT     BUFSYS
            TABLE   BUFSY
  LGBUSY :  VAL     n
            WORD    LGBUSY
  BUFSYS :  DZS     LGBUSY
            END
```

où n est un nombre quelconque supérieur à 256

Le job d'assemblage étant :

```
/JOB    BUFSYS , : S, D2
/CALL   ASM
/SI     CR
/BØ     BUFSYS . BIDRIP
/IASM
/EØJ
```

8.4.3 - Production d'une version de DRIP16

Il suffit sous RBOS/D, de donner les deux commandes suivantes :

```
/EØJ

/CC { DRLP
    | DRCD
    | DRLPCD  } . GEDRIP - : S, D2
    | DRLPSY
    | DRCDSY
    | DRIP16 }
```

pour obtenir un fichier, nommé comme l'article qui a servi à le créer, contenant le binaire translatable prêt à être mis en oeuvre, d'une version "à la carte" ou de la version complète de DRIP16.

Remarques :

. DRLP est une version de DRIP16-A avec un dialogue étendu
. DRIP16 . GEDRIP génère une version identique, aux tailles de buffers près, à la version complète DRIP16-B livrée.

9 - MISE EN OEUVRE ET UTILISATION DE DRIP16

9.1 - MISE EN OEUVRE

Celle-ci nécessite un binaire translatable de DRIP16 (bande DRIP16-A, bande DRIP16-B ou encore version générée par l'utilisateur). Elle consiste à recopier ce binaire dans un article de la bibliothèque système : BIBSYS : S à l'aide de l'éditeur de liens EDILE. Le nom de cet article doit obligatoirement commencer par les caractères : SDR ; on peut prendre par exemple : SDRIP16.

La mise en oeuvre est donc réalisée sous BOS-D à l'aide des commandes suivantes :

```
* EØJ
* CALL      EDILE
* BI        HR                    < ou BI fichier
* BØ      : SDRIP16 . BIBSYS - : S
* LØ      ZE
* ILNK
* ELNK
* EØJ
```

9.2 - UTILISATION

La mise en œuvre décrite précédemment ayant été réalisée, il faut intégrer DRIP16 en mémoire avant le lancement du programme à tester.

9.2.1 - Sous BOS-D

9.2.1.1 - L'intégration

L'intégration sous BOS-D peut se faire de deux manières :

1. Par la commande OPTION

```
*  EOJ
* ZCONF
*  OPTION  DRIP16
*  CONF
```

DRIP16 est alors un service du système jusqu'à la prochaine commande ZCONF ou INIT, n.

2. A l'intérieur d'un job par la commande LOAD

```
*  JOB    ESSAI  ,  --
*   U5   LP
*  LOAD  ESSAI  ,  DRIP16        << chargement programme
                                 << et intégration DRIP16
*  DEGR  ,  L1  ,  L2            << passage paramètres
*  START                         << lancement programme
```

9.2.1.2 - L'utilisation en monoprogrammation

dans un contexte "batch" ou interactif.

Un "job" d'essai de programme s'écrit :

```
* JOB    ESSAI 1,—,—
* U5 LP                          << affectation U5 à l'imprimante
* ONTR
* DEGR, L1, L2                   << définition de la plage imprimante
                                 << plage disque vide
* RUN ESSAI
* EOJ
```

Un autre essai peut s'enchaîner :

```
* JOB    ESSAI 2,—,—
* U5 LP
* ONTR
* DEGR, L11 L22
* RUN ESSAI
* EØJ
```

ou encore, avec l'utilisation du disque :

```
* JOB —,—,—
* U6 D3
* DFIL, 50                       << création des fichiers sur D3
* ONTR
* DEGR, O, O, D1, D2
* RUN ESSAI
* DISC, R, ESPIO1 - PW
* DISC, A, ESPIO2 - PW
* EØJ
```

L'utilisateur dispose alors de 2 fichiers contenant des informations sur le déroulement de son programme ; il pourra les dépouiller selon les critères qu'il veut, en fonction des résultats obtenus après exécution de ESSAI. Le fichier ESPIO1 - PW peut être vide s'il n'y a pas eu de basculement effectué par DRIP16.

L'utilisation simultanée de l'imprimante et du disque s'écrit :

```
* JOB —,—,—
* U5 LP
* U6 D3
* DFIL, 50
* DEGR, L1, L2, D1, D2
* ONTR
* RUN ESSAI
* DISC, R, ESPIØ3 - PW
* DISC, A, ESPIØ4 - PW
* EØJ
```

L'utilisateur dispose ainsi d'informations éditées sur imprimante qu'il peut éventuellement compléter par le dépouillement de celles qui se trouvent dans les fichiers ESPIO3 et ESPIO4.

Remarque :

Dans un contexte de monoprogrammation «batch», c'est-à-dire non interactif, l'utilisateur ne peut récupérer que 2 fichiers ; il a donc intérêt à leur donner une taille assez grande par la commande DFIL s'il ne veut pas que les extractions réalisées au début de l'exécution du programme soient écrasées par les dernières.

Par contre en mode interactif, l'utilisateur a la possibilité de faire un «appel opérateur» lorsque le message «FILE n FULL» apparaît, de déconnecter le fichier plein par la commande DISC, et de relancer l'exécution par CONT.

9.2.1.3 — L'utilisation en multiprogrammation

L'enchaînement des commandes à DRIP16 dans un tel contexte peut être :

```
* INIT, n                          << lancement vacation dans
                                   << lequel DRIP16 est intégré
* MULTI                           << définition du contexte
* U5   LP
* U6   D3
* DFIL, 50
* DEGR, O, O, 200, 255            << espionnage de la plage 200, 255

  lancement application
→   FILE   1  FULL
* DISC, R, FIC1 - PW

→   FILE   2  FULL
* DISC , R, FIC2 - PW

* DEGR, O, O, O, O
* DISC, A, FIC3 - PW              << Fin de vacation
                                   << cataloguage du fichier actif
* INIT, m
```

9.2.2 — Sous BACKM

L'intégration sous BACKM se fait en même temps que le chargement du programme à tester à l'aide de la commande LOAD. DRIP16 ne fait partie intégrante du système que pendant le temps où le programme à tester est en mémoire.

Le «job» d'essai d'un programme est alors :

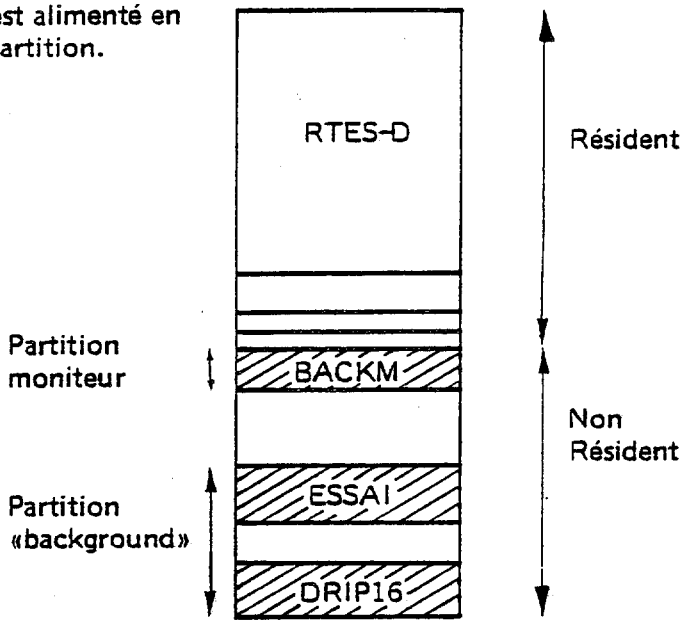
```
JØB   ESSAI , — , —
U5    LP
LØAD  ESSAI, DRIP16              << chargement programme
                                 << et intégration DRIP16
DEGR, L1, L2                     << Passage des paramètres
START                            << lancement du programme
```

```
DEGR , L11, L22
START                    << 2ème essai du programme
                         << avec d'autres paramètres
                         << de tests

