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

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
} Examinateurs

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
	SABONNADIERE 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)
	BOOS Jean-Yves	Ecole des Mines St. Etienne (Métallurgie)
	DRIVER Julian	Ecole des Mines St. Etienne (Métallurgie)

.../...

MM.	KOBYLANSKI André	Ecole des Mines St. Etienne (Métallurgie)
	LE COZE Jean	Ecole des Mines St. Etienne (Métallurgie)
	LESBATS Pierre	Ecole des Mines St. Etienne (Métallurgie)
	LEVY Jacques	Ecole des Mines St. Etienne (Métallurgie)
	RIEU Jean	Ecole des Mines St. Etienne (Métallurgie)
	SAINFORT	C.E.N. Grenoble (Métallurgie)
	SOUQUET	U.S.M.G.
	CAILLET Marcel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	COULON Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	GUILHOT Bernard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LALAUZE René	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	LANCELOT Francis	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SARRAZIN Pierre	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	SOUSTELLE Michel	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THEVENOT François	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	THOMAS Gérard	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TOUZAIN Philippe	Ecole des Mines St. Etienne (Chim. Min. Ph.)
	TRAN MINH Canh	Ecole des Mines St. Etienne (Chim. Min. Ph.)

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

MM.	BOREL	Centre d'études nucléaires de Grenoble
	KAMARINOS	Centre national recherche scientifique

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

M.	BORNARD	Centre national recherche scientifique
Mme	CHERUY	Centre national recherche scientifique
MM.	DAVID	Centre national recherche scientifique
	DESCHIZEAUX	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.BOUTTAZ, 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'equipe 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 sejour 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

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

III: UNE METHODE DE DESCRIPTION DE SYSTEMES MICROPROCESSEURS: MICROD

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

IV: UN PROCESSEUR DE GESTION DE MEMOIRES CCD.

IV: LE CONTROLEUR DISQUE SOUPLE.

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

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 périph rique, 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 micoprocesseur(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 Devices) (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.

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

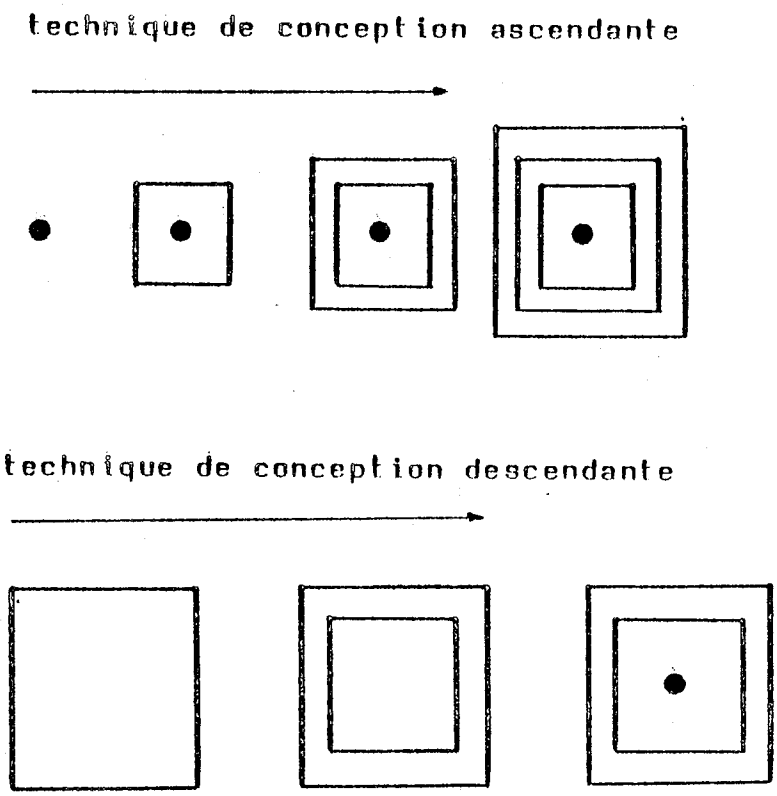


Figure 1.1 Les différentes techniques de conception.

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 trés précise des spécifications de l'application facilitera la conception et simplifera sa mise au point et son développement. Dans le cas contraire un phénomene de divergence risque d'avoir lieu.

L'imprécision des spécifications entraine une conception plus ou moins approximative d'où une mise au point problématique et un développement difficile voir impossible; ce qui aménera 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.

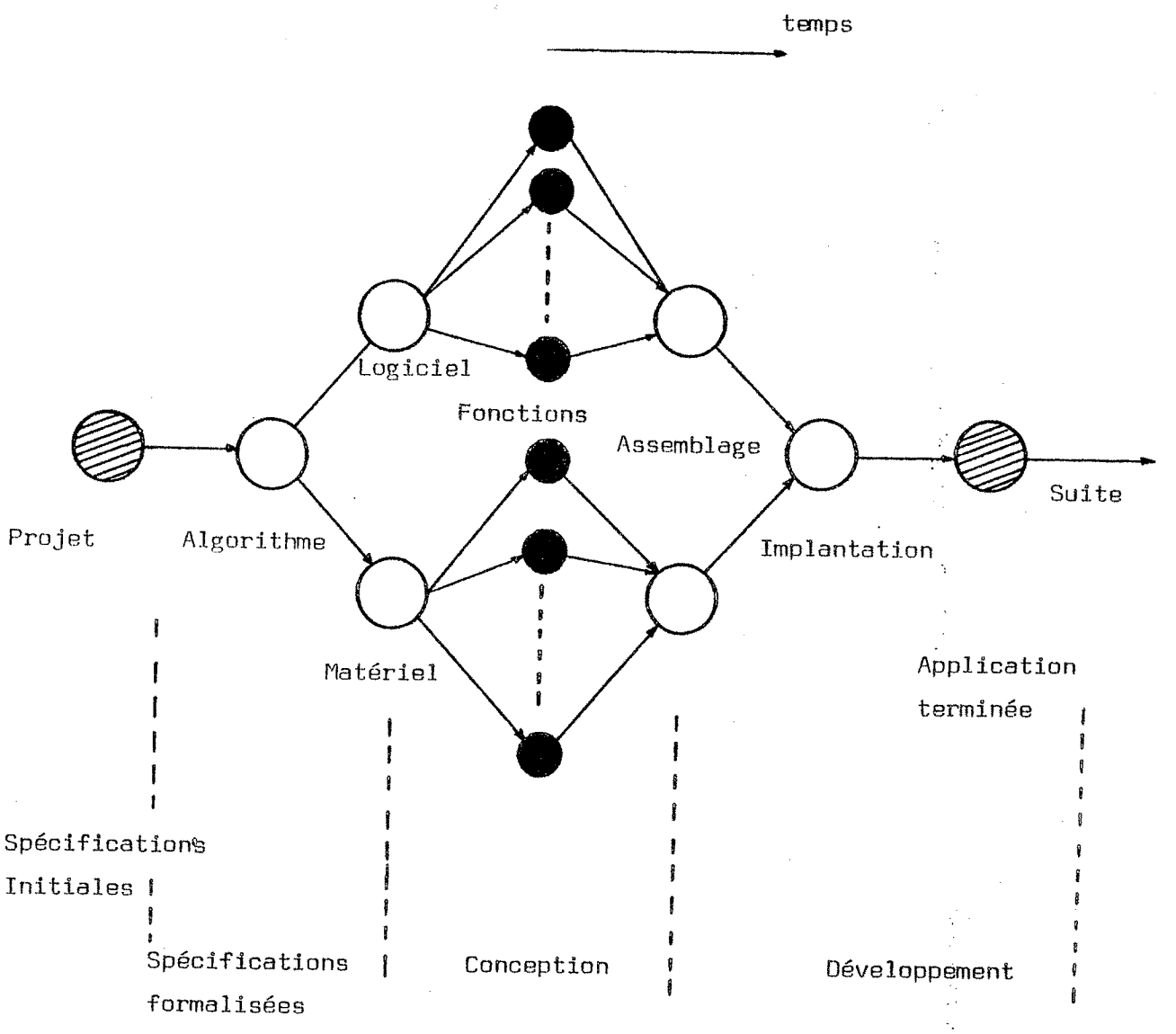


Figure 1.2 Différentes étapes et phases de la conception d'une application à base de microprocesseur.

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 trés précisement et de façon bien détaillées les actions au niveau algorithmique.

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

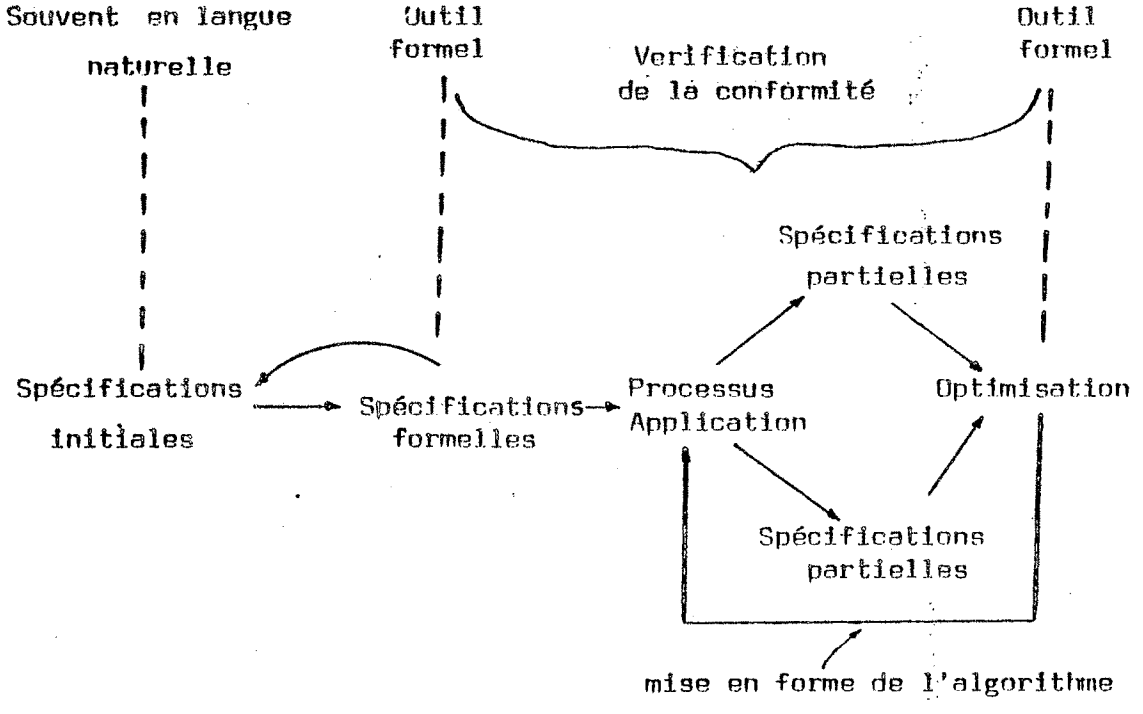


Figure 1.3 Différentes étapes de l'établissement des spécifications.

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 fontionnelle.

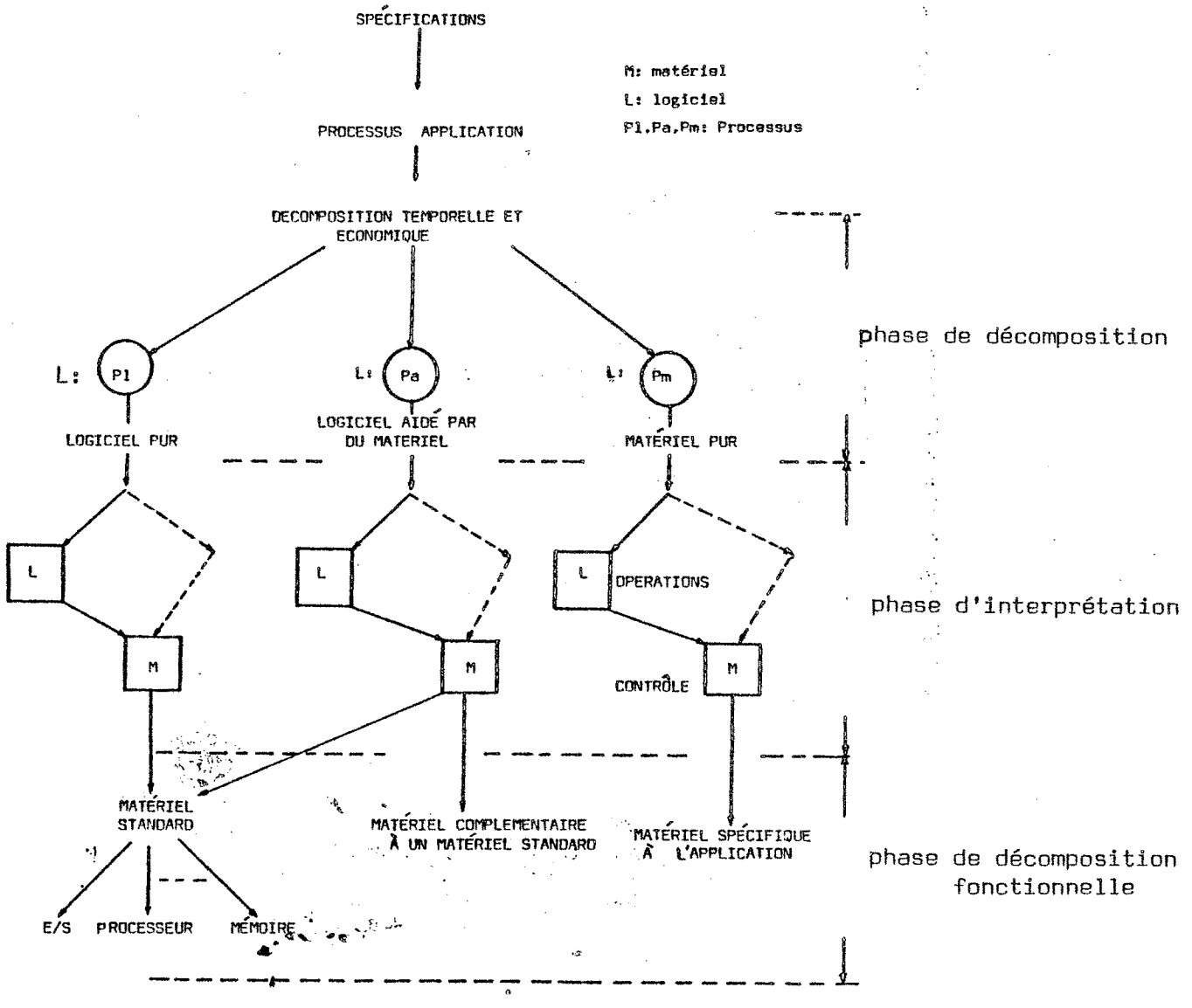


Figure 1.4 Les étapes de conception.

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 systmes 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.

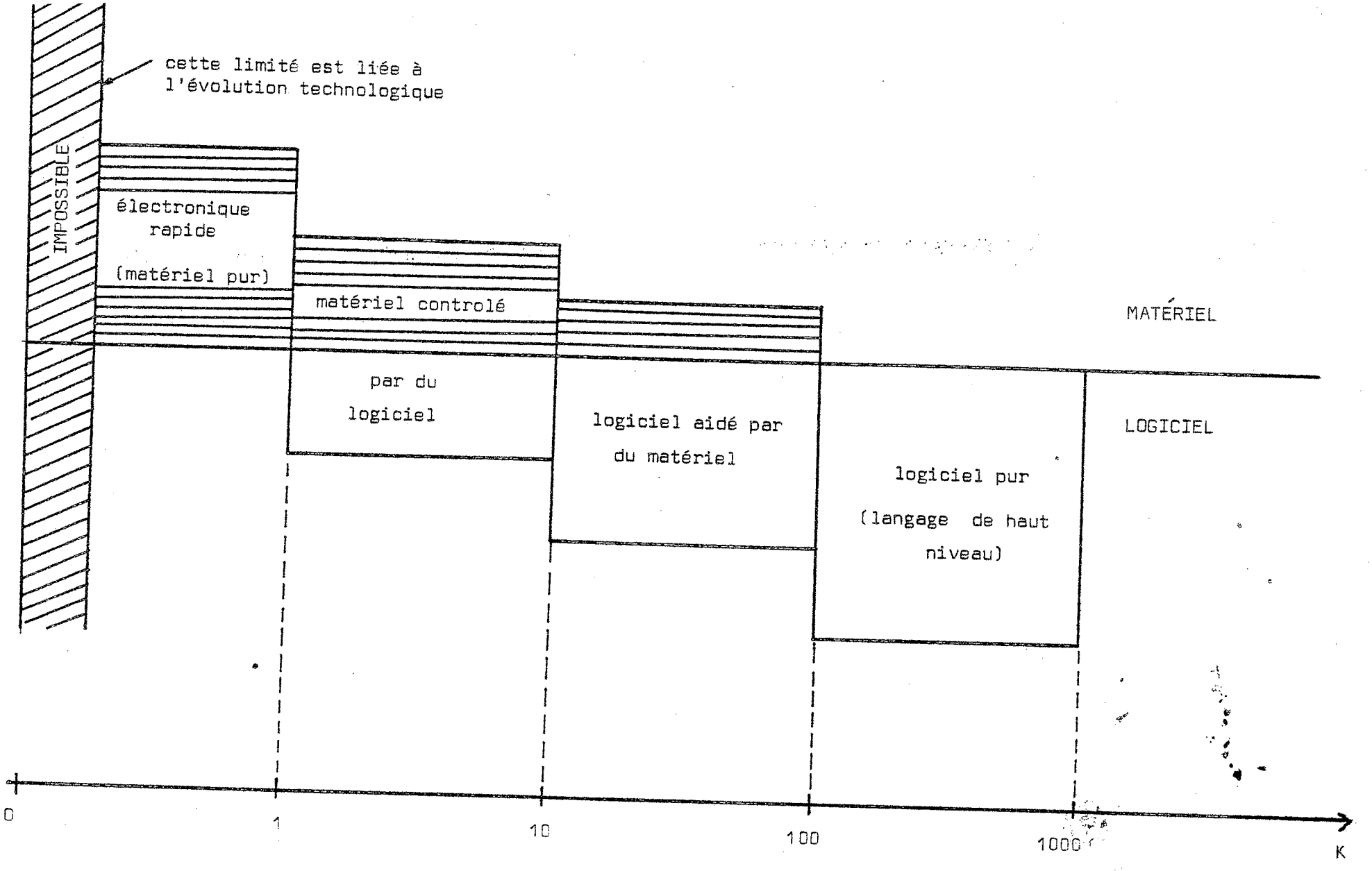


Figure 1.5 L'aspect temporel de la décomposition du processus application.

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 <--> 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 Pl,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).

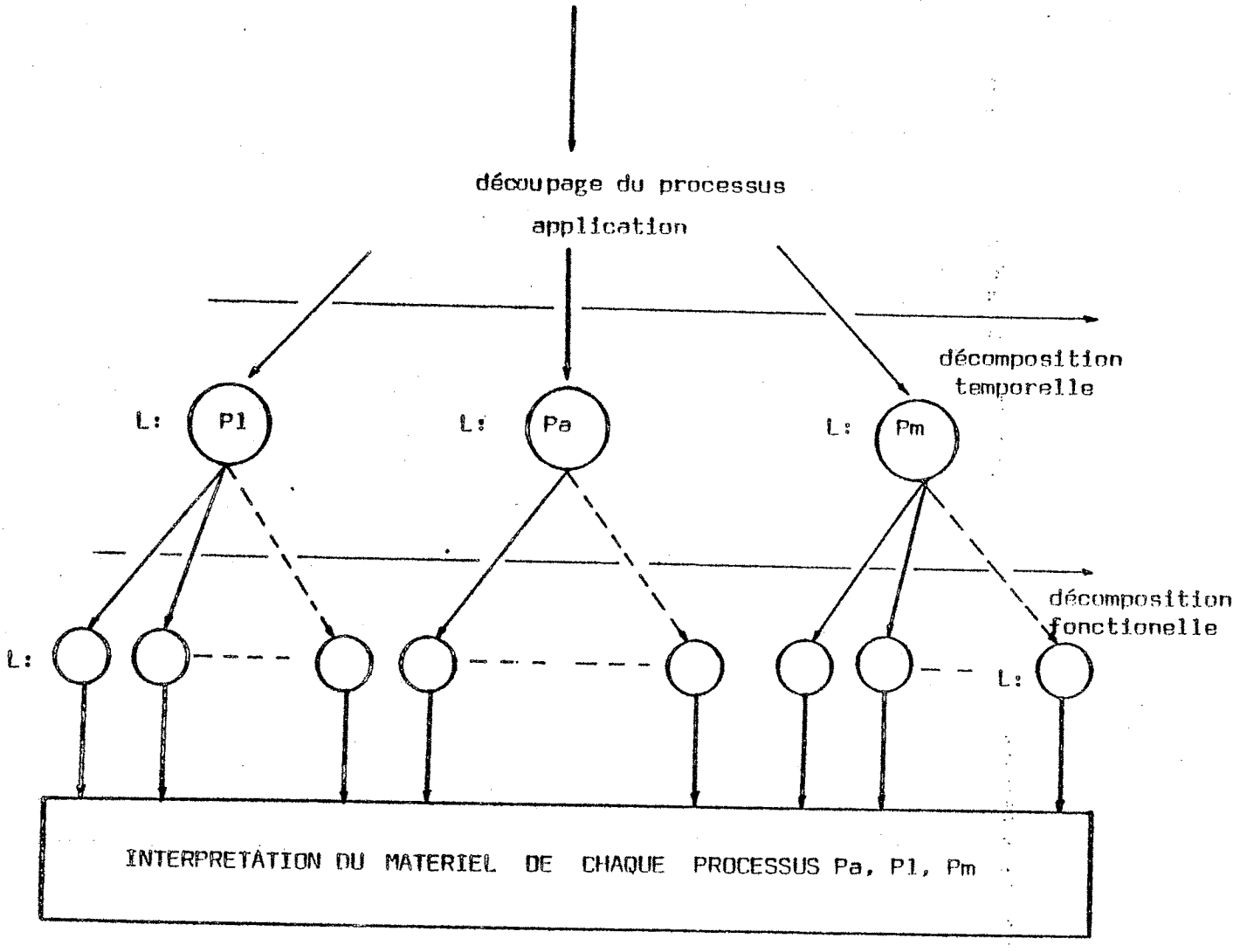


Figure I.6 La phase de décomposition du processus application.

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.

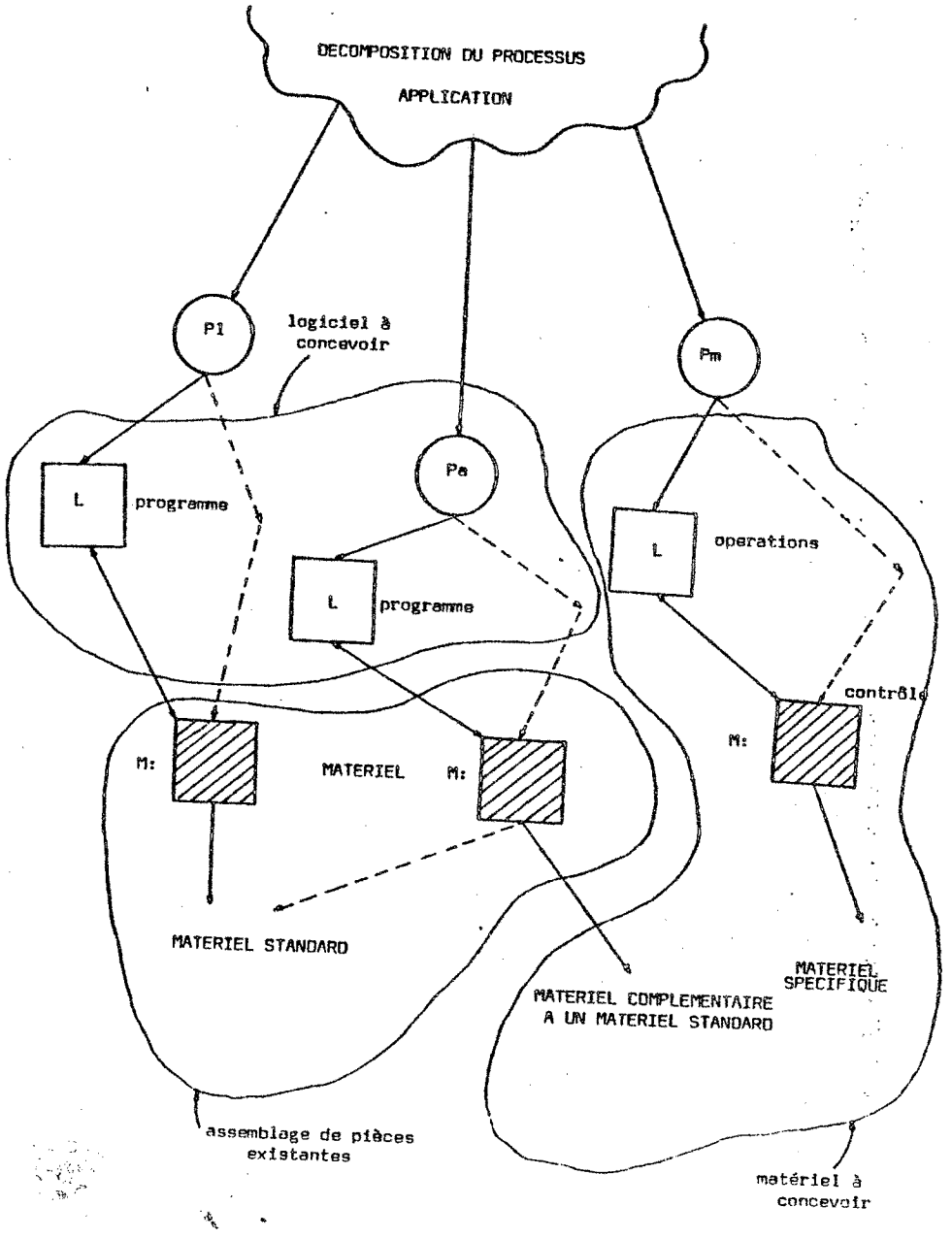


Figure 1.7 Les différentes parties à concevoir ou résultant d'assemblages de pièces existantes.

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 infaillible 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.

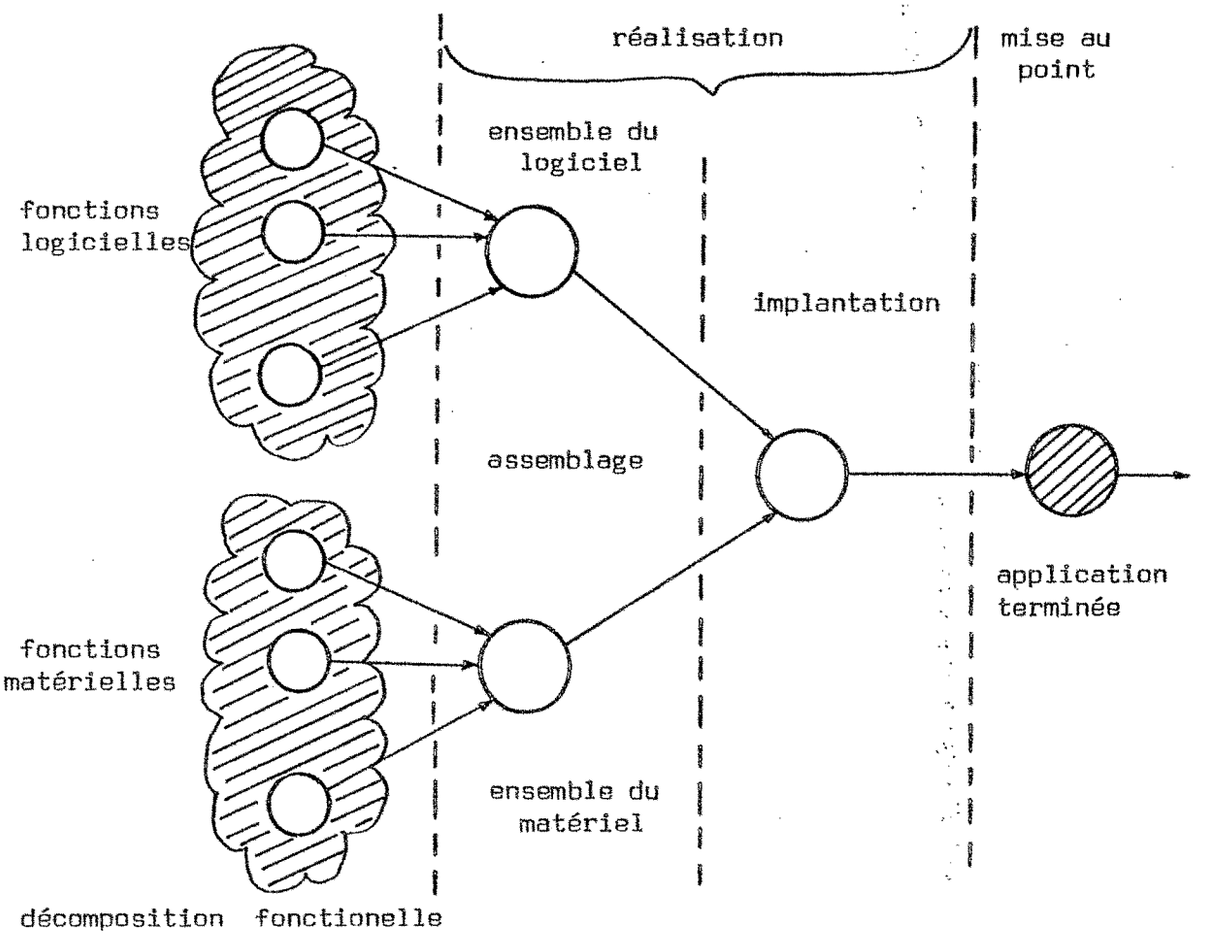


Figure 1.8 Les différentes phases du développement d'une application.

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 rigeur 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.

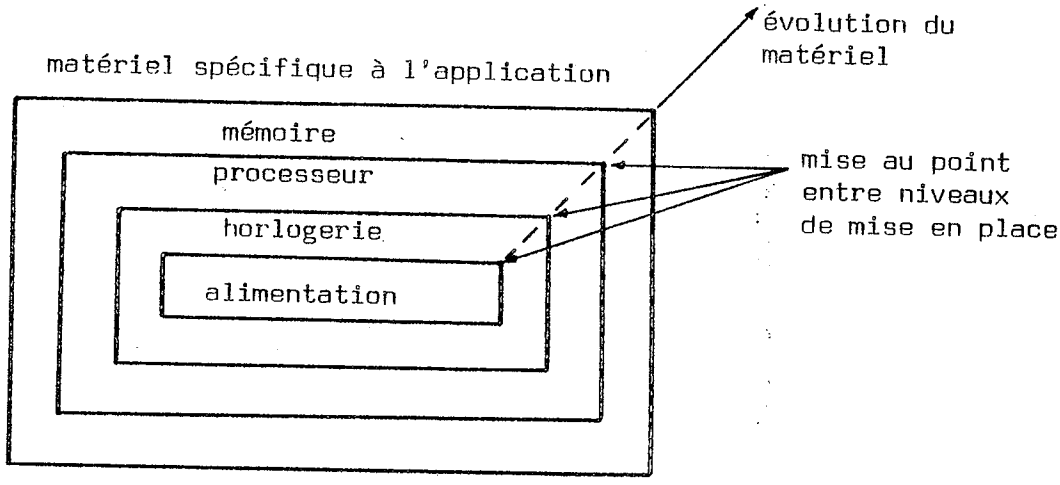


Figure 1.9 Hiérarchie de réalisation et de mise au point du matériel.

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 microproceseurs 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 malfonctions 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.

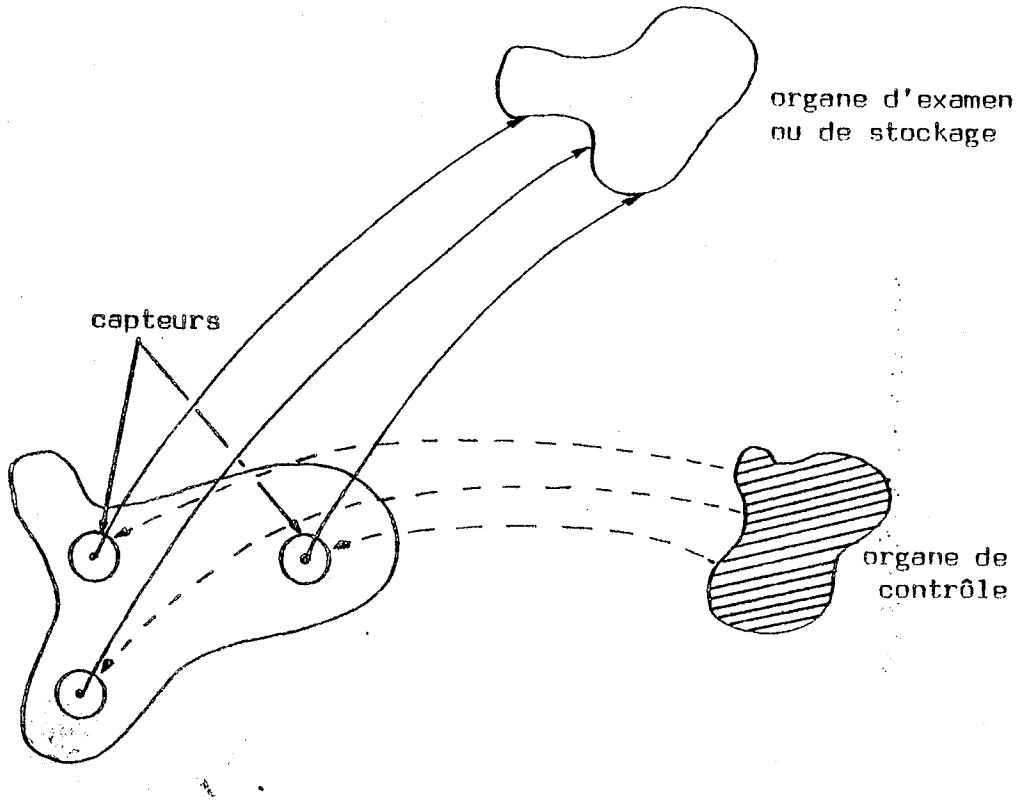


Figure 2.1 Le principe du capteur.

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).

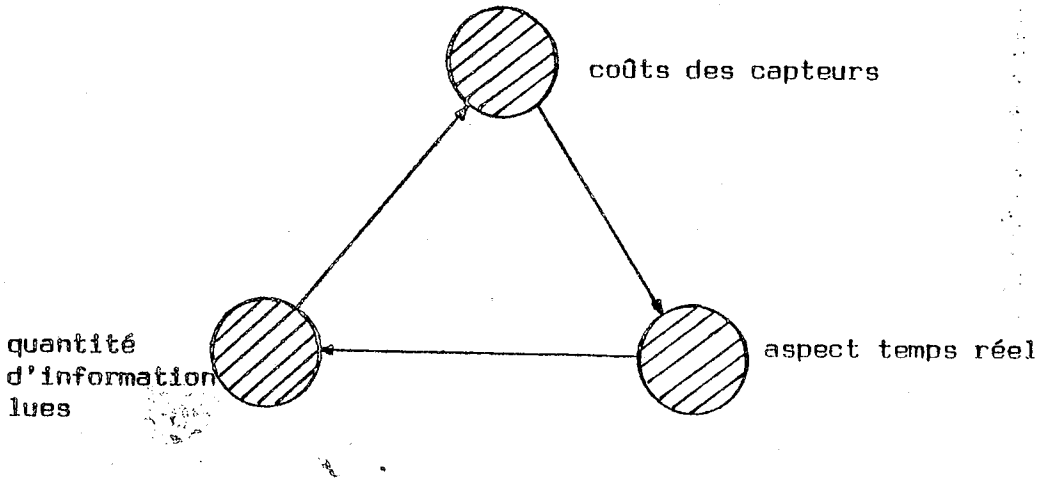


Figure 2.2 Compromis économique dans le choix des capteurs.

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.

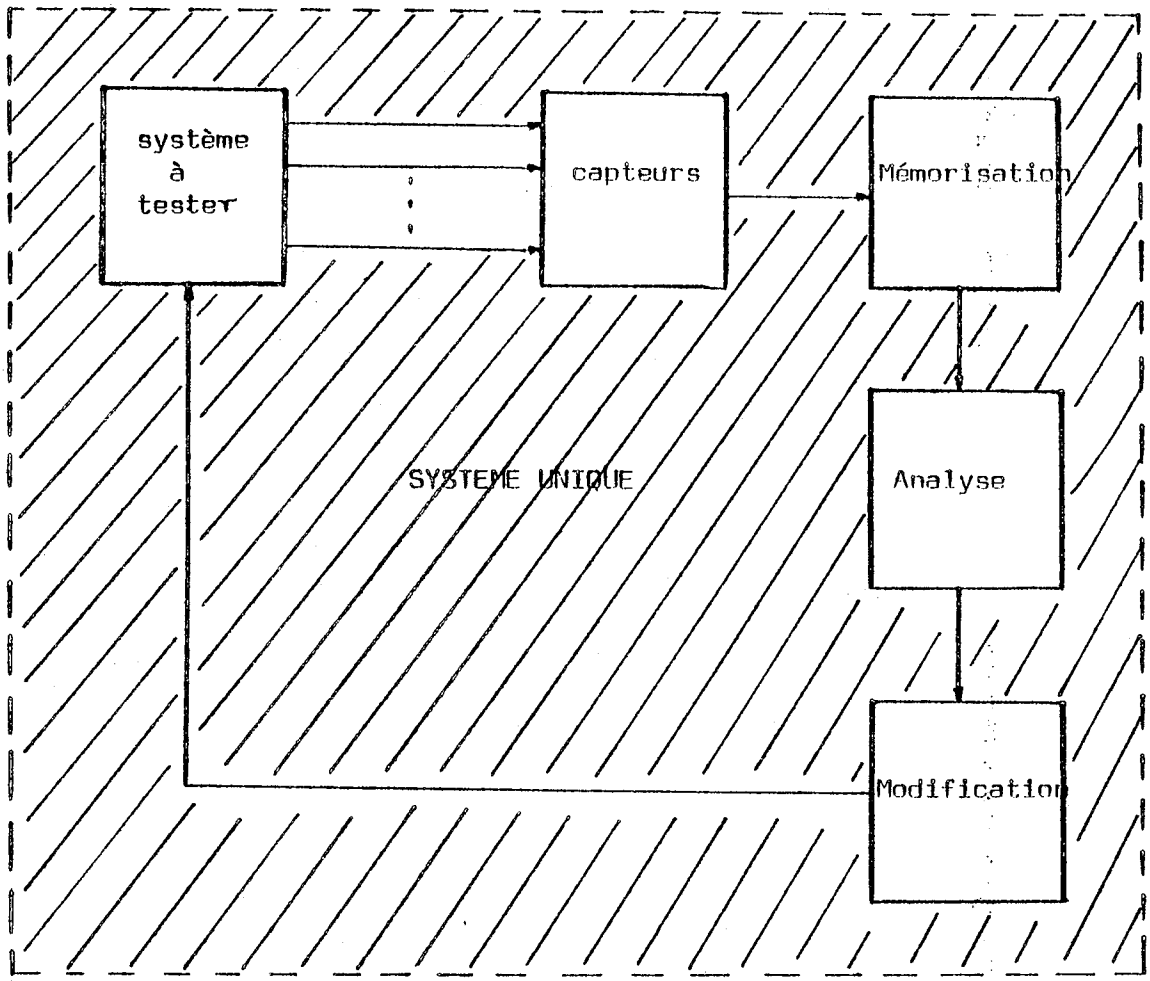


Figure 2.4 Un exemple de réalisation des étapes de la mise au point locale.

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.

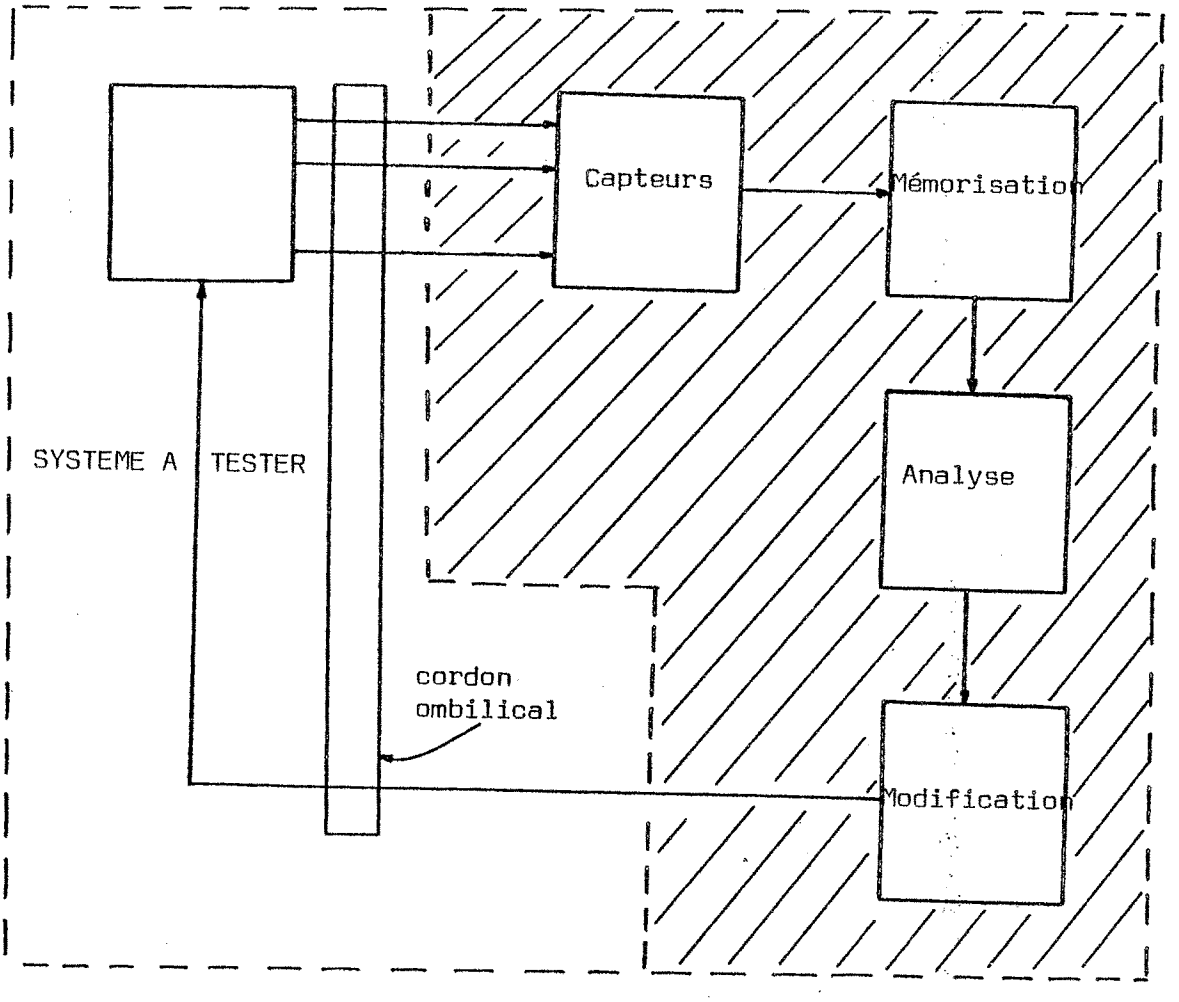


Figure 2.5 Un exemple de réalisation des étapes de la mise au point externe.

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.

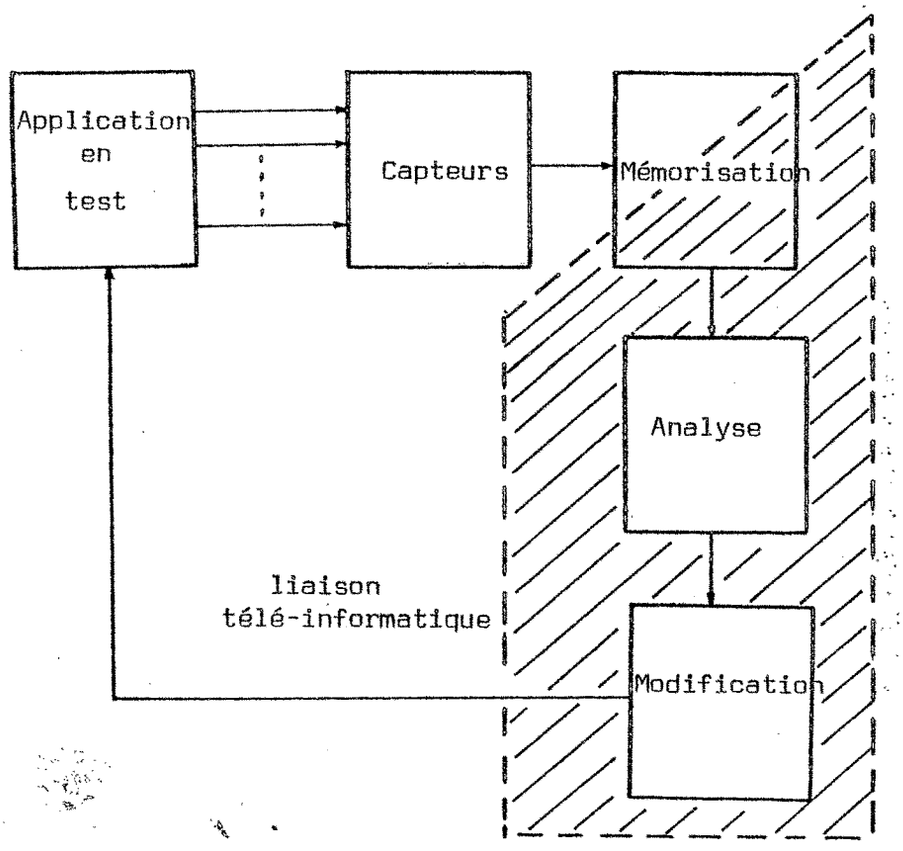


Figure 2.6 Le contexte de la mise au point semi-externe.

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 (REPROM, 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 logicel 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 strosboscopiques.

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:

- exitations périodiques par le milieu exterieur; 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'occurence 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 précedents un évènement 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 posséde 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'un façon active.

L'application est placé en mode superviseur après la rencontre d'un ou de plusieurs évènements par exemple par une écriture ou une lecture à une adresse donné, par une interruption logicielle ou matérielle ou par une requête issue de la consola.

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 traditionel).

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.

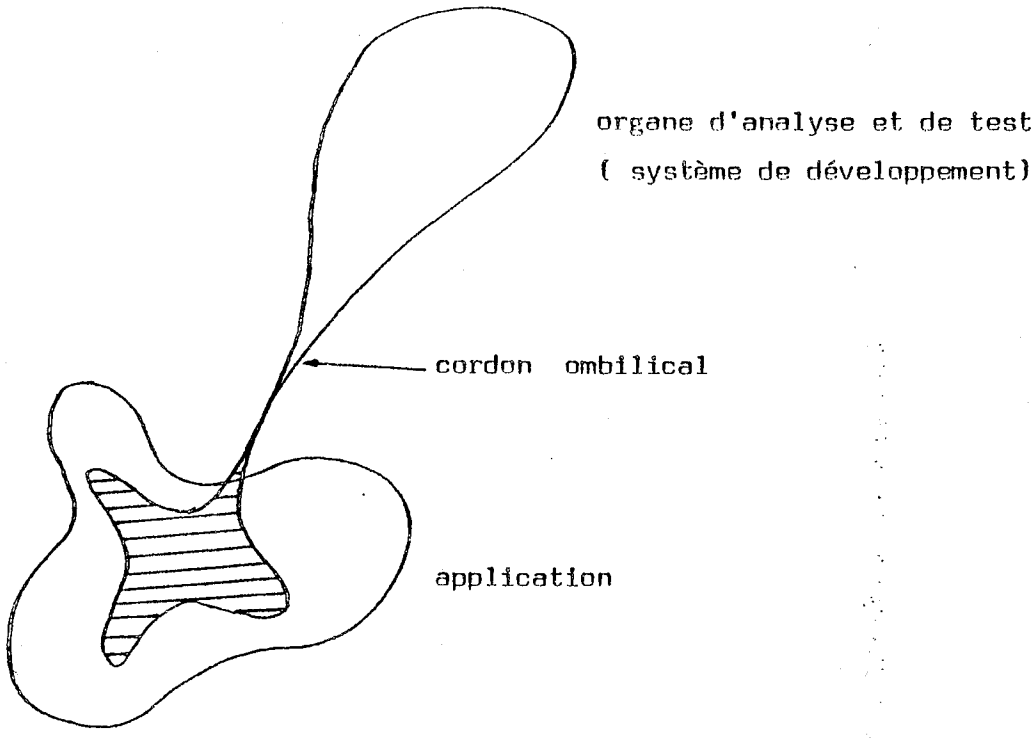


Figure 2.7 Le contexte de l'activité modification.

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 (CDL ,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.

Septs 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.
- Transducter (T): un composant qui mofifie 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-T-X

où

Mp: mémoire principale

Pc: processeur central

T : 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 hiérachie logicielle,

reliées par une relation d'implantation.

