INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

THÈSE

Présentée à

l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

pour l'obtention du

DOCTORAT EN INFORMATIQUE

par

Zoubir MAMMERI

Ingénieur

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE NANCY

CONCEPTION D'APPLICATIONS RÉPARTIES EN COMMANDE DE PROCESSUS: UNE APPROCHE PAR LA STRUCTURATION DES DONNÉES

Soutenue publiquement le 24 juin 1985, devant la Commission d'Examen

---

D 136 037803 1

Membres du Jury : MM. J.C. DERNIAME *Président*

P. DEVAUREIX
A.HAURAT
C. IUNG *Examinateurs*
P. KIENER
JP. THOMESSE

1366578031

INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

CENTRE DE RECHERCHE EN INFORMATIQUE DE NANCY

THÈSE (T) 1985 MAMMERI Z.

Présentée à

l'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

pour l'obtention du

DOCTORAT EN INFORMATIQUE

par

Zoubir MAMMERI

Ingénieur

Service Commun de Documentation
INPL
Nancy

CONCEPTION D'APPLICATIONS RÉPARTIES EN COMMANDE DE PROCESSUS: UNE APPROCHE PAR LA STRUCTURATION DES DONNÉES

Soutenue publiquement le 24 juin 1985, devant la Commission d'Examen

Membres du Jury:

MM. JC. DERNIAME *Président*

P. DEVAUREIX
A.HAURAT
C. IUNG *Examinateurs*
P. KIENER
JP. THOMESSE

A MES PARENTS

A MES AMIS

La démarche de remercier m'est toujours apparue embarrassante, tant il est difficile d'exprimer, en quelques mots, ce que l'on ressent sans commettre, par omission, une injustice qui pourrait avoir un effet contraire à celui désiré.
Néanmoins, je voudrais témoigner ma profonde reconnaissance pour tous ceux qui ont contribué, directement ou indirectement, au travail que je présente ici :

Je tiens à remercier :

Monsieur **J.C. DERNIAME**, Professeur à l'Université de Nancy I et Directeur du C.R.I.N., pour toutes les suggestions et critiques constructives qu'il m'a formulées et pour le grand honneur qu'il me fait en acceptant de présider ce jury.

Monsieur **J.P. THOMESSE**, Professeur à l'Institut National Polytechnique de Lorraine, pour m'avoir accueilli, dans son équipe, pour ses encouragements, pour ses conseils et pour toute l'aide qu'il n'a cessée de m'apporter pendant quatre ans. Je lui suis gré de participer à ce jury.

Monsieur **A. HAURAT**, Professeur à l'Ecole de Mécanique et de Microtechnique de Besançon, pour l'attention qu'il porte à mon travail, pour ses critiques qui m'ont permis d'améliorer ce travail. Il me fait un plaisir de siéger dans ce jury.

Monsieur **P. DEVAUREIX**, Ingénieur à la Société C.S.E.E., pour l'honneur qu'il me fait en participant à ce jury.

Monsieur **C. IUNG**, Professeur à l'Institut National Polytechnique de Lorraine, pour ces remarques et critiques, en tant qu'automaticien, et pour son acceptation de participer à ce jury.

Monsieur **P. KIENER**, Ingénieur à la Société SOLLAC, pour l'intérêt qu'il porte à mon travail et pour sa participation à ce jury.

Je remercie toutes les personnes qui ont assuré la réalisation matérielle de cette thèse.

Je dois, enfin, beaucoup et plus encore, aux personnes qui me sont les plus proches, ma famille, mes amis, et qui m'ont supporté et conseillé tout au long de mes études, qu'elles trouvent ici le témoignage de ma plus sincère gratitude.

SOMMAIRE

Page

INTRODUCTION 1

PARTIE I
UNE APPROCHE DE STRUCTURATION DES APPLICATIONS EN CONDUITE DE PROCEDES INDUSTRIELS

CHAPITRE I : LA COMMANDE DE PROCEDES INDUSTRIELS
DEFINITION DES APPLICATIONS
OUTILS ET METHODES POUR LES ACPI : ETAT DE L'ART

I. INTRODUCTION 5

II. LA COMMANDE DE PROCEDES : DEFINITION ET ANALYSE 5
II. 1. Définitions 5
II. 2. Caractéristiques des procédés industriels 7
II. 3. Fonctionnalités des systèmes de commande 8
II. 4. Besoins pour la conception et la réalisation des systèmes de commande 13

III. METHODES ET OUTILS POUR LA COMMANDE DES PROCEDES 14
Une étude bibliographique
III. 1. Au niveau description du cahier des charges 15
III. 2. Au niveau spécification 15
III. 3. Au niveau programmation 16
III. 4. Au niveau implantation 18
III. 5. Conclusion 18

IV. NOTRE APPROCHE DE CONCEPTION DES ACPI 18

V. CONCLUSION 21

CHAPITRE II : DEFINITION DES OBJETS ET PRINCIPES DE BASE POUR LA STRUCTURATION DES APPLICATIONS

I. INTRODUCTION 22

II. DECOMPOSITION DES APPLICATIONS : DETERMINATION DES OBJETS DE BASE 22
II. 1. Comment aborder la conception d'un système de commande ? 22
II. 2. Objectif de la décomposition 25
II. 3. Détermination des types de données et d'actions 25
II. 4. Définition des notions d'actions et de données 27

III. BASES DE DONNEES INDUSTRIELLES : DEFINITION ET PARTICULARITES 28
III. 1. Concept de base de données industrielles (BDI) 29
III. 2. Caractéristiques des BDI et leurs impacts sur la méthode de conception 30

IV. APPROCHE DE CONCEPTION 34
IV. 1. Détermination des objets 34
IV. 2. Détermination des conditions de manipulation des objets 40
IV. 3. Description des actions 41
IV. 4. Prise en compte des anomalies de fonctionnement 42

V. CONCLUSION 43

CHAPITRE III : DESCRIPTION DES APPLICATIONS : ETUDE DE L'ASPECT STATIQUE DES APPLICATIONS

I. INTRODUCTION 44

II. DESCRIPTION DES DONNEES 44
II. 1. Définition d'une donnée 45
II. 2. Utilisations d'une donnée 48
II. 3. Modèle de description des données 52

III. DESCRIPTION DES ACTIONS 54
III. 1. Structure des actions 54
III. 2. Activation des actions 55
III. 3. Modèle de description des actions 58
III. 4. Régies associées aux actions 59

IV. QUELQUES COMPARAISONS 61
IV. 1. Les BDI : Comment sont-elles vues par ailleurs ? 61
IV. 2. Approche orientée communication et notre approche 64
IV. 3. Conception orientée BDI et langages orientés objets 65
IV. 4. B. D. I. et bases de données classiques 66

V. CONCLUSION 67

CHAPITRE IV : DESCRIPTION DES REGLES DE MISE A JOUR DES DONNEES : ASPECT DYNAMIQUE DES APPLICATIONS

I. INTRODUCTION 68

II. PRISE EN COMPTE DE L'ECOULEMENT DU TEMPS 69

III. LES CHANGEMENTS D'ETAT OU EVENEMENTS 70
Définition et critères de prise en compte
III. 1. Notion d'évènements 70
III. 2. Sources d'évènements 71

III. 3. Détection des évènements 73
III. 4. Utilisation des évènements 74
III. 5. Conclusion 77

IV. PRISE EN COMPTE DE L'ETAT DU SYSTEME : LES CONDITIONS 77

V. CONSTRUCTION DES REGLES DE MISE A JOUR 78
V. 1. Les mises à jour périodiques 78
V. 2. Les mises à jour à instants fixes 80
V. 3. Les mises à jour aléatoires 80
V. 4. Règles pour l'utilisation des évènements 81

VI. CONCLUSION 81

CHAPITRE V : TRAITEMENT DE DEUX EXEMPLES : APPLICATION DES CONCEPTS PROPOSES

I. INTRODUCTION 82

II. EXEMPLE DE LA COMMANDE D'UN MOTEUR A COURANT CONTINU 82
II. 1. Détermination de la structure de l'application 83
II. 2. Recherche des conditions de mise à jour des données 88
II. 3. Description de l'application 89

III. EXEMPLE DE LA COMMANDE D'UNE POMPE SOUTERRAINE 94
III. 1. Détermination de la structure de l'application 96
III. 2. Détermination des conditions de mise à jour des données 100
III. 3. Description de l'application 101
III. 4. Remarque 107

IV. CONCLUSION 107

PARTIE II
IMPLANTATION DES APPLICATIONS

CHAPITRE VI : REPARTITION DES APPLICATIONS

I. INTRODUCTION 110

II. REPARTITION DES OBJETS : NECESSITE ET PROBLEMES POSES 110
II. 1. Nécessité de la répartition 110
II. 2. Problèmes liés à la répartition 113

III. APPROCHE POUR LA REPARTITION DES OBJETS 118

III.1. Hiérarchisation des communications 118
III.2. Hiérarchisation des machines 120
III.3. Choix de la répartition 122

IV. CONCLUSION 124

CHAPITRE VII : ETUDE D'UNE ARCHITECTURE POUR L'IMPLANTATION DES S.G.B.D.I.R. (systèmes de gestion des BDI réparties)

I. INTRODUCTION 126

II. HYPOTHESE SUR L'ARCHITECTURE MATERIELLE 126
II.1. Proposition FIP 127
II.2. Type d'architectures matérielles étudiées 128

III. ARCHITECTURE PROPOSEE POUR L'IMPLANTATION DES S.G.B.D.I.R. 129
III.1. Définitions et concepts généraux 129
III.2. Architecture d'un SGBDIL/MTR 132
III.3. Architecture d'un SGBDIL/GFIP 143
III.4. Architecture d'un SGBDIL/MAR 146
III.5. Autres cas de SBBDIL 147

IV. IMPLANTATION DES OBJETS ET CONSTRUCTION DU SGBDIR 148

V. CONCLUSION 150

PARTIE III

CHAPITRE VIII : UN RESEAU DE MICROORDINATEURS POUR LA CONSTRUCTION DES SYSTEMES TEMPS REEL ET REPARTIS

I. INTRODUCTION 153

II. LA SYNCHRONISATION : LE NOYAU TEMPS REEL SCEPTRE 156
II.1. Introduction 156
II.2. Présentation du noyau SCEPTRE 157
II.3. Conclusion 160

III. GESTION DU TEMPS : AGENCE HORLOGERIE 161
III.1. Introduction 161
III.2. Services offerts par l'agence 161
III.3. Structure de l'agence horlogerie 162
III.4. Conclusion 163

IV. LA CONNEXION AU RESEAU LOCAL GIXINET 163
IV.1. Introduction aux réseaux LOCAUX 164
IV.2. Présentation du réseau local GIXINET 165
IV.3. Gestion de la connexion GIXINET - MC 6809 167
IV.4. Conclusion 168

V. LA STATION DE TRANSPORT 169
V.1. Rôle d'une station de transport dans un système distribué 169
V.2. La station de transport réalisée 171

VI. CONCLUSION 175

CONCLUSION DE LA THESE : BILAN ET PERSPECTIVES 176

ANNEXES

A1 : Présentation d'un exemple de procédé industriel complexe et de sa commande : cas de la coulée continue de GANDRANGE 180

A2 : Cahier des charges de la commande d'un moteur à courant continu 185

A3 : Syntaxe du langage de description des applications 188

A4 : FIP : Présentation du projet FIP 193

B1 : Implantation et utilisation du noyau temps réel SCEPTRE 196

B2 : Implantation et utilisation de l'agence horlogerie 208

B3 : Connexion au réseau local GIXINET et utilisation de cette connexion 213

B4 : Implantation et utilisation de la station de transport 224

BIBLIOGRAPHIE

INTRODUCTION

Depuis le début des années 70, les bases de donnée et plus récemment les bases de données réparties constituent une approche nouvelle et intégrée au traitement de volumineux ensembles de données partagées par un nombre important d'usagers.

Mais jusqu'à une date relativement récente (3 ou 4 ans), plus de 90 % des bases de données étaient utilisées à des fins de gestion.

De nos jours, on voit apparaître de nombreux champs nouveaux d'application des bases de données tels que la CAO, le génie logiciel, les applications scientifiques, la bureautique, etc...

L'objectif de ce travail est de contribuer à l'exploration d'un nouveau champ d'application des bases de données qui est la commande des procédés industriels.

En effet, la commande d'un procédé industriel nécessite le stockage d'informations de natures très diverses (mesures effectuées sur le procédé, ordres appliqués au procédé, consignes de fonctionnement, historique du procédé, etc...). Ces informations peuvent atteindre des quantités importantes dans le cas de la commande des procédés complexes (raffineries de pétrole, complexes sidérurgiques, etc...).

Devant le nombre important des informations manipulées et des services offerts par les systèmes de commande de procédés, l'aspect base de données est de plus en plus crucial et ne peut donc être négligé.

Nous cherchons, dans ce travail, à étudier une approche de structuration et de construction des applications en commande de procédés, fondée sur la prise en compte des informations relatives à ces applications.

En partant du cahier des charges décrivant les spécifications fonctionnelles et opératoires d'un système de commande, cette approche consiste à structurer l'ensemble des informations relatives au système à réaliser pour décrire ce système à l'aide de deux ensembles complémentaires:
- un ensemble d'informations (ou de données industrielles)
- et un ensemble de traitements (ou d'actions) pouvant être effectués sur ces informations.

Par analogie aux systèmes d'informations relatifs à la gestion des entreprises, nous retiendrons l'expression "base de données industrielles" (ou BDI) pour désigner l'ensemble des informations relatives à un système de commande de procédé industriel.

Par ailleurs l'approche développée ici consiste à assurer la commande du procédé et le dialogue homme-machine (ie à effectuer tous les traitements) à partir de la BDI.

Les règles d'activation des actions (ou de déclenchement des traitements) sont déduites des règles de mise à jour des données : nous parlerons alors d'actions pilotées par les données.

Les facteurs tels que :
- l'aspect temps réel des ACPI
- la sûreté de fonctionnement exigée

- l'interaction des ACPI avec des phénomènes physiques
- et l'adaptation aux besoins des automaticiens

conduisent à de nouvelles approches pour la conception et l'implantation des BDI, les approches connues dans le cas des bases de données de gestion sont insuffisantes ou non applicables en tant que telles car elles ne prennent pas en compte les facteurs, cités précédemment, qui constituent les caractéristiques essentielles de la commande de procédés industriels.

C'est à ces facteurs que nous nous attacherons, par la suite, pour décrire les ACPI et les BDI qui leur sont associées.

Nous tenterons, dans la première partie de cette thèse, de proposer une approche permettant de mettre en relief les objets constituant une ACPI et de structurer ces objets pour décrire cette ACPI.

Le modèle de description retenu repose sur les concepts d'évènements et de communication et permet ainsi de montrer que ces deux concepts peuvent être combinés pour décrire des ACPI.

La deuxième partie de la thèse sera consacrée aux problèmes d'implantation des ACPI intégrant une BDI. Des solutions aux problèmes tels que la répartition des ACPI et la cohérence des BDI seront proposées.

L'approche d'implantation proposée repose sur la répartition des objets (données, actions...) et sur la hiérarchisation des installations de commande et ce pour minimiser les anomalies de fonctionnement dues au matériel et accroître ainsi le rendement des procédés automatisés.

Nous faisons remarquer dès à présent que le but poursuivi dans ces deux premières parties n'est pas de formaliser ou d'axiomatiser la manière de construire des ACPI, selon notre approche, mais plutôt de proposer un ensemble de concepts et de "conseils" qui servent de guide pendant les phases de conception et d'implantation des applications : les démarches proposées aident à mieux comprendre les problèmes posés et à résoudre ces problèmes de manière efficace.

L'implantation d'une ACPI conduira à l'utilisation d'un certain nombre de logiciels permettant d'assurer la gestion des données, la synchronisation des actions concurrentes et la communication entre les stations de commande dans le cas d'une ACPI géographiquement répartie.

La réalisation de ces logiciels (non effectuée dans ce travail) nécessitera l'utilisation des mécanismes de base de synchronisation, de communication, de gestion de temps, etc...

Le deuxième volet de notre travail, en l'occurrence, la troisième partie, a été consacré à l'étude et à la réalisation d'un noyau de système temps réel qui offre des services implantant ces mécanismes de base.

Ce noyau sera constitué de trois composantes :
- un noyau temps réel pour la synchronisation des entités (ou tâches) constituant une application temps-réel ;
- un module (que nous appelerons "agence") qui assure la communication entre tâches implantées sur des sites répartis et reliés par un réseau local ;
- et un module (ou "agence") qui offre des services relatifs à la gestion du temps (temporisation, etc...).

Les services offerts par ce noyau sont généraux et indépendants des concepts de données et d'actions introduits dans la partie I et permettront l'implantation de ces concepts.

Plan de l'ouvrage

La première partie est consacrée à l'étude d'une approche de structuration et de description d'ACPI en partant du cahier des charges qui décrit les spécifications fonctionnelles et opératoires du système à réaliser.

Cette partie se compose de cinq chapitres (de I à V).
Dans le chapitre I, nous étudierons la commande des procédés, dans son ensemble, afin d'analyser ses caractéristiques et ses besoins ainsi que les fonctions réalisées par un système de commande. Cette analyse nous permettra de mieux cerner les spécificités du domaine qui nous intéresse ici.

Dans le chapitre II, nous proposerons une démarche de décomposition des ACPI pour déterminer les données et les actions composant ces ACPI. L'étude d'une application générale permet de dégager les caractéristiques des données et des actions relatives aux ACPI et fixer ainsi les critères auxquels doit répondre la méthode de conception et d'implantation d'ACPI selon une approche données-actions.

Dans le chapitre III, nous étudierons les attributs liés aux objets (ie les éléments qui permettent de décrire les données et les actions) et proposerons un modèle de description des ACPI. Dans ce chapitre, nous nous intéresserons à l'aspect statique des applications.

Le chapitre IV est consacré à l'étude de l'aspect dynamique des applications ; nous y définirons une démarche et un langage de description des règles de mise à jour de la BDI. Ces règles sont appelées aussi règles d'activation des actions (c'est l'aspect synchronisation des ACPI), nous les décrirons à partir des notions de temps, de changements d'état (ou évènements) et de conditions sur l'état du système.
Les démarches et concepts introduits dans les trois chapitres précédents seront appliqués, dans le chapitre V, pour décrire deux exemples d'applications (la commande d'un moteur à courant continu et la commande d'une pompe souterraine).

La deuxième partie est consacrée aux problèmes d'implantation ; elle se compose de deux chapitres (de VI à VII).

Dans le chapitre VI, nous aborderons la répartition des ACPI pour voir la nécessité et les problèmes posés par cette répartition. Une démarche permettant d'assurer une répartition efficace des objets sera proposée (par efficace, nous entendons respect des contraintes de temps et de sûreté de fonctionnement).

Dans le chapitre VII, nous développerons les grandes lignes de l'architecture d'un système de commande manipulant une BDI pour voir comment envisager la réalisation d'un tel système et l'implantation des objets définis dans la première partie à l'aide de ce système.

La troisième partie est consacrée à l'étude et à l'implantation d'une maquette d'un noyau de système temps réel. Les services offerts par cette maquette seront présentés dans le chapitre VIII, leur implantation est donnée en annexes (annexes B1, B2, B3 et B4).

Nous terminerons ce travail par une conclusion finale dans laquelle nous esquisserons les grandes lignes des extensions possibles de notre travail.

ie signifie "c'est-à-dire"
Les références bibliographiques sont présentées sous la forme /nom, année/

PARTIE I

UNE APPROCHE DE STRUCTURATION DES APPLICATIONS EN COMMANDE DE PROCEDES INDUSTRIELS

CHAPITRE I

LA COMMANDE DE PROCEDES INDUSTRIELS

Définition des applications

Outils et méthodes pour les ACPI : état de l'art

I INTRODUCTION

Pour plusieurs raisons la conception et la réalisation des systèmes de commande de procédés industriels en temps réel sont en pleine évolution.

Tout d'abord la compétition économique que nous connaissons impose à l'industrie de produire en qualité et en quantité pour répondre à la demande dans un environnement très concurrentiel. Pour produire plus et avec une qualité meilleure, pour améliorer les conditions de travail, les industriels ont recours à l'automatisation de leurs outils de production et à l'informatisation de leur gestion.

Par ailleurs, le développement rapide de la technologie des circuits intégrés numériques, et en particulier les microprocesseurs, a permis le remplacement des dispositifs de la commande de procédés classique, c'est-à-dire en logique cablée (ou analogique) par la commande programmée.

Les avantages des techniques informatiques se font surtout sentir au niveau de la mise au point, de la maintenance et de l'évolution des applications. Mais la commande de procédé par calculateur numérique conduit, parfois, à des installations complexes impliquant des investissements considérables qu'il faudra amortir par une exploitation suffisamment longue de ces installations.

Il y a donc nécessité d'une méthode de conception qui facilite la démarche de réflexion du concepteur qui, à partir du cahier des charges de l'application, doit aboutir à la définition des ensembles matériel et logiciel nécessaires au bon fonctionnement de son application.
Nous voudrions, à travers ce chapitre, présenter les particularités de la commande de procédés industriels, ce qui nous permettra de mieux cerner les spécificités du domaine auquel nous nous intéressons ici. Ces particularités devront être à la base de toute méthode de conception et de réalisation des ACPI (applications en commande de procédés industriels).

Nous ferons, ensuite un tour d'horizon des principaux outils et méthodes utilisés pour la conception et la réalisation des ACPI, et enfin nous présenterons notre approche de conception des ACPI.

II LA COMMANDE DE PROCEDES : DEFINITION ET ANALYSE

II.1 DEFINITION

Commençons, tout d'abord, par définir les mots clés du domaine de la commande de procédés industriels, c'est-à-dire :
- la notion de procédé industriel,
- la notion de système de commande,
- la notion de commande de procédé,
- et le degré d'automatisation.

a) Notion de procédé industriel
Un procédé industriel (que nous abrégerons parfois par PI) est un ensemble physique susceptible d'évoluer en fonction d'une variable indépendante appelée temps /BOUDAREL, 67/.

Les procédés industriels couvrent de nombreux champs d'application (pétrochimie, centrales thermiques, centrales nucléaires, chimie, ciment, papier, métallurgie, agro-alimentaire et toutes les industries manufacturées).

b) Notion de système de commande

Un système de commande de procédés est un arrangement d'éléments (actionneurs, capteurs, automates, ... calculateurs) interconnectés de façon à maintenir ou affecter d'une manière pré-connue (ou déterministe) certaines grandeurs physiques /LOWE, 71/.

Les mesures de paramètres du procédé sont effectuées par des capteurs qui transforment la nature physique du signal en une donnée numérique. A l'opposé, les ordres de commande sont transmis au procédé par les actionneurs (ou actuateurs ou effecteurs /LAURGEAU, 79/).

L'organe de commande, quant à lui, élabore les actions à partir des mesures et des consignes selon les lois de commande du procédé.

c) Notion de commande de procédé

La commande en temps réel de procédés industriels, traduction de l'expression anglosaxone "Realtime digital control" désigne l'ensemble d'actions simples ou complexes, allant de l'acquisition d'une mesure à l'élaboration d'algorithmes, de l'utilisation de modèles complexes à la mise en forme d'informations économiques.

Ces actions permettent :
- de commander, une partie ou la totalité du procédé, de manière automatique,
- d'aider les opérateurs dans leur tâche de conduite des machines du procédé et les automaticiens dans leur tâche d'identification, de modélisation,... du procédé.

d) Degré d'automatisation

Le degré d'automatisation d'un système est extrêmement variable selon la nature du procédé, sa complexité, la connaissance qu'on en a et les objectifs assignés au projet.

On distingue trois degrés d'automatisation :

- la surveillance : est une fonction passive vis-à-vis du procédé. L'organe de contrôle acquiert des informations, les analyse et les présente à l'opérateur (journaux de bord, affichage de mesures, etc...)
- le mode "guide opérateur" complète le mode précédent par des traitements plus élaborés et propose aux responsables du site des actions pour conduire le procédé (le calculateur ne réagit pas directement sur le procédé)
- la commande proprement dite a une structure en boucle fermée. Elle correspond à l'automatisation complète de certaines fonctions depuis l'acquisition des informations en passant par leur traitement, pour aboutir à une action sur le procédé. L'opérateur est exclu ici de la conduite : il est chargé des fonctions de surveillance et intervient en cas d'incident pour prendre le pilotage manuel du procédé, éventuellement aidé par un "mode guide opérateur" correspondant à un mode de fonctionnement dégradé du système.

La défaillance des composants du système informatique ne doit pas conduire à l'arrêt de la production. Ainsi, les systèmes de commande sont conçus de manière à fonctionner selon plusieurs modes de marche /KIENER, 84/ (modes automatique, semi-automatique, manuel...).

II 2 CARACTERISTIQUES DES PROCEDES INDUSTRIELS

L'automatisation des procédés implique souvent des investissements importants. Une mauvaise maîtrise du procédé à automatiser peut mener à la catastrophe.

Maîtriser un procédé signifie connaître ses caractéristiques techniques, ses objectifs, ses possibilités d'évolution, etc...

Vu la diversité des procédés industriels, il est difficile de donner un ensemble exhaustif des caractéristiques des procédés. Cependant, les caractéristiques suivantes sont fondamentales lors de la conception et la réalisation d'un système de commande de procédé.

1. La "taille" du procédé,
2. la complexité de la commande,
3. les contraintes de temps imposées par le procédé,
4. la sûreté de fonctionnement exigée,
5. les possibilités d'évolution du procédé,
6. les caractéristiques physico-chimiques du procédé.

Ces caractéristiques influent aussi bien sur le matériel à utiliser que sur le logiciel à mettre en oeuvre.

1. LA TAILLE DES PROCEDES

On rencontre, d'une part, les systèmes très complexes, tels que les raffineries de pétrole et les usines sidérurgiques qui ne peuvent, en général, être fabriqués en série et pour lesquelles les coûts d'étude et de réalisation sont très importants, et d'autre part, des petits systèmes dont les coûts d'étude et de réalisation sont faibles (ou amortis sur un grand nombre d'exemplaires produits).

Les procédés physiques complexes sont souvent le résultat d'associations de plusieurs unités physiques de base.

Ces unités concourrent à la réalisation d'un objectif global, mais à chacune d'elles, peut être associé un objectif local. Elles ne sont pas nécessairement contigues et peuvent couvrir des champs plus ou moins vastes (quelques km2 pour les raffineries de pétrole et les mines de charbon), on parle alors de procédés géographiquement répartis.

2. LA COMPLEXITE DE LA COMMANDE

Le type de services demandé au système de commande varie considérablement d'une installation à une autre.

Il peut correspondre à la surveillance de certaines grandeurs, à la commande entièrement automatique du procédé...

La complexité de la commande dépend de la nature du procédé et du degré d'automatisation souhaité par l'utilisateur et des performances attendues.

3. LES CONTRAINTES DE TEMPS

L'interaction du système de commande avec le procédé fait apparaître des contraintes de temps, parfois très sévères, à respecter.

Le non respect de ces contraintes peut conduire l'application dans un état catastrophique (perte matérielle et/ou humaine).

Ces contraintes, elles aussi, varient considérablement d'une application à l'autre. Elles sont de l'ordre de la seconde ou de la minute (ou même plus) pour les systèmes lents (sidérurgie par exemple) et de l'ordre de 1/100 voire inférieures au 1/1000 de seconde pour les systèmes rapides (centrales nucléaires et moteurs électriques par exemple).

4. SURETE DE FONCTIONNEMENT

Tout système se caractérise par un état de fonctionnement normal qui lui permet d'atteindre l'objectif qui lui a été assigné.

L'état de fonctionnement normal n'est malheureusement pas le seul état que puisse prendre le système.

Les circuits et les machines s'écartent de temps en temps du bon fonctionnement et ne produisent plus ce que l'on attend d'eux.

Les retombées sur le plan économique de ces mauvais fonctionnements sont telles que l'on comprend les hésitations des industriels devant l'introduction d'outils jugés peu sûrs /LADET, 82/.

Les principaux aspects de la sûreté de fonctionnement, à savoir, la fiabilité, la sécurité, et la maintenabilité /HECHT, 76/, /ANDERSON, 81/ doivent être étudiés avec soin lors de l'établissement du cahier des charges pour l'automatisation d'un procédé donné.

5 L'EVOLUTIVITE

Dans le temps, les applications industrielles évoluent par paliers de durées relativement importantes.

L'évolution d'une application peut se traduire par :
- l'adjonction de nouveaux appareils,
- le remplacement de machines déjà existantes par d'autres,
- le retrait de certaines machines,
- l'amélioration de la qualité des services,
- etc...

6 LES CONTRAINTES DE L'ENVIRONNEMENT INDUSTRIEL

Les contraintes d'environnement sont des paramètres importants lors du choix du matériel de commande. Ces contraintes, parfois extrêmement sévères, sont appelées "agressions" /LAURGEAU, 79/ et sont, essentiellement dues :
- à l'environnement mécanique et physique (vibrations, chocs, humidité, température),
- aux parasites électromagnétiques, aux gaz, aux poussières, etc...

II.3 FONCTIONNALITES DES SYSTEMES DE COMMANDE

On distingue trois grandes classes de fonctions réalisées par un système de commande de procédé :
- le dialogue calculateur-procédé,
- les traitements,
- et le dialogue calculateur-opérateurs (automaticiens et autres personnes entrant dans la gestion du procédé).

1 LE DIALOGUE CALCULATEUR-PROCEDE

Le dialogue calculateur-procédé est la caractéristique fondamentale de la commande des procédés.

L'objectif de ce dialogue est :
- d'effectuer des mesures sur le procédé pour connaître son état (températures, vitesses, pressions,...),
- d'envoyer des ordres au procédé pour lui faire subir un changement (ouverture ou fermeture d'une vanne, déplacement par rapport à un axe, etc...).

Ce dialogue se fait par l'intermédiaire de périphériques spécifiques de l'application.

Les informations échangées entre calculateurs et procédés demandent des traitements spécifiques des équipements (conversion analogique-numérique, ou numérique-analogique, etc...).

2. LES TRAITEMENTS

On distingue trois types de traitements :
- le traitement lié à la commande du procédé,
- le traitement lié à la sûreté de fonctionnement,
- le traitement correspondant à des suivis, statistiques et calculs divers.

a) Traitement lié à la commande :
On entend par traitement, à ce niveau, l'exécution d'algorithmes qui permettent d'élaborer les ordres à envoyer au procédé en fonction :
- de l'état courant de celui-ci,
- des mesures effectuées,
- et des consignes d'exploitation.

Exemple d'algorithme
Algorithme de type PID pour la régulation de la température d'un four à 1000°C.

La commande du procédé est assurée par un enchainement d'algorithmes de commande.

Outre l'aspect exécution d'algorithmes, le terme traitement englobe aussi toutes les synchronisations et les échanges entre les divers algorithmes. Par exemple, il faut attendre qu'une pièce soit complètement usinée avant de lancer l'opération de stockage de celle-ci.

Pour l'automaticien, le traitement correspond à l'exécution de "lois de commande" /LADET, 82/.

b) Traitement lié à la sûreté de fonctionnement
La sureté de fonctionnement est le point le plus délicat à résoudre lors de la construction d'un système de commande de PI. En effet, le système a pour mission la commande d'un procédé physique caractérisé par l'existence d'anomalies de fonctionnement.

Le système de commande a pour fonction :
- la détection des anomalies de fonctionnement,
- la prise de décision lorsqu'une anomalie a été détectée (c'est-à-dire "corriger" la panne et continuer à fonctionner ou bien arrêter l'exploitation).

Certaines anomalies graves impliquent l'arrêt total ou partiel de l'exploitation du procédé par le système automatique. Pour éviter l'arrêt complet de l'exploitation du procédé dès l'apparition de la première anomalie, le système de commande est généralement conçu de manière à fonctionner sous diverses formes (on parle de systèmes reconfigurables et autorisant le fonctionnement en performances dégradées).

L'efficacité d'un système de commande dépend, en grande partie, de sa tolérance aux pannes ou encore du degré de sûreté de fonctionnement qu'il offre.

c) Traitement correspondant aux suivis divers et statistiques diverses

Le système de commande stocke des informations (concernant la commande courante, l'histoire du procédé et sa commande et même le personnel) en vue d'aider les opérateurs (pour la conduite du PI) et les automaticiens (pour l'optimisation de la commande) à accomplir leurs tâches.

Les traitements effectués à ce niveau concernent :
- les suivis,
- et les calculs et statistiques divers.

Les suivis

Ce sont essentiellement :
- les suivis de produits (quantité, qualité des produits fabriqués),
- le suivi de l'installation (caratéristiques des divers équipements : vitesses maximales, tensions minimales de ces équipements, etc...),
- et le suivi du personnel (par exemple connaître, à tout moment, le poste occupé par chaque individu).

Statistiques et calculs divers

Lors d'une première automatisation d'un procédé, celui-ci, est en général mal ou peu connu des automaticiens, et la première version de la commande n'est pas optimale. Les automaticiens essayent, de manière continue, d'améliorer la commande.

Le système de commande est conçu de manière à stocker des données (relatives aux opérations appliquées au procédé, aux mesures effectuées, à l'utilisation des matières premières, etc...) qu'il met à la disposition du personnel de commande.

L'objectif de ces calculs (et/ou statistiques) est de permettre au personnel de commande de savoir si l'exploitation du procédé est optimale ou non (dans le cas contraire ces calculs leur permettent de déterminer les paramètres ou les équipements à changer). En termes d'automatique, les calculs et suivis divers sont utilisés pour l'identification et la modélisation du procédé et pour l'optimisation de la commande.

Si les suivis et calculs divers ne sont pas importants pour la commande des procédés simples, ils sont de plus en plus demandés pour la commande des procédés complexes.

3. LE DIALOGUE OPERATEUR-CALCULATEUR

Le développement du dialogue opérateur-calculateur a suivi celui de l'automatisation des procédés. Peu différent au départ du dialogue utilisé en informatique classique, il s'est, peu à peu, distingué pour réclamer de nos jours :
- des organes de dialogue adaptés aux besoins des opérateurs.

Il existe actuellement beaucoup de moyens rendant la tâches des opérateurs de plus en plus simple et efficace /ROUSSE, 77/ (utilisation de synoptiques, courbes, tableaux...).

Ce dialogue est devenu possible grâce à l'apparition de périphériques spéciaux (écrans couleur, tables traçantes, etc...).
- et des langages de dialogue simple d'utilisation pour des personnes non initiées à l'informatique (le langage de dialogue se résume parfois à l'utilisation de touches d'un clavier spécialisé).

L'objectif du dialogue opérateur-calculateur est :
- de permettre aux opérateurs d'agir sur le procédé :
  - .initialiser et arrêter l'exploitation,
  - . fixer les consignes de production,
  - . changer les modes de fonctionnement,
  - . etc...
- et de permettre aux opérateurs (ou autres personnes) de :
  - . connaître l'état du procédé ou de l'installation de commande,
  - . connaître l'historique du procédé et de sa commande (en utilisant les statistiques et suivis nécessaires).
  - . etc...

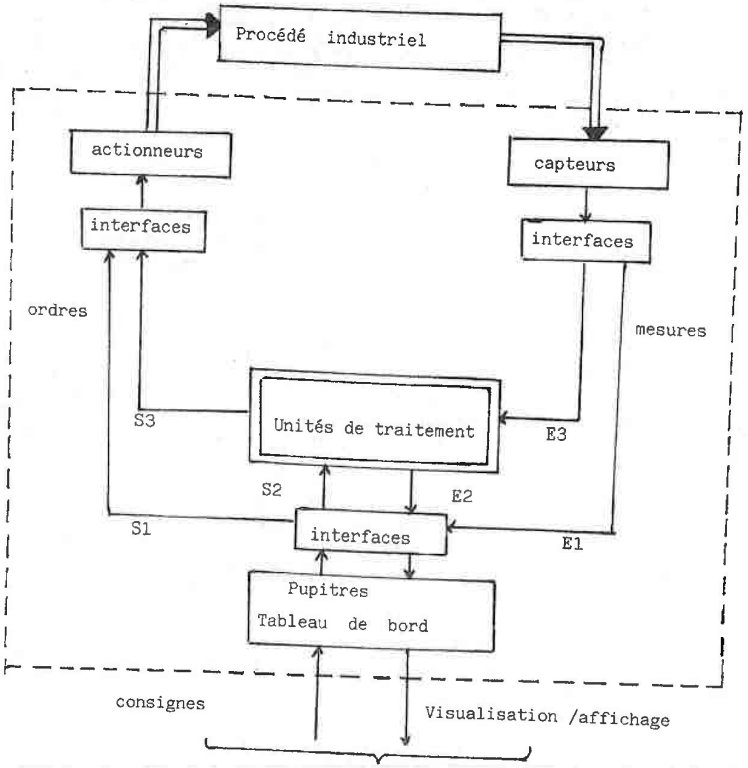
4 REMARQUE

La commande d'un procédé industriel ne doit pas reposer uniquement sur l'installation d'un système automatique de manière à permettre une exploitation du procédé, en mode manuel, si possible, en cas d'indisponibilité du système automatique.

Ce type d'exploitation suppose l'existence, au niveau du système de commande, de dispositifs permettant à l'utilisateur de commander directement le procédé (avec ou sans l'aide du calculateur). Il s'agit évidemment des systèmes non-embarqués pour lesquels une reprise en manuel est possible.

5. SCHEMA RECAPITULATIF

Le schéma de la figure ci-dessous permet de montrer les différents constituants d'un système de contrôle/commande de procédé



Architecture d'un système de commande de PI

S1 = (S1$_1$,..,S1$_{K1}$) représentent les ordres émis de l'opérateur vers le procédé (sans passer par les unités de traitement).

E1 = (E1$_1$,....,E2$_{K2}$) représentent les informations à destination de l'opérateur (sans passer par les unités de traitement).

E2 = (E2$_1$,..,E2$_{K3}$) représentent les compte-rendus envoyés, à l'opérateur, par le système informatique.

S2= (S2$_1$,..,S2$_{K4}$) représentent les consignes envoyées, par l'opérateur, au système informatique.

S3 = (S3$_1$,..,S3$_{K5}$) représentent l'ensemble des ordres émis, par le système informatique, vers le procédé.

E3 = (E3$_1$,..,E3$_{K6}$) représentent les mesures en provenance du procédé et à destination du système informatique.

NB : on utilise par la suite le terme "opérateur" pour désigner parfois les opérateurs de conduite, les automaticiens, ... à la fois.

6. QUELQUES EXEMPLES DE SYSTEMES DE COMMANDE

Depuis quelques années, un certain nombre de systèmes de commande sont devenus opérationnels dans divers secteurs (pétrochimie, métallurgie, cimenterie,...). Parmi ces systèmes, on peut citer :

- pour les grands systèmes :
  - . ICS 16 et MASC 16 de la Sems /Sems/
  - . Teleperm M de la Siemens /Siemens/
  - . Provox de Fisher Control /Fisher/
  - . Micro-Z de Control Baily /Bailey/
  - . Master Asea d'Asea /Asea/

- pour les petits systèmes :
  - . FICS10 d'Eurotherm /Eurotherm/
  - . MICRO-DCI de Fisher et Porter /Fisher et Porter/.

La liste des systèmes opérationnels est certainement plus longue, nous avons retenu, pour mémoire, ces quelques systèmes utilisés en France.

II.4 BESOINS POUR LA CONCEPTION ET LA REALISATION DES SYSTEMES DE COMMANDE

La construction d'un système de commande de procédés fait intervenir trois types de compétences complémentaires.
- l'instrumentation,
- l'automatique,
- et l'informatique.

En effet, cette construction passe par une étape d'analyses de type automatique qui permet de dégager la structure automatique de la commande, ie la nature et les spécificités de l'application.

C'est à partir de cette structure que l'informatique permet de réaliser le système de commande dans ces composantes matérielle et logicielle.

Une telle réalisation n'est possible que si l'on dispose des outils permettant d'exprimer tous les concepts relatifs à une ACPI.

Pour tenir compte des caractéristiques des procédés évoquées précédemment on devra pouvoir assurer :
- l'expression des algorithmes de commande,
- l'expression des contraintes de temps et des actions à entreprendre au cas où ces contraintes ne sont pas respectées,
- la programmation d'opérations d'entrées/sorties sur des périphériques de natures très variées.

- l'expression du parallélisme liée à la nature du procédé,
- la synchronisation des différentes fonctions (ou tâches) de l'application,
- la communication d'informations entre les différentes fonctions,
- l'expression des conditions d'éxécution des fonctions,
- et l'expression de la répartition de l'application.

Le premier besoin (c'est le besoin de base) est l'utilisation d'un langage temps réel permettant d'exprimer les concepts cités ci-cessus.

Par ailleurs, la complexité croissante des systèmes de commande nécessite, de plus en plus, l'emploi d'un guide méthodologique de conception d'application.
La méthodologie utilisée doit faciliter la démarche de réflexion et de décision menée par le concepteur qui, à partir du cahier des charges d'une application, doit aboutir à la définition des ensembles matériels nécessaires au bon fonctionnement de l'application.

La durée de conception dépend, et les expériences le montrent, de la manière dont le cahier des charges a été rédigé. Un cahier des charges clair, complet, et sans ambiguités facilite la phase de spécification de l'application. C'est pour cela qu'il est aussi indispensable que le cahier des charges soit rédigé avec l'emploi d'un guide méthodologique.

De plus, la construction d'ACPI conduit généralement à des logiciels de grande taille.

Construire ces logiciels représente un investissement important qu'il faudra amortir par une durée d'exploitation suffisamment longue.

L'aspect évolutivité du logiciel et du matériel devra être pris en compte dés l'étape de spécification.

La méthode de conception utilisée devra aboutir à une application modulaire, fiable... et simple à maintenir. Les besoins à ce niveau ne sont pas nouveaux. Ils relèvent du domaine du génie logiciel en général /ZELKOWITZ, 78/, /BOEHM, 82/, /SAKHONN,80/.

Le deuxième besoin se situe donc au niveau de la démarche de conception. Le choix d'une méthode de conception influe considérablement sur la qualité de l'application.

Le travail que nous présentons ici se situe dans cette optique. Il s'agit d'une méthode de conception d'ACPI qui cherche à satisfaire les besoins cités précédemment.

III METHODES ET OUTILS POUR LA COMMANDE DE PROCEDES

Une étude bibliographique

Nous venons de signaler que la construction des systèmes de commande de procédés implique souvent des investissements importants.

Durant les deux dernières décennies, un certain nombre de travaux ont été développés pour harmoniser l'activité de construction des systèmes informatiques : faciliter la tâche du concepteur, permettre l'obtention d'applications "robustes" et diminuer les coûts de conception et de réalisation des applications.

Nous voudrions tracer ici les grandes lignes des travaux relatifs à la commande de procédés en nous situant aux différents niveaux suivants :
- la description du cahier des charges,
- la conception (ou la spécification),
- la programmation,
- et l'implantation des applications.

III.1 AU NIVEAU DESCRIPTION DU CAHIER DES CHARGES

Les fournisseurs-concepteurs sont souvent confrontés aux imprécisions et sous-entendus au niveau de la rédaction du cahier des charges. Cela est particulièrement vrai pour les termes du jargon technique qui peuvent être pour un non-initié soit totalement hermétiques, ce qui est un moindre mal, soit interprétés à contre-sens, ce qui est une catastrophe.

La nécessité de méthodologies et d'outils normalisés pour la spécification du cahier des charges des automatismes se pose avec de plus en plus d'acuité, D'une part, à cause de l'évolution de la demande des automatismes complexes, d'autre part à cause de l'évolution des exigences quant à la qualité de service que ces automatismes doivent offrir : en particulier, pour ce qui concerne la sécurité de fonctionnement.

La clarté et la rigueur avec laquelle les objectifs du cahier des charges sont définis prennent dès lors une importance capitale.

Un pas important a été franchi, dans la rédaction des cahiers des charges, grâce à un certain nombre de travaux en France comme à l'étranger, donnant naissance à des outils dont les plus importants sont le GRAFCET du groupe AFCET "Systèmes logiques" en France /BLANCHART, 78/, /BROCHET et al, 79/, /ROUCHOUSE, 83/.. Et les réseaux de Pétri à l'étranger /PETERSON, 77/ (repris dans /BRAMS, 83/, /MOALLA et al, 78/, /THIELLIEZ, 78/).

L'utilisation de ces outils permet de préciser clairement les objectifs du projet d'automatisation et facilite les phases ultérieures de la conception du système de commande.

III.2 AU NIVEAU SPECIFICATION

Les systèmes complexes posent des problèmes importants et nouveaux d'analyse et de décomposition des applications. S'il est possible, pour les systèmes de dimension réduite, de passer directement du cahier des charges à la programmation, il n'en est pas de même pour les systèmes complexes.

Comme pour le cahier des charges, l'utilisation d'un guide méthodologique pour l'analyse et la spécification d'une application se fait sentir de plus en plus.

Une méthodologie de conception pour les applications temps-réel doit permettre d'aboutir, d'une manière efficace, à la réalisation répondant à toutes les spécifications du cahier des charges.

Le processus de développement de la solution peut être vu comme une suite ordonnée (comme le préconise Calvez /CALVEZ, 82/) d'activités qui produit une description,de plus en plus précise,du système à réaliser.

Chaque étape de ce processus correspond à la détermination d'une partie de la solution, à partir de spécifications qui sont les résultats des étapes précédentes. La solution finale consiste en un ensemble de fonctions. Le découpage de l'application en fonctions tient compte des facteurs efficacité, généralité et sûreté de fonctionnement de l'application /KAISER et al, 78/.

Voyons maintenant sur quoi reposent les méthodes préconisées dans la littérature.
Depuis une dizaine d'années, un ensemble de méthodologies pour la conception, le développement et la vérification de programmes séquentiels ont été proposées.

Ces méthodologies sont essentiellement basées sur :
- la décomposition descendante de la solution d'un problème par raffinements successifs/ WIRTH, 71A/.
- et la décomposition modulaire par laquelle chaque module est défini comme l'association d'objets et de fonctions manipulant ces objets /PARNAS, 72A, 72B et 75/.

Ces méthodologies regroupées sous le vocable "programmation modulaire et structurée" /ARSAC,77/, /DERNIAME, 74/, /DIJKSTRA, 72/ ont constitué un apport important dans la production de programmes plus fiables et plus lisibles. Elles ont permis de définir des outils pour la vérification et la correction des programmes, mais elles étaient limitées aux programmes séquentiels.

Plus récemment, de nouvelles méthodologies proposent des réponses aux problèmes liés au temps réel et à la commande de procédés. Des outils et des démarches méthodologiques ont été proposés pour la spécification et la programmation d'applications en conduite de procédés /MENDLEBAUM, 76/, /LECALVEZ,79/, /THOMESSE, 80/ , /LADET 82/ , /CALVEZ, 82/, /BENMAIZA, 84/, /ZAKARI,84/.

Ces outils et méthodes, plus ou moins complémentaires les unes des autres, sont basées sur les concepts de décomposition descendante, de module et de tâche temps-réel. D'autres méthodes, telles que Sadt /ROSS, 77/ et Mascot /JACKSON, 76/ proposent une description des applications à l'aide de diagrammes (ou blocs) fonctionnels qui par raffinements successifs fournissent le structure détaillée de l'application. A ces méthodes sont associés des langages de description d'applications.

III.3 AU NIVEAU PROGRAMMATION

Le développement de la commande de procédés industriels s'est accompagné d'une floraison de langages de programmation d'ACPI que l'on peut regrouper en trois classes :
- les langages d'applications (ou progiciels),
- les langages temps réel,
- et les langages de structuration.

a) Les langages d'applications ou progiciels
La première catégorie de langages connus sous le terme anglo-saxon de "packages" apparaît au cours des années soixante et obtint un grand succès auprès des utilisateurs.
Les langages d'application prennent en compte les besoins des automaticiens, ne nécessitant pas une connaissance de l'informatique et respectent les méthodes utilisées par les opérateurs /PIKE, 70/ d'où leur appellation "fill in the blancks languages".

Ces packages permettent la réalisation des fonctions suivantes :
- acquisition et traitement des données en provenance du procédé,
- commande du procédé,
- et dialogue homme-machine.

Les opérateurs décrivent la commande du procédé par un ensemble d'enchaînements pré-définis et contenus dans une bibliothèque. ICS de la télémécanique (et repris par la SEMS) /ICS,75/ est, de nos jours, un exemple de tels packages.

b) Les langages temps-réel
Les langages temps-réel se sont développés à leur tour au cours des années 70 pour répondre aux besoins d'applications devenues plus complexes. Ces langages sont basés sur la notion de tâche temps-réel.

Une tâche a une signification physique pour l'opérateur.
La programmation d'une tâche signifie pour l'utilisateur du langage :
- programmer les algorithmes de calcul de la tâche,
- programmer la synchronisation et la communication avec les autres tâches de l'application.

La modification d'une tâche peut remettre en cause les relations existant entre les tâches et conduit parfois à la redéfinition de ces relations.
Les langages temps-réel peuvent être regroupés en trois catégories :
- les langages qui sont des extensions de langages classiques tels que FORTRAN temps-réel faisant appel au système superviseur pour la synchronisation et la communication inter-tâches ;
- les langages qui intègrent les concepts de tâche et de synchronisation tels que LTR /PARAYRE, 84/, Pearl /ELZER, 75/, Coral /WEBB, 75/ etc...
- et les langages basés sur la communication entre tâches tels que Gypsy /AMBLER, 77/, DP /HANSEN, 78/, Plits /FELDMAN, 79/, CSP /HOARE, 78/ et Ada /ICHBIAH, 79/.

Tous ces langages ont l'inconvénient, d'être plus proches du langage d'informaticiens que du langage d'automaticiens.

Pour pallier cet inconvénient, en plus des langages de structuration une nouvelle "race" de langages temps-réel est en pleine métamorphose. Ces langages, regroupés sous le vocable "langages pour la productique" ou "langages de programmation des robots", offrent des outils adaptés à la commande des procédés (plus particulièrement à la commande des robots) et permettent une expression aisée des commandes à appliquer aux robots. Par exemple, on peut décrire directement des mouvements d'un robot à l'aide du langage LMAC /HAURAT,82/ par des constructions du style "fermer pince", "déplacer robot...", "lier pince à robot", etc...

D'autres exemples de cette catégorie de langages, tels que LAMA, RAPT et AUTOPASS, sont présentés dans /ROUSSEAU, 83/.

c) Les langages de structuration

Parallèlement aux langages temps-réel, se sont développées des études sur la structuration des applications et on a vu naître des outils utiles à cette structuration. Tous ces travaux reposent sur la distinction entre partie contrôle et partie opérative d'une application et mettent l'accent sur les mécanismes de synchronisation et de communication /MENDELBAUM, 76/, /THOMESSE, 80/, /LADET, 82/,/CALVEZ, 82/, /BENMAIZA, 84/, /ZAKARI, 84/.

Dans ces travaux, la description d'une application consiste en :
- la définition des actions dans un langage tel que PASCAL,
- la définition du schéma de contrôle qui précise les différents enchaînements de ces actions par un jeu de liens "évènements-actions", "condition-évènement-action".

La notion de modularité /PARNAS, 72A et 72B/ se retrouve au niveau de la partie opératoire (l'application est décrite à l'aide d'un ensemble de modules). Les liens entre les modules sont ceux relatifs à la synchronisation et à la communication.

Par rapport aux langages temps réel, les langages de structuration se veulent plus proches du langage des automaticiens, particulièrement au niveau de la partie contrôle de l'application. C'est ainsi qu'on trouve dans les travaux de THOMESSE , LADET et BENMAIZA des phrases du style :
"sur détection de défaut, avertir l'opérateur"
ou "sur dépassement du seuil haut de la cuve, fermer la vanne"
ou encore "à 12 heures arrêter l'exploitation".

II.4 AU NIVEAU IMPLANTATION

Le microprocesseur, nous l'avons souligné, au début du chapitre, est un outil précieux dans le développement de la commande de procédés.

Grâce à son faible coût et à sa souplesse d'utilisation, le microprocesseur permet la construction de systèmes multimachines pouvant satisfaire les contraintes de temps imposées par les procédés les plus complexes.

Par ailleurs, le microprocesseur conduit aussi à la notion de réseaux locaux (/DANG, 83/, /GIEN, 79/, /NAFFAH, 80/, /SOLMAN, 80/)qui sont de nos jours très sollicités pour assurer la communication entre les divers sites constituant un système de commande réparti.

La microélectronique se trouve ainsi le principal stimulus dans le développement de la commande de procédés.

III.5 CONCLUSION

En conclusion de cette revue rapide de la littérature se rapportant à la fabrication des ACPI, il apparaît que les idées et principes bien développés nous fournissent une bonne base de réflexion, mais aucune recherche ne traite complètement du problème de conception orientée utilisation de base de données industrielles que nous développons dans ce travail.

IV NOTRE APPROCHE DE CONCEPTION DES ACPI: IDEE FONDAMENTALE

Pour comprendre notre approche de conception des ACPI, il est nécessaire de rappeler qu'à toute organisation (une entreprise, un hôpital...) correspond un système d'informations.

Un système d'informations est un ensemble d'informations et de moyens de traitement incluant collecte, mémorisation, restitution, communication... d'informations /DUFOURD, 80/.

En informatique de gestion,on a coutume de modéliser la connaissance qu'on a sur une partie ou la totalité d'une organisation par une base de données. La manipulation d'une telle base de données est assurée par un ensemble d'opérations fixées en fonction des objectifs assignés au système informatique.

Les méthodes de conception de systèmes d'informations proposées /DUFOURD, 80/, /FOUCAULT, 82/ avaient pour objectifs :
- la définition des données,
- la définition des traitements à effectuer sur ces données,
- et la définition de la dynamique du système (à l'aide du concept d'évènement) pour coordonner les divers traitements à effectuer sur la base de données.

Aussi des approches de conception et d'implantation des bases de données ont vu le jour et nul ne peut ignorer le succès que les bases de données ont eu auprès des utilisateurs.

Ce succès est dû, en grande partie, à l'attention accordée aux informations:
- plus elles sont cohérentes, mises à jour à temps,... et plus les résultats de leur exploitation concordent avec les souhaits de l'utilisateur ;
- une modélisation en termes de données et de traitements facilite la compréhension du rôle du système automatique vis-à-vis de l'organisation (elle est naturelle pour une personne non informaticienne).

L'idée développée ici, est de chercher à étudier les informations relatives à un procédé industriel afin de proposer une approche de conception et de réalisation des ACPI analogue à celle des applications de gestion.

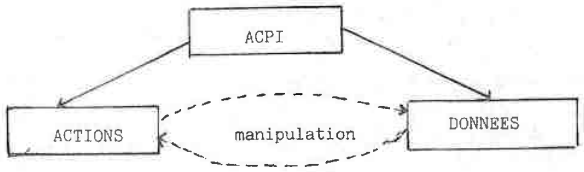
Puisque les bases de données sont bien adaptées à la gestion pourquoi ne le seraient-elles pas pour les ACPI ? et si oui, comment les adapter aux particularités des ACPI ?

C'est à ces deux questions que nous consacrerons la première et deuxième partie de la thèse.

Approche de conception
Un système de commande de procédé industriel a pour mission la commande d'un phénomène physique et le dialogue avec les opérateurs.

Cette mission n'est possible que si le système dispose des traitements et des informations nécessaires à chacun des services qu'il doit fournir.

Il peut être alors considéré comme étant composé de deux ensembles :
- les informations (ou les données)
- et les traitements (ou les actions).



C'est là le premier point de notre approche ie que nous faisons la séparation entre les données et les actions qui les manipulent.

L'ensemble des données représente, à tout instant, l'état du procédé, de la commande du procédé et de l'installation de commande.

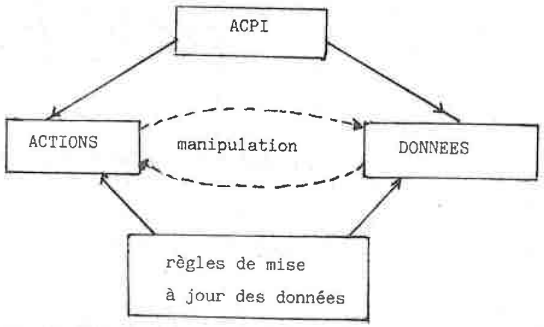
Comme le procédé évolue dans le temps, les données elles aussi devront évoluer (ou être mises à jour) pour tenir compte de l'évolution du procédé.

La mise à jour d'une donnée ne peut être effectuée que si les actions associées à cette donnée ont été activées.

Il est alors nécessaire de préciser les conditions de mise à jour des données (ou encore les conditions d'activation des actions).

Le deuxième point de notre approche consiste à piloter les actions par les données c'est-à-dire qu'à chaque donnée seront associées des règles de mise à jour, et l'ensemble des règles de mises à jour des données sera utilisé pour activer les actions.

Le concept de "pilotage" des actions par les données est à la base des "systèmes guidés par le flux de données" /DENNIS, 75/.



Ces règles sont déduites du cahier des charges en considérant l'évolution, dans le temps, du procédé et de sa commande, par exemple :
. tous les 50 ms, capter la température T,
. sur demande de l'opérateur, afficher le synoptique S.

Une fois ces ensembles définis, l'étape suivante dans la conception de l'ACPI, consiste à déterminer la structure interne de chaque action (ie déterminer les algorithmes composant l'action).

REMARQUE
Nous avons présenté dans le paragraphe III de ce chapitre, les principaux travaux relatifs à la commande de procédés, nous signalons dès à présent la différence fondamentale entre ces travaux et le nôtre.

Ces travaux sont basés sur :
- la prise en compte de l'évolution du procédé et de sa commande,
- et sur les échanges d'informations entre les divers modules de l'application.

Nous prenons ici en quelque sorte " le chemin dans le sens inverse" puisque notre travail consiste à partir de l'ensemble des données (comme en informatique de gestion) pour déduire les actions et les aspects de synchronisation et de communication entre les diverses actions. Nous reviendrons sur cette différence plus longuement dans la suite de ce travail.

V CONCLUSION

L'étude menée le long des paragraphes précédents nous a permis de faire un tour d'horizon sur la commande de procédés industriels.

En effet, nous avons tenté de dégager les particularités de ce domaine d'activité. Parmi ces particularités, les quatre suivantes vont guider nos choix dans les prochains chapitres :
- la sûreté de fonctionnement,
- l'aspect temps réel,
- la répartition,
- et l'évolutivité des applications.

A travers une étude bibliographique, nous avons résumé les principaux travaux relatifs à la commande de procédés (nous aurons l'occasion de citer ces travaux par la suite).

Nous avons présenté le principe de notre approche de conception des ACPI ; laquelle approche est basée sur la séparation entre les données et les actions et le contrôle des actions par les données.

REMARQUES

1. Pour illustrer les concepts que nous introduirons, par la suite, nous ferons référence à un exemple de procédé industriel qui est une coulée continue (son principe de fonctionnement est décrit dans l'annexe A1).

Cet exemple est de taille suffisante pour permettre d'illustrer, sur un cas réel, les concepts qui seront proposés.

2. Pour décrire des règles syntaxiques, nous utiliserons les notations de BNF (ie forme normale de Backus).

CHAPITRE II

DEFINITION DES OBJETS ET PRINCIPES DE BASE POUR LA STRUCTURATION DES APPLICATIONS

I INTRODUCTION

Nous avons vu, au chapitre précédent, la commande de procédés industriels, dans son ensemble, et dégagé les principales fonctions d'un système de commande (que nous appelons aussi application en conduite de procédés industriels abrégé par ACPI).

Nous présenterons, dans ce chapitre, les grandes lignes de la structuration d'une ACPI selon une approche "données/actions".

Pour aboutir à une démarche de description d'ACPI sous forme de données, d'actions et de règles de manipulation des données, nous procéderons comme suit :
- premièrement nous décomposerons une ACPI (qui couvre les cas usuels de la commande de procédés) pour dégager les principaux types de données et d'actions pouvant constituer une ACPI;
- deuxièmement nous étudierons les particularités des données afin de déterminer les critères à prendre en compte pour :
  . décrire les objets (données et actions),
  . décrire les règles de mise à jour des données,
  . et implanter l'ACPI.
- troisièmement nous proposerons une démarche de conception d'ACPI orientée utilisation de BDI. Cette démarche est abordée dans ce chapitre mais son développement se poursuivra dans les prochains chapitres.

Hypothèse de travail

Le travail que nous présentons ici consiste en la structuration et la description d'ACPI selon une approche données/actions. Notre point de départ est un cahier des charges complet , en ce sens qu'il décrit entièrement :
- l'aspect fonctionnel,
- et l'aspect opératoire des ACPI traitées.

Cependant la pratique montre qu'un cahier des charges est, souvent, incomplet. Nous supposons que, grâce aux questions que se posera le concepteur, pendant la phase de structuration, le cahier des charges peut être complété.

Le travail que nous présenterons, par la suite, considère que le cahier des charges est complet et transforme ce cahier des charges pour aboutir à une ACPI décrite en termes de données et d'actions.

II DECOMPOSITION DES APPLICATIONS : DETERMINATION DES OBJETS DE BASE

II.1 COMMENT ABORDER LA CONCEPTION D'UN SYSTEME DE COMMANDE ?

Une première approche, pour l'étude d'un système de commande, consiste à considérer le système comme un tout qui n'a de relations qu'avec son environnement (ie le procédé et les opérateurs).

Mais les difficultés, pour la maîtrise d'une approche globale, apparaissent, dès que les systèmes à étudier deviennent complexes. En effet l'évolution de la commande, de ses buts et de ses méthodes, s'est traduite par une augmentation de la complexité des modèles mis en jeu et des fonctions à assurer.

L'approche globale de la commande des systèmes complexes devient dès lors difficile.

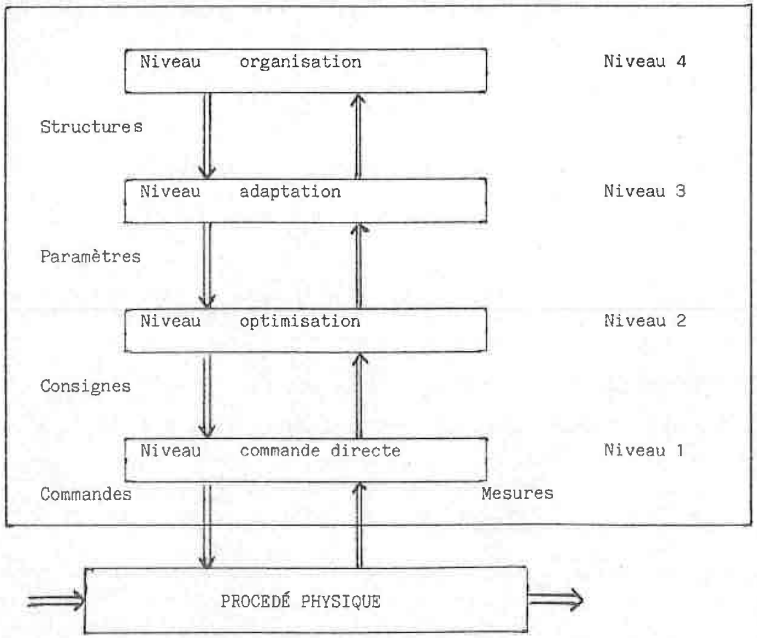
La division du travail, démarche utilisée dans beaucoup de disciplines, est retenue en automatique pour aborder la commande des systèmes complexes. Les automaticiens divisent le problème de la commande en problèmes plus simples et étudient chacun de ces problèmes séparément puis,coordonnent les solutions retenues pour assurer la commande du système global : c'est ce qu'on appelle la commande hiérarchisée /FOSSARD, 72/ , /LADET, 77/ , /TILTI, 78,79/. Avant d'utiliser ce concept de commande hiérarchisée,voyons en quoi consiste-t-il ?

On distingue, généralement deux manières de diviser ou de décomposer un système :
- la décomposition fonctionnelle,
- et la décomposition géographique.

1) Décomposition fonctionnelle

Une décomposition fonctionnelle en 4 niveaux de commande est communément admise /TILTI, 78,79/ , /FOSSARD, 72/ :
- le niveau commande directe,
- le niveau optimisation,
- le niveau adaptation,
- et le niveau organisation.

Ces niveaux sont parfois dénominés différemment et peuvent, à leur tour, être décomposés en plusieurs sous-niveaux.



Décomposition fonctionnelle de la commande

a) Le niveau commande directe correspond au niveau le plus élémentaire de la commande. Il est en liaison étroite avec le procédé : il est chargé, en effet de l'acquisition des mesures, de leur mise en forme, de la commande des actionneurs et du traitement des alarmes. Il a pour mission de maintenir les sorties à des points de consignes fixes ou de contrôler ces sorties suivant des trajectoires imposées.

b) Le niveau optimisation détermine les valeurs des points de consignes ou de trajectoires que devra respecter le niveau commande directe, en fonction du modèle du procédé. Il détermine ces consignes de façon à optimiser un certain nombre de critères (économiques ou autres).

c) Le niveau adaptation permet de modifier les paramètres intervenant dans les modèles utilisées au 1er et au 2ème niveau et assure le passage d'un mode de fonctionnement à un autre.

d) Le niveau organisation a pour mission de gérer l'ensemble des niveaux inférieurs. Il s'agit ici de la gestion intégrée qui a pour objet d'effectuer la liaison entre la production et la gestion de l'usine avec ses contraintes économiques.

2) Décomposition géographique

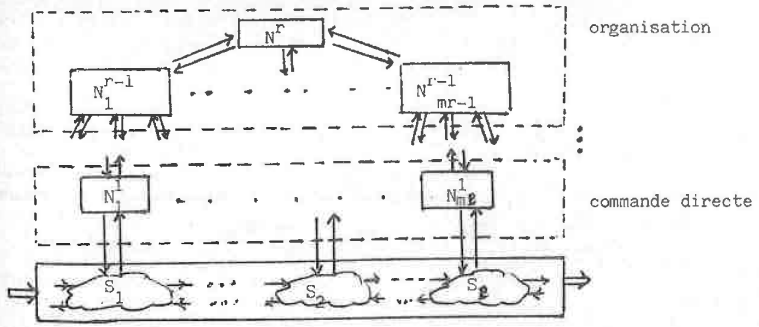
Comme nous l'avons déjà signalé, au chapitre I, les procédés complexes qui se développent, de nos jours, sont souvent le résultat d'association entre diverses unités de base ; on dit dans ce cas que le processus est composé de plusieurs sous-processus (par exemple une usine de fabrication de voitures est composée de robots de soudage, de robots de peinture, etc...)

Chaque unité peut être considérée comme un système avec ses entrées et sorties propres, avec sa dynamique propre. Considérer globalement le système de commande du procédé devient dès lors difficile, même si une commande hiérarchisée en niveaux est élaborée.

Il convient alors de décomposer le système en sous-systèmes géographiquement distincts et d'étudier séparément ces sous-systèmes;

Cependant, l'hypothèse d'indépendance n'est manifestement pas réaliste à cause des relations qui existent entre les divers sous-systèmes réalisant une tâche globale commune.

Il devient alors indispensable de coordonner l'ensemble des commandes locales.Si cette coordination est complexe , elle peut être décomposée en plusieurs niveaux d'où l'idée de commande sous forme de pyramide.



Dans la structure représentée par la pyramide précédente, la commande est décomposée en r niveaux, le procédé, quant à lui, est décomposé en ℓ sous-systèmes (ou unités de base).

Conclusion

L'utilisation simultanée de la décomposition fonctionnelle et de la décomposition géographique permet une meilleure approche et une meilleure compréhension des phénomènes et des problèmes à résoudre.

Nous considérons par la suite que les systèmes de commande sont hiérarchisés en quatre niveaux. Chacun de ces niveaux peut, à son tour, être hiérarchisé à cause de la décomposition géographique des procédés.

II.2 OBJECTIF DE LA DECOMPOSITION

Chaque niveau de la hiérarchie de commande est caractérisé par :
- un ensemble d'informations (ou de données) telles que les mesures, les consignes d'exploitation et les défauts détectés ;
- et un ensemble d'opérations (ou d'actions) manipulant ces informations, telles que les opérations d'E/S industrielles, les algorithmes PID, les tracés de courbes et les affichages de synoptiques.

Les actions opèrent sur les données pour réaliser les fonctions assignées aux différents niveaux de la hiérarchie de commande. En regroupant les différentes données et les différentes actions on peut décrire une application à l'aide de deux ensembles (celui des données et celui des actions).

