79 | 750

UNIVERSITE DE NANCY I
U.E.R. DE SCIENCES MATHÉMATIQUES

CENTRE DE RECHERCHE EN
INFORMATIQUE DE NANCY

Sc N 79/43 A

CONTRIBUTION A L'ÉTUDE D'UN PROCESSEUR DE CONSULTATION DE DONNÉES

LES COMMUNICATIONS ET L'ÉVALUATION

THÈSE

PRÉSENTÉE POUR L'OBTENTION DU
DOCTORAT DE SPÉCIALITÉ EN INFORMATIQUE

SOUTENUE LE **12 JUIN 1979**

PAR **FRANÇOIS SCHWAAB**

JURY :

PRÉSIDENT C. PAIR
EXAMINATEURS J.C. DERNIAME
M. GRIFFITHS
J.P. HATON
J. LEROUDIER
D. TUSERA

BIBLIOTHEQUE SCIENCES NANCY 1
D 095 180347 7

UNIVERSITE DE NANCY I
U.E.R. DE SCIENCES MATHÉMATIQUES

CENTRE DE RECHERCHE EN
INFORMATIQUE DE NANCY

UNIVERSITÉ DE NANCY
BIBLIOTHÈQUE

CONTRIBUTION A L'ÉTUDE D'UN PROCESSEUR DE CONSULTATION DE DONNÉES

LES COMMUNICATIONS ET L'ÉVALUATION

THÈSE

PRÉSENTÉE POUR L'OBTENTION DU
DOCTORAT DE SPÉCIALITÉ EN INFORMATIQUE

SOUTENUE LE 12 JUIN 1979

PAR FRANÇOIS SCHWAAB

JURY :

PRÉSIDENT C. PAIR
EXAMINATEURS J.C. DERNIAME
M. GRIFFITHS
J.P. HATON
J. LEROUDIER
D. TUSERA

Dans ces quelques lignes je tiens à remercier,

Monsieur le Professeur Claude PAIR, Président de l'Institut National Polytechnique de Lorraine, de me faire l'honneur de présider le jury de cette thèse; je n'oublie pas qu'il est à l'origine de mon intérêt pour l'architecture des machines et la microprogrammation;

Monsieur le Professeur Michael GRIFFITHS pour les appuis et les encouragements qu'il m'a toujours prodigués en dirigeant ma thèse;

Messieurs les Professeurs Jean-Claude DERNIAME et Jean-Paul HATON qui se sont intéressés à mon travail et qui ont accepté de le juger;

Jacques LEROUDIER pour son aide et ses conseils précieux; il a éveillé mon intérêt pour l'évaluation des performances et depuis de nombreuses années son esprit critique me guide et me stimule dans mes travaux;

Dimitri TUSERA, responsable scientifique du projet Système de Consultation de Données, qui m'a accueilli dans son équipe et m'a encouragé à y développer mes idées personnelles; l'esprit amical qui règne dans le groupe qu'il anime m'a permis de réaliser mes travaux dans un environnement idéal.

Je n'oublie pas mes camarades de l'IRIA et de NANCY qui, avec gentillesse, m'ont fait bénéficier de leur compétence, en particulier, Marc BADEL, Jean-Jacques BAECHELEN, Marion CREHANGE, Ali El MASRI, Juan FLORES, Bruno GAUDEUL, Geneviève JOMIER, Didier MERLE, Dominique POTIER, Jean ROHMER, Eunika SUPIK, et tous mes amis du CRIN.

Je tiens à ajouter que ce mémoire n'aurait pas ce bel aspect si Christiane FLOC'H n'y avait consacré tout son talent et son bon goût.

François SCHWAAB

AVANT - PROPOS

Les travaux présentés dans cette thèse ont été réalisés dans le cadre du projet Systèmes de Consultations de Données au Laboratoire de Recherche en Informatique et Automatique (LABORIA) à l'Institut de Recherche en Informatique et Automatique (IRIA).

Ce projet a pour but de développer une nouvelle architecture de machine orientée vers la consultation de données textuelles et un prototype dénommé TREFLE est en cours de réalisation.

Ma collaboration à l'équipe a plus particulièrement porté sur les problèmes de communication et d'évaluation de performances :

- la mise en oeuvre d'un mécanisme de liaison entre le contrôleur, superviseur du processeur TREFLE, et un mini-ordinateur hôte, assurant la gestion des utilisateurs, a constitué ma première tâche au sein de l'équipe. Elle fut suivie par l'étude et la mise en oeuvre des algorithmes gérant les données échangées entre le disque et le processeur TREFLE ;

- l'évaluation des performances est depuis mes débuts de chercheur mon domaine de prédilection. En effet après avoir collaboré à deux campagnes de mesures faites à l'Institut Universitaire de Calcul Automatique de Lorraine (IUCAL), j'étais en quête d'une application me permettant de développer mes idées personnelles sur les outils de mesure. Sous cet aspect, ma participation au projet TREFLE a consisté à modéliser le fonctionnement de l'architecture du processeur et à étudier un outil de mesure adapté au problème posé. La modélisation a permis de dégager les paramètres nécessitant une mesure dans le but d'une meilleure connaissance de l'utilisation faite du matériel constituant la machine, et la conception et la réalisation de l'outil de mesure m'ont permis de montrer l'adéquation de mes idées aux problèmes posés par l'évaluation de cette architecture particulière.

SOMMAIRE

INTRODUCTION

CHAPITRE 1 : LA MACHINE TREFLE
- Introduction
- I - Le langage de requêtes
- II - L'architecture du processeur TREFLE
- Conclusion : les applications

CHAPITRE 2 : LES COMMUNICATIONS
- Introduction
- I - La liaison entre TREFLE et le minicalculateur hôte
- II - La liaison entre TREFLE et le disque
- Conclusion

CHAPITRE 3 : L'EVALUATION
- Introduction
- I - L'évaluation des performances des systèmes informatiques
- II - L'évaluation des performances de la machine TREFLE
- Conclusion

CONCLUSION

BIBLIOGRAPHIE

ANNEXES
- A - Les entrées/sorties
- B - L'IOP
- C - La connexion
- D - Le disque
- E - Le formatteur
- F - Le modèle

TABLE DES MATIERES

BIBLIOTHÈQUE NANCY

INTRODUCTION

Le domaine de la conception et de la réalisation de nouvelles architectures de machines informatiques subit actuellement de profondes transformations. En effet, les contraintes techniques qui régissent ce secteur de la recherche en informatique ont été bouleversées par l'apparition récente du phénomène microprocesseur. On peut présenter ces contraintes de la manière suivante :

- les progrès de la technologie permettent une croissance exponentielle de la puissance de calcul. Depuis 1970, année de l'apparition des premiers circuits à grande échelle d'intégration, jusqu'à aujourd'hui, les performances offertes par les microprocesseurs n'ont cessé de croître en doublant tous les ans ;

- le degré de sophistication atteint aujourd'hui par les machines universelles laisse espérer peu de progrès en ce qui concerne leur architecture ; par contre, certaines applications spécialisées nécessitant une grande puissance de traitement non numérique ouvrent de nouvelles voies de recherche.

En fonction de ces contraintes l'équipe du projet Systèmes de Consultation de Données a choisi de développer une architecture de machine spécialisée dans la consultation des données textuelles, par exemple les articles de journaux, les images de télévision, etc... Ces données, caractérisées par leur codage aisément compréhensible par l'homme et par leur diversité de formes, constituent l'essentiel des informations avec lesquelles nous cohabitons tous les jours.

Contrairement aux systèmes logiciels de documentation tel MISTRAL sur IRIS 80, qui imposent des contraintes rigides sur la structure des don-

nées, la forme et la nature des questions, la machine développée par l'équipe se fixe les objectifs suivants :

- offrir une certaine tolérance aux fautes d'orthographe dans les mots clés des questions,
- accepter des références incomplètes dans les questions posées, ce qui revient à y introduire la notion de flou,
- répondre à l'utilisateur de manière nuancée pour lui permettre de faire un premier tri sur les informations restituées,
- prendre en considération la structure naturelle des textes traités qui se décomposent en mots, phrases, paragraphes, articles, etc...

En relation avec ces objectifs, la machine a été dénommée TREFLE car dans le langage des fleurs le TREFLE symbolise l'incertitude, qualité bien peu informatique jusqu'à présent.

Fonctionnellement, le prototype développé aura les caractéristiques fondamentales suivantes :

- il sera capable de scruter les données à la cadence de leur lecture sur le disque, c'est-à-dire environ 1,2 millions de caractères par seconde. Remarquons que cette performance est hors de portée des machines universelles les plus puissantes de nos jours...

- grâce à sa grande vitesse de traitement, la recherche des informations répondant à la question pourra se faire par une lecture exhaustive de tout l'espace de données : cette solution garantit l'objectivité de la recherche et permet de supprimer l'indexation par mots-clés des articles ;

- de plus, la possibilité de scrutation du flot de données en une seule lecture autorisera l'abandon de la notion de mémoire de données qui pourra être remplacée par celle d'émetteur de données, par exemple un émetteur de télévision...

La conception et la réalisation du processeur TREFLE fait l'objet d'un travail d'équipe, aussi nous ne présentons dans ce mémoire que les points pour lesquels l'auteur a apporté sa contribution.

Le chapitre I est consacré à la présentation du projet.

Partant de l'aspect externe, c'est-à-dire du langage de questions, nous décrivons le prototype et son application en cours de développement. Après avoir introduit les idées directives et leurs implications nous détaillons l'architecture de la réalisation en insistant sur les problèmes posés et les solutions adoptées.

La communication entre le processeur TREFLE et son environnement est abordée au chapitre II.

Dans une première partie nous présentons le mécanisme de dialogue entre le contrôleur, processeur supervisant le fonctionnement de la machine, et le mini-ordinateur MITRA 125. Cette connexion permet d'une part de décharger le processeur TREFLE de la gestion des utilisateurs et d'autre part de faciliter l'inscription des informations à consulter sur le support de données.

Dans une seconde partie nous décrivons la solution retenue pour relier le disque, support des données, au processeur de scrutation, opérateur de la machine TREFLE. Cette liaison associe à la structure de données utilisée sur le support les algorithmes des fonctions chargées de gérer cette structure.

La troisième partie présente succinctement les problèmes posés par la mise en oeuvre des mécanismes de communication et énumère les réali-

sations nécessaires au développement des procédures logicielles et micrologicielles utilisées.

Le chapitre III développe un autre aspect de la collaboration de l'auteur du projet.

Visant l'implantation de l'architecture de la machine sous la forme d'un circuit à très grande échelle d'intégration, il est indispensable que nous connaissions parfaitement la manière dont est utilisée la structure matérielle construite.

Après une introduction générale aux problèmes de l'évaluation des performances, nous présentons les étapes de la démarche que nous avons suivie pour évaluer le projet TREFLE.

La modélisation nous a permis d'appréhender la complexité de la machine et de dégager les grandeurs à évaluer par mesures.

L'observation directe du prototype pose des problèmes délicats à résoudre. En effet, la vitesse de traitement, la masse considérable de données traitées et la répartition sur plusieurs processeurs des fonctions de la machine nécessitent le développement d'une nouvelle technique d'observation utilisant la microprogrammation.

L'utilisation prospective du modèle de simulation construit a aidé les membres de l'équipe à estimer certains paramètres de l'architecture.

CHAPITRE 1 :

La Machine TREFLE

INTRODUCTION

I - LE LANGAGE DE REQUETES

1.1. Les phrases de description
1.2. Les phrases d'interrogation
1.3. Les phrases de notation

II - L'ARCHITECTURE DU PROCESSEUR TREFLE

2.1. Les principes fonctionnels
2.2. Le processeur TREFLE

2.2.1. Les filtres
2.2.2. Le module de calcul
2.2.3. Le module de classement
2.2.4. Le disque
2.2.5. Le formatteur
2.2.6. Le contrôleur
2.2.7. Le mini-ordinateur hôte

CONCLUSION : LES APPLICATIONS

INTRODUCTION

L'introduction des mémoires de masse à accès rapide dans la gamme des périphériques standard des ordinateurs, les facilités offertes par les machines pour le traitement des caractères et le développement de systèmes à accès multiples ont conduit à la réalisation de grands systèmes de documentation logiciels comme MISTRAL [Mis 71], SYNTOL [Mar 62], SYDONI [Mig 70] et SABIR-C [IGR 71]. L'arrivée des circuits à grande échelle d'intégration, offrant d'énormes possibilités de traitement à des coûts très bas et de nombreuses facilités de mise en oeuvre, permet de compléter cette approche en favorisant la construction de machines orientées vers la consultation de bases de données [COM 79]. De nombreux exemples de réalisations actuelles confirment le développement de cette tendance :

- les projets CASSM [Cop 73], INDY [Cop 78] et RAP [Ozs 76] proposent des architectures multicellulaires construites autour de mémoires circulantes et de microprocesseurs. Ces machines ont été spécialement conçues pour la gestion de bases de données relationnelles au sens défini par CODD [Cod 70] ;

- dans le contexte de la recherche documentaire, le projet MICROTEXT [Bum 72] a consisté à microprogrammer un interprète d'automates à états finis qui recherchait directement dans les documents les mots-clés des questions. Ce projet a démontré qu'un automate unique pour la recherche et la résolution des questions était démesuré pour un système opérationnel ;

- plus récemment, la CIA (Central Intelligence Agency) développe une architecture spécialisée dans la recherche textuelle [Rob 78]. Cette machine reprend le principe des automates à états finis pour la recherche, laissant la résolution des questions à un processeur spécialisé.

Le projet TREFLE, dont nous décrivons ici certains aspects, s'inscrit dans cette voie de recherche tout en présentant de nombreux traits originaux. En effet, la recherche documentaire aborde un domaine où l'information s'éloigne de l'ordinateur pour se rapprocher du monde humain, où la rigueur cède le pas à l'incertitude et la logique simple du vrai ou faux à une logique plus nuancée [MaR 78].

Dans ce premier chapitre nous présentons TREFLE sous son aspect externe : le langage de requêtes, et sous son aspect interne : l'architecture développée autour d'une application de recherche bibliographique.

I - LE LANGAGE DE REQUETES

Dans la plupart des grands systèmes documentaires actuels le fichier contenant les enregistrements décrivant les documents - les descripteurs - est indexé sur un certain nombre de champs. Cette indexation permet l'introduction pour chaque champ d'un fichier inverse constitué par la liste des mots-clés du champ, chaque mot-clé pointant sur l'ensemble des descripteurs dans lesquels il figure. Les interrogations directes de ces systèmes se font par l'intermédiaire de ces fichiers inverses et les requêtes permettent en général les combinaisons booléennes entre mots-clés. L'exemple présenté ci-dessous s'inspire directement du système MISTRAL.

L'utilisateur désire consulter une base documentaire ornithologique et en extraire les articles concernant les oiseaux d'eau visibles en France ou en Suisse, publiés par Buffon ou Géroudet. Sa requête sera formulée de la manière suivante :

/Mots-clés/ (OISEAUX D'EAU ou PALMIPEDES ou ECHASSIERS ou
AVIFAUNE PALUSTRE ou AVIFAUNE MARINE) et
(FRANCE ou SUISSE)
/Auteur/ BUFFON ou GEROUDET

Dans ces systèmes tous les champs ne sont pas toujours indexés et plus la complexité de l'expression booléenne est grande, plus le processus de réponse est long et coûteux. Ces limitations ne permettent pas la réponse directe pour une question ayant plusieurs opérateurs booléens comme celle présentée en exemple. De plus, la syntaxe des requêtes est en général très rigide et ce type de système n'admet que rarement des questions floues : il faut donner l'orthographe exacte des auteurs et employer les mêmes mots-clés que ceux définis dans les lexiques du système. Ces derniers inconvénients sont minimisés par l'utilisation de lexiques structurés (relation d'inclusion, synonymie, etc...).

La mise en oeuvre d'une base documentaire nécessite un travail d'indexation, long et fastidieux, qui introduit la subjectivité du documentaliste dans le choix des mots-clés. Cette étape, bien que quelquefois automatisée, alourdit considérablement l'élaboration d'un système documentaire.

Le projet qui est en cours de réalisation autour du processeur TREFLE diffère des systèmes traditionnels sur quatre points :

- la non indexation des champs :
  le principe de recherche appliqué par TREFLE ne nécessite aucune indexation appauvrissant l'information et chaque mot du document peut servir de mot-clé pour la recherche ;

- la complexité de la question n'influe pas sur le temps de son traitement, mais uniquement sur la taille de la mémoire nécessaire à son stockage sous une forme arborescente. Cette possibilité nous permet d'envisager le traitement de questions très complexes à un coût raisonnable ;

- le système tolère les fautes d'orthographe dans les mots utilisés dans les requêtes, ce qui introduit la notion de flou dans les questions. Cette tolérance n'est utilisée que sur demande de l'usager, elle peut

être d'une grande utilité dans certains cas tels la manipulation de noms propres, la recherche de mots contenant des erreurs typographiques dans la base, etc...

- les questions permettent la prise en compte de la structure naturelle des documents traités. Ainsi l'utilisateur peut préciser le sens de sa requête en s'appuyant sur les délimiteurs naturels ayant une grande importance logique dans les documents : par exemple, trouver les documents ayant dans la même phrase la liste des mots suivants : RAPACE, DIURNE.

Pour mettre en oeuvre les principes que nous venons d'énoncer nous définissons le langage de dialogue avec l'utilisateur et l'architecture de la machine qui l'interprète. Ce langage assurera trois fonctions principales :

- la fonction de description qui permet de préciser la structure naturelle des documents de la base ;

- la fonction d'interrogation qui permet d'exprimer les questions posées ;

- la fonction de notation qui permet d'attribuer des notes mesurant l'adéquation d'une réponse à une question floue.

1.1. LES PHRASES DE DESCRIPTION

Les phrases de description permettent à l'utilisateur de décrire sous une forme proche de la notation BNF [Bac 60] la structure des documents de la base sur laquelle il travaille. Une telle structure est en général engendrée par une grammaire dont chaque entité syntaxique non terminale a une grande importance logique et sémantique ; par exemple, dans un texte, le mot peut être défini comme une suite de lettres délimitée par des blancs ou des signes de ponctuation ; le mot est une entité syntaxique non terminale, les lettres, les blancs et les signes de ponctuation étant des caractères terminaux.

Dans les deux exemples qui suivent nous présentons des grammaires pouvant engendrer et reconnaître des structures de documents. Dans les questions l'utilisateur pourra référencer les entités syntaxiques non terminales en les désignant par leur identificateur déclaré dans la grammaire.

Exemple 1 :

```
BASE                ::= LISTE-ARTICLES
LISTE-ARTICLES      ::= ARTICLE | ARTICLE LISTE-ARTICLES
ARTICLE             ::= début article LISTE-PARAGRAPHES
                        fin article
LISTE-PARAGRAPHES   ::= PARAGRAPHE | PARAGRAPHE LISTE-PARAGRAPHES
PARAGRAPHE          ::= à la ligne LISTE-PHRASES
LISTE-PHRASES       ::= PHRASE | PHRASE LISTE-PHRASES
PHRASE              ::= LISTE-MOTS point
LISTE-MOTS          ::= MOT | MOT LISTE-MOTS
MOT                 ::= blanc LISTE-CARACTERES | signe de ponctu-
                        ation LISTE-CARACTERES
LISTE-CARACTERES    ::= caractère | caractère LISTE-CARACTERES
```

Exemple 2 :

```
BASE                ::= LISTE-DESCRIPTEURS
LISTE-DESCRIPTEURS  ::= DESCRIPTEUR | DESCRIPTEUR LISTE-DESCRIPTEURS
DESCRIPTEUR         ::= séparateur descripteur LISTE-CHAMPS
LISTE-CHAMPS        ::= CHAMP-AUTEUR séparateur champ CHAMP-TITRE
                        séparateur champ CHAMP-MOTS CLES
CHAMP-AUTEUR        ::= LISTE-MOTS
CHAMP-TITRE         ::= LISTE-MOTS
CHAMP-MOTS CLES     ::= LISTE-MOTS
LISTE-MOTS          ::= MOT | MOT LISTE-MOTS
MOT                 ::= blanc LISTE-CARACTERES
LISTE-CARACTERES    ::= caractère | caractère LISTE-CARACTERES
```

1.2. LES PHRASES D'INTERROGATION

Les phrases d'interrogation permettent à l'utilisateur de formuler ses questions à la base documentaire ; elles expriment les fonctions logiques ayant pour arguments les mots recherchés.

A l'aide d'exemples et en faisant abstraction de la syntaxe précise, nous présentons les possibilités offertes par le langage d'interrogation :

- Exemples d'opérateurs booléens susceptibles d'être utilisés :

+ Trouver les articles contenant : (PASSEREAUX ou PICS ou GALLINACES ou COLOMBINS mais pas CORVIDES) et (BELGIQUE ou SUISSE).

- Exemple d'utilisation du flou : l'utilisateur n'est pas assuré de la bonne orthographe de BONELLI, TROIL et BRUNNICH :

+ Trouver les articles contenant : (BONELLI [avec faute]) ou (TROIL [avec faute] et BRUNNICH [avec faute]).

- Exemple d'utilisation de la structure naturelle du texte en vue de l'augmentation de la précision de la requête : Les entités syntaxiques PHRASE et PARAGRAPHE ont été précédemment définies dans la grammaire décrivant le document. Dans ces phrases apparaît un nouvel opérateur utilisant l'analyse syntaxique.

+ Trouver les articles contenant : (PIE et GRIECHE et ECORCHEUR dans la même PHRASE) ou (PIE et BAVARDE dans le même PARAGRAPHE).

- Exemple de combinaison faisant appel sans les différencier à

plusieurs champs, le champ auteur et le champ texte : TREFLE ayant à sa charge de retrouver dans l'article les mots demandés, il n'est pas nécessaire de préciser les champs les contenant.

+ Trouver les articles contenant : (PALMIPEDES et GEROUDET) ou (LORENZ [avec faute] et CORVIDES).

1.3. LES PHRASES DE NOTATION

Les systèmes traditionnels ne restituent généralement que les articles répondant exactement à la question posée. Dans la machine TREFLE nous complétons ce mode de recherche par un mode introduisant le flou, dans lequel seuls les documents offrant la meilleure réponse à la requête sont sélectionnés. Ainsi chaque document recevra une note mesurant son adéquation à la question posée. Cette note prendra en compte plusieurs paramètres :

- il est normal qu'un document possédant un grand nombre d'occurrences des mots-clés recherchés soit mieux noté que celui n'en contenant que très peu ;

- la tolérance aux fautes d'orthographe introduit un classement rendant compte de l'éloignement entre la réponse et la question : par exemple si la question mentionne le nom de LORENZ [avec faute], les réponses contenant LORENZ, LORENTZ, LORENS etc... ne seront pas considérées comme égales, mais seront classées suivant leur éloignement par rapport au mot utilisé dans la question. La mise en oeuvre de ce nouveau mode de recherche nécessite la définition d'un troisième type de phrase permettant à l'utilisateur de spécifier les notes qu'il souhaite attribuer aux réponses. Reprenons l'exemple donné ci-dessus :

+ Trouver les articles contenant LORENZ [avec faute] et CORVIDES.
  Si LORENZ existe dans le document alors il aura la note 10.
  S'il existe un mot voisin de LORENZ à deux fautes d'orthographe près alors il aura la note 8.

- si l'utilisateur désire introduire des critères particuliers de sélection il pourra les préciser à l'aide des phrases de notation, par exemple : sélection sur la langue, sur la nuance du sens des mots, etc...

Pour éviter d'alourdir les questions, le compilateur proposera un ensemble de notes que l'utilisateur pourra éventuellement modifier par les requêtes de notation.

Nous résumons l'utilisation du langage de requêtes par un exemple complet :

- Soit une base documentaire contenant des publications ornithologiques sous la forme d'articles de périodiques, d'actes de colloques et d'extraits de livres. Chaque document possède en plus de son texte les noms de ses auteurs, le nom de l'éditeur ou du périodique, la date de parution et la bibliographie citée en référence.

- L'utilisateur francophone désire obtenir des renseignements précis sur la nourriture ingérée par les aigles et apparentés. Il se sert de la tolérance aux fautes pour sélectionner les noms des auteurs et des genres et de la notation pour classer les articles en fonction de leur langue.

- Description de la base :

BASE ::= LISTE-ARTICLES
LISTE-ARTICLES ::= ARTICLE | ARTICLE LISTE-ARTICLES
ARTICLE ::= séparateur article LISTE-RUBRIQUES
LISTE-RUBRIQUES ::= ENTETE séparateur rubrique TEXTE séparateur rubrique BIBLIOGRAPHIE
ENTETE ::= LISTE-PHRASES
TEXTE ::= LISTE-PARAGRAPHES
BIBLIOGRAPHIE ::= LISTE-PARAGRAPHES
LISTE-PARAGRAPHES ::= PARAGRAPHE | PARAGRAPHE LISTE-PARAGRAPHES
PARAGRAPHE ::= à la ligne LISTE-PHRASES
LISTE-PHRASES ::= PHRASE | PHRASE LISTE-PHRASES
PHRASE ::= LISTE-MOTS point
LISTE-MOTS ::= MOT | MOT LISTE-MOTS
MOT ::= blanc LISTE-CARACTERES | signe de ponctuation LISTE-CARACTERES
LISTE-CARACTERES ::= caractère | caractère LISTE-CARACTERES

- Interrogation de la base :

Trouver les ARTICLE contenant les MOT suivant :
((AIGLE ou EAGLE ou ADLER ou (AQUILA ou HIERAAETUS) [avec faute]) et (NOURRITURE ou PROIE ou PREY ou PREDATION ou CHASSE ou RAUB) et ((TERRASSE ou GEROUDET ou HAINARD ou STEMMLER ou CURRY-LINDHAL ou (BROWN et WATSON dans la même PHRASE)) [avec faute] dans l'ENTETE ou dans le même PARAGRAPHE) mais pas (BALBUZARD ou CIRCAETE ou PYGARGUE ou HALIAETUS [avec faute]).

- Notation :

Si l'ARTICLE contient AIGLE alors sa note sera 200.
Si l'ARTICLE contient ADLER alors sa note sera 175.
Si l'ARTICLE contient EAGLE alors sa note sera 150.

Si l'ARTICLE contient AQUILA ou HIERAAETUS
alors sa note sera 100.

Remarque : Dans cet exemple nous n'avons présenté qu'une seule requête d'interrogation ; il est clair que l'utilisateur au vu des réponses obtenues peut itérer sur d'autres questions plus précises.

II - L'ARCHITECTURE DU PROCESSEUR TREFLE

Cette seconde partie de la présentation du processeur TREFLE est introduite par une brève description des principes fonctionnels qui sera suivie d'une analyse plus détaillée des différents modules composant l'architecture de la machine.

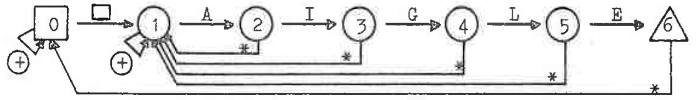
2.1. LES PRINCIPES FONCTIONNELS

Après avoir présenté les problèmes posés par la consultation de données textuelles, nous aborderons certains aspects particuliers du langage de requête en proposant des solutions pour en implémenter les principes.

Dans un système général de consultation de données on peut décomposer le processus de réponse aux questions en trois étapes principales se succédant ainsi :

- une étape de recherche des mots de la question dans la base de données,
- une étape d'évaluation de la fonction logique exprimée par la question,
- une étape de classement des articles en fonction de leur adéquation à la question.

- la recherche des mots : le problème de la reconnaissance des mots dans la base est étroitement apparenté à celui de la recherche d'une chaîne de caractères dans une autre chaîne. La méthode généralement mise en oeuvre utilise des automates à états finis, associés aux grammaires régulières de reconnaissance. L'exemple suivant présente l'automate recherchant la chaîne de caractères AIGLE dans un mot.



0 : état initial
6 : état final
□ : début de mot
⊕ : autre caractère avec progression de l'automate dans la chaîne d'entrée
* : autre caractère, l'automate restant positionné sur ce caractère.

- l'évaluation de l'expression logique : l'automate de reconnaissance a pour rôle de restituer un vecteur booléen reflétant la présence ou l'absence des mots de la question dans la base. En général on évalue par calcul séquentiel la fonction logique sur le vecteur construit, solution longue et coûteuse si la question est complexe. Pour accélérer cette évaluation nous proposons la tabulation de la fonction logique et le vecteur booléen servira d'index dans la table contenant les valeurs de la fonction.

- le classement des articles : dans un système traditionnel le classement aboutit à la conservation ou au rejet de l'article. TREFLE ayant deux modes de fonctionnement, le problème se complique. Dans le mode de recherche exacte le processeur rejette tous les documents ne satisfaisant pas complètement à la question. Dans le mode de recherche floue le classement s'apparente à un tri sur les notes affectées aux articles.

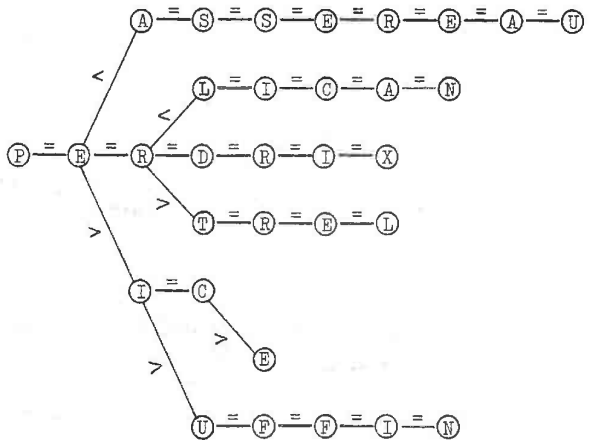
Après cette présentation des problèmes généraux posés par la consultation de données nous reprenons quelques points particuliers du langage de requête de TREFLE :

- la non indexation,
- la complexité des questions n'influençant pas le temps de traitement,
- la tolérance aux fautes,
- la prise en compte de la structure naturelle des données.

- la non indexation : l'utilisation de ce principe implique l'abandon de l'accès aléatoire et la lecture complète de tous les documents pour y effectuer la recherche des mots apparaissant dans la question. Cette méthode, nommée recherche directe, utilisée dans plusieurs réalisations dont MICROTEXT [Bum 72] a l'inconvénient d'augmenter le temps de réponse proportionnellement à la taille de la base. Pour ne pas trop pénaliser l'utilisateur, nous fixons comme objectif au processeur TREFLE de traiter les caractères à leur vitesse de transfert depuis le support de la base ; dans le cadre de l'application développée, cette vitesse est de 1,2 millions de caractères par seconde. Cet objectif introduit des contraintes très sévères sur le temps de recherche et d'évaluation des questions.

- la complexité de la question n'exerce pas d'influence sur le temps de traitement : cette contrainte implique la révision du principe d'une évaluation calculée de l'expression logique formulée par la requête et d'une comparaison purement séquentielle entre les données de la base et les mots de la question. En effet pour une requête comprenant plusieurs dizaines de mots-clés une évaluation par tabulation offrira une rapidité indépendante de la complexité de la question. Le problème des comparaisons se pose dans les termes suivants : l'alphabet étant composé de 26 lettres, il nous est impossible, avec la méthode de recherche directe, d'effectuer les 26 comparaisons possibles dans le temps séparant l'arrivée de deux octets consécutifs depuis le support de la base. Pour l'application, ce temps est d'environ 800 nanosecondes, ce qui laisserait bien peu de temps, environ 30 nanosecondes, pour un cycle de comparaison... Pour accélérer ce traitement de recherche nous appliquons le principe de la dichotomie qui réduit considérablement le nombre de comparaisons à effectuer, ainsi pour l'exemple nous le ramenons de 26 à 5. Le compilateur construit pour la question l'arbre dichotomique balancé des comparaisons. Chaque noeud de cet arbre contient le caractère à comparer et les pointeurs vers les 3 fils correspondant aux résultats de la comparaison : inférieur, égal ou supérieur. Ce mécanisme est illustré par l'exemple ci-dessous :

Soient les mots suivants à rechercher dans la base :
PIC, PETREL, PASSEREAU, PERDRIX, PIE, PELICAN, PUFFIN.
L'arbre dichotomique balancé des comparaisons pour la reconnaissance se présente ainsi :



On remarque que 2 comparaisons permettent de différencier les radicaux PA, PE, PI et PU.

- la tolérance aux fautes d'orthographe : l'application de ce principe

nécessite quelques restrictions : tolérer des fautes dans des motifs communs à plusieurs débuts de mots introduit des problèmes de retour arrière que le traitement au vol interdit par manque de temps et de place mémoire. En revanche une tolérance limitée à quelques fautes dans les branches linéaires de l'arbre de comparaison ne dégrade pas le temps de traitement de façon prohibitive.

Examinons les différents cas de fautes : généralement une faute d'orthographe peut provenir soit du remplacement d'une lettre par une autre, soit de l'insertion d'une lettre parasite, soit d'un oubli de lettre ; par exemple :

Aigle criart, aigle imppérial, aigle boté.
(d) (p de trop) (t manque)

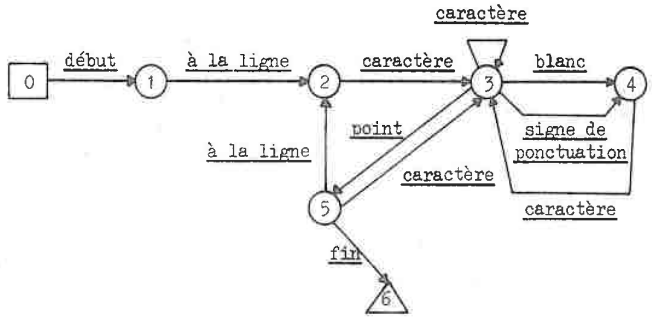
La combinaison de ces phénomènes risque de rendre inextricable la recherche d'un mot erroné. Pour cette raison TREFLE limite à deux le nombre de fautes tolérées par mot, avec la restriction qu'une seule faute sur les deux provienne d'un oubli ou d'une insertion.

- la prise en compte de la structure naturelle des textes : le langage décrivant la structure naturelle des textes est en général un langage régulier reconnu par un automate à états finis. Prenons pour exemple la structure d'un article décrite par la grammaire suivante :

ARTICLE ::= début LISTPARAGRAPHE fin
LISTPARAGRAPHE ::= PARAGRAPHE LISTPARAGRAPHE | PARAGRAPHE
PARAGRAPHE ::= à la ligne LISTPHRASE
LISTPHRASE ::= PHRASE LISTPHRASE | PHRASE
PHRASE ::= LISTMOT point
LISTMOT ::= MOT blanc LISTMOT | MOT signe de ponctuation LISTMOT | MOT
MOT ::= caractère | caractère MOT

Note : Tous les termes soulignés représentent des terminaux du langage.

L'automate associé à cette grammaire se présente ainsi :



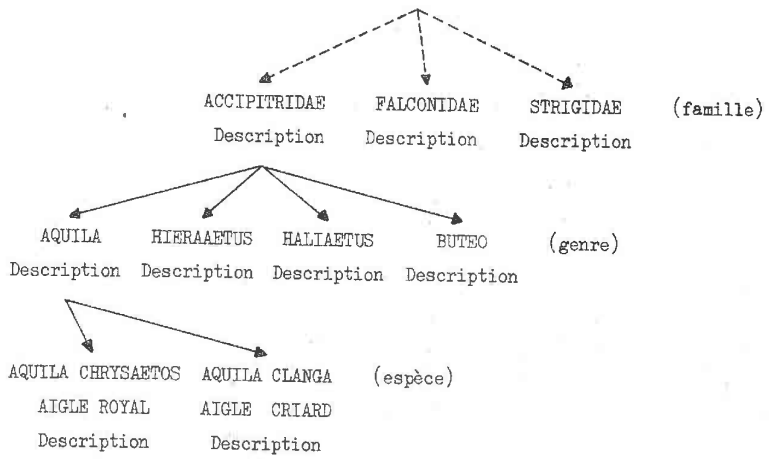
Dans certains cas particuliers la structure des données peut être plus complexe et faire appel à un langage à contexte libre [DDH 72]. L'exemple ci-dessous, inspiré de SOCRATE [Soc 74] présente une structure dont l'imbrication est similaire à un parenthèsage. Il décrit une base de documents ornithologiques ordonnés suivant une classification arborescente :

- la racine est composée de la description de la famille, par exemple : FALCONIDAE, ACCIPITRIDAE, STRIGIDAE, etc... ;
- le niveau intermédiaire est composé de la description du genre, par exemple : AQUILA, HIERAAETUS, BUTEO, etc... ;
- les feuilles sont composées de la description des espèces, par exemple : AQUILA CHRYSAETOS, AQUILA CLANGA, etc...

Structure :
ENTITE 3 (niveau famille : ACCIPITRIDAE, etc...)
début
DESCRIPTION : les caractères spécifiques à la famille
ENTITE 9 (niveau genre : AQUILA, etc...)

début
DESCRIPTION : les caractères spécifiques au genre
ENTITE 11 (niveau espèce : AQUILA CHRYSAETOS, AQUILA CLANGA, etc...)
début
DESCRIPTION : l'espèce : noms vulgaires, caractères propres à l'espèce, moeurs, bibliographie, etc...
fin
fin
fin

Ramification décrivant la structure :

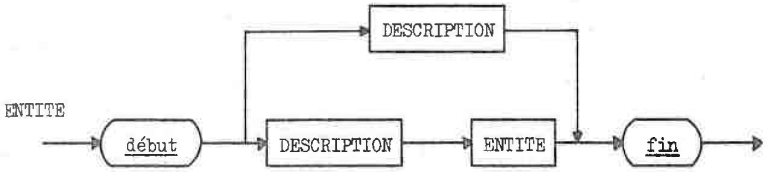


Cette structure est générée et reconnue par la grammaire suivante :

ENTITE ::= début CORPS fin
CORPS ::= DESCRIPTION ENTITE | DESCRIPTION

La grammaire peut être représentée par un graphe syntaxique implanté dans une structure de données. L'algorithme de transformation de la grammaire en une structure de données équivalente est décrit dans [CoG 70] et [Wir 76].

Graphe syntaxique de la grammaire :



La structure de données représentant le graphe syntaxique est interprétée par un algorithme d'analyse syntaxique récursif. Une réalisation simple de ce principe consiste à interpréter le graphe avec l'algorithme de séquencement d'un automate à pile, chaque noeud du graphe constituant une instruction du processeur d'analyse syntaxique.

Pour terminer ce paragraphe, nous résumons le principe de fonctionnement de la machine TREFLE de la manière suivante :

L'utilisateur pose une question composée de phrases de description, d'interrogation et de notation. La résolution de cette question peut être divisée en deux étapes :

1ère étape : le compilateur de requêtes traduit cette question :
- en une grammaire qu'il représente sous forme de graphe syntaxique. Ce graphe, décrit par une structure de données, sera interprété pour effectuer les comparaisons nécessaires à la recherche des mots dans la base. Pour accélérer cette recherche, la structure de ce graphe sera dichotomique ;

- en une série de tables qui serviront à évaluer directement la fonction logique exprimée par la question. Cette évaluation attribue une note à chaque document traité.

2ème étape : le processeur TREFLE est activé pour :
- scruter un domaine de la base en interprétant le graphe représentant la grammaire. Cette interprétation consiste à comparer les caractères émis par le support des données, aux caractères stockés dans l'automate, représenté par le graphe syntaxique, et à acheminer dans ce graphe en fonction des résultats de la comparaison, avec l'algorithme de séquencement d'un automate à pile ;
- classer les documents scrutés en fonction de la note qui leur a été attribuée. Ce classement s'accompagne d'une mémorisation des documents retenus pour les restituer à l'utilisateur.

2.2. LE PROCESSEUR TREFLE

L'architecture que nous présentons maintenant intègre des fonctions en utilisant les principes fonctionnels définis ci-dessus pour assurer une interprétation efficace du langage de requêtes.

Globalement, la machine se présente ainsi :

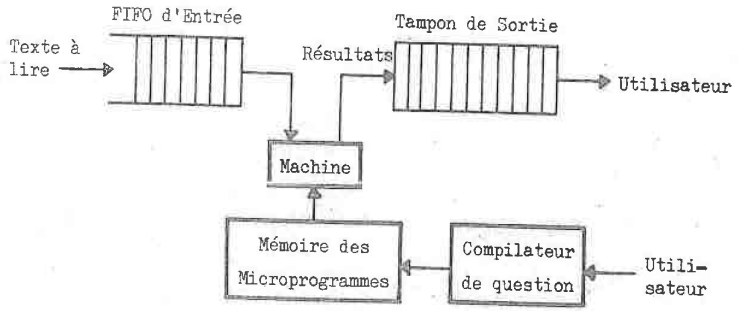


Figure I-1 : Schéma de principe de la machine TREFLE.

Le texte lu transite dans une mémoire gérée en file Premier Arrivé Premier Servi (FIFO) et les résultats sont restitués dans une mémoire tampon.

La machine est microprogrammée et la mémoire des microprogrammes est une mémoire vive dans laquelle le compilateur charge la structure de données (le microprogramme) représentant la grammaire, obtenue par la compilation de la question.

Plus précisément, le prototype développé pour l'application aura la structure suivante (cf. fig. I-2) :

- le coeur de la machine est composé d'un ensemble de processeurs appelés FILTRES spécialisés dans la scrutation des données ;
- le MODULE DE CALCUL capable de traiter les valeurs numériques assiste les FILTRES dans leur rôle de notation ;
- le MODULE DE CLASSEMENT chargé du tri et de la mémorisation des documents complète l'architecture fonctionnelle ;
- le DISQUE sert à mémoriser l'ensemble de la base de données ;
- le FORMATTEUR, gérant de la structure de données sur le support de la base, relie le disque au FILTRE et au MODULE DE CLASSEMENT ;
- le CONTROLEUR supervise les autres processeurs et assure la communication entre TREFLE et les utilisateurs à travers un minicalculateur hôte.

