



**HAL**  
open science

**Aspects méthodologiques de la conception et de la description de systèmes microprocesseurs : réalisation d'un périphérique d'aide au développement d'applications microprocesseurs et de supervision en temps réel**

Alexander Zambrano Castillo

► **To cite this version:**

Alexander Zambrano Castillo. Aspects méthodologiques de la conception et de la description de systèmes microprocesseurs : réalisation d'un périphérique d'aide au développement d'applications microprocesseurs et de supervision en temps réel. Modélisation et simulation. Institut National Polytechnique de Grenoble - INPG, 1980. Français. NNT: . tel-00292366

**HAL Id: tel-00292366**

**<https://theses.hal.science/tel-00292366>**

Submitted on 1 Jul 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# THESE

*présentée à*

**Institut National Polytechnique de Grenoble**

*pour obtenir le grade de*

**DOCTEUR INGENIEUR**

**"Génie Informatique"**

*par*

**Alexander ZAMBRANO CASTILLO**



**ASPECTS METHODOLOGIQUES DE LA CONCEPTION  
ET DE LA DESCRIPTION DE SYSTEMES MICROPROCESSEURS :  
REALISATION D'UN PERIPHERIQUE D'AIDE  
AU DEVELOPPEMENT D'APPLICATIONS MICROPROCESSEURS  
ET DE SUPERVISION EN TEMPS REEL.**



Thèse soutenue le 6 juin 1980 devant la Commission d'Examen :

Monsieur L. BOLLIET      Président

Messieurs F. ANCEAU  
S. GUIBOUD-RIBAUD  
R. BOUTTAZ  
J.P. SCHOELLKOPF

} Examineurs

# INSTITUT NATIONAL POLYTECHNIQUE DE GRENOBLE

Année universitaire 1977-1978

Président : M. Philippe TRAYNARD

Vice-présidents : M. René PAUTHENET

M. Georges LESPINARD

---

## PROFESSEURS TITULAIRES

MM. BENOIT Jean	Electronique - automatique
BESSON Jean	Chimie minérale
BLOCH Daniel	Physique du solide - cristallographie
BONNETAIN Lucien	Génie chimique
BONNIER Etienne	Métallurgie
* BOUDOURIS Georges	Electronique - automatique
BRISSONNEAU Pierre	Physique du solide - cristallographie
BUYLE-BODIN Maurice	Electronique - automatique
COUMES André	Electronique - automatique
DURAND Francis	Métallurgie
FELICI Noël	Electronique - automatique
FOULARD Claude	Electronique - automatique
LANCIA Roland	Electronique - automatique
LONGEQUEUE Jean-Pierre	Physique nucléaire corpusculaire
LESPINARD Georges	Mécanique
MOREAU René	Mécanique
PARIAUD Jean-Charles	Chimie - physique
PAUTHENET René	Electronique - automatique
PERRET René	Electronique - automatique
POLOUJADOFF Michel	Electronique - automatique
TRAYNARD Philippe	Chimie - physique
VEILLON Gérard	Informatique fondamentale et appliquée
* en congé pour études	

## PROFESSEURS SANS CHAIRE

MM. BLIMAN Samuël	Electronique - automatique
BOUVARD Maurice	Génie mécanique
COHEN Joseph	Electronique - automatique
GUYOT Pierre	Métallurgie physique
LACOUME Jean-Louis	Electronique - automatique
JOUBERT Jean-Claude	Physique du solide - cristallographie

.../...

MM. ROBERT André	Chimie appliquée et des matériaux
ROBERT François	Analyse numérique
ZADWORNY François	Electronique - automatique

#### MAITRES DE CONFERENCES

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

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

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

Personnalités habilitées à diriger des travaux de recherche (décision du Conseil Scientifique)  
E.N.S.E.E.G.

MM. BISCONDI Michel	Ecole des Mines St. Etienne (dépt. Métallurgie)
BOQS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...



MM. KOBYLANSKI André  
 LE COZE Jean  
 LESBATS Pierre  
 LEVY Jacques  
 RIEU Jean  
 SAINFORT  
 SOUQUET  
 CAILLET Marcel  
 COULON Michel  
 GUILHOT Bernard  
 LALAUZE René  
 LANCELOT Francis  
 SARRAZIN Pierre  
 SOUSTELLE Michel  
 THEVENOT François  
 THOMAS Gérard  
 TOUZAIN Philippe  
 TRAN MINH Canh

Ecole des Mines St. Etienne (Métallurgie)  
 Ecole des Mines St. Etienne (Métallurgie)  
 Ecole des Mines St. Etienne (Métallurgie)  
 Ecole des Mines St. Etienne (Métallurgie)  
 Ecole des Mines St. Etienne (Métallurgie)  
 C.E.N. Grenoble (Métallurgie)  
 U.S.M.G.  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)  
 Ecole des Mines St. Etienne (Chim. Min. Ph.)

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

MM. BOREL  
 KAMARINOS

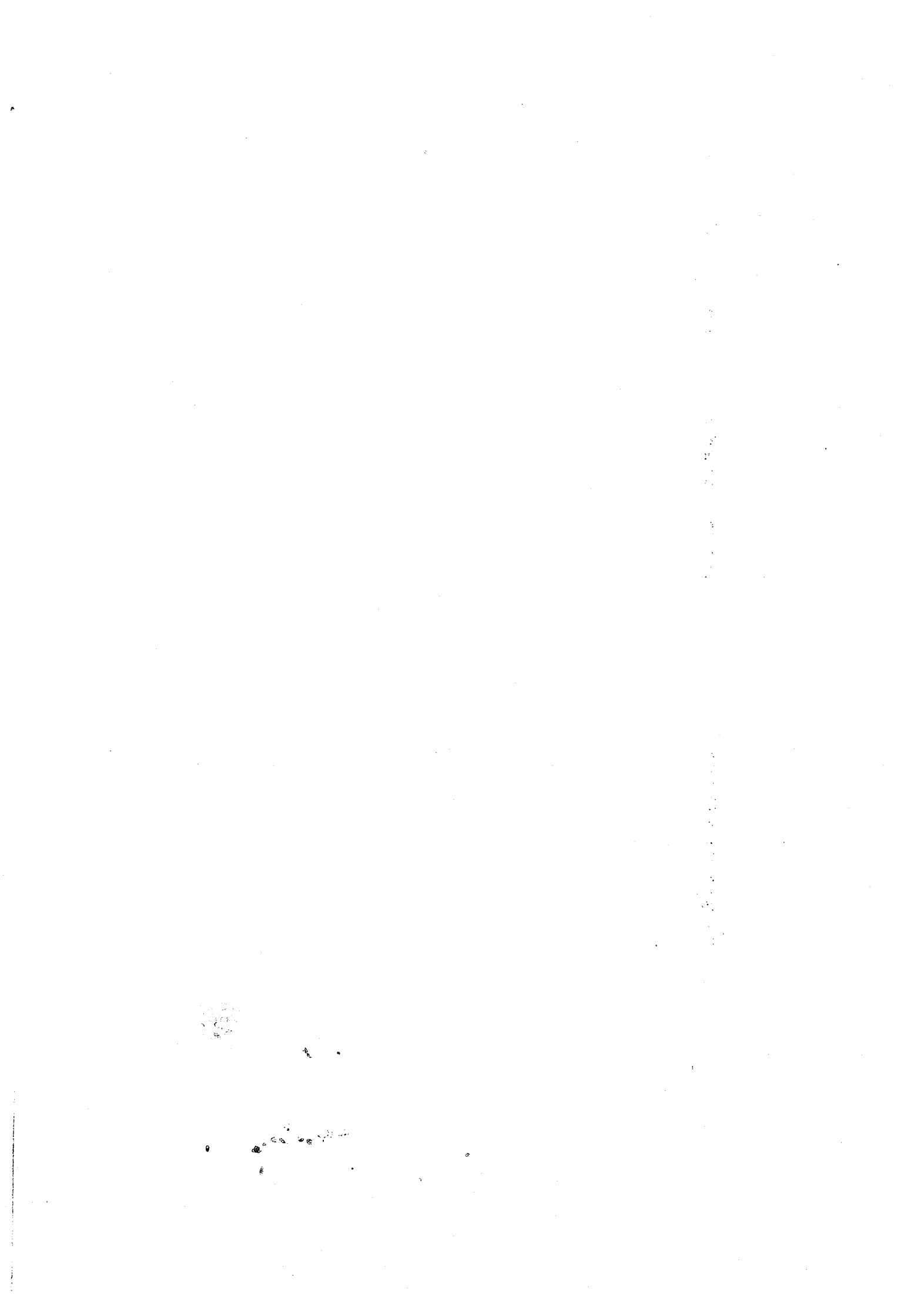
Centre d'études nucléaires de Grenoble  
 Centre national recherche scientifique

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

M. BORNARD  
 Mme CHERUY  
 MM. DAVID  
 DESCHIZEAUX

Centre national recherche scientifique  
 Centre national recherche scientifique  
 Centre national recherche scientifique  
 Centre national recherche scientifique

*A mes parents,  
ma femme,  
ma fille.*



Je tiens à remercier,

Monsieur L.BOLLIET, Professeur à l'Université de Grenoble, qui m'a fait l'honneur de présider le Jury de cette thèse,

Monsieur S.GUIBOUD-RIBAUD, Professeur à l'Ecole Nationale Supérieure des Mines de Saint Etienne, qui a bien voulu juger mon travail et faire partie du Jury,

Monsieur F.ANCEAU, Professeur à l'Ecole Nationale Supérieure d'Ingenieurs en Mathématiques Appliquées de Grenoble, qui a bien voulu m'accepter au sein de l'Equipe de Recherche en Architecture d'Ordinateurs qu'il dirige, et qui m'a guidé dans la conception et réalisation de ce travail,

Monsieur R.BOUTIAZ, Ingenieur CNRS au LA7, qui a bien voulu participer à ce Jury et qui m'a toujours fourni l'aide matérielle dont j'ai eu besoin,

Monsieur J.P.SCHOELLKOPF, chercheur au LA7, qui a bien voulu participer à ce Jury et qui m'a patiemment prodigué de nombreuses suggestions durant toute la durée de cette thèse,

Je remercie également,

les membres de l'équipe de Recherche en Achitecture des Ordinateurs avec qui j'ai eu des échanges fructueux, spécialement Messieurs G.BAILLE et J.LAURENT.

Ainsi que:

Messieurs F.RIVERO Directeur, et V.RIVERO chef du Département d'Informatique à l'IUT-RC-CARACAS, qui ont soutenu et encouragé mon dernier séjour en France,

Ma femme, pour avoir eu la gentillesse et la patience de supporter mon exil du foyer pour préparer cette thèse,

..... et mes amis Péruviens et Vénézuéliens qui m'ont apporté leur aide et leurs soutien moral spécialement Madame et Monsieur CHASSANDE.

## TABLE DE MATIERES

<u>INTRODUCTION</u> .....	11
---------------------------	----

### PREMIERE PARTIE

#### I: ASPECTS METHODOLOGIQUES DE LA CONCEPTION D'UNE APPLICATION MICROPROCESSEUR.

1.) INTRODUCTION.....	21
2.) NOTIONS UTILISEES DANS LA METHODE .....	21
.1) Application microprocesseur.....	21
.2) Processus application.....	21
.3) La hiérarchie.....	22
.4) La technique de conception.....	22
3.) LA METHODE PROPOSEE.....	25
.1) L'étape de spécifications.....	27
.1) Les spécifications initiales.....	27
.2) Les spécifications formelles.....	28
.2) L'étape de conception.....	30
.1) La phase de décomposition du processus application.....	31
.2) La phase d'interprétation.....	36
.3) La phase de décomposition fonctionnelle.....	37
.1) Les fonctions logicielles à concevoir.....	37
.2) Les fonctions matérielles à réaliser.....	39
.3) L'étape de développement.....	40
.1) La phase de réalisation.....	41
.1) La sous-phase d'assemblage.....	41
.2) la sous-phase d'implantation.....	43
.2) La phase de mise au point de l'application....	44

## II: LA MISE AU POINT DANS UN CONTEXTE D'APPLICATIONS MICROPROCESSEURS.

1.)	INTRODUCTION.....	47
2.)	LA MISE AU POINT.....	48
	.1) L'acquisition.....	48
	.1) Les capteurs.....	49
	.2) Compromis économique dans les choix des capteurs...	51
	.2) La mémorisation.....	52
	.3) L'analyse.....	52
	.4) La modification.....	53
3.)	REALISATIONS DES OUTILS DE MISE AU POINT.....	54
	.1) La mise au point locale.....	54
	.2) La mise au point externe.....	56
	.3) La mise au point semi-externe.....	58
4.)	METHODES DE MISE AU POINT.....	60
	.1) Méthodes par couches.....	60
	.1) Méthodes avec outils internes.....	60
	.2) Méthodes avec outils externes.....	61
	.2) Méthodes d'examen.....	61
	.1) Méthodes stroboscopiques.....	61
	.2) Méthodes d'exécution contrôlée.....	62
	.3) Méthodes de prise de traces.....	62
5.)	OUTILS DE LA MISE AU POINT.....	63
	.1) L'oscilloscope.....	63
	.2) L'analyseur d'états logiques.....	64
	.3) L'analyseur d'états logiques pour micro-ordinateurs....	64
	.4) La console de test.....	64
	.5) Simulateur de mémoires mortes.....	65
	.6) Système de développement.....	65
	.7) Mini ou Gros ordinateur.....	66
	.8) Emulateur/Simulateur.....	66
	.9) Le cordon ombilical.....	67

**III: UNE METHODE DE DESCRIPTION DE SYSTEMES**  
**MICROPROCESSEURS: MICROD**

1.) INTRODUCTION.....	71
2.) LE FORMALISME PMS.....	72
.1) Les composants.....	72
.2) Description d'une machine en PMS.....	73
3.) LA MACHINE INFORMATIQUE.....	74
.1) Représentation de la machine.....	74
.1) Les niveaux de décomposition du matériel.....	75
.2) Les niveaux de décomposition du logiciel.....	76
.3) La relation d'implantation.....	77
4.) LA METHODE: <u>MICROD</u> .....	78
.1) Découpage fonctionnel.....	79
.2) Description informelle de MICROD.....	80
.3) La description MICROD d'une machine.....	80
5.) LA DESCRIPTION DES MACHINES.....	81
.1) La description d'un micro-système.....	81
.1) Le bloc processeur.....	83
.1) Le module de traitement.....	83
.2) Le module de contrôle.....	85
.3) Le BUS interne.....	88
.2) Le bloc périphérique.....	89
.1) Le module interface.....	89
.2) Le BUS externe.....	90
.3) Le module fonction.....	91
.3) Le logiciel du micro-système.....	93
.4) Les relations d'implantation interne dans un micro-système.....	95
.2) La description d'un macro-système.....	96
.3) La description d'un système complexe.....	99



DEUXIEME PARTIE: UN PERIPHERIQUE D'AIDE AU DEVELOPPEMENT  
D'APPLICATIONS MICROPROCESSEURS ET DE  
SUPERVISION EN TEMPS REEL.

1.)	INTRODUCTION.....	103
2.)	DEMARCHES DANS LA REALISATION DU PERIPHERIQUE.....	103
.1)	La couche noyau.....	103
.2)	Les couches fonctionnelles.....	104
.1)	La mémoire de masse.....	104
.2)	Le processeur "espion".....	104
3.)	LA DESCRIPTION FONCTIONNELLE DU PERIPHERIQUE.....	105
IV:	<u>UN PROCESSEUR DE GESTION DE MEMOIRES CCD.</u>	
1.)	INTRODUCTION.....	109
2.)	LA TECHNOLOGIE DES MEMOIRES à CCD.....	110
.1)	Rappel sur la cellule CCD.....	110
.2)	Organisation interne des mémoires CCD.....	113
.1)	Organisation en série: serpentine.....	113
.2)	Organisation série-parallèle-série:SPS.....	116
.3)	Organisation matricielle:LARAM.....	117
3.)	LA MEMOIRE CCD CHOISIE: <u>L'INTEL 2416</u> .....	119
.1)	Organisation des mémoires CCD 2416.....	119
.2)	Méthodes d'accès.....	121
.1)	Méthode verticale:adressage par génératrice.....	121
.2)	Méthode horizontale:adressage par piste.....	122
4.)	LE PROCESSEUR CCD.....	123
.1)	Le problème.....	123
.2)	Description du processeur CCD.....	124
.1)	Le contrôleur CCD.....	125
.1)	Le générateur des phases.....	126
.2)	L'automate de contrôle.....	127
.1)	L'automate en mode d'accès génératrice...	129
.2)	L'automate en mode d'accès par piste.....	132
.3)	Le module mémoire CCD.....	135
.2)	Le logiciel-CCD,.....	137

5.) LA REALISATION: <u>le contrôleur prototype</u> .....	138
.1) La générateur des phases.....	139
.1) L'horloge maître.....	139
.2) Le générateur des phases proprement dit.....	140
.2) Le fonctionnement du contrôleur.....	143
6.) ROLE DU PROCESSEUR CENTRAL ( <u>P.central</u> ).....	149

#### IV: LE CONTROLEUR DISQUE SOUPLE.

1.) INTRODUCTION.....	153
2.) LE CONTROLEUR DISQUE SOUPLE.....	154
.1) Le matériel disque souple.....	154
.1) Le circuit contrôleur: <u>FD1771</u> .....	156
.2) L'interface P.central.....	157
.1) L'horlogerie.....	158
.2) BUS de données.....	158
.3) Sélection des registres internes du FD1771..	159
.4) Test d'une commande.....	160
.5) Techniques de transfert.....	161
.3) L'interface unité disque souple.....	163
.2) Le logiciel disque souple.....	168
.1) Caractéristiques du logiciel disque souple.....	169
.2) Le moniteur d'essais du disque souple.....	174
3.) EXTENSION DU CONTROLEUR.....	176
.1) Extention matérielle.....	176
.2) Extention logicielle.....	180

VI: UN PROCESSEUR EN TEMPS REEL APPLIQUE AU DEVELOPPEMENT  
LOGICIEL ET MATERIEL DES APPLICATIONS  
MICROPROCESSEURS: ANALYD

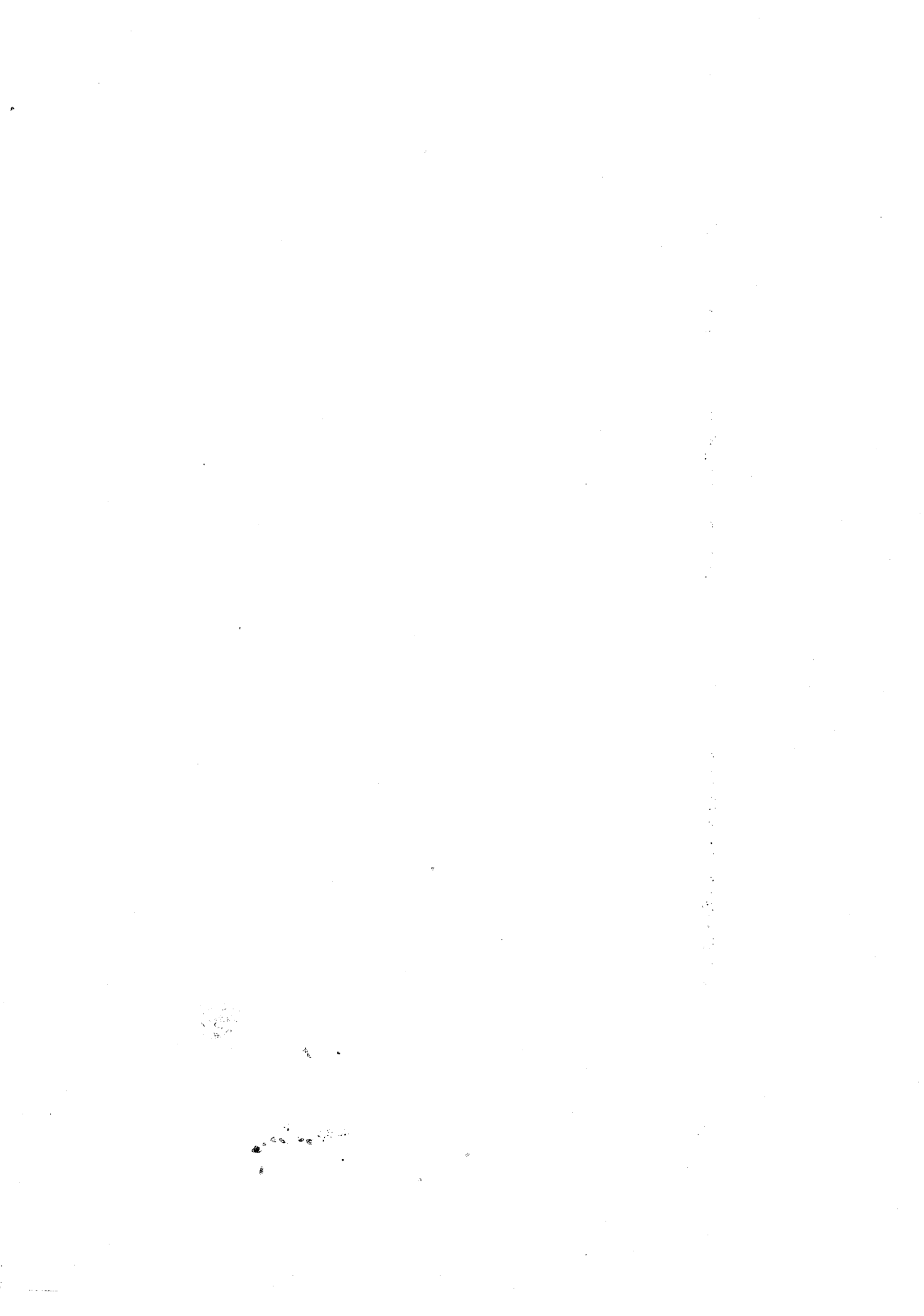
1.)	INTRODUCTION.....	183
2.)	GENERALITES.....	184
3.)	LE PROBLEME.....	185
4.)	LA CONCEPTION D'ANALYD.....	186
	.1) Le bloc processeur.....	187
	.2) Le bloc périphérique.....	187
	.1) La partie contrôle.....	187
	.1) Le système mono-profil.....	188
	.2) Le système multi-profils.....	192
	.2) La partie enregistrement.....	196
5.)	LE SYSTEME ANAMAD.....	197
	.1) Le matériel ANAMAD.....	198
	.1) Le bloc processeur.....	198
	.2) Le bloc spécialisé.....	199
	.1) Les capteurs.....	199
	.2) Le mécanisme de contrôle.....	199
	.3) Le mécanisme d'enregistrement.....	203
	.2) Le logiciel ANAMAD.....	203
6.)	L'EXTENSION D'ANALYD.....	205
	.1) Organisation de l'enregistrement.....	206
	.2) Description du tampon.....	206

CONCLUSIONS..... 217

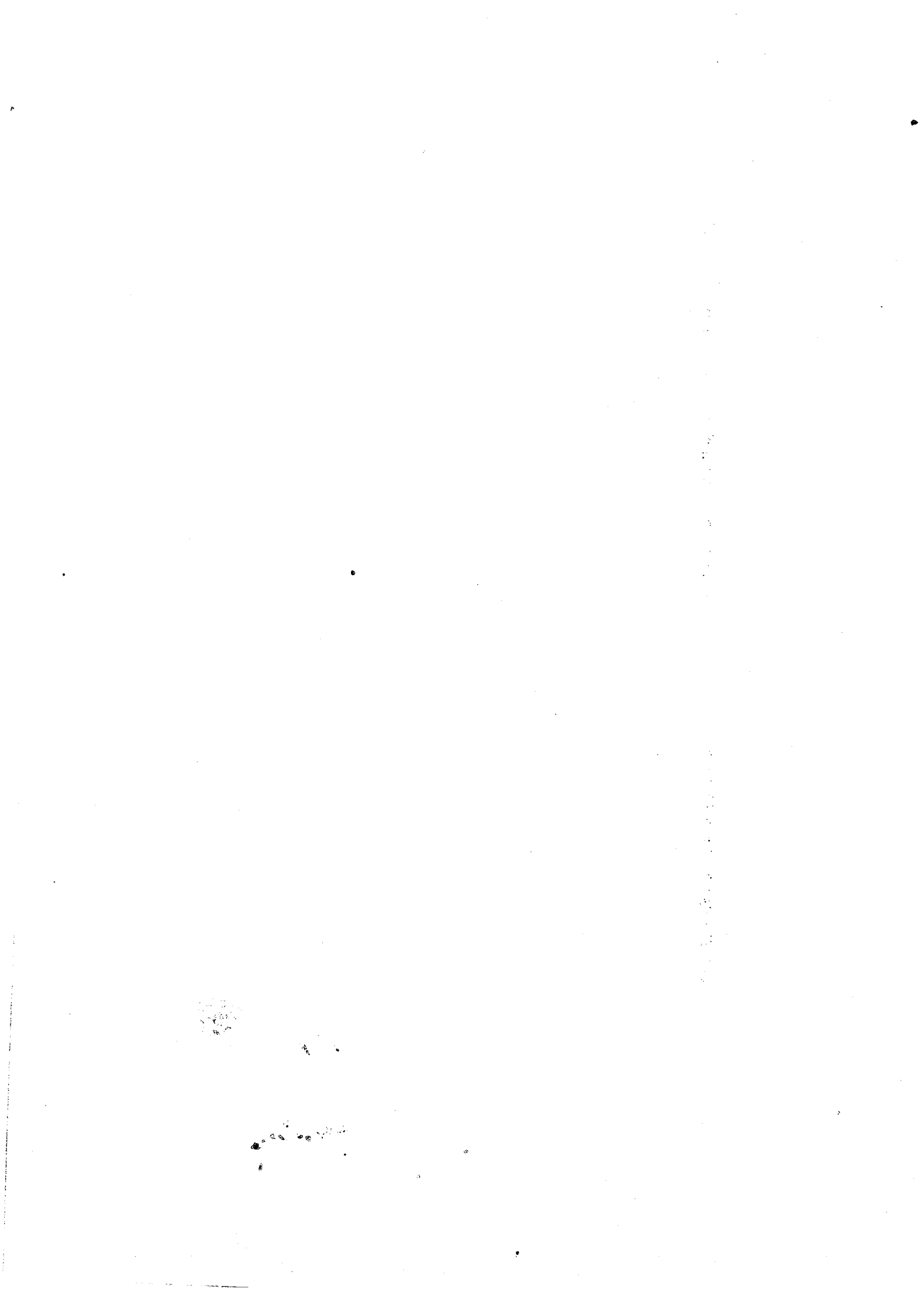
BIBLIOGRAPHIE..... 221

ANNEXE I..... 235

ANNEXE II..... 263



INTRODUCTION



## INTRODUCTION

### ENVIRONNEMENTS DES PROJETS MICROPROCESSEURS.

La nécessité d'avoir un organe d'aide au développement des applications de microprocesseurs au sein de l'équipe d'architecture de GRENOBLE, nous a amené à la conception d'un périphérique d'aide au développement d'applications à base de microprocesseurs et de supervision en temps réel.

Pour réaliser ce périphérique, nous avons commencé par implanter autour d'un microprocesseur, de type MOTOROLA MC6800, une série de fonctions matérielles (circuits périphériques, mémoires, décodeurs, ..) et des fonctions logicielles (superviseurs, ..) pour concevoir un outil de base constituant le noyau système du périphérique.

Du fait de la complexité des applications microprocesseurs (processeurs spécialisés, multi - microprocesseurs), cet outil ne peut satisfaire toutes les exigences des concepteurs (étudiants, chercheurs), par exemple: la sauvegarde des informations et la surveillance en temps réel d'applications à base de microprocesseurs.

On y a ajouté par la suite des extensions matérielles et logicielles, développées à partir du noyau système, pour avoir un outil plus sophistiqué d'aide à l'enseignement et à la recherche. Ces extensions, composants du périphérique, sont les suivantes:

- modules mémoires additionnelles,
- coupleur de disque souple,
- coupleur de mémoires à couplage de charges,
- processeur d'analyse en temps réel.



L'implantation physique du noyau système et de ces extensions, nous a permis de dégager une méthode de conception et une méthode de description grammaticale du matériel et du logiciel d'une application donnée en considérant sa facilité de réalisation.

### LES LIGNES DIRECTRICES.

Les problèmes de développement d'une application à base de microprocesseurs, de la définition de son architecture jusqu'à sa mise au point donnent les lignes directrices de cette thèse.

### LES DEMARCHES SUIVIES.

Notre travail de recherche méthodologique ne s'est pas limité au développement d'idées sur la manière d'aborder un problème; nous nous sommes aussi interrogés sur l'usage d'expressions grammaticales pour mieux définir l'architecture des applications à base de microprocesseurs.

Cette réflexion nous a conduit à développer l'idée centrale de formulation algorithmique d'une application microprocesseur, qui constitue un lien entre les méthodes que l'on veut appliquer et l'environnement informatique dont on dispose.

Ce lien apparaît à deux niveaux:

- au niveau de l'application directe des méthodes par le concepteur. Les notions algorithmiques permettent de trouver une méthode de décomposition d'une application microp processeur (projet) vis-à-vis de l'environnement informatique et notamment des langages de représentation de l'algorithme.

- au niveau même de notre démarche méthodologique. Les notions algorithmiques nous permettent de concrétiser nos idées sous la forme d'une représentation grammaticale de l'architecture d'une application.

Nous utilisons ces méthodes et les différents processus de mise au point, comme des outils d'aide à la réalisation d'une application concrète.

## PRESENTATION DE LA THESE.

Le travail que nous présentons ici est le fruit d'une réflexion qui s'adresse à tous les représentants des groupes concernés par l'architecture de systèmes à base de microprocesseurs: universitaires, chercheurs, enseignants et étudiants, mais aussi aux industriels.

Pour la clarté de l'exposé, l'ouvrage est divisé en deux parties:

### - Première partie: METHODES.

La première partie, destinée aux méthodes, est composée de trois chapitres.

Le premier est consacré aux aspects méthodologiques de la conception d'une application à base de microprocesseurs.

Il est basé sur la décomposition hiérarchique d'une application, que nous appellerons par la suite "processus application" en un ensemble de processus. Chaque processus est défini par un algorithme à partir duquel se développera l'architecture matérielle et logicielle de l'application.

Le deuxième contient une présentation des processus, méthodes et outils de mise au point.

Dans le troisième, nous proposons une méthode de description grammaticale de l'architecture matérielle et logicielle d'une application.

- Deuxième partie: REALISATION.

A partir des méthodes exposées dans la première partie, nous présentons la réalisation complète de chaque composant du périphérique d'aide au développement d'applications à base de microprocesseurs.

Cette deuxième partie est composée de trois chapitres:

Le chapitre IV, est consacré à l'étude des mémoires à couplage de charges (Charge Coupled Device) (CCD) puis à l'étude et la réalisation d'un coupleur pour connecter ces dispositifs à un système à base de microprocesseurs.

Le chapitre V, est basé sur la réalisation d'un coupleur/processeur de disque souple.

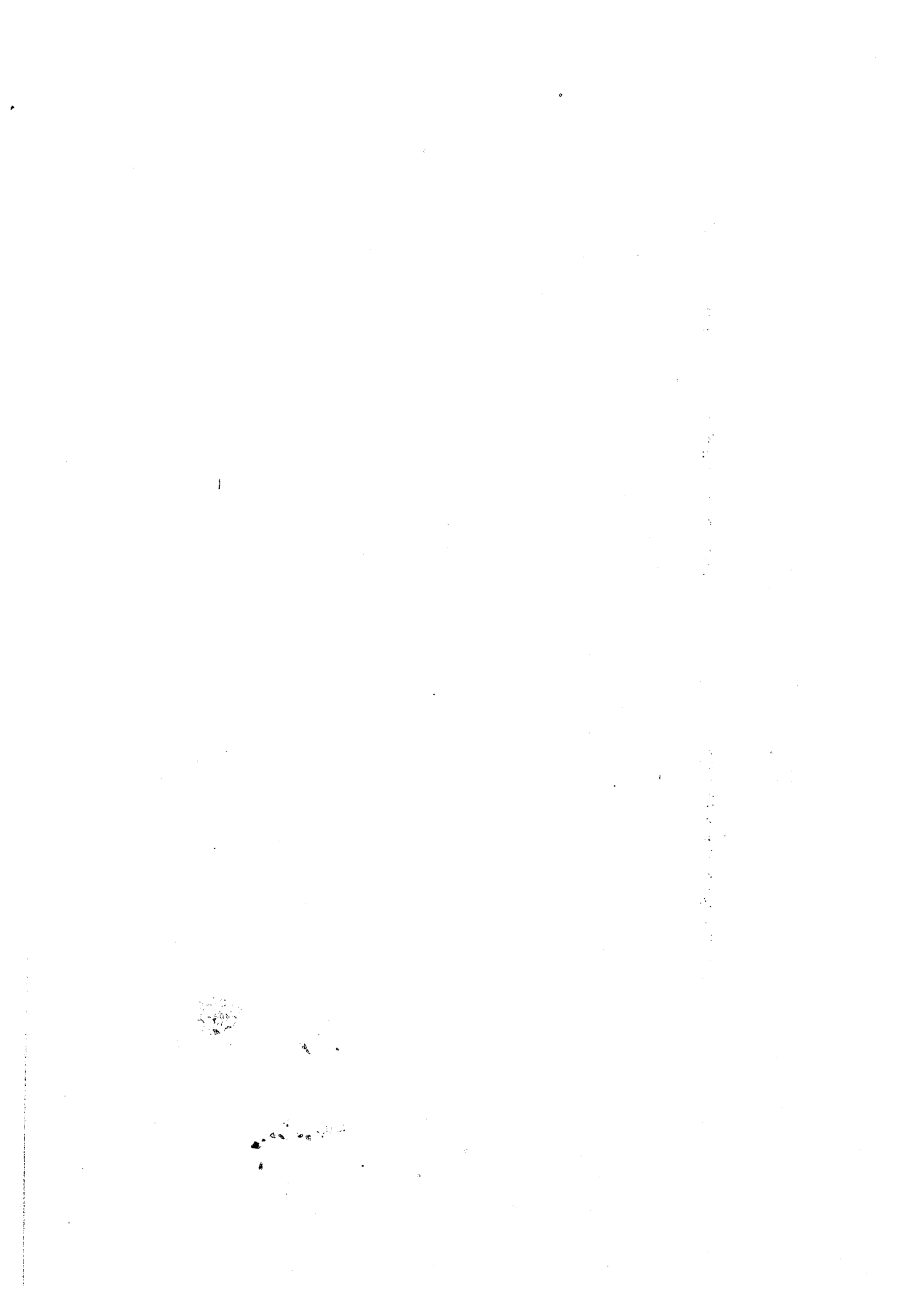
Le chapitre VI, est destiné à la conception et à la réalisation d'un processeur d'analyse en temps réel appliqué au développement matériel et logiciel des applications à base de microprocesseurs.

Les annexes décrivent le matériel et le logiciel du noyau système du périphérique.



PREMIERE PARTIE

ASPECTS METHODOLOGIQUES DE LA CONCEPTION  
ET DE LA DESCRIPTION D'APPLICATIONS  
MICROPROCESSEURS.



## CHAPITRE I

### ASPECTS METHODOLOGIQUES DE LA CONCEPTION D'UNE APPLICATION MICROPROCESSEUR.

- 1.) INTRODUCTION.
- 2.) NOTIONS UTILISEES DANS LA METHODE.
- 3.) LA METHODE PROPOSEE



## CHAPITRE I

### ASPECTS METHODOLOGIQUES DE LA CONCEPTION D'UNE APPLICATION MICROPROCESSEUR.

#### 1.) INTRODUCTION.

La conception d'applications à base de microprocesseurs pose encore des problèmes de méthodologie. Ceux-ci proviennent principalement d'une mauvaise organisation dans l'enchaînement des différentes étapes constituant la conception de telles applications.

En me basant sur l'expérience acquise au sein de l'équipe d'architecture des ordinateurs à GRENOBLE, je propose une méthode d'aide au concepteur pour le développement de telles applications.

#### 2.) NOTIONS UTILISEES DANS LA METHODE.

##### 2.1) Application microprocesseur.

Il s'agit de l'ensemble des mécanismes logiciels et matériels consistant en la mise en oeuvre d'un ou de plusieurs microprocesseurs pour satisfaire une application.

##### 2.2) Processus application

C'est le ou les algorithmes qui définissent le comportement d'une telle application.

### 2.3) La hiérarchie.

Dans tout processus d'application, il est possible de distinguer une structure hiérarchisée dans la conception d'une application à base de microprocesseurs (CAJ-74) (FIC-76).

Pour cela, on part du niveau le plus élevé de l'algorithme qui définit l'application à réaliser; puis on détaille sa structure dans le niveau qui suit et ainsi de suite jusqu'à atteindre le niveau des réalisations pré-existantes: on ne peut pas traiter un niveau si le niveau antérieur n'est pas défini ou réalisé (ANC-74).

La hiérarchie peut provenir de l'existence d'un graphe acyclique d'appel inter-modules, d'un arbre généalogique dans la décomposition des processus, d'une structure d'adressage arborescente ..etc.

Dès le début de la conception d'un processus application quelconque, il est très important de bien définir cette hiérarchie afin de pouvoir déterminer un découpage en niveaux d'abstraction (ALF-79).

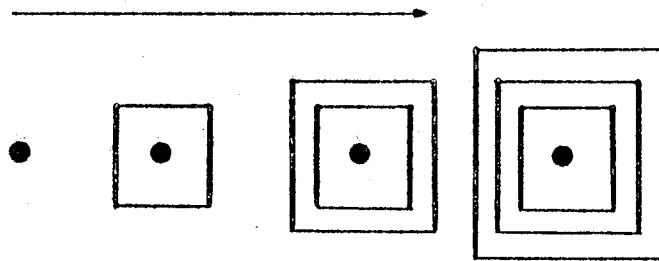
### 2.4) La technique de conception.

Il reste ensuite à choisir une technique de conception, par exemple:

- la technique descendante qui progresse des spécifications (projet) aux fonctions primitives (moyens): c'est-à-dire que l'on doit spécifier toutes les fonctions nécessaires pour construire un niveau n avant de définir le niveau inférieur n-1.

- la technique ascendante qui utilise les fonctions déjà définies (moyens) dans un niveau n-1 pour bâtir les fonctions du niveau supérieur n.

technique de conception ascendante



technique de conception descendante

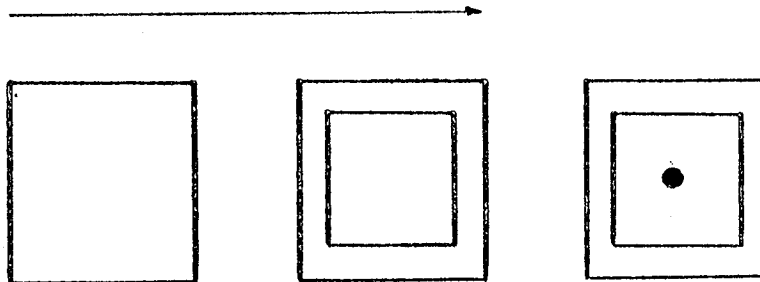


Figure 1.1 Les différentes techniques de conception.

La technique descendante paraît plus naturelle: il semble en effet préférable de connaître le rôle précis d'un sous-ensemble avant d'en déterminer son fonctionnement interne.

Cependant cette technique présente le risque de choisir a priori des caractéristiques attractives qu'il peut être difficile, voire impossible, de réaliser sur le matériel disponible.

Par contre une technique ascendante peut avancer à travers plusieurs niveaux de conception avant que l'on se rende compte que le produit final n'est pas celui que l'utilisateur avait demandé.

Les bons architectes de machines ont en général une idée intuitive de ce que seront les niveaux extrêmes dans un processus de conception, et ils utilisent ce savoir d'une manière consciente ou inconsciente pendant le processus de conception. Par exemple un architecte expérimenté, qui utilise la technique de conception descendante ne va inclure dans les niveaux supérieurs aucune caractéristique qu'il ne soit sûr de pouvoir réaliser d'une façon ou d'une autre dans les niveaux inférieurs.

### 3.) LA METHODE PROPOSEE.

La méthode proposée est basée sur l'analyse de la démarche de conception d'une application à base de microprocesseurs.

La méthode se présente comme une démarche descendante caractérisée par les étapes suivantes:

- étape de spécification,
- étape de conception proprement dite,
- étape de développement de l'application.

Ces étapes permettront de travailler à partir d'un problème dont les limites sont bien définies (algorithme), de choisir la solution matérielle la mieux adaptée, de la caractériser et de la réaliser.

La charge de ces différentes étapes doit être décroissante: une définition très précise des spécifications de l'application facilitera la conception et simplifiera sa mise au point et son développement. Dans le cas contraire un phénomène de divergence risque d'avoir lieu.

L'imprécision des spécifications entraîne une conception plus ou moins approximative d'où une mise au point problématique et un développement difficile voir impossible; ce qui amènera à préciser les spécifications, revoir la conception et refaire la mise au point.

On a schématisé les diverses étapes et phases de la conception d'une application sur la figure 1.2.

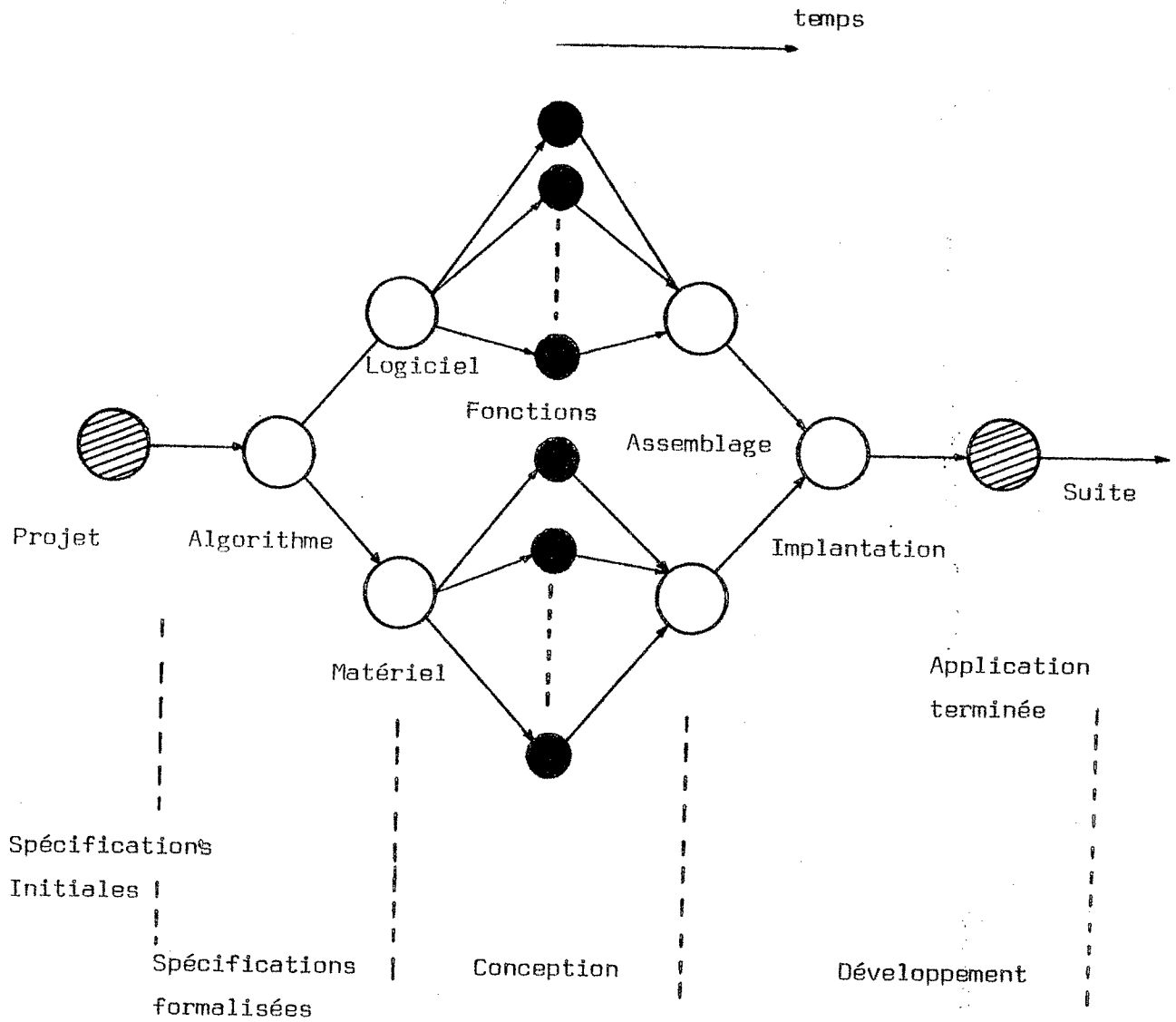


Figure 1.2 Différentes étapes et phases de la conception d'une application à base de microprocesseur.

### 3.1) L'ETAPE DE SPECIFICATION.

Les spécifications d'une application destinée à satisfaire un besoin, impliquent toujours l'existence de contraintes et de particularités qui lui sont propres.

Nous pouvons en distinguer deux :

- les spécifications initiales,
- les spécifications formelles.

#### 3.1.1) Les spécifications initiales.

Les spécifications initiales servent à brosser les grandes lignes de l'application ainsi que ses particularités principales, par exemple: critiques et comparaisons d'applications similaires ou équivalentes, étude des objectifs, des contraintes et des possibilités.

Elles sont regroupées dans le cahier des charges; celui-ci précise l'application telle qu'elle est vue par l'utilisateur.

Y sont également considérées les contraintes techniques qui dépendent de la nature de l'application; celles-ci peuvent être temporelles (durée de l'étude), électriques (consommation, puissance disponible en sortie), géométriques (taille), d'environnement (bruit, ambiance), de sécurité, de fiabilité, et de temps réel.

### 3.1.2) Les spécifications formelles.

La formalisation des spécifications donne une description précise du comportement de l'application.

Les spécifications formelles sont produites par la mise en forme des spécifications initiales à l'aide d'un outil de description formel, à partir de l'ensemble des spécifications initiales de l'application. Elles sont considérées comme les spécifications d'entrée de l'étape de conception.

Ces spécifications définissent le, ou les, processus application: en effet une représentation algorithmique du(ou des) processus application, permet de définir l'application d'une manière structurée.

D'une manière générale, l'outil formel utilisé est:

- soit un langage algorithmique de haut niveau de type ALGOL, PASCAL, SIMULA, ..
- soit un langage de forme graphique: organigrammes, GRAFSET, réseaux de PETRI, ..

L'utilisation de l'une ou de l'autre forme dépend des goûts du concepteur et de l'utilisateur et ne revêt qu'une importance secondaire. Ce qui est primordial c'est de décrire très précisément et de façon bien détaillées les actions au niveau algorithmique.



La figure ci-dessous, montre les différentes étapes de l'établissement des spécifications.

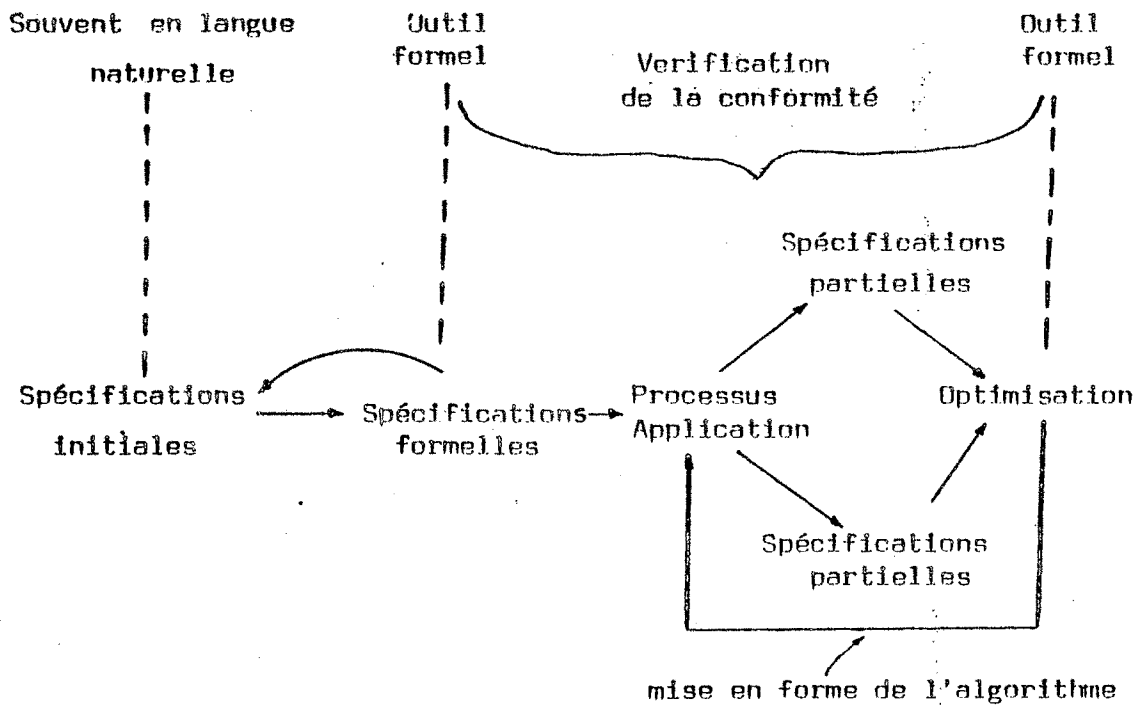


Figure 1.3 Différentes étapes de l'établissement des spécifications.

### 3.2) L'ETAPE DE CONCEPTION.

C'est l'étape la plus importante de la démarche car l'expérience et la technique du concepteur servent à choisir les solutions qui semblent apporter le maximum de garanties quant à la faisabilité, le coût, l'efficacité et la fiabilité de l'application envisagée. L'étape de conception est divisée en trois phases :

- la phase de décomposition du processus application,
- la phase d'interprétation,
- la phase de décomposition fonctionnelle.

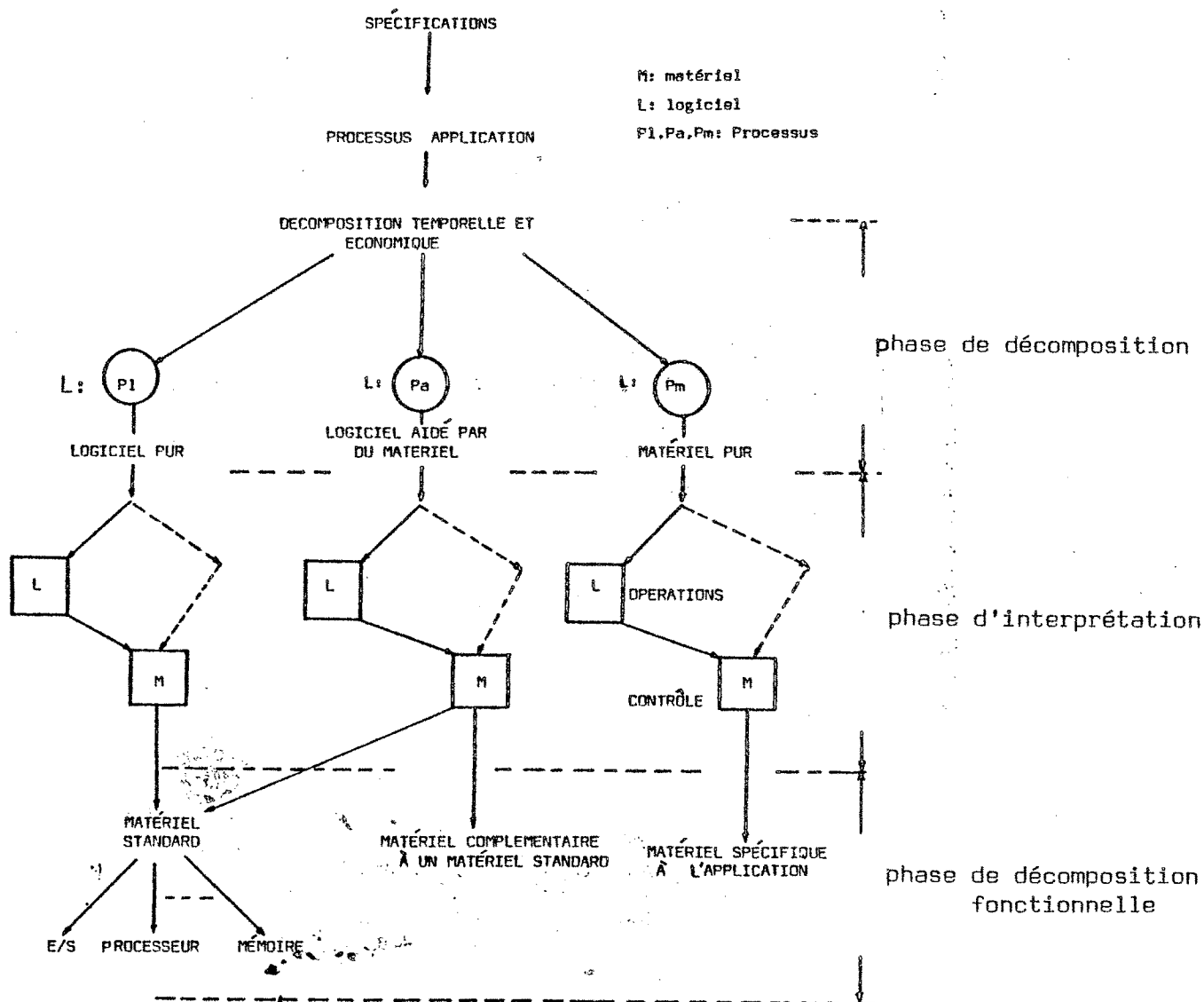


Figure 1.4 Les étapes de conception.

### 3.2.1) La phase de décomposition du processus application.

Un problème se pose dans la technologie de réalisation des fonctions. En effet celles-ci peuvent être implantées soit en logiciel soit en matériel. Le choix de leur technologie de réalisation est fortement influencé par des critères tant économiques (liés à la technologie) que temporels (liés à la fréquence des signaux manipulés).

Au niveau de la pré-série les fonctions réalisées en logiciel seront en général moins onéreuses que celles réalisées en matériel. Ceci peut n'être plus vrai pour la production, en effet cela dépend du nombre de systèmes à réaliser car la phase de développement étant amortie sur un nombre plus ou moins grand de systèmes.

Cependant certaines fonctions rapides ne pourront, pour des raisons technologiques, être réalisées que grâce à du matériel.

#### Critères techniques utilisés dans la décomposition.

##### - Le critère temporel.

Ce critère est le plus important dans la décomposition du processus application, il servira à bien distinguer les parties logicielles et matérielles ainsi qu'à en aider l'implantation (figure 1.5).

Cette décomposition dépend du rapport ( $K=H/F$ ) entre la fréquence ( $F$ ) des signaux à manipuler (générer / reconnaître / traiter) et la fréquence de l'horloge de base ( $H$ ); par exemple:

- \* la vitesse de lecture d'un octet,
- \* la reconnaissance d'un chiffre hexadécimal.

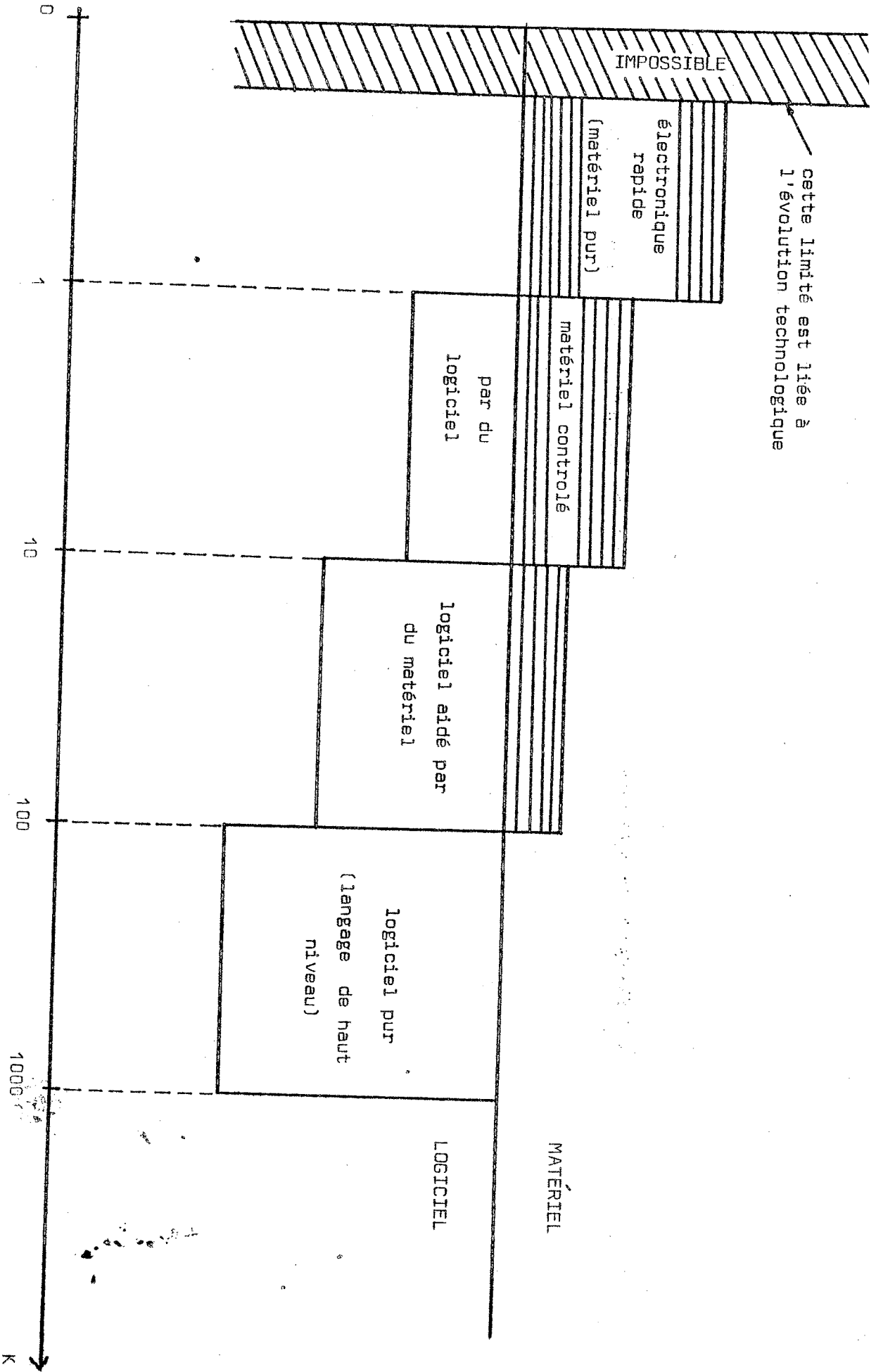


Figure 1.5 L'aspect temporel de la décomposition du processus application.

L'utilisation du matériel spécifique peut:

\* soit servir à accélérer du logiciel en lui fournissant de nouvelles instructions si la fréquence des signaux manipulés se situe entre  $H/10$  et  $H/100$ , par exemple: modification d'instructions pour effectuer un transfert direct disque souple  $\leftrightarrow$  mémoire.

\* soit constituer des fonctions indépendantes contrôlées ou paramétrées par le logiciel lorsque cette fréquence est comprise entre  $H$  et  $H/10$ : par exemple le mécanisme de DMA (Direct Memory Acces).

\* soit constituer un ensemble matériel à part ayant généralement pour but de se comporter comme un réducteur de fréquence lorsque cette fréquence dépasse  $H$ : par exemple les registres de sérialisation et désérialisation d'un disque.

En utilisant ce critère, le processus application(PA) est découpé en une série de processus coopérants en considérant le compromis entre les contraintes économiques et temporelles.

Ces processus coopérants sont les suivants:

\* le processus logiciel(P1) qui représente la partie du travail nécessaire à l'application et qui peut être réalisée uniquement par du logiciel compte tenu de la faible fréquence relative des signaux à manipuler.

\* le processus logiciel aidé par du matériel(Pa) représente la partie du travail qui peut être réalisée par du logiciel à condition que celui-ci manipule du matériel qui augmente la rapidité d'exécution de certaines séquences: par exemple l'utilisation d'une mise en halte matérielle du processeur déclenchée par une instruction et libérée par un évènement.

\* le processus matériel (Pm) représente la partie du travail qui doit être réalisée uniquement par du matériel compte tenu de la haute fréquence relative des signaux manipulés: par exemple les mécanismes de sérialisation et désérialisation d'un coupleur disque.

- le critère fonctionnel.

Les processus P1, Pa, Pm définis grâce au critère temporel peuvent aussi être décomposés par rapport à leurs tâches à effectuer en des processus coopérants (décomposition fonctionnelle)(figure 1.6).

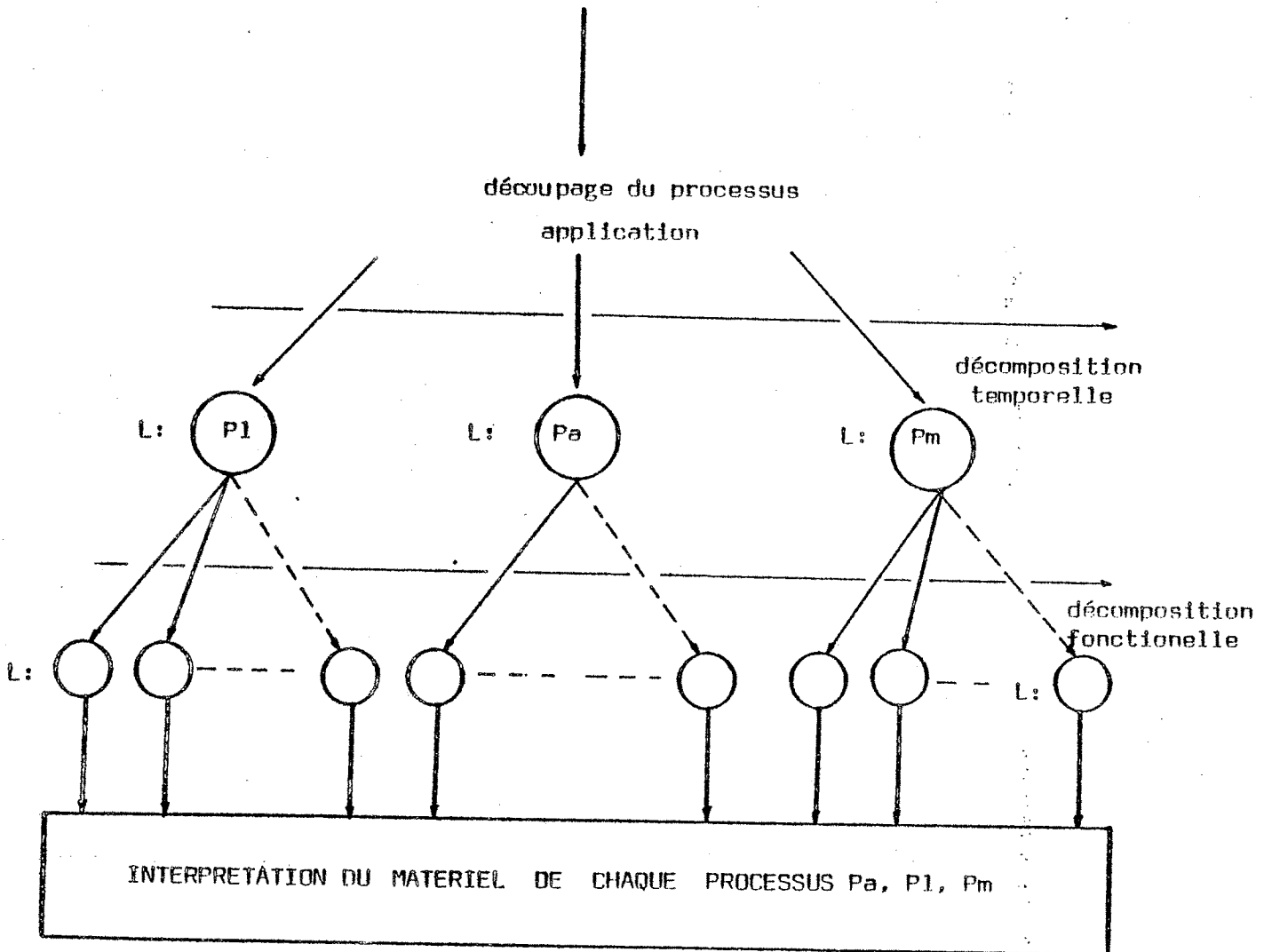


Figure 1.6 La phase de décomposition du processus application.

### 3.2.2) La phase d'interprétation.

L'objectif de cette phase est d'isoler les parties logicielles et matérielles à concevoir et celles qui résulteront de l'assemblage de pièces standard.

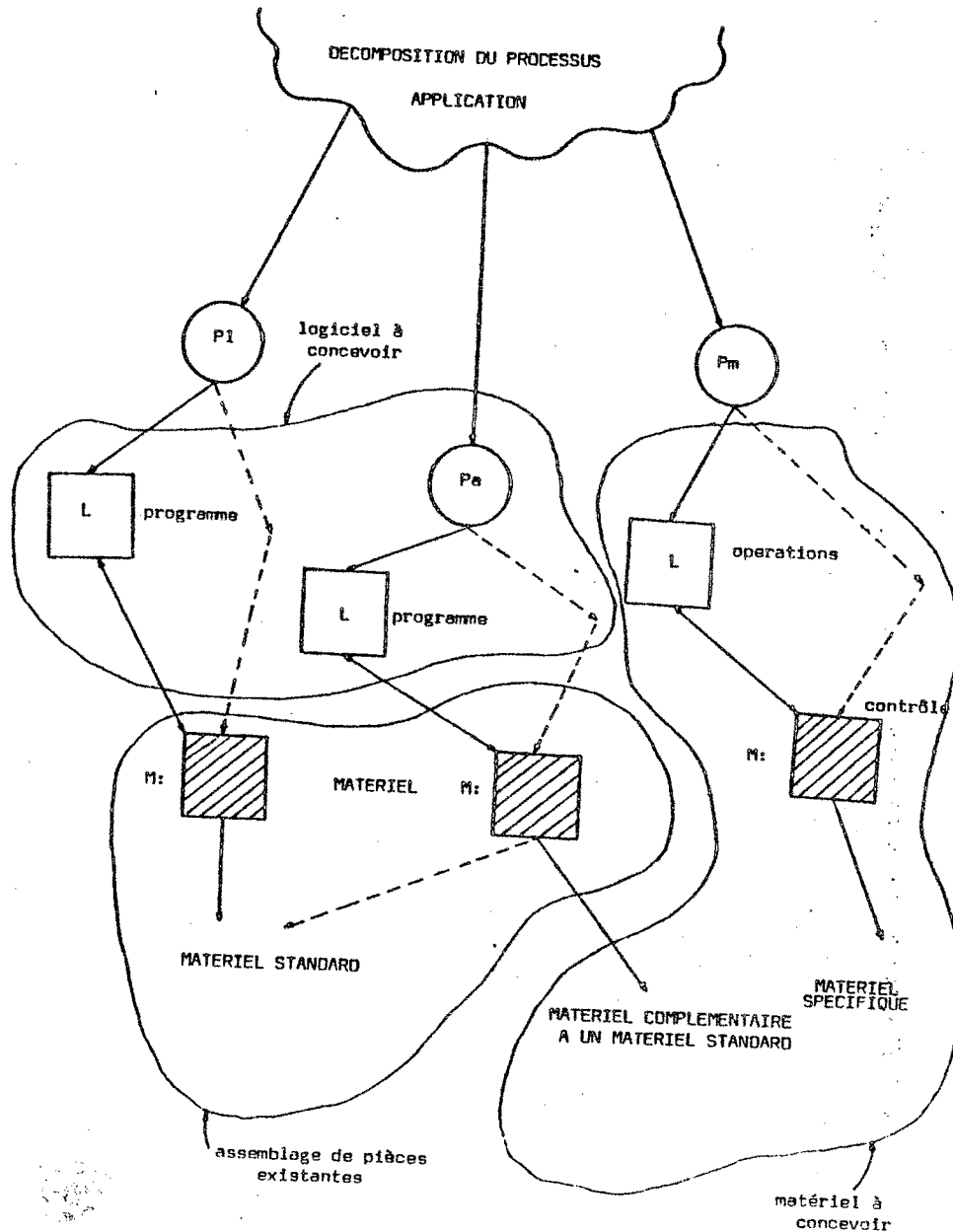


Figure 1.7 Les différentes parties à concevoir ou résultant d'assemblages de pièces existantes.



### 3.2.3) La phase de décomposition fonctionnelle.

#### 3.2.3.1) Les fonctions logicielles à concevoir.

Les fonctions logicielles sont créées à partir d'une codification des instructions des algorithmes du P1 et du Pa. Cette codification peut être faite sous forme mnémonique propre à un microprocesseur particulier.

Il est nécessaire d'isoler les fonctions spécifiques qui formeront des modules de service; par exemple:

- d'entrée et de sortie,
- de communication,
- de contrôle,
- de mise au point,
- ...etc

Il n'existe pas de méthode infallible permettant de passer d'un problème à son programme.

En fait, un problème peut être exprimé sous la forme de programmes de différentes manières, selon le critère choisi et l'inspiration du programmeur.

Citons comme critère d'optimisation:

- le nombre d'instructions générées,
- la vitesse d'exécution,
- la facilité de réalisation, ..etc.

Il faut éviter, autant que possible, les finesses permettant de trop optimiser les programmes et qui rendent leur compréhension ultérieure difficile.

Par contre, ces finesses sont nécessaires pour certaines phases critiques d'applications en temps réel (voisin de H/100): lecture par programme d'un octet sur un disque souple par exemple.

### 3.2.3.2) Les fonctions matérielles à réaliser.

Ce matériel constitue la couche d'interprétation du logiciel ainsi que les mécanismes matériels propres à l'application.

Il y aura donc plusieurs choix possibles dans la sélection de l'architecture de l'application, choix qui aboutiront principalement à la sélection des composants informatiques, des mécanismes d'interconnexion et des fonctions matérielles complémentaires.

- le matériel "standard" constitue le matériel classique utilisé dans des applications microprocesseur:
  - \* microprocesseur: 8 bit ou 16 bits,
  - \* circuits mémoires: ROM, RAM, PROM, ..
  - \* circuits périphériques d'entrée et de sortie,
  - \* circuits de contrôle des interruptions, ..etc.
- les fonctions complémentaires à un matériel standard:
  - \* l'accès direct à la mémoire.
  - \* modifications des instructions,
  - \* synchronisations, ..etc
- les fonctions de matériel pur spécifiques à l'application.

Une fois déterminées les fonctions matérielles et logicielles, on analyse les problèmes de coût. Un des objectifs de l'étape de conception consiste à prévoir les coûts tant matériels que logiciels.

### 3.3) L'ETAPE DE DEVELOPPEMENT.

Nous considérons cette étape comme étant constituée par les phases suivantes:

- la phase de réalisation,
- la phase de mise au point de l'application.

Ces phases sont schématisées dans la figure ci-dessous.

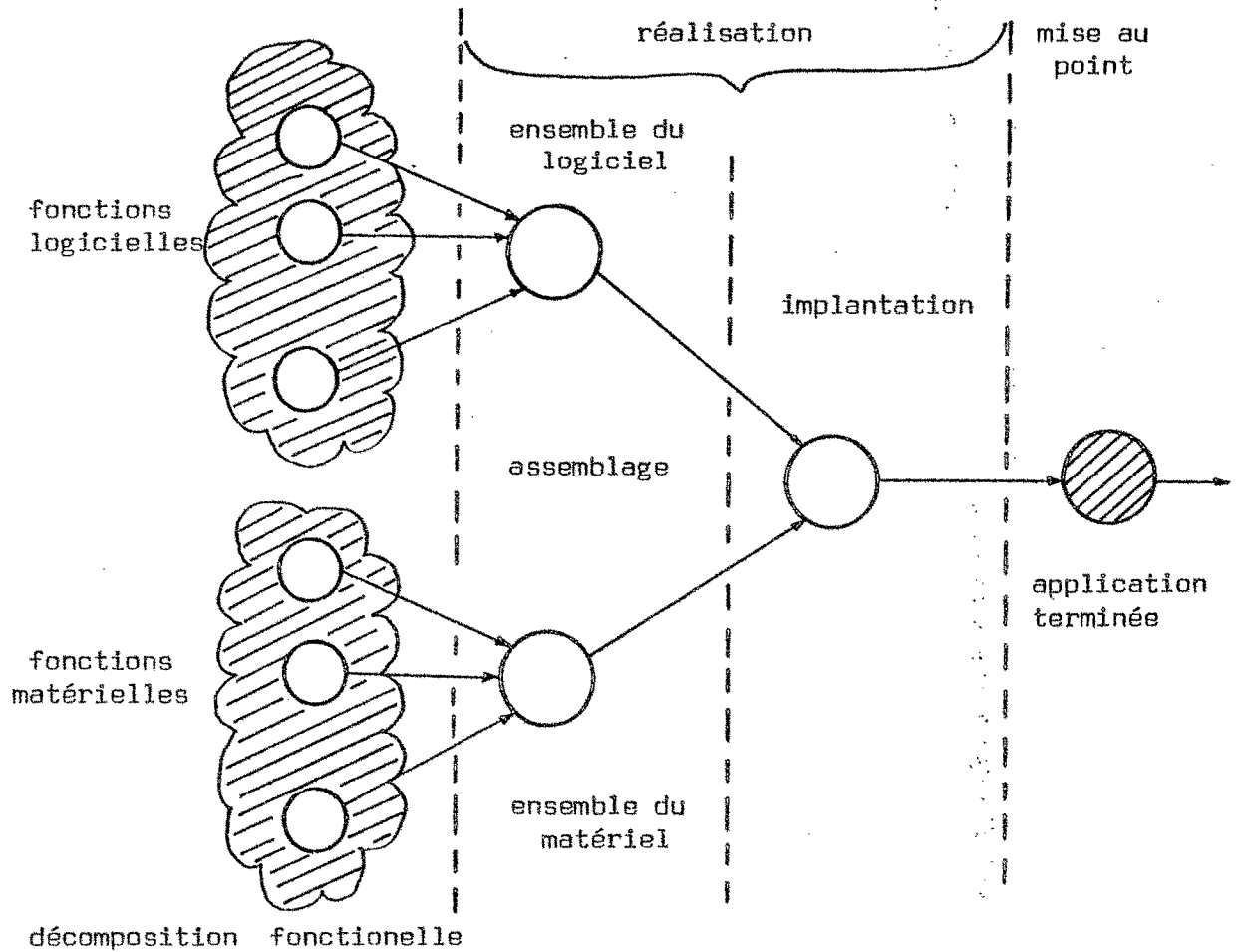


Figure 1.8 Les différentes phases du développement d'une application.

