

THESIS / THÈSE

MASTER EN SCIENCES MATHÉMATIQUES

Méthodologie de tests de programmes d'application dans des systèmes en temps réel

Weerts, Bernard

Award date:
1976

Awarding institution:
Universite de Namur

[Link to publication](#)

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal ?

Take down policy

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX, NAMUR

Institut d'Informatique

Année académique 1975-1976

**METHODOLOGIE DE TESTS DE PROGRAMMES
D'APPLICATION DANS DES SYSTEMES
EN TEMPS REEL**

Bernard WEERTS

Mémoire présenté en vue de l'obtention
du grade de Licencié et Maître en
Informatique.

Que toutes les personnes qui ont collaboré, de quelque manière que ce soit, à la réalisation de ce travail, trouvent ici l'expression de ma sincère gratitude.

Je tiens néanmoins à remercier particulièrement M. Windal, qui en fut l'instigateur principal; Melle Delhaye, ainsi que MM. Paindaveine et Delaunois, dont la collaboration me fut précieuse, principalement lors de la conception et de la mise en oeuvre du programme présenté dans la troisième partie; Mme Dormal, qui a allié patience et talent bénévoles pour la frappe de ce mémoire; enfin, M. Goven et la firme Gestetner, dont j'ai apprécié l'obligeance désintéressée, et sans qui l'ouvrage n'aurait jamais vu le jour.

Annexe 2.1 - Les systèmes IMS et CICS	
A2.1.1 Le système IMS	A2.1.1
A2.1.2 Le système CICS	A2.1.6
Annexe 2.2 - Programmation d'un appel d'entrée de message	
Annexe 2.3 - Listing de sortie du 3270 FF de BTS	
<u>3ème partie</u> - Réalisation d'un générateur de données alphanumériques - DBALPNUM	
3.1 Présentation de DBALPNUM	3.1
3.2 Analyse des entrées de DBALPNUM	3.3
3.2.1 Les cartes de type A	3.3
3.2.2 Les cartes de type P	3.8
3.2.2.1 Carte obligatoire	3.8
3.2.2.2 Carte(s) optionnelle(s)	3.9
3.3 Analyse des sorties de DBALPNUM	3.12
3.4 Structure du programme DBALPNUM	3.15
3.5 Principes de fonctionnement de la génération	3.18
3.6 Le passage de paramètres au programme exécutable DBALPNUM	3.22
Annexe 3.1 - Sorties de DBALPNUM	
Annexe 3.2 - Listings de DBALPNUM	
A3.2.1 Etablissement des liens d'entrée, de branchement et de sortie	A3.2.1
A3.2.2 Le passage de paramètres au programme DBALPNUM via la carte JCL EXEC	A3.2.4
Conclusion	c
Bibliographie	

Dans toute entreprise manufacturière réalisant la production d'"unités" mettant en oeuvre des technologies de fabrication parfois complexes (nous prendrons pour exemple - type celui de l'industrie automobile), une part importante de temps, de matériel, voire de personnel, est toujours affectée, en temps voulu, aux tests (de conformité à des normes pré-établies, de bon fonctionnement, etc) d'échantillons sélectionnés au hasard parmi la production; sans oublier les divers tests et simulations effectués, d'autre part, dans le cadre des " études et recherches " réalisées au niveau de chaque entreprise. L'exemple de l'industrie automobile (mais les industries sidérurgique ou verrière, pour ne citer que celles-là, conviendraient aussi bien, nous en sommes persuadé) vient à point pour illustrer notre propos: chacun connaît les efforts déployés par les grands constructeurs (en matière d'amélioration des performances, du confort et de la sécurité des véhicules, notamment), qui n'hésitent pas à recourir à des essais parfois poussés, " coûteux " et fort spectaculaires pour assurer à leurs " dernières nées" l'image de marque la plus alléchante qui soit.

Il ne nous appartient pas de discuter l'analogie entre une entreprise industrielle et le domaine informatique de " production de programmes", mais elle nous semble bien choisie, en ce sens que les deux ont incontestablement en commun le recours à des techniques de réalisation qui peuvent parfois être fort compliquées. Il est, dès lors, fatal que nous débouchions, pour le second domaine ci-dessus, sur une conclusion en tous points comparable aux quelques observations que nous avons mises en exergue pour le premier, concernant la nécessité de procéder à des essais avant que les produits ne soient mis à la disposition de l'utilisateur.

On peut pousser plus loin la comparaison: ainsi, dans le travail que voici, et après une première partie constituant une brève " entrée en matière " (nous y expliquons notamment les raisons rendant difficiles les tests de programmes dans le cas particulier des systèmes en temps réel), il est en effet possible de dégager deux visages particuliers des quelques techniques de test que nous abordons ensuite: certaines d'entre elles concernent des essais de " bon fonctionnement " des programmes écrits, tandis que d'autres sont plutôt axées " essais d'études

et recherches ". Ces techniques font l'objet d'une présentation assez détaillée dans la deuxième partie, où elles s'articulent sur base d'une méthodologie consistant en une succession d'étapes, et où nous nous sommes efforcé de voir comment, en pratique, ces techniques étaient mises en oeuvre. C'est ainsi que nous avons fait la part belle à certains produits existants (en nous limitant volontairement à des produits IBM) et, principalement, à l'un d'entre eux : nonobstant le fait qu'il est abondamment utilisé (et ceci suffirait déjà à justifier cela), cela nous a paru indispensable dans le cadre du travail présenté dans la troisième partie, consistant précisément en l' "amélioration" de cet outil de test opérationnel. A ce propos, nous n'avons pas voulu détailler la structure intégrale du programme que nous avons réalisé: cela n'aurait guère présenté d'intérêt. Nous avons plutôt insisté sur ses facettes " externes ", l'examinant sous l'angle de son contexte et de son utilisation (en procédant notamment à l'analyse des entrées à lui fournir et des sorties restituées), ce sur quoi nous avons d'ailleurs également tâché de mettre l'accent pour ce qui est des techniques d'essai existantes présentées dans la seconde partie.

Tout ceci situe bien la philosophie d'ensemble de ce travail, où nous avons donc résolument adopté le point de vue de l'utilisateur.

1ère partie

Contexte général des tests de programmes
d'application dans des systèmes en temps réel.

1.1 Généralités

Un système en temps réel peut être défini comme le contrôleur d'un environnement: il reçoit et traite des informations, agit ou renvoie des résultats, et cela de façon suffisamment rapide que pour affecter le comportement de l'environnement à l'instant voulu.

1.1.1. Difficultés des essais en temps réel

En général, diverses raisons rendent malaisés les tests de programmes dans des systèmes en temps réel:

- la première d'entre elles tient au fait que, dans un système "on line", les entrées et sorties se font à partir de nombreux dispositifs à distance (terminaux), dont le réseau ne sera, de surcroît, vraisemblablement pas disponible au moment des tests;
- ensuite, un système en temps réel se caractérise par le fonctionnement simultané de plusieurs services concurrents, chacun d'eux ayant été "déclenché" à un moment quelconque à partir d'un terminal de type donné, ce qui rend à tout instant aléatoire l'environnement de déroulement d'un programme;
- enfin, la diversité des erreurs susceptibles de survenir dans pareil système constitue une autre pierre d'achoppement pour les essais.

Explicitons quelque peu chacun de ces trois points:

- dans un système en temps réel, les entrées se font à partir de nombreux terminaux ou dispositifs plus ou moins éloignés. Or il va de soi qu'au moment des essais, il sera nécessaire d'utiliser des entrées prédéterminées pour les programmes à tester. Toutefois, ce serait une véritable perte de temps que d'utiliser les terminaux réels, d'autant plus que ceux-ci ne seront probablement pas encore installés au début des essais.

Les terminaux, écrans d'affichage ou autres dispositifs à distance sont aussi utilisés pour les sorties et les raisons qui, pour les entrées, rendaient impraticable l'utilisation de ceux-ci restent bien entendu valables pour les sorties.

- le problème soulevé au second point ci-dessus tient en deux notions caractéristiques des systèmes en temps réel: celles "d'imprévisibilité" et de "variabilité" de ces systèmes par rapport aux systèmes "batch".

Qu'est-ce à dire au juste ?

- dans un système en temps réel important, les messages surviennent de façon totalement aléatoire, en provenance de terminaux parfois fort différents; à tout moment, et le système étant dans un état donné (il est " au repos" ou occupé à " traiter" d'autres messages, introduits précédemment), un ou plusieurs utilisateurs peuvent se connecter au système et introduire un ou plusieurs messages qui déclencheront tôt ou tard autant de procédures de " traitement" de ceux-ci, conditionnant donc l'état futur du système, mais de façon purement aléatoire, puisque les initiatives extérieures (utilisateurs) sont tout à fait imprévisibles.
- alors que, dans un traitement en temps différé, le déroulement des événements suit une route " séquentielle" choisie parmi un petit nombre d'itinéraires prévus à l'avance selon un certain type d'entrée, dans les systèmes en temps réel et à multiprogrammation, le nombre de combinaisons possibles est pratiquement infini; l'éventail des organes d'entrée y est habituellement beaucoup plus étoffé qu'il ne l'est pour des systèmes "batch" (par exemple, toutes les variétés de terminaux, télétypes ou écrans IBM) et, dans la plupart des cas, les types de transactions (par transaction, on entend l'instauration d'un lien entre un terminal donné et le système d'ordinateur, débouchant sur un échange d'informations de l'un à l'autre et le traitement de celles-ci, pour une application déterminée) y sont nettement plus nombreux.

Semblablement, l'ensemble des actions que le système peut entreprendre est nettement plus vaste que pour des systèmes en temps différé. Dans des systèmes " on line" orientés vers la gestion des affaires, par exemple, le système réagira différemment en fonction des contenus de tels ou tels fichiers qui, du fait de la présence possible de services concurrents, peuvent changer à tout instant. En d'autres termes, le comportement du système à un moment donné est le plus souvent fonction des autres activités du système à ce moment.

En clair, ceci signifie qu'un programme testé et ayant donné satisfaction hier peut très bien ne plus fonctionner convenablement aujourd'hui, simplement parce qu'il s'exécute dans un environnement différent.

- la multiplicité des erreurs susceptibles d'entraver le bon fonctionnement d'un système en temps réel pose elle aussi de sérieux problèmes. Non pas que ces erreurs soient toutes spécifiques aux systèmes en temps réel, au contraire; toutefois, un tel système opère, dans la mesure du possible, sans intervention humaine et doit donc être capable de prendre en charge ses propres procédures de " secours" plutôt que de se reposer entièrement sur un opérateur. Ce dernier est averti quand intervient une anomalie irréparable pour le système, mais, en général, celui-ci assure lui-même le meilleur service possible aux utilisateurs.

Cette indépendance des systèmes en temps réel vis-à-vis de toute intervention humaine, ajoutée à la variété des erreurs susceptibles de s'y produire, sont loin de simplifier la tâche de l'analyste responsable de la conception d'un tel système, qui doit imaginer une procédure de réponse spécifique du système à chaque type d'erreur pouvant survenir. Si l'on exige de la part du système un haut degré de fiabilité, ces procédures peuvent être très complexes et donc difficiles à mettre au point.

La diversité des erreurs, sur laquelle nous insistons, est absolument indéniable : à côté des erreurs de programmation (que les essais ont d'ailleurs pour but de révéler), il existe en effet une multitude d'erreurs surgissant de façon absolument imprévue, sans aucune raison apparente. On peut ranger dans cette "catégorie" . les avaries survenant dans certaines unités de l'ordinateur ou de son environnement (par exemple, une panne des circuits de commande de la mémoire centrale ou de l'alimentation électrique du calculateur principal), encore que la plupart des pannes de ce genre se produisent dans des composants qui ne sont pas indispensables au fonctionnement global du système (par exemple, un terminal, une ligne de communication, un fichier d'un ensemble, etc); les erreurs dans le système d'exploitation;

- les anomalies de fonctionnement dont l'origine n'est pas une défaillance permanente d'une partie du système: il peut s'agir, par exemple, de parasites sur une ligne de communication, d'une erreur que nous qualifierons d'"humaine" (un opérateur introduit au terminal un message incorrect, comme une quantité ou un montant erronés, ou bien il se trompe dans la procédure d'introduction des données), de la défectuosité d'un capteur ou d'un instrument dont proviennent habituellement les informations d'entrée pour le système, d'une erreur survenant lors du transfert de données entre le calculateur et ses unités d'entrée-sortie.

Enfin, dans un système en temps réel complexe ne commandant pas lui-même le rythme de ses propres entrées (par exploration des instruments auxquels il est relié), il peut survenir aussi certains événements débouchant sur des situations qui, à défaut d'être des erreurs ou des pannes, n'en sont pas moins extrêmement gênantes. Ce sera le cas, par exemple, si tous les utilisateurs du réseau (ou, du moins, une forte majorité d'entre eux) expédient simultanément un message à destination de l'ordinateur central, ce qui donnera lieu à une situation de surcharge du système (soit dit en passant, ceci ne se produira jamais dans un système " conventionnel", où le débit des entrées est déterminé par l'ordinateur lui-même, qui lit des cartes, bandes magnétiques, etc suivant ses propres besoins), impliquant que le système prenne une décision d'urgence.

Somme toute, les traitements de certaines ^{des} erreurs dont nous venons de parler (notamment celles concernant des unités de l'ordinateur non indispensables au télétraitement) ressemblent beaucoup à ceux qui sont d'application pour des systèmes en temps différé, à cette différence près que, dans toute la mesure du possible, on tâchera de ne pas arrêter le système, puisque l'on travaille en temps réel, où tout retard peut avoir des conséquences catastrophiques, principalement en allongeant sensiblement le temps de réponse.

La multiplicité des erreurs que nous venons d'énumérer signifie que, malgré tout le soin apporté aux essais, et en dépit du fai

que le système y ait répondu de manière satisfaisante, on ne peut, en pratique, jamais conclure avec certitude qu'il fonctionnera, à l'avenir, sans la moindre anicroche.

En général, on peut même dire que c'est longtemps après la fin des essais, lors de l'exploitation du système, que les dernières traces d'erreurs se révèlent.

En clair, ceci veut dire qu'au cours de l'exploitation opérationnelle, quelques essais supplémentaires seront encore nécessaires. Toutefois, ces essais seront difficiles, le système en temps réel occupant en machine tout le temps qui lui est nécessaire. Un supplément de temps doit donc être disponible pour les derniers tests. En outre, toute erreur grave (par exemple, une erreur de programmation qui aurait échappé à la vigilance des responsables des essais) survenant au cours du fonctionnement risque d'altérer des informations vitales figurant dans des fichiers, d'où la nécessité que le système comporte divers moyens de protection et de reconstitution de fichiers. Dans l'optique de ces problèmes, un système fonctionnant vingt-quatre heures par jour et ne comprenant aucun hardware additionnel réservé aux derniers essais se révèle être inconcevable.

Mais nous ne soulignerons jamais assez le soin particulier à apporter aux essais lors de la période, préalable à l'exploitation, qui leur est dévolue. Il faut bien se dire que toute erreur détectée à ce moment ne devra plus l'être ultérieurement.

Les trois problèmes que nous venons de soulever rendent nécessaire, en période de tests, le recours à des procédures de simulation :

- simulation des entrées/sorties de messages par terminaux

Puisque le réseau n'est, pratiquement, pas utilisable, les entrées doivent être simulées: par exemple, dans le cas d'une application scientifique, les signaux d'entrée provenant de capteurs tels radars, jauges ou autres instruments de mesure analogues seront simulés; dans le cas, plus général, où les entrées sont de simples messages en provenance de terminaux de type " machine à écrire", on pourra enregistrer les messages

d'essais sur cartes ou sur bande(s) . Ceux-ci seront alors fournis aux programmes en cours de test par un programme spécialisé, exactement comme s'ils arrivaient d'un terminal.

En ce qui concerne les sorties (réponses du système aux messages d'entrée), on peut, de la même façon, imaginer leur enregistrement sur bande magnétique en vue d'une impression ultérieure.

- afin d'éviter les inconvénients du deuxième problème que nous avons signalé, il conviendra de simuler (en combinant différents groupes d'entrées, par exemple) le fonctionnement d'un programme dans un maximum d'environnements différents.
- enfin, en ce qui concerne les erreurs, pannes ou situations d'exception que nous avons passées en revue précédemment, le caractère foncièrement imprévisible de la majorité d'entre elles constitue un obstacle difficilement surmontable à leur reproduction systématique en vue d'investigations ultérieures. Cependant, rien ne s'oppose, pensons-nous, à ce qu'on en simule un certain nombre en vue d'y apporter, déjà, les solutions souhaitées. Par exemple, des erreurs d'opérateur peuvent être aisément simulées en intercalant, parmi les cartes représentant les messages d'entrée, des messages " incorrects". Dans le même ordre d'idées, la mise au point d'un simulateur de surcharge doit aussi être possible.

1.1.2. Implémentation d'un système en temps réel.

Les remarques que nous avons faites quant à la " variabilité" des systèmes en temps réel par rapport aux systèmes plus conventionnels expliquent la différence de conception des premiers par rapport aux seconds.

En effet, dans un ordinateur utilisé pour une application conventionnelle, la séquence des événements suit un cycle répétitif et peut donc être " planifiée" en détail par le programmeur; les opérations d'entrée-sortie ont une durée bien déterminée et peuvent donc s'effectuer simultanément, en parallèle avec le traitement " principal".

En téléprocessing, cependant, nous avons dit que les messages arrivaient " à l'improviste", variant tantôt en longueur, tantôt en nature et que la séquence des opérations était tout à fait imprévisible.

Il n'empêche que l'information doit être traitée en un minimum de temps, les diverses ressources de la machine étant utilisées au mieux. Afin d'assurer un temps de réponse raisonnablement court, il va de soi que le système devra être à même d'assurer le " service" de plusieurs transactions en même temps, et donc le déroulement des programmes qui leur correspondent, ceux-ci pouvant accéder et mettre à jour simultanément les mêmes fichiers.

Ces diverses considérations rendent nécessaire la présence d'un programme de contrôle, réalisant l'ordonnancement des travaux, les allocations de mémoire et des priorités d'entrée/sortie, le contrôle des ressources, etc, en un mot, toutes tâches qui ne peuvent être codées à l'avance, de façon rigide, par un programmeur.

On aboutit par conséquent à une inévitable séparation de fonctions dans un système en temps réel :

- d'une part, la fonction de contrôle, assurée par un programme spécialisé (par exemple IMS - Information Management System - ou CICS - Customer Information Control System - chez IBM) qui a pour but la mise du réseau de terminaux en relation avec des fichiers localisés sur des mémoires à accès direct (disques, tambours, etc), et qui comprend des programmes de gestion du réseau et des fichiers, d'accès aux terminaux et aux fichiers, ainsi qu'un programme "directeur" (ou " de contrôle"), constituant l'ossature de ce sous-système et veillant à la gestion de l'ensemble; ce programme " directeur" peut être écrit de toutes pièces ou être un emprunt de certaines fonctions du superviseur du système d'exploitation.

La figure 1.1 regroupe schématiquement les éléments dont nous venons de parler.

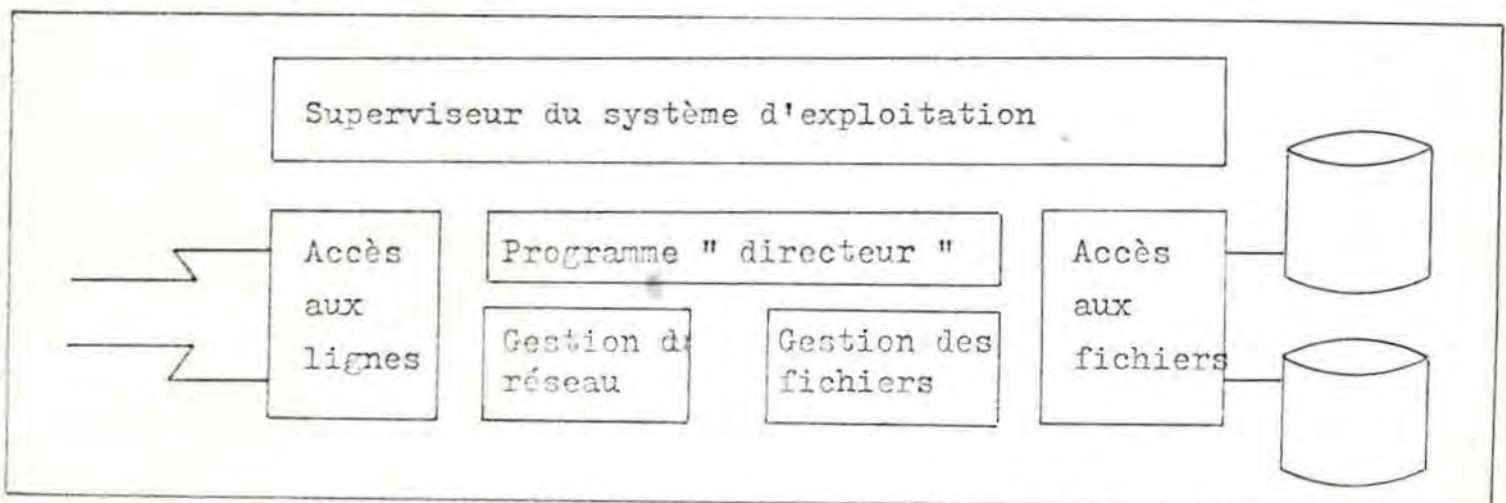


figure 1.1

- d'autre part, la fonction de traitement, dévolue aux programmes d'application, qui en sont donc "réduits" à traiter le ou les message(s) provenant d'un terminal et mis à leur disposition (dans une zone tampon) par le programme de contrôle, puis à renvoyer une ou des réponse(s) à ce terminal, à nouveau via le programme de contrôle. Les programmes d'application sont donc conçus selon une philosophie essentiellement "batch": ils n'ont aucun contact avec le monde extérieur et sont réellement "aveugles" aux problèmes posés, par exemple, par la gestion des lignes et ^{des} fichiers ou l'enfilement des messages en attente.

En général, la littérature relative à ce sujet ([1]) distingue trois sortes de programmes :

1. les programmes d'application, réalisant le traitement des transactions ou messages. Ils ne comportent aucune codification d'entrée et de sortie, excepté sous la forme de macro-instructions, qui transfèrent le contrôle à un programme de gestion des entrées / sorties ou moniteur (qui, dans un système en temps réel, fait partie des programmes superviseurs);
2. les programmes superviseurs, qui coordonnent et gèrent le travail des programmes d'application. Ils traitent les opérations d'entrée/sortie, ainsi que la gestion des files d'attente de messages et de données, les interruptions, et s'occupent des cas d'erreurs ou d'urgence;
3. les programmes de soutien (ou utilitaires), nécessaires pour mettre le système en place et le maintenir en ordre de marche sans à-coups. On peut citer, par exemple :
 - programmes de chargement ou de mise en route du système;
 - programmes de redémarrage;
 - diagnostics;
 - programmes de réorganisation de fichiers;
 - programmes de chargement de fichiers;
 - programmes générateurs de données pour la période d'essais;
 - simulateur de programme superviseur pour la période d'essais;
 - simulateur de programme d'application pour la période d'essais;
 - programme superviseur d'essais;
 - programmes d'impression de mémoire centrale;
 etc.

La distinction entre les fonctions accomplies par les programmes d'application et les programmes superviseurs n'est pas toujours évidente. Programmes d'application et programmes superviseurs peuvent accomplir des fonctions analogues, comme, par exemple, des traitements d'erreurs. Mais même dans ce cas précis, la tâche réservée aux programmes d'application restera dans le domaine des manipulations de messages, les types d'erreurs dont ils s'occuperont étant exclusivement du genre "erreurs humaines" c-à-d. où le message introduit est incorrect. C'est au niveau des programmes d'application que chaque message est décodé et testé. C'est, par contre, au niveau des programmes superviseurs que des procédures d'exception correspondant à des situations telles surcharges, etc (c-à-d. où n'intervient pas la substance de chaque message) seront exécutées; c'est aussi à ce niveau que sont exécutés des programmes de calcul d'adresses sur fichiers et, plus généralement, de préparation des messages pour leur traitement par les programmes d'application, préparation consistant essentiellement en l'"isolation" de l'information pertinente (cette expression faisant opposition à celle d'information purement " indicative", comme, le demi-mot donnant la longueur totale de chaque message) à destination de ceux-ci.

2 Méthodologie de tests de systèmes en temps réel

La séparation des fonctions, dont nous avons parlé au paragraphe 1.1.2, implique, lors de l'implémentation d'un système en temps réel, une double réalisation : celle du programme de contrôle, d'une part, des programmes d'application, d'autre part. Cette " duplication " ne manque d'ailleurs pas de poser certains problèmes : ainsi, un échelonnement des essais selon une démarche séquentielle : 1. tests des programmes superviseurs; 2. tests des programmes d'application, risquerait d'éveiller certains conflits. Expliquons-nous : un petit nombre (aussi infime soit-il) d'erreurs échappent habituellement à l'investigation lors des essais; il est donc vraisemblable que les programmes superviseurs ne seront pas parfaitement mis au point au moment (début de l'étape 2) de leur utilisation avec les premiers programmes d'application à tester. La question peut alors se poser de savoir si une erreur survenue provient d'un des programmes superviseurs ou d'un programme d'application. Si, comme cela se produit souvent, deux équipes sont chargées, séparément, de l'écriture des programmes superviseurs et d'application, cela peut être une source particulière de désaccord. Il faudrait donc que l'on dispose de moyens permettant de déterminer si une erreur est due à un programme superviseur ou à un programme d'application. (C'est d'ailleurs grosso modo le même problème qui risque de se manifester lorsqu'une erreur survient alors que deux programmes d'application sont en mémoire simultanément. Lequel est responsable ? Il se peut que, par exemple l'un d'eux écrive incorrectement sur un fichier utilisé par l'autre et soit ainsi responsable de l'erreur ultérieure qui ne manquera pas de se produire. Aucune indication ne pourra être donnée quant à l'origine réelle de l'erreur).

Ceci dit, en vue de pallier l'inconvénient que nous venons de signaler, né de la linéarité du schéma des tests, Martin ([1]) estime souhaitable de commencer à procéder aux essais des programmes d'application avant que les programmes superviseurs ne soient complètement rédigés, ce qui implique que l'on dispose, au départ, de macro-instructions simulant de façon simple celles des programmes superviseurs (à l'usage des tests de programmes d'application) et de " pseudo-programmes d'application" (à l'usage des tests de programmes superviseurs). Ceci revient à dire qu'au lieu de s'abandonner à la rigidité de la démarche du début, tout en émettant des

réerves, au début de l'étape 2, quant à la validité absolue des programmes testés à l'étape 1, il est plutôt recommandé de procéder parallèlement aux essais des programmes superviseurs et des programmes d'application.

Toutefois, dans la plupart des installations de téléprocessing (TP) actuelles, le programme de contrôle à utiliser est " pré-disponible": il s'agit d'un software écrit une fois pour toutes, livrable à la demande et plus ou moins étoffé selon les désirs des clients, ayant, en tout cas, fait ses preuves, et dont les essais de bon fonctionnement sont, par conséquent, superflus. Ainsi, chez IBM, il existe principalement deux systèmes de ce type : IMS et CICS.

Il s'ensuit que le seul problème encore à régler quant à la mise sur pied du système complet consiste en :

- le développement des programmes d'application, comprenant, d'une part, leur conception puis leur écriture, et, d'autre part, le test de leur bon fonctionnement (c-à-d la vérification de l'exacte réalisation des fonctions qui leur sont assignées).

La conception de ces programmes recouvre une double tâche, si l'on sait que la plupart des applications en temps réel actuelles travaillent sur des banques de données : les performances du système se ressentiront essentiellement de la qualité de la conception de ces bases de données, intimement liée à leur exploitation par les différents programmes d'application. L'optimisation sous-jacente qui s'impose s'inscrit dans ce que nous appellerons plus loin " l'aide au design".

- l'intégration des programmes d'application dans le système complet et la vérification finale des performances de celui-ci. Un dernier ajustement des paramètres du programme de contrôle, comme les tailles de " buffers", les longueurs de files d'attente, etc peut améliorer ces performances. Chez IBM, certains outils d'aide, soit au " design", soit aux tests de programmes proprement dits, accompagnent des produits tels IMS ou CICS dont nous avons déjà parlé. Nous allons les passer en revue dans la deuxième partie.
-

2ème partie

Méthodologie de tests de programmes d'application
dans des systèmes en temps réel.

Dans cette seconde partie, nous nous proposons de passer en revue, au travers d'une méthodologie les regroupant, sur base d'une succession chronologique, quelques techniques inhérentes aux tests de programmes d'application dans les systèmes en temps réel, tout en nous attachant à décrire comment, en pratique, celles-ci sont mises en oeuvre dans des produits existants.

Nous partons du principe énoncé au paragraphe 1.2, à savoir qu'un programme de contrôle est disponible une fois pour toutes et, cela va de soi, donne satisfaction. Comme nous le disions aussi dans la première partie, ce programme de contrôle s'accompagne généralement d'outils facilitant les tests de programmes d'application et regroupés en un programme de contrôle d'essais; celui-ci assure des fonctions analogues à certaines de celles du programme de contrôle opérationnel, mais dans un contexte différent, puisque son utilisation est limitée exclusivement à la période de tests, ceux-ci se déroulant en grande partie dans une philosophie " batch", simulant les processus d'entrée et de sortie des messages en provenance du réseau, comme nous le suggérons au paragraphe 1.1.1.

Cette similitude de fonctionnement entre certains modules des deux programmes de contrôle est indispensable, car les programmes d'application doivent pouvoir se dérouler, pendant les essais, de la même façon qu'au cours de l'exploitation ultérieure du système, sans qu'aucune modification ne soit apportée à leur structure pour passer d'une étape à l'autre. Il faut se rappeler que le programmeur d'application ne se soucie pas le moins du monde de l'environnement extérieur de ses programmes; ce sera au programme de contrôle d'essais de jouer le rôle d'interface, en alimentant en messages les programmes d'application, puis en recevant leurs réponses à destination du réseau comme si de rien n'était.

2.1 Simulation des entrées et génération des données

2.1.1. Approche théorique

Dans l'optique de ce qui précède, la première partie de la méthodologie que nous proposons recouvre ce que nous appellerons " la préparation de l'environnement extérieur des programmes à tester"; elle consiste en la génération ou la préparation de données de test qui sont soit des messages d'entrée, soit des enregistrements de fichiers ou base de données " de travail" indispensables au déroulement normal des programmes d'application (figure 2.1)

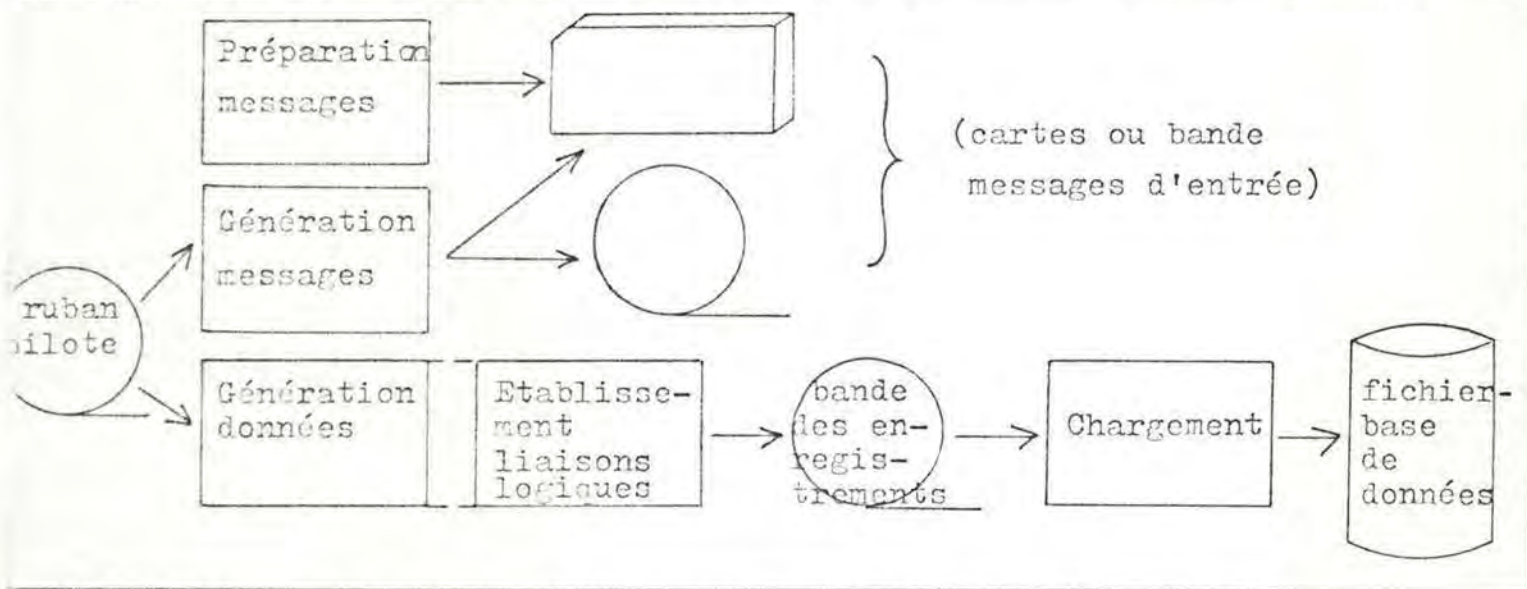


Figure 2.1 : simulation des entrées et génération des données

Nous ne reviendrons pas sur les raisons motivant la nécessité d'une simulation de l'introduction des messages d'entrée, plutôt que l'utilisation, inefficace, du réseau de terminaux à cette fin. A côté de cela, pour un système en temps réel mettant en jeu une ou des banques de données, il existe plusieurs raisons rendant souhaitable la création d'une base de données spécifique aux tests ("artificial test data base", [2]), là où pourrait intervenir une base de données opérationnelle ("live data base", [2]) :

- il est possible que cette dernière n'existe pas encore au moment des essais;
- en supposant qu'elle existe, sa taille excessive pourrait occasionner une perte de temps inutile lors de son chargement, à partir d'une bande magnétique, en préparation d'un test;

- il se peut aussi qu'elle contienne des informations "sensitives" qu'on ne souhaite pas rendre accessibles aux responsables des tests;
- enfin, les changements susceptibles d'intervenir dans une banque de données opérationnelle, entre deux essais, rendraient difficile la tâche des programmeurs, qui ne disposeraient plus d'un point de référence indispensable (précisément assuré, entre autres, par la "stabilité" des informations manipulées) pour tester l'impact de leurs programmes.

Sans compter que (et c'est une raison importante) les "déraillements" d'un programme pas encore au point risqueraient d'altérer des informations primordiales de la base opérationnelle.

Comme pour l'introduction de messages d'entrée à partir des terminaux, un chargement manuel d'une base de données de test serait fort gênant et, en tout cas, terriblement long.

Tout ceci rend opportun le recours à des programmes générateurs, produisant des données de façon automatique à partir d'une définition des formats de messages ou d'enregistrements possibles. C'est d'ailleurs à une tâche analogue que nous nous sommes attelé dans le cadre de la troisième partie de ce travail.

Ceci dit, la création d'enregistrements de fichiers pour un système important (tel que celui mis en place pour une compagnie aérienne, par exemple) n'est pas aussi simple qu'il n'y paraît: certes, les générateurs programmés peuvent produire une grosse masse d'enregistrements, mais la liaison logique nécessaire entre ceux-ci est très difficile à élaborer. On sera même probablement amené à utiliser des méthodes manuelles pour réaliser ce lien logique entre enregistrements. De même, les adresses de ces enregistrements peuvent être difficiles à produire, ce qui exige le développement de techniques d'adressage et la mise au point de programmes chargeurs assez complexes. Habituellement, toutefois, au début des essais, lorsque l'on n'a besoin que d'un nombre limité d'enregistrements, on peut créer les adresses manuellement, ce qui permet la vérification du bon fonctionnement des programmes d'adressage.

L'ensemble des enregistrements utilisés pour les essais constitue en réalité un modèle pilote de ceux qui occuperont le système dans sa phase réellement opérationnelle: les "objets" qui y sont représentés

sont les mêmes que ceux appelés à figurer dans le fichier définitif, mais ils constituent un ensemble plus restreint que celui des enregistrements nécessaires en cours d'exploitation : par exemple, l'ensemble des enregistrements pour un système de gestion de stock ne comprendra qu'une partie des articles en stock; un système de réservation de places d'une compagnie aérienne peut posséder un modèle pilote dans lequel les enregistrements ne concernent qu'un certain type d'avions (les appareils de plus faible capacité, par exemple). Bien entendu, les informations engendrées pour les essais doivent se rapporter aux " articles " du modèle pilote; dans ce but, un ruban pilote peut être élaboré, destiné à contenir les définitions d'enregistrements qui permettront d'engendrer ceux-ci au moment des essais. Cette bande pilote pourra contenir aussi les définitions de messages d'entrée et, au fur et à mesure de l'avancement de la rédaction des programmes et du déroulement des essais, de nouveaux enregistrements y seront ajoutés.

Avant l'entreprise d'une série d'essais, un programme de soutien explorera le ruban pilote pour créer les enregistrements de données et les messages d'entrée. On peut également compléter le groupe d'informations générées à cette étape par des cartes contenant des messages ou des enregistrements additionnels, créés de toutes pièces par le programmeur et ne correspondant à aucun enregistrement de la bande pilote. Une fois le chargement des fichiers de données réalisé, le test peut commencer : c'est un programme spécialisé (superviseur d'essais, [1]), faisant partie du programme de contrôle d'essais, qui lira les messages d'entrée, permettant de la sorte à ceux-ci de se présenter au système comme s'ils provenaient des lignes de communication.

La génération automatique nous paraît intéressante dans la mesure où elle assure la standardisation des données produites, conformément à un schéma pré-établi stipulant leur format. En procédant eux-mêmes aux essais de leurs programmes, les programmeurs d'application risquent peut-être même inconsciemment, mais parce qu'ils ont toujours à l'esprit les problèmes qu'ils ont rencontrés lors de la conception, de " diriger " les essais, en ayant tendance à pousser ceux-ci dans

des sections où ils ont buté sur des difficultés et à les négliger dans d'autres, où, peut-être, des erreurs subtiles leur échapperont. Les messages de test qu'ils créeraient eux-mêmes, de toutes pièces, dans cette optique, pourraient donc " dénaturer " les essais, en ne mettant à l'épreuve que certains aspects des programmes d'application. Dans ces conditions, le recours à des procédures de génération, parfois complexes, peut s'avérer utile.

Ce que nous venons de dire n'est toutefois pas généralisable: dans certains cas, une préparation manuelle soigneuse par les programmeurs eux-mêmes peut conduire à des résultats très positifs (pour des applications d'envergure limitée, par exemple). Dans la pratique, c'est même presque exclusivement cette seconde procédure qui sera utilisée: les messages à introduire sont bien souvent trop " spécifiques " (comme nous le verrons dans le cadre du paragraphe 2.3.2.1) que pour faire l'objet d'une génération automatisée. Chez IBM, il n'existe d'ailleurs pas, à notre connaissance, de générateur de ce type qui soit réellement " opérationnel". Il y a bien un produit appelé " Test Data Generator ", qui construit des fichiers de données de test à partir d'indications fournies, sur cartes, par l'utilisateur. Mais il est tout à fait inutilisé, pour diverses raisons qu'il ne nous appartient pas de discuter, et nous nous bornons donc à le citer pour mémoire.

2.1.2 Réalisation pratique

Dans le domaine de la génération de banques de données de test, par contre, un outil est fort prisé auprès des responsables d'essais de programmes d'application: il s'appelle DBPROTOTYPE et réalise, entre autres tâches, la construction de bases de test, mais de manière telle que l'utilisateur n'a pas énormément de prise sur leurs contenus. Il s'agit, à vrai dire, d'un groupe de six programmes supportant le système IMS, dont nous avons cité le nom précédemment. Pour l'instant, nous allons nous intéresser aux quatre programmes liés, de près ou de loin, à la constitution d'une base de données de test, à savoir DBTABLE, DBLOAD, DBXGEN et DBXUPDTE, dont les interactions sont schématisées sur la figure 2.2.

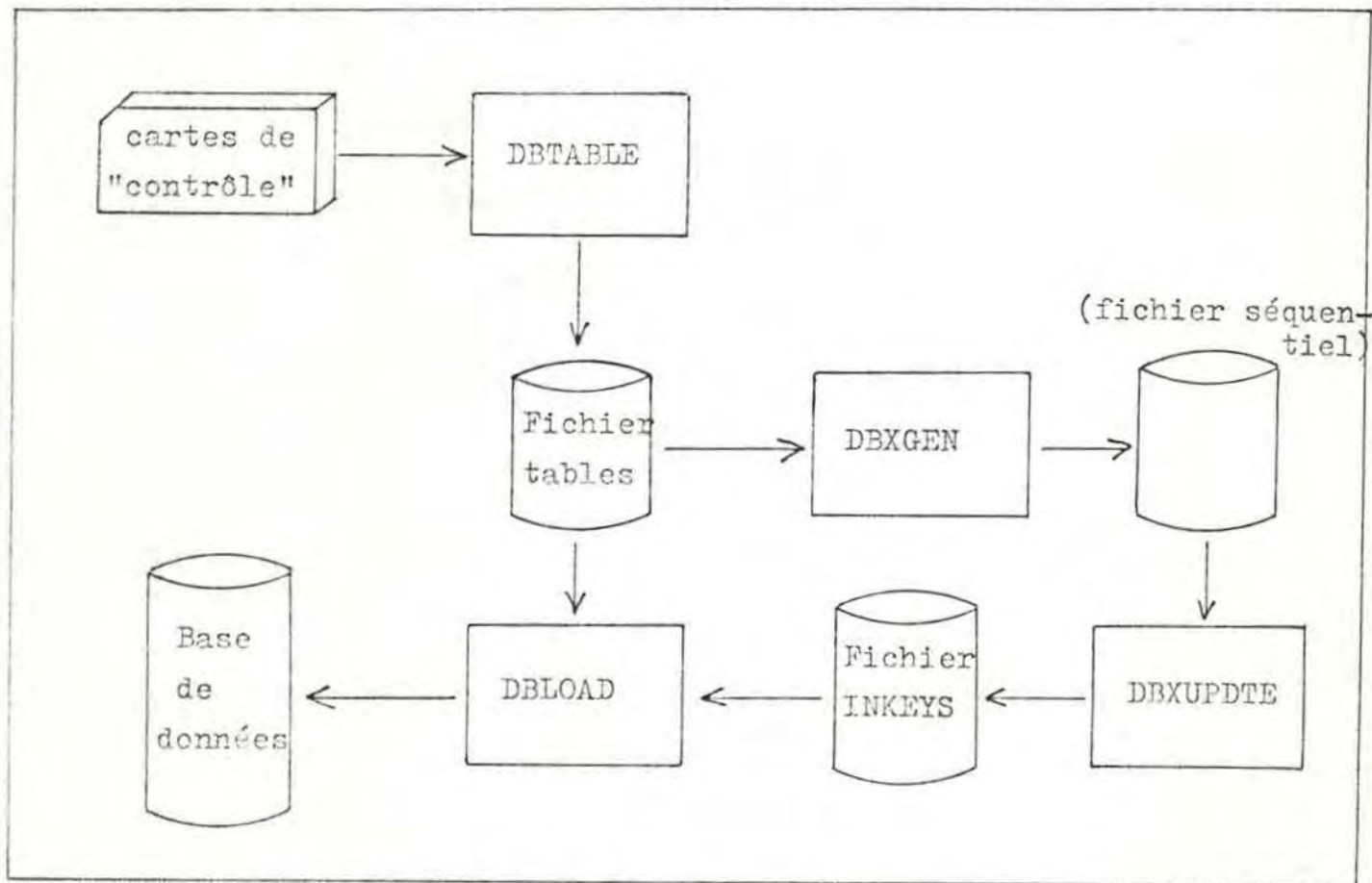


figure 2.2 : Schéma simplifié de DBPROTOTYPE

2.1.2.1 DBTABLE

Ce programme constitue des tables à partir des informations contenues dans les cartes de contrôle spécifiant les caractéristiques de la base de données à "générer".

