Etude et réalisation d'une unité de contrôle banalisée pour systèmes IBM 360/370

Serge Arnaud

THESE

présentée à

Institut National Polytechnique de Grenoble

pour obtenir le grade de

DOCTEUR INGENIEUR
(Génie Informatique)

par

Serge ARNAUD

ETUDE ET REALISATION D'UNE UNITE DE CONTROLE BANALISEE POUR SYSTEMES IBM 360/370.

Thèse soutenue le 15 octobre 1979 devant la Commission d'Examen :

Président : Monsieur Louis BOLLIET

Examinateurs : Messieurs François ANCEAU
Gérard NOGUEZ
Raymond BOUTTAZ
Alain GUYOT

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

à ma mère,
à Pierrette,
à Jérôme et
Pierre-Emmanuel

Je suis heureux de remercier

Messieurs les membres du Jury :

Monsieur Louis BOLLIET, Professeur à l'Université de Grenoble, qui a bien voulu me faire l'honneur de présider le jury de cette thèse,

Monsieur Gérard NOGUEZ, Professeur à l'Institut de Programmation de Paris VI, qui a accepté de juger ce travail,

Monsieur François ANCEAU, Maître de Conférences à l'ENSIMAG, qui m'a donné les moyens matériels et procuré les encouragements nécessaires au bon déroulement de cette réalisation,

Monsieur Raymond BOUTTAZ, Ingénieur au LA7 du CNRS, qui a bien voulu participer à ce jury et qui m'a toujours fourni l'aide matérielle et logistique dont j'ai eu besoin,

Monsieur Alain GUYOT, Assistant à l'ENSIMAG, qui a guidé mes premiers pas en Informatique et m'a toujours apporté de précieux conseils,

Ainsi que :

Monsieur Roberto SAETTONE qui a participé très activement à la réalisation de l'UCB et qui m'a largement fait bénéficier de sa grande compétence en micro-informatique,

Messieurs Gérard BAILLE, Jacques LAURENT et Jean-Pierre SCHOELLKOPF, pour le système MADAM, grâce auquel la mise au point de l'UCB s'est trouvée grandement facilitée,

Messieurs Yves BEKKERS et Pierre FONTANILLE, dont le générateur d'assembleurs GAGE m'a permis d'écrire très facilement un assembleur croisé pour le 8X300,

Monsieur René GARCIA et les opérateurs du CICG qui m'ont gentiment laissé jouer avec leur 360/67 pendant ses heures de repos dominical pourtant bien méritées,

Monsieur Abdelkader BOUGHLAM, pour son système de traitement de textes REDAK, à l'aide duquel j'ai pu effectuer l'élaboration matérielle de cet ouvrage, avant l'intervention efficace du service de reprographie de l'IMAG,

... et tous les collègues de l'équipe de Recherche en Architecture des Ordinateurs, de l'Atelier de Micro-Informatique, du CICG, de l'IMAG, qui m'ont apporté une aide ponctuelle ou un soutien moral.

ETUDE ET REALISATION

d'une

UNITE

de

CONTROLE

BANALISEE

pour

SYSTEMES IBM 360/370

TABLE DES MATIERES

page

page

page

page

page

page

PRINCIPALES ABREVIATIONS UTILISEES

BI	Bus-In
BO	Bus-Out
BCU	Bus Control Unit
BSC	Binary Synchronous Communication
CAW	Channel Address Word
CC	Code condition
CCW	Channel Command Word
CISS	Channel Initiated Selection Sequence
CPU	Central Processing Unit
CSW	Channel Status Word
CTC	Channel To Channel Adapter
CUBS	Control Unit Busy Sequence
CUIS	Control Unit Initiated Sequence
DAT	Dynamic Address Translation
LSI	Large Scale Integration
MCC	Module Contrôleur de Communication
MIC	Module Interface Canal
MST	Module Station de Transport
NOP	Non Operation
PIAR	Périphérique Intelligent d'Accès au Réseau
PROM	Programmable Read-Only Memory
PSW	Program Status Word

RAM Random Access Memory
ROM Read-Only Memory

SIO Serial Input/Output
ST Station de Transport

TI Tag-in
- AI Address-in
- MTI Metering-in
- OI Operational-in
- RI Request-in
- SI Service-in
- SLI Select-in
- STI Status-in

TO Tag-Out
- AO Address-out
- CLO Clock-out
- CO Command-out
- HO Hold-out
- OO Operational-out
- SLO Select-out
- SLOE Select-out "entrant"
- SLOS Select-out "sortant"
- SO Service-out
- SPO Suppress-out

UCB Unité de Contrôle Banalisée
UCW Unit Control Word
USRT Universal Synchronous Receiver/Transmitter

VLSI Very Large Scale Integration

PRESENTATION

PRESENTATION

Ce travail a eu pour origine certains problèmes posés par l'implantation d'une Station de Transport du Réseau CYCLADES sur l'ordinateur IBM 360/67 du Centre Interuniversitaire de Calcul de Grenoble, permettant d'offrir aux utilisateurs du réseau les ressources de cet ordinateur, en particulier le système CP67/CMS.

Il avait pour but l'étude et la réalisation d'un coupleur d'entrées/sorties connecté au bus du canal multiplexeur du 360. Ce coupleur devait satisfaire en premier lieu les spécifications de l'Interface d'E/S du système, et donc se présenter comme un contrôleur de périphérique. Mais d'autre part il devait permettre de résoudre le problème initialement posé et pouvait alors revêtir 2 aspects :

- soit celui d'un adaptateur canal-à-canal bouclé, permettant la communication entre 2 machines virtuelles travaillant sous CP-67.
- soit celui de la partie Interface-Canal d'un frontal supportant la Station de Transport.

Dans les 2 cas, le coupleur s'intégrait dans une réalisation autour d'un ou plusieurs microprocesseurs, pour des problèmes à la fois de coût d'investissement, de souplesse d'utilisation, de facilité de mise au point, et afin de profiter de l'expérience déjà acquise dans le domaine de la micro-informatique par l'équipe de Recherche en d'Architecture des Ordinateurs de l'IMAG.

La réalisation d'une première version du coupleur correspondant à la première solution a surtout permis :

- une familiarisation avec le comportement du canal IBM
- l'observation (indirecte) des séquences à respecter
- la mise au point, sur le 360, d'un programme de dialogue entre le coupleur et le canal.

Cette version utilisait un microprocesseur N-MOS standard : le MC 6800 de Motorola, peu rapide, et a conduit à réaliser une deuxième version à l'architecture plus spécialisée autour d'un microprocesseur bipolaire : le 8X300 de Signetics-RTC, beaucoup plus rapide.

Parallèlement, l'évolution des travaux de R. Saettone sur un système à microprocesseurs multiples destiné plutôt à satisfaire la deuxième solution par la définition d'une Station de Transport Externe, nous a amenés à concevoir en commun cette deuxième version du coupleur comme le troisième élément de cette structure multi-microprocesseurs.

Nous exposerons d'abord (Chap.I) les différentes façons d'envisager les entrées/sorties dans les ordinateurs en s'attachant plus spécialement à la structure des systèmes à Unité d'Echange et en rappelant les solutions utilisées dans les microprocesseurs.

Dans un deuxième chapitre, nous décrirons l'organisation du système d'E/S d'un ordinateur de la gamme IBM 360, et surtout le fonctionnement du canal multiplexeur des modèles 65-67 afin d'en extraire les éléments nécessaires à la réalisation d'un coupleur devant être vu comme un contrôleur "classique".

Le troisième chapitre nous permettra de définir une Unité de Contrôle Banalisée (UCB) pour les systèmes IBM 360/370. Nous exposerons les différents aspects de son architecture interne et de sa programmation, en faisant apparaître les spécificités de l'une et de l'autre, dues en partie au micro-processeur utilisé. Nous donnerons les résultats obtenus sous forme de séquences des signaux d'interface ainsi que les débits mesurés.

Nous rappellerons brièvement, dans le chapitre IV, le rôle d'une Station de Transport dans le réseau CYCLADES et les problèmes posés par son implémentation sur un ordinateur hôte ayant ses propres caractéristiques et son (ou ses) propre système d'exploitation. Nous exposerns également les travaux de R. Saettone ayant conduit à la réalisation du système PIAR, ce qui nous permettra de présenter une réalisation opérationnelle de l'UCB définie précédemment.

Afin de mettre au point et de tester le comportement de cette Unité de Contrôle vis-à-vis d'un certain nombre de situations, nous avons conçu un ensemble de programmes chacun spécifique d'un type d'entrée/sortie et décrits dans le chapitre V. Ces programmes seront à la base d'une sorte de méthode d'accès simplifiée réalisée en langage de haut niveau et permettant une premiére vérification du fonctionnement de toute application utilisant l'UCB.

Enfin nous essaierons au cours du chapitre VI de présenter d'autres utilisations possibles d'une telle réalisation. En particulier nous verrons comment apporter au possesseur d'un ordinateur des gammes IBM 360/370 une solution peu coûteuse au problème de la connexion de périphériques intelligents, problème rendu très actuel par la progression des circuits LSI.

Dans les annexes nous présenterons :

-1- des compléments sur le canal multiplexeur du 360

-2- des compléments sur le microprocesseur 8X300

-3- les schémas complets matériels et logiciels de l'UCB et de son application au système PIAR

-4- la première version du coupleur (canal à canal)

-5- l'assembleur réalisé pour le 8X300 à partir du Générateur d'Assembleur de Grenoble (GAGE)

-6- l'utilisation du système MADAM (Matériel d'Aide au Développement d'Applications à Microprocesseurs) dans la mise au point de l'UCB.

Il est ici nécessaire d'insister sur l'aide considérable apportée par ces 2 derniers outils dans le cadre assez particulier dans lequel se sont déroulés ces travaux, et surtout dans la mesure où ils sont le fruit des recherches de collègues de l'équipe d'Architecture des Ordinateurs et de l'Atelier de Micro-informatique (CNRS-IRIA-IMAG).

CHAPITRE I

L'ORGANISATION DES SYSTEMES D'ENTREES-SORTIES

I-1 INTRODUCTION

I-2 PRINCIPES GENERAUX ET EVOLUTION DU CONTROLE DES E/S

I-2-1 Principes généraux

I-2-2 L'évolution du contrôle des E/S

I-2-3 Exécution d'une entrée-sortie

I-3 LES SYSTEMES D'E/S AVEC UNITE D'ECHANGE

I-3-1 Généralités

I-3-2 Modes de fonctionnement

I-3-3 Réalisation d'une unité d'échange

I-3-4 Gestion des organes d'E/S

- I-3-4-1 Au niveau de l'organe d'E/S
- I-3-4-2 Au niveau de l'unité d'échange
- I-3-4-3 Au niveau de l'unité centrale
- I-3-4-4 Au niveau de la mémoire centrale

I-1 INTRODUCTION

Si d'une façon générale, on peut distinguer 2 grands types d'entrées-sorties : celles qui font communiquer les mémoires secondaires avec la mémoire centrale, et celles qui correspondent au dialogue avec le milieu extérieur, on peut également considérer que le terme d'entrée-sortie a plusieurs significations selon la position de l'utilisateur du matériel informatique.

Pour le programmeur. il s'agira essentiellement de l'entrée de ses programmes et de ses données, et de la sortie de ses résultats; son souci primordial étant la conservation de l'intégralité de ses informations pendant et entre chacun de ses travaux, quels que puissent être les éventuels transferts entre. par exemple, les différents niveaux de mémoire Cependant son comportement devra être différent suivant le langage utilisé et son niveau par rapport au langage machine

A ce moment apparaissent déjà les premiers problèmes à résoudre :

- grande diversité des formats des données
- grande variété des supports d'information
- grande gamme de vitesses de transfert de ces supports

Pour le concepteur de Système d'Exploitation il sera question, en plus des notions propres au programmeur, de transferts entre les différents niveaux de mémoire avec tous les problèmes qui s'y rattachent: utilisation optimale de l'Unité Centrale, interactions entre les différents utilisateurs, problèmes de priorités d'accès à la mémoire centrale, de protection de zones mémoire, etc... Ces problèmes seront multipliés si le système permet des travaux en temps partagé.

Pour l'utilisateur responsable du contrôle d'un processus en temps réel, les entrées-sorties se ramènent à un échange permanent de paramètres à mesurer et de commandes à générer, avec comme critères essentiels le temps de réponse et la fiabilité

Pour le concepteur de l'architecture d'une machine informatique, les entrées-sorties seront plutôt synonymes de :

- organes mis en jeu (technologie, coût, consommation)
- contrôle de ces organes et de leurs liaisons avec l'unité centrale, la mémoire l'extérieur...
- séquencement le long des chemins de données
- interfaces à envisager, aussi bien matériels que logiciels.

Grâce à ces quelques exemples très schématiques on voit apparaître la double nécessité :

- rendre transparents au maximum à certains utilisateurs les mécanismes de base des entrées-sorties, en ne leur laissant employer que des fonctions système des mots clés d'un langage évolué ou - au pire - quelques instructions d'ailleurs privilégiées...

- faire connaître au contraire le déroulement intime de ces mécanismes et leurs liens avec les fonctions d'ordre supérieur, à d'autres utilisateurs chargés de réaliser des couplages et des interfaces entre organes d'entrées-sorties et ordinateur ou entre ordinateurs.

Nous nous placerons donc plutôt dans le second cas et tout d'abord nous nous limiterons à la notion d'entrée-sortie élémentaire que nous définirons comme étant un transfert d'information entre le milieu extérieur et la mémoire centrale.

I-2 PRINCIPES GENERAUX ET EVOLUTION DU CONTROLE DES E/S

I-2-1 Principes généraux

Les problèmes à résoudre peuvent être regroupés en 2 types :

1- par rapport à notre définition du terme d'entrée-sortie élémentaire, nous aurons des problèmes d'interfaçage entre la mémoire centrale et les organes d'E/S, eux-mêmes dus par exemple

- aux différents codages et formats des informations dans les 2 éléments
- aux modes d'opération des 2 systèmes
- aux vitesses de tranfert essentiellement différentes
- aux technologies utilisées et à la circuiterie en général.

Les solutions seront soit d'ordre matériel, soit d'ordre logiciel, en fonction du problème et de l'organe d'E/S (ex: utilisation de tampons pour accommoder les vitesses et les formats : assemblage/découpage des mots mémoire). Une de ces solutions, de portée générale, sera la création d'un interface d'entrée-sortie normalisé (aussi bien physiquement que logiquement) permettant à tous les types d'organes d'E/S d'être connectés au système central ; elle sera mise en oeuvre en même temps que la notion d'Unité d'Echange que nous développerons plus loin.

2- par rapport à l'unité centrale qui sera peu ou très concernée par les transferts selon l'organisation choisie, elle-même tributaire de l'environnement technologique, technique, logique de l'époque (fig. I-1). Dans ce cas, le problème à résoudre est celui de la concurrence entre l'unité centrale et les organes d'E/S. Le but à atteindre étant une efficacité globale maximale, nécessite que les entrées-sorties ne gênent pas le bon déroulement du traitement (donc les transferts entre l'unité et la mémoire centrales) mais aussi qu'elles aient lieu

au maximum des possibilités des organes d'E/S. Ceci, compte tenu des ordres de grandeur des débits de ces 3 organes fera qu'au niveau du couple unité centrale-mémoire centrale, une très faible portion de temps sera consacrée aux entrées-sorties (fig. I-2).

I-2-2 Evolution du contrôle des Entrées-Sorties

Il s'agit seulement ici de présenter brièvement les organisations générales habituelles des entrées-sorties au sein des systèmes informatiques : <CHU>.<MEN>,<PRO>,<BER>,<ANC2>.

Si on s'en tient à notre définition préliminaire, on peut envisager l'organisation idéale simplifiée de la fig.I-1-a. Dans ce cas chaque organe d'E/S peut accéder directement à la mémoire centrale (MC). Il reste entendu que les transferts sont lancés par l'unité centrale (CPU).

En réalité, la majorité des solutions retenues font intervenir l'unité centrale de façon essentielle suivant l'organisation de la fig.I-1-b Il en découle les différentes possibilités de transferts programmés classiques:

- soit en mode bloqué, où l'unité centale est occupée pendant tout le transfert

- soit en mode synchrone dans lequel l'unité centrale détecte la fin du transfert par test d'indicateur

- soit en mode asynchrone où l'organe d'E/S prévient l'unité centrale par interruption permettant à celle-ci d'effectuer d'autres traitements en parallèle.

Ces solutions ont été mises en oeuvre dans les premiers systèmes informatiques et on les retrouve dans la plupart des microprocesseurs actuels.

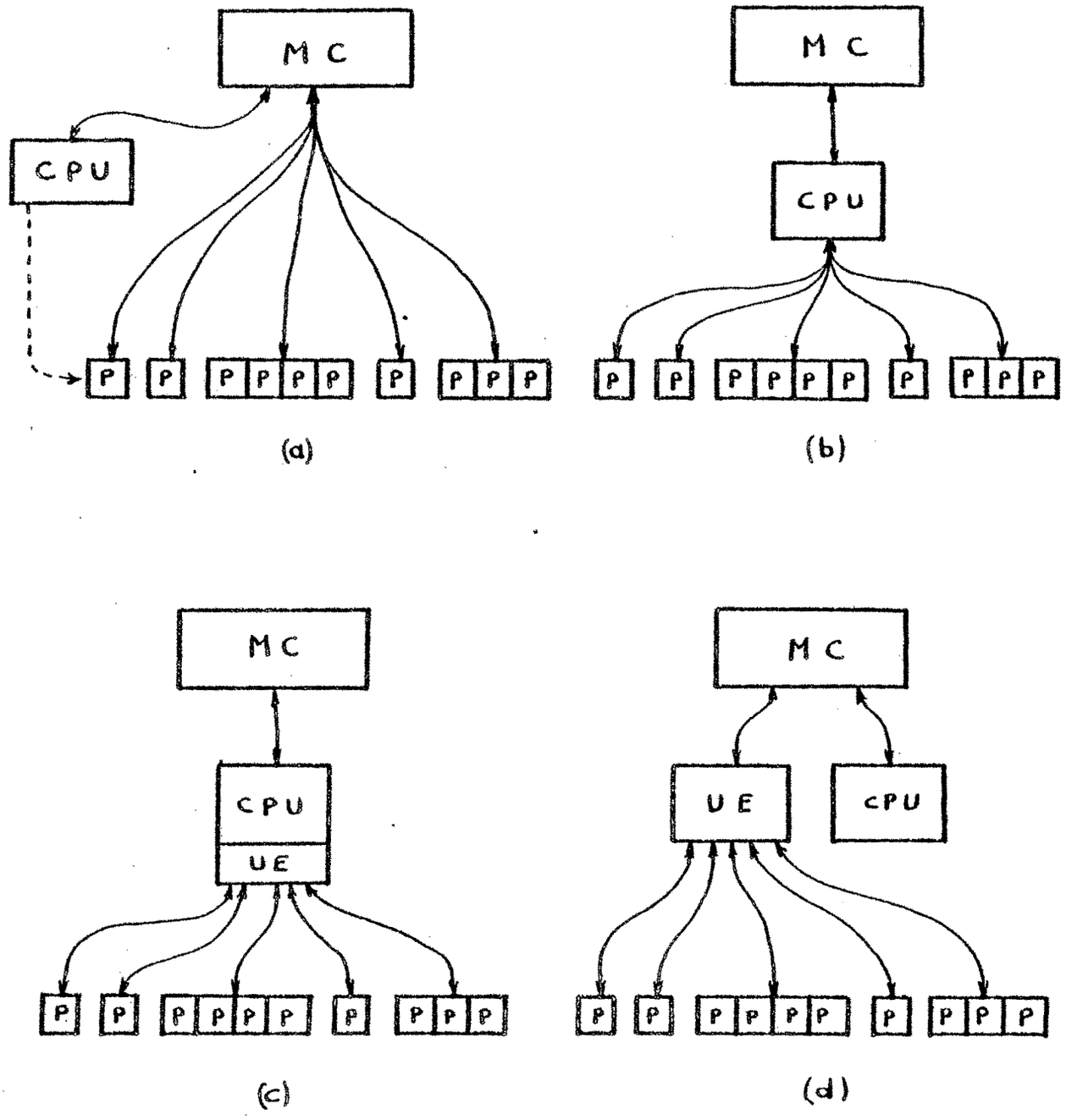


Fig.I-1 Quelques organisations de sytèmes d'E/S

Cependant l'accroissement des besoins de simultanéité entre le traitement proprement dit et les entrées-sorties a conduit à la réalisation de contrôleurs d'E/S ou unité d'échange (UE) et au concept de canal (série IBM 709 puis 7090-7094). On obtient alors les organisations des fig.I-1-c et I-1-d, suivant que le contrôleur d'E/S est intégré ou non à l'unité centrale.

Dans ces 2 cas on fera appel aux méthodes suivantes :

- transfert par instructions forcées
- transfert par vol de cycle
- transfert par accès direct à la mémoire (DMA).

Le mécanisme qui gère l'accès à la mémoire étant intégré à l'unité centrale (vol de cycle) ou à la mémoire elle-même (DMA). Dans ces 2 derniers cas il n'y a pas forcément suspension du programme, mais éventuellement ralentissement de son exécution.

Pour résumer, nous pouvons extraire de ce qui précède, 3 types d'organisation :

- Contrôle direct par l'unité centrale

- Entrées-sorties simultanées : l'unité centrale n'intervient qu'à l'initialisation, à la fin, et lors de conditions anormales survenues pendant l'échange

- Entrées-sorties indépendantes : l'unité d'échange gère tout le transfert après initialisation par l'unité centrale, et lui transmet une interruption et/ou un état de fin.

Ces fonctionnements correspondent aux schémas temporels de la fig.I-2 où l'on peut faire apparaître un "contrôleur de périphériques" ou "unité de liaison" (UL), dans le but déjà cité d'obtenir un interface d'E/S normalisé (fig.I-3).

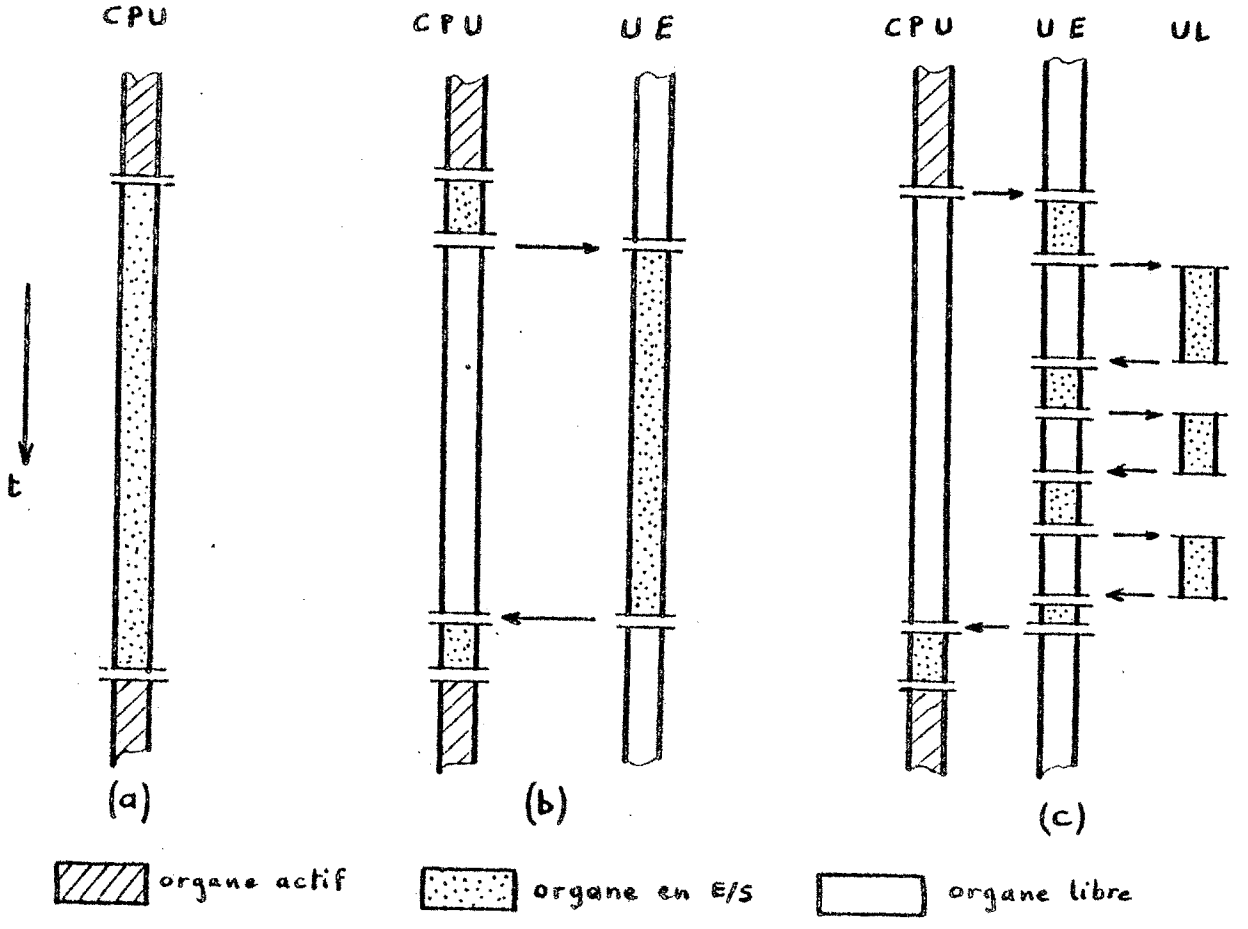


Fig.I-2 Diagrammes temporels des différents contrôles d'E/S

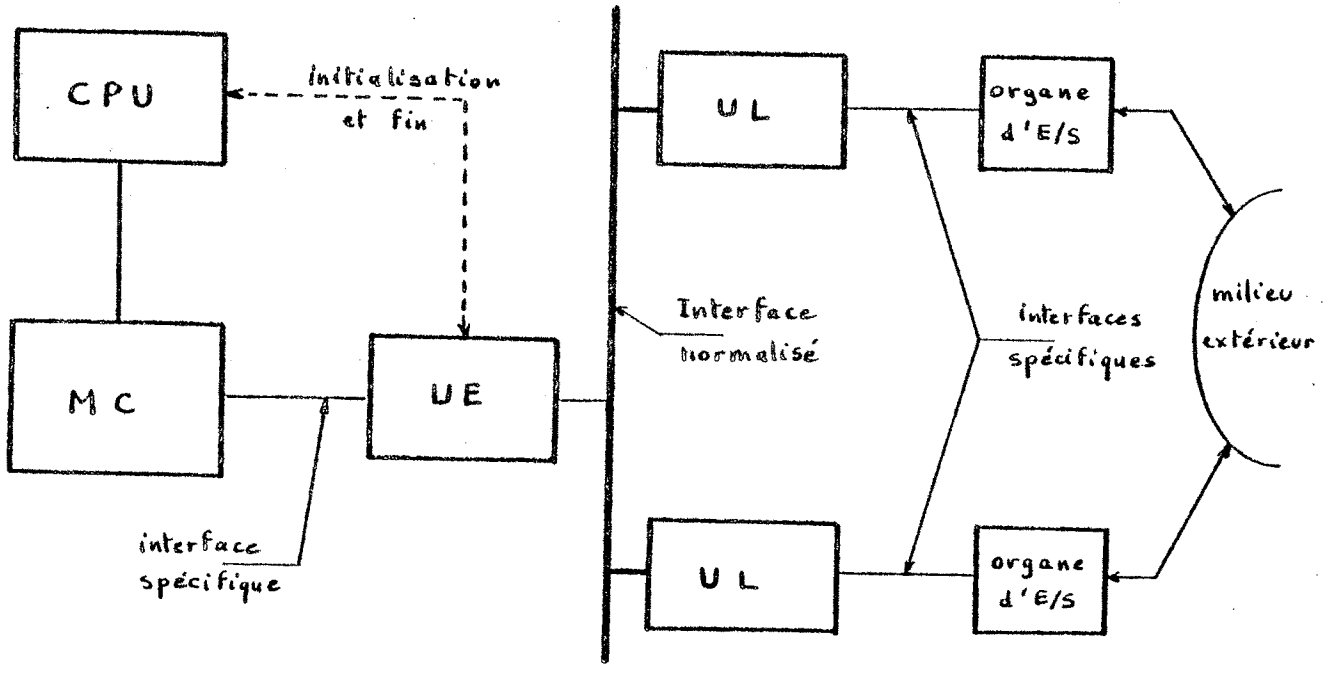


Fig.I-3 Mise en évidence d'un interface normalisé

I-2-3 Exécution d'une entrée-sortie

Quel que soit le type du contrôle des entrées-sorties, il sera nécessaire d'exécuter les opérations suivantes :

- désignation et sélection de l'organe d'E/S concerné

- interrogation de l'état de cet organe

- envoi de la commande (n'impliquant pas forcément un transfert effectif de données) indiquant le sens du transfert ou l'action que doit exécuter l'organe d'E/S

- définition des zones de mémoire concernées par l'échange aussi bien dans la mémoire centrale que dans l'organe d'E/S le cas échéant

- transfert proprement dit

- terminaison de l'entrée-sortie : obtention de l'état précisant le déroulement des opérations, et avertissement à l'unité centrale.

Nous verrons, au chapitre II, sur l'exemple concret de l'interface d'E/S du système IBM 360, comment se déroulent, au niveau le plus fin, ces différentes séquences.

I-3 LES SYSTEMES D'E/S AVEC UNITE D'ECHANGE

I-3-1 Généralités

Une unité d'échange est un organe capable d'effectuer des accès directs à la mémoire centrale et de gérer de façon asynchrone l'ensemble des entrées-sorties d'un ordinateur.

Nous aurons donc l'organisation simplifiée de la fig.I-3.

L'initialisation de l'unité d'échange est faite par l'unité centrale par l'intermédiaire de l'adresse d'un bloc de contrôle de l'échange situé en mémoire centrale : le programme canal. Ce programme contient toutes les informations nécessaires au transfert, et l'unité d'échange sera capable de le lire en mémoire.

Les 2 fonctions principales d'une unité d'échange seront alors :

- l'interprétation du programme canal.
- l'exécution des entrées-sorties.

I-3-2 Modes de fonctionnement

L'échange d'un bloc de données entre l'unité d'échange et l'organe d'E/S se décompose en échanges d'informations élémentaires : adresse, état, commande, données.

En ce qui concerne les données, les échanges élémentaires peuvent être plus ou moins rapprochés (de l'ordre de la micro-seconde à quelques milli-secondes). Ce qui fait qu'un échange de blocs, qui est la traduction d'une instruction d'E/S, peut durer assez longtemps et monopoliser l'unité d'échange.

Pour tenir compte du type d'organe connecté, il est souvent fait appel à 2 modes différents de fonctionnement de l'unité d'échange :

- le mode sélecteur ou "burst" (Fig.I-4) est celui qui correspond à une utilisation optimale des périphériques rapides. En effet, les échanges élémentaires concernant un même périphérique étant suffisamment rapprochés, on peut permettre l'échange d'un bloc entier de données avec le même organe.

Le bloc sera alors composé du nombre de mots indiqué lors de l'initialisation. Dans la suite, nous utiliserons le mot "sélecteur" pour qualifier l'unité d'échange, et "burst" pour qualifier le mode de transfert.

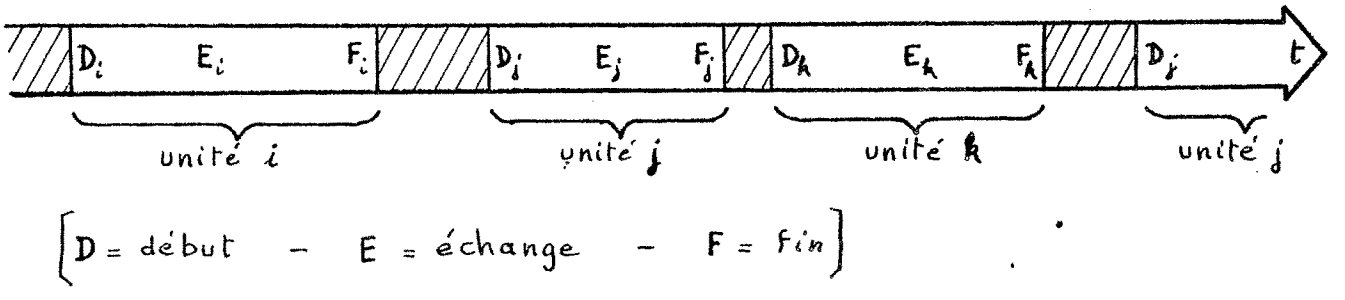


Fig.I-4 Déroulement des échanges dans le mode burst

- le mode multiplex (Fig.I-5) permet au contraire des transferts imbriqués avec plusieurs périphériques lents à la fois, l'unité d'échange laissant chacun effectuer un échange élémentaire d'un ou quelques mots.

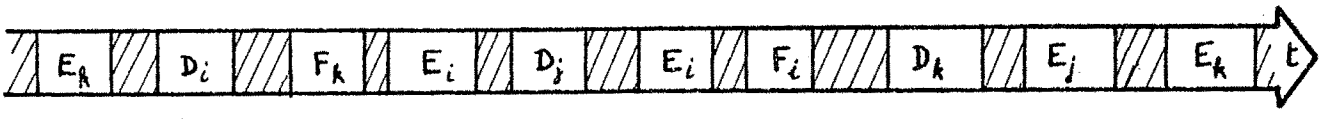


Fig.I-5 Déroulement des échanges dans le mode multiplex

Le plus souvent ces 2 modes de fonctionnement sont réalisés sur 2 types distincts d'unité d'échange.

Remarque : certaines unités d'échange pouvant travailler dans le mode multiplex, autorisent des transferts en mode "burst" avec certains périphériques.

I-3-3 Réalisation d'une Unité d'Echange

D'une façon générale, les unité d'échange peuvent être réalisées comme toute machine informatique, par exemple <ANC2> :

- soit sous la forme d'une machine câblée, physiquement distincte de l'unité centrale
- soit en micro - programmant une micro - machine universelle
- soit sous la forme d'un micro - programme supplémentaire sur la même micro-machine que l'unité centrale.

En ce qui concerne une unité d'échange multiplexée, la ressource nécessaire au contrôle d'un processus d'échange sera appelée "sous-canal". Il faudra donc prévoir :

-1- une mémorisation de l'état de l'échange avec chaque chaque organe d'E/S en cours d'exécution d'une instruction d'E/S, c'est-à-dire aussi bien l'état du transfert que l'état du programme canal correspondant.

Cette mémorisation pourra être assez conséquente (ex : 1024 mots de 72 bits dans le cas du 360/67) et se situer :

- dans une mémoire locale de l'unité d'échange
- dans une zone débanalisée de la mémoire centrale

-2- un organe commun effectuant les échanges proprement dits.

Nous verrons dans un exemple le fonctionnement d'une unité d'échange multiplexée réalisée par multi-micro-programmation.

I-3-4 La gestion des organes d'E/S

Il s'agit maintenant de savoir où localiser, dans le système d'entrée-sortie, les algorithmes temps-réel décrivant le fonctionnement complet d'un organe d'E/S. Chaque composant du système est en fait un emplacement envisageable <ANC2>.

I-3-4-1 Au niveau de l'organe lui même

C'est le cas très fréquent de l'utilisation d'une unité de liaison (unité de contrôle chez IBM) qui gère alors plusieurs organes de même type. Ces unités peuvent être réalisées aussi, par exemple, à l'aide de mini-ordinateurs comportant éventuellement des micro-programmes spéciaux pour les besoins de périphériques rapides.

I-3-4-2 Au niveau de l'unité d'échange

Dans cette solution, on réalise en fait la fusion des fonctions de l'unité d'échange et de l'unité de liaison. L'organe obtenu s'appelle alors un "processeur d'E/S" et on les trouve par exemple dans le modèle IBM 370/125, les Solar 16, le Mitra 125... Ils sont réalisés en général sur des mini-ordinateurs spécialement microprogrammés (et même multi-microprogrammés s'ils doivent gérer des organes d'E/S différents). Dans ce cas on n'a plus nécessairement d'interface d'E/S normalisé. Leur fonctionnement vis-à-vis de l'unité centrale est cependant le même que celui des unités d'échange.

I-3-4-3 Au niveau de l'unité centrale

On ajoute alors des micro-programmes supplémentaires, comme dans le cas de la réalisation d'une unité d'échange par multi-microprogrammation et on obtient la simulation d'un processeur d'E/S. Ceci donne une machine économique mais dont la commutation entre les micro-programmes est assez complexe (mécanismes de micro-interruptions). Ex: IBM 360/25, Mitra 15...

I-3-4-4 Au niveau de la mémoire centrale

Il est toujours envisageable de conserver la gestion de certains périphériques lents par programme.

I-3-5 Exemple : les entrées-sorties sur l'ordinateur IBM 360/40 <CHU>

Il s'agit seulement d'une première approche des mécanismes mis en jeu lors du déroulement d'une entrée-sortie sur un ordinateur possédant plusieurs types d'unités d'échange utilisant quelques-unes des notions que nous venons de décrire.

I-3-5-1 Présentation sommaire du modèle

La fig.I-6 montre la structure du modèle 40 qui est un modèle intermédiaire dans la gamme IBM 360. C'est une machine micro-programmée composée de 4 niveaux de mémoire, d'une unité centrale, d'un canal multiplexeur, et d'un (ou 2 en option) canal sélecteur. L'unité centrale et les canaux partagent un nombre variable de registres et de zones mémoire selon le type de canal et le mode de fonctionnement.

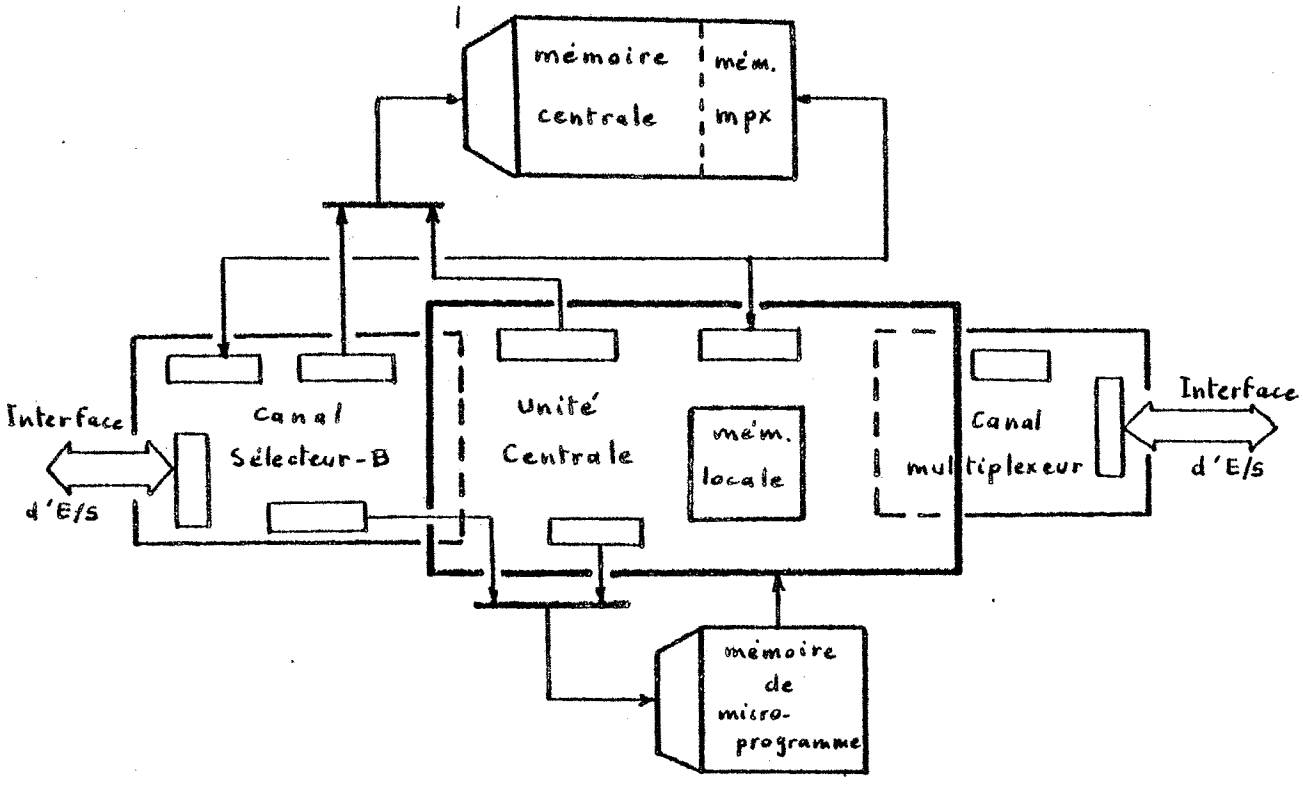


Fig.I-6 Représentation simplifiée du 360/40

La mémoire multiplex est une extension de la mémoire centrale qui contient toutes les informations de contrôle nécessaires au fonctionnement du canal multiplexeur.

La mémoire locale contient les registres généraux, des zones tampons pour le fonctionnement du canal multiplexeur, des mots de contrôle pour les canaux sélecteurs, 2 niveaux de zones de sauvegarde pour les commutations d'état entre l'unité centrale et les canaux, des zones de travail...

La mémoire de micro-programme a 2 registres d'adresse : un pour l'unité centrale, l'autre pour le canal sélecteur.

I-3-5-2 Fonctionnement commun d'une entrée-sortie dans le système IBM 360 (fig.I-7) <IBM1>

Quels que soient le modèle et le type de canal impliqués dans un échange, 5 mots de contrôle sont nécessaires :

- le CAW ou mot d'adresse du programme canal
- le CCW ou mot de commande (= instruction)
- le CSW ou mot d'état de l'échange
- le UCW ou mot de contrôle de l'échange
- le PSW ou mot d'état du programme.

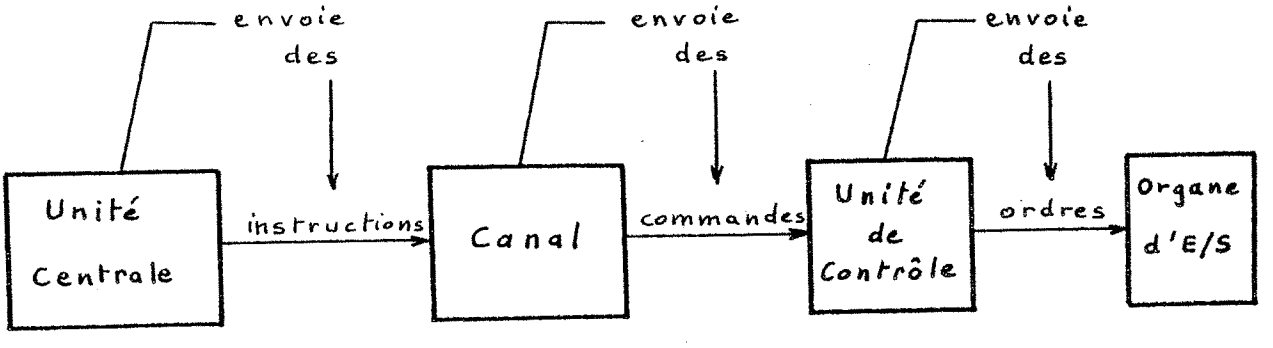


Fig.I-7 Schéma général des E/S dans le système IBM 360

Remarques :

-1- les UCW contiennent donc toutes les informations nécessaires au canal pour le déroulement d'une E/S. Leur nombre et leur format dépendent du modèle de canal.

- l'instruction fournit l'adresse de l'unité
- le CAW fournit l'adresse du 1er CCW et les clés de protection mémoire
- le CCW donne le code de la commande, les indicateurs, le nombre d'octets à transférer, les adresses des données
- le canal et l'unité fournissent leur état.

-2- les 4 autres mots de contrôle sont accessibles au programmeur.

Au niveau du programme, de l'unité centrale, du canal, une instruction d'E/S impliquant un transfert se déroule toujours de la même façon :

- test des éventuelles erreurs de programme
- test de la disponibilité des différents organes mis en jeu
- recherche des CAW, CCW, UCW
- positionnement du Code Condition
- libération de l'unité centrale et recherche de l'instruction suivante
- transfert effectif
- interruption de l'unité centrale à la fin de l'E/S.

Enfin, au niveau de l'interface d'E/S, les différentes séquences sont également communes et seront vues en détail dans le chapitre suivant.

I-3-5-3 Fonctionnement spécifique des canaux sur le modèle 360/40

Il faut envisager successivement le fonctionnement des 2 types de canaux.

a) le canal multiplexeur

Sa principale caractéristique est de n'avoir en propre que les micro-programmes qui gèrent les échanges, 2 registres (un tampon et l'autre contenant des indicateurs d'erreurs) et le contrôle des bus de l'interface d'E/S. Il peut gérer 128 sous-canaux pouvant servir simultanément 128 périphériques lents. Il utilise les autres ressources de l'unité centrale au fur et à mesure de ses besoins. Le déroulement de l'E/S peut se faire selon les 2 modes de fonctionnement définis précédemment ($ I-3-2) et conduit au schéma de la fig.I-8.

- en mode "burst" (par bloc) c'est en fait l'unité centrale qui exécute elle-même la fonction de canal. Elle charge ses registres avec les UCW pris dans la mémoire-multiplex et effectue les transferts, en mettant à jour les adresses et les comptes des données, jusqu'à obtention de l'état de fin d'E/S.

- en mode multiplex, dès qu'un organe d'E/S le réclame, une interruption de micro- programme déconnecte l'unité centrale du canal, son micro-programme est suspendu, ses registres sont sauvegardés dans la mémoire locale et sont chargés par les UCW. Cependant, l'unité centrale reprend le contrôle après chaque octet échangé avec la mémoire centrale. Ces commutations d'état entre unité centrale et canal se poursuivent jusqu'à l'obtention l'état de fin d'E/S.

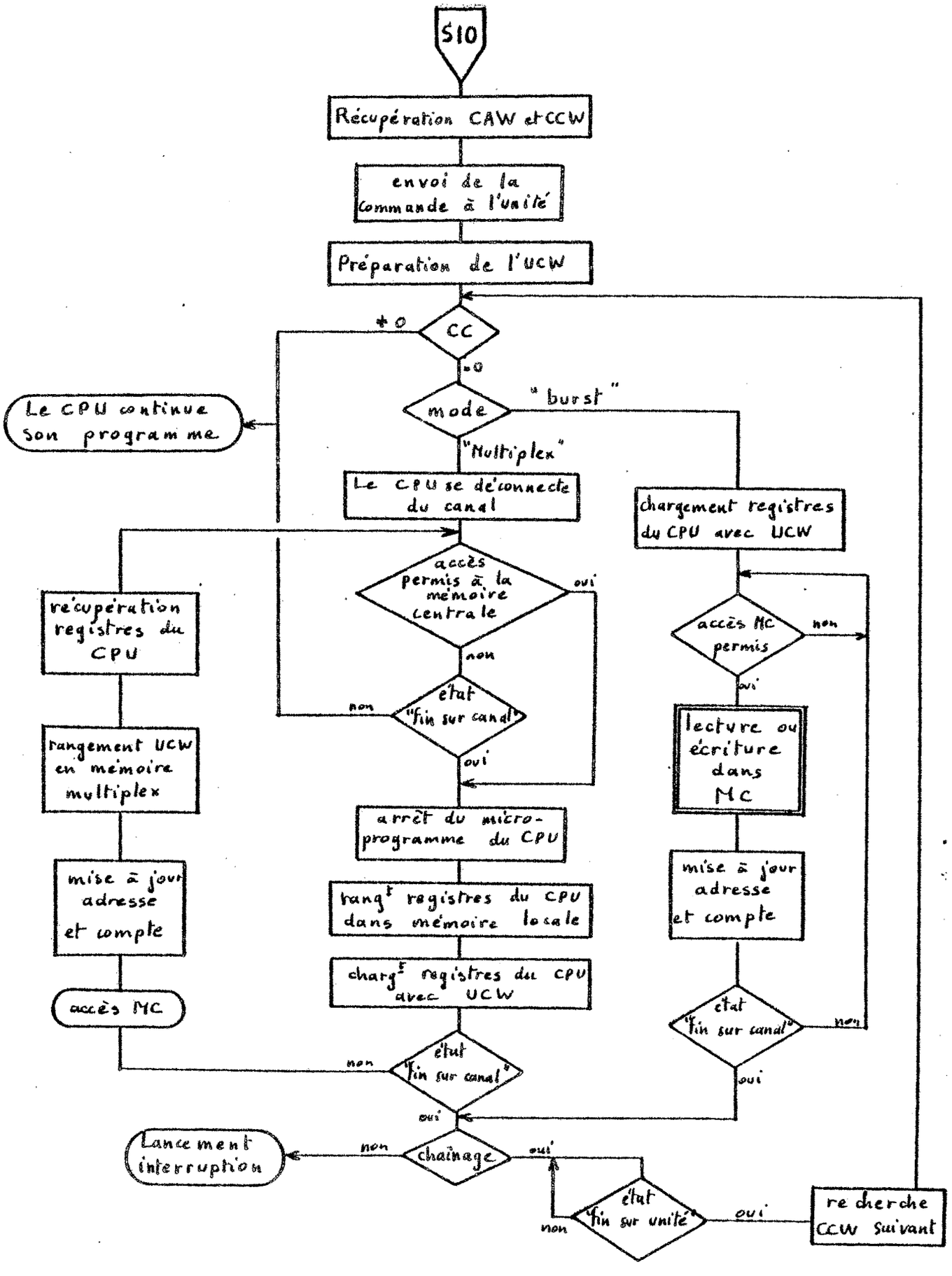


Fig.I-8 Organigramme simplifié d'un Start I/O sur le canal multiplexeur du 360/40

Remarques :

-1- Pendant ce mode de transfert, si le canal sélecteur a besoin de la mémoire centrale, il y a, à nouveau, changement d'environnement et utilisation d'une 2ème zone de sauvegarde des informations correspondant à l'échange en cours, dans la mémoire locale.

-2- La fin de l'E/S se manifeste par une interruption qui, si elle est permise, conduit à l'opération classique de l'échange des PSW et à l'exécution du programme d'interruption approprié.

-3- Le débit maximum en mode burst est de 228 K.octets/s, et il peut atteindre 26 K.octets/s en mode multiplex, avec décroissance simultanée de l'activité de l'unité centrale, cette activité pouvant même s'annuler.

b) le canal sélecteur

En réalité il y en a 2 types :

- le type A qui partage à un assez haut degré, les registres, le chemin des données et le contrôle micro-programmé avec l'unité centrale. Le transfert des données ne se fait pas directement entre la mémoire centrale et l'interface d'E/S, mais par l'intermédiaire de la mémoire locale qui fournit une zone de 16 registres tampons d'un octet. Quand cette zone est à moitié pleine, les registres de l'unité centrale sont sauvegardés et utilisés pour l'envoi des données à la mémoire centrale.

- le type B (fig.I-10) qui est le plus utilisé et qui possède sa propre circuiterie. Il ne prend presque pas de temps à l'unité centrale, n'utilise la mémoire locale que pour les UCW et non comme tampon et accède directement à la mémoire centrale dès

qu'il a 2 octets à transférer. Ceci se fait en forçant l'exécution d'un micro-programme propre car il peut accéder au registre d'adresse de la mémoire de micro-programme. Son micro-programme se contente de sauvegarder temporairement le registre de donnée de la mémoire centrale pour effectuer son propre accès avec son propre registre d'adresse. Ceci se poursuit jusqu'à épuisement du compte et réception de l'état de fin d'E/S. Alors on a encore forçage d'un micro-programme de fin d'E/S conduisant à la génération d'une interruption d'E/S à l'unité centrale.

Remarque : les débits maximum obtenus avec un canal sélecteur de type B sont de 400 K.octets/s, ou 300 K.octets/s si 2 canaux sélecteurs sont utilisés simultanément. Ces débits plus élevés sont obtenus par le fait que la plupart des opérations décrites sont contrôlées par une logique câblée et qu'il y a donc moins d'interférences avec l'unité centrale.

I-3-5-4 Conclusion

Ce paragraphe a permis de montrer plusieurs réalisations différentes d'une unité d'échange sur le même ordinateur et de voir les mécanismes de sauvegarde mis en jeu. On a ainsi une idée de l'efficacité relative des différentes solutions utilisées. Dans le prochain chapitre nous aurons l'occasion d'étudier une autre réalisation d'unité d'échange sur un modèle plus évolué de cette même famille.

canal \ mode		"multiplex"	"burst"
multiplexeur		26	228
sélecteur type B	un seul canal	--	400
	2 canaux	--	300

Fig.I-9 Tableau résumé des débits des canaux du 360/40 (valeurs maximum en K.octets/s)

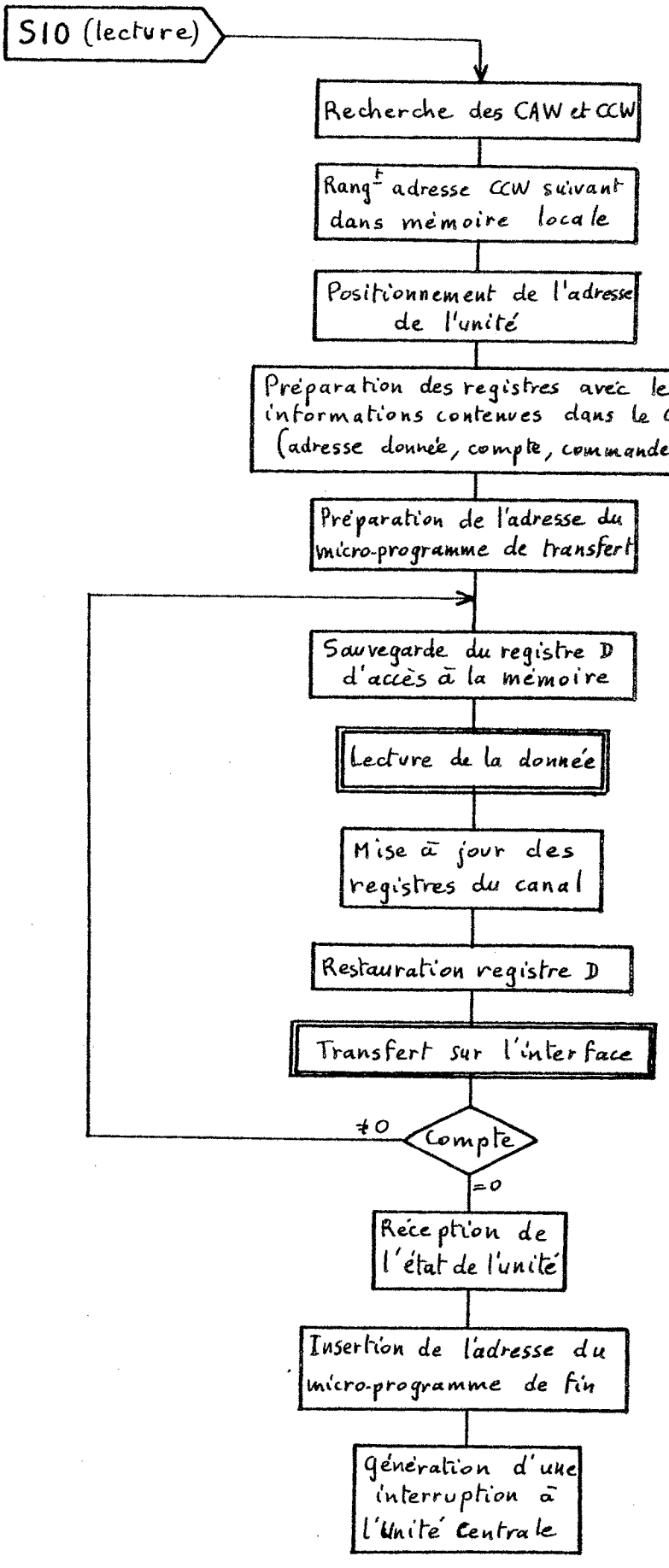


Fig.I-10 Organigramme simplifié d'une E/S sur un canal sélecteur du type B

I-4 LES ENTREES-SORTIES DANS LES MICROPROCESSEURS

I-4-1 Généralités

Il n'est pas question ici de faire un inventaire de toutes les possibilités d'E/S offertes par les microprocesseurs, ni une description exhaustive de tous les mécanismes mis en jeu (voir <ANC1>,<LIL>,<ZAK1>,<ZAK2>). Nous essaierons seulement de dégager certaines idées qui conditionnent en partie l'organisation des E/S dans les systèmes à microprocesseurs.

Des contraintes technologiques, telles que la surface de silicium utilisée, la densité d'intégration et le nombre de connexions avec l'extérieur, influent sur la quantité de fonctions offertes à l'utilisateur. Les troisièmes en particulier de conditionner la forme du dialogue avec l'extérieurî (largeur des bus d'adresse et de donnée, nombre de lignes de contrôle, mécanisme d'E/S...) et donc le nombre de boîtiers à mettre en oeuvre.

Nous trouverons ainsi définies quelques classes de microprocesseurs selon les facilités d'E/S (fig.I-12). Dans certains cas il pourra s'agir d'architectures internes réellement spéciales, comme pour le Fairchild F8, le Signetics 8X300, les Rockwell PPS-4 et 8, en particulier pour ces derniers où les boîtiers d'E/S sont de vrais processeurs d'E/S (fig.I-11).

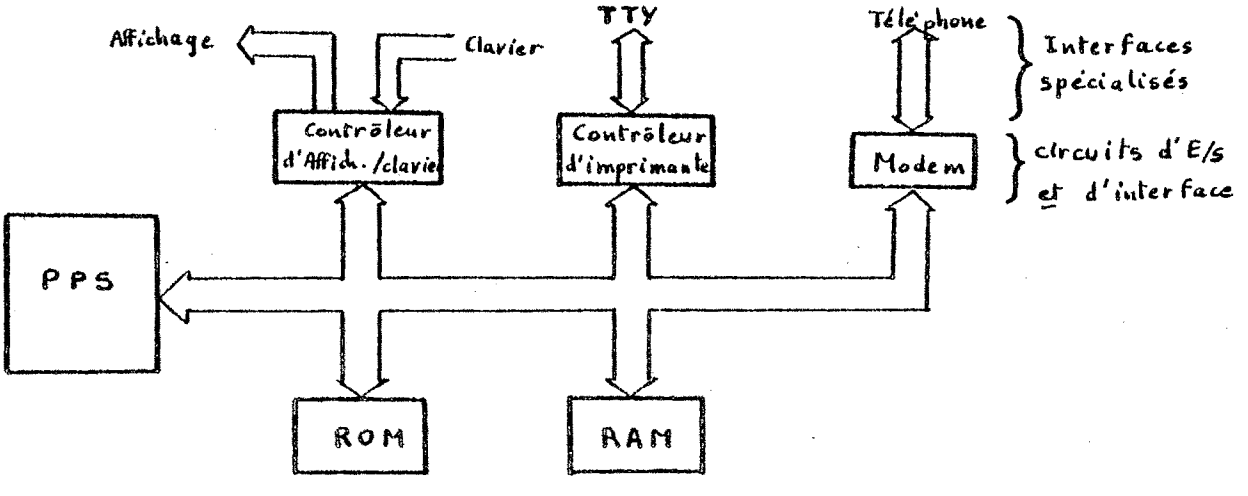
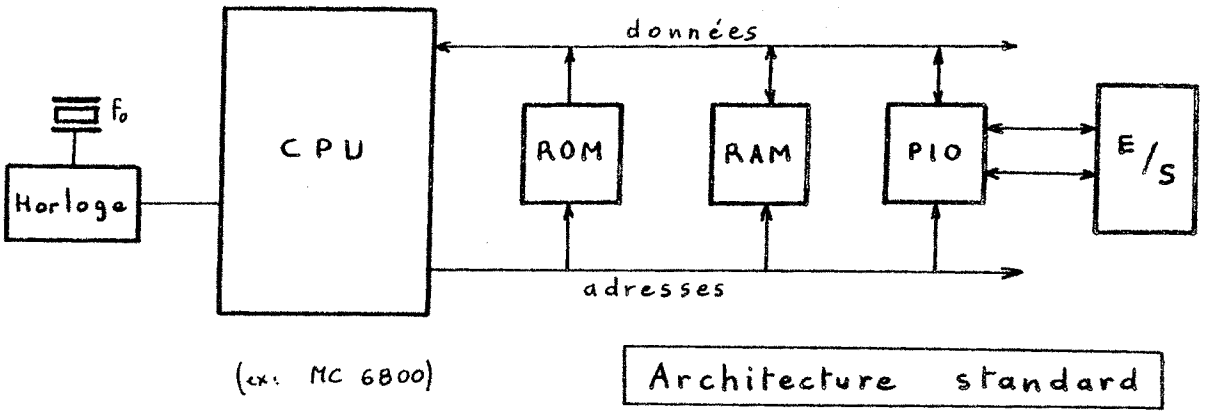
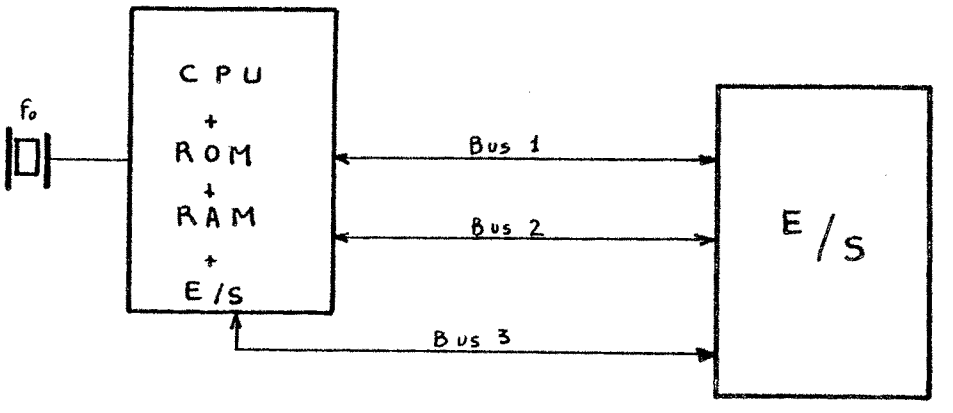


Fig.I-11 La structure du système Rockwell (PPS 4 et 8)



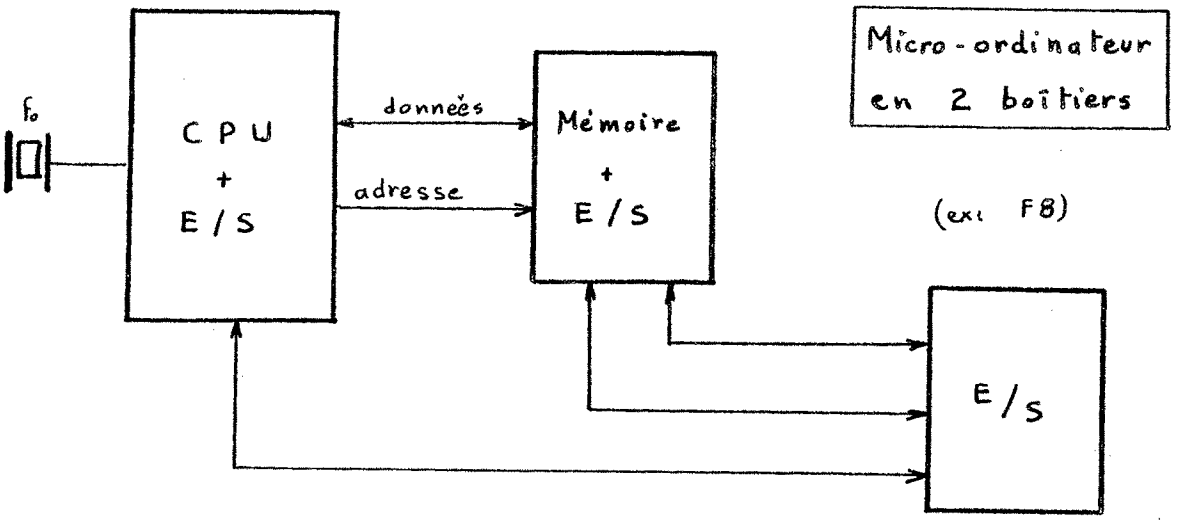
(ex: MC 6800)

Architecture standard



Micro-ordinateur en un boîtier

(ex: I 8048)



Micro-ordinateur en 2 boîtiers

(ex: F8)

Fig.I-12 Les différentes architectures de microprocesseurs

Par rapport aux ordinateurs traditionnels, il n'y a plus en général de distinction entre les instructions habituelles de traitement et les instructions d'E/S. Cependant certains microprocesseurs conservent des instructions d'E/S spécifiques (par ex. Intel 8080). Les autres considèrent les boîtiers d'E/S comme des emplacements mémoire et permettent ainsi d'utiliser les instructions sur des données se trouvant dans ces circuits.

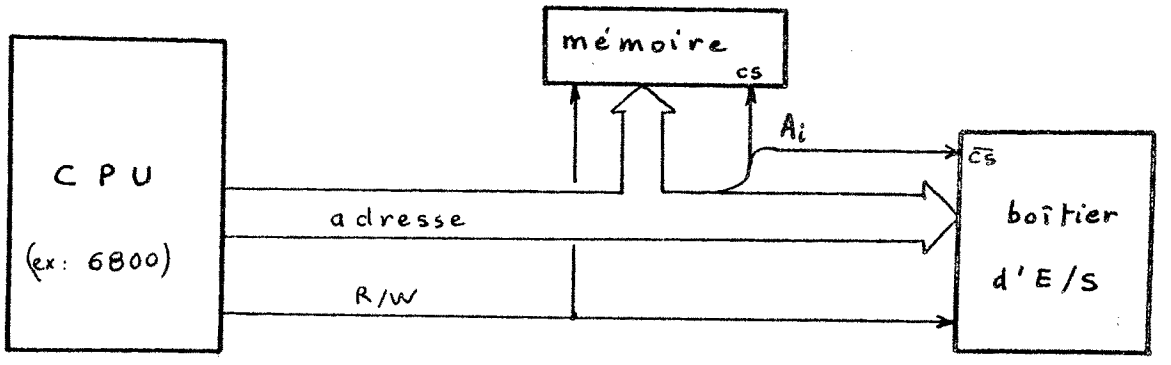
Ceci revient à étendre la puissance des opérations en échange d'une (petite) perte de capacité mémoire. L'adressage de ces circuits se fait en général en décodant les bits poids fort du bus d'adresse. Remarquons que c'était déjà le cas dans certains mini-ordinateurs (PDP 11 par ex.).

Les instructions d'E/S spécialisées sont plus courtes et nécessitent moins de circuits de décodage mais elles utilisent 2 broches de plus (I 8080 par ex.) et on tombe alors sur la contrainte du nombre limité de connexions. La fig.I-13 schématise ces 2 types d'application.

Enfin, l'intégration à grande échelle a permis la réalisation de circuits d'E/S aux fonctions parfois très complexes, programmables, assurant les échanges avec une gamme très large de périphériques standards (écrans vidéo, cassettes, disquettes) et d'applications particulières. Ils sont habituellement classés en 2 groupes : ceux qui effectuent des transferts en parallèle et ceux qui traitent les informations en série.

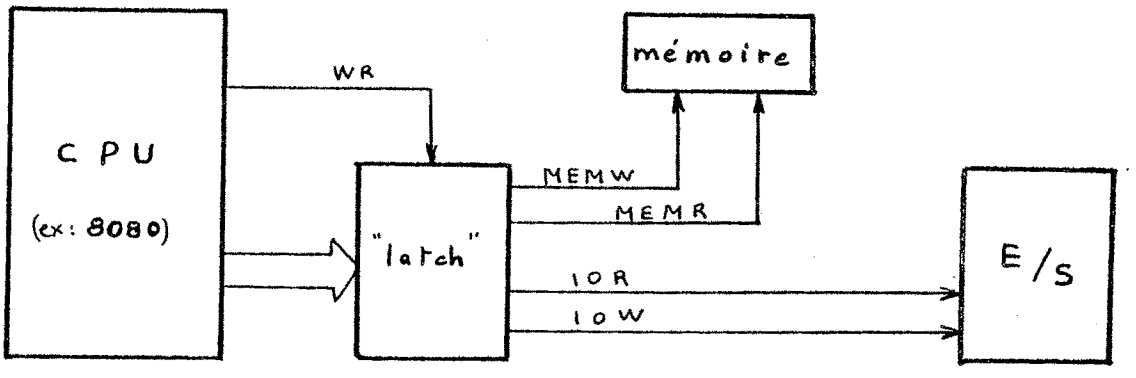
I-4-2 Les techniques utilisées

On retrouve dans les microprocesseurs les méthodes classiques avec cependant une utilisation fréquente des E/S programmées (simplicité, coût..) et des appels au mécanisme d'accès direct à la mémoire (DMA) dans les systèmes plus importants et les systèmes multi-microprocesseurs. Quant au mécanisme d'interruption il s'avère bien adapté lors de l'acquisition de données asynchrones non attendues par le programme (voir <ANC3> p.118-122).



(a) E/S banalisées avec la mémoire

ex : $\begin{cases} A_i = 0 & : \text{accès E/S} \\ A_i = 1 & : \text{accès mémoire} \end{cases}$



(b) E/S spécialisées

Fig.I-13 Les 2 types d'instructions d'E/S

I-4-2-1 Les E/S programmées

Elles nécessitent une liaison effective préalable entre le CPU et le circuit adressé. Ceci a lieu le plus souvent par un sondage ("polling") effectué par le CPU, d'indicateurs dans les boîtiers d'E/S, suivi d'une "poignée de main" ("handshaking") entre le CPU et le circuit demandeur.

Elles conduisent évidemment à une grande perte de temps pour le CPU mais sont utilisées dans la plupart des applications des microprocesseurs (commande industrielle par ex.).

I-4-2-2 Les E/S par interruptions

Elles évitent le sondage par le CPU, mais elles imposent un nombre parfois important de circuits annexes, en particulier pour gérer plus efficacement la priorité dans le cas (général) où il y a plus de boîtiers concernés que de broches d'interruptions au CPU. On doit alors encore détecter la source de l'interruption (sondage). Une solution, coûteuse en matériel, est d'effectuer la "vectorisation" des interruptions, à moins qu'elle ne soit prévue par le constructeur (Zilog Z80).

De plus les interruptions nécessitent souvent l'exécution d'un processus de sauvegarde des informations, ce qui se traduit par la gestion d'une pile (câblée ou programmée), ralentissant leur prise en compte effective.

Cependant elles sont très utiles dans les circonstances signalées au début de ce paragraphe (par ex. contrôle en temps réel).

I-4-2-3 L'accès direct à la mémoire (DMA)

Il est indispensable lorsqu'il s'agit de gérer des périphériques assurant des transferts rapides de blocs d'information (disques, écrans...). Cela oblige pratiquement à utiliser un boîtier spécial qui reçoit les demandes des circuits d'E/S et qui, par plusieurs techniques (halte du CPU, vol de cycle, multiplexage...) plus ou moins complexes, obtient du CPU la disponibilité du bus d'adresse et lance les échanges. Selon la méthode utilisée, on ralentira beaucoup (halte du microprocesseur ou pas du tout (multiplexage) le déroulement du programme ; et la prise en compte de la demande de DMA sera plus ou moins rapide.

Le contrôleur de DMA doit ête initialisé avant de gérer les échanges, et se comporte comme un canal simplifié.

La fig.I-14 résume les différentes techniques décrites.

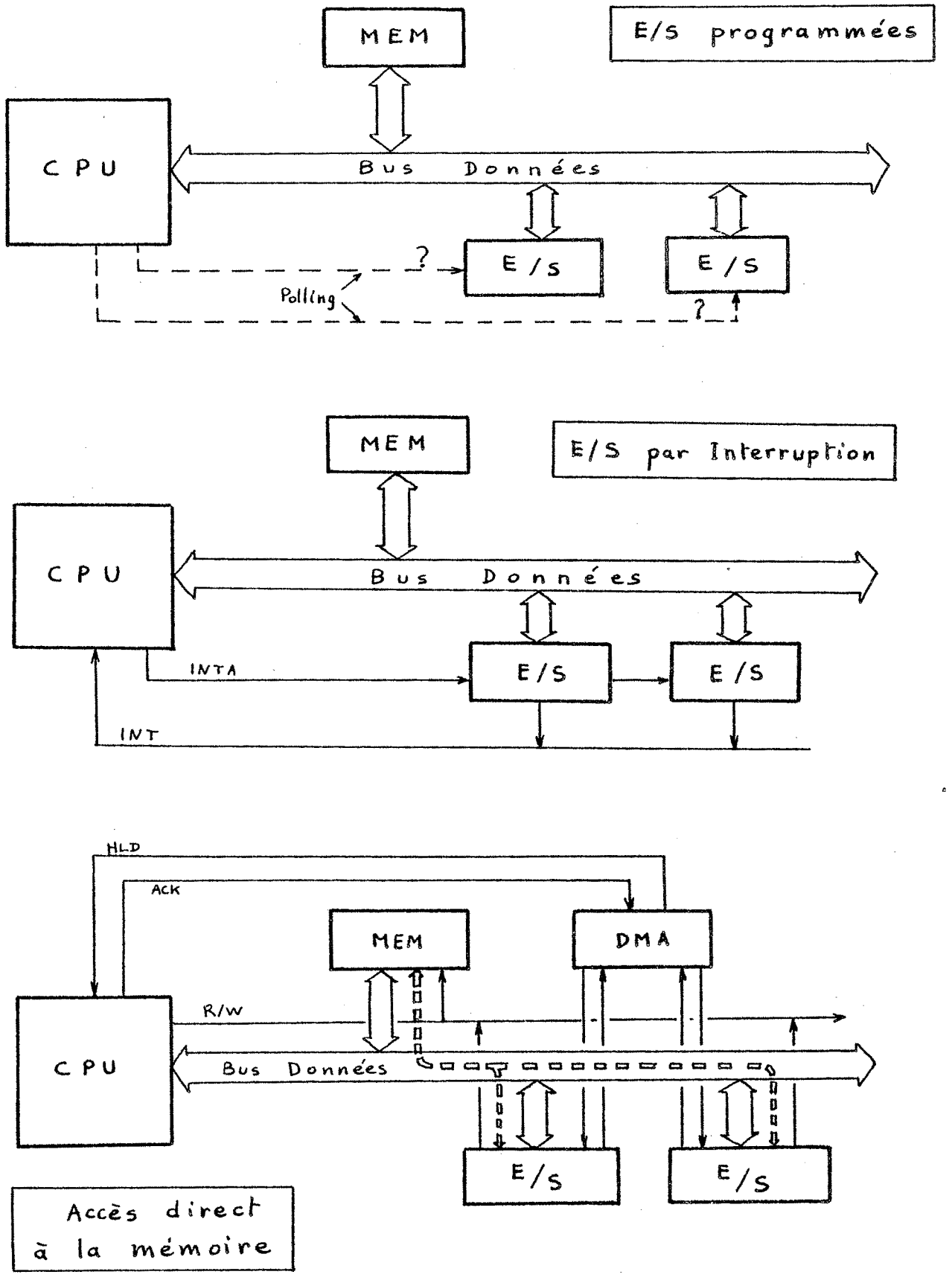


Fig.I-14 Les méthodes d'E/S sur les microprocesseurs

I-5 TENDANCES ACTUELLES

Elles sont bien évidemment liées à l'apparition des microprocesseurs. Ceux-ci ont provoqué en premier lieu une modification de la structure des contrôleurs de périphériques en remplaçant la logique câblée par de la logique programmée (ce qui a d'ailleurs été l'utilisation première des microprocesseurs dans un grand nombre de cas).

Nous pouvons alors concevoir l'organisation d'un système informatique, à 3 niveaux <ANC3> :

- les organes périphériques élémentaires
- les processeurs fonctionnels
- les applications.

I-5-1 Les organes périphériques élémentaires

Si on prend l'exemple d'un contrôleur de télécommunication type IBM 270x <ANC2>, on obtient l'évolution de la fig.I-15. Si on considère en outre que le circuit USRI (émetteur/récepteur synchrone universel) peut être réalisé en un seul circuit LSI effectuant une grande partie des fonctions du contrôleur primitif, on voit que le miniordinateur et/ou le microprocesseur peut alors accomplir des tâches assurées précédemment par du logiciel implanté sur l'ordinateur central.

On considère alors que le contrôleur a acquis une "intelligence". Ceci peut se retrouver évidemment sur l'ensemble des anciennes unités de contrôle et est courant sur les périphériques utilisés en micro-informatique : écrans de visualisation, mini-imprimantes, disques souples...

L'évolution de la technologie autorise la fabrication de circuits spécialisés pour ces types de périphériques, en un boîtier LSI (ou VLSI). (ex: FDC pour disques souples...)

On peut arriver ainsi à ce que le périphérique "écran", par exemple, devenu intelligent effectue tout un traitement d'édition de textes, de pagination, de graphisme...

On peut signaler aussi que cette tendance se manifeste en contrôle industriel, où sont proposés des circuits LSI effectuant les différentes conversions A/D et D/A, et se connectant directement au bus interne des microprocesseurs standards.

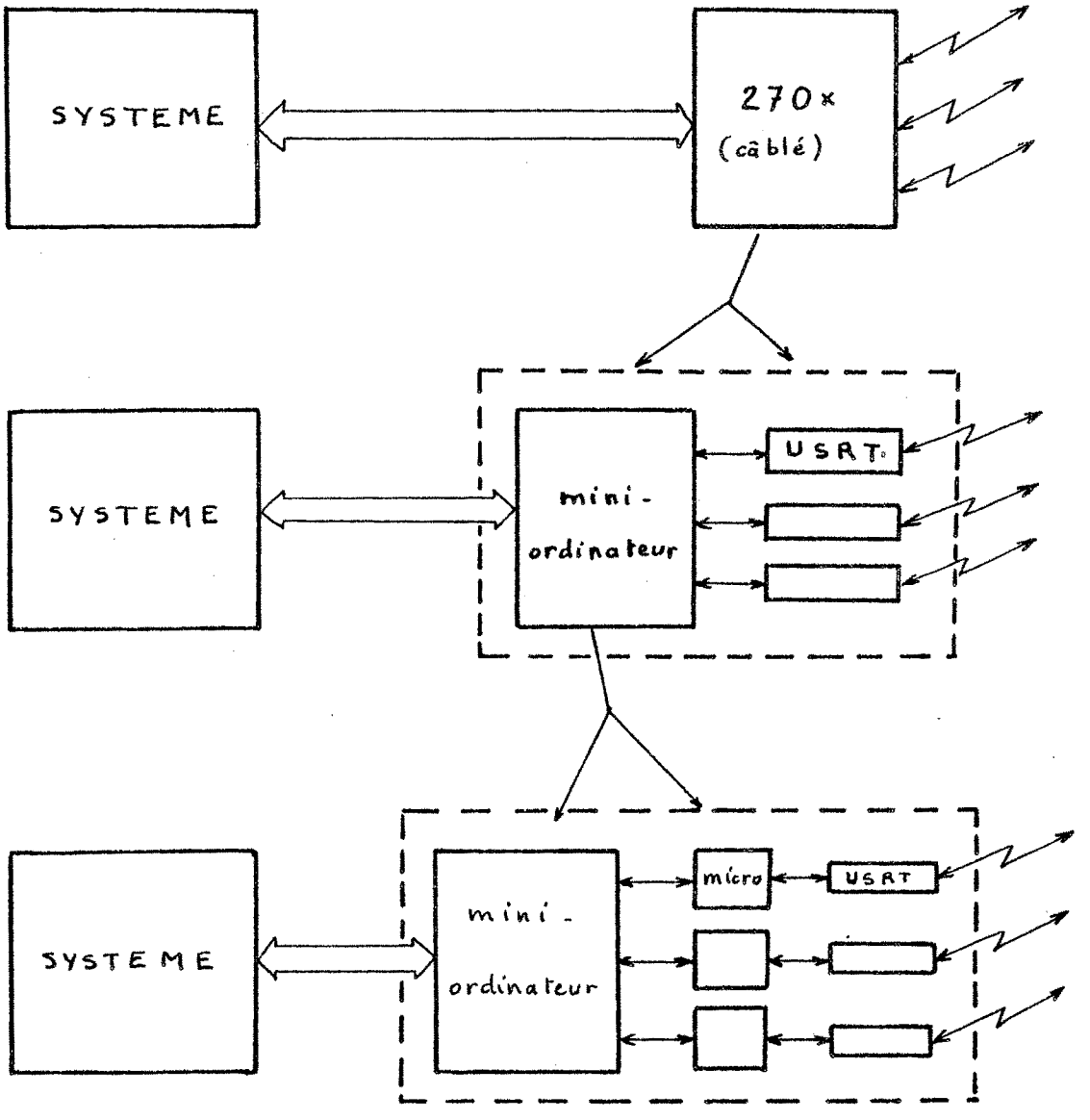


Fig.I-15 L'évolution d'un contrôleur de communication

I-5-2 Les processeurs fonctionnels

Un deuxième aspect de cette (r)évolution est la possibilité de réaliser économiquement des "processeurs fonctionnels" à partir du groupement des organes précédents autour de processeurs maîtres. Les communications se font alors par l'intermédiaire d'une mémoire commune et de "DMA" (fig.I-16).

On obtient alors une décentralisation de chaque fonction, ce qui conduit à un système plus souple.

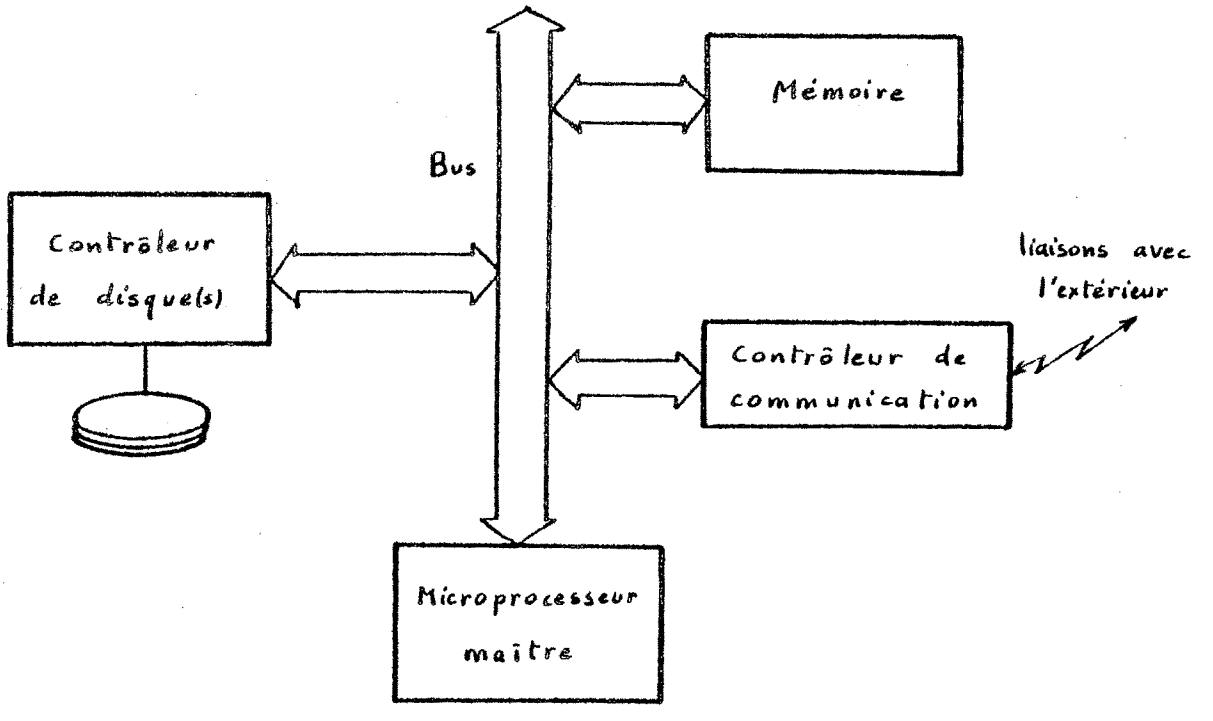


Fig.I-16 Exemple de processeur fonctionnel : un "Processeur Fichier"

Des exemples de tels systèmes modulaires sont la machine "CORAIL" <POUJ> et le "Processeur Base de Données" <BSA> réalisés par l'équipe de Recherches en Architecture des Ordinateurs de l'ENSIMAG .

I-5-3 Les applications

La liaison de plusieurs processeurs fonctionnels du niveau précédent peut alors être envisagée pour une application particulière.

Un moyen économique pour réaliser cette liaison est d'utiliser une ligne série <MAR>.

On obtient alors un réseau local tel que, par exemple, celui de la fig.I-17.

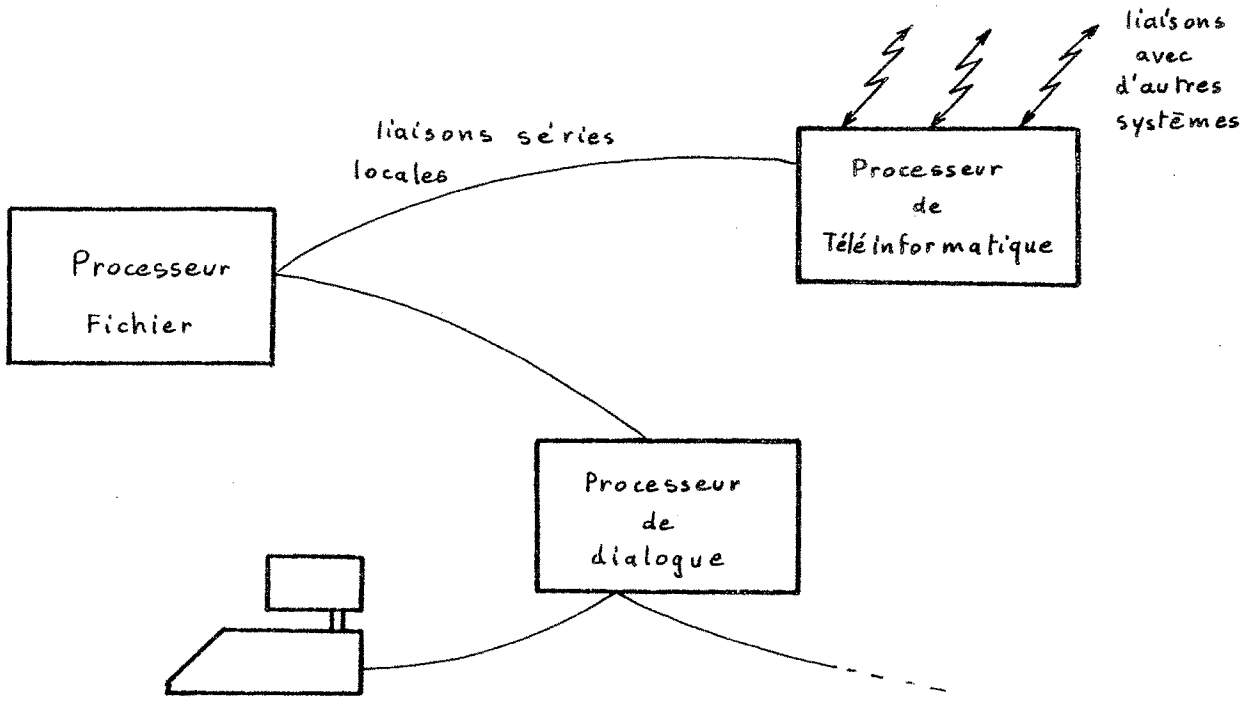


Fig.I-17 Réseau local à base de lignes série

Ces tendances se confirmeront d'autant plus que l'évolution technologique se poursuivra et qu'une normalisation des fonctions sera réalisée.