### 3.3.1) La phase de réalisation.

La phase de réalisation consiste en la réalisation des fonctions logicielles et matérielles en considérant toujours la facilité de mise au point, ainsi que la transmission des responsabilités pour l'étape de mise au point et les modifications ultérieures.

Cette phase est fondamentale et nécessite une très grande rigueur pendant sa réalisation. On peut la découper en deux sous-phases:

- l'assemblage,
- l'implantation.

#### 3.3.1.1) La sous-phase d'assemblage.

Elle consiste en la mise au point individuelle et la mise en place de chaque fonction du logiciel et du matériel de l'application.

Cette sous-phase se décompose en deux étapes qui se déroulent simultanément:

- l'étape d'assemblage du matériel.

Elle consiste à connecter les éléments testés individuellement dans un ordre de mise au point lié à l'évolution du logiciel. D'une manière générale l'assemblage se traduira par la mise en place hiérarchisée de fonctions matérielles.

La mise en place de l'élément  $n$  de chaque fonction ne s'effectue que lorsque l'élément  $n-1$  a été lui-même mis en place et testé.

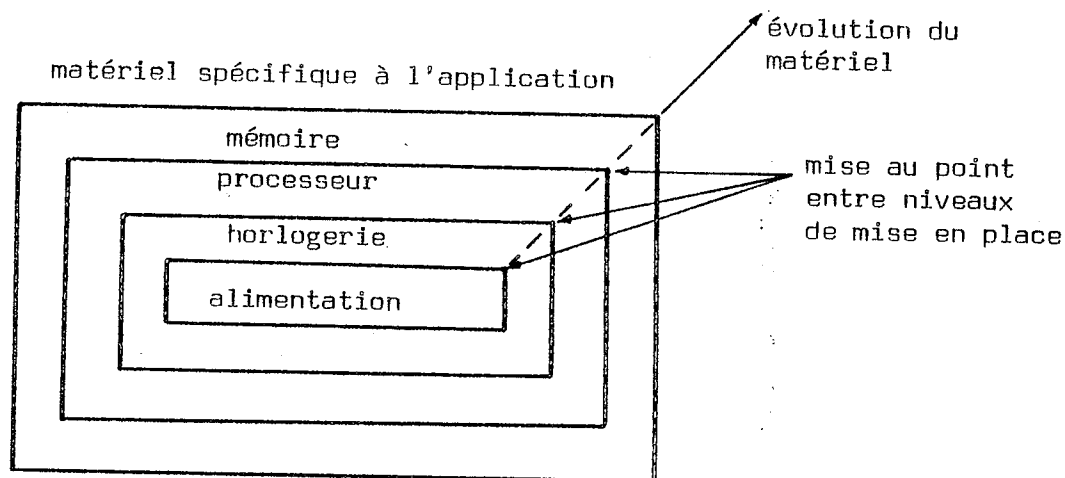


Figure 1.9 Hiérarchie de réalisation et de mise au point du matériel.

La mise au point du matériel nécessite un logiciel de test à écrire et un matériel de test existant.

- l'étape d'assemblage du logiciel.

C'est l'étape de mise au point et d'assemblage des fonctions du logiciel d'application écrites et testées séparément. Le déroulement de cette mise au point dépend de l'application et de l'environnement matériel disponible pour mettre le logiciel en service.

La mise au point du logiciel nécessite un matériel de test (pré-existant) et un logiciel de test à écrire (environnement des modules).

Cet environnement peut être constitué par:

- \* un ordinateur de grande puissance,
- \* un mini-ordinateur, un système de développement,...

Il faut remarquer qu'en général cette mise au point s'effectue en dehors de l'environnement matériel de l'application (c.f.II).

### 3.3.1.2) La sous-phase d'implantation.

Elle constitue l'implantation du logiciel dans l'environnement matériel propre à l'application pour obtenir la configuration réelle de l'application (c.f.III).

Le logiciel peut être mis en place sur le matériel de différentes manières:

- inscription de mémoires mortes,
- chargement du logiciel application dans des mémoires vives(RAM) à partir de moyens externes comme:
  - \* un environnement de développement relié par un cordon ombilical,
  - \* des périphériques de dialogue: terminaux,...
  - \* des périphériques de chargement: cassettes, ...

Dans cette phase, il faut considérer le coût de mise au point du logiciel ; par exemple celui des assemblages répétés des programmes à l'aide d'un gros ordinateur et celui du temps du programmeur.

### 3.3.2) La phase de mise au point de l'application.

C'est la phase de mise en service de l'application. Elle consiste à obtenir le fonctionnement simultané du logiciel et du matériel vis-à-vis du milieu extérieur prévu.

L'intégration des deux constitue donc, la phase critique du cycle de développement: l'épreuve de vérité.

#### Remarque.

L'élaboration du logiciel et celle du matériel doivent être effectuées de façon conjointes et progressives à l'aide d'un organe de test approprié, sinon le concepteur n'aura aucun moyen de savoir où et pourquoi des problèmes ont surgi lors de la phase d'implantation. Ceci est développé dans le chapitre II.





## CHAPITRE II

### LA MISE AU POINT DANS UN CONTEXTE D'APPLICATIONS MICROPROCESSEURS.

- 1.) INTRODUCTION.
- 2.) LA MISE AU POINT.
- 3.) REALISATION DES OUTILS DE MISE AU POINT.
- 4.) METHODES DE MISE AU POINT.
- 5.) OUTILS DE LA MISE AU POINT.

## CHAPITRE II

### LA MISE AU POINT DANS UN CONTEXTE D'APPLICATIONS MICROPROCESSEURS.

#### 1.) INTRODUCTION.

L'étape de mise au point est la plus importante dans le développement d'une application quelle qu'elle soit. On entent par importance non pas la charge en travail mais le fait que cette étape va déterminer le bon fonctionnement de l'application (c.f. I.3).

Ceci se vérifie pour les applications à base de microprocesseurs.

Les différentes solutions utilisées dans le développement d'une application à base de microprocesseurs dépendent de la démarche utilisée pour sa mise au point.

Le choix de la démarche est influencé par la complexité de l'application et par les moyens de mise au point propres à l'utilisateur; cette sélection implique le choix d'une technique, d'une méthode, et d'un ou de plusieurs outils de mise au point.

C'est pour cela que ce chapitre sera destiné à l'étude et à l'analyse des techniques, méthodes et outils de mise au point.

## 2.) LA MISE AU POINT.

La mise au point d'une application consiste en la suppression des malfunctions résultant d'erreurs, de mauvaises connaissances des moyens, de l'environnement de l'application et aussi d'une mauvaise interprétation du cahier des charges.

Le processus de mise au point consiste généralement en une suite répétitive (cycles) d'examens et de corrections des défauts constatés.

Chaque cycle peut se décomposer en les étapes suivantes:

- l'acquisition d'informations de comportement,
- la mémorisation de ces informations,
- leur analyse,
- la modification de l'application.

Les étapes de la mise au point s'enchaînent de manière cyclique jusqu'à obtenir un fonctionnement correct de l'application.

### 2.1) L'acquisition.

Cette étape a pour but la prise d'informations sur le comportement de l'application en vue de leur analyse ultérieure.

Elle s'effectue soit de manière logicielle (programmes de mesure, capteurs logiciels) soit de manière matérielle (analyseurs, oscilloscopes, capteurs matériels).

### 2.1.1) Les capteurs.

Les capteurs sont des organes matériels, ou logiciels de prise d'informations. Ils sont placés sur l'application et envoient les résultats de leurs mesures à des organes d'examen ou de stockage lorsqu'un organe de contrôle le leur demande.

Cet organe de contrôle peut être déclenché :

- de manière périodique interne,
- de l'extérieur,
- par certaines configurations de signaux sur certains capteurs.

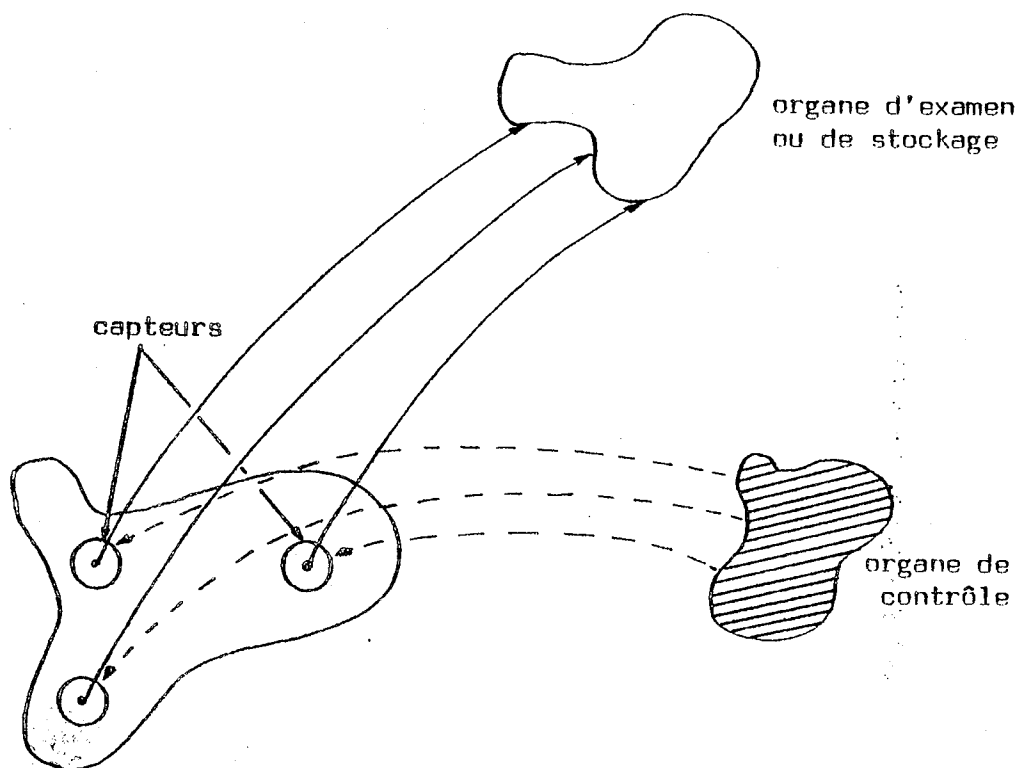


Figure 2.1 Le principe du capteur.

Il existe deux formes de capteurs:

- les capteurs matériels.

Ce sont des organes physiques externes à l'application dits d'AUSCULTATION, reliés à un organe d'affichage ou de stockage. L'organe de contrôle est alors externe à l'application.

- les capteurs logiciels.

Ce sont des sous-programmes logiciels dits d'INTROSPECTION, reliés à des modules d'affichage ou de stockage internes ou externes (reliés à l'application).

L'organe de contrôle est souvent un générateur d'interruptions de prise de mesure.

Le domaine de perception d'un capteur.

Le domaine de perception d'un capteur (réaction du capteur à un STIMULUS externe) dépend de son degré de couplage avec l'application, et de facteurs comme:

- sa technologie: temps de réaction du capteur, sa sensibilité, etc.
- la quantité d'information à lire,
- son environnement,
- son degré d'intégration dans le système: impédance,...

### 2.1.2) Compromis économique dans le choix des capteurs.

D'une part, le coût des capteurs dépend de la quantité d'informations lues à chaque requête de l'organe de contrôle.

D'autre part la lecture d'informations perturbe toujours, plus ou moins, le comportement en temps réel du système.

D'une manière générale, plus grande est la quantité d'informations lues par les capteurs à chaque cycle de mesure (connaissance de l'état instantané de la machine), plus grande est la perturbation apportée au comportement en temps réel du système examiné.

Nous pouvons considérer que ce compromis peut être vu comme l'équivalent informatique du principe d'incertitude d'HEISENBERG, qui lie la précision d'une mesure d'état (position) à celle d'un comportement dynamique (vitesse).

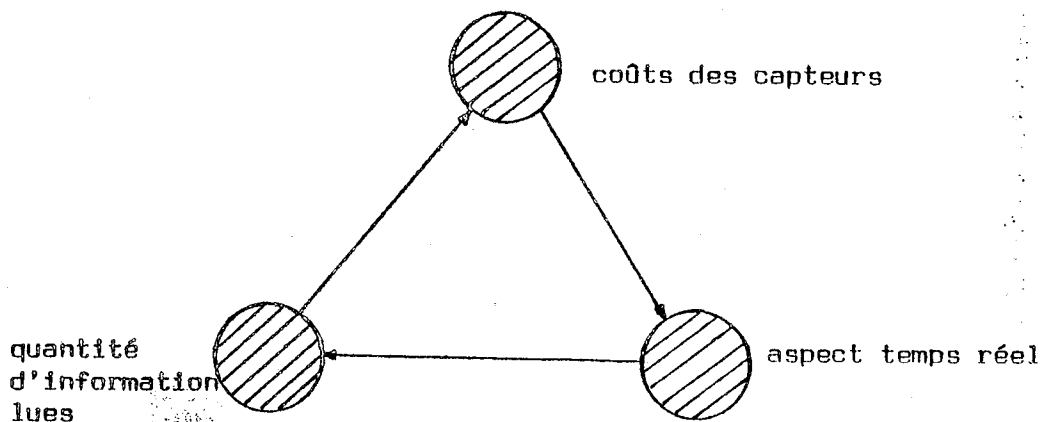


Figure 2.2, Compromis économique dans le choix des capteurs.

## 2.2) La mémorisation.

Cette étape est la phase de stockage des informations issues des capteurs.

Elle peut s'effectuer:

- soit localement à l'aide des mémoires et des périphériques de l'application elle-même (dans le cas de capteurs logiciels).

- soit à distance:

- \* dans le cas d'acquisition par des capteurs matériels connectés à un système extérieur mémorisant l'information.

- \* dans le cas de la connection indirecte de capteurs logiciels à un système externe par un lien inter-systèmes.

## 2.3) L'analyse.

Elle définit le traitement de l'information par le METTEUR au point (homme ou machine) pour en déduire la cause de mauvais fonctionnement ou pour constater le bon fonctionnement.

Cette analyse peut s'effectuer:

- sur le système lui-même muni d'un mode de mise au point,

- sur un système externe.



#### 2.4) La modification

La correction du logiciel ou du matériel de l'application se manifeste par une modification du contexte d'exécution (programme, données, matériel) qui aboutit à transformer le système initial (système objet).

Le système modifié doit être remis en test pour une nouvelle étape de mise au point.

Dans le cas de la mise au point du logiciel, la modification de l'information peut être:

-réelle: ré-inscription de mémoires mortes, modification du matériel commandé.

-virtuelle:

\* modification in situ du logiciel d'application qui doit alors être réinscrit dans des mémoires vives. Cette modification s'effectue par des outils locaux de mise au point (touches, superviseurs,...).

\* utilisation d'un contexte d'exécution extérieur à l'application, par utilisation d'un "cordon ombilical" connecté à un outil de mise au point distant.

### 3.) REALISATION DES OUTILS DE MISE AU POINT.

Les différentes techniques de réalisation des étapes de la mise au point nous permettent de distinguer:

- la mise au point locale,
- la mise au point externe,
- la mise au point semi-externe.

Chacune de ces solutions correspond à une technique de réalisation de l'application. Par exemple le système ANALYD(c.f.VI) est un outil permettant une mise au point semi-externe.

Il est aussi possible de représenter la frontière entre le monde de l'application et celui des dispositifs extérieurs raccordés pour sa mise au point(figure 2.4).

#### 3.1) La mise au point locale.

La mise au point locale d'une application comportant un microprocesseur convient surtout aux applications réalisées à l'aide d'un outil de développement détourné ou grâce à un mécano de cartes suffisamment riche et disposant de ressources adéquates.

Elle peut être faite en utilisant les facilités matérielles (périphériques,..) et logicielles (superviseurs,..) du système de développement ou disponible dans le mécano.

L'application est généralement sous le contrôle d'un logiciel de supervision permettant de surveiller l'exécution du logiciel d'application et de le corriger si cela est nécessaire.

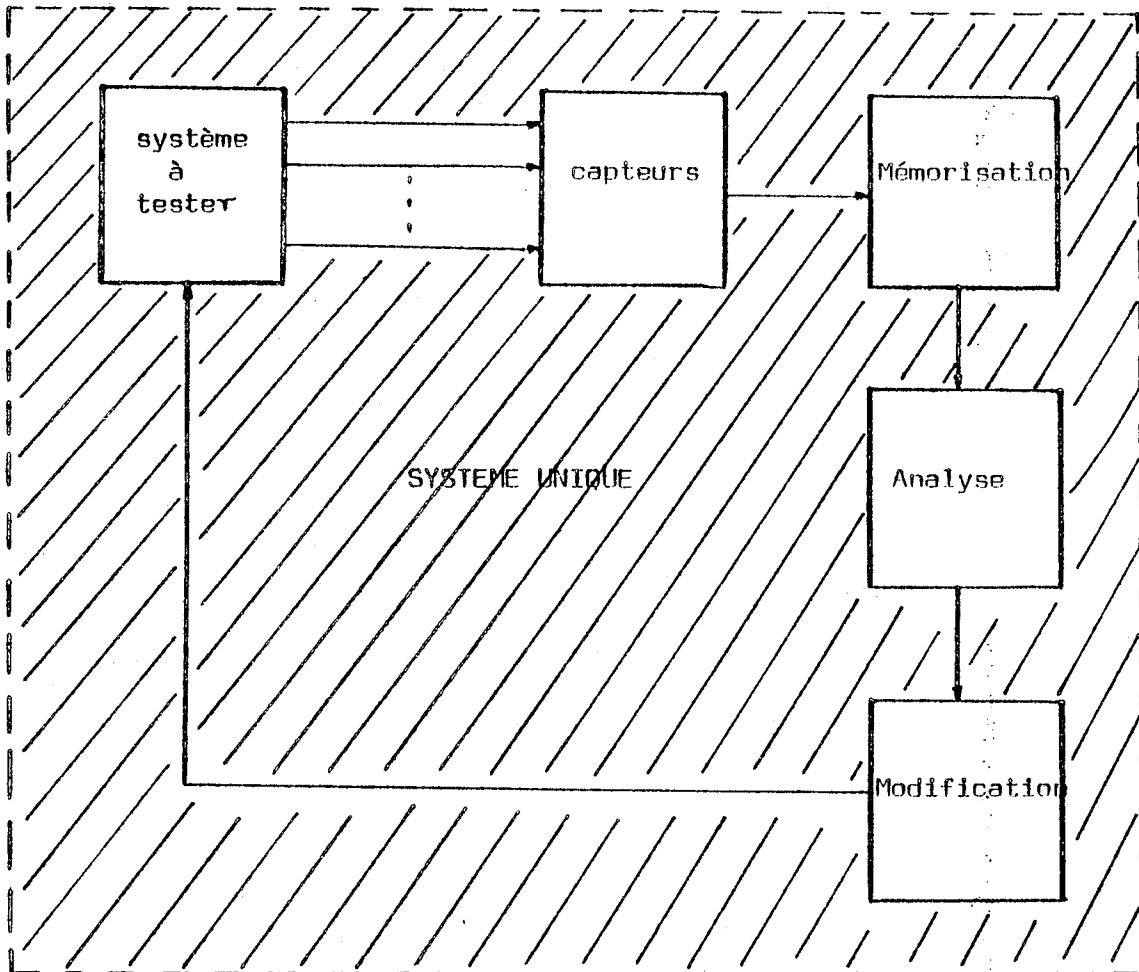


Figure 2.4 Un exemple de réalisation des étapes de la mise au point locale.

### 3.2) La mise au point externe.

L'application soumise au test est connectée à l'outil de mise au point via un cordon ombilical.

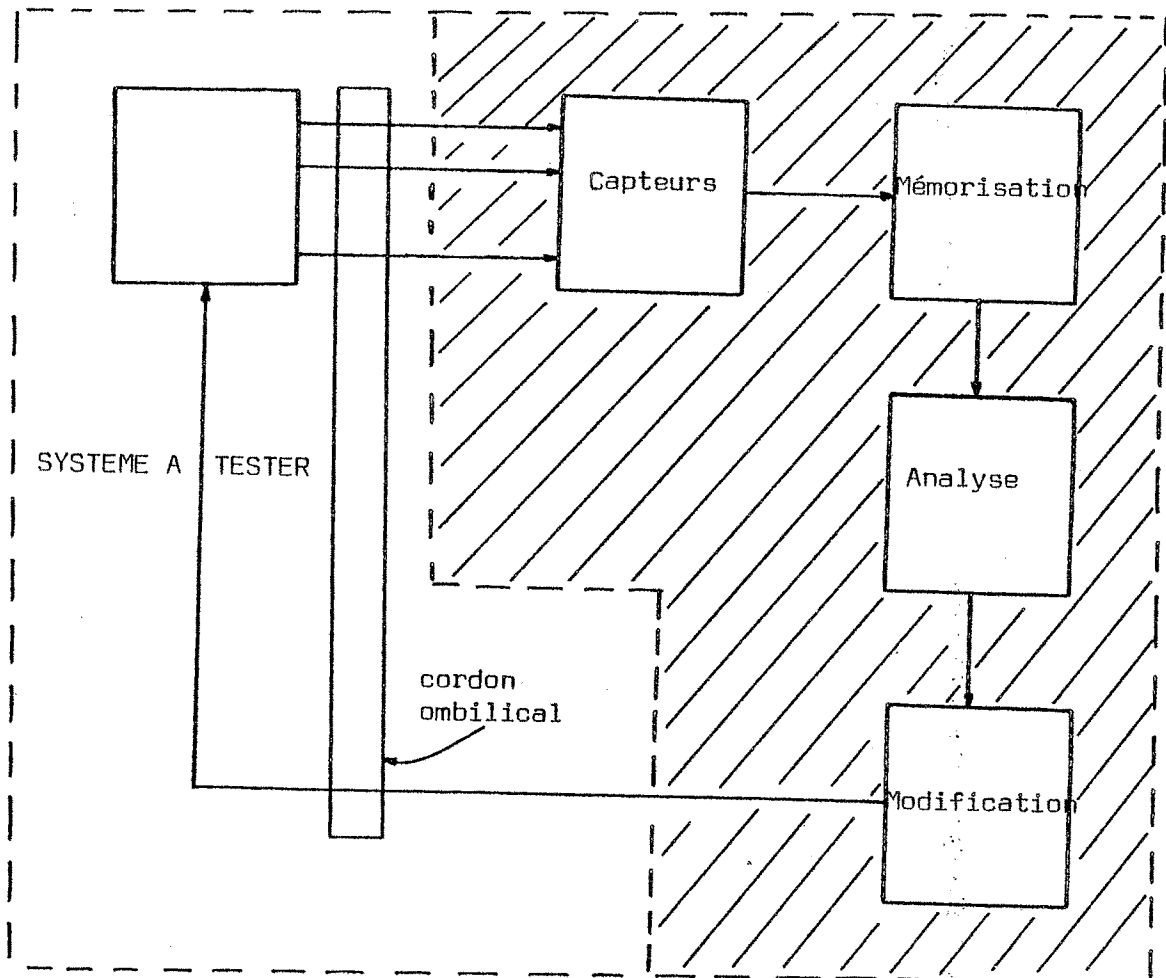


Figure 2.5 Un exemple de réalisation des étapes de la mise au point externe.

Il est possible d'observer le comportement de l'application et de modifier le logiciel de l'application à l'aide d'outils spécialisés comme :

- un système de développement utilisé pour observer le système objet et générer du logiciel. Ce logiciel est généralement chargé en accès direct à la mémoire, dans le matériel propre à l'application via un cordon ombilical qui permet également l'observation de l'exécution. Il a également des facilités matérielles pour émuler le comportement de l'application ( les systèmes TEXTRONIX, EXORCISER, .. par exemple).

- un organe de test conçu avec un matériel adapté à la supervision d'une application particulière: organe d'acquisition et de traitement des données série ou parallèle.

Ces outils sont vus par l'application d'une manière :

- active: ils peuvent modifier l'état de l'application (interruptions, arrêts temporels, ..) pour regarder son comportement.

- passive: ils sont placés en mode supervision (transparence) de manière à ne pas perturber le fonctionnement de l'application.

### 3.3) La mise au point semi-externe.

Cette technique convient surtout aux applications réalisées à l'aide d'un matériel spécifique n'offrant pas les ressources nécessaires à la mise au point mais permettant la prise d'informations (capteurs) et leur mémorisation.

Un matériel externe est utilisé pour la génération du logiciel d'application. Le système de génération du logiciel (gros ordinateur, mini-ordinateur) utilise les outils suivants: éditeurs, assembleurs, compilateurs, ..etc.

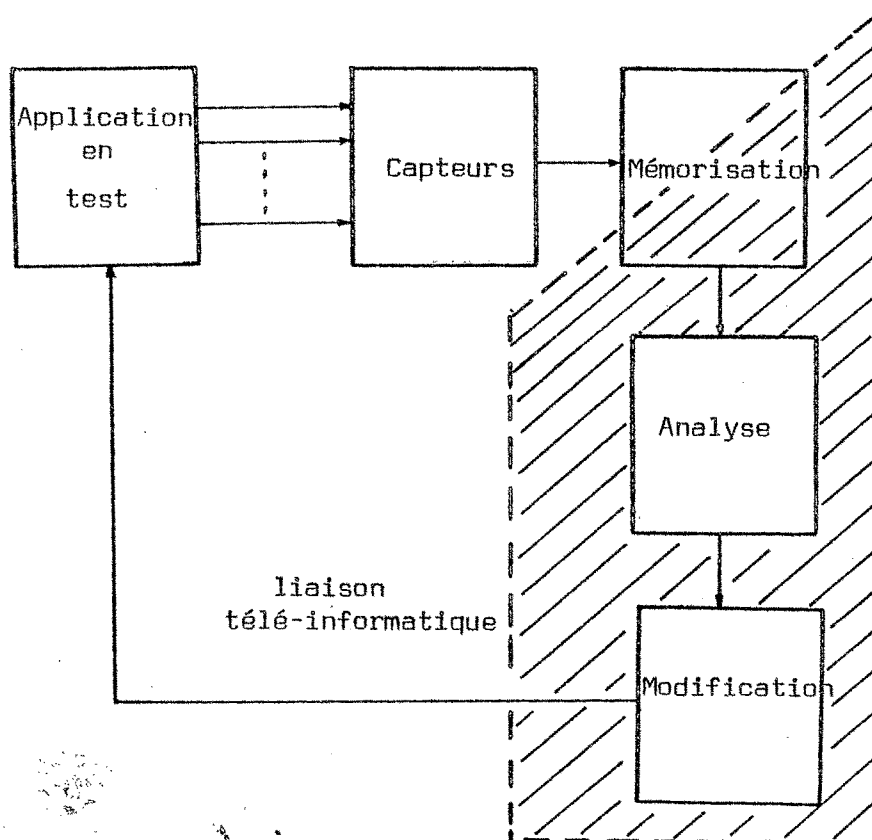


Figure 2.6 Le contexte de la mise au point semi-externe.

Les modifications logicielles nécessaires à la mise au point externe s'effectuent sous deux formes:

- soit d'une manière complètement isolées de l'application avec transport du matériel (REPRO, disquettes, ruban).
- soit reliées à l'application par des liaisons télé-informatiques: le logiciel généré est chargé dans la mémoire de l'application grâce à une liaison télé-informatique; l'application contrôle le chargement des fichiers assemblés ou sources provenant du système de génération du logiciel.

#### 4.) METHODES DE MISE AU POINT.

Il existe plusieurs méthodes de mise au point du matériel ou du logiciel. On choisira celles qui doivent être utilisées en fonction de l'évolution de la mise au point de l'application et du type de problème rencontré.

##### 4.1) Méthodes par couches.

La mise au point d'une application est découpée en couches en considérant leur facilité de mise en service.

Cette méthode appliquée à la conception d'applications est aussi utilisée pour la mise en service progressive de l'application. On commence par mettre au point la couche représentant le coeur (noyau) de l'application (micro-système: c.f.III), puis les couches qui l'entourent.

Ce noyau représente le matériel et le logiciel de base. Il devra toujours être mis au point avec des outils externes.

##### 4.1.1) Methode avec outils internes.

Une fois une couche mise en service, elle devient ainsi que les couches précédentes un outil de mise au point pour les couches suivantes.

Chaque couche devra donc contenir, en plus des fonctions propres à l'application, les outils de mise au point de la couche suivante.



#### 4.1.2) Méthode avec outils externes.

Cette méthode est utilisée dans deux cas:

- lorsqu'il n'y a pas suffisamment de ressources dans les couches déjà contruites pour permettre la mise au point (cas de la première couche).
- lorsqu'il n'y a pas les fonctions nécessaires dans les couches déjà construites pour assurer la mise au point, en particulier si l'on ne veut pas pénaliser l'application par l'existence des fonctions de mise au point qui ne lui sont pas nécessaires.

#### 4.2) Méthodes d'examen.

Elle sont utilisées pour connaître le comportement des différents organes de l'application; par exemple savoir si la mémoire est correctement adressée et si elle fonctionne bien: test de mémoires.

##### 4.2.1) Méthodes stroboscopiques.

Elles consistent à répéter un phénomène suffisamment rapidement pour le voir évoluer sans mémoriser son état. Ces techniques conviennent surtout à la mise au point des organes de base car elles fournissent beaucoup d'informations sur le comportement (simple) de ceux-ci (par exemple des informations analogiques).

Nous pouvons en distinguer deux sortes:

- excitations périodiques par le milieu extérieur; exemple: démarrages périodiques.
- fonctionnement répétitif (boucles), exemple: boucles logicielles.

L'examen proprement dit ne nécessite qu'un simple dispositif d'affichage sans mémorisation: par exemple un oscilloscope.

#### 4.2.2) Méthodes d'exécution contrôlée.

Elles consistent à contrôler l'exécution. Lorsque celle-ci atteint un état pré-défini ou effectue une action surveillée on examine son état et on essaye de déduire comment s'est déroulée l'étape d'exécution.

#### 4.2.3) Méthodes de prise de traces.

Ces méthodes consistent en l'enregistrement d'une trace d'évènements associés à une tranche d'exécution puis en leur examen dans un deuxième temps.

Elles consistent, soit à prendre des "photographies" d'une partie de l'état de l'application lorsque surviennent certains évènements puis à les examiner ultérieurement pour avoir un "film" du comportement de l'application, soit à conserver une trace de l'évolution d'un ou de plusieurs signaux caractéristiques (composantes de l'état) par leur examen périodique ou à l'occurrence d'évènements significatifs.

Les conditions de départ et d'arrêt de la prise de trace sont constituées par la reconnaissance de certains évènements tels que:

- des points d'arrêts,
- des conditions internes ou externes sur des signaux ou des composantes de l'état.

## 5.) OUTILS DE LA MISE AU POINT.

Il est souvent très difficile de savoir ce que fait exactement un logiciel ou un matériel qui ne fonctionne pas, si l'on ne dispose pas d'informations suffisantes.

La localisation des erreurs ou des pannes peut s'avérer très pénible, voire impossible dans certains cas.

La mise au point de l'ensemble logiciel et matériel se fait à l'aide d'outils spéciaux. Les outils sont soit matériels, soit logiciels, soit constitués par une combinaison matérielle et logicielle.

L'aide apportée par ces outils serait très limitée s'ils ne permettaient pas de prendre en compte le comportement des organes extérieurs auxquels est raccordée l'application examinée.

Ci-après, on décrira d'une manière générale les différents outils manipulés dans la mise au point d'une application.

### 5.1) L'oscilloscope.

L'oscilloscope est l'outil le plus élémentaire d'aide à la mise au point d'applications microprocesseurs; c'est l'outil d'analyse stroboscopique par excellence. Cet appareil permet un examen détaillé de comportements répétitifs simples.

Il sert à la mise au point des mécanismes physiques de base du système (horlogerie, mécanismes d'accès, ..etc) mais est rapidement incapable d'aider à la mise au point des couches suivantes.

### 5.2) L'analyseur d'états logiques.

Cet appareil permet l'enregistrement de l'historique booléen de quelques signaux électriques de l'application. L'enregistrement s'effectue :

- soit périodiquement grâce à une horloge propre à l'appareil,
- soit au rythme de l'horloge de l'application.

Le nombre  $n$  de pas d'enregistrement étant limité, l'appareil ne retient que les  $n$  pas suivants ou précédents un événement électriquement détectable (ou répartis de part et d'autre). Il peut aussi effectuer des comparaisons de l'état des signaux analysés avec un état de référence (c.f.VI). Il ne faut pas le confondre avec un oscilloscope car il ne fournit aucune information de nature analogique.

### 5.3) L'analyseur d'états logiques pour micro-ordinateurs.

Cet outil est une variante des analyseurs digitaux qui possède de plus la possibilité de traduire la séquence des signaux d'accès d'un microprocesseur à son programme sous la forme de mnémoniques d'instructions, c'est-à-dire passer du code binaire au code symbolique (désassemblage).

### 5.4) La console de test.

Connectée, grâce à un superviseur logiciel, à l'application la console de test est utilisée pour observer le comportement du système. Cet outil ressemble à un panneau de contrôle de micro-ordinateurs.

Il permet d'envoyer des ordres au processeur (points d'arrêts,...), de recevoir des informations de contrôle provenant du processeur et de visualiser l'état de la machine (adresses, données, registres,...).

La console de test peut ne pas exister sur l'application, c'est alors un dispositif portable qui est vu par le système d'une façon active.

L'application est placée en mode superviseur après la rencontre d'un ou de plusieurs événements par exemple par une écriture ou une lecture à une adresse donnée, par une interruption logicielle ou matérielle ou par une requête issue de la console.

#### 5.5) Simulateur de mémoires mortes.

Cet appareil vient substituer aux mémoires mortes de l'application des mémoires vives externes, qu'il devient alors possible de charger à partir d'un support informatique (ruban, cartes, lignes de communication) grâce à un périphérique connecté à cet appareil.

#### 5.6) Système de développement.

C'est un microprocesseur équipé d'un logiciel système et d'un logiciel de génération de programmes plus ou moins complexe comprenant: éditeurs, assembleurs, compilateurs, interpréteurs, éditeurs de liens, chargeurs et systèmes de gestion de fichiers (système informatique traditionnel).

### 5.7) Mini ou Gros ordinateur.

On les utilise en traitement multi-utilisateurs car ils ont un vaste support logiciel:

- plusieurs assembleurs,
- compilateurs,
- éditeurs de texte,
- simulateurs.

Ce sont des outils performants et puissants, uniquement destinés au développement du logiciel mais dont l'utilisation est généralement coûteuse.

### 5.8) Emulateurs/Simulateurs.

C'est le moyen d'exécuter sur un système de développement (émulateur) ou sur un mini ou gros ordinateur (simulateur) le logiciel de l'application. La simulation du comportement du milieu extérieur de l'application et les conflits résultants des hypothèses sur l'agencement des ressources déjà faites sur le système hôte posent toujours des problèmes: cas d'auto-émulation des systèmes de développement classiques par exemple.

#### Remarque.

Il convient de remarquer que la machine de génération de logiciel n'a aucune raison d'être identique à celle utilisée pour l'application ou son émulation.

### 5.9 ) Le cordon ombilical.

Il permet d'une part de substituer certaines ressources figées de l'application (telles que des ROM) par des ressources modifiables du système de développement (RAM), d'étendre l'ensemble des ressources de l'application par celles du système de développement et d'autre part d'examiner le comportement de l'application au moyen d'un ensemble de capteurs matériels qui lui sont généralement associés.

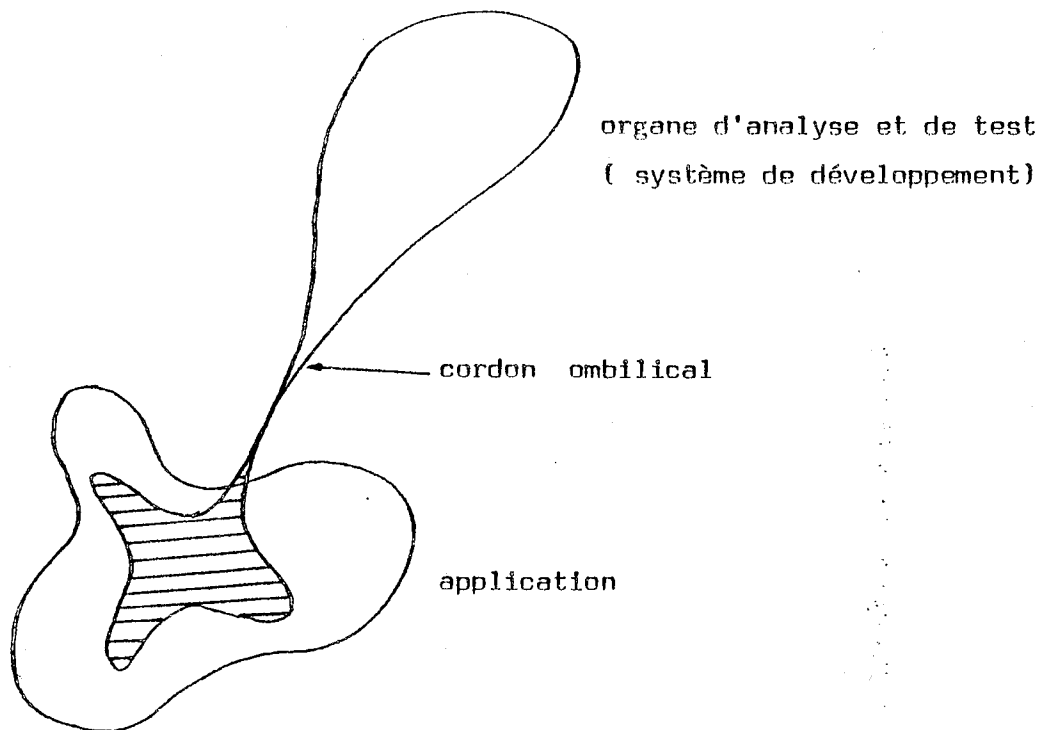


Figure 2.7 Le contexte de l'activité modification.

## CHAPITRE III

### UNE METHODE DE DESCRIPTION DE SYSTEMES MICROPROCESSEURS: MICROD

- 1.) INTRODUCTION.
- 2.) LE FORMALISME PMS.
- 3.) LA MACHINE INFORMATIQUE.
- 4.) LA METHODE: MICROD
- 5.) LA DESCRIPTION MICROD DES MACHINES.



## CHAPITRE III

### UNE METHODE DE DESCRIPTION DE SYSTEMES MICROPROCESSEURS : MICROD

#### 1.) INTRODUCTION.

IVERSON, FALKOFF ET SUSSENGUTH ont utilisé le langage APL pour décrire le comportement du matériel de l'ordinateur IBM/360(IFS-64)

BELL et NEWELL(BEN-71) ont développé un formalisme PMS (Processor, Memory, Switch), plus adapté que le précédent, pour décrire l'assemblage des différents composants matériels d'un ordinateur; sans parler des nombreux langages de description du matériel (COL ,CASSANDRE ,LASCAR,...).

Ces formes de description restent au niveau d'un formalisme et ne se présentent pas comme des méthodes d'aide au concepteur pour réaliser une machine micro-informatique (application quelconque à base de microprocesseurs).

Notre objectif, dans ce chapitre, sera de présenter une méthode de description grammaticale, voisine de PMS, pour aider à mieux comprendre la structure matérielle et logicielle des machines: MICROD( MICROsystem Description).

## 2.) LE FORMALISME PMS.

BELL et NEWELL (BEN-70) représentent une machine comme un ensemble de composants matériels (primitives) interconnectés.

### 2.1) Les composants.

Sept composants de base représentent les primitives de PMS.

- Memory (M): un composant qui stocke des informations.
- Link (L): un composant qui véhicule des informations d'un composant à un autre.
- Control (K): un composant qui supervise le fonctionnement des autres composants.
- Switch (S): un composant qui établit un lien entre composants.
- Transducer (T): un composant qui modifie la nature des entrées de l'information à partir d'une référence donnée: par exemple l'interface d'un écran de visualisation.
- Data-operation (D): Un composant qui travaille sur des données propres à l'environnement matériel (M,L,K,S,T,..)
- Processor (P): Un composant qui interprète un programme pour exécuter une série d'opérations séquentielles. Il est lui-même constitué par une série de composants tels que M,L,K,D,T, avec en plus le contrôle nécessaire pour obtenir des informations de la mémoire (M).

Une expression permet d'indiquer qu'un module, voire un composant, est constitué d'un assemblage de modules ou de composants plus élémentaires: exemple P=D-K. Cette notion est récursive.

## 2.2) Description d'une machine en PMS.

L'architecture d'une machine est décrite de la manière suivante:

machine ::= Mp-Pc-I-X

où

Mp: mémoire principale

Pc: processeur central

I : interfaces périphériques

X : environnement extérieur qui peut être éventuellement une ou plusieurs machines.

### 3.) LA MACHINE INFORMATIQUE.

Une machine informatique est un ensemble d'éléments interconnectés, eux-mêmes formés d'une série de composants assemblés (M,L,K,S,T,P).

Une machine informatique évoluée est organisée en niveaux d'interprétation imbriqués qui peuvent être représentés comme des couches concentriques: une couche n'a d'existence que par les couches qu'elle englobe(SCH-77).

La conception d'une machine informatique est généralement menée du niveau le plus abstrait (spécifications) vers le niveau le plus concret (réalisations), parce qu'il est plus aisé de concevoir une nouvelle couche à partir d'une machine existante que de définir une machine abstraite à partir d'une couche abstraite.

#### 3.1) Représentation de la machine.

Une machine peut être représentée à l'aide de deux hiérarchies de décomposition fonctionnelle:

- la hiérarchie matérielle,
  - la hiérarchie logicielle,
- reliées par une relation d'implantation.

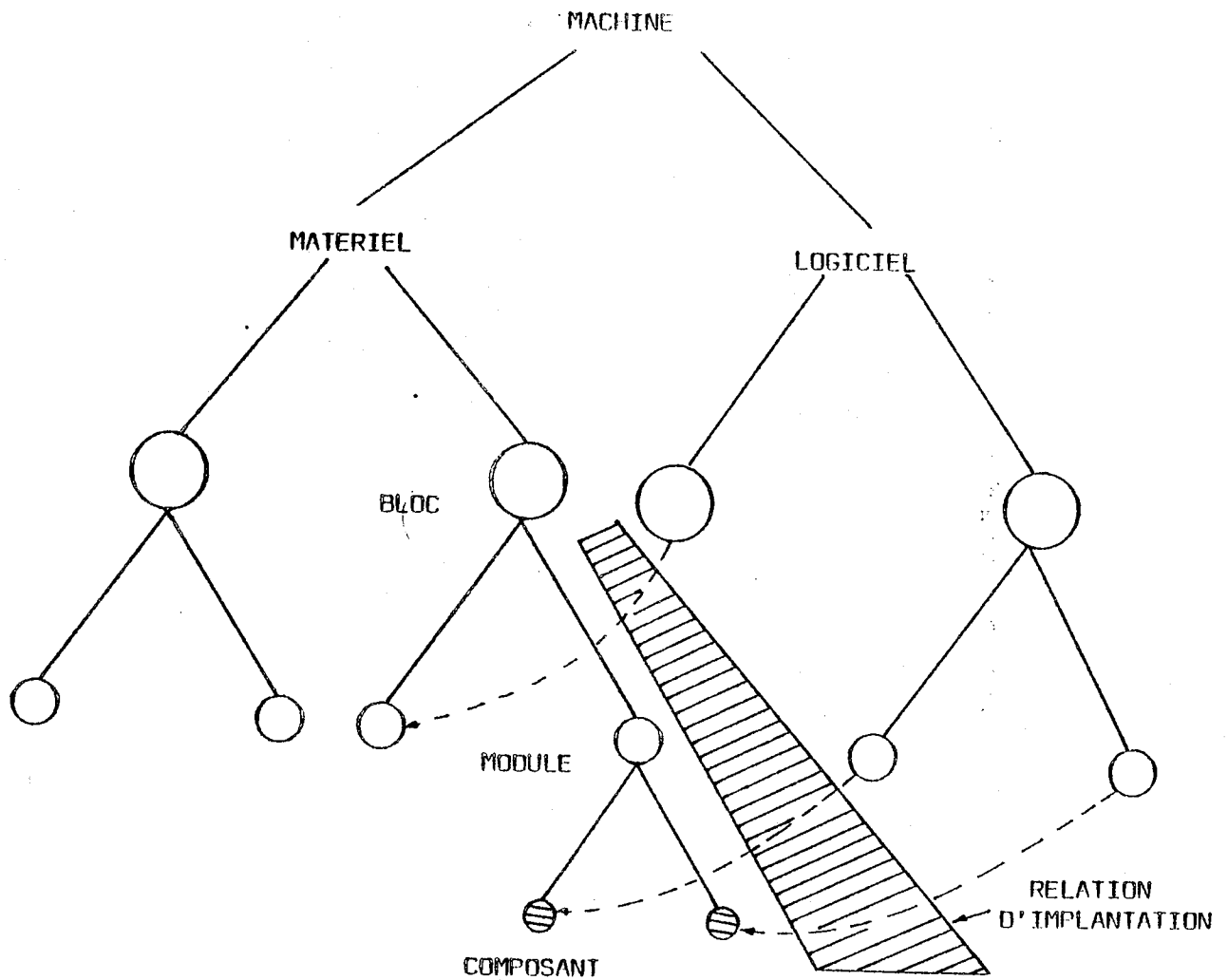


Figure 3.1 Représentation de la machine.

### 3.1.1) Les niveaux de décomposition du matériel.

bloc: interconnection d'un ou de plusieurs modules,

module: l'ensemble de composants,

composants: éléments de base de la machine (P,S,K,...)

### 3.1.2) Les niveaux de décomposition du logiciel.

L.utilisateur: langage de dialogue de l'utilisateur avec la machine.

L.spécialisé: constitue le logiciel "spécialisé" du matériel propre à une application donnée.

L.base: constitue un logiciel de base pour le contrôle d'un matériel dit de base et le contrôle du logiciel spécialisé.

L.machine: constitue le jeu d'instructions du microprocesseur.

A partir du L.spécialisé, nous aurons des logiciels plus ou moins complexes qui dépendront de la machine et de sa généralité.

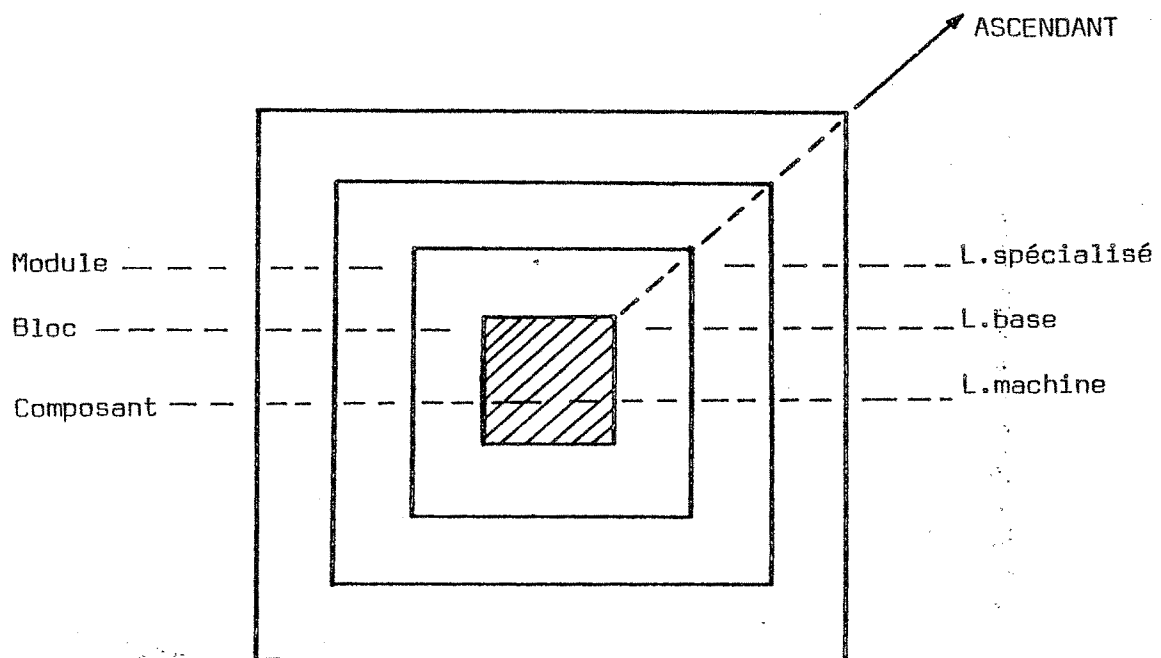


Figure 3.2 Evolution du logiciel et du matériel d'une machine.

### 3.1.3) La relation d'implantation.

Cette relation nous permet de définir l'implantation physique du logiciel dans le matériel. En effet:

- un matériel peut animer ou contenir plusieurs modules logiciels,
- un module logiciel peut être implanté dans plusieurs modules matériels (en cohabitant éventuellement avec d'autres modules logiciels).

#### 4.) LA METHODE: MICROD.

MICROD est une méthode de description de la décomposition fonctionnelle des différents éléments d'une machine.

La description d'une machine grâce à MICROD, est faite sous la forme de différents niveaux imbriqués. La structure de ces niveaux est décrite en termes de composants matériels et d'interconnexions.

Ces définitions peuvent être récursives: un élément peut être défini à l'aide d'autres éléments de niveaux inférieurs.

MICROD, nous permettra de montrer les différents types de machines et leur conception.

##### 4.1) Découpage fonctionnel.

L'expérience prouve qu'il est possible de définir une machine de base qui soit commune à une très grande variété de systèmes (BHR-75)(ORG-72)(MTV-78).

MICROD présente en outre l'avantage de permettre le regroupement dans une machine de base des fonctions communes à différents types de service.

##### Les différents types de machines.

Le micro-système est la machine de base, il est composé d'un ensemble de blocs reliés entre eux par une voie de communication interne (BUS interne).



Le macro-système est défini comme l'interconnexion de deux ou de plusieurs micro-systèmes.

Le système complexe, est défini comme l'interconnexion de deux ou de plusieurs macro-systèmes au moyen de liaisons externes: une liaison télé-informatique par exemple.

#### 4.2) Description informelle de MICROD.

Nous utilisons d'une manière informelle le méta-langage de BACKUS (Backus Normal Form) pour décrire ces structures imbriquées.

L'utilisation de ce type de description peut servir à mieux préciser la structure de la machine pour sa conception ultérieure.

#### Les règles syntaxiques.

Dans ce qui suit, nous présentons les règles syntaxiques de MICROD. Cette méthode utilise les symboles suivants:

<X> signifie "l'élément nommé X"  
 ::= signifie " est représenté par"  
 / signifie " choix d'une alternative"  
 "ε" signifie élément inexistant  
 <X>\* signifie répétition de l'élément "X"  
 (..) signifie élément terminal  
 CO signifie commentaires  
 CHOIX signifie "sélection d'un élément"  
 PARMi signifie "entre un ensemble d'éléments"

#### 4.3) La description MICROD d'une machine.

La syntaxe d'une description de machine est la suivante:

$\langle \text{machine} \rangle ::= \langle \text{D.matériel} \rangle \left[ \leftarrow \right] \langle \text{D.logiciel} \rangle$

où

$\langle \text{D.matériel} \rangle$ : représente la description fonctionnelle du matériel.

$\langle \text{D.logiciel} \rangle$ : représente la description fonctionnelle du logiciel.

$\left[ \leftarrow \right]$ : est un symbole qui indique l'existence d'une relation d'implantation.

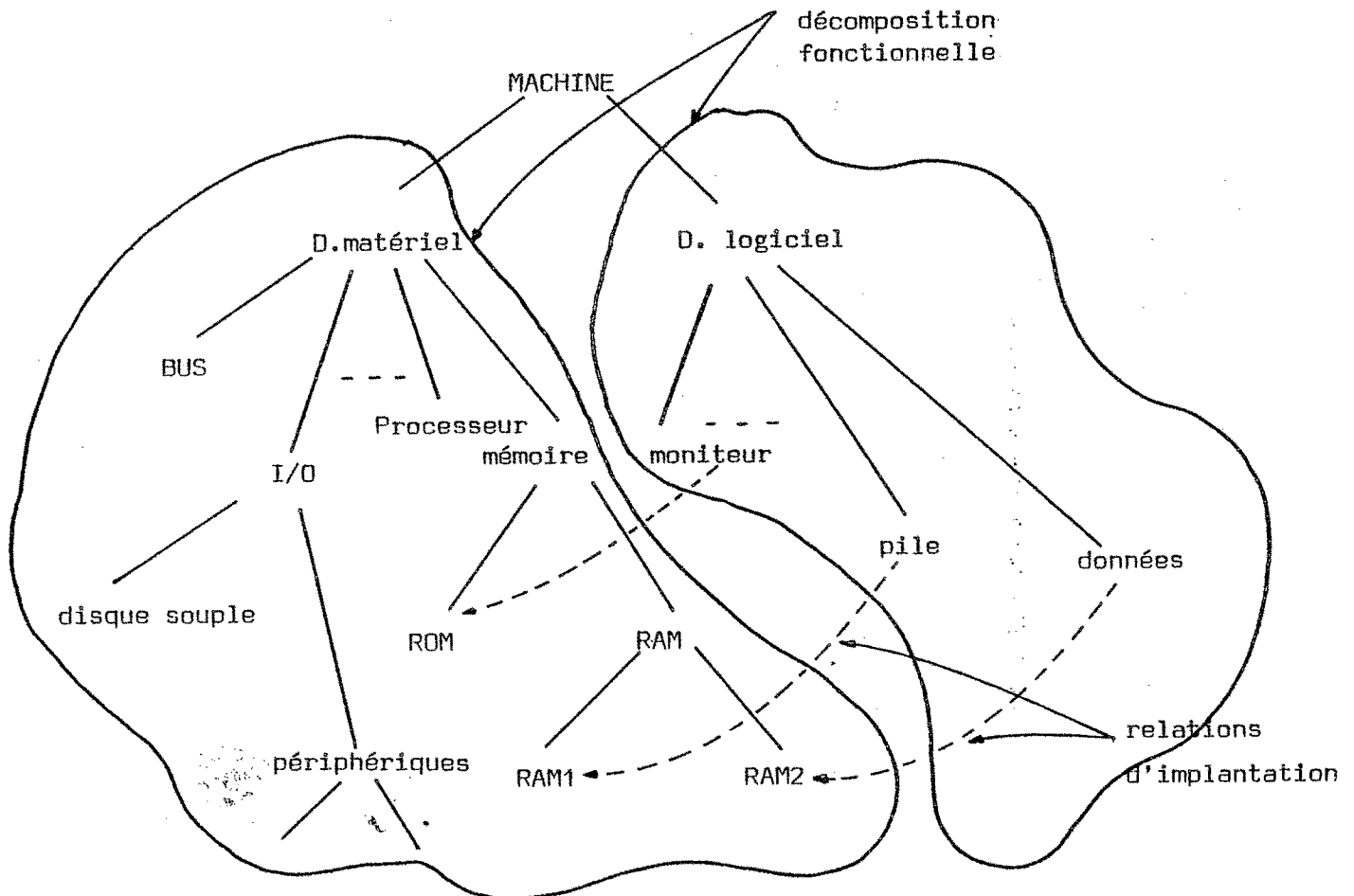


Figure 3.3 Description d'une machine en MICROD.

## 5.) LA DESCRIPTION DES MACHINES.

MICROD est une forme de description structurée de machines à base de microprocesseurs.

La structuration de la conception d'une machine découle de la méthodologie elle-même (c.f.I); du fait que la conception s'arrête au niveau:

- d'un micro-système,
- d'un macro-système,
- ou
- d'un système complexe (plus évolué).

### 5.1) La description d'un micro-système.

Le micro-système représente l'organisation de base d'une machine: le matériel et le logiciel de base.

L'organisation est faite en considérant les points suivants:

- facilité d'implantation des composants,
- facilité de mise au point,
- facilité de mise en service.

Le matériel de base se décompose en blocs où chacun à une tâche particulière à traiter.

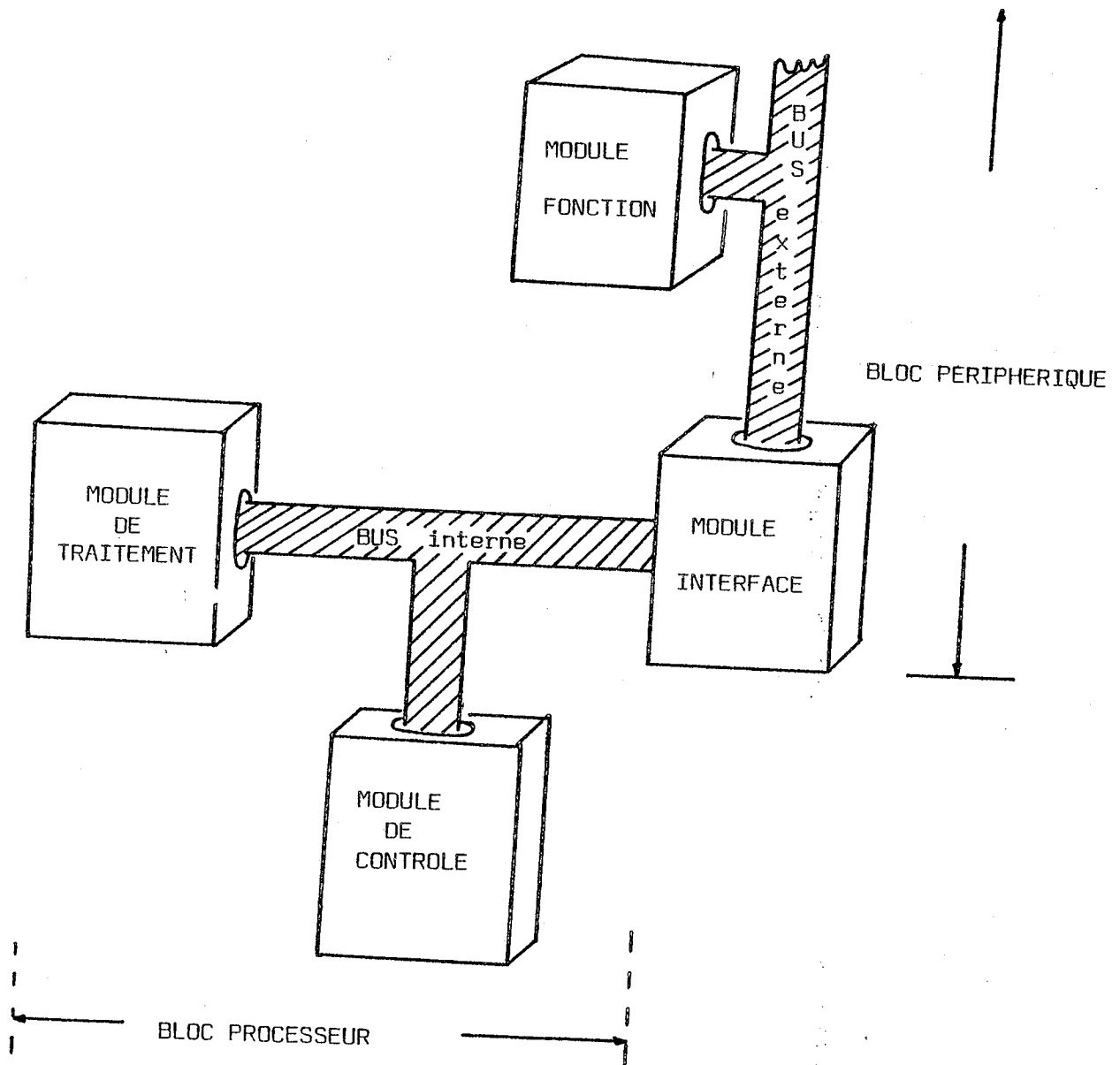


Figure 3.4 Les différents modules matériels d'un micro-système.

La syntaxe de sa description est la suivante:

```

<micro-système> ::= <matériel-base> [↔] <L.micro>
<matériel-base> ::= <bloc processeur><bloc périphérique>
  
```

### 5.1.1) Le bloc processeur.

Ce bloc est composé des modules suivants:

- le module de traitement,
  - le module de contrôle,
- et
- d'une voie de communication entre ces modules appelée BUS interne.

On peut définir le bloc processeur comme:

```

<bloc processeur> ::= <BUS interne><modules>;
<modules> ::= <module de traitement><module de contrôle>;
<BUS interne> ::= (lignes d'interconnexion entre
                  modules);
  
```

#### 5.1.1.1) Module de traitement.

Ce module est défini comme le coeur de la machine; c'est-à-dire qu'il contient les éléments indispensables pour le bon fonctionnement du système.

Pour cela, il contient:

- le processeur lui-même; exemple: MC6800, Z80, ..
  - l'horlogerie centrale,
- ainsi que des fonctions additionnelles telles que:
- les mécanismes d'accès direct à la mémoire,
  - l'horloge programmable.

Ces fonctions additionnelles dépendent de la complexité de la machine. Le module de traitement contient également toute la circuiterie nécessaire pour établir les échanges élémentaires avec le module contrôle et avec le bloc périphérique. Dans MICROD, il s'écrit:

```

<module de traitement> ::= <proc><fonc.syst>*
<proc> ::= <processeur><horlogerie centrale>
<processeur> ::= CHOIX <microprocesseur> PARMY <MC6800>
                <Z80>;<8080>....
<fonc.syst> ::= <mécanismes d'interruption>/
                <mécanismes d'accès direct à la mémoire>/
                <horloge programmable>/ ....

<horlogerie centrale> ::= <horloge centrale>
                          <demandes>*
                          <générateur de pages>

<demandes> ::= <demande de démarrage>/
               <demande d'allongement des phases>* /
               <demande d'interruption du processeur>

<demande de démarrage> ::= (matériel pour activer le
                           système) CO RESET

<demande d'allongement des phases> ::= <cas des mémoires
                                       lentes>/
<cas du rafraîchissement des mémoires dynamiques>/
<cas d'accès direct à la mémoire>
<demande d'interruption du processeur> ::= (matériel pour
la synchronisation des demandes provenant du mécanisme
d'interruption)

```







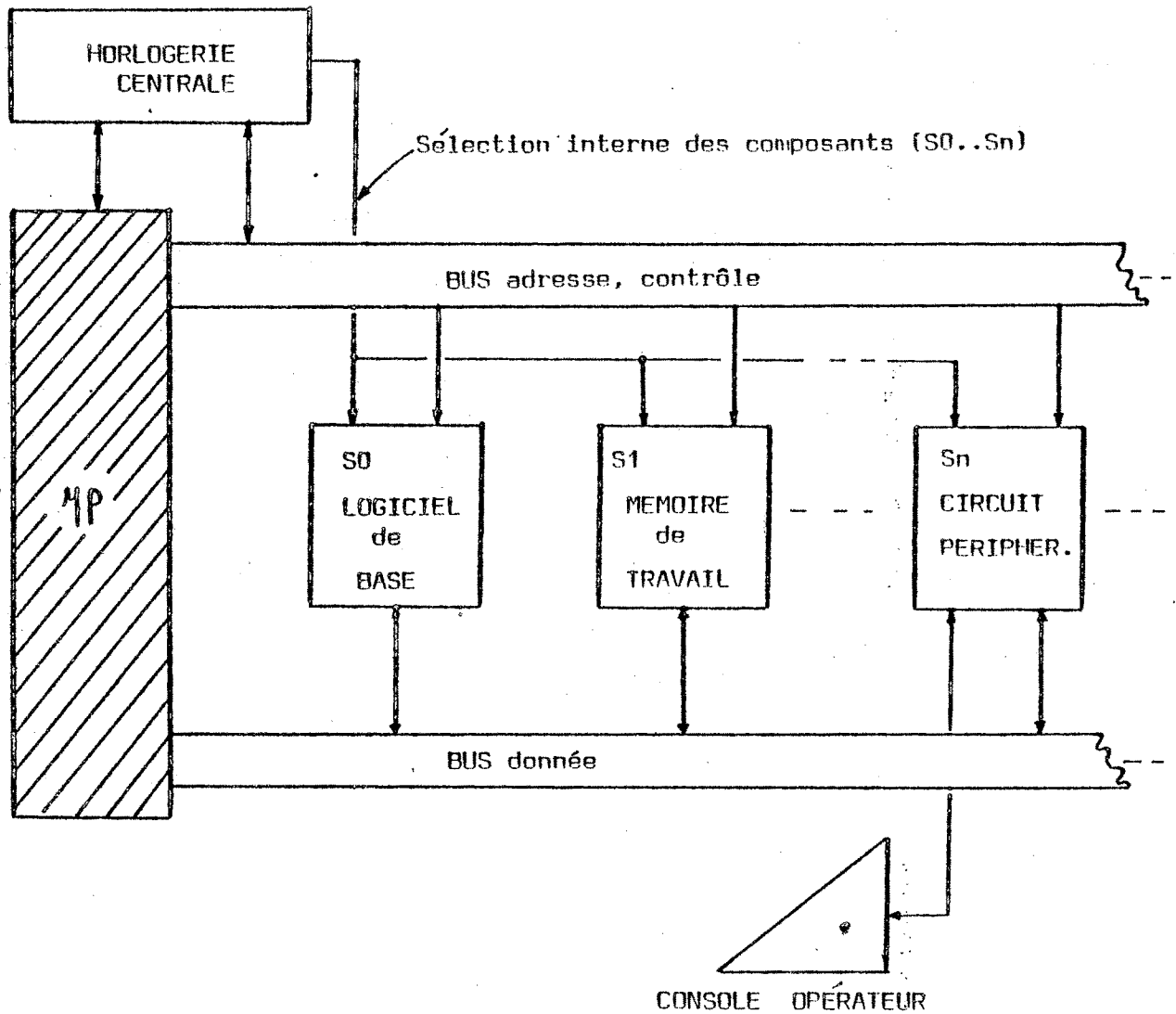


Figure 3.5 Organisation interne du bloc processeur.

Le nombre de composants du bloc processeur est fonction de la charge du BUS (adresses, données, contrôle) et de la complexité du bloc processeur.

### 5.1.1.3) Le BUS interne.

Le BUS interne représente un ensemble de lignes véhiculant des informations de type adresses ,contrôle et données.

Ces lignes peuvent être considérées comme de voies de communication entre les différents éléments du module de contrôle(figure 3.5).

```
<BUS interne>::= <lignes d'adresses>
                <lignes de données>
                <lignes de contrôle>.
```

```
<lignes d'adresses>::= (ensemble de lignes d'adresses);
CO A0,A1,A2,..
```

```
<lignes de données>::=(ensemble des lignes de données)
CO D0,D1,..
```

```
<lignes de contrôle>::= (ensemble des lignes de
                           contrôle) CO R/W, VMA,..
```

### 5.1.2) Le bloc périphérique.

Ce bloc dépend de l'usage qui est fait de la machine. Il est composé d'un ou de plusieurs modules et d'une voie de communication entre ces modules dite BUS externe.

L'extension de la machine est directement fonction des différents modules qui constituent le bloc périphérique: c'est-à-dire des fonctions que l'on y ajoute.

Le bloc périphérique est défini de la manière suivante:

<bloc périphérique> ::=  $\Sigma$  / <bloc périphérique>;  
<bloc périphérique> ::= <module interface><BUS externe>  
 <module fonction>.

#### 5.1.2.1) Le module interface.

Ce module est destiné à:

- contrôler le flux d'informations du bloc externe vers le bloc processeur relatif aux demandes suivantes:
  - \* arrêt du processeur: HALT, ISC, WAIT, ..
  - \* transfert des informations vers le bloc périphérique,
  - \* utilisation de la voie de communication externe.
- amplifier les différents signaux du BUS interne: adresses, contrôle et données.

### 5.1.2.2) Le BUS externe.

Le BUS externe est toujours l'objet d'étude soit pour sa standardisation soit pour mieux développer son comportement dans le système.

Ce BUS est présenté, aux différents modules, comme une ressource à partager car c'est lui qui véhicule les informations de dialogue et de contrôle.

Le contrôle d'une telle ressource se fera par l'unité centrale ou le DMA en suivant une politique:

- par rapport à un ordre de priorité pré-établi,
- par rapport à l'ordre d'arrivée des demandes,

Le BUS externe se représente de la manière suivante:

<BUS externe> ::= <BUS simple>/<BUS complexe>;  
<BUS simple> ::= <BUS normal>/<BUS normal><BUS sécurité>;  
<BUS complexe> ::= <BUS simple>\*;  
<BUS normal> ::= (ensemble de lignes standardisées, qui véhiculent des informations entre modules et lignes d'alimentation);  
<BUS sécurité> ::= (ensemble de lignes qui véhiculent des signaux propres à la sécurité du système); CO violation mémoire, commutation de processeurs par exemple;

### 5.1.2.3) Le module fonction.

Il représente les fonctions matérielles, propres à une application donnée de la machine: coupleurs,...etc.

Il se décrit de la manière suivante:

<module fonction> ::= <mémoires><sup>\*</sup> <matériel spécialisé>;

<mémoires> ::= <mémoires d'accès aléatoire>/  
 <mémoire en lecture seulement>/  
 <mémoire de masse>

<mémoire de masse> ::= <contrôleur><unité.masse><sup>\*</sup>

<unité.masse> ::= <disque rigide> /<disque souple>/  
 <cassettes>/<bulles>/<CCD>/...

<contrôleur> ::= (matériel pour la supervision de  
 l'unité mémoire de masse);

<matériel spécialisé> ::= (matériel spécialisé d'une  
 application donnée)

CO entrées et sorties additionnelles; coupleurs,..

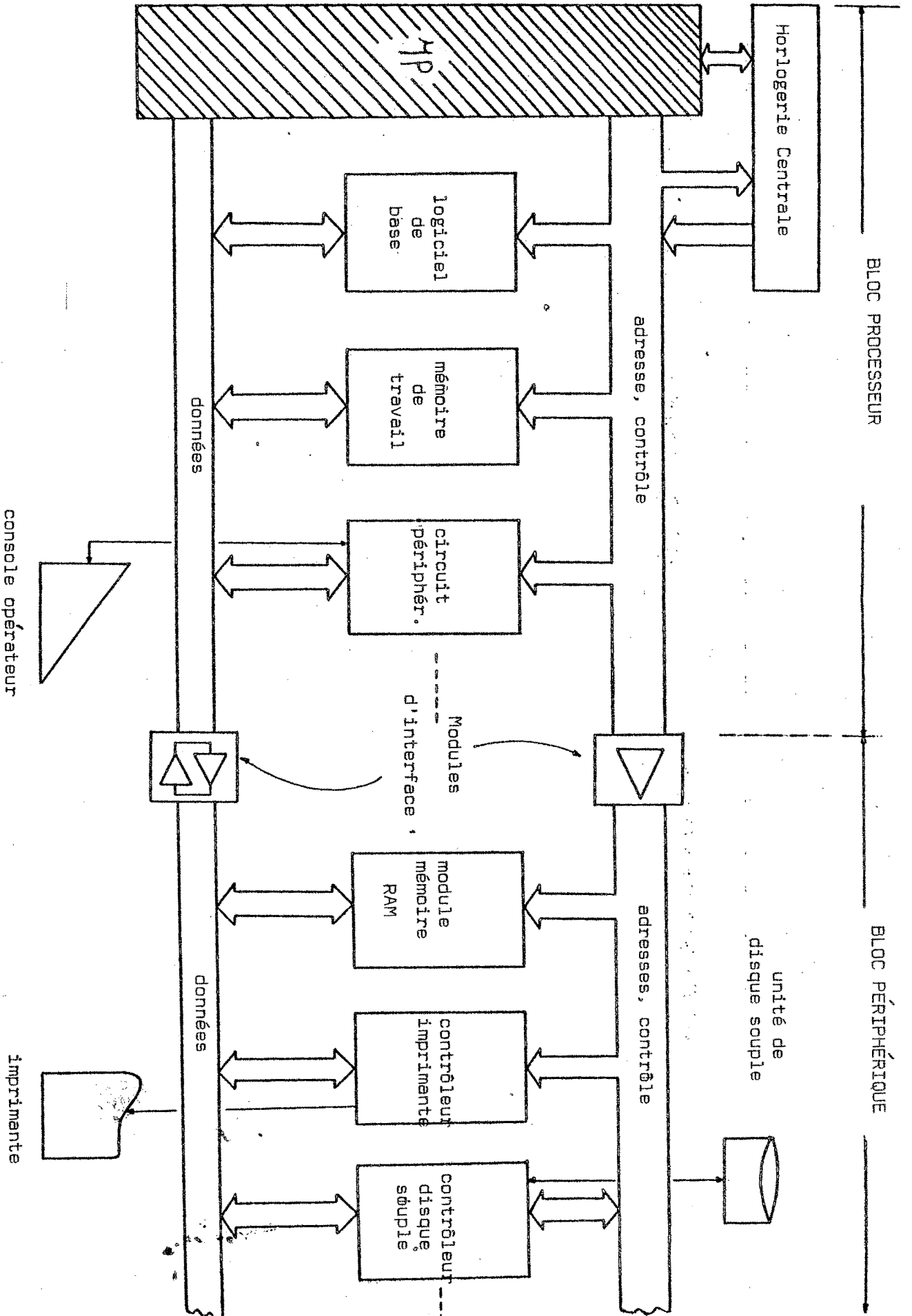


Figure 3.6 L'organisation de base d'un système de développement: un micro-système.

5.1.3) Le logiciel du micro-système: L.micro.

Le logiciel d'un micro-système est décrit à l'aide de MICROD de la manière suivante:

```

<L.micro> ::= <L.base> <L.spécialisé>
<L.spécialisé> ::= <prog.spécialisé>
<L.base> ::= <prog.moniteur>
<prog.moniteur> ::= <var.travail moniteur>
                    <contexte moniteur>
<contexte moniteur> ::= <mod.service> <corp.moniteur>
<mod.service> ::= (ensemble de modules de service) CO
initialisation des périphériques,..

```

```

<corp.moniteur> ::= <liste des points d'entrées>*
                    <liste des commandes>
<liste des points d'entrées> ::= <p.entrée><code>
<prog.spécialisé> ::= <var.travail.prog> <corp.prog>
<corp.prog> ::= <liste des points d'entrées>*
<code> ::= <L.machine>

```

```

<liste des commandes> ::= (liste des commandes propres à
commander l'environnement de la machine)

```

<p.entrée> ::= (point d'entrée fixé par le système); CO  
par exemple sur des machines qui utilisent le  
microprocesseur MC6800 le point d'entrée est caractérisé par  
son vecteur machine (RESTART,NMI,SWI,IRQ)(c.f.annexe II);

Le point d'entrée du programme, fixé par l'utilisateur,  
est l'adresse de début du programme.

