# THÈSE

présentée à

L'INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

pour obtenir le titre de

DOCTEUR INGÉNIEUR

*Spécialité Automatique*

*par*

Ziad OLAIWAN

Ingénieur de l'Ecole Supérieure d'Ingénieurs de Beyrouth (E.S.I.B.)

DEA Génie Informatique ENSIMAG - INPG

SYSTÈME MULTIMICROPROCESSEUR POUR LA COMMANDE AUTOMATIQUE

*Soutenue le 24 Septembre 1979 devant la commission d'examen*

*J U R Y*

Monsieur R. PERRET — Président

Messieurs F. ANCEAU
Z. BINDER
P. DESCHIZEAUX
D.G. LAINIOTIS
J.F. LE MAITRE
— Examinateurs

# INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Président : M. Philippe TRAYNARD

Vice-Présidents : M. Georges LESPINARD
M. René PAUTHENET

Année Universitaire
1978-1979

## PROFESSEURS TITULAIRES

| | | |
|---|---|---|
| MM | BENOIT Jean | Electronique - Automatique |
| | BESSON Jean | Chimie Minérale |
| | BLOCH Daniel | Physique du Solide - Cristallographie |
| | BONNETAIN Lucien | Génie Chimique |
| | BONNIER Etienne | Métallurgie |
| | *BOUDOURIS Georges | Electronique - Automatique |
| | BRISSONNEAU Pierre | Physique du Solide - Cristallographie |
| | BUYLE-BODIN Maurice | Electronique - Automatique |
| | COUMES André | Electronique - Automatique |
| | DURAND Francis | Métallurgie |
| | FELICI Noël | Electronique - Automatique |
| | FOULARD Claude | Electronique - Automatique |
| | LANCIA Roland | Electronique - Automatique |
| | LONGEQUEUE Jean-Pierre | Physique Nucléaire Corpusculaire |
| | LESPINARD Georges | Mécanique |
| | MOREAU René | Mécanique |
| | PARIAUD Jean-Charles | Chimie-Physique |
| | PAUTHENET René | Electronique - Automatique |
| | PERRET René | Electronique - Automatique |
| | POLOUJADOFF Michel | Electronique - Automatique |
| | ROBERT André | Chimie Appliquée et des Matériaux |
| | TRAYNARD Philippe | Chimie - Physique |
| | VEILLON Gérard | Informatique fondamentale et appliquée |

*en congé pour études.

## PROFESSEURS SANS CHAIRE

| | | |
|---|---|---|
| MM | BLIMAN Samuël | Electronique - Automatique |
| | BOUVARD Maurice | Génie Mécanique |
| | COHEN Joseph | Electronique - Automatique |
| | GUYOT Pierre | Métallurgie Physique |
| | JOUBERT Jean-Claude | Physique du Solide - Cristallographie |
| | LACOUME Jean-Louis | Electronique - Automatique |
| | ROBERT André | Chimie Appliquée et des Matériaux |
| | ROBERT François | Analyse numérique |
| | SABONNADIERE Jean-Claude | Electronique - Automatique |
| | ZADWORNY François | Electronique - Automatique |

## MAITRES DE CONFERENCES

| | | |
|---|---|---|
| MM | ANCEAU François | Informatique fondamentale et appliquée |
| | CHARTIER Germain | Electronique - Automatique |
| Mme | CHERUY Arlette | Automatique |
| | CHIAVERINA Jean | Biologie, biochimie, agronomie |
| | IVANES Marcel | Electronique - Automatique |
| | LESIEUR Marcel | Mécanique |
| | MORET Roger | Physique nucléaire - corpusculaire |
| | PIAU Jean-Michel | Mécanique |
| | PIERRARD Jean-Marie | Mécanique |
| Mme | SAUCIER Gabrielle | Informatique fondamentale et appliquée |
| | SOHM Jean-Claude | Chimie Physique |

INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

CHERCHEURS DU C.N.R.S. (Directeur et Maîtres de Recherche)

| | | |
|---|---|---|
| M | FRUCHART Robert | Directeur de Recherche |
| MM | ANSARA Ibrahim | Maître de Recherche |
| | BRONOEL Guy | Maître de Recherche |
| | CARRE René | Maître de Recherche |
| | DAVID René | Maître de Recherche |
| | DRIOLE Jean | Maître de Recherche |
| | KLEITZ Michel | Maître de Recherche |
| | LANDAU Ioan-Doré | Maître de Recherche |
| | MERMET Jean | Maître de Recherche |
| | MUNIER Jacques | Maître de Recherche |

Personnalités habilitées à diriger des travaux de recherche (Décision du Conseil Scientifique)

E.N.S.E.E.G.

| | | |
|---|---|---|
| MM | BISCONDI Michel | Ecole des Mines ST ETIENNE (dépt.Métallurgie) |
| | BOOS Jean-Yves | Ecole des Mines ST ETIENNE (Métallurgie) |
| | DRIVER Julian | Ecole des Mines ST ETIENNE (Métallurgie) |
| | KOBYLANSKI André | Ecole des Mines ST ETIENNE (Métallurgie) |
| | LE COZE Jean | Ecole des Mines ST ETIENNE (Métallurgie) |
| | LESBATS Pierre | Ecole des Mines ST ETIENNE (Métallurgie) |
| | RIEU Jean | Ecole des Mines ST ETIENNE (Métallurgie) |
| | SAINFORT | C.E.N.Grenoble (Métallurgie) |
| | SOUQUET Jean-Louis | U.S.M.G. |
| | CAILLET Marcel | E.N.S.E.E.G. (Chimie Minérale Physique) |
| | COULON Michel | E.N.S.E.E.G. (Chimie Minérale Physique) |
| | GUILHOT Bernard | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | LALAUZE René | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | LANCELOT Francis | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | SARRAZIN Pierre | E.N.S.E.E.G. (Chimie Minérale Physique) |
| | SOUSTELLE Michel | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | THEVENOT François | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | THOMAS Gérard | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |
| | TOUZAIN Philippe | E.N.S.E.E.G. (Chimie Minérale Physique) |
| | TRAN MINH Canh | Ecole des Mines ST ETIENNE (Chim.Min.Ph) |

E.N.S.E.R.G.

| | | |
|---|---|---|
| MM | BOREL Joseph | Centre d'Etudes Nucléaires de GRENOBLE |
| | KAMARINOS Georges | Centre National Recherche Scientifique |

E.N.S.E.G.P.

| | | |
|---|---|---|
| MM | BERNARD Guy | Centre National Recherche Scientifique |
| | DAVID René | Centre National Recherche Scientifique |
| | DESCHIZEAUX Pierre | Centre National Recherche Scientifique |

E.N.S.I.M.A.G.

| | | |
|---|---|---|
| MM | COURTIN Jacques | Université des Sciences Sociales |
| | LATOMBE Jean-Claude | Institut National Polytechnique GRENOBLE |
| | LUCAS Michel | Université Scientique et Médicale GRENOBLE |

* * *

Je tiens à remercier Monsieur le Professeur R. PERRET, Directeur du Laboratoire d'Automatique de Grenoble d'avoir bien voulu m'accueillir au sein de son laboratoire et de me faire l'honneur de présider le jury.

Je suis tout particulièrement reconnaissant à Monsieur Z. BINDER, Ingénieur CNRS, qui m'a témoigné sa confiance en dirigeant mes travaux et qui m'a aidé avec efficacité et compétence.

Que Monsieur F. ANCEAU, Maître de Conférences à l'ENSIMAG, trouve ici l'expression de mes sincères remerciements pour ses remarques et pour sa participation au jury.

J'assure de ma chaleureuse reconnaissance Monsieur P. DESCHIZEAUX, Chargé de Recherche au CNRS, pour ses commentaires très constructifs et qui me fait l'amitié de siéger à ce jury.

Je désire également exprimer toute ma gratitude à Monsieur J.F. LEMAITRE, Directeur du Département d'Etudes et de Recherches en Automatique de Toulouse, qui m'a fait l'honneur d'examiner mon travail et d'accepter de le juger.

Je sais gré à Monsieur le Professeur D.G. LAINIOTIS, Directeur du Laboratoire des Systèmes à l'Université Buffalo de New York, de l'intérêt qu'il a porté à ce travail.

Que toutes les personnes du Laboratoire d'Automatique de Grenoble qui m'ont apporté leur aide ou qui m'on témoigné leur amitié, trouvent ici l'expression de ma sincère gratitude.

TABLE DES MATIÈRES


Pages

I - Introduction 1
I.1 - Avant propos ..... 1
I.2 - La commande hiérarchisée ..... 1
I.3 - Cadre de travail ..... 2
I.4 - Travaux actuels et but du projet ..... 6
I.5 - Plan du travail ..... 7

II - Spécifications du centre de décision
II.1 - Introduction ..... 11
II.2 - Les algorithmes de commande multivariable ..... 12
II.2.1 - Algorithme de commande multivariable de base 12
II.2.2 - Commande par poursuite d'objectifs ..... 12
II.2.3 - Commande multimodèle ..... 14
II.2.3.1 - Commande déterministe multimodèle ... 15
II.2.3.2 - Commande stochastique adaptative multimodèle ..... 16
II.3 - Fonctions automatiques de fond ..... 17
II.3.1 - Identification du procédé ..... 17
II.3.2 - Construction de nouveaux modèles ..... 17
II.4 - Fonctions d'organisation et de surveillance ..... 18
II.4.1 - Dialogue opérateur ..... 19
II.4.1.1 - Dialogue de type informatique ..... 19
II.4.1.2 - Dialogue de type automatique ..... 19
II.4.1.3 - Dialogue de structuration ..... 20
II.4.2 - Edition de journeaux et archivages ..... 20
II.5 - Caractéristiques communes aux fonctions réalisées dans le centre de décision ..... 20
II.6 - Caractéristiques recherchées dans le système informatique du centre de décision ..... 21
II.6.1 - Modularité ..... 21
II.6.2 - Puissance de traitement ..... 22
II.6.3 - Sécurité de fonctionnement ..... 23
II.6.4 - Transparence ..... 24
II.7 - Conclusions ..... 24

III - Systèmes à plusieurs processeurs : concepts de base 21
III.1 - Historique ..... 21
III.2 - Architectures matérielles ..... 29
III.2.1 - Système SISD ..... 30
III.2.2 - Système SIMD ..... 30
III.2.3 - Système MIMD ..... 31
III.2.3.1 - Système multiprocesseur ..... 31
III.2.3.2 - Système multicalculateur ..... 35
III.3 - Concepts de logiciel ..... 38
III.3.1 - Identification du parallélisme ..... 39
III.3.1.1 - Parallélisme implicite ..... 39
III.3.1.2 - Parallélisme explicite ..... 40
III.3.2 - Allocation des tâches et de ressources .... 40
III.3.2.1 - Ressources banalisées ..... 41
III.3.2.2 - Spécification fonctionnelle ..... 41
III.3.3 - Communication interprocesseur ..... 41
III.3.3.1 - Communication par mémoire commune .. 42
III.3.3.2 - Communication par messages ..... 42
III.4 - Conclusions ..... 42

IV - Structures du logiciel et du matériel du centre de décision 47
IV.1 - Introduction ..... 47
IV.2 - Méthodologie ..... 47
IV.2.1 - Concept de module ..... 47
IV.2.2 - Classification des modules automatiques ..... 48
IV.3 - Principes de construction d'un module du centre de décision 49
IV.4 - Implémentation ..... 52
IV.4.1 - Problèmes des messages à destinations multiples non spécifiées ..... 52
IV.4.2 - Schéma de la communication entre les modules 52
IV.4.3 - Topologies possibles d'interconnexion des modules ..... 54
IV.4.4 - Les niveaux du système ..... 55
IV.4.4.1 - Niveau utilisateur ..... 56
IV.4.4.2 - Niveau communication inter-processeurs 57
IV.5 - Interface de communication ..... 58

V - Description du fonctionnement général du système et structure matérielle de l'interface de communication 61
V.1 - Description du fonctionnement général ..... 61
V.2 - Structure matérielle de l'interface de communication .... 62
V.2.1 - Synoptique ..... 62
V.2.2 - Queue First In First Out intégrée ..... 64
V.2.3 - Mémoire adressable par le contenu ..... 65
V.2.4 - Registres de contrôle des données ..... 67
V.2.5 - Mécanisme d'accès direct à la mémoire ..... 67
V.2.6 - Unité de contrôle ..... 69
V.2.6.1 - Rappel sur la microprogrammation .... 69
V.2.6.2 - Structure de contrôle adaptée ..... 69
V.2.7 - Envoi d'un message ..... 73
V.2.8 - Codage des interruptions générées par l'interface ..... 73
V.2.9 - Travaux en cours ..... 74

VI - Protocoles d'envoi des messages
VI.1 - Introduction ..... 77
VI.2 - Les approches décentralisées les plus courantes ..... 77
VI.2.1 - Approche "plateau tournant" ..... 77
VI.2.2 - Approche du "pion baladeur" ..... 78
VI.2.3 - Approche d'insertion par registre à décalage 78
VI.3 - Approche proposée ..... 80
VI.3.1 - Considérations de conception ..... 80
VI.3.2 - Similitudes avec la technique d'insertion par registre à décalage ..... 80
VI.3.3 - Protocole implémenté pour modules localisés géographiquement ..... 81
VI.4 - Etudes de cas particuliers ..... 82
VI.4.1 - Toutes les interfaces émettent simultanément des messages à destinations multiples ..... 82
VI.4.2 - Les interfaces émettent des messages à tour de rôle ..... 84
VI.4.3 - Remarques ..... 85
VI.5 - Protocole proposé pour modules dispersés géographiquement 85
VI.6 - Technique d'attribution des priorités aux interfaces .... 88

VII - Application : Commande stochastique adaptative multimodèle 91
VII.1 - Introduction ..... 91
VII.2 - Aperçu sur l'algorithme de la commande stochastique .... 91
adaptative multimodèle ..... 91
VII.3 - Décomposition de l'algorithme et organigrammes ..... 93
VII.3.1 - Décomposition ..... 93
VII.3.2 - Organigrammes ..... 94
VII.4 - Ecriture des programmes au moyen des macroinstructions . 95
VII.4.1 - Programme de la branche secondaire ..... 95
VII.4.2 - Programme de la branche principale ..... 95
VII.5 - Résultats expérimentaux ..... 96
VII.5.1 - Réservation mémoire ..... 96
VII.5.2 - Temps d'exécutions ..... 96
VII.5.3 - Influence des "frais généraux" sur l'efficacité du circuit arithmétique ..... 97

VIII - Conclusions 99

Annexe 1 - Logiciel de base 101

Annexe 2 - Description détaillée du mécanisme de communication des messages 113

# CHAPITRE I

# INTRODUCTION

## I.1 - Avant propos

L'automatique se préoccupe actuellement de systèmes de plus en plus complexes, pour lesquels une structure de commande classique à un seul niveau est largement insuffisante. Les automaticiens se sont tournés vers la solution qui consiste à diviser le problème de la commande en problèmes plus simples, plus caractérisés, et à étudier et commander le système selon différents niveaux de complexité croissante. Il est dès lors proposé de faire appel à des structures de commande hiérarchisées qui présentent une plus grande facilité d'étude et de mise en oeuvre.

## I.2 - La commande hiérarchisée : [MES 71] [TIT 78]

La commande hiérarchisée est fondée sur la décomposition du système complexe global et la coordination des commandes des sous-systèmes. Les nombreuses études menées jusqu'ici ont abouti à la définition de trois niveaux de commande (fig. 1).

1) Le niveau commande directe régulation.

Il est caractérisé par sa liaison étroite avec l'instrumentation du procédé. Il joue le rôle important d'interface entre le procédé et les niveaux de commande supérieurs.

2) Le niveau d'optimisation : c'est le niveau de l'élaboration de la commande de base générée par un algorithme déduit de l'optimisation d'un critère (quantité, qualité, ...).

3) Le niveau d'organisation : Il a pour tâche de gérer l'ensemble des niveaux inférieurs. Grâce à une vue synthétique des problèmes, de l'état du procédé et des consignes données par les opérateurs, il peut déterminer la procédure de fonctionnement adéquate pour chaque niveau.

A chaque niveau, le système et son comportement sont traduits par un modèle différent. Plus on s'élève dans la hiérarchie des niveaux et plus le système est considéré de façon générale. C'est ainsi qu'apparaissent successivement les points de vue du physicien, de l'automaticien et de l'économiste.

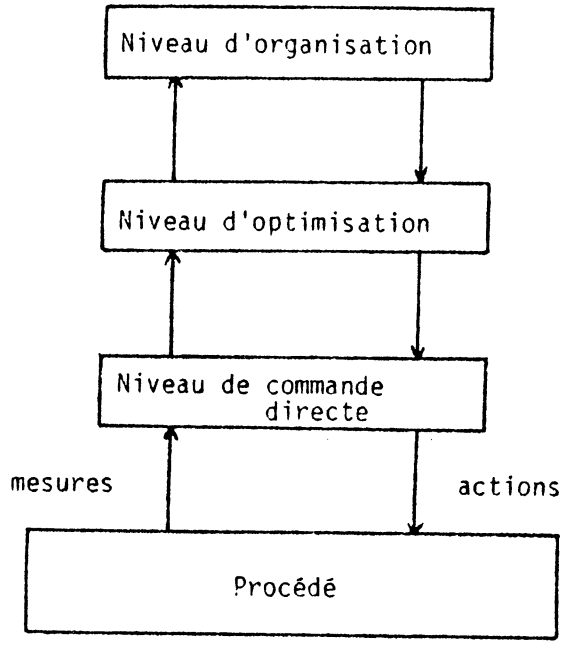



Fig. 1 : Décomposition en niveaux de la commande.

La répartition des fonctions aux différents niveaux dépend de la nature de l'action envisagée ainsi que de la fréquence à laquelle elle doit être exécutée. De même, les perturbations se répartissent sur les différents niveaux selon leur fréquence et leur action sur le procédé.

## I.3 - Cadre de travail

### I.3.1 - Automatisation du procédé pilote de distillation du laboratoire d'Automatique de Grenoble

Dans le cadre de ses travaux sur l'automatisation des systèmes complexes, le Laboratoire d'Automatique de Grenoble étudie à titre d'exemple l'automatisation de son procédé pilote de distillation. Ce procédé est constitué de deux unités colonnes de distillations. L'unité colonne 1 et l'unité colonne 2 sont responsables de la distillation des produits mélangés dans une unité de mélange en ligne. Dans le cas de connexion en série, une unité alimente l'autre et dans le cas de connexion en parallèle, les deux unités sont alimentées à partir de l'unité de mélange en ligne.

La structure automatique de commande des systèmes complexes proposée [BIN 77] est obtenue par une extension naturelle de la structure de commande hiérarchisée. Cette structure est composée de centres de décision répartis sur plusieurs niveaux (fig. 2). Ceci permet une conception ascendante à partir d'une organisation complètement décentralisée où chaque sous système possède sa propre commande locale. On peut ainsi introduire sur les niveaux supérieurs des centres de décision qui coordonnent une partie des centres des niveaux inférieurs. Cette structure converge vers un centre de coordination suprême de l'ensemble complexe.

Cette structure utilise à l'intérieur de chaque centre un modèle de la partie du système conduite par ce centre. Il est clair que la qualité de la commande dépend de la représentativité de ces modèles.

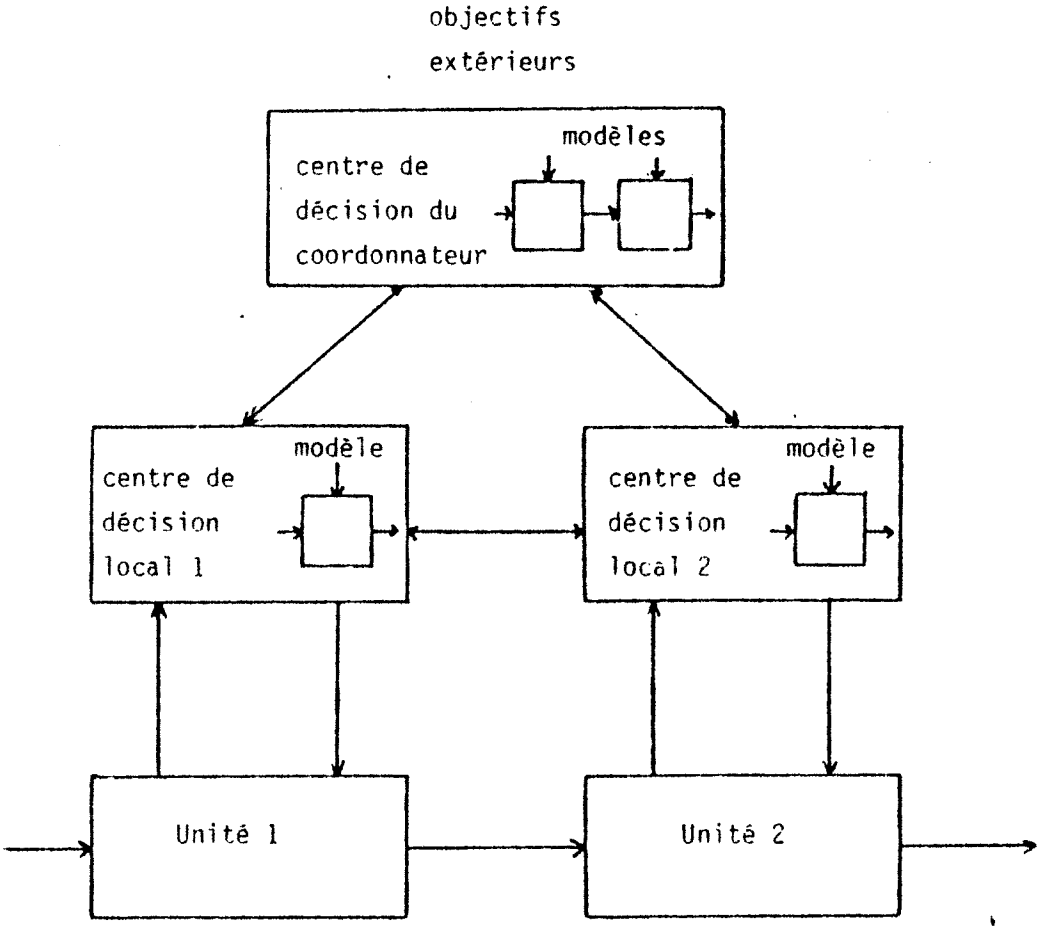



Fig. 2 : Structure automatique de la commande hiérarchisée et coordonnée.

Cette structure de commande automatique a été testée en mettant en jeu un centre local pour chacune des deux colonnes à distiller connectées en série et un centre global coordonnateur de l'ensemble.

Les modèles utilisés sont des modèles linéaires associés à des critères quadratiques de poursuite entrée-sortie. Le centre coordonnateur possède un modèle simplifié de l'ensemble de ces deux unités. Ce modèle a été pris simplifié parce que le modèle réel de l'ensemble des deux unités est considéré comme trop important. C'est d'ailleurs une des raisons de la décentralisation des algorithmes.

Cette structure a été implantée sur un minicalculateur T2000 suivant l'approche informatique centralisée en vue de la commande de l'ensemble des deux colonnes [REY 78].

Il faut noter qu'on distingue aujourd'hui deux approches informatiques pour la commande des procédés :

- approche informatique centralisée,
- approche informatique répartie.

I.3.2 - Approche informatique centralisée

C'est l'approche classique de commande de procédés par calculateur qui est basée sur le concept de la "salle de contrôle". Un calculateur unique est responsable de l'acquisition et du traitement des données, et de la gestion des périphériques (imprimante, console de dialogue ,...) (fig. 3).

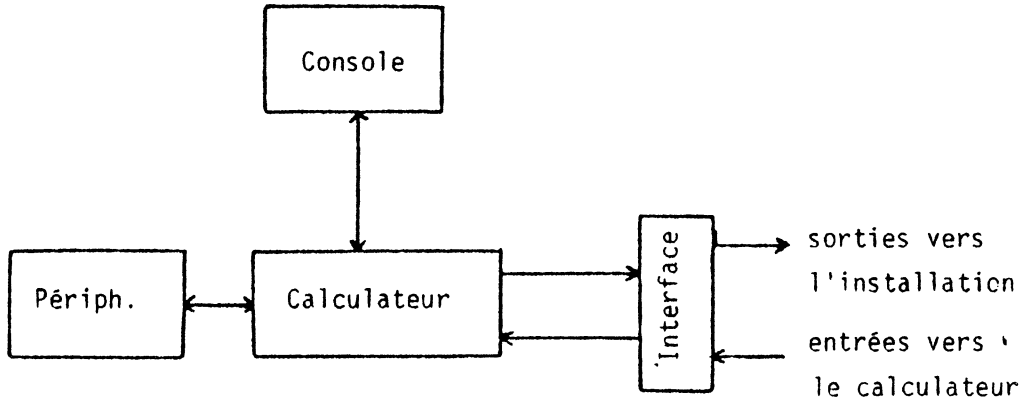



Fig. 3 : Structure de l'approche informatique centralisée.

La liaison entre les signaux et le calculateur est faite par l'intermédiaire d'une interface à laquelle sont raccordées tous les câbles en destination ou en provenance de l'installation. L'interface est constituée d'une matrice de multiplexage des signaux et elle est commandée par le calculateur. Elle peut contenir des convertisseurs analogique/digital et vice versa.

L'avantage de cette méthode est de disposer, dès le départ d'un logiciel sophistiqué à tendance universelle implanté sur le calculateur. Ce logiciel est capable de supporter des langages évolués de haut niveau pour la description du procédé et pour sa commande (logiciel temps réel multitâche pour monoprocesseur).

Ce logiciel peut devenir quelque fois très complexe lorsqu'il présume couvrir un grand champ d'application.

Les principaux inconvénients de cette approche sont :

- la nécessité d'une analyse très détaillée sur les différentes interactions entre les tâches qui s'entrelaceront de manière non déterministe, en raison d'évènements aléatoires, sur une seule unité centrale. C'est le problème aigu de la garde des temps d'exécutions des tâches.
- coût élevé du matériel et du logiciel surtout quand la sécurité du système est améliorée par l'existence d'un second calculateur pour la reprise en secours comme c'est fait couramment.

### I.3.3 - Approche informatique répartie

A l'origine, le coût important des matériels informatiques conduisit naturellement à concevoir des systèmes à base d'un seul processeur. L'évolution galopante de la technologie des circuits intégrés et les prix décroissants des circuits de grande complexité (microprocesseur, contrôleur de communication, ...) ont eu un impact important sur la conception des systèmes informatiques et en particulier sur les systèmes de commande de procédés industriels.

L'évolution du microcalculateur comme un outil industriel puissant et fiable fit que le concept du traitement informatique réparti est devenu

une alternative très prometteuse vis à vis de l'approche du traitement informatique centralisé. Du point de vue de l'automaticien, le traitement réparti peut être défini [JEK 78] comme :

C'est le concept de la séparation logique et physique des fonctions ainsi que de l'intelligence du contrôleur central sur plusieurs unités fonctionnelles chargées chacune d'une partie spécifique des tâches du système global.

Du point de vue de l'informaticien la définition du système réparti est plus complexe [ENS 78].

La répartition des fonctions de l'application sur plusieurs éléments de traitement est motivée, du moins théoriquement, par de nombreuses considérations :

a) L'application peut grandir par addition de nouveaux éléments de traitement qui exécuteront les nouvelles tâches sans une majeur modification du système déjà présent.

b) En cas de panne d'un élément, le système est capable de fonctionner en mode dégradé, ce qui permet d'obtenir une meilleure sûreté dans le matériel.

c) Le traitement réparti permet de conserver la morphologie du système fonctionnel, ce qui facilite la mise au point et la maintenance .

I.4 - Travaux actuels et but du projet

La structure automatique de commande hiérarchisée et coordonnée se prête bien à une implantation informatique répartie. Basé sur ces considérations, le Laboratoire d'Automatique de Grenoble a entamné l'implantation répartie,avec des microprocesseurs, de la commande automatique proposée.

Le respect de la morphologie de la structure automatique conduisit à la définition d'une structure informatique composée de trois systèmes fonctionnels interconnectés [IMP 78].

- Système assurant la fonction d'instrumentation et de régulation [ACQ 79],

- Système assurant les fonctions du centre de décision local : traitement algorithmique, dialogue opérateur et, visualisation,

- Système assurant la fonction de coordination entre les différents centres de décision locaux.

Ces travaux s'inscrivent dans le cadre du contrat DGRST IMPACT qui forme un cadre de notre projet. Celui-ci consiste dans la définition et la réalisation d'un centre local multiprocesseur qui permettra l'exploitation des algorithmes de commande multimodèle [LAI 76].

Ce travail a été subventionné par GIS Mini et Microinformatique Grenoble (GIS : Groupement d'Interêt Scientifique).

I.5 - Plan du travail

Après cette brève introduction, nous décrirons dans le deuxième chapitre les spécifications du centre de décision, en vue d'une implantation du centre sur une structure informatique.

Le troisième chapitre constitue un rappel des structures et des concepts informatiques pour les systèmes à plusieurs processeurs. Nous avons essayé de ressortir les caractéristiques essentielles des principales architectures existentes ainsi que de leurs inconvénients pour l'application en question.

Le quatrième chapitre définit la philosophie de la structure proposée, basée sur le concept des entrées et des sorties des modules.

Dans le cinquième chapitre nous décrirons le fonctionnement global du système ainsi que la structure matérielle de l'interface de communication associée à chacun des moduels.

Dans le sixième chapitre nous présentons le protocole d'émission des message du système de communication.

Le septième chapitre constitue une illustration des principes de base du centre en décrivant l'implantation d'une commande multimodèle.

En Annexe 1 est donné le logiciel de base développé pour le centre multimicroprocesseur.

L'Annexe 2 constitue une présentation détaillée du mécanisme de communication assuré par l'interface.

# BIBLIOGRAPHIE

[TIT 78] A. TITLI et al, "Analyse et commande des systèmes complexes" Edition CAPADUES - 1978 - Monographie A.F.C.E.T. - Division Automatique et Instrumentation.

[MES 71] M.D. MESAROVIC, "Multilevel Systems Theory : state of the art" 2nd IFAC Symposium on multivariable technical control systems - Düsseldorf - 1971.

[BIN 77] Z. BINDER, "Sur l'organisation et la conduite des systèmes complexes" - Thèse d'Etat - INPG - 1977.

[ENS 78] P.H. ENSLOW, "What is Distributed Data Processing System ?" - I.E.E.E. Computer - Janvier 1978.