CHAPITRE II

LES ENTREES-SORTIES DANS LE SYSTEME IBM 360

II-1 INTRODUCTION

II-2 LE CANAL MULTIPLEXEUR

II-2-1 Description

II-2-1-1 Le canal principal

II-2-1-2 Le canal multiplex

II-2-1-3 Les canaux sélecteurs

II-2-1-4 La mémoire locale

II-2-1-5 Les débits

II-2-1-6 Les interfaces

II-2-2 Adressage des unités

II-3 FONCTIONNEMENT

II-3-1 L'instruction START I/O

II-3-1-1 Initialisation

II-3-1-2 Exécution de la commande

II-3-1-3 Fin de l'instruction

II-3-2 L'instruction HALT I/O

II-3-3 L'instruction TEST I/O

II-3-4 Quelques remarques

II-4 L'INTERFACE D'ENTREE-SORTIE

II-4-1 Description générale
- II-4-1-1 Les Bus
- II-4-1-2 Les lignes de sélection
- II-4-1-3 Les lignes d'identification
- II-4-1-4 Les lignes d'interblocage
- II-4-1-5 Les lignes de contrôle

II-4-2 Le mécanisme d'interconnexion

II-4-3 Le polling

II-4-4 Les séquences régies par l'interface d'E/S
- II-4-4-1 Les séquences de contrôle
- II-4-4-2 Les séquences d'interface

II-1 INTRODUCTION <IBM1>

Le schéma général de la structure du système IBM 360 est donné à la fig.II-1. Il fait apparaître les éléments mentionnés au chapitre précédent; les caractéristiques de chacun d'eux et leur nombre dépend du modèle, sauf en ce qui concerne les unités de contrôle et les organes d'E/S puisque l'interface d'E/S est normalisé pour toute la famille.

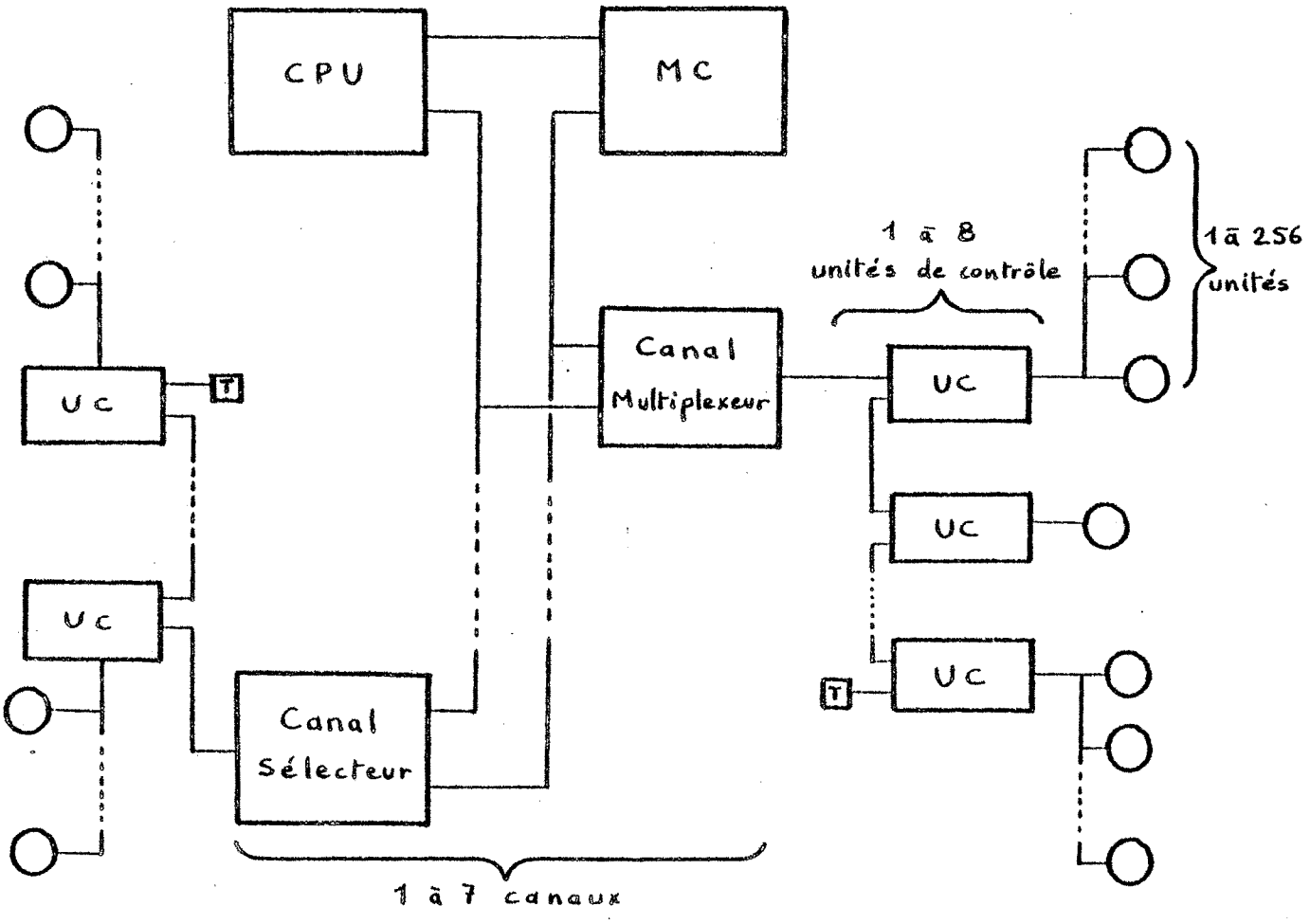


Fig.II-1 Le système IBM 360

Nous expliciterons : "organe d'E/S" quand il s'agira plus spécialement de celui relié au monde extérieur ou qui contient physiquement les données à échanger, et "unité d'E/S" (unité de contrôle) celui qui est physiquement relié à l'interface d'E/S, tout en sachant que les 2 peuvent être confondus...

En ce qui concerne plus précisément le modèle 360/67-1 qui sert de support à ce travail, l'organisation en est donnée à la fig.II-2 <IBM3>.

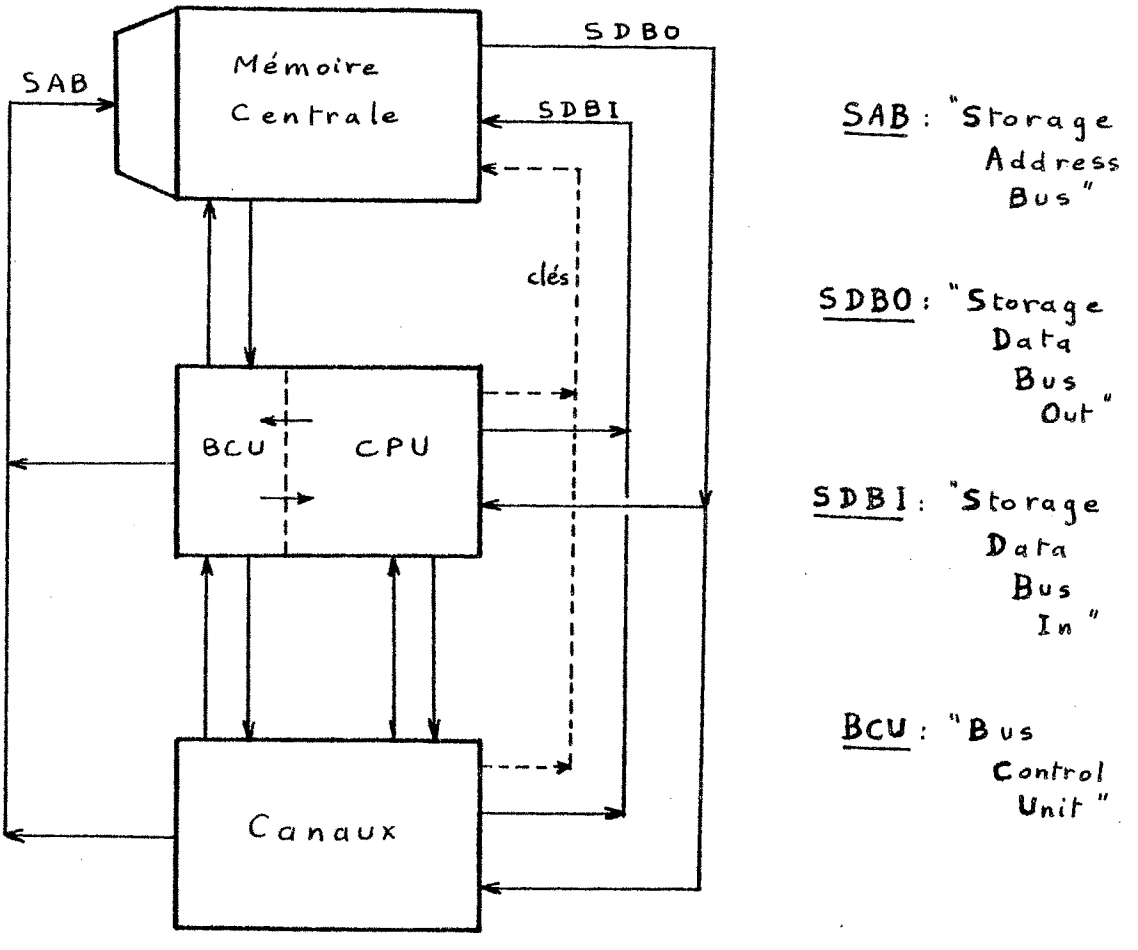


Fig.II-2 Le modèle IBM 360/67-1

Sur ce modèle, on trouve en particulier :

- une unité de contrôle du bus-mémoire ou BCU, qui gère les accès à la mémoire centrale, en particulier les conflits entre l'unité centrale et les canaux.

- 6 canaux sélecteurs et 1 canal multiplexeur : ces 7 canaux peuvent fonctionner simultanément et on peut ainsi atteindre des débits entre eux et la mémoire de 1,25 M.octets/s, mais chaque transfert a lieu par double-mot soit 8 octets

à la fois. Il y aura donc une requête du canal demandeur au mieux toutes les 6,4 µs. Seules les unités d'E/S très rapides (disques, tambours) pourront approcher ces temps ; les unités de bandes magnétiques les plus rapides provoqueront des requêtes seulement toutes les 23 ou 89 µs, par exemple.

D'un point de vue fonctionnel (fig.II-3), des "bus communs" relient toutes les unités et leur utilisation, pour une liaison donnée, est gérée au niveau du BCU au moyen de lignes de contrôle et de séquences simples du type "requête/réponse".

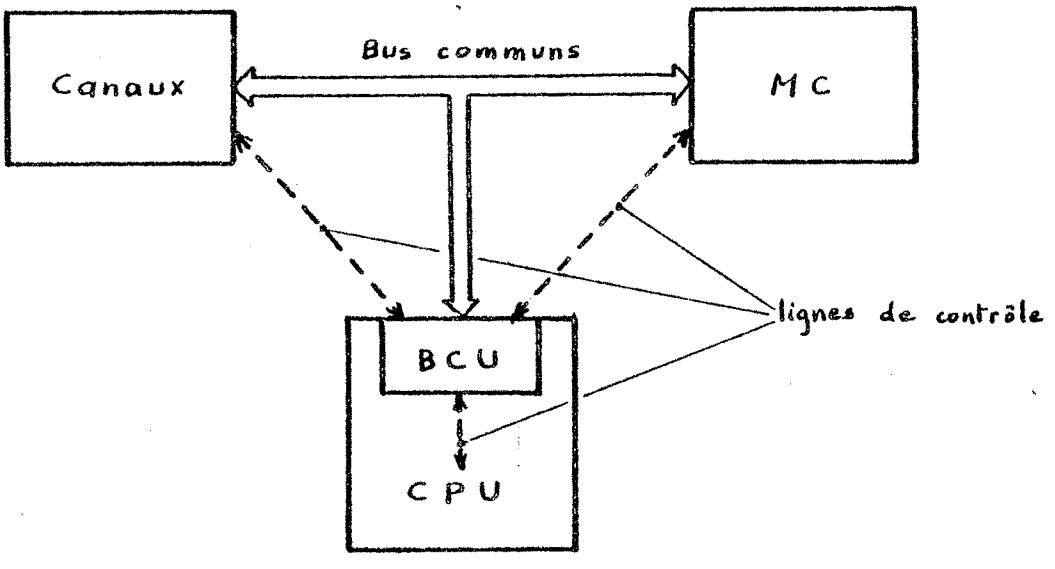


Fig.II-3 Schéma fonctionnel des 3 unités de base

Remarques :

-1- le BCU est "transparent" à certaines relations entre les canaux et le CPU.

-2- dans le modèle 67-2 (bi-processeur), les BCU (un par CPU) ne gèrent que les conflits d'accès entre les CPU et la mémoire centrale. Les accès entre les canaux et la mémoire sont assurés par des contrôleurs de canaux (CCU). Dans ce cas la mémoire doit gérer les conflits entre ces 2 voies, les bus communs étant dédoublés.

II-2 LE CANAL MULTIPLEXEUR <IBM4>

II-2-1 Description

Le canal multiplexeur (modèle 2870) est commun aux ordinateurs 360/65-67 et 360/75. Il est composé (fig.II-4) :

- d'un canal principal
- d'une mémoire locale
- d'un canal multiplex
- optionnellement, de 1 à 4 sous-canaux sélecteurs
- des différents interfaces : avec le CPU, la mémoire centrale et le BCU, les unités d'E/S
- de lignes de test et de diagnostic.

Remarque : il faut faire ici la distinction entre les sous-canaux sélecteurs et les sous-canaux du canal multiplex : les premiers possèdent une structure matérielle propre, alors que les deuxièmes ne sont que des ressources logiques.

II-2-1-1 Le canal principal

Il contient essentiellement 3 registres (Donnée, Contrôle, Adresse d'Unité) ainsi qu'un additionneur, des circuits de priorité d'accès à la mémoire locale, des horloges, etc... Il assure la gestion complète des E/S à l'aide des fonctions suivantes :

- réception de l'adresse des unités d'E/S et sélection du sous-canal correspondant
- conservation et mise à jour des adresses des données et des comptes d'octets
- transferts effectifs des informations entre les sous-canaux et les mémoires locale et centrale
- contrôle des accès à la mémoire locale
- assemblage/dés-assemblage des données pour le canal multiplex
- contrôle du chaînage le cas échéant.

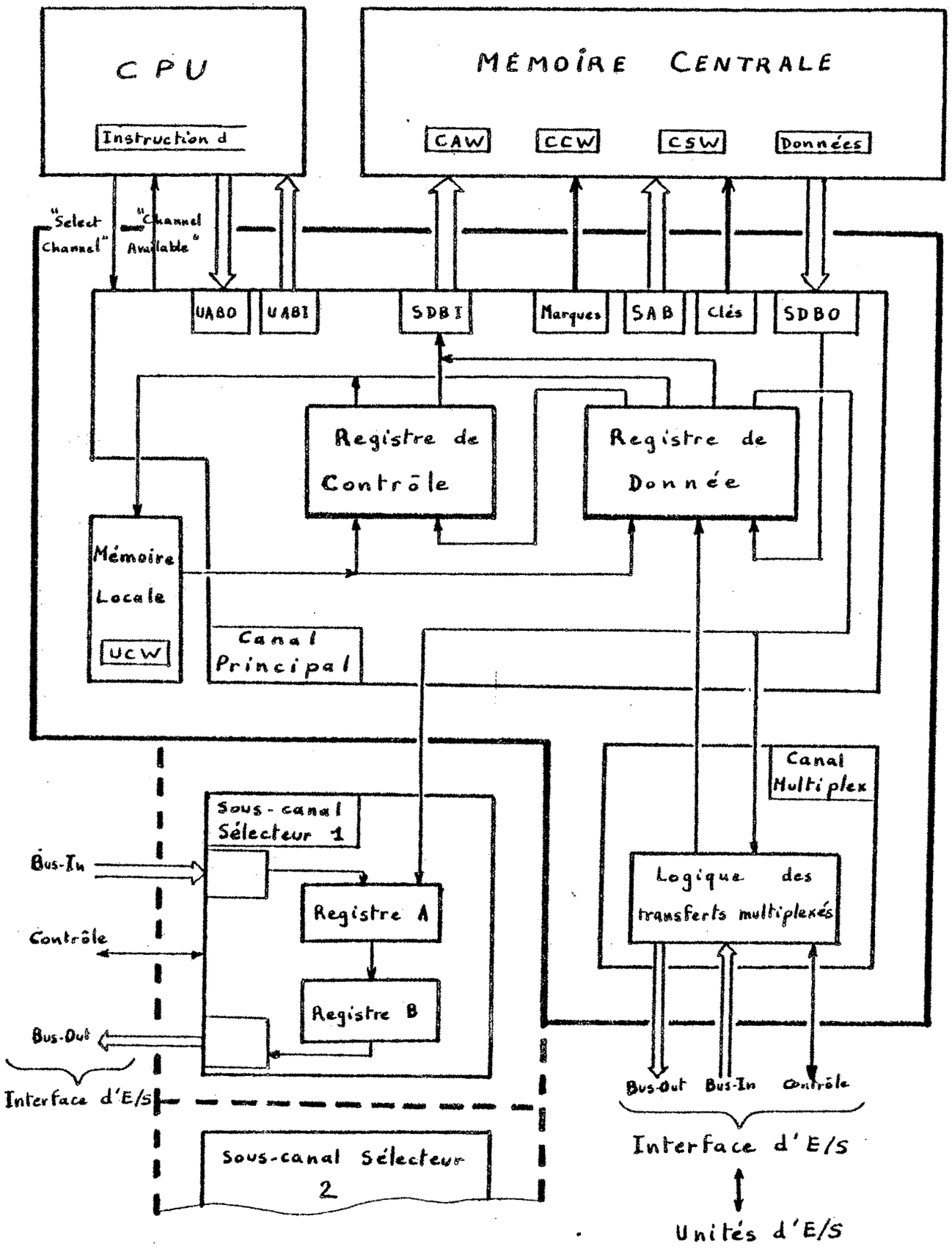


Fig.II-4 Le canal multiplexeur

II-2-1-2 Le canal multiplex

C'est le nom donné aux 192 sous-canaux attribués à ce modèle. Il opère normalement en mode "multiplex", mais un sous-canal peut imposer des transferts de blocs de données de longueur quelconque. Si la durée correspondante est supérieure à 32 µs, le transfert est en mode "burst", sinon il est en "byte-multiplex".

Le canal multiplex n'est composé que de quelques registres tampons alimentés par le registre de donnée du canal principal et par l'interface d'E/S.

La gestion simultanée des 192 sous-canaux se fait par l'intermédiaire de 8 unités de contrôle au maximum.

II-2-1-3 Les sous-canaux sélecteurs

Ils sont au nombre de 4 au maximum et n'opèrent qu'en mode "burst". Chacun d'eux peut gérer 16 organes d'E/S rapides mais ne peut être connecté à plus de 8 unités de contrôle. Ils possèdent un peu plus de circuits que le canal multiplex.

II-2-1-4 La mémoire locale

Elle contient les mots de contrôle des E/S définis au $ I-3-5-2. Chaque sous-canal nécessitant 4 UCW, elle contient 1024 mots de 72 bits. Elle est en liaison avec les 2 principaux registres du canal principal (contrôle et donnée). Son adressage est indexé sur l'adresse de l'unité.

II-2-1-5 Débits

Le canal multiplex opérant seul a un débit de 110 K.octets/s sur un sous-canal à la fois. Les 3 premiers sous-canaux sélecteurs, un débit de 180 K.octets/s ; le quatrième seulement de 100 K.octets/s. Mais le débit du canal multiplex chute de 22 K.octets/s à chaque addition d'un sous-canal sélecteur (sauf s'il s'agit du quatrième où la perte n'est que de 14 K.octets/s).

Remarques :

-1- les modèles 360/65-67-75-91 disposent aussi d'un canal sélecteur dont le débit maximum est de 1,3 M.octets/s et destiné à gérer les disques et tambours (à ne pas confondre avec les sous-canaux sélecteurs rattachés en option au canal multiplexeur 2870 et s'occupant plutôt de la gestion des bandes magnétiques par exemple).

-2- on peut ici faire un rapprochement avec les débits des canaux du modèle 360/40 (tableau de la fig.I-9), et constater le gain obtenu par le choix d'une solution la plus indépendante possible de l' unité centrale au prix d'une complexité sensible. Cette même constatation peut se faire à l'intérieur même du canal multiplexeur en comparant les débits du canal multiplex et des sous-canaux sélecteurs moins dépendants du canal principal.

II-2-1-6 Les interfaces

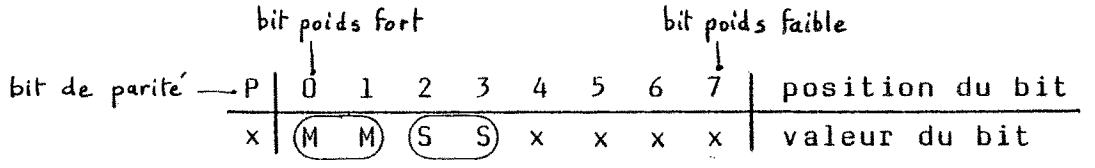
Nous décrirons plus loin en détail l'interface d'E/S qui est normalisé pour toute la famille.

Nous donnerons dans l'annexe 1, les significations des lignes les plus importantes des autres interfaces avec l'unité centrale, le BCU et la mémoire centrale.

II-2-2 Adressage des unités

En règle générale, dans tout le système 360, l'adresse d'un organe d'E/S est un mot de 16 bits composé de l'adresse du canal (sélecteur ou multiplexeur) auquel il est connecté, sur les 8 bits de poids fort, et de son adresse proprement dite sur les 8 bits de poids faible. Comme seulement 7 canaux peuvent être installés, le premier caractère de l'adresse sera un chiffre de 0 à 6, le canal multiplexeur se voyant attribué l'adresse 0.

Ce canal a une capacité d'adressage de 256 organes d'E/S : 192 pour le canal multiplex et 64 pour les sous-canaux sélecteurs (16 par canal). Le codage de l'octet de poids faible de ces adresses est donné à la fig.II-5.



. Si M-M ≠ 1-1 : l'unité est rattachée au canal multiplex et a une adresse comprise entre 000 et 0BF (192 valeurs).
. Si M-M = 1-1 : l'unité est rattaché à un sous-canal sélecteur suivant la signification des bits S-S :

S S		adresse d'unité
0 0	sous-canal sélecteur 1	0Cx
0 1	- - - 2	0Dx
1 0	- - - 3	0Ex
1 1	- - - 4	0Fx

Fig.II-5 Codage des adresses d'unités d'E/S sur le canal multiplexeur

Remarques :

-1- l'adresse indique indifféremment le sous-canal, l'unité ou l'organe d'E/S.

-2- une unité d'E/S peut être attachée à plusieurs canaux et un organe d'E/S à plusieurs unités. Si c'est le cas cette unité ou cet organe aura des adresses distinctes sur chaque voie d'accès.

-3- si aucune unité d'E/S ne répond à une adresse, l'organe correspondant est dit "non opérationnel". Si une unité répond à une adresse pour laquelle aucun organe d'E/S n'est attaché, cet organe apparaît dans l'état "non prêt"

II-3 FONCTIONNEMENT

II-3-1 L'instruction START I/O

Nous prendrons comme support de description le déroulement de l'instruction "Start I/O" (SIO), dont le schéma global est donné ci-dessous :

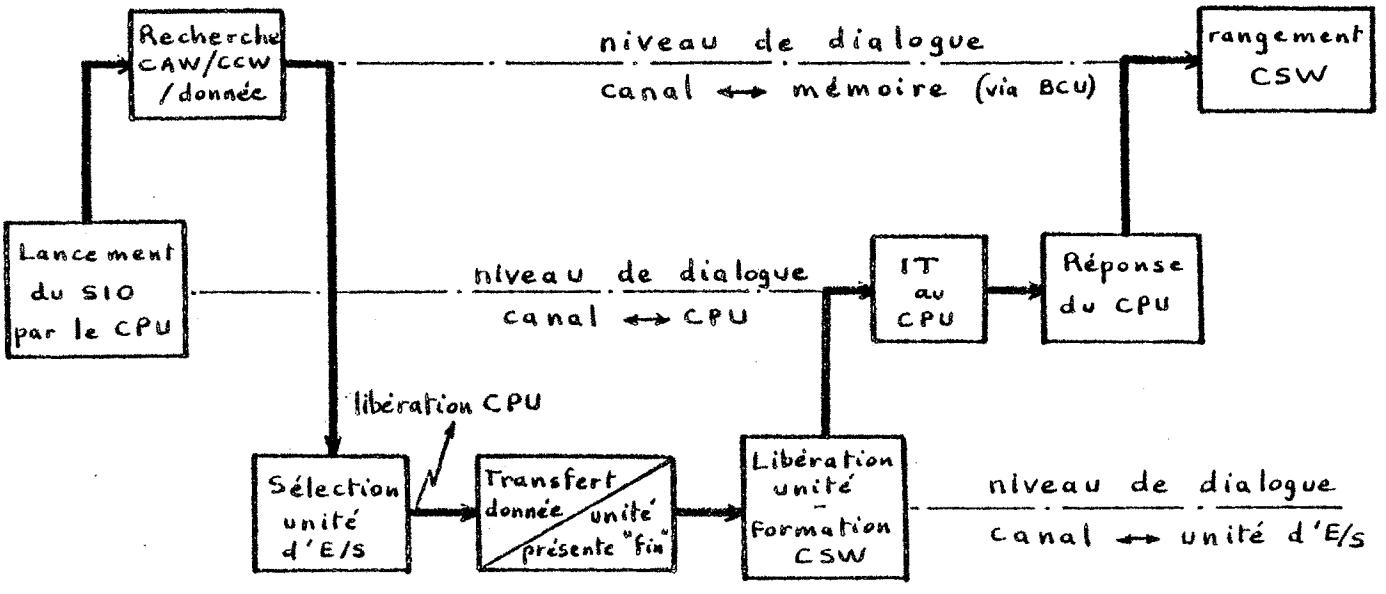


Fig.II-6 Déroulement de l'instruction Start I/O

II-3-1-1 Initialisation

Rappelons que la signification des mots de contrôle permettant l'exécution de toute instruction d'E/S est donnée au paragraphe I-3-5-2 et leur format dans l'annexe 1.

Tout d'abord, le canal est sélecté par le CPU qui lui fournit l'adresse de l'organe d'E/S sur lequel doit avoir lieu l'entrée-sortie et qui l'informe qu'il s'agit d'un SIO. S'il est libre, le canal doit se procurer le CAW dans un emplacement réservé de la mémoire centrale.

Dès que le BCU lui permet cette lecture et s'il ne détecte pas d'erreur dans ce mot ni dans l'adresse de l'organe d'E/S, il lance, dans une deuxième phase, 2 opérations en parallèle :

-1- vis-à-vis de la mémoire centrale : par l'intermédiaire du BCU, il récupère le premier CCW nécessaire à l'échange. Celui-ci est rangé après vérifications.

-2- vis-à-vis de l'unité d'E/S : il commence la Séquence de Sélection initiale (CISS) que nous décrirons ultérieurement.

Remarque : des transferts ont lieu également avec la mémoire locale pour la mise à jour des UCW : avec l'adresse du CCW suivant, les informations contenues dans le CCW courant, la donnée dans le cas d'une écriture, etc...

A ce moment, le canal dispose, d'une part de la commande à envoyer à l'unité et d'autre part de la confirmation d'une sélection correcte avec cette unité, celle-ci lui ayant renvoyé son adresse.

La troisième phase des opérations ne concerne plus que le dialogue entre le canal et l'unité, mais l'unité centrale n'est pas libérée. La séquence de sélection initiale se termine par l'obtention de "l'état initial" de l'unité. Si cet état est nul (unité libre), le canal libère le CPU avec le code condition 0.

Dans le cas où le canal détecte une erreur lors d'une de ces opérations, il peut avoir plusieurs comportements qui sont résumés sur la fig.II-7. La plupart d'entre eux conduisent au positionnement d'un bit du CSW. Dans tous les cas il y a libération du CPU avec le code condition adequat.

Si tout se passe bien on obtient l'organigramme simplifié de la fig.II-8 où n'apparaît pas encore le détail de la séquence de sélection initiale qui nécessite une description préalable de l'interface d'E/S.

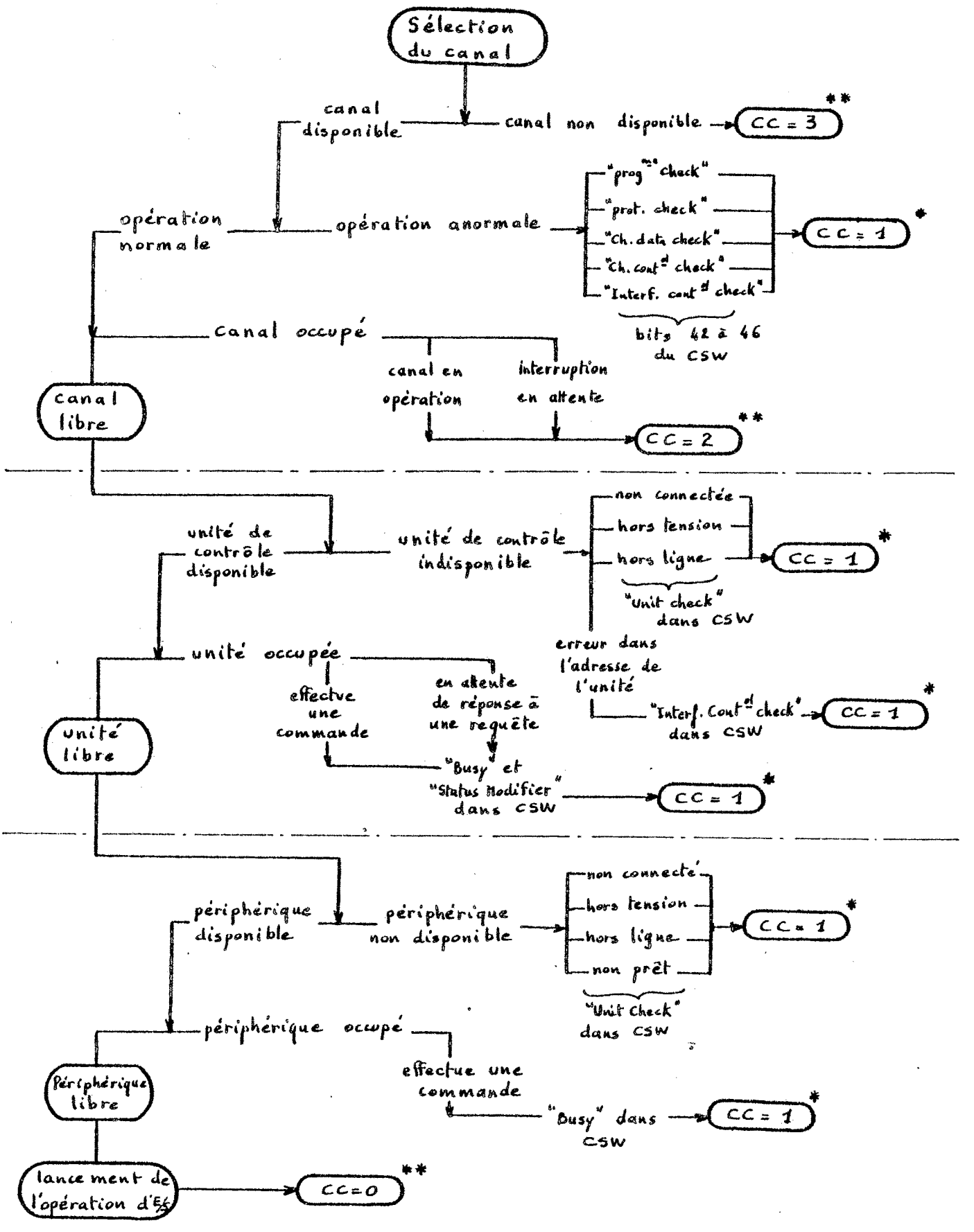


Fig.II-7 Positionnement du code condition lors d'un START I/O

(* : positionnement du CSW, CPU libéré)

(** : CPU libéré)

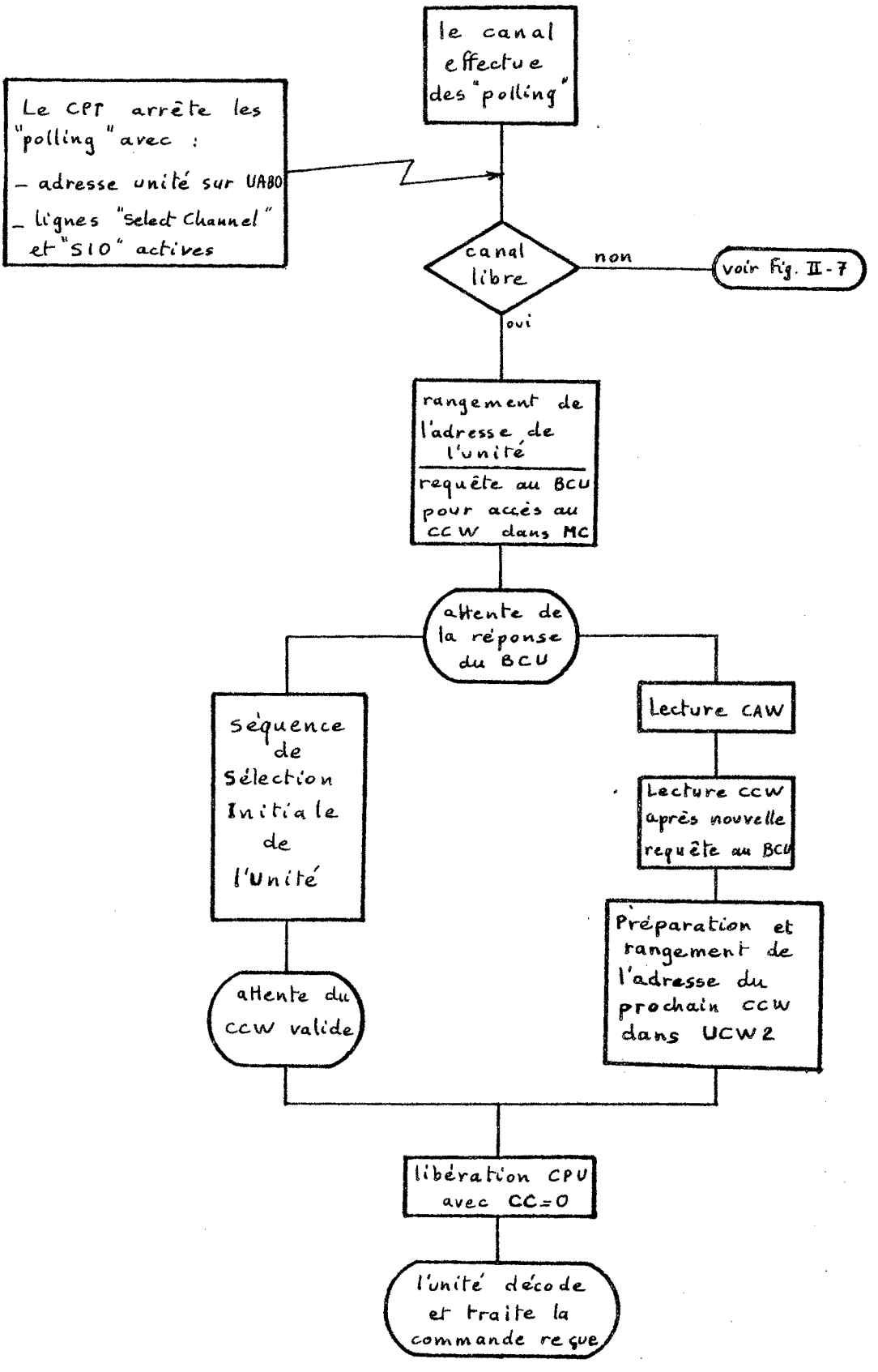


Fig.II-8 Organigramme de l'instruction START I/O

Nous pouvons rappeler les différentes valeurs du code condition résultant de l'initialisation de cette instruction :

CC = 0 : le Start I/O est correctement lancé.
CC = 1 : il y a eu rangement d'un CSW
CC = 2 : le canal ou l'unité est occupé
CC = 3 : le canal ou l'unité est non opérationnel.

II-3-1-2 Exécution de la commande

Le code de la commande est constitué par les bits 0-7 du CCW.
On distingue 3 types de commandes de base :

- sorties (écriture, contrôle)
- entrées (lecture, lecture arrière, analyse)
- branchement (transfert dans programme canal),

dont les codes sont donnés ci-dessous :

code binaire	code hexa de la commande de base (M=0)	commande
x x x x 0 0 0 0	0 0	invalide
M M M M M M 0 1	0 1	écriture
M M M M M M 1 0	0 2	lecture
M M M M M M 1 1	0 3	contrôle
M M M M 0 1 0 0	0 4	analyse
M M M M 1 1 0 0	0 C	lecture arrière
x x x x 1 0 0 0	0 8	transfert dans programme canal

- les bits x sont ignorés
- les bits M sont des "modificateurs" qui servent à étendre le champ d'action de ces commandes. (Exemple : l'écriture avec retour-chariot automatique sur console opérateur est codée 09).

Fig.II-9 Les commandes d'E/S

Nous nous intéresserons seulement aux 2 premiers types (sauf la lecture arrière).

Les commandes de lecture et d'écriture sont explicites. La 1ère adresse de rangement des données et le nombre d'octets à transférer sont fournis par le CCW.

La commande d'analyse permet d'obtenir plus de renseignements sur le déroulement d'une opération que l'état fourni par le CSW. Ces informations supplémentaires sont transmises au canal comme pour une lecture à l'aide d'un ou plusieurs octets selon l'unité d'E/S (ex: 5 octets pour une unité de contrôle de bandes magnétiques).

L'information de base, à peu près généralement utilisée sur tous les types d'unités d'E/S, est contenue dans les 6 premiers bits du premier octet (s'il y en a plusieurs). Elle est indiquée dans la fig.II-10, et la signification exacte de chaque bit est donnée dans l'annexe 1.

(bit de poids fort)

bit n	signification
0	commande rejetée
1	intervention requise
2	erreur de parité sur bus donnée
3	erreur sur matériel
4	erreur sur donnée
5	surcharge

Fig.II-10 Format de l'octet d'analyse

La commande de contrôle permet l'exécution de nombreuses actions au niveau de l'organe d'E/S (positionnement de bras, rembobinage, alarme, modifications du format de sortie des données...) grâce à l'utilisation des bits modificateurs de la commande de base (s'ils sont tous nuls il s'agit en général d'une commande NOP). Pour la plupart des organes d'E/S, le

décodage de cette commande suffit, et aucun transfert n'a lieu. Sinon, l'adresse des données dans le CCW est celle d'une zone où doivent être rangées des informations supplémentaires.

Lors de l'exécution de ces 4 commandes, les 5 bits indicateurs du CCW sont testés, sauf le bit "skip" dans le cas d'une commande de contrôle (voir format du CCW en annexe 1).

Dans le cas d'une commande liée à un transfert effectif de données, une fois qu'elle est acceptée et décodée par l'unité d'E/S, c'est celle-ci qui décide de l'instant de l'échange physique élémentaire à l'aide d'un dialogue du type producteur-consommateur qui peut être représenté par les schémas fonctionnels et temporels de la fig.II-11 <ANC2>.

Ce type de dialogue est d'ailleurs également utilisé lors de l'envoi de l'état de l'unité au canal, de l'envoi de la commande du canal à l'unité et des envois réciproques de l'adresse de l'unité.

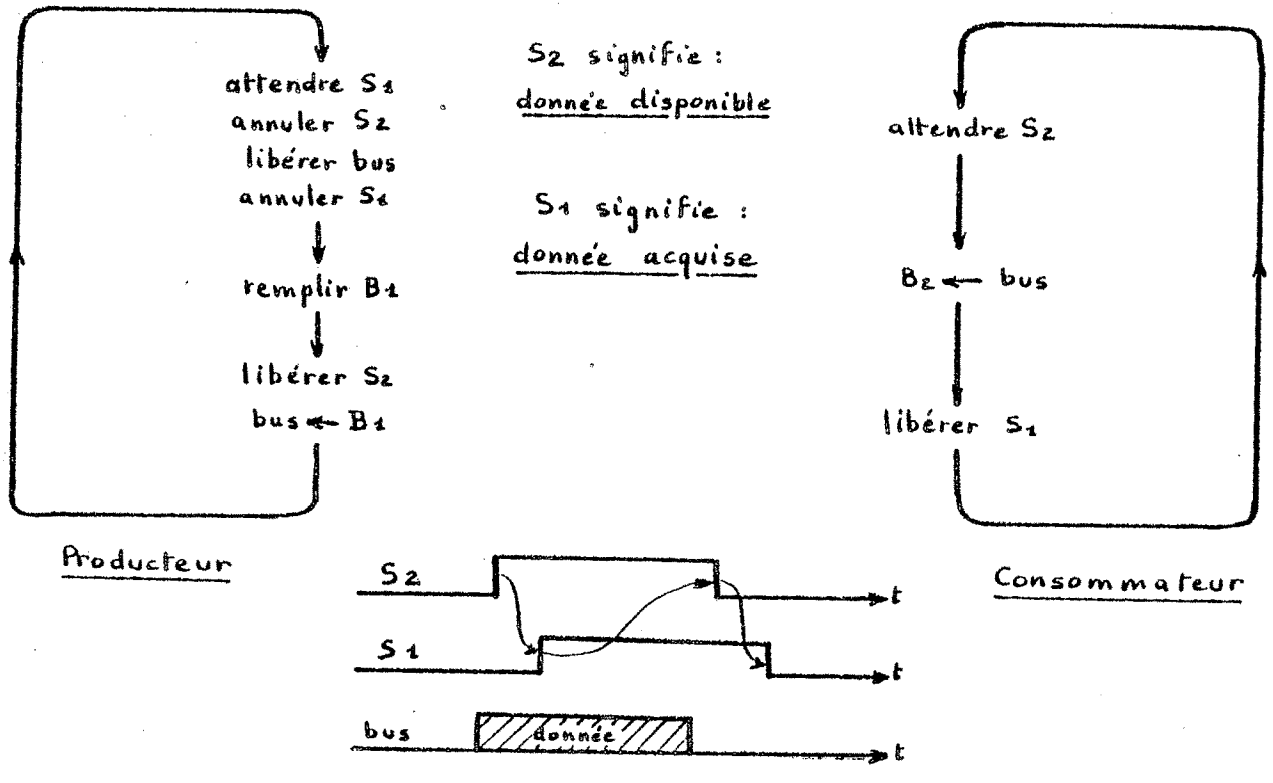


Fig.II-11 Fonctionnement général des dialogues entre le canal et les unités d'E/S

II-3-1-3 Fin de l'instruction

Deux cas principaux sont à distinguer, suivant que la commande nécessitait ou non un transfert effectif de données, auxquels on peut ajouter, sans insister davantage les terminaisons dues à la détection d'erreurs pendant l'initialisation de l'instruction (dont nous avons déjà parlé), et pendant son exécution elle-même par défaillance d'un organe. L'instruction "Halt I/O" (HIO) peut aussi terminer prématurément une instruction SIO (voir $ II-2-4).

a) S'il y a eu un transfert de données, la fin de ce transfert est détectée soit par le canal (passage par 0 du compte dans le CCW), soit par l'unité d'E/S (plus de données à envoyer ou plus de place pour en recevoir), soit par les 2 à la fois.

Dans le premier cas le transfert cesse lors de la prochaine tentative faite par l'unité pour envoyer ou demander un nouvel octet. L'unité doit alors fournir son "état de fin d'E/S" en une ou 2 fois suivant qu'elle peut ou non libérer en même temps le canal (bit "canal occupé" à 1) et l'organe d'E/S lui-même (bit "unité occupée" à 1). C'est néanmoins toujours le canal qui sera libéré le premier.

Dans les 2 autres cas, le transfert cesse de suite et l'unité envoie son état de fin avec les 2 bits positionnés de la même façon.

Tout état contenant un de ces bits à 1, crée une "condition d'interruption" au canal qui doit alors faire une tentative pour interrompre l'unité centrale et lui fournir ainsi le CSW. Il y a lieu de préciser qu'un seul sous-canal à la fois peut générer une condition d'interruption car le canal principal ne contient qu'un seul registre d'interruption (contenant alors l'adresse de l'unité demandeur). Si ce registre est plein, l'état de fin sera mis en attente au niveau de l''unité.

Si le CPU accepte cette demande, le canal range le CSW en mémoire, puis il libère le CPU en lui donnant l'adresse de l'unité, qui apparaîtra dans le code interruption du PSW. Le "basculement" des PSW correspondants (PSW d'E/S) peut alors avoir lieu pour traiter l'interruption.

Dans le cas où il y a eu mise en attente de l'état de fin dans l'unité, dès que le CPU pourra tenir compte de la demande d'interuption du canal, celui-ci forcera l'exécution d'une instruction "Test I/O" (TIO) pour obtenir l'état en attente. Si le CPU envoie une autre instruction, la requête de l'unité sera perdue et celle-ci devra présenter à nouveau son état. Pendant tout ce temps l'unité sera "occupée".

b) S'il n'y a pas eu transfert de donnée, la commande est appelée immédiate et le bit "canal occupé" est positionné lors de la présentation de l'état initial. Bien qu'alors l'octet d'état ne soit pas nul, il ne s'agit pas d'une unité non libre. Mais il n'y a pas non plus de demande d'interruption puisque le CPU n'est pas encore libéré, mais seulement rangement du CSW et positionnement du code condition à 1, puis libération du CPU. L'instruction est terminée.

II-3-2 L'instruction HALT I/O

D'une façon générale elle permet au programmeur d'arrêter un tansfert avant que toutes les données soient échangées. Elle sera utilisée par exemple pour libérer un canal sélecteur afin d'effectuer une opération jugée plus prioritaire, ou contrôler des opérations en temps réel sur un canal multiplexeur ou les transmissions sur une ligne de communication.

Nous donnons à la fig.II-12 le schéma du déroulement simplifié de cette instruction. La séquence qui nous intéresse le plus dans l'optique de la réalisation de l'UCB est celle qui est appelée "interface disconnect" et qui sera explicitée plus loin. Son effet dépend de l'organe d'E/S et de l'opération en cours.

Le code condition représente donc les états suivants reconnus pendant l'exécution de l'instruction :

CC = 0 : une interruption était en attente au canal
CC = 1 : un CSW est formé et rangé
CC = 2 : une opération en mode burst est arrêtée
CC = 3 : canal ou unité non opérationnel.

II-3-3 L'instruction TEST I/O

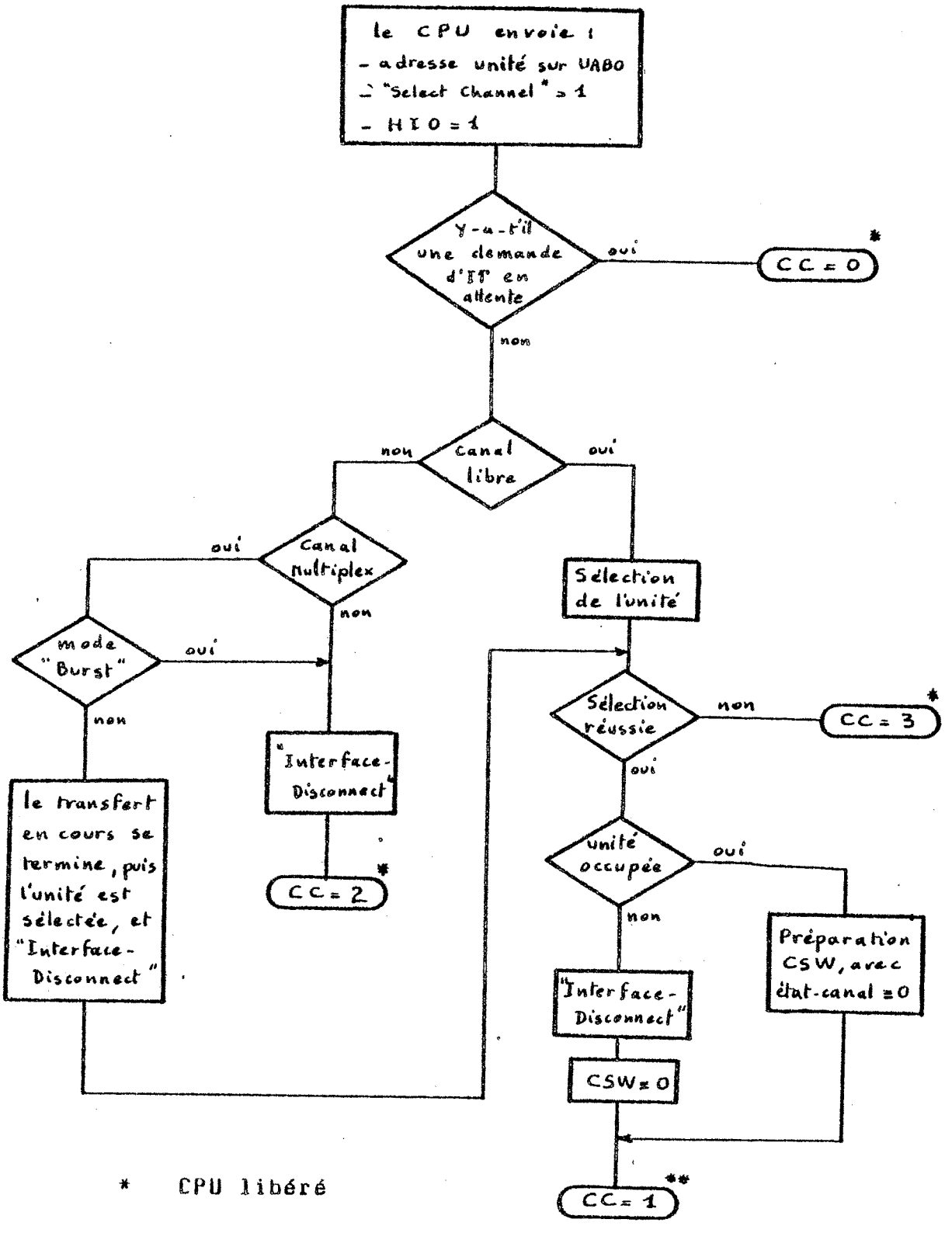
Elle est utilisée surtout pour obtenir un état d'une unité n'ayant pas pu le communiquer normalement lors d'une précédente instruction d'E/S ou lors d'une tentative de présentation d'un état asynchrone, ou tout simplement pour connaître l'état d'un organe d'E/S. Son déroulement simplifié est donné à la fig.II-13.

Un état asynchrone est dû à l'apparition dans l'unité ou dans l'organe d'E/S, d'un évènement qui doit être signalé au CPU. Par exemple le passage d'un périphérique à l'état "prêt", suite à une action de l'opérateur.

Au niveau de l'unité d'E/S, cette instruction se déroule comme un SIO dont le code de la commande serait invalide (tous les bits à 0), et de type immédiat.
Certains bits de cet état peuvent correspondre à des erreurs dans le déroulement du TIO lui-même.

Le code condition positionné a les significations suivantes :

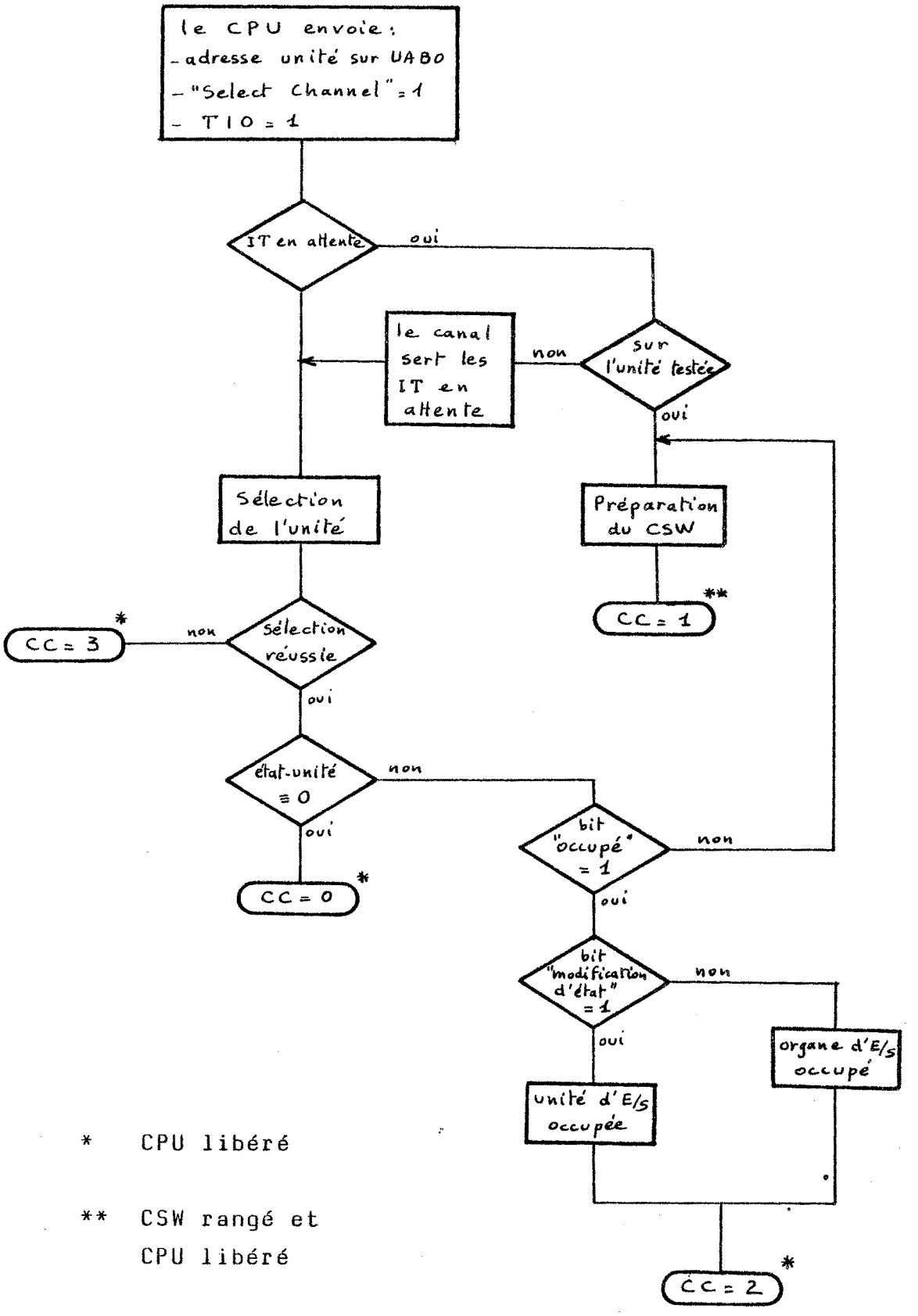
CC = 0 : canal et unité libres
CC = 1 : un CSW est rangé ; l'interruption en attente est supprimée au niveau du canal et de l'unité
CC = 2 : le canal ou l'unité est occupé
CC = 3 : le canal ou l'unité est non opérationnel.



* CPU libéré

** CSW rangé et
CPU libéré

Fig.II-12 Organigramme de l'instruction HALT I/O



* CPU libéré

** CSW rangé et CPU libéré

Fig.II-13 Organigramme de l'instruction TEST I/O

II-3-4 Quelques remarques

Nous avons volontairement passé sous silence un certain nombre de situations parmi celles qui n'étaient pas strictement nécessaires à la réalisation de notre projet, mais dont l'étude serait indispensable par exemple à un programmeur de système d'exploitation.

En particulier nous avons ignoré une facilité importante mise à la disposition de l'utilisateur de ce système et qui est le chaînage de commandes au niveau du programme canal. Il permet à l'unité centrale de lancer plusieurs E/S avec une seule instruction Start I/O. Le canal prévient l'unité, à la fin de l'échange dû à la première commande, qu'une autre va lui être envoyé. L'unité d'E/S doit alors maintenir la liaison avec l'organe d'E/S.

La détection de cette opération est simple, mais elle concerne surtout les unités qui contrôle plusieurs périphériques.

Le canal n'apporte au CPU, le résultat de l'instruction qu'à la fin de toutes les commandes, sauf si une erreur se produit, auquel cas le chaînage est arrêté.

Il a été donné le déroulement des opérations d'E/S les plus significatives, spécifiquement sur le canal multiplexeur des modèles indiqués au début du $ II-2, sans faire une étude comparée du fonctionnement des entrées-sorties par rapport à l'ensemble des possibilités offertes par le système 360 <IBM1>.

Nous avons essayé dans ce paragraphe, de donner les informations nécessaires et suffisantes pour faire la part de ce qui se situe au niveau de la communication canal-CPU et de ce qui concerne plus précisément l'interface d'E/S. Nous allons donc maintenant revenir à ce niveau qui est celui où se passent les échanges physiques élémentaires lors des entrées-sorties.

II-4 L'INTERFACE D'ENTREE-SORTIE <IBM2>

II-4-1 Description générale

La fig.II-14 donne la configuration générale des 34 lignes constituant l'interface d'E/S. Ces lignes (sauf 2) sont envoyées en parallèle à toutes les unités de contrôle (8 au maximum) mais elles ne sont significatives que pour une unité à la fois, grâce à un mécanisme d'interblocage.

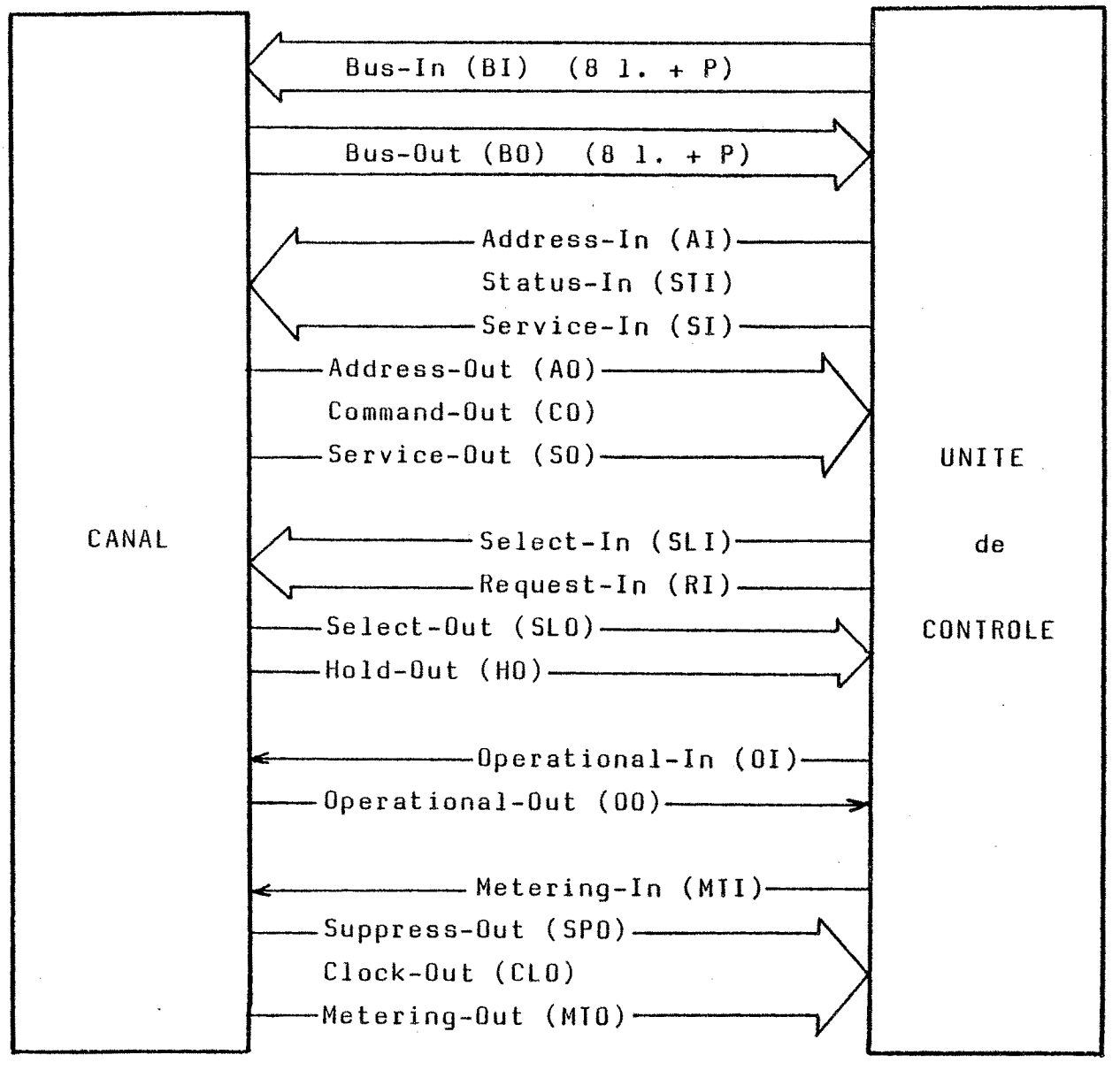


Fig.II-14 Les lignes de l'Interface d'E/S du Système IBM 360

Nous avons gardé les appellations originales de ces lignes pour éviter des traductions parfois délicates et inutiles... De plus, nous indiquons les abréviations utilisées.

Ces lignes sont regroupées en 5 types :

II-4-1-1 Les Bus ("Bus-Out" et "Bus-In")

Ce sont 2 groupes de 8 lignes + 1 ligne de parité (impaire) qui servent à véhiculer adresses, commandes, états, données. La distinction entre ces informations est assurée par les "lignes d'identification".

II-4-1-2 Les lignes de sélection ("scan controls")

Elles sont destinées à établir le dialogue entre le canal et l'unité (voir plus précisément le rôle de SLO, SLI, HO dans le $ II-4-2). Ce dialogue peut être initialisé par le canal (avec SLO) ou par l'unité (avec RI).

- SLO et SLI sont les 2 lignes qui sont envoyées en série.
- RI peut être activé par plus d'une unité à la fois.
- HO valide, pour toutes les unités en même temps, le signal SLO et permet ainsi que les unités reçoivent plus rapidement l'information de la chute de ce signal. Ceci impose par contre de tester l'intersection logique de ces 2 signaux pour affirmer la présence de SLO.

II-4-1-3 Les lignes d'identification ("tags")

Elle permettent de savoir quelle sorte d'information se trouve sur chaque bus.

Les "Tags-In" (TI) indiquent que l'information sur "Bus-In" (BI) est une adresse (AI), un état (STI) ou une donnée (SI).

Les "Tags-Out" (TO) précisent l'information en sortie sur "Bus-Out" (BO) : c'est une adresse (AO), une commande (CO), une donnée (SO).

Dans les 2 cas ces lignes sont mutuellement exclusives et sont maintenues jusqu'à ce qu'une autre ligne (de sens contraire) leur réponde.

Certaines lignes peuvent avoir plusieurs significations selon la séquence dans laquelle elles sont utilisées ; par exemple CO peut vouloir dire : "arrêtez", "conservez l'état", "allez-y", "voici une commande"...

II-4-1-4 Les lignes d'interblocage

Elles valident les lignes précédentes et confirment la sélection d'une seule unité (OI).

Celle qui est émise par le canal (OO) valide toutes les autres lignes de l'interface (sauf SPO) et permet donc en plus de ré-initialiser une unité d'E/S ou tout le système (voir "Reset").

II-4-1-5 Les lignes de contrôle

Nous ne nous intéresserons par la suite qu'à SPO qui apparaît dans différentes séquences telles que la suppression de donnée, d'état ,le chaînage de commandes, les "reset"...

CLO, MTO, MTI sont utilisées en rapport avec des mesures du temps du CPU.

II-4-2 Le mécanisme d'interconnexion

La fig.II-15 montre comment l'ordre des priorités parmi plusieurs unités connectées à l'interface d'E/S est assuré par la propagation de la ligne SLO à travers les différentes unités.

On remarquera que SLO change de nom après le bloc terminal adaptateur d'impédances, et devient SLI, formant ainsi une boucle de retour au canal, qui peut être coupée dès qu'une unité veut s'assurer les services du canal.

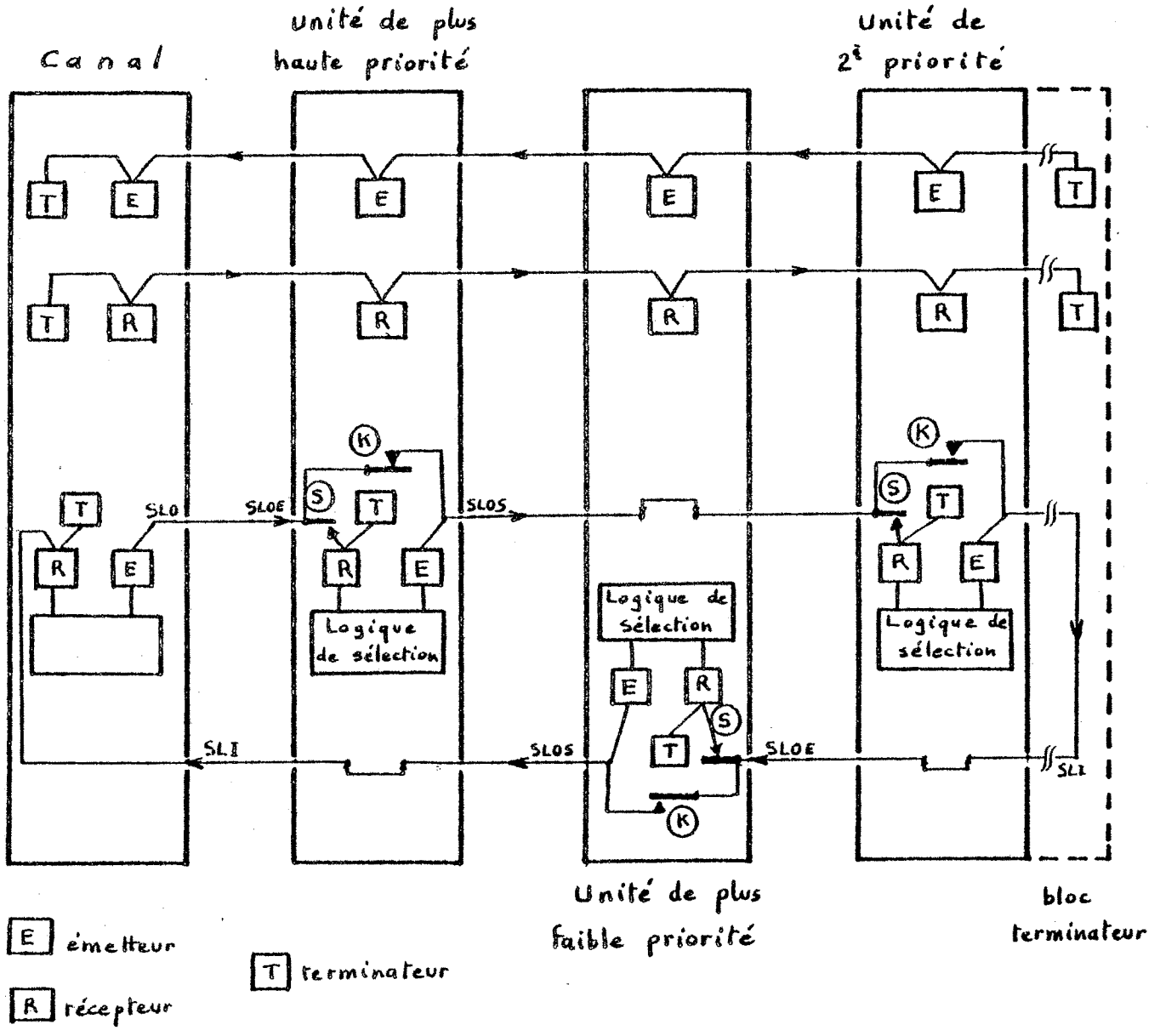


Fig.II-15 La connexion des unités sur l'Interface d'E/S

D'autre part, on constate qu'une unité peut aussi bien être installée sur la ligne SLI que SLO et que ce n'est pas sa position géographique qui définit sa priorité. On appellera par la suite SLOE (SLO entrant) et SLOS (SLO sortant), l'entrée et la sortie de cette boucle dans l'unité sans se soucier de savoir s'il s'agit effectivement de SLO ou de SLI.

Enfin, toutes les unités doivent fournir un chemin direct à SLO si elles ne sont pas en service. C'est le rôle des 2 contacts Ⓢ et Ⓚ.

II-4-3 Le "Polling"

Il s'agit de l'opération qui permet au canal, s'il n'est pas sélecté par l'unité centrale et si la ligne RI est active, de faire un balayage des unités pour savoir celle qui a une requête à satisfaire (ou la plus prioritaire s'il y en a plusieurs en même temps).

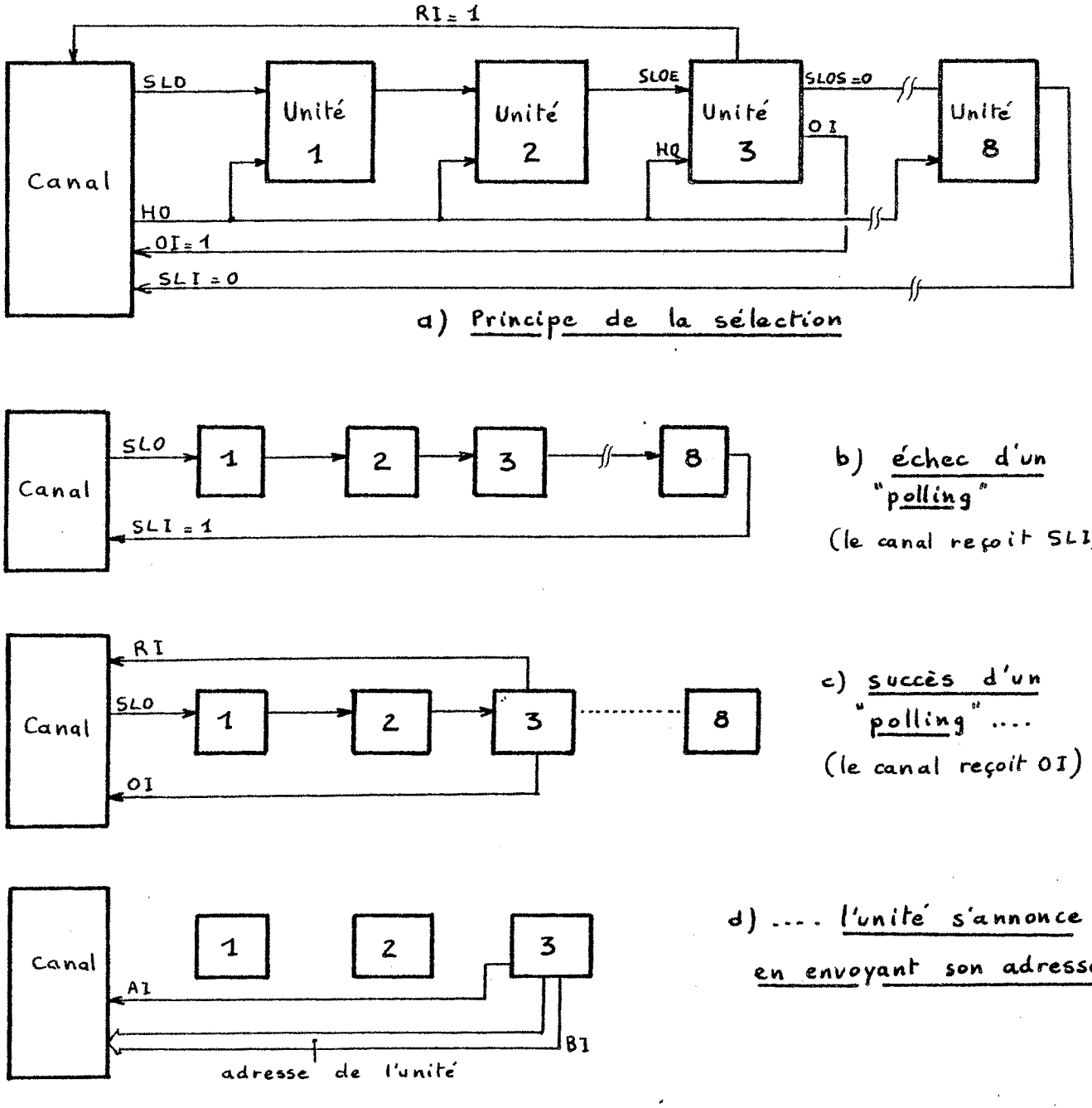


Fig.II-16 Les schémas fonctionnels du "Polling"

La fig.II-16 schématise cette opération en supposant que c'est l'unité 3 qui a positionné RI. Les unités 1 et 2 propagent le signal SLO tandis que l'unité 3 le bloque, mais elle active alors la ligne OI qui referme la boucle, prévenant ainsi le canal que son balayage, matérialisé par la propagation de SLO, a réussi. Le signal AI et l'adresse sur Bus-In préciseront de quelle unité il s'agit.

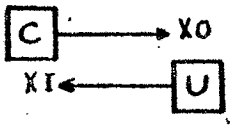
On fait souvent correspondre l'unité la plus rapide à la position la plus prioritaire pour que ses requêtes soient servies le plus vite possible.

II-4-4 Les séquences régies par l'interface d'E/S

Nous allons d'abord étudier des séquences simples réalisées par les lignes précédemment définies :

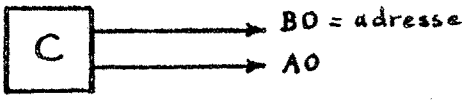
II-4-4-1 Les séquences de contrôle

Nous les représenterons par les diagrammes fonctionnels suivants, dans lesquels :

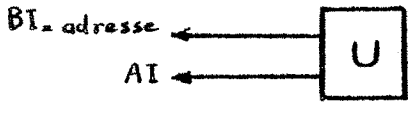


indique que le canal active la ligne XO
indique que l'unité active la ligne XI

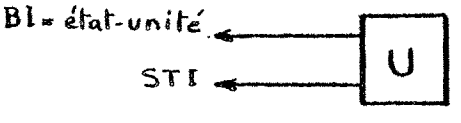
a) Sélection :



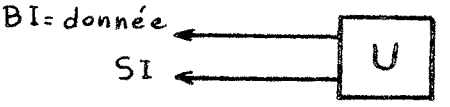
b) Adresse acceptée, départ :



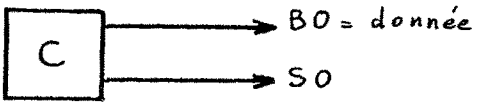
c) Etat accepté :

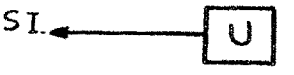


d) Donnée acceptée :

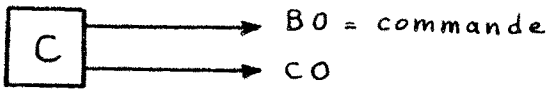


e) Donnée prête :



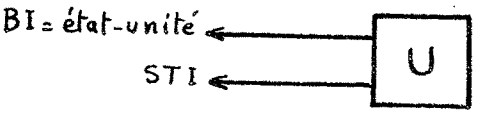


f) Commande prête :

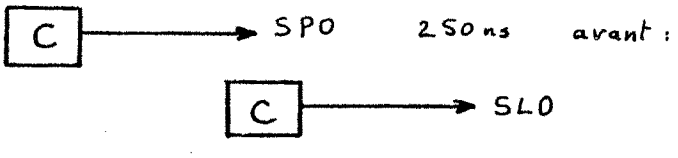


g) Stop :

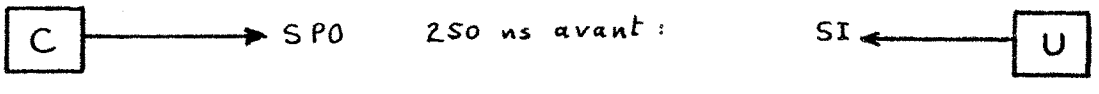
h) Etat refusé :



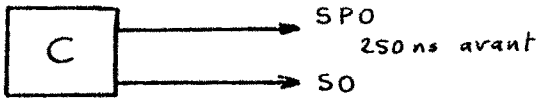
i) Suppression d'état :

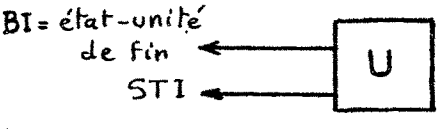


j) Suppression de donnée :

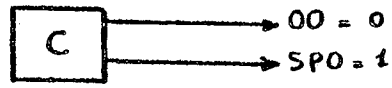


k) Chaînage de commande :

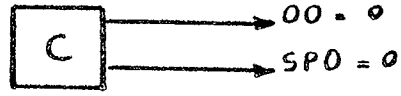




l) Reset sélectif :



m) Reset général :



n) "Interface disconnect" :

II-4-4-2 Les séquences d'interface

Il y a 3 séquences essentielles composées de la succession de certaines des séquences de contrôle précédentes :

- la séquence de sélection initiale (CISS) qui commence dès la reconnaissance d'une instruction SIO (ou TIO) par le canal et qui se termine par l'envoi de l'état initial de l'unité (fig.II-17).

- la séquence de sélection par l'unité (CUIS) consécutive à l'activation de la ligne RI, suivie par un polling (fig.II-18).

- la séquence courte d'unité occupée (CUBS) par laquelle une unité peut prévenir très vite le canal qu'elle n'est pas en mesure de recevoir de commande (fig.II-17).

Dans cette séquence on n'a pas de vérification, par le canal, de l'adresse de l'unité ; elle n'offre donc pas la sûreté des 2 autres. D'autre part, elle ne doit pas être utilisée pour un organe d'E/S qui vient de recevoir un chaînage de commande.

Nous montrerons le déroulement temporel de ces séquences en tant que résultats obtenus lors de la réalisation de l'UCB que nous allons décrire maintenant.

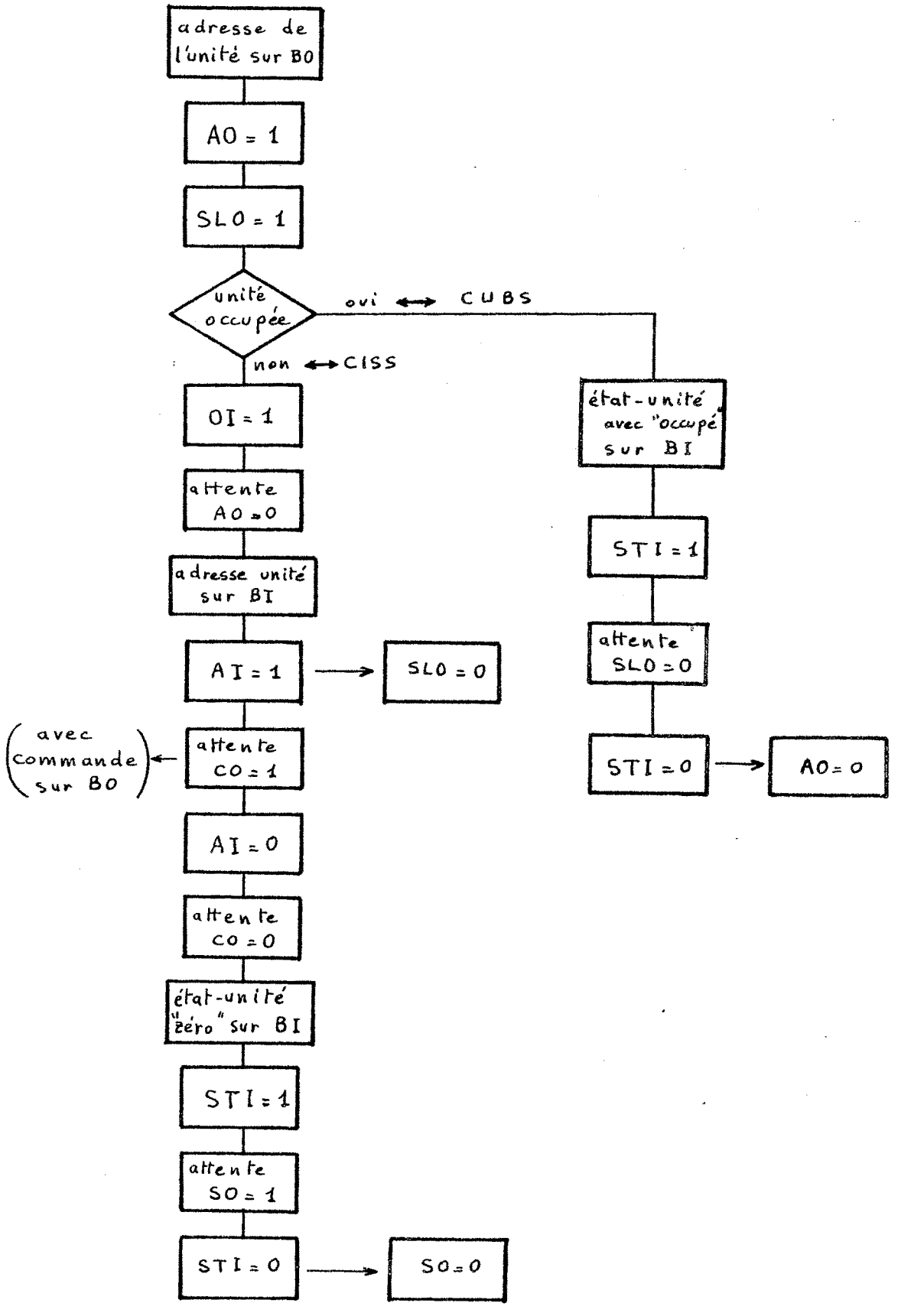


Fig.II-17 Les Séquences de Sélection Initiale (CISS)
et de Réponse Courte d'Unité Occupée (CUBS)

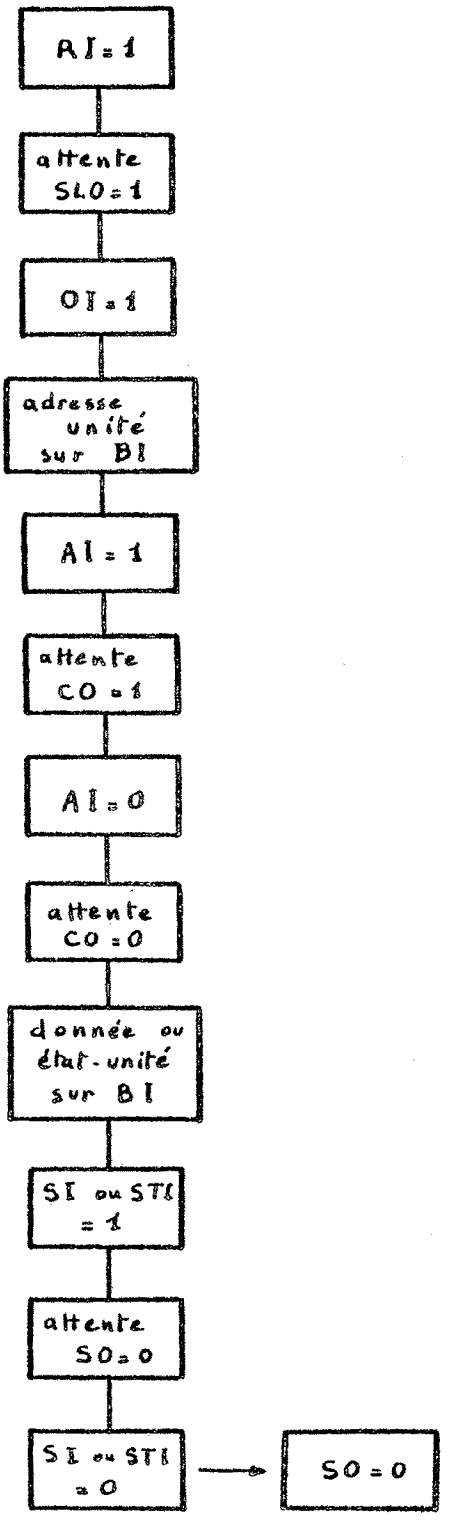


Fig.II-18 La Séquence de Sélection par l'Unité (CUIS)

CHAPITRE III

PRESENTATION DE L'UCB

III-1 INTRODUCTION

III-2 CARACTERISTIQUES GENERALES DE L'ARCHITECTURE DE L'UCB

III-2-1 Les contraintes
- III-2-1-1 Par rapport au canal
- III-2-1-1 Par rapport au microprocesseur

III-2-2 Les solutions adoptées
- III-2-2-1 Les fonctions câblées
- III-2-2-2 Les fonctions programmées

III-2-3 Les 3 modules fonctionnels

III-2-4 Les 2 logiciels

III-3 LES MODULES FONCTIONNELS

III-3-1 La partie contrôle
- III-3-1-1 Le CPU
- III-3-1-2 La mémoire de programme
- III-3-1-3 Le circuit de génération des commandes internes
- III-3-1-4 La mémoire de travail
- III-3-1-5 Les états internes

III-1 INTRODUCTION

Après une première réalisation (annexe 4) à l'aide d'un microprocesseur standard dont l'infrastructure (assembleur, superviseur...) était opérationnelle et qui a permis une vérification du déroulement des séquences précédemment décrites, il a paru nécessaire, devant les résultat obtenus, d'augmenter la vitesse des échanges avec le canal.

Le choix du microprocesseur s'est alors porté sur un des plus rapides de l'époque, réalisé en technologie bipolaire : le 8X300 de Signetics-RTC, dont le temps d'exécution d'une instruction de 16 bits est de 250 ns. Ces instructions sont en nombre restreint mais orientées vers la manipulation de bits, ce qui convenait à l'utilisation envisagée. On trouvera en annexe 2 ses principales caractéristiques.

A microprocesseur non standard : architecture spécialisée.

Pour nos besoins de rapidité dans l'échange des données, nous avons été amenés à générer certaines commandes en décodant une partie de l'instruction préalablement étendue à 32 bits. D'autre part, certaines fonctions vis-à-vis du canal ont été réalisées par logique câblée telle que, par exemple, la reconnaissance de l'adresse de l'unité pour propagation du signal de sélection (SLO), et d'autres par programmation (parfois par nécessité).

Ce travail a été mené à bien en collaboration avec Monsieur R.Saettone afin de réaliser une application immédiatement utilisable de l'UCB : soit le "Module Interface Canal" du Système PIAR, mis au point par ce chercheur (voir Chapitre IV).

Les annexes 5 et 6 décrivent les outils qui ont été nécessaires au bon déroulement de cette réalisation.

III-2 CARACTERISTIQUES GENERALES DE L'ARCHITECTURE DE L'UCB

III-2-1 Les contraintes

III-2-1-1 Par rapport au canal

Précisons que l'UCB est destinée plus particulièrement à être installée sur un canal multiplexeur et qu'elle a été plus précisément testée en étant connectée à l'interface d'E/S du canal multiplex, mais que ceci n'est pas une contrainte puisque l'interface est normalisé comme nous l'avons déjà dit.

Mise à part la nécessité de prévoir l'enchaînement correct des séquences, on peut relever 2 types de contraintes :

- temporelles : elles sont destinées soit à assurer des performances optimales au canal, soit à éviter le blocage du système par suite d'une défaillance d'un organe d'E/S. En effet les séquences de contrôle décrites au $ II-4-4-1 sont indépendantes du temps et ne sont régies que par la réponse de l'autre ligne.

Parmi les plus importantes on peut citer :

a) le temps de propagation du signal SLO si l'unité ne se connecte pas est normalement de 600 ns, et devra être au maximum de 1,8 µs.
b) un transfert de données sera dit en mode multiplex seulement s'il dure moins de 32 µs.
c) l'information émise par l'unité sur Bus-in doit être valide au plus tard 100 ns après la montée de la ligne TI correspondante et doit le rester jusqu'à la montée de la ligne TO correspondante.
d) toutes les lignes venant de l'unité doivent être descendues 100 ns après la chute de OI (sauf RI et MTI).
e) les séquences de suppression de donnée, d'état et d'indication de chaînage de commande, imposent également des contraintes temporelles indiquées lors de leur définition au $ II-4-4-1.

f) lors d'une déconnexion forcée par le canal (instruction HIO par ex.) OI doit tomber 6 µs après la détection de cette séquence.

g) lors d'un "reset" (sélectif ou général) toutes les lignes de l'interface venant de l'unité doivent être mises à 0 dans les 1,5 µs qui suivent la chute de 00.