Pour mieux illustrer, l'exposé précédent, nous avons schématisé ci-dessous la description fonctionnelle du moniteur SP-6800(c.f.annexe II) implanté pour des machines utilisant le microprocesseur MC6800.

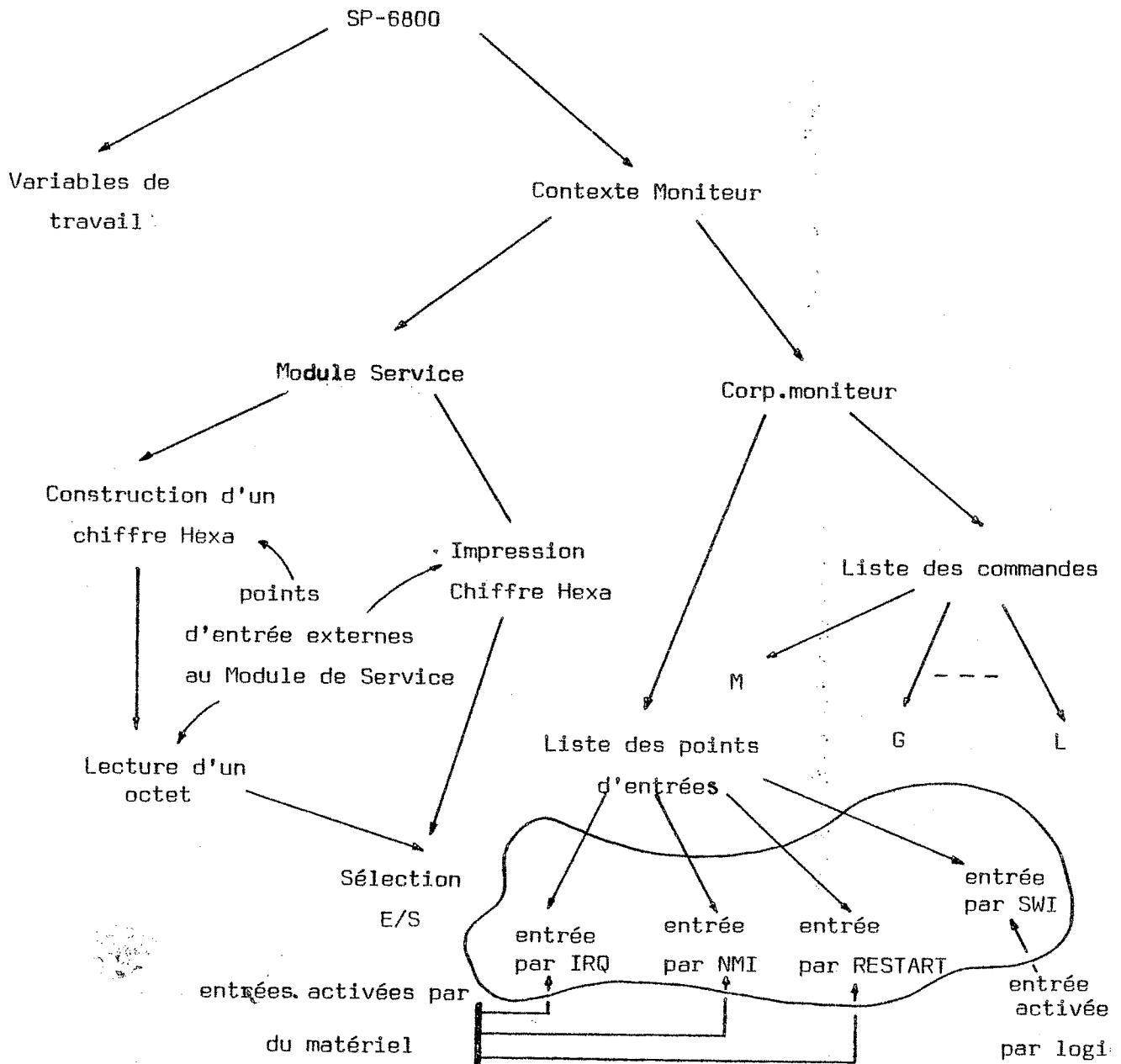


Figure 3.7 La description fonctionnelle du SP-6800.



#### 5.1.4) Les relations d'implantation interne dans un micro-système.

Au niveau moniteur:

<ROM> [ ] <prog.moniteur>  
 <ROM> [ ] <liste des points d'entrée>  
 CO point d'entrée moniteur  
 <RAM> [ ] <var.travail.moniteur>

Au niveau application:

<module mémoire> [ ] <prog.spécialisé>

Ces relations définissent une relation interne d'implantation.

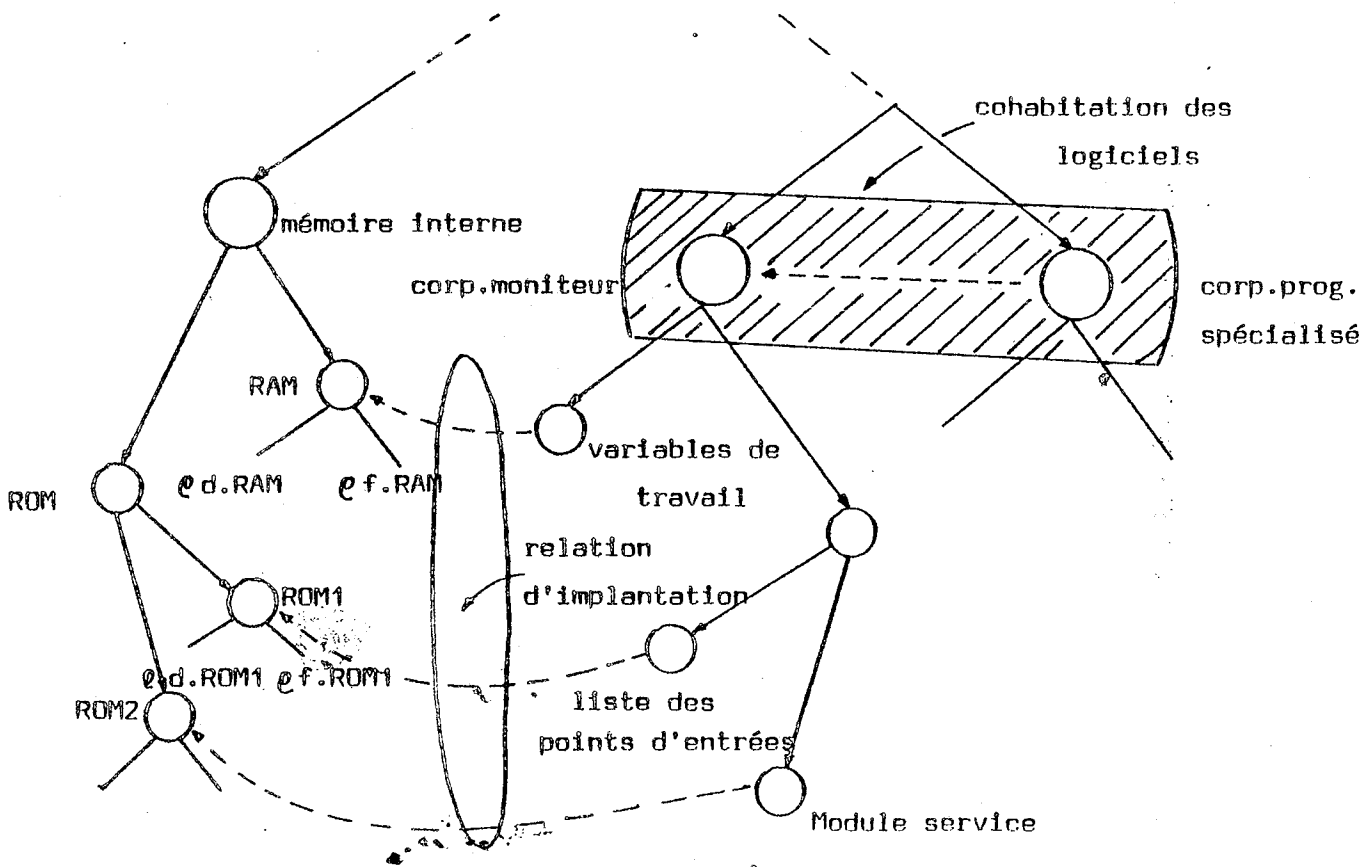


Figure 3.8 Les relations d'implantation dans un micro-système.

## 5.2) LA DESCRIPTION D'UN MACRO-SYSTEME.

Le macro-système est défini comme un ensemble de micro-systèmes reliés entre eux par un même environnement matériel.

Sa syntaxe:

<macro-système> ::= <matériel-macro> [ ] <L.macro>

<matériel-macro> ::= <micro-système> <mult-macro>\*

<élément.comm>

<élément.comm> ::=  $\Sigma$  / <élément comm>

<élément.comm> ::= (élément partageable) C0 par exemple une mémoire commune.

<mult-macro> ::= <liaison> <micro-système>

<liaison> ::= <module interface> <BUS externe>

<L.macro> ::= <L.spécialisé>\*

### Les relations externes d'implantation.

Les éléments propres à un micro-système (BUS, E/S, mémoire externe, ...) peuvent être partagés par un autre micro-système, tout en dépendant des caractéristiques de l'application du macro-système: logiciel placé dans une mémoire commune à deux micro-systèmes par exemple.

Ce partage des éléments ou la cohabitation des logiciels entre micro-systèmes traduit une relation d'implantation externe.

Un macro-système peut être représenté sous la forme de couches, entre lesquelles nous pouvons établir des relations d'implantation.

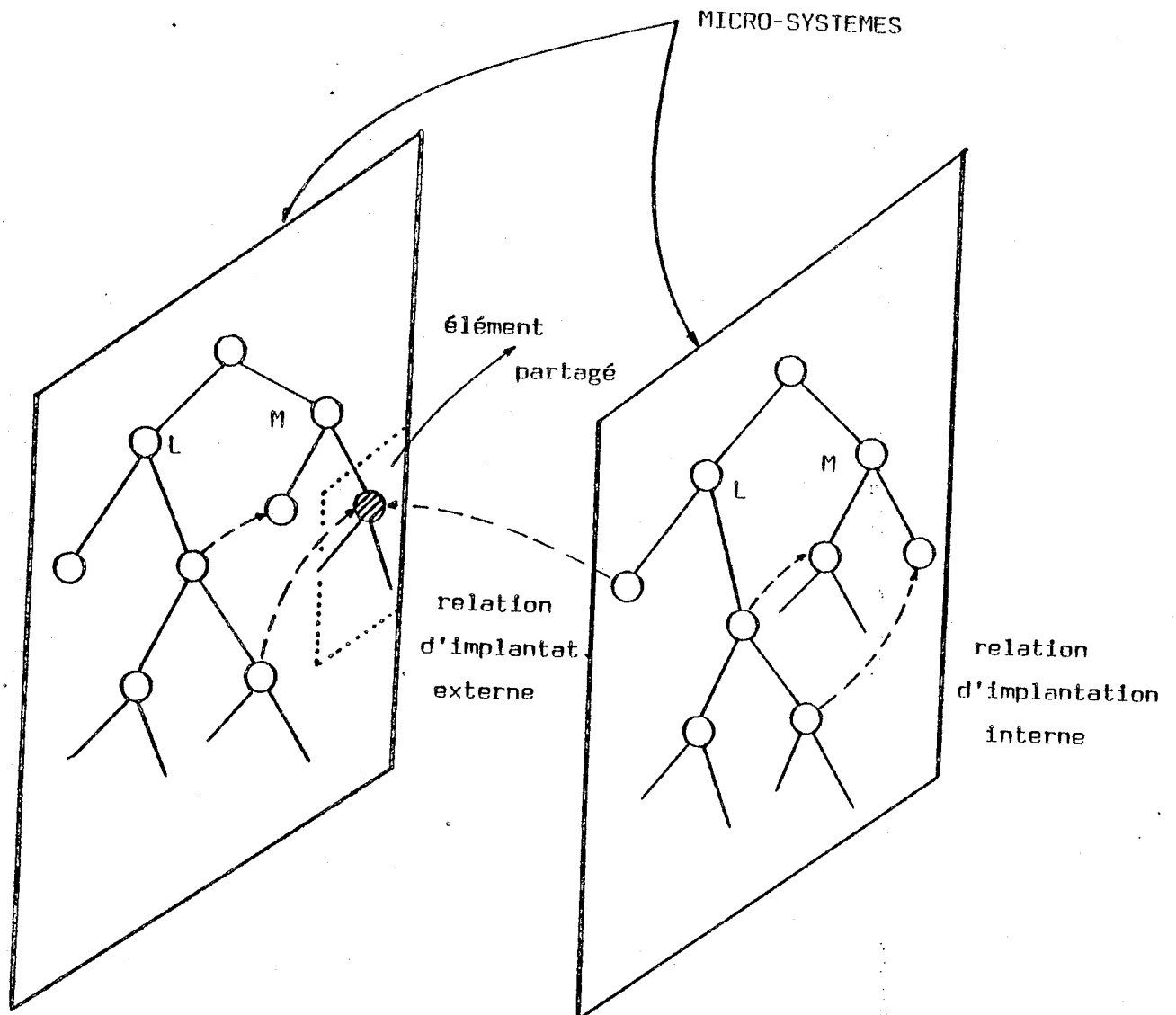


Figure 3.9 Le contexte des relations d'implantation d'un macro-système.

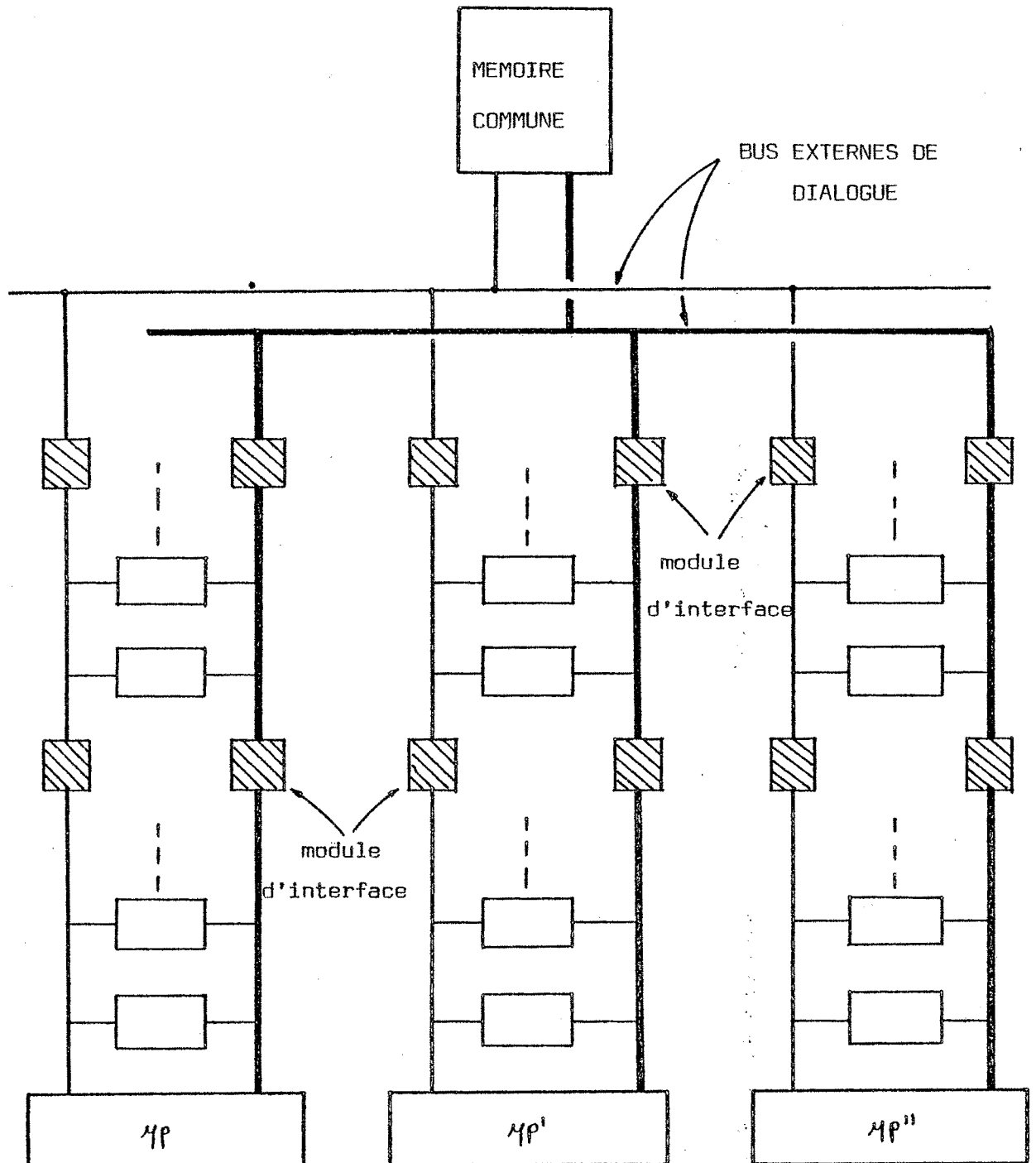


Figure 3.10 Exemple d'un macro-système: un réseau de microprocesseurs partageant une mémoire commune pour leur dialogue.

### 5.3) LA DESCRIPTION D'UN SYSTEME COMPLEXE.

Ce dernier niveau, dans la conception d'une machine est défini de la manière suivante:

$\langle \text{systeme complexe} \rangle ::= \langle \text{matériel-complexe} \rangle \left[ \leftarrow \right] \langle \text{L.évolué} \rangle$   
 $\langle \text{matériel-complexe} \rangle ::= \langle \text{macro-système} \rangle \langle \text{mult-complexe} \rangle^*$   
 $\langle \text{élément.comm} \rangle$  CO par exemple un  
 $\langle \text{élément.comm} \rangle$  peut être un macro-système utilisé comme  
 périphérique (processeur canal).  
 $\langle \text{mult-complexe} \rangle ::= \langle \text{liaison-macro} \rangle \langle \text{macro-système} \rangle$   
 $\langle \text{liaison-macro} \rangle ::= \langle \text{liaison série} \rangle /$   
 $\langle \text{liaison télé-informatique} \rangle.$   
 $\langle \text{L.évolué} \rangle ::= \langle \text{L.macro} \rangle^* ;$

La relation d'implantation du L.évolué dans le matériel-complexe se traduit comme la projection de l'ensemble des relations d'implantation internes ou externes dans chaque macro-système constituant le système complexe.

Pour représenter les notions de macro-système et de système complexe la figure ci-dessous montre, à titre d'exemple, l'organisation de la commande d'un auto-commutateur téléphonique.

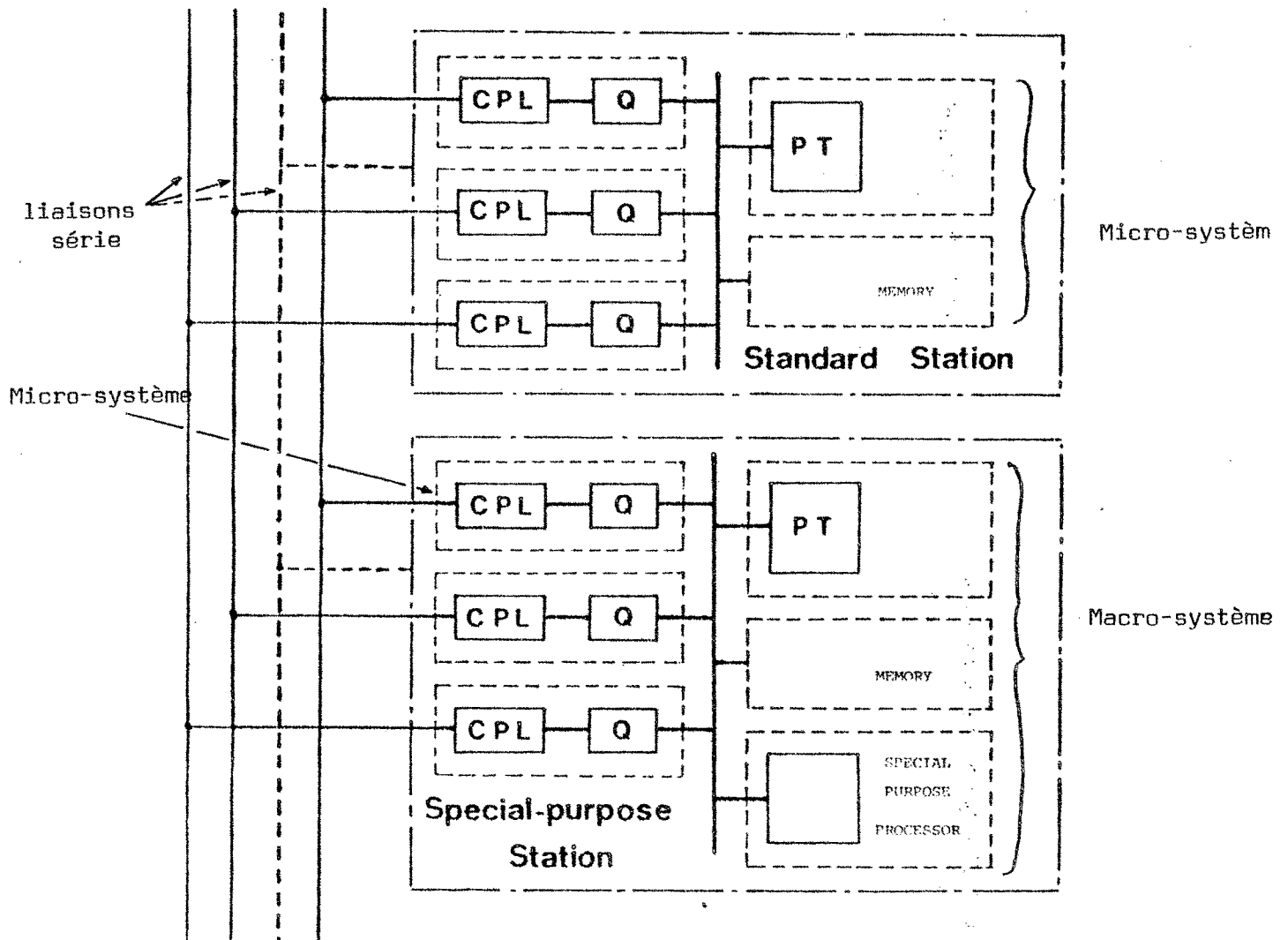
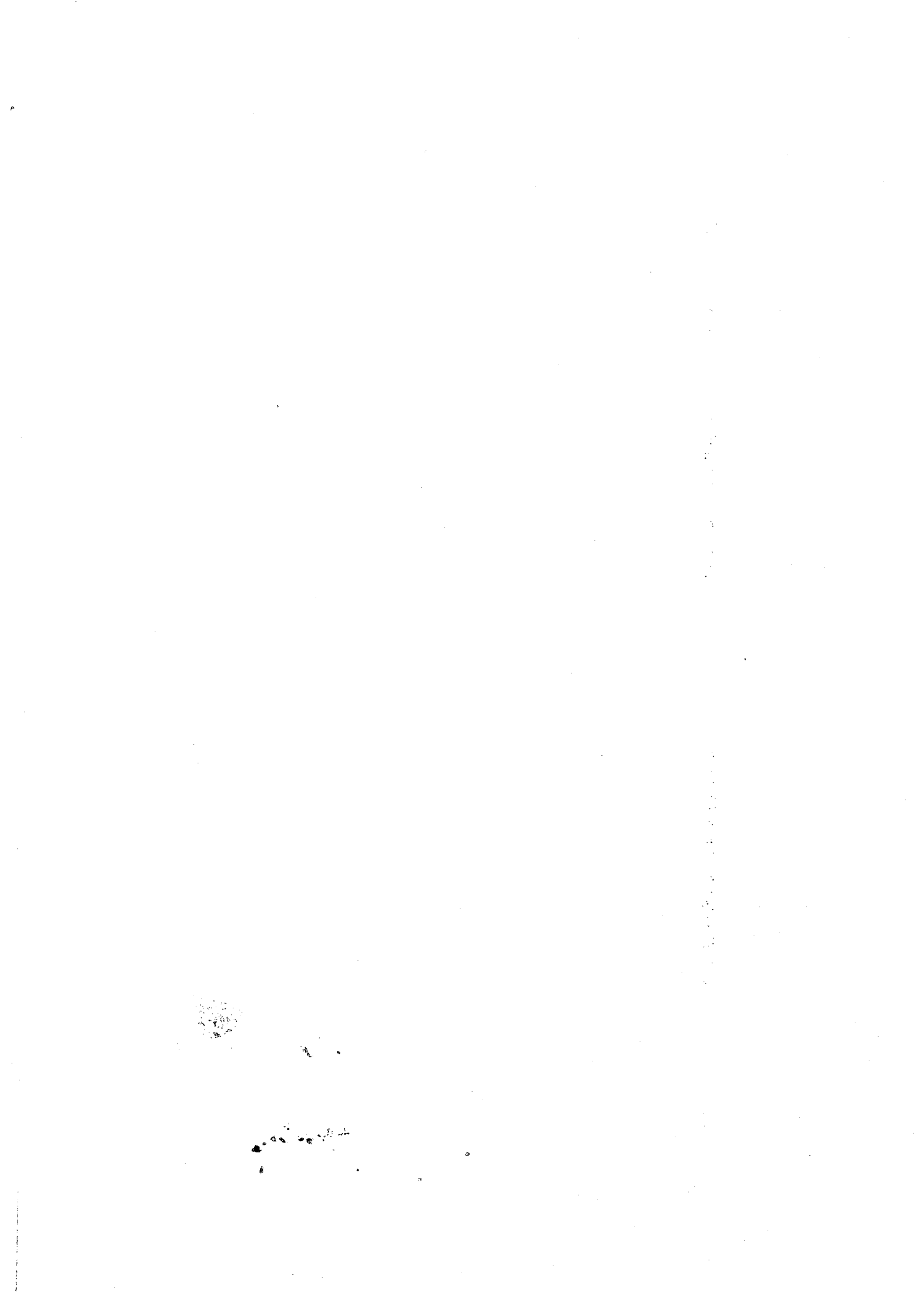


Figure 3.11 Un système complexe: commande répartie d'un autocommutateur téléphonique (CMP-79).



## DEUXIEME PARTIE

### UN PERIPHERIQUE D'AIDE AU DEVELOPPEMENT D'APPLICATIONS MICROPROCESSEURS ET DE SUPERVISION EN TEMPS REEL.

- 1.) INTRODUCTION
- 2.) DEMARCHES DANS LA REALISATION DU PERIPHERIQUE
- 3.) DESCRIPTION FONCTIONNELLE DU PERIPHERIQUE



## DEUXIEME PARTIE

### 1.) INTRODUCTION.

Cette partie décrit l'aspect matériel d'un périphérique d'aide au développement d'applications microprocesseurs et de supervision en temps réel.

Pour mieux analyser la description des différents composants du périphérique, nous utilisons le formalisme exposé dans la première partie.

### 2.) DEMARCHES DANS LA REALISATION DU PERIPHERIQUE.

Nous avons utilisé la méthode de conception par couches(c.f.II) pour réaliser et mettre au point les différents composants constituant le périphérique; nous commençons par la couche noyau (processeur central) et nous bâtissons les autres couches, dites "fonctionnelles" à partir de cette dernière.

#### 2.1) La couche noyau: processeur central.

Le processeur central(P.central) a une architecture classique et est décrit dans l'annexe I. Ce processeur est utilisé comme processeur standard pour plusieurs applications microprocesseurs développées dans le cadre de l'équipe de recherche en architecture de l'IMAG.

## 2.2) Les couches fonctionnelles.

### 2.2.1) La mémoire de masse.

#### Les mémoires à couplage de charges: CCD

L'avance de la technologie dans la conception des mémoires à grosse capacité de stockage, comme par exemple les mémoires à couplage de charges, permet d'augmenter la complexité du périphérique en lui ajoutant un mécanisme de mémoire de masse grâce à ce type de composant.

#### L'unité de disque souple.

L'état volatil des informations contenues dans la mémoire du P.central, et dans les mémoires CCD, nous a amené à incorporer un mécanisme de stockage permanent dans le système sous la forme de la réalisation d'un coupleur de disque souple.

### 2.2.2) Le processeur "espion".

La mise au point et l'analyse en temps réel d'applications microprocesseurs et multiprocesseurs (analyse du dialogue entre processeurs), nous a conduit à envisager une dernière étape dans la réalisation du périphérique: un processeur appliqué au développement logiciel et matériel en temps réel des applications microprocesseurs: ANALYD.

Ce processeur peut être utilisé pour la mise au point de plusieurs types de systèmes:

- soit dans l'environnement du périphérique, pour regarder le comportement de nouvelles fonctions supplémentaires,
- soit connecté à l'extérieur, pour examiner le comportement d'une application s'exécutant sur un autre matériel que celui du périphérique.

3.) LA DESCRIPTION FONCTIONNELLE DU PERIPHERIQUE.

Le périphérique est organisé comme un macro-système(c.f.III). Sa description matérielle et logicielle est présentée dans le graphe montré dans la figure II.1

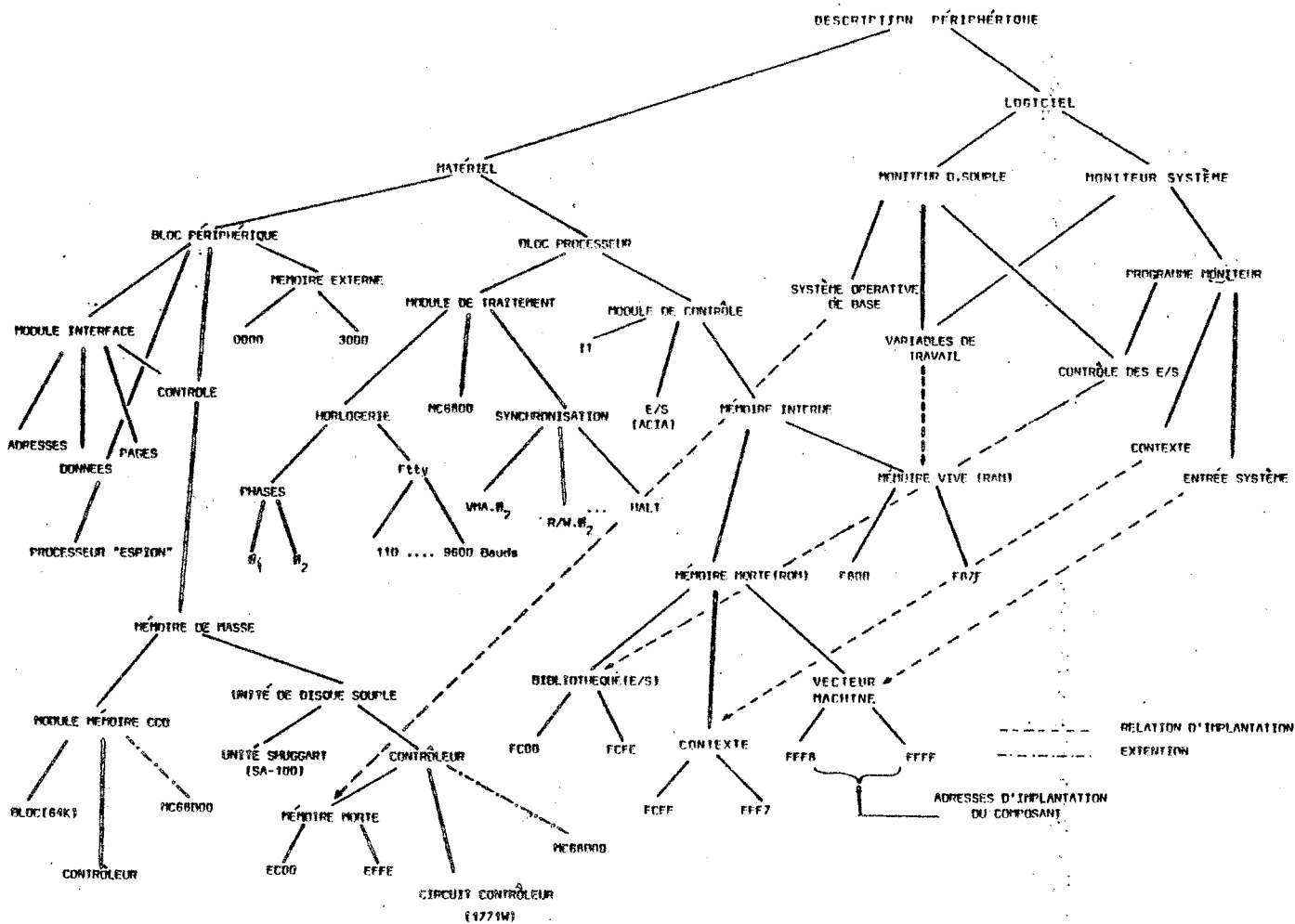


Figure II.1 Description fonctionnelle du périphérique.

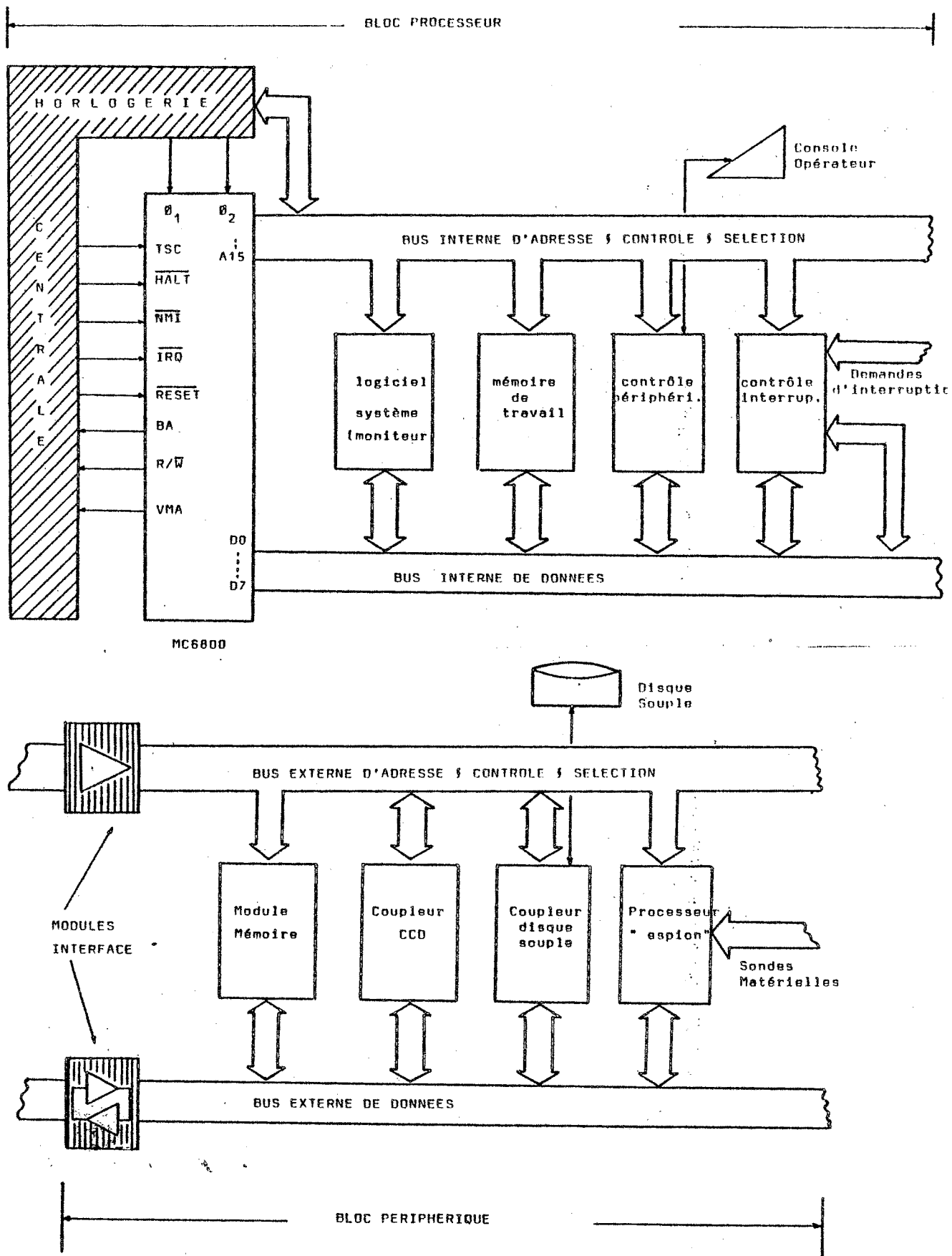
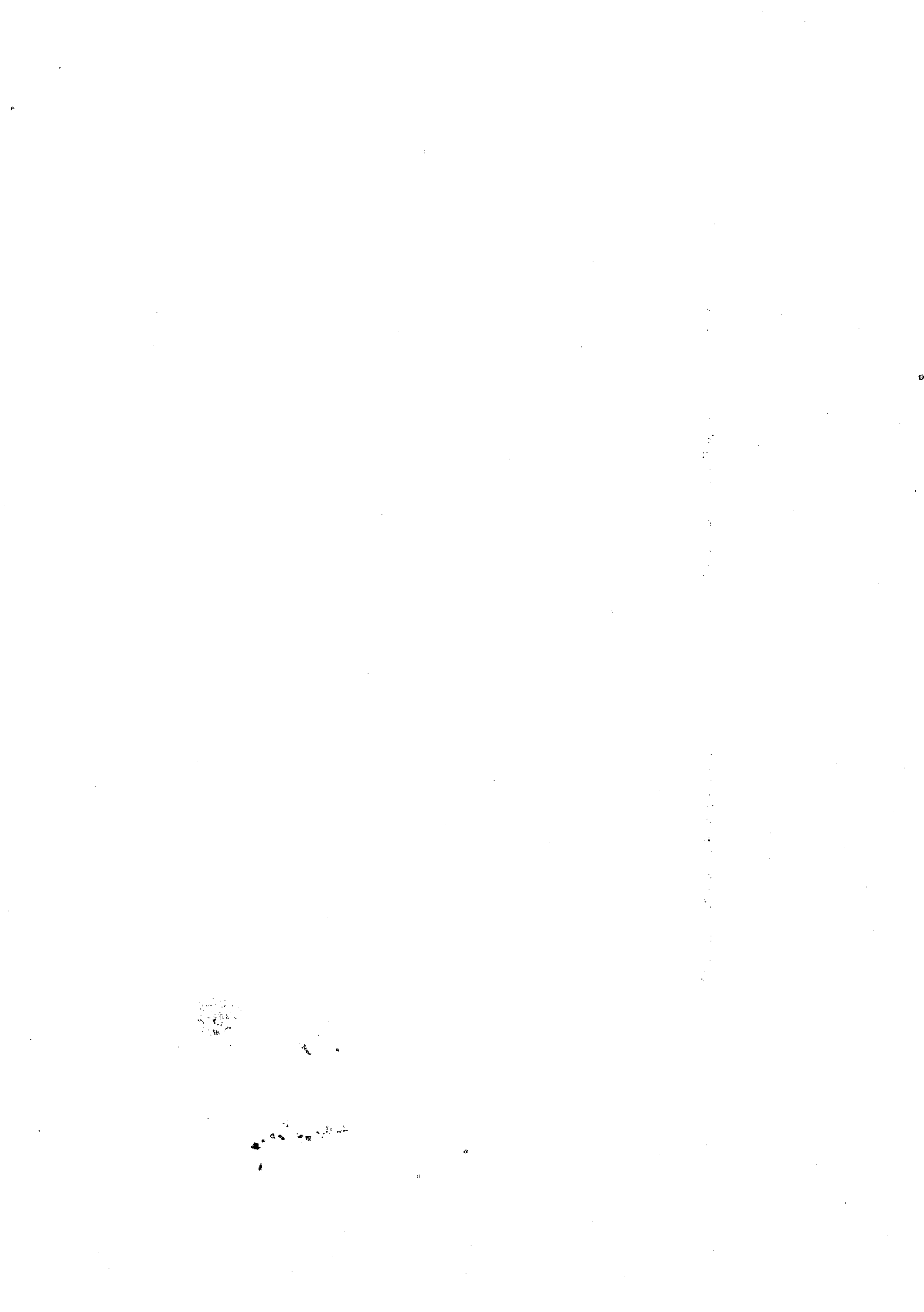


Figure II.2 L'architecture du périphérique.



## CHAPITRE IV

### UN PROCESSEUR DE GESTION DE MEMOIRES CCD.

- 1.) INTRODUCTION
- 2.) LA TECHNOLOGIE DES MEMOIRES à CCD
- 3.) LA MEMOIRE CCD CHOISIE
- 4.) LE PROCESSEUR CCD
- 5.) LA REALISATION: Le contrôleur prototype
- 6.) ROLE DU P.CENTRAL.

## CHAPITRE IV

### UN PROCESSEUR DE GESTION DE MEMOIRES

#### CCD.

##### 1.) INTRODUCTION.

Le développement de mémoires utilisant des technologies nouvelles (mémoires à couplage de charges, mémoires à bulles,..) permet d'envisager des dispositifs à faible temps d'accès et de capacité équivalente à celle des dispositifs mécaniques (disques souples).

Pendant ces dernières années, de nombreux travaux ont été réalisés sur les mémoires CCD (Charge Coupled Device). Il est possible d'envisager l'utilisation de ces dispositifs comme des blocs de base dans la construction de grandes mémoires périphériques (LIP-77).

Il existe plusieurs articles qui décrivent le composant mais il en existe peu qui analysent son utilisation dans un système de mémoires (HOR-76)(CCE-76)(ZIB-77).

Ce chapitre est consacré à la description d'une telle mémoire, gérée par un microprocesseur, elle peut être utilisée dans les dispositifs travaillant en temps réel nécessitant une mémoire de masse rapide.

## 2.) LA TECHNOLOGIE DES MEMOIRES à CCD.

La mémoire CCD, développée dans les laboratoires de la BELL TELEPHONE COMPANY, se présente comme un registre à décalage. Celui-ci est construit à partir d'une série de capacités (Metal Oxide Semiconducteur) dont la charge se propage d'un élément à l'autre sous l'effet de signaux d'horloge.

### 2.1) Rappel sur la cellule CCD.

Dans le transistor MOS, la grille de contrôle (GATE) est isolée du SUBSTRAT formant une capacité de grille.

Une tension appliquée à la grille provoque la formation d'une cellule de stockage (Q) qui peut contenir ou non une charge d'espace dont la valeur représente l'information courante (figure 4.1).

En appliquant des tensions adéquates ( $V2 > V1$ ) sur les grilles; la charge contenue dans une cellule de stockage peut être transférée partiellement, ou totalement, à la cellule de stockage adjacente.

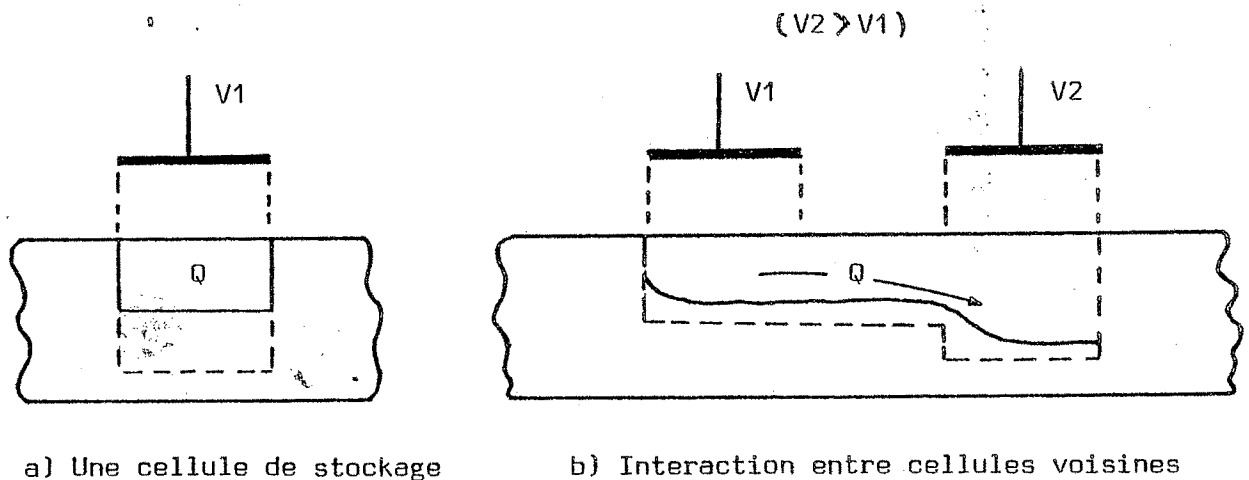


Figure 4.1 La cellule CCD (ZIB-77).



Cette propriété constitue la caractéristique des CCD qui permet de les utiliser pour la réalisation de registres à décalage. Nous pouvons décrire une étape de décalage utilisant quatre phases d'horloge.

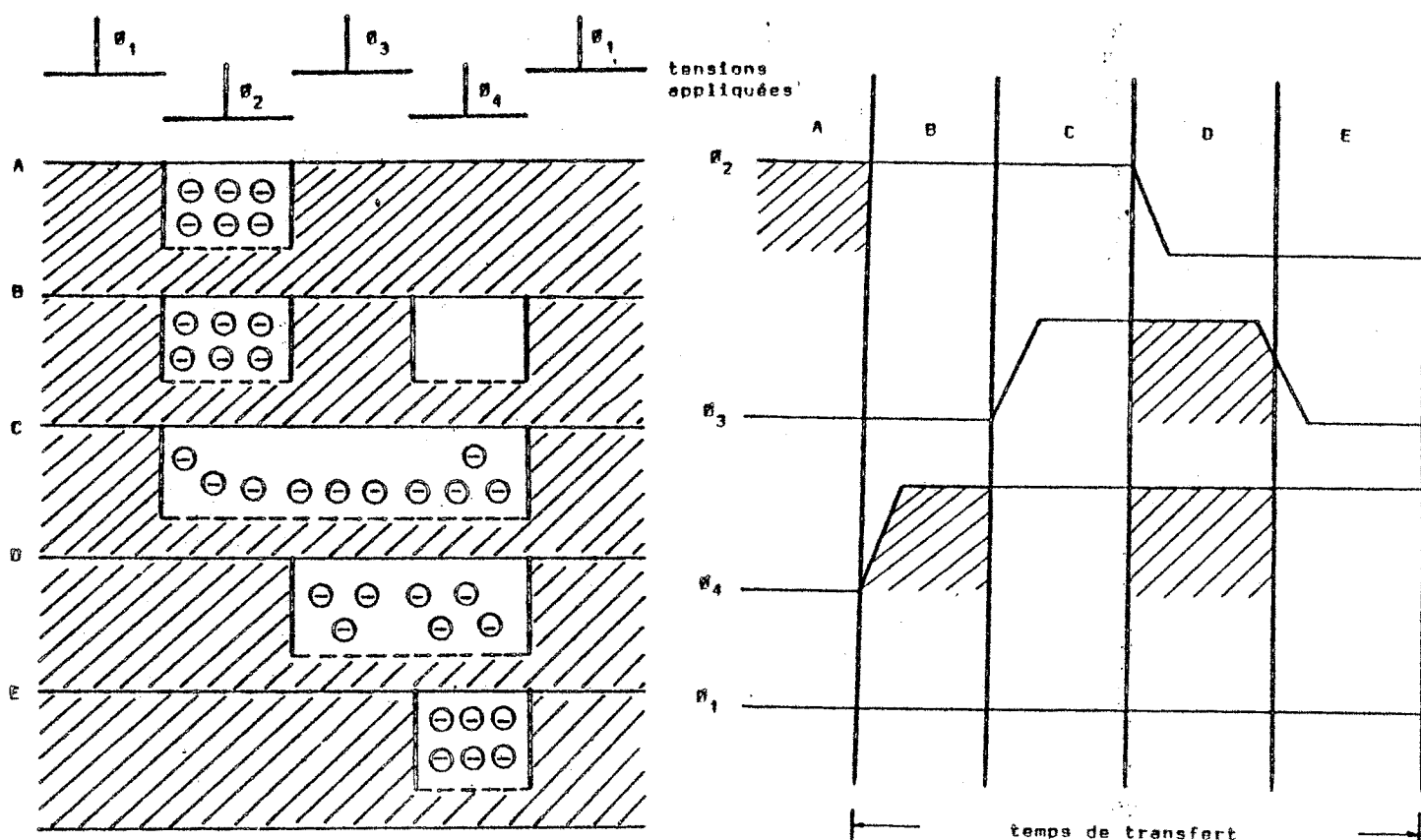


Figure 4.2 Le transfert de charges utilisant quatre phases d'horloge.

La description du mode de fonctionnement (figure 4.2) est simple, supposons, dans un premier temps, que la cellule de stockage créée par la tension positive ( $\varnothing 2$ ) soit chargée (point A), puis dans un second temps, instant (B), les tensions sur les grilles  $\varnothing 2$  et  $\varnothing 4$  sont positives; sous la grille  $\varnothing 4$  se forme une autre cellule de stockage qui ne contient pas de charge.

A l'instant (C), la tension sur la grille  $\varnothing 3$  forme une cellule de stockage qui recouvre les autres cellules formées sous les grilles  $\varnothing 2$  et  $\varnothing 4$  à l'instant (B). Il existe une cellule de stockage continue qui comprend la cellule formée sous la grille  $\varnothing 2$ , celle formée sous la grille  $\varnothing 3$  et celle formée sous la grille  $\varnothing 4$ .

La charge stockée dans la cellule sous la grille  $\varnothing 2$  est répartie jusqu'à la cellule sous la grille  $\varnothing 4$ ; cette répartition des charges dépend des tensions positives appliquées au point C. A l'instant (D), la tension appliquée à la grille  $\varnothing 2$  devient négative et élimine la cellule de stockage, précédemment formée, poussant les charges vers les cellules adjacentes.

A l'instant (E), le transfert est complet car la tension sur la grille  $\varnothing 3$  devient négative et pousse les charges vers la cellule formée sous la grille  $\varnothing 4$ . Le cycle peut alors se répéter, sous réserve de permuter les rôles des grilles  $\varnothing 4$  et  $\varnothing 2$ ,  $\varnothing 1$  et  $\varnothing 3$ .

Le temps de décalage (SHIFT TIME) est la durée du transfert complet entre les grilles  $\varnothing 2$  et  $\varnothing 4$  soit 4 phases d'horloge.

## 2.2) Organisation interne des mémoires CCD.

Plusieurs organisations ont été proposées et développées pour construire des mémoires en utilisant la technologie CCD. Le choix de l'organisation dépend de la performance désirée du système; c'est-à-dire de son temps d'accès, du nombre de phases, de la consommation et de la fréquence de l'horloge de décalage. Le choix de cette fréquence est critique car celle-ci est limitée par:

- la vitesse maximale de transfert de l'information entre cellules: charge et décharge.
- le temps de rafraichissement nécessaire pour sauvegarder l'information: temps de régénération.

En utilisant ces critères de choix, les CCD peuvent être utilisées comme:

- registres à décalages normaux,
- mémoires à stockage temporel.

Nous nous intéressons au deuxième cas.

### 2.2.1) Organisation en série:"serpentine".

Cette organisation est constituée d'un ensemble de registres à décalages connectés en série formant une boucle unique qui constitue le support de mémorisation (figure 4.3).

Des tampons de lecture et d'écriture sont placés à chaque extrémité de la boucle.

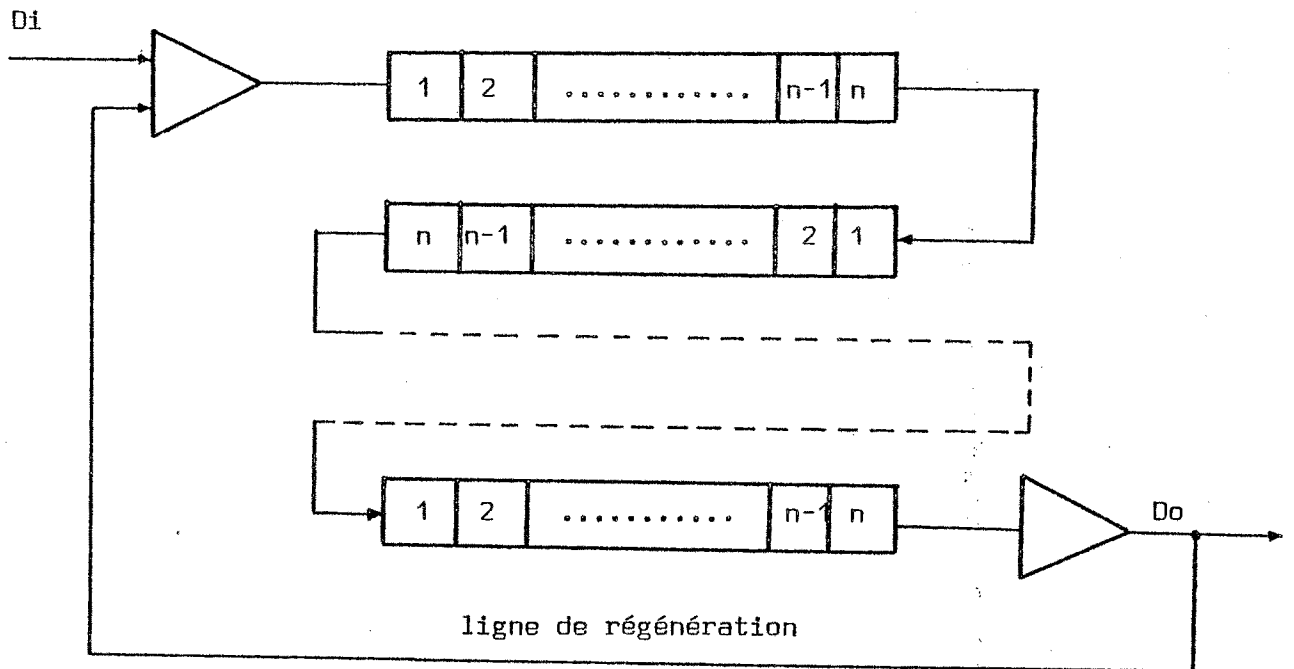


Figure 4.3 Organisation série.

La vitesse de transfert et le nombre de bits entre le point d'entrée ( $D_i$ ) et le point de sortie ( $D_o$ ) déterminent le temps maximum d'accès à l'information ( $t_m$ ) (figure 4.4).

La consommation, pour cette organisation, est proportionnelle à la fréquence d'accès ( $K_{fa}$ ) car tous les bits de l'information sont transférés à la même fréquence (AME-75)(CCE-76)(RT5-75).

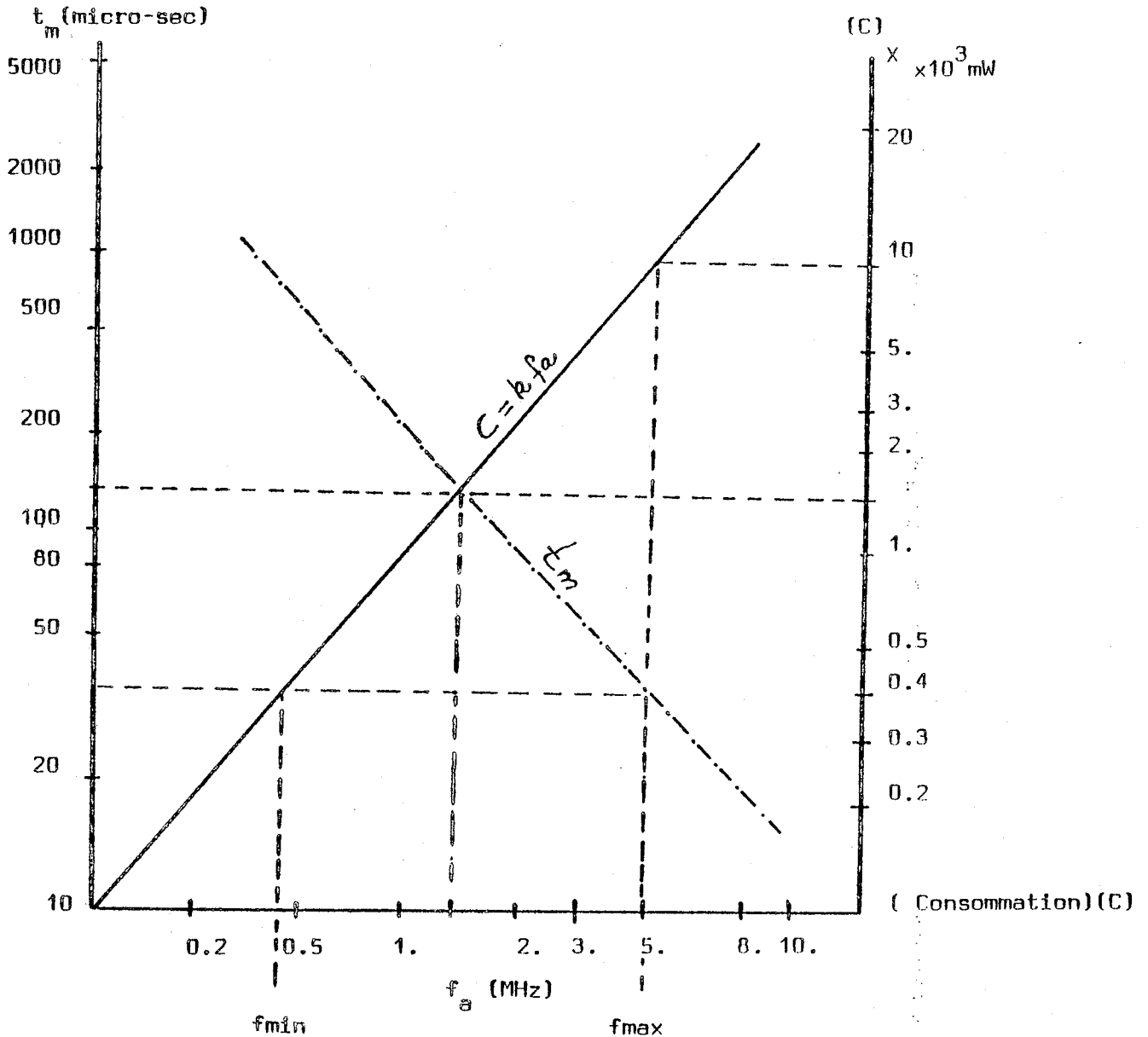


Figure 4.4 Relation entre le  $t_m$  et la consommation  $(C)$  par rapport à la fréquence d'accès (RT5-75).

Remarque:

$f_{min}$  et  $f_{max}$  représentent les fréquences minimale et maximale de travail des CCD. Elles dépendent des caractéristiques électriques et dynamiques propres à chaque type de CCD: consommation, temps d'accès, temps de rafraîchissement, ..etc.

### 2.2.2) Organisation série-parallèle-série (SPS).

La caractéristique principale de cette organisation, est d'avoir deux registres à décalage série de  $N$  bits qui servent à un registre multi-canal contenant  $N$  registres à décalage en parallèle.

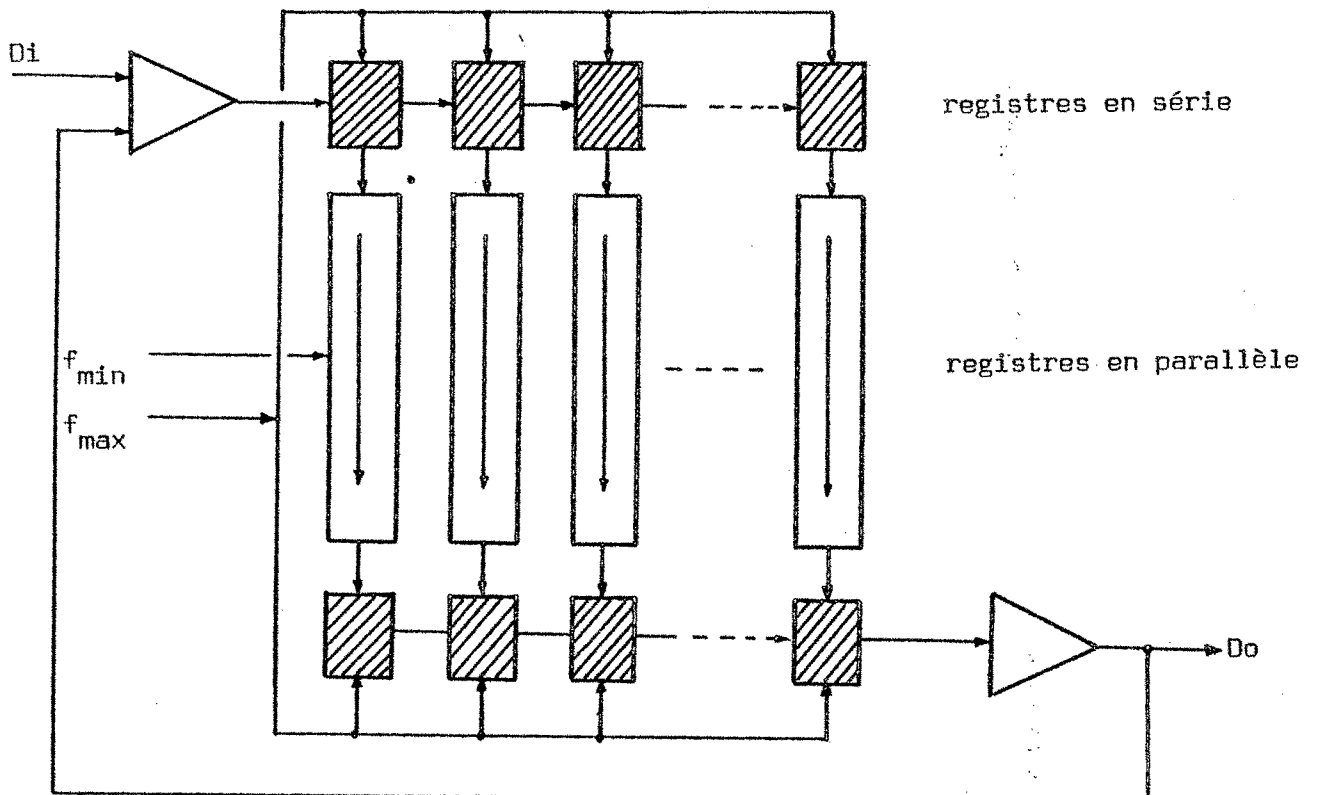


Figure 4.5 La configuration interne d'une organisation en SPS.

L'information est placée, en entrée et en sortie, à fréquence maximale ( $f_{max}$ ) dans les deux registres à décalage série, puis l'information est décalée à fréquence minimale ( $f_{min}$ ) dans les registres parallèles. Le temps d'accès ( $t_a$ ) à l'information est le temps de décalage dans le registre parallèle plus le temps de décalage dans le registre série.

Cette organisation est caractérisée par un transfert de données rapide (E/S) et une consommation faible, car les bits de données sont transférés dans les registres à décalage en parallèle à fréquence minimale.

L'inconvénient de cette organisation est d'avoir deux horloges de fréquences différentes et d'avoir un temps d'accès long ( $t_a$ ).

### 2.2.3) Organisation matricielle: LARAM (Line Adressable Ramdon Acces Memory)

La caractéristique principale de cette organisation est d'avoir une matrice d'adressage (sélection) des lignes de registres à décalages.

Ces lignes sont sélectionnées aléatoirement mais leur contenu est accédé en série (figure 4.6).

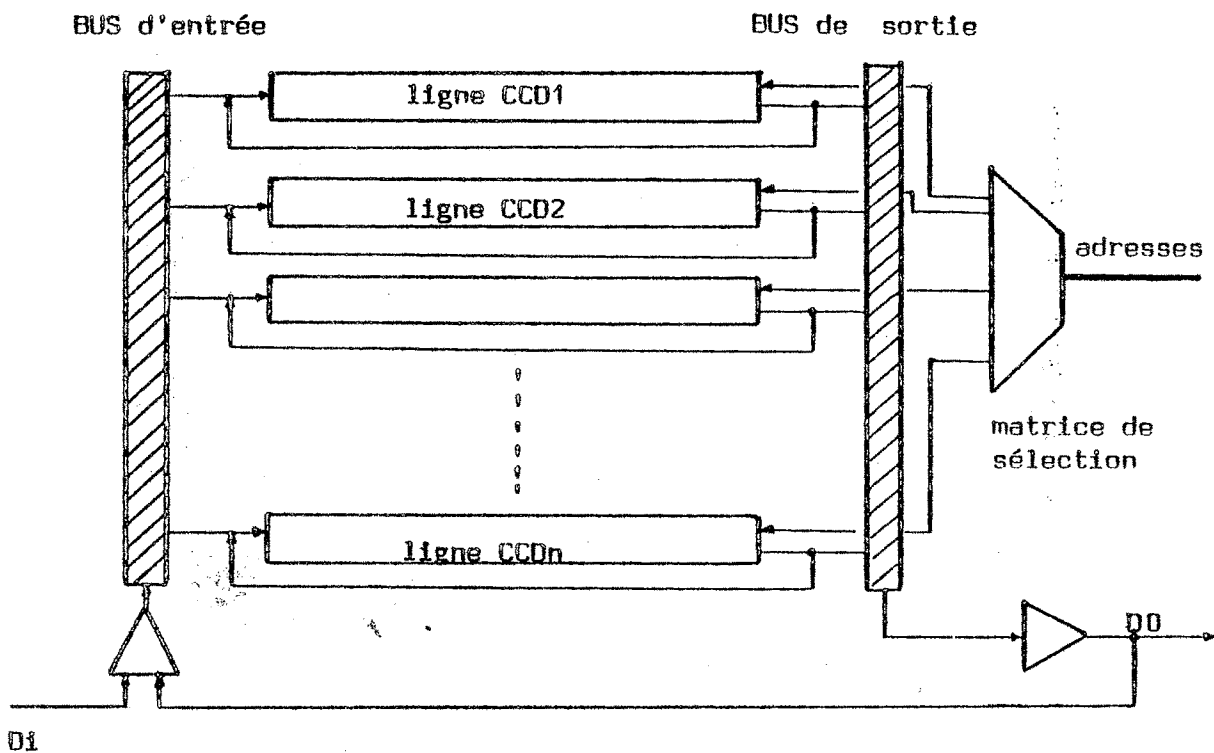


Figure 4.6 Organisation en LARAM.

Au moment de sélectionner une ligne (lecture/écriture), une horloge rapide est appliquée seulement à la ligne CCD sélectionnée; les autres lignes restent en mode de décalage de rafraîchissement (les différentes pistes restent synchronisées) la consommation de l'ensemble et les besoins d'horlogerie sont minimaux.

Le temps d'accès dépend du nombre de bits par ligne, de l'organisation de ces bits (mots) et de la fréquence de l'horloge de décalage (RT5-75).

2.3) Tableau comparatif de ces organisations à même capacité.

<u>Caractéristique</u>	<u>Serpentine</u>	<u>SPS</u>	<u>LARAM</u>
Capacité(bits)	16384	16384	16384
Organisation	256 x 16 x 4	4096 x 4	128 x 32 x 4
Temps d'accès (à 5 Mhz)	51.2 micro-sec	819.2 micro-sec	25.6 micro-sec
Consommation	400 mW	150 mW	250 mW
Temperature	85 °C	70 °C	55 °C
Surface	1.	0.9	1.1
Capacité d'entrée	1000 pf	1000 pf	100 pf
Coût de fabrication	1.0	0.8	1.1

Tableau 1. Les différentes caractéristiques à chaque organisation (CCE-76).



### 3.) LA MEMOIRE CCD CHOISIE: L'INTEL 2416

Nous avons choisi pour l'étude et la conception de modules mémoires CCD, les mémoires organisées de façon LARAM: l'INTEL 2416.

Ces mémoires seront assemblées de telle manière qu'elles s'adaptent aux structures d'adressage et aux données des microprocesseurs. Toutefois le problème d'adresser l'information contenue dans la mémoire CCD demeure entier.

#### 3.1) Organisation des mémoires CCD 2416.

Elles sont organisées en 64 registres à décalages de 256 bits chacun. L'information contenue dans ces registres est simultanément décalée par un système de phases (RT4-75).

Il faut remarquer que les CCD sont des mémoires dynamiques elles doivent donc être rafraichies périodiquement. Le rafraichissement se fait automatiquement par un simple décalage.

En fixant une adresse de 6 bits(A0..A5) on sélectionne un des 64 registres contenus dans les CCD et on accède à un bit. Leur organisation est montrée par la figure 4.7

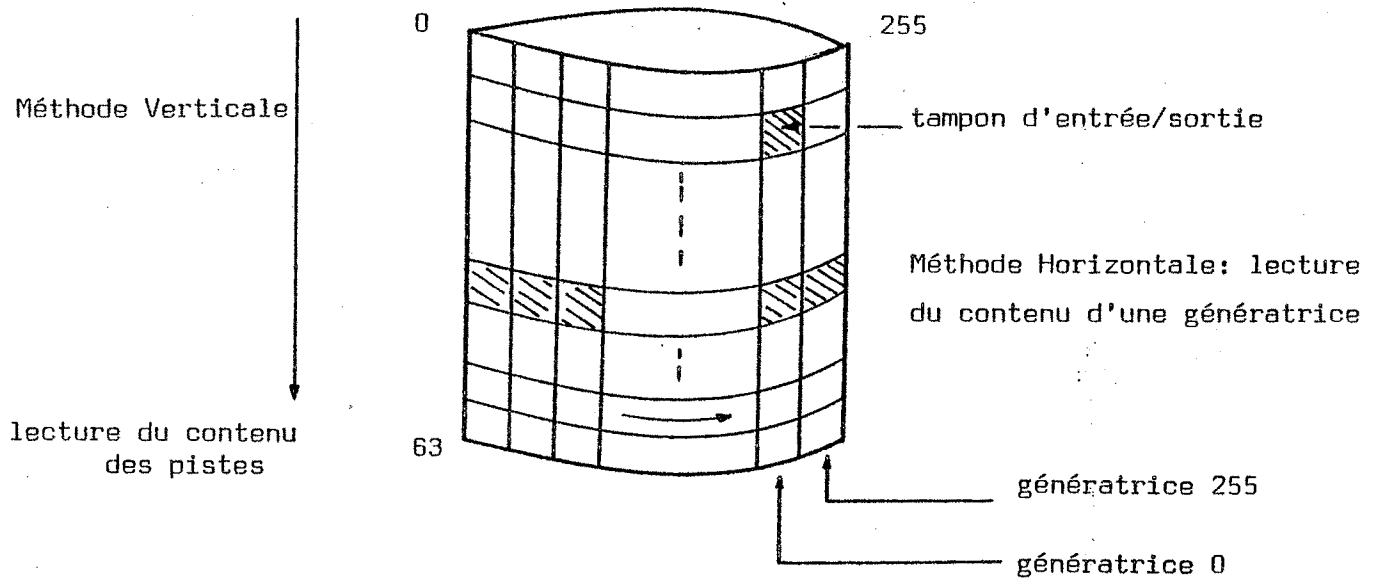


Figure 4.7 Organisation des CCD 2416.

Dans la figure 4.7, la CCD est représentée comme un cylindre contenant 64 pistes avec 256 bits par piste.

Les pistes représentent les registres à décalage et les génératrices les cellules à stockage. La vitesse de rotation du cylindre est contrôlée par la vitesse de succession des phases.

Il existe un tampon d'entrée/sortie par piste de façon à permettre des lectures et écritures sur les CCD. Nous pouvons considérer le cylindre d'informations comme tournant en face des tampons; chaque décalage place le contenu de la prochaine génératrice dans le tampon.

### 3.2) Méthodes d'accès.

Deux méthodes d'accès peuvent être considérées:

- la méthode verticale: lecture du contenu de plusieurs pistes pour un même décalage.
- la méthode horizontale: lecture du contenu d'une génératrice par décalages successifs.

#### 3.2.1) La méthode verticale: adressage par génératrice.

On accède à la génératrice en décalant le cylindre(phases) jusqu'à ce qu'on arrive au bon secteur(figure 4.8) , la lecture du contenu des pistes se fait en balayant les adresses A0...A5.

Le caractère dynamique des CCD fait que dans la pratique on ne peut pas accéder à toutes les informations contenues dans toutes les pistes d'une génératrice dans le temps d'un décalage. En conséquence il faut faire des décalages de rafraichissement de temps en temps. L'accès se fait donc par groupes séparés comme le montre la figure 4.8

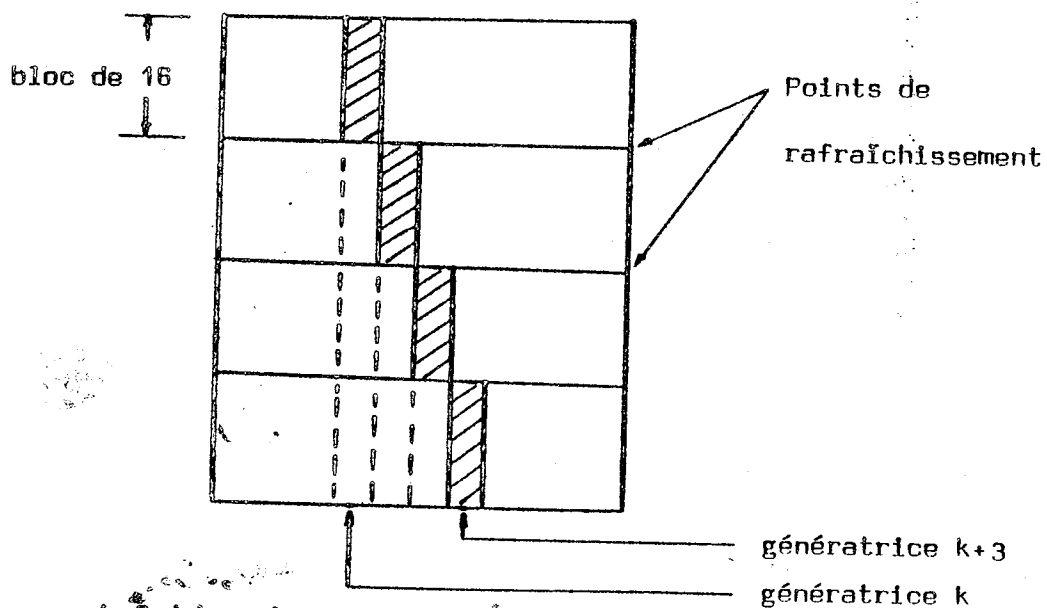


Figure 4.8 L'accès par bloc de 16.