[REY 78] D. REY, "Sur la commande décentralisée coordonnée. Application à un procédé pilote de distillation" - Thèse de Docteur Ingénieur - INPG - 1978.

[IMP 78] Rapport DGRST IMPACT

[ACQ 79] J.P. ACQUADRO, "Système Multiprocesseur pour l'Instrumentation et la Régulation" - MECO 79 - GRENOBLE - Juin 1979.

[LAI 76] D.G. LAINIOTIS, "Partionning : a unifying framework for adaptive system" - I.E.E.E. Proceedings - Août 1976.

[JEK 78] D.W. JENKINS, "Distributed or No ? The Choise Between Distributed and Central Computing Control" - Control Engineering Juin 1978.

# CHAPITRE II

# SPECIFICATIONS DU CENTRE DE DECISION

Spécifications du centre de décision local

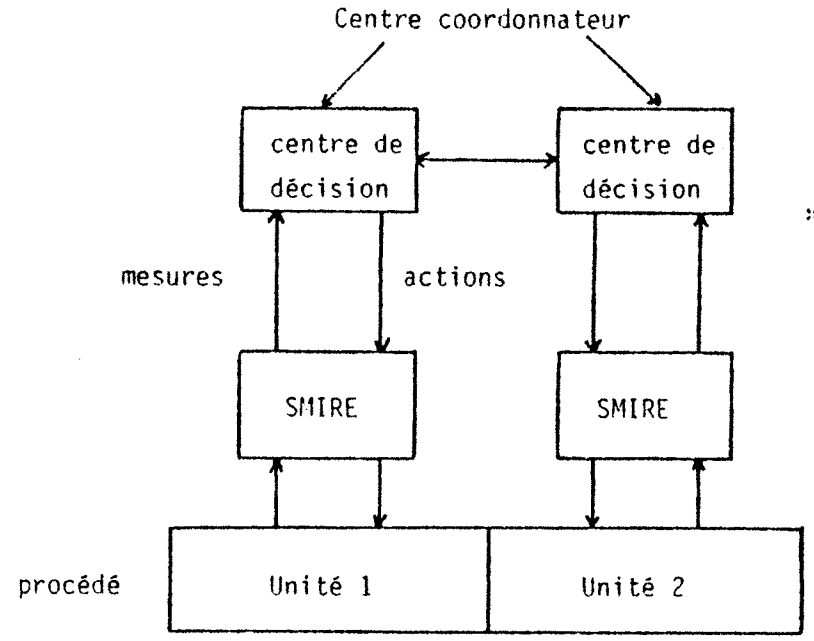
II.1 - Introduction

Le centre de décision local se situe au deuxième niveau de la structure hiérarchique de commande qui est celui de l'optimisation de certains critères économiques. Dans ce niveau est élaborée la commande de base qui détermine les points de consigne que devra faire exécuter le niveau inférieur qui est celui de l'instrumentation et de la régulation (fig. 1), assuré par le système SMIRE .

La commande est calculée soit pour asservir le système soit pour le réguler. L'asservissement du système consiste à commander ce système pour passer d'un point de fonctionnement à un autre selon une certaine dynamique. La régulation consiste à garder le système autour du même point de fonctionnement en compensant les perturbations qui éloignent le système de son point de fonctionnement.

Dans ce centre se déroulent :

- l'algorithme ou les algorithmes de commande multivariable,
- les fonctions automatiques de fond,
- les fonctions d'organisation et de surveillance.




* SMIRE : Système Multiprocesseur pour l'Instrumentation et la Régulation [ACQ 79]

Fig. 1 : Situation du centre de décision

## II.2 - Les algorithmes de commande multivariable :

La commande multivariable est une commande qui prend en compte les différentes interactions entre les paramètres à contrôler. La commande est rendue optimale par optimisation de certains critères comme la qualité ou la quantité du produit fini. Le choix de l'optimum est affecté par les conditions de l'ambiance du processus industriel et par la nature même de celui-ci. Diverses techniques mathématiques ont été appliquées aux problèmes d'optimisation des systèmes. Nous pouvons citer : la programmation dynamique et linéaire, le principe du maximum de Pontryagine, méthodes des gradients, ... La méthode d'optimisation choisie est un facteur déterminant dans la complexité numérique des algorithmes de commande. Ces algorithmes vont du simple jusqu'au très compliqué.

Nous considérons tout d'abord l'algorithme de commande multivariable de base.

### II.2.1 - Algorithme de commande multivariable de base

Un système linéaire multivariable est décrit sous forme d'équation d'état :

$$X(i+1) = A.X(i) + B.U(i)$$
$$Y(i) = C.X(i)$$

i est l'instant d'échantillonnage
U est le vecteur des entrées de dimension r
X est le vecteur des états de dimension m
Y est le vecteur de sortie de dimension n.

Les dimensions des matrices A, B et C s'en déduisent.

### II.2.2 - Commande par poursuite d'objectifs

Soit Zi l'objectif à poursuivre pour u, Zo l'objectif à poursuivre pour y. Nous cherchons la commande u(i) qui assure au mieux cette poursuite c'est-à-dire dans le cas considéré qui minimise un critère quadratique portant sur les erreurs de poursuite d'entrée et de sortie :

$$J = \sum_{i=0}^{N} (e_i^T(i) R e_i(i) + e_o^T(i+1) Q e_o(i+1))$$

où i est l'instant d'échantillonnage, N le nombre d'échantillons

$e_i$ = Zi-u est l'écart entre u et son objectif

$e_o$ = Zo-y

R et Q sont des matrices de pondérations de ces erreurs respectivement en entrée et en sortie avec $R > 0$ et $Q \geq 0$.

Le critère quadratique a été choisi pour la facilité de résolution de l'optimisation.

La commande qui découle est simple à appliquer en temps réel. Elle s'écrit sous la forme

$$u(i) = -LX(i) + n(i)$$

commande = bouclage + anticipation.

Le terme de bouclage dépend des paramètres du système à commander. La partie anticipation dépend de la dynamique des objectifs poursuivis [REY 78].

a) Objectifs constants

La commande s'écrit :

$$u(i) = -LX(i) + M_1 Zi - M_2 Zo$$

$M_1$ et $M_2$ sont calculées en différé.

b) Poursuite de modèle de référence (fig. 2)

La commande s'écrit :

$$u(i) = - LX(i) + M_1 Zi + N_1 Zi - N_2 X_R(i)$$

$M_1$, $N_1$ et $N_2$ calculées en différé.

Ces algorithmes simples ont été implantés [EYN 79] sur un microprocesseur 8080A avec une arithmétique en virgule flottante programmée. La commande est calculée au bout de 2 secondes pour un modèle avec 2 entrées, 2 sorties et 4 états.

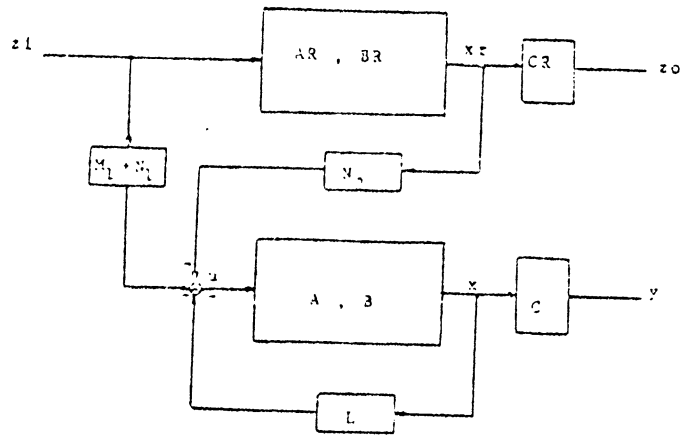



Fig. 2 : Algorithme de poursuite avec modèle de référence.

II.2.3 - Commande multimodèle

L'idée est née de l'étude de la commande des systèmes lors de grandes transitions. La description unique d'un système physique conduit à prendre en compte les non linéarités qui rendent la commande difficile à calculer. De plus la spécificité des non linéarités fait perdre à la commande ses possibilités de généralisation.

Devant ce problème de représentation du système, on fait appel à la modélisation linéaire déterministe, où lui est associé un ensemble cohérent et performant d'outils d'identification et de commande. La multiplicité des modèles linéaires peut résoudre le problème de la représentation du système et par conséquent on peut couvrir un large domaine de fonctionnement en affectuant à chaque modèle une zone restreinte de validité.

L'idée de base de cette structure multiple a été utilisée pour la première fois par MAGILL [MAG 65]. Les grands travaux dans ce domaine ont été réalisés par LAINIOTIS et son équipe à partir de 1970.

Cette technique a été appliquée dans trois domaines importants :

- dans l'aéronautique [ATH 77] ,
- dans l'étude des systèmes économiques [LAI 76],
- dans l'étude des systèmes énergétiques et physico-chimiques [MON 77] .

II.2.3.1 - Commande déterministe multimodèle : [MON 77]

La technique de la commande CDM repose sur deux parties principales : la localisation et la commande. La localisation consiste à positionner les modèles par rapport au système selon un certain critère de performance. Puis on applique la commande à partir du modèle le mieux situé.

La structure générale de la commande CDM est la suivante :

- A chaque modèle composant de la représentation multiple est associé une commande (dite de base) calculée à partir du seul modèle correspondant et en utilisant le critère de performance de la commande du système.

- Cette commande sera appliquée au système, tant que le modèle qui a permis de la calculer est encore valable à l'instant considéré. Ainsi, le problème consiste en une recherche à chaque instant du modèle représentatif du système. La fig. 3 montre la structure générale de la commande.

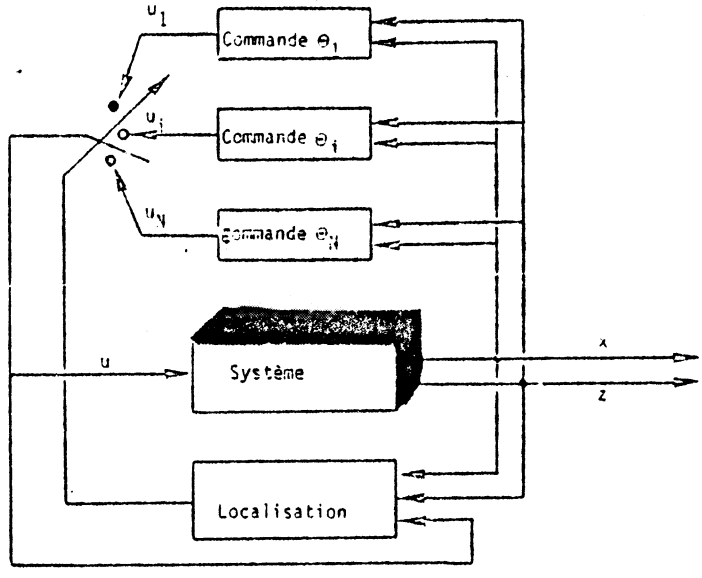



Fig. 3 : Commande déterministe multimodèle

### II.2.3.2 - Commande stochastique adaptative multimodèle [LAI 72] [JAW 78]

L'aspect stochastique des modèles utilisés intervient pour tenir compte des incertitudes paramétriques et structurelles des modèles et par conséquent pour avoir une représentation plus réaliste et plus naturelle d'une situation physique donnée.

La connaissance à chaque instant des valeurs du vecteur d'état est nécessaire pour le déroulement des différentes étapes de l'algorithme . Ayant accès aux entrées et aux sorties, il faut construire ce vecteur d'état à partir de ces données. Pour celà, on fait appel à la théorie de l'estimation. Un filtre de Kalman est utilisé pour construire le vecteur d'état à partir des mesures bruitées. Le calcul de la commande de base est obtenu à partir du théorème de séparation linéaire qui est appelé couramment Principe d'Equivalence Déterministe. Celui-ci exprime que la stratégie optimale peut être séparée en deux parties : une partie d'estimation, laquelle donne le meilleur estimé du vecteur d'état du système à partir des sorties observées, l'autre partie donne les matrices de bouclages comme dans le cas déterministe. La commande globale est obtenue par une recherche de toutes les commandes de base de tous les modèles candidats et leurs probabilités correspondantes. La structure d'une telle commande est donnée en fig. 4 [DES 73].

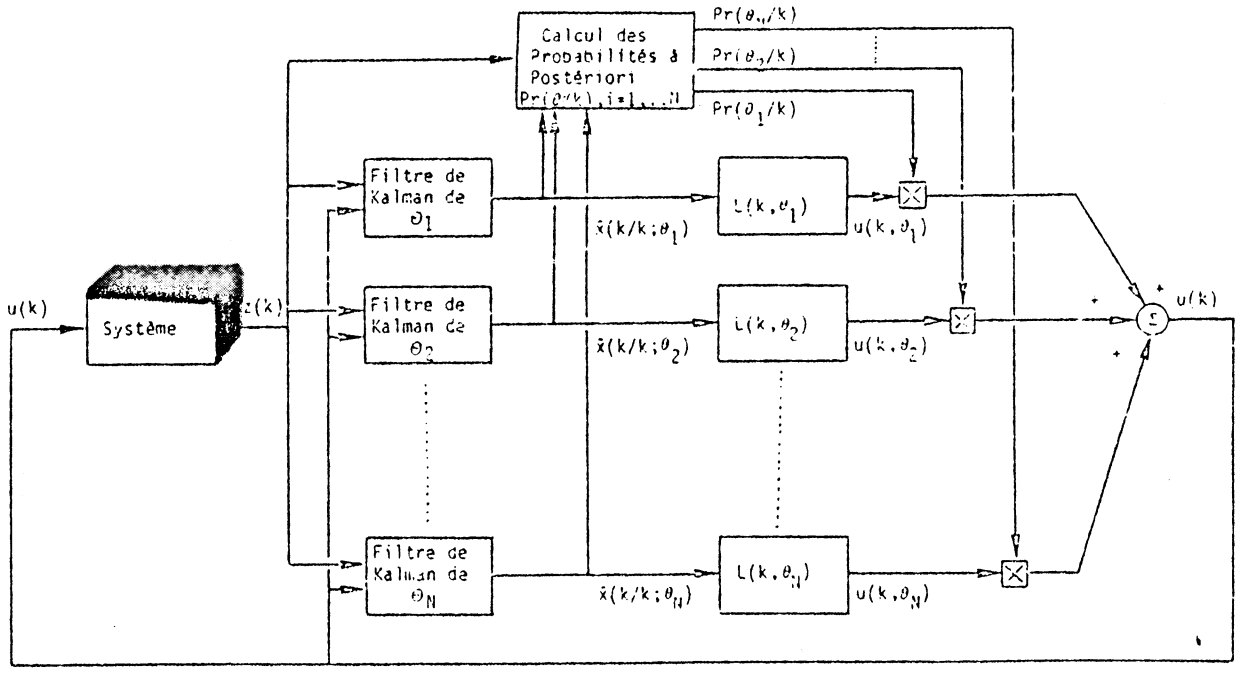



Fig. 4 : Structure de la commande stochastique adaptative multimodèle

L'algorithme de commande CSAM a été implanté [JAW 78] sur un IBM 360 du CICG (Centre Interuniversitaire de Calcul de Grenoble). Il a été écrit en Fortran IV. Cet algorithme occupe 117K octets. Au cours de la simulation, le temps d'exécution de cet algorithme augmentait linéairement avec le nombre N des modèles utilisés : $t_e \simeq 1{,}38\ N + 2.2$ secondes.

La commande multimodèle permet de réaliser une certaine forme de commande adaptative et présente une sécurité accrue grâce à l'utilisation simultanée de plusieurs algorithmes indépendants. Cette commande testée jusqu'ici en simulation, sera implantée dans le centre de décision pour l'expérimenter sur un procédé réel.

## II.3 - Fonctions automatiques de fond

### II.3.1 - Identification du procédé

Les paramètres physiques du procédé qui ont servi dans la construction du modèle peuvent changer avec le temps. Les causes de ces changements sont nombreuses :

- bruit sur les signaux,
- changement dans l'environnement tel que la température ambiante, l'humidité,
- déplacement du point de fonctionnement dans une région assumée linéaire alors qu'elle ne l'est pas.

Le rôle de l'identification est d'évaluer les nouveaux paramètres du procédé pour permettre une mise à jour du modèle du procédé.

Les techniques mathématiques utilisées sont longues et complexes.

### II.3.2 - Construction de nouveaux modèles

Les paramètres du procédé fournis par l'identification sont utilisés pour calculer les nouveaux coefficients du modèle qui remplacera l'ancien pour l'élaboration de la commande. Le nouveau modèle est caractérisé

par sa matrice d'état, la matrice de bouclage, et éventuellement les matrices $M_1$ et $M_2$ dans le cas d'une commande avec poursuite d'objectifs.

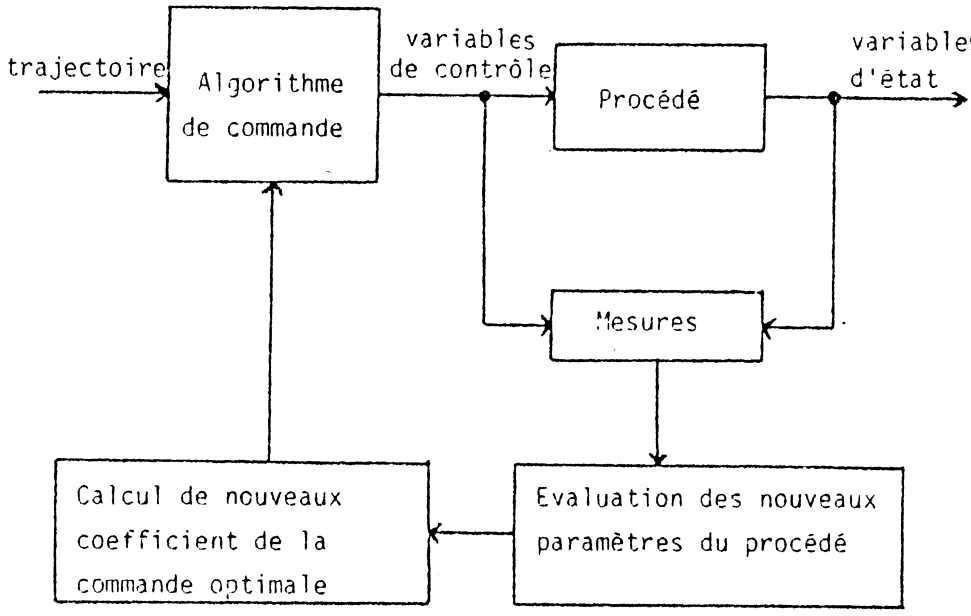



Fig. 5 : mise à jour des modèles

Nous disposons ainsi d'une identification active [BAD 77] qui permet de réactualiser le modèle défaillant en temps réel (fig. 5).

Le modèle est changé lorsque son indice de performance, caractéristique de sa validité de représentation, devient faible.

II.4 - Fonctions d'organisation et de surveillance

Les fonctions d'organisation s'occupent de la gestion et/ou de l'élaboration de cinq flots d'informations [ANC 78] :

- le flot de commande : ce sont les mesures qui servent à l'élaboration des commandes pour le procédé,

- le flot opérateur : c'est le flot des mesures caractéristiques du fonctionnement de l'installation,

- le flot d'archivage : c'est le flot des mesures à stocker,

- le flot de transfert des données de nature informatique, (programmes),

- le flot de sécurité : il contient les mesures de surveillance du procédé (dépassement d'un certain niveau....).

Ces différents flots sont gérés par deux fonctions :

- fonction dialogue opérateur,
- fonction d'édition de journeaux et d'archivage.

II.4.1 - Dialogue opérateur

L'automatisation d'une installation ne supprime pas la présence de l'opérateur de salle de contrôle. Cette présence peut être plus ou moins active selon le degré d'automatisation du procédé, continue ou passagère et selon que le procédé est figé ou en développement. L'opérateur doit avoir à tout instant une bonne vision de la marche de l'installation qu'il peut d'ailleurs être amené à reprendre en partie la commande en manuel. Nous pouvons distinguer trois types de dialogue [LAD 77] :

. dialogue de type informatique,
. dialogue de type automatique,
. dialogue de structuration.

II.4.1.1 - Dialogue de type informatique

Le dialogue de type informatique consiste à définir, à mettre en place et à maintenir tous les programmes, tous les outils informatiques nécessaires à un moment ou un autre à la réalisation de l'application. Mettre en place les programme c'est les charger, les assembler et les allouer à des processeurs à partir de périphériques d'entrée/sortie.

II.4.1.2 - Dialogue de type automatique

Ce dialogue consiste pour les opérateurs à recevoir des informations sur l'état du procédé, sur les évènements graves qui pourraient survenir sur l'évolution de variables représentatives ainsi que des renseignements concernant l'évolution probable du procédé ou sa gestion économique (état des stocks, quantités des matières, ...). Les opérateurs ont la possibilité de demander telle ou telle information de façon aléatoire ou de placer une variable sous surveillance constante.

II.4.1.3 - Dialogue de structuration

L'évolution constante des applications industrielles exige que soit souvent complétées, voire restructurées des parties de l'application. Il faut pouvoir les modifier facilement et en ligne sans arrêter l'exploitation du procédé et l'exécution des processus déjà définis. Un certain langage est alors nécessaire pour permettre à l'opérateur, d'une part de définir chaque processus, et d'autre part de décrire les liens qui peuvent exister entre eux.

Ce dialogue constitue le stade ultime de l'application et concentre en lui-même une large partie du développement du logiciel. C'est du au fait de l'utilisation des langages d'assemblages comme langages de base pour la programmation des microprocesseurs [SCH 78].

II.4.2- Edition de journaux et archivage

Certaines valeurs ou mesures en provenance du procédé peuvent être critiques, leurs contrôles par l'opérateur est fait par une édition continue de leurs valeurs ou par une représentation sous forme de courbes en fonction du temps sur un écran de visualisation.

L'archivage de certaines mesures permet d'avoir un historique du procédé qui sera utilisé pour les diagnostics et le dépouillement de statistiques en vue de l'établissement d'un bilan du procédé.

II.5 - Caractéristiques communes aux fonctions réalisées dans le centre de décision.

Nous pouvons tirer certaines caractéristiques concernant les fonctions à réaliser dans le centre décision :

- Malgré leur diversité de classes, les algorithmes de commande multivariable sont décomposables en blocs fonctionnels automatiques. Souvent, des algorithmes plus performant sont construitspar l'addition de fonctions numériques à un algorithme de base.

- Les algorithmes ont un degré élevé de traitement numérique. Une arithmétique en virgule flottante est nécessaire.

- Les fonctions du centre opèrent sur le même flot de données (valeurs des entrées, des états, des sorties, ...). Les données manipulées sont généralement des matrices ou des vecteurs de dimensions limitées jusqu'à ce jour. Des matrices d'état de dimension 6x6 ont été utilisées dans les algorithmes présentés. L'emploi de modèles avec des dimensions plus grandes est envisagé.

- Certaines fonctions opèrent sur des échantillons des données du flot. Par exemple la fonction de construction de nouveaux modèles. L'information peut être disponible plus qu'elle ne l'est consommée.

- Un fort parallélisme existe entre ces fonctions automatiques. Nous retrouvons la même caractéristique au niveau des algorithmes eux mêmes. Les algorithmes sont représentés la plupart du temps par des schémas blocs qui visualisent les interconnexions entre les fonctions automatiques constituant ces algorithmes. Ces fonctions sont fondamentalement parallèles ce qui est un cas exceptionnel en informatique [ANC 76]. Dans le cas de la commande multimodèle, le parallèlisme est idéal. C'est une commande qui se prête très bien au traitement parallèle.

## II.6 - Caractéristiques recherchées dans le système informatique du centre de décision

La terminologie actuelle des caractéristiques ou des objectifs visés par les systèmes informatiques est très vaste et très imprécise. Le lexique de ces objectifs contient déjà plus d'une quinzaine d'éléments [ENS 78]. On ne ressortira que les caractéristiques les plus significatives pour notre centre de décision.

### II.6.1 - Modularité

La modularité d'un système est dite parfois : flexibilité, extensibilité, ...

La modularité est le degré de modification des fonctions et des performances sans avoir à remanier la conception du système (des deux points de vue : logique et matériel).

Les deux principaux avantages de la modularité sont : les facilités de modifier et d'accroître le système :

a) Facilité de modification :

Elle va dans le sens de la simplicité dans le remplacement d'une fonction logique ou d'un élément matériel en dépit des dépendances avec d'autres éléments.

b) Facilité d'accroissement :

Elle permet :

- de tailler la configuration aux besoins immédiats,
- de modifier la configuration existente avec un minimum de manipulations logicielles aux niveaux de la synchronisation des processus et de la communication des données.

Le système informatique monoprocesseur conventionnel présente une modularité contestée. Par contre un système avec plusieurs processeurs offre potentiellement beaucoup plus de possibilités.

II.6.2 - Puissance de traitement

Elle est impliquée par le degré de complexités des calculs que peuvent présenter les algorithmes. La puissance de traitement diminue le temps de réponse du système informatique et réduit les risques de violation des contraintes de temps. La puissance de traitement d'un système monoprocesseur peut être augmentée de différentes manières : utilisation de technologie plus rapide (plus grande vitesse de commutation des portes élémentaires , ou bien améliorer le recouvrement (pipelining) des instructions. Il existe cependant certaines limites qui sont dues principalement à deux raisons :

- le multiplexage en temps réel des exécutions des fonctions sur une seule unité centrale augmente considérablement les frais généraux de commutation "overhead" ce qui conduit à une dégradation du temps de réponse.

- la taille et la complexité du logiciel croissent très rapidement d'autant plus qu'on veut se rapprocher des limites de performances du processeur .

Des architectures avec plusieurs processeurs permettent potentiellement de dépasser ces limites.

Le temps de réponse peut être diminué par addition de nouveaux processeurs. L'occurence d'évènements critiques peut activer un ou plusieurs processeurs spécialisés sans suspendre le traitement. Le prix de cette diminution du temps de réponse est une sous exploitation des processeurs.

La puissance de traitement se trouve améliorée par l'exécution des fonctions en parallèle sur plusieurs processeurs. En automatique, le parallélisme de traitement peut être revêtu d'un certain aspect de sûreté : les fonctions automatiques vitales peuvent être redondantes, ce qui facilite la surveillance et la reprise en secours.

Notons que dans le cas du procédé pilote du Laboratoire d'Automatique de Grenoble, considéré comme procédé lent, la commande doit être appliquée toutes les 5 secondes environ.

II.6.3 - Sécurité de fonctionnement :

Les incorrections de fonctionnement dans les systèmes de commande de procédé peuvent avoir des conséquences graves sur les plans humain ou matériel. La sécurité de fonctionnement est conditionnée d'une part par la structure générale du système informatique et d'autre part par les moyens mis en oeuvre pour la détection des erreurs.

Le principal moyen que puisse offrir la structure informatique pour améliorer la sécurité du système est de réduire au minimum la dépendance mutuelle des constituants du système.

Le principal avantage de la réduction du nombre de liens est une diminution des erreurs [JEN 78].

Il est souhaité qu'il y ait une correspondance directe ou homomorphisme entre la topologie réelle du système informatique et celle des algorithmes et des fonctions du centre de décision. Cet homomorphisme permet de bénéficier d'une collaboration efficace de l'automaticien dans la détection et l'analyse des erreurs.

La présence dans le système de plusieurs processeurs indépendants permet d'avoir un fonctionnement en mode dégradé en cas de défaillance d'un processeur. La décentralisation du vecteur d'état du système informatique contribue dans la validation de cette notion.

Le développement de moyens de détection des erreurs est une part importante dans la conception du système. Dans un système à plusieurs processeurs l'identification et l'isolation de la source d'erreur est possible grâce à des contrôles au niveau du destinataire des bits de parité ou du "checksum" de l'information qui lui est transmise. Cependant, comme l'expérience le montre, le processeur en panne peut provoquer des symptômes de défaillance chez d'autres processeurs. L'isolation et la correction de la faute sont très difficiles à effectuer. L'auto-détection des erreurs est un moyen efficace dans ces conditions (checksum des tables vitales, utilisation des minuteries pour l'exécution de tâches bien spécifiques).

Pour améliorer la sécurité de fonctionnement, le chemin de communication doit être redondant ou du moins pouvoir localiser facilement l'élément défaillant dans la communication pour l'isoler du reste.

Notons quand même qu'il n'y a pas de théorie générale sur la sécurité. Cette notion relève de très près des considérations techniques propre à chaque application.

### II.6.4 - Transparence

A partir de la console, l'opérateur doit pouvoir examiner les variables critiques dans le système, changer les paramètres et parfois même modifier la structure de la commande : scruter de nouvelles mesures, introduire des calculs de contrôle de grandeur, ... Ceci implique que des données et les paramètres dans le système soient référencées mnémoniquement et accessibles à l'opérateur [SCH 72], quelque soit leur lieu de résidence dans le système. Ce dialogue en quelque sorte interactif contribue au resserement du lien homme-machine.

## II.7 - Conclusions

Pour ces diverses raisons, une structure informatique à base de plusieurs processeurs peut répondre le mieux à nos besoins. Il reste à

définir les deux structures, matérielle et logique, du centre de décision parmi une grande variété de structures possibles et existentes. Une revue succinte des systèmes à plusieurs processeurs ainsi que les moyens de mise en oeuvre feront l'objet du chapitre suivant.

# BIBLIOGRAPHIE

[BIN 77] Z. BINDER, "Sur l'organisation et la conduite des systèmes complexes" - Thèse Docteur ès Sciences - Grenoble - 1977

[MON 77] B. MONNIER, "Contributions à la commande d'une classe de procédés dynamiques industriels dans de grands domaines de fonctionnement" - Thèse de Docteur Ingénieur - INP Grenoble 1977

[REY 78] D. REY, "Sur la commande décentralisée coordonnée. Application à un procédé pilote de distillation" - Thèse Docteur Ingénieur spécialité Automatique - Grenoble 1978

[JEK 78] D.W. JENKINS, "Distributed or No ? The choices between Distributed an Central Computing Control" - Control Engineering - Juin 1978 - pp. 61-64

[ENS 78] P.H. ENSLOW, "What is Distributed Data Processing System ?" - I.E.E.E. Computer, Janvier 1978 - pp. 13-21

[ACQ 78] J.P. ACQUADRO, P. LADET, D. REY, "Un système multicalculateur pour la commande d'un procédé industriel complexe" - Journées AFCET Informatique - Mai 1978

[SMI 79] J.P. ACQUADRO, "Système multiprocesseur pour l'instrumentation et la régulation" - MECO 79 - Juin 1979 - Grenoble

[JAW 78] S. JAWARI, "Sur la commande stochastique adaptative multimodèles" Thèse de 3ème Cycle - INP Grenoble 1978

[LAD 77] P. LADET, "Outils de structuration temps réel dans la commande des procédés industriels" - Thèse de 3ème Cycle - INP Grenoble 1977

[ATH 77] M. ATHANS, "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptative Control Method-Part I" - I.E.E.E. Trans. Automatique Control - Octobre 1977 - pp.