- électroniques : sans entrer dans les détails des valeurs à respecter, signalons que ce sont essentiellement des contraintes sur la validité des niveaux haut et bas des signaux, sur les résistances des câblages internes et externes, sur les impédances de charge des câbles, sur la précision des résistances des blocs terminateurs des lignes (SLO seulement pour nous)...

Ces problèmes seront résolus généralement par l'utilisation de câbles et de connecteurs spécifiés par IBM et par l'emploi de circuits d'interface compatibles avec les spécifications IBM, en particulier les émetteurs et récepteurs de lignes.

III-2-1-2 Par rapport au microprocesseur

Ce sont essentiellement :

a) la présence d'un bus adresse spécifique

A la différence de la plupart des microprocesseurs, le cheminement des instructions se fait d'une façon complètement séparée de celui des données. Ces dernières se trouvent alors sur un bus bi-directionnel : l'IV-BUS, en concurrence avec les adresses des différentes portes d'E/S, et leur transfert nécessite 2 instructions : une pour l'adressage du circuit gestionnaire, une autre pour le transfert lui-même.

b) l'absence de mécanisme d'interruption

Ceci obligera à prévoir une gestion programmée de l'occurence de certains événements asynchrones. Ce sera le cas par exemple de la détection de "l'interface disconnect".

III-2-2 Les solutions adoptées

De ces diverses contraintes vont découler certains choix quant à la réalisation des différentes fonctions.
On distinguera donc :

III-2-2-1 Les fonctions câblées

Ce sont :

- la reconnaissance de l'adresse de l'unité parmi les 256 qui peuvent être connectées au canal multiplexeur
- la propagation de SLO
- la sélection des boitiers d'E/S à partir d'une extension de l'instruction
- la génération des lignes TI : on appellera à partir de maintenant Tag-in (TI) et Tag-out (TO) les lignes, autres que les Bus, respectivement qui entrent et sortent du canal
- la détection des "reset"
- la génération et la vérification du bit de parité
- la génération de 2 lignes de synchronisation rapide destinées à l'interface avec l'utilisateur de l'UCB.

Pour les 2 premières fonctions citées, ce choix résulte d'une part de l'absence d'interruption du 8X300 et d'autre part de la contrainte temporelle liée à la propagation de SLO à travers l'unité.

En ce qui concerne les troisième et quatrième fonctions, il fallait satisfaire les performances optimales du canal et éviter le recours à 2 instructions pour la réception et l'envoi des informations. D'autre part, on peut considérer qu'elles ne sont pas purement câblées puisqu'elles sont liées à l'obtention d'une instruction de la mémoire de programme comme on le verra plus loin. Néanmoins elles sont indépendantes du traitement interne de cette instruction par le CPU, puisqu'issues d'un décodage câblé.

Les "reset" étant complètement asynchrones et ne nécessitant qu'une ré-initialisation de l'unité, il était plus simple (et suffisant) de les câbler en utilisant la broche RESET du 8x300.

Le choix du câblage des sixième et septième fonctions est évident : circuits spécialisés, bascule.

III-2-2-2 Les fonctions programmées

Le fait de ne pas réaliser l'UCB de façon entièrement câblée n'est évidemment pas à considérer comme un inconvénient puisque cela apporte une simplicité de mise en oeuvre et une souplesse de fonctionnement et d'adaptation à différentes applications.

Nous avons assuré ainsi la majorité des fonctions nécessaires à la connexion avec le canal et envisagé quasiment toutes celles concernant la gestion d'une application quelconque :

- génération de toutes les séquences d'interface
- gestion de quelques états internes et des états-unité à conserver en cas de refus momentané du canal
- décodage et exécution des commandes
- déroulement complet des transferts des données avec le canal et le dispositif utilisateur
- reconnaissance et envoi au canal d'un état-unité asynchrone
- choix du mode de fonctionnement utilisé
- détection de l' "Interface Disconnect".

En ce qui concerne le dernier point, il faut signaler que c'est là que l'absence de mécanisme d'interruption est la plus sensible puisque cette opération est asynchrone mais ne peut se traduire seulement par une ré-initialisation de l'unité.

Nous envisagerons, au cours du Chapitre VI, la réalisation câblée d'un tel mécanisme pour le 8X300.

Remarque : nous appellerons maintenant "état-unité" un des états définis par IBM ("status") pour le distinguer de l'état interne de l'UCB.

III-2-3 Les 3 modules fonctionnels

Nous pouvons maintenant donner la structure générale adoptée pour l'UCB en tenant compte cette fois d'une décomposition en 3 modules fonctionnels, selon la fig.III-1 ci-dessous :

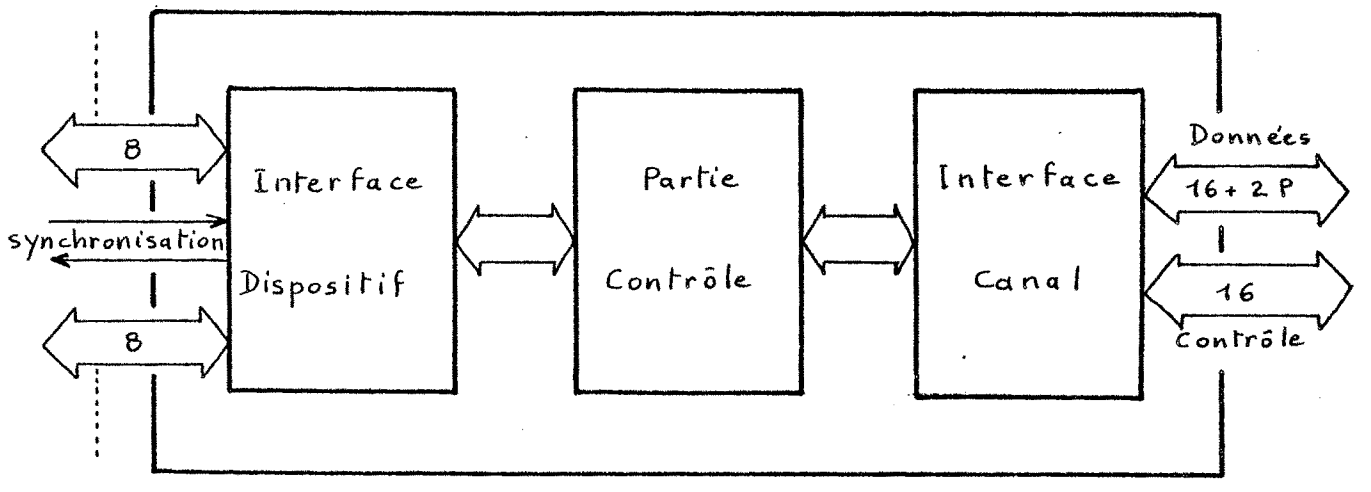


Fig.III-1 Représentation simplifiée de l'UCB

La partie contrôle assure le traitement de toutes les informations :

- séquencement des lignes de contrôle (TI, TO)
- transit des données
- mise à jour des états internes de l'UCB ainsi que de ceux du dispositif.

L'Interface canal assure l'émission et la réception des lignes de contrôle et de données du canal en respectant les spécifications IBM concernant les niveaux, les impédances, etc...

On y trouve :

- la génération des Tags-In (8 lignes) et de Bus-Out (8 lignes) et la génération du bit de parité impaire (circuit spécialisé)
- la réception des Tags-Out (8 lignes) et de Bus-Out (8 lignes) et la vérification du bit de parité (circuit spécialisé)
- la reconnaissance de l'adresse de l'unité et la propagation ou le blocage de SLO.

L'Interface Dispositif se compose principalement :

- d'au moins 24 lignes bi-directionnelles progammables et extensibles selon le dispositif à connecter au canal
- de 2 lignes de synchronisation câblée.

III-2-4 Les 2 logiciels

Deux tâches essentielles se partagent la gestion de l'unité : les programmations des 2 interfaces. L'inter-communication entre ces 2 tâches se fait selon un mécanisme de tests imbriqués de variables propres à chacun des 2 traitements (chacune teste en fait une variable de l'autre).

Ces 2 logiciels sont implantés, sous forme modulaire, dans 2 mémoires mortes différentes (des PROM dans notre situation).

Celui qui gère l'Interface Canal peut être considéré comme figé, à l'exception d'un décodage et d'un traitement supplémentaires dus à l'extension éventuelle des commandes de base (en particulier la commande "contrôle") pour un dispositif qui le nécessiterait.

Ils effectuent donc les fonctions programmées définies au paragraphe précédent.

III-3 LES MODULES FONCTIONNELS

La fig.III-3 donne l'architecture simplifiée de l'UCB ainsi que son raccordement à l'interface d'E/S. Nous allons maintenant décrire cette architecture en insistant plus particulièrement sur les fonctions câblées citées précédemment.

III-3-1 La partie Contrôle

Les schémas complets sont donnés dans les planches 1 à 4 de l'annexe 3 ; cependant, nous voyons sur la fig.III-3 qu'elle se décompose en 5 blocs fonctionnels, qui sont :

III-3-1-1 : Le CPU

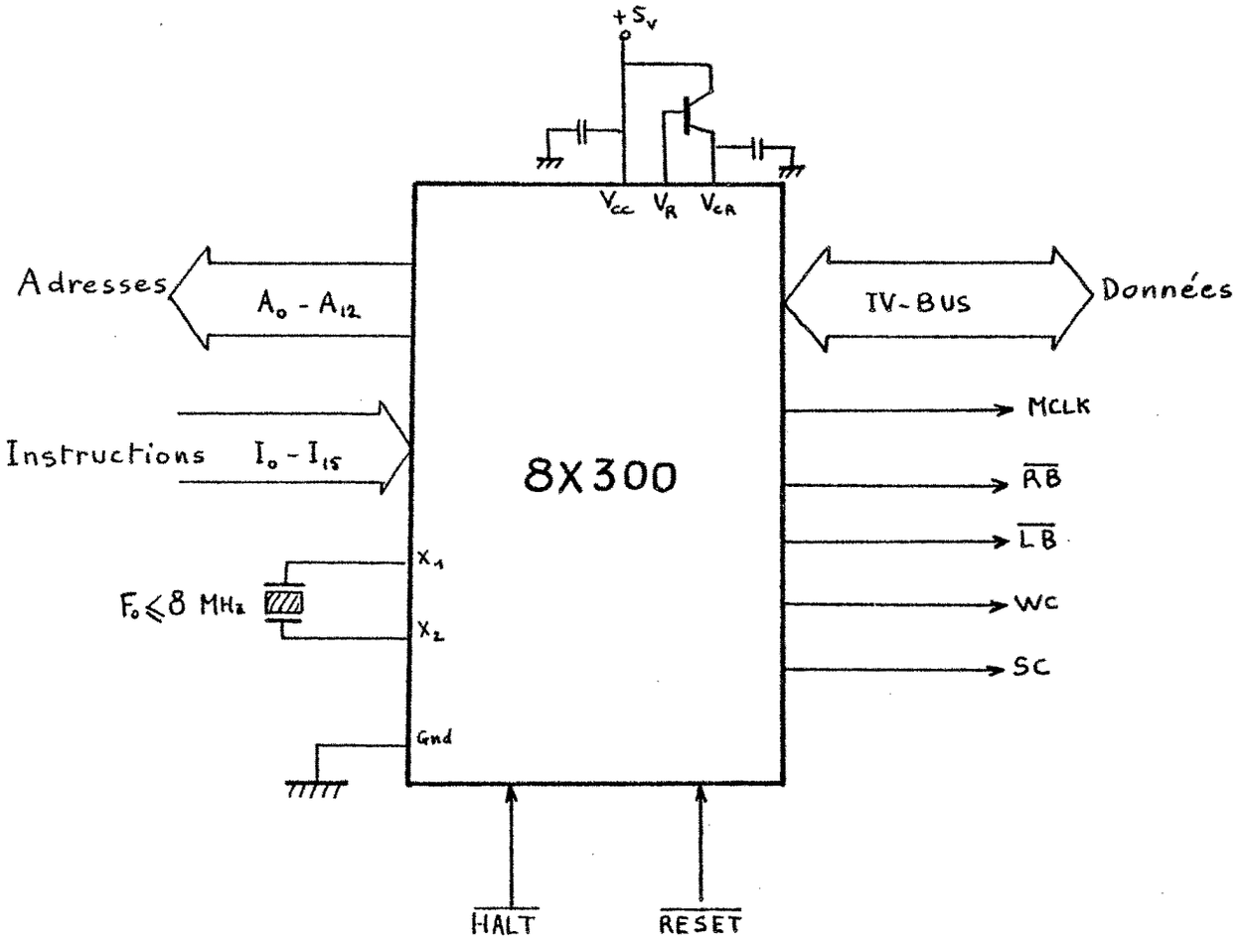


Fig.III-2 Le microprocesseur 8X300 Signetics-RTC

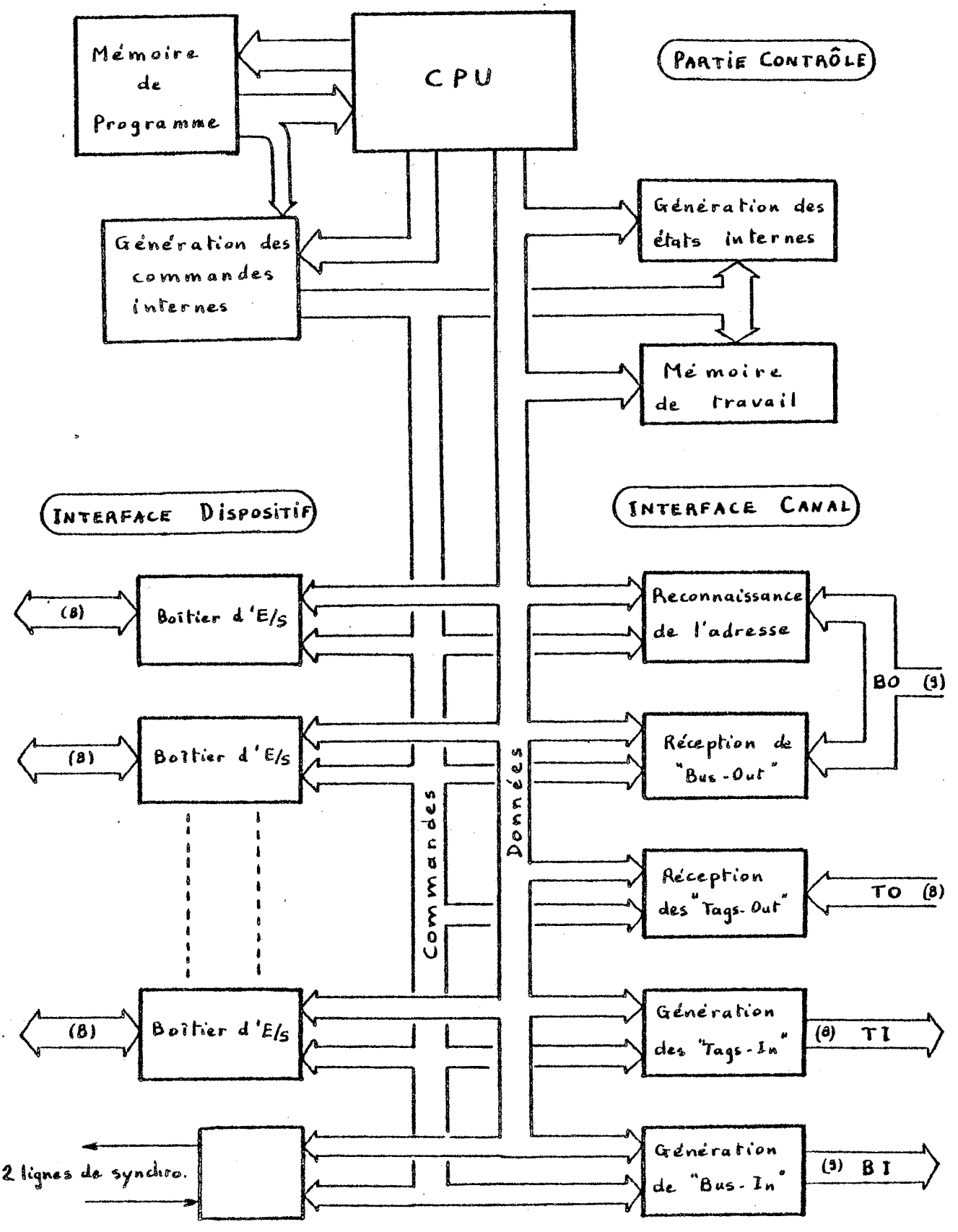


Fig.III-3 Représentation fonctionnelle de l'UCB

En plus des caractéristiques déjà citées et sans en faire une description complète (voir annexe 2), nous donnerons ici quelques précisions nécessaires à la compréhension de la réalisation des fonctions cablées.

Une horloge (MCLK : fréquence max. 8 MHz) fournit la synchronisation interne et externe ; la période correspond à l'exécution d'une instruction et est décomposée en 2 phases (fig.III-4) elles-mêmes divisées en 2.

L'instruction est disponible au cours du premier quart du cycle (à un instant défini par le temps d'accès à la mémoire de programme) et elle reste valide jusqu'à la moitié du cycle ; l'adresse de l'instruction suivante étant envoyée au cours du troisième quart du cycle.

Le Bus IV ("Interface Vector") est un bus bi-directionnel "trois états" relié à tous les circuits du système (registres externes, boitiers d'E/S, mémoire additionnelle...). Ces circuits sont groupés en 2 "bancs" (gauche et droit) adressables par les instructions dont l'IV-Bus est un opérande

Les lignes de contrôle $\overline{LB}$ et $\overline{RB}$ permettent de sélecter un des 2 bancs ($\overline{LB}$ = 0 : banc gauche ; $\overline{RB}$ = 0 : banc droit). Chaque banc étant accessible séparément on peut donc connecter 2 fois 256, soit 512 circuits d'E/S ou mots-mémoire.

Les lignes de contrôle SC et WC ne peuvent être activées que pendant la phase de sortie et elles précisent le type de l'information disponible sur le bus (SC : adresse ; WC : donnée).

L'entrée $\overline{RESET}$ (active niveau bas) permet l'initialisation du CPU. En ce qui nous concerne, celle-ci est déclenchée lors des séquences de Reset du canal par la fonction suivante, issue de la fig.III-5 :

$$\overline{RESET} = OO + SPO \cdot \overline{SEL}$$

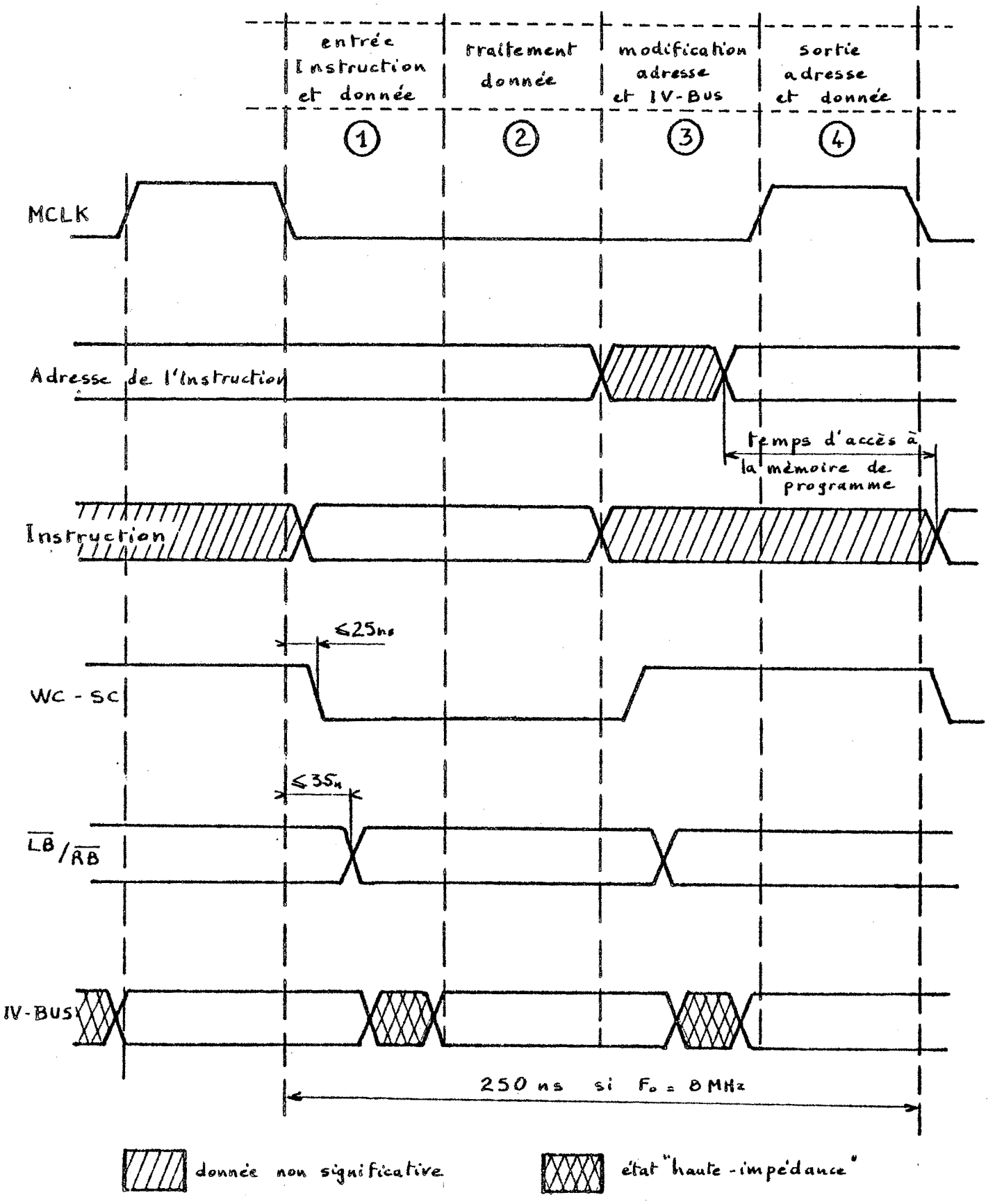


Fig.III-4 Le cycle d'instruction du 8X300

En effet, on peut résumer ces séquences dans le tableau suivant :

		OO	SPO	SEL	RESET	
Reset général		0	0	x	0	niveau actif
Reset sélectif	UCB sélectée	0	1	1	0	
	UCB non -	0	1	0	1	

Fig.III-5 Tableau de vérité de la fonction RESET

III-3-1-2 La mémoire de programme

C'est une PROM (mémoire morte programmable par l'utilisateur) qui comprend actuellement 512 mots de 32 bits et qui peut être étendue à 8 Kmots.

Peu de chose à dire sur cette mémoire, puisque le microprocesseur possède les bus d'adresse et de donnée correspondants, si ce n'est que pendant la phase de mise au point des programmes, elle était remplacée par une mémoire vive de même taille gérée par le système MADAM (voir annexe 6).

III-3-1-3 Le circuit de génération des commandes internes

Il fournit, à partir du décodage d'une extension de l'instruction :

- 12 commandes pour l'Interface Canal
- 5 commandes pour l'Interface Dispositif
- 1 commande pour la partie contrôle elle-même (6 bits d'extension non utilisés permettent de pouvoir disposer de 64 commandes supplémentaires au moins).

L'échantillonnage des bits de l'extension a lieu pendant le premier quart de cycle (par MCLK retardée) et le décodage nécessite que l'instruction en cours active la ligne $\overline{LB}$, donc que le banc gauche soit adressé (adresses de 10 à 17).

L'extension est découpée en 4 champs : les deux premiers, de 3 bits sont décodés et les 2 suivants (3 et 1 bits) fournissent directement l'adresse et la valeur immédiate pour positionner un parmi les 8 tags-in.

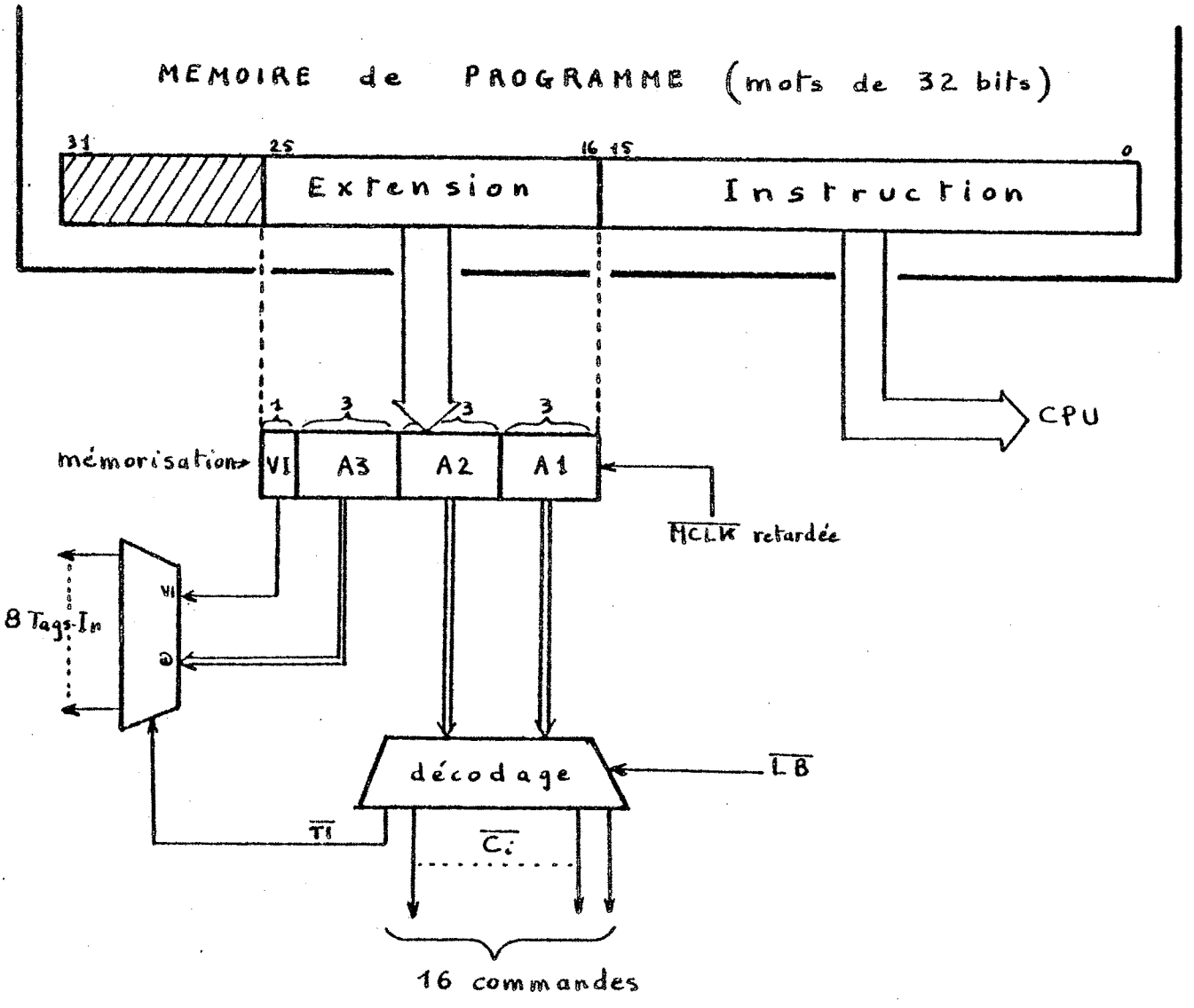


Fig.III-6 Décodage de l'extension
(le circuit complet est en annexe 3, planche 2)

La sélection de chaque circuit concerné par ces commandes se fait donc dès la première phase de l'instruction s'ils sont utilisés en entrée (contrainte de temps la plus serrée).

Ces circuits fournissent alors leurs informations à l'instant requis par le CPU (fig.III-7).

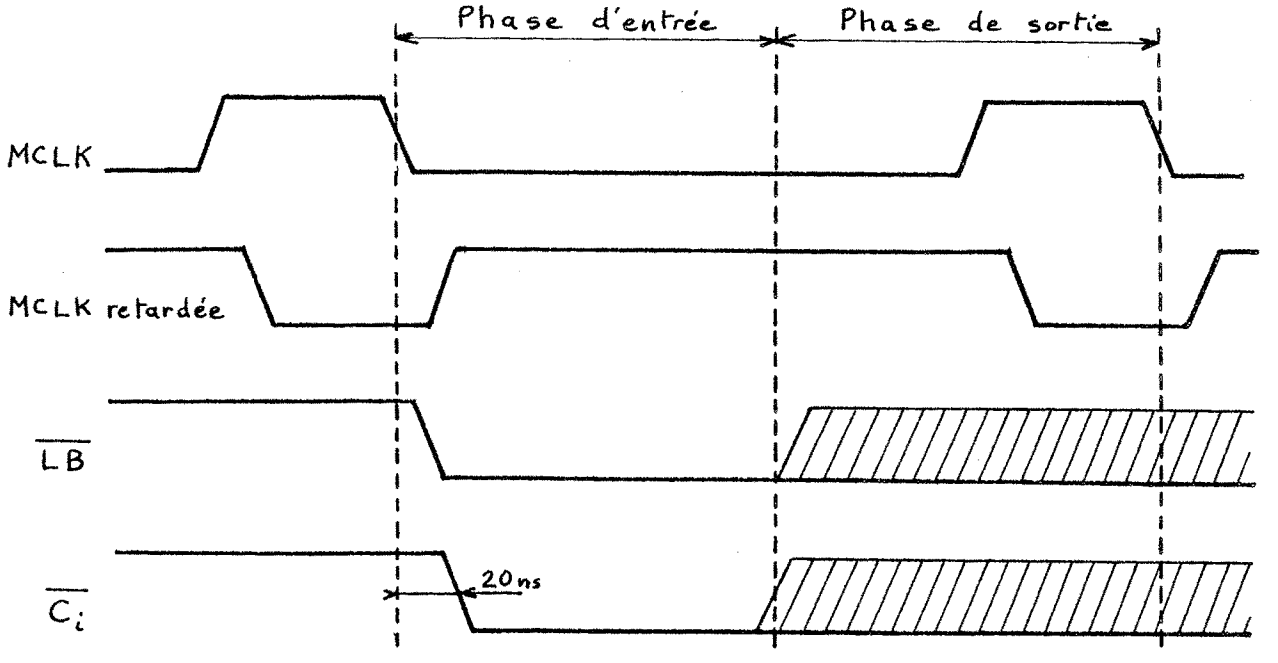


Fig.III-7 Diagramme de temps pour une commande générée pendant la phase d'entrée

III-3-1-4 La mémoire de travail (fig.III-8)

Elle contient 16 variables internes de 8 bits. Trois mots sont utilisés pour la gestion de l'Interface Canal :

- UNIT contient l'adresse attribuée à l'UCB
- STAPEND contient l'état-unité qui a été refusé par le canal
- SENSEBYT contient l'octet d'analyse.

Les autres sont utilisés pour la gestion de l'Interface Dispositif.

Cette mémoire est adressée de façon "orthodoxe", en 2 instructions. En effet, ses accès sont peu fréquents et en général hors des séquences privilégiées avec le canal. L'adressage se fait par des instructions accédant au banc droit et permet donc une possible extension jusqu'à 256 mots.

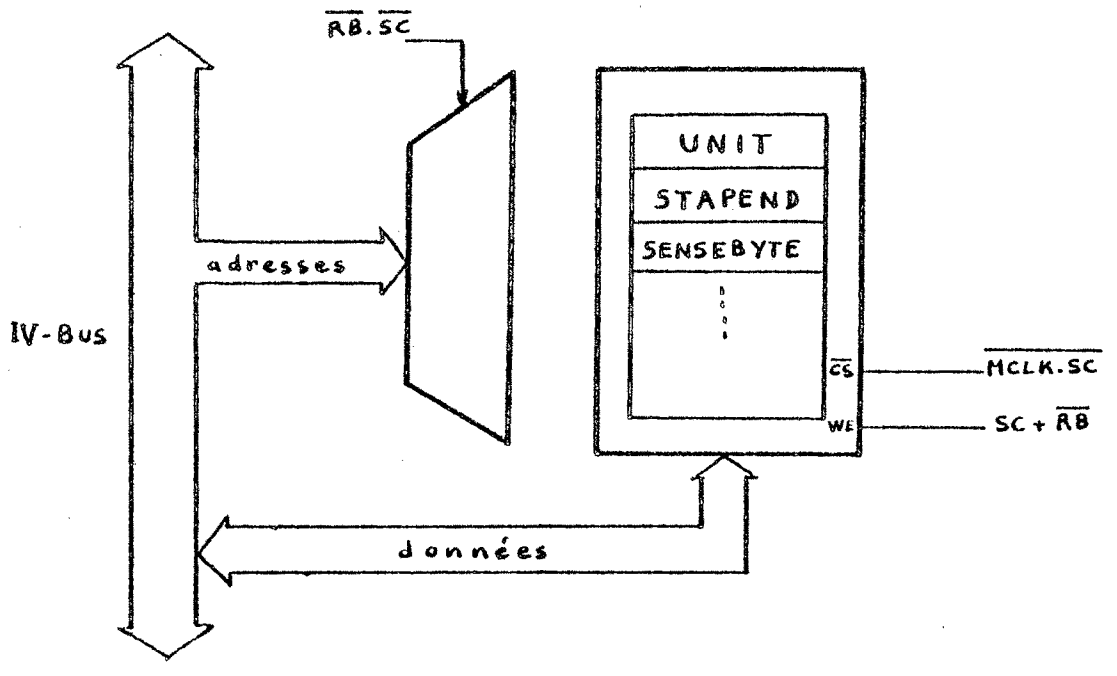


Fig.III-8 La mémoire vive de 16 mots de 8 bits

III-3-1-5 Les états internes

Ils sont rangés dans un registre externe de 8 bits adressé par la commande STA. Ils sont tous positionnés par programme mais utilisés de 2 façons :

- soit directement par câblage, en particulier lors de la reconnaissance de l'adresse (PRET, ASK) et aussi pour invalider plus rapidement les lignes "in" de l'interface d'E/S (SEL). La signification des états de ce premier groupe est la suivante :

  - PRET : l'UCB est disponible
  - ASK : l'UCB est demandeur et attend un "polling"
  - SEL : l'UCB a bloqué SLO et est sélectée.

- soit par programme en tant qu'indicateurs de branchement. Ce sont :

  - ST-DA : il y a transfert d'une donnée si ST-DA = 0, et transfert d'un état-unité si ST-DA = 1
  - STAK : un état-unité a été refusé par le canal
  - OCCUPE : l'unité a un état-unité en attente et ne peut recevoir de commande autre que TEST I/O
  - REQ : la requête de l'UCB n'est pas entièrement satisfaite (mode multiplex seulement)
  - CHAICOM : l'UCB a détecté un chaînage de commande (seulement pour vérification puisque l'UCB ne gère actuellement qu'une seule adresse).

Remarque : tous ces états internes ne concernent que la gestion de l'interface canal. Ceux nécessaires au dispositif utilisateur sont rangés dans un registre semblable mais ne seront détaillés que dans un exemple d'application de l'UCB.

III-3-2 L'Interface Canal

Nous nous intéresserons plus particulièrement à la description du circuit de reconnaissance automatique de l'adresse de l'UCB.

Rappelons néanmoins que les Tags-in sont générés par l'extension et sont positionnés dès la première phase de l'instruction, indépendamment de son exécution.

Précisons que Tags-in et Bus-in sont remis à 0 globalement de 2 façons :

  - par l'extension à l'aide des commandes $\overline{\text{RZTI}}$ et $\overline{\text{RZBI}}$ à l'intérieur des registres qui les génèrent
  - par câblage au niveau des émetteurs de lignes lors d'une situation anormale telle qu'un Reset, ou unité non prête, et de même lorsque l'unité n'est pas sélectée. Seuls RI et SLOS ne sont pas concernés.

La commande interne correspondante est :

PRET0 = PRET.SEL.00

Les récepteurs des Tags-out sont validés par 00, puisque ces lignes ne sont pas significatives en l'absence de ce signal.

Pour Bus-in et Bus-out nous devons prévoir l'émission et la réception d'un bit de parité. Pour IBM il s'agit toujours d'une parité "impaire", c'est-à-dire que le nombre de bits à 1 doit être impair.

Revenons donc aux circuits de reconnaissance d'adresse et de propagation de SLO.

III-3-2-1 Validation de l'adresse attribuée à l'UCB

Cette adresse est représentée par la présence d'un bit à 1 dans une mémoire (MEMU) de 256 mots de 1 bit, pouvant être accédée soit par l'UCB en écriture à l'aide de la commande $\overline{ECR}$, soit par le canal en lecture.

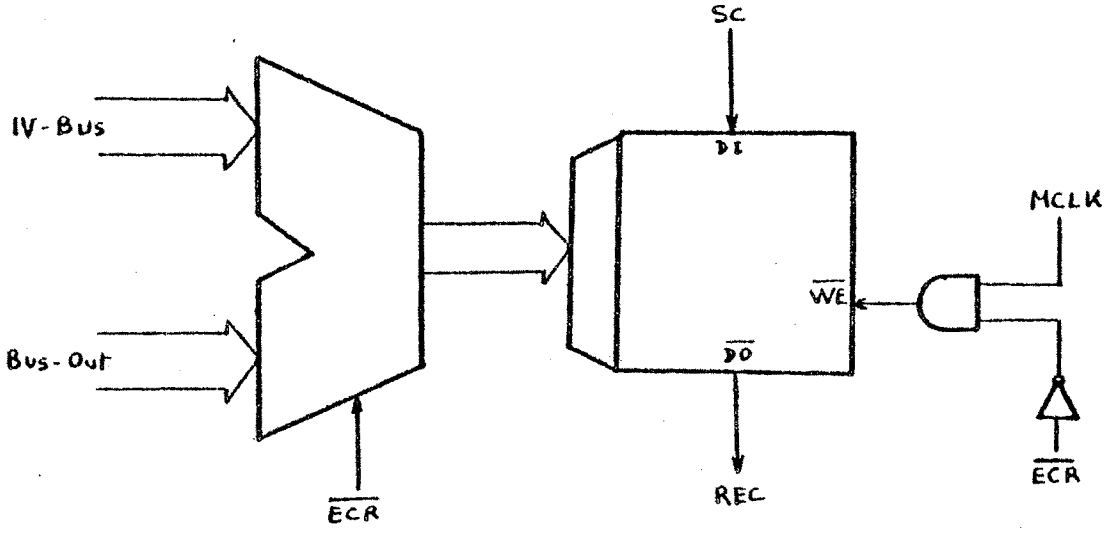


Fig.III-9 La mémoire des unités

La validation se fait par programme au cours de la phase d'initialisation (voir $ III-4-2).

III-3-2-2 Reconnaissance automatique de l'adresse

Après initialisation, MEMU est toujours en mode lecture et la présence de REC = 1 indique seulement que l'information sur Bus-out est identique à l'adresse de l'UCB. Il est donc nécessaire de fabriquer la fonction :

RECO = ADR.REC.PARITE où ADR = PRET.AO.SLO

ADR indique la présence effective d'une adresse sur BO (l'UCB devant évidemment être prête et la parité de l'adresse correcte).

III-3-2-3 Propagation de SLO

Rappelons que nous avons appelé SLOE le signal SLO entrant dans l'UCB, et SLOS le signal SLO sortant, qui est lui-même le SLOE de l'unité suivante. Nous avons conservé SLO comme nom du signal (SLOE.HO) qui sera testé par programme, puisque SLOE n'est valide pour l'unité que si HO = 1.

Nous devons propager SLOE quand :

- l'UCB n'est pas prête
- l'UCB n'est pas demandeur lors d'un "polling"
- l'UCB ne s'est pas reconnue lors d'une CISS
- il y a une erreur de parité dans l'adresse (CISS).

Nous résumons ces conditions par la fonction :

SLOS = SLOE (PRET + ASK.Polling + ADR.NONREC)

avec ADR défini comme précédemment, et NONREC = REC + PARITE .

Au contraire nous devons bloquer la propagation :

- si l'UCB est demandeur et si le canal effectue un polling
- si l'UCB est reconnue lors d'une CISS, qu'elle soit ou non demandeur.

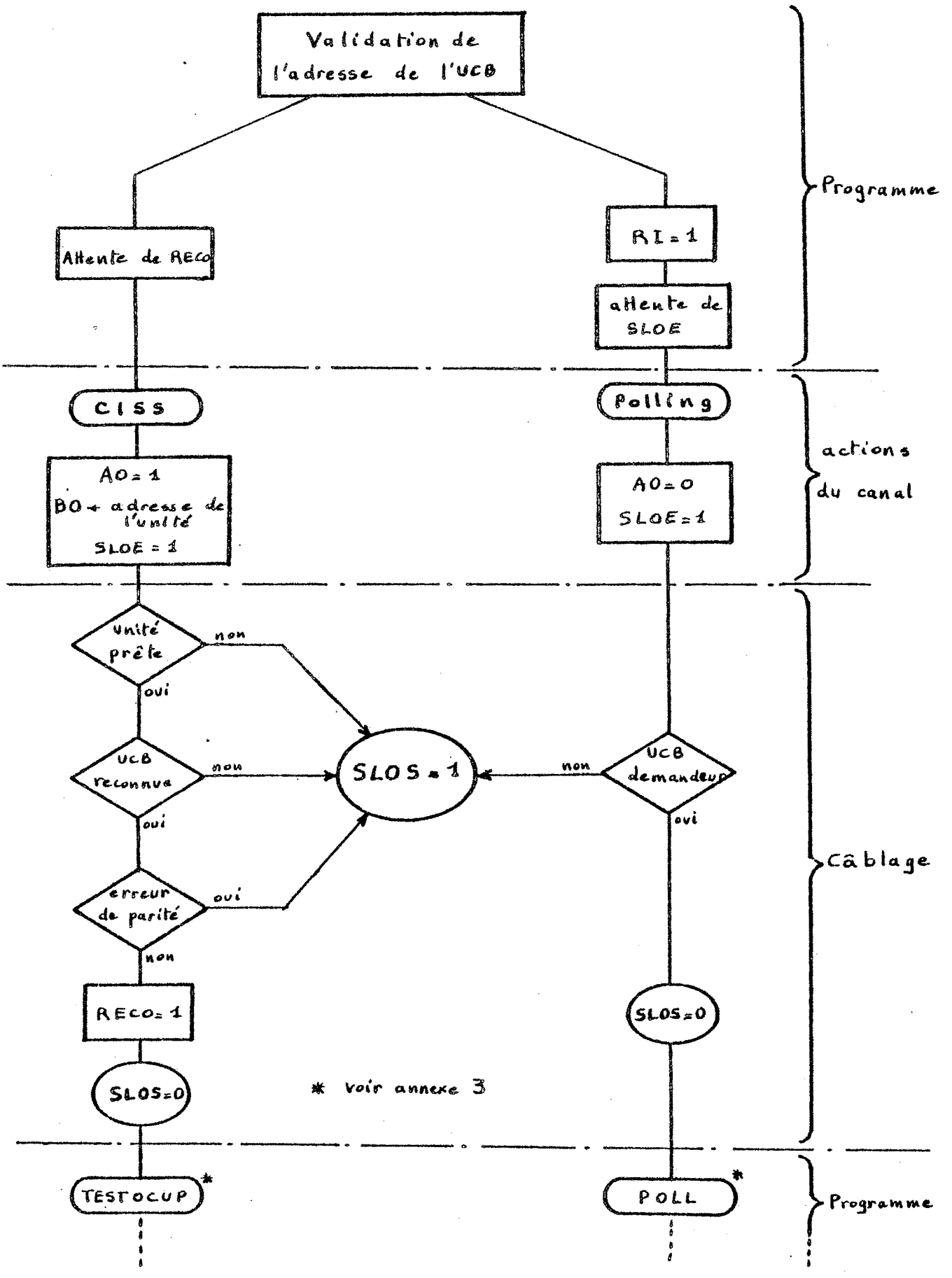


Fig.III-10 Le mécanisme de génération de SLOS

Nous pouvons alors préciser le terme (ASK.Polling) et nous utiliserons (ASK.RECO') où RECO' est un signal issù d'une bascule positionnée par RECO.

Ainsi, l'UCB n'étant pas demandeur, il y aura blocage de SLOE si elle est reconnue (ce qui sous-entend l'occurence d'une CISS), et propagation s'il y a "polling".

Remarque : RECO' sera obligatoirement remis à 0 en fin de sélection par la commande VSOS.

Nous pouvons résumer tout ceci par la fig.III-10 qui fait apparaître les parts de ce qui est résolu par câblage et par programme.

III-3-2-4 Modes de fonctionnement

Une dernière fonction assurée au sein du module Interface Canal est la sélection des modes de fonctionnement de l'UCB.

Elle se fait par l'utilisateur, au moyen d'un interrupteur qui permet de choisir un des 3 modes suivants :

- mode multiplex
- mode burst
- mode multi-byte

Cela revient à positionner un indicateur qui sera toujours à 0 dans le premier cas, toujours à 1 dans le deuxième cas et qui sera SLOE dans le troisième.

Le test de cet indicateur a lieu par programme après chaque transfert d'octet et conduit à une opération dans l'un des 2 modes connus, le mode multi-byte étant simplement une façon de transférer un certain nombre d'octets sans se dé-sélecter, tant que SLOE est actif. Ceci permet des transferts de blocs d'une dizaine d'octets et évite une monopolisation complète du canal comme en "burst" tout en assurant un débit supérieur au mode multiplex.

Nous donnerons les débits et diagrammes de temps obtenus, à la fin du prochain chapitre ; les schémas complets de l'Interface Canal sont en annexe 3, planches 5 à 8.

III-3-3 L'Interface Dispositif

Avant de donner un exemple concret (Chap.IV) et de faire d'autres suggestions (Chap.VI) sur ce que pourrait être un tel interface, rappelons que nous pouvons disposer actuellement :

-1- de 24 lignes programmables issues de 3 ports d'E/S de 8 lignes chacun.

Ces lignes peuvent être définies mono ou bi-directionnelles (par 8) ; les ports peuvent être utilisés aussi pour conserver des états internes du dispositif connecté ; ils sont sélectés, vis-à-vis de l'UCB, par les commandes issues de l'extension (donc très facilement extensibles) mais le sont indépendamment par le dispositif.

La planche 10 de l'annexe 3 donne un exemple de réalisation : le partage d'une mémoire. Nous générons alors 12 lignes d'adresse, 8 lignes de données bi-directionnelles, une ligne de lecture/écriture. Les 3 positions libres dans un des ports permettent le rangement de 3 états concernant la mémoire.

-2- de 2 lignes de synchronisation rapide

R en sortie est validée par la commande $\overline{\text{RQST}}$ pour toute requête de l'UCB vers le dispositif ; elle suspend alors ses activités jusqu'à réception de l'acceptation par $\overline{\text{G}}$ (une temporisation peut être prévue s'il y a lieu). La commande $\overline{\text{SUPRQ}}$ supprime l'effet de $\overline{\text{RQST}}$ et annule la requête (ceci peut être fait aussi lors d'un Reset).

Ce mécanisme a été utilisé dans l'application citée précédemment et concernant la gestion d'une mémoire partagée (voir Chap.IV et annexe 3, planche 9).

III-4 LES LOGICIELS

III-4-1 L'utilisation des instructions du 8X300

III-4-1-1 Généralités

Les différentes instructions (format, rôle...) sont détaillées à l'annexe 2. Rappelons qu'on peut les regrouper en 4 fonctions :

- des instructions de mouvement (Reg --> Reg, Reg --> Bus, Bus --> Reg, Bus --> Bus) accompagnées d'addition, de ET logique, de OU exclusif sur les données : ce sont MOVE, ADD, AND, XOR.

- des instructions de branchement : inconditionnel ou conditionnel par rapport au contenu d'un registre interne ou externe ; dans ce cas on est limité à l'intérieur d'une page (de 32 ou 256 instructions) : NZT, JMP.

- une instruction de chargement d'une valeur immédiate de 5 ou 8 bits : XMIT.

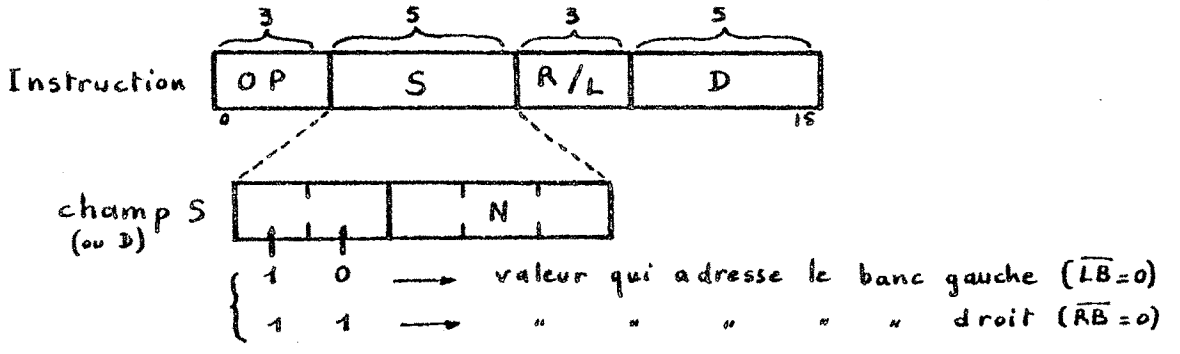
- une instruction forçant l'exécution d'une instruction quelconque de la même page : XEC.

L'exécution de l'instruction la plus générale se déroule en 3 phases :

- une phase d'entrée, où l'opérande peut subir une rotation et/ou être masqué
- une phase de traitement
- une phase de sortie, pendant laquelle l'opérande peut être "décalé" et/ou incorporé à sa valeur originale ("merge").

Toutes ces opérations (décalage, masquage, rotation, "merge") se font sur un nombre quelconque de bits et pendant la durée d'une seule instruction.

Les 5 bits des champs S ou D réservés à l'adressage d'une donnée Source (S) ou Destination (D) sur le bus IV se présentent sous la forme :



N spécifie, en fait, le bit poids faible de la zone de 1 à 8 bits concernée, sa longueur étant elle-même précisée dans le champ R/L de l'instruction. En numération hexadécimale, toute donnée sera donc identifiée par une valeur comprise entre 10 et 17 si elle est sur le banc gauche, et entre 18 et 1F sur le banc droit.

Lors d'une instruction destinée à sélectionner un registre externe (au sens large), le champ D sera obligatoirement 07 si ce registre est sur le banc gauche et 0F s'il est sur l'autre banc. Ces valeurs peuvent être vues comme les adresses de 2 pseudo-registres (respectivement IVL et IVR) contenant temporairement l'adresse des registres à sélecter, elle-même définie par les champs S et R/L. Rappelons que nous n'avons utilisé cette technique que pour adresser les mots de la mémoire de travail.

III-4-1-2 L'utilisation de l'extension

Chaque champ de l'extension est codé sur 3 bits (sauf VI sur 1 bit). Un code est réservé à la fonction NOP dans chaque champ afin de permettre une absence de commande pour certaines instructions.

Le tableau de la fig.III-11 résume les commandes crées. Elles seront écrites dans un triple champ opérande annexe reconnu par l'assembleur (voir annexe 5) où VI et A3 sont regroupés.

champ / code	A3	A2	A1
0	NOP	TO	(DATA)
1	RI	BO	(ADHI)
2	MTI	STA	(ADLO)
3	SOS	RZTI	RZBI
4	SI	RQST	TI
5	AI	SUPRQ	BI
6	STI	VSOS	ECR
7	OI	NOP	NOP

} exemples de commandes pour mémoire extérieure

Fig.III-11 Les différents codes de l'extension

Nous avons déjà dit que le positionnement des Tags-in se fait directement par l'extension. Nous aurons donc par exemple :

STI,x,TI positionne STI = 1
RI_,x,TI positionne RI = 0
(x pouvant être une des commandes du champ A2)

III-4-1-3 Les mécanismes de base

Nous décrirons sous ce terme quelques instructions ou groupes d'instructions fréquemment utilisés, sous forme de quelques exemples tels que :

a) préparation d'un octet d'analyse dans la mémoire de travail (RAMI)

```
(1)  XMIT   IVR,SENSEBYT
(2)  XMIT   COMMRJCT,1,0
```

La première instruction envoie sur le banc droit (IVR) du bus IV la valeur immédiate SENSEBYT qui est l'adresse symbolique de l'octet d'analyse dans RAMI.

La deuxième instruction envoie la valeur 0, sur 1 bit, à la position symbolique COMMRJCT du bus IV (par ex. 1B = bit IV3 sur le banc droit).

En l'occurence, nous avons mis à 0 le bit "commande rejetée" de l'octet d'analyse.

b) validation de l'adresse de l'unité dans MEMU

```
(1)  XMIT   IVR,UNIT
(2)  MOVE   RB7,0,LB7   z,z,ECR
```

La première instruction envoie sur le banc droit, l'adresse (UNIT) du mot de RAMI contenant l'adresse attribuée à l'UCB.

La deuxième instruction transfère ce mot sur le banc gauche tandis que la commande $\overline{\text{ECR}}$ sélecte MEMU. L'écriture se fait pendant le quatrième quart du cycle.

RB7 et LB7 représentent les 8 bits du bus IV respectivement sur les bancs droit et gauche, tandis que le 0 représente un champ de longueur 8 bits.

On remarque ici la faculté de transférer 1 octet entre les 2 bancs en une seule instruction.

c) envoi d'une information au canal

```
MOVE   R1,0,LB7   z,z,BI
```

L'information est placée sur Bus-in à partir du registre R1. On doit ensuite positionner le Tag-in qui précisera son type.

```
(1)  XMIT   IVR,STAPEND
(2)  MOVE   RB7,0,LB7   z,z,BI
```

Cette fois-ci, elle était contenue dans le mot STAPEND de RAMI (état-unité en attente).

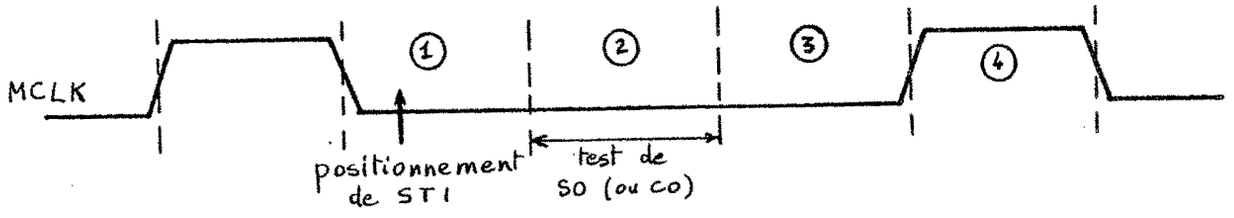
d) positionnement d'un TI et attente du TO correspondant

```
(1)  NZT   SO,1,*+2   STI,TO,TI
(2)  JMP   OK
(3)  NZT   CO,1,*-2   z,TO,z
```

la première instruction positionne STI = 1 par l'extension et lit le signal SO codé par exemple : 14 (donc le bit 4 du registre externe du banc gauche, sélecté par TO).

Si ce bit est à 1 on effectue l'instruction (2), en effet le bus IV complémente les données, sinon on passe à (3) qui elle-même teste le signal CO de la même façon.

Remarque : par rapport au déroulement de l'instruction on aura le diagramme de temps suivant :

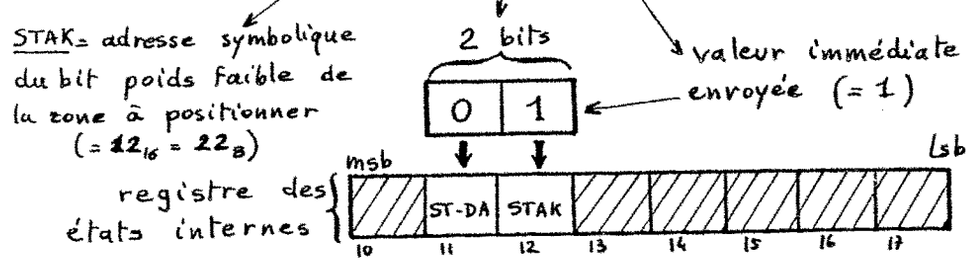


Le positionnement de STI se fait à un instant précis du premier quart, alors que le test de SO a lieu pendant le deuxième quart. Nous en tirerons une conséquence importante plus loin.

e) positionnement des états internes

```
XMIT   STAK,2,1   z,STA,z
```

Cette instruction exécute l'action suivante :



On positionne donc à la fois 2 états (ou plus) à des valeurs différentes, si les les bits correspondants sont adjacents.

f) utilisation d'une table de branchement

```
(1)  XEC   R11,*+1
(2)  JMP   I0
(3)  JMP   I1
(4)  JMP   I2
(5)  JMP   I3
```

Si le registre R11 contient les valeurs 0, 1, 2 ou 3, on exécutera les instructions (2), (3), (4) ou (5). Ces instructions étant des branchements inconditionnels le compteur ordinal est modifié ce qui n'est pas le cas habituel de l'instruction XEC.

Nous utiliserons fréquemment cette technique pour le retour des sous-programmes communs aux traitements correspondant aux différentes commandes envoyées par le canal (le registre R11 contiendra alors le code de la commande). En effet nous ne disposons pas de couple d'instructions du type JSR/BSR-RTS.

III-4-2 Le logiciel de l'Interface Canal (INTCAN)

III-4-2-1 Présentation générale

Il contient 252 instructions dans sa version actuelle, soit la moitié de tout le logiciel implanté dans la mémoire de programme de l'UCB. Il gère toutes les séquences de contrôle définies au $ II-4-4-1 du Chapitre II sauf la détection d'une sélection et des Reset (opérations câblées).

Il décode les commandes de base (NOP, lecture, écriture, analyse) ainsi que la commande correspondant à l'instruction Test I/O ; cependant il n'assure pas l'exécution des commandes étendues et dans ce cas il envoie comme status initial : "erreur sur unité" et positionne le bit "commande rejetée" dans l'octet d'analyse.

Il reconnaît également les séquences de déconnexion imposées par l'instruction HALT I/O, y compris en mode burst, au prix d'un "alourdissement" des algorithmes et d'une diminution de la vitesse de transfert (voir $ III-5-2).

RECO et le mode de fonctionnement utilisé sont détectés comme les Tags-out puisqu'ils sont rangés dans le même registre externe qu'eux.

L'ensemble du logiciel INTCAN est représenté dans les planches A à L2 de l'annexe 3.

III-4-2-2 Les séquences critiques

a) vis-à-vis du fonctionnement de l'UCB : Ce sont les séquences telles que la séquence de sélection de l'UCB par polling (CUIS) et la détection de l' "interface disconnect".

- "La C.U.I.S" : nous devons effectuer cette séquence :

  - à chaque transfert de donnée en mode multiplex
  - pour envoyer l'état-unité de fin dans ce même mode
  - pour envoyer un état-unité précemment refusé par le canal et mis en attente dans l'UCB
  - pour envoyer un état-unité correspondant à un changement d'état du dispositif (p.ex. : "fin sur unité" lors du passage à l'état "Prêt").

Les principales règles à observer sont alors:

-1- on ne doit pas faire une requête au canal pour présenter un état en attente lorsque SPO = 1. Dans ce cas il peut se produire, soit une commande TIO (consécutive à une instruction Test I/O du CPU ou à une initiative du canal lui-même) que nous reconnaîtrons en testant RECO (CISS normale), soit la chute de SPO permettant la requête.

-2- on peut alors faire RI = 1 et attendre SLO (polling). Le problème qui se pose est que la reconnaissance de SLO est programmée et que le blocage/propagation de SLOE est câblé et commandé par l'état ASK quand l'UCB est demandeur. Il ne faut donc pas bloquer SLOE quand on détecte SLO = 1, sans être sûr que ce signal est bien destiné à l'UCB. Pour cela nous avons mis au

point l'algorithme de la planche E de l'annexe 3, qui permet de reconnaître le signal SLO destiné à l'UCB, au risque de perdre une occasion de se connecter (perte d'au maximum 2 us).

Remarque : nous avons introduit un autre indicateur (REQ) qui permet d'éviter l'exécution de cet algorithme lorsqu'une précédente requête a été émise et que l'unité n'a pas terminé l'envoi ou la réception de tous les octets, en mode multiplex seulement puisque les transferts en mode burst n'ont pas besoin de CUIS. Cela revient à maintenir RI = 1 à partir du premier octet, et donc à empêcher le service d'une unité moins prioritaire tout en conservant le mode multiplex pour les autres unités.

Nous pouvons résumer le déroulement de la séquence CUIS par le schéma suivant :

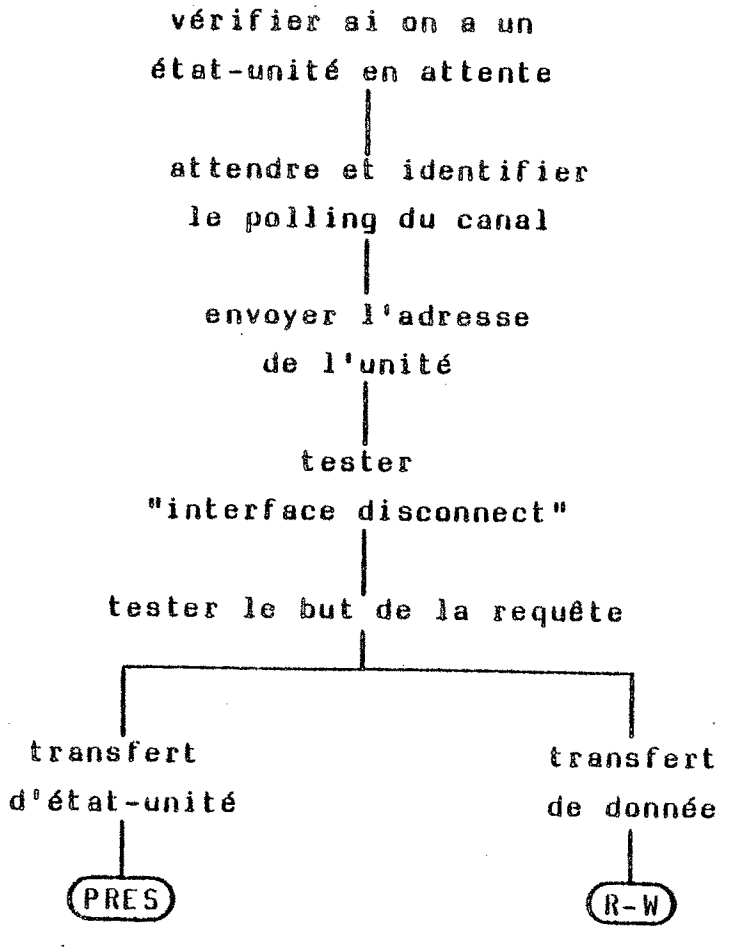


Fig.II-12 Organigramme simplifié de CUIS

- "L'interface disconnect" : comme nous l'avons dit, le problème essentiel est que cet événement est imprévisible.

Si nous sommes en mode multiplex, et même si un transfert est en cours, le canal laisse l'échange élémentaire se terminer et lancera la séquence de déconnexion par une CISS, ou bien nous la détecterons lors de la prochaine CUIS.

Comme nous voulons également pouvoir travailler en mode burst nous sommes obligés d'introduire un test programmé de la séquence pendant les transferts de données, avec les conséquences sur la vitesse que nous verrons plus loin.

b) vis-à-vis des performances de l'UCB

De ce point de vue nous ne parlerons ici que de la vitesse de transfert des données.

L'échange d'un octet s'effectue en 2 phases :

-1- une initialisation qui se fait dans le logiciel du dispositif et qui consiste à préparer la réception ou l'émission d'un octet (adresse, mise à jour du reste, etc...). Elle est donc indépendante de l'Interface Canal.

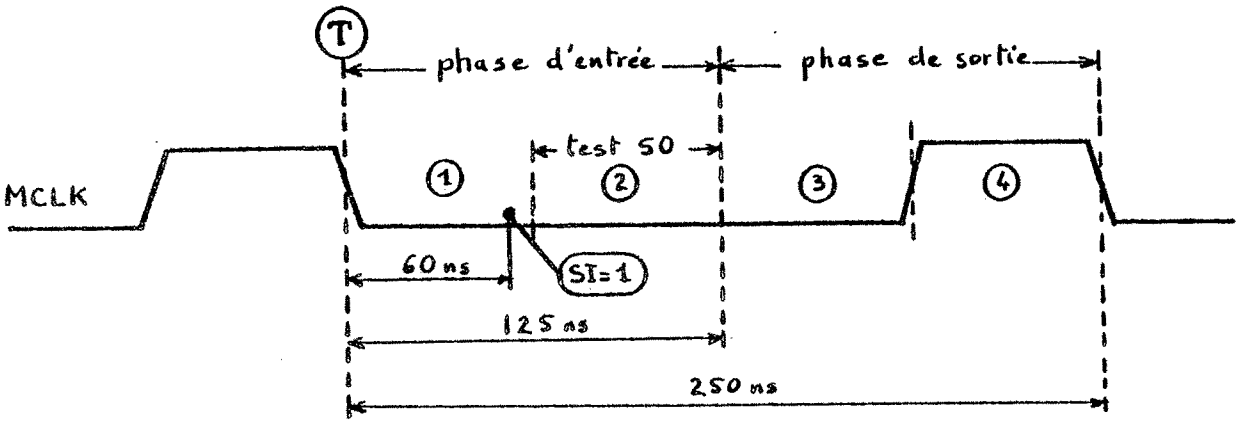
-2- l'échange lui-même qui tient en 4 instructions :

```
(1)  NZT   SO,1,*+2   SI,TO,TI
(2)  JMP   OK
(3)  NZT   CO,1,*-2   z,TO,z
(4)  JMP   STOP
```

suivies, si on est en écriture, de : `MOVE  LB7,0,R2  z,BO,z`
ou précédées, si on est en lecture, de : `MOVE  R2,0,LB7  z,z,BI`

(OK : l'échange est accepté, SO = 1)
(STOP : arrêt de l'échange, CO = 1)

Par rapport à l'exécution de l'instruction (1), on a le diagramme de temps suivant (déjà donné au $ III-4-1-3) :



SI est positionné à T+60 ns, et SO est testé entre T+62,5 ns et T+125 ns. Il est donc peu probable que le canal ait déjà répondu (avec SO par ex.). En effet, on sait que, en lecture, la donnée doit être valide sur bus-in dans les 100 ns qui suivent la montée de SI. On exécutera donc l'instruction (3) avant de tester à nouveau SO, ce qui conduit à une attente de 500 ns au lieu de 250 ns si on avait interverti (1) et (3).

Le gain de 250 ns sur la durée totale minimum théorique d'un échange élémentaire (1 µs) serait de 25 %. Cependant nos observations ont montré que le canal ne répondait pas à SI avant 1 à 2 µs ; il devient alors illusoire de chercher le meilleur emplacement du test de SO.

Par contre les obligations de tester la suppressibilité d'un octet de donnée (SPO = 1) et l'"interface disconnect", conduisent à une baisse sensible de la vitesse. Voir les résultats théoriques au $ III-5-2.

III-4-3 Le Logiciel de l'Interface Dispositif (INIDISP)

Il est essentiellement fonction de l'application envisagée ou du périphérque connecté. Il devra lui aussi gérer des états internes propres à ce dispositif, ainsi que le dialogue qu'il nécessite , probablement plus simple que le précédent.

Par contre la phase d'initialisation sera certainement plus importante et même relativement complexe si le dispositif est multiple (équivalent à une unité de contrôle qui gère plusieurs organes d'E/S).

De plus il contient les phases de contrôle, à un niveau logique, des échanges dont nous venons de parler.

Sachant que nous en verrons un exemple précis au cours du Chapitre IV, nous nous contenterons d'en rappeler la structure générale en annexe 3, planches M-N.

III-4-4 L'Inter-communication entre les 2 Logiciels

Pour des questions de souplesse, nous avons donc décomposer le logiciel de l'UCB en 2 tâches. Il s'agit maintenant de prévoir un mécanisme de communication entre les 2.

D'un point de vue fonctionnel, on peut distinguer :

III-4-4-1 Les tests imbriqués

D'une façon générale, c'est le logiciel-dispositif qui a la main puisque le canal n'est pas en dialogue permanent avec l'UCB. Il devra donc tester l'état de la relation entre l'UCB et le canal le plus souvent possible par l'intermédiaire de la variable RECO.

D'autre part, lors de l'arrivée d'une commande de transfert de donnée (lecture ou écriture), le logiciel-canal doit s'assurer de la possibilité de ce transfert : d'où les test des indicateurs BUV et BUP, conduisant le cas échéant à la réponse "occupé" dans l'état-unité initial.

On voit ainsi la justification du terme "tests imbriqués" en ce sens où chaque logiciel teste une variable ou un indicateur appartenant à l'autre module. Ces tests apparaissent également dans l'annexe déjà citée.

III-4-4-2 Les appels inconditionnels

A la suite des tests précédents ou si cela est nécessaire (par ex. un changement d'état du dispositif, ou la préparation du transfert d'une donnée) le logiciel en cours d'exécution passe la main à l'autre tâche à l'aide d'un simple branchement inconditionnel. La portée de l'instruction JMP étant totale il n'y a pas de problème de changement de page comme pour les instructions NZT et XEC.

III-5 RESULTATS OBTENUS

Nous montrerons d'une part, que les séquences d'interface s'effectuent correctement et d'autre part, que nous avons obtenus des débits intéressants.

III-5-1 Les diagrammes temporels des séquences d'interface

fig.III-13 à fig.III-21

III-5-2 Les débits

III-5-2-1 Débits théoriques

Nous nous placerons dans le cas le plus favorable où le canal répond de suite à nos tests de lag-out et où le dispositif a préparé une file d'attente des données (on n'aura alors besoin que de 3 instructions dont 2 pour la communication entre les 2 logiciels !). Nous obtenons alors un "débit maximum théorique" en comptant le nombre d'instructions exécutées.

En mode burst :

- en effectuant seulement le test de donnée suppressible :
  400 Koctets/s
- en effectuant en plus le test d' "interface disconnect" :
  300 Koctets/s
- en ajoutant le test de l'indicateur REQ :
  250 Koctets/s

En mode multiplex :

- en forçant la priorité de l'UCB (REQ) :
  160 K.octets/s
- en ne le faisant pas :
  100 K.octets/s

Dans tous les cas, nous sommes donc au-delà des débits du canal multiplex , sans atteindre ceux d'un canal sélecteur...

III-5-2-2 Débits mesurés

Ceux obtenus dans la gestion d'une mémoire commune comme celle décrite dans le Chapitre IV et l'UCB y ayant un accès prioritaire, ont égalé le débit maximum du canal comme on pouvait s'y attendre.

C'est-à-dire :

- de l'ordre de 110 K.octets/s en mode burst (débit maximum du canal multiplex)
- et environ 40 K.octets/s en mode multiplex.

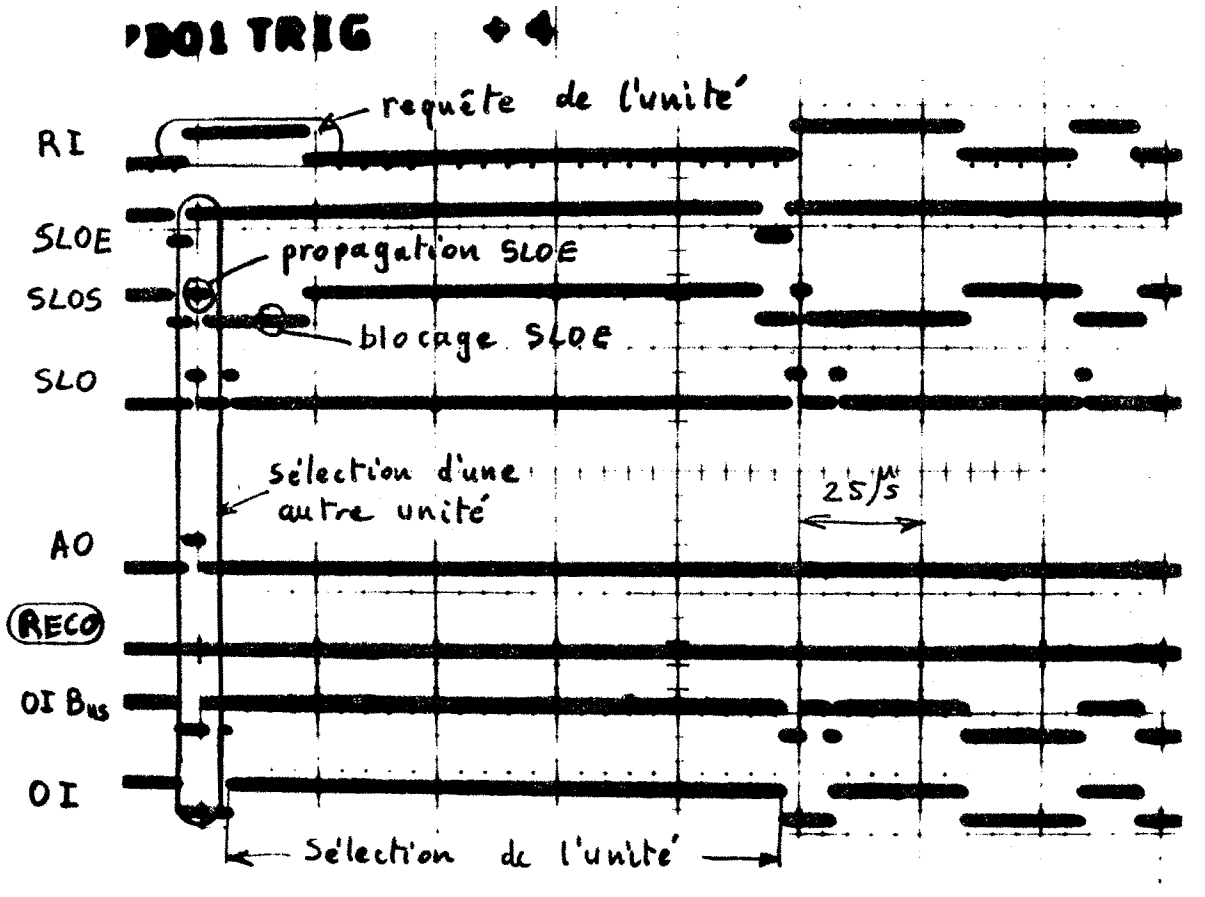


Fig.III-13 Les lignes de connexion pendant les requêtes de l'UCB

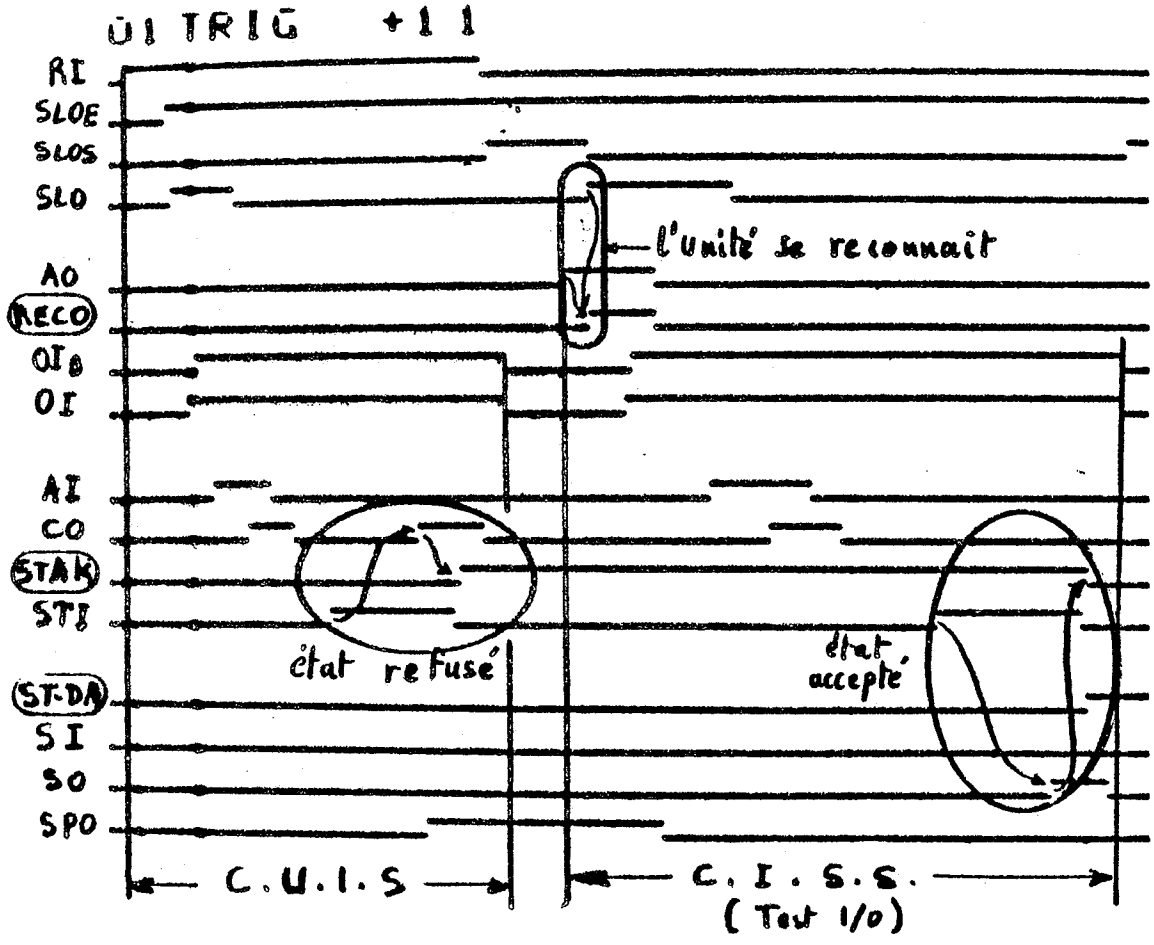


Fig.III-14 Séquence CUIS avec état-unité refusé, suivie d'un IIO par le canal pour obtenir cet état

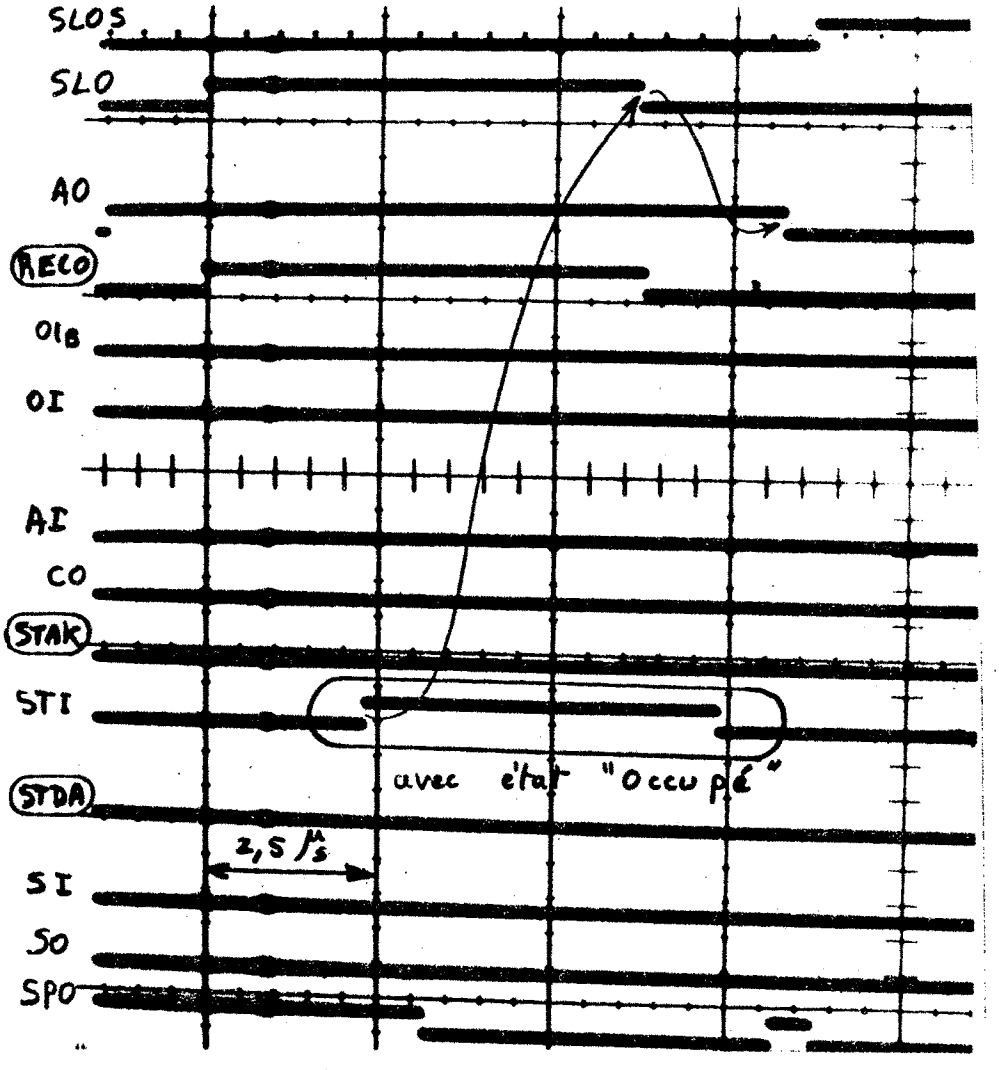


Fig.III-15 Séquence CUBS ("Control-Unit-Busy-Sequence")

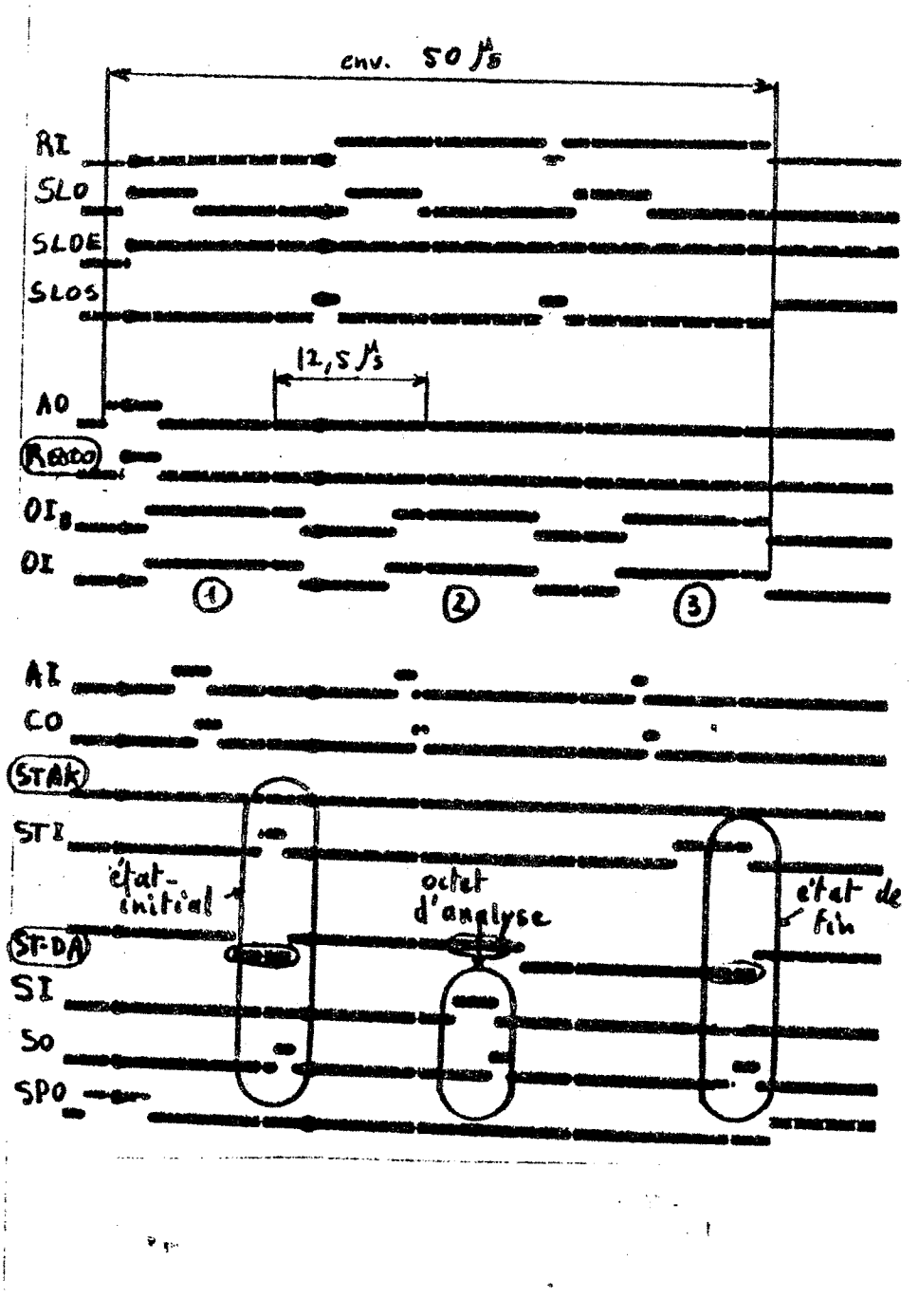


Fig.III-16 Commande d'analyse en mode "multiplex"

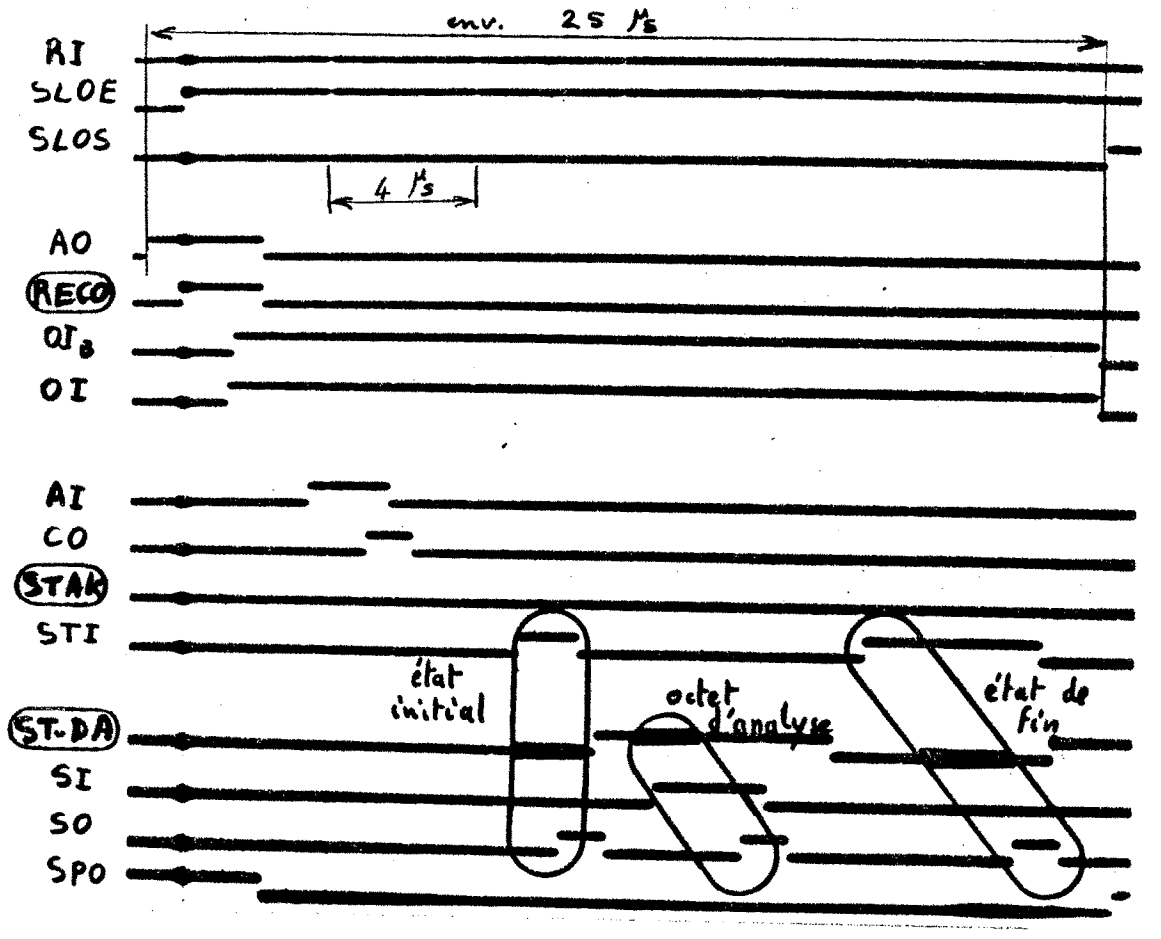


Fig.III-17 Commande d'analyse en mode "burst"