L'objectif de la décomposition d'une ACPI, pour nous ici, est de déterminer l'ensemble des données et l'ensemble des actions.

II.3 DETERMINATION DES TYPES DE DONNEES ET D'ACTIONS

Pour définir une méthode de conception fondée sur l'utilisation des données, il est nécessaire de connaître les divers domaines que couvrent ces données et les traitements pouvant être effectués sur ces données. Ainsi on pourra dégager les caractéristiques de ces données et par là même définir les règles de conception associées à la méthode.

Nous tentons ici de répertorier les principaux types de données et d'actions pouvant intervenir dans la construction d'une ACPI.

Pour ce faire nous allons passer en revue chacun des quatre niveaux de la hiérarchie de commande afin d'en extraire les données et actions pouvant rentrer dans sa composition.

1) Niveau commande directe

a) Les données
Pour assurer la commande directe du procédé, le système de commande utilise les informations suivantes :
- les mesures qui le renseignent sur l'état du procédé et de l'installation de commande (par exemple la température de la lingotière de la coulée continue ou la vitesse de rotation d'un moteur) ;
- les commandes appliquées au procédé (par exemple fermeture d'une vanne ou déplacement d'un chalumeau)
- les consignes à respecter (envoyées par le niveau optimisation) (par exemple "la vitesse de rotation du moteur doit être égale à 1000 tr/mm");

- les défauts élémentaires détectés (moteur défaillant, capteur hors service etc...) ;
- plus généralement toute information nécessaire à la commande directe.

b) Les actions
Ce sont des opérations :
- d'acquisition de mesures et de transformation des signaux en provenance du procédé ;
- d'application des ordres (ou commandes) au procédé ;
- de tests pour la détection des défauts ;
- de régulation, etc...

2) Niveau optimisation

a) Les données
Le niveau optimisation dont le rôle est de répondre à des critères économiques ou autres,
- utilise les paramètres des modèles fournis par le niveau adaptation et les comptes-rendus en provenance du niveau commande directe ;
- calcule les consignes qu'il transmet au niveau commande directe ;
- rend des comptes-rendus au niveau adaptation ;
- et utilise des informations qui lui sont internes et qui dépendent de l'optimisation choisie.

b) Les actions
Ce sont des algorithmes d'optimisation.

3) niveau adaptation

a) Les données
Ce niveau, qui a pour mission d'estimer et de renouveler les paramètres des modèles de commande, utilise :
- d'une part les informations en provenance du niveau organisation qui décrivent les structures de commande à appliquer ;
- d'autre part les comptes-rendus en provenance des niveaux inférieurs ;
et transmet :
- les paramètres des modèles au niveau optimisation,
- et les comptes-rendus au niveau organisation.
Ce niveau, aussi, a ses propres données internes.

b) Les actions
Ce sont des algorithmes d'adaptation.

4) Niveau organisation

a) Les données
Le niveau organisation est le niveau le plus haut de la hiérarchie de commande, il a pour mission d'assurer la commande globale du procédé.

Pour ce faire il utilise :
- les structures de commande à appliquer qu'il transmet au niveau adaptation ;
- les différents comptes-rendus en provenance des couches inférieures ;
- les caractéristiques des divers composants de l'installation ;
- etc...

b) Les actions
Ce sont des algorithmes liés à l'organisation de la commande.

5) Le dialogue homme-machine

L'opérateur intervient aux différents niveaux de la hiérarchie de commande pour s'informer et/ou pour commander.

Outre les données et les actions évoquées précédemment pour les quatre niveaux de la hiérarchie de commande, chacun de ces niveaux englobe les données et les actions nécessaires au dialogue homme-machine lié à ce niveau. Ces données et actions dépendent du niveau de commande auquel elles sont associées et peuvent, généralement, correspondre:

a) pour les données :
- aux ordres envoyés par l'opérateur,
- aux informations constituant les synoptiques,
- aux informations relatives aux suivis de produit, du personnel et de l'installation
- aux statistiques et calculs divers,
- aux mesures,
- aux commandes appliquées au procédé,
- aux défauts détectés,
- etc...

b) pour les actions
- à l'impression de messages,
- à la prise en compte des ordres opérateurs,
- au tracé de courbes,
- au déclenchement d'alarmes,
- à la visualisation de synoptiques,
- aux suivis (de produit, du personnel et de l'installation),
- à des calculs divers (statistiques, ...),
- etc...

c) Remarque

Nous venons de dégager les principaux types de données et d'actions pouvant rentrer dans la composition d'une ACPI dans un cas général. Pour tenir compte des cas particuliers, nous avons signalé l'existence d'informations et/ou d'actions propres à chaque niveau de commande, ce qui permet à notre étude, de rester applicable à des ACPI de natures diverses.

II.4 DEFINITION DES NOTIONS D'ACTIONS ET DE DONNEES

Après la brève analyse des principaux types de données et d'actions nous allons préciser ce que voudront désigner, par la suite, les notions d'actions et de données.

1) Les données

Nous appelons donnée toute information simple ou composée pouvant correspondre à :
- une mesure,
- un ordre de commande,
- un défaut (élémentaire ou calculé),
- une information décrivant la structure de la commande,
- un paramètre de modèle,
- une consigne d'exploitation,
- un ordre opérateur,
- une information correspondant à un suivi, des statistiques ou à un calcul quelconque,
- et éventuellement à une information propre à un niveau donné de la commande.

Une donnée est dite "simple" si elle prend des valeurs d'un type élémentaire donné (des entiers ou des réels par exemple).

Exemple de donnée simple :
La température de la lingotière délivrée par un thermocouple (elle est représentée par un réel).
Une donnée simple est porteuse d'une seule information élémentaire.

Une donnée est dite "composée" si elle est le résultat d'association entre plusieurs informations.

Exemples de données composées
- un fichier statistique concernant les défauts de fabrication,
- une donnée représentant le suivi d'un produit (qualité , heure de début de fabrication...)

2) Les actions

On appelle action toute opération réalisant une fonction "ayant un sens pour la commande".

L'action est l'unité de base de décomposition d'une application.

En vertu de la brève analyse faite précédemment une action peut correspondre à :
- une opération relative au dialogue homme-machine (affichage, impression, déclenchement d'alarme, etc...) ;
- une opération relative au dialogue calculateur-procédé (acquisition de mesures et application des commandes au procédé) ;
- un algorithme de type automatique (de régulation, de surveillance, de protection, d'optimisation, d'adaptation, d'organisation, de coordination, etc...) ;
- un suivi ;
- ou à un calcul quelconque (statistique ou autre).

REMARQUE

Quelque soit sa complexité, toute action produit une et une seule donnée. Le sens de l'action est liée à la donnée qu'elle produit (c'est en ce sens qu'on dit "ayant un sens pour la commande").

Une action peut, à son tour, se décomposer en plusieurs opérations (ceci est particulièrement vrai pour les actions qui produisent des données composées) mais nous nous intéressons, dans ce travail, à l'action en tant qu'entité indivisible.

III BASES DE DONNEES INDUSTRIELLES : DEFINITION ET PARTICULARITES

L'analyse précédente a permis de dégager les principales composantes d'une ACPI (ie les données et les actions). L'étape suivante dans la conception de l'application est d'utiliser ces objets pour affiner davantage la solution déjà obtenue.

Cette utilisation repose évidemment sur les particularités liées aux objets (par exemple l'aspect temps réel des données conduira à une activation des actions en temps réel). Il est donc important de commencer par étudier les particularités des données.

III.1 CONCEPT DE BASE DE DONNEES INDUSTRIELLES (BDI)

Les données relatives à un procédé constituent un fond commun que nous appelons "base de données industrielles" (cette appellation a été retenue par analogie aux applications de gestion).

Nous définissons une BDI comme suit : "une base de données industrielles est l'ensemble des informations relatives à un procédé industriel, à sa commande et à sa conduite. Ces informations relatent l'histoire du procédé et de sa commande depuis une certaine date".

Ce qui signifie en d'autres termes qu'une BDI permet de représenter les informations qu'on a sur :
- le passé du procédé et de sa commande ie :
  . la commande qui a été appliquée au procédé,
  . les mesures effectuées,
  . les différents défauts détectés,
  . l'utilisation des divers appareils du procédé,
  . la qualité et la quantité des produits fabriqués,
  . etc...
- l'état courant du procédé et de sa commande ie :
  . les mesures les plus récentes,
  . les dernières commandes appliquées au procédé,
  . les défauts présents,
  . l'état des différents paramètres utilisés (consignes, paramètres de modèle, structures de commande, ...) ;
  . etc...
- le passé de l'installation de commande et du personnel ie :
  . les diverses utilisations des différents équipements de l'installation,
  . les anomalies de fonctionnement détectées,
  . les divers postes occupés par le personnel (opérateurs, automaticiens,...)
  . etc...
- et l'état courant de l'installation et du personnel ie :
  . l'état courant des divers composants de l'installation de commande (ceux opérationnels, ceux hors service...),
  . les divers postes actuellement occupés par le personnel,
  . etc...

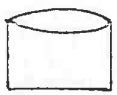
Remarques

- Il est certainement difficile de dresser une liste exhaustive des diverses données que peut englober la BDI, nous en avons mentionné les plus significatives.

- Nous convenons de représenter une BDI par le schéma suivant :



au lieu de



parce que le 2e schéma, utilisé en informatique de gestion,laisse croire qu'il s'agit d'une base de données sur disque et comme nous le verrons il n'en est point.

III.2 CARACTERISTIQUES DES BDI ET LEURS IMPACTS SUR LA METHODE DE CONCEPTION

1) Caractéristiques des BDI

Par analogie à une base de données classique (/ ANSI, 75/ , /CODD, 70/ , /DATE, 75/) qui permet de représenter la connaissance qu'on a sur une partie du monde réel (une entreprise ou un hôpital par exemple), une BDI est un modèle d'un phénomène physique évoluant dans le temps.

Elle ne peut être une traduction fidèle du procédé et de sa commande que si ses caractéristiques tiennent compte de celles liées au procédé et à sa commande (cf paragrapheII/Chapitre Ipour les caractéristiques des PI).

De telles caractéristiques résident essentiellement dans les points suivants :
- l'aspect temps réel de la BDI,
- sa répartition,
- sa cohérence,
- la diversité des données qu'elle peut contenir,
- les relations entre les données qu'elle contient,
- et son évolutivité.

Voyons avec un plus de détails, ces caractéristiques.

a) Aspect temps réel

La BDI conserve des informations permettant de décrire un phénomène physique évoluant dans le temps.

La mise à jour de la BDI devra tenir compte des contraintes de temps imposées par l'évolution du procédé (c'est une mise à jour en temps réel). La BDI ne peut être une traduction fidèle de l'état du procédé que si elle prend en compte, et au bon moment, tous les changements d'état caractéristiques de l'évolution du procédé et de sa commande;

Si le concept de temps n'est pas important (généralement) pour la mise à jour des BDC (BDC pour Base de Données Classique c'est-à-dire non temps réel) il est la caractéristique essentielle des BDI. Le choix de la structure d'exécution (installation informatique) devra tenir compte des spécificités temps réel de l'application.

Remarque

Les données correspondant au passé du procédé, de sa commande et du personnel constituent un historique des divers états (du procédé de sa commande et du personnel) et peuvent être regroupées en une base de données, à part, puisque ces données ne sont pas sujettes aux mêmes contraintes temps réel que les données utilisées dans la commande du procédé. Une telle base de données est à rapprocher des bases de données historiques /FOUCAULT, 82/ développées, en gestion, pour décrire les différents modes de gestion d'une organisation depuis une certaine date. L'analogie entre ces deux types de BD peut être illustrée par l'exemple suivant :
- une BD historique relative à une entreprise permet de connaître le salaire versé à un ouvrier à une date t donnée ;
- une BD historique relative à une coulée continue permet de connaître les qualités de l'acier ayant servi à la production d'un bloom donné.

Les mécanismes développés pour la gestion des BDC historiques peuvent alors servir à gérer (stocker, accéder...) les données décrivant le passé du procédé, de sa commande et du personnel ; la gestion (ie l'organisation, l'implantation et l'exploitation) de ces données n'est pas traitée dans le présent travail.

b) La répartition

Pour tenir compte de la répartition géographique des procédés, les systèmes informatiques chargés de commander ces procédés sont souvent géographiquement répartis. Cette répartition a comme conséquence la distribution des BDI que les systèmes informatiques utilisent.

La répartition de la BDI est parfois nécessaire pour respecter les contraintes de temps imposées par certains procédés complexes, si un seul site ne peut, à lui seul, suffir pour supporter toute l'application.

c) La cohérence

La défaillance des circuits, l'aspect temps réel et la répartition de la BDI sont des facteurs qui rendent le problème de la cohérence de la BDI extrêmement difficile à résoudre.

Une opération (par exemple la communication entre deux sites ou la réponse d'un capteur) non achevée ou achevée en retard peut se traduire par une perte de mise à jour de la BDI.

Si, dans les BDC on peut "rattraper" les pertes de mise à jour en relançant les opérations ratées /BOUCHET, 79/ , /HAERDER, 83/ le problème pour une BDI est tout autre puisqu'un système temps réel traite des couples (opérations, temps) et non pas des opérations seulement.

Une incohérence au sein de la BDI peut conduire, à court ou à long terme, à des erreurs de commande si cette incohérence n'a pas été détectée ; plus la BDI est cohérente et plus la commande est saine.

d) Diversité des types de données

Les données relatives à un procédé et à sa commande sont de natures très diverses. La conséquence d'une telle diversité est que les moyens à mettre en oeuvre pour rechercher, définir et implémenter une donnée dépendent du type de cette donnée.

Par exemple la définition d'un synoptique (sa structure et les actions qui permettent de le manipuler) diffère de la définition d'un ordre à appliquer au procédé.

e) Relations entre les données

Si l'on se réfère aux BDC les entités (ie les informations stockées concernant par exemple des personnes, des maisons ou des voitures) ne sont pas construites les unes à partir des autres. La BD est définie à l'aide de ces entités et des relations ou liens qui existent entre elles (par exemple X possède la voiture R, X habite Z,...).

Le problème de liens entre les données d'une BDI se pose autrement. Les principales relations entre les données de la BDI résident au niveau des points suivants :
- la construction des données,
- la redondance des données,
- et le contrôle de la mise à jour des données.

- Construction des données

On peut regrouper les données de la BDI en deux groupes :
. les données "primaires qui ne peuvent pas être construites à partir des autres données, c'est notamment le cas des mesures ;
. et les données "construites" c'est-à-dire obtenues en appliquant un algorithme quelconque à une ou plusieurs données (par exemple une donnée statistique -un fichier- est obtenue en traitant une ou plusieurs données).

Ce regroupement fait apparaître une relation de "construction" entre les données. Cette relation est très importante car elle permet de déterminer :
. la hiérarchie des données,
. le flux d'informations relatives à une ACPI et l'utilisation de ces informations.

Elle apparaîtra de manière implicite lors de la recherche des données (cf au paragraphe IV).

- Redondance

Pour des problèmes de sûreté de fonctionnement, le concepteur peut associer à plusieurs données différentes un même sens, par exemple une mesure captée par plusieurs voies différentes. On dit alors qu'une donnée est synonyme d'une autre donnée.

di synonyme de Dj.

- Contrôle de mise à jour

Toute donnée est mise à jour à une fréquence qui lui est spécifique. Cette fréquence peut être définie à partir d'autres données ; par exemple dans la phrase "sur dépassement de vitesse avertir l'opérateur par un message de type x". La donnée message de type x est mise à jour (communiquée à l'opérateur) si la vitesse dépasse le seuil de sécurité, on dit alors que la donnée message est contrôlée par la donnée vitesse.

Remarque

Les trois relations précédentes définissent les liens qui existent entre les données de la BDI mais n'apparaissent pas dans la description finale de l'application (elles sont déduites de la description de l'application).

f) L'évolutivité

Une caractéristique que nous avons soulignée au chapitre I est l'aspect évolutif des procédés et de leur commande.

Toute modification, répondant à un objectif quelconque, se traduit par une évolution au sein de la BDI et peut consister en :
- l'adjonction de nouvelles données,
- la suppression de données déjà existantes car jugées inutiles,
- ou en la modification des données existantes (changement de leur structure interne ou de leurs règles de mise à jour).

Une conception modulaire de l'application faciliterait une telle évolution.

2) Impacts des caractéristiques des BDI sur la méthode de conception

Il est naturel qu'une méthode de conception et d'implantation d'applications tienne compte des spécificités de ces applications.

Pour prendre en compte les caractéristiques des BDI énoncées précédemment, la méthode de conception et d'implantation choisie doit respecter les contraintes suivantes :

a) au niveau conception :
- la description de la mise à jour de la BDI doit refléter l'aspect temps réel de l'application et permettre une expression aisée des règles d'évolution du procédé et de sa commande ;
- la description de l'application doit être modulaire (pour faciliter la répartition et l'évolutivité de l'application) ;
- la prise en compte des anomalies de fonctionnement entrainant des incohérences de la BDI doit se faire dès l'étape de conception ;
- la recherche des données et des actions doit tenir compte de la diversité de données et des relations qui existent entre elles (et ce pour faciliter la recherche des données et des actions).

b) au niveau implantation
- la structure d'exécution doit être telle que les contraintes de temps soient respectées ;
- les moyens de dialogue avec le procédé doivent assurer un maximum de cohérence de la BDI (ie permettre de transmettre et de recevoir les informations de manière fiable et au bon moment) ;
- les moyens d'implantation d'une donnée (ou d'une action) doivent tenir compte de la nature de cette donnée (ou cette action) afin de renforcer les points "vulnérables" du système ;
- la structure d'exécution (aspects matériel et logiciel) doit être modulaire (pour s'adapter à l'évolution de l'application et à sa maintenance) ;
- la structure d'exécution doit tenir compte de la répartition géographique du procédé.

Conséquence

Si la méthode de conception et d'implantation des ACPI respecte les contraintes ci-dessus, les caractéristiques des ACPI, ie :
- l'aspect temps réel,
- l'interaction avec le procédé,
- la répartition géographique,
- la sûreté de fonctionnement,
- et l'évolutivité du procédé et de sa commande.

sont alors prises en compte.

3) Conclusion

Nous avons dégagé les caractéristiques des BDI et les contraintes que doit respecter la méthode de conception et d'implantation des ACPI. Les solutions retenues pour la prise en compte de ces caractéristiques seront développées dans la suite de ce travail.

A partir du paragraphe suivant nous commençons à introduire les concepts et démarches de conception et d'implantation des ACPI, nous reviendrons, une fois que la méthode sera complètement définie, sur les contraintes citées précédemment pour voir comment elles sont prises en compte par notre approche.

IV APPROCHE DE CONCEPTION

La détermination des types de données et d'actions ainsi que les particularités des bases de données industrielles nous a permis de fixer les objectifs à atteindre ie les propriétés que doit vérifier notre approche de conception des ACPI.

Nous développerons dans ce paragraphe les grandes lignes de la démarche à suivre par le concepteur pour déterminer :
- l'ensemble des données (ie la BDI),
- l'ensemble des actions,
- et les conditions de la mise à jour de la BDI.

Cette démarche consiste à décrire une ACPI en 3 étapes :
1. recherche et description des objets (données et actions),
2. recherche et description des règles de mise à jour des données,
3. description de la structure interne des actions.

La description des applications se fait à l'aide d'un modèle que nous décrirons dans les chapitres III et IV.

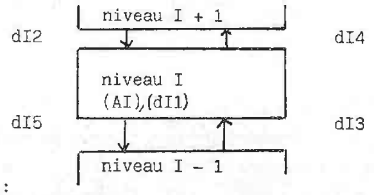
IV.1 DETERMINATION DES OBJETS

La première étape de conception de l'application est la définition des objets de base de cette application.

Comme nous l'avons signalé au paragraphe II, il s'agit de parcourir les différents niveaux de la hiérarchie de commande pour déterminer les objets de base constituant l'application.

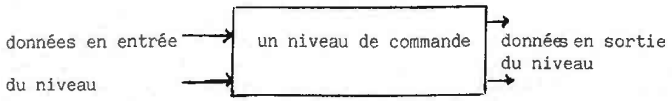
1) Principe de décomposition d'une application

Chaque niveau de la hiérarchie de commande est caractérisé par un ensemble d'informations (dI1, dI2, dI3, dI4, et dI5) et un ensemble d'actions (AI)

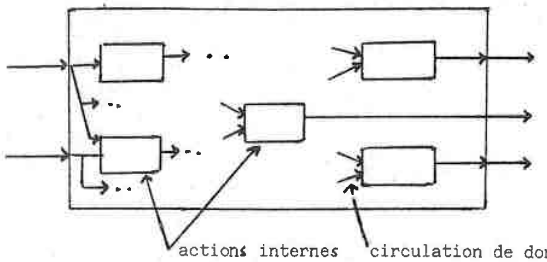


où :
dI1 : désigne les informations propres (ou informations internes) au niveau
dI2 : désigne les informations qu'il reçoit du niveau supérieur
dI3 : désigne les informations qu'il reçoit du niveau inférieur
dI4 : désigne les informations qu'il transmet au niveau supérieur
dI5 : désigne les informations qu'il transmet au niveau inférieur
AI : désigne l'ensemble des actions réalisées par le niveau.

Le principe de recherche des constituants d'une application (données et actions) consiste :
- premièrement à considérer chaque niveau comme un tout afin de déterminer les données circulant entre les divers niveaux de la hiérarchie.



- et ensuite s'intéresser à la structure interne du niveau pour en déterminer :
  - les actions internes
  - et les données internes.



Chacune des fonctions réalisées par le niveau de commande considéré doit être raffinée jusqu'à l'obtention d'actions (ayant un sens pour la commande et ne pouvant plus, ou ne nécessitant plus d'être décomposées).

Le principe de décomposition repose donc sur les concepts d'abstraction et de raffinements successifs /PARNAS, 72A,72B, 75/ , /SAKHONN, 80/ , /WIRTH, 71A/ permettant d'obtenir une application modulaire /DERNIAME, 74/ , /SAKHONN, 80/ , /ZELKOWITZ, 78/.

L'abstraction consiste donc à délimiter (ou à encapsuler) tout ou partie d'un niveau afin de décrire son interaction avec son environnement (dans un même niveau ou vis-à-vis des autres niveaux).

Le raffinement consiste à décrire une structure (une partie d'un niveau ou la totalité d'un niveau) par une suite de structures plus simples qui, elles à leur tour, seront raffinées jusqu'à l'obtention d'actions et de données ayant le sens donné dans paragraphe II.3).

Les actions obtenues sont indépendantes : il n'existe, entre elles, que des relations de partage de données ; d'où la modularité de la structure obtenue.

2) Décomposition d'un niveau de commande

Chaque niveau de commande réalise un ensemble de fonctions qui participent à la réalisation d'un objectif global qui est la commande de l'ensemble du procédé physique.

niveau I : $\{Fi_1, \ldots, F_i n_i\}$ Fij : est une fonction

Le concepteur détermine des fonctions par analyse du cahier des charges puis s'intéresse à leur description.

Réaliser une fonction signifie obtenir des résultats R à partir d'informations en entrée E :

$$F(E) = R$$

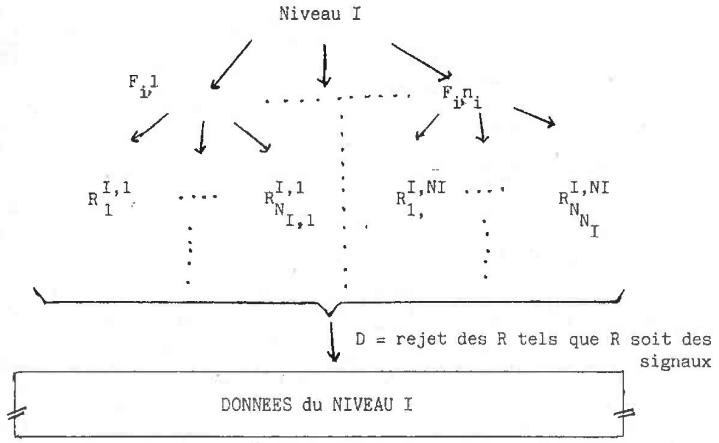
où $R = \{R1, .., Rn\}$

$E = \{E1, \ldots En\}$

Les éléments de R (respectivement de E) représentent soit des données soit des signaux à destination (respectivement en provenance) du procédé et/ou de l'opérateur.

Règle

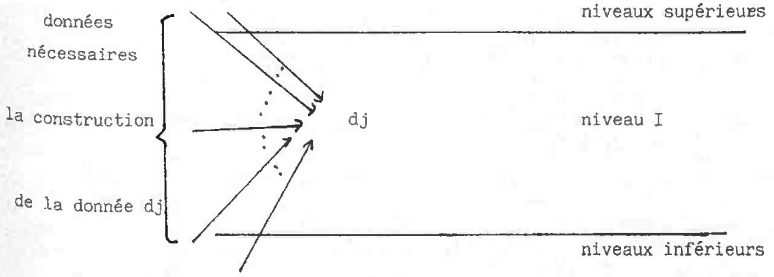
Si un élément Ri de R n'est pas une application d'une commande au procédé ou une communication d'information à l'opérateur **alors** Ri est une donnée du niveau considéré.



Les données du niveau I sont déduites de l'ensemble des résultats des fonctions relatives au niveau I en éliminant les résultats qui correspondent à des signaux à destination du procédé ou de l'opérateur.

L'examen de chacune de ces données permet de déterminer les entrées nécessaires à sa construction.

Ces entrées sont soit des signaux en provenance du procédé ou de l'opérateur, soit des données en provenance d'autres niveaux ou du même niveau.



Exemples

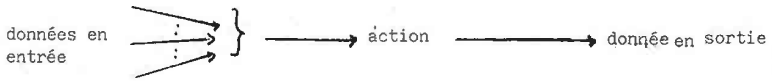
. Pour déterminer les consignes à respecter par le niveau commande directe, le niveau optimisation utilise des données en provenance du niveau adaptation et des données en provenance du niveau commande directe.
. Le synoptique général d'une installation (c'est une donnée) est obtenu en traitant l'état des différents composants de l'installation (cet état est défini par des données).

Il apparaît ainsi qu'une même donnée peut être utilisée à plusieurs niveaux de la commande.

L'utilisation dépend du niveau de commande considéré; par exemple une mesure est utilisée, au niveau de la commande directe, pour définir les actions à appliquer au procédé et utilisée, au niveau optimisation, pour déterminer les consignes d'exploitation.

Une donnée est obtenue à partir d'autres données (ou de signaux en provenance du procédé ou de l'opérateur) moyennant la transformation adéquate : par exemple un algorithme PID qui traite des mesures et des consignes pour définir une commande à appliquer au procédé.

Le deuxième ensemble que doit définir le concepteur est celui des actions qui produisent les données. les actions représentent les moyens de transformations des entrées .

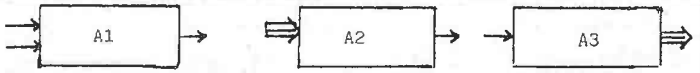


3) Graphe associé au flux de données

Au fur et à mesure que le concepteur progresse dans l' exploration des niveaux de commande, il construit le graphe correspondant au flux de données relatives à l'application qu'il traite.

Représentation schématique

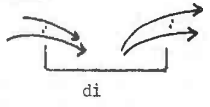
. les actions : une action est représentée par un rectangle ; les flèches désignent les entrées et les sorties de l'action.



⇒ désigne une communication avec l'environnement

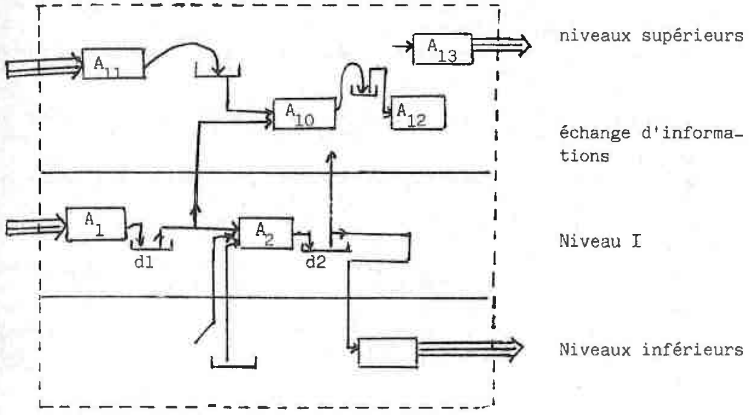
A1 ne communique pas avec l'enrironnement,
A2 reçoit des signaux en provenance de l'environnement,
A3 émet des signaux vers l'environnement ;

. les données : une donnée est représentée par une sorte de "cuve" pour désigner le fait qu'elle peut correspondre à une information simple ou à une série d'informations



. graphe associé à une application.

En utilisant les représentations schématiques retenues pour les données et les actions, une application peut être décrite par un graphe semblable à celui de MASCOT /JACKSON, 76/ mais avec deux types de noeuds, dans lequel, on sépare les objets relatifs aux différents niveaux de la hiérarchie.



Graphe associé à une application

Intérêts du graphe

- il donne une représentation schématique de l'application qui illustre :
  . le flux de données (échanges entre les actions d'un même niveau et échanges entre les actions de niveaux différents)
  . l'utilisation des données,
  . la construction des données (pour chaque donnée on schématise les données qui sont utilisées pour la construire) ;
- et il sert surtout à la validation de l'application :
  . il permet de vérifier si toutes les données référencées à un niveau, pour définir une donnée, ont été définies, à leur tour, ou non,
  . il permet de vérifier si le concepteur a associé, ou non, à toutes les données, les actions qui permettent de les produire.

L'exploitation du graphe permet donc de s'assurer de la complétude de la structure obtenue.

4) Remarque : une autre hiérarchisation des objets

Une première hiérarchisation des objets est celle donnée précédemment : on associe à chaque niveau de commande sa BDI et les actions qui la manipulent. Une autre hiérarchisation fondée sur les fonctionnalités du système de commande peut être utilisée conjointement avec la précédente.

Voyons tout d'abord en quoi consiste cette deuxième hiérarchisation.

Les données et actions obtenues par la décomposition précédente peuvent être classées selon les fonctionnalités du système auxquelles elles sont associées. Nous rappelons que les fonctionnalités d'un système de commande sont de trois types (cf chapitre I paragraphe II).

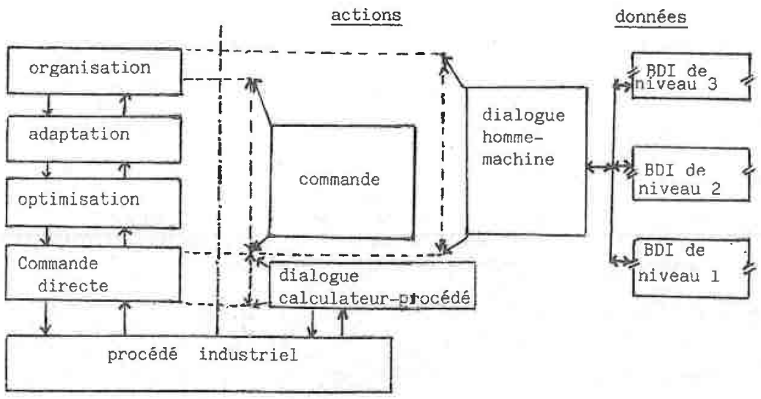
- le dialogue opérateur-calculateur,
- le dialogue procédé-calculateur,
- et la commande du procédé (avec tous les algorithmes qu'elle utilise).

Cette classification conduit à une hiérarchisation des actions et des données comme le montre la figure ci-après.

Les actions sont regroupées en trois classes :
. actions liées au dialogue opérateur-calculateur,
. actions liées au dialogue procédé-calculateur,
. actions de commande (optimisation, adaptation, régulation...).

la BDI se subdivise en trois BDI (ou sous-BDI).
. la BDI de niveau 1 contient les mesures et les ordres à appliquer au procédé ; les informations qu'elle contient sont en relation directe avec le procédé.
. la BDI de niveau 2 contient les informations nécessaires à la commande (ie les consignes, les paramètres, les structures de commande etc...) ; ces informations traduisent l'aspet purement automatique de la commande de procédés ;
. la BDI de niveau 3 contient d'une part toutes les informations complexes liées au dialogue homme-machine (ie les statiques diverses, les synoptiques et les informations relatives aux divers suivis) et d'autre part les ordres émis par l'opérateur vers le système de commande (demandes d'information, demande de visualisation de synoptiques, changements de mode de marche etc...)

Les trois niveaux de BDI ne sont pas indépendants puisque les informations d'un niveau sont construites en utilisant les informations des autres niveaux.



Pourquoi une telle hiérarchisation ?

Parmi les qualités attendues d'une méthode de conception et d'implantation les deux suivantes ont motivé l'introduction de cette hiérarchisation :
- la méthode doit être basée sur des critères qui facilitent la tâche du concepteur lors de la recherche et de la description des objets ;
- la prise en compte des caractéristiques des objets résultant de la phase de conception devra permettre une implantation efficace de l'application.

Le découpage de l'application, en trois classes, utilisé conjointement avec le modèle de la commande hiérarchisée de cette application tient compte des deux qualités énoncées précédemment puisque :
- les trois classes, au niveau fonctionnel, sont différentes et devraient être déduites du cahier des charges de manière séparée. Elles demandent des compétences différentes ou font intervenir des personnes différentes: instrumentistes, automaticiens, économistes, concepteurs d'interface homme-machine etc...

Le concepteur (ou les concepteurs) aura à traiter l'application selon trois points de vue différents (dialogue homme-machine , dialogue procédé-machine et commande) ; ce qui lui permet de sérier les problèmes de conception.
- les mécanismes et outils nécessaires à l'accomplissement de chacune des trois fonctionnalités diffèrent. Par exemple :
  . au niveau du dialogue calculateur-procédé on a besoin d'un maximum de redondance des circuits, alors que ce même besoin se fait moins sentir au niveau du dialogue homme-machine ;
  . ou encore la définition d'une statistique sur la vie du procédé fait intervenir des méthodes statistiques, des méthodes d'organisation de fichiers, etc... alors que la définition d'une mesure fait appel à des méthodes de traitemet du signal (filtrage, échantillonnage, etc...).

La conséquence des deux raisons évoquées est que les objets constituant une ACPI ont des caractéristiques liées aux classes des fonctions auxquelles appartient chacun de ces objets. Ces caractéristiques influent aussi bien, sur la phase de conception (recherche et définition des objets) que sur la phase d'implantation (choix des outils).

La séparation des objets permet au concepteur de mieux comprendre les spécificités de chaque classe d'objets et de dégager plus facilement les caractéristiques à prendre en compte pour la description et l'implantation de chaque objet.

Nous verrons dans la 2è partie que la hiérarchisation de la BDI permet de minimiser les incohérences de la BDI si on implante différemment les trois niveaux de la BDI (il s'agit de renforcer les points vulnérables du système).

IV.2 DETERMINATION DES CONDITIONS DE MANIPULATION DES OBJETS

Le concepteur qui vient de déterminer et structurer les objets de base constituant son application continue son étude en cherchant à déterminer l'aspect dynamique de son application (ie chercher à définir les circonstances dans lesquelles les données sont mises à jour).

L'évolution, dans le temps, du procédé et de sa commande se traduit par des mises à jour de la BDI (rafraîchissement des données et utilisation des informations déjà produites). La prise en compte d'une telle évolution laisse supposer l'existence de règles à respecter pour assurer la mise à jour de la BDI.

Exemples
Si dans le cahier des charges on rencontre une phrase du style "régulation d'une température à 1000°C avec une période de scrutation de 100 ms..." le concepteur de l'application devra définir une règle de mise à jour de la donnée température, cette règle pourra s'énoncer ainsi"tous les 100 ms produire une nouvelle mesure de température"
ou encore si l'on rencontre une phrase du genre "sur dépassement du seuil de température avertir l'opérateur" le concepteur devra définir une règle associée aux messages-opérateur,par "sur détection de l'évènement de dépassement de température , activer l'action d'impression (ou de visualisation) d'un message-opérateur".

Le concepteur de l'application sera donc amené, durant cette 2ème étape de conception, à définir un ensemble de règles exprimant les instants (ou les conditions) de mise à jour des données.

Pour ce faire il aura à considérer :
- l'état du système et du procédé
- les changements d'état du système et du procédé,
- et l'évolution du temps.

Ainsi un troisième ensemble (celui des règles de mise à jour) vient s'ajouter aux deux ensembles (les actions et les données) déjà définis.

La recherche des règles de mise à jour de la BDI est habituellement appelée recherche des règles d'activation des actions /LADET, 82/ ; /THOMESSE, 80/, /BENMAIZA, 84/. En effet, la détermination de l'ensemble des règles de mise à jour de la BDI implique directement l'obtention des règles d'activation des actions.

C'est là un point important de notre approche qui peut s'énoncer ainsi "les actions sont pilotées par les données".

Une action est activée pour utiliser et produire des données, donc pour mettre à jour une donnée de la BDI. La règle d'activation de cette action est déduite des conditions de mise à jour de la donnée dont elle est productrice.

En associant à une donnée un ensemble de producteurs et un ensemble d'utilisateurs et des règles de production et d'utilisation, on a défini complètement la gestion des accès à cette donnée.

Les règles, ainsi que le langage, de description des conditions de mise à jour de la BDI feront l'objet du chapitre IV.

IV.3 DESCRIPTION DES ACTIONS

Il s'agit de détailler le comportement de toutes les actions intervenant dans la structure fonctionnelle précédemment définie, et ce en respectant les spécifications opératoires du cahier des charges.

A chaque action le concepteur associe un programme séquentiel structuré à l'aide d'un ensemble de traitements (ou d'algorithmes).
- la recherche des algorithmes doit se faire indépendamment de tout choix technologique et sans hypothèse sur la durée d'exécution des instructions;
- la recherche des algorithmes doit se faire par raffinements successifs (/PAIR,79/ ,/WIRTH, 71A/) . Cette technique permet d'aboutir à des algorithmes structurés.

Remarque
L'enchainement des différents algorithmes constituant une action n'est pas étudié dans le présent travail.

IV.4 PRISE EN COMPTE DES ANOMALIES DE FONCTIONNEMENT

Comme le dit VON NEWMANN, cité dans /VERNEL, 77/, "il n'est pas question d'éliminer les pannes ou de supprimer complètement leurs effets. Tout ce que l'on peut essayer, c'est d'arranger le système de telle manière qu'il puisse continuer à fonctionner en présence de la plus grande partie d'entre elles".

En effet, le mode de fonctionnement normal (ie sans présence de pannes) doit toujours être compatible avec l'existence de pannes ; ce qui permet au système de continuer à accomplir sa tâche, en performances dégradées s'il le faut.

Cependant, le fonctionnement en performances dégradées n'est possible que si l'on dispose :
- de moyens de détection des situations de mauvais fonctionnements,
- et de procédures de reprise en cas de détection d'anomalies.

La solution que nous proposons, ici, consiste à déterminer, à partir du cahier des charges, les situations de mauvais fonctionnements desquelles il faut tenir compte et définir les actions à entreprendre en cas d'apparition de ces mauvais fonctionnements.

Exemple
Soit un cahier des charges contenant la phrase suivante "réguler la vitesse d'un moteur à 2000tr/mn et déclencher l'alarme si la vitesse dépasse 2010 tr/mn". Dans ce cahier des charges la situation de mauvais fonctionnement est définie par le dépassement des 2010 tr/mn. Au niveau de l'application le concepteur devra définir une action alarme (sa donnée c'est les informations relatives à l'alarme) activée si l'évènement dépassement de vitesse est détecté.

Le traitement de situations de mauvais fonctionnements est analogue à celui des situations normales ; les mêmes concepts et les mêmes démarches sont applicables dans les deux cas :
- le traitement des mauvais fonctionnements se fait par l'intermédiaire d'actions qui opérent sur des données,
- la détection de niveau fonctionnements se fait en considérant :
  . l'écoulement du temps : la non-présence d'un appareil est, généralement, détectée par l'intermédiaire de temporisations.
  . les changements d'état du système : la vérification de conditions particulières sur les données (ie que ces données atteignent des valeurs critiques) peuvent être synonymes de mauvais fonctionnements ;
  . et l'état du système : l'écoulement du temps et les changements d'état du système ne peuvent conduire à l'activation d'actions de traitement d'anomalies que si l'état instantané du système permet la prise en compte de ces anomalies (par exemple si le système est en mode "guide-opérateur" un dépassement de température ne conduit pas à une action directe sur le procédé car elle n'est pas autorisée par l'état du système).

Ecoulement du temps, changement d'état et état du système servent à exprimer, aussi bien, les conditions d'activation des actions associées au fonctionnement normal que les conditions d'activation des actions associées aux mauvais fonctionnements.

V CONCLUSION

Nous avons commencé, ce chapitre, en dressant un répertoire des principales données et actions pouvant constituer une ACPI. Les données ainsi recensées constituent un fond commun que nous avons appelé "base de données industrielles". Les particularités de ce type de bases de données ont été présentées ; elles servent de guide pour la recherche des règles de description des ACPI.

Nous avons, ensuite, abordé la présentation d'une approche de conception d'ACPI. Laquelle approche repose essentiellement sur :
- la séparation entre les données et les actions,
- et le "pilotage" des actions par les données.

Pour mieux comprendre les concepts introduits nous avons tenté de proposer une démarche de conception d'ACPI dont le développement se poursuivra dans les prochains chapitres. Cette démarche consiste à :

1. uciliser comme outil de départ le cahier des charges de l'ACPI à réaliser mais vu sous l'angle commande hiérarchisée ;

2. décomposer l'application pour déterminer les objets de base (ie les données et les actions);

3. structurer ces objets en :
   - hiérarchisant les données et les actions,
   - et en associant un graphe à l'application générale;

4. en étudiant la dynamique du procédé et de sa commande, chercher à définir les conditions de mise à jour des données (ou encore définir les règles d'activation des actions) ;

5. et enfin s'intéresser à la description interne des actions.

CHAPITRE III

DESCRIPTION DES APPLICATIONS : ETUDE DE L'ASPECT STATIQUE DES APPLICATIONS

I INTRODUCTION

Dans le chapitre précédent nous avons effectué une décomposition des ACPI qui nous a permis de mettre en relief les objets fondamentaux pour la conception d'une application selon une approche données/actions. Un début d'une démarche méthodologique, pour la recherche de ces objets, a été développée aussi.

Au cours de ce chapitre nous nous intéressons à la description de l'aspect statique des applications (ie à la structure des données, à leur construction et à leur utilisation). La description de l'aspect dynamique (ie la synchronisation de l'application ou l'activation des actions ou encore la mise à jour des données dans le temps) sera abordée ici mais son développement sera traité dans le prochain chapitre.

Après avoir dégagé les paramètres caractéristiques (ie les attributs) des objets nous proposerons une approche de description de ces objets.

Les attributs des objets découlent essentiellement de la manipulation de la BDI puisque notre méthode de conception repose sur le concept de données. L'approche de description comprendra :
- un langage de description d'objets,
- et des règles à respecter pour décrire les objets.

Le langage de description des objets est un langage de programmation. Pendant la phase d'implantation de l'application on étendra la solution obtenue pendant cette phase de conception par l'adjonction d'attributs liés à l'implantation des objets (priorités des actions par exemple).

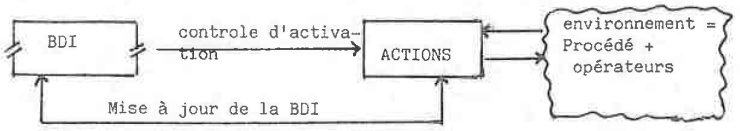
En reprenant les travaux cités dans le chapitre I paragraphe III nous essaierons de les comparer au nôtre.

II DESCRIPTION DES DONNEES

Comme pour les langages de programmation (PL1, PASCAL,...) ou pour les bases de données classiques l'utilisation des données d'une BDI implique, pour chaque donnée :
- la description de la structure de cette donnée,
- et la description de la manière dont elle est exploitée par l'application.

Nous essayons ici, de dégager les attributs permettant de décrire la structure interne d'une donnée et l'utilisation de celle-ci par les actions.
Nous nous plaçons dans un contexte où les données sont séparées des actions qui les utilisent.

La description des données devra tenir compte du principe (lié à notre approche de conception) qui consiste à "piloter les actions par les données" (cf paragraphe IV du chapitre II) :



Le contrôle de l'application se fait en partant des données : une telle approche s'apparente aux "data flow languages" /DENNIS,75/.

II.1 DEFINITION D'UNE DONNEE

II.1.1 Sa structure : de quoi est-elle constituée ?

Une donnée dj appartenant à un niveau de commande est :
- soit composée d'une seule information élémentaire (auquel cas elle est dite "donnée simple") ;
- soit composée de plusieurs informations (auquel cas elle est dite "donnée composée").

$d_j = (C_1, \ldots, C_n)$ où $n \geq 1$

Les C_k sont les composantes de la donnée d_j

Exemples :
- Appelons DBLOOM la donnée composée associée au tableau de production des blooms(dans le cas de la coulée continue). Cette donnée peut être représentée par :

$DBLOOM = (V_1, V_2, V_3)$

$V_i = (V_{i,1} \ldots V_{i,n})$

n = nombre de lignes de coulée

V1,i , Vi,2 et Vi,3 représentent respectivement le nombre de blooms produits, la taille moyenne des blooms et les rejets pour la ligne de coulée n° i (i=1,...n).

- Appelons F-DEPAS-TEMP la donnée composée associée à un fichier statistique dans lequel on stocke les températures mesurées et qui sont en dehors de l'intervalle de sécurité, suivies des instants de dépassement de température.

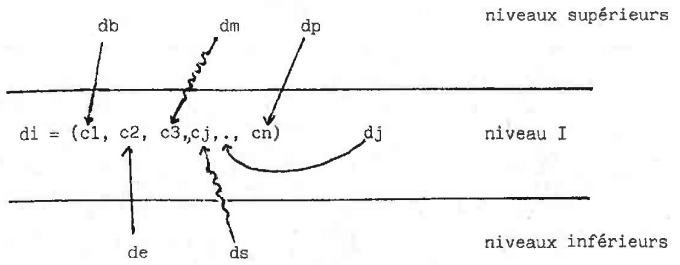
F-DEPAS-TEMP $(E_1, \ldots E_n)$

n = nombre d'enregistrement du fichier.

$E_i = (TP_i, T_i)$ TP_i : température mesurée à l'instant Ti

Des deux exemples précédents (et en général pour toutes les données composées) nous pouvons déduire que :

1. La struture d'une donnée de la BDI correspond à la notion de RECORD de PASCAL /WIRTH, 71 B/.
2. Les composantes d'une donnée d'un niveau I peuvent être :
   - soit des données d'autres niveaux (ou du même niveau), c'est le cas des mesures de températures enregistrées dans le fichier associé à la donnée F-DEPAS-TEMP;
   - soit des informations obtenues en appliquant une transformation à des données appartenant à d'autres niveaux (ou au même niveau) ou à des signaux en provenance de l'environnement.



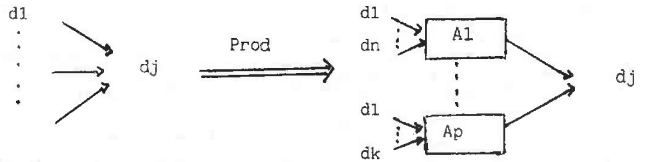
→ ; signifie qu'une donnée db apparaît directement dans la donnée di ; la donnée db est imbriquée dans la donnée di.

⇝ : signifie qu'une donnée apparaît indirectement (c'est-à-dire participe à la construction de la donnée dj mais moyennant une transformation) dans la donnée di .

Une composante C_k peut être aussi obtenue par transformation de plusieurs données.

II.1.2 Sa production: Qui produit la donnée ? et quand la produit-il ?

Connaissant les composantes d'une donnée (ie les données ou signaux qui composent la donnée) le concepteur définit la (ou les) action(s) qui permet(tent) de produire cette donnée. De telles actions sont dites des "producteurs" de la donnée ou "sources de production" de la donnée.



a) Producteurs d'une donnée : définition
Une source de production d'une donnée dj permet de transformer une ou plusieurs données (ou des signaux en provenance de l'environnement) rentrant dans la composition de dj pour obtenir de nouvelles valeurs de la donnée dj.

Les producteurs de données peuvent être classés en deux groupes et ce en fonction des informations qu'ils transforment :
- les producteurs (ou sources) internes,
- et les producteurs (ou sources) externes.

Sources externes
Pour acquérir des informations en provenance de l'environnement (opérateur et procédé) le concepteur définit des actions spécifiques des signaux à acquérir. Ces actions sont appelées sources de production externes.

Sources internes
Les données telles que les suivis, les statistiques et les commandes élaborées sont des informations obtenues par le système en effectuant des calculs internes (ie ne faisant pas appel au dialogue avec l'environnement).

Ces données sont dites à sources internes et les actions qui les produisent sont dites sources de production internes.

b) Détermination des producteurs d'une donnée : application PROD
Il s'agit, en fonction de la donnée, de déterminer la (ou les) action(s) nécessaire(s) à la production de la donnée.

On dira que la donnée dj est produite par les actions A1...Ap.

Chacune des Actions A1,...Ap permet de modifier toutes (ou une partie) des composantes de dj pour fournir de nouvelles valeurs de dj.

Les producteurs d'une donnée sont définis en tenant compte de :
- la complexité de la donnée
L'association d'actions de production à une donnée doit tenir compte de la complexité (ie de la structure) de la donnée pour obtenir des actions simples. Par exemple dans le cas de la coulée continue le synoptique général de la coulée peut être construit par plusieurs actions (une qui modifie l'état de la lingotière, une autre qui modifie l'état des zones de refroidissement,...).
- la sûreté de fonctionnement et des changements de mode de marche
Pour des problèmes liés à la sûreté de fonctionnement une donnée peut être produite de plusieurs manières possibles qui tiennent compte de l'état du système ; par exemple une consigne peut être élaborée par le niveau optimisation ou bien fournie par l'opérateur si le niveau optimisation n'est pas opérationnel.

Le concepteur peut donc associer à une même donnée plusieurs sources de production et choisir à un moment donné une source parmi ces sources et ce en fonction de l'état du système.

Remarques
1. Si la simplicité, la modularité et la sûreté de fonctionnement ne sont pas compromises il est préférable d'associer à une donnée, un seul producteur.
   L'association de plusieurs producteurs, à une même donnée, conduit à des difficultés d'ordonnancement des diverses valeurs produites par des producteurs parfois géographiquement répartis (c'est le problème de synchronisation dans les systèmes répartis)

c) Conditions de production d'une donnée
Une fois que le concepteur a déterminé l'ensemble des actions qui peuvent produire une donnée, il devra préciser les conditions dans lesquelles ces actions sont activées. En effet, les données de la BDI ne sont pas produites de manière quelconque mais plutôt selon un ordre que le concepteur précisera.

Une production ne peut avoir lieu qui si elle est, explicitement, attendue, par exemple :
. lecture d'une nouvelle température toute les 500 ms,
. rafraîchissement (mise à jour des informations) d'un synoptique toutes les 2 s.

il s'agit, pour le concepteur, de savoir si, à un instant t donné, une source de production Ai d'une donnée dj, peut, ou non, délivrer une nouvelle valeur de la donnée dj.

On parle alors de condition de production associée à l'action Ai qu'on notera Cp(Ai).

En tenant compte des caractéristiques des procédés industriels (cf chapitre I) cette condition peut correspondre, généralement :
- à un écoulement de temps ou à une date précise,
  par exemple tous les 500 ms, capter la température,
- ou à un changement d'état du système
  par exemple sur dépassement de température, mémoriser dans le fichier des défauts la température ainsi mesurée.

Nous reviendrons en détails sur la manière d'exprimer les conditions de production des données dans le prochain chapitre.

II.2 UTILISATIONS D'UNE DONNEE

II.2.1 Notion d'exemplaires d'une donnée

Avec l'évolution du temps, du procédé et de sa commande les données de la BDI changent d'état.

Par exemple Pt1,...Pt représentent les différentes valeurs mesurées, entre les instants t1 et t, pour une pression P.

Une donnée est considérée comme une suite, ordonnée, dans le temps, de valeurs : Vi1,...Vin

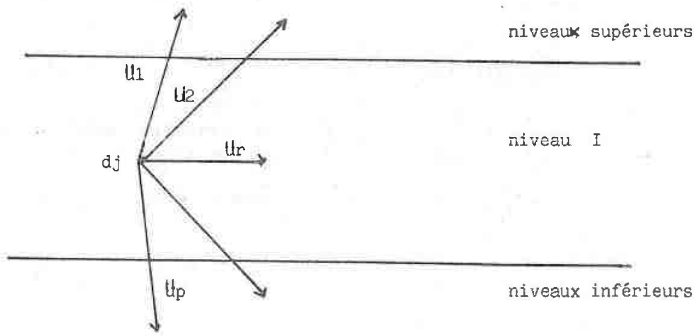


Chacune des valeurs de la donnée di est appelée exemplaire de di. Nous appelons, comme dans /THOMESSE,80/ , récent exemplaire et ancien exemplaire les valeurs respectives Vi1 et Vi,n.

II.2.2 Différentes utilisations d'une donnée

1) Utilisation pour la construction d'autres données

Une donnée dj produite à un niveau I est utilisée par les autres niveaux (et éventuellement par ce même niveau) pour assurer la commande globale du procédé, ou encore, en termes de données, pour produire d'autres données ou pour dialoguer avec l'environnement.



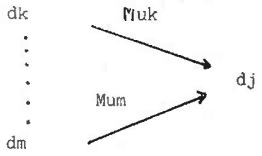
Par exemple une mesure de température peut être utilisée pour :
- la détermination des commandes à appliquer au procédé,
- la détection des défauts élémentaires,
- le déclenchement des alarmes,
- le suivi de produit,
- la visualisation de synoptiques,
- etc...

Il apparaît ainsi qu'une même donnée peut être utilisée à diverses fins. Mais en examinant les diverses utilisations d'une donnée, il apparaît que ces utilisations ne s'intéressent pas toutes aux mêmes exemplaires de cette donnée. C'est ainsi que, dans l'exemple précédent, le suivi de produit s'intéresse à toutes les mesures alors que la visualisation d'un synoptique s'intéresse à la dernière valeur présente au moment où on désire rafraîchir le synoptique.

En effet, certaines utilisations s'intéressent à la dernière valeur de la donnée, d'autres à la première valeur.

Conséquence sur la construction des données :
La construction d'une donnée dj est une forme d'utilisation d'autres données. Le concepteur devra alors préciser le mode d'utilisation de ces données pour construire la donnée dj.

MuK : signifie mode d'utilisation de la donnée K



La donnée dj se définit alors de manière statique par :
- les données dk, ..., dm
- et les fonctions d'utilisation de ces données $Mu_k, .. Mu_m$

2) Utilisation pour la synchronisation de l'application

Une deuxième utilisation des données réside au niveau du contrôle de l'application. Par exemple, dans la phrase suivante "si la température dépasse 1200°C, arrêter le four et avertir l'opérateur", la donnée Température n'est pas utilisée, par sa valeur, pour arrêter le four et avertir l'opérateur, mais plutôt lorsqu'elle dépasse 1200°C.

Si les données de la BDI vérifient des conditions particulières, elles peuvent conduire à l'activation de certaines fonctions du système.

II.2.3 Utilisateurs et modes d'utilisation d'une donnée : leur définition

1) Utilisateur d'une donnée

Toute donnée de la BDI est utilisée :
- soit pour construire une autre donnée,
- soit pour dialoguer avec l'environnement,
- soit pour synchroniser l'application.

Nous nous intéressons dans ce chapitre aux deux premières utilisations, l'aspect synchronisation sera traité dans le prochain chapitre.

Chaque action Ai qui utilise une donnée dj est dite "utilisateur" de la donnée dj.

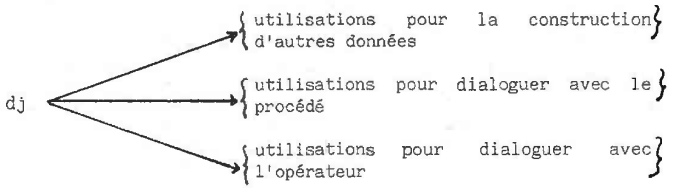
Les utilisateurs d'une donnée peuvent être classés en deux groupes :
- les utilisateurs externes,
- et les utilisateurs internes;

Utilisateur externe
La donnée est utilisée pour dialoguer avec l'environnement (par exemple application d'une commande au procédé ou affichage d'un synoptique). Cette utilisation nécessite l'emploi d'une action de sortie (ou d'envoi d'information à l'extérieur) qu'on appelle "utilisateur externe".

Utilisateur interne

La donnée est utilisée sans faire appel à la communication avec l'environnement, comme par exemple, pour l'accès aux mesures par les algorithmes de commande. Cette utilisation nécessite l'emploi d'une action interne dite "utilisateur interne".

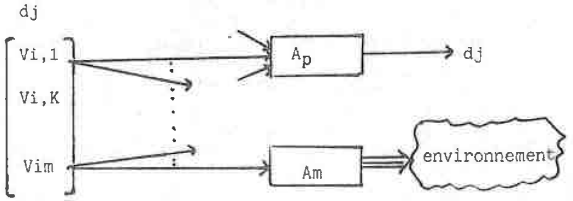
La distinction entre utilisateur interne et utilisateur externe permet au concepteur de préciser les utilisations de la donnée pour construire d'autres données et les utilisations de cette même donnée pour commander le procédé ou envoyer des informations à l'opérateur.



il est évident qu'une donnée peut n'être utilisée qu'à une seule fin (ie qu'elle a un seul utilisateur).

2) Modes d'utilisation d'une donnée

Etant donnée une suite de valeurs Vi,1,...Vi,n représentant l'état d'une donnée di, à l'instant t, l'utilisation de la donnée di, par l'un de ses utilisateurs, soulève les problèmes suivants :
- à quelle valeur cet utilisateur s'intéresse t-il ?
- s'agit-il d'une consultation ou d'une consommation ?



Les réponses aux deux questions précédentes dépendent de l'utilisation faite de la donnée (par exemple pour dialoguer avec le procédé on utilise toujours la dernière valeur alors que pour établir des statistiques on utilise toutes les valeurs).

En fonction de cette utilisation l'accès à une donnée peut correspondre :
- soit à la consultation d'un exemplaire de cette donnée,
- soit à la consommation d'un exemplaire de cette donnée.

La consultation
L'accès en consultation à une donnée permet de connaître la valeur d'un exemplaire (récent ou ancien) sans modifier la suite correspondant à cette donnée.

La consommation
Si l'on considère que l'état initial d'une donnée Di est représenté par la suite Vi,1,....Vi,n après la consommation :
- du récent exemplaire, la suite devient : Vi,1...Vi,n-1
- de l'ancien exemplaire, la suite devient : Vi,2...Vi,n
L'exemplaire consommé n'est plus accessible aux autres utilisateurs.

REMARQUE
Comme une donnée correspond à une suite d'exemplaires on a tendance à vouloir étendre le mécanisme d'accès à cette suite.

Etendre le mécanisme d'accès signifie chercher à accéder de manière aléatoire à un exemplaire J de la donnée (comme dans /ZAKARI, 84/).

Il est nécessaire de pouvoir autoriser de tels accès. Toutefois, il nous serait possible de les définir par le biais de la définition d'une autre donnée composée de n exemplaires d'une même donnée. Ces exemplaires deviennent des composantes de la donnée composée et l'accès à ces composantes peut être quelconque (c'est un accès à une structure composée).

Ceci nous permet de ne pas augmenter l'interface des actions (en leur rajoutant le paramètre J) :c'est là un aspect important de la modularité qu'est le couplage minimal /SAKHONN, 80/ , /DERNIAME, 74/.

Nous faisons remarquer, à ce sujet, que dans /THOMESSE, 80/ , /LADET, 82/ , /KRAMER, 81/ on opte pour une utilisation déterministe de la suite des valeurs (correspondant à une connexion ou à un port selon le mécanisme de communication retenu dans ces travaux) : c'est-à-dire l'accès est soit FIFO (First in First Out) soit LIFO (Last in First Out). Nous optons aussi pour l'utilisation déterministe.

II.2.4 Conditions d'utilisation d'une donnée

Comme pour la production, les conditions d'utilisation associées à une donnée expriment le fait, qu 'à un instant t donné, un utilisateur de cette donnée peut, ou non, accéder à la donnée.
On notera par Cu (Ai, dj) la condition d'utilisation de la donnée dj par l'action Ai.

Par exemple :
- toutes les 2 secondes, afficher la mesure de température de la lingotière,
- sur dépassement du seuil haut de la température, visualiser le message M à l'opérateur.

Les conditions d'utilisation d'une donnée s'expriment de la même manière que les conditions de production et seront détaillées dans le prochain chapitre.

II.3 MODELE DE DESCRIPTION DES DONNEES

Pour exprimer les divers attributs liés à une donnée
nous avons retenu le modèle suivant : toute donnée de la BDI est décrite par un 6-tuplet (N,V,C,S,P,U) où :
N : le nom de la donnée,
V : le niveau auquel elle appartient,
C : la liste des données qui la composent,
S : sa structure interne,
P : la liste de ses producteurs,
U : la liste de ses utilisateurs.

- le nom de la donnée est exprimé par :

DONNEE ‹identificateur› ;

le nom de la donnée est unique pour toute l'application.

- le niveau de commande auquel appartient la donnée est exprimé par :

DE NIVEAU ‹entier› ;

- la liste des données qui composent la donnée traitée désigne le flux de données qui intervient dans la construction de cette donnée. Cette liste est exprimée par :

COMPOSEE A PARTIR DE : { ‹identificateur›; }*

ou bien PHYSIQUE si la donnée correspond à un signal en provenance de l'opérateur ou du procédé.

- la liste des producteurs est exprimée par :

PRODUCTEURS = { ‹nom d'action› [/‹champs›]:‹condition de production›; }*

La condition de production est exprimée en termes d'évènements-temps-conditions que nous détaillerons au cours du prochain chapitre.
‹champs› désigne le ou les champs de la donnée qui sont modifiés par l'action. ‹champs› est absent si toute la donnée est modifiée par l'action (ie que tous les champs sont produits par l'action);
- la liste des utilisateurs est exprimée par :

UTILISATEURS = { ‹nom d'action›[: ‹condition d'utilisation›]

avec { consommation / consultation } du { récent / ancien } exemplaire ; }*

- la structure de la donnée désigne :
. d'une part la structure des exemplaires de la donnée,
. d'autre part le mode de gestion de la suite représentant la donnée :
si toutes les utilisations de la donnée sont de la forme "consultation du récent exemplaire", la donnée est gérée en mode VARIABLE (puisqu'on a toujours besoin de la dernière valeur uniquement) sinon elle sera gérée en mode SUITE.

La structure d'une donnée est exprimée par :

STRUCTURE { VARIABLE / SUITE } de < type des exemplaires >;

"type des exemplaires" est pris,ici, au sens type de PASCAL /WIRTH, 71 B/.

Récapitulatif

Le schéma de description d'une donnée de la BDI est le suivant :

DONNEE < nom de donnée>;

DE NIVEAU <entier>;

{ COMPOSEE A PARTIR DE: {<nom de donnée>}* / PHYSIQUE ;

STRUCTURE { VARIABLE / SUITE } de < type des exemplaires >;

PRODUCTEURS = {<nom d'action> [<champs>] : <condition de production>;}*

UTILISATEURS = {<nom d'action> [<condition d'utilisation>]

AVEC { CONSULTATION / CONSOMMATION } du { RECENT / ANCIEN } EXEMPLAIRE ;}*

FIN-DONNEE < nom de donnée >;

Propriétés du modèle de description des données

Le modèle de description de données permet :

1. un contrôle statique des accès aux données : les listes des producteurs et des utilisateurs permettent de préciser explicitement, pour chacune des données de la BDI, les actions qui ont le droit de lui accéder.

2. Une aide à la validation et au contrôle de l'application : en construisant le graphe associé au flux de données correspondant à un niveau ou à toute l'application on peut :

- vérifier si chaque donnée est bien produite (et utilisée) par au moins une action,
- vérifier la correspondance entre les services que doit fournir le système et ce flux de données. Il s'agit en d'autres termes de vérifier si le cahier des charges a été respecté et entièrement traité ou non.

III DESCRIPTION DES ACTIONS

III.1 STRUCTURE DES ACTIONS

1) Types d'actions

Toute action a pour objectif la manipulation et la transformation d'informations pour mettre à jour la BDI ou bien pour communiquer avec l'environnement. Elle ne peut assurer un travail déterminé que si elle reçoit des informations en entrée et elle ne peut participer à la commande globale que si elle communique ses résultats au reste de l'application.

Dans une approche de conception données/actions, les entrées (respectivement les sorties) d'une action peuvent être :
- des données de la BDI,
- ou des signaux en provenance (respectivement à destination) de l'environnement.

Ces entrées et sorties définissent l'interface de l'action.

Selon les types des entrées et des sorties associées à une action on peut regrouper les actions en trois groupes :
- les actions d'entrée : elles sont associées à des jeux d'entrée en provenance de l'environnement,
- les actions de sortie : elles sont associées à des jeux de sortie vers l'environnement,
- et les actions internes qui n'impliquent aucune communication avec l'environnement pendant leur exécution..

Cette distinction entre les actions intervient surtout au moment de l'implantation de l'application (généralement les opérations d'entrées/sorties sont programmées dans un langage assembleur et dépendent de la machine cible alors que les autres algorithmes sont programmés à l'aide d'un langage de haut niveau).

2) Comportement d'une action

Chaque action est définie par un ensemble de traitements enchainés d'une manière spécifique à cette action. Nous dirons qu'il s'agit d'actions multiprocédures.

a) Son interface

L'approche de décomposition que nous avons développée au chapitre II consiste à obtenir des actions élémentaires ayant un sens vis-à-vis de la commande globale. Le sens attribué à une action (dans une approche de conception orientée données) est celui de la donnée dont cette action est productrice d'où :
- une action produit une et une seule donnée si cette action est une action d'entrée ou une action interne, par exemple,
  - un algorithme PID pour le calcul de la vitesse à appliquer à un moteur,
  - l'acquisition d'une température.
- une action utilise, une et une seule donnée si cette action est une action de sortie, par exemple,
  - déclenchement d'une alarme (la donnée utilisée est constituée des informations relatives à l'alarme),
  - ou visualisation d'un synoptique (la donnée utilisée c'est les informations relatives au synoptique).

b) Fonction réalisée par l'action

Toute action est introduite par le concepteur :
- soit pour produire une donnée de la BDI (action interne ou action d'entrée),
- soit pour communiquer des informations à l'environnement (action de sortie).

Nous ne faisons, ici, aucune hypothèse sur la manière dont la concepteur enchaine les différents traitements pour définir les actions.

Cependant pour prendre en compte les situations des mauvais fonctionnements internes aux actions, par exemple,
- la panne d'un capteur qui a comme conséquence l'impossibilité d'acquérir une nouvelle mesure via ce capteur,
- la coupure d'une liaison entre deux sites qui empêche les actions d'un site, d'obtenir leurs entrées si elles sont implantées sur l'autre site,

Le concepteur devra préciser l'état du résultat de l'action afin d'éviter de prendre en compte des sorties erronées.

Le résultat d'une action peut être :
- valide : il est donc pris en compte par le système,
- ou invalide : il n'est pas pris en compte.

Pour préciser que l'état d'une action est invalide, le concepteur utilisera l'instruction DEFAUT dans le corps de l'action (le résultat valide est pris implicitement) . le langage de programmation retenu pour la description des actions devra posséder (ou devra être étendu par) l'instruction DEFAUT.

Ce qui signifie que parmi les enchaînements possibles des traitements d'une action, il doit y exister, au moins, un qui contient l'instruction DEFAUT si l'action peut se trouver en situation de défaut.

III.2 ACTIVATION DES ACTIONS

Le concepteur dispose d'un ensemble d'actions. Il doit décrire les circonstances dans lesquelles ces actions peuvent être activées.

L'activation d'une action implique des changements d'état du système. En d'autres termes (puisqu'on utilise une BDI), l'activation d'une action permet de changer l'état de la BDI.

D'après le paragraphe précédent, nous avons dit qu'une BDI
n'est accessible qu'à des moments particuliers durant la vie du procédé.

Ce sont ces moments qui permettent de décrire les règles d'activation des actions (rappelons que notre approche consiste à piloter les actions par les données).