[SCH 72] J. SCHEFFLER, "The developement of process Control Software" - Spring Joint Computer Conf. - 1972 - pp. 907-914

[ANC 76] F. ANCEAU, P. DESCHIZEAUX, "Impact of Hardware Problems on Real Time Languages" - Proc. of the 1976 IFAC/IFIP Workshop on Real Time Programming - Rocquencourt 1976 - pp. 115-132

[LAI 76] D.G. LAINIOTIS, "Partitioning : a unifying framework for adaptative systems" - I.E.E.E. Proceedings - Août 1976

[MAG 65] D.T. MAGILL, "Optimal Adaptive Estimation of Sampled Stochastic Processes" - I.E.E.E. Trans. on Automatic Control - Octobre 1965

[LAI 72] D.G. LAINIOTIS, J.C. DESHPANDE, T.N. UPADHYAY, "Optimal Adaptive Control : a non-linear Separation theorem" - Int. J. Cont. - 1972 - vol. 15 - n° 5 - pp. 855-888

[SCH 78] G. SCHMIDT, R. SWIK, "Microcomputers in Automatic Control Applications. A software Point of View" - Preprints of the Seventh Triennal World Congress of the IFAC - Juin 1978

[ANC 78] F. ANCEAU, "Application des structures de machine de type CORAIL à l'automatisation décentralisée de processus complexes" - Journées AFCET Informatique Multiprocesseurs et Multiordinateurs en temps réel - Paris - Mai 1978

# CHAPITRE III

# SYSTEMES A PLUSIEURS PROCESSEURS

## CONCEPTS DE BASE

# Systèmes à plusieurs processeurs : Concepts de base

## III.1 - Historique

La notion de traitement parallèle a eu sa génèse dans la théorie des automates cellulaires. Des systèmes regroupant un grand nombre de cellules élémentaires de traitement qui travaillent en coordination ont été réalisés vers les années 60. Les exploitations futures de tels systèmes n'ont pas donné les résultats escomptés si l'on juge le succès par le nombre d'exemplaires réalisés.

Une idée nouvelle est alors apparue. Elle consiste à connecter des processeurs simples, plus intelligents que les cellules, via un certain réseau. L'idée était plus prometteuse sur les plans de faisabilité et de pratique. En 1964, une machine (CDC 6600 de Control Data) formée d'un processeur central puissant entouré d'une périphérie de dix processeurs ouvrit l'ère d'une nouvelle forme de parallélisme qui va subsister jusqu'à nos jours.

Le développement spectaculaire de la technologie des circuits LSI et l'avènement des microprocesseurs vers l'année 1970 ont donné beaucoup d'espoir dans la réalisation de systèmes économiques et pratiques. Les prix relativement faibles des CPU ont fait que l'évolution des systèmes informatiques fut accélérée vers les systèmes à plusieurs processeurs.

Nous discuterons dans les paragraphes suivants les différents types de systèmes informatiques existants selon deux points de vue : architecture matérielle et structure logicielle.

## III.2 - Architectures matérielles

Les systèmes informatiques sont classés actuellement en trois grandes catégories suivant le nombre d'instructions exécutées simultanément et suivant le nombre de flots de données qui entrent dans le système [FLY 72] :

1 - Système SISD (Single Instruction, Single Data Stream)
2 - Système SIMD (Single Instruction, Multiple Data Stream)
3 - Système MIMD (Multiple Instruction, Multiple Data Stream).

III.2.1 - Système SISD

C'est l'organisation d'un ordinateur classique (fig. 1). Nous n'entrerons pas en détail.

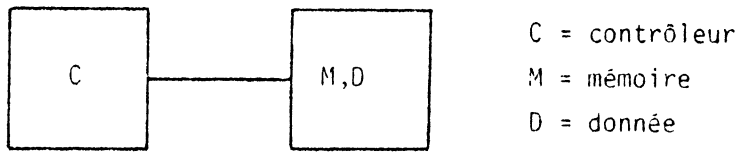



Fig. 1 : Système SISD

III.2.2 - Système SIMD

Dans ce type de système, un contrôleur unique exécute une instruction qui porte sur plusieurs arguments à traiter simultanément par le même code opératoire (fig. 2).

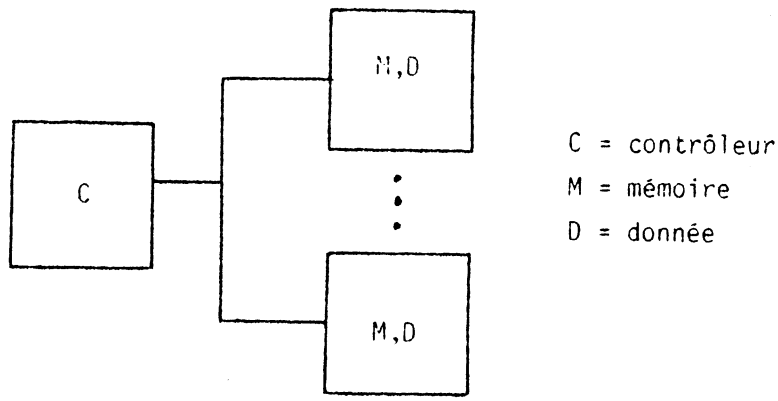



Fig. 2 : Système SIMD

Ce type de système est destiné à des traitements vectoriels ou matriciels décrits par des langages de programmation spécialisés de haut niveau (ex : APL). Les processeurs sont regroupés le plus souvent sous forme d'un tableau où chaque élément est capable de communiquer avec les quatres éléments qui lui sont adjacents (Solomon[SLO 62], ILLIAC IV [BAR 68]). Beaucoup d'autre moyens de connexion de processeurs ont été réalisés ou proposés.

Ces systèmes sont utilisés pour des applications de grande complexité numérique telles que la modélisation du temps ou la prévention

contre les collisions d'avions dans l'environnement d'un aéroport (calculateur STARAN dans l'aéroport de HOUSTON aux U.S.A.). Le principal inconvénient de ce type de système, à part son coût prohibitif, est son domaine très restreint d'applications.

Il est utile de rappeler que c'est la catégorie des supers calculateurs.

III.2.3 - Système MIMD

On distingue deux classes dans cette catégorie :

- système multiprocesseur MIMD
- système multicalculateur MIMD

la dénomination de ces deux classes a semé la confusion un certain moment dans la littérature.

III.2.3.1 - Système multiprocesseur MIMD

Le système multiprocesseur MIMD (ou multiprocesseur tout court) est caractérisé par :

1 - une mémoire commune accessible par tous les processeurs,
2 - des entrées/sorties (E/S) partagées et allouées aux différents processeurs suivant leurs demandes,
3 - un système d'exploitation unique qui gère l'ensemble des ressources et les alloue au fur et à mesure qu'elles sont demandées.

Les conditions 1 et 2 n'excluent pas la possibilité d'associer à chaque processeur une certaine mémoire privée ou bien de réserver certaines voies d'E/S pour les processeurs spécifiques.

Le trait le plus important est l'unicité du système d'exploitation.

L'espace d'adressage est commun à tous les processeurs.

La communication entre les processeurs est faite par dépôt de l'information dans des tables en mémoire à des endroits réservés pour les processeurs destinataires.

Plusieurs méthodes sont utilisées pour connecter les processeurs à la mémoire.

a) Une interface ayant autant de connecteurs que de processeurs est attachée à la mémoire (fig. 3).

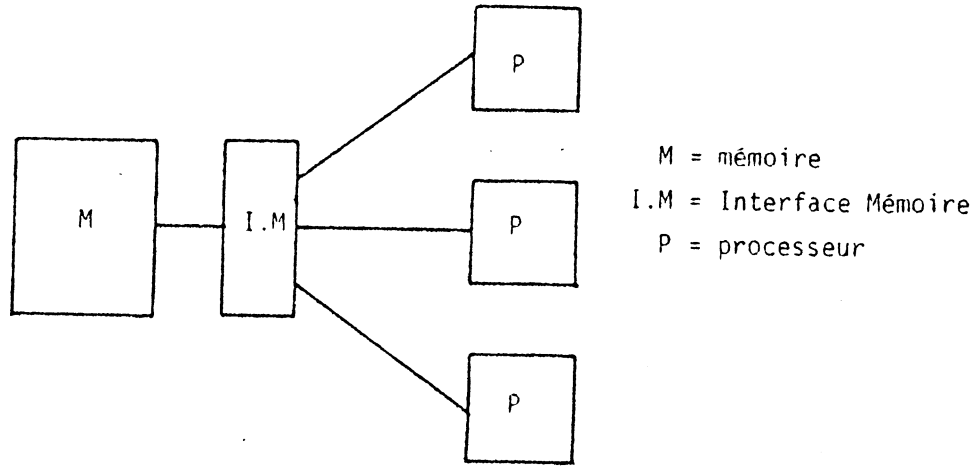
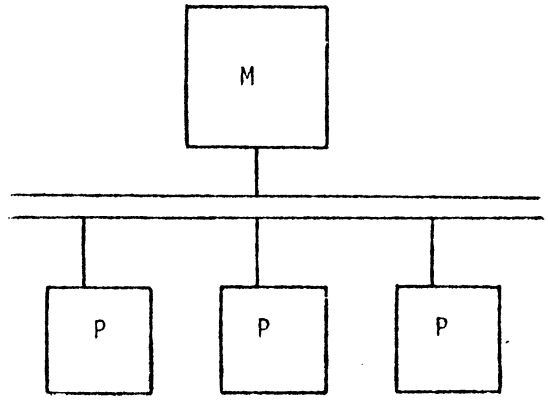



Fig. 3 = Mémoire commune gérée par une interface.

Ainsi chaque processeur possède son propre bus de connexion. Le contrôle d'accès à la mémoire est réalisé par l'interface mémoire qui règle les conflits selon un certain schéma de priorité.

L'inconvénient majeure d'une telle structure est sa modularité médiocre. L'addition d'un nouveau processeur est impossible sans la modification de l'interface mémoire et de son mécanisme de contrôle d'accès. Cependant certains estiment [SPE 77] qu'avec une mémoire vive d'un temps d'accès égale à 70 ns (donc très rapide), il est possible de connecter efficacement six ou huit processeurs.

b) L'accès à la mémoire est effectué au moyen d'un bus commun (fig. 4).




P = processeur
M = mémoire

Fig. 4 : bus commun

Dans ce cas, le contrôle d'accès au bus est effectué par un mécanisme d'allocation du bus qui peut être centralisé ou réparti. Différentes stratégies de contrôle d'accès sont possibles [SCR 77] :

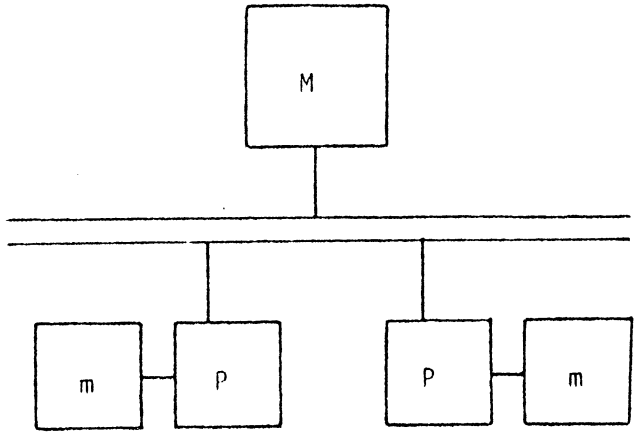
- contrôle par chaînage successif,
- contrôle par sélection,
- contrôle par requêtes individuelles,
- contrôle par requêtes indépendantes.

L'accès à la mémoire étant toutefois unique, une seule communication est possible à la fois. La fréquence des communications est limitée par la bande passante de la mémoire.

Lorsque la mémoire commune est utilisée comme organe de stockage des programmes et des données en plus que moyen de communication, le nombre de processeurs qui peuvent travailler efficacement en parallèle se trouve fortement diminué. C'est une structure limitée pratiquement à deux ou à trois processeurs.

Lors d'une spécialisation partielle des processeurs, certains programmes peuvent être implantés dans des mémoires locales aux processeurs (fig. 5).

La charge du bus se trouve ainsi bien réduite.




P = processeur
M = mémoire
m = mémoire locale

Fig. 5 : Bus avec mémoire locale pour chaque processeur

c) Dans le but d'augmenter la bande passante de la mémoire commune, plusieurs structures ont été conçues. L'élément commun de ces structures est la partition de la mémoire commune en N blocs séparés, accessibles indépendemment par les processeurs. La connexion des P processeurs aux N blocs mémoire est faite :

- par l'emploi du bus réservé (fig. 6) :

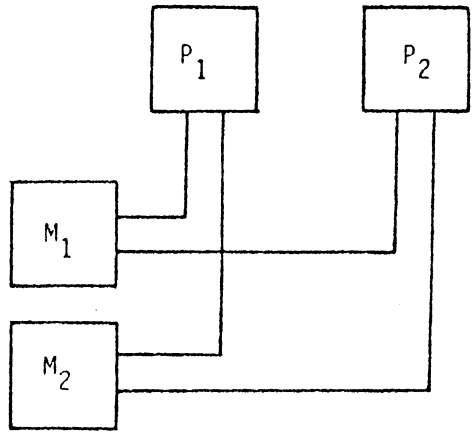



Fig. 6 : Partage de la mémoire commune en blocs avec accès par bus privés.

Dans cette structure chaque processeur possède plusieurs ports dont chacun est relié à un bloc de mémoire. Cette structure possède deux inconvénients majeurs : beaucoup de câbles et une modularité médiocre.

- par l'emploi d'une matrice de commutation (fig. 7) :

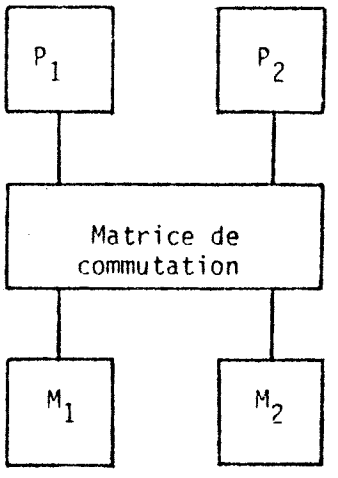



Fig. 7 : Système avec matrice de commutation (cross-bar)

L'inconvénient de la matrice de commutation est que sa complexité croît avec la largeur du bus et avec le produit P x M. Le contrôle du cross-bar peut être assez complexe.

Les avantages majeurs des systèmes multiprocesseurs sont :

- échange d'information interprocesseur très rapide (processeurs fortement couplés),

- grande reconfigurabilité du système sous le contrôle du logiciel. Les processeurs sont tous identiques et se partagent le même espace d'adressage. Il est relativement aisé d'activer un processus sur n'importe quel processeur.

Les principaux inconvénients sont les conflits du logiciel sur les ressources partagées (telles que tables en mémoire) et la centralisation du vecteur d'état du système. La largeur du bus (16 ou 32 fils parallèles plus les signaux de contrôle) fait des systèmes multiprocesseurs des systèmes très localisés géographiquement à moins de sérialiser le bus et utiliser un câble de très haut débit ( 32 Mb/s) pour les grandes distances.

III.2.3.2 - Système multicalculateur MIMD

Dans ce type de système, chaque processeur exécute une tâche spécifique dans le cas idéal (fig. 8)

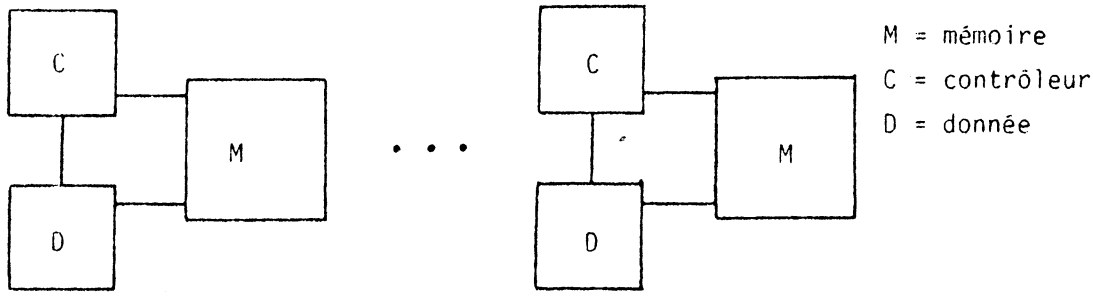



Fig. 8 : Système multicalculateur

Les processeurs sont généralement dispersés géographiquement.

Les informations échangées entre les différents processeurs sont peu nombreuses relativement au flot d'information du système tout entier. Il n'existe pas un système d'exploitation unique. Chaque processeur du système peut opérer indépendemment des autres mais doit coopérer avec eux pour former le système global. C'est une structure très modulaire par nature. Les processeurs peuvent être matériellement identiques mais différenciés par le logiciel.

Il existe plusieurs méthodes pour connecter ces processeurs. Les plus connus sont :

- la boucle (ou anneau),
- le câble commun dit aussi bus commun par abus de langage.

a) La boucle

Dans cette structure les processeurs sont ordonnés (fig. 9)

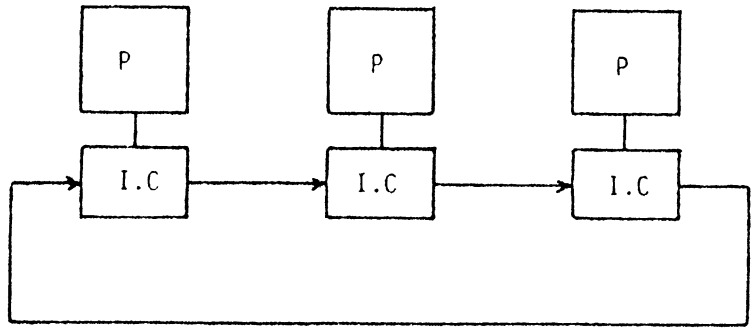



Fig. 9 : la boucle

Le chemin de communication est constitué par des tronçons de câbles reliant l'interface de communication d'un processeur avec celle du suivant.

L'information circule en général dans un seul sens de sorte que chaque processeur ne reçoit d'informations que de son prédécesseur et ne les transmet qu'au suivant. Un processeur Pi voulant transmettre un message à un processeur Pj, ajoute à l'information qu'il veut transmettre, l'adresse du destinataire puis la met sur la boucle. Chaque interface décide alors si le message lui est destiné ou pas. Dans l'affirmatif, elle le transmet au processeur qui lui est associée. Dans le cas contraire elle le retransmet à l'interface suivante.

Chaque interface doit régler le seul conflit existant entre un message qui lui est transmis et un autre qui est émis par son propre processeur. Ce contrôle peut être réparti comme dans le système DCS [FAR 73] ou centralisé comme dans le système SPIDER [FRA 75]. En France, l'exemple le plus connu est le système table ronde [COU 74].

La technique de communication généralement utilisée est la transmission série synchrone. Le principal avantage de cette structure est sa modularité. Un nouveau processeur peut être inséré n'importe où dans la boucle. Cependant, une défaillance en un point quelconque du chemin de communication peut arrêter toute la communication (du moins entre les processeurs séparés par l'élément défaillant).

b) Le câble commun

Dans cette structure, tous les processeurs sont connectés à un bus unique sur lequel transite toutes les informations échangées entre les différents processeurs (fig. 10)

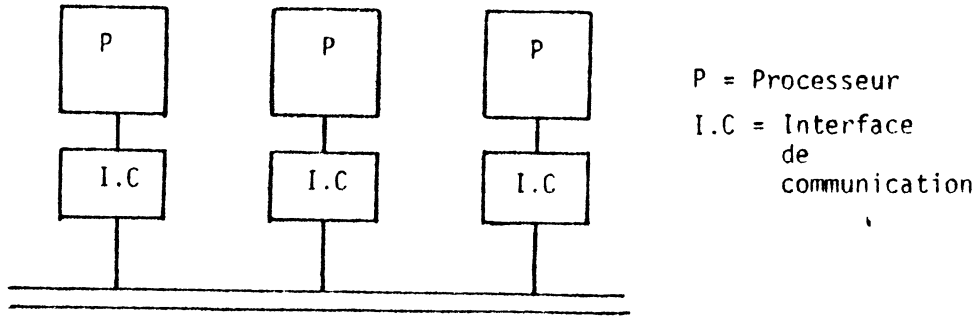



Fig. 10 : Câble commun

Un seul processeur à la fois possède le contrôle du câble. Ce contrôle est partagé temporellement entre les différents processeurs selon une certaine politique d'allocation. Le mécanisme d'allocation peut devenir assez compliqué quand les processeurs sont dispersés géographiquement. Nous citons quelques mécanismes récents d'allocation du câble commun :

- le VDPA (Vector-Driven Proportional Access) réalisé par Honeywell [JEN 78]

- l'Autoallocation développé dans le cadre du projet CANOPUS [MAR 79]

Le principal avantage de la communication par câble commun est la simplicité et l'extensibilité (si le mécanisme d'allocation le permet facilement). Son inconvénient est que la défaillance du câble implique un arrêt total de la communication. Le câble est considéré quand même légèrement plus sûr que la boucle du fait que :

- l'interface d'un câble est passive. L'information est effacée implicitement du câble par les constantes physiques de ce dernier,

- l'interface de la boucle est active. L'information est effacée explicitement de la boucle par l'interface.

Pour plus de détail sur les architectures des systèmes, la littérature est abondante. Quelques articles [RUS 77] [AND 75] [LIE 78] [LIP 78].

### III.3 - Concepts de logiciel

La conception d'un logiciel pour un système à plusieurs processeurs doit tenir compte des problèmes rencontrés sur les monoprocesseurs plus beaucoup d'autres nouveaux problèmes. Les problèmes sont :

- l'identification du parallélisme de l'application,
- l'allocation des tâches et des ressources,
- la communication entre les processeurs (ou les processus résidants sur ces processeurs).

Nous décrivons brièvement les techniques utilisées pour résoudre ces problèmes. Le choix d'une technique est certainement modulé par les spécifications de l'application.

### III.3.1 - Identification du parallélisme

La décomposition d'une application en plusieurs tâches élémentaires exécutables parallèlement fut l'objet de beaucoup d'efforts tant sur le plan théorique que sur le plan pratique. Les avantages potentiels du traitement parallèle sont une meilleure utilisation des ressources et une réduction du temps d'éxécution. La parallélisation de l'application peut être implicite ou explicite.

#### III.3.1.1 - Parallélisation implicite

a) Expression séquentielle associée à un identificateur de parallélisme.

Le texte du programme séquentiel décrivant l'algorithme (parfois en Fortran) de l'application est analysé par un identifieur de parallélisme qui détecte les tâches exécutables en parallèles [RAM 69] [RAM 71]. Les tâches sont allouées aux processeurs du système suivant des stratégies multiples [GON 72] [CHA 72]. L'allocation des tâches est basée sur le partage des processeurs d'une mémoire commune. C'est une technique viable dans un centre de calcul sous la condition que le programme objet soit destiné à tourner de nombreuses fois.

b) Expression par le flux (ou flot) de données.

L'algorithme de l'application est considéré comme un graphe orienté dont les noeuds représentent des opérateurs élémentaires (ex : additionneur) et les flèches représentent les chemins des données qui lient les différents opérateurs. Un opérateur devient actif (c'est-à-dire exécute son opération) lorsque toutes ses variables en entrée sont prêtes.

Ainsi plusieurs opérateurs peuvent fonctionner en parallèle et en asynchronisme suivant la fréquence des arrivées des variables sur les entrées [DEN 74] [RUM 77] [TRE 79]. L'intérêt de cette technique est de substituer le schéma de contrôle du parallélisme par celui des identificateurs des données. C'est une technique apparemment déterministe. Des études sont menées actuellement pour démontrer la faisabilité de la communication interprocessus non déterministe pour les systèmes pilotés par le flux des données [FRA 79].

III.3.1.2 - Parallélisation explicite

Nous pouvons distinguer grossièrement deux techniques :

a) Expression du parallélisme liée à la structure de données : opérateurs vectoriels par exemple. Le parallélisme est bien contrôlé mais il est relativement restreint. Cette technique est basée sur des langages utilisant des primitives de contrôle.

b) Expression par ordonnancement des tâches parallèles. Ce sont principalement les langages d'écriture de systèmes et de conduite de processus. Il est laissé au programmeur le soin de gérer l'ensemble correctement, avec l'aide de primitives de synchronisation : sémaphores, évènements, ...

Les différentes méthodes décrites jusqu'ici, sont basées sur l'emploi des langages évolués. Leur mise en oeuvre nécessite l'introduction d'un processeur de compilation ou de traduction. Celui-ci n'a besoin d'exister que dans des centres de calculs ou lorsque des développements de logiciel sont envisagés. Ces méthodes sont développées la plupart du temps sur des calculateurs universels. Dans la majorité des cas de contrôle de procédé, les différentes tâches et leurs intéractions sont bien connues. La répartition des tâches est prise en charge par le concepteur du système.

III.3.2 - Allocation des tâches et de ressources.

Lorsqu'un travail est découpé en tâches élémentaires, l'allocation des tâches peut être réalisée de différentes manières dont voici deux cas extrêmes :

III.3.2.1 - Ressources banalisées :

L'inconvénient dû à la disponibilité d'un processeur est limité ou diminué par l'introduction de plusieurs processeurs physiques en principe tous identiques. Le nombre de ces processeurs dépend du nombre moyen de tâches exécutables en parallèle et du rapport coût/performance.

Pour que ces processeurs puissent travailler effectivement en parallèle, il faut plusieurs accès simultanés à la mémoire. D'autre part, l'ensemble des processeurs doit pouvoir accéder à l'ensemble des périphériques. La banalisation des ressources est largement adoptée dans les structures multiprocesseurs.

III.3.2.2 - Spécialisation fonctionnelle :

Dans ce cas, chaque tâche élémentaire a les ressources nécessaires pour être exécutée. Un inconvénient de cette solution est la sous utilisation des processeurs logiques, ce qui n'est pas forcément grave, car les processeurs seront adaptés à des tâches spécifiques et seront donc moins complexes et moins coûteux que des processeurs d'utilisation générale et très sophistiqués.

Un grand nombre de systèmes à processeurs multiples sont dérivés de ces deux cas extrêmes. Ainsi, dans l'approche 1) on peut introduire un sous ensemble de processeurs spécialisés dans les entrées/sorties. Soit chaque processeur spécialisé utilise l'ensemble des périphériques, soit chacun est associé à un périphérique. Une approche partielle du type 2) souvent rencontrée dans des systèmes existants, consiste à implanter plusieurs tâches élémentaires sur un processeur muni de sa mémoire privée. Cette dernière approche caractérise les systèmes dit distribués. Vue la multiplicité de structures rentrant dans cette catégorie, une définition formelle des systèmes distribués fut établie par [ENS 78].

III.3.3 - Communication interprocesseur

Il existe deux mécanismes de communication interprocesseur suivant que l'espace d'adressage des processeurs est commun ou pas.

III.3.3.1 - Communication par mémoire commune

La mémoire commune sert de boîte aux lettres où les processeurs viennent déposer des informations dans les cases correspondantes aux processeurs destinataires. C'est un mécanisme utilisé dans les systèmes multiprocesseurs.

Le principal avantage de ce mécanisme est sa rapidité dans l'échange d'information. Par contre les inconvénients sont nombreux :

- modularité contestée,
- mise en oeuvre délicate lorsque les processeurs sont nombreux,
- danger potentiel d'interblocage des processeurs lorsque la primitive TEST and SET (TAS) est implémentée en logiciel. C'est le cas de tous les microprocesseurs actuels. Cet inconvénient est en vue d'être remédié dans la nouvelle génération de microprocesseurs par l'implémentation matérielle de la primitive TAS (INTEL 8086),
- mécanismes de protection compliqués.

III.3.3.2 - Communication par message

Elle consiste à transmettre d'un processeur à un autre les données elles-mêmes et les éléments nécessaires aux synchronisations.

Le principal avantage de ce mécanisme est de permettre à des processus autonomes de coopérer indépendemment de leurs lieus de résidences dans le système. Son inconvénient est sa faible bande passante due à la transmission effective des données et au décodage des messages.

C'est un mécanisme qui s'est avéré bien adapté aux structures multicalculateurs. Dans ces structures le flot de données qui circule entre les processeurs est beaucoup moins important que dans les systèmes multi-calculateurs. Les processeurs sont spécialisés dans des tâches spécifiques et ne se communiquent que juste le nécessaire.

III.4 - Conclusions

Il nous apparaît qu'il n'existe pas une structure unique qui soit la meilleure pour toutes les applications. Tout dépend du choix des critères posés et des spécifications de l'application. Malgré la décomposition de notre étude sur les systèmes selon deux points de vue, les deux structures matérielles et logicielles sont néanmoins fortement liées.

# BIBLIOGRAPHIE

[FLY 72] M.J. FLYNN, "Some Computer Organizations and Their Effectiveness" I.E.E.E. Trans. Computers - Septembre 1972 - pp. 948-960.

[SPE 77] W.L. SPETZ, "Microprocessor Networks" - I.E.E.E. Computer - Juillet 1977 - pp. 64-70.

[SCR 77] A. SCRIZZI, "Architecture matérielle d'un système multiprocesseurs" Thèse 3ème cycle (Paris VI) - Janvier 1977.

[FAR 73] D.J. FARBER, F.R. HEINRICH, "The Distributed Computing System" Proc. Seventh Annual I.E.E.E. Computer Society Internat. Conf. Février 1973 - pp. 31-34.

[FRA 75] A.G. FRASER, "SPIDER - An Experimental Data Communications System", Proc. I.E.E.E. Internat. Conf. on Communications - Juin 1974.

[JEN 78] E.D. JENSEN, "The Honeywell Experimental Distributed Processor. An Overview", I.E.E.E. Computer - Janvier 1978 - pp. 28-38.

[MAR 78] M. MARINESCU, P. NICOPOPOULOS, "Mécanisme de communication par bus série pour des réseaux informatiques locaux" - Actes du Congrès Informatique 78 - Editions Hommes et Techniques - Tome 1 - pp. 59-70.