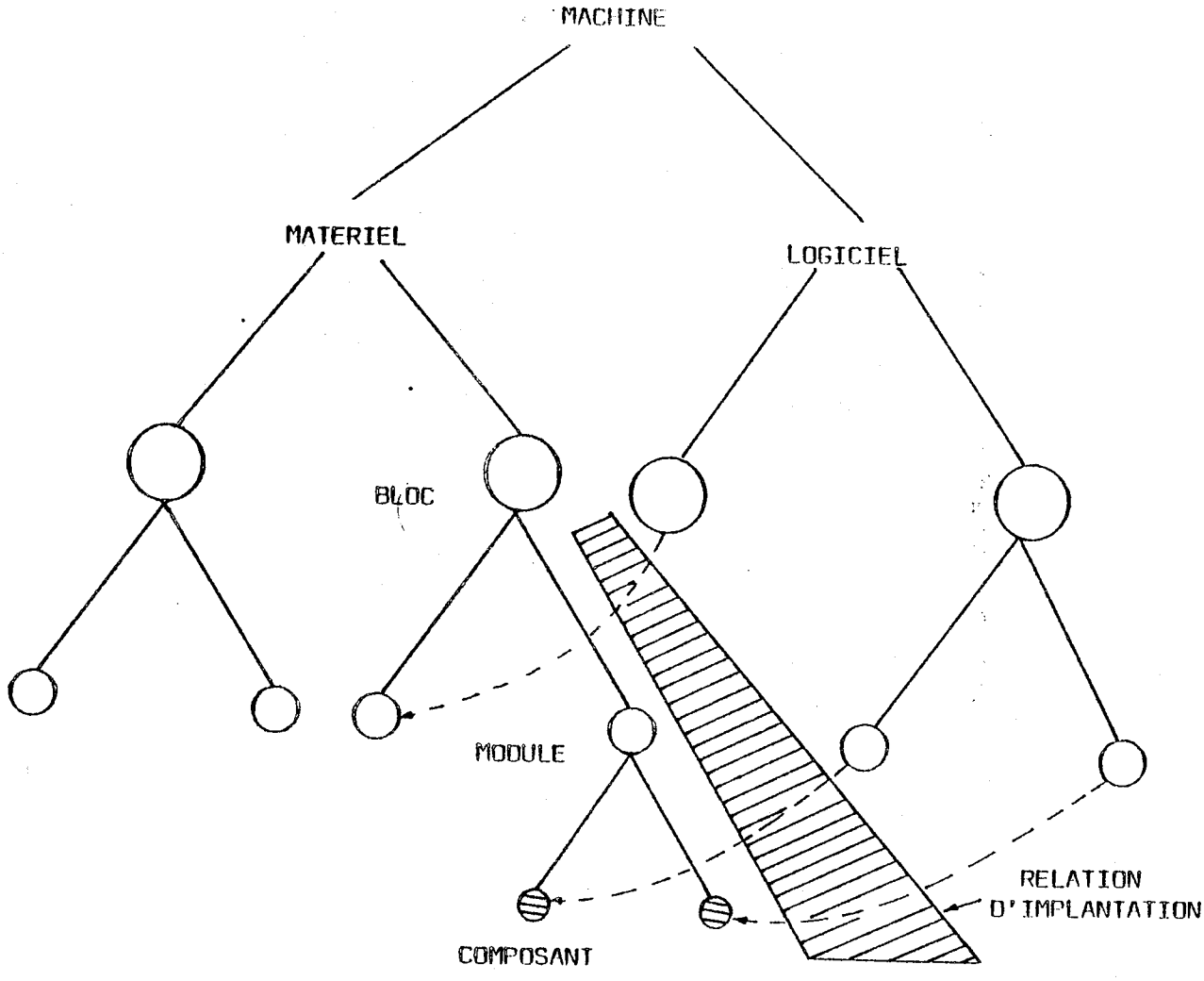


Figure 3.1 Représentation de la machine.

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.

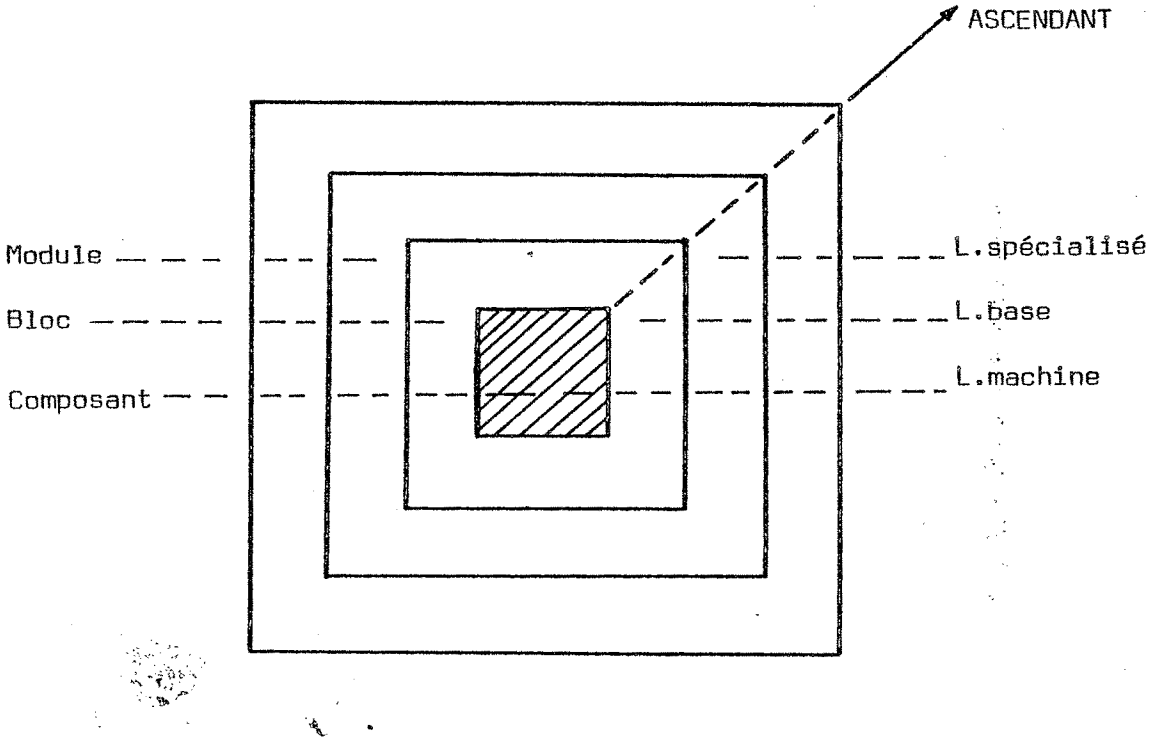


Figure 3.2 Evolution du logiciel et du matériel d'une machine.

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'interconnections.

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"
"Ɛ" 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:

<machine>::= <D.matériel>[←]<D.logicel>

où

<D.matériel>: représente la description fonctionnelle du matériel.

<D.logiciel>: représente la description fonctionnelle du logiciel.

[←] : est un symbole qui indique l'existence d'une relation d'implantation.

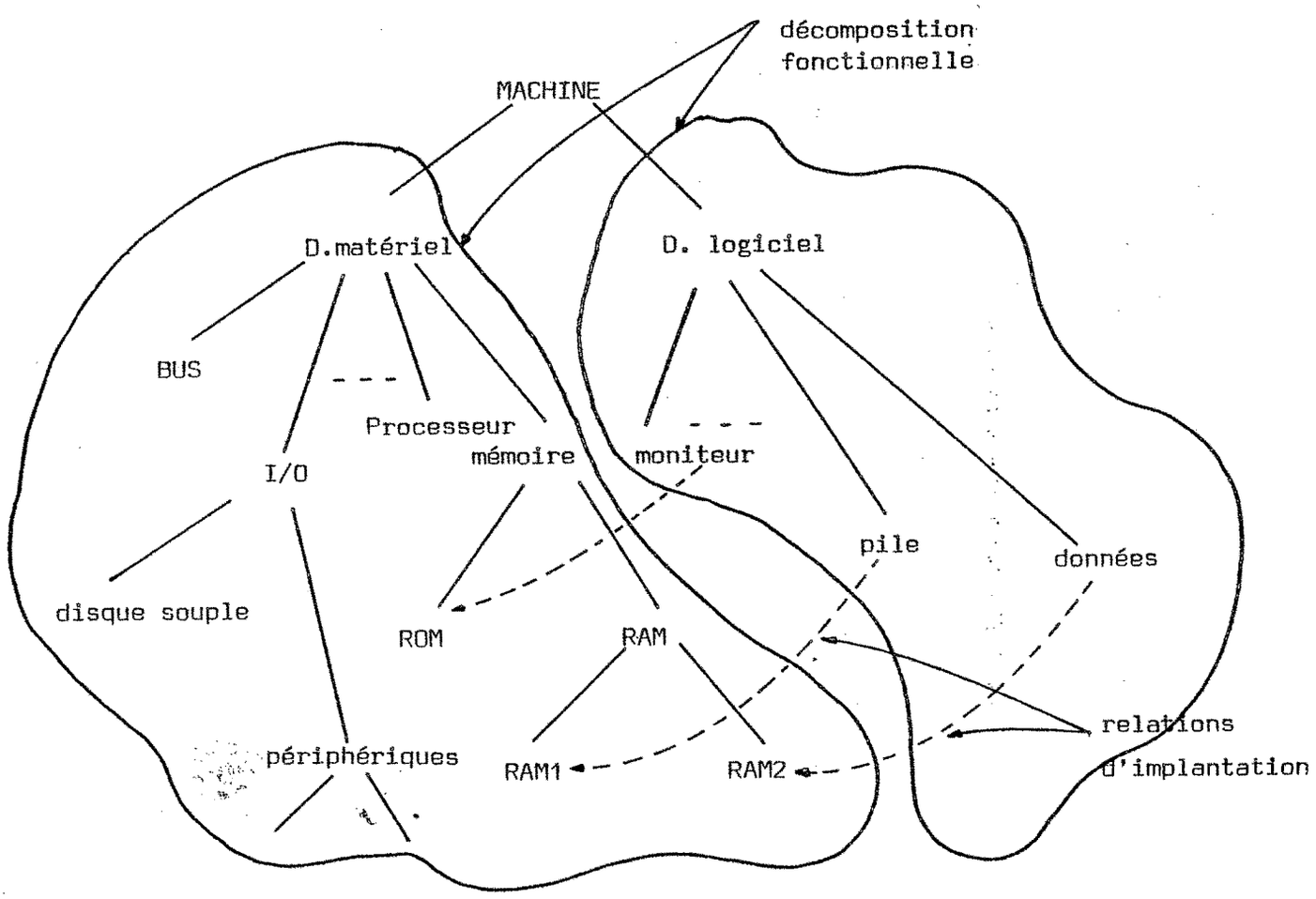


Figure 3.3 Description d'une machine en MICROD.

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.

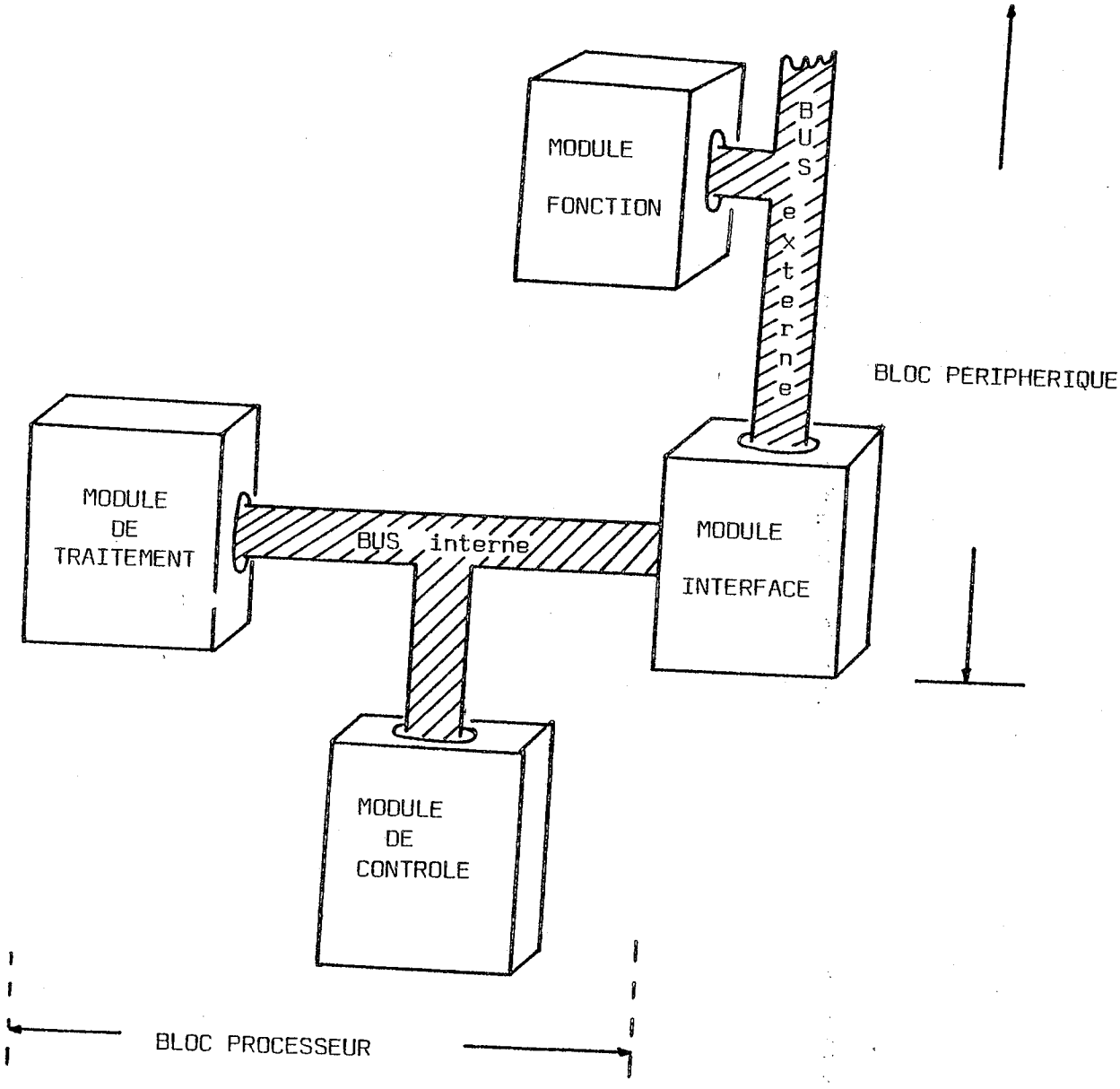


Figure 3.4 Les différents modules matériels d'un micro-système.

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> PARMI <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)

<générateur des phases>::= (matériel qui génére les signaux de travail du processeur)
<synchronisation>::= (matériel pour synchroniser les signaux propres au système) CO par exemple les signaux de lecture/écriture avec les phases du processeur.

<générateur de pages>::= <page système>
<page utilisateur>
<page système>::= <sélection comp.contrôle>
<sélection comp.contrôle>::= <sélection mémoire interne>/
<sélection périphérie>/
<sélection mécanismes d'interruption>/....

<sélection mémoire interne>::= <sélection superviseur>/
<sélection variables de travail>;

<pages utilisateur>::= (ensemble de pages en dehors de la page système)

5.1.1.2) Le module de contrôle.

Dans ce module, on place les éléments de base du système: superviseur, mémoires de travail, mécanismes de surveillance, circuits périphériques généraux (indépendants de l'application).

Sa représentation en MICROD, est la suivante:

<module de contrôle>::= <contrôle> <surveillance>
<contrôle>::= <mémoire interne>* <périphérie>* <commutat>
<mémoire interne>::= <mémoire de travail> /
<mémoire en lecture seulement> CO superviseur.

<mémoire de travail>::= <adresse d.RAM> <adresse f.RAM>
<mode.opérat.RAM> CO mémoire d'accès aléatoire.
<mémoire en lecture seulement>::= <adresse d.ROM>
<adresse f.ROM>
<mode.opérat.ROM>;

<périphérie>::= <ACIA>/<PIA>/<SIO>/.. CO périphérie de dialogue opérateur.
<commutat>::= Ɛ/<commutateur>;
<surveillance>::= Ɛ/<protect>*;
<protect>::= <protection mémoire>/
<protection d'entrée dans le système>;

<commutateur>::= (matériel pour la mise en/hors service du superviseur);
<mode.opérat.RAM>::= (lecture ou écriture); CO R/W;
<mode.opérat.ROM>::= (lecture);
<protection mémoire>::= (matériel pour le contrôle des accès dans une zône mémoire protégée);

<protection d'entrée dans le système>::= (matériel pour le contrôle des entrées dans le système); CO par exemple empécher:

- l'arrivée des interruptions de l'extérieur,
- l'arrêt du processeur,
- la sélection des composants du bloc interne;

Il faut remarquer que les fonctions des modules de traitement et du module contrôle dépendent d'une part de la machine (micro-système, macro-système,..) et d'autre part de la technologie du moment.

Nous pouvons en citer quelques exemples : le Motorola 6801, l'INTEL 8048 qui contiennent toutes ces fonctions incorporées dans un même circuit.

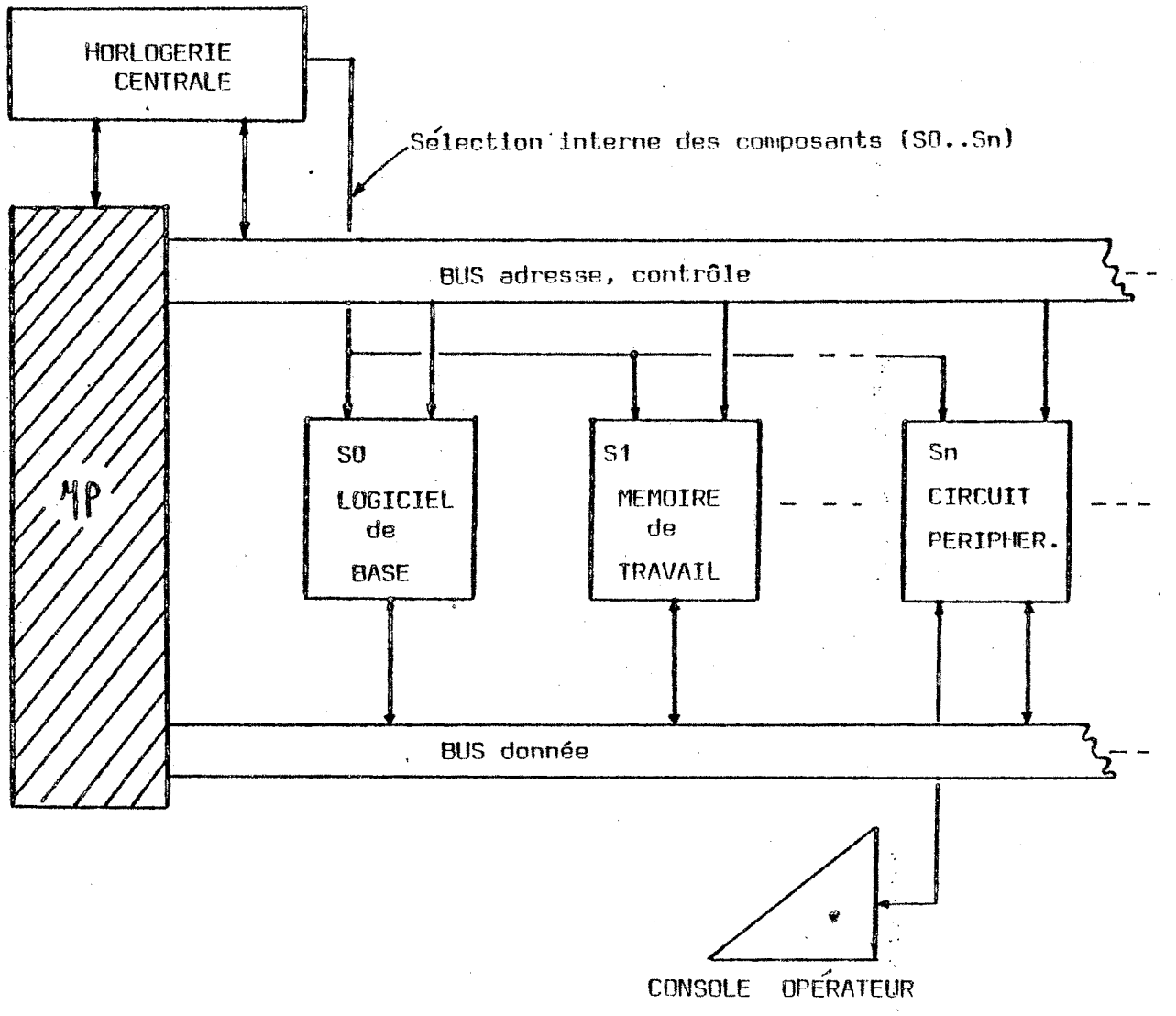


Figure 3.5 Organisation interne du bloc processeur.

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>::= ε/<bloc périphérique>;

<bloc périphérique>::= <module interface><BUS externe>
<module fonction>.

5.1.2.1) Le module interface.

Ce module est destiné a:

- contrôler le flux d'informations du bloc externe vers le bloc processeur relatif aux demandes suivantes:
  * arrêt du processeur: HALT,TSC,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>* <matériel spécialisé>;
<mémoires>::= <mémoires d'accès aléatoire>/
<mémoire en lecture seulement>/
<mémoire de masse>
<mémoire de masse>::= <contrôleur><unité.masse>*
<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,..

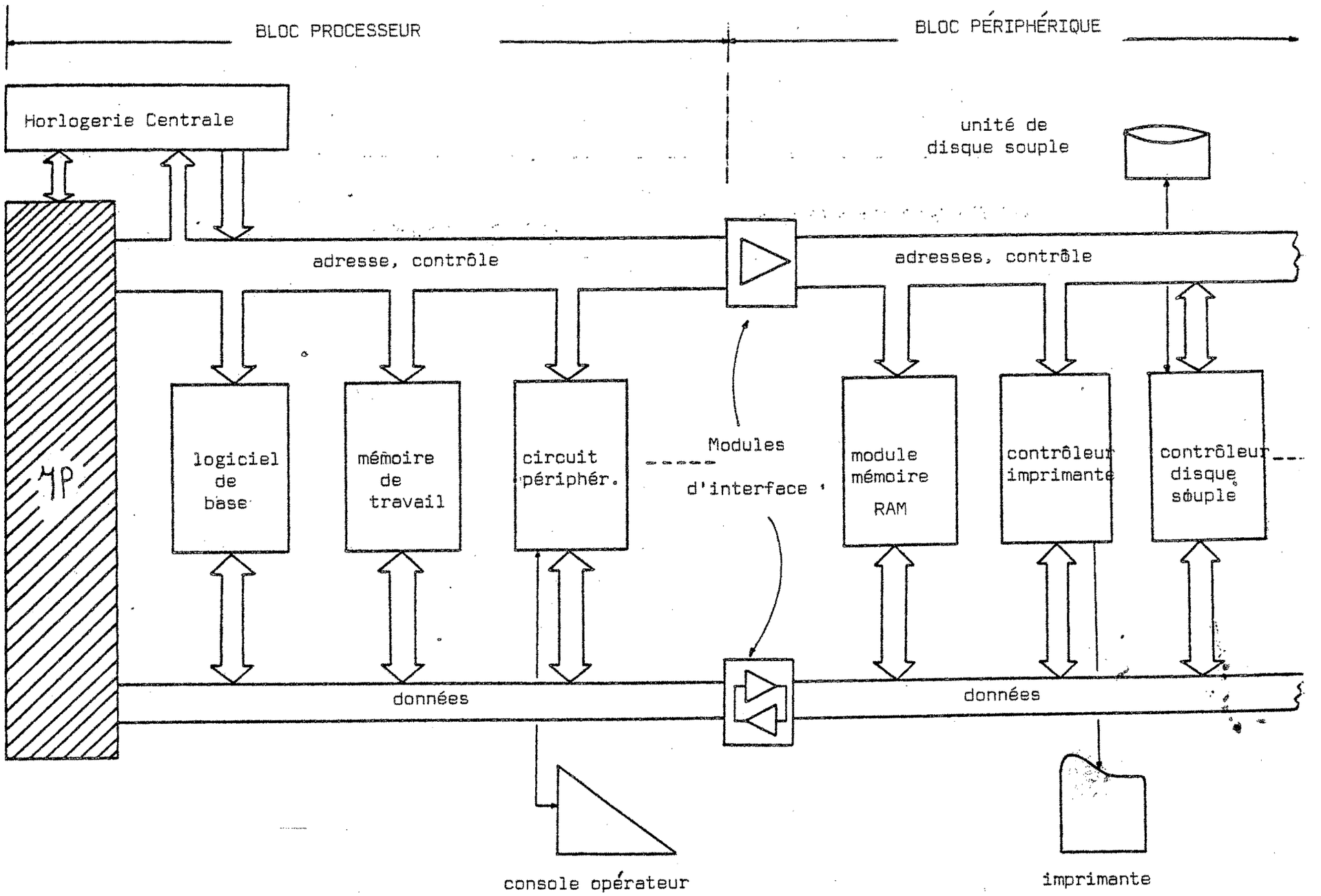


Figure 3.6 L'organisation de base d'un système de développement: un micro-système.

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.

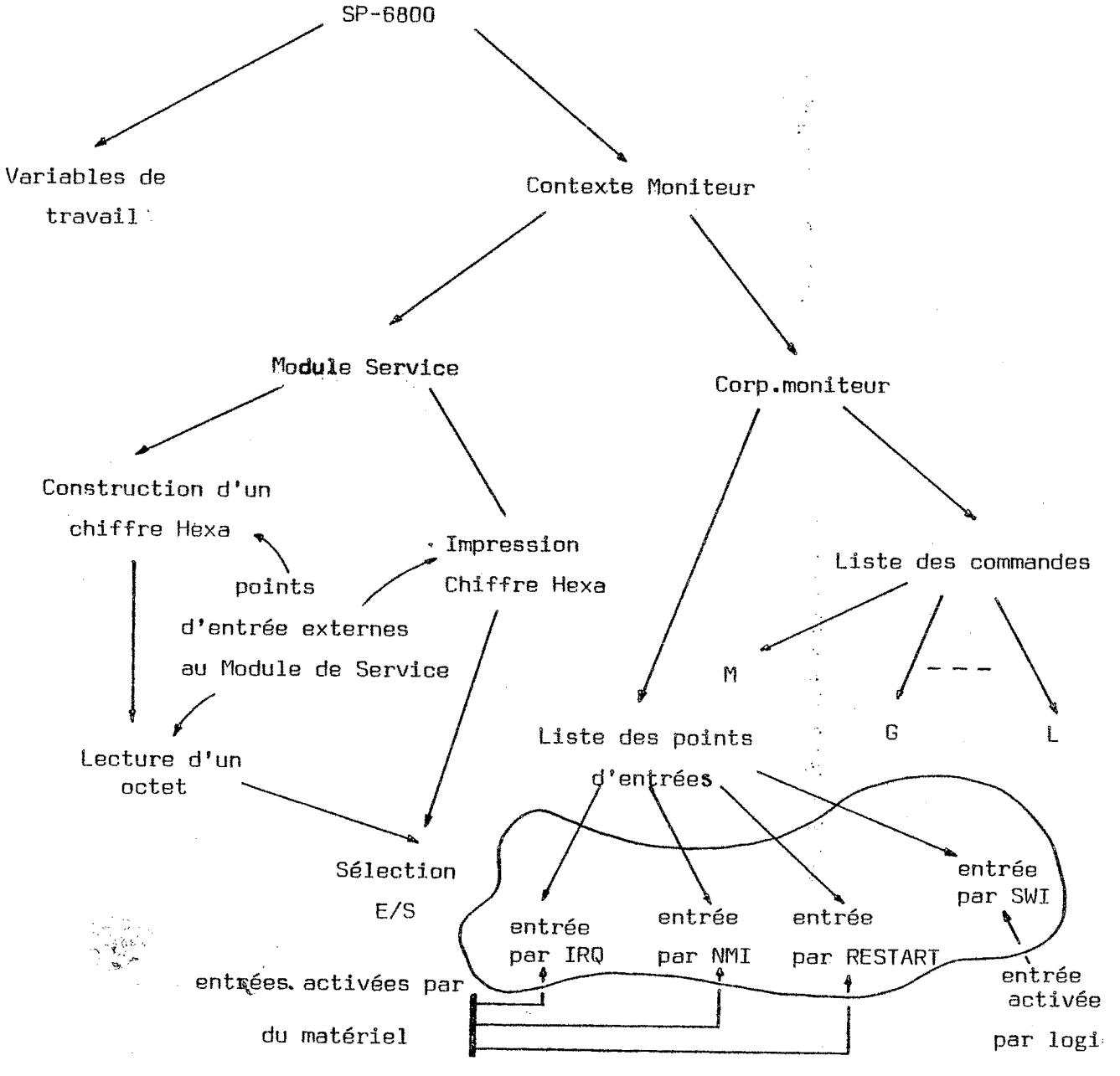


Figure 3.7 La description fonctionnelle du SP-6800.

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.

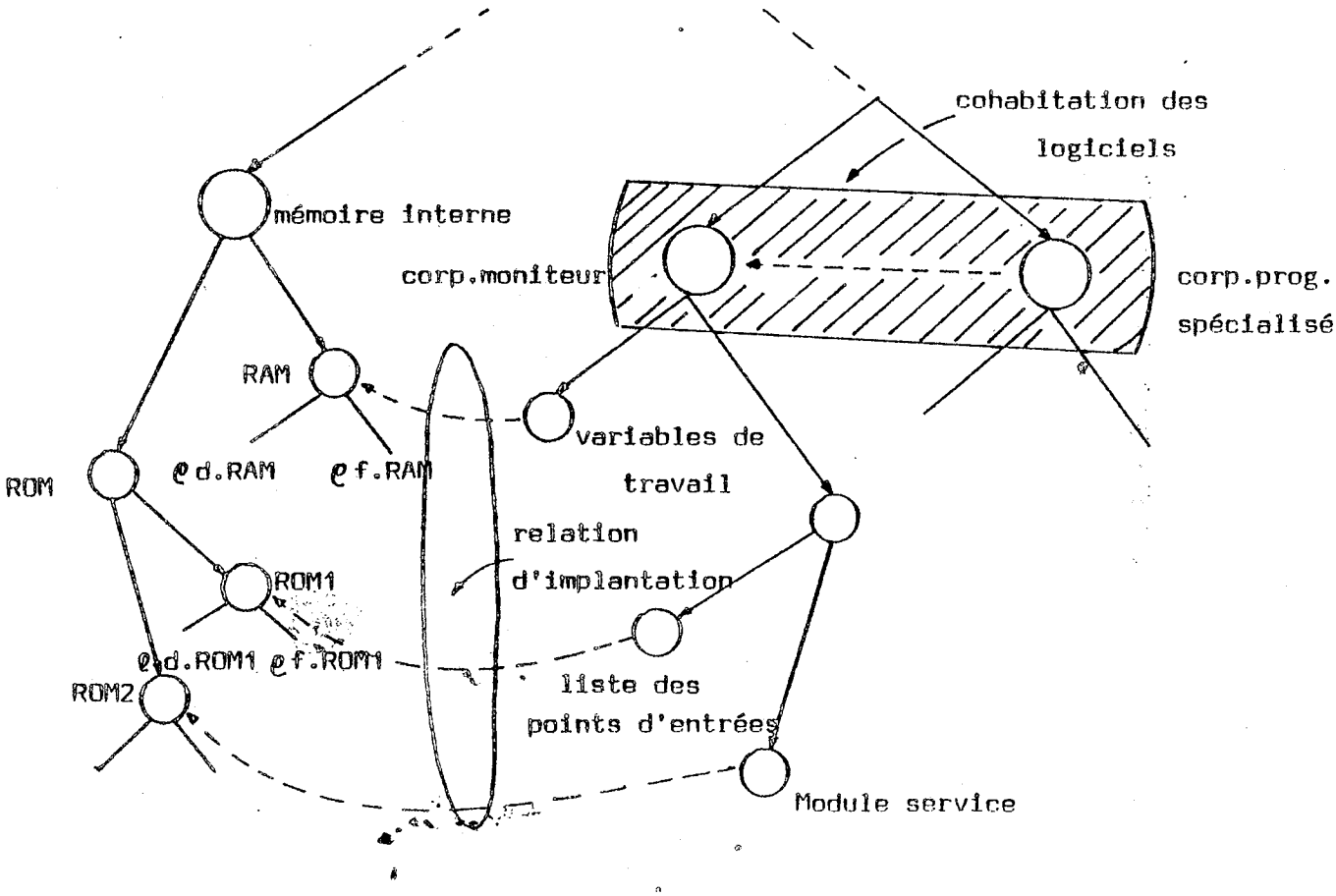


Figure 3.8 Les relations d'implantation dans un micro-système.

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>::= Σ/<élément comm>

<élément.comm>::= (élément partageable) CO 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.

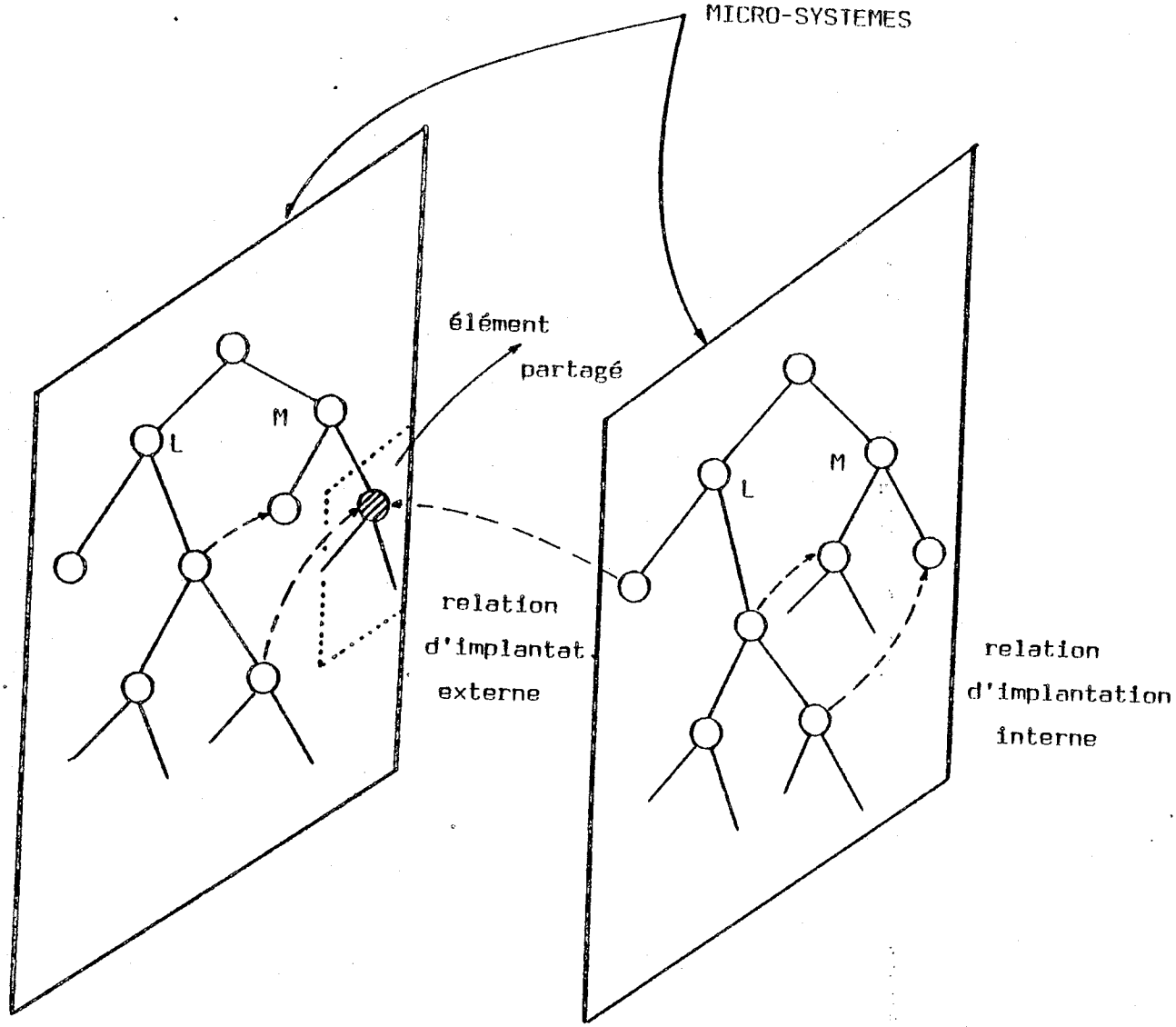


Figure 3.9 Le contexte des relations d'implantation d'un macro-système.

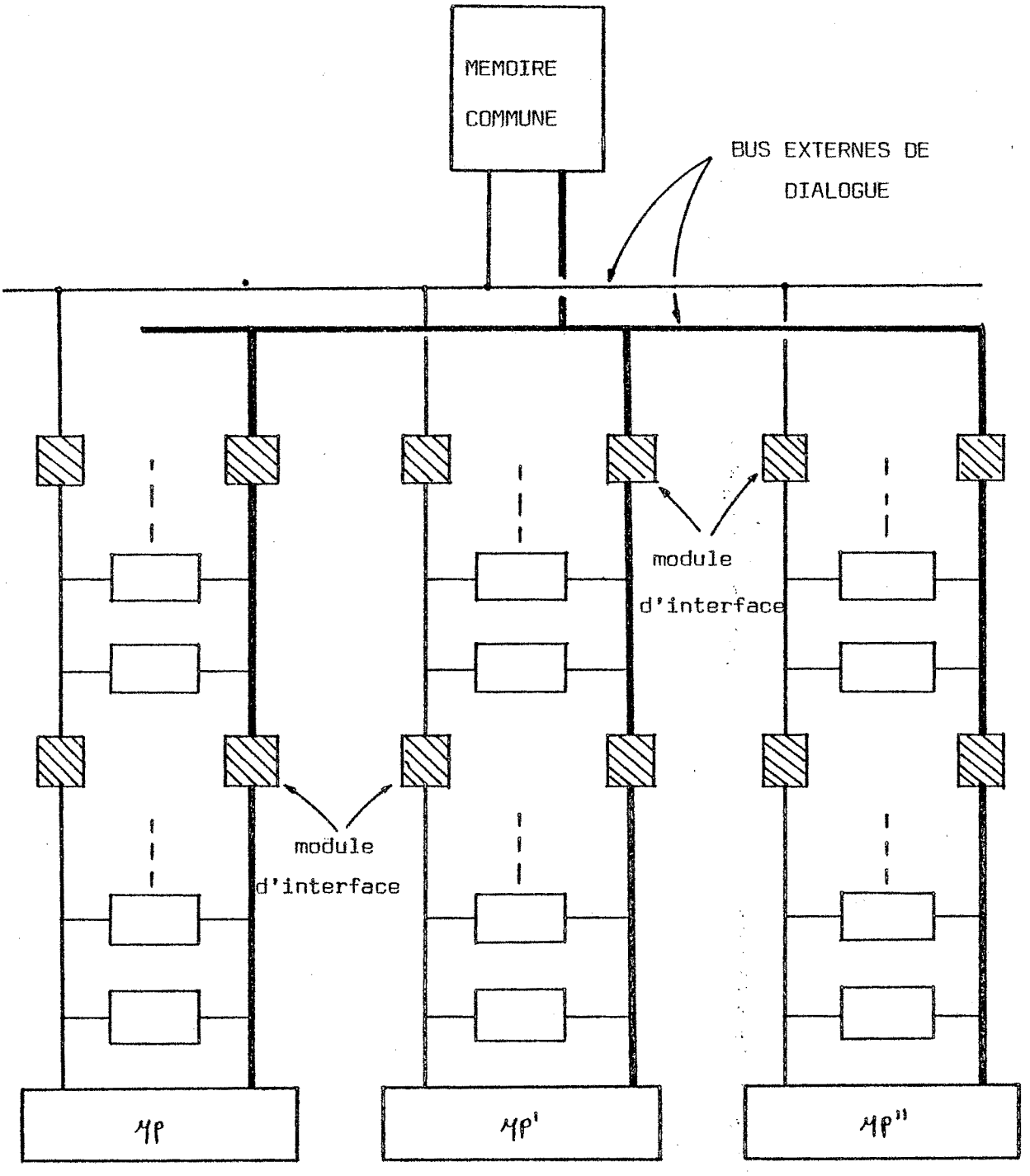


Figure 3.10 Exemple d'un macro-système: un réseau de microprocesseurs partageant une mémoire commune pour leur dialogue.

5.3) LA DESCRIPTION D'UN SYSTEME COMPLEXE.

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

<système complexe>::= <matériel-complexe>[←]<L.évolué>
<matériel-complexe>::= <macro-système><mult-complexe>*
<élément.comm> CO par exemple un <élément.comm> peut être un macro-système utilisé comme périphérique (processeur canal).
<mult-complexe>::= <liaison-macro><macro-système>
<liaison-macro>::= <liaison série> /
<liaison télé-informatique>.
<L.évolué>::= <L.macro>* ;

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.

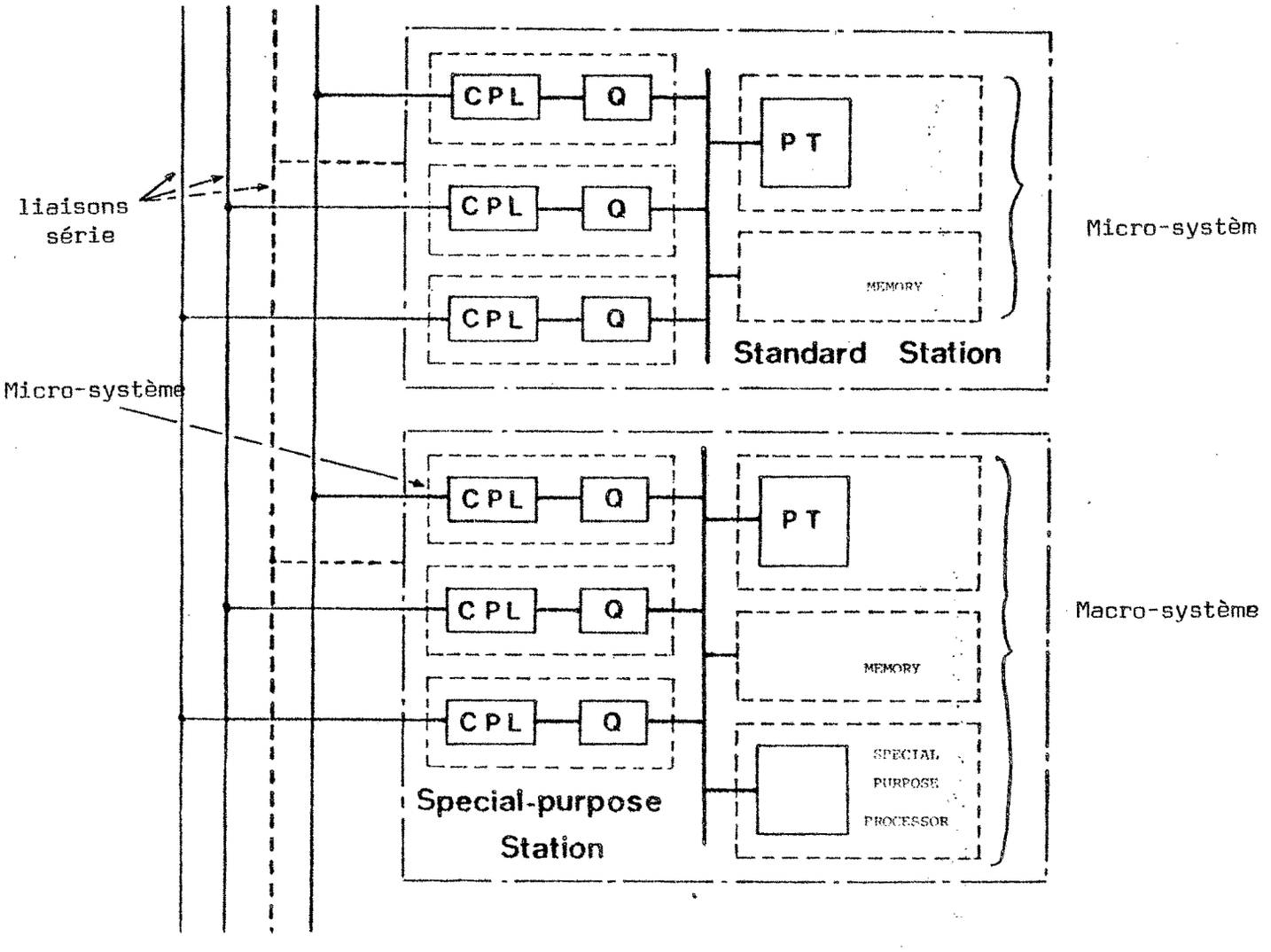


Figure 3.11 Un système complexe:commande répartie d'un autocommutateur téléphonique (CMP-79).

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 ammené 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

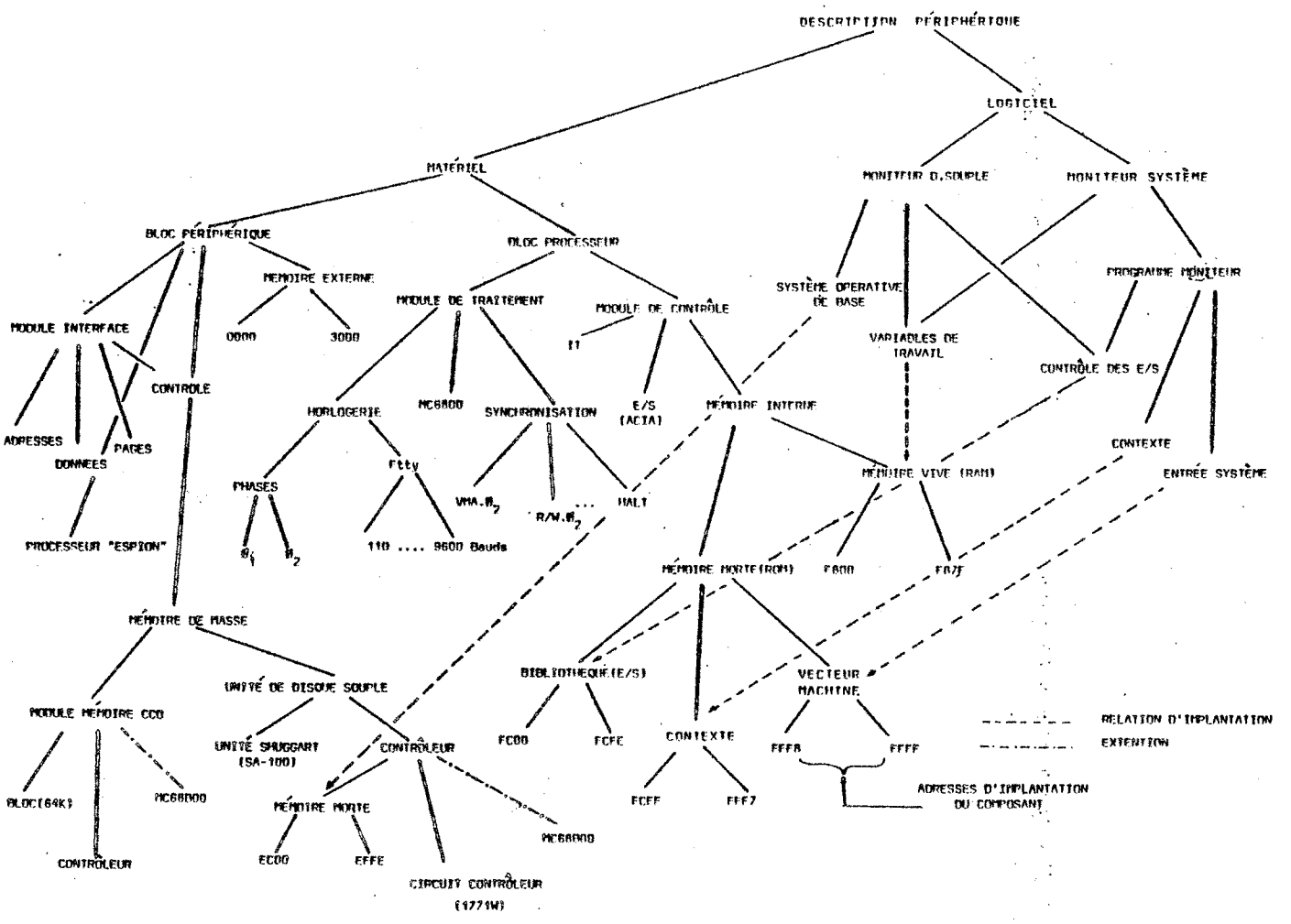


Figure II.1 Description fonctionnelle du périphérique.