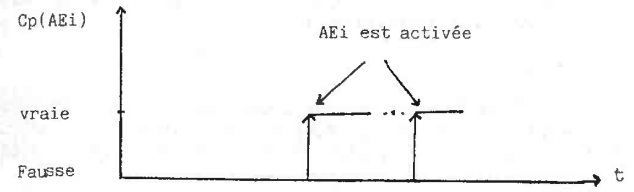
Voyons maintenant les circonstances dans lesquelles une action d'entrée, de sortie ou interne peut être activée.

1) Activation des actions internes

Une entrée en provenance de l'environnement peut être :
- le résultat d'une demande du système (par exemple lecture d'une nouvelle température),
- ou bien à l'initiative de l'opérateur (par exemple demande d'affichage d'un synoptique).

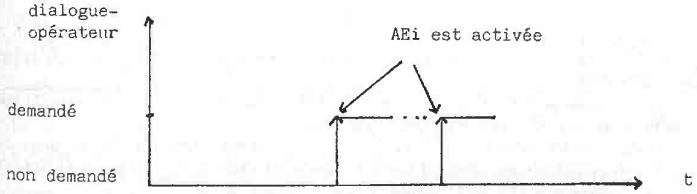
Ceci nous conduit à distinguer deux types d'activation des actions d'entrée :

a) l'action interne (soit AEi) est activée parce que le système a besoin de ses résultats pour mettre à jour la BDI. Nous dirons, dans ce cas, que l'activation de l'action est subordonnée à la réalisation de la condition de production de la donnée dont elle est productrice (il s'agit de la condition de production de AEi = Cp (AEi))



Exemple
la phrase "prélever la mesure de température tous les 2 ms" implique l'activation de l'action d'acquisition de température tous les 2 ms.

b) l'action d'entrée (soit AEi) est activée sur détection d'une demande de dialogue opérateur. Ce dialogue se fait de manière aléatoire; on peut l'exprimer (pour le moment) par :
si dialogue-opérateur demandé alors activer AEi



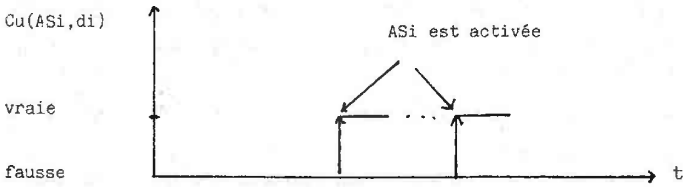
Ceci sous-entend évidemment l'existence d'un moyen de détection du début du dialogue-opérateur.

2) Activation des actions de sortie

Une sortie vers le procédé (ou vers l'opérateur) est parfaitement connue par le système.

Une sortie devient possible dès le moment où la condition d'utilisation de la donnée à transmettre à l'extérieur devient vraie.
si Cu (ASi, di), alors activer ASi.

ASi est l'action de sortie qui communique la donnée di à l'environnement.

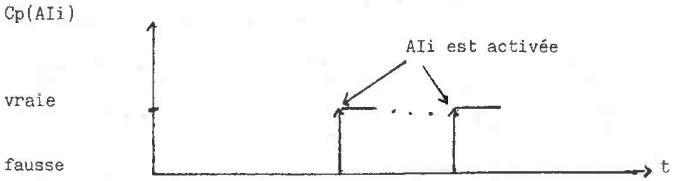


Exemple
- la phrase "appliquer une nouvelle tension tous les 50 ms"implique l'activation de l'action de sortie associée à l'application d'une tension tous les 50 ms ;
- la phrase "sur demande de l'opérateur afficherle synoptique x" implique l'activation de l'action d'affichage du synoptique x si la demande d'affichage du synoptique a été enregistrée.

3) Activation des actions internes

Une action interne (soit AIi) est activée chaque fois que ses résulats (ie sa sortie) sont nécessaires pour poursuivre la commande du procédé. La condition d'activation d'une action interne de production associée à cette action.

si Cp(AIi) alors activer AIi.



4) Règles d'activation
En vertu de ce qui vient d'être dit,les règles d'activation des actions sont les suivantes :
- règle ACT1 : une action d'entrée est activée chaque fois que la condition de production qui lui est associée est vraie,
- règle ACT2 : une action de sortie est activée chaque fois que la condition d'utilisation qui lui est associée est vraie (une action de sortie n'utilise qu'une seule donnée),
- règle ACT3 : une action interne est activée chaque fois que la condition de production qui lui est associée est vraie (elle est identique à la règle ACT2).

Conséquence
Les conditions d'utilisation d'une donnée ne sont obligatoires que dans le cas d'une action de sortie (elles constituent sa règle d'activation). Ainsi dans la description des données on ne précisera les conditions d'utilisation que pour les actions de sortie (car ces conditions sont redondantes pour les actions internes et inexistantes pour les actions d'entrée).

III.3 MODELE DE DESCRIPTION DES ACTIONS

Rappelons que la description d'une action se fait en trois étapes :
- l'étude de la partie statique permet de dégager le nom et l'interface de l'action ;
- l'étude de la partie contrôle (ou synchronisation) permet de dégager la règle d'activation de l'action ;
- et enfin une analyse interne de l'action permet de définir la structure interne de cette action.

Nous dirons qu'une action est décrite selon le modèle de description des actions si elle est décrite à l'aide d'un quadruplet (N,E,S,L,)
N : nom de l'action
E : vecteur d'entrée de l'action
S : vecteur de sortie de l'action (une seule donnée)
L : sa logique interne (ie son corps).

Ce quadruplet est décrit à l'aide du langage suivant :
- le nom de l'action
ACTION < identificateur > ;
le nom d'une action est unique pour toute l'application.
- l'interface de l'action
INTERFACE
ENTREES :{<nom de donnée>;}* | PHYSIQUE ;
SORTIE :<nom de donnée>; | PHYSIQUE ;
FIN-INTERFACE

Le mot clef PHYSIQUE désigne une entrée (ou une sortie)en provenance de (ou vers) l'environnement;
- le corps de l'action
Pour la description des actions (ie de leur comportement interne) nous préconisons l'emploi d'un langage de haut niveau : le langage PASCAL /WIRTH,71/

Les points suivants justifient le choix du PASCAL :

. il permet une programmation structurée par raffinements successifs,
. il utilise des variables typées utiles (pour la manipulation des données composées),
. il est largement répandu et peut aider à la "portabilité" des applications,
. sa syntaxe est actuellement bien adaptée à la description des algorithmes séquentiels.

Le langage PASCAL s'adapte bien aux traitements indépendants des machines cibles. Or les opérations d'acquisition (ou de transmission) d'informations en provenance (ou à destination) de l'environnement dépendent, en grande partie, des machines cibles puisqu'elles sont généralement écrites en assembleur. De plus les traitements spéciaux (tels que la gestion des fichiers) utiles à la mise à jour de la BDI ne s'expriment pas directement en PASCAL.

Pour tenir compte de ces cas particuliers la description des actions se fait à l'aide du langage PASCAL mais avec des appels à des opérations spéciales écrites dans d'autres langages (essentiellement l'assembleur).

d) récapitulatif
Une action se décrit à l'aide du schéma suivant :

```
ACTION          < identificateur >;
INTERFACE
        ENTREES :{< nom de donnée >;}+ PHYSIQUE;
        SORTIE  : < nom de donnée > PHYSIQUE;
FIN-INTERFACE;
        CORPS;
     < description des algorithmes réalisant l'action >
FIN-ACTION  < identificateur >;
```

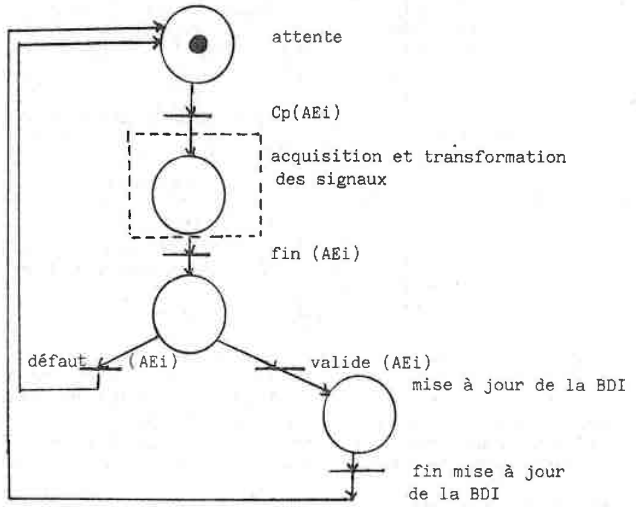
III.4 REGLES ASSOCIEES AUX ACTIONS

Nous résumons, ici, l'ensemble des règles à respecter lors de la description des actions

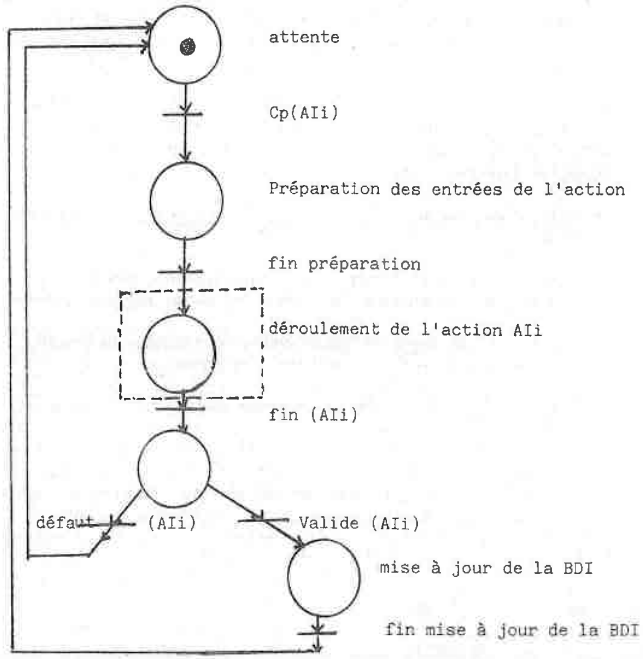
1. Une action interne possède au moins une donnée en entrée et une et une seule donnée en sortie,
2. une action de sortie possède une seule donnée en entrée,
3. une action d'entrée possède une seule donnée en sortie,
4. une action se comporte comme une machine cyclique,
5. l'activation d'une action se fait à partir de la BDI et selon les règles ACT1, ACT2, et ACT3 (vues au paragraphe III.2) ;
6. le temps d'exécution des actions n'intervient pas dans la définition de ces actions,
7. les données utilisées par une action interne (ou de sortie) doivent être prêtes avant d'activer cette action (ie qu'on "remplit" les entrées de l'action par les exemplaires des données exprimées dans les règles d'utilisation des données en entrée de l'action avant de l'activer),
8. une action est indivisible (ie non observable pendant son exécution),
9. à la fin de l'exécution d'une action d'entrée (ou d'une action interne) la BDI est mise à jour si le résultat de cette action n'a pas été déclaré invalide (dans l'autre cas le résultat est rejeté),
10. le comportement interne d'une action doit être tel que les anomalies de fonctionnement liées à cette action sont prises en compte et ce en rendant inutilisable l'exemplaire produit par l'action,
11. l'état "défaut" d'une action ne conduit pas à l'exécution d'une séquence d'exception (au sens d'ADA /ICHBIAH, 79/) appartenant à cette action mais, peut conduire à l'activation d'une action qui traite le mauvais fonctionnement, ayant conduit à l'état de défaut : les mauvais fonctionnements sont pris en compte par les mêmes mécanismes que le fonctionnement normal.

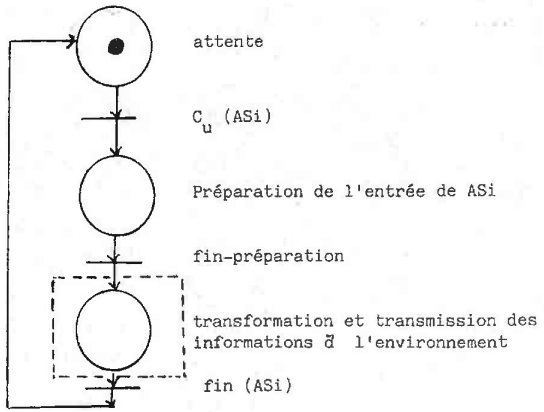
Le comportement des trois types d'actions est résumé par les réseaux de PETRI synchronisés et interprétés /MOALLA, 78/ suivants :

- pour une action d'entrée (AEi)



- pour une action interne (AIi)





Remarque
La fonction réalisée par une action telle qu'elle est décrite par le concepteur, est représentée par la partie encadrée du graphe. Les autres opérations (qui constituent le prologue et l'épilogue de l'action) sont réalisées implicitement par le système de gestion de la BDI que nous verrons dans la deuxième partie de ce travail.

IV QUELQUES COMPARAISONS

IV.1 LES BDI : COMMENT SONT-ELLES VUES PAR AILLEURS ?

Dans le domaine de la commande de procédés industriels, beaucoup de propositions ont vu le jour (nous en avons fait un résumé au chapitre I). Nous nous intéressons ici à la place qu'occupent les données dans ces travaux.

Dans les travaux tels que /THOMESSE, 78,80/ , /LADET, 77, 82/ , /DERNIAME, 82A,82B,82C,83/ , /KRAMER 81,82/ , /CALVEZ, 82/ , /ZAKARI, 84/ des méthodes et des outils ont été proposés pour la conception et la programmation des ACPI.

Ces approches reposent sur le concept de module et de communication entre modules.

un module se définit (comme une action ou un ensemble d'actions selon notre approche) par ses entrées, ses sorties et sa logique interne.

Les applications sont définies par des réseaux de modules entre lesquels existent des relations de partage d'informations.

L'élément principal de définition des applications, c'est le module.

Au module sont associés des attributs exprimant les communications d'informations et les synchronisations entre les divers modules d'une application. L'activation d'un module implique l'utilisation des informations en entrée pour produire des résultats. La communication inter-modules se fait, généralement, par l'intermédiaire de ports /SILBERSCHATZ 81/ , /KRAMER, 82/ , /MAO, 81/ etc... et de zones communes.

Les règles d'activation des actions sont :
- soit basées sur les concepts d'évènements et de conditions (comme ce que nous présentons ici) (THOMESSE et LADET),
- soit basées sur les types de messages reçus par les modules /KRAMER, 81,82/ , /LISKOV, 79,81/ , /DERNIAME, 82A,82B,82C,83/ , /ZAKARI, 84/ , /CALVEZ, 82/ c'est-à-dire qu'à chaque type de message reçu correspond un traitement dans le module récepteur du message.

En règle générale ces travaux mettent plus l'accent sur les traitements et sur le contrôle de ces traitements que sur les informations utilisées (c'est une vue duale à celle que nous proposons ici).

Même si ces informations forment une base de données industrielles, ils ne le disent pas de manière explicite.
Cependant il faut signaler que l'approche de description d'ACPI, présentée dans ce travail, repose sur les concepts développés dans ces travaux (il s'agit essentiellement des aspects de modularité, de communication et de synchronisation exprimées à l'extérieur des corps des modules

Si le concept de BDI est absent dans ces travaux, il l'est moins dans les produits industriels. En effet, certains produits industriels incorporent la notion de BDI et leur architecture est fondée sur cette notion de BDI. C'est le cas, notamment, d'ICS /ICS,75/ de télémécanique.

ICS (Industrial Control System) est un système de commande de procédés composé de deux ensembles (voir figure ci-après) :
- un ensemble de données industrielles appelé base de données d'ICS,
- et un ensemble de traitements (acquisition de mesures, algorithmes PID, etc...).

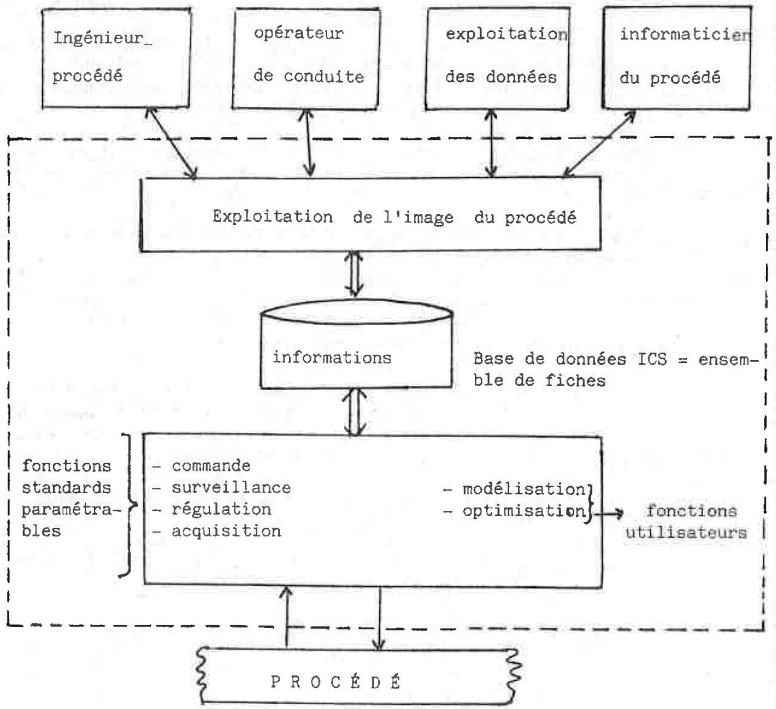
La BD d'ICS contient un ensemble de FICHES d'identification.

Chaque fiche peut identifier :
- une mesure, une commande, un défaut,
- une information calculée,
- etc...

L'ensemble des fiches permet de représenter le passé et l'état courant du procédé et de sa commande. Toutes les fiches sont composées selon des modèles prédéfinis.

C'est l'ingénieur procédé qui crée :
- les modules de traitement,
- et les fiches d'identification,
en fonction des services demandés au système de commande.

Les données emmagasinées dans la BD d'ICS sont accessibles aux opérateurs de conduite, aux automaticiens etc...



Architecture d'un système utilisant ICS

Chaque module se décrit par :
- sa structure interne (enchaînement d'algorithmes),
- ses entrées et ses sorties,

et chaque variable, identifiée par une fiche, se décrit essentiellement par :
- un nom,
- une valeur et un type,
- des seuils de sécurité,
- une fréquence d'élaboration et un module qui élabore la variable.

Cependant ICS est surtout un "Fill in the blanks language" (c'est-à-dire un progiciel) et ne propose pas de démarche de conception de l'ACPI.

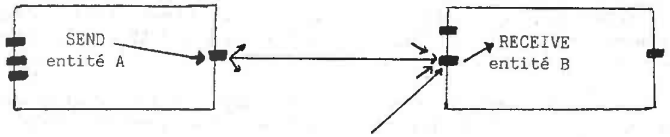
Notre approche se veut plus générale et cherche à aider le concepteur dans sa tâche de fabrication d'ACPI.

IV.2 APPROCHE ORIENTEE COMMUNICATION ET NOTRE APPROCHE

Par approche orientée communication nous désignons l'ensemble des travaux dans lesquels les applications se décrivent à l'aide d'un ensemble d'entités (ou tâches, ou modules, ...) qui se synchronisent par échanges de messages.

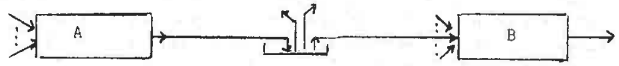
Dans ces travaux on rencontre d'une part un ensemble de langages DPL /BRINCH,78/, GYPSY /AMBLER, 77/, ADA /ICHBIAH, 79/, CSP /HOARE, 78/ PLITS /FELDMAN, 78/ , /LISKOV,79,81/ et d'autre part des approches de conception fondée sur la communication /DERNIAME,82A,82B/,/KRAMER,82/ ,/LISKOV,79,81/.

Comme nous venons de le dire, dans de tels travaux les entités se décrivent à l'aide de leurs entrées, de leurs sorties et de leur logique interne. Les envois et réceptions de messages sont explicités à l'intérieur des entités communicantes (généralement par les primitives SEND et RECEIVE).
Entre les entités communicantes, peuvent exister des liaisons bipoints, multipoints, monodirectionnelles et/ou bidirectionnelles.



Les messages reçus sont consommés dans leur ordre d'arrivée.
Voyons maintenant comment se pose le problème de la communication dans notre cas.

Nous définissons une application à l'aide d'un ensemble de données et d'un ensemble d'actions. A partir des producteurs et des utilisateurs des données nous déduisons l'ensemble des connexions existant entre les actions.



Les communications sont donc déduites à partir des productions/utilisations des données.

Les liaisons entre les actions peuvent être bipoints ou multipoints mais toujours monodirectionnelles. Les données constituent en quelque sorte des points de concentration/diffusion des messages échangés entre les actions (une donnée, a pour nom, le sens de "donnée transmissible" de SYGARE /THOMESSE,80/).

L'envoi et la réception de messages (ie la réception des entrées et l'envoi de la sortie d'une action) sont implicites : lorsqu'une action se termine, elle transmet automatiquement son résultat (il n'y a ni opération SEND ni opération RECEIVE).

De plus l'accès aux messages (ie les exemplaires pour nous) relatifs à une connexion (ie une donnée pour nous) peuvent correspondre à des consommations ou des consultations du plus récent message (ou du plus ancien message) reçu sur cette connexion .

Les échanges entre les actions se font via des données, ce qui signifie que les actions ne se connaissent pas mutuellement (il n'y a pas de désignation des partenaires pour effectuer des échanges entre actions).

Par ailleurs, les moments de communication (ils correspondent aux moments de mise à jour de la BDI) sont définis à l'extérieur des actions (la partie contrôle est définie à part) alors que, dans les travaux fondés sur la communication (ceux cités précedemment) ces moments sont exprimés à l'intérieur des entités communicantes en fonction des messages reçus (le comportement d'une entité dépend des messages qu'elle reçoit).

IV.3 CONCEPTION ORIENTEE BDI ET LANGAGES ORIENTES OBJETS

La conception orientée BDI repose sur l'utilisation d'objets (données et actions) et permet de définir une ACPI à l'aide de ces objets et de leurs comportements. Elle se place, ainsi, entre deux courants :
- celui des approches classiques de conception d'ACPI (évènements / actions) puisque les applications qu'elle permet de décrire sont des ACPI;
- et celui des approches de conception orientées objets (utilisées essentiellement en intelligence artificielle /LAURIERE, 82/) puisqu'elle repose sur l'utilisation d'objets.

Nous avons déjà situé notre travail par rapport à la première catégorie d'approches, nous cherchons, ici, à comparer notre approche à celle fondée sur les Langages Orientés Objets (LOO).

1) Notion de programmation orientée objet

La programmation orientée objet a les mêmes buts que la programmation structurée, elle se veut plus fonctionnelle qu'algorithmique et permet de construire les applications de manière progressive /ROCHE,84/.

Nous pouvons résumer l'approche LOO de la façon suivante : contrairement aux systèmes informatiques représentés par deux types d'objets (les données et les procédures), les systèmes conçus à l'aide de LOO sont représentés par un seul type d'objet. L'objet contient les données et les procédures d'utilisation de ces données (c'est le principe des types abstraits /DERNIAME, 79/ et des moniteurs /HOARE, 74/).

Le "représentant" actuel des langages orientés objet est le langage SMALL TALK /COINTE,83/ , /ROBSON,81/ ;Ses principes fondamentaux sont les suivants :
- tout objet se définit par un couple :
  OBJET=< ETAT, ACTIONS>
  l'état définit les connaissances (ou attributs) de l'objet.
  les actions déterminent l'ensemble des opérations que l'objet peut effectuer ;
- tout objet est activé à la réception d'un message (un message est défini par le nom de l'action à appliquer sur l'objet et les arguments à utiliser).
- Le comportement de l'objet dépend du type du message reçu (c'est l'idée retenue dans /KRAMER, 81/ dans ADA /ICHBIAH, 79/ et dans Distributed Process /BRINCH HANSEN, 78/);
- tout objet est une intanciation d'une classe d'objets (l'utilisateur crée et enrichit ses objets et ses classes d'objets).

2) Discussion

a) La définition d'une application en termes de données et d'actions permet de séparer les données des actions qui les exploitent (c'est l'idée que nous avons retenue). Le concept de données tel que nous l'avons défini correspond à la notion d'objet des LOO puisqu'une donnée se définit par :
. une structure (ou des attributs),
. et par l'ensemble des actions qu'on peut appliquer sur cette donnée (pour faire l'analogie avec les bases de données de gestion nous avons regroupé ces actions en producteurs et utilisateurs de la donnée).

b) Si, au niveau structuration, les LOO sont bien adaptés à la définition progressive d'applications modulaires ils sont quasiment inadaptés au temps réel. En effet :
- l'absence de la notion de temps,
- l'absence de la notion d'évènements,
- et l'absence de parallélisme,

rendent les LOO inefficaces pour la conception des ACPI dont les caractéristiques fondamentales résident justement dans ces trois points.

c) L'activation des actions dans les LOO se fait sur réception de messages dont l'émission est contrôlée à l'intérieur des actions (ie le contrôle se situe à l'intérieur des actions). Dans notre travail il y a séparation entre partie contrôle et partie opérative.

d) Les concepts que nous avons proposés peuvent être repris pour enrichir les LOO, par exemple SMALLTALK , par des moyens qui permettent de prendre en compte les aspects temps réel. Un LOO temps réel s'adapterait bien à la desription progressive et modulaire d'ACPI utilisant les BDI (ie fondées sur le concept d'objets).

e) Signalons, enfin, que l'approche d'activation des actions retenues ici (ie activation d'une action si la condition de mise à jour qui lui est associée est vraie) est analogue à celle utilisée pour la construction des systèmes experts /LAURIERE, 82/ (condition ⟶ action) à la différence près que dans notre cas il n'y a pas de retour arrière dans les actions : la recherche d'une solution, dans le cas des systèmes experts, se fait par exploration de plusieurs "chemins" conduisant à des retours arrières.

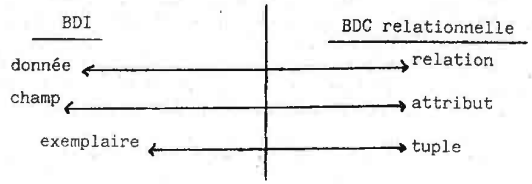
IV.4 BDI ET BASES DE DONNEES CLASSIQUES

Le modèle de description des ACPI que nous avons développé est similaire aux modèles de représentation et d'accès aux bases de données utilisés en informatique de gestion. Cette similitude apparaît essentiellement aux deux niveaux suivants :

1) Niveau modèle de représentation des données

Rappelons qu'il existe deux types de modèles de représentation des informations constituant une base de données classiques (ou de gestion) :
- le modèle entité/association /TARDIEU, 83/ , /ABRIAL, 74/ où les données peuvent être représentées sous forme d'une "arborescence" ou sous forme d'un "réseau".
- et le modèle relationnel /CODD, 70/ où les données sont représentées à l'aide d'un ensemble de "relations".

Les BDI telles que nous les avons structurées dans ce travail se rapprochent des bases de données relationnelles. En effet une donnée de la BDI est décrite à l'aide d'un ensemble de champs et peut être assimilée à une relation (au sens de CODD) et chaque exemplaire de cette donnée représente une occurrence de ces champs et peut être assimilé à un tuple (au sens de CODD).



2) Niveau accès aux informations

L'accès aux données d'une BDC s'effectue à l'aide de transactions construites à partir des requêtes de l'utilisateur.

Ces transactions sont activées sur initiative de l'utilisateur (ie lorsqu'une requête de l'utilisateur a été enregistrée) pour mettre à jour la BD. Les actions jouent un rôle analogue à celui des transactions pour la mise à jour de la BDI mais elles sont activées pour tenir compte, à la fois, de l'évolution du phénomène physique commandé et des ordres de l'utilisateur.

{actions} ⟷ {transactions}

V CONCLUSION

Nous venons de présenter un modèle de description d'ACPI qui présente les propriétés suivantes :
- séparation entre données et traitements,
- séparation entre partie contrôle et partie opérative,
- résolution des problèmes liés au partage d'objets à l'aide de règles de production et d'utilisation des données (ces problèmes sont la source de beaucoup d'anomalies dans les systèmes multitâches).

Ce modèle reprend des concepts divers développés dans les travaux relatifs à :
- la conception des bases de données classiques ;
- la conception d'applications selon une approche par les évènements ;
- la conception d'applications selon une approche par les communications,
- et à la conception d'applications fondée sur le concept d'objets.

Il permet, ainsi, de montrer que ces idées sont complémentaires. Nous avons tenté, dans le paragraphe précédent, de situer notre travail par rapport à ces quatre approches.

CHAPITRE IV

DESCRIPTION DES REGLES DE MISE A JOUR DES DONNEES : ASPECT DYNAMIQUE DES APPLICATIONS

I INTRODUCTION

Nous venons d'étudier la description des objets constituant une ACPI. Laquelle description nous a conduit à définir l'aspect mise à jour de la BDI. Aussi, nous avons signalé que les instants de mise à jour de la BDI correspondent aux instants d'activation des actions : la synchronisation de l'application est déduite de la mise à jour de la BDI.

L'objectif de ce chapitre est d'étudier les éléments utiles au concepteur pour déterminer les instants de mise à jour de la BDI qu'il décrira dans un langage que nous proposerons ici.

Pour déterminer les instants de mise à jour de la BDI le concepteur considère :
- l'écoulement du temps,
- les changements d'état du système,
- et l'état du système,

et ce, en se référant au cahier des charges décrivant l'aspect dynamique du procédé et des divers comportements du système de commande qui s'adaptent à l'état instantané du procédé évoluant dans le temps.

Exemples de comportements
1. Sur dépassement du seuil haut de la cuve X,fermer la vanne Y
2. sur arrivée du chariot au point A,vider le charriot
3. 5 minutes après avoir commencé à chauffer le liquide A, verser le liquide B et chauffer encore pendant 10 mn.

Les trois concepts (temps, état du système et changements d'état du système) peuvent être considérés séparément (exemples 1 et 2) pour décrire le comportement du système. Mais ils peuvent aussi être combinés (exemple 3) pour exprimer des comportements faisant intervenir plusieurs facteurs à la fois. Ces trois concepts seront étudiés séparément, dans un premier temps, pour déterminer les besoins que chacun d'entre eux permet d'exprimer et seront combinés, ensuite, pour décrire des règles de mise à jour des données de la BDI.

Le concept de temps inclut tout ce qui est périodique par nature. Il y a deux points de vue différents à ce sujet :
- considérer tous les évènements comme étant complétement asynchrones en faisant abstraction de la façon dont ils sont détectés puis choisir une détection périodique comme méthode de détection;
- séparer d'une part tout ce qui est périodique (et plus généralement tout ce qui est en relation directe avec le temps) et d'autre part, les évènements détectés véritablement de façon asynchrone.

Si cette deuxième solution a été choisie malgré le fait qu'elle ne fasse pas abstraction de la méthode de détection, c'est essentiellement pour tenir compte des contraintes et usages industriels. Une telle solution est en effet plus lisible et plus naturelle au niveau de l'expression.

On pourrait toutefois choisir la première solution comme le fait BENMAIZA / BENMAIZA, 84/ qui considère les instants précis, l'écoulement de périodes etc... comme des évènements.

Rappelons que nous avons déjà défini, au cours du chapitre précédent, un langage de description des objets dans lequel manquaient les règles de mise à jour des données ; c'est à l'expression de ces règles de mise à jour que nous nous intéresserons ici pour compléter le langage de description des ACPI. Ainsi nous obtiendrons une description d'ACPI où :
- la partie statique est définie à l'aide des concepts de données et d'actions ;

- et la partie dynamique est définie à l'aide des concepts de temps, d'évènements et de conditions.

Outre le langage de description des règles de mise à jour, nous proposerons une démarche de description de ces règles.

II PRISE EN COMPTE DE L'ECOULEMENT DU TEMPS

Nous avons signalé, dans le premier chapitre, que l'une des caractéristiques essentielles des ACPI est leur aspect temps-réel.

Par fonctionnement en temps-réel, on entend généralement une exécution d'opérations qui satisfait à des contraintes de temps.

De façon simple, on peut dire qu'un évènement ou un ensemble d'évènements demande, dès qu'il apparaît, l'exécution d'une opération, effective au plus tard après un délai Δt appelé temps maximum de réponse.

L'utilisation du concept temps pour savoir si une opération s'est achevée, ou non, est l'utilisation la plus fréquente de la notion de temps dans la commande de PI (essentiellement pour des raisons de sûreté de fonctionnement).

Cependant le concept de temps peut intervenir sous d'autres formes (par exemple "à 12 h arrêter le fonctionnement de la machine Y"). Nous cherchons, à travers ce paragraphe, à expliciter les principales manières de faire intervenir le temps lors de la description de l'aspect dynamique (évolution dans le temps) d'une ACPI.

Généralement, dans une ACPI, le temps intervient pour exprimer un des besoins suivants :
- une temporisation,
- une activation à un instant précis,
- une activation périodique,
- une utilisation d'un instant précis comme repère (ie activation d'une ou de plusieurs opérations à partir d'un instant précis : cet instant sert comme repère).

Voyons maintenant comment exprimer chacun de ces besoins.

1) Les temporisations

Une temporisation permet, à partir d'un instant X donné, d'attendre un laps de temps Δt avant de déclencher une certaine opération.

Exemple
La phrase "après avoir commencé à introduire le mannequin, si après un certain temps Δt le mannequin n'arrive pas à la position désirée, lancer l'opération qui traite cet incident" montre comment sont détectés les défauts de fonctionnement du mannequin.

L'instant X (qui est donné) sera exprimé à l'aide de la notion d'évènement qu'on verra dans le prochain paragraphe.

Nous exprimons une temporisation par :
ATTENDRE (<durée>)

2) Activation à un instant précis

On exprime à ce niveau le fait qu'une opération est activée à des instants définis par le concepteur.

Exemple
"A 12h arrêter l'exploitation et la relancer à 14h"
On décrit ceci par la structure de contrôle suivante :
A l'instant t_x

3) Activation périodique

Les algorithmes de régulation, les acquisitions et les envois des ordres au PI sont des opérations, souvent, exécutées de manière cyclique.

Nous exprimons ceci par :
tous les Δt "faire opération".

4) Un instant comme point de repère

On considère qu'un instant t_x est un point de repère pour un ensemble d'opérations données, si ces opérations ne peuvent s'exécuter qu'à partir de l'instant tx.

Ce besoin est exprimé par :
Après l'instant tx faire...

Exemple
.. produire des pièces de caractéristiques X à partir de 8h jusqu'à 20h et des pièces de caractéristiques Y à partir de ...

5) Remarque

L'utilisation du concept temps est insuffisante à elle seule pour exprimer les diverses situations pendant lesquelles le concepteur de l'ACPI désire activer des actions.

En effet, des phrases du style :
- "sur dépassement de température, déclencher l'alarme",
- ou "10 secondes après le passage du chariot au point A, déclencher l'opération X)

ne peuvent être exprimées en terme de temps uniquement. Les concepts de conditions et surtout d'évènements développés dans les prochains paragraphes, vont remédier à ces insuffisances.

III LES CHANGEMENTS D'ETAT OU EVENEMENTS : DEFINITION ET UTILISATION

II.1 NOTION D'EVENEMENTS

L'évolution permanente du procédé et de l'installation de commande est caractérisée par des instants particuliers pendant lesquels le PI et/ou l'installation de commande changent d'état (un dépassement de température, la fin d'usinage d'une pièce, la fin de calcul d'une valeur sont des exemples d'instants de changement d'état).

Ces instants de changement d'état sont importants pour le concepteur de l'ACPI puiqu'ils lui permettent de définir les instants de lancement des opérations de commande (en fonction de chacun de ces changements). C'est ainsi qu'on rencontre, dans les cahiers des charges, des phrases du style "sur dépassement de température, déclencher l'alarme", qui signifie que l'alarme n'est déclenchée que si on a détecté un dépassement de température (l'instant de dépassement de température étant un instant de changement d'état). Un changement d'état ne peut intervenir dans la commande que s'il a été pris en compte (ou détecté) par le système. Il est clair que le concepteur de l'ACPI ne s'intéresse pas à tous les changements d'état. Seul un sous-ensemble de ces changements lui est utile pour la définition de l'application (par exemple une température qui passe de 100° à 101° n'intéresse pas le concepteur alors que si cette même température passe de 999° à 1001° elle a une signification pour le concepteur si l'on considère que le seuil haut a été fixé à 1000°).

Définition

On appelle occurrence d'évènement, l'apparition d'un changement d'état et évènement l'ensemble des occurrences d'un changement d'état défini par l'utilisateur (ou le concepteur de l'application) /BENMAIZA, 84/.

Une occurrence d'un évènement est considérée comme un "signal" dont la prise en compte peut provoquer le déclenchement d'une opération particulière.

Exemple
Sur débordement de la cuve, fermer la vanne.

Beaucoup de travaux ont été consacrés à ce concept d'évènement pour définir et programmer les ACPI /NONN, 78/ , /LADET, 77,82/ , /THOMESSE, 80/, /BENMAIZA, 84/.

L'étude menée par BENMAIZA pour la définition des évènements au sein de notre laboratoire (ie le CRIN) pourrait être reprise ici telle qu'elle, mais nous avons choisi d'exprimer ces mêmes concepts autrement, de manière à tenir compte de l'approche de structuration données/actions developpée ici et de la lisibilité des applications.

Pour une étude approfondie du concept d'évènement, nous renvoyons le lecteur à l'étude développée dans /BENMAIZA, 84/.

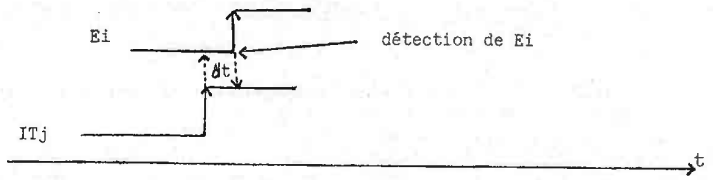
Nous préciserons ici les aspects relatifs aux évènements (ie leur source, leur détection et leur utilisation) pour pouvoir définir des règles de mise à jour des données.

III.2 SOURCES D'EVENEMENTS

Les diverses sources d'évènements (exepté le temps puisque nous séparons temps et évènement) développés dans /BENMAIZA, 84/ se ramènent dans notre cas à trois cas :
- évènements liés aux interruptions,
- évènements liés aux changements d'état des données de la BDI,
- évènements liés aux changements d'état des actions.

1) Evènement lié à une interruption

Un évènement lié à une interruption ITj est détecté pour chaque apparition de l'interruption ITj.

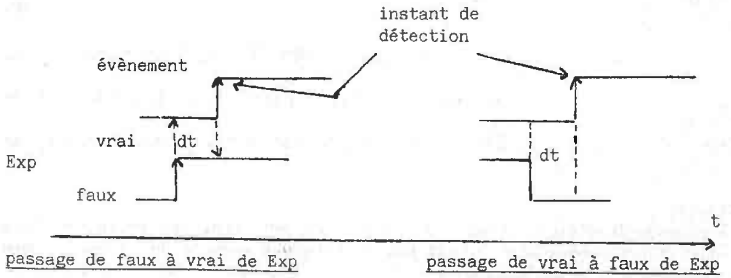


2) Evènement lié à la BDI

Un évènement lié à la BDI traduit un changement d'état d'une (ou de plusieurs) donnée (s) de la BDI.

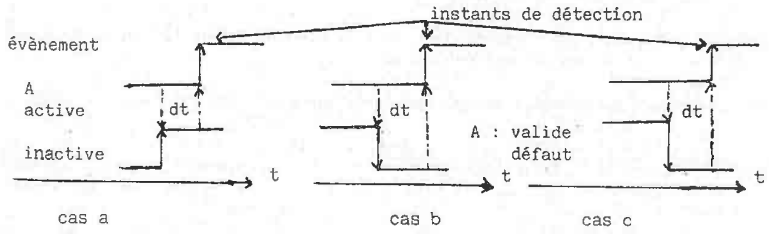
Exemple
" Un dépassement de température"

A chaque changement d'état d'une expression booléenne exp (qui porte obligatoirement sur les données de la BDI et/ou sur des constantes) il y a occurrence de l'évènement liée à cette expression.



3) Evènement lié aux actions

Un évènement lié à une action traduit un changement d'état de l'action c'est-à-dire :
. passage de l'état "actif" à l'état "inactif" et vice-versa (cas a et b) ;
. passage de l'état "valide" à l'état "invalide" ou "défaut" (cas c).



Description de la source d'un évènement

La source d'un évènement se décrit, en tenant compte des trois formes possibles qu'elle peut prendre, par la règle syntaxique suivante.

< déclaration d'évènement > :: = EVENEMENT < nom d'évènement > : < source > FIN-EVT ;
< source > : : = < EVT-IT > | < EVT-COND > | < EVT-ACT >

- < EV-IT > désigne un évènement de type apparition d'interruption qu'on décrit par :
IT (<numéro physique d'interruption >).

On pourrait associer des interruptions logiques aux interruptions physiques pour faire, à un certain niveau, abstraction du matériel.

- < EVT-COND > désigne un évènement portant sur l'état de la BDI (ou sur des constantes) que nous décrivons ainsi :
{VF / FV} (<expression logique>)
VF : signifie passage de "vrai" à "faux" ;
FV : signifie passage de "faux" à "vrai".
<expression logique> est une expression à valeur "vrai" ou "faux" et porte sur des données de la BDI et/ou sur des constantes ;
-<EVT-ACT >: désigne un évènement portant sur les changements d'état d'une action et se décrit par :
{FIN / DEBUT / ERREUR} (<nom-action>)

Fin (a) : désigne le passage de l'état "actif" à l'état "inactif" de l'action a
Début (a) : désigne le passage de l'état "inactif" à l'état "actif" de l'action a
Erreur (a) : désigne le passage de l'état "valide" à l'état "défaut" de l'action a.

REMARQUES
- le passage de l'état "défaut" à l'état "valide" n'est pas retenu car une action passe dans l'état valide dès qu'elle est activée (il serait alors redondant avec début (action)) ;
- une action est une tâche au sens informatique du terme (à ne pas confondre avec les étapes du GRAFCET ou les places des réseaux de PETRI); action active ou inactive signifie tâche active ou inactive ;
-la description de la source d'évènement décrite précédemment correspond au lien physique-logique développé dans /BENMAIZA, 84/ et exprimé par :
ASSIGNER < évènement > à < source>

III.3 DETECTION DES EVENEMENTS

Nous distinguons trois manières de détecter un évènement (et ce en fonction du type de la source de cet évènement).

1) Evènement dont la source est une interruption

Les interruptions étant reconnues par le système de gestion des interruptions de l'installation, la détection des évènements qui leur sont liés est garantie par le matériel (ie le système de gestion des interruptions).

2) Evènement lié à la BDI

Le principe consiste à tester la présence de l'évènement au moment de la mise à jour (modification) d'une donnée de la BDI : si la donnée rentre dans la composition d'un évènement, on teste alors si le nouvel exemplaire crée, ou non, une occurrence de l'évènement.

3) Evènement lié à une action

Au niveau exécution, les actions sont matérialisées soit par des tâches temps réel (si la machine support est munie d'un exécutif temps réel) soit par des sous-programmes (si la machine support est un automate programmable). Le début, la fin et la déclaration de l'état "défaut" d'une action sont des changements d'état connus par l'exécutif de la machine temps réel ou de l'automate programmable (puisque c'est lui qui est chargé de la gestion des actions, c'est-à-dire les lancer, les arrêter...).

Les évènements associés aux actions sont détectés au moment où l'exécutif lance, arrête, ... les actions.

Remarque
Une autre solution a été retenue dans /BENMAIZA, 84/ pour la détection des évènements : il s'agit de scruter (ou de tester) la présence d'un évènement à une fréquence précise et fixe pour cet évènement.

La solution que nous avons retenue (et qui est liée au fait que nous séparons temps et changement d'état) évite de préciser les fréquences d'échantillonnage qui ne sont pas toujours, à notre avis, simples à déterminer.

III.4 UTILISATION DES EVENEMENTS

1) Occurrences prises en compte

L'objet "évènement" défini par le concepteur ne suffit pas à lui seul, pour déterminer les instants de déclenchement des actions de mise à jour de la BDI.

La relation entre la prise en compte d'un groupe d'occurrences et son exploitation dépend de la nature des actions activées par ces occurrences. Selon le sens que donne l'utilisateur au lien évènement-action, il peut s'intéresser soit à la première occurrence de l'évènement soit à toutes les occurrences de cet évènement.

Ainsi l'apparition d'un défaut grave conduisant à l'arrêt total de l'application n'est pris en compte qu'une seule fois (la première occurrence) alors que l'évènement "fin d'usinage d'une pièce" est pris en compte à chaque fois qu'il apparaît.

Pour décrire le mode d'utilisation des occurrences d'un évènement nous retenons la syntaxe suivante :

{Pour chaque / sur première occurrence de} < évènement >

l'expression "sur première occurrence " sera, parfois, abrégée par "sur 1ERE OCCUR"

2) Composition d'évènements

Pour décrire son application, l'utilisateur a parfois besoin de déclencher une opération, non pas, sur détection d'un seul évènement, mais plutôt si une série d'évènements se sont succédés d'une certaine manière dans le temps.

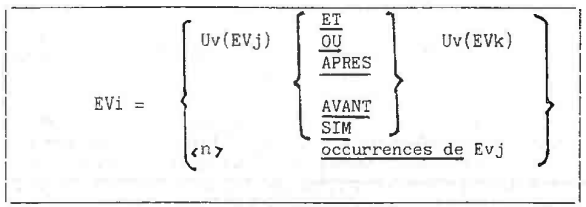
Par exemple
- soit un système d'emballage de produits; Si dix produits ont été emballés fermer la caisse (il s'agit du déclenchement de l'opération de fermeture de la caisse si l'évènement "fin d'emballage" d'un produit a été détecté dix fois),
- dans l'exemple de KRAMER /KRAMER, 82/. Si le niveau haut du méthane ou le niveau bas de l'eau sont atteints ou bien si l'ordre d'arrêt est donné, alors arrêter la pompe (il s'agit ici de l'arrêt de la pompe si l'un, au moins, des trois évènements (niveau du méthane atteint, niveau haut de l'eau atteint ou arrêt demandé par l'opérateur) a été détecté).

On remarque aisément à travers ces exemples que l'utilisateur a besoin de combiner les évènements pour décrire son application.

Pour uniformiser la démarche de description d'une application, une combinaison d'évènements sera considérée comme un évènement "complexe" dit évènement composé.

Un évènement composé est défini par une expression d'évènements construite à partir d'autres évènements et des opérateurs de composition.

Nous retenons ici les opérateurs de composition développés dans /BENMAIZA,84/ et qui permettent d'exprimer une règle de composition d'évènements comme suit :



. SIM : signifie que les deux évènements EVj et EVk sont simultanés (ie détectés au même instant) ;

. Evi est l'évènement composé à partir de EVj (et de EVk) ;

. Uv(x) ::= { pour chaque / sur première occurrence de } X

. la déclaration d'un évènement composé se fait selon la règle suivante :
⟨évènement composé⟩ ::= évènement composé ⟨momevt⟩ : ⟨composition⟩ FIN-EVT ;
⟨composition⟩ ::= ⟨n⟩ occurrences de ⟨nom-evt⟩ | ⟨EV1⟩ ⟨opérat⟩ ⟨EV1⟩

⟨EV1⟩ ::= pour chaque ⟨nom-evt⟩ | sur première occurrence de ⟨nom-EVT⟩ | (⟨composition⟩)

⟨operat⟩ : : = ET | OU | AVANT | APRES | SIM

3) Le partage d'évènements

Dans /BENMAIZA, 84/ on considère qu'un évènement est une suite d'occurrences d'un même changement. Cette suite peut être utilisée pour activer plusieurs opérations différentes (toutes, ou une seule, à la fois) : on dit alors que l'évènement est partagé .

Les règles de partage résident au niveau de l'expression :
- de la consommation et/ou de la consultation des diverses occurrences de l'évènement ;
- et de la duplication ou non, de ces occurrences aux diverses opérations qui les utilisent

et on obtient alors des expressions du style

EVENEMENT... [PARTAGE] [DUPLIQUE]
{AVEC / SANS} CONSOMMATION ....

Cette solution a l'avantage d'expliciter de manière claire l'utilisation d'un évènement. Cependant les attributs qu'elle introduit sont plus proches du langage des informaticiens que de celui des automaticiens.

Nous avons évité de faire apparaître ces attributs pour obtenir une description d'ACPI plus naturelle et ce en tenant compte des points suivants:

a) un évènement apparaissant dans plusieurs règles de mise à jour et/ou dans plusieurs compositions d'évènements est implicitement partagé entre ces règles de mise à jour et/ou compositions d'évènements ; il est aussi implicitement dupliqué.

Exemple soit la phrase suivante :
"sur dépassement de température, avertir l'opérateur, enregistrer la mesure de température dans le fichier des anomalies et arrêter le système de chauffage".

L'évènement "dépassement de température" est utilisé à trois fins différentes. Il est alors implicitement partagé par (et dupliqué pour) ces trois utilisations : les règles de mise à jour associées aux actions de modification du fichier des défauts, d'arrêt du système de chauffage et d'avertissement de l'opérateur seront toutes construites à partir de l'évènement "dépassement de température".

b) Une occurrence d'évènement est un signal qui peut, ou non, provoquer l'exécution d'une (ou de plusieurs) opération(s).

Comme le sens évènement ⟶ action tient compte des faits suivants :
- une occurrence d'évènement est dupliquée pour toutes les opérations pouvant être activées sur apparition de cet évènement ;
- à l'apparition d'un changement d'état donné, un ou plusieurs traitements peuvent être déclenchés. Mais chacun de ces traitements n'est déclenché qu'une seule fois pour la même apparition du changement d'état ie une apparition d'un changement d'état ⟶ un traitement 1, un traitement 2...

Toute copie d'une occurrence d'évènement (copie dupliquée pour une opération donnée) est implicitement consommée par l'activation de l'opération possédant cette copie.

Dans notre cas une occurrence d'un évènement est utilisée une et une seule fois pour évaluer chacune des règles de mise à jour (ou de composition d'évènements) dans lesquelles apparaît cet évènement.

III.5 CONCLUSION

Le concept d'évènement est fondamental en commande de PI : les évènements traduisent les instants de changements d'état qui ont une signification vis-à-vis de la commande.

L'ensemble des évènements relatifs à une ACPI, constitue un "point dur" de celle-ci puisque, presque toutes les opérations sont activées sur changement d'état (on a bien dit "presque" parce que, comme on le verra après, les évènements sont parfois insuffisants, à eux seuls, pour exprimer certaines conditions d'activation d'opérations).

Il est donc naturel qu'une méthode et qu'un langage de description d'ACPI soient fondés sur le concept d'évènement.

Déjà au niveau programmation, beaucoup de langages temps réel sont basés sur le concept d'évènement.

Nous avons, à travers ce paragraphe, donné une manière de décrire et d'utiliser les évènements, nous verrons dans le prochain paragraphe comment combiner évènement-condition-temps pour exprimer les divers cas de synchronisation des opérations d'une ACPI.

IV PRISE EN COMPTE DE L'ETAT DU SYSTEME : LES CONDITIONS

L'état du système est caractérisé, à un instant donné, par :
- l'état de la BDI,
- l'état des différentes actions constituant l'application (actions actives, actions inactives et actions en défaut) ;
- et l'état des divers niveaux d'interruptions.

Les automaticiens traduisent l'évolution, dans le temps, du procédé et de sa commande grâce à la notion d'évènements et/ou de temps comme nous venons de le voir, mais aussi grâce à la notion de condition /THOMESSE, 80/ , /LADET, 82/ , /BENMAIZA, 84/.

Une condition porte sur l'état du système et est utilisée sous la forme suivante :

sur évènement si condition faire action

ce qui signifie qu'un évènement, ne suffit pas, à lui seul parfois, pour déclencher une opération : le déclenchement de l'opération exige, en plus de l'arrivée de l'évènement que certaines conditions soient vérifiées sur l'état du système.

Par exemple les opérations cycliques associées à la production ne peuvent être activées que si les matières premières, les machines... sont disponibles.

Comment exprimer une condition ?

Une condition porte sur l'état du système (ie sur l'état de la BDI, l'état des actions et l'état des interruptions), pour l'exprimer nous avons retenu la syntaxe suivante :

< condition > : : = < expression logique > | < expression logique >< op > < expression logique >

< op > : : = et | ou | non

<expression logique>: = <expr-const> |<expr-BDI> |<expr-ACT> |<expre-IT>

<expr-const> : : =<constante > <opr > <constante>

< opr>: : = > | < |>=|<=| ≠ |

<expr-BDI>: : = <terme><opr><terme >

<terme>: : = <constante> | <donnée de la BDI>

<expr-ACT> : : = ACTIVE (<nom action>)| INACTIVE (< nom action>)| DEFAUT (< nom action >)

< expr-IT > : : = SIGNALE (< niveau IT >)

V CONSTRUCTION DES REGLES DE MISE A JOUR

Nous venons de voir les concepts de temps, d'évènement et de condition et la façon dont ils peuvent être employés pour activer des actions.

Le concepteur qui a défini l'ensemble des données et des actions, du point de vue statique, continue son étude pour définir les conditions de mise à jour de la BDI à partir de ces trois concepts.

Nous voudrions donner, ici, non pas une méthodologie de description de la partie contrôle d'une ACPI mais plutôt les idées directrices d'une telle méthodologie.

Nous avons regroupé les conditions de mise à jour, en fonction des difficultés liées à leur description, en trois groupes :
- les mises à jour périodiques,
- les mises à jour à instants fixes,
- et les mises à jour aléatoires.

Voyons maintenant comment le concepteur est amené à raisonner pour déterminer chacun de ces trois types de conditions de mise à jour.

V.1 LES MISES A JOUR PERIODIQUES

Dans le domaine de l'automatique (/BOUDAREL, 67/ , /DEPEYRE, 79/ , /GROOVER, 82/) les opérations s'exécutent souvent de manière cyclique (ou périodique). C'est notamment le cas pour les algorithmes de régulation (exemple les PID), les acquisitions de mesures et plus généralement chaque fois que la commande est régie par la théorie de systèmes échantillonnnés.

Le concepteur cherchant à définir les règles de mise à jour périodiques se posera successivement les questions suivantes :

1) la plus évidente c'est la périodicité de l'opération cyclique
   tous les Δt faire opération
   par exemple produire une pièce toutes les 10 mn

2) mais comme il devra tenir compte de facteurs tels que les pannes matérielles ou l'épuisement des matières premières, il cherchera à savoir dans quelles conditions l'opération est possible, c'est-à-dire étant un état E du système, est-ce que l'opération est compatible avec l'état E ? ou non ?
   ... si condition alors faire action

3) comme toute opération a un début, le concepteur devra préciser à partir de quel instant cette opération peut être activée pour la première fois, par exemple à partir de 8 h faire...
   ou sur ordre de l'opérateur faire...

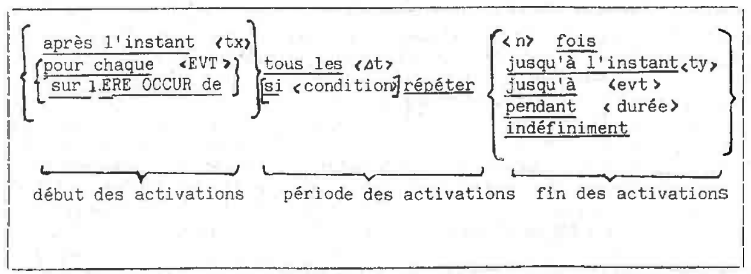
4) les systèmes de commande sont généralement conçus pour fonctionner 24h/24h mais à cause des pannes des éléments, de la maintenance des équipements, etc, la commande est arrêtée, de temps en temps, pour effectuer des opérations spéciales (maintenance, secours...).

Ceci a comme conséquence, pour le concepteur, de définir les conditions qui mettent fin à une action cyclique donnée, par exemple "produire une pièce toutes les 10 mn de 8h à 18h"; 18h indique la fin de l'opération.

Les instants indiquant la fin d'une opération cyclique peuvent correspondre à :
- un instant fixe,
- un évènement,
- un nombre de fois,
- une durée à partir de la 1ère activation de l'opération,
- l'infini.

Règle d'expression d'une mise à jour périodique

Pour répondre aux quatre points précédents le concepteur décrit une règle de mise à jour périodique suivant la syntaxe suivante :



V.2 MISES A JOUR A INSTANTS FIXES

Les opérations de mise à jour à instants fixes sont rares comparées aux opérations cycliques. Pour définir la règle d'activation d'une opération de ce type, le concepteur devra préciser :
- l'instant de déclenchement de l'opération,
- et éventuellement, comme dans le cas précédent, la condition de déclenchement.

La règle de mise à jour à instant fixe se décrit par :

A L'INSTANT ‹tx›[si<condition>]

Par exemple, "à 18h faire une sauvegarde, sur bandes, de tous les fichiers-statistiques".

V.3 LES MISES A JOUR ALEATOIRES

Une mise à jour est dite aléatoire si elle n'est pas périodique et si elle ne se fait à des instants fixes : elle se déclenche sur détection d'évènements.

Les opérations à activation aléatoire sont celles qui tiennent compte des changements d'état du système.

Pour définir leur règle d'activation le concepteur procède en plusieurs étapes :

1) chercher à définir la cause d'activation de l'opération (ie l'évènement dont l'appparition peut déclencher l'opération).

2) Une fois que cet évènement a été détecté :
- est-ce que l'opération est systématiquement activée ? ou bien faut-il que l'état du système vérifie certaines conditions ?
- est-ce que toutes les occurrences de l'évènement peuvent déclencher l'opération ? ou bien seule, la première, est significative ?

3) Parfois après détection d'un évènement, le système doit attendre un certain temps avant d'entreprendre une action.
Par exemple : si après avoir lancé une opération de perçage d'une pièce l'évènement de fin de perçage n'est pas détecté au bout d'un intervalle de temps Δ t le système lance une opération spéciale traitant cet incident .

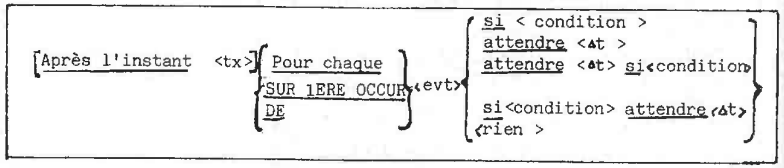
Ceci a pour conséquence de chercher à savoir si un évènement est pris directement en compte ou bien s'il faut temporiser avant de le prendre en compte.

4) L'apparition d'un changement d'état n'est pas toujours significative ; le concepteur devra savoir :
   - si l'évènement est suffisant à lui seul,
   - ou si cet évènement n'est pris en compte qu'à partir d'un instant fixé.

Par exemple, si on désire faire des statistiques sur les défauts survenant le soir, la règle d'activation de l'opération de mémorisation des défauts peut se décrire par "à partir de 18h pour chaque incident, activer l'opération de mémorisation des défauts".

Ainsi un évènement ne peut avoir un sens qu'à partir d'un instant choisi par le concepteur.

En résumé :
une règle d'expression d'une mise à jour aléatoire se décrit par :



V.4 REGLES POUR L'UTILISATION DES EVENEMENTS

Un évènement peut rentrer dans la construction de plusieurs règles de mise à jour différentes.

Lorsqu'une occurrence de cet évènement est détectée :
- règle 1 : toutes les règles de mise à jour utilisant cet évènement sont évaluées (l'occurrence est implicitement dupliquée pour toutes ces règles de mise à jour) ;
- règle 2 : l'évaluation d'une règle de mise à jour peut conduire, ou non, à l'activation d'une action, mais l'occurrence utilisée dans cette évaluation n'est plus accessible pour évaluer la règle de m.a.j concernée (il y a consommation implicite des occurrences de l'évènement).

VI CONCLUSION

Nous venons de définir un langage de description des conditions de mise à jour de la BDI. Basé sur les concepts de temps, d'évènements et de conditions, ce langage permet d'exprimer d'une manière assez naturelle les conditions de manipulation des données associées à une ACPI. A l'aide de ce langage on peut exprimer les règles d'activation des actions associées au mode de fonctionnement normal :
Par exemple : - tous les 50 ms capter la mesure de vitesse,
- produire une pièce toutes les 10 mn
mais aussi on peut exprimer les règles d'activation liées au mode de fonctionnement anormal, par exemple :
sur dépassement de température, avertir l'opérateur.

La prise en compte des problèmes de sûreté de fonctionnement se trouve ainsi facilitée puisque les mêmes mécanismes et les mêmes démarches permettent d'exprimer les deux modes de fonctionnement .

Nous avons, aussi, donné quelques règles pour aider le concepteur dans sa recherche des conditons de mise à jour de la BDI.

En précisant les règles de mise à jour des données nous venons de compléter le langage de description des ACPI déjà entamé dans le chapitre précédent.

CHAPITRE V

TRAITEMENT DE DEUX EXEMPLES : APPLICATION DES CONCEPTS PROPOSÉS

I INTRODUCTION

Nous avons developpé, dans les chapitres précédents une approche de conception des ACPI fondée sur l'utilisation des données.

Avant de passer aux problèmes d'implantation des applications, nous voudrions, dans ce chapitre, illustrer l'utilisation des concepts et démarches qui ont été proposés.

Pour ce faire nous avons choisi deux exemples :
- la commande d'un moteur à courant continu (développé dans /IUNG et Al, 83/) pour traiter un problème continu ;
- et la commande d'une pompe souterraine /KRAMER, 82./ pour traiter un problème séquentiel (séquentiel et continu sont pris ici au sens automatique des termes).

Pour chaque exemple nous chercherons à définir la structure de l'application de commande correspondante, et ce en procédant en deux étapes:
- détermination des données et des actions constituant l'application
- et ensuite détermination des conditions de mise à jour des données.

Une description (partielle car les corps des actions ne seront pas décrits) des applications sera donnée en utilisant le langage de description d'ACPI défini dans les chapitres III etIV.

Nous faisons remarquer que, dans les deux exemples traités, nous nous intéresserons au niveau commande directe seulement (les applications obtenues seront donc constituées d'un seul niveau).

II EXEMPLE DE LA COMMANDE D'UN MOTEUR A COURANT CONTINU

Cet exemple, développé par l'équipe "commande des machines électriques" du CRAN (à NANCY) consiste en la commande d'un moteur à courant continu par calculateur (on trouvera, dans l'annexe A2, les principales idées de la commande du moteur).

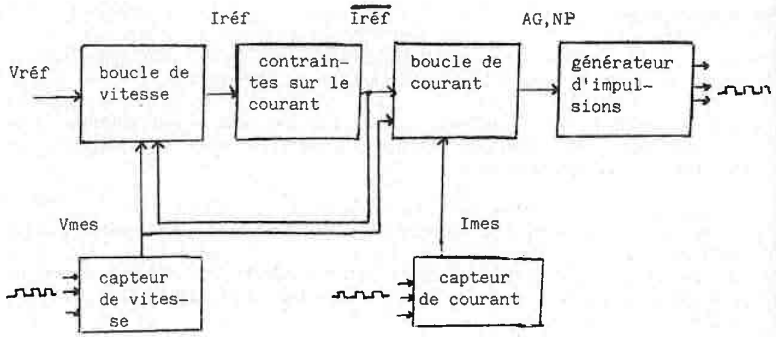
Enoncé du problème

Le problème posé est une régulation de vitesse d'un moteur à courant continu alimenté par un pont réversible à thyristors /IUNG et Al 83/.

La mesure de vitesse est réalisée de manière numérique.

La structure de régulation est la structure classique : boucle de courant boucle de vitesse de type proportionnel intégral. Les sécurités de fonctionnement en courant maximum et variation maximum du courant doivent être prises en compte au niveau de la référence de courant.

La figure ci-dessous illustre la structure de régulation choisie :



structure de régulation du moteur

La boucle de vitesse lit une vitesse de référence Vref, une mesure de vitesse Vmes et une référence de courant Ireˉf et grâce à un algorithme PI, élabore une référence de courant Iréf. Par comparaison avec deux constantes (Imax, courant maximum, et di/dt max, variation maximum du courant) on produit une référence de courant Ireˉf tenant compte des contraintes sur le courant. La boucle de courant calcule l'angle d'allumage (AG) et le numéro de (NP) à partir de Ireˉf, Vmes et Imes grâce à un algorithme PI.

Le générateur d'impulsions génère les impulsions nécessaires en fonction de AG et NP.

Les différentes boucles sont activées périodiquement :
- la boucle de vitesse est activée tous les Δtm,
- la boucle de courant est activée tous les Δtn,
- la boucle de contraintes sur courant est activé tous les Δtl
- et la boucle générateur d'impulsions est activée tous les Δ tj.

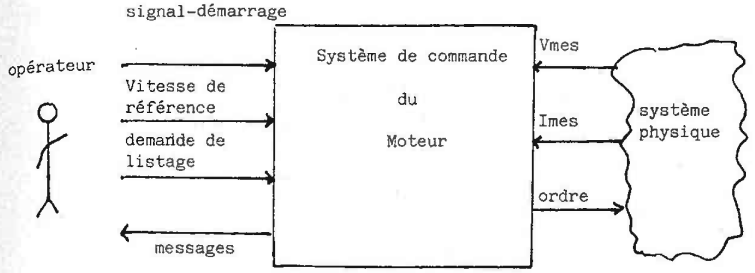
Δtm, Δtn, Δtl, et Δtj sont des périodes fixes.

Remarques
1. L'arrêt et le relancement de l'application ne sont pas traités : l'application est réinitialisée à chaque fois qu'elle est activée ;
2. nous avons défini le dialogue opérateur-calculateur comme suit :
   a- sur dépassement de vitesse, l'opérateur est averti par un message,
   b- le système mémorise, dans un fichier tous les dépassements de vitesse avec les instants de dépassement et met ce fichier à la disposition de l'opérateur qui peut le lister,
   c- à l'initialisation, l'opérateur fournit, au système, la vitesse de référence (Vref).

Les deux fonctions a et b ont été rajoutées, au cahier des charges initial /IUNG, 83/.

II.1. DETERMINATION DE LA STRUCTURE DE L'APPLICATION

Vis-à-vis de son environnement le système de commande du moteur se comporte comme une "boîte noire". Il reçoit, de l'opérateur, l'ordre de démarrage et la vitesse de référence (Vréf) et en cas de dépassement de vitesse il avertit l'opérateur à l'aide d'un message. Du côté moteur, le système de commande effectue des mesures sur la vitesse et le courant du moteur et applique les commandes élaborées au moteur.



L'analyse fonctionnelle de la commande du moteur est détaillée dans /IUNG, et Al 83/. Nous nous baserons sur cette analyse pour déterminer la structure de l'application de commande en termes de données et d'actions.

Même si l'exemple est élémentaire (comparé à l'automatisation des procédés de fabrication tels que la coulée continue) nous regrouperons les fonctions réalisées par le système de commande en trois classes :
- les fonctions relatives au dialogue avec le moteur,
- les fonctions relatives à la commande du moteur,
- et les fonctions relatives au dialogue avec l'opérateur.

Nous allons étudier brièvement chacune de ces trois classes de fonctions pour déterminer les ensembles de données et d'actions constituant l'application.

La structure finale de l'application est illustrée par la figure donnée à la page 88.

1) Dialogue avec le moteur

Les fonctions relatives au dialogue avec le moteur sont :
- la mesure de vitesse,
- la mesure de courant,
- et l'application de commandes au moteur.

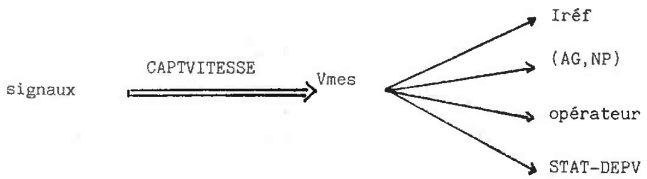
a) Mesure de vitesse

Le capteur de vitesse fournit, à la demande, une valeur Vmes qui représente la plus récente valeur de la vitesse du moteur.

Cette vitesse constitue une donnée que nous appelons Vmes. A cette donnée nous associons une action d'entrée CAPTVITESSE qui permet de produire cette donnée (ie acquérir la vitesse).

La donnée vitesse est utilisée par :
- le régulateur de vitesse pour déterminer la référence de courant Iréf,
- et le régulateur de courant pour déterminer l'angle d'allumage AG et le n° de pont à exciter NP.