[ECK 78] R.H. ECKHOUSE, J.A. STANKOVIC, "Issues in Distributed Processing. An Overview of Two Workshops" - I.E.E.E. Computer - Janvier 1978 - pp. 22-26.

[RAM 69] C.V. RAMAMOORTHY, M.J. GONZALEZ, "A survey of Techniques for Recognizing Parallel Processable Streams in Computer Programs" - AFIPS Conf. Proc. AFIPS Press - 1969 - pp. 1-15.

[RAM 71] C.V. RAMAMOORTHY, L.C. CHANG, "System Segmentation for the Parallel diagnosis of Computers" - I.E.E.E. Trans. Computer - Mars 1971 - pp. 262-270.

[GON 72] M.J. GONZALEZ, C.V. RAMAMOORTHY, "Parallel Task Execution in a Decentralized System" - I.E.E.E. Trans. Computer - Décembre 1972 - pp. 1310-1322.

[CHA 72] K.M. CHANDY, M.J. GONZALEZ, ...,"Optimal Scheduling Strategies in a Multiprocesseur System" - I.E.E.E. Trans. Computer - Février 1972 - pp. 137-146.

[DEN 74] J.B. DENNIS, "First Version of a Data Flow Procedure Language" Lecture-Notes in Computer Science, Springer-Verlag, NY - 1974 - pp. 362-376.

[RUM 77] J. RUMBAUGH, "A Data Flow Multiprocesseur" - I.E.E.E. Trans. Computer - Février 1977 - pp. 138-146.

[TRE 79] P.C. TRELEAVEN, "Exploiting Program Concurrency in Computing Systems" - I.E.E.E. Computer - Janvier 1979 - pp. 42-50.

[RUS 77] P.M. RUSSO, "Interprocessor Communication for Multi-Microcomputer System" - I.E.E.E. Computer - Avril 1977 - pp. 67-76.

[AND 75] G.A. ANDERSON, E.D. JENSEN, "Computer Interconnection Structures : Taxonomy, Characteristics, and Examples" - Computing Surveys - Décembre 1975 - pp. 197-213.

[LIE 78] B.H. LIEBOWITZ, "Multiple Processor Minicomputer Systems - Part 1 : Design Concepts" - Computer Design - Octobre 1978 - pp. 87-95.

[LIP 78] G.J. LIPOVSKI, K.L. DOTY, "Developments and Directions in Computer Architecture", I.E.E.E. Computer - Août 1978 - pp. 54-67.

[CON 63] M.E. CONWAY, "A Multiprocessor System Design" - AFIPS Conf. Proc. - 1963 - pp. 139-146.

[ENS 78] P.H. ENSLOW, "What Is Distributed Data Processing System ?" - I.E.E.E. Computer - Janvier 1978 - pp. 13-21.

[FRA 79] N.D. FRANCESCO, G. PEREGO,...., "On the Feasibility of Nondeterministic and Interprocess Communication construts in Data-Flow Computing Systems" - European Conference on Parallel and Distributes Processing - Toulouse - Février 1979 - pp. 93-100.

[SOL 62] D.L. SLOTNICK et al, "The Solomon Computer" - AFIPS Conference Proceefings - 1962 - pp. 97-107.

[BAR 78] G.H. BARNIER et al, "The ILLIAC IV Computer" - I.E.E.E. Trans. on Computers - Août 1968 - pp. 746-757.

[COU 74] M. COUR, "Description du système table ronde" - Automatisme - Octobre 1974.

# CHAPITRE IV

# STRUCTURES DU LOGICIEL ET DU MATERIEL DU CENTRE DE DECISION

# Organisation et principe du centre de décision

## IV.1 - Introduction

Avec les possibilités offertes aujourd'hui par les micro-processeurs, il paraît que la méthodologie de conception de systèmes à plusieurs processeurs converge vers la même méthodologie de conception de logiciel [SAM 76] : notion de modules, programmation structurée, ... Il est devenu difficile de séparer la conception matérielle d'un système de la philosophie générale de son organisation logicielle.

La démarche que nous avons suivie dans la définition de notre système à plusieurs processeurs est empruntée des principes généraux de conception de logiciels modulaires.

## IV.2 - Méthodologie

### IV.2.1 - Concept de module

En général, un système est conçu par étapes successives et par des personnes différentes. Pour qu'un tel système fonctionne correctement avec le moins de difficultés, il est nécessaire de le décomposer en plusieurs petits problèmes. La notion de module en programmation fut posée par Gauthier et Pont [GAU 70] : "une bonne segmentation du projet assure la modularité du système. Chaque tâche constitue un module distinct et séparé. Les entrées et les sorties du module sont bien définies au moment de son implémentation. Son interface avec d'autres modules ne comporte pas de confusion. Lors des tests, chaque module est testable dans son intégrité de la manière indépendante ; les problèmes de synchronisation avec d'autres tâches sont peu nombreux...".

La décomposition de l'application en modules offre plusieurs avantages [PAR 72] :

- possibilité de définir et de concevoir chaque module séparement, avec un minimum d'information sur les autres modules . Le résultat est une meilleure conception du système parce qu'il est mieux compris.

- Un module peut être modifié ou retiré sans problèmes majeurs pour les autres modules.

Cependant, il n'existe pas de critères généraux de décomposition en modules. Tout dépend de la nature intrinsèque et des possibilités de décomposition de l'application en question. Le découpage fonctionnel du traitement automatique est convenu comme le meilleur critère de décomposition en modules [DES 75].

IV.2.2 - Classification des modules automatiques

Le centre de décision est destiné à une application temps réel ; il sert son environnement tant qu'il continue de fonctionner normalement. Les modules du centre peuvent être classifiés suivant le caractère de continuité des fonctions qu'ils exécutent [ANC 76]. Un module fonctionnel est un élément qui effectue un traitement et qui élabore une sortie en fonction des entrées ainsi qu'en fonction de son état interne.

a) Modules continus

Ce sont les modules ayant des sorties et des entrées définies de manière permanente : par exemple module d'instrumentation.

b) Modules discrets ou discontinus

Ce sont des modules dont les sorties et/ou les entrées sont fonction du temps à des instants précis. Les sorties sont plutôt cycliques : par exemple, module de traitement algorithmique.

c) Modules temporaires

Ils sont caractérisés par leur possibilité de prendre deux états : actif ou inactif. Le module est mis à l'état actif pour exécuter un processus donné puis retourne à l'état inactif dès qu'il fournit ses résultats contrairement aux deux types précédents qui continuent leurs activités indéfiniment. Un exemple de processus sporadique : construction d'un nouveau modèle pour la commande. Le caractère sporadique de ce type de modules permet de les traiter sur monoprocesseur en multitâche.

Notons que cette classification correspond à une certaine hiérarchie de modules :

- les modules continus sont au niveau le plus bas donc tout près du procédé,

- les modules discrets au milieu,

- les modules temporaires sont au sommet de la hiérarchie.

La construction des modules continus et discrets avec des microprocesseurs doit être proche du mode de raisonnement des automaticiens, mode de raisonnement dû en grande partie à l'utilisation du matériel analogique et à la définition sous forme de blocs diagrammes des différents travaux nécessaires à la commande.

IV.3 - Principes de construction d'un module du centre de décision

Les modestes qualités des systèmes d'exploitation temps réel existants ont montré clairement la nécessité de recourir à des approches simples [BRI 75].

Une approche simple consiste à considérer le module comme un processus séquentiel défini et crée une fois pour toute après l'initialisation du système.

Le module est considéré comme un ensemble d'actions exécutables séquentiellement. Le nombre de processus est connu à priori ainsi que les activités parallèles entre ces processus. Il n'y a pas de créations dynamiques de processus.

Les règles principales qui régissent la construction d'un module sont :

a) Un module est constitué de la paire microprocesseur/mémoire privée. Il réalise des tâches spécifiques qui lui sont allouées statiquement.

b) Chaque module est lié à son environnement par un ensemble de variables définies en entrée ou en sortie du module. Le module est visible par ses variables déclarées en sortie.

c) La communication entre les modules est basée sur des messages à destinations multiples non spécifiées. Un message est émis dès qu'une information en sortie est disponible. L'information contenue dans le message est référencée par un nom. Le module ayant ce nom défini en entrée, est considéré un module consommateur de ce message.

d) Le processus du module continue en séquence lorsque tous les modules consommateurs éventuels possèdent des copies de l'information qu'il vient de fournir. L'émission d'un message par un module correspond à une action d'actualisation (ou de mise à jour) des copies que possèdent d'autres modules.

e) L'actualisation d'une entrée-module n'est possible que si l'accès est autorisé par le module possédant cette entrée. Il n'y a pas de tamporisation automatique de l'information. L'actualisation d'une variable est considérée comme un évènement.

f) Une copie dans un module est accessible par ce dernier en lecture seulement.

g) Un message ne renfermant pas de données effectives est considéré comme un signal.

Chaque module pilote les autres par les messages qu'il émet. Ses liaisons avec les autres sont définis implicitement par les noms des variables communiquées entre les modules. Ces variables sont dites les variables visibles du système. L'intérêt principal de cette organisation est la substitution du flot de contrôle de synchronisation par celui des données. Cette approche nous dispense de l'utilisation de signaux explicites de synchronisation entre les modules.

Le module est considéré comme un élément actif à l'encontre des autres approches qui le considère comme un élément passif. En effet prenons le cas typique de l'édition d'une valeur en temps réel. Dans une approche classique, le module d'édition reçoit un message, qui lui est destiné explicitement, renfermant une information structurée tel que nom, type et valeur de la variable. Cette action d'envoi explicite du message au module d'édition est soit programmée même dans le module qui

évalue cette variable, soit établie par l'intermédiaire d'un moniteur global. Notre approche consiste à définir cette variable comme une entrée du module d'édition. Ce module effectue une copie et imprime la valeur chaque fois que cette variable est émise sous forme d'un message à destinations multiples non spécifiées. Cette entrée-module peut être définie ou annulée à volonté.

Représentation graphique

Les liens du module avec son environnement sont établis implicitement par les références aux entrées-modules et aux sorties-modules de son interface. L'ensemble de ces variables constitue une partie de l'ensemble fini X des variables externalisées (ou visibles) dans le système.

Cette approche peut être représentée par un hypergraphe [BER 70] où les sommets représentent l'ensemble des variables externalisées et les arrêtes (ou surfaces) représentent les modules présents dans le système (fig. 1)

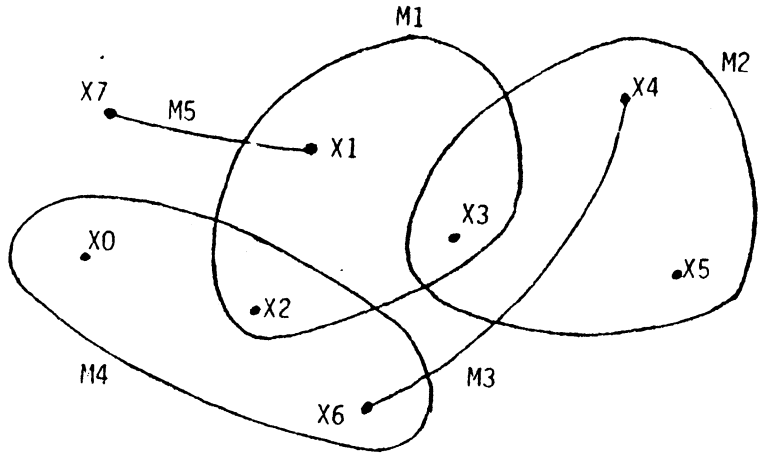



Fig. 1 : Exemple d'un hypergraphe

Cette représentation permet de ressortir l'entité des variables pilotant les modules.

## IV.4 - Implémentation

### IV.4.1 - Problèmes des messages à destinations multiples non spécifiées

Le fait que le message est destiné à tout le monde, le système de communication doit permettre d'avertir les n-1 autres modules de la tentative de mise à jour de leurs entrées-modules correspondantes. Deux problèmes sont à résoudre :

- comment obtenir un accusé de réception définitif quand la source du message ne connaît pas le nombre des destinations ?

- comment retransmettre le message sachant que des entrées-module l'ont reçu et d'autres pas.

### IV.4.2 - Schéma de la communication entre les modules

Avec le mécanisme des messages à destinations multiples non spécifiées, il est tout à fait normal qu'à un message (délivré par une sortie-module) corresponde plusieurs entrées-module (inconnues par la sortie-module). Les entrées-module peuvent ne pas être toutes prêtes à recevoir ce message (ou plutôt faire une copie du message). Le mécanisme de communication doit tenir compte de ce fait.

La récupération des défauts de transmission est aussi un problème à résoudre. Les entrées-module sont capables de détecter les erreurs des transmissions dans le message mais incapables de détecter les messages perdus. La perte de l'accusé de réception d'un message laisse la sortie-module et les entrées-module relatives à ce message dans un état ambigu. Un processus de réactivation du mécanisme de communication est nécessaire.

#### Schématisation du mécanisme

Quant une sortie-module émet un message, deux situations sont possibles pour les entrées-module : elles sont prêtes pour faire des copies ou ne le sont pas. Sachant que la sortie-module a délivré un message, les entrées-module doivent être considérées dans l'un des deux nouveaux états suivants :

a) entrée-module actualisée. Le transfert d'information eu lieu pour deux raisons :

- entrée-module était prête à faire une copie
- il n'y a pas eu d'erreur de transmission.

b) entrée-module non actualisée. Le transfert d'information n'a pas eu lieu pour l'une des deux raisons :

- entrée-module n'était pas prête pour faire une copie
- entrée-module était prête mais il y a eu erreur de transmission.

Ceci peut être schématisé par la fig. 2 où :

- les entrées-module pour une variable déterminée sont considérées comme des particules $M_i$ situées initialement dans le niveau 1.

- la première occurrence d'un message est considérée comme une excitation $Ex_0$ qui fait passer chacune des particules (les entrées-module) au niveau 3 ou 2 suivant que le transfert d'information fut accompli ou pas respectivement.

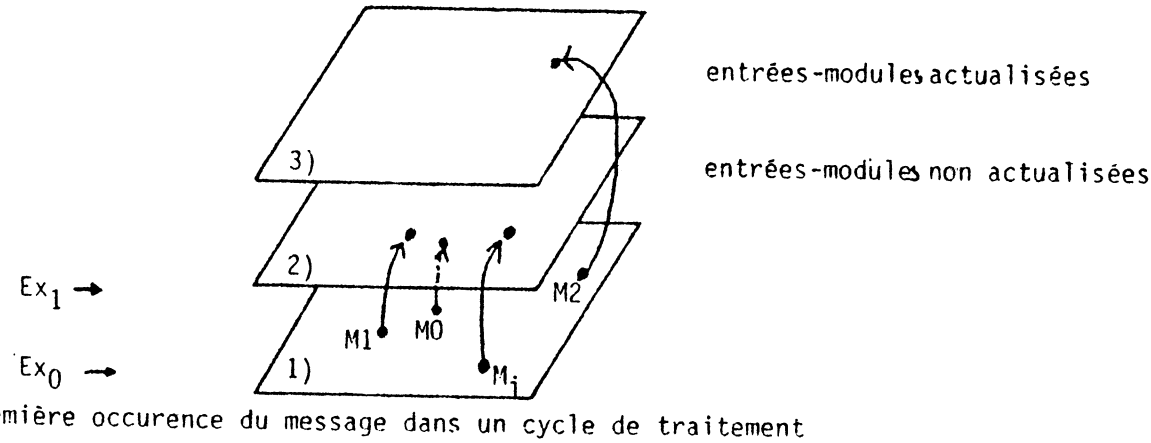



Fig. 2 : Niveaux possibles pour les entrées-modules.

Les entrées actualisées peuvent accepter, après un certain moment plus ou moins long, une nouvelle excitation $Ex_0$ pour commencer un nouveau cycle de traitement. Ce cycle ne doit pas commencer avant que toutes les particules du niveau 2 passent au niveau 3. Ce passage ne peut pas être amorcé par une répétition de l'excitation $Ex_0$. Ces excitations seront considérées par les particules du niveau 3 (entrées-modules actualisées) comme des excitations de débuts de nouveaux cycles de traitement. Pour éviter cet asynchronisme, il est nécessaire de différencier la première excitation $Ex_0$ de toutes les autres excitations qui ont pour rôle d'amener les particules du niveau 2 au niveau 3. Ceci se traduit concrètement par la nécessité d'assigner un certain code opération aux messages retransmis. Ceci correspond dans notre schéma à l'excitation $Ex_1$. Cette excitation sera entretenue automatiquement chaque fois qu'il y a une erreur de transmission dans le message. Quand une entrée-module du niveau 2 devient prête à être actualisée, il faut qu'elle provoque une excitation $Ex_1$ pour l'amener jusqu'au niveau 3. Ceci se traduit concrètement par un message de demande de retransmission de la sortie-module effectué par l'entrée-module en question.

La technique proposée permet de récupérer les erreurs de transmission sans nuir à la synchronisation du système. Le deuxième type d'erreurs, perte du message, est plus difficile à résoudre. Le message est considéré perdu quand la source du message ne reçoit pas d'accusé de réception. La récupération de ce type d'erreur est délicate mais possible. La technique est proche de celle des transactions [SCH 78]. Elle est basée sur la poursuite de la trace du message. La technique est exposée plus loin.

### IV.4.3 - Topologies possibles d'interconnexion des modules

La diffusion du message vers tous les modules est facilement obtenue dans deux topologies d'interconnexions :

- topologie d'interconnexion en anneau,
- topologie d'interconnexion par bus.

Dans l'anneau, le message est diffusé dans tous les modules après un tour complet.

L'accusé de réception final, qui intéresse le module source du message, peut-être obtenu par un "ou" des accusés de réception élémentaires de tous les autres modules.

- Dans un bus, ce "ou" est câblé.
- Dans un anneau, ce "ou" est logique. Un champ spécial dans le message, accessible par tous les modules, est réservé pour établir cette opération. En général ce champ se trouve dans la queue du message.

La dispersion géographique des modules défavorise la structure du bus parce qu'il est peu pratique de tirer plusieurs signaux de contrôle sur de grandes distances. Dans un anneau, l'accusé de réception est inclus dans le message, dont il passe par le même chemin que les données.

Nous avons adoptés la structure en anneau. Le système est parfaitement valable pour les modules localisés ou dispersés géographiquement.

IV.4.4 - Les niveaux du système

Le système est composé de deux niveaux (fig. 3) :

- le niveau le plus haut qui est celui de l'utilisateur,
- le niveau le plus bas dans lequel est implanté la communication entre les sorties-module et les entrées-module.

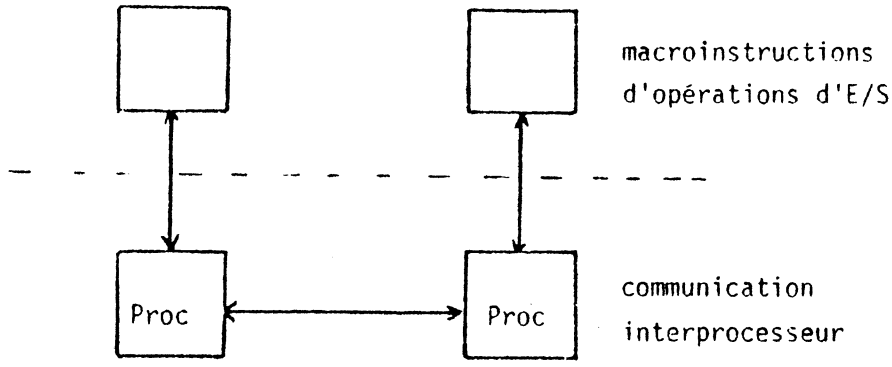



Fig. 3 : Les deux niveaux du système

IV.4.4.1 - Niveau utilisateur

Du point de vue de l'utilisateur, tout ce qui entre dans le module ou en sort est considéré comme une opération d'entrée ou de sortie. L'opération d'entrée acquiert alors une certaine autonomie et ne bloque plus l'exécution du programme du module. A l'endroit, où les résultats de cette lecture sont nécessaires, nous plaçons une macroinstruction d'attente d'un évènement ATTENDRE (E) où E est simplement l'identificateur de la donnée demandée par l'opération d'entrée.

Les opérations d'entrée sont demandées par la macroinstruction ENTRER(identificateurs des variables). Par contre, l'opération de sortie bloque le processus qui l'effectue jusqu'à ce qu'elle soit réalisée.

A l'endroit où un résultat est déjà disponible dans le module nous plaçons une macroinstruction SORTIR (E) où E est l'identificateur de la donnée à sortir.

exemple :

```
Module traitement-algorithmique
     Entrée-module  :  MESURES
     Sortie-module  :  COMMANDE

Début
     ENTRER MESURES

     ATTENDRE (MESURES)

          Calcul de la commande

     SORTIR (COMMANDE)

Fin
```

Remarque : Si la macroinstruction ENTRER (E) est placée directement avant la macroinstruction ATTENDRE (E), la donnée référencée par E n'est communiquée que lorsque le module producteur et le module consommateur seront bloqués respectivement derrière les macroinstructions SORTIR et ENTRER. C'est un des principes d'une autre approche de synchronisation [HOA 78].

IV.4.4.2 - Niveau communication interprocesseur

a) Registres de contrôle des données :

A chaque variable déclarée en entrée ou en sortie d'un module est associé un registre de quatre drapeaux (ou indicateurs) : $i_1$, $i_2$, $i_3$ et $i_4$. Ces indicateurs sont mis à "1" lors de l'initialisation du module.

Suivant le type de la variable (en entrée ou en sortie d'un module) les drapeaux ont des significations différentes :

- variable en entrée du module

$i_1 = 0$ actualisation autorisée. Le drapeau est mis à zéro par la macroinstruction ENTRER (E)

$i_2 = 0$ actualisation réalisée

$i_3 = 0$ erreur physique due à la transmission

$i_4 = 0$ demande d'actualisation effectuée par le module qui élabore la variable

- variable en sortie du module

$i_1 = 0$ variable accessible en lecture par d'autres modules. Le drapeau est mis à "0" par la macroinstruction SORTIR (E)

$i_2 = 0$ accusé de réception définitif : toutes les entrées dans les autres modules ont été actualisées

$i_3 = 0$ erreur physique de transmission

$i_4 = 0$ accusé de réception non définitif : au moins une entrée-module n'est pas actualisée parce que l'accès n'est pas encore autorisé

b) Structure et types des messages :

Le message est constitué : - d'un entête,
- d'un corps,
- et d'une queue.

(voir Annexe-2 pour les détails).

Il existe quatre types de messages :

- message à destinations multiples non spécifiées,
- message à destination unique,
- message de retransmission (ou de répétition),
- message de demande de retransmission (constitué d'un entête uniquement).

Le type du message est codé dans un champ spécial de l'entête.

Un autre champ dans la queue est consacré au résumé des "remarques" effectuées par les modules rencontrés par le message :

- erreur de transmission,
- erreur de spécifications,
- accusé de réception définitif,
- accusé de réception non définitif.

Les intéractions des codes opérations des messages avec les états des drapeaux sont données en détail en Annexe 2.

L'utilisateur engendre directement, par le moyen de la commande de sortie, un seul type de message : message à destinations multiples non spécifiées. Les messages à destination unique sont utilisés par le système pour les opérations de chargement des programmes et de mises au point. Les autres types de messages sont engendrés par les intéractions des drapeaux avec les codes opérations.

IV.5 - Interface de communication

Une interface microprogrammée assure pour chaque module :

- la communication physique des messages,
- le contrôle des erreurs de transmission,

- l'exécution de l'algorithme des intéractions des drapeaux avec les codes opérations des messages,
- un mécanisme d'interruptions vectorisées.

L'interface microprogrammée est pour le module le noyau qui assure "l'administration" de ses entrées et de ses sorties. Elle permet :

- d'éviter l'interruption du processeur pour vérifier chaque message émis dans le système de communication,

- d'améliorer le temps de réponse du système de communication, élément essentiel pour les systèmes temps réel. Une interface performante permet de s'affranchir de la notion de priorité des messages, technique souvent utilisée pour combler l'inertie du moyen de communication.

La structure de l'interface sera présentée au chapitre V.

# BIBLIOGRAPHIE

[PAR 72] D.L. PARNAS, "On the Criteria To Be Used in Decomposing Systems into Modules", Communications of the ACM - Décembre 1972, pp. 1053-1058

[GAU 70] R. GAUTHIER, S. PONT, "Designing Systems Programs", Prentice-Hall Englewood Cliffs - N.J., 1970

[SAM 76] R. SAMUEL, H.G. MENDELBAUM, F. MADAULE, "On Hardware/Software Distribution of Real Time Systems", Proc. of the EFAC/IFIP Workshop on Real Time Programming - Rocquencourt 1976 - pp. 191-201

[BER 70] C. BERGE, "Graphes et hypergraphes", Monographie Universitaires de Mathématiques - Dunod - 1970

[DEN 75] J.B. DENNIS, D.P. MISUNAS, "A Preliminary Data Flow Architecture for a Basic Data Flow Processor", Proc. Sec. Symposium Computer Architecture, 1975 - pp. 126-132

[HOA 78] C. HOARE, "Communicating Sequential Processes", Communications of the ACM - Août 1978 - pp. 666-677

[ANC 76] F. ANCEAU, P. DESCHIZEAUX, "Impact of Hardware Problems on Real Time Languages", Proc. of the 1976 IFAC/IFIP Workshop on "Real Time Programming" - Rocquencourt FRANCE - pp. 115-132

[SCH 78] J.D. SCHOEFFLER, "Software Architecture for Distributed Data Acquisition and Control Systems", Preprints of the Seventh Triennal World Congress of the IFAC - Juin 1978

[DES 75] P. DESCHIZEAUX, Z. BINDER, "Spécification de structures informatiques pour la conduite de procédés", Proceedings of the Intern. Symposium MIMI'75 - Zurich - Juin 1975 - pp. 167-171

[BRI 75] P. BRINCH HANSEN, "The Programming Language Concurrent PASCAL", I.E.E.E. Trans. on Software Engineering - vol SF-1 - n° 2 - pp. 199-207

# CHAPITRE V

# DESCRIPTION DU FONCTIONNEMENT GENERAL DU SYSTEME ET STRUCTURE MATERIELLE DE L'INTERFACE DE COMMUNICATION

# Description du fonctionnement général du système et de la structure matérielle de l'interface de communication.

## V.I - Description du fonctionnement général [OL1 78] [OL2 78] [OL3 78].

L'interface de communication est adressée par le processeur comme une zône mémoire. La mémoire associative de l'interface contient la liste des noms (ou étiquettes) des variables d'entrées/sorties du module. L'information complémentaire de la mémoire associative contient les longueurs de ces variables ainsi que leurs adresses d'implantation en mémoire. Ces informations sont chargées dans l'interface au départ (quoiqu'elles peuvent être changées dynamiquement). Ces variables constituent les variables visibles de l'extérieur du module. Les accès à ces variables en lecture ou en écriture sont contrôlés par les registres de contrôle de données qui sont couplés avec la mémoire associative. Par analogie avec les langages de programmation, la variable dont le nom est placée dans la mémoire associative est pour le module ce qu'est une variable déclarée globale pour un bloc de programme.

Chacune des variables visibles du système n'est déclarée comme variable de sortie qu'en un seul lieu. C'est une conséquence de la règle de l'assignement unique qui fait qu'une variable déterminée n'est modifiable que par un seul module.

Comme dans tous les systèmes en anneau, les messages émis vont circuler entre les différents modules à travers les queues FIFO câblées (fig. 1). On dispose au niveau de chaque interface d'une fenêtre d'observation par laquelle un module peut émettre ou recevoir des messages Le message émis est constitué d'un entête qui contient l'étiquette de la donnée qu'il enferme, le numéro du processeur source et un code opération spécifiant la nature de la transmission. Ce message va circuler en passant à tour de rôle sous toutes les fenêtres d'observations. Le module intéressé par l'étiquette recevra dans sa mémoire par accès direct à la mémoire (ADM) la donnée qui suit immédiatement l'entête. La donnée est reconnue valide pour le module quand l'étiquette du message coïncide avec l'une des étiquettes de la mémoire associative et lorsque le vecteur d'état correspondant autorise l'accès. S'il y a autorisation de transfert, le mécanisme d'ADM est initialisé par l'information associée de la M.A. qui

est l'adresse où doit être implantée la donnée dans la mémoire privée du processeur. En mode "broadcast" le message est effacé de l'anneau après un tour complet par son émetteur. Les "remarques" que puissent faire les modules telles que entrée non valide encore, erreur de transmission, sont notées dans la queue du message par modification de son code opération. L'interface source consulte la queue de son message après un tour et décide de l'action à entreprendre. Un message de demande de lecture est constitué d'un entête seulement. Le module qui a cette variable en sortie, répondra en plaçant derrière cet entête par ADM l'information demandée et en modifiera le code opération.

Pour envoyer un message le processeur dépose dans un registre de l'interface l'entête du message à émettre puis cède le contrôle.

L'interface interprète cet entête comme si c'était lui qui passait sous sa fenêtre d'observation, et agit comme si c'était une demande de lecture faite de l'extérieur. Ceci a permis de standariser le mécanisme des messages interface-interface et interface-processeur.

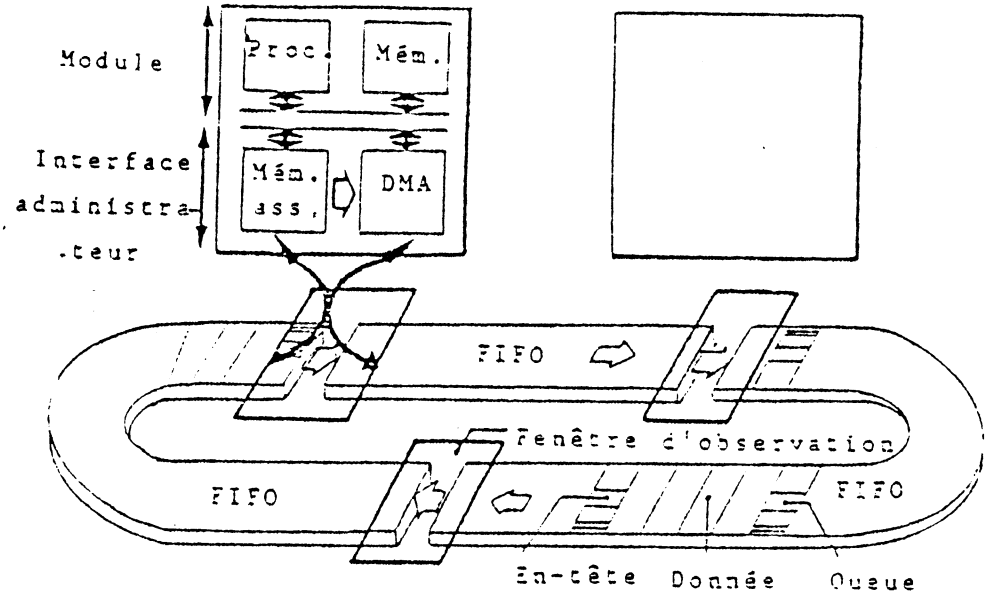



Fig. 1 : Schéma représentatif

V.2 - Structure matérielle de l'interface de communication

V.2.1 - Synoptique - fig. 2

C'est une interface microprogrammée constituée d'un chemin de données interne de 16 bits dont 8 bits sont utilisés seulement avec les

microprocesseurs 8 bits. L'interface se connecte sur le bus du processeur et elle est adressée comme une zône mémoire.