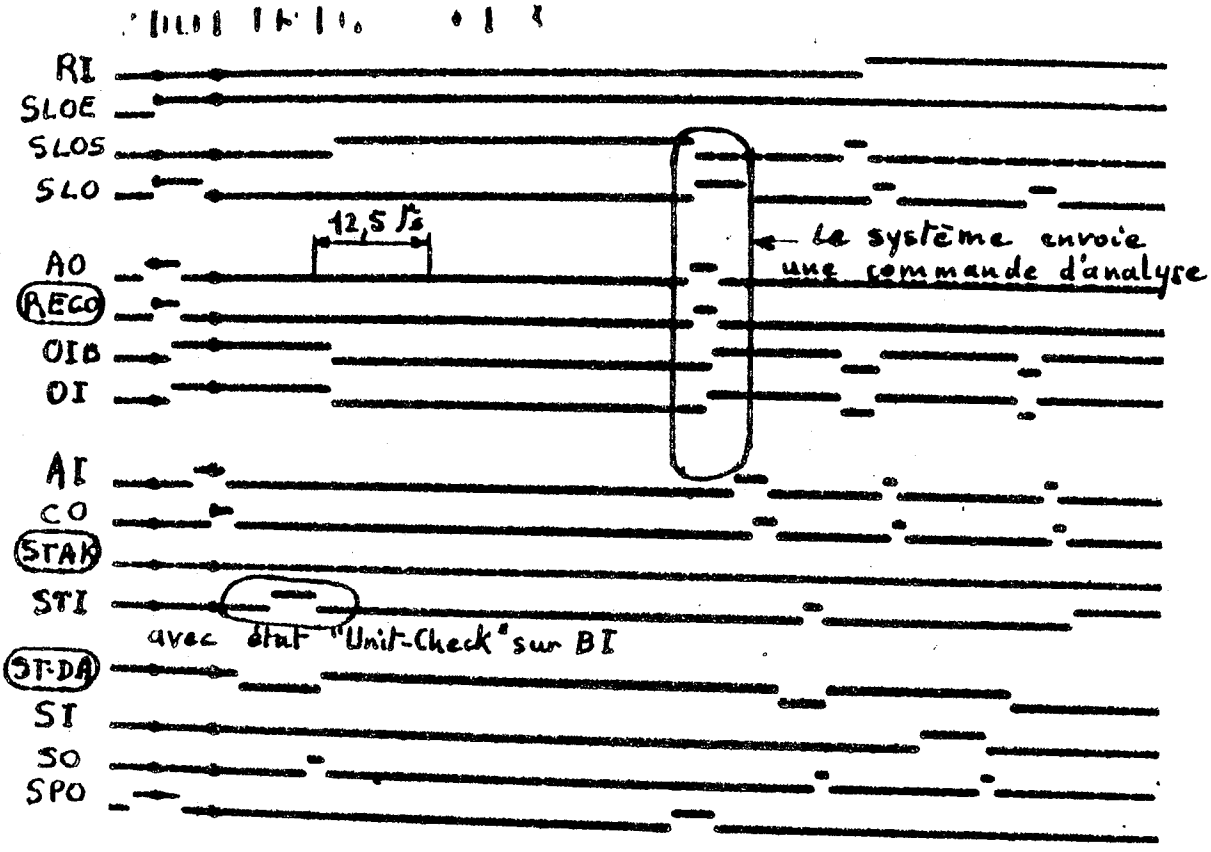


Fig.III-18 Commande de contrôle non décodée par l'UCB, suivie d'une analyse par le système

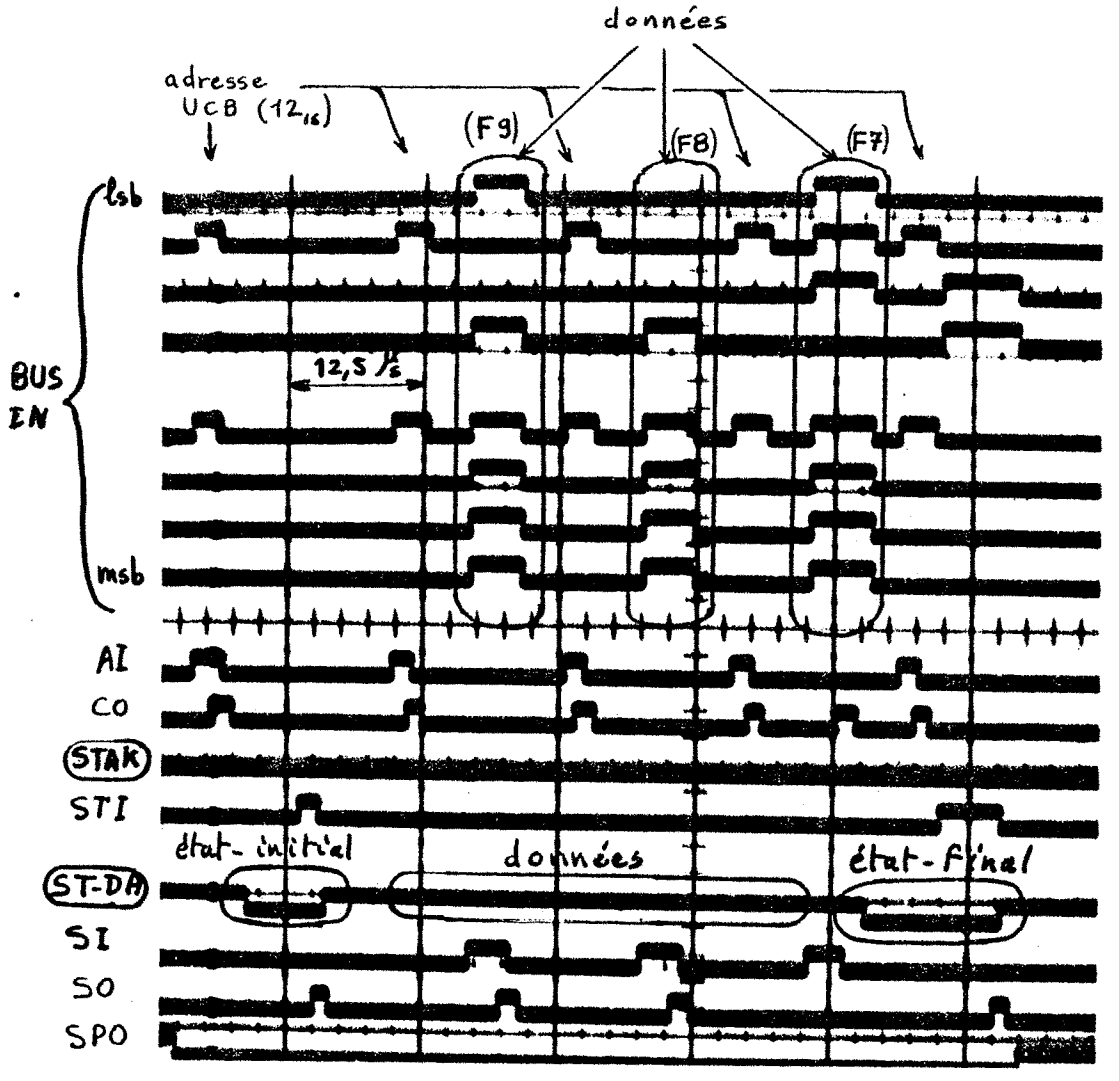


Fig.III-19 Extraits d'une opération de transfert de données

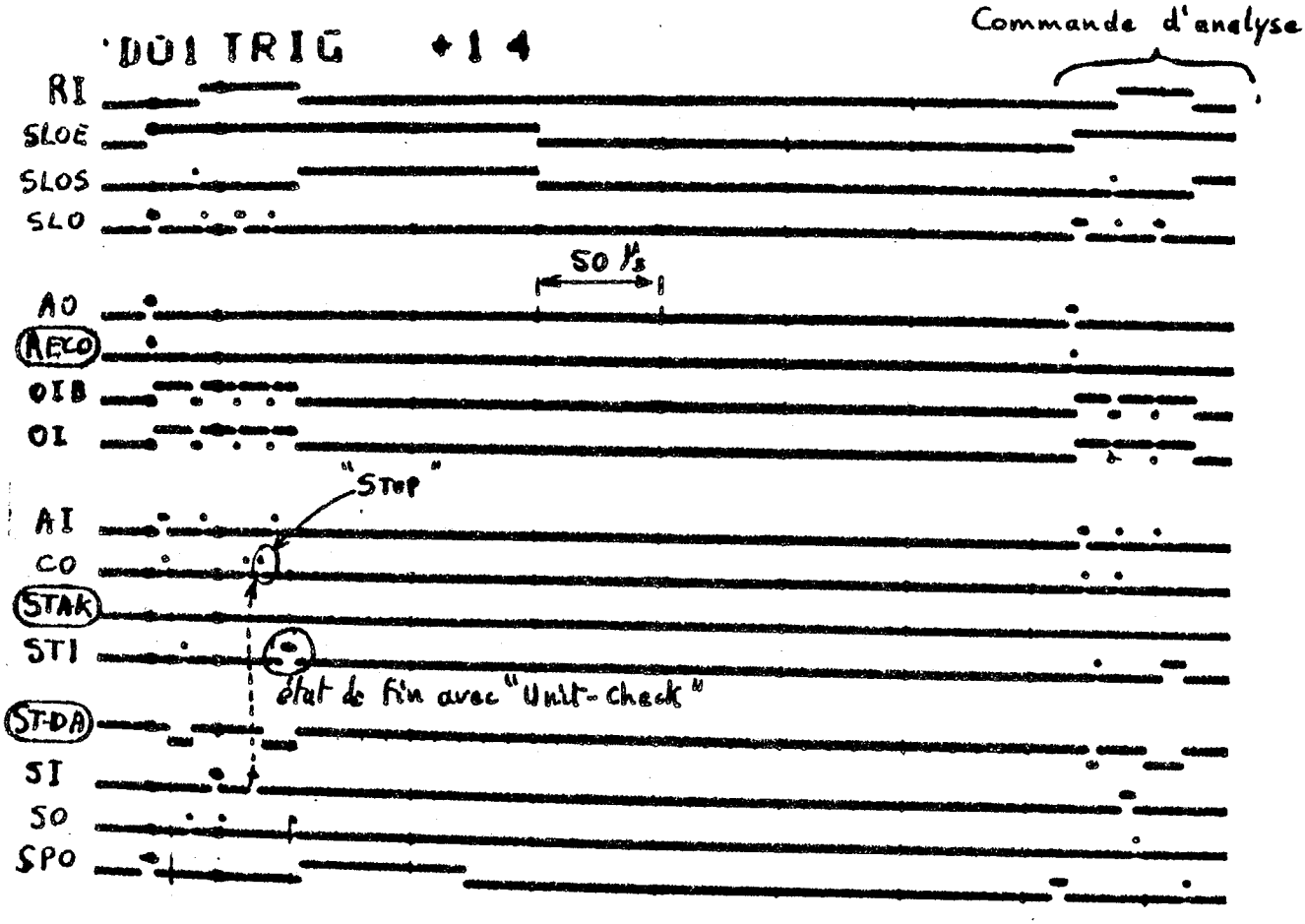


Fig.III-20 Commande d'écriture avec détection d'une erreur de parité (le système fait alors une analyse)

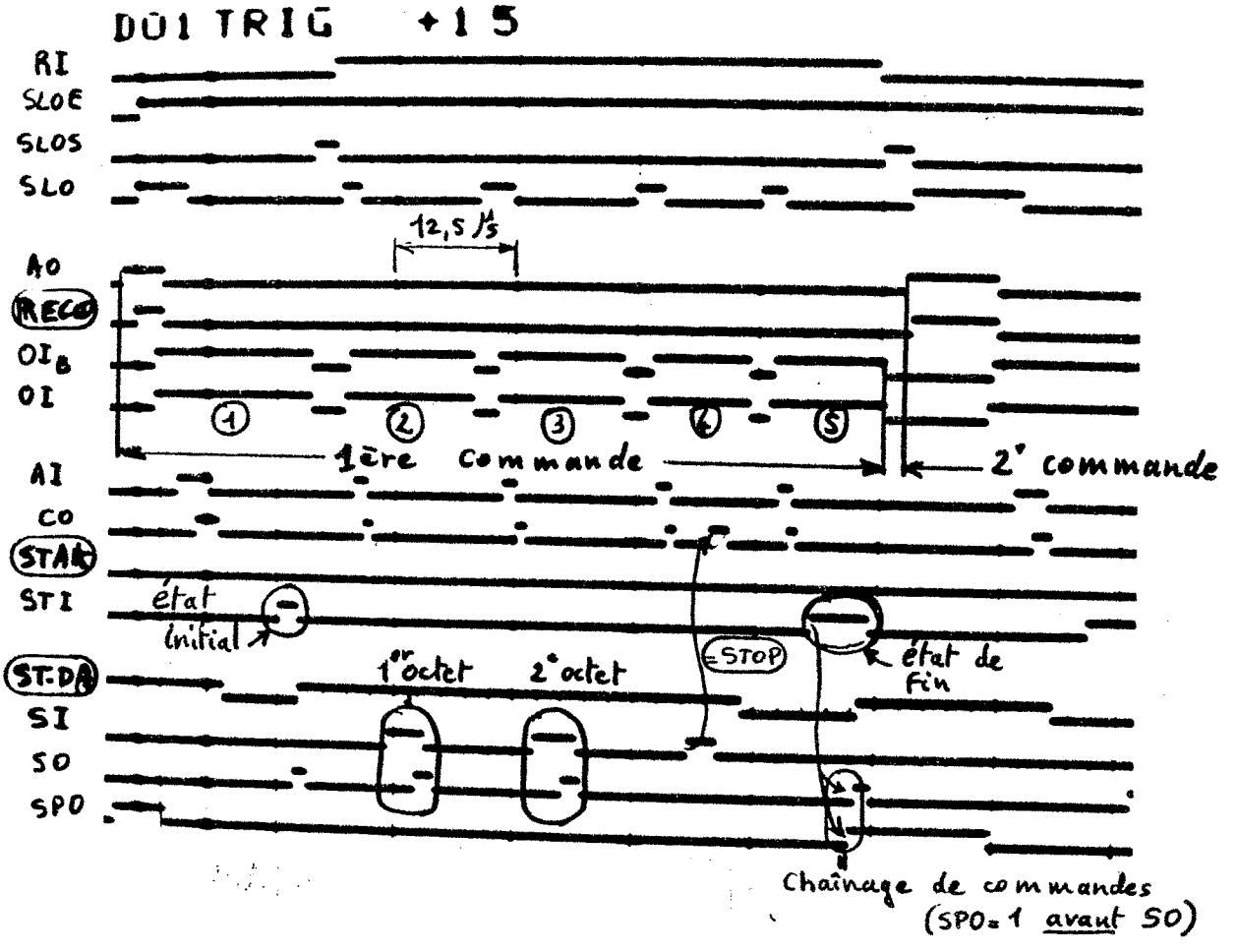


Fig.III-21 Commande d'écriture de 2 octets, suivie d'un chaînage de commandes

CHAPITRE IV

UNE APPLICATION DE L'UCB
LE MODULE INTERFACE-CANAL DU SYSTEME PIAR

IV-1 INTRODUCTION

IV-2 LA STATION DE TRANSPORT DE CYCLADES

IV-2-1 Présentation Générale

IV-2-1-1 Le Réseau CYCLADES

IV-2-1-2 Les Tâches de la Station de Transport

IV-2-1-3 L'implémentation de la Station de Transport

IV-2-2 Décomposition de la Station de Transport

IV-2-2-1 Introduction

IV-2-2-2 La Station Résiduelle

IV-2-2-3 La Station Externe

IV-3 LE SYSTEME PIAR

IV-3-1 Caractéristiques de PIAR

IV-3-1-1 Caratéristiques fonctionnelles

IV-3-1-2 Caractéristiques architecturales

IV-3-2 Les modules fonctionnels de PIAR

IV-3-2-1 Le Module Contrôleur de Communication

IV-3-2-2 Le Module Station Transport

IV-3-2-3 Le Module Interface Canal

IV-1 INTRODUCTION

La connexion d'un ordinateur à un réseau tel que CYCLADES présente un certain nombre de difficultés. Parmi celles-ci, nous rappellerons seulement les problèmes posés par l'implémentation de la Station de Transport sur l'ordinateur hôte, en particulier au point de vue fiabilité, efficacité, en tenant compte des coûts correspondants. Nous verrons ainsi qu'une des solutions les plus intéressantes, quoique coûteuse, est l'implémentation sur un ordinateur frontal.

Or, d'autre part, l'évolution de l'architecture des ordinateurs, accentuée par l'apparition et le développement des microprocesseurs et des circuits d'entrée-sortie LSI, va dans le sens d'une "migration de l'intelligence" à la périphérie des systèmes, pour des coûts très faibles, comme nous l'avons déjà signalé au Chapitre I.

La confrontation de ces 2 situations fait clairement apparaître comme réaliste et fructueuse la réalisation d'un frontal intelligent à base de microprocesseur et supportant l'implémentation de la Station de Transport en déchargeant d'autant l'ordinateur du centre de calcul concerné.

C'est ce qui a été réalisé par R. Saettone dans le système PIAR (Périphérique Intelligent d'Accès au Réseau) que nous allons décrire dans ce chapitre <SAE>.

Nous pourrons également à cette occasion voir une application de l'UCB qui constitue un des modules du système PIAR, chargé évidemment de la connexion à l'ordinateur hôte, en l'occurence, l'IBM 360/67 du CICG.

IV-2 LA STATION DE TRANSPORT DU RESEAU CYCLADES

IV-2-1 Présentation générale

IV-2-1-1 Le Réseau CYCLADES

Nous ne ferons ici que rappeler les caractéristiques essentielles du Réseau CYCLADES qui a été déjà abondamment décrit <POUZ>.

C'est un réseau hétérogène d'ordinateurs communiquant entre eux par le réseau de communication CIGALE <GRA>, qui utilise la technique de commutation de paquets et le routage adaptatif. Contrairement à d'autres réseaux de ce type, CIGALE n'assure pas de traitements complexes ; cette simplicité est la première cause du poids logiciel que devront supporter les ordinateurs participants.

Les utilisateurs du réseau (abonnés), accèdent au réseau de communication par des stations de transport (ST) contenues dans chaque ordinateur du réseau. Ces stations de transport utilisent le service de communication offert par CIGALE pour assurer le service de transport en suivant le protocole de transport.

Pour leur part, les abonnés dialoguent entre eux en suivant un protocole d'utilisateur qui leur est propre. Un exemple largement utilisé est le Protocole d'Appareil Virtuel <ZIM1>.

Ces différents services et protocoles sont résumés dans la fig.IV-1 qui fait apparaître également les interfaces de communication et de transport.

D'un point de vue pratique :

- CIGALE est constitué de plusieurs mini-ordinateurs MITRA 15, formant les noeuds d'un réseau maillé de lignes téléphoniques louées

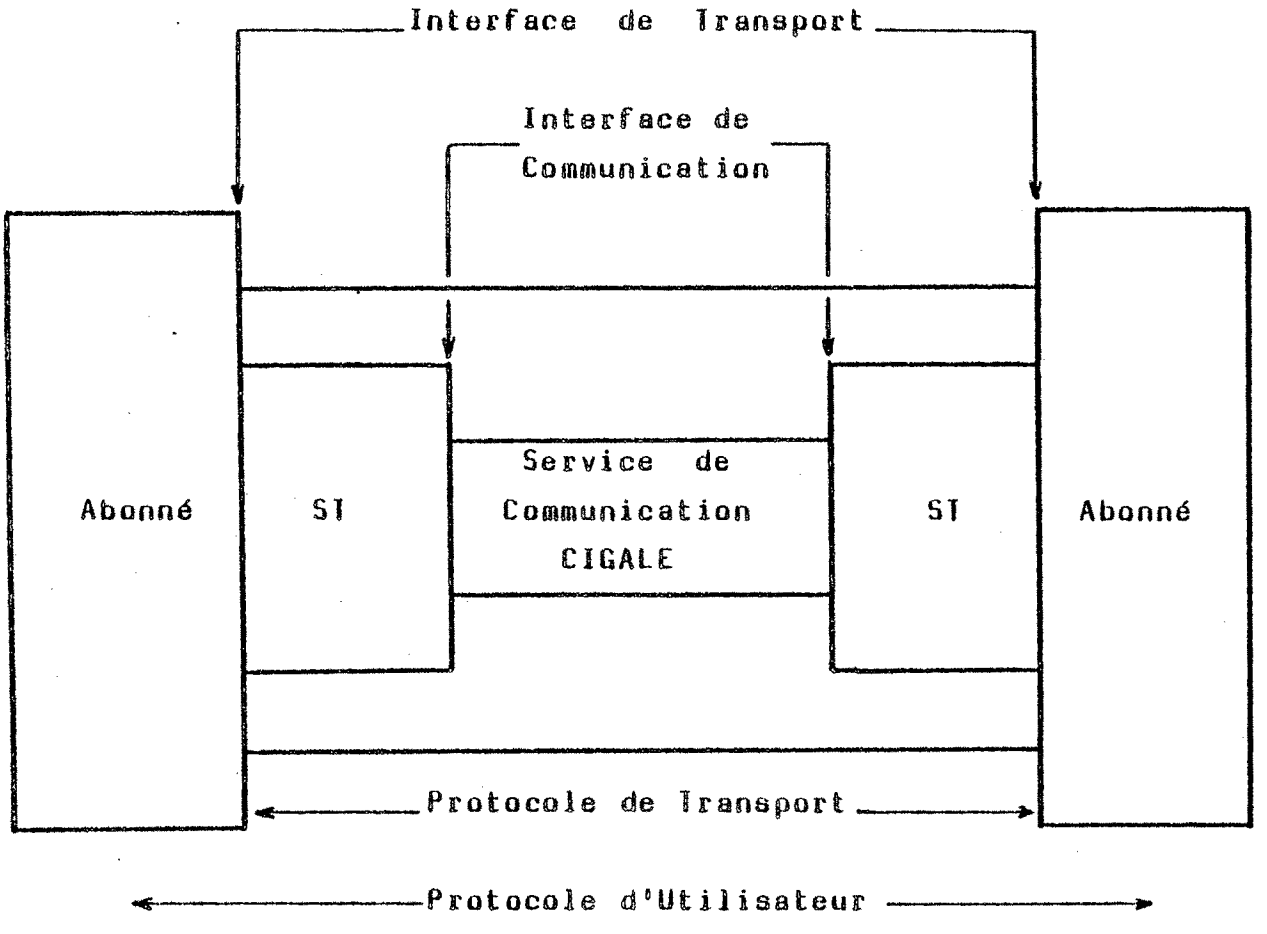


Fig.IV-1 Représentation symbolique du fonctionnement du réseau CYCLADES

- Les SI sont des couches de logiciel implantées de diverses façons sur les ordinateurs du réseau, et en constituent la machine de transport.

D'un point de vue <u>fonctionnel</u>, le réseau CYCLADES se caractérise donc par :

- une indépendance entre les ordinateurs connectés et le réseau de communication qui les relie
- une hiérarchisation des différents niveaux de matériel et de logiciel.

IV-2-2-2 Les tâches de la Station de Transport

D'une façon générale la Station de Transport est chargée :

- d'offrir aux abonnés un accès au réseau CYCLADES
- de reprendre les erreurs de CIGALE
- de multiplexer plusieurs voies logiques (flots) sur une voie seule voie physique.

Nous rappellerons brièvement les différents services offerts par la Station de Transport ST2 <GAR>.

Ce sont :

- l'adressage : l'identification d'un flot (FL-ID) ouvert entre 2 abonnés se fait par le couple des 2 adresses (PI-ID à l'intérieur de la ST) des portes origine et destination. Chaque adresse de porte est unique dans le réseau par le couple (ST-AD, PI-ID), où ST-AD identifie la Station de Transport.

- la négociation de session : un flot est ouvert et fermé par un abonné local ou distant à l'aide des commandes FL-INIT et FL-TERM, dont les paramètres définissent les services qui seront exécutés sur ce flot.

- les services de base : un abonné pouvant avoir plusieurs portes ouvertes en même temps et plusieurs abonnés pouvant être actifs ensemble, la ST assure le multiplexage et le démultiplexage des flots sur la ligne série entre l'ordinateur hôte et le noeud de CIGALE. D'autre part elle effectue la fragmentation et le réassemblage des messages de taille variable (lettres). Ces 2 services sont transparents aux abonnés.

- les services additionnels : contrôles d'erreurs, de flux, et adressage réduit, ne sont fournis qu'en option et après accord préalable entre les 2 abonnés.

IV-2-2-3 L'implémentation de la Station de Transport

Le choix de l'implémentation de la ST est important aussi bien du point de vue de la fiabilité que des performances obtenues.

Les problèmes de fiabilité sont axés vers l'indépendance des fonctions de façon à ce que la défaillance d'un composant n'altère pas le bon fonctionnement des autres ; en particulier les erreurs chez les abonnés ne doivent pas avoir de conséquence pour la ST ; de même que la présence de la ST ne doit pas perturber le système d'exploitation de l'ordinateur donc l'ensemble des centres usagers <QUI>.

Le maintien des performances intrinsèques du système d'exploitation est également à prendre en compte lors de l'implémentation d'une ST. Inversement les contraintes "temps réel" imposées à la ST par un système d'exploitation travaillant en temps partagé peuvent nuire à son efficacité.

Parmi un certain nombre de possibilités d'implémentation de la ST <ZIM2>, les plus connues sont : (voir fig.IV-3)

-1- l'intégration au système d'exploitation
-2- l'implémentation en mode usager
-3- l'implémentation dans un frontal.

Dans le premier cas, les problèmes de fiabilité sont bien résolus si l'intégration est faite par le concepteur du système ; mais la gestion de la connexion au réseau (ligne synchrone et temporisations) risque de diminuer les performances du système.

La deuxième solution est la plus couramment utilisée. Dans ce cas, la ST se présente comme un usager du système, et ce dernier ne voit pas ses performances intrinsèques diminuer, mais ses performances globales vis-à-vis des autres utilisateurs sont là encore réduites. Les abonnés peuvent être ou non sur la même

tâche utilisateur que la Station de Transport. Dans le premier cas,les fonctions doivent être bien séparées pour conserver une bonne fiabilité. Dans le deuxième cas il faut disposer d'un moyen efficace de communication entre les tâches utilisateurs et la ST. Dans cette solution un sous-système de multiprogrammation doit également être défini et implémenté.

La troisième solution offre l'avantage de la séparation des abonnés et de la ST, de la conservation de l'intégrité du système d'exploitation et du déchargement de l'ordinateur hôte de la gestion du logiciel de transport. Cependant, il reste les problèmes de la fiabilité de la liaison du frontal à l'ordinateur de traitement, de l'écriture du logiciel du frontal gérant les interfaces avec le réseau et l'ordinateur, et du coût d'un tel élément (en général un mini-ordinateur).

En ce qui concerne le cas particulier du CICG, l'ordinateur IBM 360/67 supporte alternativement 2 systèmes d'exploitation : CP-67 et ASP/OS-MVT. Dans le premier cas, tout le logiciel réseau (processus ST et abonnés) est implémenté dans une seule machine virtuelle et géré par le sous-système SYNCOP <DAN> (fig.IV-2). Ce choix est dû aux contraintes citées plus haut par rapport au système d'exploitation et au frontal, et à l'absence d'un mécanisme assez rapide de communication entre machines virtuelles. Sous ASP/MVT l'implémentation est du même type.

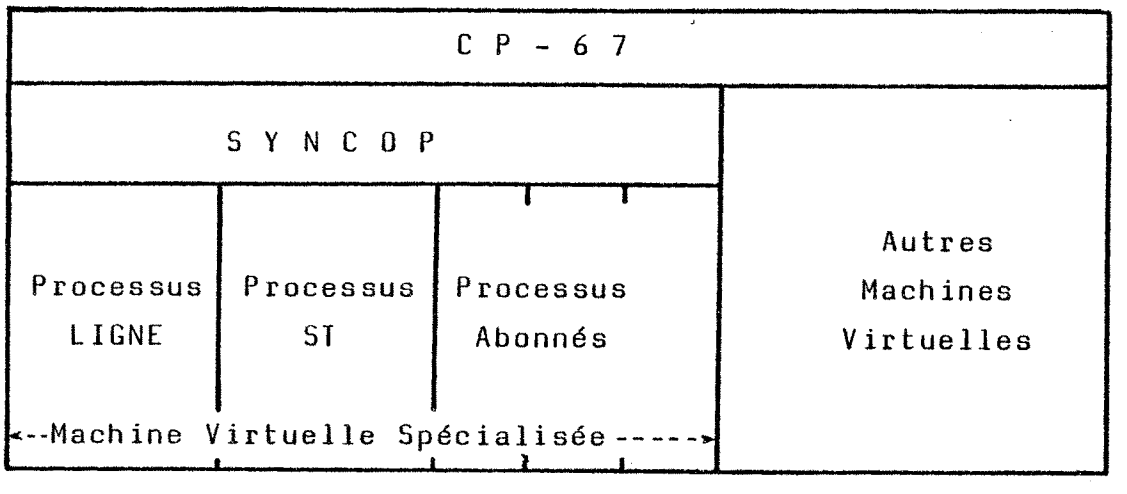
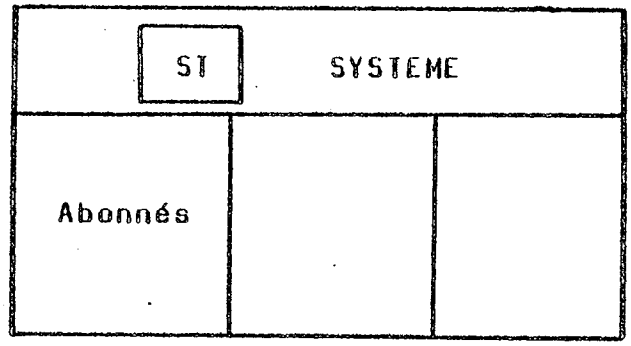
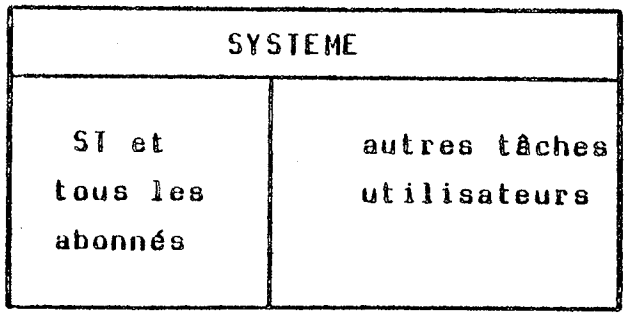


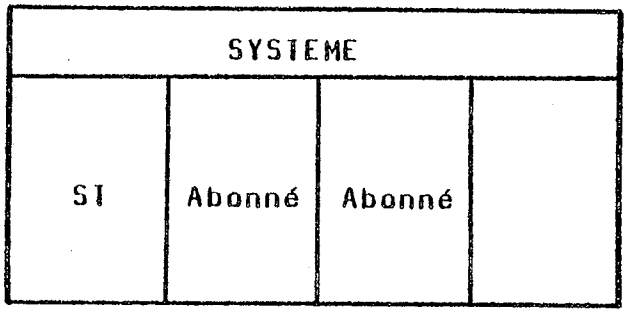
Fig.IV-2 Implémentation de la Station de Transport ST2 au CICG



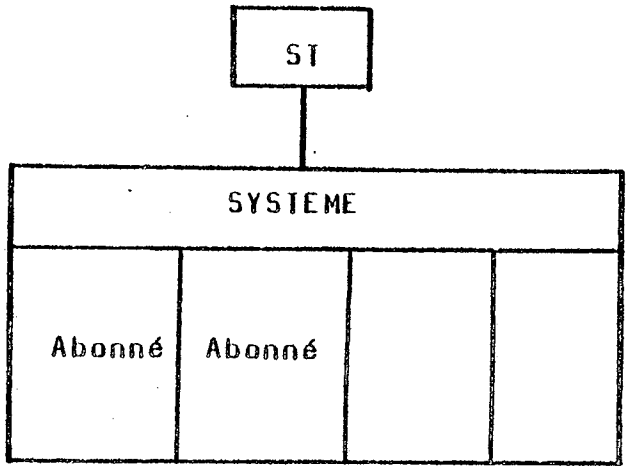
-1- La SI est intégrée au Système d'Exploitation Les abonnés sont des tâches utilisateurs.



-2- La SI et les abonnés se trouvent dans la même tâche utilisateur.



-3- La SI et les abonnés sont des tâches utilisateurs. Le Système leur fournit un moyen de communication.



-4- Un ordinateur frontal héberge la SI.
Les abonnés sont des tâches utilisateurs.

Fig.IV-3 Quelques exemples d'implémentation d'une SI de CYCLADES

IV-2-2 Décomposition de la Station de Transport

IV-2-2-1 Introduction

D'après ce que nous venons de voir, 2 voies semblent intéressantes pour améliorer l'efficacité globale de la ST et du site d'implémentation : la réalisation d'un frontal fiable et peu coûteux, ou la mise au point d'un mécanisme de communication rapide entre machines virtuelles. Nous analyserons par la suite la première solution.

Quant à la deuxième voie, une solution est de réaliser matériellement un "canal-à-canal" (CTC), mais qui soit bouclé sur le canal multiplexeur du 360. Ce CTC permettrait la communication entre plusieurs machines virtuelles, par couples. Nous en présentons une première approche réalisée avec un microprocesseur Motorola MC 6800 dans l'annexe 4.

Le choix de décomposer la ST en 2 éléments découle du fait qu'l est préférable de laisser exécuter certaines fonctions de la ST par l'ordinateur hôte : l'interface avec les abonnés et les tâches de fragmentation/réassemblage. En outre la gestion de la ligne de connexion avec CIGALE n'étant plus du ressort du 360 par l'intermédiaire de l'unité de transmission 2701, il devenait inutile de continuer à utiliser la procédure "half-duplex" propre à IBM : BSC.

IV-2-2-2 La Station Résiduelle

Elle consiste en une mince couche de logiciel sans contrainte de temps réel et exécute les fonctions suivantes :

- interface de la ST externe avec les abonnés locaux
- interface avec l'ordinateur hôte
- fragmentation et réassemblage des lettres sur les tampons de mémoire fournis par les abonnés, sous le contrôle de la ST Externe.

Elle peut être implémentée en mode usager comme l'est actuellement la ST version-ST2, sous SYNCOP, en reprenant les processus ECRITURE, LECTURE, ACCES et en éliminant les fonctions relatives au protocole de transport.

Remarque : Elle pourrait également être implémentée d'une façon intégrée au système, en tenant compte du fait qu'elle peut être vue comme la méthode d'accès à la ST Externe et qu'il est possible de faire des analogies entre le réseau CYCLADES et une unité de disques <SAE,III-3>.

IV-2-2-3 La Station Externe

C'est en fait la Station de Transport puisqu'elle en contrôle directement ou indirectement toutes les fonctions. Elle contient en particulier les tables de contexte associées aux flots.

Rappelons ces fonctions :

- négociation de session
- contrôle de la fragmentation et du réassemblage
- contrôles d'erreurs et de flux en émission et en réception
- contrôle de toutes les temporisations prévues par le protocole de transport
- envoi et réception de télégrammes.

A celles-ci s'ajoute l'interface avec la ST Résiduelle consistant en une dizaine de commandes.

Elle est donc implémentée dans un frontal offrant la particularité intéressante d'être constitué d'un ensemble de 3 microprocesseurs, chacun exécutant une fonction bien définie, et formant le système PIAR que nous allons décrire maintenant.

La fig.IV-4 représente les distributions du logiciel d'accès au réseau, dans la situation actuelle et dans le cas d'une décomposition de la station de transport.

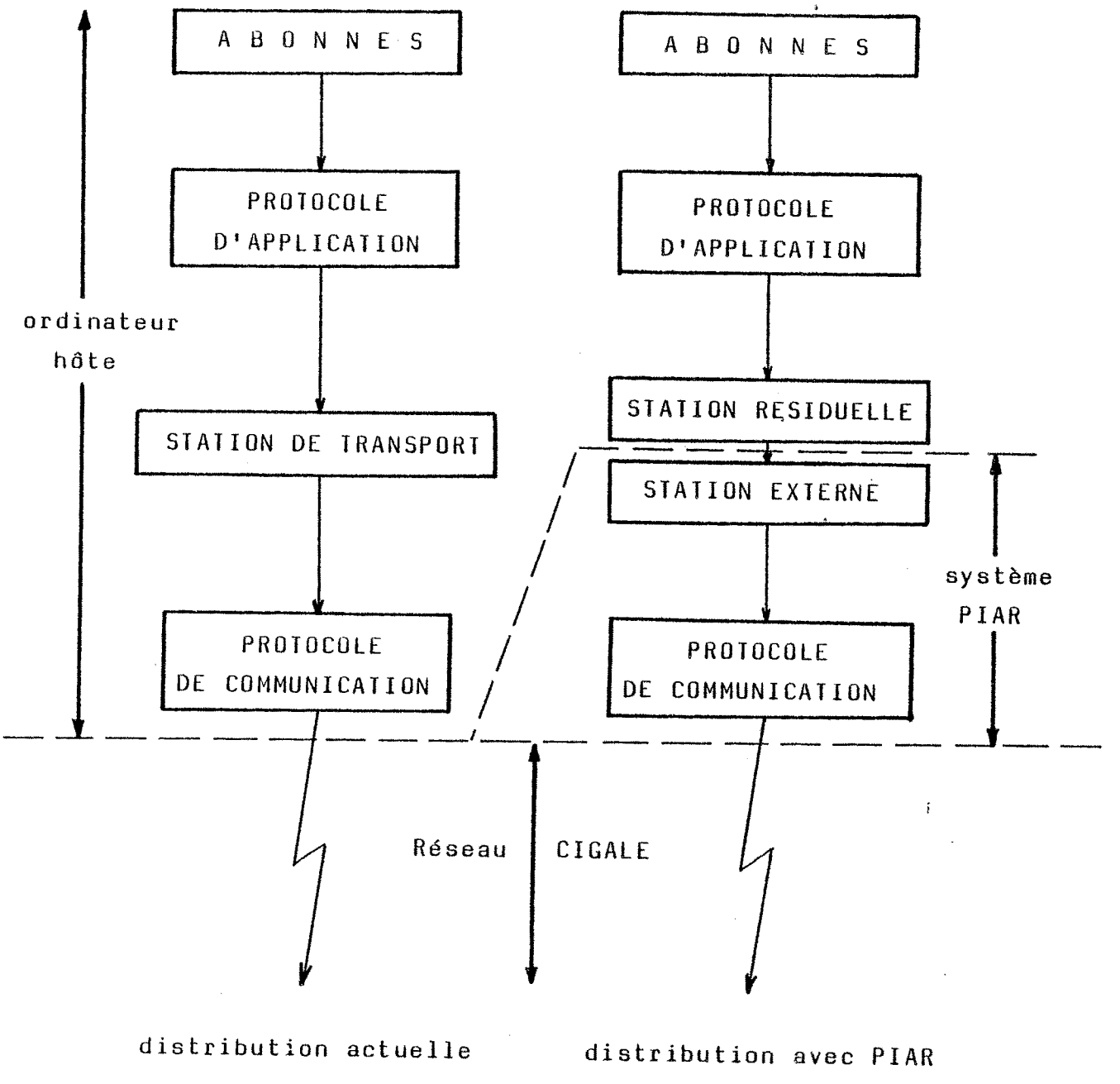


Fig.IV-4 Les distributions du logiciel d'accès au réseau

IV-3 LE SYSTEME PIAR

Il fournit les moyens matériels et logiciels pour la connexion d'un ordinateur IBM 360/370 au réseau CYCLADES, en déchargeant au maximum cet ordinateur des tâches logicielles correspondantes.

IV-3-1 Caractéristiques de PIAR

IV-3-1-1 Caractéristiques fonctionnelles

Ce sont celles qui concernent essentiellement le travail que doit effectuer PIAR afin qu'il atteigne les objectifs fixés.

Les fonctions à exécuter y sont séparées en 3 ensembles bien définis :

- les fonctions relatives à la Station de Transport
- les fonctions relatives à la liaison avec CIGALE
- les fonctions relatives à la liaison avec l'ordinateur hôte.

Les premières fonctions ont été rappelées au paragraphe précédent et sont celles attribuées à la SI Externe.

La liaison avec CIGALE est faite à l'aide d'une ligne de transmission synchrone suivant la procédure habituelle du réseau CYCLADES, qui est la procédure "full-duplex" IMM-UCIG.

Le système PIAR se présente comme un périphérique standard connecté au canal multiplexeur du 360, ce qui permet des transferts de données en parallèle par octets. L'utilisation de l'UCB pour réaliser cette liaison évite de faire appel à une unité de transmission du constructeur et à une ligne série et permet une indépendance vis-à-vis du site tout en déchargeant l'ordinateur hôte de la gestion de cette ligne série.

Ces différents groupes fonctionnels sont indépendants et asynchrones, c'est-à-dire qu'une tâche activée se déroule indépendamment des autres. Deux de ces groupes contiennent des tâches de liaison avec des milieux extérieurs au système et devront donc tenir compte des contraintes électriques et temporelles dues à ces milieux.

Enfin, de ce même point de vue on peut dire que l'implémentation de la Station de Transport dans PIAR reste transparente aux abonnés, locaux et distants et aux autres ST du réseau.

IV-3-1-2 Caractéristiques architecturales

Elles découlent d'une part de critères de décision relatifs aux architectures à plusieurs microprocesseurs <FLY>, <ENS> et d'autre part des caractéristiques précédentes. A ceci s'ajoute des considérations sur les critères de choix des microprocesseurs eux-mêmes, en particulier leur disponibilité, leur support logiciel, et leur puissance (même sous-employée).

Les caractéristiques fonctionnelles conduisent "naturellement" à un découpage du logiciel en 3 tâches allouées statiquement à 3 microprocesseurs dans 3 modules fonctionnels.

La communication entre ces modules sera du type "producteur-consommateur" par l'intermédiaire d'une mémoire commune selon le principe de boîte aux lettres. Le couplage sera donc lâche.

Ce mécanisme de communication conduit à une structure d'interconnexion basée sur un bus parallèle commun. De plus, afin d'éviter un trop grand nombre de conflits d'accès aux éléments communs, seules les ressources nécessaires à la communication entre les modules seront partagées par les modules. Les moyens de ce partage seront des allocateurs imbriqués car le demandeur doit accéder au bus et à la mémoire.

Chaque module contiendra :

- une unité centrale
- une mémoire morte de programme
- une mémoire vive
- éventuellement des circuits spécifiques aux fonctions à réaliser si la technologie en offre
- des circuits de contrôle et d'E/S.

Cet ensemble définira son environnement interne ou privé. Tandis que mémoire et bus communs constituent son environnement externe.

D'une façon générale PIAR se présente comme un système à microprocesseurs multiples organisés selon une architecture répartie.

Remarque : Dans CYCLADES les messages sont traités par des protocoles indépendants, chacun d'eux étant supporté par un logicielspécialisé.

On a concrètement un empilement des protocoles auquel correspond une hiérarchie des logiciels, du type "utilisateur-ressource" , l'activité étant propagée d'une couche à l'autre. C'est bien ce qui se passe dans le système PIAR, où chaque module est activé par le précédent dans le sens de la communication.

Cependant la réalisation matérielle choisie ne restitue pas cette hiérarchie. C'est ainsi que seuls 2 niveaux matériels représentent les 3 niveaux logiciels : celui des modules fonctionnels et celui du module de communication. Les relations entre les modules d'un même niveau sont alors des liens fonctionnels <ANCI>.

L'organisation matérielle du système PIAR est donc celle de la fig.IV-5.

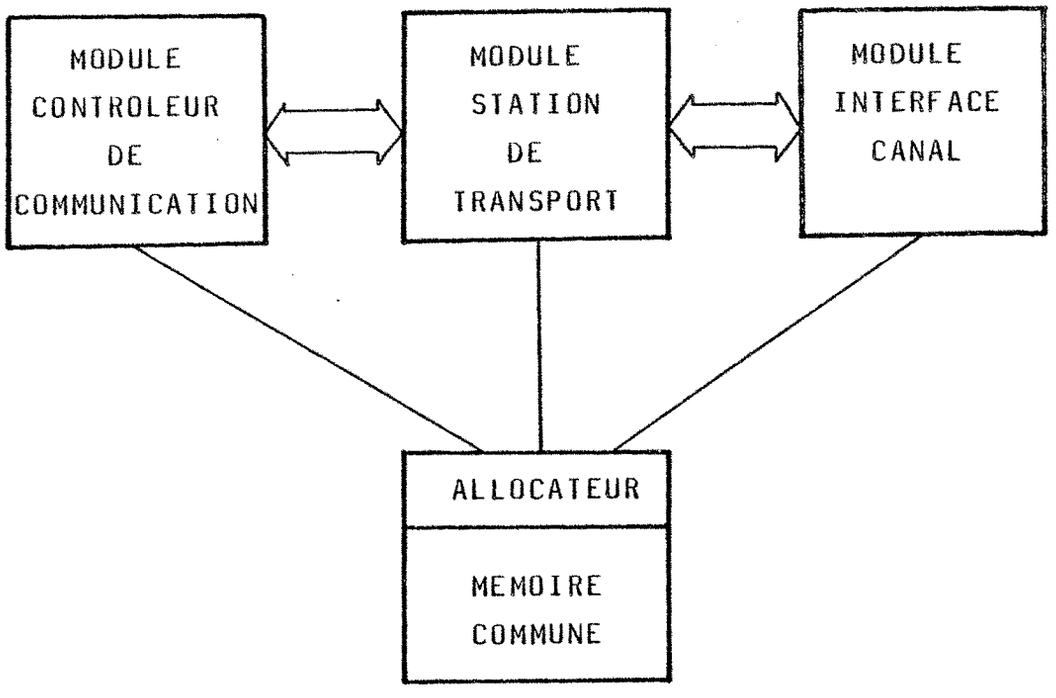


Fig.V-5 Structure matérielle de PIAR

IV-3-2 Les modules fonctionnels de PIAR

IV-3-2-1 Le Module Contrôleur de Communication (MCC)

Comme nous l'avons déjà vu, il gère la liaison entre PIAR (donc l'ordinateur hôte) et le noeud de CIGALE. Il exécute donc les fonctions de :

-1- gestion de la ligne physique : ce sont les fonctions assurées normalement par un coupleur de ligne série

-2- gestion de la ligne logique : ce sont celles de la procédure de communication, ici TMM-UCIG

-3- gestion de l'interface avec les autres modules : elle consiste à exécuter l'algorithme d'allocation des tampons de mémoire que nous décrirons plus loin. De plus, le module MCC doit prévenir le module Station de Transport en cas de panne grave sur la ligne série.

La configuration matérielle de ce module est celle annoncée au $ IV-3-1-2 : CPU (Zilog Z80), mémoire morte (2 K.octets), mémoire vive (128 octets), et comme circuits plus spécifiques : le circuit SIO Zilog qui gère la ligne physique (alors que le CPU gère la ligne logique), des circuits d'interface aux normes V-24 et des circuits de génération des signaux de contrôle et de temporisation.

Le SIO et le CPU "dialoguent" grâce au mécanisme d'interruptions vectorisées du système Zilog.

Le logiciel exécute 6 processus sous le contrôle d'un septième : MONITEUR.

Trois de ces processus sont directement liés à la gestion de la ligne série et sont activés par les interruptions du circuit SIO. Ce sont TRANCTL (émission de messages de contrôle), TRANSMESS (émission de messages de données), RECMESS (réception de messages).

Deux autres processus (EMISSION et RECEPTION) exécutent des algorithmes de la procédure de communication.

Le dernier (INTMEM) gère l'interface du MCC avec les autres modules.

IV-3-2-2 Le Module Station de Transport (MST)

Sa caractéristique principale est de ne pas avoir d'interface avec l'extérieur. Il exécute les fonctions attribuées à la SI Externe lors de la décomposition de la Station de Transport SI2.

Sa structure matérielle est également bâtie autour d'un microprocesseur Zilog Z80 et comprend 8 K.octets de mémoire morte et 4 K.octets de mémoire vive contenant, entre autre, les tables décrivant les flots.

On trouve également les circuits générateurs des signaux de contrôle, et un circuit compteur de temporisations : le CTC.

Le logiciel est constitué de 5 processus inter-réagissant entre eux et directement issus de ceux de la ST2 portable :

- P-INTRES gère l'interface entre la ST externe et la ST Résiduelle
- P-EMIS prépare l'envoi des messages sur le réseau
- P-RECEP traite les messages reçus de la ligne et transférés par le module MCC
- P-REVEIL gère les temporisations lancées par les autres processus et est activé par le circuit CTC
- P-MONITEUR gère les événements internes et externes du module MST en activant les autres processus et en reprenant la main après leur traitement.

IV-3-2-3 Le module Interface Canal (MIC)

C'est le module qui gère la liaison de PIAR avec l'ordinateur hôte et il représente une application de l'UCB que nous venons de décrire.

Nous détaillerons ce module au cours du paragraphe suivant.

IV-3-3 Les mécanismes d'intercommunication

IV-3-3-1 Fonctionnement général du système

L'empilement des protocoles dans le réseau CYCLADES conduit à une propagation de l'activité telle que le protocole de plus haut niveau serait à la pointe d'un triangle et celui de plus bas niveau à la base. L'activité se déplaçant du haut vers le bas en émission et à l'inverse en réception.

Nous retrouvons dans le système PIAR une propagation de l'activité de module en module (fig.IV-6).

En réception, le module MCC, activé par les messages venant de la ligne série, traite des messages et transfère vers le module MST les blocs de données exempts d'erreurs.

Ce module exécute alors les actions prévues par le protocole de transport et normalement envoie le bloc de données vers le module MIC, qui prévient le canal de la présence de données à lire, puis le transfert s'effectue alors octet par octet sur le bus de données du canal multiplex.

En émission les blocs de données sont reçus par le module MIC et suivent le chemin inverse.

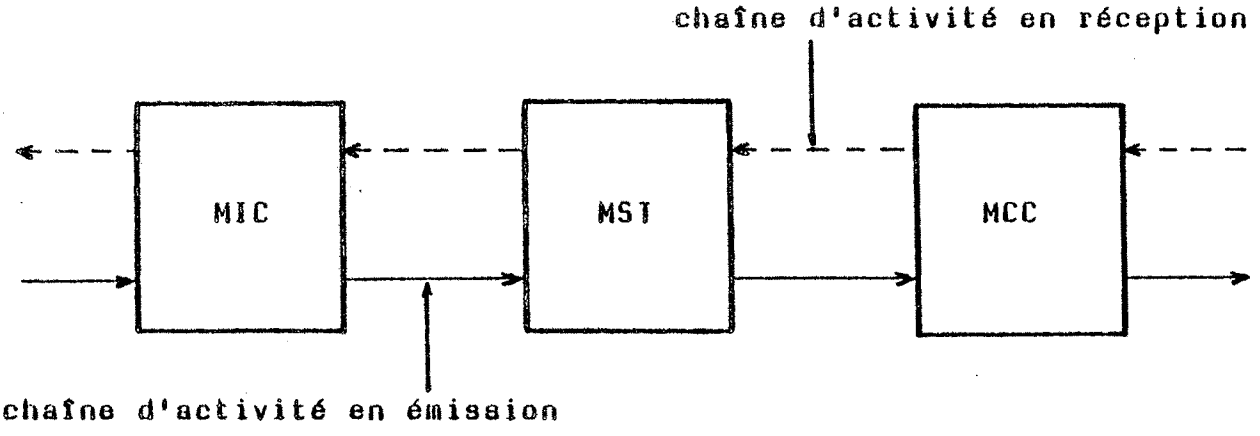


Fig.IV-6 Propagation de l'activité dans le système PIAR

Remarque : A l'intérieur du système PIAR les transferts de données entre les modules sont seulement logiques : les informations restent au même emplacement physique.

IV-3-3-2 Les ressources de communication

Ce sont les 2 ressources partageables par les 3 modules fonctionnels en l'occurence la Mémoire Commune et le Bus parallèle.

-1- La mémoire commune de 4 K.octets est divisée en 10 tampons de taille fixe et égale qui matérialisent les liens fonctionnels entre les modules, et sont utilisés comme des boîtes aux lettres. Les modules n'ont donc aucune liaison directe entre eux. Par contre chaque lien fonctionnel se voit attribué un groupe de tampons.

L'allocateur de mémoire a pu être implémenté dans le logiciel car chaque module travaille le plus souvent dans son environnement privé et en raison également du débit global du système, limité par la ligne série vers CIGALE.

Il consiste essentiellement à la gestion d'un automate d'états finis par groupe de tampons. Un état indique le propriétaire instantané du tampon, et une transition représente le changement de ce propriétaire. Cette transition est faite par l'écriture d'une seule variable (PROPI) contenue dans le 1er octet de chaque tampon ; cette opération est rendu indivisible par l'allocateur de bus.

La règle générale du mécanisme d'allocation est que tous les modules accèdent en lecture à tous les tampons (ce qui permet l'attente active d'une attribution) mais que seul le propriétaire instantané d'un tampon y accède en écriture.

La tâche consommatrice finale d'un tampon le rend à son propriétaire permanent indiqué dans le deuxième octet du tampon.

L'algorithme d'allocation est distribué et exécuté dans chaque module de façon indépendante et asynchrome.

-2- Le bus parallèle est formé de 21 lignes partagées entre les 3 modules dont un seul peut en contrôler l'état à un instant donné. Chaque module peut ainsi mettre son bus interne en liaison avec le bus parallèle à l'aide de portes "3-états" assurant également l'isolement.

L'utilisation du bus parallèle ne permettant que l'accès des modules à la mémoire commune, les 21 lignes sont classiquement :

- 8 lignes bi-directionnelles de données
- 12 lignes d'adresses
- 1 ligne de lecture/écriture.

Les accès à la mémoire sont limités à une opération (environ 500 ns) ; l'allocateur de bus (fig.IV-7) a donc été câblé pour respecter l'inégalité :

temps de réponse de l'allocateur	$\ll$	temps d'utilisation de la ressource

Il effectue les fonctions suivantes :

- résolution des conflits d'accès au bus parallèle avec l'ordre de priorité suivant :

MIC > MST > MCC

- mémorisation des requêtes
- indivisibilité d'un accès élémentaire.

Chaque module communique avec l'allocateur de bus à l'aide de 2 lignes de requête ($\overline{R}$) et d'autorisation ($\overline{G}$).

Remarque : Cet allocation n'est pas équitable et pourrait conduire à un blocage de la requête d'un module de plus faible priorité, si la mémoire commune ne contenait pas que des données globales, ce qui n'est pas le cas de PIAR.

IV-3-3-3 La synchronisation entre les modules

Elle est faite implicitement par l'algorithme d'allocation des tampons.

IV-4 LE MODULE INTERFACE CANAL

Nous avons vu qu'il constituait une application de l'UCB et nous le décrirons donc à un double niveau : fonctionnement du système PIAR dans le contexte des choix effectués lors de la conception de l'UCB.

IV-4-1 Rappel des tâches à effectuer

Ce module ne fait aucun traitement sur les blocs de données qu'il transfère. Il exécute uniquement des fonctions d'interface.

- interface avec l'ordinateur hôte
- interface avec les autres modules du sytème.

Les premières sont effectuées par la partie Interface Canal de l'UCB et ont été décrites en détail au chapitre III, aussi bien d'un point de vue matériel que logiciel. Nous préciserons seulement au cours de ce paragraphe les différentes définitions des états-unité, bits du mot d'analyse, etc... utilisés par le système PIAR vis-à-vis du canal.

La liaison matérielle avec les autres modules du système obéit à la nécissité commune : fournir les lignes d'accès à la mémoire partagée et à l'allocateur de bus.

Le logiciel d'interface avec l'ordinateur hôte est complètement assuré par le logiciel INTCAN de l'UCB décrit au $ III-4-2. Celui qui assure la communication avec les autres modules (INTMEM) exécute essentiellement l'algorithme d'allocation des tampons de mémoire.

Les mécanismes de communication entre les logiciels de ce module sont décrits au $ III-4-4.

Dans ce qui suit, il ne sera donc question que de la réalisation de l'interface entre le module MIC et les autres modules du système PIAR. Ce qui, d'un autre point de vue, correspond à la présentation d'une réalisation concrète de la partie Interface Dispositif de l'UCB (matériel et logiciel).

IV-4-2 Aspects matériels

Il s'agit seulement de fournir les 21 lignes du Bus Parallèle et les 2 lignes de dialogue avec l'allocateur de Bus.

Une des contraintes imposées par l'UCB est que le microprocesseur 8X300 ne dispose pas de bus d'adresse pour accéder aux données (l'espace d'adressage est réduit à l'accès à la mémoire de programme).

IV-4-2-1 Génération du Bus Parallèle

Nous avons donc dû utiliser 21 des 24 lignes offertes par l'interface dispositif de l'UCB. Ces lignes sont issues de circuits d'E/S compatibles avec le CPU, dont les caractéristiques sont données dans l'annexe 2.

Rappelons les principales :

- 8 lignes bi-directionnelles programmables
- mémorisation des informations
- indépendance des accès CPU et milieu extérieur (avec priorité donnée à ce dernier en entrée)
- les entrées de l'utilisateur peuvent être synchrones ou asynchrones par rapport à l'horloge du CPU
- sorties "3-états".

Les lignes de liaison avec l'extérieur sont programmées par l'allocateur de bus. Les lignes d'adresses sont positionnées en sortie. Le sens de transfert sur les lignes de données est commandé par la ligne de lecture/écriture (R/W) avec :

R/W = 1 : écriture dans la mémoire commune
R/W = 0 : lecture - - - -

Nous utiliserons donc 3 circuits de ce type : 2 pour générer les 12 lignes d'adresse et la ligne R/W, 1 pour générer les 8 lignes de données. Les 3 positions restantes, dans le boîtier générant les 4 bits poids fort de l'adresse, sont utilisées pour ranger des états supplémentaires.

La sélection interne (côté microprocesseur) des boîtiers se fait par la technique décrite au $ III-3-1-3 et les commandes générées à partir de l'extension, soit : $\overline{\text{ADHI}}$ et $\overline{\text{ADLO}}$ pour le positionnement des adresses, et $\overline{\text{DATA}}$ pour l'écriture et la lecture des données.

Remarque : La mémorisation des informations au niveau des boîtiers d'E/S facilite les traitements sur les informations (surtout les adresses) sans utiliser les registres ni la mémoire interne.

IV-4-2-2 Le Circuit de synchronisation

Il répond à la conception câblée de l'allocateur de bus. Il génère une ligne de demande ($\overline{R}$) et reçoit une ligne d'acquittement ($\overline{G}$). Cependant il est activé à partir d'une instruction par l'intermédiaire des commandes $\overline{\text{RQST}}$ et $\overline{\text{SUPRQ}}$ fournies par l'extension et qui, respectivement, positionne et désactive la requête $\overline{R}$.

De plus la requête $\overline{R}$ place le CPU dans l'état "halte", dans lequel l'horloge reste active mais où tout traitement est suspendu.

Remarque : Pour que l'instruction en cours au moment de l'émission de la commande $\overline{RQST}$ soit bien exécutée, nous ne validons cette commamde qu'au milieu du cycle .

Le circuit de synchronisation ainsi que le bus parallèle et son allocateur (simplifié) est représenté à la fig.IV-7.

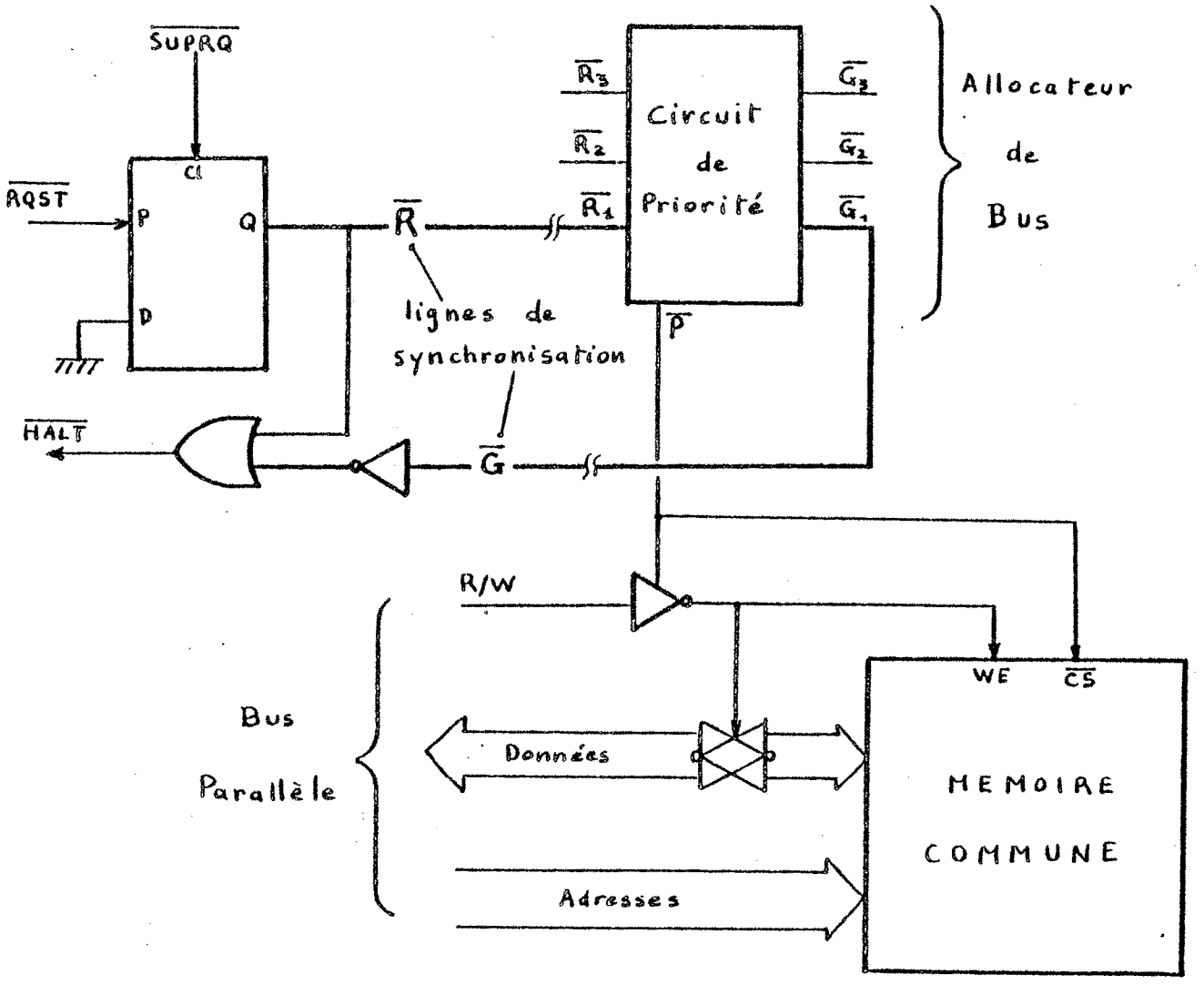


Fig.IV-7 Mécanisme d'accès à la mémoire commune

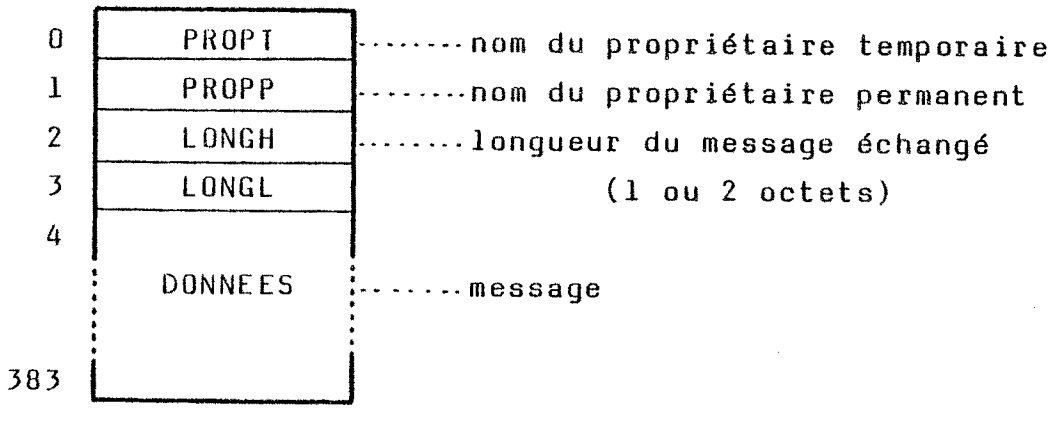
IV-4-3 Aspects Logiciels

Nous décrivons ici plus en détail le mécanisme d'allocation des tampons de la mémoire commune citée au IV-3-3-2, ainsi que les mécanismes élémentaires d'accès à cette mémoire en faisant de nouveau apparaître la spécificité de la programmation du 8X300.

IV-4-3-1 L'allocateur de mémoire

Nous avons vu qu'il était implanté dans le logiciel et distribué dans chaque module.

Les tampons qui composent la mémoire commune sont d'une part donnés à un "propriétaire permanent" et d'autre part alloués à un "propriétaire temporaire". Leur taille est de 384 octets dont les 4 premiers contiennent les informations suivantes :



Le mécanisme d'allocation, appliqué à MIC, peut être résumé de la façon suivante :

- Pour savoir s'il dispose d'un tampon plein (réception d'un message émis par un autre module) le module doit lire la variable PROPT des tampons dont il n'est pas le propriétaire permanent. Ce message est obligatoirement destiné à la Station Résiduelle et le module doit prévenir le CPU via le canal pour qu'il exécute une commande de lecture.

Le tampon doit ensuite être rendu à son propriétaire permanent.

- Pour savoir s'il dispose d'un tampon vide, le module doit lire la variable PROPT des tampons dont il est propriétaire permanent. Si le canal remplit un tampon (message envoyé par la Station Résiduelle) le module MIC doit l'affecter au module MST.

Il nous faut introduire certains états et variables supplémentaires pour que le logiciel INTMEM effectue l'algorithme correspondant. Ce sont :

- 4 états :
  - BUV indique la présence d'un tampon vide
  - BUVANT est l'état antérieur de BUV; il indique qu'un tampon vient d'être rempli si BUV = 0
  - BUP indique la présence d'un tampon plein
  - BUPANT est l'état antérieur de BUP; il indique qu'un tampon vient d'être vidé si BUP = 0.

- 6 variables :
  - ADHOBUV et ADLOBUV forment l'adresse du tampon vide (éventuellement qui vient d'être rempli)
  - ADHIBUP ET ADHIBUV forment l'adresse du tampon plein (éventuellement qui vient d'être vidé)
  - LONGH et LONGL forment le nombre d'octet du message.

Ces variables sont rangées dans la mémoire interne RAMI décrite au $ III-3-1-4, et seront utilisées pour tout accès ultérieur au tampon correspondant, soit lors des séquences de transfert de données activées par le logiciel INTCAN, soit lors des séquences de changement de propriétaire temporaire après ces transferts.

Remarque : Seuls BUV et BUP apparaissent dans le logiciel INTCAN.

La fig.IV-8 résume le logiciel INTMEM dont la description complète est donnée en annexe 3 planches O à Q.

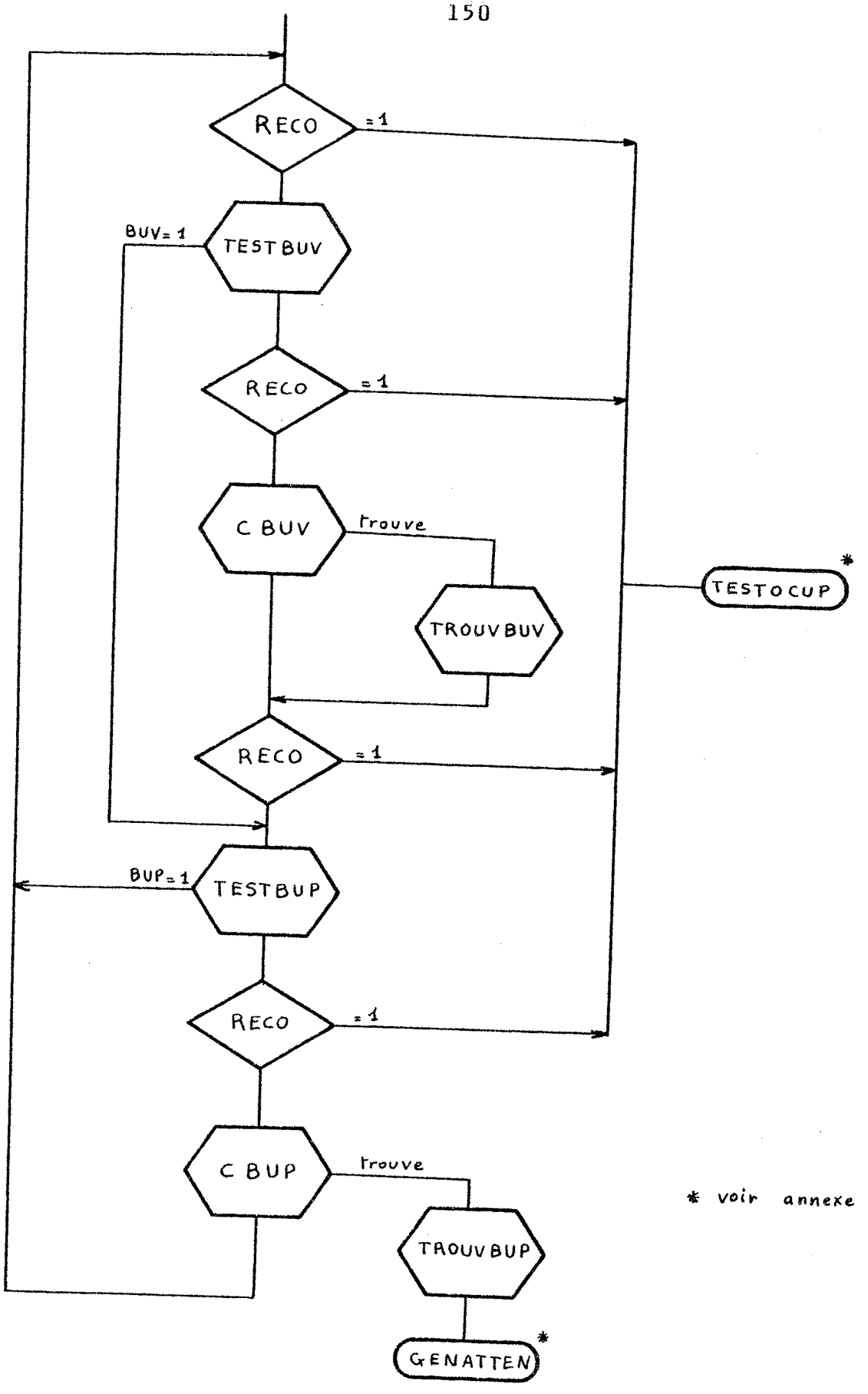


Fig.IV-8 Organisation simplifiée du logiciel INTMEM

IV-4-3-2 Les accès élémemtaires à la mémoire commune

La principale difficulté vient de ce que le cycle d'instruction du 8X300 est très court et donc que la durée des signaux de contrôle et des données (env. 60 ns) n'est pas compatible avec le temps d'accès à cette mémoire (450 ns).

Un accès se décomposera donc en 5 phases :

-1- Positionnement de l'adresse et de la ligne R/W, à l'aide d'un nombre variable d'instructions (selon l'endroit où se trouvent rangés les 2 octets constituant l'adresse). Par exemple, le cas le plus défavorable, où l'adresse avait été sauvegardée dans RAMI, nécessite 5 instructions :

```
(1)  XMIT   IVR,ADLOBUV
(2)  MOVE   RB7,0,LB7      z,z,ADLO
(3)  XMIT   IVR,ADHIBUV
(4)  MOVE   RB7,0,LB7      z,z,ADHI
(5)  XMIT   RWRAM,1,x      z,z,ADHI
```

(x indique la valeur de R/W, et RWRAM sa position dans le registre d'E/S)

-2- Positionnement de la requête par la commande $\overline{\text{RQST}}$. Cette action est entreprise simultanément avec la dernière instruction de la séquence précédente dans le cas d'une lecture en mémoire commune. Dans l'exemple donné ci-dessus, on aurait alors :

```
(5)  XMIT   RWRAM,1,x      z,RQST,ADHI
```

Par contre, dans le cas d'une écriture dans cette mémoire, la requête se fera en même temps que l'envoi de la donnée :

```
(6)  MOVE   R2,0,LB7       z,RQST,DATA
```

(si la donnée à envoyer se trouve dans le registre R2)

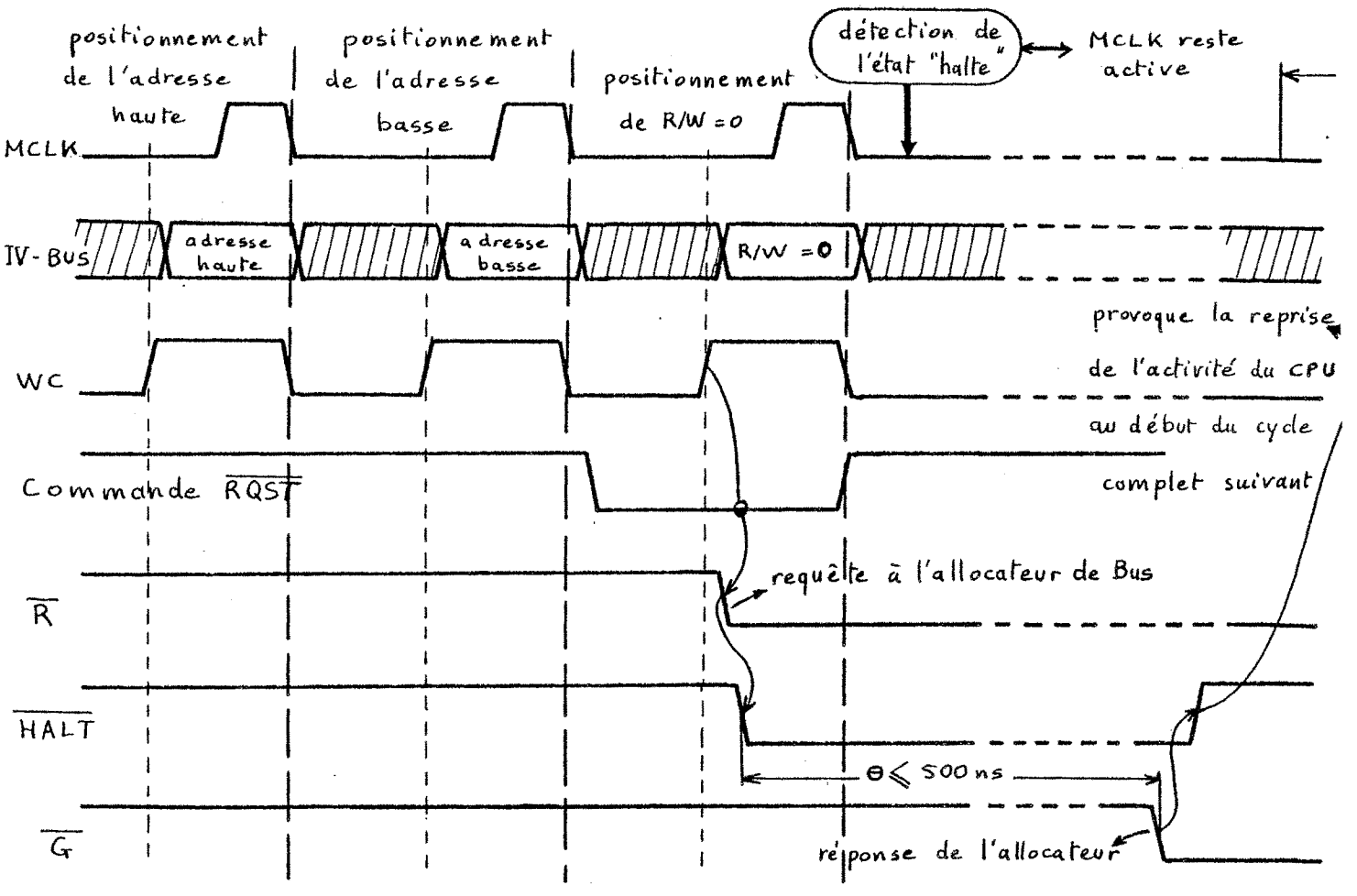


Fig.IV-9 Diagramme d'une lecture

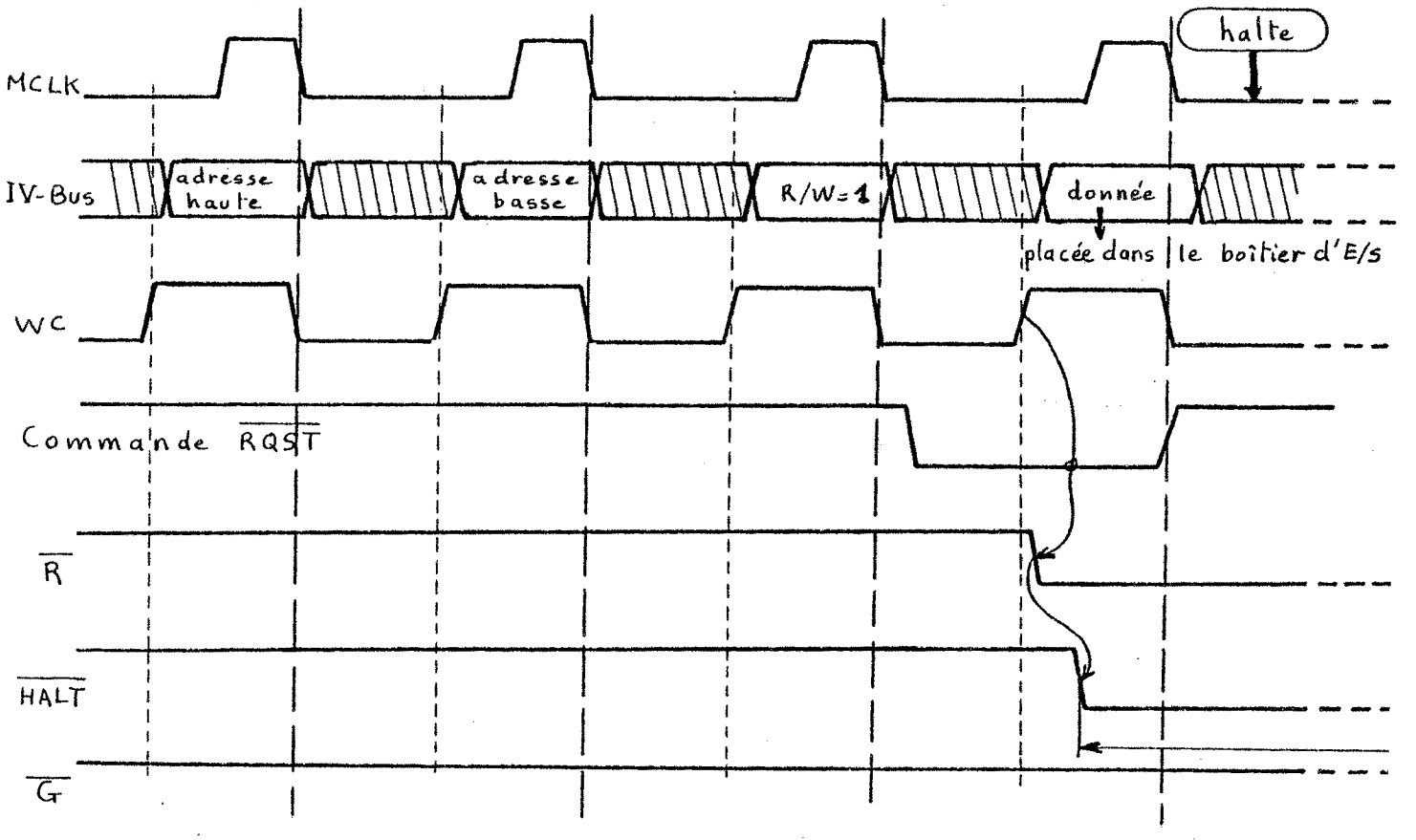
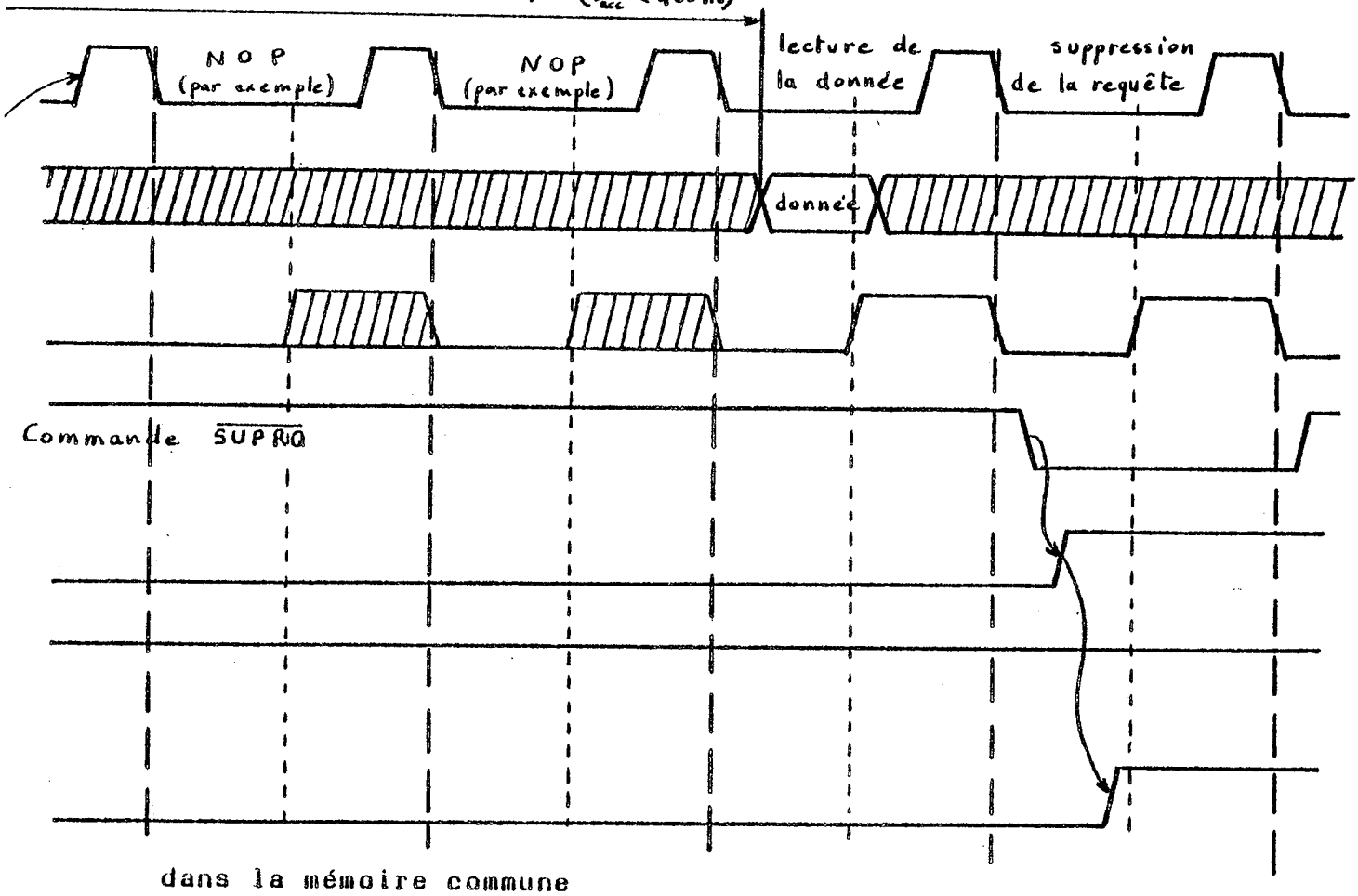
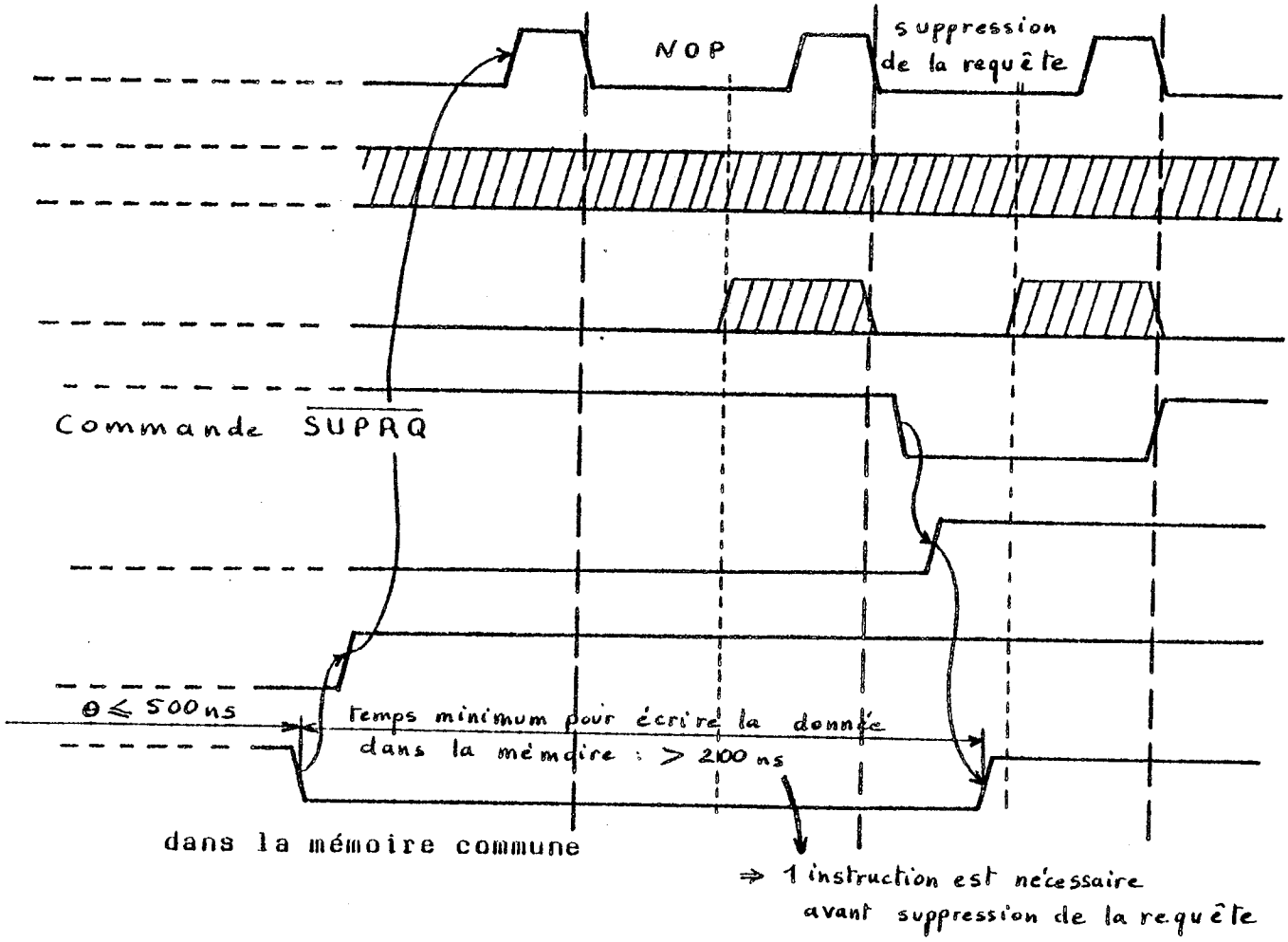


Fig.IV-10 Diagramme d'une écriture

temps (> 500ns) nécessaire, et suffisant, pour que la donnée soit dans le boîtier d'E/S ($t_{acc} \leq 450$ns)





⇒ 1 instruction est nécessaire avant suppression de la requête

-3- Attente de l'acceptation de la requête qui correspond à la suspension du programme ("halte" du microprocesseur) et qui peut durer au maximum 500 ns (durée de l'accès élémentaire d'un autre module) puisque le module MIC est le plus prioritaire.