En IMS, l'unité d'information décrite lors de la constitution d'une banque de données est le segment (chaque segment doit faire l'objet d'une définition apparaissant dans une "Data Base Definition" ou DBD) : c'est sur base de cette notion que sont conçues les cartes de contrôle figurant en entrée. Nous allons, à l'aide d'un exemple, étudier brièvement la signification de certaines d'entre elles.

Considérons une application manufacturière, dans laquelle un produit peut être fabriqué en usine (ce qui requiert un certain nombre "d'opérations") ou acheté à un ou des vendeur(s) local(locaux), ce que nous schématiserons comme suit, chacun des rectangles de la figure représentant un segment :

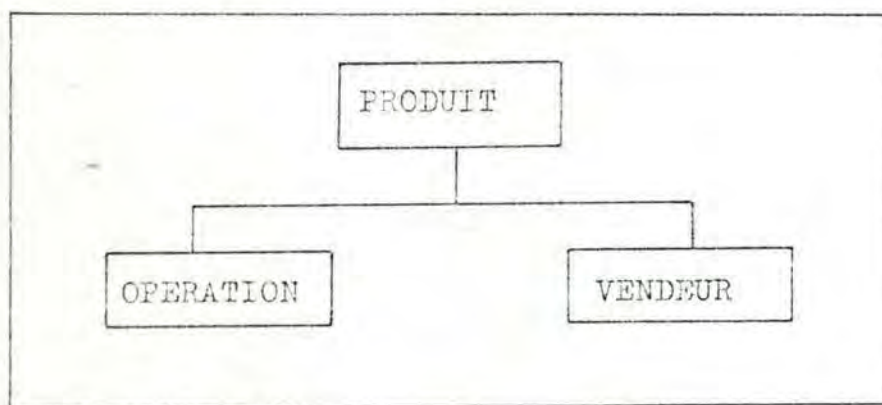


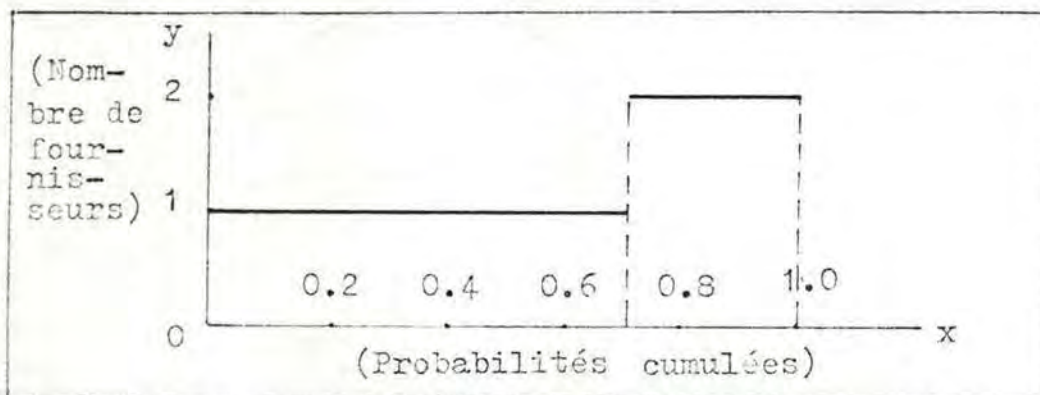
figure 2.3

Un produit donné n'aura de relation qu'avec un des deux segments OPERATION ou VENDEUR, selon qu'il est fabriqué ou acheté. Ceci signifie que la présence ou l'absence d'une occurrence d'OPERATION détermine l'absence ou la présence d'une occurrence de VENDEUR. On dira que OPERATION est un contrôleur du segment VENDEUR. (A première vue, rien n'empêche que VENDEUR soit considéré comme contrôleur de OPERATION; pour éviter cette ambiguïté, on convient que le segment décrit en premier lieu dans la DBD est le contrôleur). Plus généralement, un segment contrôleur est un segment dont la présence (ou l'absence) détermine la fréquence d'apparition d'un autre segment, ce dernier ne pouvant pas être un "enfant" du précédent.

Sur la carte de type 1, l'utilisateur précisera notamment, pour un segment donné :

- le nom de ce segment (ex: VENDEUR);
- le nom du " parent " de ce segment (ex : PRODUIT);
- le nom du contrôleur de ce segment (ex : OPERATION).

Outre ces renseignements, l'utilisateur devra donner des indications quant aux distributions de fréquence des différents segments définis. Reprenons notre exemple et supposons que les produits achetés proviennent d'un fournisseur dans 70 % des cas et de deux dans les 30 % restants, ce que l'on peut illustrer comme suit :



L'information " nombre maximal d'occurrences à générer pour le segment considéré " (2, dans notre exemple) figurera sur la carte de type 1 se rapportant à celui-ci. L'utilisateur spécifiera la distribution, en lui donnant un numéro de référence (par exemple, le n° 11), sur une carte de classe 2, en ramenant l'axe vertical à une échelle décimale (de 0 à 1.000, soit donc de 0 à 1000) et en couplant les coordonnées X_i et Y_i correspondantes:

000 (X_1) 500 (Y_1) 700 (X_2) 500 (Y_2) 700 (X_3) 999 (Y_3)

Ceci constitue la distribution " absolue" (normal distribution, [5]) du segment VENDEUR. Précisons d'emblée que sept distributions pré définies, numérotées de 01 à 07, existent une fois pour toutes, et, parmi elles, la distribution particulière " générer zéro segment" (n° 01) dont nous allons reparler.

A côté de cette distribution " autonome", l'utilisateur est tenu de préciser, pour les segments contrôlés (et VENDEUR en est un, nous l'avons vu) une distribution particulière (controlled distribution, [5]), exprimant la fréquence d'apparition du segment contrôlé lorsqu'une ou plusieurs occurrence(s) du segment contrôleur sont présentes. Dans le cas qui nous occupe, nous savons que lorsque PRODUIT possède un (ou plusieurs) segment(s) dépendant(s) OPERATION, aucun segment VENDEUR n'existe pour ce produit. La distribution "contrôlée" du segment VENDEUR sera donc la distribution 01 évoquée ci-dessus.

La longueur du segment, fixe ou variable (auquel cas elle est donnée sous forme d'une distribution) sera aussi indiquée.

En résumé, les cartes de types 1 et 2 contiendront principalement les informations suivantes:

<u>Type 1</u>	<u>Type 2</u>
Nom du segment du parent de ce segment du contrôleur de ce segment	Pour chaque n° de distribution, suite des couples (X_i , Y_i) définissant celle-ci.
N° de référence distributions absolue et " contrôlée", avec nombre maximal d'occurrences s'y rapportant	

Longueur fixe ou n° de référence
de distribution si elle est
variable, avec valeurs mini-
male, maximale et moyenne de
la longueur.

L'utilisateur a alors le loisir de diviser chacun des segments qu'il a définis en "fields", et de préciser la position, la longueur et surtout le type de chacun de ceux-ci sur une carte de classe 3; dans l'état actuel des choses, nous ne nous étendrons guère sur ce point. Disons simplement qu'il peut s'agir d'une constante (dans ce cas, sa valeur alphanumérique, binaire ou packed sera aussi indiquée), d'une clé (numérique; sa valeur ne sera pas donnée explicitement, mais bien sous forme d'un incrément s'additionnant de proche en proche, pour chaque occurrence créée de ce segment), d'un nombre (construit en multipliant le nombre aléatoire, généré par une routine interne, par un incrément s'additionnant comme dans le cas précédent etc.

Incontestablement, c'est ici que le bât blesse quant à la véritable portée de cet outil. L'utilisateur, en effet, n'a guère la possibilité d'intervenir efficacement pour exprimer, par exemple, les formats de données qu'il désire voir enregistrées dans la base. Celles-ci sont essentiellement numériques, ou alors il s'agit de constantes ce qui ne présente évidemment qu'un intérêt très relatif. C'est dans cette optique que nous avons réalisé le travail présenté dans la troisième partie, afin de donner à l'utilisateur plus de champ libre quant à l'expression de ses besoins en informations, et ce par le biais d'une présentation simple des formats-types de celles-ci.

2.1.2.2. DBLOAD

Comme son nom l'indique, ce programme est responsable du chargement de la banque de données pour son exploitation ultérieure. Il réalise cela en partant du fichier des tables produit par DBTABLE, de la DBD et aussi, éventuellement, du fichier INKEYS produit par DBXUPDTE (voir plus loin). Il produit en outre un certain nombre de

"statistiques" et divers diagnostics.

2.1.2.3. DBXGEN et DBXUPDTE

L'exécution de ces deux programmes sera commandée dans des circonstances spéciales, notamment lorsque la base de données fait intervenir des relations logiques ou bien contient des segments sources d'indexage secondaire. (Nous verrons, dans un exemple du paragraphe 2.2.2.1, qu'il existe un troisième cas d'utilisation de ces deux programmes).

Nous allons décrire ces deux dernières notions sur des exemples simples, sans nous étendre outre mesure sur des considérations théoriques que nous risquerions de ne pas bien cerner.

- A. Pour expliquer les relations logiques, reprenons notre exemple PRODUIT-OPERATION-VENDEUR, constituant une ligne hiérarchique (L.H.)¹ et mettons celle-ci en relation avec une ligne hiérarchique 2, comprenant les informations " centres de travail " et, pour chaque occurrence de l'un de ceux-ci, les " identifications " des opérations susceptibles d'y être effectuées, chacune de celles-ci pointant vers l'occurrence correspondante du segment OPERATION (figure 2.5).

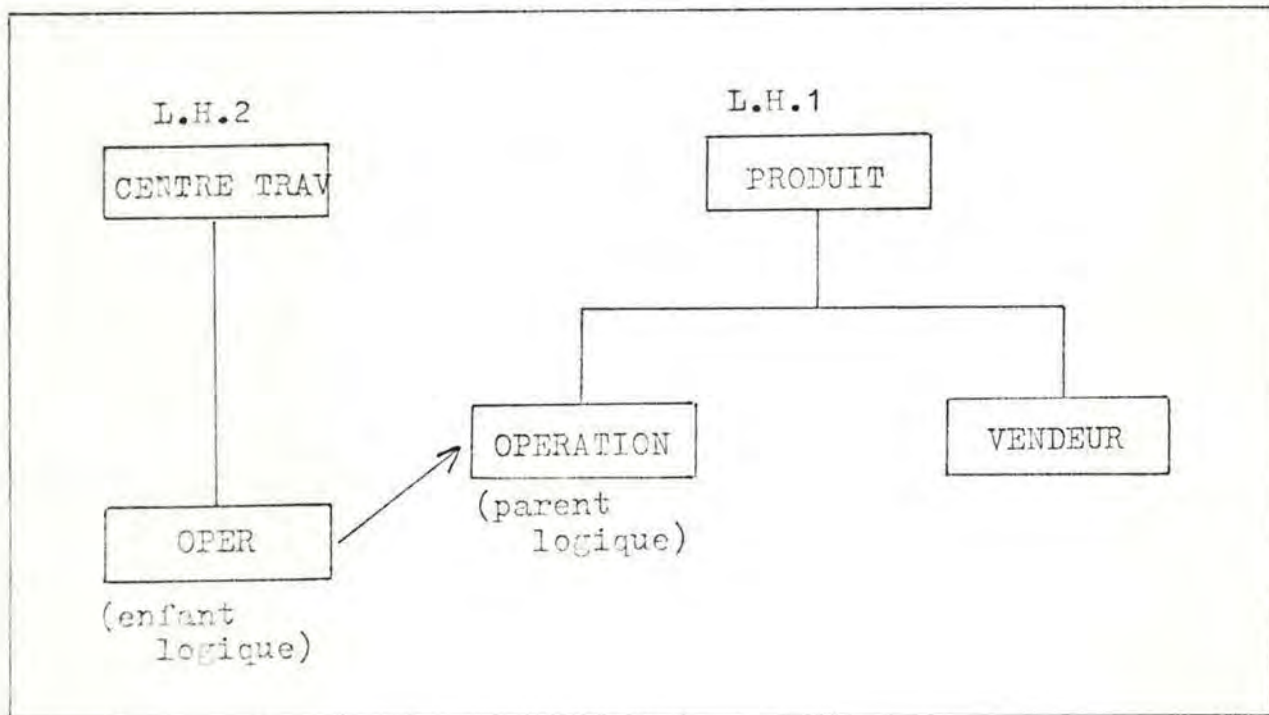


figure 2.5

Dans ce cas précis, on recourra aux services du programme DBXGEN pour qu'il crée un index constitué des clés concaténées des occurrences du parent logique, clés destinées à alimenter les " fields ", prévus à cet effet, des occurrences de l'enfant logique.

La traduction d'une telle relation au niveau des cartes de contrôle fournies à DBTABLE et relatives à L.H.1 est réalisée par la codification d'une carte de classe 6 contenant, en regard de chaque segment " indexé " (ou parent logique, par exemple OPERATION), le numéro de référence et le nombre maximal d'occurrences concernant la distribution " relative " des enfants logiques (nous entendons par là : " sous le contrôle du parent logique"), ainsi que le nom d'index identifiant la portion du fichier, constitué par DBXUPDTE à partir de celui produit par DBXGEN, qui contiendra les clés concaténées des occurrences du parent logique. Pour chacune de ces occurrences, DBXGEN utilise la fonction de distribution de la carte de classe 6 correspondante pour déterminer combien d'enregistrements, contenant les clés concaténées de cette occurrence, il devra produire. Soulignons ici que, pour une relation logique donnée, le nombre de clés écrites par DBXGEN sur le fichier séquentiel qu'il produit doit être aussi proche que possible du nombre total d'occurrences de l'enfant logique effectivement créées, par déduction de la distribution donnée sur la carte de classe 1 de définition de celui-ci dans la ligne hiérarchique à laquelle il appartient. Si ce n'est pas le cas et que, par exemple, le nombre d'enregistrements du fichier des clés est nettement plus grand que le nombre d'occurrences présentes d'enfant logique, on s'expose à l'obtention de résultats indésirables, dans la mesure où la distribution effective des enfants logiques dans la base générée ne sera pas celle qu'on escomptait, vu qu'une partie seulement des clés générées serviront lors du chargement.

B. Pour expliquer l'indexage secondaire, considérons le cas d'un segment PERSONNE contenant un numéro matricule (index primaire), associé à un nom de personne et à divers renseignements concernant cette personne. On peut vouloir accéder à ceux-ci sans passer par l'index primaire, mais en donnant simplement le nom de la personne en question. Ceci revient à créer une relation d'indexage secondaire du segment, à partir du field " nom de personne ". En règle générale, on introduit les notions de segments source et cible de l'indexage secondaire, qui, dans notre petit exemple, ne sont qu'un seul et même segment.

L'exécution du programme DBXGEN sera requise, ici, pour qu'il construise un index de valeurs destinées à alimenter les segments sources de l'indexage. Cette exécution est " commandée" par la présence d'une carte de classe 7, nommant le segment source et donnant, outre un nom d'index, une distribution exprimant, comme sur les cartes de classe 6, la manière dont devront être générées les valeurs d'indexage (par exemple, 20 % des valeurs " disponibles" alimenteront 80 % du nombre total d'entrées de l'index, et donc aussi du nombre d'occurrences du segment source).

Dans la double optique de ce qui précède, la tâche du programme DBXGEN consiste donc en l'exploration de la série des cartes de classes 6 et 7 et en la constitution d'un fichier séquentiel avec les résultats de la génération des clés concaténées de parents logiques et de valeurs sources pour indexage secondaire, les résultats étant mis en rapport avec le nom d'index leur correspondant. Un tri des grandeurs générées, avec un nombre aléatoire comme indicatif, a lieu ensuite : ceci évite que, dans le cas des relations logiques, par exemple, la suite des enfants logiques soit alimentée en clés concaténées, lors du chargement, de façon exactement " parallèle" à celle des parents logiques.

La tâche de DBXUPDTE consiste, pour l'essentiel, en la transformation du fichier séquentiel produit par DBXGEN en un fichier à accès direct (INKEYS), contenant un " annuaire " de " membres " (un " membre " correspondant à une série de clés ou valeurs générées par DBXGEN), chacun de ceux-ci étant identifié univoquement par un nom d'index.

Les " fields" d'enfants logiques destinés à contenir les clés concaténées, comme ceux des segments sources d'indexage secondaire destinés à contenir les valeurs d'indexage, sont dits de type externe; la carte de type 3 leur correspondant contiendra un nom d'index, référant le " membre " du fichier INKEYS dont devront être extraites les valeurs destinées à alimenter ces " fields".

2.2. L'aide au design

2.2.1 Approche théorique

Dans notre esprit, un outil d'aide au " design " joue un rôle essentiellement " documentaire ". Préalablement ou parallèlement à l'écriture des programmes d'application, le concepteur peut procéder à des tests afin de sélectionner la configuration de banque de données la mieux adaptée à ses besoins. Pour cela, il importe que l'outil d'aide dont il dispose lui permette, une fois créée une base de test, d'exploiter celle-ci par le biais d'ébauches de programmes d'application, et lui fournisse une série d'informations (principalement de type " performances " ou " statistiques ") concernant ces brèves séquences de traitement (figure 2.6).

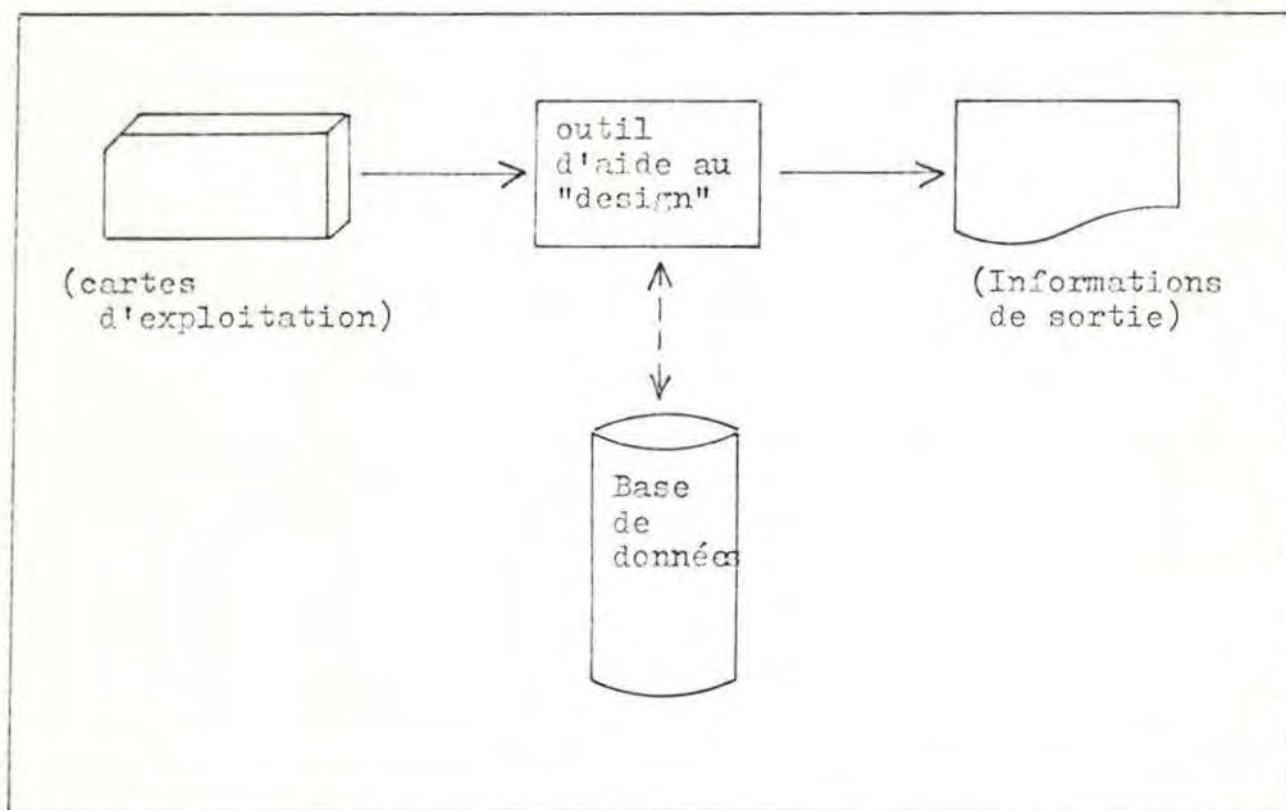


figure 2.6

2.2.2 Réalisation pratique

Nous allons reparler ici de DBPROTOTYPE, et, plus précisément, des programmes DBPROC et DBCALL qui, tous deux, exécutent des séquences d'"appels" plus ou moins complexes sur la base de données chargée par DBLOAD (à cet effet, ils consultent, le cas échéant, le fichier INKEYS produit par DBXUPDTE) et fournissent à l'utilisateur divers diagnostics en découlant.

Les " appels " auxquels nous avons fait allusion sont la concrétisation de différentes formes d'accès que l'on peut vouloir réaliser sur une banque de données, et qui se traduisent, chez IBM, par les fonctions standard suivantes:

- GET UNIQUE (GU) : pour retrouver un segment d'information unique de la base (exemples d'utilisation : accès "random" ou localisation de l'endroit où un "traitement" séquentiel va commencer);
- GET NEXT (GN)// GET NEXT WITHIN PARENT (GNP) : provoquent un " déplacement " séquentiel dans la base pour retrouver le segment d'information suivant; GNP limite la recherche aux segments dépendant d'un parent donné;
- GET HOLD UNIQUE (GHU)/ GET HOLD NEXT (GHN)/ GET HOLD NEXT WITHIN PARENT (GHNP) : pour retrouver un segment d'information qui sera effacé ou remplacé (mis à jour);
- INSERT (ISRT)/ DELETE (DLET)/ REPLACE (REPL): pour insérer, effacer ou remplacer (mettre à jour) un segment d'information dans la base.

2.2.2.1 DBPROC

Incontestablement, DBPROC est le moins évolué des deux programmes dont nous venons de parler. A notre avis, les résultats qu'il produit sont plutôt limitatifs, et restreignent son utilisation à deux cas précis :

- pour " valider " une base de données (nous entendons par là, s'assurer que tous les segments sont accessibles, ce qui est, évidemment, primordial);
- pour déterminer rapidement laquelle, parmi un nombre important d'alternatives conceptuelles, mérite une investigation plus poussée.

Ces résultats sont de deux ordres, outre le temps de début et de fin d'exécution :

- une table contenant, par segment, le nombre d'appels de chaque type achevés avec succès et le nombre de ceux s'étant mal terminés
- des statistiques sur le " pool " des buffers de la base.

Ce sont, une fois encore, les cartes de contrôle fournies à DBTABLE qui contiendront les séquences d'appels définies par l'utilisateur, plus exactement les cartes de classes 4 et 5.

Les premières contiendront, repérés par une lettre, les groupes d'accès à réaliser.

L'utilisateur indiquera, si nécessaire, pour chacun d'entre eux, l'argument de recherche du segment concerné (Segment Search Argument, SSA), qui peut apparaître sous différentes formes (éventuellement accompagné d'un nom d'index), comme le suggèrent les exemples suivants:

- (groupe A) GN - ;
- (groupe B) GU NAME *xxxx* (AGE *xxxxxx* = 000040) provoque la recherche d'un segment NAME dont un field AGE vaut 40 ;
- (groupe C) GU ROOT *xxxx* (ROOTKEY *xx* = @@@@@@)

La présence de " @ " indique à DBPROC qu'il doit générer lui-même un field clé de six bytes destiné à servir d'argument de recherche avant d'entamer celle-ci ;

- GU INDEX 1 DEP1 *xxxx* (@@@@@@@@@@@@@@)

Cette fois, la présence d'un nom d'index et de " @ " dans le SSA indique à DBPROC qu'il peut compléter ce SSA en remplaçant les " @ " par une clé concaténée sélectionnée aléatoirement parmi le groupe de celles produites par DBXGEN pour l'index INDEX1. (Ces clés auront été générées suite à la présence

d'une carte de classe 6 renseignant, en regard d'INDEX 1, le segment indexé DEP 1).

Il s'agit donc ici de l'exemple annoncé d'utilisation supplémentaire possible de DBXGEN. La possibilité de citer un nom d'index dans l'argument de recherche garantit à l'utilisateur que cette dernière sera fructueuse, nous entendons par là qu'il existe réellement une occurrence de DEP1 possédant la clé qui vient d'être générée, ce qui n'aurait vraisemblablement pas été le cas si l'utilisateur avait, lui-même, "imaginé" une clé.

En plus des fonctions standard d'accès que nous avons renseignées, DBPROC autorise l'utilisateur à recourir à deux fonctions supplémentaires lui permettant de simuler un "traitement" qu'il ferait subir aux données qu'il vient d'extraire de la base.

Ces fonctions sont PROC et WAIT. La première entraîne l'écoulement d'une milli-seconde de temps CPU (simulation du déroulement de la suite d'un programme d'application venant d'accéder à la base); la seconde provoque l'entrée, durant une milli-seconde, de DBPROC dans un état d'attente, correspondant au statut d'un programme d'application pendant la réalisation physique d'une opération d'entrée/sortie non supportée par IMS.

Alors que les cartes de classe 4 identifient, à l'intention de DBPROC, les groupes (" sets", [5]) de fonctions à réaliser, les cartes de classe 5 définissent les séquences de répétition de ceux-ci.

groupes de

Par exemple, en reprenant, en guise de références, les fonctions définies précédemment :

- * GB * A entraîne la génération d'appels " GN " jusqu'à ce que le code d'état " GB " (signifiant la fin physique de la base de données) soit renvoyé. Une telle séquence répétitive provoquera donc le balayage séquentiel de la banque de données toute entière;

- 100 (5 B, 10 C) entraînera la génération de : cinq fois l'accès défini dans B; puis dix fois ceux définissant le groupe C (avec des clés d'accès différentes à chaque fois); le tout répété à cent reprises.

Il ressort de tout ceci que DBPROC est véritablement un outil d'aide très élémentaire : les séquences d'accès y sont plutôt embryonnaires, réduisant à leur plus simple expression les possibilités offertes à l'utilisateur de simuler certains " aspects" des programmes d'application.

2.2.2.2. DBCALL

Cet outil est nettement plus complet que le précédent, en ce qu'il procure à l'utilisateur des informations ("temporelles" et statistiques) beaucoup plus détaillées que DBPROC, et lui permet par conséquent d'évaluer complètement différentes alternatives de conception de bases de données.

A cet effet, DBCALL présente son propre langage à la disposition de l'utilisateur: la possibilité lui est offerte, entre autres, d'effectuer des branchements dans des séquences d'accès, branchements pouvant, d'ailleurs, être conditionnés " statistiquement" (ce qui n'existait pas dans le cas de DBPROC, les séquences étant purement "statiques ").

Plusieurs exécutions répétitives de DBCALL devraient permettre à l'utilisateur de déterminer l'impact, sur la durée de réalisation des accès, de paramètres tels: taille des " buffers", dimension des enregistrements et des blocs, méthode d'organisation des données dans la base, options de constitution de pointeurs possibles, etc.

Nous ne pouvons toutefois pas nous étendre en long et en large sur la présentation de cet outil, aussi intéressant soit-il : cela nous prendrait trop de place. Disons toutefois que le langage proposé est orienté entités: il s'agit là des objets " manipulables " du

langage, localisés en mémoire, sous forme de valeurs (définies par le système ou spécifiées par l'utilisateur), et possédant un ou plusieurs attributs, décrivant les caractéristiques de chaque entité. Par exemple, une entité " SAVE LOCATION " est repérable par le mnémonique SVn, où n est l'attribut " numéro d'identification". Les attributs d'entités définies par le système ne peuvent qu'être accédés par le programmeur, qui ne peut donc les modifier, ce qu'il est, par contre, susceptible de faire sur les attributs de " ses propres entités". Précisons néanmoins que lorsque nous parlons de " ses propres entités", nous risquons de créer une ambiguïté, puisque les entités sont des objets constitutifs (pré-établis, tout comme les attributs de ces entités) du langage-source de DBCALL et qu'il n'est donc nullement question, pour l'utilisateur, d'en imaginer et d'en déclarer de nouvelles. Il y a simplement deux ensembles disjoints d'entités, existant une fois pour toutes: celles à usage du système, mais accessibles par le programmeur, et les autres, à usage de ce dernier, qui précise, dans des instructions de spécification, lesquelles il a l'intention d'utiliser; il assigne aussi, à certains de leurs attributs, des valeurs initiales.

Le corps du programme consiste alors en une suite d'instructions de manipulation des entités et de leurs attributs (compte tenu des restrictions dont nous avons parlé), entrecoupées d'instructions de réalisation d'accès à la base, de branchement, etc. A vrai dire, ce programme se présente comme une simulation, parfaitement contrôlée par l'utilisateur, du comportement d'un programme d'application sous l'angle exploitation de base de données. Le langage laisse au programmeur le loisir d'emmagasiner, au long de l'exécution et selon ses besoins, les informations qui lui semblent pertinentes et qui lui seront restituées sur le listing de sortie.

Voici, en guise d'illustration, quelques exemples d'entités-système :

- . Current Instruction Register - CIR : contient le n° de l'instruction en cours;

- Elapsed CPU Time - CPU : contient le temps CPU écoulé depuis le début de l'exécution;
- Input/Output Area : la zone d'entrée/sortie en question est celle utilisée pour la réalisation des fonctions d'accès standard déjà évoquées;
 - IOA (n,m)- permet de référencer chaque field alphanu-
mérique de la zone d'entrée/sortie, avec

n = position de départ	}	du field;
m = longueur		

variantes:

- IOAB(n,m) - permet de référencer(en vue d'une modifica-
tion)un field binaire;
- IOAP(n,m) - permet de référencer(en vue d'une modifica-
tion) un field packed;
- IOAZ(n,m) - permet de référencer(en vue d'une modifica-
tion) un field zoned;
- Random Number Generator - R(a,b) . demande la génération d'un nombre entier uniformément distribué entre a et b, où a et b sont des attributs "quelconques";
- Return Instruction Register -RIR-: contient le n° d'une instruction où le programmeur désire brancher après l'exécution d'une " sous-routine " en cours;

etc.

A côté de cela, quinze compteurs ("buffer pool statistics counters", [5]) contiennent en permanence des informations concernant par exemple, le nombre total de " requêtes", éventuellement satisfaites sans entrée/sortie physique, le nombre de lectures, de mises à jour, etc.

Quant aux entités-utilisateur, les plus essentielles nous semblent être:

- les tables de distribution : des données y seront enregistrées en cours d'exécution, converties automatiquement en distributions statistiques et restituées avec moyenne et écart-type;

• les fonctions, définies par des suites de couples (argument, valeur de la fonction) et référencées par le mnémonique FN (n,m) où n = n° de la fonction et m = valeur de l'argument.

Sans oublier les tables de références (suite séquentielle d'entrées accessibles et modifiables en cours d'exécution), " Segment Search Arguments", variables (expressions arithmétiques constituées d'attributs d'entités DBCALL), constantes, zones de sauvegarde pour valeurs d'attributs (SAVE LOC's) etc .

Le listing de sortie de DBCALL contient, outre le programme-source (avec les messages d'erreurs éventuels) et une " cross-reference", ainsi qu'un " dump " (optionnel), la restitution des contenus de SAVE LOC's, des tables de référence et de distribution etc...

La présentation que nous venons de faire illustre à suffisance, nous semble-t-il, la richesse de DBCALL, qui constitue indiscutablement un outil d'aide fort précieux au développement d'applications mettant en jeu des bases de données.

2.3 Tests de bon fonctionnement - Contexte batch

2.3.1 Approche théorique

Une fois les programmes d'application écrits et l'environnement extérieur (messages d'entrée et fichiers de test) constitué, les premiers essais peuvent débuter, qui devraient permettre au programmeur de cerner les imperfections des programmes d'application qu'il vient de rédiger. Les essais sont dirigés par le programme de contrôle d'essais, dont nous avons déjà parlé, et qui est responsable de la lecture des messages d'entrée, de l'ordonnancement éventuel des programmes d'application en conséquence, via les services du programme de contrôle opérationnel, et de la restitution, en sortie, des messages - réponses (figure 2.7).

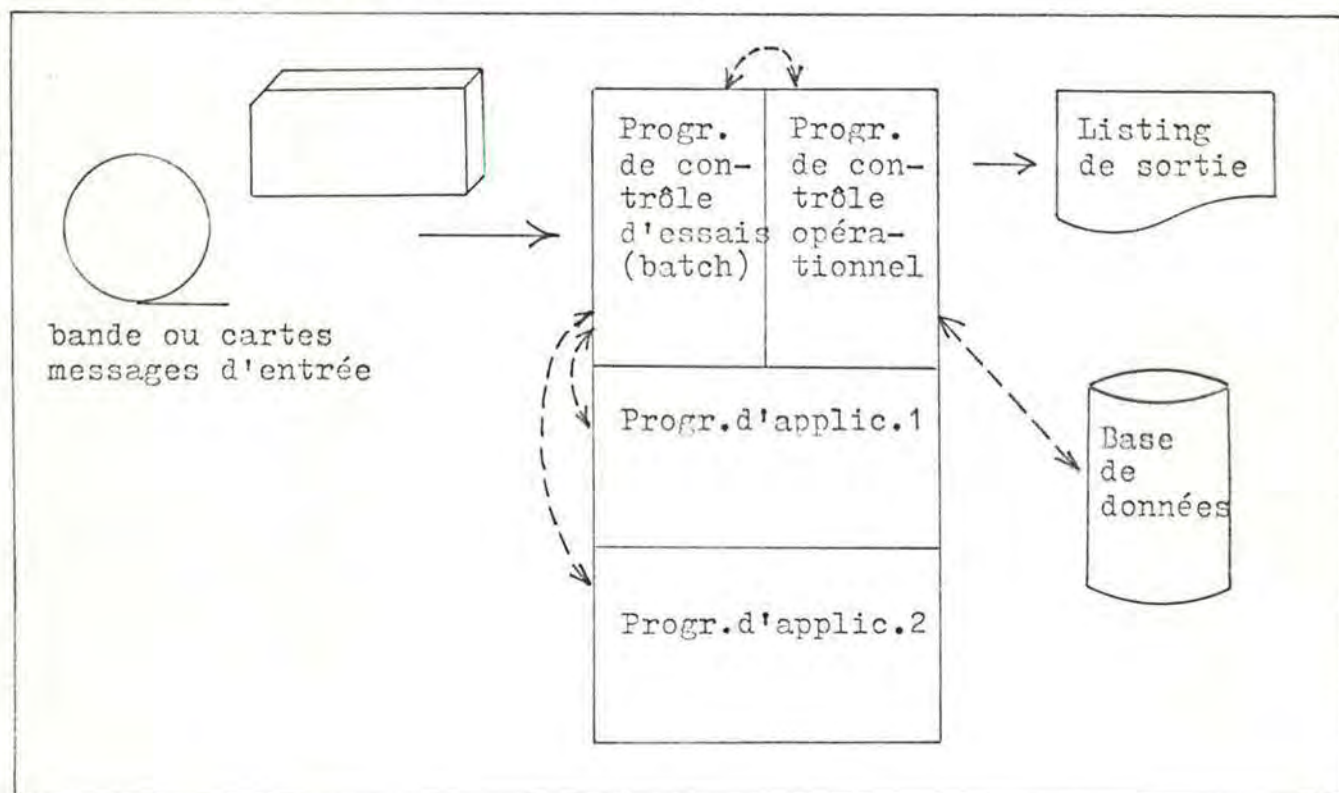


figure 2.7

Plutôt que de passer au crible, d'un point de vue théorique, la constitution interne du programme de contrôle d'essais, nous nous proposons maintenant d'examiner le fonctionnement de produits existants, liés aux deux systèmes IMS ou CICS.

Au préalable, nous invitons instamment le lecteur à prendre connaissance de la description des caractéristiques principales de chacun de ces deux systèmes, figurant en annexe 2.1.

2.3.2 Réalisation pratique

Les produits existants que nous allons présenter sont des outils de test liés intimement à l'un des systèmes IMS ou CICS.

2.3.2.1 Batch Terminal Simulator (BTS), sous-produit d' IMS

Il s'agit d'un programme "batch", simulant l'activité de télétraitement d'IMS, tout en procurant à l'utilisateur une série d'informations sur le fonctionnement de ses programmes d'application, informations résultant tant de l'interception des interactions entre programmes et bases de données que de la simulation des " appels TP " traditionnels.

Tous les langages de programmation supportés par IMS le restent par BTS et, de plus, les activités du simulateur sont totalement " transparentes " aux applications, ce qui signifie qu'aucune modification, tant dans le code des programmes que dans la constitution des différents blocs de contrôle (du type TP PCB's, dont nous parlons en annexe) n'est nécessaire. La condition que nous posons au début de la deuxième partie, à savoir que les programmes puissent se dérouler, pendant les essais, de la même façon qu'au cours de l'exploitation ultérieure, est donc pleinement vérifiée. Les entrées fournies au simulateur, figurant sur cartes, sont de deux types :

- des commandes, qui sont des instructions de contrôle, indiquant au BTS, notamment :
 - les transactions concernées et les programmes d'application à tester;
 - le terminal logiqué simulé;
 - le langage de programmation utilisé;
 - les symboles de fin-de-segment et de fin-de-message;
 - les "aides au debugging" souhaités: impression des SSA's des accès base de données, de la file d'attente de messages et de statistiques concernant le déroulement des

programmes d'application (principalement, un compte rendu du nombre d'appels à IMS, rangés par code PCB et par fonction d'accès) et, pour les programmes conversationnels, restitutions de la SPA;

- les messages d'entrée ("simulator statements", [10] , chacun de ceux-ci représentant une ligne d'informations en provenance d'un terminal).

La structure du BTS est décrite sur la figure 2.8 :

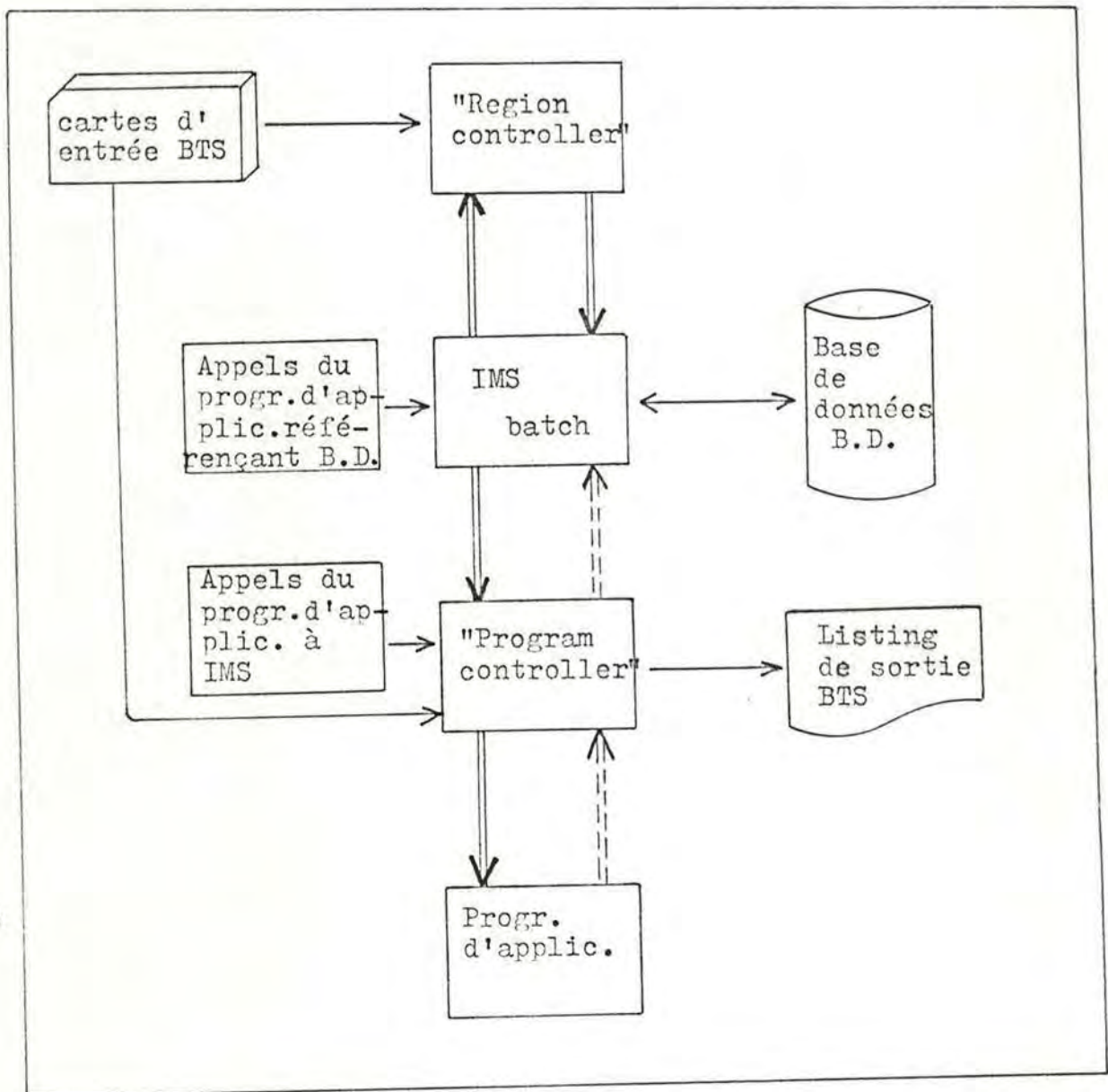


figure 2.8

==== : flux indépendant de la logique des programmes d'application
 ===== : flux dépendant " " " " " " "

Le "region controller" joue essentiellement un rôle d'ordonnancement des programmes à tester dans la région batch, en utilisant les services des "features" (d'ordonnancement, de chargement et de lancement) d'IMS dans celle-ci.

Cet ordonnancement intervient à deux niveaux :

- une transaction "primaire" peut en "générer" d'autres, secondaires, dont les programmes d'application devront donc être ordonnancés;
- plusieurs programmes d'application, correspondant à des transactions différentes, peuvent être testés en même temps par différents utilisateurs à l'aide de l'outil BTS, ce qui nécessitera aussi un ordonnancement.

Dans ce dernier cas, toutefois, il convient que chaque utilisateur ait constitué, au préalable, sa propre base de données de test. Ceci provient du fait que les programmes d'application se déroulant habituellement en "batch" sont théoriquement indépendants les uns des autres, en ce sens qu'ils travaillent sur des fichiers différents; il devrait donc en aller de même des programmes TP soumis à des essais en environnement "batch", pour la bonne raison qu'il n'existe pas, au niveau d'IMS-"batch", de méthodes de protection interdisant, par exemple, un accès simultané à un segment de base de données par plusieurs programmes concurrents. On comprend donc aisément que si tous les programmes-utilisateurs accèdent, en même temps, à une seule banque de données de test, on risque d'aboutir à des catastrophes.

Le "program controller" intervient à un niveau plus local, celui de la structure de chaque programme d'application : il intercepte tous les appels de celui-ci, transmet à IMS ceux d'entre eux qui référencent une base de données (leur "réalisation" est la même, que l'on soit en "batch" ou en TP) et, pour les autres, concernant exclusivement les messages simulés, effectue diverses opérations analogues à celles qui sont habituellement le lot du programme de contrôle TP d'IMS (en particulier, la manipulation simulée des

TP PCB's et l'impression des réponses de chaque programme).

Le simulateur BTS dont nous venons de parler s'est vu adjoindre un " feature " élargissant son action, en permettant la simulation des entrées et sorties de messages provenant de terminaux 3270. Ces unités sont des écrans de visualisation permettant l'établissement de conversations avec l'ordinateur, et différenciant des terminaux de type machine à écrire par de nombreux points: le support d'information est à deux dimensions (un axe ligne et un axe colonne) alors que les productions d'une machine à écrire sont purement linéaires, et les touches du clavier y sont beaucoup plus nombreuses (à côté de la traditionnelle touche " ENTER " d'expédition d'un message, présence de touches fonctionnelles multiples et aussi possibilité d'utilisation du crayon sélecteur ou "selector pen " sur l'écran).