Le cycle machine (de 1 MHz) est constitué de quatre phases construites à partir de l'horloge du microprocesseur.

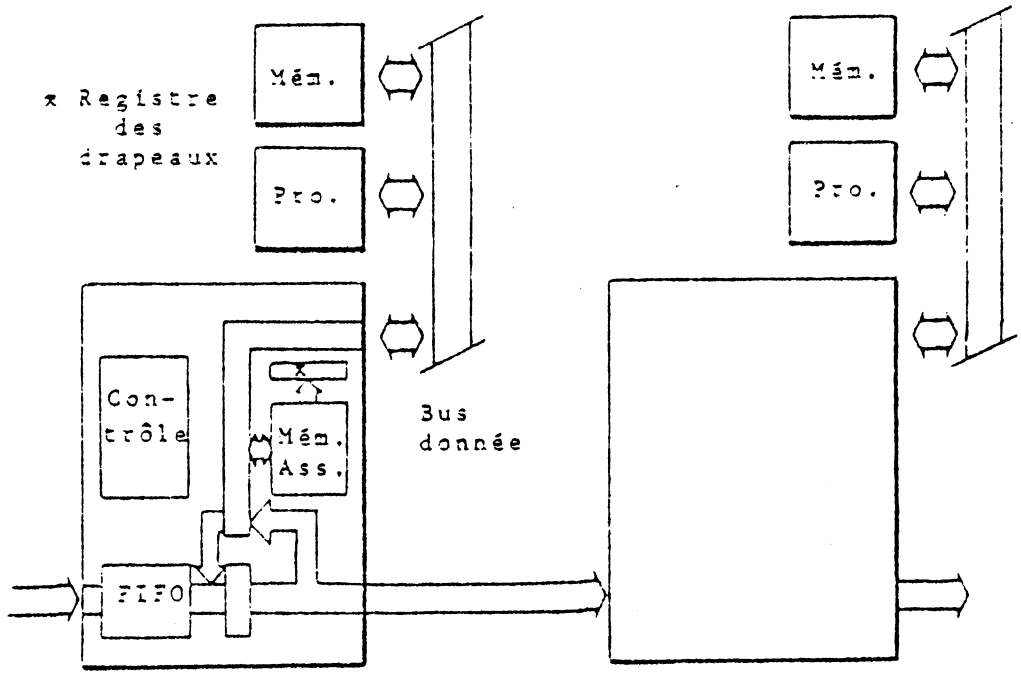



Fig. 2 : Chemin des données

Les constituants essentiels de l'interface sont :

- une queue First In First Out (FIFO) réalisée par des circuits intégrés spécialisés,
- une mémoire adressable par le contenu implémentée par deux couches de mémoire vive (RAM),
- un ensemble de registres de contrôle des données implémentée aussi par des $RAM_S$,
- un mécanisme d'accès direct à la mémoire,
- une unité de contrôle microprogrammée.

Pour avoir accès aux ressources de l'interface, le processeur positionne à "0" le drapeau de requisition, dans le mot de commande et attend le positionnement d'un drapeau dans le mot d'état de l'interface pour avoir le contrôle. L'interface cède le contrôle dès qu'elle termine

le message en main. La libération de l'interface s'effectue en mettant REQ à 1.

- Mot de commande :
  - . adresse 80ED
  - . $b_0$ = drapeau REQ
  - . $b_1$ = drapeau AINT. Ce drapeau est mis à "1" par une primitive du logiciel pour acquiter l'interruption générée par l'interface.

- Mot d'état :
  - . adresse 80EE
  - . $b_0$ = 0 : message local pas encore émis
  - . $b_1$ = 0 : accès interdits aux ressources de l'interface
  - . $b_2$ = 0 : une interruption est générée par l'interface.

V.2.2 - Queue First In First Out (FIFO) intégrée

Le rôle de cette queue est la tamporisation des messages en provenance de l'interface en amont. Sa profondeur est de 128 mots (18 bits de largeur) et elle est implémentée par des circuits intégrés spécialisés (AM 2841 A)[AMD 76]. Cette queue est constituée d'un ensemble de registres à travers desquels le mot entré par le sommet de la queue se propage automatiquement vers le fond pour se placer juste derrière les derniers mots non sortis de la queue.

Ce dispositif matériel constitue une solution idéale pour résoudre le problème classique du producteur-consommateur implémenté par des pointeurs, de sémaphores et d'une mémoire vive. Un avantage propre au circuit intégré utilisé est son asynchronisme entre l'entrée d'un mot et la sortie d'un autre : producteur et consommateur peuvent fonctionner avec des horloges différentes.

L'inconvénient des queues intégrées est de ne pas pouvoir disposer d'assez de profondeur sans coût excessif. Le protocole d'insertion des messages dans l'anneau adopté nous a permis de n'utiliser que des FIFO, de profondeur limitée (voir chapitre suivant).

V.2.3 - Mémoire adressable par le contenu (simulée par des RAMs)

C'est une mémoire dans laquelle une position mémoire est sélectionnée non pas par son adresse mais d'après son contenu. Une fois sélectionnée, la position mémoire pointe vers une information associée (ou complémentaire). C'est une technique de recherche en table plus rapide que la technique programmée. Les étiquettes des variables d'entrées ou de sorties du module sont listées dans une telle mémoire. Les longueurs de ces variables ainsi que leurs adresses d'implantation dans la mémoire privée formeront les informations associées.

L'idée d'implémenter cette mémoire avec des circuits intégrés spécialisés nous a révélé la nécessité d'un nombre assez grand de composants même pour n'avoir que des tables de taille réduite (48 composants pour une table de 32 étiquettes).

Nous nous sommes tournés vers une solution plus "compacte" qui assure les mêmes fonctions avec moins de circuits.

Solution réalisée

Elle consiste à utiliser des RAMs et de les adresser par les valeurs binaires des étiquettes (fig. 3).

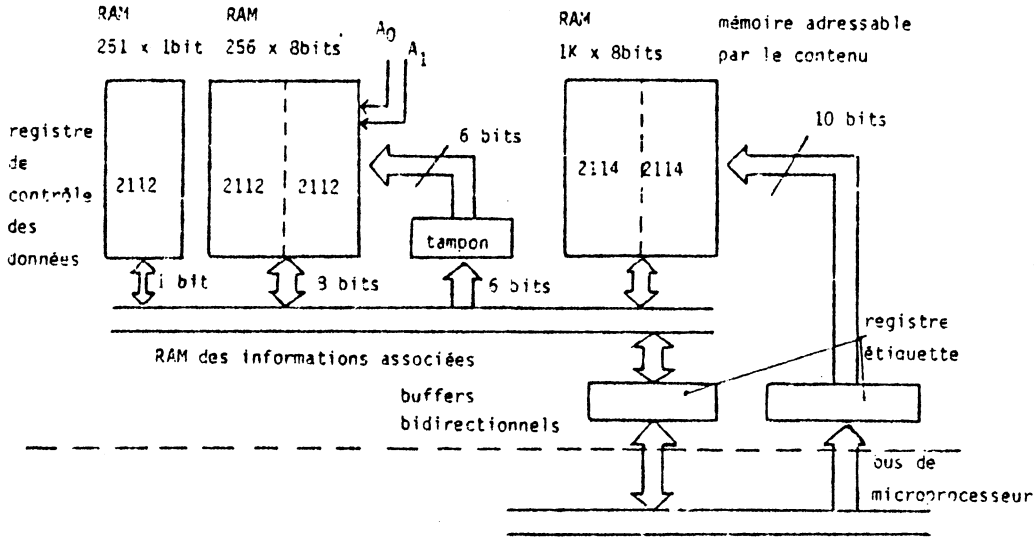



Fig. 3 : Implémentation de la mémoire adressable par le contenu avec des RAMs

Les dix bits de l'étiquette sont utilisés comme 10 bits d'adresses pour la RAM de 1K x 8 bits (2 x 2114 Intel). Nous inscrivons dans cette position mémoire un numéro d'étiquette (sur 6 bits) qui sera tamporisé pour devenir une adresse dans la RAM réservée à l'information associée.

La définition d'une étiquette est faite comme suit :

a) Dépôt de l'étiquette dans "registre étiquette"
   . adresse 80F1 et 80F2

b) Dépôt du numéro de l'étiquette
   . adresse 80FF
   . $b_5$ - $b_0$ : numéro étiquette
   . $b_7$ - $b_6$ : sont utilisés pour spécifier la nature de l'étiquette

| $b_7$ | $b_6$ | |
|---|---|---|
| 1 | 0 | étiquette de sortie |
| 0 | 1 | étiquette d'entrée |
| 0 | 0 | étiquette d'entrée + interruption |
| 1 | 1 | étiquette non définie |

c) chargement de l'information associée
   - dépôt numéro routine (8bits) si l'étiquette est définie en entrée plus interruption
     . adresse 80F8
   - dépôt longueur de la variable
     . adresse 80F9
   - dépôt adresse implantation en mémoire
     . adresse 80FA pour octet poids fort
     . adresse 80FB pour octet poids faible.

Il est possible de définir jusqu'à 64 étiquettes en même temps. La liste des étiquettes peut être modifiée dynamiquement par le processeur.

V.2.4 - Registres de contrôle des données

Leur rôle est de contrôler les accès aux variables déclarées en entrée ou en sortie des modules. Les registres sont implémentés dans une RAM de 256 x 1 bits (4 drapeaux x 64 étiquettes). Le vecteur d'état est adressé en même temps que l'information associée (fig. 3).

Les séquences de consultations ou de modifications des drapeaux suivent le schéma suivant :

- Dépôt étiquette dans "registre étiquette"
  . adresses 80F1 et 80F2

- lecture de la mémoire adressable par le contenu. Par ex :
  LDA 80FF
  Cette instruction a pour rôle de faire sortir le numéro étiquette de la MAC sur le bus interne pour qu'il soit mémorisé dans le "tampon" qui sert de registre d'adresse pour l'information associée

- le vecteur d'état de l'étiquette est maintenant disponible.
  . Drapeau DIS adresse 80F4
  . Drapeau RT adresse 80F5
  . Drapeau ACC adresse 80F6
  . Drapeau PRT adresse 80F7

V.2.5 - Mécanisme d'accès direct à la mémoire (ADM)

C'est un mécanisme de transfert qui permet d'introduire ou d'extraire de l'information de la mémoire sans passer par le processeur. Le fonctionnement du processeur est suspendu pendant un laps de temps mais non interrompu (pas de changement de contexte). Ce mécanisme présente deux avantages majeurs :

- vitesse de transfert d'information plus grande que celle du mode programmé . Les retards au niveau des interfaces sont réduits.

- évite au processeur servi de dupliquer inutilement l'information qu'il veut émettre dans des tampons de l'interface.

Le mécanisme d'accès direct à la mémoire est initialisé par le contenu des informations associées de la mémoire adressable par le contenu (fig. 4).

Fonctionnement :

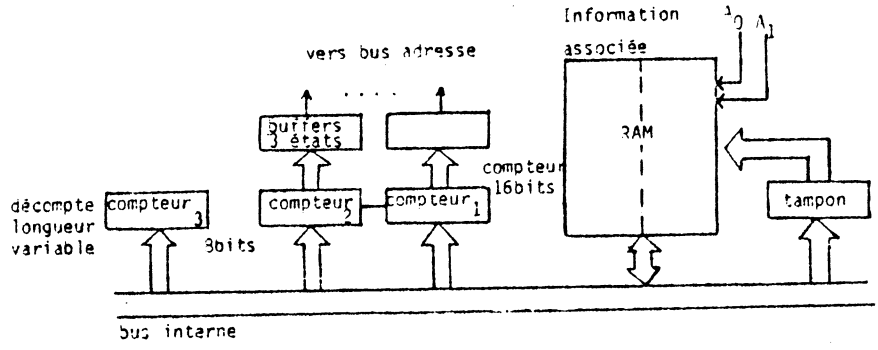



Fig. 4 : Initialisation du mécanisme de l'ADM

L'unité de contrôle de l'interface envoit l'étiquette de l'entête du message qu'elle vient d'analyser dans un registre 10 bits (connectés en 3 états avec le registre étiquette précédent) qui sert de registre d'adresse pour la mémoire adressable par le contenu. Cette mémoire délivre le n° étiquette qui sera mémorisé dans tampon. L'information associée à l'étiquette devenant disponible, les compteurs 1, 2 et 3 sont chargés en 3 cycles machines.

Le compteur 3 permet de décompter jusqu'à 255 mots pour un message. Cette valeur peut paraître limitée, mais elle est suffisante pour notre application. Pour des valeurs plus grandes que 255 mots, il faut définir dans l'unité de contrôle que la longueur du message est un multiple de 2 ou 4 de la longueur déclarée. La représentation en virgule flottante des valeurs numériques (4 octets) facilite cette technique.

V.2.6 - Unité de contrôle

V.2.6.1 - Rappel sur la microprogrammation

L'idée de la programmation est normalement attribué à Wilkes. Il a défini l'unité de contrôle microprogrammée comme composée de deux parties (fig. 5) :

- une première partie qui contrôle le chemin des données à l'aide de microopérations générées par des microinstructions : (ouvertures de portes; positionnement de bascules, ...)

- et une deuxième partie qui contrôle la sélection de la commande de l'étape suivante dite adresse de retour.

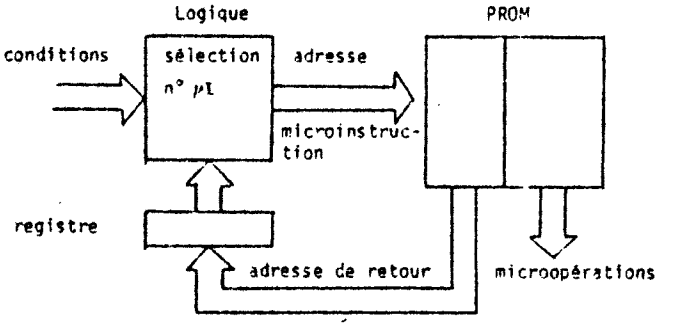



Fig. 5 : Le Modèle microprogrammé de Wilkes

La sélection de la microinsturction suivante à exécuter est faite en fonction des conditions externes et de l'adresse de retour.

A partir de ce principe, une grande variété de structures microprogrammées ont été proposées ou réalisées. Pour une revue sommaire de ces structures voir [NIK 78].

V.2.6.2 - Structure de contrôle adoptée

Un moyen pour réduire et simplifier la logique de la partie de sélection de la microinstruction suivante consiste à ne tenir en compte dans chaque étape que des signaux qui peuvent influencer sur la décision à prendre. Il est peu fréquent que la sélection d'une microinstruction puisse dépendre d'un grand nombre de signaux externes dans la même étape

(ou cycle machine). La technique consiste à utiliser des multiplexeurs qui sélectionnent, à chaque étape (ou cycle machine) $n_1$ signaux externes (parmi n) considérés utiles pour la décision à prendre. Le problème revient à étudier les $2^{n1}$ branchements possibles à partir de la microinstruction courante au lieu des $2^n$ branchements. Les $2^n - 2^{n1}$ branchements ne contribuant pas dans la logique de décision sont ainsi "court-circuités".

La figure 6 montre la structure d'une telle unité de contrôle.

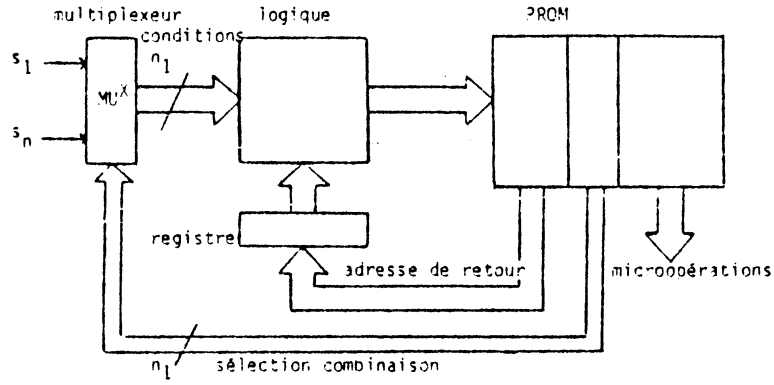



Fig. 6 : La structure de l'unité de contrôle de l'interface

A partir de l'ensemble X des signaux externes de cardinalité n nous construisons une famille de partitions de cardinalité $2^{n'}$ dont chaque partition est de cardinalité $n_1$.

Démarche de construction

1) L'algorithme de l'unité de contrôle est mis sous forme d'une table composite des transitions et des sorties avec $n_1$ entrées. Une ligne représente un état avec $2^{n1}$ possibilités de branchements.

A l'intersection d'une ligne et d'une colonne nous inscrivons :

- l'état suivant,
- le numéro de partition qui contient les signaux utiles pour la décision à prendre dans l'état suivant,
- le numéro de la microinstruction à exécuter immédiatement.

La table peut être incomplètement spécifiée : certaines combinaisons des entrées ne se réalisent jamais pour un état donné. Par exemple

la partition de l'état présent contient un signal que renfermait la partition de l'état précédent et dont on connait que le signal ne change pas entre ces deux états.

Prenons l'exemple suivant

Soient $X = s_1, s_2, s_3$ l'ensemble des signaux externes, et $f_1 = s_1, s_2$ et $P_2 = \{s_1, s_3\}$ les partitions (fig. 7). Supposons que $s_1$ ne change pas à partir de l'état initial.

| | | | | | P1 | P2 |
|---|---|---|---|---|---|---|
| Condition / état | 0 | 0 | 1 | 1 | $s_1$ | $s_1$ |
| | 0 | 1 | 0 | 1 | $s_2$ | $s_3$ |
| état initial | $0,1,\mu I_3$ | - | $b,2,\mu I_2$ | $b,2,\mu I_3$ | (P1) | |
| a | $b,2,\mu I_1$ | $b,2,\mu I_1$ | - | - | (P1) | |
| b | - | - | $EI,1,\mu I_1$ | $EI,1,\mu I_2$ | (P2) | |

Fig. 7 : Table composite de l'exemple

Le rond à droite de chaque ligne indique dans quelle partition nous sommes.

Puisque $s_1$ ne change pas à partir de l'état initial, les deux états a et b son incomplètement spécifiées :

- dans l'état a) les combinaisons 10 et 11 sont interdites. L'état a) a été atteint de l'état initial parce que $s_1$ était à zéro et restait depuis inchangé (par hypothèse)

- même raisonnement pour l'état b).

2) Nous procédons après à la minimisation du nombre des états internes par les méthodes conventionnelles de minimisation [CHI 67] d'une table incomplètement spécifiée. Le n° partition est considérée comme uhe sortie.

Avec l'exemple précédent nous obtenons :

| condition C1 / C2 / états | 0 / 0 | 0 / 1 | 1 / 0 | 1 / 1 | P1 : $s_1$ | P2 : $s_1$ |
|---|---|---|---|---|---|---|
| | | | | | $s_2$ | $s_3$ |
| A | A,1,$\mu I_3$ | - | B,2,$\mu I_2$ | B,2,$\mu I_3$ | A = EI | |
| B | B,2,$\mu I_1$ | B,2,$\mu I_1$ | A,1,$\mu I_1$ | A,1,$\mu I_2$ | B = a, b | |

Fig. 8 : Table après minimisation des états internes

3) Implantation sur des $PROM_S$ (Programmable Read Only Memory). Les états A et B sont considérés comme des pages et les conditions C1 C2 comme des lignes dans ces pages. L'état futur mentionné dans chaque case constitue l'adresse de retour.

L'adresse de retour concaténée avec la condition formeront une adresse dans la PROM où sont inscrites toutes les informations nécessaires pour chaque état.

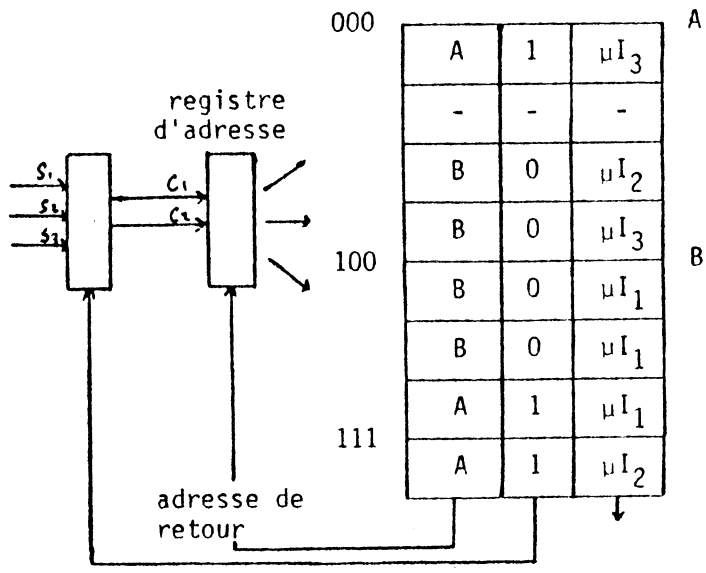
Avec l'exemple précédent nous obtenons l'unité de contrôle suivante :

codage :

Fig. 9 : Structure finale de l'unité de contrôle de l'exemple

A = page zéro $\quad P_1 = 1$

B = page 1 $\quad P_2 = 0$



L'unité de contrôle de l'interface comporte :

- 24 états internes (64 avant l'optimisation effectuée manuellement)
- 8 partitions (n' = 3) chacune de cardinalité 4 ($n_1$ = 4)
- 19 signaux externes (n = 19)
- 32 microinstructions différentes.

Les microinstructions sont implantées dans des $PROM_S$ 74S288 (32 x 24 bits). Les adresses de retour et les n° microinstructions sont implantées dans deux $EPROM_S$ 2708..

Les microinstructions sont codées par champs ( programmation verticale) et sont quadriphases (microopérations réparties sur les quatre tops d'horloge sans recouvrement qui constituent le cycle machine de l'interface).

V.2.7 - Envoi d'un message

Pour envoyer un message le processeur dépose dans le registre MES l'entête du message à envoyer (fig.10)

. adresses 80E5 et 80E6

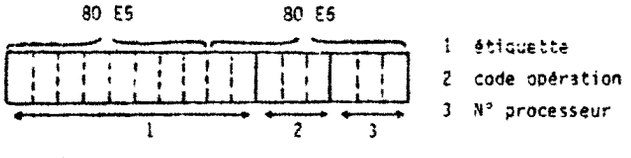



Fig.10 : Registre message

Un entête avec un code opération invalide génère une interruption au processeur.

V.2.8 - Codage des interruptions générées par l'interface

Les types des interruptions sont codés sur les 2 bits, $b_3$ et $b_4$, du mot d'état (adresse 80ED)

| $b_4$ | $b_3$ | |
|---|---|---|
| 0 | 1 | interruption générée par une entrée définie avec interruption |
| 1 | 0 | code opération invalide dans l'entête du message local à émettre |
| 1 | 1 | Message Erreur. L'information envoyée n'a pas les mêmes spécifications dans les autres modules. |

V.2.9 - Travaux en cours

Actuellement trois interfaces ont été réalisées ( 95 circuits MSI chacune) par la technique des connexions enroulées (wrapping). Pour faciliter la duplication, les interfaces seront réalisées par la technique des circuits hybrides à couches épaisses avec la collaboration de l'Ecole Nationale Supérieure d'Electronique et de Radioélectricité de Grenoble (ENSERG). Les travaux actuels sont au stade des dessins des masques des modules hybrides.

Un module hybride sérialisateur-désérialisateur sera associé à chaque interface pour assurer la communication série, asynchrone et différentielle (norme RS-432) sur ligne bifilaire adaptée, entre modules dispersés géographiquement. La vitesse de transmission visée est de l'ordre de 400 Kbauds/s.

# BIBLIOGRAPHIE

[AMD 76] "Memory Data Book" - Notices techniques - 1976

[INT 76] "Memory Data Book" - Notices techniques - 1976

[NS 76] "TTL Data Book" - Notices techniques - 1976

[NIK 78] NIKITAS A. ALEXANDRIDIS, "Bit-Sliced Microprocessor Architecture" - I.E.E.E. Computer - Juin 1978 - pp. 56-80

[CHI 67] J. CHINAL, "Techniques booléennes et calculateurs arithmétiques" Dunod - 1967 - pp. 508

[OL1 78] Z. OLAIWAN, "Centre multiprocesseur de traitement automatique" - Journée AFCET, Groupe temps réel - Février 1978

[OL2 78] Z. OLAIWAN, Z. BINDER, "Algorithme Processing Unit for Automatic Control" - Proc. 1978 International Conference on Parallel Processing - I.E.E.E. - Michigan USA - Août 1978

[OL3 78] Z. OLAIWAN, Z. BINDER, "Centre multiprocesseur pour traitement en Automatique" - Congrès Informatique 78 - AFCET - Paris - Novembre 1978

# CHAPITRE VI

# PROTOCOLES D'ENVOI DES MESSAGES

# Protocole d'envoi des messages dans l'anneau

## VI.1 - Introduction

Le chemin de communication interprocesseur constitue une ressource partagée entre plusieurs utilisateurs qui sont les interfaces. Les protocoles d'accès au chemin de communication dans l'anneau doivent répondre à deux questions importantes :

1) Comment insérer un message dans l'anneau sans détruire l'information déjà présente ?

2) Comment éviter l'autoblocage du système de communication ?
Cette situation peur arriver si toutes les interfaces veulent émettre la suite de leurs messages alors qu'il n'y a plus de places disponibles dans les tampons de l'anneau.

Pour préserver la modularité logicielle et matérielle du système, nous n'étudierons que les approches décentralisées.

## VI.2 - Les approches décentralisées les plus courantes

Il existe trois approches courantes pour répondre au deux questions précédentes.

- approche "plateau tournant" (Lazy susan)
- approche du "Pion baladeur" (roundrobin token)
- approche "insertion par registre à décalage".

### VI.2.1 - Approche "plateau tournant"

Elle consiste à fixer la longueur des messages et de laisser circuler dans l'anneau un nombre fixe de blocs (wagons) de tailles égales. Ces blocs circulent continuellement dans l'anneau avec ou sans message. L'interface qui veut émettre un message attend le passage d'un bloc libre dans lequel elle dépose son message. Le bloc est marqué par un code approprié pour avertir les autres interfaces qu'il est devenu occupé. Il est examiné par les interfaces pour déterminer s'il est adressé à eux ou pas. Si oui, le contenu du bloc est récupéré par l'interface adressée qui libère le bloc de nouveau.

Cette technique est dite aussi anneau de Pierce [PIE 72].

L'intérêt de cette approche est sa grande simplicité de mise en oeuvre. Il existe cependant plusieurs inconvénients :

- Pour maximiser l'efficacité de la ligne de transmission, la taille unitaire du bloc est rendue très petite ( 40 octets). Des techniques de décomposition et de recomposition des messages plus longs sont indispensables.

- Pour les messages très courts, le bloc est sous utilisé. Beaucoup d'espaces sont perdus dans l'anneau.

L'exemple le plus célèbre de cette approche est celui du DCS [FAR 73].

VI.2.2 - Approche du "Pion baladeur"

Elle consiste à ne laisser émettre qu'une seule interface à la fois. Un signal ou un pion est passé tour à tour d'une interface à une autre. L'interface qui veut émettre un message attend que ce pion soit à sa disposition. Une fois qu'elle le possède, elle l'efface puis envoit son message suivi d'un nouveau pion. Cette approche permet d'émission des messages de tailles variables. Elle est appelée aussi anneau de NEWHALL [NEW 72].

C'est une approche simple à implémenter mais présente deux inconvénients :

- manque de simultanéité dans l'émission des messages,
- la perte du pion (par défaut de transmission) bloque le système de communication. Un mécanisme de regénération du pion est nécessaire.

VI.2.3 - Approche d'insertion par registre à décalage

Le but principal de cette approche consiste à tirer partie des avantages des deux approches précédentes, c'est-à-dire :

- la simultanéité d'émission des messages,
- longueurs de messages variables mais limitées quand même à une valeur maximale.

Comme dans tout système de communication, l'interface doit accepter deux flots de messages :

- le flot à relayer qui provient de l'interface en amont,
- le flot à transmettre qui provient du processeur servi.

Dans cette approche, les conflits qui naissent des arrivées simultanées de messages des deux flots, sont résolus de la manière suivante [REA 75] (fig. 1)

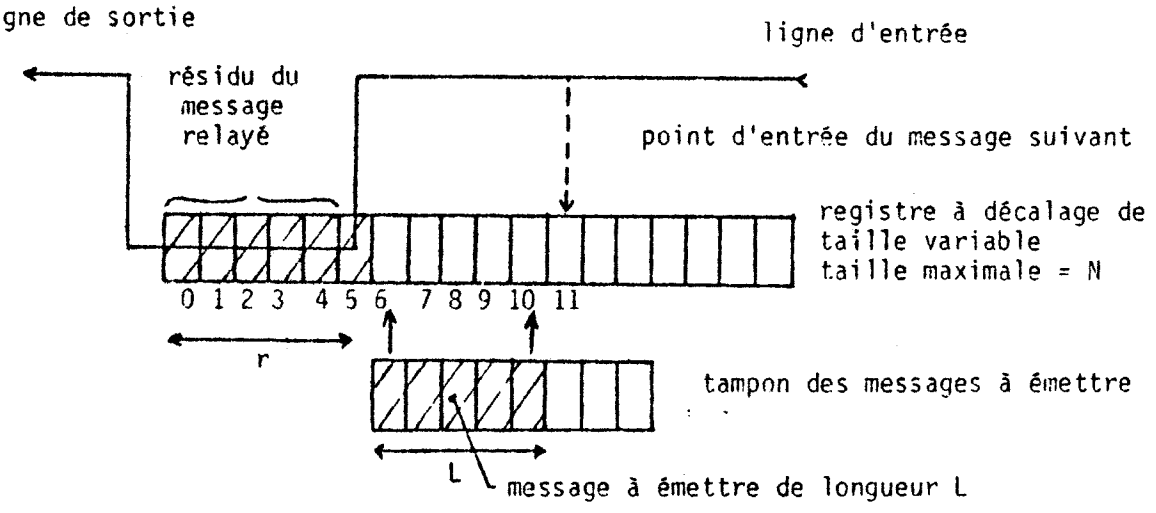



Fig. 1 : schéma de la technique d'insertion par registre à décalage.