Il faut remarquer qu'avec cette méthode la fréquence de rotation est minimale pour permettre un nombre maximum d'accès entre cycles de rotation de rafraîchissement de manière à obtenir un débit d'informations maximal.

Par la suite, on fera une analyse plus détaillée de la méthode. Les conditions spécifiées par le fabricant sont les suivantes:

- le temps maximal entre décalages est de 9000 ns,
- la durée d'un cycle d'accès élémentaire est de 460 ns,
- la durée d'un cycle de rafraîchissement est de 250 ns,
- chaque accès(E/S) prend 572.5ns (9000/16) pour un débit d'informations de 1.77 Mbits/sec.

Pour résoudre le problème de contrôle des différents accès entre décalages, on uniformise la quantité d'informations accédée à chaque cycle de décalage: on utilisera donc quatre cycles de rafraîchissement servant chacun à réaliser 16 accès à des pistes successives (figure 4.8).

### 3.2.2) Méthode horizontale.

Dans ce cas, l'information est placée sur les pistes. L'accès à cette information se fait au moyen d'un décalage suivi d'un accès pour chaque bit d'information, la sélection restant alors fixée sur la même piste. On obtient le contenu d'un registre de 256 bits après 256 décalages.

Dans cette méthode, la fréquence de rotation du cylindre doit être maximale pour minimiser le temps d'accès.

L'intervalle minimum d'un décalage est de 576 ns; c'est-à-dire un débit de 1.7 Mbits/sec (spécifications du constructeur) (RT4-75).

#### 4.) LE PROCESSEUR CCD.

##### 4.1) Le problème.

Il est souhaitable qu'un système, lorsqu'il est connecté à une unité périphérique transfère une grande partie du traitement des fichiers de l'unité centrale à l'unité périphérique.

L'utilisation des nouveaux matériels, tels que les microprocesseurs, et des techniques de la programmation structurée, facilitent la construction de processeurs spécialisés pour effectuer des tâches diverses; par exemple la gestion des fichiers dans des mémoires à masse.

Un domaine d'application des microprocesseurs, en architecture de machines informatiques, est leur utilisation pour munir des unités périphériques, d'une capacité de traitement.

Nous proposons d'associer un microprocesseur à une ressource de stockage (mémoire de masse) pour la conception des unités périphériques permettant des traitements répartis effectifs dans l'architecture des machines informatiques futures: comme exemple nous avons défini un processeur CCD permettant à l'utilisateur de travailler sur des fichiers (procédures de création, sauvegarde, et d'accès aux fichiers).

#### 4.2) Description du processeur CCD.

La description du processeur CCD, sous MICROD, est la suivante:

```

<processeur CCD> ::= <matériel-CCD> [←] <L.CCD>
<matériel-CCD> ::= <bloc processeur> <bloc périphérique>
<bloc périphérique> ::= <contrôleur CCD>
<L.CCD> ::= (logiciel de supervision du contrôleur CCD)
  
```

Le <bloc processeur> est représenté par le P.central.

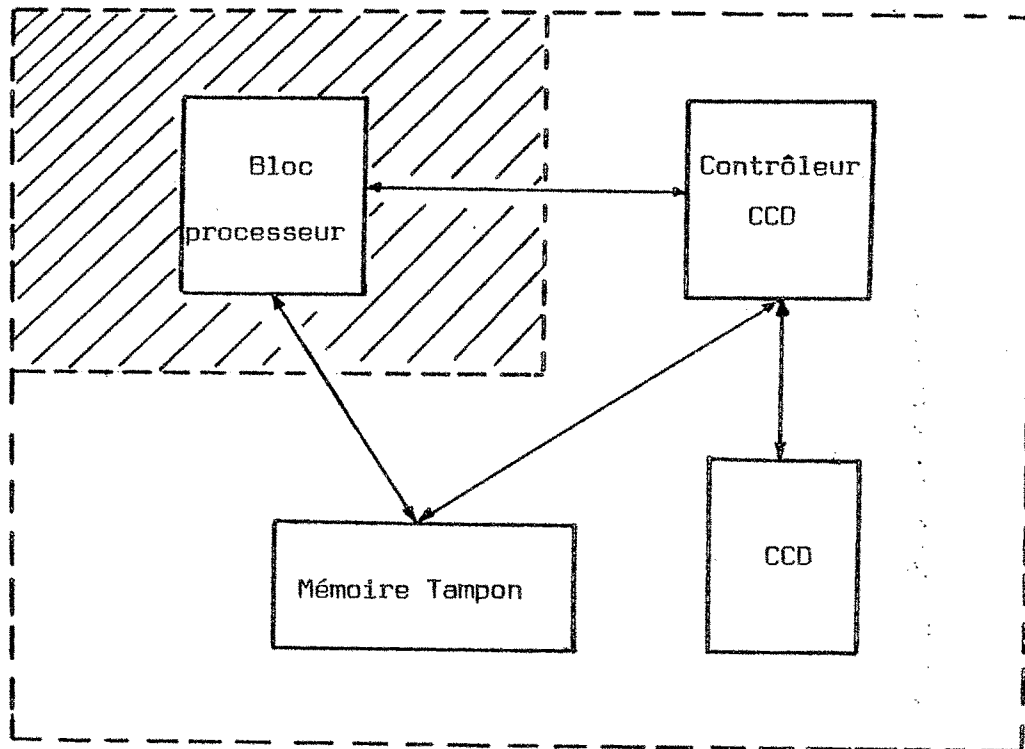


Figure 4.9 L'architecture du processeur CCD.

#### 4.2.1) Le contrôleur CCD.

Il est chargé d'effectuer des échanges entre la mémoire du P.central (TAMPON) et le module CCD, ainsi que de superviser et d'exécuter des ordres provenant du P.central.

Nous utilisons une procédure de DMA (accès direct à la mémoire) pour faciliter les échanges entre le P.central et le contrôleur CCD.

Les informations échangées entre le P.central et le contrôleur sont les suivantes :

- adresse piste/secteur,
- adresse TAMPON en mémoire,
- nombre de secteurs/pistes à lire ou écrire,
- état, synchronisation.

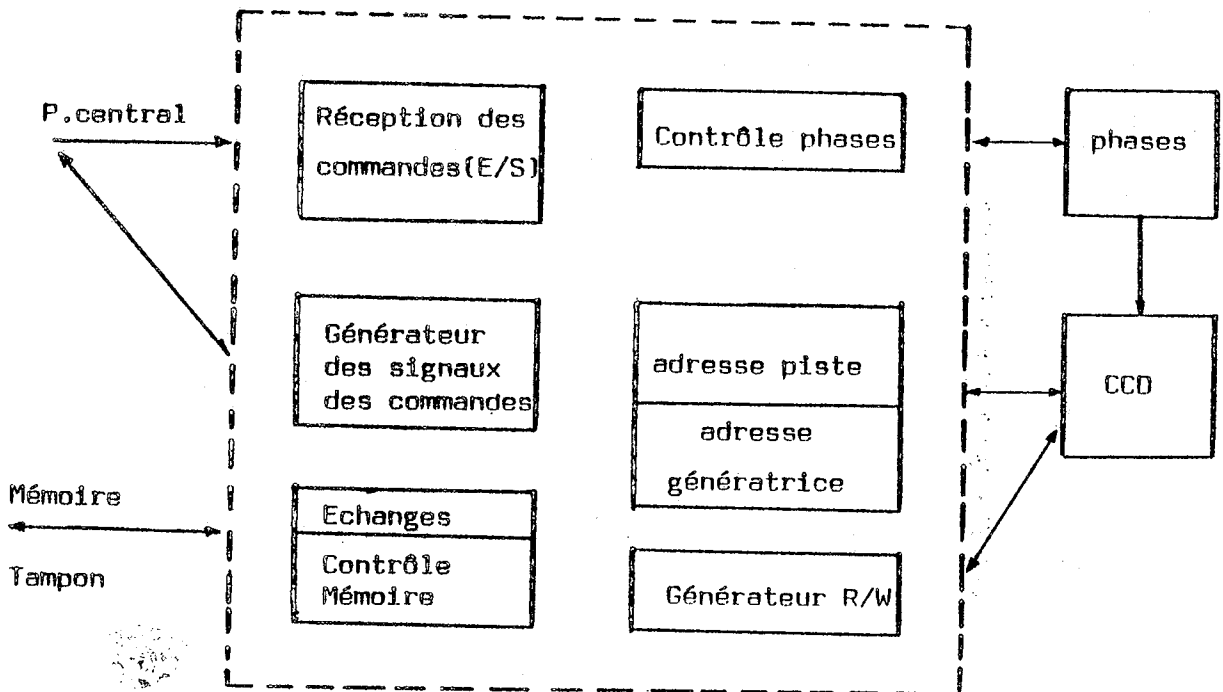


Figure 4.10 Le contrôleur CCD.

Il se décrit en MICROD de la manière suivante:  
 <contrôleur CCD> ::= <générateur des phases>  
                           <automate de contrôle>  
                           <module CCD>;

#### 4.2.1.1) Le générateur des phases.

Il permet de contrôler la vitesse de rotation du module CCD et les signaux de décalages par la génération des phases. Le circuit est représenté globalement par la figure 4.11.

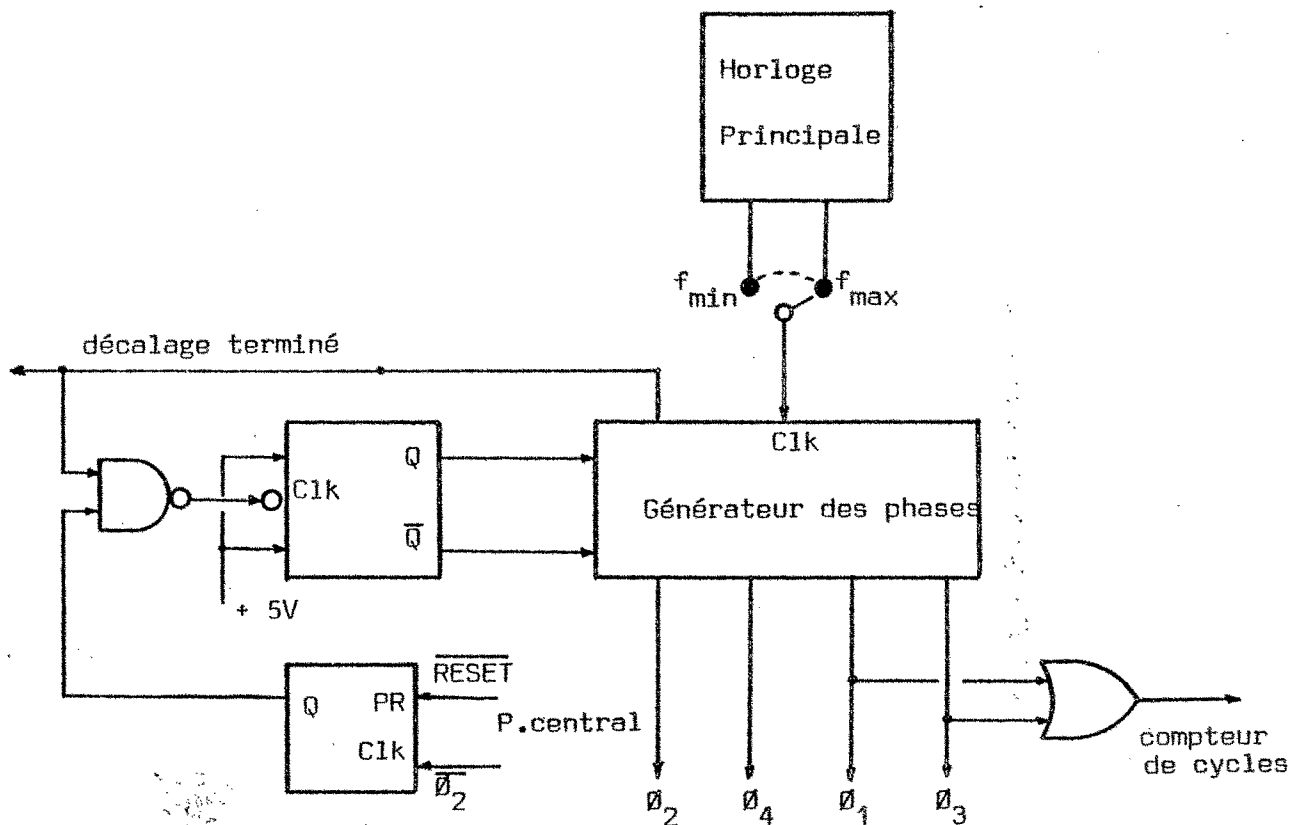


Figure 4.11 Le générateur de phases.



4.2.1.2) L'automate de contrôle.Méthode d'accès à l'information.

Les actions effectuées par le contrôleur CCD, chaque fois qu'il est appelé par le P.central pour lire ou écrire dans le module CCD, peuvent être décrites par les deux procédures ci-dessous.

PROCEDURE ACTIV;DEBUT

CO activation du contrôleur par le P.central;  
 stocker l'adresse du module CCD et l'adresse du  
 TAMPON dans le registre interne;  
 DEMAR;  
 accès à l'information dans le TAMPON / CCD

FIN;PROCEDURE DEMAR;DEBUT;

CO processus d'accès au TAMPON / CCD;  
TANTQUE non activation FAIRE rafraichissement;  
DEBUT  
 recevoir commande du P.central;  
 P.central ::= HALT; CO arrêt du P.central;  
 attente réponse du P.central; CO réponse à l'arrêt;  
 transfert des données au TAMPON ou au module CCD;  
 P.central ::= HALT; CO redémarrage du P.central;  
FIN;

FIN;

Le fonctionnement de l'automate.

Le transfert des données entre le module CCD et la mémoire TAMPON doit être fait le plus rapidement possible. C'est pourquoi on doit utiliser un automate câblé qui réalise cette fonction(DMA) de manière à éviter le transfert suivant:

Module CCD.....registre du microprocesseur.....TAMPON

La fonction principale de l'automate sera de commander le transfert du module (secteur/piste) dans le TAMPON en mémoire. De plus il permet de:

- générer l'horlogerie de contrôle pour les CCD (phases),
- générer les ordres vers le P.central:
  - \* arrêt (HALT),
  - \* fin du transfert (HALT),
  - \* recevoir et analyser l'ordre d'activation du contrôleur.

La réalisation de l'automate , du point de vue matériel, dépend du mode d'accès choisi (vertical/ horizontal).

Par la suite, on analysera l'automate selon les deux modes d'accès:

- vertical (adressé par secteur): automate GENERATRICE.
- horizontal (adressé par piste): automate PISTE.

4.2.1.2.1) L'automate en mode d'accès GÉNÉRATRICE.

L'algorithme correspondant à cette méthode est décrit dans la procédure GENDAM.

PROCEDURE GENDAM,

DEBUT

CO adressé par génératrice;

TANTQUE adr.gen.courant  $\neq$  adr.gen.recherché FAIRE

DEBUT

décalages;

adr.gen.courant ::= adr.gen.courant + 1

FIN;

adr.piste.courante ::= 0;

POUR i=1 JUSQUA 4 FAIRE

DEBUT

POUR j=1 JUSQUA 16 FAIRE

DEBUT

CO transfert bloc de 16 octets;

transfert données au TAMPON / CCD;

adr.piste.courante ::= adr.piste.courante+1

FIN

rafraichissement; CO point A de la figure 4.7;

FIN

FIN.

La réalisation matérielle de cette méthode d'accès est illustrée dans la figure 4.12

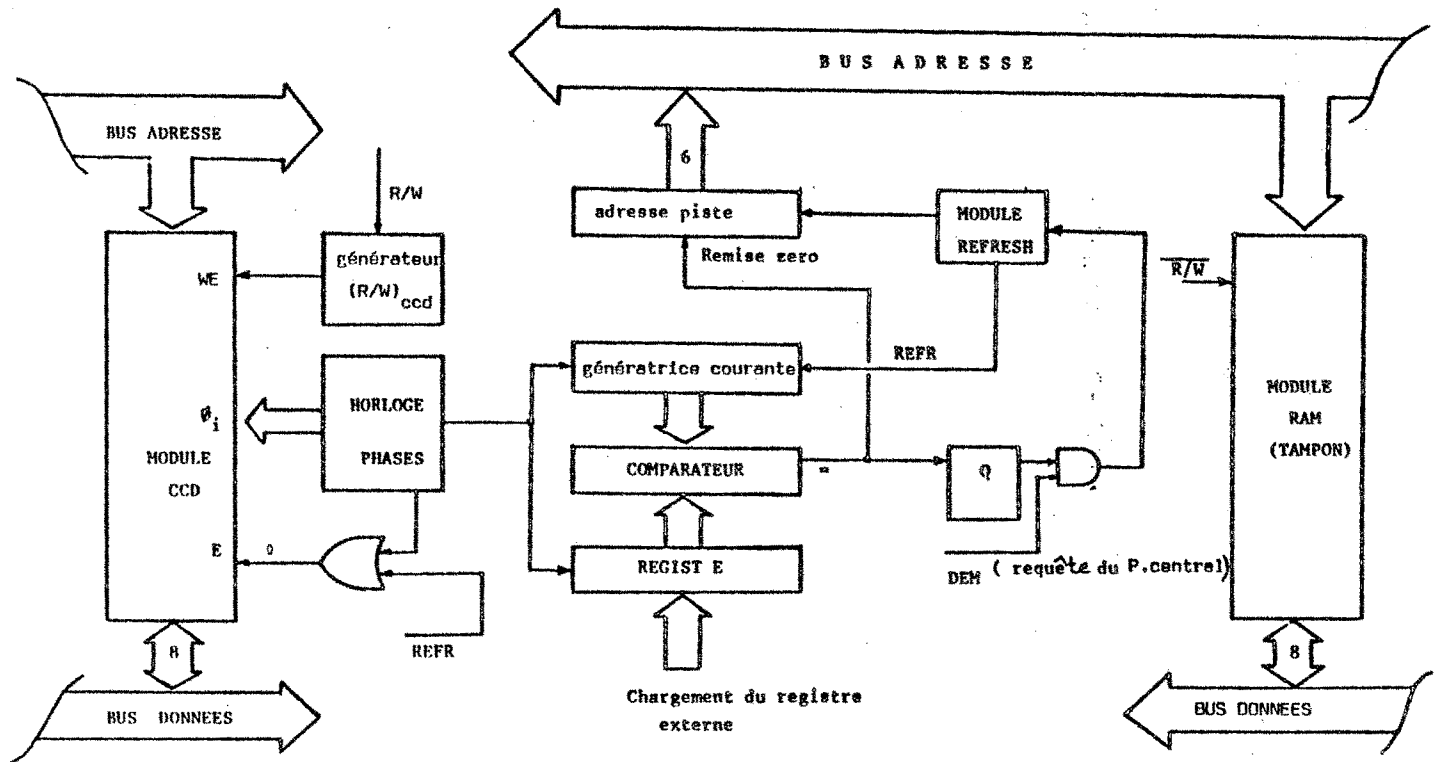


Figure 4.12 L'automate GENERATRICE.

Caractéristiques électriques de cette méthode.

Pour ce type d'adressage, nous avons les conditions suivantes:

\* la structure de l'information est la suivante:

<décalage><plusieurs E/S><décalage>

\* le temps du décalage est de 572.56 ns

\* le temps pour faire une E/S est de 750 ns

Comme nous l'avons vu dans la procédure GENDAM, il n'est pas possible de faire plus de 16 E/S à cause du rafraîchissement de la mémoire CCD. C'est pourquoi, au moment du transfert de l'information le contrôleur utilisera la technique suivante (figure 4.7).

<décalage><16 E/S><décalage><16 E/S>.....<décalage>

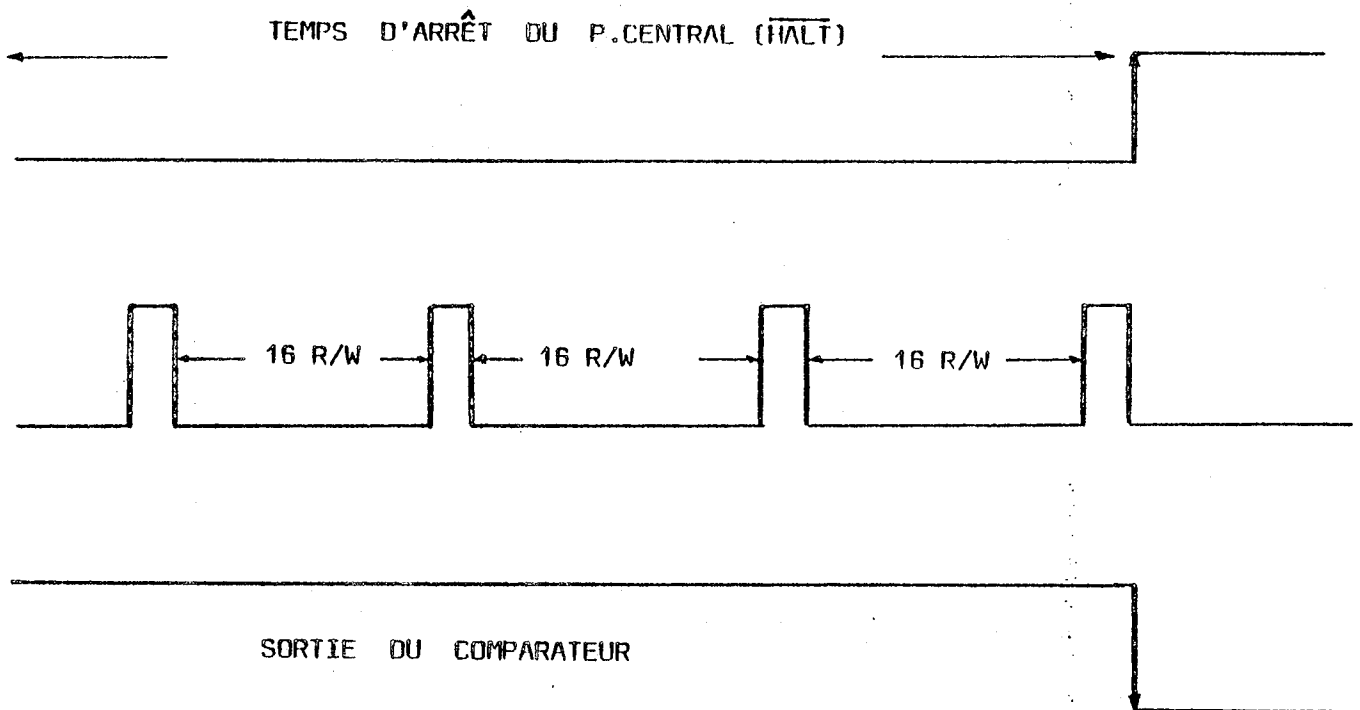


Figure 4.13 Les signaux de contrôle de cette méthode.

#### 4.2.1.2.2) L'automate en mode d'accès par PISTE.

L'algorithme de cette méthode est décrit ci-dessous.

```

PROCEDURE PISDMA;
DEBUT CO Piste;
  sélection piste (A0..A5);
  POUR j=1 JUSQUA 255 FAIRE
    DEBUT
      décalage;
      transfert octet au TAMPON / CCD
    FIN
  
```

FIN.

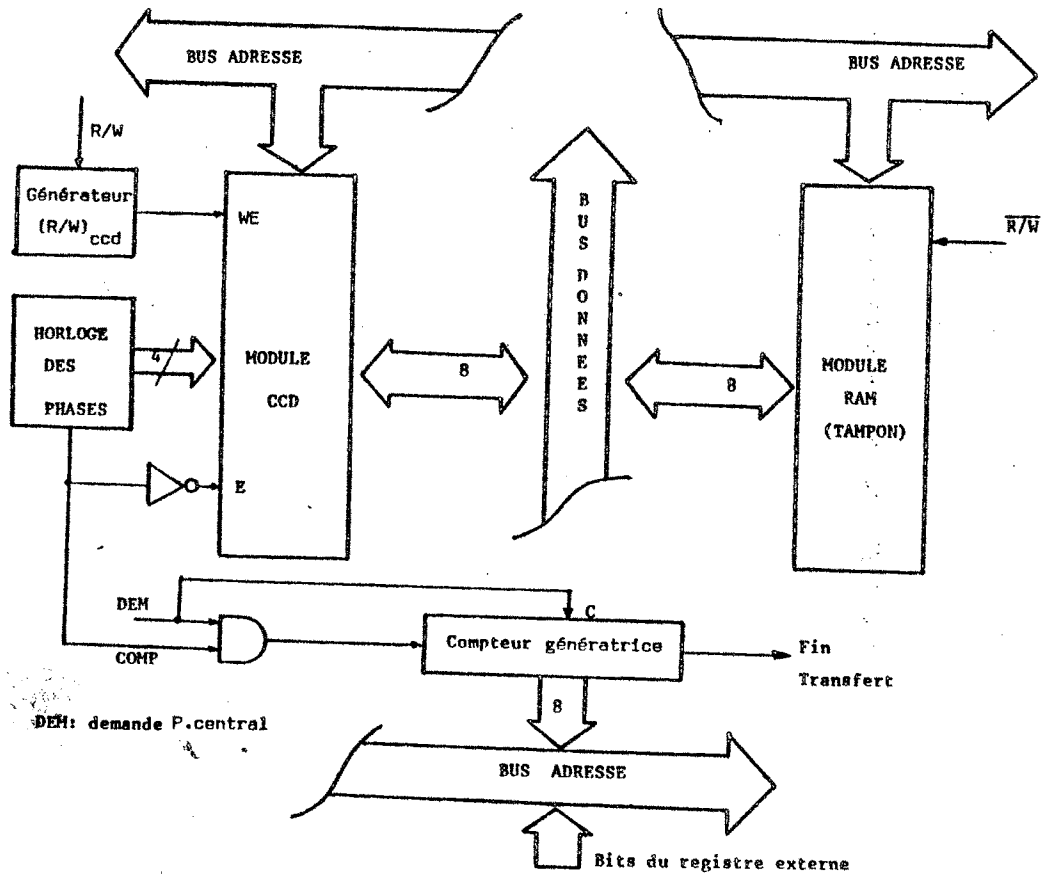


Figure 4.14 L'automate PISTE.

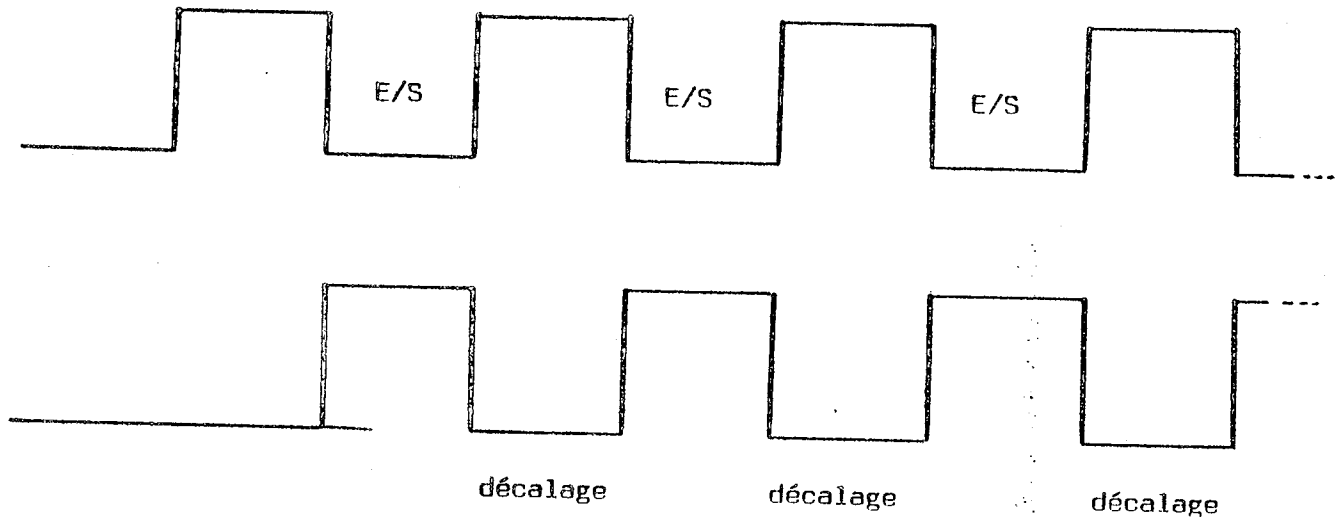
- caractéristiques électriques.

\* fréquence maximale de phases à 666.66 ns (précisée par le constructeur),

\* la structure des accès est la suivante:

<décalage><une E/S><décalage>....

Nous pouvons montrer ci-dessous les signaux correspondants:



Quand on n'accède pas à un bloc dans la mémoire CCD, il faut que les registres de la mémoire CCD tournent à la vitesse minimale (considérations de consommation minimale).

TABLEAU 2. Analyse comparative des méthodes.

	METHODE VERTICALE	METHODE HORIZONTALE
$C_{e/s}$	572.56 ns	500 ns
$F_r$	Minimale(55.55 KHz)	Maximale(666.66 KHz)
$D_i$	1.77 Mbits/sec	1.74 Mbits/sec
$N_a$	16	1
$T_d/2$	9000 ns	750 ns
$t_T$	38.29 micro-sec	320 micro-sec
$t_R$	160 micro-sec	-
$t_a$	198.29 micro-sec ( 1 bloc de 64 octets)	320 micro-sec ( 1 bloc de 256 octets)

Légende. $C_{e/s}$  : temps d'accès à un octet $T_d/2$  : temps entre décalages $D_i$  : débit d'information $t_R$  : temps de recherche $F_r$  : fréquence de rotation $t_T$  : temps de transfert $t_a$  : temps d'accès au bloc $N_a$  : nombre d'accèsDonnées.

$$t_a = t_T + t_R$$

$$t_T(\text{génératrice}) = 256(500 + 750) = 320 \text{ micro-sec}$$

$$t_T(\text{piste}) = 4(572.56 + 9000) = 38.29 \text{ micro-sec}$$

$$t_R(\text{génératrice}) = (1/2)t_T = 160 \text{ micro-sec}$$

$$t_R(\text{piste}) = \text{accès direct (A0, ..., A5)}$$



#### 4.2.1.3) Le module mémoire CCD.

Ce module est étudié sous une organisation de 16 pages de 64k-octets chacune. Chaque page est divisée en 4 sous-pages de 16K-octets. Pour l'adresser il nous faut 22 bits (vecteur d'adresses), considérant 1K-octets de mémoire TAMPON.

Ce module doit être organisé de façon à ce que la longueur du mot, soit égale à celle du microprocesseur utilisé (MC6800). Pour cela on groupe 8 boitiers pour obtenir 8 bits (octet) en parallèle et une capacité de 16K-octets, c'est-à-dire une sous-page.

Comme on peut avoir plusieurs pages dans le module, il est convenable d'avoir un automate et un générateur de phases par page. De cette manière, seule la page sélectionnée consomme une quantité importante d'énergie (fréquence de rotation maximale) pendant que les autres tournent à fréquence minimale.

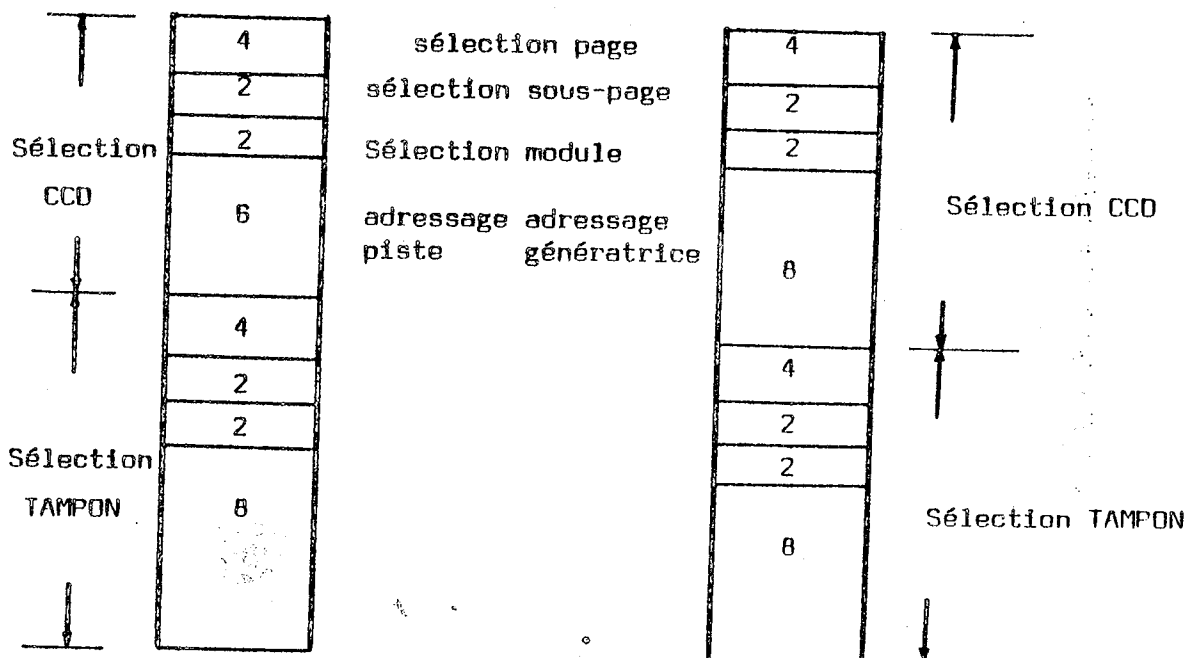


Figure 4.15 Les vecteurs d'adresses pour le cas génératrice/piste.

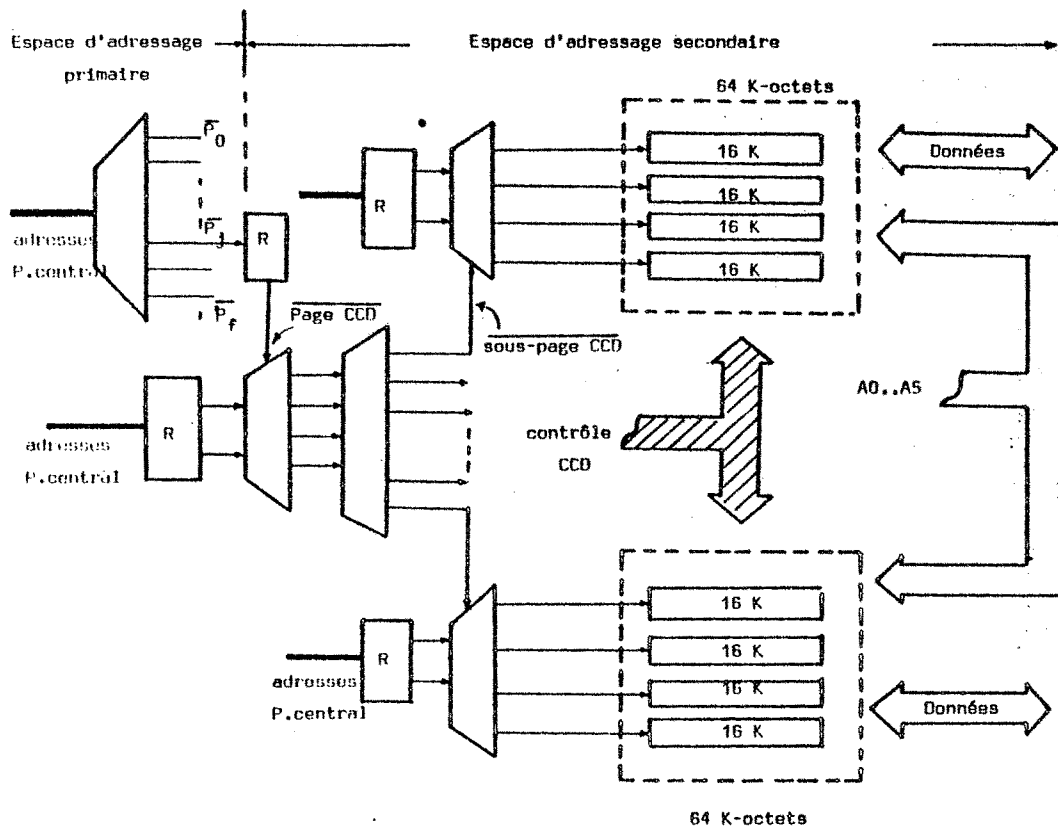


Figure 4.16 Organisation générale d'un système de mémoire CCD.

Nous pouvons implanter (16\*64 K-octets) dans l'espace de  $\overline{P_j}$  sous la forme de mémoires de massé.

#### 4.2.2) Le logiciel CCD: L.CCD

Le logiciel CCD (L.CCD) a été conçu comme un outil de mise au point et de maintenance des mémoires CCD et aussi, comme un noyau à partir duquel nous bâtissons des couches logicielles supérieures via un système d'exploitation.

Le L.CCD est représenté de la manière suivante:

<L.CCD> ::= <L.base><Log.CCD>

où

<L.base> ::= (logiciel système); CO moniteur SP-6800

<Log.CCD> ::= (logiciel d'essais et de contrôle du matériel de l'automate CCD)

Le L.CCD est présenté comme un moniteur qui a la fonction de dialoguer avec la console opérateur du P.central en utilisant une série de commandes pour:

- lire ou écrire une ou plusieurs pistes,
- lire ou écrire une ou plusieurs génératrices,
- tester le fonctionnement du module CCD.

5.) LA REALISATION: le contrôleur prototype.

Il opère avec la technique "STOP-START". Cette méthode, qui s'applique bien aux transferts des données, ne nécessite pas un gros matériel pour assurer la synchronisation entre le P.central et l'élément de stockage(module CCD).

Dans la figure 4.21, nous montrons comment une simple bascule peut synchroniser le P.central. Le contrôleur agit de la façon suivante:

Une instruction de sortie (respectivement d'entrée) s'exécute, actionne la bascule qui arrête le P.central (HALT); celui-ci envoie une réponse(BA) qui commande le démarrage du contrôleur, qui débute en cherchant les adresses (précédemment placées par le P.central) où il va ranger les données demandées.

Une fois qu'il a décidé que le transfert était terminé, il renvoie le contrôle au P.central(HALT). Cette interface est simple et elle a un temps de réponse comparable à celui d'un transfert DMA.

De son côté le logiciel est aussi simple:

- les boucles d'attente ne sont pas nécessaires,
- on utilise des procédures cablées de gestion des CCD: réalisées par l'automate,
- l'instruction de sortie(STA) arrête le P.central jusqu'à la réponse du contrôleur.

Nous avons étudié le contrôleur PISTE (relatif à l'accès par PISTE) d'abord, pour avoir une idée plus concrète des méthodes d'accès aux mémoires à CCD, mais aussi pour faire des essais sur les débits d'informations à transmettre entre le TAMPON et le module CCD.

### 5.1) Le générateur de phases.

Ce module permet de contrôler la vitesse de rotation du module CCD et les signaux de décalage propres à chaque phase.

Il est divisé en deux parties:

- l'horloge maître,
- le générateur des phases proprement dit.

#### 5.1.1) L'horloge maître.

L'horloge maître oscille à une fréquence de 5.33 Mhz ( $f_{max}$ ) laquelle est divisée par huit pour la génération des quatre phases. On a alors une fréquence de 666.6 KHz par phase, ce qui correspond aux 1500ns données par le constructeur.

De la même manière on divise par huit la fréquence minimale de 444.44KHz ( $f_{min} = f_{max}/12$ ), qui correspond aux 18000ns spécifiée par le constructeur.

Le changement de fréquence est fait par le P.central au moyen de BA, de telle manière que lorsque l'on accède à la mémoire CCD, soit en lecture, soit en écriture, on le fera à la fréquence maximale; par contre quand on n'y accède pas, les phases tournent à fréquence minimale ( $f_{min}$ ). Ce changement de fréquence est destiné à obtenir une consommation minimale des CCD.

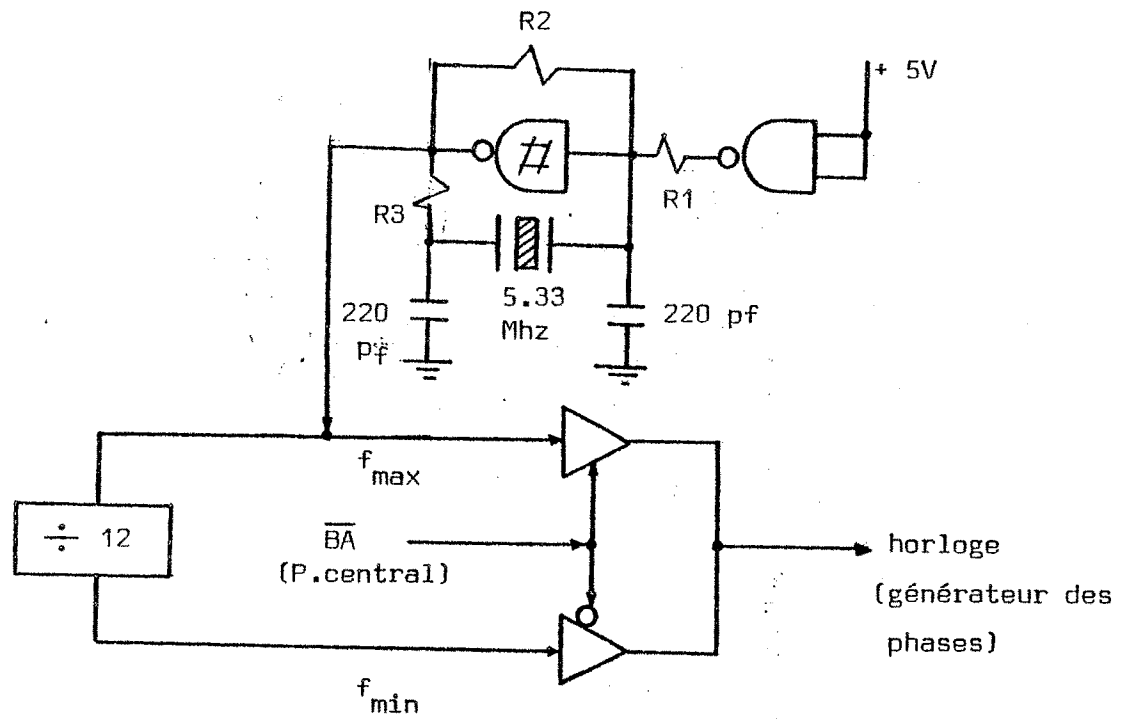


Figure 4.17 L'horloge maître.

#### 5.1.2) Le générateur des phases.

Les CCD du type 2416, ont besoin de quatre phases pour effectuer le décalage des données et pour leur rafraîchissement. On montre le circuit réalisé par la figure 4.18.

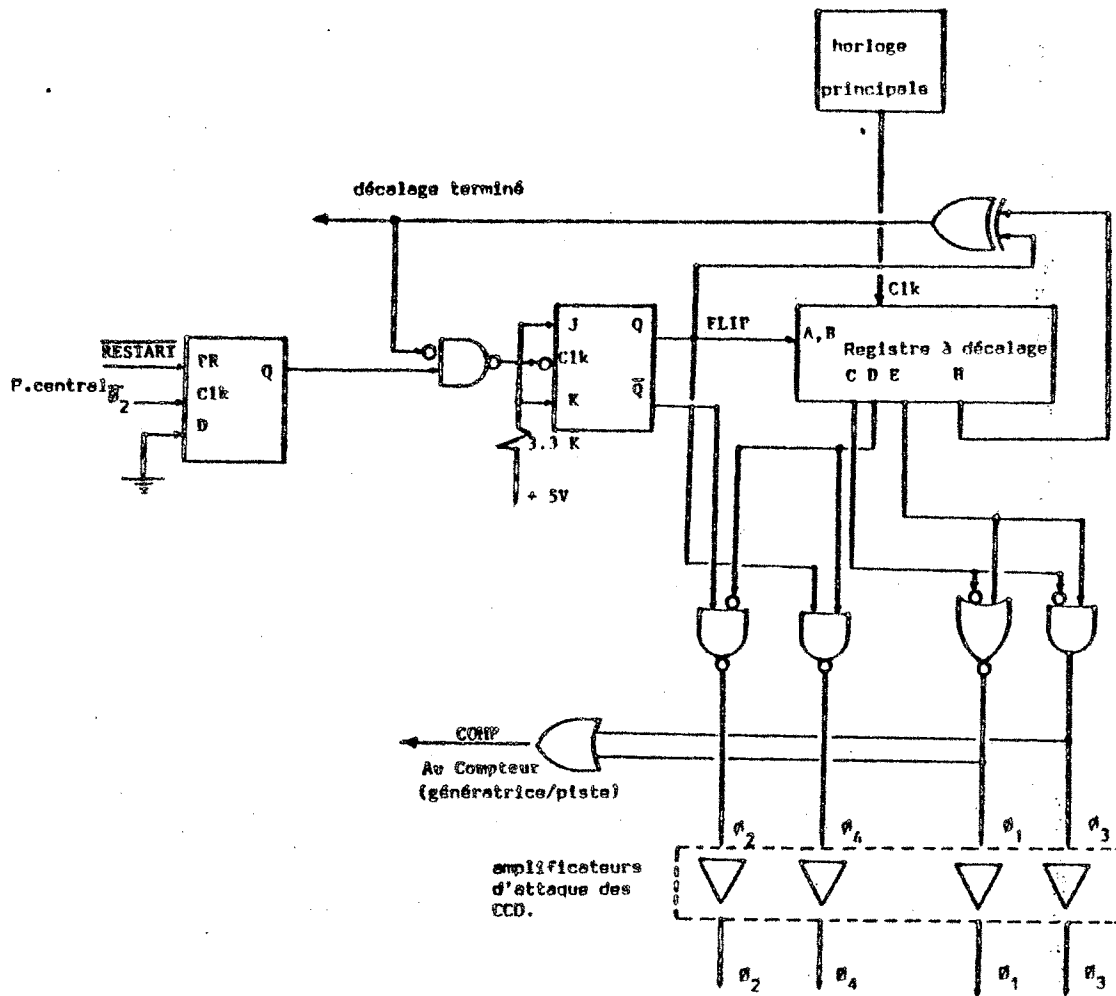


Figure 4.18 L'horloge de commande des CCD.

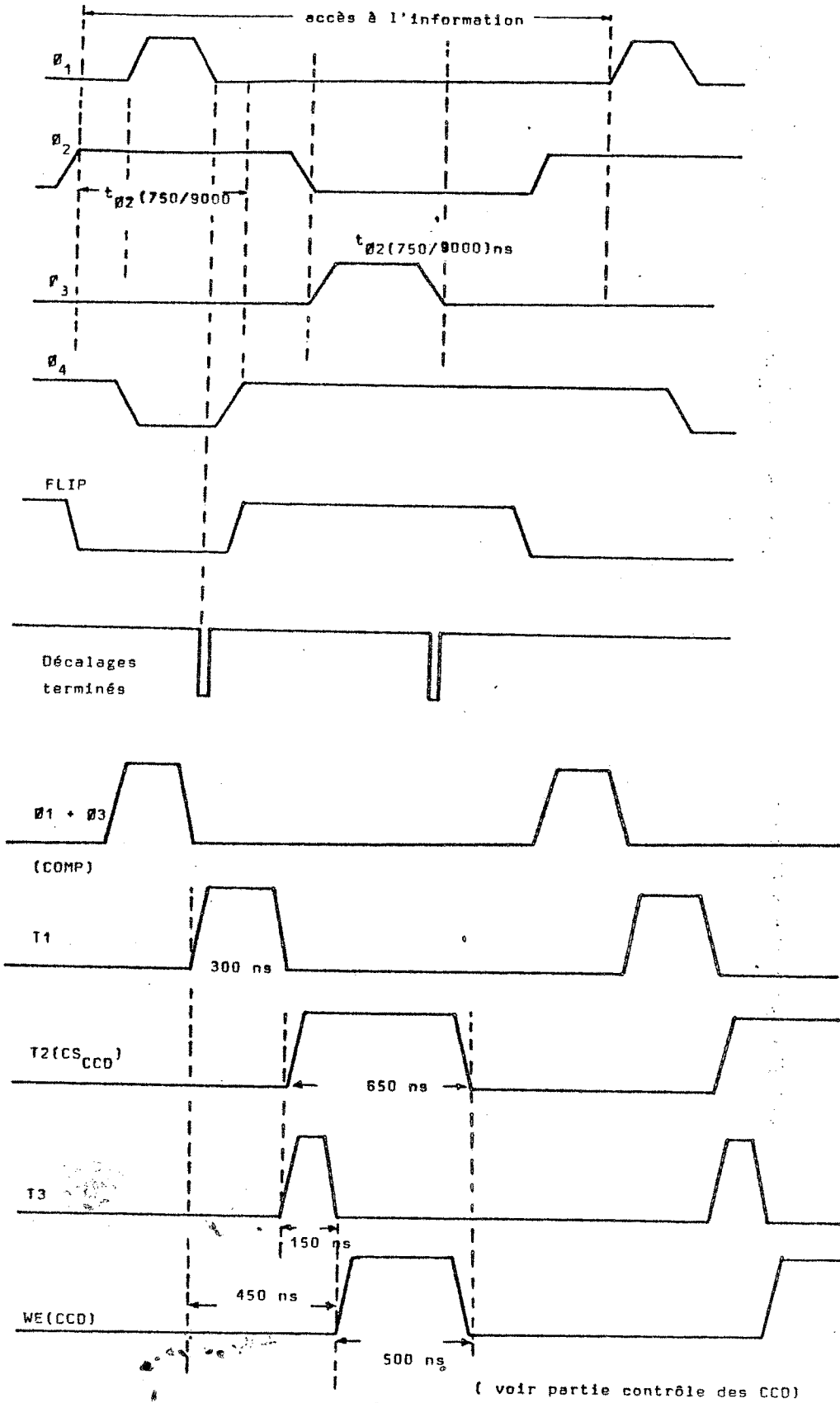


Figure 4.19 Les signaux propres au module CCD.



### 5.2) Le fonctionnement du contrôleur.

Pour accéder aux CCD en écriture ou en lecture, il faut le faire entre les intervalles  $\emptyset 1$  et  $\emptyset 4$  ou entre  $\emptyset 3$  et  $\emptyset 2$ , on prend donc comme point de départ les phases  $\emptyset 1$  et  $\emptyset 3$  pour générer les signaux de sélection (CHIP SELECT, CHIP ENABLE, WRITE ENABLE) propres aux CCD (figure 4.20). Ainsi le signal ( $\emptyset 1 + \emptyset 3$ ) détermine la fréquence à laquelle les données sont décalées.

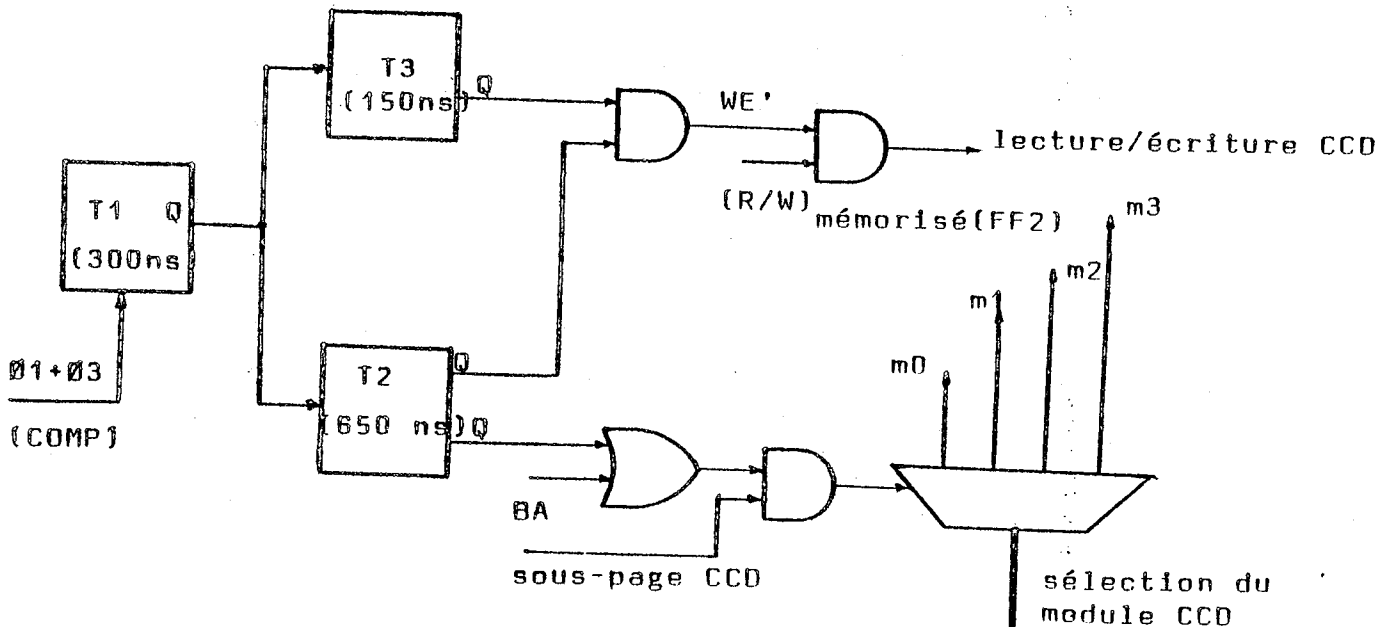


Figure 4.20 La partie contrôle des CCD.

Le diagramme de la figure 4.20, montre que le module CCD est sélectionné lorsque le P.central envoie l'ordre de transfert activant la ligne BA, sinon le module est isolé.

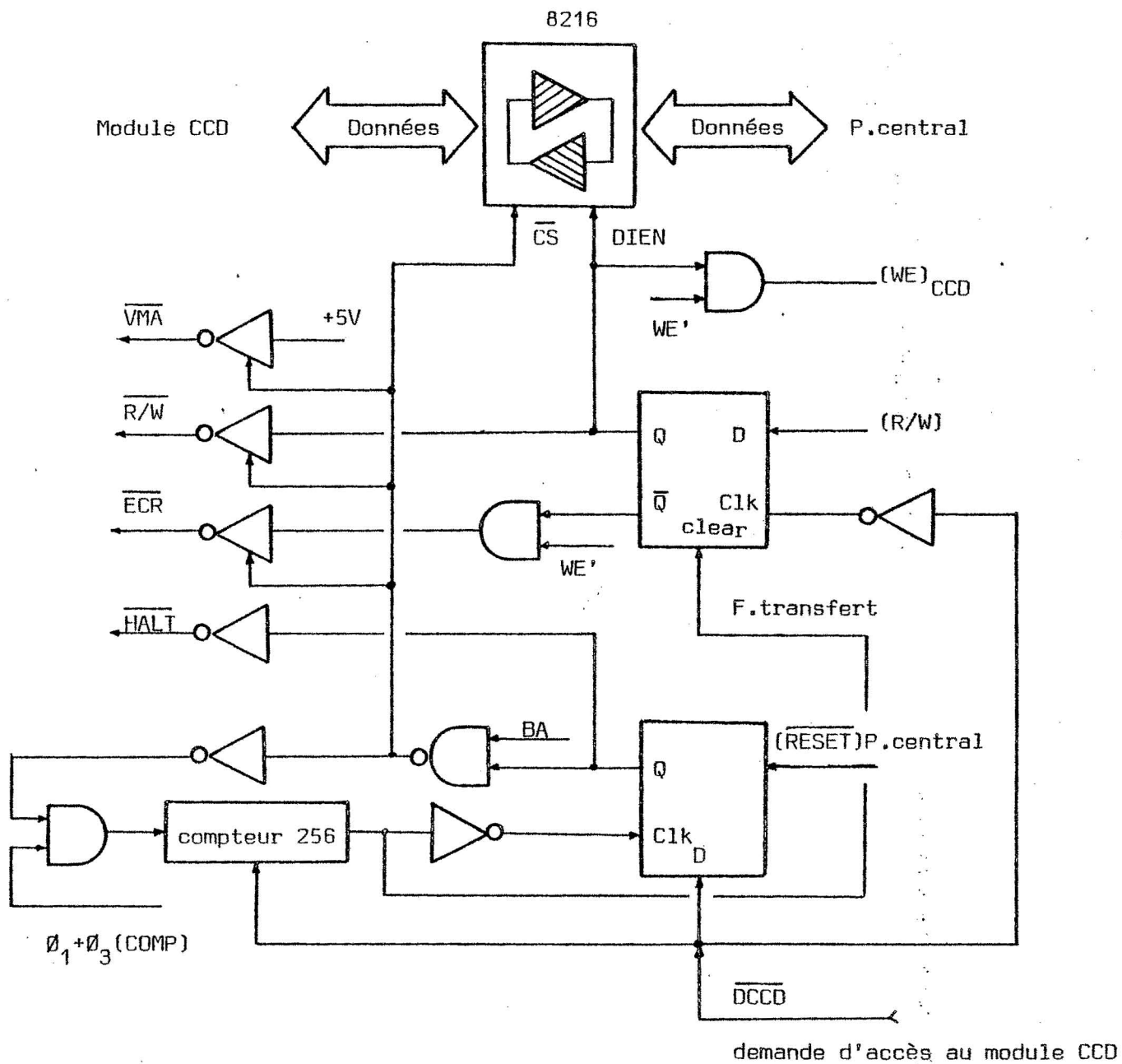


Figure 4.21 La partie commande vers la mémoire et le P.central

- Analyse d'écriture / lecture dans les CCD.

Le contrôleur CCD est activé par l'adresse  $\overline{DCCD}$ , la bascule(FF1) s'active arrêtant le P.central( $\overline{HALT}$ ).

La ligne  $\overline{HALT}$  reste active jusqu'à ce que les 256 décalages(01+03) soient passés, puis le contrôleur réinitialise la bascule pour redémarrer le P.central. Simultanément pendant que le P.central est arrêté la bascule(FF2) est actionnée par l'adresse( $\overline{DCCD}$ ) et sa sortie  $\overline{Q}$  prend la valeur du R/W qui sera "1" pour une écriture et "0" pour une lecture (figure 4.21).

Pendant que le P.central est arrêté, le contrôleur fournit avec l'aide des registres REG1 et REG2 (figure 4.25) l'adresse de la mémoire dans le TAMPON ou dans le module CCD, d'où il doit transférer le contenu d'un bloc de base (256 octets) vers le module CCD sélectionné (écriture) ou vers la mémoire dans le TAMPON (lecture).

Remarque.

On pourrait accélérer le contrôleur en adressant la mémoire par le point où en est la mémoire CCD et effectuer un transfert bouclé.

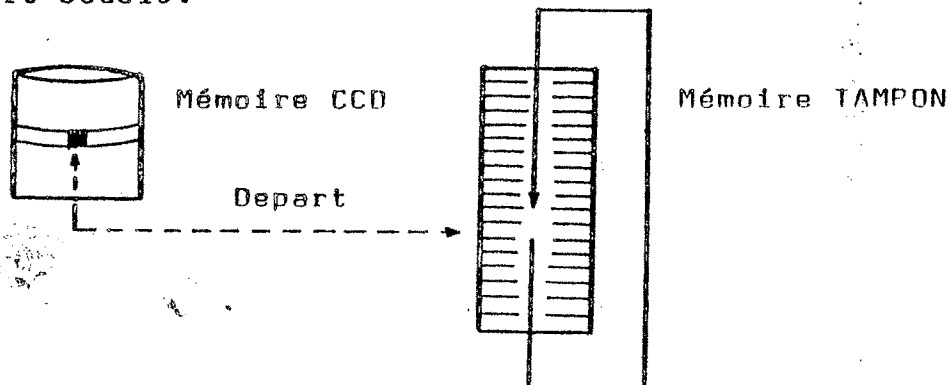


Figure 4.22 Le transfert bouclé.

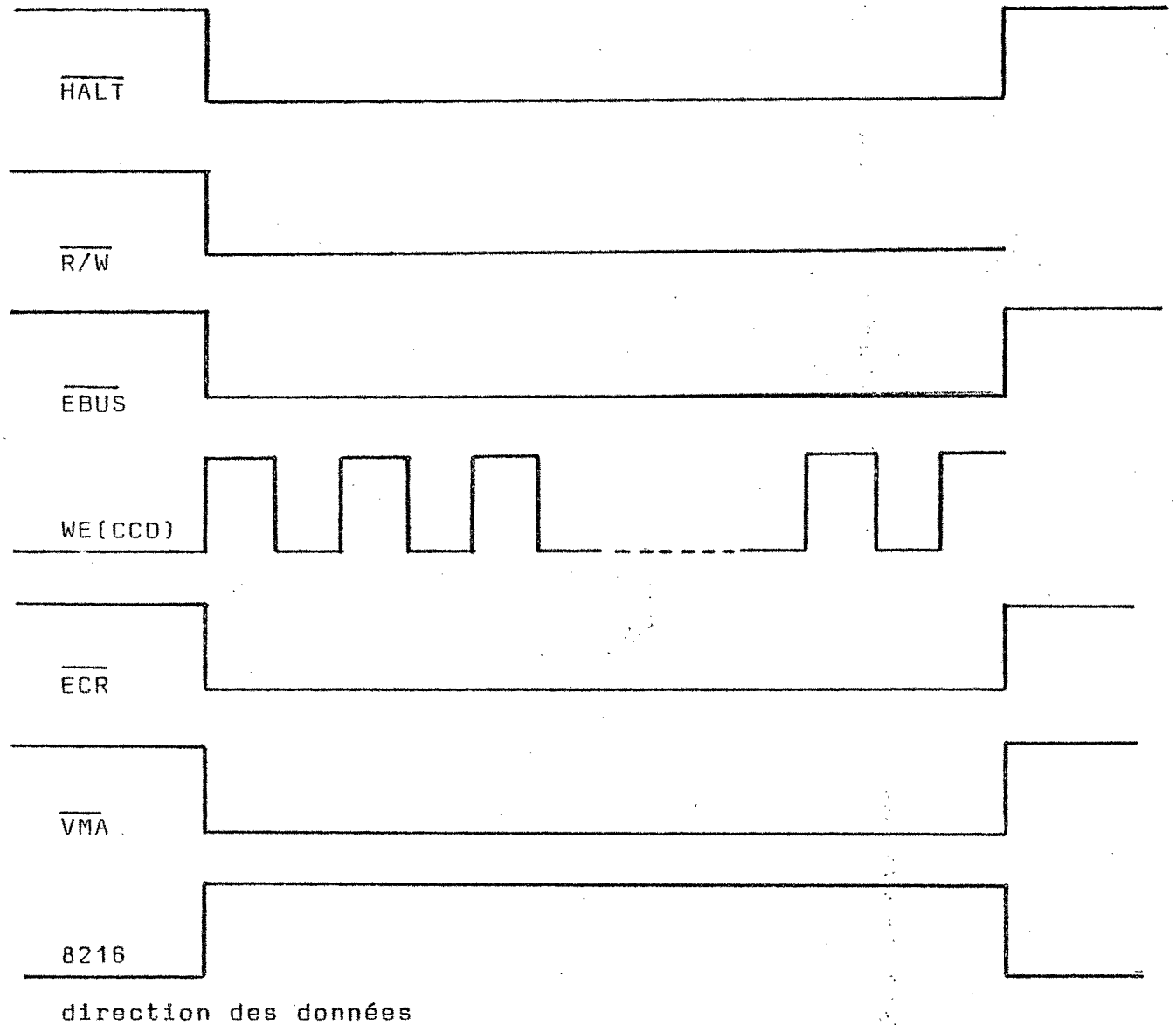


Figure 4.23 Diagramme des signaux en écriture.

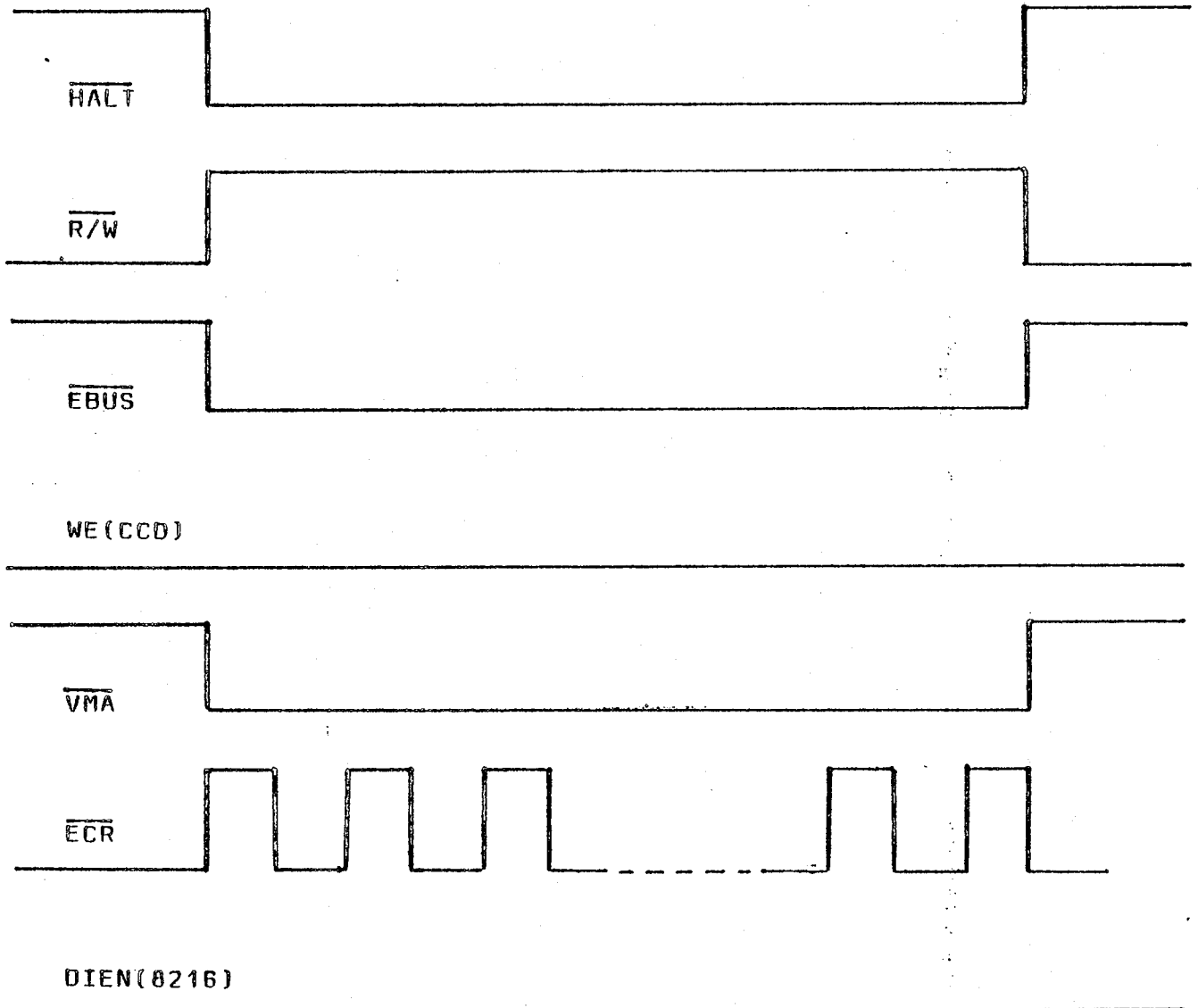


Figure 4.24 Diagramme des signaux en lecture.

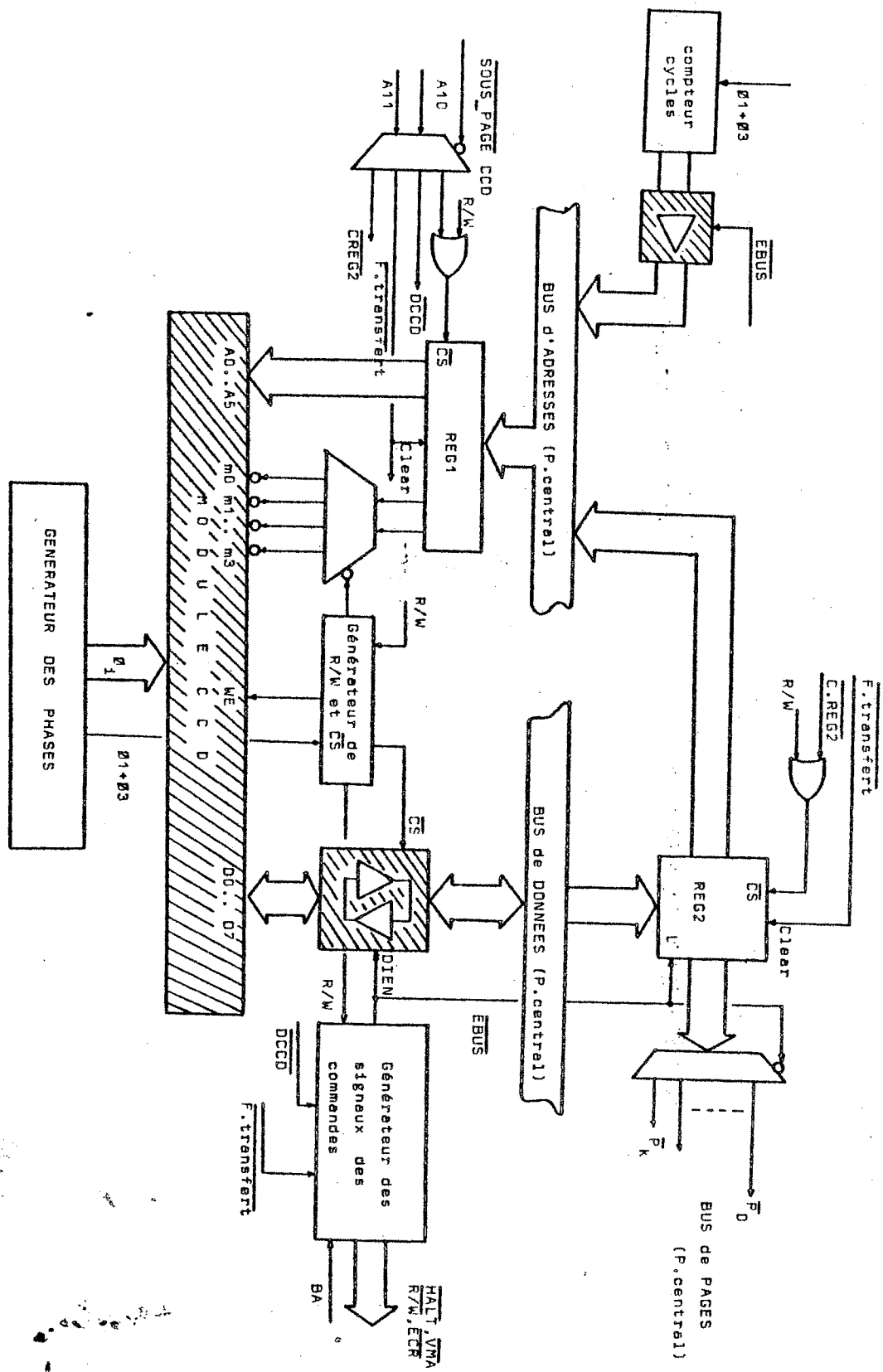


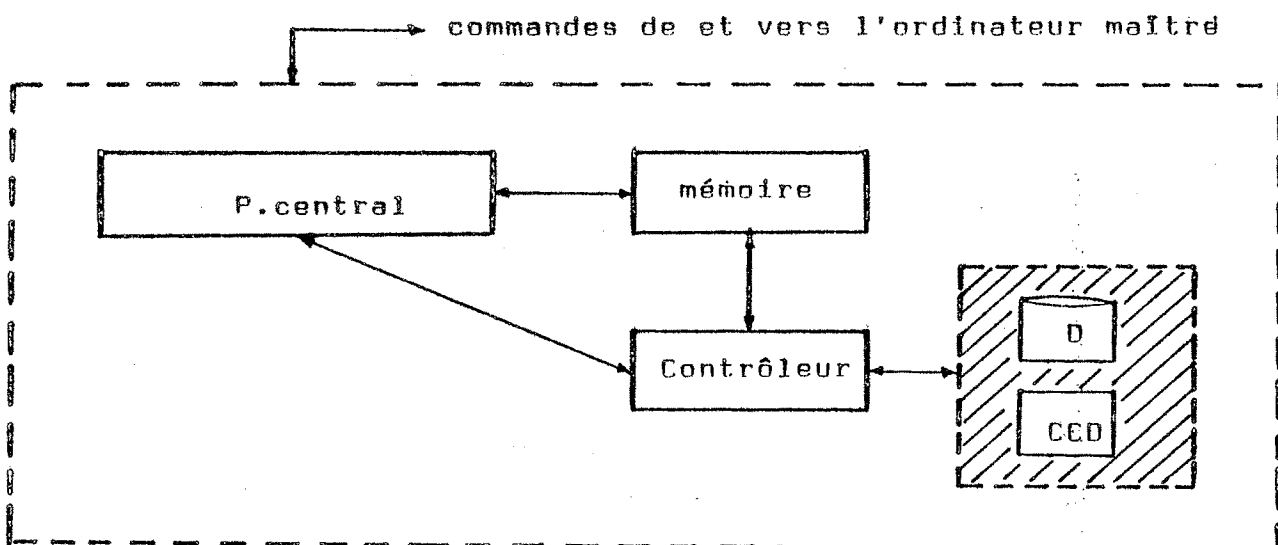
Figure 4.25 Diagramme général du contrôleur CCD.

## 6.) ROLE DU P.central.

L'utilisation d'un microprocesseur permet d'effectuer aisément la gestion de la mémoire à couplage de charges.

Les principaux rôles du P.central sont:

- de gérer les échanges entre l'ordinateur et les dispositifs mémoires:
  - \* la réception de commandes émises par l'ordinateur,
  - \* l'initialisation des échanges,
  - \* le lancement des opérations cablées,
  - \* la supervision des échanges en déclenchant une alarme si nécessaire.
- d'assurer l'adaptation entre le langage de l'ordinateur maître et le langage de commande (L.CCD) de la mémoire CCD.



D: disque souple

Figure 4.26 Le processeur de contrôle.

## CHAPITRE V

### LE CONTROLEUR DISQUE SOUPLE

- 1.) INTRODUCTION.
- 2.) LE CONTROLEUR DISQUE SOUPLE.
- 3.) EXTENSION DU CONTROLEUR: LE PROCESSEUR "FLOPPY"



## CHAPITRE V

### LE CONTROLEUR DISQUE SOUPLE

#### 1.) INTRODUCTION.

Ce chapitre constitue la réalisation tant matérielle que logicielle d'un contrôleur de disque souple périphérique du P.central.