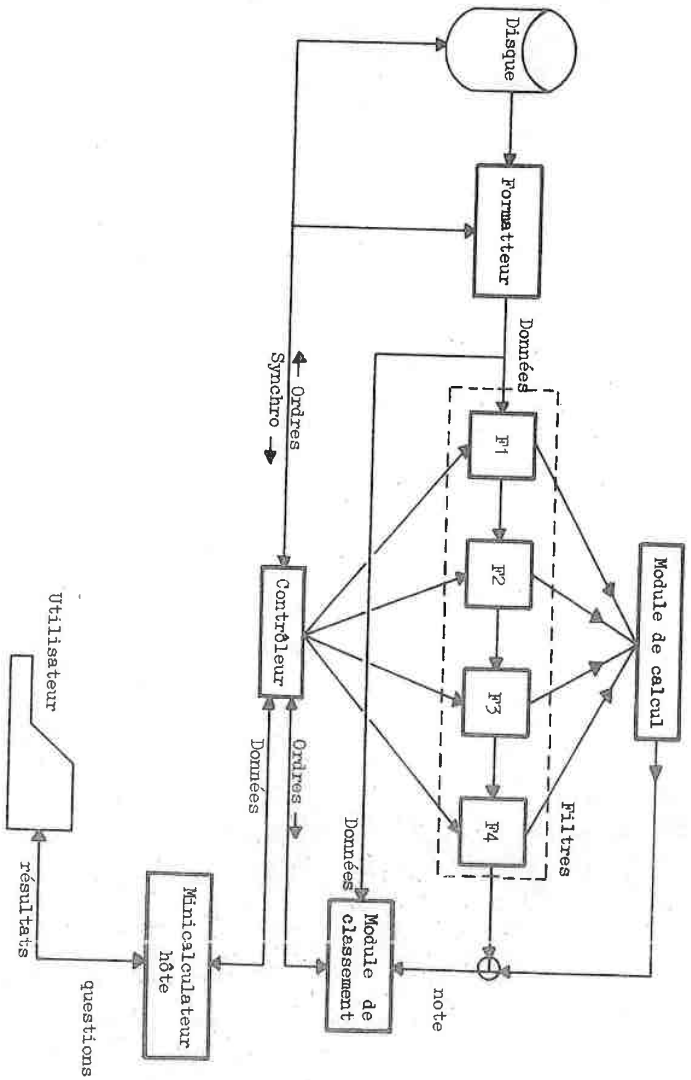


Figure I-2 : Structure de la réalisation expérimentale du processeur TREFLE.

2.2.1. LES FILTRES

Les FILTRES [Tus 77] ont pour rôle d'interpréter le microprogramme résultant de la compilation de la question et d'évaluer par tabulation la fonction logique exprimée par la requête.

Ces processeurs sont ordonnés "en série" selon un schéma "Pipe Line", chacun traitant une partie de la grammaire issue de la question (cf. fig. I-3). Par exemple :

- la première station traite les caractères pour les regrouper en mots,
- la seconde station traite les mots pour les regrouper en phrases,
- la dernière station traite les paragraphes et attribue la note globale à l'article.

Pour des raisons d'efficacité et de modularité ces processeurs sont indépendants et asynchrones, les communications entre eux étant assurées par des mémoires gérées FIFO, réalisations physiques des files d'attente de communication.

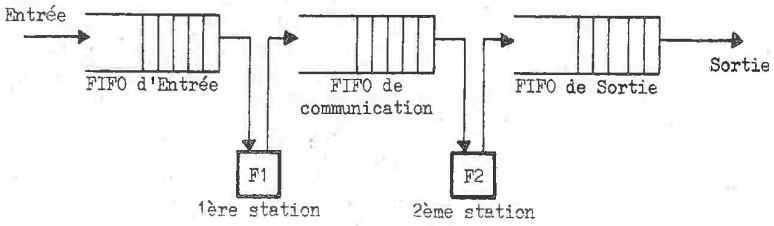


Figure I-3 : "Pipe Line" de FILTRES à 2 stations.

Le mécanisme de séquencement d'un processeur FILTRE est directement inspiré de celui d'un automate à pile [GrL 67], [Pai 65]. La grammaire interprétée par cet automate est mise sous la forme de graphe syntaxique représenté par une structure de données. Les alternatives des règles de la grammaire seront ordonnées de sorte que leur choix puisse être accéléré par une méthode dichotomique. Chaque micro-instruction représente un noeud du graphe, le microprogramme complet formant la grammaire. Les micro-instructions seront du type long et découpées en champs regroupés selon leur fonction :

- les champs décrivant la structure de données et concernant le séquencement du processeur ;
- les champs commandant l'évaluation de la fonction logique et concernant la partie opérative du processeur.

2.2.1.1. L'interprétation de la grammaire

Les champs de la micro-instruction décrivant la structure de données sont les suivants (cf. fig. I-4) :

- le champ CAR-DEF contient
  . soit la valeur du caractère terminal à comparer avec le caractère courant de la chaîne d'entrée,
  . soit l'adresse d'un noeud dans le cas du traitement d'un non terminal ;

- le champ TAG précise la signification de CAR-DEF : Si TAG=0 alors CAR-DEF contient un caractère et le noeud traité est une feuille de l'arborescence, sinon CAR-DEF pointe sur un autre noeud ;

- le champ SUC contient l'adresse du noeud où il faut se brancher en cas de succès lors de la comparaison ;

- les champs ALT1 et ALT2 indiquent les adresses des noeuds à exécuter

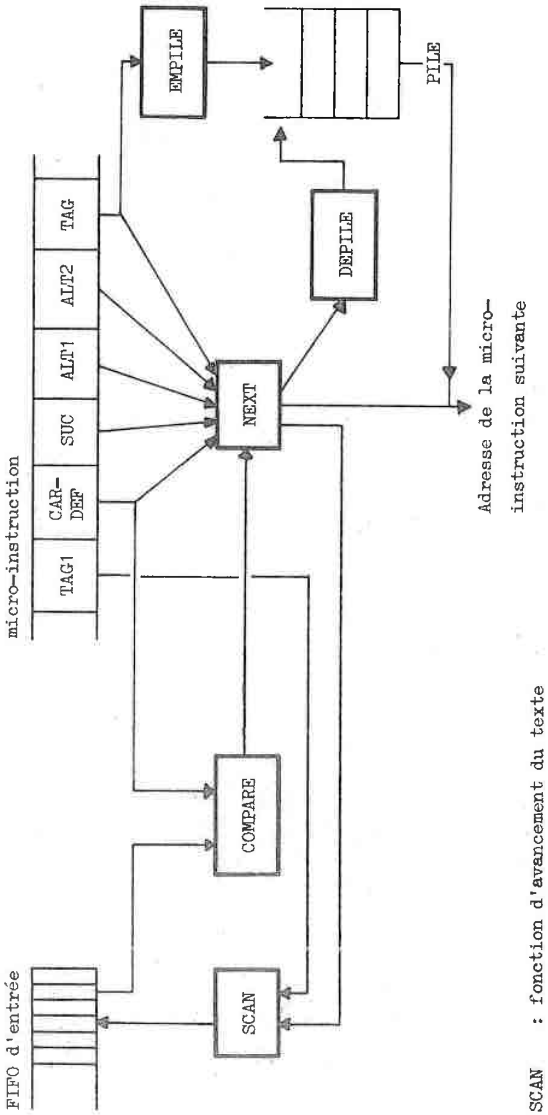


Figure I-4 : Schéma fonctionnel de l'interprète de la structure de données. (automate à pile)

en cas d'échec lors de la comparaison avec la convention suivante :

- si CAR est supérieur au caractère entré alors ALT1 contient l'adresse du noeud suivant dans l'arborescence,
- si CAR est inférieur au caractère entré alors c'est ALT2 qui contient cette adresse.

Remarques : . nous assimilons le branchement vers un noeud non terminal de l'arborescence (TAG=1) à un appel de sous-programme. Grâce à une pile câblée les adresses de retour sont sauvegardées automatiquement, leur restauration intervenant dès la rencontre de la fin de l'arborescence appelée. L'empilement est commandé par le champ TAG et l'apparition de la valeur NIL comme adresse choisie provoque le dépilement ;

. la grammaire interprétée ne doit pas être ambiguë car les retours arrières dans le texte source sont impossibles.

L'algorithme récursif d'interprétation de la grammaire est donné ci-dessous dans une forme proche de sa réalisation câblée :

```
ANALYSE (X)
BOUCLE : si TAG(X)≠0 alors Y:=ANALYSE(DEF(X))
         sinon début
               si CAR(X)=FIFO(I) alors I:=I+1 % progression
                                         dans la chaîne d'entrée %
                                         Y:=0
               sinon si CAR(X) > FIFO(I) alors Y:=1
                     sinon Y:=2
               fin
         si Y=0 alors X:=SUC(X)
         sinon si Y=1 alors X:=ALT1(X)
               sinon X:=ALT2(X)
         si X≠NIL alors allera BOUCLE
fin ANALYSE
```

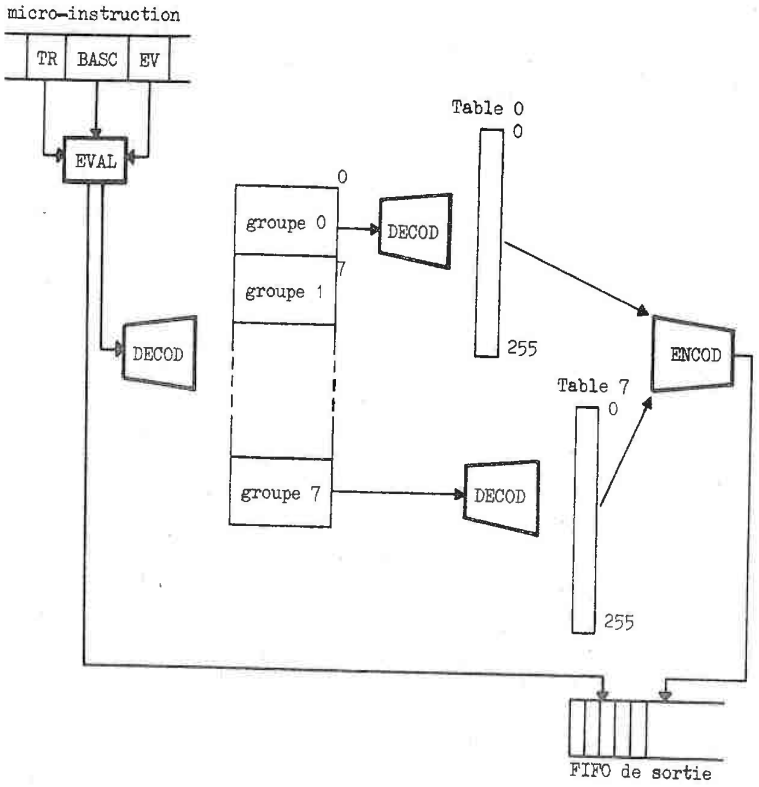

Le FILTRE fonctionne comme un analyseur syntaxique reconnaissant que la suite des caractères entrés est une phrase engendrée par la grammaire. Notre objectif n'est pas de déterminer si un texte entier est conforme à une grammaire mais plutôt s'il contient certains motifs satisfaisant celle-ci, séparés entre eux par des séquences ne nous intéressant pas. Pour ce faire nous introduisons dans la micro-instruction le champ TAG1 commandant l'avancement dans le texte indépendamment du résultat de la comparaison.

2.2.1.2. L'évaluation de la fonction logique

Le rôle d'un FILTRE n'est pas seulement d'analyser le texte lu mais aussi de transcoder les motifs reconnus pour leur attribuer une note. Ce transcodage peut être réalisé directement ou par l'intermédiaire de tables (cf. fig. I-5).

Pour transcoder directement un motif il suffit de commander la sortie du caractère de transcodage dans la micro-instruction terminant la branche de l'arborescence reconnaissant ce motif. Le caractère de transcodage est contenu dans le champ BASC de la micro-instruction, la commande de sortie dans le champ TR.

Le transcodage par tabulation de la fonction d'évaluation est plus complexe. En fonction de la valeur du champ EV, le champ BASC peut adresser 64 bascules formant 8 groupes de 8. Chaque groupe référence une table de vérité de 256 × 1 bits. Nous obtenons ainsi 8 tables de vérité représentant chacune une fonction logique de 8 variables. La gestion de ce mécanisme est entièrement automatisée par le compilateur qui charge les tables de vérité avec les valeurs de la fonction logique de la question, et qui construit les micro-instructions contenant les caractères nécessaires au transcodage.



EVAL : fonction de choix de la sortie
DECOD : fonction d'adressage
ENCOD : encodeur prioritaire

Figure I-5 : Le mécanisme d'évaluation des fonctions logiques.

2.2.1.3. La résolution des fautes d'orthographe

La scrutation du texte se complique par la tolérance de l'automate aux fautes d'orthographe. L'algorithme de détection des fautes nécessite la comparaison en parallèle du caractère de la micro-instruction (champ CAR-DEF) avec 3 caractères consécutifs du texte d'entrée pour découvrir les erreurs provenant de l'absence ou de l'insertion d'une lettre dans un motif. Ainsi, à la fin d'un motif reconnu avec une faute provoquée par l'insertion d'une lettre parasite, le texte sera en retard d'un caractère par rapport au graphe des micro-instructions. Ce décalage introduit par l'omission ou l'insertion d'une lettre nécessite un mécanisme rétablissant le synchronisme entre le texte et le programme. La fonction SCAN commandée par le champ TR dispose des renseignements nécessaires sur la présence et la nature des fautes pour corriger le décalage en avançant ou en retardant le texte par rapport au programme.

2.2.1.4. Les caractéristiques de la réalisation

Pour conclure cette description du processeur FILTRE, nous présentons quelques caractéristiques du prototype développé.

Construit avec des composants de technologie TTL Schottky, son cycle de base est de 150 nanosecondes. Conçu comme un processeur "pipe line" à deux stations, il traite deux micro-instructions à la fois :

- pour la première il exécute la partie évaluation (Actions),
- pour la seconde il détermine l'adresse de la micro-instruction suivante (Séquencement).

Une micro-instruction a besoin de deux cycles pour s'exécuter, mais l'avancement dans le microprogramme est d'une micro-instruction par cycle.

La mémoire des microprogrammes contient 256 micro-instructions de 55 bits.

La pile de récursivité est de profondeur 16 et les mémoires FIFO d'entrée et de sortie ont 256 emplacements chacune.

Une évaluation grossière de ses performances à l'aide d'un oscilloscope à mémoire nous permet de prédire une augmentation de vitesse de l'ordre de la centaine de fois par rapport au MITRA 125 de même technologie.

2.2.2. LE MODULE DE CALCUL

Les FILTRES étant inaptes à effectuer des opérations arithmétiques sur les données numériques, nous confions ce rôle à un MODULE DE CALCUL [GaR 78] spécialement conçu pour effectuer des comptages, histogrammes et autres calculs simples. Les résultats élaborés par ce processeur serviront à moduler la note obtenue par le document à l'issue de son filtrage. Par exemple, le MODULE DE CALCUL compte les occurrences des mots de la question dans chaque document afin de restituer ceux qui en possèdent le plus grand nombre.

Les adresses des instructions exécutées par ce processeur sont émises par les FILTRES auxquels il est relié par un ensemble de mémoires FIFO de communication (cf. fig. I-6). Stockées dans une mémoire spécialisée, les instructions adressent leurs opérandes rangés en mémoire locale de données ou dans les mémoires de communication. Les opérations qu'elles réalisent sont les suivantes :

- complémentation d'un opérande,
- addition de deux opérandes,
- évaluation des fonctions Minimum et Maximum sur deux opérandes,
- comptage sur un opérande.

Nous terminons cette description sommaire du MODULE DE CALCUL en citant quelques unes de ses principales caractéristiques.

Les opérandes traités par ce processeur ont une taille de 8 bits et les instructions à deux adresses sont codées sur 30 bits. Les mémoires de données et d'instructions contiennent chacune 256 éléments. Les mémoires FIFO de communication ont une largeur de 16 bits et une longueur de 64 mots. La mémoire FIFO de sortie a une longueur de 8 bits et une longueur de 256 caractères.

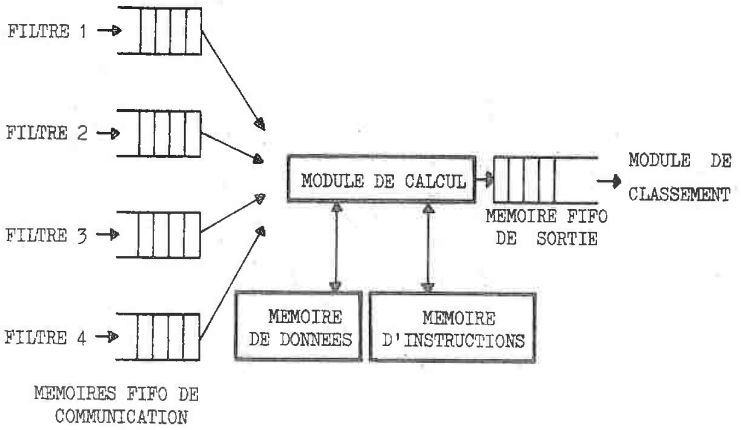


Figure I-6 : Le MODULE DE CALCUL.

2.2.3. LE MODULE DE CLASSEMENT

Parallèlement à la scrutation du flot des données par les FILTRES, le MODULE DE CLASSEMENT [Mas 77] mémorise et classe les documents au rythme de leur arrivée. Cette "mémoire intelligente" offre à l'utilisateur le choix entre deux modes de fonctionnement :

- le mode de recherche exacte,
- le mode de recherche floue.

Dans le mode de recherche floue l'utilisateur désire recevoir les N meilleures réponses à sa question. La valeur de N est déterminée par la taille de la mémoire tampon et la longueur des documents. Le MODULE DE CLASSEMENT trie des documents arrivant selon leur note et lorsqu'il atteint un seuil de remplissage il rejette les documents les moins bien notés pour libérer l'espace nécessaire à l'acquisition des documents suivants. Cette méthode ne correspond pas tout à fait à un tri car on rejette des documents peut-être meilleurs que ceux qui arrivent. A la fin de la recherche le MODULE DE CLASSEMENT contient les N meilleures réponses plus la dernière arrivée, sans aucun a priori sur la qualité de ces réponses.

Dans le mode de recherche exacte l'utilisateur désire recevoir tous les documents qui vérifient le critère donné dans la question. Dans ce mode, le MODULE DE CLASSEMENT conserve tous les documents ayant une note suffisante. Il peut donc arriver que sa mémoire se remplisse complètement auquel cas il prévient le contrôleur qui en effectue le transfert vers l'utilisateur.

Un automate à états finis reconnaissant les différents caractères séparateurs de la structure de donnée : début et fin de descripteurs, début et fin de champ, etc... opère le découpage de la chaîne d'entrée.

Un autre rôle de ce processeur est la réalisation de l'opération de projection au sens défini par CODD [Cod 70]. Cette opération consiste à ne mémoriser pour chaque descripteur que les champs spécifiés par l'utilisateur.

La taille de la mémoire de classement du prototype développé est de 16 K octets ; elle est gérée par allocation dynamique de blocs de taille fixe. Pour obtenir la grande vitesse de traitement requise par le débit du disque, les algorithmes de tri, de reconnaissance de la structure et de gestion de la mémoire sont câblés.

2.2.4. LE DISQUE

L'unité de disque reliée au processeur TREFLE a une capacité de 300 millions d'octets et un débit d'environ 10 millions de bits par seconde. Le disque magnétique sert à mémoriser l'ensemble des descripteurs des documents de la base dans une structure de données physique composée des secteurs inscrits sur le support.

Cette unité et la structure de données associée sont présentées dans la seconde partie du chapitre II et dans l'annexe D.

2.2.5. LE FORMATTEUR

Le FORMATTEUR [Tus 78] est chargé de traiter la structure de données implantée sur le support magnétique en utilisant les signaux de contrôle issus de l'unité de disque. Recevant le flot sériel de bits, il le transforme en flot de caractères et sépare les informations propres à la structure physique des caractères composant les descripteurs de documents. Ces derniers sont envoyés octet par octet dans la mémoire d'entrée de la première station de filtrage ainsi que dans la mémoire d'entrée du module de classement.

Le FORMATTEUR fait lui aussi l'objet d'une description détaillée dans la seconde partie du chapitre II et dans l'annexe E.

2.2.6. LE CONTROLEUR

Le CONTROLEUR a pour rôle de superviser le fonctionnement du processeur

TREFLE. Cette gestion peut être décomposée en trois étapes :

- première étape : compilation de la question et initialisation du traitement. Le CONTROLEUR charge le microprogramme résultant de la compilation de la question dans les mémoires des processeurs FILTRES. Puis il initialise le MODULE DE CLASSEMENT et active le FORMATTEUR qui commence à lire le disque ;

- seconde étape : surveillance et gestion du traitement. Le CONTROLEUR gère les différentes interruptions en provenance du disque, du FORMATTEUR ou du MODULE DE CLASSEMENT et assure la synchronisation entre les différents processeurs ;

- troisième étape : restitution des résultats. En fin de traitement, le CONTROLEUR renvoie à l'utilisateur les informations contenues dans la mémoire du MODULE DE CLASSEMENT.

Le CONTROLEUR est réalisé autour d'un système déjà existant, celui du microprocesseur TEXAS INSTRUMENTS TMS 9900.

2.2.7. LE MINI-ORDINATEUR HOTE

Le rôle de ce mini-ordinateur est double : il assure la gestion des utilisateurs et il permet un chargement aisé de la base documentaire sur le disque.

En effet, plutôt que de réaliser un logiciel conversationnel sur le contrôleur nous préférons utiliser celui déjà existant sur le mini-ordinateur.

Le problème du chargement de la base se pose dans les termes suivants :

- le disque supportant la base ne peut être lu et écrit qu'à travers le FORMATTEUR,

- le CONTROLEUR ne possède pas de périphérique à grand débit, par exemple disque ou bande magnétique.

Il nous faut donc relier le CONTROLEUR à un calculateur hôte capable d'utiliser des fichiers déjà existants.

Le mini-ordinateur utilisé est le MITRA 125 SEMS et la première partie du chapitre II présente la solution retenue pour sa connexion avec le CONTROLEUR.

CONCLUSION : LES APPLICATIONS

La caractéristique technique fondamentale du processeur TREFLE est de traiter le flot de données en une seule lecture et à la vitesse de son débit, ce qui lui permet de traiter non seulement des données mémorisées sur un support mais ausi celles transitant sur une voie, ligne téléphonique ou onde électromagnétique par exemple. Les applications que nous développons actuellement explorent ces deux types d'utilisations.

La première application mise en oeuvre consiste à automatiser la recherche bibliographique dans un centre de documentation. Tous les descripteurs des ouvrages de cette bibliothèque sont enregistrés sur bande magnétique sans aucun fichier inverse. Nous espérons aboutir avant la fin de l'année 79 à la mise en oeuvre du prototype sur cette base documentaire.

La seconde, plus proche de la TELEMATIQUE, est fondée sur l'utilisation du système ANTIOPE développée par le CCETT (Centre Commun d'Etudes pour la Télécommunication et la Télévision). Ce système permet l'émission de textes sur les ondes et leur réception dans les postes de télévision. On peut supposer que dans un avenir proche certains journaux spécialisés - par exemple des cours de la bourse ou l'état des routes - transitent par cette voie. Il serait intéressant que les téléspectateurs puissent

disposer d'un outil tel que TREFLE pour consulter les données émises.

Au-delà de ces deux applications se profile une utilisation de TREFLE très ouverte vers l'avenir :

- en tant que processeur spécialisé dans la consultation de bases de données, périphérique d'un ordinateur hôte,

- en tant qu'outil de recherche dans les domaines du traitement d'informations non numériques par exemple comme processeur d'analyse syntaxique en Reconnaissance des Formes [HaM 76], [Mas 78] ou comme opérateur spécialisé dans l'interprétation du langage SNOBOL 4 [Jar 75],

- en tant que processeur de consultation de vidéo disque : le vidéo disque, disque de la taille et du prix d'un "33 tours", est destiné à contenir des images de télévision, mais il peut aussi stocker environ 1 milliard de caractères. Son lecteur sera d'ici quelques années de la complexité et du prix d'un ensemble haute fidélité. Il est évident que la consultation d'un tel volume de données ne peut se concevoir sans l'outil informatique, et un processeur du type TREFLE semble plus adapté qu'un système de gestion de base de données.

CHAPITRE 2 :

Les Communications

INTRODUCTION

I - LA LIAISON ENTRE TREFLE ET LE MINICALCULATEUR HOTE

1.1. La structure du mécanisme de communication standard entre le Mitra 125 et les unités périphériques

1.1.1. Les mécanismes de communication
1.1.2. La structure du processeur d'entrée/sortie (IOP)

1.2. La communication entre TREFLE et le MITRA 125

1.2.1. Les principes
1.2.2. La connexion matérielle
1.2.3. Le protocole logiciel

II - LA LIAISON ENTRE TREFLE ET LE DISQUE

2.1. Les principes
2.2. La structure de données implantée sur le disque
2.3. Les algorithmes des fonctions opérant sur la structure de données
2.4. La structure du formatteur

III - LE SYSTEME DE PRODUCTION DE LOGICIEL

3.1. Les principes
3.2. Les réalisations

CONCLUSION

INTRODUCTION

Ce chapitre décrit les solutions aux problèmes posés par la communication entre le processeur TREFLE et son environnement. Pour introduire les réalisations effectuées il nous semble nécessaire de résumer succinctement les descriptions des processeurs intervenant dans la communication. Dans le domaine où se situent les problèmes à résoudre, l'appréhension de l'environnement très complexe constitue la principale difficulté. En effet, les algorithmes mis en oeuvre sont élémentaires, mais les solutions nécessitent à la fois des structures matérielles, des microprogrammes [Wil 51], [Wil 69], [Ros 73] et des procédures logicielles.

Dans une première partie nous étudions le mécanisme de communication entre le processeur TREFLE et le minicalculateur hôte MITRA 125 [Mit 75]. La solution adoptée utilise un IOP-Input Output Processor- processeur standard d'entrée/sortie, microprogrammable, accédant directement à la mémoire centrale du calculateur et spécialisé dans les transferts rapides d'informations [IOP 75]. Nous avons doté ce processeur d'un interface spécifique à la machine TREFLE lui permettant d'accéder à la mémoire du CONTROLEUR et nous avons développé des microprogrammes spécialisés qui interprètent les commandes particulières d'échanges.

Dans une seconde partie nous présentons la liaison effectuée entre le processeur TREFLE et le disque, support des données de la base. Après la présentation de la démarche suivie pour la conception de la liaison, nous décrivons les différents processeurs y intervenant - disque et FORMATTEUR - en détaillant la structure de données implantée sur le disque et les fonctions qui la gèrent. La microprogrammation des algorithmes des fonctions sur le FORMATTEUR a permis de valider l'architecture conçue pour ce processeur et a montré son adéquation au pro-

blème posé par la connexion.

Une troisième partie décrit brièvement les outils de production de logiciel réalisés pour faciliter la mise en oeuvre des microprogrammes. Ces outils se composent essentiellement de simulateurs et de micro-assembleurs développés sous le système A.P.L. [Ive 62].

I - LA LIAISON ENTRE TREFLE ET LE MINICALCULATEUR HOTE

Le processeur TREFLE est relié aux utilisateurs par un calculateur hôte qui est chargé de la gestion des messages formant les questions, et qui permet une création aisée de la base documentaire. L'objet de cette étude est de développer et d'implanter un mécanisme de communication entre le CONTROLEUR, superviseur du processeur TREFLE, et la mémoire du calculateur hôte.

Afin de réaliser cette connexion, une connaissance approfondie du logiciel et du matériel MITRA 125 est indispensable. Dans un premier paragraphe nous présentons la structure matérielle et les instructions d'entrées/sorties du mécanisme standard. La seconde partie décrira la solution adoptée pour réaliser la communication. Cette solution fait appel à des procédures logicielles, des microprogrammes et une structure matérielle établissant les liaisons physiques.

1.1. LA STRUCTURE DU MECANISME DE COMMUNICATION STANDARD ENTRE LE MITRA 125 ET LES UNITES PERIPHERIQUES

Nous décrivons dans cette partie la structure et le fonctionnement des mécanismes permettant de faire des échanges d'informations entre la mémoire centrale du calculateur, l'unité centrale, le processeur d'entrées/sorties et les unités périphériques standard.

1.1.1. LES MECANISMES DE COMMUNICATION

1.1.1.1. Présentation du calculateur Mitra 125

Le calculateur MITRA 125 est construit autour d'une structure à deux bus : le bus mémoire et le bus périphérique (cf. fig. II-1). Jusqu'à quatre processeurs peuvent être connectés sur le bus d'accès mémoire : une unité centrale (CPU) et trois processeurs d'entrées/sorties (IOP) ; chacun d'eux accède directement à la mémoire.

Les périphériques rapides à grand débit, disques par exemple, sont connectés aux IOP pour accélérer les échanges avec la mémoire et contrôler leur gestion par des processeurs microprogrammables. Ces processeurs interprètent les programmes canaux construits par le logiciel.

Pour l'implantation d'une structure de communication, il est nécessaire de disposer :

- des microprogrammes d'interprétation sur l'IOP des programmes canaux,
- des programmes canaux situés en mémoire centrale et générés par le logiciel système,
- des tables de communication implantées en mémoire centrale et accessibles à l'unité centrale et à l'IOP,
- du logiciel système pour initialiser les tables et contrôler le déroulement de l'entrée/sortie.

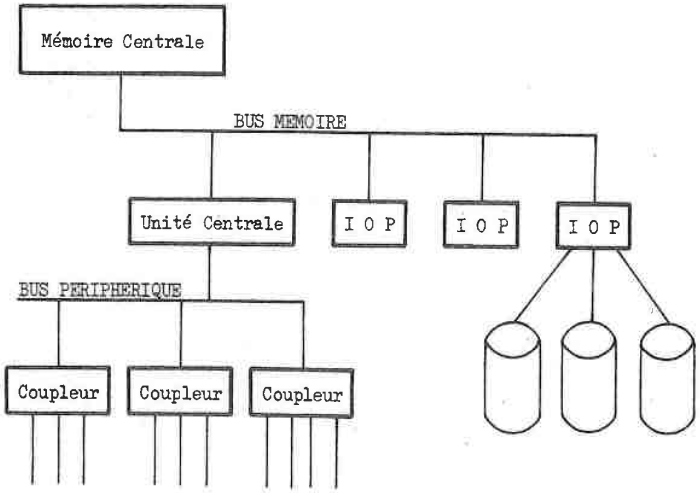


Figure II-1 : Structure du MITRA 125.

1.1.1.2. Structure des voies de communication

La communication entre le processus émettant une requête d'entrée/sortie et l'IOP se fait à travers la mémoire centrale. La synchronisation entre l'unité centrale et l'IOP est assurée par des instructions spéciales et par le système d'interruptions du calculateur.

- Structure matérielle :

Entre les trois unités fonctionnelles : mémoire centrale, unité centrale, IOP, existe un réseau de bus (cf. fig. II-2 et annexe A).

- la mémoire centrale et l'IOP sont reliés par deux bus unidirectionnels :
  + le bus de données (16 bits) sortant de la mémoire et arrivant

dans la tête de canal, interface entre la mémoire centrale et l'IOP,
  + le bus adresse-donnée (20 bits) sortant de la tête de canal et arrivant à la mémoire centrale.

- l'unité centrale et l'IOP sont reliés par un bus fonctionnel. Ce bus est l'assemblage des fils véhiculant l'information de contrôle nécessaire à la synchronisation.

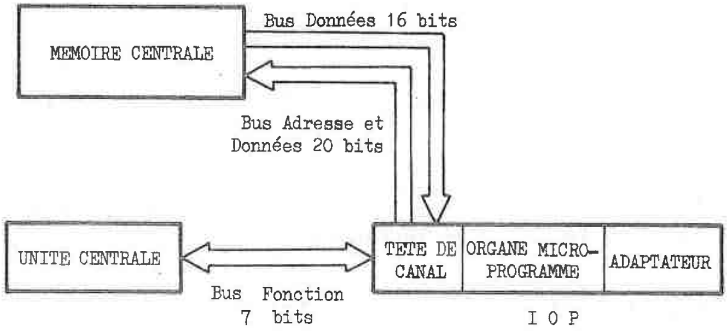


Figure II-2 : Les voies de communication entre la mémoire centrale, l'unité centrale et l'IOP.

- Structure logicielle utilisée pour la communication :

L'IOP est un processeur indépendant fonctionnant en parallèle avec l'unité centrale. Il est chargé d'interpréter les requêtes d'entrée/sortie faites par les tâches s'exécutant sur le calculateur. Ces requêtes sont formulées à l'aide des programmes canaux décrits en annexe A.

Les commandes constituant ces programmes sont situées en mémoire centrale et l'IOP retrouve celles qui lui sont destinées en parcourant une liste de

tables chaînées. L'instruction SIO établit les liens entre la requête du processus et le bloc descripteur de l'unité concernée par l'entrée/sortie. La description des tables utilisées est donnée en annexe A.

1.1.1.3. Mécanisme de communication implanté dans l'unité centrale

Ce mécanisme se compose :

- des instructions de synchronisation entre le calculateur et l'IOP,
- du système de suspension permettant de recevoir les acquittements des requêtes,
- du système d'interruption permettant de recevoir les acquittements des transferts.

- Les instructions de synchronisation :

Au nombre de deux, elles permettent à l'unité centrale d'activer ou d'arrêter l'IOP.

- SIO : (Start Input Output). Cette instruction active l'IOP par l'intermédiaire du bus fonction. On trouvera ci-dessous l'algorithme simplifié des actions effectuées par cette instruction.

- HIO : (Halt Input Output). L'instruction HIO arrête le transfert en cours en positionnant un indicateur propre à l'IOP par l'intermédiaire du bus fonction. L'algorithme de l'interprétation de l'instruction HIO est similaire à celui de l'instruction SIO.

Algorithme SIO :

Début :
% Vérification de la validité des paramètres de l'instruction %
- comparer le numéro du contrôleur IOP au nombre de contrôleurs du système
- tester le mode de fonctionnement et la validité du pointeur vers la table des contrôleurs

% Calcul de l'adresse de l'élément décrivant l'unité %
- rechercher le lien de chaînage dans la table des contrôleurs
- rechercher l'élément dans la table des unités

% Lecture et test des indicateurs de l'unité %
- si unité occupée alors SIO refusée, fin SIO

% Modification des indicateurs et des chaînages %
- positionner l'indicateur "unité occupée"
- verrouiller l'accès aux tables
- modifier la tête de la file des SIO en cours sur cet IOP
- activer l'IOP par le bus fonctionnel (ligne SIO)
- mettre à jour les liens SIO
- déverrouiller l'accès aux tables

% Attente acquittement en provenance de l'IOP
- attendre la suspension de niveau 0
- retour à la phase de récupération (fetch phase)

Fin SIO

Note : La modification des chaînages est exécutée en exclusion mutuelle. En effet, il faut garantir la cohérence des pointeurs des files, ceux-ci pouvant être modifiés par l'unité centrale ou par les IOP.

- Le système de suspension :

Les instructions SIO et HIO attendent un acquittement en provenance de l'IOP. Cet acquittement leur parvient sous la forme d'une suspension prioritaire (suspension de niveau 0). Après réception de l'acquittement, l'unité centrale termine le microprogramme de l'instruction SIO ou HIO et passe à la phase de récupération "fetch phase" pour chercher l'instruction suivante.

- Le système d'interruptions :

Lorsque le transfert est achevé, l'IOP prévient l'unité centrale par une interruption. L'interruption active la tâche immédiate attachée à son niveau qui est chargée de gérer le transfert (handler d'entrée / sortie).

L'excitation de l'interruption est commandée depuis l'IOP par l'intermédiaire du bus fonction.

1.1.2. LA STRUCTURE DU PROCESSEUR D'ENTREE / SORTIE (IOP)

L'IOP est composé de trois parties communiquant par un bus unique (cf. fig. II-3) :

- la tête de canal (TK) assure l'interface entre le processeur d'entrée/sortie, la mémoire centrale et l'unité centrale,
- l'organe microprogrammé (OMP) est chargé d'interpréter les commandes canal,
- l'adaptateur (ADP) gère les périphériques.

Chacune de ces trois parties forme un automate complet, mais l'organe microprogrammé assure la supervision de la tête de canal et de l'adaptateur.

- LA TETE DE CANAL

La partie contrôle de la tête de canal étant inscrite en mémoire morte, elle est inaccessible à l'utilisateur. Notre description se limitera donc à la partie opérative composée de trois ensembles :

- les voies de liaison avec la mémoire,
- les interfaces,
- la gestion du séquencement.

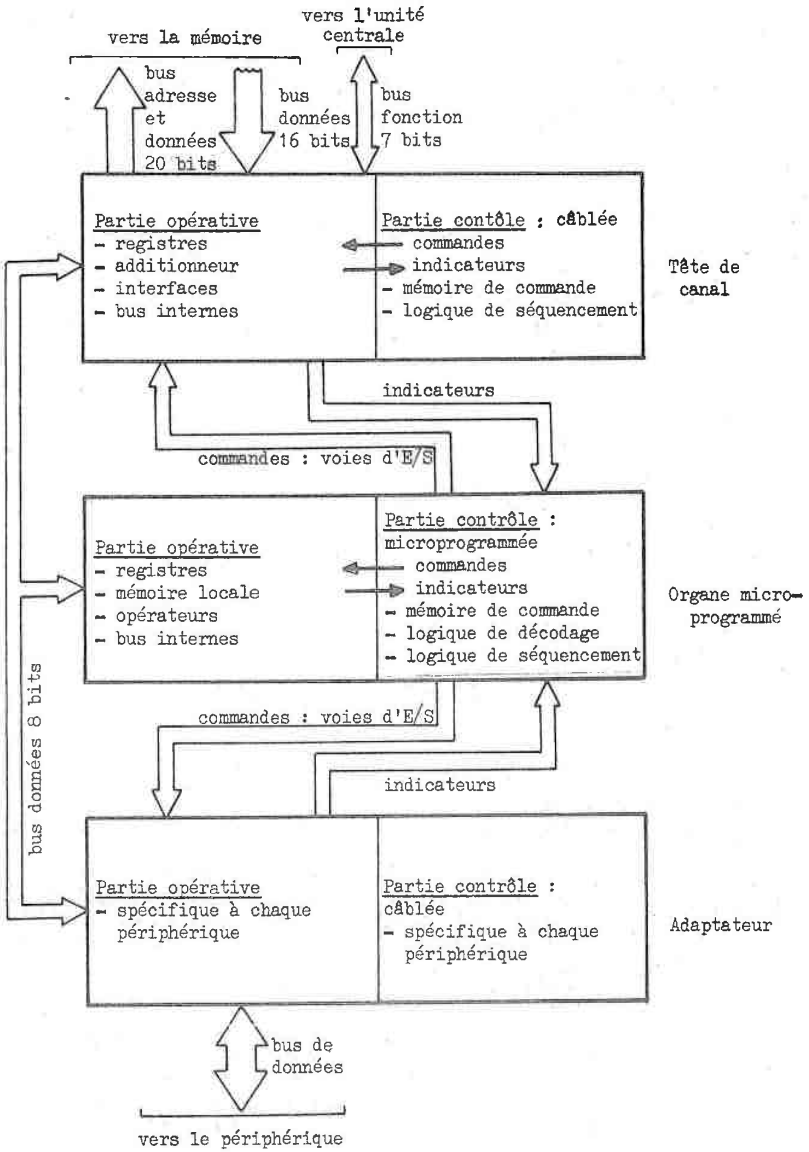


Figure II-3 : Synoptique des liaisons dans l'IOP.

- les voies de liaison sont constituées des registres et des bus par lesquels cheminent les adresses et les données à destination ou en provenance de la mémoire centrale. Pour augmenter leur débit, elles sont unidirectionnelles.

- les interfaces regroupent les fonctions logiques chargées de la communication entre la tête de canal, l'unité centrale et la mémoire centrale. Ces fonctions gèrent les protocoles des liaisons électroniques entre les processeurs reliés.

- la gestion du séquencement est chargée de l'enchaînement des opérations effectuées sur les voies de liaison et les interfaces. Elle est commandée par des ordres codés sur un octet, en provenance de l'organe microprogrammé.

- L'ORGANE MICROPROGRAMME

L'organe microprogrammé a une structure opérative universelle organisée en motif tourbillonnaire. Elle se compose d'un opérateur arithmétique et logique, d'un opérateur de décalage, de registres accumulateurs et d'une mémoire locale. Cette structure est commandée par une partie contrôle microprogrammée à micro-instructions de type court. Les inconvénients liés à ce type de microprogrammation, essentiellement le faible taux de parallélisme et la pauvreté des branchements, sont compensés par la souplesse des combinaisons autorisées par la logique de décodage. En effet, celle-ci est entièrement combinatoire et permet la construction de micro-instructions par l'union logique d'autres micro-instructions. Les performances de l'organe microprogrammé sont élevées - cycle micro-instruction de 160 nanosecondes - et homogènes avec la souplesse de microprogrammation.

- L'ADAPTATEUR

Le rôle de l'adaptateur est d'assurer l'interface entre le périphérique et l'organe microprogrammé. Chaque adaptateur est donc spécifique au type de périphérique couplé et nous ne présenterons pas ici de réalisation particulière.

Note : L'annexe B est consacrée à la description détaillée de l'IOP.

Nous concluons ce paragraphe de présentation du mécanisme standard par une description succincte de son fonctionnement.

Le déroulement d'une opération d'entrée/sortie met en oeuvre trois processeurs :

- le processeur virtuel (MITRA 125) sur lequel s'exécute le logiciel de gestion des entrées/sorties.
  Ce logiciel, appelé handler d'entrées/sorties, construit le programme canal associé à l'échange et émet une requête par l'instruction SIO. Il assurera la surveillance du déroulement de l'entrée/sortie à l'aide du mécanisme de synchronisation composé du système d'interruptions du processeur virtuel.

- la machine réelle (micromachine Q2) sur laquelle sont interprétées des instructions d'entrées/sorties SIO et HIO.
  Les microprogrammes d'interprétation de ces instructions gèrent les tables de communication et activent l'IOP chargé du transfert.

- le processeur d'entrée/sortie (IOP) sur lequel sont interprétés les programmes canaux.

Ce processeur est relié à la machine réelle par un ensemble de lignes véhiculant des signaux fonctionnels et au processeur virtuel par le canal d'accès mémoire et les lignes d'activation du système d'interruption.