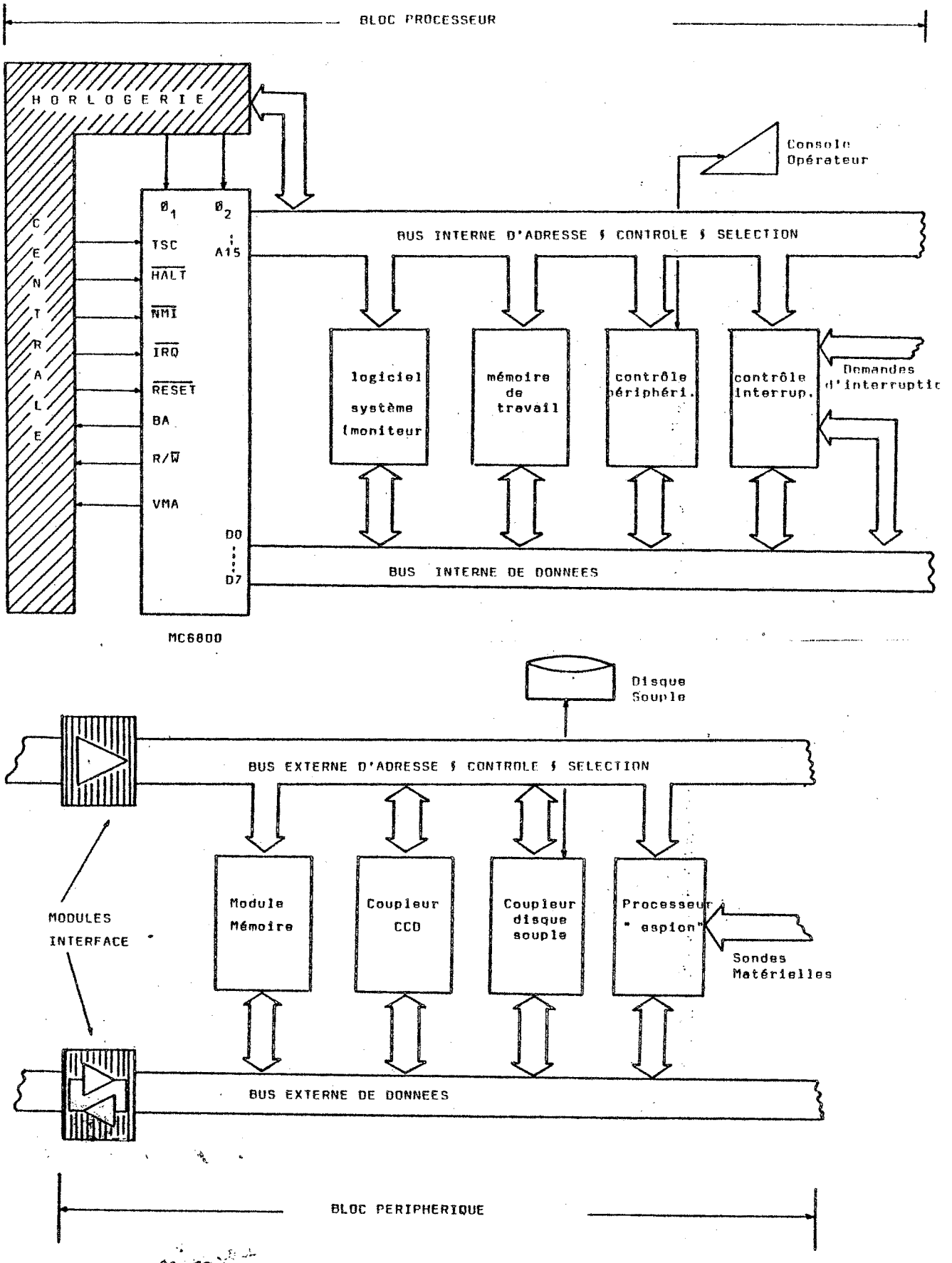


Figure II.2 L'architecture du périphérique.

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.

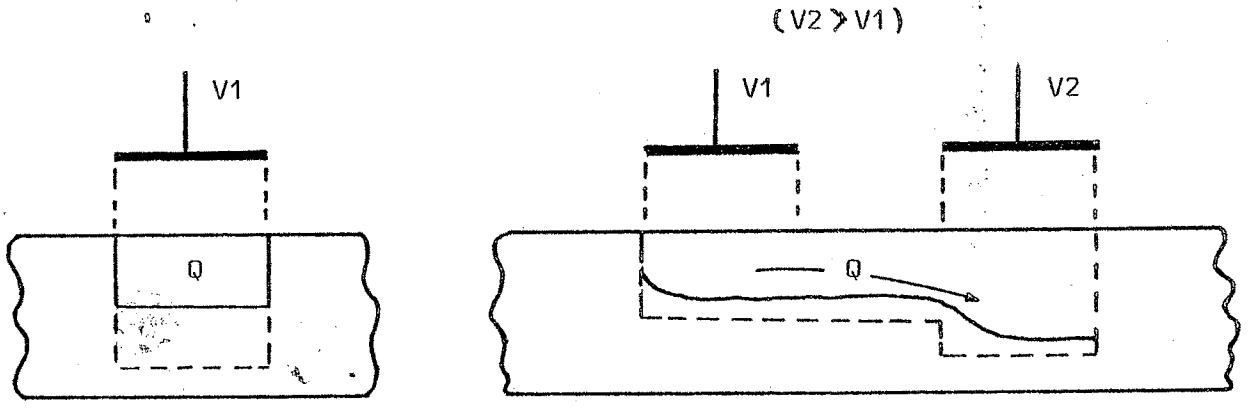
La mémoire CCD, développée dans les laboratoires de la BELL TELEPHONE COMPANY, se présente comme un registre à décalage. Celui-ci est construit à partir d'une série de capacités (Metal Oxide Semiconducteur) dont la charge se propage d'un élément à l'autre sous l'effet de signaux d'horloge.

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.



a) Une cellule de stockage b) Interaction entre cellules voisines

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.

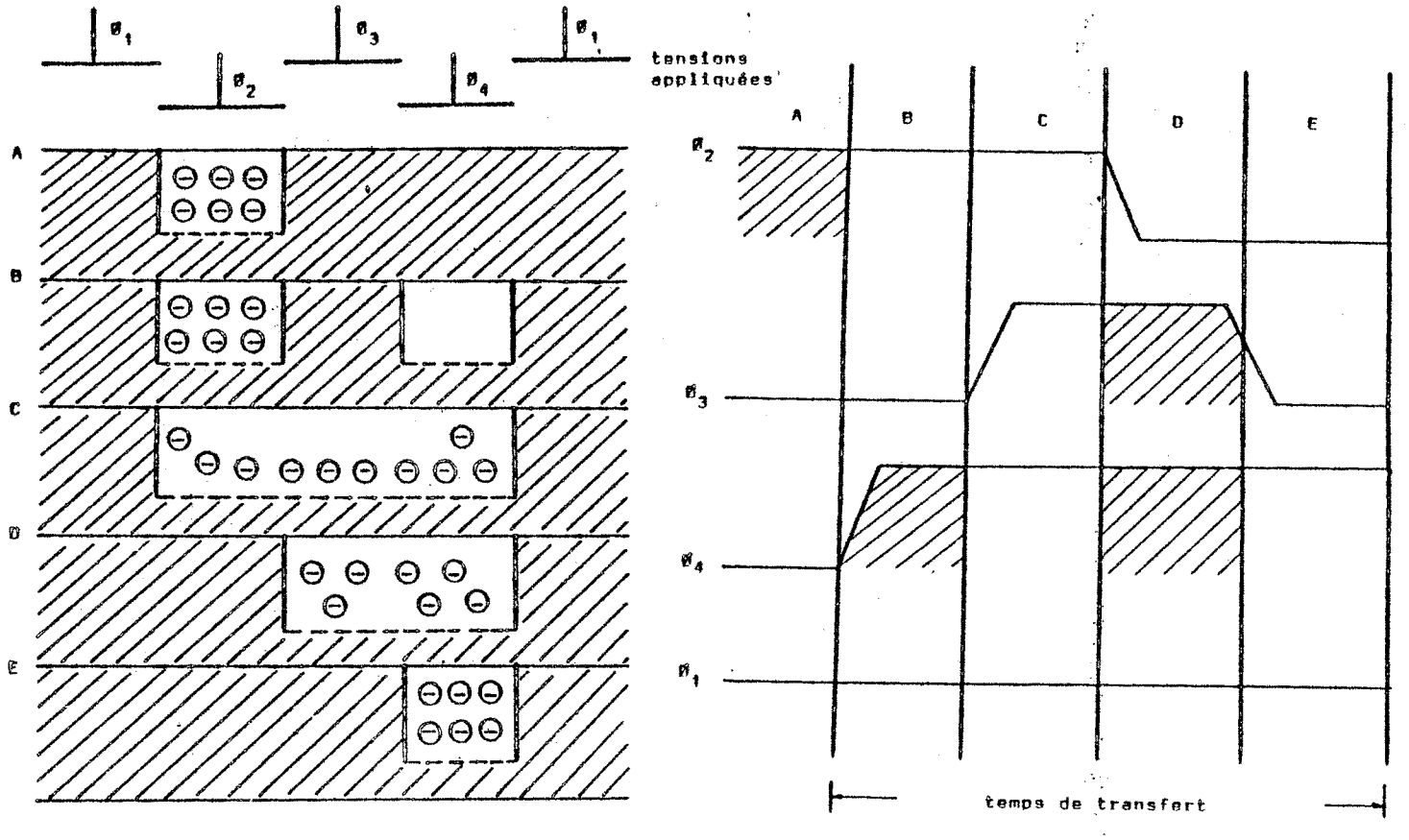


Figure 4.2 Le transfert de charges utilisant quatre phases d'horloge.

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 (Ø2) soit chargée(point A), puis dans un second temps, instant(B), les tensions sur les grilles Ø2 et Ø4 sont positives; sous la grille Ø4 se forme une autre cellule de stockage qui ne contient pas de charge.

A l'instant (C), la tension sur la grille Ø3 forme une cellule de stockage qui recouvre les autres cellules formées sous les grilles Ø2 et Ø4 à l'instant (B). Il existe une cellule de stockage continue qui comprend la cellule formée sous la grille Ø2, celle formée sous la grille Ø3 et celle formée sous la grille Ø4.

La chargé stockée dans la cellule sous la grille Ø2 est répartie jusqu'à la cellule sous la grille Ø4; cette répartition des charges dépend des tensions positives appliquées au point C. A l'instant (D), la tension appliquée à la grille Ø2 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 Ø3 devient négative et pousse les charges vers la cellule formée sous la grille Ø4. Le cycle peut alors se répéter, sous réserve de permuter les rôles des grilles Ø4 et Ø2, Ø1 et Ø3.

Le temps de décalage(SHIFT TIME) est la durée du transfet complet entre les grilles Ø2 et Ø4 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 te ps de rafraîchissement 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 extrêmité de la boucle.

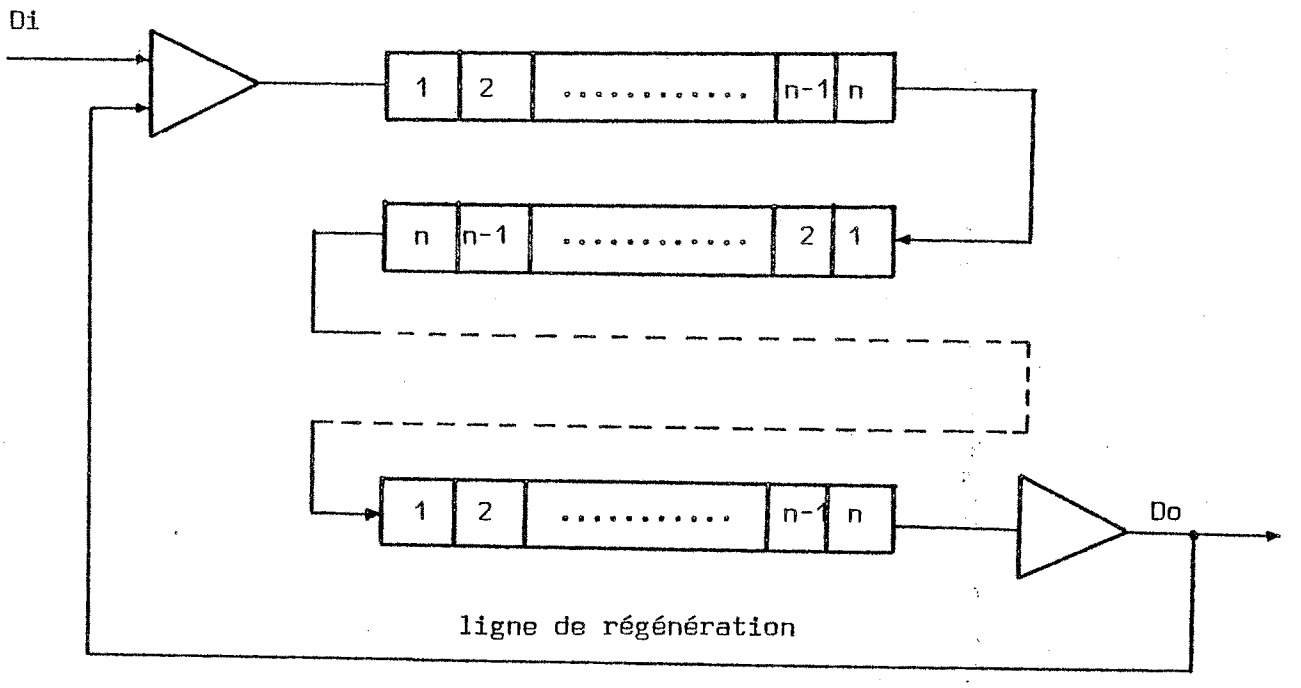


Figure 4.3 Organisation série.

La vitesse de transfet et le nombre de bits entre le point d'entrée (Di) et le point de sortie(Do) déterminent le temps maximun d'accès à l'information (tm) (figure 4.4).

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

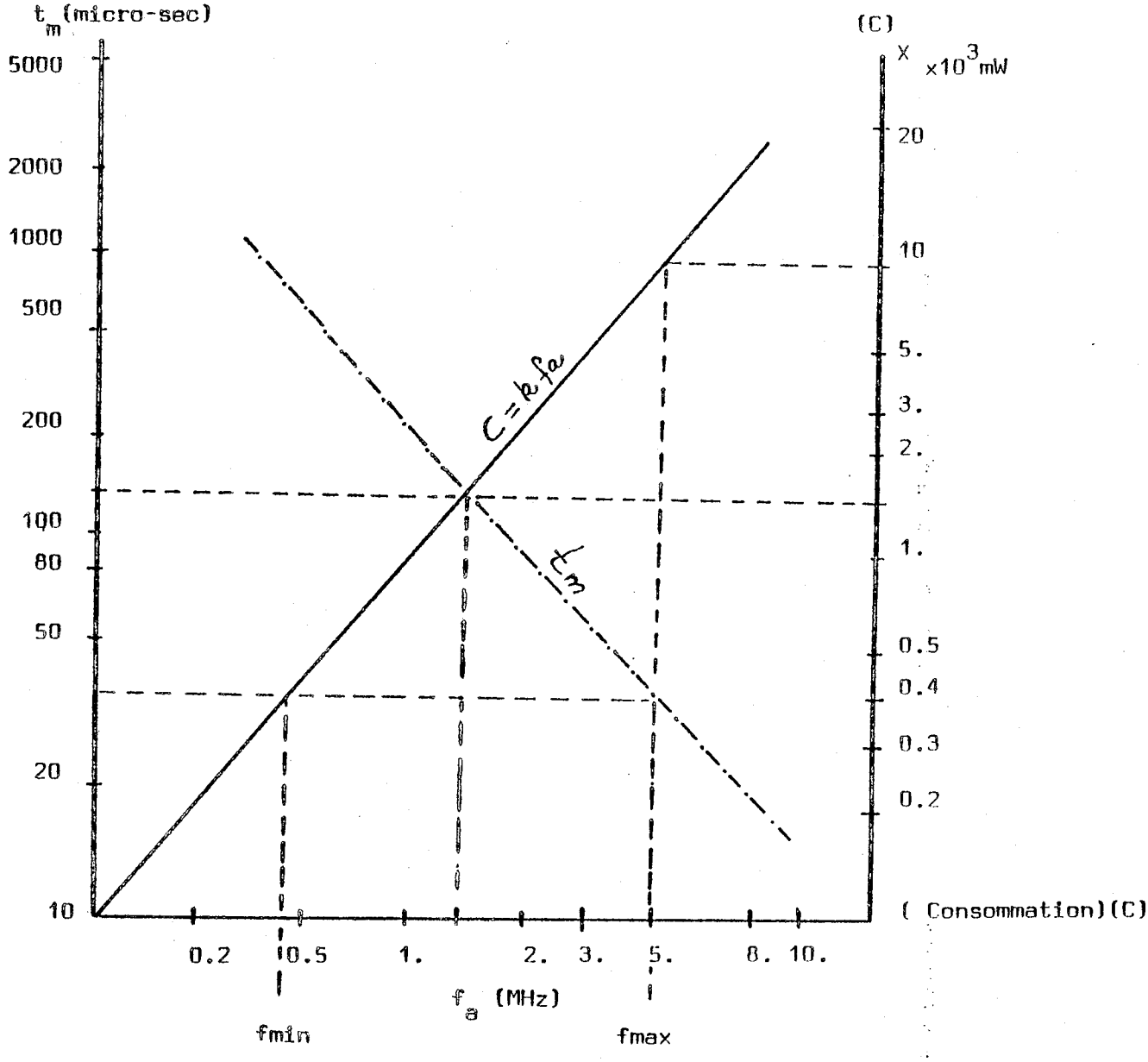


Figure 4.4 Relation entre le tm et la consommation(C) par rapport à la fréquence d'accès(RT5-75).

Remarque:

fmin et fmax représentent les fréquences minimale et maximale de travail des CCD. Elles dependent 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.

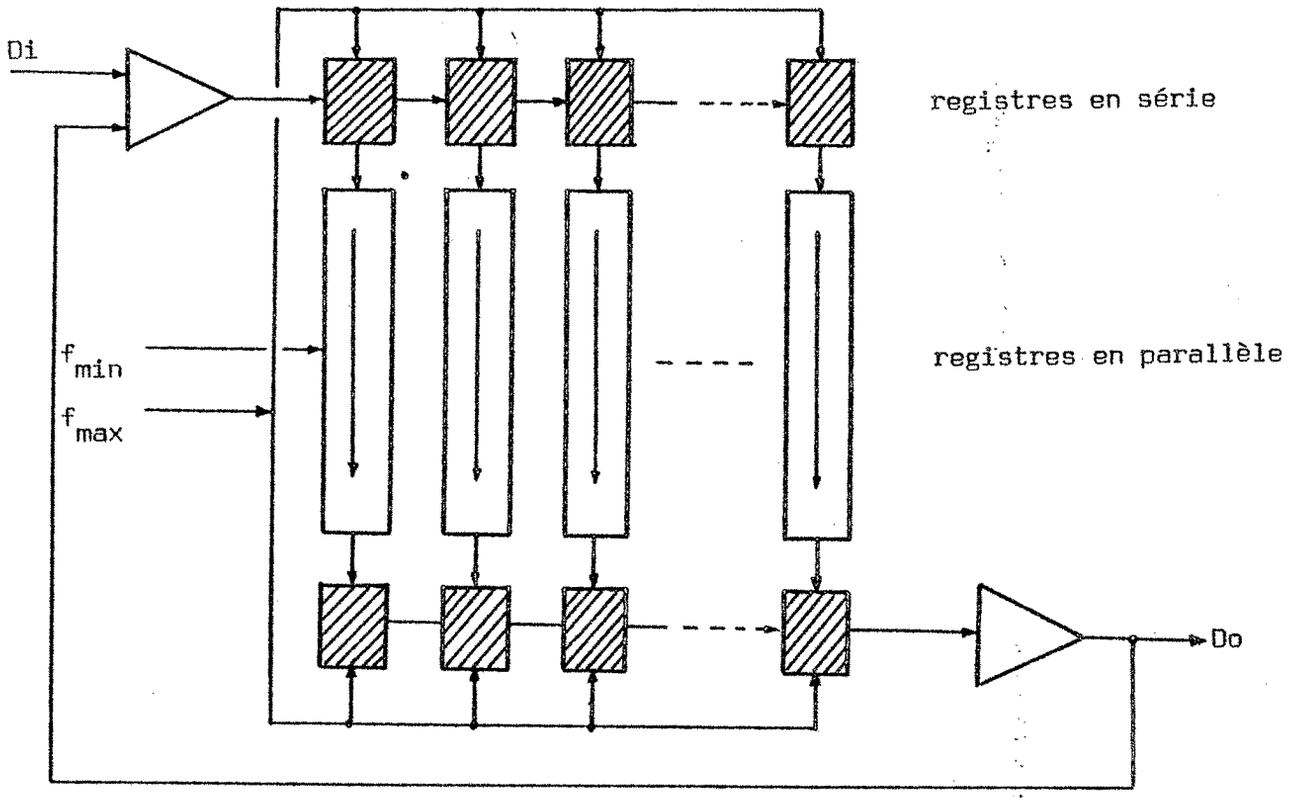


Figure 4.5 La configuration interne d'une organisation en SPS.

L'information est placée, en entrée et en sortie, à fréquence maximale (fmax) dans les deux registres à décalage série, puis l'information est décalée à fréquence minimale(fmin) dans les registres parallèles. Le temps d'accès (ta) 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 (ta).

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).

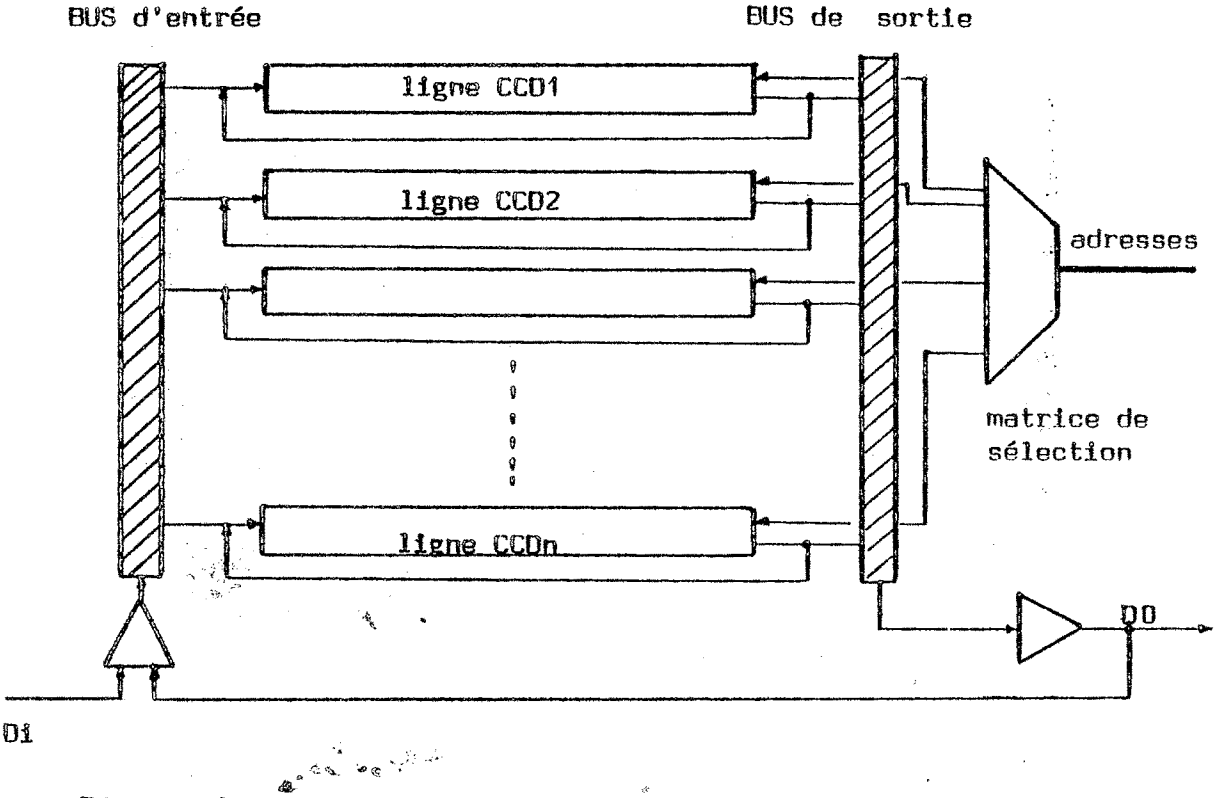


Figure 4.6 Organisation en LARAM.

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.

Caractéristique	Serpentine	SPS	LARAM
Capacité(bits)	16384	16384	16384
Organisation	256 x 16 x 4	4096 x 4	128 x 32 x4
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 rafraîchies périodiquement. Le rafraîchissement 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

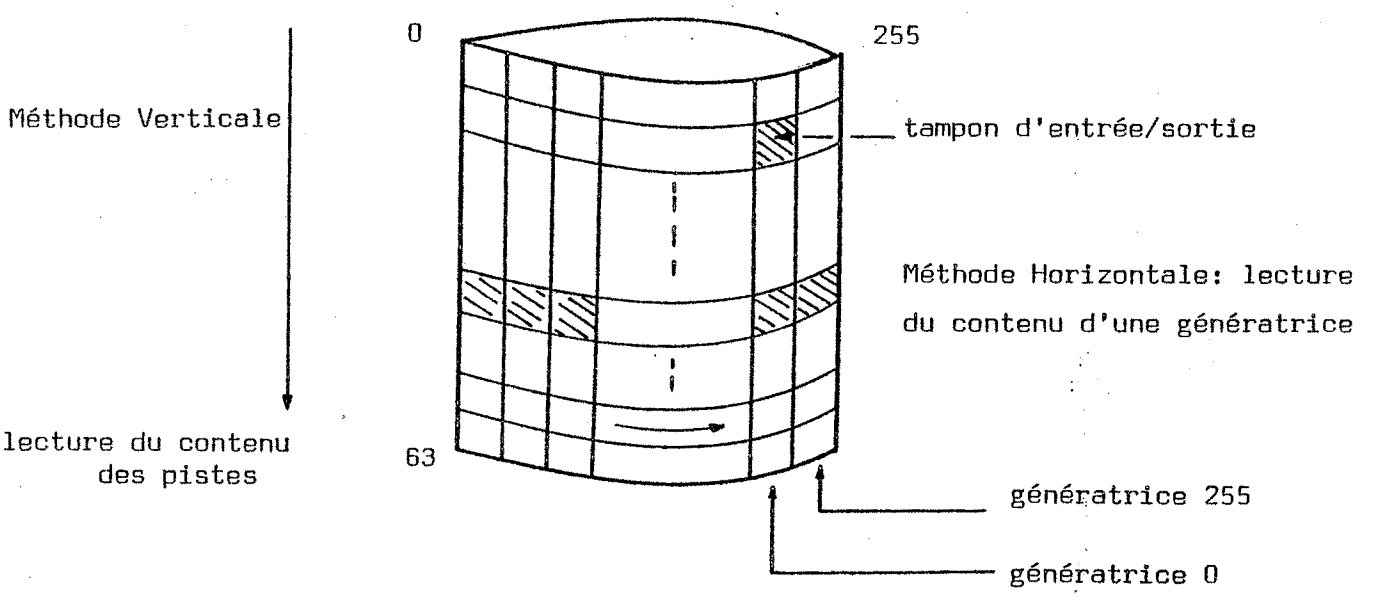


Figure 4.7 Organisation des CCD 2416.

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 controlée 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 sucessifs.

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 rafraîchissement de temps en temps. L'accès se fait donc par groupes séparés comme le montre la figure 4.8

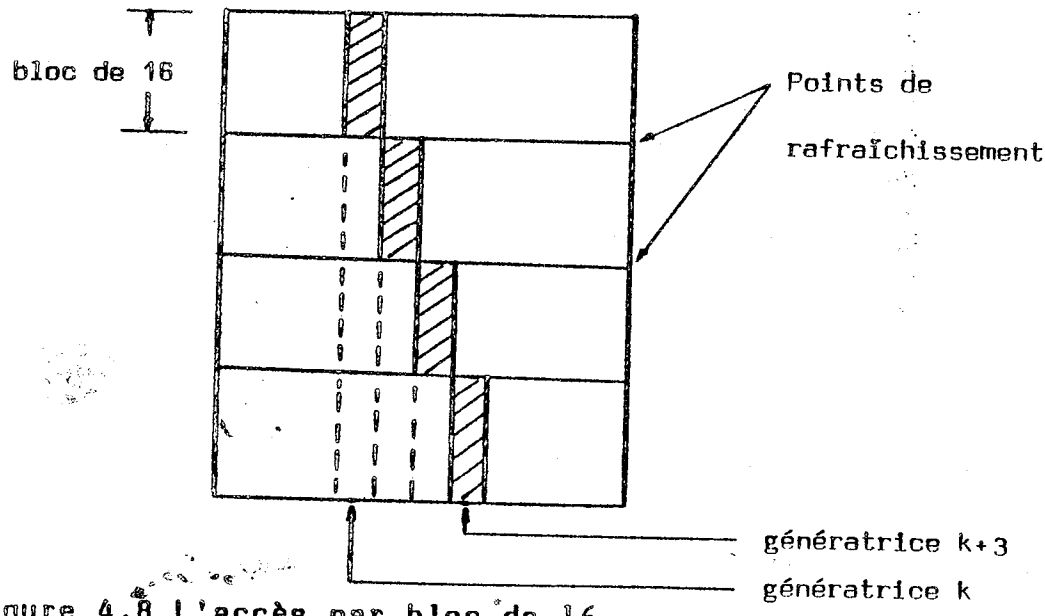


Figure 4.8 L'accès par bloc de 16.

Il faut remarquer qu'avec cette méthode la fréquence de rotation est minimale pour permettre un nombre maximun 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'accés élémentaire est de 460 ns,
- la durée d'un cycle de rafraichissement 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'intervale 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, lorqu'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.

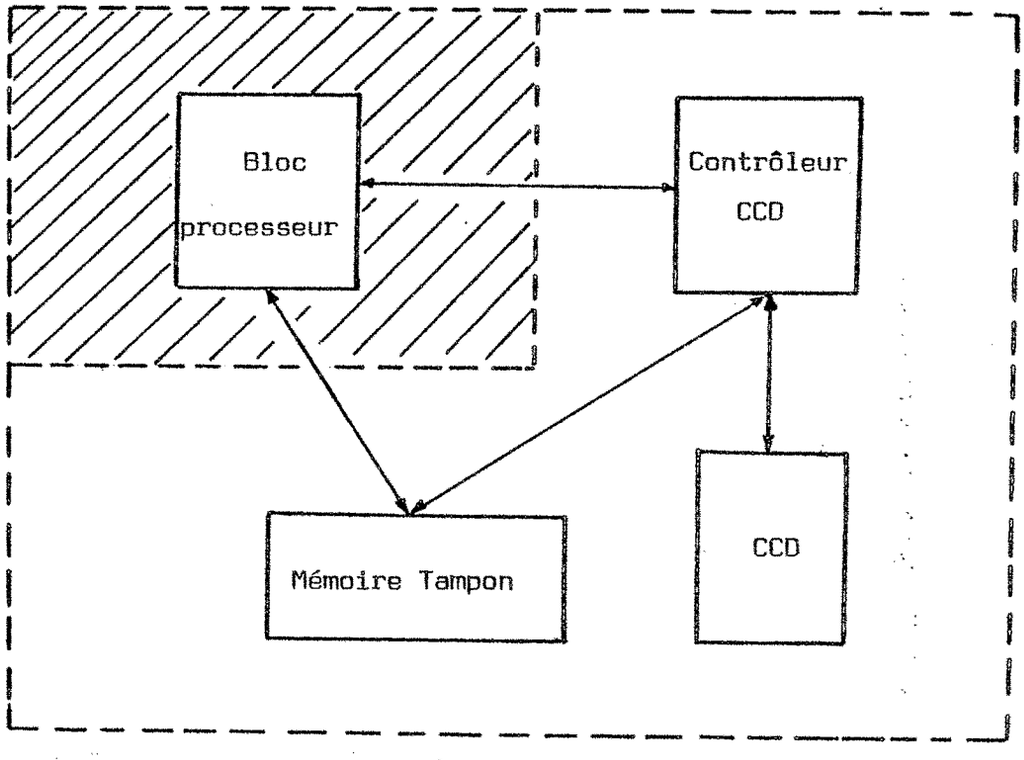


Figure 4.9 L'architecture du processeur CCD.

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.

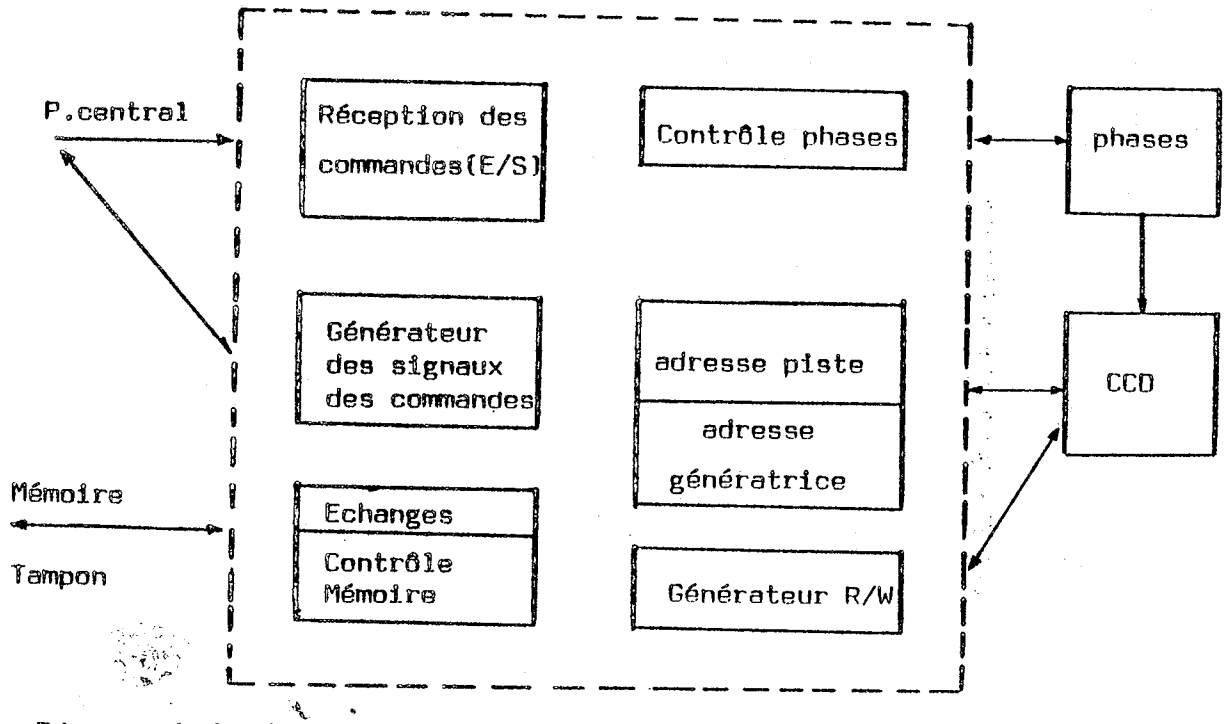


Figure 4.10 Le contrôleur CCD.

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.

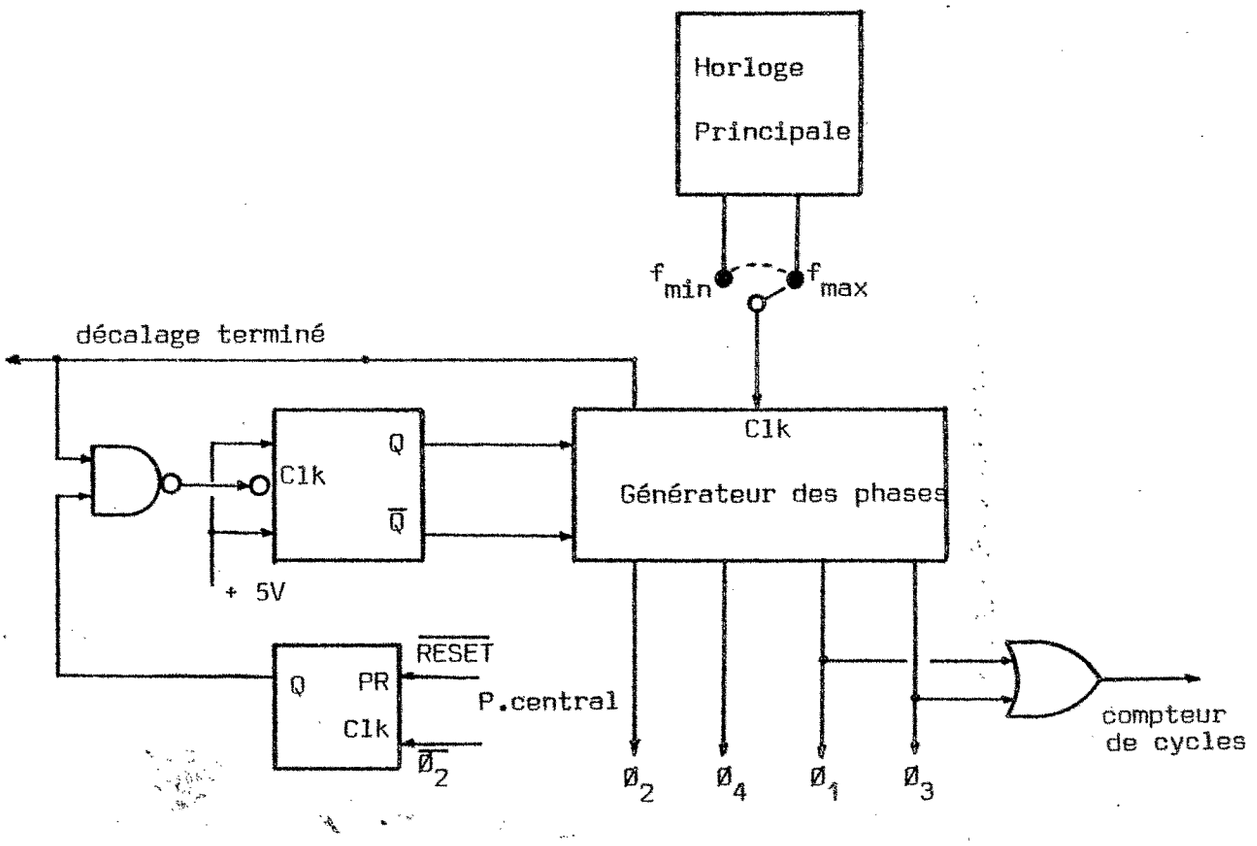


Figure 4.11 Le générateur de phases.

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 rafraîchissement;
     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 cablé 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 mèmoire. De plus il permet de:

- générer l'horlogerie de contrôle pour les CCD (phases),
- générer les ordres vers le P.central:
* arrêt ($\overline{HALT}$),
* 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 GENERATRICE.

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

```
PROCEDURE GENDAM,
DEBUT
CO adressé par génératrice;
     TANTQUE adr.gen.courant ≠ 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
     rafraîchissement; 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

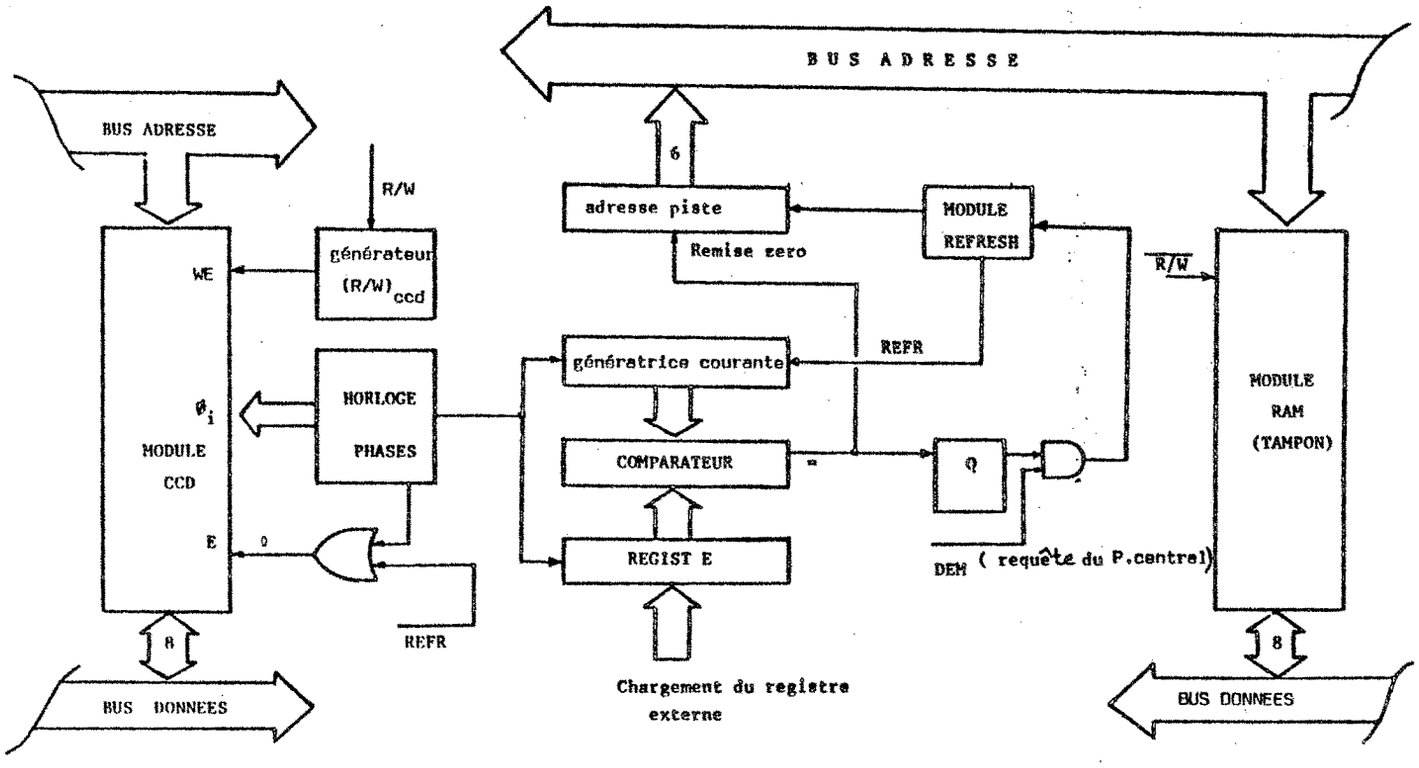


Figure 4.12 L'automate GENERATRICE.

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>

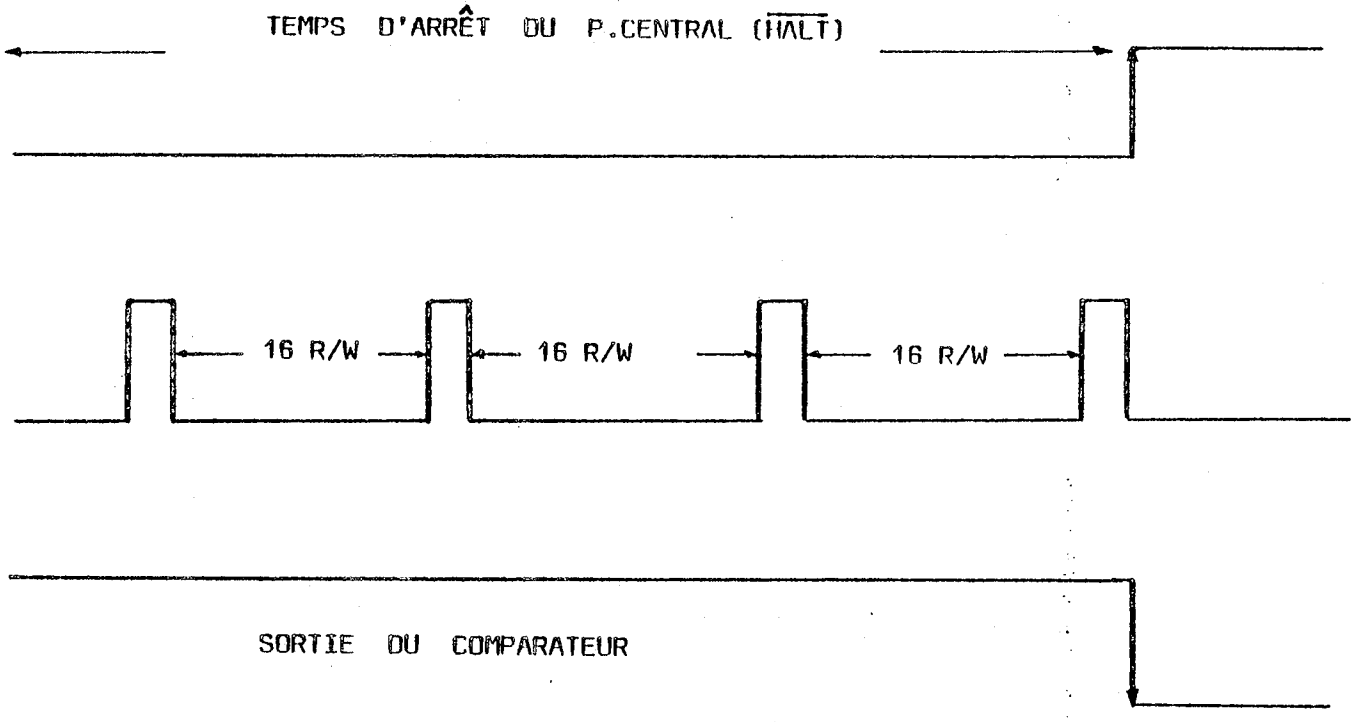


Figure 4.13 Les signaux de contrôle de cette méthode.

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.
```

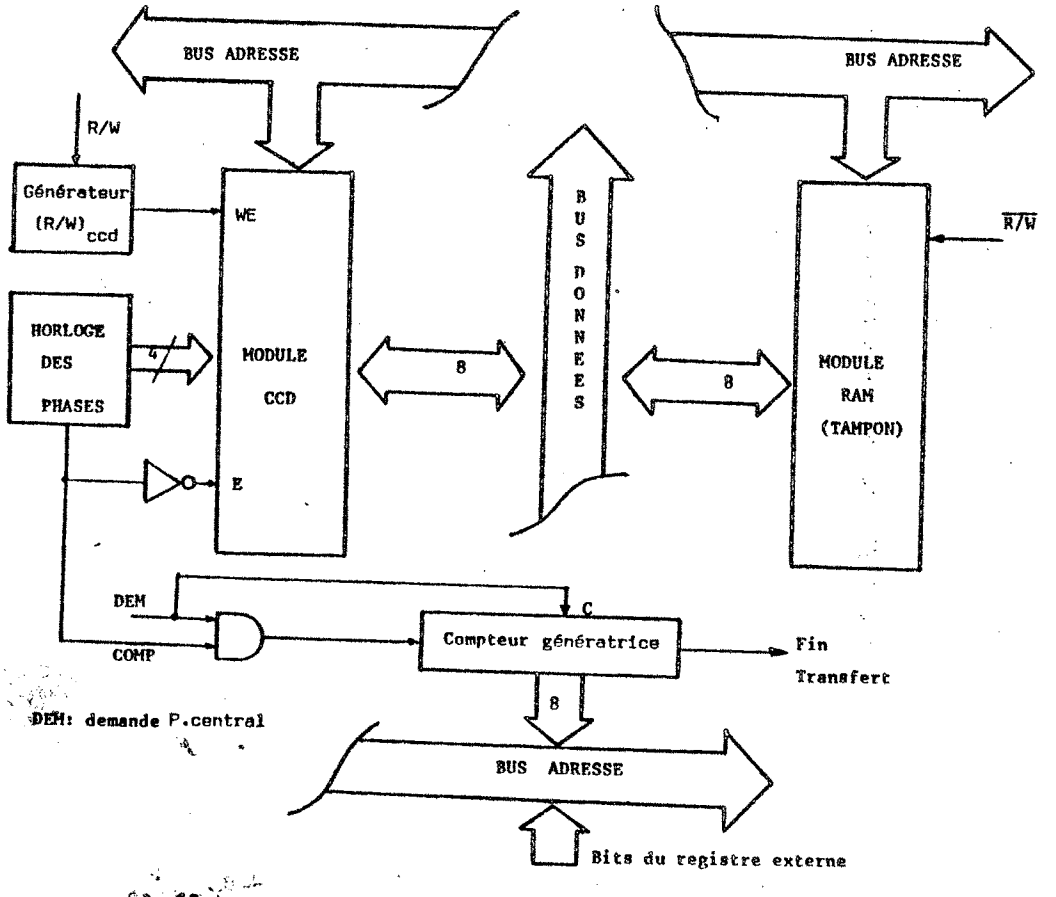


Figure 4.14 L'automate PISTE.