Le " BTS 3270 formatting feature " (3270 FF-c'est le nom donné à cette extension de BTS) met en jeu les différents blocs de contrôle du MPS contenant les informations de service relatives au " formattage " de l'écran. Quant aux " simulator statements ", leur spécification contiendra des indications de lignes et colonnes donnant la position de chaque message sur l'écran, accompagnées des codes - fonctions d'introduction habituellement exprimés par l'enfoncement d'une des touches du terminal.

Le listing de sortie contiendra notamment les " photographies " successives de l'écran de visualisation tout au long de la simulation, bâties et imprimées à partir d'une image interne de l'écran, figurant en permanence en mémoire centrale, où elle est maintenue à jour par le 3270 FF, et où apparaissent les caractères de contrôle de chaque "field" constituant l'écran (ceux-ci concernent l'intensité lumineuse, la "protection",etc ...des " fields").

Il nous a semblé intéressant de présenter un exemplaire de listing de sortie du 3270 FF; il occupe l'annexe 2.3 et

fait apparaître les renseignements suivants :

- un en-tête de simulation reprenant les cartes-commandes introduites par l'utilisateur : la première (./T) identifie le code-transaction et le nom du programme (MBR) à tester, et contient en outre une indication de taille de la SPA et du langage de programmation utilisé; la seconde (./D) identifie le terminal logique dont " proviendront" les simulations et donne quelques indications sur l'unité sous-jacente (il s'agit d'un 3270, modèle 1), ainsi que la taille d'un " buffer " destiné à contenir les blocs de contrôle du MFS, identifiés par le nom figurant dans la troisième carte-commande (/FOR).
- une succession d'instantanés de l'écran de visualisation de l'unité tel que le verrait l'opérateur du terminal, les caractères de contrôle en plus, alternant avec l'impression de la zone d'entrée / sortie et la restitution de la SPA entre chaque tranche de conversation.

Dans notre exemple, l'écran comporte douze lignes et quarante colonnes; la première photo est celle du " format" que recevrait l'utilisateur, assis au terminal, s'il venait de frapper le code-transaction (procédure initiale de connection à l'ordinateur): le programme d'application répond en expédiant à l'écran un tableau récapitulatif des renseignements dont il a besoin. Entre les lignes figurent les " attribute bytes " de chaque " field" constituant l'écran, dont la signification est donnée en-dessous de chaque image de celui-ci. Les signes " + " y apparaissant déterminent les zones de réponse que va devoir donner l'utilisateur.

La visualisation de ces renseignements est fort utile car elle restitue au programmeur l'état de l'écran dans ses moindres détails (il y a des zones d'intensité lumineuse forte ou faible, des zones " protégées", accessibles ou non avec le crayon sélecteur, etc).

Sur la même page apparaît (INPUT RECORD) l'impression du premier " simulator statement ", contenant la réponse de l'utilisateur au " format" qui vient de lui être présenté, réponse d'ailleurs " formattée " à l'écran sur la page suivante, telle que la recevrait le MFS d'IMS (ACTION =ENTER). Il s'agit d'une demande de recherche d'un individu possédant une identification donnée.

La page suivante contient une première restitution de la zone d'entrée / sortie et de la SPA, encore vierge à ce moment et appelée, par un " GU ", depuis le programme d'application qui a été lancé.

Le deuxième groupe d'impression du bas de la p. A2.3.3 constitue l'envoi de la réponse du programme à la requête de l'opérateur (code-fonction ISRT correspondant à une demande d'insertion de la réponse dans la file d'attente des messages de sortie); en l'occurrence, l'individu n'existe pas. Le " stockage " d'informations dans la SPA est ensuite commandé et cette dernière est restituée sur les pages suivantes, comme elle se présente à l'issue de la première tranche de conversation.

Ensuite, la p. A2.3.5 présente le nouveau format de l'écran, contenant la réponse du programme, avec les nouveaux " attribute bytes ", suivi du deuxième " simulator statement". Le processus d'échange se poursuit de la même façon, jusqu'à la fin de la simulation.

Le listing de sortie se termine par un rapport de statistiques concernant les types d'appels effectués par transaction et par code PCB, ainsi que l'indication de la portion réellement utilisée du " buffer " des blocs de contrôle MFS.

2.3.2.2 CICS 3270 Simulator

Comme son nom l'indique, ce simulateur " batch " permet à l'utilisateur de tester le bon fonctionnement des programmes d'application supportant les terminaux de type 3270.

L'idée de base dans la réalisation de ce simulateur est, non plus de " chapeauter " le programme de contrôle par un programme spécialisé, comme c'était le cas pour le " region controller " de BTS vis-à-vis d'IMS, mais d'emprunter des " déviations " (" exits", [12]) aux programmes CICS eux-mêmes.

Comme il s'agit d'un simulateur " batch ", le programme de gestion du réseau n'a plus sa raison d'être: il s'agira donc de simuler ses actions habituelles pour que le système global se comporte comme si de rien n'était.

Le fonctionnement du simulateur est résumé sur le schéma suivant :

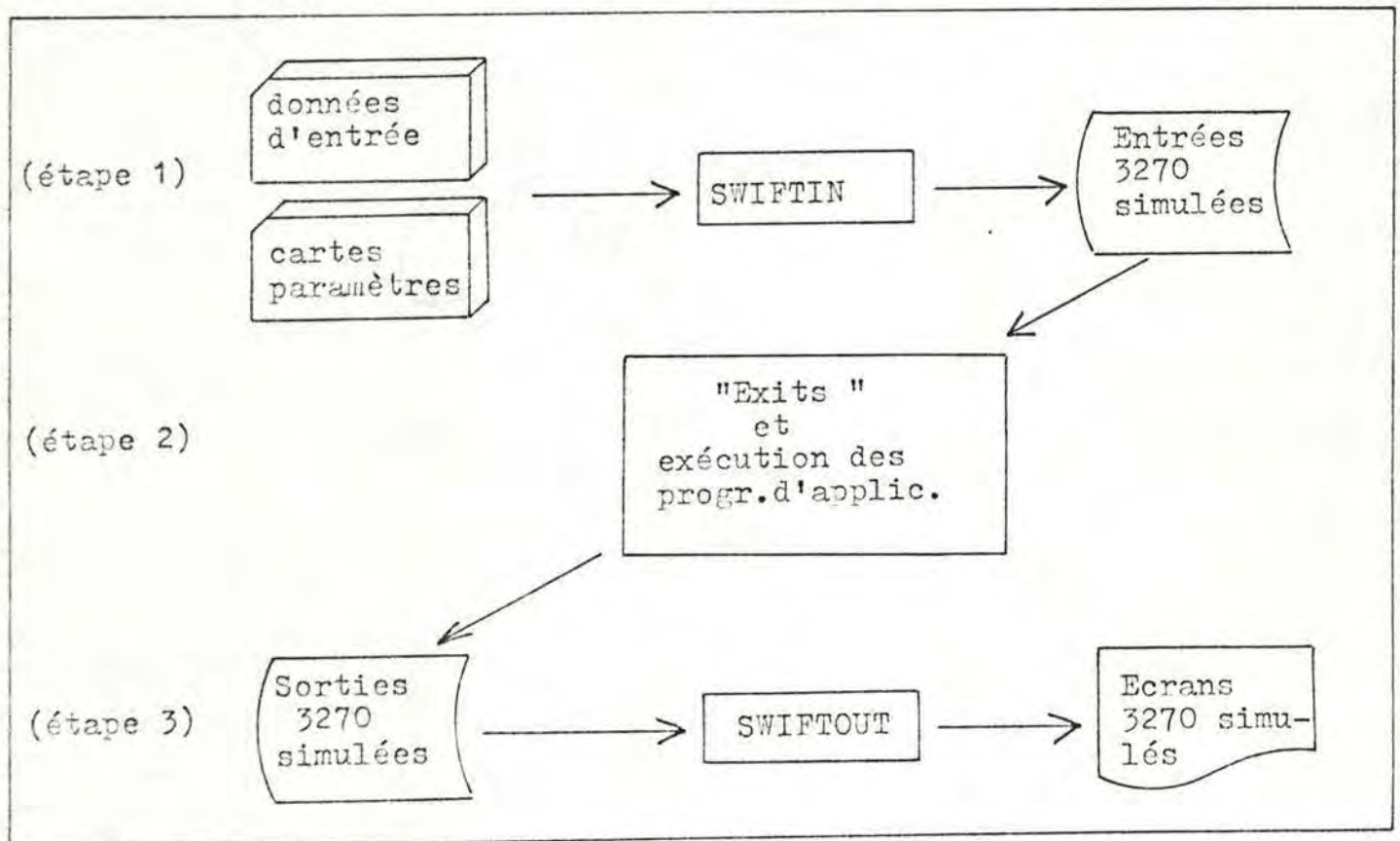


figure 2.9

Les cartes paramètres et les données d'entrée contiennent grosso modo les mêmes renseignements que les commandes et " simulator statements " du 3270 FF de BTS.

Dans un premier temps, le programme SWIFTIN produit, à partir de celles-ci :

- un bref listing de sortie reprenant les entrées telles qu'elles figurent sur les cartes fournies par l'utilisateur
- un fichier disque contenant tous les groupes ("streams", [12]) de données d'entrée, précédés d'un en-tête de cinq bytes comprenant principalement une lettre I (Input), le numéro de modèle du terminal simulé, ainsi que la représentation hexadécimale de l'action fonctionnelle souhaitée (ENTER, CLEAR, etc).

La seconde étape consiste en la lecture du fichier produit par SWIFTIN et l'exécution des programmes d'application, dans la région batch, sous le contrôle du PCP de CICS. Cette exécution est précédée par celle d'une " input exit ", réalisant la préparation, ^{semblable à celle} habituellement accomplie par le TCP, des données produites par SWIFTIN, en vue de leur présentation au programme d'application soumis à l'essai.

Cette préparation consiste principalement en l'extraction de l'en-tête de cinq bytes constitué par SWIFTIN hors de la zone d'entrée / sortie où figure le groupe de données qui vient d'être lu. Les renseignements apparaissant dans cet en-tête viendront garnir une partie de l'entrée correspondant de la table de contrôle des terminaux, tenue à la disposition du programme d'application; il est, en effet, fréquent que celui-ci désire tester le type de touche fonctionnelle " enfoncée " par l'utilisateur, pour s'aiguiller différemment en conséquence (c'est le but des touches PFK1 à 12).

Un indicateur est également positionné dans la TCTTE, signalant qu'il s'agit d'un terminal simulé.

Une deuxième routine de déviation de TCP ("attach exit ") vérifie si l'action souhaitée n'est pas " CLEAR ", l'utilisateur ayant simulé par là l'appui sur la touche du terminal demandant l'effacement de l'écran de ce qu'il vient de frapper auquel cas la lecture du groupe de données d'entrée qui vient d'être faite est ignorée, comme cela se produirait pour un véritable 3270. Si ce n'est pas le cas, la routine rend la main au TCP; celui-ci passe le contrôle au PCP, qui entre-

prendra le chargement et le lancement du programme dans la région batch.

Le déroulement du programme s'étant exécuté sans heurts, les résultats produits seront traités par une " output exit ", qui constitue un en-tête de cinq bytes devant chaque groupe de sortie, contenant principalement une lettre O (Output), le numéro de modèle et le type du terminal, ainsi qu'un caractère-code, " interprétable " par le programme SWIFTOUT, qui imprimera quelques notes d'explication indiquant l'état dans lequel serait le clavier s'il avait réellement reçu le message de retour : verrouillé ou pas, avec, c'est possible, la présence d'un signal sonore, etc.

Le programme SWIFTOUT, précisément, est responsable de l'édition, sur listing, des formats d'écrans tels que l'opérateur les aurait reçus s'il avait été au terminal, ainsi que de ceux, provenant du disque des entrées simulées, qui auraient été reçus par le système, les " messages " correspondants ayant été placés sur le disque des sorties par la " TCP input exit ".

Comme le permettait le BTS, ce simulateur rend possible le test du bon fonctionnement des programmes dans les conditions les plus diverses: ainsi, la quasi-réentrance des programmes d'application peut être vérifiée sans problème : il suffit d'exécuter plusieurs fois le programme SWIFTIN, à partir de données d'entrée différentes et pour un même programme, chaque groupe de données correspondant à un terminal simulé. En outre, le fait qu'il s'agisse d'un produit " batch " (constitué des programmes SWIFTIN et SWIFTOUT, ainsi que des différentes " exits ") n'est pas dédaignable, car il permet le déroulement des programmes testés dans une région séparée, ce qui rend efficace l'instauration de méthodes de protection.

2.3.3. Quelques conclusions

La similitude du contexte d'utilisation de ces deux types de simulateurs justifiait leur présentation conjuguée et nous permet maintenant d'examiner conjointement les principaux avantages de l'un et de l'autre :

- en premier lieu, il s'agit de simulateurs " batch ", ce qui permet leur déroulement pendant qu'éventuellement le reste du système fonctionne normalement. En d'autres termes, si l'on ne se situe pas dans la période de conception des tout premiers programmes d'application mais que l'on procède simplement à un élargissement du " parc " des applications, on n'est pas tenu d'arrêter le système pour procéder aux essais des nouveaux programmes.

- le fait qu'il s'agit de simulateurs " batch ", dont le support d'entrée (cartes, mais ceci resterait valable pour un support bande) est bien constitué, autorise la " répétitivité " des tests. Nous entendons par là qu'après la détection, par le programmeur, d'une erreur révélée par les essais, une nouvelle simulation, avec les mêmes données d'entrée, fera apparaître nettement le résultat, sur les sorties, de l'action correctrice entreprise.

Dans le même ordre d'idées, si, pour une raison ou pour une autre, on est amené, au cours de l'exploitation du système, à revoir légèrement la conception d'un module CICS, par exemple, ou à modifier la logique de certaines applications, ces données d'entrée peuvent être " réutilisées " à volonté pour les tests du " nouveau système " ou des programmes modifiés; elles servent en quelque sorte de " références ", puisque, en comparant les nouvelles sorties avec les précédentes, l'utilisateur se rendra aisément compte de l'évolution subie par le système ou par les programmes d'application.

- l'analyse des résultats produits ayant lieu en aval de la simulation et " off line " (à l'instar de la préparation des données d'entrée, qui se déroulait, elle, en amont) n'en sera que meilleure; le programmeur peut, à tête reposée et donc de façon plus

efficace, examiner dans les moindres détails tous les aspects du fonctionnement des programmes qu'il a conçus et rédigés.

- enfin, les listings de sortie de l'un ou l'autre de ces simulateurs constituent, croyons-nous, une documentation intéressante à joindre au petit " dossier " accompagnant chaque programme d'application. Ils font apparaître, mieux que toute explication ne pourrait le faire, les fonctions exactes du programme, la manière dont il se comporte et répond à des événements précis.

2.4 Tests en temps réel

La dernière phase de la période des essais consiste, une fois installés les premiers terminaux du futur réseau, à tester les programmes d'application dans leur environnement " on line ".

En général, quel que soit le système de télétraitement utilisé, il s'agit toujours de préparer soigneusement les transactions et messages qui seront introduits aux terminaux, afin que les programmes à tester puissent s'exécuter dans un maximum de contextes différents.

Il est également souhaitable de continuer à travailler sur des fichiers de test, reprenant dans la mesure du possible, les mêmes types de données que les fichiers opérationnels, de manière telle:

- que chaque section de programme d'application soit testée;
- que les traitements correspondant à des erreurs ou à des conditions exceptionnelles susceptibles de survenir dans le fichier réel puissent aussi être testés.

Préalablement au lancement des derniers essais, l'utilisateur doit s'assurer que les conditions de bon fonctionnement du système sont remplies; en CICS, par exemple, il faudrait vérifier que chaque programme est bien " link edited " et répertorié dans la table des programmes (Processing Program Table), que chaque terminal utilisé correspond à une entrée dans la table de contrôle des terminaux, que chaque fichier figure dans la table de contrôle des fichiers (File Control Table) et est accessible par un programme s'exécutant " on line ", etc.

Notons encore qu'en CICS, les programmes d'application " on line " et les modules CICS n'étant pas protégés, des violations sont toujours possibles; selon le désir de l'utilisateur, elles seront court-circuitées et leur nombre simplement imprimé ou bien la première d'entre elles donnera lieu à la production d'un " dump " traditionnel.

2.5 Prolongements

En guise de prolongement au paragraphe précédent, nous nous proposons de passer en revue quelques outils d'aide au " debugging " utilisables " on line " et qui nous paraissent intéressants dans le cadre de ce travail.

2.5.1. Approche théorique

Les quelques idées théoriques que nous exposons ici sont extraites de l'ouvrage de Yourdon, [2]'. Elles concernent l'implémentation de ce que l'auteur appelle des transactions de test: il s'agit de transactions particulières d'aide à la recherche de solutions pour des problèmes (le mauvais fonctionnement de programmes d'application, notamment) survenus au cours de l'exploitation opérationnelle du système, l'objectif avoué étant de ne pas devoir arrêter celui-ci, de telle manière que les autres utilisateurs ne soient pas affectés par cette défectuosité passagère.

Il va de soi que l'utilisation de ces transactions serait réservée à un groupe de privilégiés.

Les transactions seraient traitées par un module spécialisé, comme le suggère le schéma simplifié suivant :

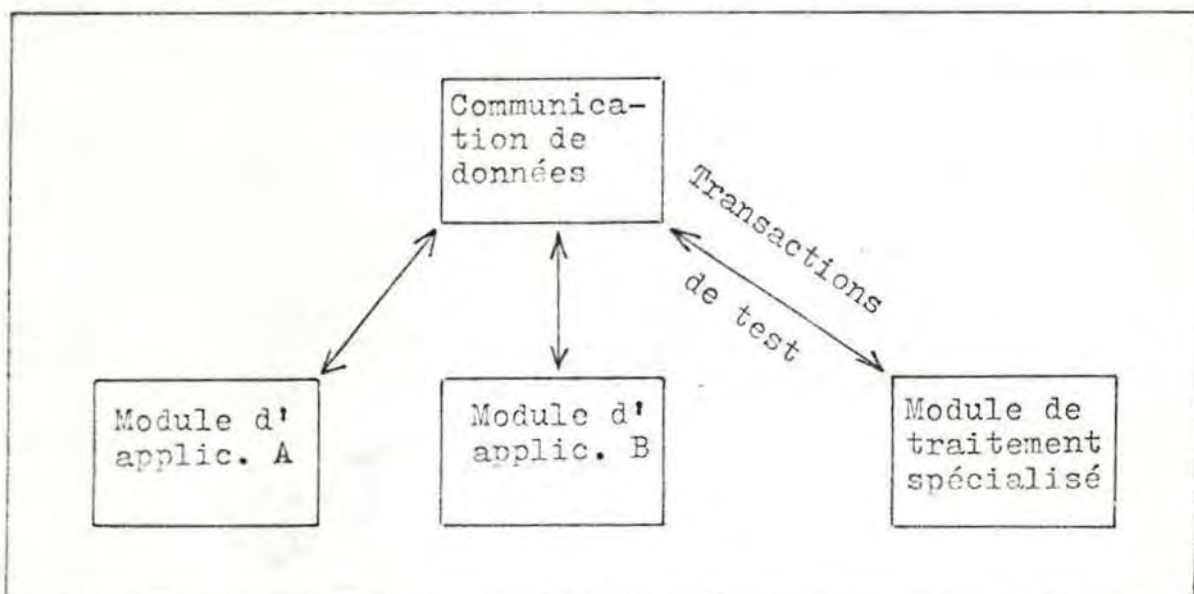


figure 2.10

Passons en revue quelques-unes des transactions dont Yourdon propose l'implémentation :

- EXAMINE : devrait permettre à l'utilisateur d'examiner les contenus, soit d'un enregistrement de base de données, d'une table, d'une file d'attente ou de tout autre groupe d'informations concernant les programmes d'application.
L'opérateur devrait toutefois être persuadé de l'éventuelle " obsolescence " des informations qui lui sont restituées au terminal: d'autres transactions, concurrentes de l'EXAMINE, pourraient avoir entretemps modifié certains éléments de données qu'il vient d'examiner.

- CHANGE : devrait permettre à l'opérateur de modifier dynamiquement un enregistrement, une entrée de table, etc.

Soulignons le danger provenant d'une utilisation abusive ou d'erreurs de la part de l'opérateur dans le processus d'introduction de ce type de transaction.

- TRACE : cette dernière, la plus difficile à implémenter, selon l'auteur, permettrait de suivre le déroulement d'un programme d'application ou de " superviser " les " appels " à certaines portions de base de données.

L'implémentation d'une transaction assurant le suivi des accès base de données ne serait possible que dans la mesure où tous les programmes d'application accèdent à celle-ci via un système de gestion commun. La survenance d'une telle transaction provoquerait alors l'envoi d'un message à ce gestionnaire, lui indiquant le ou les enregistrement(s) à " pister ", ainsi que le terminal de destination de l'output.

En ce qui concerne l'implémentation d'une transac-

tion analogue assurant le suivi du déroulement d'un programme, ce serait beaucoup plus compliqué: cette procédure impliquerait en effet la nécessité d'intervenir à l'intérieur d'un programme au moment de la compilation (l'insertion dynamique d'un " trace ", soit donc pendant l'exécution, étant encore beaucoup plus délicate).

Dans un ^{premier} temps, il serait certainement suffisant (ce serait aussi plus simple à implémenter) de suivre les appels d'un module d'application à l'autre ou les passages aux points d'entrée de sous-routines, étant entendu que ces différents transferts se fassent par l'intermédiaire d'un module de contrôle unique, ce qui nous ramène à la condition d'instauration de la transaction TRACE pour le suivi des accès-base de données.

Les autres transactions nous paraissent moins importantes: l'une d'entre elles concerne, par exemple, la déconnection(KILL) d'un terminal donné, à propos duquel il aurait été découvert que les programmes d'application ou les fichiers qu'il utilise causent des difficultés, et la notification (INHIBIT), à son opérateur, de cette déconnection. Les autres sont de la même veine.

2.5.2. Réalisation pratique

Le système CICS met à la disposition des utilisateurs " on line " de terminaux bien spécifiques (pour des raisons de sécurité évidentes) deux programmes d'aide au " debugging ", DFHBUG et DFHPAS, accessibles grâce à l'introduction au terminal de codes transactions spéciaux et qui, approximativement, réalisent certaines des fonctions que nous venons de mettre en exergue(transactions EXAMINE et CHANGE, en l'occurrence).

Le premier programme, DFHBUG, est activé pour réaliser l'impression ou la modification, sans qu'un " réassemblage " ou une " ré-édition de liens " ne s'avère nécessaire, d'une zone de

mémoire centrale, d'un bloc de contrôle, d'une table, voire d'un programme. Précisons d'emblée que les changements éventuels apportés à ces différents éléments n'ont qu'un effet limité, en ce sens qu'ils restent valables jusqu'au moment du prochain " rafraîchissement " (" refreshment", [14]) du système.

En clair, ceci signifie donc que, pour un programme d'application, par exemple, les mises à jour à apporter au code-source que l'on veut effectives (définitives) doivent être exécutées " off line ". DFHBUG permet aussi la restitution et / ou la modification (effective au plein sens du terme, cette fois) de portions de fichiers.

Le second programme, DFHPAS (Program Address Stop) permet l'interruption du déroulement d'une tâche CICS au beau milieu d'un programme d'application, ce qui offre la possibilité d'avancer pas à pas dans l'exécution; en outre, le recours, à chaque étape de la progression, aux services du programme précédent, permet la restitution des contenus de zones considérées comme " critiques ", comme il permet à l'utilisateur de " monitorer " l'exécution du programme interrompu, en lui offrant la possibilité de modifier ce dernier.

Après chargement automatique du programme, la commande d'interruption est stockée à l'adresse voulue sous forme d'un code opération invalide, l'adresse d'arrêt, ainsi que l'identification du programme étant mémorisées dans une table de DFHPAS. Lorsque, lors de l'exécution du programme, le code opération invalide est rencontré, une interruption est générée et un programme de traitement d'interruptions analyse celle-ci : dans le cas qui nous occupe (à savoir l'interruption a été déclenchée par la rencontre d'un code opération invalide placé intentionnellement et ne résultant donc pas d'une erreur d'écriture de l'utilisateur, auquel cas une suspension ABEND serait générée), un message est envoyé à l'utilisateur, lui indiquant que le programme a été interrompu. A partir de là, diverses options sont possibles:

l'utilisateur peut, notamment, appeler DFHBUG pour modifier le programme arrêté.

Pour conclure, on peut dire que l'implémentation d'outils d'"aide à la détection d'erreurs", tels ceux dont nous venons de parler, doit résulter d'un compromis entre, d'une part, la solution qui fournirait au programmeur des informations trop fragmentaires que pour qu'il puisse localiser la raison d'un mauvais fonctionnement, et, d'autre part, celle qui lui en restituerait trop, ce qui ralentirait le système de façon excessive et s'accompagnerait d'une impression trop longue.

Annexe 2.1 - Les systèmes IMS et CICS

Il s'agit là de deux systèmes fort complexes remplissant la fonction de contrôle dont nous avons parlé dans la première partie.

A.2.1.1. Le système IMS

L'originalité de ce système, par rapport au second, dont nous parlerons par la suite, réside, non pas dans le fait qu'il supporte aussi bien les applications " batch " que celles orientées TP, mais bien dans son architecture, basée sur un principe de partition de l'espace-mémoire disponible en régions (cf figure A2.1.1), ce qui assure une intégrité accrue à l'ensemble, puisque l'instauration de méthodes de protection d'une région à l'autre est possible:

- la région de contrôle, destinée à contenir les modules du système IMS;
- les régions réservées au traitement de transactions en provenance du réseau de terminaux (programmes d'application TP);
- "la" région réservée au traitement de " transactions " en différé (programmes " batch"), que nous considérerons comme " unique " (tout en sachant pertinemment qu'elle est, elle aussi, partitionnée), car elle ne nous intéresse guère dans le cadre de ce travail.

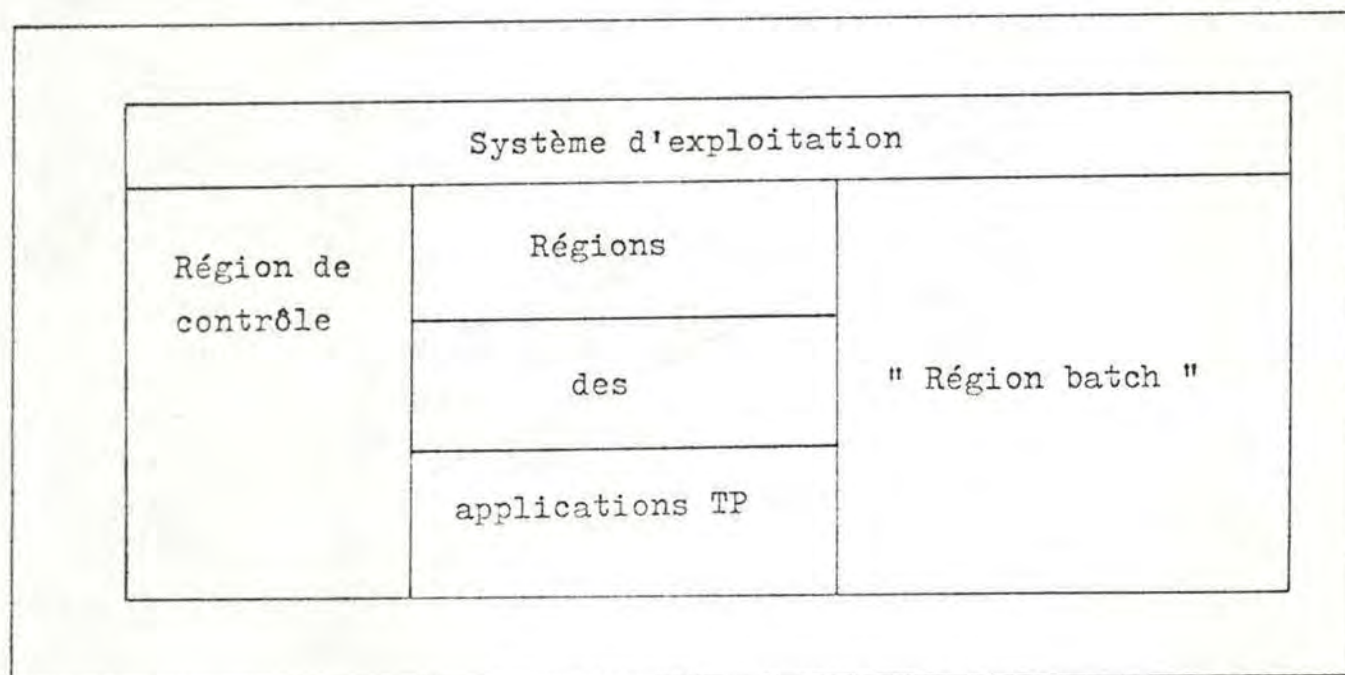


figure A2.1.1

Les services offerts par IMS en matière de gestion de base de données (" DB Management Services") assurent une parfaite indépendance des programmes d'application vis-à-vis de l'aspect physique des données figurant dans la base, et ce à partir de la description, par l'utilisateur, de toutes les structures logiques (c-à-d vues par les programmes d'application) de données hiérarchisées qu'il désire (l'unité d'information décrite étant le segment) et des relations établies entre ces données. Les outils de base relatifs à la création physique ainsi qu'à la réorganisation et au recouvrement éventuels des banques de données sont aussi fournis par IMS, sans oublier les fonctions d'accès et de maintenance des données, l'accès simultané (et donc concurrent) à une base de données par les programmes " batch " et " on line " étant chose possible. Cet aspect d'IMS concernant la gestion des bases de données est commun aux deux types de programmes d'application que nous avons distingués.

Quant aux services offerts par le système en matière de gestion des entrées / sorties d'informations en provenance des terminaux (" Data Communication ou DC Management Services "), ils assurent eux aussi l'indépendance totale des programmes d'application par rapport aux " unités " d'entrée ou de sortie : les programmes d'application référencent en effet des terminaux logiques (par l'intermédiaire de " Program Communication Blocks " ou TP PCB's), le système effectuant la conversion entre ceux-ci et le réseau physique en consultant les tables associatives qui auront été assemblées à cet effet lors du " lancement " du système. D'autres tables donnent les associations entre codes transactions et programmes d'application.

Le noyau IMS contient en outre un module particulier (" Message Format Service " ou MFS) qui réalise, à partir d'indications fournies par l'utilisateur et contenues dans des blocs de contrôle, le " formatage " des données de sortie du système à destination des terminaux de type écrans de visualisation, où la présentation de l'écran pose des problèmes assez épineux, puisque l'on éclate à deux dimensions des informations (messages) de substance "linéaire", ce qui n'est pas le cas pour des terminaux de type " machine

à écrire ", par exemple. Il va de soi que l'opération inverse de " dé-formatage " des messages d'entrée à destination des programmes d'application est également exécutée par le module MFS (figure A2.1.2.)

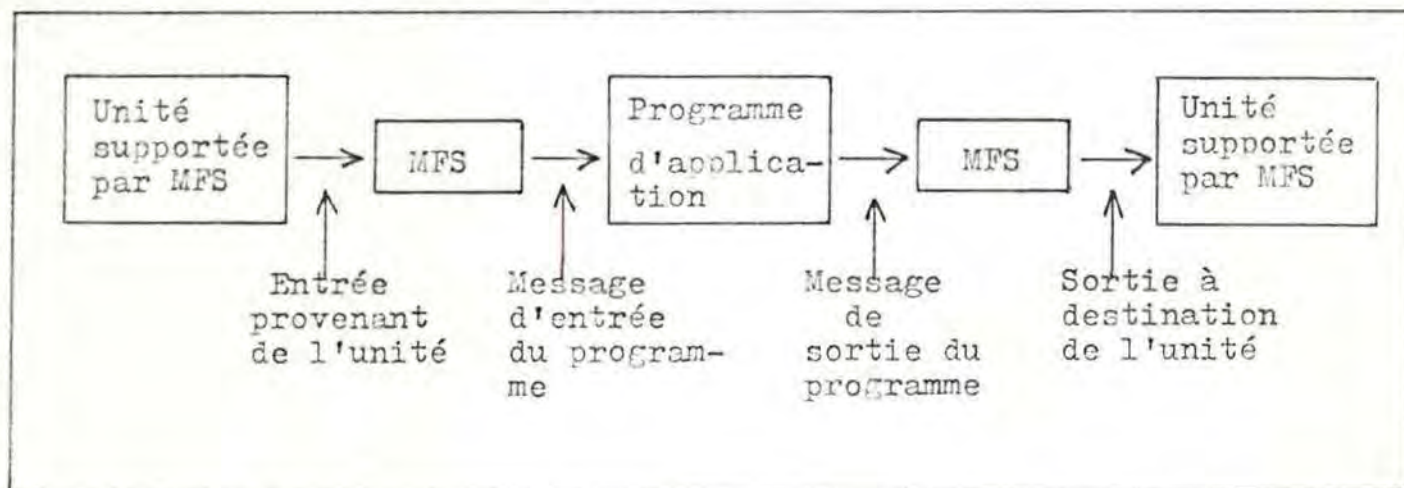


figure A2.1.2

Les entrées provenant d'unités de ce type contiennent des informations de contrôle intercalées parmi les données, indiquant, notamment, la position sur l'écran des éléments constituant ces entrées. MFS agit dès lors comme un interface, prélevant les caractères de contrôle en entrée, ou les insérant dans le flux de sortie, pour éviter que les programmes d'application perdent leur indépendance vis-à-vis de tout ce qui touche aux caractéristiques physiques des unités connectées, ce qui n'empêche que la logique des programmes d'application soit tout de même dépendante du type de terminal supporté: nous entendons par là que les possibilités offertes par des terminaux " machines à écrire " sont différentes de celles offertes par des écrans de visualisation.

En outre, le système IMS assure lui-même l'enfilement des messages d'entrée en attente (en mémoire centrale ou sur disque), rend possible l'articulation du réseau de terminaux autour d'un terminal "maître" (qui est en quelque sorte le " poste de commande" du réseau) ou encore la mise d'un terminal en mode " test" (nous entendons par là sa mise en état d'essai de fonctionnement, après

une panne, par exemple). Il permet aussi le déroulement de conversations entre l'ordinateur et un terminal : cette notion évoque la succession d'échanges d'informations de l'un à l'autre, mais où intervient un facteur " historique " d'un échange à l'autre. Il s'agit réellement d'un processus complexe de " questions-réponses " où la continuité de l'entretien est assurée par la mémorisation , au niveau de l'ordinateur, de renseignements concernant les phases antérieures de la conversation. Cette mémorisation fait partie de la logique du programme d'application sous-jacent; une zone de sauvetage, la "scratch pad area " (ou SPA) est utilisée à cet effet : elle contient donc, à un moment donné, toutes les informations, relatives aux échanges survenus au cours des étapes précédentes de la conversation, dont a besoin le programme pour s'aiguiller.

Pour ce qui est des programmes d'application, IMS autorise trois langages de programmation : Cobol, PL/I, Assembler. Dans chacun de ces trois cas, l'utilisateur fera appel aux services d'IMS en matière DB ou DC via le Data Language/I (ou DL/I) du système: il s'agit d'un important module IMS exécutant sur les banques de données ou les files de messages en attente les fonctions de base requises par l'utilisateur (ainsi, les fonctions d'accès évoquées au paragraphe 2.2.2 sont des fonctions DL/I. En guise d'illustration, nous présentons, en annexe 2.2, la façon dont peut être programmé un appel d'entrée de message.

Pendant toute la durée de fonctionnement opérationnel du système, les programmes d'application sont stockés en bibliothèque; IMS procédera en temps voulu à leur ordonnancement, en fonction des transactions reçues de l'extérieur, sur base d'un coefficient de priorité ou de la classe accordés à celles-ci ou encore en fonction d'un algorithme spécial; les programmes seront alors chargés, à tour de rôle, dans les régions adéquates.

De plus, un groupe important de " facilities ", comme disent les brochures de description, assurent tout ce qui touche:

- à la sécurité des ressources: que ce soit entre l'utilisateur d'un

terminal et les programmes d'application TP (système de mots de passe), ou entre ces derniers et les bases de données, les tentatives de violation des sécurités établies seront enregistrées ou notifiées au terminal " maître ", qui prendra les mesures requises;

- à l'intégrité des données: chaque programme d'application TP occupant une région de la mémoire centrale fait l'objet d'une protection; en ce qui concerne l'exploitation d'une base de données par des programmes d'application TP, des contrôles d'exclusivité d'utilisation au niveau des segments physiques sont effectués.

Enfin, le redémarrage de tout ou partie du système est facilité par l'existence de méthodes de recouvrement, à partir de " checkpoints " et de l'enregistrement ("logging ") des messages et des mises à jour de bases de données survenues.

La figure A2.1.3. schématise grossièrement les relations entre les éléments principaux du système et les programmes d'application " on line ".

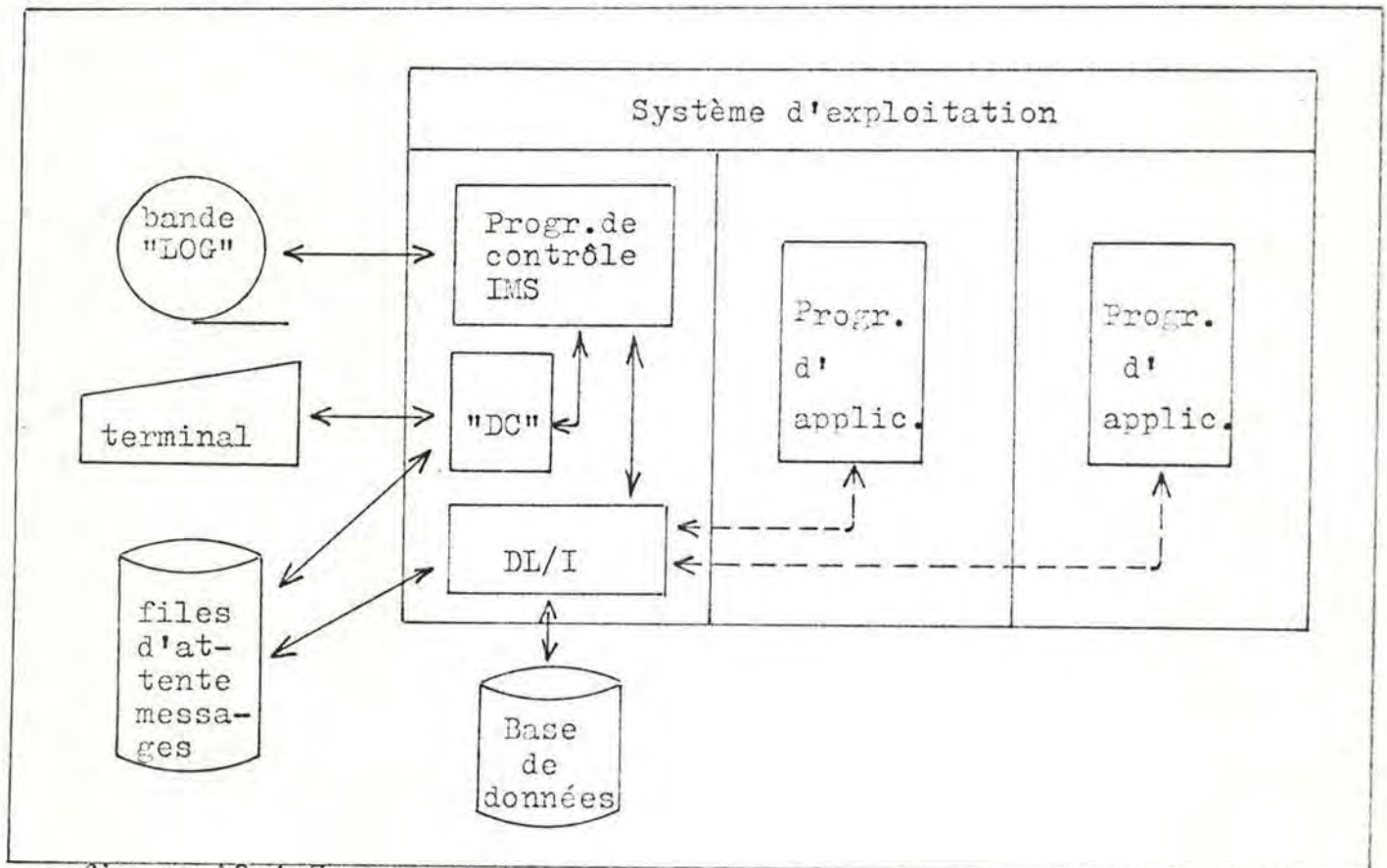


figure A2.1.3

A2.1.2 Le système CICS

ferait

A côté du précédent, le système CICS figure de " parent pauvre " s'il ne possédait, à peu de choses près, les mêmes éléments; mais alors qu'IMS opère conjointement sous les angles DB/DC (avec un langage de service commun à la disposition de l'utilisateur, à savoir DL/I), CICS n'était bâti , au départ, qu'en fonction de l'aspect DC, l'utilisateur faisant appel aux services du système en cette matière en recourant à des macros spécifiques au CICS. Ce n'est que par après que l'on se rendit compte de l'utilité d'adjoindre au système existant la possibilité de gérer des bases de données, le DL/I servant d'interface avec les programmes d'application.

L'architecture du système CICS est, aussi, profondément différente de celle d'IMS, en ce sens que le programme de contrôle CICS et les programmes d'application TP occupent cette fois une région unique, ce qui rend inopérantes les méthodes de protection qui existaient en IMS. Des " violations " (d'un programme d'application dans un autre, par exemple) sont donc possibles.

Pour ce qui est des services offerts par le système en matière de gestion des entrées / sorties d'informations en provenance de l'extérieur, la mise des messages en files d'attente n'est plus assurée par le CICS, mais laissée à la responsabilité de l'utilisateur.

Le module de gestion du réseau (Terminal Control Program ou TCP) et celui responsable de l'ordonnancement des programmes (Program Control Program ou PCP), d'ailleurs appelé par le précédent dès que l'opération de lecture initiale d'un message et de remplissage des " buffers " d'entrée est terminée, ont, par contre, sensiblement les mêmes fonctions que dans le système IMS.

Le " noyau " CICS contient, en outre, plusieurs tables, dont la principale nous semble être la table de contrôle des terminaux (Terminal Control Table ou TCT); chaque entrée de cette table concerne un terminal du réseau et renseigne les caractéristiques physiques de l'unité, en lui associant le nom logique utilisé par les programmes d'application, chacun de ceux-ci ne " percevant " donc

de la table que l'entrée qui lui correspond (Terminal Control Table Terminal Entry ou TCTTE).

Les dernières caractéristiques du système CICS sur lesquelles nous insisterons tiennent au fonctionnement des programmes d'application et se résument en deux termes, " quasi-multitasking " et " quasi-réentrance ", que nous expliquons brièvement ci-après.

A. Quasi - multitasking

Le phénomène à décrire fait intervenir la notion de " tâche ", que l'on peut définir de façon " indirecte " en disant que l'exécution d'une " tâche " est déclenchée par l'apparition d'une transaction.