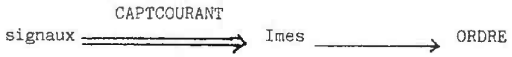
Elle est aussi :
- communiquée à l'opérateur en cas de dépassement de vitesse,
- et enregistrée dans le fichier des dépassements de vitesse (en cas de dépassement de vitesse).



b) Mesure de courant

Le capteur de courant délivre, à la demande, une valeur Imes qui représente la plus récente valeur du courant. A cette information (ie la valeur du courant) nous associons une donnée Imes qui est produite (ou captée) par une action d'entrée CAPTCOURANT .

La donnée Imes est utilisée par le régulateur de courant pour déterminer l'ordre à appliquer au moteur (ie AG et NP).



la mesure de courant est effectuée tous les Δ tl.

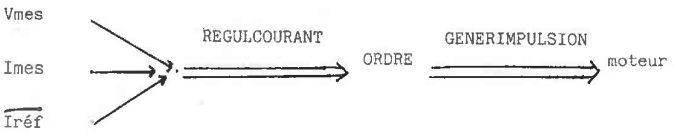
c) Application des commandes au moteur

Le système de commande applique, tous les Δtj, une commande (ou ordre) au moteur. Cette commande est constituée d'un angle d'allumage AG et d'un numéro de pont à exciter NP ; nous lui associons une donnée appelée ORDRE (ORDRE = (AG,NP)).

La commande est déterminée par le régulateur de courant et appliquée au moteur via une action de sortie GENERIMPULSION (qui génère les impulsions adéquates pour agir sur le moteur).

Pour déterminer la commande à appliquer, le régulateur de courant utilise la référence de courant $\overline{\text{Iréf}}$, la mesure de courant Imes et la mesure de vitesse Vmes.

La donnée ORDRE est produite par une action interne REGULCOURANT (ie le régulateur de courant).



2) La commande du moteur

Les fonctions relatives à la commande du moteur sont :
- la régulation de vitesse,
- les contraintes sur le courant,
- et la régulation du courant.

a) Régulation de vitesse

La boucle vitesse lit une référence de vitesse Vréf, une vitesse mesurée Vmes et une référence de courant $\overline{\text{Iréf}}$, tous les Δ tn, pour élaborer une référence de courant Iréf qui sera traitée par la fonction "contraintes sur le courant" pour déterminer $\overline{\text{Iréf}}$.

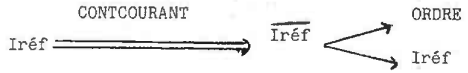
A la référence de courant Iréf nous associons une donnée Iréf qui est produite par une action interne REGULVITESSE.



b) Contraintes sur le courant

Les valeurs absolues du courant et de sa dérivée sont bornées. La fonction "contraintes sur le courant" reçoit une référence de courant Iréf, tous les Δ tl, et élabore une référence de courant $\overline{\text{Iréf}}$ (c'est une donnée) en tenant compte des contraintes sur le courant.

Cette donnée $\overline{\text{Iréf}}$ est utilisée par le régulateur de vitesse, et par le régulateur de courant. Pour produire $\overline{\text{Iréf}}$ nous définissons une action interne CONTCOURANT qui permet de produire $\overline{\text{Iréf}}$ à partir de Iréf.



c) Régulateur de courant

La régulation de courant se fait par un algorithme PI (ie une action interne REGULCOURANT) qui lit la référence de courant $\overline{\text{Iréf}}$, la mesure de vitesse et la mesure de courant pour déterminer la commande à appliquer au moteur (c'est-à-dire la donnée ORDRE vue précédemment).

3) Dialogue avec l'opérateur

Ce dialogue se résume aux trois points suivants :
- l'opérateur fournit, au moment du démarrage, la vitesse de référence Vréf
- à la demande de l'opérateur, le système liste le fichier des dépassements de vitesse qu'il construit au fur et à mesure de l'évolution de la commande du moteur,
- sur dépassement de vitesse, l'opérateur est averti par un message.

a) Phase de démarrage

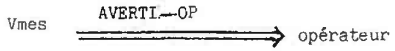
Lorsque l'opérateur désire activer la commande du moteur il fournit, au système, la vitesse de référence Vréf.

La donnée Vréf est produite par une action d'entrée associée au dialogue opérateur appelée DEMARRAGE et utilisée par le régulateur de vitesse pour déterminer Iréf.



b) Messages d'avertissement

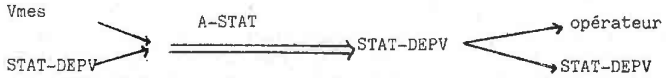
Pour tenir compte des dépassements de vitesse, le système avertit l'opérateur chaque fois que la vitesse est inférieure à Vmin ou supérieure à Vmax (Vmin et Vmax sont des valeurs fixées) en lui donnant la vitesse hors des bornes. Cette communication d'information est effectuée par une action de sortie AVERTI_OP.



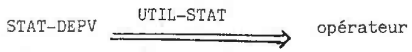
c) Construction et utilisation du fichier de dépassements de vitesse

Le système crée un fichier STAT-DEPV dans lequel il mémorise les instants de dépassements de vitesse ainsi que les vitesses mesurées hors des bornes.

Un nouvel enregistrement est rajouté au fichier STAT-DEPV chaque fois qu'il y a dépassement de vitesse (ie Vmes > Vmax ou Vmes < Vmin).
Le fichier STAT-DEPV est mis à jour par une action interne A-STAT qui prend en compte les dépassements de vitesse et les instants de dépassements.
Ce fichier est listé, au besoin, à l'opérateur.



Le fichier STAT-DEPV est listé (ou communiqué) à l'opérateur via une action de sortie UTIL-STAT.

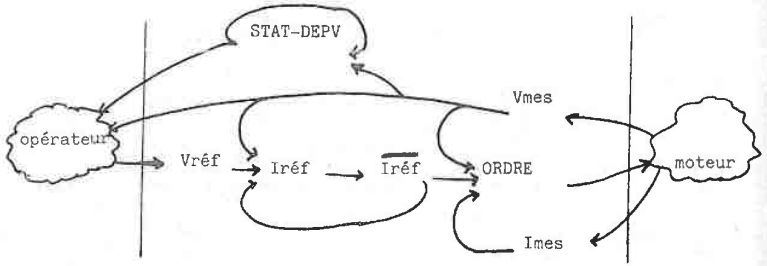


4) Remarque

En cas de dépassement de vitesse la commande du moteur s'arrête ; pour relancer la commande l'opérateur devra passer par la phase de démarrage (ie qu'il devra fournir au système une nouvelle vitesse de référence Vréf). L'application est réinitialisée apres chaque dépassement de vitesse.

5) Résumé

Nous résumons à l'aide des deux figures suivantes. Le flux de données associé à l'application de commande du moteur et les composantes (en termes de données/actions) de cette application.



Flux de données

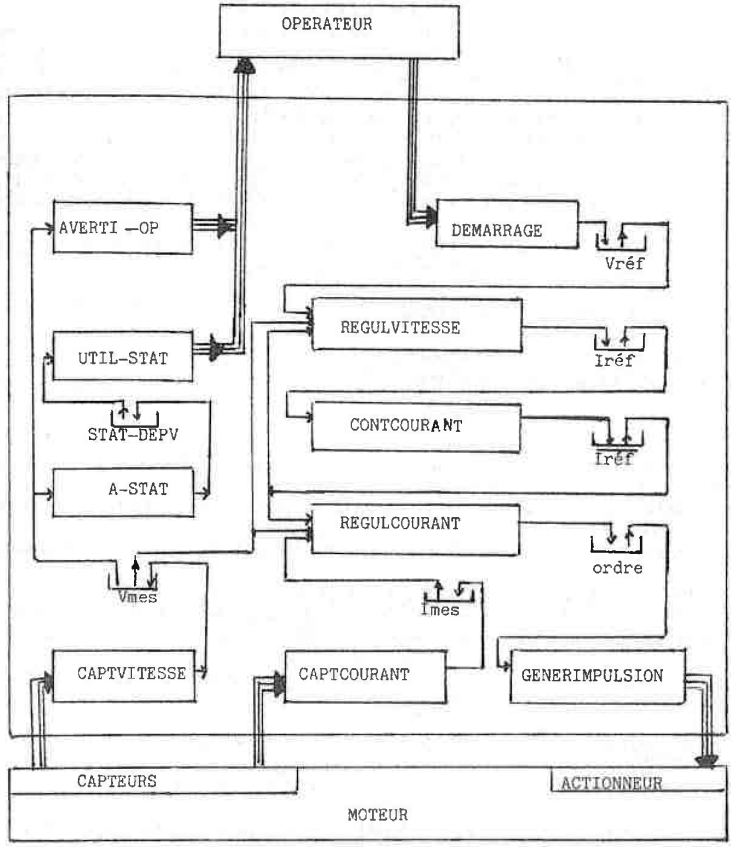


Schéma de l'application et de son environnement

II.2 RECHERCHE DES CONDITIONS DE MISE A JOUR DES DONNEES

Le cahier des charges du système de commande décrit le comportement du système de la manière suivante :

- toutes les opérations du système (sauf le dialogue avec l'opérateur) se font de manière cyclique ;
- la commande du système c'est-à-dire toutes les opérations (sauf le dialogue avec l'opérateur) s'arrête dès que la vitesse se trouve hors de l'intervalle de sécurité $]Vmin, Vmax[$;
- l'accès à une donnée se fait toujours en prenant en compte la dernière valeur de la donnée.

La conséquence des trois règles précédentes est que :
- la production des données Iréf, $\overline{\text{Iréf}}$, ordre, Imes et Vmes est périodique. La production de ces données commence dès que le signal de démarrage (il s'agit d'un évènement qu'on appelera EVTDEMARRAGE) a été donné et s'arrête dès que la vitesse se trouve hors de l'intervalle de sécurité (il s'agit d'un évènement qu'on appelera EVTDEPVITESSE) ;
- le fichier statistique STAT-DEPV est mis à jour sur détection de EVTDEPVITESSE et communiqué à l'opérateur s'il en a besoin (à la demande de "listage" du fichier STAT-DEPV est associé l'évènement EVTDEMFICH) ;
- la donnée Vmes est communiquée à l'opérateur si l'évènement EVTDEPVITESSE à été détecté ;
- et enfin toutes les données sont gérées en mode variable (car on s'intéresse toujours à la dernière valeur).

II, 3 DESCRIPTION DE L'APPLICATION

Nous décrivons, ici, l'application de commande du moteur à l'aide du langage défini dans les chapitres III et IV.

a) Définition des évènements
La mise à jour des données nécessite l'utilisation des trois évènements suivants :
- EVTDEMARRAGE : il détermine la demande de démarrage de l'application par l'opérateur,
EVTDEMARRAGE est associé à un niveau d'interruption N1 (générée par exemple par un bouton poussoir) ;
- EVTDEPVITESSE qui indique les dépassements de vitesse ;
- et EVTDEMFICH qui indique que l'opérateur demande le listage du fichier statistique (on associe cet évènement à un niveau d'interruption N2).

En utilisant le langage de description des évènements (cf chapitre IV) ces trois évènements se décrivent ainsi :

```
EVENEMENT        EVTDEMARRAGE : IT (N1)  FIN-EVT ;

EVENEMENT        EVTDEMFICH   : IT (N2)  FIN-EVT ;

EVENEMENT        EVTDEPVITESSE: FV  ((Vmes < Vmin)  ou  (Vmes
                                > Vmax))FIN-EVT ;
```

b) Description des données
Le mode d'utilisation "avec consultation du récent exemplaire" est pris par défaut dans les descriptions qui vont suivre :

```
DONNEE   Vréf ;

DE NIVEAU  1 ;

PHYSIQUE ;

STRUCTURE          VARIABLE DE  INTEGER ;

PRODUCTEURS =      DEMARRAGE : pour chaque EVT-DEMARRAGE ;

UTILISATEURS =     REGULVITESSE ;

FIN-DONNEE         Vréf ;
```



```
DONNEE   Vmes ;

DE NIVEAU 1 ;

PHYSIQUE ;

STRUCTURE          VARIABLE DE INTEGER  ;

PRODUCTEURS =      CAPVITESSE : pour chaque EVTDEMARRAGE   tous les
                     Δtm répéter jusqu'à EVTDEPVITESSE ;

UTILISATEURS =     REGULVITESSE ;
                   REGULCOURANT ;
                   A-STAT ;
                   AVERTI-OP : pour chaque EVTDEPVITESSE ;
FIN-DONNEE         Vmes ;
```

```
DONNEE   ORDRE ;

DE NIVEAU  1 ;

PRODUITE à PARTIR  de :        Vmes ; Imes ; Iréf ;

STRUCTURE          VARIABLE DE RECORD
                   AG :  INTEGER ;
                   NP :  INTEGER ;
                   END;

PRODUCTEURS =      REGULCOURANT ; pour chaque EVTDEMARRAGE tous les
                     Δtn répéter jusqu'à EVTDEPVITESSE ;

UTILISATEURS =     GENERIMPULSION pour chaque EVT DEMARRAGE tous les
                     Δtj répéter jusqu'à EVTDEPVITESSE ;

FIN-DONNEE         ORDRE ;
```

```
DONNEE   STAT-DEPV ;

DE NIVEAU 1 ;

PRODUITE à PARTIR  de :   Vmes ;STAT-DEPV ;

STRUCTURE          VARIABLE DE FILE OF  RECORD
                                vitesse : INTEGER ;
                                temps   : RECORD ;
                              HR, MN, SEC: INTEGER ;
                                        END ;
                                   END ;
PRODUCTEURS =      A-STAT : pour chaque EV1DEPVITESSE ;

UTILISATEURS =     UTIL-STAT : pour chaque EVTDEMFICH ;

FIN-DONNEE         STAT-DEPV ;
```

```
DONNEE   Iréf ;
DE NIVEAU 1 ;
PRODUITE à PARTIR de :        Vréf; Vmes ; Īréf;
STRUCTURE          VARIABLE DE INTEGER ;
PRODUCTEURS =      REGULVITESSE : pour chaque EVTDEMARRAGE tous les
                     Δtm répéter jusqu'à EVTDEPVITESSE;
UTILISATEURS =     CONTCOURANT ;
FIN-DONNEE         Iréf ;
```

```
DONNEE   Īréf ;
DE NIVEAU 1 ;
PRODUITE à PARTIR de :   Iréf ;
STRUCTURE          VARIABLE DE INTEGER ;
PRODUCTEURS =      CONTCOURANT : pour chaque EVTDEMARRAGE tous les
                     Δtl répéter jusqu'à EVTDEPVITESSE ;
UTILISATEURS =     REGULCOURANT ; REGULVITESSE ;
FIN-DONNEE         Īréf ;
```

```
DONNEE   Imes ;
DE NIVEAU 1 ;
PHYSIQUE  ;
STRUCTURE          VARIABLE DE INTEGER  ;
PRODUCTEURS =      CAPTCOURANT : pour chaque EVTDEMARRAGE tous les
                     Δtn répéter jusqu'à EVTDEPVITESSE ;
UTILISATEURS =     REGULCOURANT;
FIN-DONNEE  Imes ;
```


c) Description des actions
On ne s'intéresse pas, ici, à la description du comportement interne des actions

```
ACTION A-STAT ;
INTERFACE  ENTREES :  Vmes ; STAT-DEPV ;
           SORTIE  :  STAT-DEPV;
FIN-INTERFACE ;
CORPS....
FIN-ACTION   A-STAT  ;
```

```
ACTION REGULCOURANT ;
INTERFACE  ENTREES : Īréf ; Imes ; Vmes ;
           SORTIE  : ORDRE ;
FIN-INTERFACE ;
CORPS....
FIN-ACTION REGULCOURANT;
```

```
ACTION GENERIMPULSION;
INTERFACE  ENTREES : ORDRE ;
           SORTIE  : PHYSIQUE ;
FIN-INTERFACE ;
CORPS....
FIN-ACTION GENERIMPULSION;
```

```
ACTION DEMARRAGE ;

INTERFACE  ENTREES : PHYSIQUE ;
           SORTIE  : Vréf ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION DEMARRAGE;
```

```
ACTION AVERTI-OP;

INTERFACE  ENTREES : Vmes ;
           SORTIE  : PHYSIQUE ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION AVERTI-OP ;
```

```
ACTION UTIL-STAT ;

INTERFACE  ENTREES :  STAT-DEPV ;
           SORTIE  :  PHYSIQUE ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION UTIL-STAT ;
```

```
ACTION CAPTVITESSE ;

INTERFACE  ENTREES : PHYSIQUE ;
           SORTIE  : Vmes ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION CAPTVITESSE ;
```



```
ACTION  REGULVITESSE ;

INTERFACE  ENTREES :Vréf, Iréf ; Vmes
           SORTIE  : Iréf ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION REGULVITESSE  ;
```

```
ACTION CAPTCOURANT ;

INTERFACE  ENTREES : PHYSIQUE ;
           SORTIE  : Imes ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION CAPTCOURANT ;
```

```
ACTION CONTCOURANT ;

INTERFACE  ENTREES : Iréf ;
           SORTIE  : Iréf ;

 FIN-INTERFACE ;

 CORPS....

 FIN-ACTION  CONTCOURANT ;
```

III EXEMPLE DE LA COMMANDE D'UNE POMPE SOUTERRAINE

Enoncé du problème

L'exemple de la commande d'une pompe souterraine est développé dans /KRAMER,82/. Le fonctionnement simplifié de l'installation est illustré par la figure ci-dessous :

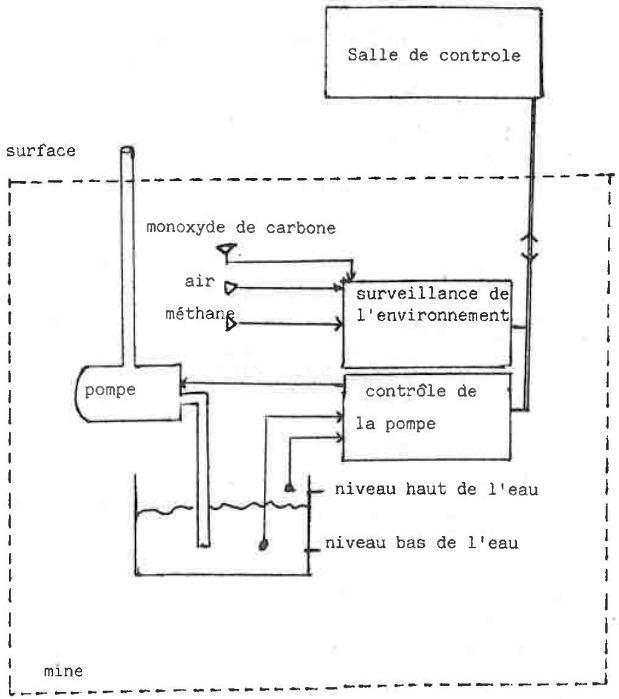


Schéma de la pompe de KRAMER

La pompe permet d'extraire de l'eau d'une mine de charbon pour la rejeter à la surface.

Le lancement de la pompe se fait par l'opérateur. Une fois lancée, la pompe fonctionne automatiquement, en tenant compte du niveau de l'eau (haut ou bas) et du taux de méthane à l'intérieur de la mine. La pompe doit s'arrêter si le niveau bas de l'eau est atteint ou bien si le taux du méthane dépasse le seuil de sécurité. L'opérateur peut intervenir pour arrêter la pompe même si les conditions précédentes ne sont pas satisfaites.

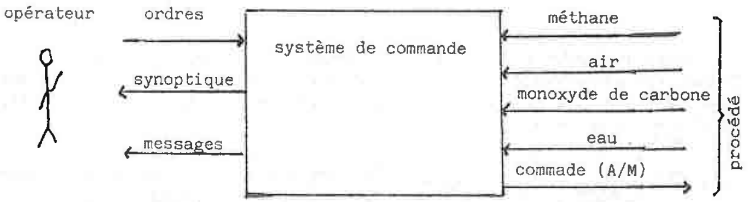
L'installation de commande doit assurer les 2 fonctions suivantes :
- la commande de la pompe qui consiste à :
  . détecter les niveaux de l'eau,
  . démarrer ou arrêter la pompe en fonction de l'ordre de l'opérateur, du niveau de l'eau et du taux de méthane,
  . et à fournir à l'opérateur l'état de la pompe ;
- et la surveillance de l'environnement qui consiste à mesurer et fournir, à la demande de l'opérateur, des informations sur le méthane, le monoxyde de carbone et l'air.

Remarque
L'énoncé initial /KRAMER, 82 / est étendu, ici, par une fonction synoptique. L'état de la pompe, du méthane, de l'air,...est visualisé, à la demande de l'opérateur, sous forme d'un synoptique rafraîchi toutes les secondes.

Nous nous intéressons aux données permettant de construire ce synoptique et non à la manière de construire ce synoptique (ie le dessin).

III.1 DETERMINATION DE LA STRUCTURE DE L'APPLICATION

Le système de commande :
- reçoit les ordres de l'opérateur
  . arrêt et mise en marche de la pompe,
  . demande d'informations sur l'état de l'air, du méthane, du monoxyde de carbone et de la pompe,
  . visualisation du synoptique.
- traite ces informations puis :
  . transmet, à l'opérateur, des messages en fonction de ce qu'il a demandé,
  . ou applique des commandes (arrêt ou marche) à la pompe en fonction de ce qui a été demandé.



Pour déterminer la structure de ce système de commande, nous décomposons les fonctions qu'il réalise en trois classes :

- le dialogue opérateur-calculateur,
- le dialogue procédé-calculateur,
- et la commande du procédé,

Nous faisons remarquer, pour aider à la compréhension de l'introduction des données et des actions, que le schéma illustrant l'application obtenue, après décomposition de chacune des trois classes, se trouve à la page 99 .

1) Le dialogue calculateur-opérateur

L'opérateur transmet des ordres au système de commande. Ces ordres permettent :
- soit d'agir sur la pompe,
- soit de fournir des informations à l'opérateur,
- soit de visualiser le synoptique.

a) Action sur la pompe

Les ordres d'arrêt/marche donnés par l'opérateur au système de commande définissent une donnée appelée A/M à laquelle nous associons une action d'entrée A-AR/AM.
Cette donnée est utilisée par le système de commande pour déterminer la commande à appliquer à la pompe.



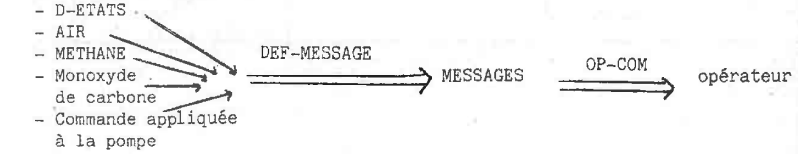
L'ordre produit par A-AR/AM est soit "A" (pour arrêt) soit "M" (pour marche).

b) Obtention d'informations

L'opérateur demande, en envoyant un ordre au système, l'état de l'air, du monoxyde de carbone, du méthane ou de la pompe.
A ce type d'ordre nous associons une donnée D-ETATS qui est produite par une action d'entrée A-ETAT.
Cette donnée est utilisée pour déterminer le message à transmettre à l'opérateur.



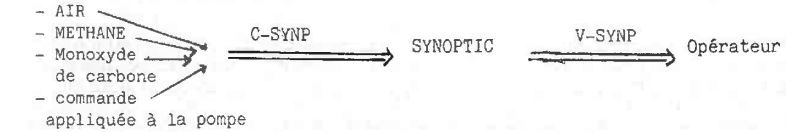
L'ensemble des messages transmis à l'opérateur constitue une donnée MESSAGES qui est produite par une action interne DEF-MESSAGE (en utilisant l'état de l'air,...) et utilisée par une action de sortie OP-COM pour être communiquée à l'opérateur.



A chaque ordre D-ETATS correspond un caractère qui indique le type du message demandé par l'opérateur.

c) Fonction synoptique

Le synoptique représente une donnée appelée SYNOPTIC définie à partir de l'état du méthane, de l'air, ... et de la pompe. Il est construit par une action interne C-SYNP et transmis à l'opérateur sur demande de celui-ci par une action de sortie V-SYNP.



2) Le dialogue calculateur-procédé

Le système de commande effectue des mesures sur l'état du procédé (ie le niveau de l'eau, le taux de méthane, le monoxyde de carbone et l'air).

Chacune de ces mesures constitue une donnée produite par une action d'entrée spécifique de la mesure et utilisée pour déterminer la commande à appliquer à la pompe, pour dialoguer avec l'opérateur et pour construire le synoptique.

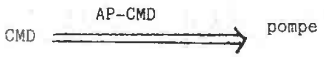
Ainsi nous avons :
- une donnée METHANE à laquelle est associée une action d'entrée AC-METHAN



- une donnée OXYCAR (pour le monoxyde de carbone) à laquelle est associée une action d'entrée AC-OXYCAR ;
- une donnée AIR à laquelle est associée une action d'entrée AC-AIR ;
- une donnée EAU à laquelle est associée une action d'entrée AC-EAU ;

les données OXYCAR et AIR sont utilisées pour dialoguer avec l'opérateur et pour construire le synoptique. La donnée EAU est utilisée seulement pour déterminer la commande à appliquer à la pompe.

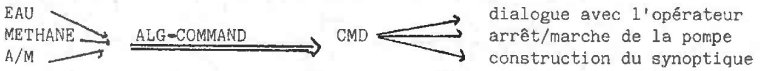
La commande déterminée par le système de commande (c'est la donnée CMD) est appliquée à la pompe via une action de sortie AP-CMD.



3) La commande de la pompe

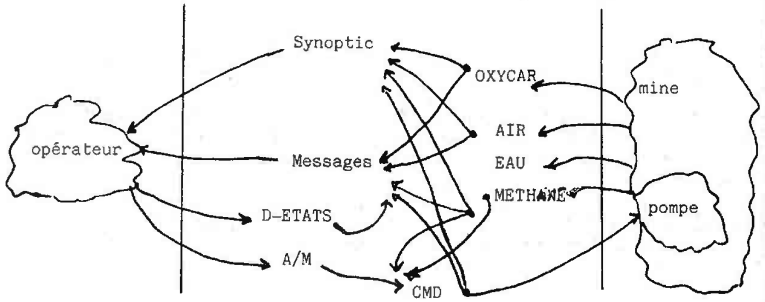
L'algorithme de commande permet de déterminer la commande à appliquer à la pompe en fonction du niveau de l'eau, du taux de méthane et de l'ordre de l'opérateur.

La commande à appliquer à la pompe constitue une donnée CMD produite par une action interne ALG-COMMAND et utilisée pour arrêter/faire marcher la pompe, pour constuire le synoptique et pour envoyer des messages à l'opérateur. La donnée CMD permet de connaître l'état de la pompe (en marche ou en arrêt).

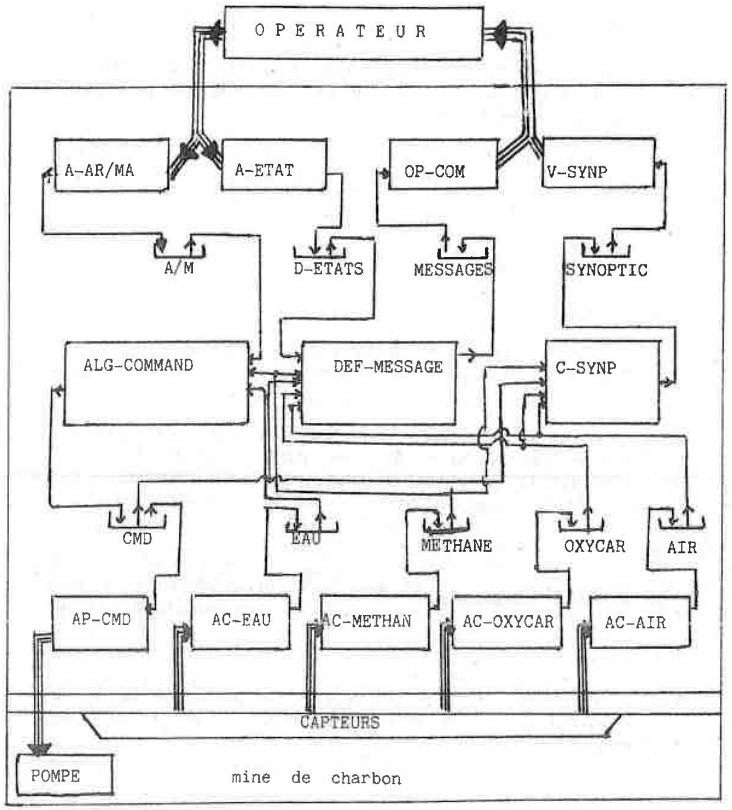


4) Résumé

Nous résumons, à l'aide des deux figures suivantes le flux de données associé à l'application de commande de la pompe et les composantes, (en termes de données et d'actions) de cette application.



Flux de données



Application de commande de la pompe et son environnement

III.2 DETERMINATION DES CONDITIONS DE MISE A JOUR DES DONNEES

a) La donnée A/M

La donnée A/M est produite chaque fois que l'opérateur demande l'arrêt ou la mise en marche de la pompe. Pour tenir compte de ce type de demande nous introduisons un évènement (soit EVTARMA) détecté sur présence de l'interruption associée au bouton d'arrêt/marche : A/M est produite pour chaque occurrence de EVTARMA.

b) La donnée D-ETATS

Comme A/M, la donnée D-ETATS est produite sur demande de l'opérateur (il s'agit ici des demandes d'obtention d'informations sur l'état de l'air,...).

Pour tenir compte de ce type de demandes nous introduisons un évènement (soit EVTOP) détecté sur présence de l'interruption affectée au dialogue opérateur.

c) La donnée MESSAGES

Chaque fois qu'une demande d'informations émise par l'opérateur, a été enregistrée, l'action DEF-MESSAGE est activée pour construire un message; il y a donc production de MESSAGES pour chaque fin de l'action A-ETAT (cette fin sera matérialisée par un évènement soit EVTFETAT).

La donnée MESSAGES est communiquée à l'opérateur via OP-COM chaque fois qu'un message a été construit, c'est-à-dire à chaque fin de l'action DEF-MESSAGE (cette fin sera matérialisée par un évènement soit EVTFMESS).

d) La donnée SYNOPTIC

La donnée SYNOPTIC est rafraîchie (ie mise à jour) toutes les secondes. SYNOPTIC est communiquée (c'est-à-dire visualisée) à l'opérateur à chaque fin de mise à jour de cette donnée.

Nous introduisons :
- un évènement EVTSYNP pour indiquer la fin de l'action C-SYNOP ;
- et un évènement EVTDSYNP pour indiquer que l'opérateur a demandé la visualisation du synoptique (on suppose ici que cette demande se fait via un bouton spécial auquel nous associons un n° d'interruption N5 activée en appuyant sur ce bouton).

e) La donnée CMD

La donnée CMD est produite :
. pour mettre en route la pompe :
  - si le niveau haut de l'eau est atteint, si le méthane n'a pas dépassé le seuil de sécurité et si l'ordre de marche automatique a été donné par l'opérateur ;
  - ou bien si l'opérateur vient de donner l'ordre de mise en route et que le niveau de l'eau est haut et que le niveau de méthane n'a pas dépassé le seuil de sécurité ;
. ou bien pour arrêter la pompe :
  - si l'ordre d'arrêt est demandé par l'opérateur,
  - si le niveau bas de l'eau a été détecté,
  - ou bien si le niveau de méthane a dépassé le seuil de sécurité.

A chacune des conditions précédentes est associé un évènement qui indique la réalisation de la condition.

Toutes les commandes élaborées sont traitées par l'action de sortie AP-CMD pour les appliquer à la pompe (il y a donc activation de AP-CMD sur chaque fin d'activation de ALG-COMMAND).

f) La donnée EAU

La donnée EAU prend deux valeurs 0 et 1 (0 pour niveau bas, 1 pour niveau haut). Elle est produite par l'action d'entrée AC-EAU chaque fois que le niveau bas ou le niveau haut de l'eau est atteint (niveaux haut et bas sont détectés par deux interruptions associées aux capteurs de niveau haut et de niveau bas : il y a introduction de deux évènements EVTHAUT et EVTBAS associés à ces deux interruptions).

g) Les données METHANE, AIR et OXYCAR

Les données METHANE, AIR et OXYCAR sont produites respectivement par les actions d'entrée AC-METHAN, AC-AIR et AC-OXYCAR à des périodes respectives $\Delta t1$, $\Delta t2$ et $\Delta t3$.

III.3 DESCRIPTION DE L'APPLICATION

1) Définition des évènements

Nous décrivons ici les évènements nécessaires à la définition des règles de mise à jour des données. Ces évènements sont :

- EVTFSYNP : il indique la fin de la mise à jour du synoptique,
  EVENEMENT EVTFSYNP : FIN (C-SYNP) FIN-EVT ;

- EVTDSYNP : il indique la demande de la visualisation du synoptique
  EVENEMENT EVTDSYNP : IT (N5) FIN-EVT ;

- EVTDEMSYNP composé à partir de EVTFSYNP et EVTSYNP, il indique que le synoptique doit être rafraîchi sur l'écran ;
  EVENEMENT COMPOSE EVTDEMSYNP : pour chaque EVTFSYNP APRES
  pour chaque EVTDSYNP FIN-EVT ;

- EVTFMESS : il indique la fin de construction d'un message :
  EVENEMENT EVTFMESS : FIN (DEF-MESSAGE) FIN-EVT ;

- EVTMETHANE : il indique un dépassement du seuil du taux de méthane :
  EVENEMENT EVTMETHANE : FV (METHANE > SEUIL METHANE) FIN-EVT ;

- EVTARRET : il indique une demande d'arrêt de la pompe ;
  EVENEMENT EVTARRET : FV (A/M = "A") FIN-EVT ;

- EVTMARCHE : il indique une demande de mise en marche de la pompe :
  EVENEMENT EVTMARCHE : FV (A/M = "M") FIN-EVT ;

- EVTBAS : il indique le dépassement du seuil bas de l'eau (la détection se fait sur interruption causée par le capteur, soit N1 cette interruption,
  EVENEMENT EVTBAS : IT (N1) FIN-EVT ;

- EVTHAUT : il indique un dépassement du seuil haut de l'eau (la détection se fait sur génération de l'interruption (N2) associée au capteur de niveau haut) ;
  EVENEMENT EVTHAUT : IT (N2) FIN-EVT ;

- EVTEAU : il indique soit EVTBAS soit EVTHAUT
  EVENEMENT COMPOSE EVTEAU : pour chaque EVTBAS ou
  pour chaque EVTHAUT FIN-EVT ;

- EVTCOM : il indique que les conditions de production d'une nouvelle commande à appliquer à la pompe sont réalisées ;
  Il y a production d'une commande :
  . pour chaque demande de l'opérateur (arrêt ou marche)
  . pour chaque détection d'un dépassement (bas ou haut) du niveau de l'eau et pour chaque dépassement du taux de méthane (à condition qu'une démande de marche ait été déjà enregistrée).
  EVENEMENT COMPOSE EVTCOM : Pour chaque EVTARRET ou
  Pour chaque EVTMARCHE ou
  Pour chaque EVTEAU ou
  Pour chaque EVTMETHANE FIN-EVT ;

- EVTFINCOM : évènement qui indique la fin de production d'une commande ;
  EVENEMENT EVTFINCOM : FIN (ALG-COMMAND) FIN-EVT ;

- EVTOP : évènement qui indique le début d'une demande de dialogue pour l'obtention d'informations (il est associé au niveau d'interruption N3) ;
  EVENEMENT EVTOP : IT (N3) FIN-EVT ;

- EVTARMA : évènement qui indique qu'une demande d'arrêt ou de marche a été émise par l'opérateur (il est associé à un niveau d'interruption N4);
  EVENEMENT EVTARMA : IT (N4) FIN-EVT ;

- EVTETAT : évènement qui indique qu'une demande d'informations a été enregistrée par le système ;
  EVENEMENT EVTETAT : FIN (A-ETAT) FIN-EVT.

2) Description des données

Le mode d'utilisation "avec consultation du récent exemplaire" est pris par défaut dans les descriptions qui vont suivre.

```
DONNEE   A/M ;
DE NIVEAU 1 ;
PHYSIQUE  ;
STRUCTURE            VARIABLE DE CHAR ;
PRODUCTEURS =        A-AR/AM : pour chaque EVTARMA ;
UTILISATEURS =       ALG-COMMAND ;
FIN-DONNEE A/M ;
```

```
DONNEE   D-ETATS ;
DE NIVEAU 1 ;
PHYSIQUE  ;
STRUCTURE          SUITE DE CHAR ;
PRODUCTEURS =      A-ETAT  : pour chaque EVTOP ;
UTILISATEURS = DEF-MESSAGE : avec consommation
                             de ancien exemplaire ;
FIN-DONNEE  D-ETAT :
```

```
DONNEE   MESSAGES :
DE NIVEAU 1 ;
PRODUITE à PARTIR de : D-ETATS; AIR ; METHANE ; OXYCAR ; CMD ;
STRUCTURE          SUITE DE array   1..80 of char ;
PRODUCTEURS =      DEF-MESSAGE : pour chaque EVTETAT ;
UTILISATEURS =     OP-COM : pour chaque EVTFMESS
                   avec consommation de ancien exemplaire ;
FIN-DONNEE  MESSAGES ;
```

```
DONNEE   CMD ;
DE NIVEAU 1 ;
PRODUITE à PARTIR de : A/M ; METHANE ; EAU ;
STRUCTURE          VARIABLE DE BOOLEAN  ;
PRODUCTEURS =      ALG-COMMAND : pour chaque EVTCOM ;
UTILISATEURS =     AP-COM : pour chaque EVTFINCOM ;
                   DEF-MESSAGE ;
                   C-SYNP ;
FIN-DONNEE  CMD ;
```



```
DONNEE   SYNOPTIC ;
DE NIVEAU 1 ;
PRODUITE à PARTIR de : AIR ; OXYCAR ; METHANE ; CMD;
STRUCTURE          VARIABLE DE RECORD ;
                   S-AIR : REAL ;
                   S-OXYCAR : REAL ;
                   S-METHANE : REAL ;
                   S-ETAT-POMPE : BOOLEAN ;
PRODUCTEURS =      C-SYNP : tous les ls  ;
UTILISATEURS =     V-SYNP : pour chaque EVTDEMSYNP ;
FIN-DONNEE  SYNOPTIC ;
```

```
DONNEE   EAU  ;
DE NIVEAU 1 ;
PHYSIQUE  ;
STRUCTURE          VARIABLE DE BOOLEAN  ;
PRODUCTEURS =      AC-EAU : pour chaque EVTEAU ;
UTILISATEURS =     ALG-COMMAND ;
FIN-DONNEE  EAU  ;
```

```
DONNEE   METHANE ;
DE NIVEAU 1 ;
PHYSIQUE  ;
STRUCTURE          VARIABLE DE REAL    ;
PRODUCTEURS =      AC-METHAN  : tous les  Δtl ;
UTILISATEURS =     ALG-COMMANDE ; DEF-MESSAGE ; C-SYNC ;
FIN-DONNEE  METHANE ;
```

les données AIR et OXYCAR sont similaires à la donnée METHANE.

3) Description des actions

```
ACTION OP-COM ;

INTERFACE  ENTREES : MESSAGES ;
           SORTIE  : PHYSIQUE ;

 FIN-INTERFACE ;

 CORPS % description de l'impression des messages %

 FIN-ACTION  OP-COM ;
```

```
ACTION DEF-MESSAGE ;

INTERFACE  ENTREES : D-ETATS; AIR ; CMD ;
                     METHANE ; OXYCAR ;
           SORTIE  : MESSAGES ;
 FIN-INTERFACE ;

 CORPS % construction de messages en fonction de la
         demande de l'opérateur %

 FIN-ACTION DEF-MESSAGE ;
```

```
ACTION ALG-COMMAND ;

INTERFACE  ENTREES : METHANE ; EAU ; A/M ;
           SORTIE  : CMD ;

 FIN-INTERFACE ;

 CORPS % détermination de la commande à appliquer à la
         pompe %

 FIN-ACTION ALG-COMMAND;
```



```
ACTION AP-CMD ;

INTERFACE  ENTREES : CMD ;
           SORTIE  : PHYSIQUE ;

 FIN-INTERFACE ;

 CORPS  % application d'une commande à la pompe %

 FIN-ACTION AP-CMD ;
```

```
ACTION AC-EAU ;

INTERFACE  ENTREES : PHYSIQUE ;
           SORTIE  : EAU ;

 FIN-INTERFACE ;

 CORPS % détermination du nivau bas (ou haut) de l'eau %

 FIN-ACTION    AC-EAU ;
```

```
ACTION A-ETAT ;

INTERFACE  ENTREES : PHYSIQUE ;
           SORTIE  : D-ETATS ;

 FIN-INTERFACE ;

 CORPS  % description de l'acquisition des demandes-
          opérateur %

 FIN-ACTION A-ETAT ;
```

```
ACTION C-SYNP ;

INTERFACE  ENTREES : AIR ; OXYCAR ; METHANE ; CMD ;
           SORTIE  : SYNOPTIC ;

 FIN-INTERFACE ;

 CORPS % description de la construction du synoptique %

 FIN-ACTION C-SYNP ;
```

```
ACTION V-SYNP ;

INTERFACE  ENTREES : SYNOPTIC ;
           SORTIE  : PHYSIQUE ;

FIN-INTERFACE ;

CORPS % description de la visualisation du synoptique %

FIN-ACTION V-SYNP ;
```

Les actions AC-METHAN , AC-AIR et AC-OXYCAR se décrivent de manières similaires à celle de AC-EAU.

L'action A-AR/AM est décrite de manière similaire à l'action A-ETAT.

III.4 REMARQUE

Dans un cas continu, les données prennent (généralement) leurs valeurs dans des intervalles alors qu'elles ne prennent (souvent) que des valeurs booléennes dans des cas séquentiels.

Les solutions obtenues (en termes de données-actions-évènements) pour les cas séquentiels présentent les caractéristiques suivantes :
- la synchronisation de l'application, se faisant sur changement d'état des données (passage de 0 à 1 ou de 1 à 0), implique l'introduction d'un nombre considérable d'évènements (pour tenir compte des changements);
- les actions sont peu importantes au sens cohésion /DERNIAME, 74/ , /SAKHONN, 80/ et leur nombre risque d'être important puisqu'elles ne produisent que des valeurs booléennes.

Ces deux caractéristiques ne sont pas, à notre avis, des incovénients pour les solutions obtenues car elles sont intrinsèques aux problèmes séquentiels.

IV CONCLUSION

Nous avons traité deux exemples de procédés simples (un moteur à courant continu et une pompe souterraine). Ces deux exemples nous ont permis d'illustrer nos propositions pour la structuration des ACPI et de montrer que les concepts introduits sont suffisants pour décrire des ACPI. Les solutions obtenues, pour ces deux applications, sont décrites à l'aide d'un langage de programmation et peuvent être implantées directement (et ce, évidemment, en utilisant le système opératoire adéquat (cf IIème partie)). Les solutions obtenues sont, aussi, modulaires.

Le fait d'avoir choisi deux types d'exemples (un problème continu et un problème séquentiel) nous a permis de montrer que nos propositions sont générales et ne se limitent pas à des problèmes particuliers.

PARTIE II

IMPLANTATION DES APPLICATIONS

De plus, nous avons remarqué que les cas séquentiels conduisent, de part leur nature, à l'introduction d'un nombre important d'évènements.

Cependant, les concepts et démarches proposés ne peuvent être complètement validés que s'ils sont utilisés pour concevoir des applications à grande échelle telles que la coulée continue /ANNEXE A1/.

En effet, la conception des applications à grande échelle permet de mieux montrer l'intérêt des bases de données industrielles. Car, comme pour les applications classiques, les bases de données ne sont intéressantes (au sens utilité) que si les quantités d'informations manipulées sont importantes.

De plus, la conception de ce type d'applications nécessite l'emploi d'un guide méthodologique, vues la complexité des installations et l'importance des investissements qu'elles impliquent.

La description d'une application du type commande de coulée continue aurait aidé à mieux comprendre l'importance des concepts proposés et de la démarche de conception telle que nous l'avons "caricaturée".

La commande de la coulée continue (présentée en annexe A1) montre :
- l'importance (en nombre et en types) des données manipulées par une application industrielle ;
- et l'utilisation de la commande hiérachisée pour la conception d'un système de commande.

Ces deux aspects, fondamentaux pour notre approche, n'ont pu être illustrés par les deux exemples que nous avons traités vue leur simplicité.

CHAPITRE VI

REPARTITION DES APPLICATIONS

L'approche de conception qui vient d'être proposée consiste à définir une ACPI à l'aide de trois ensembles :
- un ensemble de données,
- un ensemble d'actions,
- et un ensemble de règles de mise à jour des données.

Nous nous intéresserons dans cette partie à l'implantation de ces trois ensembles d'objets. Cette implantation consiste à affecter les objets aux diverses machines composant l'installation de commande et soulève les questions suivantes :

1 Comment affecter les objets de manière à tenir compte des contraintes liées à la sûreté de fonctionnement et aux temps de réponse ?

2 De quoi a-t-on besoin comme système opératoire, pour gérer les objets ?

3 Et comment implanter les objets en fonction de ce système opératoire ?

La réponse à la première question sera développée dans le chapitre VI.

Les questions 2 et 3 seront traitées dans le chapitre VII.

I INTRODUCTION

Une des caractéristiques des procédés industriels, soulignée dans le chapitre I, est leur répartition géographique. Cette répartition géographique implique souvent la répartition des systèmes de commande et on parle alors de systèmes composés de plusieurs stations de commande /KRAMER,81/.

Les stations de commande coopèrent à la réalisation d'une tâche commune qui est la commande du procédé ; chacune d'elles étant responsable de la commande d'une partie du procédé et/ou du dialogue homme-machine.

La coopération entre les stations suppose l'existence d'un réseau de communication qui leur permet de s'échanger des informations.

Répartir un système de commande (ou une ACPI) signifie, pour nous, affecter les données, les actions et les évènements constituant le système, aux différents sites du réseau (ou stations de commande).

Nous cherchons à travers ce chapitre, à expliciter le pourquoi de la répartition, les problèmes qu'elle soulève et la manière de structurer l'installation de commande pour aider à l'obtention d'une répartition efficace de l'application (c'est-à-dire qui tient compte des contraintes liées à la sûreté de fonctionnement et aux temps de réponse).

II REPARTITION DES OBJETS : NECESSITÉ ET PROBLEMES POSÉS

II 1 NECESSITÉ DE LA REPARTITION

II 1.1 Répartition de la BDI

Comme pour les bases de données classiques, deux solutions peuvent être envisagées pour la gestion de la BDI :
- BDI centralisée,
- BDI répartie.

1) BDI centralisée

Gérer une BDI en mode centralisé revient à stocker l'ensemble des informations de la BDI sur une seule station du système de commande. Toutes les autres stations devront s'adresser à cette station pour mettre à jour les données de la BDI.

Cette solution a l'avantage de simplifier le problème de gestion de la BDI puisqu'une seule station en est responsable ; mais elle peut compromettre les deux points suivants, vitaux pour la commande de procédés :

- Le temps de réponse,
  La communication fréquente, entre les stations de commande, introduit des délais dûs d'une part, à la propagation du signal sur les médium, et d'autre part, aux retransmissions causées par les erreurs de transmission et à l'exécution des protocoles de communication.
  Ces délais, s'ils sont importants, ont comme conséquence le non-respect des contraintes de temps (mise à jour effectuée en retard).

- La sécurité des données
  Une panne de la station conservant la BDI, immobilise toutes les autres stations : elles n'ont plus d'informations pour assurer leur tâche ; mais aussi une ligne de transmission coupée empêche la station reliée au réseau, par cette ligne, de continuer son travail même si cette station assure un travail relativement indépendant de ceux des autres stations.

Si les problèmes liés au respect des contraintes de temps et de la sécurité des données sont résolus par la technologie des équipements utilisés (c'est le cas essentiellement des petites installations où le nombre de stations et des entrées/sorties est restreint et des installations où les contraintes de temps ne sont pas sévères), la centralisation de la BDI peut être retenue.

Malheureusement ces deux problèmes ne trouvent pas toujours leur solution dans la technologie utilisée (soit pour des raisons techniques, performances limitées, soit pour des raisons économiques) dans le cas de la commande de procédés complexes (les raffineries par exemple) et la répartition des informations s'avère nécessaire.

2) BDI ditribuée

L'ensemble des données de la BDI est réparti sur les différentes stations de commande.

L'unité de répartition étant la donnée : une donnée ne peut être répartie sur plusieurs stations.

Chaque station qui permet de conserver une partie de la BDI est dite point de stockage des informations :

S_m : conserve les données D_i, ..., D_{nm}

les données D_j sont dites "implantées sur S_m"

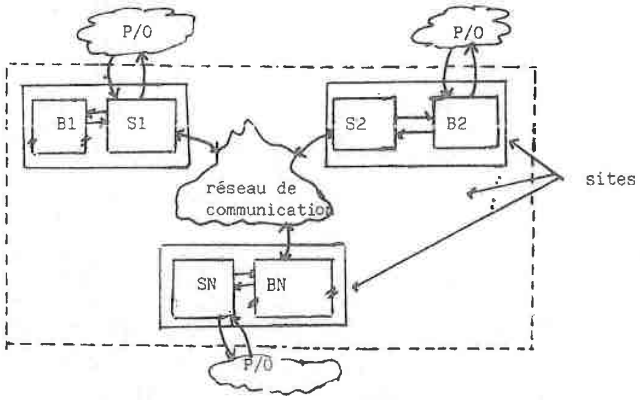
D_j implantée sur S_m $\quad j \quad [\, i,n_m \,]$

La BDI distribuée sur plusieurs sites (ou stations) est dite "base de données industrielles répartie" (ou BDIR).

Chacune des sous bases Bj obtenues en répartissant la BDIR est dite "base de données industrielles locale" (ou BDIL).

BDIR = (B1,...Bm) où m est le nombre de BDIL

Une BDIL conservée sur un site est accessible aux autres sites du réseau.



REMARQUE

Nous avons évoqué deux types de gestion d'une BDI (centralisée ou répartie). Pour les BDC, il existe un autre type de gestion : c'est la duplication de toute ou d'une partie d'une BDC sur les différents sites (on aboutit à la notion de copies dupliquées /THOMAS, 76/ avec tous les mécanismes que cela entraine pour assurer la cohérence de toutes les copies).

Un tel type de gestion est parfois nécessaire pour les applications où la redondance du matériel et du logiciel est exigée par la nature des procédés commandés (par exemple les systèmes embarqués où les installations de commande sont doublées, triplées,...).

Nous ne nous intéressons pas à ce type de gestion des BDI dans le présent travail.

II 1.2 Répartition des actions

Chaque station (ou site) assure une fonction particulière faisant partie de l'ensemble des services que doit fournir l'ACPI. Cette fonction est définie à l'aide d'actions (par exemple l'ensemble des actions permettant d'assurer la commande d'un moteur pas à pas). Il est alors nécessaire au moment de l'implantation de préciser le lieu d'affectation de chacune des actions en fonction du rôle qu'on veut faire jouer à chacune des stations. Chaque station S_m réalise un ensemble d'actions A_i, ... A_{lm}.

Les actions A_k $\quad K \in [\, i,l_m \,]$ sont dites implantées sur S_m

II 1.3 Répartition des évènements

Le problème de répartition des évènements est le même que celui des données.
En effet, on peut envisager deux types de gestion des évènements :
- gestion centralisée,
- gestion répartie.

1) Gestion centralisée

Toute occurrenee d'évènement détectée est transmise à un site de gestion d'évènements. Comme les mises à jour de la BDI se font, essentiellement, sur détection d'évènements, ce site est chargé d'activer l'ensemble des actions de l'ACPI : il y a contrôle centralisé.
Le contrôle centralisé simplifie les problèmes de gestion des évènements.

Cependant les délais résultant des diverses communications entre la station de contrôle (celle qui gère les évènements) et les autres stations peuvent devenir importants et remettre en cause le respect des contraintes de temps: par exemple un évènement associé à un traitement urgent (tel que la coupure de courant) risque de ne pas être traité, à temps, à cause de ces communications.

De plus, la panne de la station de contrôle a comme conséquence le blocage de toutes les autres stations.

Comme pour la BDI, si les deux problèmes évoqués précédemment sont résolus par le matériel, une solution centalisée peut être retenue sinon la répartition du contrôle est indispensable.

2) Gestion répartie

Chaque site est chargé de la gestion d'un sous-ensemble d'évènements.
Nous dirons qu'un évènement EVi est implanté sur le site Si;
EVi implanté sur Si
c'est-à-dire que la détection des occurrences de EVi est à la charge de Si;
Ces occurrences peuvent être utilisées par le site Si et par les autres sites :
- soit pour activer des actions locales ou distantes,
- soit pour évaluer les évènements composés locaux ou distants.

La communication des occurrences d'évènements se fait par des échanges de messages entre les différentes stations.

II 1.4 Avantages et inconvénients de la répartition

a) Les avantages visés par l'utilisation des systèmes répartis sont :
- la souplesse,
- la diminution du coût global,
- la puissance de traitement,
- et la sûreté de fonctionnement.

La littérature décrivant les avantages de la répartition est abondante /CORNAFION, 81/ , /MACCHI, 79/ , /LORRAINS, 79/ , /LELANN,77, 79, 81/.

Pour les BDI nous noterons essentiellement les avantages suivants :
- la puissance de traitement qui se traduit par le respect des contraintes de temps (mise à jour, à temps, de la BDI).
- la possibilité de fonctionner en performances dégradées : la panne d'un site n'affecte pas, ou affecte peu, les autres sites (disponibilité de la BDI) ;
- et la modularité de la BDI qui rend aisée sa maintenance.

b) l'inconvénient majeur des systèmes répartis réside dans les deux points suivants :
- difficulté de synchroniser l'ensemble des sites,
- et accroissement des temps de réponse (dû d'une part aux communications inter-stations et aux mauvais choix de répartition d'autre part) qui peut être fatal pour la commande du procédé.

II.2 PROBLEMES LIES A LA REPARTITION

Ces problèmes ne sont pas nouveaux, on les rencontre aussi, dans le cas des BDC /GARDARIN, 76 , 78/ , /PARENT, 77/ mais se présentent avec plus d'acuité pour les ACPI, il s'agit :
- de la difficulté de synchronisation qui se traduit ici, par une difficulté d'ordonnancement des exemplaires des données et des occurrences d'évènements ;
- et du manque d'une vue unique de la BDI.

II 2.1 Problème d'ordonnancement

1) A quel niveau se pose t-il ?

a) Ordonnancement des exemplaires d'une donnée
Au moment de la mise à jour de la BDI nous avons considéré qu'une donnée Di était représentée par une suite d'exemplaires :

$Vi_1, Vi_2, \ldots Vi_n$ ordonnés dans le temps,

Cette hypothèse n'est vraie que dans le cas où le système de commande est centralisé : toutes les productions sont faites par le même site, il peut les ordonner à l'aide de son horloge interne.

Cependant, dans le cas d'un système réparti, le manque d'une horloge, permettant la synchronisation des différents sites, rend le problème d'ordonnancement des exemplaires d'une donnée, difficile à résoudre (car une donnée peut être produite par plusieurs producteurs éventuellement répartis sur des sites différents).

b) Composition d'évènements

Les opérateurs de composition d'évènements SIM, AVANT et APRES introduits dans le chapitre IV font intervenir les instants de détection des évènements lors de l'évaluation des évènements composés à l'aide de ces opérateurs. Le problème réside, ici, au niveau de comparaison des instants d'apparition des évènements implantés sur des sites différents.

2) Solutions au problème d'ordonnancement

Le problème d'ordonnancement dans les systèmes répartis a été étudié principalement par LAMPORT qui propose une méthode de synchronisation par datation des messages échangés entre les différents sites du réseau.

Dans notre cas, l'application de l'algorithme des horloges logiques de LAMPORT /LAMPORT, 78/ se traduit par l'association d'un instant "logique" et d'un numéro de site à chaque exemplaire produit et à chaque occurrence d'évènement qui rentre dans une règle de composition d'évènement contenant, au moins, l'un des opérateurs AVANT, APRES et SIM.

Cependant si on prend en compte l'aspect temps réel de la commande de processus, l'utilisation de l'algorithme de LAMPORT présente l'incovénient suivant : /DESCHIZEAUX, 81/ :
les occurrences d'évènements et les productions d'exemplaires sont datées grâce aux envois et aux réceptions de messages.

Or les transferts de messages peuvent introduire des délais supérieurs aux constraintes de temps du procédé commandé et des erreurs graves peuvent apparaître.

L'inconvénient de ce genre de synchronisation est que la précision obtenue est fonction des temps de transfert (et non de la vitesse d'évolution du procédé) :
- le volume d'échanges nécessaires augmente les risques d'erreurs ;
- l'ordonnancement est arbitraire à cause de l'affection des numéros aux sites (toutefois ce problème peut être résolu en tenant compte des priorités des sites) ;
- dans le cas d'ordonnancement des exemplaires d'une donnée ; il peut y avoir attente des utilisateurs même si certains exemplaires ont été déjà reçus par le site de stockage. Cette attente est due au fait que la décision d'ordonnancement selon la méthode de LAMPORT ne peut être prise que si tous les sites ont envoyé leurs messages (ie, ici, que tous les producteurs ont transmis un exemplaire chacun).

Si les inconvénients cités précédemment sont acceptables par le type de l'application traitée, l'algorithme de LAMPORT peut être retenu.

Dans le cas contraire, nous proposons les solutions suivantes (pour lesquelles nous nous limiterons à énoncer le principe ; elles peuvent faire l'objet d'une étude approfondie dans l'avenir).

a) Ordonnancement des exemplaires d'une donnée

Les différents exemplaires produits par les diverses sources de production d'une donnée sont transmis au site de stockage de ces exemplaires (une donnée est stockée sur un et un seul site). Dès qu'un exemplaire est produit il est directement transmis au site de stockage (par "directement" on sous entend que la source de production doit transmettre le plus vite possible l'exemplaire qu'elle a produit car le temps pendant lequel elle le garde influe sur l'ordonnancement des exemplaires).

L'ordonnancement des exemplaires peut se faire de deux manières possibles :

1 affecter une priorité à chaque source de production et au moment où un exemplaire est reçu par le site de stockage cet exemplaire est stocké (ie rangé dans la file associée à la donnée à laquelle appartient cet exemplaire) en tenant compte de la priorité de la source de production : si le site de stockage n'a qu'un seul exemplaire à ranger, il le met directement dans la file sinon (ie que plusieurs exemplaires de la même donnée viennent d'être reçus) il rangera les exemplaires reçus en tenant compte des priorités associées aux sources.

Cette solution est à retenir si :
* on peut établir des priorités entre les sources de production d'une même donnée ;
* le système de communication garantit le transfert des exemplaires (ie le temps séparant la production d'un exemplaire, par une source, et la réception de cet exemplaire, par le site de stockage, est acceptable par les contraintes de temps du processus commandé) ;

2 définir une règle d'ordonnancement statique qui dépend des fréquences de production associées aux diverses sources de production des données. L'ordonnancement se fera à l'aide de cette règle.
Par exemple : prendra 2 exemplaires en provenance d'une source S1 pour chaque exemplaire en provenance d'une source S2 (si on considère que la fréquence de production associée à S2 et le double de celle associée à S1).

Une telle règle peut s'exprimer de manière analogue à une expression de chemins / CAMPBELL, 74/.

Exemple : 2S1-S2.

Cette solution permet de résoudre de manière statique le problème d'ordonnancement mais pose d'autres problèmes difficiles à résoudre :
. les règles d'ordonnancement ne sont simples à exprimer que dans le cas des productions périodiques ;
. comment continuer à ordonner les exemplaires en cas de panne d'une source (changement dynamique de la règle d'ordonnancement).

Mais si on tient compte des tendances actuelles pour la conception du matériel c'est-à-dire :
- matériel de plus en plus fiable,
- les services offerts par ce matériel sont synchrones (ce qui conduit à transformer tout ce qui n'est pas périodique en des phénomènes périodiques),

Cette deuxième solution peut être retenue pour l'ordonnancement des exemplaires des données.

b) Ordonnancement des occurrences d'évènements

La composition d'évènements fait, parfois, intervenir les instants de détection des évènements. Pour pouvoir comparer ces instants nous proposons deux solutions basées sur la "qualité" du matériel utilisé :

1 avec le développement technologique actuel on peut associer aux différents sites des horloges synchronisées à un Δt près (acceptable par les contraintes de temps du processus commandé).

Les occurrences d'évènement peuvent être datées à l'aide de ces horloges (qui, à un Δt près, constituent une seule horloge globale).

L'ordonnancement est donc fondé sur l'utilisation des instants réels de détection.

2 les différentes occurrences des évènements rentrant dans la composition d'un évènement sont transmises "immédiatement" au site sur lequel se trouve l'évènement composé. Ces occurrences sont datées, à l'arrivée, et ordonnancées à l'aide de cette datation.

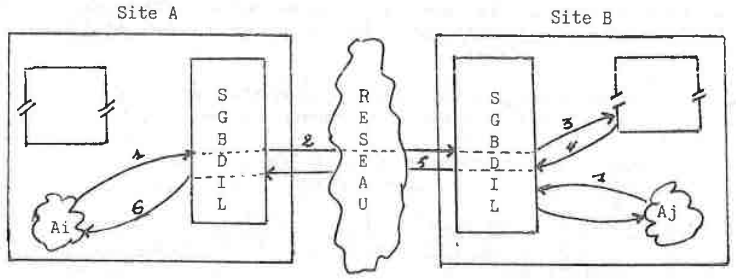
Une telle solution est à retenir si les délais de transmission sont négligeables vis-à-vis des contraintes de temps imposées par le processus commandé.

II 2.2 Etat approximatif de la BDI

La gestion de la BDI est assurée par un ensemble de gestionnaires locaux, chacun d'entre eux étant responsable de la gestion d'une partie de la BDI globale (on notera SGBDIL un gestionnaire local de BDI).

Une action qui désire accéder à une donnée s'adresse au gestionnaire de données du site local.

Ce dernier permet d'assurer l'accès à une donnée :
- soit en mettant directement la donnée à la disposition de l'action qui a fait la demande, si la donnée est implantée sur le site local ;
- soit en entrant en communication avec le gestionnaire distant de la donnée à accéder (d'où l'existence sur chaque site d'une table indiquant les sites d'implantation pour toute donnée de la BDI utilisée ou produite par le site).



Circuit d'accès à une donnée

La figure ci-dessus montre les accès, par deux actions Ai et Aj à la BDI. Ai accède à une donnée distante alors que Aj accède à une donnée locale.

Le temps nécessaire pour effectuer un accès à une donnée distante dépend non seulement du temps pris par les SGBDIL pour accéder à la donnée, mais aussi, du temps nécessaire à toutes les communications entre les SGBDIL.

Les communications inter-gestionnaires locaux de BDI peuvent avoir comme conséquence l'absence d'une image unique de la BDI. A cause des délais de communication et de l'évolution parfois très rapide des procédés, chaque site a sa propre vue de la BDI.

Cette vue est appelée état approximatif de la BDI.

Prenons l'exemple suivant :

Un site A désire consulter une donnée Di implantée sur un site B. Le site A communique avec le site B qui lui transmet l'état de la donnée Di à l'instant t1. Le site A reçoit la valeur de la donnée à l'instant t2 (t2>t1).

Si entre t1 et t2 la donnée Di a changé d'état le site A a une vue de Di différente de celle du site B.

Les écarts entre les différentes vues de la BDI peuvent être, plus ou moins, importants (ils dépendent de la qualité des communications et de la vitesse d'évolution du procédé) et pose un problème quant à l'acceptation de l'état approximatif :
- d'une part au niveau de l'utilisation des données par les actions (risque d'utilisation de mauvaises valeurs) ;
- et d'autre part au niveau de l'évaluation des conditions d'activation des actions (risque d'activation, à tort, d'actions).

Outre les incohérences qui peuvent résulter des diverses vues de la BDI, l'utilisation des évènements distants peut conduire à des mises à jour de la BDI inopportunes. En effet à tout évènement est associé un temps de réponse ; si le temps de communication des occurrences de l'évènement est supérieur au temps de réponse associé à l'évènement, alors la prise en compte de l'évènement, par le site distant, ne répond pas aux contraintes temps réel (la mise à jour sera inopportune dans un contexte temps réel).

Notion d'état approximatif acceptable

On appelle état approximatif acceptable par la commande tout état du système caractérisé par des vues différentes de la BDI (éventuellement différentes) qui ne conduit pas à des erreurs de commande et qui respecte les contraintes temps réel.

La répartition de l'application R(A,D,E) doit être telle que :
- toute donnée Di est utilisable sans conduire à des erreurs (cohérence des données) ;
- toute occurrence Oi d'évènement prise en compte ne conduit pas à des mises à jour inopportunes (ie faites en retard).

R (A,D,E) $\Rightarrow \forall di \in D$ utilisation (di) ne conduit pas à des erreurs
et $\forall Oi \in E$ la prise en compte (Oi) ne conduit pas à des MAJ inopportunes

A = action D = BDI E = évènement

Certes la preuve de l'existence d'un état approximatif acceptable, à tout instant de l'évolution du procédé est délicate, mais elle doit être étudiée avec soin lors du choix de la répartition du système de commande : la commande est basée sur l'exactitude des informations qu'elle utilise. Dans le prochain paragraphe nous étudierons des éléments qui permettent d'aider à minimiser les écarts entre les diverses vues de la BDI, c'est-à-dire qui conduisent à l'existence d'un état approximatif acceptable.

III APPROCHE POUR LA REPARTITION DES OBJETS

Nous venons d'évoquer la nécessité de la répartition des ACPI, mais aussi, les problèmes que soulève cette répartition.

La cohérence de la BDI (et en général, la sûreté de fonctionnement de l'application) dépend, en grande partie, de la qualité de cette répartition: elle dépend de la manière dont sont affectés les objets aux diverses machines de commande et des moyens de communication entre ces machines.

Plus les informations échangées sont cohérentes (ie qu'elles décrivent correctement l'état du procédé et de sa commande), plus la commande est saine. Les moyens de communication constituent "les artères" du système de commande : il est donc important pour l'"implanteur " d'en tenir compte.

La mise en place des mécanismes nécessaires au traitement des mauvais fonctionnements (données non acquises à temps, erreurs de communications etc...) est nécessaire mais insuffisante, à elle seule, car si le choix de la répartition et/ou du matériel supportant l'application est mauvais, le système passe une grande partie de son temps à traiter des anomalies de fonctionnement : ce système est inefficace aux yeux de l'industriel. Il est donc important de minimiser les situations de mauvais fonctionnements et ce en essayant de "renforcer" les points "vulnérables" du système. Pour y aboutir, nous proposons une approche qui consiste à :
- hiérarchiser le système de communication de manière à prendre en compte les problèmes liés aux échanges entre les divers équipements /THOMESSE,84/ ;
- hiérarchiser les machines supportant l'application pour mieux connaître les services qu'elles peuvent offrir ;
- et enfin répartir les objets en fonction des caractéristiques de ces objets et de celles des machines cibles.

III.1 HIERARCHISATION DES COMMUNICATIONS

En quoi consiste t-elle ?

L'analyse des flux d'informations manipulées par un système de commande fait apparaître deux classes de communications /THOMESSE,84/ :

a) les communications correspondant à l'échange d'informations entre capteurs, actionneurs et automates réflexes.

Ces communications sont :
. directement liées au procédé physique, elles sont donc soumises aux perturbations de l'environnement industriel ;
. Généralement, très fréquentes (les fréquences d'échantillonnage des informations en provenance du procédé et des applications des ordres au procédé sont élevées) ;
. et de natures diverses (les informations manipulées correspondent à des sources très variées).

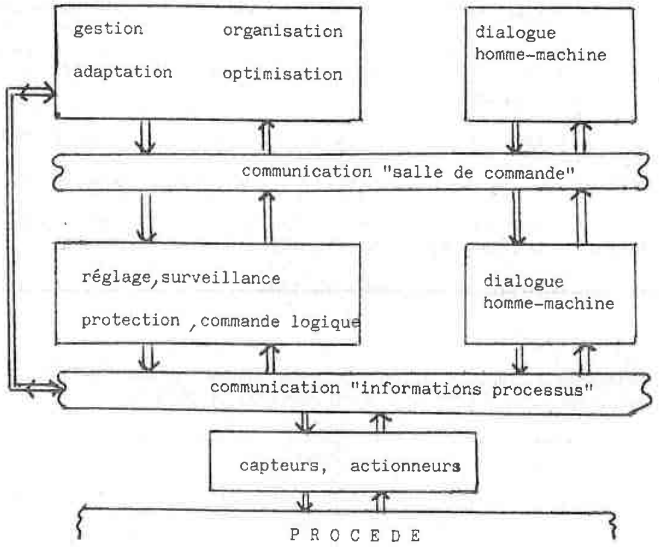
b) Les communications correspondant aux échanges d'informations entre les niveaux de la hiérarchie de commande.