EØJ
```

La carte mémoire se schématise ainsi :

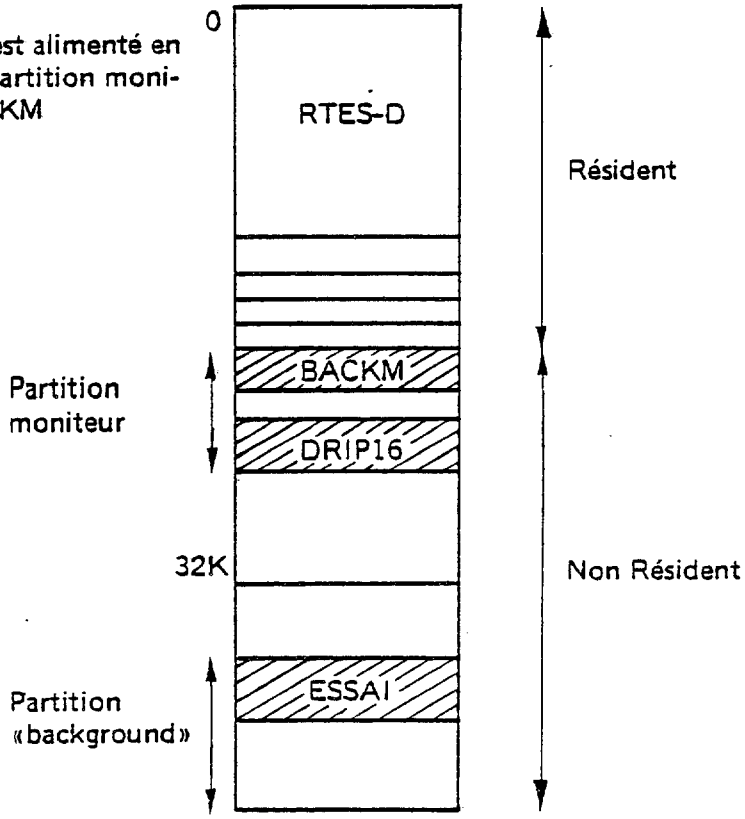
a) La partition «background» est en-deça de 32K :

DRIP16 est alimenté en fond de partition.



b) La partition «background» est au-delà de 32K :

DRIP16 est alimenté en fond de partition moniteur BACKM

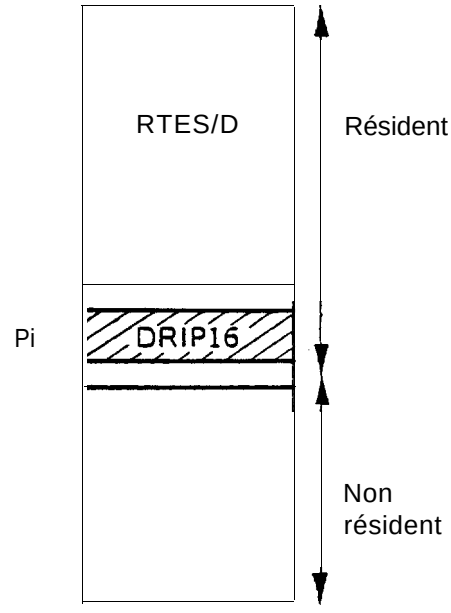


9.2.3 - Sous RTES-D et RTES-C

9.2.3.1 - Intégration

L'intégration sous RTES-D ou RTES-C se fait, comme tout processeur foreground, par la commande EXEC. Elle demande, au préalable, la constitution par le BUILDER d'une image mémoire de DRIP16 à l'adresse de la partition qui lui a été réservée. Si le nom donné à cette image mémoire est DRIP16-IM, la commande d'intégration de DRIP16 dans la partition Pi est :

```
* EXEC, R, DRIP16-IM, Pi

* MULTI                 < < DRIP16 est employé
                        < < dans un contexte de
                        < < multiprogrammation
  autres intégrations

* SAVE..
```



9.2.3.2 - Utilisation

Nous donnons ici quelques séquences de commandes à DRIP16 sous le dialogue opérateur de RTES-D ou RTES-C.

```
* INIT, n                 << lancement système dans lequel
                          << DRIP16 est intégré
* MULTI                   << définition du contexte
* U5LP
* U6 D3
* DFIL, 50
* DEGR, 0, 0, 200, 255    << La plage 200, 255 est à espionner

   lancement de l'application

   "FILE 1 FULL"
* DISC. R, FIC1 - PW


   "FILE 2 FULL"
* DISC, R, FIC2 - PW

* DEGR, 0, 0, 0, 0
* DISC, A, FIC3 - PW      << Fin de session
                          << catalogage du fichier actif
* INIT, m
```

Remarques

1. Pour arrêter correctement les extractions sur disque demandées par une application, il faut émettre les commandes suivantes :

```
* DEGR, 0, 0, 0, 0          << pour arrêter les
                            << extractions sur disque
* DISC, R, FIC1 - PW        << pour récupérer les
                            << dernières extractions
* DISC, A, FIC2 - PW        << écrites sur disque
```

2. Pour relancer correctement les extractions sur disque d'une application il faut d'abord les arrêter correctement par les 3 commandes précédentes et ensuite émettre les commandes suivantes :

```
* U6 FU disque
* DFIL, taille
* DEGR, L1, L2, D1, D2      << pour relancer les
                            << extractions sur disque
```

3. En cas d'erreur intervenant pendant l'écriture des extractions sur fichier disque (oubli de la commande MULTI, FU saturée etc...), l'opérateur doit émettre les commandes vues précédemment pour arrêter et éventuellement relancer les extractions sur disque.

9.2.4 - Sous MPES

9.2.4.1 - Intégration

L'intégration sous MPES se fait comme tout processeur foreground, par la commande EXEC passée sous le dialogue opérateur suivie de la commande MULTI, propre à DRIP16. Elle demande, au préalable, la constitution par le builder d'une image mémoire de DRIP16 à l'adresse de la partition qui lui a été réservée. Si le nom donné à cette image mémoire est DRIP16 - IM, les commandes d'intégration de DRIP16 dans la partition Pi sont :

```
* EXEC, R, DRIP16 - IM, Pi
* MULTI


* SAVE - - -
```

9.2.4.2 - Utilisation dans l'environnement batch

a) au niveau dialogue opérateur

II faut affecter l'unité symbolique U6 à l'unité fonctionnelle disque choisie comme support des extractions par la commande :

```
DUMB 0, 15, Fu
```

b) au niveau de l'utilisateur

Les commandes à DRIP16 sont précédées de la clé CDES.
Un "job" d'essai de programme avec extractions sur imprimante s'écrit :

```
/JOB - - -
/U5      LO
/CDES ONTR
/CDES  DEGR,  L1,  L2
/RUN  ESSAI
/EOJ
```

Un "job" d'essai avec extractions sur imprimante et disque s'écrit :

```
/JOB - - -
/U5      LO
/U6      D6
/CDES ONTR
/CDES  DEGR,  1,  80,  1,  255
/CDES  DFIL,  50
/RUN  ESSAI
/CDES  DISC,  R,  ESPIO1  -  PW
/CDES  DISC,  A,  ESPIO2  -  PW
/EOJ
```

L'utilisateur dispose alors de deux fichiers contenant toutes les informations relatives au déroulement de son programme. II peut dans un "job" complémentaire dépouiller ces fichiers par PRODEP pour éventuellement compléter les informations de degrés 1 à 80 dont il dispose sur l'imprimante. Le fichier ESPIO1 - PW peut être vide s'il n'y a pas eu de basculement effectué par DRIP16.

Le "job" précédent peut aussi s'écrire de la manière suivante :

```
/JOB
/U5      LO
/U6      D6
/CDES ONTR
/CDES  DEGR,  1,  80,  1,  255
/CDES  DFIL,  50
/CONNECT  D,  PRODEP-NN,  D5,  2,  4
/RUN  ESSAI
/EOJ
```

Le fichier PRODEP-NN connecté par la commande CONNECT contient les commandes nécessaires au dépouillement des fichiers créés précédemment. Ce "job" est exécuté en superprioritaire immédiatement derrière le "job" d'essai. II s'écrit :