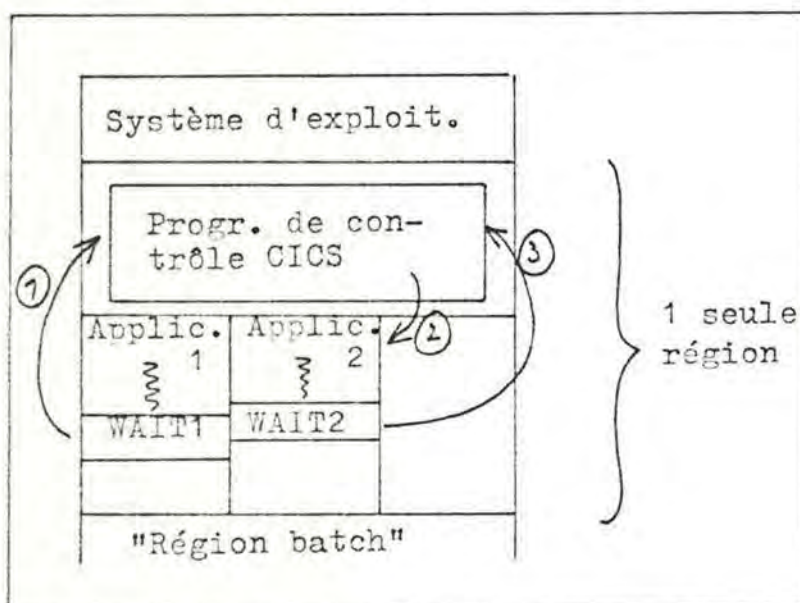


figure A2.1.4

Envisageons la situation schématisée ci-contre en supposant qu'une tâche 1 soit occupée à utiliser un programme d'application 1; lorsque celui-ci passe en état d'attente (consécutif à une demande d'acquisition de message, par exemple), provoqué par une macro WAIT, le contrôle est rendu au système CICS (①) qui, selon les exigences du moment, pourra lancer une autre tâche, utilisant, par exemple, le programme d'application 2 (②). Lorsque, à son tour, celui-ci est "arrêté" provisoirement, le contrôle repasse au système (③), qui vérifie si les conditions ayant provoqué le WAIT1 sont remplies (en l'occurrence, l'achèvement de l'opération d'acquisition de message demandée). Dans l'affirmative, le contrôle est rendu au programme 1; sinon, le lancement d'une troisième tâche peut, éventuellement, commencer.

Cette notion de " quasi - multitasking " diffère de celle de " multitasking " pur et simple où, après le lancement de la tâche 2, la survenance, à un moment quelconque, de l'achèvement

de l'opération ayant occasionné le WAIT 1, rendrait automatiquement la main à la tâche et au programme d'application 1.

B. Quasi - réentrance

L'espace-mémoire disponible pour l'ensemble des programmes d'application TP étant assez limité, le système tolère que plusieurs tâches utilisent le même programme d'application, ce qui permet de ne conserver qu'une seule copie de celui-ci en mémoire.

Reprenons notre exemple précédent (figure A2.1.4) et supposons qu'une tâche 1 soit occupée à utiliser le programme d'application 1; lorsque celui-ci tombe sur le WAIT1, le contrôle est rendu au système, qui initialise une tâche 2, utilisant elle aussi ce programme; ce processus de " partage " implique, pour éviter les interférences entre tâches, une duplication des zones de travail (variables, etc) du programme partagé, chaque copie ainsi obtenue étant rattachée à une tâche précise.

Encore une fois, la notion de " quasi-réentrance " est à distinguer de celle de " réentrance " pure et simple, pour laquelle l'"entrée" de la tâche 2 dans le programme d'application pourrait avoir lieu à tout moment, indépendamment du fait que la tâche 1 vient de " buter " sur un WAIT et de perdre la main.

L'économie réalisée grâce à cette méthode est évidente: il est bien plus avantageux de " dupliquer " des zones de travail de 3 ou 4K bytes au maximum plutôt que de procéder à la copie systématique de programmes d'application qui peuvent parfois occuper 20 ou 30 K.

Annexe 2.2 - Programmation d'un appel d'entrée de message.

Considérons un message composé de trois segments d'information (la longueur d'un segment étant limitée à 130 caractères).

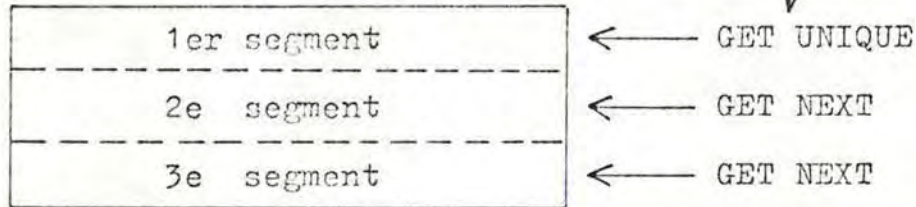
Un appel d'entrée de segment de message se présente comme un accès à une base de données, exception faite du SSA, qui n'aurait aucune utilité ici. Ainsi, dans notre exemple, un GU sera utilisé pour l'acquisition du premier segment et deux GN pour celle des segments suivants.

En Cobol, la séquence d'appel serait la suivante:

```
ENTER LINKAGE .
```

```
CALL ' CBLTDLI ' USING IFUN , LTPCB, IMSG- IO - AREA .
```

```
ENTER COBOL .
```



'CBLTDLI ' est le nom d'une routine d'interface entre le programme d'application et de DL/I d'IMS. Les trois paramètres passés au DL/I sont, dans l'ordre :

- le code de la fonction d'appel;
 - l'adresse du PCB d'entrée/ sortie (TP PCB) du terminal logique dont proviennent les messages;
 - l'adresse d'une zone de réception de chaque segment de message.
-

Annexe 2.3 - Listing de sortie du 3270 FF de BTS

A2.3.1

RTS00071 SIMULATION STARTED. TIME=10:39:10, DATE=75.269.
RTS00021 INPUT RECORD: 00000030
/T TC=PX01 MAP=PXSCD SPA=6144 LANG=PLI PLF=10
RTS00021 INPUT RECORD: 00000040
/D LTERM=MASTER DDDF=327011 FORMBUF=8000
RTS00021 INPUT RECORD: 00000050
/FOP PXM0001

R T S 3270 FORMATTED SCREEN IMAGE. MID/MOD= PXM0001, CUPSD0= LOIC13, ACTION= OUTPUT

```
-----1-----2-----3-----4
*01* TRANSAC *01*
*02* 09+++3C *02*
*03* OPERAT *03*
*04* 3C 309+3C *04*
*05* ARGUM *05*
*06* 33 3C09+++++++3C *06*
*07* IDENTIF *07*
*08* 3C 09+++3C *08*
*09* *09*
*10* *10*
*11* *11*
*12* *12*
-----1-----2-----3-----4
```

THE ATTRIBUTE BITS ARE: PRPNDRM, WHERE P IS RESERVED, ZERO; P MEANS PROTECTED WHEN 1; N MEANS NUMERIC IF 1; M MEANS MODIF. IF 1.
THE MEANING OF THE DD BITS IS: 00=DISPLAY/NO PEN DETECT; 01=DISPLAY/DETECT; 10=BRIGHT/DETECT; 11=DARK/NO PEN DETECT.

RTS00021 INPUT RECORD:
LOIC10 'PX01' LO3C10 'RF' LO5C10 'FM=CLAES' LO7C10 'XY01' ENTER *
RTS00541 TERMINATING ACTION IN CARD NO: 01 ,COL NO: 59

A2.3.9

R T S 3270 FORMATTED SCREEN IMAGE. MID/MOD= PXMDD02, CURSOR= LOIC02, ACTION= OUTPUT

```

-----1-----2-----3-----4
*01* PXSRC003 OR      HRS LIMITE EN R1 *01*
 30-
*02*
 30-
*03*
 30-
*04*
 30-
*05*
 30-
*06*
 30-
*07*
 30-
*08*
 30-
*09*
 30-
*10*
 30-
*11* PRUSSEZ SUR PA1 ET CORRIGEZ SUR *11*
 30-
*12*
 30-
-----1-----2-----3-----4

```

THE ATTRIBUTE BITS ARE: RRRDDDDM, WHERE R IS RESERVED, ZER0; P MEANS PROTECTED WHEN 1; N MEANS NUMERIC IF 1; M MEANS MODIF. IF 1. THE MEANING OF THE DD BITS IS: 00=DISPLAY/NO PEN DETECT; 01=DISPLAY/DETECT; 10=RIGHT/DETECT; 11=DARK/NO PEN DETECT.

RIS00021 INPUT RECORD:
PA1 *

RIS00541 TERMINATING ACTION IN CARD NO: 01 ,COL NO: 01

R T S 3270 FORMATTED SCREEN IMAGE. MID/MOD= PXMDD01, CURSOR= LOIC02, ACTION= OUTPUT

```

-----1-----2-----3-----4
*01* 05 08 09 PC FM-CLAFS *01*
 30-
*02*
 30-
*03* AUCUN 30
 30-
*04*
 30-
*05*
 30-
*06*
 30-
*07*
 30-
*08*
 30-
*09*
 30-
*10*
 30-
*11*
 30-
*12*
 30-
-----1-----2-----3-----4

```

THE ATTRIBUTE BITS ARE: RRRDDDDM, WHERE R IS RESERVED, ZER0; P MEANS PROTECTED WHEN 1; N MEANS NUMERIC IF 1; M MEANS MODIF. IF 1. THE MEANING OF THE DD BITS IS: 00=DISPLAY/NO PEN DETECT; 01=DISPLAY/DETECT; 10=RIGHT/DETECT; 11=DARK/NO PEN DETECT.

RIS00021 INPUT RECORD:
LOIC02 * * LOIC08 * * ENTER 5

RIS00541 TERMINATING ACTION IN CARD NO: 01 ,COL NO: 25

B T S 3270 FORMATTED SCREEN IMAGE. VIEW/MODE=PYM0001, CURSOR= L01C13, ACTION= OUTPUT

```

-----1-----2-----3-----4
#01# TRANSAC PX01 #01#
#02# 30 00***30 #02#
#03# OPERAT #03#
#04# 30 309*30 #04#
#05# ARGUM EW=CLAFS #05#
#06# 30 3000*****30 #06#
#07# IDENTIF XV01 #07#
#08# 30 00***30 #08#
#09# #09#
#10# #10#
#11# #11#
#12# #12#
-----1-----2-----3-----4

```

THE ATTRIBUTE BITS ARE: P=PROTECTED, WHERE P IS RESERVED, ZERO P MEANS PROTECTED WHEN 1; N MEANS NUMERIC IF 1; * MEANS MODIF. IF 1. THE MEANING OF THE DD BITS IS: 00=DISPLAY/NO PEN DETECT; 01=DISPLAY/DETECT; 10=RIGHT/DETECT; 11=DARK/NO PEN DETECT.

BTS0001 END OF INPUT DATA SET ENCOUNTERED.

** SPA CALL=LINE=00 ,PCR=MASTER ,STATUS=00,MESSAGE NUMBER=00000

BTS0021 STATISTICS REPORT FOR TRANSACTION:PX01
 PERNAME CU CU CU CAN DHT DHT DHT DHT ISPT PURG REPL QLET OFD CHKP LOG CHNG
 MASTER 0074 0773 0770 0768 0765 0762 0759 0756 0753 0750 0747 0744 0741 0738 0735 0732 0729 0726 0723 0720 0717 0714 0711 0708 0705 0702 0699 0696 0693 0690 0687 0684 0681 0678 0675 0672 0669 0666 0663 0660 0657 0654 0651 0648 0645 0642 0639 0636 0633 0630 0627 0624 0621 0618 0615 0612 0609 0606 0603 0600 0597 0594 0591 0588 0585 0582 0579 0576 0573 0570 0567 0564 0561 0558 0555 0552 0549 0546 0543 0540 0537 0534 0531 0528 0525 0522 0519 0516 0513 0510 0507 0504 0501 0498 0495 0492 0489 0486 0483 0480 0477 0474 0471 0468 0465 0462 0459 0456 0453 0450 0447 0444 0441 0438 0435 0432 0429 0426 0423 0420 0417 0414 0411 0408 0405 0402 0399 0396 0393 0390 0387 0384 0381 0378 0375 0372 0369 0366 0363 0360 0357 0354 0351 0348 0345 0342 0339 0336 0333 0330 0327 0324 0321 0318 0315 0312 0309 0306 0303 0300 0297 0294 0291 0288 0285 0282 0279 0276 0273 0270 0267 0264 0261 0258 0255 0252 0249 0246 0243 0240 0237 0234 0231 0228 0225 0222 0219 0216 0213 0210 0207 0204 0201 0198 0195 0192 0189 0186 0183 0180 0177 0174 0171 0168 0165 0162 0159 0156 0153 0150 0147 0144 0141 0138 0135 0132 0129 0126 0123 0120 0117 0114 0111 0108 0105 0102 0099 0096 0093 0090 0087 0084 0081 0078 0075 0072 0069 0066 0063 0060 0057 0054 0051 0048 0045 0042 0039 0036 0033 0030 0027 0024 0021 0018 0015 0012 0009 0006 0003 0000

BTS0061 FORMAT CONTROL BLOCK BUFFER SPACE USED: 1010

BTS0001 END OF BTS RUN.

3ème partie

Réalisation d'un générateur de données alphanumériques - DBALPNUM

3.1 Présentation de DBALPNUM

Ce générateur s'inscrit dans le cadre d'un essai d'amélioration de l'outil DBPROTOTYPE, qui présente en effet un inconvénient majeur : celui de ne " produire " que des information essentiellement numériques, sans que l'utilisateur puisse intervenir de quelque manière que ce soit dans ce processus, notamment pour " formater " à sa guise les données qui seront placées dans la future base de test. Ceci ne présenterait guère d'inconvénient si la base générée n'était destinée qu'à être exploitée par un programme comme DBPROC, par exemple, pour lequel la constitution interne des données importe peu et est d'ailleurs pratiquement " transparente " à l'utilisateur. Mais cette lacune est plus durement ressentie dans le cadre de l'exploitation de la base, en période de tests, par des programmes d'application élaborés soumis à des essais : ainsi, un " field " nom de personne purement numérique ne correspond pas spécialement à la réalité. C'est avec le souci de résoudre cette (double) difficulté que nous avons entrepris la réalisation de DBALPNUM. Il s'agit donc d'un programme générant des données alphanumériques aléatoires, à partir de " moules " introduits sur cartes par l'utilisateur.

DBALPNUM est conçu comme un module autonome accomplissant une tâche " parallèle " à celle de DBXGEN (à savoir la constitution d'un fichier séquentiel de sortie, contenant les résultats de la génération), comme l'indique la figure 3.1.

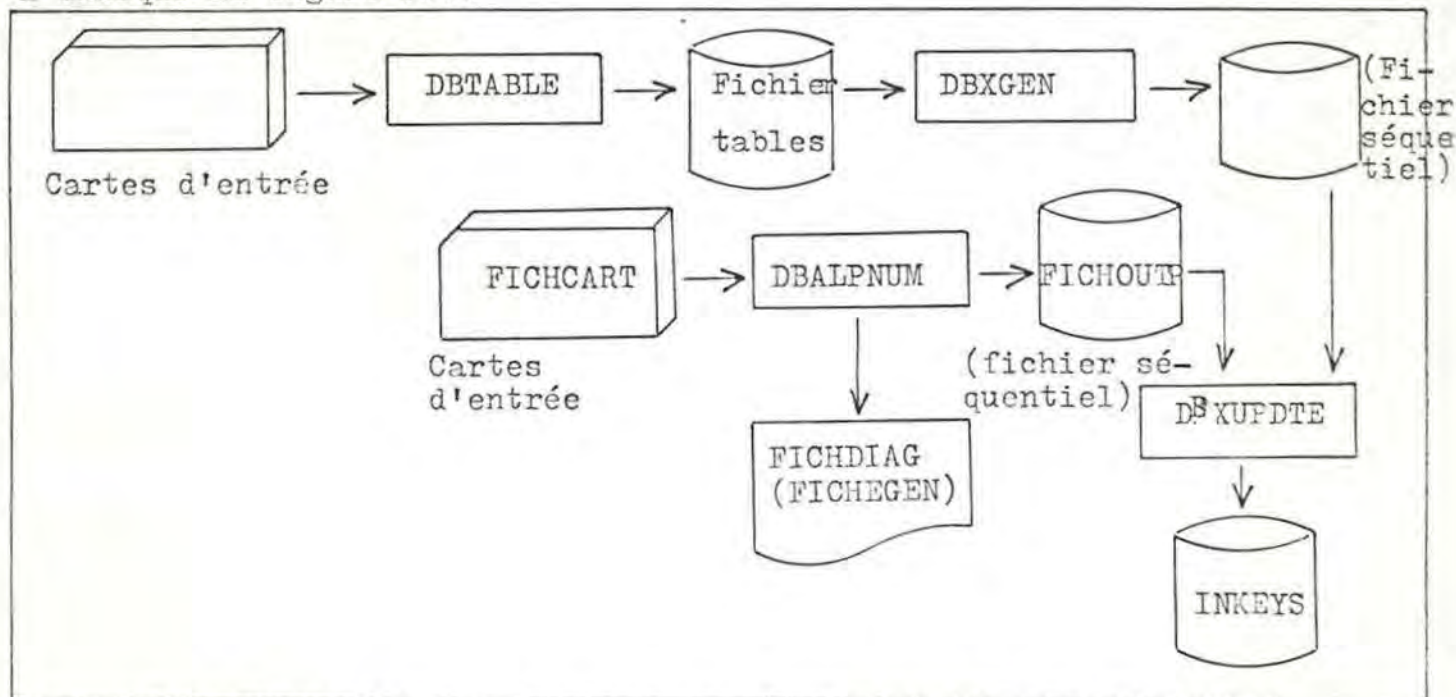


figure 3.1 : DBALPNUM dans le contexte (simplifié) de DBPROTOTYPE

La figure 3.1 souligne l'indépendance de DBALPNUM vis-à-vis de son "entourage"; l'utilisateur en commandera l'exécution s'il désire une génération de données alphanumériques (comme il commande l'exécution de DBXGEN en d'autres temps) et lui soumettra un " input " spécifique constitué de cartes de types P et A (fichier FICH CART), qui sont à DBALPNUM ce que les cartes de classes 6 ou 7 traditionnelles (entrées via DBTABLE) sont à DBXGEN. Les cartes de type P jouent un rôle de " paramètres " pour DBALPNUM, celles de type A ayant par contre une valeur essentiellement " descriptive ": l'utilisateur y indiquera notamment les formats que devront avoir les données alphanumériques qu'il désire voir figurer dans la base.

Les sorties de DBALPNUM seront de deux ordres : d'abord, comme il se doit, un listing de diagnostics (FICHDIAG) et ensuite, un fichier séquentiel (FICHOUTP), de même constitution que celui produit par DBXGEN. Cette analogie de format est indispensable, puisque les deux fichiers serviront d'entrée au module DBXUPDTE.

Mise à part la substitution des cartes de classes 6 ou 7 par celles de type P et A là où une génération alphanumérique est exigée, les autres cartes traditionnelles de DEPROTOTYPE conservent bien entendu leur rôle: ainsi le chargement complet de la base de données (DBLOAD) ne pourra-t-il s'effectuer que dans la mesure où une exécution préalable de DBTABLE (avec les cartes de classes 1 à 3 et éventuellement 6 ou 7, là où une génération alphanumérique n'a pas été demandée) aura été entreprise et le fichier des tables constitué.

En ce qui concerne plus précisément DBALPNUM, nous dirons qu'il réalise principalement deux tâches :

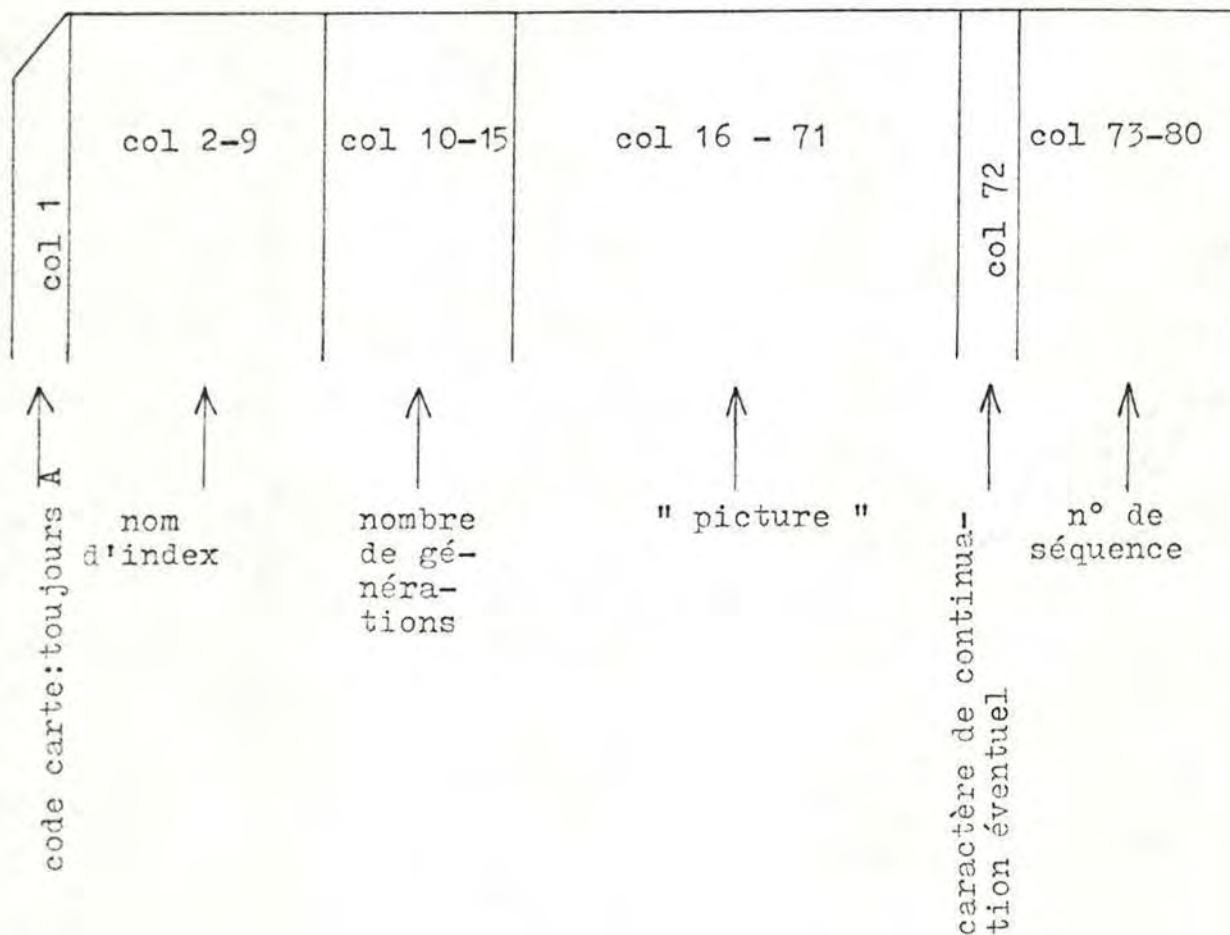
- le décodage des cartes qui lui sont fournies;
- la génération des données.

Avant d'étudier brièvement sa structure interne, nous allons passer en revue les entrées qu'il " accepte " et les sorties qu'il produit.

3.2 Analyse des entrées de DBALPNUM

3.2.1 Les cartes de type A

Elles ont un rôle descriptif :



Passons brièvement en revue les différents éléments y figurant:

- code carte : il est obligatoirement A (nous analyserons plus loin le dessin des autres cartes, de type P celles-là).
- nom d'index : cette zone est destinée à contenir le nom d'un "membre " du fichier INKEYS (produit par DBXUPDTE, rappelons-le), comprenant la suite des données qui auront été générées pour cet index.

Ce nom apparaîtra notamment dans une carte de classe 3 (description de " field ") de l'"input" traditionnel de DBTABLE, correspondant à un " field " de type externe, indiquant par là que les valeurs générées

pour le " field " en question sont à extraire du fichier INKEYS.

Le nom d'index doit être cadré à gauche et complété par des blancs s'il occupe moins de huit positions; il sera constitué uniquement de caractères alphanumériques et doit commencer obligatoirement par un caractère alphabétique.

Exemples:

INDEX1	est valide
INDEX1	est invalide
1INDEX	est invalide
INDEX 1	est invalide
I?DEX	contient un caractère invalide

- nombre de générations: l'utilisateur y indique le nombre exact de générations qu'il désire pour l'index pré-cité. Ce " field " est purement numérique; les blancs n'y sont pas admis.

Exemples:

000789	est valide
789	contient trois caractères invalides.

- picture : ce " field " constitue l'"image" de la donnée à générer. Cette notion est " empruntée " au Cobol; ici, toutefois, l'utilisateur dispose d'une gamme de possibilités moins étoffée:
 - a) s'il désire la génération d'un caractère " alphabétique " (la présence de guillemets, pour étrange qu'elle puisse paraître, se justifiera plus loin), il perforera un A; il placera un N s'il souhaite la génération d'un caractère numérique (soit un chiffre de 0 à 9)

Exemple: AAAAAANNNNN provoquera la génération d'une donnée constituée de cinq caractères " alphabétiques" puis de cinq caractères numériques.

Dans un but, assez évident, d'économie d'écriture (et comme c'est d'ailleurs la cas en Cobol), l'utili-

sateur peut recourir à la présence d'un facteur de répétition placé entre parenthèses; ainsi l'exemple précédent peut-il se ramener à :

A(5)N(5) .

Sans doute l'économie d'écriture ne saute-t-elle pas aux yeux dans ce cas précis, puisque le nombre de caractères à perforer n'a diminué que de deux unités (10 dans le premier cas pour 8 dans le second). Toutefois, lorsqu'on saura l'étendue maximale autorisée pour l'expression d'une " picture ", cette économie ne fera plus de doute.

b) s'il désire l'insertion pure et simple d'une " constante littérale " dans la donnée à générer, l'utilisateur la placera entre quotes dans la " picture " génératrice, en dédoublant les quotes éventuels y figurant.

Reprenons l'exemple précédent et supposons que l'utilisateur désire intercaler le texte RUE DE L' avant les cinq caractères A, et le texte ,NO avant les cinq caractères N; il lui suffira de perforer :

'RUE DE L''A(5)',NO'N(5)
 cste 1 cste 2
 (avec quote 2
 dédoublé) Pour le processus de décodage,

la fin de " picture " coïncide avec le premier blanc rencontré, étant entendu qu'il ne s'agit nullement d'appliquer cette règle aux blancs figurant éventuellement dans une constante littérale, celle-ci constituant un sous-ensemble " clos " de la " picture " globale.

La " picture " ne peut évidemment commencer ni par un blanc ni par un facteur de répétition. A ce sujet, notons aussi que tout facteur de répétition, placé entre parenthèses, ne peut comprendre plus de trois caractères; en outre, sa valeur doit être supérieure ou égale à 2, ce qui ne signifie pas pour autant que

L'utilisateur réalise un quelconque gain d'écriture, au contraire, en écrivant, par exemple, A(2) ou N(3) en lieu et place de AA ou NNN.

La "picture" ne peut entraîner la génération d'une donnée de plus de 256 caractères. Toutefois, le nombre de caractères " physiquement perforés " constituant ce moule, pris dans son intégralité, peut atteindre 280. Etant donné que chaque carte de type A est susceptible d'en contenir au plus 56 (colonnes 16 à 71), cela signifie qu'une " picture " peut s'étendre sur un maximum de cinq cartes (une " tête de série " et quatre de continuation); la présence d'une carte de continuation est signalée par la perforation d'un caractère quelconque (sauf blanc, cela va de soi) en colonne 72. Sur une carte de continuation, les zones habituellement occupées par les " fields" nom d'index et nombre de générations seront obligatoirement maintenues à blanc (cette restriction constitue une précaution utile, principalement contre tout décalage accidentel de la " picture " d'une ou plusieurs positions vers la gauche - à la suite d'une erreur de tabulation, par exemple -sur une carte de continuation, un test en ce sens étant effectué dans le programme, au cours du décodage).

L'exemple ^{suisvant} illustre notre propos du début concernant le nombre de caractères " perforables " et le nombre de ceux donnant lieu, " effectivement ", à génération :

A(260)

et

'RUE DE L''A(240)',NO'N(10)

sont des " pictures " invalides car, bien que constituées respectivement de 6 et 28 caractères seulement, elles entraîneraient la génération de données de plus de 256 caractères.

Le " divorce " apparent existant entre, d'une part, le nombre maximal de caractères " perforables " (280) et, d'autre part, le nombre maximal de caractères donnant lieu à une génération effective (256) s'explique par le fait que, dans la " picture " telle que nous l'avons conçue, il existe certains caractères ne jouant qu'un rôle de délimiteurs, à défaut d'être des générateurs : c'est le cas des parenthèses ouvrantes et fermantes, mais aussi des quotes entourant une constante littérale. De la sorte, limiter à 256 le nombre maximal de caractères effectivement " perforables" aurait imposé d'emblée une contrainte restreignante, illustrée par la remarque suivante :

$$\begin{array}{ccc} \uparrow & \underbrace{\text{'constante littérale de longueur maximale'}} & \uparrow \\ 1 \text{ caract.} & 254 \text{ caract.} & 1 \text{ caract.} \end{array}$$

(le nombre maximal de caractères constituant une constante littérale était ramené à 254)

A fortiori, lorsque l'utilisateur codifiait une picture du style ci-dessous,

$$\underbrace{\text{'A' 'A' 'A' 'etc'}}_{256 \text{ caract.}}$$

l'apparition d'un nombre élevé de quotes délimiteurs diminuait d'autant plus l'étendue de la future donnée générée.

En tolérant une marge de (280 - 256), soit donc 24 caractères délimiteurs par " picture ", nous ne prétendons toutefois pas avoir atteint " la " solution optimale, mais il nous semble qu'il s'agit là d'une marge assez appréciable.

Lorsqu'une " picture " se termine entre les colonnes 16 et 71 d'une carte (qu'il s'agisse d'une "tête de série" ou d'une carte de continuation), les caractères suivants, jusqu'à la colonne 71, doivent être maintenus à blanc.

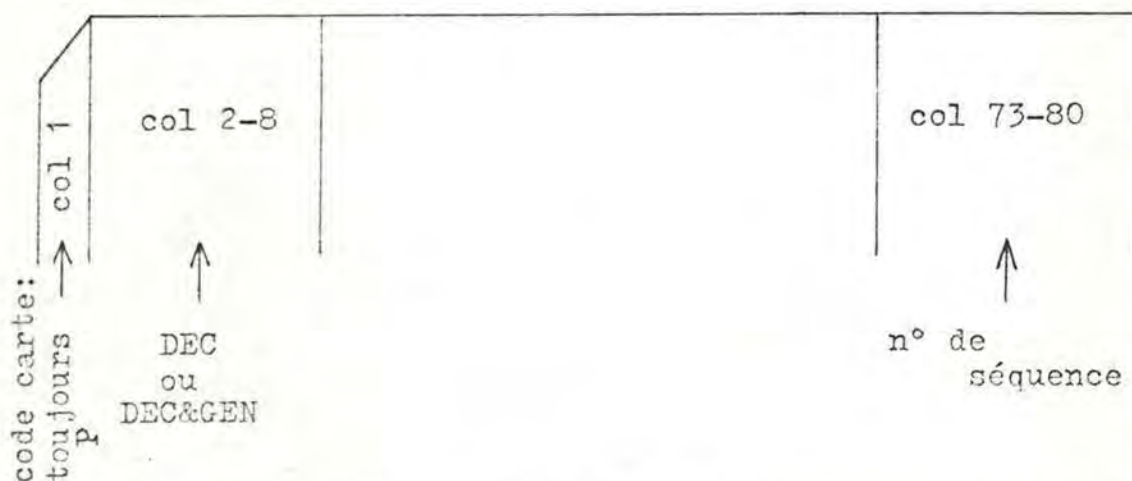
- n° de séquence : facultatif mais recommandé, pour des raisons pratiques, de classement ou d'identification, par exemple. Il ne joue toutefois aucun rôle particulier pour le programme DBALPNUM.

Lorsqu'une erreur est détectée en cours de décodage d'une carte de type A, un message en avertit l'utilisateur et la carte erronée n'entre pas en ligne de compte pour le processus de génération; elle est ignorée, ainsi que la série dont elle fait éventuellement partie.

3.2.2 Les cartes de type P

Elles offrent à l'utilisateur la possibilité de " paramétrer " le comportement du programme DBALPNUM. Elles précèdent le groupe des cartes de type A et sont de deux ordres :

3.2.2.1 Carte obligatoire



L'utilisateur y indique un mot-clé conditionnant le déroulement de DBALPNUM.

S'il est scrupuleux ou simplement prudent, il peut demander un simple décodage (n'entraînant aucune génération) de ses cartes de type A, en perforant le code carte P, suivi du mot-clé DEC ; ceci provoquera l'impression de deux types d' "outputs" à savoir :

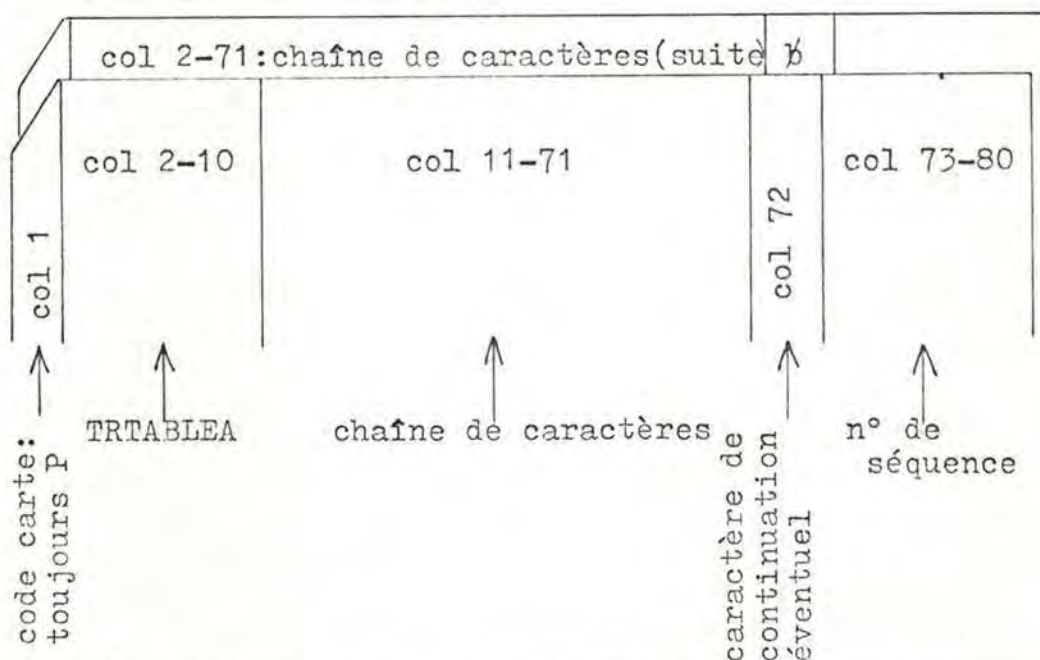
- la liste des diagnostics de décodage du programme DBALPNUM;
- les "éléments des générations " tels qu'ils seraient passés au générateur, dans le cas où le décodage s'accompagnerait de génération.

Si, par contre, il souhaite que le décodage soit accompagné d'une génération, l'utilisateur perforera le code carte P, suivi du mot-clé DEC&GEN; dans ce cas, seuls les diagnostics de décodage lui seront restitués, mais le fichier séquentiel FICHOUTP, contenant les résultats de la génération, aura également été constitué pas à pas au cours de l'exécution.

Une et une seule carte de type P DEC ou DEC&GEN est obligatoire en "input" : elle doit obligatoirement précéder le groupe des cartes de type A, faute de quoi DBALPNUM ne pourra se dérouler.

Le contenu des colonnes 9 à 72 n'entre pas en ligne de compte ici. Enfin, nous n'insisterons plus sur le rôle de pure "figuration" du numéro de séquence.

3.2.2.2 Carte(s) optionnelle (s)



Jusqu'à présent, nous avons laissé entendre que c'était une des 26 lettres de l'alphabet qui était générée lors de la rencontre d'un caractère A dans une "picture".

La chaîne de caractères comprise entre les colonnes 11 et 71 de la première carte et se poursuivant éventuellement en colonnes 2 - 71 de la suivante constitue une "table de translation" spéciale des caractères A. En d'autres termes, c'est parmi cette série de caractères que le processus de génération choisira

aléatoirement, lors de chaque détection d'un A, celui qui figurera à l'endroit correspondant dans la donnée finale.

La table de translation, nous l'avons dit, s'étend sur deux cartes au maximum; elle commence (en colonne 11 de la carte initiale) et se termine obligatoirement par un quote. Tous les caractères compris entre ces deux délimiteurs font partie de la table : blancs et caractères spéciaux peuvent y apparaître sans la moindre restriction. Toutefois, les quotes appelés à figurer dans la table doivent être dédoublés. Le nombre maximal de caractères pouvant constituer celle-ci est donc de :

$$\left[(71 - 12) + 1 \right] + \left[(70 - 2) + 1 \right] = 129,$$

compte tenu que les quotes indicateurs de début et de fin n'en font pas partie.

Sur la seconde carte éventuelle, la colonne 1 est réservée au code carte (P); la colonne 72 est, quant à elle, obligatoirement à blanc.

La zone allant du quote indicateur de fin de chaîne jusqu'à la colonne 71 de la carte où il figure doit être maintenue à blanc (il s'agit, ici encore, d'une précaution contre la présence "abusive", à la suite d'une erreur de frappe de l'utilisateur, par exemple, d'un quote seul au milieu de la chaîne de caractères).

Dès que cette table de translation - utilisateur est décodée et constituée en mémoire, elle reste valable pendant l'entière-té de la génération; elle remplace automatiquement (en l'écrasant purement et simplement) la " table par défaut " comprenant les 26 lettres de l'alphabet.

Cette possibilité offerte à l'utilisateur de paramétrer la table de translation des caractères A ne constitue bien entendu qu'une option : il n'est nullement obligé de s'y conformer et peut estimer qu'une table simplement formée des 26 lettres de l'alphabet suffit à ses besoins. Néanmoins, l'intérêt qu'elle présente est incontestable , dans la mesure où l'utilisateur peut, par exemple, dédoubler, détripler etc...certains caractères auxquels il voudrait attribuer une probabilité de sélection.

tion plus grande.

Dans le même ordre d'idées, remarquons que cette possibilité n'existe pas dans le cadre de la translation des caractères N qui, eux, entraînent automatiquement la génération d'un chiffre choisi aléatoirement (chacun ayant donc la même probabilité de sélection) entre 0 et 9.

Soulignons enfin le peu de sens qu'aurait la présence simultanée d'une carte de type P demandant un décodage seul (PDEC) et d'une autre, donnant une table de translation spéciale des caractères A, celle-ci n'ayant sa raison d'être que lorsque le décodage s'accompagne d'une génération.

Lorsqu'une erreur est détectée en cours de décodage d'une carte de ce type, un message en avertit l'utilisateur et l'exécution du programme DBALPNUM est suspendue.

La ou les carte(s) de type P éventuelle(s) donnant une table de translation spéciale pour les caractères A doit(doivent) obligatoirement figurer avant le groupe des cartes de type A, faute de quoi elle(s) ne sera (seront) pas prise(s) en considération. Dans le groupe des cartes de type P, cependant, elle(s) peut (peuvent) précéder ou suivre la carte DEC ou DEC&GEN. Pour une question de " hiérarchie ", nous recommandons toutefois qu'elle(s) la suive(nt).

3.3 Analyse des sorties de DBALPNUM

Le programme que nous avons mis au point est susceptible de produire trois " outputs " bien distincts :

- dans tous les cas, il restitue à l'utilisateur un listing des diagnostics de décodage, intitulé " DBALPNUM-DIAGNOSTICS DE DECODAGE" et figurant sur le fichier FICHDIAG.

Ce listing se présente de la façon suivante :

1. éventuellement, une suite de messages d'erreurs;
2. un " diagnostic final " indiquant soit le nombre total d'erreurs détectées en cours de décodage, soit que le programme DBALPNUM n'a pu se dérouler normalement, une erreur grave (ou, en tout cas, irréparable) en ayant occasionné l'arrêt.

Les messages d'erreurs sont en général constitués de deux parties:

1. d'abord une courte phrase indiquant la nature de l'erreur détectée;
Exemple: Code carte invalide

Caractère invalide/Field picture

2. ensuite, en regard de chaque phrase, la carte sur laquelle l'erreur a été détectée; dans la plupart des cas, c-à-d lorsqu'un caractère incorrect a été trouvé ou qu'un " field " a été déclaré invalide, une visualisation ^{de l'emplacement} du caractère " litigieux " est ajoutée, sous la forme d'un curseur (une petite ligne verticale), imprimé sous ce caractère Ceci doit permettre à l'utilisateur une localisation aisée de son (ses) erreur(s).

Il est toutefois des cas où une telle restitution ne revêt pas de signification : ainsi, par exemple, lorsque l'utilisateur a omis la carte de type P DEC ou DEC&GEN dans le paquet d'entrée, le message d'erreur ne comportera, bien évidemment, aucune impression de carte erronée. On y indiquera simplement :

Carte de type P (DEC ou DEC&GEN) manquante en tête d'input

Suspension du programme

- dans le cas où, à la demande de l'utilisateur, un décodage seul a été exécuté, un second listing est imprimé à la suite du précédent; il est intitulé " DBALPNUM-ELEMENTS DES GENERATIONS", figure sur le fichier FICHEGEN, et restitue à l'utilisateur l'état de zones "critiques" telle qu'elles seraient entrées en jeu, à des titres divers, au cours de la

génération, si celle-ci avait été entreprise.

Ce listing est " formaté " de la façon suivante :

nom d'index	∅	nbre de générations	∅	version définitive de la picture,
(8 caract.)		(6 caract.)		découpée en groupes de 116 caract.

Par " version définitive de la picture ", il faut entendre l'éventuelle mise bout à bout des différents fragments la constituant, tirés des colonnes 16 à 71 de la série de cartes de type A se rapportant à l'index correspondant, et où les facteurs de répétition entre parenthèses ont disparu, faisant place à l' "extension" des caractères concernés.

Exemple :

Considérons la carte de type A mettant en jeu les éléments suivants:

INDEX1 000500 A(5)N(5)

Elle donnerait lieu au " formatage " de sortie suivant :

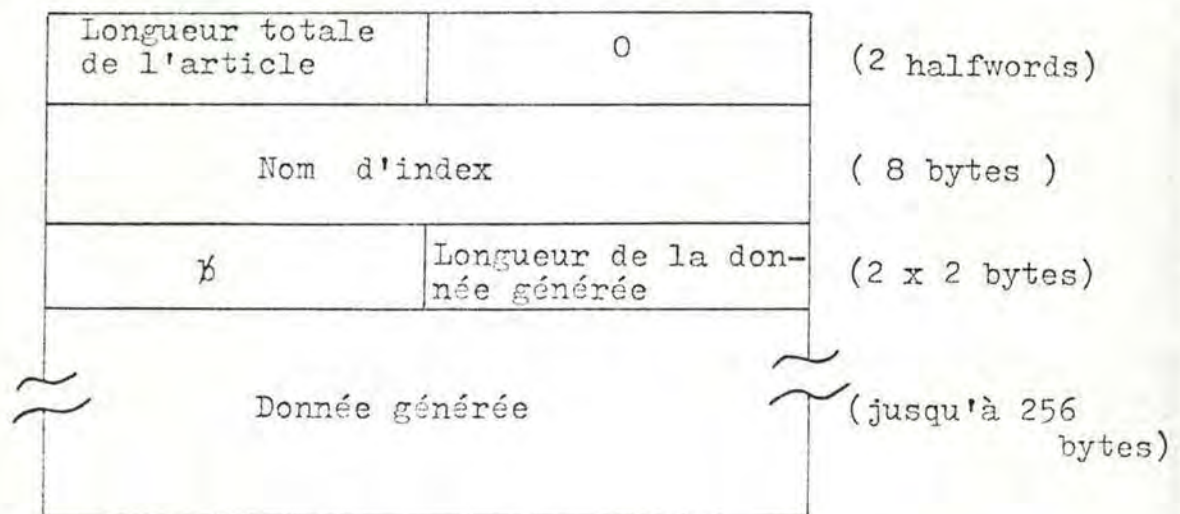
INDEX1 ∅000500∅AAAAANNNNN

Afin de faciliter à l'utilisateur la lecture de ce listing, un en-tête de numérotation des colonnes est imprimé au-dessus de chaque page (en plus des titres de chacun des trois éléments); il est constitué de quatre lignes : celle de la centaine, celle des dizaines, celle des unités et une ligne blanche de séparation.