Ce contrôleur est réalisé à l'aide d'un circuit FDI771 de WESTERN DIGITAL, qui transfère les bits d'informations entre l'unité disque souple choisie et le P.central.

Une extension prévue de ce contrôleur est la réalisation d'une carte processeur fonctionnel associée à l'unité disque souple. Un microprocesseur MC68800 rapide inséré sur cette carte se charge alors des transferts, sur ordre du P.central ou de sa propre initiative.

## 2.) LE CONTROLEUR DISQUE SOUPLE.

Le contrôleur de disque souple est un module du bloc périphérique (c.f.III) . Ce module est défini, sous MICROD, de la manière suivante:

$\langle \text{contrôleur} \rangle ::= \langle \text{matériel d.souple} \rangle \left[ \leftarrow \right] \langle \text{L.souple} \rangle$

où:

$\langle \text{matériel d.souple} \rangle$ : représente l'ensemble du matériel pour le contrôle électrique de l'unité de disque souple.

$\langle \text{L.souple} \rangle$ : ce logiciel est représenté par un moniteur dit moniteur "floppy" qui réalise les opérations de base pour la lecture, l'écriture et le contrôle du mouvement du bras de l'unité de disque souple.

### 2.1) Le matériel disque souple: matériel d.souple.

$\langle \text{matériel d.souple} \rangle ::= \langle \text{circuit contrôleur} \rangle$   
 $\langle \text{interface P.central} \rangle$   
 $\langle \text{interface unité.d.souple} \rangle;$

$\langle \text{interface unité.d.souple} \rangle ::= \langle \text{circuit.trans.récep} \rangle$   
 $\langle \text{liaison} \rangle$   
 $\langle \text{unité d.souple} \rangle;$

où:

$\langle \text{circuit contrôleur} \rangle$ : est représenté par le circuit FD1771 qui a une fonction d'interface entre le P.central et l'unité de disque souple. C'est-à-dire qui gère l'exécution des ordres de lecture/écriture pour l'unité de disque souple, envoyés par le P.central.

<interface P.central>: c'est un ensemble de circuits pour:

- la mise en forme des signaux provenant du P.central (R/W,  $\#2$ , adresses).
- la sélection du circuit contrôleur par le P.central,
- le contrôle des entrées/sorties du circuit contrôleur par le P.central.

<circuit.trans.récep>: C'est l'ensemble des circuits pour envoyer et recevoir des informations de l'unité de disque souple (amplificateurs de ligne).

<liaison>: ensemble de lignes véhiculant des informations entre l'unité de disque souple et le contrôleur.

<unité d.souple>: unité de disque SHUGART modele SA-100.

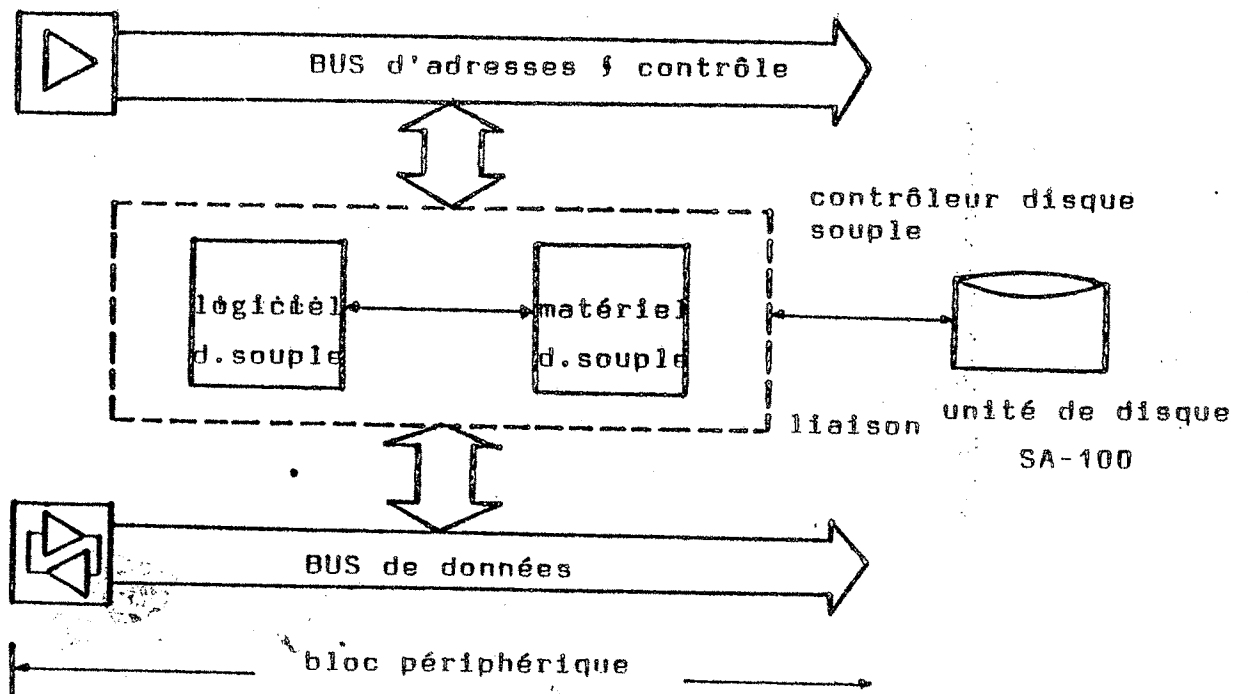


Figure 5.1 L'environnement du contrôleur disque souple.

### 2.1.1) Le circuit contrôleur: FD1771

Le contrôleur disque souple est réalisé à l'aide du circuit FD1771 de WESTERN DIGITAL (RT7-78). Ce contrôleur de disque souple est un boîtier standard de 40 pattes. Il est réalisé en MOS canal N et est compatible en entrée/sortie avec la technologie TTL.

Ce circuit contrôleur est programmable par le logiciel via un BUS bidirectionnel, à trois états, de 8bits qui permet l'accès à tous ses registres.

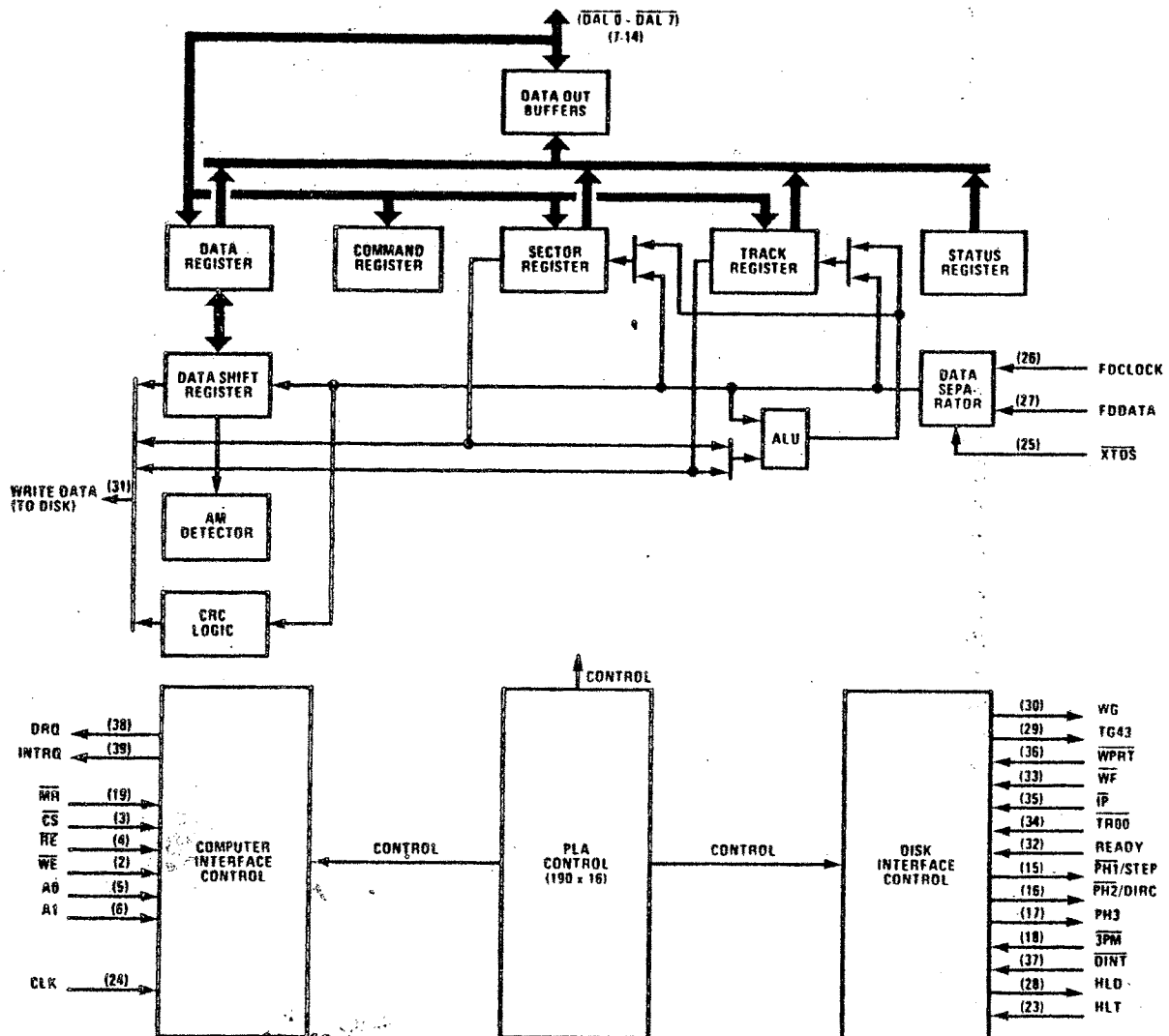


Figure 5.2 Synoptique du circuit contrôleur de disque souple.

2.1.2) L'interface P.central.

L'interface avec le P.central doit principalement éliminer les bruits qui peuvent circuler sur les différentes lignes d'adresses du BUS externe du P.central, en utilisant des circuits à seuil et hystérésis du type SN7414, et gérer la synchronisation en lecture et en écriture avec le P.central.

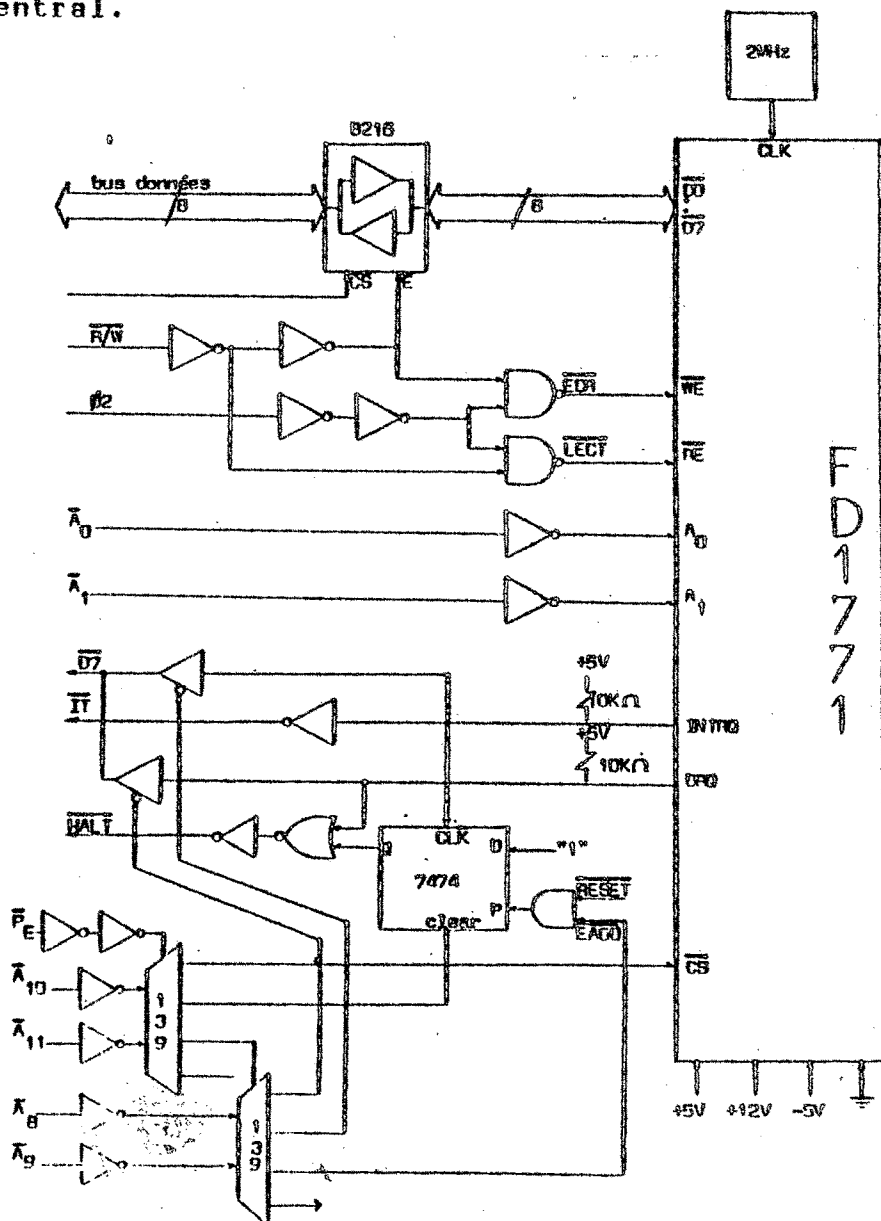


Figure 5.3 L'interface P.central.

L'interface est conçu pour travailler soit en mode programme (boucle d'attente des octets) soit en mode de transfert des données (accès direct à la mémoire).

### 2.1.2.1) L'horlogerie.

Le FD1771 admet deux fréquences différentes pour l'horloge: 1Mhz ou 2Mhz pour que le FD1771 travaille le plus rapidement possible, notamment pour le déplacement du bras portant la tête de lecture-écriture. L'horloge est réalisée par un boîtier SFF96871A1 qui aussi prévu pour générer les signaux propres au microprocesseur destinés à l'extension: on utilise une fréquence de 2Mhz.

### 2.1.2.2) Bus de données.

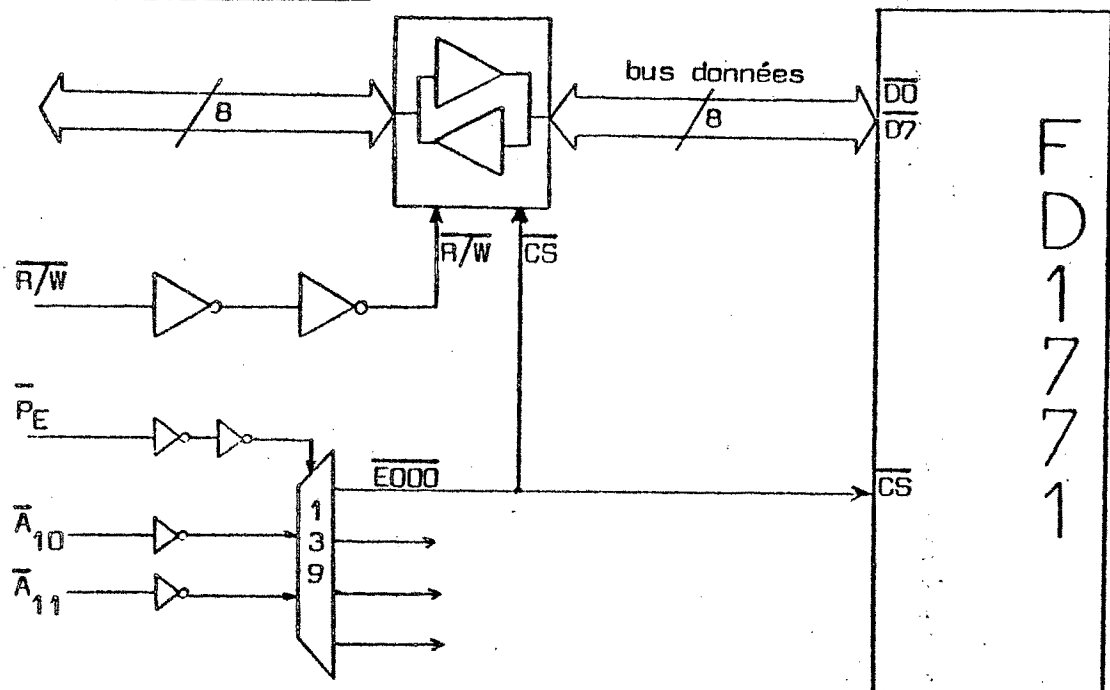


Figure 5.4 Le contrôle du BUS de données.

Les données sont complémentées à l'intérieur du FD1771. Le BUS de données est composé de 8 lignes qui passent par un amplificateur d'isolement: le circuit 8216 INTEL.

La carte contrôleur disque souple se trouve dans la page(E) de l'espace d'adressage du P.central.

Tant qu'elle n'a pas été sélectionnée par ce dernier, le circuit d'interface 8216 est bloqué et aucune information ne peut arriver au FD1771 ni en sortir. Quand le P.central veut travailler avec le contrôleur, il sélectionne l'adresse Exxx.

Dans ce cas le 8216 va amplifier les signaux dans un sens ou dans l'autre, d'après la valeur de la ligne de contrôle  $\overline{R/W}$ .

### 2.1.2.3) Sélection des registres internes du FD1771.

L'adressage des cinq registres du FD1771 se fait grâce aux deux bits de poids faible de l'adresse ( $A_0$  et  $A_1$ ) associés à l'ordre de lecture-écriture échantillonné sur la phase  $\emptyset 2$ . Ainsi la sélection d'un registre se fait sur  $\emptyset 2$ .

En effet pendant la phase  $\emptyset 1$  le P.central positionne les lignes de commande et d'adresses et, pendant la phase  $\emptyset 2$  le transfert des données peut s'effectuer. Ces données sont validées sur le BUS, en lecture et en écriture, à la fin de la phase  $\emptyset 2$ .

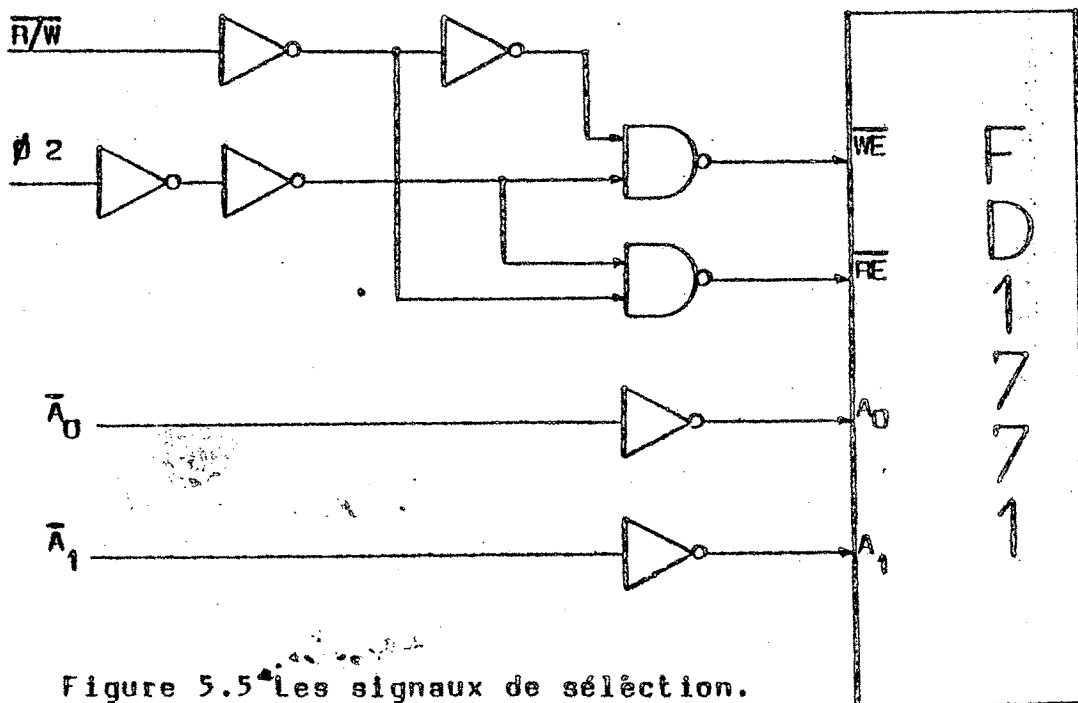


Figure 5.5 Les signaux de sélection.

#### 2.1.2.4) Test d'une commande (RT7-78): INTRQ

Le circuit contrôleur de disque souple génère une interruption vers le P.central quand une commande (seek, écriture / lecture d'une piste, écriture / lecture d'un secteur, ...) a été terminée.

Cette interruption permet au P.central, par la lecture du registre d'état du contrôleur, de savoir si la commande a été bien dérivée.

La présence de l'interruption (INTRQ) est détectée par le circuit contrôleur d'interruption du P.central.

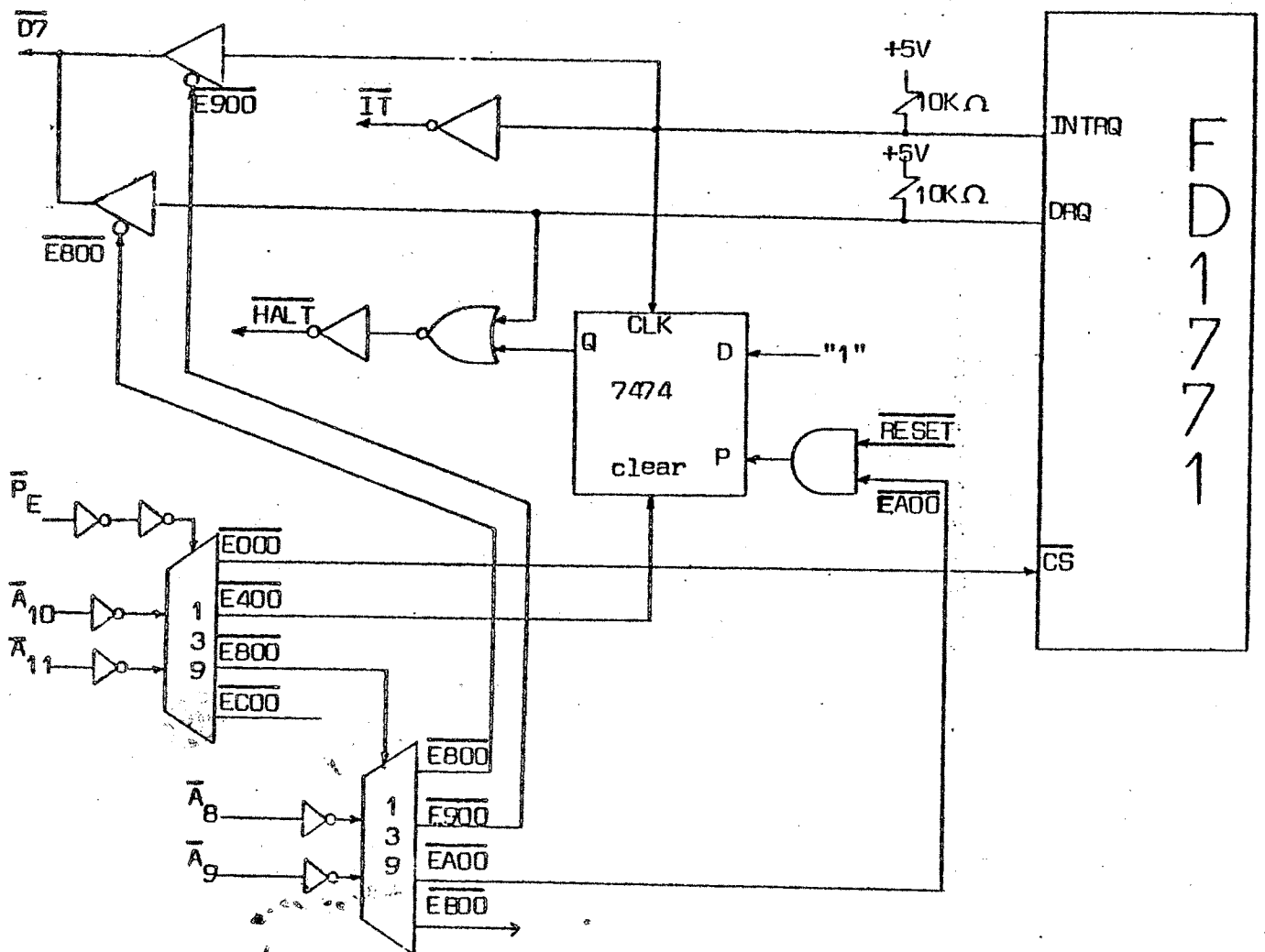


Figure 5.6 Le contrôle des interruptions.



On peut aussi attendre la présence de l'interruption du circuit contrôleur (INTRQ) par une boucle d'attente logicielle en faisant un chargement à une adresse donnée.

```
UP LDA A,=XE9XX
   BNE UP
```

#### 2.1.2.5) Techniques de transfert.

##### - Transfert des données par boucle logicielle.

De la même manière que l'interruption INTRQ, on peut attendre le signal DRQ (Data Request) (R17-78) par boucle logicielle:

```
UPI LDA A,=XE8xx
   BNE UPI
```

##### - Transfert des données en STOP-START: HALT.

Cette technique de transfert est appliquée en utilisant une simple bascule (figure 5.6) qui synchronise la lecture ou l'écriture des données provenant du circuit contrôleur FDI771 avec le P.central ou viceversa.

La ligne HALT ou la ligne TSC du P.central peuvent être utilisées pour ce type de transfert, tout en dépendant du temps autorisé d'arrêt du P.central. Dans notre cas, la ligne TSC ne peut pas être utilisée car le contrôleur disque souple envoie ou reçoit les données en 32 micro-secondes: le temps maximum d'arrêt du P.central, en utilisant la ligne TSC, est de 4.5 micro-secondes (c.f.annexe I).

Le principe de fonctionnement est le suivant: le P.central est en fonctionnement normal quand la bascule est désactivée ( $Q=0$ ) ( $\overline{HALT}=1$ ) et lorsqu'elle est activée le signal  $\overline{HALT}$  suit l'évolution du DRQ (figure 5.6).

Le P.central doit récupérer ou écrire l'octet d'information avant que le signal DRQ repasse à "0" ce qui le bloque à nouveau.

On voit qu'il est possible de se servir de ce système pour récupérer ou envoyer des octets au registre de données (DR) du circuit contrôleur FD1771 lors d'un transfert. La fin du transfert est déterminée par la sortie INTRQ du FD1771.

### 2.1.3) L'interface unité disque souple.

Cet interface, composé de plusieurs circuits, a la finalité d'amplifier les signaux de lecture-écriture, les signaux de contrôle vers l'unité de disque souple et vers le circuit FD1771. Ces signaux peuvent être classés de la manière suivante:

- commandes du bras,
- positionnement de la tête,
- transfert des données.

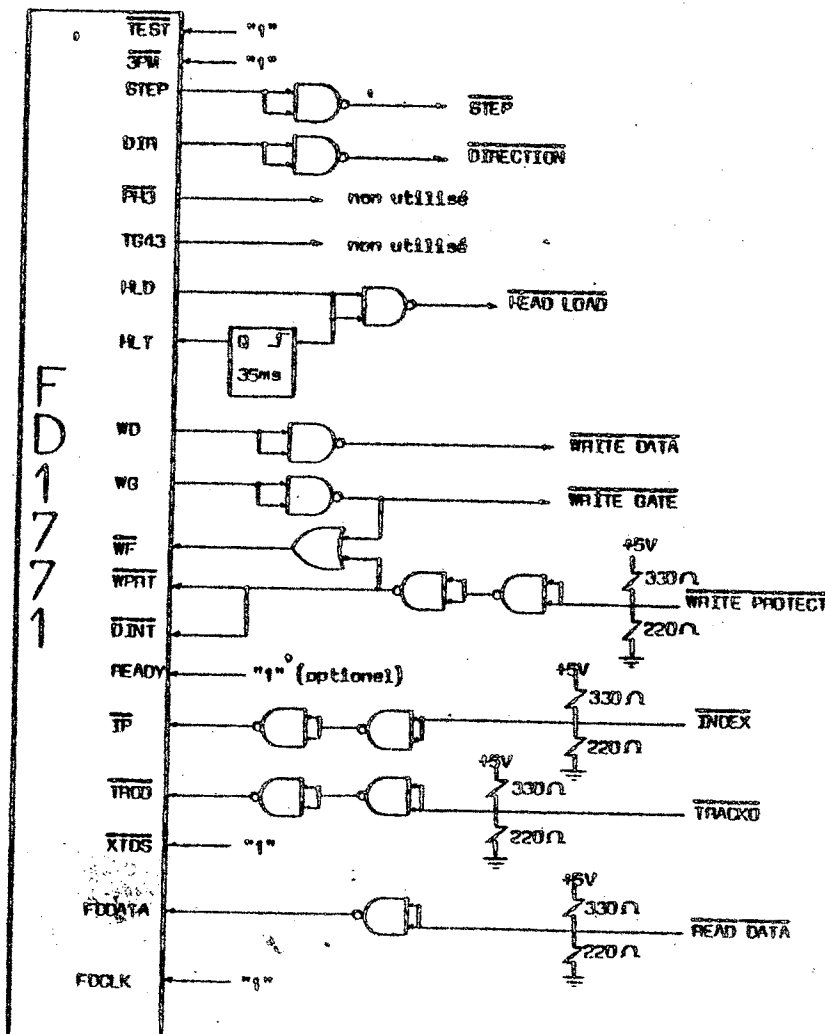


Figure 5.7 Interface vers l'unité de disque souple.

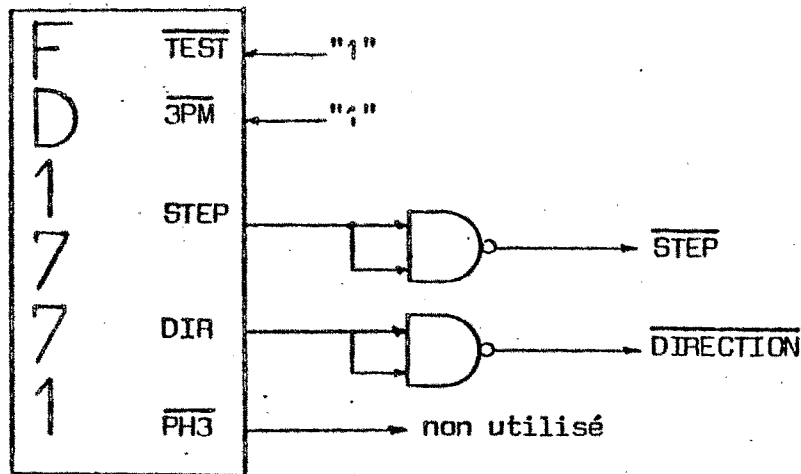


Figure 5.8 Commandes du bras.

TEST est relié au 5V. Ainsi la vitesse du bras est programmée par les 2 bits de poids faible de la commande (r1 et r0). L'entrée 3PM étant à niveau haut le FD1771 commande le moteur pas à pas seulement avec les deux lignes STEP et DIRECTION. La troisième ligne PH3 est donc inutilisée.

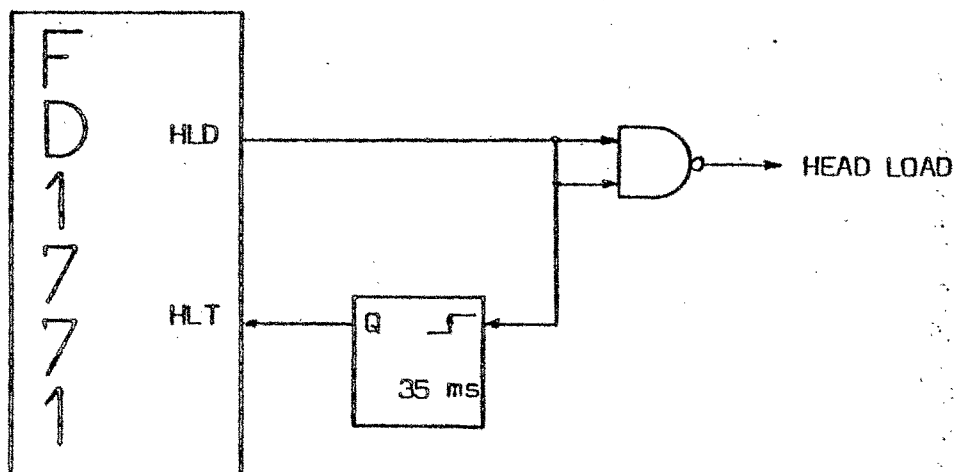


Figure 5.9 Positionnement de la tête.

Quand le FD1771 envoie l'ordre de poser, sur la disquette, la tête de lecture-écriture (HLD) il faut attendre avant de tester l'entrée HLT. Quand HLT passe à "1" le FD1771 estime que la tête est effectivement en contact avec le disque et peut donc lancer un ordre de transfert.

Le temps de positionnement de la tête est de 35 ms. Il faut donc placer un retard de 35 ms entre le signal HLD et le signal HLI (RT6-78).

- réception / transfert des données entre le contrôleur et l'unité de disque souple.

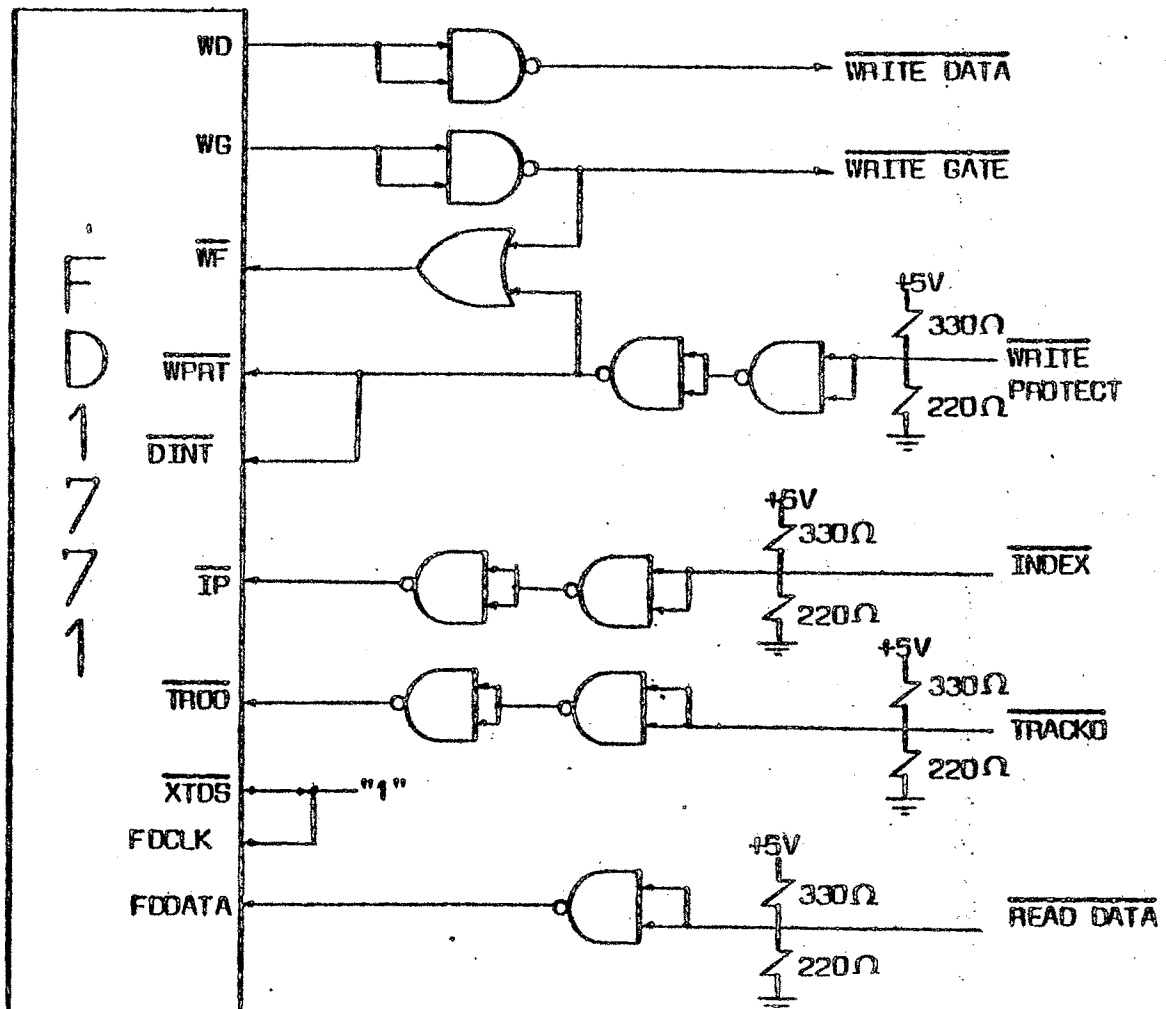


Figure 5.10 Réception/transfert des données entre le contrôleur et l'unité disque souple.

L'entrée XTDS (RT6-78) est normalement au niveau logique haut, le FD1771 reçoit donc sur la ligne FDDATA les bits de données mélangés avec l'horloge. La séparation des bits de données et d'horloge se fait à l'intérieur du FD1771, l'entrée FDCLK étant au niveau logique haut (figure 5.10).

L'entrée  $\overline{WF}$  est active quand on veut écrire (WG active) et quand l'unité de disque souple signale une écriture protégée ( $\overline{WPRT}$  actif). De plus, le signal d'écriture protégée positionne l'entrée  $\overline{DINT}$  qui est testée avant le lancement de l'écriture d'une piste. Si  $\overline{DINT}$  est actif (niveau 0) l'opération n'est pas exécutée.

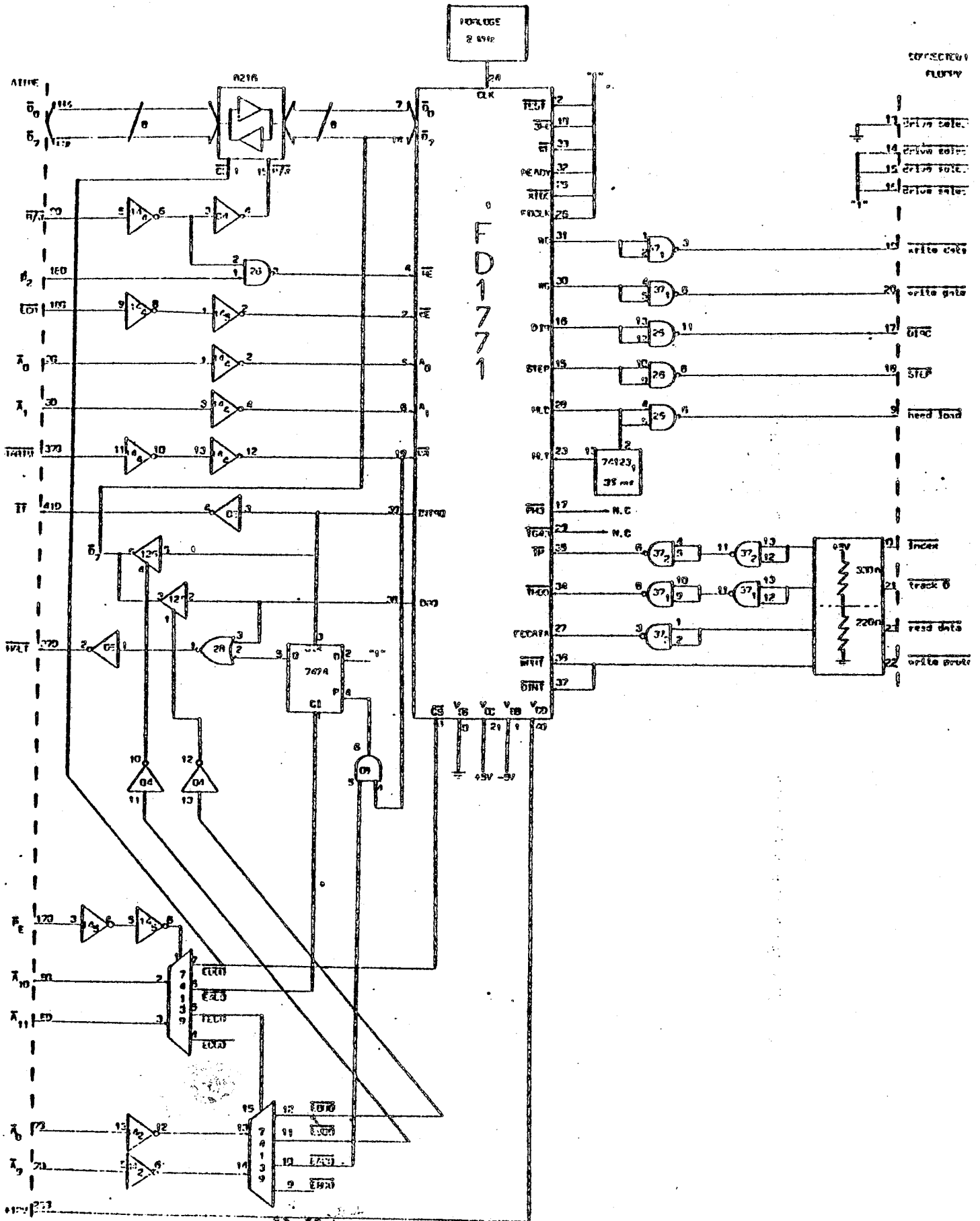


Figure 5.17 Câblage du contrôleur de disque souple.

## 2.2) Le logiciel disque souple.

Le logiciel disque souple est représenté de la manière suivante:

$\langle \text{logiciel d.souple} \rangle ::= \langle \text{L.base} \rangle \langle \text{L.d.souple} \rangle$

où:

$\langle \text{L.base} \rangle$ : logiciel de base du P.central(moniteur)

$\langle \text{L.d.souple} \rangle$ : le logiciel de contrôle du matériel disque souple.

Le logiciel d.souple est une couche logicielle située au dessus de la couche système représentée par le L.base. Celle ci implique que le logiciel d.souple n'est pas indépendant; il doit utiliser des fonctions, par exemple les entrées/sorties, du L.base.

Le logiciel d.souple est conçu comme un moniteur qui a la fonction de dialoguer avec l'opérateur du P.central en utilisant une série de commandes.



2.2.1) Caractéristiques du logiciel d.souple: Moniteur Floppy.

- Lecture et écriture des octets sur le disque.

La vitesse de rotation du disque est de 360 tours par minute, ce qui correspond à une révolution toutes les 166,6 ms. La piste entière est donc lue ou écrite en 166,6 ms.

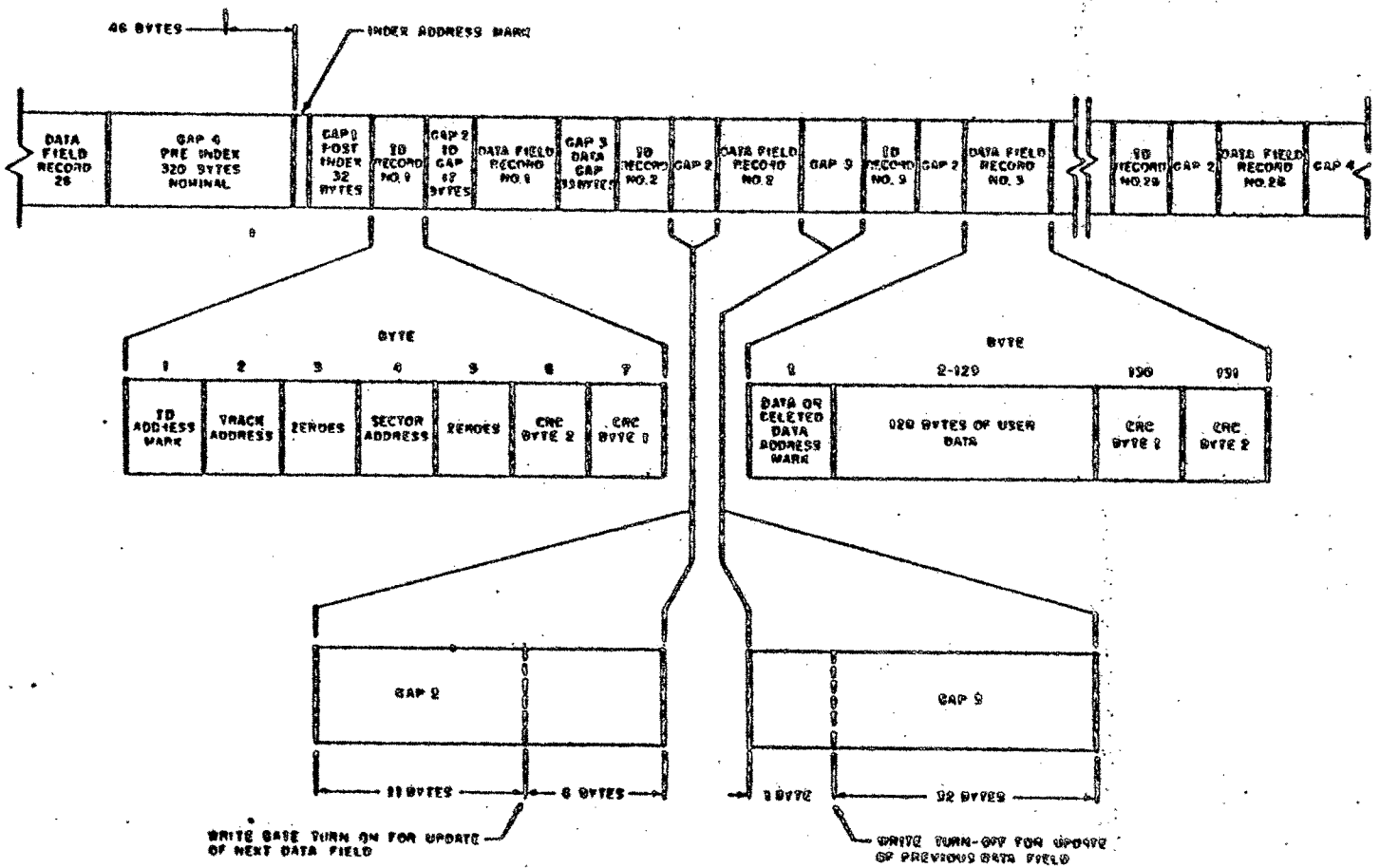


Figure 5.12 Format d'une piste.

La piste est formée de 26 secteurs (RT6-78). Avant le premier secteur il y a un espace de 46 octets et après le dernier secteur un espace de 274 octets, ce qui fait 320 octets inutilisés entre les secteurs 26 et 1.

Chaque secteur est composé de:

- un champ d'identification: 7 octets
- une zone de données: 131 octets
- \* marque d'adresse: 1 octet
- \* données: 128 octets
- \* CRC: 2 octets
- un espace: 50 octets

Soit un total de 188 octets par secteur. Le nombre total d'octets par piste est donc:

$$320 + 26 * 188 = 5208 \text{ octets}$$

Le temps de lecture ou d'écriture d'un octet est donc de:

$$166.6 / 5208 = 0.032 \text{ ms} = 32 \text{ micro-sec}$$

Ceci signifie que le contrôleur de disque souple envoie un DRQ toutes les 32 micro-sec. Le P.central, quand il reçoit le signal DRQ, doit vider ou remplir le registre DR dans le délai de 32 micro-sec, sinon un octet sera perdu.

Le programme Moniteur Floppy doit donc avoir une boucle de récupération des octets en cas de lecture ou, d'envoi des octets en cas d'écriture, dont le temps d'exécution soit inférieur à 32 micro-sec. Le cycle de base du P.central étant d'une 1 micro-sec, ces boucles peuvent donc prendre au maximum M cycles, où  $M = 32$ . Le temps correspondant aux incertitudes sur la vitesse de rotation du moteur, au démarrage et à l'arrêt du microprocesseur oblige à réduire ce nombre d'environ 5 cycles (5 micro-sec).

-Solution utilisant le pointeur de pile.

\* Boucle de lecture.

```

UP1 STA B,ITH pour prendre en compte une éventuelle IF.
UP2 LDA A,DRQ
    BNE UP2 attente DRQ
    LDA A,DR lecture contenu du registre DR
    PSHA rangement en mémoire
    BRA UP1
  
```

Toutes ces instructions prennent 4 cycles ce qui fait un total de 24 cycles, soit 24 micro-sec, donc il n'y a pas de risque de perdre un octet.

Cependant cette solution n'a pas été retenue car l'instruction PSHA a pour effet de ranger le contenu de l'accumulateur A en mémoire à l'adresse donnée par le pointeur de pile SP avant de décrémenter ce pointeur de 1.

Les informations lues sur le disque sont donc rangées en mémoire dans le sens décroissant des adresses. Un bloc de données ainsi chargé doit être "remis à l'endroit" c'est-à-dire réécrit dans l'ordre croissant des adresses avant d'être utilisé. C'est pour cette raison que cette solution a été abandonnée.

- solution utilisant le registre Index.

\* Boucle de lecture.

```

UPI STA B,ITH .. activation circuit d'IT
UP2 LDA A,DRQ
    BNE UP2
    LDA A,DR
    STA A,O(X)
    INCX
    BRA UPI

```

Dans cette solution, l'instruction PSHA a été remplacée par deux instructions: une pour ranger l'accumulateur A à l'adresse contenue dans le registre index(X) et une seconde pour incrémenter ce registre. Le rangement à adressage indexé prend 6 cycles et l'incrémentation de l'index 4 cycles, ce qui fait un total de 30 cycles. La boucle est donc exécutée en 30 micro-sec.

\* Boucle d'écriture.

```

WUPI STA B,ITH
WUP2 LDA A,DRQ
    BNE WUP2
    LDA a,O(X)
    STA A,DR
    INCX
    BRA WUPI

```

Toutes ces instructions prennent 4 cycles sauf le chargement à adressage indexé qui en prend 5. La boucle d'écriture prend donc 29 cycles, soit 29 micro-sec pour son exécution.

L'exécution du programme de lecture ou d'écriture continue jusqu'à ce qu'une interruption soit envoyée par le contrôleur de disque souple qui indique au P.central que la commande est terminée.

Cette solution a été retenue, car cette fois, le rangement en mémoire se fait dans l'ordre croissant des adresses et aussi parce que nous avons prévu l'utilisation d'un microprocesseur rapide (MC68800) avec un cycle de 500ns; ceci ramène la durée des boucles à 15 micro-sec.

Il faut remarquer que le contrôleur dispose de facilités matérielles pour effectuer des transferts de données en accès direct à la mémoire si l'on désire utiliser ce mode de transfert.

### 2.2.2) Le moniteur d'essais du disque souple.

Pour réaliser des opérations de type lecture, écriture ou formattage, le moniteur doit reconnaître un certain nombre de commandes venant d'un périphérique (télé-imprimeur, visu).

#### - lecture d'une piste.

La syntaxe est la suivante:

> L <V><adresse><piste>(RC)

Cette commande a pour effet de lire la piste dont le numéro est donné par <piste>. Le résultat de la lecture (gaps, marques d'adresses, CRC, identification, données) est rangé en mémoire à partir de l'adresse donnée par les deux octets de <adresse>.

De plus le résultat de la lecture est imprimé sur le terminal si <V>=V.

<piste>::=(00,4C).. nombre de pistes entre 00 et 4C.

#### - lecture d'un secteur.

> L <V><adresse><piste> <secteur>(RC)

Cette commande lit le <secteur> de la <piste>, c'est-à-dire les données seulement, et les range en mémoire à partir de l'adresse <adresse>.

De plus, le résultat de la lecture est imprimé sur le terminal si <V>=V.

<piste>::= (00,4C);

<secteur>::= (00,1A)... nombre de secteurs.

Il faut remarquer que le logiciel d'essais du floppy utilise les fonctions de base du moniteur système en plus des fonctions de base du contrôleur de disque souple.

- lecture de plusieurs secteurs.

> L <V><adresse><piste> <secteur1> <secteur2>

Cette commande lit les données contenues dans les secteurs compris entre <secteur1> et <secteur2>, bornes incluses, de la <piste> et range le résultat à partir de l'adresse<adresse>.

<secteur1> ::= (00,1A);

<secteur2> ::= (00,1A);

Ecriture d'un secteur.

> W <adresse><piste><secteur>(RC)

Cette commande lit les informations qui se trouvent en mémoire à partir de l'adresse<adresse> et les range dans le <secteur> de la <piste>.

- Ecriture plusieurs secteurs.

> W <adresse><piste><secteur1> <secteur2>

Cette commande lit les informations rangées en mémoire à partir de la <adresse> et les écrit sur le disque dans les secteurs compris entre <secteur1> et <secteur2>, bornes incluses, de la <piste>.

- Formatage du disque.

Cette commande(>F) remplit une zone de mémoire avec le profil d'une piste complète: gaps, marques d'adresses, CRC, champs d'identification, champs de données. Cette zone est ensuite écrite sur une piste complète. Ceci est recommencé 77 fois pour formater les 77 pistes du disque.

- Retour au logiciel de base de la maquette: moniteur système.

Cette commande(>R) a pour effet de ramener la tête de lecture-écriture sur la piste 0, puis le contrôle est donné au moniteur système (SP-6800).

### 3.) EXTENSION DU CONTROLEUR: LE PROCESSEUR FLOPPY.

L'extention que nous pouvons apporter à ce problème est de cabler un microprocesseur MC68800 sur la carte du contrôleur afin de réaliser un processeur "floppy". Ce processeur se charge alors de transfert des données entre la mémoire et le disque, et inversement. Il fait ce travail sur ordre du P.central dans le cadre d'une machine multiprocesseurs; dans ce cas le processeur floppy est esclave du P.central.

#### 3.1) Extention matérielle.

Nous envisageons deux modes de travail, il faut donc placer un interrupteur sur le tableau de commandes. Cet interrupteur sera positionné par l'opérateur qui pourra ainsi choisir les modes de travail. Ce choix sera mémorisé par une bascule positionnée par l'interrupteur.



- mode superviseur.

Dans ce mode de travail, on veut permettre au processeur floppy de dialoguer avec un opérateur. Il faut donc placer sur la carte des facilités matérielles pour effectuer ce dialogue:

- \* une ROM contenant le même logiciel de base que le P.central (moniteur),
- \* une unité de contrôle périphérique(ACIA) assurant la connexion avec un terminal.

- mode floppy.

Dans ce mode de travail, le processeur floppy est esclave du P.central et il attend donc les ordres de lecture ou d'écriture venant du P.central. Il faut donc permettre un dialogue entre les deux processeurs.

Ce dialogue pourra se faire à travers une zone de mémoire particulière, réservée à cet effet. Cette zone de mémoire est constituée de

- \* une position mémoire pour donner l'ordre au processeur floppy de commencer son travail.
- \* une zone mémoire contenant la commande à exécuter.

Le processeur floppy va donc tester continuellement cette position mémoire. Une fois activé, le processeur floppy donne le contrôle à la partie analyse des commandes de son moniteur; cette partie ira chercher la commande à exécuter dans la zone de mémoire partagée, où elle a été rangée précédemment par le P.central.

Le processeur floppy envoie une interruption au P.central pour lui indiquer que l'exécution de la commande est terminée.

- accès à la mémoire partagée.

Les deux processeurs de la maquette partagent une mémoire commune. Il faut donc assurer une mutuelle exclusion de l'accès à cette mémoire; pour cela les processeurs travaillent en phases complémentées (allocation des ressources la plus simple). C'est-à-dire que le P.central lit ou écrit en mémoire sur la phase Ø2 tandis que le processeur floppy a accès à cette même mémoire sur la phase Ø1. Il faut donc que les deux processeurs aient une horloge commune.

Cette horloge doit se trouver sur la carte du P.central et est envoyée au processeur floppy à travers le BUS de contrôle après avoir été amplifiée par un amplificateur spécial d'horloge qui retarde le signal de 50 ns.

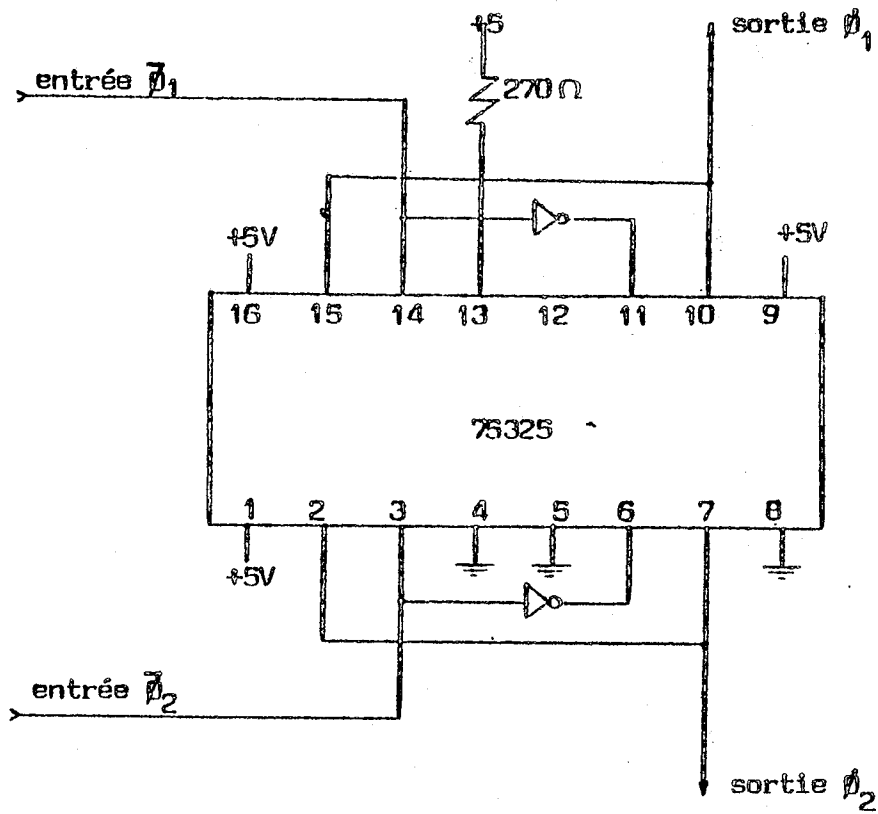


Figure 5.13 Schéma du driver.

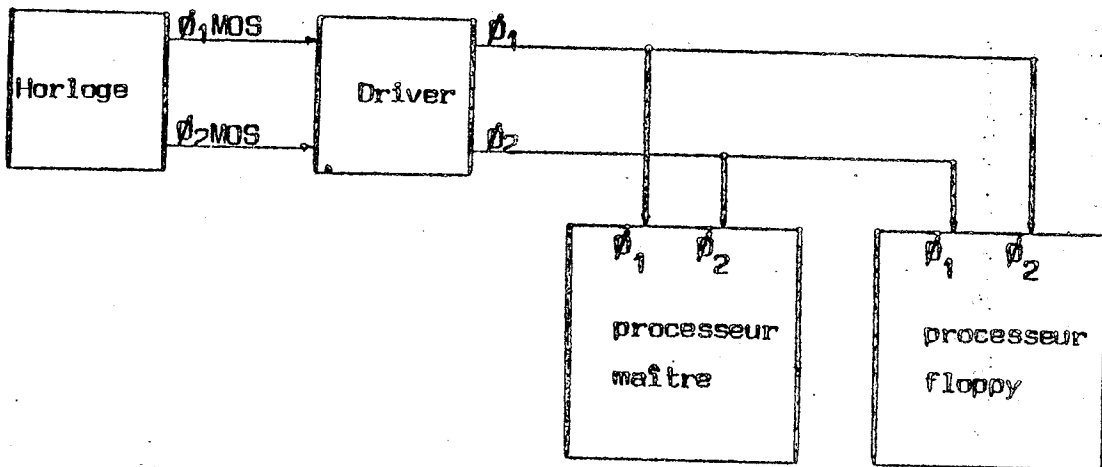


Figure 5.14 Le mécanisme à phases complémentées.

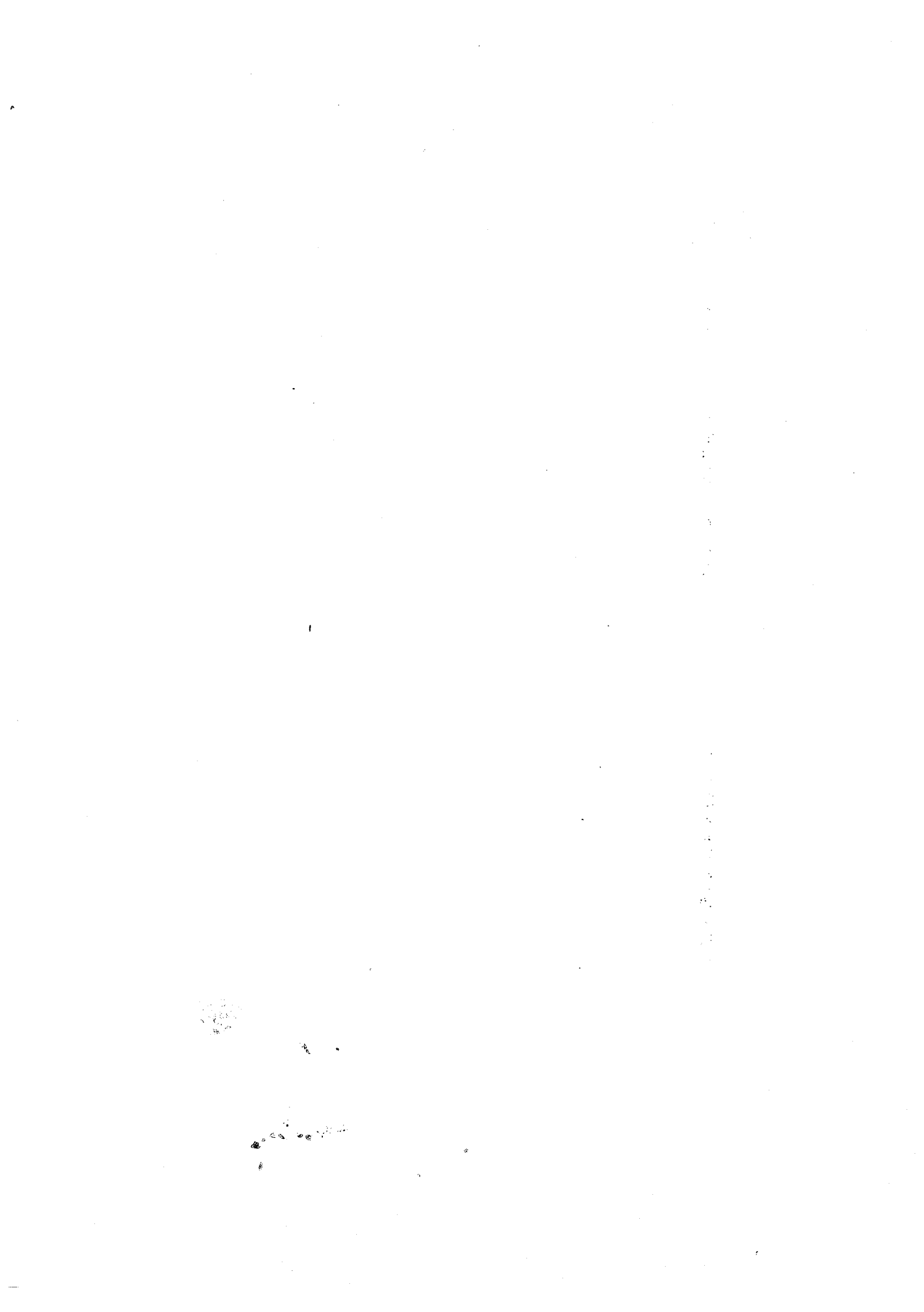
### 3.2) Extention logicielle.

Nous avons également augmenté la puissance du programme Moniteur floppy en lui ajoutant une commande qui est un embryon de gestion de fichiers:

```
>G <nom fichier><adresse><E>(RC)
```

<nom fichier> est le nom d'un fichier qui se trouve sur la disquette (catalogue). Cette commande a pour effet de charger ce fichier en mémoire vive à partir de l'adresse <adresse>. L'exécution de ce fichier est lancée si <E>=E.

Le catalogue des fichiers a la même format que celui du catalogue MDOS MOTOROLA (RT8-78).



## CHAPITRE VI

### UN PROCESSEUR EN TEMPS REEL APPLIQUE AU DEVELOPPEMENT LOGICIEL ET MATERIEL DES APPLICATIONS MICROPROCESSEURS: ANALYD

- 1.) INTRODUCTION.
- 2.) GENERALITES
- 3.) LE PROBLEME
- 4.) LA CONCEPTION D'ANALYD
- 5.) LE SYSTEME ANAMAD
- 6.) L'EXTENTION D'ANALYD

## CHAPITRE VI

### UN PROCESSEUR EN TEMPS REEL APPLIQUE AU DEVELOPPEMENT LOGICIEL ET MATERIEL DES APPLICATIONS MICROPROCESSEURS: ANALYD

#### 1.) INTRODUCTION

Le principal problème dans le développement d'une application microprocesseur est l'analyse en temps réel pour sa mise au point. L'ensemble de la mise au point en temps réel dépend des facteurs suivants:

- les coûts des capteurs et des moyens d'enregistrement,
- la rapidité d'acquisition de l'information; elle dépend de la fréquence d'arrivée des événements.
- la vitesse de traitement de l'information (ex: visualisation).

Ce chapitre sera donc consacré:

- à l'étude et à la conception du système ANALYD,
  - à l'application de ce système: ANAMAD
- et
- à l'extension du système ANALYD.





### 3.) LE PROBLEME.

Il existe sur le marché des outils d'aide au concepteur comme :

- des analyseurs digitaux (série, parallèle, etc),
- des équipements plus ou moins complexes avec de multiples fonctions souvent superflues qui ne garantissent pas que le système microprocesseur soumis au test, marche correctement.

Notre but sera la conception d'un système capable d'offrir, d'une part, les mêmes fonctions que les analyseurs digitaux conventionnels :

- observer l'exécution d'un processus,
- positionner des points d'arrêt matériels sur :

- \* les adresses,
- \* les données,
- \* les signaux de contrôle.

et d'y ajouter des fonctions, que nous considérons comme indispensables, comme :

- \* l'enregistrement de l'historique du processus,
- \* impression de cet historique pour avoir un film des événements enregistrés.
- \* le positionnement des points d'arrêt définis par des prédicats formés sur des combinaisons de données, de signaux de contrôle et d'adresses.
- \* la faculté d'être mono ou multi profils.

#### 4.) LA CONCEPTION D'ANALYD.

Le système est conçu avec la même philosophie que celle utilisée dans MICROD, c'est-à-dire que le système sera décomposé en deux blocs:

- le bloc processeur,
- le bloc périphérique: module spécialisé.

La figure 6.2, schématise d'une manière fonctionnelle le système ANALYD.

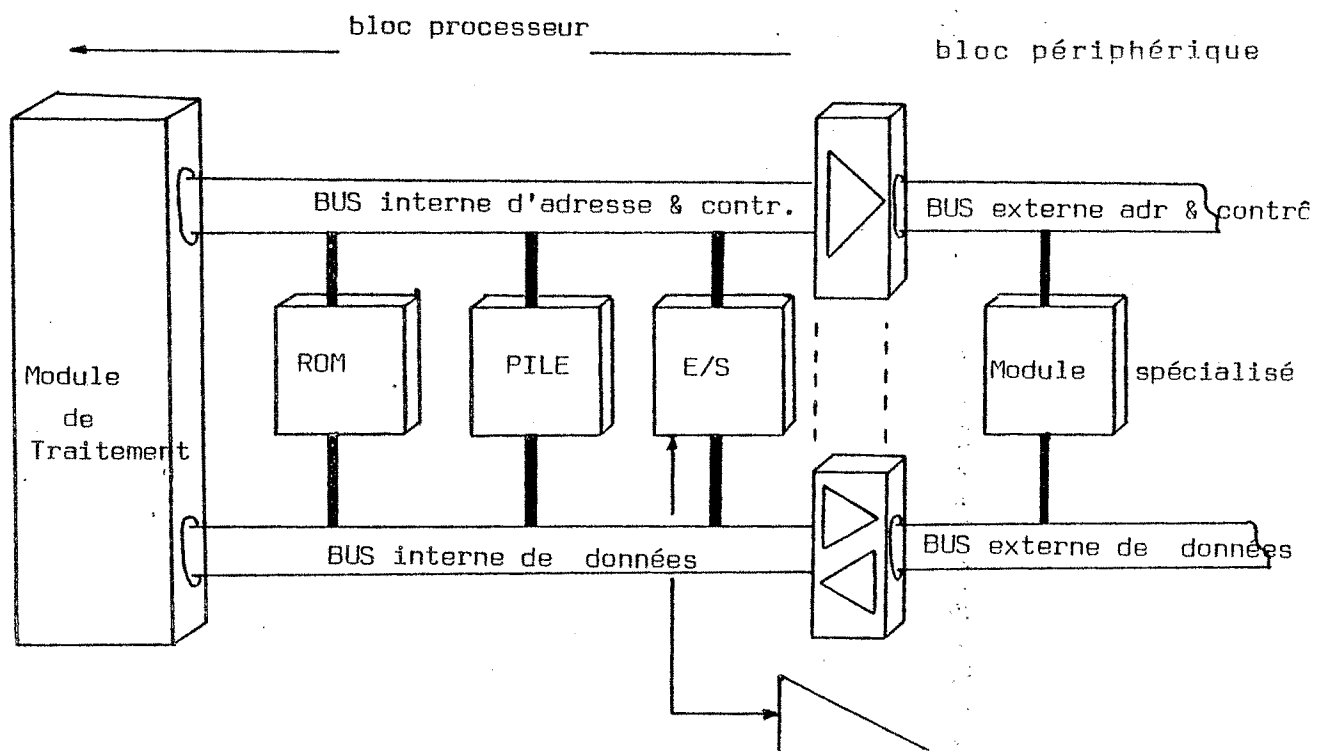


Figure 6.2 L'architecture du système ANALYD.

#### 4.1) Le bloc processeur.

le bloc processeur est constitué des éléments de base suivants :

- le microprocesseur,
- le logiciel système (moniteur),
- la mémoire de travail de ce logiciel,
- le logiciel spécialisé, qui partage avec le logiciel système la mémoire de travail,
- l'interface d'entrée/sortie avec un téléimprimeur.

#### 4.2) Le bloc périphérique.

Ce bloc contient d'une part le module interface (c.f. III) et d'autre part le matériel propre à l'application.

Ce bloc a la fonction de définir la nature de l'application, dans notre cas espionner le comportement d'un processus.

Cet espionnage se traduit par le traitement et le stockage de l'information provenant de l'application. Ce bloc est divisé en deux parties:

- une partie contrôle,
- une partie enregistrement.

##### 4.2.1) La partie contrôle.

Nous allons analyser cette partie d'abord comme un système mono-profil puis comme un système multi-profil.

Pour mieux analyser cette partie, on la décomposera en:

- le mécanisme de sélection et de détection du profil,
- le mécanisme de contrôle, proprement dit du système.

#### 4.2.1.1) Le système mono-profil.

Ce type de système a la caractéristique de détecter un seul profil. La partie contrôle de ce système sera décrite ci-après.

##### - Le mécanisme de sélection et de détection du profil.

Le profil (PROFIL) est défini comme une configuration de (n) bits que l'utilisateur veut détecter dans le flot de données provenant de l'application (adresses, données, contrôle, autres configurations).

Ce profil est placé par l'utilisateur dans des registres. Le nombre de ces registres dépend de la configuration du système (nombre de bits) (figure 6.3).

##### \* Le mécanisme de sélection: SELECT.

Ce mécanisme (SELECT) sert à choisir le profil entre une ou plusieurs voies de données de l'application (figure 6.3).

SELECT est composé de (k) voies de n bits chacune. Le nombre de bits (k) dépend de la taille du profil; chaque voie représente une source de données de l'application où on peut rechercher un profil quelconque.

La complexité de ce mécanisme est fonction du degré de souplesse que l'on veut donner au système; c'est-à-dire, la possibilité de choisir un profil parmi les différentes voies. Ces voies peuvent représenter soit des données, des adresses, soit des signaux de contrôle ou soit une combinaison de ceux-ci.

SELECT dépend du PROFIL. Par exemple, si l'on veut choisir une adresse comme profil on sélectionne la voie correspondant aux adresses dans SELECT. De la même manière si on veut comme profil sélectionner les bits les plus forts d'une adresse puis les bits les plus forts d'une donnée, il faudra choisir, dans SELECT, les voies correspondant aux données et aux adresses.

\* Le mécanisme de détection.

Ce mécanisme est chargé:

- d'initialiser l'enregistrement,
- de contrôler l'écriture dans la mémoire après ou avant la détection du profil (RECONNU. figure 6.3),
- de générer une interruption vers le processeur qui effectue la gestion de la mémoire,
- de contrôler la sélection du PROFIL: génération des validations et des masquages de voies,
- de générer l'adressage de la mémoire pour l'enregistrement des événements.