Ce mécanisme standard appelle quelques remarques :

- le système d'entrée/sortie décrit est adapté à la gestion d'un ensemble formé de plusieurs unités. La structure de données mise en oeuvre est complexe à manipuler et nécessite de grandes précautions d'utilisation comme la mutuelle exclusion d'accès, afin d'en assurer la cohérence ;
- le système d'entrée/sortie est hybride entre le matériel et le logiciel. Les tables de communication sont modifiées par le logiciel et par les microprogrammes d'interprétation des instructions d'entrée/sortie. Cette solution alourdit l'utilisation du mécanisme d'entrées/sorties hors système.

1.2. LA COMMUNICATION ENTRE TREFLE ET LE MITRA 125

L'analyse du problème posé par la connexion du processeur TREFLE au mini-calculateur hôte MITRA 125 nous a permis de dresser la liste des contraintes que la solution proposée devra respecter :

- la vitesse de transfert doit être très élevée : le chargement et la sauvegarde des 300 millions de caractères du disque ne doit pas immobiliser trop longtemps le processeur TREFLE.
- le mécanisme de communication doit être indépendant du logiciel standard du MITRA 125 et il serait souhaitable que la mise au point du mécanisme puisse se faire hors système.

1.2.1. LES PRINCIPES

Pour résoudre le problème posé par la liaison, en respectant les contraintes énoncées ci-dessus, nous avons fait les choix suivants :

- pour obtenir une vitesse de transfert élevée nous proposons d'assurer les échanges par l'IOP. Cette solution implique le développement d'un adaptateur, organe symétrique à la tête de canal, réalisant un accès à la mémoire du CONTROLEUR ;

- pour assurer une mise en oeuvre simple et pour dissocier la réalisation de la liaison du logiciel standard, nous proposons un mécanisme de communication utilisant une boîte à lettres [Cro 75] et le système d'interruptions du MITRA 125. Activé sur l'IOP, le processus chargé du transfert scrute régulièrement la boîte à lettres. Lorsqu'une requête y a été déposée il effectue l'échange demandé. Les paramètres de cet échange - nombre d'octets à transférer, adresses des mémoires tampon, etc... - sont précisés dans une commande canal associée à la boîte à lettres. Cette solution nous permet de nous détacher des instructions standard d'entrées/sorties avec leur système de tables matérielles-logicielles.

Pour développer cette solution nous avons retenu les principes suivants :

- seul le MITRA 125 peut activer l'IOP en positionnant l'indicateur de demande de transfert dans la boîte à lettres. Ce choix rend le MITRA 125 toujours maître dans ses échanges avec le CONTROLEUR ;

- le CONTROLEUR, après avoir traité la question, doit rendre compte au MITRA 125 de la manière dont s'est déroulée le traitement. Ce dernier demandera le transfert des résultats stockés dans la mémoire du MODULE DE CLASSEMENT vers la mémoire centrale. Le compte-rendu sera envoyé en parallèle au MITRA 125 par les lignes d'interruption externes, et à l'IOP par les indicateurs internes à l'adaptateur ;

- l'IOP signale la fin du transfert aux autres processeurs par des lignes d'interruptions. Ce mécanisme permet d'activer les processus en attente de la fin du transfert

Nous concluons ce paragraphe en résumant schématiquement le fonctionnement du mécanisme de communication mis en oeuvre :

- première étape : le MITRA 125 initialise l'échange en positionnant l'indicateur de demande de transfert dans la boîte à lettres ;

- seconde étape : l'IOP scrutant régulièrement la boîte à lettres y trouve la demande de transfert ; il la prend en charge et il effectue l'échange précisé par la commande canal.

Remarques : . le MITRA 125 initialise tous les transferts
. l'IOP est maître des échanges
. le CONTROLEUR doit activer le MITRA 125 pour obtenir un transfert.

1.2.2. LA CONNEXION MATERIELLE

Dans le cadre de la liaison entre le MITRA 125 et le CONTROLEUR du processeur TREFLE, l'adaptateur que nous développons pour l'IOP réalise un accès direct à la mémoire (ADM) du CONTROLEUR. Cette solution a l'avantage de ne pas faire intervenir le CONTROLEUR dans la communication, car il ne ferait que ralentir la vitesse de transfert.

La partie opérative de l'adaptateur est composée de quatre voies véhiculant les informations. Ces voies sont chargées de transformer le chemin interne de l'IOP, de largeur 8 bits, en chemins externes de largeur 16 bits :

- la voie des adresses vers la mémoire du CONTROLEUR comprenant un bus et deux registres,
- la voie des données vers la mémoire du CONTROLEUR comprenant un bus,

deux registres et un multiplexeur,
- la voie des données venant de la mémoire du CONTROLEUR comprenant un bus, deux registres et un multiplexeur,
- la voie des informations de contrôle véhiculant les interruptions et les indicateurs.

La partie contrôle de l'adaptateur est réduite à un décodeur réalisant les combinaisons logiques entre les signaux de commandes issus de l'organe microprogrammé. Les signaux de contrôle issus de ces combinaisons sont chargés de commander la partie opérative de l'adaptateur - registres et voies -, de gérer les accès à la mémoire du CONTROLEUR et de transmettre les signaux de synchronisation vers le CONTROLEUR.

Note : La description détaillée de la structure matérielle et de son contrôle est donnée en annexe C.

La figure II-4 résume l'ensemble des liaisons existant entre le MITRA 125 et le CONTROLEUR (cf. page II-16).

1.2.3. LE PROTOCOLE LOGICIEL

Nous regroupons les processus qui coopèrent pour la communication entre TREFLE et le MITRA 125 en trois ensembles :

- les precessus activés sur le MITRA 125 qui initialisent et contrôlent le déroulement des échanges :

+ le processus de prise en compte d'une demande de transfert,
+ le processus d'initialisation du transfert,
+ le processus de contrôle de fin de transfert.

- le processus activé sur l'IOP qui scrute la boîte à lettres et effectue le transfert demandé :

Figure II-4 : Synoptique des liaisons entre TREFLE et MITRA 125.

Remarques : - les liaisons entre le MITRA 125 et l'IOP sont les liaisons standard dont seules certaines possibilités (SIO, HIO, etc...) sont inutilisées ;

- les liaisons entre l'IOP et le CONTROLEUR de la machine TREFLE sont totalement gérées par l'adaptateur spécialement conçu pour cette connexion ;
- la liaison entre le CONTROLEUR et le MITRA 125 utilise le système des interruptions externes de l'unité centrale.

+ le processus d'interprétation des commandes de transfert.

- les processus activés sur le CONTROLEUR qui demandent et vérifient l'échange :

+ le processus de demande de transfert,
+ le processus de contrôle de fin de transfert.

Les procédures décrites dans la suite de ce paragraphe sont exécutées par les processus énumérés ci-dessus.

1.2.3.1. Procédure d'initialisation du transfert

But : Cette procédure construit le programme canal décrivant le transfert et met à jour l'indicateur de demande d'échange. Elle est activée soit par le système de gestion de messages qui reçoit les questions, soit par l'opérateur qui désire charger la base documentaire sur le disque, soit sur demande du processeur TREFLE lorsqu'il a terminé son traitement.

Structure de données utilisée :

- la boîte à lettres : elle est rangée dans l'entrée réservée à l'IOP de connexion dans la table des contrôleurs,
- le programme canal : il est construit dans la zone mémoire de l'utilisateur et son adresse est rangée dans la boîte à lettres.

Ces deux structures sont décrites en annexe C.

Algorithme :

Cette procédure reçoit en paramètre : le code opération, le chaînage, les adresses des tampons et leur taille. Elle construit alors le programme canal et met à jour la boîte à lettres avec l'adresse de la première com-

mande canal puis positionne l'indicateur de demande de transfert.

1.2.3.2. Procédure de transfert

But : Cette procédure scrute régulièrement l'indicateur de transfert. Lorsque celui-ci indique qu'un échange a été initialisé par une tâche activée sur le MITRA 125, la procédure charge dans sa mémoire de travail le programme canal trouvé grâce aux pointeurs. Elle analyse le code opération et effectue les transferts demandés. En fin de transfert elle remplit le mot de compte-rendu et envoie une interruption de fin de transfert.

Structure de données utilisées :

La procédure de transfert est exécutée par l'IOP. Elle utilise les structures de données précédemment définies et les recopie dans son espace de travail formé des registres rapides de la mémoire locale de l'organe microprogrammé. Les conventions sont précisées en annexe C.

Algorithme :

- Remarques préliminaires :
  + l'algorithme d'interprétation des commandes canal étant microprogrammé, certaines notions matérielles y interviennent - coupure de secteur, interruption, etc... -, et sa présentation mettra en évidence les contraintes physiques rencontrées ;
  + le point d'entrée - étiquette DEBUT - est accédé par l'apparition du signal d'initialisation - bouton RAZ du panneau de commande MITRA 125 - et par la détection d'une coupure secteur. Cet algorithme est une boucle perpétuelle.

- Description :

DEBUT :

1. si coupure secteur alors attente : allera attente
   % la boucle s'arrête lorsque le secteur est coupé %
2. Temporisation % légère attente pour décharger le bus mémoire %
3. Transfert de la boîte à lettres dans les registres de la mémoire locale
4. si demande de transfert = 0 alors allera 2
5. Transfert de la commande canal dans les registres de la mémoire locale
6. Initialisation des registres de la Tête de Canal et de l'Adaptateur
7. si code ordre de la commande canal = lecture
   alors transfert CONTROLEUR ⟶ MEMOIRE
   sinon transfert MEMOIRE ⟶ CONTROLEUR
8. si chaînage commande canal = 1
   alors calcul de l'adresse de la commande canal suivante
   allera 5
9. Emission d'une interruption de fin de transfert vers le CONTROLEUR et le MITRA 125
10. Attente fin de traitement TREFLE
    % boucle de scrutation des indicateurs en provenance du CONTROLEUR %
11. Transfert du mot de compte-rendu
12. allera 2.

1.2.3.3. Procédures annexes

Nous regroupons dans ce paragraphe la description sommaire des autres procédures faisant partie du mécanisme de communication.

- La procédure de demande de transfert :

Activée sur le contrôleur, elle permet à celui-ci de demander le transfert vers le MITRA 125 des informations que le processeur TREFLE a élaborées. Ces informations sont de deux types :

- les données formant la réponse à la question posée par l'utilisateur,
- le compte-rendu précisant au logiciel les conditions dans lesquelles s'est déroulé le traitement.

En effet, le traitement du processeur TREFLE peut se terminer de quatre manières différentes :

- fin normale : le domaine des données a été exploré, la réponse se trouvant dans le module de classement est complète, le CONTROLEUR en demande le transfert vers l'utilisateur ;

- interruption de traitement : le processeur TREFLE fonctionne en mode de recherche exacte, le domaine des données n'est pas encore complètement exploré et déjà la mémoire du MODULE DE CLASSEMENT est pleine. Le CONTROLEUR demande le vidage de la mémoire pour poursuivre les recherches ;

- erreur matérielle : le FORMATTEUR, le MODULE DE CLASSEMENT ou le CONTROLEUR ont détecté une erreur matérielle : erreur d'entrée/sortie sur le disque, erreur de positionnement du bras, panne matérielle, etc... Le CONTROLEUR demande le transfert du compte-rendu vers le logiciel du MITRA 125 ;

- erreur logicielle : le contrôleur a détecté une erreur dans la question ou dans un module logiciel du système. Il demande le transfert du compte-rendu vers le logiciel du MITRA 125.

Chaque fin de traitement nécessite un transfert pour permettre soit la poursuite du traitement, soit l'analyse de la cause de l'arrêt. Le rôle de la procédure de demande de transfert sera de positionner les indicateurs surveillés par l'IOP et d'émettre les interruptions externes vers le MITRA 125. Le micrologiciel de l'organe microprogrammé et le logiciel de la machine hôte se chargeront de l'analyse des indicateurs, de la réception des interruptions et de l'analyse du compte-rendu.

- Les procédures de prise en compte des demandes de transfert :

Au nombre de quatre, elles sont activées par les interruptions externes en provenance du contrôleur. Elles correspondent chacune à une fin particulière du traitement effectué par le processeur TREFLE et ont pour rôle de préparer les paramètres nécessaires au transfert. Ces paramètres sont déduits de l'analyse du compte-rendu puis ces procédures activent le processus d'initialisation du transfert, décrit précédemment.

- Les procédures de contrôle de fin de transfert :

A chaque fin de transfert l'IOP prévient le CONTROLEUR et le MITRA 125 par l'émission d'une interruption, qu'il vient d'achever l'échange d'informations. Ce signal est pris en compte par les procédures de contrôle de transfert, activées sur le MITRA 125 et sur le CONTROLEUR. Ces procédures vérifient le bon déroulement de l'échange et autorisent la poursuite du traitement des processus en attente de la fin du transfert.

1.2.3.4. Coopération des processus

Dans ce paragraphe nous présentons le mécanisme de synchronisation des différents processus décrits précédemment en déroulant sur un exemple le protocole défini.

Cet exemple met en scène l'utilisateur qui pose une question et attend la réponse, le MITRA 125 qui gère les messages, l'IOP qui assure la communication avec TREFLE et le CONTROLEUR qui en supervise le fonctionnement.

1. L'utilisateur pose une question et se met en attente de la réponse.
2. Le système de gestion des messages du MITRA 125 reçoit la question, l'analyse puis appelle les procédures du protocole.
3. Ces procédures initialisent le transfert en construisant le programme canal, déposent la demande dans la boîte à lettres puis se mettent en attente.
4. L'IOP prend en compte la demande et effectue le transfert de la question vers le CONTROLEUR.
5. Il prévient les processus en attente sur le MITRA 125 et le CONTROLEUR de la fin du transfert ; ces derniers vérifient le bon déroulement de l'échange.
6. Les processus actifs sur le MITRA 125 et l'IOP se mettent en attente de la fin du traitement du processeur TREFLE.
7. Le CONTROLEUR compile la question puis initialise et active les autres processeurs de TREFLE pour traiter la requête.
8. En fin de traitement il positionne les indicateurs vers l'IOP et envoie les interruptions au MITRA 125 pour demander le transfert de la réponse.
9. L'IOP transfère le mot compte-rendu vers le MITRA 125 et se met en attente de demande de transfert (scrutation cyclique de la boîte à lettres).
10. Les procédures du protocole de communication activées sur le MITRA 125 initialisent le transfert, déposent la demande dans la boîte à lettres et se mettent en attente.

11. L'IOP prend en compte la demande d'échange et effectue le transfert de la réponse depuis le CONTROLEUR vers le MITRA 125.
12. En fin de transfert il prévient le CONTROLEUR et le MITRA 125 qui vérifient le bon déroulement de l'échange, puis il se remet en attente de demande de transfert.
13. Le MITRA 125 renvoie, par son système de gestion de messages, la réponse à l'utilisateur.

Le schéma de la figure II-5 illustre cet exemple en faisant ressortir le parallélisme dans le déroulement des processus.

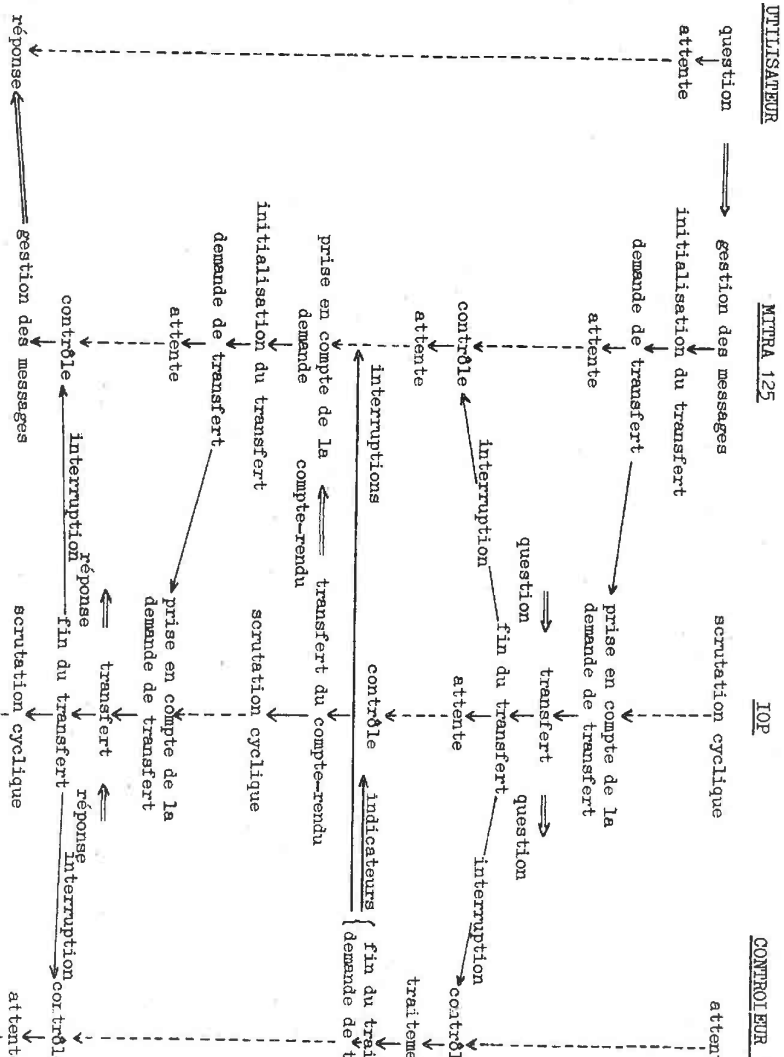


Figure II-5 : La synchronisation des processus de communication.

II - LA LIAISON ENTRE TREFLE ET LE DISQUE

Dans le cadre du prototype que nous développons, les données consultées par le processeur TREFLE se trouvent rangées sur un disque magnétique. Ce support a été choisi pour son accès aléatoire, son débit élevé (1,2 million d'octets par seconde) et sa grande capacité (300 millions d'octets). Le but de la seconde partie de ce chapitre est de présenter la solution choisie pour relier le disque au processeur TREFLE par l'intermédiaire du FORMATTEUR. Après la présentation de la démarche suivie pour la conception du processeur de formattage, nous décrivons la structure de données implantée sur le disque et les algorithmes des fonctions qui la gèrent.

2.1. LES PRINCIPES

Interface entre le disque et le coeur du processeur TREFLE, le FORMATTEUR interprète les ordres reçus du CONTROLEUR et achemine les données lues vers leurs destinataires : CONTROLEUR, FILTRE, MODULE DE CLASSEMENT. Contrairement à l'IOP qui a une structure universelle, le FORMATTEUR est très adapté à son rôle d'interface.

Cette adaptation est le résultat de la démarche suivie pour sa conception :

- celle-ci a commencé par une étude approfondie du fonctionnement de l'unité de disque sélectionnée pour la connexion. La description de cette unité est donnée en annexe D ;

- le premier choix effectué ensuite a porté sur la structure d'informations implantée sur le support des données. Cette structure fait appel à des notions physiques (secteur, piste, etc...) et à des notions logiques (mots-clés, inscriptions, etc...) ;

- la troisième phase de l'étude a permis de définir l'ensemble des fonctions opérant sur la structure de données. Ces fonctions ont été décrites sous la forme d'algorithmes ;

- la quatrième étape a consisté à concevoir une structure matérielle se prêtant bien à l'interprétation des fonctions. Il en est résulté une structure de machine adaptée au problème posé par l'interface. Les algorithmes seront réalisés à l'aide de microprogrammes interprétés par la structure matérielle ;

- cette démarche se termine par la vérification de l'adéquation du processeur au problème. La microprogrammation des algorithmes dans le langage de micro-instructions permet de valider la structure matérielle définie au cours des étapes antérieures.

Remarque : Cette démarche se justifie par la technologie électronique actuelle qui met à la disposition de l'architecte de machines un ensemble d'outils (macrocomposants, microprocesseurs en tranche, etc...) lui permettant une approche de son problème plutôt logique que physique.

Cette deuxième partie présentera les résultats obtenus au cours des étapes successives de la conception :

- la structure de données,
- les algorithmes,
- la structure du FORMATTEUR.

Note : L'interface entre le disque et le processeur TREFLE est composé de deux parties : la gestion du déplacement du bras et la gestion des données. Le CONTROLEUR, pourvu d'un système d'interruptions rapide, prend en charge le pilotage du bras ; le FORMATTEUR avec son cycle très court assure le transfert des informations. Ce découpage fonctionnel rend le formatteur esclave du contrôleur qui lui envoie les ordres à interpréter (cf. fig. II-1).

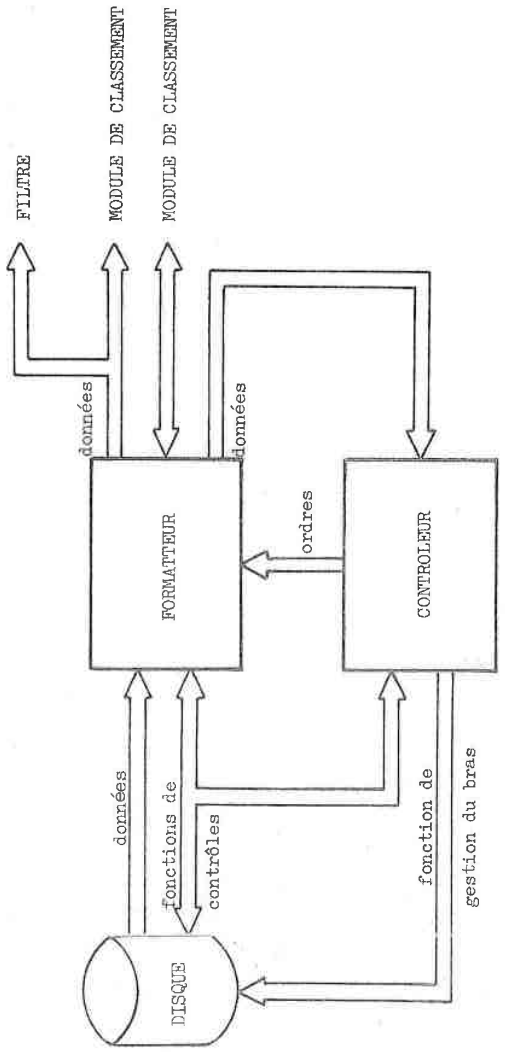


Figure II-1 : Les liaisons entre le disque et le processeur TREFLE.

2.2. LA STRUCTURE DE DONNEES IMPLANTEE SUR LE DISQUE

Le rôle de la structure de données que nous décrivons est de mettre en relation les notions physiques liées à la technologie du disque et les notions logiques liées à la méthode d'accès implantée dans le FORMATTEUR et le CONTROLEUR.

Les contraintes physiques concernent essentiellement le découpage des pistes en secteurs et des secteurs en champs. En effet, chaque piste doit contenir un nombre fixe de secteurs de longueur constante et chaque champ est séparé du suivant par une zone réservée aux signaux de synchronisation, le nombre de champs pouvant varier d'un secteur à l'autre.

Nous considérons deux formats de secteur différenciés par leur découpage (cf. fig. II-3) :

- le secteur "tête de piste",
- le secteur "courant".

Il y a un secteur tête de piste en début de chaque piste, les autres secteurs étant du type courant.

Les champs du secteur tête de piste sont les suivants :

1. champ d'identification (entête)
2. champ de validité de piste
3. champ de clés
4. champ de données.

Les champs du secteur courant sont les suivants :

1. champ d'identification
3. champ de clés
4. champ de données.

Remarque : chaque champ commence par deux zones liées à la technologie du support :
- une zone de synchronisation pour les signaux de lecture ou d'écriture,
- une marque de début de champ utilisée pour le démarrage de l'horloge.

2.2.1. LE CHAMP D'IDENTIFICATION

Ce champ contient quatre zones :

- il commence par une zone de synchronisation qui permet le verrouillage des signaux d'horloge de lecture et d'écriture,

- un octet de marque de début de champ permet de détecter trois évènements :
  + le début d'un secteur
  + le début d'un secteur invalide
  + le début d'une piste invalide,

- une zone de trois octets contient l'adresse du secteur,

- une zone de deux octets contient le code de détection d'erreurs CRC (Cyclic Redundancy Code). La description précise de la méthode de calcul de ces deux octets est donnée en annexe E.

2.2.2. LE CHAMP DE VALIDITE DE PISTE

Ce champ n'existe que dans le secteur tête de piste, il contient :

- une zone de synchronisation,
- une marque de début de champ,
- le vecteur des secteurs non valides : ce vecteur est composé d'une suite de n bits représentant les n secteurs de la piste avec la convention suivante :
  le $i^{ème}$ bit est positionné si le $i^{ème}$ secteur est non valide,
- une zone de deux octets contenant le code de vérification CRC.

2.2.3. LE CHAMP DE CLES

Ce champ est divisé en deux sous-champs :

- un sous-champ longueur,
- un sous-champ clé.

Le sous-champ longueur contient :

- une zone de deux octets qui précisent la longueur de la zone de clés,
- une zone de deux octets qui précisent la longueur de la zone de données.

Le sous-champ clé contient :

- une zone, de longueur précisée dans le sous-champ longueur, contenant les mots-clés permettant la recherche sur un ou plusieurs critères.

Ces deux sous-champs sont contigüs et précédés par :

- une zone de synchronisation,
- une marque de début de champ.

Ils sont suivis par :

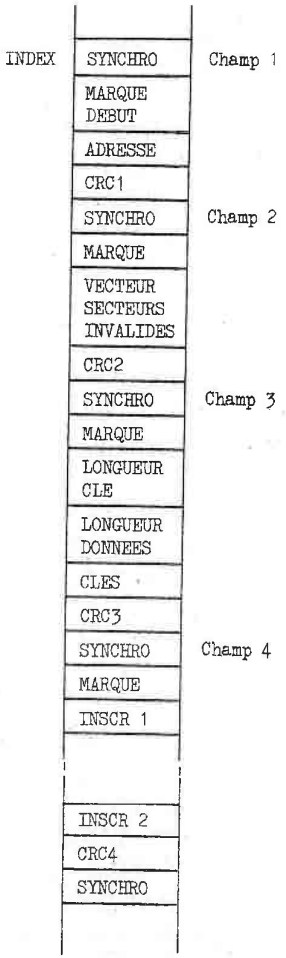
- une zone contenant les deux octets de vérification CRC.

2.2.4. LE CHAMP DE DONNEES

Ce champ contient :

- une zone de synchronisation,
- une marque de début de champ,
- une zone, de longueur précisée dans le sous-champ longueur du champ clé, contenant les données regroupées sous la forme d'inscriptions permettant la mémorisation des descripteurs des documents de la base,
- une zone contenant les deux octets de vérification CRC.

Format du secteur entête de piste



Format du secteur courant

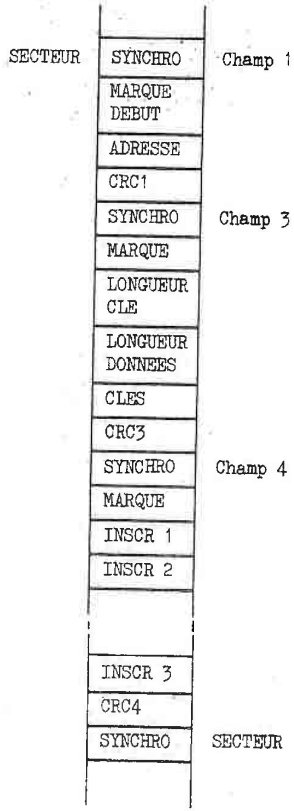


Figure II-3 : Format des secteurs sur le disque.

2.3. LES ALGORITHMES DES FONCTIONS OPERANT SUR LA STRUCTURE DE DONNEES

Dans ce paragraphe nous décrivons les ordres interprétés par le FORMATTEUR, leur fonction étant présentée sous la forme d'algorithme.

Nous classons les fonctions réalisées en deux groupes :

- les fonctions d'écriture :

+ écriture de l'entête d'un secteur (champ 1) - ordre EE,
+ écriture du vecteur de secteurs invalides (champ 2 du secteur en tête de piste) - ordre EVS,
+ écriture des clés et des données (champs 3 et 4) - ordre ECD,
+ écriture de l'entête d'un secteur suivie de zéros (commande de formattage du disque) - ordre EEO ;

- les fonctions de lecture :

+ lecture du vecteur de secteurs invalides (champ 2 du secteur en tête de piste) - ordre LVS,
+ lecture des clés et des données (champs 3 et 4) - ordre LCD,
+ lecture de l'entête suivie de zéros (commande de vérification du formattage) - ordre LEO.

Remarques : . chaque ordre ne concerne qu'un seul secteur,
. si un ordre ne traite que certains champs du secteur, la lecture devra néanmoins commencer au début du secteur.

Note : Les contraintes technologiques que nous avons fait apparaître dans ces algorithmes sont essentiellement liées à la logique de gestion du disque. Elles seront prises en compte lors de la conception du FORMATTEUR qui doit assurer efficacement l'interprétation de ces fonctions.

2.3.1. ALGORITHME DE LA FONCTION D'ECRITURE DE L'ENTETE D'UN SECTEUR - ORDRE EE

L'algorithme de cette fonction est commun à l'ordre EE et à l'ordre EEO. Nous en faisons une procédure que nous appelons "Ecriture Champ Entete".

ECE :

1. Envoi de l'ordre d'écriture au disque
2. Attente de 224 périodes de l'horloge de service
3. Ecriture de l'octet "début de secteur"
4. Demande des octets d'adresse
   - sérialisation puis écriture de ces octets
   - comptage de ces octets
   - calcul du CRC sur ces octets
5. Ecriture des octets de CRC.

La fonction Ecriture de l'Entête d'un secteur s'écrira :

EE :

1. Appeler ECE
2. Désactiver l'ordre d'écriture, fin de l'ordre EE.

2.3.2. ALGORITHME DE LA FONCTION DE FORMATTAGE - ORDRE EEO

1. Appeler ECE
2. Ecriture de zéros jusqu'à la fin du secteur
3. Désactiver l'ordre d'écriture, fin de l'ordre EEO.

2.3.3. ALGORITHME DE LA FONCTION D'ECRITURE DU VECTEUR DES SECTEURS INVALIDES - ORDRE EVS

Avant de commencer l'écriture de ce champ il nous faut lire l'adresse du secteur et vérifier son égalité avec les registres "adresse courante".

Nous appelons cette séquence "Lecture Entête" et nous la décrivons séparément pour l'utiliser comme procédure dans les autres algorithmes.

Lecture Entête - LE :

1. Attente de 58 périodes de l'horloge de service
2. Envoi de l'ordre de lecture au disque
3. Attente de 88 périodes de l'horloge de service
4. Lancer la détection de la marque de début de secteur
5. Lecture et analyse de la marque de début de secteur :
6. si secteur invalide alors activer l'interruption SI du contrôleur, désactiver la lecture, terminer l'ordre en cours

   sinon
7. si piste invalide alors activer l'interruption PI du contrôleur, désactiver la lecture, terminer l'ordre en cours

   sinon
8. Activer le générateur de CRC
9. Lecture des octets d'adresse
   - parallélisation des bits lus
   - comptage des octets
   - calcul du CRC sur ces octets
10. Comparer l'adresse lue avec celle stockée dans les registres "Adresse Courante"
11. si elles ne sont pas égales alors activer l'interruption ERAD du contrôleur, désactiver la lecture, terminer l'ordre en cours

    sinon
12. si le résultat du calcul de CRC est différent de zéro alors activer l'interruption CRCER du contrôleur, désactiver la lecture, terminer l'ordre en cours

    sinon fin procédure LE - retour.

Ecriture du vecteur de secteurs invalides - EVS :

1. Appeler LE
2. Désactiver la lecture et attente de 8 périodes de l'horloge de service
3. Activer l'écriture
4. Attente de 88 périodes de l'horloge de service
5. Ecriture de l'octet "Début de champ"
6. Demande au contrôleur des octets composant le vecteur de secteurs invalides
   - sérialisation de ces octets
   - comptage de ces octets
   - calcul du CRC sur ces octets
7. Ecriture des octets de CRC
8. Désactiver l'ordre d'écriture, fin de l'ordre EVS.

2.3.4. ALGORITHME DE LA FONCTION D'ECRITURE DES CHAMPS-CLES ET DONNEES - ORDRE ECD

1. Appeler LE
2. si on traite le secteur zéro alors - attente de 96 périodes de l'horloge de lecture
   - attente de n périodes de l'horloge de lecture, n étant la longueur du champ 2
3. Désactiver la commande de lecture et attente de 8 périodes de l'horloge de service
4. Activer la commande d'écriture
5. Attente de 88 périodes de l'horloge de service
6. Ecriture de l'octet "Début de champ"
7. Demande au contrôleur des octets de longueur de clé et de donnée suivis des octets de clé
   - sérialisation de ces octets
   - calcul du CRC sur ces octets

   - rangement des octets de longueur dans des compteurs spécialisés.

   Pendant l'écriture des octets de clé, décrémenter le compteur de longueur de clé et détecter la fin du champ 3
8. A la fin du champ de clés écrire les 2 octets de CRC
9. Attente de 96 périodes de l'horloge de service
10. Ecriture de l'octet "Début de champ"
11. Demande au contrôleur des octets de donnée à écrire
    - sérialisation de ces octets
    - calcul du CRC sur ces octets.

    Pendant l'écriture des octets de donnée, décrémenter le compteur de longueur de donnée et détecter la fin du champ 4
12. A la fin du champ de données écrire les 2 octets de CRC
13. Désactiver la commande d'écriture, fin de l'ordre ECD.

2.3.5. ALGORITHME DE LA FONCTION DE LECTURE DU VECTEUR DES SECTEURS INVALIDES - ORDRE LVS

1. Appeler LE
2. Attente de 96 périodes de l'horloge de lecture
3. Lancer la détection de la marque de début de champ
4. Arrivée de la marque de début de champ
   - lecture des octets du vecteur de secteur invalide
   - calcul du CRC sur ces octets
   - comptage de ces octets
   - envoi des octets au contrôleur
5. A la fin du champ vérifier le CRC calculé

   si CRC est différent de zéro alors activer l'interruption CRCER du contrôleur
6. Désactiver la commande de lecture, fin de l'ordre LVS.

2.3.6. ALGORITHME DE LA FONCTION DE LECTURE DU CHAMP-CLE - ORDRE LC

L'algorithme de cette fonction est commun à l'ordre LC et à l'ordre LCD, nous en faisons donc une procédure appelée "Lecture Champ-Clé".

LCC :

1. si on traite le secteur zéro alors - attente de 96 périodes de l'horloge de lecture
   - attente de n périodes de l'horloge, n étant la longueur du champ 2
2. Attente de 96 périodes de l'horloge de lecture
3. Lancer la détection de la marque de début de champ
4. Arrivée de la marque de début de champ
   - lecture des octets du champ-clé
   - calcul du CRC sur ces octets
   - comptage de ces octets
5. Ranger les octets "longueur de clé" dans les compteurs spécialisés
6. Ranger les octets "longueur de données" dans les compteurs spécialisés
7. Envoyer les octets vers le contrôleur en décrémentant le compteur "longueur de clé"
8. En fin de champ-clé vérifier le CRC calculé
   si CRC est différent de zéro alors activer l'interruption CRCER du contrôleur

Lecture des clés - ordre LC :

1. Appeler LE
2. Appeler LCC
3. Désactiver la commande de lecture, fin de l'ordre LC.

2.3.7. ALGORITHME DE LA FONCTION DE LECTURE DES CLES ET DES DONNEES - ORDRE LCD

1. Appeler LE
2. Appeler LCC
3. Attente de 96 périodes de l'horloge de lecture
4. Lancer la détection de la marque de début de champ
5. Arrivée de la marque de début de champ
   - lecture des octets du champ donnée
   - calcul du CRC sur ces octets
   - comptage de ces octets en décrémentant le compteur "longueur de données"
   - envoi de ces octets dans les mémoires d'entrée du FILTRE et du MODULE DE CLASSEMENT
6. En fin de champ donnée vérifier le CRC calculé
   si CRC est différent de zéro alors activer l'interruption CRCER du contrôleur
7. Désactiver la commande de lecture, fin de l'ordre LCD.

2.3.8. ALGORITHME DE LA FONCTION DE VERIFICATION DU FORMATTAGE - ORDRE LEO

1. Appeler LE
2. Vérifier si les octets qui suivent sont égaux à zéro - jusqu'à la fin du secteur
3. Désactiver la commande de lecture, fin de l'ordre EEO.

Remarques : Nous concluons ce paragraphe par quelques remarques sur ces algorithmes :

- la notion de procédure apparaît fréquemment dans la description de ces algorithmes ; il sera très intéressant que la logique de séquencement du FORMATTEUR accepte les appels de sous-programmes comme type de branche-

ment dans sa mémoire de commande ;

- la description séquentielle de ces algorithmes masque les nombreuses boucles nécessaires à la réalisation des lectures, des écritures et des attentes. Il serait là aussi très intéressant que la logique de séquencement offre un mécanisme de bouclage simple permettant une grande efficacité et une importante économie de place en mémoire ;

- le grand nombre d'actions à effectuer simultanément dans une même étape de l'algorithme nous oriente vers une microprogrammation horizontale maximisant le parallélisme.

2.4. LA STRUCTURE DU FORMATTEUR [Tus 78]

L'inventaire des ressources nécessaires à l'interprétation des algorithmes précédemment décrits nous permet d'établir la liste des organes opératifs et des fonctions de contrôle.

La partie opérative du FORMATTEUR :

- un registre à décalage effectuant la sérialisation et la parallélisation des données pour leur écriture et leur lecture sur le disque,
- une logique de génération du CRC,
- un ensemble de compteurs spécialisés. Ils permettront le comptage des octets de clé et de données - champs 3 et 4 - des cycles d'attente, etc...
- une logique de détection d'erreur composée de comparateurs pour la vérification des adresses, du champ CRC, etc... ,
- une logique de gestion d'horloge car le FORMATTEUR peut avoir son fonctionnement cadencé par deux horloges : l'horloge de lecture et l'horloge de service,
- une logique de gestion de bus pour gérer les échanges avec les autres processeurs de la machine TREFLE.

La partie contrôle :

- une mémoire de commande contenant les micro-instructions longues (39 bits),
- une logique de séquencement acceptant les appels de sous-programmes, les bouclages sur une micro-instruction, les branchements simples et les éclatements sur le codage de l'ordre.

Note : La structure du FORMATTEUR est détaillée en annexe E.

III - LE SYSTEME DE PRODUCTION DE LOGICIEL

Dans cette troisième partie nous décrivons les outils utilisés pour la mise en oeuvre des microprogrammes développés. En général, la grande pauvreté des outils d'aide à la microprogrammation augmente considérablement la difficulté à réaliser les algorithmes écrits. En effet, le microprogrammeur ne peut se satisfaire d'un simple assembleur traduisant le texte source en texte binaire, mais il est nécessaire qu'il dispose d'autres outils plus sophistiqués tels que :

- des implanteurs de microprogrammes en mémoire qui assurent la gestion des adresses de branchement,
- des simulateurs fonctionnels qui permettent la mise au point rapide des algorithmes,
- des émulateurs matériels qui facilitent la mise au point des microprogrammes dans l'environnement réel [Gre 66], [Hus 70], [Guy 75].

3.1. LES PRINCIPES

Nous présentons dans ce paragraphe les problèmes spécifiques à la microprogrammation et la méthode adoptée par l'équipe pour les résoudre.

3.1.1. LES PRINCIPALES CONTRAINTES

Les caractéristiques des microprogrammes placent leurs auteurs dans une catégorie très particulière d'usagers de l'informatique : en général les microprogrammes sont courts, ils ne sont jamais modifiés et l'ordinateur n'est utilisé que pour la phase de compilation et de mise au point.

Le parallélisme dans les actions et la complexité des branchements engendrent une forme de programmation pour laquelle les langages algorithmiques sont peu adaptés. Ce phénomène explique l'utilisation fréquente de micro-assembleurs par les microprogrammeurs.

3.1.2. LES CHOIX

Le principe fondamental que nous appliquons est le développement de nos outils de production de micrologiciel sous un système interactif. Le choix du système APL composé d'un interprète du langage, d'un bibliothécaire et d'un éditeur de texte s'est rapidement imposé à l'équipe pour les facilités qu'il offre pour l'écriture et la mise au point des programmes [Rob 75].

Un second principe est celui d'utiliser pour chaque processeur un langage unique :

- comme langage source de l'assembleur,
- comme langage à interpréter par le simulateur fonctionnel.

Ainsi le passage de la simulation à l'assemblage est immédiat et le simulateur disposera d'un langage d'entrée plus synthétique que le code binaire généré par l'assembleur.

3.1.3. LA METHODE

Nous réalisons les micro-assembleurs et les simulateurs nécessaires au développement de notre projet selon une méthode originale issue de l'équipe [Roh 79] et qui, développée par un industriel, a permis l'élaboration d'un produit commercialisé nommé APL 2M [Rou 78]. Cette méthode s'inspire des constatations suivantes :

- on peut assimiler un assembleur à un programme qui a pour but de générer du code objet en fonction de paramètres donnés par le texte source. De même un simulateur fonctionnel est un programme qui applique les fonctions définies par le texte source à une structure de données représentant la machine. Une utilisation de cette idée revient à choisir comme langage source de l'assembleur et du simulateur un langage dont la syntaxe est déjà reconnue par un compilateur, et à écrire dans ce langage les fonctions de génération du code objet ou les fonctions de simulation de la machine. L'assemblage ou la simulation d'un programme consistera à compiler et à exécuter ce programme écrit en langage évolué.

  Exemple :
  L'instruction LOAD R1,A peut être interprétée comme l'appel d'une fonction LOAD ayant deux arguments R1 et A.
  Cette fonction
  - pour l'assembleur : génère un vecteur binaire correspondant au codage de l'instruction LOAD avec ses arguments
  - pour la simulation : applique la fonction de chargement (LOAD) à la structure de données représentant la machine ;

- nous avons choisi comme langage évolué APL pour sa puissance de traitement et son système de développement de programmes, mais PASCAL ou FORTRAN auraient pu convenir. L'exemple ci-dessous compare la syntaxe d'APL à celle de PASCAL.

LOAD R1,A : appel de la fonction APL LOAD avec un argument vecteur composé de la concaténation des deux variables R1 et A,

LOAD(R1,A); : appel de la procédure PASCAL LOAD avec deux arguments R1 et A.

Remarque : Le langage source commun nous permet de n'écrire qu'un seul logiciel : assembleur-simulateur paramétré.

3.2. LES REALISATIONS

Dans le cadre de ses travaux de microprogrammation pour le projet TREFLE l'auteur a développé les assembleurs-simulateurs des deux processeurs suivants :

- l'IOP dont le langage de micro-instructions est défini en annexe B,
- le FORMATTEUR décrit en annexe E.