```
/JOB   PRODEP,  PW,  D5,  MPES,  1
/CDES  DISC,  R,  ESPIO1  -  PW
/CDES  DISC,  A,  ESPIO2  -  PW
/
```

dépouillement

9.2.4.3 - Utilisation par les tâches de l'application

Toutes les commandes sont passées au niveau du dialogue opérateur.
L'affectation de U6 à l'unité fonctionnelle supportant les fichiers disque est réalisée par les commandes suivantes :

```
DUMB 0, 15, Fu    pour l'environnement système
DUMB 2, 15, Fu    pour l'environnement tâches temps réel
```

Pour l'utilisation de DRIP16 se reporter au § 9.2.3.2 concernant RTES D :

9.3 - CONSEILS D'UTILISATION

9.3.1 - Intégration des différentes versions de DRIP16 dans BIBSYS

L'utilisateur a intérêt à intégrer toutes les versions de DRIP16 dont il est possesseur dans la bibliothèque : BIBSYS en leur donnant un nom significatif :

```
DRIPA     pour la version minimum DRIP16 - A
DRIPB     pour la version complète DRIP16 - B
```

et éventuellement :

```
DRLP
DRCD
DRLPCD    pour les versions "à la carte"
DRLPSY
DRCDSY
DRIPB     pour la version complète possédant des buffers définis par l'utilisateur
```

(les 2 premières lettres doivent être DR).

II peut ainsi à tout moment et selon son contexte de travail intégrer la version de DRIP16 de son choix, à la manière dont elle a été décrite dans les paragraphes précédents.

9.3.2 - Mise au point. de processeurs sous BOS-D, BACKM ou MPES

Dans ce cas, l'utilisation du disque parait facultative. En effet, l'utilisateur se servira surtout de l'imprimante pour réaliser les extractions concernant une certaine phase d'avancement de ses tests. II peut, le cas échéant, se servir du disque pour réaliser systématiquement les extractions des phases antérieures au cas où il en aurait besoin pour l'interprétation de celles sorties sur imprimante.

Les versions DRIP16 - A ou DRLPCD sont alors conseillées.

9.3.3 - Mise au point d'applications sous BOS-D, RTES-D ou RTES-C

Dans ce cas, l'utilisation de toutes les ressources de DRIP16 parait souhaitable dans les premières phases de la mise au point.
Dans les phases d'évaluation de l'application ou d'exploitation de celle-ci la ressource disque devient la seule vraiment efficace pour assurer une perturbation minimum ; la ressource imprimante doit alors être réservée seulement aux éditions de messages prioritaires concernant l'état de l'application et permettant une réaction appropriée de la part de l'opérateur. Le dépouillement des fichiers intervient en "background" et permet un suivi "on line" de l'application pour en contrôler les pannes ou le bon fonctionnement.

Les versions DRIP16 - B ou DRCDSY paraissent dans ce cas bien adaptées.

9.3.4 - Mise au point simultanée de processeurs et d'applications

Celle-ci est rendue possible par la présence simultanée d'un module DRIP16 dans la sphère "foreground", pour la mise au point de l'application, et d'un module DRIP16 dans la sphère "background", pour la mise au point d'un processeur (ou d'une tâche avant son intégration).

Afin d'avoir une occupation minimale des modules de mise au point, nous conseillons l'utilisation de DRCDSY en "foreground" et de DRIP16-A en "background".

Dans le cas d'une utilisation de la ressource disque dans la sphère "background", l'utilisateur doit respecter la politique de partage des périphériques préconisée par RTES-D afin que les fichiers de DRIP16 soient dans des FU différentes pour les deux sphères.

10 - LES MESSAGES D'ERREURS DE DRIP16

Les messages d'erreur ont la forme suivante :

ERDRIP nn 'XXXX PTY = mmm

ils spécifient toujours un numéro d'erreur nn
et, éventuellement, pour certains numéros d'erreur :

. le compte rendu de FMS
. la priorité de l'appelant qui a provoqué l'erreur

Ces messages sont imprimés sur l'unité symbolique EL.
Les numéros d'erreurs possibles sont répertoriés dans le tableau suivant :

n °	Nature de l'erreur	Exemple	Action à faire
01	Taille du buffer système, BUFSYS, insuffisante pour récupérer les informations système	La liste des opérations sur horloge dépasse la taille de BUFSYS	Regénérer DRIP16 avec un buffer plus grand
02	Refus de la part de FMS d'écrire une extraction sur le fichier actif	- l'utilisateur n'a pas donné la commande DFIL - en multiprogrammation la commande MULTI n'a pas été donnée	. en monoprogrammation repasser le "job" après correction . en multiprogrammation l'application n'est pas arrêtée, mais la plage disque est invalidée. Envoyer les commandes MULTI DFIL --- DEGR ---
03	Sur la commande le fichier à déconnecter est inexistant	DRIP16 ou système défaillants	Interpréter le compte rendu de FMS. Si nécessaire regénérer l'application
04	Réservé		
05	Réservé		
06	Paramètre invalide dans une commande	- DEGR, L1, L2 avec . .L1>L2 . L1 ou L2 > 255 - DEGR, L1, L2, D1, D2 alors que MODCD est absent - ONTR, 10 en monoprogrammation	Refrapper la commande
07	Sur la commande DFIL erreur détectée par FMS à la création des fichiers	- tables saturées - FU saturée	Interpréter le compte rendu de FMS. Si nécessaire regénérer l'application
08	Erreur de syntaxe dans une commande	DEGR 10, 20 manque une virgule DEGR, 10 ;	Refrapper la commande

11 - LE PROCESSEUR DE DEPOUILLEMENT PRODEP

11.1 - PRESENTATION

Ce processeur, indépendant du module DRIP16, est chargé du dépouillement en différé des fichiers permanents écrits par DRIP16 et déconnectés par la commande DISC. Pour cela, il réalise une lecture séquentielle de ces fichiers et opère une sélection des enregistrements en fonction des critères de recherche donnés par des commandes avant son lancement. Les enregistrements sélectés sont décodés, formatés et listés sur l'unité symbolique U5

Les critères de recherche sont :

- sélection par priorité
- sélection par degré ou plage de degré
- sélection par nature d'information système
- sélection par tranche horaire

1 1.2 - APPEL DE PRODEP

Le fichier image mémoire, PRODEP- : S, ayant été créé sur D2 à l'aide du binaire translatable fourni, l'appel du processeur de dépouillement est réalise par la commande :

```
CALL PRODEP
```

II est ainsi alimenté en mémoire dans son état initial en même temps qu'il passe au superviseur un certain nombre de clés par la requête CAMO.

11.3 - CRITERES DE RECHERCHE

Ils sont passés en paramètres de PRODEP par l'intermédiaire de commandes. Par souci de conformité avec DRIP16, les critères de sélection des extractions sont les mêmes (sauf un) que ceux de DRIP16. Les commandes correspondantes sont donc les mêmes si ce n'est la dernière lettre qui est remplacée par la lettre X.

Ces critères sont :

- sélection par priorité

réalisée par les commandes :

```
ONTX [ Pi, Pj, Pk . . . ]
```

et

```
OFTX [ Pi, Pj, Pk . . . ]
```

équivalentes aux commandes ONTR et OFTR de DRIP16.

- sélection par degré ou plage de degré

réalisée par la commande :

```
DEGX,  L1,  L2  [,P]
```

qui définit la plage [L1, L2] pour la priorité P.
Lorsque P est absent la plage L1, L2 est commune à toutes les priorités (tout comme la commande DEGR de DRIP16).

- sélection par nature d'information système

réalisée par les commandes :

```
ONSX [ , Ni, Nj, Nk . . . ]
```

et

```
OFSX [ , Ni, Nj, Nk . . . ]
```

équivalentes aux commandes ONSYS et OFSYS de DRIP16.

- sélection par tranche horaire

réalisée par la commande :