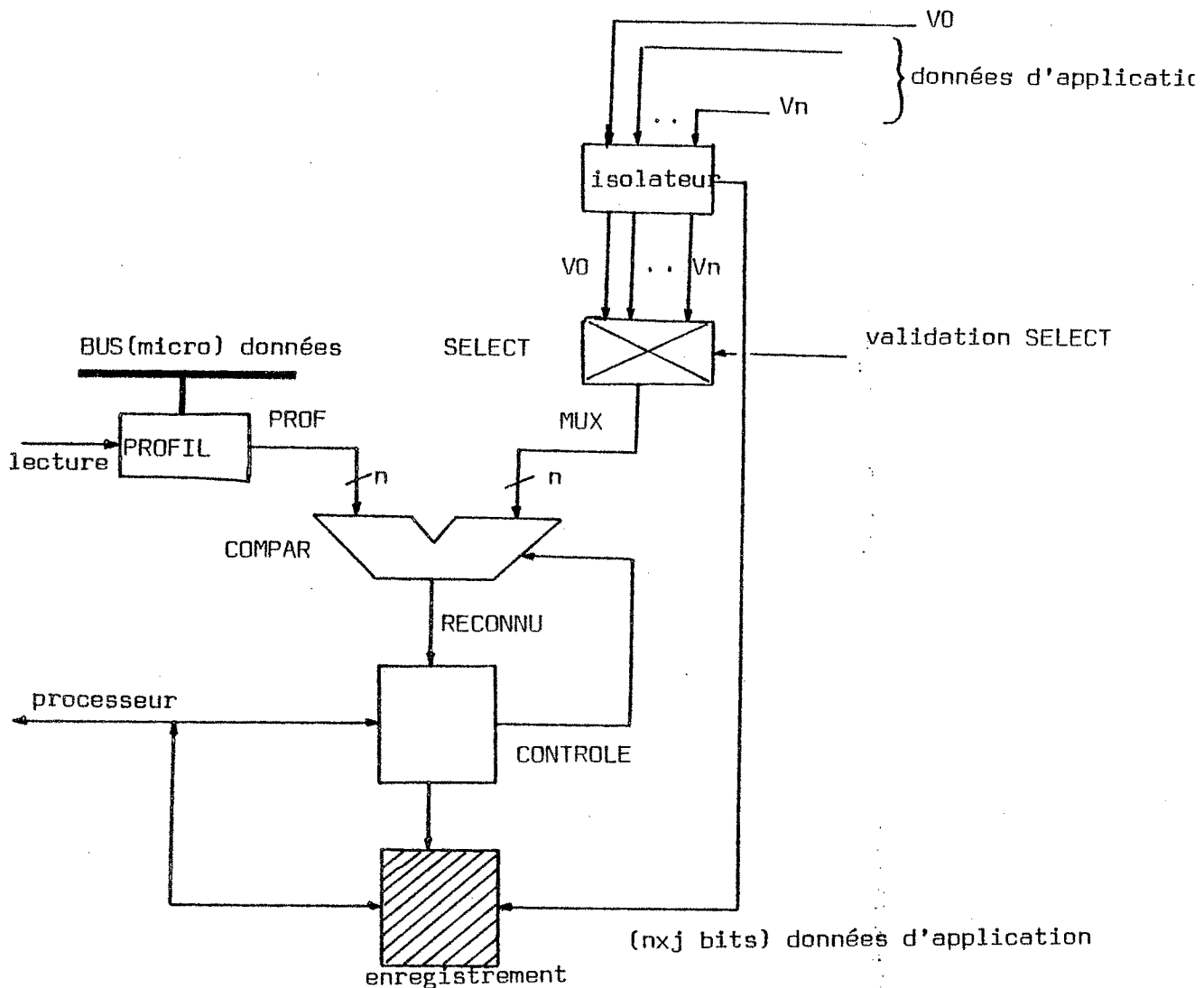


Figure 6.3 Schéma de ANALYD comme système mono-profil.

- Mode de travail.

La procédure ci-dessous donne l'algorithme correspondant au système mono-profil.

PROCEDURE mono-profil,  
DEBUT CO; exemple logiciel d'enregistrement d'ANAMAD;  
 chargement profil dans REGISTRE;  
 sélection type de profil;  
CO; sélection de la(ou les) voie(s) correspondante(s);  
 validation du comparateur; CO COMPAR;  
 i::=0;  
TANT QUE (REGISTRE)  $\neq$  (SELECT) FAIRE  
 M(0,i)::= enregistrement des données;  
DEBUT  
 arrêt compteur de cycles;  
TANT QUE i+1  $\neq$  256 FAIRE  
 M(1,i+1)::= enregistrement;  
 Lecture compteur de cycles;  
CO pour retrouver les 256 événements antérieurs (0,i+1)  
 (0,i+2) ,... et les 256 événements postérieurs (1,i-1) ,  
 (1,i),  
 ...  
FIN  
FIN;

#### 4.2.1.2) Le système multi-profils.

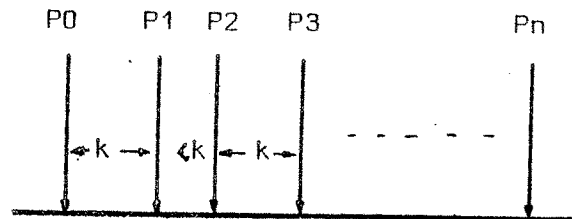
Il utilise le même principe que le système mono-profil, le REGISTRE est seulement remplacé par un mécanisme qui permet de comparer plusieurs profils.

Dans cette configuration multi-profils, l'utilisateur aura à sa disposition deux possibilités pour tester l'arrivée des profils:

- d'une manière ordonnée,
- ou
- d'une manière aléatoire.

##### - Ordonnée.

Les profils sont reconnus dans un ordre pré-défini; l'utilisateur doit donc connaître, par avance, l'ordre d'arrivée des événements.



k = nombre max d'événements enregistrés

Figure 6.4 Les profils ordonnés.



Ces profils sont placés, dans la mémoire de profils(MEMPROF), d'une manière ordonnée.

Le mécanisme de contrôle fonctionne de la manière suivante: une fois détecté le premier profil, le profil suivant est placé dans PROF et le comparateur (COMPAR) est validé, jusqu'à la détection d'un nouveau profil.

Ceci est fait cycliquement jusqu'au dernier profil stocké dans MEMPROF (figure 6.3). A chaque fois qu'un profil est détecté, on lit et on sauvegarde le compteur d'enregistrement pour la gestion ultérieure des profils.

Les enregistrements des événements propres aux différents profils sont faits en continu et la taille de ces enregistrements est donnée par le compteur d'enregistrements.

- Aléatoire.

Les profils sont enregistrés dans un ordre quelconque; dans ce cas, il faudra donc réaliser le stockage des profils par un stockage associatif.

L'utilisation de mémoires associatives permet de stocker les profils d'une manière aléatoire. Ces mémoires ont la fonction de comparaison interne.

Grâce à ces mémoires associatives, la détection du profil dans les données de l'application se fait plus rapidement dans cette configuration que dans la précédente.

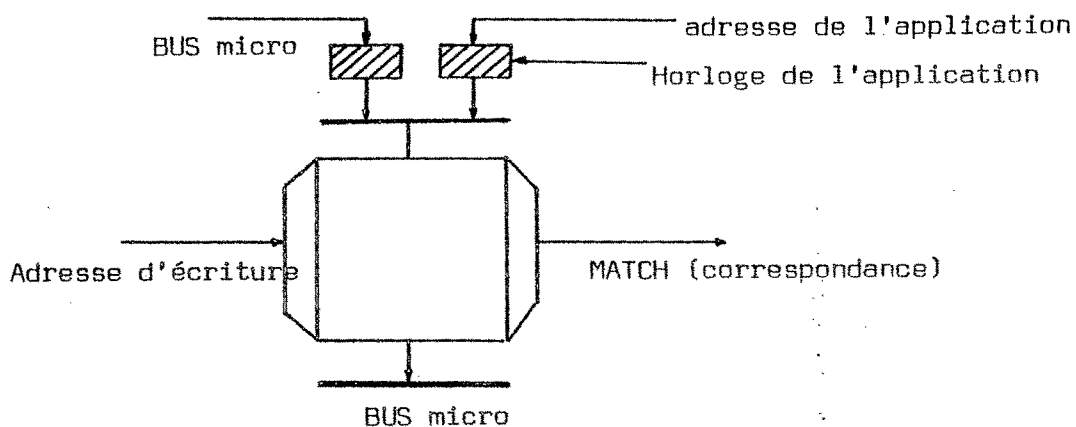


Figure 6.5 Configuration de la mémoire de PROFILS.

la figure 6.6, montre l'organisation d'ANALYD dans le cas d'un système multi-profil.

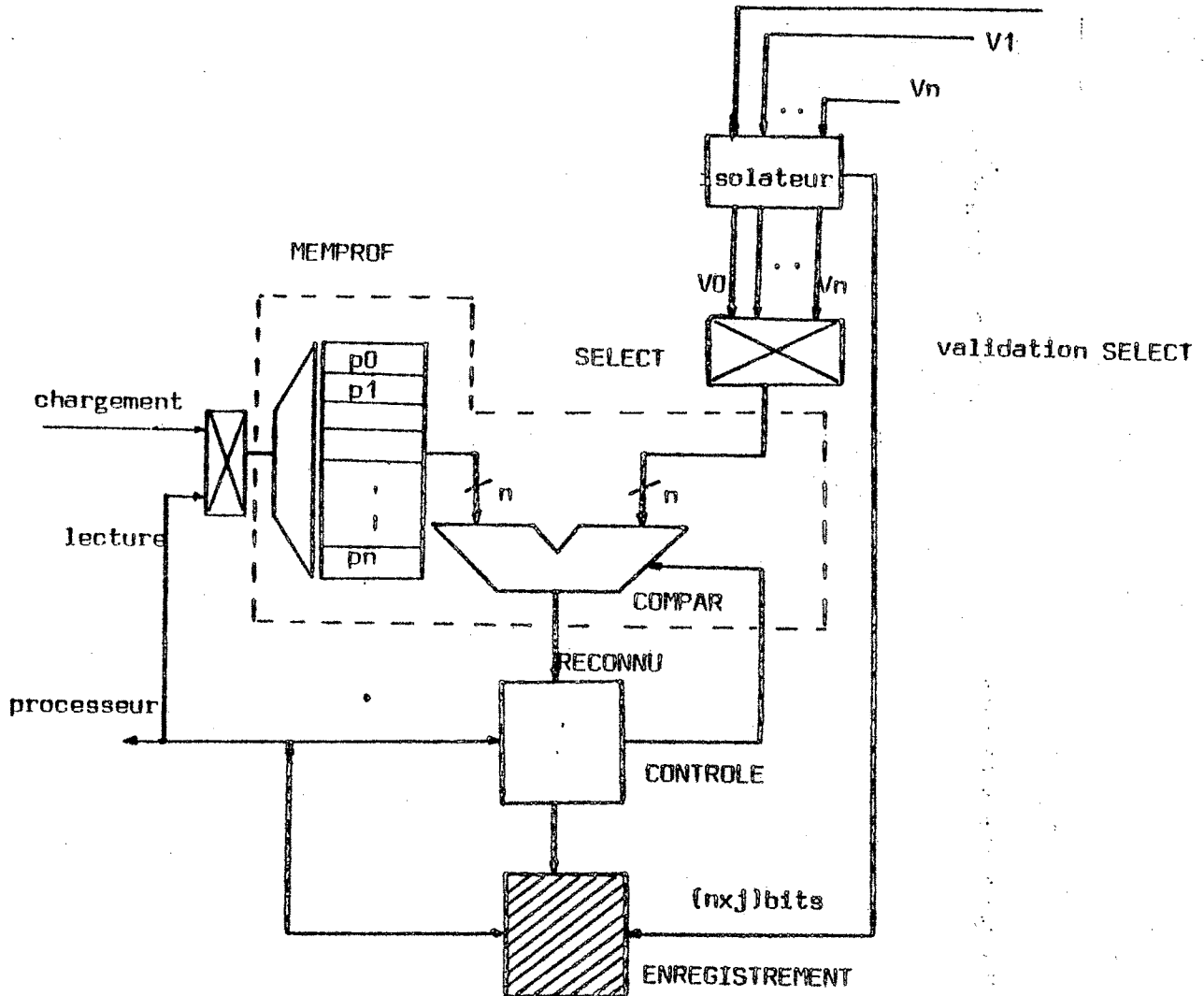


Figure 6.6 Le système multi-profil aléatoire.

#### 4.2.2) La partie enregistrement.

L'enregistrement est fait dans une mémoire rapide dont le temps d'accès est inférieur au temps de lecture et d'écriture du micro-processeur objet.

Cet enregistrement est organisé en mots de  $n*j$  bits où  $n ::=$  nombre de voies, et  $j ::=$  nombre de lignes par voie. La mémoire est organisée en  $K*m$  bits où  $k ::=$  nombre de mots mémoires à  $m$  bits, et  $m := (n*j)$  bits.

Nous considérons que l'enregistrement des événements s'effectue suivant trois possibilités:

- 1)  $k$  enregistrements avant et  $k$  après le profil,
- 2)  $2k$  enregistrements après le profil,
- 3)  $2k$  enregistrements avant le profil.

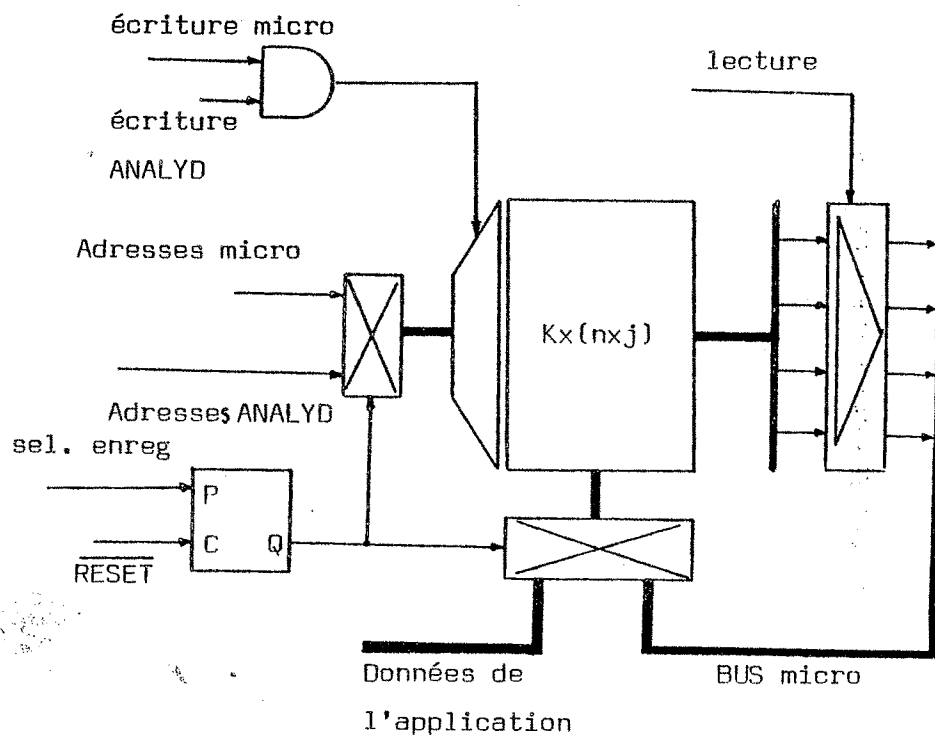


Figure 6.7 La partie enregistrement.

La partie enregistrement est commune aux systèmes mono et multi-profils.

## 5.) LE SYSTEME ANAMAD.

Ce système développé, dans le cadre de l'équipe d'architecture de Grenoble, a été fait à partir du système MADAM (RT3-78).

ANAMAD a été conçu pour

- analyser plusieurs lignes parallèles de données,
- aider à la mise au point d'applications logicielles,
- regarder le comportement d'un processus qui se déroule dans une autre machine.

ANAMAD a les caractéristiques matérielles suivantes:

- d'être mono-profil,
- d'avoir des capteurs matériels(parallèles),
- de sauvegarder 256 événements avant et après la détection du profil.

Suivant le formalisme de MICROD, le processeur ANAMAD se décrit:

```

<ANAMAD> ::= <matériel ANAMAD> [ ] <logiciel ANAMAD>
<matériel ANAMAD> ::= <bloc processeur> <bloc spécialisé>;
<bloc spécialisé> ::= <capteurs matériels>
                    <mécanisme de contrôle>
                    <mécanisme d'enregistrement>;
  
```

### 5.1) Le matériel ANAMAD.

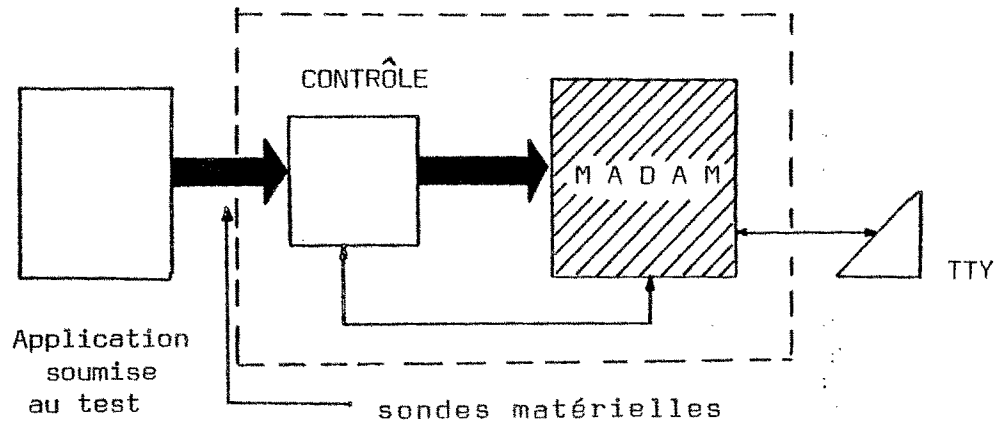


Figure 6.8 Le système ANAMAD.

#### 5.1.1) Le bloc processeur.

ANAMAD est piloté par un microprocesseur, du type MC6800, et par un ensemble d'éléments matériels pour la surveillance du système.

Ces éléments matériels sont les suivants:

- l'horlogerie centrale développée autour du circuit d'horloge MC6871A.
- le logiciel propre d'ANAMAD, implanté sur une mémoire morte d'une capacité d'un 1K-octets.
- la mémoire de travail de ce logiciel: MC6810(128 octets).
- l'interface de communication asynchrone pour le dialogue avec l'utilisateur.
- interface avec la mémoire rapide d'enregistrement de MADAM, et les amplificateurs d'adresses, des données et les circuits TTL pour générer les signaux de contrôle du bloc processeur et du système MADAM (ordre d'écriture,...).

## 5.1.2) Le bloc' spécialisé.

### 5.1.2.1) Les capteurs.

Les capteurs amènent ,au système ANAMAD, les différentes lignes d'information de la machine soumise au test. Ces lignes au nombre de 32, représentent généralement

- 16 lignes d'adresses,
- 8 lignes de données,
- 8 lignes,appelées, de contrôle(ex:R/ $\bar{W}$ ,VMA,..).

### 5.1.2.2) Le mécanisme de contrôle.

Le mécanisme de contrôle a les fonctions suivantes :

- de choisir le type de profil sur les lignes de données de l'application.
- de placer le profil à reconnaître dans le registre PROFIL.
- de valider la comparaison entre les données provenant de l'application'et le profil.
- de déclencher l'enregistrement avant et après l'arrivée du profil(MATCH)

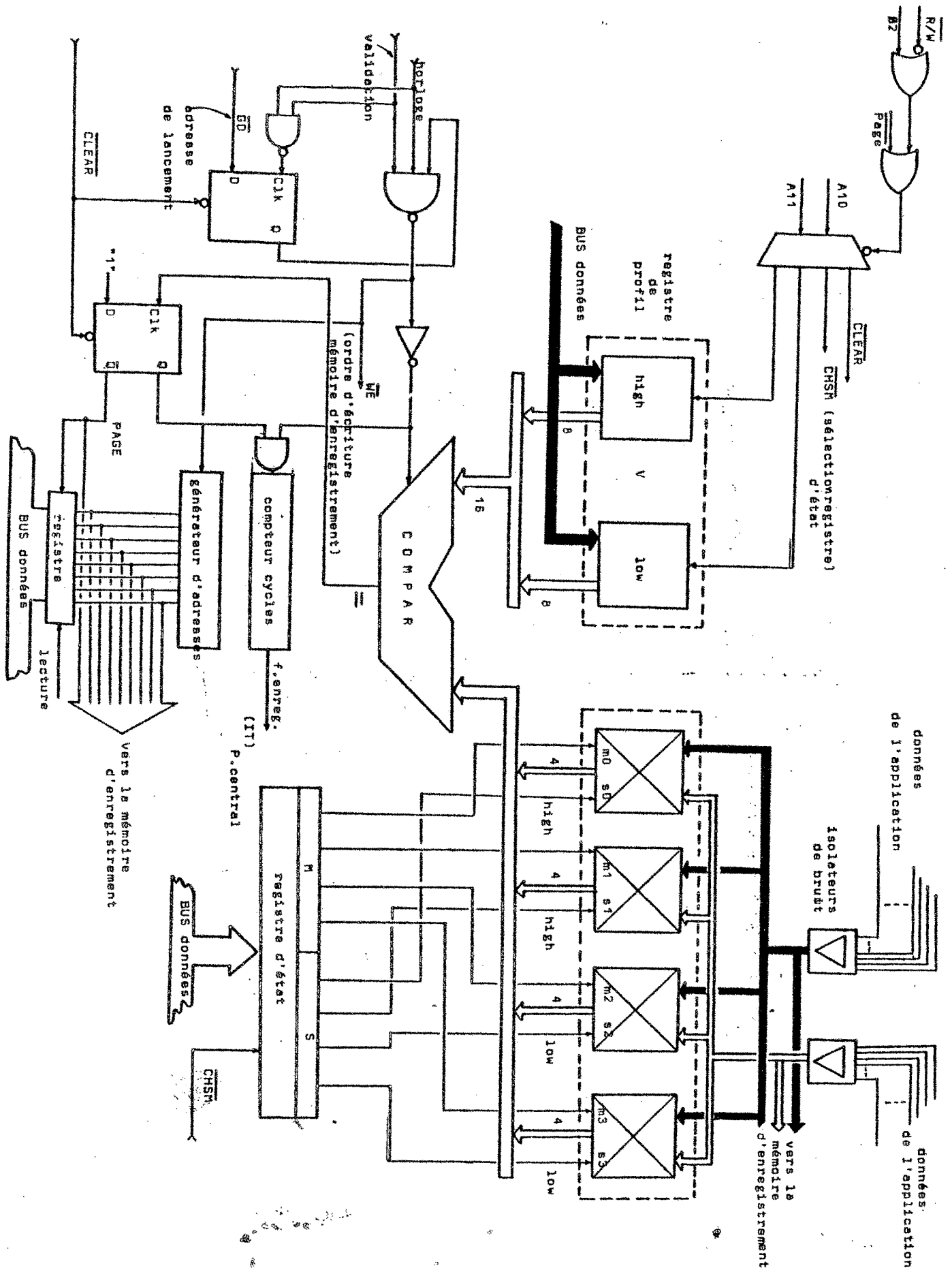


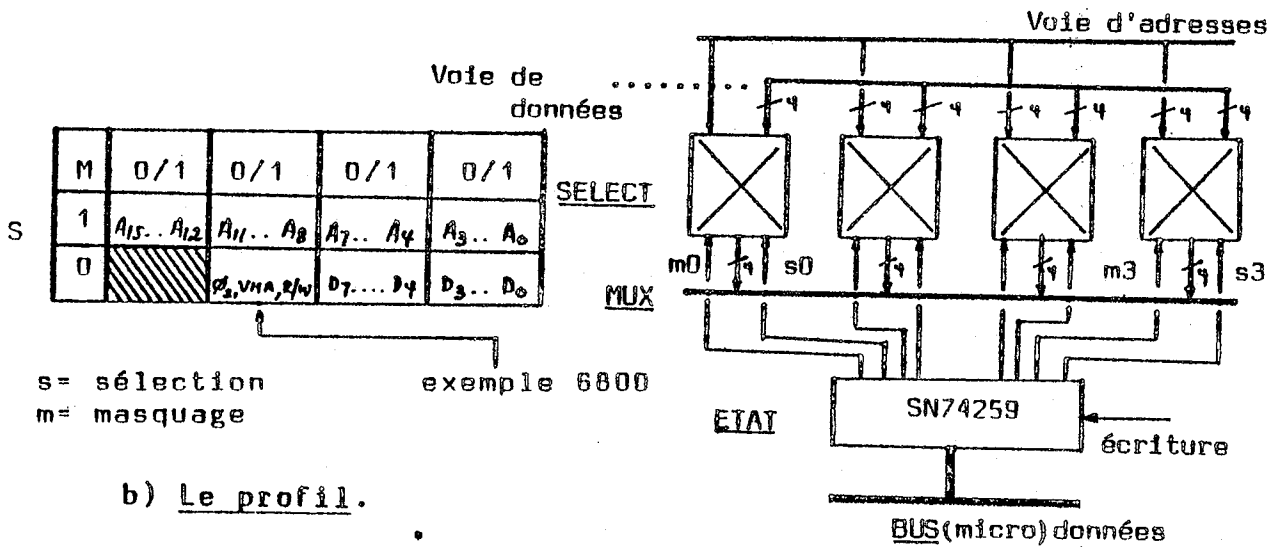
Figure 6.9 Le mécanisme de contrôle.



Nous allons décrire ci-après les caractéristiques matérielles des différents éléments du mécanisme de contrôle.

a) Le type de profil est choisi dans SELECT. Le contrôle de SELECT est fait par ETAT. Ce dernier est chargé de sélectionner chacun des circuits qui forment SELECT.

SELECT est câblé conformément au tableau suivant.



b) Le profil.

Le profil que l'utilisateur veut détecter est placé dans deux registres du type SN74116, de la manière suivante:

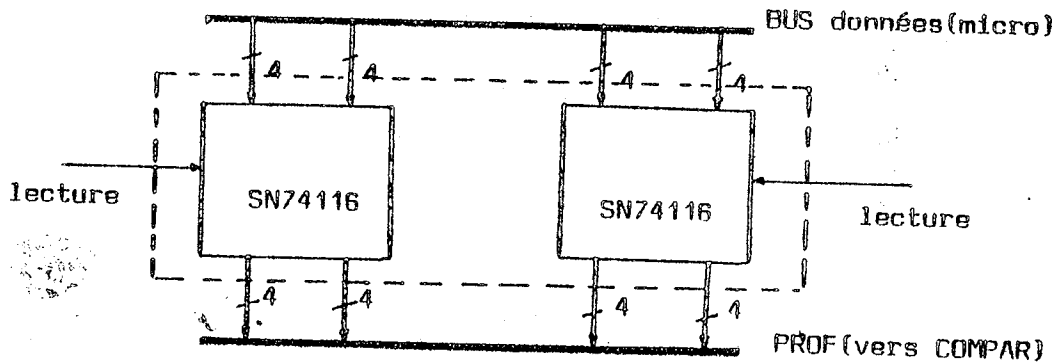


Figure 6.10 Le profil.

c) Le comparateur.

Le comparateur COMPAR est chargé de détecter le profil PROF parmi les signaux sélectionnés dans MUX. Il génère également l'ordre de lancement de l'enregistrement une fois détecté le profil.

Il faut remarquer qu'une fois mis en route le système, il enregistre cycliquement les informations jusqu'à l'arrivée du profil.

Le profil détecté, il commande la commutation de page d'écriture et déclenche un compteur de cycles ( 256 pas).

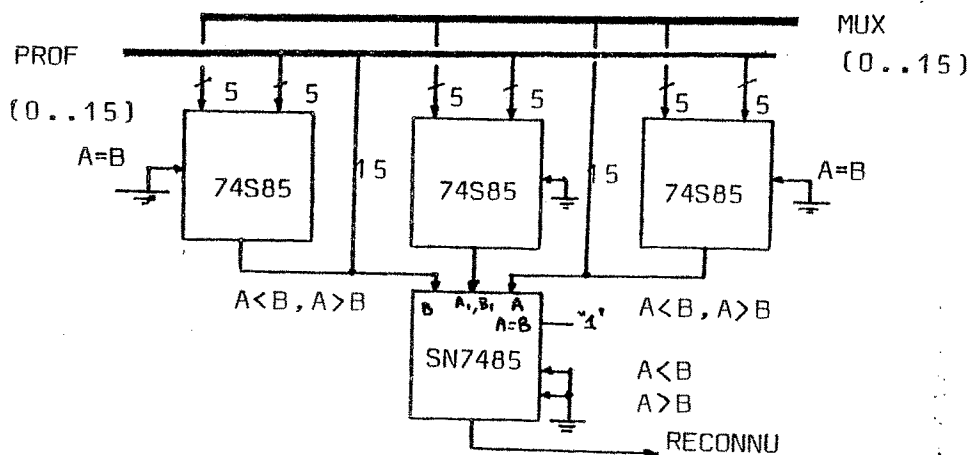


Figure 6.11 Configuration du comparateur.

### 5.1.2.3) Le mécanisme d'enregistrement.

La mémoire d'enregistrement est utilisée par le processeur, pour la lire et la tester, et par le système pour y stocker les données de l'application.

Cette mémoire est organisée en deux modules de 256\*32 bits

Cette organisation est due au fait que le système MADAM est prévu pour enregistrer 256 événements avant et après la détection du profil. Le mécanisme d'enregistrement se présente de la même manière que celui schématisé dans la figure 6.7.

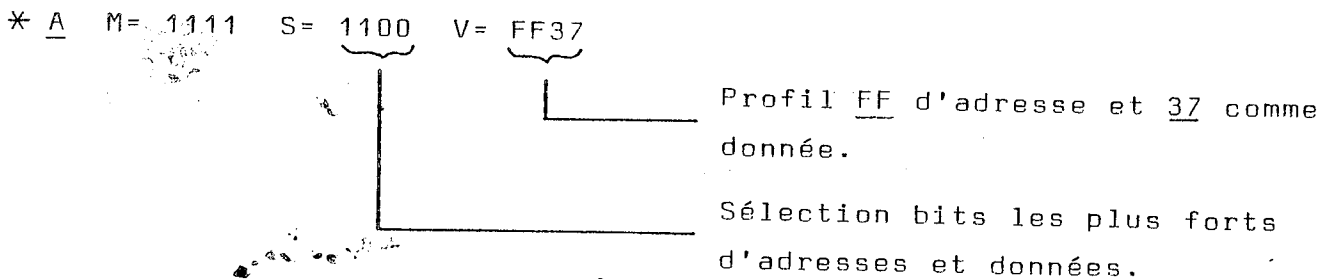
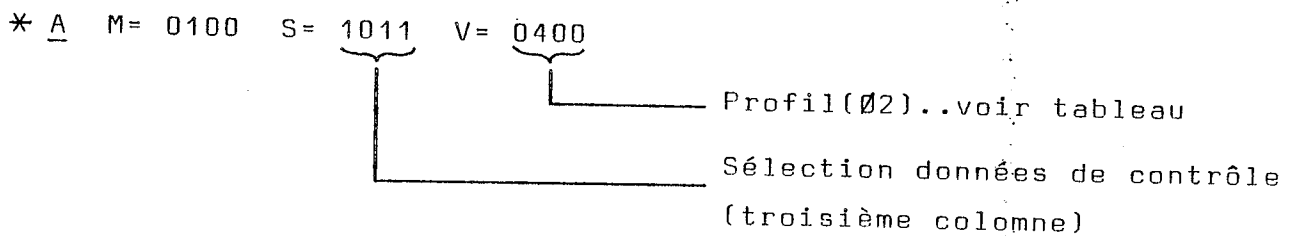
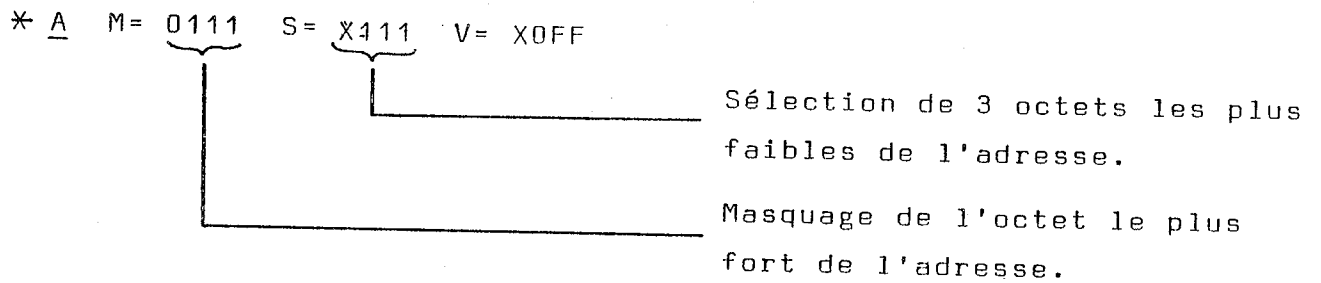
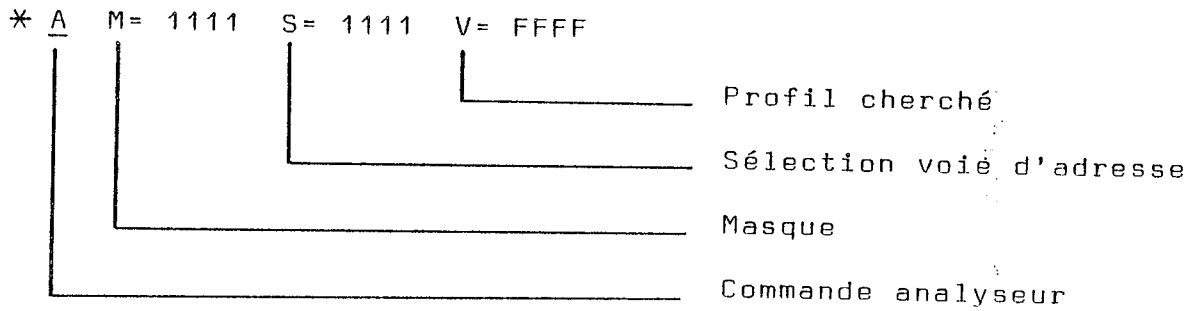
### 5.2) Le logiciel ANAMAD.

Le logiciel système contient un ensemble de commandes, propres au système ANAMAD, qui permettent:

- de positionner le profil,
- de choisir la voie où se trouve le profil,
- de valider et de sélectionner SELECT,
- de visualiser sur une imprimante les événements enregistrés avant et après l'arrivée du profil. Il contient également les commandes classiques de maintenance d'un moniteur système (ex: analyse de la mémoire, test,...).

Exemples.

Nous montrons ci-après les diverses commandes du logiciel ANAMAD utilisées.



## 6.) L'EXTENSION d'ANALYD.

Lorsqu'il s'agit de surveiller une grande quantité d'informations provenant d'un processus en temps réel, le mécanisme d'enregistrement présenté auparavant ne sera pas suffisant pour les stocker.

Nous ajoutons donc au système une mémoire de masse de manière à pouvoir stocker ces informations.

L'incorporation de ces mémoires au système impose une contrainte sur leur vitesse d'enregistrement. En considérant que ces mémoires sont lentes, il faudra trouver une méthode d'enregistrement des événements suffisamment rapide.

Le temps d'enregistrement ( $T_{enr}$ ) dépend de la fréquence d'arrivée des informations (événements).

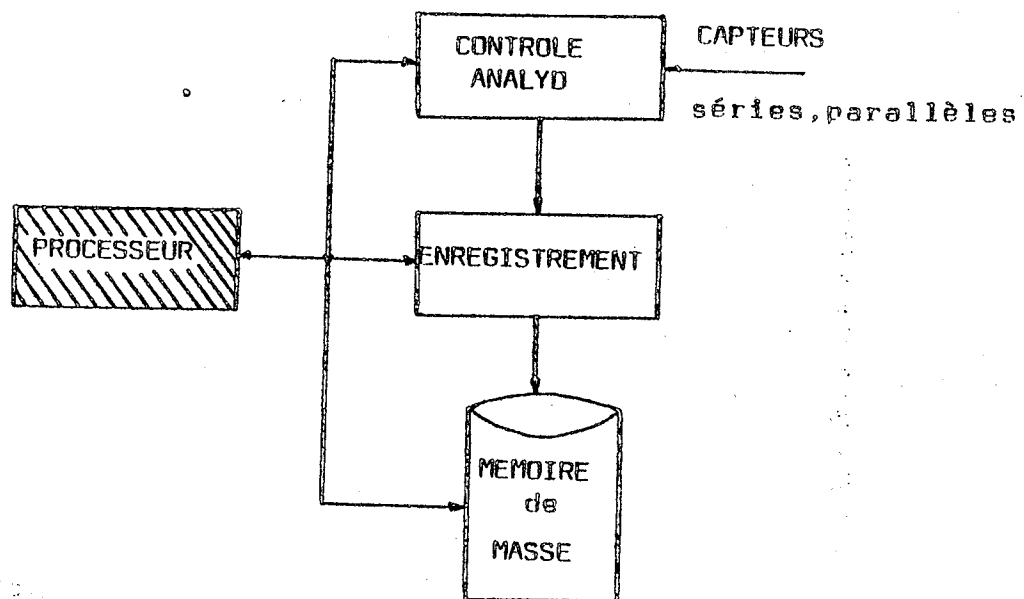


Figure 6.12. L'extension d'ANALYD.

### 6.1) Organisation de l'enregistrement.

Le mécanisme d'enregistrement est organisé à l'aide d'une mémoire d'enregistrement temporaire (tampon), et d'une mémoire de masse.

La mémoire d'enregistrement temporaire est chargée de stocker les événements qui arrivent par les capteurs séries ou parallèles.

Pour cette extension, on a choisi comme mémoire de masse un disque souple. Ce disque, organisé en pistes et secteurs, nous permet de fixer une capacité d'enregistrement égale à la capacité maximale d'une piste ou de  $n$  pistes.

Le critère mentionné, ci-dessus, impose que la taille de la mémoire d'enregistrement, temporaire, soit fonction de la capacité de stockage permise pour ANALYD.

Pour économiser cette mémoire tampon, on définit un mécanisme qui gère et contrôle les informations vers le disque souple.

### 6.2) Description du tampon.

Nous pouvons représenter cette mémoire comme une série de tampons circulaires.

Le choix du nombre de tampons et de leur capacité dépend d'une part, de la mémoire disponible et d'autre part, de la fréquence d'enregistrement de ces tampons ( $1/T_{enr}$ ), ainsi que de la fréquence de vidage de ces tampons vers le disque souple ( $1/T_{vid}$ ).

Nous considérons le nombre de tampons ( $n$ ) borné.

En supposant qu'il s'agit de ranger  $n$  tampons; nous analyserons deux cas pour le choix du nombre de tampons:

1.  $f_{enr} > f_{vid}$ ;
2.  $f_{enr} \leq f_{vid}$ .

1. Etudions d'abord le cas ( $f_{enr} > f_{vid}$ ), avec un exemple. Admettons que nous avons une ( $f_{enr} = f_{vid} * 2$ ) et que nous voulions un nombre de tampons à vider égal à deux.

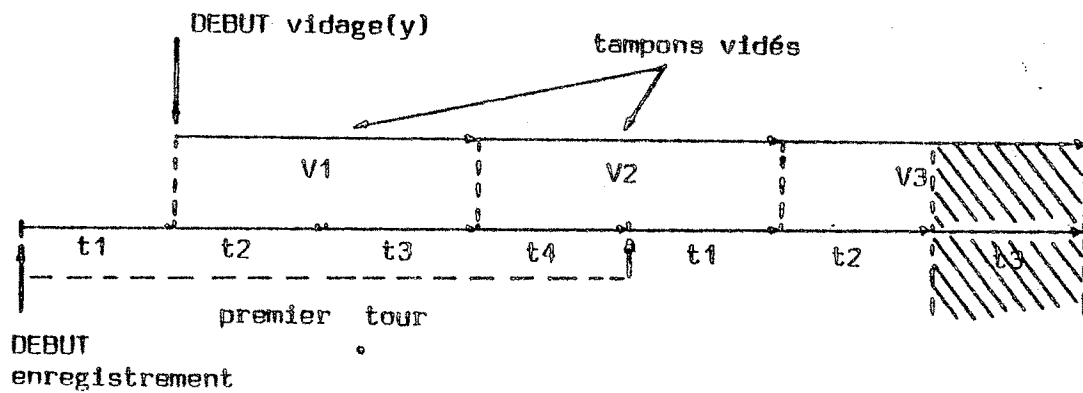


Figure 6.13 Différents temps d'enregistrement et de vidage.

Comme nous pouvons le constater dans la figure ci-dessus, on arrive à vider seulement deux tampons en deux cycles non complets; à la fin de ce deuxième cycle le troisième tampon est écrasé par le nouvel enregistrement.

Les relations suivantes permettent de connaître le nombre de tampons qu'on peut vider sans écrasement de données.

Soit  $f$  le rapport

$$f = t(\text{vidage}) / t(\text{enregistrement});$$

Durant le vidage de  $V_1$ , s'enregistrent les tampons compris entre  $t_1$  et  $t(f+1)$ . Ainsi

Vidage                      Enregistrement

$$\begin{array}{rcl} V_1 & & t_1 \dots \leq t(f+1) \\ V_2 & & \leq t(f+1) \dots \leq t(2f+1) \\ & * & * \\ & * & * \\ & * & * \\ V_n & & \leq t((n-1)f+1) \dots \leq t(nf+1) \end{array}$$

puisque nous avons supposé que le nombre total  $n$  de tampons est borné. Lorsque  $V_n$  finit de se vider,  $t(nf+1)$  est en écriture.

si

$k$ : nombre de cycles complets,  
 $X$ : nombre de tampons en mémoire,  
 $x$ : indice du tampon en écriture,

alors  $nf+1 = kX+x$ . Afin qu'il n'y ait pas d'écrasement il faut que le tampon  $V_n$  à vider soit d'un indice supérieur ou égal à  $(k-1)X+x+1$  donc  $n \geq (k-1)X+x+1$ ;

$$\text{d'où } X \geq n(f-1)+2$$

On peut généraliser les relations précédentes. Soit  $(y)$  le début du vidage:

d'où

$$X \geq n(f-1)+2y;$$

$$k \leq (nf+y)/X;$$

$$x \geq nf+y-kX;$$



Ces relations, permettent de connaître par exemple le nombre de tampons mémoires, à des fréquences d'enregistrement et de vidage connues, nécessaire pour sauvegarder(n) de ces tampons dans la mémoire de masse.

/Exemple.

Admettons que nous voulions sauvegarder des événements qui arrivent toutes les 25 micro-secondes(40 K-octets/sec) sur un disque souple pour leur traitement ultérieur.

Nous supposons qu'au moment où commence l'enregistrement, l'adresse de la tête du disque souple est correctement positionnée (piste, secteur).

Pour cela, nous avons les relations suivantes:

$t_{enr} = 25 * \text{capacité du tampon}$

$t_{vid} = 32 * \text{capacité du tampon}$

$f = t_{vid}/t_{enr}$ ; relation d'enregistrement;

$f = 1.28$

Avec ces données, nous pouvons calculer

- le nombre de tampons mémoires(X) pour sauvegarder n tampons
- le nombre de cycles k,
- le dernier tampon d'écriture(x) après k cycles

On choisira pour l'exemple  $n=15$ ; c'est-à-dire quinze tampons à vider

d'où

$X=7; k=2; x \geq 6.20$ ;  $x = 7$  après k cycles.

Les procédures pour l'enregistrement et le vidage sont montrées ci-après

Procédure enregistrement;

Début

j ::= 0;

Début

recherche adresse; CO positionnement de la tête;

Tant que j < capacité tampon faire

M(i,j) ::= enregistrement; CO événements;

Fin.

Procédure vidage;

Début

attente temps de départ de vidage(y);

CO; i ::= numéro du tampon i;

k ::= 0;

Tant que k < capacité tampon faire

vidage ::= M(i,k); CO transfert des données vers le disque souple;

Fin.

L'implantation logicielle du contrôle de l'enregistrement et du vidage vient d'être définie ci-dessus.

L'algorithme général du système est présenté dans la procédure SANAMAD.

Procédure SANAMAD;

Début

i:=1;CO i compteur d'enregistrements;

Tant que  $i \leq (nf+2y)$  faire

Debut

enregistrement(i);

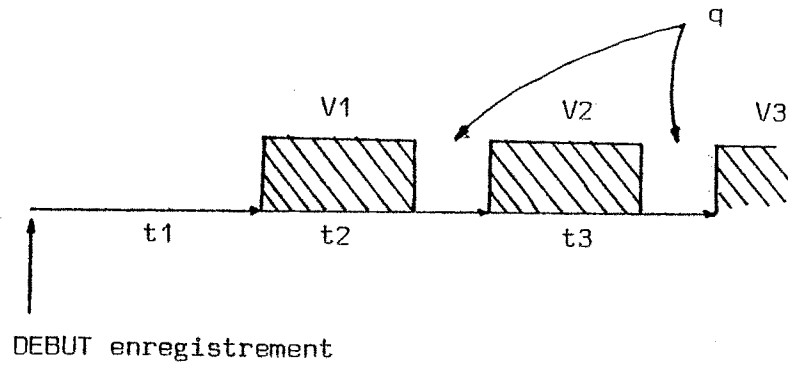
PARALLEL(enregistrement(i+1),vidage(i));

Fin;

Fin.

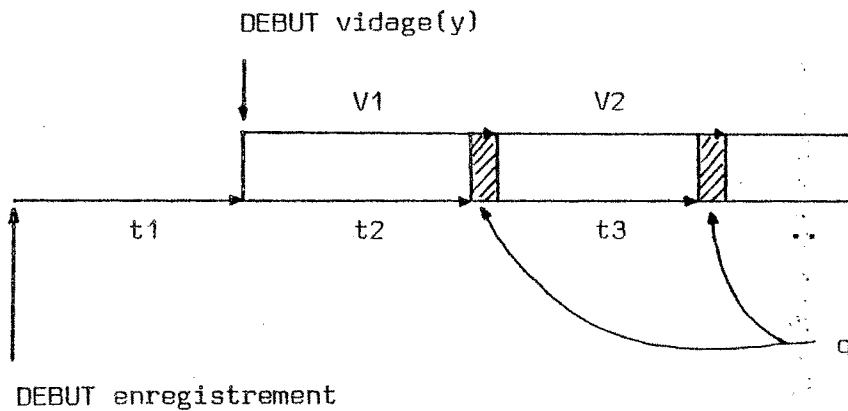
/2. Dans le cas où  $t_{enr} \leq t_{vid}$ , le problème se pose différemment car on a du temps pour sauvegarder les informations et il nous reste le temps(q) pour exécuter des actions diverses(ex: chercher la prochaine adresse dans le disque souple).

Dans ce cas deux tampons seront suffisants pour enregistrer toutes les données de l'application. Le diagramme présenté ci-dessous, nous montre les différents temps( $t_{enr}, t_{vid}, q$ )



$$f_{enr} < f_{vid}$$

On considère le cas où  $f_{enr} = f_{vid}$  comme faisant partie de celui où  $f_{enr} > f_{vid}$  car, le mécanisme de vidage a besoin d'un temps ( $q$ ) pour positionner la nouvelle adresse d'enregistrement (piste, secteur) par exemple.

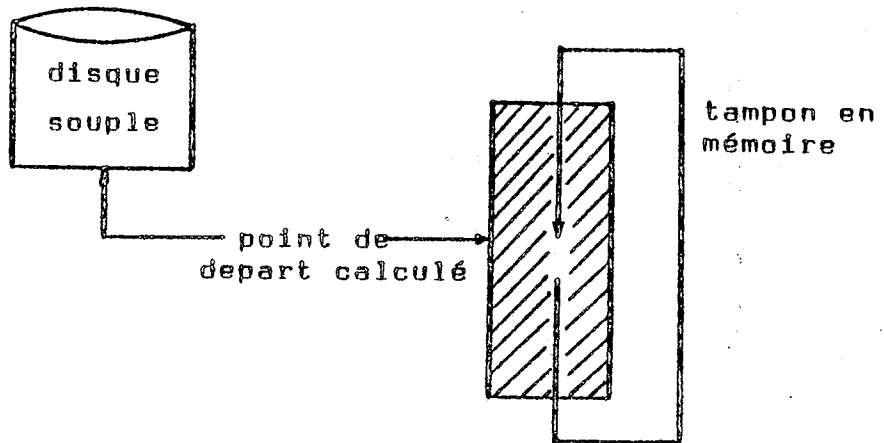


$$f_{enr} = f_{vid};$$

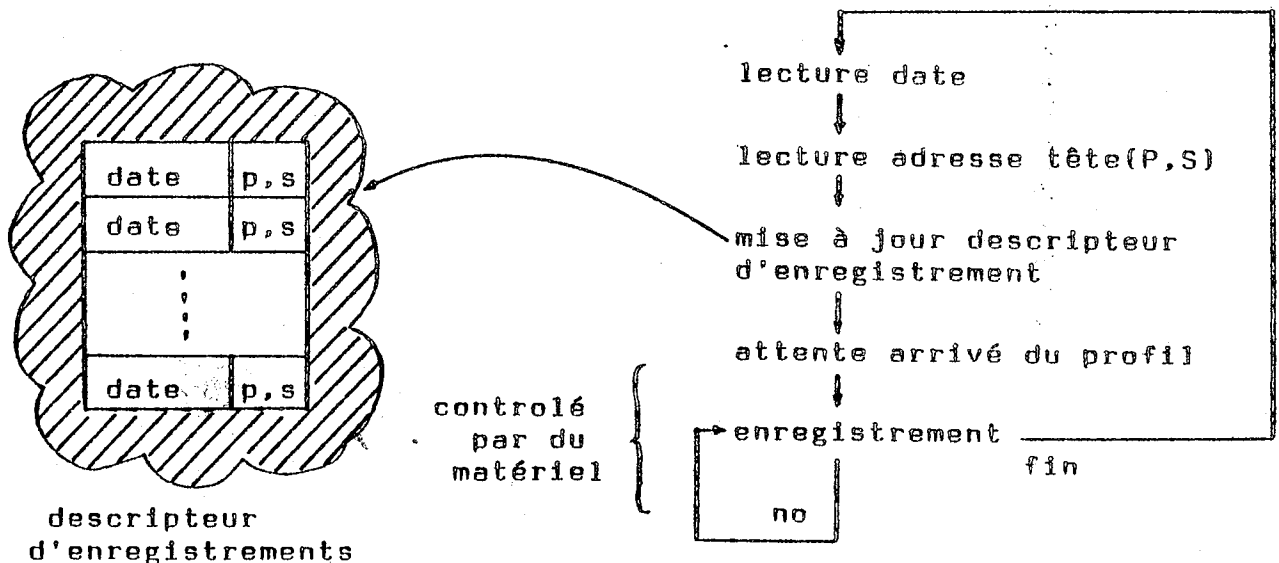
Il faut remarquer que ce temps ( $q$ ) doit toujours être considéré pour  $f_{enr} \geq f_{vid}$  et  $f_{enr} < f_{vid}$ .

Remarque.

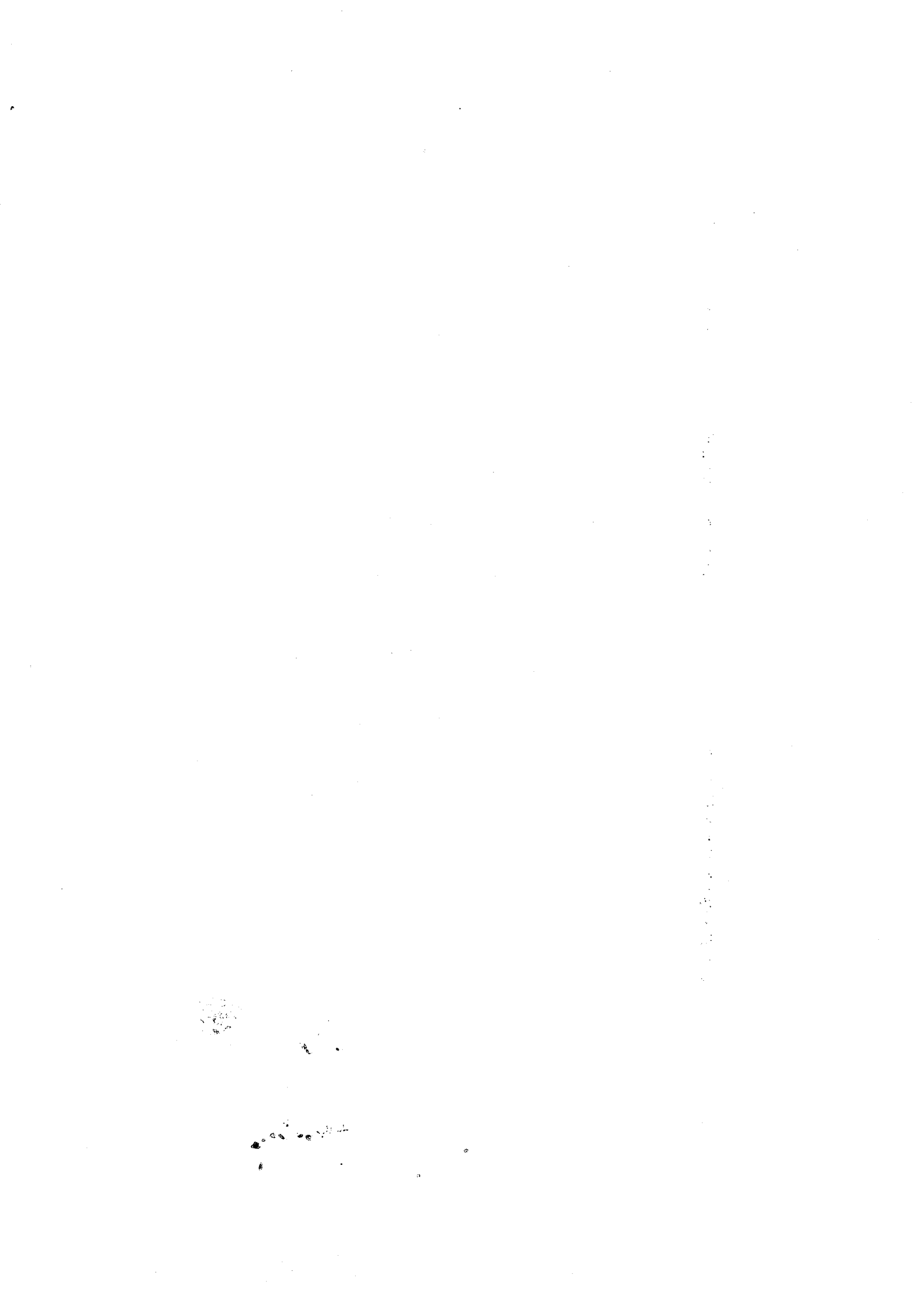
Nous pouvons améliorer la vitesse d'enregistrement en utilisant le mécanisme du chargement à la volée, laquelle suppose des points de départ calculés (adresses dans la mémoire tampon fonction de l'adresse courante sur le disque souple).



On utilise un descripteur d'enregistrements pour repérer la date d'arrivée des enregistrements et l'adresse où ont été enregistrés les événements sur le disque souple (piste, secteur).



•  
CONCLUSIONS



## CONCLUSIONS

Nous avons essayé dans cet ouvrage, de décrire un méthode de conception et de description d'applications microprocesseurs vis-à-vis de leurs réalisations physiques.

Trois principaux buts, ont été poursuivis et atteints dans ce travail:

- le premier était d'utiliser les concepts informatiques (processeur, procédure, langage) pour mieux définir et décrire le logiciel et le matériel d'applications microprocesseurs.

- le deuxième était d'utiliser ces méthodes dans l'enseignement et la recherche. Dans cet esprit les étudiants de fin d'année de l'ENSIMAG (Troisième année, Section Spéciale) s'en servent; également ces méthodes s'appliquent dans la même optique à l'Université Central de Vénézuéla et aussi à l'IUT-RC-CARACAS.

- le troisième était de nous familiariser avec les lignes directrices de la mise au point d'applications microprocesseurs (méthodes, outils,..) en réalisant un périphérique d'aide au développement et d'analyse de telles applications.



La réalisation, conception et implantation, physique de chaque composant du périphérique, nous a permis d'acquérir une expérience dans la conception et dans la mise au point d'applications similaires: ce périphérique est utilisé au sein de l'Equipe d'Architecture des Ordinateurs, comme un outil de mise au point d'applications microprocesseurs.

De manière à faciliter l'utilisation du périphérique, dans un contexte d'une machine générale ou, dans le contexte d'un outil de mise au point évolué; l'élaboration d'une dernière couche logicielle, englobant les différents logiciels de base de chaque composant, reste à faire.

BIBLIOGRAPHIE



## BIBLIOGRAPHIE

(ANC-74) F. ANCEAU

Contribution à l'étude de systèmes hiérarchisés de ressources dans l'architecture de machines informatiques.

Thèse d'état, Grenoble, le 5 Décembre 1974

(AME-75) G. F. AMELIO

Charge Coupled Devices for Memory Applications  
AFIPS, Conference Proceeding, Vol 44, May 1975

(AND-76) D. R. ANDERSON

Data Base Processor Technology

AFIPS Conf., Proc 76

NCC, Vol 45, Juin 1976

(ALL-77) D. R. ALLISON

A Design Philosophy for Microcomputer Architectures

Computer IEEE, February 1977

(ANC-78) F. ANCEAU

Principes et mise en oeuvre de microprocesseurs

ENSIMAG, Janvier 1978

(ALF-79) M. ALFORD

Requirements for Distributed Data Processing Design

IEEE, The 1st International Conference on Distributed Computing Systems

Huntsville, Alabama, October, 1979

- (AKE-79) L.AKEJOHANSSON  
Virtual Memory Management for Microcomputers in  
Real Time applications  
Euromicro Journal, Vol 5, N.4, July 1979
- (BEN-70) C.G.BELL, A.NEWELL  
The PMS and ISP descriptive Systems for Computer  
Structures  
1970, Sprint Joint Computer  
Conf, AFIPS, Vol 33, Monvale, N.J, 1970
- (BCF-70) C.G.BELL, R.CARRY, A.McFARLAND, B.DEGALI  
J.O'LAUGHLIN, R.NOONAN  
A new Architecture for mini-computer:  
The DEC-PDP-11, Sprint Joint Computer Conf.,  
AFIPS, Montvale, N.J, Vol 33, 1970
- (BEN-71) C.G.BELL, A.NEWELL  
Computer Structure Reading and Exemples  
Mc Graw Hill, New York, 1971
- (BAR-75) M.R.BARBACCI  
A comparison of registre transfert Languages for  
describing Computer and Digital Systems  
IEEE transaction in Computer, Vol C-24, N.2, February  
1975
- (BHR-75) Y.BEKKERS, D.HERMAN, M.RAYNAL  
Conception et réalisation d'une machine langage de  
haut niveau adaptée à l'écriture de systèmes.  
Thèse 3ème cycle, Rennes, Septembre 1975.
- (BSR-75) S.BROFFERIO, G.SCIRE  
Proposal for a Normalized Microcomputer System  
Euromicro Newsletter, Vol 1, N.4, July 1975

- (BLG-78) E.BLOCH,D.GALAGE  
Its effect on High-Speed computer architecture and  
machine organisation.  
Computer IEEE, April 1978
- (BCG-78) BEAUFILS,COHEN,GLIZE,LITAIZE  
Random-Like Sequential Memories Used as Main  
Memories  
Euromicro Symposium, Munich, October 1978
- (CAJ-74) H.CAPLENTER,J.A.JANKU  
Top-Design approach to LSI system design  
Computer Design, August 1974
- (CRO-75) C.R.O.C.U.S  
Système d'exploitation des ordinateurs  
Dunod, Paris, 1975
- (CCE-76) H.CROUCH,J.CORNETT,R.EDWARD  
CCDs in Memory System move into sight  
Computer Design, September 1976
- (CAP-78) R.CAPECE  
Microprocessor and Microcomputer  
Electronic, October 1978
- (CMP-79) B.COURTOIS,M.MARINESCU,J.P.PONS  
LECERF,LEPETIT,VOMSCHEID  
Local Network as Support of Dedicated Distributed  
System  
Rapport Interne, Equipe d'Architecture des  
Ordinateurs  
ENSIMAG,Grenoble,Décembre 1979

- (DWS-75) H.DWIGHT,SAWIN III  
Microprocessor Systems  
Euromicro Newsletter,Vol 1,N.5,July 1975
- (DUP-78) J.DUCHENE,J.PRO  
Microprocessor based prototype development problems  
and solutions: a low-cost development system.  
Euromicro Journal,Vol 4,N.6,November 1978
- (FIS-75) E.FISHER  
Speed Microprocessor responses without interrup or  
DMA techniques  
Electronic Design 23, Novembre 8,1975
- (FIC-78) M.L.FICHTENBAUM  
Top-Down Design Streamlines Digital System Projects  
Computer Design,September 1976
- (LUZ-78) F.LUZIO  
Behavioral Description of Microcomputer with aid of  
the PMS notation  
Euromicro,Vol 4,N.2,1978
- (GUI-78) S.GUIBOUD-RIBAUD  
Architecture de systèmes. Une vue synthétique:  
logicielle et matérielle.  
Ecole d'été du Forez, Juillet 1978
- (HOR-76) S.HOEVER,W.ROEHDER  
Modular Multi-processor Architecture with Virtual  
Memory.  
Proc,2nd Euromicro Symposium,1976

(HUS-78) S.S. HUSSON

System Development Methodology  
Euromicro, Vol 5, January 1978

(IFS-64) K.E. IVERSON, A. FALFOFF, B. SUSSENGUTH

A formal Description of System/360  
IBM Systems Journal, Vol 3, N.3, 1964

(JOS-74) H.F. JORDAN, B.J. SMITH

Structure of Digital System Description Languages  
Proceeding of the First Annual Symposium on  
Computer Architecture  
University of Florida, December 1974

(LEE-72) J. LEE

Computer Semantic Studies of Algorithms Processor  
and Languages  
Van Nostrand, Reinhold Company 1972

(LEE-74) J. LEE

VDL- A definition System for all level  
Proceeding of the first Annual Symposium on  
Computer Architecture  
University of Florida, Decembre 1974

(LIP-77) G.F. LIPOSKY

On imaginary field, token transfert and floating  
codes in intelligent secondary memories.  
Rapport Interne du département de Genie Electrique  
University of Texas, 1976-1977

(LIS-77) A.M. LISTER, P.J. SAVER

Hierarchical Monitors  
Software-Practice and Experience, Vol 7, 1977



- (MRW-76) G.MICHEL,P.ROLIN & C.WAGNER  
Microprocessor based disc system  
Microprocessor, Vol 1, Septembre 1976
- (McK-78) M.S.McKENDI  
The use of monitors in microprocessor software  
development  
Euromicro Journal,VOL 4,1978
- (MET-78) M.MEZZALAMA,P.TORASSO  
A microprocessor executive for real-time process  
control  
Euromicro Journal,Vol 4,N.2,March 1978
- (MTV-78) B.MAILLOT,A.TARABOUT,I.VATTON  
Un outil de recherche en systèmes informatiques  
Thèse I.N.P.G,Grenoble,11 de Décembre 1978
- (PLM-77) J.F.PEDENON,S.MONCHAUD,B.LEMAIRE  
Microprogrammed Biprocessor System for Real Time  
Processing  
Euromicro Symposium,Amsterdam,October 1977
- (ORG-72) E.I.ORGANICK  
The MULTICS System: an examination of its structure  
M.I.T Press, Cambridge,1972
- (ORG-78) E.I.ORGANICK  
New Directions in Computer Architectures  
Euromicro,Vol 5,1978

(SCH-77) J.P.SCHOELLKOPF

Machine PASC-HLL: Définition d'une architecture  
pipe-line pour une unité centrale adaptée au  
langage PASCAL.

Thèse de 3ème cycle, Grenoble, le 28 juin 1977.

(SCL-79) P.C.SCHOLL

Vers une programmation systématique:

Etude de quelques méthodes, techniques et outils

Thèse d'état, Grenoble, le 29 Juin 1979

(TOV-78) B.TOURSEL, P.VANLAER

A New Organisation of Information Streams for  
Processing with Shifting Memories

Euromicro Symposium, Munich, October 1978

(WAT-73) E.WAYNE, J.TRACEY

FLOWWARE: A Flow Charting Procedure to Describe  
Digital Network

Proceeding of the 1st Annual Symposium on Computer  
Architecture

University of Florida, December 1973

(WYL-75) D.WYLAND

Increased Microcomputer Efficiency

Electronic Design, November 1975

(ZUR-68) F.W.ZURCHER & B.RANDELL

Iterative multi-level modeling a methodology for  
computer systems design.

Proceeding of the AFIPS Congress, 1968

(ZAM-76) A.ZAMBRANO

Le processeur de traduction d'un système  
téléphonique

Projet du D.E.A, ENSIMAG, Grenoble, 1976

(ZIB-77) T.A.ZIMMERMAN, D.F.BARBE

A New Role for Charge Coupled Devices: Digital  
Signal Processing

Electronic, March 1977.

## REFERENCES TECHNIQUES.

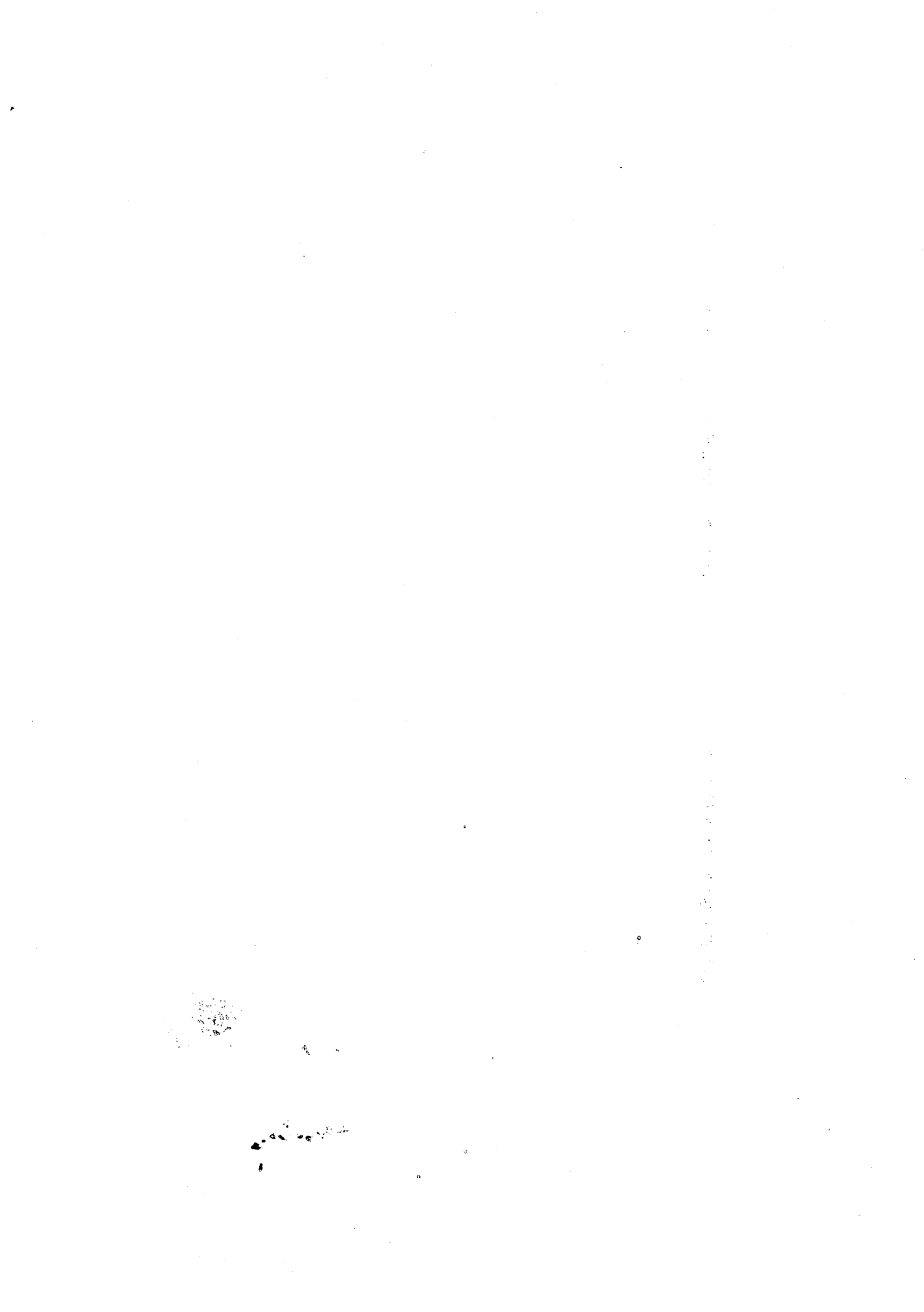
- (RT1-77) Systems Reference and Data Sheets  
Motorola Semiconductor Product Inc, 1977
- (RT2-78) Systèmes universels de développement pour  
microprocesseur 8002/8001  
Manuel système, 1978
- (RT3-78) J.P. SCHOELLKOPF  
Presentation du système MADAM  
Matériel d'Aide au Développement d'Applications  
Microprogrammées.  
ENSIMAG, 1978
- (RT4-75) BOB.PAPENBERG  
Design and applications of INTEL'S 2416 16K Charge  
Coupled Device  
Application note, INTEL Corporation, 1975
- (RT5-75) S.L. REGE  
Performance and Power dissipation analysis for CCD  
memory systems  
BORROUHS Corporation, New Jersey, 1975
- (RT6-78) SHUGART ASSOCIATES  
SAB00/801 Diskette Storage Drive  
OEM Manual, 1978

(RT7-78) WESTERN DIGITAL  
INS1771-1 Floppy Disk Formatter/Controller  
General Description, 1978

(RT8-78) M6800 DISK OPERATING SYSTEM  
Preliminary USER'S Guide  
MOTOROLA Microsystems, Phoenix, Arizona, 1978

ANNEXES

- I. LE NOYAU: LE PROCESSEUR CENTRAL
- II. UN OUTIL DE MISE AU POINT ET DE MAINTENANCE IN SITU  
D'APPLICATIONS MICROPROCESSEURS.



## ANNEXE I

### LE NOYAU: LE PROCESSEUR CENTRAL(P.central)

#### 1.) Le bloc processeur

##### .1) Le module de traitement

- .1) Le microprocesseur MC6800
- .2) L'horlogerie centrale
- .3) L'espace d'adressage du P.central
  - .1) L'espace système
  - .2) L'espace utilisateur

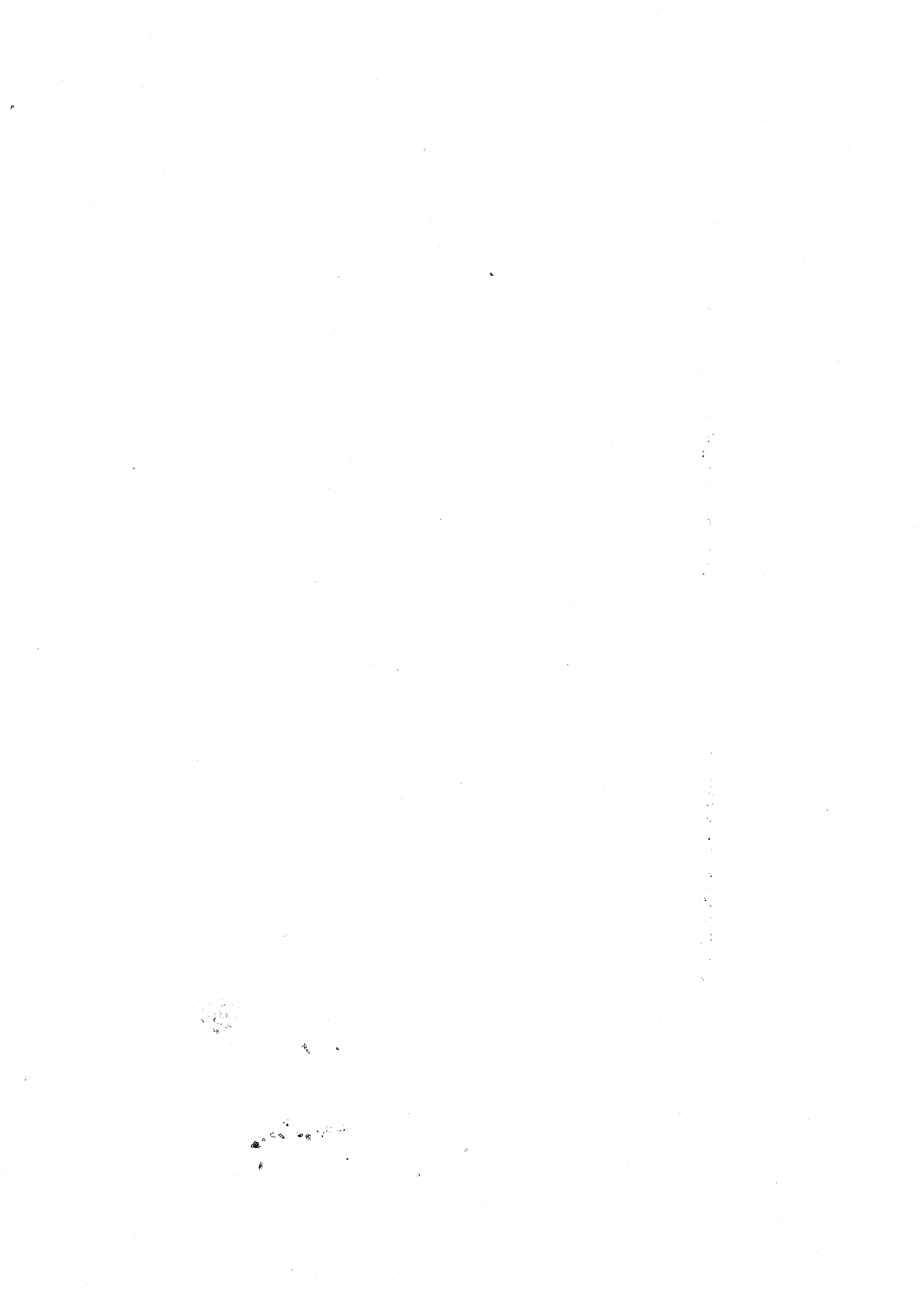
##### .2) Le module contrôle

- .1) La mémoire interne
  - .1) La mémoire morte du logiciel système
  - .2) La mémoire de travail
- .2) Le circuit contrôleur du périphérique opérateur
- .3) Le circuit contrôleur d'interruptions

#### 2.) Le bloc périphérique

- .1) Le module interface
- .2) Le module mémoire
- .3) Le module d'extensions des entrées/sorties





## ANNEXE I

### LE PROCESSEUR CENTRAL: P.central

#### 1.) LE BLOC PROCESSEUR.

##### 1.1) Le module de traitement

##### 1.1.1) Le microprocesseur MC6800: rappel du fonctionnement

Le MC6800 est un microprocesseur monolithique constituant le centre de la famille 6800. Compatible en entrée/sortie avec la technologie TTL, il ne requiert qu'une seule alimentation(+5V). Il peut adresser 64K-octets de mémoire; son BUS de données est de 8 bits bidirectionnel.