Pour la mise au point des microprogrammes de l'IOP nous avons disposé d'un émulateur de mémoire morte (ROM Simulator) et du pupitre de maintenance de l'IOP. Le microprocesseur Texas TMS 9900, utilisé comme support du CONTROLEUR, assurera les fonctions d'émulateur pour le FORMATTEUR à l'aide d'un logiciel adapté à cette fonction.

CONCLUSION

Nous concluons ce chapitre décrivant notre collaboration à la réalisation de la machine TREFLE par quelques remarques sur les méthodes utilisées :

- le choix de processeurs microprogrammables comme unités de communication mérite d'être souligné. En effet, les qualités essentielles de cette technique résident dans la grande vitesse de transfert atteinte et la facilité de modification des algorithmes mis en oeuvre. Ainsi l'adaptation de la machine TREFLE à une autre source de données est aisément réalisable grâce à la souplesse du FORMATTEUR microprogrammé ;

- la réalisation des mécanismes de communication nécessite l'utilisation de techniques issues de plusieurs disciplines : l'électronique pour le développement des liaisons et la microprogrammation pour la mise en oeuvre des algorithmes microprogrammés. La méthode utilisée par l'équipe du projet TREFLE a consisté à utiliser l'outil informatique le plus souvent possible. Cette voie nous a amené à développer un ensemble d'outils logiciels destinés à nous assister dans les différentes étapes de la réalisation : simulateurs fonctionnels du matériel, micro-assembleurs et implanteurs du micrologiciel.

CHAPITRE 3 :

L'Evaluation

INTRODUCTION

I - L'EVALUATION DES PERFORMANCES DES SYSTEMES INFORMATIQUES

1.1. Etude des méthodologies utilisées pour l'évaluation par mesures d'un système informatique

1.1.1. Anatomie d'un évaluateur
1.1.2. Les méthodes de prises de mesures

1.2. La modélisation, outil d'évaluation

1.2.1. Les méthodes de résolution des modèles composés de réseaux de files d'attente
1.2.2. Les langages de simulation

II - L'EVALUATION DES PERFORMANCES DE LA MACHINE TREFLE

2.1. La modélisation de la machine

2.1.1. L'étude du fonctionnement de TREFLE
2.1.2. La définition du modèle
2.1.3. La résolution du modèle

2.2. L'évaluation par mesures du prototype

2.2.1. Les contraintes
2.2.2. Les principes

2.2.3. Les réalisations

2.3. Les résultats des premières simulations

2.3.1. Etude des variations de la taille des files d'attente
2.3.2. Etude des variations du taux d'activité des stations FILTRE

CONCLUSION

INTRODUCTION

La conception et la réalisation d'une nouvelle machine informatique s'accompagne en général d'une étude approfondie de son comportement [CaE 68], [CaH 67], [Maz 78]. Dans le cadre du projet TREFLE le but de cette réflexion est double :

- d'une part, visant l'intégration de l'architecture complète de la machine dans un seul circuit à très grande échelle d'intégration (VLSI), il est essentiel que nous connaissions la partie active du prototype lorsqu'il est soumis à une charge déterminée. En effet, plusieurs modules matériels ont été surdimensionnés au moment de la conception afin d'assurer le bon fonctionnement de l'ensemble de la machine ;

- d'autre part, l'étude du comportement du prototype dans des conditions limites de charge nous permettra d'adapter la machine à des utilisations particulières. En effet, il nous apparaît comme très important de bien connaître l'impact de la variation du débit du flot des données sur le comportement de la machine TREFLE.

L'étude que nous poursuivons est fondée sur deux approches complémentaires :

- la modélisation par simulation nous permet d'appréhender la complexité de la machine et d'interpréter aisément les mesures obtenues sur le prototype ;

- une technique particulière de prise de mesure facilite la collecte des informations internes aux processeurs. Ces mesures seront utilisées pour valider le modèle dans une première étape, puis pour l'exploiter ultérieurement.

Après une introduction générale aux problèmes posés par l'évaluation des performances des systèmes informatiques nous présentons le modèle et l'outil de mesures développés dans le but d'étudier le comportement de la machine TREFLE.

I - L'EVALUATION DES PERFORMANCES DES SYSTEMES INFORMATIQUES

L'évaluation quantitative des performances des systèmes informatiques peut être abordée selon deux approches :

- mesurer le système réel par des outils matériels ou logiciels,
- modéliser le système et résoudre analytiquement, numériquement ou par simulation le modèle obtenu.

Les distinctions opposant apparemment ces deux démarches les rendent étroitement complémentaires. En effet, si les mesures permettent d'obtenir une image fidèle du comportement du système, elles ne peuvent ni expliquer ni prédire son fonctionnement. En revanche, le rôle de la modélisation est de simplifier la réalité complexe du système dans le but de faciliter l'interprétation des mesures de son comportement, et de prédire ses performances dans des situations où l'expérimentation est difficile voire impossible. Réciproquement, les mesures sont nécessaires à l'alimentation des modèles et un modèle permet de dégager les paramètres essentiels à mesurer.

Dans la suite de cette première partie nous nous intéresserons particulièrement aux problèmes liés aux mesures et à la conception des outils de mesure, ainsi qu'aux méthodes et langages de simulation à événements discrets.

1.1. ETUDE DES METHODOLOGIES UTILISEES POUR L'EVALUATION PAR MESURES D'UN SYSTEME INFORMATIQUE

Le principal problème soulevé lors de la conception d'un outil de mesure est la difficulté d'accéder aux grandeurs et aux objets à observer. D'autres problèmes tels que les perturbations engendrées par l'outil et l'utilisation faite des mesures interviennent aussi dans la conception de l'évaluateur. Pour résoudre le premier problème posé, on peut choisir entre deux stratégies :

- concevoir, en fonction de l'objet à observer et du contexte dans lequel il se trouve, un outil très spécifique. Par cette approche, on obtient des outils très performants mais dont l'intérêt se limite à la campagne de mesures pour laquelle ils ont été conçus ;

- utiliser un outil général, paramétrable, qui s'adapte aux mesures à effectuer sur le site. Certains constructeurs, par exemple UNIVAC [Bor 71] et IBM [Schu 67], développent de tels produits pour leur usage interne.

Dans la suite de ce paragraphe nous présentons l'anatomie d'un évaluateur (outil de mesure) puis les méthodologies de mesure.

1.1.1. ANATOMIE D'UN EVALUATEUR

Le rôle d'un évaluateur est de transformer les événements observés dans le système en mesures interprétables par le concepteur. En général, un outil de mesure accomplit trois fonctions principales :

- l'acquisition des événements,
- la transformation des événements en valeurs de mesures,
- la sauvegarde des mesures.

+ L'ACQUISITION DES EVENEMENTS

Deux sous-ensembles composent cette première partie :

- les sondes : elles sont les terminaisons sensibles de l'évaluateur et sont insérées dans le système à évaluer. Leur rôle est de transmettre à l'évaluateur tout changement d'état se produisant à leur extrémité sensible. Dans un logiciel d'observation, les sondes sont composées des instructions qui provoquent les déroutements aux modules de l'évaluateur chargés de traiter les événements ainsi reconnus. Dans un logimètre, les sondes sont branchées sur les circuits et elles détectent les changements d'état des bascules et les passages des signaux ;

- les détecteurs : ce sont les mécanismes de validation ou d'invalidation d'un événement sur un ensemble de sondes. Ils permettent la mesure conditionnelle de certains événements par rapport à d'autres.

+ LA TRANSFORMATION DES EVENEMENTS EN VALEURS DE MESURES

Cette partie est composée des collecteurs qui, activés par les événements, les transforment en valeurs de mesures. Ces organes peuvent être de plusieurs types :

- des compteurs permettant de connaître le nombre d'occurrences ou la durée d'apparition des événements observés,

- des files pour conserver l'historique des occurrences de l'événement dans le temps (trace de l'événement),

- des vecteurs de compteurs pour construire automatiquement l'histogramme par occurrence ou par durée des événements,

- des tables de transformations permettant le codage et le regroupement d'événements.

+ LA SAUVEGARDE DES MESURES

Cette partie utilise généralement un périphérique de grande capacité et d'accès rapide, par exemple disque ou bande magnétique. L'unité de sauvegarde permet la conservation des mesures jusqu'à leur exploitation. Afin d'éviter des pertes d'informations dues à la lenteur relative des transferts, les entrées/sorties sont gérées par un mécanisme asynchrone à la prise de mesure.

Remarque : Certains évaluateurs analysent directement des mesures qu'ils prélèvent. Ainsi certains logimètres sont couplés à des mini-calculateurs ou à des périphériques spéciaux d'affichage [Den 73], [Hon 78]. L'évaluateur développé dans le projet OSSYOSCOPE [BoD 73] injecte directement ses mesures dans un modèle de prédiction du comportement du système observé.

1.1.2. LES METHODES DE PRISES DE MESURES

Pour aborder les différents aspects du problème de l'évaluation par mesures nous allons étudier le processus de prise de mesure relativement aux objets observés. Dans ce but nous découpons le système en quatre couches principales :

- la machine réelle,
- le processeur virtuel,
- le système d'exploitation,
- les utilisateurs.

Cette décomposition permet le classement des grandeurs à mesurer en quatre types, en fonction de la couche dans laquelle elles existent. Notre étude consistera à nous placer successivement dans chaque couche pour y analyser les problèmes qui y sont posés par l'implantation d'un évaluateur.

Remarque : Nous avons adopté cette décomposition car elle met l'accent sur la relation d'interprétation existante dans un système [Anc 74]. En effet, nous considérons que le système d'exploitation interprète les demandes des utilisateurs, et est lui-même interprété par le processeur virtuel. Cette relation d'interprétation permet d'expliquer les problèmes de privilèges et de protection qui limitent les possibilités des évaluateurs.

1.1.2.1. Mesures par un évaluateur implanté dans une tâche utilisateur

Le principal intérêt de cette solution au problème de l'évaluation réside dans la souplesse de la mise en oeuvre d'une session de mesure. En effet, il suffit d'activer depuis son terminal ou depuis le train de travaux, la tâche évaluateur pour obtenir dans un délai très court l'ensemble des mesures demandées.

Plusieurs techniques de mise en oeuvre s'offrent au choix du concepteur de systèmes. Une solution simple consiste à utiliser une tâche de type temps réel, activé sur un niveau d'interruption de priorité élevée.

En général, le système protège les utilisateurs mutuellement les uns des autres et l'espionnage entre tâches de différents usagers est impossible. La synchronisation entre l'évaluateur et les tâches mesurées ne pourra se faire qu'en introduisant des sondes dans les modules du système gérant les requêtes des usagers.

L'observation des objets propres au système est plus délicat, car l'évaluateur est limité dans ses privilèges. Alors que les utilisateurs ne sont protégés entre eux que par des techniques d'adressage, comme les registres de base, le système exclut, en général, toute lecture dans les zones qui lui sont allouées.

Nous citerons deux exemples de réalisations utilisant ce type de solution :

- l'évaluateur appelé TBM (Task Behaviour Monitor) [Ben 75] espionne les requêtes faites par une tâche utilisateur au système d'exploitation. Pour permettre ces mesures on introduit dynamiquement des sondes dans les sous-programmes traitant les appels moniteur (SVC ou CAL). L'évaluateur observe ainsi le comportement de la tâche relativement aux entrées/sorties, à la gestion mémoire, etc...

- l'évaluateur du système SIRIS 8 développé au CERT [BoC 76] est plus ambitieux. En effet, par une méthode analogue à TBM il observe le comportement interne du système ; pour des mesures ponctuelles il obtient des résultats intéressants.

Les inconvénients de tels outils sont essentiellement liés à leur faible portée et aux perturbations engendrées. Dans le cas de TBM ces problèmes sont minimisés car l'outil est transparent à la tâche observée mais les perturbations engendrées par l'outil de mesure du système SIRIS 8 limitent l'usage de cet évaluateur.

1.1.2.2. Mesures par un évaluateur implanté dans le système d'exploitation

Nous divisons cette partie en trois paragraphes en fonction de la nature des objets observés.

+ Mesure de la charge soumise au système :

La connaissance de la charge particulière à un site est extrêmement utile à l'adaptation des paramètres du système. Les objets à observer sont en général aisément accessibles car ils sont traités par des modules assurant l'interface entre l'utilisateur et le système de gestion des tâches.

Deux méthodes complémentaires permettent cette évaluation :

- l'exploitation du fichier comptable,
- la construction d'un évaluateur collectant les informations caractérisant le comportement des utilisateurs, par exemple les cartes de contrôle utilisées pour le passage des programmes dans le train des travaux ou les requêtes envoyées au système de temps partagé. Cette technique est fréquemment utilisée [BoC 75], [Ler 73], [Sch 67], [NoS 74] car elle permet d'obtenir une image très fine du comportement des usagers.

Le module de sauvegarde des mesures utilisera des fichiers analogues au fichier comptable, et ses entrées/sorties seront synchronisées avec les collecteurs afin de limiter le nombre de mesures perdues.

+ Mesure de la variation des grandeurs propres au système :

La connaissance du fonctionnement interne du système nécessite l'évaluation des grandeurs liées à son activité, par exemple le taux de multiprogrammation, la taille des files d'attente, etc... Tous ces objets sont propres au logiciel et un évaluateur placé dans la couche système y accède aisément.

Les collecteurs évaluant le comportement des objets évalués peuvent être activés selon deux principes :

- on mesure l'objet aux instants significatifs de son existence en introduisant les sondes dans les primitives-système de manipulation de cet objet [JaL 74], [BeK 74] ;

- on mesure l'objet à intervalle de temps régulier, une horloge activant les collecteurs [BoC 75].

Le problème posé par la sauvegarde des mesures est délicat à résoudre car l'évaluateur doit être transparent au système : il ne peut donc pas lui soumettre ses requêtes d'entrées/sorties pour la sauvegarde de ses mesures.

Les perturbations engendrées par ce type d'outil ont deux origines principales : l'introduction des sondes et des collecteurs dans le système provoque une augmentation de l'autoconsommation (overhead) qui peut être importante [BoC 75] et la sauvegarde des mesures par des entrées/sorties perturbe le système de gestion des unités périphériques.

+ Mesure de l'utilisation des ressources physiques par le système :

Les perturbations engendrées par les évaluateurs logiciels lorsqu'ils évaluent la configuration matérielle peuvent être suffisamment importantes pour rendre erronnées les mesures recueillies. Cet inconvénient rend ce type d'évaluateur inapte aux mesures sur les ressources physiques [Buz 76].

1.1.2.3. Mesures par un évaluateur implanté dans le processeur virtuel

Nous définissons le processeur virtuel comme étant la réalisation de la structure apparente de la configuration. En général cette couche est composée des microprogrammes qui s'exécutent sur une micromachine (réalisation physique) pour interpréter les instructions d'une macromachine (structure apparente). Dans certains systèmes particuliers cette couche s'élargit au logiciel simulant les machines fournies aux utilisateurs, par exemple CP 67 sur IBM 360/67 [MeS 70].

Le principal intérêt d'un évaluateur implanté dans la couche correspondant au processeur virtuel est de pouvoir accéder aux informations se trouvant dans les couches interprétées de façon totalement transparente. Cette transparence est assurée par l'exécution de l'évaluateur dans un espace de temps différent de celui des couches observées car l'horloge de la macromachine est gérée par l'interprète microprogrammé, et par la fonction d'interprétation qui permet l'analyse des instructions exécutées.

Nous présentons différents exemples d'implantation d'outils dans cette

couche en séparant les réalisations micrologicielles des réalisations logicielles :

- certains calculateurs sont orientés vers l'exécution de langages évolués. Les interprètes de ces langages sont microprogrammés et un évaluateur implanté dans ce micrologiciel peut observer l'utilisation des langages et les performances de l'interprète. Cette technique a été mise en oeuvre pour l'évaluation du calculateur Burroughs B1700 [Den 73] ;

- la structure de données représentant le processeur virtuel se trouve en général dans une zone particulière de la mémoire du calculateur. Cette structure contient les registres, le mot d'état, le vecteur d'interruptions, etc... Un évaluateur microprogrammé permet l'observation de cette structure pour améliorer la connaissance de l'utilisation faite du matériel par les usagers et le système. Cette démarche a été suivie à l'Université de Stanford pour évaluer le calculateur IC 7000 SCC [SaS 72], et chez IBM pour évaluer le 360 modèle 30 [Kee 68].

La notion de machine virtuelle a permis le développement d'évaluateurs logiciels :

- le projet VEST [Ben 73] consiste à virtualiser le processeur sur lequel s'exécute le système et permet d'évaluer l'ensemble formé par le système et sa charge ;

- une technique analogue a été développée à Rennes [BuI 77] pour l'évaluation du comportement des programmes par un interprète logiciel ;

- le projet SPY [Lef 72] permet l'observation des programmes s'exécutant sur une machine virtuelle générée par le système CP 67.

La sauvegarde des mesures prélevées pose le problème des perturbations

entraînées par le conflit entre l'évaluateur et le système de gestion des entrées/sorties. La solution généralement adoptée consiste à sauvegarder le contexte complet de la machine virtuelle, ainsi le système d'entrées/sorties trouve toujours le processeur dans l'état normal.

Les perturbations engendrées par l'évaluation sont susceptibles d'être mesurées de façon précise parce qu'elles sont systématiques et qu'elles ne portent que sur un ralentissement de l'exécution. Ce ralentissement peut être important : il est de l'ordre de la dizaine de fois dans le projet VEST.

Les inconvénients principaux de ce type d'outil sont liés à la difficulté technique de mise en oeuvre de microprogrammes et à l'accès depuis l'évaluateur à des informations d'un niveau logique suffisant pour en faciliter l'interprétation.

1.1.2.4. Mesures par un logimètre

Les logimètres ou "hardware monitors" ont été parmi les premiers outils de mesure conçus pour évaluer le rendement d'une configuration informatique. Le nombre grandissant de modèles disponibles sur le marché et leur succès chez les utilisateurs montrent l'importance avec laquelle les problèmes de performance des systèmes d'exploitation sont ressentis.

Leur principal intérêt est de permettre le prélèvement d'informations caractérisant les phénomènes dont l'échelle de temps est le cycle de l'unité centrale. De plus l'indépendance logique et physique entre les processeurs observés et le logimètre permet l'évaluation en parallèle de plusieurs unités et assure une observation exempte de toute perturbation [Hon 78].

L'inconvénient majeur d'un logimètre est son inaptitude à observer les phénomènes logiciels et l'interprétation de ses résultats est souvent

délicate :

- le logimètre "HARMONIE" développé à l'IRIA [Gau 75] et commercialisé par la société TITN est conçu pour une analyse différée des mesures,

- le projet OSSYOSCOPE [BoD 73] permet le traitement en ligne des données collectées,

- le projet CPM-X [Ruu 72] réalise la connexion entre un logimètre et un processeur microprogrammé.

Nous concluons cet exposé sur les méthodes d'évaluation par mesure par quelques remarques sur l'interprétation des résultats obtenus.

Le problème a d'abord été abordé de façon pragmatique par l'analyse des mesures brutes fournies par les évaluateurs. Cette approche se justifie pour l'interprétation de données telles que les mesures relatives aux appels des modules du système, mais elle est limitée pour l'analyse de phénomènes plus complexes comme l'engorgement des files d'attente, l'écroulement des performances, etc...

La modélisation apporte une solution générale au problème posé par l'interprétation des mesures et permet la prédiction du comportement du système évalué.

1.2. LA MODELISATION, OUTIL D'EVALUATION

Nous distinguons deux approches complémentaires de la modélisation : la modélisation statistique qui synthétise les mesures recueillies [Bio 76], [Jai 78] et la modélisation mathématique ou par simulation qui permet l'interprétation et la prédiction du comportement du système étudié [Moo 71], [BrB 74], [Ler 77]. Si un modèle statistique facilite l'appréhension du volume des mesures, il ne prend pas en compte la structure et

le fonctionnement du système évalué, aussi dans ce paragraphe nous nous intéressons plus particulièrement aux modèles mis sous la forme de réseaux de files d'attente dont la structure se rapproche des systèmes étudiés.

1.2.1. LES METHODES DE RESOLUTION DES MODELES COMPOSES DE RESEAUX DE FILES D'ATTENTE

Les méthodes de résolution des modèles de systèmes sont divisées en trois classes :

- les méthodes mathématiques exactes,
- les méthodes mathématiques approchées,
- la simulation.

+ LES METHODES MATHEMATIQUES EXACTES

Nous distinguons deux familles dans les méthodes qui permettent la résolution exacte de certains modèles :

- la méthode d'analyse de la chaîne de Markov associée au modèle [Ste 76] est une méthode numérique très intéressante pour résoudre des modèles simples, mais limitée par l'augmentation du nombre des états de la chaîne de Markov dès que le modèle se complique ;

- la méthode analytique BCMP est fondée sur les travaux de BASKETT, CHANDY, MUNTZ et PALACIOS et permet d'obtenir explicitement les probabilités d'états stationnaires pour certains réseaux particuliers [Bac 75]. Cette méthode exige des hypothèses très contraignantes sur les lois d'arrivées des clients et les lois de service des stations.

+ LES METHODES MATHEMATIQUES APPROCHEES

Ces méthodes sont moins restrictives sur les hypothèses que les méthodes exactes :

- les méthodes itératives acceptent les modèles comprenant des stations monoserveur à gestion de file FIFO et à loi de service générale [ChH 75] ;

- la méthode de diffusion permet de résoudre certains réseaux multi-classes [GeP 77].

+ LA SIMULATION [Ler 77]

Pour évaluer les systèmes informatiques par la simulation, on utilise généralement une simulation à événements discrets [LeP 76]. Cette méthode est fondée sur le fait que les variables d'état du système à simuler sont discrètes. Il en découle que les changements d'états du système ou événements se produisent à des instants bien définis, ce qui permet leur classement selon l'ordre chronologique et ainsi leur simulation sur un calculateur séquentiel.

La seule contrainte à satisfaire pour simuler un système est la possibilité de décrire les événements par des algorithmes.

Les buts poursuivis par la simulation peuvent être multiples :

- les simulations fonctionnelles permettent de vérifier le bon fonctionnement d'un système. La troisième partie du chapitre II de ce mémoire décrit une méthode utilisée pour l'écriture de simulateurs fonctionnels ;

- les simulations qui s'achèvent par l'apparition d'un événement particulier permettent l'étude des circonstances provoquant cet événement ;

- les simulations dont l'arrêt est commandé par la précision désirée sur les mesures prélevées permettent l'étude du comportement du modèle en état stationnaire.

La simulation pose de nombreux problèmes dont certains ne sont pas complètement résolus [Ler 77] :

- le prélèvement des mesures : pour être utilisables, il est nécessaire que les observations des variables du modèle vérifient certaines hypothèses, par exemple stationnarité du phénomène observé [Cox 72], existence de points de regénération [Cra 74], etc...

- le choix des entrées pour le simulateur : celui-ci peut être alimenté par des traces d'événements mesurés sur le système réel ou par les estimations des lois des variables du système réel. Ce problème montre le lien étroit qui existe entre un évaluateur et le modèle d'interprétation des mesures recueillies ;

- la validation du modèle reste la difficulté majeure posée par la simulation. En effet, on ne peut pas tirer de conclusions des résultats provenant de simulations si le modèle n'a pas été ajusté à la réalité qu'il doit représenter. Néanmoins, l'utilisation prospective d'un modèle permet de cerner les plages des valeurs tolérables pour les paramètres relatifs au fonctionnement de la réalisation.

1.2.2. LES LANGAGES DE SIMULATION

La description des modèles de simulation peut utiliser tous les langages généraux de programmation comme FORTRAN qui est très souvent adopté pour sa vitesse d'exécution et son universalité. L'inconvénient de ce choix est qu'il oblige le programmeur à écrire toutes les fonctions de gestion du temps. C'est pourquoi sont apparus des langages adaptés à la simulation incorporant ces primitives. SIMSCRIPT [Mar 66], GPSS [IBM 67] et

SIMULA 67 [Dah 66] sont les langages généraux de simulation les plus répandus. Ils offrent au programmeur la possibilité de manipuler les événements et les processus de son modèle, mais ils ne proposent aucun outil d'aide à l'ajustement du modèle à la réalité, par exemple échantillonnage et calculs de statistiques sur les mesures. Les langages spécialisés de simulation de systèmes informatiques viennent compléter cette énumération. OSSL [Dew 72], CSS [Sea 69], CASSANDRE [Mer 73] et LASCAR [Bor 76] sont adaptés à la définition de modèles de systèmes ou d'architectures de machines.

Plus récemment est apparu un nouveau type de produit : l'analyseur de modèles à réseau de files d'attente [Mer 77]. Ces produits permettent au concepteur de système de porter tous ses efforts sur la définition du modèle, en le secondant dans les tâches de contrôle de la réalisation. Nous décrivons ici certains aspects de QNAP [MeP 78] que nous avons utilisé pour la modélisation de l'architecture de TREFLE.

QNAP met à la disposition de l'utilisateur un langage de spécification de modèle se composant :

- de commandes de description de réseaux : ces commandes permettent de déclarer les stations, composées par les serveurs et les files d'attente, et le routage du réseau. Le traitement effectué par les serveurs peut être décrit par un algorithme et le routage peut être modifié dynamiquement ;

- de commandes d'activation des modules de résolution : QNAP offre à l'utilisateur plusieurs méthodes pour résoudre son modèle : les méthodes mathématiques, exactes et approchées, et la simulation. Ces commandes permettent aussi de calculer des statistiques, par exemple les intervalles de confiance, les fonctions d'autocorrélation sur les mesures échantillonnées pendant la simulation et d'initialiser le modèle.

Nous venons de résumer dans cette partie les principaux aspects des problèmes posés par l'évaluation des performances des systèmes informatiques en soulignant la complémentarité des différentes approches. L'étude approfondie de ce domaine de recherche nous a permis d'appréhender le cas particulier posé par la machine TREFLE. Notre réflexion a essentiellement porté sur une analyse de son comportement plutôt que sur une évaluation quantitative de ses performances. Dans ce but nous avons développé un modèle de simulation reflétant les principales caractéristiques de son architecture et nous avons implanté dans le prototype une technique originale de prise de mesure.

II - L'EVALUATION DES PERFORMANCES DE LA MACHINE TREFLE

En évaluant les performances de la machine TREFLE nous poursuivons plusieurs objectifs :

- la connaissance précise de l'utilisation des ressources matérielles de la machine nous permettra d'adapter les paramètres de l'architecture à des utilisations particulières. L'intégration de l'architecture dans un seul circuit exige un dimensionnement exact de ses modules fonctionnels ;

- la prédiction du comportement du prototype dans certaines conditions limites, par exemple traitement d'un flot ininterrompu de données, influence des variations du débit du flot de données, etc...

- l'ajustement de certains paramètres de l'architecture aux performances des processeurs, par exemple choix de la taille des secteurs sur le disque, etc...

Pour obtenir une image utilisable des performances nous proposons la

démarche suivante : après une étude de l'aspect fonctionnel de la machine, nous en dégageons un modèle à réseau de files d'attente. Ce modèle sera utilisé pour la détermination de l'ensemble des grandeurs à mesurer et pour l'interprétation des mesures obtenues sur le prototype. Considérant l'ensemble des paramètres à observer nous définissons l'outil de mesure qui sera intégré dans la machine TREFLE. La validation puis l'exploitation du modèle constituent la dernière étape de cette approche.

2.1. LA MODELISATION DE LA MACHINE

Dans ce paragraphe nous abordons la définition du modèle et les choix effectués pour sa résolution.

2.1.1. L'ETUDE DU FONCTIONNEMENT DE TREFLE

L'approche que nous suivons pour définir le modèle est fondée sur l'étude du fonctionnement de la machine. Cette démarche nous a permis de distinguer deux modes de fonctionnement :

- le mode cheminement,
- le mode balayage-traitement.

La machine TREFLE se trouve dans le mode cheminement quand elle recherche le domaine des informations à traiter. Le CONTROLEUR dirige cette recherche en guidant le bras du disque à l'aide du champ clé des secteurs lus par le FORMATTEUR ; les autres processeurs de l'architecture sont inactifs durant ce cheminement.

Elle se trouve dans le mode balayage-traitement lorsqu'elle explore le domaine des informations concernées par la question posée. Dans ce mode, les FILTRES et le MODULE DE CLASSEMENT sont actifs en parallèle et le

CONTROLEUR surveille les clés des secteurs traités pour orienter le balayage dans les limites du domaine.

Remarques : . par cette approche le FORMATTEUR se confond avec le disque. Ensemble ils constituent la source des caractères qui vont transiter dans les différents processeurs ;
. le mode cheminement ne constitue pas de phase critique dans le fonctionnement de la machine, car les déplacements du bras du disque sont plus lents que le traitement des clés par le CONTROLEUR. Ainsi l'étude portera sur le comportement des processeurs FILTRES et du MODULE DE CLASSEMENT dans le mode balayage-traitement.

Le modèle que nous proposons s'inspire directement de l'architecture physique de la machine TREFLE dont il reprend les principaux composants : les FILTRES, le MODULE DE CLASSEMENT avec leurs files d'attente respectives, et le disque. Il décrit une réalisation ayant trois FILTRES en cascade. Pour aborder sa description nous allons examiner en détail le fonctionnement de TREFLE en mode balayage-traitement, sur trois points précis :

- l'émission des caractères depuis l'ensemble formé par le FORMATTEUR et le disque ;

- le fonctionnement des modules du FILTRE ;

- l'influence du MODULE DE CLASSEMENT sur le flot de données.

+ L'EMISSION DES CARACTERES

Le rôle du FORMATTEUR est de lire les secteurs enregistrés sur le disque et d'émettre les caractères lus vers leurs destinataires : le champ clé est envoyé au CONTROLEUR, le champ données est envoyé en parallèle à

la première station du FILTRE et au MODULE DE CLASSEMENT. Un observateur placé sur le bus qui relie le FORMATTEUR au FILTRE percevrait ces transferts comme une succession de "bouffées". Chaque bouffée est constituée d'un nombre variable de caractères, chaque caractère étant séparé du précédent par un intervalle de temps constant, égal au débit du disque. Les temps inter-bouffées masquent les phénomènes suivants :

- lecture par le FORMATTEUR des champs ne concernant pas le FILTRE, par exemple les champs de synchronisation, les champs en-tête, les espaces intersecteurs, etc...

- déplacement du bras commandé par le CONTROLEUR.

+ LE FONCTIONNEMENT DES MODULES FILTRES

Les stations du FILTRE sont chargées de la reconnaissance et du transcodage des chaînes de caractères issues du FORMATTEUR. Ces deux fonctions utilisent la grammaire générée par le compilateur des questions. Le fonctionnement d'un module FILTRE peut être résumé par l'algorithme suivant :

Définition :

Nous appelons MOT une suite de CARACTERES. Ces caractères peuvent être des symboles du vocabulaire terminal ou du vocabulaire non terminal.

Algorithme :

si le MOT traité appartient à la grammaire extraite de la question
alors transcoder ce MOT par un ou plusieurs caractères
sinon ne pas transcoder ce MOT

Remarque : le transcodage d'un MOT peut générer plusieurs caractères si ce MOT est commun à plusieurs des requêtes combinées pour former une question.

La durée nécessaire à la vérification de l'appartenance d'un MOT à la grammaire est fonction de la forme du graphe généré pour la question. En effet, nous utilisons le principe de la dichotomie pour accélérer cette recherche ce qui ne permet pas de connaître a priori le nombre de cycles utilisés pour le traitement d'un caractère.

+ L'INFLUENCE DU MODULE DE CLASSEMENT SUR LE FLOT DE DONNEES

Le MODULE DE CLASSEMENT a deux modes de fonctionnement possibles :

- le mode de recherche exacte,
- le mode de recherche floue.

En mode de recherche floue il ne garde les informations triées sur leur note que dans les limites de la mémoire disponible.

En mode de recherche exacte cette mémoire peut se remplir totalement avant la fin de l'exploration du domaine des informations à traiter. Ainsi il faut vider le MODULE DE CLASSEMENT en bloquant les autres processeurs pendant la durée du transfert.

Remarque : ce blocage peut être toléré si le flot des informations est issu d'un disque, mais il provoque des pertes d'informations si les données sont traitées au vol.

Cette étude est restée très générale, notre but étant de modéliser le fonctionnement global de la machine TREFLE.

2.1.2. LA DEFINITION DU MODELE

L'étude du fonctionnement de la machine nous permet de préciser les critères de performance que nous allons utiliser pour évaluer son comportement. Ces critères font intervenir les performances globales du prototype et la validité des hypothèses retenues lors de sa conception.

Le critère principal concerne le débit du flot des données : quelle est sa valeur optimale pour le prototype construit et quelles sont les limites qu'il peut atteindre sans provoquer d'erreur de fonctionnement. Deux paramètres accessibles au concepteur sont intimement liés au débit : la taille des secteurs sur le disque qui fixe la longueur des bouffées et la taille des files d'attente devant le premier module du FILTRE et devant le MODULE DE CLASSEMENT. L'évaluation selon ce critère nous permettra d'affiner les choix pris lors du développement du prototype.

Le second critère est orienté vers la connaissance précise de l'utilisation des ressources du prototype dans le but d'une intégration complète de l'architecture. Les taux d'occupation des processeurs et des files d'attente fourniront des données quantifiées au concepteur pour lui permettre d'améliorer l'architecture de la machine.

Le modèle construit reprend les principes de l'architecture physique en tenant compte des caractéristiques particulières des processeurs précédemment étudiés. Les clients du modèle représentent les caractères traités par la machine et les stations modélisent les processeurs et leur file d'attente. Nous présentons les trois aspects du modèle :

- la génération des clients,
- les stations,
- les problèmes de synchronisation.

+ LA GENERATION DES CLIENTS

La station de génération des clients modélise le disque et le FORMATTEUR. Elle émet des clients par bouffées selon le fonctionnement étudié. L'algorithme de génération est la réalisation d'un automate à deux états :

- un état dans lequel les clients sont émis avec une fréquence fixe,
- l'autre état modélise les temps interbouffées.

Les transitions d'un état à l'autre sont fonction du temps passé dans cet état.

+ LA MODELISATION DES MODULES FILTRES

Une station du FILTRE est analogue, de par son comportement, à un serveur absorbantla plupart de ses clients et n'en propageant que quelques uns. Le processus stochastique modélisant le routage d'un serveur est régi par trois paramètres principaux : P, Q et R.

- P est la probabilité que le caractère traité soit le dernier d'un mot,
- Q est la probabilité qu'un mot de la question apparaisse dans la base,
- R est la probabilité qu'un mot de la question soit commun à plusieurs sous-questions (taux d'apparition moyen d'un même mot dans la question).

Nous obtenons ainsi les deux probabilités de transition :

- le client traité est envoyé dans la station suivante avec la probabilité $\frac{P \times Q}{R}$;

- il est envoyé dans l'état absorbant avec la probabilité $PA = 1 - \frac{P \times Q}{R}$.

Le temps de service d'une station du FILTRE est impossible à estimer a

priori car cette variable aléatoire est fonction du microprogramme généré par le compilateur. Nous possédons néanmoins quelques informations sur son comportement :

- le traitement d'un caractère par le filtre peut prendre un ou plusieurs cycles,
- le nombre maximal de cycles est fonction de la profondeur du graphe des comparaisons des différents mots : $n = \log_2$ (nombre de mots différents),
- un cycle a une durée constante.

L'enchaînement des trois stations est modélisé par le routage du réseau et la dernière station envoie ses clients au MODULE DE CLASSEMENT. Pour ce dernier ils représentent les notes des documents qu'il doit conserver.

+ LA MODELISATION DU MODULE DE CLASSEMENT

Nous modélisons le MODULE DE CLASSEMENT par un interrupteur à deux positions :

- en position fermée il laisse passer les clients depuis la station de génération vers la file d'attente de la première station du FILTRE,
- en position ouverte il interrompt le passage.

La transition de la position fermée à la position ouverte est faite selon la probabilité B :

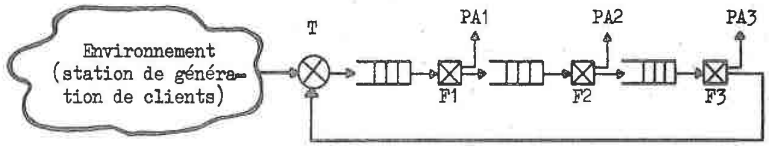
B est la probabilité que la note arrivant au MODULE DE CLASSEMENT provoque son blocage par suite de l'épuisement de la mémoire libre.

Remarque : la probabilité PNB utilisée dans le modèle est complémentaire à B : PNB = 1 - B.

La transition de la position ouverte à la position fermée est automatique et se produit un laps de temps constant après l'ouverture. Cette durée constante correspond au temps nécessaire au transfert du contenu de la mémoire du MODULE DE CLASSEMENT vers l'utilisateur.

Remarque : l'interrupteur qui modélise le comportement du MODULE DE CLASSEMENT peut être remplacé ultérieurement par un modèle plus fin prenant en compte la stratégie utilisée pour la gestion de la mémoire.

Nous concluons cette présentation générale du modèle par un schéma décrivant le réseau construit :



Environnement : modélisation du disque, du FORMATTEUR et du CONTROLEUR
T : modélisation du MODULE DE CLASSEMENT
F1, F2, F3 : modélisation des stations de FILTRAGE
PA1, PA2, PA3 : probabilités d'absorption des clients par les FILTRES.

Figure II-1 : Le modèle de la machine TREFLE.

2.1.3. LA RESOLUTION DU MODELE

L'architecture de la machine TREFLE nous oriente naturellement vers une modélisation par un réseau de files d'attente. Pour résoudre le modèle défini nous utilisons l'outil s'y prêtant le mieux, à savoir l'analyseur de modèles à files d'attente QNAP. Ce produit offre à l'utilisateur plusieurs méthodes pour résoudre son modèle :

- les méthodes mathématiques exactes,
- les méthodes mathématiques approchées,
- la simulation.

Les contraintes imposées par le modèle, par exemple la loi des arrivées, les lois de service et les problèmes de synchronisation, nous ont fait choisir une résolution par simulation.

Nous présentons dans ce paragraphe la réalisation du modèle avec ses paramètres et ses résultats.

Le modèle complet comporte sept stations :

- la station DISQUE de génération des clients,
- les stations V1 et V2 permettant le blocage des clients,
- la station T modélisant le MODULE DE CLASSEMENT,
- les stations F1, F2, F3 modélisant les FILTRES,
- le sémaphore de synchronisation S1.

+ LA STATION DISQUE

Son service est exprimé par un algorithme réalisant un automate à deux états.

Dans l'état 0 l'automate génère la taille TS du prochain secteur à lire suivant la loi mesurée expérimentalement sur le prototype. Ce mécanisme modélise le remplissage variable des secteurs. Puis l'automate attend un délai généré suivant une loi, elle aussi évaluée expérimentalement, avant de changer d'état. Ce délai modélise les temps séparant deux bouffées.

Dans l'état 1 l'automate génère TS clients en attendant un temps constant (0.8 microseconde) entre chaque émission, puis retourne à l'état 0.

Les clients sortant de cette station sont envoyés dans la station V2.

+ LES STATIONS V1 ET V2

Les deux stations V1 et V2 réalisent l'interrupteur de blocage des clients en entrée. Elles se synchronisent à l'aide du sémaphore S1 assurant la mutuelle exclusion.

La station V1 absorbe tous ses clients qui arrivent du MODULE DE CLASSEMENT. Son service consiste à émettre une requête P vers le sémaphore S1, à attendre un délai TV correspondant au temps de transfert de la mémoire du MODULE DE CLASSEMENT, puis à émettre une requête V.

La station V2 reçoit les clients émis par la station DISQUE et les envoie vers la première station du FILTRE F1. Dans le cas où V1 a émis une requête P vers S1 elle absorbe ses clients en les comptant pour mémoriser le nombre de clients perdus par suite du blocage.

+ LA STATION T

Le rôle de la station T est d'activer la station V1 avec la probabilité B et d'absorber les autres clients avec la probabilité PNB = 1 - B. Son temps de service TC est constant, il représente le temps de prise en compte de la note.

+ LES STATIONS F1, F2 ET F3

Elles absorbent les clients avec la probabilité PA1, PA2 et PA3 respectivement pour les FILTRES F1, F2 et F3. Leur temps de service est généré selon une loi mesurée sur le prototype.

Remarque : le fonctionnement du modèle présente un point de régénération que nous utiliserons pour échantillonner nos mesures. Ce point est situé à l'instant précédent la transition de l'état 0 à l'état 1 de l'automate de génération des clients. En effet, à cet instant le système est totalement vide de clients, ces derniers ayant tous été traités pendant le délai qui vient de s'écouler. Cette propriété n'est plus vérifiée si les processeurs sont bloqués par le transfert du MODULE DE CLASSEMENT.

La figure III-2 décrit la réalisation complète du modèle.

Les paramètres utilisés pour la résolution du modèle sont les suivants :

- la loi de service du DISQUE résultant de la combinaison de deux lois,
- la loi de service des FILTRES,
- les probabilités : PA1, PA2, PA3 et PNB, et les constantes TC et TV.

Ces paramètres seront évalués par l'outil de mesure présenté dans le paragraphe suivant, qui est en cours de réalisation sur le prototype.

Aussi dans une première phase nous allons exploiter le modèle avec des paramètres réalisant une approximation intuitive de la réalité :

- la loi de service du DISQUE sera une combinaison de deux lois :
  + une loi uniforme entre A et Z permettant le tirage de la valeur TS,
  + une loi exponentielle de moyenne BRAS générant des temps interbouffées.
  Les valeurs A, Z et BRAS ont été choisies en conformité avec le disque.;

- la loi de service des FILTRES : après avoir expérimenté les lois exponentielles et les lois d'Erlang nous avons retenu une loi uniforme sur l'intervalle BI, BS, en respectant pour BI et BS des valeurs conformes au fonctionnement des FILTRES ;

- les probabilités PA1, PA2, PA3 et PNB ainsi que les valeurs TC et TV seront estimées.

Nous présentons en annexe F la liste du modèle écrit en QNAP.