```
HIST,  H1,  M1,  H2,  M2
```

qui permet de ne s'intéresser qu'aux extractions réalisées entre H1 heures, M1 minutes et H2 heures, M2 minutes.

L'utilisateur peut combiner tous ces critères selon l'intérêt qu'ils présentent pour lui.

L'état initial de PRODEP est ainsi défini :

. tous les appels "locaux" sont valides pour toute priorité
. tous les appels "système" sont inhibés
. la plage des degrés valides est [1, 255] pour toute priorité
. la tranche horaire est égale à 24 heures.

Faute de commandes vues précédemment, cet état décrit donc les paramètres du dépouillement.

11.4 - LANCEMENT DU DEPOUILLEMENT

Les paramètres du dépouillement ayant été définis par les commandes précédentes ou laissés dans l'état initial si aucune commande n'est donnée, le lancement de PRODEP s'effectue par la commande :

```
EXAM , FICNAM -PW
```

où FICNAM - PW est le nom du fichier à dépouiller.

Ce fichier doit être sur l'unité fonctionnelle disque affectée à l'unité symbolique U6

Le dépouillement peut être interrompu à tout instant par un appel opérateur ; l'utilisateur peut ainsi donner de nouveaux critères de sélection avant de relancer et terminer le dépouillement par la commande :

```
CXAM
```

11.5 - FORMAT DES INFORMATIONS DEPOUILLEES

Le format des enregistrements listés par PRODEP est identique à celui des enregistrements listés par le module imprimante de DRIP16 si ce n'est :

. les caractères ⊖ de la partie droite sont remplacés par des ⊛

. l'heure où l'extraction a eu lieu spécifie les millisecondes.

. les dumps des files du scheduler et des états des événements sont effectués en format binaire : le rang d'un bit à 1 dans une file est imprimé en clair ; on obtient ainsi les numéros de priorité des tâches armées, des tâches non masquées, et des tâches prêtes ainsi que les numéros des événements dans l'état SET.

De plus, l'utilisateur peut par la commande :

VISU

obtenir des enregistrements de 72 caractères adaptés à la taille d'un écran de dispositif de visualisation alphanumérique.

11.6 - LES ERREURS

Les messages d'erreur ont la forme suivante :

ERDEP nn 'XXXX

éventuellement

où

. nn est le numéro d'erreur

. 'XXXX est le compte rendu de FMS dans le cas d'une erreur intervenant à l'ouverture ou à la lecture du fichier à dépouiller.

Ces messages sortent sur l'unité symbolique EL.
Les numéros nn possibles sont :

01	:	paramètre invalide
Exemple	:	HIST, 10, 40, 9, 50
		L'heure de début est supérieure à l'heure de fin
02	:	erreur de syntaxe
Exemple	:	DEGX , 10 ;
03	:	erreur FMS sur OPEN, READ ou CLOSE du fichier.

ANNEXE

I - SEQUENCE D'APPEL AU MODULE DRIP16

1 - SEQUENCE D'APPEL

	PSR	A, B, X, Y	< SAUVEGARDE CONTEXTE
	PSR	C, L, W, K	< APPELANT
	SVC	DRIP16	< SVC 7
	Degré de l'appel	Type appel	
	n° ligne de l'appel		
INFOS	Informations spécifiques du type de l'appel		
	PLR	C, L, W, K	< RESTAURATION CONTEXTE
	PLR	A, B, X, Y	< APPELANT

Remarque

Cette séquence d'appel détruit les indicateurs V et C du registre S.

2 - TYPES D'APPELS

Le paramètre type d'appel spécifie l'extraction que doit réaliser DRIP16. II est directement lié à l'instruction d'appel du module DRIP16.

II existe :

1. Passage par label
2. Entrée de procédure
3. Sortie de procédure

Les informations spécifiques de ces types d'appel sont la chaîne ASCII du nom du label ou de la procédure :

INFOS	nb de mots pour le nom	1er caractère du nom
	(n - 1)^{e} caractère	n ^{e} caractère

Exemple : Pour signaler l'entrée dans la procédure ECRIRE les informations spécifiques de l'appel n° 2 seront :

4	E
C	R
I	R
E	

} 4 mots

④ Dump de zone (SNAP)

dont les informations spécifiques sont :

INFØS

nb de mots pour le nom	1er caractère nom de zone
(n - 1)^{e} caractère	n^{e} caractère
adresse de la zone	
nb de mots à dumper	
Format	

chaîne ASCII = nom de la zone

Le format peut prendre les valeurs suivantes :

1 : format hexadécimal
2 : format ASCII
3 : format décimal

Remarque

Dans le cas où l'adresse de la zone à visualiser est dans le registre RA

. le nom de la zone est RA
. le paramètre adresse est égal à 0
. le paramètre format a le bit 0 positionné à 1.

⑤ Dump des registres (RSNAP)

pas de paramètres spécifiques

⑥ Edition de message (XPRINT)

les informations spécifiques sont identiques aux types 1, 2 et 3, la chaîne ASCII étant ici le message.

⑦ Inhibition dynamique des appels 1, 2 et 3 (SILENCE)

pas de paramètres

⑧ Validation dynamique des appels 1, 2 et 3 (NOSILENCE)

pas de paramètres

(9) Dump des files du "scheduler"

(10) Etat des événements

(11) Phase d'avancement de l'application

(12) Occupation des partitions

(13) Taux d'attente des partitions non résidentes

(14) Taux d'occupation des zones dynamiques du système

(15) Portion de la Z.D.R.

} Sans paramètres

avec pour paramètres :

INFØS

nb de paramètres = 2
Numéro du 1er mot de la ZDR à dumper
nombre de mots

(16) Etat d'un périphérique avec pour paramètres :

INFØS

nb de paramètre = 1
n° de FU ou SU

(17) Bloc de contrôle d'une tâche avec pour paramètres :

INFØS

nb de paramètre = 1
n° de priorité

(18) Liste des opérations différées et périodiques

(19) Liste des tâches en attente d'événement