Ces communications n'entrainent pas directement un dialogue avec le procédé. Comparées à la classe précédente ces communications :
- ne sont pas très fréquentes,
- mais peuvent correspondre à des échanges d'informations de grandes tailles, des fichiers, par exemple, pour établir des études statistiques .

Pour tenir compte des caractéristiques de ces communications (ie des caractéristiques des informations échangées) nous associons, à chaque classe de communication un support de communications (par la suite, nous étudierons le cas où ces deux supports sont distincts, mais pour les petites installations il est plutôt recommandé d'en avoir un seul , pour des raisons économiques et pour faciliter la gestion des communications).

Cette hiérarchisation des communications conduit à l'obtention d'un système de commande architecturé autour de deux canaux de communications (comme le montre la figure ci-après).

Nous appelons, comme dans /THOMESSE, 84/ :
- communication "informations processus" pour désigner la première classe de communications,
- et communication "sable de commande" pour désigner la deuxième classe de communications.



Hiérarchisation des communications dans une ACPI

Intérêt de la hiérarchisation des communications

Nous avons signalé, dans le chapitre II, que les objets relatifs à une ACPI sont de natures très diverses et peuvent être étudiés séparément et que la nature des objets doit être prise en compte au moment de l'implantation pour renforcer les points vulnérables du système: la hiérarchisation des communications répond à ce deuxième point.

La hiérarchisation des communications est importante dans la mesure où :
- les contraintes (fiabilité et temps de transfert) diffèrent en fonction de la nature des informations échangées ;
- les protocoles d'échange diffèrent aussi : si la communication entre deux sites distants nécessite l'emploi d'un protocole d'échange tel que celui de l'ISO /BNI, 81/, un tel protocole s'avère trop lourd et pénalisant pour la communication avec le procédé.

En associant à chaque classe de communication les moyens qui lui sont nécessaires, la hiérarchisation des communications permet de mieux respecter les contraintes liées à la sûreté de fonctionnement et aux temps de réponse (rappelons nous, la sûreté de fonctionnement et les contraintes de temps sont les principales caractéristiques d'une ACPI).

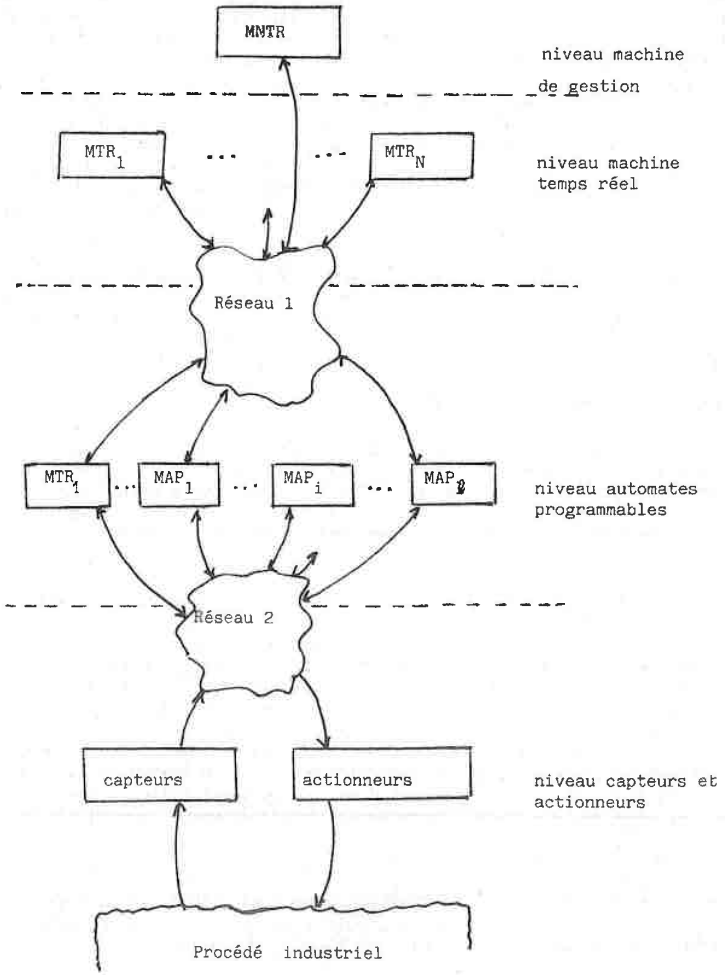
III.2 HIERARCHISATION DES MACHINES

Le développement incessant de la microélectronique a conduit à des restructurations dans les installations de commande. La commande qui était réalisée par une "grosse" machine, dont le coût était très élevé, est, de nos jours, répartie entre plusieurs "petites" machines spécialisées (les installations ainsi obtenues sont modulaires et permettent une grande disponibilité des applications).

Selon leur complexité, on distingue, dans les installations actuelles, trois types de machines :
- les machines non-temps réel utilisées pour la gestion de l'usine (comptabilité, factures...), généralement une seule machine est utilisée pour toute une installation (nous noterons ce type de machines par MNTR);
- les machines munies d'un exécutif temps réel et offrant les services classiques des systèmes d'exploitation(nous les noterons par MTR) ;
- et les machines type automates programmables industriels /LAURGEAU et Al,79/ ne disposant ni d'exécutif temps réel ni de système d'exploitation et spécialisées dans le dialogue avec le procédé (elles tiennent compte des temps de réponse et s'adaptent à l'environnement industriel),

Nous les noterons par MAP.

Cette classification des machines supportant une ACPI (il s'agit évidemment des ACPI pour la commande de procédés complexes) conduit à des systèmes ayant l'architecture illustrée par la figure ci-après :



Hiérarchisation des équipements dans une ACPI complexe

A titre d'exemple, le système de commande de la coulée continue de la Sollac-Sacilor (voir annexe A1) est basé sur la hiérarchisation des machines.

Pourquoi hiérarchiser les machines ?

La hiérarchisation des machines, introduite au moment de l'implantation des applications, est importante pour plusieurs raisons, entre autres :

1 Chaque type de machine offre des services qui lui sont particuliers ; l'implanteur , qui doit tenir compte de ces services, est guidé dans sa tâche de répartition des objets puisqu'il sera amené à choisir, pour un objet donné , la machine qui permet de supporter "le mieux" cet objet; par exemple il implante les actions relatives à l'acquisition des mesures au niveau automates programmables et les données statistiques au niveau machine de gestion (ou au niveau machines temps réel). Ainsi la connaissance des spécificités d'une machine aidera l'implanteur lors de son choix des objets à implanter sur cette machine.

2 Les objets à implanter ont été décrits, pendant la phase de conception, à l'aide d'un langage de description d'ACPI ; l'implantation de ces objets implique des transformations de la description des objets pour les adapter à la machine cible. Le type de la machine cible est important ici car il permet de déterminer les transformations à effectuer sur la description des objets (les transformations dépendent du type de la machine).

CONCLUSION

Avec la hiérarchisation des communications et des machines, nous voulions souligner l'importance du choix d'une structure matérielle pour l'implantation des ACPI et les conséquences d'un tel choix sur le respect des contraintes de temps et de sûreté de fonctionnnement.

Conseiller le concepteur pour l'aider à obtenir une description efficace (ie qui prévoit tous les cas, qui soit modulaire, etc...) est nécessaire mais insuffisant, il faut aussi rappeler à l'implanteur que ses choix sont décisifs et ont une grande influence sur la qualité de l'APCI réalisée. Les tendances actuelles concernant la hiérarchisation des systèmes de commande, cherchent à offrir à l'implanteur des moyens plus solides et plus performants pour lui permettre de répondre, le mieux possible, aux objectifs que s'est fixé le concepteur et qui se traduisent, pour l'industriel, par une augmentation du rendement du procédé.

III.3 CHOIX DE LA REPARTITION

1) Quelques critères de choix

Si l'on considère que l'implanteur choisit le matériel d'implantation en tenant compte de l'apport de la hiérarchisation des installations de commande, il doit aussi, répartir efficacement les objets sur ce matériel. Pour y parvenir il peut être guidé par :

- la fréquence et le temps d'exécution des actions, ce qui lui permet d'évaluer les contraintes de temps ;
- la fréquence de manipulation des données et des évènements : implanter chaque objet sur le site qui le manipule le plus souvent ;

- les priorités associées aux objets (priorités des actions et priorités de détection des évènements) desquelles devront tenir compte les machines (par exemple si on implantait tous les traitements urgents sur une même machine on risquerait de compromettre les contraintes de temps et de sûreté de fonctionnement) ;
- la nature des objets, par exemple :
  . une action d'entrée ou de sortie est liée à l'environnement, elle doit être implantée sur le site affecté à la partie de l'environnement à laquelle cette action est associée ;
  . un évènement doit être implanté près de sa source ;
- et la répartition géographique du procédé pour réduire les communications inter-stations. Le graphe associé à l'application (cf chapitre II) permet d'effectuer des regroupements d'objets et de voir ceux qui s'adaptent le mieux à l'environnement et qui créent le moins de liaisons entre les stations.

Le choix d'une structure d'exécution et d'une répartition des objets est dit efficace si :
- les contraintes de temps et de sûreté de fonctionnement sont respectées,
- tout objet est implanté sur le site qui le manipule le plus souvent,
- le taux d'occupation des machines est optimal (ie proche de 1).

L'implanteur cherchera donc à satisfaire, au mieux, les trois critères ci-dessus.

2) Description de la répartition

Après avoir affecté des objets, l'implanteur doit décrire les configurations de chacune des machines de l'installation de commande : c'est-à-dire décrire les objets implantés sur chaque machine. Pour ce faire, nous avons retenu la syntaxe suivante :

```
REPARTITION
        { MACHINE < nom de machine > =
                {nom d'objet ;}*
          FIN-MACHINE;}*
FIN-REPARTITION;
```

Cette structure qu'on peut qualifier de "table de répartition" est utilisée lors de la construction des systèmes opératoires des diverses machines : elle permet pour chaque site de connaître les lieux d'implantation des objets distants que ce site manipule.

3) Détermination des priorités associées aux objets :

Les traitements effectués par un système de commande sont de natures très diverses et chacun de ces traitements a une importance particulière vis-à-vis de la commande globale.

Si on considère qu'une même station (ou une même machine) peut assurer la gestion de plusieurs objets (donc effectuer plusieurs traitements différents), l'implanteur doit préciser les priorités associées à ces objets (par exemple l'arrêt d'urgence d'un appareil est plus prioritaire que l'impression d'un journal de bord).

Ces priorités sont prises en compte au moment de la définition de la charge d'une machine donnée (c'est-à-dire au moment de l'attribution des objets à cette machine).

Dans notre cas, il s'agit de préciser :

- la priorité de détection des évènements : chaque évènement correspond à un changement d'état particulier, la priorité associée à cet évènement découle du sens du changement d'état représenté par l'évènement. Le système opératoire devra tenir compte de cette priorité au moment de la détection des évènements.
- et la priorité associée aux actions: Cette priorité indique l'importance d'une action par rapport aux autres actions de l'application. Elle sera utilisée par le système opératoire au moment de l'allocation du processeur aux actions à activer (respect des priorités et préemption du processeur si nécessaire).

Pour tenir compte de l'attribut de priorité, l'implanteur étendra les définitions déjà obtenues, pour les actions et les évènements, comme suit :

- pour les évènements : EVENEMENT..... (reste inchangé) [PRIORITE = <entier>] FIN-EVT;
- pour les actions : ACTION <non action> [: PRIORITE = <entier>] ;
  :
  : reste inchangé
  FIN-ACTION <non action>;

Dans les deux cas si la priorité est omise, elle est remplacée par la priorité la plus faible.

IV CONCLUSION

La répartition des applications est nécessaire dans le cas de la commande des procédés industriels géographiquement distribués (contraintes de temps et de sûreté de fonctionnement obligent). Mais cette répartition soulève des problèmes :
- difficulté de synchronisation des différents sites,
- et existence de plusieurs vues de l'état du système.

L'étape du choix du matériel et de la répartition des objets sur ce matériel est, à notre avis, l'étape la plus délicate et la plus décisive dans la fabrication d'ACPI car la qualité du produit fini (ie la ACPI réalisée) dépend, en grande partie, de ce choix.

Quelques voies possibles, pour la résolution de ces problèmes, ont été discutées.

Nous avons essayé de guider l' implanteur en lui proposant une démarche qui consiste à :
- hiérarchiser l'installation de commande (ie hiérarchiser le système de communication et les machines utilisées),
- et répartir convenablement les objets sur cette installation.

Ce qui lui permet d'obtenir des installations :
- modulaires et fiables,
- qui assurent le respect des contraintes de temps et de sûreté de fonctionnement,
- et qui tiennent compte de la commande hiérarchisée des procédés complexes (à la hiérarchisation des fonctions correspond la hiérarchisation des équipements matériels).

Pour nous résumer, le respect des contraintes de temps et de sûreté de fonctionnement, dans le cas des ACPI doit être étudié :
- au niveau de la conception :
  . étudier le temps de réponse associé aux divers stimulus
  . recenser les cas de mauvais fonctionnements,
  . prévoir les traitements associés à ces mauvais fonctionnements,
  . décrire les applications de manière modulaire.
  etc...
- et au niveau implantation, en choisissant convenablement :
  . les équipements matériels,
  . et la répartition des objets.

CHAPITRE VII

ETUDE D'UNE ARCHITECTURE POUR L'IMPLANTATION DES SGBDIR

I INTRODUCTION

Nous venons de présenter l'aspect répartition des ACPI et souligner les conséquences qu'a la hiérarchisation des installations de commande sur le respect des contraintes de temps et de sûreté de fonctionnement.

Dans ce chapitre, nous nous interesserons à une architecture pour l'implantation des systèmes de commande ou SGBDIR (pour systèmes de gestion de BDI réparties) puisque les systèmes de commande sont fondés, selon notre approche, sur le concept de BDI.

Cette architecture repose sur la hiérarchisation de l'installation de commande (ie hiérarchisation des communications et des machines cibles utilisées).

Après avoir défini le critère de hiérarchisation retenu ici (ie pour lequel nous définissons une architecture) nous analysons les composantes logicielles nécessaires à la gestion de chaque machine de l'installation : à chaque type de machine correspondra un exécutif tenant compte des fonctions assurées par cette machine.

L'ensemble des composantes obtenues constitue le SGBDIR.

L'analyse de ces composantes nous permet de voir comment envisager l'implantation d'un SGBDIR fondé sur la hiérarchisation de l'installation de commande.

II HYPOTHESE SUR L'ARCHITECTURE MATERIELLE

Nous présentons ici les grandes lignes de l'architecture matérielle pour laquelle nous allons définir un SGBDIR.

L'hypothèse que nous faisons sur le matériel se situe au niveau du choix des supports de communication.

Le système de communication esthiérarchisé comme suit :
- la communication type "salle de commande" (cf chapitre VI paragrapheIII) est assurée en utilisant un ou plusieurs réseaux (on pourra consulter /CARPENTIER, 80/ , /COUR, 82/ , /DANG, 83/ , /NAFFAH, 80/ , /SOLMAN 80/ pour voir l'intérêt d'utiliser les réseaux locaux à ce niveau de communication) ;
- la communication type "informations processus" est assurée en utilisant un support particulier appelé FIP /THOMESSE, 84/ que nous présentons par la suite.

Nous faisons remarquer que le support FIP n'existe pas, pour le moment en tant que produit, et que nous cherchons, ici, à définir une architecture matérielle et logicielle basée sur la proposition FIP.

La conception orientée BDI, est à notre avis, complémentaire de la proposition FIP : elles sont toutes les deux fondées sur la manipulation des données et cherchent à accroître au maximum la cohérence des informations relatives à la commande d'un procédé industriel.

II.1 PROPOSITION FIP

Le groupe RLI (réseaux locaux industriels) du Ministère de l'Industrie et de la Recherche, après avoir étudié les réseaux locaux industriels actuels et les besoins des industriels en termes de communication avec les procédés, a défini un système pour les échanges d'informations entre capteurs, actionneurs et automates réflexes baptisé FIP (Flux d'Informations Processus) /THOMESSE, 84/ qui permet de :
- réduire le coût global du câblage,
- améliorer les services offerts par les systèmes de commande,
- et garantir les temps de réponse imposés par les procédés (même les plus sévères en contraintes de temps).

Un résumé des grandes lignes du projet FIP est donné dans l'annexe A4.

Principe de FIP

Le protocole FIP peut se résumer ainsi :

il s'agit d'attribuer une valeur ou une nomenclature à chaque objet transmis sur le support de communication (mesure, ordre,...).

Une station spéciale, appelée gestionnaire des nomenclatures, émet sur le médium des trames (une trame est composée d'une nomenclature et de l'information correspondant à cette nomenclature)..

La station productrice de l'objet dont la nomenclature est émise, modifie l'information correspondante (ie elle dépose la nouvelle valeur). Les autres stations, reliées au support, recopient l'information si elles veulent l'utiliser.

Les trames sont émises de manière continue en fonction des fréquences liées à la manipulation des objets (ie leur production et leur utilisation).

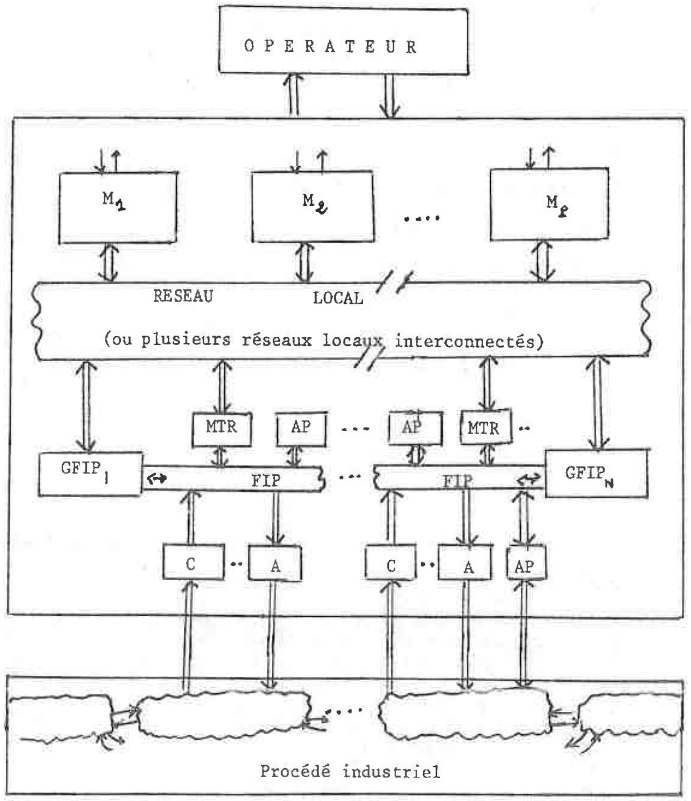
Une trame part du (et revient au) gestionnaire des nomenclatures. Lorsque le retour d'une trame est détecté il y a émission d'une nouvelle trame.

Avantages escomptés

L'utilisation de FIP présente les avantages suivants :
- l'accroissement de la fiabilité des informations échangées,
- le débit de transmission s'adapte à toutes les applications et permet de rafraîchir , à temps, la BDI et améliorer ainsi sa cohérence ;
- l'échange d'informations est synchrone : l'information est disponible en même temps, pour tous les équipements connectés à FIP, ce qui induit une auto-synchronisation des différents automatismes grâce au partage de données uniques ;
- le protocole FIP est simple d'utilisation. Il permet de vérifier aisément si les contraintes de temps sont vérifiées conformément aux fréquences de mise à jour de la BDI. Cette vérification est très difficile si on utilise les protocoles classiques (ISO ou autres) où la succession des couches fait que l'évaluation du temps de transfert d'un message est difficile à obtenir.

II.2 TYPE D'ARCHITECTURES MATERIELLES ETUDIEES

En utilisant le support FIP, nous structurons une installation de commande en deux niveaux (cf figure ci après) :
- un niveau composé d'un ensemble de machines temps réel connectées entre elles via un réseau local qui répond aux contraintes de temps et de sûreté de fonctionnement imposées à ce niveau ;
- et un niveau composé d'un ensemble (le cardinal de cet ensemble peut être 1) de parties du procédé contrôlées chacune par un ensemble d'équipements (capteurs, actionneurs, automates, ...) reliées par un support FIP.



Architecture d'une installation de commande utilisant FIP

Notations

M : Une machine temps réel (ou non)
GFIP : un gestionnaire FIP
MTR : un équipement connecté à FIP de type machine temps réel
AP : un équipement connecté à FIP de type automate programmable
C : un équipement connecté à FIP de type capteur
A : un équipement connecté à FIP de type actionneur.

A cette architecture matérielle correspond une architecture logicielle qui est celle du SGBDIR.

Définir l'architecture du SGBDIR revient à définir l'architecture logicielle de chaque équipement rentrant dans la gestion des données et/ou des actions manipulant ces données.

REMARQUE

L'architecture matérielle à laquelle nous nous intéressons, ici, répond aux critères d'hiérarchisation (cf chapitre VI). Le SGBDIR gérant cette architecture matérielle tient compte du principe "implanter chaque objet en fonction des caractéristiques de l'objet pour renforcer les points vulnérables du système".

III ARCHITECTURE PROPOSEE POUR L'IMPLANTATION DES SGBDIR

III.1 DEFINITIONS ET CONCEPTS GENERAUX

L'architecture matérielle retenue dans le paragraphe précédent conduit à la distinction de quatre types de machines, et ce en fonction des échanges auxquels participent ces machines :
- les machines temps-réel connectées au réseau local uniquement (elles ne sont pas en relation directe avec le procédé) ;
- les machines temps réel connectées à FIP (et éventuellement au réseau local) ;
- les machines jouant le rôle de gestionnaires FIP;
- et les automates programmables, actionneurs, ... connectés au support FIP.

Chaque machine gère une ou plusieurs données (ie utilise, produit et/ou stocke des données).

a) Définition du concept de SGBDIL

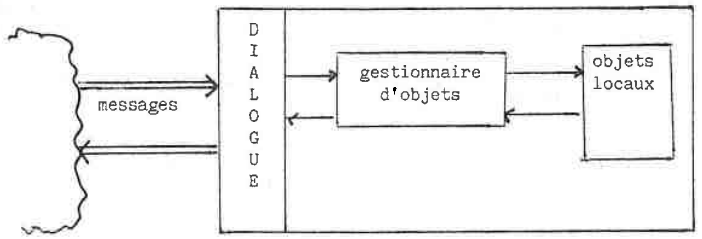
Nous appelons SGBDIL (ou système de gestion de base de données industielles local) le logiciel de gestion d'une machine donnée.

Ce logiciel doit être capable de :
- gérer les données assignées à la machine, c'est-à-dire :
  . pouvoir accéder aux données locales et aux données distantes,
  . pouvoir communiquer les données locales aux SGBDIL distants.

- gérer les conditions de mise à jour des données, c'est-à-dire :
  . détection des évènements et gestion du temps,
  . évaluation des conditions de mise à jour,
  . et activation des actions.

- et d'assurer l'exécution des actions implantées sur la machine.

De manière générale un SGBDIL peut se décomposer en trois parties :
- un ensemble d'objets locaux (données, actions et évènements),
- un gestionnaire d'objets qui assure la manipulation de ces objets,
- et une interface de dialogue avec les SGBDIL distants (envoi et réception des demandes et des réponses pour l'accès aux objets)



Architecture générale d'un SGBDIL

b) Types de SGBDIL

La complexité de l'architecture d'un SGBDIL dépend du type de la machine qu'il gère.

En tenant compte de la hiérarchisation des machines nous dégageons cinq types de SGBDIL :
- SGBDIL/MTR désigne un SGBDIL pour la gestion d'une machine temps réel (MTR) connectée au réseau local ;
- SGBDIL/GFIP désigne un SGBDIL pour la gestion d'une machine jouant le rôle d'un gestionnaire FIP ; cette machine est connectée au réseau local et ne supporte pas d'actions ;
- SGBDIL/MAP désigne un SGBDIL pour la gestion d'une machine type automate programmable, actionneur ou capteur connecté à FIP. Ce type de machine n'a pas d'éxécutif ;
- SGBDIL/MTR-FIP désigne un SGBDIL pour la gestion d'une machine temps réel connectée à FIP et au réseau local ;
- SGBDIL/GFIP-RL désigne un SGBDIL pour la gestion d'une machine jouant le rôle d'un gestionnaire FIP sur laquelle sont implantées des actions.

Les trois premiers types de SGBDIL sont des types de base, les deux derniers sont construits à partir des précédents.

c) Définition d'un SGBDIR

Nous appelons SGBDIR l'ensemble des SGBDIL associés aux différentes machines rentrant dans la composition de l'installation de commande.

$$SGBDIR = \{SGBDIL_1, .., SGBDIL_N\}$$

N : nombre de machines

Définir l'architecture du SGBDIR revient à définir l'architecture des divers types de SGBDIL le composant.

Dans les paragraphes suivants nous allons esquisser les grandes lignes de la structure de chacun des types de SGBDIL.

d) Implantation des actions

Pour comprendre les architectures des divers SGBDIL, il est important de préciser dès à présent la solution retenue pour implanter les actions.

Une action sera composée de trois parties :
- un prologue qui permet de demander les entrées de l'action au gestionnaire des données,
- un corps qui représente l'action elle-même,
- et un épilogue qui permet d'appeler le gestionnaire des données pour mettre à jour la BDI en fonction des résultats obtenus par cette action.

Action ⟶ { prologue, corps de l'action, épilogue }

Le prologue de l'action doit tenir compte de la non-disponibilité des entrées de l'action (par exemple en cas de panne d'un site distant sur lequel sont implantées une ou plusieurs entrées de l'action).

Pour détecter qu'une entrée est non disponible le prologue devra armer un "réveil" pour cette entrée. Si au bout d'un temps, fixé au moment de l'implantation, l'entrée n'est pas reçue par le prologue, celui-ci déclare cette entrée comme étant non disponible.

Lorsqu'une ou plusieurs entrées de l'action ne sont pas disponibles le prologue de cette action décide :
- soit de déclarer l'action en état de défaut (elle ne peut pas s'exécuter puisque ces entrées ne sont pas toutes disponibles) ;
- soit de "relancer" l'action en utilisant, pour les entrées non disponibles, les valeurs précédentes ;

Cette deuxième possibilité, si elle est acceptée, conduit l'implanteur à définir le nombre de fois qu'une même valeur peut être réutilisée.

Ainsi l'implanteur devra fournir les paramètres nécessaires à la prise de décision en cas d'indisponibilité des entrées : il associe à chaque entrée une durée et le nombre de fois qu'une même valeur de cette donnée peut être réutilisée (si ces deux paramètres sont absents ils sont considérés comme nuls).

Ces paramètres s'ajoutent dans la clause ENTREES de l'action :

INTERFACE
ENTREES : { <nom de donnée> [<la durée>][/<nombre>] , }*

En fonction des SGBDIL l'action peut être implantée sous forme d'une tâche temps réel (cas des SGBDIL/MTR, SGBDIL/MTR-FIP et SGBDIL/FIP-RL) ou sous forme d'un sous-programme (cas des SGBDIL/MAP).

e) Quelques remarques

1 L'implantation physique des données de la BDI (sous forme de tables en mémoire centrale, de fichiers sur mémoire secondaire, etc...) n'est pas traitée ici.

2 Les SGBDIL que nous allons développer, par la suite, ont une structure modulaire que nous obtenons par des raffinements successifs /WIRTH, 71A/.

Les structures que nous donnerons se situent à un niveau de détail suffisant permettant de comprendre les idées directrices pour implanter un SGBDIR.

Elles peuvent être raffinées davantage pour aboutir à des structures directement "implémentables".

3 Pour aboutir à des modules asynchrones et indépendants, le lien retenu, pour faire coopérer ces modules, c'est la communication de messages via des files.

Une file permet, à un module, de déposer des messages à un autre module qui les traitera une fois qu'il est réveillé.

En raffinant la structure d'un module une file peut se scinder en plusieurs files, chacune de ces files contient un type particulier de messages : on va du général au particulier.

4 Pour présenter les architectures des SGBDIL nous utiliserons les représentations schématiques suivantes :

 : désigne, selon les cas, un module multi-tâches, une seule tâche ou un sous-programme

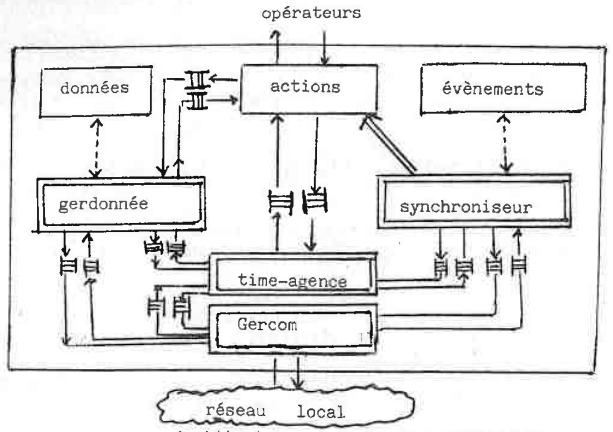
 : désigne une file

 : désigne un lien d'accès à des informations

 : désigne un lien de contrôle ou d'activation

III.2 ARCHITECTURE D'UN SGBDIL/MTR

Nous nous intéressons ici à l'architecture du système de gestion d'une machine temps réel connectée au réseau local. Cette architecture se compose des éléments suivants :
- un module de gestion des communications via le réseau local : GERCOM,
- un module de gestion de l'accès aux données : GERDONNEE,
- un module de synchronisation : SYNCHRONISEUR,
- un module de gestion du temps : TIME-AGENCE
- et les objets gérés par le site (données, actions et évènements).



Architecture générale d'un SGBDIL/MTR

1) Module TIME-AGENCE

Ce module offre des services relatifs au temps :
- demande de délais (ou de temporisations),
- demande de l'heure locale.

Une demande de délai est constituée d'une période et d'un message à retourner une fois le délai dépassé.

Il reçoit des demandes en provenance des autres modules et les traite :
- en donnant l'heure locale au demandeur,
- ou en initialisant un réveil pour le demandeur et avertir celui-ci par un message lorsque le délai demandé vient à échéance.

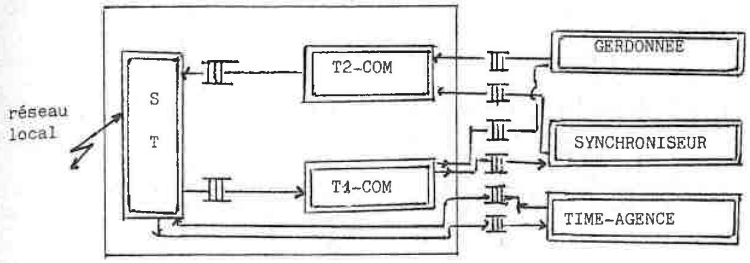
Le noyau central de ce module est étudié en détail dans la troisième partie de cette thèse (il s'agit d'une agence horlogerie).

2) Module GERCOM

Le module GERCOM permet d'assurer les échanges d'informations entre les modules GERDONNEE et SYNCHRONISEUR d'une part et les autres SGBDIL d'autre part.

Il reçoit les messages à transmettre et les émet à l'aide d'une station de transport (la station de transport sera étudiée en détail dans la troisième partie de cette thèse). A l'aide de cette même station de transport, il reçoit des messages en provenance des SGBDIL distants, traite les messages pour connaître le module destinataire puis les transmet soit à GERDONNEE soit à SYNCHRONISEUR.

Il peut alors se décrire par :
- une station de transport ST,
- une tâche d'aiguillage des messages reçus de SGBDIL distants (T1-COM),
- et d'une tâche qui reçoit les demandes d'émission de messages en provenance des modules locaux (T2- COM)



Module GERCOM

Pour les problèmes de gestion des time-outs liés aux communications, la station de transport s'adresse au module de gestion du temps.

3) Le module GERDONNEE

Ce module permet de gérer et de contrôler les accès aux données (accès au sens production ou utilisation).

Il reçoit des demandes d'accès aux données et les traite comme suit :

1er cas : la demande émane d'une action interne.

. Si la donnée à accéder se trouve sur le site local, l'accès à la donnée est effectué directement.
. Si la donnée à accéder se trouve sur un site distant il y a envoi d'une demande d'accès à ce site. Cette demande est véhiculée via le module GERCOM. Le SGBDIL local attend la réponse du SGBDIL distant concerné par cet accès. Une fois la réponse obtenue, le SGBDIL local répondra à l'action qui a fait la demande. Mais, pour une raison quelconque (erreur de communications, site distant inactif), cette réponse peut ne pas être obtenue, au bout d'un délai fixé, il est alors indispensable pour le module GERDONNEE de connaître la non- réponse et de signaler le fait à l'action ayant effectué la demande. Les temporisations associées aux diverses demandes sont gérées par le module TIME-AGENCE sur demande de GERDONNEE.

2ème cas : la demande provient d'un site distant.

C'est une action distante qui désire accéder à une donnée locale (locale par rapport à ce site). La demande est alors directement traitée par le module GERDONNEE.

Dans les deux cas si la demande d'accès est une utilisation, GERDONNEE transmet au demandeur l'exemplaire accédé.

Une structure qui tient compte des fonctions réalisées par le module GERDONNEE peut corrrespondre à :
- une tâche T1-GERDON qui reçoit les demandes d'accès et les traite comme suit :
  . si la demande d'accès correspond à une donnée locale, la demande est transmise à la tâche d'accès local (T4-GERDON) ;
  . si la demande d'accès correspond à une donnée distante, la demande est transmise à la tâche des accès distants (T2-GERDON).

- une tâche T2-GERDON qui s'occupe des demandes d'accès aux données distantes. Elle traite ces demandes comme suit :
  . envoi d'une demande au site distant,
  . attente de la réponse du site distant,
  . envoi de la réponse (en cas d'accès en utilisation) à la tâche des réponses T3-GERDON.
- T3-GERDON permet d'envoyer la réponse au demandeur, cette réponse lui provient soit de la tâche des accès distants soit de la tâche des accès locaux ;
- T4-GERDON permet d'accéder aux données locales. Elle rend la réponse (si nécessaire) à la tâche T3-GERDON pour la communiquer au demandeur (action local ou GERCOM) ;
- et à une structure de données à laquelle accèdent les 4 tâches précédentes ; nous la schématisons par la table TABDATA définie comme suit:

Nom de la donnée	Pointeur sur la structure de la donnée	Indicateur de localisation	Site de stockage

Cette table contient une entrée pour chaque donnée manipulée par les actions locales et/ou stockée sur le site local.

L'indicateur de localisation indique si la donnée se trouve sur le site local ou bien si elle est stockée sur un site distant désigné par "site de stockage".

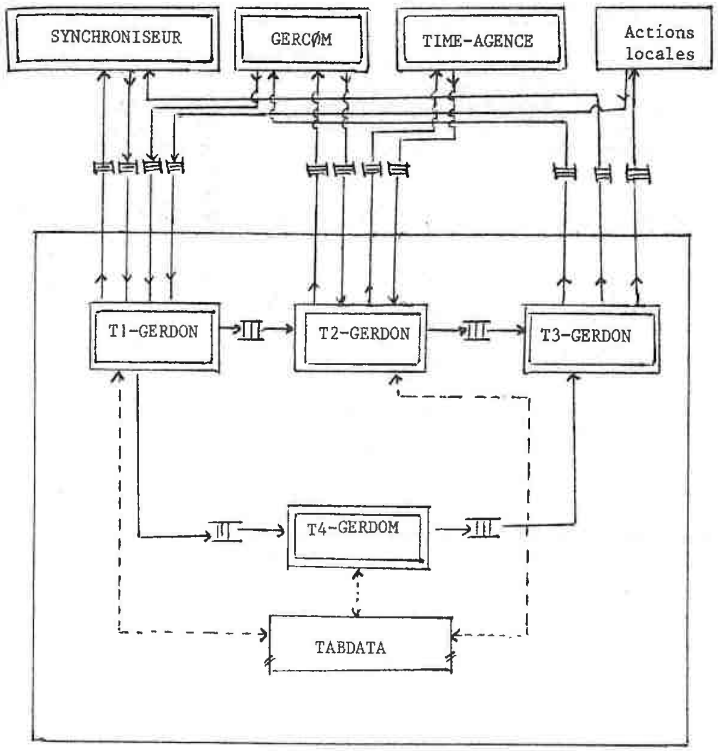
REMARQUES :

1 La gestion des évènements et des règles de mise à jour, en général, fait intervenir l'état des données soit pour détecter un évènement soit pour évaluer une condition sur l'état du système (revoir éventuellement le chapitre IV). Ainsi le module SYNCHRONISEUR peut demander (accès en consultation) l'état des données. Mais aussi nous avons signalé, dans le chapitre IV, que la détection des évènements liés aux données se fait à chaque fois qu'une donnée est modifiée (une production d'une donnée peut ou non conduire à la détection d'un évènement) ; de ce fait la tâche T1-GERCOM doit, au moment de la prise en compte d'une demande d'accès, communiquer, au module SYNCHRONISEUR, le nom et le nouvel état de la donnée à accéder (si cet accès est une production).

2 La structure de la donnée pointée à partir de la table TABDATA contient :
  - les informations représentées par la donnée (par exemple sous forme d'un "Record PASCAL") ;
  - et les procédures d'accès à la donnée (ie sa production, sa consultation et sa consommation).

Nous obtenons, ainsi, une représentation similaire à celle des types abstraits /DERNIAME, 79/, composée d'informations et d'opérations d'accès à ces informations.

La structure obtenue pour le module GERDONNEE est la suivante :



Module GERDONNEE

4) Module SYNCHRONISEUR

Ce module représente le "moteur" des activités du SGBDIL. Il a pour fonctions :
- la détection des évènements simples et composés,
- la gestion des évènements détectés en vue d'une activation, d'une composition ou d'une communication de ces évènements aux SGBDIL distants qui utilisent ces évènements ;
- l'activation des actions ;
- et la gestion des états des actions (ces états interviennent dans la détection des évènements et dans les conditions associées aux règles d'activation des actions).

A) Détection des évènements simples

Rappelons qu'un évènement simple correspond soit :
- à une apparition d'interruption (ou IT),
- à un changement de valeur d'une donnée de la BDI,
- à un changement d'état d'une action.

La détection des évènements simples (ou élémentaires) peut être alors réalisée par une tâche DTCT-EVS qui reçoit :
- les interruptions détectées grâce à une tâche DTCT-IT de détection des interruptions ;
- les changements d'état des données via le module GERDONNEE ;
- et les changements d'état des actions via la tâche GEST-ETAT.

Les messages reçus par DTCT-EVS sont analysés et peuvent conduire ou non à la détection d'évènements simples.

Cette analyse implique l'utilisation de trois tables :
- une table TB-IT qui indique, pour chaque IT, la liste des évènements construits à partir de cette IT ;
- une table TB-AC qui indique, pour chaque action, la liste des évènements construits à partir de cette action (en fonction d'un état particulier de cette action) ;
- et une table TB-DON qui indique, pour chaque donnée, la liste des évènements construits à partir de cette donnée (en fonction d'une condition particulière sur l'état de cette donnée).

Numéro d'inter-ruption	Liste des évène-ments

TB-IT

Nom d'action	Etat	Liste des évènements

TB-AC

Nom de donnée	Liste des évènements

Tout évènement détecté par DTCT-EVS est signalé à la tâche de gestion des évènements, GEST-EVT, qui gère les évènements détectés.

B) Gestion des évènements détectés

La tâche GEST-EVT s'occupe de la gestion des évènements simples ou composés détectés par le site local ou transmis pour les sites distants.

Elle reçoit :
- les évènements simples détectés par la tâche DTCT-EVS,
- les évènements composés détectés par la tâche DTCT-EVC,
- et les évènements détectés par les sites distants, via le module GERCOM.

Elle utilise deux tables :
- une table TB-ACT qui indique pour chaque évènement la liste des actions pouvant être activées via cet évènement (ces actions peuvent être locales ou distantes).

Nom-evt	Liste des actions

→ | action | site |

type des éléments de la liste

- et une liste TB-COMP qui indique pour chaque évènement la liste des évènements composés à l'aide de cet évènement (ces évènements pouvant être locaux ou distants)

Nom évènement	Liste des évènements composés

TB-COMP

→ | non-event | site |

type des éléments de la liste

Les traitements effectués par la tâche GEST-EVT sont :
- transmission de l'évènement détecté à la tâche de composition des évènements si cet évènement apparaît dans une ou plusieurs compositions d'évènements ;
- transmission de l'évènement détecté à la tâche d'activation des actions (tâche TESTEUR) si cet évènement apparaît dans une ou plusieurs règles d'activation locales ;
- transmission de l'évènement aux sites distants utilisant l'évènement soit pour des activations d'actions soit pour des compositions d'évènements.

C) Détection des évènements composés

Tout évènement reçu par la tâche GEST-EVT est transmis à la tâche de détection des évènements composés DTCT-EVC si celui-ci apparaît dans la composition d'un évènement local.

En utilisant une table TAB-EVCOMP indiquant pour chaque évènement la liste des évènements composés à l'aide de cet évènement (il s'agit des évènements composés implantés sur le site local) et les évènements reçus, la tâche DTCT-EVC permet de détecter les évènements composés. Les éléments pointés par la table TAB-EVCOMP contiennent les noms des évènemnts composés.
A chaque évènement composé est associé une entrée dans la table TB-EVCOMP qui contient :
- le nom de l'évènement,
- et la structure de composition.

Par structure de composition nous entendons :
- liste des évènements rentrant dans la composition; pour chacun de ces évènements on a le nom de l'évènement et un compteur qui indique le nombre d'occurrencesde cet évènement qui n'ont pas encore été utilisées pour produire l'évènement composé (revoir éventuellement le chapitre IV pour les règles de compositon d'évènements)
- la liste des opérateurs de composition et la manière dont ils sont utilisés.

A la réception d'un évènement la tâche DTCT-EVC traite tous les évènements composés à l'aide de cet évènement comme suit :
- détection de l'évènement composé et remise à jour des compteurs d'occurrences associés aux divers évènements rentrant dans la composition, et ce, en fonction des opérateurs de composition ;
- ou bien si on ne peut pas détecter l'évènement composé, incrémenter le compteur d'occurrences associé à cet évènement dans la liste des évènements composant l'évènement composé traité.

Toute occurrence d'évènement détectée, par la tâche DTCT-EVC, est transmise à la tâche GEST-EVT.

TAB-EVCOMP

Nom évènement	Structure de composition	→ liste d'évènements et opérateur de composition

D) Gestion des états des actions

Tout changement d'état d'une action (fin, début ou défaut d'une action) est signalé à la tâche GEST-ETAT par l'action qui change d'état.

Ces états sont stockés dans une table TB-ETAT

Nom action	Etat

TB-ETAT

et sont communiqués :
- à la tâche DTCT-EVS pour la détection des évènements simples liés aux actions ;
- et à la tâche TESTEUR sur demande de celle-ci (nous rappelons qu'une règle d'activation peut faire intervenir une condition portant sur l'état des actions).

Dans le cas où la tâche TESTEUR demande l'état d'une action distante, la tâche GEST-ETAT procède à une communication avec le site distant pour connaître l'état demandé; d'où l'utilisation par GEST-ETAT d'une table qui indique les sites d'implantation des actions qu'elle référence (appelons cette table TB-ET-ACT).

Nom action	Site

De manière réciproque la tâche GEST-ETAT peut recevoir, via le module GERCOM, des demandes sur les états des actions locales. Elle répond à ces demandes via GERCOM.

E) Activation des actions

L'activation des actions fait intervenir les règles d'activation de ces actions (ou encore les règles de mise à jour de la BDI).
Une règle d'activation peut devenir vraie (ie il y a activation de l'action associée à cette règle) si :
- l'évènement qu'elle attendait a été signalé par GEST-EVT ;
- ou bien si le délai qu'elle attendait vient à échéance .

Pour assurer l'activation des actions, nous utilisons une tâche TESTEUR qui reçoit :
- les évènements détectés,
- et les fins de délais demandés pour les actions (un délai est signalé ici à la tâche TESTEUR par le module TIME-AGENCE, le message reçu par TESTEUR contient le nom de l'action à laquelle correspond ce délai).

Puis décide d'activer une ou plusieurs actions. L'activation des actions (matérialisées par des tâches temps réel) est assurée par un noyau temps réel, NOYAU-TR (dans la troisième partie de cette thèse nous présentons le concept de noyau temps réel en détail).

Pour décider de l'activation des actions, la tâche TESTEUR utilise :
- une table TB-EVT-ACT qui indique, pour chaque évènement, les règles d'activation où il intervient :

nom-évènement	liste d'actions	⇢ Noms d'actions

TB-EVT-ACT

- et une table TB-REGLESMAJ qui indique, pour chaque action locale, la règle d'activation associée à cette action

La règle d'activation est décrite à l'aide :
. du type de la règle de mise à jour (périodique, aléatoire, à instant fixe,...) ; ce type conditionne la forme de la structure de la règle d'activation ;
. et de la structure de la règle d'activation. Cette structure peut contenir :
  - les noms des évènements activant l'action et les occurrences prises en compte (toutes, la première uniquement) pour ces évènements,
  - des périodes,
  - des instants précis,
  - des conditions de prise en compte,

et ce en fonction du type de la règle de mise à jour.

Nom-action	Type de règle	Structure de la règle

TB-REGLESMAJ

1/A la réception d'un évènement la tâche TESTEUR opère de la manière suivante (pour chacune des actions concernées par cet évènement):

a) elle teste si l'action peut être activée :
   - si l'action est activée il y a transmission d'un message au noyau temps réel et consommation de l'occurrence pour cette action ;
   - si l'action n'est pas activée l'occurrence est consommée et si dans la règle d'activation il y a expression de délai après l'évènement (ie sur evt ... attendre Δt) alors il y a demande d'un délai (l'action pourra être activée à la fin de ce délai).
     Cette demande est adressée au module TIME-AGENCE. La tâche TESTEUR mettra une marque pour indiquer que cette action est en attente d'un délai.

b) Nous avons signalé (chapitre IV) que dans certaines règles de mise à jour :
   - un évènement ne peut être pris en compte qu'après un autre évènement E,
   - des activations cycliques débutent sur détection d'un évènement E1 et/ou se terminent sur détection d'un évènement E2.

Si l'évènement reçu, par la tâche TESTEUR, a le sens de l'évènement E, E1 ou E2, la structure associée à la règle d'activation concernée par l'évènement reçu est mise à jour pour tenir compte de cet évènement :
   - les opérations cycliques peuvent commencer ou bien sont terminées,
   - les occurrences d'un autre évènement peuvent être prises en compte.
   - etc...

2/A la réception d'un message de type fin de délai, la tâche TESTEUR opère comme suit :
- si la fin de délai indique la fin d'attente après la détection d'un évènement dèjà pris en compte, il y a activation de l'action concernée ;
- si la fin de délai indique l'écoulement d'un Δt pour une activation périodique, il y a activation de l'action concernée ;
- si la fin de délai correspond à un instant précis,
  . l'action concernée est activée (il s'agit d'une activation à instant fixe),
  . la structure associée à la règle d'activation traitée est mise à jour si cet instant sert de repère pour la règle (voir chapitre IV).

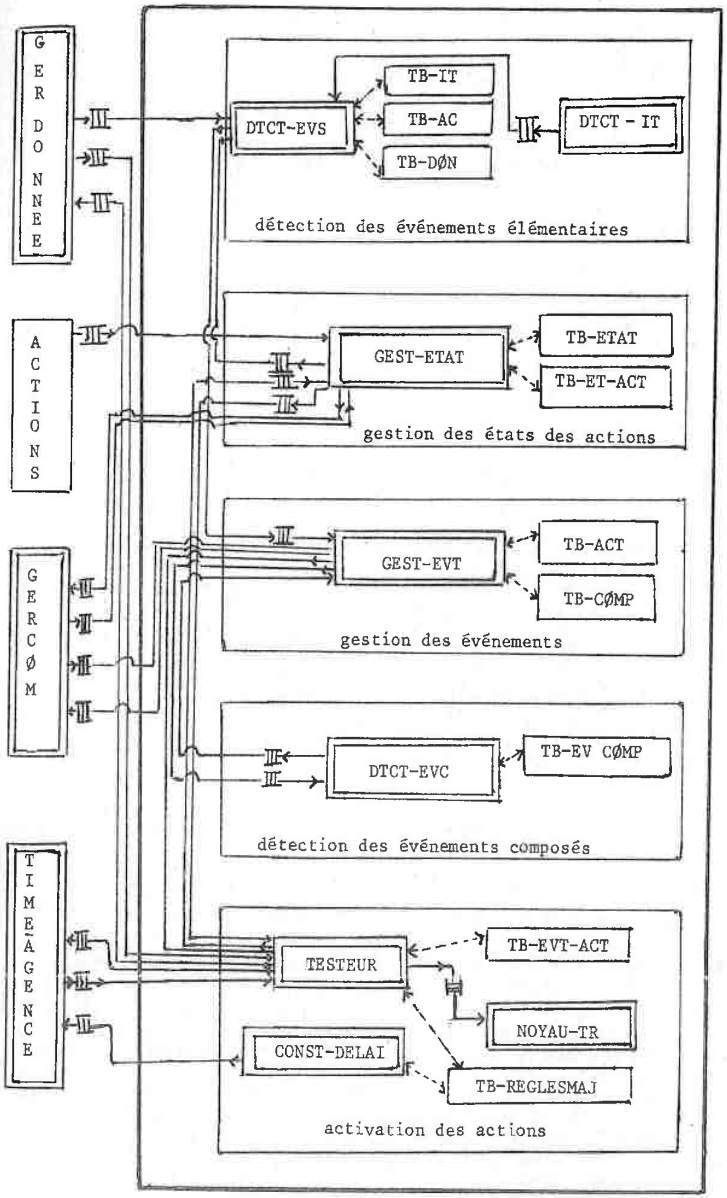
Il est clair que les activations précédentes ne sont possibles que si les conditions associées à la règle traitée sont vérifiées.

Gestion du temps

Les mises à jour périodiques et celles à instants fixes, font intervenir le temps. Pour assurer l'activation des actions utilisant ce type de règles d'activation, nous utilisons une tâche appelée CONST-DELAI (pour la construction des délais). La tâche CONST-DELAI accède cycliquement à la table TB-REGLESMAJ pour connaître les règles nécessitant le déclenchement d'un délai (ce sont les mises à jour périodiques ou à instants à fixes).

Les délais construits (ils sont composés d'une période et du nom de l'action concernée par le délai) sont transmis au module TIME-AGENCE. L'expiration de ces délais sera signalée à la tâche TESTEUR qui la traitera comme indiquée précédemment.

La structure obtenue pour le module SYNCHRONISEUR est la suivante :



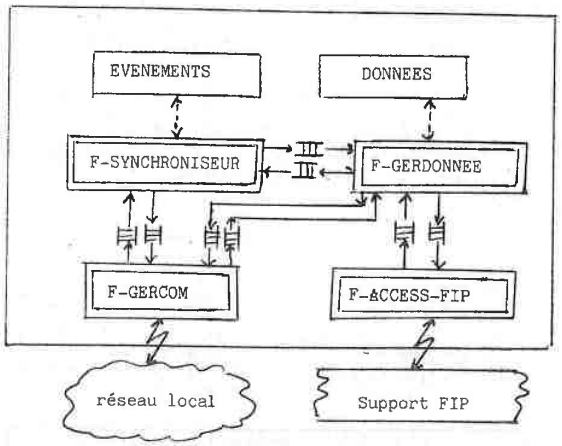
Module synchronisateur

III.3 ARCHITECTURE D'UN SGBDIL/GFIP

Il s'agit ici de l'architecture d'un système pour la gestion d'un support FIP, appelé aussi gestionnaire de nomenclatures.

Ce système gère d'une part le support FIP, en diffusant des trames selon les fréquences associées à la manipulation des objets, et d'autre part la communication avec les autres sites reliés au réseau local puisque tous les SGBDIL coopèrent à la réalisation d'un objectif global commun.

Nous pouvons alors décomposer cette architecture comme suit :
- un module pour la gestion de la communication avec les autres sites du réseau : F-GERCOM
- un module pour gérer les fréquences d'émission des trames sur le support FIP : F-SYNCHRONISEUR,
- un module pour la gestion des données : F-GERDONNEE,
- un module pour la gestion du support FIP (émission et réception des trames): F-ACCESS-FIP ;
- et les objets, données et évènements, implantés sur le site (pas d'actions).



Architecture générale d'un SGBDIL/GFIP

A) Module F-GERCOM

Ce module permet d'assurer la communication entre le gestionnaire FIP et les autres sites du réseau. Pour ce faire il reçoit :
- des demandes d'accès aux données locales par des sites distants ; ces demandes sont adressées au module F-GERDONNE qui les traite. Les réponses rendues, par F-GERDONNEE, sont transmises, par F-GERCOM, aux sites distants concernés ;
- et des messages liés à la détection des évènements distants utilisés par le site local. Ces messages sont adressés au module F-SYNCHRONISEUR. De même, le module F-SYNCRHONISEUR transmet, via F-GERCOM, les messages associés aux évènements utilisés par les autres sites.

La décomposition de ce module en tâches est similaire à celle de GERCOM du SGBDIL/MTR.

B) Module F-SYNCHRONISEUR

Comme dans le cas d'un SGBDIL/MTR, le module F-SYNCHRONISEUR s'occupe de tous les problèmes de synchronisation.

Ce module a la même structure que celle du module SYNCHRONISEUR d'un SGBDIL/MTR à la différence près :
les tests faits par la tâche TESTEUR ne conduisent pas à l'activation d'actions (il n'ya pas d'actions et de noyau temps réel ici) mais à un message transmis au module de gestion des données qui désigne la nomenclature à transmettre sur le support FIP. Les tables sont semblables à celles de SYNCHRONISEUR mais les actions sont remplacées, dans ce cas, par des nomenclatures.

Les fréquences de manipulation des objets sont exprimées, ici, de la même manière que les règles d'activation des actions.

Comme le module F-SYCHRONISEUR utilise des données locales et distantes et des évènements locaux et distants, afin de connaître les nomenclatures à transmettre, il s'adresse aux modules F-GERCOM et F-GERDONNEE pour obtenir les informations qui lui sont utiles.

C) Module F-GERDONNEE

Ce module assure deux fonctions :
- la gestion des accès aux données par les sites distants ou par le module F-SYNCRHONISEUR ;
- et la gestion des trames.

1) Gestion des accès aux données
- accès aux données locales :

Ces accès sont demandés par le module F-SYNCHRONISEUR ou par les sites distants. Les demandes d'accès sont définies à l'aide des noms des données auxquelles on veut accéder. Ces noms sont globaux (ie des noms de données de la BDI). Mais de l'autre côté le gestionnaire FIP manipule des objets élémentaires réprésentés par une table TB-NCLATURE.

Nom de nomenclature	Valeur

TB-NCLATURE

A une donnée-BDI(Di) peut correspondre un ou plusieurs objets FIP; $O_1 \ldots O_n$.

Pour pouvoir construire des données-BDI à partir des objets FIP, on utilise une table qui indique pour chaque donnée-BDI utilisée la liste des objets FIP qui la construisent. Cette table appelée TB-FOBJET est définie comme suit :

Nom donnée	Liste objets	numéros de nomenclatures

A l'aide de ces deux tables et, en utilisant les procédures d'accès adéquates, on peut satisfaire les demandes d'accès aux données locales.

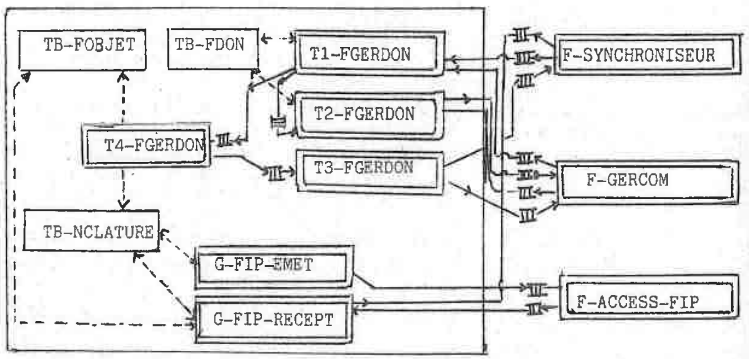
- Accès aux données distantes :
Ces accès sont demandés par le module F-SYNCHRONISEUR seulement (pour évaluer les règles d'émission des trames).

Pour satisfaire ces demandes le module F-GERDONNE rentre en communication avec les sites distants via le module F-GERCOM.

La fonction de gestion des données peut être assurée par les tâches suivantes :
- une tâche T1-F-GERCOM qui analyse la demande d'accès, puis la transmet à la tâche T2-F-GERCOM si c'est une demande d'accès à une donnée distante ou à la tâche T4-F-GERCOM si c'est une demande d'accès à une donnée locale. Une table TB-FDON permet de connaître les sites d'implantation des données.
- une tâche T2-F-GERDON qui s'occupe des demandes d'accès à distance (elle rentre en communication avec les sites distants pour satisfaire les demandes d'accès). Elle transmet les réponses obtenues à la tâche T3-F-GERDON.
- une tâche T3-F-GERDON qui rend la réponse au demandeur.
- une tâche T4-F-GERDON qui accède aux données locales (elle a connaissance de leur structure) et rend la réponse à la tâche T3-FGERDON si la demande est une utilisation.

2) Gestion des trames

Cette fonction consiste, d'une part à recevoir du module F-SYNCHRONISEUR les nomenclatures pour lesquelles il faut émettre des trames, construire des trames en utilisant la table des nomenclatures TB-NCLATURE et à communiquer ces trames au module d'accès au support FIP, F-ACCESS-FIP, d'autre part à recevoir des trames en provenance du module F-ACCESS-FIP et à mettre à jour la table TB-NCLATURE en fonction de ces trames.
Cette deuxième fonction peut être assurée par deux tâches :
- une tâche G-FIP-EMET qui s'occupe des émissions des trames,
- et une tâche G-FIP-RECEPT qui s'occupe des réceptions des trames.
Comme pour le SGBDIL/MTR les nouvelles valeurs des données doivent être communiquées au module de synchronisation (ie F-SYNCHRONISEUR) afin de détecter les évènements liés aux données. C'est la tâche G-FIP-RECEPT qui communique à F-SYNCHRONISEUR les changements de données lorsqu'elle reçoit une trame et ce, en utilisant les tables TB-FOBJET et TB-NCLATURE.



Module F-GERDONNEE

D) Module F-ACCESS-FIP

Ce module gère le support physique (il dépend de ce support).

Il s'occupe de la diffusion des trames proposées par le module F-GERDONNEE et détecte la réception des trames (ie le retour des trames qu'il a déjà émises) et les transmet au module F-GERDONNEE.

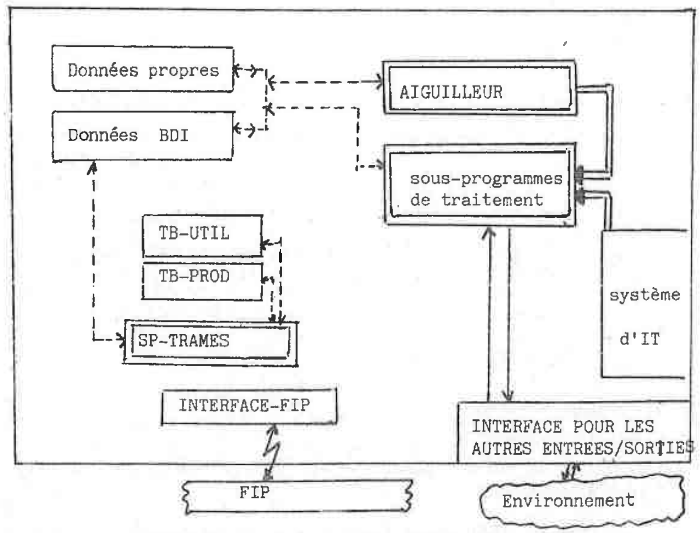
III.4 ARCHITECTURE D'UN SGBDIL/MAP

Un SGBDIL/MAP correspond au logiciel de gestion d'un automate progammable. Nous nous intéressons ici aux automates qui ne peuvent pas supporter d'exécutifs multitâches.

Les traitements effectués par l'automate sont matérialisés par des sous-programmes (calcul, entrées/sorties industrielles, gestion d'interruptions...).

L'activation de ces traitements se fait :
- par des tests continuels (attente active),
- après des temporisations,
- ou sur détection d'interruptions.

L'architecture proposée, pour la gestion d'un automate programmable, est la suivante :
- un ensemble de sous-programmes correspondant aux traitements tels que les entrées/sorties et les calculs ;
- un sous-programme spécial appelé AIGUILLEUR qui permet de connaître le sous-programme à activer à un instant donné (il s'occupe de sous-programmes non liés aux interruptions) ;
- un sous-programme de traitement de l'émission et de la réception des trames : SP-TRAMES ;
- un ensemble de données :
  . des données propres aux diverses opérations, (ie les données internes)
  . des données matérialisant des données de la BDI,
- deux tables qui contiennent la liste des objets FIP manipulés :
  . une table TB-PROD qui indique la liste des objets produits par le site,
  . une table TB-UTIL qui indique la liste des objets utilisés par le site.



architecture générale d'un SGBDIL/MAP

Gestion des trames

Lorsque le sous-programme SP-TRAMES est réveillé, par l'interface physique de gestion du support FIP, il analyse la trame comme suit :
- si la nomenclature associée à la trame correspond à un objet produit par l'automate, il dépose la valeur de cet objet dans la trame et demande son émission. L'utilisation de la tâche TB-PROD lui permet de retrouver la valeur de l'objet.

Numéro de nomenclature	Adresse de l'objet

TB-PROD

- si la nomenlature associée à la trame correspond à un objet utilisé par l'automate, il prélève la valeur et libère la trame.

La valeur prélevée est déposée dans la table TB-UTIL.

Numéro de nomenclature	Valeur

TB-UTIL

Sous programmes de traitement

Les sous-programmes de traitement correspondent aux actions implantées sur l'automate programmable.

Remarque

Les capteurs et actionneurs connectés à FIP sont considérés comme des automates élémentaires.

Un capteur connecté à FIP est chargé d'acquérir une mesure donnée et de la mettre sur FIP à la demande du gestionnaire FIP. Il permet donc d'implanter une action d'entrée et de produire une donnée.

Un actionneur connecté à FIP permet de recevoir un ordre (il utilise donc une donnée) et de l'appliquer au procédé (il réalise donc une opération de sortie).

Ces deux types d'automates sont gérés par des SGBDIL élémentaires constitués d'une action et d'une donnée chacun.

III.5 AUTRES CAS DE SGBDIL

Les trois types de SGBDIL que nous venons de présenter constituent les SGBDIL de base, les autres types peuvent être construits en combinant ces types de base.

1) Cas d'un SGBDIL/MTR-FIP

Dans le cas où une machine temps réel est connectée à FIP, il faut rajouter au SGBDIL/MTR un module pour le dialogue avec FIP (acquisition et émission de trames voir un SGBDIL/MAP).

2) Cas d'un SGBDIL/GFIP-RL

Si, en plus de la gestion du support FIP, une machine peut gérer des actions, son SGBDIL est obtenu en rajoutant au SGBDIL/GFIP l'aspect gestion des actions (tables et tâches) comme pour un SGBDIL/MTR et en changeant le module SYNCHRONISEUR de manière à ce qu'il puisse indiquer à la fois les activations des actions et les demandes d'émission des trames.

IV IMPLANTATION DES OBJETS ET CONSTRUCTION DU SGBDIR

Pendant l'étape de répartition, l'implanteur a choisi, pour chaque machine utilisée, l'ensemble des objets que cette machine doit supporter ; il ne lui reste maintenant que l'implantation de ces objets en fonction des spécificités de cette machine.

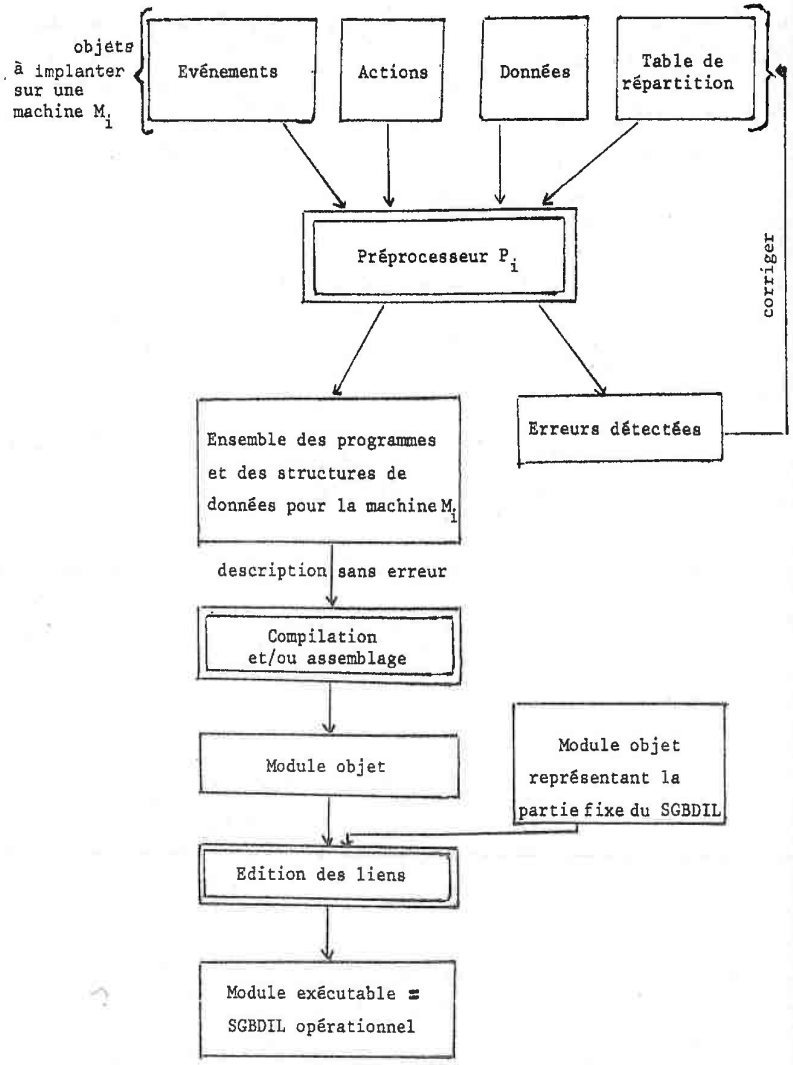
Il devra transformer la description des objets obtenus pendant la phase de conception pour l'adapter aux machines cibles.

A chaque machine cible Mi correspond une fonction de transformation Fi qui tient compte du type du SGBDIL qui gère cette machine.

Nous appelons préprocesseur Pi, l'automatisation de la fonction de transformation Fi.

La fonction de transformation permet :
- de générer les tâches et les structures de données nécessaires à la gestion des objets implantés sur la machine en fonction du type du SGBDIL à construire,
- et d'effectuer des contrôles statiques sur les objets à implanter (ie contrôler si les règles de description ont été respectées ou non).

Nous pouvons résumer le travail effectué à ce niveau de fabrication d'une ACPI par la figure ci-dessous :



La transformation des objets permet de déterminer les programmes et les structures de données à utiliser. Si la transformation obtenue est exempte d'erreur (des erreurs telles que les données non utilisées, les actions sans entrée ou sans sortie...peuvent être détectées par le préprocesseur) elle est utilisée pour produire le SGBDIL final. Par "partie fixe du SGBDIL" nous entendons l'ensemble des tâches, des sous-programmes et des tables qui sont directement liées au type du SGBDIL et pouvant être fixés indépendamment de l'ensemble des objets que gère le SGBDIL.

Le travail nécessaire pour définir les transformations et leurs résultats est similaire à celui que nous avons développé dans /MAMMERI, 82/ pour la transformation de structures écrites dans le langage FLEXI (/ZAKARI, 84/ ; /DERNIAME, 82C,83/) en programmes écrits en PASCAL.