En analysant le listing, l'utilisateur est à même de se rendre compte du profil définitif de chacune des " pictures " sur lesquelles devait fonctionner le processus de génération; il peut, le cas échéant, " rectifier le tir " en modifiant certaines " pictures " dont il ne serait pas satisfait. Après qu'il ait corrigé les erreurs éventuellement signalées dans les diagnostics de décodage, il peut, par acquit de conscience, " re demander " un décodage seul ou estimer, au contraire, que le moment est venu de demander un décodage accompagné de génération.

Les six premières pages de l'annexe 3.1 contiennent quelques échantillons de chacun de ces deux types de sorties de DBALPNUM.

- dans le cas où le décodage s'est accompagné d'une génération plus ou moins complète, nous savons déjà que le listing " éléments des générations " n'est pas produit; mais les données résultant de la génération sont stockées sur un fichier séquentiel, FICHOUTP, dont l'article logique (de longueur variable, puisque la donnée générée occupe de 1 à 256 caractères) se présente comme suit :



Cet article a exactement le même format que celui du fichier "output" de DBXGEN. Les deux premiers " halfwords " constituent le mot descripteur de l'article, traditionnel lorsque, comme c'est le cas ici, on travaille avec des articles de longueur variable.

On remarquera aussi que les deux bytes précédant la longueur de la donnée générée sont laissés à blanc; ces deux bytes sont normalement occupés par un nombre aléatoire généré au cours du déroulement de DBXGEN et servant de clé pour un tri ayant lieu ultérieurement, visant à arranger de façon totalement " fortuite " la suite des valeurs produites par DBXGEN, avant de les présenter au chargement. Dans le cadre de DBALPNUM, ceci ne présentait pas d'intérêt, vu que les données alphanumériques générées sont, par définition, elles-mêmes produites de façon aléatoire.

A côté de cela, dans le cas qui nous occupe, c-à-d lorsque les données générées sont de type alphanumérique, l'utilisateur peut parfois souhaiter un tri de celles-ci avant leur chargement. Ceci peut être réalisé simplement par l'adjonction d'une étape ("step") en " Job Control Language "(JCL), demandant l'exécution de l'utilitaire de tri sur le fichier FICHOUTP, juste après l'étape commandant l'exécution du programme DBALPNUM.

3.4 Structure du programme DBALPNUM

Nous ne nous étendrons pas trop longuement sur l'analyse de la structure interne du programme DBALPNUM. Disons d'abord qu'il a été rédigé en Assembleur et qu'en gros, il est constitué des étapes suivantes:

1. Phase d'initialisation

Ouverture des fichiers FICH CART et FICH DIAG. Impression du titre de ce dernier.

2. Phase de décodage

a. lecture et décodage des cartes de type P.

Ceci comprend :

- . d'une part, la mémorisation (un indicateur d'un bit, appelons-le SW, est utilisé à cet effet) de la demande de l'utilisateur :
décodage seul ($SW \leftarrow 0$) ou décodage accompagné d'une génération ($SW \leftarrow 1$);
- . d'autre part, l'éventuelle constitution, en mémoire, de la table de translation spéciale des caractères A indiquée par l'utilisateur.

b. Si, comme nous le supposerons, tout s'est déroulé correctement à l'étape 2a, lecture et décodage des cartes de type A.

Pour chaque série donnée de cartes (une " tête de série " et les cartes de continuation éventuelle), ceci comprend :

- . le contrôle de validité puis le sauvetage des " renseignements " figurant sur la carte " tête de série " (à savoir nom d'index et nombre de générations);
- . la constitution, en zone de travail, d'une " picture globale ", obtenue en mettant bout à bout les morceaux (colonnes 16 à 71) figurant sur les cartes de la série, puis le contrôle de validité de la zone de travail ainsi constituée; au fur et à mesure, construction de la version définitive de la picture (notion évoquée précédemment) dans une zone tampon.

3. Phase d'exploitation

Lorsque, comme nous le supposerons, le contrôle de validité de la zone de travail s'est achevé avec succès, il reste à exploiter le contenu de la zone tampon; à cet effet, un test préalable de SW est effectué :

- a- s'il est à 0, l'utilisateur a demandé un décodage seul et la zone tampon est dès lors destinée à l'impression; avec les " fields " nom d'index et nombre de générations, elle constituera le contenu du fichier FICHEGEN, qu'il reste à " formatter " convenablement. Aller en 2b pour le traitement d'une autre série de cartes.
- b- S'il est à 1, l'utilisateur a demandé un décodage accompagné d'une génération et la zone tampon est destinée à être balayée caractère par caractère par le processus de génération, autant de fois que le précise le " field " nombre de générations:
- . la détection d'un A provoque la génération d'un caractère " alpha-bétique";
 - . " " " N " " " " " " " numérique;
 - . " " d'une constante littérale alimente en conséquence l'article du fichier de sortie FICHOUTP, qu'il reste à " formatter " conformément au dernier schéma du paragraphe 3.3.
- Aller en 2b pour le traitement d'une autre série de cartes.

En y regardant bien, on peut isoler nettement trois types de tâches dans la séquence d'étapes énumérées ci-dessus, chacune de ces tâches alimentant d'ailleurs un fichier spécifique.

En premier lieu, l'étape 2 recouvre tout ce qui est décodage: à partir du fichier FICH CART (cartes de types P et A) en entrée, elle constitue un fichier de diagnostics FICHDIAG. Ensuite, selon le contenu de SW, deux orientations sont possibles. D'abord, l'étape 3a représente en quelque sorte un prolongement du décodage, alimentant le fichier FICHEGEN. Enfin, l'étape 3b réalise la génération proprement dite, dont les productions constituent FICHOUTP.

Ces quelques considérations suggèrent une découpe modulaire du programme DBALPNUM, comme l'indique la figure 3.2.

Il va de soi que le module CSECT1 est de loin le plus volumineux; à vrai dire, cependant, une grosse partie du " coding " y est consacrée aux traitements d'erreurs, ce qui est logique si on songe qu'il réalise un travail de décodage, comprenant inévitablement une partie importante réservée à la détection de celles-ci, d'une part, aux

procédures d'exception leur correspondant, d'autre part.

Les routines IMPENTET et GENER auxquelles il est fait allusion dans le schéma jouent un rôle relativement important dans le module auquel

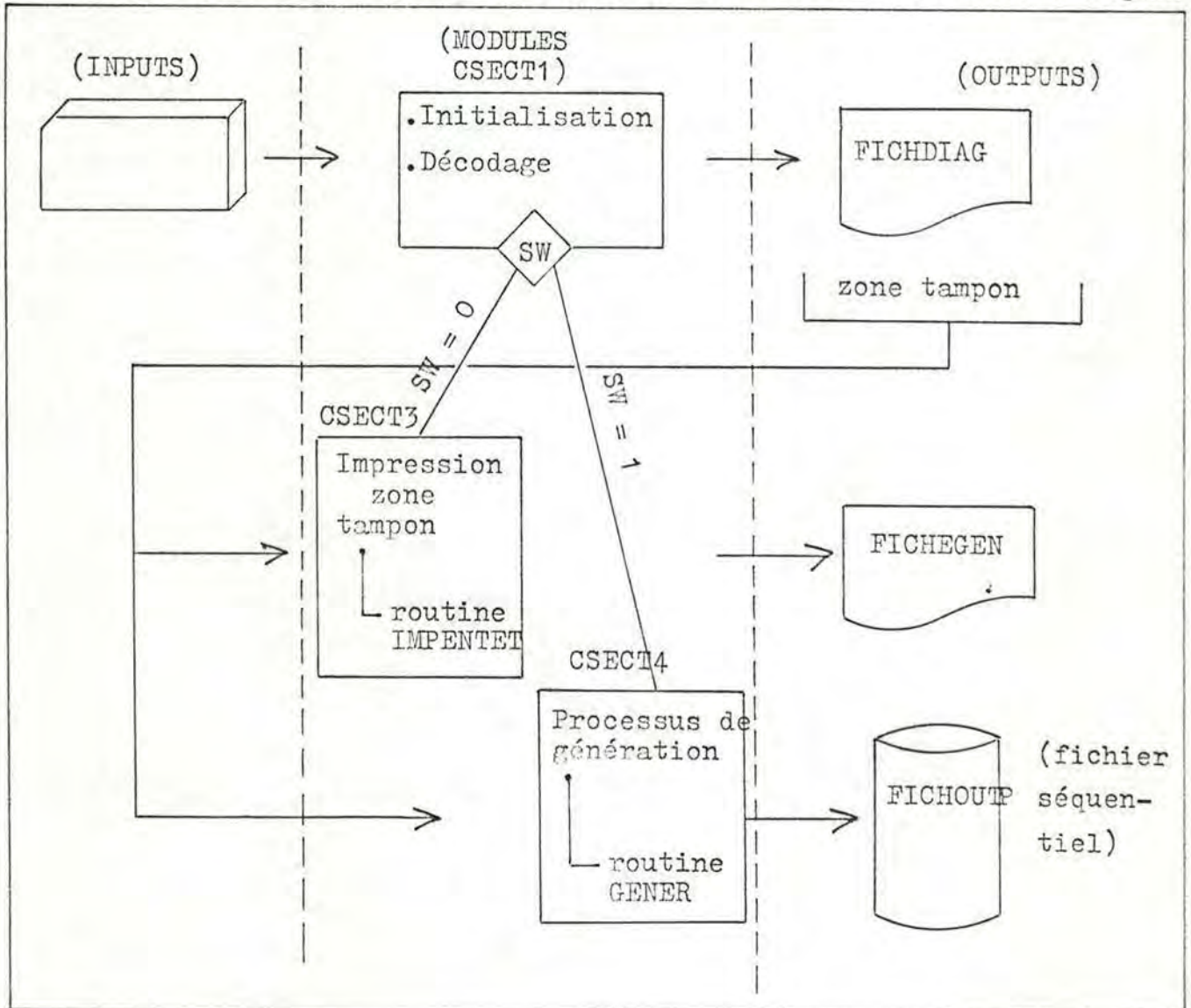


figure 3.2

chacune d'elles est attachée: ainsi, la routine IMPENTET (module CSECT3) effectue l'impression de l'en-tête de page du fichier FICHEGEN (numérotation des colonnes/titres de page); quant à la routine GENER (module CSECT4), elle réalise la génération de nombres aléatoires qui permettront la " confection " des données alphanumériques. Nous allons d'ailleurs détailler quelque peu les principes de fonctionnement du processus de génération.

Enfin, il ne faut pas s'étonner de ne trouver aucune trace, sur la figure 3.2, d'un module CSECT2 : il en existe bien un, mais nous

n'avons pas jugé utile d'en parler à ce stade; nous le ferons au paragraphe 3.6

L'annexe 3.2 contient les listings de chacun des modules constitutifs de DBALPNUM.

3.5 Principes de fonctionnement de la génération

Le processus de génération joue bien entendu un rôle primordial dans le programme DBALPNUM, puisque c'est lui qui produit un par un les caractères qui constitueront les données appelées à figurer dans la future base de test. Ceci est réalisé en deux étapes :

1. la génération de nombres aléatoires selon un algorithme bien précis (routine GENER);
2. la transformation de ceux-ci en caractères numériques ou " alphabétiques ".

L'algorithme de génération utilisé a été " emprunté " au produit " Test Data Generator ", dont nous avons cité le nom dans la deuxième partie (paragraphe 2.1.1), car il correspondait assez bien à nos besoins; cet algorithme modifie les nombres produits par un générateur de type " congruentiel mixte " au moyen d'opérations effectuées à partir de deux compteurs. Le générateur congruentiel mixte seul, en effet, ne produit pas des résultats satisfaisants du point de vue statistique lorsqu'il est programmé sur des machines ayant un " petit " mot-mémoire, comme l'IBM 360 ou 370.

La formule de récurrence de ce générateur est la suivante :

$$X_{n+1} = A X_n \pmod{2^b} + c$$

$$\text{avec } A = 8t + 3 \quad \forall t ;$$

b = capacité du mot-machine en nombre de bits=32;

c = constante quelconque choisie arbitrairement égale à 162.

Dans le cas qui nous occupe, t a été fixé à 1024, ce qui donne

$$A = 8(1024) + 3 = 8195$$

En réalité toutefois, et comme nous l'avons indiqué plus haut, ce type de générateur ne donne pas de bons résultats s'il est appliqué à une machine dont le mot comprend moins de 35 bits; en outre, le nombre ini-

tial (X_0) doit, obligatoirement, être impair. Afin de tourner cette double difficulté, Herbert C. Ratz et J.V.Hildebrand ont (dans un article paru en décembre 1969 sous la rubrique " IEEE Transactions on Electronic Computers") proposé de " corriger " ce générateur à l'aide d'opérations effectuées sur deux " compteurs ". Ceci permet l'utilisation du générateur sur une machine dont le mot est de capacité inférieure à 35 bits (32, notamment) avec de meilleurs résultats qu'auparavant, et cela quelle que soit la nature du nombre initial.

Le processus est très simple : les deux compteurs, appelons-les CTR1 et CTR2, occupent chacun un fullword et sont initialisés, en début d'exécution, aux valeurs hexadécimales suivantes : 12AB9793 et EA34784C. A chaque passage dans la routine de génération, CTR1 est incrémenté de une unité (il a donc une période de 2^{32}), tandis que CTR2 est décrémenté de la même grandeur, sauf lorsque CTR1 = 0 (il a donc une période de $2^{32} + 1$). Considérés " globalement ", CTR1 et CTR2 cyclent donc sur leurs valeurs initiales avec une période de 2^{64} .

Lorsque les compteurs ont été mis à jour, un nouveau nombre X_{n+1} est calculé en appliquant la formule de récurrence; X_{n+1} est alors transformé de la façon suivante :

1. opération " exclusive or " entre les deux compteurs;
2. " " " " " X_{n+1} et le résultat du 1.

Enfin, le bit d'ordre le plus élevé du résultat du 2 est positionné à 0, ce qui assure que le nouveau nombre est bien positif. Celui -ci est alors converti en décimal, les sept " digits " de droite constituant le nombre aléatoire initial (X_n) pour la génération suivante (c-à-d celle devant fournir le nouveau X_{n+1}).

La séquence des nombres générés ne cyclera que si le nombre de sortie X_{n+1} est produit avec la même combinaison de valeurs des compteurs CTR1 et CTR2. Toutefois, ceux-ci ayant " globalement " une période de 2^{64} , ceci n'a une chance de se produire que toutes les 2^{64} fois, ce qui signifie que la période minimale de la séquence des nombres générés est 2^{64} , c-à-d approximativement 10^{19} .

Ceci étant dit, il reste à transformer les nombres aléatoires de sept chiffres dont nous disposons à présent, suyant le cas, en " lettres " ou en chiffres. Pour fixer les idées, nous allons décrire le passage

du nombre en question à l'une des 26 lettres de l'alphabet.

Commençons par accorder une valeur numérique à chaque lettre :

A=0 ; B=1 ; C=2 ; ... ; Z=25.

Le problème revient par conséquent à ramener le nombre généré entre les limites de l'intervalle $[0,25]$; pour cela, il suffit de considérer ce nombre comme une fraction comprise entre 0 et 1 et de multiplier celle-ci par la limite numérique supérieure. Par exemple, si le nombre généré est 6327980, on se ramène à l'échelle voulue par l'opération :

$$.6327980 * 25 \ .$$

En pratique toutefois, on ne considérera que les trois chiffres d'ordres les plus élevés du nombre primitif pour effectuer la multiplication : ainsi, pour en revenir à l'exemple précédent ,

$$\ . \ 632 \ * \ 25 \ ,$$

soit donc 15.8, constitue le résultat final de la génération.

Arrondi à l'unité (ce qui donne 16 dans le cas qui nous occupe), ce dernier permet un adressage aisé de la table de translation, considérée (celle des caractères A ou N suivant le cas). Puisque nous désirions générer une lettre de l'alphabet, rappelons-le, cette table était donc constituée des 26 caractères de A à Z, chacun d'entre eux occupant un byte. Ajoutant 16 à l'adresse de début de cette table, nous atteignons de la sorte l'emplacement de la lettre Q. Il reste à transférer celle-ci à la place qui lui revient dans la zone d'entrée/sortie du fichier FICHOUTP.

Nous n'avons plus qu'à élucider la question posée par l'expression " suivant le cas " soulignée auparavant; en d'autres termes, nous venons de décrire arbitrairement le passage d'un nombre aléatoire de sept chiffres à une lettre de l'alphabet, mais comment le processus de génération " distinguera"-t-il les cas où il doit produire un caractère " alphabétique " de ceux où il doit produire un caractère numérique? C'est la fameuse limite numérique supérieure qui doit solutionner cette difficulté, puisque nous avons vu qu'il suffit de la multiplier à chaque fois par les trois chiffres d'ordres les plus élevés du nombre qui vient d'être généré pour se ramener à l'échelle voulue. Il faudra donc " passer " cette limite comme paramètre au générateur lors du branchement vers celui-ci.

En Assembler, ceci est réalisé de la façon suivante:

- dans le cas de la génération d'un caractère " alphabétique ":

```
BAL R09,GENER
LIMSUPA DS PL2
```

Cette séquence d'instructions provoque le branchement vers la routine GENER, tout en rendant disponible pour celle-ci le contenu du symbole LIMSUPA, dont le registre R09 contient l'adresse. LIMSUPA aura été garni précédemment avec la valeur de la limite numérique supérieure pour caractères A. Cette valeur égale toujours le nombre de caractères constituant la table de translation moins un. Par exemple, lorsque l'utilisateur n'a pas spécifié de table de translation spéciale, la table est constituée, par défaut, des 26 lettres de l'alphabet, et la valeur figurant dans LIMSUPA sera 25.

- dans le cas de la génération d'un caractère numérique :

```
BAL R09,GENER
DC PL2'9'
```

Cette fois, la limite supérieure est de type " constante " et vaut 9, puisqu'on a vu que la possibilité de paramétrer la table de translation des caractères N n'existait pas; celle-ci contient systématiquement dix caractères (les chiffres de 0 à 9).

Les pages A3.1.7 à A3.1.9 de l'annexe 3.1 présentent les résultats de la génération de données conformément au "schéma" de la p.

3.6 Le passage de paramètres au programme exécutable DBALPNUM

L'exposé des possibilités offertes à l'utilisateur de " paramétrer " certains " éléments " du programme DBALPNUM serait incomplet si nous n'en signalions pas une dernière, assez astucieuse.

Lorsqu'il commande l'exécution du programme DBALPNUM, stocké en bibliothèque sous forme d'un module " link-edited " et donc exécutable, l'utilisateur peut, sur la carte JCL en question, faire figurer entre quotes et derrière le mot-clé " PARM=", des paramètres qu'il destine à ce programme. Ceux-ci sont de deux ordres:

- il y a d'abord le nombre aléatoire initial pour la formule de récurrence du générateur (Xo de départ). Ce nombre contient obligatoirement six " digits ". La valeur par défaut de ce paramètre est 184095.
- il y a ensuite le nombre de lignes par " page imprimée " du fichier FICHEGEN.

Rappelons, que, pour la parfaite "lisibilité" du listing, nous avons prévu l'impression d'un en-tête au-dessus de chaque page. Le paramètre " nombre de lignes par page " est intéressant parce qu'il conditionne cette impression, en ce sens qu'il en donne, en quelque sorte, la fréquence. Pour être précis, il intervient comme une borne supérieure à ne pas dépasser pour le compteur de lignes imprimées du fichier FICHEGEN. Lorsque cette limite est atteinte, un saut de page est effectué, l'en-tête de page imprimé et le compteur de lignes "réinitialisé ".

Ce " paramétrage " du nombre de lignes par page ne présente pas d'intérêt particulier dans le cadre de l'impression des diagnostics (fichier FICHDIAG), puisque nous n'y avons pas prévu l'impression d'un en-tête de page.

Le paramètre nombre de lignes par page occupe de une à trois positions numériques; sa valeur est obligatoirement supérieure ou égale à 20. Quant à sa valeur par défaut, elle est de 55.

Remarquons que ces deux paramètres ont des " vocations " diamétralement opposées; le premier, puisqu'il conditionne le générateur, est orienté " décodage & génération "; quant au second, vu qu'il concerne le fichier FICHEGEN ("output " de décodage), il est orienté " décodage seul". Ceci

n'empêche pas qu'ils puissent figurer tous deux sur la carte JCL d'exécution de DBALPNUM, ce qui simplifie la tâche de l'utilisateur, qui n'a donc pas à se servir de deux cartes EXEC différentes, suivant qu'il choisisse l'option " décodage seul " ou " décodage & génération " (carte de type P appropriée).

Lorsque les deux paramètres figurent simultanément sur la carte JCL EXEC, ils sont obligatoirement séparés par une virgule; par ailleurs, leur ordre de rangement importe peu.

Exemples:

```
//GO EXEC PGM=DBALPNUM,PARM='50,612785' est valide;
      "      "      "      '612785,60' est valide;
      "      "      "      '326?77' contient un caractère invalide;
      "      "      "      '326587,15' contient une valeur invalide
                                  pour le paramètre nombre de
                                  lignes par page.
```

Pour être complet, signalons que la partie programmation concernant le décodage de ces paramètres passés à DBALPNUM via la carte JCL EXEC est englobée dans le module CSECT2, s'insérant juste après la phase d'initialisation du plan sommaire ébauché au paragraphe 3.4.

Lorsqu'une erreur apparaît lors de ce décodage, un message en avertit l'utilisateur et l'exécution du programme DBALPNUM est suspendue.

Les pages ^{A3.1.10 et}
_{A3.1.11} de l'annexe 3.1 donnent des exemples de diagnostics de décodage de ces paramètres ayant fait apparaître une erreur.

Annexe 3.1 - Sorties de DBALPNUM

Les listings que nous présentons ici constituent quelques échantillons de sorties de DBALPNUM.

DSALPNUM-DIAGNOSTICS DE DECODAGE

CARACTRE AUTRE QU'ESPACE DETECTE APRES QUOTE INDICATEUR DE FIN DANS CHAINE DE CARACTERES TRTABLEA:

PRTABLEA='ABCDEFGHIJKLMNPOQRSTUVWXYZ'

AAAAARRCCDDDD

*0000001

1

SUSPENSION DU PROGRAMME

A3.1.1

DBALPNUM-DIAGNOSTICS DE DECODAGE

PAS DE QUOTE INDICATEUR DE FIN DANS CHAINE DE CARACTERES DONNANT TRTABLEA:
PSSSTUVA AAAARCCDDDEEEF GHIIJKLLLLMMMMNNNOO PPPORRRSSSTTTUUVY ' ' ' ' ' ' 0000002

SUSPENSION DU PROGRAMME

A3.1.2

A3.1.3

FIELD NDM D'INDEX INVALIDE:

CARACTERE INVALIDE/FIELD NR DE GENERATIONS:

PAS DE QUOTE INDICATEUR FIN DE COTE LITTERALE DANS

PARENTHESE FERMANTE MANQUE DANS PICTURE:
NOMBRE D'ERREURS DETECTEES: 4

DRALPMM-DIAGNOSTICS DE DECODAGE

AINDEX1 1000050A(69) | 1A(69) | 1A(30) | 1ANN |

AINDEX1 000050A(69) | 1A(69) | 1A(30) | 1ANN |

AINDEX1 000050A(69) | 1A(69) | 1A(30) | 1ANN |

AINDEX1 000050A(69) | 1A(69) | 1A(30) | 1ANN |

1N(5)A(24)

00000010

1N(5)A(24)

00000010

1N(5)A(24)

00000010

1N(5)A(24)NNNNNNNNNN(1) 00000010

FIELD PICTURE INVALIDE-ENTRAINANT GENERATION D'UNE DONNEE DE PLUS DE 256 CARACTERES:

CARACTERE INVALIDE/FIELD PICTURE:

CARACTERE INVALIDE/FIELD PICTURE:

FIELD PICTURE INVALIDE:

NOMBRE D'ERREURS DETECTEES: 4

A	18)	00000011
AINDEX2	000800AAAAAAAAAAN(20A(20)	00000020
AINDEX3	012345AAAAAAAAAANNNNNNNNNNNAAAAAAAAAANNNNNN?NNN	00000030
AINDEX4	003210NNNNNNNNNNNAAAAAN(20)(20)	00000040

Carte de continuation d'une tête de série contenant des renseignements identiques à ceux figurant sur la dernière carte erronée apparaissant dans les diagnostics de la p. A3.1.3

A3.1.4

NOMBRE D'ERREURS DETECTEES: 0

A3.1.5

A3.1.7

INDEX1	SMLEFT	CORTAEGR	RYBJEJYZ	OXTECVRH	IMXKIRJW	ILROTSRI	UJUNUEPB	WFSILHZY	QLRACAWG	B YMYMD	MILUQPV
KNFQYLR	NQUQNPUC	CUYNNMOT	GVEQMLQ	ARZDQPY	XZFRPRXU	XVVKYER	MSSFDCD	SFMKCCQW	KFONRVXX	CJWBPE 6	16 4
5674NDNA	LLBOKIAW	STEDJNLL	NQXV*								
INDEX1	EJKMW	UCGCTQUE	TALICBKL	XGCMVYJX	DI IACXSV	ZJJEMIUC	ZWNGOQOB	ERXPFWQV	DPZPYMLB	Y MBVMYK	FNDCKQKS
XLVRFNY	SLVICJNW	WVDEIEHE	LRDMFNMJ	TGFWRQEE	TGYFFKLW	OYHCYIT	ONGIMDTI	PCUWPCXG	XDCCJPUH	MIKQSQ 8	73 6
3633CBKF	NKGRPAZC	NRLKOTCI	NROW*								
INDEX1	ENCRR	RPMFDIQS	WTXGRHCE	ZYNRHKCB	QWRHKPIE	SFPNR IXG	RKJTRPHZ	PSYPWFYN	KGLXZIGO	W CXEGTK	ONEQILZT
YMC IWFRS	CQIRBHTK	CPLVXLEG	VPCZWMJR	NCEMHQKM	CUNJWFCI	QVL PRR	SJXUPEGR	GWXJRVKH	XRPTVKME	VHAGWQ 8	31 7
5546BRGW	FXGKWVUV	ULECYGSH	FLEY*								
INDEX1	EFGGH	HYJKQVJP	KPMMEDQQ	PYXVLP IW	JFQBMQDI	EYKSZKNF	IXROJLTH	GTXMVRQW	SNNZVFI	J YJIIDZ	PBUHQJFW
GSFRORNY	UONNYTRT	LQYASXP	LIBUKMX	ESAO LPFQ	RHOZMIJF	VXEF IZP	JFEURWKS	RCZRTKEP	FNPKRHJI	QHHFLV 4	77 7
R164HL VM	VQJL VMHN	SMGJDRGR	FZBB*								
INDEX1	EDVYH	USROXWRH	FRHVGAMH	MKPHYVPE	YSYLLAMU	MMKIXLEA	HYVOKJNP	UMKKCEXF	WIDNNOWP	H WQDQDV	RXRFXFD
YMLIAIWTJ	CMKNBPJR	NYWDVGLM	HUPEEWYL	KJFKSDJM	WVVPHPG	HYYSITL	INIONSHS	NDEQFIYG	AOSMULYJ	NZVJXE 6	52 2
3637VLLU	WBXIWNCO	CCMJVXRB	PKHE*								
INDEX1	EGTFU	QTHVSBWK	HIFFNUGV	JVLGQIHN	CGXMSEGS	YDPIPXYY	XBHGGJYH	QUEJKHHN	RKCEWDSF	E FLEGUN	HASTNKS
XDMQJWPK	NTCHNXMR	GVWYYZTH	IRTOVSLH	JFTUGXNY	NDKEWRCC	QNFBTIC	ZUGSJXKN	SADDCCUN	HFJTFQDF	RBKAOM 1	30 7
8084GWXR	ZBXP TSNF	NJ IQLTKY	QUHN*								
INDEX1	EJFZN	RFULFIUZ	VHRSUXGD	TTVFARCF	MKGUWEPD	LGATMRON	YRNWJWAI	POOWTRQL	SPKNMQQD	L IKVJXN	JNBHPRMG
RHWEXXCZ	RGRVGLYM	KWEQAACY	JTFFIJGK	JJAJJAYP	IRIEBLOU	TPFVMD	JYJITQCD	LOODLFEK	RMNDBXAV	UMAYKY 6	07 2
7207JLKN	RICUTQYD	KDRGNYYE	KEGG*								
INDEX1	EPXGV	QJCFDBNS	QIIFSXZP	JKFSQNMN	PKQVTTQY	YPOVRRZO	YPDWKGJO	HBXQUBNL	VQZFBXNW	D CUSNXG	KRVWSJDM
PCUSFFLG	GHHFTVHM	LGVGVCWR	VMHXHPCD	GVVCMCFW	AVWTLSRN	YKHTHZD	KRMCOSTE	PLRYKKDS	ZWHWSPIK	TDWDLU 7	57 5
2424NSYE	SUBXCYCW	WKEHVBRM	RYCU*								
INDEX1	ENLMM	TIOQCBTO	CKFPDKUL	UFSEVDD	IJYVBGK	NIVYIPOS	BPAXDWQV	XCZIHUW	EYROVBXV	M BRIASL	ZLNKOSOP
QRVJUBO	YQQUGVEK	WQIHWZUD	QGMQYSSP	VWRJTDCC	AQCVELWW	LIWEFVJ	IXJBSRBK	WLXFTBBZ	NXUZTVLL	ZZWEWN 2	48 7
7494JYZE	BNQDMCUW	JJQFAHPP	SJIC*								
INDEX1	EXWMR	SZFNQNSMA	WHNRKQX	BRJQNMNS	AQZYWOLL	RVXOHTDT	QIVMUIRD	ZOQRNHFH	RLPLSMRX	G ISPOSB	WCMPTPO
QFPNMSSN	TEWKTWCD	ECWVFHXZ	EYWCTFIH	WCPWNTFS	PJJUMUWR	GSERTXE	JONEPXBH	VWPKMCZG	GHGYJHID	CLGVVL 2	42 6
7834SIHV	WIDFRUU	JETYMEXF	ZVTF*								
INDEX1	ENXRV	TSFSFVVP	VSKGTLHI	DTUDJTC	GUMYTUXX	YAQITRRM	JBMLGNOE	YKIDMGCS	QWLLHXFT	I YZTUYQ	HBUAQJEV
WMGLHWC	KBWIXJOU	JGCYCLIJ	UFTQWPE	TIOLGQIJ	UIMQDSMP	JCQDKFR	LRHWRWUH	YUCEJUGX	YGYANUCO	YSRZOU 5	41 4
1331JCFQ	ILNHWZBL	XZTCHMTG	BAGG*								
INDEX1	EIJQJ	TIOBLRXP	ZTFQSBNT	JRTKQEVJ	LLMEVWRI	LNFWJJIJ	PXKTFJJD	CXVSDCAD	FWQRYILC	S HWXQFX	QMUIUJFB
QXDMQDF	QDIOAMPN	GPSKMKSM	CULCNEKO	HMEXYMUK	DMXGSQZN	INXLODI	TEWMHFMJ	VXJHQZUI	RWOKKYGJ	HGNYGU 6	55 2
5652GJSJ	FPEUFZEV	EIRHBTWC	ADHN*								
INDEX1	EPBGP	WHMFYIUQ	JIOVGHBA	KJPPFVTV	GDMNHTJV	ZBCGMVNY	YILNKAUS	FXXMYXBG	QUMKGEVB	C WZKUM	RBYPGLYC
XQAMMDMP	NGRQHPVM	YULEFLYA	DCQXXPAG	CLPVV CXX	ZMONQJNW	IGTHEB	UQIAGBNO	PJIRWFOQ	QOQWVONE	VJGLFS 2	48 2
0657AYIG	THCKJJHR	NMLNQNJE	HVAZ*								
INDEX1	EJNMP	CQGBWIL	EVYBQCFV	GDNHJDIM	RZZCILUJ	AGVMQRRX	HVPLFUGI	WRBSSGND	IUEXSSTB	H VRTJWA	ODHEDUFT
YTTMLJIO	IDSVMQHE	XPJFLPHK	KCGUGRBR	WRNKKDDH	DQGTBDMI	YDXPJUU	RSGSSWVC	JBWRPGSS	WDDGGZWY	WXQCG 4	72 5
9879FYCT	RYCCVBOI	TSIJYGTQ	GRWS*								
INDEX1	EIPHG	SPVOXKWY	AUCHRXSZ	IKXRFZEK	BTEFJRRL	FWPFYTUC	LYBUAJKJ	VRTZHAFV	SUDYVKSS	M RFSGHQ	JCYMBSRC
PJFCSJDT	VCMQHRJ	VIJCFQKO	JKSSXLVP	PUCPIJUO	LRJFIMTU	LXYSZU	TNARNLFL	GICIWOUR	FIMYKNQY	UHTWOU 8	92 1
8586DJEN	EGDYNLYM	ZAUJ IUXU	HJOA*								
INDEX1	ETINF	CKJDYVLR	QGEDCTML	NHMMVQPN	ZNNMMCKT	JHYVUMXV	KVFYAIXN	KXXWLHFN	EBPJLOOJ	U WBOQFB	QSHMURYH
JSCUYFZB	QFWPKYLY	CIYHYKVI	NOTVCCGO	UXFOAVRB	OPLOYTNR	UJCEKJF	EPNGKEZO	KIVPCIKI	WNBWBBFK	HDWRE 4	77 2
6631TVDF	TAVGXCD	QNBKXNZB	MBYQ*								
INDEX1	ENAPM	BEVHPIFW	GFBWKLIT	KQSYIIFI	PTPIRNSH	VYZNYASY	CFAITYGO	QMRBAGPA	NGJVTGDG	K GOMJDD	PYKIKSXS
Q7RXVHTM	DAFWXTUT	RVCPFUNC	GVQEEVCF	ZVZUXTHI	IMDGUJSQ	UHYPKID	YVRXQXQJ	OJQUJSID	HLPGWFLS	QPLFGM 1	66 3
5371WPFS	CWSGEMKC	TGUXVGVX	MCQW*								
INDEX1	ENQNM	OYTRQYKO	ORIAAKMJ	CPDVVICM	NPUSMUUM	KLBMVDDB	GHWKBVZO	DFMJOUTG	QRSCTHNB	K VBCCIC	KUCOOLIG
FMESHJNG	QBKHMFOR	YLHULERI	GUHMOHVE	WHMFYPYD	HRLDZWCJ	MXHDSRY	HHAIPAJB	FFWRCKDD	IXMTXYL	OFIQWH 9	01 5
2777WGT	CUGHXJRH	GJAJTCRIJ	HXIE*								
INDEX1	ERPKR	MBVEOUVI	RMKQMXCU	NJGGTIUH	JYYVQDQY	ZIMUEKVO	ZFHTBLWO	SJEIHPCR	CBFFSGDM	P UHNVNO	YYWCGXIM
IGKHVQVB	FBTXMMTJ	MSCFQERT	ASBLVSHD	NNFMWYDS	QETDJANV	AKFTRHO	QIXPBMED	KSEDPQCU	GYRDOEA	DMIUOD 7	21 2
5927LIZI	QKQWCPY	CUJAMINV	HCYN*								
INDEX1	EXTWY	REKEQDUW	XS IUXBVC	ENLPKHNE	WRMLNASI	ULSFHGGG	MNFKYPXY	CDNTLMXF	LHPFINRM	E RLICME	EEENXPWC
PCRYPXF	KXHCVZZJ	NPHXXFST	PYHKCWMN	QEFQPDJD	ZCNMOKWI	ZRDXXMI	HRGCYJNE	NOZJMTYY	EWHONFXR	TTGJTJ 0	35 5

FHURPCDD	CKRDSRDU	BNKMYUKL	RVMVISLD	OWYSBIOC	JNHCOCSZ	EKKFAFH	DFWVYPOU	DYTCQQOI	PJEIJSWW	ZTGTNY 1	44	8
4565QCKS	HDPQZFWO	RIESCKZG	BCWM*									
INDEX1	EQAKG	MLPWBNR	QAMHGQDW	RCIFIDWO	BRCCCHHA	LDRDLUWP	NLDUMQCK	WEFMWVJD	EYKRYVNN	U THULBI	WHJJIVYI	
LWXPFLMP	YNKCRFPH	DGQWZLTR	DSIXTMUC	PNCMLJJB	PFZGSUXL	MROKMOV	KYUSINMQ	LTUSHUWV	WSQVGOQL	QRTMXD 9	99	4
5564YDIX	YSWZMJVU	LJAHNLMW	RFIS*									
INDEX1	EISEV	EPRNYYS	NURVUXY	MUMGTFTT	RCLAYFOO	UNYXTOX	ICBWWQNK	LZFWJXOR	NHJEBVSH	X PYFCRO	DVVGHZXA	
PURFQSYD	LEGRQBOR	VRRQXBSG	HLIUHDBA	KNQIVXHQ	VSRKKNKV	FJVHJHN	DQHILBNHR	UBQJBRXR	BZFLDRGS	RLTMUW 3	42	6
5629PQZU	WDDAUJAW	YTOHFVKI	FFSK*									
INDEX1	EGBJB	XSIVHHWU	SORORDJP	SYWHRFIN	FQYKAKHA	NOSHJHYG	GCYDCXT	JXDUBEJF	ICRPHTOP	S GYLCOV	IGLEPPIK	
ZGTIZNSM	NYBUHHMF	ADILKNES	LSPBHKUT	WUWAEWJE	FMVTKHOG	KFVBEGK	CGUPHEPB	MDGFNUMR	NIEWFLNJ	GMHUZO 6	90	4
3856VKJO	WQMUDSFI	PLQEAACY	CCNL*									
INDEX1	EVMHO	UVYJQAOO	MZPTHNFS	HDXFKYLB	MYIIUNWO	SJPECNSQ	JUEKQMSO	YOVDQDKD	PFBSHFIL	H BSBSIT	BXENAZLF	
IPNSCBGT	EQLWENXX	FQNHGFPZ	TUEDVGYA	CIRDHOUE	KVMPKBON	EDLJCKX	WAKYIGJJ	YIKTTMYE	WCRDLZWP	JPLUDT 2	86	1
1124PUNS	WTSWQDSH	MNXFAMPZ	YNOQ*									
INDEX1	EXTBY	DCAGTTKY	WIJPOXWV	XHRRPIHG	CERKCAHG	XVISPFLE	IUBVQOHN	NIQQRHBY	AUKEWVSN	V XUGRNH	SKRMNGCJ	
QOZLPTVJ	UHOENJIY	YXEOFTDJ	DCYXODJB	WXYIARPO	CKBXAJZM	BCRFTJH	FSHESQOQ	PEHXTQT	HIGLNVTR	NYQYNG 3	73	6
7121WPVJ	DJXUFWCB	QZXLBERR	MGMW*									
INDEX1	ENDNW	VQJITYUF	AZJBCYHL	GNVDVSOY	PDHRGNGN	DNYSRMMN	CRWRKRHC	UWJVBYDQ	EYAGOLPG	X MBWBUR	DWRFPEAV	
QFXRRNQ	KIPKFKWK	RWGRAWR	SWDAHPAZ	YRUWLTWI	YSCWLBCH	HMDJORD	OKYCJGHK	NFRYJVTL	KWCJDUYJ	BFAZAY 6	41	2
5111NJJAJ	VWJRNFOK	RMRTTGGR	XZDY*									
INDEX1	EARTH	GMWIDIMH	DHNDRIGK	RWRDIGVI	GLHGIFDS	NVKHKEAR	KEZJMVQX	TYIYDXJP	WFGCQXWK	D PWZOMS	ZDTETIID	
WKWNLPRO	BMQYVPLE	HZWIVDWH	GPPUWNHQ	JCCBPGSH	BMFOKELV	EUDXNVL	BZUSRZIR	GLWHWVUE	NRMXHBX	YSMIIC 3	65	1
1341LQWS	HITKDIUI	JQBWDJDV	MJGO*									
INDEX1	EgzQV	EMEXHKMJ	BKBYKPSX	WTBNIQPB	JKWHMDSV	KJSTIWKI	NDTQRTHO	EYNSXVQB	PUPJQGLU	F EBDFRU	FQDLQYNP	
GERRNJPR	HORYBCNC	QWARVDDR	FIKDYGFX	WJXAQOLM	TKYNNMZG	OJIYRTP	FKEQUAPU	QKWSLMUM	HYWURUYU	DWDFLX 2	33	6
4213RPSQ	TLSUEXPM	LYCBNGHB	NROK*									
INDEX1	EEDDL	IAWZHxVE	CDFBCKOL	QJWVWIJV	CKQOFOSQ	IYBCRUSH	WMIQLWVG	SETTUQWG	TGRLEJJK	C QLJXDH	SPFQSTUD	
LGNIZZXL	LWNAPBGP	PVAKAUJK	VL AJKRNG	DMHNGLAZ	JTLFMSNO	QHRJWPW	DOERCDB	UQQKRPRN	REPVUWMX	QJQUPD 2	61	2
7368RKBV	XFPWPSPJ	EIMGIFVS	DCSI*									
INDEX1	ECXHH	VRFFCNRB	UYXTIFIO	PXVHWGDx	BBRJYNND	ZUHGPTIU	NGGMTDQW	UBQNLHVB	NSQGPCVI	E THQJX	DBVAMLGG	
QLQYZQHY	URIGBBSC	CNKQRQEX	XUQGRSXN	CGTSTMJY	FTUUMFOQ	DXKLDDM	GGIRLMII	HMSDQMEC	URYTTMHK	YWPPON 0	87	3
8751SQPP	KYRJZZDS	PSEUGVNR	WXSS*									

DRALPNUM-DIAGNOSTICS DE DECODAGE

CARACTERE INVALIDE/FIELD PARM SUR CARTE JCL EXEC: 50,123?56

|

SUSPENSION DU PROGRAMME

A3.1.10

FORMAT INVALIDE POUR PARAMETRE(S) SUR CARTE JCL EXEC (RAPPEL: NB DE LIGNES PAR PAGE: AU PLUS 3 DIGITS-NB ALEAT INITIAL: 6 DIGITS)
SUSPENSION DU PROGRAMME

Les paramètres introduits étaient les suivants: '1234,56789' .

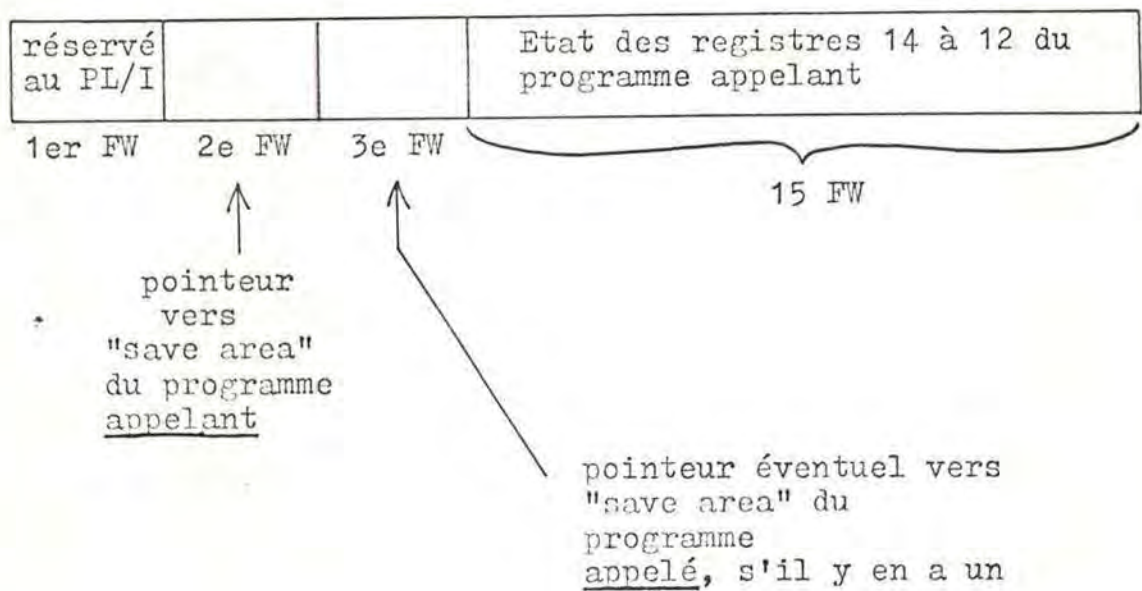
A3.1.11

Annexe 3.2 - Listings de DBALPNUM

Nous présentons ici au lecteur les listings de chacun des quatre modules constitutifs de DBALPNUM. Auparavant, voici deux remarques à l'intention de celui qui désirerait examiner de plus près l'un ou l'autre de ceux-ci.