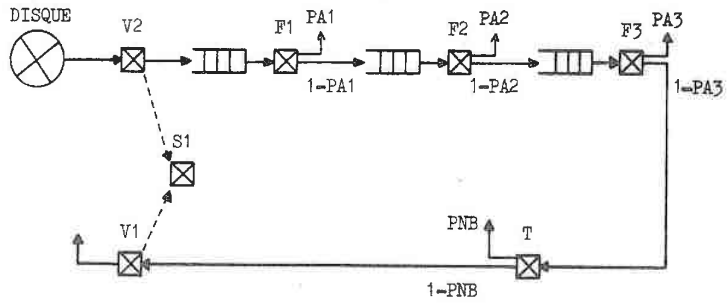


Figure III-2 : Description complète du modèle.

QNAP fournit à la fin de chaque simulation les résultats suivants (cf. fig. III-3) :

Résultats globaux :

- le temps exact de simulation,
- le nombre d'échantillonnages effectués pour la prise de mesure,
- le taux d'utilisation de la mémoire libre de QNAP.

Résultats pour chaque station identifiée par son nom :

- le temps de service moyen,
- le taux d'occupation,
- la longueur moyenne de la file d'attente,

- le temps de réponse moyen,
- le débit,
- le nombre de clients bloqués,
- le nombre de clients servis.

L'utilisateur peut obtenir en option les intervalles de confiance calculés par la méthode des blocs [Ler 77] et les fonctions d'autocorrélation des échantillons mesurés.

```
*** SIMULATION BEGINS ... ***
***TIME =     19999.92 , NB SAMPLES =     25 , CONF. LEVEL = 0.95
*************************************************************************
*                                                                       *
*QUEUE DISQ   SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .971     1.000    1.000     .971    1.030     .000    20599*
*                                                                       *
*QUEUE F1     SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .522      .538     .546      .530    1.030     .000    2059R*
*       +/-      .002      .038     .038      .002     .073     .000         *
*                                                                       *
*QUEUE F2     SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .528      .110     .112      .538     .208     .000     4163*
*       +/-      .006      .008     .008      .007     .016     .000         *
*                                                                       *
*QUEUE F3     SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .516      .011     .011      .518     .021     .000      426*
*       +/-      .016      .002     .002      .016     .003     .000         *

*                                                                       *
*QUEUE T      SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .800      .001     .001      .800     .001     .000       20*
*                                                                       *
*QUEUE V2     SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .000      .000     .000      .000    1.030     .000    20599*
*                                                                       *
*************************************************************************
*** END OF SIMULATION : TIME =     19999.92
```

Figure III-3 : Les résultats offerts par QNAP.

Nous avons complété ces résultats standard par des observations particulières effectuées en introduisant des points de mesure supplémentaires dans le modèle. Ainsi nous avons évalué la taille maximale atteinte par les files d'attente devant les stations FILTRE et le temps pris sur l'intervalle inter-bouffées pour traiter les clients en attente dans la première file. Les premiers résultats obtenus sont présentés dans la troisième partie de ce chapitre.

2.2. L'EVALUATION PAR MESURES DU PROTOTYPE

La phase de modélisation nous a permis de dégager les grandeurs à évaluer par le biais d'un outil de mesure. Nous pouvons remarquer que les paramètres du modèle s'expriment très simplement et ne nécessitent pas de traitements complexes pour l'accès à leurs valeurs. Cette propriété est inhérente aux modèles à réseau de files d'attente qui n'exigent en général que la connaissance des lois d'arrivées et de service, et des probabilités de transition.

Nous présentons dans ce second paragraphe les problèmes et les solutions de l'évaluation par mesures de la machine TREFLE.

2.2.1. LES CONTRAINTES

La principale contrainte rencontrée pendant la conception de l'évaluateur est essentiellement liée à l'échelle de temps microscopique à laquelle se déroulent les phénomènes à observer. En effet, la grande vitesse de traitement des processeurs, l'observation de phénomènes de courte durée et les échantillonnages très fréquents nous contraignent à réaliser un évaluateur qui n'introduit aucune perturbation engendrant des ralentissements dans le traitement.

Remarque : les premières simulations mettent en évidence la grande sensibilité des processeurs FILTRE aux perturbations en montrant la croissance de la longueur moyenne de la file d'attente devant la première station FILTRE, en fonction du temps de service de celle-ci.

La seconde contrainte est liée à l'architecture multiprocesseurs adoptée pour la machine TREFLE. En effet, les grandeurs à évaluer sont spécifiques et internes à chaque processeur physique de TREFLE ce qui nous contraint à répartir la fonction d'évaluation dans les différents mo-

dules fonctionnels.

Les autres contraintes sont engendrées par la forme des paramètres alimentant le modèle. Celui-ci nécessite pour sa résolution la connaissance des lois de service des différentes stations : DISQUE, FILTRES et MODULE DE CLASSEMENT, et des probabilités de transition dans le réseau. L'outil de mesure devra être adapté à l'observation des temps de service des processeurs et du routage des caractères dans l'architecture.

Les contraintes que nous venons de résumer vont nous permettre d'énoncer les principes que nous allons appliquer pour la conception de l'évaluateur.

2.2.2. LES PRINCIPES

Le premier choix que nous avons effectué a porté sur la méthodologie de mesure. Nous avons exclu la solution d'un évaluateur micrologiciel parce qu'il introduit des ralentissements néfastes au fonctionnement de la machine ; par ailleurs les logimètres courants ont un pouvoir de résolution insuffisant et sont trop lents pour évaluer sans pertes de mesures, les performances des processeurs de TREFLE. Ces remarques nous ont orienté vers la conception d'un évaluateur hybride adapté à la machine TREFLE. Les principes à l'origine de cette réalisation sont les suivants :

- les grandeurs à observer sont fonction du matériel et du micrologiciel, par exemple le temps de service des FILTRES est fonction du cycle du processeur et de l'algorithme de recherche. Cette constation nous incite à évaluer de préférence le micrologiciel parce que les mesures que nous pourrons ainsi prélever seront plus riches en informations que de simples observations du matériel ;

- les paramètres nécessaires à la résolution du modèle de simulation

peuvent être exprimés à l'aide de modèles statistiques élémentaires, par exemple des histogrammes approchant les lois de probabilité. La transformation des événements observés en mesures directement exploitables par le modèle sera faite par les collecteurs de l'évaluateur ;

- la sauvegarde des mesures sur une mémoire périphérique est en général source de perturbations ou de pertes d'observations. Comme la solution proposée ne nécessite aucun traitement différé des données, nous proposons de conserver les mesures dans les mémoires de travail des collecteurs ;

- l'évaluateur sera réparti dans les différents processeurs physiques pour avoir accès à toutes les grandeurs à observer. Ce principe implique un fonctionnement asynchrone des composants de l'outil de mesure ce qui nous interdit toute observation globale de la machine faisant intervenir le temps ;

- on peut considérer un évaluateur comme étant un processeur spécialisé ayant les caractéristiques suivantes :
  + une partie contrôle répartie : ce sont les événements se produisant dans le système observé qui activent l'évaluateur. Ces commandes transitent par les sondes qui sont dispersées dans le système évalué,
  + une partie opérative spécialisée : les collecteurs chargés de la transformation des événements en mesure et le mécanisme de sauvegarde des observations composent la partie opérative de l'évaluateur. Elle a pour particularité de ne pas renvoyer d'informations vers la partie contrôle.

La figure III-4 illustre l'évaluation d'un processeur microprogrammé par un logimètre.

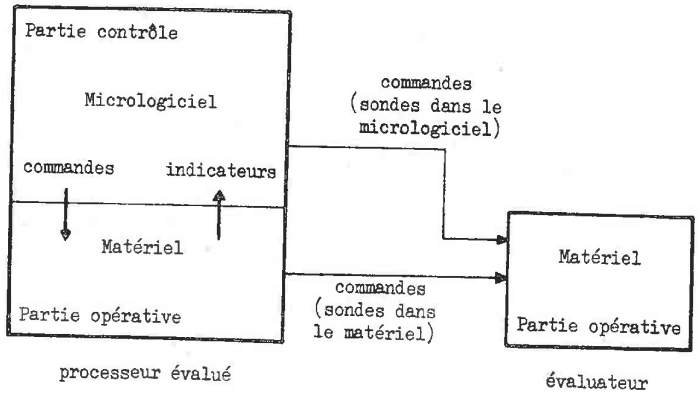
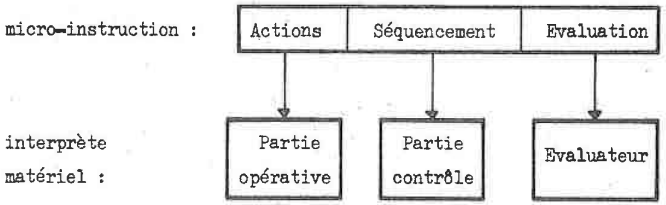


Figure III-4 : Evaluation d'un processeur microprogrammé.

Le parallélisme entre la partie opérative de l'évaluateur et la partie opérative du système observé peut être réel, comme dans un logimètre, ou simulé, lorsque les collecteurs sont logiciels. Si ce parallélisme est réel dans la partie opérative, il nous semble très intéressant de le conserver dans la partie contrôle. Cette propriété est en général vérifiée pour des sondes matérielles qui sont branchées en parallèle dans le processeur observé, mais elle ne l'est pas pour des sondes logicielles composées d'instructions s'exécutant séquentiellement sur un processeur.

La mise en oeuvre de l'évaluateur de la machine TREFLE utilise l'accès au parallélisme offert par la microprogrammation : la partie opérative de l'outil de mesure est activée parallèlement à l'exécution des micro-instructions de l'algorithme à évaluer par un champ particulier de la micro-instruction. Ce champ est décodé par une logique de contrôle dont les actions activent les collecteurs. Nous pouvons ainsi évaluer le comportement des algorithmes en activant l'évaluateur aux instants significatifs de l'exécution des microprogrammes. Le schéma ci-dessous

illustre le découpage des micro-instructions d'un processeur évalué par cette méthode.



Remarque : les principes énoncés ci-dessus ne sont applicables qu'aux processeurs microprogrammés ; le MODULE DE CLASSEMENT, le MODULE DE CALCUL et le CONTROLEUR ne pourront donc pas être évalués à l'aide de cette méthode. Leur importance dans l'étude du comportement de la machine TREFLE étant moindre, nous accéderons aux paramètres nécessaires au modèle par des outils adaptés aux grandeurs à observer.

2.2.3. LES REALISATIONS

Nous présentons dans ce paragraphe les réalisations des fonctions d'évaluation implantées dans le FORMATTEUR et dans les processeurs FILTRE. Ces fonctions ont pour rôle d'évaluer les paramètres suivants :

- pour le FORMATTEUR :
  + la distribution de la longueur des bouffées,
  + la distribution de la durée des temps interbouffées ;

- pour le FILTRE :
  + la distribution des temps de service des processeurs FILTRE,
  + la probabilité de propagation d'un caractère à travers les processeurs.

2.2.3.1. Evaluation des paramètres internes au FORMATTEUR

Les sondes sont constituées par le champ évaluation de la micro-instruction. Ce champ permet de détecter 3 types d'événements :

- l'événement début de l'émission d'une bouffée vers le FILTRE,
- l'événement émission d'un caractère de la bouffée,
- l'événement fin de l'émission de la bouffée.

Ces trois événements contrôlent les deux lois à évaluer car le début et la fin d'une bouffée coïncident avec la fin et le début d'un temps interbouffées.

Les collecteurs associés aux événements commandés par ces champs permettent l'élaboration des histogrammes des distributions des lois des paramètres. Ils se composent d'un ensemble de compteurs organisés en vecteurs. Chaque élément des vecteurs correspond à une classe de l'histogramme et les fonctions d'adressage utilisent des compteurs d'événements incrémentés par l'exécution des champs sondes.

Exemple :

Pour illustrer ce principe nous présentons le collecteur conçu pour l'évaluation de la longueur des bouffées émises par le disque.

Description :

- l'histogramme est élaboré dans le vecteur des compteurs CO. Chaque compteur est associé à une classe de l'histogramme et son adressage provoque son incrémentation ;
- le compteur CE compte les occurrences des événements liés à une même bouffée ;
- la logique d'adressage permet d'accéder au vecteur des compteurs CO en considérant la valeur du compteur CE comme adresse.

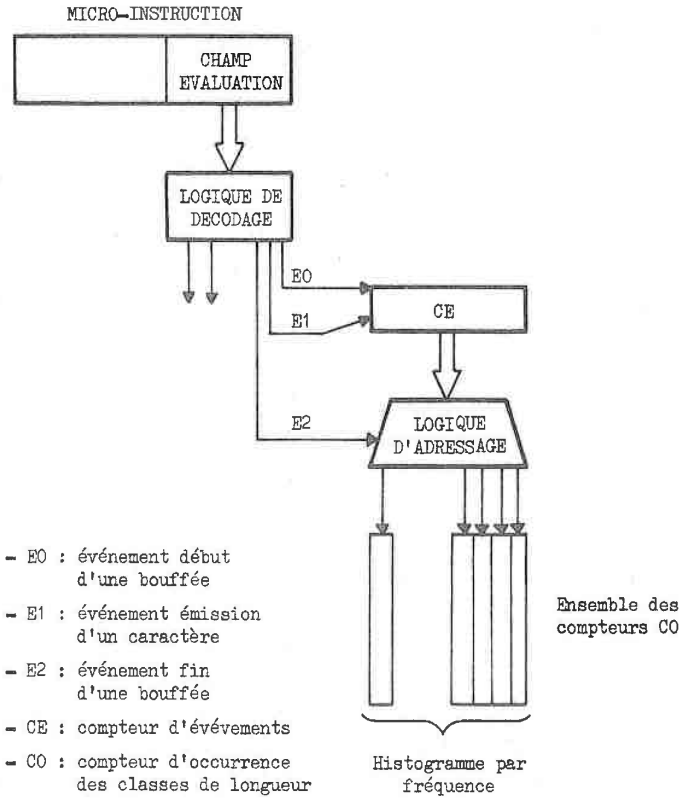


Figure III-5 : Collecteur élaborant l'histogramme de la distribution des longueurs de bouffées.

Fonctionnement :

Nous décrivons succinctement le fonctionnement du collecteur décrit ci-dessus :

1 - le microprogrammeur a codé le champ évaluation de la micro-instruction en respectant les principes suivants :

. activation de l'événement E0 dans la micro-instruction

précédant l'émission du premier caractère vers le FILTRE,

. activation de l'événement E1 dans la micro-instruction d'émission des caractères vers le FILTRE,

. activation de l'événement E2 dans les micro-instructions de fin de bouffées ;

2 - l'exécution d'une micro-instruction activant l'événement E0 provoque la remise à zéro du compteur CE ;

3 - l'exécution d'une micro-instruction activant l'événement E1 provoque l'incrémentation du compteur CE. Ce compteur contient le nombre de caractères émis dans la bouffée ;

4 - l'exécution d'une micro-instruction activant l'événement E2 provoque l'incrémentation du compteur CO adressé par la valeur de CE à cet instant. Cette méthode nous permet une élaboration rapide de l'histogramme des longueurs des bouffées.

Remarques : . le regroupement en classes des valeurs de CE sera assuré par la prise en compte des poids forts ;

. l'évaluation des temps interbouffées utilise le même principe que celui que nous avons décrit ci-dessus.

2.2.3.2. Evaluation des paramètres internes aux FILTRES

Mesurer les paramètres internes aux FILTRES est un problème plus complexe que celui posé par l'évaluation du FORMATTEUR. En effet, les fonctions opératives de ce processeur ne sont pas seulement activées par les micro-instructions, mais aussi par la logique combinatoire réalisant l'automate à pile. Cette contrainte nous oblige à compléter le mécanisme d'activation des collecteurs précédemment décrit par des sondes matérielles piégeant certains signaux de contrôles essentiels au fonctionnement du processeur (cf. chapitre I : fonctionnement du FILTRE). Nous présentons sur deux exemples la mise en oeuvre de ce principe.

Exemple 1 : L'évaluation des probabilités d'absorption des processeurs FILTRES.

Cette probabilité est calculée en utilisant le nombre de caractères entrés NCE et le nombre de caractères sortis NCS d'un processeur FILTRE :

$$PA = 1 - \frac{NCS}{NCE}$$

Nous évaluons NCE et NCS par le comptage des événements suivants :

- NCE : nous comptons les occurrences de l'événement E0.
  Evénement E0 : activation de la fonction SCAN qui fait progresser la chaîne d'entrée dans la mémoire FIFO de communication ;

- NCS : nous comptons des occurrences de l'événement E1.
  Evénement E1 : activation de la fonction de transcodage.

La détection de l'événement E0 est assurée par une sonde matérielle et c'est le champ évaluation de la micro-instruction qui active le collecteur associé à l'événement E1.

Exemple 2 : L'évaluation des temps de service des processeurs FILTRES.

La durée séparant deux progressions dans la chaîne d'entrée constitue le temps de service d'un processeur FILTRE. Pour évaluer cette durée nous utilisons un collecteur similaire à celui décrit pour le FORMATTEUR. Il aura pour rôle d'élaborer l'histogramme de la distribution de la loi de service en comptant le nombre de cycles micro-instructions nécessaires au

traitement d'un caractère. Le seul événement nécessaire à son activation est l'événement EO - Fin de service d'un caractère - qui est détecté par une sonde matérielle piégeant les progressions dans la chaîne d'entrée en observant les activations de la fonction SCAN (cf. chapitre I : fonctionnement du FILTRE).

Remarque : le fonctionnement du processeur FILTRE évite l'observation de l'événement début de service car son horloge se bloque automatiquement lorsque la mémoire FIFO d'entrée est vide.

Description de l'évaluateur :

- le compteur CH compte les cycles nécessaires au traitement d'un caractère ;
- l'histogramme est établi dans le vecteur des compteurs CD.

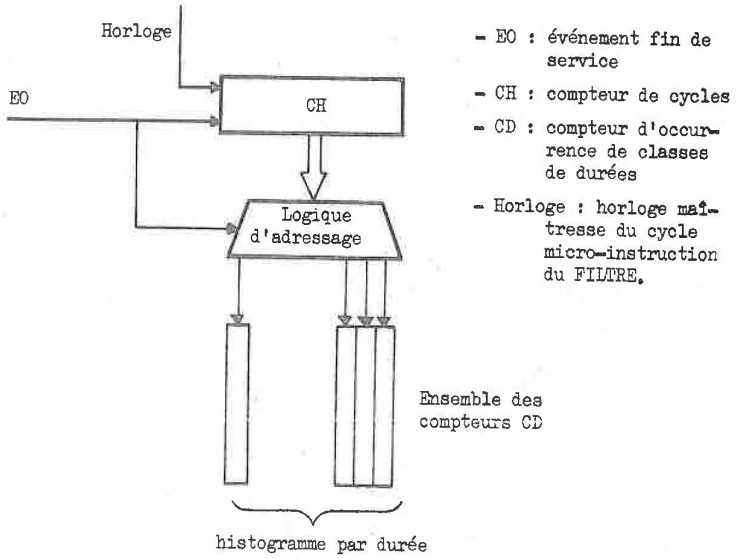


Figure III-6 : Collecteur élaborant l'histogramme de la distribution des durées de service.

Fonctionnement :

Nous décrivons brièvement le fonctionnement du collecteur décrit ci-contre :

1 - le compteur CH est incrémenté à chaque cycle d'horloge ;
2 - l'arrivée de l'événement EO provoque l'incrémentation du compteur CD adressé par la valeur de CH à cet instant et la remise à zéro de CH.

2.2.3.3. Remarques générales sur la méthode de prise de mesure

La méthode de prise de mesure que nous avons développée est adaptée à l'observation du comportement de processeurs microprogrammés, néanmoins elle soulève plusieurs problèmes parmi lesquels nous pouvons citer :

- la synchronisation entre l'évaluateur et le processeur observé,
- le démarrage et l'arrêt d'une session de mesure,
- l'accès aux valeurs des mesures prélevées,
- le choix de la taille des collecteurs.

Pour résoudre ces différents problèmes nous proposons les méthodes suivantes :

- les problèmes temporels seront résolus par l'asservissement de l'horloge de l'évaluateur aux signaux de commande de la prise de mesure. Ainsi le démarrage et l'arrêt de la prise de mesure se traduisent respectivement par un déblocage ou un blocage de l'horloge du collecteur concerné. De plus, la synchronisation entre le processeur observé et l'évaluateur sera assurée par la propagation de l'horloge du premier vers le second ;

- l'accès aux mesures prélevées sera facilité par un processeur particulier permettant la consultation aisée des mémoires des collecteurs et offrant des possibilités de visualisation des résultats. Dans notre

réalisation nous utilisons un microprocesseur Texas TMS 9900 pour assurer cette fonction. Une console alphanumérique et un moniteur logiciel permettront à l'utilisateur de gérer l'évaluation ;

- pour mettre en oeuvre les collecteurs il est nécessaire de figer le nombre des compteurs représentant les classes des histogrammes à élaborer. Ce choix n'est pas aisé lorsqu'on ne possède que peu d'informations sur la taille et la forme des valeurs échantillonnées. Nous avons rencontré ce problème pour le dimensionnement du collecteur élaborant l'histogramme de la distribution de la durée du temps inter-bouffées. Pour le résoudre nous avons choisi un nombre arbitraire de classes et nous avons découpé le compteur d'événements en champs ce qui permet le partitionnement du vecteur des classes. Cette méthode nous permet après plusieurs essais de centrer l'histogramme sur les valeurs prélevées.

Nous terminons cette présentation de l'évaluateur en citant quelques caractéristiques de sa réalisation :

- les collecteurs sont réalisés en technologie TTL Schottky,
- les compteurs des collecteurs sont en double accès :
  + un accès pour l'incrémentation,
  + un accès pour la lecture,
- le processeur de visualisation des mesures accède aux collecteurs par son mécanisme d'entrées/sorties,
- la taille des compteurs est fonction de l'usage des collecteurs ; nous présentons ci-dessous quelques exemples :
  - le compteur CE totalisant le nombre de caractères par bouffées a une longueur de 16 bits,
  - le compteur CH totalisant le nombre de cycles nécessaires au traitement d'un caractère a une longueur de 4 bits,
  - les compteurs NCS - nombre de caractères sortis - et NCE - nombre de caractères entrés - ont une longueur de 32 bits.

2.3. LES RESULTATS DES PREMIERES SIMULATIONS

Les applications développées sur la machine TREFLE n'étant pas encore opérationnelles, il ne nous est pas possible de disposer actuellement de résultats de mesures. Nous avons donc choisi dans un premier temps de prospecter les plages des valeurs des différents paramètres du modèle par un ensemble de simulations. Nous présentons dans ce troisième paragraphe les études que nous avons effectuées sur les variations de la taille de la file d'attente de la première station du FILTRE et sur le taux d'activité des stations du FILTRE.

2.3.1. ETUDE DES VARIATIONS DE LA TAILLE DES FILES D'ATTENTE

Le but que nous avons poursuivi est la connaissance de l'influence de la taille des secteurs de la base sur le comportement de la file d'attente d'entrée du premier module FILTRE. L'intérêt de cette étude est multiple :

- la détermination des seuils d'engorgement nous permet de connaître les limites de fonctionnement de la machine TREFLE ;

- la connaissance précise de la taille moyenne et maximale de la file d'attente nous permet d'ajuster le matériel du prototype aux conditions de son utilisation ;

- l'appréciation de l'influence du temps de service des processeurs FILTRE sur la taille de leur file d'attente d'entrée nous permet de cerner les valeurs extrêmes tolérables pour les temps de service.

2.3.1.1. Les conditions de l'expérimentation

L'ensemble des simulations composant cette expérience nous a permis de

faire varier deux paramètres :

- la durée du temps de service des processeurs FILTRE,
- la taille des secteurs stockés sur le disque.

- la durée de service : Pour simuler le temps de service d'une station du FILTRE nous utilisons une variable aléatoire T suivant une loi uniforme sur l'intervalle BI, BS. Le tableau ci-dessous présente les valeurs choisies pour BI et BS ainsi que leur moyenne.

	BI	BS	$\bar{t}$	E(t)
simulation n° 1	0.150	0.90	0.522 $\pm$ 0.003	0.525
2	0.150	1.05	0.598 $\pm$ 0.003	0.600
3	0.150	1.20	0.672 $\pm$ 0.003	0.675
4	0.150	1.35	0.747 $\pm$ 0.004	0.750
5	0.150	1.50	0.821 $\pm$ 0.004	0.825
6	0.150	1.65	0.896 $\pm$ 0.004	0.900

BI : borne inférieure de l'intervalle
BS : borne supérieure
$\bar{t}$: estimateur de la moyenne avec son intervalle de confiance
E(t) : moyenne théorique sur l'intervalle
Note : l'échelle de temps est la microseconde.

- la taille des secteurs : Pour simuler la taille des secteurs nous utilisons une variable aléatoire L suivant également une loi uniforme sur l'intervalle A, Z. Le tableau ci-dessous précise les valeurs choisies pour A et Z.

	A	Z
courbe I	800	1000
courbe II	1600	2000
courbe III	3200	4000

2.3.1.2. Les résultats obtenus

Les courbes présentées en figure III-7 montrent la progression de la taille moyenne - courbes I, II et III - et de la taille maximale - courbe IV - de la longueur de la file d'attente d'entrée dans la première station du FILTRE en fonction du temps de service de cette station.

Note : L'échelle des abscisses est linéaire et présente les valeurs des temps de service moyen.
L'échelle des ordonnées est logarithmique et présente les longueurs des files d'attente.

Précision des résultats :

Le tableau ci-dessous présente les variations des intervalles de confiance pour les valeurs de la courbe I prise en exemple.

simulation n°	temps de service moyen	nombre moyen de clients	intervalle de confiance
1	0.522	0.546	$\pm$ 0.038
2	0.598	0.681	$\pm$ 0.057
3	0.672	0.926	$\pm$ 0.076
4	0.747	1.77	$\pm$ 0.137
5	0.821	15.203	($\pm$ 3.259)
6	0.896	95.701	($\pm$ 21.528)

Remarque : les intervalles de confiance obtenus pour les simulations 5 et 6 sont sujets à caution ; les fonctions d'autocorrélation présentées ci-dessous nous montrent une dépendance dans les échantillons prélevés au cours de ces deux simulations.

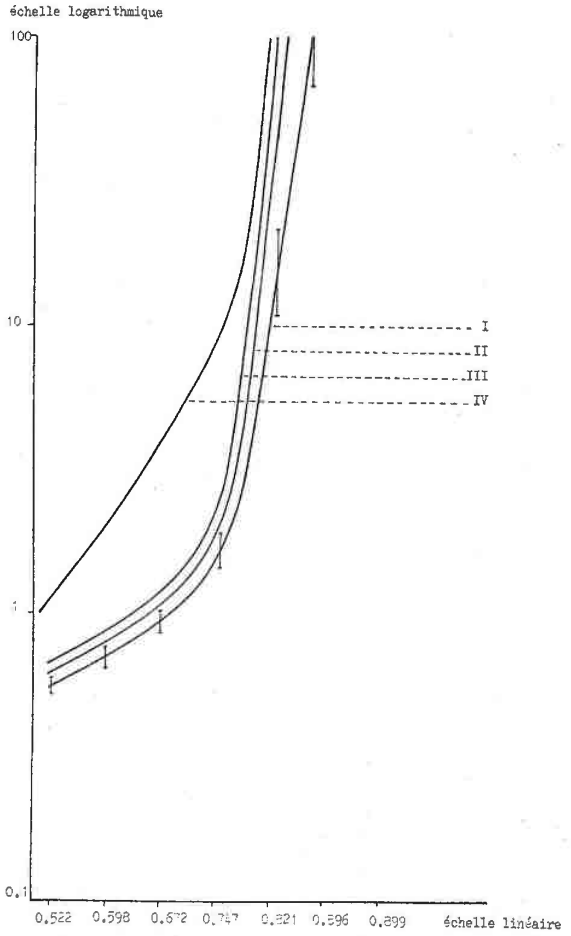


Figure III-7 : Variations de la taille de la file d'entrée de la première station du FILTRE.

Tableau des valeurs des fonctions d'autocorrélation calculées sur les tailles de la file d'attente (courbe I) :

	ordre de la fonction d'autocorrélation				
	1	2	3	4	5
simulation n° 1	-0.081	-0.106	-0.104	0.082	0.038
2	0.001	0.004	0.038	0.056	-0.095
3	-0.042	-0.012	-0.010	0.107	0.082
4	-0.126	0.070	-0.019	0.125	0.144
5	-0.150	0.185	0.140	-0.235	0.385
6	0.233	0.201	-0.177	-0.091	-0.263

Les simulations 5 et 6 ont un taux d'autocorrélation plus élevé car les échantillonnages ne sont pas toujours effectués aux points de régénération, le modèle étant parfois à la limite de la saturation. Pour remédier à ce défaut les simulations auraitent dû se prolonger de façon prohibitive.

2.3.1.3. L'interprétation des résultats

Nous interprétons les trois courbes - I, II et III - visualisant les variations de la taille moyenne des files d'attente en les décomposant en deux parties :

- la première partie est comprise entre les abscisses 0.5 et 0.7 : cette partie de courbe peut être approchée par une droite, ce qui nous permet de déduire que l'accroissement de la taille moyenne de la première file d'attente est exponentiel. Le modèle est en régime stationnaire et les caractères sont traités sans qu'il n'y ait d'accumulation dans la file d'attente, la taille maximale enregistrée étant de 4 ;

- la seconde partie se situe au-delà de l'abscisse 0.7. Très rapidement le nombre moyen de caractères dans la file augmente et fait diverger

le modèle. Nous plaçons le seuil d'engorgement du système vers 0.750 microseconde de temps de service. Le tableau ci-dessous présente les valeurs caractéristiques de la file pour ce seuil :

	taille moyenne	taille maximale	temps de réponse
courbe I	1.771 ± 0.15	9	1.647
courbe II	1.873 ± 0.15	9	1.665
courbe III	1.995 ± 0.15	10	1.690

Nous remarquons les seuils à partir desquels la longueur de la file d'attente du modèle devient infinie :

+ au-delà d'un temps de service de 0.899 pour la courbe I
+ au-delà d'un temps de service de 0.821 pour la courbe II
+ au-delà d'un temps de service de 0.821 pour la courbe III.

Remarque : la machine TREFLE forme un système à capacité limitée : nous considérons donc comme seuil d'écroulement la valeur à partir de laquelle la file d'attente peut atteindre une longueur supérieure à 256 caractères.

L'interprétation de la courbe IV présentant les variations de la taille maximale atteinte par la file d'attente est analogue à celle formulée pour les variations des tailles moyennes :

- la première partie de la courbe nous permet d'envisager une file d'attente, de longueur 16 au maximum, capable de stocker les caractères en attente dans le cas où le temps de service moyen est inférieur à 0.750 microseconde ;

- la seconde partie montre également le phénomène d'écroulement.

Les conclusions que nous formulons au vu des résultats de cette première série d'expérimentations nous permettent de préciser la limite des temps

de service acceptables pour assurer un bon fonctionnement de la machine TREFLE.

Nous pouvons fixer à 8 le nombre maximum d'instructions nécessaires au traitement d'un seul caractère, à condition que les temps de service soient uniformément distribués entre 1 et 8 cycles.

De plus, nous pouvons négliger l'influence de la taille des secteurs sur l'engorgement de la première file d'attente si le modèle reste en régime stationnaire et nous pouvons limiter à 16 le nombre des emplacements de la file d'entrée. En effet les courbes I, II et III sont très voisines dans leur première partie et les phénomènes d'engorgement sont trop brutaux pour être enrayés par une file d'attente de grande capacité.

2.3.2. ETUDE DES VARIATIONS DU TAUX D'ACTIVITE DES STATIONS FILTRE

Le but que nous avons poursuivi est la connaissance de l'influence de la taille des secteurs et le temps de service de la première station du FILTRE sur son taux d'activité, et l'étude de l'influence du taux d'absorption de la première station sur le taux d'activité de la deuxième. Ces deux études offrent de nombreux éléments de réflexion :

- l'étude de la variation du taux d'activité de la première station du FILTRE nous permet d'apprécier les valeurs des seuils d'engorgement et de les confronter avec les résultats de l'étude précédente ;

- la connaissance des variations du taux d'activité de la seconde station du FILTRE nous permet d'évaluer l'influence de la complexité de la question, modélisée par le taux d'absorption, sur le comportement des stations du FILTRE. Cette évaluation apporte des informations utiles au choix d'implantation des stations du FILTRE par matériel ou par logiciel.

Remarque : la troisième station ayant un taux d'activité très faible, en général inférieur à 2 %, nous n'avons pas abordé l'évaluation de son comportement dans cette étude.

2.3.2.1. Les conditions de l'expérimentation

Nous avons repris les mêmes conditions que celles présentées dans le paragraphe précédent :

- pour la durée de service des processeurs FILTRE,
- pour la taille des secteurs stockés sur le disque.

De plus, nous avons fait varier le taux d'absorption de la première station du FILTRE : PA1. Les valeurs que nous avons choisies pour nos expérimentations sont les suivantes :

PA1 = 0.5
0.8
0.95

Sachant que pour PA1=0 le taux d'activité de F2 est égal au taux d'activité de F1, et que pour PA1=1 le taux d'activité de F2 est nul, ces cinq valeurs nous ont permis de tracer les courbes présentées en figure III-9, III-10 et III-11.

2.3.2.2. Les résultats obtenus

- les courbes présentées en figure III-8 montrent la progression du taux d'activité de la première station du FILTRE en fonction de son temps de service. La courbe I présente les résultats obtenus avec une taille de secteur comprise entre 800 et 1000 caractères, la courbe II entre 1600 et 2000 et la courbe III entre 3200 et 4000.

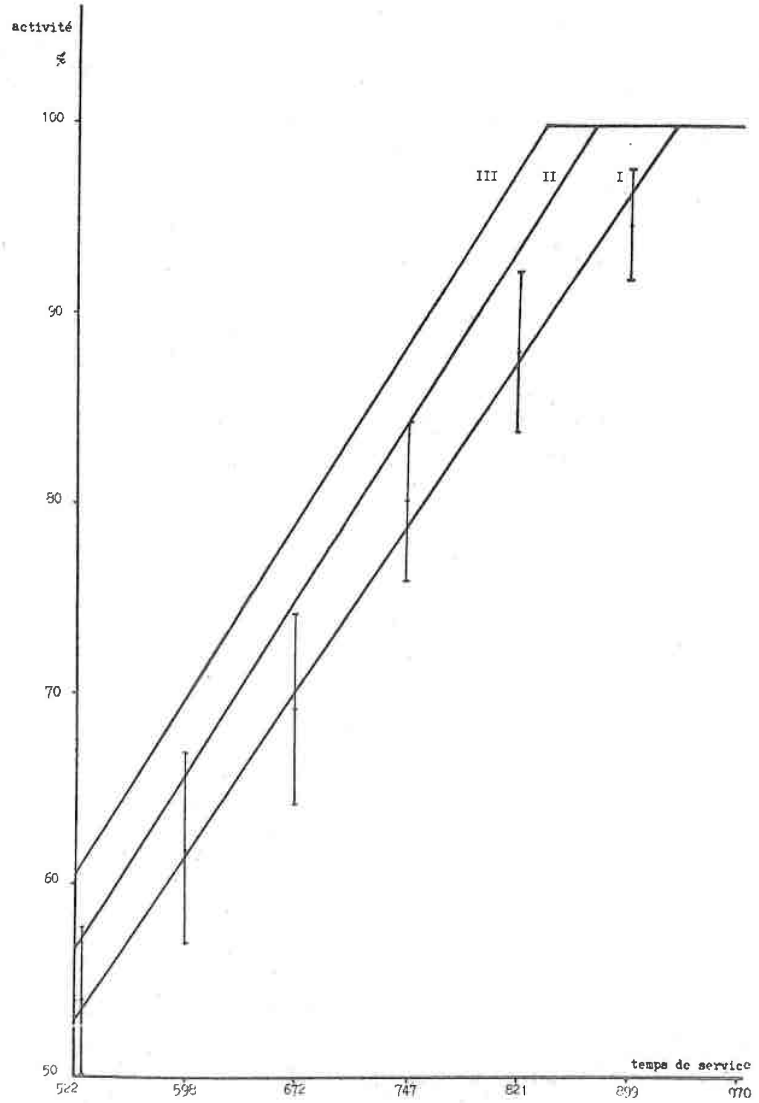


Figure III-8 : Taux d'activité F1 en fonction du temps de service.

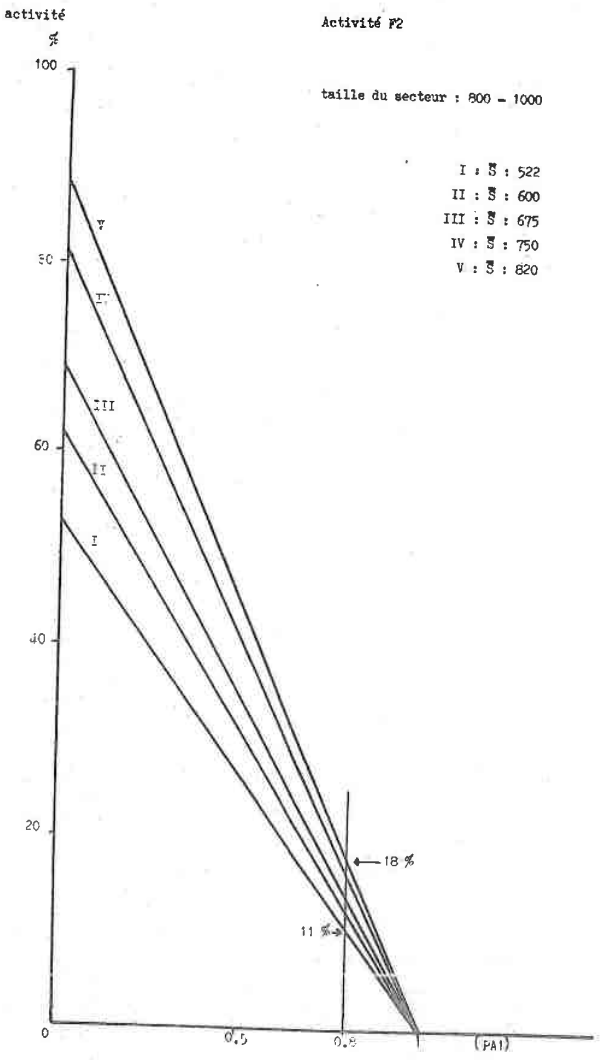


Figure III-9 : Taux d'activité de F2 en fonction de PA1.

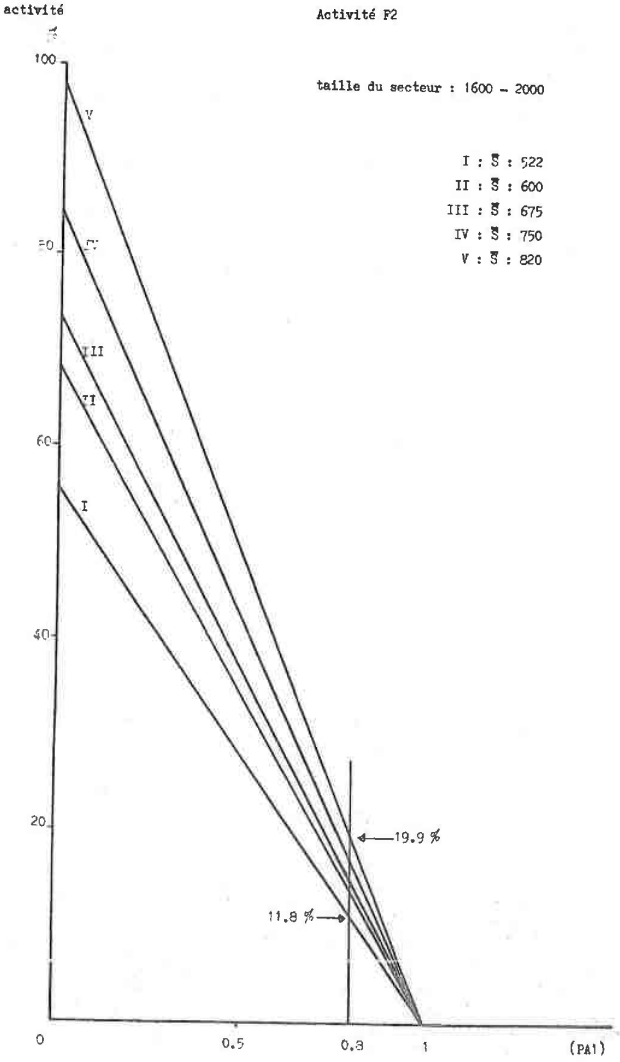


Figure III-10 : Taux d'activité de F2 en fonction de PA1.

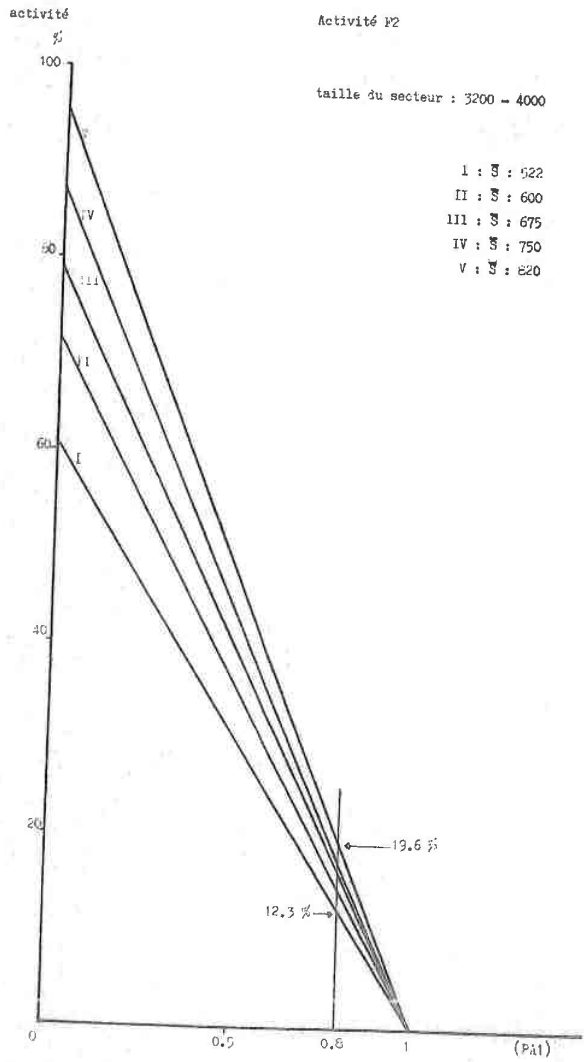


Figure III-11 : Taux d'activité de F2 en fonction de PA1.