Résumé

En implantant l'ensemble des objets, l'implanteur aura construit complètement le SGBDIR.

Le processus de fabrication d'une ACPI s'arrête à ce niveau, il ne reste plus qu'à exploiter, maintenir et étendre, en fonction des besoins, cette ACPI.

V CONCLUSION

Nous venons de présenter de manière générale une architecture pour l'implantation des systèmes de commande des procédés industriels géographiquement répartis.

Cette architecture présente les avantages suivants :

- elle tient compte de la hiérarchisation des installations de commande; ce qui permet de respecter les contraintes de temps et de sûreté de fonctionnement ; l'utilisation du support FIP permet d'accroître la cohérence de la BDI et la disponibilité de l'ACPI ;

- les problèmes liés au partage d'objets (ie les problèmes de blocage et de perte d'opérations rencontrés dans le cas des BDI /GARDARIN, 78/ à cause des accès simultanés aux objets) sont résolus à partir du moment où toutes les demandes d'accès sont adressées à un même gestionnaire d'objets pour l'ensemble des entités concurrentes d'un site donné ;

- le système de commande est défini à l'aide d'un ensemble de sous-systèmes (appelés SGBDIL) sans contrôle centralisé : la construction d'un tel système tient compte de la répartition géographique des procédés ;

- et enfin l'architecture proposée est décrite de manière modulaire ce qui permet de la raffiner aisément pour aboutir à la structure détaillée de l'ACPI (ou du SGBDIR) qu'on peut directement implanter sur des machines cibles.

Nous faisons remarquer que l'approche développée, ici, pour l'implantation des BDI diffère de celle connue dans le cas des BDC : les données sont stockées de manières diverses et non sous forme d'une vraie base de données.

Les données de la BDI peuvent être stockées en mémoire centrale (sous forme de table) ou en mémoire secondaire (sous forme de fichiers). Le choix du support pour le stockage d'une donnée dépend de l'utilisation de cette donnée (nous n'avons fait aucune hypothèse quant à ce choix).

Signalons enfin, que les systèmes de gestion de BDC peuvent être retenus pour la gestion des données relatives à l'historique du procédé, de sa commande et du personnel, étant donné que ces données peuvent atteindre des quantités importantes et ne sont pas, généralement, utilisées en temps réel.
Ainsi un SGBDIR peut avoir comme composante un SGBD classique.

PARTIE III

Les divers travaux relatifs à la commande de procédés industriels conduisent à la définition de systèmes en temps réel, généralement, répartis.

L'expérimentation de tels systèmes doit passer inévitablement par l'utilisation d'un ensemble de mécanismes de base relatifs :
- à la synchronisation des entités coopérantes ;
- et à la communication, entre ces mêmes entités, via un réseau de communication.

Mais ces mécanismes sont communs et, généralement, indépendants des approches développées (évènements, base de données industrielles, communication, etc...). Il est alors important de dégager un noyau commun à tous ces systèmes pour faciliter leur implantation : en disposant des mécanismes de base (portables par leurs spécificités) l'implanteur s'intéresse essentiellment à l'implantation des concepts spécifiques au système à implanter.

C'est à ce besoin (ie la définition d'un noyau de base) que nous répondons dans cette troisième partie de notre thèse.

Nous allons présenter une maquette réalisée à l'aide d'un ensemble de microordinateurs reliés via un réseau local.

Cette maquette offre un ensemble d'outils opératoires relatifs à :
- la synchronisation entre tâches temps réel
- la communication entre tâches distantes
- et la gestion du temps.

Lesquels outils constituent un support de base pour la construction des systèmes temps réel et répartis.

L'utilisation de cette maquette contribue à la validation des idées proposées pour la structuration/implantation des ACPI au sein de l'équipe parallélisme du CRIN à laquelle nous apporterons (ie implantation des BDI, du langage Flexi /ZAKARI,84/ et des évènements /BENMAIZA,84/).

Les services offerts par les diverses composantes de la maquette sont présentés dans le chapitre IX, leur implantation et le mode de leur utilisation sont donnés dans les annexes B1, B2, B3 et B4 (une annexe par composante).

CHAPITRE VIII

UN RESEAU DE MICROORDINATEURS POUR LA CONSTRUCTION DES SYSTEMES TEMPS REEL ET REPARTIS

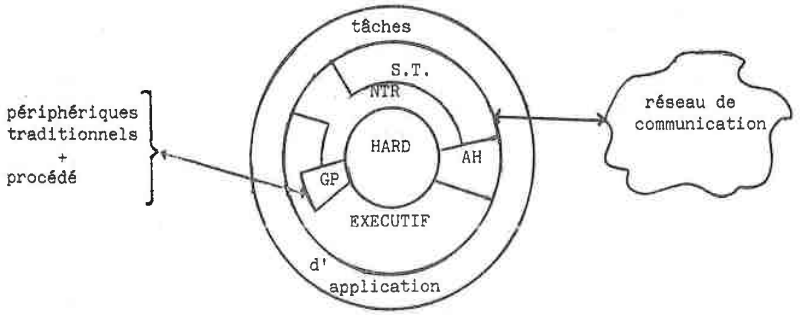
I INTRODUCTION

Un système informatique réparti est constitué de plusieurs sites interconnectés via un réseau de communication. Chacun de ces sites possède son propre exécutif et contribue à un objectif global en supportant une partie de l'application. Cette application est constituée d'un ensemble de programmes (ou tâches ou processus ou entités) répartis sur les différents sites du réseau et réalise une fonction ayant une signification pour l'utilisateur (par exemple commande d'un laminoir).

Dans le cas d'une application en commande de procédés, les exécutifs utilisés doivent assurer essentiellement :
- la synchronisation des entités coopérantes (activation sur détection d'évènements, temporisations..) pour mener à bien leur travail ;
- la communication entre entités locales ou distantes ;
- et la gestion des ressources (mémoire, entrées/sorties industrielles ou non,...).

Pour répondre à ces trois besoins, chacun des sites supportant l'application devra posséder un exécutif ayant, au moins,les composantes suivantes :
- le calculateur et ses périphériques (par périphériques nous entendons les périphériques traditionnels (imprimante,...) et les dispositifs de dialogue avec les procédés) ;
- un noyau temps réel pour assurer la synchronisation des entités coopérantes et la gestion des ressources de la machine ;
- une connexion au réseau de communication et un logiciel de gestion de cette connexion que nous appellerons "station de transport" ;
- un logiciel que nous appellerons "agence horlogerie" (au sens SCEPTRE /BNI, 80,82/) qui permet aux entités coopérantes d'effectuer des opérations relatives au temps (temporisation, datation de messages, ...);
- un logiciel de gestion des entrées/sorties (industrielles et classiques);
- et des composantes classiques (compilateur, systèmes de gestion de fichiers, ...) qui sont nécessaires.

En tenant compte de ces composantes le "squelette" d'un site réparti peut être illustré par la figure suivante :



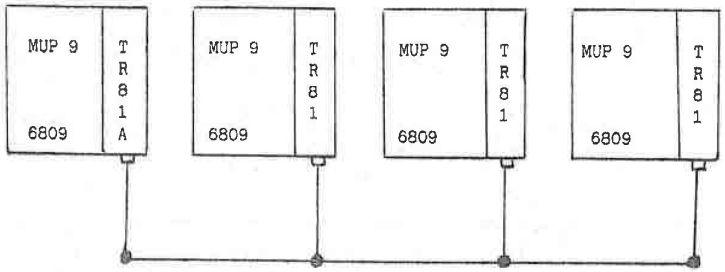
Notation :
ST : station de transport
NTR : noyau temps réel
AH : agence horlogerie
GP : gestion de périphériques

Parmi les composantes évoquées précédemment nous avons étudié et réalisé une maquette /MAMMERI, 84 B/ autour du réseau local GIXINET /GIXI,81/ qui offre des services relatifs à :
- la synchronisation des tâches
- la connexion d'un site au réseau local GIXINET
- la gestion du temps
- et la communication entre tâches distantes.

Ces services constituent un support de base pour l'implantation des systèmes temps réel et répartis.

Architecture générale de la maquette

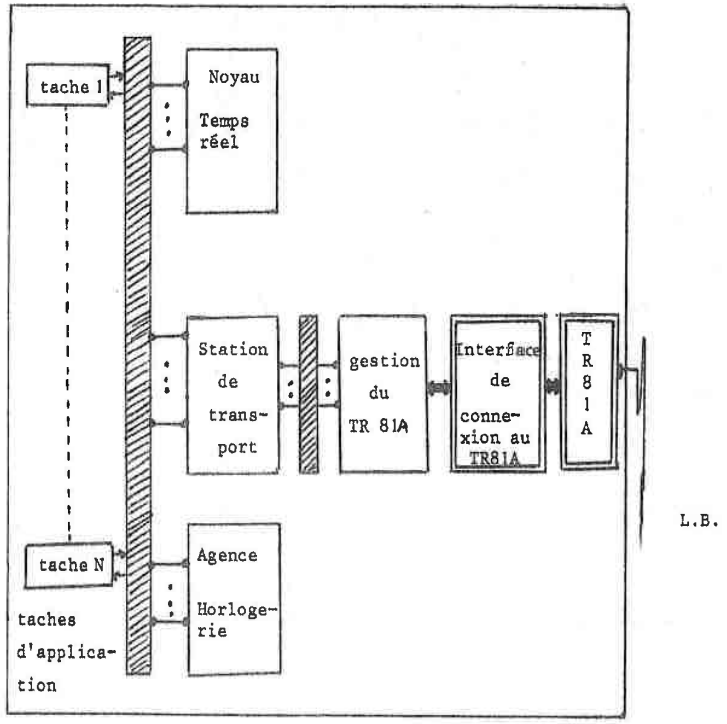
La maquette réalisée est composée de 4 stations constituées chacune d'un microordinateur à base de 6809 de MOTOROLA relié au réseau local GIXINET via le communicateur TR81A de GIXINET.



Ligne bifilaire

L'architecture d'une station est définie par :
- les modules logiciels suivants :
  . un noyau temps réel
  . une agence horlogerie
  . une station de transport
  . un module de gestion du communicateur.
- et les modules matériels suivants :
  . la carte du microordinateur MPU9 /EFCIS/ contenant le microprocesseur 6809 et une interface de connexion au communicateur ;
- et le communicateur TR 81 A de GIXINET.

Cette architecture est illustrée par la figure ci-après :



Struture d'une station

Notation

: module logiciel

: module matériel

: interface d'accès (définie par des procédures)

: accès à un module matériel

L.B. : ligne bifilaire

Remarque

Comme les composantes réalisées sont des composantes de base, elles sont indépendantes entre elles (sauf pour la ST et la connexion à GIXINET).

II LA SYNCHRONISATION : LE NOYAU TEMPS REEl SCEPTRE

II.1 INTRODUCTION

Une application en temps réel est un système de traitement de l'information ayant pour mission la commande d'un phénomène physique en respectant des contraintes de temps /BROWAEYS, 84/. Cette application est composée de programmes s'exécutant sur une (ou plusieurs) machine(s) gérée (s) par un (ou plusieurs) exécutif(s).

Toute exécution d'un programme est mise au compte d'agents actifs appelés tâches (ou processus au sens de /CROCUS, 75/).

Ces tâches coopérent pour la réalisation d'un objectif global commun qui est la commande du phénomène physique.

Par coopération des tâches nous entendons synchronisation et communication inter-tâches.

Nous nous intéressons, ici, à l'aspect synchronisation/communication entre tâches situées sur un même site (il s'agit de la synchronisation et communition locales).

Beaucoup de mécanismes ont été proposés dans la littérature /ANDLER 78/ pour l'expression de la coopération des tâches ou des processus concurrents. Nous noterons particulièrement pour la synchronisation de processus les travaux de DIJKSTRA (pour les sémaphores , de HOARE (pour les moniteurs /HOARE, 74/) d'HABERMANN et CAMPBELL (pour les expressions de chemin /CHAMPBELL, 74/), de ROBERT et VERJUS (pour les modules de contrôle /ROBERT, 77/) et de REED et CANODIA (pour les compteurs d'évènements /REED, 79/).

Pour ce qui est de la communication inter-processus, les concepts qui attirent le plus, les concepteurs des systèmes et des langages temps réel sont, depuis longtemps, la notion de boîte à lettres /CROCUS, 75/ et plus récemment la notion de port de communication /SILBERSCHATZ, 81/, / MAO, 80/. Le choix des mécanismes de coopération entre tâches (moniteurs ou sémaphores ?... Boîte à lettre ou port ? ...) et de l'implantation de ces mécanismes influe considérablement sur le temps de réponse (ie sur l'over-head) de l'exécutif et sur sa maintenance.

Ainsi avant d'opter pour le choix de tels ou tels autres mécanismes de coopération il est important de dégager les mécanismes élémentaires indispensables à la contruction de tout exécutif temps réel pour faciliter:
- la construction modulaire (et saine dès la base) des exécutifs,
- et la portabilité des applications et des exécutifs réalisés.

Le projet SCEPTRE /BNI, 80.82/ propose une solution à ce problème (ie la recherche des mécanismes élémentaires).

En effet, en se placant à un niveau proche des mécanismes offerts par les machines temps réel, le projet SCEPTRE (standard du coeur des exécutifs des produits temps réel européens) a permis de définir un ensemble d'objets et d'opérations élémentaires permettant de construire de manière structurée et portable les mécanismes de coopération entre activités parallèles d'un exécutif temps réel.

C'est ainsi que notre choix a été porté sur le noyau SCEPTRE pour construire le noyau temps réel nécessaire à notre maquette.

Les services offerts par le noyau SCEPTRE seront présentés dans le prochain paragraphe, leur implantation et le mode de leur utilisation sont donnés dans l'annexe B1.

II.2 PRESENTATION DU NOYAU SCEPTRE

Nous rappelons, ici, brièvement les objets et opérations SCEPTRE, ce qui permettra de comprendre les possibilités, mais aussi, les limites, de la maquette réalisée ; on trouvera une étude détaillée du projet SCEPTRE dans /BNI, 80,82/.

1. Définition des concepts de base

a) Notion de machine
Une machine est un environnement matériel assurant l'exécution de processus cablés, microprogrammés ou programmés. Une machine comporte une (ou plusieurs) famille(s) de processeur(s) paratgeant une mémoire commune.

b) Notion de tâche
Une tâche est un agent actif responsable de l'exécution, par une machine, d'un programme composé à partir du répertoire d'instructions de cette machine. Une tâche correspond à la notion de processus /CROCUS,75 /.

c) Notion d'ordonnanceur
On appelle ordonnanceur un module chargé de gérer un ensemble de processus pour le compte d'une famille de tâches. Cette gestion consiste à allouer les processeurs aux tâches selon un algorithme de choix spécifique (un exemple de choix est la priorité des tâches).

d) Notion d'agence
L'agence est l'unité fonctionnelle de construction d'un exécutif temps-réel. Une agence est construite à partir de tâches, de procédures et de fonctions, de données locales ou globales, etc...

Une agence offre un ensemble de services au travers d'une collection de procédures et de types d'objets (comme exemples d'agences on peut citer l'agence de gestion des fichiers, l'agence de gestion de la mémoire, l'agence de gestion du temps, etc...).

La représentation interne de ces procédures et de ces types n'est connue qu'à l'intérieur de l'agence (principe des types abstraits /DERNIAME, 79/); ce qui facilite la construction progressive et modulaire des applications.

2. Les services SCEPTRE

Parmi les services que doit offrir un exécutif temps réel /CLOUT, 82/ le projet SCEPTRE s'intéresse uniquement à la gestion et la synchronisation de tâches et à la communication entre tâches. Les autres fonctions de l'exécutif sont confiées à des agences spécialisées (gestion de la mémoire, gestion du temps, gestion des fichiers, etc...).

Pour assurer la gestion des tâches et la synchronisation et la communication entre tâches, le projet SCEPTRE a introduit quatre types d'objets élémentaires :
- le type TACHE : un objet TACHE désigne une tâche gérée par le noyau,
- le type EVENEMENT : un objet de type EVENEMENT mémorise le signal marquant qu'une condition est devenue vraie ;

- le type REGION : un objet de type REGION contient une marque indiquant si un objet partagé est possédé, ou non, par une tâche; il permet d'assurer l'exclusion mutuelle /CROCUS, 75/ entre tâches.
- le type FILE : un objet de type FILE range des informations et les restitue suivant la politique "premier entré, premier sorti" ; il constitue un moyen élémentaire de communication inter-tâches.

A chacun de ces 4 types est associé un ensemble d'opérations qui permettent la manipulation des objets de ce type :

a) Gestion des tâches
La manipulation d'une tâche se fait par l'intermédiaire de son descripteur (ie le nom qui lui est donné par le noyau au moment de sa création) et s'effectue par appel à l'une des opérations du tableau suivant :

OPERATION	PARAMETRES	FONCTION
Démarrer	tâche	Lancer l'exécution de la tâche
Arrêter	tâche	Arrêter l'exécution de la tâche
Continuer	tâche	Continuer l'exécution de la tâche
Se terminer		Terminer l'exécution de la tâche qui l'exécute
Changer-priorité	tâche, nouvelle priorité	Donner à la tâche, la nouvelle priorité
Etat	tâche	Fournir l'état de la tâche
Priorité	tâche	Fournir la priorité de la tâche
Tâche courante		Fournir l'identité de la tâche qui l'exécute
Statut		Fournir le statut de l'opération élémentaire qui vient d'être exécutée par la tâche

b) Gestion des évènements

Principe
La signalisation entre deux tâches nécessite une opération dans chacune des deux tâches : attente de signal dans l'une et envoi de signal dans l'autre.

La signalisation est donc symétrique par nature. Une tâche doit donc attendre successivement, à des instants différents de son exécution, des évènements lui appartenant. Un évènement représente chaque cause possible de continuation d'une tâche lors de ses opérations d'attente explicite.

Dans le noyau SCEPTRE un évènement est propre à une seule tâche; il est "privé."

Opérations
un évènement est un objet qui peut être dans l'un des deux états ARRIVÉ ou non ARRIVÉ. Il est géré par les opérations du tableau suivant :

OPERATIONS	PARAMETRES	FONCTION
Signaler	évènement, tâche	Signaler à la tâche que l'évènement est arrivé
Attendre	Liste d'évènements	Attendre que l'un (au moins) des évènements de la liste soit arrivé
Arrive	Liste d'évènements	Prédicat qui est vrai si tous les évènements de la liste sont arrivés
Effacer	Liste d'évènements	Remettre les évènements de la liste dans l'état non arrivé

c) Gestion des régions

Principe
Le parallèlisme des tâches induit un phénomène de compétition pour la manipulation des objets partagés. Pour assurer la cohérence de ces manipulations, il est nécessaire d'utiliser un mécanisme d'exclusion mutuelle /CROCUS, 75/.

Le mécanisme d'exclusion mutuelle proposé par le noyau SCEPTRE est fondé sur le concept de REGION.

On associe une région à tout objet qu'on veut protéger.

Opérations
Un objet de type REGION peut être dans l'un des deux états OCCUPÉ ou LIBRE. Il est manipulé par les deux opérations suivantes :

OPERATION	PARAMETRES	FONCTION
Entrer	Région	Demander la propriété exclusive de la région
Sortir	Région	Relâcher la propriété exclusive de la région

d) Gestion des files

Principe

La stratégie la plus couramment utilisée pour la communication entre tâches est la structure de file. Le noyau SCEPTRE propose un ensemble d'opérations élémentaires pour la gestion de files à stratégie FIFO. Les files constituent un mécanisme de base pour la communication, tout mécanisme de communication complexe peut être construit à partir de ces files.

Opérations

Une file est gérée par les opérations suivantes :

OPERATION	PARAMETRES	FONCTION
Envoyer	élément,file	Envoyer l'élément,à la queue,dans la file
Retirer	élément,file	Retirer le premier élément de la file
Vide	file	Prédicat qui est vrai si la file est vide
Pleine	file	Prédicat qui est vrai si la file est pleine

e) Propriétés communes aux opérations du noyau

- Propriété d'exclusivité : les opérations du noyau partageant un objet identique s'excluent mutuellement ;
- propriété d'atomicité (ou d'indivisi bilité) : les opérations du noyau sont atomiques (ou indivisibles) ;
- sécurité des opérations : les opérations du noyau peuvent donner lieu à des cas d'erreurs.Ceux-ci sont alors mémorisés dans un compte rendu global et communiqués à la tâche appelante via l'opération STATUT.

II.3 CONCLUSION

Si la normalisation des principaux langages classiques permet aujourd'hui, en programmation séquentielle, de respecter certains critères de portabilité, le temps réel est perçu encore comme un domaine qui entraine le recours à des solutions spécifiques. La diversité des mécanismes de base pour la coopération des tâches rend difficile, voir impossible, la portabilité des exécutifs temps réel.

Même si le noyau SCEPTRE n'est pas portable en lui-même (il faut le réécrire pour chaque machine) il est à la base de la portabilité des exécutifs temps réel. De plus le noyau SCEPTRE est quasi-minimal,ce qui permet de réduire au maximum les modifications en cas de transport d'exécutifs ; les programmes qui se limitent à l'utilisation des primitives du noyau SCEPTRE sont portables sur toutes les machines qui disposent de ce noyau.

Tous les objets et opérations SCEPTRE ont été réalisés. L'essentiel de cette réalisation est donné dans l'annexe B1. Le manuel d'utilisation du noyau réalisé est donné dans /MAMMERI, 84 A/.

Grâce aux propriétés de SCEPTRE le noyau réalisé se prête bien à l'extension et à l'enrichissemnt de la maquette que nous avons réalisée.

III GESTION DU TEMPS : AGENCE HORLOGERIE

III.1 INTRODUCTION

Une application temps réel est,avant tout, fonction du temps. La notion de temps apparaît dans ce qu'on appelle "temps de réponse à un stimulus(revoir éventuellement les caractéristiques des ACPI dans le chapitre I).

Nous avons vu dans le chapitre IV que l'activation des tâches peut faire intervenir :
- des temporisations ;
- et des instants précis.

Mais aussi, nous avons signalé que la synchronisation dans les systèmes répartis fait intervenir la datation /LAMPORT, 78/ des informations communiquées entre les divers sites d'un système réparti.

Pour répondre à ces deux besoins (ie l'activation des tâches et la datation des informations) tout exécutif temps réel doit posséder des mécanismes de gestion du temps.

L'ensemble des opérations relatives à la gestion du temps constitue ce que nous appelons "agence horlogerie" (pour reprendre le vocabulaire SCEPTRE).

III.2 SERVICES OFFERTS PAR L'AGENCE HORLOGERIE

L'agence horlogerie est une collection de tâches et de procédures exécutant des opérations relatives au temps.

Deux fonctions sont réalisées par cette agence :
- gestion du temps absolu ;
- gestion des délais et des temporisations.

1) Gestion du temps absolu

La gestion du temps absolu consiste à :
- initialiser une horloge locale matérialisée par des compteurs (heure, minute, seconde,...) ; la précision de l'horloge étant celle demandée par l'utilisateur ;
- faire évoluer les compteurs de l'horloge locale à chaque interruption émanant de l'horloge physique de la machine ;
- et fournir, à une tâche, les valeurs des compteurs de l'horloge locale si celle-ci les demande.

A ce niveau la gestion du temps est réalisée par :
- une opération INIHOR qui permet d'initialiser l'horloge locale ;
- une opération HORLOC qui permet de fournir, à une tâche, l'heure locale ;
- et d'une tâche immédiate H,activée sur interruption de l'horloge physique, qui permet de faire évoluer les compteurs de l'horloge locale.

2) Gestion des délais

La gestion des délais consiste à :
- donner la possibilité,à une tâche,de demander ou d'annuler un délai ;
- d'avertir une tâche qui a demandé un délai lorsque ce délai vient à échéance, en lui signalant l'évènement (au sens SCEPTRE) qu'elle a précisé dans sa demande de délai ;
- et de gérer les compteurs associés aux demandes de délai. Ils sont decrémentés par la tâche immédiate H.

Pour gérer les délais, l'agence horlogerie offre aux tâches utilisateur les opérations suivantes :
- DELAI : pour demander un délai ;
- ANULTM : pour annuler un délai déjà demandé ;
- LIRETM : pour lire le temps restant à l'expiration d'un délai.

3 Gestion des temporisations

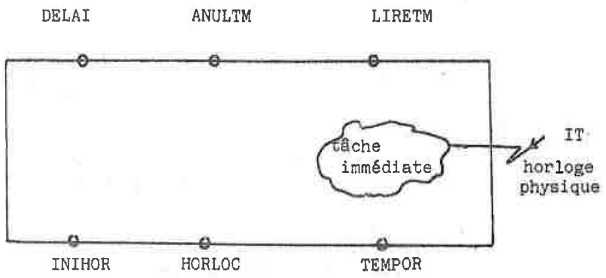
Les temporisations diffèrent, selon la définition que nous leur donnons ici, des délais par le fait qu'une tâche qui demande une temporisation est bloquée pendant le temps qu'elle a précisé dans sa demande et ne peut continuer son exécution que si ce temps est écoulé alors qu'une demande de délai ne bloque pas la tâche qui a fait la demande.

Les tâches effectuent leurs demandes de temporisation en appelant une opération appelée TEMPOR qui traite les demandes de temporisation.

Nous avons séparé délai et temporisation car une tâche a parfois besoin de se bloquer jusqu'à une certaine date (ou pendant une certaine durée) avant de continuer son travail et, parfois, de changer son comportement après écoulement d'un temps qu'elle a fixé (tout en restant active jusqu'à l'écoulement de ce temps).

III.3 STRUCTURE DE L'AGENCE HORLOGERIE

L'agence horlogerie est constituée, comme le montre la figure ci-après, d'une tâche immédiate H et 6 procédures.



Architecture générale de l'agence horlogerie

L'agence horlogerie n'est connue, des tâches utilisateur, qu'à travers l'interface composée des 6 opérations (DELAI, ANULTM, LIRETM, INIHOR, HORLOC, TEMPOR) utilisées comme suit :

OPERATION	PARAMETRE en ENTREE	PARAMETRE en SORTIE	FONCTION
Delai	période, evt		demander un délai
AnultM			annuler un délai
LiretM		période	lire le temps restant à un délai
Inihor	heure		initialiser l'horloge locale
Horloc		heure	lire l'horloge locale
Tempor	période		demander une temporisation

Période : désigne un nombre de fractions de seconde ;
Heure : désigne l'heure, la minute, la seconde et la fraction de seconde;
evt : désigne un n° d'évènement à signaler après expiration du délai.

La tâche immédiate permet, sur interruption associée à l'horloge physique :
- d'incrémenter les compteurs associés à l'horloge locale ;
- de décrémenter les compteurs associés aux demandes de délais et de temporisations. Lorsqu'un compteur s'annule la tâche immédiate H permet :
  . soit de signaler l'évènement associé à ce compteur si celui-ci est associé à une demande de délai ;
  . soit d'activer la tâche correspondant à ce compteur si celui-ci est associé à une demande de temporisation.

III.4 CONCLUSION

Nous venons de présenter des primitives de base pour la gestion du temps. Ces primitives sont nécessaires à la construction d'un exécutif temps réel. L'implantation de l'agence horlogerie et le mode d'utilisation de ses primitives sont donnés dans l'annexe B2.

IV LA CONNEXION AU RESEAU LOCAL GIXINET

Pour assurer la communication entre entités distantes gérées par des exécutifs géographiquement distants, il est nécessaire, pour ces exécutifs, de disposer de moyens leur permettant de s'échanger des informations (ie émettre et recevoir des messages).

La première composante à mettre en oeuvre, pour fournir ces moyens, réside au niveau de la connexion des divers sites à un réseau de communication. C'est à cette connexion que nous nous intéressons dans ce paragraphe.

De nos jours, il existe deux grandes familles de réseaux de communication :
- les réseaux à longue distance tels que TRANSPAC et ARPANET /LORRAINS, 79/, /MACCHI, 79/ ;
- et les réseaux locaux dont l'étendue géographique est limitée à quelques kilomètres /GIEN, 79/, /REED,78/, /NAFFAH, 81/ , /POWELL, 81/.

Pour une maquette expérimentale, le choix du réseau de communication a été fait parmi les réseaux locaux vues les qualités qu'offrent ce type de réseaux (prix, souplesse,...).

Le réseau expérimental qui a été réalisé /MAMMERI, 84 B/ est constitué de microordinateurs reliés à un réseau local de type GIXINET /GIXI,81/.

Dans ce paragraphe nous allons présenter brièvement :
- le concept de réseaux locaux, dans un contexte général,
- les caractéristiques du réseau GIXINET,
- et les services offerts par le réseau local GIXINET aux couches supérieures (pour notre maquette ces services sont utilisés par la station de transport -cf paragraphe V-).

IV.1 INTRODUCTION AUX RESEAUX LOCAUX

1) Utilisation et définition des réseaux locaux :

Les réseaux locaux connaissent aujourd'hui un grand succès auprès des utilisateurs de l'informatique. Ce succès est dû aux caractéristiques internes des réseaux locaux (un seul câble, contrôle décentralisé, diffusion,...) et au coût des circuits gérant la fonction d'accès, très réduit (cette réduction est due au développement, sans cesse, de la microélectronique et des micro-processeurs).

Nombreuses sont les utilisations des réseaux locaux. Dans les applications générales, nous pouvons citer la bureautique (messagerie, production de documents), l'informatique (base de données, partage de ressources, gestion) et la communication en temps réel des images et de la parole /NAFFAH, 80/.

Dans les applications spécialisées, nous avons à considérer essentiellement les applications industrielles (contrôle de chaînes de montage, systèmes de mesure et de commande) et les applications militaires (communication radio).

Parmi les réseaux locaux développés de nos jours, nous mentionnons :
ETHERNET /METCAFLE,76/ , PROWAY /AUGER,82/ , DANUBE /NAFFAH,80/ , FACTOR /APSIS,83/ , GIXINET /GIXI,81/ , GAMMANET /EBEHIRA,83/ etc...

Pour le définir, un réseau local est un moyen de transport d'information entre équipements dans un même site. Les dimensions d'un réseau local peuvent varier entre une dizaine de mètres et quelques kilomètres (ex : un immeuble, une université, une usine...).
Pour les organisations géographiquement décentralisées (raffineries, mines etc...), il est nécessaire d'avoir des réseaux locaux interconnectés.

2) Caractéristiques des réseaux locaux

Les principales caractéristiques des réseaux locaux résident dans les points suivants :
- une large bande passante est exigée ;
- un contrôle décentralisé ;
- possibilité de reconfiguration (ajoût et retrait de stations) ;
- fiabilité de la transmission ;
- possibilité d'interconnexion de plusieurs réseaux locaux.

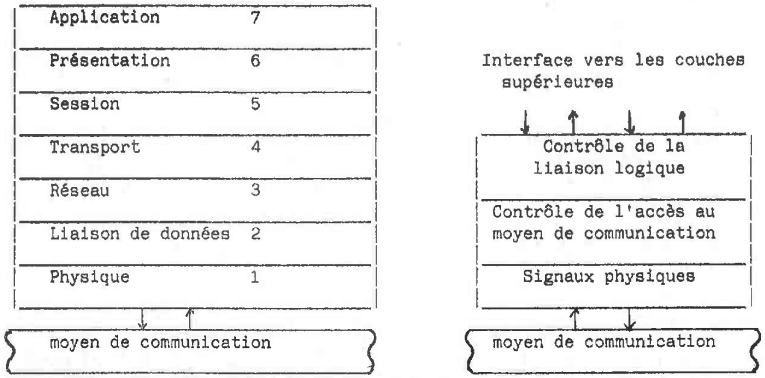
3) Les réseaux locaux et le modèle de référence de l'ISO

Rappelons que le sous comité ISO/TC 97/SC 16 -interconnexion des systèmes ouverts- a mis au point un modèle de référence pour l'interconnexion des sytèmes ouverts /ISO 81/. Le modèle ISO est structuré en sept couches (application, présentation, session, transport, réseau, liaison de données, liaison physique).

Les couches application, présentation, session, transport, réseau, ne sont pas du ressort des réseaux locaux.

Un ressort local est constitué de trois couches permettant l'implantation des couches liaison de données et liaison physique, ce sont :
- le contrôle de la liaison logique : couche qui initialise et gère le transfert, interprète les commandes émises par les couches supérieures et résoud le problème d'erreurs ;
- le contrôle de l'accès au moyen de communication, concerne la méthode d'accès (CSMA/CD, jeton ou autres).
- la signalisation physique gère la liaison physique (modulation/démodulation, synchronisation des signaux, etc...).

La figure ci-après illustre la correspondance entre le modèle de l'ISO et les réseaux locaux :

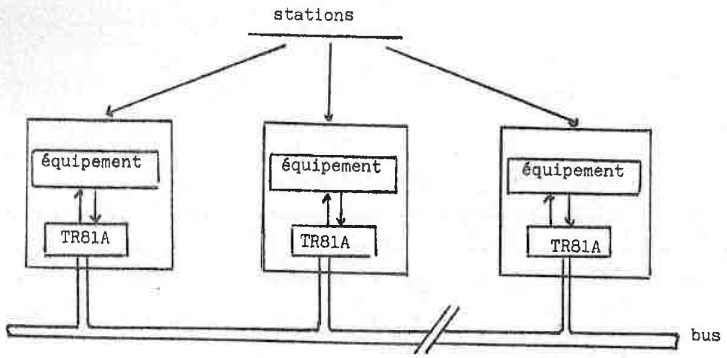


IV.2 PRESENTATION DU RESEAU LOCAL GIXINET

Pour gérer la connexion au réseau local GIXINET, pour comprendre les services offerts par cette connexion, il est important de présenter l'architecture générale et le principe de fonctionnement GIXINET (on trouvera une description détaillée du réseau GIXINET dans /GIXI, 81/).

Le réseau local GIXINET est un réseau de communication destiné à relier des matériels informatiques hétérogènes.

La structure du réseau comporte un support de communication organisé en bus sur lequel des équipements viennent se greffer à travers la carte du contrôleur TR 81 A.

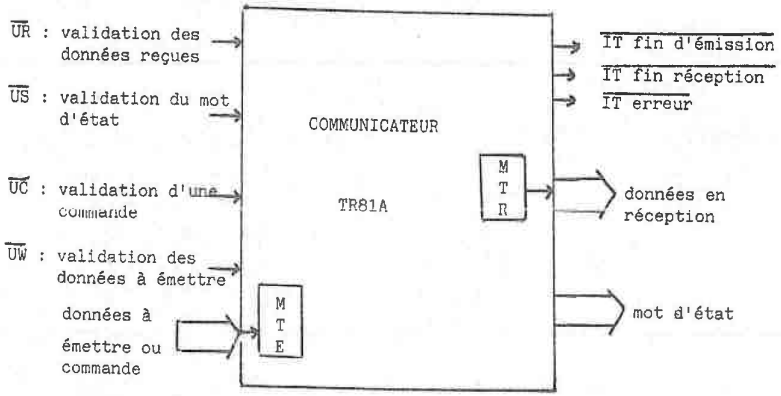


L'échange des données entre stations se fait via des mémoires tampons implantées sur les communicateurs TR 81 A.

Cet échange s'effectue selon l'algorithme TABLE-Ronde /NAFFAH, 78/ caractérisé par la circulation d'un jeton qui donne le droit, aux stations, d'émettre des informations sur le bus à une vitesse de 125 K bits/s.

Le dialogue entre l'usager et le communicateur est assuré par les organes suivants (voir figure ci-après) :
- une mémoire tampon d'émission (MTE) ;
- une mémoire tampon de réception (MTR) ;
- trois interruptions (fin d'émission, fin de réception, erreur) ;
- une logique qui règle les accès aux tampons et fournit à l'usager toutes les informations sur l'état du communicateur via un octet de mot d'état.

Cette logique reçoit et exécute les commandes de l'usager (demande de connexion au réseau, demande d'émission, demande de réception et demande de deconnexion du communicateur).



IV.3 GESTION DE LA CONNEXION GIXINET-MC 6809

La description de l'interface physique utilisée pour connecter un micro-ordinateur à base de MC 6809 de MOROTOLA est donnée dans l'annexe B3 ; nous nous limiterons ici, à la présentation des primitives qui permettent, à la station de transport, d'utiliser le réseau local GIXINET. Le mode d'utilisation de ces paramètres (ie le passage des paramètres) est donné dans l'annexe B3. Ces primitives sont :

1) Connexion logique du communicateur

L'opération OUVRGX permet à la station de transport (ST) de demander la connexion logique du communicateur au réseau, c'est-à-dire autoriser le communicateur à émettre et recevoir des messages.

C'est la première opération que l'utilisateur (la ST) doit appeler avant toute tentative d'émission et/ou de réception de messages...

En cas d'erreur irrécupérable (mauvais branchement au réseau,...(cf /GIXI, 81/)) le communicateur ne peut être "connecté logiquement au réseau" et l'essai de connexion échoue.

2) Deconnexion logique du communicateur

L'opérateur FERMGX permet, à la ST, de demander la deconnexion logique du communicateur, c'est-à-dire interdire au communicateur de recevoir et/ou d'émettre des messages, mais le communicateur reste connecté physiquement au réseau et joue un rôle de répéteur uniquement.

3) Obtention de l'état du communicateur

La ST demande l'état du communicateur (connecté, non connecté ou en état d'erreur) via l'opérateur ETATGX.

4) Emission de messages

La ST demande l'émission d'un message sur le réseau via l'opération EMETGX qui reçoit le message de la ST et assure sa transmission à la station distante concernée.

5) Obtention des résultats d'une émission

Dans les cas où :
- le communicateur distant, concerné par le message transmis, n'est pas connecté (physiquement ou logiquement) ou bien si la mémoire de réception de ce communicateur n'est pas libre ;
- il y a des erreurs de transmission (dues à des parasites quelconques) ;
- le communicateur local n'est pas connecté (physiquement ou logiquement) au réseau ou bien si sa mémoire d'émission est occupée (par le message précédent),

Le message à transmettre ne peut arriver à destination .

Pour connaître le résultat d'une émission (émission terminée ou impossible) la ST appelle l'opération RESMGX qui lui rend le résultat de l'émission. Lorsqu'une demande d'émission est complètement traitée (ie que le message a été transmis ou bien qu'il y a eu un échec (pour les raisons citées précédemment) la ST est avertie par une interruption (interruption de fin d'émission)).
Dans la ST doit exister un sous-programme, appelons le ITEMET, pour traiter cette interruption (dans ce sous-programme la ST appelle l'opération RESMGX pour connaître le résultat de l'émission).

6) Obtention des informations relatives à un message reçu

Lorsque le communicateur local reçoit un message en provenance d'un communicateur distant,il signale la réception de ce message à la ST et ce en activant un sous-programme de la ST qui traite les interruptions de fin de réception (ce sous-programme est appelé ITRECP).

Dans le sous-programme (ou S/P) ITRECP , la ST demande les informations relatives au message reçu (ie sa longueur, la station distante qui l'a émis et l'adresse où il est rangé) par l'intermédiaire de l'opération RECPGX.

7) Libération de la mémoire de réception

Après avoir prélevé le message reçu la ST libère la mémoire de réception du communicateur si elle veut l'autoriser à recevoir d'autres messages. La libération de la mémoire de réception se fait via l'opération LIBRGX.

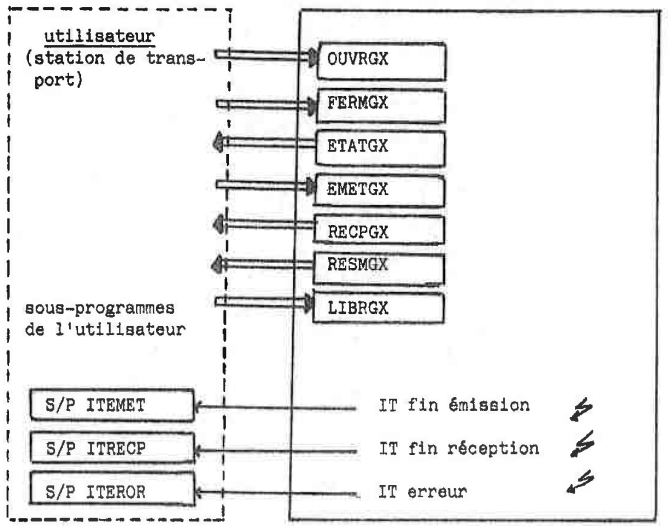
Remarques
1. En cas d'erreur irrécupérable du communicateur il y a génération d'une interruption erreur du communicateur ; pour traiter cette interruption la ST doit contenir un S/P qui traite les erreurs du communicateur (demander la déconnexion logique du communicateur, avertir l'opérateur,...) ce S/P est appelé ITEROR.
2. Les trois S/P ITEMET, ITRECP et ITEROR permettent un asynchronisme entre la ST et le communicateur.

IV.4 CONCLUSION

Avec la connexion au réseau local GIXINET nous offrons à l'utilisateur le moyen de communiquer avec des sites distants. Ce moyen de communication sera utilisé dans les prochain paragraphe pour construire une station de transport. Vis-à-vis de cette ST le module de gestion du communicateur offre les services de niveau 1 et 2 de l'ISO (le niveau 3,c'est-à-dire la couche réseau,n'est pas utilisé dans notre cas puisque le problème de routage ne se pose pas. Il faudrait réaliser le niveau 3 de l'ISO si on interconnectait plusieurs réseaux locaux).

Ce module est accessible à la ST via les 7 opérations décrites précédemment. La figure ci-après permet de résumer l'interaction qui existe entre la ST et le module de gestion du communicateur.

⟹ : désigne un appel sans demande de résultat
⟸ : désigne un appel avec demande de résultat



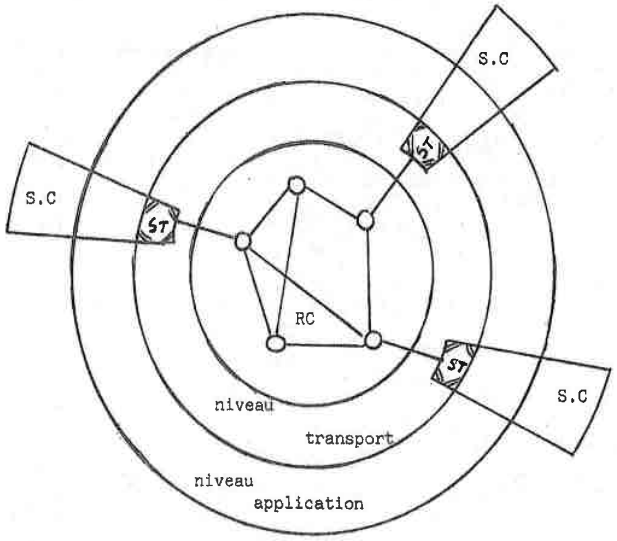
V LA STATION DE TRANSPORT

Nous venons de présenter la connexion d'un microordinateur au réseau local GIXINET, nous allons maintentant, en utilisant cette connexion, construire une "station de transport" ; nous nous limiterons ici aux grandes lignes de cette station de transport, les détails de réalisation et d'utilisation sont donnés dans l'annexe B4.

V.1 ROLE D'UNE STATION DE TRANSPORT DANS UN SYSTEME DISTRIBUE

Pour aborder le problème posé par l'accès à un réseau de communication on divise généralement /LORRAINS, 79/, /MACCHI, 79/ le réseau informatique en deux niveaux :
- le niveau transport qui constitue les outils nécessaires à la communication entre entités de traitement ;
- et le niveau traitement qui est spécifique à la forme d'utilisation du réseau.

Notation :
RC : réseau de communication
ST : station de transport
SC : site connecté au RC



Fonction de transport dans un système distribué

Le fonctionnement interne et les matériels du réseau de transport doivent être indépendants des sites connectés à ce réseau.

Pour les sites connectés au réseau, le réseau de transport doit apparaître comme un boîte noire ayant un fonctionnement fiable et permanent.

Tout site connecté s'adresse au réseau de transport via une interface d'accès adéquate que nous appelons "station de transport", comme dans CYCLADES /LORRAINS,79 /.
La station de transport correspond à la couche 4 du modèle de référence ISO /ISO, 81/ pour l'interconnexion des systèmes ouverts.

Les utilisateurs de la couche transport (ie de la station de transport) ont les moyens d'établir,de maintenir et d'arrêter des connexions de transport qui constituent des voies logiques de transmission bilatérales entre deux entités de la couche supérieure.

Les données sont transmises et fournies aux entités distantes dans le même ordre que celui avec lequel elles ont été émises et sans erreur de transmission. Pour assurer ce service la couche transport fait appel aux ressources de communication du réseau inférieur (ie la couche de gestion du réseau de transmission).

Les services demandés à la station de transport résident dans les points suivants /LORRAINS, 79/ , /MACCHI, 79/:

1) Initialisation des connexions

La ST permet de définir, pour les entités qui désirent entrer en communication des "points d'accès" ou "portes" qui, en les associant deux à deux, constituent des "connexions" ou "liaisons logiques". C'est en utilisant ces connexions que les entités d'application peuvent s'échanger des messages.

2) Arrêt (ou fermeture) des connexions

La ST permet de fermer une connexion et de libérer les ressources qu'elle occupe lorsque les entités qui l'ont demandée ont terminé leurs échanges.

3) Gestion des échanges de messages

La ST permet aux entités d'application ayant déclaré des connexions d'émettre et/ou de recevoir des messages via ces connexions.

4) Contrôle des erreurs de transmission

La ST permet de véhiculer aux entités d'application des messages exempts d'erreur de transmission. Elle détecte les erreurs, les corrige(éventuellement) réémet les messages erronés, etc...

5) Fragmentation et réassemblage

Les messages trop longs sont découpés en messages plus courts (car moins le message est long, plus il a des chances d'arriver à destination, sans erreurs) et réassemblés, à l'arrivée, pour constituer les messages d'origine.

6) Contrôle de flux

Pour éviter de saturer les sites récepteurs de messages, les ST dialoguent entre elles pour savoir si elles peuvent, ou non, s'échanger des messages.

V.2 LA STATION DE TRANSPORT REALISEE

La station de transport qui a été réalisée pour notre maquette est basée sur le standard du DOD /POSTEL, 80 / pour le protocole de transport (un résumé de ce standard est donné dans l'annexe B4).

Nous avons fait deux restrictions sur les fonctions d'une ST évoquées précédemment :
- les messages transmis par les entités d'application sont de taille limitée à 1010 octets ce qui évite la fragmentation et le réassemblage et permet d'envoyer directement ces messages via le réseau local GIXINET ;
- le contrôle de flux se fait de manière statique;pour chaque liaison on définit un facteur d'anticipation (ie le nombre maximum de messages en attente d'émission pour cette connexion) permettant de rejeter les demandes d'émission si le facteur est atteint.

La description de la réalisation de la ST est donnée dans /PICHON, 84/, le mode d'utilisation et le principe général de la réalisation sont donnés dans l'annexe B4. Nous présentons ici les services offerts par la ST aux entités d'application et l'interaction entre la ST et le logiciel de gestion de la connexion au réseau local GIXINET.

V.2.1 Primitives d'accès à la ST

Les services offerts par la ST sont accessibles aux entités d'application via une interface constituée des opérations suivantes :
OPEN, CLOSE, SEND, RECEIVE, ABORT et STATUS.

1) Etablissement d'une connexion

Il y a deux façons, pour une tâche, de se mettre à l'écoute du réseau :
- demande d'une connexion en mode passif : la ST crée un descripteur pour cette connexion ;
- demande d'une connexion en mode actif : la ST crée un descripteur pour cette connexion et entre en dialogue avec la ST distante concernée pour établir la connexion.

Les demandes de connexions se font via la primitive OPEN.

2) Fermeture d'une connexion

Lorsque les entités communicantes terminent leurs échanges ou ne veulent plus s'échanger des messages sur une connexion donnée, elles demandent à leur ST de fermer cette connexion.

Les demandes de déconnexion (ou de fermeture de connexions) se font via l'opération CLOSE.

3) Réception de messages

Une tâche peut demander, à la ST, de lui fournir un message qu'elle a reçu sur une connexion donnée. Ce type de demande se fait via la primitive RECEIVE.

4) Emission de messages

L'émission de message consiste à demander à la ST d'émettre un message sur une connexion qui a été établie auparavant.
Les demandes d'émission se font via la primitive SEND.

5) Avortement d'une connexion

Une tâche peut demander à la ST d'ignorer tous les messages en attente d'émission et/ou de consommation (ie ceux reçus) pour une connexion donnée ; on dit que la connexion est avortée .
Les demandes d'avortement se font via l'opérateur ABORT.

6) Etat d'une connexion

Pour connaître l'état d'une connexion (ie les paramètres relatifs à une connexion (connexion établie, ou non, nombre de messages en attente d'émission et nombre de messages en attente de réception) on s'adresse à la ST via la primitive STATUS.

V.2.2 Utilisation de la connexion au réseau GIXINET

Les services offerts par le réseau local GIXINET constituent les couches 1,2 et 3 du modèle de référence de l'ISO /BNI, 81/.

La ST utilise ces services pour assurer les échanges de messages entre les diverses stations du réseau, et ce de la manière suivante :

1) Initialisation du communicateur

Avant toute opération d'accès au support de transmission, la ST doit initialiser le communicateur via l'opération OUVRGX.
Si l'initialisation est impossible la ST devient non opérationnelle.

2) Déconnexion du communicateur

Une fois que toutes les connexions établies par les entités d'application sont fermées, la ST demande la déconnexion du communicateur via l'opération FERMGX.

3) Envoi de messages

Lorsque la ST termine la préparation d'un message à transmettre (le format d'un tel message est décrit dans l'annexe B4) elle décide de transmettre ce message. Elle construit alors un deuxième message (à partir du premier message (ou message initial)) ayant la structure suivante :

N° station	Longueur du message initial	Message Initial

et demande la transmission de ce message via l'opération EMETGX. A la fin de la transmission le communicateur active le S/P de traitement de l'interruption de la fin d'émission ITEMET . La ST appelle alors l'opération RESMGX pour savoir si l'émission s'est bien passée ou non. Si le message transmis a été bien transmis la ST agit comme suit :
- si le message est un message de données (ie appartenant à une entité d'application) la ST avertit l'entité concernée par ce message pour lui signaler que le message a été bien transmis.
- si le message est un message-fonction (ie défini par la ST pour le dialogue entre ST) la ST change l'état de la connexion en fonction de ce message, par exemple passage de l'état fermé à l'état établi (cf l'annexe B4 pour les changements d'état d'une connexion).

Si le message n'a pas été transmis, il est remis dans la file des messages en attente d'émission associée à la connexion traitée (il appartient aux entités d'application de purger les connexions en cas d'impossibilité de transmission).

4) Réception de messages

Au moment où le communicateur reçoit un message, il informe la ST via le S/P de traitement de l'interruption fin réception (ITRECP). La ST prélève alors le message en appelant auparavant l'opération RECPGX qui lui fournit les informations relatives au message reçu.

Après prélèvement du message, la ST détermine la connexion concernée par ce message et agit comme suit :
- si le message reçu est un message de données, la ST chaine ce message à la liste des messages reçus sur la connexion ;
- si le message reçu est un message fonction, la ST agit sur l'état de la connexion en fonction de ce message (établissement de la connexion, fermeture de la connexion etc... -Cf annexe B4-).

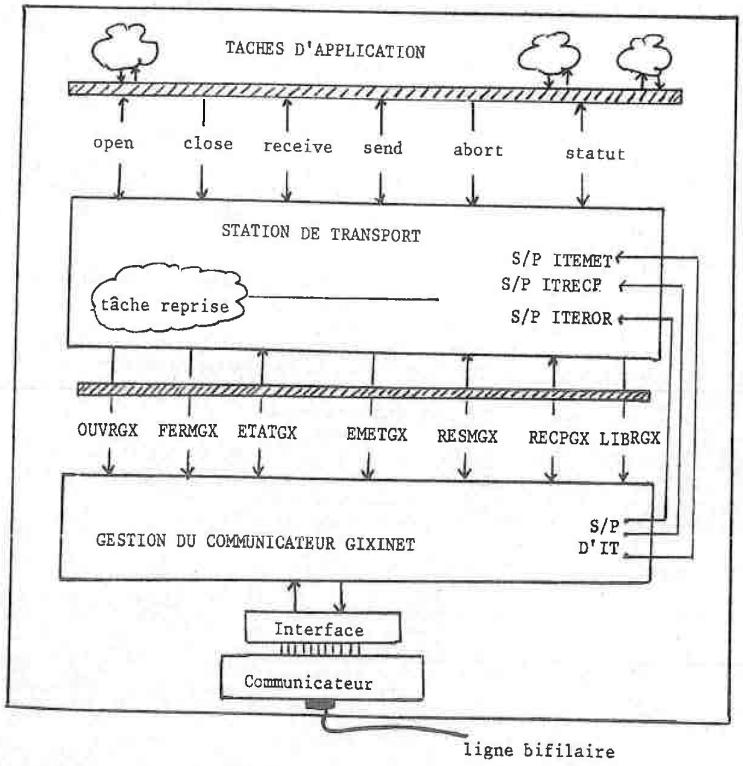
6) Cas d'erreur du communicateur

Dans le cas où le communicateur détecte une erreur irrécupérable le S/P de traitement de l'erreur-communicateur (ITEROR) est activé; il fait passer la ST dans l'état non-opérationnel.

Lorsque la ST passe dans l'état non-opérationnel une tâche dite de "reprise" tente de manière cyclique de réactiver le communicateur.

V.2.3 Schéma récapitulatif

Nous résumons, à l'aide de la figure ci-après, l'interaction entre les tâches d'application et la ST d'une part et le logiciel de gestion du communicateur GIXINET et la ST d'autre part.



Composantes matérielles et logicielles pour la communication

VI CONCLUSION

Nous venons de présenter de manière assez rapide un ensemble de composantes que nous rappelons pour mémoire :
- un noyau temps réel ;
- une agence horlogerie ;
- et une station de transport construite autour d'un réseau local, et qui constitue un noyau de base pour l'implantation des systèmes de temps réel et répartis.

En ce qui concerne la maquette réalisée, nous pouvons faire deux remarques, l'une concernant sa modularité, l'autre concernant son coût.
- la modularité de la maquette : les différentes composantes de la maquette ont été réalisées et testées avec une certaine indépendance. Cette maquette se prête bien à des extensions futures (par exemple l'implantation d'un à partir de cette maquette) ;
- le coût de la maquette, relativement faible, est dû au prix des systèmes à base de microprocesseurs et surtout aux réseaux locaux qui offrent, de nos jours, des moyens de communication fiables, rapides avec des prix de revient intéressants. Avec ces caractéristiques les réseaux locaux facilitent et stimulent l'expérimentation des systèmes répartis.

Le travail que nous présentons ici constitue une contribution aux méthodes de conception des applications réparties en commande de procédés industriels.

Nous voudrions, à travers cette conclusion :
- tout d'abord rappeler les principales idées de notre travail
- et ensuite esquisser quelques perspectives de développement que nous envisageons comme suite à ce travail.

Bilan : résultats essentiels
L'approche de conception d'ACPI qui a été développée est fondée sur la prise en compte des informations relatives à commande et à la conduite des procédés industriels.

Une telle approche permet de structurer une ACPI à l'aide :
- d'un ensemble de données (ou BDI),
- d'un ensemble de traitements à effectuer sur les données (ou actions),
- et de règles de manipulation des données.

Nous nous sommes fixés, comme objectif, la prise en compte et l'expression relativement naturelle, des principales caractéristiques des ACPI, c'est-à-dire :
- leur aspect temps-réel,
- la sûreté de fonctionnement qu'elles exigent,
- leur répartition,
- et leur évolutivité.

Pour atteindre cet objectif nous avons proposé une approche de conception et d'implantation des ACPI. Les idées directrices d'une telle approche résident dans les points suivants :
1. séparation entre données et traitements,

2. pilotage des traitements par les données,

3. séparation entre partie opérative et partie contrôle (ie que la synchronisation des traitements est exprimée à l'extérieur de ces traitements),

4. expression de l'aspect dynamique des applications à l'aide des concepts de temps, d'évènements et de conditions,

5. conception modulaire des applications. Cette conception se fait en trois étapes :
   - recherche des données et des actions,
   - recherche de conditions de manipulation des données,
   - et description des algorithmes composant les actions.

6. attribution d'un sens à chaque objet (ie que toute action, ou toute donnée, introduite doit avoir un sens vis-à-vis de la commande globale),

7. description homogène des applications : tous les modes de fonctionnement (ie fonctionnement normal et fonctionnement en cas d'anomalies) s'expriment par les mêmes concepts,

8. hierarchisation des installations de commande,

9. et répartition des applications.

CONCLUSION
BILAN ET PERSPECTIVES

Cette approche intègre les aspects :

- structures de données,

- événement/condition (/LADET,77,82/, /THOMESSE,80/, /BENMAIZA, 84/),

- et flux de données /DERNIAME,82A, 82B, 82C, 83/, /ZAKARI, 84/; et permet de montrer que ces aspects sont complémentaires et peuvent être combinés décrire des ACPI.

Si l'on juge de l'intérêt de l'approche de conception/implantation orientée données on retient particulièrement ses apports au niveau de :

- complément du cahier des charges,

- l'existence d'une base de données unique (unicité au sens fonctionnel),

- la prise en compte des caractéristiques des procédés,

- et de la vérification des applications.

1) Complément du cahier des charges

L'approche de conception développée précédemment permet d'aider à compléter le cahier des charges, à lever les ambiguités qui s'y trouvent, etc... avant de passer à une réalisation.

En effet, pour chaque service que devra réaliser le système de commande, nous associons un ensemble de données, un ensemble d'actions et des règles de manipulation des données.

Pour définir une donnée le concepteur devra remplir un certain nombre de rubriques (structure de la donnée, ses producteurs, etc...). Il cherchera alors à répondre aux questions relatives à ces rubriques en se référant au cahier des charges. A travers cette recherche il décélera les oublis (par exemple, de description des conditions de manipulation d'une certaine information), les imprécisions... du cahier des charges et demandera alors des éclaircissements à son client.

Comme les questions posées sont connues d'avance (elles dépendent du modèle de description des applications) le concepteur sera guidé dans son exploration du cahier des charges.

2) Base de données unique

L'ensemble des traitements relatifs à la "vie" d'un procédé, à savoir :

- la commande en temps réel du procédé,
- la conduite par les opérateurs, du procédé,
- l'identification et la modélisation du procédé,
- la conception des modèles de commande, de fabrication de pièces...
- la gestion de l'usine (ie la facturation, le traitement des commandes,..)

- etc...

manipulent des informations de natures très diverses, il est alors impératif que ces informations soient cohérentes entre elles. La base de données industrielles constitue un moyen unique de communication d'informations entre les diverses équipes (opérateurs, automaticiens,..,gestionnaires).

Toutes les données sont décrites et gérées par les mêmes mécanismes et les mêmes règles ; c'est en ce sens que la BDI est dite un moyen unique de communication.

3) Prise en compte des caractéristiques des procédés

Notre approche de conception/implantation des applications tient compte des caractéristiques des procédés. En effet :

- la modularité des applications facilite leur maintenance, leur évolutivité et leur répartition,

- l'implantation d'applications, à l'aide d'installations hiérarchisées, aide à assurer :

  . les contraintes de temps imposées,
  . et la cohérence des informations échangée entre les divers équipements.

4) Vérification et validation des applications

- Les actions constituant une application sont indépendantes entre elles. Cette indépendance facilite leur vérification. Chaque action peut être vérifée et validée de manière séparée (en lui associant un réseau de Pétri),

- les règles de mise à jour sont exprimées à l'extérieur des actions ; ce qui permet au concepteur de valider l'aspect dynamique de l'application et de s'assurer que tous les cas ont été prévus.

La proposition d'une approche de conception/implantation d'ACPI constitue le premier volet de notre travail. Nous avons, aussi, présenté une maquette réalisée autour d'un réseau local.

Les services offerts par cette maquette constituent un ensemble de mécanismes de base (de synchronisation et de communication) nécessaires à l'implantation des systèmes temps réel et répartis (et plus particulièrement à l'implantation des systèmes de gestion des bases de données industrielles réparties).

Une réalisation complète rapprochant les bases de données industrielles et les systèmes temps réel et répartis n'a pas été effectuée dans le présent travail ; espérons qu'elle le sera un jour pour mieux valider les propositions faites dans cette thèse.

Perspectives

Dans ce travail nous avons soulevé beaucoup de problèmes allant de l'analyse des cahiers des charges jusqu'à l'implantation des ACPI et proposé quelques voies de solutions à ces problèmes.

Il convient maintenant de jeter un regard vers l'avenir et présenter les grands axes des développements que nous envisageons comme suite à ce travail.

1) La cohérence de la BDI

C'est là, à notre avis, le point le plus crucial pour la sécurité des ACPI; après l'avoir esquissé, dans ce travail, nous préconisons d'étudier une méthode permettant :

- de mieux guider le concepteur à rechercher les situations de mauvais fonctionnements qui conduisent à des incohérences de la BDI ;

- de reprendre l'exploitation de la BDI après détection d'incohérences: Il s'agit de développer des mécanismes de reprise ;

- de s'assurer, à tout instant, que l'état de la BDI est cohérent pour l'ensemble de la commande ;
- ...

2) Aspect méthodologique

Pour que la méthode proposée dépasse le stade d'un ensemble de conseils et soit réellement profitable il faut la faire reposer sur des fondements théoriques. Les travaux à effectuer à ce niveau résident dans :

- la définition des règles permettant de guider le concepteur (comme le fait CALVEZ /CALVEZ, 82/) ;

- la formalisation des concepts proposés (la fomalisation des donnée, sous forme de type abstraits /DERNIAME, 79/, serait une voie de solution) ;

- la vérification, de manière statique, des solutions obtenues en utilisant une approche de preuve axiomatique /OWICKI, 76/ , /KELLER, 76/ ou des réseaux de Pétri /PETERSON, 77/, /MOALLA, 78/.
- ...

3) Réalisations

Outre les aspects théoriques, des réalisations devraient être effectuées pour valider les propositions faites dans cette thèse :

- premièrement implanter (autour de la maquette déjà réalisée) un système de commande manipulant une BDI ;

- deuxièmement réaliser un système d'aide à la conception (en se basant sur les règles associés à la méthodologie de conception).

- ...

ANNEXES A1, A2, A3 et A4

A1 : coulée continue

A2 : moteur à courant continu

A3 : syntaxe du langage de description des ACPI

A4 : projet FIP

ANNEXE A1

PRESENTATION D'UN EXEMPLE DE PROCEDE INDUSTRIEL COMPLEXE ET DE SA COMMANDE

Cas de la coulée continue de Gandrange (Sollac)

L'objectif de cette annexe est de présenter :
- le processus de coulée continue de Gandrange ;
- et les idées directrices de la commande de cette coulée continue

Vu la complexité de la coulée continue et de sa commande nous nous limiterons à ne citer que les concepts généraux.

I PRESENTATION DE LA COULEE CONTINUE DE SOLLAC-SACILOR

La coulée continue de SOLLAC-SACILOR de Gandrange /SOLLAC, 80/ est constituée de 6 brins (ou machines de coulée continue).

Objectif
Une machine constitue une ligne de coulée et a pour rôle de couper et d'évacuer des blooms de longueurs comprises entre 4 et 11 m.

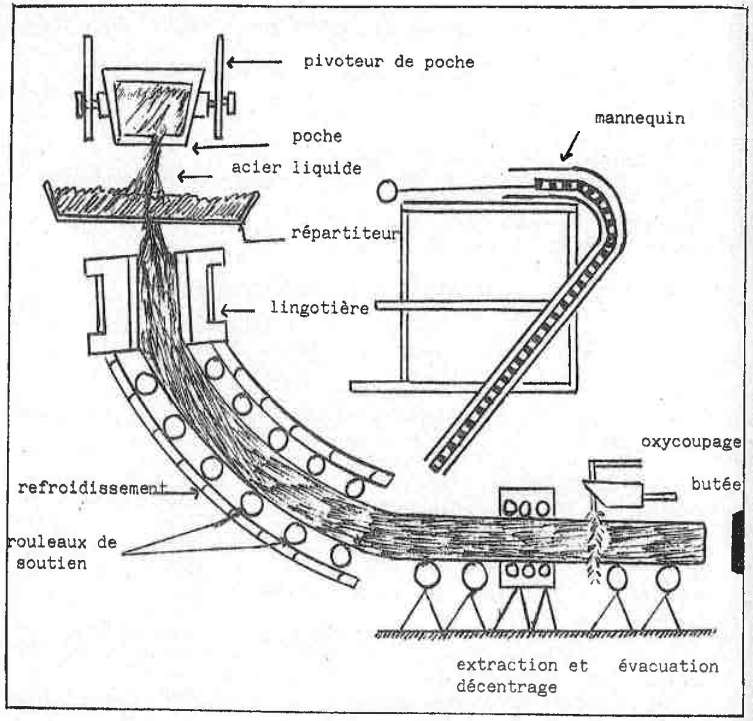
Principe de la coulée continue
Le procédé consiste à obtenir, à partir d'acier liquide, directement des demi-produits laminables en produits finis.

L'acier liquide est versé de façon régulière et ininterrompue dans un moule sans fond, ou lingotière, énergétiquement refroidi.

Le produit extrait en continu de la lingotière est une barre dont le coeur, encore liquide, achève de se solidifier sous l'effet d'un violent refroidissement. Il résulte de ce principe que le procédé est basé sur l'obtention rapide d'une peau suffisamment solide pour retenir le métal liquide.

I.1STRUCTURE D'UNE MACHINE DE COULEE CONTINUE

Comme le montre la figure ci-après, une machine de coulée continue est composée d'un ensemble d'éléments dont les principaux sont les suivants
- une poche de coulée
- un distributeur (répartiteur, panier, etc...)
- une lingotière
- des rouleaux de soutien
- des zones de refroidissement
- des cages d'extraction et de décintrage
- des tables d'oxycoupage
- des tables d'évacuation
- et un mannequin.



Structure générale d'une ligne de coulée continue

1) Le pivoteur de poche

Son rôle est de passer les poches pleines d'acier liquide de la halle de chargement à la halle de coulée et inversement de passer les poches vides de la halle de coulée à la halle de chargement.

2) Le distributeur

Sa fonction est de distribuer l'acier aux différentes lignes de coulée. Il sert aussi de tampon entre poche et lingotière.

3) La lingotière

La lingotière est le premier élément (c'est le plus important) de la confection et du refroidissement du produit.

A cet effet, 4 plaques en cuivre sont refroidies par circulation d'eau et assemblées par des mécanismes conçus pour maintenir l'ensemble stable tout au long de la coulée.

4) Les refroidissements

Pendant la coulée, la fonction de refroidissement (à l'aide de circulation d'eau) est de protéger et d'accroître l'épaisseur de la peau du produit jusqu'à la solidification du coeur.

Les segments placés directement sous la lingotière et jusqu'aux cages extractrices comportent des rouleaux de guidage et des rampes de pulvérisation.

5) Le mannequin

C'est un ensemble comportant un corps à maillons articulés relié à une tête en acier moulé et sert principalement :
- à boucher hermétiquement le fond de la lingotière lors de son remplissage en début de coulée ;
- à guider et à entrainer la tête du bloom dans les cages jusqu'à la sortie de l'extraction.

6) Les cages d'extraction et de décintrage

Elles ont pour rôle :
- d'introduire le mannequin,
- d'extraire le bloom par le mannequin,
- d'entrainer le bloom par contact direct,
- de décintrer le bloom à la température normale, au fur et à mesure de son avance,
- et de permettre le dégagement d'un bloom froid non décintré.

7) Les tables d'oxycoupage

Chacune des machines d'oxycoupage assure la coupe du produit de la ligne à laquelle elle est associée, à longueurs désirées, à l'aide d'un équipement de coupe comportant un chalumeau.