Supposons que l'interface est en train de relayer un message. Soit r le résidu du message à un instant donné et L la longueur du message à émettre. L'interface ne peut prendre l'initiative pour émettre son propre message que juste au moment où elle reçoit la fin du message qu'elle relaye. A cet instant précis, l'interface compare la longueur L de son message avec l'espace disponible N-r dans le registre à décalage si $L \leq N-r$ l'interface envoit son message du registre tampon dans le registre à décalage, et pointe toute nouvelle entrée à partir de la position L+r. Si $L > N-r$, l'interface attend qu'il y ait suffisamment d'espaces pour contenir son message sans refuser de relayer d'autres messages.

L'étude en simulation de cette approche [REA 76] a montré qu'elle était plus souple et plus performante que les deux approches précédentes. Cependant sa mise en oeuvre est plus compliquée.

## VI.3 - Approche proposée

### VI.3.1 - Considérations de conception

Nous avons tenu compte de plusieurs considérations dans la définition du protocole de communication. Elles sont :

1)Le temps mort de transition au niveau de chaque interface doit être réduit au minimum. Ce temps mort dépend essentiellement de la quantité d'information tamporisée au niveau de chaque interface.

2) La réduction de la quantité d'information tamporisée permet d'améliorer le temps de réponse du système de communication.

3) Toutes les interfaces doivent pouvoir avoir accès à l'anneau autant que possible. Aucune interface n'est autorisée de dominer le flot dans l'anneau.

4) Avoir la possibilité d'émettre des messages de longueurs variables.

5) Simultanéité dans l'émission des messages pour conserver l'aspect décentralisé du système de communication.

### VI.3.2 - Similitudes avec la technique d'insertion par registre à décalage

La technique adoptée pour insérer des messages dans les queues FIFO (First In First Out) présentent beaucoup de similitudes avec celle de l'insertion par registre à décalage. Les similitudes portent sur les points suivants :

- au tampon du message de sortie correspond la zône mémoire où, est implantée la donnée à émettre puisque le mécanisme d'accès direct à la mémoire est utilisé pour le transfert d'information.

- au registre à décalage correspond la queue FIFO intégrée en aval c'est-à-dire celle de l'interface suivante. En négligeant le temps de propagation des mots dans la queue FIFO (8 µs pour une profondeur de 64 mots pour l'AM 2841 A), la queue se comporte exactement comme un registre à décalage de taille variable. Un avantage : montage nettement plus simple (suppression des décodeurs et des compteurs).

Avec la queue FIFO en circuit intégré nous n'avons que des informations très élémentaires sur son taux de remplissage : elle est pleine ou vide et sur certaine à moitié pleine .

Il faut considérer alors le cas le plus défavorable pour l'émission d'un message. Il doit être pris comme s'il a toujours la longueur maximale L.

VI.3.3 - Protocole implémenté pour modules localisés géographiquement

Soient L la longueur maximale fixée pour des messages et L+1 la profondeur de la queue FIFO de chaque interface. Le protocole est :

a) chaque interface ne peut émettre son message que si la queue FIFO qui la suit est totalement vide.

b) Le transit d'un message d'une FIFO à la suivante est permis tant qu'il y a de positions libres dans cette dernière.

c) Devant une fenêtre d'observation, le transit d'un message vers la FIFO suivante est prioritaire sur l'émission du message local.

Discussions :

La règle a) du protocole suppose l'existence d'un signal $V_1$ qui part d'une interface en aval vers une autre en amont (fig. 2) pour donner l'état de la FIFO en aval : vide ou pas. La règle b) suppose l'existence d'un deuxième signal $V_2$, dans le même sens que $V_1$, qui renseigne s'il y a encore de positions libres ou pas dans la FIFO en aval.

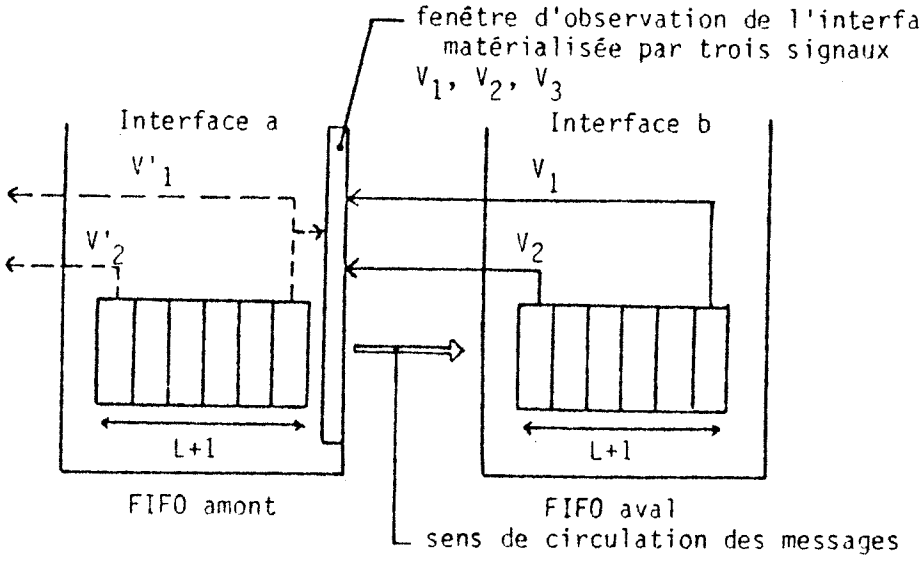



Fig. 2 : Les signaux de coordinations entre deux interfaces successives

Rem 1 : une position correspond à un mot

Rem 2 : les signaux $V_1$ et $V_2$ correspondent physiquement aux signaux OUTPUT Ready et INPUT Ready [AMD 76]

Ce protocole évite l'autoblocage du système de communication parce que l'interface s'assure qu'il y a assez de positions libres pour contenir tout son message.

## VI.4 - Etudes de cas particuliers

### VI.4.1 - Toutes les interfaces émettent simultanément des messages à destinations multiples

Ce cas ne se produit que lorsque l'anneau est totalement vide avant l'émission simultanée. S'il y a un seul message dans l'anneau, il y a au moins une interface occupée par son relayage ou par sa récupération.

#### Calcul du débit moyen théorique

Soit L la longueur maximale des messages

Soit L+1 la longueur de chacune des FIFO dans les interfaces.

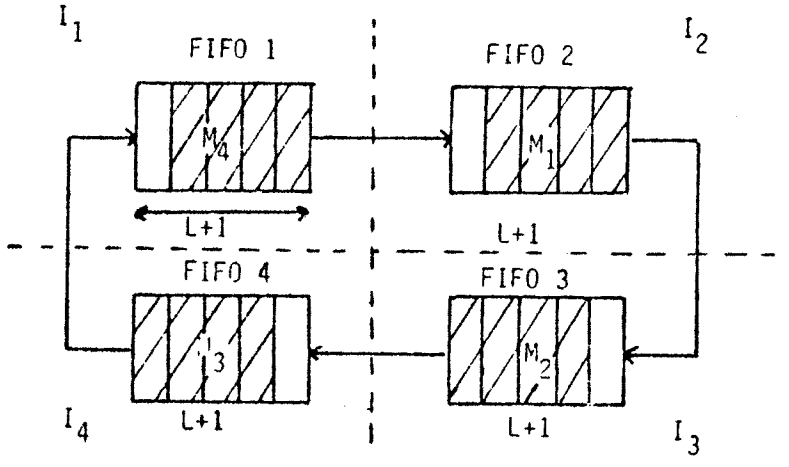



Fig. 3 : Cas particulier ou toutes les interfaces émettent simultanément (anneau vide au départ)

Soit $\tau$ le temps maximal du retard au niveau de chaque interface. Ce temps comprend le temps nécessaire pour décoder l'entête du message plus le temps d'une requisition de l'interface par son module (test d'un drapeau chargement étiquette, ...).
Soit $D_{max}$ le débit maximal de la ligne entre deux interface successives, exprimé en bit ou mot (suivant L)/s.
Soit $t_o$ l'instant initial quand toutes les interfaces émettent à la fois.

A l'instant $t_o - \Delta t$ toutes les FIFOs sont vides, donc toutes les conditions du protocole d'émission de messages sont favorables pour toutes les interfaces.

A $t_o$ , toutes les interfaces émettent ensembles.

A $t_o + \frac{L}{D_{max}}$ l'état des FIFOs est montré par la fig. 3 : il reste une position libre dans chaque FIFO (l'anneau est saturé).

D'après la condition b) du protocole, le passage d'un mot d'une interface à une autre est permis. Chaque message se déplace donc d'une position à chaque "top". Il faut donc (n-1) (L+1) tops pour chaque message pour faire un tour complet et L tops pour qu'il soit balayé par son interface source. Le temps total $t_r$ écoulé est :

$$n\tau + \frac{L}{D_{max}} + \frac{(n-1)(L+1)}{D_{max}} + \frac{L}{D_{max}} = \frac{1}{D_{max}} [L(n+1)+(n-1)] + n\tau$$

Pendant ce temps les n-1 autres messages ont fait exactement pareil. Le nombre total de mots écoulés dans l'anneau pendant ce temps est n.L .

Le débit moyen $D_M$ est donc :

$$D_M = \frac{nL}{\frac{1}{D_{max}} [L(n+1)+(n-1)] + n\tau}$$

en mojorant $\frac{n+1}{n}$ par 1,3 ($n \geqslant 3$) et en négligeant le terme $\frac{n-1}{nL} \approx 0$ ($L \gg 10$) nous avons approximativement :

$$D_M \simeq \frac{D_{max}}{1,3 + \frac{\tau D_{max}}{L}}$$

Le temps tr peut être considéré comme le temps de réponse du système de communication puisque après ce temps l'interface est capable d'émettre un nouveau message.

$$\text{Approximativement} \quad tr = \frac{nL}{D_{max}} [1,3 + \frac{D_{max}}{L}]$$

pour

L = 100 octets
$\tau = 5.10^{-4}$s
$D_{max} = 5.10^5$ mots/s (parfaitement possible par DMA)
pour n = 8 processeurs, nous avons :

$D_m \simeq 160$ Kmot/s $\qquad$ $tr \simeq 5$ ms

VI.4.2 - Les interfaces émettent des messages à tour de rôle

Soit n-1 interfaces relayant en pipeline le même message. Ceci arrive à deux conditions :

- le message a été mis au départ dans un anneau complètement vide,
- la somme des retards $s_r$ que le message subie au niveau des n-1 interfaces est inférieure ou égale au temps d'émission du message c'est-à-dire $\frac{L}{D_{max}}$.

Dans le cas $s_r < \frac{L}{D_{max}}$ l'entête du message retourne à l'interface source avant que celle-ci ait eu le temps de finir l'émission de son message en totalité (fig. 4).

Supposons qu'à partir de ce moment les n-1 interfaces ($I_i$ , $i > 1$) veulent émettre des messages. L'interface $I_2$ aura la première occasion puisque l'interface en aval $I_1$ sera occupée un moment pour retirer son message à l'anneau. La deuxième occasion sera pour $I_3$ et ainsi de suite.

VI.4.3 - Remarques

La première remarque que nous pouvons tirer est qu'à forte charge l'anneau se comporte comme un anneau de Pierce d'autant plus que le retard est petit au niveau de chaque interface. Par contre plus $\tau$ est grand, plus l'anneau se comporte comme un anneau de Newhall qui autorise l'émission simultanée de plusieurs messages. Le débit moyen du système de communication qui est pénalisé par les retards , est compensé par l'émission simultanée de plusieurs messages. Il y a régulation automatique du traffic.

La deuxième remarque est qu'aucune interface est capable de dominer l'anneau tant qu'une interface au moins veut émettre un message. Chaque interface est contrainte à dépenser un certain temps pour retirer son dernier message. D'où l'occasion pour l'interface en aval d'émettre.

Pour éviter l'autoblocage du système de communication, la longueur maximale d'un message ne doit pas dépasser théoriquement la profondeur L de la FIFO de chaque interface. En pratique et l'expérience le prouve, il est toujours possible d'émettre des messages plus longs. Il est rare que toutes les interfaces émettent simultanément dans un temps d'une dizaine de microsecondes, temps de propagation d'un mot dans la FIFO.

VI.5 - Protocole proposé pour modules dispersés géographiquement

Le protocole précédent était basé sur deux signaux de coordination :

- signal $V_1$ qui indique que la FIFO en aval est vide
- signal $V_2$ qui indique qu'une position au moins est libre dans la FIFO en aval.

Ces deux signaux nécessitent deux câbles pour les acheminer d'une interface à une autre. Le protocole est inadéquat pour les modules dispersés..

Pour réduire le nombre des signaux de coordination, il faut considérer maintenant que la position dans une FIFO correspond à un message complet de longueur L mots et non pas à un mot unique.

En considérant la FIFO de l'interface de longueur 2L + 1 mots le protocole devient :

a) l'émission ou le transit d'un message est autorisé si la FIFO en aval est au moins à moitié vide,

b) Devant la fenêtre d'observation, le transit d'un message vers la FIFO suivante est prioritaire sur l'émission du message local (même règle qu'avant).

Nous avons besoin cette fois-ci d'un seul signal de coordination $V_3$ (fig. )

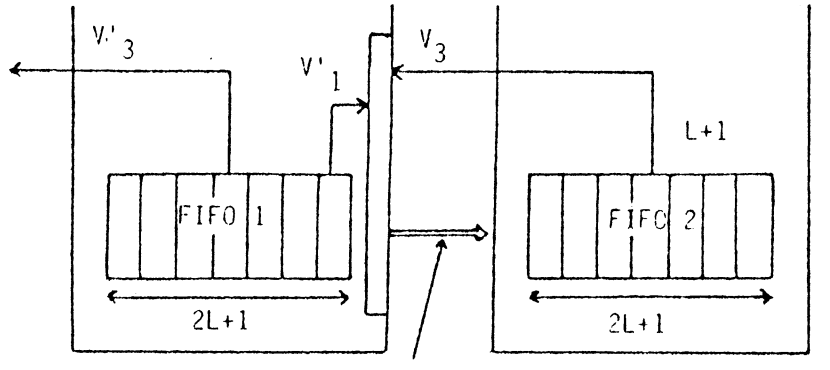



Fig. 4 : Prélèvement du signal de coordination $V_3$ pour modules dispersés.

Les propriétés de ce protocole sont comparables à celles du précédent sauf que pout la même profondeur de FIFO, il faut considérer des messages de moitié plus courts.

Cas particulier

Le retard $\tau$ que subi le message au niveau d'une interface peut provenir :

- soit de l'émission d'un message local. Ce retard est de $\frac{L}{D_{max}}$ dans le cas le plus défavorable.

- soit d'une requisition de l'interface par le processeur pour faire une modification. Soit $\tau_0$ ce retard.

- soit des deux causes précédentes arrivant l'une après l'autre.

Le retard $\tau$ s'exprime dans le cas le plus défavorable par :

$$\tau = \tau_0 + \frac{L}{D_{max}}$$

Pour remplir une FIFO de longueur 2L + 1, il faut un temps à peu près égal à $\frac{2L}{D_{max}}$ .

Le signal $V_3$ devient inutile si nous nous assurons que :

$$\tau < \frac{2L}{D_{max}} \quad \rightarrow \quad \tau_0 < \frac{L}{D_{max}}$$

La suppression du signal $V_3$ conduirait à transmettre les mots des messages sans tenir compte du taux de remplissage des FIFOs. Le protocole est réduit à la priorité du transit des messages sur l'émission du message local.

Pour une ligne de haut débit ($\simeq$ 400 Kbauds) nous avons $D_{max}$ = 20 Kmots de 16 bits à peu près. Pour L égal à 100 mots nous obtenons $\tau_0 < 5$ ms. Cette valeur peut être respectée facilement puisque les manipulations faite par le processeur dans l'interface sont très simples (définition étiquette, positionnement drapeau). Une condition importante à respecter : ces programmes doivent être ininterruptibles.

VI.6 - Technique d'attribution de priorités aux interfaces

Nous avons considéré jusqu'à maintenant que toutes les interfaces ont la même priorité. Il est possible cependant d'attribuer certaines priorités d'émission de messages pour quelques interfaces. Soit $I_1$ l'interface que nous voulons améliorer sa priorité. Les protocoles proposés jusqu'ici énoncent tous que devant la fenêtre d'observation, il y a priorité du transit des messages sur l'émission du message local. Lorsqu'il y a un message à émettre, nous pouvons décaler la fenêtre d'observation de L+1 dans la FIFO à condition de rendre celle-ci de longueur 3L+1.

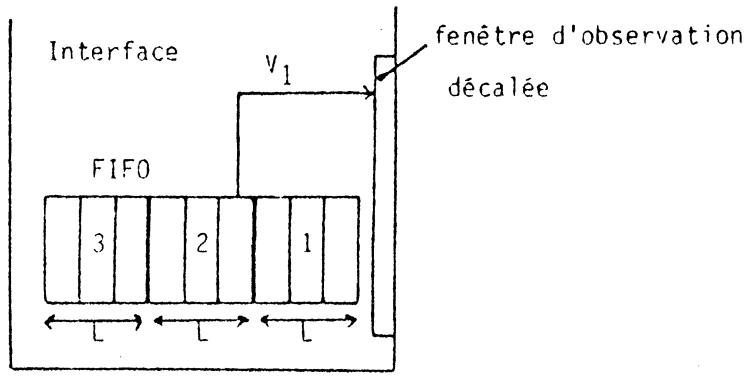



Fig. 5 : Technique d'attribution de priorité aux interfaces par décalage des signaux de la fenêtre d'observation.

L'effet de ce décalage est de permettre l'émission du message local même s'il y a un message à transiter dans la partie 1 de la FIFO.

Cas limite :

Plus la FIFO est profonde plus nous pouvons décaler la fenêtre d'observation. La priorité d'émission du message local devient elle aussi de plus en plus grande dans l'interface en question. Si la FIFO est assez grande, le protocole donnera toujours la priorité à l'émission du message local au lieu du transit des messages.

Les protocoles qui donnent la priorité à l'émission des messages nécessitent donc des tampons assez grands pour éviter l'autoblocage du système de communication. Nous retrouvons ainsi une conséquence bien connue des protocoles de communications qui donnent la priorité à l'émission du message local. La minimisation de la taille de ces tampons repose sur des études de nature stochastique [MEI 75] [MEI 78].

# BIBLIOGRAPHIE

[PIE 72] J.R. PIERCE, "How Far Can Data Loops Go ?" - I.E.E.E. Trans. on Communication,- Juin 1972 - pp. 527-530

[FAR 73] D.J. FARBER, J. FELDMAN, .... "The Distributed Computing System" - Seventh Annual I.E.E.E. Computer Society Intl. Conf. Proc. COMPCON 73 - Février 1973 - pp. 31-34

[NEW 72] E.E. NEWHALL, M.L. YUEN, "Traffic flow in a Distributed Loop Switching System" -Proc on Computer - Communications Networks and Telegraffic, Avril 1972 - pp. 29-46

[REA 76] C.C. REAMES, M.T. LIU, "Design and Simulation of the Distributed Loop Computer Network (DLCN)" - Proc. of the Third Annual Symposium on Computer Architecture - I.E.E.E. - Janvier 1976 pp. 124-129

[REA 75] C.C. REAMES, M.T. LIU, "A Loop Network For Simultaneous Transmission of Variable-Length Messages" - Proc. of the Second Annual Symposium on Computer Architecture - I.E.E.E. - Janvier 1975 - pp. 7-12

[AMD 76] "Memory Data Book" - Notices techniques - 1976

[MEI 75] B. MEISTER, "Queuing Analysis of General Loop Systems", Computing n° 14 - 1975 - pp. 349-365

[MEI 78] B. MEISTER, "A Tandem Queuing System with Priorities" - Computing n° 19 - 1978 - pp. 203-208.

# CHAPITRE VII

# APPLICATION : COMMANDE STOCHASTIQUE ADAPTATIVE MULTIMODELE

# Application d'une commande multimodèle

## VII.1 - Introduction

L'idée d'une description multiple d'un système a été utilisée pour la première fois par [MAG 65]. Les grands travaux dans ce domaine ont été réalisés par D.G. Lainiotis et son équipe à partir de 1970. Les domaines d'applications de cette technique sont très variés : systèmes économiques [LAI 76], aéronautique [ATH 77], systèmes énergétiques et physico-chimique [MON 77]. Cependant la plupart des travaux sont restés dans le domaine de simulations.

L'importance de cette méthode de commande découle de la représentation multiple du système à commander pour couvrir un large domaine de variations. Chaque modèle utilisé dans la commande a une zône restreinte de validité.

L'algorithme de commande détecte la qualité des modèles utilisés en attribuant à chacun d'entre eux une valeur d'indice de performance.

Une commande basée sur ces principes fut étudiée et testée [JAW 78] en simulation sur un IBM 360 (du Centre Interuniversitaire de Calcul de Grenoble) dans le cadre des travaux de l'équipe des systèmes complexes du laboratoire d'Automatique de Grenoble (LAG). Cet algorithme dit commande stochastique adaptative multimodèle (CSAM) est la première application du système multimicroprocesseur. Sa complexité numérique et son fort parrallélisme le prêtent bien au traitement parallèle.

Le but de ce chapitre est de présenter cet algorithme tel qu'il a été programmé sur le système multimicroprocesseur. Soulignons que cet algorithme a été testé du point de vue numérique seulement : les modèles utilisés dans les études précédentes de simulation ont été repris et nous avons comparés les résultats. L'expérimentation de cette commande sur un procédé réel (procédé pilote du laboratoire d'Automatique de Grenoble) sera l'objet d'une étude ultérieure.

## VII.2 - Aperçu sur l'algorithme de commande stochastique adaptative multimodèle.

La commande stochastique adaptative multimodèle consiste à utiliser un ensemble fini de modèles linéaires stochastiques. Le modèle

candidat est un modèle mathématique qui représente une image d'une situation physique particulière. La forme générale d'un tel modèle est une forme linéaire stachastique. Cet aspect stochastique du modèle intervient pour tenir compte des incertitudes paramétriques et structurelles des modèles et par conséquent pour avoir une représentation plus réaliste et plus naturelle d'une situation physique donnée.

Chaque modèle candidat $\theta_i$ est représenté par les équations suivantes :

$$\theta_i \begin{cases} x_i(k+1) = A_i X_i(k) + B_i U(k) + F_i w(k) \\ y_i(k) = C_i X_i(k) \end{cases}$$

$i = 1, 2, \ldots N$

Soit $Z(k)$ la sortie du système à commander. Le système peut prendre la forme de l'un des modèles candidats, $\theta_i (i \in [1,N])$ auquel s'ajoute un bruit à la sortie de celui-ci. Cette sortie peut donc être représentée par :

$$Z(k) = y_i(k) + v_i(k)$$

où $X_i(k)$ est le vecteur d'état à l'instant K de dimension $n_i$,
$U(k)$ est le vecteur de commande à l'instant k de dimension r,
$Z(k)$ est le vecteur de mesure à l'instant K de dimension m.

$W_i(k)$ et $V_i(k)$ sont des vecteurs de bruits considérés blancs gaussiens et de moyennes nulles connues à priori.

La connaissance à chaque instant des valeurs du vecteur d'état est nécessaire pour le déroulement des différentes étapes de l'algorithme. Ayant accès aux entrées et aux sorties, il faut construire ce vecteur d'état à partir de ces données. Pour celà on fait appel à la théorie de l'estimation, puisqu'on travaille dans un contexte stochastique. Un filtre de Kalman est utilisé pour construire le vecteur d'état à partir de mesures bruitées (pour chaque modèle). C'est un observateur asymptotique donnant une estimation du vecteur d'état dans l'environnement stochastique, lorsque les statistiques des bruits sont connues.

Nous disposons ainsi des moyens nécessaires aux calculs de la commande élémentaire de chaque modèle $\theta_i$.

La localisation consiste à situer les modèles candidats par rapport au système selon un certain critère de performance. La mesure de cette performance se fait à l'aide du calcul de la probabilité conditionnelle d'utiliser un modèle candidat. Le calcul de cette probabilité est basé sur l'approche Bayesienne et sur le calcul de la densité à posteriori de l'erreur de prédiction du modèle candidat.

La commande résultante ou globale est de la forme :

$$u(k) = \sum_{i=1}^{N} \hat{u}(k,\theta_i) Pr(\theta_i/k)$$

où $Pr(\theta_i/k)$ est la probabilité à postériori de $\theta_i$
$\hat{u}(k,\theta_i)$ est la commande élémentaire du modèle $\theta_i$
$\hat{u}(k)$ est la commande globale.

Une présentation mathématique détaillée est donnée dans [JAW 78].

## VII.3 - Décomposition de l'algorithme et organigrammes

### VII.3.1 - Décomposition

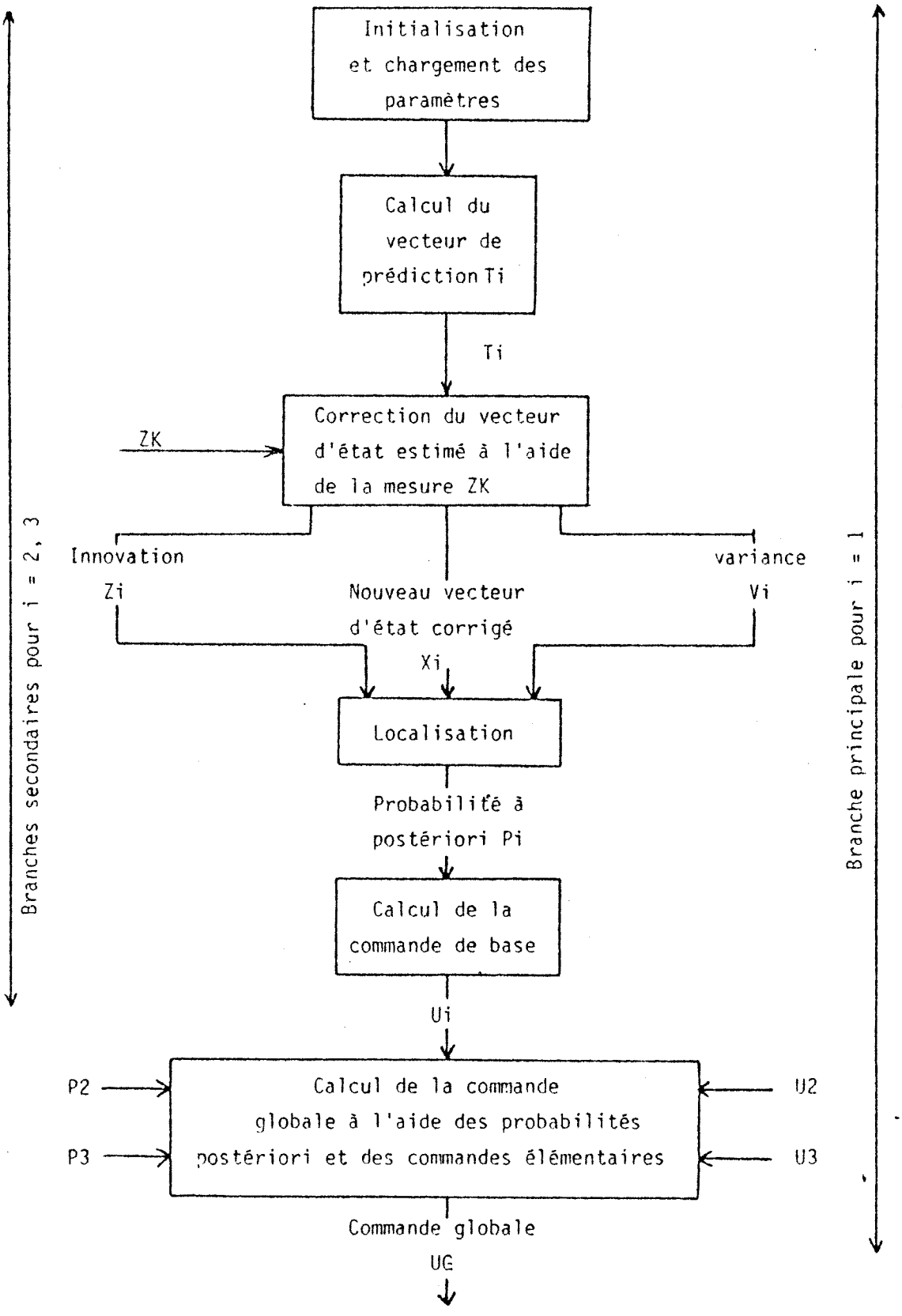
L'algorithme de commande multimodèle présente un parallélisme idéal. Le découpage le plus naturel consiste à implanter l'algorithme de chacun des modèles sur un module de traitement.

Nous avons pour chaque modèle :

- calcul d'estimation du vecteur d'état par un filtre de Kalman,
- calcul de la commande de base (ou élémentaire).

La synthèse de la commande globale a été associée avec l'un des modèles (modèle 1 dans notre cas). La branche de calcul ainsi formée sera appelée branche principale de l'algorithme. Les autres seront appelées branches secondaires (modèle 2 et modèle 3).

VII.3.2 - Organigrammes



VII.4 - Ecriture des programmes au moyen des macroinstructions

Les calculs relatifs à l'algorithme ont été écrits avec l'aide des macroinstructions des opérateurs matriciels développés par le laboratoire de Microinformation de Grenoble [EYN 78].

Les opérations arithmétiques de base (addition, multiplication,...) sont effectuées en virgule flottante avec le circuit arithmétique AM 9511 (voir annexe 1).