Note : L'échelle des abscisses est linéaire et présente les valeurs des temps de service moyens.
L'échelle des ordonnées est linéaire et présente les taux d'activité de la station F1.
Nous avons visualisé les intervalles de confiance pour la courbe I.

- les courbes présentées en figures III-9, III-10 et III-11 décrivent les variations du taux d'activité de la deuxième station du FILTRE en fonction de la probabilité d'absorption de la première station.

+ les courbes présentées en figure III-9 visualisent les résultats obtenus pour une taille de secteur comprise entre 800 et 1000 caractères. Cette taille n'intervient pas directement mais elle est combinée au temps de service pour donner le taux d'activité ;

+ les courbes présentées en figures III-10 et III-11 sont analogues à celle de la figure III-9 pour les tailles de secteurs comprises entre 1600 et 2000, et 3200 et 4000 respectivement.

Note : L'échelle des abscisses est linéaire et présente les valeurs de la probabilité d'absorption.
L'échelle des ordonnées est linéaire et présente les taux d'activité de la station F2.

Précisions des résultats

Pour cette deuxième série d'expérimentations les paramètres sont identiques à ceux des premières simulations. Nous présenterons cependant les valeurs des fonctions d'autocorrélations car elles ne présentent plus l'anomalie mentionnée précédemment.

Les tableaux ci-dessous regroupent les variations des intervalles de confiance pour les valeurs des courbes I, II et III de la figure

III-8 représentant les variations du taux d'activité de la station F1 en fonction du temps de service et de la taille des secteurs sur le disque

	simulation n°	temps de service moyen	taux d'activité moyen (F1)	intervalle de confiance
courbe I	1	522	53.8	± 3.8
	2	598	61.8	± 5.1
	3	672	69.3	± 5.5
	4	747	80.3	± 4.2
	5	821	88	± 4.2
	6	896	94.7	± 3.3
courbe II	1	522	55.4	± 3.8
	2	598	68.6	± 3.9
	3	672	74	± 4
	4	747	84	± 5
	5	821	98	± 2.1
courbe III	1	525	60.6	± 2.1
	2	599	71.7	± 2.9
	3	673	78.5	± 4.3
	4	747	86.8	± 5.2
	5	821	95.7	± 3.8

Nous avons vérifié l'indépendance des échantillons prélevés en calculant les fonctions d'autocorrélation sur les mesures de la file F1. Le tableau ci-dessous présente les résultats obtenus pour les simulations visualisées par la courbe I.

simulation n°	ordre de la fonction d'autocorrélation sur les taux d'activité du serveur F1				
	1	2	3	4	5
1	-0.080	-0.106	-0.102	0.078	0.037
2	0.007	0.004	0.030	0.066	-0.108
3	-0.010	-0.051	-0.005	0.031	0.048
4	0.000	-0.005	-0.044	-0.059	0.142
5	-0.075	0.030	0.018	0.040	-0.015
6	0.083	-0.046	-0.104	0.038	0.032

Aucune valeur n'étant très élevée nous considérons que nos intervalles de confiance sont valides.

Le tableau ci-dessous présente les variations des intervalles de confiance pour les valeurs des courbes de la figure III-9 pour une probabilité PA1 égale à 0.8. Cette courbe visualise les taux d'activité de la station F2 lorsque la taille du secteur est comprise entre 800 et 1000 octets et que son temps de service varie de 0.522 à 0.821 microsecondes.

courbe n°	temps de service	taux d'activité moyen	intervalle de confiance
I	0.522	11.0	± 0.8
II	0.598	12.7	± 1.2
III	0.673	14.1	± 1.3
IV	0.747	16.4	± 1.0
V	0.821	17.8	± 1.1

Les fonctions d'autocorrélations sur la file F2 étant normales nous considérons que ces intervalles sont valides.

2.3.2.3. L'interprétation des résultats

Les trois courbes de la figure III-8 confirment les résultats des expérimentations sur la taille des files d'attente en mettant en valeur les seuils d'écroulement de la première station du FILTRE. Si nous confrontons ces deux expériences nous pouvons en déduire que le taux d'activité maximal de la première station du FILTRE est compris entre 75 % et 80 %.

Nous remarquons que la taille des intervalles de confiance ne permet pas une interprétation graphique très précise.

L'étude des courbes présentant les variations du taux d'activité de la station F2 nous permet d'affirmer que la deuxième station du FILTRE n'aura jamais un taux d'activité très élevé. Pour une probabilité d'absorption de 0.8 le taux d'activité est compris entre 11 % $\pm$ 0,8 et 19.9 % $\pm$ 1,0 pour un ensemble d'expérimentations pendant lesquelles la durée de service du processeur et la taille des secteurs ont varié. De plus, la comparaison des courbes des figures III-9, III-10 et III-11 met en évidence l'importance primordiale du temps de service du processeur sur son taux d'activité.

Remarque : nous considérons 0,8 comme étant un taux d'absorption minimal que nous pouvons décomposer de la manière suivante (cf. 2.1.2) :
$1 - \frac{P \times Q}{R} = 0,8$.
Soit une question contenant 10 mots différents : R = 0,1.
Si la longueur moyenne d'un mot est de 5 caractères alors P = 0,20.
Le taux d'apparition d'un mot de la question dans la base serait alors Q = 0,10, ce qui est peu réaliste...

Nous concluons cette deuxième étude par la constatation que la première station du FILTRE a une influence prépondérante sur le fonctionnement global de l'architecture de TREFLE. Cette remarque peut influer le choix de l'implantation des stations du FILTRE en optimisant leur coût.

Par exemple :

- la première station est réalisée dans une technologie rapide, par exemple ECL ou TTL Schottky ;

- la seconde station est réalisée dans une technologie plus lente mais plus économique, par exemple H-MOS ;

- les autres stations sont multiplexées temporellement sur une même machine support, par exemple un microprocesseur classique.

CONCLUSION

Nous avons présenté dans ce chapitre les méthodes que nous avons utilisées pour étudier le comportement de la machine TREFLE. Après une présentation générale des problèmes posés par l'évaluation des systèmes informatiques nous avons montré l'intérêt d'une démarche utilisant à la fois la modélisation et l'observation directe du prototype ;

- le modèle nous a permis d'appréhender la complexité de l'architecture de la machine pour en dégager les paramètres essentiels à son évaluation, et son utilisation pour une étude prospective a mis en évidence la sensibilité du comportement de la machine TREFLE à la variation de ses principaux paramètres ;

- l'évaluateur nous permet de prélever les valeurs caractérisant le fonctionnement du prototype pour en alimenter le modèle dans le but de le valider et de l'exploiter.

L'intérêt de l'approche que nous avons suivie pour la conception et la réalisation de l'évaluateur spécifique à la machine TREFLE ne se limite pas à cette étude particulière car la microprogrammation peut apporter plus de souplesse à la mise en oeuvre des outils de mesure.

En effet, les techniques de microprogrammation permettent d'étendre les fonctions du processeur qu'on désire évaluer :

- par l'adjonction d'instructions orientées vers la construction d'évaluateurs, par exemple des instructions "sondes", des instructions "collectrices", etc...

- par l'augmentation de la fonction des instructions de communication, par exemple les instructions d'appel au moniteur, les instructions d'entrées/sorties et les instructions de synchronisation, dans le but de faciliter l'accès de l'évaluateur aux valeurs représentatives de l'activité du système.

L'intérêt de la méthode de prélèvement de mesure que nous proposons est accru par les possibilités de microprogrammation offertes par de nombreux calculateurs actuels.

CONCLUSION

Nous avons présenté dans la première partie de ce mémoire le projet TREFLE qui propose une nouvelle architecture de machine informatique fondée sur la notion d'automate à pile. Le prototype construit est orienté vers la consultation de données textuelles, application utilisant les grammaires à contexte libre pour lesquelles cette classe de processeur est particulièrement adaptée.

Les prolongements éventuels de l'étude qui a conduit à la réalisation de la machine TREFLE sont d'une part d'améliorer la version actuelle en ce qui concerne particulièrement l'interface logiciel entre l'utilisateur et la machine et d'autre part à élaborer une architecture facilement intégrable sous la forme d'un circuit à grande échelle d'intégration (VLSI). Les applications visent à court terme les organismes de documentation qui pourront utiliser TREFLE dans sa conception et sa forme actuelle : celle d'une machine de la taille et du coût d'un mini-ordinateur. A moyen terme il s'agit d'applications grand public, d'un prix abordable ce qui suppose la réalisation de TREFLE sous la forme d'un microprocesseur. L'étude que nous poursuivons actuellement avec le CCETT a pour but d'implanter TREFLE dans les téléviseurs.

La seconde partie de ce mémoire décrit en détail les solutions proposées pour résoudre les problumes de communications entre la machine TREFLE et son environnement. Elle a permis de montrer l'intérêt de la microprogrammation comme méthode de conception de processeurs spécialisés et comme technique de réalisation de mécanismes de communication.

La troisième partie a présenté la démarche utilisée pour aborder l'évaluation du comportement d'une architecture de cette complexité.

L'intérêt de ces travaux est multiple :

- la modélisation par un modèle à réseau de files d'attente s'est avérée particulièrement adaptée au problème posé par l'évaluation de l'architecture de TREFLE. En effet, cette classe de modèle permet la prise en compte de la structure du prototype dans lequel la notion de file d'attente existe physiquement. Cette phase de modélisation a débouché sur une vision plus abstraite du fonctionnement de la machine TREFLE ce qui a permis à ses concepteurs de mieux appréhender son fonctionnement global ;

- la mise en oeuvre d'un évaluateur adapté aux problèmes spécifiques posés par les processeurs à évaluer nous a permis d'affiner et de valider nos idées personnelles sur la conception des outils de mesure. La technique d'observation développée pour la machine TREFLE allie une grande simplicité de réalisation à une grande efficacité et sa transposition sur d'autres architectures nous paraît aisément envisageable ;

- les premières expérimentations ont eu pour but de prospecter les plages des valeurs des paramètres du modèle. Comme nous désirions rester proche de la réalité en ce qui concerne les lois d'arrivées et les lois de service, nous avons opté pour une résolution du modèle par simulation. Néanmoins, une étude analytique permettrait de confirmer la robustesse de notre modèle en quantifiant ses réactions face à des lois d'arrivée et des lois de service totalement différentes.

Par ailleurs, nous tenons à souligner l'intérêt pour une équipe d'architectes de machines à collaborer avec un spécialiste de l'évaluation. En effet, les progrès spectaculaires de la technologie permettent la construction de machines dont la complexité croît constamment. Pour appréhender cette complexité les concepteurs ont besoin d'outils leur apportant une vision globale du comportement de l'architecture qu'ils développent.

Notre enthousiasme pour les domaines abordés dans cette étude ne limitera pas au seul projet TREFLE notre collaboration à l'équipe Systèmes de Consultation de Données.

Nous espérons que d'autres machines succèderont au prototype construit et que ma contribution à leur élaboration s'effectuera avec autant de succès.

BIBLIOGRAPHIE

[Anc 74] ANCEAU F. - "Contribution à l'étude des systèmes hiérarchisés de ressources dans l'architecture des machines informatique".
Thèse d'Etat - Université de Grenoble - Décembre 1974.

[Bac 60] BACKUS J.W. - "Report on the algorithmic language ALGOL 60".
Nümerische Mathematik 2 - 1960.

[BaC 75] BASKETT F., CHANDY K.M., MUNTZ R.R., PALACIOS F.G. - "Open, closed and mixed networks of queues with different classes of customers".
JACM - Vol. 22, n° 2 - Avril 1975.

[BeK 74] BETOURNE C., KRAKOWIAK S. - "Mesures sur un système conversationnel".
Rapport IRIA-LABORIA - n° 46 - Janvier 1974.

[Ben 73] BENNET M.J. - "A virtual evaluation tool for a system having a virtualizeable processor" -
SIGME SYMPOSIUM - Palo Alto - 1973.

[Ben 75] BENNET M.J. - "Task Behaviour Monitor".
contrat SESORI n° 74127 - 1975.

[Bio 76] BIONDI J. - "Application de l'analyse des données à la description de la charge d'un calculateur multiprogrammé en vue de l'amélioration de ses performances dans un traitement par lots" -
Thèse de 3ème cycle - Université de Nice - Novembre 1976.

[BoC 75] BOI L., CROS P., DRUCBERT J.P., BOURRET P., TREPOS R. - "Evaluation des performances du système CII-SIRIS 7 - Présentation des résultats et conclusions de l'évaluation" -
Rapport final de contrat IRIA-SESORI - Mars 1975.

[BoC 76] BOI L., CROS P., DRUCBERT J.P., ROUSSELOT J.Y., BOURRET P., TREPOS R.- "A performance evaluation of the CII-SIRIS 8 operating system - methodology, tools and first results" -Workshop on modelling and performance evaluation of computer systems - Stresa - Italie - Octobre 1976

[BoD 73] BOI L., DRUCBERT J.P. - "OSSYOSCOPE : system on auxiliary processors for measuring operating systems" - International computing symposium 1973 - Davos - Suisse - Septembre 1973.

[Bor 71] BORDSEN D.T. - "Univac 1108 instrumentation system" - SIGOPS - ACM - Workshop on system performance evaluation - Harward University - Avril 1971.

[Bor 76] BORRIONE D. - "LASCAR, un langage pour la simulation et l'évaluation des architectures d'ordinateurs" - Thèse de 3ème cycle - Université de Grenoble - Avril 1976.

[Bou 78] BOUTRY D. - "APL2M - Manuel de Référence" - GIXI - 1978.

[BrB 74] BRANDWAJN A., BUZEN J., GELENBE E., POTIER D. - "A model of performance for virtual memory systems" - Proceedings of ACM - SIGMETRICS 74 - Montreal - Octobre 1974.

[BuI 77] BURGEVIN P., INGELS Ph. - "Mesures de comportements de programmes en vue de la modélisation de systèmes informatiques" . Thèse de Docteur Ingénieur - Université de Rennes - Mai 1977.

[BuM 72] BULLEN R.H.Jr, MILLER J.K. - "Microtext - the design of a microprogrammed finite state search machine for full text retrieval". AFIPS Conf. Proc. FJCC - 1972.

[Buz 76] BUZEN J.P. - "Fundemental laws of computer system performance". Proceedings of international Symposium on Computer Performance Modeling and Evaluation - Harvard University - Mars 1976.

[CaE 68] CANTRELL H.N., ELLISON A.L.- "Multiprogramming system performance measurement and analysis" - AFIPS. SJCC - 1968.

[CaH 68] CAMPBELL D.J., HEFFNER W.J. - "Measurement and analysis of large operating systems during system development" . AFIPS. FJCC - 1968.

[ChH 75] CHANDY K.M., HERZOG U., WOO L. - "Approximate analysis of general queueing networks" - IBM Journal of Research and Development - Janvier 1975.

[Cod 70] CODD E.F. - "A relationnal model of data for large shared Data Banks". CACM vol. 13 n° 6 - 1970.

[CoG 70] COHEN, GOTLIEB - "A list structure form of grammars for syntactic analysis" - Computing Surveys, vol. 2, n° 1 - Mars 1970.

[COM 79] COMPUTER Vol. 12, n° 3 - IEEE COMPUTER SOCIETY - numéro spécial DATA BASE MACHINES - Mars 1979.

[Cop 73] COPELAND G.P., LIPOVSKI G.J., SU S.Y.W. - "The architecture of CASSM : A cellular system for non numeric processing" . Proceedings 1st Annual Symp. on Computer Architecture - p. 121, 128 Décembre 1973.

[Cop 78] COPELAND G.P. - "String Storage and searching for Data base applications : implementation on the INDY Back-end kernel". - Proceedings 4th Workshop on Computer Architecture for non numeric processing - Aout 1978.

[Cox 72] COX D.R., MILLER H.D. - "The theory of stochastic processes" . Chapman and Hall - London - 1972.

[Cra 74] CRANE M.A., IGLEHARDT D.L. - "Simulating stable stochastic".
JACM - Vol. 21 , n° 1 - Janvier 1974.

[Cro 75] CROCUS - "Systèmes d'exploitation des ordinateurs".
Dunod - 1975.

[Dah 66] DAHL O.J., NYGAARD K. - "SIMULA - an Algol based simulation language".
CACM - Vol. 9, n° 9 - Septembre 1966.

[Den 73] DENNY W.M. - "Microprogramming measurement techniques for the BURROUGHS B1700" -
Lecture Notes in Computer Science n° 26 - 1973.

[Dew 72] DEWAN P.B., DONACHEY C.E., WYATT J.B. - "OSSL - A specialized language for simulating computer systems" -
Spring joint computer conference - 1972.

[DDH 72] DAHL O.J., DIJKSTRA E.W., HOARE C.A.R. - "Structured Programming".
Academic Press - 1972.

[GaR 78] GAUDEUL B., ROHMER J. - "Le module de calcul" .
Note interne du projet TREFLE - 1978.

[Gau 75] GAUDEUL B. - "HARMONIE" -
Note technique - IRIA-STI - Juin 1975.

[GeP 77] GELENBE E., PUJOLLE G. - "A diffusion Model for multiple class queueing networks".
Rapport de recherche n° 142 - IRIA-LABORIA - Août 1977.

[Gre 66] GREEN J. - "Microprogramming, Emulators and Programming Languages".
CACM Vol. 9 , n° 3 - Mars 1966.

[GrL 67] GROSS M., LENTIN A. - "Notions sur les grammaires formelles".
Gauthier-Villars - 1967.

[Guy 75] GUYOT A. - "Un compilateur de microprogrammes".
Thèse de 3ème cycle - Université de Grenoble - 1975.

[HaM 76] HATON J.P., MOHR R. - "A parsing algorithm for imperfect patterns and its application".
Proceedings of 3d Int. joint Conf. on Pattern Recognition - San Diego - Novembre 1976.

[Hel 74] HELNESS K.M. - "Implementation of a Parallel Cyclic Redundancy check generator" .
Computer Design - Mars 1974.

[Hon 78] HONIG H.P. - "Data path analysis : analysing large I/O environments".
ACM Sigmetrics PER Vol. 7, n° 1, 2 - 1978.

[Hus 70] HUSSON S.S. - "Microprogramming Principles and Practices" -
Prentice Hall - 1970.

[IBM 67] IBM Corp. - "General Purpose Simulation System/360 - Users Manual".
IBM Corp. - H20 - 0326.0 - 1967.

[IGR 71] INSTITUT GUSTAVE ROUSSY.- "SABIR-C : Système Automatique de Bibliographie, d'Information et de Recherche en Carcinologie".
CANCER - Octobre 1971.

[IOP 75] "IOP-MITRA 125" - Produit 1961 - SEMS - 1975.

[Ive 62] IVERSON K.E. - "A Programming Language".
John Wiley and Sons - New York - 1962.

[Jai 78] JAIN A.K. - "A guideline to statistical approaches in computer performance evaluation studies" -
ACM SIGMETRICS PER Vol. 7 n° 1, 2 - 1978.

[JaL 74] JALICS P.J., LYNCH W.C. - "Selected measurement of the PDP - 10 TOPS - 10 time sharing operating system" -
Proceedings of IFIP 74 Conference - Stockolm - Août 1974.

[Jar 75] JARAY J. - "Le langage SNOBOL 4, ses applications, son implémentation".
Thèse de 3ème cycle - Université de Nancy - 1975.

[Kee 68] KEEFE D.D. - "Hierarchical control programs for systems evaluation".
IBM System Journal n° 2 - 1968.

[Lef 72] LEFEBVRE P. - "SPY : un système de contrôle. La mise au point de systèmes de programmation grâce à un couplage de machines virtuelles ".
Thèse de 3ème cycle - Université de Grenoble - Juin 1972.

[LeP 76] LEROUDIER J., PARENT M. - "Discrète event simulation modelling of computer systems for performance evaluation ".
Rapport IRIA-LABORIA n° 177 - Juin 1976.

[Ler 73] LEROUDIER J. - "Une analyse de système ".
Thèse de 3ème cycle - Université de Grenoble - Avril 1973.

[Ler 77] LEROUDIER J. - "Systèmes adaptatifs à mémoire virtuelle".
Thèse d'Etat - Université de Grenoble - Mai 1977.

[Mar 62] Association MARC BLOCH - "Etude d'un système général de documentation automatique".
Publication EURATOM - 1962.

[Mar 66] MARKOWITZ H.M. - "Simulating with SIMSCRIPT".
Management Science - Vol. 12, n° 10 - Juin 1966.

[MaR 78] EL MASRI A., ROHMER J., TUSERA D. - "A machine for Information Retrieval".
Proceedings 4th Workshop on Computer Architecture for non numeric processing -Août 1978.

[Mas 77] EL MASRI A. - "Insertion du module de classement dans la machine TREFLE".
Note interne du projet TREFLE - Mars 1977.

[Mas 78] MASINI G. - "Réalisation d'un système de reconnaissance structurelle et d'interprétation de dessins".
Thèse de 3ème cycle - Université de Nancy - Octobre 1978.

[Maz 78] MAZARE G. - "Structures multimicro-processeurs. Problèmes de parallélismes. Définition et évaluation d'un système particulier".
Thèse d'Etat - Université de Grenoble - Juin 1978.

[MeP 78] MERLE D., POTIER D., VERAN M. - "A tool for computer performance analysis ".
Proc. ICPCI - Juin 1978.

[MeS 70] MEYER P.A., SEAWRIGHT L.H. - "A virtual machine time - sharing system".
IBM system Journal - Vol. 9, n° 3, 1970.

[Mer 73] MERMET J. - "Etude méthodologique de la conception assitée par ordinateur des systèmes logiques" - Thèse d'Etat - Université de Grenoble.
Avril 1973.

[Mer 78] MERLE D. - "Contribution à l'étude d'un analyseur de modèles à réseaux de files d'attente. Algorithmes de calcul et applications".
Thèse de Docteur Ingénieur - Université de Nancy - Octobre 1978.

[Mig 70] MIGNOT H. - "Recherche documentaire notariale. La réalisation du CRIDON de LYON" -
Informatique et Gestion - Novembre 1970.

[Mis 71] "MISTRAL : Manuel d'utilisation" - CII - HB - Réf : 4017 P/FR - 1971.

[MIT 75] "MITRA 125" - Manuel de présentation - SEMS - 1975.

[Moo 71] MOORE C.G. - "Network models for large scale time-sharing systems".
Ph. D. Thesis - University of Michigan - 1971.

[NoS 74] NOLAN L.E., STRAUSS J.C. - "Workload characterization for time sharing system selection".
Software Practice and Experience . Vol. 4 - 1974.

[OzS 76] OZKARAHN E.A., SCHUSTER S.A., SMITH K.C. - "RAP - An associative processor for data base management" - Proc. AFIPS Ncc Vol. 44 - 1976.

[Pai 65] PAIR C.- "Etude de la notion de pile. Application à l'analyse syntaxique".
Thèse d'Etat - Université de Nancy - Décembre 1965.

[Rob 75] ROBICHAUD L.P.A. et Al. - "Interactive tools for research and education in microprogramming".
EUROMICRO - Nice - 1975.

[Rob 78] ROBERTS D.C. - "A specialized computer architecture for text retrieval".
4th workshop on computer architecture for non numeric processing - Août 1978.

[Roh 79] ROHMER J. A.P.L. - "The APL2M System : language and code generation for mini and micro computer are straight forward with an APL based cross software".
APL 79 - New York - Juin 1979.

[Ruu 72] RUUD R.J. - "The CPM-X a system approach to performance measurement".
AFIPS Conf. Proc. FJCC - p. 949-957 - 1972.

[Ros 73] ROSIN R.F. - "The significance of microprogramming" -
ICS - 1973.

[SaS 72] SAAL H.J., SHUSTECK L.J. - "Microprogrammed implementation of computer measurement".
SLAC Publications - CS 294 - Juillet 1972.

[Sch 67] SCHERR A.L. - "An analysis of time shared computer systems".
MIT Press Cambridge MASS - 1967.

[Schu 67] SCHULMAN F.D. - "Hardware measurement device for IBM System 360 Time sharing evaluation" -
Proceedings 22nd - ACM Conf. - 1967.

[Sea 69] SEAMAN P.H., SOUCY R.C. - "Simulating operating systems".
IBM System Journal n° 4 - 1969.

[Soc 74] "SOCRATE" . Manuel d'utilisation - CII-HB - 1974.

[Ste 76] STEWART W.J. - "MARCA : Markov chain Analyser".
IRISA - Rapport n° 45 - Juin 1976.

[Swa 75] SWANSON R. - "Understanding Cyclic Redundancy Codes".
Computer Design - Novembre 1975.

[Tus 77] TUSERA D. - "Le Filtre".
Note interne au projet TREFLE - Juin 1977.

[Tus 78] TUSERA D. - "Le Formatteur" -
Note interne au projet TREFLE - Janvier 1978.

[Wil 51] WILKES M.V. - "The best way to design an automatic calculating machine".
Manchester University - Comp. Inaugural Conf. - 1951.

[Rot 76] ROHMER J., TUSERA D. - "Structures Matérielles pour le filtrage" .
Note interne au projet SCD. Décembre 1976.

[Wil 69] WILKES M.V. - "The growth of interest in microprogramming ".
Comp. Surv. 1, 3 - Septembre 1969.

[Wir 76] WIRTH N. - "Algorithms + data structures = programs".
Prentice Hall - 1976.

BIBLIOTHEQUE NANCY

ANNEXES

A - LES ENTREES/SORTIES

B - L'IOP

C - LA CONNEXION

D - LE DISQUE

E - LE FORMATTEUR

F - LE MODELE

A- Les Entrées/Sorties

I - LES CONNEXIONS

Le réseau de bus reliant l'unité centrale, la mémoire centrale et l'IOP est formé de 3 bus :

- le bus de donnée (16 bits), unidirectionnel, de la mémoire vers l'IOP,

- le bus d'adresse et de donnée (20 bits), unidirectionnel, de l'IOP vers la mémoire,

- le bus fonctionnel (7 bits), bidirectionnel entre l'unité centrale et l'IOP. Il est composé de 7 fils ayant chacun un rôle particulier :

  + les signaux issus de l'unité centrale :
    . SIO : signal d'activation, généré par l'éxécution sur le calculateur d'une instruction SIO
    . HIO : signal d'arrêt, généré par l'éxécution sur le calculateur d'une instruction HIO
    . CS : signal transmettant la coupure secteur
    . DIT et PIT : signaux de demande d'interruption et de suspension.

  + les signaux issus de l'IOP :
    . IT : interruption indiquant à l'unité centrale la fin d'un transfert
    . ADES : signal d'acquittement d'une demande de transfert lancée par l'unité centrale.

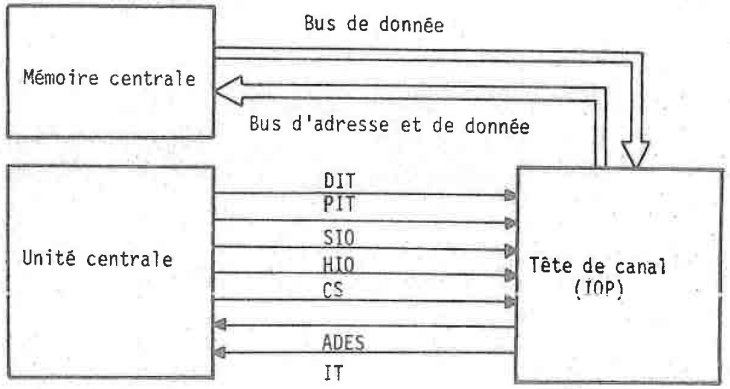


Figure A-1 : Les voies de communication entre la mémoire centrale, l'unité centrale et la tête de canal de l'IOP

II - LES TABLES D'ENTREE/SORTIE

L'interface entre le matériel et le logiciel est composé d'un ensemble de tables chaînées accessibles à l'unité centrale et à l'IOP. Cette structure est gérée par les procédures système, les microprogrammes d'interprétation des instructions d'Entrée/Sortie sur l'unité centrale et les microprogrammes interprétant le programme canal sur l'IOP.

La structure de donnée utilise tois types de tables : table des contrôleurs, table des unités et programme canal ainsi que les pointeurs permettant l'accès à ces tables (cf figure A-2).

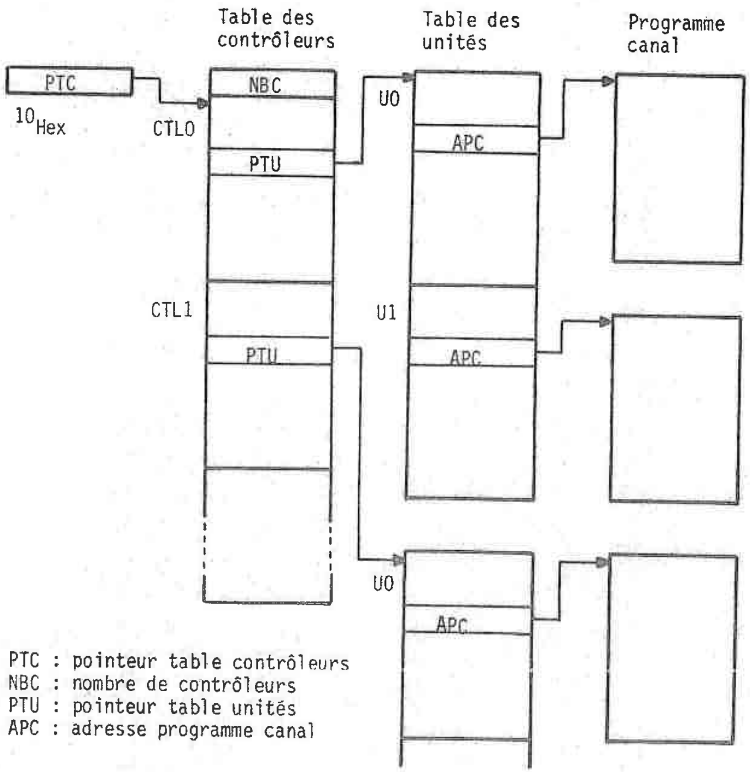


Figure A-2 : La structure des tables de communication entre le matériel et le logiciel d'entrée/sortie

2.1. LA TABLE DES CONTROLEURS

Cette table décrit les contrôleurs (IOP) des unités périphériques. Elle a autant d'éléments que de contrôleurs présents dans le système. Elle est adressée par le pointeur PTC se trouvant à l'adresse 10 Hexadécimal et le mot qui précède la première entrée contient le nombre d'entrées de la table (NBC). Chaque élément est indexé par le numéro du contrôleur multiplié par 32, et contient 16 mots regroupant les informations suivantes : (cf figure A-3)

Mot 00 : bits 0-7 : réservé
bits 8-15 : NBU : nombre d'unités rattachées à ce contrôleur

Mot 02 : bits 0-15 : PTU : pointeur vers la table des unités

Mot 04 : bits 0-15 : VQ : verrous des listes SIO et HIO

VQ = 0 : interdiction d'accès aux listes
VQ ≠ 0 : autorisation d'accès pour lecture et modification.
Au moment de la lecture il faut mettre VQ à 0 pour empêcher l'autre processeur d'accéder aux listes.

Mot 06 : bits 0-7 : TSIO : tête de la liste des SIO : numéro de la première unité de la liste des SIO
bits 8-15 : QSIO : queue de la liste des SIO : numéro de la dernière unité de la liste des SIO

Mot 08 : bits 0-7 : THIO : tête de la liste des HIO
bits 8-15 : QHIO : queue de la liste des SIO
} cf ci-dessus

Mots 0A à 1E : message d'entrée/sortie : ce message est composé de l'ensemble des informations rendues par l'IOP à la fin de l'exécution d'un programme canal.

Mot 0A : bits 0-7 : NC : numéro du contrôleur
bits 8-15 : NUIT : numéro de l'unité ayant généré une interruption

Mot 0C-0E : ACCIT : adresse de la commande canal ayant demandé l'interruption
0C : Base
0E : Déplacement

Mot 10 : ETAT : mot d'état résumant l'entrée/sortie

Mot 12-1E : MSG : message complémentaire propre à chaque contrôleur en fonction des unités gérées.

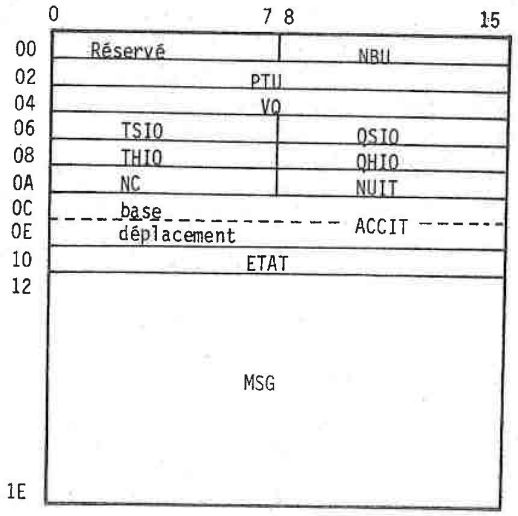


Figure A-3 : Elément de la table des contrôleurs

Remarque : Du fait que chaque contrôleur peut gérer jusqu'à quatre unités périphériques, il est possible que dans un intervalle de temps très court il lui parvienne quatre demandes d'Entrée/Sortie. Il doit donc être en mesure de les mémoriser afin de les traiter ultérieurement. Cette mémorisation s'effectue dans les listes SIO et HIO et le verrou VQ assure la mutuelle exclusion entre l'unité centrale et l'IOP pour l'accès à ces listes.

2.2. LA TABLE DES UNITES

La table des unités décrit les unités attachées à un contrôleur (IOP). Elle a autant d'éléments que d'unités présentes sur le contrôleur. Elle est adressée par le contenu du pointeur PTU de la table des contrôleurs avec une base nulle. Chaque élément est indexé par le numéro de l'unité multiplié par 32 et contient 16 mots regroupant les informations suivantes : (cf figure A-4)

Mot 00 : bits 0-7 : INDU : mot d'état de l'unité
bits 8-15 : réservé : extension du mot d'état

Mot 02 : bits 0-7 : LIENSIO : numéro de l'unité qui suit celle-ci dans la liste SIO. La valeur FF indique que l'unité est la dernière de la liste des SIO.
bits 8-15 : LIENHIO : numéro de l'unité qui suit celle-ci dans la liste HIO. La valeur FF indique que l'unité est la dernière de la liste des HIO.

Mots 04-06 : APC : adresse du programme canal
mot 04 : base
mot 06 : déplacement

Mots 07-1E : RUH : zone réservée au programme de gestion de l'unité (Handler)

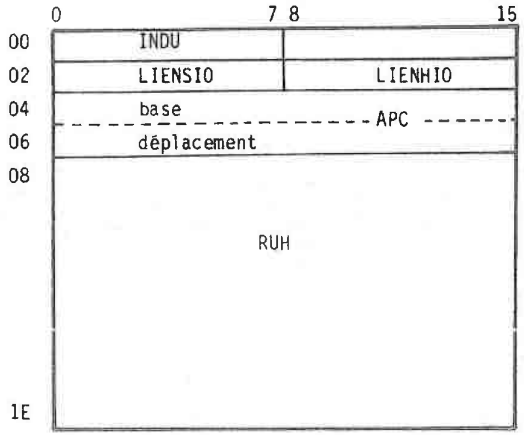


Figure A-4 : Elément de la table des unités

2.3 LE PROGRAMME CANAL

Le programme canal, constitué par plusieurs commandes chaînées, décrit le transfert à effectuer par une unité. Il est adressé par le pointeur APC de la table des unités et est interprété par les microprogrammes de l'IOP. Chaque commande contient 4 mots regroupant les informations suivantes : (cf figure A-5)

Mot 00 : bits 0-7 : Ordre : code opération de la requête
bits 8-15 : Indicateur : indicateurs permettant le chaînage des commandes ou des données et la protection mémoire

Mot 02-04 : Données : double mot contenant les paramètres nécessaires à l'interprétation du code opération

Mot 06 : Compte d'octets : nombre d'octets à transférer pour la requête

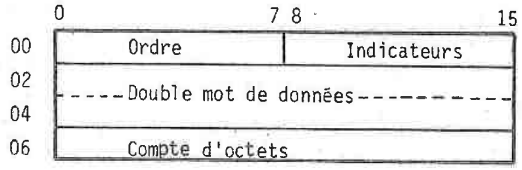


Figure A-5 : Programme canal

B- L'I.O.P.

I - LA TETE DE CANAL

La partie opérative de la Tête de canal est constituée de trois ensembles :

- les voies de liaison avec la mémoire
- les interfaces
- la gestion du séquencement

1.1. LES VOIES DE LIAISON AVEC LA MEMOIRE (cf figure B-1)

Cette partie est composée des voies sur lesquelles transitent les informations à destination ou en provenance de la mémoire.

+ Chemin de l'adresse mémoire :

L'adresse mémoire est formée des 20 bits résultant de l'addition des 16 bits de la base décalés à gauche de 3 positions, et des 16 bits du déplacement. Les 19 bits de poids fort permettent l'accès au mot référencé, le 20ème bit donnant la parité de l'octet de ce mot.

+ Chemin des données vers la mémoire :

La transformation du chemin de largeur 8 bits de l'organe micro-programmé en chemin de 16 bits pour la mémoire sera faite à l'aide d'un multiplexeur relié à un registre de 16 bits. Deux cycles de l'organe microprogrammé chargent le registre qui est alors envoyé vers la mémoire sur le même bus que l'adresse.

+ Chemin des données venant de la mémoire :

Les données arrivent sur le bus de 16 bits et ne sont utilisées que par octet. La transformation parallèle-série sera faite par un registre de 16 bits relié à un multiplexeur. A chaque cycle on dégarnira alternativement la partie gauche ou la partie droite du registre.

1.2. LES INTERFACES

La Tête de canal est reliée à trois modules fonctionnels par des interfaces spécialisés :

+ L'interface avec le canal d'accès mémoire :

Le canal d'accès mémoire est composé de 3 bus :
- le bus LS unidirectionnel de largeur 20 bits sert à transférer depuis la tête de canal les données et les adresses vers la mémoire.

- le bus LE unidirectionnel de largeur 16 bits véhicule les données venant de la mémoire.
- le bus fonction assure la synchronisation entre la mémoire et la tête de canal. En effet, la mémoire est toujours maîtresse dans les échanges et la tête de canal ne peut émettre vers elle que si sa requête est acceptée.

+ L'interface avec le bus d'Entrée/Sortie de l'unité centrale :

Les signaux de synchronisation entre l'IOP et l'unité centrale transitent sur un bus fonction formé de 7 fils unidirectionnels, chacun ayant une signification propre :

- deux fils permettent à l'IOP d'interrompre l'unité centrale :
  . un fil de suspension pour acquitter la requête SIO
  . un fil d'interruption pour signaler la fin d'un transfert

- cinq fils permettent à l'unité centrale de positionner des indicateurs dans la tête de canal, car l'IOP ne possède pas de système d'interruption .
  Ces fils transmettent les requêtes suivantes :
  . SIO : demande d'entrée/sortie
  . HIO : arrêt d'entrée/sortie
  . IT31 : coupure secteur
  . PIT : requête de suspension
  . DIT : requête d'interruption

+ L'interface avec l'organe microprogrammé :

L'organe microprogrammé dispose de 32 voies de lecture/écriture pour commander la partie opérative de la tête de canal. Inversement la tête de canal positionne 3 indicateurs auxquels le microprogramme de l'organe microprogrammé peut accéder.

Les voies de lecture/écriture permettent au microprogramme de charger les registres de la partie opérative et d'activer la partie commande de la tête de canal.
Liste des voies utilisées pour la communication organe microprogrammé-tête de canal :

- en lecture : les données vont de la tête de canal à l'organe microprogrammé
  . LC00 : lecture des données depuis le registre donnée en entrée DOE
  . LC01 : lecture du numéro du contrôleur
  . LC02 } lecture du compte d'octets, poids forts et poids faibles
  . LC03 }
  . LC04 : lecture des bascules SIO, HIO, IT31, Etat tête de canal
  . LC05 : lecture de l'état mémoire

- en écriture : les données sont chargées dans les registres de la tête de canal depuis l'organe microprogrammé
  . EC11 } chargement du registre de base poids forts, poids faibles
  . EC12 }
  . EC13 } chargement du déplacement et de la bascule Sélection
  . EC14 } Octet poids forts, poids faibles
  . EC15 } chargement du compte d'octets poids forts, poids faibles
  . EC16 }
  . EC17 : chargement de l'octet de mode et lancement de l'automate propre à la tête de canal
  . EC18 : remise à zéro de la bascule SIO
  . EC19 : remise à zéro de la bascule HIO
  . EC1A : positionnement interruption
  . EC1E : chargement du registre donnée en sortie DOS

Liste des indicateurs utilisés pour la communication organe microprogrammé-tête de canal :

numéro de l'indicateur	origine	nom	fonction
13	mémoire	ERTK	mémorise l'occurence d'une erreur lors d'un échange avec la mémoire
14	unité centrale	HIO	mémorise l'arrivée d'une requête HIO
15	tête de canal compte d'octets	COZ	mémorise le passage à zéro du compte d'octets

1.3. LA GESTION DU SEQUENCEMENT ET DU COMPTEUR D'OCTETS

La tête de canal possède sa propre partie contrôle activée par l'organe microprogrammé au moment du chargement de l'octet de mode (voie EC17). A partir de cet instant, la tête de canal devient autonome par rapport à l'organe microprogrammé dont le rôle sera de surveiller les indicateurs affichés en permanence. La logique de la tête de canal, inscrite en mémoire morte, lui permet de gérer complétement les échanges avec la mémoire centrale.

Description de l'octet de mode ("instruction" interprétée par la tête de canal) :

N° bit	Valeur	Fonction activée
0	1	Lecture de données de la mémoire
	0	Ecriture de données vers la mémoire
1	1	Ecriture et lecture normale
	0	Ecriture de zéros
2	1	Incrémentation automatique des adresses
	0	Décrémentation automatique des adresses
3	1	Passe partout pour les zones protégées
	0	Protection écriture respectée
4	1	Adresse memoire modifiable
	0	Adresse mémoire fixe
5	1	Validation des phases mémoire
	0	Blocage du dialogue avec la mémoire
6	inutilisé	
7	1	Remise à zéro des erreurs mémoire
	0	Mémorisation des erreurs mémoire

L'octet de mode contient tous les renseignements nécessaires à l'échange. La logique de séquencement aura pour rôles :

- de gérer les adresses en incrémentant ou en décrémentant le déplacement
- de gérer le compte d'octets
- de gérer les erreurs mémoire
- de gérer le registre de donnée en sortie dans le cas d'écriture de zéros en mémoire

L'enchaînement de ces fonctions respecte le schéma suivant :

+ En lecture : transfert de la mémoire vers la tête de canal.