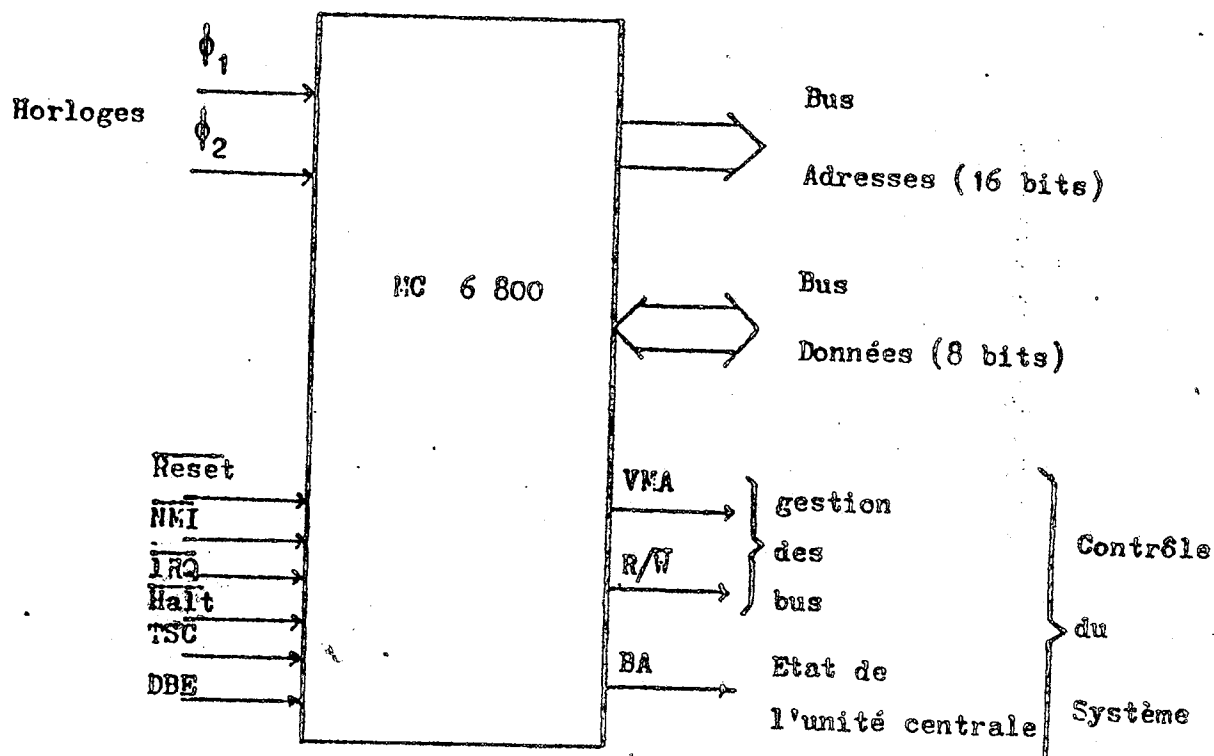


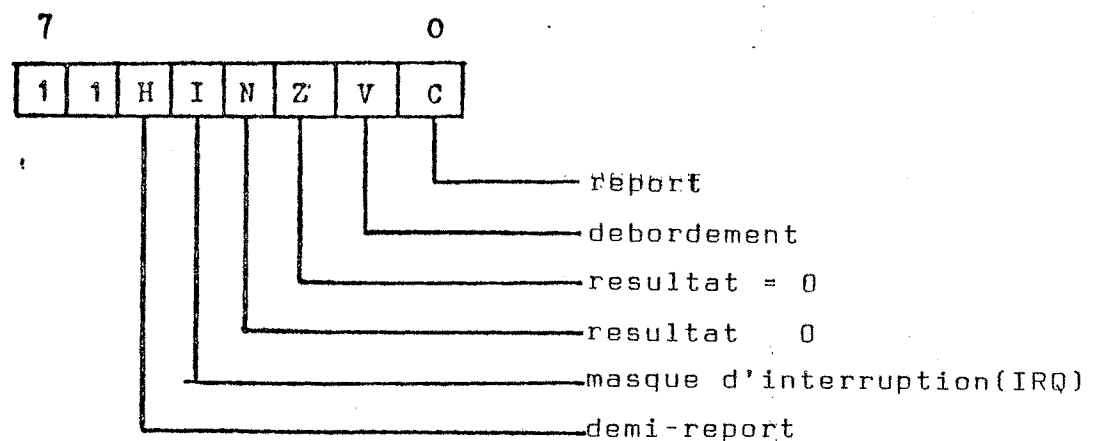
Figure A1.1 La structure externe du MC6800

a) Architecture du MC6800.

Le MC6800 possède 6 registres internes:

- un compteur ordinal(Program Counter) de 16 bits
- un registre d'index(Index Register) de 16 bits
- un pointeur de pile(Stack Pointer) de 16 bits
- deux accumulateurs(A et B) de 8 bits chacun
- un registre d'état(Condition Code Register)

Le registre d'état de 8 bits contient 6 indicateurs organisés de la manière suivante:



b) Signaux externes au MC6800.

- Valid Memory Address(VMA).

Ce signal indique aux organes extérieurs qu'une adresse valable est présentée sur le BUS d'adresse. Cette sortie est couramment utilisée pour activer les organes extérieurs.

Elle est active à l'état haut et n'est jamais en haute impédance.

- Read/Write(R/ $\bar{W}$ ).

Le signal R/ $\bar{W}$  est une sortie qui indique aux mémoires et aux périphériques si le MC6800 est en lecture(R/ $\bar{W}$ =1) ou en écriture(R/ $\bar{W}$ =0). Le niveau normal de ce signal est l'état haut.

Il peut être mis en haute impédance par le signal TSC, ou quand le microprocesseur est en arrêt(HALT).

- Bus Available(BA).

Ce signal est normalement à l'état 0. Quand il est à l'état 1, il indique que le microprocesseur est arrêté et que le BUS d'adresse est utilisable par un autre organe.

Ceci arrive quand le microprocesseur est en arrêt par HALT ou quand il exécute une instruction d'attente (WAIT).

- Tri-State Control(TSC).

Quand TSC est à l'état haut, le BUS d'adresse et le signal R/ $\bar{W}$  sont en haute impédance. Les signaux VMA et BA sont forcés à 0 de manière à éviter de fausses lectures ou écritures sur les organes activés par le signal VMA.

Quand TSC vaut 1, les phases du microprocesseur ( $\phi 1$  et  $\phi 2$ ) doivent être maintenues respectivement haute et basse, dans le but de retarder l'exécution du programme. Le signal TSC est utilisé dans les applications d'accès direct à la mémoire (DMA) de courte durée.

- Data Bus Enable (DBE).

DBE est le contrôle de tri-state du Bus de données. Quand il est à l'état haut, les amplificateurs d'attaque du BUS (DRIVERS) sont activés. Cette entrée est compatible TTL. En général, elle est activée par la phase  $\phi 2$ .

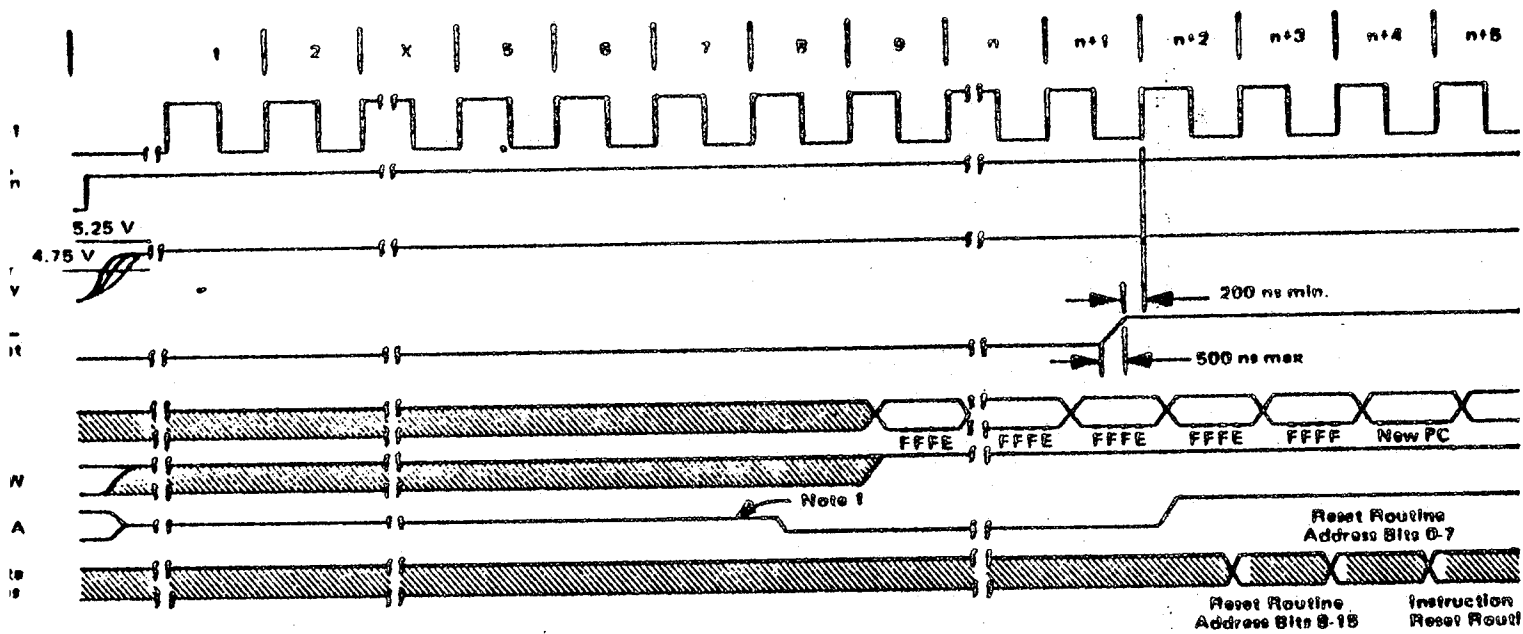
c) Signaux d'interruptions.

Le microprocesseur possède 4 autres entrées de contrôle: RESET, NMI, IRQ, HALT.

- RESET.

Cette entrée est utilisée pour démarrer le microprocesseur à la mise sous tension, ou lors d'un mauvais fonctionnement. Les deux dernières cases mémoires (FFFFE-FFFFF) contiennent l'adresse de la première instruction du programme soit système (moniteur) soit utilisateur.

Cette entrée peut être utilisée pour réinitialiser le microprocesseur à n'importe quel moment. La figure A1.2 illustre le comportement du MC6800 lors d'une mise sous tension ou d'un RESET.



NOTE 1: Midrange waveform indicates high impedance state.

Figure A1.2 Les signaux de démarrage du MC6800.

- Non Maskable Interrup(NMI).

Le MC6800 peut prendre en compte 2 types d'interruptions:  $\overline{\text{IRQ}}$  qui est masquable par un bit du registre d'état(CCR), alors que  $\overline{\text{NMI}}$  ne l'est pas. La prise en compte interne de ces interruptions par le microprocesseur est la même ( NMI marche sur front et IRQ sur niveau), excepté qu'elles n'ont pas le même vecteur d'adresse. Le comportement du microprocesseur sous interruption est résumé par la figure A1.4.a

L'interruption est prise en compte à la fin du cycle en cours. L'instruction suivante n'est pas exécutée; le contexte du microprocesseur est rangé dans la pile (successivement:PC,IX,A,B,CCR).

Le bit de masque d'interruption est alors positionné. L'adresse de la première instruction du programme "interruption" est cherché dans les cases- mémoire FFFC-FFFF pour  $\overline{\text{NMI}}$  et FFF8-FFF9 pour  $\overline{\text{IRQ}}$ .

- HALT.

La ligne  $\overline{\text{HALT}}$  permet le contrôle de l'exécution d'un programme par une source extérieure. Quand  $\overline{\text{HALT}}$  passe à 0, le MC6800 s'arrête; BA passe à 1, indiquant que toute activité est stoppée.

Le BUS d'adresse, des données et le signal  $\text{R}/\overline{\text{W}}$  sont en haute impédance. VMA est à 0, de manière à ce que le système n'active aucun organe. Si une interruption arrive, elle est enregistrée, et sera prise en compte quand le microprocesseur sortira du mode  $\overline{\text{HALT}}$ . La figure A1.3 montre l'effet de  $\overline{\text{HALT}}$  sur le microprocesseur.

Pour tester des programmes, il est intéressant de les exécuter instruction par instruction. Pour cela, HALT doit être maintenu haut et retomber comme indiqué à partir du point A sur la figure A1.3.

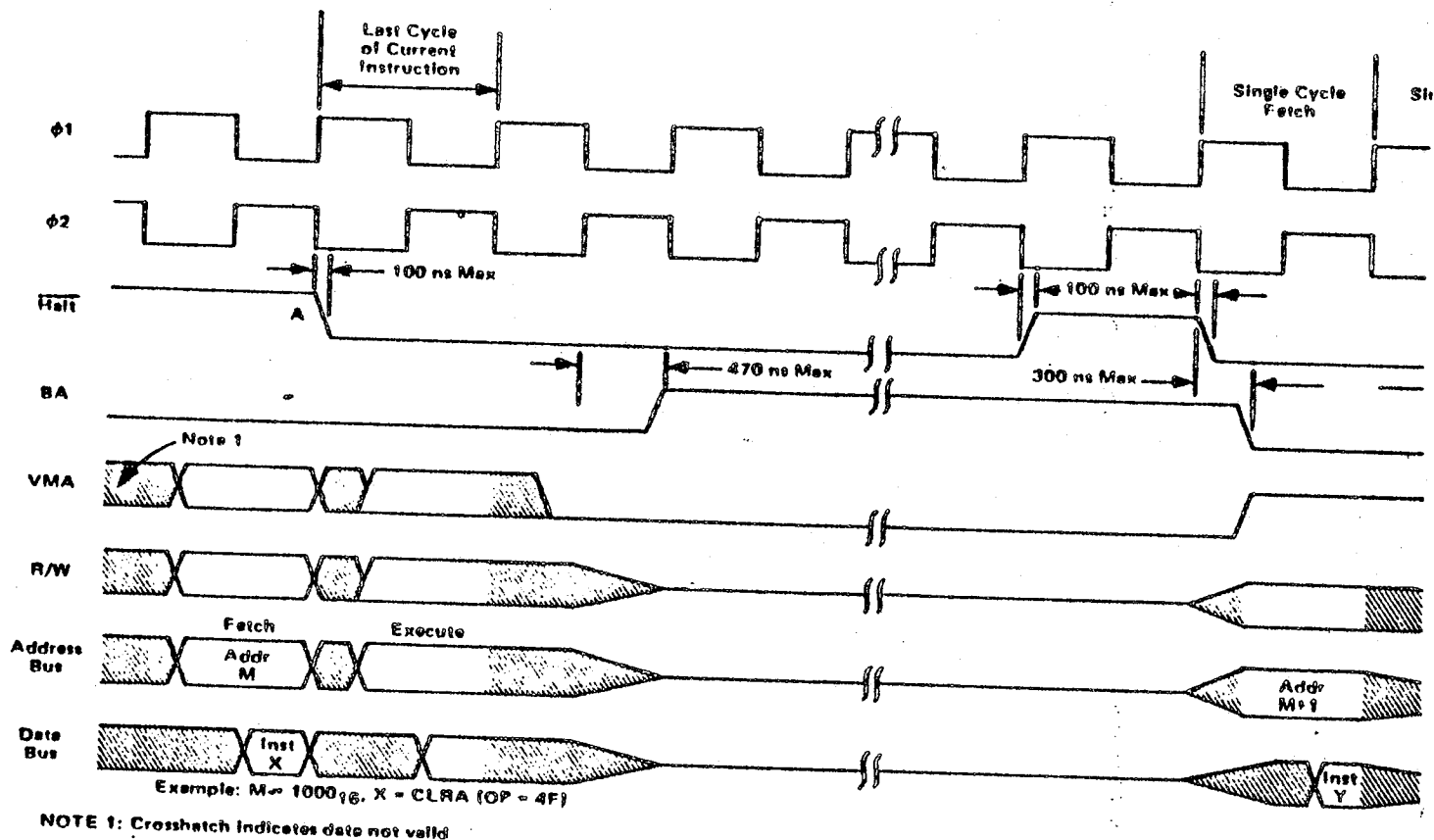


Figure A1.3 Fonctionnement pas à pas.

#### d) Le jeu d'instructions (RV1-77).

Le jeu d'instructions du MC6800 est caractérisé par un grand nombre d'instructions à référence mémoire (additions, soustractions, opérations logiques, décalages, ..). Un autre de ses avantages est que les branchements vers les sous-programmes sont faits automatiquement. Le contexte d'exécution est rangé dans la pile sans intervention du programmeur.



Quelques intructions méritent une plus grande attention:

- Software Interrup(SWI).

L'instruction SWI amène le microprocesseur à exécuter entièrement la procédure d'acquittement d'une interruption, comme si celle-ci était due à une cause externe. Ceci permet de stopper un programme.

- Wait For Interrup(WAI).

Quand le MC6800 exécute l'instruction WAIT, il ne fait rien d'autre qu'attendre une interruption. Le contexte est sauvegardé avant la réception de l'interruption ce qui accélère sa prise en compte.

L'usage le plus courant de l'instruction WAI est le transfert de données en DMA. Le BUS étant flottant, tous les registres étant sauvegardés; le cycle de la DMA peut durer aussi longtemps que l'on désire. Il suffira, à la fin du transfert, de générer une interruption. Le microprocesseur reprendra alors le contrôle des BUS.

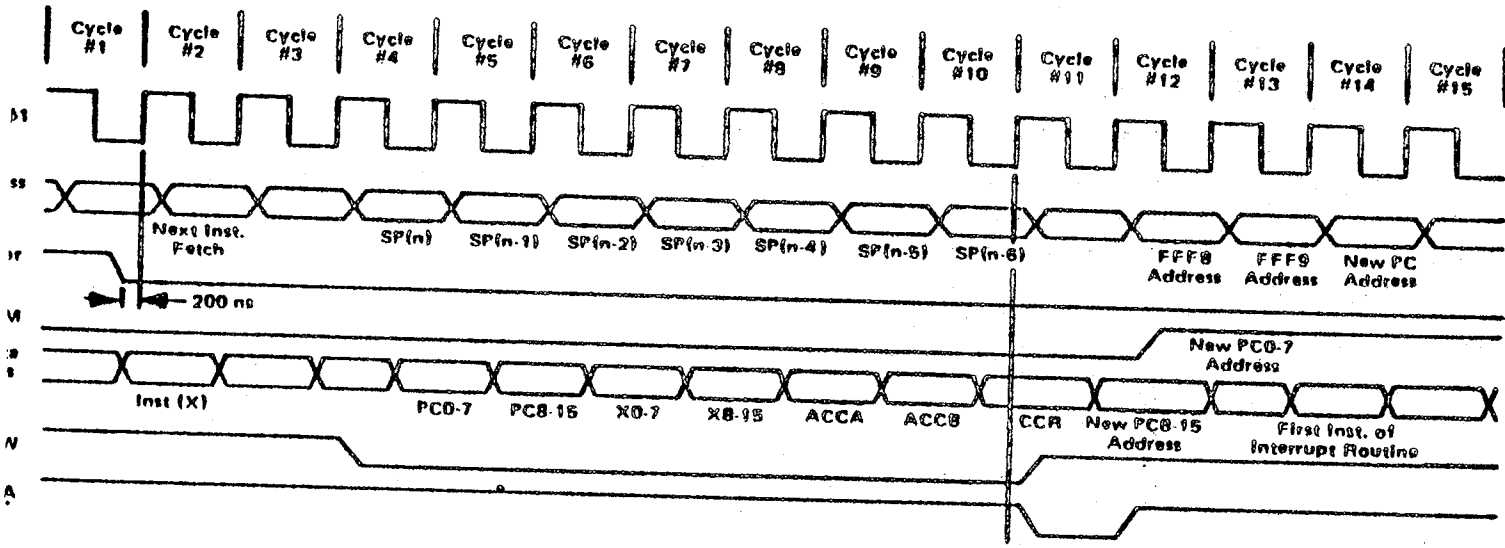


Figure A1.4a Diagramme des temps pour les interruptions.

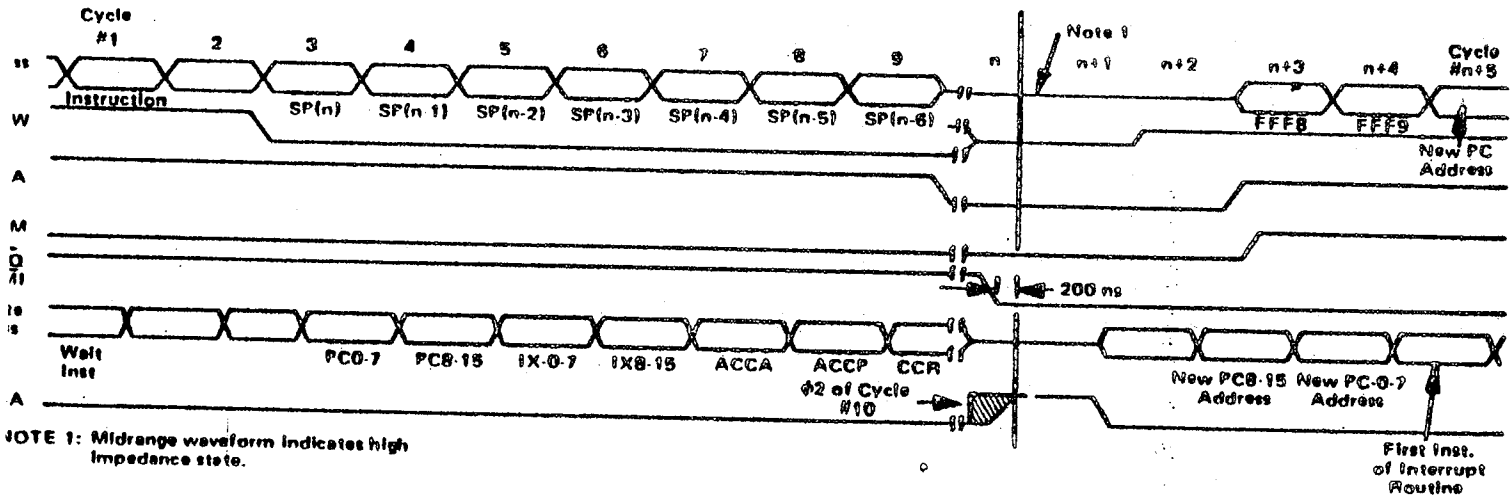


Figure A1.4b Diagramme des temps pour l'instruction WAIT.

### 1.1.2) L'horlogerie centrale.

L'horlogerie centrale est organisée autour d'un circuit MOTOROLA (MC6871B) qui génère les phases de travail du MC6800 ( $\emptyset 1$  et  $\emptyset 2$ ) et des signaux de contrôle propres au MC6800. Il génère également les signaux de synchronisation des composants du P.central.

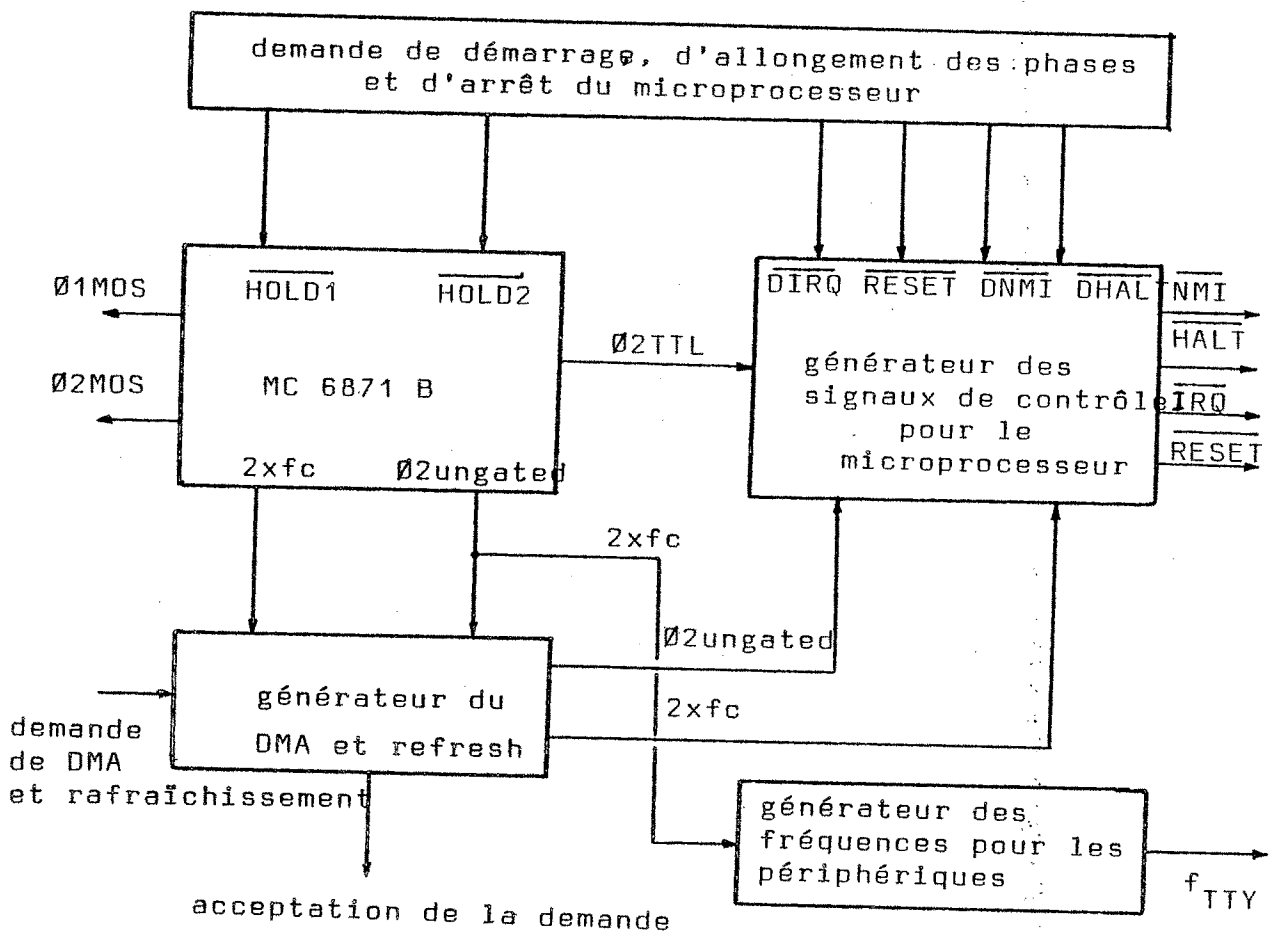


Figure A1.5 L'horlogerie centrale

### 1.1.2.1) L'horloge principale: MC68710

#### - Caractéristiques des signaux d'entrée et de sortie du MC68710

\* HOLD2: Demande d'allongement de la phase Ø2, pour l'accès aux périphériques lents et aux mémoires lentes (temps d'accès à l'information > du cycle de lecture/écriture du microprocesseur) pour effectuer des lectures ou écritures.

\* HOLD1: Demande de retarder l'exécution du microprocesseur (allongement de la phase Ø1)

\* Ø2-UNGATED: Ce signal, en sortie, correspond à la phase Ø2; elle n'est pas perturbée par les demandes HOLD2 et HOLD1.

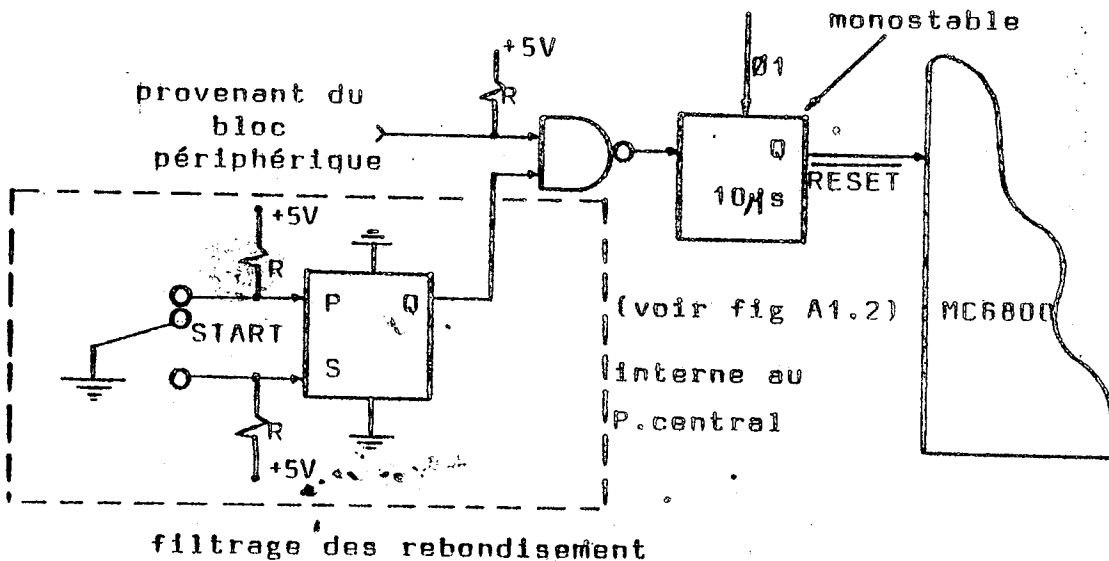
\* Ø2TIL: Ce signal correspond à la phase Ø2TIL.

\* Ø1M0s, Ø2M0s: Ce sont les signaux de travail du MC6800

\* 2\*fc: deux fois la fréquence de l'horloge principale.

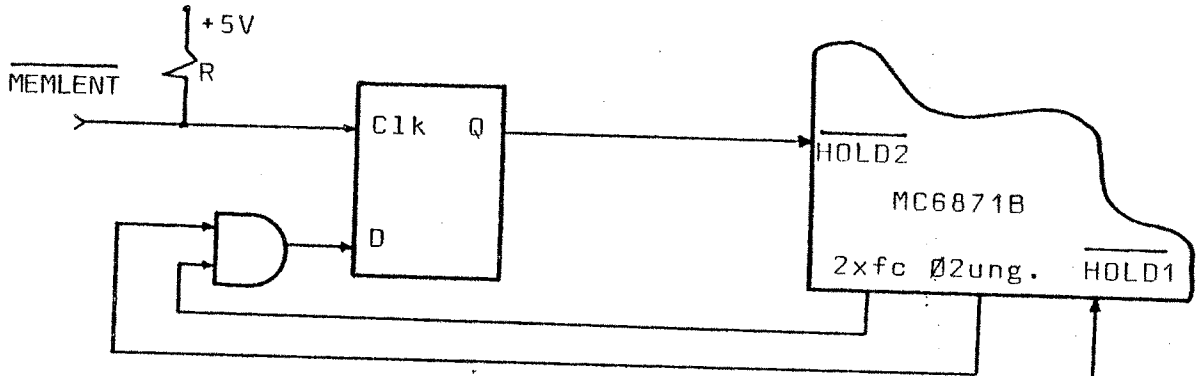
#### - Demands de démarrage, d'allongement des phases et d'arrêt du microprocesseur.

\* demande de démarrage.

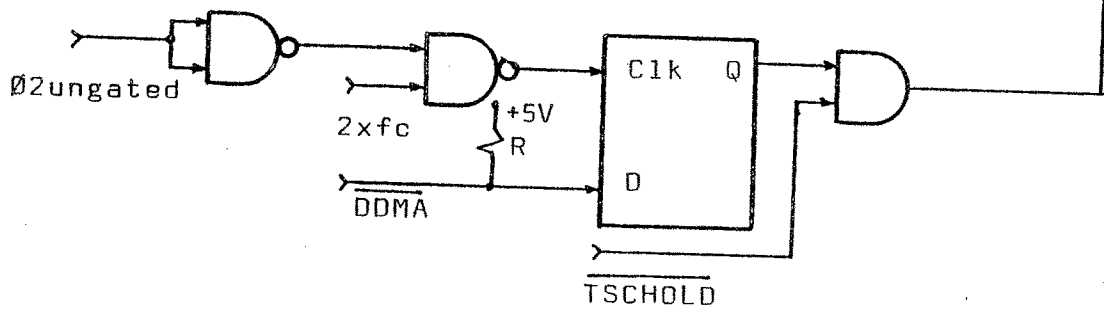


\* demande d'allongement.

a) cas des mémoires lentes



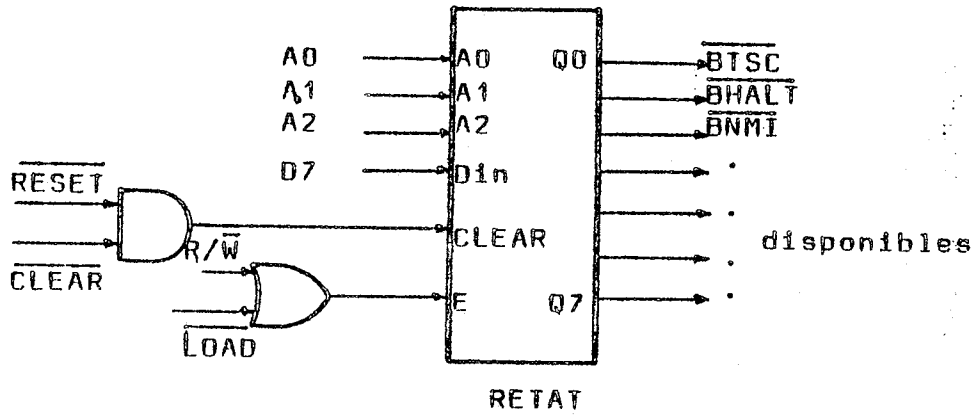
b) cas du rafraichissement ou de transfert en DMA



DDMA: demande de rafraichissement ou d'accès direct à la mémoire.

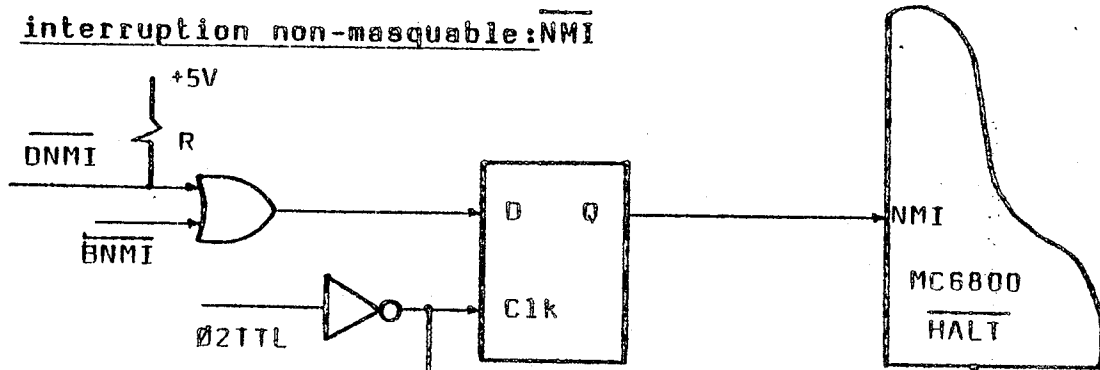
- Demandes d'interruption du microprocesseur.

Grâce à un registre d'état (RETAT), nous pouvons masquer, par le logiciel système, l'arrivée des demandes d'interruption (NMI, HALT, TSC) au MC6800. Le démarrage du MC6800 permet de démasquer ces interruptions.

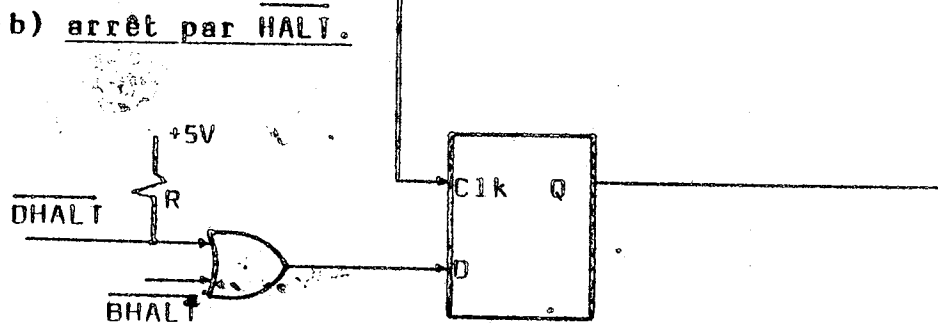


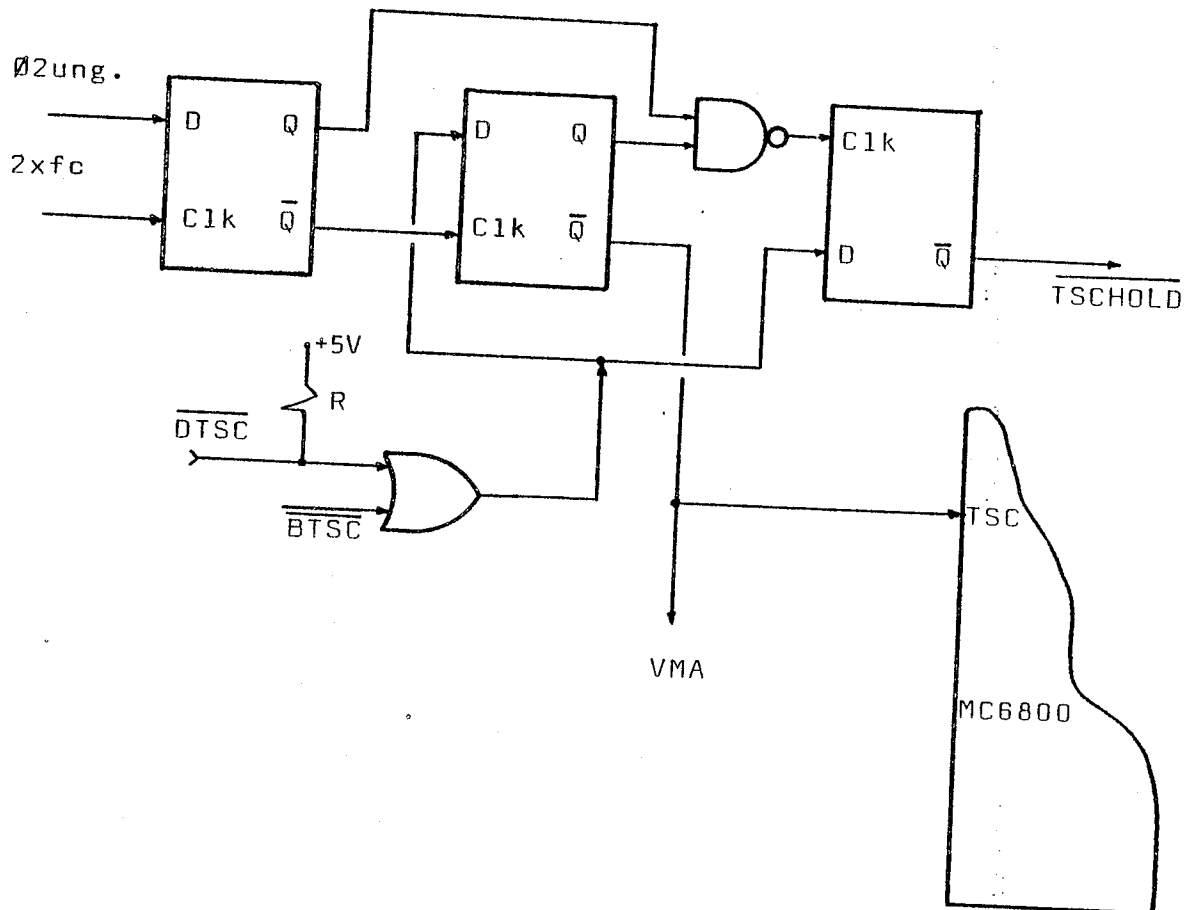
\* synchronisation des demande d'interruption.

a) interruption non-masquable: NMI



b) arrêt par HALT.



c) arrêt temporaire: TSC

### 1.1.3) L'espace d'adressage du P.central.

L'espace d'adressage du P.central est divisé en 16 pages de 4K-octets par page. Sa réalisation matérielle est faite par un circuit multiplexeur(4\*16) du type SN74154 qui génère le BUS de pages.

#### 1.1.3.1) L'espace système.

Cet espace est représenté par l'espace où se trouve le vecteur machine du microprocesseur (c.f.annexe II) ; dans notre cas c'est la dernière page( $\overline{PF}$ ) car le vecteur machine du MC6800( $\overline{RESTART}$ ,  $\overline{NMI}$ ,  $\overline{IRQ}$ ,  $\overline{SWI}$ ) s'y trouve(R11-77).

Dans cet espace, sont connectés les différents composants du bloc processeur; par exemple la mémoire morte, le circuit interface avec la console opérateur, la mémoire de travail,...

#### 1.1.3.2) L'espace utilisateur.

Tout espace en dehors de l'espace système est considéré comme espace utilisateur; c'est-à-dire qu'il peut être utilisé pour les applications utilisateur.



Remarque.

Dépendant de l'application du P.central, l'espace système peut prendre plusieurs pages; ces pages doivent être continues et situées au dessus de la dernière page ( $\overline{PF}$ ).

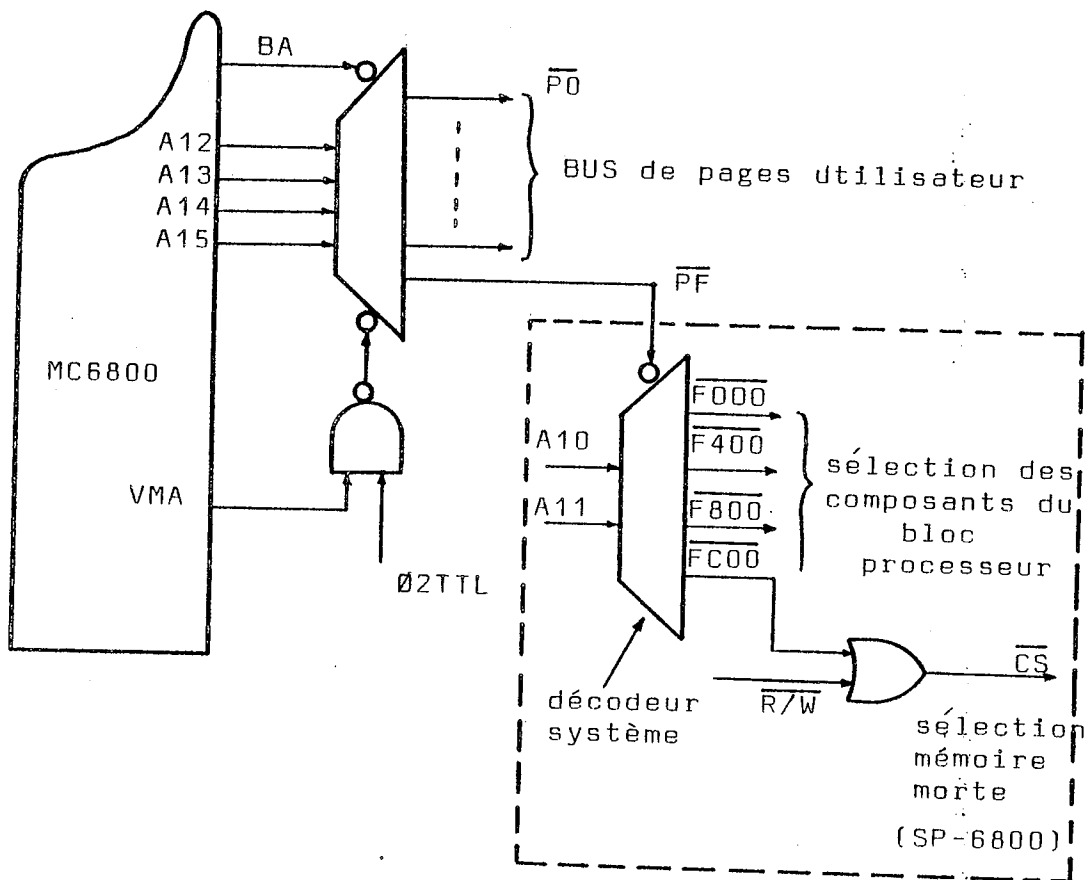


Figure A1.6 L'espace d'adressage du P.central

## 1.2) LE MODULE CONTROLE.

### 1.2.1) La mémoire interne.

#### 1.2.1.1) La mémoire morte: voir Annexe II

#### 1.2.1.2) La mémoire de travail: MC6810

Cette mémoire placée à l'adresse(F800-F87F) sert comme variable de travail et aussi comme pile du système. Elle est organisée en 128 octets.

### 1.2.2) Le circuit périphérique opérateur.

C'est un composant qui permet d'établir une communication avec la console opérateur (télé-imprimeur, visu,...). La liaison avec cette console est une communication standard du type RS-232C. Elle est assurée par les circuits suivants:

- une unité de contrôle de transmissions asynchrone(ACIA:MC6850)
- un circuit générateur de fréquences propres à la console opérateur (MC14411) provenant de l'horlogerie centrale.
- un circuit amplificateur pour la transmission des caractères et des signaux de contrôle du lecteur de ruban (MC1488).
- un circuit de mise en forme de caractères reçus (MC1489).

Le signal de réception des caractères n'est pas relié au mécanisme d'interruption du microprocesseur. L'attente de l'émission ou de la réception se fait donc à l'aide de boucles d'attente.

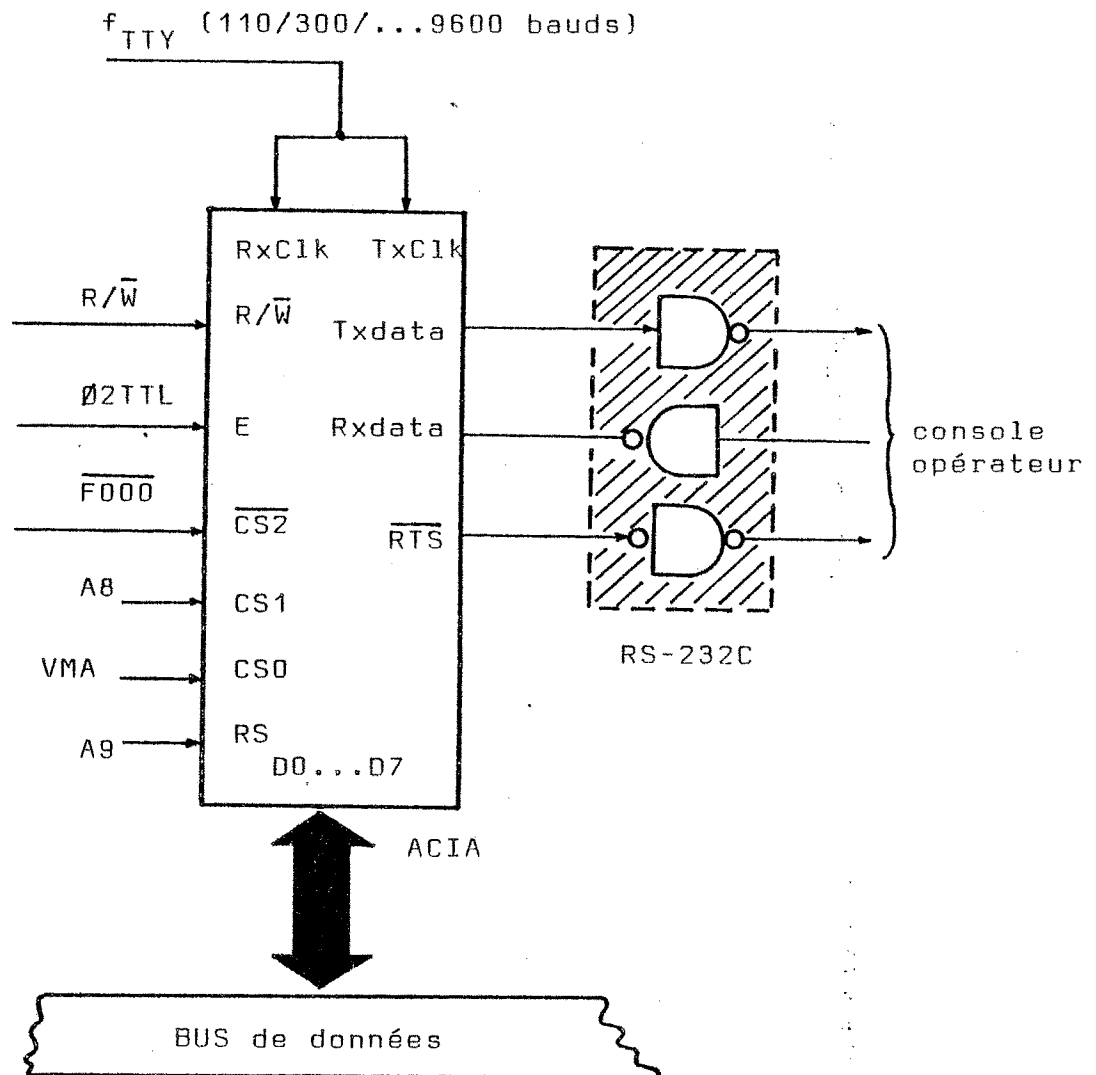


Figure A1.7 Connexion de l'ACIA au système.

### 1.2.3) Le contrôleur des interruptions.

Le contrôle des interruptions de type IRQ, se fait grâce à un circuit 8214INTEL qui permet de choisir une interruption parmi huit avec un ordre de priorité fixé précédemment par le système.

Le microprocesseur peut lire le numéro du signal d'entrée (A0,A1,A2) qui l'interrompt (interruption étant déjà acceptée par le MC6800). Ce numéro est utilisé par le logiciel système(SP-6800) pour indexer dans un tableau dit "d'interruption" le vecteur(IIi) qui contient l'adresse d'exécution du programme utilisateur d'interruption (PIIi).

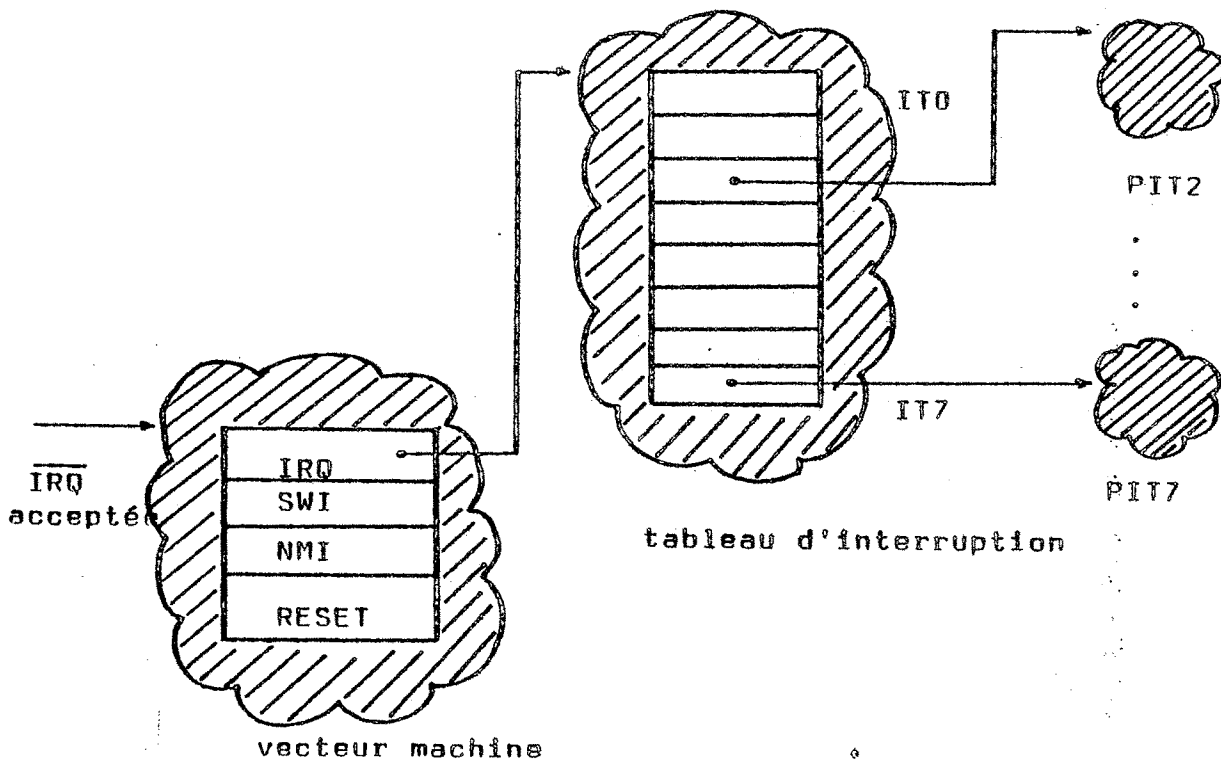


Figure A1.8 Le déroulement d'une interruption IRQ.

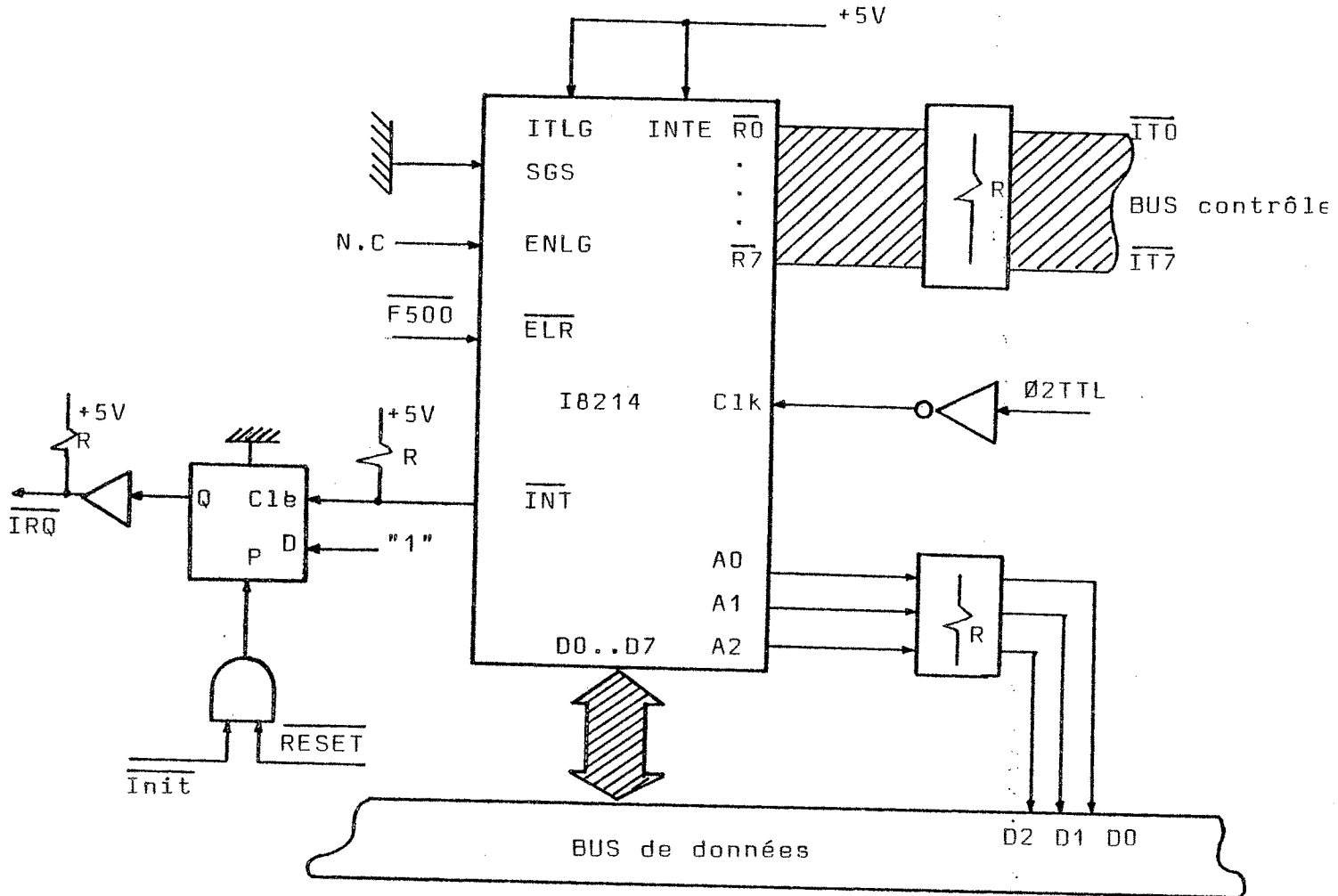


Figure A1.9 Le circuit contrôleur des interruptions.

## 2.) LE BLOC PERIPHERIQUE.

### 2.1) Le module interface.

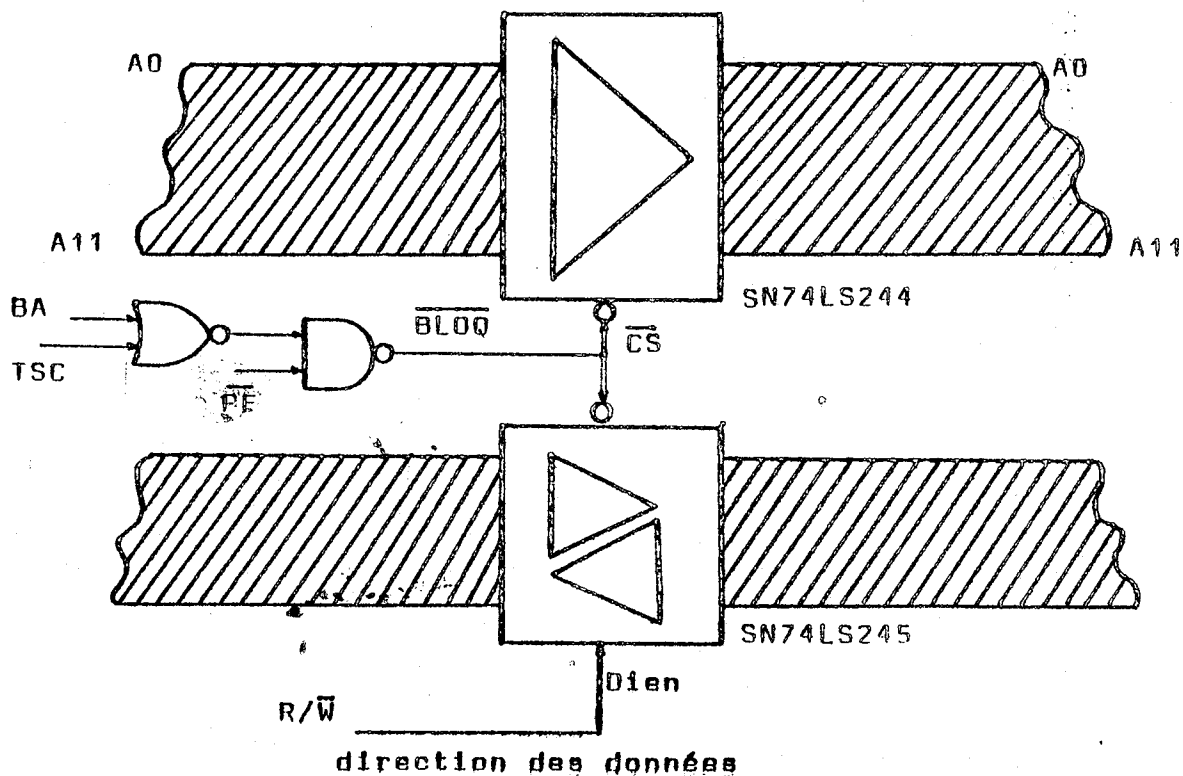
#### 2.1.1) L'interface adresse.

Cette interface est conçue par des circuits à trois états du type SN74LS244; chaque circuit contient huit amplificateurs unidirectionnels.

#### 2.1.2) L'interface données.

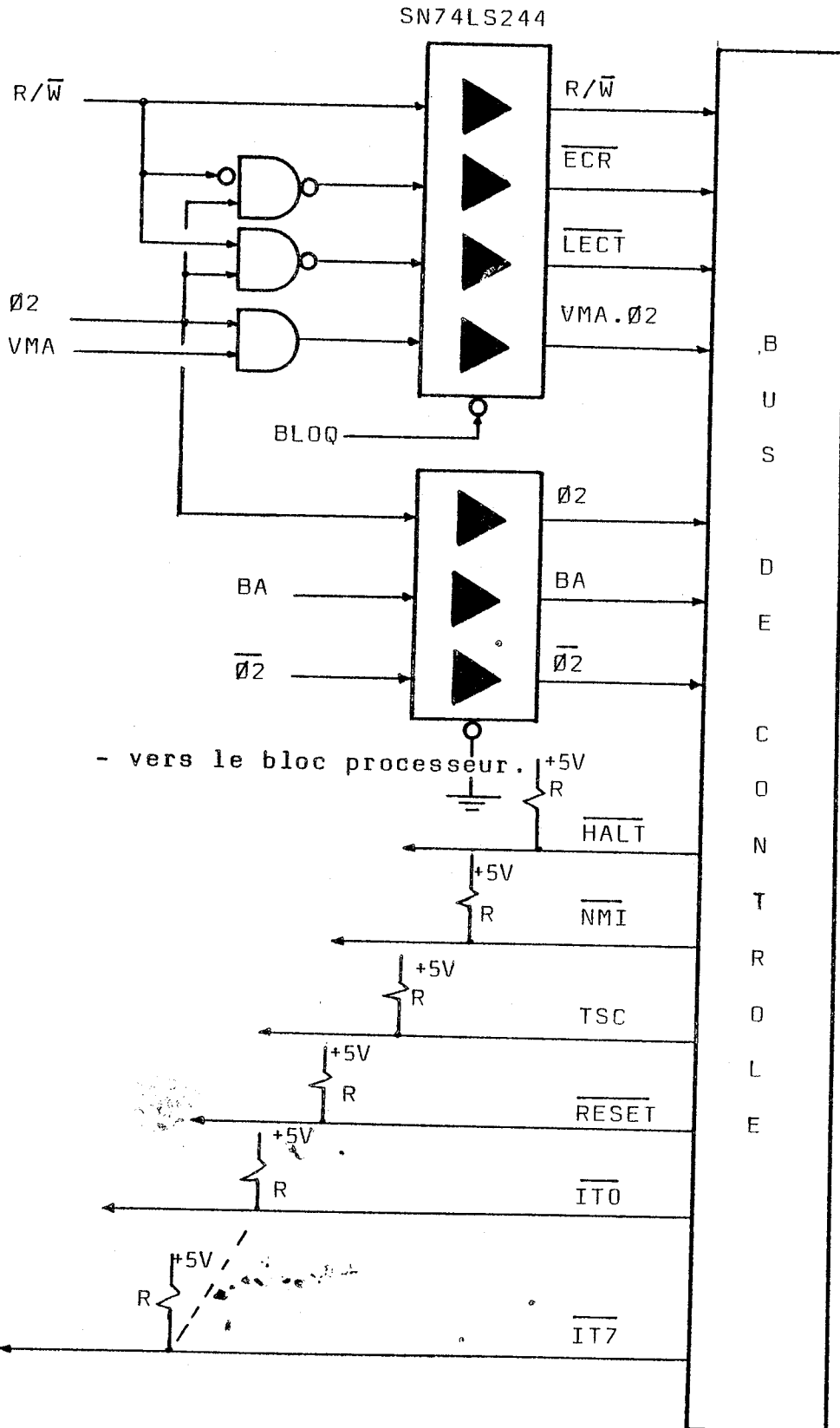
L'interface pour les données est conçue par un circuit à trois états du type SN74LS245; ce circuit est un amplificateur bidirectionnel.

La fonction principale du module interface est d'isoler le bloc processeur du bloc périphérique quand une interruption d'arrêt au microprocesseur est acceptée ( $\overline{\text{HALT}}$ , TSC) et aussi quand la page système est sélectionnée ( $\overline{\text{PF}}$ ).



2.1.3) L'interface contrôle.

- vers le bloc périphérique.



## 2.2) Module mémoire.

Le module mémoire est composé de mémoires du type 2114 à 200ns, organisées en 1K\*4bits chacune. Ce module a une capacité de 16K-octets (4 pages) et est placé de l'adresse 0000 à l'adresse 3FFF (figure A1.10).

## 2.3) Module extension des entrées/sorties.

Ce module (figure A1.11) contient, en plus d'une circuiterie générale (décodeurs, horloges, circuits d'interface RS-232C,..), deux interfaces de communication série (MC6850) et un logiciel d'extention du moniteur système qui permet:

- de contrôler ces interfaces (initialisation, programmation de leurs vitesses de travail,..)
- d'établir des liaisons soit:
  - \* en mode "transparent": la console operateur du P.central est vue de l'exterieur comme un console utilisateur.
  - \* en mode "chargeur": le P.central a la fonction de stocker l'information provenant de ces interfaces (P1,P2).
  - \* en mode "multiplexeur": le P.central peut
    - / établir des dialogues entre ces interfaces,
    - / envoyer des profils quelconques vers ces interfaces,
    - / recevoir des informations provenant de chaque interface.



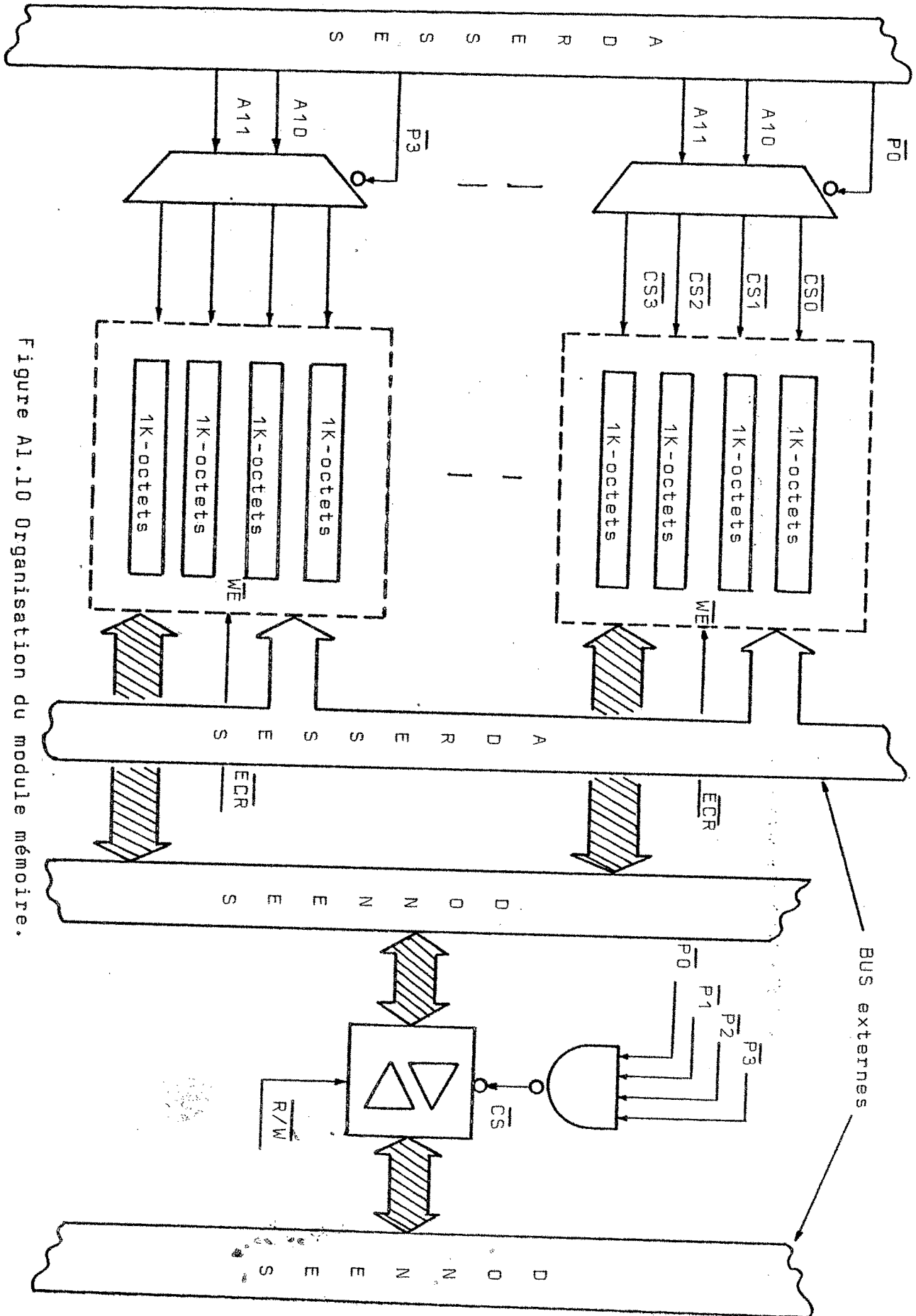


Figure A1.10 Organisation du module mémoire.

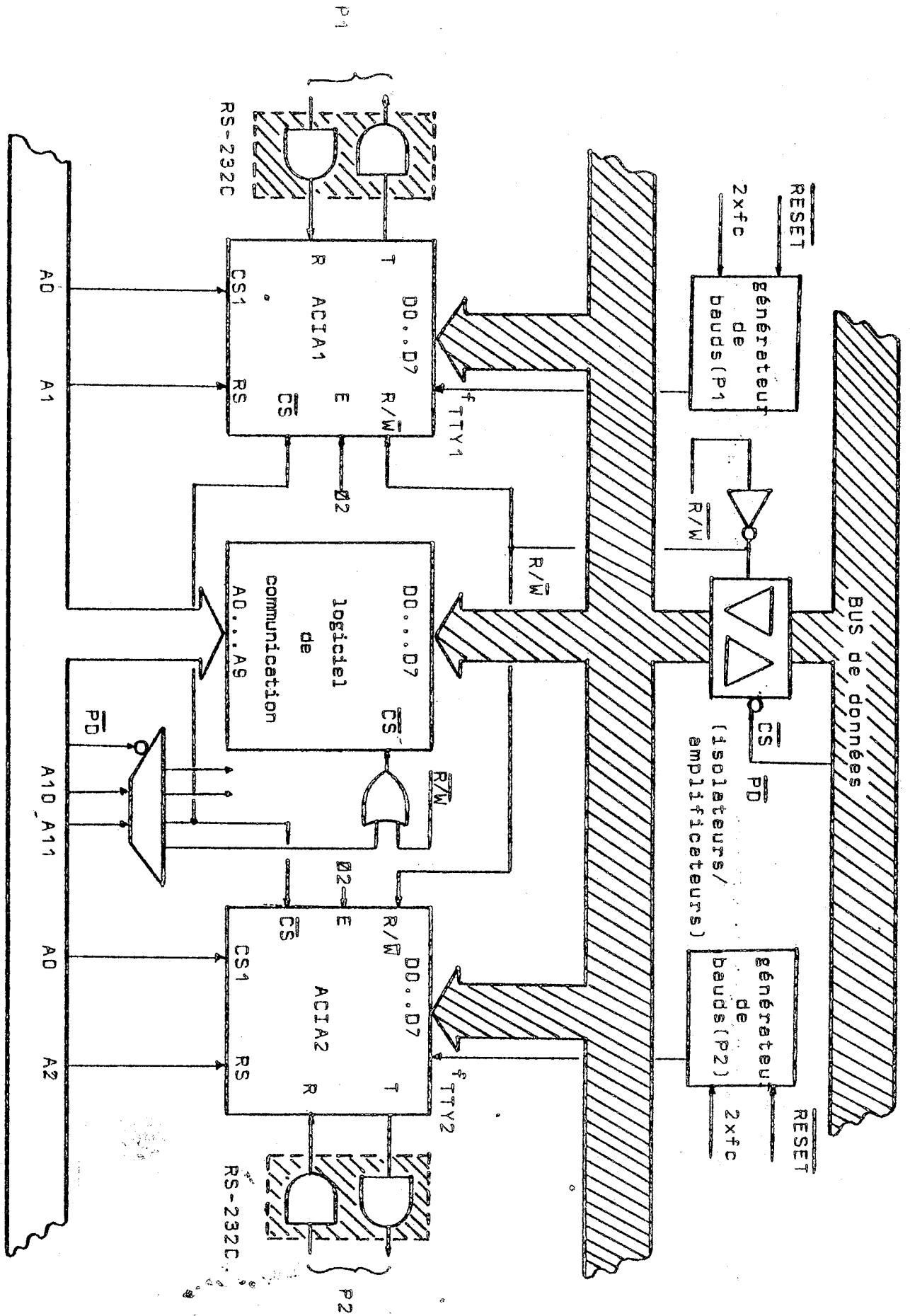
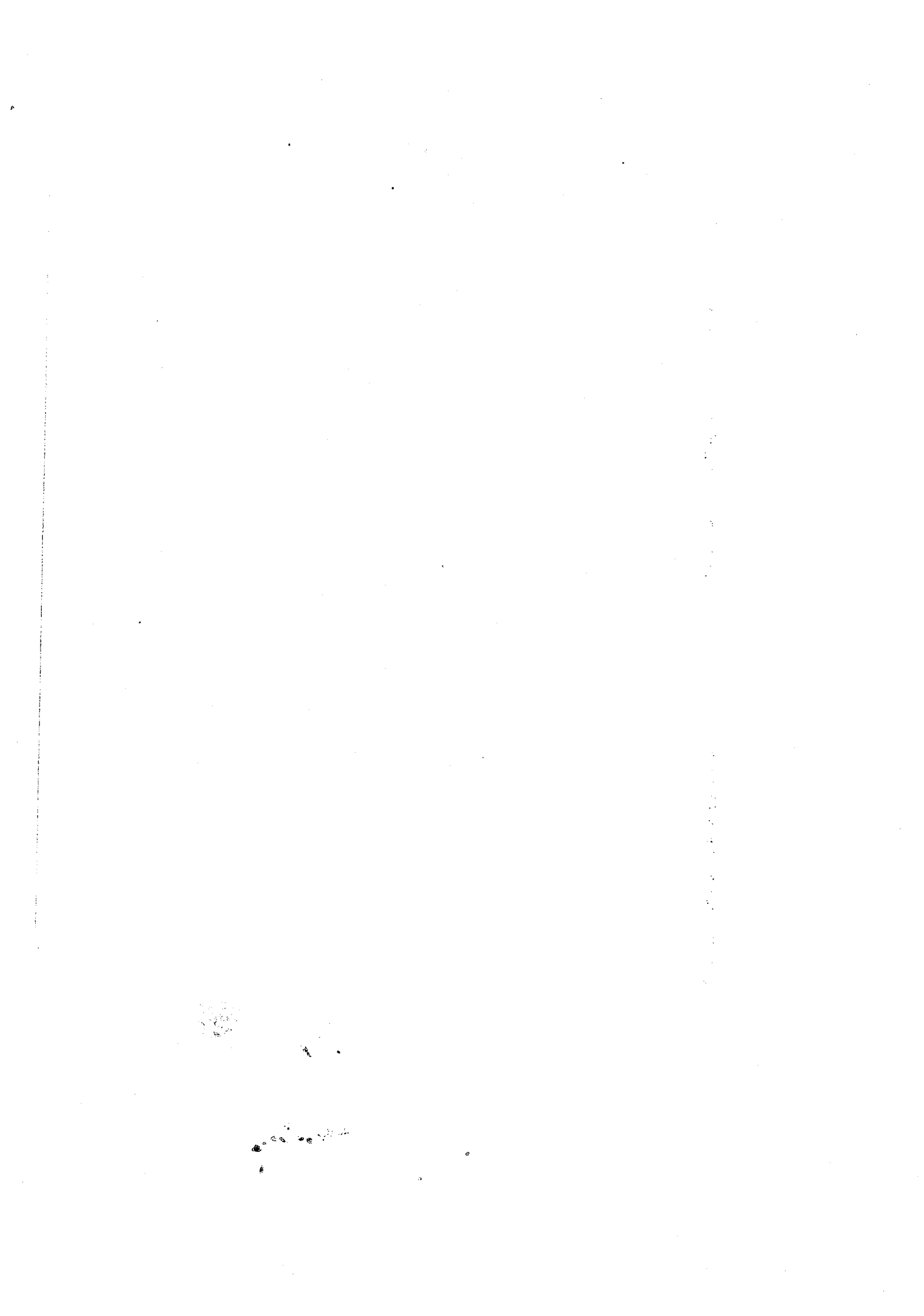
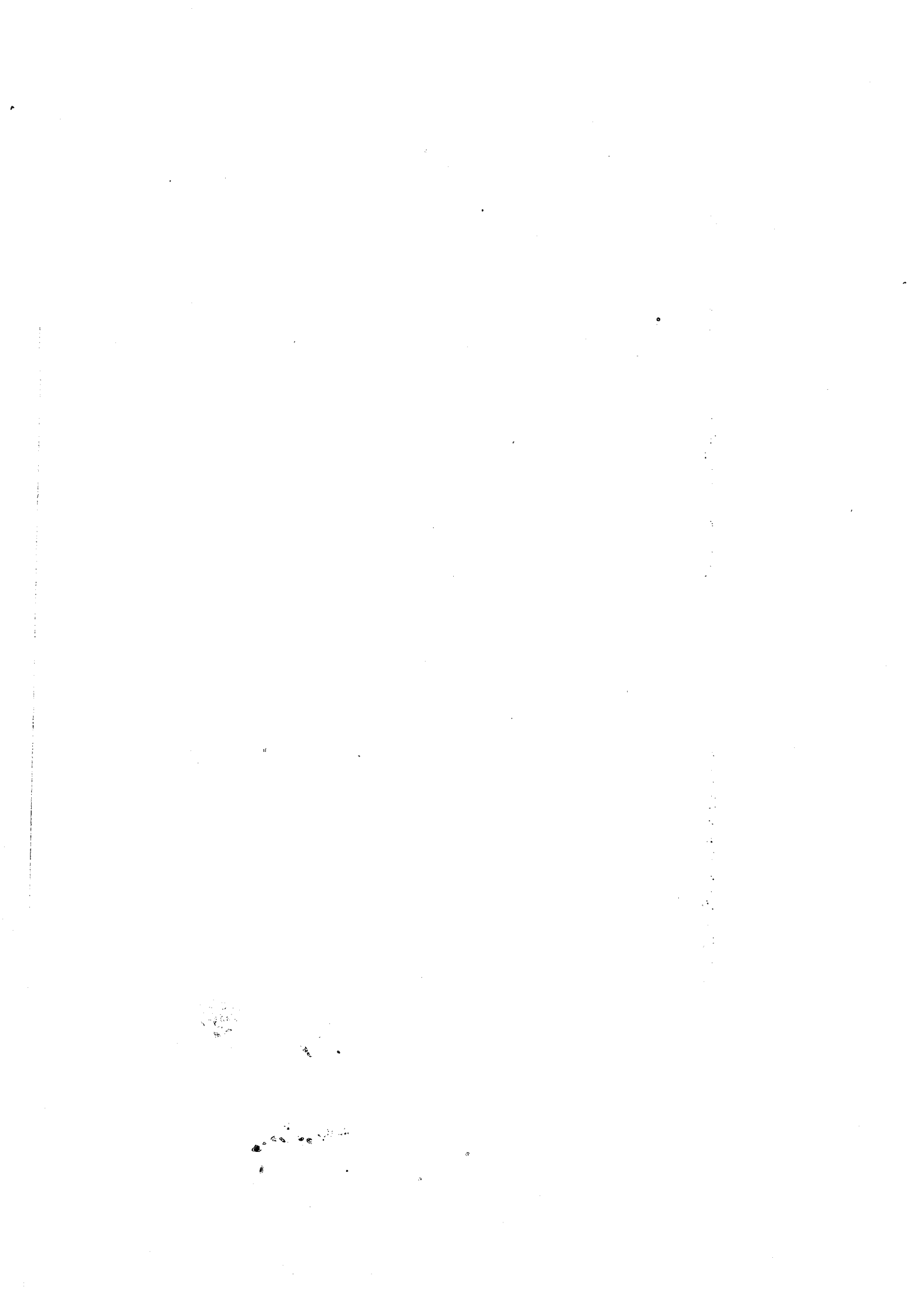


Figure A1.11 Organisation du module d'extention des entrées/sorties.