A3.2.1. Etablissement des liens d'entrée, de branchement et de sortie.

Les premières instructions d'un programme Assembler réalisent toujours ce que nous appellerons, un peu abusivement, le "sauvetage des registres" (il s'agit en fait du sauvetage de leurs contenus) du programme appelant dans une zone de 18 fullwords(FW) réservée dans ce dernier et organisée comme suit:



On constate que les zones de sauvetage sont chaînées les unes aux autres dans l'ordre de succession des appels de programmes.

Dans le cas qui nous occupe, l'"initiateur" de programmes provoque un branchement vers le point d'entrée du module CSECT1 en exécutant une instruction

```
BALR 14,15
```

(le registre 15 ayant été chargé au préalable avec l'adresse du point d'entrée).

Soit dit en passant, ceci explique aussi la présence, en fin de

CSECT1, et après la restauration des registres, d'une instruction

BR 14 ,

qui "rend la main " à l'initiateur."

Enfin, au cours de l'exécution d'un programme quelconque, le registre 13 est censé contenir en permanence l'adresse de la "save area " du programme en cours. Le programmeur, une fois encore, est responsable du chargement et de la restauration de ce registre en début et en fin de programme. Il lui incombe aussi de s'assurer de la maintenance du contenu de ce registre durant l'entièreté de l'exécution de son programme.

Ces quelques considérations sont suffisantes pour comprendre le mécanisme des liens d'entrée, de branchement et de sortie, que nous détaillons maintenant :

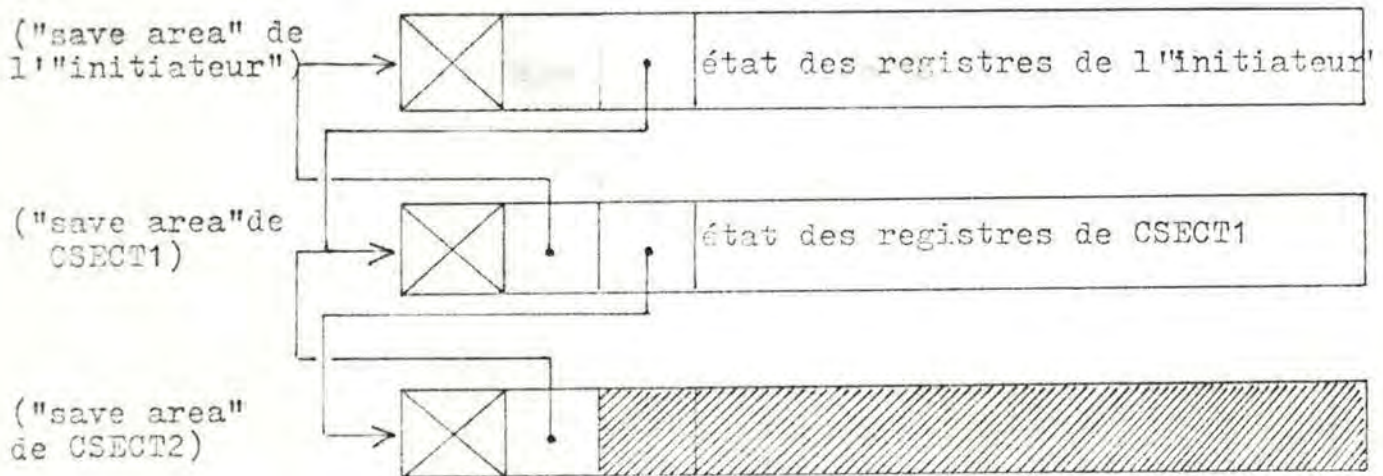
Liens d'entrée:

- sauvetage des registres 14 à 12 (dans cet ordre) du programme appelant dans la " save area " de celui-ci (du 4e au 18e FW);
- désignation et chargement du(des) registre(s) de base du programme courant;
- garnissage du 2e FW de la " save area " du programme courant avec l'adresse de la " save area " du programme appelant (qui figure toujours dans le registre 13). Opération de chaînage " amont ";
- chargement du registre 13 avec l'adresse de la " save area " du programme courant;
- garnissage du 3e FW de la " save area " du programme appelant avec l'adresse de la " save area " du programme courant.

Opération de chaînage " aval ".

Considérons par exemple l'enchaînement : " initiateur " → CSECT1 → CSECT2.

Après l'établissement des liens d'entrée de CSECT2, les zones de sauvetage sont organisées comme suit :



(la partie hachurée est inutilisée à ce moment; les mots marqués d'une croix sont réservés à un autre usage).

Liens de branchement (du module principal CSECT1 vers CSECT2, CSECT3 ou CSECT4):

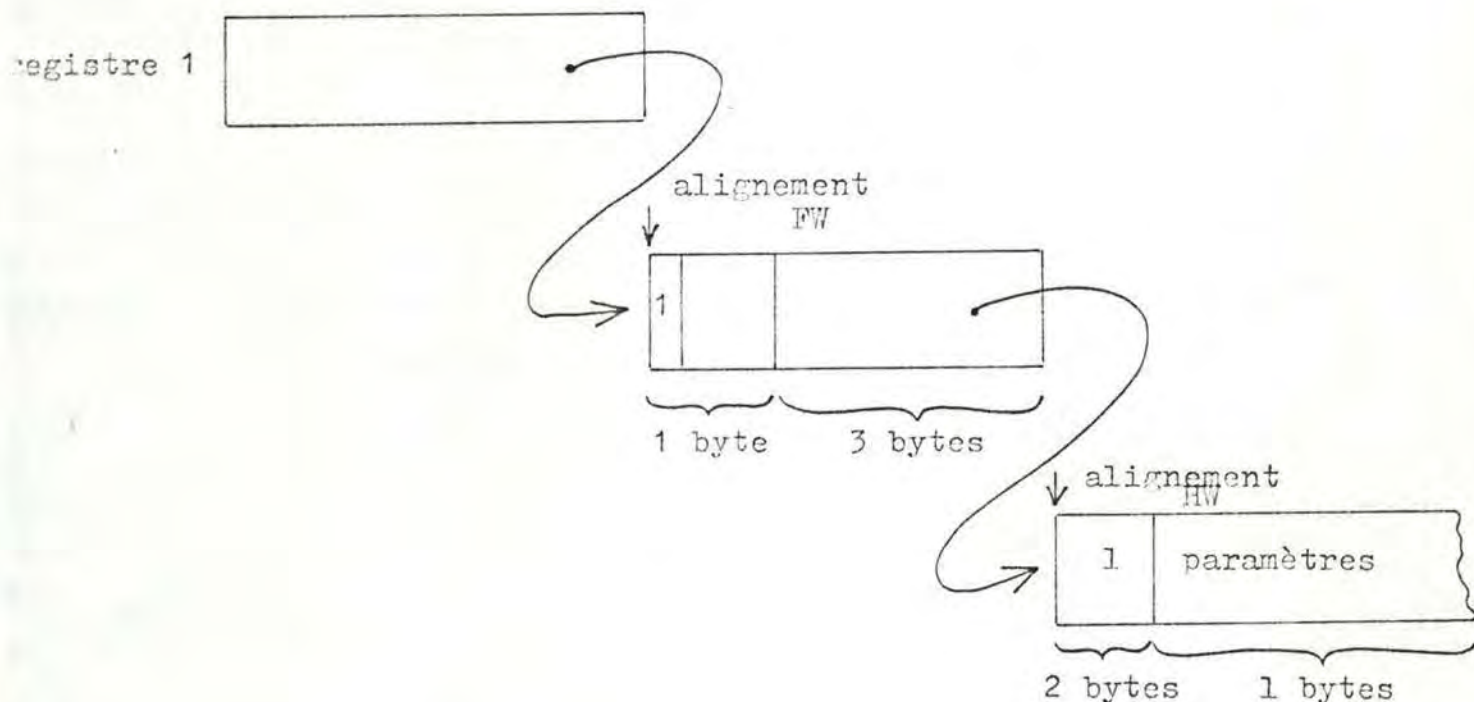
- chargement du registre 14 avec l'adresse du point d'entrée du module où on désire brancher;
- branchement à cette adresse.

Liens de sortie (retour au programme appelant):

- restauration du registre 13 : on y charge l'adresse de la " save area " du programme vers lequel on va brancher; cette adresse est tirée du 2e FW de la " save area " du programme courant;
- restauration des registres 14 à 12 à partir de la " save area " du programme appelant (du 4e au 18e FW);
(variante dans le cas du retour de CSECT3 ou CSECT4 au module principal CSECT1: chargement du registre 14 avec l'adresse de retour, puis restauration des registres 15 à 12 à partir de la " save area " de CSECT1 (du 5e au 18e FW); quant à la programmation du retour de CSECT2 vers CSECT1, elle est légèrement plus complexe, quoique basée sur le même principe: pour plus de renseignements, nous renvoyons le lecteur intéressé aux listings présentés plus loin);
- branchement à l'adresse de retour (contenue dans le registre 14).

A3.2.2 Le passage de paramètres au programme DBALPNUM via la carte JCL EXEC

Lors de l'entrée dans le programme, les registres ou zones de mémoire intervenant dans le mécanisme de passage de ces paramètres se présentent comme suit :



"1" désigne la longueur (exprimée en binaire) de la zone contenant les paramètres, zone dont ont été extraits les quotes délimiteurs qui figuraient sur la carte JCL EXEC.

Les opérations effectuées préalablement au "décodage" des paramètres sont les suivantes :

- "masquage" du premier byte du mot pointé par le registre 1 (le premier bit de ce mot est positionné à 1, indiquant qu'il s'agit d'un mécanisme d'adressage indirect), afin d'"isoler" l'adresse figurant sur les trois bytes suivants;
- sauvetage du contenu de ce mot dans un FW pour exploitation ultérieure.

EXTERNAL SYMBOL DICTIONARY

PAGE 1
09.59 5/17/76

SYMBOL TYPE ID ADDR LENGTH LD ID

CSECT1	SD	01	000000	001650	
IMPRESS3	LD		000744		01
REINIT	LD		0006FF		01
FINCSECT1	LD		0007F6		01
SWITCHES	LD		000830		01
ZONE0115	LD		000848		01
IOAREA2	LD		00091C		01
TABLE3	LD		0008A1		01
ZONETAMP	LD		000DB9		01
CSECT2	FR	02			
CSECT3	FR	03			
CSECT4	FR	04			
EW	FR	05			
TRTABLEA	FR	06			

```

1 * MODULE CSECT1, RESPONSABLE DU DECODAGE DES CARTES DE TYPES P ET A ET
2 * DE L'IMPRESSION DES DIAGNOSTICS DE DECODAGE (FICHIER FICHDIAG)
3 * *****
4 * CONVENTIONS DE PROGRAMMATION:
5 * -----
6 * SWITCHES EST UNE ZONE DE 8 BITS CONSTITUANT CHACUN UN INDICATEUR OU
7 * SWITCH, DONT LES 6 PREMIERS SONT UTILISES DANS CSECT1. DE GAUCHE A
8 * DROITE (EN PARCOURANT DONC LES BITS PAR ORDRE DE POIDS DECROISSANT),
9 * ON A:
10 * - SW1: INITIALISE A 0 ET POSITIONNE A 1 DES QU'UNE CARTE DE TYPE A A
11 *     ETE LUE. A PARTIR DE CE MOMENT, LA LECTURE DE TOUTE CARTE DE
12 *     TYPE P EST CONSIDEREE COMME ERRONEE;
13 * - SW2: INITIALISE A 0 ET POSITIONNE A 1 DES QU'UNE CARTE DE TYPE P
14 *     DEC OU DEC&GEN A ETE LUE. LA PRESENCE D'UNE CARTE DE TYPE P
15 *     EST OBLIGATOIRE EN TETE D'INPUT, INDIOUANT CE QUE DESIRE L'U-
16 *     TILISATEUR, A SAVOIR: UN DECODAGE SEUL OU UN DECODAGE ACCOM-
17 *     PAGNE DE GENERATION. PAR CONSEQUENT, SI, LORS DE LA LECTURE
18 *     DE LA 1ERE CARTE DE TYPE A, SW2 EST TOUJOURS A 0, LE PROGRAM-
19 *     ME NE PEUT SE DEROULER. PAR AILLEURS, DES QU' SW2 A ETE POSI-
20 *     TIONNE, LA LECTURE D'UNE AUTRE CARTE DE TYPE P DEC OU DEC&GEN
21 *     EST CONSIDEREE COMME ERRONEE;
22 * - SW3: INITIALISE A 0 ET POSITIONNE A 1 SI L'UTILISATEUR DESIRE UN
23 *     DECODAGE ACCOMPAGNE DE GENERATION (CARTE DE TYPE P DEC&GEN);
24 * - SW4: INITIALISE A 0 ET POSITIONNE A 1 LORS DU DECODAGE D'UNE CARTE
25 *     DE TYPE P CONTENANT L'EXPRESSION-CLE TRTAPLEA=. A PARTIR DE
26 *     CE MOMENT, LA LECTURE D'UNE CARTE DE TYPE P ANALOGUE EST CON-
27 *     SIDEREE COMME ERRONEE;
28 * - SW5: INITIALISE A 1 ET POSITIONNE DE CETTE FACON, IL INDIQUE QUE
29 *     LA CARTE DE TYPE A QUI VA ETRE LUE EST UNE TETE DE SERIE,
30 *     CONTENANT PAR CONSEQUENT LES RENSEIGNEMENTS ATTENDUS (NOM D'
31 *     INDEX, NR DE GENERATIONS) POUR LA SERIE A VENIR: POSITIONNE A
32 *     0, IL INDIQUE QUE LA CARTE QUI VA ETRE LUE EST UNE CARTE DE
33 *     CONTINUATION;
34 * - SW6: INITIALISE A 0 ET POSITIONNE A 1 LORSQUE L'IMPRESSION D'UN
35 *     DIAGNOSTIC AVEC VISUALISATION DU CURSEUR '!' EST REQUISE.
36 *     CET INDICATEUR EST TESTE DANS LA ROUTINE D'IMPRESSION ET DE-
37 *     TERMINE LA PRESENCE OU L'ABSENCE DU CURSEUR IDENTIFIANT UN
38 *     CARACTERE "LITIGIEUX" ET APPARAISSANT SOUS LA CARTE ERRONEE
39 *     IMPRIMEE DANS LE MESSAGE D'ERREUR.
40 *
41 *

```

000000

```

42 CSECT1  CSECT
43          PRINT NOGEN
44          ENTRY IMPRESS3
45          ENTRY REINIT
46          ENTRY FINCSEC1
47          ENTRY SWITCHES
48          ENTRY ZONEQ15
49          ENTRY IOAREA2
50          ENTRY TABLE3
51          ENTRY ZONEIAMP
52 *      ETABLISSEMENT DES LIENS D'ENTREE
53          STM  R14,R12,12(R13)
54          USING CSECT1,R03,R04
55          LR   R03,R15

```

000000 90EE 000C
000000
000004 183E

00000C

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
000006	4143 0FFF		00FFF	56	LA R04,4095(P03)
00000A	4144 0001		00001	57	LA R04,1(R04)
00000E	5000 388R		0088R	58	ST R13,SAVEAREA+4
000012	1850			59	LR R05,R13
000014	4100 3884		00884	60	LA R13,SAVEAREA
000018	5005 000R		0000R	61	ST R13,R(R05)
				62 *	MASQUE LE 1ER BYTE DE LA ZONE ADRESSE DES PARAMETRES EVENTUELLEMENT
				63 *	PASSES AU PROGRAMME
00001C	9400 1000	00000		64	NI 0(R01),X'00'
				65 *	SAUVETAGE DE L'ADRESSE DANS FW
000020	58F0 45F8		015F8	66	L R14,=V(FW)
000024	0203 F000 1000 00000 00000			67	MVC 0(4,R14),0(R01)
				68 *	OUVERTURES FICHIERS:
				69 *	- FICHCART: CARTES EN ENTREE;
				70 *	- FICHDIAG: DIAGNOSTICS IMPRIMES.
				71	OPEN (FICHCART,(INPUT),FICHDIAG,(OUTPUT))
				79 *	IMPRESSION TITRE FICHDIAG
00003A	4150 391C		0091C	80	LA R05,IOAREA2
00003E	1866			81	SR R06,R06
000040	4170 0085		00085	82	LA R07,133
000044	4180 0020		00020	83	LA R08,LTITR
000048	1878			84	SR R07,R08
00004A	5060 45FC		015FC	85	D R06,=F'2'
00004E	4157 5000		00000	86	LA R05,0(R07,R05)
000052	021E 5000 3F44 00000 00F44			87	MVC 0(LTITR,R05),TITRDIAG
				88	PUT FICHDIAG,IOAREA2
000066	9250 5000	00000		93	MVI 0(R05),C' *'
00006A	5880 45FC		015FC	94	S R08,=F'2'
00006E	4480 3800		00800	95	EY R08,MVC11
				96	PUT FICHDIAG,IOAREA2
000080	0650			101	RCTR R05,R00
000082	021E 5001 5000 00001 00000			102	MVC 1(LTITR,R05),0(R05)
				103 *	
				104 *	BRANCHEMENT MODULE CSECT2
000088	58F0 3E38		00E38	105	L R15,ADRCSECT
00008C	05EF			106	BALR R14,R15
				107 *	
00008E	1888			108	SR R11,R11
000090	18CC			109	SR R12,R12
				110 *	
				111 *	LECTURE D'UNE CARTE
				112	LECTGART GET FICHCART,IOAREA1
				117 *	
0000A0	9507 38CC	008CC		118	CLI IOAREA1,C'P'
0000A4	4770 3282		00282	119	BNE DECTYPA
				120 *	DECODAGE CARTE DE TYPE P
0000A8	9180 3830	00830		121	TM SWITCHES,R'10000000'
0000AC	4710 31E6		001E6	122	BD ERR21
0000B0	0509 38CD 4620 008CD 0152C			123	CLC IOAREA1+1(10),=C'TRTARLEA='''
0000B6	4770 30C6		000C6	124	BNE TM
0000BA	9110 3830	00830		125	TM SWITCHES,R'00010000'
0000BE	4710 31E4		001E4	126	BD ERR2201
0000C2	47F0 310A		0010A	127	B GARNTBLA
0000C6	9140 3830	00830		128 TM	TM SWITCHES,R'01000000'
0000CA	4780 30E6		000E6	129	BZ CLC1

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
00000E	0506 38CD	4642	008CD	01642	130 CLC IOAREA1+1(7),=C'DEC
000004	4780 31FC			001FC	131 BE ERR2202
000008	0506 38CD	4649	008CD	01649	132 CLC IOAREA1+1(7),=C'DEC&&GEN'
00000F	4780 31FC			001FC	133 BE ERR2202
0000F2	47F0 320A			0020A	134 B ERR23
0000F6	0506 38CD	4642	008CD	01642	135 CLC1 CLC IOAREA1+1(7),=C'DEC
0000FC	4770 30F8			000F8	136 BNE CLC2
0000F0	9640 3830		00830		137 OI SWITCHES,B'01000000'
0000F4	47F0 3092			00092	138 B LECTCART
0000F8	0506 38CD	4649	008CD	01649	139 CLC2 CLC IOAREA1+1(7),=C'DEC&&GEN'
0000FE	4770 320A			0020A	140 BNE ERR23
000102	9660 3830		00830		141 OI SWITCHES,B'01100000'
000106	47F0 3092			00092	142 B LECTCART
00010A	9610 3830		00830		143 GARNITRLA OI SWITCHES,B'00010000'
				144 *	DECODAGE CARTE(S) DOMINANT TABLE DE TRANSLATION SPECIALE DES CA-
				145 *	RACTERES A ET REMPLISSAGE DE LA TABLE EN MEMOIRE (TRTABLEA) AU
				146 *	FUR ET A MESURE
00010E	4150 3CA1			00CA1	147 LA R05,ZOMETRAV
000112	4160 3807			00807	148 LA R06,IOAREA1+11
000116	4170 0038			00038	149 LA R07,59
00011A	4470 3806			00806	150 EX R07,MVC12
00011E	9540 3913		00913		151 CLI IOAREA1+71,C' '
000122	4780 316A			0016A	152 BE LA1
000126	4155 003C			0003C	153 LA R05,60(R05)
00012A	4160 38CD			008CD	154 LA R06,IOAREA1+1
00012E	024E 3069	38CC	00069	008CC	155 MVC ZOMETRAV+200(R0),IOAREA1
					156 GET FICHOCART,IOAREA1
000142	9507 38CC			008CC	161 CLI IOAREA1,C'P'
000146	4770 3218			00218	162 BNE ERR24
00014A	9540 3913		00913		163 CLI IOAREA1+71,C' '
00014E	4770 322E			0022E	164 BNE ERR25
000152	4170 0045			00045	165 LA R07,69
000156	4470 3806			00806	166 EX R07,MVC12
00015A	4150 3CA1			00CA1	167 LA R05,ZOMETRAV
00015E	5860 4600			01600	168 L R06,=VIRTARLEA)
000162	41F0 3023			00023	169 LA R15,ZOMETRAV+130
000166	47F0 316E			0016E	170 B CL11
00016A	41F0 3000			00C00	171 LA1 LA R15,ZOMETRAV+60
00016E	9570 5000		00000		172 CL11 CLI 0(R05),C'1111
000172	4780 3198			00198	173 BE TSTSU1V1
000176	0200 6000	5000	00000	00000	174 MVC01 MVC 0(1,R06),0(R05)
00017C	4155 0001			00001	175 LA R05,1(R05)
000180	4166 0001			00001	176 LA R06,1(R06)
000184	195E				177 CR R05,R15
000186	4740 316E			0016E	178 RL CL11
00018A	024A 391C	404E	0091C	0104E	179 MVC IOARFAZ(L26),MFS26
000190	4190 37F6			007F6	180 LA R09,FIMCSEC1
000194	47F0 372A			0072A	181 B IMPRESS1
000198	4155 0001			00001	182 TSTSU1V1 LA R05,1(R05)
00019C	195E				183 CR R05,R15
00019E	4780 31C4			001C4	184 BE S1
0001A2	9570 5000		00000		185 CL1 0(R05),C'1111
0001A6	4780 3176			00176	186 BE MVC01
0001AA	5950 4604			01604	187 C R05,=A(7ZOMETRAV+1)
0001AE	4780 3244			00244	188 BE ERR27

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

FO1FFR73 5/17/76

```

0001R2 9540 5000      00000      189 CLI2    CLI  0(R05),C' '
0001R6 4770 3260      00260      190      RNF  FRR2R
0001RA 4155 0001      00001      191      LA   R05,1(R05)
0001RF 195F          001R2      192      CP   R05,R15
0001C0 4740 31R2      001R2      193      BL   CLI2
194 *
195 *      INSERTION LONGUEUR EFFECTIVE DE TRTABLEA DANS LE BYTE DESCRIPTEUR
196 *
0001C4 5860 4600      01600      197 S1      S   R06,=V(TPTARLEA)
0001C8 58F0 4600      01600      198      L   R14,=V(TPTARLEA)
0001CC 06F0          00000      199      BCTR R14,R00
0001CE 426F 0000      00000      200      STC  R06,0(R14)
201 *
0001D2 9240 3CA1      00CA1      202      MVI  ZOMETRAV,C' '
0001D6 02FF 3CA2 3CA1 00CA2 00CA1 203      MVC  ZOMETRAV+1(256),ZOMETRAV
0001D0 0216 3DA2 3DA1 00DA2 00DA1 204      MVC  ZOMETRAV+257(23),ZOMETRAV+256
0001F2 47F0 3092      00092      205      R    LECTCAPT
206 *      TRAITEMENTS D'ERREURS SPECIFIQUES AU DECODAGE DES CARTES DE TYPE
207 *      P
0001F6 0234 391C 3F64 0091C 00F64 208 ERR21  MVC  IOAREA2(L21),MESS21
0001F0 4190 3092      00092      209      LA   R09,LECTCAPT
0001F0 47F0 373F      0073E      210      R    IMPRESS2
0001F4 4190 3298      00298      211 FRR2201 LA   R09,LOOP
0001F8 47F0 3200      00200      212      R    MVC02
0001FC 4190 3092      00092      213 FRR2202 LA   R09,LECTCAPT
000200 0218 391C 3F99 0091C 00F99 214 MVC02  MVC  IOAREA2(L22),MESS22
000206 47F0 373F      0073E      215      R    IMPRESS2
00020A 0227 391C 3F85 0091C 00F85 216 FRR23  MVC  IOAREA2(L23),MESS23
000210 4190 37FA      007F6      217      LA   R09,FINCSFC1
000214 47F0 373F      0073E      218      R    IMPRESS2
000218 0225 391C 3F00 0091C 00F00 219 FRR24  MVC  IOAREA2(L24),MESS24
00021F 4180 3951      00951      220      LA   R08,IOAREA2+53
000222 4190 37FA      007F6      221      LA   R09,FINCSFC1
000226 9604 3830      00830      222      DI   SWITCHES,R'00000100'
00022A 47F0 373F      0073E      223      R    IMPRESS2
00022E 0248 391C 4003 0091C 01003 224 ERR25  MVC  IOAREA2(L25),MESS25
000234 4180 3998      00998      225      LA   R08,IOAREA2+124
000238 4190 37FA      007F6      226      LA   R09,FINCSFC1
00023C 9604 3830      00830      227      DI   SWITCHES,R'00000100'
000240 47F0 372A      0072A      228      R    IMPRESS1
000244 0252 391C 409A 0091C 0109A 229 FRR27  MVC  IOAREA2(L27),MESS27
000248 59F0 4608      01608      230      C    R15,=A(ZOMETRAV+60)
00024E 47F0 3258      00258      231      BE   LA2
000252 024F 38CC 3069 008CC 00069 232      MVC  IOAREA1,ZOMETRAV+200
000258 4190 37FA      007F6      233 LA2   LA   R09,FINCSFC1
00025C 47F0 372A      0072A      234      R    IMPRESS1
000260 0263 391C 40E0 0091C 010E0 235 ERR28  MVC  IOAREA2(L28),MESS28
000266 5950 4608      01608      236      C    R05,=A(ZOMETRAV+60)
00026A 4780 3280      00280      237      RNL  S2
00026F 024F 38CC 3069 008CC 00069 238      MVC  IOAREA1,ZOMETRAV+200
000274 5B50 460C      0160C      239      S    R05,=A(ZOMETRAV)
000278 4180 395C      0095C      240      LA   R08,IOAREA2+64
00027C 47F0 3288      00288      241      R    LA3
000280 5B50 4608      01608      242 S2    S    R05,=A(ZOMETRAV+60)
000284 4180 3952      00952      243      LA   R08,IOAREA2+54

```

LOC	OBJECT	CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT
000288	4185	8000		00000	244	LA3	LA R08,0(R05,R08)
00028C	4190	37F6		007F6	245		LA R09,FINGSECI
000290	9604	3830	00830		246	0I	SWITCHES,R'00000100'
000294	47F0	372A		0072A	247	R	IMPRESS1
000298	9540	3913	00913		248	LOOP	CLI IOAREA1+71,C' '
00029C	4780	3092		00092	249	RF	LECTCART
					250	GFT	FICHCART,IOAREA1
0002AF	47F0	3298		00298	255	R	LOOP
					256	*	DECODAGE CARTE DE TYPE A
0002B2	95C1	38CC	008CC		257	DECTYP	CLI IOAREA1,C'IA'
0002B6	4780	3200		00200	258	RF	TSTDEBIT
0002BA	0214	391C	4151	0091C	01151	MVC	IOAREA2(L31),MESS31
0002C0	4190	3951		00951	260	LA	R08,IOAREA2+53
0002C4	4190	36F4		006F4	261	LA	R09,PASSCONT
0002C8	9604	3830	00830		262	0I	SWITCHES,R'00000100'
0002CC	47F0	373F		0073F	263	R	IMPRESS2
0002D0	9180	3830	00830		264	TSTDEBIT	TM SWITCHES,R'10000000'
0002D4	4710	32FF		002FF	265	RD	DTYPCART
0002D8	9140	3830	00830		266	TM	SWITCHES,R'01000000'
0002DC	4710	32FF		002FF	267	RD	INIT
0002E0	023A	391C	4166	0091C	01166	MVC	IOAREA2(L32),MESS32
0002E6	4190	37F6		007F6	269	LA	R09,FINGSECI
0002EA	47F0	3744		00744	270	R	IMPRESS3
0002EE	9680	3830	00830		271	INIT	0I SWITCHES,R'10000000'
0002F2	4150	3CA1		00CA1	272	LA	R05,ZONETRAV
0002F6	4160	3857		00857	273	LA	R06,ZONE7280
0002FA	47F0	3306		00306	274	R	SCANINDX
					275	*	
					276	*	DETERMINATION DU "TYPE" DE LA CARTE QUI VIEN D'ETRE LUE: TETE DE
					277	*	SERIE OU CARTE DE CONTINUATION
0002FE	9108	3830	00830		278	DTYPCART	TM SWITCHES,R'00001000'
					279	*	
000302	4780	330F		0030F	280	RZ	TVALCONT
					281	*	DANS LE CAS D'UNE TETE DE SERIE, BALAYAGE ET TEST DE VALIDITE Z0-
					282	*	ME NOM D'INDEX,...
000306	4110	38CD		008CD	283	SCANINDX	LA R01,IOAREA1+1
00030A	1822				284	SR	R02,R02
00030C	4170	0007		00007	285	LA	R07,7
000310	4470	3818		00818	286	FX1	R07,TRT1
000314	4780	3386		00386	287	RC	R,SCANNGEN
000318	4170	331C		0031C	288	LA	R07,R+4
00031C	47F2	7000		00000	289	R	0(R02,R07)
000320	47F0	332C		0032C	290	R	DIGIT
000324	47F0	3348		00348	291	R	SPACINDX
000328	47F0	3372		00372	292	R	FRR33
00032C	5910	4610		01610	293	DIGIT	C R01,=A(IOAREA1+1)
000330	4780	3390		00390	294	RF	FRR3401
000334	4111	0001		00001	295	LA	R01,1(R01)
000338	4170	3805		00805	296	LA	R07,IOAREA1+9
00033C	1871				297	SR	R07,R01
00033E	4780	3386		00386	298	RZ	SCANNGEN
000342	0670				299	RCR	R07,R00
000344	47F0	3310		00310	300	R	FX1
000348	5910	4610		01610	301	SPACINDX	C R01,=A(IOAREA1+1)
00034C	4780	3390		00390	302	RF	FRR3401

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT
000350	4111 0001		00001	303	LA	R01,1(R01)
000354	4170 3805		00805	304	LA	R07,IOAREA1+9
000358	1871			305	SR	R07,R01
00035A	4780 3386		00386	306	BZ	SCANNGEN
00035F	9540 1000	00000		307	CLI3	CLT O(R01),C' '
000362	4770 3398		00398	308	RNF	ERR3402
000366	4111 0001		00001	309	LA	R01,1(R01)
00036A	4670 335F		0035F	310	RCT	R07,CLI3
00036F	47F0 3386		00386	311	B	SCANNGEN
000372	0225 391C	41A1 0091C	011A1	312	FRR33	MVC IOAREA2(L33),MESS33
000378	5810 4614		01614	313	S	R01,=A(IOARFA1)
00037C	4180 3951		00951	314	LA	R08,IOAREA2+53
000380	4181 8000		00000	315	LA	R08,0(R01,R08)
000384	4190 36F4		006F4	316	LA	R09,PASSCONT
000388	9604 3830	00830		317	OI	SWITCHES,R'00000100'
00038C	47F0 373F		0073F	318	B	IMPRESS2
000390	4180 3952		00952	319	FRR3401	LA R08,IOAREA2+54
000394	47F0 33A4		003A4	320	B	MVC03
000398	5810 4614		01614	321	FRR3402	S R01,=A(IOARFA1)
00039C	4180 3951		00951	322	LA	R08,IOAREA2+53
0003A0	4181 8000		00000	323	LA	R08,0(R01,R08)
0003A4	0218 391C	41C7 0091C	011C7	324	MVC03	MVC IOAREA2(L34),MESS34
0003AA	4190 36F4		006F4	325	LA	R09,PASSCONT
0003AE	9604 3830	00830		326	OI	SWITCHES,R'00000100'
0003B2	47F0 373F		0073E	327	B	IMPRESS2
				328	* PUIS BALAYAGE ET TEST DE VALIDITE ZONE NR DE GENERATIONS
0003B6	0005 3805	38A1 00805	008A1	329	SCANNGEN	TRT IOARFA1+9(6),TABLE3
0003BC	4780 342F		0042E	330	BC	R,SAVE0115
0003C0	0228 391C	41E3 0091C	011E3	331	MVC	IOAREA2(L35),MESS35
0003C6	5810 4614		01614	332	S	R01,=A(IOARFA1)
0003CA	4180 3951		00951	333	LA	R08,IOAREA2+53
0003CE	4181 8000		00000	334	LA	R08,0(R01,R08)
0003D2	4190 36F4		006F4	335	LA	R09,PASSCONT
0003D6	9604 3830	00830		336	OI	SWITCHES,R'00000100'
0003DA	47F0 373F		0073E	337	B	IMPRESS2
				338	*	DANS LE CAS D'UNE CARTE DE CONTINUATION, TEST DE SA "VALIDITE"
0003DE	4170 38CD		008CD	339	TVALCONT	LA R07,IOAREA1+1
0003E2	41F0 000F		0000E	340	LA	R14,14
0003E6	9540 7000	00000		341	CLI4	CLT O(R07),C' '
0003EA	4770 3410		00410	342	RNF	ERR37
0003EE	4177 0001		00001	343	LA	R07,1(R07)
0003F2	46F0 33E6		003E6	344	RCT	R14,CLI4
0003F6	4188 0001		00001	345	LA	R11,1(R11)
0003FA	4980 4636		01636	346	CH	R11,=H'5'
0003FE	4740 3438		00438	347	BL	SAVE1680
000402	0258 391C	420F 0091C	0120F	348	MVC	IOAREA2(L36),MESS36
000408	4190 36F4		006F4	349	LA	R09,PASSCONT
00040C	47F0 372A		0072A	350	B	IMPRESS1
000410	0240 391C	4268 0091C	01268	351	FRR37	MVC IOAREA2(L37),MESS37
000416	5870 4614		01614	352	S	R07,=A(IOARFA1)
00041A	4180 3951		00951	353	LA	R08,IOAREA2+53
00041E	4187 8000		00000	354	LA	R08,0(R07,R08)
000422	4190 36F4		006F4	355	LA	R09,PASSCONT
000426	9604 3830	00830		356	OI	SWITCHES,R'00000100'
00042A	47F0 372A		0072A	357	B	IMPRESS1

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

FO1FER73 5/17/76

```

358 *
359 * SAUVETAGE DANS ZONE0115 DU CONTENU DES COL. 1-15 D'UNE TETE DE
360 * SERIE
00042E 020E 3848 38CC 00848 008CC 361 SAVF0115 MVC ZONE0115,IOARFA1
362 *
000434 94E7 3830 00830 363 NI SWITCHES,R'11110111'
364 *
365 * POUR TOUTES CARTES, SAUVETAGES DANS ZONETRAV DU CONTENU DES COL.
366 * 16-71 ET DANS ZONE7280 DU CONTENU DES COL. 72-80
000438 0237 5000 38DB 00000 008DB 367 SAVF1680 MVC 0(56,R05),IOARFA1+15
00043E 0208 6000 3913 00000 00913 368 MVC 0(9,R06),IOARFA1+71
369 *
000444 4155 0038 00038 370 LA R05,56(R05)
000448 4166 0009 00009 371 LA R06,9(R06)
00044C 9540 3913 00913 372 CLI IOAREA1+71,C' '
000450 4770 3092 00092 373 BNE LECTICART
000454 9608 3830 00830 374 OI SWITCHES,R'00001000'
000458 18F5 375 LR R15,R05
00045A 5850 460C 0160C 376 S R05,=A(ZONETRAV)
00045E 18F5 377 LR R14,R05
000460 4150 3CA1 00CA1 378 LA R05,ZONETRAV
000464 4170 3DB9 00DB9 379 LA R07,ZONETAMP
000468 1811 380 SR R01,R01
00046A 1822 381 SR R02,R02
00046C 18AA 382 SR R10,R10
383 * BALAYAGE ET TEST DE VALIDITE ZONETRAV GARNIE CI-DESSUS; AU FINE ET
384 * A MESURE, CONSTITUTION DANS ZONETAMP DE LA VERSION DEFINITIVE DE
385 * LA PICTURE
00046E 44F0 381E 0081E 386 EX2 EX R14,IRT2
000472 4780 3588 00588 387 RC R,LR2
000476 41F0 347A 0047A 388 LA R14,#+4
00047A 47F2 F000 00000 389 R 0(R02,R14)
00047E 47E0 348E 0048E 390 R PARTHOIV
000482 47E0 351C 0051C 391 R QUOTE
000486 47E0 359A 0059A 392 R SPACPICT
00048A 47E0 350E 0050E 393 F ERR3802
394 * TRAITEMENT CORRESPONDANT A LA DETECTION D'UNE PARENTHESE OU-
395 * VRANTE
00048E 1861 396 PARTHOIV LR R06,R01
000490 1865 397 SR R06,R05
000492 4780 35EE 005EE 398 BZ ERR3902
000496 0660 399 RC1R R06,R00
000498 1AA6 400 AR R10,R06
00049A 4980 4638 01638 401 CH R10,=H'25A'
00049E 4720 35E8 005E8 402 PH ERR4001
403 *
404 *
405 * TRANSFERT DANS ZONETAMP DES CARACTERES PRECEDANT LA PARENTHES-
406 * SE OUVRANTE, DEPUIS LE DERNIER ARRET
0004A2 4460 380C 0080C 406 EX R06,MVC13
407 *
0004A6 4176 7000 00000 408 LA R07,0(R06,R07)
0004AA 4151 0001 00001 409 LA R05,1(R01)
0004AE 41F0 0003 00003 410 LA R14,3
411 *
412 * BALAYAGE, TEST DE VALIDITE, PUIS CONVERSION DU FACTEUR DE RE-
PETITION ENTRE PARENTHESSES

```

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
0004R2	44E0	3824	00824	413	EX P14,TRT3
0004R6	4780	3614	00614	414	RC P,FRR41
0004R4	41E0	348E	0048E	415	LA P14,*+4
0004RE	47F2	F000	00000	416	R O(R02,P14)
0004C2	47E0	3508	00508	417	B FRR3801
0004C6	47E0	34CE	004CE	418	R PARTHEEP
0004CA	47E0	3508	00508	419	B FRR3801
0004CE	1861			420	PARTHEEP LP P06,R01
0004D0	1865			421	SR P06,R05
0004D2	4780	35FE	005FE	422	R7 FRR3902
0004D6	0660			423	RCTR P06,R00
0004D8	41E0	0007	00007	424	LA P14,7
0004DC	88E0	0004	00004	425	SLA P14,4
0004E0	16F6			426	OR P14,R06
0004E2	44E0	382A	0082A	427	EX P14,PACK
0004E6	4FE0	3838	00838	428	CVR P14,DW
0004EA	49E0	463A	0163A	429	CH R14,=H'1'
0004EE	47D0	3622	00622	430	RNH FRR42
0004F2	1AAE			431	AR R10,R14
0004F4	49A0	4638	01638	432	CH R10,=H'256'
0004F8	4720	3608	00608	433	RH FRR4002
0004FC	58E0	45FC	015FC	434	S R14,=F'2'
				435 *	
				436 *	EXTENSION DANS ZONETAMP DU CARACTERE PRECEDENT
000500	44E0	3812	00812	437	EX P14,MVC14
				438 *	
000504	417E	7002	00002	439	LA R07,2(R14,R07)
000508	4111	0001	00001	440	LA R01,1(R01)
00050C	1851			441	LR R05,R01
00050E	18FE			442	LR P14,R15
000510	18E1			443	SR R14,R01
000512	4780	36D0	006D0	444	RZ TESTSUIT
000516	06E0			445	RCTR P14,R00
000518	47E0	346F	0046E	446	R EX2
				447 *	TRAITEMENT CORRESPONDANT A LA DETECTION D'UN QUOTE, DEBUT DE
				448 *	CSTE LITTERALE
00051C	1861			449	LR R06,R01
00051E	1865			450	SR R06,R05
000520	4780	3538	00538	451	BZ LR1
000524	1AA6			452	AR R10,R06
000526	49A0	4638	01638	453	CH R10,=H'256'
00052A	4720	35E8	005E8	454	RH FRR4001
00052E	0660			455	RCTR R06,R00
				456 *	
				457 *	TRANSFERT DANS ZONETAMP DES CARACTERES PRECEDANT LE QUOTE,
				458 *	DEPUIS LE DERNIER ARRET
000530	4460	380C	0080C	459	EX R06,MVC13
				460 *	
000534	4176	7001	00001	461	LA R07,1(R06,R07)
000538	1851			462	LR R05,R01
00053A	4111	0001	00001	463	LA R01,1(R01)
00053E	181E			464	CR R01,R15
000540	4780	362E	0062E	465	RE FRR43
000544	8570	1000	00000	466	CLI O(R01),C''''
000548	4780	355C	0055C	467	RE TSTSUIV2

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT
00054C	41AA 0001		00001	468	LA5	LA R10,1(R10)
000550	49A0 463R		0163R	469		CH R10,=H'256'
000554	4720 360A		0060A	470		BH FRR4003
000558	47F0 353A		0053A	471		B LA4
00055C	4111 0001		00001	472	TSTSUITV2	LA R01,1(R01)
000560	191F			473		CR R01,R15
000562	4780 356F		0056E	474		RE ST
000566	9570 1000	00000		475		CLI 0(R01),C''''
00056A	4780 354C		0054C	476		RE LA5
00056F	5010 3834		00834	477	ST	ST R01,SAVREG01
000572	1815			478		SR R01,R05
000574	5910 45FC		015FC	479		C R01,=F'2'
000578	4700 35ER		005ER	480		BMH FRR3901
00057C	0610			481		BCTR R01,R00
				482	*	
				483	*	TRANSFERT DANS ZONETAMP DE LA CSTE LITTERALE FIGURANT ENTRE
				484	*	QUOTES
00057F	4410 380C		0080C	485		EX R01,MVC13
				486	*	
000582	4171 7001		00001	487		LA R07,1(R01,R07)
000586	5810 3834		00834	488		L R01,SAVREG01
00058A	1851			489		LR R05,R01
00058C	18FF			490		LR R14,R15
00058F	18F1			491		SR R14,R01
000590	4780 3600		00600	492		BZ TESTSUIT
000594	06F0			493		BCTR R14,R00
000596	47F0 346E		0046E	494		B EX2
				495	*	TRAITEMENT CORRESPONDANT A LA DETECTION DU 1ER CARACTERE ESPACE
00059A	5910 460C		0160C	496	SPACPICT	C R01,=A(ZONETRAV)
00059E	4780 35FF		005FF	497		BE FRR3902
0005A2	4161 0001		00001	498		LA R06,1(R01)
0005A6	9540 6000	00000		499	CLI5	CLI 0(R06),C' '
0005AA	4770 363C		0063C	500		BMH FRR44
0005AE	4166 0001		00001	501		LA R06,1(R06)
0005B2	196F			502		CR R06,R15
0005B4	4740 35A6		005A6	503		BL CLI5
0005B8	1861			504	LR2	LR R06,R01
0005BA	1865			505		SR R06,R05
0005BC	4780 3600		00600	506		BZ TESTSUIT
0005C0	1AA6			507		AR R10,R06
0005C2	49A0 463R		0163R	508		CH R10,=H'256'
0005C6	4720 35ER		005ER	509		BH FRR4001
0005CA	0660			510		BCTR R06,R00
				511	*	
				512	*	TRANSFERT DANS ZONETAMP DES DERNIERS CARACTERES, DEPUIS L'AR-
				513	*	RET PRECEDENT
0005CC	4460 380C		0080C	514		EX R06,MVC13
				515	*	
0005D0	4176 7001		00001	516		LA R07,1(R06,R07)
0005D4	47F0 3600		00600	517		B TESTSUIT
				518	*	TRAITEMENTS D'ERREURS SPECIFIQUES A LA NON-VALIDITE DE LA PIC-
				519	*	TURE
0005D8	191F			520	FRR3901	CR R01,R15
0005DA	4780 364R		0064R	521		RE FRR45
0005DE	0221 391C 4286 0091C 01286		01286	522	FRR3902	MVC IDARFA2(L3R),MESS3R