8) Les tables d'évacuation

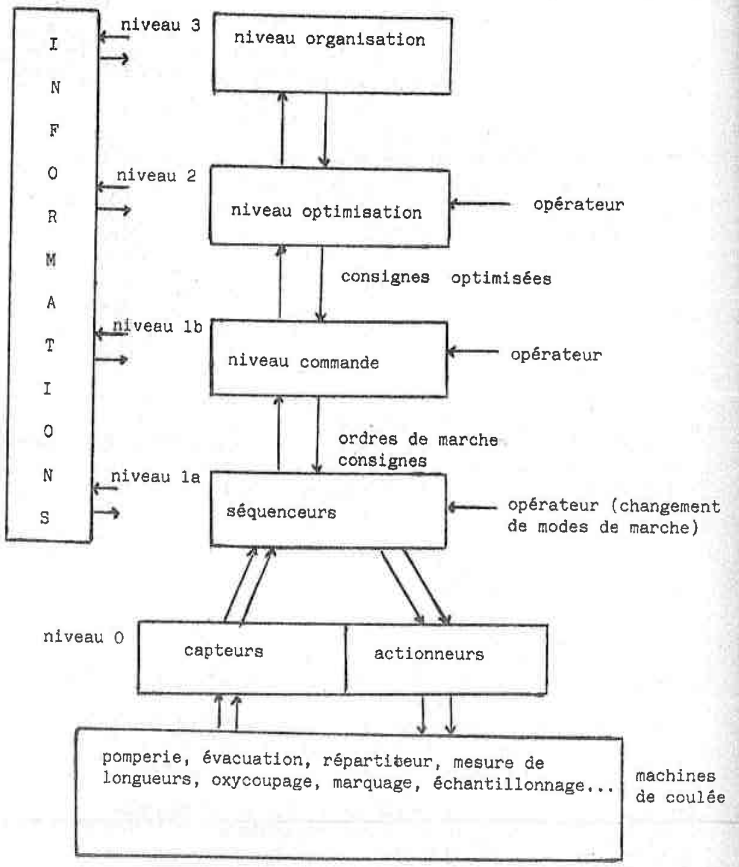
L'évacuation des blooms se fait par un ensemble de rouleaux automoteurs répartis en 3 tables :
- une table entre extracteurs et l'oxycoupage,
- une table entre l'oxycoupage et les butées mobiles,
- et une table entre les butées mobiles et les butées fixes.

II ARCHITECTURE DU SYSTEME DE COMMANDE DE LA COULEE CONTINUE

Nous présentons, ici, les grandes lignes du système de commande de la coulée continue de Gandrange.

II.1 STRUCTURE GENERALE DU SYSTEME DE COMMANDE

Le système de commande est hiérarchisé en 5 niveaux (cf la figure ci-après):



Structure hiérarchisée de la commande de la coulée continue

- le niveau 0 est celui des équipements de la machine de coulée continue avec les actionneurs et les capteurs ;
- le niveau 1a est celui des séquenceurs de commande des équipements ;
- le niveau 1b est celui du calculateur de commande (un solar 16-65) chargé de coordonner les actions des séquenceurs ;
- le niveau 2 est celui du calculateur d'optimisation (un solar 16-65) ;
- le niveau 3 est celui de l'ordinateur central de gestion (un IBM).

Toutes les informations (états, défauts) créées à un niveau, passent au niveau supérieur si celui-ci les utilise.

Les niveaux 2 et 3 sont dotés de moyens de mémorisation (disques) pouvant stocker beaucoup d'informations sur la vie du procédé et de sa commande.

Les divers équipements de la machine de coulée continue peuvent fonctionner suivant plusieurs modes de marche (ie automatique, semi-automatique, manuel, test et maintenance).

II.2 LES FONCTIONS REALISEES PAR LE SYSTEME

a) au niveau séquenceurs
- recueil des états des équipements
- recueil des défauts primaires des équipements
- commande des équipements compte-tenu des informations recueillies et des ordres venant soit de l'opérateur (en mode semi-automatique) soit du niveau 2 (en mode automatique) ;
- création des défauts calculés et transmission aux niveaux supérieurs ;
- possibilité de commande en manuel.

b) au niveau calculateur de commande
- coordination de l'ensemble des séquences et enchaînement répétitif des séquences réalisées au niveau 1a ;
- suivi de produit alimenté par les mesures de longueur sur chaque ligne de coulée ;
- gestion du dialogue avec l'opérateur :
  . commandes centralisées : envoi d'ordres à l'aide de grille de commande ;
  . visualisation des défauts et des informations demandées par l'opérateur.
- transmission des informations utiles vers le niveau supérieur.

c) au niveau optimisation
- suivi complet de produits, y compris le repérage des défauts métallurgiques ;
- visualisation de synoptiques sur écrans animés par les changements d'état et les défauts ;
- acquisition, stockage et visualisation sous forme de courbes et de tableaux d'informations telles que les mesures... ;
- édition de rapports concernant l'exploitation et les défauts ;
- exploitation des modèles d'optimisation ;
- liaison avec l'ordinateur central de gestion.

d) au niveau gestion
gestion de l'ensemble de l'usine :
- personnel,
- statistiques sur les produits,
- commandes et factures,
- comptabilité,
- etc...

II.3 REMARQUE

L'exemple de la coulée continue permet :
- d'illustrer l'utilisation de la commande hiérarchisée pour la construction d'un système de commande d'un procédé complexe ;
- et de montrer la diversité des types de données et d'actions constituant une application en conduite de procédés.

ANNEXE A2

CAHIER DES CHARGES DE LA COMMANDE D'UN MOTEUR A COURANT CONTINU
(tiré de /IUNG, 83/)

Le problème posé est une régulation de vitesse d'un moteur à courant continu alimenté par un pont réversible à thyristor. La mesure de la vitesse est réalisée de manière numérique. La structure de régulation est la structure classique : boucle de courant, boucle de vitesse de type proportionnel intégral. Les sécurités en courant maximum et variation maximum du courant doivent être prises en compte au niveau de la référence de courant. On désire avoir une possibilité de fonctionnement autonome du générateur d'impulsionsqui devra donc posséder ses propres sécurités.

Le passage d'un pont sur l'autre devra se faire sans surintensité. Le démarrage du dispositif doit se faire en respectant les contraintes de sécurité.

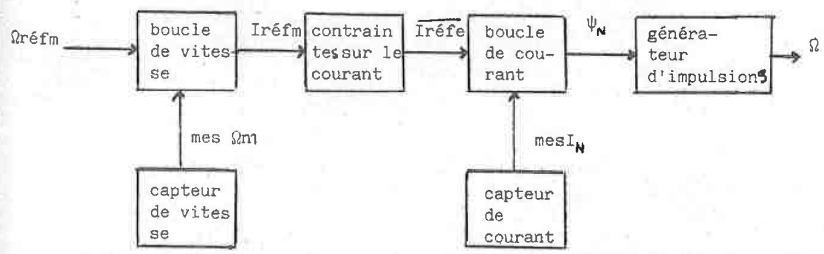
1) Phase de démarrage

Partant de l'état non alimenté et sur une demande de l'extérieur, on doit passer par une étape où le système de commande est alimenté (alimentation passive) et teste les conditions suivantes de bonfonctionnement :
- initialisation des différents régulateurs ;
- Ωréf = 0 (vitesse de référence) ;
- I réf = 0 (courant de référence) ;
- ψ = π (butée onduleur) ;
- présence réseau ;
- configuration correcte des phases.

Si une ou plusieurs de ces conditions ne sont pas vérifiées un message doit être émis. Si toutes ces conditions sont satisfaites le système peut être alimenté et peut prendre en compte la consigne de vitesse (alimentation active).

2) Structure de la régulation

La structure choisie pour l'ensemble de la régulation est la suivante :



La boucle de vitesse lit la référence de vitesse Ω refm et grâce à un algorithme PI, élabore une référence courant I refm. Par comparaison avec Imax et (dI/dt)max on produit ensuite la référence courant $\overline{Irefe}$ qui tient compte des contraintes sur le courant.

La boucle de courant calcule l'angle d'allumage Ψ n par un algorithme P.I à partir de $\overline{Irefe}$ et de la valeur du courant MeS_N.

3) Le générateur d'impulsions

Le générateur d'impulsions est du type générateur à coïncidence. Il inclut butée onduleur et redresseur et assure, le basculement sans courant de circulation.

En entrée, il a :

- Ψ j : l'angle d'allumage ;
- N (= 1 ou 2) numéro du pont

et en sortie
- il génère, des impulsions, sur un pont donné (N= 1 ou 2), sur un thyristor donné (K= 0,..5) à une date J (Jème activation de la fonction).

4) Régulation de courant

La régulation de courant doit lire la référence $\overline{Iréfe}$ et en déduire la politique de commande à appliquer.

Tant que la référence de courant ne change pas de signe, Ψ_N est calculé par un algorithme P.I.

Lorsque la référence courant change de signe, la boucle de courant doit attendre que I soit nul avant d'envoyer le changement du pont au générateur d'impulsions. Il faut alors calculer l'angle d'allumage initial Ψ o sur le nouveau pont de manière à ce que le basculement s'effectue sans surintensité et le plus rapidement possible. L'intégrale PI doit être réinitialisée de manière à assurer la continuité des angles d'allumage Ψ_N à partir de Ψo.

5) Contraintes sur le courant

Les valeurs absolues du courant et de sa dérivée sont bornées respectivement par IMax et Δ Imax. Ces contraintes sont reportées au niveau de la référence courant. Cette fonction reçoit une référence courant $\overline{Irefm}$ du régulateur de vitesse et élabore une référence de courant $\overline{Irefe}$ tenant compte des contraintes.

6) Régulation de vitesse

Le régulateur de vitesse est un PI de type classique. Cependant pour éviter la dérive, lorsque le respect des contraintes conduit à $\overline{I\ refe} \neq$ Irefm il faut réinitialiser le régulateur de vitesse de manière à suivre l'évolution de Iref tant que $\overline{Iréfe} \neq$ Irefm. On ne prend pas en compte les fonctionnements anormaux du type capteur de vitesse défaillant ou régime de survitesse.

La boucle de vitesse lit la référence de vitesse Ω refm et grâce à un algorithme PI, élabore une référence courant I refm. Par comparaison avec Imax et (dI/dt)max on produit ensuite la référence courant $\overline{Irefe}$ qui tient compte des contraintes sur le courant.

La boucle de courant calcule l'angle d'allumage Ψ n par un algorithme P.I à partir de $\overline{Irefe}$ et de la valeur du courant MeS_N.

3) Le générateur d'impulsions

Le générateur d'impulsions est du type générateur à coïncidence. Il inclut butée onduleur et redresseur et assure, le basculement sans courant de circulation.

En entrée, il a :

- Ψ j : l'angle d'allumage ;
- N (= 1 ou 2) numéro du pont

et en sortie
- il génère, des impulsions, sur un pont donné (N= 1 ou 2), sur un thyristor donné (K= 0,..5) à une date J (Jème activation de la fonction).

4) Régulation de courant

La régulation de courant doit lire la référence $\overline{Iréfe}$ et en déduire la politique de commande à appliquer.

Tant que la référence de courant ne change pas de signe, Ψ_N est calculé par un algorithme P.I.

Lorsque la référence courant change de signe, la boucle de courant doit attendre que I soit nul avant d'envoyer le changement du pont au générateur d'impulsions. Il faut alors calculer l'angle d'allumage initial Ψ o sur le nouveau pont de manière à ce que le basculement s'effectue sans surintensité et le plus rapidement possible. L'intégrale PI doit être réinitialisée de manière à assurer la continuité des angles d'allumage Ψ_N à partir de ΨO.

5) Contraintes sur le courant

Les valeurs absolues du courant et de sa dérivée sont bornées respectivement par IMax et Δ Imax. Ces contraintes sont reportées au niveau de la référence courant. Cette fonction reçoit une référence courant Irefm du régulateur de vitesse et élabore une référence de courant $\overline{Irefe}$ tenant compte des contraintes.

6) Régulation de vitesse

Le régulateur de vitesse est un PI de type classique. Cependant pour éviter la dérive, lorsque le respect des contraintes conduit à $\overline{I\ refe} \neq$ Irefm il faut réinitialiser le régulateur de vitesse de manière à suivre l'évolution de Iref tant que $\overline{Iréfe} \neq$ Irefm. On ne prend pas en compte les fonctionnements anormaux du type capteur de vitesse défaillant ou régime de survitesse.

7) Capteur de vitesse

Sur demande, le capteur de vitesse délivre une grandeur numérique signée proportionnelle à la vitesse de rotation à partir des signaux émis par un disque incrémental.

On désire une précision constante sur la plage Ω min = 12t/mn, Ω max = 2000 t/mn.

Si Ω>2000t/ mn il faut prévenir la régulation.

Vu les contraintes de temps dues à la mécanique du système, il faut que le temps d'intégration de la vitesse ne soit supérieur à 2 ms, pour les hautes vitesses, et 5 ms, pour les basses vitesses.

7) Capteur de vitesse

Sur demande, le capteur de vitesse délivre une grandeur numérique signée proportionnelle à la vitesse de rotation à partir des signaux émis par un disque incrémental.

On désire une précision constante sur la plage Ω min = 12t/mn, Ω max = 2000 t/mn.

Si Ω>2000t/ mn il faut prévenir la régulation.

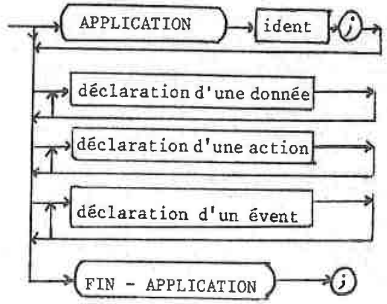
Vu les contraintes de temps dues à la mécanique du système, il faut que le temps d'intégration de la vitesse ne soit supérieur à 2 ms, pour les hautes vitesses, et 5 ms, pour les basses vitesses.

ANNEXE A3

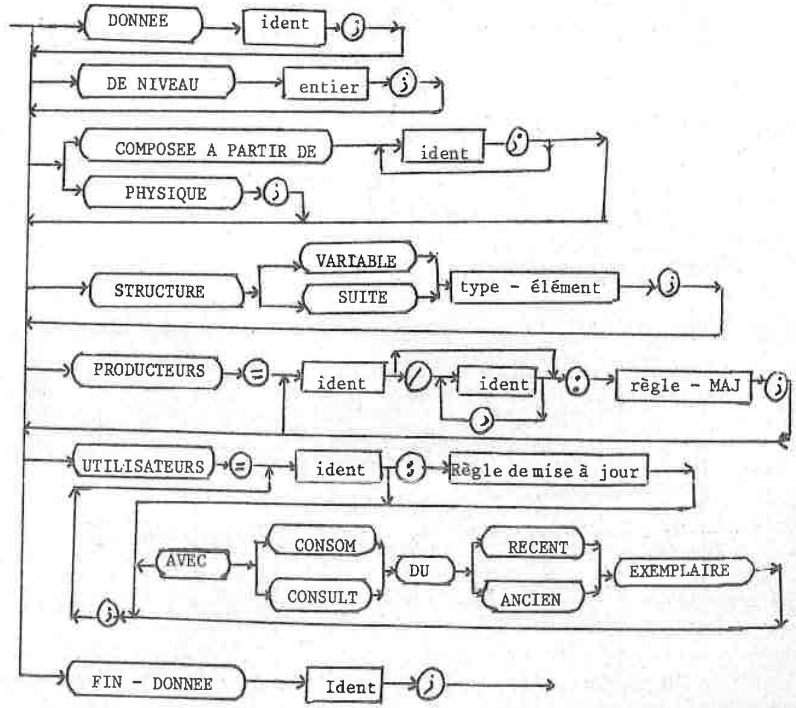
SYNTAXE DU LANGAGE DE DESCRIPTION DES APPLICATIONS

1) Description des applications

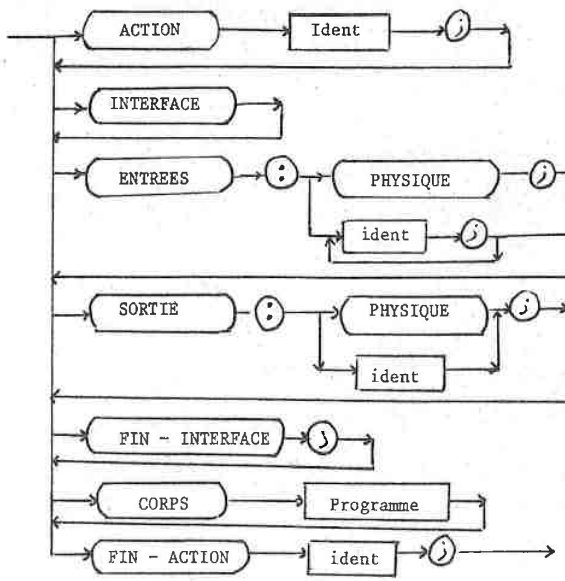
Application



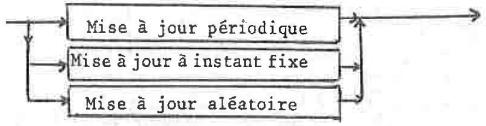
Déclaration d'une donnée :



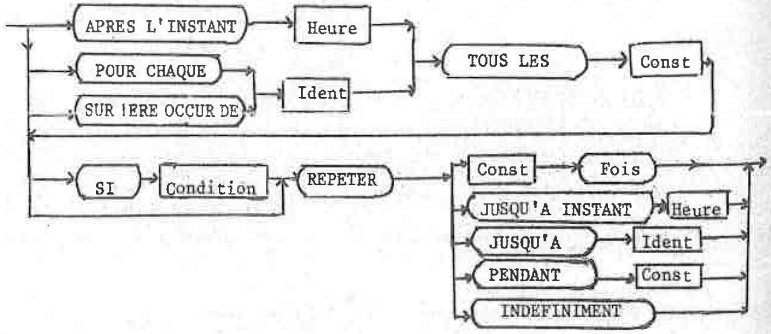
Déclaration d'une action



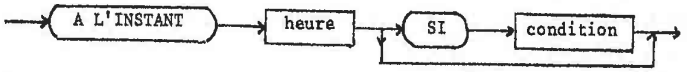
Règle de mise à jour (ou règle - MAJ)



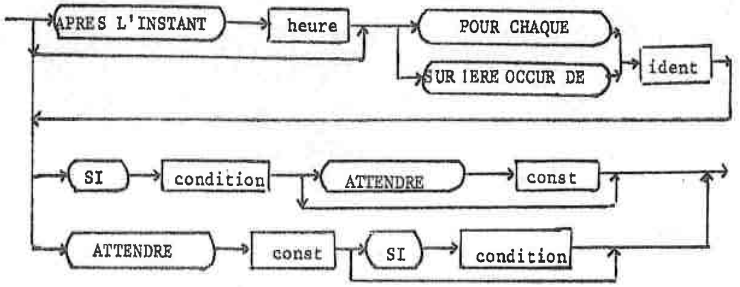
Mise à jour périodique



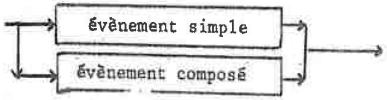
Mise à jour à instant fixe



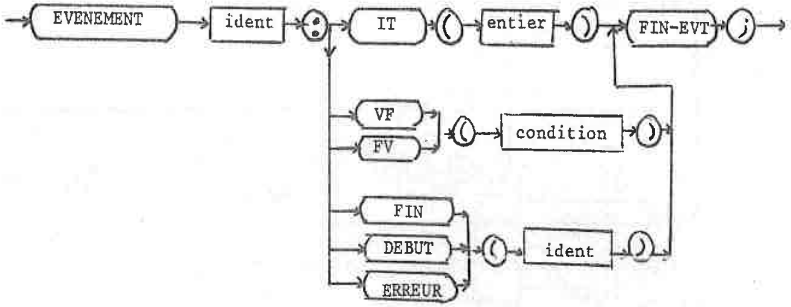
Mise à jour aléatoire



Déclaration d'évènement



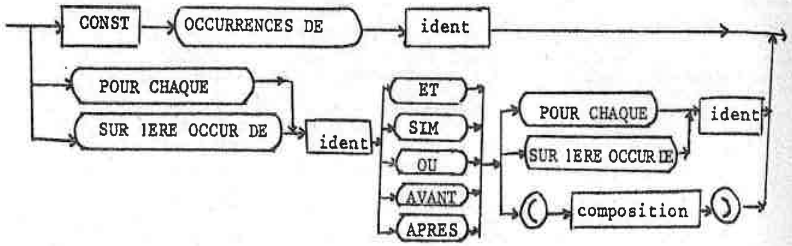
Evènement simple



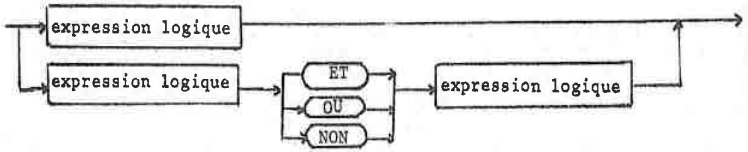
Evènement composé



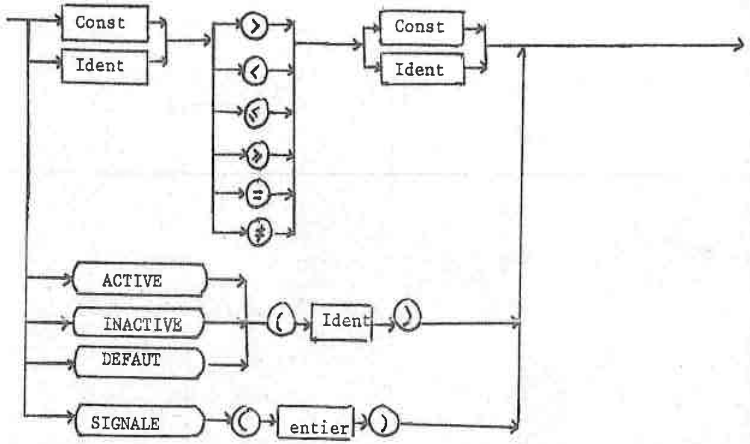
Composition



Condition

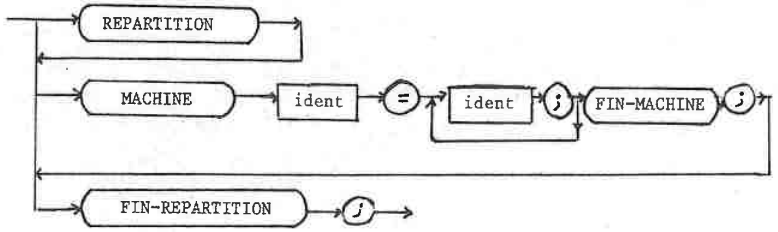


Expression logique



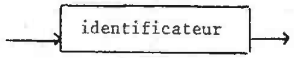
2) Description de la répartition des applications

Table de répartition

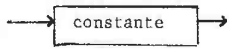


3) Eléments syntaxiques communs

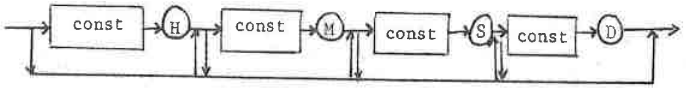
ident :



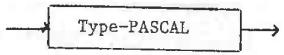
const :



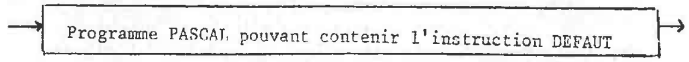
heure :



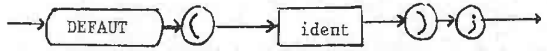
type - élément :



programme :



instruction DEFAUT :



identificateur, constante, type et programme sont conformes aux règles données pour le langage PASCAL /WIRTH, 71a/.

ANNEXE A4 PRESENTATION DU PROJET FIP

FIP : PROPOSITION D'UN SYSTEME DE TRANSMISSION SERIE MULTIPLEXE POUR LES ECHANGES D'INFORMATIONS ENTRE DES CAPTEURS, DES ACTIONNEURS ET DES AUTOMATES REFLEXES

I INTRODUCTION

On présente, ici, un nouveau système de communication adapté à la transmission d'informations entre capteurs, actionneurs et automates réflexes dans une installation industrielle.

FIP est le sigle pour"Flux Information Processus".
FIP est défini par le groupe RLI (Réseaux Locaux Industriels) du Ministère de l'Industrie Française.

Il permet de nombreuses propriétés, entre autres :
- amélioration de la compétitivité,
- amélioration de l'évolutivité,
- haute sécurité,
- et facilité de mise en oeuvre.

Techniquement, FIP permet d'assurer que les temps de transfert d'informations soient garantis et que les débits d'informations soient adaptés à la quasi-totalité des automatismes industriels.

FIP n'existe pas en tant que produit pour le moment.

Quand il existera, FIP permettra des coûts réduits pour le raccordement d'équipements.

II PRESENTATION DE FIP

Il s'agit de définir un protocole d'échange d'informations entre des équipements de traitement d'informations (automates programmables ou calculateurs industriels), des équipements d'instrumentation (capteurs ou actionneurs) et des interfaces opérateurs.

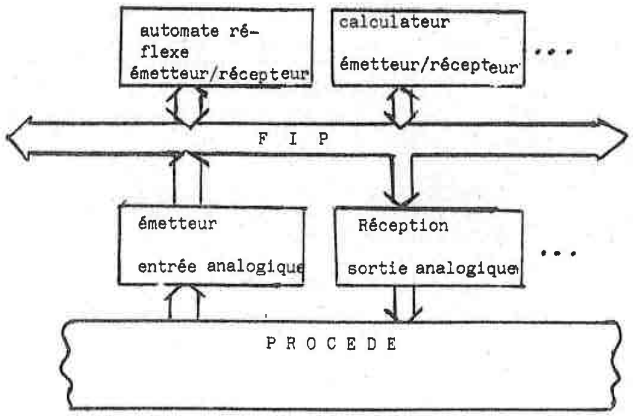
Ce protocole doit être suffisamment général pour permettre d'interconnecter facilement des équipements hétérogènes.

Le composant FIP

Le composant FIP de base permet de recevoir/émettre des informations en provenance/ à destination du support de communication (câble coaxial fibre optique...).

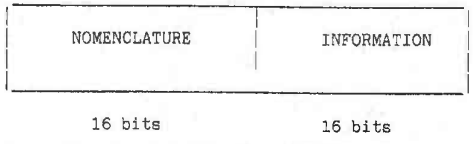
Il assure une liaison physique et logique entre les outils de ligne et l'interface d'application.

Les différents équipements connectés à FIP peuvent se résumer comme suit :
- des équipements de réception (sortie analogique, sortie logique, visualisation...) ;
- des équipements d'émission (acquisition analogique ou logique, commandes issues d'automates, échanges d'informations interautomates...) ;
- et des équipements d'émission/réception (automates réflexes, calculateurs...).



III FORMAT DE LA TRAME

Un échange sur le support se traduit par la succession :
- d'un mot qui correspond à la nomenclature de l'information qui va être mise à disposition sur le médium ;
- d'un mot qui suit la nomenclature et qui correspond à la valeur de l'information correspondant à cette nomenclature.



L'information associée à la nomenclature peut être :
- le résultat d'une conversion analogique-numérique d'un capteur,
- une commande à destination d'un automate,
- une variable interne à un automate.

IV TECHNIQUE DE TRANSMISSION

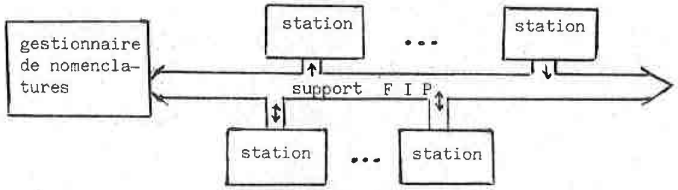
FIP est une liaison série multiplexée qui n'utilise qu'un mode unique de transmission: la diffusion générale.

Les trames d'informations sont émises cycliquement sur le support.

Le cycle d'émission n'est pas figé mais peut être simplement un multiple du cycle de base de scrutation.

Fonctionnement

Un gestionnaire de nomenclatures diffuse sur le support une nomenclature. La station qui dispose de cette information se reconnaît.
La station qui s'est reconnue met l'information, associée à la nomenclature qui vient d'être émise, à disposition sur le médium.
Les stations qui veulent consommer cette information lisent l'information émise par la station productrice.
le gestionnaire de nomenclatures consomme également l'information afin de s'assurer que la transaction s'est effectuée correctement et le cycle continue par l'envoi d'une nouvelle nomenclature,
l'émission des nomenclatures se fait par consultation d'une table de nomenclatures qui fournit, à un moment donné, la prochaine nomenclature à émettre.



V FONCTIONNALITES DU SYSTEME DE COMMUNICATION FIP

V.1 LE GESTIONNAIRE DE NOMENCLATURES

Les fonctions du gestionnaire de nomenclatures se résument comme suit :
- gestion de la nomenclature à émettre en fonction de la fréquence d'émission de cette nomenclature ;
- vérification de l'émission de la donnée associée à la nomenclature ;
- cadencement des trames ;
- test de bon fonctionnement de FIP (détection d'erreurs...).

V.2 LE RECEPTEUR D'INFORMATIONS

Les fonctions des récepteurs d'informations se résument comme suit :
- détection de présence d'une nomenclature sur le support ;
- reconnaissance d'utilité de réception d'une information associée à cette nomenclature ou attente d'une nouvelle nomenclature ;
- réception de la donnée associée à la nomenclature ;
- rangement de la donnée associée à la nomenclature qui vient d'être lue.

V.3 L'EMETTEUR D'INFORMATIONS

- détection de présence d'une nomenclature sur le médium ;
- réception d'une nomenclature ;
- reconnaissance de l'utilité d'émission de la donnée associée à cette nomenclature ;
- sélection de la donnée à émettre (associée à la nomenclature qui vient d'être lue);
- émission de la donnée associée à la nomenclature.

V.4 L'EMETTEUR/RECEPTEUR D'INFORMATIONS

Ses fonctions correspondent à toutes combinaisons des fonctions d'un émetteur et d'un récepteur d'informations.

ANNEXES B1, B2, B3 et B4

SPECIFICATION D'IMPLANTATION ET MODE D'UTILISATION D'UN LOGICIEL DE BASE POUR LA CONSTRUCTION DES SYSTEMES TEMPS-REEL ET REPARTIS

ANNEXE B1

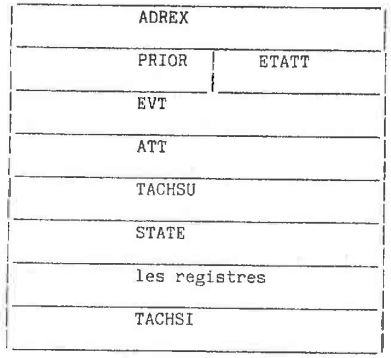
IMPLANTATION ET UTILISATION DU NOYAU TEMPS REEL SCEPTRE

Nous présentons ici les grandes lignes de l'implantation du noyau temps réel SCEPTRE réalisé. Nous commençons par définir les structures de données utilisées pour la matérialisation des objets de base du noyau puis, nous verrons comment ont été implantés les opérations du noyau ainsi que l'ordonnanceur et enfin nous présenterons le manuel d'utilisation de ces opérations et un exemple de programme construit à l'aide des primitives SCEPTRE.

I IMPLANTATION DES OBJETS SCEPTRE

I.1 CONTEXTE D'UNE TACHE

Le contexte d'une tâche contient les informations suivantes :

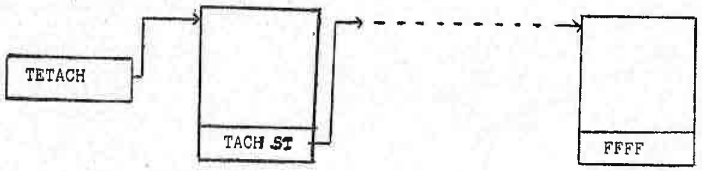


1. ADREX : adresse d'éxécution de la tâche ;
2. PRIOR : priorité de la tâche, de 0 à 5 (0 étant la plus élevée et 5 la plus faible) ;
3. ETATT : état de la tâche (qui est respectivement égal à 1, 2, 3, 4 ou 5 si la tâche est respectivement active, bloquée, prête, inexistante, ou existante).
4. EVT : liste (16 bits) représentant l'état des évènements de la tâche si EVT (I) = 1 l'évènement n°I est dans l'état ARRIVÉ sinon il est dans l'état NON ARRIVÉ.

Le nombre d'évènements d'une tâche est fixé , pour notre réalisation, à 16 évènements numérotés de 0 à 15 ;

5. ATT : liste représentant les évènements attendus par la tâche. Si ATT (I) = 1 l'évènement n° I est attendu par la tâche ;
6. TACHSU : pointeur sur le descripteur d'une tâche se trouvant dans le même état ;
7. STATE : état d'erreur de la tâche (pour les exceptions) ; après invocation d'une opération du noyau, le champ STATE est positionné pour indiquer s'il y a eu erreur , ou non ;
8. les régistres : ils dépendent de la machine cible, ce sont (dans le cas du 6809) les registres A, B, X, Y, U, S, DP, PC et CC.
9. TACHSI : pointeur pour le chaînage de toutes les tâches de l'application.

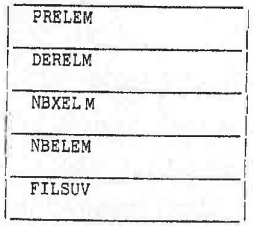
Les tâches d'application sont chainées pour constituer une liste dont la tête est pointée sur TETACH.



Cette liste permet au noyau de détecter si l'utilisateur référence une tâche inexistante lors d'un appel d'une primitive du noyau.

I.2 CONTEXTE D'UNE FILE

Une file est représentée, au niveau du noyau, par un contexte contenant les informations suivantes :



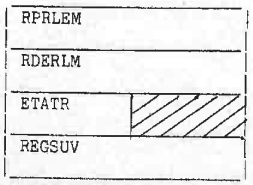
1. PRELEM : pointeur sur le premier élément de la file
2. DERELM : pointeur sur le dernier élément de la file
3. NBXELM : nombre maximum d'éléments que peut contenir la file
4. NBELEM : nombre d'éléments se trouvant dans la file
5. FILSUV : pointeur sur la file suivante.

Les files de l'application sont chainées et constituent une liste pointée par TETFIL.

Cette liste permet au noyau de détecter si l'utilisateur référence une file inexistante lors d'un rappel au noyau.

I.3 CONTEXTE D'UNE REGION

Une région est matérialisée par un contexte contenant les informations suivantes :



1. RPRELM : pointeur sur la première tâche en attente de la libération de la région
2. RDERLM : pointeur sur la dernière tâche en attente de la libération de la région
3. ETATR : état de la région : = 1 si la région est dans l'état LIBRE,
   = 0 si la région est dans l'état OCCUPE
4. REGSUV : pointeur sur la région suivante.

Les régions d'application sont chaînées pour constituer une liste dont la tête est pointée par TETREG.

Cette liste permet au noyau de détecter une erreur si l'utilisateur référence une région inexistante.

II IMPLANTATION DES OPERATIONS SCEPTRE

L'implantation des opérations du noyau suppose que la machine cible offre au moins les possibilités suivantes :
- un mécanisme d'interruptions incluant un changement de contexte partiel d'éxécution d'une tâche à partir d'un signal externe adressé au processeur ;
- des instructions permettant le masquage et le démasquage des interruptions ;
- des instructions d'appel et de retour de sous programmes

Ces trois possibilités sont offertes.

Les deux premières possibilités permettent l'indivisibilité des opérations.

Ainsi toute opération du noyau est programmée comme suit :

```
< masquage des interruptions    >
<opération>
< démasquage des interruptions  >
```

Pour assurer le changement de contexte des tâches, l'appel aux opérations du noyau se fait par l'intermédiaire de l'interruption software (SWI). l'appel à une opération se fait alors comme suit :

```
< préparation des paramètres    >
< registre B = N° de l'opération >
SWI
```

Dans le sous-programme de traitement de l'interruption SWI, on fait un branchement au sous-programme de traitement de l'opération dont le numéro se trouve dans le registre B (cela suppose que les opérations sont numérotées).

Avec cette solution les tâches d'application peuvent être compilées (ou assemblées) séparément du noyau temps réel , Il suffit de connaître les numéros des opérations à appeler et non les adresses dessous-programmes.

Pour faciliter la programmation des appels, nous avons défini des macro-instructions, ce qui évite à l'utilisateur de connaître les numéros des opérations. L'appel précédent devient alors :

< nom-opération > < paramètres >

tous les appels seront explicités par la suite.

L'expérience montre que les applications commencent leur vie en essayant de se "débarrasser" de leurs erreurs (étape de mise au point). C'est pour cela que le noyau assure un certain contrôle et permet de détecter des erreurs.

Celles que nous avons jugées utiles sont résumées dans le tableau ci-après.

Toute erreur détectée par le noyau est mémorisée dans le contexte de la tâche. la tâche pourra connaître le code de l'erreur en invoquant l'opération STATUT .

CODE	DESIGNATION
1	invocation d'une tâche qui n'a pas été créée
2	mode d'exécution $\neq 0$ et $\neq 1$
3	démarrer une tâche qui a déjà été démarrée
4	priorité $\neq 0$ ou $\neq 5$
5	signaler un évènement > 15
6	masque (utilisé dans un EFFACE, ARRIVE ou ATTEND) = 0
7	nombre des éléments d'une file < 0
8	envoi d'un élément dans une file déjà pleine
9	nombre maximum de tâches déjà atteint
10	nombre maximum de files déjà atteint
11	invocation d'une file qui n'a pas été créée
12	situation de blocage en fin d'application
13	demande de suspension d'une tâche non démarrée
14	demande de continuation d'une tâche non suspendue
15	invocation d'une région non déclarée

REMARQUE 1
Il est bon de détecter les erreurs causées par les appels au noyau mais une fois que la mise au point est achevée toute la partie contrôle d'erreurs ne sert plus à rien, et, dans une implantation réelle, elle occupe de la place mémoire et consomme du temps machine.

Comme les facteurs encombrement mémoire et temps d'exécution sont précieux pour les applications en temps réel, nous préconisons l'utilisation de deux versions de noyau : l'une avec contrôle pendant la phase de mise au point, l'autre sans controle pendant l'exploitation réelle de l'application ; ceci est d'autant plus intéressant si la phase de mise au point a écarté toutes les erreurs potentielles . Nous remarquons, aussi, que l'utilisation de deux versions d'un même logiciel est valable pour tout autre composante d'un système informatique destiné à commander un environnement, en temps réel.

REMARQUE 2
Dans le projet SCEPTRE les opérations de déclaration d'objets ne sont pas définies.

C'est pour cela que nous avons rajouté trois opérations :

DCLTSK : pour la déclaration d'une tâche
DCLFIL : pour la déclaration d'une file
DCLREG : pour la déclaration d'une région.

Ainsi les opérations d'initialisation et de gestion des contextes associés à ces objets sont sous la responsabilité du noyau et ne sont pas connues des agences (on pourra alors changer d'implantation des contextes sans toucher aux agences -concept de type abstrait- /DERNIAME, 79/).

III IMPLANTATION DE L'ORDONNANCEUR

Fonctions
Pour permettre de réaliser les opérations SCEPTRE, l'ordonnanceur est décomposé en 3 fonctions :
- choix d'une tâche prête pour l'attribution du processeur (l'algorithme implanté choisit toujours la tâche la plus prioritaire et la plus ancienne dans l'état prêt) ;
- ajoût d'une tâche à l'ensemble des tâches prêtes ;
- retrait d'une tâche de l'ensemble des tâches prêtes.

Ces fonctions ont été réalisées par des procédures.

Listes gérées par l'ordonnanceur
L'ordonnanceur dispose d'une liste par priorité (donc 6 listes) qu'il gère par l'intermédiaire des procédures de choix, d'ajoût et de retrait des tâches prêtes.

Invocation de l'ordonnanceur
L'ordonnanceur est invoqué :
- soit par les opérations DEMARRER, SIGNALER, CONTINUER et CHANGER-PRIORITE pour modifier les listes des tâches prêtes ;
- soit par les opérations ATTENDRE, SETERMINER et ARRETER pour attribuer le processeur.

Types d'ordonnancement
L'ordonnanceur peut être avec ou sans préemption.

IV MODE D'APPEL DES OPERATIONS DU NOYAU

Pour des raisons de programmation, essentiellement la limitation de la taille des identificateurs à 6 caractères, par l'assembleur du THEMIS -THOMSON -EFCIS- sur lequel nous avons réalisé notre noyau, nous avons abrégé les noms des opérations définies dans le projet SCEPTRE.

Pour chaque opération nous donnerons la fonction SCEPTRE correspondante et le mode d'appel de cette opération.

Pour plus de détails sur le mode d'utilisation du noyau, consulter le manuel d'utilisation présenté dans /MAMMERI, 84 A/.

Notation

V1, V2 : valeur immédiate sur un octet, ou adresse d'un octet ;
W 1 : valeur immédiate sur 2 octets ou adresse de 2 octets consécutifs.

1) Opération INITIA

* Fonction
Il n'y a pas d'opération de SCEPTRE qui correspond à cette opération. L'opération INITIApermet d'initialiser les variables du noyau et de choisir le mode d'éxécution (préemptible ou non).

Cette opération est la première opération que l'utilisateur doit appeler pour accéder aux services du noyau.

* Appel
INITIA : V1, V2, DESC
V1 : mode d'exécution (=1 pour préemptible, = 0 pour non préemptible)
V2 : priorité de la tâche mère de l'application
DESC : adresse où sera rangée l'adresse du descripteur de la tâche mère de l'application.

2) Opération DCLTSK

* Fonction
Il n'y a pas d'opération de SCEPTRE qui correspond à cette opération. L'opération DCLTSK permet de déclarer une tâche en lui créant un contexte.

* Appel
DCLTSK V1, A1, DESC
V1 : priorité de la tâche à déclarer,
A1 : adresse de début d'exécution de cette tâche,
DESC : adresse où sera rangée l'adresse du descripteur après création du contexte.

3) Opération TACOUR

* Fonction
Cette opération correspond à l'opération TACHE-COURANTE de SCEPTRE. Elle permet de rendre dans le registre X l'adresse du descripteur de la tâche courante.

* Appel
TACOUR
Sans paramètres.

4) Opération SETERM
* Fonction
L'opération SETERM correspond à l'opération SETERMINER de SCEPTRE indique la fin normale d'exécution d'une tâche.

* Appel
SETERM
Sans paramètres.
5) Opération DEMARE

* Fonction
L 'opération DEMARE correspond à l'opération DEMARRER de SCEPTRE. Elle permet de mettre une tâche dans l'état PRET.

* Appel
DEMARE DESC
DESC : adresse du descriptaur de la tâche à démarrer

6) Opération CHNGPR

* Fonction
L'opération CHNGPR correspond à l'opération CHANGER-PRIORITE de SCEPTRE.

* Appel
CHNGPR V1, DESC
V1 : nouvelle priorité,
DESC : adresse du descripteur de la tâche dont on veut changer la priorité.

7) Opération PRIORT

* Fonction
L'opération PRIORT correspond à l'opération PRIORITE de SCEPTRE. Elle permet de rendre dans le registre A la priorité de la tâche donnée en paramètre.

* Appel
PRIORT DESC
DESC : adresse du descripteur de la tâche dont on veut connaître la priorité.

8) Opération ETAT

* Fonction
L'opération ETAT correspond à l'opération ETAT de SCEPTRE. Elle permet de rendre dans le registre A l'état de la tâche donnée en paramètre :
A = 1 si tâche active, = 3 si tâche PRETE,
= 2 si tâche bloquée, = 4 si tâche inexistante,
= 6 si tâche suspendue, = 5 si tâche existante.

* Appel
ETAT DESC
DESC : adresse du descripteur de la tâche dont on veut connaître l'état.

9) Opération SUSPND

* Fonction
L'opération SUSPND correspond à l'opération ARRETER de SCEPTRE.

* Appel
SUSPND DESC
DESC : adresse du descripteur de la tâche dont on veut suspendre l'exécution.

10) Opération CONTIN

* Fonction
L'opération CONTIN correspond à l'opération CONTINUER de SCEPTRE.

* Appel
CONTIN DESC
DESC : adresse du descripteur de la tâche dont on veut reprendre l'éxécution.

11) Opération EFFACE

* Fonction
L'opération EFFACE correspond à l'opération EFFACER de SCEPTRE.

* Appel
EFFACE W1
W1: masque utilisé
Pour I ou 0 à 15 :
Si bit I de W1 = 1 alors l'évènement I est à effacer,
Si bit I de W1 = 0 alors l'évènement I reste dans le même état.

12) Opération ARRIVE

* Fonction
L'opération ARRIVE correspond au prédicat ARRIVE de SCEPTRE• elle rend dans A la valeur du prédicat.

* Appel
ARRIVE W1
W1 : masque utilisé
Pour I de 0 à 15 :
Si bit I de W1 = 1 alors l'état de l'évènement I est testé
Si bit I de W1 = 0 alors l'état de l'évènement I n'est pas testé.

13) Opération SIGNAL

* Fonction
L'opération SIGNAL correspond à l'opération SIGNALER de SECPTRE.

* Appel
SIGNAL V1, DESC
V1 : numéro de l'évènement à signaler
DESC : adresse du descripteur de la tâche pour laquelle l'évènement sera signalé.

14) Opération ATTEND

* Fonction
L'opération ATTEND correspond à l'opération ATTENDRE de SCEPTRE.

* Appel
ATTEND W1
W1 : masque utilisé
Pour I de 0 à 15 :
Si bit I de W1 = 1 alors l'évènement I sera attendu
Si bit I de W1 = 0 alors l'évènement I ne sera pas attendu.

15) Opération DCLFIL

* Fonction
Il n'y a pas d'opération de SCEPTRE qui correspond à cette opération. L'opération DCLFIL permet de créer le contexte d'une file.

* Appel
DCLFIL W1, A2
W1 : nombre maximum des éléments de la file
A1 : adresse où sera rangée l'adresse du descripteur de la file à créer.

16) Opération VIDE

* Fonction
L'opération VIDE correspond au prédicat VIDE de SCEPTRE. Elle rend, dans le registre A, la valeur du prédicat.

* Appel
VIDE DESCF
DESF : adresse du descripteur de la file testée.

17) Opération PLEINE

* Fonction
L'opération PLEINE correspond au prédicat PLEINE de SCEPTRE. Elle rend, dans le registre A, la valeur du prédicat.

* Appel
PEINE DESCF
DESCF : adresse du descripteur de la file testée.

18) Opération ENVOIE

* Fonction
L'opération ENVOIE correspond à l'opération ENVOYER de SCEPTRE.

* Appel
Envoie A1, A2
A1 : adresse du descripteur de la file à laquelle on veut ajouter un élément ;
A2 : adresse d'un bloc de 4 octets (repésentant le descripteur de l'élément à mettre dans la file) organisés comme suit :
A2, A2+1 : adresse où se trouve le contenu de l'élèment,
A2+2, A2+3 : quelconque mais à réserver obligatoirement car ces 2 octets servent pour le chaînage.

19) Opération RETIRE

* Fonction
L'opération RETIRE correspond à l'opération RETIRER de SCEPTRE.
Elle rend dans le registre X :
- l'adresse du descripteur du premier élément de la file, si celle-ci n'est pas vide.
- la valeur $FFFF, si la file est vide.

* Appel
RETIRE DESCF
DESCF : adresse du descripteur de la file à partir de laquelle on veut retirer un élément.

20) Opération ENTREE

* Fonction
Cette opération correspond à l'opération ENTRER de SCEPTRE.

* Appel
ENTREE A1
A1 : adresse du descripteur de la région à occuper.

21) Opération SORTIE

* Fonction
Cette opération correspond à l'opération SORTIR de SCEPTRE.

* Appel
SORTIE DESCR
DESCR : adresse du descripteur de la région à libérer.

22) Opération DCLREG

* Fonction
Cette opération n'existe pas dans SCEPTRE. Elle permet de déclarer un contexte pour un objet de type REGION.
Au retour, le registre X contient l'adresse du descripteur de la région déclarée.

* Appel
DCLREG
Sans paramètre

23) Opération STATUT

* Fonction
Elle correspond à l'opération STATUT de SCEPTRE. Elle rend, dans le registre A, le code d'erreur (l'erreur étant celle provoquée par l'appel qui a précédé l'opération STATUT).

Les codes des erreurs signalées par le noyau sont résumés dans le tableau, paragraphe II.

* Appel
STATUT
Sans paramètre

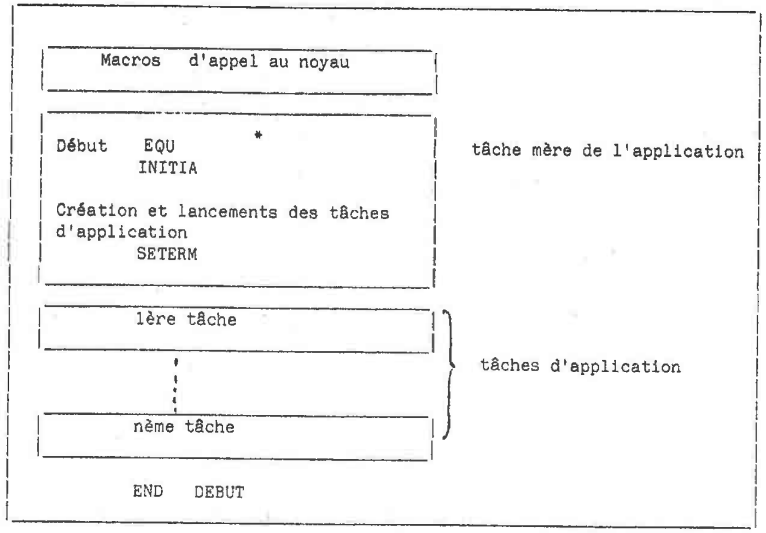
V EXEMPLE D'UTILISATION DU NOYAU SCEPTRE

1) Structure d'une application utilisant le noyau SCEPTRE

L'exécution d'une application, faisant référence au noyau SCEPTRE, commence dans le code de l'application. Nous appelons "tâche MERE de l'application" le code de l'application dans lequel celle-ci déclare les tâches qui la composent.

Le noyau réserve automatiquement un contexte pour cette tâche et par l'intermédiaire de l'opération INITIA (opération que nous avons rajoutée) l'application demande l'initialisation de ce contexte.

Ainsi tout programme utilisant le noyau réalisé devra avoir la structure suivante :



2) Exemple

```
                        *-------------------------------------------------*
                        * EXEMPLE D UTILISATION DU NOYAU SCEPTRE REALISE'.  *
                        * ON A CHOISI LE MODELE DU PRODUCTEUR CONSOMMATEUR. *
                        *DANS CET EXEMPLE LE PRODUCTEUR PRODUIT 10 MESSAGES *
                        * QU IL DEPOSE UN PAR UN DANS UNE FILE A LAQUELLE  *
                        * ACCEDE LE CONSOMMATEUR ;                         *
                        * LE CONSOMMATEUR SUSPEND LE PRODUCTEUR APRES AVOIR *
                        * PRELEVE' LES 10 MESSAGES PUIS DEMANDE LA FIN DE  *
                        * L APPLICATION EN REVEILLANT LA TACHE MERE         *
                        *-------------------------------------------------*
00159A 3000     9999    A ELEMD1 FDB    $9999
00160A 3002     3000    A ELEM1  FDB    ELEMD1,0
00161A 3006     0A      A CPT    FCB    10
00162A 3007     0000    A DESCF  FDB    0
00163A 3009     3000    A DESCT0 FDB    0
00164A 300B     2000    A DESCT1 FDB    0
00165A 300D     3000    A DESCT2 FDB    0
                        *-------------------------------------*
                        * MODELE DU PRODUCTEUR-CONSOMMATEUR *
                        *-------------------------------------*
                300F    A DEBUT  EQU    *
00170A 300F 86  30      A        LDA    #$30
00171A 3011 1F  9A      A        TFR    A,CC
00172A 3013 86  00      A        LDA    #0
00173A 3015 1F  8B      A        TFR    A,DP
                        *  DECLARATION DU PRODUCTEUR ET DU CONSOMMATEUR
00175A 3017                      INITIA #0,#4,DESCT0
00176A 3033                      DCLTSK #2,#TACH1,DESCT1
00177A 302E                      DCLTSK #3,#TACH2,DESCT2
                        * LANCEMENT DES DEUX TACHES
00179A 3039                      DEMARE DESCT1
00180A 303F                      DEMARE DESCT2
                        * ATTENDRE LA FIN DES DEUX TACHES
00182A 3045                      ATTEND #1
00183A 304C                      SETERM
                        *-----------------*
                        * PRODUCTEUR       *
                        *-----------------*
                304F    A TACH1  EQU    *
00188A 304F                      DCLFIL #1,DESCF
00189A 3059 9F  3007    A        STX    DESCF
                305C    A BOUCLP EQU    *
00191A 305C                      EFFACE #1
00192A 3063                      ATTEND #1
00193A 306A                      ENVOIE DESCF,ELEM1
00194A 3074                      SIGNAL #0,DESCT2
00195A 3076 20  DE   305C        BRA    BOUCLP
                        *----------------*
                        * CONSOMMATEUR  *
                        *----------------*
                307E    A TACH2  EQU    *
                307E    A BOUCLC EQU    *
00201A 307E                      EFFACE #1
00202A 3085                      ATTEND #1
00203A 308C                      RETIRE DESCF
00204A 3092 B6  3006    A        LDA    CPT
00205A 3095 4A          A        DECA
00206A 3096 B7  3006    A        STA    CPT
00207A 3099 81  00      A        CMPA   #0
00208A 309B 27  02   309F        BEQ    FCONSS
00209A 309D 20  DF   307E        BRA    BOUCLC
                309F    A FCONSS EQU    *
                        * SUSPENDRE LE PRODUCTEUR
00212A 309F                      SUSPND DESCT1
                        * INDIQUER LA FIN D EXECUTION
00214A 30A5                      SIGNAL #0,DESCT0
00215A 30A0                      SETCOM
                3080    A FINAPP EQU    *
                300F    A        END    DEBUT
TOTAL ERRORS 00000--00000
TOTAL WARNINGS 00000--00000
```


ANNEXE B2

IMPLANTATION ET UTILISATION DE L'AGENCE HORLOGERIE

Nous présentons ,ici, l'implantation de l'agence horlogerie et son utilisation par une tâche d'application.

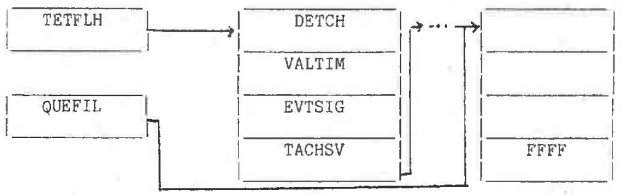
I STRUCTURES DE DONNEES UTILISEES

I.1 COMPTEURS ASSOCIES A L'HORLOGE LOCALE

L'implantation de l'horloge locale nécessite l'utilisation de 4 compteurs pour gérer l'heure, la minute, la seconde et la fraction de seconde (la fraction retenue est 1/50).

I.2 FILE DES DEMANDES DE DELAI (ET TEMPORISATION)

Afin de mémoriser l'ensemble des demandes de délai,l'agence gère une file pointée par TETFLH et ayant des élèments composés des champs suivants :
- DETCH : descripteur de la tâche ayant fait la demande de délai,
- EVTSIG : n° de l'évènement à signaler à la fin du délai,
- VALTIM : valeur actuelle du délai,
- TACHSV : pointeur sur la demande suivante .



II IMPLANTATION D'UNE HORLOGE PHYSIQUE

Pour réaliser l'agence horlogerie, nous avons utilisé un "timer" d'un des trois VIA disponibles sur la carte MPU9 /EFCIS/ (la présentation du VIA et de la carte MPU9 est donnée dans la partie étude de la connexion au réseau local GIXINET).

Pour obtenir une horloge physique il a fallu :

1) choisir les registres :
- le registre IFR (interrupt flag register) et IER (interrupt enable register) pour provoquer une interruption du microprocesseur 6809 via l'entrée IRQ.
- le registre ACR (auxiliary control register) pour contrôler l'interruption en provenance du "timer" T1.
- T1-C (T1 low-order latch) et T1C-H (T1 high order counter) qui sont 2 registres de 8 bits chacun et qui constituent un compteur de 16 bits. Ce compteur est décrémenté à chaque top du quartz (1 MHZ), le passage à zéro de ce compteur provoque une interruption mémorisée dans IFR et qui sera, par la suite, traitée par le processeur.

2) initialiser correctement les registres :
- IFR : valeur quelconque ;
- IER = $40 pour autoriser l'IT timer T1 ;

- ACR = $40 après chaque passage par zéro du compteur composé de T1C-H et T1L-L, il y a chargement automatique de ce compteur et la décrémentation continue ;
- T1C-H et T1C-L constituent la valeur initiale du compteur pour obtenir une fréquence de 50 Hz le compteur est initialisé à 2000 (puisque la fréquence du quartz associé au VIA est de 1 MHZ).

REMARQUE

le nombre de demandes de délai et de temporisation et le choix de la fréquence de l'horloge locale influent considérablement sur le temps machine réellement affecté aux tâches d'application.

Nous démontrons ceci, en utilisant des calculs assez approximatifs, soit :
N : le nombre moyen de demandes de délai ou de temporisation,
L : nombre moyen d'instructions pour le traitement d'une de ces demandes,
X : temps moyen d'exécution d'une instruction (X ≃ 4µs),
T : temps approximatif d'exécution de la tâche immédiate,
F : fréquence de l'horloge logique,
R : taux d'occupation de l'unité centrale par la tâche immédiate,

alors :

$T \simeq L * N * X$

$R \simeq F * T * 10^{-6}$ (fréquence quartz = 1MHZ).

Ainsi, si on ne choisit pas correctement F et N (L quant à lui fixé une fois la programmation terminée) on risque de n'utiliser le processeur que pour la tâche immédiate horlogerie.

III IMPLANTATION DES OPERATIONS DE L'AGENCE

Les opérations de l'agence horlogerie sont réalisées par des procédures. Pour que ces opérations soient indivisibles, il y a masquage des interruptions, au début de chaque opération, et démasquage, à la fin de l'opération, de ces interruptions.

Ces procédures permettent de détecter deux types d'erreurs :
- valeur de délai négative,
- numéro d'évènement supérieur à 15,
- valeurs initiales des compteurs de l'horloge hors des bornes (ex. heure >23).

S'il y a une erreur, au moment de l'invocation de l'agence horlogerie, l'appel n'est pas pris en compte.

IV IMPLANTATION DE LA TACHE IMMEDIATE

Chaque fois que le compteur du timer T1 (voir paragraphe II) passe à zéro, il y a génération d'une interruption IRQ.

A la prise en compte de cette interruption, par le processeur (le 6809), la tâche immédiate horlogerie est lancée.

A l'initialisation de l'application, la tâche immédiate horlogerie est créee, par l'agence horlogerie, et elle ne peut prendre en compte les interruptions horloge physique que si l'utilisateur a initialisé l'horloge locale par l'opération INIHOR.

V MODE D'APPEL DES PRIMITIVES DE L'AGENCE HORLOGERIE

Comme pour le noyau temps réel, les noms des opérations de l'agence horlogerie ont été limités à 6 caractères.

Nous précisons, ci-après, la manière dont on utilise les services de l'agence horlogerie réalisée. Pour faciliter la programmation en assembleur, ces opérations sont appelées par l'intermédiaire de macro-instructions.

1) Opération DELAI

* Fonction
L'opération DELAI permet, à une tâche, de demander un délai et d'être avertie une fois que ce délai vient à échéance. La tâche appelante n'est pas bloquée.

* Appel
DELAI : W1, V2 ;
W1 : période (valeur sur 2 octets) en unités de temps,
V2 : n° de l'évènement à signaler à la fin du délai

2) Opération INIHOR

* Fonction
l'opération INHIOR permet d'initialiser les compteurs de l'horloge locale à une valeur exprimée en heures, minutes, secondes et fractions de seconde (1/50).

* Appel
INIHOR V1, V2, V3, V4
V1 : valeur de l'heure initiale (1 octet)
V2 : valeur de la minute initiale (1 octet)
V3 : valeur de la seconde initiale (1 octet)
V4 : valeur de la fraction de seconde initiale (1 octet)

3) Opération HORLOC

* Fonction
L'opération HORLOC permet à une tâche de connaître la valeur des différents compteurs de l'horloge locale. Elle rend à la tâche appelante :
dans l'octet poids fort de x l'heure
dans l'octet poids faible de X la minute
dans l'octet poids fort de Y la seconde et
dans l'octet poids faible de Y la fraction de seconde.

* Appel
HORLOC

4) Opération TEMPOR

* Fonction
Demande de temporisation (demande de délai + blocage de la tâche)

* Appel
TEMPOR W1
W1 : durée d'attente.

Remarque : pour bloquer la tâche appelante, l'agence horlogerie utilise toujours l'énèvement 0 de cette tâche.

5) Opération LIRETM

* Fonction
L'opération LIRETM permet à une tâche de connaître le temps restant à la fin d'un délai. Le registre Y contient ce temps.

* Appel.
LIRETM.

6) Opération ANULTM

* Fonction
L'opération ANULTM permet à une tâche d'annuler une demande de délai.

* Appel
ANULTM

Exemple

```
00149                          *-------------------------------------*
00150                          * EXEMPLE D UTILISATION DE L AGENCE    *
00151                          * HORLOGERIE .                         *
00152                          * L EXEMPLE CHOISI EST CELUI DU        *
00153                          * PRODUCTEUR  CONSOMMATEUR             *
00154                          * LE PRODUCTEUR PRODUIT UN MESSAGE TOUTES*
00155                          * LES 500 FRACTIONS DE SECONDE(1/50) QU' *
00156                          * IL DEPOSE DANS UNE FILE AU CONSOMMATEUR*
00157                          *-------------------------------------*
00158A 3000      9999      A ELEM01 FDB    $9999
00159A 3002      3000      A ELEM1  FDB    ELEM01,0
00160A 3006      0A        A CPT    FCB    10
00161A 3007      0000      A DESCF  FDB    0
00162A 3009      0000      A DESCT0 FDB    0
00163A 300B      0000      A DESCT1 FDB    0
00164A 300D      0000      A DESCT2 FDB    0
00165                          *-------------------------------------*
00166                          * MODELE DU PRODUCTEUR-CONSOMMATEUR    *
00167                          *-------------------------------------*
00168            300F      A DEBUT  EQU    *
00169A 300F 86   80        A        LDA    #$80
00170A 3011 1F   8A        A        TFR    A,CC
00171A 3013 86   00        A        LDA    #0
00172A 3015 1F   8B        A        TFR    A,DP
00173                          *  DECLARATION DU PRODUCTEUR ET DU CONSOMMATEUR
00174A 3017                           INITIA #0,#4,DESCT0
00175                          * INITIALISATION DE L HORLOGE
00176A 3023                           INIHOR #11,#11,#11,#11
00177A 3032                           OCLTSK #2,#TACH1,DESCT1
00178A 3030                           OCLTSK #3,#TACH2,DESCT2
00179                          * LANCEMENT DES DEUX TACHES
00180A 3048                           DEMARE DESCT1
00181A 304E                           DEMARE DESCT2
00182                          * ATTENDRE LA FIN DES DEUX TACHES
00183A 3054                           ATTEND #1
00184A 3058                           LIRETM
00185A 305E                           SETERM
00186                          *-------------------*
00187                          * PRODUCTEUR        *
00188                          *-------------------*
00189            3061      A TACH1  EQU    *
00190A 3061                           OCLFIL #1,DESCF
00191A 3068 BF   3007      A        STX    DESCF
00192            306E      A BOUCLP EQU    *
00193A 306E                           DELAI  #500,#0
00194A 3077                           EFFACE #1
00195A 307E                           ATTEND #1
00196A 3085                           ENVOIE DESCF,ELEM1
00197A 308F                           SIGNAL #0,DESCT2
00198A 3097 20   05   306E           BRA    BOUCLP
00199                          *---------------*
00200                          * CONSOMMATEUR  *
00201                          *---------------*
00202            3099      A TACH2  EQU    *
00203            3099      A BOUCLC EQU    *
00204A 3099                           EFFACE #1
00205A 30A0                           ATTEND #1
00206A 30A7                           RETIRE DESCF
00207A 30AD B6   3006      A        LDA    CPT
00208A 30B0 4A                        DECA
00209A 30B1 B7   3006      A        STA    CPT
00210A 30B4 81   00        A        CMPA   #0
00211A 30B6 27   02   30BA           BEQ    FCONSS
00212A 30B8 20   0F   3099           BRA    BOUCLC
00213            30BA      A FCONSS EQU    *
00214                          * SUSPENDRE LE PRODUCTEUR
00215A 30BA                           SUSPND DESCT1
00216                          * INDIQUER LA FIN D EXECUTION
00217A 30C0                           SIGNAL #0,DESCT0
00218A 30C8                           SETERM
00219
00220            300F      A        END    DEBUT
TOTAL ERRORS 00000--00000
```

ANNEXE B3

CONNEXION AU RESEAU LOCAL GIXINET ET UTILISATION DE CETTE CONNEXION

Nous présentons ici, la connexion d'un microordinateur 6809 de MOTOROLA au réseau local GIXINET. Cette connexion servira comme support pour la réalisation d'une station de transport.

Nous commencerons par présenter le matériel utilisé puis nous verrons la connexion elle-même et enfin nous présenterons la manière d'utiliser, par les tâches des couches supérieures, cette connexion.

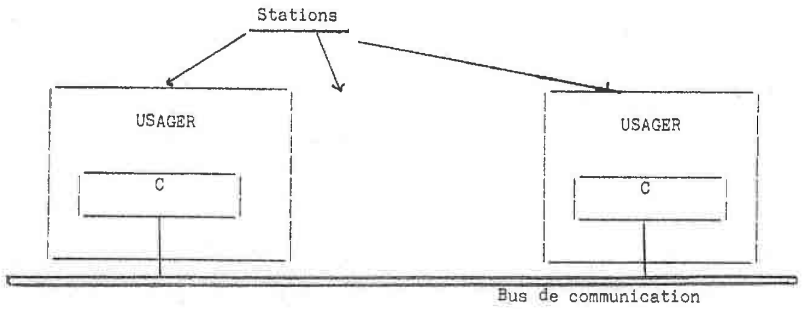
I PRESENTATION DU MATERIEL

I.1. PRESENTATION DU RESEAU LOCAL GIXINET

GIXINET est un réseau de communication destiné à relier des matériels informatiques hétérogènes.

La structure la plus générale du réseau comporte un support de communication organisé en bus, sur lequel des équipements viennent se greffer à travers la carte du contrôleur intelligent TR 81 A /GIXI, 81/.

En appelant USAGER tout équipement relié au réseau, par un communicateur TR 81 A, et STATION l'ensemble "Usager-communicateur", le réseau GIXINET se schématise ainsi :



C : carte TR81 A

1) Fonctionnement du réseau GIXINET

L'échange des données entre stations s'effectue via des mémoires-tampons implantées sur les communicateurs.

Cet échange se fait selon l'algorithme TABLE-RONDE /NAFFAH, 78/ caractérisé par :
- la circulation d'un droit à émettre ;
- une gestion sans élément central ni station privilégiée ;
- la détection et la correction des erreurs de transmission ;
- le contrôle du flux de données ;
- la gestion de la configuration du réseau.

Disposant de ces possibilités, le réseau GIXINET ne laisse, à la charge de l'usager, qu'un minimum de tâches qui concernent l'échange de données proprement dites et la supervision du réseau. Le service offert par GIXINET est conforme à la définition LIGNE du modèle de référence de l'ISO pour l'interconnexion des systèmes ouverts.

Le raccordement d'une station au réseau s'effectue par une liaison de type parallèle autorisant un transfert au rythme d'un mégaoctets/seconde.

Cette liaison est gérée, du côté réseau, par le communicateur.