(20) Liste des ressources utilisateur

(21) Liste des fichiers ouverts (tables de FMS)

} pas de paramètres

(22) Dump d'une zone système (d'adresse absolue)

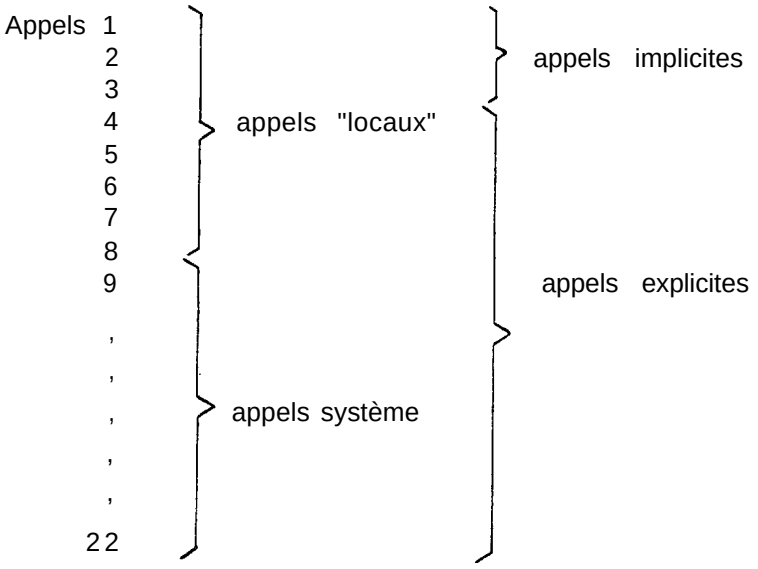
avec pour paramètres

INFØS

nb de paramètres = 2
adresse absolue de la zone
nombre de mots

Les appels de type 1, 2 et 3 sont générés implicitement par le compilateur. Les autres sont générés explicitement sur des instructions spécifiques.

On peut adopter la nomenclature suivante :



II - DESCRIPTION D'UNE METHODOLOGIE D'EMPLOI DE DRIP16

1 - INTRODUCTION

Une bonne utilisation d'un outil d'aide à la mise au point tel que DRIP16 suppose de la part de l'utilisateur le choix d'une méthodologie de test adaptée à son projet. Celle-ci doit être fixée dès l'analyse du projet car elle sera mise en oeuvre dès la programmation de celui-ci jusqu'à sa maintenance comprise.

Nous proposons ici une méthodologie possible, ce n'est pas la seule.

Elle est basée sur :

. les possibilités de compilation optionnelle du langage PL1600.
Nous utiliserons ces possibilités en ayant pour objectif un vidage aisé du programme de ses points de tests et non pas celui de définir par compilation des NIVEAUX dans la progression des tests, méthode qui nous semble lourde car elle nécessite une recompilation à chaque changement de phase de test.

. la notion de DEGRE introduite par DRIP16 qui permet de contrôler dynamiquement le flot des extractions suivant l'état d'avancement des tests. On peut ainsi définir

- des NIVEAUX de mise au point

Ex : plage 10 à 20 pour une phase de test
20 à 30 pour la suivante
etc . . .

- des PRIORITES au niveau des extractions

Ex : plage 10 à 15 pour les données peu prioritaires
15 à 20 pour les données prioritaires
pour la plage 10 à 20

. la notion de NATURE introduite par DRIP16 qui permet une classification des informations "système" (commandes ONSYS, OFSYS)

. une optimisation des points de tests afin de perturber au minimum le programme à tester au point de vue de

- sa taille grâce à une optimisation du volume des extractions
- son temps d'exécution grâce à une optimisation de la fréquence des extractions.

2 - DEFINITIONS

L'ANALYSE d'un projet logiciel conduit à le découper logiquement en un ensemble de MODULES qui sont eux mêmes découpés en unités de programmation. Chaque module est caractérisé par sa fonction et son interface (vecteur d'entrée, vecteur de sortie) ; il en est de même pour chacune des unités de programmation. A cette découpe logique en unités correspond en général une découpe physique ou unités de compilation.

LA REALISATION d'un projet consiste en deux phases :

- PROGRAMMATION et ESSAIS de chaque unité séparément ; chaque programmeur est indépendant
- INTEGRATION progressive de tous les composants au projet final.

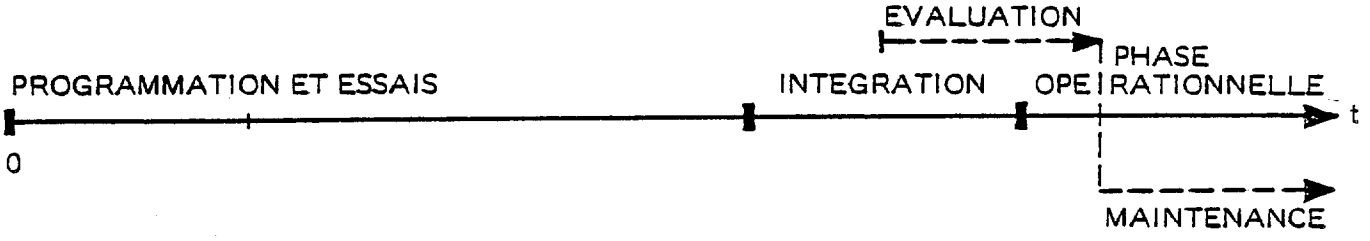
Dans un contexte de MONOPROGRAMMATION, l'intégration se fera par édition de liens ou recompilation après réunion des textes sources.

Dans un contexte de MULTIPROGRAMMATION, l'intégration se fera par addition d'une nouvelle tâche à l'application en cours de mise au point.

Les différentes PHASES d'avancement d'un projet peuvent se résumer en :

- PHASE INITIALE de programmation et essais de chaque unité
- PHASE D'INTEGRATION des modules et d'EVALUATION de certains paramètres (taille de kstores, de zones dynamiques pour RTES-D, partitionnement mémoire sous RTES-D . . . . .).
- PHASE OPERATIONNELLE qui est validée par l'évaluation précédente
- MAINTENANCE

On peut les représenter ainsi dans le temps :



3 - DESCRIPTION

① Les sous plages de degrés

Nous proposons la constitution d'une GRILLE des degrés de mise au point. Il s'agit de découper la plage [1 , 255] des degrés d'appel possibles en autant de sous plages qu'il y a de modules participant au projet en ajoutant :

- une sous-plage pour la phase d'intégration et d'évaluation
- une sous-plage pour la phase opérationnelle éventuellement

De cette manière :

. Chaque responsable de module dispose d'une sous-plage pour la mise au point de son module.

. Pendant la phase d'intégration, on dispose d'une sous-plage pour tester l'enchaînement des modules, de plus, on peut par une simple commande validant sa plage, relancer les extractions de tel ou module qui se montre défaillant dans cette phase.

. on dispose d'une sous-plage pour réaliser les extractions nécessaires à l'évaluation du programme

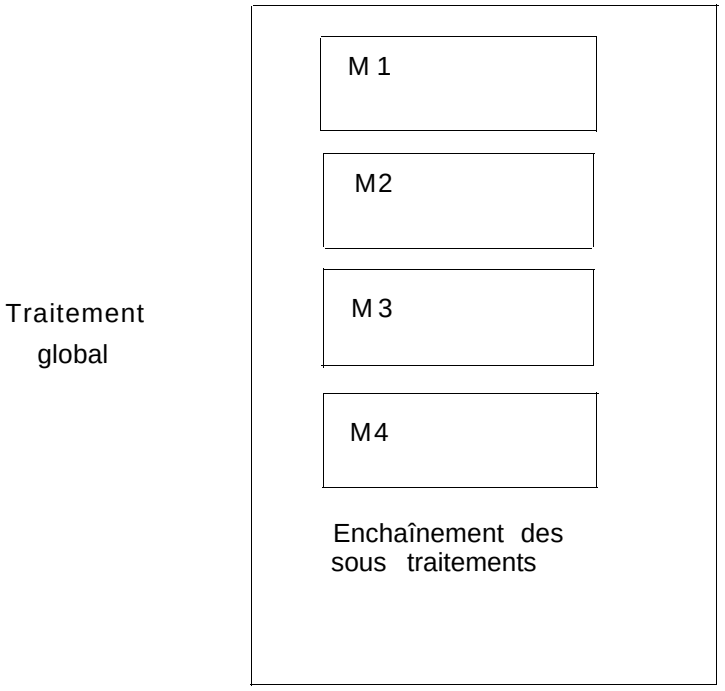
. on dispose enfin d'une sous-plage pour réaliser certains espionnages en phase opérationnelle.

Il n'y a pas de sous-plage réservée à la maintenance ; en effet celle-ci valide la ou les sous-plages qui concernent les extractions qu'elle veut réaliser.

Les limites des sous-plages doivent être adaptées au projet et ensuite respectées. Elles peuvent se chevaucher ou ne pas exister ; on peut ainsi prévoir les mêmes tests en phase d'intégration et phase opérationnelle ou ne prévoir aucun test dans cette dernière.

Exemple :

Soit un programme réalisant un certain traitement par l'appel de 4 sous traitements :



On découpe la plage 1 à 255 de la manière suivante :