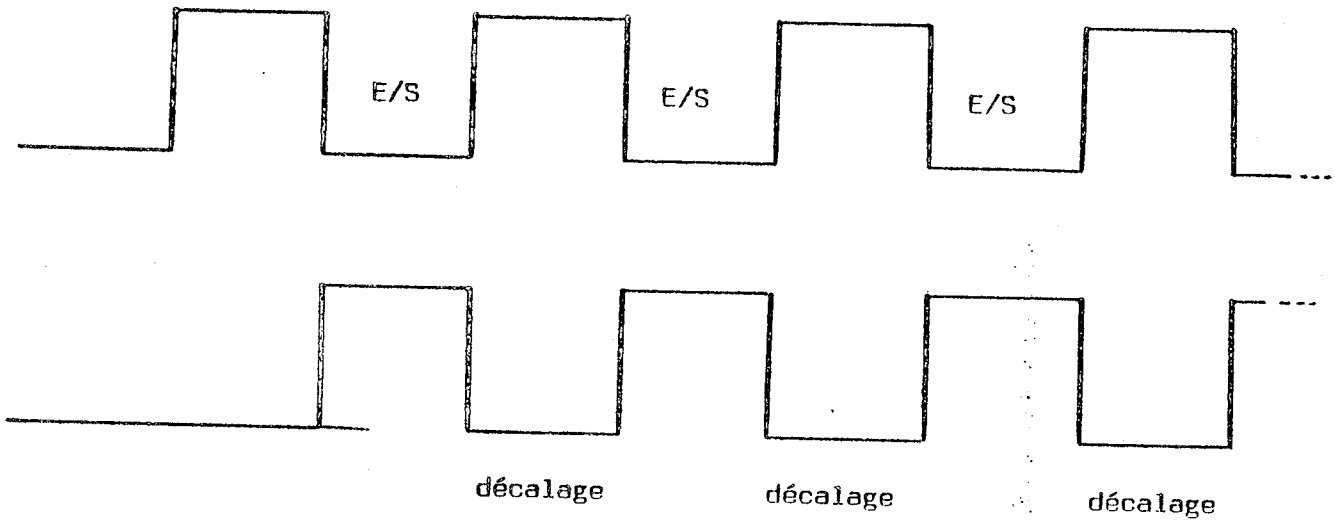
- <u>caractéristiques électriques</u>.

* 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
Fr	Minimale(55.55 Khz)	Maximale(666.66 Khz)
Di	1.77 Mbits/sec	1.74 Mbits/sec
N_a	16	1
Td/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
Di: débit d'information
F_r: fréquence de rotation
t_a: temps d'accès au bloc
$T_{d/2}$: temps entre décalages
t_R: temps de recherche
t_T: temps de transfert
N_a: nombre d'accès

Données.

$t_a = t_T + t_R$

t_T(génératrice) = 256(500 + 750)= 320 micro-sec

t_T(piste) = 4(572.56 + 9000) = 38.29 micro-sec

t_R(génératrice) = (1/2)t_T = 160 micro-sec

t_R(piste) = 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.

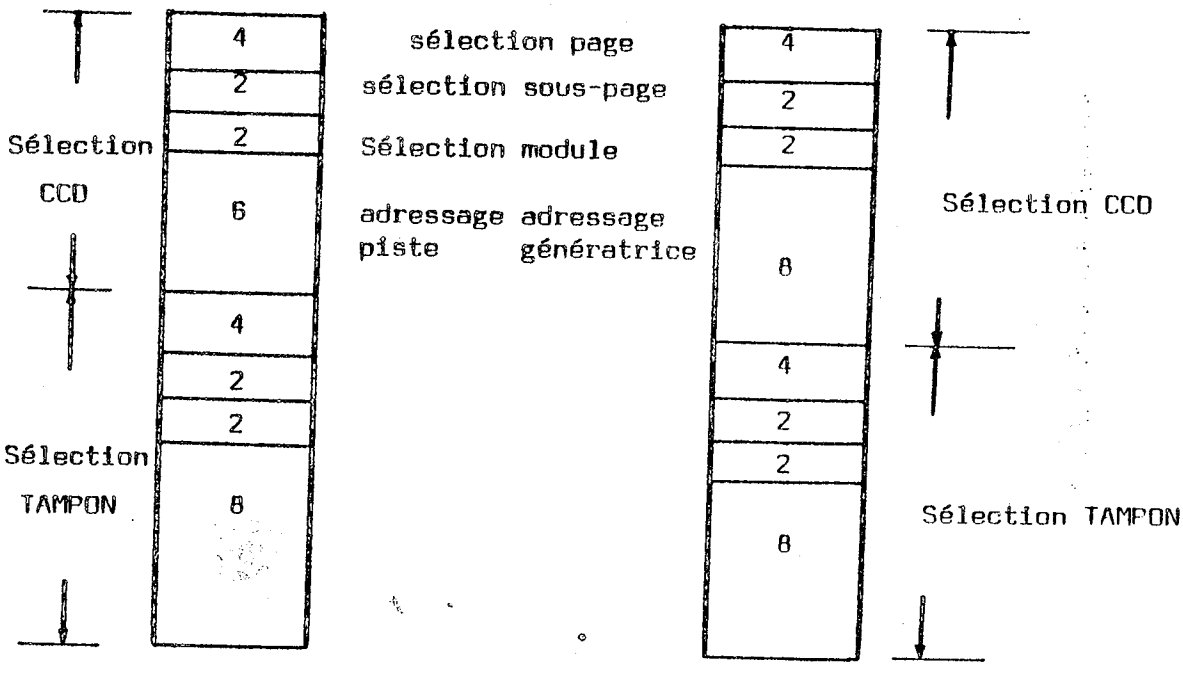


Figure 4.15 Les vecteurs d'adresses pour le cas génératrice/piste.

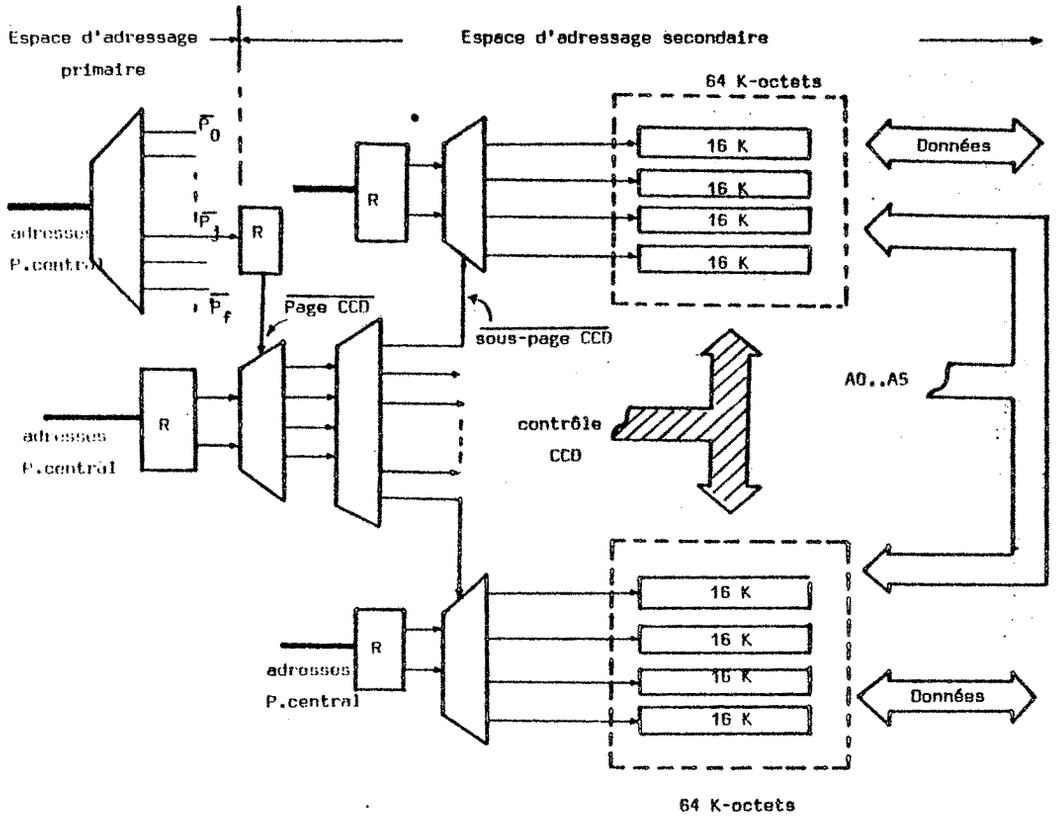


Figure 4.16 Organisation générale d'un système de mémoire CCD.

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

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 ($\overline{\text{HALT}}$); 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(fmax) laquelle est divisée par huit pour la génération des quatres 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(fmin=fmax/12), qui correspond aux 18000ns spécifiée par le contructeur.

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(fmin). Ce changement de fréquence est destiné à obtenir une consommation minimale des CCD.

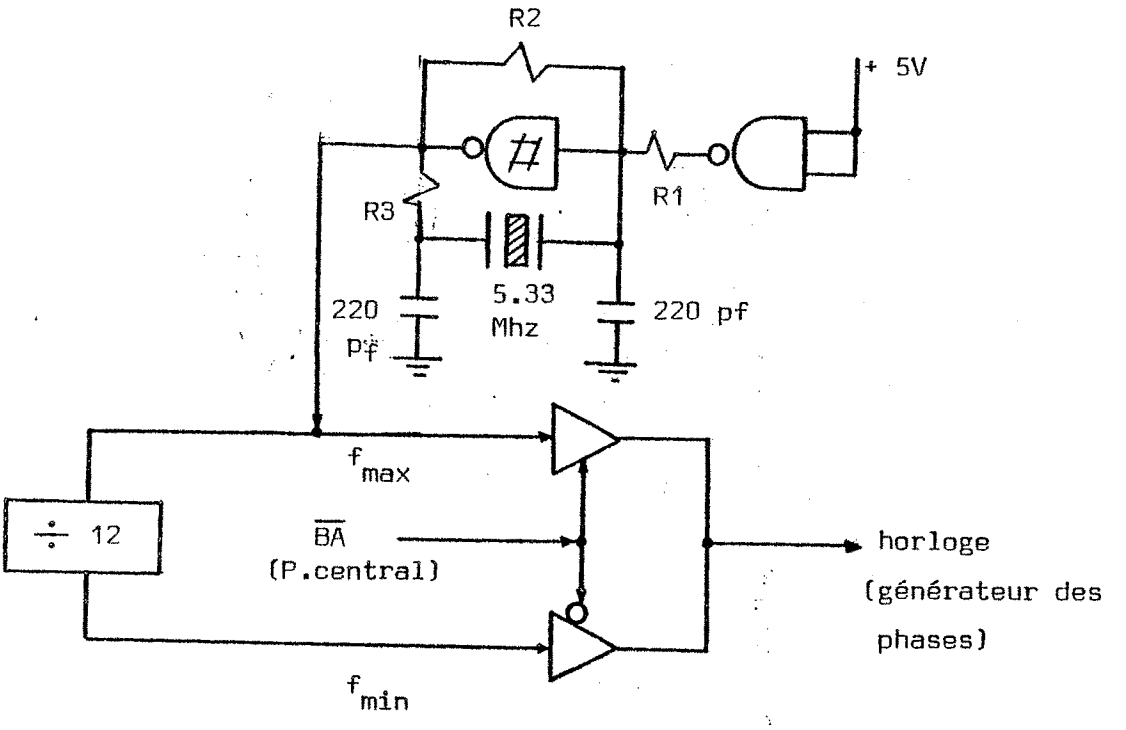


Figure 4.17 L'horloge maître.

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.

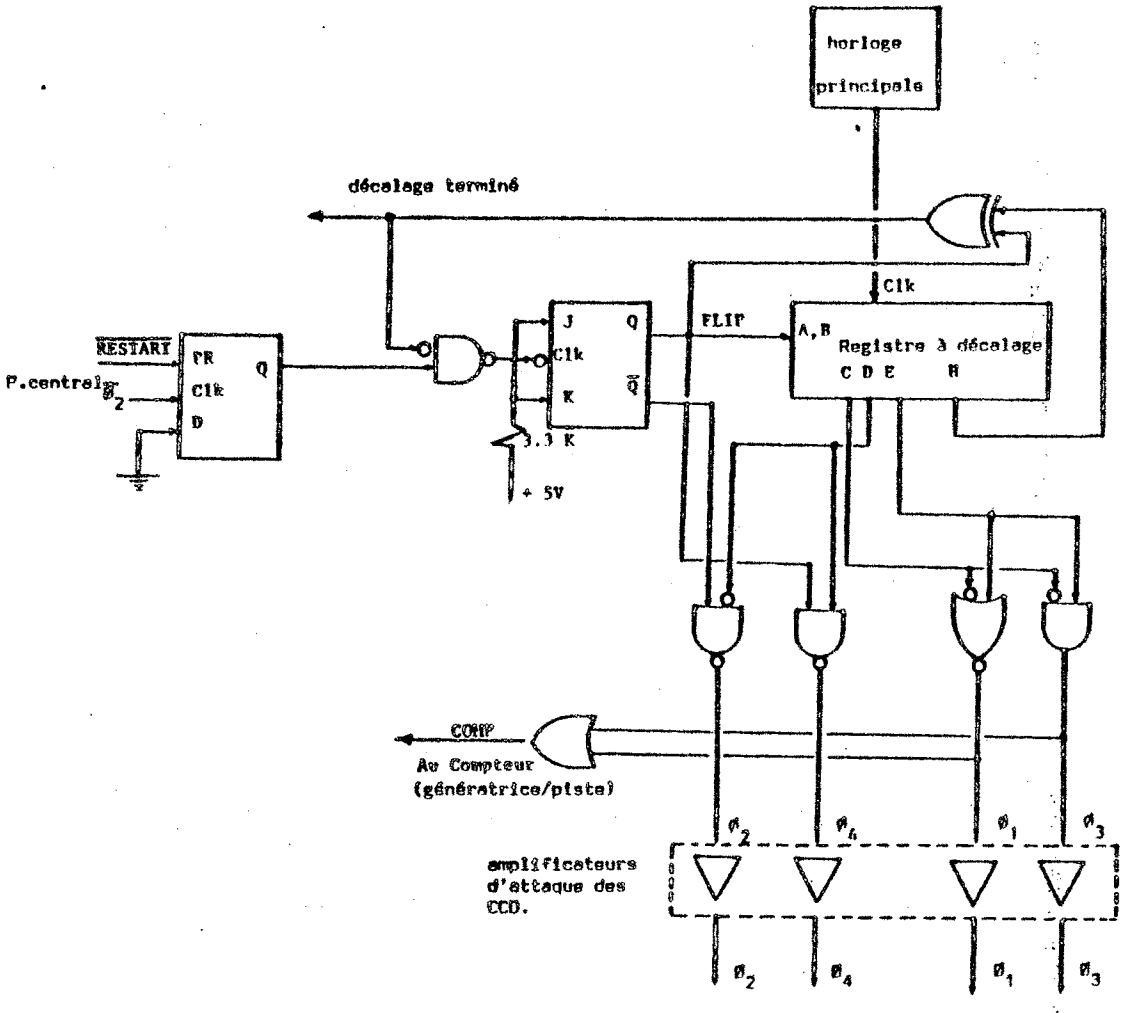


Figure 4.18 L'horloge de commande des CCD.

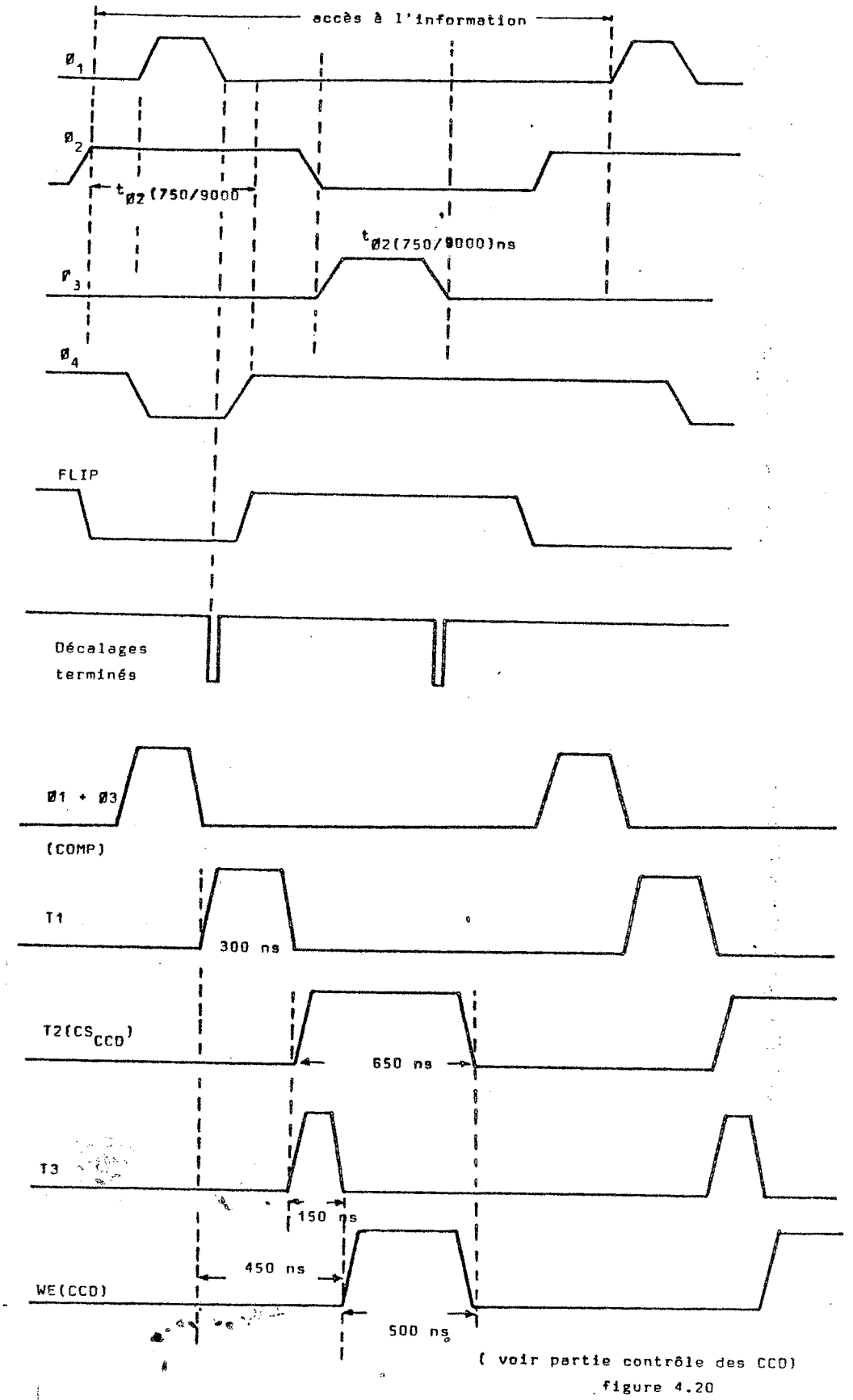


Figure 4.19 Les signaux propres au module CCD.

5.2) Le fonctionnement du contrôleur.

Pour accéder aux CCD en écriture ou en lecture, il faut le faire entre les intervalles Ø1 et Ø4 ou entre Ø3 et Ø2, on prend donc comme point de départ les phases Ø1 et Ø3 pour générer les signaux de sélection(CHIP SELECT, CHIP ENABLE, WRITE ENABLE) propres aux CCD (figure 4.20). Ainsi le signal(Ø1+Ø3) détermine la fréquence à laquelle les données sont décalées.

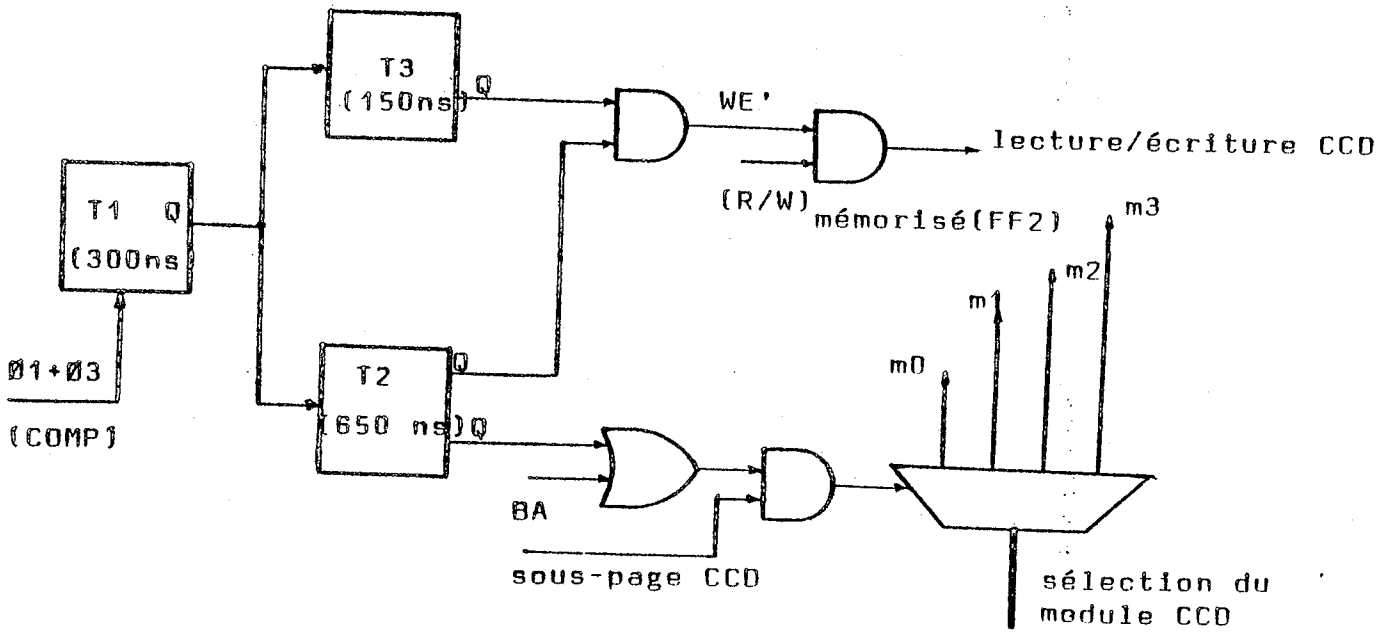


Figure 4.20 La partie contrôle des CCD.

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.

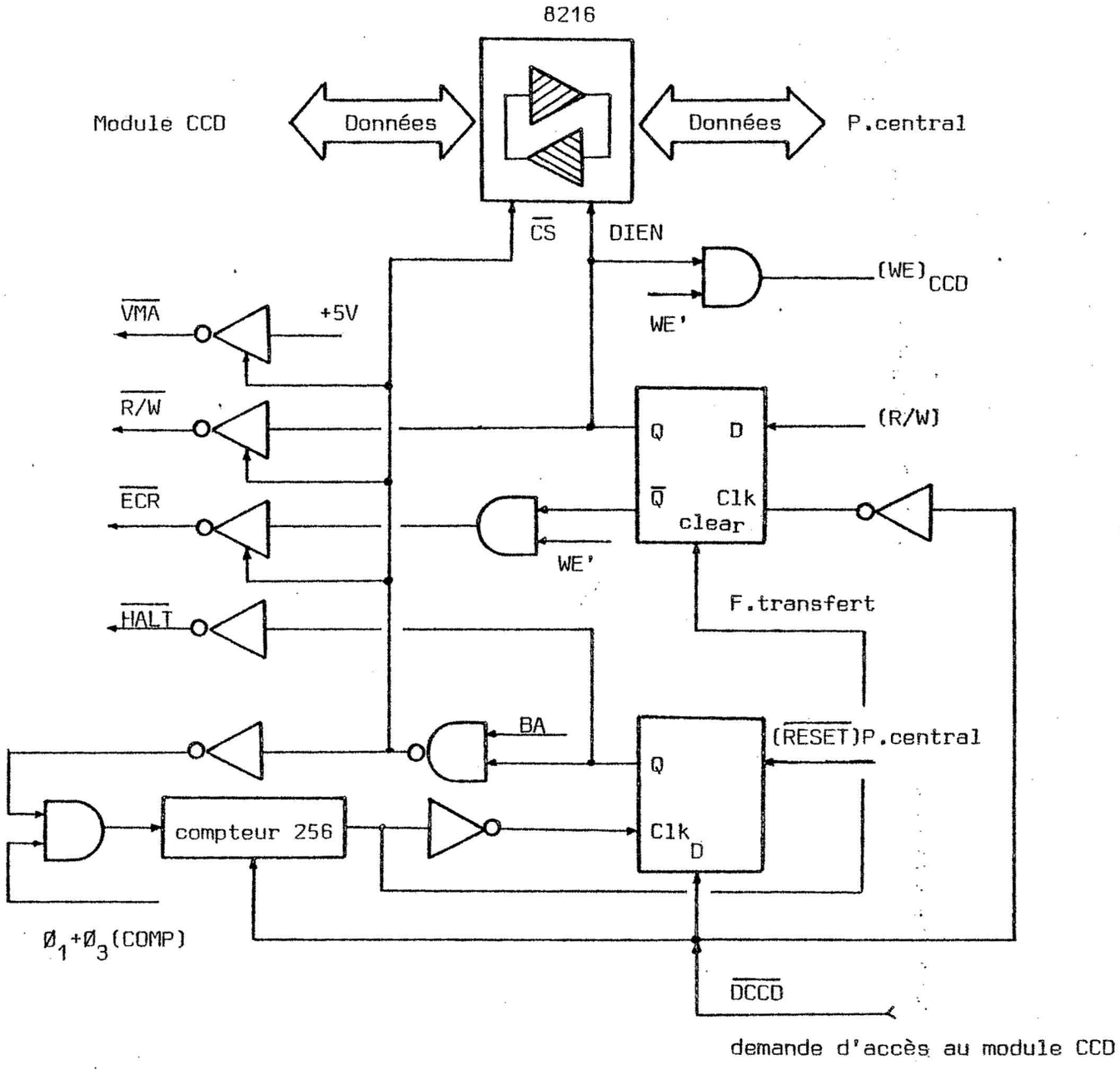


Figure 4.21 La partie commande vers la mémoire et le P.central

- 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(Ø1+Ø3) 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 selectioné (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.

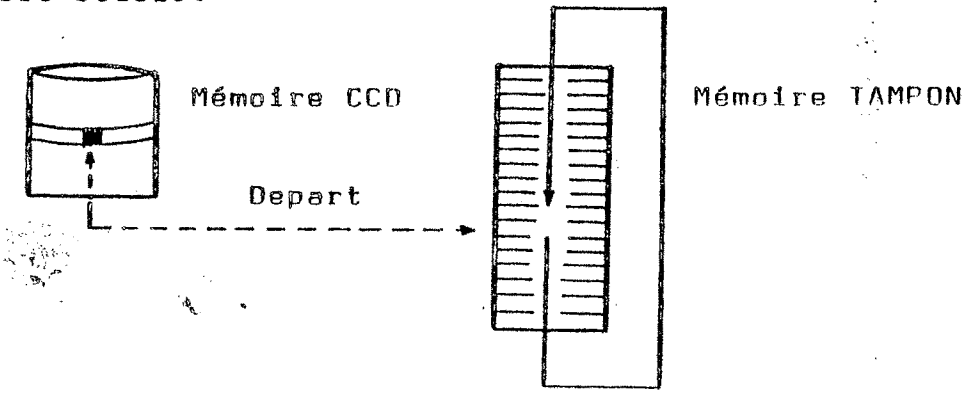


Figure 4.22 Le transfert bouclé.

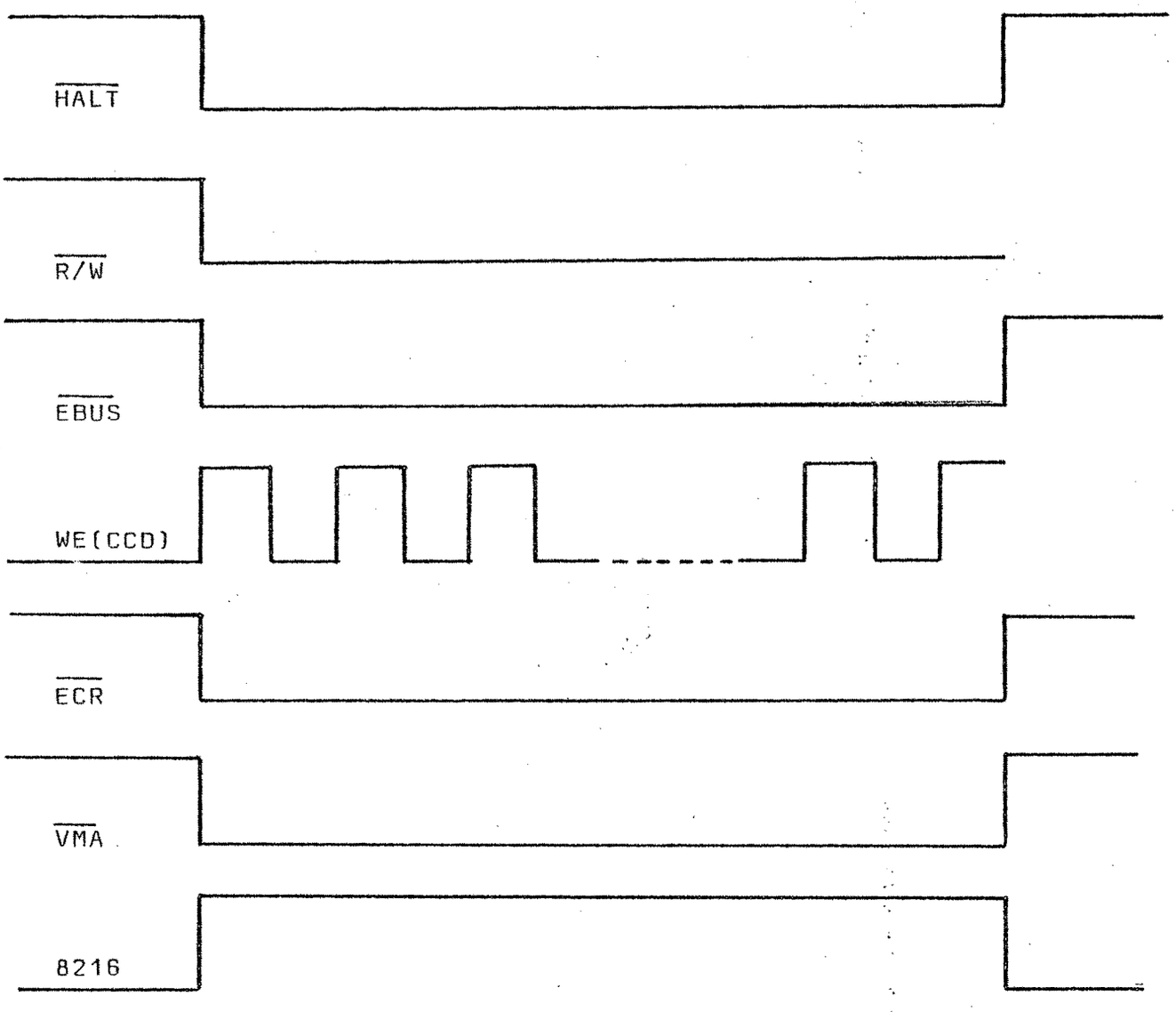


Figure 4.23 Diagramme des signaux en écriture.

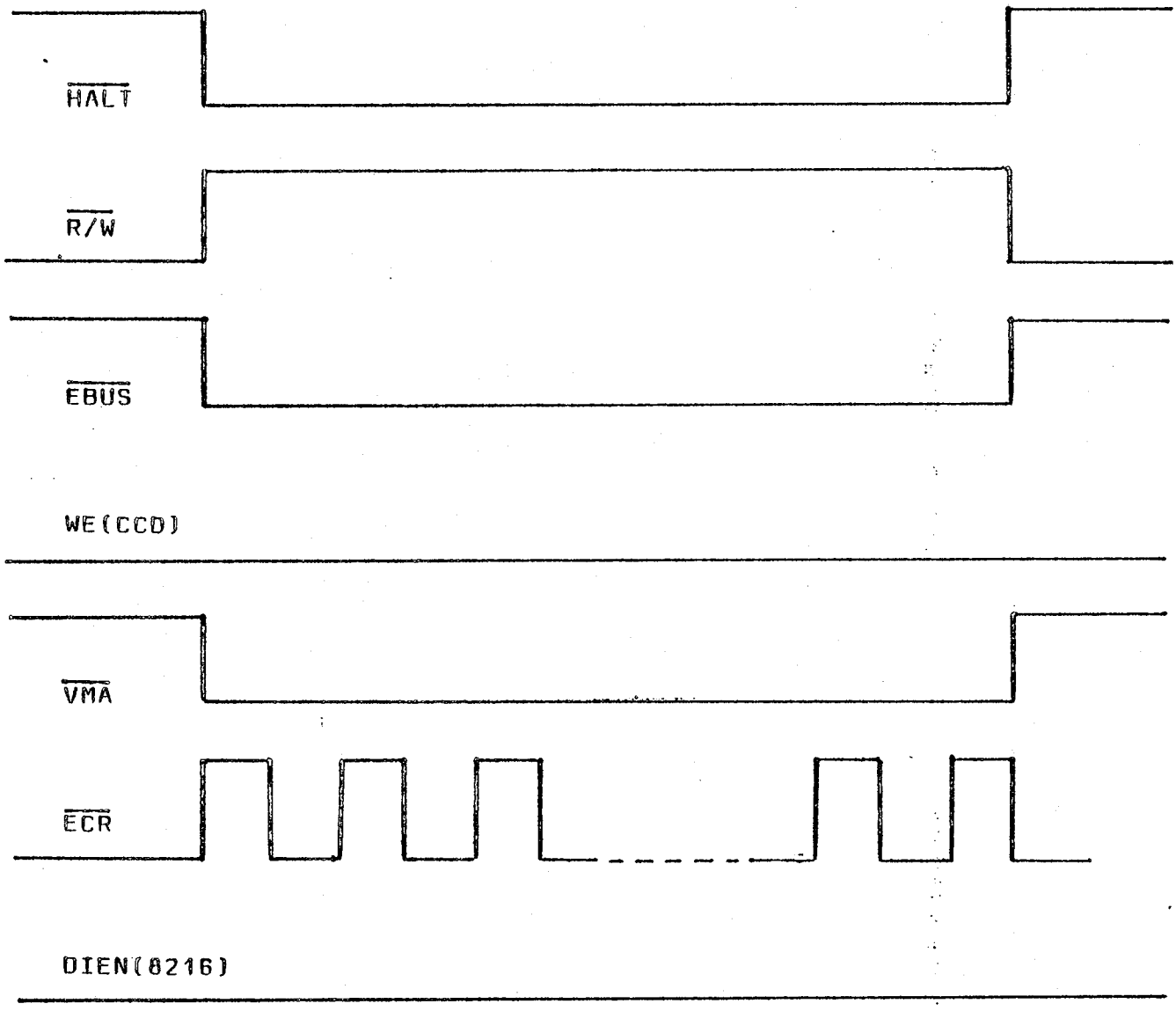


Figure 4.24 Diagramme des signaux en lecture.

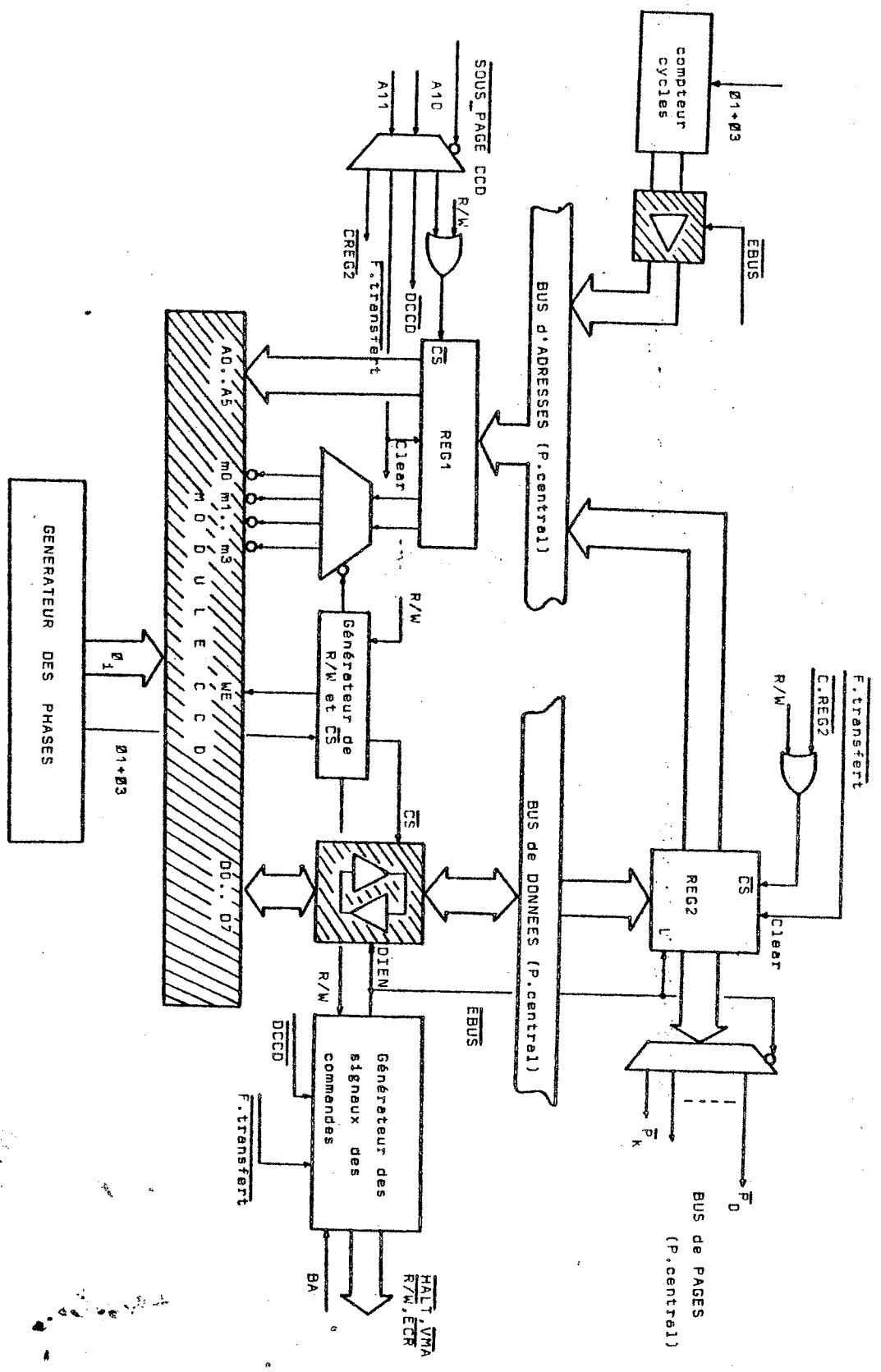


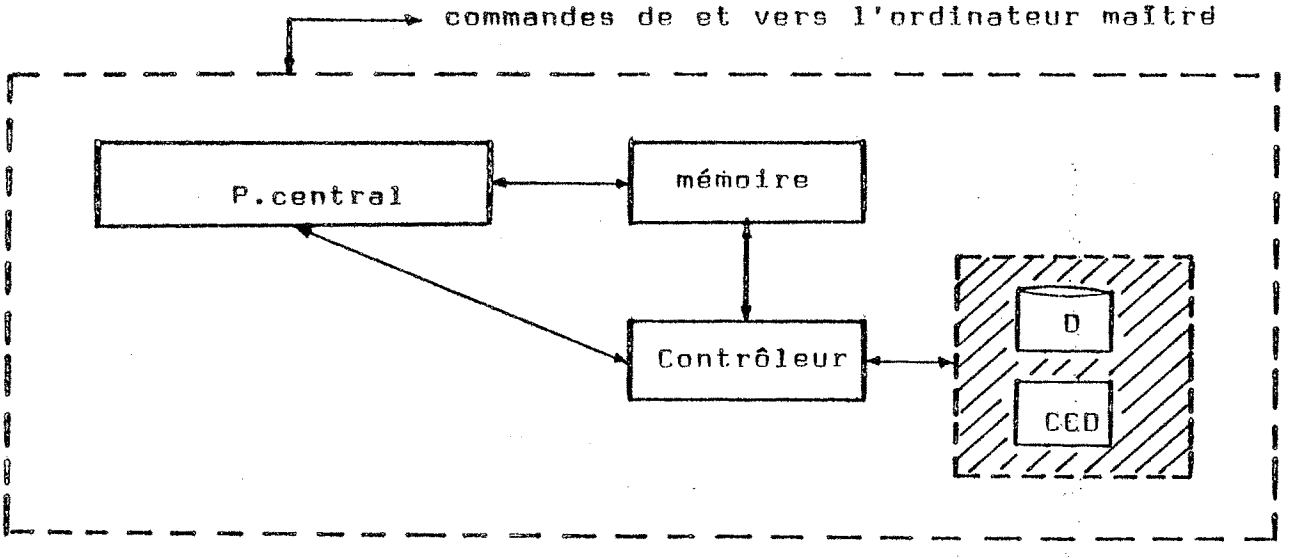
Figure 4.25 Diagramme général du contrôleur CCD.

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 la 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.) EXTENTION 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 FD1771 de WESTERN DIGITAL, qui transfère les bits d'informations entre l'unité disque souple choisie et le P.central.

Une extention prévue de ce contrôleur est la réalisation d'une carte processeur fonctionnel associée à l'unité disque souple. Un microprocesseur MC68B00 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:

<contrôleur>::= <matériel d.souple>[←]<L.souple>

où:

<matériel d.souple>: représente l'ensemble du matériel pour le contrôle électrique de l'unité de disque souple.

<L.souple>: 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.

<matériel d.souple>::= <circuit contrôleur>
<interface P.central>
<interface unité.d.souple>;

<interface unité.d.souple>::= <circuit.trans.récep>
<liaison>
<unité d.souple>;

où:

<circuit contrôleur>: 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, Ø2, 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.

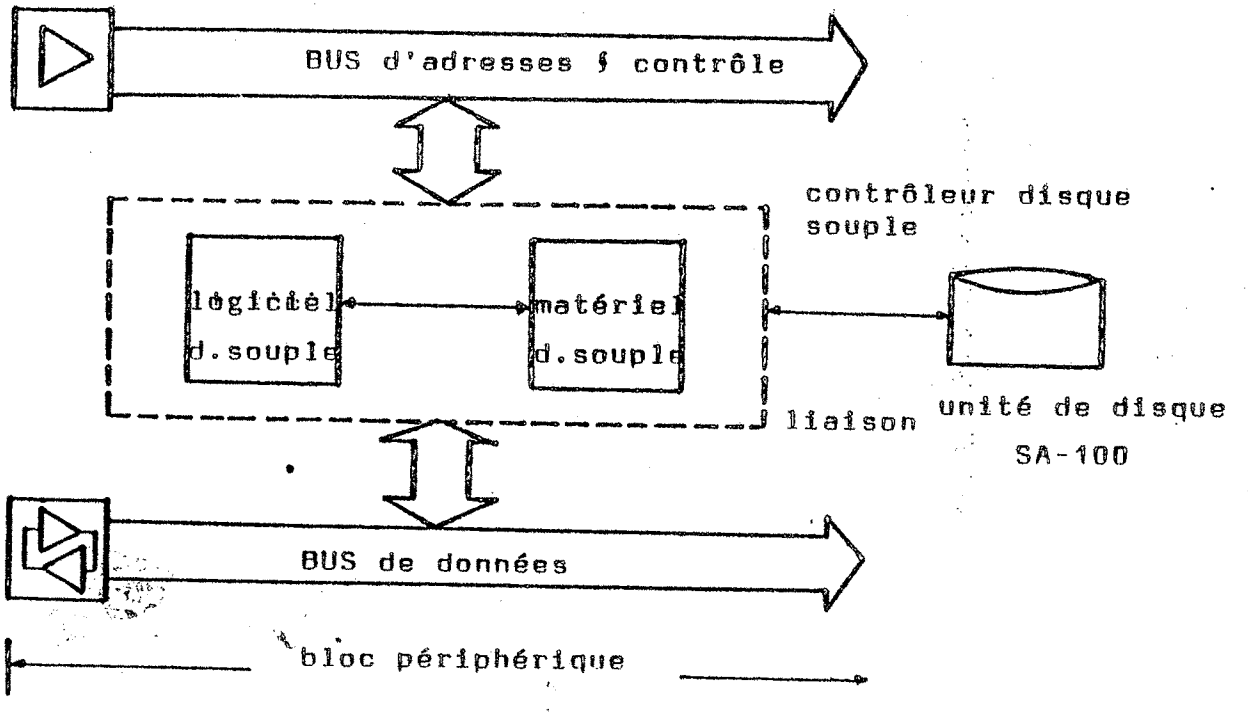


figure 5.1 l'environnement du contrôleur disque souple.

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.

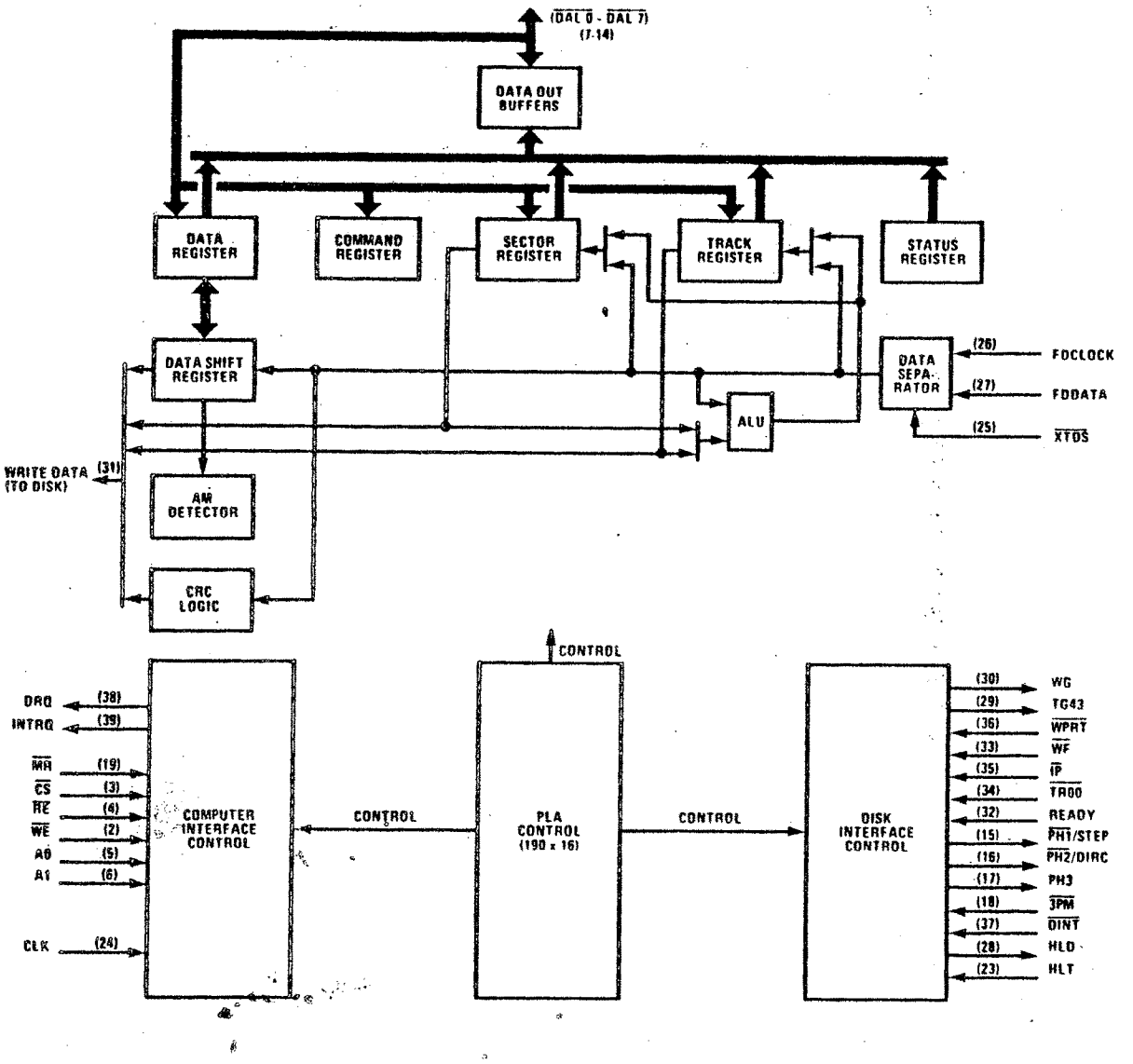


Figure 5.2 Synoptique du circuit contrôleur de disque souple.

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.