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT	
0005F4	47F0	3656		00656	523	B RECONST	
0005F8	5810	3834		00834	524	ERR3901 L R01,SAVREG01	
0005FC	0610				525	RCTR R01,R00	
0005FF	0217	391C	4208	0091C	01208	526	ERR3902 MVC IOAREA2(L39),MESS39
0005F4	47F0	3656		00656	527	B RECONST	
0005F8	0610				528	ERR4001 RCTR R01,R00	
0005FA	48A0	4638		01638	529	SH R10,=H'256'	
0005FF	0610				530	RCTR2 RCTR R01,R00	
000600	46A0	35FF		005FF	531	RCT R10,RCTR2	
000604	47F0	360A		0060A	532	B FRR4003	
000608	0610				533	ERR4002 RCTR R01,R00	
00060A	0254	391C	42E0	0091C	012F0	534	ERR4003 MVC IOAREA2(L40),MESS40
000610	47F0	3656		00656	535	B RECONST	
000614	0240	391C	4345	0091C	01345	536	ERR41 MVC IOAREA2(L41),MESS41
00061A	4115	0003		00003	537	LA R01,3(R05)	
00061E	47F0	3656		00656	538	B RECONST	
000622	0248	391C	4393	0091C	01393	539	ERR42 MVC IOAREA2(L42),MESS42
000628	0610				540	RCTR R01,R00	
00062A	47F0	3656		00656	541	B RECONST	
00062F	0238	391C	430F	0091C	0130F	542	ERR43 MVC IOAREA2(L43),MESS43
000634	4190	36FF		006FF	543	LA R09,REINIT	
000638	47F0	372A		0072A	544	B IMPRESS1	
00063C	023F	391C	4418	0091C	01418	545	ERR44 MVC IOAREA2(L44),MESS44
000642	1816				546	LR R01,R06	
000644	47F0	3656		00656	547	B RECONST	
000648	0228	391C	445A	0091C	0145A	548	ERR45 MVC IOAREA2(L45),MESS45
00064F	4190	36FF		006FF	549	LA R09,REINIT	
000652	47F0	373F		0073F	550	B IMPRESS2	
					551	* RECONSTITUTION DE LA CARTE ERRONEE EN ZONE D'E/S IOAREA1 A	
					552	* PARTIR DE LA LOCALISATION DANS ZONETRAV DU CARACTERE "LITI-	
					553	* GIFUX" ET DES RENSEIGNEMENTS SAUVES DANS ZONE7280 ET, EVEN-	
					554	* TUELLEMENT, DE CEUX FIGURANT DANS ZONE0115 (POUR UNE CARTE	
					555	* TETE DE SERIF ERRONEE)	
000656	5810	4618		01618	556	RECONST S R01,=A(ZONETRAV-1)	
00065A	1866				557	SR R06,R06	
00065C	1877				558	SR R07,R07	
00065F	5910	461C		0161C	559	C R01,=F'56'	
000662	4700	366E		0066E	560	RNH GRN70115	
000666	5810	461C		0161C	561	S R01,=F'56'	
00066A	4177	0038		00038	562	LA R07,56(R07)	
00066F	4970	463C		0163C	563	GRN70115 CH R07,=H'0'	
000672	4720	3688		00688	564	RH GRNSPAC	
000676	020F	38CC	3848	008CC	00848	565	MVC IOAREA1(15),ZONE0115
00067C	4150	3CA1		00CA1	566	LA R05,ZONETRAV	
000680	4170	3857		00857	567	LA R07,ZONE7280	
000684	47F0	36A6		006A6	568	B GRN71680	
000688	0201	38CC	463E	008CC	0163E	569	GRNSPAC MVC IOAREA1(2),=C'A'
00068F	020C	38CF	38CD	008CF	008CD	570	MVC IOAREA1+2(13),IOAREA1+1
000694	1857				571	LR R05,R07	
000696	5060	461C		0161C	572	D R06,=F'56'	
00069A	5C60	4620		01620	573	M R06,=F'9'	
00069E	5A50	460C		0160C	574	A R05,=A(ZONETRAV)	
0006A2	5A70	4624		01624	575	A R07,=A(ZONE7280)	
0006A6	0237	38DB	5000	008DB	00000	576	GRN71680 MVC IOAREA1+15(56),0(P05)
0006AC	0208	3913	7000	00913	00000	577	MVC IOAREA1+71(9),0(R07)

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FER73 5/17/76

```

0006R2 4180 395F          0095F 578      LA    R08,IOAREA2+67
0006R6 4181 8000          00000 579      LA    R08,0(R01,R08)
0006R8 4190 36FF          006FE 580      LA    R09,REINIT
0006RF 9604 3830          00830 581      OI    SWITCHES,R'00000100'
0006C2 0501 3951 4640 00951 01640 582      CLC  IOAREA2+53(2),=C' '
0006C8 4770 372A          0072A 583      BNE  IMPRESS1
0006CC 47F0 373F          0073F 584      R    IMPRESS2
585 *   EN CAS DE DECODAGE SEUIL, BRANCHEMENT MODULE CSECT3; SINON, BRANCHE-
586 *   MFMT MODULE CSECT4
0006D0 9120 3830          00830 587 TESTSUIT TM SWITCHES,R'00100000'
0006D4 4710 36DE          006DE 588      BO    L
0006D8 58F0 3F3C          00F3C 589      L    R15,ADRCSEFC3
0006DC 07FF          590      BR    R15
0006DF 58F0 3F40          00F40 591 L    L    R15,ADRCSEFC4
0006E2 07FF          592      BR    R15
593 *   PASSAGE DE LA SERIE DE CONTINUATION EVENTUELLE D'UNE CARTE ERRONEE;
594 *   RESYNCHRONISATION SUR TETE DE SERIE SUIVANTE
0006F4 9540 3913          00913 595 PASSCONT CLI IOAREA1+71,C' '
0006F8 4780 36FF          006FF 596      RE  REINIT
597      GET  FICHCART,IOAREA1
0006FA 47F0 36F4          006FA 602      R    PASSCONT
603 *   REINITIALISATIONS DIVERSES AVANT TRAITEMENT D'UNE NOUVELLE SERIE DE
604 *   CARTES DE TYPE A
0006FF 9240 3848          00848 605 RFINIT MVI  ZONE0115,C' '
000702 023A 3849 3848 00849 00848 606      MVC  ZONE0115+1(59),ZONE0115
000708 9240 3CA1          00CA1 607      MVI  ZONETRAV,C' '
00070C 02FF 3CA2 3CA1 00CA2 00CA1 608      MVC  ZONETRAV+1(256),ZONETRAV
000712 0293 3FA2 3FA1 00EA2 00FA1 609      MVC  ZONETRAV+512(148),ZONETRAV+512
000718 4150 3CA1          00CA1 610      LA   R05,ZONETRAV
00071C 4160 3857          00857 611      LA   R06,ZONE7280
000720 1888          612      SR  R11,R11
000722 9608 3830          00830 613      OI    SWITCHES,R'00001000'
000726 47F0 3092          00092 614      R    LECTCART
615 *   IMPRESSION DIAGNOSTICS PREPARES PRECEDEMMENT
000738 0233 391D 391C 0091D 0091C 616 IMPRESS1 PUT FICHD1AG,IOAREA2
00073E 024F 3951 38CC 00951 008CC 621      MVC  IOAREA2+1(52),IOAREA2
622 IMPRESS2 MVC  IOAREA2+53(80),IOAREA1
623 IMPRESS3 PUT FICHD1AG,IOAREA2
000752 9104 3830          00830 628      TM  SWITCHES,R'00000100'
000756 4780 3776          00776 629      RZ  MVC04
00075A 0283 391D 391C 0091D 0091C 630      MVC  IOAREA2+1(132),IOAREA2
000760 924F 8000          00000 631      MVI  0(R08),C'1'
632      PUT  FICHD1AG,IOAREA2
000772 94FB 3830          00830 637      NI  SWITCHES,R'11111011'
000776 0283 391D 391C 0091D 0091C 638 MVC04 MVC  IOAREA2+1(132),IOAREA2
00077C 41CC 0001          00001 639      LA   R12,1(R12)
000780 5990 4628          01628 640      C    R09,=A(FINCSSEC1)
000784 4770 379C          0079C 641      BNE  BR
000788 0217 391C 4504 0091C 01504 642      MVC  IOAREA2(19901),MESS9901
643      PUT  FICHD1AG,IOAREA2
00079C 07F9          648 BR  BR  R09
649 *   TRAITEMENT CORRESPONDANT A LA DETECTION DE FIN DU FICHIER FICHCART
00079E 91C0 3830          00830 650 FINCART TM SWITCHES,R'11000000'
0007A2 4710 37B4          007B4 651      BO  TVALDERN
0007A6 020F 391C 4483 0091C 01483 652      MVC  IOAREA2(L91),MESS91

```

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT
0007AC	4190 37F6		007F6	653	LA	R09,FINCSEC1
0007B0	47F0 3744		00744	654	R	IMPRESS3
0007B4	9108 3830	00830		655	TVALDERN	TM SWITCHES,R'00001000'
0007B8	4710 37CA		007CA	656	RD	DERNDIAG
0007BC	0271 391C	4492 0091C	01492	657	MVC	IOAREA2(L92),MESS92
0007C2	4190 37CA		007CA	658	LA	R09,DERNDIAG
0007C6	47F0 3744		00744	659	R	IMPRESS3
				660 *		IMPRESSION DERNIER DIAGNOSTIC, CONCERNANT LE NR TOTAL D'ERREURS
				661 *		DETECTEES EN COURS DE DECODAGE
0007CA	4FC0 3838		00838	662	DERNDIAG	CVD R12,DW
0007CF	0F07 3840 383C	00840	0083C	663	ED	MASQUE,DW+4
0007D4	4180 391C		0091C	664	LA	R08,IOAREA2
0007D8	021B 8000 451C	00000	0151C	665	MVC	0(L9902,R08),MESS9902
0007DE	4188 001C		0001C	666	LA	R08,L9902(R08)
0007E2	0207 8000 3840	00000	00840	667	MVC	0(R,R08),MASQUE
				668	PUT	FICHDIAG,IOAREA2
				673 *		ETABLISSEMENT DES LIENS DE SORTIE
0007F6	580D 0004		00004	674	FINCSEC1	L R13,4(R13)
0007FA	98EC 000C		0000C	675	LM	R14,R12,12(R13)
0007FE	07FF			676	BR	R14
				677 *		
				678 *		
000800	0200 5001 5000	00001	00000	679	MVC11	MVC 1(0,R05),0(R05)
000806	0200 5000 6000	00000	00000	680	MVC12	MVC 0(0,R05),0(R06)
00080C	0200 7000 5000	00000	00000	681	MVC13	MVC 0(0,R07),0(R05)
000812	0200 7001 7000	00001	00000	682	MVC14	MVC 1(0,R07),0(R07)
000818	0000 1000 39A1	00000	009A1	683	TRT1	TRT 0(0,R01),TABLE1
00081E	0000 5000 3AA1	00000	00AA1	684	TRT2	TRT 0(0,R05),TABLE2
000824	0000 5000 38A1	00000	008A1	685	TRT3	TRT 0(0,R05),TABLE3
00082A	0200 3838 5000	00838	00000	686	PACK	PACK DW(0),0(0,R05)
000830	08			687	SWITCHES	DC R'00001000'
000834				688	SAVEPFG01	DS F
000838				689	DW	DS D
000840	4020202020202120			690	MASQUE	DC X'4020202020202120'
000848				691	ZONF0115	DS XL15
000857				692	ZONF7280	DS XL45
000884				693	SAVEARFA	DS 18F
0008CC				694	IOAREA1	DS XL80
00091C	4040404040404040			695	IOAREA2	DC 133C' '
0009A1	0C0C0C0C0C0C0C0C			696	TABLE1	DC 64X'0C'
0009F1	08			697	DC	X'08'
0009F2	0C0C0C0C0C0C0C0C			698	DC	128X'0C'
000A62	0000000000000000			699	DC	9X'00'
000A68	0C0C0C0C0C0C0C			700	DC	7X'0C'
000A72	0000000000000000			701	DC	9X'00'
000A78	0C0C0C0C0C0C0C0C			702	DC	8X'0C'
000A83	0000000000000000			703	DC	8X'00'
000A88	0C0C0C0C0C0C0C			704	DC	6X'0C'
000A91	0404040404040404			705	DC	10X'04'
000A98	0C0C0C0C0C0C0C			706	DC	6X'0C'
000AA1	1010101010101010			707	TABLE2	DC 64X'10'
000AF1	0C			708	DC	X'0C'
000AF2	1010101010101010			709	DC	12X'10'
000AFF	04			710	DC	X'04'
000AFF	1010101010101010			711	DC	47X'10'

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FER73 5/17/76

000R1F	08			712	DC	X'08'
000R1F	1010101010101010			713	DC	67X'10'
000R62	00			714	DC	X'00'
000R63	1010101010101010			715	DC	19X'10'
000R76	00			716	DC	X'00'
000R77	1010101010101010			717	DC	42X'10'
000RA1	0C0C0C0C0C0C0C0C			718	TABLE3	DC 93X'0C'
000RFE	08			719	DC	X'08'
000RFF	0C0C0C0C0C0C0C0C			720	DC	13X'0C'
000C0C	04			721	DC	X'04'
000C0D	0C0C0C0C0C0C0C0C			722	DC	132X'0C'
000C91	0000000000000000			723	DC	10X'00'
000C9B	0C0C0C0C0C0C			724	DC	6X'0C'
000CA1	4040404040404040			725	ZONFTRAV	DC 256C' '
000DA1	4040404040404040			726	DC	24C' '
000DR9	4040404040404040			727	ZONFTAMP	DC 256C' '
000FR9	4040404040404040			728	DC	126C' '
000F37	00					
000F38	00000000			729	ADRCSEFC2	DC V(CSFC2)
000F3C	00000000			730	ADRCSEFC3	DC V(CSFC3)
000F40	00000000			731	ADRCSEFC4	DC V(CSFC4)
000F44	C4C2C1D3D7D5E4D4			732	TITRDIAG	DC C'DRBPNUM-DIAGNOSTICS DE DECODAGE'
000F64	40C3C1D9F3C540C4			733	MESS21	DC C' CARTE DE TYPE P FIGURANT APRES 1ERE CARTE DE TYPE A:'
000F99	40C3C1D9F3C540C4			734	MESS22	DC C' CARTE DE TYPE P REDONDANTE:'
000FR5	40C3C1D9F3C540C4			735	MESS23	DC C' CARTE DE TYPE P DON'T COL 2-8 INVALIDES:'
000FDD	40C3D6C4C540C3C1			736	MESS24	DC C' CODE CARTE INVALIDE (DEVRAIT ETRE P):'
0010D3	40C3C1D9C1C3E3C5			737	MESS25	DC C' CARACTERE DE CONTINUATION INTERDIT SUR 2E CARTE DE TYPE P DONNANT TRTABLEA:'
00104E	40D7C1E240C4C540			738	MESS26	DC C' PAS DE QUOTE INDICATEUR DE FIN DANS CHAINE DE CARACTERES DONNANT TRTABLEA:'
00109A	40C3C8C1C9D5C540			739	MESS27	DC C' CHAINE DE CARACTERES TRTABLEA INVALIDE (AUCUN CARACTERE ENTRE QUOTES DELIMITEURS):'
0010FD	40C3C1D9C1C3E3C5			740	MESS28	DC C' CARACTERE AUTRE QU'ESPACE DETECTE APRES QUOTE INDICATEUR DE FIN DANS CHAINE DE CARACTERES TRTABLEA:'
001151	40C3D6C4C540C3C1			741	MESS31	DC C' CODE CARTE INVALIDE:'
001166	40C3C1D9F3C540C4			742	MESS32	DC C' CARTE DE TYPE P (DFC OU DEC&GEN) MANQUANTE EN TETE D'INPUT'
0011A1	40C3C1D9C1C3E3C5			743	MESS33	DC C' CARACTERE INVALIDE/FIELD NOM D'INDEX:'
0011C7	40C6C9C5D3C440D5			744	MESS34	DC C' FIELD NOM D'INDEX INVALIDE:'
0011F3	40C3C1D9C1C3E3C5			745	MESS35	DC C' CARACTERE INVALIDE/FIELD NB DE GENERATIONS:'
00120F	40F3D9D6D740C4C5			746	MESS36	DC C' TROP DE CARTES CONTINUATION DE TYPE A (MAX AUTORISE: #4) A PARTIR DE LA CARTE CI-DESSOUS:'
001268	40C3C1D9F3C540C4			747	MESS37	DC C' CARTE DE CONTINUATION INVALIDE-CARACTERE AUTRE QU'ESPACE DETECTE EN COL 2-16:'
001286	40C3C1D9C1C3E3C5			748	MESS38	DC C' CARACTERE INVALIDE/FIELD PICTURE:'
0012D8	40C6C9C5D3C440D7			749	MESS39	DC C' FIELD PICTURE INVALIDE:'
0012F0	40C6C9C5D3C440D7			750	MESS40	DC C' FIELD PICTURE INVALIDE-ENTRAINANT GENERATION D'UNE DOWNEE DE PLUS DE 256 CARACTERES:'
001345	40C6C9C5D3C440D7			751	MESS41	DC C' FIELD PICTURE INVALIDE-PLUS DE 3 CHIFFRES DETECTES APRES PARENTHESE OUVRANTE:'
001393	40C6C9C5D3C440D7			752	MESS42	DC C' FIELD PICTURE INVALIDE-VALEUR INVALIDE ENTRE PARENTHESSES (MIN AUTORISE: 2):'
0013DE	40D7C1E240C4C540			753	MESS43	DC C' PAS DE QUOTE INDICATEUR FIN DE CSTE LITTERALE DANS PICTURE:'
				754	MESS44	DC C' CARACTERE AUTRE QU'ESPACE DETECTE APRES ESPACE FIN D*

LOC OBJCT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FER73 5/17/76

LOC	OBJCT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
00141B	40C3C1D9C1C3E3C5				E PICTURE:'
00145A	40D7C1D9C5D5E3C8			755 MESS45	DC C' PARENTHESE FERMANTE MANQUE DANS PICTURE:'
001483	40C9D5D7E4E340C9			756 MFSS91	DC C' INPUT INVALIDE'
				757 MFSS92	DC C' DERNIERE CARTE DE TYPE A INVALIDE (PRESENCE ERRONEE C* ARACTERE DE CONTINUATION: CARTE DE CONTINUATION INEXISTANTE)'
001492	40C4C5D9D5C9C5D9			758 MFSS9901	DC C' SUSPENSION DU PROGRAMME'
001504	40E2E4E2D7C5D5E2			759 MESS9902	DC C' NOMBRE D'ERREURS DETECTEES:'
00151C	40D5D6D4C2D9C54D			760 FICHOCART	DCR DDNAME=SYSIN,DSORG=PS,MACRF=(GM),EODAD=FINOCART
				814 FICHODIAG	DCR DDNAME=SYSOUT1,DSORG=PS,MACRF=(PM),RECFM=FBA,LRECL=133,B* LKSIZE=1463
000020				868 LTITR	EQU L'TITRDIAG
000035				869 L21	EQU L'MESS21
00001C				870 L22	EQU L'MESS22
000028				871 L23	EQU L'MESS23
000026				872 L24	EQU L'MESS24
00004C				873 L25	EQU L'MESS25
00004B				874 L26	EQU L'MESS26
000053				875 L27	EQU L'MESS27
000064				876 L28	EQU L'MESS28
000015				877 L31	EQU L'MESS31
00003B				878 L32	EQU L'MESS32
000026				879 L33	EQU L'MESS33
00001C				880 L34	EQU L'MESS34
00002C				881 L35	EQU L'MESS35
000059				882 L36	EQU L'MESS36
00004F				883 L37	EQU L'MESS37
000022				884 L38	EQU L'MESS38
000018				885 L39	EQU L'MESS39
000055				886 L40	EQU L'MESS40
00004F				887 L41	EQU L'MESS41
00004C				888 L42	EQU L'MESS42
00003C				889 L43	EQU L'MESS43
00003F				890 L44	EQU L'MESS44
000029				891 L45	EQU L'MESS45
00000F				892 L91	EQU L'MESS91
000072				893 L92	EQU L'MESS92
000018				894 L9901	EQU L'MESS9901
00001C				895 L9902	EQU L'MESS9902
000000				896 R00	EQU 0
000001				897 R01	EQU 1
000002				898 R02	EQU 2
000003				899 R03	EQU 3
000004				900 R04	EQU 4
000005				901 R05	EQU 5
000006				902 R06	EQU 6
000007				903 R07	EQU 7
000008				904 R08	EQU 8
000009				905 R09	EQU 9
00000A				906 R10	EQU 10
00000B				907 R11	EQU 11
00000C				908 R12	EQU 12
00000D				909 R13	EQU 13
00000E				910 R14	EQU 14
00000F				911 R15	EQU 15

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

FO1FEB73 5/17/76

```

          912          END
0015F8 00000000          913          =V(FW)
0015FC 00000002          914          =F'2'
001600 00000000          915          =V(TRTABLEA)
001604 00000CA2          916          =A(ZONETRAV+1)
001608 00000C00          917          =A(ZONETRAV+60)
00160C 00000CA1          918          =A(ZONETRAV)
001610 000008C0          919          =A(IOAREA1+1)
001614 000008CC          920          =A(IOAREA1)
001618 00000CA0          921          =A(ZONETRAV-1)
00161C 00000038          922          =F'56'
001620 00000009          923          =F'9'
001624 00000857          924          =A(ZONE7280)
001628 000007F6          925          =A(FINCSEC1)
00162C E3D9F3C1C2D3C5C1  926          =C'TRTABLEA='''
001636 0005          927          =H'5'
001638 0100          928          =H'256'
00163A 0001          929          =H'1'
00163C 0000          930          =H'0'
00163E C140          931          =C'A '
001640 4040          932          =C' '
001642 C4C5C340404040    933          =C'DEC '
001649 C4C5C350C7C5D5    934          =C'DEC&&GEN'
```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADRCSEC2	00004	000F38	00729	0105
ADRCSEC3	00004	000F3C	00730	0589
ADRCSEC4	00004	000F40	00731	0591
RCTR2	00002	0005FE	00530	0531
BR	00002	00079C	00648	0641
CLC1	00006	0000E6	00135	0129
CLC2	00006	0000F8	00139	0136
CL11	00004	00016E	00172	0170 0178
CL12	00004	0001B2	00189	0193
CL13	00004	00035E	00307	0310
CL14	00004	0003E6	00341	0344
CL15	00004	0005A6	00499	0503
CSFCT1	00001	000000	00042	0054
DECTYPA	00004	0002B2	00257	0119
DERNDIAG	00004	0007CA	00662	0656 0658
DIGIT	00004	00032C	00293	0290
DTYPCART	00004	0002FE	00278	0265
DW	00008	000838	00689	0428 0662 0663 0686
FRR21	00006	0001E6	00208	0122
FRR2201	00004	0001F4	00211	0126
FRR2202	00004	0001FC	00213	0131 0133
FRR23	00006	00020A	00216	0134 0140
FRR24	00006	000218	00219	0162
FRR25	00006	00022E	00224	0164
FRR27	00006	000244	00229	0188
FRR28	00006	000260	00235	0190
FRR33	00006	000372	00312	0292
FRR3401	00004	000390	00319	0294 0302
FRR3402	00004	000398	00321	0308
FRR37	00006	000410	00351	0342
FRR3801	00002	000508	00520	0417 0419
FRR3802	00006	00050E	00522	0393
FRR3901	00004	0005E8	00524	0480
FRR3902	00006	0005EF	00526	0398 0422 0497
FRR4001	00002	0005F8	00528	0402 0454 0509
FRR4002	00002	000608	00533	0433
FRR4003	00006	00060A	00534	0470 0532
FRR41	00006	000614	00536	0414
FRR42	00006	000622	00539	0430
FRR43	00006	00062E	00542	0465
FRR44	00006	00063C	00545	0500
FRR45	00006	000648	00548	0521
FX1	00004	000310	00286	0300
FX2	00004	00046F	00386	0446 0494
FICHCART	00004	001538	00764	0075 0113 0157 0251 0598
FICHDIAG	00004	001598	00818	0077 0089 0097 0617 0624 0633 0644 0669
FIMCART	00004	00079E	00650	0782
FIMCSEC1	00004	0007F6	00674	0046 0180 0217 0221 0226 0233 0245 0269 0640 0653 0925
GARNIBLA	00004	00010A	00143	0127
GRMSPAC	00006	000688	00569	0564
GRN70115	00004	00066E	00563	0560
GRN71680	00006	0006A6	00576	0568
IMPRESS1	00004	00072A	00617	0181 0228 0234 0247 0350 0357 0544 0583
IMPRESS2	00006	00073E	00622	0210 0215 0218 0223 0263 0318 0327 0337 0550 0584
IMPRESS3	00004	000744	00624	0044 0270 0654 0659

SYMBOL	LEN	VALUE	DEFN	REFERENCES
INIT	00004	0002EF	00271	0267
INARFA1	00080	0008CC	00694	0114 0118 0123 0130 0132 0135 0139 0148 0151 0154 0155 0158 0161 0163 0232 0238 0248 0252 0257 0283 0293 0296 0301 0304 0313 0321 0329 0332 0339 0352 0361 0367 0368 0372 0565 0569 0570 0570 0576 0577 0595 0599 0622 0919 0920
INARFA2	00001	00091C	00695	0049 0080 0090 0098 0179 0208 0214 0216 0219 0220 0224 0225 0229 0235 0240 0243 0259 0260 0268 0312 0314 0319 0322 0324 0331 0333 0348 0351 0353 0522 0526 0534 0536 0539 0542 0545 0548 0578 0582 0618 0621 0621 0622 0625 0630 0630 0634 0638 0638 0642 0645 0652 0657 0664 0670
L	00004	0006DE	00591	0588
LA1	00004	00016A	00171	0152
LA2	00004	000258	00233	0231
LA3	00004	000288	00244	0241
LA4	00004	00053A	00463	0471
LA5	00004	00054C	00468	0476
LFACTCART	00004	000092	00113	0138 0142 0205 0209 0213 0249 0373 0614
LPOP	00004	00029A	00248	0211 0255
LR1	00002	000538	00462	0451
LR2	00002	000588	00504	0387
LTITR	00001	000020	00868	0083 0087 0102
L21	00001	000035	00869	0208
L22	00001	00001C	00870	0214
L23	00001	000028	00871	0216
L24	00001	000026	00872	0219
L25	00001	00004C	00873	0224
L26	00001	000048	00874	0179
L27	00001	000053	00875	0229
L28	00001	000064	00876	0235
L31	00001	000015	00877	0259
L32	00001	000038	00878	0268
L33	00001	000026	00879	0312
L34	00001	00001C	00880	0324
L35	00001	00002C	00881	0331
L36	00001	000059	00882	0348
L37	00001	00004E	00883	0351
L38	00001	000022	00884	0522
L39	00001	000018	00885	0526
L40	00001	000055	00886	0534
L41	00001	00004E	00887	0536
L42	00001	00004C	00888	0539
L43	00001	00003C	00889	0542
L44	00001	00003F	00890	0545
L45	00001	000029	00891	0548
L91	00001	00000F	00892	0652
L92	00001	000072	00893	0657
L9901	00001	000018	00894	0642
L9902	00001	00001C	00895	0665 0666
MASONE	00008	000840	00690	0663 0667
MFSS21	00053	000F64	00733	0208 0869
MFSS22	00028	000F99	00734	0214 0870
MFSS23	00040	000FB5	00735	0216 0871
MFSS24	00038	000F00	00736	0219 0872
MFSS25	00076	001003	00737	0224 0873
MFSS26	00075	00104F	00738	0179 0874
MFSS27	00083	00109A	00739	0229 0875
MFSS28	00100	0010FD	00740	0235 0876

SYMBOL	LEN	VALUE	DEFN	REFERENCES
MESS31	00021	001151	00741	0259 0877
MESS32	00059	001166	00742	0268 0878
MESS33	00038	0011A1	00743	0312 0879
MESS34	00028	0011C7	00744	0324 0880
MESS35	00044	0011E3	00745	0331 0881
MESS36	00089	00120F	00746	0348 0882
MESS37	00078	001268	00747	0351 0883
MESS38	00034	0012B6	00748	0522 0884
MESS39	00024	0012D8	00749	0526 0885
MESS40	00085	0012F0	00750	0534 0886
MESS41	00078	001345	00751	0536 0887
MESS42	00076	001393	00752	0539 0888
MESS43	00060	0013DF	00753	0542 0889
MESS44	00063	001418	00754	0545 0890
MESS45	00041	00145A	00755	0548 0891
MESS91	00015	001483	00756	0652 0892
MESS92	00114	001492	00757	0657 0893
MESS9901	00024	001504	00758	0642 0894
MESS9902	00028	00151C	00759	0665 0895
MVC01	00006	000176	00174	0186
MVC02	00006	000200	00214	0212
MVC03	00006	0003A4	00324	0320
MVC04	00006	000776	00638	0629
MVC11	00006	000800	00679	0095
MVC12	00006	000806	00680	0150 0166
MVC13	00006	00080C	00681	0406 0459 0485 0514
MVC14	00006	000812	00682	0437
PACK	00006	00082A	00686	0427
PARTHFFR	00002	0004CF	00420	0418
PARTHQUV	00002	00048F	00396	0390
PASSCNT	00004	0006E4	00595	0261 0316 0325 0335 0349 0355 0602
QUOTF	00002	00051C	00449	0391
RECINST	00004	000656	00556	0523 0527 0535 0538 0541 0547
REINIT	00004	0006FF	00605	0045 0543 0549 0580 0596
R00	00001	000000	00896	0101 0199 0299 0399 0423 0445 0455 0481 0493 0510 0525 0528 0530 0533 0540
R01	00001	000001	00897	0064 0067 0283 0293 0295 0295 0297 0301 0303 0303 0305 0307 0309 0309 0313 0315 0321 0323 0332 0334 0380 0380 0396 0409 0420 0440 0440 0441 0443 0449 0462 0463 0463 0464 0466 0472 0472 0473 0475 0477 0478 0479 0481 0485 0487 0488 0489 0491 0496 0498 0504 0520 0524 0525 0528 0530 0533 0537 0540 0546 0556 0559 0561 0579 0683
R02	00001	000002	00898	0284 0284 0289 0381 0381 0389 0416
R03	00001	000003	00899	0054 0055 0056
R04	00001	000004	00900	0054 0056 0057 0057
R05	00001	000005	00901	0059 0061 0080 0086 0086 0087 0093 0101 0102 0102 0147 0153 0153 0167 0172 0174 0175 0175 0177 0182 0182 0183 0185 0187 0189 0191 0191 0192 0236 0239 0242 0244 0272 0367 0370 0370 0375 0376 0377 0378 0397 0409 0421 0441 0450 0462 0478 0489 0505 0537 0566 0571 0574 0576 0610 0679 0679 0680 0681 0684 0685 0686
R06	00001	000006	00902	0081 0081 0085 0148 0154 0168 0174 0176 0176 0197 0200 0273 0368 0371 0371 0396 0397 0399 0400 0406 0408 0420 0421 0423 0426 0449 0450 0452 0455 0459 0461 0498 0499 0501 0501 0502 0504 0505 0507 0510 0514 0516 0546 0557 0557 0572 0573 0611 0680
R07	00001	000007	00903	0082 0084 0086 0149 0150 0165 0166 0285 0286 0288 0289 0296 0297 0299 0304 0305 0310 0329 0341 0343 0343 0352 0354 0379 0408 0408 0439 0439 0461 0461 0487 0487 0516 0516 0558 0558 0562 0562 0563 0567 0571 0575 0577 0681 0682

SYMBOL	LEN	VALUE	DEFN	REFERENCES
				0682
R08	00001	000008	00904	0083 0084 0094 0095 0220 0225 0240 0243 0244 0244 0260 0314 0315 0315 0319 0322 0323 0323 0333 0334 0334 0353 0354 0354 0578 0579 0579 0631 0664 0665 0666 0666 0667
R09	00001	000009	00905	0180 0209 0211 0213 0217 0221 0226 0233 0245 0261 0269 0316 0325 0335 0349 0355 0543 0549 0580 0640 0648 0653 0658
R10	00001	00000A	00906	0382 0382 0400 0401 0431 0432 0452 0453 0468 0468 0469 0507 0508 0529 0531
R11	00001	00000B	00907	0108 0108 0345 0345 0346 0612 0612
R12	00001	00000C	00908	0053 0109 0109 0639 0639 0662 0675
R13	00001	00000D	00909	0053 0058 0059 0060 0061 0674 0675
R14	00001	00000E	00910	0053 0066 0067 0106 0198 0199 0200 0340 0344 0377 0386 0388 0389 0410 0413 0415 0416 0424 0425 0426 0427 0428 0429 0431 0434 0437 0439 0442 0443 0445 0490 0491 0493 0675 0676
R15	00001	00000F	00911	0055 0105 0106 0169 0171 0177 0183 0192 0230 0375 0442 0464 0473 0490 0502 0520 0589 0590 0591 0592
SAVEAREA	00004	000884	00693	0058 0060
SAVE0115	00006	00042F	00361	0330
SAVE1680	00006	000438	00367	0347
SAVREG01	00004	000834	00688	0477 0488 0524
SCANINDEX	00004	000306	00283	0274
SCANPAGE	00006	000386	00329	0287 0298 0306 0311
SPACEINDEX	00004	000348	00301	0291
SPACEICT	00004	00059A	00496	0392
ST	00004	00056E	00477	0474
SWITCHES	00001	000830	00687	0047 0121 0125 0128 0137 0141 0143 0222 0227 0246 0262 0264 0266 0271 0278 0317 0326 0336 0356 0363 0374 0581 0587 0613 0628 0637 0650 0655
S1	00004	0001C4	00197	0184
S2	00004	000280	00242	0237
TARLE1	00001	0009A1	00696	0683
TARLE2	00001	000AA1	00707	0684
TARLE3	00001	000BA1	00718	0050 0320 0685
TESTSUIT	00004	000600	00587	0444 0492 0506 0517
TITROJAG	00032	000F44	00732	0087 0868
TM	00004	0000C6	00128	0124
TRT1	00006	000818	00683	0286
TRT2	00006	00081F	00684	0386
TRT3	00006	000824	00685	0413
TSTDEFUIT	00004	000200	00264	0258
TSTSUUV1	00004	000198	00182	0173
TSTSUUV2	00004	00055C	00472	0467
TVALCOUNT	00004	00030F	00339	0280
TVALDEFM	00004	000784	00655	0651
ZONETAMP	00001	000080	00727	0051 0379
ZONETRAV	00001	000CA1	00725	0147 0155 0167 0169 0171 0187 0202 0203 0203 0204 0204 0230 0232 0236 0238 0239 0242 0272 0376 0378 0496 0556 0566 0574 0607 0608 0608 0609 0609 0610 0916 0917 0918 0921
ZONF0115	00015	000848	00691	0048 0361 0565 0605 0606 0606
ZONF7280	00045	000857	00692	0273 0567 0575 0611 0924

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

STATISTICS SOURCE RECORDS (SYSIN) = 770 SOURCE RECORDS (SYSLIB) = 2713
 OPTIONS IN EFFECT LIST, NOBACK, LOAD, NORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT = 55
 1083 PRINTED LINES

EXTERNAL SYMBOL DICTIONARY

PAGE 1
10.00 5/17/76

SYMBOL TYPE ID ADDR LENGTH LD ID

CSECT2	SD	01	000000	0003A2	
FW	LD		0001AC		01
IMPRESS3	FR	02			
FINCSEC1	FR	03			
IOARFA2	FR	04			
TARLF3	FR	05			
NRLIGPAG	FR	06			
OLDNUMB	FR	07			
SWITCHFS	FR	08			

1 * MODULE CSECT2, RESPONSABLE DU DECODAGE, DE LA CONVERSION, PUIS DU
 2 * SAUVETAGE DES PARAMETRES EVENTUELLEMENT PASSES AU PROGRAMME; LEUR A-
 3 * DRESSE FIGURE DANS FW
 4 * *****
 5 *
 6 *

000000

7 CSECT2 CSECT
 8 PRINT MOGEN
 9 ENTRY FW

10 *
 11 * ETABLISSEMENT DES LIENS D'ENTREE

000000 90FC 000C 0000C
 000000
 000004 183F
 000006 50D0 31BC 001BC
 00000A 184D
 00000C 41D0 3188 00188
 000010 50D4 0008
 000014 5810 31AC 001AC
 000018 1822
 00001A 5840 3380 00380

12 STM R14,R12,12(R13)
 13 USING CSECT2,R03
 14 LR R03,R15
 15 ST R13,SAVEAREA+4
 16 LR R04,R13
 17 LA R13,SAVEAREA
 18 ST R13,R(R04)
 19 *
 20 L R01,FW
 21 SR R02,R02
 22 L R04,=V(INAREA2)

00001E 4851 0000 00000
 000022 4950 3398 00398
 000026 4780 3188 00188
 00002A 4950 339A 0039A
 00002E 4720 311A 0011A
 000032 4111 0002 00002
 000036 0650
 000038 58A0 3384 00384
 00003C 58B0 3388 00388
 000040 58C0 338C 0038C

23 *
 24 * CHARGE DANS R05 LA LONGUEUR DE LA ZONE DES PARAMETRES
 25 LH R05,0(R01)
 26 *
 27 CH R05,=H'0'
 28 BE RETOUR
 29 CH R05,=H'10'
 30 BH ERR11
 31 LA R01,2(R01)
 32 BCTR R05,R00
 33 L R10,=V(TARLE3)
 34 L R11,=V(NRLIGPAG)
 35 L R12,=V(OLONGUMR)

000044 4450 3198 00198
 000048 4780 3060 00060
 00004C 41F0 3050 00050
 000050 47F2 F000 00000
 000054 47F0 30A6 000A6
 000058 47F0 3124 00124
 00005C 47F0 3124 00124

36 * BALAYAGE ET TEST DE VALIDITE ZONE DES PARAMETRES
 37 EX EX R05,TRT
 38 BC R,LA
 39 LA R14,*+4
 40 B 0(R02,R14)
 41 B VIRGULE
 42 B FRR12
 43 B FRR12

000060 4155 0001 00001
 000064 4950 339C 0039C
 000068 4720 308A 0008A
 00006C 0650
 00006F 5650 3390 00390
 000072 4450 319F 0019F
 000076 4F60 3180 00180
 00007A 4960 339F 0039F
 00007E 4740 315A 0015A
 000082 5068 0000 00000

44 * UN SEUL PARAMETRE PASSE AU PROGRAMME: NR ALEATOIRE INITIAL OU NR
 45 * DE LIGNES PAR PAGE DE FICHEGEN.
 46 LA LA R05,1(R05)
 47 CH R05,=H'3'
 48 BH CH1
 49 BCTR R05,R00
 50 O R05,=X'00000070'
 51 EX R05,PACK1
 52 CVR R06,DW
 53 CH R06,=H'20'
 54 BL FRR13
 55 ST R06,0(R11)

LOC. OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FFB73 5/17/76

LOC.	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
00086	47F0 3188		00188	56	B RETOUR
0008A	4950 33A0		003A0	57	CH1 CH R05,=H'6'
0008E	4770 3164		00164	58	BNE ERR14
00092	0650			59	BCTR R05,R00
00094	5650 3390		00390	60	O R05,=X'00000070'
00098	4450 319F		0019F	61	EX R05,PACK1
0009C	0203 C000	31B4 00000	00184	62	MVC 0(4,R12),DW+4
000A2	47F0 3188		00188	63	B RETOUR,
				64	* DEUX PARAMETRES PASSES AU PROGRAMME; UNE VIRGULE LES SEPRE
000A6	1865			65	VIRGULE LR R06,R05
000A8	1851			66	LR R05,R01
000AA	5870 31AC		001AC	67	L R07,FW
000AF	4177 0002		00002	68	LA R07,2(R07)
000B2	1857			69	SR R05,R07
000B4	4950 339C		0039C	70	CH R05,=H'3'
000B8	4720 30F0		000F0	71	BH CH2
000BC	1865			72	SR R06,R05
000BE	0650			73	BCTR R05,R00
000C0	5650 3390		00390	74	O R05,=X'00000070'
000C4	4450 31A4		001A4	75	EX R05,PACK2
000C8	4F70 31B0		001B0	76	CVR R07,DW
000CC	4970 339F		0039F	77	CH R07,=H'20'
000D0	4740 315A		0015A	78	BL ERR13
000D4	5078 0000		00000	79	ST R07,0(R11)
000D8	4111 0001		00001	80	LA R01,1(R01)
000DC	1856			81	LR R05,R05
000DE	4950 33A0		003A0	82	CH R05,=H'6'
000E2	4770 3164		00164	83	BNE ERR14
000E6	0650			84	BCTR R05,R00
000E8	920C A068	0006B		85	MVI 107(R10),X'0C'
000EC	47F0 3044		00044	86	B FX
000F0	4950 33A0		003A0	87	CH2 CH R05,=H'6'
000E4	4770 3164		00164	88	BNE ERR14
000F8	1F65			89	SR R06,R05
000FA	0650			90	BCTR R05,R00
000FC	5650 3390		00390	91	O R05,=X'00000070'
00100	4450 31A4		001A4	92	EX R05,PACK2
00104	0203 C000	31B4 00000	00184	93	MVC 0(4,R12),DW+4
00108	4111 0001		00001	94	LA R01,1(R01)
0010E	1856			95	LR R05,R06
00110	0650			96	BCTR R05,R00
00112	920C A068	0006B		97	MVI 107(R10),X'0C'
00116	47F0 3044		00044	98	B FX
				99	* TRAITEMENTS D'ERREURS SPECIFIQUES AU DECODAGE DES PARAMETRES
00114	0259 4000	3208 00000	00208	100	FRR11 MVC 0(L11,R04),MESS11
00120	47F0 316A		0016A	101	B BRCHIMP3
00124	0231 4000	3262 00000	00262	102	FRR12 MVC 0(L12,R04),MESS12
0012A	5870 31AC		001AC	103	L R07,FW
0012E	4857 0000		00000	104	LH R05,0(R07)
00132	0650			105	BCTR R05,R00
00134	4177 0002		00002	106	LA R07,2(R07)
00138	4450 3192		00192	107	EX R05,MVC
0013C	1817			108	SR R01,R07
0013E	4184 0035		00035	109	LA R08,53(R04)
00142	4181 8000		00000	110	LA R08,0(R01,R08)