-4- Délai de 1 (en écriture) ou 2 (en lecture) instructions après la reprise du programme, dû au cycle plus lent de la mémoire commune de PIAR. En général ces instructions seront les appels à des sous-programmes.

-5- Lecture de la donnée le cas échéant et/ou la suppression de la requête par la commande $\overline{\text{SUPRQ}}$. On aura donc :

```
NOP                    z,SUPRQ,z
```

précédé, en lecture seulement, de :

```
NOP   LB7,0,R2     z,z,DATA
```

On remarque donc qu'une opération de lecture dure 1 µs et une écriture seulement 500 ns, à partir du positionnement de la requête jusqu'à sa suppression, plus le temps de réponse de l'allocateur de bus : .

Les fig.IV-9 et IV-10 donnent les diagrammes temporels correspondant à ces 2 opérations élémentaires d'accès à la mémoire commune.

IV-4-4 Comportement du système PIAR vis-à-vis du canal

Le résultat des recherches des tampons "vide" et "plein" met le système dans un certain état par rapport au canal. Cet état représente la possibilité qu'aura le canal d'émettre une commande donnée, mais également peut imposer l'émission d'une commande appropriée.

IV-4-4-1 Généralités

Nous prendrons les exemples suivants de 2 éventualités concrètes pour PIAR :

-1- La Station Résiduelle veut envoyer un message à la Station Externe :

Ceci, ramené au niveau du canal, se traduira par l'envoi d'une commande d'écriture. Cet envoi peut alors soit être immédiat au risque d'une réponse négative ("occupé") si le module MIC ne dispose pas de tampon vide (BUV=0), soit être précédé d'une commande d'analyse. A cet effet, nous avons introduit 2 indicateurs dans l'octet d'analyse :

- WRITEN indique la présence d'un tampon vide
- READEN - - - - - plein.

Ces 2 indicateurs sont positionnés lors des séquences TROUVBUV et TROUVBUP de INTMEM, et sont annulés par INTCAN à la fin des séquences de transfert d'octets correspondantes.

-2- La Station Externe veut envoyer un message à la Station Résiduelle :

Comme précédemment, si le programme de la Station Résiduelle le prévoit, le bit READEN de l'octet d'analyse peut indiquer la possibilité, et même plutôt dans ce cas la nécessité, de lancer une commande de lecture.

Cependant il est possible de prévenir le canal d'une façon asynchrone, par l'envoi d'un état-unité "Attention" ce qui provoque normalement une interruption à l'unité centrale, donc au programme, qui devra alors lancer une commande d'analyse ou même directement de lecture.

La séquence GENATTEN effectue cette demande asynchrone (annexe 3 planche M)

IV-4-4-2 Les différents états-unité du système PIAR

Nous rappellerons qu'une unité d'E/S informe le canal sur son état, celui de l'organe d'E/S et sur celui de l'échange en cours, d'une part à l'initialisation d'une commande ou lors d'une instruction Test I/O, et d'autre part à la fin d'un échange de données.

Cette information se fait sous la forme d'un "état-unité initial" et d'un "état-unité de fin d'échange". Elle est ensuite fournie par le canal au programmeur dans le CSW.

Dans le cas du système PIAR en tant qu'unité d'E/S, nous donnons dans le tableau de la fig.IV-11, les significations des états-unité que nous lui avons attribué.

Remarques :

-1- par rapport à l'UCB, nous n'utilisons pas l'indicateur de chaînage de commande (CHAICOM), ni la séquence courte d'unité occupée (CUBS) détectée par l'indicateur OCCUPE. Les 2 bits correspondants dans l'octet des états internes ont été remplacés par BUVANT et BUPANT.

-2- la combinaison des états "fin sur canal - fin sur unité - exception sur unité", a ici le même sens que dans une unité de contrôle de transmission (type 2701/02/03) où elle signifie "fin de transmission" (EOT), <IBM7>.

-3- nous avons effleuré dans ce paragraphe, des aspects de la programmation , au niveau du 360, pour l'accès à PIAR. Nous y reviendrons au cours du chapitre suivant.

a) Etat-unité initial :

nom	code hexa	significations
zéro	0	commande acceptée
occupé (BUSY)	10	pas de tampon disponible
erreur sur unité (UCK)	02	-1- commande rejetée (non décodée) avec COMMREJCT = 1 dans l'octet d'analyse -2- erreur de parité dans la commande, avec BUSOCHK = 1
fin sur canal et fin sur unité (CE.DE)	0C	commande immédiate (NOP est la seule acceptée par l'UCB et PIAR)

b) Etat-unité final :

nom	code	significations
fin sur canal et fin sur unité	0C	tout s'est bien passé
fin sur canal et fin sur unité et exception sur unité (CE.DE.UE)	0D	. plus de place, ou plus de donnée à envoyer . se produit quand le compte dans le CCW égale la taille du tampon (EOT)
fin sur canal et fin sur unité et erreur sur unité (CE.DE.UCK)	0E	une anomalie a eu lieu pendant le transfert : - erreur de parité en écriture (BUSOCHK = 1) - "interface disconnect" (OVERRUN = 1)

Fig.IV-11 Signification des états-unité fournis par PIAR

IV-5 RESULTATS OBTENUS

Nous donnerons ici , sous forme de photographies, les séquences des échanges de données entre le module MIC du système PIAR et le canal de l'ordinateur hôte.

Ces séquences représentent une illustration du fonctionnement de l'UCB et nous ferons donc apparaître les différents modes de transfert définis au $ III-3-2 ainsi que les indicateurs de l'UCB (RECO, STAK et ST-DA) et ceux de cette application (BUV et BUP).

IV-5-1 Séquences de transfert de données

photos Fig.IV-12-13-14-15-16-17

IV-5-2 Séquences spéciales

IV-5-2-1 "Interface Disconnect" (HIO)

photos Fig.IV-18-19-20-21

IV-5-2-2 "Genatten"

photo Fig.IV-22

IV-5-3 Débits mesurés

Comme on le constate sur les fig.IV-12 et IV-13, le débit atteint pendant les échanges élémentaires est de :

- 100 K.octets/s.

Si on inclue la séquence de sélection de l'unité et celle de présentation de l'état-unité de fin de transfert, on obtient :

- 80 K.octets/s dans les modes "burst" et "multi-byte" (fig.IV-14-15-16).
- 60 K.octets/s en mode "multiplex" (fig.IV-16).

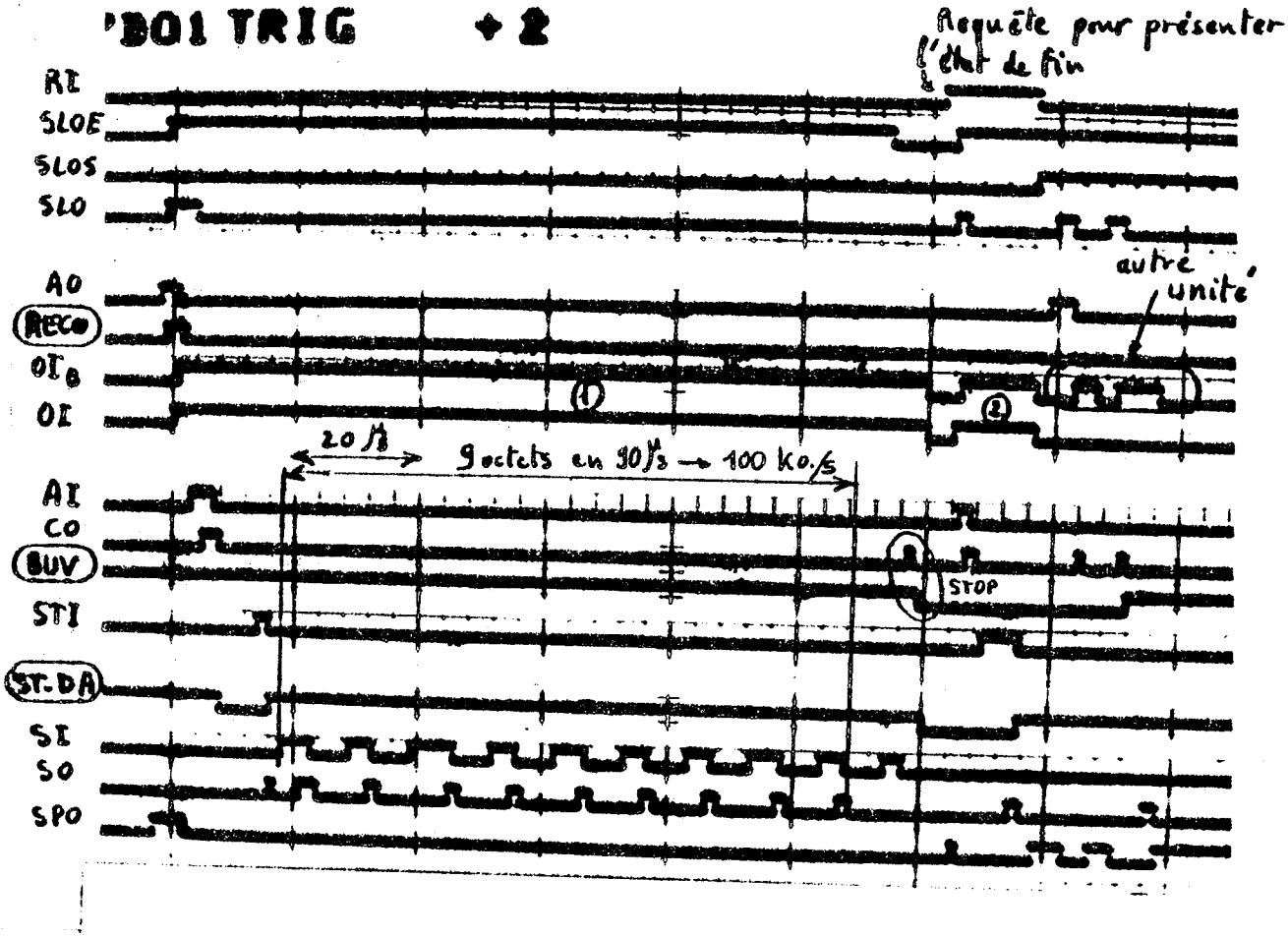


Fig.IV-12 Ecriture de 9 octets en mode "multi-byte"

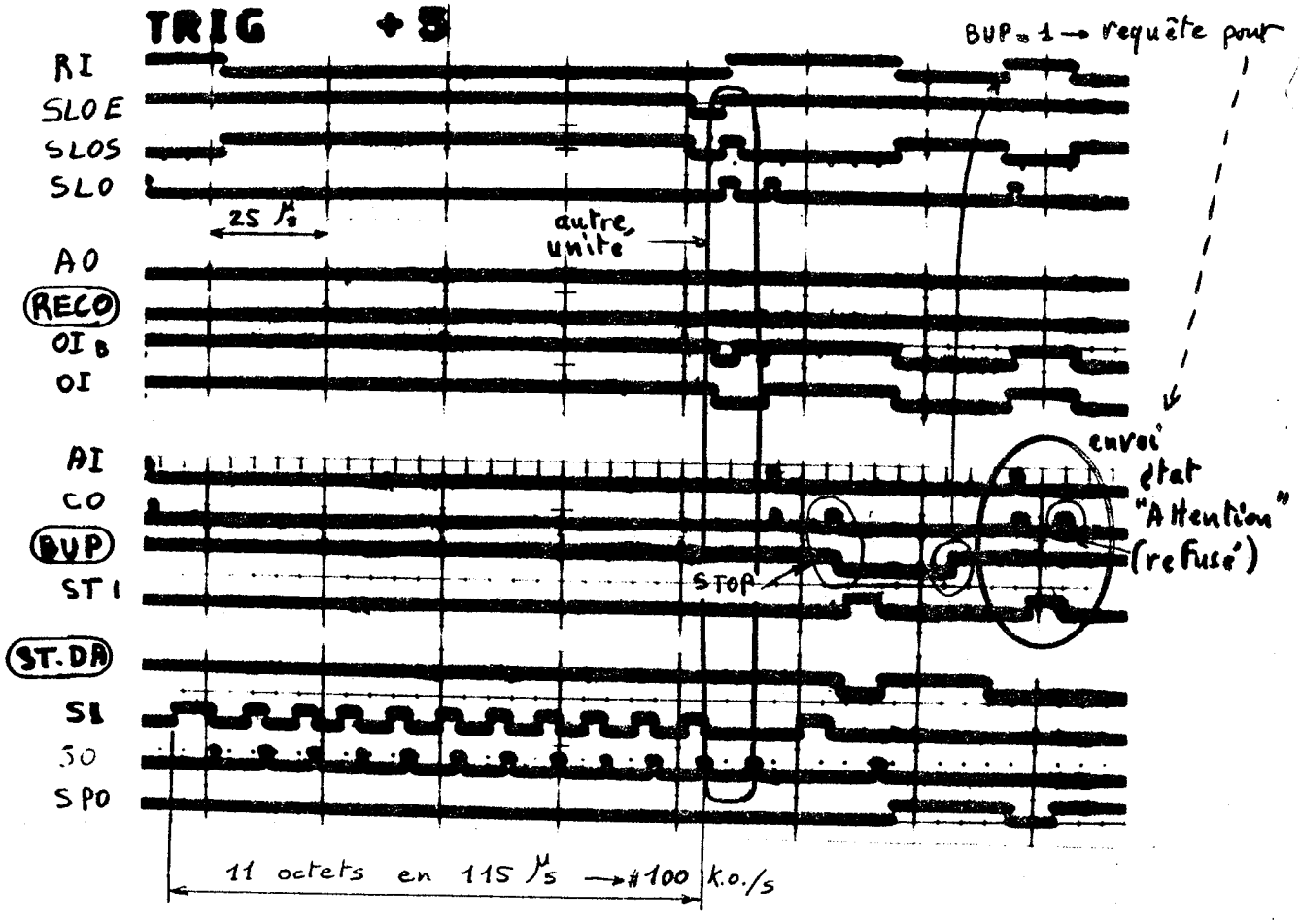


Fig.IV-13 Fin d'une lecture de 32 octets en mode "multi-byte"

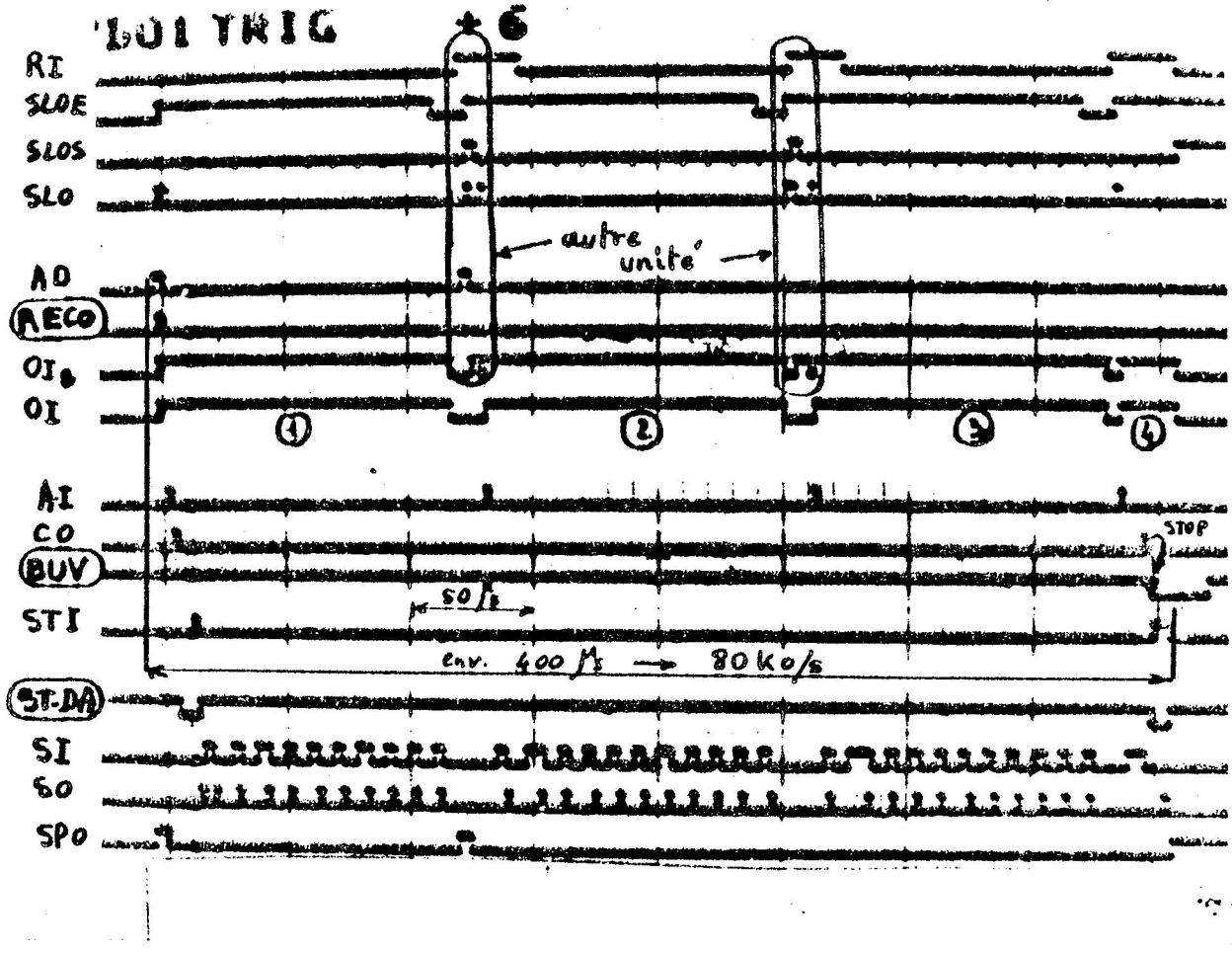


Fig.IV-14 Ecriture de 32 octets en mode "multi-byte"

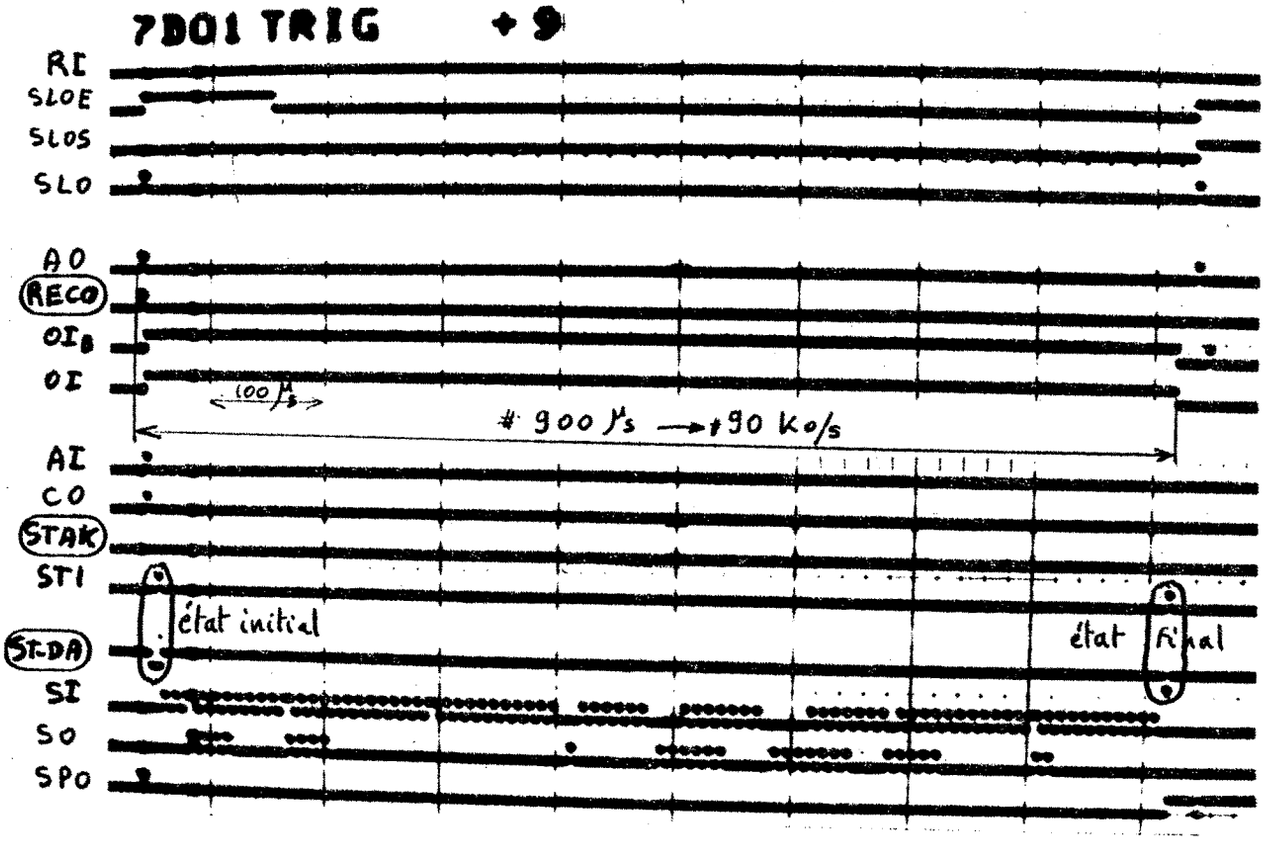


Fig.IV-15 Ecriture de 80 octets en mode "burst"

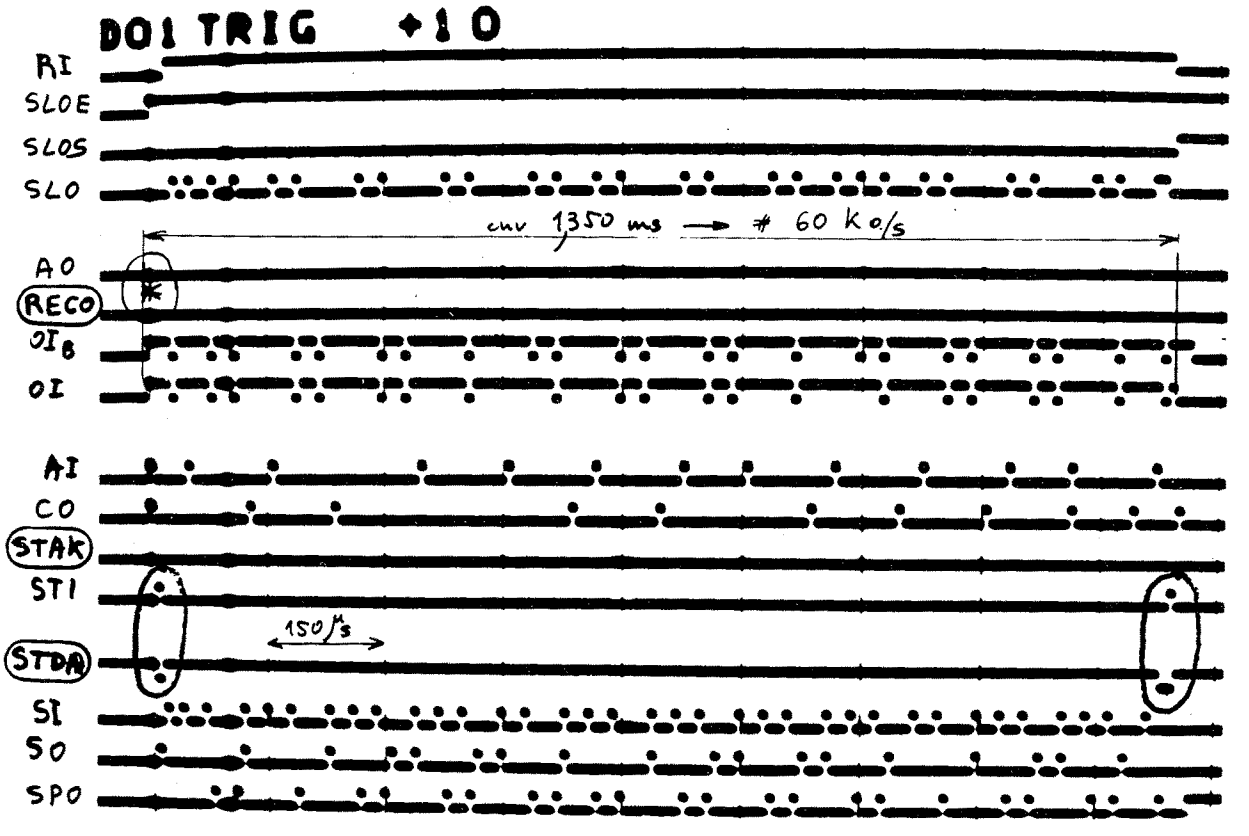


Fig IV-16 Ecriture de 80 octets en mode "multiplex"

* à cette cadence d'échantillonnage. on ne peut pas capter certains signaux brefs.

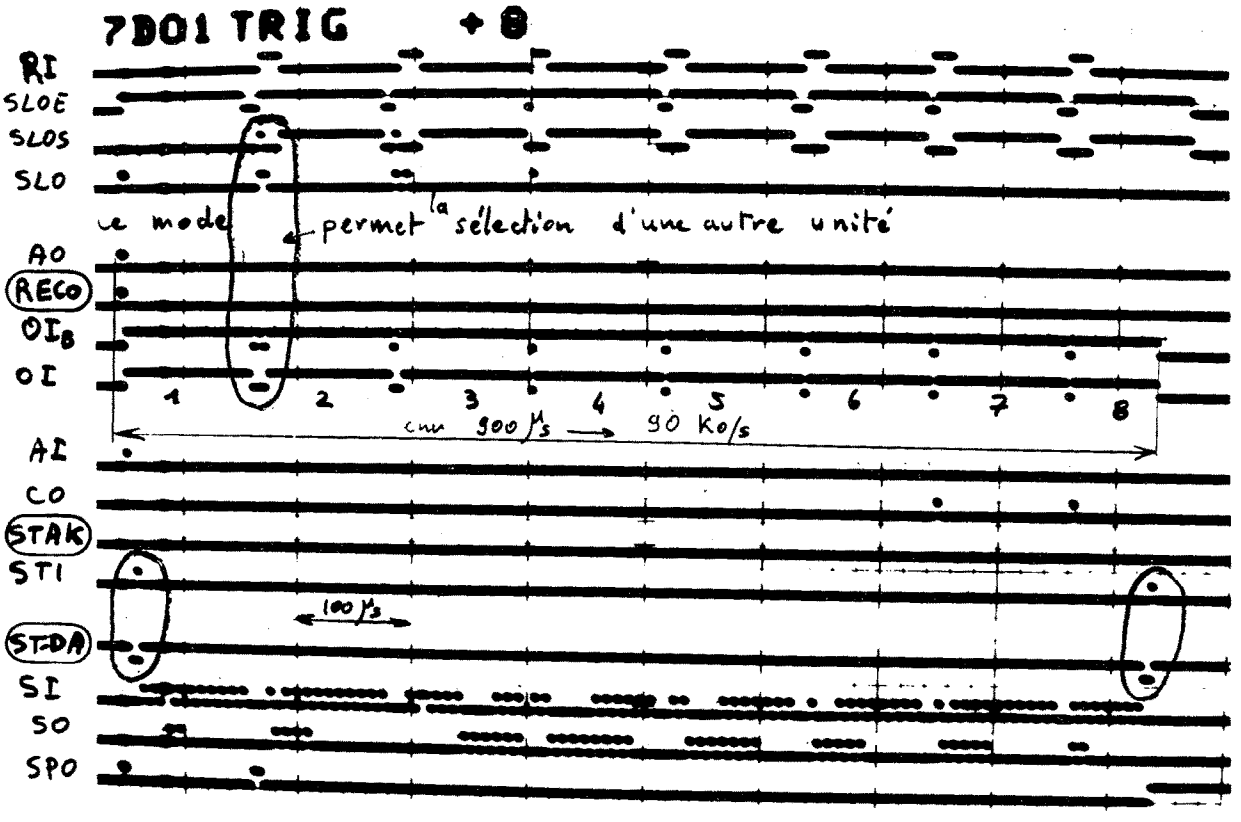


Fig.IV-17 Ecriture de 80 octets en mode "multi-byte"

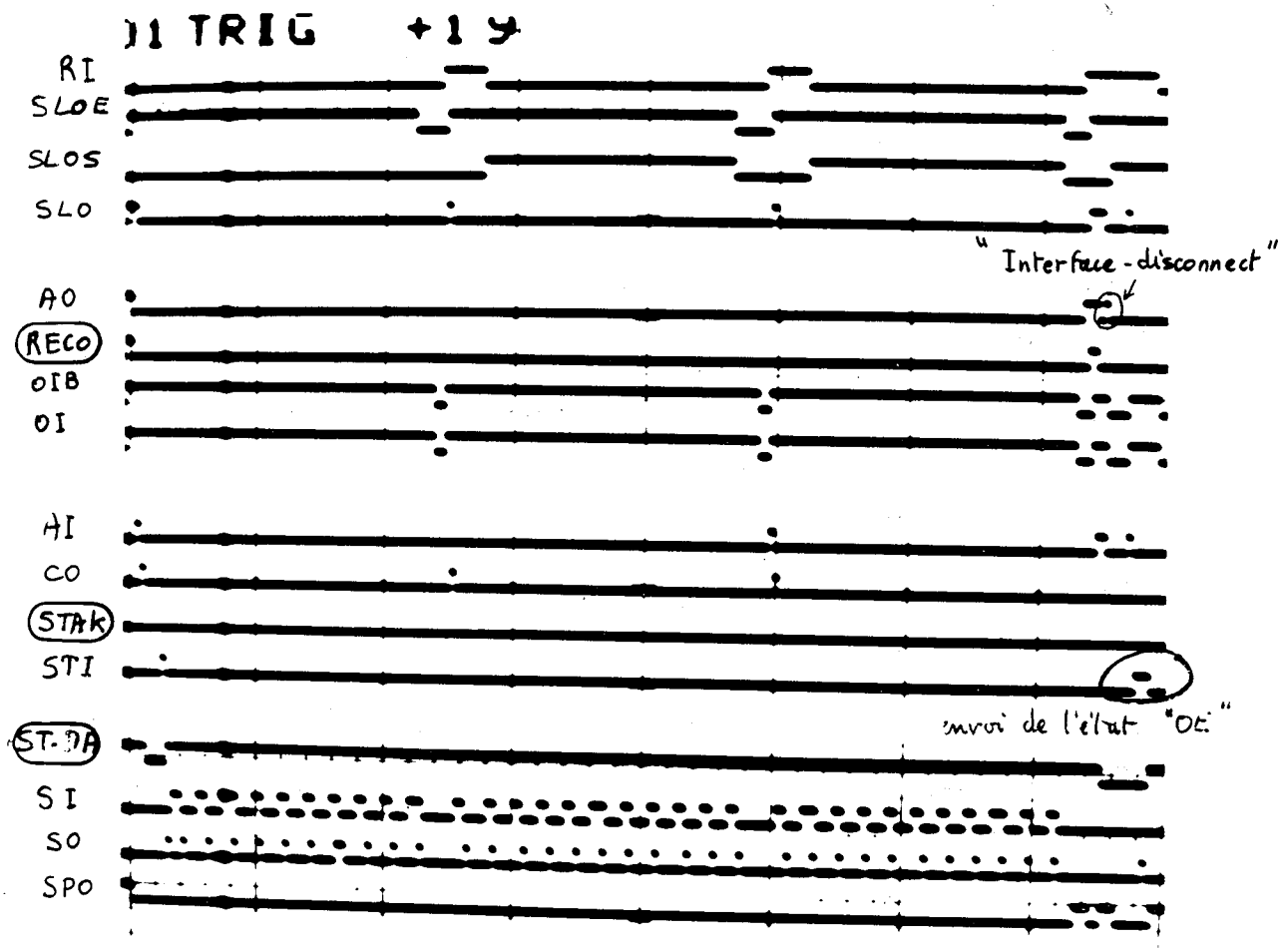


Fig IV-18 Ecriture de 32 octets terminée par un HALT I/O

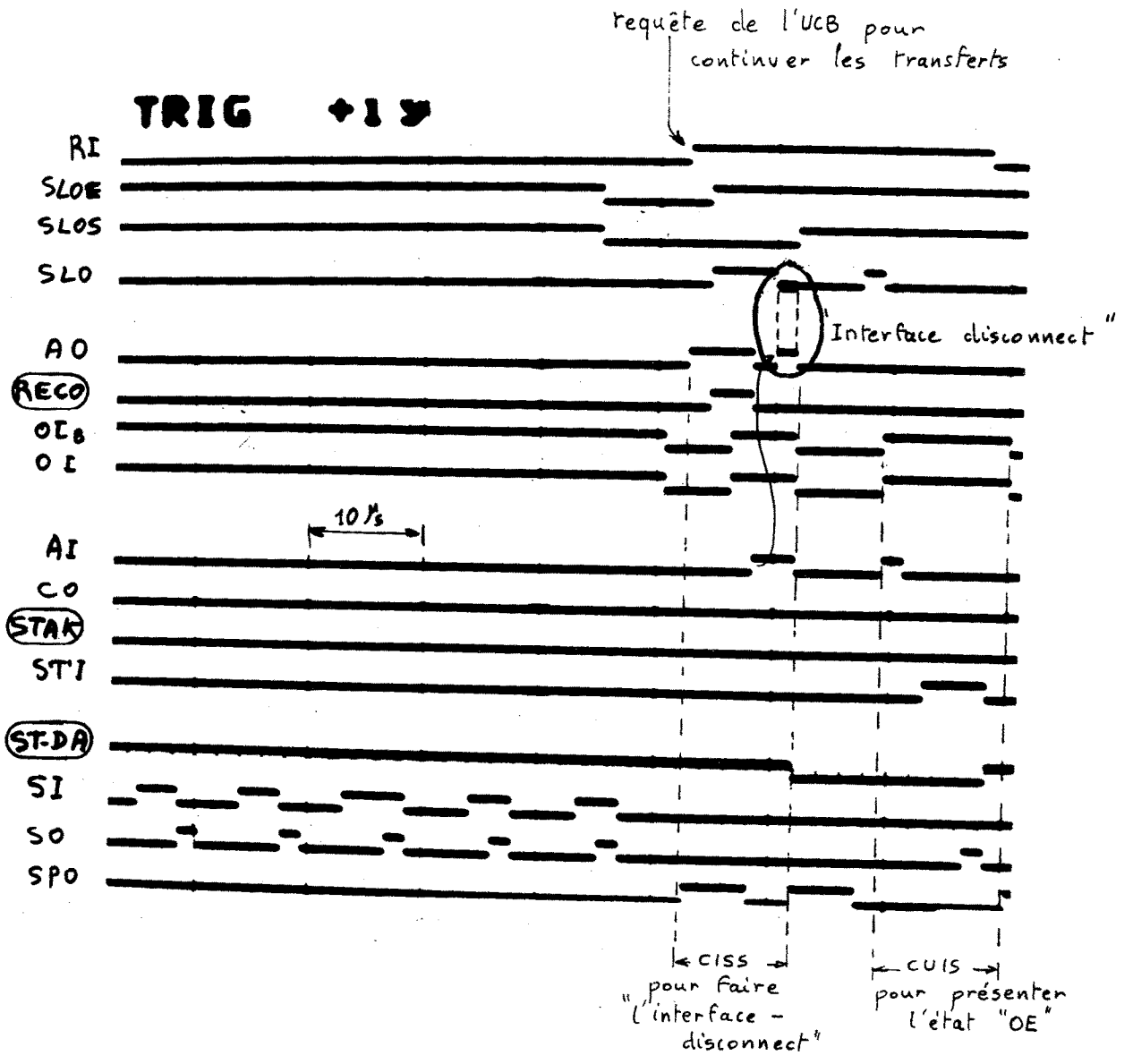


Fig.IV-19 HALT I/O pendant un transfert en mode "multi-byte"

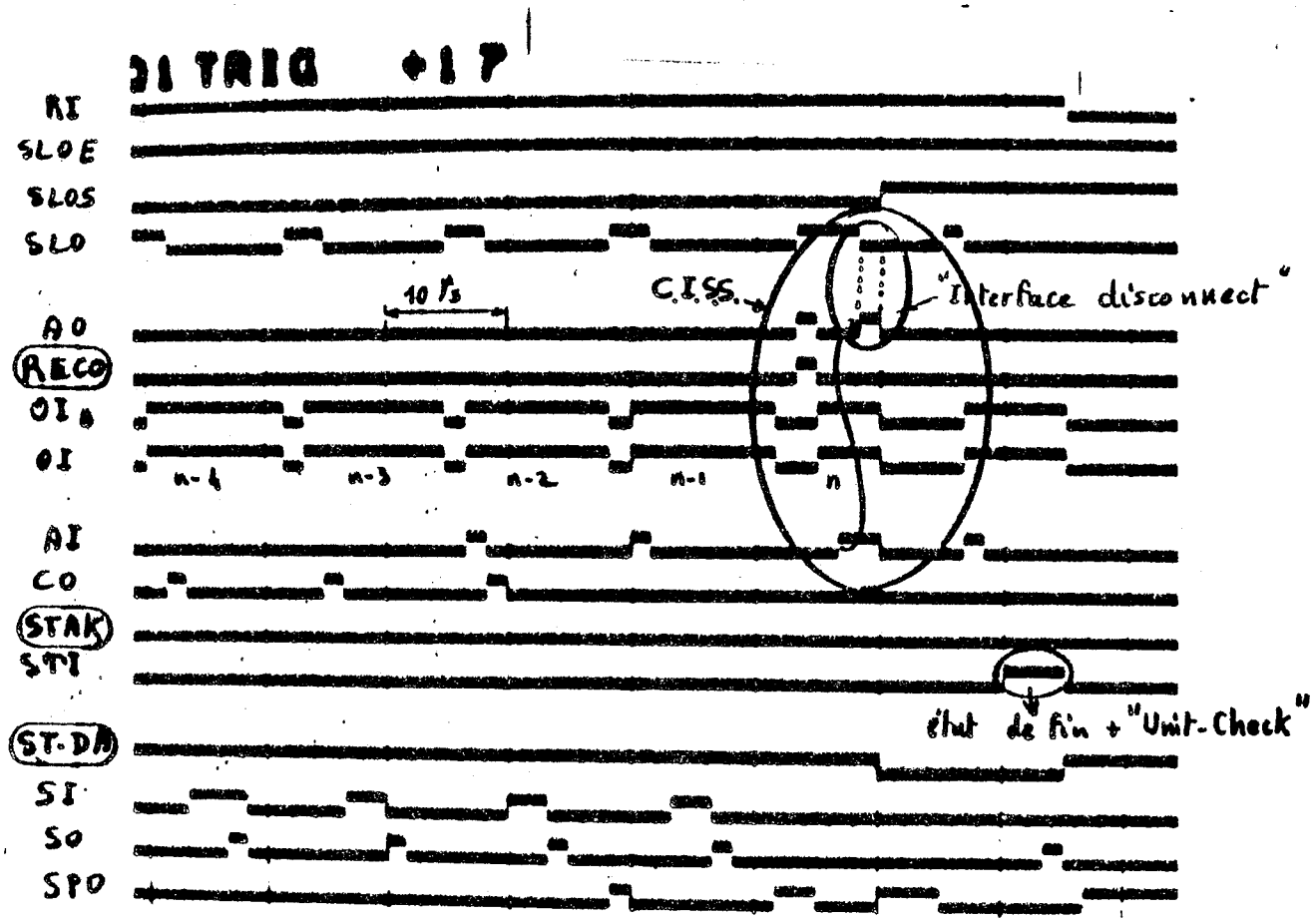


Fig.IV-20 HALT I/O pendant un transfert en mode "multiplex"

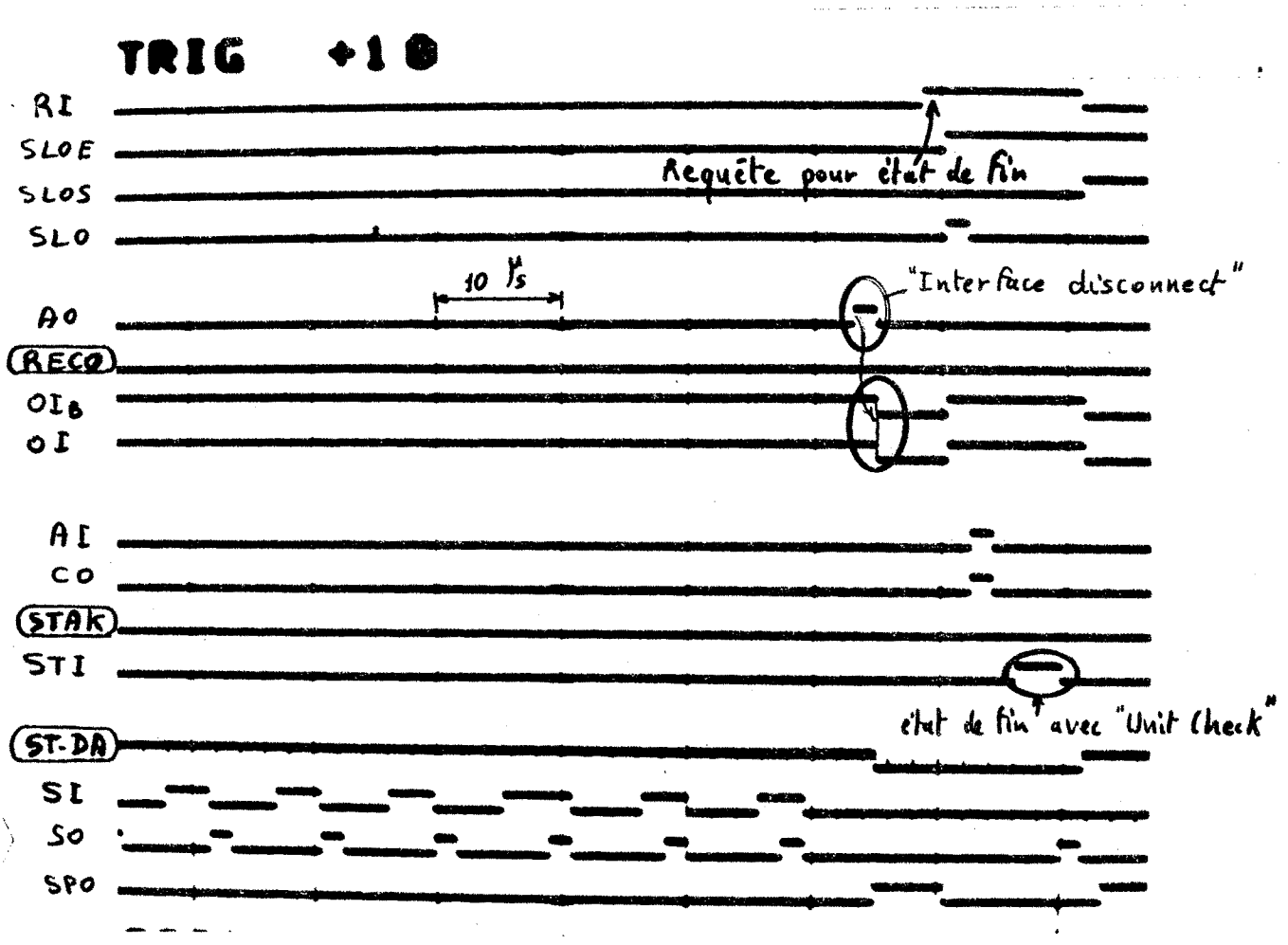


Fig.IV-21 HALT I/O pendant un transfert en mode "burst"

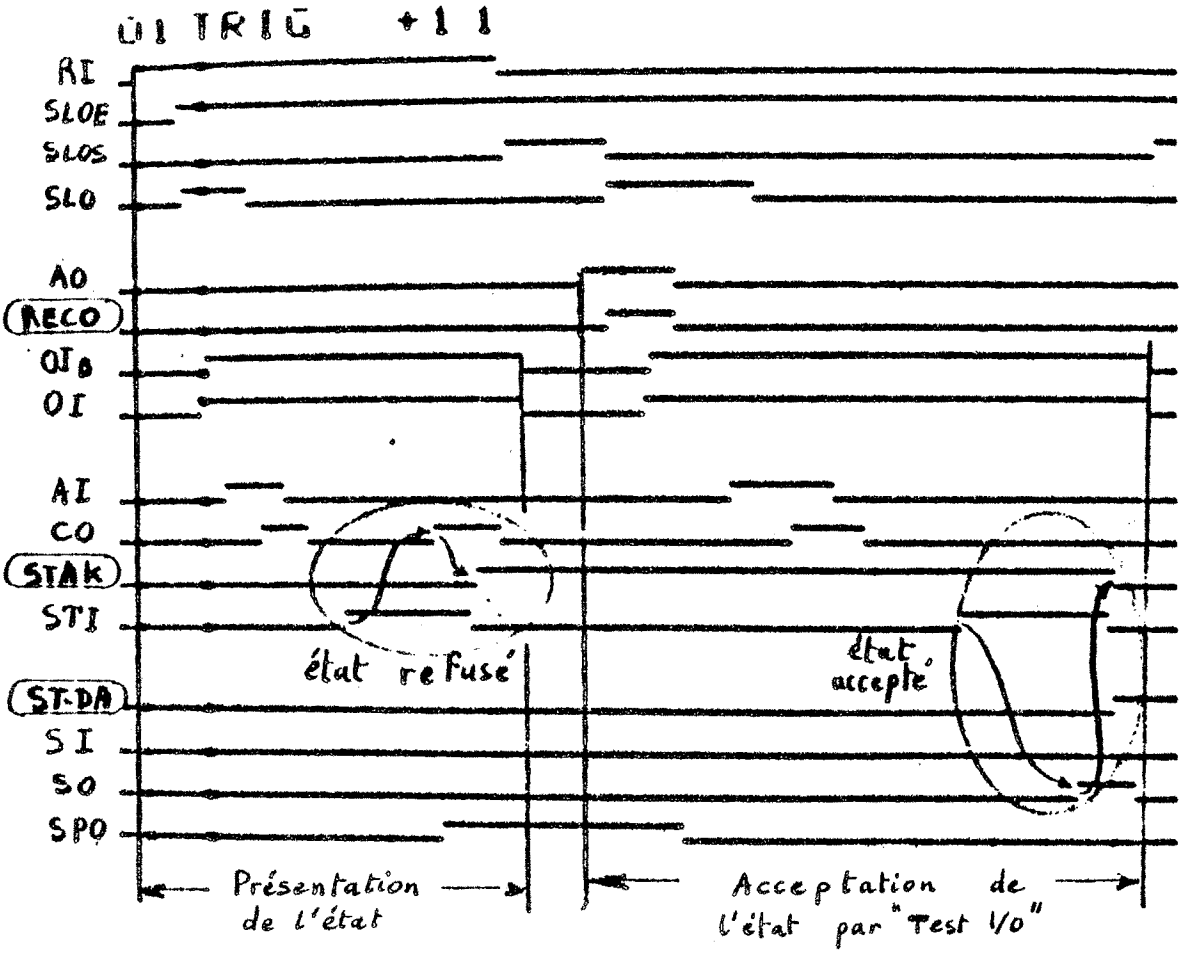


Fig IV-22 Séquence GENATTEN de présentation de l'état-unité "attention" suite à la découverte d'un tampon plein

CHAPITRE V

UN LOGICIEL DE TEST POUR L'UCB

V-1 INTRODUCTION

V-2 PROGRAMMATION ELEMENTAIRE

V-2-1 Rappels sur la programmation des E/S sur IBM 360

V-2-2 Le programme de contrôle de l'UCB

V-2-2-1 Généralités

V-2-2-2 Structure du programme

V-2-2-3 Exemples de dialogues

V-3 PROGRAMME MONITEUR EN LANGAGE EVOLUE

V-3-1 Intérêt

V-3-2 Conventions de passage Fortran-Assembleur 360

V-3-3 Les Primitives utilisées

V-3-3-1 Définitions

V-3-3-2 Structure des Primitives

V-3-4 Le programme moniteur

V-3-4-1 Structure

V-3-4-2 Exemples de dialogues

V-3-5 Utilisation des fonctions du système d'exploitation

V-3-5-1 Intérêt

V-3-5-2 Gestion des interruptions d'E/S dans CP/CMS

V-3-5-3 Modification des primitives et du moniteur

V-3-5-4 Difficultés rencontrées

V-1 INTRODUCTION

Afin de tester le comportement de l'UCB, nous avons été amenés à mettre au point, sur le 360, un ensemble de programmes conversationnels permettant l'envoi des commandes et la récupération des états-unité correspondant à leur exécution.

Ceci s'est fait dans un premier temps à l'aide d'un programme écrit en langage assembleur 360 par Y. Angélidès dans le cadre d'un projet de DEA <ANG>. L'avantage d'un tel programme est évidemment son faible coût en occupation mémoire et en temps d'exécution, mais une éventuelle extension nécessaire au test complet d'une application telle que PIAR, par exemple, demande alors de bonnes connaissances en langage 360, ce qui n'était pas notre cas...

Nous avons donc modifié la structure de ce logiciel en le décomposant en primitives correspondant chacune au lancement d'une commande particulière et écrite encore en langage 360 ; ces primitives sont appelées par un programme moniteur écrit en langage évolué (Fortran en l'occurence), facilement modifiable par l'utilisateur pour n'importe quelle application.

D'autre part, ces logiciels ont été volontairement prévus pour pouvoir s'exécuter sur un 360 nu et comprennent donc tous les mécanismes assurant le bon déroulement des E/S, même lorsqu'ils ont été utilisés sous un système d'exploitation (CP/CMS au CICG). Dans un deuxième temps nous avons essayé d'utiliser certaines fonctions du système d'exploitation, en particulier pour la récupération des interruptions.

V-2 PROGRAMMATION ELEMENTAIRE

V-2-1 Rappels sur la programmation des E/S sur 360 <IBM1>

On peut décomposer la programmation d'une entrée-sortie en 3 phases. Nous prendrons l'exemple de l'instruction Start I/O :

- la préparation
  - écriture du ou des CCW composant le programme canal
  - chargement du CAW avec l'adresse du premier CCW
  - écriture du "nouveau PSW d'E/S" (NEWPSWIO) qui contiendra l'adresse du sous-programme de traitement de l'interruption le cas échéant
  - masquage des interruptions d'E/S pendant l'exécution du SIO.

- le lancement
  - écriture de l'instruction SIO contenant l'adresse de l'unité sur laquelle doit avoir lieu l'E/S :

Adresse = (Registre de Base) + Déplacement (sur 12 bits)

  - test du Code Condition résultant. Ici plusieurs éventualités sont à envisager (voir fig.V-1)
  - si l'opération a été correctement lancée on peut avoir besoin de démasquer le CPU vis-à-vis du canal, dans ce cas on peut alors continuer l'exécution du programme ou se mettre en attente d'interruption. On peut aussi tester la fin de l'opération par une boucle sur une instruction Test I/O jusqu'à ce que le code condition soit nul.

- la fin de l'E/S
  - qu'elle soit détectée par interruption ou par TIO, il faut s'assurer de son déroulement correct (CSW)
  - pendant l'analyse du CSW, et tout autre traitement, les interruptions d'E/S doivent être masquées.

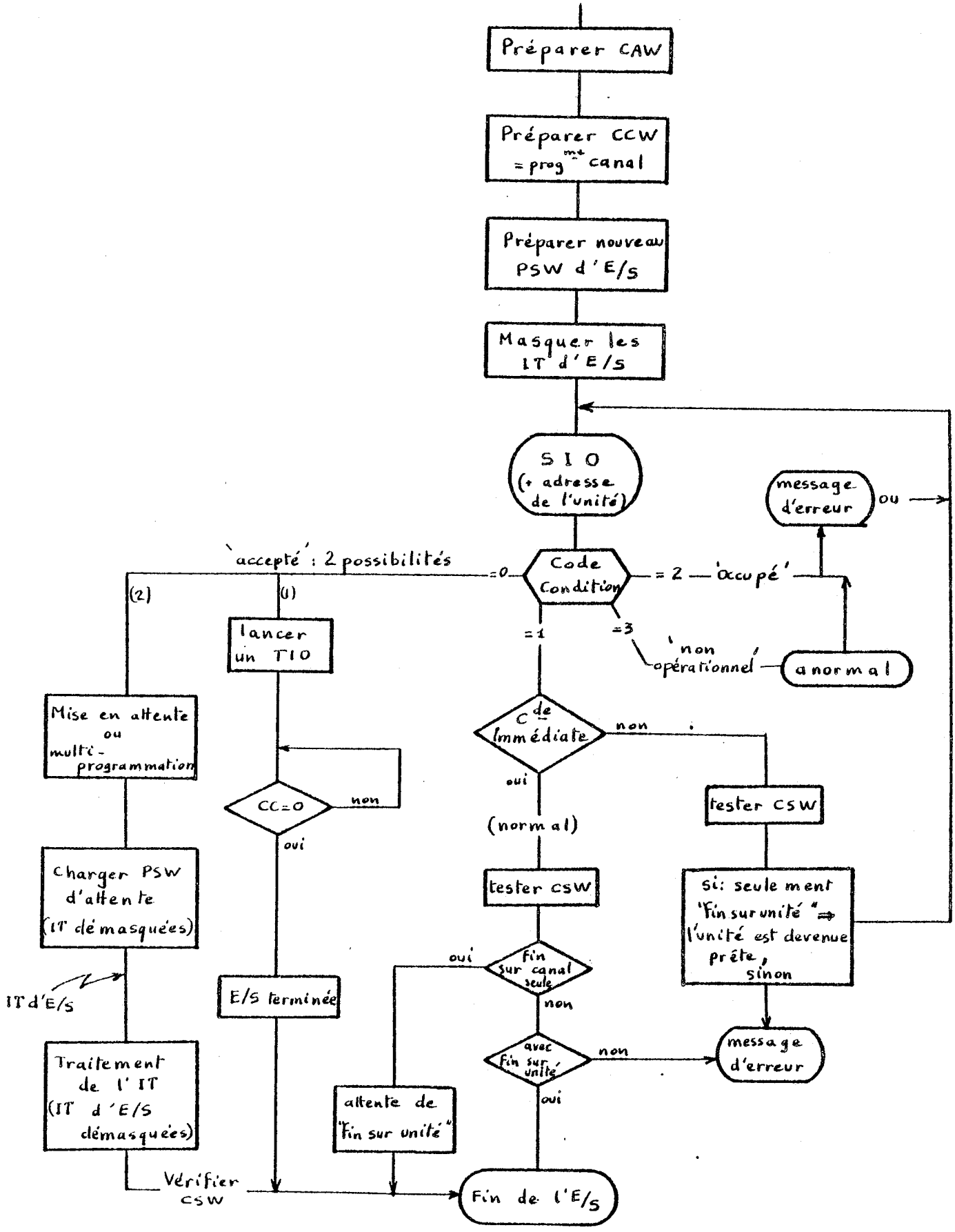


Fig.V-1 Organigramme de la programmation d'un START I/O

Remarques :

-1- on évite le plus souvent d'utiliser la programmation par la boucle SIO-TIO qui sont des instructions longues, et surtout sous CP-67 qui doit les simuler puisqu'elles sont privilégiées.

-2- en général la programmation des E/S est du ressort d'un programme superviseur ou même du système d'exploitation. Cependant si cela est nécessaire (en particulier dans notre cas) la gestion des anciens et nouveaux PSW d'E/S impose de respecter certaines règles :

- le masquage des interruptions n'est possible que vis-à-vis des interruptions d'E/S et Externes, de quelques interruptions de programmes et des erreurs machine, par le positionnement du bit correspondant dans le PSW.
- la prise en compte des interruptions suit une règle de priorité parmi les 5 classes : erreur machine, programme et appel superviseur (SVC), externe, E/S, dans l'ordre décroissant.

  Si 2 interruptions sont simultanées (eu égard au mécanisme d'échange des PSW), et de classes différentes, les traitements correspondants ont lieu dans l'ordre inverse des priorités (l'unité centrale ne possèdant qu'un seul emplacement pour le PSW) ; sauf pour les interruptions programme et SVC s'il y a eu une erreur machine, auquel cas elles sont perdues.

  Si une interruption est ignorée par masquage, elle est également perdue, sauf s'il s'agit d'une interrruption d'E/S ou externe qui sont mises en attente.

. en ce qui concerne les interruptions d'E/S plus précisément, nous avons vu qu'elles seront masquées lors de l'exécution du SIO et pendant le traitement de l'une d'elles. Par contre, s'il y a eu attente de l'interruption, son traitement doit prévoir la suppression du bit "wait" de l'ancien PSW d'E/S.

-3- la partie "code interruption" de l'ancien PSW d'E/S contient l'adresse de l'unité, cause de l'interruption

-4- il reste à rappeler le mécanisme de la gestion des PSW par le matériel:

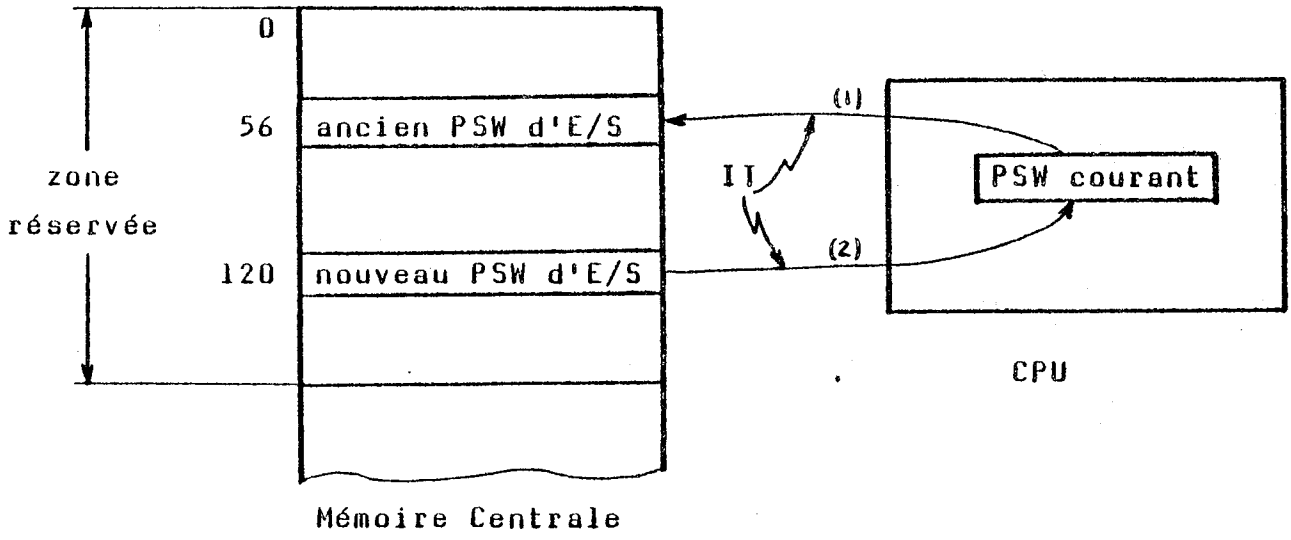


Fig.V-2 Le basculement des PSW lors d'une interruption d'E/S

V-2-2 Le programme de contrôle de l'UCB

V-2-2-1 Généralités

Il s'agit donc d'un programme qui permet de tester une unité d'E/S d'une façon conversationnelle à partir de la console opérateur d'un 360.

Ce programme est auto-chargeable, afin d'être exécuté sur machine nue, par la procédure IPL à partir d'un lecteur de cartes ou d'un dérouleur de bande magnétique <DUM>.

Il peut être également exécuté sous CMS sur une machine virtuelle générée par CP-67 ; il est alors implanté sous forme d'un "module", et dans ce cas il contient, en plus, des sections de sauvegarde et de restauration des nouveaux PSW d'E/S et externe, et l'adresse de la console opérateur sera "09" au lieu de "1F".

Il permet :

- de préciser l'adresse de l'unité destinataire de la commande (cas d'une unité d'E/S simulant plusieurs adresses)

- de connaître le résultat de l'initialisation de l'instruction

- de lancer les différentes commandes d'E/S, y compris celle correspondant à l'instruction Test I/O

- de connaître les états-unité obtenus à la fin de l'opération d'E/S, ou s'il y a eu rangement du CSW

- d'envoyer une interruption externe (touches REQUEST de la console opérateur ou INTERRUPT du 360, ou commande "external" sous CMS) lorsqu'on est en attente d'événement, ou si l'interruption d'E/S attendue ne se produit pas...

V-2-2-2 Structure du programme

La fig.V-3 montre l'organisation du programme en faisant apparaître également les différentes liaisons avec l'opérateur et avec le superviseur le cas échéant.

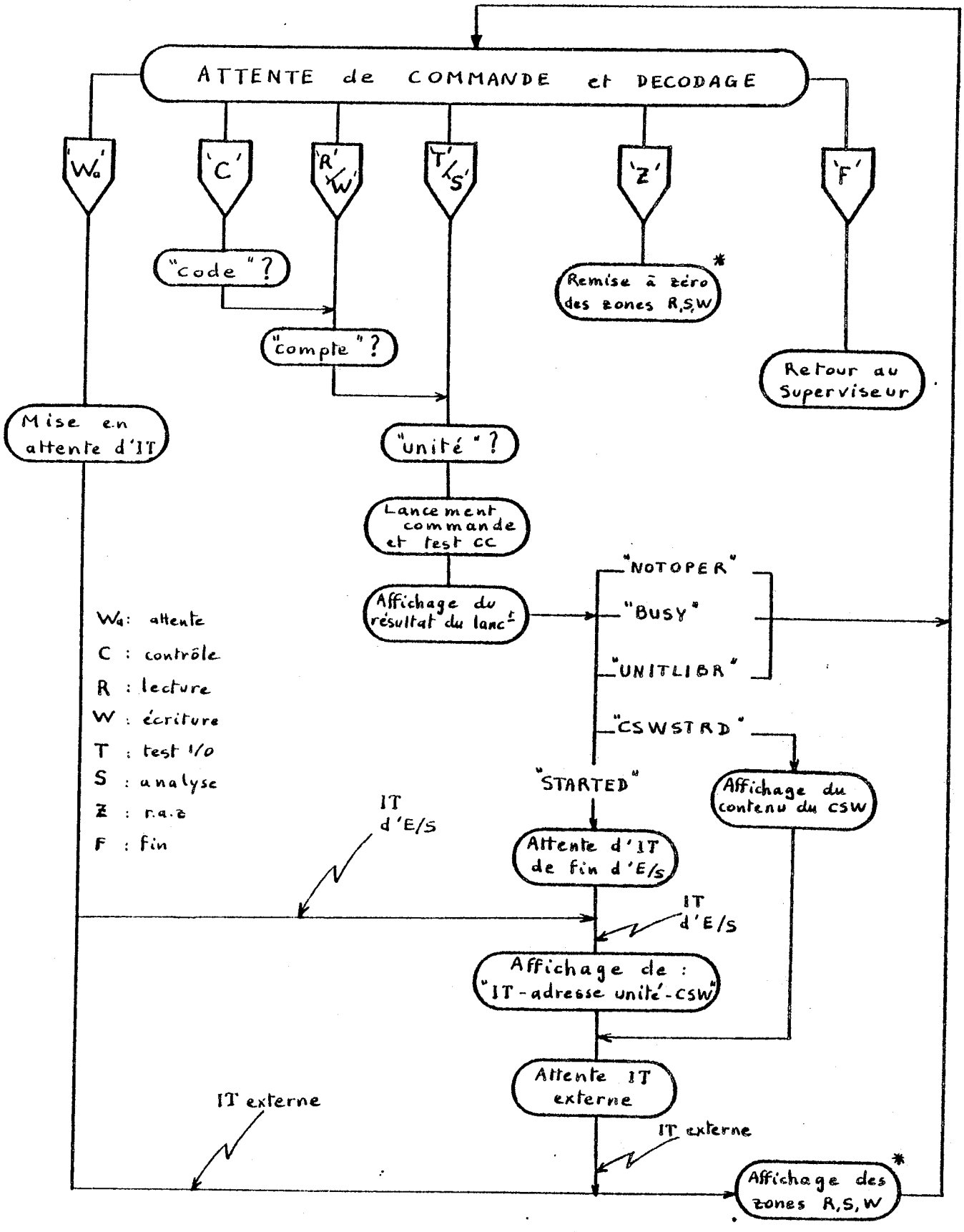


Fig.V-3 Structure du programme de contrôle de l'UCB

* (zone R = zone de lecture : information reçue (4 lers bits)
(zone W = zone d'écriture : - envoyée (- -)
(zone S = zone d'anayse : octet d'analyse

V-2-2-2 Exemples de dialogues

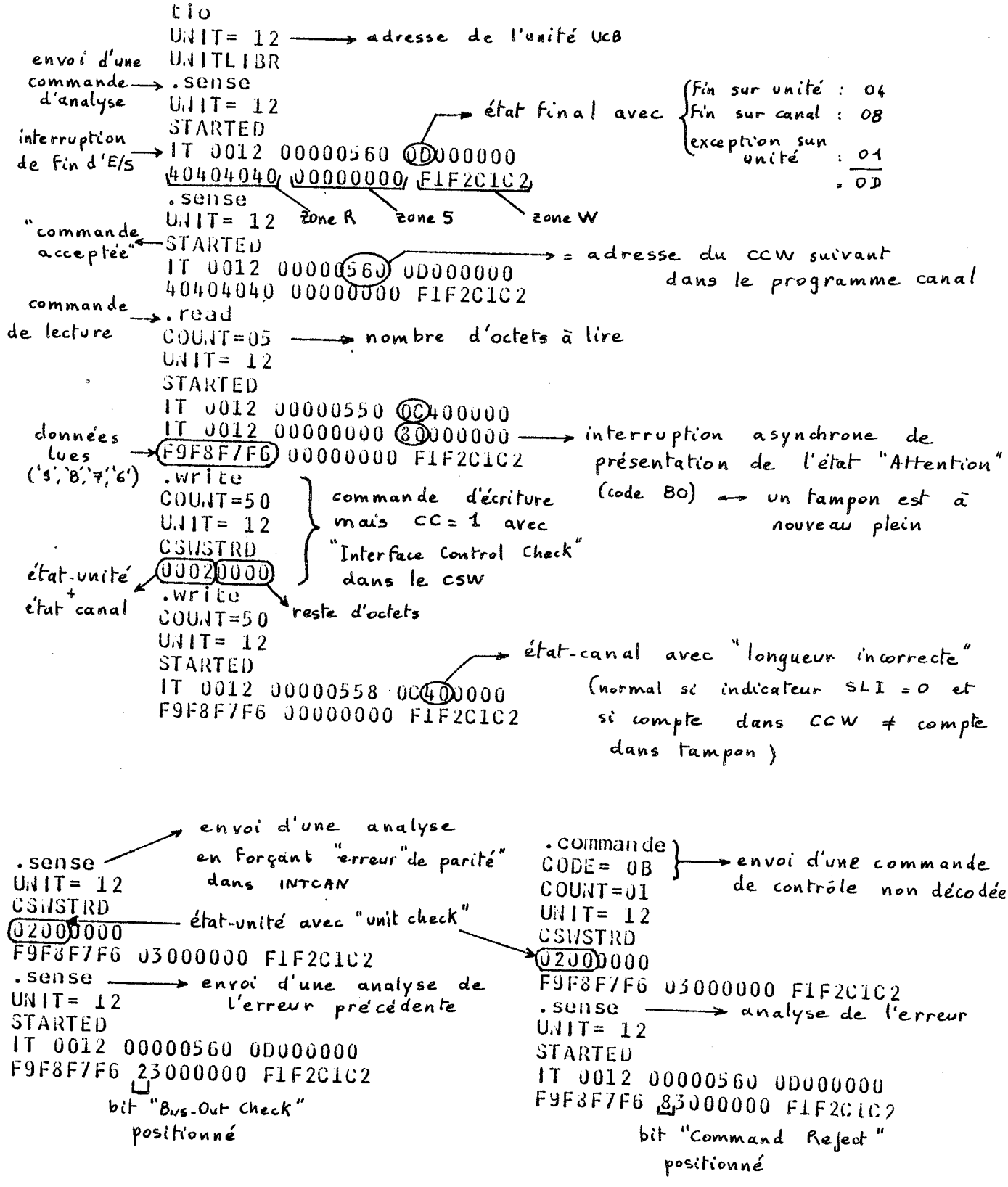


Fig.V-4 Quelques dialogues avec l'UCB

V-3 PROGRAMME MONITEUR EN LANGAGE EVOLUE

V-3-1 Intérêt

Le programme précédent, très suffisant pour tester le comportement de l'UCB vis-à-vis du canal (c'est-à-dire plutôt le module Interface Canal) ne permet pas un enchaînement de commandes destinées à une application particulière de l'UCB.

L'acquisition des connaissances en langage assembleur 360, nécessaires à l'écriture d'une méthode d'accès optimum, semblait inopportun dans le contexte du changement très prochain du matériel du CICG. De plus une telle méthode d'accès ne se justifie pas dans le cadre d'une phase de mise au point d'une application, et il était préférable de rester dans l'idée contenue dans le nom même de l'UCB.

C'est donc dans un triple souci de :

- réalisme
- simplicité
- souplesse

que nous avons écrit des programmes, ramenés à leur plus simple expression en langage 360, et organisés autour d'un moniteur d'enchaînememt écrit en Fortran. Dans une première configuration orientée vers le test de l'UCB, nous garderons la structure du programme précédent. Cependant il devient maintenant très facile de tester le comportement de l'application elle-même.

Il reste évident que ce genre de programme est beaucoup moins performant au niveau de l'occupation mémoire et du temps CPU. De plus, il impose de respecter les protocoles de liaison entre programmes Fortran et sous-programmes Assembleur 360, que nous allons maintenant décrire.

V-3-2 Conventions de passage Fortran-Assembleur

Ce sont typiquement celles de la méthode de communication entre les programmes au niveau de l'exécution, dans le système IBM 360 ; en particulier entre le superviseur et les utilisateurs (voir annexe 1).

Nous rappellerons ici brièvement les moyens mis en jeu pour effectuer la liaison et ceux permettant le transfert des paramètres.

- Les outils de liaison : ce sont 5 registres et une zone de sauvegarde d'adresses et de registres.

- Le passage des paramètres : il se fait, à l'aide d'un ou 2 registres pointant sur une zone de travail, par l'intermédiaire, par exemple, de : CALL SUB (A,B,C) La zone de travail contiendra alors, dans le même ordre, les adresses des paramètres A, B, C.

- Application : dans le cas particulier où le programme appelant est un programme Fortran, le compilateur génère, lors de l'exécution d'une instruction d'appel à un sous-programme assembleur 360 :

  . une liste d'arguments, dont l'adresse est dans le registre R1, et contenant les adresses des paramètres (variables, tableaux, sous-programmes). Ces paramètres peuvent être aussi bien des données pour le sous-programme, que des résultats à renvoyer au programme principal

  . une zone de sauvegarde de 18 mots

  . une séquence d'appel pour transférer le contrôle au sous-programme.

A partir de là, et pour respecter l'ensemble des conventions, le sous-programme assembleur aura la structure suivante, dans le cas simple où ce programme n'appelle pas un autre sous-programme

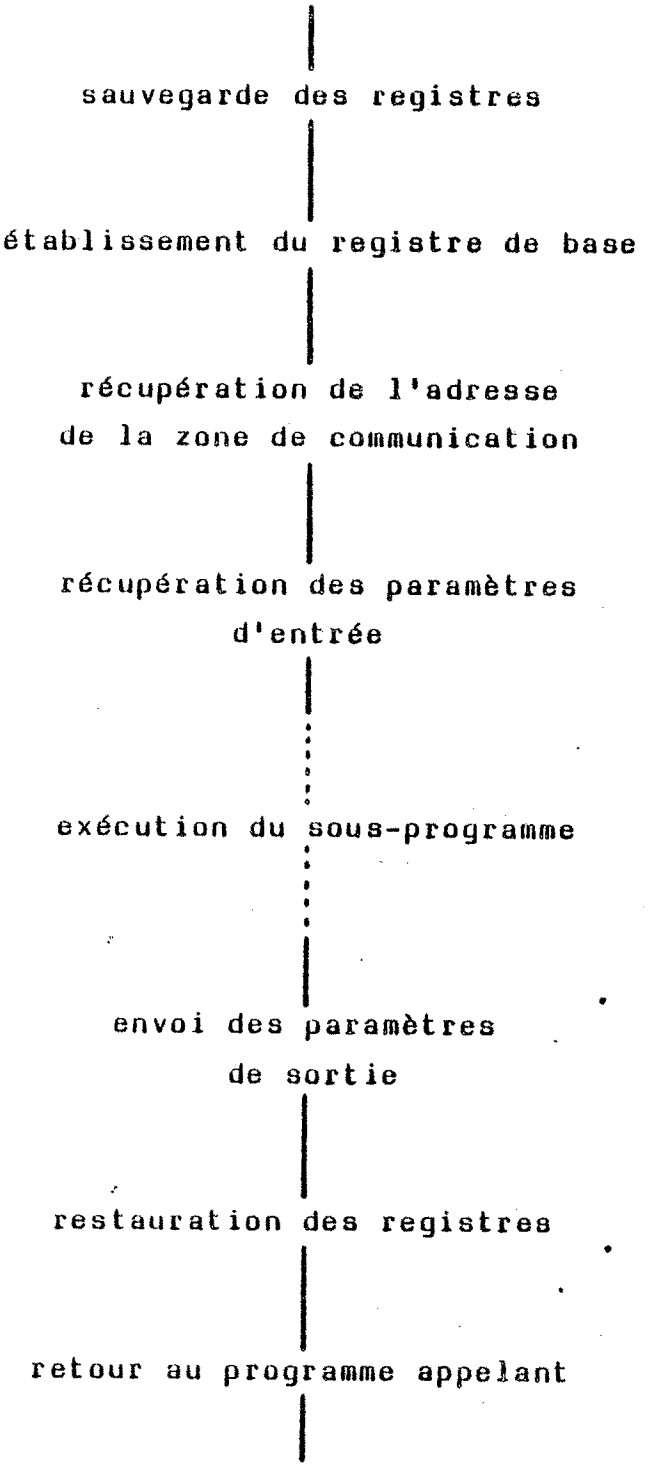


Fig.V-5 Structure d'un sous-programme assembleur 360 appelé par un programme Fortran

V-3-3 Les primitives utilisées

Actuellement nous disposons de 6 primitives correspondant aux 5 commandes de base et à un sous-programme d'attente. Cette attente est soit celle d'une interruption asynchrone de l'UCB, soit celle d'une interruption externe provoquée par l'opérateur qui veut envoyer une commande à l'UCB.

V-3-3-1 Définitions

Les primitives sont nommées comme un sous-programme Fortran et leur appel se fait par l'instruction classique :

```
CALL NOM (PARAM1, ...., PARAMn)
```

où les PARAMi sont aussi bien des paramètres d'entrée que de sortie vis-à-vis du sous-programme.

Nous avons :

```
ATTENT (ADUNIT, CODIT, STUNIT, STCHAN)

TIO (ADUNIT, CC, STUNIT, STCHAN)

SENSE (ADUNIT, OCTSEN, CC, STUNIT,STCHAN)

READ (ADUNIT, COMPTE, DONNEE, CC, RESTE, STUNIT, STCHAN)

WRITE ( -  ,  -  ,  -  , - ,  -  ,  -  ,  -  )

CNTROL (ADUNIT, CODOP, CC, STUNIT, STCHAN)
```

Avec les paramètres suivants :

- ADUNIT est l'adresse de l'unité à tester
- CC est le "code condition" résultat de l'exécution de l'instruction d'E/S
- STUNIT et STCHAN sont les états de l'unité et du canal extraits du CSW
- COMPTE et RESTE sont respectivement le nombre d'octets à échanger et restants
- DONNEE est l'adresse de la zone émettrice ou réceptrice des données à échanger
- OCTSEN est l'adresse de l'octet d'analyse envoyé par l'unité
- CODIT est le "code interruption" contenu dans l'ancien PSW qui identifie la source de l'interruption
- CODOP est le code de la commande de contrôle à envoyer.

Nous appellerons "primitives immédiates" les primitives TIO (correspondant à l'exécution d'une instruction Test I/O) et CNTROL, par opposition aux "primitives normales" qui provoquent un échange de données.

V-3-3-2 Structure des primitives

Les primitives que nous venons de décrire ont une structure commune sauf ATTENT. D'autre part, la structure des primitives immédiates est plus simple puisqu'il n'est pas nécessaire d'attendre une interruption.

La fig.V-6 donne l'organisation générale des primitives normales en tenant compte des protocoles de communication avec le programme Fortran.

La fig.V-7 donne l'organisation de la primitive ATTENT.

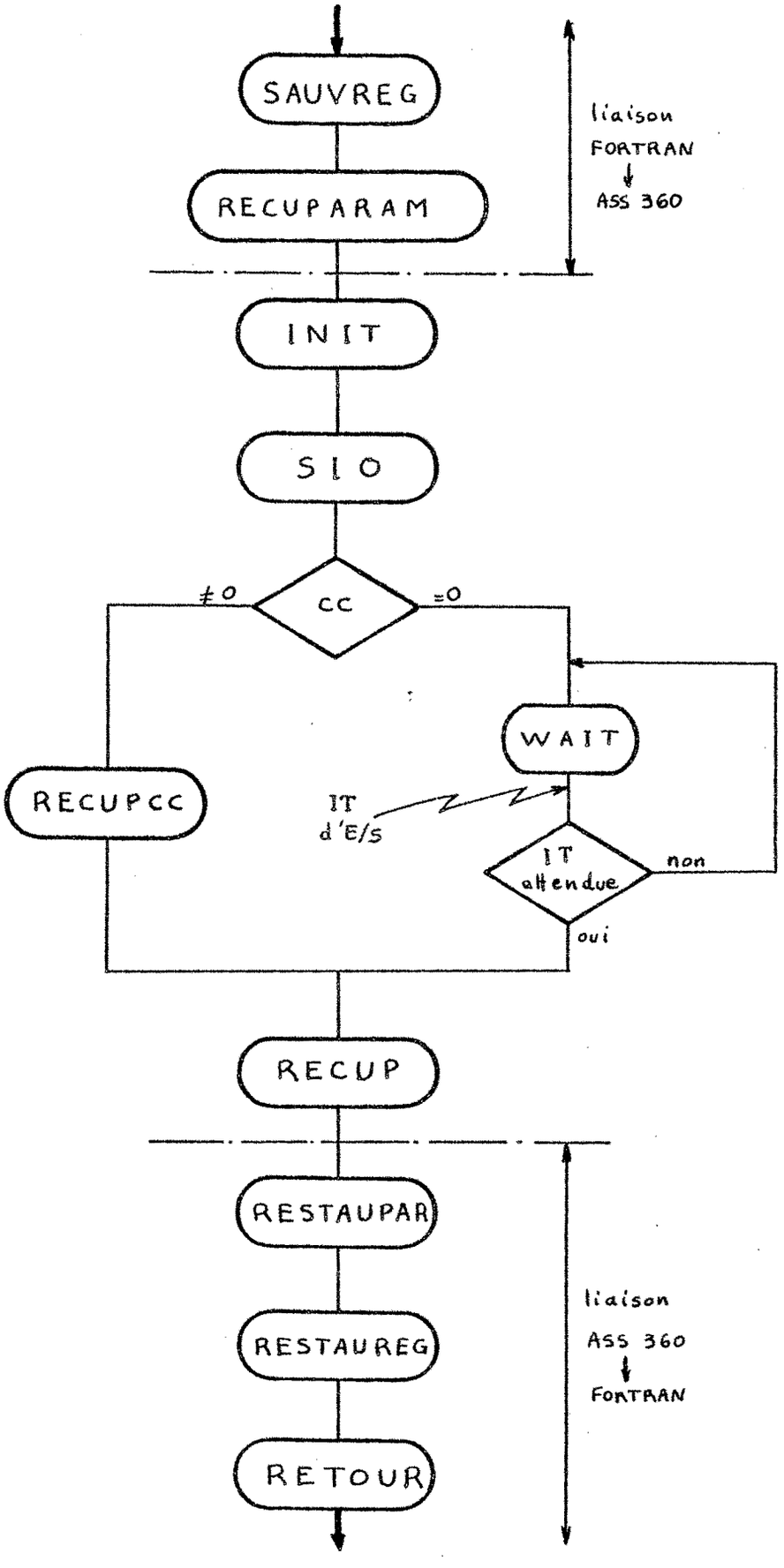


Fig.V-6 Structure simplifiée d'une primitive normale

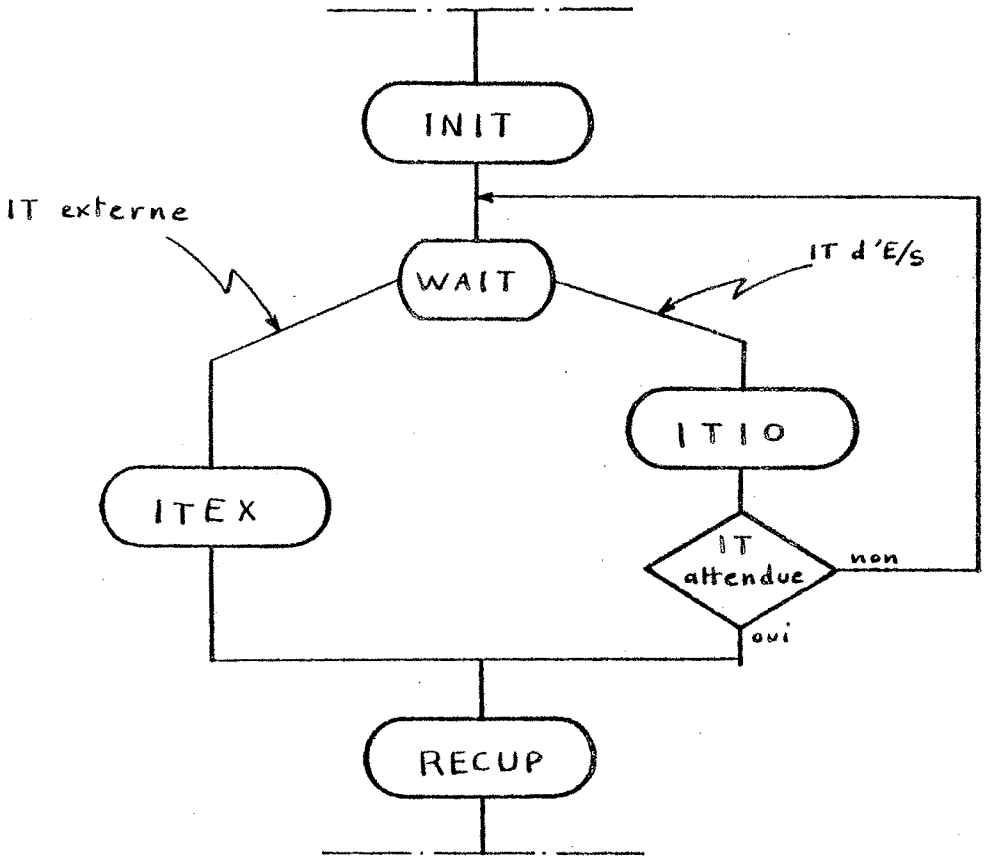


Fig.V-7 Structure de la primitive ATTENT

Remarques :

-1- Les procédures RECUP sont différentes selon les primitives. Elles sont chargées de préparer les paramètres de sortie pour le programme moniteur (états, code condition, données, reste...)

-2- La procédure INIT correspond à la préparation du Start I/O telle que nous l'avons envisagée au $ V-2-1

-3- Le test de l'interruption consiste d'une part à vérifier qu'elle vient bien de l'unité testée et d'autre part qu'elle correspond bien à la fin de l'E/S (ceci permet de tester des unités qui génèrent 2 conditions d'interruption ; c'est-à-dire : "fin sur canal" avant "fin sur unité"

-4- Si nous travaillons sous un système d'exploitation, la procédure INIT doit également sauvegarder les nouveaux PSW des interruptions attendues (d'E/S et externe pour ATTENT) et la procédure RETOUR doit les restaurer.

V-3-4 Le programme moniteur

V-3-4-1 Structure

Dans une première phase de mise au point de l'unité, ce programme conserve la structure du programme précédent dans la mesure où il ne fait que demander à l'opérateur les variables nécessaires à l'exécution des primitives (données, comptes, ...) et porter à sa connaissance les résultats obtenus.

L'utilisation d'un langage évolué permet cependant une écriture aisée des procédures conversationnelles, ce qui conduit à une présentation plus explicite des dialogues (comparer les différents exemples des fig.V-4 et V-9).

Sur la représentation simplifiée de l'organisation de l'ensemble du programme (fig.V-8), on remarque la division du moniteur en 4 tâches :

- DIAL qui exécute le dialogue avec l'opérateur
- APPEL qui lance l'exécution des différentes primitives après réception des ordres et des données par DIAL
- RESUL qui analyse les différents résultats obtenus et les transmet à DIAL
- MONITEUR qui gère les 3 tâches précédentes.

Dans une deuxième phase, l'organisation conversationnelle peut s'opérer à un niveau supérieur en définissant des ordres évolués pour l'exécution desquels il sera fait appel à un enchaînement des primitives précédentes, à un traitement éventuel des erreurs (p.ex. envoi automatique d'une commande d'analyse)...