VII.4.1 - Programme de la branche secondaire (ex : modèle 2)

Le chargement des valeurs des conditions initiales est supposé effectué au moment de l'initialisation du module.

```
Début :
     ENTRER "ZK"
     Calcul du vecteur de prédiction
     SORTIR "Y2"
     ATTENDRE "ZK"
     Correction du vecteur d'état
     SORTIR "Z2"
     SORTIR "X2"
     SORTIR "V2"
     Calcul localisation
     SORTIR "P2"
     Calcul de la commande de base
     SORTIR "U2"
Fin.
```

VII.4.2 - Programme de la branche principale

```
Début :
     ENTRER "ZK,P2,U2,P3,U3"
     Calcul du vecteur de prédiction
     SORTIR "T1"
     ATTENDRE "ZK"
     Correction du vecteur d'état estimé
     SORTIR "Z1"
     SORTIR "X1"
```

```
        SORTIR "V1"
        Calcul de la localisation
        SORTIR "P1"
        Calcul de la commande de base
        SORTIR "U1"
        ATTENDRE "P2,U2,P3,U3"
        Calcul de la commande globale
        SORTIR "UG"
Fin
```

## VII.5 - Résultats expérimentaux

### VII.5.1 - Réservation mémoire

Le programme de chaque modèle est constitué entièrement d'appels aux opérateurs matriciels et des chargements de leurs paramètres. Le programme de la branche principale fait 2 Koctets environs et nécessite 2,5 Koctets de mémoire vive pour les modèles ayant deux entrées, deux sorties et six états.

Notons quand même qu'il est envisagé d'utiliser des modèles de tailles plus grandes.

### VII.5.2 - Temps d'exécutions

Avec un modèle scalaire (mono entrée et mono sortie) le temps d'exécution du programme de la branche principale est 0,5 seconde environ.

Avec un modèle à deux entrées, deux sorties et six états (modèle envisagé pour la représentation de la colonne du laboratoire d'Automatique de Grenoble) le temps d'exécution s'élève à trois secondes pour le programme de la branche principale. Les calculs relatifs au filtre de Kalman sont les plus prépondérants.

Les résultats numériques obtenus sont conformes à $10^{-4}$ près à ceux obtenus sur l'IBM 360.

### VII.5.3 - Influence des "frais généraux" sur l'efficacité du circuit arithmétique

Le transfert d'information du ou vers le circuit arithmétique représentent un pourcentage important du temps d'exécution totale d'une opération arithmétique, chargement des opérandes et récupération du résultat sous entendu.

Les "frais généraux" de ces programmes sont de l'ordre de 600 cycles en moyenne pour chaque opération effectuée dans le circuit. L'opération effective est réalisée par le circuit en 200 cycles environ. Avec une horloge à 2 MHz, pour le microprocesseur et pour le circuit, le temps d'exécution total devient 400 µs (800 x 0,5 µs). A peu près 75 % de ce temps est nécessaire aux "frais généraux".

Avec une arithmétique flottante programmée, le temps d'exécution d'une multiplication est de l'ordre de 2 ms (FPAL-Intel). Le gain réel apporté par l'emploi du circuit arithmétique se situe aux alentours d'un coefficient égal à 5.

Ce coefficient peut être amélioré en optimisant les sous-programmes de chargement conjointement avec les procédures d'appels aux opérateurs matriciels.

# BIBLIOGRAPHIE

[MAG 65] D.T. MAGILL, "Optimal Adaptive Estimation of Sampled Stochastic Processes" - I.E.E.E. Trans. on Automatic Control - Octobre 1965

[ATH 77] M. ATHANS et al., "The Stochastic Control of the F-8C Aircraft Using a Multiple Model Adaptive Control (MMAC) Method-Part I : Equilibrium Flight" - I.E.E.E. Trans. on Automatic Control - Octobre 1977

[LAI 76] D.G. LAINIOTIS, "Partitionning : a Unifying Framework for Adaptive Systems" - I.E.E.E. Proceeding, Août 1976

[MON 77] B. MONNIER, "Contributions à la commande d'une classe de procédés dynamiques industriels dans de grands domaines de fonctionnement" - Thèse de Docteur Ingénieur - INP Grenoble 1977

[JAW 78] S. JAWHARI, "Sur la commande stochastique adaptative multimodèle" Thèse 3ème Cycle - INP Grenoble 1978

[EYN 78] J.P. EYNARD, "Réalisation d'outils logiciels pour la mise en oeuvre de microprocesseurs dans la conduite automatique de procédés complexes" - Mémoire d'Ingénieur CNAM - Février 1979.

# CHAPITRE VIII

# CONCLUSIONS

# CONCLUSIONS

L'idée de base du système multiprocesseur du centre de décision était d'obtenir un système :

- puissant par son traitement parallèle,
- pratique par sa modularité,
- et adapté à l'application avec un bagage minimal de logiciel.

Le centre multimicroprocesseur pour la commande automatique est constitué d'un ensemble de modules (paire microprocesseur/mémoire privée) interconnectés en anneau et spécialisés fonctionnellement. L'allocation statique des fonctions aux modules a permis de sauvegarder l'homomorphisme ou la correspondance directe entre la topologie réelle du système informatique et celle des fonctions automatiques.

Le mécanisme de communication de base entre les différents modules, constitué par des messages explicites à destinations multiples non spécifiées, a rendu le logiciel plus modulaire et moins consistant. Le module n'a pas à connaître explicitement les modules producteurs ou consommateurs de ces variables en entrée ou en sortie. L'ensemble de ces variables définit les liens du module avec son environnement. Il n'est visible de l'extérieur que par ces variables de sorties. Cette organisation reflète en quelque sorte le mode de raisonnement de l'automaticien qui est habitué à utiliser des schémas blocs et des fonctions analogiques.

Comme tout système comportant un traitement parallèle un outil de synchronisation est nécessaire. Le mécanisme adopté ne cherche pas à résoudre des problèmes académiques. Il est basé sur un mécanisme d'attente d'évènements simplifié en considérant :

- que le module est un processus séquentiel,
- qu'il n'y a pas de création dynamique de processus,
- que les activités parallèles sont connues à l'avance.

L'interconnexion des modules est mise en oeuvre en associant à chacun de ces modules une interface de communication microprogrammée. L'interface constitue le noyau du module. Elle assure :

- la communication et la reconnaissance des messages,
- le contrôle des erreurs de transmission,
- la gestion locale de l'envoi des messages,
- l'implémentation des primitives de synchronisation et de temps réel.

La surpuissance de l'interface de communication (réalisée avec des mécanismes spéciaux : accès direct à la mémoire, queue FIFO intégrée...) nous a valu des ... critiques. Notre point de vue est que cette surpuissance nous a permis d'une part de s'affranchir de la notion des priorités des messages souvent source de problèmes et d'autre part de simplifier dans de grandes proportions la liaison entre l'interface et son module.

Une grande partie des travaux a été consacrée à l'étude et à la réalisation d'un système triprocesseur et au développement d'un logiciel de base (dialogue opérateur, édition, visualisation graphique). En application, nous n'avons pu tester la commande multimodèle que numériquement. Son application sur un procédé réel est l'objet d'études futures.

L'évaluation finale du système ne peut pas être donnée sans implanter la totalité de l'application et sans effectuer des mesures rationnelles. Ces deux axes constituent les prolongements normaux de notre projet.

# ANNEXE 1

# LOGICIEL DE BASE

## Annexe 1

### A.1.1 - Modules opérationnels

Les modules prévus pour Cemuta sont :

- module de dialogue,
- plusieurs modules de traitement (2 ou 3),
- module de communication avec le réseau externe,
- module d'instrumentation (Smire).

Actuellement trois modules sont déjà opérationnels :

Deux modules de traitement équipés chacun :

- d'une carte SBC 80/10 Intel (µp 8080 A)
- d'une carte extension qui comprend :
  - . une mémoire vive (RAM) de 6 Koctets
  - . une mémoire morte ($PROM_S$ de la famille 2708) de 4 Koctets
  - . un circuit arithmétique AM 9511 (2 MHz)
- une bibliothèque d'opérateurs matriciels développée par le laboratoire de microinformatique. Des routines spéciales à l'utilisation de l'interface sont disponibles ( 400 octets).

Un module de dialogue équipé :

- d'une carte SBC 80/10 Intel (µp 8080)
- d'une carte extension mémoire qui comprend :
  - . une mémoire vive (RAM) de 4 Koctets;
  - . une mémoire morte (PROM) de 16 Koctets
- d'un moniteur d'exploitation et de mise au point.

### A.1.2 - Dialogue d'exploitation

Le dialogue d'exploitation développé jusqu'ici a été limité aux besoins immédiats pour la mise en route et le test des programmes implantés sur le processeur de traitement.

Nous avons recherchés le maximum de transparence pour l'utilisateur c'est-à-dire que le système lui apparaît autant que possible comme un système monoprocesseur dans l'initialisation, la modification et la consultation des variables.

Rappels : Les variables visibles dans Cemuta[::] constituent l'ensemble des variables déclarées en entrées ou en sorties dans toutes les interfaces donc accessible en écriture ou en lecture respectivement.

Les variables visibles du système ont des identificateurs limités à deux caractères, commençant obligatoirement par un caractère alphabétique. Le deuxième caractère est alphabétique ou numérique (de 0 à 7). Nous disposons ainsi d'un jeu de 675 identificateurs possibles pour les variables visibles (dite aussi globales).

Les environnements des variables

Il existe trois types d'environnements pour les variables manipulées par l'opérateur (fig. 1)

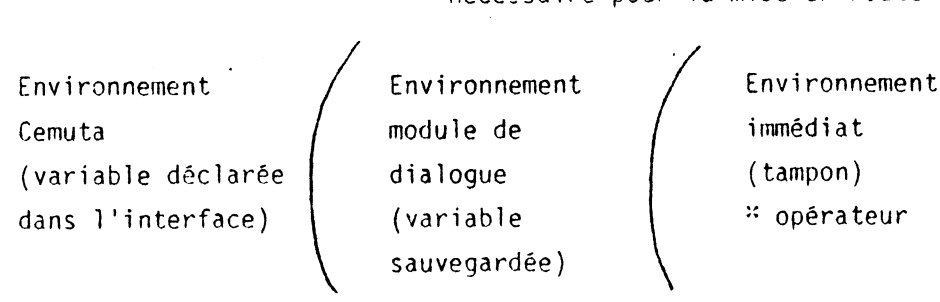



Fig. 1 : Les environnements d'une variable

Les commandes

L'étoile imprimée en début de ligne indique que nous sommes sous le moniteur de dialogue de Cemuta.

---

[::] CEMUTA : Centre multiprocesseur pour le traitement automatique.

1. :: Identificateur <nb lignes ; nb colonnes> ←
   permet de définir une variable avec les dimensions spécifées dans l'environnement immédiat. Les éléments sont entrés colonne par colonne s'il s'agit de matrices.

Rem : Nb lignes et nb colonnes maximum deux chiffres pour chacun.

Chaque élément entré est délimité par un blanc. Il y a retour chariot automatique pour entrer l'élément suivant.

Ex : :: RT <2 ; 2> ←

```
COL 01
 1
 2
COL 02
 3
 4
::
```

Le numéro de colonne est imprimé par le moniteur.

Les élements sont convertis en flottant AM9511 sur quatre octets.

2. ::/Sauvegarder CR (retour chariot)
   permet de sauvegarder la variable définie auparavant dans une zône mémoire du module gérée dynamiquement (à partir de l'adresse $4400_H$ et de capacité 3 Koctets). La variable devient dans l'environnement du module de dialogue.

3. ::/Lister CR
   permet de lister les identificateurs définis dans le module de dialogue plus leurs adresses d'implantations dans la mémoire (en hexadécimal).

Ex : ::/L CR

```
 RT     4400
 X1     4410
::
```

4. Identificateur :
   Imprime le contenu de la variable référencée par l'identificateur. Un point d'interrogation est imprimé si l'identificateur n'a pas été défini (et sauvegardé) auparavant.

5. :/Annuler Identificateur CR
   permet de faire un "Kill" de la variable référencée par l'identificateur. La variable n'est plus définie dans l'environnement du module de dialogue.

6. :/Emettre identificateur CR
   permet de faire des copies de la variable dans les modules qui ont cette variable définie comme une entrée. Cette commande est utilisée pour les initialisations des variables dans les autres modules. Le moniteur la définit comme une sortie dans l'interface du module de dialogue.

7. :/Journal identificateur CR
   permet d'imprimer le contenu de la variable chaque fois qu'elle est transmise dans l'anneau par le module qui l'élabore. Le moniteur définira cette variable dans l'interface du module de dialogue comme une entrée à laquelle est associée une interruption qui génère la routine d'impression.

8. :/Consulter identificateur CR
   permet de visualiser à la demande le contenu d'une variable visible.

9. :/Graphique CR
   cette commande est utilisée pour afficher des variables sous forme de courbes sur la console.

Après retour chariot le processeur de dialogue répond par : DEF.COUR = 0 ou DEF.CONF = 1 ? en répondant par 0 ou 1 nous passons respectivement à la définition d'une courbe ou à la définition d'une configuration.

Définition configuration :

Le dialogue est effectué par une série de messages envoyés par le processeur de dialogue auxquels l'utilisateur doit répondre (le dialogue peut être suspendu par la touche ESCAPE à tout moment ). Toutes les réponses sont suivies par CR.

PAS : deux digits hexadécimaux définissent le pas de la courbe (pas minimal 04)

ABSCISSE : quatre digits hexadécimaux définissent l'abscisses de départ

NOMBRE DE ZONES : un seul digit (de 1 à 4). L'écran peut être subdivisé jusqu'en quatre zônes (quatre axes d'abscisses)

NOMBRE DE COURBES DEFINIES : un seul digit (de 1 à 9). Le dialogue configuration est terminé par impression d'un astérisque au début de la ligne suivante.

Définition courbe

NUM. COURBE : un seul digit (de 1 à 9)

NOM : nom de la variable à afficher (deux caractères alphanumériques)

GRAPHISME : la courbe peut être visualisée suivant l'un des trois graphismes
1 - continu
2 - discontinu : un pas sur deux
3 - discontinu : un pas sur trois
un seul digit (de 1 à 3) défini le graphisme

NUM. ZONE : permet de visualiser la courbe dans la zône spécifiée. Il est possible de superposer trois courbes dans une même zône. Un digit (de 1 à 4) spécifie la zône

VALEUR INIT : permet d'initialiser l'ordonnée de la courbe en question. C'est une valeur réelle.

ECART MAXI : permet de cadrer la courbe dans les limites de la zône ou elle est définie. C'est une valeur réelle.

Commandes de mise au point

:/X n° processeur, identificateur,[entrée = 2, sortie = 1, entrée et interruption = 0], adresse (en hexadécimal), n° routine. Longueur CR (en hexa sur quatre digits).

Cette commande permet de définir une étiquette dans l'interface du processeur spécifié. L'adresse en hexa indique le lieu d'implantation de la variable relative à l'identificateur. Si une interruption est associée à l'identificateur, le numéro de la routine correspondante est concaténé avec la longueur de la variable. Il est pris égal à zéro par défaut.

:/Insérer n° processeur, adresse (en hexadécimal) CR

Permet d'entrer un ou plusieurs octets à partir de l'adresse spécifiée dans la mémoire du processeur spécifié. Deux digits hexadécimaux sont entrés pour chaque octet. Les blancs entre les digits sont ignorés. La commande est terminée par le caractère ESCAPE.

:/Display n° processeur, adresse inférieure, adresse supérieure CR

Imprime en hexadécimal le contenu de la zône mémoire, délimitée par les deux adresses, du processeur spécifié.

:/Modifier n° processeur, adresse CR

Sélectionne l'octet spécifié par l'adresse . Le contenu est imprimé en hexadécimal suivi d'un tiret. La modification est faite en tapant la nouvelle valeur suivie d'un blanc. Le contenu de la position mémoire suivante est imprimée et ainsi de suite. La commande est terminée par un retour chariot (CR).

:/W n° processeur, adresse inférieure, adresse supérieure CR

Permet de sortir le contenu de la zône mémoire du processeur spécifié sur un ruban (format hexadécimal Intel)

:/R n° processeur CR

Permet de charger le contenu du ruban perforé (format hexadécimal Intel) dans la mémoire du processeur spécifié.

Les programmes du dialogue d'exploitation font 7 Koctets dont 2,5 K pour les programmes de conversion décimal ~ flottant.

## A.1.3 - Utilisation de l'arithmétique flottante

### A.1.3.1 - Circuit arithmétique Am 9511

C'est un circuit MOS LSI monolithique permettant de réaliser des opérations arithmétiques et trigonométriques avec de hautes performances. Son utilisation dans le processeur de traitement à permis d'une part d'améliorer le temps d'exécution des algorithmes et d'autre part de disposer d'une bibliothèque complète d'opérations numériques classiques en virgule flottante et en virgule fixe.

Les transferts d'information (opérande, résultat, mot d'état du circuit, et mot de commande) sont faits par l'intermédiaire d'un bus bidirectionnel 8 bits. Les opérandes sont entrés d'abord dans une pile interne au circuit puis le mot de commande est envoyé pour définir la nature de l'opération à effectuer. Le résultat est fourni au sommet de la pile. Son extraction est faite par des lectures successives du sommet de la pile.

### A.1.3.2 - Format

L'opérande est représentée sur 32 bits suivant le format de la figure 1.

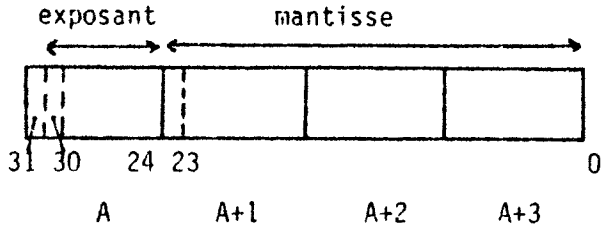



Fig. 1 : Format d'un mot en virgule flottante Am 9511

Plage des valeurs : $\pm(2.7 \times 10^{-20}$ à $9.2 \times 10^{18})$
Le bit 31 est le signe de la mantisse,
Le bit 30 est le signe de l'exposant,
Le bit 23 est le bit le plus significatif de la mantisse.

Le bit 23 est toujours égal à 1 (pour assurer la normalisation) sauf dans le cas où l'opérande est nul. Les 32 bits sont alors tous nuls.

Les quatres octets du mot sont placés dans la mémoire avec octet exposant à l'adresse A et octet le moins significatif de la mantisse à l'adresse A+3.

A.1.3.3 - Montage du circuit

Le circuit AM 9511 peut être couplé au microprocesseur suivant des configurations allant du montage le plus simple jusqu'au montage le plus compliqué :

- Transfert d'informations par des instructions d'entrée-sorties (IN et OUT) ou de transfert (MOV, LDA, STA) et attente active du résultat par le microprocesseur (test du bit de disponibilité du résultat dans le mot d'état du circuit)

- Transfert d'information par mécanisme d'accès direct à la mémoire et disponibilité du résultat signalée par interruption.

Nous avons adopté le premier montage avec transfert d'informations par des instructions telles que LDA et STA. Le circuit arithmétique est vue par le microprocesseur comme une zône mémoire (avec les instructions IN et OUT le circuit arithmétique est monté comme un "port").

A.1.3.4 - Sous programmes élémentaires d'exploitation du circuit arithmétigues

Ces sous programmes sont appelés par les opérateurs matriciels.

a) Chargement d'un opérande dans le circuit :
   - sous programme FLOAD : point d'entrée 05E0H
   - le registre 16 bits DE soit contenir l'adresse dans la mémoire de l'octet le moins significatif de la mantisse (A+3) du mot de l'opérande à charger
   - aucun registre n'est détruit

b) chargement d'un opérande plus un code opération
   - sous programmes FMUL : point d'entrée 0BF1H
     FDIV : point d'entrée 0BF6H
     FSUB : point d'entrée 0BFBH
     FADD : point d'entrée 0BE5H

- le registre 16 bits DE doit contenir l'adresse dans la mémoire de l'octet le moins significatif de la mantisse (A+3) du mot du 2ème opérande.

Ces sous programmes chargent le 2ème opérande dans le circuit suivi du code opération approprié. L'opération porte sur le premier opérande chargé par FLOAD et mémorisé dans la pile du circuit, et sur ce deuxième opérande.

- aucun registre n'est détruit.

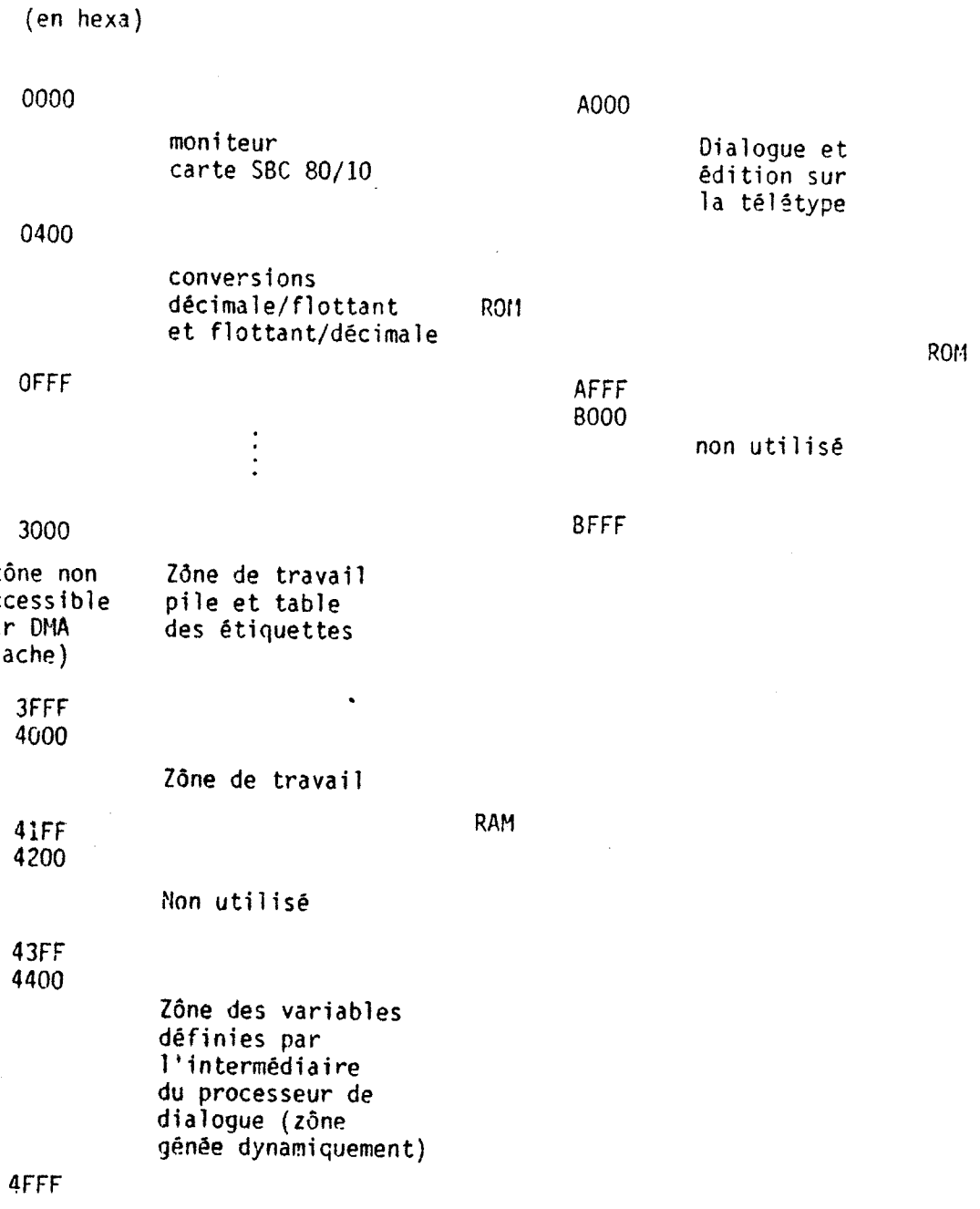
c) Récupération du résultat et chargement dans la mémoire

- sous programme FSTOR : point d'entrée 05F5H
- le registre 16 bits DE doit contenir l'adresse de l'octet le moins significatif de la nombre (A+3)
- l'attente active de la disponibilité du résultat est incluse dans le sous programme FSTOR
- aucun registre n'est détruit par ces routines.

ORGANISATION DES ESPACES MEMOIRES DES MODULES DE TRAITEMENT

| | module de traitement 1 | | | module de traitement 2 | |
|---|---|---|---|---|---|
| 0000 | moniteur carte BSC 80/10 | | | moniteur cartes SBC 80/10 | |
| 03FF | initialisation interface exploitation circuit arithmétique | | | initialisation interface exploitation circuit arithmétique | |
| 07FF | | | | | |
| 0800 | opérateurs matriciels | ROM | | opérateurs matriciels | ROM |
| 0BFF | | | | | |
| 0C00 | visualisation graphique | | | communication avec SMIRE | |
| 0FFF | | | | | |
| 1000 | Algorithme modèle 1 | | 1000 H | Algorithme modèle 2 | |
| 17FF | | | 17FF | | |
| 1800 | Zône utilisateur | | 1800 | Zône utilisateur | |
| 1FFF | | | 1FFF | | |
| | : | | | : | |
| 3000 | Zône de travail et pile | | 3000 | Zône de travail et pile | |
| zône non accessible par DMA (cache) | non utilisé | | 3E00 | non utilisé | |
| 4000 | zône de travail | RAM | 4000 | zône de travail | RAM |
| 41FF | paramètres visualisation | | 41FF | paramètres communication | |
| 43FF | | | 43FF | | |
| 4400 | variables modèles | | 4400 | variables modèles | |
| 4FFF | | | 4FFF | | |
| 5000 | zône utilisateur | | 5000 | zône utilisateur | |
| 57FF | | | 57FF | | |
| 5FF0 | donnée | | 5FF0 | donnée | circuit |
| 5FF1 | commande | | 5FF1 | commande | arithmétique |

Organisation de l'espace mémoire
du module de dialogue



# ANNEXE 2

# DESCRIPTION DETAILLEE DU MECANISME DE COMMUNICATION DES MESSAGES

ANNEXE 2

Description détaillée du mécanisme de communication des messages

A.2.1 - Structure du message

Le message est composé de trois parties (fig. 1).

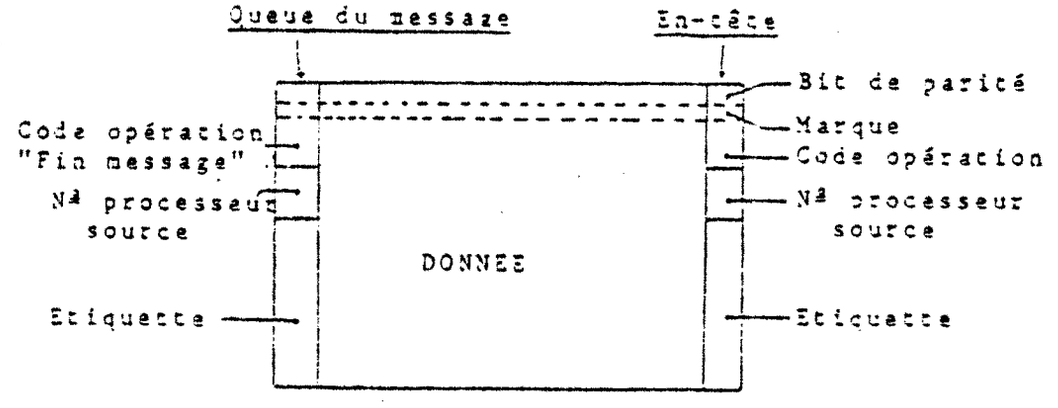



Fig. 1 : structure d'un message

- d'un entête,
- d'un corps renfermant une donnée,
- et d'une queue.

Un bit de parité est associé à chaque mot du message. La marque permet de différencier l'entête et la queue du corps du message : marque = 0 pour l'entête et la queue et marque = 1 pour le corps. Le mot de 16 bits a été adopté en vue d'une compatibilité avec des µp 16 bits.

A.2.1.1 - Entête

Il est composé :

- d'un code opération qui spécifie la nature du message : message de donnée/ ou commande et destination unique/ou multiple ,

- d'un numéro processeur (celui de la source) inséré par l'interface. Lorsque plusieurs messages circulent dans l'anneau, le numéro processeur constitue un repère pour l'interface émetteur au lieu du repère temporel obtenu par l'émission d'un seul message à la fois dans le système de communication,

- d'une étiquette qui identifie la donnée contenue dans le message. Les étiquettes sont déduites des identificateurs utilisés par le programmeur par concaténation des cinq premiers bits des deux premiers caractères de l'identificateur (fig. 2.) Ceci implique qu'il y ait un moins un caractère distinct sur les deux premiers caractères des identificateurs des variables d'entrée ou de sortie des modules. Si les identificateurs sont codés en code ASCII, le jeu de combinaisons est de 676 étiquettes. Par contre, en code EBCDIC nous avons 324 étiquettes. C'est largement suffisant pour notre application dans les deux cas.

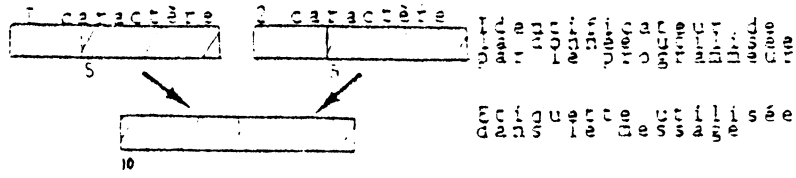



Fig. 2 : Construction d'une étiquette

Le premier caractère est nécessairement alphabétique. Le deuxième peut être alphabétique (de A- 0 ), numérique (1 à 9) ou un blanc. Les combinaisons restantes sont utilisés pour les messages spéciaux.

A.2.1.2 - Corps du message.

Il renferme la donnée à communiquer. Avec des microprocesseurs 8 bits, seule la partie supérieure du mot est utilisée. Les autres 8 bits du mot sont mis à "un".

A.2.1.3 - Queue

La queue possède une structure identique à celle de l'entête. Elle retourne toujours à la source du message pour valider le transfert ou demander une retransmission en cas d'erreurs... La queue joue pour la commande de sortie un rôle principal dans la connaissance de l'état des entrées dans les autres modules. Les différentes situations sont résumées dans le code opération de la queue.

A.2.2 - Les registres de contrôle des données

Les registres de contrôle des données constituent le noyau du système de communication et de synchronisation. Leur rôle est de contrôler les accès en lecture ou en écriture aux variables déclarées en sorties ou en entrées dans les interfaces de communication. Ces registres sont pour le module le seul moyen de connaître l'état des autres modules puisque tous les transferts d'information se font par accès direct à la mémoire donc sans la participation immédiate du processeur.