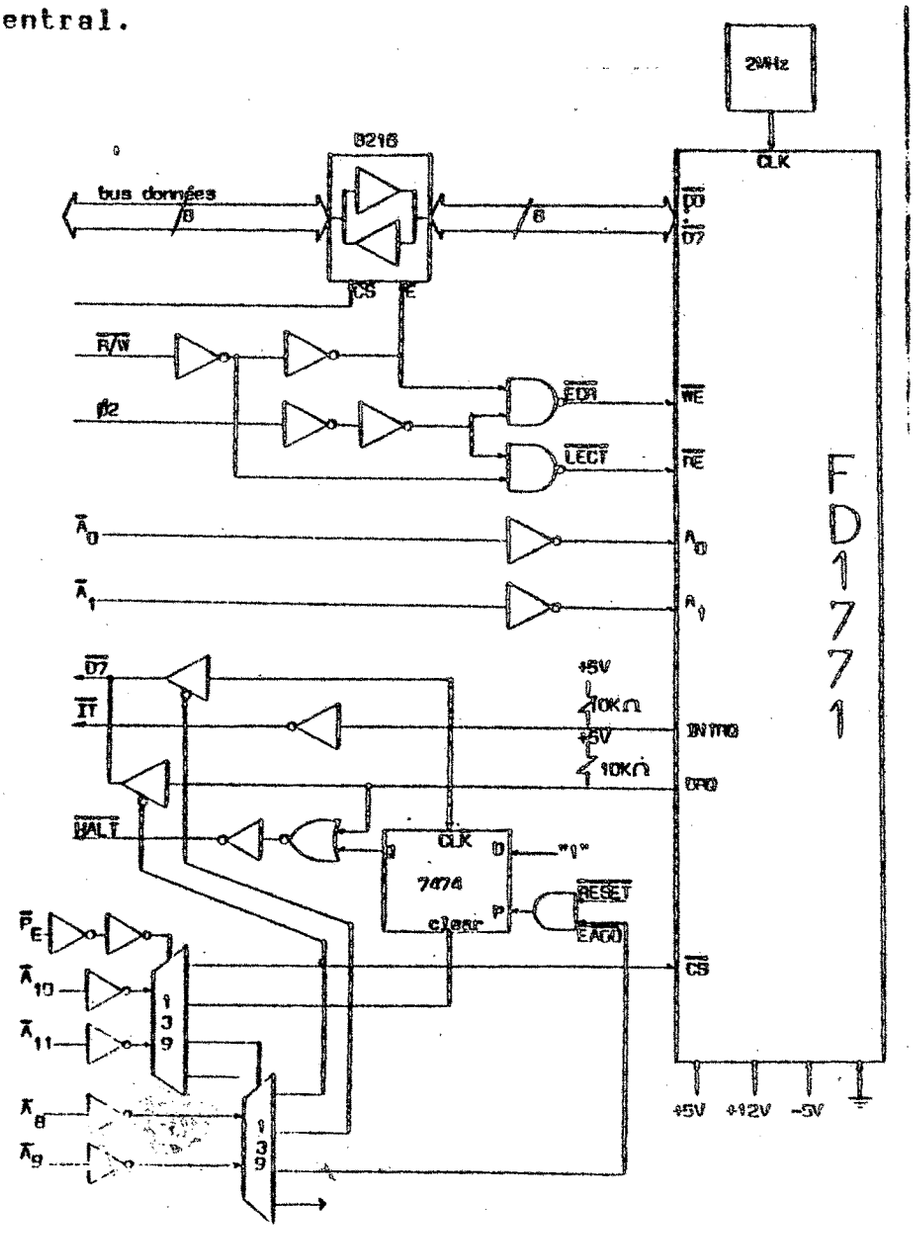


Figure 5.3 L'interface P.central.

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 boitier SFF96871A1 qui aussi prévu pour générer les signaux propres au microprocesseur destinés à l'extention: on utilise une fréquence de 2Mhz.

2.1.2.2) Bus de données.

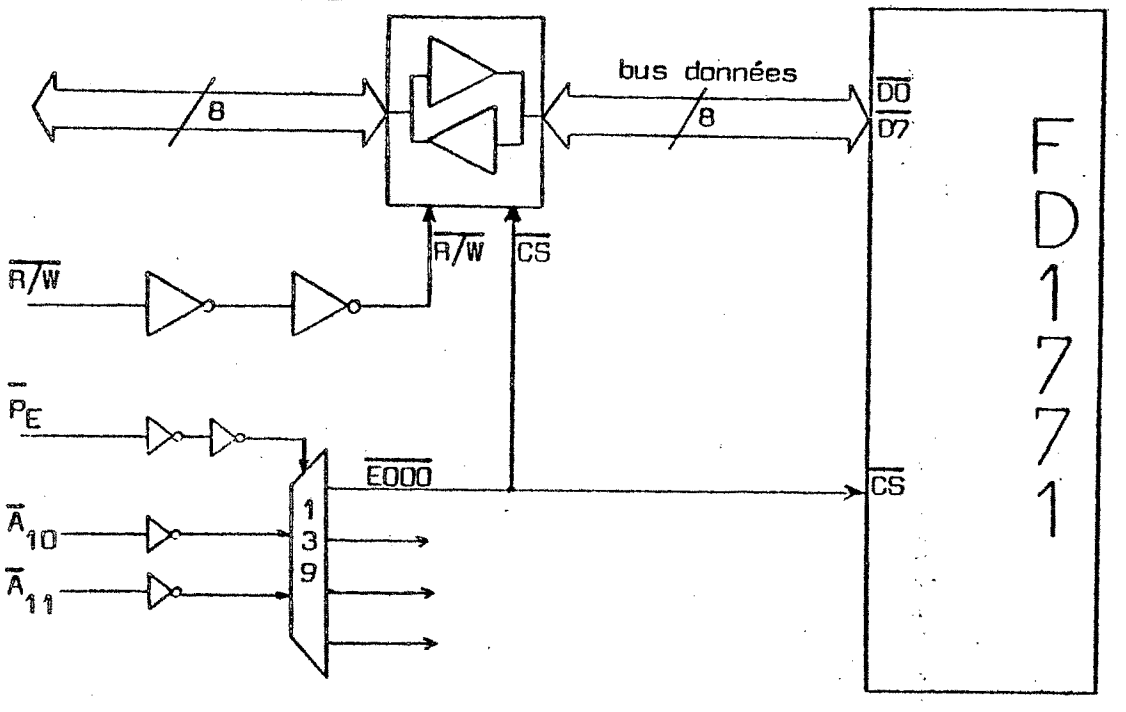


Figure 5.4 Le contrôle du BUS de données.

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 (A0 et A1) associés à l'ordre de lecture-écriture échantilloné sur la phase Ø2. Ainsi la sélection d'un registre se fait sur Ø2.

En effet pendant la phase Ø1 le P.central positionne les lignes de commande et d'adresses et, pendant la phase Ø2 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 Ø2.

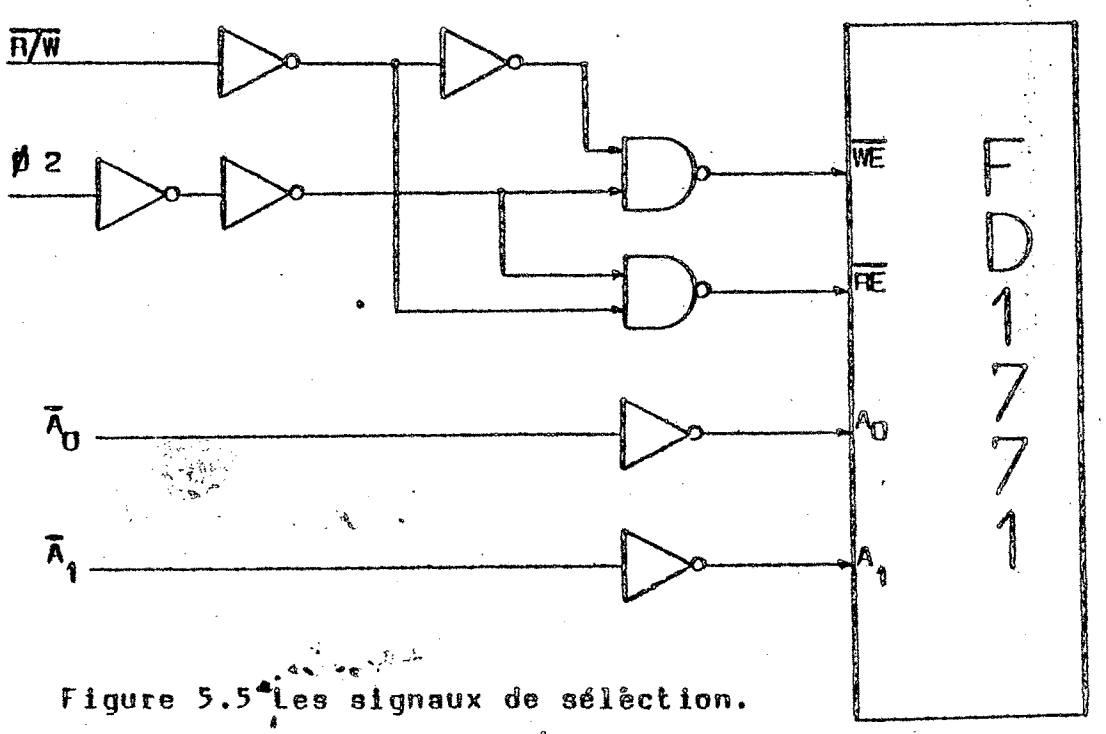


Figure 5.5 les signaux de sélection.

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 déroutée.

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

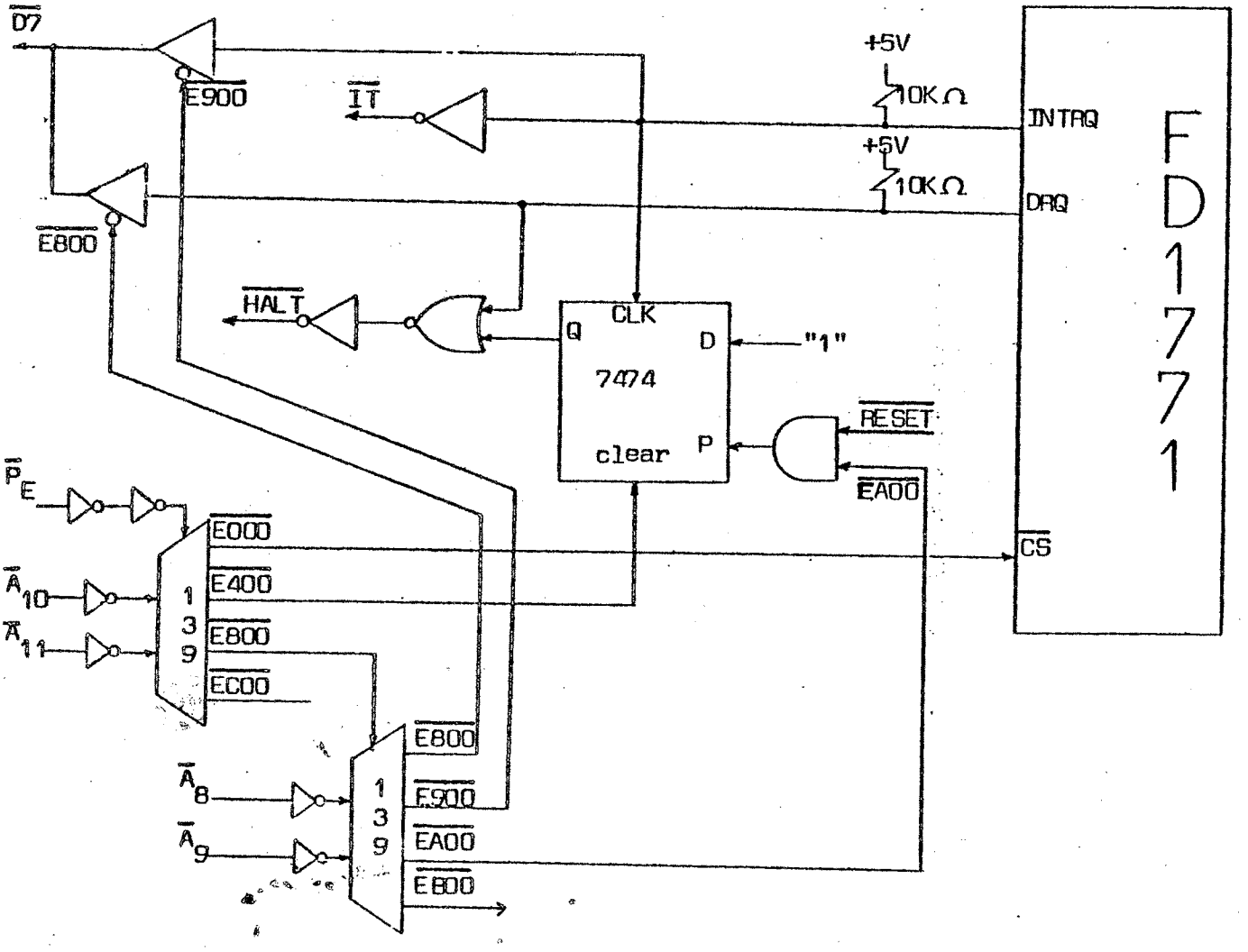


Figure 5.6 Le contrôle des interruptions.

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) (RT7-78) par boucle logicielle:

```
UP1 LDA A,=XE8xx
    BNE UP1
```

- 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 FD1771 avec le P.central ou viceversa.

La ligne $\overline{HALT}$ 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.

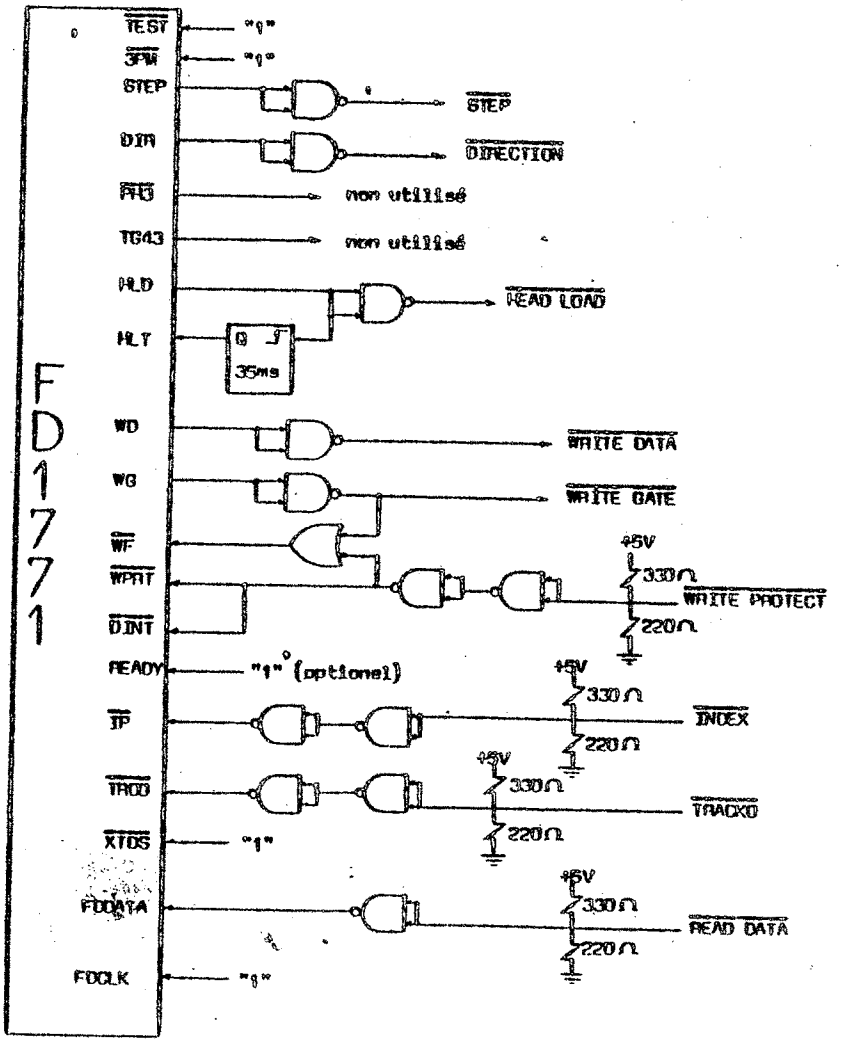


Figure 5.7 Interface vers l'unité de disque souple.

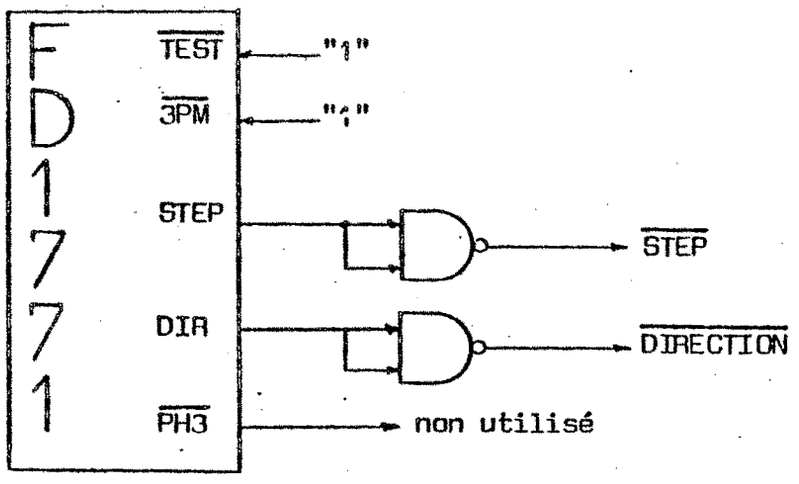


Figure 5.8 Commandes du bras.

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.

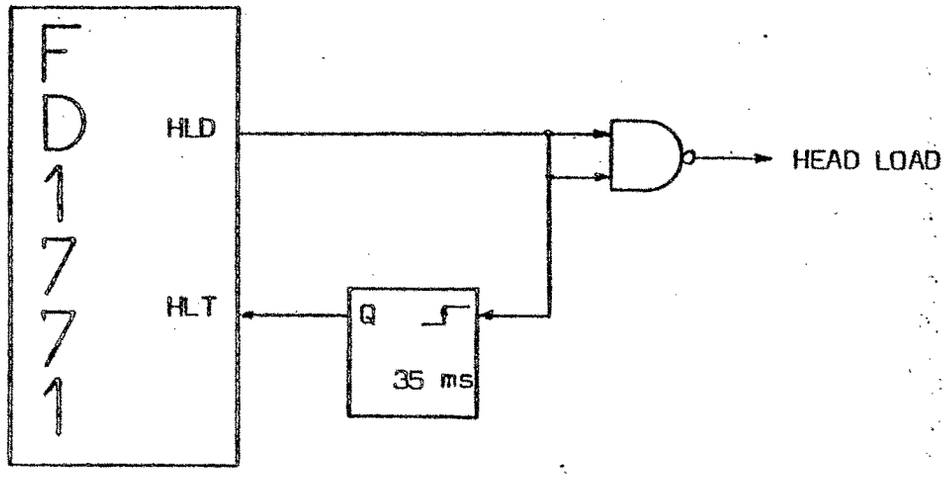


Figure 5.9 Positionnement de la tête.

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 HLT (RT6-78).

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

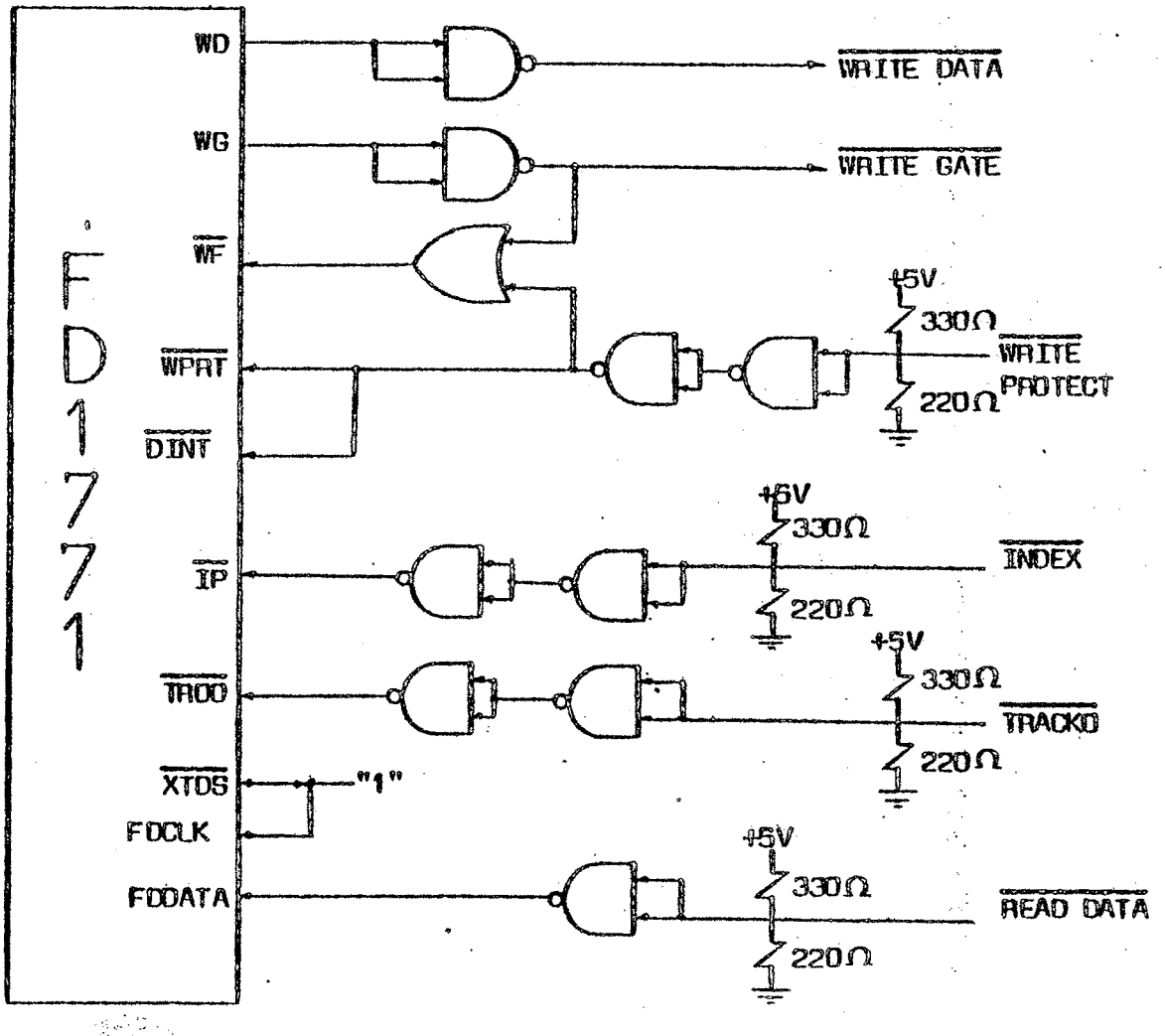


Figure 5.10 Réception/transfert des données entre le contrôleur et l'unité disque souple.

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.

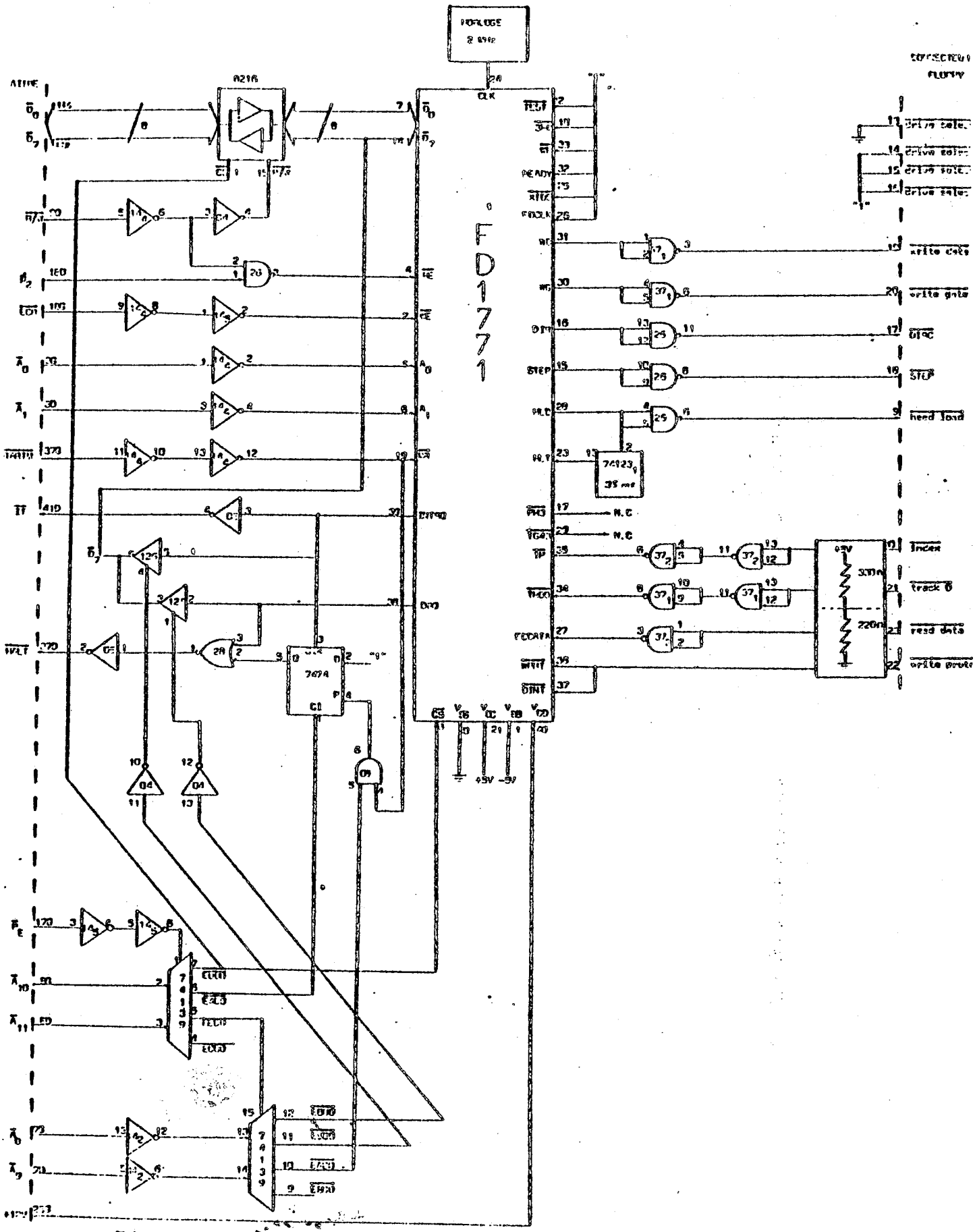


Figure 5.11 Cablage du contrôleur de disque souple.

2.2) Le logiciel disque souple.

Le logiciel disque souple est représenté de la manière suivante:

<logiciel d.souple>::= <L.base><L.d.souple>
où:
<L.base>: logiciel de base du P.central(moniteur)
<L.d.souple>: 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.

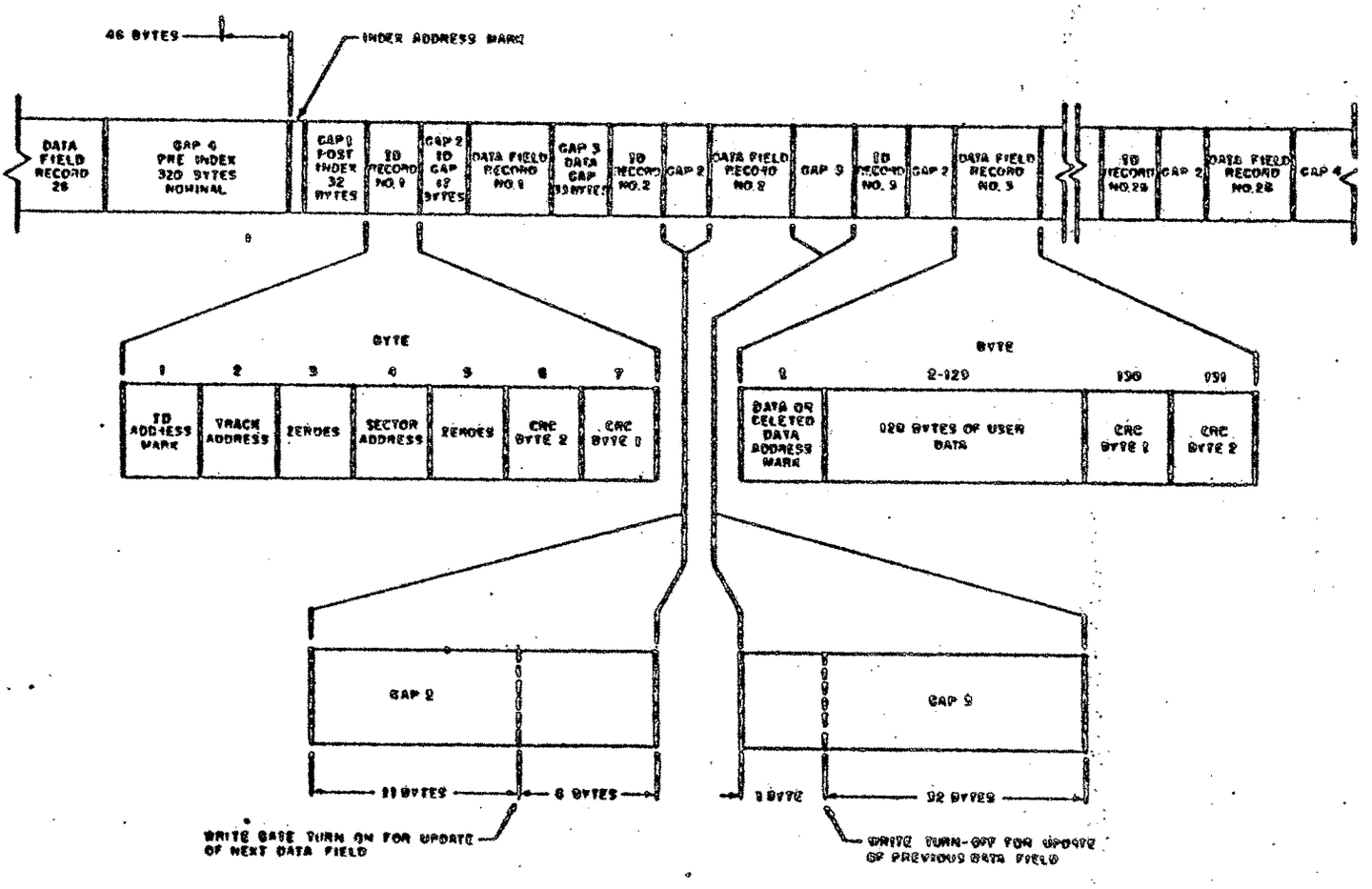


Figure 5.12 Format d'une piste.

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 octets

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

166.6/5208=0.032 ms= 32 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 maximun 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 IT.
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.

```
UP1 STA B,ITH .. activation circuit d'IT
UP2 LDA A,DRQ
    BNE UP2
    LDA A,DR
    STA A,0(X)
    INCX
    BRA UP1
```

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.

```
WUP1 STA B,ITH
WUP2 LDA A,DRQ
    BNE WUP2
    LDA a,0(X)
    STA A,DR
    INCX
    BRA WUP1
```

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(MC68B00) 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>.

- Formattage 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 formatter 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.) EXTENTION DU CONTROLEUR: LE PROCESSEUR FLOPPY.

L'extention que nous pouvons apporter à ce problème est de cabler un microprocesseur MC68B00 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 facilitées 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.

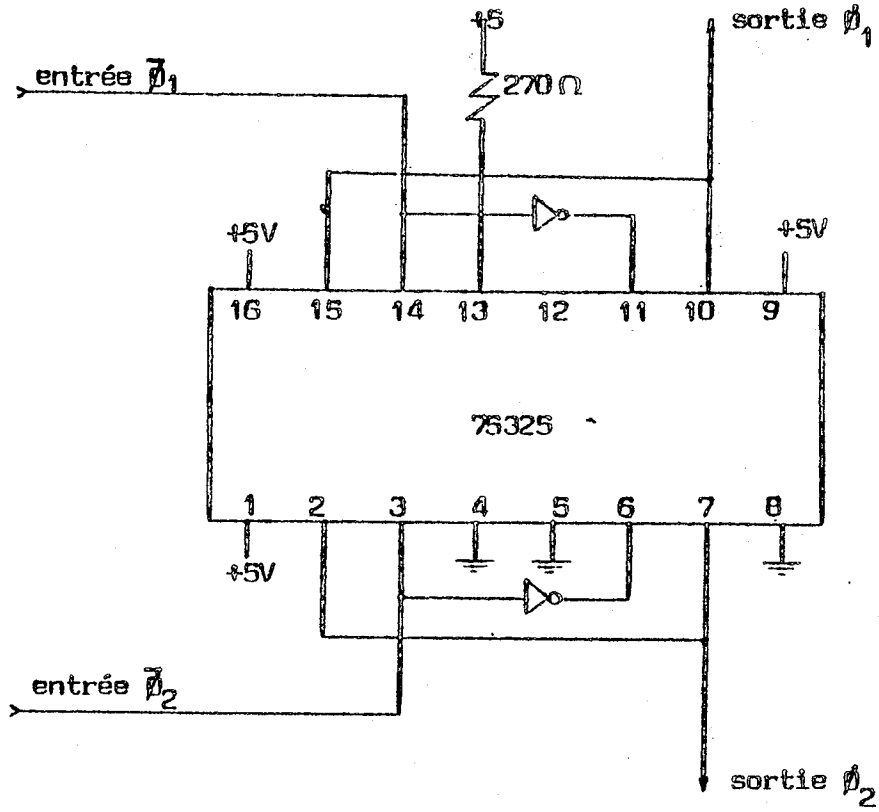


Figure 5.13 Schéma du driver.

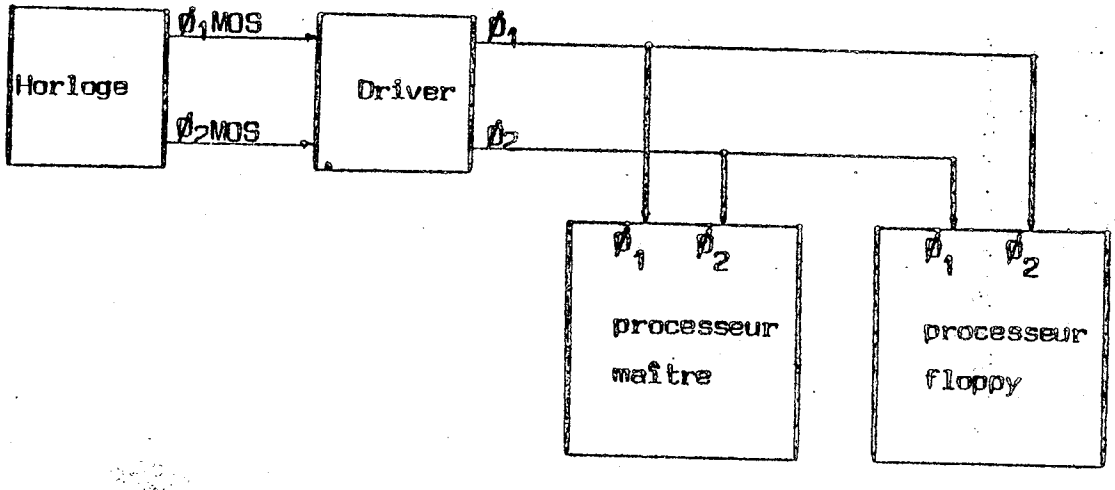


Figure 5.14 Le mécanisme à phases complémentées.

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

CHAPITRE VI

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

1.) INTRODUCTION

Le principal problème dans le devéloppement 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'extention du système ANALYD.

2.) GENERALITES.

Le système ANALYD est un processeur spécialisé, chargé de la surveillance d'un processus qui s'exécute sur une autre machine(figure 6.1).

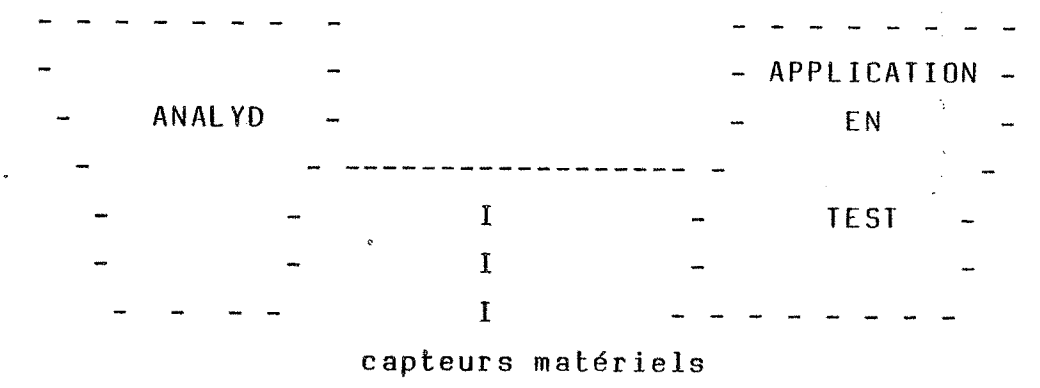


Figure 6.1 L'environnement d'ANALYD.

Il est relié à l'application grâce à un ensemble de capteurs matériels qui prennent les informations nécessaires à enregistrer et à analyser sur le système objet. Il doit donc avoir la caractéristique d'être invisible depuis ce système, il ne doit pas perturber son fonctionnement (électrique et logique).

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.

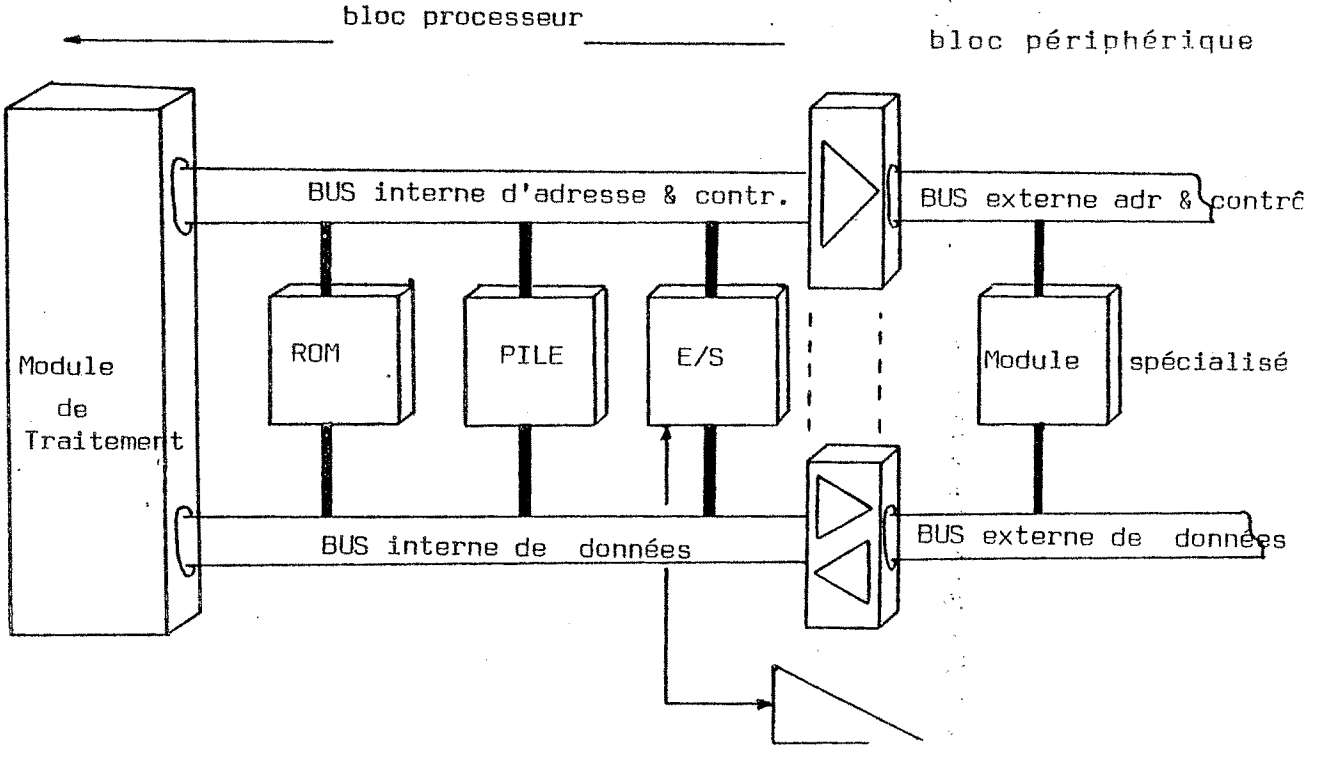


Figure 6.2 L'architecture du sytème ANALYD.

4.1) Le bloc pocesseur.

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, pròprement 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 selectionner 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.

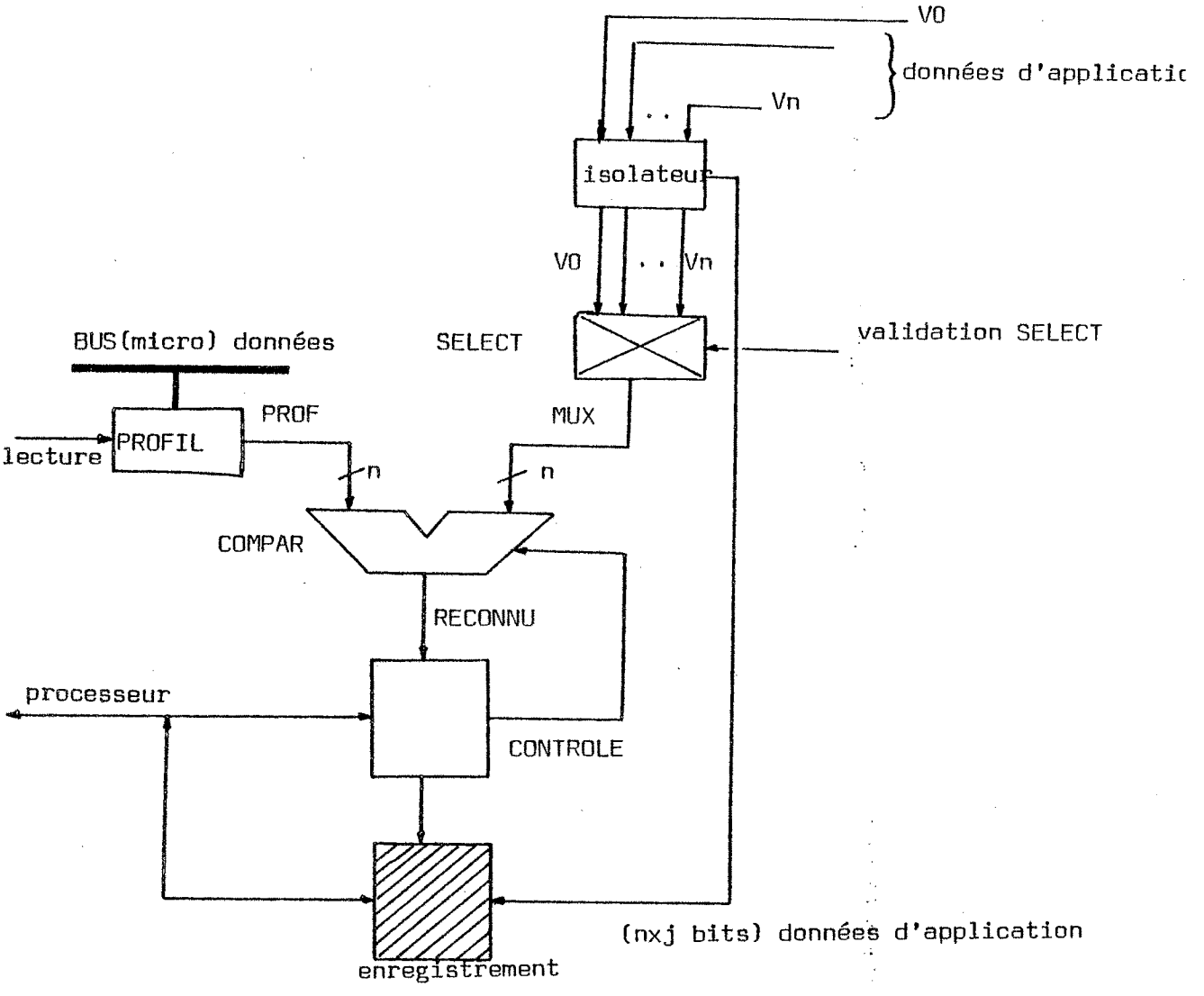


Figure 6.3 Schéma de ANALYD comme système mono-profil.

- 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) ≠ (SELECT) FAIRE
M(0,I):= enregistrement des données;
DEBUT
arrêt compteur de cycles;
TANT QUE i+1 ≠ 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:

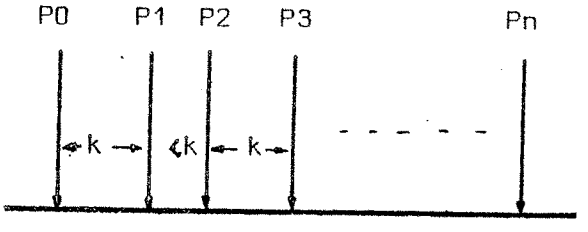
d'une manière ordonnée,

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.

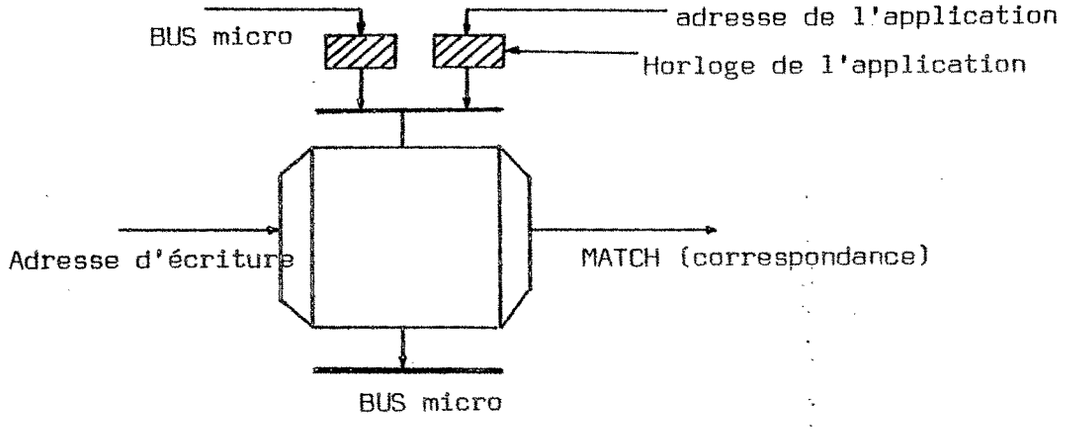


Figure 6.5 Configuration de la mémoire de PROFILS.

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

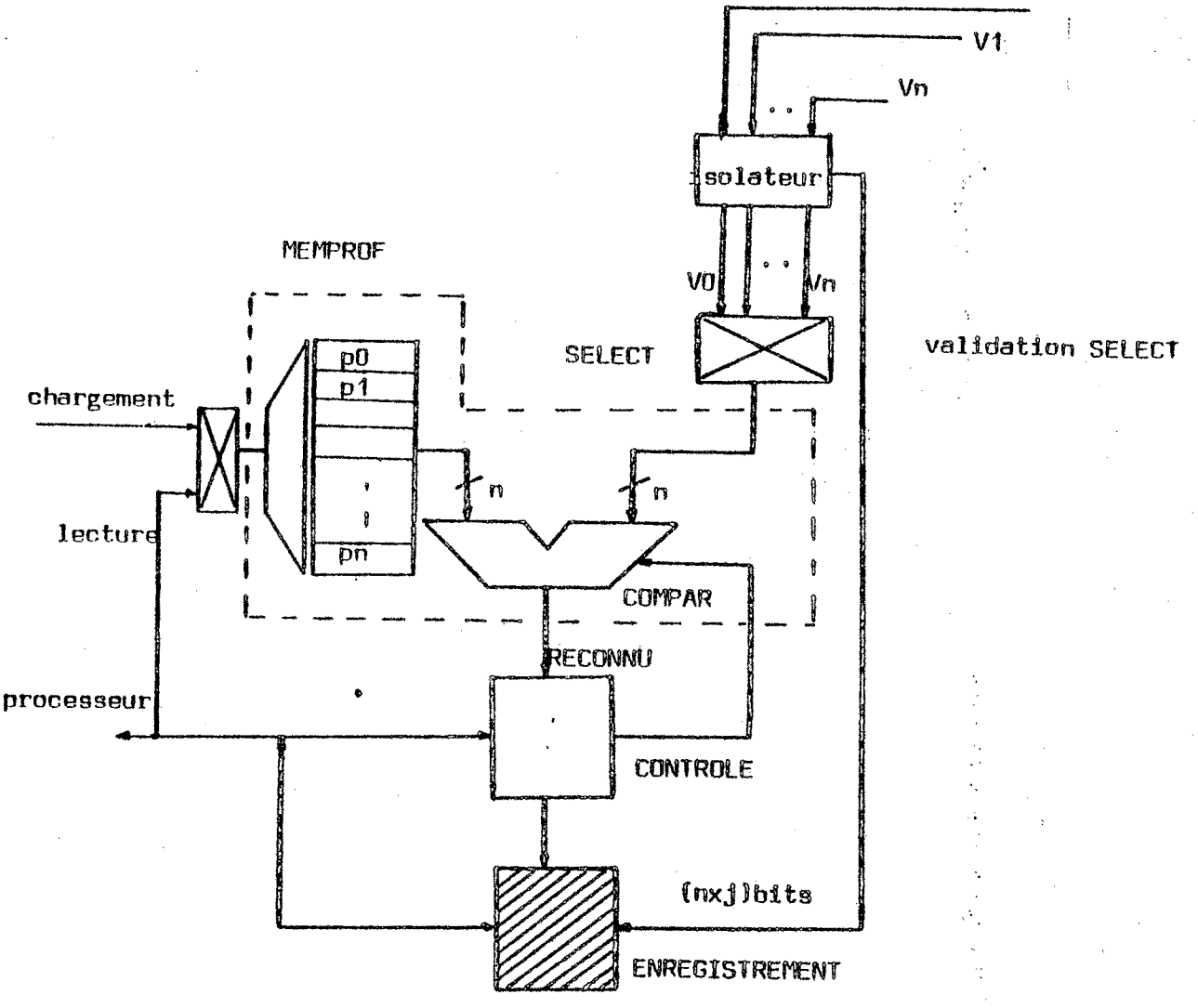


Figure 6.6 Le système multi-profil aléatoire.