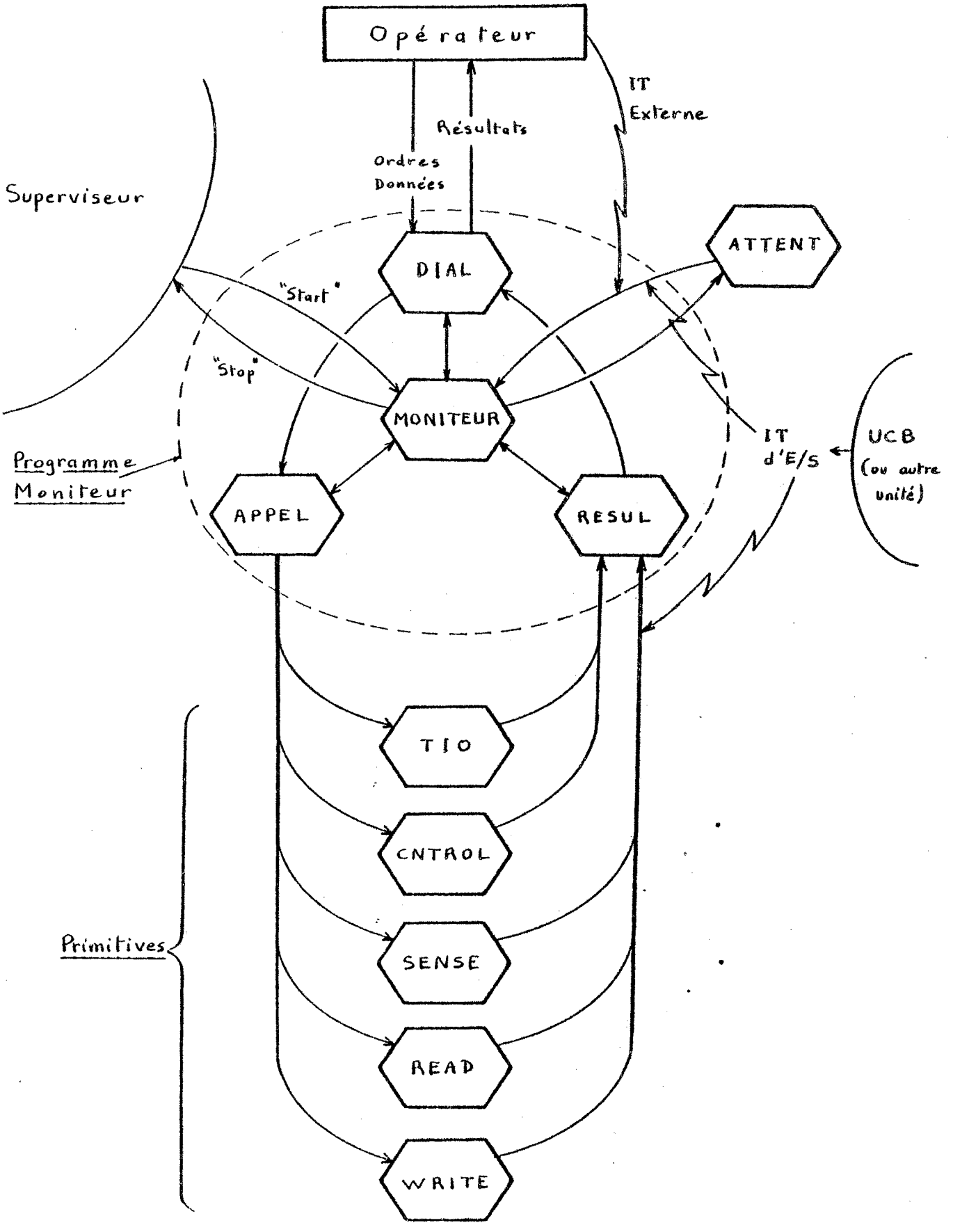


Fig.V-8 Organisation générale du programme moniteur

Pour cela il suffira de modifier la tâche MONITEUR du programme Fortran en introduisant des procédures de traitement des résultats des primitives utilisées, et de ne fournir à l'opérateur qu'un résultat global ou de l'informer de l'occurence d'une erreur fatale.

V-3-4-2 Exemples de dialogues

Voir fig.V-9.

V-3-5 Utilisation des fonctions du sytème d'exploitation

V-3-5-1 Intérêt

Jusqu'à présent nous avons essayé de nous affranchir de la configuration logicielle du site dans lequel serait implantée l'UCB, et nous avons donc assuré complètement la gestion des E/S.

Si on cherche à utiliser certaines fonctions ou facilités offertes par le système d'exploitation de l'ordinateur on peut logiquement s'attendre à pouvoir simplifier les programmes précédents.

On se heurte cependant à des problèmes dus à la relative complexité de certains systèmes, ce qui nécessite un apprentissage pouvant être disproportionné par rapport à la simplification escomptée. D'autre part on s'écarte de la préhension immédiate des mécanismes, ce qui était notre règle jusqu'à présent.

Malgré ces quelques inconvénients relatifs, il était intéressant d'aborder cet aspect de la gestion des E/S. Pour notre part nous n'avons utilisé qu'un appel à la fonction qui récupère les interruptions d'E/S pour l'utilisateur, ce qui revient à ne plus s'occuper des "nouveaux PSW d'E/S", ni à "trier" les interruptions.

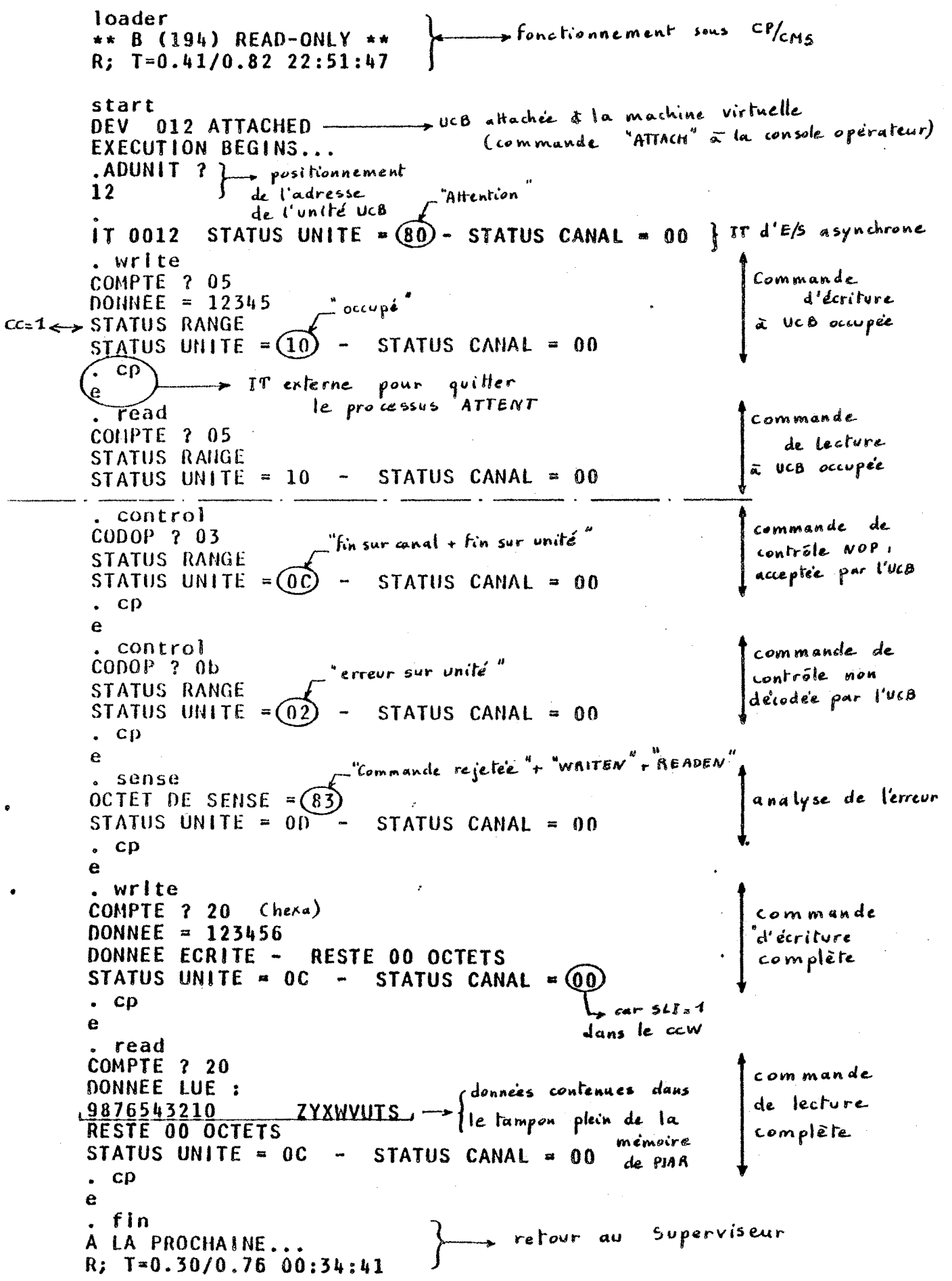


Fig.V-9 Quelques dialogues avec l'UCB (programme Fortran)

V-3-5-2 Gestion des interruptions d'E/S dans CP/CMS

Rappelons que CP-67 est un système qui, tournant sur un ordinateur IBM 360-67 (à cause de son mécanisme de traduction automatique des adresses : DAT), génère pour plusieurs utilisateurs, des ordinateurs 360 virtuels et leurs unités d'E/S. Chaque utilisateur dispose du système conversationnel CMS par l'intermédiaire de sa "console opérateur".

Il est hors de question de décrire ici le fonctionnement du système CP-67, ni même de voir comment se déroule une instruction Start I/O d'un programme utilisateur <FIN>, <BEL>.

Rappelons seulement que cette instruction comme toutes les instructions privilégiées est simulée par CP qui, après compilation du programme canal et vérification de la liberté d'accès à l'unité d'E/S concernée, exécute l'instruction réelle puis "réfléchit" l'interruption d'E/S finale à l'utilisateur en réactivant sa machine virtuelle.

Pour résumer on a globalement :

-1- Transformation du SIO virtuel en SIO réel
-2- Transformation de l'interruption réelle en interruption virtuelle.

Au niveau de la réception de l'interruption, se place le module IOINT qui :

- identifie l'unité en cause
- transmet toutes les informations nécessaires au sous-programme qui gère les interruptions d'E/S (ancien PSW d'E/S, CSW,...)
- redonne la main au "DISPACHER" de CP.

Si nous voulons simplifier notre gestion d'E/S, tout en conservant l'accès aux informations indispensables au programme moniteur, nous pouvons faire appel à la fonction HNDINT qui donne à IOINT l'adresse d'un programme de gestion des interruptions d'E/S provenant d'une unité d'E/S donnée, autre que celles normalement gérées par CMS.

IOINT active alors ce programme en lui fournissant les informations qu'il a recueilli par l'intermédiaire des registres R0 à R4 ainsi que l'adresse de retour dans R14.

Le retour se fait en utilisant aussi R15, dont le contenu est nul si tout s'est bien passé. A la suite de quoi, IOINT s'informe de la présence du bit "wait" dans l'ancien PSW et selon le cas le laisse ou le supprime avant de recharger ce PSW.

V-3-5-3 Modification des programmes

Au niveau du programme moniteur nous ajoutons les appels à 2 primitives.

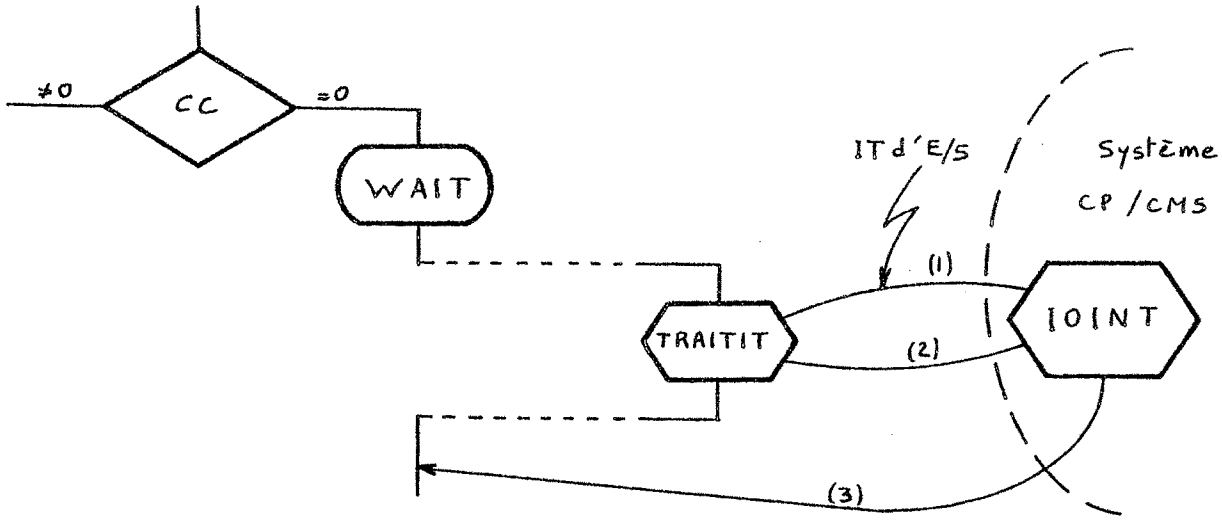
L'une d'elles (INITIO), appelée en début de programme initialise le transfert de la gestion des interruptions avec comme paramètres : le nom symbolique et l'adresse de l'unité, l'adresse symbolique du programme de traitement de l'interruption (TRAITI) et des indicateurs précisant à quel moment doit se faire ce transfert : soit dès l'arrivée de l'interruption, soit seulement quand le programme est en attente d'interruption.

Nous ferons également un appel (par FINIO) à la fonction HNDINT en fin de session pour libérer IOINT de ce transfert de gestion.

Au niveau de la structure des primitives, il devient donc inutile de faire le traitement des nouveaux PSW d'E/S ainsi que la reconnaissance de l'unité qui provoque l'interruption ; ces actions sont exécutées par le système. De même l'analyse du CSW pour connaître la cause de l'interruption sera fait pour toutes

les primitives concernées, par le sous-programme appelé par IOINT, qui pourra également extraire du CSW les "états-unité" et les restes d'octets à transférer, avant le retour à IOINT.

Les primitives ATTENT, SENSE, READ, WRITE, voient donc leur structure simplifiée selon le schéma suivant :



V-3-5-4 Difficultés rencontrées

a) D'une part, lorsque nous provoquons des erreurs pour tester le comportement de l'UCB, nous nous apercevons que le contrôle de l'E/S nous échappe et que le système d'exploitation CP/CMS prend des initiatives pour tenter de "récupérer" ces erreurs (voir photos III-18 et III-20 du $ III-5).

Ceci a lieu, en général d'abord sous la forme de l'envoi d'une commande d'analyse pour connaître la cause de l'erreur et, le cas échéant de commandes de contrôle liées à la "vision" de l'UCB qu'a le système : en l'occurence celle d'une unité de bandes magnétiques. On reçoit alors des commandes du genre : "rembobinage et déchargement" !...

Il est évidemment normal que le système effectue ce genre de travail, mais nous avons alors des difficultés à reprendre la suite du test ; et ceci d'autant plus que l'erreur est souvent

"répétitive" : par exemple si on force l'UCB à détecter une erreur de parité dans la commande, toutes celles qui seront envoyées par le système seront traitées de la même façon, et le système devra déclarer l'UCB "défaillante"...

b) D'autre part, l'envoi, par une "pseudo-unité de bandes", d'un état-unité asynchrone contenant le bit "Attention", est également mal perçu par le système, puisque ce bit n'est pas utilisé par une unité de contrôle de bandes !...

Une gestion directe des entrées-sorties, au niveau des PSW, est alors préférable pour obtenir cet état dans de bonnes conditions.

c) en conclusion, une connaissance plus approfondie de ce système d'exploitation permettrait de mieux définir certains aspects de la programmation de l'UCB sur 360.

CHAPITRE VI

AMELIORATIONS ET PERSPECTIVES

VI-1 INTRODUCTION

VI-2 AMELIORATIONS

VI-2-1 Par rapport au canal
- VI-2-1-1 Reconnaissance de l'adresse de l'unité
- VI-2-1-2 Vitesse de transfert
- VI-2-1-3 Commandes

VI-2-2 Par rapport au dispositif
- VI-2-2-1 Les lignes banalisées de l'Interface Dispositif
- VI-2-2-2 Les lignes de synchronisation
- VI-2-2-3 Les variables de communication

VI-2-3 Par rapport à la programmation 360

VI-2-4 Définition d'une version CTC de l'UCB
- VI-2-4-1 Introduction
- VI-2-4-2 Application à l'UCB
- VI-2-4-3 Extension à une unité multi-coupleur

VI-3 PERSPECTIVES D'UTILISATIONS DE L'UCB

VI-3-1 Généralités

VI-3-2 Utilisation de l'UCB comme unité de contrôle classique

VI-3-2-1 Raccordement d'un périphérique standard

VI-3-2-2 Simulation d'une unité de contrôle

VI-3-3 L'UCB comme élément d'une unité de contrôle intelligente

VI-4 MOYENS

VI-4-1 Interface d'accès à une mémoire partageable

VI-4-2 Interface avec un bus standard pour microprocesseur

VI-4-3 Interface standard avec un mini-ordinateur

VI-4-3-1 Généralités

VI-4-3-2 Définition d'un interface banalisé pour mini-ordinateur

VI-1 INTRODUCTION

Dans le chapitre IV nous avons montré que l'UCB était opérationnelle, dans le cadre d'une application orientée uniquement vers des échanges d'informations entre le canal et une mémoire externe.

Dans ce chapitre nous essaierons de définir quelques groupes d'applications envisageables à partir de ce coupleur, et les moyens nécessaires pour y parvenir.

Nous proposerons en premier lieu certaines améliorations qui pourraient aisément être apportées à la liaison avec le canal, ainsi que des précisions sur la définition de l'interface avec le dispositif.

VI-2 AMELIORATIONS

VI-2-1 Par rapport au canal

VI-2-1-1 Reconnaissance de l'adresse de l'unité

Nous pouvons déjà signaler que le circuit de reconnaissance automatique de l'adresse pourrait être simplifié dans le cas où l'adresse du dispositif d'E/S utilisant l'UCB serait unique et définitivement choisie.

En effet, un comparateur câblé suffit alors pour positionner l'indicateur RECO.

Par contre la réalisation d'une unité capable de gérer plusieurs sous-canaux du canal multiplex, impose de conserver une mémoire contenant les différentes adresses de ces sous-canaux. Cette mémoire, vive pendant la période de mise au point, serait remplacée alors par une mémoire morte afin d'éliminer le mécanisme (câblé et programmé) de validation des adresses des unités correspondant à ces sous-canaux.

VI-2-1-2 Vitesse de transfert

Dans le chapitre III ($ III-5-2-2) nous avons vu que les vitesses de transfert atteintes étaient de l'ordre de la vitesse maximum du canal multiplex en mode burst. Nous avons vu aussi qu'il était possible d'atteindre des débits de 400 K.octets/s.

Un canal sélecteur du type 2860 a un débit maximum de 1,3 M.octet/s. Nous pouvons envisager une connexion à un canal de ce type.

Des simplifications apparaissent alors dans l'algorithme de transfert des données : en particulier les tests du mode de fonctionnement et de l'indicateur REQ ($ 4-2-2-a) sont inutiles puisque ce type de canal ne peut fonctionner qu'en mode "burst".

Par contre, il faut prévoir certaines modifications aussi bien matérielles que logicielles afin de garantir un débit maximum. Parmi celles-ci, nous pouvons citer :

-1- utilisation d'une file d'attente

-2- détection de la séquence "interface disconnect" par un mécanisme câblé

-3- optimisation des instructions qui testent les lignes SO et CO indiquant respectivement l'acceptation et l'arrêt du transfert (voir $ III-4-2-2-b)

-4- éventuellement emploi de commandes internes supplémentaires.

Nous estimons que 4 instructions seraient alors suffisantes pour effectuer un transfert élémentaire d'1 octet, portant ainsi le débit maximum à 1 M.octet/s.

Remarques :

-1- la file d'attente devra être :
- soit d'une taille au moins égale à celle attribuée par convention à un bloc de données et remplie avant (ou vidée après) l'échange par le dispositif, au prix d'un coût supérieur en matériel.
- soit plus petite et accédée simultanément par l'UCB et le dispositif si celui-ci à un débit suffisant, ce qui paraît concevable puisqu'on le connecte au canal sélecteur.

Dans les 2 cas son temps d'accès devra être inférieur ou égal à 1 μs. Sa gestion se fera à la fois par le logiciel (+ extension) et les lignes de contrôle du 8X300 pour assurer le synchronisme du transfert avec le bus de données du canal.

On peut trouver actuellement des files d'attente (ou mémoires FIFO) de 32 mots de 8 bits en technologie MOS à 1MHz ou 16 mots de 4 bits en bipolaire à 10 MHz.

-2- la séquence de déconnexion forcée par le canal étant particulièrement rare, il est évidemment peu rationnel de la détecter par une instruction supplémentaire lors de chaque échange d'un octet. C'est, comme nous l'avons déjà signalé, l'absence de mécanisme d'interruption qui est à l'origine de cette solution. Lors d'un branchement de l'UCB sur un canal sélecteur , il devient nécessaire d'implanter un tel mécanisme.

Une solution générale consiste à forcer l'exécution d'une instruction de saut à un sous-programme tout en sauvegardant le compteur ordinal et éventuellement l'état du processus courant, c'est-à-dire bâtir un mécanisme externe d'interruption.

Dans notre cas précis ces dernières actions sont inutiles puisque la séquence à détecter conduit à un abandon du service en cours, et le branchement à une séquence de présentation d'un état-unité de fin de transfert avec éventuellement "erreur sur unité".

Le circuit de la fig.VI-1 schématise ce mécanisme dans ce cas simple.

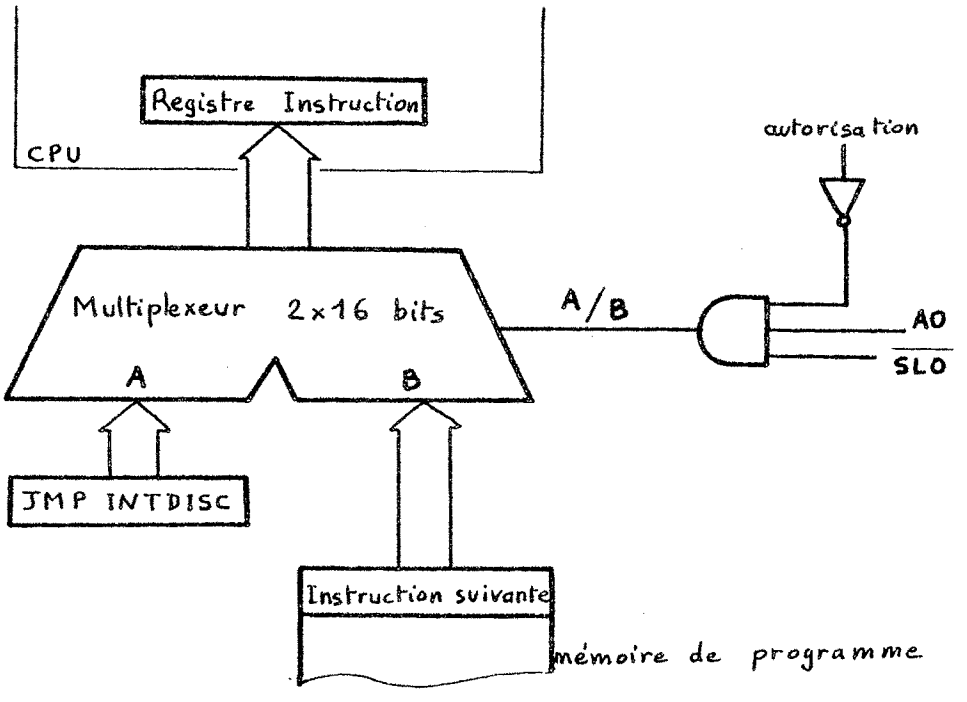


Fig.VI-1 Exemple d'un mécanisme externe d'interruption pour le 8X300

VI-2-1-3 Commandes

L'UCB ne décode actuellement que les commandes de base émises par le 360.

Seule une limitation de la taille de la mémoire programme fournie par le système MADAM (annexe 6) utilisé pendant la phase de mise au point de l'UCB, en est la cause. Nous pouvons en fait admettre une extension de mémoire programme jusqu'à 8192 instructions (512 actuellement). Ceci permettrait d'augmenter la taille du logiciel INTCAN (256 instructions actuellement) pour effectuer le décodage de toute commande "étendue" à l'aide des bits de modification M (voir $ II-3-1-2).

En particulier, des commandes de diagnostic, en lecture et écriture, seraient intéressantes pour tester globalement le comportement de la partie Interface Canal de l'UCB. Ceci consisterait à écrire puis relire une information arbitraire sans utiliser la partie Interface Dispositif (voir le Parallel Data Adapter (PDA) de l'unité 2701,<IBM7>).

VI-2-2 Par rapport au dispositif

Dans ce cas, il est plus difficile de faire des propositions précises d'améliorations de l'UCB puisque cette partie de l'UCB est essentiellement banalisée... par définition.

VI-2-2-1 Les lignes banalisées de l'Interface Dispositif

En premier lieu nous pouvons remarquer que leur nombre actuel de 24 peut facilement être augmenté (par multiples de 8 afin de conserver le bénéfice de l'utilisation des boîtiers d'E/S compatibles avec le 8X300).

D'autre part leur rôle est très lié au dispositif lui-même. Nous avons vu un exemple de définition de ces lignes dans le cadre du système PIAR. Nous verrons au paragraphe suivant quelques exemples du même type.

Cependant on peut imaginer de leur donner des fonctions plus précises dans le cadre d'une réalisation originale, ou de la connexion d'un organe d'E/S particulier.

Le choix de la direction des lignes bi-directionnelles offertes par l'Interface Dispositif est essentiellement souple.

Cette direction peut être :

- soit définitivement fixée (toujours en entrée ou toujours en sortie)

- soit établie par une des lignes d'un autre groupe (ex. : R/W commande la direction instantanée des 8 lignes de données du bus parallèle de PIAR)
- soit commandée par le dispositif selon les besoins.

Remarque : La direction choisie est néanmoins commune aux 8 lignes d'un boîtier. D'autre part, il est possible de placer ces lignes dans un état inactif "haute impédance" (plus de détails sur ces boîtiers sont donnés en annexe 2).

VI-2-2-2 Les lignes de synchronisation (R et G)

Nous en avons vu une utilisation au chapitre IV comme lignes de demande et d'autorisation liées à l'allocateur de bus qui gère l'accès à la mémoire commune du système PIAR. L'organisation logique de cette mémoire et la structure de l'allocation de bus en font un cas particulier bien que très intéressant pour une autre application de ce genre.

Nous pouvons également utiliser ces lignes comme lignes d'interruption (INT) et d'acquittement (INTA) dans le cas d'un dispositif comportant un autre ordinateur (mini ou micro), ou comme ligne de demande (DMARQ) et d'autorisation (DMACK) de DMA si un tel mécanisme est utilisable au sein du dispositif connecté.

La génération de ces lignes se ferait de la même façon que R et G (annexe 3 planche 9).

La fig.VI-2 donne un exemple de leur utilisation dans le cas où le dispositif est un microprocesseur standard type 6800.

VI-2-2-3 Les variables de communication

Si aucun dialogue n'est en cours avec le canal , le processus INTDISP est actif et gère le dispositif en préparant, le cas échéant, des indicateurs qui seront testés par le processus INTCAN comme nous l'avons vu au $ III-4-4-1.

Dans le cas particulier où le dispositif offre une mémoire partagée, les variables de communication pourront y être rangées.

Par exemple dans le système PIAR, INTMEM (qui joue le rôle de INTDISP) analyse les variables PROPI afin de positionner les indicateurs BUV et BUP.

Une autre solution est d'utiliser les boîtiers d'E/S de l'Interface Dispositif non plus seulement comme émetteurs/récepteurs de lignes, mais comme registres contenant les variables de communication entre l'UCB et le dispositif. En effet, ces circuits ont la particularité intéressante d'être inscriptibles et lisibles indépendamment par le 8X300 et par le milieu extérieur (voir annexe 2).

L'UCB permet donc d'établir, sans modification majeure, des communications avec le dispositif aussi bien à l'aide d'une mémoire commune que par des registres adressables.

Remarque : Dans tout ce qui précède, nous constatons que l'UCB est "maître" vis-à-vis du dispositif en décidant, par les processus INTDISP et INTCAN de l'instant où les variables de communication seront utilisées.

Si on veut que le dispositif active directement une tâche dans un de ces 2 processus, il faut doter l'UCB d'un mécanisme d'interruption plus élaboré que celui présenté au § VI-2-1-2. En effet, il faut alors pouvoir conserver l'instruction "volée" au CPU à l'extérieur de celui-ci, pour la restaurer à la fin du programme de service de l'interruption. Certains états et/ou variables de la tâche en cours seront éventuellement sauvegardés de façon analogue.

En résumé, le choix de l'utilisation du 8X300 fait qu'il vaut mieux éviter cette solution et laisser à l'UCB le soin d'interrompre le dispositif. L'utilisation de "microprocesseurs en tranches" aurait permis de disposer d'un mécanisme d'interruption, mais leur mise en oeuvre est plus difficile et ne se justifiait guère dans cette application.

VI-2-3 Par rapport à la programmation 360

On peut se demander si celle-ci ne pourrait pas se faire en utilisant une méthode d'accès connue. On constate alors que le comportement de l'UCB est assez semblable à celui d'une unité de bandes magnétiques.

En effet, exceptées certaines commandes de contrôle et de sélection de mode (nombre de pistes, densité, ...), la gestion des transferts de données dans ces 2 types d'unité est analogue. En particulier les commandes de lecture, écriture, analyse sont les commandes de base. Une commande de lecture arrière et certaines commandes de contrôle (rembobinage, effacement, sauts arrières de fichiers ou d'enregistrements...) peuvent être introduites et décodées sans difficulté.

Cependant, certains bits de l'octet d'état-unité ne sont pas utilisés dans l'UCB, comme le bit "attention", le bit "modificateur d'état" ou le bit "fin sur unité de contrôle", mais risquent d'être testés par la méthode d'accès dans certaines conditions.

De la même façon, une unité de bandes magnétiques envoie 5 octets d'analyse lors de la commande correspondante. Ceci peut provoquer l'état-canal "longueur incorrecte" si le programmeur ne positionne pas l'indicateur SLI dans le CCW.

Une connaissance plus précise de cette méthode d'accès serait nécessaire afin de prévoir l'emploi correct des bits de l'octet d'état-unité et de celui d'analyse, par rapport aux commandes émises et à certaines situations concrètes.

VI-2-4 Définition d'une version CIC de l'UCB

VI-2-4-1 Introduction

Nous appelons ici CIC une version programmée sur microprocesseur, du "Channel Io Channel Adapter" qui permet des échanges d'informations entre deux canaux connectés à 2 CPU distincts ou dépendants du même CPU ; dans ce dernier cas il permet des échanges de données entre 2 zones de la mémoire centrale.

Un CIC comprend essentiellement <IBM6) :

- 2 unités de contrôle attachées à chaque canal
- et une mémoire tampon d'un octet.

Une simplification vient du fait que le CIC est normalement uni-directionnel, c'est-à-dire qu'il initialise, pour un couple de sous-canaux donné, une communication dans un seul sens ; le transfert effectif n'a lieu que lorsque les 2 sous-canaux ont préalablement initialisé cette liaison logique à l'aide de 2 commandes compatibles.

Exemple : le sous-canal d'adresse 12 lance une commande de lecture ; le CIC doit en informer le sous-canal correspondant (envoi d' une interruption "Attention") ; celui-ci après analyse de la commande, doit lancer à son tour la commande compatible, en l'occurence une écriture.

Ce problème a été abordé d'une façon simplifiée lors de la première version d'un coupleur destiné à faire communiquer 2 machines virtuelles sous CP/CMS (voir annexe 4). Dans ce cas le canal est unique (canal multiplex) et le CIC simule les 2 unités de contrôle, ou plus précisément les 2 sous-canaux qui doivent échanger des informations.

Ceci suppose :

- qu'une convention définisse l'adresse du sous-canal "destinataire" de la première commande reçue par le CTC. Nous utilisons des adresses "complémentaires" (bits de poids faible opposés ; ex. 12 et 13)

- que le CTC reconnaisse un ou plusieurs jeu d'adresses complémentaires (problème résolu par le mécanisme de reconnaissance automatique d'adresse)

- qu'il gère l'enchaînement des différentes commandes et des états-unité correspondants : ceci était réalisé sous forme d'un automate à 6 états internes dans la première version du coupleur

- qu'il effectue l'échange proprement dit : il s'agit alors d'un mécanisme "ping-pong" entre les 2 sous-canaux, suivi de l'envoi des 2 états-unité de fin.

L'unité CTC ainsi réalisée se comporte donc vis-à-vis du canal comme une unité multiple. La réalisation qui en a été faite étant implantée sur un microprocesseur standard 6800, la gestion de l'échange , entièrement programmée, profitait de la souplesse et de la puissance offertes par les instructions et les modes d'adressage de ce microprocesseur. Elle est décrite plus en détail dans l'annexe 4 .

Remarque : une unité CTC n'a pas à traiter les erreurs transmises dans les informations et les algorithmes en sont d'autant simplifiés.

VI-2-4-2 Application à l'UCB

Le problème se pose différemment en raison essentiellement du traitement que peut effectuer le 8X300 . En effet, si sa rapidité et son jeu d'instruction sont bien adaptés à la gestion des

lignes de l'interface avec le canal, ces mêmes instructions le sont beaucoup moins pour la gestion d'un protocole de niveau supérieur. Cependant, en se limitant au cas simple exposé au début de ce paragraphe (un seul transfert à la fois pour un seul couple de sous-canaux) il est possible de transposer l'algorithme écrit pour le 6800 et une architecture plus classique.

Il est également possible de le modifier sensiblement pour l'adapter à la structure interne de l'UCB, en particulier profiter de l'extension de l'instruction (travail au niveau du "firmware").

Malheureusement, les conditions d'accès difficiles au 360/67 du CICG en raison de la charge de cet ordinateur, et la volonté de mettre au point le système PIAR avant son départ, n'ont pas permis de réaliser cette deuxième version du CIC.

VI-2-4-3 Extension à une unité multi-coupleur

Nous pouvons l'envisager sous 2 aspects, toujours dans le sens défini au $ VI-2-4-1, qui est de permettre des échanges entre 2 zones de la mémoire centrale :

- chaque sous-canal peut initialiser simultanément un transfert dans les 2 sens (CIC bi-directionnel)
- l'unité gère simultanément plusieurs couples de sous-canaux.

Pour atteindre l'un ou l'autre de ces objectifs, ou les deux, il semble préférable d'utiliser 2 microprocesseurs : le 8X300 qui exécute essentiellement le processus INICAN, et un microprocesseur standard qui exécute INTDISP (baptisé CIC pour la circonstance) pour lequel les contraintes de temps sont moins grandes.

Les principales fonctions que doit réaliser ce dernier processus sont :

- la mise à jour des tables contenant l'état instantané de chaque sous-canal (état-unité à envoyer, commande à transférer, etc...)
- le positionnement et le test des variables de communication avec le processus INTCAN.

Pour les raisons évoquées au $ VI-2-2-3, c'est l'Interface Canal qui active les processus CTC, et l'utilisation d'un Z80 et de son mécanisme d'interruptions vectorisées semble une solution à la fois simple et puissante : l'adresse du programme de gestion d'un sous-canal serait obtenu par transcodage à partir de l'adresse émise par le canal, à l'aide d'une mémoire morte ou d'un PLA.

Un dispositif de DMA, par lequel le 8X300 et INTCAN accéderaient directement aux tables d'état des sous-canaux est également possible, sous réserve de l'existence d'un mécanisme de synchronisation plus complexe.

Remarques :

-1- un problème semblable a été résolu par utilisation d'un Zilog Z80, d'un PLA et de files d'attente, dans un multi-contrôleur de 4 périphériques pour mini-ordinateur <BIN>.

-2- nous ne nous sommes à aucun moment placé au niveau de la programmation 360 de ces unités multiples, et a fortiori sous un système d'exploitation comme CP/CMS !...

VI-3 PERSPECTIVES D'UTILISATION DE L'UCB

VI-3-1 Généralités

D'un point de vue fonctionnel, il apparaît que l'UCB, à l'aide des modules Contrôle et Interface Canal et du logiciel INTCAN, réalise la fonction de couplage au système 360/370. Cette fonction, aussi importante soit elle, ne peut, seule, rendre l'UCB opérationnelle.

La fonction introduite par le dispositif à l'aide de l'Interface Dispositif et du logiciel INITDISP, définira le type d'application que supportera l'UCB.

Nous distinguerons alors :

- la fonction d'exécution qui fera de l'UCB une unité de contrôle au sens habituel IBM
- la fonction de dialogue qui permettra à l'ensemble UCB-Dispositif de se comporter en unité de contrôle "intelligente".

Dans le premier cas l'UCB permet de connecter un périphérique classique, donc de simuler une unité de contrôle existante.

Dans le deuxième cas, il s'agira de la connexion d'un périphérique lui-même "intelligent" (périphériques modernes) ou du raccordement du système IBM à un autre système informatique.

Nous pensons que les quelques améliorations proposées au début de ce chapitre, doivent permettre la réalisation, à partir de l'UCB, d'une large gamme d'applications dont nous donnons maintenant les principaux types.

Remarques :

-1- l'Interface Dispositif doit évidemment assurer lui aussi la fonction de couplage

-2- en conséquence, la fonction d'exécution n'exclue pas un certain dialogue, mais ramené au niveau élémentaire du transfert physique des informations.

VI-3-2 Utilisation de l'UCB comme unité de contrôle classique

VI-3-2-1 Raccordement d'un périphérique standard

La fonction d'exécution peut alors être directement implantée dans le logiciel INTDISP, la partie câblée de l'Interface Dispositif fournissant les lignes spécifiques de commamde du périphérique et de gestion du transfert.

L'UCB peut également fournir une ou deux mémoires tampons et effectuer certains traitements sur les données (conversion, formattage...).

VI-3-2-2 Simulation d'une unité de contrôle

Il est probable que nous devions alors utiliser, pour implanter la fonction d'exécution, soit un autre microprocesseur, soit un mini-ordinateur.

Si tel est le cas, l'Interface Dispositif exécutera principalement une fonction de couplage, le traitement principal se faisant dans ce nouvel élément.

La structure de communication se trouve être alors relativement simplifiée et réduite à un transfert de commandes, d'états et de données.

Dans le cas de l'utilisation d'un microprocesseur standard, l'UCB pourra être vue comme un de ses périphériques intelligents. Nous définirons alors certains registres adressables ($ VI-2-2-3) contenant les variables de communication, et le dialogue se fera par le bus parallèle du microprocesseur fourni par l'Interface Dispositif.

Les techniques d'interruption et de DMA peuvent également être envisagées (fig.VI-2).

Si un mini-ordinateur est nécessaire pour réaliser les fonctions de l'unité de contrôle, l'UCB devra probablement assurer la conversion du format des informations (assemblage / désassemblage des octets en mots de 16 bits) et de la même façon que précédemment, préparer les variables de communication qui sont présentées au micro-ordinateur selon son mode d'exécution des E/S : mode programmé simple ou mode canal microprogrammé sur le T1600, par registres adressables sur le PDP 11, etc...

La fig.VI-3 donne deux exemples d'utilisation de l'UCB couplée à un mini-ordinateur pour réaliser la simulation d'unités compatibles IBM 2701 et 2703.

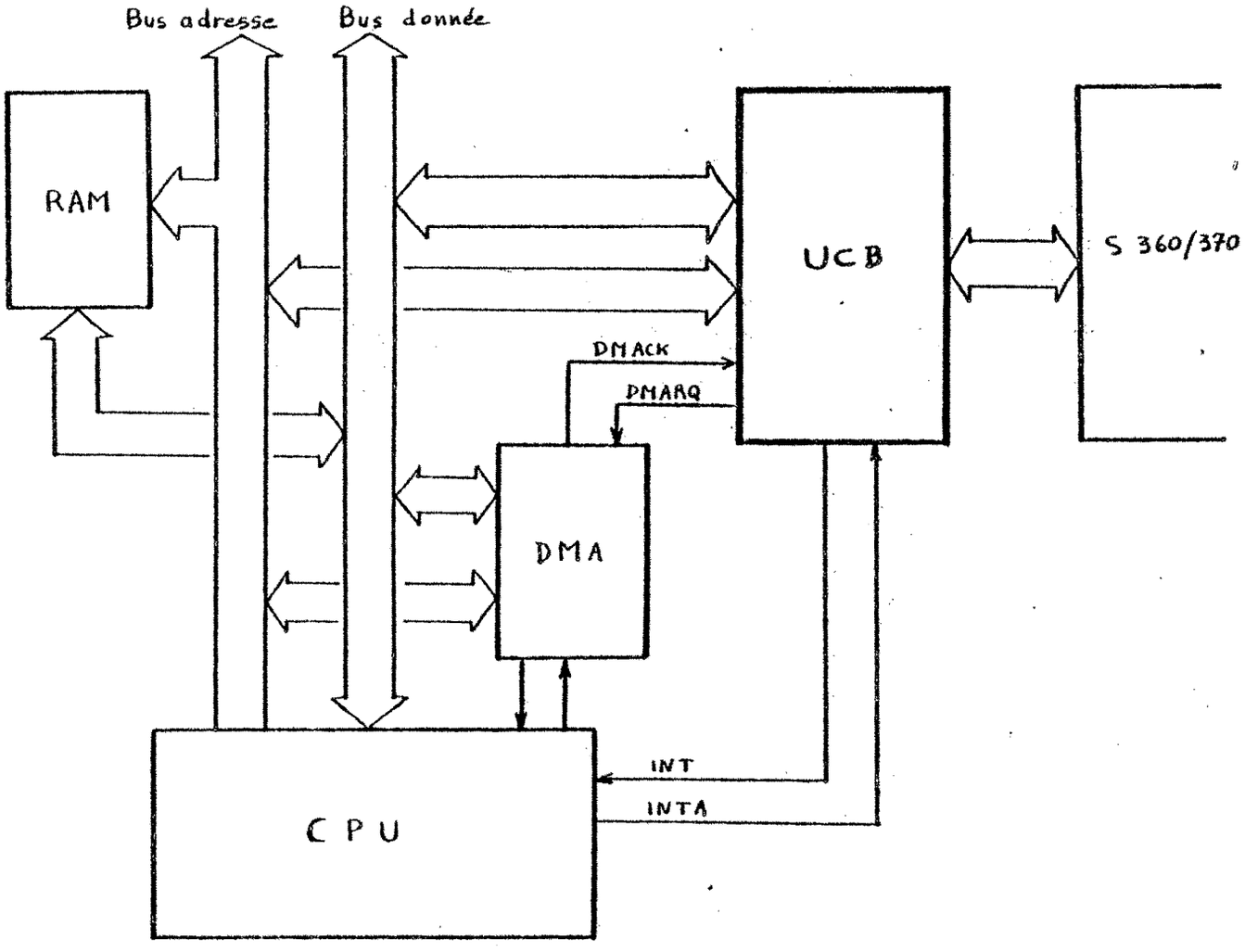
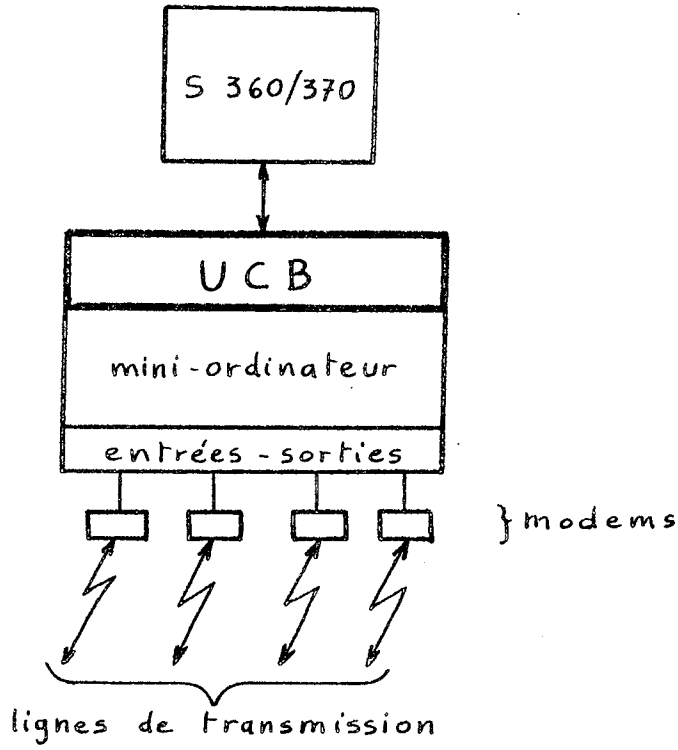
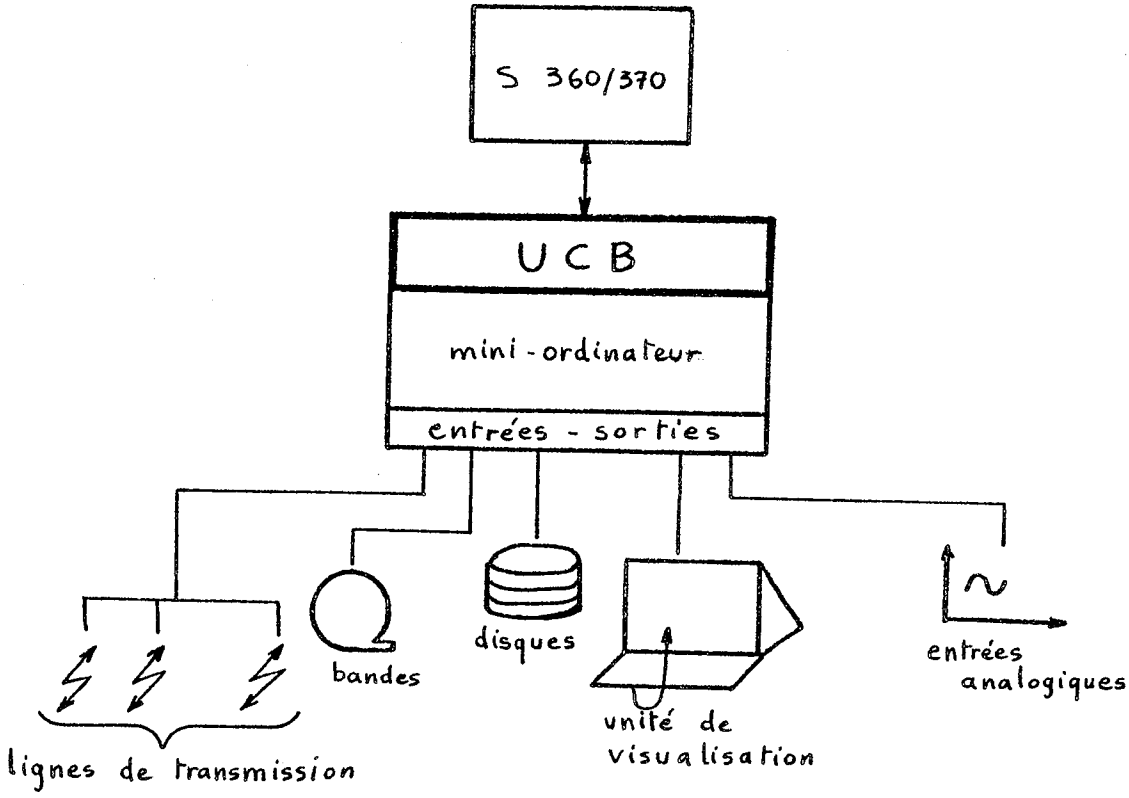


Fig.VI-2 Connexion de l'UCB à un microprocesseur standard



(a) simulation d'une unité IBM 2703



(b) simulation d'une unité IBM 2701

Fig.VI-3 Connexion de l'UCB à un mini-ordinateur pour simuler des unités de contrôle compatibles IBM

Remarques :

-1- l'utilisation d'un mini-ordinateur peut permettre d'autres applications que la simple simulation d'unités existantes. En particulier ses possibilités de traitement en temps réel, ajoutées à la souplesse et à la vitesse de transfert des données de l'UCB, ouvrent une gamme variée d'applications par exemple dans le domaine du contrôle de processus.

-2- le fait d'être amené à implanter la fonction d'exécution dans un microprocesseur ou un mini-ordinateur ne doit pas limiter le rôle de l'unité de contrôle à remplir des fonctions habituellement câblées. Il est intéressant au contraire de profiter de leur présence pour transférer à cette unité certaines fonctions habituellement confiées au logiciel d'exploitation (ou à celui de l'utilisateur) sur l'ordinateur du système 360/370.

C'est ainsi que l'on peut introduire la notion "d'intelligence" au niveau de la périphérie d'un système informatique.

VI-3-3 L'UCB comme élément d'une unité de contrôle "intelligente"

Le système PIAR fournit un bon exemple de ce que nous entendons par là, puisqu'il prenait à sa charge la plupart des fonctions d'une Station de Transport du réseau CYCLADES, préalablement confiées à un logiciel relativement important et contraignant dans l'ordinateur hôte.

Dans cette réalisation à base d'un système de microprocesseurs, l'UCB assurait essentiellement une fonction de dialogue avec les autres éléments du système (la fonction de couplage étant toujours exécutée au niveau de la partie Interface Canal).

A partir de cet exemple on peut imaginer la "faisabilité" d'autres applications analogues où le mécanisme d'intercommunication pourrait éventuellement être implanté autrement que par l'intermédiaire d'une mémoire commune. On retrouve alors les problèmes habituels soulevés par les sytèmes à plusieurs microprocesseurs.

De la même façon, une structure où l'intelligence serait implantée dans un ou plusieurs mini-ordinateurs est concevable à condition que les problèmes posés justifient leur emploi, eu égard au coût moindre d'une solution à base de microprocesseurs.

Par rapport au paragraphe précédent, la différence est que le rôle de l'UCB et du dispositif qui lui serait plus précisément connecté, serait celui d'assurer un dialogue avec les autres éléments de la structure. La fonction d'exécution étant soit distribuée, soit attribuée à un processeur spécialisé, autre que celui réalisé avec l'UCB.

Remarques :

-1- la connexion d'un périphérique "intelligent" simple (mini-imprimante, terminal à écran...) peut être inclue dans ce groupe d'applications, si on confie à l'UCB certaines fonctions supplémentaires afin d'accroître l'indépendance du dispositif par rapport à l'ordinateur central.

-2- nous voyons apparaître 2 connexions privilégiées parmi celles que devrait pouvoir offrir l'Interface Dispositif de l'UCB : connexion à un microprocesseur, connexion à un mini-ordinateur. Il nous a semblé intéressant, afin de respecter notre souci de définir une Unité de Contrôle Banalisée, de proposer quelques structures standards que pourrait prendre cet interface.

VI-4 MOYENS

VI-4-1 Interface d'accès à une mémoire partageable

Nous rappellerons ici la structure utilisée dans le système PIAR, car elle peut être une solution simple, commune à différentes applications dans lesquelles interviendraient un ou plusieurs microprocesseurs.

Nous fournirons, par exemple :

- 16 lignes d'adresse permettant l'accès à une mémoire de 64 K.mots. Ces lignes seraient soit positionnées en sortie, soit inactivées (lignes "3-états") par un processeur maître ou un allocateur de bus (cf. PIAR)

- 8 ou 16 lignes de données bi-directionnelles, placées sous les contrôles : du mécanisme d'allocation du bus mémoire et de celui qui impose le sens de transfert.

- 1 ligne de lecture/écriture

- 2 lignes de requête/acquittement permettant la synchronisation de l'accès à la mémoire.

VI-4-2 Interface avec un bus standard pour microprocesseur

Si le système 360/370 doit être connecté à un système de plusieurs microprocesseurs, il est intéressant que l'UCB propose un raccordement standard à ce système.

Ce pourra être soit un bus spécialisé reprenant les caractéristiques du bus du microprocesseur utilisé, soit un bus "universel".

Dans ce dernier cas, l'Interface Dispositif fournira les lignes nécessaires, tandis que le logiciel INIDISP exécutera les procédures de reconnaissance, d'initialisation et de gestion des transferts de données.

Parmi les standards de bus parallèles (nous nous limiterons à ceux-ci) adaptés aux applications avec microprocesseurs, on peut citer <ZAK"> :

- le bus S100 qui fournit 8 lignes d'entrée et 8 lignes de sortie, 16 lignes d'adresse, 8 lignes d'interruption, 39 lignes de contrôle et 3 lignes d'alimentation ; les 18 lignes restantes sont réservées pour des utilisations futures. Son inconvénient est le grand nombre de lignes de contrôle souvent inutiles. IL est plutôt employé dans le monde des applications individuelles.

- le bus IEEE 488 (ou HPIB, ou GPIB) plus orienté vers les systèmes intelligents d'acquisition de données.

Il possède :

- 8 lignes de données bi-directionnelles transportant des commandes, des adresses, des données
- 3 lignes de contrôle des transferts
- 5 lignes de contrôle général : type de l'information sur les lignes de données, initialisation, demande de service, fin de transfert, activation lointaine.

La fig.VI-4 représente schématiquement ce bus et la façon d'y connecter plusieurs modules. Son fonctionnement, simplifié, est le suivant :

Un module contrôleur initialise un transfert en envoyant sur le bus de donnée une adresse, puis une commande, successivement à 2 modules. L'un sera le "parleur" et l'autre "l'écouteur". Le transfert se fera ensuite du premier vers le deuxième, octet par octet, par l'intermédiaire du bus de données et selon une technique de "poignée de main" en utilisant les 3 lignes de contrôle des transferts.

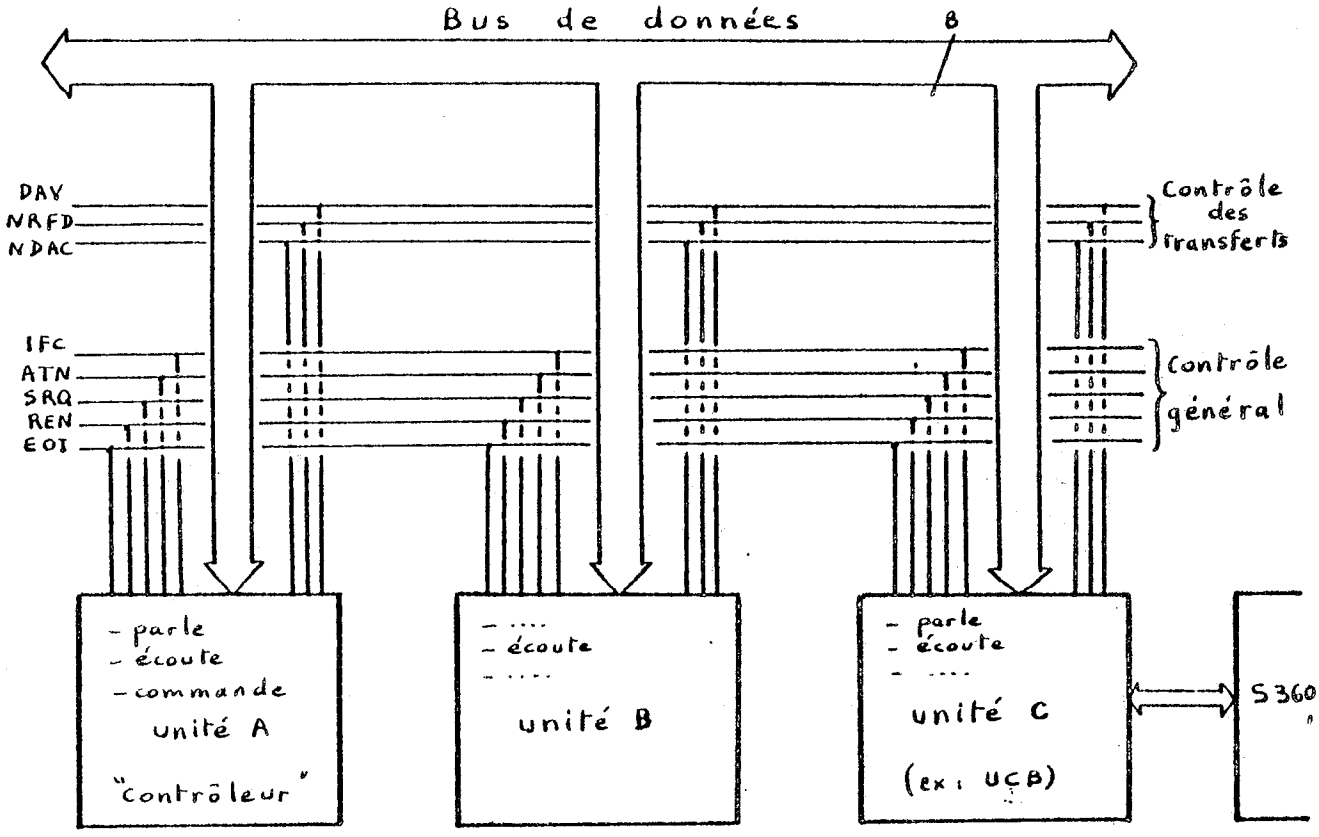


Fig.VI-4 Exemple de raccordement de l'UCB au bus IEEE 488

VI-4-3 Interface standard avec un mini-ordinateur

VI-4-3-1 Généralités

La configuration générale de la connexion du système 360/370 à un mini-ordinateur pourrait être celle de la fig.VI-5.

Ce serait par exemple le cas d'un T1600, d'un Mitra 15, d'un Solar 16, qui ont un bus mémoire et un bus d'E/S différents (sauf le modèle 16-05).

La série des ordinateurs PDP-11 se connecterait comme un microprocesseur standard puisqu'elle utilise un espace d'adressage banalisé pour la mémoire et les organes d'E/S.

La structure des 2 groupes de liaisons : celle avec le bus d'E/S et celle avec le système d'interruptions, est donc nécessairement dépendante du mini-ordinateur connecté.

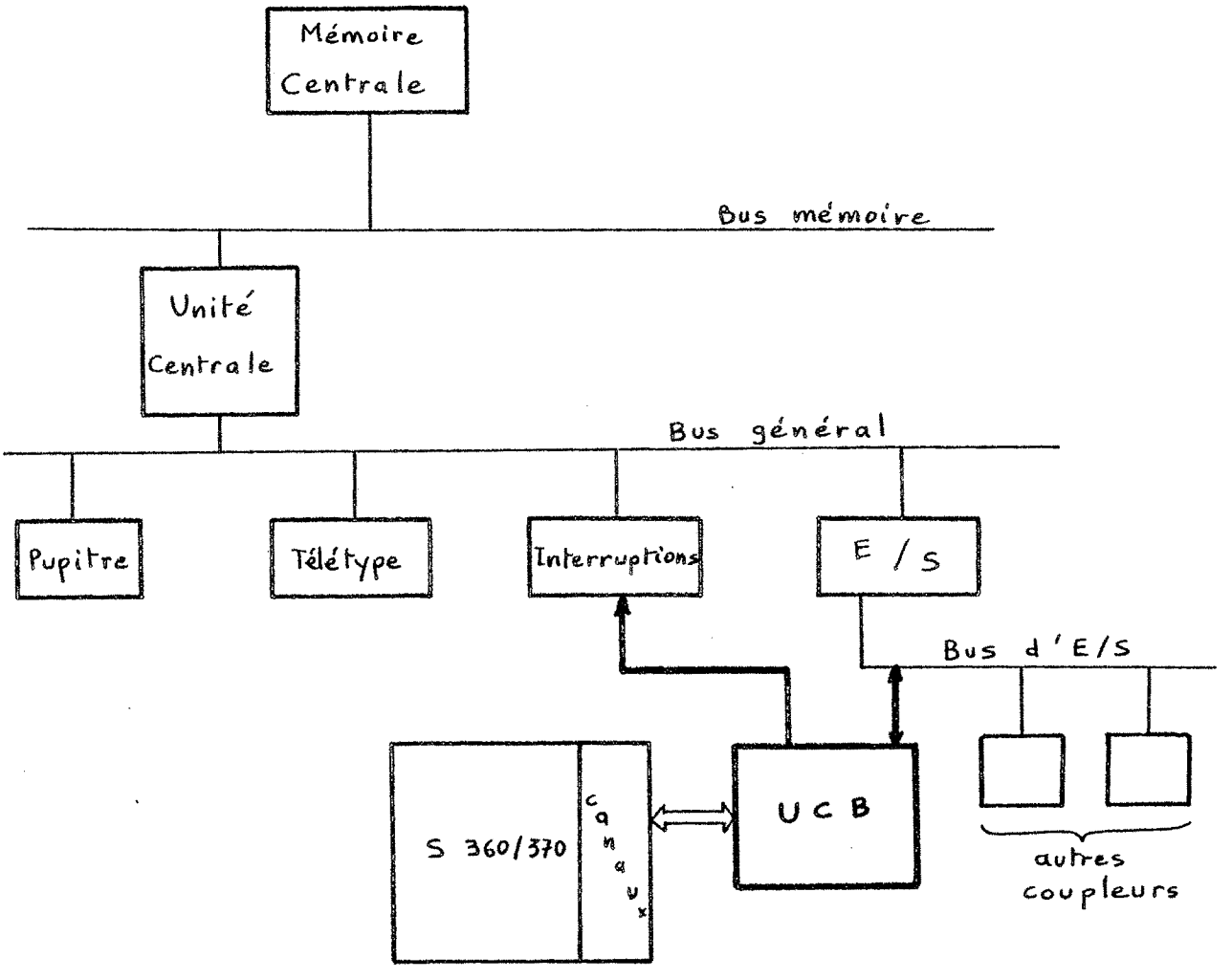


Fig.VI-5 Connexion du système 360 à un mini-ordinateur par l'intermédiaire de l'UCB

D'une façon générale, un mini-ordinateur dialogue avec ses périphériques suivant 2 modes :

- mode programmé (simple ou prioritaire)
- mode canal.

Indépendamment du transfert des données elles-mêmes, une opération d'E/S nécessite, d'une part des échanges d'adresses, de commandes, d'états, etc... et d'autre part une utilisation fréquente des nombreux niveaux et sous-niveaux d'interruptions.

Les informations nécessaires à l'E/S sont, pour le programmeur, manipulées dans des registres de l'unité d'E/S. Ce peut être des informations lues (états, données) ou écrites

(contrôle, commande, donnée) par l'unité centrale, au moyen d'instructions du type SIO qui précisent également l'adresse de l'unité et celle du registre concerné.

Ne pouvant ici envisager les cas particuliers de chaque mini-ordinateur, nous proposons un interface "banalisé" qui fait apparaître le périphérique "UCB" d'une façon comparable à celle vue par le programmeur.

VI-4-3-2 Définition d'un interface banalisé pour mini-ordinateur

Nous en donnons une représentation à la fig.VI-6, en montrant également les quelques registres internes de l'UCB accessibles au programme du mini-ordinateur. On trouvera donc :

- 8 ou 16 lignes de données bi-directionnelles permettant l'échange des données et des autres informations contenues dans les registres adressables

- 16 lignes d'adresses bi-directionnelles qui sont utilisées d'une part en sortie pour des accès direct à la mémoire du mini-ordinateur (le cas échéant), d'autre part pour que ce dernier puisse sélectionner un registre interne

- une ligne de lecture/écriture (R/W) précisant le sens du transfert sur les lignes de données

- 2 lignes de demande (INI) et d'acquittement (INTA) d'interruption

- 2 lignes de demande (RDMA) et d'acquittement (GDMA) d'accès direct à la mémoire.

- 2 lignes de synchronisation des échanges directs avec la mémoire (le cas échéant) : R et G.

Si nous nous replaçons dans le contexte de la fig.VI-5, un interface de ce type introduit un coupleur supplémentaire qui est chargé de l'adaptation de l'interface banalisé avec l'interface d'E/S, ou le bus d'E/S ou même le bus unique, du mini-ordinateur.

La réalisation de ce coupleur spécialisé serait laissée au soin de l'utilisateur de l'UCB (fig.VI-7).

Remarques :

-1- en ce qui concerne les interruptions, la ligne INI, si elle est unique, doit être reliée au niveau de priorité accordé à l'UCB ; il est aussi possible de l'utiliser conjointement aux lignes de données ou d'adresse qui définiraient précisément le niveau ou le sous-niveau de cette interruption (vectorisation).

-2- à partir de ce qui a été vu au $ VI-2-2, on peut envisager également des échanges entre le système 360 et le mini-ordinateur par l'intermédiaire de plusieurs sous-canaux gérés chacun par un processus contrôlé par le mini-ordinateur. Dans ce cas il ne s'agirait plus évidemment d'un fonctionnement CTC, mais d'une utilisation de la faculté de gestion simultanée de plusieurs adresses par l'UCB, les tables étant cette fois implantées dans le mini-ordinateur.

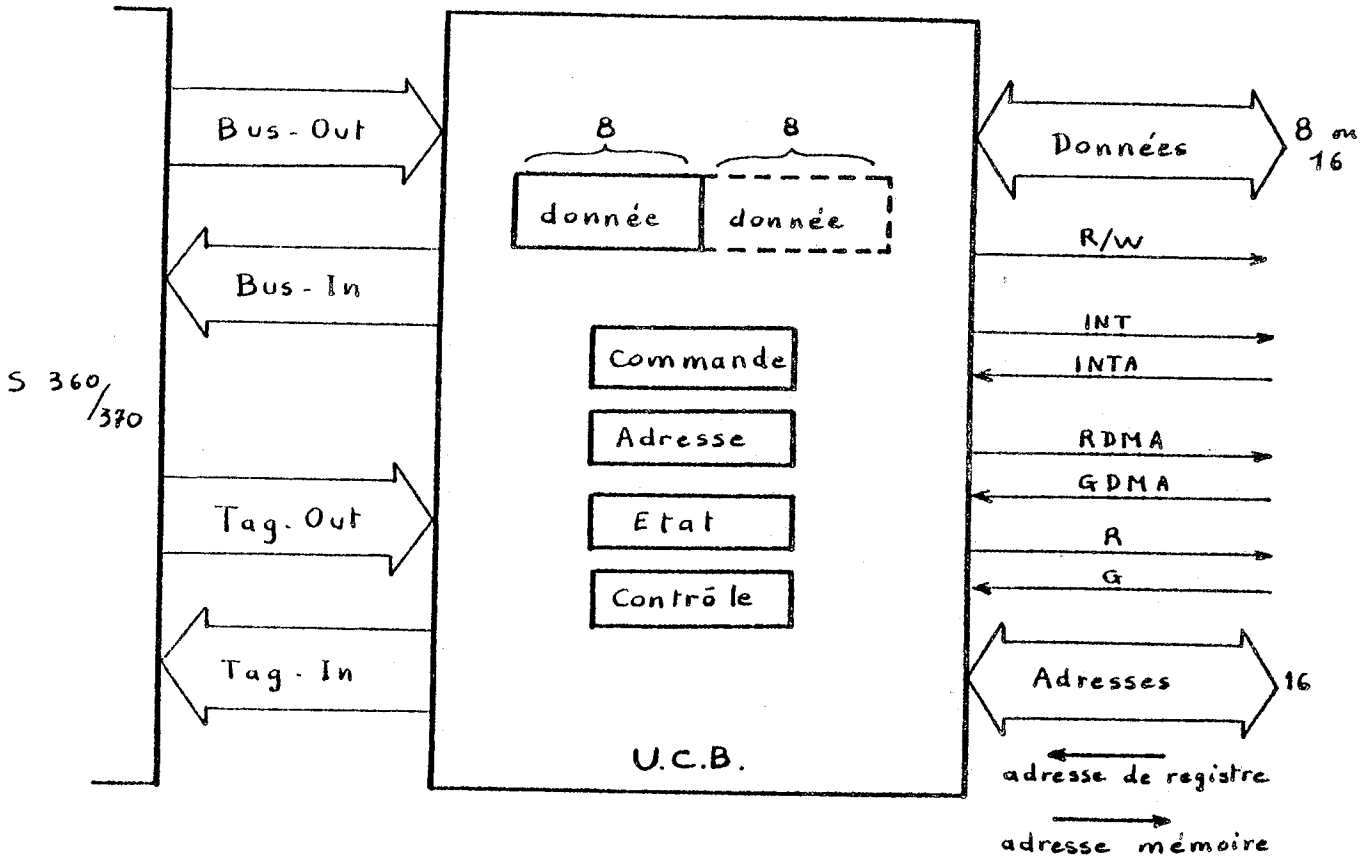


Fig.VI-6 Un interface banalisé pour connexion de l'UCB à un mini-ordinateur

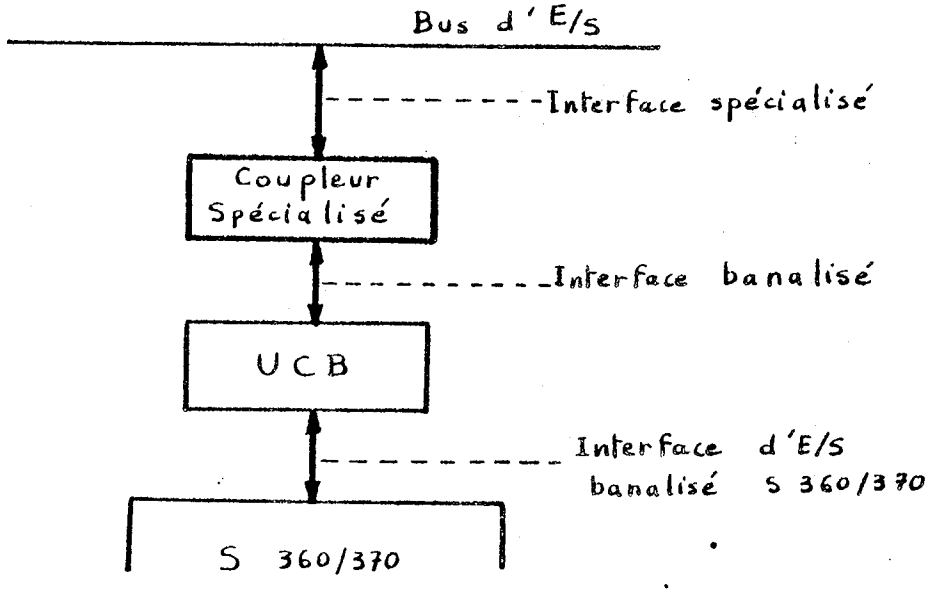


Fig.VI-7 Introduction d'un coupleur spécialisé

CONCLUSION

CONCLUSION

Nous avons essayé, dans cet ouvrage, de décrire l'ensemble des problèmes posés par le couplage d'une application, a priori quelconque, à un système informatique relativement évolué tel que le système IBM 360/370.

Nous avons également exposé les solutions utilisées qui, si elles n'ont pas toujours le mérite d'être des applications directes de formalismes théoriques, ont au moins eu celui d'aboutir à une réalisation opérationnelle.

D'un point de vue pratique, nous avons abordé :

- des aspects matériels aussi bien en architecture, qu'en interface et en "connectique"...

- des aspects logiciels :

  . au niveau micro-informatique, pour l'implantation des processus de gestion des interfaces
  . au niveau assembleur 360 pour la réalisation des primitives de test de l'UCB
  . au niveau programmation système pour essayer de comprendre et d'utiliser les traitements exécutés par un système d'exploitation en temps partagé sur les entrées-sorties effectuées par une machine virtuelle.

Il est possible que quelques-uns de ces aspects n'aient pas été traités de façon assez approfondie, pour des raisons diverses... D'autres, qui étaient néanmoins primordiaux quant au sujet abordé, ont peut-être été trop abondamment développés...

Nous avons néanmoins essayé de mettre en évidence une fonction de couplage au Système IBM 360/370, pour laquelle le microprocesseur 8X300 nous semble particulièrement adapté, à condition de posséder les outils de développement correspondants.

L'architecture modulaire utilisée permet également d'implanter aisément les fonctions d'exécution et/ou de dialogue, répondant aux besoins de tout utilisateur ou possesseur d'un tel Système.

Nous ne voudrions pas terminer sans exprimer un regret et formuler un espoir.

Regret dû au départ d'un matériel sur le fonctionnement duquel nous commencions à avoir des idées de plus en plus précises, tout au moins au niveau de ses mécanismes d'entrées-sorties.

Espoir que l'expérience ainsi acquise sur un type de matériel puisse être réinvestie sur un autre, ce qui sous-entend que se fassent jour des besoins, et qu'apparaissent dans les centres de calcul des documentations aussi abondantes que précises sur l'organisation matérielle des E/S des ordinateurs.

REFERENCES

REFERENCES

<ANC1> "Principes et mise en oeuvre des microprocesseurs"
F.Anceau - Cours ENSIMAG - Janv.78

<ANC2> "Architecture des ordinateurs" - Tome 2
F.Anceau - Cours de Ecole d'été du Forez - Mai 78

<ANC3> "Microprocesseurs et applications"
F.Anceau - Cours de Ecole d'été IRIA.EDF.CEA - 1977

<ANG> "Moyens de communication entre machines virtuelles"
Y.Angelidès - Rapport de DEA Grenoble - Juin 77

<ANS> "Système interactif dans un environnement réseau. Connexion d'une machine à mémoire virtuelle (IBM 360-67) au réseau Cyclades.
J.P.Ansart - Thèse Dr.Ingénieur INPG - Fév.76

<BIN> "Designing a microprocessor driven multipurpose peripheral controller"
R.F.Binder - Computer Design - Avr.79

<BEK> "Le système de production d'assembleurs GAGE"
Y.Bekkers - Atelier de micro-informatique de Grenoble - Oct.79

<BEL> "Systèmes de programmation générateurs de machines virtuelles"
M.Bellino, L.Siret, J.Guillou, M.Rey, J.P.Le Heiget, J.P.Dupuy - Cours CP-67 C.I.C.G. - Déc.73

<BER> "I/O Microprogramming, an overview"
H.Berndt - Euromicro Newsletters, vol.2, n°3 - Avr.76

<BSA> "Conception d'un processeur de base de donnée adapté à des petits systèmes"
G.Berger-Sabbatel, P.Navaux, M.Sarre - Session de perfectionnement INPG - Oct.78

<CHU> "Computer organisation and microprogramming"
Y.Chu - Prentice Hall - 1972

<DAN> "SYNCOP, Implémentation sous CP-67"
Ng.Dang, R.Fournier, V.Quint - IMAG - Sept.75

<DUM> "Introduction à la programmation système"
J.Du Masle - Cours Institut de Programmation de Grenoble - 1974

<ENS> "Multi-processors organisation. A survey"
P.Enslow - Computing Surveys, vol.9, n°1 - Mar.78

<FIN> "Les E/S dans les systèmes d'exploitation"
L.Finet - Thèse 3ème cycle USMG - 1973

<FLY> "Some computer organisations and their effectivness"
M.Flynn - IEEE Transactions on computers, vol.C21, n°9 - Sept.72

<GAR> "Spécifications de réalisation de la Station de Transport ST2 portable"
Garcia et al. - Réseau Cyclades SCH.536.2 - Mai 75

<GRA> "CIGALE, implementation, tool and technics"
J-L.Grange - SEAS Dublin - Sept.75

<IBM1> "Systems 360/370 Principles of Operations"
Manuels IBM GA22.6821 et GA22.7000.2

<IBM2> "I/O Interface Channel to Control Unit"
Manuel IBM GA22.6974.1 - Juil.72

<IBM3> "System 360 model 67 Functionnal characteristics"
Manuel IBM GA27.2719

<IBM4> "IBM 2870 Multiplexor Channel Theory of Operation"
Manuel IBM Y22.2908.0

<IBM5> "CP-67/CMS Control Program Cambrigde Monitor System
User's guide"
Manuel IBM GH20.0859

<IBM6> "System 360 Special Feature Channel to Channel Adapter"
Manuels IBM A22.6892.1 et 223.2901.0

<IBM7> "IBM 2701 Data Adapter Unit"
Manuel IBM A22.6844.1

<IBM8> "System 360 Assembler Programmers' Guide"
Manuel IBM C26.3756

<LIL> "Du microprocesseur au micro-ordinateur"
H.Lilen - Editions Radio - 1977

<MAD> "Présentation du Système MADAM"
G.Baille, J.Laurent, J.P.Schoellkopf - Rapport de contrat IRIA-SESORI 78.204 - Juin 79

<MAR> "Mécanisme de communication par Bus série pour des Réseaux locaux"
M.Marinescu - Thèse 3ème cycle - Sept.78

<POUJ> "Système réparti CORAIL"
G.Poujoulat - Rapport de Recherche IMAG n°53 - Nov.76

<POUZ> "Présentation du Réseau CYCLADES"
L.Pouzin - Note Cyclades GAL 506 - Nov.73

<PRO> "Structure et technologie des ordinateurs"
A.Profit - A.Colin - 1970

<QUI> "Protection logicielle contre les erreurs dans un Réseau d'ordinateurs hétérogènes. Application au réseau CYCLADES"
V.Quint - Thèse Dr.Ingénieur INPG - Déc.76

<SAE> "Conception d'un périphérique intelligent d'accès au réseau CYCLADES. Réalisation sur une architecture à microprocesseurs multiples"
R.Saettone - Thèse Dr.Ingénieur INPG - Juil.79

<ZAK1> "Les microprocesseurs"
R.Zaks, P.Le Beux - Sybex C4 - 1977

<ZAK2> "Techniques d'interface aux microprocesseurs"
R.Zaks, A.Lesea - Sybex C5 - 1978

<ZIM1> "Transport protocol standard end-to-end protocol for hererogeneous computer network"
H.Zimmerman, M.Elie - Réseau Cyclades SCH.5919.2 - Mai 75

<ZIM2> "Insertion d'une station de transport dans un système d'exploitation"
H.Zimmerman - Réseau Cyclades SCH-546 - Janv.75

<8X> "Environnement et Application du microprocesseur bipolaire 8X300"
Brochure Signetics.RIC - 1977

ANNEXES

ANNEXE 1

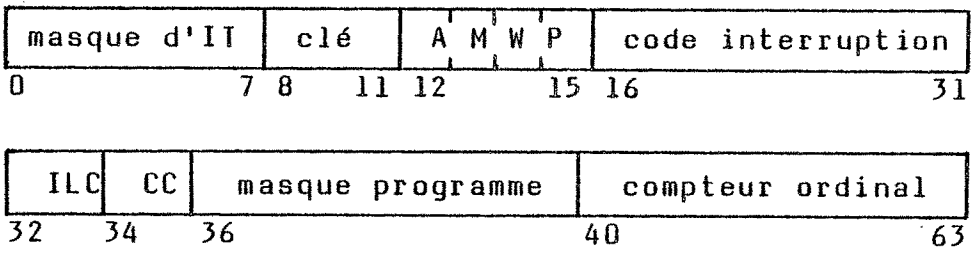
COMPLEMENTS SUR LE SYSTEME 360 ET LE CANAL MULTIPLEXEUR 2870 <IBM1> - <IBM2> - <IBM4>

A1-1 Format et composition des mots de contrôle des E/S

a) le PSW est le double-mot d'état programme.

Il n'est pas spécifique des entrées-sorties. Il indique les états de l'Unité Centrale et du programme à un instant donné. Lors d'une interruption, il est rangé à un emplacement de la zone mémoire débanalisée (adresses 0 à 120) correspondant au type de l'interruption, et remplacé par un nouveau PSW qui contient le nouvel état de la machine pendant le traitement de cette interruption.

Son format est le suivant :



La signification des différents champs est :

- masque d'interruption : les bits 0 à 6 permettent d'autoriser (valeur 1) ou de suspendre (valeur 0) les interruptions d'E/S venant des canaux 0 à 6 (le canal multiplexeur est le canal 0). Le bit 7 autorise ou suspend l'interruption externe venant soit d'un signal externe, soit de l'horloge, soit de la touche "Interrupt" du pupitre opérateur.

- clé : c'est la clé de protection mémoire. Ces 4 bits (8 à 11) doivent correspondre à ceux de la clé du bloc (2 K.o) accédé par le PSW courant.
- A : le bit 12 indique le code interne utilisé par le CPU (code ASCII si A = 1, code EBCDIC si A = 0).
- M : si le bit 13 a la valeur 1, l'interruption causée par une erreur machine est prise en compte. Sinon elle est ignorée.
- W : le bit 14 à 1 place le CPU en mode "attente d'interruption". Sinon il est en mode "exécution".
- P : le bit 15 indique que le CPU est en mode "problème/ esclave" (P = 1), ou en mode "superviseur/maître" (P = 0).
- code interruption (bits 16 à 31) : il fournit la cause de l'interruption. Dans le cas d'une interruption d'E/S, il s'agit de l'adresse de l'unité responsable.
- ILC : les 2 bits 32 et 33 donnent la longueur en demi-mots (2 octets) de la dernière instruction exécutée.
- CC : les bits 34 et 35 sont positionnés après certaines instructions, en particulier les instructions d'E/S (voir paragraphe II-3).
- masque de programme (bits 36 à 39) : il permet de suspendre les conditions d'interruption survenant après certaines opérations de calcul.
- compteur ordinal (bits 40 à 63) : il contient l'adresse de la prochaine instruction qui doit être exécutée.

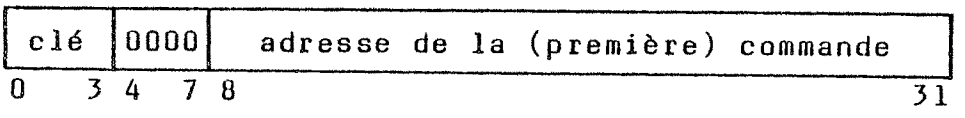
Remarque : toutes ces informations sont en réalité dispersées dans plusieurs registres et bascules de l'Unité Centrale, et ne sont regroupées qu'au moment d'une interruption.

Le PSW ne peut être modifié, en totalité, que par une interruption ou l'instruction privilégiée LPSW, et en partie, que par les instructions privilégiées SSM (qui positionne le masque d'interruption), SSK qui positionne la clé de protection mémoire et l'instruction normale LSPW (qui positionne le masque de programme).

b) le CAW est le mot d'adresse du canal, qui se trouve à l'adresse réservée 72.

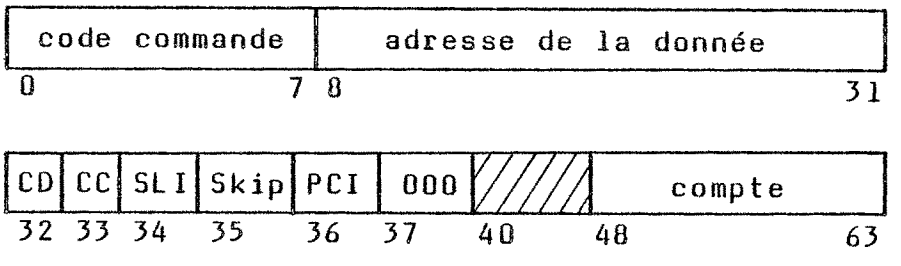
Il contient essentiellement l'adresse du premier mot de commande du programme canal, ainsi que la clé de protection mémoire telle que les seuls blocs de 2048 octets qui pourront être accédés par le canal, en lecture ou en écriture, sont ceux qui possèdent la même clé.

Son format est le suivant :



c) les CCW , mots de commande du canal, composent le programme canal.

Leur format est :



Les bits 0 à 7 définissent la commande à envoyer à l'unité d'E/S (voir $ II-3-1-).

Les bits 8 à 31 et 48 à 63 donnent l'adresse et la longueur du bloc de donnée à transférer, le cas échéant.

Les bits 40 à 47 sont ignorés.

Les bits 32 à 36 sont des indicateurs (flags) dont la signification est la suivante :

- CD (chaînage de donnée) : s'il est à 1, et quand la commande courante est terminée, le canal ré-exécute la même fonction mais avec des données différentes.

- CC (chaînage de commande) : s'il est à 1, le canal exécute la commande qui se trouve dans le CCW suivant.

- SLI (suppression de longueur incorrecte) : s'il est à 1, le canal ne tient pas compte de l'erreur qui se produit quand le nombre d'octets transférés est différent de la longueur indiquée dans le compte.

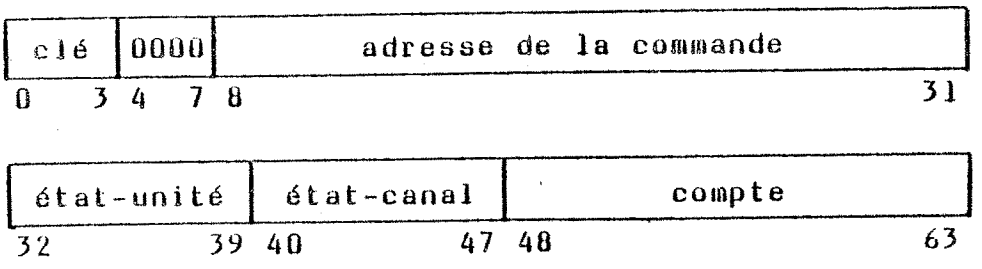
- Skip : ce bit n'est significatif que pour des commandes de lecture et d'analyse et, s'il est à 1, indique au canal de ne pas transférer en mémoire les octets reçus.

- PCI (interruption contrôlée par programme) : ce bit à 1 permet de suivre le déroulement d'un programme canal en générant une interruption, même s'il y a chaînage.

Remarque : la commande "Transfert dans le programme canal" ignore les indicateurs et le compte ; l'adresse de la donnée est en fait celle du prochain CCW à prendre en compte.

d) le CSW , mot d'état du canal, fournit au programme l'état d'une unité d'E/S et/ou les conditions dans lesquelles s'est terminée une opération d'E/S. Il est rangé à l'emplacement 64 de la mémoire à chaque interruption d'E/S, et parfois à l'exécution des instructions SIO, TIO et HIO.

Son format est le suivant :



La clé de protection mémoire a la même signification que celle du CAW.

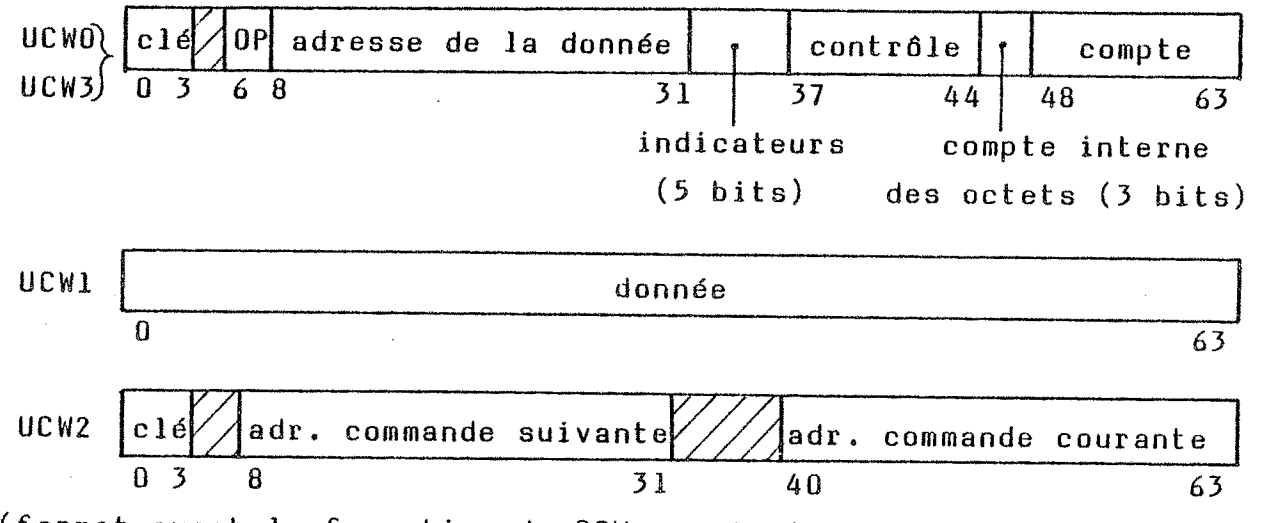
Les bits 8 à 31 contiennent l'adresse du dernier CCW utilisé + 8, si le CSW est rangé par une interruption marquant la libération d'un sous-canal (fin normale ou non d'une E/S). Ils ont des valeurs quelconques si le CSW est rangé lors du lancement d'une instruction d'E/S.

Les bits 32 à 47 forment 2 octets qui donnent les conditions, aux niveaux de l'unité et du canal, qui ont causé le rangement du CSW. Nous expliciterons les 8 premiers au paragraphe suivant.

Les bits 48 à 63 contiennent le nombre d'octets restant à transférer.

e) Les UCW sont les double-mots de contrôle des unités et forment la mémoire locale du canal multiplexeur.

Dans ce modèle, il y a 4 UCW par sous-canal, donc par opération d'E/S, dont les formats sont les suivants :



(format avant la formation du CSW ; puis identique à ce dernier)

Remarques :

-1- UCW0 est assemblé dans le registre de contrôle du canal principal à partir du CCW courant. Le champ OP de 2 bits détermine l'opération effectuée par le canal. Celui-ci peut ainsi reconnaître seulement les 3 types de commandes suivants :

OP	
00	.....sous-canal libre
01	.....écriture ou contrôle
10	.....lecture ou analyse
11	.....lecture arrière

La distinction entre lecture et analyse, ou entre écriture et contrôle est faite par l'unité.