```
  1 à  40   pour le module 1
 41 à  80   pour le module 2
 81 à 120   pour le module 3
121 à 160   pour le module 4
161 à 200   pour le module d'enchaînement et la phase d'intégration
180 à 220   pour la phase d'évaluation
180 à 255   pour la phase opérationnelle
```

Dans la phase initiale, le réalisateur du module 3 donnera la commande

DEGR, 81, 120

Dans la phase d'intégration, les extractions seront validées par

DEGR, 161, 200

si le dépouillement de ces dernières le demande on pourra relancer les tests du module 2 par

DEGR, 41, 80

ou tous les tests par

DEGR, 1, 200

L'évaluation du programme se commandera par :

DEGR, 180, 220

on constate que les extractions de degrés 180 à 200 concernent à la fois l'intégration et l'évaluation.

On pourra aussi dans cette phase faire :

DEGR, 161, 220

ou

DEGR, 1, 220 si l'on veut en savoir plus.

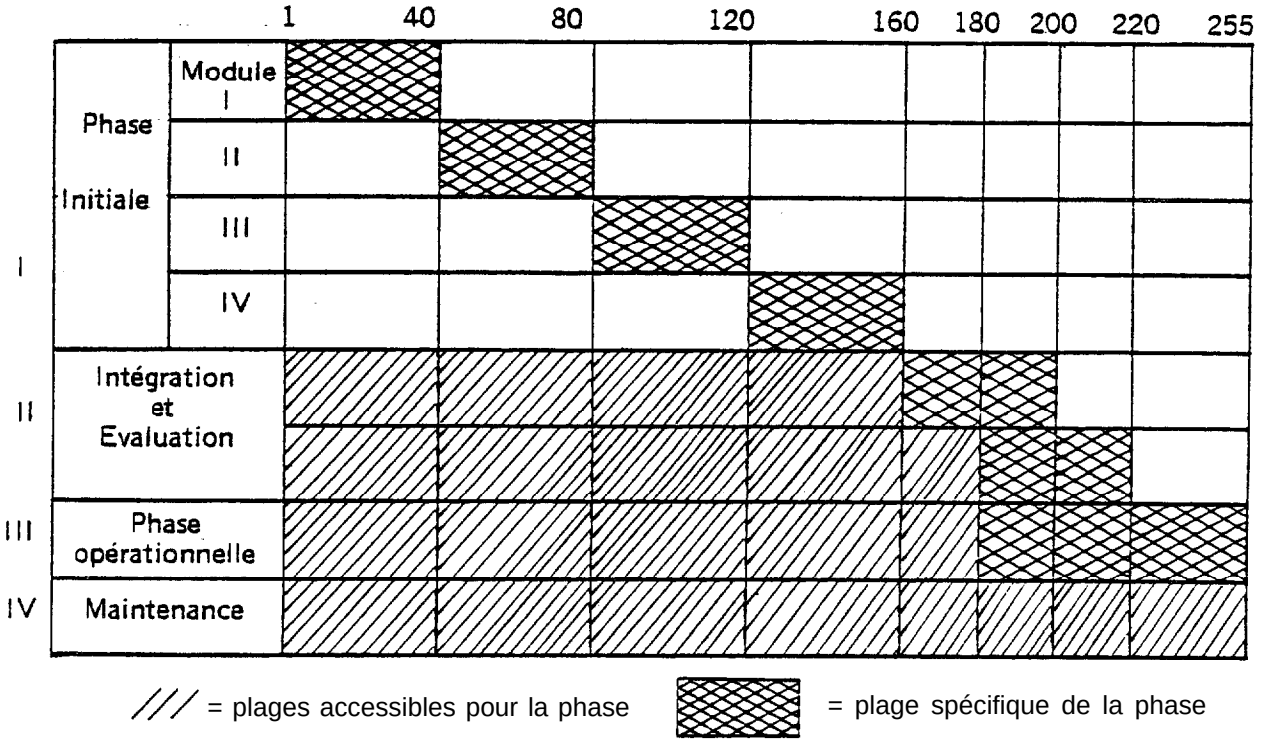
Dans la phase opérationnelle, les tests prévus sont lancés par

DEGR, 180, 255

cette phrase comporte des extractions propres (220 à 255) et des extractions des phases précédentes (180 à 220).

La phase de maintenance pourra, tout comme la phase opérationnelle, commander toute extraction.

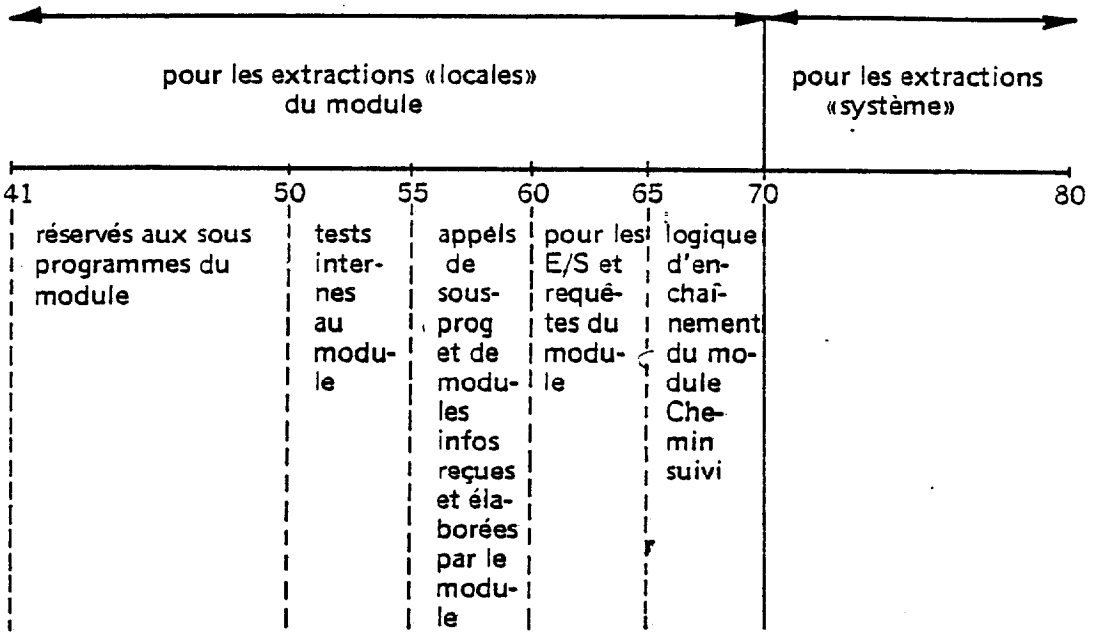
La grille des degrés pour ce programme serait :



② Découpage de la sous-plage

Chaque sous-plage doit être redécoupée pour la mise au point de chaque module.

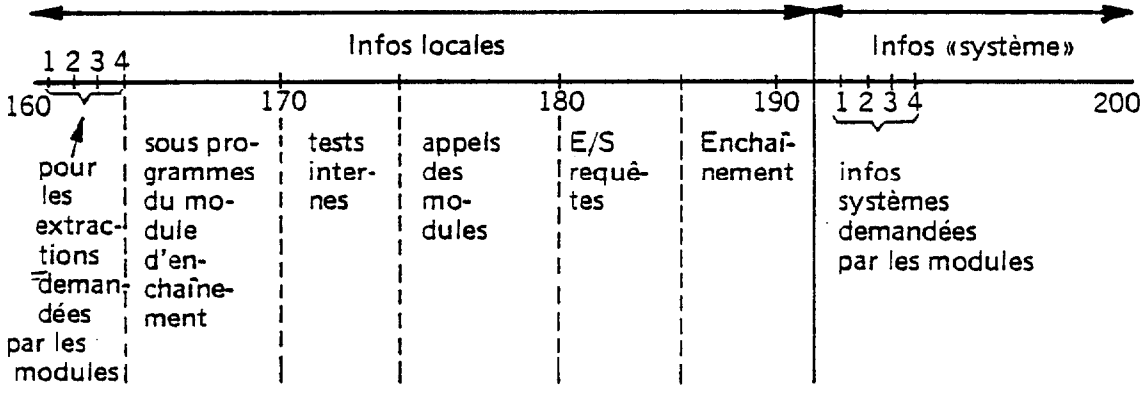
On peut proposer :



③ Communications entre phases

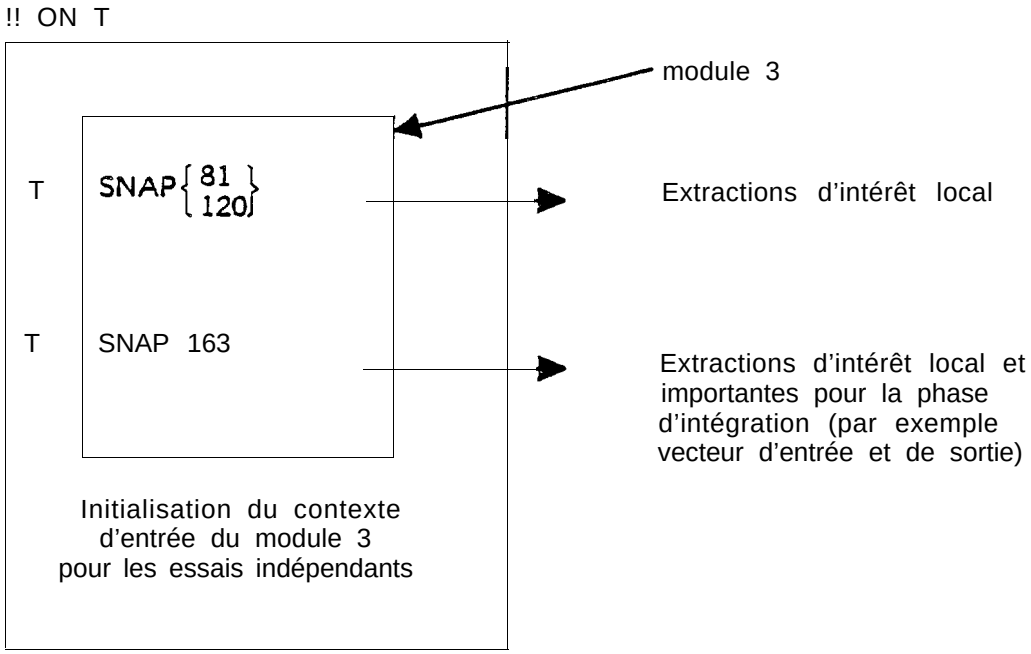
Un module peut détenir des informations ayant un intérêt non seulement pour lui-même, mais aussi pour une phase ultérieure d'avancement du projet (intégration ou phase opérationnelle) ; dans ce cas, le module devra demander une extraction de ces informations avec un degré appartenant à la sous-plage de cette phase ultérieure.

Ainsi les sous-plages de la phase d'intégration et de la phase opérationnelle devront comporter des degrés réservés aux modules : par exemple 1 par module.
Si on reprend notre exemple, la découpe de la sous-plage du module d'enchaînement serait :



De même pour la phase opérationnelle.

Ainsi le module 3 s'écrira d'une manière indépendante :



. pour la mise au point séparée on validera les degrés 81 à 255
. pour la phase intégration en validant- la plage qui lui est réservée (161 à 200) on autorise les extractions de degré 163.

④ La phase opérationnelle

Les tests effectués dans cette phase doivent être très optimisés pour être "transparents". II faut donc sélecter les informations importantes :

. les E/S et requêtes
. l'enchaînement des modules et les informations principales qui cheminent
. les informations système - à caractère prioritaire (degré 254)
- à caractère moins prioritaire
. les espionnages d'une application temps réel
- à caractère prioritaire (degré 255)
- à caractère moins prioritaire

Dans cette phase on fera :

DEGR, 254, 255, 181, 265

254, 255 : informations très prioritaires à sortir sur terminal d'édition pour avertir l'opérateur

181, 265 : suivie de l'application sur disque avec dépouillement différé si c'est nécessaire

(5) La compilation conditionnelle

Tous les projets ne nécessitent pas obligatoirement des tests en phase opérationnelle. Nous distinguerons 2 groupes :

Groupe 1 : les programmes qui, en fin d'évaluation, éliminent tous leurs appels à DRIP16.
Ces programmes ont 2 états :

- en cours de test
- en cours d'exploitation

Groupe 2 : les applications temps réel qui, en fin d'évaluation conservent des appels à DRIP16 (espionnages divers). Elles ont 3 états :

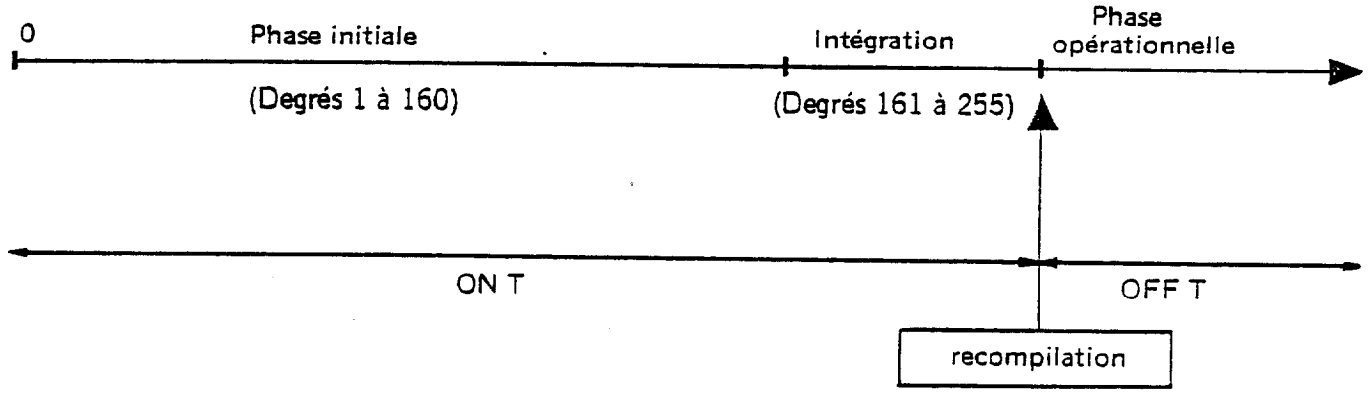
- en cours de test (beaucoup d'appels à DRIP16)
- sous surveillance (peu d'appels)
- sans surveillance (pas d'appels)

C'est au niveau de la compilation que l'on définit l'état d'un programme afin d'optimiser le code généré, suivant son état d'avancement.

Pour le groupe 1 on a une seule clé de compilation qui est ON ou OFF :