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.

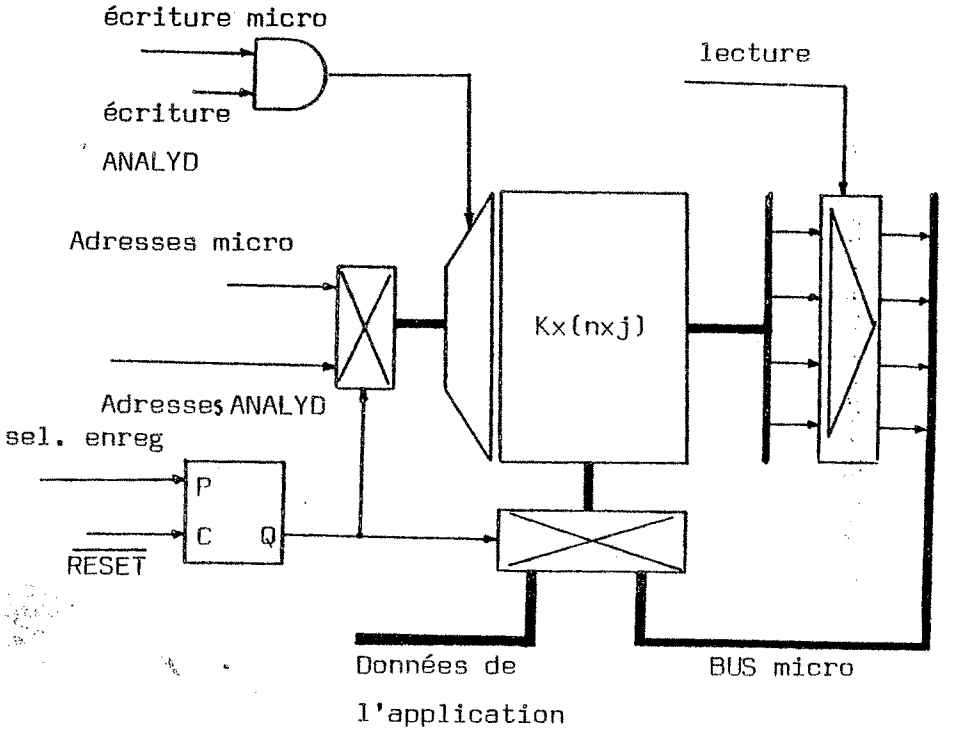


Figure 6.7 La partie enregistrement.

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 (R13-78).

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.

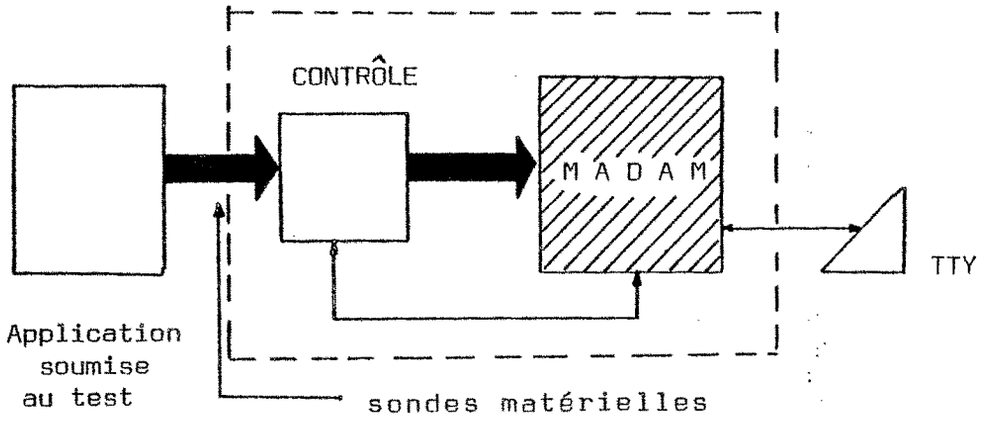


Figure 6.8 Le système ANAMAD.

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/$\overline{W}$,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 reconnaitre 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)

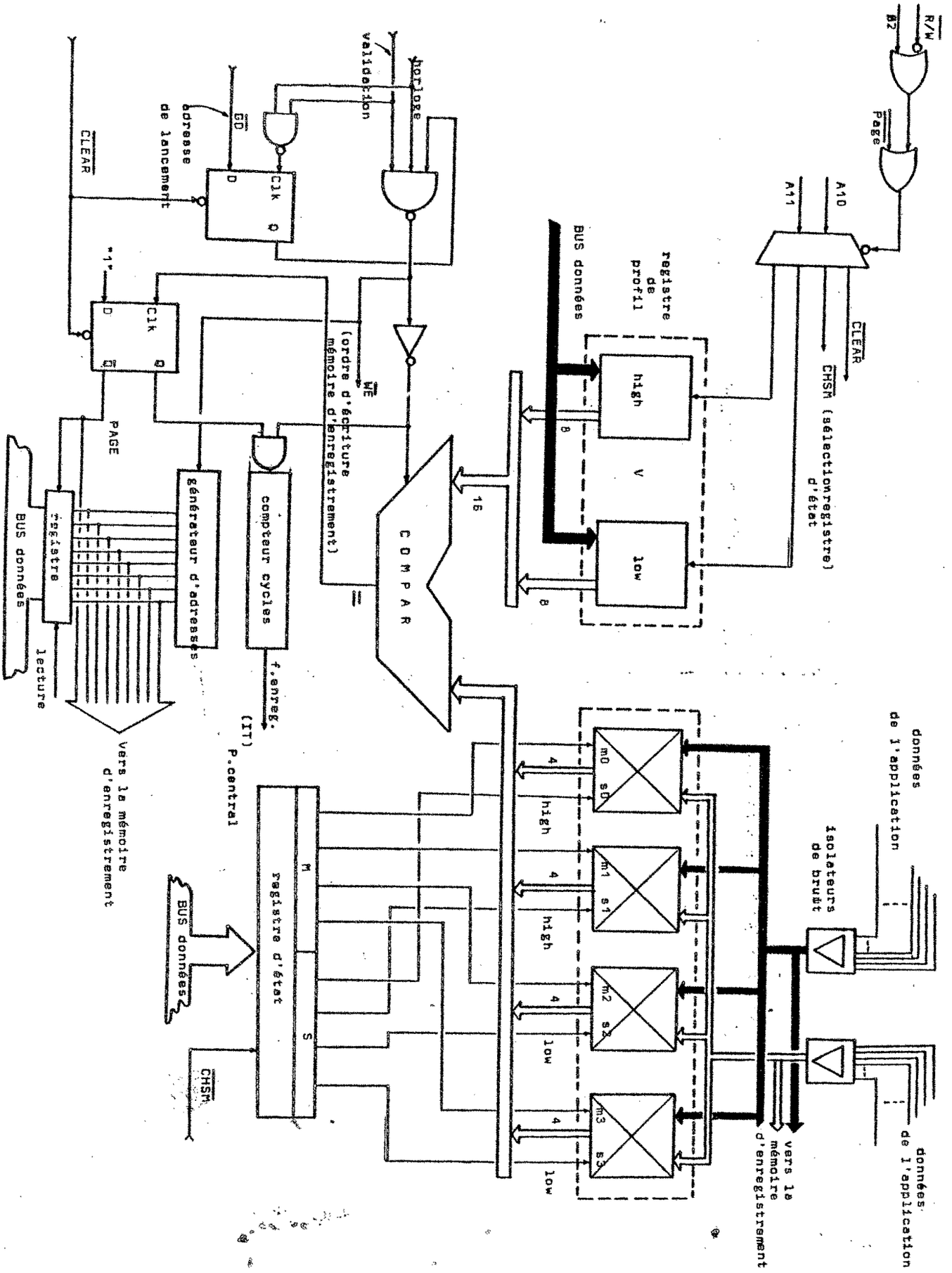
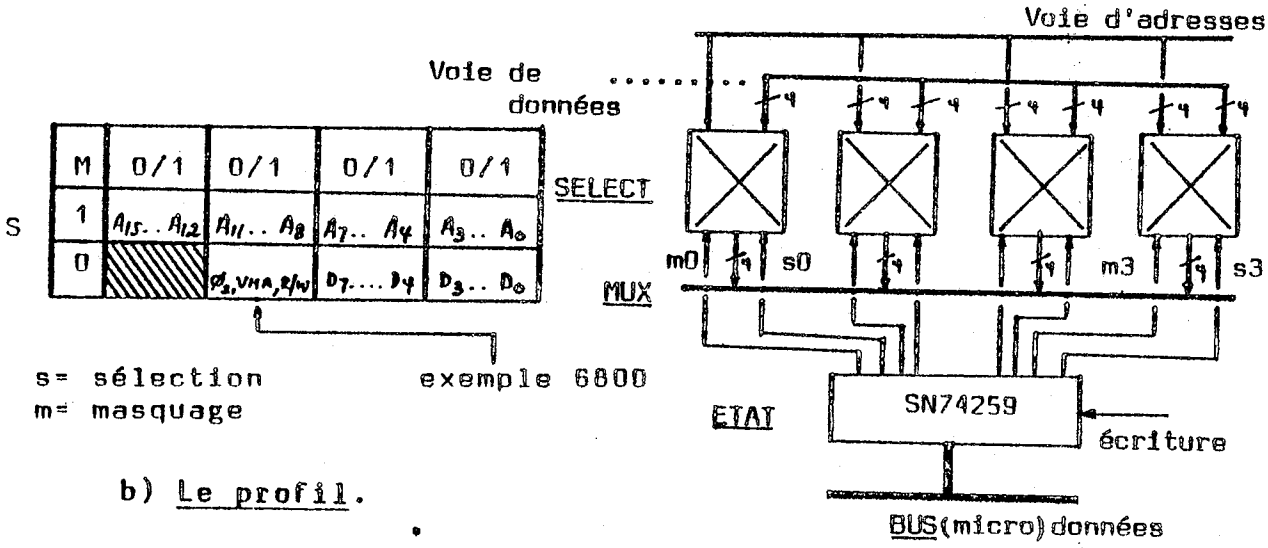


Figure 6.9 Le mécanisme de contrôle.

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 cablé 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:

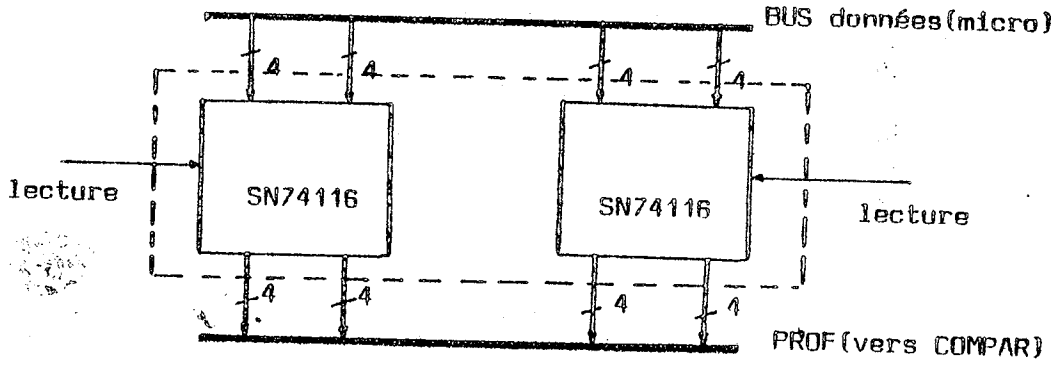


Figure 6.10 Le profil.

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).

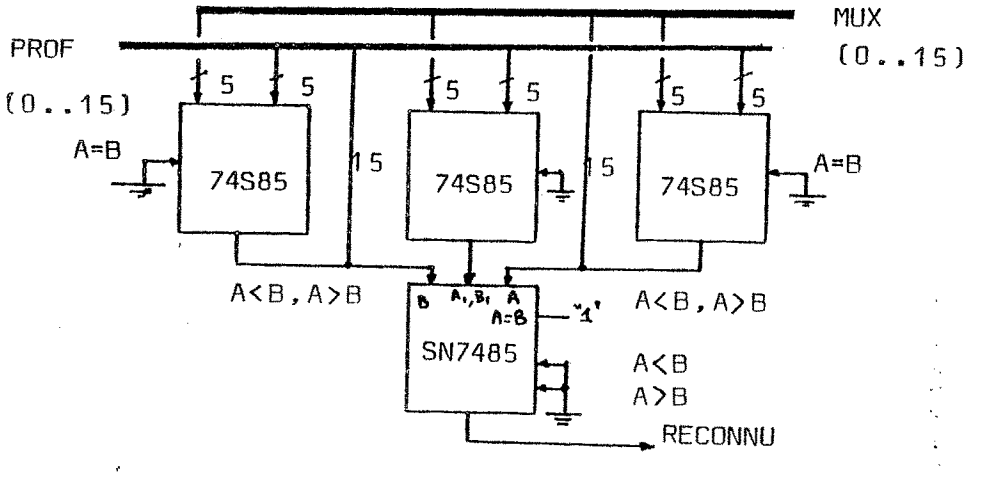


Figure 6.11 Configuration du comparateur.

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 dûe 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,

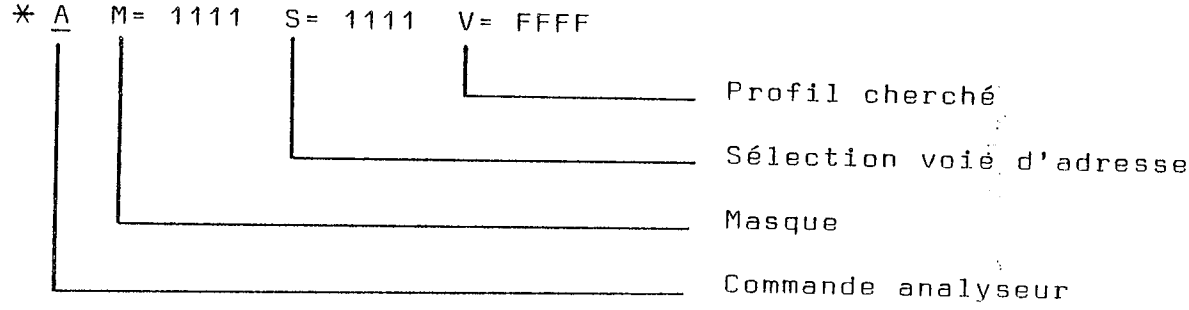
-de choisir la voie où se trouve le profil,

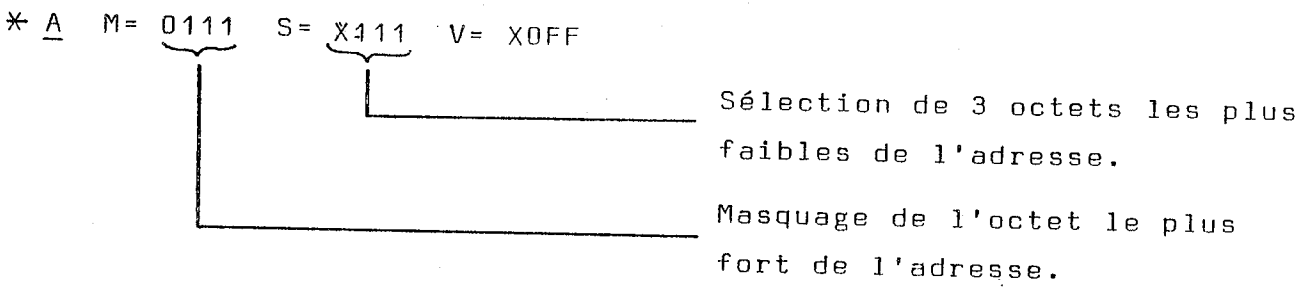
-de valider et de sélectionner SELECT,

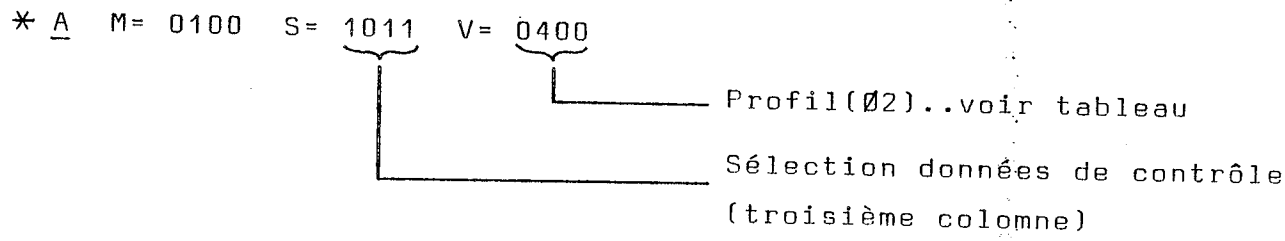
-de visualiser sur une imprimante les événements enregistrés avant et après l'arrivée du profil. Il contient également les commandes classiques de maintenance d'un moniteur système (ex: analyse de la mémoire, test,..).

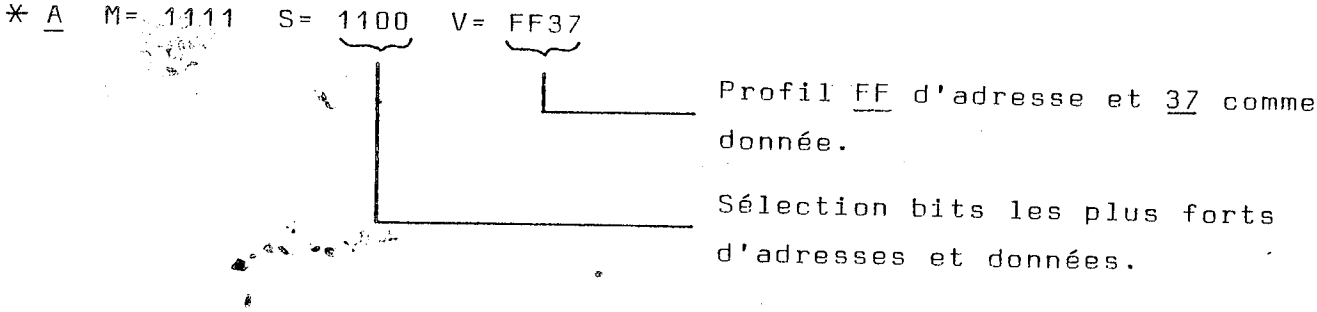
Exemples.

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









6.) L'EXTENTION 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 (Tenr) dépend de la fréquence d'arrivée des informations (événements).

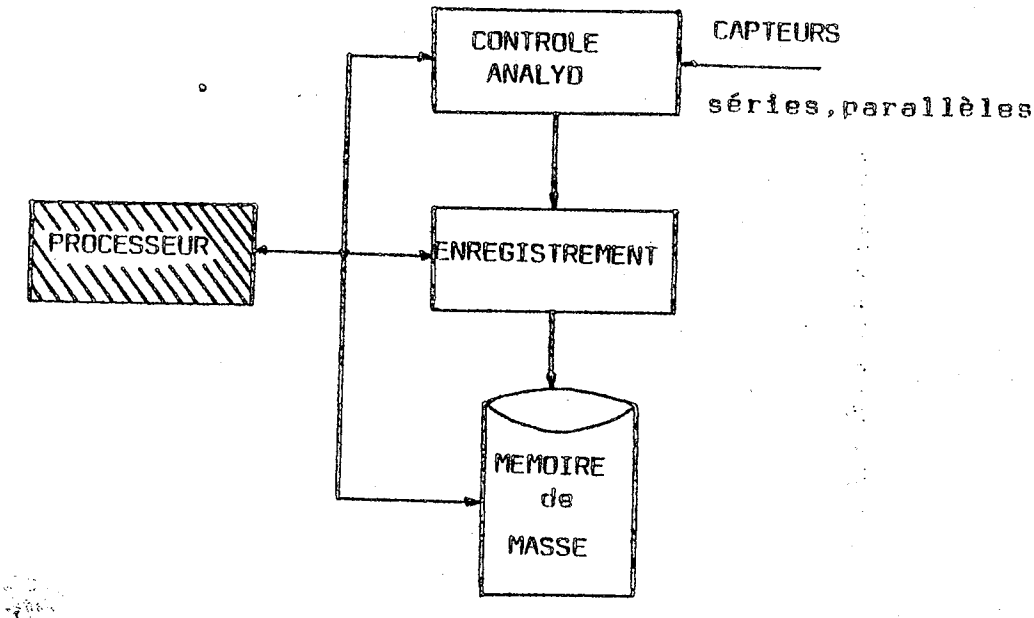


Figure 6.12 L'extention d'ANALYD.

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 extention, 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/Tenr), ainsi que de la fréquence de vidage de ces tampons vers le disque souple(1/Tvid).

Nous considérons le nombre de tampons (n) borné.

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

1. fenr>fvid;
2. fenr<=fvid.

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

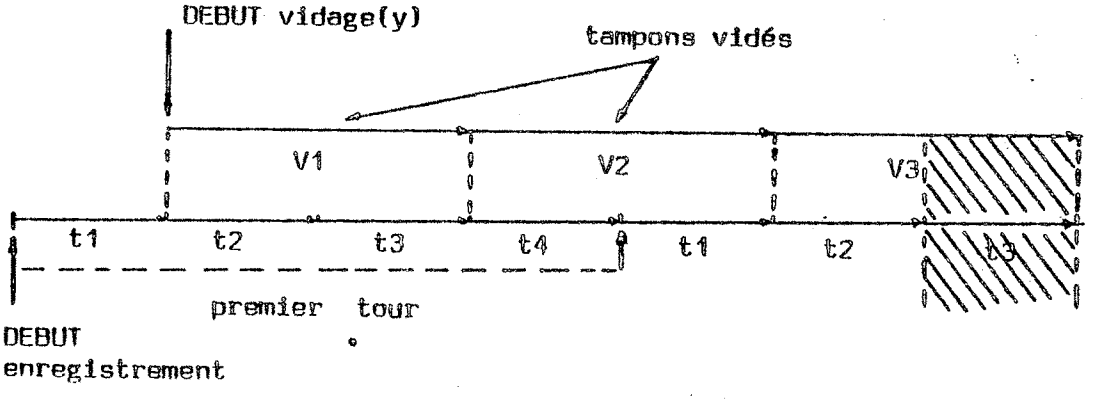


Figure 6.13 Différents temps d'enregistrement et de vidage.

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(vidage)/t(enregistrement);

Durant le vidage de V1, s'enregistrent les tampons compris entre t1 et t(f+1).Ainsi

Vidage	Enregistrement
V1	t1...⩽t(f+1)
V2	⩽t(f+1)...⩽t(2f+1)
	* *
	* *
	* *
Vn	⩽t((n-1)f+1) ...⩽t(nf+1)

puisque nous avons supposé que le nombre total n de tampons est borné. Lorsque Vn 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 Vn à vider soit d'un indice supérieur ou égal à (k-1)X+x+1 donc n⩾ (k-1)X+x+1;

d'où X⩾ n(f-1)+2

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

d'où

X>= n(f-1)+2y;

k<= (nf+y)/X;

x>= 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:

tenr = 25*capacité du tampon

tvid = 32*capacité du tampon

f = tvid/tenr; 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>=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.

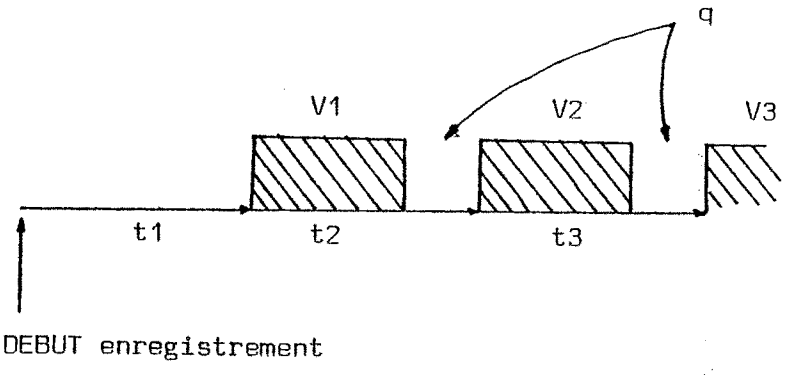
L'algorithme général du système est présenté dans la procédure SANAMAD.

```
Procédure SANAMAD;
Début
i:=1;CO i compteur d'enregistrements;
Tant que i≠(nf+2y) faire
Debut
      enregistrement(i);
      PARALLEL(enregistrement(i+1),vidage(i));

Fin;
Fin.
```

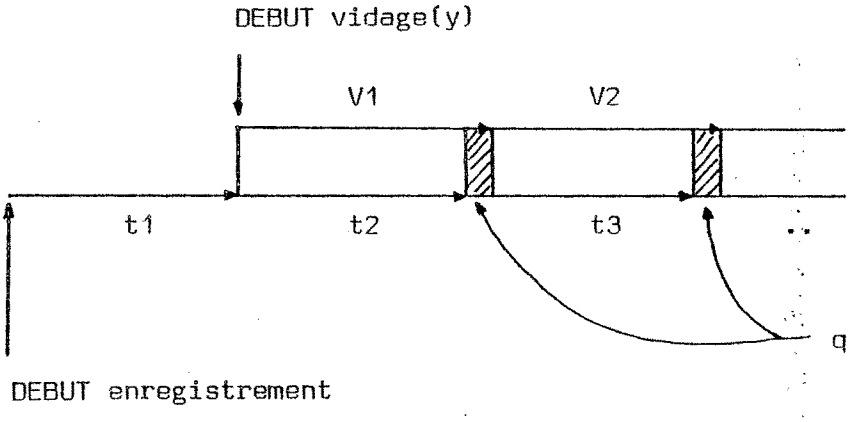
/2. Dans le cas où fenr<=fvid, le problème se pose différemment car on a du temps pour sauvegarder les informations et il nous reste le temps(q) pour exécuter des actions diverses(ex: chercher la prochaine adresse dans le disque souple).

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(tenr,tvid,q)



fenr<fvid

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

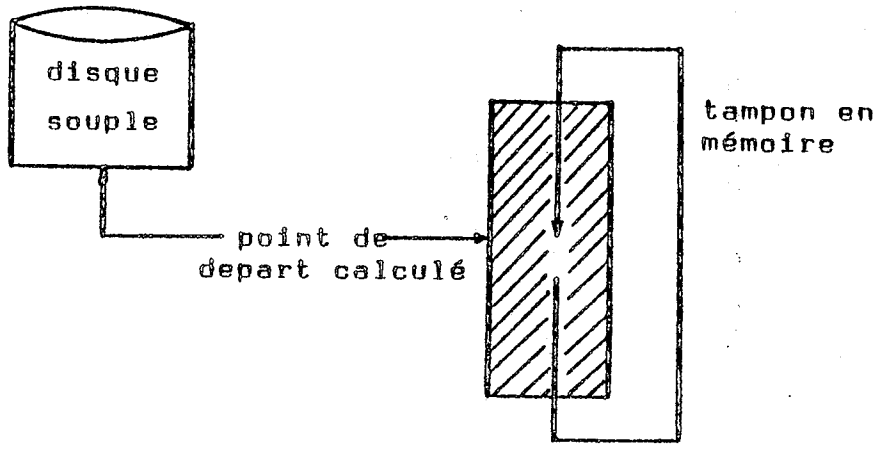


fenr=fvid;

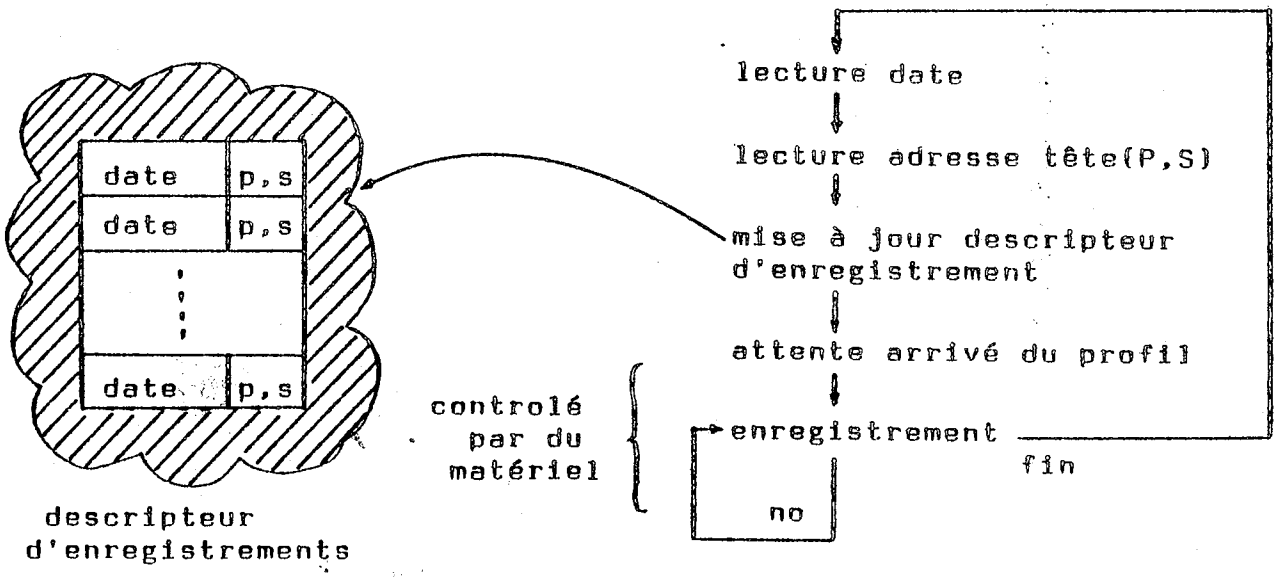
Il faut remarquer que ce temps (q) doit toujours être considéré pour fenr>=fvid et fenr<fvid.

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, procedure, language) pour mieux definir 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'anné de l'ENSIMAG (Troisième anné, 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 permi d'acquerir 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.EWARD
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 Algorithmes 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.MALISTER,P.J. SAYER
Hierarchical Monitors
Software-Pratice 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
SA800/801 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.

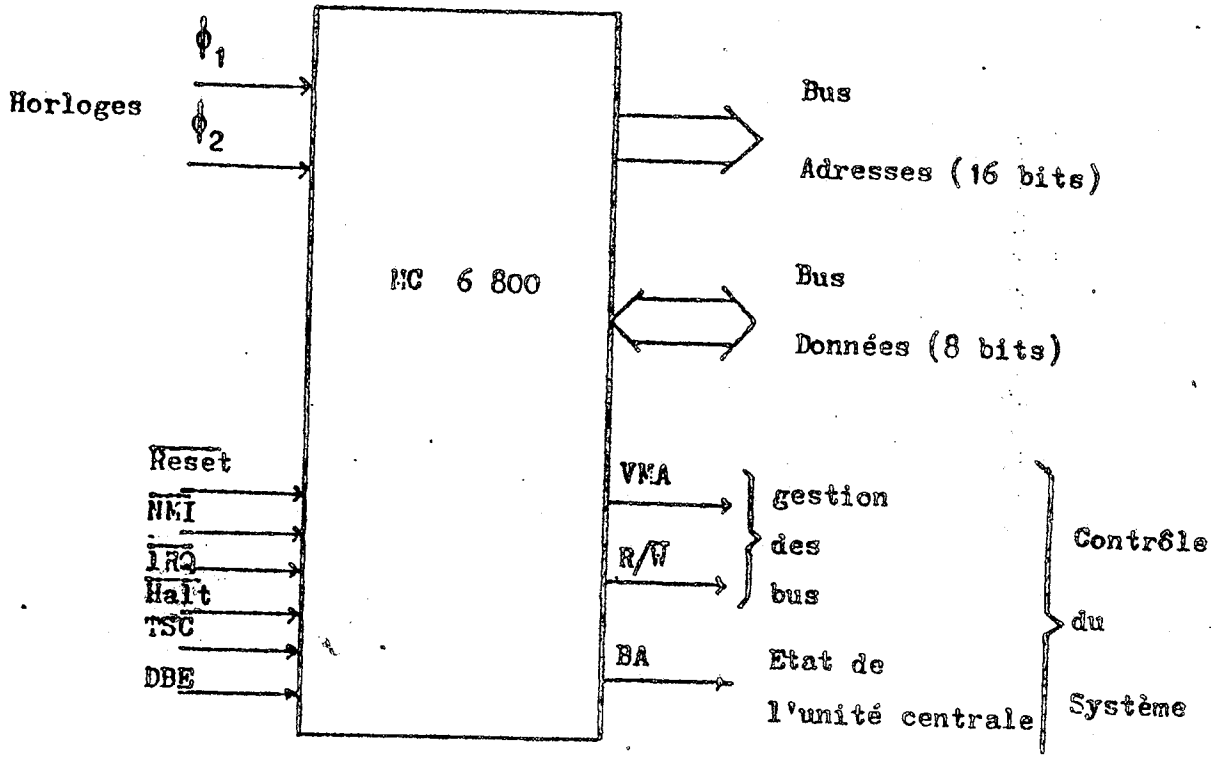
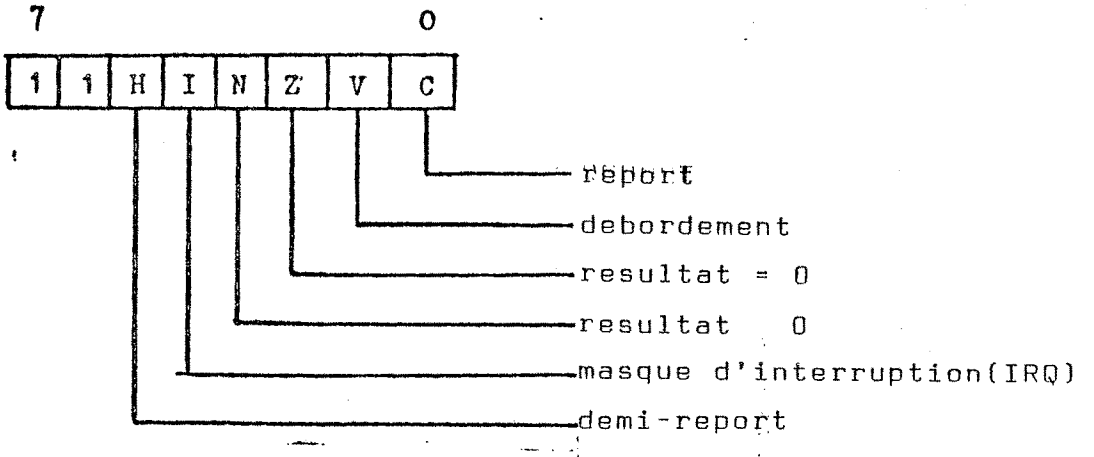


Figure A1.1 La structure externe du MC6800

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 Adress(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($R/\overline{W}$).

La signal $R/\overline{W}$ est une sortie qui indique aux mémoires et aux périphériques si le MC6800 est en lecture($R/\overline{W}=1$) ou en écriture($R/\overline{W}=0$). 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($\overline{HALT}$).

- 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 éxécute une instruction d'attente (WAIT).

- Tri-State Control(TSC).

Quand TSC est à l'état haut, le BUS d'adresse et le signal $R/\overline{W}$ 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(Ø1 et Ø2) doivent être maintenues respectivement haute et basse, dans le but de retarder l'éxécution 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 donnés. 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 Ø2.

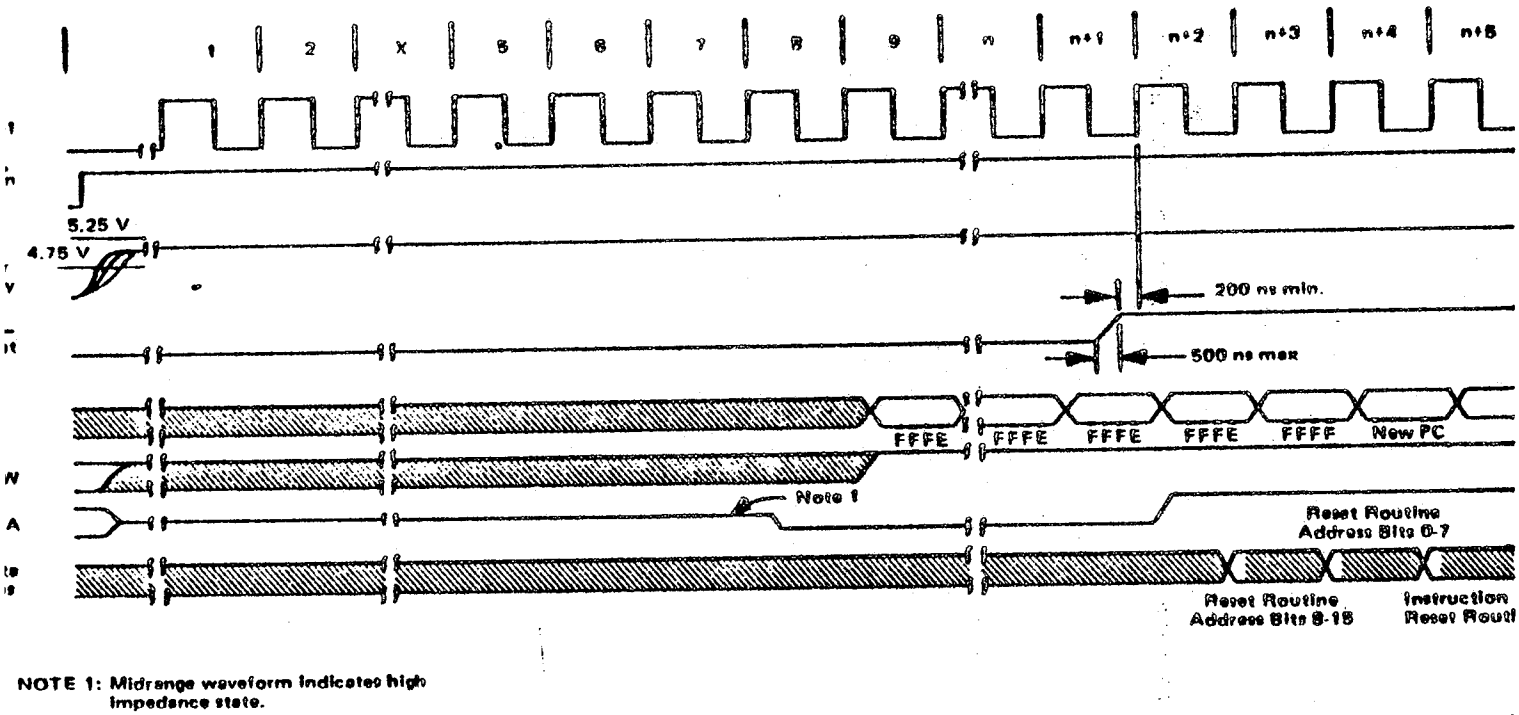
c) Signaux d'interruptions.

Le microprocesseur possède 4 autres entrées de contrôle: $\overline{\text{RESET}}$,$\overline{\text{NMI}}$,$\overline{\text{IRQ}}$, $\overline{\text{HALT}}$.

- RESET.

Cette entrée est utilisée pour démarrer le microproceseur à la mise sous tension, ou lors d'un mauvais fonctionnement. Les deux dernières cases mémoires (FFFE-FFFF) 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($\overline{NMI}$).

Le MC6800 peut prendre en compte 2 types d'interruptions: $\overline{IRQ}$ qui est masquable par un bit du registre d'état(CCR), alors que $\overline{NMI}$ 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), exepté 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 éxécutée; 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-FFFD pour $\overline{NMI}$ et FFF8-FFF9 pour $\overline{IRQ}$.

- $\overline{HALT}$.

La ligne $\overline{HALT}$ permet le contrôle de l'éxécution d'un programme par une source extérieure. Quand $\overline{HALT}$ 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 R/$\overline{W}$ 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 $\overline{HALT}$. La figure A1.3 montre l'effet de $\overline{HALT}$ sur le microprocesseur.

Pour tester des programmes, il est interessant de les éxécuter instruction par instruction. Pour cela, HALT doit être maintenu haut et retomber comme indiqué à partir du point A sur la figure A1.3.

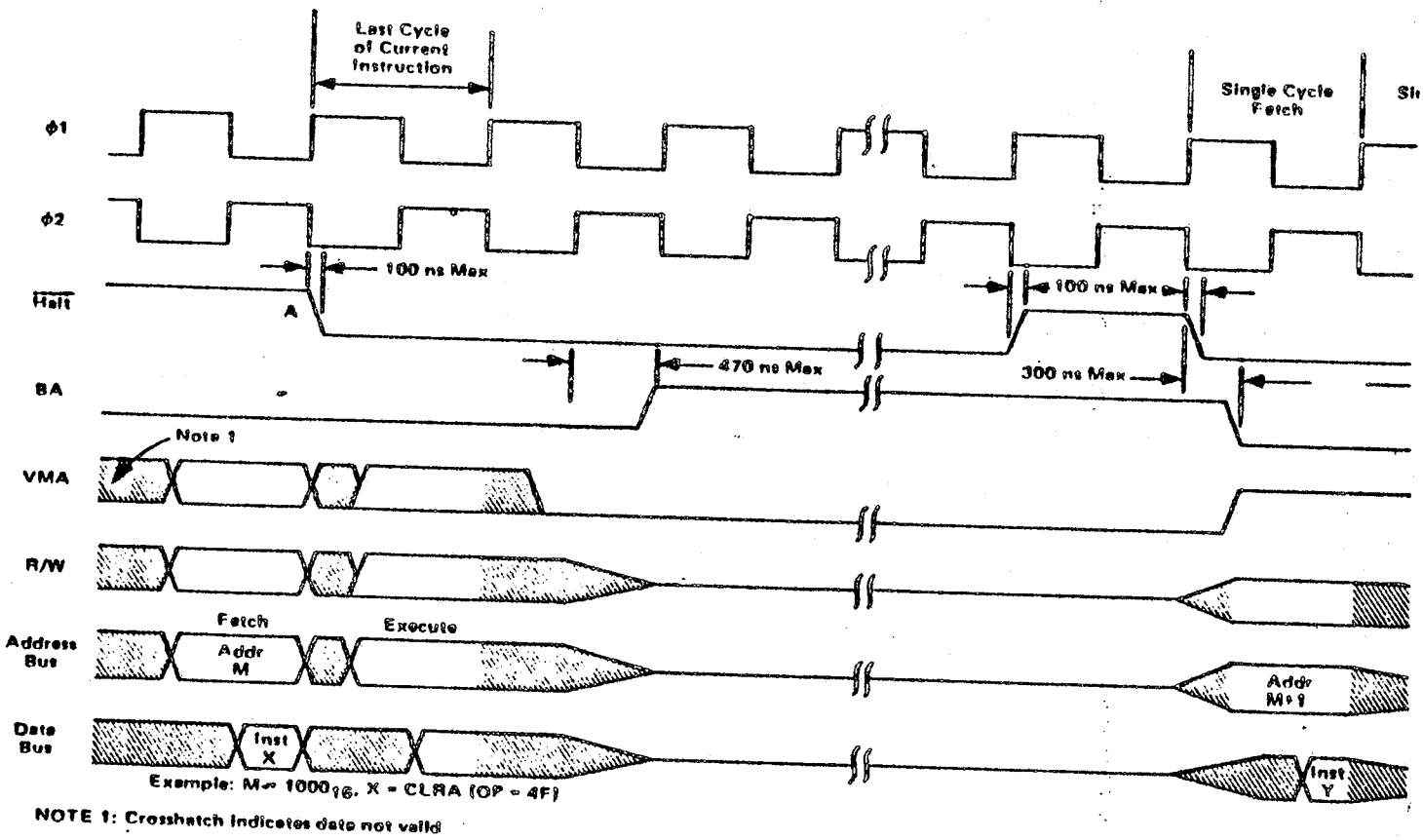


Figure A1.3 Fonctionnement pas à pas.

d) Le jeu d'instructions(RTI-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 éxécute 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.

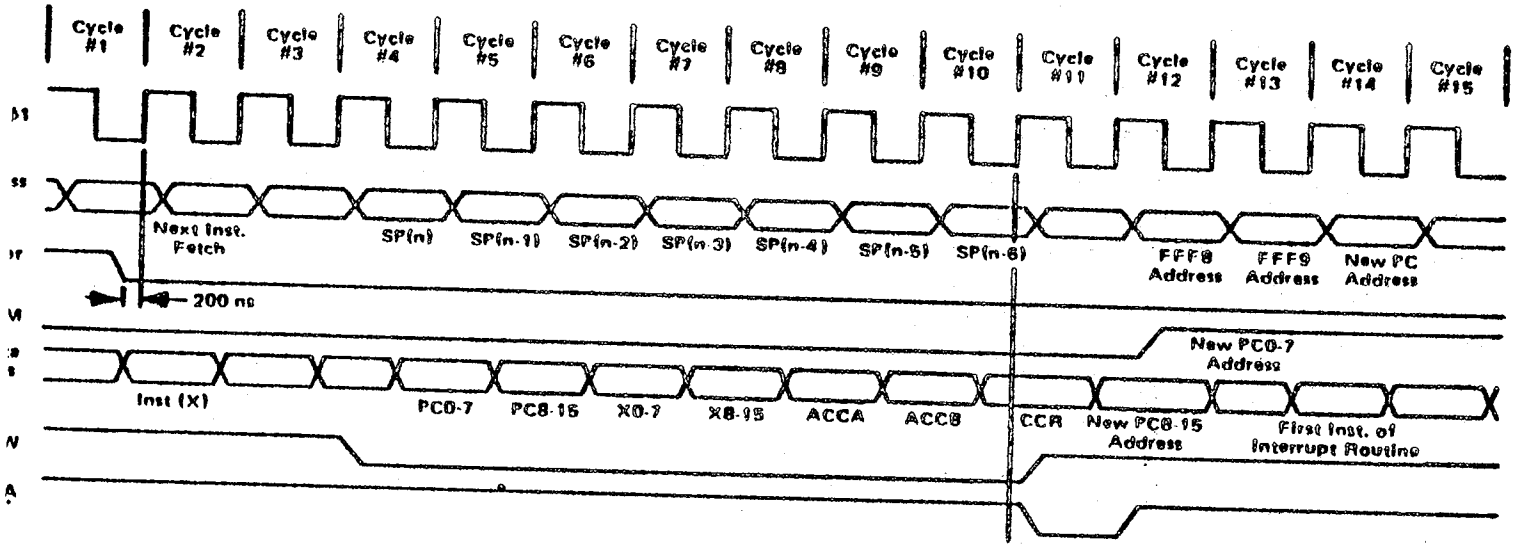


Figure A1.4a Diagramme des temps pour les interruptions.

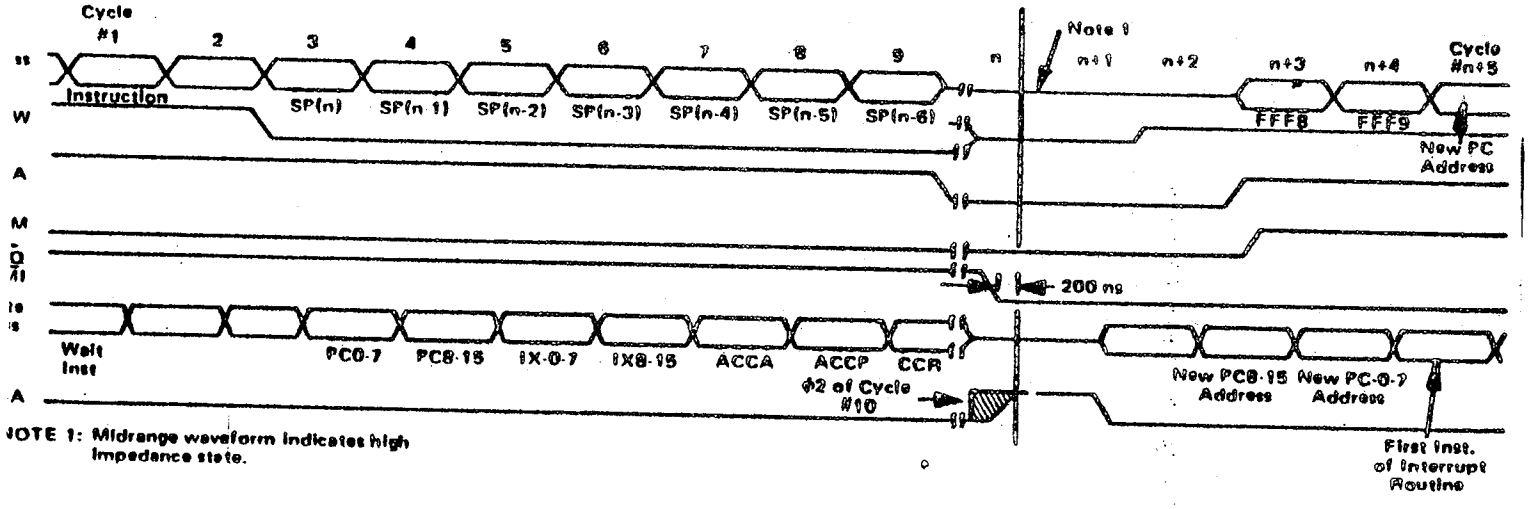


Figure A1.4b Diagramme des temps pour l'instruction WAIT.