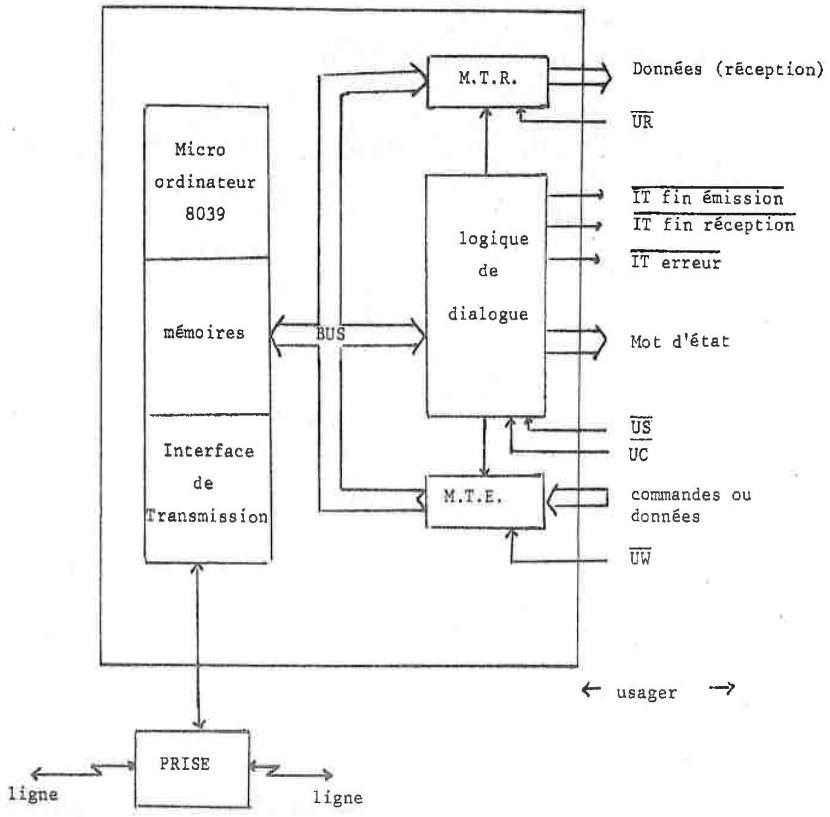
2) Caractéristiques techniques du réseau GIXINET
- vitesse de transmission = 125 K bits/s,
- nombre de stations configurables : 64,
- débit utilisé supérieur à 10 K octets/s,
- transmission transparente de trames de longueurs variables (max. 1021 octets),
- fonctions de diffusion à toutes les stations.

3) Architecture du communicateur TR 81 A

Les principales fonctions du TR 81 A sont :
- la détection de début de messages et de l'absence de transmission ;
- le passage du droit à émettre ;
- réception des données et rangement de ces données dans la mémoire tampon de réception (MTR) ;
- émission des données à partir de la mémoire tampon d'émission (MTE).

Le dialogue entre l'usager et le communicateur est assuré par les organes suivants (voir figure ci-après) :
- une mémoire tampon d'émission (MTE),
- une mémoire tampon de réception (MTR),
- trois interruptions (fin d'émission, fin de réception, erreur),
- une logique qui règle les accès aux tampons et fournit à l'usager toutes les informations sur l'état du communicateur via un octet de mot d'état. Cette logique reçoit et exécute les commandes de l'usager (demande de connexion au réseau, demande d'émission, demande de réception, demande de déconnexion du communicateur).

- Les indicateurs :
UR : signal de lecture de la MTR,
UW : signal d'écriture dans la MTE,
US : validation du mot d'état,
UC : validation des commandes usager.



Synoptique du TR 81 A

4) Mode d'emploi des services du communicateur

Vis-à-vis du réseau, le communicateur est, soit actif, soit passif.
- passif : il ne joue aucun rôle dans les transmissions sur le réseau ;
- actif : il participe aux échanges.

La notion d'état passif ou actif est importante car elle détermine quelles opérations ont un sens et quand.

Les commandes en mode actif sont :
- la restauration du communicateur (commande 6) ;
- les connexion et déconnexion du communicateur (commandes 4 et 5) ;
- l'envoi et la réception de messages (utilisation des commandes 0, 1, 2, et 3).

Les commandes en mode passif sont :
- le test par écho des mémoires tampons,
- changement/consultation des paramètres du communicateur (parmi les paramètres, on peut citer les adresses physiques des stations, les compteurs intervenant dans les reprises sur erreur de transmission).

L'utilisation des trois interrruptions (fin d'émission, fin de réception, erreur de transmission (ie problème au niveau du communicateur)) permet un asynchronisme entre le traitement correspondant à une transmission d'un message et le reste de l'application.

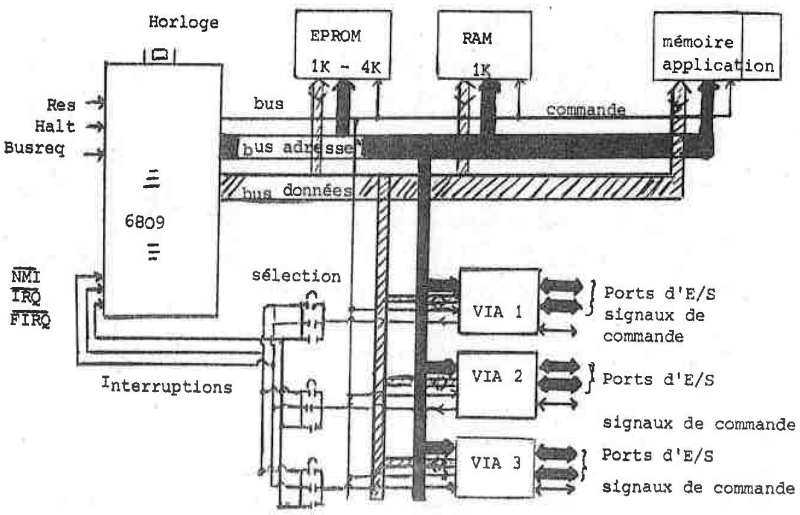
I.2 PRESENTATION DE LA CARTE MPU9 D'EFCIS

Chaque station du réseau local est consituée d'un micro-ordinateur ayant comme structure de base la carte MPU 9 d'EFCIS /EFCIS/.

Le module MPU 9 offre un système complet dont la structure peut être étendue grâce aux différents modules mémoires ou périphériques de la famille EUROPE.

Ce module est constitué (figure suivante) essentiellement de :
- une unité centrale (6809 de MOTOROLA) ;
- une zone mémoire RAM de 1 K octets ;
- 2 supports de boîtiers pouvant supporter des EPROM d'une capacité d'1, de 2 ou 4 K octets ;
- trois circuits d'interfaçage VIA.

Les caractéristiques des circuits 6809 et VIA sont résumées dans ce qui suit à titre de rappel (se référer éventuellement à /EFICS/ et /SYNTERNEK/ pour une étude plus détaillée de ces deux circuits



SYNOPTIQUE DU MODULE MPU9

1) Présentation du microprocesseur 6809

* Généralités

C'est un microprocesseur de la famille MOTOROLA, 8 bits, compatible avec toute la famille 6800, ayant pour caractéristiques essentielles :
- 6 possibilités d'interruption, dont 3 par logiciel ;
- possibilité de DMA ;
- 10 modes d'adressage ;
- UAL permettant des opérations sur 8 et 16 bits ;
- multiplication 8 bits ;
- 59 codes mnémoniques :
- possibilité d'adresser 64 K octets ;
- 9 registres (2 registres d'index, 2 pointeurs de pile, un registre de pages directes, 2 accumulateurs 8 bits pouvant être concaténés en un accumulateur 16 bits, un registre d'état (8 bits) et un compteur ordinal de 16 bits).

2) Présentation du circuit d'interfaçage VIA

* Généralités

Le circuit SY 6522 du SYNERTEK coorporation ou VIA (versatile interface adapter) est caractérisé par :
- deux ports d'entrée/sortie de 8 bits chacun. Le sens de ces ports est déterminé par la configuration de deux ports de direction (DDRA et DDRB);
- deux timers de 16 bits ;
- génération d'une interruption ;
- des signaux de contrôle (CA1, CA2, CB1, CB2).

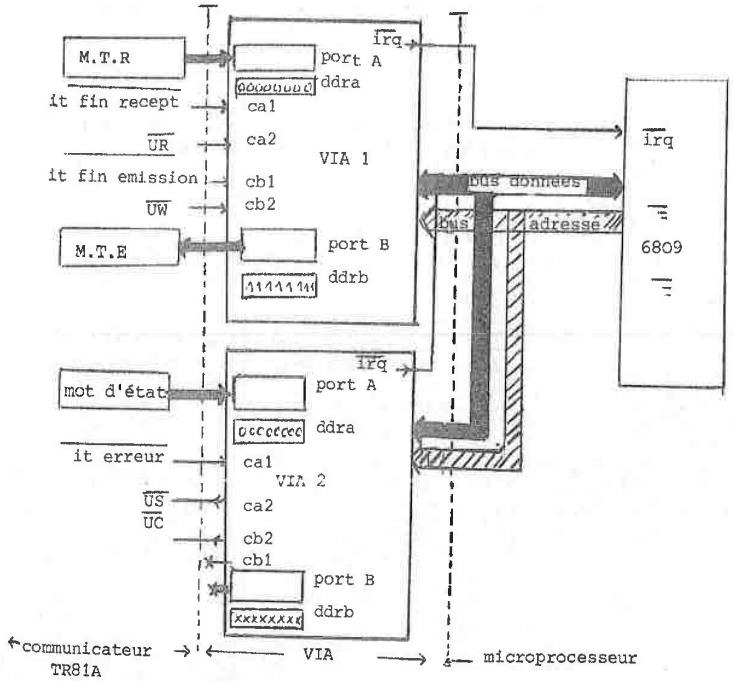
II CONNEXION DU COMMUNICATEUR TR 81 A AU MC 6809

La première phase de réalisation du réseau local est la connexion du communicateur TR 81 A au microprocesseur 6809. Ce paragraphe présente de manière succincte cette connexion.

Pour réaliser la connexion TR 81 A - 6809 nous avons utilisé deux des trois VIA disponibles sur la MPU 9.

Utilisation des VIA

La figure ci-après illustre la connexion logique du TR 81 A au microprocesseur 6809 par l'intermédiaire des 2 VIA.



CONNEXION COMMUNICATEUR-MICROPROCESSEUR

1) Utilisation du 1er VIA

a) connexion de la mémoire tampon de réception
La MTR (8 bits en sortie du TR 81 A) est reliée au port A. Le registre de direction DDRA contient alors la valeur hexadécimale $00 ;

b) connexion de la mémoire tampon d'émission
La MTE (8 bits en entrée du TR 81 A) est reliée au port B. Le registre de direction DDRB contient la valeur $FF ;

c) connexion de l'interruption de fin de réception à l'entrée CA1 ;
d) connexion de l'interruption de fin d'émission à l'entrée CB1 ;
e) connexion du signal de lecture de la MTR (UR) à CA2 ;
f) connexion du signal d'écriture (UW) à CB2.

2) Utilisation du 2ème VIA

a) connexion du mot d'état (8 bits en sortie du TR 81 A) au port A. Le registre de direction DDRA contient la valeur $00 ;

b) connexion de l'interruption erreur à l'entrée CA1 ;

c) connexion du signal de validation du mot d'état (US) à CA2 ;

d) connexion du signal de validation des commandes de l'usager (UC) à CB2.

III GESTION DU COMMUNICATEUR

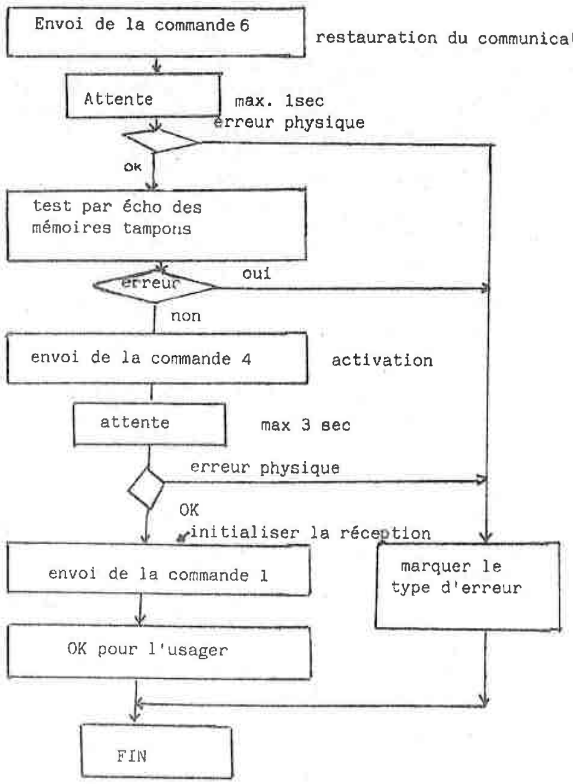
Une fois la connexion physique terminée, il a fallu écrire un logiciel de gestion de cette connexion.

Ce logiciel assure les fonctions suivantes :
- la connexion logique du communicateur ,
- l'obtention de l'état du communicateur,
- la déconnexion du communicateur,
- l'émission d'un bloc,
- l'obtention du résultat d'une émission,
- la réception d'un message,
- la libération du communicateur,
- et l'avertissement de l'utilisateur en cas de fin d'émission, de fin de réception ou d'erreur du communicateur.

1) Connexion logique du communicateur : opération OUVRGX

Il s'agit de la connexion logique du communicateur au réseau lui permettant de participer aux envois et réceptionsde messages.

En cas d'erreur la connexion ne peut être établie. Nous donnons à l'aide de l'organigramme suivant la démarche suivie pour établir la connexion logique du communicateur.



L'utilisateur active le communicateur en appelant l'opération OUVRGX (sans paramètre).

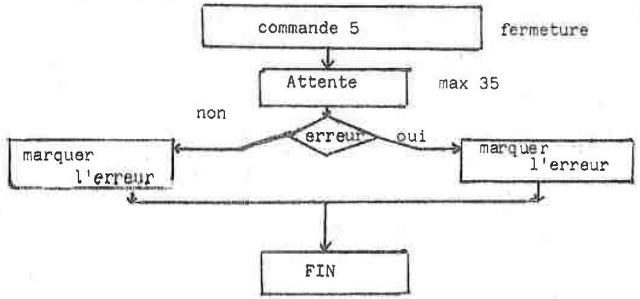
2) Etat du communicateur : opération ETATGX

L'utilisateur, après chaque opération, peut connaître, s'il le désire, l'état du communicateur en appelant l'opération ETATGX. Cette opération rend, dans le registre A, un code-état (l'interprétation de ce code est donnée dans /GIXI, 81/).

3) Deconnexion du communicateur : opération FERMGX

A la fin de transfert, l'utilisateur demande la déconnexion du communicateur. Le communicateur ne jouera alors plus aucun rôle dans les échanges, sauf celui de répéteur.

La déconnexion se fait via l'opérateur FERMGX que nous décrivons grâce à l'organigramme suivant :



L'opération FERMGX est sans paramètre .

4) Emission d'un bloc : opération EMETGX

L'utilisateur peut émettre un bloc d'informations en appelant l'opération EMETGX et en lui donnant comme paramètres:
- dans le registre A : le n° de ST destinatrice,
- dans le registre Y : l'adresse où est rangé le message
- dans le registre X: la longueur de ce message.

L'opération EMETGX construit un bloc qui contient l'adresse de destination, la longueur du texte de l'utilisateur et le texte de l'utilisateur puis demande, par les commandes 2 et 3, l'émission de ce bloc.

REMARQUE

Pour éviter la duplication des messages, on numérote les messages à l'émission (la numerotation se fait modulo 2). L'octet représentant le numéro est rajouté au message initial à l'émission, et il est supprimé à la réception (après vérification).

5) Résultat d'une émission : opération RESMGX

Après une émission, l'utilisateur a besoin de savoir si l'émission s'est bien passée, ou non. En appelant l'opération RESMGX l'utilisateur reçoit, dans le registre A, le résultat de l'émission.

A = 0 si bonne émission
A = 1 si épuisement des reprises
A = 2 si duplication ou perte
A = 3 si destinataire inactif.

En fonction de ce résultat l'utilisateur choisit la manière de continuer son traitement.

6) Réception d'un message : opération RECPGX

Lorsque le communicateur reçoit un message, il stocke ce message. L'utilisateur peut récupérer ce message en commençant par appeler l'opération RECPGX qui lui rend comme résultat :
- dans le registre A : l'adresse de la station émettrice,
- dans le registre Y : l'adresse où est rangé le message,
- dans le registre X : la longueur du message.

L'utilisateur accède ainsi au message et le recopie dans une zone qui lui est propre.

7) Libération du communicateur : opération LIBRGX

Une fois que l'utilisateur a récupéré le message, il libère la mémoire de réception du communicateur pour autoriser la prochaine réception de message. La libération du communicateur (et non sa fermeture) se fait via l'opération LIBRGX (elle est sans paramètre).

8) Traitement des interruptions du communicateur

a) traitement de l'interruption fin émission : ITEMEX
Lorsque le processeur détecte une IT fin émission, le S/P ITEMEX est exécuté.

Le S/P ITEMEX permet d'accéder, au mot d'état, pour récupérer le résultat de l'émission qui sera mis à la disposition de l'utilisateur, puis donne la main à un S/p de l'utilisateur, appelé impérativement ITEMET (pour faire connaître le résultat de l'émission à l'utilisateur).

b) traitement de l'interruption fin de réception : ITRECX
Le S/P de traitement de l'IT fin de réception a pour rôle :
1. de tester si le message reçu est dupliqué (auquel cas le message est ignoré) ;
2. si le message est bon, il est stocké dans une zone fixée à l'implantation
3. avertir l'utilisateur en exécutant un S/P, défini par l'utilisateur, (ayant pour nom ITRECP) qui est chargé de récupérer le message reçu.

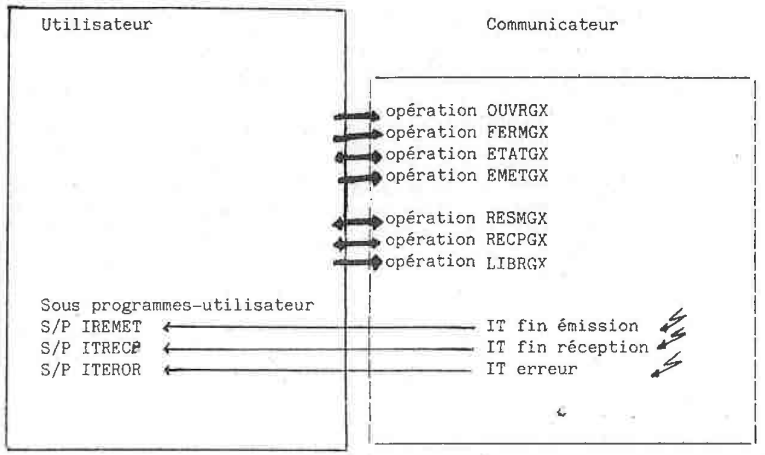
c) traitement de l'interruption erreur du communicateur : ITERXC
Le S/P de traitement de l'IT erreur du communicateur a pour rôle :
1. d'accéder au mot d'état pour récupérer le code erreur qui sera mis à la disposition de l'utilisateur,
2. avertir l'utilisateur de l'erreur en exécutant un S/P défini par l'utilisateur et ayant pour nom ITEROR et pour objectif le traitement de l'erreur.

REMARQUE

L'utilisation des 3 sous-programmes ITEMET, ITRECP et ITEROR permet un asynchronisme entre l'application et les traitements liés au communicateur. On aurait pu opter pour une solution, où l'utilisateur teste périodiquement la fin d'émission,... et l'erreur (cette solution a été écartée car elle introduit de l'attente active).

9) Interface usager-communicateur

Le communicateur est considéré, par l'utilisateur, comme une "boîte noire" à laquelle il peut s'adresser via les opérations décrites précédemment.



On remarquera que la boîte noire, définie ici, n'a pas la même sémantique que les boîtes noires, définies en génie logiciel /SAKHONN, 80/, puisque des S/P définis chez les utilisateurs sont activés à la suite d'interruptions liées au communicateur.

ANNEXE B4

IMPLANTATION ET UTILISATION DE LA STATION DE TRANSPORT

Nous présentons ici les grands traits du protocole de transport du DOD qui a été réalisé /PICHON, 84/, de son implantation et de son utilisation par la couche application.

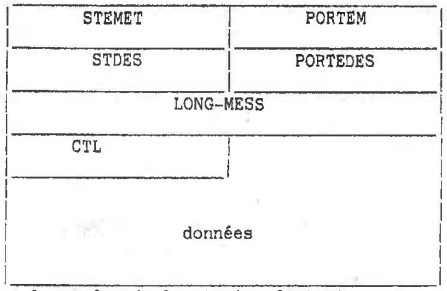
I SPECIFICATION FONCTIONNELLE DU PROTOCOLE DE TRANSPORT DU DOD

Pour présenter le protocole de transport réalisé nous étudierons successivement :
- le format des messages échangés entre les diverses stations de transport,
- l'ensemble des états par lesquels passe une connexion,
- l'établissement d'une connexion,
- la fermeture d'une connexion,
- et l'échange des données.

ST : station de transport

I.1 FORMAT DES MESSAGES

Les messages reçus ou transmis par la ST ont la structure suivante :



STEMET : désigne le numéro de la station émettrice
STDES : désigne le numéro de la station réceptrice
PORTEM : désigne le numéro de la porte émettrice
PORTEDES : désigne le numéro de la porte réceptrice
LONG-MESS : désigne la longueur du message.

CTL = 0 si le message contient des données-utilisateur
CTL = 1 si le message contient une fonction qui concerne la ST et non l'utilisateur.données : peuvent être soit,des données-utilisateur soit,une fonction à réaliser par la ST.

Cette fonction peut avoir les significations suivantes :
fonction = 1 : demande d'ouverture d'une liaison logique,
fonction = 2 : demande de fermeture d'une liaison logique,
fonction = 3 : acquittement du message de données précédent,
fonction = 4 : acceptation de l'ouverture de la liaison logique,
fonction = 5 : acceptation de la fermeture de la liaison logique,
fonction = 6 : reprise de l'établissement d'une connexion (après anomalie).

Dans le cas où il s'agit de données utilisateur, le premier octet de ces données indique le type des messages (urgent ou non)... Les messages urgents sont prioritaires à l'émission et à la réception.

I.2 Automate lié au fonctionnement d'une connexion

Une connexion, ou liaison logique, passe par une suite d'états entre le moment où l'utilisateur demande l'établissement de cette connexion (par la primitive OPEN) et le moment où il demande sa fermeture (par la primitive CLOSE).

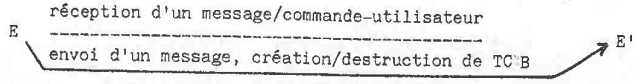
Nous avons résumé à l'aide du diagramme de la figure ci-après l'ensemble de ces états, les transitions d'un état à un autre et les conditions associées à ces transitions.

Commençons par définir les états.

Nous avons repris l'abréviation utilisée dans /POSTEL, 80/ :
- état CLOSED : connexion inexistante (pas de contexte pour la connexion),
- état ESTABLISHED : la connexion est établie, elle est prête pour assurer des échanges entre les 2 entités qui l'ont déclarée ;
- état LISTEN : attente de la demande de connexion en provenance de la ST distante ;
- état SYN-SENT : représente l'attente de l'acquittement, par la station distante, de la demande de connexion envoyée ;
- état SYN-RECEIVED : représente l'attente de la confirmation de l'acquittement de la demande de connexion après avoir reçu et émis une demande de connexion ;
- état FIN-WAIT1 : représente l'attente de la demande de déconnexion par le site distant, ou bien l'attente de l'acquittement de la demande de déconnexion déjà envoyée ;
- état FIN-WAIT2 : représente l'attente de la demande de déconnexion par le site distant ;
- état TIME-WAIT : temporisation pour s'assurer que le site distant a bien reçu l'acquittement à sa demande de déconnexion ;
- état CLOSE-WAIT : attente de la demande de déconnexion par le site local;
- état CLOSING : représente l'attente de l'acquittement de la demande de déconnexion envoyée au site distant.

Notation

Dans le diagramme ci-après, le passage d'un état E à un état E' est soumis à une condition (réception d'un message, d'une station distante, ou bien une commande de l'utilisateur local) et à l'exécution d'actions (création/destruction de contexte de connexion (noté TCB) ou envoi d'un message à un site distant).



Les messages-fonctions échangés entre ST sont :
- SYN : demande de connexion,
- ACK de SYN : acceptation de la connexion
- FIN : demande de déconnexion
- ACK de FIN : acceptation de la déconnexion
- RST : reprise après anomalie.

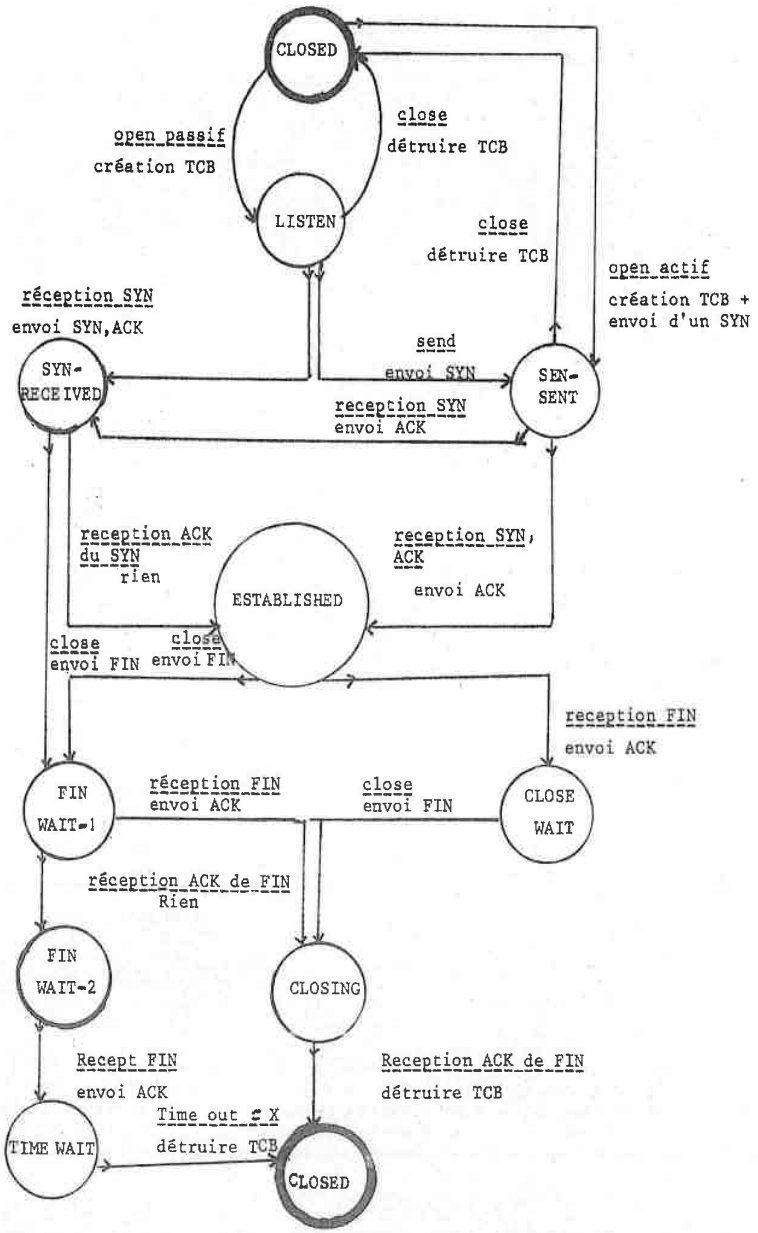


Diagramme des états d'une connexion

I.3 ETABLISSEMENT D'UNE CONNEXION

L'établissement de connexion est la phase où il y a le plus de messages de services qui circulent dans le réseau. A la fin de cette phase la connexion est mise à la disposition des entités qui l'ont demandée pour communiquer.

On distingue trois cas dans l'établissement d'une connexion :
- une entité fait la demande avant l'autre entité,
- les deux entités font la demande en même temps,
- un problème (anomalie) surgit durant la phase d'établissement.

En reprenant les états d'une connexion, l'envoi et la réception de messages de contrôle, nous donnons le déroulement, dans le temps, de l'établissement des connexions dans chacun des trois cas.

Notation

xxx -->	désigne un envoi de message
--> xxx	désigne la réception d'un message et passage dans un nouvel état
<y >	signifie que le message contient la fonction Y
...	signifie que le message envoyé est encore dans le réseau.

1er cas

	Site A		Site B
1	closed		listen
2	syn-sent	--> < syn >	--> syn-received
3	established	<-- < ack-syn >	<-- syn-received
4	established	--> < ack >	--> established
5	envoi et réception de messages		

Dans ce cas une entité du site B a fait la demande en premier.

2ème cas

Les deux sites tentent en même temps d'établir la connexion

	Site A		Site B
1	closed		closed
2	syn-sent	--> < syn >	...
3	syn-received	<-- < syn >	<-- syn-sent
4		< syn >	--> syn-received
5	syn received	--> < ack >	...
6	established	<-- < ack >	<-- syn-received
7		<-- < ack >	established

3ème cas

Un acquittement < ack > ou une demande d'établissement de connexion < syn > peut se perdre dans le réseau; ce fait oblige l'émetteur à envoyer un deuxième message (il y a un risque : est-ce que c'est le premier message envoyé qui n'est pas arrivé ou bien c'est la réponse qui n'est pas parvenue à destination?) et crée ainsi des duplications de certains messages.

Le protocole du dod prévoit ce genre de problèmes et les résoud de manière simple : dès que la ST reçoit des messages en double, elle les ignore et passe à l'état initial (CLOSED) et tente de nouveau l'établissement de la connexion (cas 1 ou cas 2) après avoir averti la ST distante concernée par la connexion en lui envoyant un message fonction < RST>.

I.4 FERMETURE D'UNE CONNEXION

Dès que l'une des deux entités communicantes n'a plus de données à transmettre ou ne veut plus recevoir des données sur une connexion déjà établie elle demande à sa ST de fermer (ou rompre) cette connexion. Comme dans le cas de l'établissement d'une connexion on distingue ici 3 cas :
- cas ou une entité décide de rompre avant l'autre,
- cas où les deux entités décident de rompre en même temps,
- et cas où il y a une anomalie.

1er cas

	SITE A		SITE B
1	established		established
2	close de l'utilisateur local		
	fin-wait1	--> <fin >	--> close-wait
3	fin-wait2	<--	< ack>close-wait
			close de l'utilisateur locale
4	time-wait	<-- <fin >	<-- closing
5	time-wait	--> <ack >	closed
	après temporisation		
6	closed		

2ème cas

	SITE A		SITE B
1	established		established
2	close de l'utilisateur local		close de l'utilisateur local
	fin-wait1	--> < fin >	... fin-wait2
		<-- < fin >	<--
		...	
3	closing	--> < ack >	... closing
		<-- < ack >	<--
		... < ack >	-->
4	closed		closed

3ème cas

On procède comme dans le cas 3 de l'établissement de connexion.

I.5 COMMUNICATION DE DONNEES

Une fois la connexion établie, les entités qui ont demandé la connexion peuvent l'utiliser pour émettre/recevoir des messages.
Les messages échangés peuvent être urgents ou normaux. Les messages urgents sont traités en priorité.

REMARQUE
Après avoir envoyé un message la ST attend une réponse pour ce message. Afin de ne pas attendre indéfiniment la réponse, la ST arme un réveil (i.e demande un délai à l'agence horlogerie). Lorsque ce réveil sonne la ST déclare l'émission impossible.

II IMPLANTATION DE LA STATION DE TRANSPORT

L'objectif de ce paragraphe est d'esquisser les grandes lignes de l'implantation de la ST réalisée.

II.1 INTRODUCTION

Habituellement, une station de transport est implantée de l'une des deux manières suivantes /ZIMMERMANN, 75/:
- le logiciel de la ST se présente comme une tâche en mode usager. Les programmes d'application (ou tâches) communiquent avec la ST en utilisant un mécanisme de communication inter-tâches (boîtes à lettre, files d'attente, etc...).

L'avantage de cette solution réside dans le fait qu'elle ne nécessite aucune modification du système existant. L'inconvénient réside dans l'"overhead" dû à la communication inter-tâches.

- ST est intégrée au système au même titre qu'une autre méthode d'accès (gestion de fichiers, gestion de terminaux, ...).

La St est donc une composante directe du système d'exploitation. La difficulté d'une telle solution tient à l'intervention qui doit être faite dans le système d'exploitation : c'est, en général, la solution "du contructeur".

Pour ce qui nous concerne, nous avons retenu la 2ème solution puisque nous sommes en train de construire un système sous forme de couches (ou d'agences

II.2 QUELQUES REMARQUES SUR L'IMPLANTATION DE LA ST

1) Implantation des opérations de la ST

La ST utilise un ensemble de files et de variables qui sont modifiées via les différents appels de la ST. Pour assurer la cohérence de ces structures de données, les opérations de la ST sont indivisibles.

Comme pour les opérations du noyau temps réel et celles de l'agence horlogerie, l'indivisibilité des opérations de la ST est assurée par le masquage, en début d'opération, et démasquage des interruptions en fin d'opération.

2) Initialisation de la ST

La St utilise des variables et des listes internes; avant d'autoriser l'utilisateur à accèder aux services de la ST, la ST doit initialiser toutes ses structures de données, demander l'activation du communicateur (via l'opération OUVRGX) et démarrer la "tâche reprise".

Une procédure a été réalisée pour assurer cette initialisation. Cette procédure n'est pas connue par l'utilisateur, elle est appelée par le noyau temps-réel au moment de l'initialisation de tout le système.

3) Cas de saturation de la ST

Il arrive des moments où la ST reçoit des messages, à une vitesse plus grande que celle avec laquelle l'ensemble des tâches d'application consomment ces messages. Dans une telle situation, l'espace réservé, par la ST pour les messages en attente de consommation, est saturé et la ST ne peut plus recevoir de messages tant que les tâches d'application n'ont pas prélevé, au moins, un des messages en attente de consommation.

La ST bloque alors la réception de messages en n'autorisant plus le communicateur à recevoir des messages (pour assurer ce blocage il suffit de ne pas libérer la mémoire tampon de réception du communicateur).

4) Problème des reprises

Le problème des reprises, après anomalie, est l'un des problèmes les plus cruciaux des systèmes répartis /CORNAFION, 81/.

Au niveau de la ST, le problème de reprise se pose dans les 2 cas suivants:
- un message transmis n'arrive pas à destination (erreur de transmission, destinataire inactif) ;
- le communicateur local devient inactif, après détection d'une erreur irrécupérable.

1er cas

La ST ne tente pas une réémission juste après lecture du code de réponse de l'émission car le communicateur a déjà tenté de réémettre 5 fois.

Comme on limite les messages en attente d'émission, pour toutes les connexions, on se permet de ne pas rejeter le message non transmis par le communicateur.

Afin de pouvoir le réémettre une nouvelle fois, ce message est remis dans la file des messages en attente d'émission.

Il est possible que les messages recyclés (ie remis dans la file après un essai d'émission négatif) n'arrivent jamais à destination. Comme les tâches d'application utilisant cette connexion critique (ie celle sur laquelle on n'arrive pas à transmettre des messages) participent au contrôle de flux, elles doivent avorter cette connexion si elles se rendent compte qu'au bout d'un certain délai, aucun message n'est transmis sur cette connexion ; ainsi on évite la saturation de la ST par des messages dont le destinataire est non disponible.

2ème cas

Le communicateur devient inactif et aucune communication avec l'extérieur n'est possible.

Un code =77 est retourné aux tâches d'application si celles-ci demandent un service à la ST, pour leur indiquer que la ST est momentanément non opérationnelle.

La ST va essayer, de manière cyclique, de réactiver le communicateur. Cette réactivation se fait par une tâche appelée "tâche de reprise".

Dans ce cas aussi, le communicateur risque de ne jamais repasser dans l'état actif, la décision d'abandonner le traitement incombe aux tâches d'application.

III INTERFACE UTILISATEUR-STATION DE TRANSPORT

L'utilisateur (ou une tâche d'application) accède aux services de la ST en appelant l'une des opérations suivantes :
- OPEN
- SEND
- RECEIVE
- CLOSE
- ABORT
- STATUS.

Pour les tâches d'application la ST est considérée comme une "boîte noire" accessible uniquement à travers les 6 primitives citées précédemment.

Nous allons développer ici la fonction réalisée par chacune de ces primitives et la manière dont elle est appelée par les tâches d'application.

III.1 PRIMITIVE D'ETABLISSEMENT DE CONNEXION : OPEN

1) Sa fonction

Format :

Open (<identificateur de la porte locale >, <identificateur de la porte distante >, <mode> , <n° d'évènement> , <code réponse> , <descripteur>

L'opération OPEN permet à une tâche d'application de demander l'établissement d'une connexion entre deux portes .

Pour chaque porte il faut préciser :
- l'identificateur de la porte locale (n° de station et n° de porte) ;
- l'identificateur de la porte distante ;
- le mode d'ouverture : passif ou actif
  . passif : la ST réserve un contexte pour la connexion. Laquelle connexion ne peut être établie que si une entité distante demande la même connexion mais en mode actif.
  . actif : la ST réserve un contexte pour la connexion et envoie un message à la ST distante pour négocier l'établissement de la connexion (cf I.3).

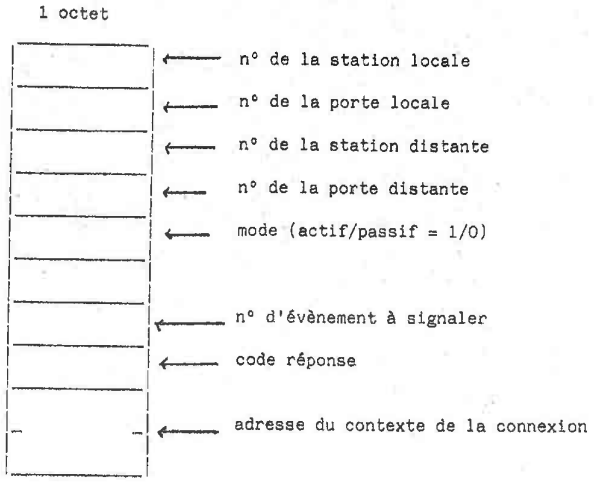
Comme résultat, la tâche qui a fait la demande reçoit :
- l'adresse du descripteur de la connexion déclarée dans < descripteur> ; c'est ce descripteur qui servira à référencer la connexion (il représente son nom). Ce résultat n'implique pas automatiquement l'établissement de la connexion.
- un code réponse:
  = 0 si contexte réservé
  = 1 si connexion déjà existante
  = 2 si pas d'espace pour créer le contexte
  = 77 si la ST est hors service.

L'évènement (au sens SCEPTRE) passé en paramètre est signalé à la tâche lorsque l'établissement de la connexion est terminé .

2) Appel de la primitive OPEN

Comme la version actuelle de la ST n'est utilisable qu'à partir d'un langage assembleur, toutes les primitives sont appelées par les tâches d'application via des macro-instructions qui font appel à leur tour aux services demandés.

Syntaxe OPEN BLOCOP
où BLOCOP est l'adresse d'un bloc contenant les informations suivantes :



III.2 PRIMITIVE DE RUPTURE DE CONNEXION : CLOSE

1) Sa fonction

Format :

CLOSE (<nom de connexion>)

La tâche précise le nom de la connexion à fermer.

La ST agit selon le principe donné dans I.4.

Si la connexion n'existe pas, l'opération est sans effet.

2) Mode d'appel de la primitive CLOSE

Syntaxe: CLOSE ADRCCB
où ADRCCB désigne l'adresse du descripteur de la connexion à fermer.

III.3 PRIMITIVE D'EMISSION DE MESSAGE : SEND

1) Sa fonction

Format :

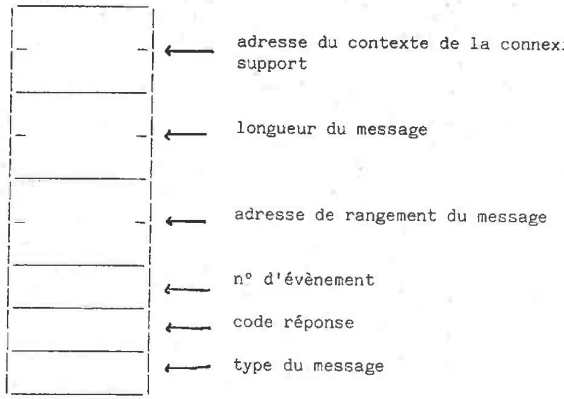
SEND (<nom de la connexion> , <adresse du buffer> , <nombre d'octets> <n° d'évènement> , <code réponse> , <type du message>

Cette opération permet à une tâche d'application d'émettre un message sur une connexion déjà établie.

Les paramètres d'appel sont :
- le nom de la connexion support ;
- l'adresse où est rangé le message à transmettre ;
- le nombre d'octets du message ;
- le type du message (=0 pour message normal et = 1 pour message urgent) ;
- le n° de l'évènement à signaler à la tâche appelante lorsque le message aura atteint sa destination ;
- un code réponse qui contiendra :
. 0 si le message a été accepté par la ST ;
. 1 si la connexion support est inexistante ou non établie ;
. 2 si le nombre maximum de messages en attente d'émission, pour cette connexion, est déjà atteint (le message est rejeté dans ce cas).
. 77 si la ST est non opérationnelle.

2) Mode d'appel à la primitive SEND

Syntaxe : SEND BLOCSD
où BLOCSD désigne l'adresse d'un bloc qui contient les informations suivantes :



III.4 PRIMITIVE DE RECEPTION DE MESSAGE : RECEIVE

1) Sa fonction

Format :

RECEIVE (<nom de connexion> , <adresse buffer> , <nombre d'octets>, <code réponse>)

Cette opération permet, à la tâche appelante, de recevoir le premier message de la file des messages reçus (et non encore consommés) sur la connexion spécifiée en paramètre.

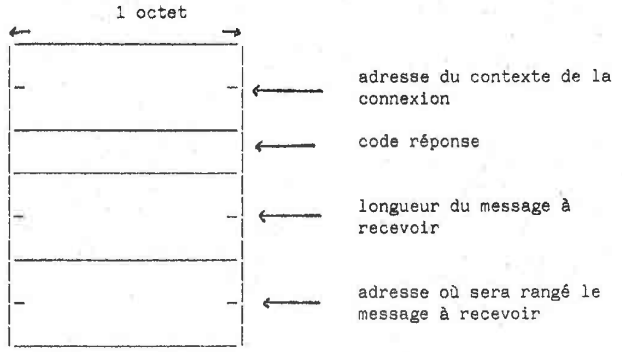
Il y a troncature du message reçu si sa longueur est supérieure à celle spécifiée en paramètre (et dans le cas contraire, le message est complété par des blancs).

Il est évident que cette demande n'a de sens que s'il y a au moins un message en attente de consommation. Avant d'appeler l'opération RECEIVE, la tâche peut s'assurer de l'existence d'un tel message en appelant l'opération STATUS (cf III.6).

Le code réponse contient :
- 0 si un message est délivré à l'appelant
- 1 s'il n'y a pas de message à consommer ;
- 2 si la connexion n'existe pas ou bien si elle n'est pas établie ;
- 77 si la ST est hors service.

2) Mode d'appel de la primitive RECEIVE

Syntaxe : RECEIVE BLOCRV
où BLOCRV désigne un bloc qui contient les informations suivantes :



III 5 PRIMITIVE D'AVORTEMENT D'UNE CONNEXION : ABORT

1) Sa fonction

Format :

ABORT (<nom de connexion>)

l'opération ABORT permet, à une tâche, de demander à sa ST, d'ignorer tous les messages en attente d'émission et/ou de réception sur la connexion spécifiée en paramètre (la connexion est purgée).

Si la connexion n'existe pas, l'opération est sans effet.

2) Mode d'appel de la primitive ABORT

Syntaxe : ABORT ADR
où ADR désigne l'adresse du contexte de la connexion à avorter.

III.6 PRIMITIVE D'OBTENTION DE L'ETAT D'UNE CONNEXION : STATUS

1) Sa fonction

Format :

STATUS (nom de connexion), bloc résultat)

L'opération STATUS permet, à une tâche, de connaître l'état d'une connexion. Comme résultat, elle rend, à la tâche, appelante un bloc d'informations contenant :
- le nom de la porte locale
- le nom de la porte distante
- le nombre de messages en attente d'émission
- le nombre de messages en attente de consommation,
- l'état de la connexion ;

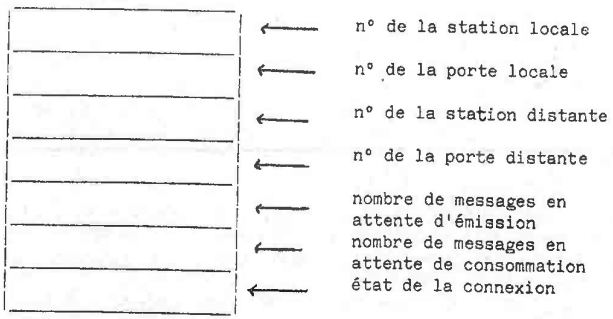
0 pour non établie
1 pour établie
2 pour inexistante
77 pour ST hors service.

Dans le cas où la connexion est inexistante, les informations du bloc résultat n'ont pas de sens.

Grâce à l'opération STATUS, une tâche peut savoir si elle peut (ou non) émettre et/ou recevoir des messages.

2) Mode d'appel de la primitive STATUS

Syntaxe STATUS ADRCCB, BLOC ST
où ADRCCB désigne l'adresse du contexte de la connexion et BLOCST désigne un bloc qui va contenir les informations suivantes :



III 7 REGLES D'UTILISATION DES CONNEXIONS

1. Dans la réalisation actuelle, une connexion établie entre deux tâches n'est pas privée : toute autre tâche qui connaît le nom de cette connexion peut l'utiliser. Pour avoir des connexions privées on peut :
- soit ajouter un nouveau paramètre, à la primitive OPEN, qui précise le type de la connexion (privée ou non) ;
- soit cacher le nom de la connexion aux autres tâches.
2. le code réponse doit être exploité par la tâche après l'appel.
3. lorsque le code réponse est égal à 77, la ST est hors service et aucun appel n'a d'effet tant que la ST n'est pas redevenue opérationnelle.

BIBLIOGRAPHIE

/J.R. ABRIAL 74/
Data semantics
Data base management - Klimbic et Koffeman eds
North Holland pub comp 1974 - P1-59.

/M. ADIBA 77/
Présentation générale d'un système de coopération de bases de données réparties.
Afcet journées BDDR PARIS mars 77.

/A.L. AMBLER 77/
Gypsy : a language for specification and implementation of verificable programs
Sigplan acm mars 77 p1-10.

/G.A. ANDERSON-D.E. KNIGHT 81/
Pratical software fault tolerance for real-time software.
Computer laboratory england june 81.

/S. ANDLER 78/
Synchronisation primitives and the verification of concurrent programs.
Colloque sur les systèmes d'exploitation IRIA 78.

/ANSI-X3-SPARC 75/
Interim report on data base management systems
ISO/TC97/SC5 mars 75.

/APSIS 81/
LDP : Langage de définition de programmes
Manuel d'utilisation

/APSIS 83/
Factor
Apsis Grenoble 83

/J. ARSAC 77/
La construction de programmes structurés
Dunod 77

/M. AUGER 82/
Présentation de PROWAY
EDF HN/231 82.

/ASEA/
Master Asea
Manuel de présentation du système.

/M. BENMAIZA 84/
Le concept d'évènement dans la spécification et la programmation d'applications temps réel.
Thèse Doct-Ingénieur INPL 12 mars 84.

/S. BENNET and DA. LINKENS 82/
Computer control of industrial process
Edition Peter Peregrins LTD 82.

/M. BLANCHARD 78/
Rapport de la "commission de normalisation de la représentation du cahier des charges d'un automatisme logique".
Automatisme mars-avril 78 P66-83.

/M. BLANCHARD 79/
Comprendre, maîtriser et appliquer le GRAFCET
Edition cepadues 79.

/BNI 80/
Projet Sceptre : proposition de standard de noyau d'exécutif temps réel.
1ère édition BNI 79-2-36-0-55 mai 80.

/BNI 81/
Réseaux informatiques : introduction à l'interconnexion des systèmes ouverts.
Afnor-BNI normes et techniques 81.

/BNI 82/
Projet Sceptre : proposition de standard de noyau d'exécutif temps réel.
2ème édition BNI 26-2 spet 82.

/B. BOEHM 82/
Les facteurs du coût du logiciel
RAIRO-TSI Vol1 janvier 82 P5-24

/D. BOUCHET 79/
Les procédures de reprise.
Edition hommes et techniques 79.

/I. BORNE 83/
Les objets dans LOGO.
Bigre n°37 dec 83 ; P2-12.

/R. BOUDAREL, et Al 67/
Commande optimale des processus
Edition Dunod 67.

/G.W. BRAMS 83/
Réseaux de Pétri : théorie et pratique , tome 1 et tome 2
Edition Masson 83.

/P. BRINCH HANSEN 78/
Distributed processes : a concurrent programming concept
COM OF ACM Vol 21 N11 Nov 78 p 934-941.

/P. BROCHET - P.LOPEZ 80/
Automatismes séquentiels ; la synthèse par le GRAFCET
A2I N83 Janv 80 P19-27

/F. BROWAEYS et Al 84/
Sceptre : proposition de noyau normalisé pour les exécutifs temps réel
Afcet-TSI 1984 P45-62.

/J.P. CALVEZ 82/
Une méthodologie de conception des systèmes multimicroordinateurs pour les applications de commande en temps réel
Thèse d'état université de Nantes, novembre 82.

/RH. CAMPBELL - AN. HABERMANN 74/
The specification of process synchronisation by path expressions
LNCS Vol. 16 spring verlag 74.

/R. CARPENTIER - J. SOKOL 80/
Servicing users with a local area network
Computer networks 4 (1984) P209-214.

/E. CATIER 79/
La nouvelle robotique industrielle
A21 N81 novembre 79 P18-27.

/DD. CLARCK - KT. PROGRAM- DP. REED 78/
An introduction to local area networks
IEEE Vol 66 N11 novembre 78.

/PN. CLOUT 82/
Real-time operating system needs
Interfaces in computing 1 (1982) P3-10.

/EF. CODD 70/
A relational model of data for large shared data banks
Com of ACM vol 13 n°6 juin 1970 P377-387.

/P. COINTE 83/
Comprendre Smalltalk, penser Smalltalk
Bigre n°37 dec 83 P13-54.

/CONTROL BAILEY/
Micro-Z
Manuel de présentation du système

/JM. COUR 82/
La communication entre ordinateurs dans le domaine privé les réseaux locaux
Telesoft février-mars 82, P82-88.

/M. COURVOISIER et al 83/
Réseaux d'automates et ateliers flexibles
Afcet congrès automatique 15-17 novembre 83, Besançon.

/CORNAFION 82/
Systèmes informatiques répartis : concepts et techniques
Edition Dunod 81.

/CROCUS 75/
Systèmes d'exploitation des ordinateurs
Edition Dunod 75.

/JF. CUDULOU et Al 84/
Modélisation de spécification sous forme d'objets et d'actions
Bigre N41 novembre 84 P81-100.

/M. DANG and Al 83/
Local networks for distributed process control systems
DCS academic press London 83.

/CJ. DATE 75/
An introduction to data base systems
Addison-Wesley, Reading, Mass 1975.

/R. DEMOLOMBE 84/
Bases de données : nouvelles perspectives
INRIA bulletin de liaison n 2/1984.

/J.B. DENNIS 75/
First version of a data flow procedure language
Mac technical mémorandium 61 MIT may 1975

/D. DEPEYRE 79/
Le contrôle-commande par ordinateurs des processus industriels
A 2I N 82 décembre 79 P23-31.

/JC. DERNIAME 74/
Le projet Civa : un système de programmation modulaire
Thèse d'état Nancy I janvier 74.

/JC. DERNIAME et JP. FINANCE 79/
Types abstraits de données : specification, utilisation et réalisation
Ecole d'été de l'afcet monastir 79-E-57 1979.

/JC. DERNIAME - GR. PERRIN - JP. THOMESSE 82A/
Une classification des concepts relatifs à la communication en vue de définir une démarche de conception de systèmes parallèles
Crin rapport interne 82.

/JC. DERNIAME - GR. PERRIN - JP. THOMESSE 82B/
Communication design of distributed systems
Crin rapport interne 82.

/JC. DERNIAME - A. ZAKARI 82C/
Spécification d'application de conduite d'un atelier flexible
Crin rapport interne 82.

/FC. DERNIAME - A. ZAKARI 83/
Langages de conception pour les applications réparties en conduite de procédés
Bigre n° 35, 17-19 octobre 83 P363-376.

/P. DESCHIZEAUX - P. LADET 77/
Programmation en temps réel des synchronisations par gestion des liens évènements-processus
Afcet programme des journées temps réel 77.

/P. DESCHIZEAUX 82/
Temps et évènements dans les systèmes distribués de contrôle de procédés
RAIRO automatique vol 16 n° 3 1982 P252-274.

/DW. DIJKSTRA 72/
Notes on stuctured programs
Apic studies on data processing n°8 72 ; P1-81.

/Y. EBIHARA and Al 83/
Gamma-net : a local computer network coupled by high speed optical fiber ring bus ;
System concept and structure
Computer networks 7 (1983) P375-388.

/ECMA 82/
Standard ECMA-72 transport protocol
ECMA 2ème édition septembre 82.

/EI 83/
Special automates programmables
Electronique industrielle N57 avril 83.

/EFCIS /
Module MPU9
Thomson-Efcis.

/A. ELFAZIKI 85/
Contribution à la structuration et à la programmation des applications de contrôle de procédés industriels
Thèse 3ème cycle Nancy I février 85.

/JF. ELZER 75/
PEARL
Journées AFCET "langages temps réel" Paris 75.

/EUROTHERM/
Système FICS10
Manuel de présentation du système

/JA. FELDMAN 79/
High level programming language for distributed computing
Com of ACM vol 22 n° 6 juin 79 P353-368.

/J. FERBER 83/
Les langages orientés dans la représentation des connaissances.
Bigre n° 37 décembre 83 P160-172.

/JP. FINANCE 79/
Etude de la construction des programmes : méthode et langage de spécification et de résolution de problèmes
Thèse d'état Nancy I octobre 79.

/FISHER CONTROL/
Provox
Manuel de présentation du système.

/FISHER et PORTER/
Système MICRO-DCI
Manuel de présentation du système.

/AJ. FOSSARD et Al 72/
Aperçus sur la commande hiérarchisée.
Rairo août 72 P3-40.

/O. FOUCAULT 82/
Modèle et outil pour la conception des systèmes d'information dans les organisations: projet REMORA
Thèse d'état université Nancy I juin 82.

/G. GARDARIN et AL 76/
Spécification fonctionnelle des systèmes de gestion de bases de données réparties.
Convention informatique Paris 76 P84-90.

/G. GARDARIN et AL 77/
Analyse de la conception d'un système de gestion des bases de données réparties
AFCET journées BDDR Paris mars 77.

/G. GARDARIN 78/
Résolution des conflits d'accès simultanés à un ensemble d'informations-application aux bases de données réparties.
Thèse es-sciences avril 78 Paris VI.

/M. GIEN 79/
Introduction aux réseaux locaux
INRIA DUNE RES1-50N avril 79.

/GIXI 81/
Présentation technique des communicateurs TR81-A
GIXI décembre 81.

/J. GORSKI 84/
Designing a data base for real-time applications
Afcet congrès génie logiciel Nice juin 84.

/MP. GROOVER 82A/
Fabrication assistée par ordinateur
Edition Hermès 82.

/MP. GROOVER 82B/
Automatisation des systèmes de production
Edition Hermès 82.

/T. HAERDER - A. REUTER 83/
Principles of transaction-oriented data base recovery.
Computing surveys vol 15 n° 4 décembre 83 P287-317.

/A. HAURAT et AL 82/
LMAC : un langage modulaire de commande de robots industriels
AFCET premier colloque de génie logiciel Paris Juin 82

/H. HECHT 76/
Fault tolerant software for real-time applications
Computing surveys vol n° 4 décembre 1976 P391-407.

/CAR. HOARE 74/
Monitors : an operating system structure concept
Com of ACM vol 17 n° 10 octobre 74 p549-557.

/CAR. HOARE 78/
Communicating sequential processes
Com of ACM vol 21 n° 8 august 78 P666-677.

/ICS 75/
Industrial control system
Telemecanique électrique 75.

/JD. ICHBIAH and AL 79/
Rationale for the design of ADA programming language
Sigplan notices ACM vol 14 juin 79.

/C. IUNG et AL 83/
Conception d'une chaine complexe de commande numérique de machines électriques.
Conumel, Toulouse mai 83.

/K. JACKSON - HF. HARTE 76/
The achievement of well structured software in real-time applications.
Ifac/Ifip programmation temps reel 1976.

/ISO 81/
Data processing-Open system interconnection-basic reference model.
Computer com review vol 11 n° 2 avril 81.

/M. JOUVE 77/
Etude des problèmes de fiabilité (permanence aux pannes) dans un SGBD réparti
Congrès afcet Paris 17-18 mars 77.

/C. KAISER et Al 78/
Système informatique réparti utilisé pour la conduite d'un atelier mécanique.
Afcet Paris 13-15 novembre 78 P57-69.

/R.M. KELLER 76/
Formal verification of parallel programs
Com of ACM Jul 76 vol 19 n° 7 P 372-394

/ P. KIENER et Al 84/
Automatisation de la coulée continue de Sollac-Sacilor
Cours à l'ENSEM 84.

/J. KRAMER - J. MAGEE - M. SOLMAN 81/
Communication system architecture for monitoring and control in coal mines.
dept of computing imperial college London 81.

/J. KRAMER - J. MAGEE - M. SLOMAN 82/
A softwarre architecture for distributed computer control systems.
Ifac New-Delhi 5-7 janvier 82.

/P. LADET 77/
Outils de structuration en temps réel pour la commande des procédés industriels
Thèse 3ème cycle INPG 77.

/P. LADET 82/
Contribution à l'étude des systèmes informatiques répartis pour la commande des procédés industriels
Thèse d'état INPG juin 82.

/L. LAMPORT 78/
Time clocks and the ordering of events in a distributed system
Com of ACM vol 21 n° 7 juillet 78 P558-565.

/C. LAURGEAU - G. MICHEL - B. ESPIAU 79/
Les automates programmables industriels
Edition Dunod 79.

/JL. LAURIERE 82/
La représentation et l'utilisation des connaissances
1ère partie : les systèmes experts
2ème partie : représentation des connaissances
Afcet-TSI vol n° 1 1982
Afcet-TSI vol N° 2 1982

/F. LECALVEZ 79/
Définition d'un langage de description des applications en temps réel.
Thèse 3ème cycle Paris VI janvier 79.

/G. LELANN 77/
Distributed systems : toward a formal approach
Ifip 77 p155-160.

/G. LELANN 79/
Le contrôle dans les systèmes informatiques répartis : nature des problèmes et quelques solutions.congrès Afcet juin 79 Nancy P266-287.

/G. LELANN and Al 81/
Distributed systems : architecture and implementation
LNCS spring verlag 81.

/B. LISKOV 79/
Primitives for distributed computing
7th symp on operating systems principles december 79.

/B. LISKOV and Al 81/
Guardians and actions : linguistic support for robust distributed programs
Cambridge MA 02139 81.

/J. LONCHAMP 83/
Structuration logique en termes d'agents communicants des applications en commande/contrôle de processus
Journées bigre 1983 le Cap d'Agde.

/LORRAINS 79/
Réseaux téléinformatiques
Edition Hachette 79.

/EI. LOWE - AE. HIDDEN 71/
Computer control in process industries
Peter Peregrinus LTD 71.

/C. MACCHI - JF. GUILBERT 79/
Téléinformatique : transport et traiement de l'information dans les réseaux et les systèmes téléinformatiques
Edition Dunod 79.

/Z. MAMMERI 82/
Etude et réalisation d'un noyau de communication
Rapport de DEA INPL 82

/Z. MAMMERI 84A/
Manuel d'utilisation du noyau temps réel sceptre implanté sur 6809-THEMIS
Crin 84-R-007 Nancy 84.

/Z. MAMMERI - JP. THOMESSE 84B/
Un réseau local de microordinateurs pour l'enseignement des systèmes temps réel et répartis
Ifip africom Tunis mai 84.

/HG. MENDELBAUM 76/
Structuration dans la conception et la réalisation des systèmes en temps réel
Thèse d'état IPP janvier 76.

/RM. METCAFLE and DR. BOGGS 76/
Ethernet : distributed packet switching for local computer networks
Com of ACM vol 19 n° 7 76 p395-404.

/B. MEYER 80/
La specification
EDF HI/3520-01 80.

/MIMI 84/
Réseaux locaux pour applications bureautiques
Une récapitulation globale
Minis et micros n° 207 84 P67-70.

/M. MOALLA - J. PULLOU - J. SIFAKIS 78/
Réseaux de Pétri synchronisés
Rairo automatique vol 12 n° 2 78 P103-130.

/W. MYERS 82/
Toward a local network standard
IEEE micro 82 P28-45.

/N. NAFFAH 78/
Taro : un système de transmission en boucle pour les réseaux locaux.
Inria rel 2-5101 juin 78.

/H. NAFFAH 80/
Les applications des réseaux locaux
Inria Kayak rel 2-525 juillet 80.

/N. NAFFAH 81/
Les réseaux locaux : définition et exemple de réalisation (Danube)
Inria Kayak rel 2-540 novembre 81.

/S.OWICKI - D. GRIES 76/
An axiomatic proof technique for parallel programs
Acta informatica 6 1976 P319-340

/C. PAIR 79/
La construction des programmes
Rairo informatique vol 13 n° 2 79 P113-137.

/C. PARENT 77/
Intégrité des données d'une base répartie
Congrès Afcet Paris 17-18 mars 77.

/P. PARAYRE 84/
LTR V3 ; manuel officiel de référence
CRIN Nancy 84.

/DL. PARNAS 72A/
A technique software module specification with examples
Com of ACM vol 15 n° 5 mai 72 P330-336.

/DL. PARNAS 72B/
On the criteria to be used in decomposing systems into modules
Com of ACM vol 15 n° 12 décembre 72 P1053-1058.

/DL. PARNAS 75/
Use of the concept of transparency in the design of hierachically structured systems
Com of ACM vol 18 n° 18 juillet 75 P401-408.

/J. PETERSON 77/
Petri nets
Computing surveys vol 9 n°3 september 77 P223-252.

/E. PICHON 84/
Ecriture d'une station de transport
DEA INPL 84

/HE. PIKE 70/
Process control software
IEEE proceedings vol 58 n°1.

/J. POSTEL 80/
Dod standard transmission control protocol
Isi south california 90291 1980.

/M. RAYNAL - P. LE GUERNIC 80A/
Expression de la communication dans les langages
Irisa Rennes rapport n° 129 80

/M. RAYNAL - P. LE GUERNIC 80B/
Elèments d'un langage adapté à la communication entre processus
Afcet logiciel et matériel 24-27 novembre 80 P667-676.

/DP. REED - RK KANODIA 79/
Synchronisation with event counters and sequencers
Com of ACM vol 22 n° 2 février 79 P115-123.

/P. ROBERT - JP. VERJUS 77/
Toward autonomous descriptions of synchronisation modules
Ifip 77 P981-986.

/D. ROBSON 81A/
Object-oriented software systems
Byte august 81 P74-90.

/D. ROBSON 81B/
The smalltalk-80 system
Byte august 81 ; P36-48.

/C. ROCHE 84/
LRO : langage de représentation d'objets
BIGRE n° 41 journées langages orientés objets nov. 84

/A. ROCHFELD 77/
Intérêt des bases de données réparties
Afcet journées bd répartis I P Paris mars 77.

/R. ROUSSEAU 83/
Définition et implantation d'un système modulaire de programmation pour la productique : LMAC
Thèse université de Franche-Comté Avril 83.

/DT. ROSS 77/
Structured analysis (SA) : a language for communication ideas
IEEE tose vol SE-3 N° 1 janvier 77 , p16-34.

/G. ROUCHOUSE 83/
Etude et description du cahier des charges d'automatismes le grafcet - le gemma.
Cetim n° 1.4N.08.1 février 83.

/WB. ROUSSE and Al 77/
Design of man-machine interfaces in process control
Ifac computer applications to process control 77.

/S. SAKHONN CHAK 80/
Evolutivite du logiciel
Thèse 3ème cycle CRIN Nancy, septembre 80.

/J.B. SCHOEFFLER 80/
Distributed control system data base updating and error recovery
Automatica vol 16 1980 P65-72.

/SEMS/
MAC16
Manuel de présentation du système.

/SIEMENS/
Système de conduite de procédé TELEPERM.
Manuel de présentation du système.

/A. SILBERSCHATZ 81/
Port directed communication
The computer journal vol 24 n°1 81 P78-82.

/SOLLAC 80/
Présentation de la coulée continue de Gandrange.
Documentation Sollac-Sacilor.

/MS. SOLMAN - S. PRINCE 80/
Local netword architecture for process control
Ifip ZUrich august 80.

/SYNERTEK/
Versatile interface adapter via.
Synertek corporation SY6522.

/H. TARDIEU et Al 83/
La méthode merise : principes et outils
Edition d'organisation 83.

/S. THIELLIEZ 78/
Pratique séquentielle et réseaux de Pétri
Editions Eyrolles 78.

/RH. THOMAS 76/
A solution to the update problem for the multiple copy data base wich uses distributed control.
BBN report N 3342 july 76.

/JP. THOMESSE - JF. AUBRY 79/
Conception et implantation
Exemple d'une application répartie
Bigre 15-16 janvier 79 Nancy p337-354.

/JP. THOMESSE 80/
Sygare : une structuration pour la conception d'applications en temps réel et réparties
Thèse d'état INPL mai 80.

/JP. THOMESSE - M. BENMAIZA 83/
A specification of real-time applications by events and links
Ifac mexico jan 71-20 83.

/JP. THOMESSE - D. GALARA 84/
Fip : proposition d'un système de transmission série multiplexé pour les échanges d'informations entre des capteurs, des actionneurs et des automates réflexes
Groupe RLI du ministère de l'industrie 1984.

/A. TILTI - MG. SINGH 78/
Systems decomposition, optimisation and control
Edition pergamon 78

/A. TILTI - MG. SINGH 78/
Analyse et commande des systèmes complexes
Edition cepadues 79.

/JM. TOULOTTE 78/
Réseaux de pétri et automates programmables
Automatisme juillet-août 78 P200-211.

/P. VERNEL 77/
Conception et réalisation d'un multicalculateur temps-réel à grande sûreté de fonctionnement
Thèse d'état INPL 1977.

/JT. WEBB 75/
CORAL66
Afcet "langages temps réel" Paris 75.

/N. WIRTH 71A/
Program development by stepwise refinement
Com of ACM vol 14 n° 4 avril 71 P221-227.

/N. WIRTH 71 B/
The programming language PASCAL
Acta informatica Vol 1 N1

/N. WIRTH 77/
Modula : a language for modular programming
Software practice and experience vol 7 77 P3-P35.

/A. ZAKARI 84/
Flexi : langage de conception d'applications de conduite de procédés industriels
Thèse 3ème cycle Nancy I mars 84.

/MV. ZELKOWITZ 78/
Perspectives on software engineering
Computing surveys vol 10 n° 2 juin 78 p197-216.

/H. ZIMMERMANN 76/
Insertion d'une station de transport dans un système d'exploitation.
Inria res-cyclades SCH 546, 75.

institut
national
polytechnique
de lorraine

Le Président,

AUTORISATION DE SOUTENANCE DE THESE

DU DOCTORAT DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

****

VU LES RAPPORTS ETABLIS PAR :

Messieurs les Professeurs DERNIAME
HAURAT

le Président de l'Institut National Polytechnique de Lorraine, autorise:

Monsieur MAMMERI Zoubir

à soutenir devant l'I.N.P.L. une thèse de DOCTORAT intitulée :

"CONCEPTION D'APPLICATIONS REPARTIES EN COMMANDE DE PROCESSUS :

UNE APPROCHE PAR LA STRUCTURATION DES DONNEES"

en vue de l'obtention **du titre de**

DOCTEUR DE L'INSTITUT NATIONAL POLYTECHNIQUE DE LORRAINE

Spécialité : "INFORMATIQUE"

Fait à NANCY, le 7 Juin 1985

Le Président de l'I.N.P.L.

H. LUCIUS

RESUME

Cette thèse est une contribution aux méthodes de conception des applications de commande en temps réel.

L'approche développée dans ce travail, consiste à structurer l'ensemble des données relatives à la commande et à la conduite des procédés, pour décrire toute application de commande, à l'aide de deux ensembles complémentaires :

- un ensemble de données (ou base de données industrielles)
- et un ensemble de traitements (ou actions).

L'activation des actions est exprimée à l'aide des concepts temps, événement et condition et est contrôlée par la mise à jour des données.

Il y a séparation entre données et actions et entre partie contrôle et partie opérative.

L'approche d'implantation d'applications réparties est fondée sur la hiérarchisation des installations de commande.

MOTS CLES

Application de commande en temps réel - Structuration - Modularité - Données/actions - Base de données industrielles - Répartition - Sûreté de fonctionnement - Evénement - Implantation - Noyau de système temps réel et réparti.