```
soit  [  !!   ON   T          << TESTS
soit  [  rien                 pas de tests

      T! TRACE---

      T! DEGRE---

      T    SNAP---

      T    RSNAP
```

ce qui donne au niveau des phases d'avancement :



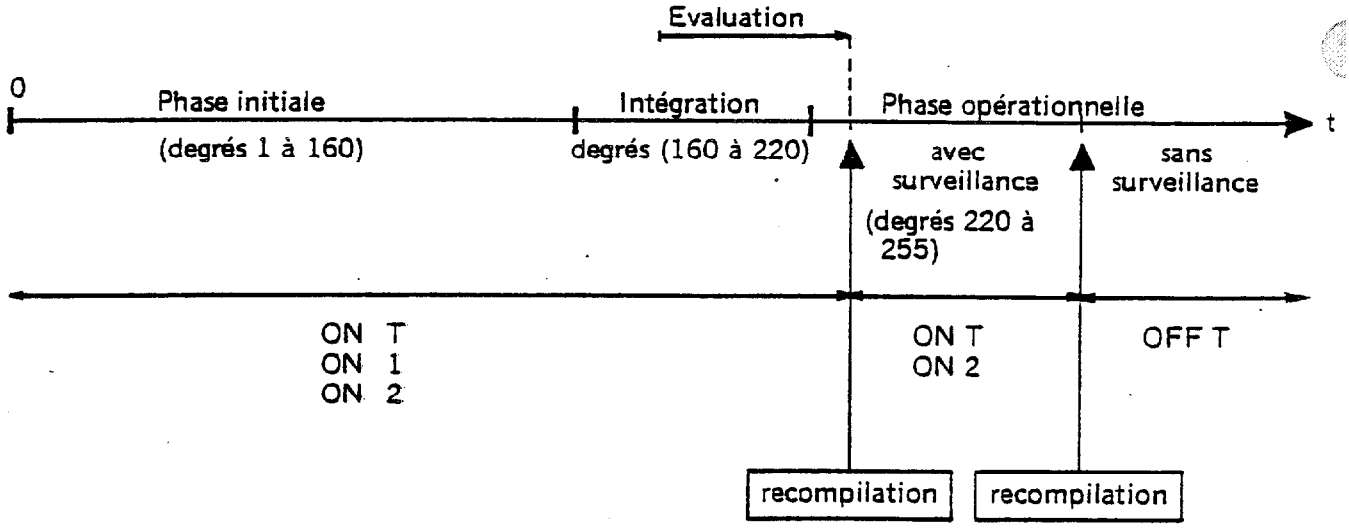
Il y a une seule carte à enlever pour donner un programme propre.

Pour le groupe 2 on a 3 clés de compilation utilisées ainsi :