1.1.2) L'horlogerie centrale.

L'horlogerie centrale est organisée autour d'un circuit MOTOROLA (MC6871B) qui génére les phases de travail du MC6800(Ø1 et Ø2) et des signaux de contrôle propres au MC6800. Il génére également les signaux de synchronisation des composants du P.central.

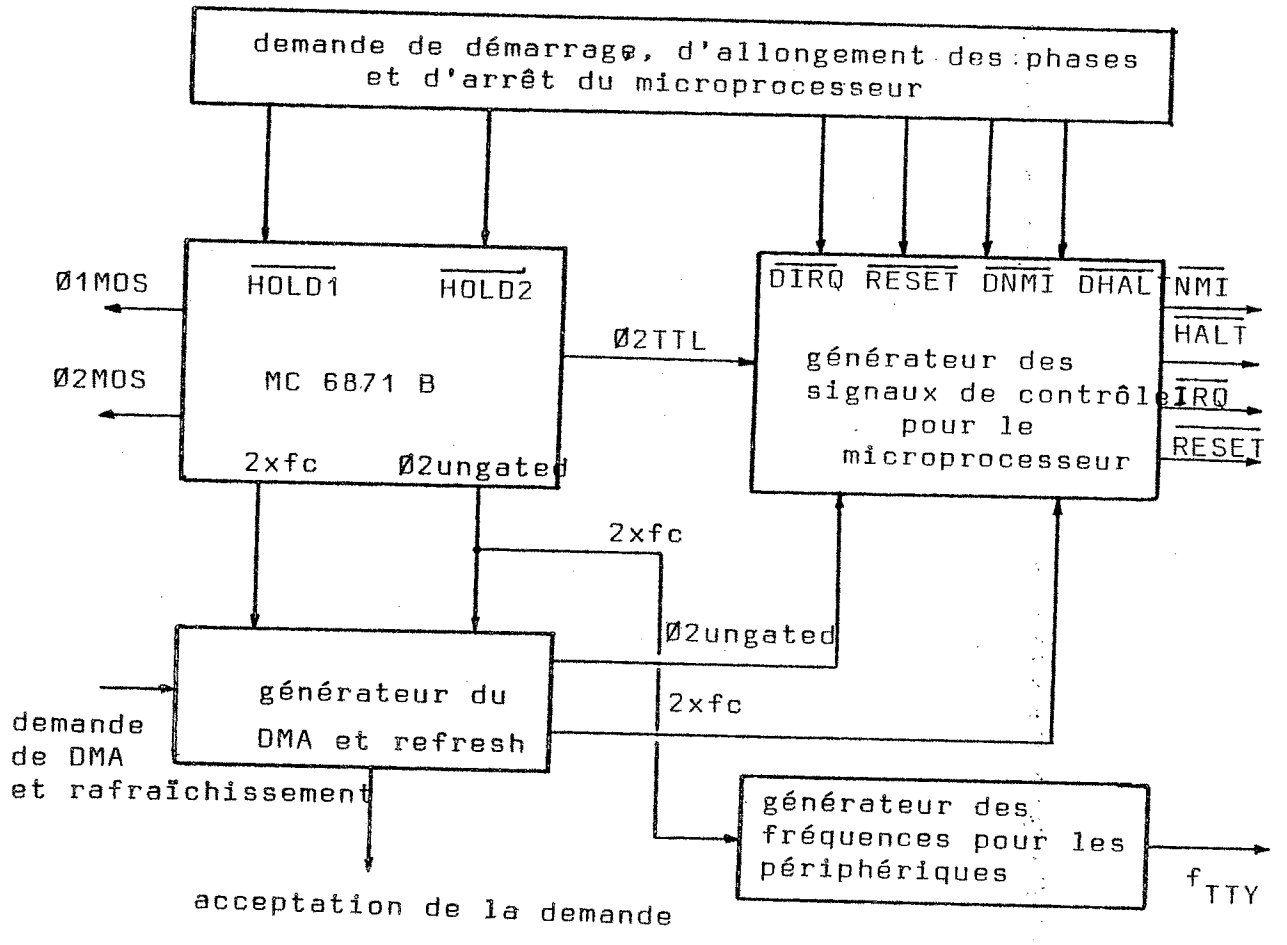


Figure A1.5 L'horlogerie centrale

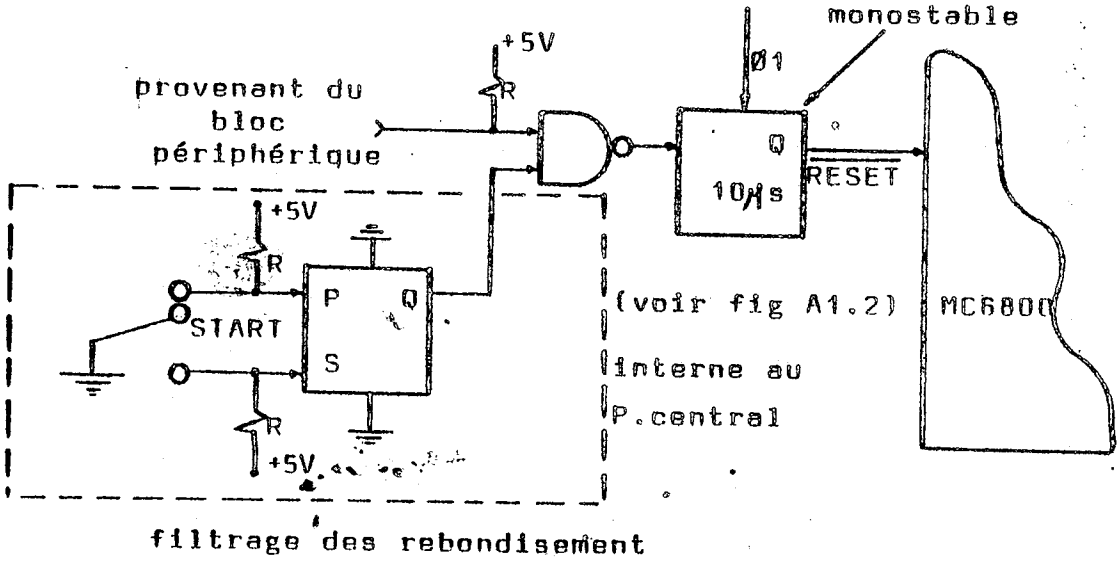
1.1.2.1) L'horloge principale: MC6871B

- Caractéristiques des signaux d'entrée et de sortie du MC6871B

* $\overline{\text{HOLD2}}$: 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.
* $\overline{\text{HOLD1}}$: Demande de retarder l'éxécution 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 $\overline{\text{HOLD2}}$ et $\overline{\text{HOLD1}}$.
* Ø2TTL: Ce signal correspond à la phase Ø2TTL.
* Ø1MOs,Ø2MOs: Ce sont les signaux de travail du MC6800
* 2*fc: deux fois la fréquence de l'horloge principale.

- Demandes de démarrage, d'allongement des phases et d'arrêt du microprocesseur.

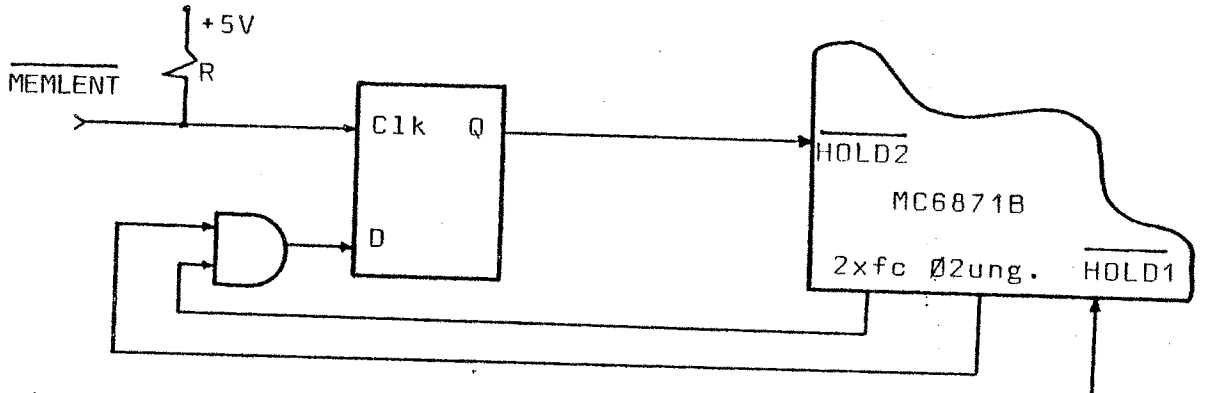
* demande de démarrage.



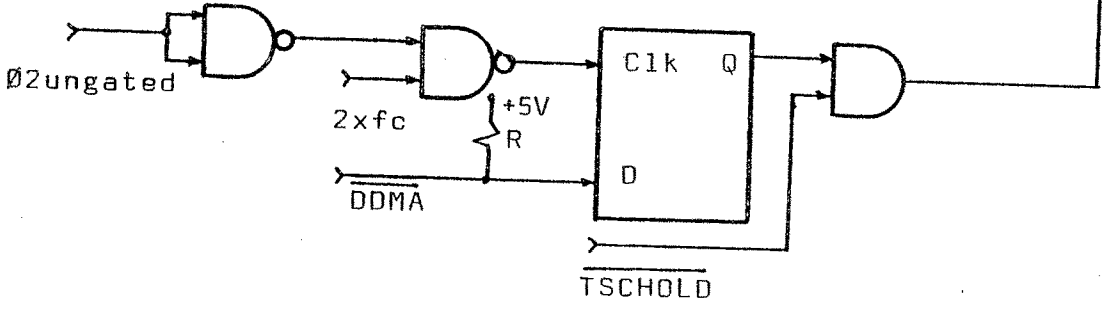
filtrage des rebondisement

* demande d'allongement.

a) cas des mémoires lentes



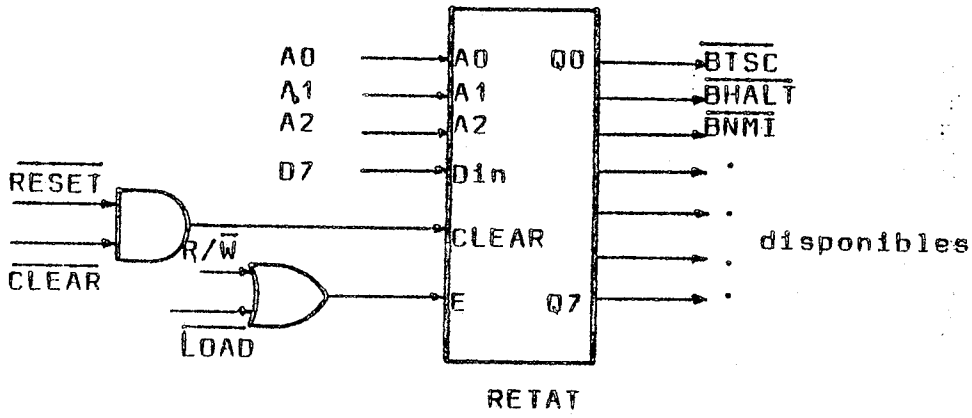
b) cas du rafraîchissement ou de transfert en DMA



$\overline{\text{DDMA}}$: demande de rafraîchissement ou d'accès direct à la mémoire.

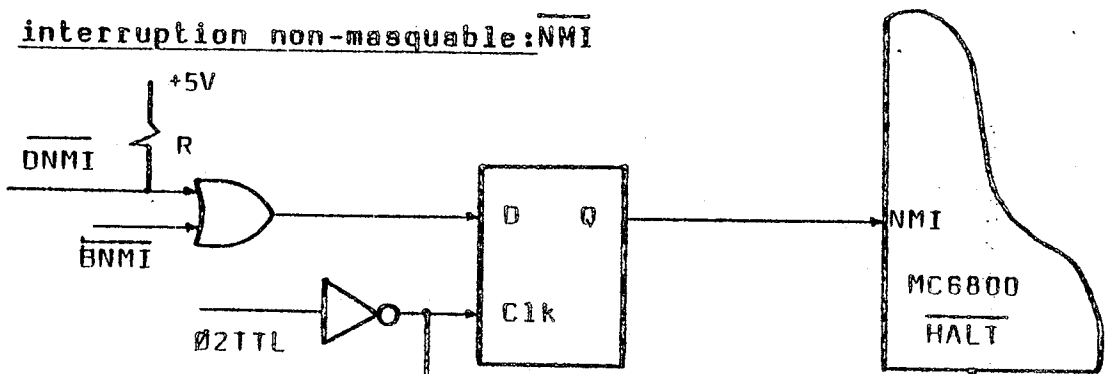
- Demandes d'interruption du microprocesseur.

Grâce à un registre d'état(RETAT), nous pouvons masquer, par le logiciel système, l'arrivée des demandes d'interruption($\overline{NMI}$,$\overline{HALT}$,TSC) au MC6800. Le démarrage du MC6800 permet de démasquer ces interruptions.

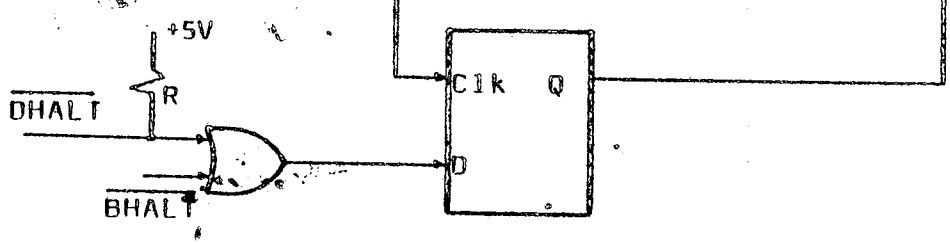


* synchronisation des demande d'interruption.

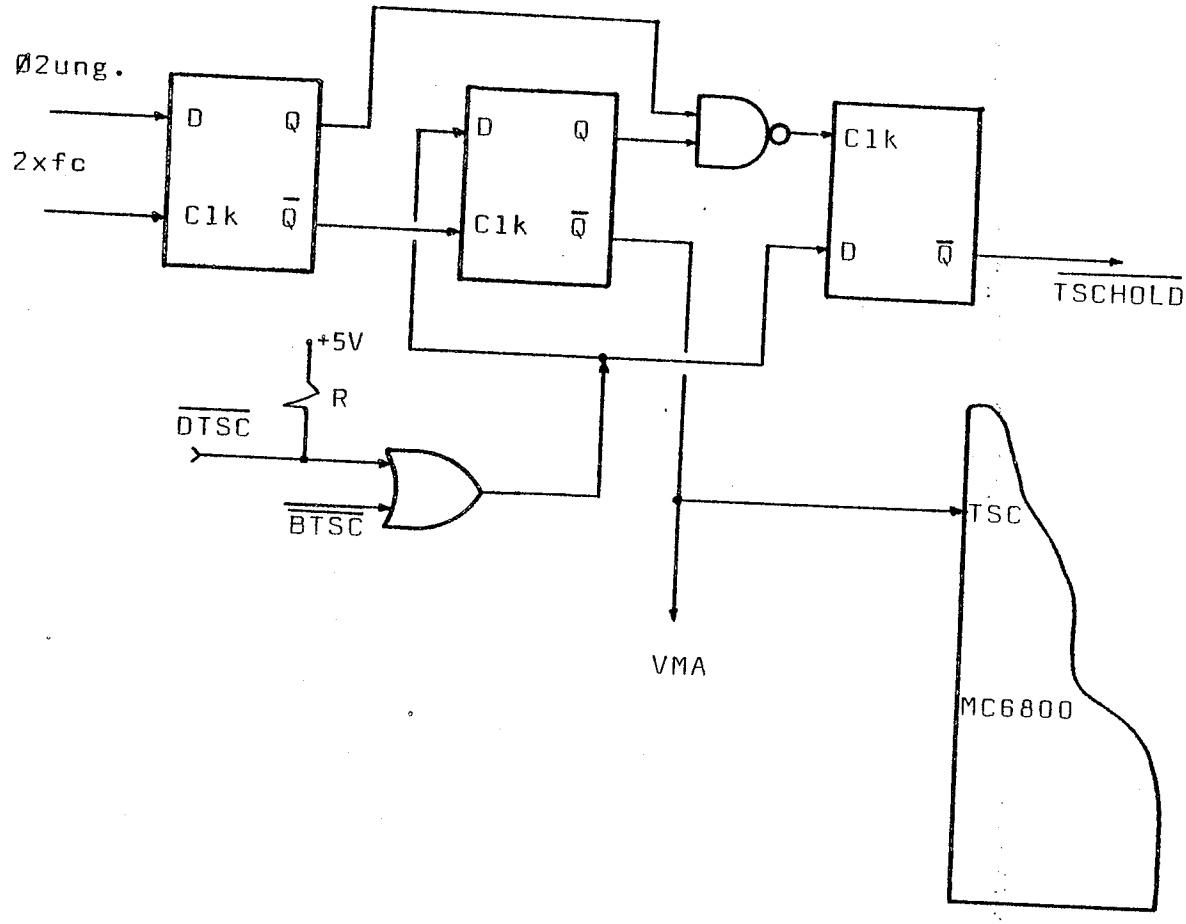
a) interruption non-masquable:$\overline{NMI}$



b) arrêt par $\overline{HALT}$.



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 génére le BUS de pages.

1.1.3.1) L'espace système.

Cet espace est représenté par l'espace ou 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($\overline{RESTART}$,$\overline{NMI}$,$\overline{IRQ}$,SWI) s'y trouve(RTI-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($\overline{PF}$).

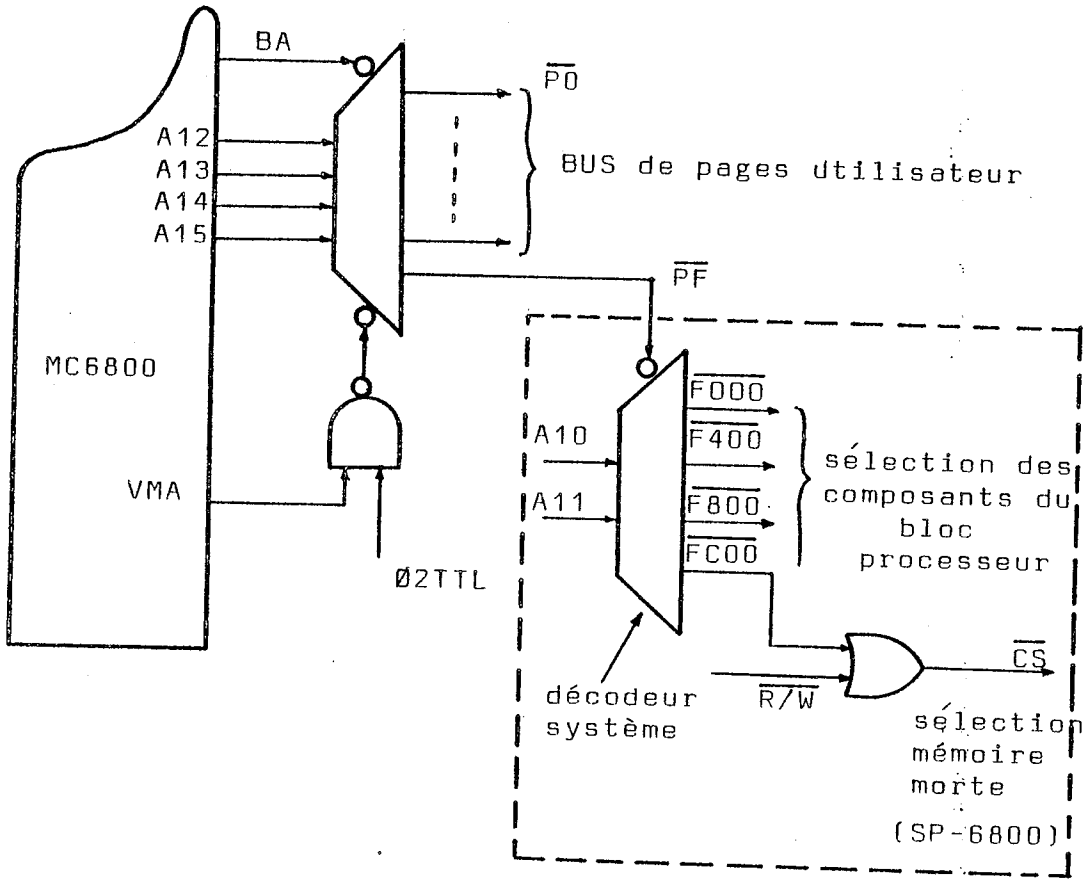


Figure A1.6 L'espace d'adressage du P.central

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.

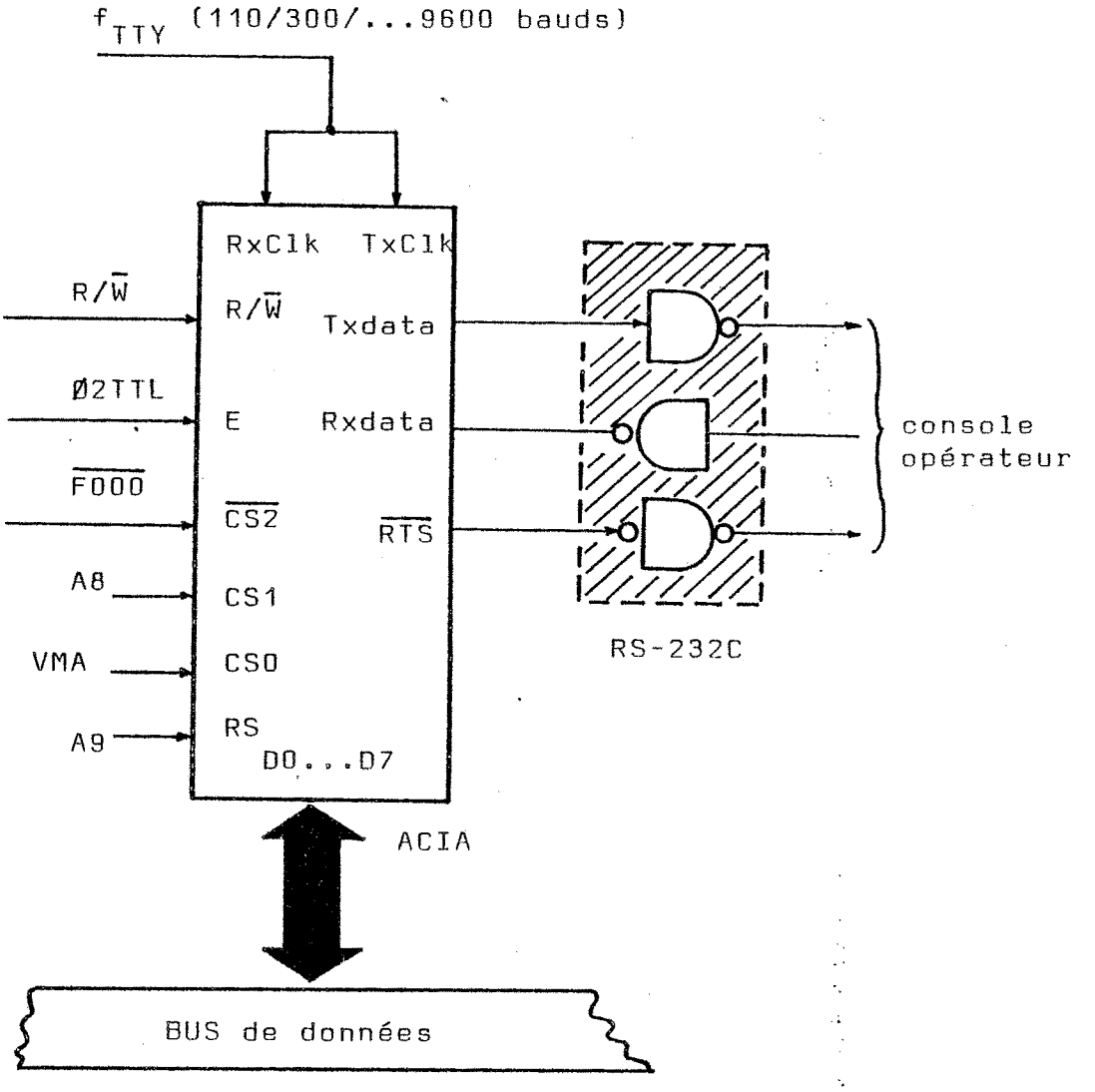


Figure A1.7 Connection de l'ACIA au système.

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 précédement 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(ITi) qui contient l'adresse d'éxécution du programme utilisateur d'interruption (PITi).

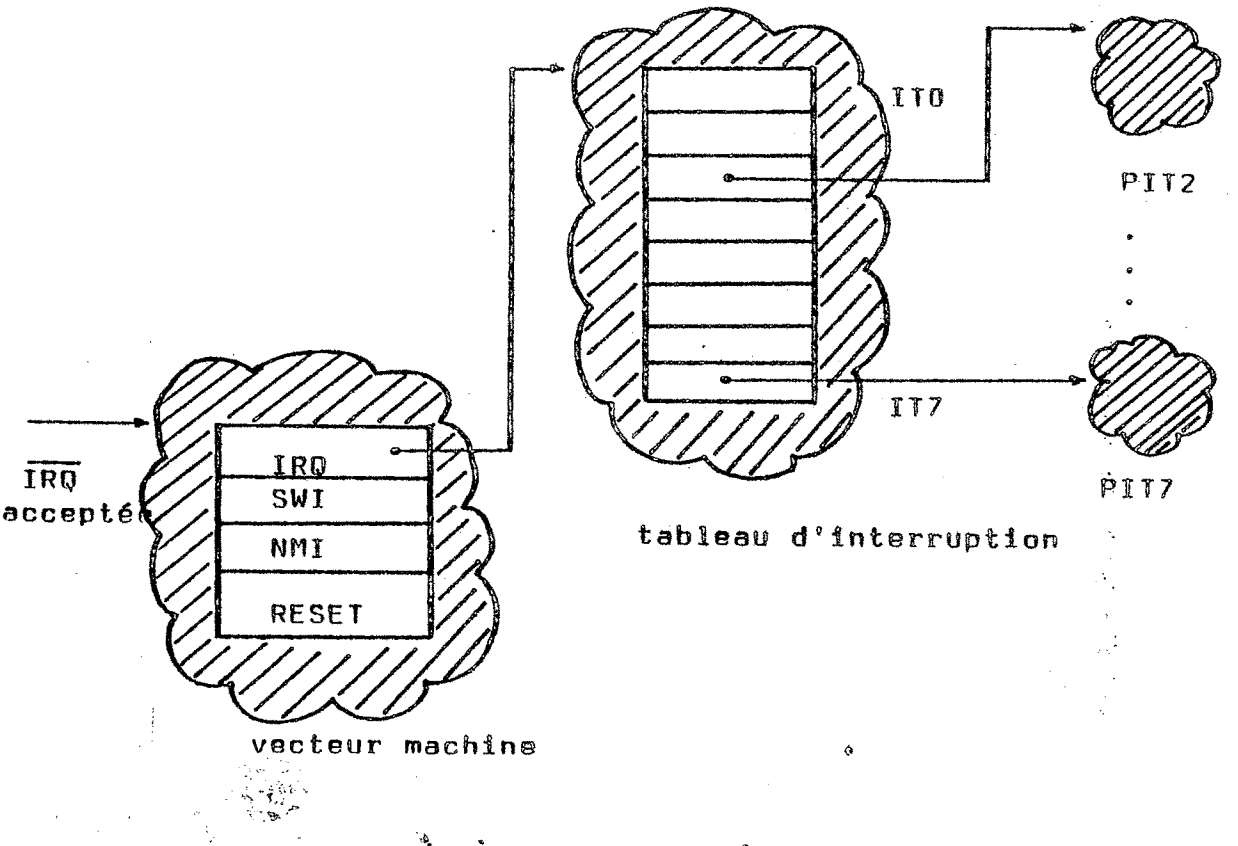


Figure A1.8 Le déroulement d'une interruption IRQ.

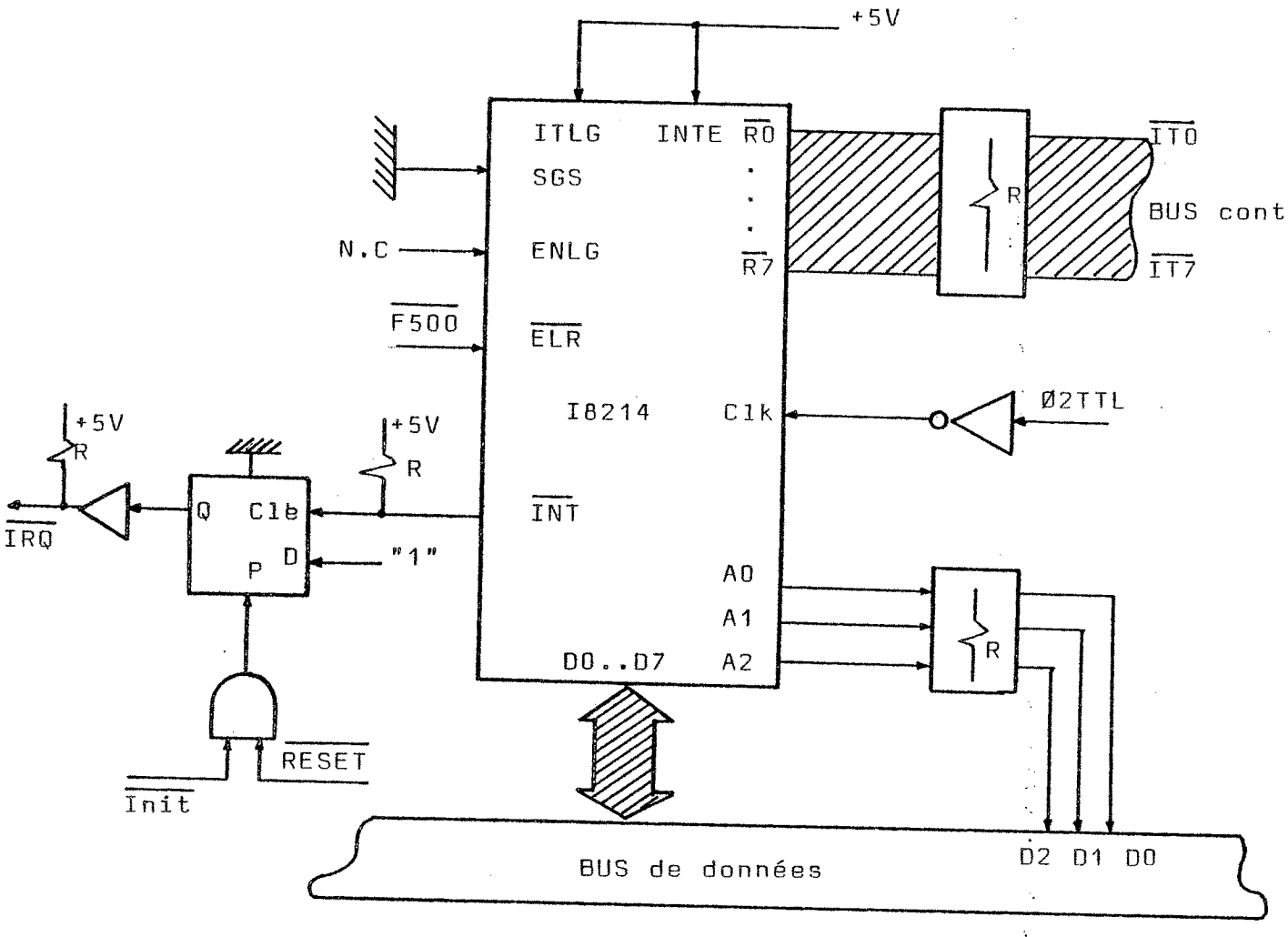


Figure A1.9 Le circuit contrôleur des interruptions.

2.) LE BLOC PERIPHERIQUE.

2.1) Le module interface.

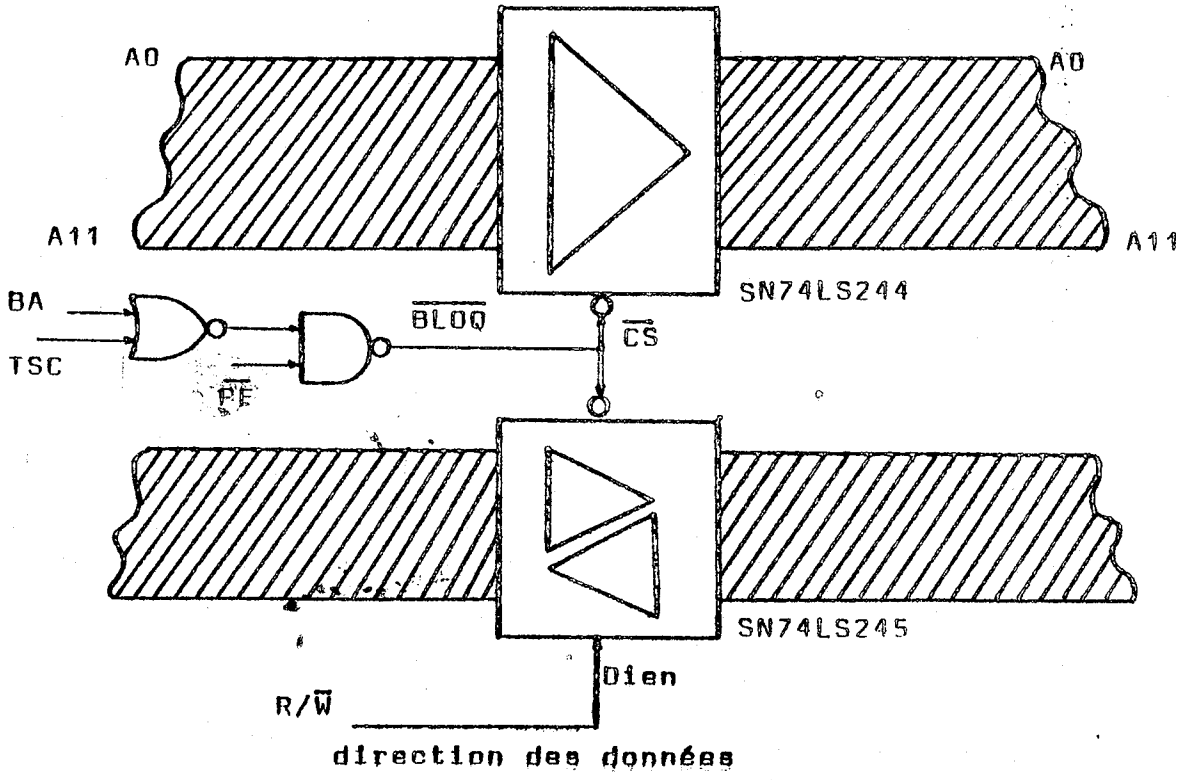
2.1.1) L'interface adresse.

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.

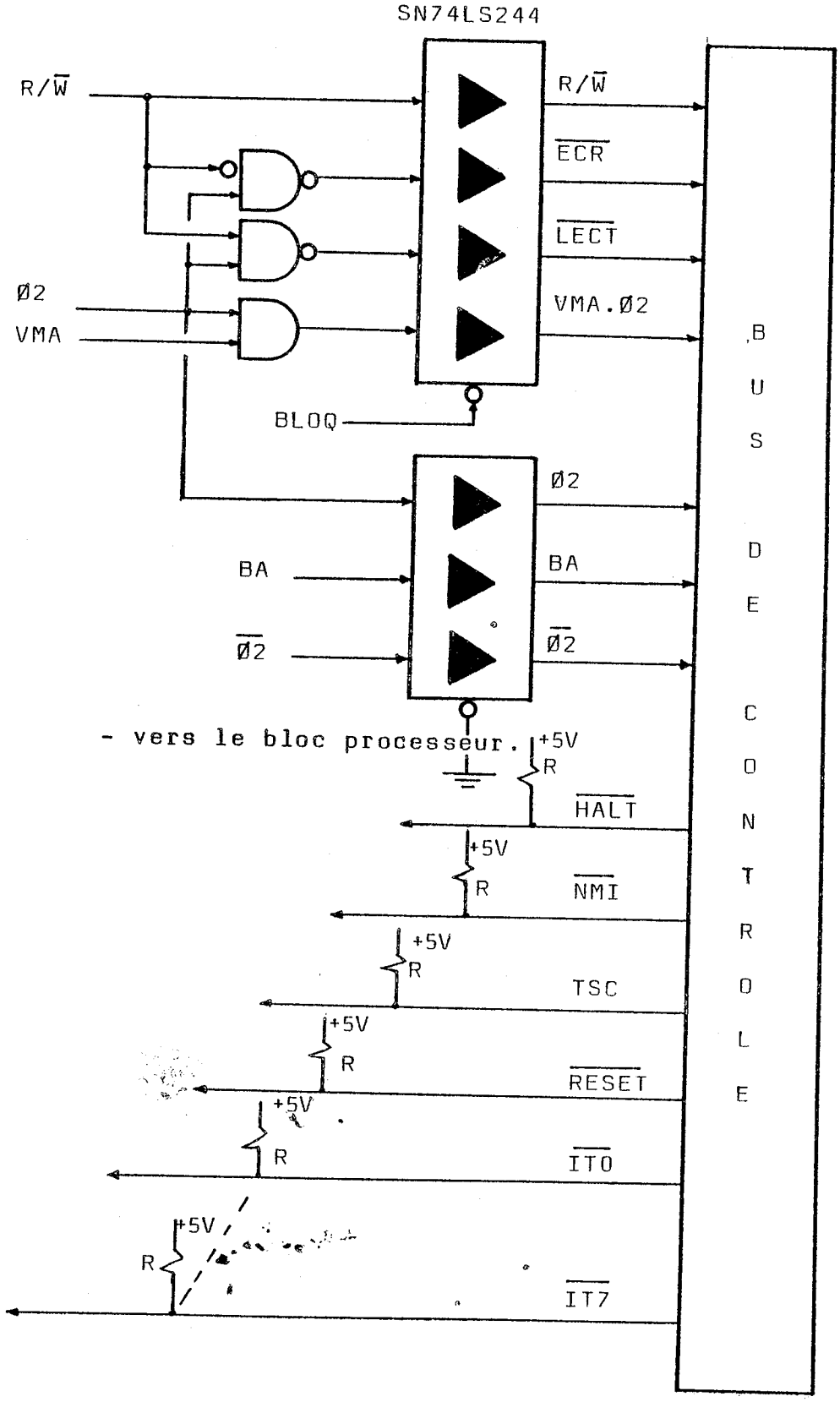
L'interface pour les données est conçue par un circuit à trois états du type SN74LS245; ce circuit est un amplificateur bidirectionnel.

La fonction principale du module interface est d'isoler le bloc processeur du bloc périphérique quand une interruption d'arrêt au microprocesseur est acceptée ($\overline{HALT}$,TSC) et aussi quand la page système est sélectionnée ($\overline{PF}$).



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.

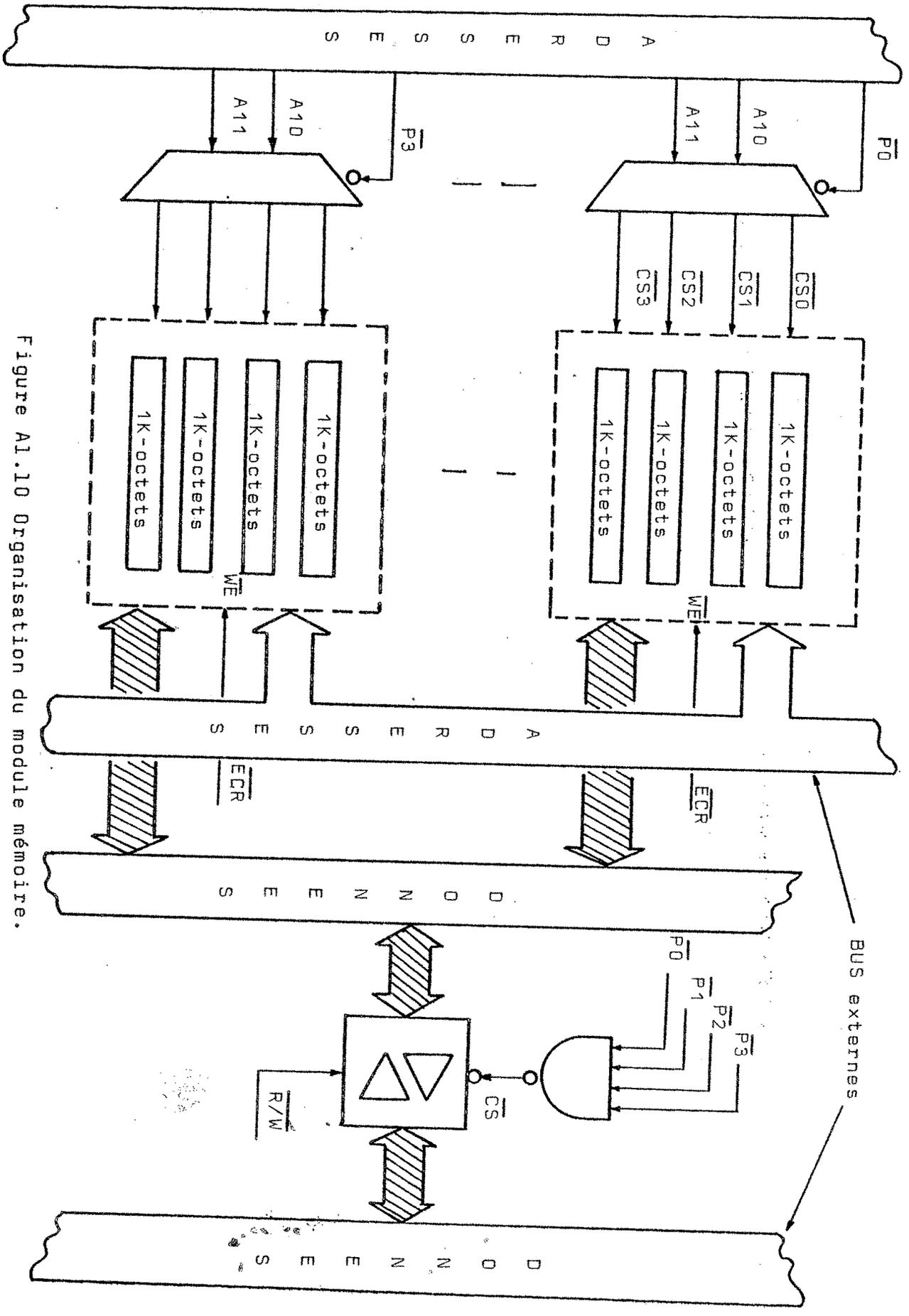


Figure A1.10 Organisation du module mémoire.

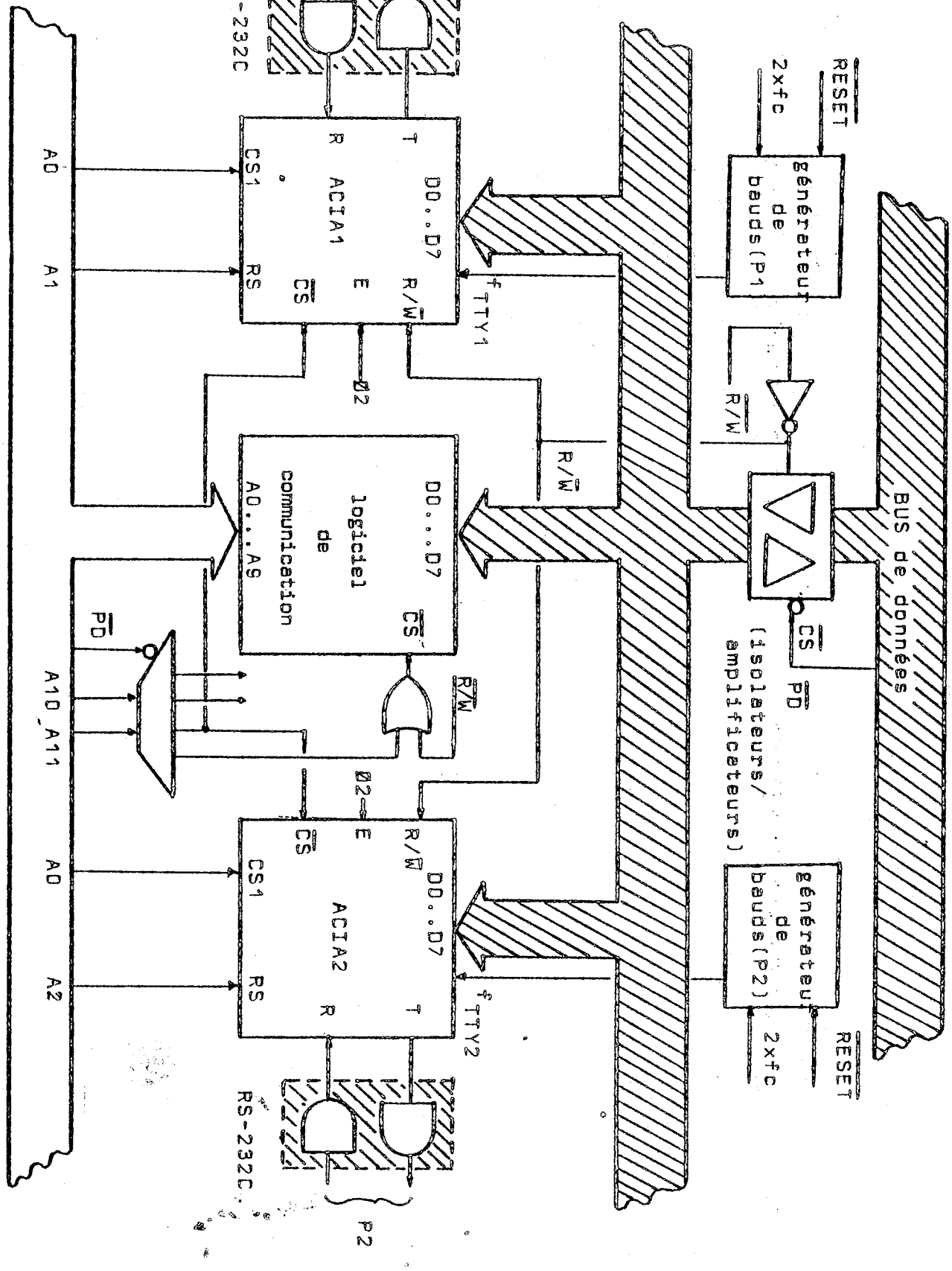


Figure A1.11 Organisation du module d'extention des entrées/sorties.

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 dès commandes des qu'il trouve le caractère de contrôle S9.

Le format est décrit dans la figure A2.1.