LNCR OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FEB73 5/17/76

```

000146 58E0 0004          00004 111 L R14,4(R13)
00014A 508F 0034          00034 112 ST R08,52(R14)
00014E 58F0 3394          00394 113 L R15,=V(SWITCHES)
000152 9604 F000      00000 114 OI 0(R15),B'00000100'
000156 47F0 316A          0016A 115 B BRCHIMP3
00015A 0265 4000 3294 00000 00294 116 ERR13 MVC 0(L13,R04),MESS13
000160 47F0 316A          0016A 117 B BRCHIMP3
000164 0281 4000 32FA 00000 002FA 118 ERR14 MVC 0(L14,R04),MESS14
119 * LIENS DE SORTIE BRANCHEMENT EN IMPRESS3, DE CSECT1, POUR IMPRESSION
120 * DIAGNOSTICS D'ERREURS
00016A 5800 0004          00004 121 BRCHIMP3 L R13,4(R13)
00016E 58F0 3200          00200 122 L R14,ADIMPR3
000172 98F2 0010          00010 123 LM R15,R02,16(R13)
000176 9848 0024          00024 124 LM R04,R08,36(R13)
00017A 5890 3204          00204 125 L R09,ADFNC51
00017E 98AC 003C          0003C 126 LM R10,R12,60(R13)
000182 5830 0020          00020 127 L R03,32(R13)
000186 07FF          128 BR R14
129 * LIENS DE SORTIE RETOUR NORMAL CSECT1
000188 5800 0004          00004 130 RETOUR L R13,4(R13)
00018C 98FC 000C          0000C 131 LM R14,R12,12(R13)
000190 07FF          132 BR R14
133 *
134 *
000192 0200 4035 7000 00035 00000 135 MVC MVC 53(0,R04),0(R07)
000198 0000 1000 8000 00000 00000 136 TRT TRT 0(0,R01),0(P10)
00019E F200 3180 1000 00180 00000 137 PACK1 PACK DW(0),0(0,P01)
0001A4 F200 3180 7000 00180 00000 138 PACK2 PACK DW(0),0(0,R07)
0001AC          139 FW DS F
0001B0          140 DW DS D
0001B8          141 SAVFAREA DS 18F
000200 00000000          142 ADIMPR3 DC V(IMPRESS3)
000204 00000000          143 ADFNC51 DC V(FINCSEC1)
144 MESS11 DC C' FIELD PARM TROP LONG SUR CARTE JCL EXEC (MAX AUTORISE*
: 10 CARACTERES, SEPARATEUR COMPRIS)'
000208 40C6C9C5D3C440D7          145 MESS12 DC C' CARACTERE INVALIDE/FIELD PARM SUR CARTE JCL EXEC:'
000262 40C3C1D9C1C3E3C5          146 MESS13 DC C' VALEUR INVALIDE POUR PARAMETRE NR DE LIGNES PAR PAGE *
(MIN AUTORISE: 20)/FIELD PARM SUR CARTE JCL EXEC'
000294 40F5C1D3C5E4D940          147 MESS14 DC C' FORMAT INVALIDE POUR PARAMETRE(S) SUR CARTE JCL EXEC *
(RAPPEL: NR DE LIGNES PAR PAGE: AU PLUS 3 DIGITS-NR ALEA*
T INITIAL: 6 DIGITS)'
0002FA 40C6D6D9D4C1E340          148 L11 EQU L'MESS11
00005A          149 L12 EQU L'MESS12
000032          150 L13 EQU L'MESS13
000066          151 L14 EQU L'MESS14
000082          152 R00 EQU 0
000000          153 R01 EQU 1
000001          154 R02 EQU 2
000002          155 R03 EQU 3
000003          156 R04 EQU 4
000004          157 R05 EQU 5
000005          158 R06 EQU 6
000006          159 R07 EQU 7
000007          160 R08 EQU 8
000008          161 R09 EQU 9
000009

```

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FE873 5/17/76

00000A				162	R10	EQU	10
00000B				163	R11	EQU	11
00000C				164	R12	EQU	12
00000D				165	R13	EQU	13
00000E				166	R14	EQU	14
00000F				167	R15	EQU	15
				168		END	
000380	00000000			169		=V(IQAREA2)	
000384	00000000			170		=V(TABLE3)	
000388	00000000			171		=V(NBLIGPAG)	
00038C	00000000			172		=V(OLDNUMB)	
000390	00000070			173		=X'00000070'	
000394	00000000			174		=V(SWITCHES)	
000398	0000			175		=H'0'	
00039A	000A			176		=H'10'	
00039C	0003			177		=H'3'	
00039E	0014			178		=H'20'	
0003A0	0006			179		=H'6'	

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADFINCS1	00004	000204	00143	0125
ADIMPR3	00004	000200	00142	0122
BROHIMP3	00004	00016A	00121	0101 0115 0117
CH1	00004	00008A	00057	0048
CH2	00004	0000F0	00087	0071
CSECT2	00001	000000	00007	0013
DW	00008	000180	00140	0052 0062 0076 0093 0137 0138
FRR11	00006	00011A	00100	0030
FRR12	00006	000124	00102	0042 0043
FRR13	00006	00015A	00116	0054 0078
FRR14	00006	000164	00118	0058 0083 0088
EX	00004	000044	00037	0086 0098
FW	00004	0001AC	00139	0009 0020 0067 0103
LA	00004	000060	00046	0038
L11	00001	00005A	00148	0100
L12	00001	000032	00149	0102
L13	00001	000066	00150	0116
L14	00001	000082	00151	0118
MESS11	00090	000208	00144	0100 0148
MESS12	00050	000262	00145	0102 0149
MESS13	00102	000294	00146	0116 0150
MESS14	00130	0002FA	00147	0118 0151
MVC	00006	000192	00135	0107
PACK1	00006	00019E	00137	0051 0061
PACK2	00006	0001A4	00138	0075 0092
RETOUR	00004	000188	00130	0028 0056 0063
R00	00001	000000	00152	0032 0049 0059 0073 0084 0090 0096 0105
R01	00001	000001	00153	0020 0025 0031 0031 0066 0080 0080 0094 0094 0108 0110 0136 0137
R02	00001	000002	00154	0021 0021 0040 0123
R03	00001	000003	00155	0013 0014 0127
R04	00001	000004	00156	0016 0018 0022 0100 0102 0109 0116 0118 0124 0135
R05	00001	000005	00157	0025 0027 0029 0032 0037 0046 0046 0047 0049 0050 0051 0057 0059 0060 0061 0065 0066 0069 0070 0072 0073 0074 0075 0081 0082 0084 0087 0089 0090 0091 0092 0095 0096 0104 0105 0107
R06	00001	000006	00158	0052 0053 0055 0065 0072 0081 0089 0095
R07	00001	000007	00159	0067 0068 0068 0069 0076 0077 0079 0103 0104 0106 0106 0108 0135 0138
R08	00001	000008	00160	0109 0110 0110 0112 0124
R09	00001	000009	00161	0125
R10	00001	00000A	00162	0033 0085 0097 0126 0136
R11	00001	00000B	00163	0034 0055 0079
R12	00001	00000C	00164	0012 0035 0067 0093 0126 0131
R13	00001	00000D	00165	0012 0015 0016 0017 0018 0111 0121 0121 0123 0124 0126 0127 0130 0130 0131
R14	00001	00000E	00166	0012 0039 0040 0111 0112 0122 0128 0131 0132
R15	00001	00000F	00167	0014 0113 0114 0123
SAVFARFA	00004	000188	00141	0015 0017
TRT	00006	000198	00136	0037
VIRGULE	00002	000046	00065	0041

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

STATISTICS SOURCE RECORDS (SYSIN) = 172

OPTIONS IN EFFECT LIST, NODECK, LOAD, NORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT = 55

265 PRINTED LINES

EXTERNAL SYMBOL DICTIONARY

PAGE 1
10.01 5/17/76

SYMBOL TYPE ID ADDR LENGTH LD ID

CSECT3	SD	01	000000	000503	
NRLIGPAG	LD		000278		01
REFINIT	FR	02			
SWITCHES	FR	03			
ZONF0115	FR	04			
ZONFTAMP	FR	05			

```

1 * MODULE CSECT3, RESPONSABLE DE L'IMPRESSION DES ELEMENTS DES GENERA-
2 * TIONS (FICHIER FICHEGEN)
3 * *****
4 * CONVENTION DE PROGRAMMATION (SUITE):
5 * -----
6 * - 547: INITIALISE A 0; POSITIONNE A 1, IL INDIQUE QUE LES PREPARA-
7 * TIFS INITIAUX D'OUVERTURE DU FICHIER FICHEGEN ET D'IMPRESSION
8 * DE SON EN-TETE ONT ETE EXECUTES.
9 *
10 *
11 CSECT3 CSECT
12 PRINT MOGEN
13 ENTRY NBLIGPAG
14 *
15 * ETABLISSEMENT DES LIENS D'ENTREE
16 STM R14,R12,12(P13)
17 USING CSECT3,R03
18 LR R03,R15
19 ST R13,SAVEAREA+4
20 LR R04,R13
21 LA R13,SAVEAREA
22 ST R13,8(R04)
23 *
24 ST R07,SAVREG07
25 L R04,=V(SWITCHES)
26 TM 0(R04),B'00000010'
27 RL L
28 * PREPARATIFS INITIAUX
29 OPEN (FICHEGEN,(OUTPUT))
30 LA R05,IOAREA
31 SR R06,R06
32 LA R07,133
33 LA R08,LTITR
34 SR R07,R08
35 D R06,=F'2'
36 LA R05,0(R07,R05)
37 MVC 0(LTITR,R05),TITREGEN
38 PUT FICHEGEN,IOAREA
39 MVI 0(R05),C' #'
40 S R08,=F'2'
41 EX R08,MVC11
42 PUT FICHEGEN,IOAREA
43 RCTR R05,R00
44 MVC 1(LTITR,R05),0(R05)
45 BAL R09,IMPENTET
46 MVC SAVREG12,=F'9'
47 DI 0(R04),B'00000010'
48 * PREPARATION ET IMPRESSION FLEMENTS DES GENERATIONS: NOM D'INDEX ET
49 * NR DE GENERATIONS (FIGURANT DANS ZONE0115) - VERSION DEFINITIVE DE
50 * LA PICTURE (FIGURANT DANS ZONETAMP) DECOUPEE EN GROUPES DE 116 CA-
51 * RACTERES
52 L R04,=V(ZONE0115)
53 L R05,=V(ZONETAMP)
54 SR R06,R06
55 L R07,SAVREG07

```

000000

000000 90FC 000C
000000
000004 183F
000006 5000 3280
00000A 1840
00000C 4100 327C
000010 5004 0008

00000C

00280

0027C

00008

000014 5070 3270
000018 5840 34F0
00001C 9102 4000
000020 4710 308A

00270

004F0

00000

0008A

00002E 4150 32C4
000032 1866
000034 4170 0085
000038 4180 0021
00003C 1878
00003E 5060 34F4
000042 4157 5000
000046 0220 5000 3350 00000 00350
00005A 925C 5000
00005E 5880 34F4
000062 4480 3264

002C4

00085

00021

004F4

00000

00350

00000

004F4

00264

000074 0650
000076 0220 5001 5000 00001 00000
00007C 4590 315A
000080 0203 3274 34E8 00274 004F8
000086 9602 4000 00000

00000

00000

0015A

004F8

00000

00008A 5840 34FC
00008E 5850 34F0
000092 1866
000094 5870 3270

004FC

004F0

00270

68

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
000098	58C0 3274		00274	69	L R12,SAVREG12
				70 *	
				71 *	BRANCHEMENT IMPENTET SI NB LIMITE DE LIGNES PAR PAGE DE FICHEGEN
				72 *	ATTEINT
00009C	59C0 3278		00278	73	C R12,NBLIGPAG
0000A0	4740 3080		00080	74	BL SR
0000A4	92F1 32C4	002C4		75	MVI IOAREA,C'1'
0000A8	4590 315A		0015A	76	BAL R09,IMPENTET
0000AC	41C0 0007		00007	77	LA R12,7
				78 *	
0000B0	1B75			79	SR SR R07,R05
0000B2	0207 32C5 4001	002C5	00001	80	MVC IOAREA+1(R),1(R04)
0000B8	9240 32CD		002CD	81	MVI IOAREA+9,C' '
0000BC	0205 32CE 4009	002CE	00009	82	MVC IOAREA+10(6),9(R04)
0000C2	9240 32D4		002D4	83	MVI IOAREA+16,C' '
0000C6	5970 34F4		004F4	84	C R07,=F'116'
0000CA	4700 3116		00116	85	BNH TSAUPAG2
0000CE	5060 34F4		004F4	86	D R06,=F'116'
				87 *	
				88 *	BRANCHEMENT IMPENTET SI NR LIMITE DE LIGNES PAR PAGE DE FICHEGEN
				89 *	ATTEINT
0000D2	59C0 3278		00278	90	TSAUPAG1 C R12,NBLIGPAG
0000D6	4740 30F6		000F6	91	BL MVC01
0000DA	92F1 32C4	002C4		92	MVI IOAREA,C'1'
0000DE	4590 315A		0015A	93	BAL R09,IMPENTET
0000E2	41C0 0007		00007	94	LA R12,7
				95 *	
0000E6	0273 32D5 5000	002D5	00000	96	MVC01 MVC IOAREA+17(116),0(R05)
				97	PUT FICHEGEN,IOAREA
0000FA	41C0 0001		00001	102	LA R12,1(R12)
0000FE	0283 32C5 32C4	002C5	002C4	103	MVC IOAREA+1(132),IOAREA
000104	4155 0074		00074	104	LA R05,116(R05)
000108	4670 30D2		000D2	105	BCR R07,TSAUPAG1
00010C	4960 34FC		004FC	106	CH R06,=H'0'
000110	4780 3142		00142	107	BE MVC02
000114	1876			108	LR R07,R06
				109 *	
				110 *	BRANCHEMENT IMPENTET SI NR LIMITE DE LIGNES PAR PAGE DE FICHEGEN
				111 *	ATTEINT
000116	59C0 3278		00278	112	TSAUPAG2 C R12,NBLIGPAG
00011A	4740 312A		0012A	113	BL BCTR1
00011E	92F1 32C4	002C4		114	MVI IOAREA,C'1'
000122	4590 315A		0015A	115	BAL R09,IMPENTET
000126	41C0 0007		00007	116	LA R12,7
				117 *	
00012A	0670			118	BCTR1 BCTR R07,R00
00012C	4470 326A		0026A	119	EX R07,MVC12
				120	PUT FICHEGEN,IOAREA
00013E	41C0 0001		00001	125	LA R12,1(R12)
000142	0283 32C5 32C4	002C5	002C4	126	MVC02 MVC IOAREA+1(132),IOAREA
000148	50C0 3274		00274	127	ST R12,SAVREG12
				128 *	LIENS DE SORTIE RETOUR EN REINIT, DE CSECT1
00014C	5800 0004		00004	129	L R13,4(R13)
000150	58F0 334C		0034C	130	L R14,ADREINIT
000154	98FC 0010		00010	131	LM R15,R12,16(R13)

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT	FO1FER73	5/17/76
000158	07FF			132	RR R14		
				133 *	ROUTINE IMPENTFT, REALISANT L'IMPRESSION DE L'EN-TETE DE PAGE (NU-		
				134 *	MEROTATION DES COLONNES / TITRES DE PAGE) DU FICHER FICHEGEN		
				135 *			
				136 *	PREPARATION ET IMPRESSION 1ERE LIGNE DE NUMEROTATION (CENTAINE)		
00015A	92F1 3328	00328		137	IMPENTFT MVI IOAREA+100,C'1'		
				138	PUT FICHEGEN,IOAREA		
				143 *			
00016C	9240 32C4	002C4		144	MVI IOAREA,C'1'		
				145 *	PREPARATION ET IMPRESSION 2EME LIGNE DE NUMEROTATION (DIZAINES)		
000170	41A0 32CF		002CF	146	LA R10,IOAREA+10		
000174	188B			147	SR R11,R11		
000176	43B0 3502		00502	148	IC R11,=C'1'		
00017A	42BA 0000		00000	149	STC1 STC R11,0(R10)		
00017F	41AA 000A		0000A	150	LA R10,10(R10)		
000182	59A0 34F8		004F8	151	C R10,=A(IOAREA+132)		
000186	4720 319F		0019F	152	RH PUT1		
00018A	418B 0001		00001	153	LA R11,1(R11)		
00018E	49B0 34FF		004FF	154	CH R11,=H'250'		
000192	4740 317A		0017A	155	BL STC1		
000196	48B0 3500		00500	156	SH R11,=H'10'		
00019A	47F0 317A		0017A	157	B STC1		
				158	PUT1 PUT FICHEGEN,IOAREA		
				163 *			
				164 *	PREPARATION ET IMPRESSION 3EME LIGNE DE NUMEROTATION (UNITES)		
0001AC	0283 32C5 32C4	002C5	002C4	165	MVC IOAREA+1(132),IOAREA		
0001B2	41A0 32C5		002C5	166	LA R10,IOAREA+1		
0001B6	188B			167	SR R11,R11		
0001B8	43B0 3502		00502	168	IC R11,=C'1'		
0001BC	42BA 0000		00000	169	STC2 STC R11,0(R10)		
0001C0	41AA 0001		00001	170	LA R10,1(R10)		
0001C4	59A0 34F8		004F8	171	C R10,=A(IOAREA+132)		
0001C8	4720 31E0		001E0	172	RH PUT2		
0001CC	418B 0001		00001	173	LA R11,1(R11)		
0001D0	49B0 34FF		004FF	174	CH R11,=H'250'		
0001D4	4740 318C		0018C	175	BL STC2		
0001D8	48B0 3500		00500	176	SH R11,=H'10'		
0001DC	47F0 318C		0018C	177	B STC2		
				178	PUT2 PUT FICHEGEN,IOAREA		
				183 *			
001FE	0283 32C5 32C4	002C5	002C4	184	MVC IOAREA+1(132),IOAREA		
				185 *			
				186 *	IMPRESSION LIGNE DE SEPARATION		
				187	PUT FICHEGEN,IOAREA		
				192 *	PREPARATION ET IMPRESSION LIGNES TITRES DE PAGE		
00202	0284 32C4 3371	002C4	00371	193	MVC IOAREA,TITRPAG1		
				194	PUT FICHEGEN,IOAREA		
00216	020F 32C5 32C4	002C5	002C4	199	MVC IOAREA+1(15),IOAREA		
0021C	0284 32C4 33F6	002C4	003F6	200	MVC IOAREA,TITRPAG2		
				201	PUT FICHEGEN,IOAREA		
				206 *			
				207 *	PREPARATION ET IMPRESSION LIGNE DE SOULIGNEMENT		
00230	9260 32C5		002C5	208	MVI IOAREA+1,C'-'		
00234	0206 32C6 32C5	002C6	002C5	209	MVC IOAREA+?(7),IOAREA+1		
0023A	9260 32CF		002CF	210	MVI IOAREA+10,C'-'		

LOC	OBJCT	CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT
00023E	0204	32CF	32CF	002CF	002CE	211	MVC IOAREA+11(5), IOARFA+10
000244	9260	32D5		002D5		212	MVI IOAREA+17,C'-'
000248	0272	32D6	32D5	002D6	002D5	213	MVC IOAREA+18(115), IOARFA+17
						214	PUT FICHEGEN, IOARFA
						219	*
000250	0283	32C5	32C4	002C5	002C4	220	MVC IOAREA+1(132), IOARFA
000262	07F9					221	RR R09
						222	*
						223	*
000264	0200	5001	5000	00001	00000	224	MVC11 MVC I(0,R05),0(R05)
00026A	0200	32D5	5000	002D5	00000	225	MVC12 MVC IOAREA+17(0),0(R05)
000270						226	SAVREG07 DS F
000274						227	SAVREG12 DS F
000278	00000037					228	NALIGPAB DC F'55'
00027C						229	SAVEARFA DS 18F
0002C4	4040404040404040					230	IOAREA DC CL133'
000349	000000						
00034C	00000000					231	ADRFINIT DC V(REINIT)
000350	C4C2C103D7D5E4D4					232	TITREGEN DC C'DR PALPNUM-FLEMENTS DES GENERATIONS'
000371	4040D5D6D440C47D					233	TITRPAG1 DC C' NOM D' NOMBRE'
000381	4040404040404040					234	DC 117C' '
0003F6	4040C9D5C4C5E74D					235	TITRPAG2 DC C' INDEX GENER'
000405	4040404040404040					236	DC 44C' '
000431	F5C5D9F2C9D6D54D					237	DC C'VeRSION DEFINITIVE DE LA PICTURE'
000451	4040404040404040					238	DC 42C' '
000021						239	LTITR EQU L'TITREGEN
						240	FICHEGEN DCB DDNAME=SYSOUT2,DSORG=PS,MACRF=(PM),RECFM=FBA,LRECL=133,RS=LKSI7E=1463
000000						294	R00 EQU 0
000001						295	R01 EQU 1
000002						296	R02 EQU 2
000003						297	R03 EQU 3
000004						298	R04 EQU 4
000005						299	R05 EQU 5
000006						300	R06 EQU 6
000007						301	R07 EQU 7
000008						302	R08 EQU 8
000009						303	R09 EQU 9
00000A						304	R10 EQU 10
00000B						305	R11 EQU 11
00000C						306	R12 EQU 12
00000D						307	R13 EQU 13
00000E						308	R14 EQU 14
00000F						309	R15 EQU 15
						310	END
0004F0	00000000					311	=V(SWITCHES)
0004F4	00000002					312	=F'2'
0004F8	00000009					313	=F'9'
0004FC	00000000					314	=V(ZONE0115)
0004F0	00000000					315	=V(ZONE1AMP)
0004F4	00000074					316	=F'116'
0004F8	00000348					317	=A(IOAREA+132)
0004FC	0000					318	=R'0'
0004FE	00FA					319	=H'250'
000500	000A					320	=H'10'

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
000502	F1			321	=C'1'

F01FEB73 5/17/76

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADRFINIT	00004	00034C	00231	0130
RCTR1	00002	00012A	00118	0113
CSECT3	00001	000000	00011	0017
FICHFGEM	00004	00047C	00244	0033 0044 0052 0098 0121 0139 0159 0179 0188 0195 0202 0215
IMPENTET	00004	00015A	00137	0058 0076 0093 0115
INARFA	00133	000204	00230	0035 0045 0053 0075 0080 0081 0082 0083 0092 0096 0099 0103 0103 0114 0122 0126 0126 0137 0140 0144 0146 0151 0160 0165 0165 0166 0171 0180 0184 0184 0189 0193 0196 0199 0199 0200 0203 0208 0209 0209 0210 0211 0211 0212 0213 0213 0216 0220 0220 0225 0317
L	00004	00008A	00065	0027
LTJTR	00001	000021	00239	0038 0042 0057
MVC01	00006	0000F6	00096	0091
MVC02	00006	000142	00126	0107
MVC11	00006	000264	00224	0050
MVC12	00006	00026A	00225	0119
NRLICPAG	00004	000278	00228	0013 0073 0090 0112
PIIT1	00004	00019F	00159	0152
PIIT2	00004	0001E0	00179	0172
R00	00001	000000	00294	0056 0118
R01	00001	000001	00295	
R02	00001	000002	00296	
R03	00001	000003	00297	0017 0018
R04	00001	000004	00298	0020 0022 0025 0026 0060 0065 0080 0082
R05	00001	000005	00299	0035 0041 0041 0042 0048 0056 0057 0057 0066 0079 0096 0104 0104 0224 0224 0225
R06	00001	000006	00300	0036 0036 0040 0067 0067 0086 0106 0108
R07	00001	000007	00301	0024 0037 0039 0041 0068 0079 0084 0105 0108 0118 0119
R08	00001	000008	00302	0038 0039 0049 0050
R09	00001	000009	00303	0058 0076 0093 0115 0221
R10	00001	00000A	00304	0146 0149 0150 0150 0151 0166 0169 0170 0170 0171
R11	00001	00000B	00305	0147 0147 0148 0149 0153 0153 0154 0156 0167 0167 0168 0169 0173 0173 0174 0176
R12	00001	00000C	00306	0016 0069 0073 0077 0090 0094 0102 0102 0112 0116 0125 0125 0127 0131
R13	00001	00000D	00307	0016 0019 0020 0021 0022 0129 0129 0131
R14	00001	00000E	00308	0016 0130 0132
R15	00001	00000F	00309	0018 0131
SAVEAREA	00004	00027C	00229	0019 0021
SAVEG07	00004	00027D	00226	0024 0068
SAVEG12	00004	000274	00227	0059 0069 0127
SR	00002	000080	00079	0074
STC1	00004	00017A	00149	0155 0157
STC2	00004	00018C	00160	0175 0177
TITPAGEM	00033	000350	00232	0042 0239
TITPAGE1	00016	000371	00233	0193
TITPAGE2	00015	0003F6	00235	0200
TSAIPAG1	00004	000002	00090	0105
TSAIPAG2	00004	000116	00112	0085

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

STATISTICS SOURCE RECORDS (SYS1) = 209 SOURCE RECORDS (SYS1R) = 2704

OPTIONS IN EFFECT LIST, MODECK, LOAD, MORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT = 55

301 PRINTED LINES

EXTERNAL SYMBOL DICTIONARY

PAGE 1
10.02 5/17/76

SYMBOL	TYPE	ID	ADDR	LENGTH	LD	ID
CSFCT4	SD	01	000000	000424		
PLDNMB	LD		0001A0			01
TRTARLFA	LD		0001FF			01
RFINIT	FR	02				
SWITCHES	FR	03				
ZONF0115	FR	04				
ZONETAMP	FR	05				

```

1 * MODULE CSECT4, RESPONSABLE DE LA GENERATION DES DONNEES ALPHANUMERI-
2 * QUES ALEATOIRES ET DE LA CONSTITUTION DU FICHIER SEQUENTIEL FICHOUTP
3 * *****
4 * CONVENTION DE PROGRAMMATION (SUITE):
5 * -----
6 * - SWR: INITIALISE A 0; POSITIONNE A 1, IL INDIQUE QUE LES PREPARA-
7 * TIFS INITIAUX D'OUVERTURE DU FICHIER FICHOUTP ET DE GARNISSA-
8 * GE DE LA ZONE LIMITE NUMERIQUE SUPERIEURE POUR CARACTERES A
9 * (LIMSUPA) ONT ETE EXECUTES.
10 *
11 *
000000 12 CSECT4 CSECT
13 PRINT MOGEN
14 ENTRY OLDNUMR
15 ENTRY TRTABLEA
16 *
17 * ETABLISSEMENT DES LIENS D'ENTREE
000000 90EC 000C 0000C 18 STM R14,R12,12(R13)
000000 19 USING CSECT4,R03
000004 183F 20 LR R03,R15
000006 5000 3180 00180 21 ST R13,SAVEAREA+4
00000A 1840 22 LR R04,R13
00000C 4100 31AC 001AC 23 LA R13,SAVEAREA
000010 5004 0008 00008 24 ST R13,8(R04)
25 *
000014 5840 33F8 003F8 26 L R04,=V(SWITCHES)
000018 9101 4000 00000 27 TM 0(R04),B'00000001'
00001C 4710 3044 00044 28 BN L
29 *
30 * PREPARATIFS INITIAUX
00002A 1855 31 OPEN (FICHOUTP,(OUTPUT))
00002C 4350 31FE 001FE 37 SR R05,R05
000030 4F50 3190 00190 38 IC R05,TRTABLEA-1
000034 FB10 3196 3420 00196 00420 39 CVD R05,DW
00003A 0201 3084 3196 00084 00196 40 SP DW+6(2),=PL1'1'
000040 9601 4000 00000 41 MVC LIMSUPA,DW+6
42 DI 0(R04),B'00000001'
43 *
000044 5840 33FC 003FC 44 L R04,=V(ZONMFO115)
000048 5850 3400 00400 45 L R05,=V(ZONNETAMP)
00004C 4160 3292 00292 46 LA R06,IOAREA+16
000050 1875 47 LR R07,R05
000052 4177 017F 0017F 48 LA R07,382(R07)
000056 F275 3190 4009 00190 00009 49 PACK DW,9(6,R04)
00005C 4FC0 3190 00190 50 CVR R12,DW
51 * BALAYAGE PICTURE (FIGURANT DANS ZONETAMP) CARACTERE PAR CARACTERE
000060 95C1 5000 00000 52 TESTCAR CLI 0(R05),C'A'
000064 4780 3080 00080 53 BE APIC
000068 9505 5000 00000 54 CLI 0(R05),C'M'
00006C 4780 309C 0009C 55 BE NPIC
000070 9570 5000 00000 56 CLI 0(R05),C' '
000074 4780 3088 00088 57 BE QUINTPIC
000078 9540 5000 00000 58 CLI 0(R05),C' '
00007C 4780 30F8 000F8 59 BE FINPIC
60 *

```

LOC	OBJECT	CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
					61 *	BRANCHEMENT GENER CORRESPONDANT A LA RENCONTRE D'UN CARACTERE A
000080	4590	3126		00126	62	APICT BAL R09,GENER
000084					63	LIMSUPA DS PL2
					64 *	
000086	5AF0	3404		00404	65	A R14,=A(TRTABLEA)
					66 *	
					67 *	TRANSFERT CARACTERE "GENERE" DANS IOAREA (ZONE D'E/S DE
					68 *	FICHOUTP)
00008A	D200	6000	E000	00000	69	MVC O(1,R0%),O(R14)
					70 *	
000090	4155	0001		00001	71	LA R05,1(R05)
000094	4166	0001		00001	72	LA R06,1(R06)
000098	47F0	3060		00060	73	B TESTCAR
					74 *	
					75 *	BRANCHEMENT GENER CORRESPONDANT A LA RENCONTRE D'UN CARACTERE N
00009C	4590	3126		00126	76	NP ICT BAL R09,GENER
0000A0	009C				77	DC PL2'9'
					78 *	
0000A2	5AF0	3408		00408	79	A R14,=A(TRTABLEN)
					80 *	
					81 *	TRANSFERT CARACTERE "GENERE" DANS IOAREA
0000A6	D200	6000	E000	00000	82	MVC O(1,R06),O(R14)
					83 *	
0000AC	4155	0001		00001	84	LA R05,1(R05)
0000B0	4166	0001		00001	85	LA R06,1(R06)
0000B4	47F0	3060		00060	86	B TESTCAR
					87 *	TRAITEMENT CORRESPONDANT A LA RENCONTRE D'UN QUOTE, DEBUT DE CSTE
					88 *	LITTERALE
0000B8	4155	0001		00001	89	QUOTPICT LA R05,1(R05)
0000BC	957D	5000	00000		90	CLI O(R05),C''''
0000C0	4780	30D2		000D2	91	BE TESTSUIV
					92 *	
					93 *	TRANSFERT DE LA CSTE LITTERALE, CARACTERE PAR CARACTERE, DANS
					94 *	IOAREA
0000C4	D200	6000	5000	00000	95	MVC MVC O(1,R06),O(R05)
					96 *	
0000CA	4166	0001		00001	97	LA R06,1(R06)
0000CE	47F0	30B8		000B8	98	B QUOTPICT
0000D2	4155	0001		00001	99	TESTSUIV LA R05,1(R05)
0000D6	1957				100	CR R05,R07
0000D8	4780	30E8		000F8	101	BE FINPICT
0000DC	957D	5000	00000		102	CLI O(R05),C''''
0000E0	4780	30C4		000C4	103	BE MVC
0000E4	47F0	3060		00060	104	B TESTCAR
					105 *	TRAITEMENT CORRESPONDANT A LA DETECTION DE FIN DE PICTURE
0000E8	5B60	340C		0040C	106	FINPICT S R06,=A(IOAREA+16)
0000EC	4060	3290		00290	107	STH R06,IOAREA+14
0000F0	4166	0010		00010	108	LA R06,16(R06)
0000F4	4060	3282		00282	109	STH R06,IOAREA
0000F8	D207	3286	4001	00286	110	MVC IOAREA+4(8),1(R04)
					111 *	
					112 *	ECRITURE D'UN ENREGISTREMENT SUR FICHOUTP
					113	PUT FICHOUTP,IOAREA
					118 *	
00010C	5850	3400		00400	119	L R05,=V(ZONETAMP)

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
000110	4160 3292		00292	120	LA R06,IOAREA+16
000114	46C0 3060		00060	121	RCT R12,TESTCAP
				122 *	LIFMS DE SORTIE RETOUR EN REINIT, DE CSECT1
000118	5800 0004		00004	123	L R13,4(R13)
00011C	58F0 3394		00394	124	L R14,ADREINIT
000120	98FC 0010		00010	125	LM R15,R12,16(R13)
000124	07FF			126	RR R14
				127 *	ROUTINE DE GENERATION DE NB ALFATOIRES - INSPIREE DU "RANDOM NUMBER
				128 *	GENERATOR", MODULE TDG00, STATEMENTS TDG0600 A TDG0801, DE TEST DA-
				129 *	TA GENERATOR
000126	98AB 3198		00198	130	GENER LM R10,R11,COUNTRS
00012A	F873 3198	31A0 00198	001A0	131	ZAP COUNTRS,OLDNUMR
000130	4FF0 3198		00198	132	CVR R15,COUNTRS
000134	4AA0 341C		0041C	133	AH R10,=H'1'
000138	4780 3140		00140	134	R7 M
00013C	48B0 341C		0041C	135	SH R11,=H'1'
000140	5CF0 3410		00410	136 M	M R14,=F'8195'
000144	5AF0 3414		00414	137	A R15,=F'162'
000148	17AB			138	XR R10,R11
00014A	17FA			139	XR R15,R10
00014C	54F0 3418		00418	140	N R15,=X'7FFFFFFF'
000150	4FF0 3198		00198	141	CVD R15,COUNTRS
000154	0203 31A0	319C 001A0	0019C	142	MVC OLDNUMR,COUNTRS+4
00015A	F831 31A5	9000 001A5	00000	143	ZAP ZONECALC+1(4),0(2,R09)
000160	0100 319D	319F 0019D	0019F	144	MVM COUNTRS+5(1),COUNTRS+7
000166	FC31 31A5	319C 001A5	0019C	145	MP ZONECALC+1(4),COUNTRS+4(2)
00016C	FA31 31A5	341F 001A5	0041F	146	AP ZONECALC+1(4),=PL2'500'
000172	90AB 3198		00198	147	STM R10,R11,COUNTRS
000176	9200 31A4		001A4	148	MVI ZONECALC,X'00'
00017A	F042 31A4	3421 001A4	00421	149	DP ZONECALC(5),=PL3'1000'
000180	F871 3190	31A4 00190	001A4	150	ZAP DW,ZONECALC(2)
000186	4FF0 3190		00190	151	CVR R14,DW
00018A	4199 0002		00002	152	LA R09,2(R09)
00018E	07F9			153	RR R09
				154 *	
				155 *	
000190				156	DW DS R
000198	12AB9793FA34784C			157	COUNTRS DC X'12AB9793FA34784C'
0001A0	0184095C			158	OLDNUMR DC PL4'184095'
0001A4	00			159	ZONECALC DC X'00'
0001A5				160	DS PL4
0001AC				161	SAVEAREA DS 18F
0001E4	F0F1E2E3E4E5E6E7			162	TRTABLEM DC C'0123456789'
0001EE	1A			163	DC X'1A'
0001FF	C1C2C3C4C5C6C7C8			164	TRTABLEA DC C'ABCDEFGHIJKLMNQPORSTUVWXYZ'
000219				165	DS 104X
000282				166	IOAREA DS H
000284	0000			167	DC H'0'
000286				168	DS YLR
00028E	4040			169	DC CL2'
000290				170	DS P
000292				171	DS XL256
000392	0000				
000394	00000000			172	ADREINIT DC V(ADREINIT)
				173	FICHOUTP DCA DDNAME=OUTPUT,DSORG=PS,MACRF=(PM),RECFM=VBS,LRECL=256,RL#

LOC OBJECT CODE ADDR1 ADDR2 STMT SOURCE STATEMENT

F01FEB73 5/17/76

```

                                KSIZE=264
000000          227 R00      EQU 0
000001          228 R01      EQU 1
000002          229 R02      EQU 2
000003          230 R03      EQU 3
000004          231 R04      EQU 4
000005          232 R05      EQU 5
000006          233 R06      EQU 6
000007          234 R07      EQU 7
000008          235 R08      EQU 8
000009          236 R09      EQU 9
00000A          237 R10      EQU 10
00000B          238 R11      EQU 11
00000C          239 R12      EQU 12
00000D          240 R13      EQU 13
00000E          241 R14      EQU 14
00000F          242 R15      EQU 15
                                END
0003F8 00000000          244      =V(SWITCHES)
0003FC 00000000          245      =V(ZONE0115)
000400 00000000          246      =V(ZONETAMP)
000404 000001FF          247      =A(TRTABLFA)
000408 000001F4          248      =A(TRTABLFM)
00040C 00000292          249      =A(IQAREA+16)
000410 00002003          250      =F'8195'
000414 000000A2          251      =F'162'
000418 7FFFFFFF          252      =X'7FFFFFFF'
00041C 0001              253      =H'1'
00041F 500C              254      =PL2'500'
000420 1C                255      =PL1'1'
000421 01000C           256      =PL3'1000'

```

SYMBOL	LEN	VALUE	DEFN	REFERENCES
ADRFINIT	00004	000394	00172	0124
APICT	00004	000080	00062	0053
COUNTRS	00008	000198	00157	0130 0131 0132 0141 0142 0144 0144 0145 0147
CSECT4	00001	000000	00012	0019
DW	00008	000190	00156	0039 0040 0041 0049 0050 0150 0151
FICHOUTP	00004	000398	00177	0035 0114
FINPICT	00004	0000E8	00106	0059 0101
GENFR	00004	000126	00130	0062 0076
INAREA	00002	000282	00166	0046 0106 0107 0109 0110 0115 0120 0249
L	00004	000044	00044	0028
LIMSUPA	00002	000084	00063	0041
M	00004	000140	00136	0134
MVC	00006	0000C4	00095	0103
MPICT	00004	00009C	00076	0055
OLDNUMB	00004	0001A0	00158	0014 0131 0142
QUOTPICT	00004	000088	00089	0057 0098
R00	00001	000000	00227	
R01	00001	000001	00228	
R02	00001	000002	00229	
R03	00001	000003	00230	0019 0020
R04	00001	000004	00231	0022 0024 0026 0027 0042 0044 0049 0110
R05	00001	000005	00232	0037 0037 0038 0039 0045 0047 0052 0054 0056 0058 0071 0071 0084 0084 0089 0089 0090 0095 0099 0099 0100 0102 0119
R06	00001	000006	00233	0046 0069 0072 0072 0082 0085 0085 0095 0097 0097 0106 0107 0108 0108 0109 0120
R07	00001	000007	00234	0047 0048 0048 0100
R08	00001	000008	00235	
R09	00001	000009	00236	0062 0076 0143 0152 0152 0153
R10	00001	00000A	00237	0130 0133 0138 0139 0147
R11	00001	00000B	00238	0130 0135 0138 0147
R12	00001	00000C	00239	0018 0050 0121 0125
R13	00001	00000D	00240	0018 0021 0022 0023 0024 0123 0123 0125
R14	00001	00000E	00241	0018 0065 0069 0079 0082 0124 0126 0136 0151
R15	00001	00000F	00242	0020 0125 0132 0137 0139 0140 0141
SAVEAREA	00004	0001AC	00161	0021 0023
TESTAR	00004	000060	00052	0073 0086 0104 0121
TESTSUIV	00004	000002	00099	0091
TRTABLEA	00026	0001FF	00164	0015 0038 0065 0247
TRTABLEM	00010	0001F4	00162	0079 0248
ZONECALC	00001	0001A4	00159	0143 0145 0146 0148 0149 0150

NO STATEMENTS FLAGGED IN THIS ASSEMBLY

STATISTICS SOURCE RECORDS (SYSIN) = 182 SOURCE RECORDS (SYSLIB) = 2704

OPTIONS IN EFFECT LIST, NOECHK, LOAD, NORENT, XREF, NOTEST, ALGN, OS, NOTERM, LINECNT = 55

270 PRINTED LINES

CONCLUSION

La troisième partie de ce travail aurait dû être l'occasion pour nous de développer un " programme de gestion de tests de programmes d'application " (l'équivalent de ce que nous avons appelé " programme de contrôle d'essais " reprenant les fonctions de " simulation " en environnement " batch " du contrôle du déroulement normal des programmes d'application TP.

Si nous nous sommes, en définitive, écarté de cet objectif premier, c'est pour diverses raisons dont la moindre n'est certes pas que les produits existants relatifs à ce sujet, que nous avons décrits dans la deuxième partie, donnent entière satisfaction à l'heure actuelle et qu'il ne nous semblait donc pas opportun d'en développer un nouveau. En outre, chacun des simulateurs que nous avons présentés est intimement lié à un système plus global dans le cadre duquel il s'insère (en l'occurrence IMS ou CICS), si bien que la tâche de développement d'un nouveau simulateur s'annonçait réellement fort ardue, pour ne pas dire " impossible ", compte tenu de la limite imposée quant au délai de réalisation.

Suivant les conseils de personnes avisées, nous avons finalement opté pour une solution intermédiaire, consistant (nous l'avons suffisamment expliqué) en l'amélioration de l'outil DBPROTOTYPE, dont les lacunes étaient d'autant plus durement ressenties qu'il est très usité. Certes il s'agissait là d'un travail à plus petite échelle que le précédent (nous nous sommes d'ailleurs attaché à décrire le plus complètement possible ses interactions avec l' " existant") mais, après coup, nous avons l'impression qu'il fut aussi enrichissant que ne l'aurait été le premier (que nous n'aurions peut-être pas achevé, alors que celui-ci est prêt à fonctionner). Il s'avère en tout cas profitable, car il évitera désormais, un chargement manuel fastidieux de données alphanumériques dans toute base de test. Nous avons également voulu qu'il soit simple à l'usage (en dépit de quelques conventions que nous avons dû fixer, nous croyons y être parvenu), tout en accordant aux utilisateurs diverses options (témoin celles concernant le choix entre un décodage seul ou accompagné de génération, ou celles relatives au paramétrage de la table de translation des caractères A), et qu'il restitue à ces derniers des informations aussi claires et aussi concises que possible (témoin la présentation des listings de sortie).

En ce qui concerne plus spécifiquement les techniques de test de programmes d'application présentées tout au long de la seconde partie, nous retiendrons qu'elles " gravitent " quasi toutes autour du mot "simulation"; ce n'est toutefois pas frappant à proprement parler, puisque le fait de ramener les tests de programmes d'application " on line " dans un environnement " batch " (et nous avons vu que c'était nécessaire) entraîne automatiquement le recours à des procédures de simulation, dont les principales " retracent " les processus d'entrée et de sortie de messages en provenance ou à destination du réseau de terminaux, tout en contrôlant l'exécution des programmes d'application (ces derniers étant à un stade d'élaboration plus ou moins avancé) dans un contexte autre que leur contexte futur, pour veiller, en définitive à ce que le déroulement de ceux-ci ne soit pas affecté par le fait que l'on se situe en période de tests ou en période d'exploitation " opérationnelle " du système.

BIBLIOGRAPHIE

- [1] Utilisation et programmation des ordinateurs en temps réel - J. Martir
(Prentice Hall INC., 1965 - Traduction: Les Editions d'Organisation,
1969)
- [2] Design of on-line computer systems - E. Yourdon (Prentice Hall INC.
1972)
- [3] Queue management in a control program for a real-time system - J.P.
Windal (IBM Belgium Functional Systems Group, 1970)
- [4] Introduction au télétraitement - J.P. Windal (1974)
- [5] DBPROTOTYPE/VS
Program Description / Operations Manual (SH20-1391)
- [6] Information Management System / Virtual Storage (IMS / VS)
General Information Manual (GH20-1260)
- [7] Information Management System / Virtual Storage (IMS/VS)
System / Application Design Guide (SH20-9025)
- [8] Information Management System / Virtual Storage (IMS/VS)
Message Format Service
User's Guide (SH20-9053)
- [9] Customer Information Control System / Virtual Storage (CICS/VS)
General Information Manual (GH20-1280)
- [10] BTS Batch Terminal Simulator
Program Description / Operations Manual (SH20-1306)
- [11] 3270 Formatting Feature for Batch Terminal Simulator (BTS)
Program Description / Operations Manual (SH20-1360 complété par
SN20-3409)
- [12] CICS 3270 Simulator (SB21-1036)
- [13] Customer Information Control System / Virtual Storage (CICS/VS)
Installation Guide (DOS) (SH20-9051)
- [14] S/370 CICS On-Line Test / Debug
Program Description / Operations Manual (SH20-1358)
- [15] Test Data Generator
Program Description / Operations Manual (SH20-1361)
- [16] Test Data Generator
Systems Guide (LY20-0932)
- [17] IBM System / 370
Principles of operation (GA22-7000)

- [18] OS / VS Data Management
Services Guide (GC26-3783)
 - [19] OS / VS Data Management
Macro Instructions (GC26-3793)
 - [20] OS / VS2
Supervisor Services and Macro Instructions (GC28-0683)
-