- 1 : Initialisation des signaux
- 2 : Calcul de l'adresse pour l'échange
- 3 : Envoi de l'adresse
- 4 : Attente des données en provenance de la mémoire
- 5 : Chargement du registre Donnée Mémoire en Entrée (DME)
- 6 : Décrémentation et test du compte d'octet - si 0 alors fin du transfert
- 7 : Incrémentation ou décrémentation de l'adresse (en fonction du mode d'adressage)
- 8 : Attente de deux lectures consécutives du registre DOE par l'organe microprogrammé
- 9 : Aller en 3

+ En écriture : transfert de la tête de canal vers la mémoire.

- 1 : Initialisation des signaux
- 2 : Calcul de l'adresse pour l'échange
- 3 : Envoi de l'adresse
- 4 : Envoi du registre Donnée Mémoire en Sortie (DMS) ou zéro (en fonction du mode d'écriture)
- 5 : Incrémentation ou décrémentation de l'adresse (en fonction du mode d'adressage)
- 6 : Décrémentation et test du compte d'octets - si 0 alors fin du transfert
- 7 : Attente de deux chargements consécutifs du registre DOS par l'organe microprogrammé
- 8 : Aller en 3

Remarque :

Lorsque l'organe microprogrammé lit (voie LC00) ou charge (voie EC1E) les registres de Données (DOE et DOS respectivement), il le fait à raison de 8 bits à la fois. Il faut donc deux ordres de lecture ou de chargement pour que la logique de la tête de canal puisse regarnir ou vider ses registres (DME et DMS respectivement) en lançant un échange avec la mémoire. On remarquera que le garnissage de DMS avec DOS et de DOE avec DME est automatique. La logique de séquencement de la tête de canal assure le multiplexage temporel des données pour gérer efficacement la transformation 8 bits - 8 bits série en 16 bits parrallèle.

Figure B-1 : Synoptique de la partie opérative de la tête de canal

II - L'ORGANE MICROPROGRAMME

Nous décrivons dans ce paragraphe l'organe microprogrammé sous trois aspects différents :

- la partie opérative
- la partie contrôle
- le langage de microprogrammation

2.1. LA PARTIE OPERATIVE

Nous présenterons successivement les différents éléments composant la partie opérative :

- les opérateurs
- les registres
- la mémoire locale
- les indicateurs

+ Les opérateurs :

La structure de la partie opérative est centrée sur l'utilisation de deux opérateurs :
- un opérateur universel (UAL) opérant sur 8 bits
- un opérateur de décalage opérant sur 16 bits

Ces deux opérateurs sont intégrés dans un motif de type "tourbillonnaire" où deux accumulateurs et une mémoire locale complètent l'architecture. (cf figure B-2)

+ Les registres :

L'ensemble des registres peut être classé en trois catégories :

- les accumulateurs :
  . l'accumulateur A (8 bits) : il contient les opérandes pour les deux opérateurs
  . l'accumulateur B (8 bits) : il contient les opérandes pour l'Unité Arithmétique et Logique exclusivement
- les registres à décalage :
  . les registres C et D (8 bits chacun) : ils forment l'opérateur de décalage
    - séparés, ils permettent les décalages sur 8 bits
    - associés, ils permettent les décalages sur 16 bits
- les registres d'adresse :
  . le registre T (8 bits) : registre adresse pour la mémoire locale, il possède en outre une entrée de comptage
  . le registre U (4 bits) : registre adresse de page pour la mémoire locale

+ La mémoire locale :

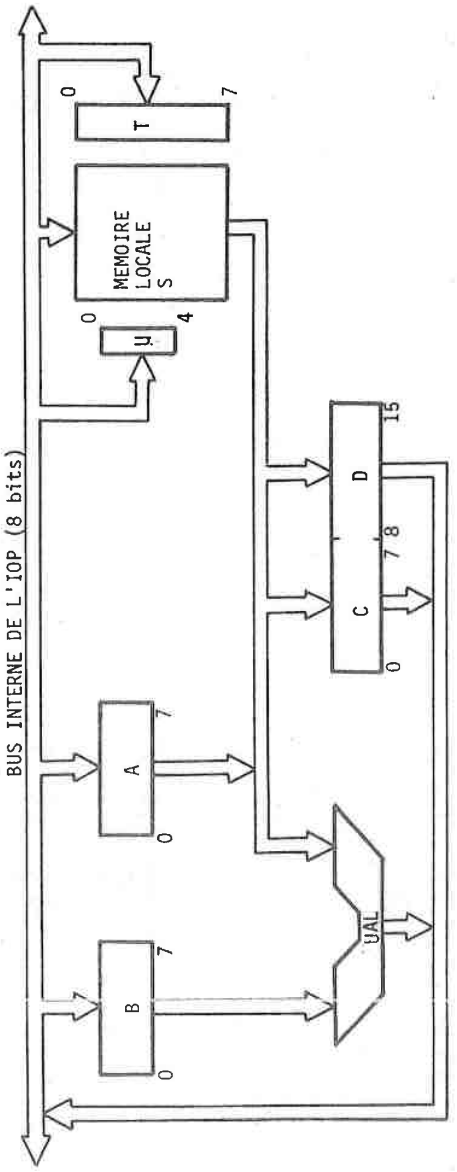
La mémoire locale S contient 256 registres de 8 bits divisés en 16 pages de 16 registres. Trois modes d'adressage permettent d'y accéder :

- le mode ST : adressage absolu, le registre concerné a pour adresse le contenu de T;
- le mode SU : adressage relatif par rapport à U, le registre concerné a pour adresse de page le contenu de U et pour déplacement la valeur d'un champ de la microinstruction;
- le mode SM : adressage relatif par rapport à T, le registre concerné a pour adresse de page le contenu des poids forts de T et pour déplacement la valeur d'un champ de la microinstruction.

+ Les indicateurs :

Seize indicateurs de un bit chacun sont à la disposition de la partie contrôle. Ces indicateurs ont trois origines possibles :

- 3 indicateurs proviennent de la tête de canal :
  . n° 13 : ERTK
  . n° 14 : HIO
  . n° 15 : COZ
- 4 indicateurs proviennent de l'adaptateur :
  . n° 0)
  . n° 1) ils sont spécifiques au périphérique
  . n° 2)
  . n° 3)
- 9 indicateurs sont internes à l'organe microprogrammé :
  . n° 4 : ALU00 : bit 0 de l'unité arithmétique et logique
  . n° 5 : ALU07 : bit 7 de l'unité arithmétique et logique
  . n° 6 : NOCARY : inverse du "carry"
  . n° 7 : ALFF : passage à X'FF' de l'UAL
  . n° 8 : I1) indicateurs de microprogramme
  . n° 9 : I2)
  . n° 10 : A00 : bit 0 de l'accumulateur A
  . n° 11 : A07 : bit 7 de l'accumulateur A
  . n° 12 : VRAI : indicateur toujours à "1"



Registres et opérateurs :
A,B : accumulateurs
C,D : registres à decalage
T,U : registres adresse de la mémoire locale
UAL : unité arithmétique et logique

Figure B-2 : Syncptique de la partie opérative de l'organe microprogrammé

Remarque : Le schéma fait ressortir la structure tourbillonnaire de la partie opérative.

2.2. LA PARTIE CONTROLE

La partie contrôle de l'organe microprogrammé est orientée vers une microprogrammation de type vertical implantée dans une mémoire morte. La mémoire de commande contient 156 pages de 16 mots de 16 bits, elle est adressée par le registre AC de 12 bits décomposé en numéro de page (bits 0 à 7) et en adresse dans la page (bits 8 à 11). Le registre M de 16 bits contient la microinstruction éxécutée (cf figure B-3).

Il existe deux modes d'adressage pour la mémoire de commande :

- le mode absolu : ce mode permet d'adresser toute la mémoire. L'adresse chargée dans AC provient de la mémoire locale S ou de l'accumulateur A pour le numéro de la page et de la microinstruction pour le déplacement.
- le mode relatif : ce mode permet l'adressage dans la page courante, seul le déplacement est modifié par AC.

2.3. LE LANGAGE DE MICROINSTRUCTIONS

Les microinstructions de l'organe microprogrammé sont regroupées en quatre familles :

- les microinstructions de transfert et de commande
- les microinstructions de branchement
- les microinstructions d'entrée/sortie
- les microinstructions de traitement sur l'opérateur arithmétique et logique.

Chaque microinstruction est codée sur 16 bits. La logique de décodage des microinstructions est purement combinatoire et permet de faire l'union logique entre plusieurs microinstructions pour en créer de nouvelles; il suffit pour cela que les champs de codage ne soient pas incompatibles.

+ Les microinstructions de transferts simples :

Au nombre de 45 elles ont pour sémantique le transfert de la valeur désignée par "origine" dans le registre nommé "destinataire". L'origine peut être soit un champ de la microinstruction soit un des registres suivants : A, C, D, ST, SU, SM. Le destinataire est choisi parmi l'ensemble des registres : A, B, C, D, ST, SM, T, U. Le codage de ce type de microinstruction est le suivant :

bits 0-2 : origine
bits 6-8 : destinataire
bits 13-15 : compléments de destinataire

+ Les microinstructions de transferts doubles :

Elles sont construites en faisant l'union des microinstructions de transferts simples qui les composent. Elles sont classées en 2 types :

- type : Origine - Destinataire1 - Destinataire2
- type : Origine1 - Destinataire1 - Origine2 - Destinataire2

+ Les microinstructions de commande :

Cette famille regroupe 13 microinstructions opérant sur les registres : décalage, remise à zéro, positionnement et incrémentation sont les principales opérations effectuées; elles sont codées dans le champ compris entre les bits 3 à 8 de la microinstruction.

+ Les microinstructions de transfert avec commande :

Elles sont obtenues par l'union des microinstructions de transfert et des microinstructions de commande.

+ Les microinstructions de branchement :

Il existe deux types de branchements liés aux deux modes d'adressage de la mémoire de commande :

- le branchement relatif ou branchement dans la page : cette microinstruction nécessite 4 paramètres :
  . le numéro de l'indicateur testé
  . la valeur (Vrai ou Faux) testée
  . le déplacement du branchement
  . l'endroit où l'on sauvegarde éventuellement l'adresse page.

  le codage est le suivant :
  . bits 0-3 : code opération : 1110
  . bit 4 : valeur testée : 1 : V , 0 : F
  . bits 5-8 : numéro de l'indicateur testé
  . bits 9-12 : déplacement
  . bits 13-15 : adresse de sauvegarde (A,ST)

- le branchement absolu ou branchement sur la mémoire : le codage est le même que pour la microinstruction de branchement relatif seul le code opération change (1111). De plus, le codage du champ origine - destinataire est plus complet car l'adresse page doit être fournie par un registre (A ou S)

+ Les microinstructions d'Entrée/Sortie :

Ces microinstructions permettent à l'organe microprogrammé de communiquer avec la tête de canal et l'adaptateur. Elles ont pour particularité de commander 32 voies d'entrée/sortie, ce qui permet la combinaison dans une même microinstruction d'une lecture et d'une écriture, depuis l'adaptateur vers la tête de canal ou inversement. Elles sont toujours associées à une microinstruction de transfert simple ou double.

Leur codage est le suivant :
- . bits 0-1 : code opération (01)
- . bit 2 : ordre de lecture
- . bit 3 : ordre d'écriture
- . bits 4-7 : numéro de voie
- . bits 8-15 : origine et destinataire

La signification des différentes voies d'entrée/sortie est décrite dans la présentation de la tête de canal et de l'adaptateur.

+ Les microinstructions de traitement sur l'opérateur arithmétique et logique :

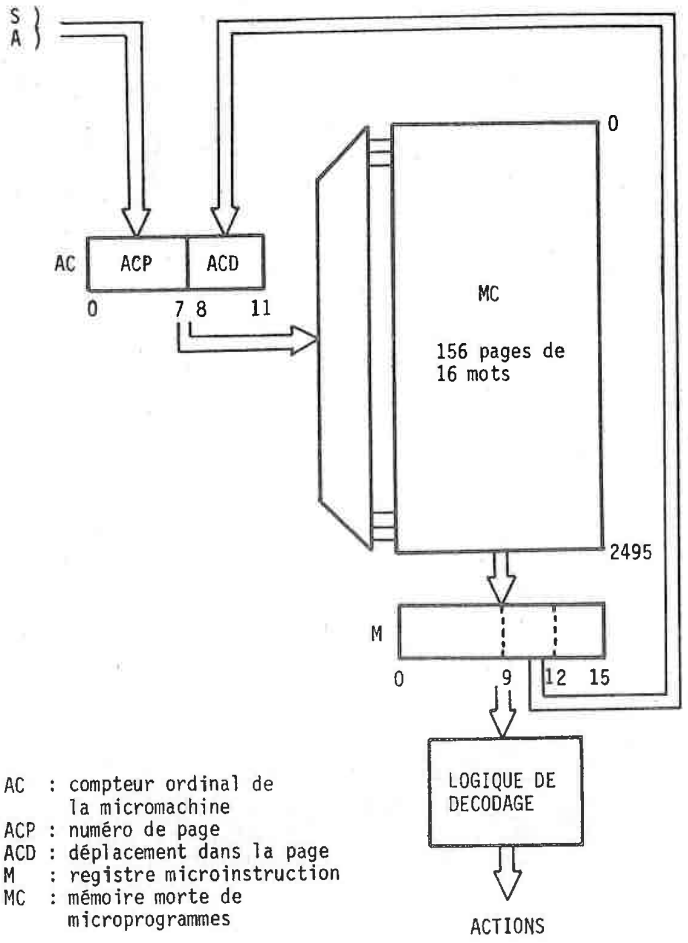
Ces microinstructions se répartissent en trois groupes en fonction de leur action :

- les opérations logiques
- les opérations arithmétiques
- les opérations arithmétiques et logiques

Ces microinstructions nécessitent en général 3 paramètres :

- le code de la fonction : ce code peut être construit par l'union d'un code de fonction arithmétique et d'un code de fonction logique
- l'origine
- le destinataire

Nous rappelons que l'origine ne précise que l'un des opérandes, l'autre étant implicitement le registre B (cf figure B-2)



AC : compteur ordinal de la micromachine
ACP : numéro de page
ACD : déplacement dans la page
M : registre microinstruction
MC : mémoire morte de microprogrammes

Figure B-3 : Partie contrôle de l'organe microprogrammé

NANCY
BIBLIOTHÈQUE

C- La Connexion

I - L'ADAPTATEUR

Le rôle de l'adaptateur est d'assurer l'interface entre l'IOP et le CONTROLEUR. Nous décrivons celui que nous avons conçu sous son aspect opératif et contrôle.

1.1. LA PARTIE OPERATIVE DE L'ADAPTATEUR

La partie opérative de l'adaptateur est composée de trois voies véhiculant les informations :

- la voie des adresses vers la mémoire du contrôleur : cette voie de largeur 16 bits comprend deux registres de 8 bits chacun, ADCTLG et ADCTLD, pouvant être chargés, incrémentés et remis à zéro depuis l'organe microprogrammé.

- la voie des données vers la mémoire du contrôleur : cette voie de largeur 16 bits comprend un registre de 8 bits, COS, un multiplexeur et un registre de 16 bits accessible par octets séparéments, DCS. Cette voie est symétrique à la voie de la tête de canal recevant les données de la mémoire.

- la voie des données venant de la mémoire du contrôleur : cette voie de largeur 16 bits comprend un registre de 8 bits, COE, un multiplexeur et un registre de 16 bits accessible par octets séparément, DCE. Cette voie est symétrique à la voie de la tête de canal émettant les données vers la mémoire.

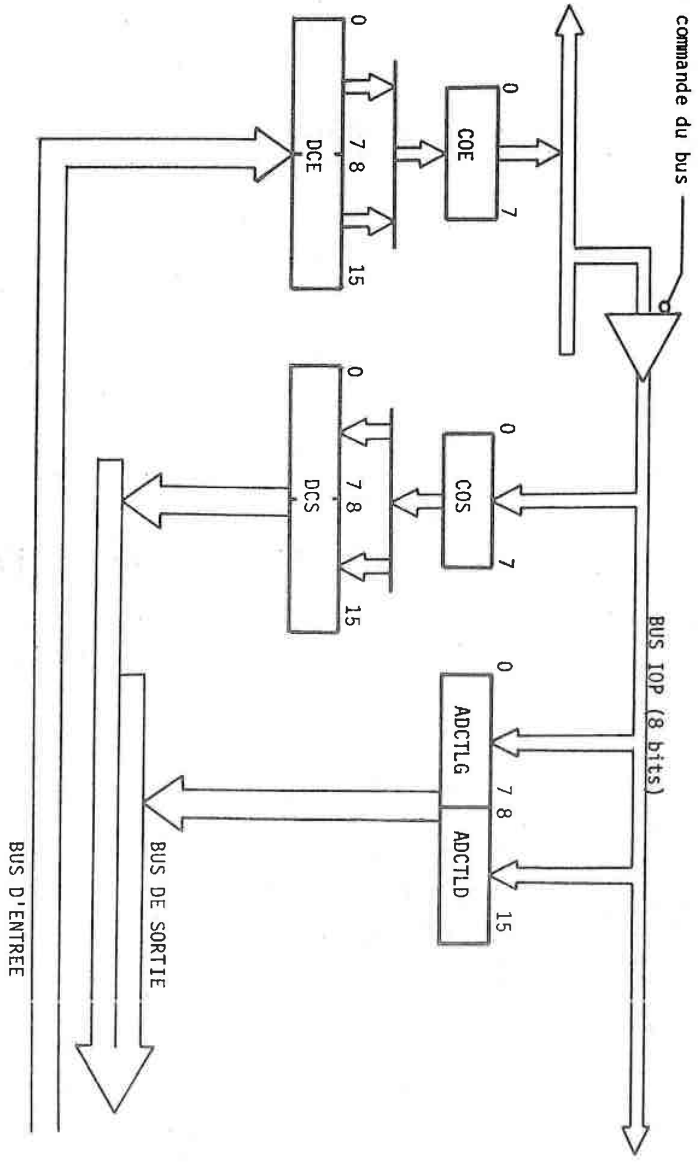


Figure C-1 : Partie opérative de l'adaptateur

1.2. LA PARTIE CONTROLE DE L'ADAPTATEUR

- Les commandes :

Nous décrivons dans ce paragraphe les commandes, voies de lecture-écriture, utilisées par l'organe microprogrammé pour contrôler l'adaptateur.

+ commandes d'initialisation de l'adaptateur :

  . ECOA : chargement du registre Adresse contrôleur poids forts (ADCTLG)
  . ECOB : chargement du registre Adresse contrôleur poids faibles (ADCTLD) et remise à zéro du registre Adresse contrôleur poids forts
  . ECOD : initialisation à zéro de la bascule MPX de gestion des multiplexeurs et initialisation à un de la bascule HO de mémorisation du signal HOLD.

+ commande de transfert en écriture : les données sortent de l'IOP par l'adaptateur.

  . ECOO : chargement de l'octet à envoyer dans le registre contrôleur Sortie (COS) et incrémentation de la bascule MPX.

+ commande de transfert en lecture : les données entrent dans l'IOP par l'adaptateur.

  . LC1E : chargement de l'octet pointé par la bascule MPX dans le registre contrôleur Entrée (COE) et incrémentation de la bascule MPX.

+ commande d'incrémentation de l'adresse :

  . ECOC : incrémentation du registre Adresse contrôleur poids faible (ADCTLD) avec propagation du report dans le registre contenant les poids forts (ADCTLG).

+ commande de l'interruption de fin de transfert :

  . ECOE : envoi d'un signal d'interruption vers le contrôleur (microprocesseur) et remise à zéro de la bascule HO.

Remarque : la bascule MPX commande le multiplexeur qui définit la partie des registres DCE et DCS utilisée. Elle est initialisée à zéro par la commande ECOD et oscille ensuite par les commandes ECOO ou LC1E.

Principe de fonctionnement :

Nous décrivons les contraintes sur le séquencement des ordres dans l'algorithme microprogrammé de commande des transferts :

+ en lecture : transfert du contrôleur vers l'adaptateur :

1 - chargement de ADCTLD (EC0B)
2 - chargement de ADCTLG (EC0A)
3 - chargement de MPX (EC0D)
4 - chargement de DCE } (LC1E)
5 - chargement de COE }
le décalage dans le temps est assuré par la logique de décodage propre à l'adaptateur, COE étant chargé avec la partie pointée par la bascule MPX.
6 - chargement de COE (LC1E)
7 - incrémentation des registres adresse (EC0C)
8 - aller en 4 jusqu'à fin du microprogramme (fin détectée grâce à l'indicateur COZ de la tête de canal)
9 - fin du microprogramme; envoi de l'interruption de fin de transfert (EC0E)

+ en écriture : transfert de l'adaptateur vers le contrôleur :

1 - chargement de ADCTLD (EC0B)
2 - chargement de ADCTLG (EC0A)
3 - chargement de MPX (EC0D)
4 - chargement de COS (EC00)
5 - chargement de DCS
la bascule MPX pointe sur la partie du registre DCS à charger; la commande EC00 provoque le chargement de DCS après un retard assuré par la logique de décodage.
6 - chargement de COS (EC00)
7 - chargement de DCS
8 - incrémentation des registres adresse (EC0C)
9 - aller en 4 jusqu'à fin du microprogramme (fin détectée grâce à l'indicateur COZ de la tête de canal)
10 - fin du microprogramme; envoi de l'interruption de fin de transfert (EC0E)

Remarques :

. l'adaptateur a une partie contrôle beaucoup plus simple que la tête de canal parce que tout transfert est toujours complet (Mémoire MITRA 125 - Contrôleur) donc piloté par la logique de la tête de canal.

. la logique de décodage des commandes de lecture-écriture possède un petit automate qui lui permet de séquencer les ordres vers le contrôleur.

- L'interface entre les commandes IOP et les requêtes vers la mémoire du contrôleur :

Le rôle de l'adaptateur est de multiplexer les informations venant de l'organe microprogrammé et de la tête de canal, dans ses registres pour les transformer en adresse et en données. De plus il transforme les commandes internes de l'IOP en provenance de l'organe microprogrammé, en requêtes externes de dialogue avec la mémoire. Pour cela il dispose d'une logique combinatoire et d'un automate qui lui permettent le décodage des commandes et le séquencement des requêtes vers la mémoire du contrôleur.

L'adaptateur reçoit un bus de commande de 7 fils :

- 5 fils indiquent le numéro de la voie de lecture-écriture utilisée
- 2 fils indiquent le type de la voie : lecture ou écriture

Il émet vers la mémoire du contrôleur les requêtes suivantes :

- R/W : requête de lecture ou d'écriture
- ME : requête mémoire (memory enable)
- HOLD: signal permettant de suspendre le microprocesseur pour obtenir un accès direct mémoire (mise du microprocesseur dans le troisième état)
- IT : interruption signalant au microprocesseur la fin du transfert

Principes du décodage et du séquencement :

- le signal R/W est issu des commandes EC00 et LC1E. Il est activé lorsque le registre DCS est prêt à être envoyé ou que le registre DOE doit être garni. Il est validé par l'impulsion ME.
- le signal HOLD est issu de la bascule H0 initialisé par EC0D, il est maintenu pendant toute la durée de l'échange et n'est remis à zéro que par la commande EC0E qui émet l'interruption de fin de transfert vers le contrôleur.

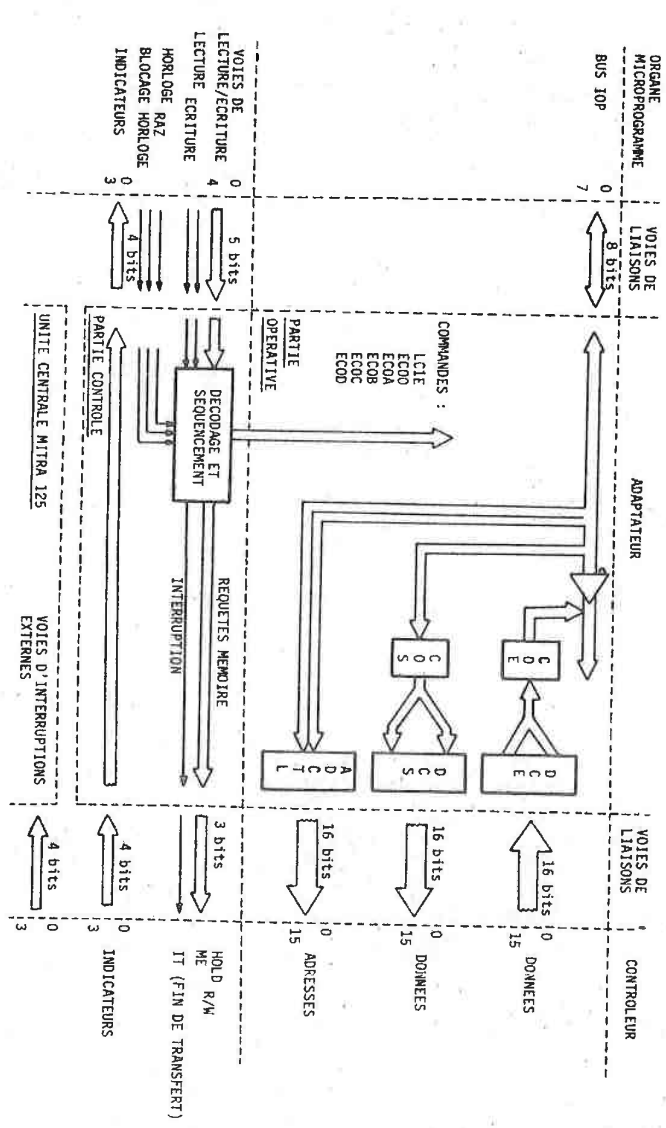


Figure C-2 : Les voies de communication entre l'IOP et le CONTROLEUR

II - LES TABLES DE COMMUNICATION

Nous décrivons dans ce paragraphe les tables de communication entre les procédures logicielles et l'IOP :

- la boîte à lettres
- le programme canal

2.1. LA BOITE A LETTRES

Elle est rangée dans l'entrée de la table des contrôleurs réservée à l'IOP 2.

Contenu de l'entrée :

Mot 00 : EVT1 : mot réservé au logiciel
Mot 02 : bits 0-7 : inutilisé
bits 8-15 : indicateur de demande de transfert
Mot 04 : EVT2 : mot réservé au logiciel
Mot 06 : BASE : valeur de la base) adresse du programme
Mot 08 : DEPL : valeur du déplacement) canal
Mot 0A : MECTL : mot d'état du contrôleur

On accède à la boîte à lettre par l'accès standard : le contenu du mot d'adresse 10 Héxadécimal pointe sur la table des contrôleurs (cf figure C-3).

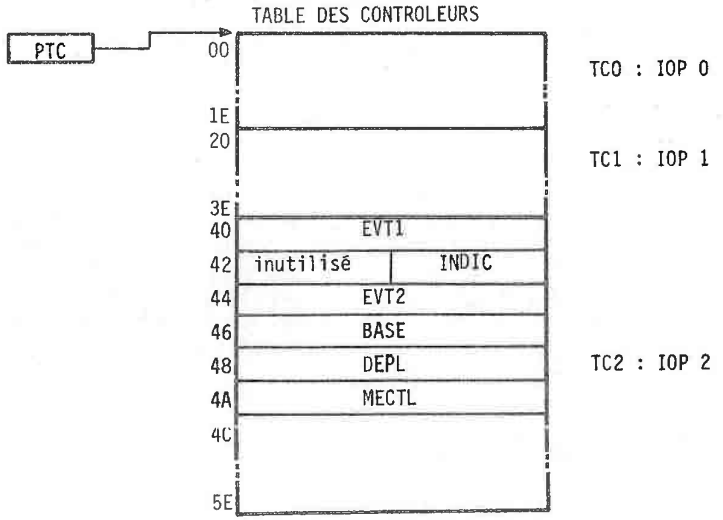


Figure C-3 : La boîte à lettres servant à la communication

2.2. LE PROGRAMME CANAL

Il est rangé dans une zone occupant 10 octets et pointée par l'adresse contenue dans les mots BASE et DEPL de la boite à lettres. Il est formé par une liste de commandes chaînées, contigües en mémoire.

Description d'une commande :

Mot 00 bits 0-7 : COP : code opération du transfert
écriture : transfert MITRA 125 → TREFLE
lecture : transfert TREFLE → MITRA 125
bits 8-15 : CHN : chaînage de commande : cet indicateur permet l'enchaînement de plusieurs commandes pour une seule demande de transfert
Mot 02-04 : ATMC: adresse de la zone tampon en mémoire centrale : mot 02 : Base
mot 04 : Déplacement
Mot 06 : ATCTL: adresse de la zone tampon en mémoire du contrôleur
Mot 08 : LONG: taille en octets du tampon à transférer

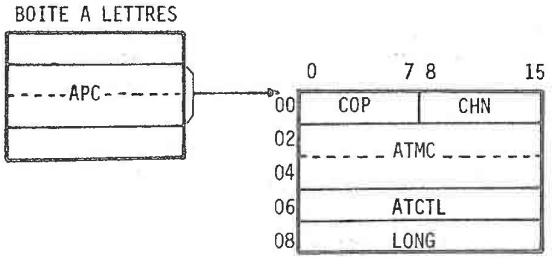


Figure C-4 : Le Programme canal

III - LE MICROPROGRAMME

3.1. LES CONVENTIONS DE LIAISONS

La procédure de transfert recopie la boite à lettres et la commande canal en cours de traitement dans les registres rapides de la mémoire locale de l'organe microprogrammé.

Nous décrivons ci-dessous les 2 groupes de registres avec les conventions utilisées par l'algorithme :

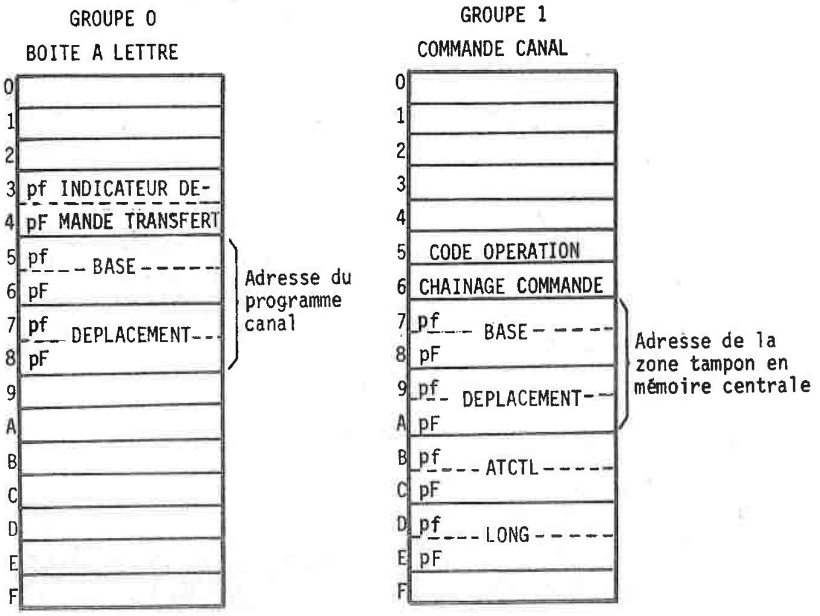


Figure C-5 : Les registres utilisés par le microprogramme de transfert

3.2. TEXTE DU MICROPROGRAMME

```
    ∇ MPCOM
[1]   TR K,15,T
[2]   ESTR ETATTK,A
[3]   TR K,4,B                          TEST ETAT TK
[4]   L ORI,A,A
[5]   TR K,P(CPSECT),A
[6]   →BA VRAI,ALFF,O(CPSECT),A
[7]  TEMPO:TR K,254,A
[8]   'TEMPORISATION'
[9]   TR A,SM,1
[10] ET2:TR K,254,A
[11]  TR K,P(ET1),ST
[12] ET1:AR INCR,A,A                    BOUCLE D'ATTENTE
[13]  →BA FAUX,ALFF,O(ET1),ST
[14]  TR SM,1,A
[15]  AR INCR,A,SM,1
[16]  TR K,P(ET2),ST
[17]  →BA FAUX,ALFF,O(ET2),ST
[18] TESTSEM:TR K,0,A
[19]  'LECTURE PTC'
[20]  ESTR A,BASED
[21]  ESTR A,BASEG
[22]  ESTR A,DEPLG
[23]  TR K,16,A                         LECTURE POINTEUR TABLE
[24]  ESTR A,DEPLD                      DES CONTROLEURS
[25]  TR K,240,A
[26]  ESTR A,COD
[27]  ESTR A,MODETK
[28]  NOP
[29]  ESTR DETK,SM,3
[30]  NOP
[31]  ESTR DETK,SM,4
[32]  'LECTURE TC'
[33]  TR K,8,A
[34]  ESTR A,BASED
[35]  ESTR SM,3,DEPLG
[36]  ESTR SM,4,DEPLD
[37]  TR K,16,T                         LECTURE TABLE DES CONTROLEURS
[38]  TR K,10,A                         (BOITE A LETTRES)
[39]  ESTR A,COD
[40]  TR K,240,A
[41]  ESTR A,MODETK
[42]  NOP
[43] BOUC:ESTR DETK,ST
[44]  TRC K,P(BOUC),A,INCRTX
[45]  →BA FAUX,COZ,O(BOUC),A
[46]  TR SM,3,A
[47]  TR K,31,T                         TEST INDICATEUR DE DEMANDE
[48]  TR K,P(TEMPO),ST                  DE TRANSFERT
[49]  →BA FAUX,A07,O(TEMPO),ST
```



```
[50] LECTCC:ESTR SM,6,BASEG
[51]  'LECTURE PC'
[52] ESTR SM,7,BASED
[53]  ESTR SM,8,DEPLG
[54]  ESTR SM,9,DEPLD
[55]  TR K,10,A
[56]  ESTR A,COD                        LECTURE COMMANDE CANAL
[57]  TR K,240,A
[58]  ESTR A,MODETK
[59]  TR K,5,T
[60] LEC:ESTR DETK,ST
[61]  TRC K,P(LEC),A,INCRTX
[62]  →BA FAUX,COZ,O(LEC),A
[63]  TR K,15,T
[64]  ESTR SM,7,BASEG
[65]  ESTR SM,8,BASED                   INITIALISATION DES REGISTRES
[66]  ESTR SM,9,DEPLG                   POUR L'ECHANGE
[67]  ESTR SM,10,DEPLD
[68]  ESTR SM,13,COG
[69]  ESTR SM,14,COD
[70]  ESTR SM,11,ADCTLG
[71]  ESTR SM,12,ADCTLD
[72]  TR SM,5,A
[73]  TR K,1,B
[74]  L ORI,A,A
[75]  TR K,P(CCECR),ST                  TEST SENS DE TRANSFERT
[76]  →BA VRAI,ALFF,O(CCECR),ST
[77]  TR K,P(CCLEC),ST
[78]  →BA VRAI,TRUE,O(CCLEC),ST
[79] CCECR:TR K,240,A
[80]  'TRANSFERT MEMOIRE→CONTROLEUR'
[81]  ESTR A,MODETK                     BOUCLE DE TRANSFERT
[82]  TR K,P(ECR),A                     MEMOIRE - CONTROLEUR
[83] ECR:ESTR DETK,ACTL
[84]  ESTR A,INCRAD
[85]  →BA FAUX,COZ,O(ECR),A
[86] CCCC:TR K,15,T
[87]  TR K,P(FINCC),ST                  TEST CHAINAGE DE COMMANDE
[88]  TR SM,6,A
[89]  →BA FAUX,A07,O(FINCC),ST
[90]  TR K,31,T
[91]  TR SM,9,B
[92]  TR K,10,A
[93]  AR PLUS,A,SM,9                    CALCUL DE L'ADRESSE DE LA
[94]  TR K,P(NODEB1),ST                 COMMANDE SUIVANTE
[95]  →BA VRAI,NOCARRY,O(NODEB1),ST
[96]  TR SM,8,A
[97]  AR INCR,A,SM,8
[98] NODEB1:TR K,P(LECTCC),A
[99]  →BA VRAI,TRUE,O(LECTCC),A
```

```
[100]FINCC:TR K,15,T
[101] 'TRANSFERT MOT ETAT CTL→TC'
[102] TR K,0,A
[103] ESTR A,BASEC
[104] ESTR A,ADCTLC
[105] ESTR A,ADCTLD
[106] TR K,8,A
[107] ESTR A,BASED
[108] TR K,10,B
[109] AR PLUS,SM,4,A
[110] TR K,P(NODEB2),ST
[111] →BA VRAI,NOCARRY,O(NODEB2),ST
[112] TR SM,3,B
[113] AR INCR,B,SM,3                    FIN PROGRAMME CANAL
[114]NODEB2:ESTR SM,3,DEPLC
[115] ESTR A,DEPLD
[116] TR K,124,A
[117] ESTR A,COD
[118] ESTR A,MODETK
[119] ESTR DECTL,ATK
[120] TR K,0,A
[121] ESTR A,ATK
[122] 'EMISSION IT FIN DE TRANSFERT'
[123] TR K,P(TEMPO),A
[124] →BA VRAI,TRUE,O(TEMPO),A
[125]CCLEC:TR K,P(CCLEC),A
[126] 'LE CONTROLEUR A T IL TERMINE SON TRAITEMENT ?1 SI OUI'
[127] INDIC[FTR]←□
[128] →BA FAUX,FTR,O(CCLEC),A
[129] 'TRANSFERT CONTROLEUR→MEMOIRE'
[130] TR K,124,A                        BOUCLE DE TRANSFERT
[131] ESTR A,MODETK                     CONTROLEUR - MEMOIRE
[132] TR K,P(LECT),A
[133]LECT:ESTR DECTL,ATK
[134] ESTR A,INCRAD
[135] →BA FAUX,COZ,O(LECT),A
[136] TR K,P(CCCC),A
[137] →BA VRAI,TRUE,O(CCCC),A
[138]CPSECT:TR K,0,A
[139] 'COUPURE SECTEUR'                 BOUCLE ATTENTE
[140] ESTR A,MODETK                     COUPURE SECTEUR
[141] TR K,P(WAIT),ST
[142]WAIT:→BA VRAI,TRUE,O(WAIT),ST
      v
```


3.3. SIMULATION DE L'EXECUTION DU MICROPROGRAMME

```
      INITIOP
INITIALISATION DE LA MEMOIRE : ENTREZ LES COUPLES ADRESSE VALEUR .
POUR ARRETER FRAPPEZ →
□:
      16 110
□:                                      PTC
      18 0
□:
      174 0
□:
      176 1
□:
      178 0                             BOITE A LETTRES
□:
      180 1000
□:
      182 500
□:
      8500 257                          PROGRAMME CANAL
□:                                        COMMANDE 1 : écriture et chaînage
      8502 2000                                        tampon mémoire : 2600
□:                                                     tampon contrôleur : 30
      8504 600                                         compte d'octets : 10
□:
      8506 30                           COMMANDE 2 : lecture
□:                                                   tampon mémoire : 3700
      8508 10                                        tampon contrôleur : 20
□:                                                   compte d'octets : 6
      8510 512
□:
      3000
      v
      8512 3000
□:
      8514 700
□:
      8516 20
□:
      8518 6
□:
      16600 511
□:
      16602 1023
□:
      16604 2047                        TAMPON MEMOIRE
□:
      16606 257
□:
      16608 32760
□:
      →
```

```
      INITMEMCTL
 INITIALISATION DE LA MEMOIRE DU CONTROLEUR .
R ENTREZ LES COUPLES ADRESSE VALEUR .POUR ARRETER FRAPPEZ →
□:
      20 255
□:
      21 15
□:
      22 1
□:
      23 1
□:
      24 55
□:
      25 77
□:
      0 255
□:
      →
```

DEBUT DE LA SIMULATION

```
      MPCOM

 TEMPORISATION

 MPCOM[17]
      →17
 LECTURE PTC
 ADRESSE MEMOIRE =16 EN LECTURE
 DME=110
 ADRESSE MEMOIRE =18 EN LECTURE
 DME=0
 LECTURE TC
 ADRESSE MEMOIRE =174 EN LECTUR"
 DME=0
 ADRESSE MEMOIRE =176 EN LECTUR"
 DME=1
 ADRESSE MEMOIRE =178 EN LECTURE
 DME=0
 ADRESSE MEMOIRE =180 EN LECTUR"
 DME=1000
 ADRESSE MEMOIRE =182 EN LECTURE
 DME=500
 LECTURE PC
 ADRESSE MEMOIRE =8500 EN LECTURE
 DME=257
 ADRESSE MEMOIRE =8502 EN LECTURE
 DME=2000
```



```
 ADRESSE MEMOIRE =8504 EN LECTURE
 DME=600
 ADRESSE MEMOIRE =8506 EN LECTURE
 DME=30
 ADRESSE MEMOIRE =8508 EN LECTURE
 DME=10
 TRANSFERT MEMOIRE→CONTROLEUR
A DRESSE MEMOIRE =16600 EN LECTURE
 DME=511
 ADRESSE CONTROLEUR =30 EN ECRITUR"
 ACTL=1
 ADRESSE CONTROLEUR =31 EN ECRITURE
A CTL=255
 ADRESSE MEMOIRE =16602 EN LECTURE
 DME=1023
 ADRESSE CONTROLEUR =32 EN ECRITURE
 ACTL=3
 ADRESSE CONTROLEUR =33 EN ECRITURE
 ACTL=255
 ADRESSE MEMOIRE =16604 EN LECTUR"
 DME=2047
 ADRESSE CONTROLEUR =34 EN ECRITURE
 ACTL=7
 ADRESSE CONTROLEUR =35 EN ECRITUR"
 ACTL=255
 ADRESSE MEMOIRE =16606 EN LECTURE
 DME=257
 ADRESSE CONTROLEUR =36 EN ECRITURE
 ACTL=1
 ADRESSE CONTROLEUR =37 EN ECRITUR"
 ACTL=1
 ADRESSE MEMOIRE =16608 EN LECTURE
 DME=32760
 ADRESSE CONTROLEUR =38 EN ECRITURE
 ACTL=127
 ADRESSE CONTROLEUR =39 EN ECRITUR"
 ACTL=248
 LECTURE PC
 ADRESSE MEMOIRE =8510 EN LECTUR"
 DME=512
 ADRESSE MEMOIRE =8512 EN LECTURE
 DME=3000
 ADRESSE MEMOIRE =8514 EN LECTURE
 DME=700
 ADRESSE MEMOIRE =8516 EN LECTURE
 DME=20
 ADRESSE MEMOIRE =8518 EN LECTUR"
 DME=6
```

```
LE CONTROLEUR A T IL TERMINE SON TRAIT-MENT ?1 SI OUI
[]:

      1
TRANSFERT CONTROLEUR→MEMOIRE
ADRESSE CONTROLEUR =20 EN LECTURE
DECTL=255
ADRESSE CONTROLEUR =21 EN LECTURE
DECTL=15
ADRESSE MEMOIRE =24700 EN ECRITURE
DMS =65295
ADRESSE CONTROLEUR =22 EN LECTURE
DECTL=1
ADRESSE CONTROLEUR =23 EN LECTURE
DECTL=1
ADRESSE MEMOIRE =24702 EN ECRITURE
DMS =257
ADRESSE CONTROLEUR =24 EN LECTURE
DECTL=55
ADRESSE CONTROLEUR =25 EN LECTURE
DECTL=77
ADRESSE MEMOIRE =24704 EN ECRITURE
DMS =14157
LE CONTROLEUR A T IL TERMINE TOUT LES TRANSFERTS ? 1 SI OUI
[]:

      1
TRANSFERT MOT ETAT CTL→'C
ADRESSE CONTROLEUR =0 EN LECTURE
DECTL=255
ADRESSE MEMOIRE =184 EN ECRITURE
DMS =65280
EMISSION IT FIN DE TRANSFERT
```


AFFICHAGE DES MEMOIRES :

```
      MEM
   16   110
   18     0
  174     0
  176     0
  178     0
  180  1000
  182   500
 8500   257
 8502  2000
 8504   600
 8506    30
 8508    10
 8510   512
 8512  3000
 8514   700
 8516    20
 8518     6
16600   511
16602  1023
16604  2047
16606   257
16608 32760
24700 65295
24702   257
24704 14157
  184 65280


    MEMCTL
20 255
21  15
22   1
23   1
24  55
25  77
 0 255
30   1
31 255
32   3
33 255
34   7
35 255
36   1
37   1
38 127
39 248
```

TEST COUPURE SECTEUR

```
T EMPORISATION

 MPCOM[17]

       MEM[3;1]
 1
       MEM[3;1]←0
       →17
 LECTURE PTC
 ADRESSE MEMOIRE =16 EN LECTURE
 DME=110
 ADRESSE MEMOIRE =18 EN LECTURE
 DME=0
 LECTURE TC
 ADRESSE MEMOIRE =174 EN LECTURE
 DME=0
 ADRESSE MEMOIRE =176 EN LECTURE
 DME=0
 ADRESSE MEMOIRE =178 EN LECTURE
 DME=0
 ADRESSE MEMOIRE =180 EN LECTURE
 DME=1000
 ADRESSE MEMOIRE =182 EN LECTURE
 DME=500
 TEMPORISATION

 MPCOM[17]
       ESREG[ETATTK-10]←4
       →1
 COUPURE SECTEUR

 MPCOM[144]
       →16
```

D- Le Disque

I - CARACTERISTIQUES ET PERFORMANCES

L'unité de disque "pack" CONTROL DATA CORPORATION 97 bb SMD a une capacité d'environ 300 Millions d'octets et son débit est de 9 677 MHz ce qui correspond à environ 1,2 Millions d'octets par seconde.