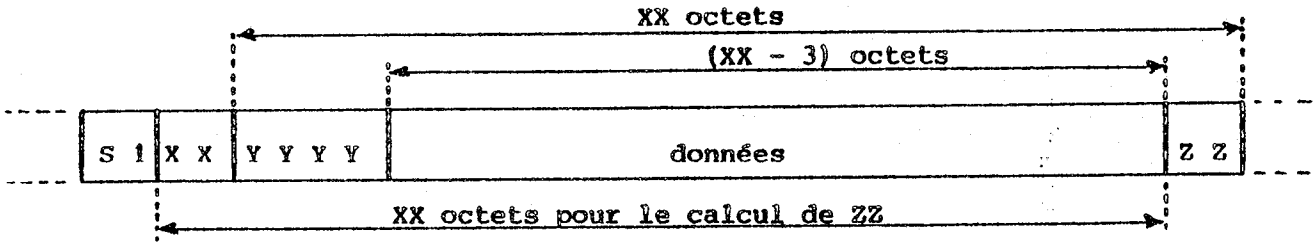


Figure A2.1 Format des données en entrée du chargeur

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> <adrl>
<adr>::= <adrl>::= <hexa4>
<hexa4>::= (4 caractères hexadécimaux)

Exemple :
```
* P 0012 0043
S1130012000OCC0000CC5D504D42CF400000CC40EB 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>
<déc.adr>::= (U) CO <déc.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 adressse 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 ($R/\overline{W}$) 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><adr1>(RC)
<commande> ::= (E)
<adr>::= (adresse de départ)
<adr1>::= (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:

<modreg>::= <commacu>/<commreg>(RC)
<commacu>::= <accu><état accumulateur>
<accu>::= (A)/(B)
<état accumulateur>::= (=)<contenu de l'accumulateur>
<contenu de l'accumulateur> ::= <hexa2>
<commreg>::= <registres><état registres>
<registres>::= (C)/(S)/(X)/(Z); CO C est le compteur ordinal et Z le code condition de la machine;
<état registres>::= (=)<contenu du registre>
<contenu du registre>::= <hexa4>

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 interrup(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.

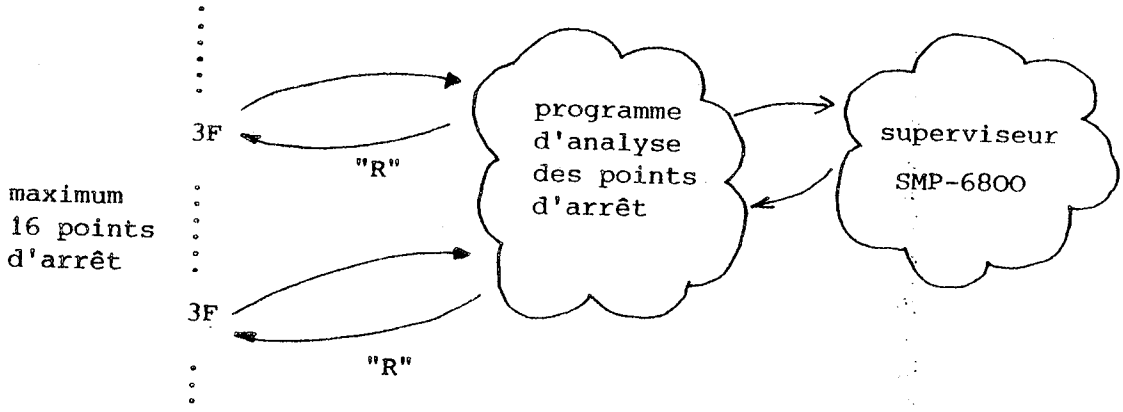


Figure A2.2 Le mécanisme des points d'arrêts.

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 A1. Commandes normales.

* M <adr> = <contenue>(RC) ... examen du contenu d'une adresse

<contenu>::= xx; CO un octet;

* M <adr> = xx yy(RC); CO modification(yy) du contenu d'une adresse

* M <adr> = xx yy zz ss(RC)

* M <adr> = ss(U); CO demande d'examen du contenu de l'adresse précédente.

<adr-1> = cc(RC); CO retour aux commandes par (*)

* M <adr> = rr(N); CO demande d'examen de la prochaine adresse

<adr+1> = pp xx(RC)

* G <adr>; CO lancement de programmes

* L .... chargement d'un ruban

* P <adr> <adr1>; CO impression/perforation d'une zone mémoire sous format chargeur.

* E <adr> <adr1>; CO test d'une zone mémoire

* F .... élimination des points d'arrêt

* K <adr>(RC)... pose de points d'arrêts: max 16

* T <adr> R .... test électronique en lecture

* T <adr> W XX ... test électronique en écriture

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

RTCH: 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 chaine 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.

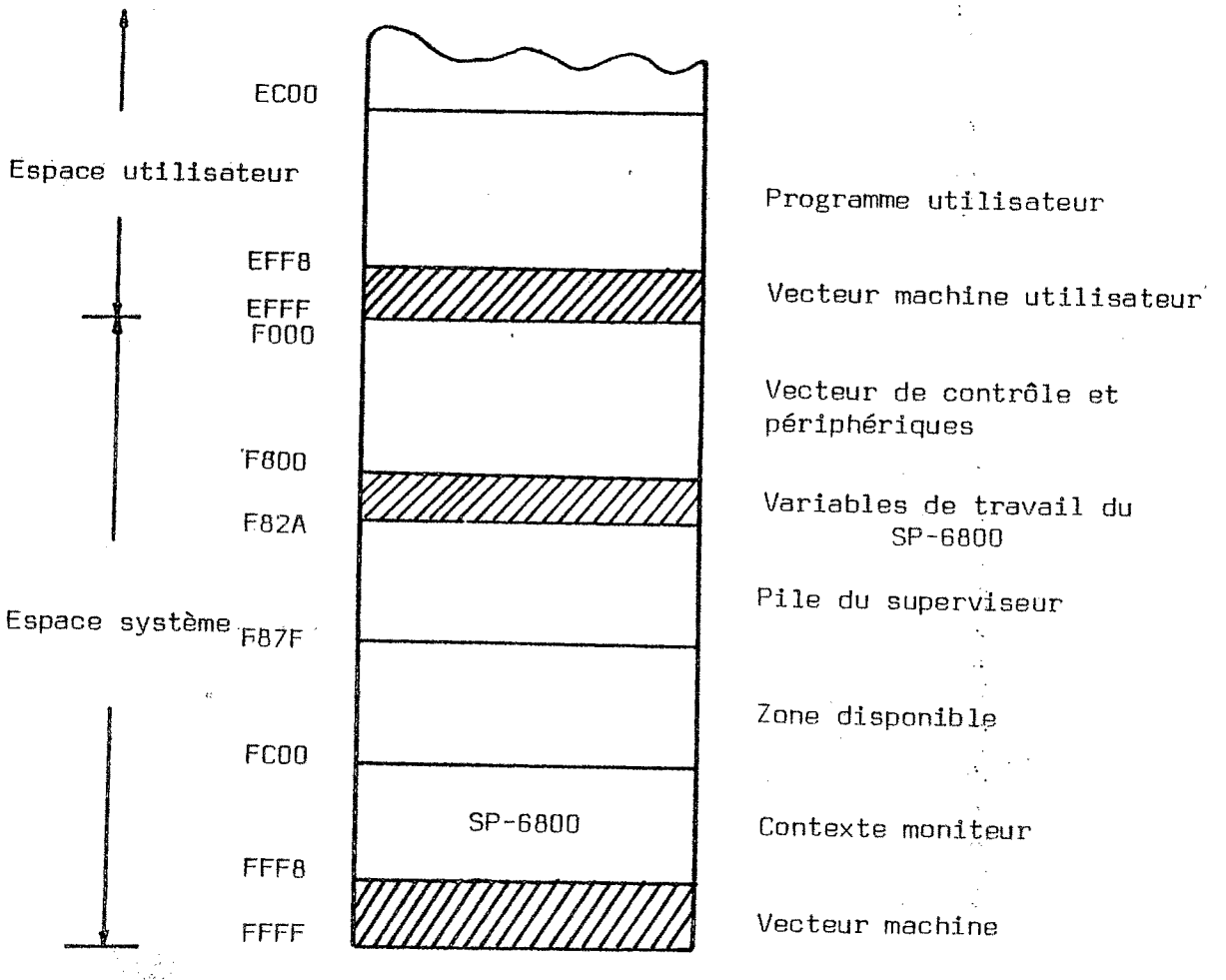


Figure A2.3. Organisation de l'espace mémoire.

Cette zone se décompose en:

a) Vecteur machine: FFF8-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 normallement 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 la pile système.

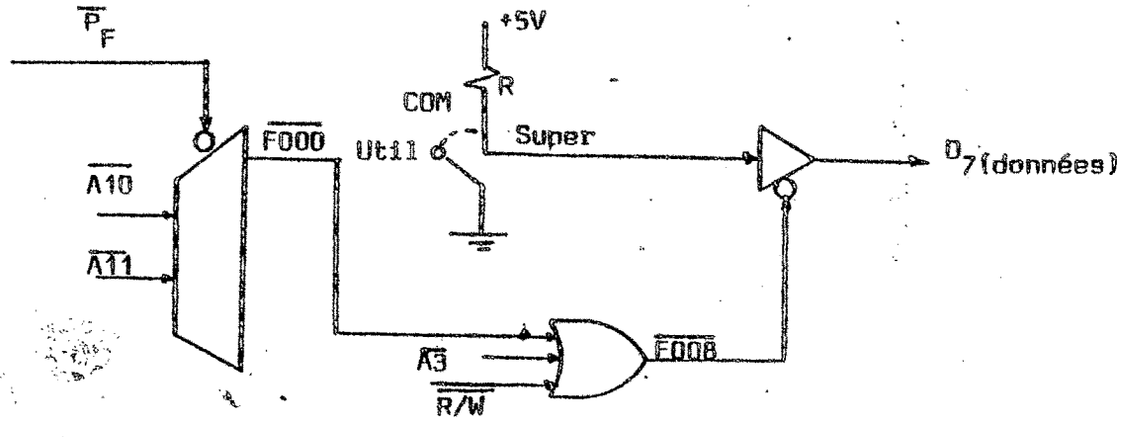


Figure A2.4 Mécanisme logiciel.

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(EEE8-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.

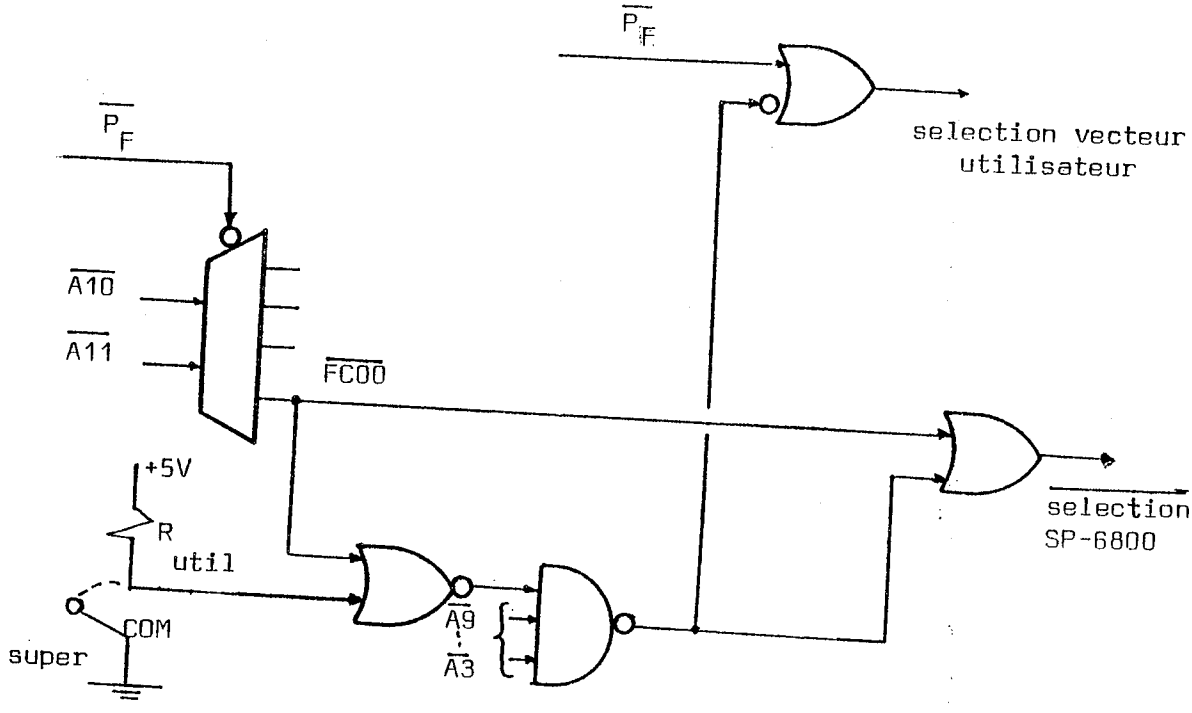


Figure A2.5 Confusion des vecteurs utilisateur et machine.

2) <u>Confusion sur l'espace utilisateur.</u>

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.

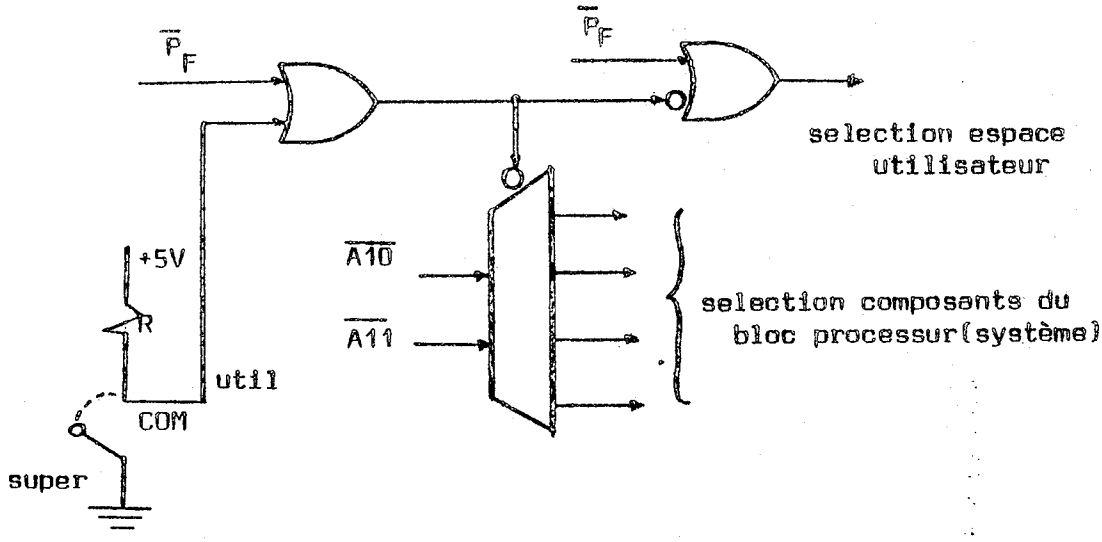


Figure A2.6 Confusion sur l'espace utilisateur.

5.) LISTING DU PROGRAMME MONITEUR SP-6800.

```
PROGRAMME : SP-6800          ASSEMBLEUR MICROPROCESSEUR  EQUIPE ARCHITE

CE  CODE                          INST   OPERANDES      COMMENTAIRES

                    1 ****************************************
                    2 *          SUPERVISEUR 6800            *
                    3 *          A.ZAMBRANO  E.N.S.I. .A.S   *
                    4 ****************************************
0000 0000           5 ZERO      EQU     0
0000 0001           6 A         EQU     1
0000 0002           7 XHX       EQU     2
0000 0011           8 RON       EQU     =X11            R.AFFER ON
0000 0013           9 RESE      EQU     =X13            R-AFFER-OFF
0000 0012          10 PERA      EQU     =X12            PERF.ON
0000 0014          11 PERI      EQU     =X14            PERF.OFF
                   12 *
                   13 * ADRESSE ACIA MAQUETTE
                   14 *
0000 E200          15 ETAT      EQU     =XE200
0000 E300          16 DONA      EQU     =XE300
0000 E97E          17 PILE      EQU     =XE97E
0000 E400          18 MODE      EQU     =XE400
                   19 ********************************
E800 E800          20           ORG     =XE800
                   21 ********************************
E800 0000          22 PCI       DCN     1
E801 0000          23 XSAV      DCN     2
E803 0000          24 SPTI      DCN     2
E805 0000          25 SWI       DCN     2
E807 0000          26 VAR       DCN     2
E809 0000          27 CKSM      DCN     1
E80A 0000          28 RCOM      DCN     1
E80B 0000          29 AFCC      DCN     1               ECHO
E80C 0000          30 BEGA      DCN     2
E80E 0000          31 ERDA      DCN     2
E810 0000          32 TBIT      DCN     24              8 POINTS D'ARRET
E828 E829          33 ETBT      EQU     *+0
                   34 ***************
                   35 * VECTEUR IRQ
                   36 ***************
E828 0028          37 VIRQ      DCN     ?
E82A 0028          38 SECU      DCN     1
                   39 ***************
```

```
[illegible] : SP-6800          ASSEMBLEUR MICROPROCESSEUR [illegible] ARCHIT

C[illegible]                        INST   OPERANDES      COMMENTAIRES

FC00 [illegible]                    ********************************************
                                    ORG    =XFC00
                                    ********************************************
                                    *              RECONNAISSANCE [illegible] CHIFFRE HEXA
                                    * L'ECHO EST SUPPRIMEE SI LE CARACTERE N'EST
                                    * RETOUR: CARRY=0 SI HEXA
                                    *          " =1 NON HEXA
                                    ********************************************
FC00 8059                           PSR    SPAC
FC02 7CF809                 LHEX    INC    AECO           SUPRESSION ECHO
FC05 BDFCD7                         JSR    LCAR
FC08 7AF809                         DEC    AFCO           RESTAURE ECHO
FC0B 8140                   LHRR    CMPI   A,=X40           <A
FC0D 2F08                           PLE    EI
FC0F 8146                           CMPI   A,=X46           >F
FC11 2215                           PHI    FRH
FC13 8009                           ADDI   A,=X09
FC15 2008                           PRA    RETR
FC17 812F                   EI      CMPI   A,=X2F           <0
FC19 2F09                           PLE    ERH
FC1B 8139                           CMPI   A,=X39           >9
FC1D 2209                           PHI    FRH
FC1F 840F                   RETR    ANDI   A,=X0F
FC21 36                             PSHA
FC22 BDFCAA                         JSR    PHEX
FC25 32                             PULA
FC26 0C                             CLC
FC27 39                             RTS
                                    *
                                    *
FC28 0D                     ERH     SEC
FC29 39                     RTET    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 OCCT   BSR    SPAC
FC2C 8D04                         80 LCCT   BSR    LHEX
FC2E 2514                         81        BCS    ERR1
FC30 37                           82 LCC1   PSHB
FC31 48                           83        ASLA   A
FC32 48                           84        ASLA   A
FC33 48                           85        ASLA   A
FC34 48                           86        ASLA   A
FC35 16                           87        TAB
FC36 8DCA                         88        BSR    LHEX
FC38 2504                         89        BCS    ERR1
FC3A 1B                           90        ABA
FC3B 16                           91        TAB
                                  92 *
                                  93 * MISE AU POINT CHECK-SUM
                                  94 *
FC3C FBF809                       95        ADDB   B,CKSM
FC3F F7F809                       96        STA    B,CKSM
FC42 33                           97        PULB
FC43 39                           98        RTS
                                  99 *
FC44 7DF802                      100 ERR1   TST    AFCO
FC47 2703                        101        BEQ    CLAV
FC49 7EFF64                      102        JMP    ERDR           ERREUR CHARGEME
FC4C 2029                        103 CLAV   BRA    CO1            ERREUR CLAVIER
                                 104 *
                                 105 *   CONSTRUCTION D'UNE ADRESSE
                                 106 *
FC4E CEF801                      107 CADR   LDX1   XSAV
FC51 8DD7                        108 CAD1   BSR    OOCT
FC53 A700                        109        STA    A,ZERO(X)
FC55 8DC5                        110        BSR    LCCT
FC57 A701                        111        STA    A,UN(X)
FC59 EF00                        112        LDR    X,ZERO(X)      X.... ADRESSE
                                 113 *
                                 114 * IMPRESSION D'UN ESPACE
FC5B 8620                        115 *
FC5D 2051                        116 SPAC   LDA    A,=X20
                                 117        BRA    PCA2
                                 118 *
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITI

```
CE  CODE                                  INST   OPERANDES      COMMENTAIRES

                          120 *
                          121 *
FC5? 3155                 122 MCCT  BSR    PCCT
FC61 8D9D                 123       BSR    LHX1
FC63 25C4                 124       BCS    RTTT
FC65 8D79                 125       BSR    LOCI
FC67 A700                 126       STA    A,ZERO(X)
FC69 20F4                 127       BRA    MCCT
                          128 *
                          129 *
                          130 *
                          131 *          EXAMEN/MODIF MEMOIRE
                          132 *
FC6B BDE1                 133 MEM   BSR    GAER
FC6D 9DF0                 134 MEM1  BSR    MOCT
FC6F B14E                 135       CMPI   A,=X4E          NEXT INSTRUCTIO
FC71 2704                 136       BEQ    MEM3
FC73 B155                 137       CMPI   A,=X55          'U'
FC75 2703                 138       BEQ    MEM2
FC77 7EED45               139 CCI   JMP    CCMM
FC7A 09                   140 MEM2  DEGX
FC7B 2001                 141       BRA    MEM4
FC7D 09                   142 MEM3  INCX
FC7E BD4A                 143 MEM4  BSR    RTCH
FC80 9DEFDC               144       JSR    IMAD
FC83 20EA                 145       BRA    MEM1
                          146 *
                          147 *          MISE A JOUR /EXAMEN   ACCUMUL
                          148 *
FC85 30                   149 CC    TSX                    CODE CONDITION
FC86 2006                 150       BRA    MCC
                          151 *
FC88 30                   152 ACCP  TSX
FC89 2002                 153       BRA    MACC
                          154 *
FC8B 30                   155 ACCA  TSX
FC8C 09                   156       INCX
FC8D 09                   157 MACC  INCX
FC8E 9DCF                 158 MCC   BSR    MOCT
FC90 20E5                 159 CC2   BRA    CCI
                          160 *
FC92 CEEB03               161 REGS  LDXI   SPIL
FC95 2009                 162       BRA    MIOR
                          163 *
FC97 30                   164 REGX  TSX
FC98 2003                 165       BRA    MACR
                          166 *
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITEC

CF CODE		INST	OPERANDES	COMMENTAIRES
FC9A 30	168 RSCP	TSX		
FC9B 08	169	INCX		
FC9C 08	170	INCX		
FC9D 08	171 MADR	INCX		
FC9E 08	172	INCX		
FC9F 08	173	INCX		
FCA0 8D14	174 MIDR	BSR	PRCT	
FCA2 08	175	INCX		
FCA3 8D13	176	BSR	PRCT	
FCA5 09	177	DEX		
FCA6 8DA9	178	BSR	CAR1	
FCA8 20F6	179 CF3	BRA	C02	
	180 *			
	181 *			
	182 *	IMPRESSION D'UN CHIFFRE HEXA		
	183 *			
FCAA 9109	184 PHEX	CMPA	A,9	
FCAC 2204	185	BHI	ALPH	
FCAE 8A30	186	ORA	A,=X30	
FCB0 202E	187 PCA2	BRA	PCA1	
FCB2 9B37	188 ALPH	ADDA	A,=X37	
FCB4 202A	189	BRA	PCA1	

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

CE CODE			INST	OPERANDES	COMMENTAIRES
	1 1	*			
	1 2	*			
	1 3	*		IMPRESSION D'UN OCTET PRECODE DE	
	1 4	*			
	1 5	*			
FCB6 9D0E	1 6	PRCT	BSR	EQUL	=
FCB8 A600	1 7	OCCT	LDA	A,ZERO(X)	
FCBA 36	1 8	IMPC	PSHA		
FCBB 44	1 9		LSRA	A	
FCBC 44	2 0		LSRA	A	
FCBD 44	2 1		LSRA	A	
FCBE 44	2 2		LSRA	A	
FCBF 8DE9	2 3		BSR	PHEX	
FCC1 32	2 4		PULA		
FCC2 840F	2 5		ANDA	A,=X0F	
FCC4 20E4	2 6		BRA	PHEX	
	2 7	*			
	2 8	*			
	2 9	*		IMPRESSION DE =	
	2 10	*			
	2 11	*			
FCC6 863D	2 12	EQUL	LDA	A,=X3D	"="
FCC8 2018	2 13		BRA	PCAR	
	2 14	*			

PROGRAMME : SB-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITE

```
CE  CODE                          INST   OPERANDES      COMMENTAIRES

                          216 *
                          217 *        RETOUR CHARIOT
                          218 *
FCCA FEF801               219 RTCH   STX    X,XSAV         SAUVEGARDE L'IN
FCCD CEFEF9               220        LDXI   RCLF
FCD0 BDFF48               221        JSR    PDAT
FCD3 FEF801               222        LDX    X,XSAV         RESTAURE INDEX
FCD6 39                   223 RTH    RTS
                          224 *
                          225 *           LECTURE UN CARACTERE   PUIS EC
                          226 *        SI ECHO=0 SEULEMENT
                          227 *
FCD7 B6E200               228 LCAR   LDA    A,ETAT         LECTURE ETAT
FCDA 44                   229        LSRA   A
FCDB 24FA                 230        BCC    LCAR           BOUCLE ATENTE
FCDD B6E300               231        LDA    A,DONA
                          232 *
                          233 *
FCE0 7DF30B               234 RCA1   TST    AECC
FCE3 260C                 235        BNE    RET3
                          236 *
                          237 *IMPRESSION D'UN CARACTERE
                          238 *
FCE5 B7E300               239 PCAR   STA    A,DONA         ECRITURE CARACT
FCE8 37                   240        PSHB                  SAUVEGARDE
FCE9 F6E200               241 ECR    LDA    B,ETAT
FCEC 54                   242        LSRA   B
FCED 54                   243        LSRA   B
FCEE 24F9                 244        BCC    ECR
FCF0 33                   245        PULB
FCF1 39                   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 *
ECE2 8612           255 PAMA    LDI    A,PEMN
ECE4 BDEE           256         BSR    PCAR
                    257 *
ECE6 C620           259 PAMF    LDI    B,=X20         32 NULS
ECE8 4F             259         CLRA   A
ECE9 BDEA           260 PNUL    BSR    PCAR
ECEB 5A             261         DECB   B
ECEC 26FB           262         BNE    PNUL
ECEE 39             263 RET2    RTS
                    265 *
                    266 *
                    267 *           RESET ACIA
                    268 *
                    269 *
ECEF B603           270 USB7    LDI    A,=X03         RESET ACIA
ED01 B7E200         271         STA    A,ETAI
ED04 8601           272 IAC1    LDI    A,=X01         INIT ACIA SUPPRE
                    273 *
                    274 * 7 BITS + 2 BITS STOP
                    275 *
ED06 B7E200         276 USTP    STA    A,ETAI
ED09 B6E300         277         LDA    A,DONA         LECTURE DUMMY
EDOC 39             278         RTS
                    279 *
                    280 * OFF-READER/PUNCH
                    281 *
ED0D 8613           282 OFPR    LDI    A,REOF         OFF-SOFT
ED0F BDD4           283         BSR    PCAR
ED11 8614           284         LDI    A,PEOF
ED13 2000           285         BRA    PCAR
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCH

CE	CODE		INST	OPERANDES	COMMENTAIRES
		287 *			
		288 *	RETOUR/LANCEMENT		
		289 *			
FD15	BDFC4E	290 CC	JSR	CADR	
FD18	3DB0	291	BSR	RTCH	
FD1A	6E00	292	JMP	ZERO(X)	
FD1C	BDAC	293 RTRA	BSR	RTCH	
FD1E	3B	294 RTI	RTI		RETOUR INTERR
		295 *			

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR ECOLE ARCHITE

```
CE   CODE                       INST   OPERANDES      COMMENTAIRES

                *******************************
FD1F CEE910     CLER    LDXI    TRIT
FD22 86CC       INBT    LDI     A,=XCC         CODE INVALIDE
FD24 A702               STA     A,DEUX(X)
FD26 08                 INCX
FD27 08                 INCX
FD28 08                 INCX
FD29 9CE829             CPXI    FTBT
FD2C 26F4               BNE     INBT
FD2E B7E82A             STA     A,SECU
FD31 39                 RTS
                ***************************************
                *          ENTREE PAR RESTART         *
                ***************************************
FD32 8EE87F     INIT    LDSI    PILE                 INIT    PII
FD35 8DC8               BSR     WSRZ           INIT ACIA
FD37 8DD4               BSR     DEPR
FD39 8DE4               BSR     CLER           INIT TRIT/SECU
                *
                *
                *
FD3B 86CC       NAIS    LDI     A,=XCC         CODE
FD3D B1E82A             CMPA    A,SECU
FD40 2620               BNE     ERIN
FD42 BFE803             STR     S,SPIL
                *
FD45 BEE803     COMM    LDR     S,SPIL         REINIT PILE
FD48 8D8A               BSR     IACI
FD4A 8DC1               BSR     CEPR
FD4C BDFCCA             JSR     RTCH           RETOUR CHARIOT
FD4F 862A               LDI     A,=X2A         "*"
FD51 BDFCF5             JSR     PCAR
FD54 7FE803             CLR     ACCO           LECTURE AVEC EC
FD57 BDFC07             JSR     LCAR              LECTURE D'UN
                *
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR ... IRF ARCH

```
CE   CODE                              INST    OPERANDES       COMMENTAIRES
                       *
                       * ANALYSIS DES COMMANDES
                       *
                       *
FD5A CEFFB3                    LDXI    TPCC            COMMANDES
FD5D A100              XTCM    CMPA    A,ZERO(X)
FD5F 2604                      BNE     CMSV
FD61 EE01                      LDR     X,UN(X)
FD63 6E00                      JMP     ZERO(X)
FD65 08                CMSV    INCX
FD66 08                        INCX
FD67 08                        INCX
FD68 8CFFF3                    CPXI    FTCO            FINAL COMMAND
FD6B 2708                      BEC     CCMM
FD6D 20EF                      BRA     XTCM
                       *
                       *
                       *
FD6F 8EF87E            ERIN    LDSI    PILE
FD72 BFF803                    STR     S,SPIL
FD75 BDFD1F                    JSR     CLER
FD78 BDFCFF                    JSR     MSRZ
FD7B C6E3                      LDB     B,=XE3
                       *
                       * CODE ERREUR DANS A
                       *
FD7D BDFCCA            ERRE    JSR     RTCH
FD80 17                        TBA
FD81 BDFCBA                    JSR     IMDC
FD84 20BF              UPK     BRA     CCMM

                       *            MESSAGES D'ERREUR
                       *           1     ERREUR LECTURE
                       *           2     ERREUR  CAR NON HEXA
                       *           3     DEBORDEMENT PILE
                       *           4      ADRESSE NON TROUVEE / CODE
                       ******************************************
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR FONT ARCHITEC

```
CE  CODE                          INST   OPERANDES      COMMENTAIRES
                        371 ****************************************
                        372 *    POSE DE POINT D'ARRET               *
                        373 ****************************************
FD86 CEFB10             374 BREK   LDX1   TBIT           TABLE DE POINTS
FD89 A602               375 DEB    LDA    A,DEUX(X)      CODE
FD8B 81CC               376        CMPI   A,=XCC         CODE INEXISTANT
FD8D 2701               377        BEQ    CNT
FD8F 08                 378        INCX
FD90 08                 379        INCX
FD91 08                 380        INCX
FD92 9CFB28             381        CPX1   FTBT           FINAL TBIT
FD95 26F2               382        BNE    DEB
FD97 2022               383        BRA    ERR4
                        384 *
                        385 *
FD99 FFFB05             386 CNT    STR    X,SWIT
FD9C BDFC51             387        JSR    CADI
FD9F A600               388        LDA    A,ZERO(X)
FDA1 813F               389        CMPI   A,=X3F         SWI
FDA3 2716               390        BEQ    ERR4
FDA5 FEF905             391        LDR    X,SWIT
FDA8 A702               392        STA    A,DEUX(X)
FDAA FFD7               393        LDR    X,ZERO(X)
FDAC 963F               394        LDI    A,=X3F
FDAE A700               395        STA    A,ZERO(X)
FDB0 A109               396        CMPA   A,ZERO(X)
FDB2 2700               397        BEQ    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 ERR4   LDI    B,=XE4
FDBD 20BE               409        BRA    ERRE
```

```
PROGRAMME : SD-6800          ASSEMBLEUR MICROPROCESSEUR  EQUIPE ARCH

CE  CODE                          INST   OPERANDES      COMMENTAIRE

                          411 *
                          412 * TRAITEMENT DES POINTS D.ARRET
                          413 *
FDBF BEE803               414 PRIT    STR    S,SPIL
FDC2 30                   415         TSX
FDC3 EE05                 416         LDR    X,5(X)         PC
FDC5 09                   417         DECX
FDC6 BEE807               418         STR    X,VAR
                          419 *
                          420 * VAR: ADRESSE D.ARRET
FDC9 CEE810               421         LDX    TBIT
FDCC 8CE828               422 RETM    CPX    ETBT
FDCF 27FA                 423         BEQ    ERR4
FDD1 BEE805               424         STR    X,SWIT
FDD4 E500                 425         LDR    X,ZERO(X)
FDD6 BCE807               426         CMPR   X,VAR          VAR
FDD9 2708                 427         BEQ    SCR
FDDB BEE805               428         LDR    X,SWIT
FDDE 09                   429 SWI1    INCX
FDDF 09                   430         INCX
FDE0 09                   431         INCX
FDE1 20F9                 432         BRA    RETM           RETOUR
                          433 *
                          434 * SWIT: ADRESSE RELATIVE DANS TBIT
                          435 *
FDE3 BEE805               436 SCR     LDR    X,SWIT
FDE6 A602                 437         LDA    A,DEUX(X)      CODE
FDE8 81CC                 438         CMPI   A,=XCC
FDEA 27F2                 439         BEQ    SWI1
FDEC EE00                 440         LDR    X,ZERO(X)
FDEE A700                 441         STA    A,ZERO(X)      REMETTRE CODE
FDF0 BDFEDF               442         JSR    IMAI           IMPRESSION AI
                          443 *
                          444 * POSITIONNER CODE INVALIDE
                          445 *
FDF3 86CC                 446         LDI    A,=XCC
FDF5 BEE805               447         LDR    X,SWIT
FDF8 A702                 448         STA    A,DEUX(X)
FDFA 30                   449         TSX
FDFB 6D06                 450         TST    6(X)
FDFD 2602                 451         BNE    SWI2           PC:=PC-1
FDFF 6A05                 452         DEC    5(X)
FE01 6A06                 453 SWI2    DEC    6(X)
FE03 7EED45               454 FCS     JMP    COMM           RETOUR COMMAN
                          455 *
```

PROGRAMME : [illegible] ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHITE

CE	CODE		INST	OPERANDES	COMMENTAIRES
		458 *			
		459 * GENERATEUR DES SIGNAUX LECTURE/ECRITURE			
		460 *			
FE06	BDEC4E	461 TEST	JSR	CACR	
FE09	BDEC07	462	JSR	LCAR	
FE0C	9152	463	CMPI	A,=X52	"R" LECTURE
FE0E	270A	464	BEQ	READ	
		465 *			
FE10	BDEC2A	466	JSR	DOCT	
FE13	A700	467 OKW	STA	A,ZERO(X)	
FE15	BDEEA4	468	JSR	TCAR	
FE18	20E9	469	BRA	OKW	
FE1A	A600	470 READ	LDA	A,ZERO(X)	
FE1C	BDEEA4	471	JSR	TCAR	
FE1F	20E9	472	BRA	READ	
		473 *			
		474 * RETOUR DE 'TEST' PAR CARACTER TOUCHE			
		475 *			

```
 **  CODE                          INST    OPERANDES       COMMENTAIRE
FE21 7CE809                477 LCAD    INC     AECC            LECTURE SANS
                           478 *
                           479 * ACTIVATION READER
                           480 *
FE24 8611                  481         LDI     A,#RCON         ON READER
FE26 BDEDE5                482         JSR     PCAR
FE29 8641                  483         LDI     A,=X41          RTS:=1
FE2B B7E200                484         STA     A,ETAT
FE2E BDEDD7                485 CHR3    JSR     LCAR            LECTURE CHAR.
FE31 8153                  486         CMPI    A,=X53          "S"
FE33 26F9                  487         BNE     CHR3
FE35 BDEDD7                488         JSR     LCAR
FE38 8139                  489         CMPI    A,=X39          "9"
FE3A 2707                  490         BEQ     CO5
FE3C 8131                  491         CMPI    A,=X31          "1"
FE3E 26EE                  492         BNE     CHR3
FE40 7FE809                493         CLR     CKSM            ZERO CKSM
FE43 BDEC2C                494         JSR     LOCT            LECTURE OCTE
FE46 8002                  495         SUBI    A,=X02
FE48 B7E80A                496         STA     A,#COM
FE4B BDEC4E                497         JSR     CADR            CONSTRUCTION
FE4E BDEC2C                498 LCO1    JSR     LOCT
FE51 7AE80A                499         DEC     #COM
FE54 2724                  500         BEQ     LCO5            COMPTEUR OCT
FE56 A700                  501         STA     A,ZERO(X)
                           502 *
                           503 * VERIFICATION ENREGISTREMENT
                           504 *
FE58 A100                  505         CMPA    A,ZERO(X)
FE5A 2718                  506         BEQ     CONN            CONTINUATION
FE5C 7AE809                507         DEC     AECC            ERREUR CHARG
FE5F BDECCA                508         JSR     RTCH
FE62 8078                  509         BSR     IMAC
                           510 *
                           511 * TRAITEMENT ERREUR
                           512 *
FE64 863F                  513 ERDR    LDI     A,=X3F          "?"
FE66 BDEDE5                514         JSR     PCAR
FE69 BDED04                515 CH21    JSR     IACI
FE6C BDED0D                516         JSR     OEPR
FE6F 7CE809                517         INC     AECC
FE72 BDEDD7                518         JSR     LCAR            LECTURE DUMM
FE75 209C                  519 CC9     BRA     CO5
                           520 *
                           521 *
                           522 *
FE77 09                    523 CONN    INCX
FE78 2004                  524         BRA     LCO1
FE7A 7CE809                525 LCO5    INC     CKSM
FE7D 27AF                  526         BEQ     CHR3
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHIT

```
CE  CODE                          INST   OPERANDES      COMMENTAIRES
                    530 *
                    531 *
                    532 * E <ADR1       <ADR2
                    533 *
                    534 * TEST MEMOIRE-PREMIERE TEST-LECT/ECRITURE
                    535 *
FE7E BDEE5C         536 TEB6    JSR     ACRC           CONTRUCTION.VE
FE82 BDECCA         537         JSR     RTCH
FE85 8611           538         LDI     A,=X11         INIT COMPTEUR
FE87 B7E800         539 LE83    STA     A,PC1
                    540 *
                    541 * PREMIERE TEST
                    542 *
FE8A FEE80C         543         LDX     X,REGA
FE8D A700           544 UP1     STA     A,?ERR(X)
FE8F E600           545         LDA     B,?ERC(X)
FE91 11             546         CBA
FE92 2702           547         BEQ     SUIT
FE94 8D20           548         BSR     ERRR           ERREUR
                    549 *
                    550 *
FE96 BCE80E         551 SUIT    CMPX    X,FNDA
FE99 2719           552         BEQ     DCW2           TRAITEMENT ADR
FE9B 08             553         INCX
FE9C 8D11           554         ADCI    A,=X11
FE9E 2403           555         BCC     UP1
FEA0 4C             556         INCA    A
FEA1 20EA           557         BRA     UP1
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR POUR ARCHIT

CODE

			INST OPERANDES	COMMENTAIRES
		559 *		
		560 * MULTIADRESSAGE		
		561 *		
FEA3	BDFCCA	562 DCW3	JSR RTCH	
FEA6	BDFCC6	563	JSR EQUL	"="
FEA9	96E800	564	LDA A,PC1	
FEAC	FEE30C	565	LDX X,REGA	
FEAF	E600	566 CMPC	LDA B,ZERO(X)	
FEB1	11	567	CBA	
FEB2	2702	568	BEQ SWWU	
FEB4	8D0D	569	BSR ERRR	ERREUR
FEB6	8CE80E	570 SWWI	CMPX X,ENDA	
FEB9	2704	571	BEQ COO	
FEBB	08	572	INCX	
FEBC	A211	573	ADDI A,=X11	
FEBE	24EF	574	BCC CMPO	
FEC0	4C	575	INCA A	
FEC1	20EC	576	BRA CMPC	
		577 *		
		578 *		

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARCHI

CC	CODE			INST	OPERANDES	COMMENTAIRES
		580	*			
		581	*			
		582	*			
		583	*	TRAITEMENT ERREUR		
		584	*			
FEC3	36	585	ERRR	PSHA		
FEC4	BDFECA	586		JSR	PTCH	
FEC7	3D13	587		DSR	INAD	
FEC9	BDFE5B	588		JSR	SPAC	
FECC	32	589		PULA		
FECD	36	590		PSHA		
FECE	BDFECBA	591		JSR	INDC	
FED1	BDFE5B	592		JSR	SPAC	
FED4	BDFEA4	593		JSR	TCAR	
FED7	BDFCB9	594		JSR	OUCT	
FEDA	32	595		PULA		
FEDB	39	596		RTS		
		597	*			
		598	*			
FEDC	FEF807	599	INAD	STR	X,VAR	
FEDF	B6F807	600	INAI	LDA	A,VAR	
FEE2	BDFECBA	601		JSR	INDC	
FEE5	B6F809	602		LDA	A,VAR+1	
FEE8	7EFECBA	603		JMP	INDC	
		604	*			

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR EQUIPE ARC

```
CF    CODE                             INST    OPERANDES        COMMENTAIRE

                       *
                       * PUNCH <ADR1  <ADR2
                       *
FEEB 8D6F              PUNC     BSR     ADRC             DOUBLE ADRE
FEED BDFCF2                     JSR     PNON             PERFO ON ET
FEF0 8D6C              PUN1     BSR     TSAD             ENDA-BEGA
FEF2 2604                       BNE     PUN2
FEF4 8110                       CMPI    A,=X10
FEF6 2502                       BCS     PUN3
FEF8 860F              PUN2     LDI     A,=X0F
FEFA 4C                PUN3     INCA    A
FEFB B7F800                     STA     A,PC1
FEFE 8B03                       ADDI    A,=X03
FF00 B7F807                     STA     A,VAR
FF03 BDFCCA                     JSR     BTCH
FF06 BDFEA4                     JSR     TCAR             TEST CARACTE
FF09 CEFFF3                     LDXI    MTIP
FF0C 8D3A                       BSR     PDAT
FF0E 7FF80?                     CLR     CKSM             CLEAR CHECK
FF11 CEF807                     LDXI    VAR
FF14 8D39                       BSR     PUNT
                       *
                       *
                       * IMPRESION(PUNCH) ADRESSE
                       *
FF16 CEF80C                     LDXI    BEGA
FF19 8D34                       BSR     PUNT
FF1B 8D32                       BSR     PUNT
                       *
                       * PUNCH DATA
                       *
FF1D FEF80C                     LDR     X,BEGA
FF20 8D2D              PU32     BSR     PUNT
FF22 7AF800                     DEC     PC1
FF25 26F9                       BNE     PU32
FF27 FFF80C                     STR     X,BEGA
FF2A 53                         COMA    B
FF2B 37                         PSHB
FF2C 30                         TSX
FF2D 8D20                       BSR     PUNT
FF2F 33                         PULB
FF30 FEF80C                     LDR     X,BEGA
FF33 0[illegible]               DECX
FF34 BCF80[illegible]           CMPR    X,ENDA
FF37 26[illegible]              BNE     PUN1
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR [illegible] ARCH

```
CE CODE                                  INST   OPERANDES      COMMENTAIRE

                                  653 *
                                  654 * FIN DE RUBAN
                                  655 *
FF39 CEFFF6                       656         LDX1   $9
FF3C BD0A                         657         BSR    PDAT
FF3E BDFCF6                       658         JSR    PNOF       PUNCH-OFF
                                  659 *
FF41 7EE045                       660 ARK     JMP    CCMM       RETOUR SUPER
                                  661 *
                                  662 *
                                  663 * SORTIE DES MESSAGES
                                  664 *
FF44 BDFCF5                       665 PCA2    JSR    PCAR
FF47 08                           666         INCX
FF48 A600                         667 PDAT    LDA    A,ZERO(X)
FF4A 8104                         668         CMPI   A,=X04
FF4C 26F6                         669         BNE    PCA2
FF4E 39                           670         RTS
                                  671 *
                                  672 *
                                  673 * PUNCH CARACTERE ET CALCUL CHECK-SUM
                                  674 *
                                  675 *
FF4F BDFC88                       676 PINT    JSR    OCCT       IMPRESION OCTE
FF52 F67809                       677 CKS1    LDB    B,CKSM
FF55 FB00                         678         ADDB   B,ZERO(X)
FF57 F77809                       679         STB    B,CKSM
FF5A 08                           680         INCX
FF5B 39                           681         RTS
                                  682 *
FF5C 810F                         683 ADRT    BSR    ACR4       CHEKING ADRES
FF5E B47B0F                       684 TSAD    LDA    A,ENDA+1
FF61 F67B0D                       685         LDA    B,ENDA
FF64 B07B09                       686         SUBB   A,BEGA+1
FF67 F27B0C                       687         SBCA   B,BEGA
FF6A 25D5                         688         BCS    ARK        RETOUR
FF6C 39                           689         RTS
                                  690 *
                                  691 *
                                  692 *
FF6D CE7B0C                       693 ACR4    LDX1   BEGA       SANS CHEKING AI
FF70 BDFC5E                       694         JSR    CAD1
FF73 CE7B0E                       695         LDX1   ENDA
FF76 BDFC51                       696         JSR    CAD1
FF79 39                           697         RTS
                                  698 *
                                  699 *
                                  700 *
```

PROGRAMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR [illegible] APC

	CE CODE		INST	OPERANDES	COMMENTAIR
701		*			
702		* EFFACEMENT POINTS D ARRETS			
703		*			
704		CKEI	LDX	TBTT	
705		BRCA	LDA	A,CEUX(X)	CODE
706			CMP	A,=XCC	
707			BNE	RCCC	
708		INCT	INCX		
709			INCX		
710			INCX		
711			CPX	FTBT	
712			BNE	PRON	
713			JSR	CLER	
714		BKBK	BRA	PRK	RETOUR
715		RCOD	STR	X,VAR	SAUVEGARDE
716			LDR	X,ZERO(X)	
717			STA	A,ZERO(X)	
718			LDR	X,VAR	INDEX...VAR
719			BRA	INCT	
720		*			
721		*			

FF7A	CEF810
FF7D	1622
FF7F	81CC
FF81	26[illegible]
FF83	08
FF84	08
FF85	08
FF86	[illegible]828
FF89	26F2
FF8B	BDFD1F
FF8E	2081
FF90	FEF807
FF93	FF00
FF95	A700
FF97	FEF807
FF9A	20F7

[illegible]AMME : SP-6800 ASSEMBLEUR MICROPROCESSEUR [illegible] ARCHIT

[illegible] CODE

			INST	OPERANDES	COMMENTAIRES
		723 *			
		724 *			
		725 * VECTEUR D'INTERRUPTION(F828)			
		726 *			
		727 *			
F80C	FEF807	728 IRQ	STR	X,VAR	SAUVEGARDE IND
F80F	FEF929	729	LDR	X,VIRQ	
F812	6E00	730	JMP	ZERO(X)	INDIRECTION IR
		731 *			
F814	B6F200	732 TCAR	LDA	A,ETAT	
F817	44	733	LSRA	A	
F818	2501	734	BCS	RTSW	
F81A	39	735	RTS		
F81B	B6F300	736 RTSW	LDA	A,DONA	
F81E	20FE	737	BRA	BKBK	RETOUR COMMANDE
		738 *			
		739 *			
F820	3E	740 WAIT	WAI		WAIT
F821	20FD	741	BRA	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	DC2	MEM	"M"
FFB6	47	756	DC1	=X47	
FFB7	FD15	757	DC2	GO	LANCEMENT DE
FFB9	52	758	DC1	=X52	"R"
FFBA	FD1C	759	DC2	RTRN	RETOUR INTERR
FFBC	41	760	DC1	=X41	"A"
FFBD	FC88	761	DC2	ACCA	ACCU A
FFBF	42	762	DC1	=X42	
FFC0	FC89	763	DC2	ACCB	ACCU B
FFC2	58	764	DC1	=X58	"X"
FFC3	FC97	765	DC2	REGX	INDEX
FFC5	43	766	DC1	=X43	"C"
FFC6	FC9A	767	DC2	REGP	PROGRAMME CO
FFC8	53	768	DC1	=X53	"S"
FFC9	FC92	769	DC2	REGS	PILE
FFCB	4B	770	DC1	=X4B	"K"
FFCC	FD86	771	DC2	BREK	POSE DE POINT
FFCE	54	772	DC1	=X54	"T"
FFCF	FE06	773	DC2	TEST	TEST SIGNAUX
FFD1	4C	774	DC1	=X4C	"L"
FFD2	FE21	775	DC2	LOAD	CHARGEMENT DE
FFD4	45	776	DC1	=X45	"E"
FFD5	FE7F	777	DC2	TERW	ESSAI MEMOIRI
FFD7	50	778	DC1	=X50	"P"
FFD8	FEFB	779	DC2	PUNC	PUNCH MEMOIRI
FFDA	46	780	DC1	=X46	"F"
FFDB	FE7A	781	DC2	CKF1	EFFACEMENT PI
FFDD	5A	782	DC1	=X5A	"Z"
FFDE	FC85	783	DC2	CC	CODE CONDITI
FFE0	57	784	DC1	=X57	"W"
FFE1	FFB0	785	DC2	WAIT	WAIT
FFE3	FFE3	786 FTCD	EQU	*+0	FINAL TBCD
		787 *			

CODE		INST	OPERANDES	COMMENTAIRES
	789 *			
FFE3 53	790 WTIP	DC1	=X53	"S"
FFE4 31	791	DC1	=X31	"1"
FFE5 04	792	DC1	=X04	
	793 *			
	794 *			
FFE6 53	795 SS	DC1	=X53	"S"
FFE7 30	796	DC1	=X30	"0"
FFE8 04	797	DC1	=X04	
	798 *			
	799 *			
FFE9 00	800 RCLF	DC1	=X0C	
FFEA 0A	801	DC1	=X0A	
FFEB 00	802	DC1	=X00	
FFEC 00	803	DC1	=X00	
FFED 00	804	DC1	=X00	
FFEE 04	805	DC1	=X04	
	806 **			
	807 *			
	808 *			
	809 *			
	810 **********			
	811 *	VECTEUR MACHINE		*
	812 **********			
FFF8 FFE3	813	ORG	=XFFF8	
FFF8 FF0C	814 IRQS	DC2	IRQ	
FFFA FD9E	815 SWI	DC2	PRIT	
FFFC FD3B	816 NMI	DC2	NMIS	
FFFE FD32	817 RST	DC2	INIT	
0000 00	818	END		

dernière page de la thèse

AUTORISATION DE SOUTENANCE

VU les dispositions de l'article 3 de l'arrêté du 16 Avril 1974,

VU les rapports de présentation de Messieurs :

- 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