Le champ des 3 bits 45 à 47 est utilisé pendant l'assemblage et le désassemblage des double-mots de données (8 octets) échangés avec la mémoire centrale.

-2- UCW1 contient les données. Pendant une lecture, il est rangé en mémoire locale à chaque transfert d'un octet, et en mémoire centrale tous les 8 octets ou quand le compte s'annule. Pendant une écriture, il est rempli juste après l'obtention du CCW.

-3- UCW2 reçoit ses informations du CAW, dès l'initialisation du Start I/O. Quand un CSW doit être fournit au programme, il est assemblé dans le registre de contrôle, à partir de UCW2, des états du canal et de l'unité et du compte restant dans UCW0.

-4- UCW3 n'est utilisé que s'il y a chaînage de données, pour une opération de lecture.

-5- Les différentes phases des opérations de lecture et d'écriture, au niveau de tous les organes mis en jeu, sont résumées dans les fig.A1-1 et A1-2.

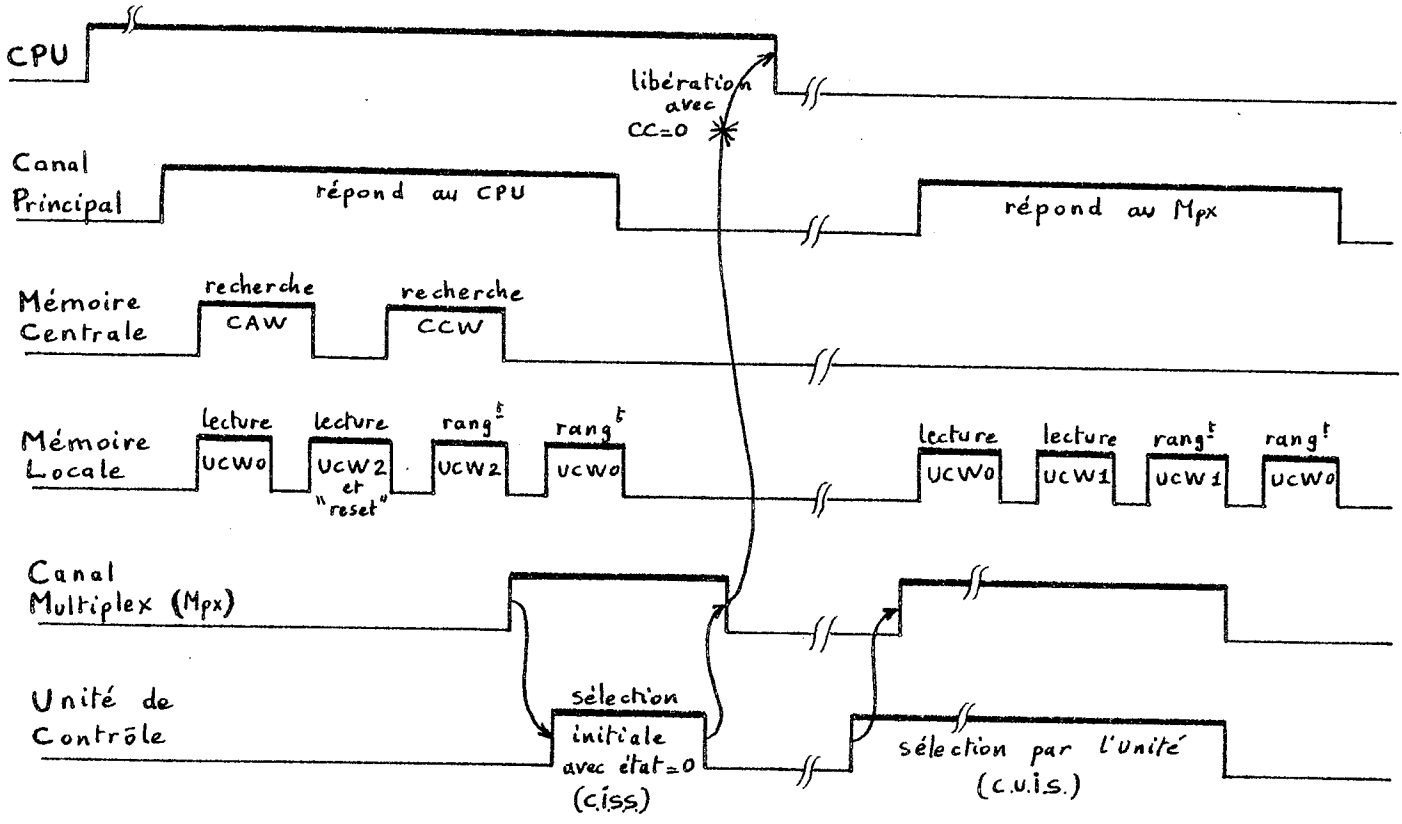


Fig.A1-1 Opération de lecture

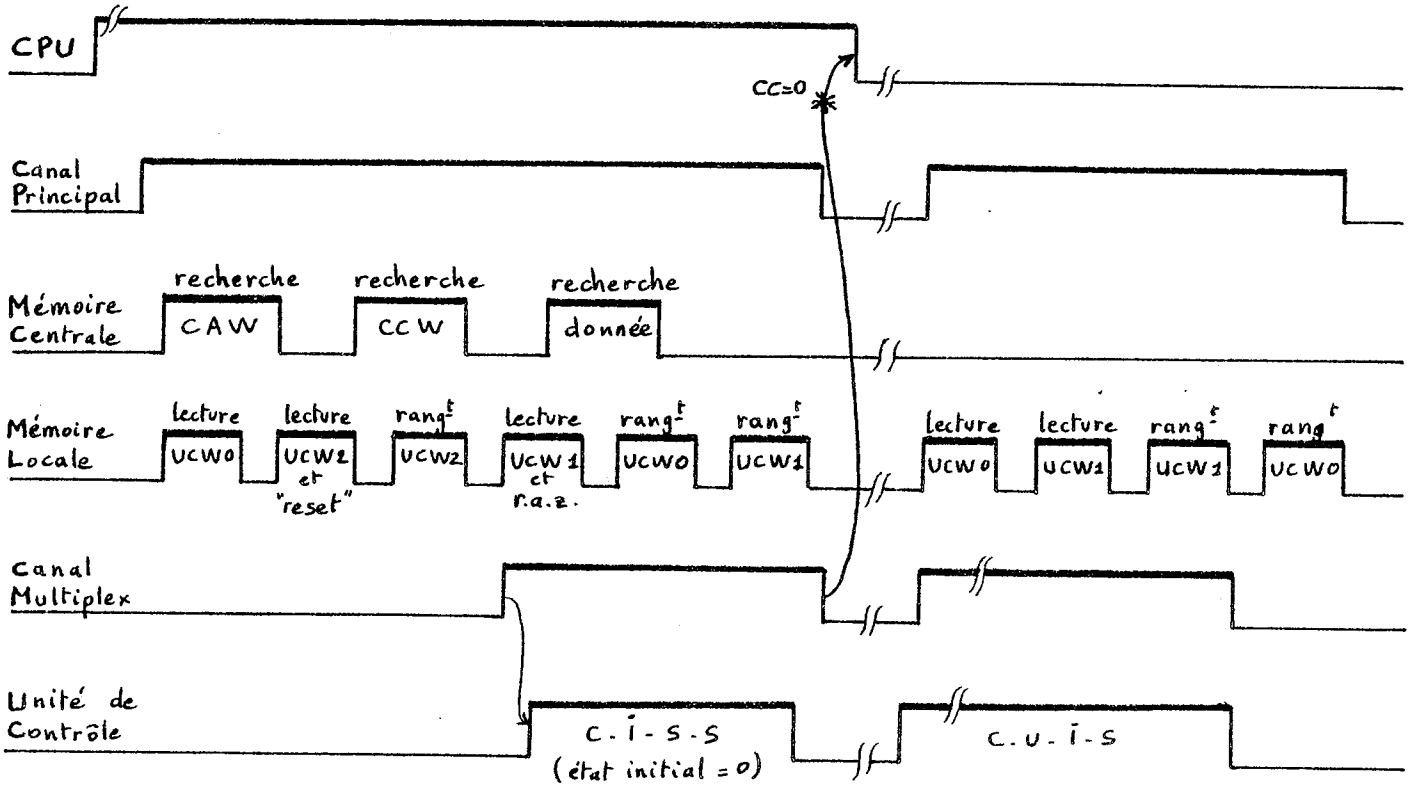


Fig.A1-2 Opération d'écriture

A1-2 Significations des bits des octets d'état-unité et d'analyse

a) état-unité (bits 32 à 39 du CSW) :

- Attention indique, pour certains organes d'E/S, l'apparition d'un événement externe, indépendant de toute opération d'E/S.

- Modification d'état change la signification de certains autres bits de l'état-unité. Avec "occupé", il indique que c'est l'unité de contrôle et non le périphérique qui est occupé. Avec "fin sur unité", il y a un saut d'un CCW dans le programme canal, en cas de chaînage de commandes. Il peut être envoyé seul en réponse à une commande TIO, si l'unité ne peut fournir son état.

- Fin sur unité de contrôle indique qu'une unité devient disponible après avoir répondu "occupé" à une précédente sélection, ou qu'une unité détecte une anomalie après avoir envoyé "fin sur canal". Cet état n'est utilisé que pour des unités de contrôle partagées.

- Occupé : l'unité et/ou l'organe d'E/S ne peuvent effectuer la commande.

- Fin sur canal permet de libérer le canal lors d'une opération d'E/S impliquant un transfert de donnée, ou lorsqu'une commande immédiate est acceptée par l'organe d'E/S.

- Fin sur unité indique la fin de l'E/S au niveau de l'organe d'E/S. Il peut être associé à "fin sur canal" si les 2 conditions sont simultanées. Il est utilisé aussi pour annoncer au canal le passage d'une unité de contrôle ou d'un périphérique à l'état "prêt".

- Erreur sur unité est positionné quand l'unité détecte une anomalie, qui est alors détaillée dans l'octet d'analyse. Il y a toujours arrêt du chaînage, le cas échéant.

- Exception sur unité a une signification précise pour chaque type d'unité. Il indique en général une situation anormale ou exceptionnelle, mais pas nécessairement une erreur à corriger (exemple : magasin vide d'un lecteur de cartes). Il peut être associé à "attention" en cas de découverte d'une condition inhabituelle, concernant une E/S, survenant après l'envoi de "fin sur unité".

Remarque : ces quelques indications sont simplifiées et le sens précis de chacun de ces états dépend de l'unité et de l'organe d'E/S qui l'émet.

b) Octet d'analyse :

Seuls les 6 bits de poids fort de cet octet ont une signification commune à la plupart des unités d'E/S du système.
Ce sont :

- Commande rejetée : l'unité ne peut exécuter la commande reçue, soit parce qu'elle ne dispose pas des organes nécessaires, soit qu'il lui manque une information

- Intervention requise : la commande ne sera pas exécutée car elle nécessite une intervention de l'opérateur : imprimante sans papier, périphérique non prêt...

- Erreur sur Bus-out : un octet de commande ou de donnée (en écriture) a une parité incorrecte

- Erreur machine : un mauvais fonctionnement a eu lieu entre l'interface d'E/S et le support externe de l'information. Un arrêt de l'E/S en cours se produit seulement si l'erreur empêche la suite normale de l'opération

- Erreur de donnée : une donnée invalide a été détectée ou est probable dans le milieu extérieur ; même suite que dans le cas précédent

- Surcharge : le canal a trop tardé à répondre à la demande de données (cas de certaines unités non tamponnées) ; ou il a envoyé la nouvelle commande trop tard lors d'un chaînage.

A1-3 Interfaces du canal multiplexeur

a) Interface avec le CPU (fig.A1-3) :

Sur cette figure, ne sont représentées que les lignes principales mises en jeu dans les situations normales. Les 2 opérations essentielles sont :

- la sélection du canal par le CPU : dans ce cas, le CPU place l'adresse du canal et de l'unité concernés sur le bus UABO et active les lignes "Select Channel" et "SIO" (ou "TIO", ou "HIO", ou "Test Channel" suivant l'opération d'E/S). Le canal, s'il n'est pas disponible, envoie le code condition correspondant sur les 2 lignes "Condition Bits" et active la ligne "Release" qui libère le CPU.

Si le canal est libre, il entame le dialogue avec le BCU et la mémoire, pour accéder au CAW, puis au CCW (dans le cas d'un SIO). Le CPU n'est libéré qu'après sélection de l'unité s'il n'y a pas d'erreur.

- la demande d'interruption au CPU par le canal : celui-ci active la ligne "Interrupt" et attend que le CPU, après vérification de la priorité et en fonction de sa disponibilité, autorise l'interruption par "Interrupt Response". Le canal place alors l'adresse de l'unité demandeur sur le bus UABI, et active la ligne "Release". Le CPU range cette adresse et celle du canal dans le PSW, pendant que le canal lance un cycle d'écriture du CSW en mémoire.

Remarques :

-1- les opérations précédentes ne peuvent avoir lieu que si la ligne "Channel Available" est à 1, indiquant que le canal est actif

-2- la ligne "IPL", associée à "Select Channel", indique que le CPU utilise ce canal pour une opération de chargement d'un programme d'initialisation ; l'adresse de l'unité étant positionnée par l'opérateur au pupitre de commande

-3- la ligne "Master Reset" effectue un "reset" général de toutes les unités connectées au canal.

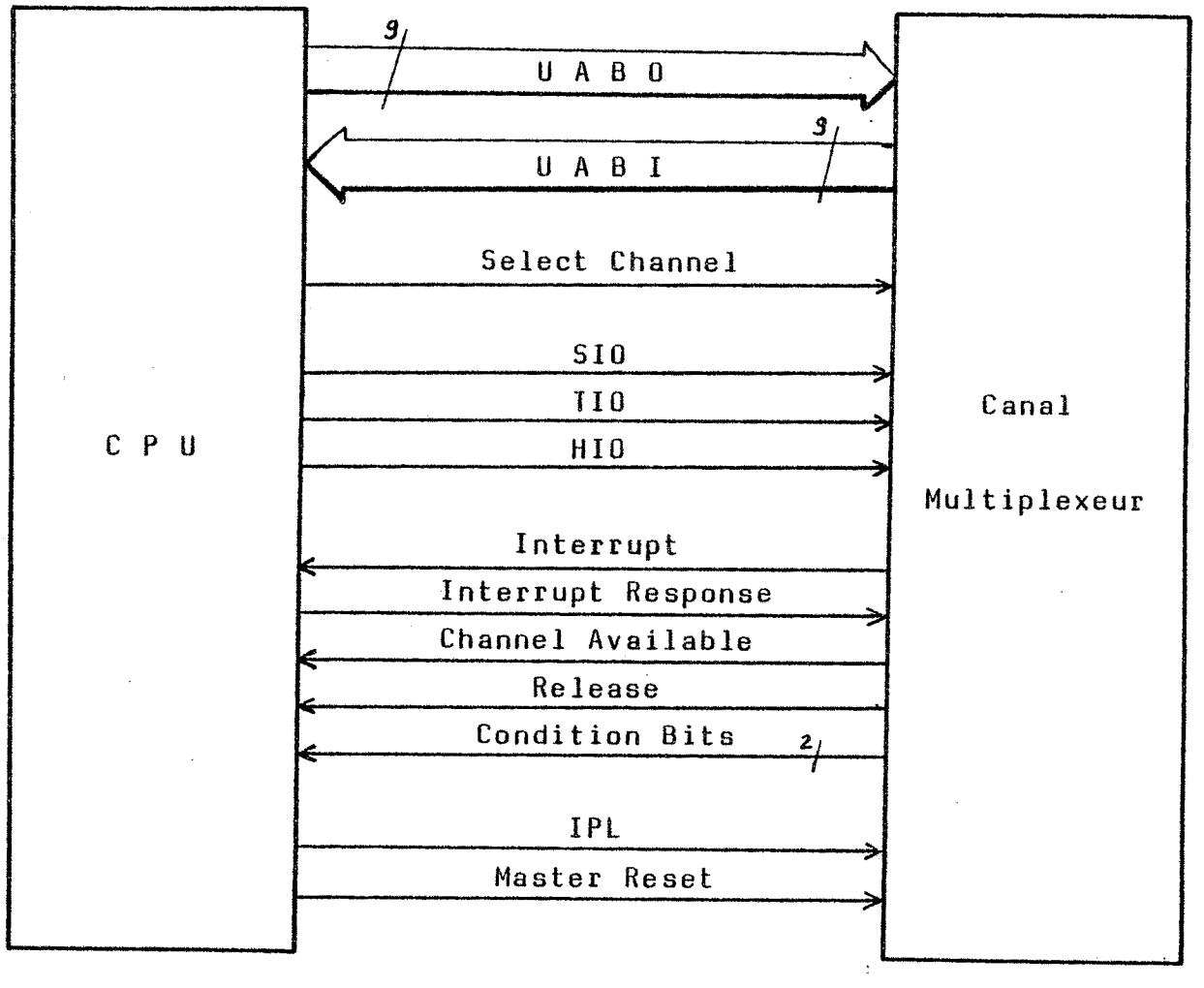


Fig.A1-3 Les principales lignes de l'interface Canal-CPU

b) Interface avec le BCU et la mémoire (fig.A1-4) :

Lorsque le canal veut accéder à la mémoire, il effectue la séquence suivante :

- activation de "Storage Request"
- attente de "BCU Response"
- adresse de la donnée sur SAB et positionnement de "Address Valid"

Le BCU indique alors le début du cycle mémoire en envoyant une impulsion sur "Accept", suivie du signal "BCU Data Request". Ce signal indique normalement qu'une donnée est attendue sur SDBI (lorsque le canal écrit en mémoire, il positionne en plus la ligne "Store"), mais il est actif aussi lors d'une lecture ; dans ce cas, la donnée apparaît sur SDBO environ 200 ns après une impulsion sur la ligne "Advance Pulse".

Remarque : la clé de protection mémoire est envoyée en même temps que l'adresse de la donnée avec un bit de parité.

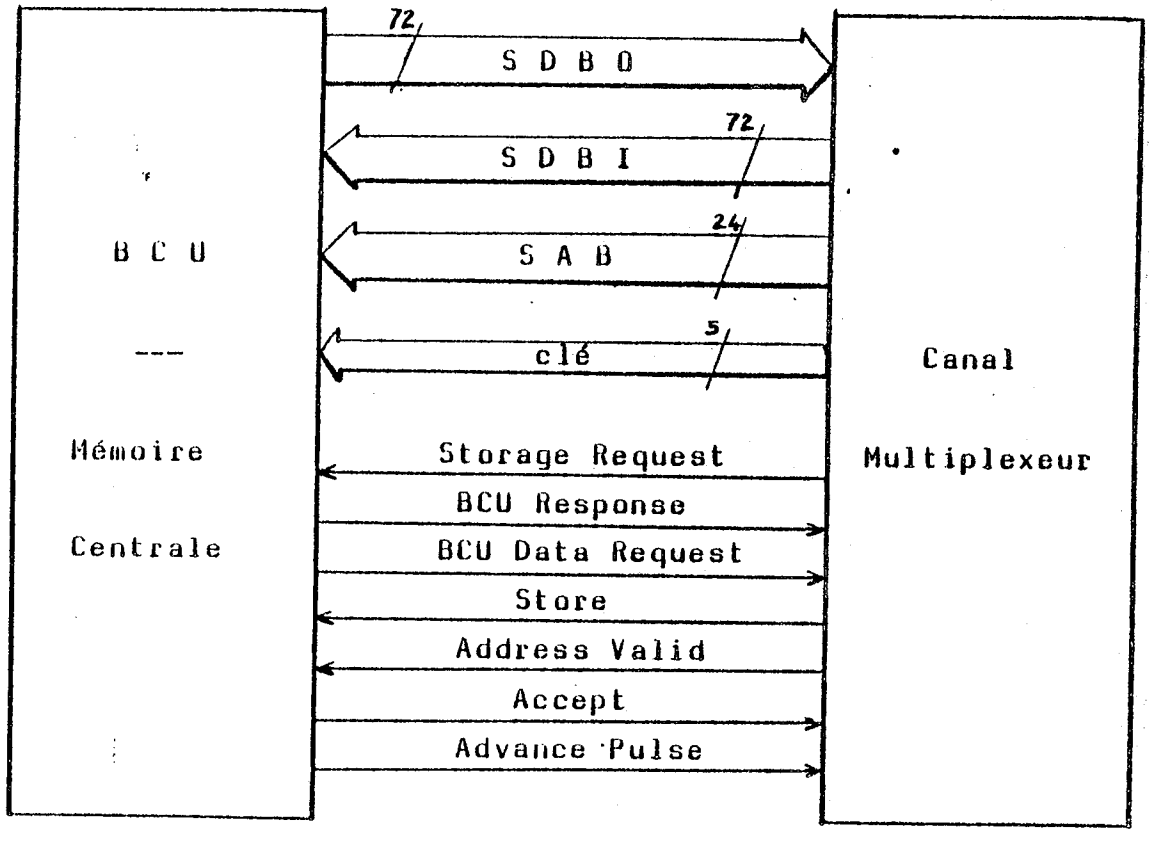


Fig.A1-4 Les principales lignes de l'interface Canal-BCU/Mémoire

A1-4 Conventions de communication entre programmes <IBM8>

Nous explicitons ici les quelques renseignements donnés dans le Chapitre V au sujet de l'appel de sous-programmes "Assembleur 360" par un programme "Fortran", en décrivant rapidement les moyens habituellement utilisés dans le système 360 pour permettre la communication dynamique entre les programmes.

a) Les outils de liaison :

- les registres :

- R15 contient l'adresse du point d'entrée dans le programme appelé
- R14 contient l'adresse de retour dans le programme appelant
- R13 contient l'adresse absolue d'une zone de sauvegarde des registres
- R0 et R1 peuvent contenir les adresses de zones de travail mises en commun par l'appelant et l'appelé, servant au transfert de paramètres.

- la zone de sauvegarde :

adresse	mot	contenu
A	1	utilisé par compilateur Fortran (par exemple)
A+4	2	adresse de la zone de sauvegarde précédente
A+8	3	- - - - - suivante
A+12	4	adresse de retour au programme appelant (R14)
A+16	5	adresse d'entrée (R15)
A+20	6	contenu de R0
A+24	7	contenu de R1
A+68	18	contenu de R12

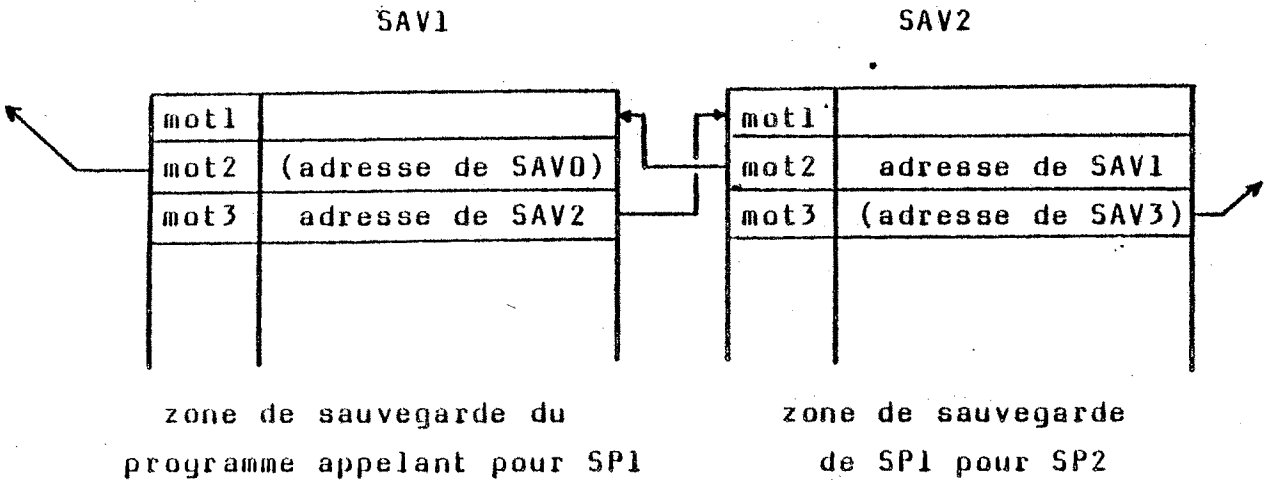
Remarques :

-1- Si le programme appelant n'est pas lui-même un sous-programme, et si le programme appelé n'en appelle pas un autre, les mots 2 et 3 ne sont pas utilisés.

-2- Au contraire, si le programme appelé (SP1 par exemple) en appelle un autre (SP2), il devra effectuer en plus :

- la préparation de sa propre zone de sauvegarde (SAV2)
- la liste d'arguments pour SP2
- la séquence d'appel de SP2
- la sauvegarde des adresses des 2 zones de sauvegarde dans les mots 2 et 3 des zones correspondantes, c'est-à-dire : adresse de SAV2 dans le troisième mot de SAV1 et adresse de SAV1 dans deuxiéme mot de SAV2.

On peut résumer ceci dans le schéma suivant :



b) le passage des paramètres :

Dans le cas d'un appel de programme avec paramètres, tel que, par exemple :

CALL SUB (A,B,C)

le registre R1 (par exemple) contient l'adresse de la zone de travail suivante :

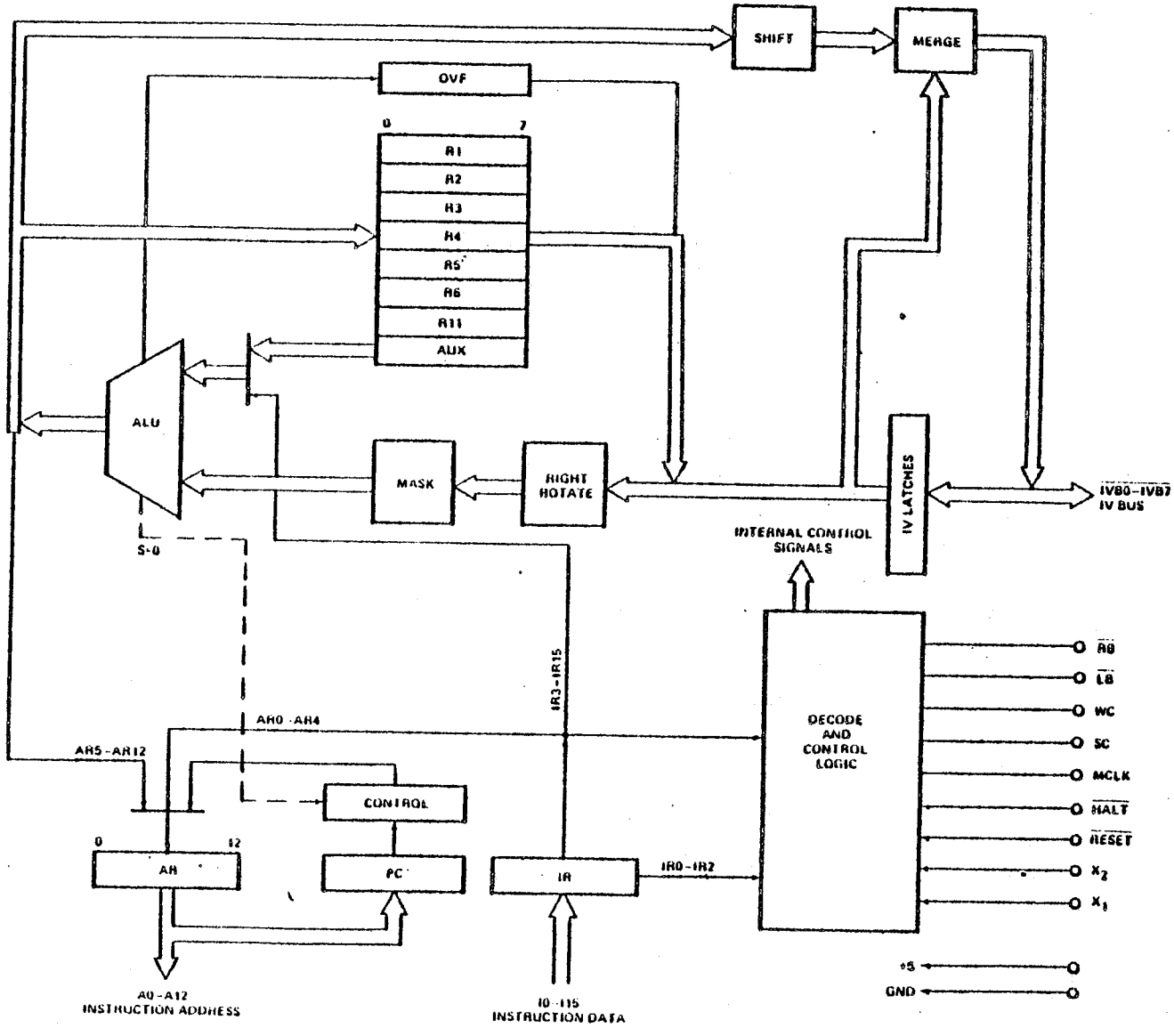
00000000	adresse du paramètre A
00000000	- - - B
10000000	- - - C
0 7	8 31

Les paramétres A, B, C, sont des variables, des tableaux ou des sous-programmes, et peuvent être des données ou des résultats.

ANNEXE 2

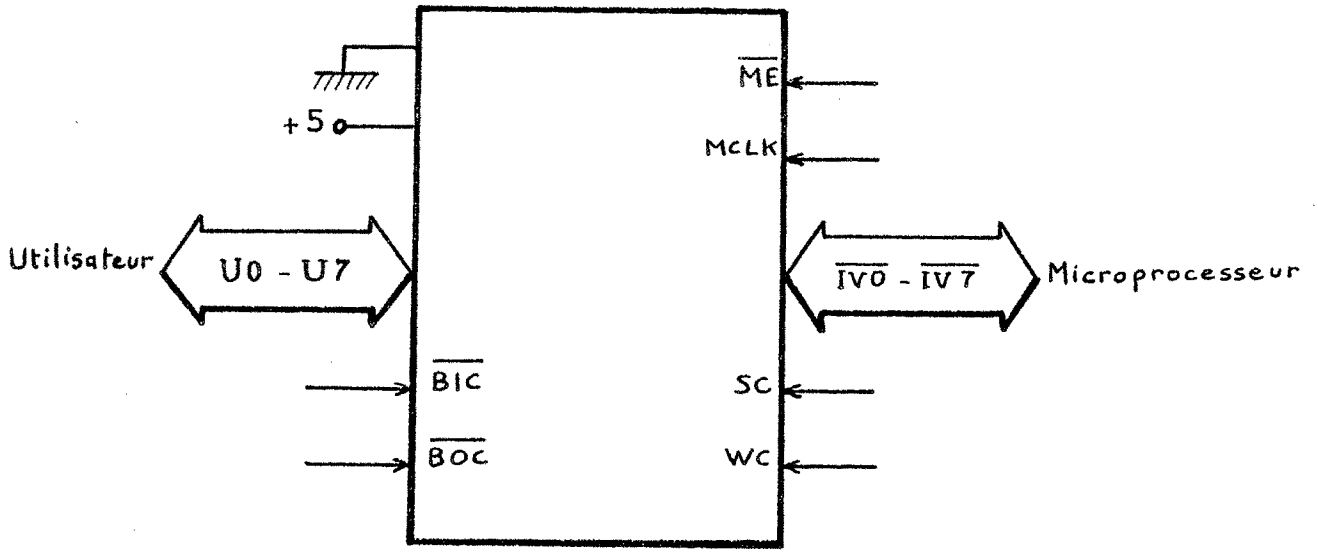
COMPLEMENTS SUR LE MICROPROCESSEUR 8X300
<8X>

A2-1 Architecture interne



A2-2 Les E/S dans un système 8X300

a) Description des boîtiers d'E/S compatibles :



Sur ce schéma, on distingue :

- 2 bus directionnels de 8 bits : le bus utilisateur (U0-U7) et le bus d'E/S du 8X300 ($\overline{\text{IV-Bus}}$) définissant 2 ports indépendants : le port utilisateur et le port microprocesseur.

- 2 lignes de contrôle de la direction des lignes du port utilisateur :

  $\overline{BIC}$ = 0 et $\overline{BOC}$ = 1 : lignes en entrée

  $\overline{BIC}$ = 1 et $\overline{BOC}$ = 0 : lignes en sortie

- une ligne de sélection du port microprocesseur ($\overline{ME}$)

- 3 lignes de contrôle du CPU : WC, SC et MCLK.

Remarque : Ces boîtiers sont disponibles en 4 versions (par rapport au port utilisateur) : synchrone/asynchrone et 3-états/collecteur ouvert.

b) Fonctionnement des 2 ports :

Le fonctionnement du port utilisateur ne dépend que de l'état des lignes $\overline{BIC}$ et $\overline{BOC}$, sauf la version synchrone où une donnée ne peut être entrée que pendant le quatrième quart du cycle ($ III-3-1 et fig. III-4).

Une priorité lui est donnée par rapport au port microprocesseur, dans cette même phase d'entrée.

Le fonctionnement du port microprocesseur (fig. A2-1) est sous la dépendance des lignes $\overline{ME}$, SC, WC, MLCK et d'un registre d'état.

Ce registre doit avoir été préalablement positionné par l'égalité entre l'adresse du boîtier (pré-établie) et une adresse envoyée sur le Bus-IV. Rappelons qu'une adresse est sur le Bus-IV quand SC = 1 et WC = 0.

$\overline{ME}$	SC	WC	MCLK	$\overline{BIC}$	Registre d'état	Fonction
0	0	0	x	x	Validé	Sortie donnée
0	0	1	1	1	Validé	Entrée donnée
0	0	1	1	0	x	Inactif
0	0	x	x	x	Non validé	Inactif
0	1	0	1	x	x	Entrée adresse
1	x	x	x	x	x	Inactif

Fig. A2-1 Tableau simplifié des fonctions du port microprocesseur

c) Exemple d'un système 8X300 (fig.A2-2) :

Dans ce système, les boîtiers d'E/S sont connectés au banc gauche (sélecté globalement par $\overline{LB}$ = 0) tandis que la mémoire de travail est adressée par le banc droit ($\overline{RB}$ = 0) (sachant que $\overline{LB}$ et $\overline{RB}$ sont complémentaires).

L'adressage d'un boîtier particulier se fait ici de la façon suivante :

- positionnement du registre d'état, c'est-à-dire envoi de l'adresse du boîtier avec SC = 1
- lecture ou écriture d'une donnée, sur le banc gauche.

Tant qu'une autre adresse n'est pas envoyée et que $\overline{LB}$ = 0, le boîtier est sélecté. Si un autre boîtier doit être adressé, la non concordance des adresses va annuler le bit d'état du précédent.

Remarque : Un moyen plus rapide de sélection des boîtiers d'E/S est, après positionnement de leur registre d'état lors de l'initialisation du système, de les sélecter seulement par la ligne $\overline{ME}$, par exemple par décodage de bits supplémentaires dans l'instruction.

L'adressage et l'utilisation se font alors en une seule instruction. Ceci est particulièrement intéressant si le boîtier n'est sélecté que pour une opération d'une instruction (voir Chapitre III).

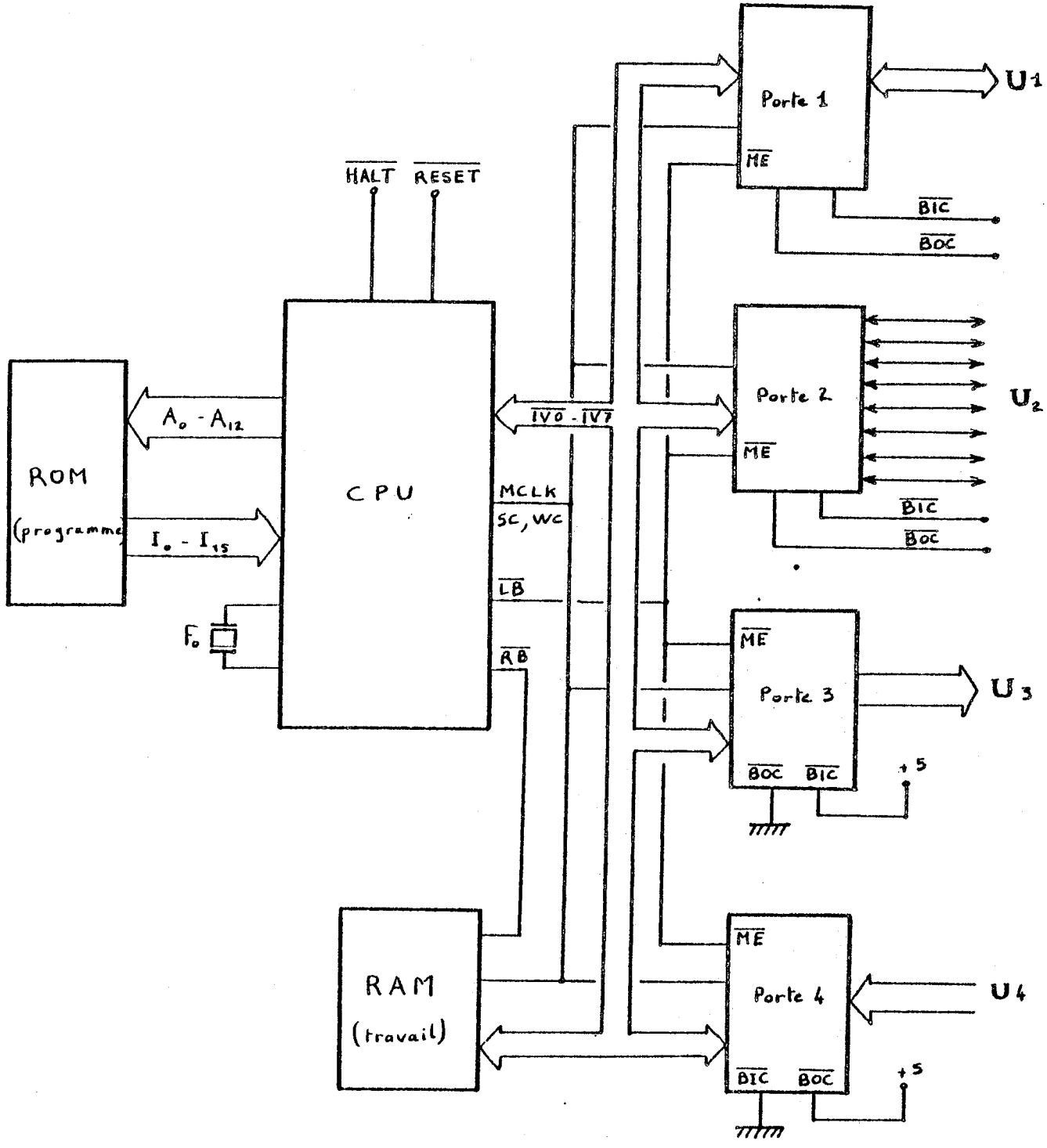


Fig.A2-2 Un contrôleur construit autour du 8X300

A2-3 Le jeu d'instructions du 8X300

a) Description :

Toutes les instructions sont de 16 bits et sont lues, décodées, et exécutées en seul cycle (250 ns avec une horloge de 8 MHz).

Le tableau fig.A2-3 regroupe les différentes instructions du 8X300 en liaison avec les formats décrits à la fig.A2-4.

nom	code	format	opération
MOVE	0	I et II	La donnée adressée par S et R/L remplace la donnée adressée par D et R/L
ADD	1		Même action que MOVE avec addition du contenu du registre AUX à la donnée source
AND	2		Même action, mais intersection logique au lieu d'addition
XOR	3		Même action, mais disjonction logique au lieu d'addition
XEC	4	III et IV	Exécution de l'instruction se trouvant à l'adresse obtenue par remplacement des 5 ou 8 bits poids faible du compteur ordinal, par la somme de I et de la donnée spécifiée par S et L
NZT	5		Exécution de l'instruction suivante si la donnée adressée par S (et L) est nulle. Sinon, exécution de l'instruction obtenue par remplacement des 5 ou 8 bits poids faible du compteur ordinal, par I
XMIT	6		La valeur immédiate I remplace la donnée se trouvant à l'adresse spécifiée par S et L
JMP	7	V	A remplace le contenu du compteur ordinal

Fig.A2-3 Les instructions du 8X300

b) Formats :

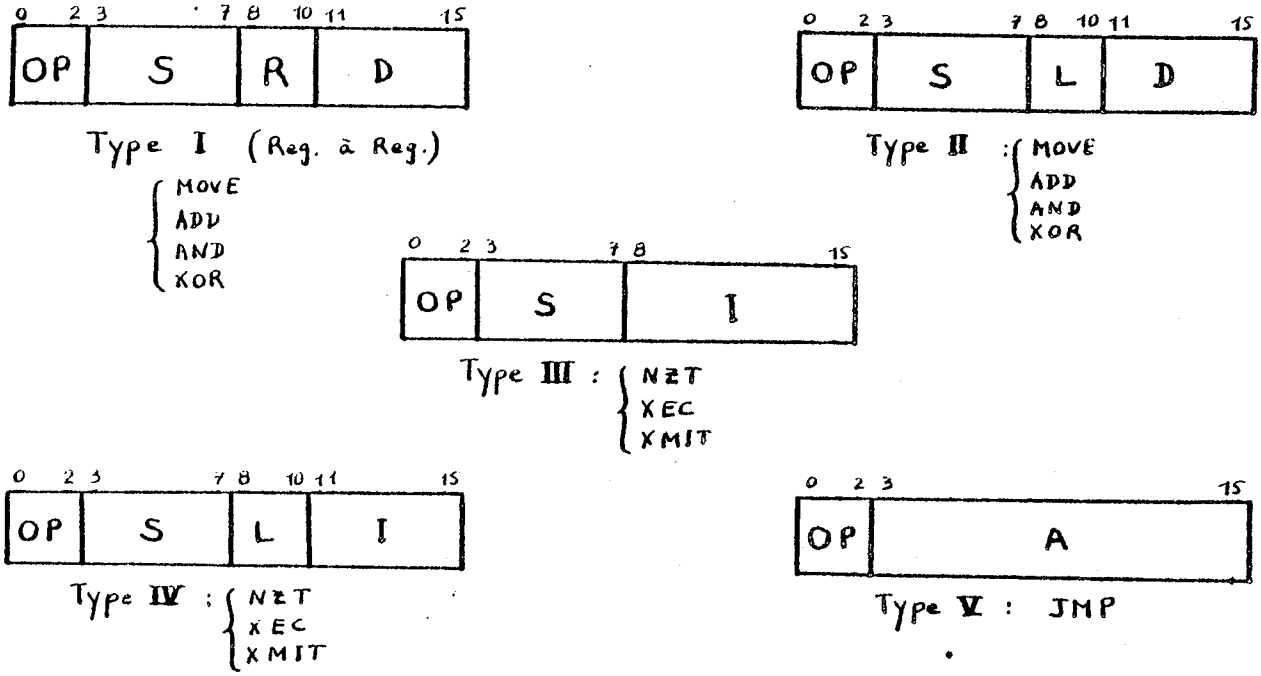


Fig.A2-4 Les 5 types de formats d'instruction

c) Signification des champs S, R/L, D :

Les champs "Source" (S) et "Destination" (D) représentent :

-1- des registres ayant les adresses suivantes (en octal) :

- 00 : registre auxiliaire (AUX)
- 01 à 06 : registres R1 à R6
- 07 : pseudo-registre IVL
- 10 : registre OVF (débordement)
- 11 : R11
- 17 : pseudo-registre IVR

Remarque : Les registres IVL et IVR ne peuvent être que des "Destinations" tandis que OVF ne peut être qu'une "Source".

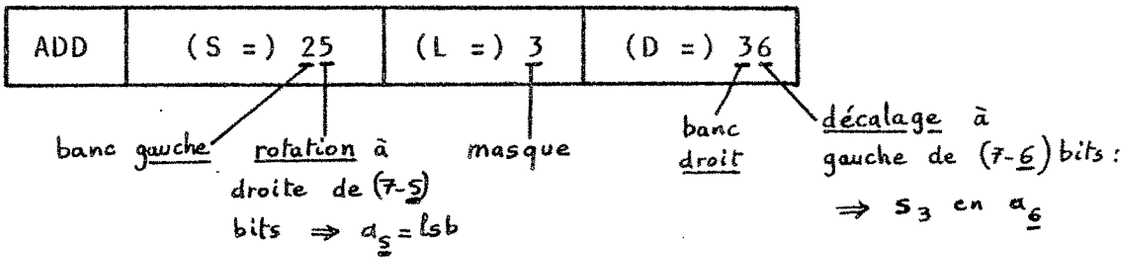
-2- des données sur le Bus-IV : les champs S et D spécifient alors le bit de poids faible de la donnée à l'intérieur des 8 bits du Bus.

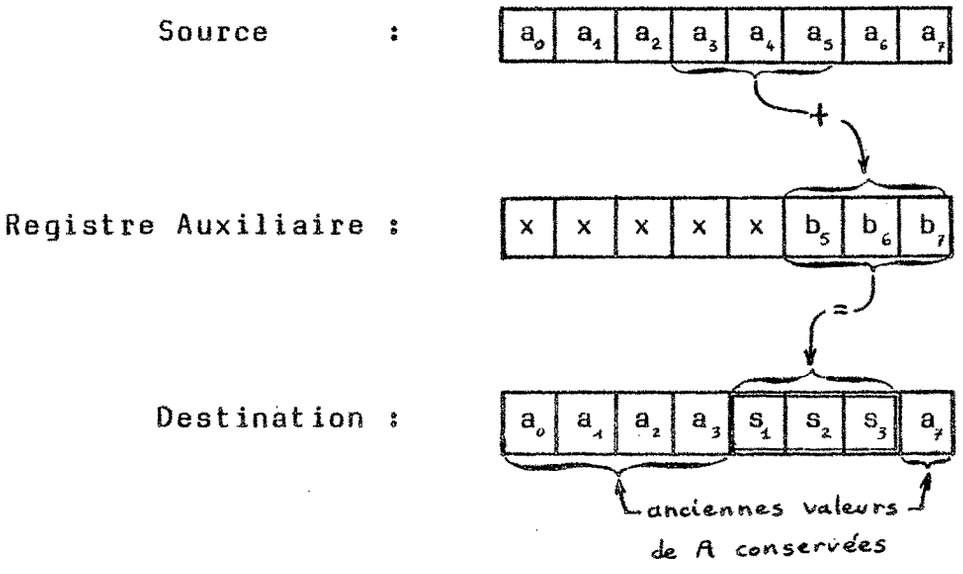
Les valeurs 20 à 27 (en octal) spécifient des données se trouvant sur le banc gauche, et les valeurs 30 à 37 des données sur le banc droit.

Si S et D sont des registres, R indique une rotation à droite (de 0 à 7 bits) de la donnée avant l'action spécifiée par l'instruction.

Si S ou/et D est une adresse sur l'IV-Bus, L indique un masquage de L bits de la donnée en entrée, ou la modification de seulement L bits de la donnée d'entrée ("merge").

Exemple : instruction permettant d'additionner 3 bits contenus dans le registre auxiliaire, aux bits 3, 4, 5 d'une donnée (A) placée dans un registre d'E/S du banc gauche, et de placer le résultat (S) dans un mot d'une mémoire adressée par le banc droit :





d) Le champ I :

Sa longueur est fonction de la Source : si S est un registre, I est un champ de 8 bits, sinon c'est un champ de 5 bits.

Ce champ contient soit une valeur chargée dans un registre ou sur le Bus-IV, soit une valeur servant à modifier le contenu du compteur ordinal. Dans ce cas on pourra avoir des sauts à l'intérieur de pages de 32 (I : 5 bits) ou 256 (I : 8 bits) instructions (mais il ne s'agit pas là d'un adressage relatif).

e) Le champ A :

D'une longueur de 13 bits, il place une nouvelle adresse dans le compteur ordinal, et permet des sauts de 8 K.instructions dans le programme.

ANNEXE 3

PRINCIPALES ABREVIATIONS UTILISEES DANS LES PLANCHES A - Q

Etats-unité envoyés par l'UCB :

ATTENT : Attention
BUSY : Occupé
CE : "Channel End" (fin sur canal)
DE : "Device End" (fin sur unité)
UE : "Unit Exception" (exception sur unité)
UCK : "Unit Check" (erreur sur unité)

Bits de l'octet d'analyse :

BUSOCHK : "Bus-Out Check" (erreur de parité sur Bus de données)
COMMRJCT : "Command Reject" (rejet de la commande)
OVERRUN : l'UCB a reçu un Halt I/O
READEN : lecture possible
WRITEN : écriture possible

Divers :

VSOS : commande de propagation de SLOS

Remarque : les significations des Tags-in et des Tags-out sont données dans la liste générale des abréviations, au début de l'ouvrage.

ORGANIGRAMMES DU LOGICIEL INICAN

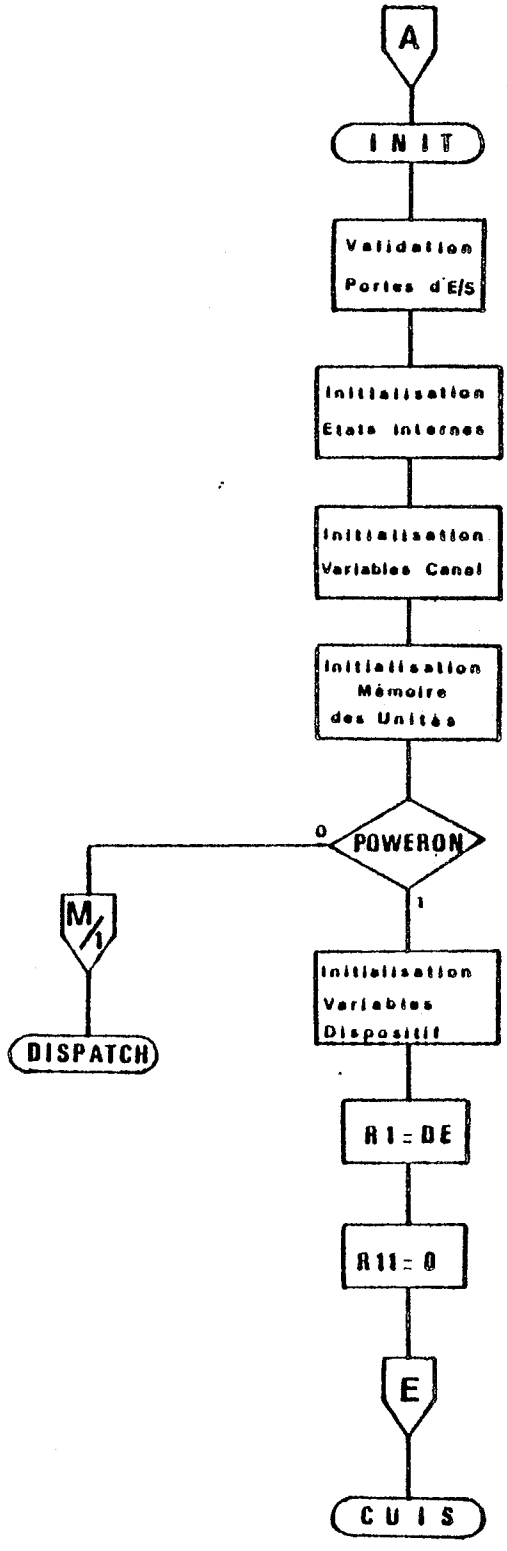


Planche A : Séquence d'initialisations

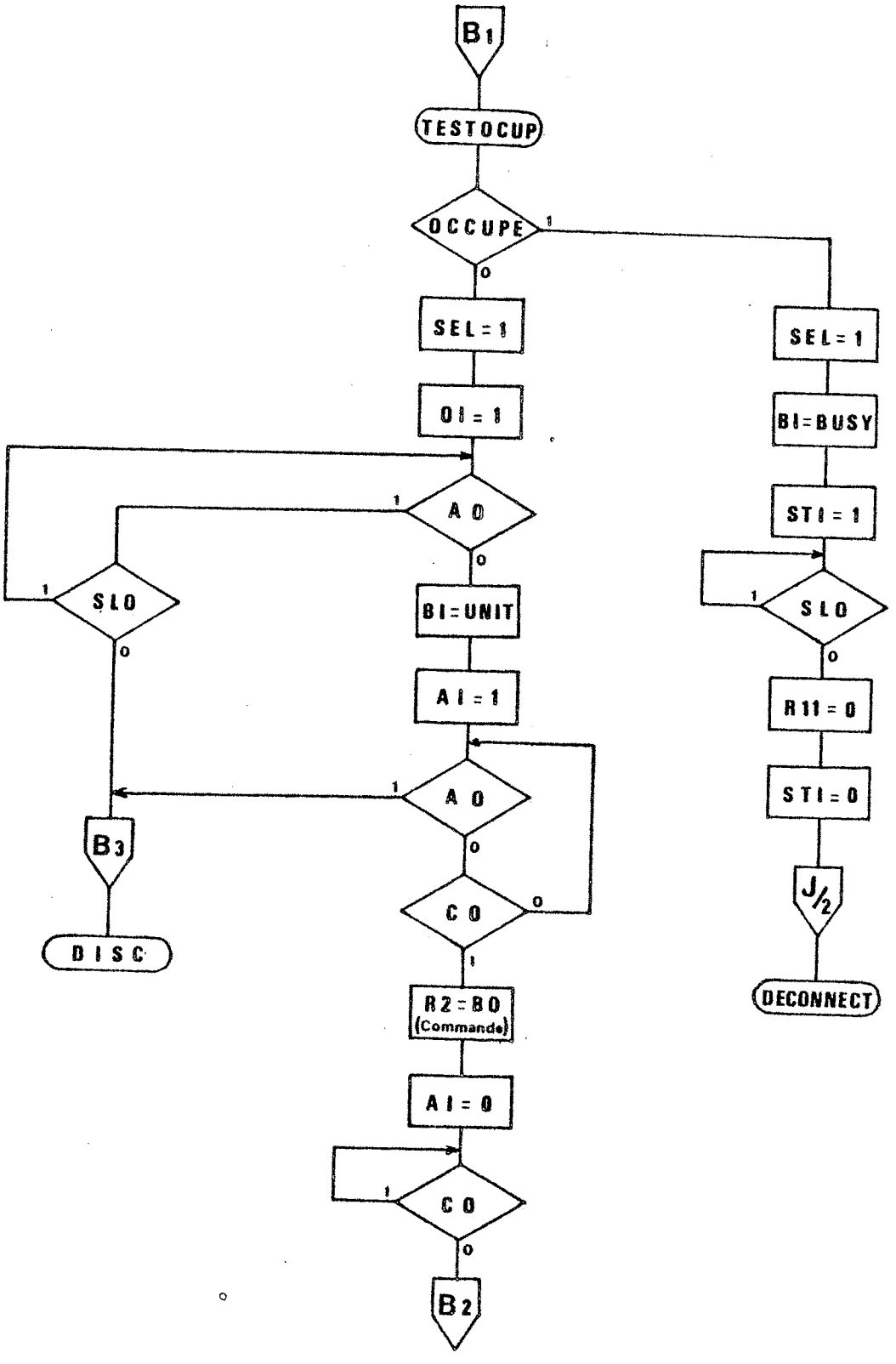


Planche B1 : Réponses à une sélection par le canal

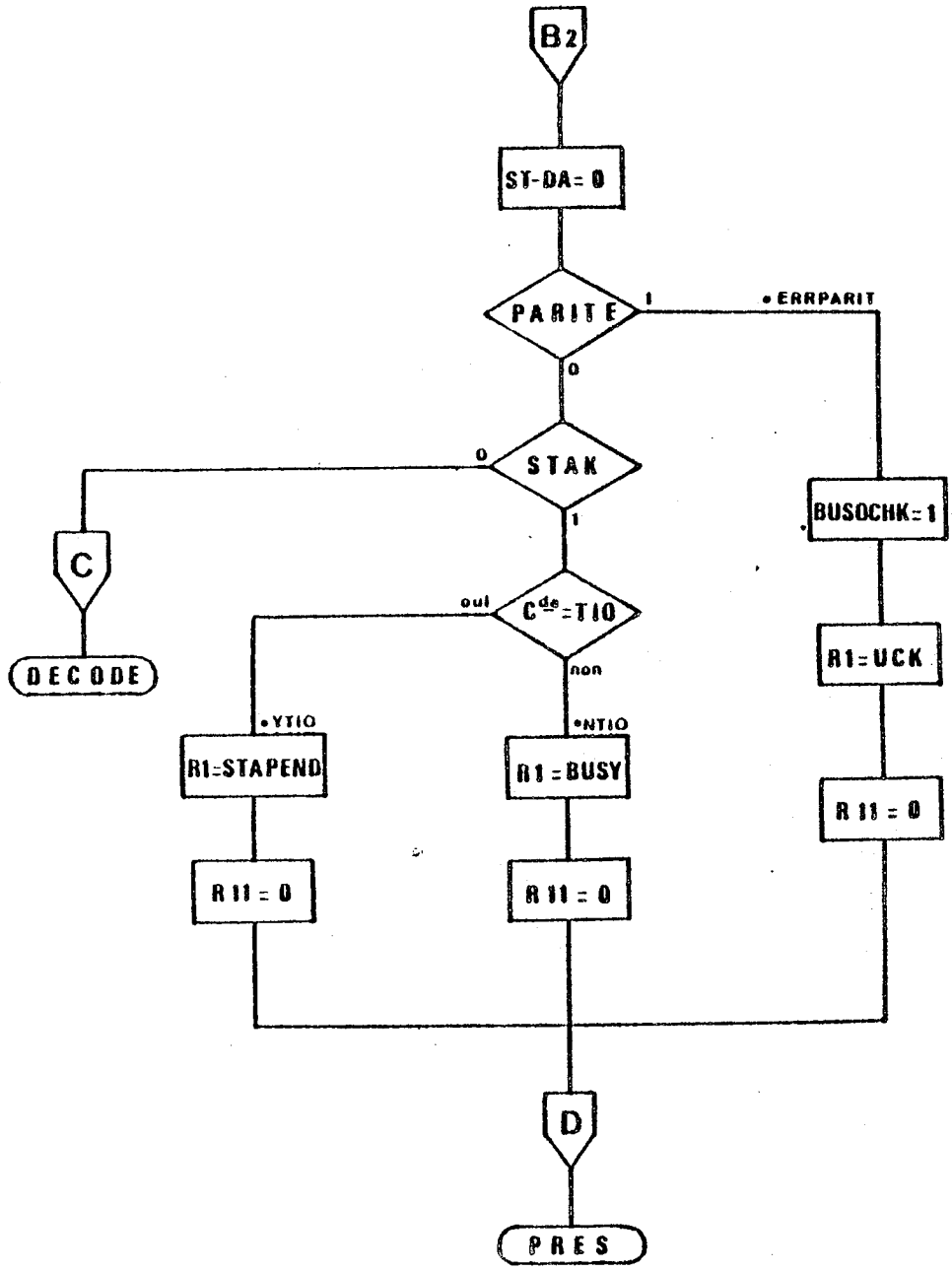


Planche B2 : Suites de la CISS : - DECODE : commande acceptée
- ERRPARIT : erreur de parité dans la commande reçue
- YTIO et NTIO : réponses dans le cas d'un état-unité en attente

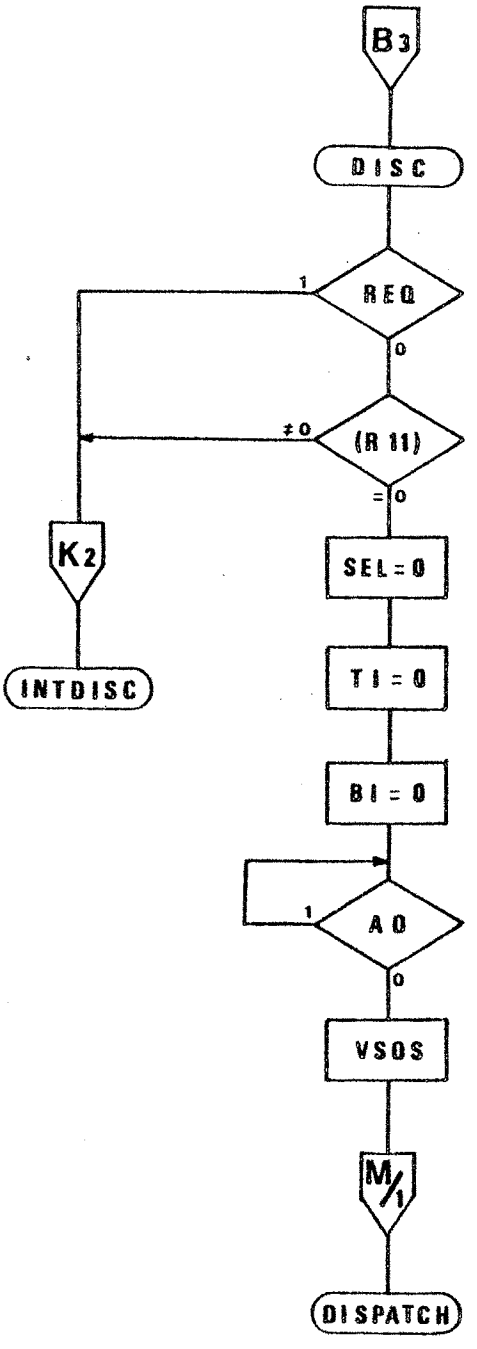


Planche B3 : Réponse à "Interface Disconnect" pendant la CISS

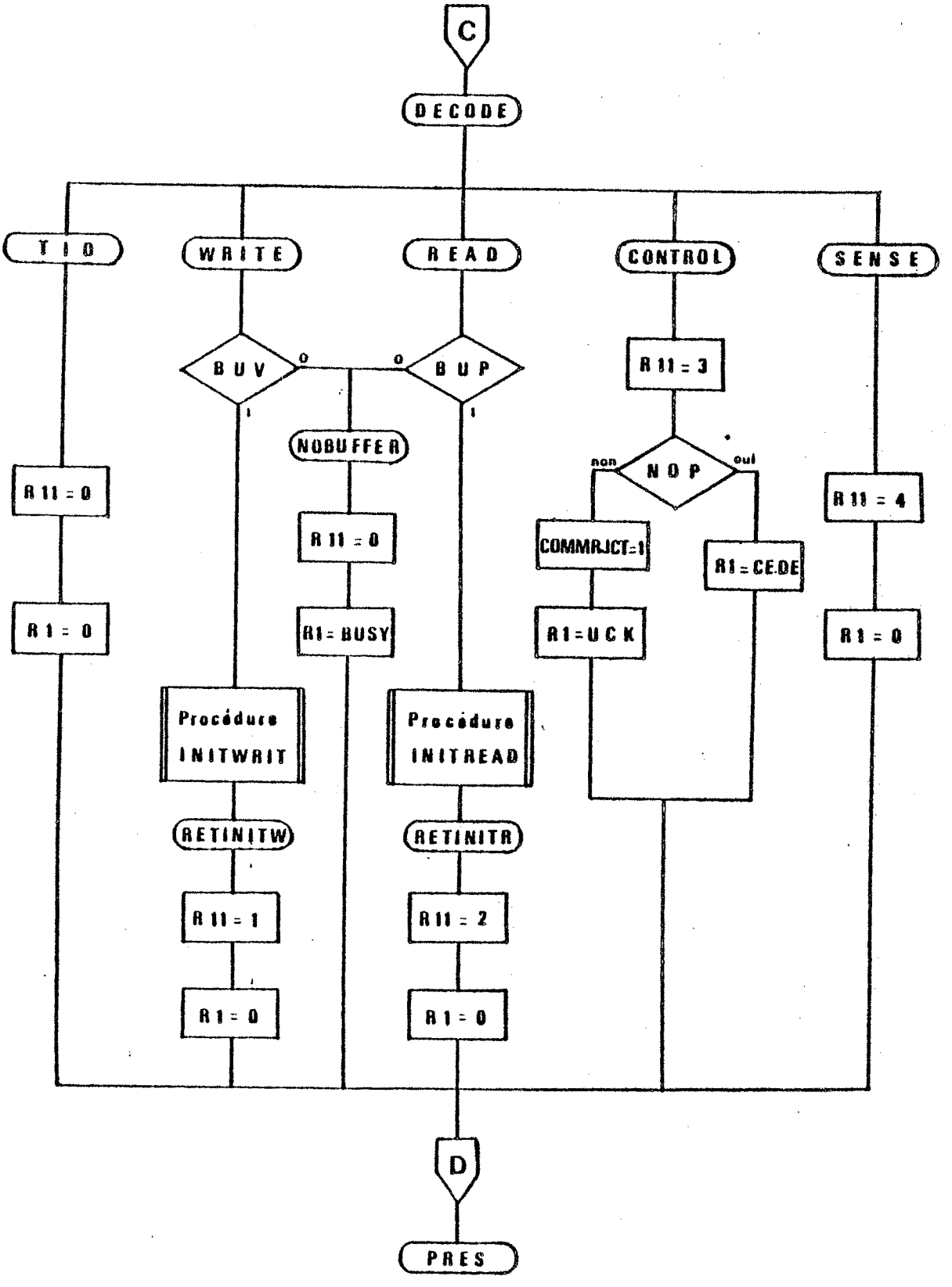


Planche C : Décodage de la commande

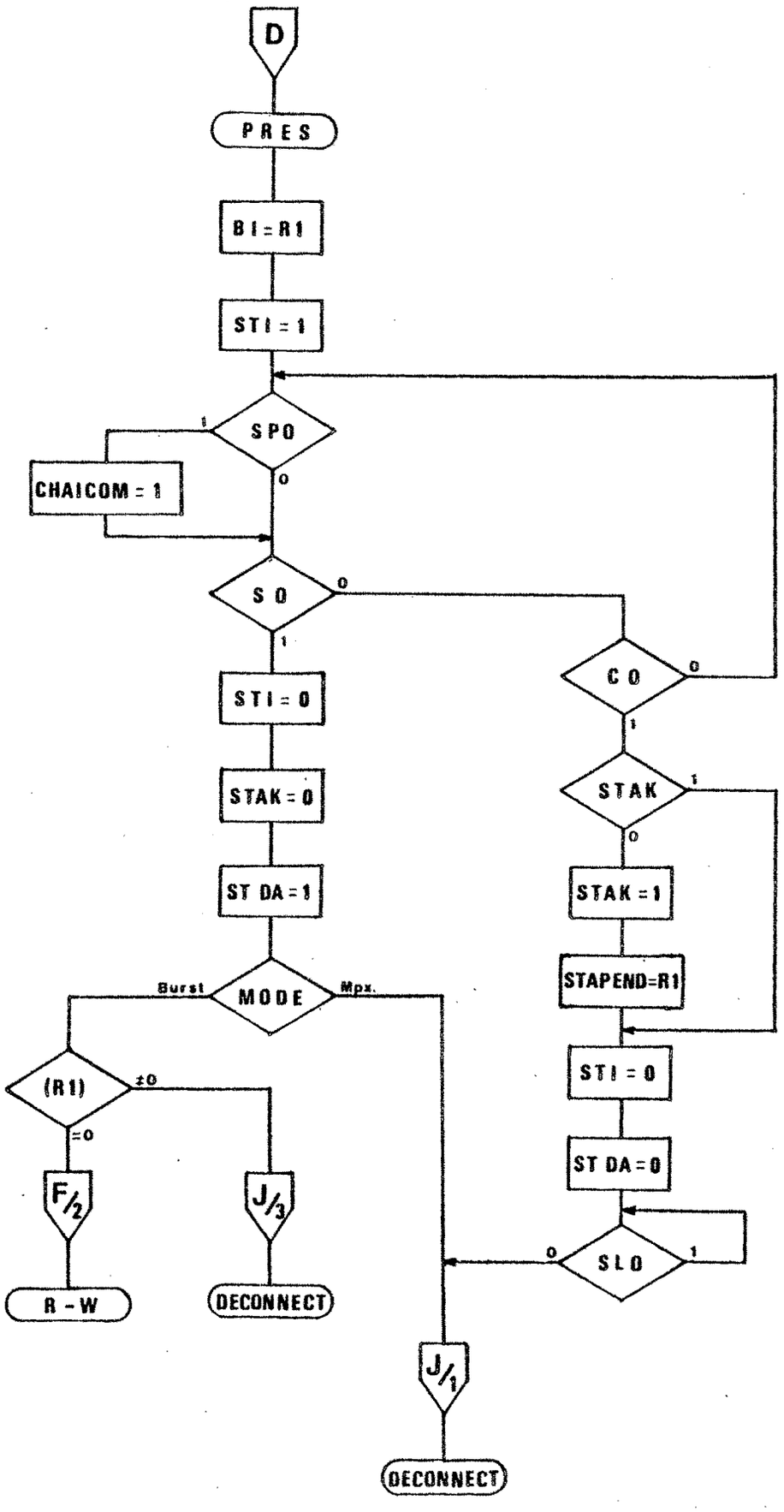


Planche D : Présentation de l'état-unité

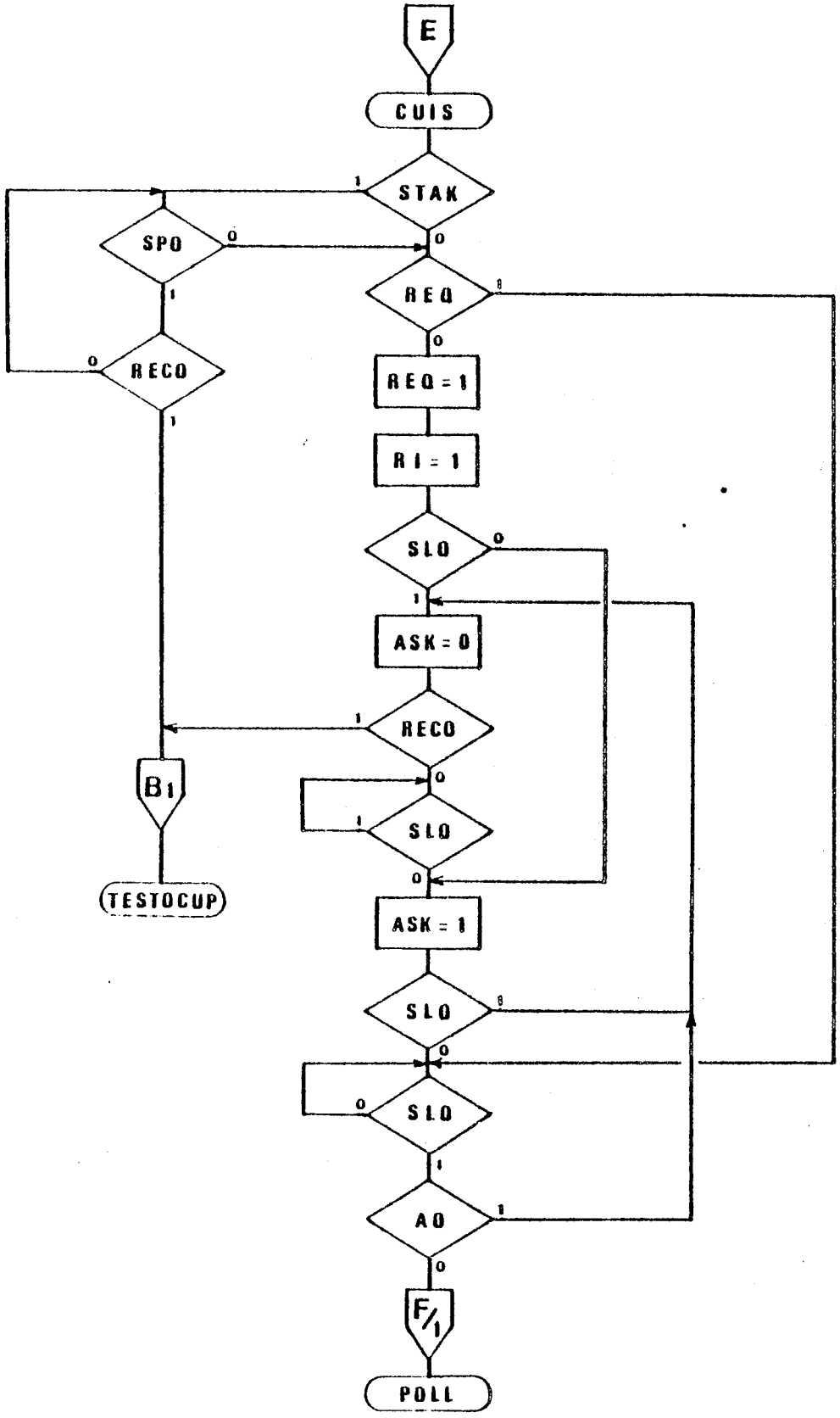


Planche E : Demande de sélection par l'unité
("Control-Unit-Initiated-Sequence")

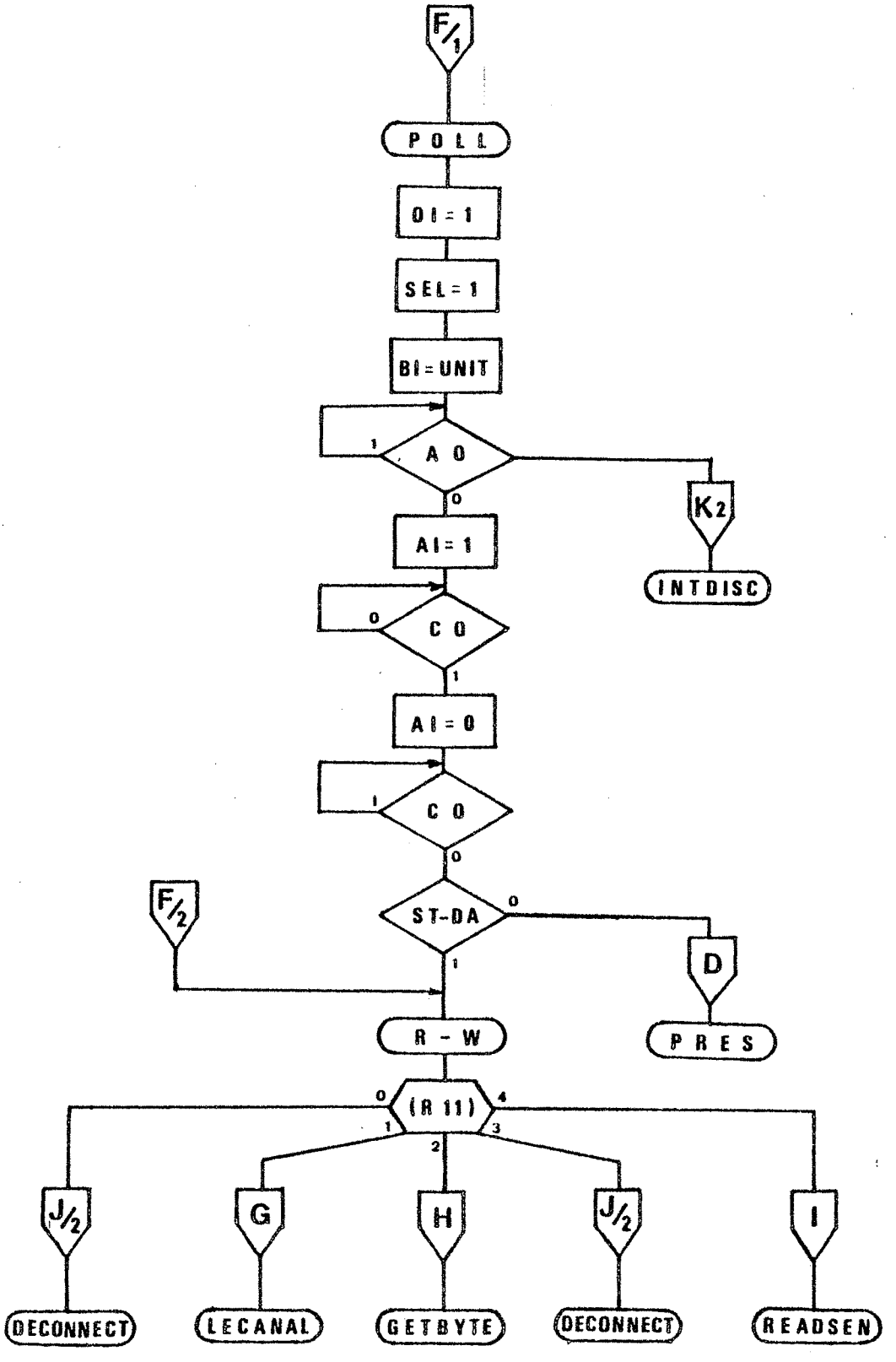


Planche F : Connexion après "Polling" du canal

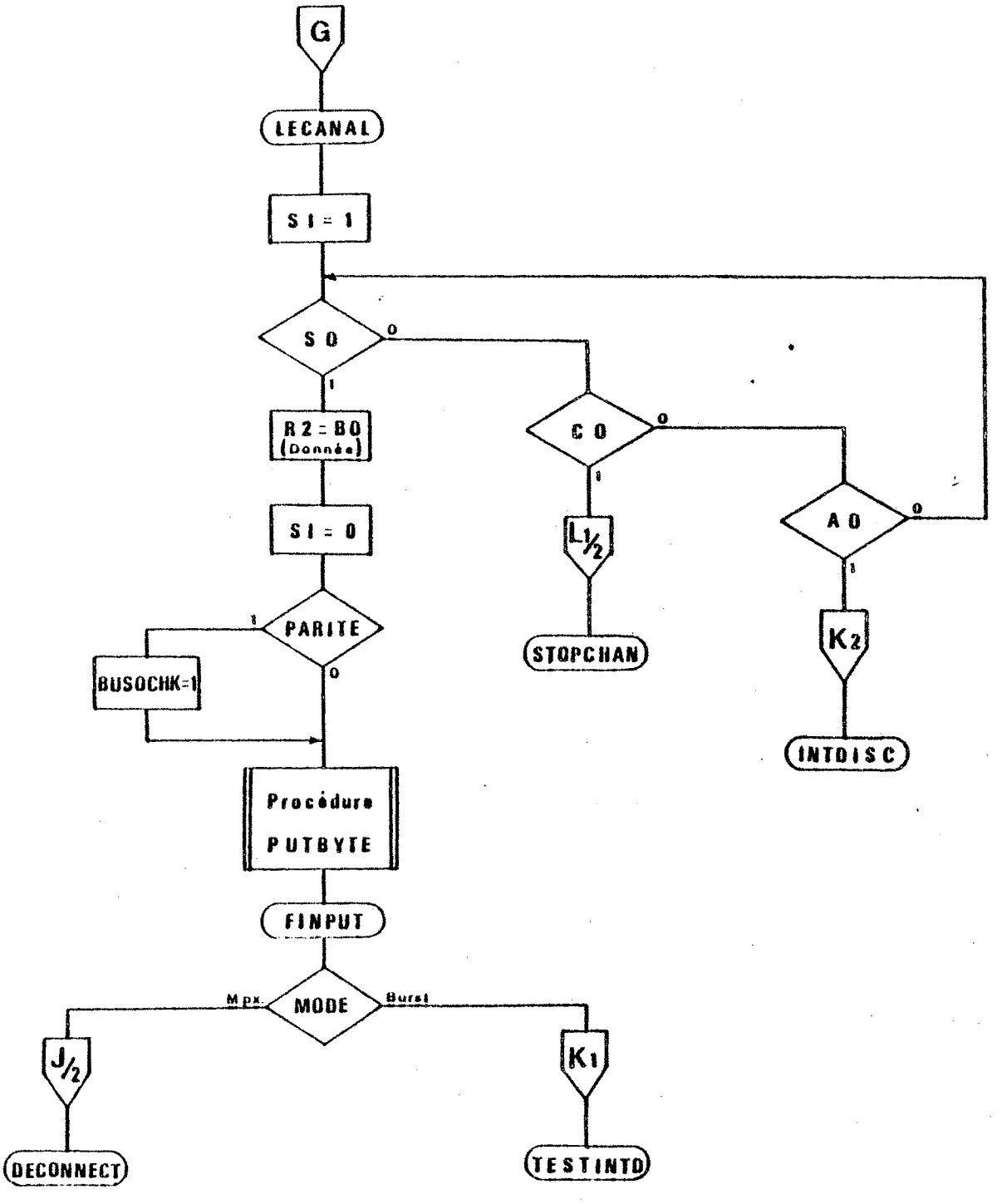


Planche G : Réponse à une commande d'écriture
(réception de l'octet)

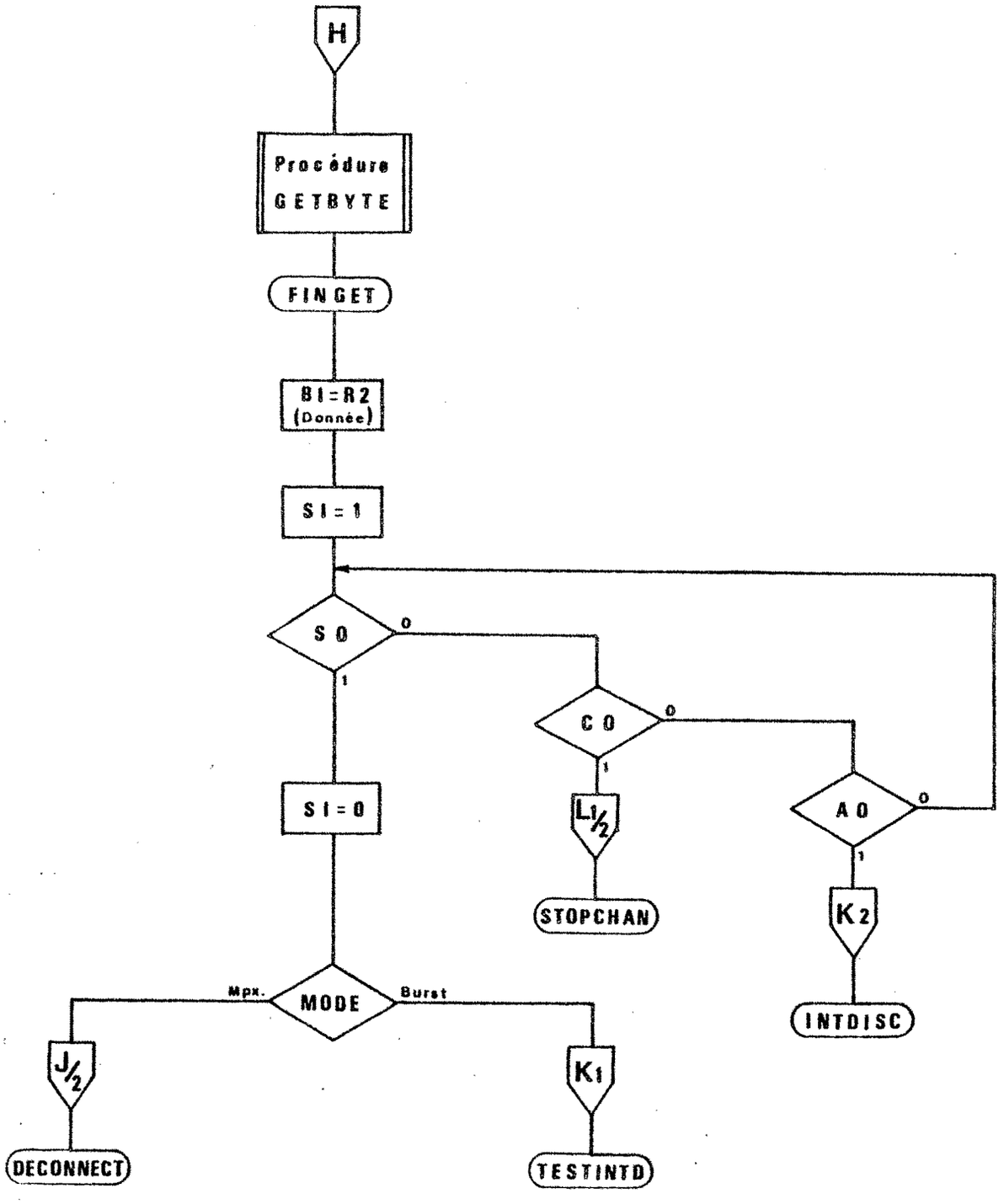


Planche H : Réponse à une commande de lecture
(envoi de l'octet)