## ANNEXE II

- 1.) Introduction
- 2.) Fonctions du superviseur
  - .1) Les commandes
    - .1) Commandes normales
    - .2) Commandes en mode interrompu
- 3.) Sous-programmes de service
- 4.) Utilisation de la mémoire morte
  - .1) Organisation de l'espace mémoire
  - .2) L'interface du superviseur avec la console opérateur
  - .3) Mécanismes de commutation
    - .1) Mécanisme matériel
    - .2) Mécanisme logiciel
5. Listing du programme moniteur SP-6800



SUPERVISEUR SP-6800  
DE MISE AU POINT ET DE MAINTENANCE "IN SITU"  
D'APPLICATIONS MICROPROCESSEURS

1.) INTRODUCTION.

La mise au point et la maintenance des applications organisées autour d'un microprocesseur MC6800, peuvent être grandement facilitées par l'utilisation d'un système superviseur installé, à demeure, sur l'application.

Les quelques circuits consacrés à cette fonction permettent l'économie d'outils complexes de mise au point des microprocesseurs, en fournissant "in situ" les moyens nécessaires.

Son utilisation ne nécessite qu'un téléimprimeur standard (ou un organe de visualisation) et des appareils standard de laboratoire (oscilloscope).

Un outil a été conçu pour répondre à ce besoin. Il est constitué d'une mémoire morte, programmée avec le superviseur SP-6800 conçu par l'équipe de recherche en Architecture d'Ordinateurs de l'Ecole Nationale Supérieure d'Informatique et de Mathématiques Appliquées de Grenoble (ENSIMAG) et amélioré par la SESCOSEM

Cet outil est conçu pour pouvoir cohabiter de plusieurs manières possibles avec les programmes de l'application en fonction des performances désirées.

Il sait se mettre en service par le simple positionnement d'une clef ou d'un commutateur DIL, pouvant être placé sur la carte même de l'application. Il fournit par ailleurs à l'utilisateur, l'accès à tout un ensemble de sous-programmes utilitaires qui allègent son travail de programmation pour les fonctions de dialogue utilisant le téléimprimeur.

## 2.) FONCTIONS DE SUPERVISION.

Il offre les fonctions classiques de supervision suivantes:

- examen et modification du contenu d'une adresse; cette fonction permet l'examen et la modification de l'état de la mémoire ainsi que l'examen et l'excitation des organes d'entrée-sortie;
- demande d'examen et de modification du contenu de la prochaine ou précédente adresse;
- perforation d'une zone de mémoire en format chargeable;
- lancement d'un programme;
- pose de points d'arrêt;
- entrée impromptue dans le superviseur;
- examen de l'état du processeur, registre (A, B, C, X, S, P) après la rencontre d'un point d'arrêt ou après une entrée impromptue;
- retour au point d'interruption;
- élimination des points d'arrêts.

Cet outil contient également des fonctions spécifiques à la maintenance :

- génération de signaux d'écriture ou de lecture périodiques pour la mise au point de dispositifs périphériques.
- test de mémoire vive,

## 2.1) Les commandes

Au moment de l'activation du système (RESTART), le superviseur répond par le caractère '\*'. Dès cet instant, il est prêt à recevoir des commandes. Les tableaux A1 et A2 décrivent les commandes et leur signification; il y en a de deux sortes:

- commandes générales,
- commandes en mode interrompu.

### 2.1.1) Commandes générales.

#### a) Chargeur de ruban.

Cette commande permet de charger un ruban dans la mémoire vive du microprocesseur à partir du téléimprimeur.

Le chargeur contrôle le début et la fin de la lecture du ruban, il ignore tous les caractères différents de ceux de contrôle situés au début du ruban.

Ces caractères de contrôle sont vérifiés au début de chaque fichier, puisqu'un ruban peut être composé de plusieurs fichiers concaténés qui se terminent par un caractère S9.



Si, pendant le chargement, on détecte des erreurs provoquées par :

- calcul de la somme logique (CS),
- caractères non hexadécimaux,
- adresse physique non existante ou en panne, un message d'erreur sera écrit sur le téléimprimeur.

Exemple :

\*L

DE07 ? erreur à l'adresse DE07 ce que signifie que le superviseur n'a pas réussi à écrire une information à cette adresse.

Dans le cas où l'on désire ignorer l'erreur (cas de somme logique incorrecte), on frappe la commande (L) et le "chargeur" repart pour charger le fichier suivant sur le ruban.

Exemple :

\*L

000000E45FE3ABCDFF caractères ignorés

S106017D01146FF7

                  somme logique

S1040180C9B1

S9

\*

Le chargeur finira son travail en revenant à l'interprétation des commandes dès qu'il trouve le caractère de contrôle S9.

Le format est décrit dans la figure A2.1.

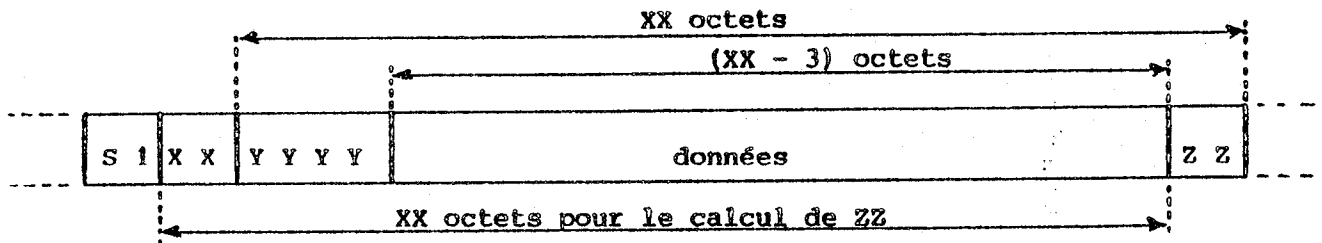


Figure A2.1 Format des données en entrée du chargeur  
FORMAT MOTOROLA

XX = nombre d'octets de l'adresse, des données, et de la somme logique.

YYYY = adresse de chargement du premier octet de donnée

ZZ = somme logique de contrôle. C'est le résultat de la complémentation de la somme (modulo 256) des XX octets qui suivent S1.

Remarque :

Avant S1 et après ZZ peut apparaître n'importe quelle suite de caractères ne contenant pas S1 qui définit le début d'un nouveau bloc et S9 qui définit sa fin.

Tous les caractères d'un bloc doivent être des caractères hexadécimaux (0 ... 9, A ... F).

b) Impression/perforation du contenu d'une zone de mémoire.

Cette procédure permet à l'utilisateur :

- d'imprimer le contenu d'une zone de mémoire,
- de perforer sur un ruban le contenu de cette zone dans le format du chargeur.

Le format de la procédure est le suivant :

<dump>::= <commande><zone mémoire>(RC)

<commande>::= (P)

<zone mémoire>::= <adr> <adr1>

<adr>::= <adr1>::= <hexa4>

<hexa4>::= (4 caractères hexadécimaux)

Exemple :

\* P 0012 0043

S11300120000CC0000CC5D504D42CF400000CC40EB somme logique  
S1 .....

S10500423D2358S9

fin bloc

\*

La fin de la commande signalée par l'impression du caractère '\*' montre que le programme est bien perforé. Ce ruban est directement chargeable par la commande L.

La frappe d'un caractère quelconque arrête l'impression en revenant à l'analyse des commandes.

c) Modification/examen du contenu d'une adresse.

Cette commande permet, d'une part, l'examen du contenu d'une adresse et d'autre part, sa modification. Son format est le suivant:

```

<modif> ::= <commande> <hexamod>
<commande> ::= (M)
<hexamod> ::= <adresse><contenu><contrôle>
<adresse> ::= (adresse de lecture)
<contenu> ::= (=)<lecture>
<lecture> ::= <hexa2>
<contrôle> ::= <retour superviseur>/
                <modification>/
                <demande d'examen ou modification de la
                précédente ou prochaine adresse >

<retour superviseur> ::= <retour chariot>
<modification> ::= <octet à modifier><contenu>
<octet à modifier> ::= <hexa2>
<demande d'examen ou modification de la précédente
ou prochaine adresse> ::= <incré.décré.adr><RC><hexamod>
<incré.décré.adr> ::= <inc.adr> / <décr.adr>
<décr.adr> ::= (U) CO <décr.adr> ::= <adresse-1>;
<inc.adr> ::= (N) CO <inc.adr> ::= <adresse+1>;
<RC> ::= <retour chariot>
<retour chariot> ::= (RC) ;
<hexa2> ::= (2 caractères hexadécimaux);

```

Exemple :

\*M 1000 = 11(RC)

\*M 1200 = 44 11 = 11(RC)

└─ vérification modification

\*M 1300 = 11(U) demande le contenu de l'adresse  
précédente

12FF = 44 33 = 33(RC)

\*M 1340 = 44(N)

1341 = 55(N) demande le contenu de la prochaine  
adresse

1342 = 66(N)

1342 = 77(RC)

\*

Une modification impossible peut provenir :

- d'une erreur de lecture,
- d'une erreur d'écriture,
- de l'accès à un organe en lecture seule,
- de l'accès à un organe périphérique,
- de l'accès à aucun élément.

L'annulation d'une commande se fait par la frappe d'un  
retour chariot.

Exemple :

\*M(RC)

\*

c) Lancement de programme.

Cette commande lance l'exécution d'un programme à une adresse donnée; son format est le suivant :

<GO> ::= <commande><adr>;

<commande> := (G) ;

Exemple :

\*G 1000.

d) Mise en arrêt (WAIT).

Cette commande (\* W) provoque la mise en arrêt du microprocesseur pour permettre une intervention sous tension exemple: extraction d'une carte.

Le retour au superviseur après cette commande se fait : soit par RESTART, soit par une interruption NMI.

e) Test électronique de lecture/écriture.

Ce test permet de générer périodiquement des signaux (au niveau électronique) de :

- sélection,
- lecture/écriture (R/ $\bar{W}$ ) à des adresses choisies.

Le format est le suivant :

<test> ::= <commande><adr><contrôle>

<commande> ::= (T)

<contrôle> ::= <lecture>/<écriture>

<lecture> ::= (R)

<écriture> ::= (W) <octet à écrire>

Le retour au superviseur après cette commande se fait par la frappe d'un caractère quelconque.

Exemple :

\*T 1000 R .. boucle de lecture périodique à l'adresse  
1000

\*T 1000 W AA boucle d'écriture périodique de AA à  
l'adresse 1000.

f) Test mémoire.

Le format de cette commande est le suivant :

<test mémoire> ::= <commande><adr><adrl>(RC)

<commande> ::= (E)

<adr> ::= (adresse de départ)

<adrl> ::= (adresse de fin)

Ce test écrit différentes configurations en mémoire, puis les relit. En cas de non correspondance, il imprime l'adresse, la valeur que l'on a voulu écrire et celle lue.

La frappe d'un caractère quelconque arrête l'impression en revenant à l'analyse des commandes.

Cette commande détruit le contenu de la zone mémoire spécifiée.

2.1.2) Commandes en mode interrompu: après un point d'arrêt ou un appel superviseur par NMI.

a) Examen/modification du contenu des registres.

Ces commandes permettent de connaître et de modifier l'état des registres de la machine; le format de cette commande est le suivant:

$\langle \text{modreg} \rangle ::= \langle \text{commacu} \rangle / \langle \text{commreg} \rangle (\text{RC})$   
 $\langle \text{commacu} \rangle ::= \langle \text{accu} \rangle \langle \text{état accumulateur} \rangle$   
 $\langle \text{accu} \rangle ::= (\text{A}) / (\text{B})$   
 $\langle \text{état accumulateur} \rangle ::= (=) \langle \text{contenu de l'accumulateur} \rangle$   
 $\langle \text{contenu de l'accumulateur} \rangle ::= \langle \text{hexa2} \rangle$   
 $\langle \text{commreg} \rangle ::= \langle \text{registres} \rangle \langle \text{état registres} \rangle$   
 $\langle \text{registres} \rangle ::= (\text{C}) / (\text{S}) / (\text{X}) / (\text{Z});$  CO C est le compteur ordinal et Z le code condition de la machine;  
 $\langle \text{état registres} \rangle ::= (=) \langle \text{contenu du registre} \rangle$   
 $\langle \text{contenu du registre} \rangle ::= \langle \text{hexa4} \rangle$

Exemple :

\*A = 12(RC)  
 \*B = DA(RC)  
 \*X = 1200(RC)  
 \*C = FC00(RC)  
 \*S = 0200(RC)  
 \*A = 34 AA(RC)  
 \*S = 0300 0190(RC)



b) Pose de points d'arrêt.

Cette commande permet de placer des points d'arrêt dans les programmes utilisateurs rangés dans des RAM.

Elle permet de placer au maximum 16 points d'arrêts; la technique utilisée consiste à remplacer le code opération de l'instruction placée à l'adresse d'arrêt choisie, par un code de software interrupt(3F) pour provoquer une entrée dans le superviseur lors de l'exécution.

L'entrée dans le superviseur, consécutive au passage sur un point d'arrêt, remet en place l'octet de code de l'instruction du programme située à cette adresse et permet l'analyse de l'état du microprocesseur par les commandes normales du superviseur.

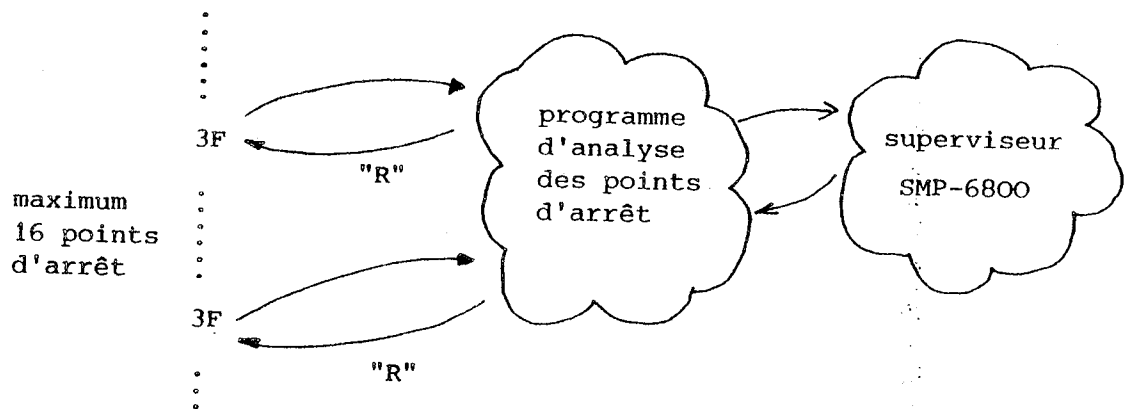


Figure A2.2 Le mécanisme des points d'arrêts.

Le format de cette commande est le suivant :

```
<break> ::= <commande> <arrêt>;
<commande> ::= (K);
<arrêt> ::= <adresse d'arrêt> ::= <adr>;
```

Exemples.

```
* K 1EE0
* K 1EE5 pose de points d'arrêt
* G 1ED8 lancement du programme, exemple
```

1EE0 ..... adresse d'arrêt

vérification état de la machine

```
* Z = A1(RC)
* C = 1EE0(RC)
* X = 1EF8(RC)
* B = ED(RC)
* R ..... retour à l'exécution du programme
```

1EE5 ..... adresse d'arrêt

```
* A = 0F AA(RC) .. modification du contenu de A
* R ..... retour à l'exécution du programme
```

c) Elimination des points d'arrêts: \* F

d) Retour d'interruption.

Cette commande (\*R) permet de reprendre une exécution suspendue par un point d'arrêt ou une requête NMI.



TABLEAU A2. Commandes en mode interrompu

- \* Z = CC(RC)
- \* A = AA(RC)
- \* B = BB(RC) ..... examen du contenu des registres
- \* S = <adr>(RC)
- \* X = <adr>(RC)
- \* C = <adr>(RC)
  
- \* A = AA CC(RC)
- \* B = BB yy(RC)
- \* X = <adr> <adr1>(RC) ..... modification du contenu des  
registres
- \* S = <adr> <adr1> <adr2>....<adrn>(RC)
- \* C = <adr> <adr1>(RC)
- \* R .... retour d'interruption

3.) SOUS-PROGRAMMES.

LHEX: lecture d'un chiffre hexa

adresse: FC00

sortie: A

modification: aucun registre modifié

LOCT: lecture de deux caractères hexadécimaux

adresse: FC2C

sortie: A

modification: aucun registre modifié

CADR: construction d'une adresse sous forme de 4 caractères hexadécimaux

adresse: FC4E

sortie: X

modification: A

IMOC: impression d'un octet sous la forme de 2 caractères hexadécimaux

adresse: FCBA

entrée: A

modification: aucun registre modifié

PHEX: impression d'un chiffre hexadécimal

adresse: FCAA

entrée: A

modification: B

LCAR: lecture d'un caractère

adresse: FCD7

sortie: A

modification: aucun registre modifié

PCAR: impression d'un caractère

adresse: FCE5

entrée: A

modification: aucun registre modifié

RICH: retour chariot et saut de ligne

adresse: FCCA

modification: A

REST: reset et initialisation de l'ACIA

adresse: FCFF

INIT: entrée superviseur par RESTART

adresse: FD32

NMI: entrée superviseur par NMI

adresse: FD3B

IMCR: impression d'une chaîne de caractères pointée par l'index et finie par le chiffre hexadécimal 04.

adresse: FF48

#### 4.) UTILISATION DE LA MEMOIRE MORTE.

##### 4.1) Organisation de l'espace mémoire.

Avant de voir les différentes possibilités d'utilisation du superviseur au niveau logiciel et au niveau matériel, nous allons présenter les différentes zones de la mémoire qu'il utilise.

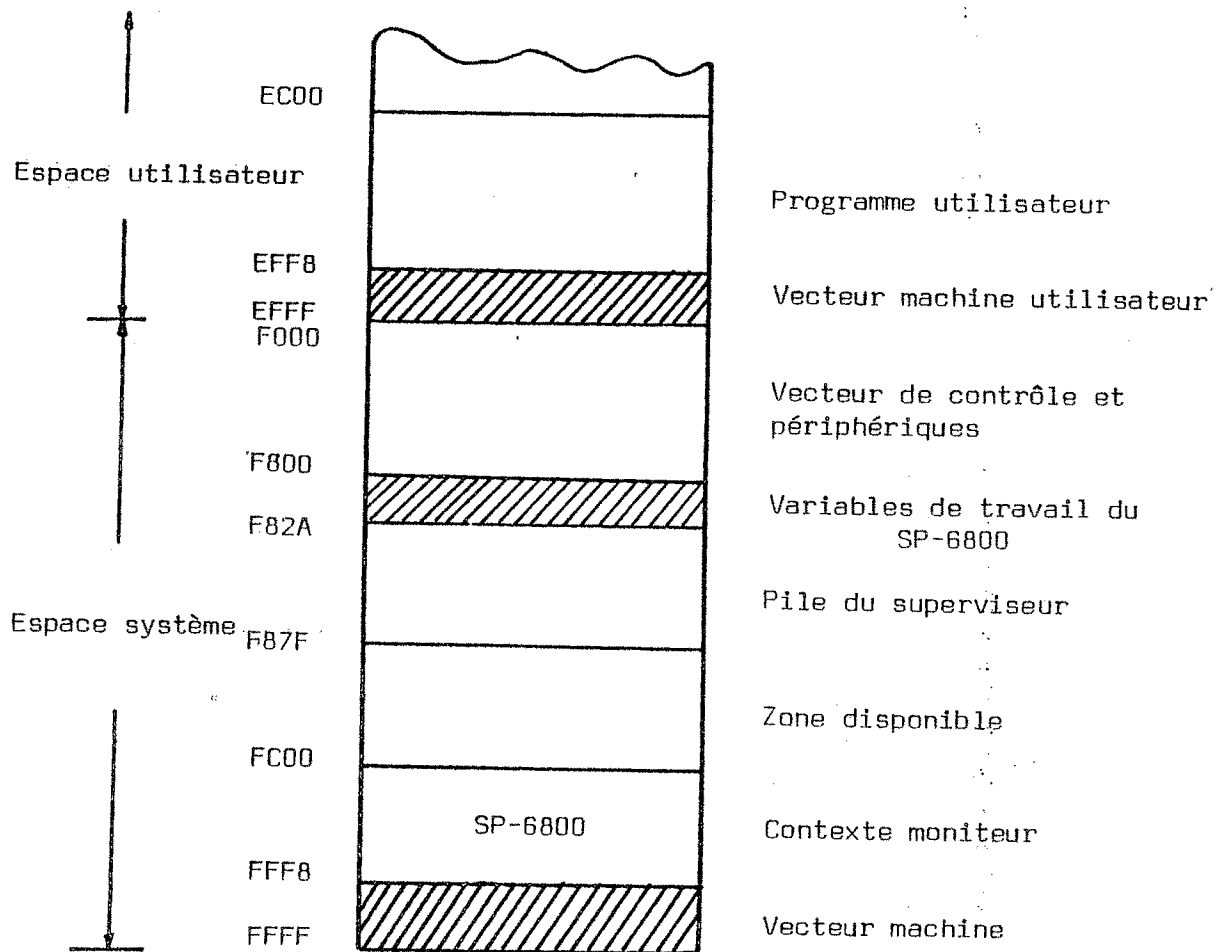


Figure A2.3. Organisation de l'espace mémoire.

Cette zone se décompose en:

a) Vecteur machine: FFFB-FFFF

Ce vecteur contient l'adresse de:

- lancement initial(RESTART),
- interruption non masquable(NMI),
- interruption logicielle(SWI),
- interruption normale(IRQ).

Lors de l'initialisation du système, le processeur lira dans les deux octets de lancement initial, l'adresse du point d'entrée dans le superviseur. En cas d'interruption, il lira l'adresse correspondante dans le vecteur machine, puis déroulera les programmes associés.

c) Octet de protection.

Cet octet de sécurité(AA), placé à l'adresse F82A, est utilisé pour protéger la zone de travail du superviseur des erreurs de l'utilisateur (destruction accidentelle).

d) Mode de travail superviseur/utilisateur: adresse F008

Le choix du mode de travail correspond à la lecture de l'état d'une clef (COM) (figure A2.4):

- mode superviseur: D7=1
- mode utilisateur: D7=0



e) Vecteur utilisateur: EFF8-EFFF

Dans cette zone, l'utilisateur bâtit un vecteur d'adresse (RESTART, NMI, SWI, IRQ) semblable, vis-à-vis de l'application, au vecteur machine situé en FFF8-FFFF.

4.2) Interface du superviseur avec le téléimprimeur.

Le superviseur utilise un circuit SFF 96850 pour la connexion du téléimprimeur constituant la console opérateur.

L'adresse des mots des données (F300) et d'état (F200) de l'ACIA constitue le vecteur périphérique.

L'horloge de contrôle de l'ACIA est générée à partir du circuit MC14411 qui donne en sortie une gamme de fréquences entre 1740Hz et 9600Hz (voir Annexe I: implantation de l'ACIA).

#### 4.3) Mécanismes de commutation.

Nous analyserons deux mécanismes permettant le passage de l'activité entre le superviseur et les programmes d'application, et réciproquement.

Ces mécanismes sont :

- logiciel,
- matériel.

##### 4.3.1) Mécanisme logiciel.

Le démarrage se fait normalement sur l'adresse de lancement initial du vecteur machine qui donne le contrôle au superviseur (figure A2.4), celui-ci teste ensuite la clef COM pour :

- en mode utilisateur : passer le contrôle aux programmes d'application par adressage indirect sur l'adresse RESTART du vecteur utilisateur.
- en mode superviseur : entrer dans le superviseur et initialiser le circuit périphérique (ACIA) et le pile système.

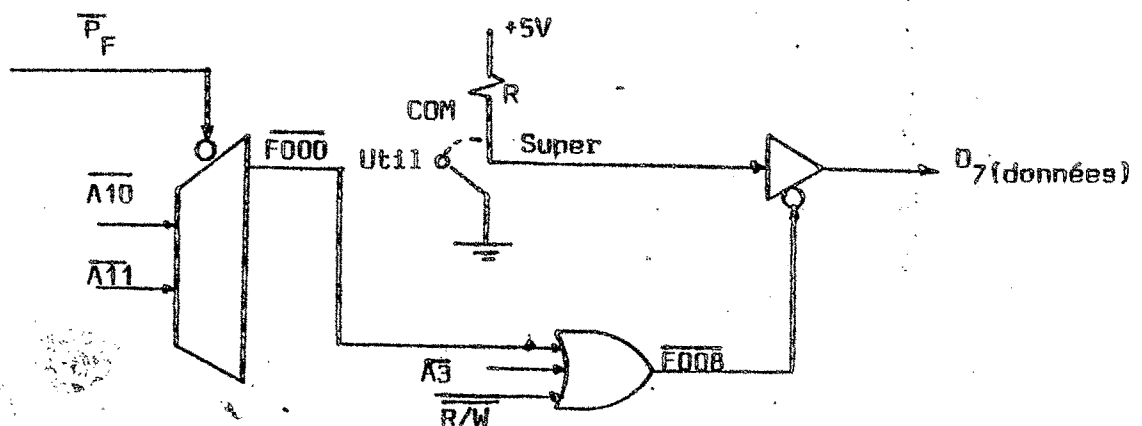


Figure A2.4 Mécanisme logiciel.

Les interruptions IRQ reçues par le système sont transmises, dans les deux modes, aux programmes utilisateurs par adressage indirect sur l'adresse IRQ du vecteur utilisateur.

Les interruptions NMI reçues par le système provoquent l'entrée dans le superviseur; celui-ci teste l'interrupteur COM pour:

- en mode utilisateur: passer le contrôle aux programmes d'application par adressage indirect sur l'adresse NMI du vecteur utilisateur;
- en mode superviseur: entrer dans le superviseur.

Avantages:

- \* les programmes "utilisateurs" peuvent utiliser les sous-programmes de service du superviseur qui restent dans l'espace d'adressage.
- \* cette solution ne nécessite que peu de matériel de commutation.

Inconvénient:

- \* le passage par le superviseur ralentit la vitesse de prise en compte des interruptions de l'utilisateur.

4.3.2) Mécanisme matériel.

Le système précédent peut être perfectionné, en utilisant une commutation matérielle, de manière à améliorer le temps de réaction aux interruptions. Nous considérons deux options:

### 1) Confusion des vecteurs utilisateur et machine.

Lorsque l'on désire fonctionner en mode utilisateur, l'accès au vecteur utilisateur (EEEE-EFFF) est électriquement substitué à l'accès au vecteur machine FFF8 à FFFF (figure A2.5) par l'action de la clef COM.

L'entrée dans le superviseur se fait:

- soit en démarrant en mode superviseur, ce qui initialise la pile et l'ACIA,
- soit à l'aide d'une interruption NMI avec la clef COM en position superviseur (le superviseur suppose alors la pile initialisée et travaille à partir de sa valeur courante), il initialise l'ACIA.

#### Avantages.

- \* rapidité de prise en compte des interruptions par l'utilisateur,
- \* l'ACIA et les sous-programmes utilitaires du superviseur restent utilisables.

#### Inconvénients.

- \* l'utilisateur doit initialiser la pile et l'ACIA qui ne sont plus initialisées par le superviseur pour l'utilisateur,
- \* coût matériel.

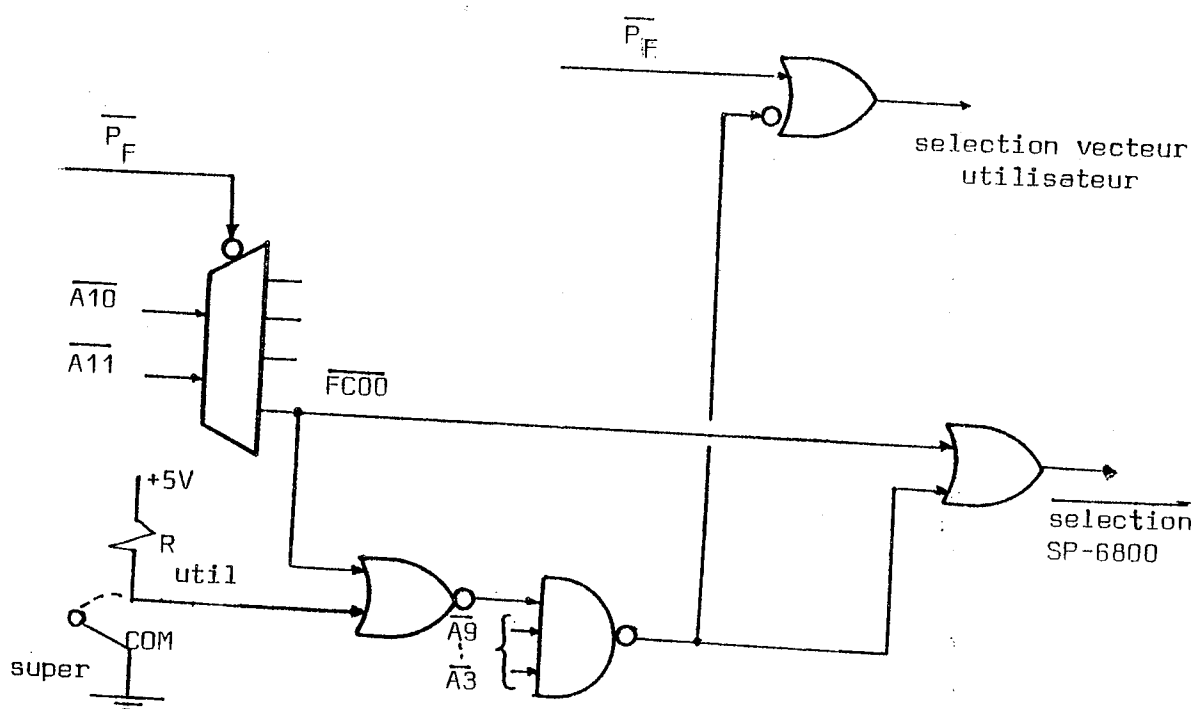


Figure A2.5 Confusion des vecteurs utilisateur et machine.

## 2) Confusion sur l'espace utilisateur.

En mode utilisateur, on confond la page utilisateur (E) (E000-EFFF) et la page système (F) (F000-FFFF). L'entrée au superviseur se fait de la même manière que précédemment.

Avantages.

- \* rapidité,
- \* simplicité matérielle.

Inconvénients.

- \* impossibilité d'accès, en mode utilisateur, ni aux sous-programmes utilitaires du superviseur, ni à l'ACIA.

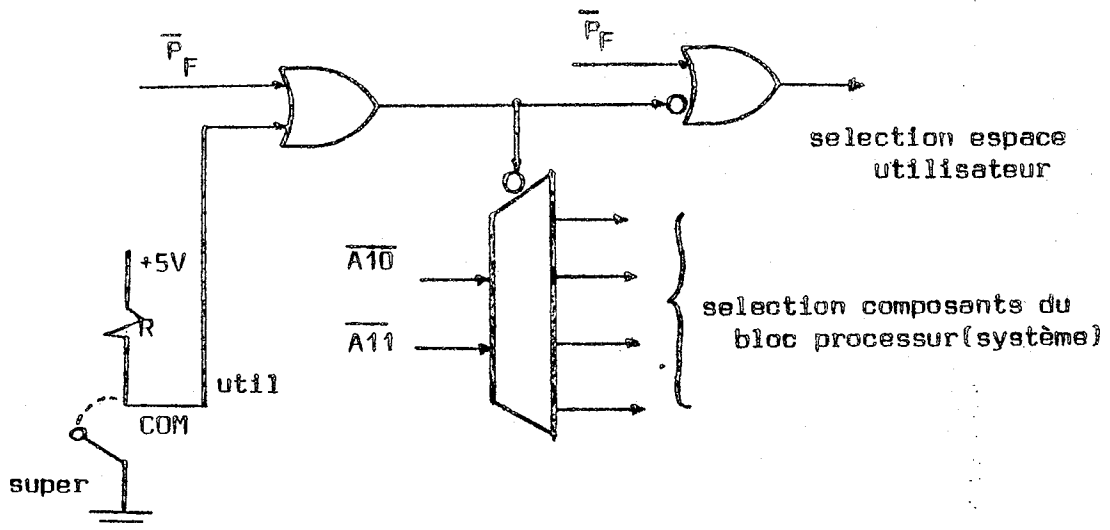


Figure A2.6 Confusion sur l'espace utilisateur.

5.) LISTING DU PROGRAMME MONITEUR SP-6800.

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CE	CODE	INST	OPERANDES	COMMENTAIRES
		1	*****	
		2	* SUPERVISEUR 6800 *	
		3	* A.ZAMBRANO E.N.S.I. .A.S *	
		4	*****	
0000	0000	0	ZER0 EQU 0	
0000	0001	1	A EQU 1	
0000	0002	2	00X EQU 2	
0000	0011	1	AEON EQU =X11	REF. ON
0000	0013	1	AEON EQU =X13	REF. OFF
0000	0012	1	AEON EQU =X12	REF. ON
0000	0014	1	AEON EQU =X14	REF. OFF
		12	*	
		13	* ADRESSE ACIA MAQUETTE	
		14	*	
0000	F200	15	ETAI EQU =XF200	
0000	F300	16	DCNA EQU =XF300	
0000	F87E	17	P7LE EQU =XF87E	
0000	F400	18	MODE EQU =XF400	
		19	*****	
F800	F800	20	ORG =XF800	
		21	*****	
F800	0000	22	PCI DCN 1	
F801	0000	23	XSAV DCN 2	
F803	0000	24	SPFL DCN 2	
F805	0000	25	SWT1 DCN 2	
F807	0000	26	VAD DCN 2	
F809	0000	27	CKSM DCN 1	
F80A	0000	28	PCON DCN 1	
F80B	0000	29	AFCO DCN 1	
F80C	0000	30	BEGA DCN 2	ECHO
F80E	0000	31	FRDA DCN 2	
F810	0000	32	TRIT DCN 24	
F828	F828	33	ETBT EQU **0	8 POINTS D'ARRET
		34	*****	
		35	* VECTEUR IRQ	
		36	*****	
F828	0028	37	VIRQ DCN 2	
F82A	0028	38	SECU DCN 1	
		39	*****	

PROG : SP-6800

ASSEMBLEUR MICROPROCESSEUR 6800 ARCHIT

ADR	INST	OPERANDES	COMMENTAIRES
FC00 0000	41	*****	*****
	42	ORG =XFC00	
	43	*****	*****
	44	*	RECONNAISSANCE DE CHIFFRE HEXA
	45	*	LE CHIFFRE EST SUPPRIME SI LE CARACTERE N'EST
	46	*	RETOUR: CARRY=0 SI HEXA
	47	*	" =1 NON HEXA
	48	*****	*****
FC09 0059	49	LHX1 RSR	SPAC
FC02 7CF80B	50	LFX INC	RECO
FC05 80FC07	51	JSR	LCAR
FC08 7AF80B	52	DEC	AFCO
FC0B 8140	53	LHRB CMPI	B,=X40
FC0D 2F0B	54	PLB	EI
FC0F 8146	55	CMPI	A,=X46
FC11 2215	56	RHI	FRH
FC13 8009	57	ADDI	A,=X09
FC15 200B	58	PRB	RETR
FC17 912F	59	F1 CMPI	A,=X2F
FC19 2F0D	60	PLB	ERH
FC1B 9139	61	CMPI	A,=X39
FC1D 2209	62	RHI	FRH
FC1F 840F	63	RETR	A,=X0F
FC21 36	64	PSHA	
FC23 80FC0A	65	JSR	PHX
FC25 32	66	PULA	
FC26 0C	67	CLC	
FC27 39	68	RTS	
	69	*	
	70	*	
FC29 0D	71	EPH	SFC
FC2B 39	72	RTH	RTS



PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CE	CODE	INST	OPERANDES	COMMENTAIRES
		74 *		
		75 *		
		76 *	CONSTRUCTION D'UN OCTET	
		77 *		
		78 *		
FC2A	8D2F	79	DOCT BSR SPAC	
FC2C	8D04	80	LOCT BSR LHEX	
FC2E	2514	81	PCS ERH1	
FC30	37	82	LOCL PSHF	
FC31	48	83	ASLA A	
FC32	48	84	ASLA A	
FC33	48	85	ASLA A	
FC34	48	86	ASLA A	
FC35	16	87	TAP	
FC36	9DCA	88	BSR LHEX	
FC38	250A	89	PCS ERH1	
FC3A	18	90	APA	
FC3B	16	91	TAP	
		92 *		
		93 *	MISE AU POINT CHECK-SUM	
		94 *		
FC3C	F7F809	95	ADCA B,CKSM	
FC3E	F7F809	96	STA B,CKSM	
FC42	33	97	PULB	
FC43	39	98	RTS	
		99 *		
FC44	7DF809	100	FDHI TST AFCD	
FC47	2703	101	PFC CLAV	
FC49	7EFF64	102	JMP EROR	
FC4C	2029	103	CLAV APA COI	ERREUR CHARGEME ERREUR CLAVIER
		104 *		
		105 *	CONSTRUCTION D'UNE ADRESSE	
		106 *		
FC4F	0EE801	107	CADP LCXI XSAV	
FC51	8D07	108	CAD1 BSR DOCT	
FC53	A700	109	STA A,ZERO(X)	
FC55	8D05	110	BSR LOCT	
FC57	A701	111	STA A,UN(X)	
FC59	FF00	112	LDR X,ZERO(X)	X..... ADRESSE
		113 *		
		114 *	IMPRESSION D'UN ESPACE	
		115 *		
FC5B	8620	116	SPAC LOI A,=X20	
FC5D	2051	117	PRA PCA2	
		118 *		

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CE	CODE	INST	OPERANDES	COMMENTAIRES
		120 *		
		121 *		
FC61	8090	122 MOCT	BSR PCCY	
FC63	2504	123	BSR LFX1	
FC65	3019	124	BCS RTTY	
FC67	A700	125	BSR LOCI	
FC69	20F4	126	STA A,ZERO(X)	
		127	BRA MCCY	
		128 *		
		129 *		
		130 *		
		131 *		
		132 *		EXAMEN/MODIF MEMOIRE
FC68	8DE1	133 MEM	BSR GADR	
FC69	90F0	134 MEM1	BSR MOCT	
FC6E	814E	135	CMPI A <sub>0</sub> =X&E	NEXT INSTRUCTIO
FC71	270A	136	RFC MEM3	
FC73	9155	137	CMPI A <sub>0</sub> =X55	'U'
FC75	2703	138	RFC MEM2	
FC77	7EFD45	139 CCI	JMP CCM	
FC7A	09	140 MEM2	DEGX	
FC7B	2091	141	BRA MEM4	
FC7D	08	142 MEM2	INCX	
FC7E	824A	143 MEM4	BSR RTCH	
FC80	90EFC	144	JSR IMAD	
FC83	20FA	145	BRA MEM1	
		146 *		
		147 *		MISE A JOUR / EXAMEN ACCUMUL
		148 *		
FC85	30	149 CC	TSX	
FC86	2006	150	BRA MCC	CODE CONDITION
		151 *		
FC88	30	152 ACCP	TSX	
FC89	2002	153	BRA MACC	
		154 *		
FC8B	30	155 ACCA	TSX	
FC8C	09	156	INCX	
FC8D	08	157 MACC	INCX	
FC8E	90CF	158 MCC	BSR MOCT	
FC90	20F5	159 C02	BRA C01	
		160 *		
FC92	0FF803	161 REGS	LDT	SPIL
FC95	2000	162	BRA MDR	
		163 *		
FC97	30	164 REGX	TSX	
FC98	2003	165	BRA MACR	
		166 *		

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITECT

CE - CODE		INST	OPERANDES	COMMENTAIRES
FC9A 30	168 R5CP	TSX		
FC9B 08	169	INCX		
FC9C 08	170	INCX		
FC9D 08	171 MADR	TACX		
FC9E 08	172	INCX		
FC9F 08	173	INCX		
FCA0 8D14	174 MIDR	BSR	P0CT	
FCA2 08	175	INCX		
FCA3 8D13	176	BSP	Q0CT	
FCA5 09	177	DECX		
FCA6 8DA9	178	BSR	CAD1	
FCA8 20F6	179 C03	PRA	C02	
	180 *			
	181 *			
	182 *			
	183 *			IMPRESSION D'UN CHIFFRE HEXA
FCAA 9109	184 0HFX	CMPI	A,0	
FCAC 2204	185	RHI	ALPH	
FCAD 8A30	186	CRJ	A,=X30	
FCB0 202E	187 PCA2	PRA	PCA1	
FCB2 9B37	188 ALPH	ADDI	A,=X37	
FCB4 202A	189	BRA	PCA1	

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CF	CODE	INST	OPERANDES	COMMENTAIRES
		191 *		
		192 *		
		193 *		IMPRESSION D'UN OCTET PRECEDE DE
		194 *		
		195 *		
FCB6	900F	196 PCT	PSR	FCUL =
FCB7	A600	197 PCT	LDA	A, ZFR0(X)
FCB8	36	198 IMPC	PSHA	
FCB9	44	199	LSRA	A
FCBC	44	200	LSRA	A
FCBD	44	201	LSRA	A
FCBE	44	202	LSRA	A
FCBF	8DE9	203	PSR	PHEX
FCC1	32	204	PIJLA	
FCC2	840F	205	ANCT	A, =X0F
FCC4	20F4	206	PRA	PHEX
		207 *		
		208 *		
		209 *		IMPRESSION DE =
		210 *		
		211 *		
FCC6	863D	212 PCHI	LCI	A, =X3D ""=""
FCC8	2018	213	PRA	PCAR
		214 *		

PROGRAMME : SP-6900

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITE

CE	CODE	INST	OPERANDES	COMMENTAIRES
				216 *
				217 *
				218 *
				RETOUR CHARACT
FCCA	FFF801			
FCCD	0FFFF9			
FC00	RDFE49			
FC03	FFF801			
FC06	39			
				219 RICH
		STR	X,XSAV	SAUVEGARDE L'IND
		LDXI	RCLF	
		JSR	PDAT	
		LDR	X,XSAV	RESTAURE INDEX
		RTS		
				223 PTH
				224 *
				225 *
				LECTURE UN CAPACTERE PUIS ECI
				SI ECHC=0 SEULEMENT
				226 *
				227 *
FC07	R6E200			
FC0A	44			
FC0B	24FA			
FC0D	R6E300			
				228 LCAR
		LDA	A,ETAI	LECTURE ETAT
		LSRA	A	
				229
				230
		RCC	LCAR	BOUCLE ATENTE
		LDA	A,DDNA	
				231
				232 *
				233 *
FC0E	70E300			
FC0B	2600			
				234 PCAT
		TST	AECC	
		RNE	RET3	
				235
				236 *
				*IMPRESSION D'UN CARACTERE
				237 *
FC05	R7E300			
FC03	37			
FC09	E6E200			
FC0C	54			
FC0D	54			
FC0F	24E0			
FC0D	33			
FC0E	39			
				238 PCAR
		STA	A,DDNA	ECRITURE CARACT
		PSHP		SAUVEGARDE
				240
				241 FCP
		LCA	R,ETAI	
				242
		LSRA	R	
				243
		LSRA	R	
				244
		RCC	FCR	
				245
		PULB		
				246 RET3
		RTS		RESTAURAT
				247 *
				248 *

PROGRAMME : SP-6900

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITE

CE	CODE	INST	OPERANDES	COMMENTAIRES
		250 *		
		251 *		
		252 *	EMBRAYAGE PERFORATEUR RUBAN	
		253 *	PERFORATION AMORCE	
		254 *		
FCF2	8612	255	PROR LDI A,PEON	
FCF4	8DEF	256	PSR PCAR	
		257 *		
FCF6	8620	258	PROR LDI B, $\times 20$	32 NULS
FCF8	4F	259	CLRA A	
FCF9	8DEA	260	PRUL PSR PCAR	
FCFB	5A	261	DECA B	
FCFC	26FB	262	BNE PNUL	
FCFE	39	263	RETZ RTS	
		265 *		
		266 *		
		267 *	RESET ACIA	
		268 *		
		269 *		
ED0E	8603	270	MSOZ LDI A, $\times 03$	RESET ACIA
ED11	87E200	271	STA A,ETA1	
ED04	8601	272	IAC1 LDI A, $\times 01$	INIT ACIA SUPPRE
		273 *		
		274 *	7 BITS + 2 BITS STOP	
		275 *		
ED06	87E200	276	USTP STB A,ETA1	
ED09	85E300	277	LDA A,DOVA	LECTURE DUMMY
ED0C	39	278	RTS	
		279 *		
		280 *	OFF-READER/PUNCH	
		281 *		
ED07	8613	282	DEPR LDI A,REDF	OFF-SOFT
ED0F	8004	283	PSR PCAR	
ED11	8614	284	LDI A,PEDF	
ED13	2000	285	PRR PCAR	

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCH

CF	CODE	INST	OPERANDES	COMMENTAIRES
		287	*	
		288	*	RETOUR/LANCEMENT
		289	*	
F015	BDFC4E	290	CC JSR	CADR
F018	9DB0	291	RSP	RTCH
F01A	6E00	292	JMP	ZERO(X)
F01C	8DAC	293	PTRN PSP	RTCH
F01E	38	294	RTI	
		295	*	RETOUR INTERR

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSOR EQUIPE ARCHITE

CE	CODE	INST	OPERANDES	COMMENTAIRES
		297	*****	
FD1E	CFE910	298	CLER LDXI YBIT	
FD27	86CC	299	INBT LDI A <sub>0</sub> =XCC	CODE INVAI IOE
FD24	A702	300	STA A <sub>0</sub> DEUXIXI	
FD26	09	301	INCX	
FD27	09	302	INCX	
FD28	08	303	INCX	
FD29	9CF829	304	CPXI FTBT	
FD2C	26F4	305	BNE INBT	
FD2E	87F82A	306	STA A <sub>0</sub> SECU	
FD31	30	307	RTS	
		308	*****	
		309	* ENTREE PAR RESTART *	
		310	*****	
FD37	9EF87F	311	INIT LDSI PILE	INIT PII
FD35	8DC8	312	RSR MSRZ	INIT ACIA
FD37	8DD4	313	RSR CFPR	
FD39	8DE4	314	RSR CLER	INIT TRIT/SECU
		315	*	
		316	*	
		317	*	
FD38	86CC	318	NRYS LDI A <sub>0</sub> =XCC	CODE
FD39	31E82A	319	CPMA A <sub>0</sub> SECU	
FD40	2620	320	BNE ERIN	
FD42	35E803	321	STR S <sub>0</sub> SPIL	
		322	*	
FD45	9EE803	323	CPMV LDR S <sub>0</sub> SPIL	REINIT PILE
FD48	8D8A	324	RSR YACI	
FD4A	30C1	325	PSR CFPR	
FD4C	30ECCA	326	JSR RTCH	RETOUR CHARIOT
FD4E	862A	327	LCI A <sub>0</sub> =X2A	"*"
FD51	8DECE5	328	JSR PCAR	
FD54	7EFA93	329	CLA AFCD	LECTURE AVEC EC
FD57	8DECD7	330	JSR LCAR	LECTURE D'UN
		331	*	



PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR 6800 ARCH

CE	CODE	INST	OPERANDES	COMMENTAIRE
		333	*	
		334	*	ANALYSIS DES COMMANDES
		335	*	
		336	*	
ED5A	CEFFB3	337		
ED5D	A100	338	XTCM	COMMANDES
ED5E	2604		LDCI	TPCC
ED61	2E01		CMPA	A,ZERO(X)
ED63	6E00		RNE	CMSV
ED65	08		LDR	X,UN(X)
ED66	03		JMP	ZERO(X)
ED67	02		INCV	
ED68	8CEFF3		INCV	
ED6B	27D8		INCV	
ED6D	20EF		CPXI	FTCC
			BEC	COMM
			PRA	XTCM
		348	*	
		349	*	
		350	*	
ED6F	8FFA7E		LDCI	PILE
ED72	8FF803		STR	S,SPIL
ED75	80ED1E		JSR	CLER
ED78	80FCFF		JSR	MSRZ
ED7B	C6F3		LDI	R,XE3
		356	*	
		357	*	CODE ERREUR DANS A
		358	*	
ED7D	80FCCA		JSR	RTCH
ED80	17		TBA	
ED81	80FCBA		JSR	IMDC
ED84	20BF		PRA	COMM
		362	UPK	
		364	*	MESSAGES D'ERREUR
		365	*	1 ERREUR LECTURE
		366	*	2 ERREUR CAR NON HEXA
		367	*	3 DERBOREMENT PILE
		368	*	4 ADRESSE NON TROUVEE / CODE
		369	*	*****

PROGRAMME : SP-6000

ASSEMBLEUR MICROPROCESSEUR FORME ARCHITEC

CE	CODE	INST	OPERANDES	COMMENTAIRES
	371	*****		
	372	* POSE DE POINT D'ARRET *		
	373	*****		
FD86	CEFB10	374	BREK LCX1 TBIT	TABLE DE POINTS
FD89	A602	375	DEF LDA A,DEUX(X)	CODE
FD8A	R1CC	376	CMPI A,=XCC	CODE INEXISTANT
FD8D	2701	377	BFC CMT	
FD8E	08	378	INCX	
FD90	08	379	INCX	
FD91	08	380	INCX	
FD92	8CFB28	381	CPX1 FTBT	FINAL TBIT
FD95	26F2	382	BNE DEB	
FD97	2022	383	BRA ERR4	
		384	*	
		385	*	
FD99	FFFB05	386	CMT STR X,SWIT	
FD9C	8DFC51	387	JSR CACI	
FD9E	A600	388	LDB A,ZERO(X)	
FDA1	813E	389	CMPI A,=X3F	SWI
FDA3	2716	390	BFC ERR4	
FDA5	FEF905	391	LCR X,SWIT	
FDA9	A702	392	STA A,CFUX(X)	
FDAA	FE01	393	LDR X,ZERO(X)	
FDAE	963E	394	LDI A,=X3F	
FDAF	A700	395	STA A,ZERO(X)	
FDB0	A109	396	CMPI A,ZERO(X)	
FDB2	2700	397	BFC UPK	
FDB4	FEF905	398	LDR X,SWIT	
FDB7	86CC	399	LDI A,=XCC	
FDB9	A702	400	STA A,DEUX(X)	
		401	*	
		402	*	
		403	* POINT D'ARRET NON PRIS EN COMPTE	
		404	* TBIT PLEINE OU	
		405	* SWI DEJA INSCRIT OU	
		406	* POINT D'ARRET EN ROM	
		407	*	
FDBB	C6F4	408	FPR4 LCI B,=XE4	
FDBD	20BE	409	PRA ERRE	

PROGRAMME : SN-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCH

CE	CODE	INST	OPERANDES	COMMENTAIRE
		411	*	
		412	* TRAITEMENT DES POINTS D'ARRET	
		413	*	
FDBF	FFF803	414	PRIT STR S,SPIL	
FDC2	30	415	TSX	
FDC3	EE05	416	LDR X,5(X)	PC
FDC5	09	417	DECX	
FDC6	FFF807	418	STR X,VAR	
		419	*	
		420	* VAR: ADRESSE D'ARRET	
FDC9	CFE810	421	LDXI TBIT	
FDC7	8CF828	422	RETM CPXI FTBT	
FDCF	27FA	423	BEC ERR4	
FDD1	FFF805	424	STR X,SWIT	
FDD4	E500	425	LDR X,ZERO(X)	
FDD6	PCF807	426	CMPR X,VAR	VAR
FDD9	2708	427	BEC SCR	
FDD8	FFF805	428	LDR X,SWIT	
FDDF	09	429	SWIT INCX	
FDDF	09	430	INCX	
FDE0	09	431	INCX	
FDF1	20F9	432	BRA RETM	RETOUR
		433	*	
		434	* SWIT: ADRESSE RELATIVE DANS TBIT	
		435	*	
FDE3	FFF805	436	SCR LDR X,SWIT	
FDF6	A602	437	LDA A,DEUX(X)	CODE
FDF8	81CC	438	CMPI A,=XCC	
FDEA	27F2	439	BEC SWIT	
FDFC	EE00	440	LDR X,ZERO(X)	
FDEF	A700	441	STA A,ZERO(X)	REMETTRE CODE
FDF9	B0FE0F	442	JSR IMAI	IMPRESSION AI
		443	*	
		444	* POSITIONNER CODE INVALIDE	
		445	*	
FDF3	86CC	446	LDI A,=XCC	
FDE5	FFF805	447	LDR X,SWIT	
FDF8	A702	448	STA A,DEUX(X)	
FDEA	30	449	TSX	
FDEB	6D06	450	TST 6(X)	
FDED	2602	451	PNE SWI2	PC:=PC-1
FDFE	AA05	452	DEC 5(X)	
FE01	5A06	453	SWI2 DEC 6(X)	
FE03	7EED45	454	CC5 JMP COMM	RETOUR COMMAN
		455	*	

PROGRAMME : 40-600

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITE

DE	CODE	INST	OPERANDES	COMMENTAIRES
		458 *		
		459 *	GENERATEUR DES SIGNAUX LECTURE/ECRIURE	
		460 *		
FE06	0DFC4E	461	TEST JSR	CAER
FE09	0DFCD7	462	JSR	LCAR
FE0C	9152	463	CMPI	A,=X52 "0" LECTURE
FE0F	270A	464	DEC	READ
		465 *		
FE10	0DFC2A	466	JSR	DOCT
FE13	A700	467	STB	A,ZEROIX)
FE15	0DFFA4	468	JSR	TCAR
FE18	20E9	469	PRR	OKM
FE1A	A600	470	READ	A,ZEROIX)
FE1C	0DFFA4	471	JSR	TCAR
FE1E	20E9	472	PRR	READ
		473 *		
		474 *	PETOUR DE 'TEST' PAR CARACTER TOUCHE	
		475 *		

CODE	INST	OPERANDES	COMMENTAIRE
FE21 70E809	477 LOAD	TAC AEOO	LECTURE SANS
	478 *		
	479 *	ACTIVATION READER	
	480 *		
FE24 8611	481	LDI A,REON	ON READER
FE26 BDFCE5	482	JSR PCAR	
FE29 8641	483	LDT A,=X41	RTS:=1
FE2B 87F200	484	STA A,ETAI	
FE2E BDFCD7	485 CHR3	JSR LCAR	LECTURE CHAR.
FE31 8153	486	CMPI A,=X53	"S"
FE33 26F9	487	BNE CHR3	
FE35 BDFCD7	488	JSR LCAR	
FE38 8139	489	CMPI A,=X39	"9"
FE3A 2707	490	BEC CO5	
FE3C 8131	491	CMPI A,=X31	"1"
FE3E 26E9	492	BNE CHR3	
FE40 7FF809	493	CLR CKSM	ZERO CKSM
FE43 BDFC2C	494	JSR LOCT	LECTURE OCTE
FE46 8002	495	SUBI A,=X02	
FE49 87F80A	496	STA A,PCON	
FE4B BDFC4E	497	JSR CADR	CONSTRUCTION
FE4E BDFC2C	498 LCOI	JSR LCOI	
FE51 7AF80A	499	DEC BCON	
FE54 2724	500	BEC LCO5	COMPTEUR OCT
FE56 A700	501	STA A,ZERO(X)	
	502 *		
	503 *	VERIFICATION FAREGISTREMENT	
	504 *		
FE58 A100	505	CMPI A,ZERO(X)	
FE5A 2718	506	BEC CONN	CONTINUATION
FE5C 7AF80A	507	DEC AFEO	ERREUR CHARG
FE5E BDFCCA	508	JSR RTCH	
FE62 8078	509	BSR IMAC	
	510 *		
	511 *	TRATTEMENT ERREUR	
	512 *		
FE64 863E	513 EROR	LDI A,=X3E	"E"
FE66 BDFCE5	514	JSR PCAR	
FE69 BDFD04	515 CH21	JSR IACI	
FE6C BDFD00	516	JSR OFPR	
FE6E 70E809	517	INC AFEO	
FE72 BDFCD7	518	JSR LCAR	LECTURE DUMM
FE75 208C	519 COO	BRA CO5	
	520 *		
	521 *		
	522 *		
FE77 08	523 CONN	INCX	
FE79 2004	524	BRA LCOI	
FE7A 70E809	525 LCO5	INC CKSM	
FE7 27AF	526	PEC CHR3	

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CE	CODE	IAST	OPERANDES	COMMENTAIRES
		530	*	
		531	*	
		532	* F <ADR1	<ADR2
		533	*	
		534	* TEST MEMOIRE-PREMIERE	TEST-LECT/ECRIURE
		535	*	
FE7E	BDFF5C	536	VERB JSR	ACRC
FE82	BDFCCA	537	JSR	RTCH
FE85	8611	538	LDI	A,=X11
FE87	B7F800	539	STA	A,PCI
		540	*	
		541	* PREMIERE TEST	
		542	*	
FE8A	FEF90C	543	LCR	X,REGA
FE8D	A700	544	STA	A,ZER0(X)
FE8F	5600	545	LDI	A,ZER0(X)
FE91	11	546	CEB	
FE92	2702	547	REG	SUIT
FE94	0020	548	RSR	ERRR
		549	*	ERREUR
		550	*	
FE96	BCF00E	551	SETT	CMR
FE97	7719	552	REG	DCW?
FE98	08	553	INCR	
FE9C	8011	554	ADCI	A,=X11
FE9E	2450	555	PCC	UPI
FEA1	65	556	INCB	A
FEA1	20FA	557	RRR	UPI

TRAITEMENT ADR

PROGRAMME : SP-6900

ASSEMBLEUR MICROPROCESSEUR EQUILIBRÉ ARCHIT

ADRESSE	CODE	INST	OPERANDES	COMMENTAIRES
		559	*	
		560	* MULTIADRESSAGE	
		561	*	
FEA3	BDFCCA	562	DCW3 JSR	R1CH
FEA6	BDFCC6	563	JSR	EQUIL
FEA9	86F800	564	LDA	A,PC1
FEAC	FFF30C	565	LDR	X,REGA
FEAF	E600	566	CMPC	R,ZERO(X)
FEB1	11	567	CPA	
FEB2	2702	568	BEC	SWWU
FEB4	8D0D	569	RSR	ERRR
FEB6	BCF80E	570	SWW1	CMPP
FEB9	270A	571	PFC	CO9
FEBB	0B	572	INCX	
FEBD	9211	573	ADCI	A,=X11
FEBE	24EF	574	RCC	CMPO
FEC0	4C	575	INCA	A
FEC1	20EC	576	BRA	CMPC
		577	*	
		578	*	

"="

ERREUR

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHI

CE	CODE	INST	OPERANDES	COMMENTAIRES
		580	*	
		581	*	
		582	*	
		583	*	TRAI TEMENT ERREUR
		584	*	
FEC3	36	585	ERRR	PSHA
FEC6	BDFCCA	586	JSR	PICH
FEC7	3D13	587	DSR	IMAD
FEC9	BDFC5R	588	JSR	SPAC
FEC0	32	589	PULA	
FEC0	36	590	PSHA	
FEC5	BDFCBA	591	JSR	IMDC
FED1	BDFC5R	592	JSR	SPAC
FED4	BDFFA4	593	JSR	TCAR
FED7	BDFC8R	594	JSR	QDCT
FEDA	32	595	PULA	
FEDB	39	596	RTS	
		597	*	
		598	*	
FE0C	FFFB07	599	IMAD	STR X,VAR
FE0E	06F807	600	IMAD	LDA A,VAR
FE07	BDFCBA	601	JSR	IMDC
FE09	06F809	602	LDA	A,VAR+1
FE0B	7FFCBA	603	IMP	IMDC
		604	*	



PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARC

CF	CODE	INST	OPERANCES	COMMENTAIRE
		606 *		
		607 *	PUNCH <ACR1 <ACR2	
		608 *		
FEEB	8D6F	609 PUNC	BSR ACRC	DOUBLE ADRE
FEED	8DFCF2	610	JSP PNCN	PERFORN ET
FEF0	8D6C	611 PUN1	BSR TSAD	ENDA-BEGA
FEF2	2604	612	RAF PUN2	
FEF4	8110	613	CMPI A,=X10	
FEF6	2502	614	PCE PUN3	
FEF8	860F	615 PUN2	LDI A,=X0F	
FEFA	4C	616 PUN3	INCA A	
FEFB	87F800	617	STA A,PC1	
FEFE	9803	618	ADCI A,=X03	
FE00	87F807	619	STA A,VAR	
FE03	8DFCCA	620	JSR RTCH	
FE06	8DFFA4	621	JSR TCR	TEST CARACTE
FE09	CEFFF3	622	LDXI MTIP	
FE0C	8D3A	623	BSR PDAT	
FE0E	7FF809	624	CLR CKSM	CLEAR CHECK
FE11	CEFP07	625	LDXI VAP	
FE14	8D39	626	BSR PLNT	
		627 *		
		628 *		
		629 *	IMPRESION(PUNCH) ADRESSE	
		630 *		
FE16	CEFB0C	631	LDXI REGA	
FE19	8D34	632	BSR PLNT	
FE1B	8D32	633	BSR PLNT	
		634 *		
		635 *	PUNCH DATA	
		636 *		
FE1D	FFF80C	637	LDR X,BEGA	
FE20	8D20	638 PUN2	PSR PLNT	
FE22	7AF800	639	DEC PC1	
FE25	26F9	640	PNE PUN2	
FE27	FFF80C	641	STR X,BEGA	
FE2A	53	642	CCMA P	
FE2B	37	643	PSHP	
FE2C	30	644	TSX	
FE2D	8D20	645	BSR PLNT	
FE2E	33	646	PULP	
FE30	FFF80C	647	LDR X,BEGA	
FE33	30	648	DECX	
FE34	8DF807	649	CMPP X,FADA	
FE37	7607	650	RAF PLNT	

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR TYPE ARCH

CE	CODE	INSTR	OPERANDES	COMMENTAIRE		
		652 *				
		653 *	FIN DE RUPAN			
		654 *				
FF39	CEFFE6	655	LXI	S9		
FF3C	RDOA	656	RSR	POBT		
FF3E	RDECE6	657	JSR	PNDF		
		658 *		PUNCH-OFF		
FF41	7EED45	659	RRK	JMP	CCPW	RETOUR SUPER
		660 *				
		661 *				
		662 *	SORTIE DES MESSAGES			
		663 *				
FF44	RDECF5	664	PCAR	JSR	PCAR	
FF47	08	665	INCR			
FF48	A600	666	PCAT	LDA	A,ZERO(X)	
FF4A	R104	667	CMPI		A,=X04	
FF4C	26E6	668	RNE		PCAR	
FF4E	39	669	RTS			
		670 *				
		671 *				
		672 *	PUNCH CARACTERE ET CALCUL CHECK-SUM			
		673 *				
		674 *				
FF4E	RDECR8	675	PINT	JSR	OCCT	IMPRESION OCTET
FF51	E61809	676	CKS1	LDA	R,CKSM	
FF55	E800	677		ADCB	R,ZERO(X)	
FF57	E7E809	678		STA	R,CKSM	
FF5A	08	679		INCR		
FF5B	39	680		RTS		
		681 *				
FF5C	R70E	682	ADRT	RSR	ACR4	CHEKING ADRES
FF5E	R6F80E	683	TSAD	LDA	A,END4+1	
FF61	E6E807	684		LDA	R,END4	
FF64	R0F80D	685		SUPR	A,BEGA+1	
FF67	E2E80C	686		SBCA	R,BEGA	
FF6A	25D5	687		BCS	RRK	RETOUR
FF6C	39	688		RTS		
		689 *				
		690 *				
		691 *				
FF6D	CEFR0C	692	ACR4	LXI	R,CA	SANS CHEKING AI
FF70	RDFC5L	693		JSR	CAD1	
FF73	CEFR0E	694		LXI	R,CA	
FF76	RDFC5I	695		JSR	CAD1	
FF79	39	696		RTS		
		697 *				
		698 *				
		699 *				

PROGRAMME : SP-6800

ASSEMBLEUR MICROPROCESSEUR 8080 APC

HE	CODE	INST	OPERANCES	COMMENTAIRE		
		701	*			
		702	* F EFFACEMENT POINTS D ARRETS			
		703	*			
FF7A	CEFB10	704	CKE1 1DXI	TRBT		
FF7D	16D2	705	PRCA	A,CEUX(X)	CODE	
FF7E	810C	706		A,=XCC		
FF81	2630	707		RNF	RCCD	
FF83	08	708	INCT	INCX		
FF84	08	709		INCX		
FF85	08	710		INCX		
FF86	87E828	711		CPXI	FTBT	
FF89	26E2	712		RNE	PRON	
FF8B	BDEF1F	713		JSP	CLER	
FF8E	20B1	714	BKRK	BRA	PRK	RETOUR
FF90	EEF807	715	RCCD	STR	X,VAR	SAUVEGARDE
FF93	EE00	716		LDR	X,ZERO(X)	
FF95	A700	717		STA	A,ZERO(X)	
FF97	EEF807	718		LDR	X,VAR	INDEX...VAR
FF9A	20E7	719		BRA	INCT	
		720	*			
		721	*			

PROG. NOME : SP-6800

ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

ADR	CODE	INST	OPERANDES	COMMENTAIRES
		723	*	
		724	*	
		725	* VECTEUR D'INTERRUPTION(F828)	
		726	*	
		727	*	
EE00	EEF807	728	IPQ STR X,VAR	SAUVEGARDE INDI
EE0C	EEF928	729	LDR X,VIR0	
EEA2	5E00	730	JMP ZERO(X)	INDIRECTION IRC
		731	*	
EEA4	84E200	732	TCAR LDA A,ETA1	
EEA7	44	733	LSRA A	
EEA9	2501	734	PCS RTSH	
EEAA	30	735	RTS	
EEAB	86E300	736	RTSH LDA A,DCNA	
EEAF	23EF	737	PRA RKRK	RETOUR COMMANDE
		738	*	
		739	*	
EE90	3E	740	WAIT WAI	WAIT
EEF1	20E0	741	PRA WAIT	
		742	*	
		743	*	
		744	*	

PROGRAMME : Sp=6800

ASSEMBLEUR MICROPROCESSEUR

EQUIPE ARCHIT

CE	CODE	INST	OPERANDES	COMMENTAIRES
		749 *		
		750 *		
		751 *		
		752 *	TABLE DE COMMANDES	
		753 *		
FFB3	4D	754	TBCD DC1 =X4D	
FFB4	FC6B	755	CC2 MFM	"M"
FFB6	47	756	DC1 =X47	
FFB7	FD15	757	CC2 GO	LANCEMENT DE
FFB9	52	758	CC1 =X52	"P"
FFBA	FD1C	759	CC2 RTPA	RETOUR INTERRI
FFBC	41	760	DC1 =X41	"A"
FFBD	FC9B	761	CC2 ACCA	ACCU A
FFBE	42	762	DC1 =X42	
FFC0	FC8B	763	CC2 ACCB	ACCU B
FFC2	58	764	DC1 =X58	"X"
FFC3	FC97	765	CC2 REGY	INDEX
FFC5	43	766	DC1 =X43	"C"
FFC6	FC9A	767	CC2 REGP	PROGRAMME CO
FFC8	53	768	DC1 =X53	"S"
FFC9	FC92	769	CC2 REGS	PILE
FFCB	4B	770	DC1 =X4B	"K"
FFCC	FD86	771	CC2 PREK	POSE DE POINT
FFCE	54	772	DC1 =X54	"T"
FFCF	FE06	773	CC2 TFST	TEST SIGNAUX
FFD1	4C	774	DC1 =X4C	"L"
FFD2	FE21	775	CC2 LCAD	CHARGEMENT DE
FFD4	45	776	DC1 =X45	"E"
FFD5	FE7E	777	CC2 TERW	ESSAI MEMOIRE
FFD7	50	778	DC1 =X50	"D"
FFD8	FEFB	779	CC2 PLAC	PUNCH MEMOIRE
FFDA	46	780	DC1 =X46	"E"
FFDB	FE7A	781	CC2 CKF1	EFFACEMENT DI
FFDD	5A	782	DC1 =X5A	"Z"
FFDE	FCB5	783	CC2 CC	CODE CONDITIO
FFE0	57	784	DC1 =X57	"W"
FFE1	FFB0	785	CC2 WAIT	WAIT
FFE3	FFE3	786	FTCD FOL *+0	FINAL TBCD
		787 *		

PROGRAMME : 80-8800

ASSEMBLEUR MICROPROCESSEUR ARCHIT

ADRESSE	CONT	INST	OPERANDES	COMMENTAIRES
		789 *		
FFB3	53	790	W7IP DC1	=X53
FFB4	31	791	DC1	=X71
FFB5	04	792	DC1	=X04
		793 *		
		794 *		
FFB6	51	795	CS DC1	=X51
FFB7	70	796	DC1	=X70
FFB8	06	797	DC1	=X06
		798 *		
		799 *		
FFB9	00	800	RCLF DC1	=X00
FFBA	0A	801	DC1	=X0A
FFBB	00	802	DC1	=X00
FFBC	00	803	DC1	=X00
FFBD	00	804	DC1	=X00
FFBE	04	805	DC1	=X04
		806 **		
		807 *		
		808 *		
		809 *		
		810	*****	
		811 *	VECTEUR MACHINE	
		812	*****	
FFFB	FFFB	813	ORC	=XFFFB
FFFC	FFFC	814	IRQ	DC2
FFFD	FFFD	815	SWI	DC2
FFFE	FFFE	816	AMI	DC2
FFFF	FFFF	817	PRST	DC2
0000	00	818	ENC	

LEGUEUR DU PROGRAMME 1058 OCTET

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :


- F. ANCEAU, Professeur à l'Institut National Polytechnique de GRENOBLE
- S. GUIBOUD-RIBAUD, Professeur à l'Ecole Nationale Supérieure des Mines de SAINT ETIENNE

Monsieur Alexander ZAMBRANO CASTILLO

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 29 Mai 1980

Le Président de l'I.N.P.G.

  
**Ph. TRAYNARD**  
Président  
de l'Institut National Polytechnique