```
carte 1        ! ! ON      T                    << TESTS
carte 2        T!    ON    1
               T!    ON    2
               1!  DEGRE  81

                     degré courant 81

               2 SNAP 163 ---                   info importante de surveillance
```

Il suffit d'enlever la carte 2 pour éliminer la génération des tests initiaux et n'avoir que la surveillance ; il suffit d'enlever la carte 1 pour éliminer tous les appels à DRIP16.

On a donc au niveau des phases d'avancement :



(6) La maintenance

Lorsque le produit passe en maintenance la documentation correspondante doit comporter :

- les dés de compilation
- les grilles de degrés au niveau projet et au niveau module
- les résultats des extractions obtenues en phase de développement

Le service de maintenance est alors capable de relancer n'importe quel test.

En conclusion il y a un compromis à trouver entre :

- un contrôle statique des extractions par compilation conditionnelle
- un contrôle dynamique des extractions par la notion de degré.

III - DESCRIPTION DE LA BIBLIOTHEQUE GEDRIP

1 - ARTICLE DRLP

```
/ JOB     DRLP, : S, D2
/ CALL    EDILE
/ LØ      ZE
/ BØ      DRLP-BT
/ BI      MCDOUT . BIDRIP          << PAS MODCD
/ ILNK
/ CLOSE BI
/ BI      MSYOUT . BIDRIP          << PAS MODSYS
/ CLNK
/ CLOSE BI
/ BI      LANTRA . BIDRIP
/ CLNK
/ CLOSEBI
/ BI      DIATRA . BIDRIP
/ CLNK
/ CLOSE BI
/ OPEN OLD 1, BIDRIP               << LINK DE
/ LLNK 1                           << CSTRAS ET MODLP
/ CLOSE 1
/ RLNK
/ ELNK
/ EOJ
```

2 - ARTICLE DRCD

```
/ JOB     DRCD , : S, D2
/ CALL    EDILE
/ LØ      ZE
/ BØ      DRCD-BT
/ BI      MLPØUT . BIDRIP          << PAS MODLP
/ ILNK
/ CLOSE BI
/ BI      MSYOUT . BIDRIP          << PAS MODSYS
/ CLNK
/ CLOSE BI
/ BI      LANTRA . BIDRIP
/ CLNK
/ CLOSE BI
/ BI      DIATRA . BIDRIP
/ CLNK
/ CLOSE BI
/ OPEN OLD 1, BIDRIP               << LINK DE
/ LLNK 1                           << CSTRA, MODCD
/ CLOSE 1                          << ET BUFCD
/ RLNK
/ ELNK
/ EOJ
```

3 - ARTICLE DRLPCD

```
/ JOB    DRLPCD , : S, D2
/ CALL EDILE
/  LØ    ZE
/ BØ     DRLPCD-BT
/ BI     MSYOUT . BIDRIP                 << PAS MODSYS
/ ILNK
/ CLOSE BI
/ BI     LANTRA . BIDRIP
/ CLNK
/ CLOSE BI
/ BI     DIATRA . BIDRIP
/ CLNK
/ CLOSE BI
/ OPEN OLD 1, BIDRIP                     << LINK DE
/ LLNK 1                                 << CSTRAS , MODLP
/ CLOSE 1                                << MODCD et BUFCD
/ RLNK
/ ELNK
/ EOJ
```

4 - ARTICLE DRLPSY

```
/ JOB DRLPSY , : S, D2
/  CALL    EDILE
/ LØ      ZE
/ BØ    DRLPSY-BT
/  BI     MCDØUT . BIDRIP                << PAS MODCD
/  ILNK
/ CLOSE BI
/  BI     LANTRA . BIDRIP
/  CLNK
/  CLOSE  BI
/ BI     DIATRA . BIDRIP
/ CLNK
/  CLOSE  BI
/  OPEN  OLD  1,  BIDRIP                 << LINK DE
/ LLNK 1                                 << CSTRAS , MODLP
/ CLOSE 1                                << MODSYS et BUFSYS
/ RLNK
/  ELNK
/  EOJ
```

5 - ARTICLE DRCDSY

```
/ JOB     DRCDSY , : S, D2
/ CALL    EDILE
/ LØ      ZE
/ BØ      DRCDSY-BT
/ BI      MLPØUT . BIDRIP            << PAS MODLP
/ ILNK
/ CLOSE BI
/ BI      LANTRA . BIDRIP
/ CLNK
/ CLOSE BI
/ BI      DIATRA . BIDRIP
/ CLNK
/ CLOSE BI
/ OPEN OLD 1, BIDRIP                 << LINK DE
/ LLNK 1                             << CSTRAS
/ CLOSE 1                            << MODCD et BUFCD
/ RLNK                               << MODSYS et BUFSYS
/ ELNK
/ EOJ
```

6 - ARTICLE DRIP16

```
/ JOB     DRIP16, : S, D2
/ CALL    EDILE
/ LØ      ZE
/ BØ      DRIP16-BT
/ BI      LANTRA . BIDRIP
/ ILNK
/ CLOSE BI
/ BI      DIATRA . BIDRIP
/ CLNK
/ CLOSE BI
/ OPEN OLD 1, BIDRIP                 << LINK DE
/ LLNK 1                             << CSTRAS
/ CLOSE 1                            << MODLP
/ RLNK                               << MODCD et BUFCD
/ ELNK                               << MODSYS et BUFSYS
/ EOJ
```

S Y N O P T I Q U E

I - LES INSTRUCTIONS ET LES COMMANDES DE CONTROLE RECONNUES PAR DRIP16-A

a - Instructions

SNAP [d] ({ [@] nom de variable [+deplt] } , n [, F])
RA

RSNAP [d]

XPRINT [d] “chaîne”

SILENCE [d]

NO SILENCE [d]

XSYST est reconnue mais pas traitée

b - Commandes

ONTR

OFTR

DEGR, L1, L2

REGS

CONT

MULTI

II - LES INSTRUCTIONS ET LES COMMANDES DE CONTROLE RECONNUES PAR DRIP16-B

a - Instructions

SNAP [d] ({ [@] nom de variable [+dplt] } , n [, F])
RA

RSNAP [d]

XPRINT [d] “chaîne”

SILENCE [d]

NO SILENCE [d]

XSYST [d] (n [, liste de nombres])

n = 1 Files du scheduler
2 Etats des événements
3 Phases avancement
4 Occupation des partitions
5 Taux attente des partitions
6 Taux occupation des zones dynamiques
7 Portion ZDR
8 Etat périphérique
9 BCT d’une tâche
10 Liste opérations différées et périodiques
11 Liste des tâches en attente d’événement
12 Liste des ressources
13 Liste des fichiers ouverts
14 Dump de zone système

b - Commandes

MULTI

ONTR [, Pi, Pj , Pk -]	$0 \leq Pi \leq 127$
OFTR [, Pi, Pj , Pk -]	
DEGR, L1, L2, D1, D2 [, P]	$1 \leq Li, Di \leq 255$
ONSYS [, Ni, Nj -]	$1 \leq Ni \leq 14$
OFSYS [, Ni, Nj -]	
DFIL [, taille]	
DISC, A, FICNAM - PW	
DISC, R, FICNAM - PW	
REGS	
CONT	
FLTR	

III - LES COMMANDES DE CONTROLE DE PRODEP

ONTX [, Pi, Pj -]

OFTX [, Pi, Pj -] $0 \leq Pi \leq 127$

DEGX , L1, L2 [,P] $1 \leq Li \leq 255$

ONSX [, Ni, Nj -]

OFSX [, Ni, Nj -] $1 \leq Ni \leq 14$

HIST, H1, M1, H2, M2 $0 \leq Hi \leq 24$ heure
$0 \leq Mi \leq 60$ minute

VISU
EXAM, FICNAM - PW

CXAM

Réseaux
et systèmes
d'information

Bull

Distribution codes/Codes de diffusion			
Customers : Clients :			
Internal : Interne :			

DELIVERY ADDRESS

ETIQUETTE ADRESSE

Bull MTS

1, Rue de Provence
B.P. 208
38432 ÉCHIROLLES CEDEX / FRANCE

Bull

Sems