La technologie utilisée pour la logique de commande est essentiellement TTL avec un noyau ECL nécessité par le grand débit des modules de lecture/écriture. Le mode de mémorisation est la modulation de fréquence (MFM), le mode de communication étant NRZ (non retour à zéro).

La densité varie de 6038 Bpi pour une piste interne à 4038 Bpi pour une piste externe. Le nombre de disques d'une pile est de 12 dont 10 réservés aux données, les 2 autres servant de protection. Une face est réservée aux pistes de synchronisation et d'asservissement et 19 faces permettent le stockage des données à raison de 808 pistes utiles par face.

Performances :

- Accès aux données : pour ce critère, 3 grandeurs sont à considérer :
  - . le temps de positionnement : maximum : 55 ms
    minimum : 6 ms
    moyen : 30 ms
  - . le délai rotationnel : vitesse de rotation : 3600 t/mn
    maximum : 17,2 ms
    moyen : 8,33 ms
  - . le temps de commutation des têtes : entre 2 faces : 24 µs
    en lecture/écriture : 0,3 µs
    en écriture/lecture : 10 µs
- Capacité de stockage : nous précisons les capacités moyennes :
  - . nombre de bits par piste : 161 280 (environ 20 000 octets)
  - . nombre de bits par cylindre : 3 064 320
  - . nombre de bits par unité : 2 475 970 560 (environ 309 496 000 octets)
- Fiabilité : les valeurs précisées sont des probabilités moyennes :
  - . probabilité d'une erreur récupérable (en lecture) :
    $1/10^9$ bit erronné
  - . probabilité d'une erreur non récupérable (en écriture) :
    $1/10^{10}$ bit erronné
  - . probabilité d'une erreur d'accès :
    $1/10^6$ positionnement erronné

II - INTERFACE AVEC LE CONTROLEUR ET LE FORMATTEUR

La logique de contrôle d'une unité de disque magnétique est extrêmement complexe car elle opère la transition entre les données analogiques échantillonnées en lecture - écriture, ou nécessitées pour la commande du bras, et les données numériques échangées avec le milieu extérieur.

Ce paragraphe sera consacré à l'explication détaillée des signaux émis ou reçus par l'unité de disque. En effet, la gestion de ceux-ci est entièrement prise en compte par le processeur TREFLE :

- le contrôleur gère les mouvements du bras et supervise le formatteur
- le formatteur effectue les transferts d'informations.

2.1. LES SIGNAUX EMIS PAR L'UNITE DE DISQUE :

Nous décrivons les signaux, leur destinataire et l'utilisation qui en est faite.

- Marque Adresse : impulsion émise vers le formatteur lors de la rencontre d'un champ de synchronisation sur le disque. L'algorithme chargé de la lecture utilise ce signal pour activer les fonctions de réception.

- Horloge de Lecture : Horloge synchrone aux données lues sur le disque, elle permet au formatteur d'échantillonner les données qui lui sont envoyées.

- Données en lecture : Données sous la forme NRZ envoyées vers le formatteur.

- Erreur : une erreur de l'un des types suivants vient d'être détectée :

  . coupure secteur
  . défaut tête de lecture
  . défaut écriture
  . tentative de lecture/écriture pendant un déplacement de bras
  . positionnement d'une demande d'écriture pendant une lecture

  L'erreur provoque immédiatement l'inhibition de la fonction d'écriture pour éviter la destruction de données.

- On cylindre : signal indiquant au contrôleur la fin d'un mouvement de bras.

- Index : signal indiquant le début d'une piste (début du secteur 0) au contrôleur.

- Secteur : signal indiquant au contrôleur et au formatteur le début d'un secteur. En effet, la taille des secteurs est fixée par l'utilisateur grâce à des boutons poussoirs situés dans le chassis de l'unité.

- Horloge de service : Horloge servant à la génération de la modulation de fréquence en écriture, elle est disponible en permanence et assure le fonctionnement normal du formatteur.

- Erreur de positionnement : le bras est sorti du domaine adressable, un retour à la position zéro commandé par le contrôleur permet de refaire une tentative.

- Unité prête : signal indiquant au contrôleur que l'unité est prête à recevoir les demandes de lecture/écriture.

- Unité selectée : signal indiquant au contrôleur que l'adresse envoyée a été reconnue.

- Fin de positionnement : signal résultant du "OU" entre le signal On cylindre et le signal erreur de positionnement.

- Protection écriture : signal utilisé par la maintenance et indiquant que l'écriture est interdite.

2.2. LES SIGNAUX RECUS PAR L'UNITE DE DISQUE :

Le nombre de signaux reçus par l'unité de disque est beaucoup plus grand que celui des signaux émis car les adresses d'unité, de cylindre et de tête sont reçues en parallèle. Afin de minimiser le nombre de broches du connecteur, un partage temporel de certaines voies par des signaux différents a été mis en oeuvre. Pour recevoir les adresses de cylindre (10 bits), de tête de lecture/écriture (5 bits) et divers signaux de contrôle, un bus de 10 lignes est partagé dans le temps.

Le multiplexage est assuré par 3 fils de fonction nommés TAG1, TAG2 et TAG3 qui spécifient la nature des signaux véhiculés par le bus et qui en permettent l'échantillonage :

- l'adresse cylindre 10 bits (bits 0 à 9) est échantillonnée par TAG1
- l'adresse tête de L/E 5 bits (bits 0 à 4) est échantillonnée par TAG2
- les signaux de contrôle 9 bits (bits 0 à 8) sont échantillonnés par TAG3

Description de ces signaux :

- bit 0 : porte d'écriture : ce signal rend disponible la fonction d'écriture

- bit 1 : porte de lecture : ce signal rend disponible la fonction de lecture et arrête le mécanisme de synchronisation chargé de reconnaître le début du secteur

- bit 2 : déplacement positif du bras
- bit 3 : déplacement négatif du bras

} ces deux signaux gèrent le mécanisme d'asservissement du bras

- bit 4 : remise à zéro de la bascule erreur

- bit 5 : prise en compte de la Marque Adresse : ce signal permet l'accès au secteur

- bit 6 : retour à zéro : ce signal remet le bras sur la piste zéro

- bit 7 : échantillonnage en avance
- bit 8 : échantillonnage en retard

} ces commandes permettent la relecture d'un signal déphasé

Les autres signaux de contrôle sont véhiculés par des fils séparés :

- Détection de la coupure secteur ou de la rupture de connexion : ce signal permet de protéger l'unité des ruptures de câble. Si une rupture est détectée alors l'unité est mise dans un état inopératif.

- Sélection d'unité : ce signal valide l'adresse de l'unité véhiculée par les quatre lignes numéro d'unité.

- Adresse unité (4lignes) : ces quatre lignes portent le numéro de l'unité demandée par le contrôleur.

- Données en écriture : ce fil véhicule les données destinées à être écrites; elles sont sous la forme NRZ et proviennent du formatteur.

- Horloge d'écriture : cette horloge est obtenue en rebouclant l'horloge de service sur cette entrée.

UNIVERSITE NANCY BIBLIOTHEQUE

E- Le Formatteur

INTRODUCTION

Interface entre le disque et la machine TREFLE, le FORMATTEUR [Tus 78] ventile les informations lues vers leurs destinataires. Nous présentons dans cette annexe la structure matérielle chargée d'interprêter les algorithmes de gestion de la structure de données implantée sur le disque. Nous décrirons successivement : la partie opérative, la partie contrôle et le langage de microprogrammation; un exemple de micro-programme complètera cette présentation.

I - LA PARTIE OPERATIVE DU FORMATTEUR

La partie opérative du FORMATTEUR se compose de six ensembles fonctionnels reliés entre eux par un réseau de bus (cf figure E-1) :

- l'interface avec le disque : sérialisation et parallélisation des données
- la génération du code de vérification
- les compteurs
- la détection des erreurs
- les voies de communication avec les autres modules de TREFLE
- la gestion des horloges

1.1. SERIALISATION ET PARALLELISATION

Un registre à décalage et un registre tampon de 8 bits chacun, assurent cette fonction (cf figure E-2).

+ En lecture : le registre à décalage reçoit les bits en provenance du disque et permet leur parallélisation. L'octet reconstitué est mis dans le registre tampon qui permet son transfert vers les destinataires.

+ En écriture : il reçoit le contenu du registre tampon et les décalages successifs lui permettent de sérialiser les octets en bits.

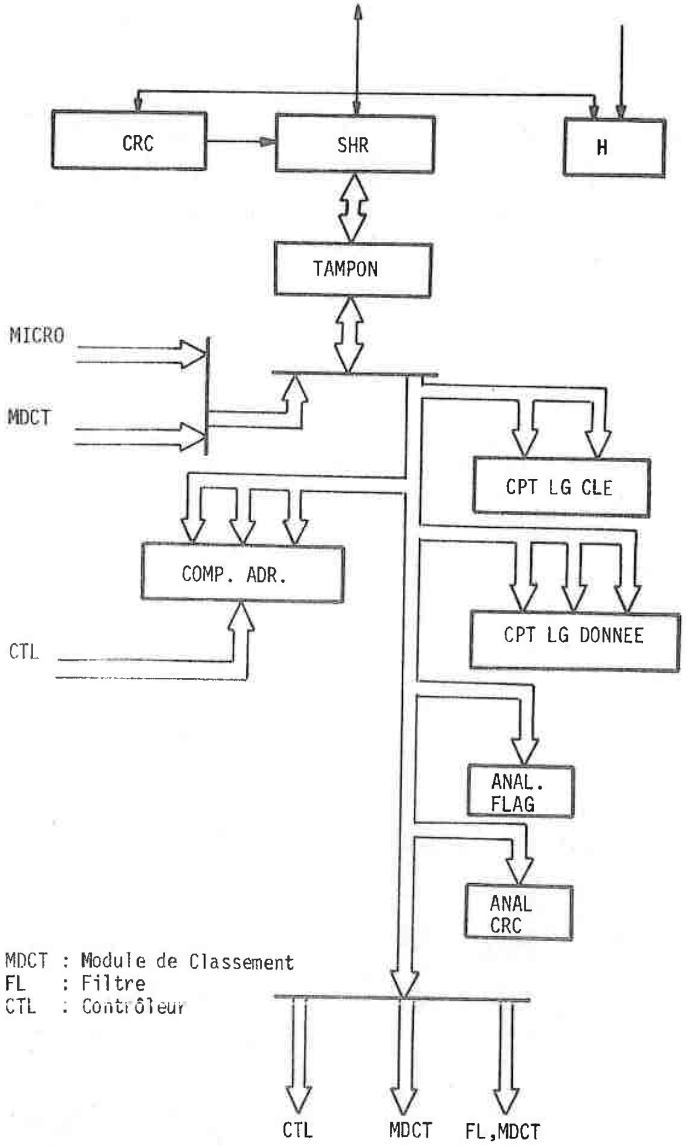
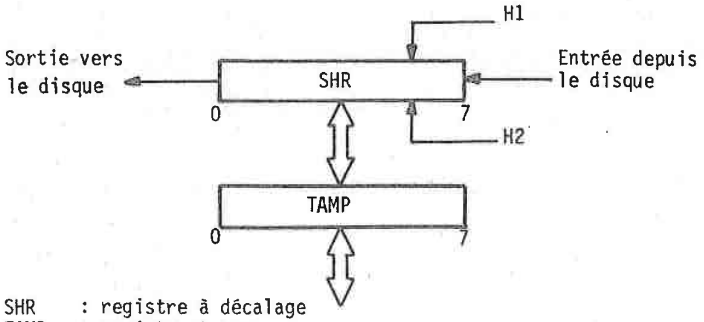


Figure E-1 : La partie opérative du FORMATTEUR



SHR : registre à décalage
TAMP : registre tampon
H1, H2 : horloges

Figure E-2 : Sérialisation et Parallélisation

Remarque :

- l'horloge H1 permet le chargement du registre SHR
- l'horloge H2 permet son décalage
- cet ensemble fonctionne en pipe line :
  - . pendant la parallélisation d'un octet, l'octet précédent est disponible dans le registre TAMP
  - . il faut charger l'octet dans le registre TAMP pendant la sérialisation de l'octet précédent.

1.2. GENERATION DU CODE DE VERIFICATION

Principe :

Pour découvrir les éventuelles erreurs de lecture/écriture sur le disque, nous utilisons la technique du CRC (Cyclic Redundancy Code) [Hel 74] [Swa 75] .

Elle consiste à calculer le reste de la division d'une chaîne de bits par un polynôme, le reste étant ensuite placé à la suite de la chaîne.

Cette méthode a l'avantage de permettre une vérification aisée de la conformité de la chaîne de caractères lue.

Remarque : Le polynôme choisi est : $X^{16} + X^{12} + X^5 + 1$

1.3. LES COMPTEURS

Le rôle des compteurs est de permettre la gestion des champs de longueur variable en décomptant le nombre de caractères lus ou écrits. Au nombre de trois, ils sont spécialisés dans une fonction de comptage particulière :

- le compteur de longueur du champ Clé : formé de deux registres de 8 bits

- le compteur de longueur du champ Donnée : formé de trois registres de 8 bits

- le compteur de boucle : formé d'un registre de 8 bits, il offre au microprogrammeur la possibilité d'écrire des boucles avec comptage

Le passage à zéro de ces compteurs peut être testé par la microinstruction.

1.4. LA DETECTION DES ERREURS

Trois ensembles de comparateurs forment ce module :

- les comparateurs d'adresse courante
- l'analyseur de Flag
- l'analyseur de CRC

+ les comparateurs d'adresse courante vérifient l'égalité entre les octets du champ Adresse du secteur lu sur le disque et les octets envoyés par le CONTROLEUR. La vérification est explicitement demandée dans la microinstruction.

+ le comparateur analysant le Flag de début de secteur vérifie si le secteur ou la piste sont valides. Dans le cas contraire, il produit un déroutement dans le microprogramme.

+ le comparateur analysant le CRC permet la vérification de l'égalité à zéro des octets transitant sur le bus. Cette comparaison permet de vérifier que le formattage du disque est correct et que le CRC est conforme.

1.5. LES VOIES DE COMMUNICATION AVEC LES AUTRES MODULES DE TREFLE

Les voies de communication entre le FORMATTEUR et les autres modules de TREFLE sont unidirectionnelles et commandées depuis la microinstruction.

- la voie d'entrée :

  le MODULE DE CLASSEMENT peut envoyer des caractères au FORMATTEUR par un accès au registre TAMPON décrit en 1.1. Cette voie peut aussi être utilisée pour envoyer une valeur immédiate depuis la microinstruction vers le disque.

- les voies de sortie :

  Au nombre de trois, elles permettent l'envoi des informations suivantes :

  . le FORMATTEUR peut envoyer les octets de données vers le FILTRE et le MODULE DE CLASSEMENT.

  . il peut aussi envoyer les octets de clé vers le MODULE DE CLASSEMENT.

  . le vecteur des secteurs invalides sera envoyé vers le CONTROLEUR.

1.6. LA GESTION DES HORLOGES

La logique du disque utilise deux horloges :

- une horloge de lecture, décodée en même temps que les signaux d'informations

- une horloge de service utilisée pour l'écriture et le cadencement de la logique.

Le FORMATTEUR utilise ces deux horloges en fonction de son traitement :

- en lecture, il utilise l'horloge de lecture pour cadencer la gestion du registre à décalage et la génération du CRC et génère l'horloge principale.

- en écriture, il utilise l'horloge de service pour assurer son propre fonctionnement et pour valider les informations envoyées au disque.

Le choix de l'horloge est effectuée par la microinstruction.

II - LA PARTIE CONTROLE DU FORMATTEUR

La partie contrôle du FORMATTEUR se compose de quatre parties fonctionnelles qui sont : (cf figure E-3)

- l'unité de contrôle
- la mémoire de microprogramme
- la logique de bouclage
- la logique de déroutement

2.1. L'UNITE DE CONTROLE

Elle a pour rôle de gérer l'enchaînement des microinstructions mémorisées en mémoire de commande. Le contrôle est assuré par deux unités AM 2909 de la série AMD 2900 de microprocesseurs en tranche de 4 bits.

Le choix du type de séquencement est fonction d'un champ de la microinstruction.

- Les quatre sources d'adresses sont les suivantes :

  1 - l'entrée directe d'une valeur immédiate codée dans la microinstruction qui permet les sauts inconditionnels.

  2 - l'entrée à travers un registre : nous utilisons cette possibilité pour nous brancher aux débuts des algorithmes des ordres.

  3 - la pile interne qui autorise l'appel de sous-programmes en permettant la sauvegarde des adresses de retour.

  4 - le contenu du compteur ordinal de microprogramme qui permet la progression séqentielle.

- Avant d'adresser la mémoire de commande, le bus adresse traverse une logique qui permet de réaliser simplement des déroutements : le signal ZERO permet de forcer à zéro la valeur du bus adresse et déroute ainsi le microprogramme à l'adresse zéro de la mémoire de commande.

- Le compteur ordinal de microprogramme reçoit la valeur de la retenue entrante CIN et émet la retenue sortante COUT. Ce dispositif permet la connexion en cascade de plusieurs unités de commande. Il permet aussi le bouclage sur une instruction (CIN=0) et il suffit alors de commander le passage de CIN à 1 en fin de bouclage.

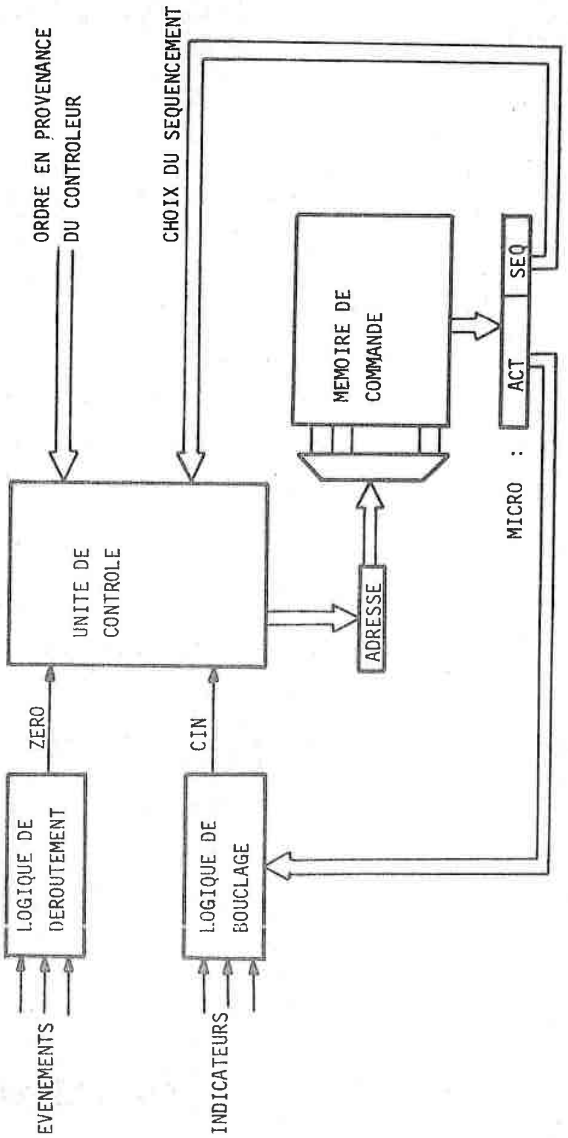


Figure E-3 : La partie contrôle du FORMATTEUR

2.2. LA MEMOIRE DE MICROPROGRAMME

Elle est constituée d'une mémoire vive de 256 x 37 bits ce qui permet le stockage de 256 microinstructions de 37 bits chacune. L'accès en écriture permet la modification des microinstructions, mais seul l'accès en lecture est normalement utilisé.

2.3. LA LOGIQUE DE BOUCLAGE

Cette logique, commandée par la microinstruction permet d'effectuer les différents bouclages pour les comptages nécessaires aux algorithmes. Les indicateurs suivants sont pris en compte par cette logique pour le passage de '0' à '1' du signal CIN :

- le passage à zéro du compteur longueur de clé
- le passage à zéro du compteur longueur de donnée
- le passage à zéro du compteur de boucles
- l'attente et la reconnaissance du Flag (passage de la valeur '1111' sur les bits de poids fort)
- l'attente d'un ordre du CONTROLEUR.

La microinstruction précise l'indicateur dont le passage à 1 permettra la sortie de la boucle.

2.4. LA LOGIQUE DE DEROUTEMENT

Les événements suivants génèrent un déroutement à l'adresse zéro grâce à cette logique particulière :

- la remise à zéro (réinitialisation) par le CONTROLEUR (RESET)
- la détection d'une erreur CRC
- la détection d'une erreur d'adresse
- la détection de l'invalidité de la piste
- la détection de l'invalidité du secteur
- la détection d'une valeur nouvelle lors du formattage.

Parallèlement à ce déroutement, une interruption est envoyée au CONTROLEUR.

III - LE LANGAGE DE MICROPROGRAMMATION

La mémoire de commande peut contenir jusqu'à 256 microinstructions de 37 bits. Ces microinstructions longues sont découpées en 18 champs selon le format suivant :

n° du champ	longueur	nom	action
1	1	W	écriture sur le disque
2	1	R	lecture sur le disque
3	1	SEL1	choix de l'horloge
4	1	ACT1	activation du générateur de CRC
5	1	SCC	décrémentation du compteur de longueur du champ clé
6	1	SCD	décrémentation du compteur de longueur du champ donnée
7	1	SEL3	choix de l'entrée dans le registre Tampon
8	2	SELD3	choix du bus de sortie
9	4	SELDEC	commande principale de la gestion du bus interne : - comparaison des adresses - vérification du CRC et du Flag - chargement des compteurs
10	1	INP	registre Tampon dans SHR
11	8	DATA	valeur immédiate
12	2	SOS1	choix du type de séquencement
13	1	FE	accès à la pile de sauvegarde
14	1	PUP	empilement et dépilement
15	8	JUMP	adresse de branchement ou valeur immédiate
16	1	LD	chargement du compteur de boucle avec la valeur du champ JUMP
17	1	OVF	test débordement du compteur de boucle
18	1	SEL3A	soustraction du polynôme dans le générateur de CRC

Le langage de microinstructions que nous avons développé introduit un mnémonique pour les valeurs significatives des champs de la microinstruction. Une microinstruction sera décrite par la liste des mnémoniques qui codent ses actions.

Exemple :

la microinstruction : W SV AG CC T BI AS CRC correspond à :

- une écriture avec l'horloge de service (W et SV)
- le générateur de CRC est activé (AG et CRC)
- les données écrites viennent du MODULE DE CLASSEMENT (T) et sont comptées par le compteur de clés (CC)
- le bus interne est bloqué (BI)
- le séquencement est un bouclage (AS et CC)

IV - EXEMPLE DE MICROPROGRAMME

4.1. LE MICROPROGRAMME

MICROPROG :

```
[1]    →EX D,BI,S 11,AR,NO
[2]    →EX W,D,BI,S 11,I,AS,FE,J 32,LD,NO
[3]    →LX W,D,BI,S 11,I,FE,OV
[4]    →EX W,AG,D,BI,S 11,I,DA 240,AJ,FE,NO,CR
[5]    →EX W,AG,T,BI,S 11,I,AS,FE,NO,CR
[6]    →EX W,AG,T,BI,S 11,I,AS,FE,NO,CR
[7]    →EX W,AG,T,BI,S 11,I,AS,FE,NO,CR
[8]    →EX W,AG,D,BI,S 11,AS,FE,NO
[9]    →EX W,AG,D,BI,S 11,AS,FE,NO
[10]   →EX D,BI,S 11,AP,PL,NO
```


4.2. SIMULATION DE L'EXECUTION

```
      INITFOR
SIMULATION OU ASSEMBLAGE ? 0 OU 1
□:
      0
CODE DE L'ORDRE ?
□:
      1
      MICROPROG
PC=0
PC=1
ECRITURE SUR LE DISQUE
0
BOUCLE SUR 2
BOUCLE
D'ECRITURE DE224OCTETS DE
SYNCHRO
PC=3
ECRITURE SUR LE DISQUE
240
RECEPTION DEPUIS LE TAMPON
□:
      11
PC=4
ECRITURE SUR LE DISQUE
11
RECEPTION DEPUIS LE TAMPON
□:
      22
PC=5
ECRITURE SUR LE DISQUE
22
RECEPTION DEPUIS LE TAMPON
□:
      33
PC=6
ECRITURE SUR LE DISQUE
33
PC=7
ECRITURE SUR LE DISQUE
OCTET DE CRC
PC=8
ECRITURE SUR LE DISQUE
OCTET DE CRC
PC=9
```

UNIVERSITE
NANCY
BIBLIOTHEQUE

F- Le Modèle

I - LISTE DU MODELE

```
!QNAPO
MODEL?FSMOD3
EXECUTION F4LIB VERSION V10-09 14*32*29*
*** QUEUING NETWORK ANALYSIS PACKAGE (22-3-79) ***
```

```
 1  /DECLAR/QUEUE DISQUE,F1,F2,F3,T,V1,V2,S1;
 2          REAL  TS,PA1,PA2,PA3,PNB,A,B,BRAS,BI,BS,TV,TC;
 3          INTEGER G1,CP,TM1,TM2,TM3,TR1;
 4          POINTER QUEUE PQ;
 5  /STATION/NAME=DISQUE;
 6           TYPE=GEN;
 7           TRANSIT=V2;
 8           SERVICE=BEGIN
 9                     IF G1=0 THEN BEGIN
10                        G1:=1;
11                        IF (CUSTNB(F1)>TR1) THEN TR1:=CUSTNB(F1);
12                        TS:=UNIFORM(A,B);
13                        EXP(BRAS);
14                        SAMPLE;
15                                 END
16                             ELSE BEGIN
17                        TS:=TS-1;
18                        CST(0.8);
19                        IF TS<0 THEN G1:=0;
20                                 END;
21                   END;
22  /STATION/NAME=V2;
23          TYPE=DELAY;
24          TRANSIT=PQ;
25          SERVICE=BEGIN
26                    IF S1.VALUE=0 THEN BEGIN
27                          CP:=CP+1;
28                          PQ:=OUT;
29                                        END
30                                   ELSE PQ:=F1;
31                   END;
32  /STATION/NAME=F1;
33          TRANSIT=OUT,PA1,F2;
34          SERVICE=BEGIN
35                    IF (CUSTNB(QUEUE)>TM1) THEN TM1:=CUSTNB(QUEUE);
36                    UNIFORM(BI,BS);
37                  END;
```

```
38  /STATION/NAME=F2;
39          TRANSIT=OUT,PA2,F3;
40          SERVICE=BEGIN
41                  IF (CUSTNB(QUEUE)>TM2) THEN TM2:=CUSTNB(QUEUE);
42                  UNIFORM(BI,BS);
43           END;
44  /STATION/NAME=F3;
45          TRANSIT=OUT,PA3,T;
46          SERVICE=BEGIN
47                  IF (CUSTNB(QUEUE)>TM3) THEN TM3:=CUSTNB(QUEUE);
48                  UNIFORM(BI,BS);
49                END;
50  /STATION/NAME=T;
51          TYPE=DELAY;
52          TRANSIT=OUT,PNB,V1;
53          SERVICE=CST(TC);
54  /STATION/NAME=V1;
55          TYPE=DELAY;
56          TRANSIT=OUT;
57          SERVICE=BEGIN
58                  P(S1);
59                  CST(TV);
60                  V(S1);
61        END;
62  /STATION/NAME=S1;
63          TYPE=SEMAPHORE,MULTIPLE(1);
64  /METHOD/TMAX=20000;
65          PERIOD=20000;
66          ACCURACY=F1,F2,F3;
67          CORREL=F1(5);
68  /EXEC/BEGIN
69           A:=1600;
70           B:=2000;
71           BRAS:=150;
72           G1:=0;
73           BI:=0.15;
74           BS:=1.35;
75           TC:=0.8;
76           TV:=5000;
77           PA1:=0.8;
78           PA2:=0.9;
79           PA3:=0.95;
80           PNB:=0.999;
81           CP:=0;
82           TM1:=0;
83           TM2:=0;
84           TM3:=0;
85           TR1:=0;
86           SIMUL;
87           PRINT("CP=",CP);
88           PRINT("TR1=",TR1);
89           PRINT("TM1=",TM1);
90           PRINT("TM2=",TM2);
91           PRINT("TM3=",TM3);
92        END;
```


II - LES RESULTATS

2.1. LES MESURES SUR LES FILES

```
*** SIMULATION BEGINS ... ***
***TIME =    19999.96 , NB SAMPLES =     14 , CONF. LEVEL = 0.95
*****************************************************************************
*                                                                           *
*QUEUE DISQ  SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .889     1.000     1.000     .889     1.125      .000     22498*
*                                                                           *
*QUEUE F1    SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .747      .840     1.873    1.665     1.125      .000     22494*
*      +/-       .004      .050      .148     .084      .067      .000         *
*                                                                           *
*QUEUE F2    SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .756      .172      .183     .804      .227      .000      4542*
*      +/-       .010      .009      .010     .014      .012      .000         *
*                                                                           *
*QUEUE F3    SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .744      .016      .016     .747      .022      .000       433*
*      +/-       .025      .002      .002     .029      .003      .000         *
*                                                                           *
*QUEUE T     SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .800      .001      .001     .800      .001      .000        18*
*                                                                           *
*QUEUE V2    SERVICE  BUSY PCT  CUST NB  RESPONS  THRUPUT  BLOCKED  SERV NB*
*GLOBAL          .000      .000      .000     .000     1.125      .000     22498*
*                                                                           *
*****************************************************************************
*** END OF SIMULATION : TIME =    19999.96
```

2.2. LES FONCTIONS D'AUTOCORRELATION SUR LES FILES

AUTOCORRELATION FUNCTIONS ON QUEUE MEASURES

AUTOCORRELATION ON QUEUE F1							
ORDER	BLOC SIZE	SERV TIME	BUSY TIME	QUEUE LENGTH	RES TIME	NB SERVED	BLOCKED TIME
WARNING : MAXIMUM ORDER OF AUTOCORRELATION FUNCTION LIMITED TO (NUMBER OF SAMPLES)/4 = 3							
1	-.125	-.029	-.029	-.043	-.043	-.034	.000
2	-.047	-.049	-.049	-.037	-.037	-.054	.000
3	-.019	.045	.045	.163	.162	.030	.000

END OF AUTOCORRELATION COMPUTATIONS

2.3. LES RESULTATS PARTICULIERS

```
CP=        0.
TR1=       7.
TM1=       9.
TM2=       3.
TM3=       1.

  93   /END/
*STOP* 0
```

CP : nombre de clients perdus

TR1 : nombre de clients restant à traiter après une bouffée

TM1 : taille maximale file F1

TM2 : taille maximale file F2

TM3 : taille maximale file F3

TABLE DES MATIERES

INTRODUCTION

CHAPITRE 1 : LA MACHINE TREFLE

Introduction I - 1

I - Le langage de requêtes I - 2

1.1. les phrases de description I - 4
1.2. Les phrases d'interrogation I - 6
1.3. Les phrases de notation I - 7

II - L'architecture du processeur TREFLE I - 10

2.1. Les principes fonctionnels I - 10
2.2. Le processeur TREFLE I - 18
2.2.1. Les filtres I - 21
2.2.1.1. L'interprétation de la grammaire I - 22
2.2.1.2. L'évaluation de la fonction logique I - 25
2.2.1.3. La résolution des fautes d'orthographe I - 27
2.2.1.4. Les caractéristiques de la réalisation I - 27
2.2.2. Le module de calcul I - 28
2.2.3. Le module de classement I - 29
2.2.4. Le disque I - 31
2.2.5. Le formatteur I - 31
2.2.6. Le contrôleur I - 31
2.2.7. Le mini-ordinateur hôte I - 32

Conclusion : les applications I - 33

CHAPITRE 2 : LES COMMUNICATIONS

Introduction II - 1

I - La liaison entre TREFLE et le minicalculateur hôte II - 2

1.1. La structure du mécanisme de communication standard entre le MITRA 125 et les unités périphériques II - 2
1.1.1. Les mécanismes de communication II - 3
1.1.1.1. Présentation du calculateur MITRA 125 II - 3

1.1.1.2. Structure des voies de communication II - 4
1.1.1.3. Mécanisme de communication implanté dans l'unité centrale II - 6
1.1.2. La structure du processeur d'entrée/sortie (IOP) II - 8
1.2. La communication entre TREFLE et le MITRA 125 II - 12
1.2.1. Les principes II - 13
1.2.2. La connexion matérielle II - 14
1.2.3. Le protocole logiciel II - 15
1.2.3.1. Procédure d'initialisation du transfert II - 17
1.2.3.2. Procédure de transfert II - 18
1.2.3.3. Procédures annexes II - 20
1.2.3.4. Coopération des processus II - 22

II - La liaison entre TREFLE et le disque II - 25

2.1. Les principes II - 25
2.2. La structure de données implantée sur le disque II - 28
2.2.1. Le champ d'identification II - 29
2.2.2. Le champ de validité de piste II - 30
2.2.3. Le champ de clés II - 30
2.2.4. Le champ de données II - 31
2.3. Les algorithmes des fonctions opérant sur la structure de données II - 34
2.3.1. Algorithme de la fonction d'écriture de l'entête d'un secteur II - 34
2.3.2. Algorithme de la fonction de formattage II - 34
2.3.3. Algorithme de la fonction d'écriture du vecteur des secteurs invalides II - 34
2.3.4. Algorithme de la fonction d'écriture des champs-clés et données II - 36
2.3.5. Algorithme de la fonction de lecture du vecteur des secteurs invalides II - 37
2.3.6. Algorithme de la fonction de lecture du champ-clé II - 38
2.3.7. Algorithme de la fonction de lecture des clés et des données II - 39
2.3.8. Algorithme de la fonction de vérification du formattage II - 39

2.4. La structure du formatteur II - 40

III - Le système de production de logiciel II - 41

3.1. Les principes II - 41
3.1.1. Les principales contraintes II - 42
3.1.2. Les choix II - 42
3.1.3. La méthode II - 43
3.2. Les réalisations II - 44

Conclusion II - 45

CHAPITRE 3 : L'EVALUATION

Introduction III - 1

I - L'évaluation des performances des systèmes informatiques III - 2

1.1. Etude des méthodologies utilisées pour l'évaluation par mesures d'un système informatique III - 3
1.1.1. Anatomie d'un évaluateur III - 3
1.1.2. Les méthodes de prises de mesures III - 5
1.1.2.1. Mesures par un évaluateur implanté dans une tâche utilisateur III - 6
1.1.2.2. Mesures par un évaluateur implanté dans le système d'exploitation III - 7
1.1.2.3. Mesures par un évaluateur implanté dans le processeur virtuel III - 9
1.1.2.4. Mesures par un logimètre III - 11
1.2. La modélisation, outil d'évaluation III - 12
1.2.1. Les méthodes de résolution des modèles composés de réseaux de files d'attente III - 13
1.2.2. Les langages de simulation III - 15

II - L'évaluation des performances de la machine TREFLE III - 17

2.1. La modélisation de la machine III - 18
2.1.1. L'étude du fonctionnement de TREFLE III - 18
2.1.2. La définition du modèle III - 22
2.1.3. La résolution du modèle III - 25

2.2. L'évaluation par mesures du prototype III - 31
2.2.1. Les contraintes III - 31
2.2.2. Les principes III - 32
2.2.3. Les réalisations III - 35
2.2.3.1. Evaluation des paramètres internes au formatteur III - 36
2.2.3.2. Evaluation des paramètres internes aux filtres III - 38
2.2.3.3. Remarques générales sur la méthode de prise de mesure III - 41
2.3. Les résultats des premières simulations III - 43
2.3.1. Etude des variations de la taille des files d'attente III - 43
2.3.1.1. Les conditions de l'expérimentation III - 43
2.3.1.2. Les résultats obtenus III - 45
2.3.1.3. L'interprétation des résultats III - 47
2.3.2. Etude des variations du taux d'activité des stations FILTRE III - 49
2.3.2.1. Les conditions de l'expérimentation III - 50
2.3.2.2. Les résultats obtenus III - 50
2.3.2.3. L'interprétation des résultats III - 58

Conclusion III - 59

CONCLUSION

BIBLIOGRAPHIE

ANNEXES

A - Les entrées/sorties
I - Les connexions A - 1
II - Les tables d'entrée/sortie A - 2
2.1. La table des contrôleurs A - 3
2.2. La table des unités A - 5
2.3. Le programme canal A - 6

B - L'IOP
I - La tête de canal B - 1
1.1. Les voies de liaison avec la mémoire B - 1

1.2. Les interfaces B - 1
1.3. La gestion du séquencement et du compteur d'octets B - 3
II - L'organe microprogrammé B - 7
2.1. La partie opérative B - 7
2.2. La partie contrôle B - 10
2.3. Le langage de micro-instructions B - 10

C - La connexion
I - L'adaptateur C - 1
1.1. La partie opérative de l'adaptateur C - 1
1.2. La partie contrôle de l'adaptateur C - 3
II - Les tables de communication C - 7
2.1. La boîte à lettres C - 7
2.2. Le programme canal C - 8
III - Le microprogramme C - 9
3.1. Les conventions de liaison C - 9
3.2. Texte du microprogramme C - 10
3.3. Simulation de l'exécution du microprogramme C - 13

D - Le disque
I - Caractéristiques et performances D - 1
II - Interface avec le contrôleur et le formatteur D - 2
2.1. Les signaux émis par l'unité de disque D - 2
2.2. Les signaux reçus par l'unité de disque D - 3

E - Le formatteur
I - La partie opérative E - 1
1.1. Sérialisation et parallélisation E - 1
1.2. Génération du code de vérification E - 3
1.3. Les compteurs E - 4
1.4. La détection des erreurs E - 4
1.5. Les voies de communication avec les autres modules de TREFLE E - 4
1.6. La gestion des horloges E - 5
II - La partie contrôle du formatteur E - 6
2.1. L'unité de contrôle E - 6
2.2. La mémoire de microprogrammation E - 8
2.3. La logique de bouclage E - 8

2.4. La logique de déroutement E - 8
III - Le langage de microprogrammation E - 9
IV - Exemple de microprogramme E - 10
4.1. Le microprogramme E - 10
4.2. Simulation de l'exécution E - 11

F - Le modèle
I - Liste du modèle F - 1
II - Les résultats F - 3
2.1. Les mesures sur les files F - 3
2.2. Les fonctions d'autocorrélation sur les files F - 4
2.3. Les résultats particuliers F - 4

UNIVERSITE NANCY I

NOM DE L'ETUDIANT : Monsieur SCHWAAB François

NATURE DE LA THESE : Thèse de Spécialité : Informatique

VU, APPROUVE

et PERMIS D'IMPRIMER

NANCY LE 28 MAI 1979 481

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

UNIVERSITE NANCY I
LE PRESIDENT
M. BOULANGE