A chaque étiquette de sortie ou d'entrée du module définie dans l'interface correspond un vecteur de contrôle des données composé de 6 drapeaux (a, b, c, d, e et f).

Les drapeaux sont classés en deux groupes :

- le 1er groupe est mis à jour par le processeur et consulté par l'interface. Ce groupe reflète l'état interne des entrées et des sorties du module.

- le 2ème groupe est mis à jour par l'interface et consulté par le module. Ce groupe de drapeaux reflète les états des sorties et entrées des autres modules.

Signification des drapeaux

Les drapeaux ont des significations légèrement différentes suivant que l'étiquette correspond à une entrée ou une sortie pour le module.

Le tableau I donne le rôle de chaque drapeaux (a, b, c, d) pour les deux types. Un drapeau est dit positionné quand il est à l'état logique zéro.

En vue de faciliter l'exposé, nous attribuons aux drapeaux les mnémoniques suivantes :

drapeau a = DIS (de disponible)
drapeau b = RT (de retransmission)
drapeau c = ACC (de accompli)
drapeau d = PRT (de prêt)

| type étiquette / drapeau | en entrée | en sortie |
|---|---|---|
| a = 0 | indique que la sortie-module est devenue prête tandis que l'entrée ne l'était pas (l'entrée passe au niveau 2) | indique la réception d'un accusé non définitif : une entrée-module au moins n'est pas à l'état prêt |
| b = 0 | indique que l'interface est en attente de retransmission (l'entrée-module passe au niveau 2) | indique qu'une demande de transmission s'est manifestée |
| c = 0 | indique que le transfert d'information avec la sortie-module a été accompli (l'entrée passe donc au niveau 3) | indique la réception de l'interface d'un accusé définitif de transmission et de synchronisation |
| d = 0 | indique que l'entrée-module est prête pour le transfert d'information | indique que la sortie-module est prête au transfert d'information |

Tableau I : Signification des drapeaux

Les drapeaux c et f ont pour rôle la définition du type de l'étiquette :

| e | f | type étiquette |
|---|---|---|
| 0 | 1 | sortie-module |
| 1 | 0 | entrée-module |
| 0 | 0 | entrée + interruption |

La combinaison (e,f) = 00 indique à l'interface qu'une interruption doit être générée au processeur à la fin de chaque transfert valide.

A.2.3 - Les codes opérations

Comme nous l'avons vu l'entête est la queue du message contiennent des codes opérations (fig. 3)

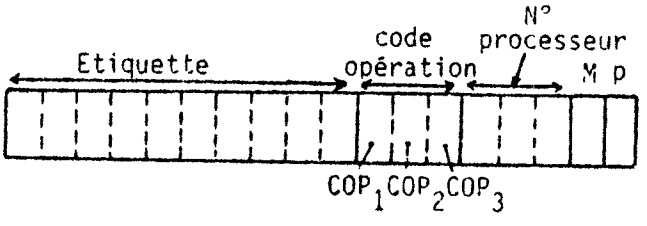



Fig. 3 : Structure d'un entête ou d'une queue

Les différents codes opérations sont classés en deux groupes :

a) le 1er groupe constitue des directives pour les interfaces sur la manière d'interpréter le message. Ces codes opérations apparaissent uniquement dans un entête.

| $COP_3$ | $COP_2$ | $COP_1$ | |
|---|---|---|---|
| 0 | 0 | 0 | message retransmis à destination multiples (MRT) |
| 0 | 0 | 1 | message à destination unique (MDU) |
| 1 | 0 | 0 | message de demande de transmission (MDRT) |
| 1 | 1 | 1 | message à destinations multiples (MDM) |

b) le 2ème groupe de codes opérations sert de bilan à l'interface source sur l'état du transfert d'information avec les entrées dans les autres modules. Ces codes apparaissent uniquement dans la queue.

| $COP_3$ | $COP_2$ | $COP_1$ | |
|---|---|---|---|
| 0 | 1 | 0 | indique un message d'erreur (ME). Une entrée-module au moins n'est pas définie avec les mêmes spécification que la source |
| 1 | 0 | 1 | indique un accusé définitif de réception. Toutes les entrées-modules sont satisfaites. C'est le code opération initial "fin message" (FM) non modifié après un tour complet |

| | | | |
|---|---|---|---|
| 1 | 1 | 0 | indique un accusé non définitif (AND) une entrée-module au moins n'est pas à l'état prêt. C'est un code opération de synchronisation |
| 0 | 1 | 1 | indique qu'une erreur de transmission a été détectée quelque part par le contrôle de parité (ETR) |

A.2.4 - Evolutions du code opération de la queue

Le code opération de la queue du message est sujet à des changements rencontrées par les interfaces rencontrées le long du parcours. Le code opération final constitue le bilan de ce parcours sur les entrées-modules rencontrées. Nous décrivons leurs changements et leurs actions sur les drapeaux des vecteurs de contrôle des données en fonction des différents types de messages.

A.2.4.1 - Message à destinations multiples (MDM)

Au départ de l'interface, le message contient le code "fin message" dans sa queue. Ce message va circuler dans l'anneau et passer sous toutes les fenêtres d'observations pour consulter toutes les entrées-modules possibles pour son étiquette.

A.2.4.1.1 - Mécanisme au niveau de l'entrée-module

a) Entrée-module prête et pas d'erreur de transmission. Le transfert d'information est accompli. L'interface positionne le drapeau ACC (accompli) de l'entrée-module. (fig. 4)

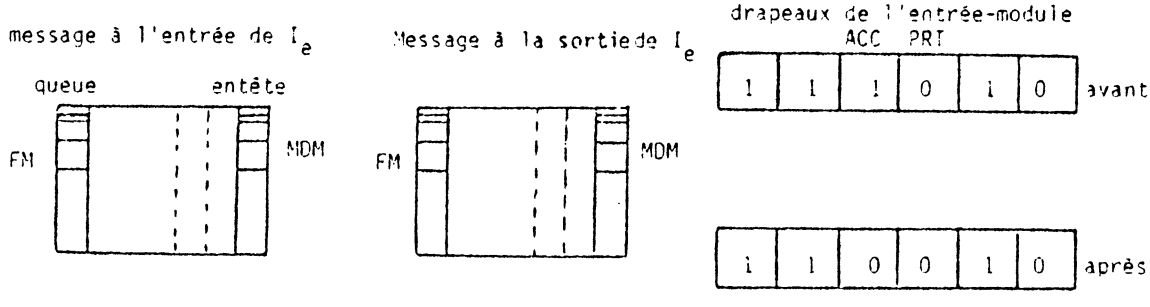



Fig 4 : entrée-module prête et pas d'erreurs de transmission.

Ie = interface contenant une entrée-module.

b) entrée-module non prête et pas d'erreurs de transmission. Le drapeau DIS est positionné pour indiquer à l'entrée-module que la sortie est déjà prête (fig. 5). Le code opération "fin message" de la queue est changée en accusé non définitif.

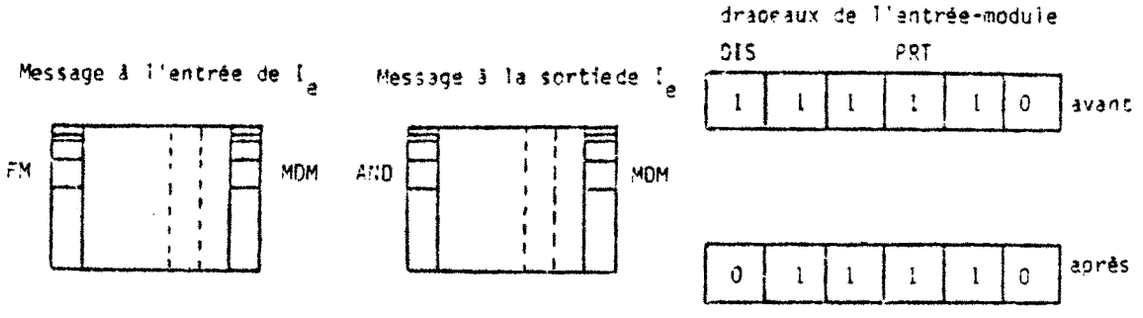



Fig. 5 : entrée-module non prête et pas d'erreurs de transmission

c) entrée-module prête et erreur de transmission. Le drapeau RT est positionné et le code opération de la queue est changé en code "erreur de transmission" (ETR) (fig. 6)

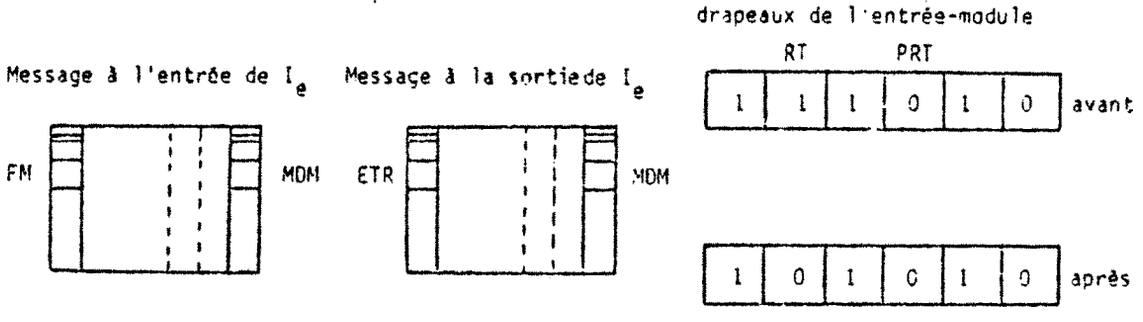



Fig. 6 : entrée-module prête mais erreur de transmission

Remarque importante : Si le code de la queue a été changée en code "accusé définitif" par une interface en amont, il y a priorité au code "erreur de transmission".

d) entrée-module pas prête et erreur de transmission. Le même mécanisme qu'en c) sauf que DIS est positionné en lieu de RT. Le contrôle de la transmission est effectué uniquement par les interfaces "intéressées" par le message.

A.2.4.1.2. - Mécanisme au niveau de la sortie-module au retour du message

Le message est retiré après un tour complet de l'anneau par l'interface qui l'a émis. Cette interface consulte le code de la queue pour connaitre le bilan du transfert d'information avec les entrées des modules.

a) Code opération "fin message" :

Le transfert d'information est donc accompli avec toutes les entrées des modules. Le drapeau ACC est alors positionné (fig. 7)

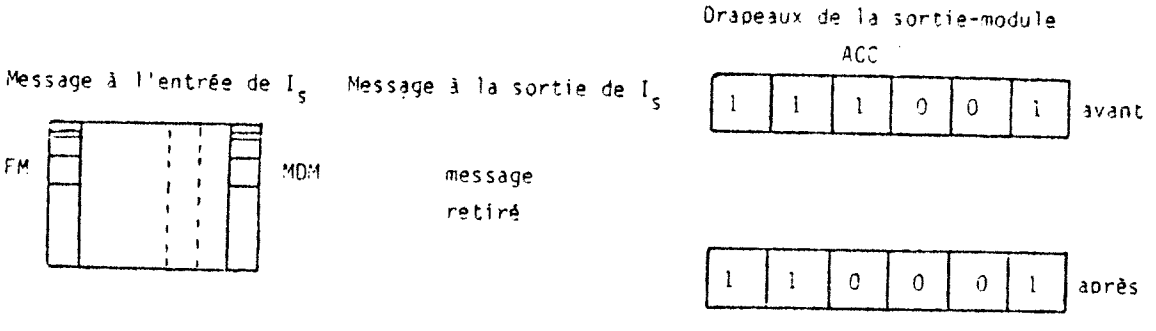



Fig. 7 : actions du code "fin message" sur les drapeaux de la sortie-module

Is = interface source du message.

b) Code opération "accusé non définitif". Le drapeau DIS est positionné.

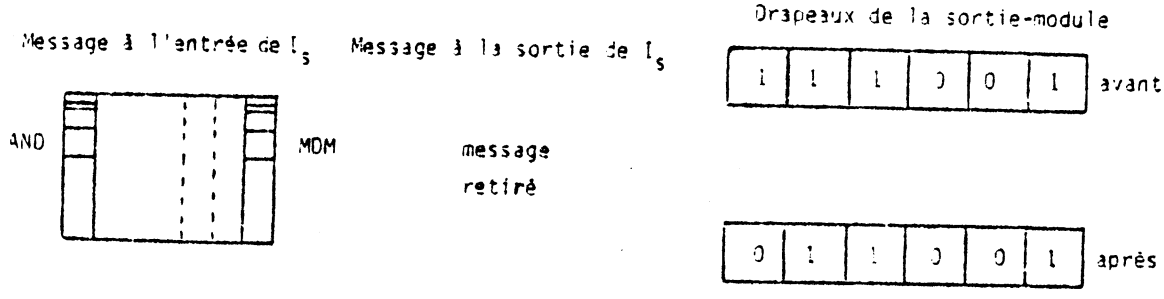



Fig. 3 : actions du code "accusé non définitif" sur les drapeaux de la sortie-module.

c) Code opération "erreur de transmission". Le drapeau RT est positionné. Le module retransmet le message avec le code opération "retransmission" dans l'entête du message. Le drapeau RT est restauré par le module.

La retransmission automatique du message est possible. Ce mécanisme a été implanté dans les interfaces réalisées. Il permet de répondre aux retransmissions très rapidement et simolifie le logiciel. L'inconvénient est de na pas pouvoir contrôler le nombre des retransmissions effectuées.

d) Code opération "erreur". Ce code génère une interruption au processeur quand il est rencontré.

A.2.4.2 - Message à destination unique (MDU)

Ce message est retiré de l'anneau après la première entrée rencontrée. Seule la queue persiste et retourne à l'interface émettrice du message. Les mécanismes aux niveaux des entrées et des sorties des modules sont parfaitement identiques aux précédents.

A.2.4.3 - Message retransmis (MRT)

Le rôle du code retransmission dans l'entête du message est de "court-circuiter" toutes les interfaces ayant reçu le message correctement auparavant. Ce message ne concerne que les entrées-module qui ont leurs drapeaux RT ou DIS positionnés.

A.2.4.3.1 - Mécanisme au niveau de l'entrée-module

a) Drapeau RT positionné et pas d'erreurs de transmission de nouveau. Le fait que RT ait été positionné, signifie qu'à la première occurence du message, l'entrée-module était prête mais il y a eu erreur de transmission.

Dans ce cas, RT est restauré (RT ← 1) et ACC est positionné. Le code de la queue est inchangé (fig. 9)

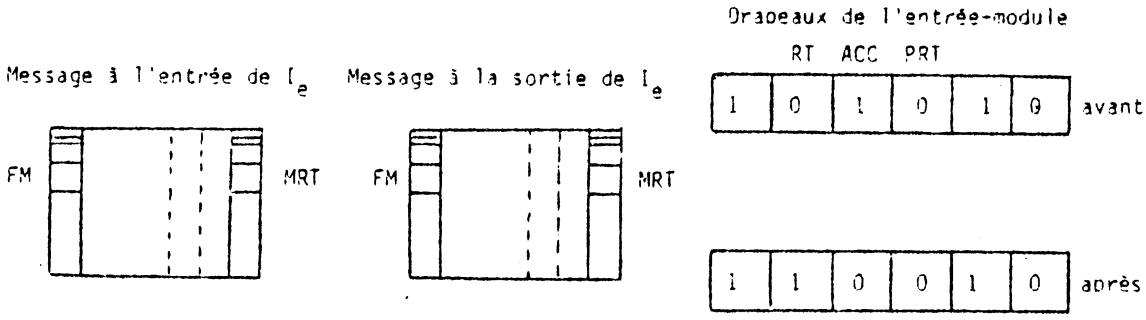



Fig. 9 : action du code "retransmission" sur les drapeaux d'une entrée-module.

b) Drapeau DIS positionné et pas d'erreurs de transmission. Le fait que le drapeau DIS soit toujours positionné signifie que même à cet instant l'entrée du module n'est pas encore à l'état prêt de recevoir. Quand elle le devient, elle restaure DIS, positionne RT et fait une demande de retransmission.

Le code de la queue est changé en code "accusé non définitif" et DIS est toujours positionné (fig. 10).

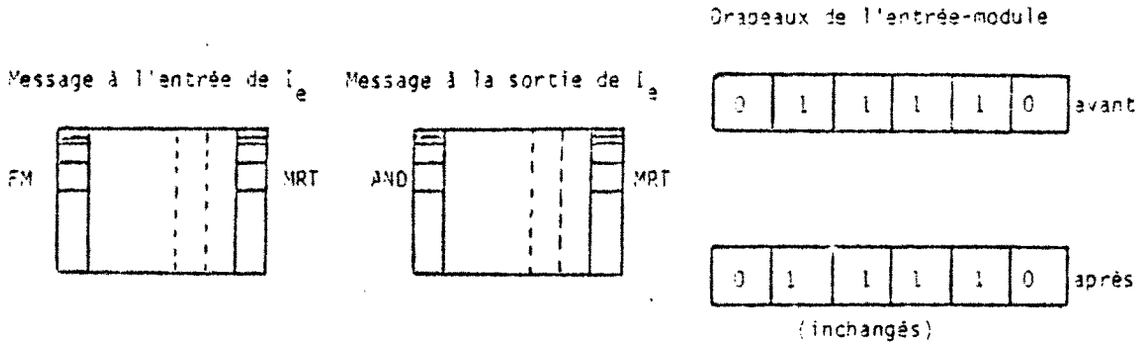



Fig. 10 : actions du code "retransmission" sur les drapeaux d'une entrée-module

c) Drapeaux DIS ou RT positionnés mais erreur de transmission. Les drapeaux DIS et RT sont laissés tels qu'ils sont mais le code opération de la queue est forcé au code"erreur de transmission".

A.2.4.3.2 - Mécanisme au niveau de la sortie lors du retour message

- C'est toujours le même mécanisme standard comme en A.2.4.1.2.

A.2.4.4 - Message de demande de retransmission (MDRT)

C'est un message constitué uniquement d'un entête, émis par une entrée-module pour demander une retransmission de la sortie.

A.2.4.4.1 - Mécanisme au niveau de l'entrée-module

Cette demande est effectuée quand le drapeau DIS est positionné ce qui signifie que la sortie s'est manifestée prête quand l'entrée ne l'était pas encore. Avant l'envoi du message, le module positionne PRT, restaure DIS et positionne RT (fig. 11).

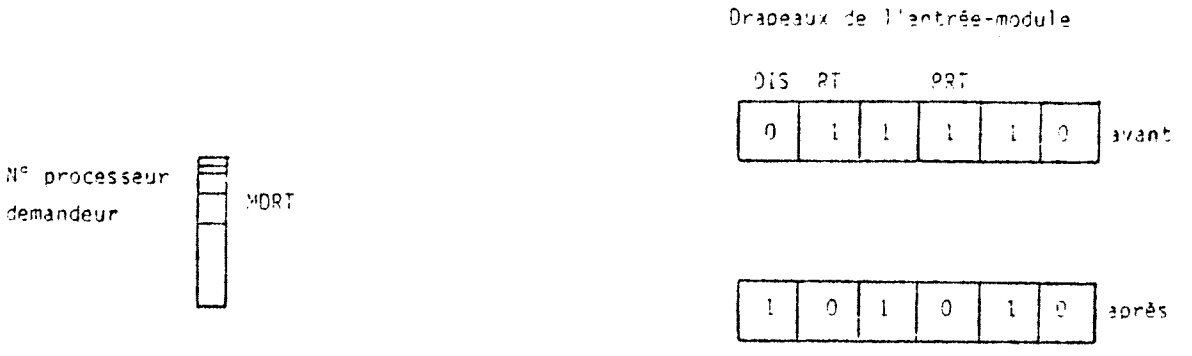



Fig. 11 : actions faites par le module sur les drapeaux d'une entrée-module avant l'envoi d'une demande de retransmission.

A.2.4.4.2 - Mécanisme au niveau de la sortie-module

a) Drapeau DIS positionné.

Ceci signifie que la sortie-module est au courant qu'il y a au moins une entrée-module qui n'a pas été encore actualisée. C'est un état stable et sans ambiguité pour la sortie-module.

A la demande de retransmission le drapeau DIS est restauré (fig. 12). La sortie pourra ainsi détecter les défauts de transmission si aucun des deux drapeaux DIS ou ACC est positionné au bout d'un certain moment.

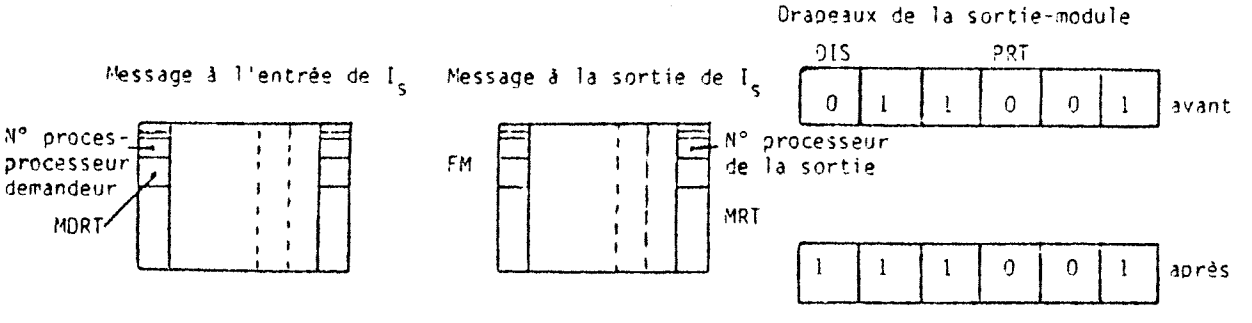



Fig. 12 : actions d'un message de demande de retransmission sur les drapeaux d'une sortie-module.

Le message de demande de retransmission engendre dans l'interface renfermant la sortie-module une émission d'un message de retransmission.

b) Drapeau DIS positionné

Ceci signifie qu'un message de sortie-module n'est pas encore de retour. La demande de retransmission est retirée de l'anneau. Le vecteur d'état reste inchangé.

A.2.5 - Organigrammes des macroinstructions

A.2.5.1 - Macroinstruction ATTENDRE

a) module continu

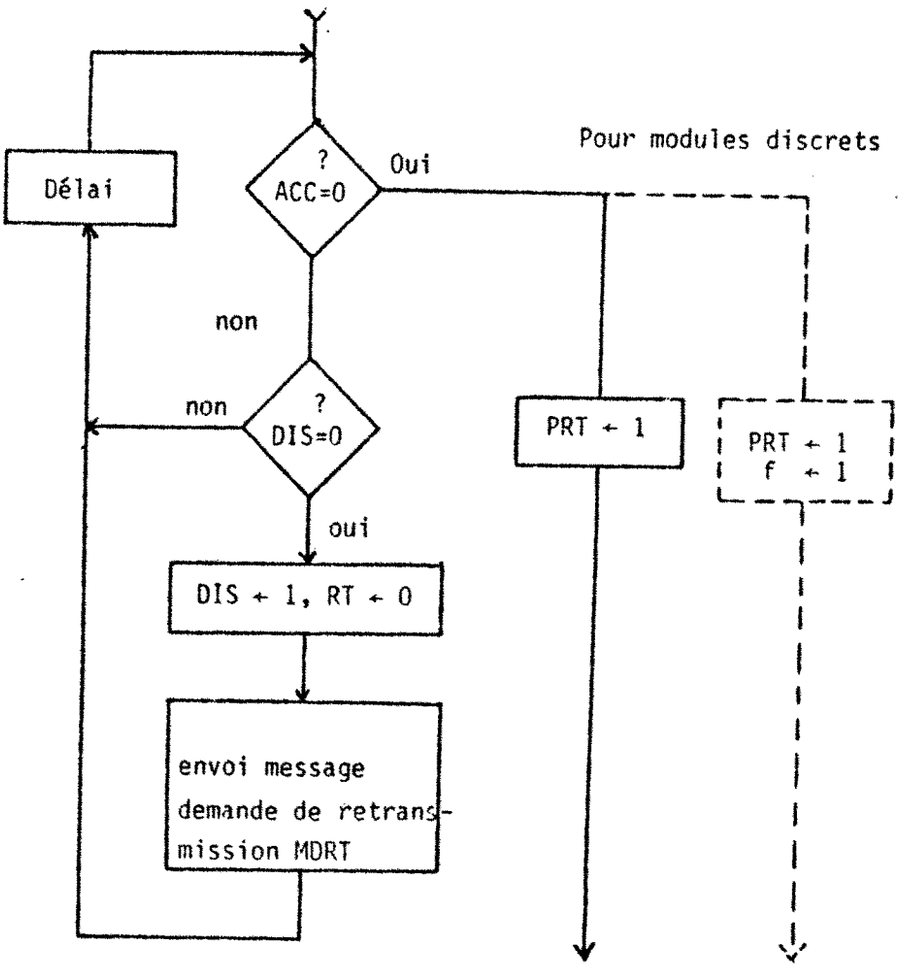



Fig. 13 : organigramme de la macroinstruction ATTENDRE

Les tests des drapeaux ACC et DIS sont effectués par le processeur après une requisition de l'interface. Le délai libère l'interface ce qui l'autorise à analyser les messages et d'effectuer des transferts de données vers la mémoire si c'est nécessaire.

b) Module direct

Même organigramme sauf qu'il y a masquage de l'entrée dès qu'il y a eu transfert d'information avec l'entrée. Le masquage s'effectue en positionnant le drapeau f.

A.2.5.2 - Macroinstruction SORTIR

Les drapeaux ACC et DIS tous les deux à "un" forment un état ambigu pour la commande de sortie. Les transactions ne sont effectuées qu'au bout de n délais. Le nombre n est modelé en fonction du temps de réponse du système de communication et de la durée du délai.

A.2.5.3 - Initialisation des drapeaux

- Modules continus

Chaque fois que le processus du module continu est lancé de nouveau :

. les $PRT_i$ des i entrées sont remis à zéro,
. les $ACC_i$ des i entrées sont mis à un,
. les $PRT_j$ des j sorties sont mis à un,
. les $ACC_j$ des j sorties sont mis à un.

- Modules directs

Même initialisation avec en plus la remise à zéro des drapeaux $f_i$.

A.2.5.4 - Les transactions

Les transactions sont effectuées par la source du message lorsqu'elle ne reçoit aucune information de retour de lui. Une erreur de parité dans l'entête ou dans la queue du message peuvent être les raisons de perdition de l'information de retour.

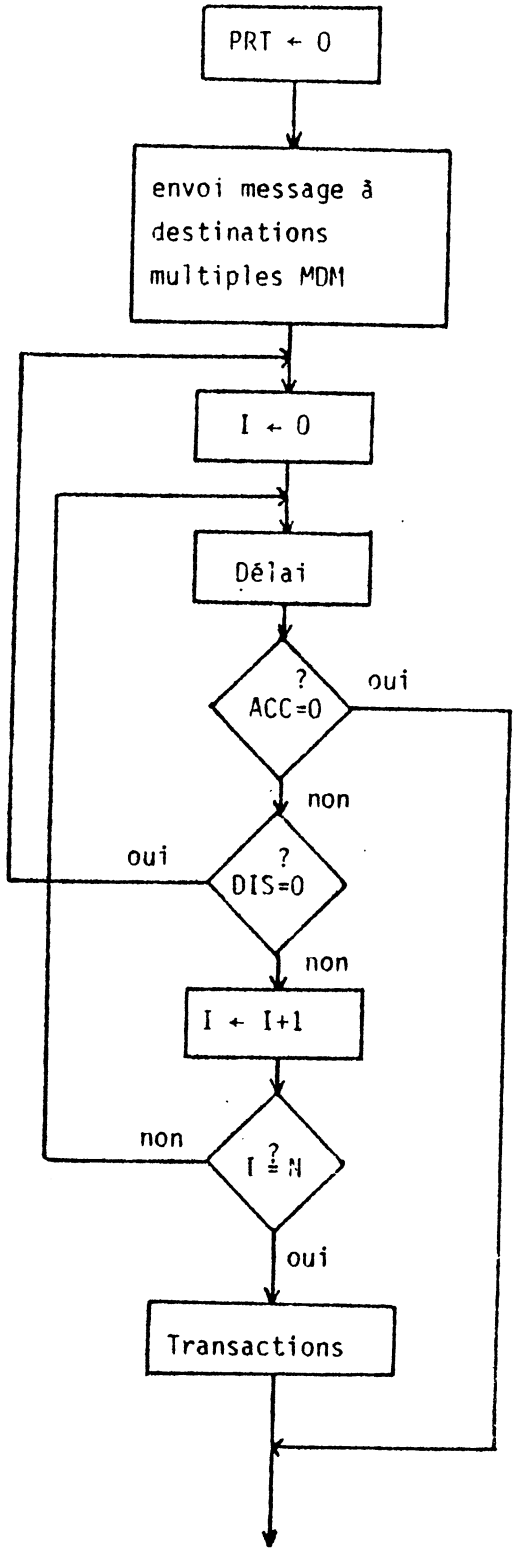



Fig. 14 : organigramme de la macroinstruction SORTIR

Après expiration du délai, la source envoit un message spécial (étiquette ; ;) de longueur fixe qui contient l'étiquette du message perdu et l'information qu'il contenait. Ce message génère une interruption au processeur de chaque module pour comparer l'original avec la copie qui possède chaque module :

- s'il y a inégalité, le processeur positionne dans l'interface le drapeau RT (RT ← 0) pour entrée-module prête et DIS ← 0 pour entrée-module non prête

- s'il y a égalité le processeur n'effectue rien dans le vecteur des drapeaux de l'étiquette.

Après réception de l'accusé du message spécial de transactions, l'interface source retransmet le message initial (avec code retransmission).

On pourrait se demander pourquoi l'interface retransmet le message de nouveau puisque l'information a été communiquée dans le message des transactions ? Sans une retransmission du message initial, la source ne peut pas connaître l'état des entrées dans les autres modules, état qui est résumé dans le champ code opération de la queue.

Remarque : Une transaction ne génère pas une transaction si son message spécial est perdu. Le message de transactions est retransmis tant que l'interface source ne reçoit pas l'accusé définitif de ce message.

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- Z. BINDER, Docteur ès-sciences - Ingénieur C.N.R.S. -
- J.F. LE MAITRE, Directeur du CERT-DERA - TOULOUSE -

Monsieur Ziad O L A I W A N

est autorisé à présenter une thèse en soutenance pour l'obtention du titre de DOCTEUR-INGENIEUR, spécialité "Automatique".

Grenoble, le 14 Septembre 1979

Ph. TRAYNARD
Président
de l'Institut National Polytechnique