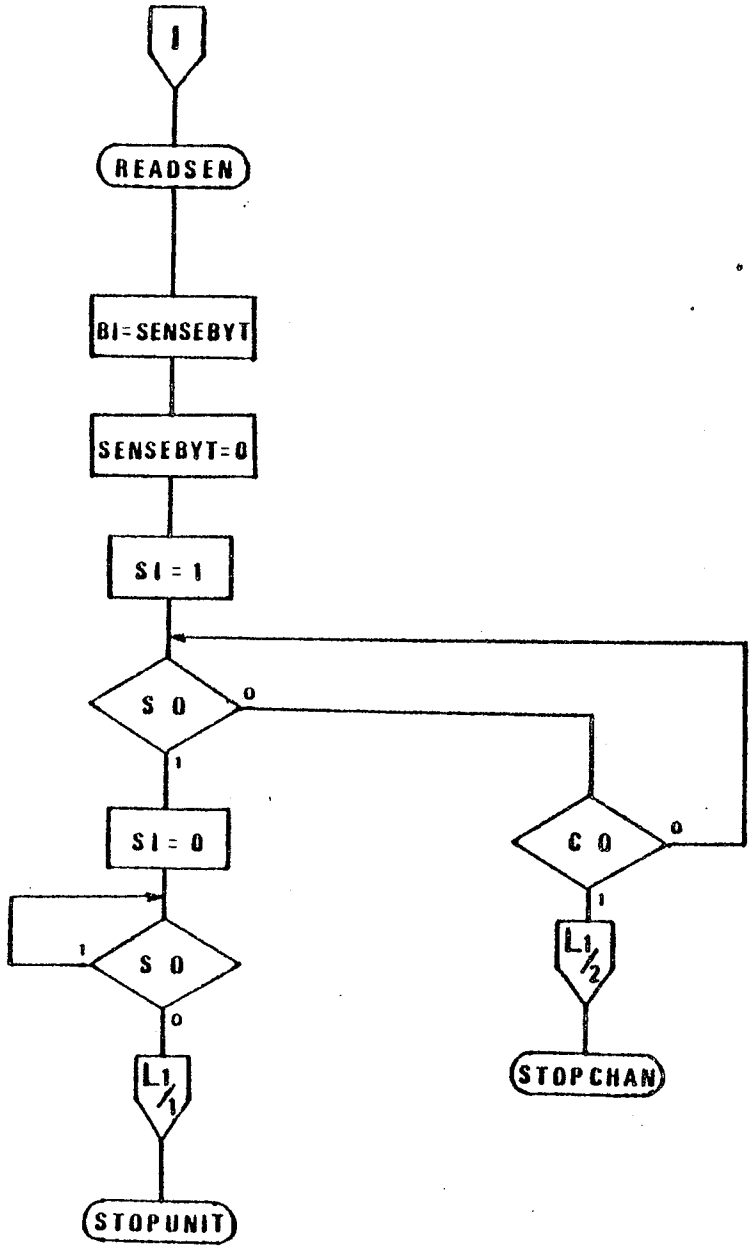


Planche I : Réponse à une commande d'analyse

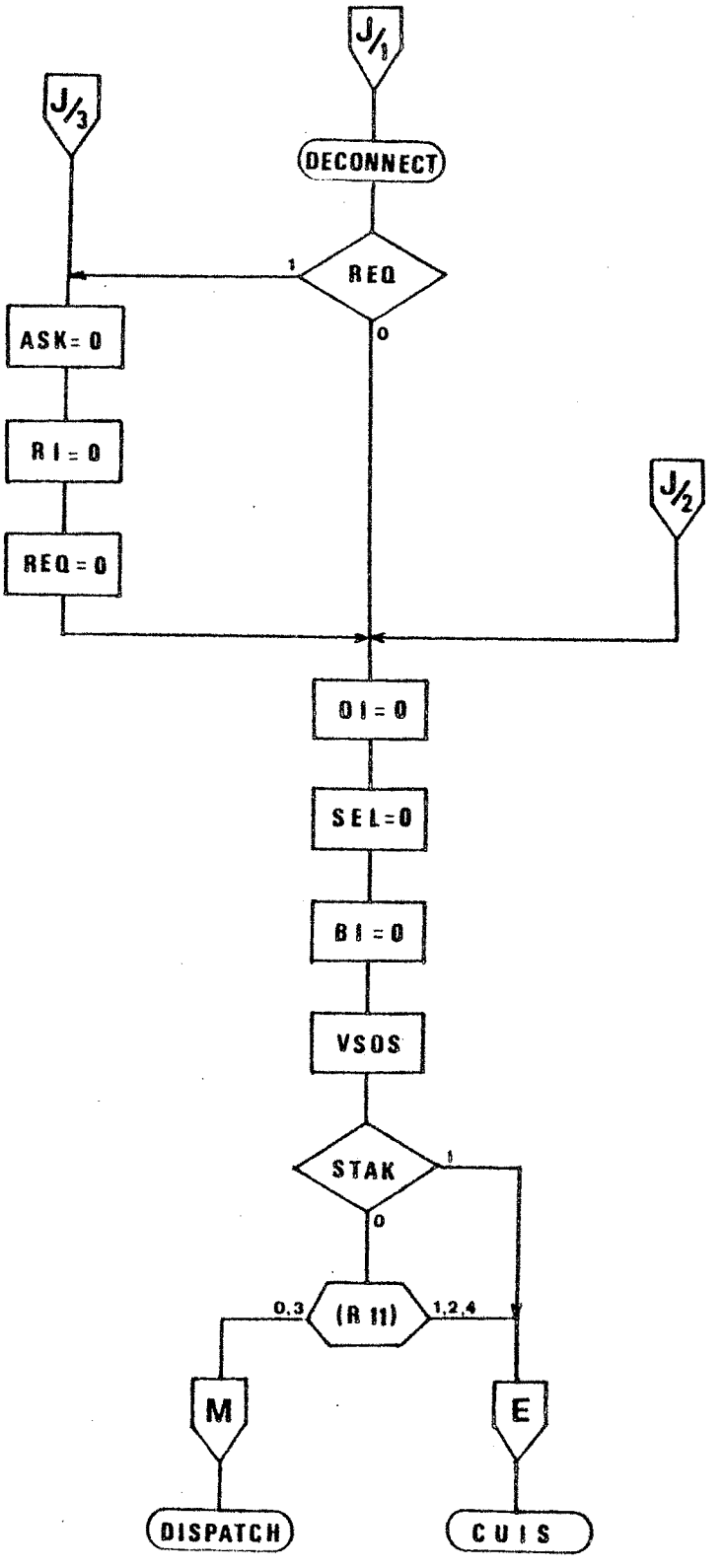


Planche J : Séquence de déconnexion du canal

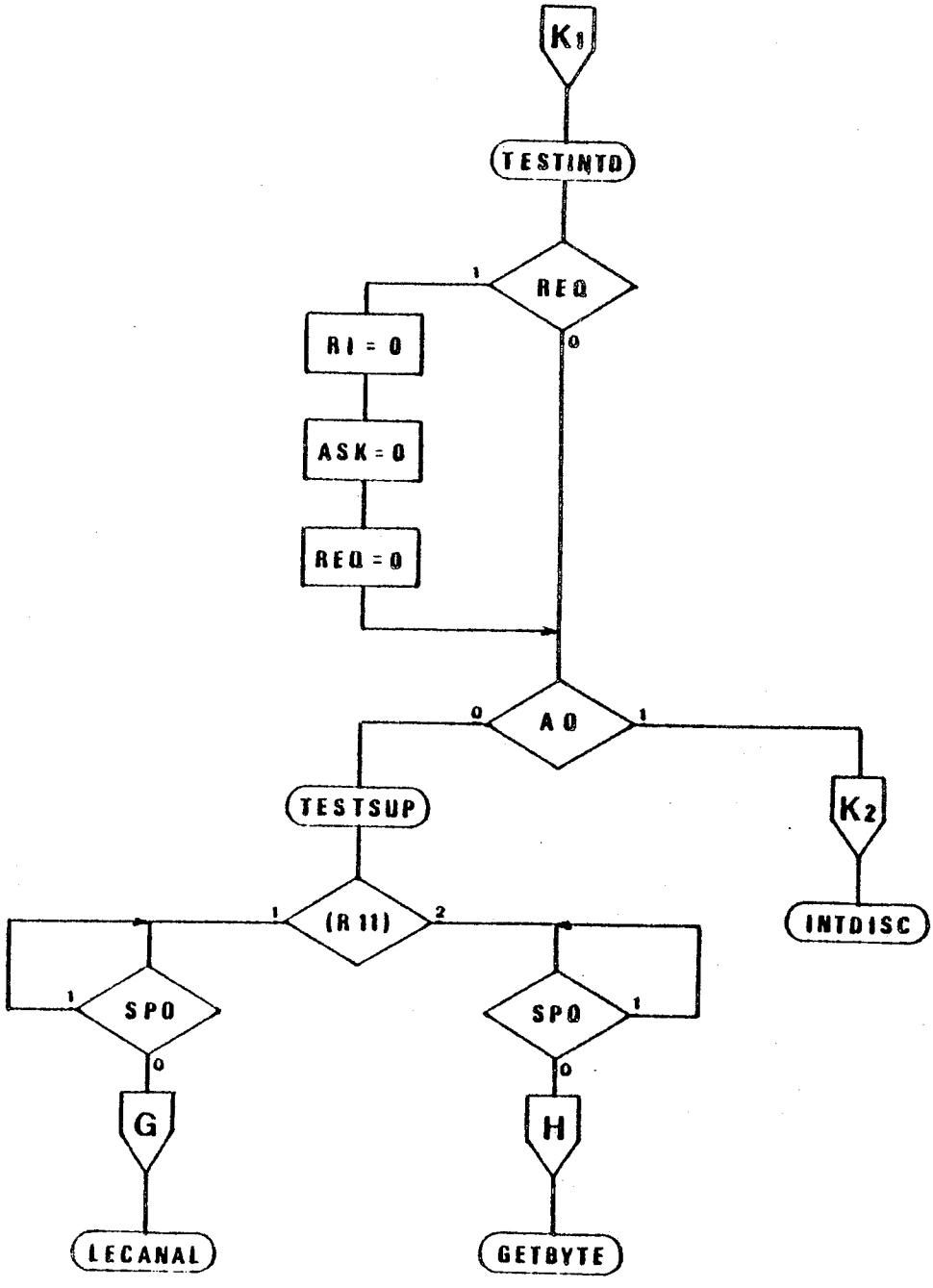


Planche K1 : Tests d'"Interface Disconnect" et de suppression des données pendant un transfert

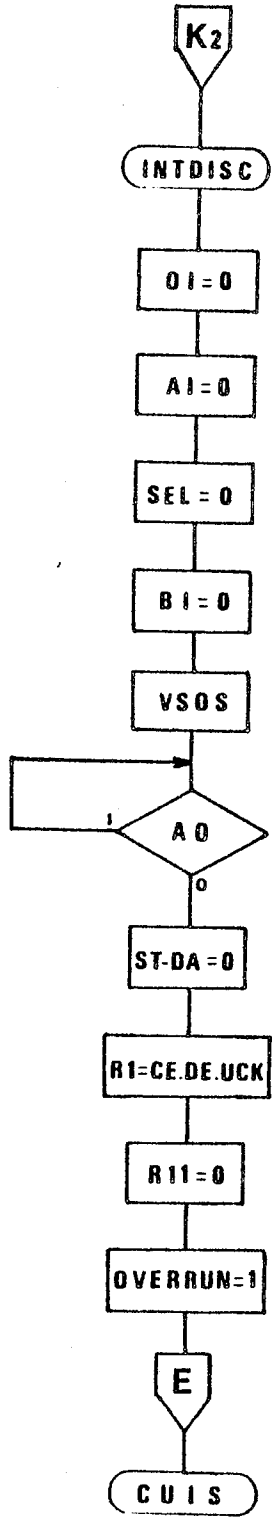


Planche K2 : Réponse à "Interface Disconnect"

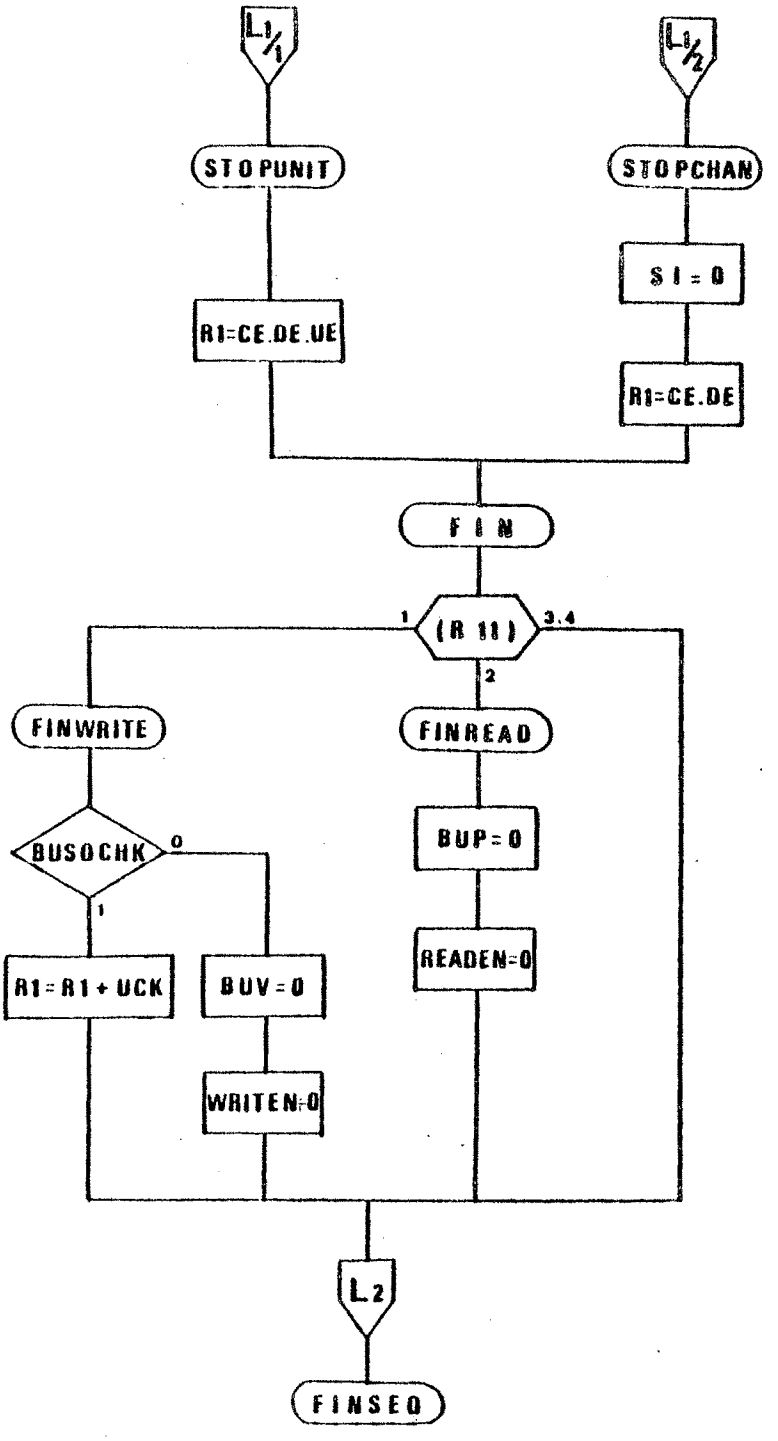


Planche L1 : Séquence de fin de transfert (première partie)

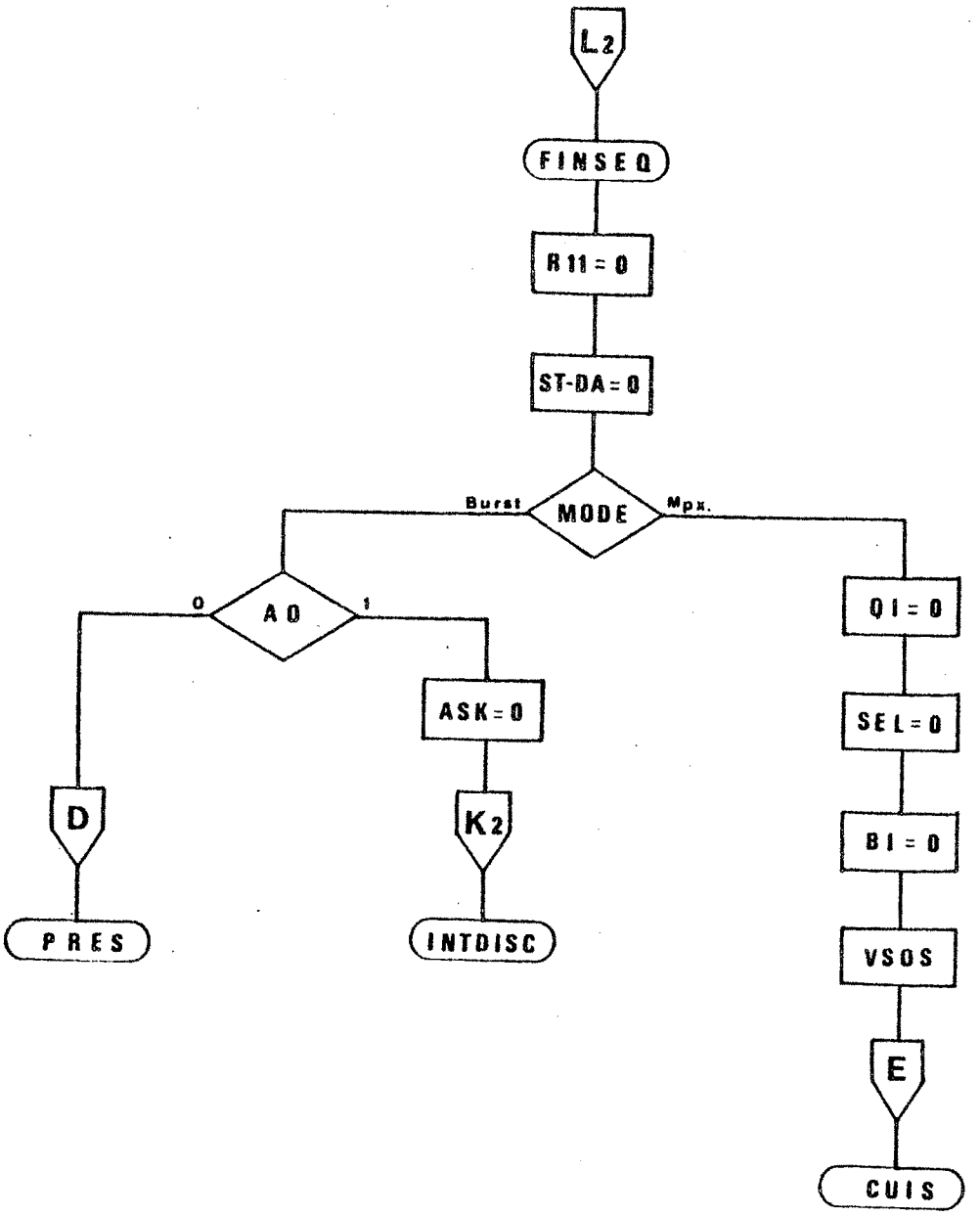


Planche L2 : Fin de transfert (deuxième partie)

ORGANIGRAMMES DU LOGICIEL INTDISP

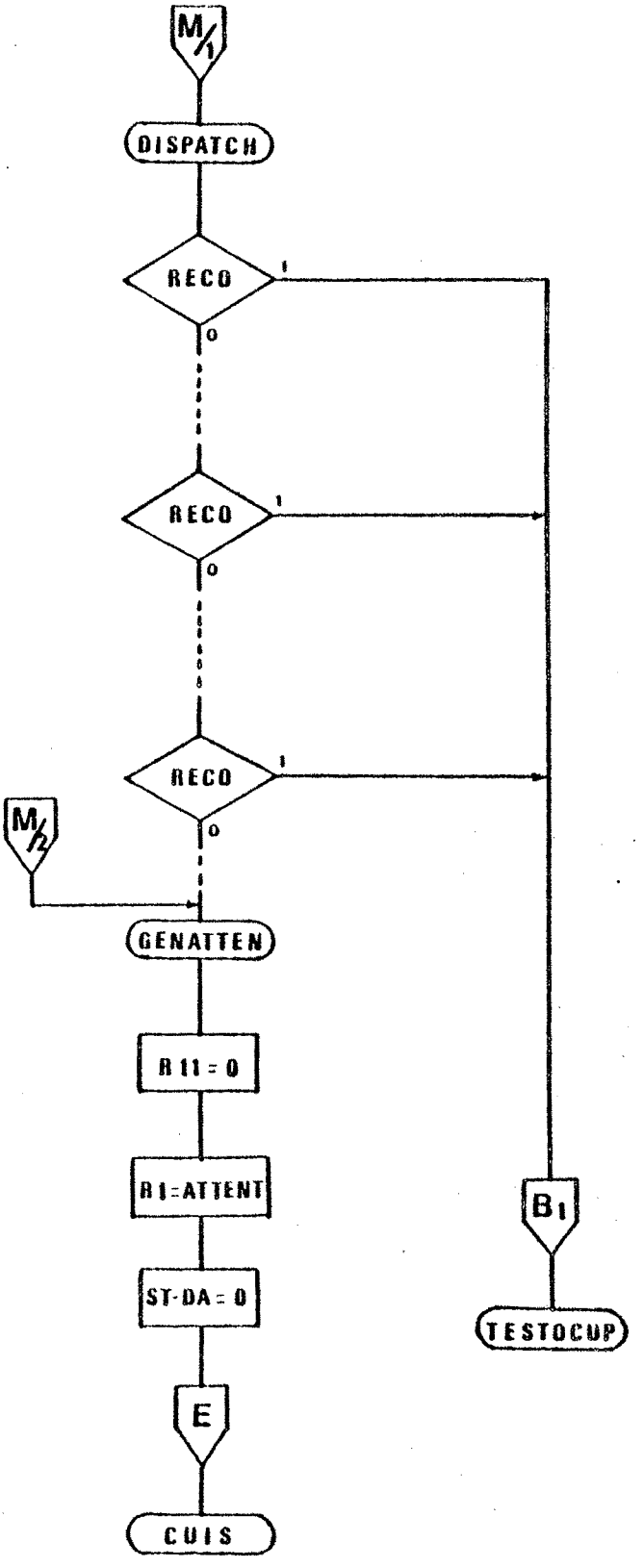
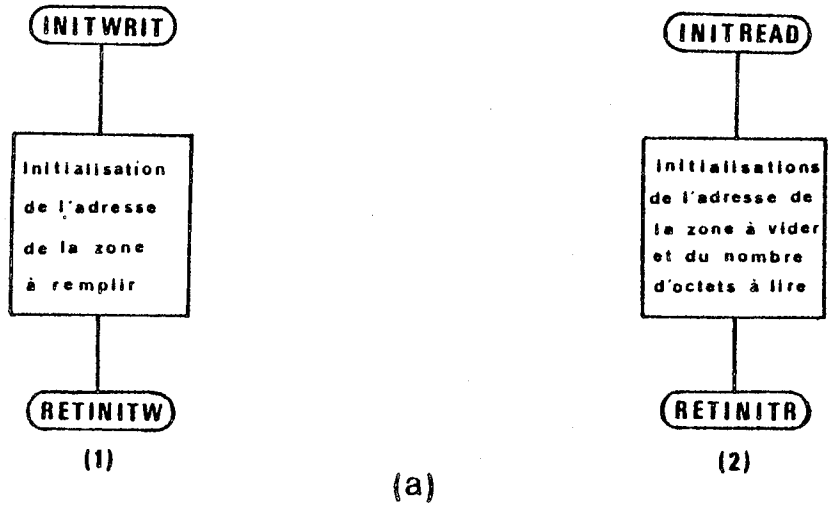
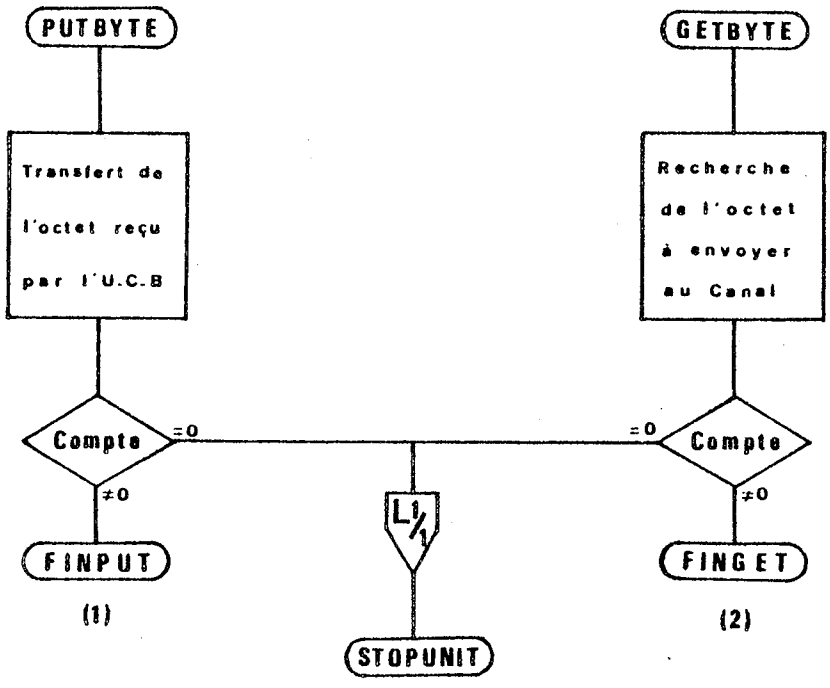


Planche M : Organisation générale du logiciel INTDISP



(a)



(b)

Planche N : Sous-programmes de INTDISP appelés par INTCAN pour effectuer les transferts de données

(a) Initialisation des transferts
(b) Transferts effectifs
(1) Cas d'une écriture
(2) Cas d'une lecture

ORGANIGRAMMES DU LOGICIEL INTMEM

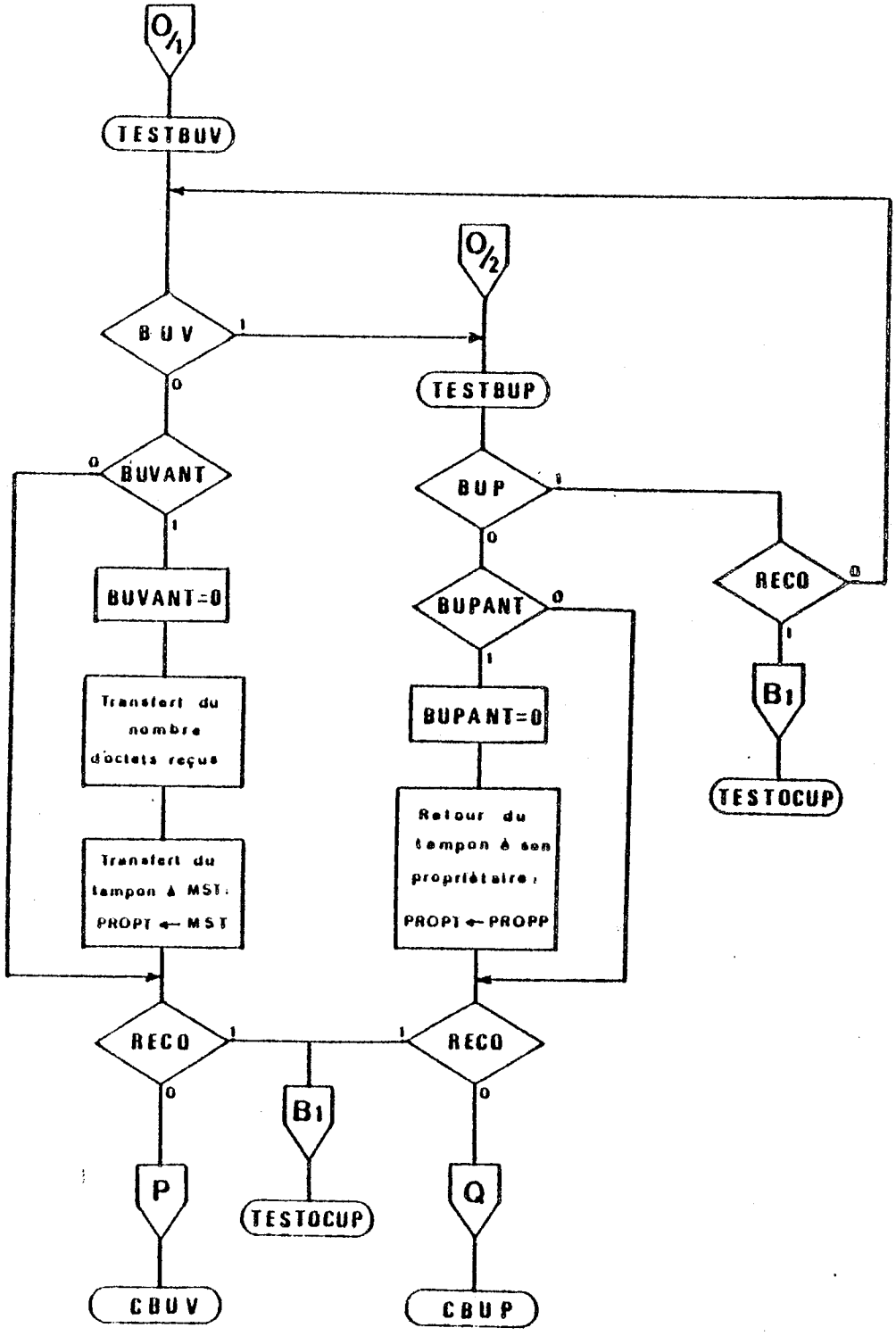


Planche 0 : Séquences de mise à jour après une lecture ou une écriture

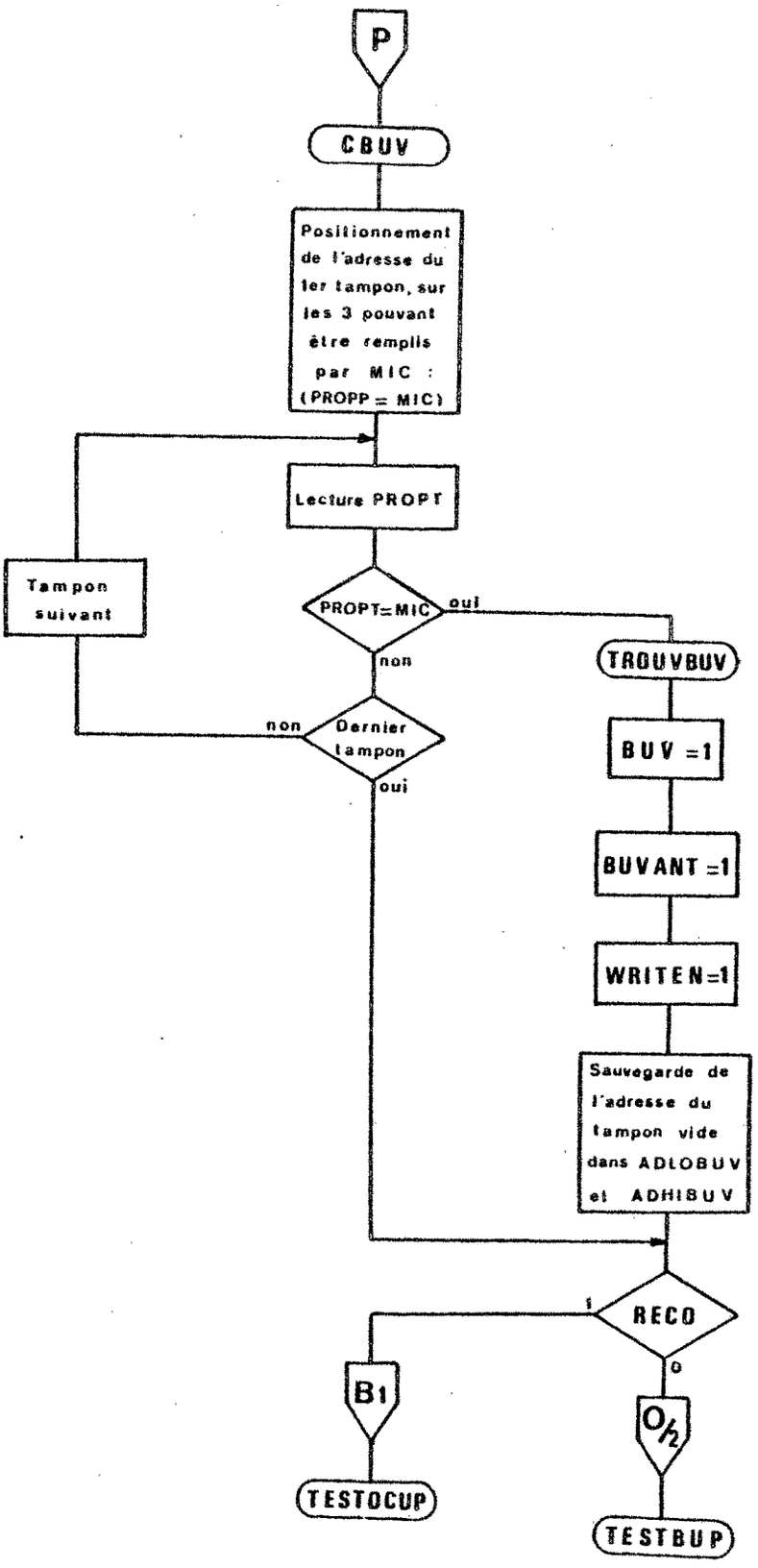


Planche P : Séquence de recherche d'un tampon vide

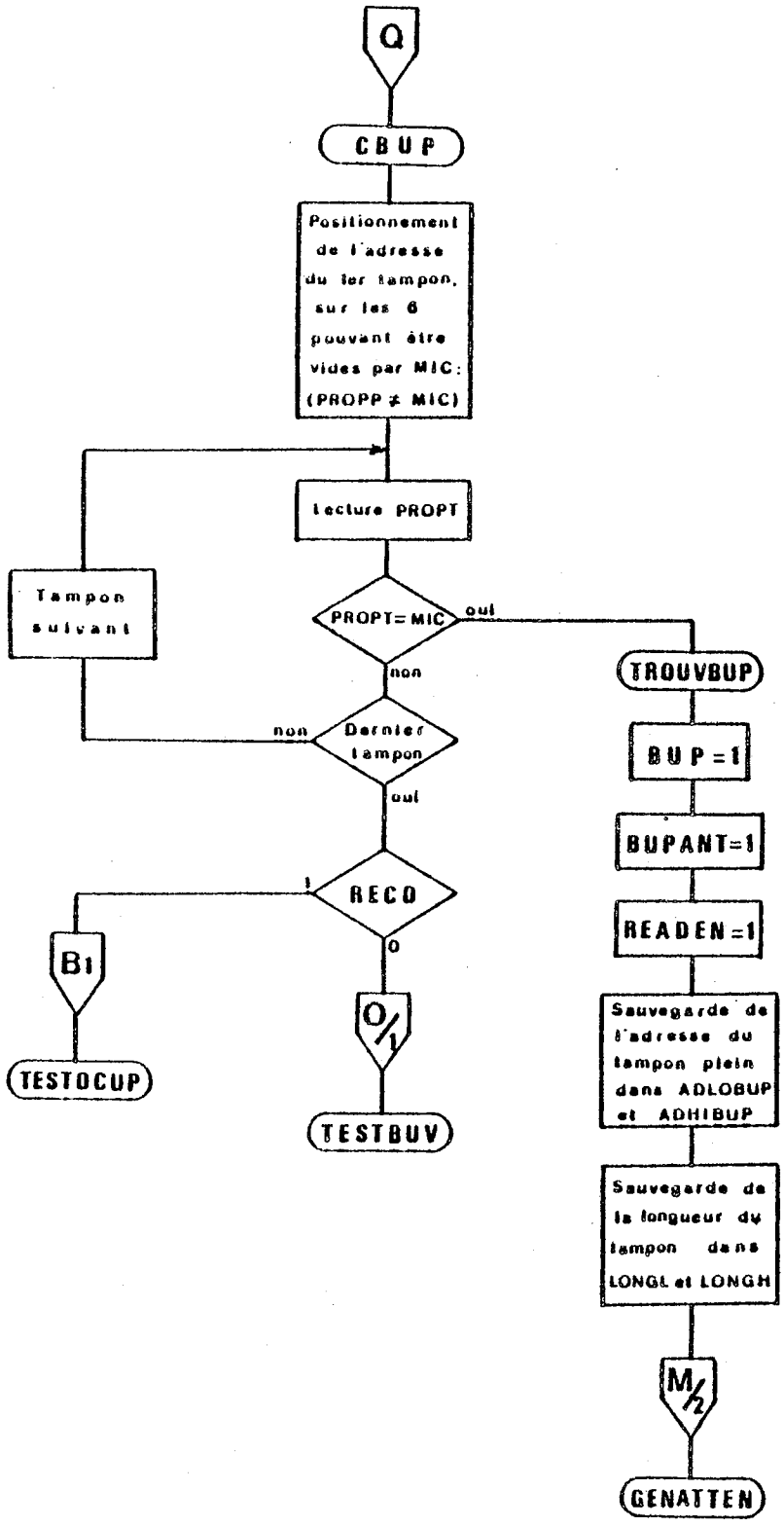


Planche Q : Séquence de recherche d'un tampon plein

SCHEMAS DE L'UCB

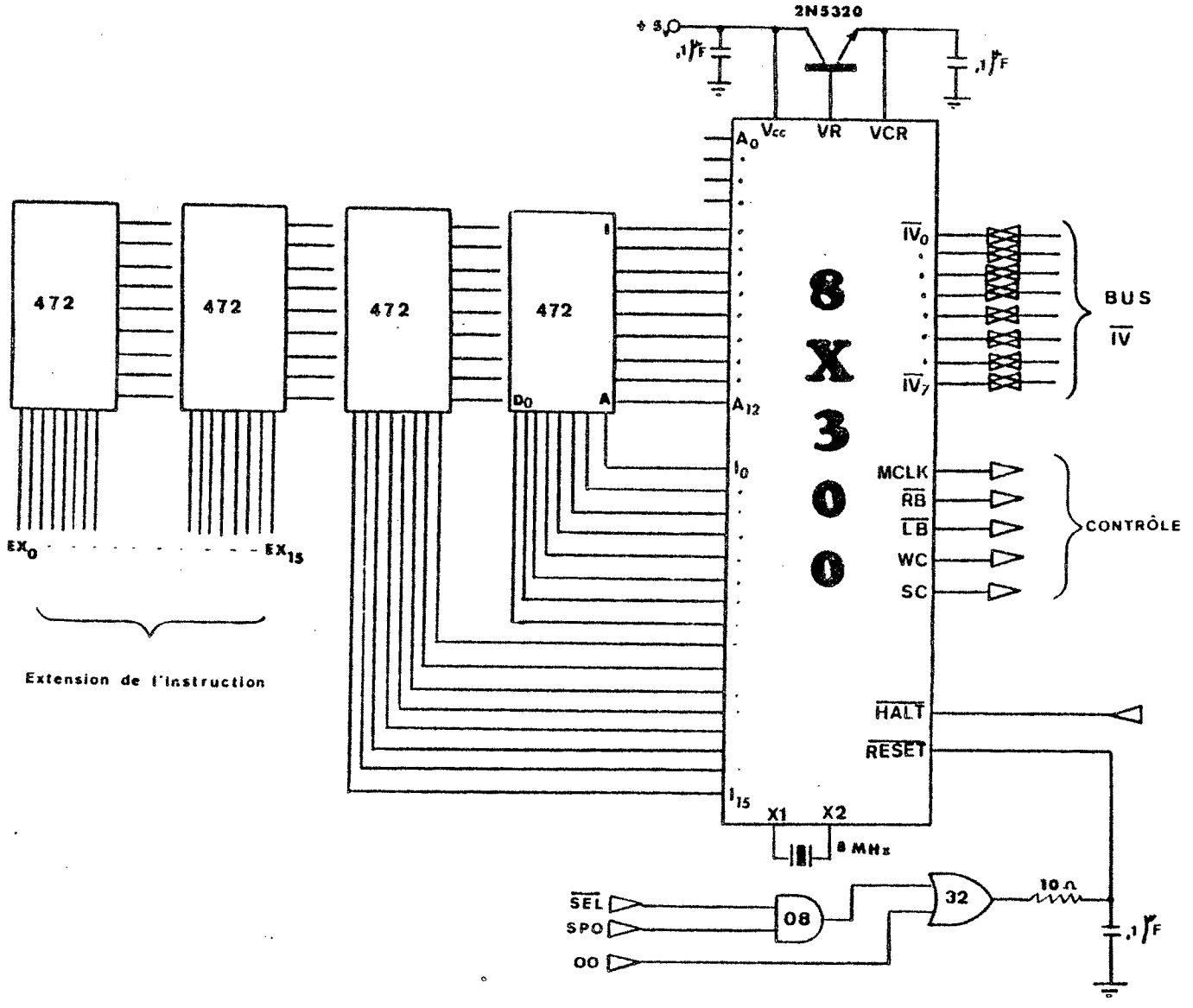


Planche 1 : Le CPU et sa mémoire de programme

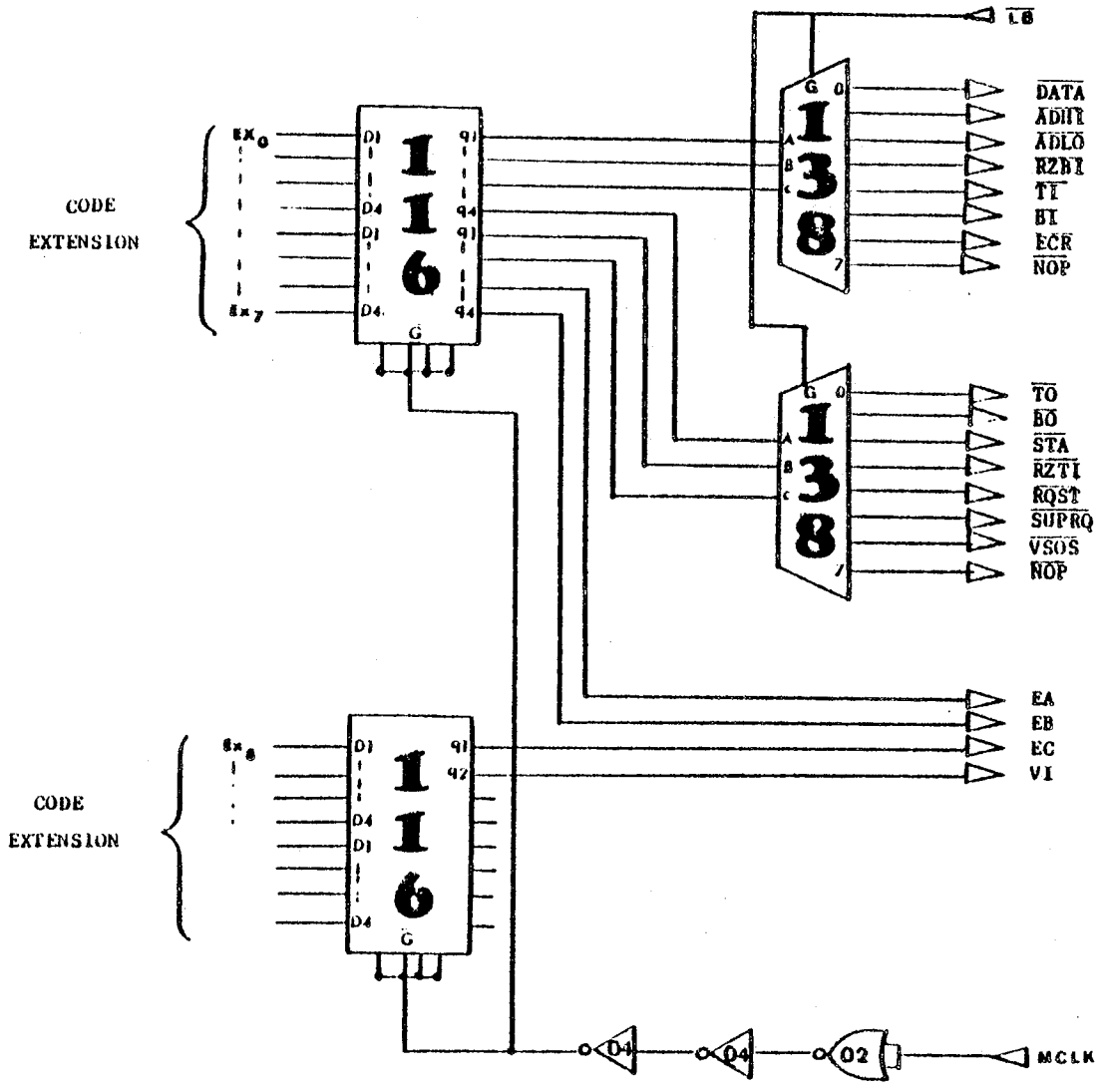


Planche 2 : Décodage de l'extension

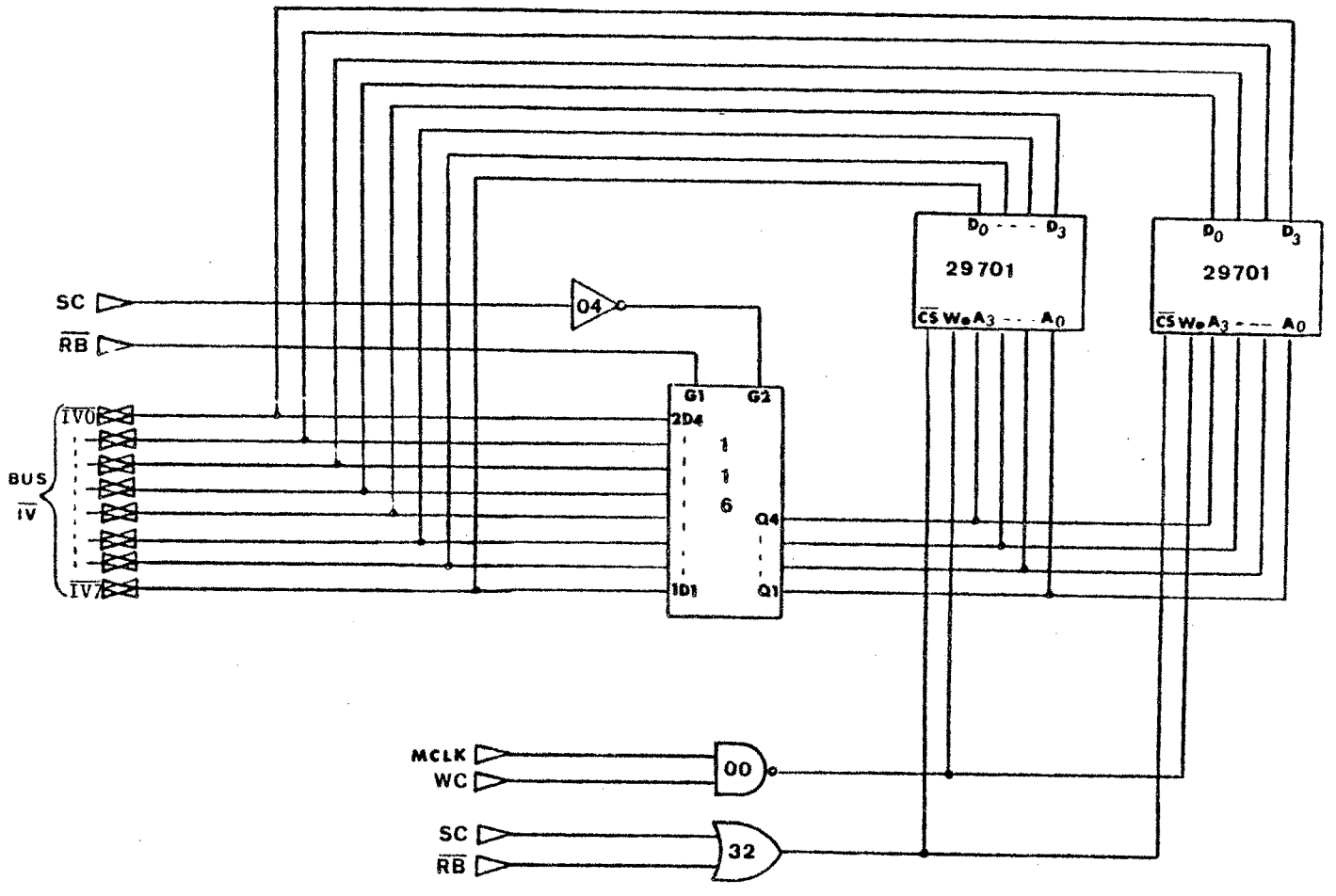


Planche 3 : La mémoire de travail

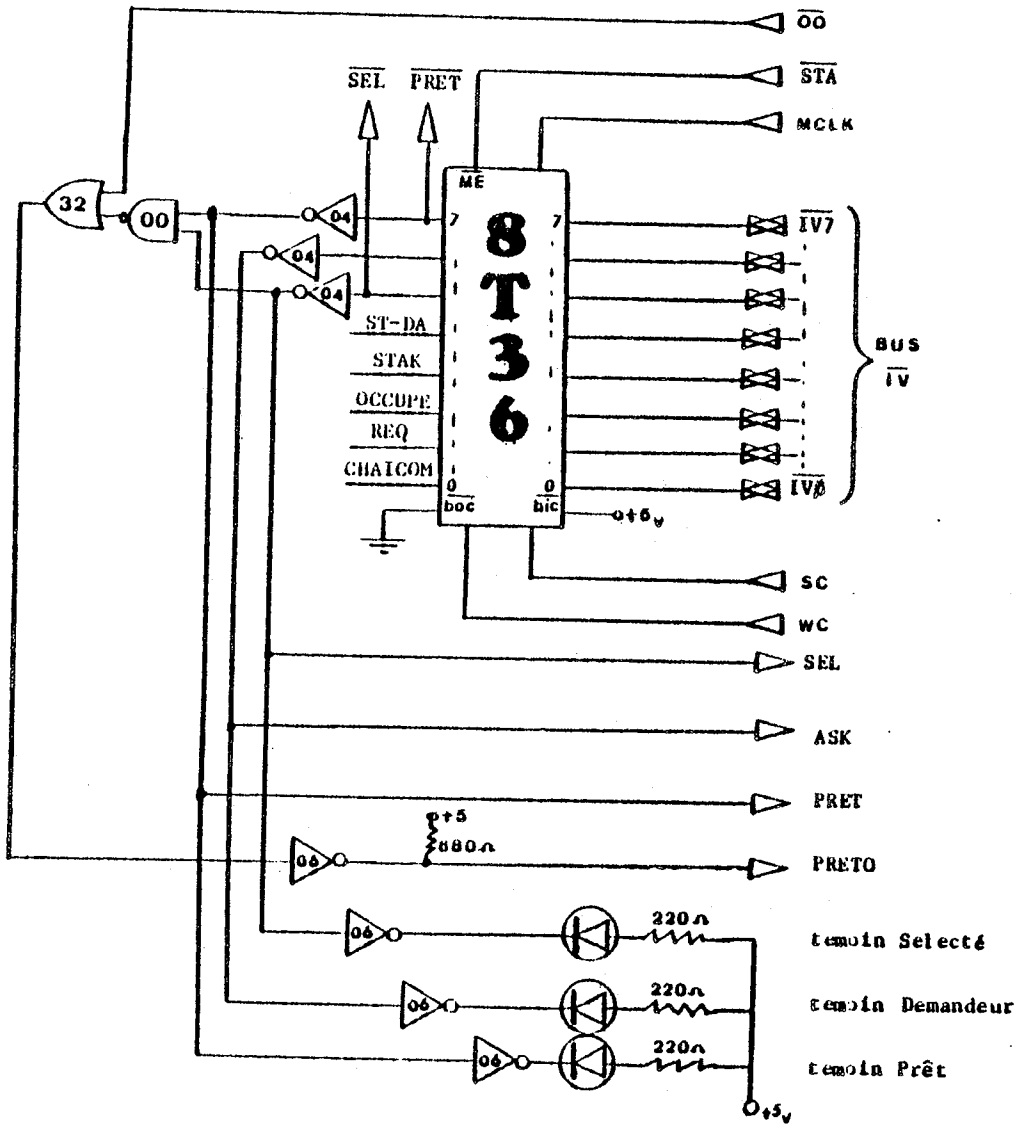


Planche 4 : Le registre des états internes

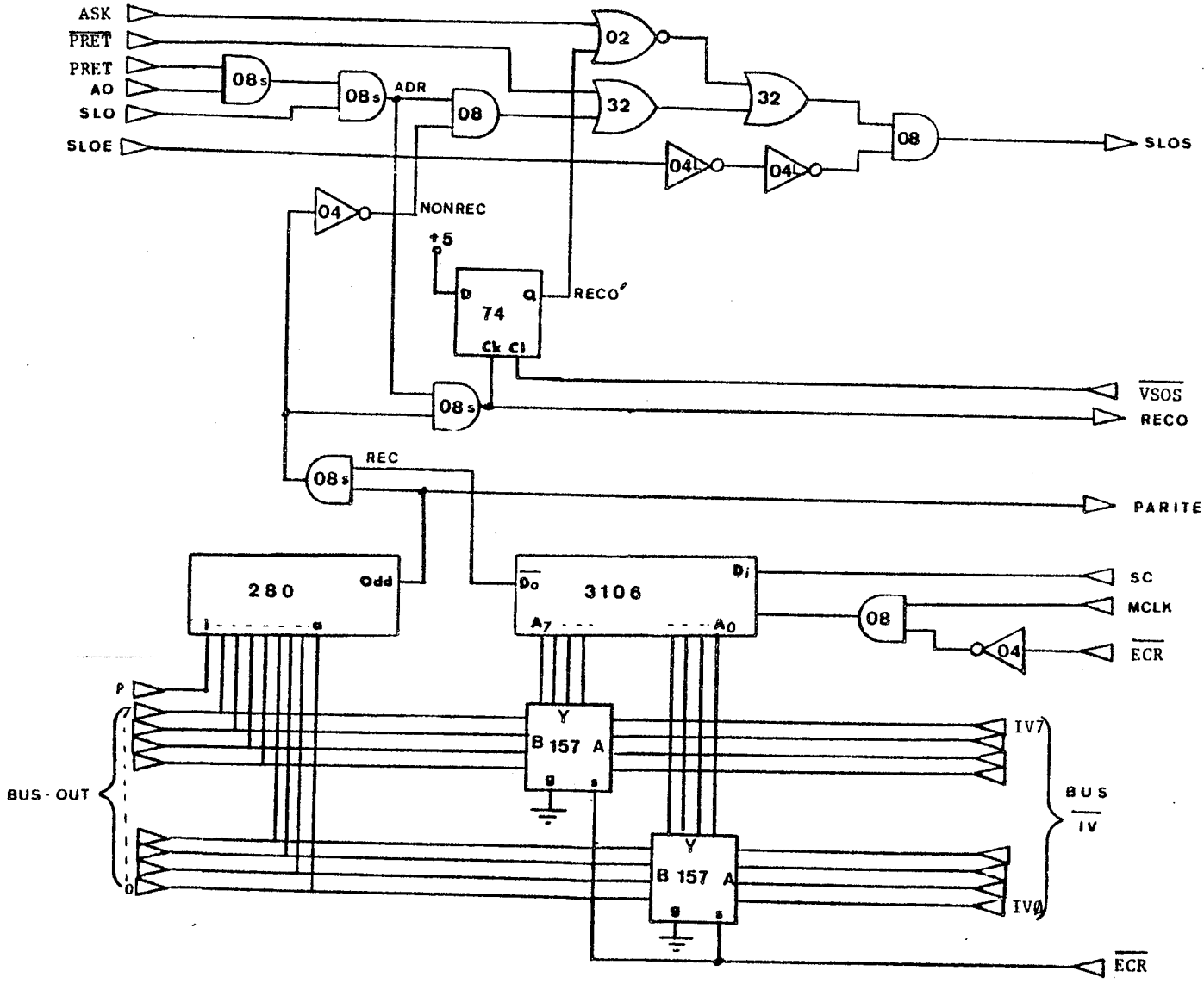


Planche 5 : Reconnaissance de l'adresse et propagation de SLO

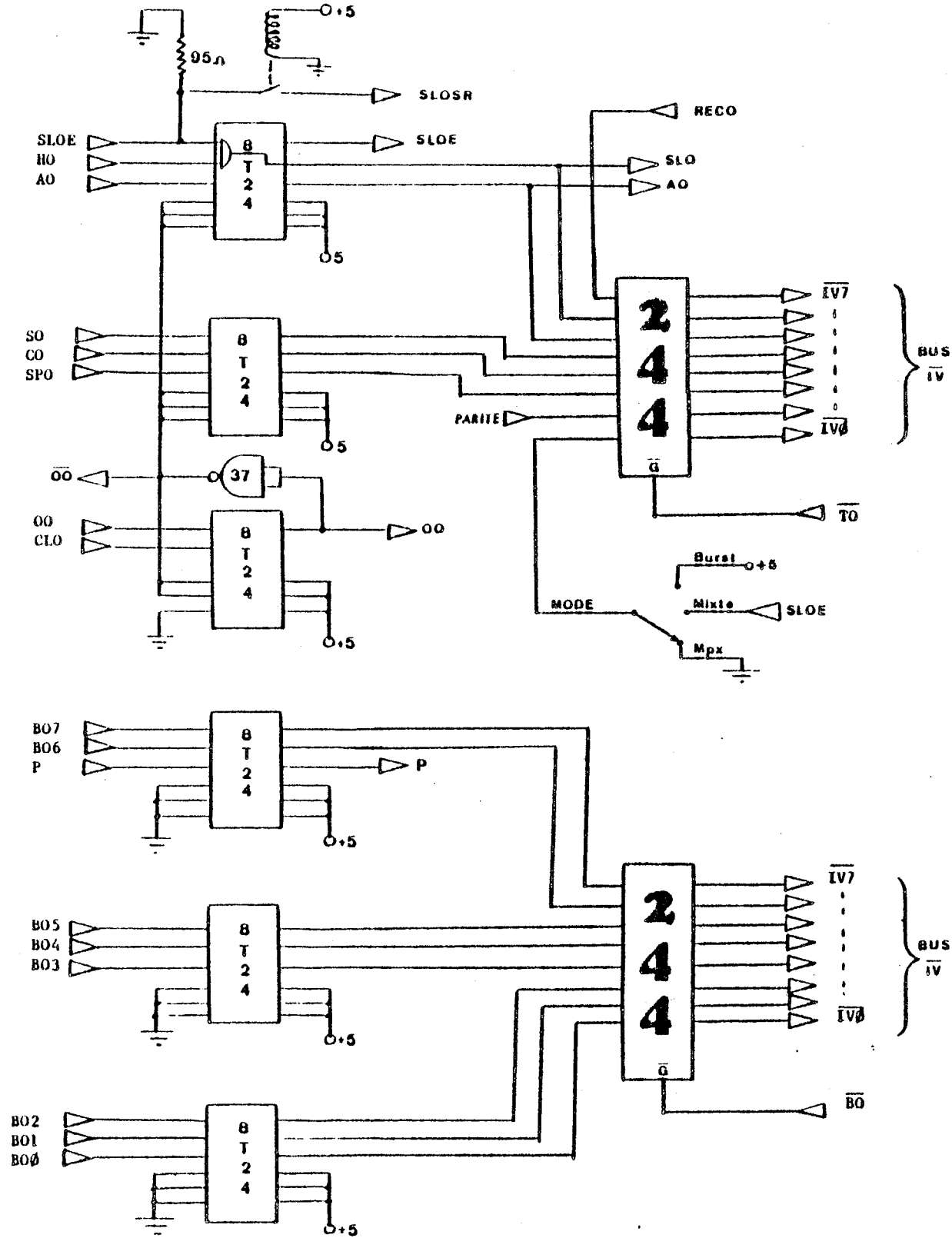


Planche 6 : Les circuits d'interface "Bus-out" et "Tag-out"

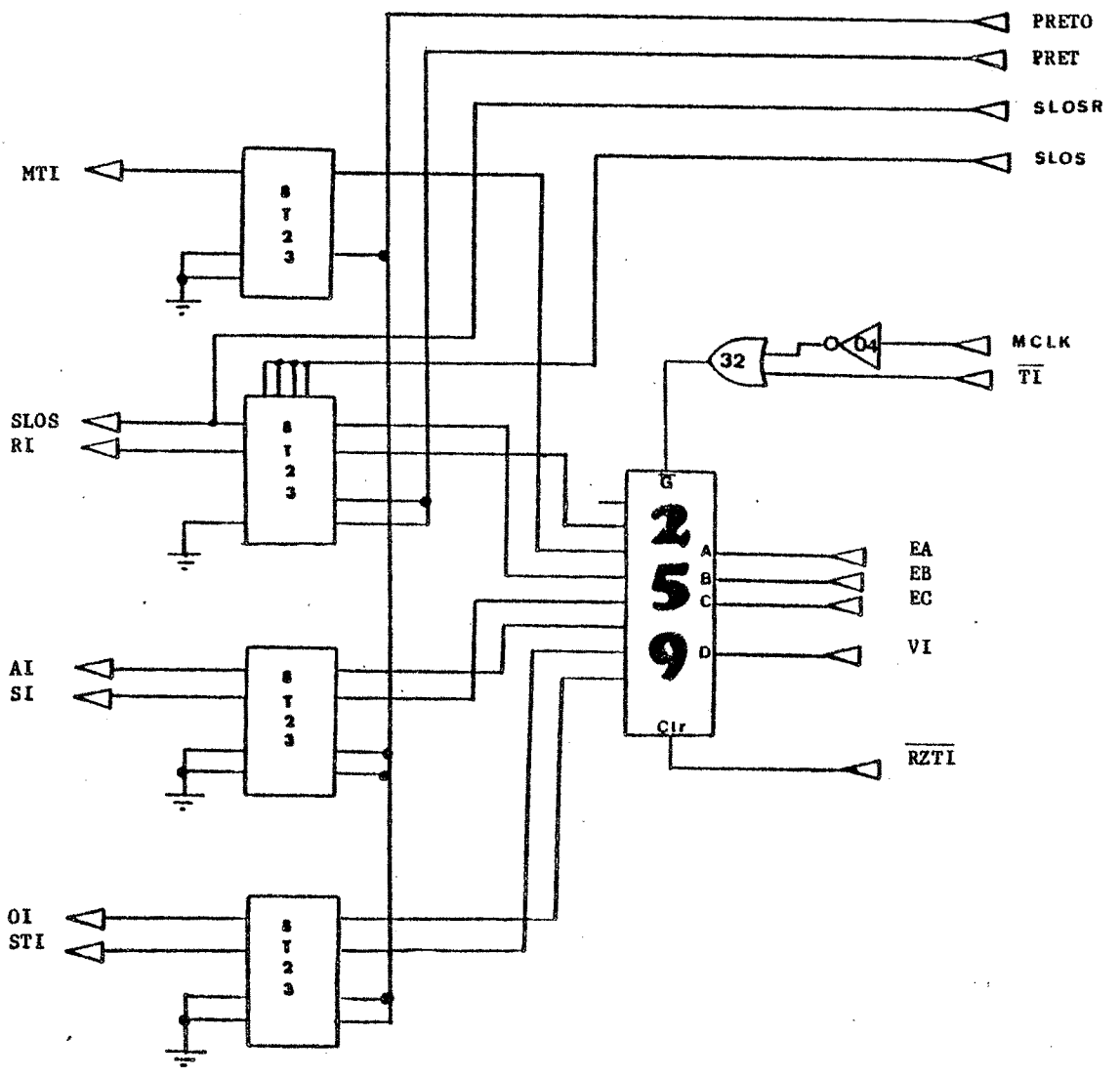


Planche 7 : Les circuits de génération et d'interface des "Tags-in"

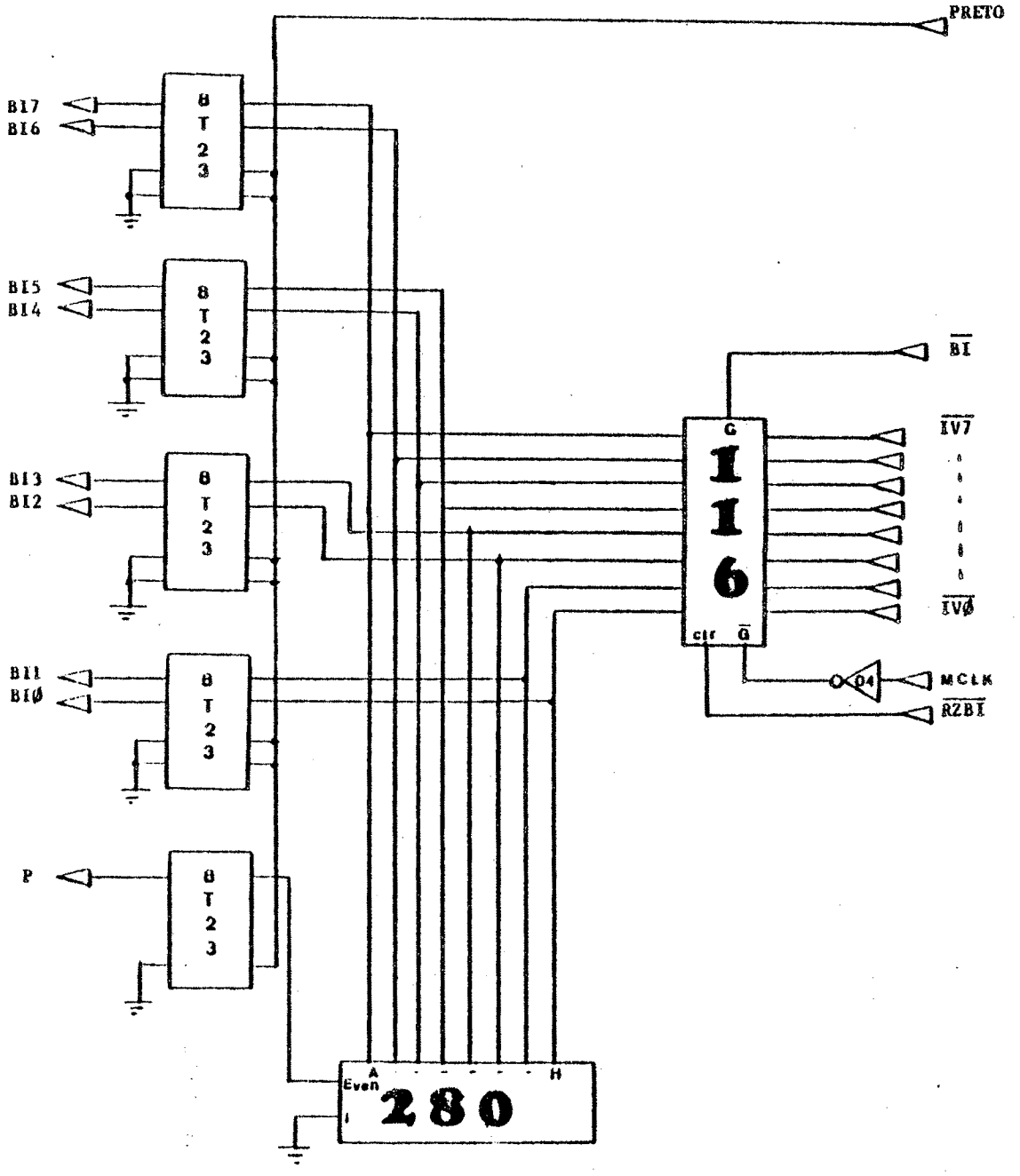


Planche 8 : Les circuits de génération et d'interface de "Bus-in"

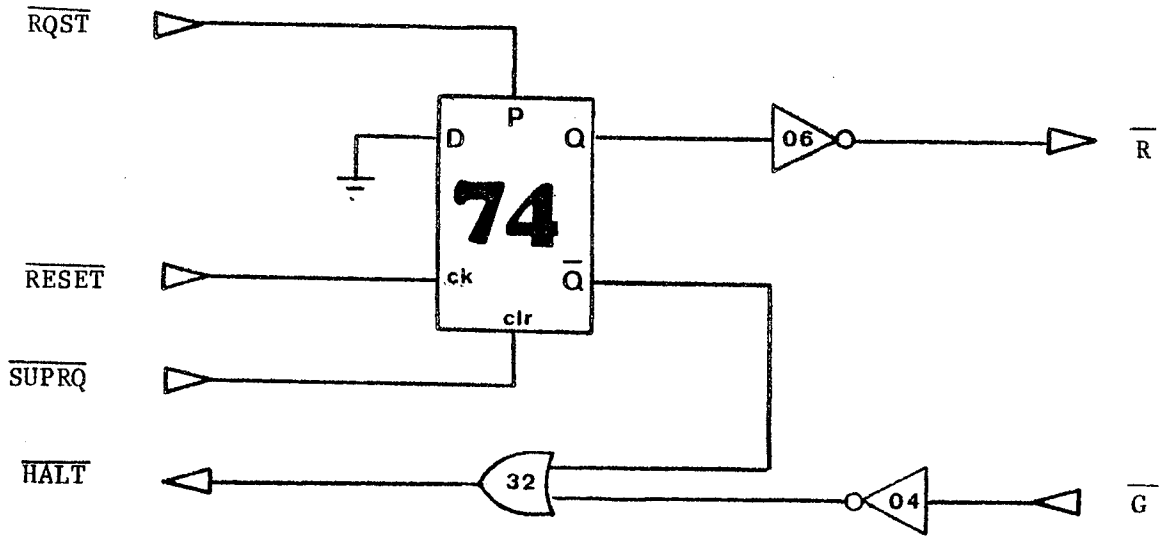


Planche 9 : Les lignes de synchronisation rapide

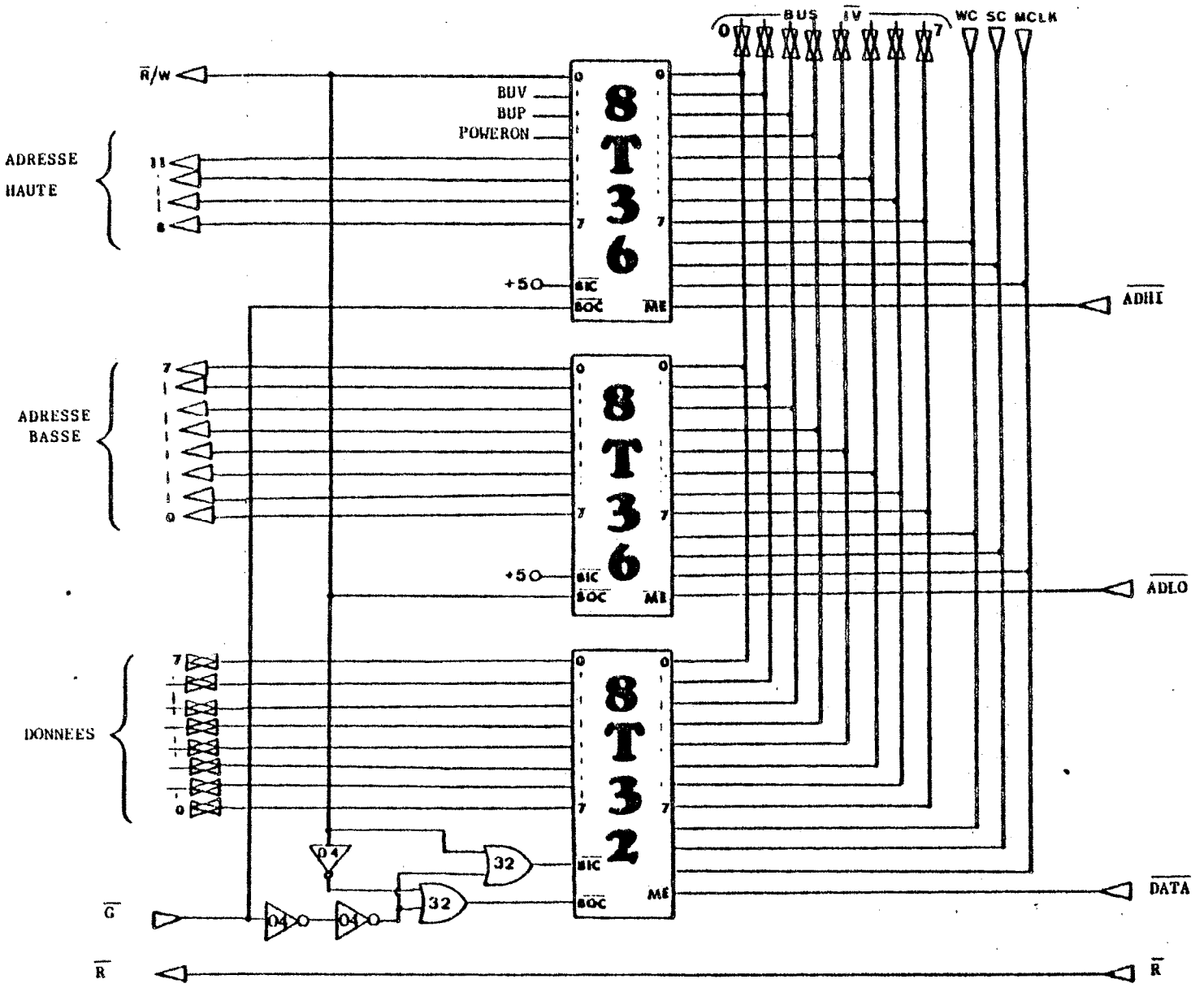


Planche 10 : Exemple d'application : partage d'une mémoire

ANNEXE 4

REALISATION D'UN CANAL-A-CANAL BOUCLE SUR LE CANAL MULTIPLEXEUR D'UN ORDINATEUR IBM 360

A4-1 Présentation du Canal-à-canal IBM (CTC) <IBM6>

La fonction principale du CTC est de fournir un moyen rapide de communication entre 2 systèmes IBM, ou entre 2 zones de mémoire d'un même système, par l'intermédiaire de 2 canaux de ces (ou de ce) systèmes.

Il est constitué de 2 unités de contrôle et d'une mémoire tampon de 1 octet. De cette façon, chaque canal est vu par l'autre comme une unité de contrôle. En général, le CTC est physiquement installé dans un des 2 canaux, et en est alors l'unité de contrôle de priorité maximale.

Cependant, s'il répond aux canaux auxquels il est connecté de la même façon qu'une unité classique, le CTC n'utilise pas les commandes reçues pour ses propres opérations. Il ne fait qu'ouvrir un chemin pour le passage des données entre les 2 canaux, donc entre les 2 mémoires ou les 2 zones de mémoire ; en outre, il n'assure aucun contrôle d'erreur sur les données transférées.

Les fig.A4-1 et A4-2 représentent 2 montages possibles d'un CTC.

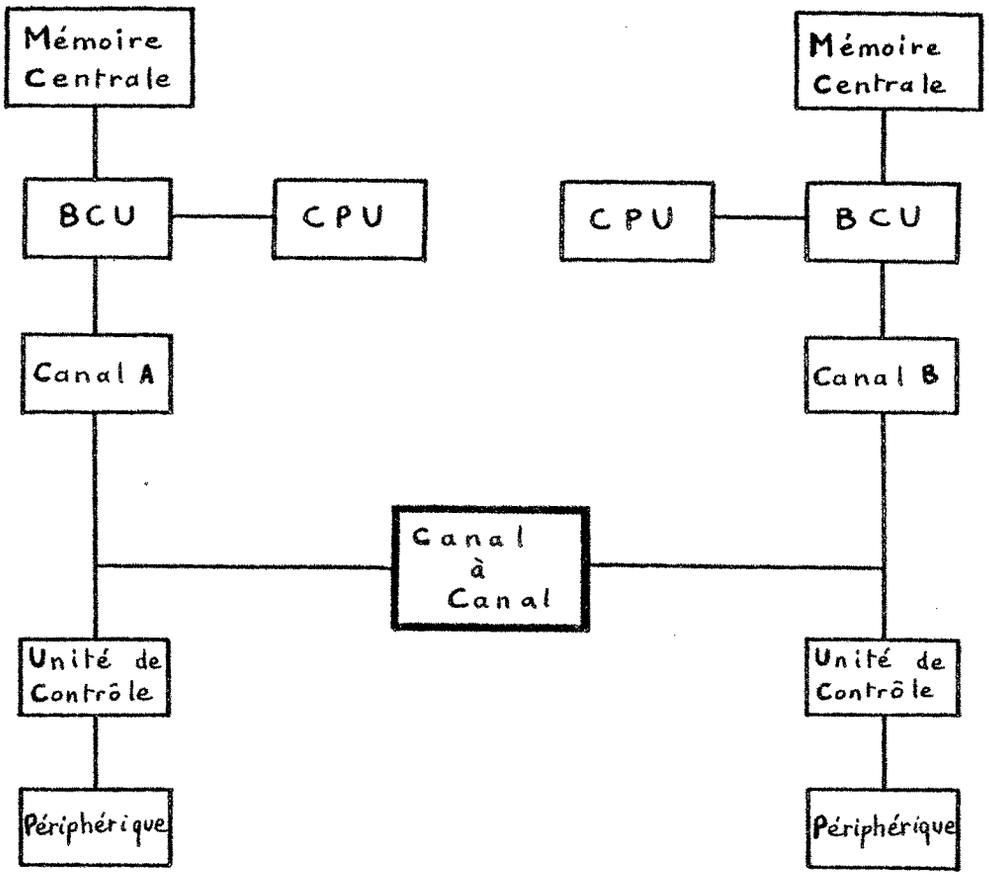


Fig.A4-1 Systèmes IBM communiquant par CTC

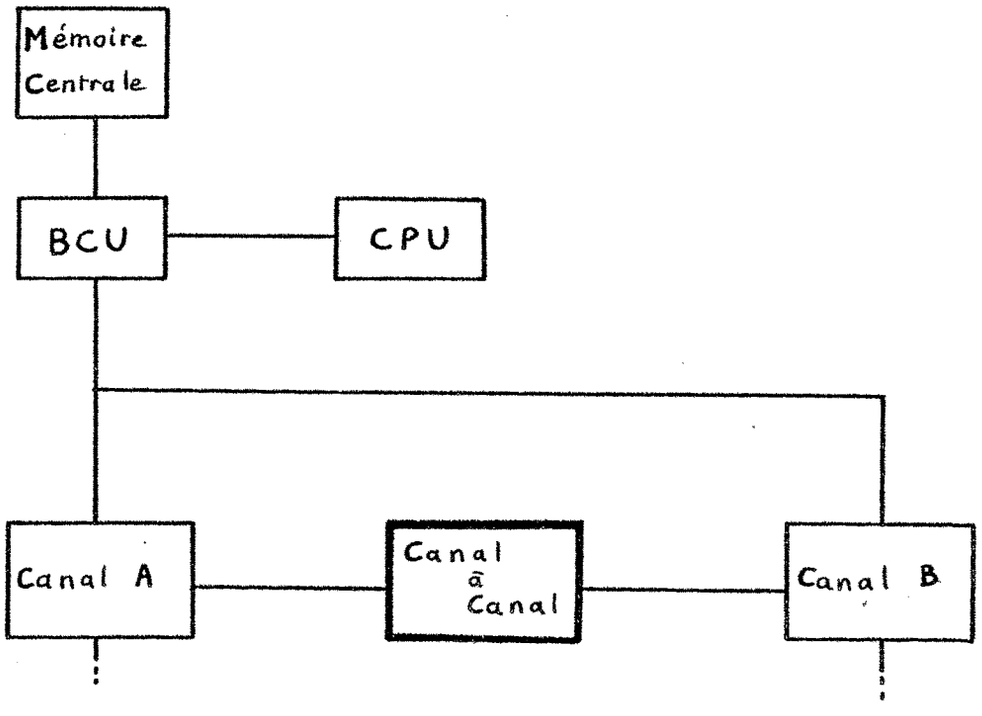


Fig.A4-2 CTC connecté à 2 canaux du même système

A4-2 Fonctionnement du CTC

Le CTC décode les commandes de base (lecture, lecture arrière, écriture, analyse et test I/O). En ce qui concerne la commande contrôle, il reconnaît les codes 03 (commande contrôle de base : NOP) et 07, 0B, 0F, etc...

Il n'utilise que 4 bits de l'octet d'état : "attention", "occupé", "fin sur canal" et "fin sur unité".

Il est donc chargé d'établir un chemin pour les transferts des données. Ceci implique que les 2 canaux (A et B par exemple) aient émis des commandes "compatibles", c'est-à-dire un couple "lecture-écriture". Pour cela, dès qu'il reçoit une de ces 2 commandes, de la part du canal A par exemple, le CTC, s'il est libre, opère de la façon suivante :

- il place l'octet de commande reçu dans sa mémoire et envoie l'état "attention" à l'autre canal (B dans notre exemple)

- normalement, à la réception de cet état, le programme doit prévoir l'envoi d'une commande d'analyse au CTC. Celui-ci transmet alors au canal B, comme octet d'analyse, l'octet de commande émis par le canal A

- après décodage de cette commande, le programme doit envoyer, via le canal B, la commande réciproque

- après s'être assuré que c'est bien le cas, le CTC commence le transfert des données depuis le canal "expéditeur" (celui qui a émis la commande d'écriture) vers le canal "destinataire" (qui a émis la commande de lecture). Ces transferts ont lieu en mode "burst" sur le CTC IBM jusqu'à ce qu'un des 2 canaux reconnaisse un compte nul dans le CCW

- le CTC envoie alors les états "fin sur canal" et "fin sur unité" sur les 2 canaux A et B.

Remarques :

-1- Entre chaque transfert, l'information (commande ou donnée) est rangée dans la mémoire tampon du CTC.

-2- La fin du transfert provoque le positionnement du bit "longueur incorrecte" dans l'octet d'état-canal comme indiqué dans le tableau ci-dessous, sauf si l'indicateur SLI est à 1 dans le CCW :

lecture du canal A		écriture du canal B	indication de longueur incorrecte	
			au canal A	au canal B
compte dans A	=	compte dans B	non	oui
- - -	>	- - -	oui	oui
- - -	<	- - -	oui	oui
- - -	=	- - -	oui	non
		moins 1		

-3- Si un canal envoie une commande contrôle, le CTC doit la ranger dans sa mémoire et prévenir l'autre canal à l'aide de l'état "attention" pour qu'il fasse une "analyse" afin de récupérer la commande contrôle initiale.

-4- Un canal ayant émis une commande de lecture, d'écriture ou de contrôle, reste dans l'état occupé jusqu'à ce que l'autre canal émette la commande nécessaire, ou que l'un des 2 fasse un Halt I/O.

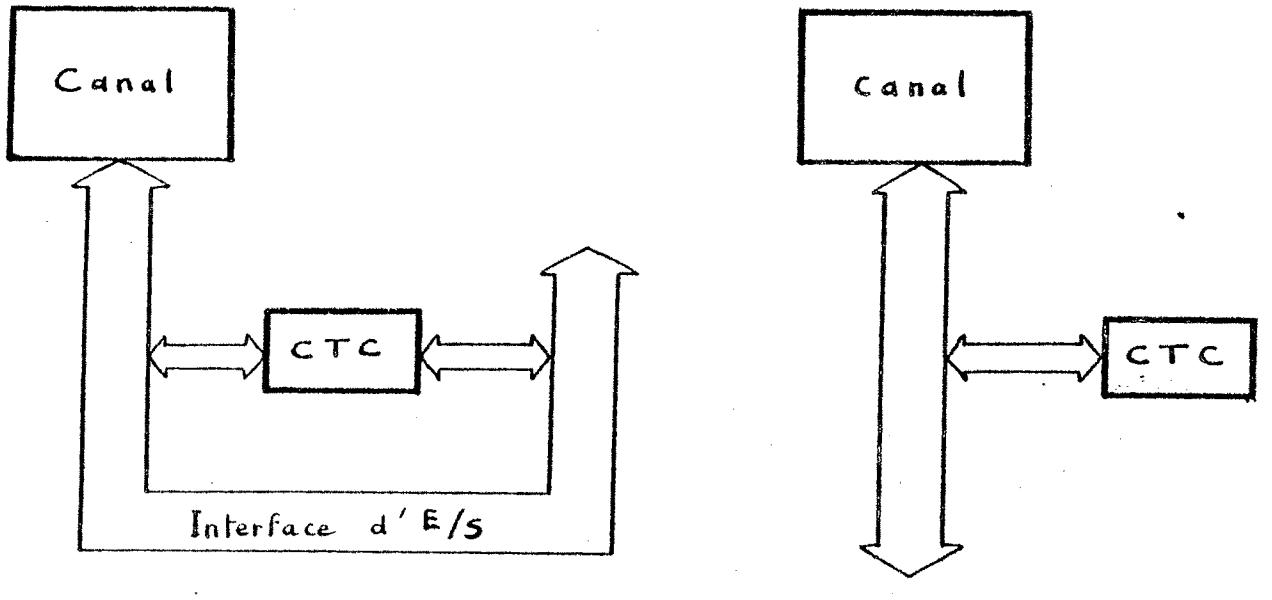
-5- Dans ce dispositif, la commande d'analyse ne sert donc qu'à fournir à un des canaux, la commande émise par l'autre ; sauf si le CTC est libre, auquel cas l'octet envoyé est identiquement nul.

La fig.A4-4 résume les traitements effectués par le CTC pour chaque commande envoyée par le canal A (par exemple), en fonction de la commande en cours le cas échéant, et de la commande envoyée par le canal B.

A4-3 Application : mécanisme de communication entre 2 machines virtuelles

Cette application s'inscrit dans le cadre de l'implémentation d'une station de transport du réseau CYCLADES sur le 360/67 du CICG (voir Chapitre IV). Une des solutions étant d'implémenter la SI comme tâche utilisateur du système d'exploitation, au même titre que les abonnés, le CIC offre alors un moyen de communication entre ces tâches. Il peut être vu comme une alternative matérielle au système "CVC" (CPU Virtual Communication) qui permet la liaison entre machines virtuelles sous CP-67 <ANS>.

Dans ce cas, le CIC se trouve connecté à un seul canal, comme l'indique la fig.A4-3.



configuration fonctionnelle configuration réelle

Fig.A4-3 Le CIC dans une application à canal unique

Nous nous placerons donc dans la suite, dans le cas où le système d'exploitation est CP-67 et où chaque tâche est dans une machine virtuelle. Si d'autre part on se contente dans un premier

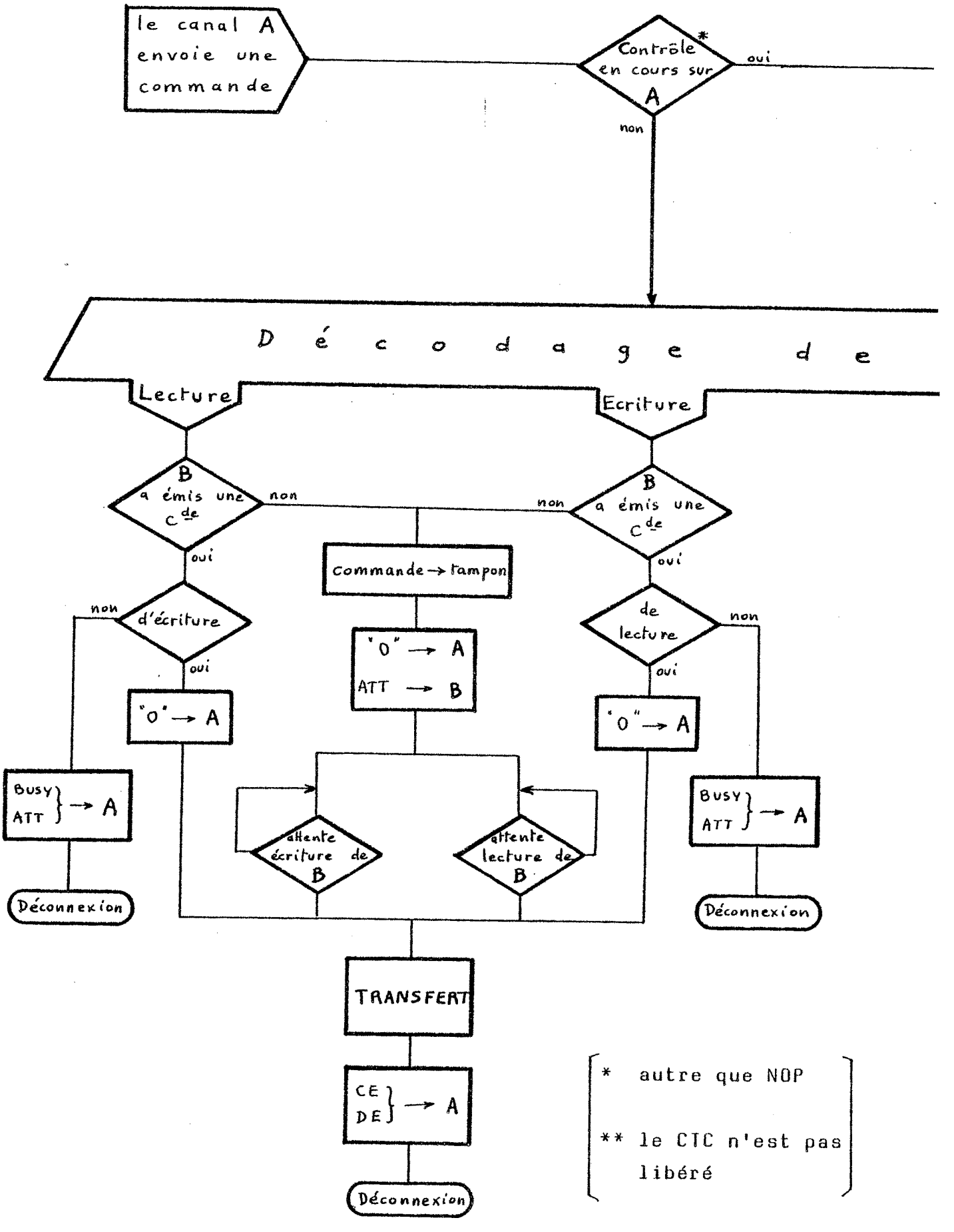
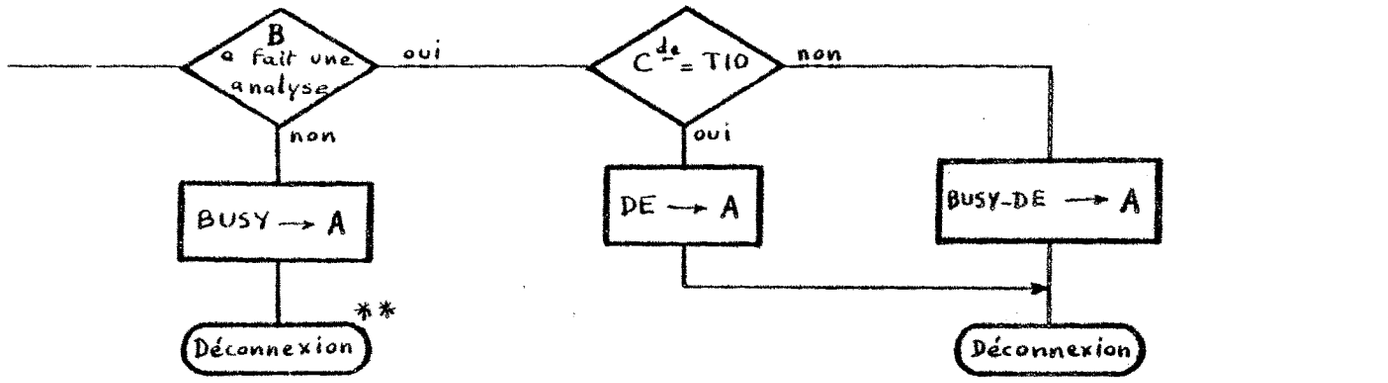
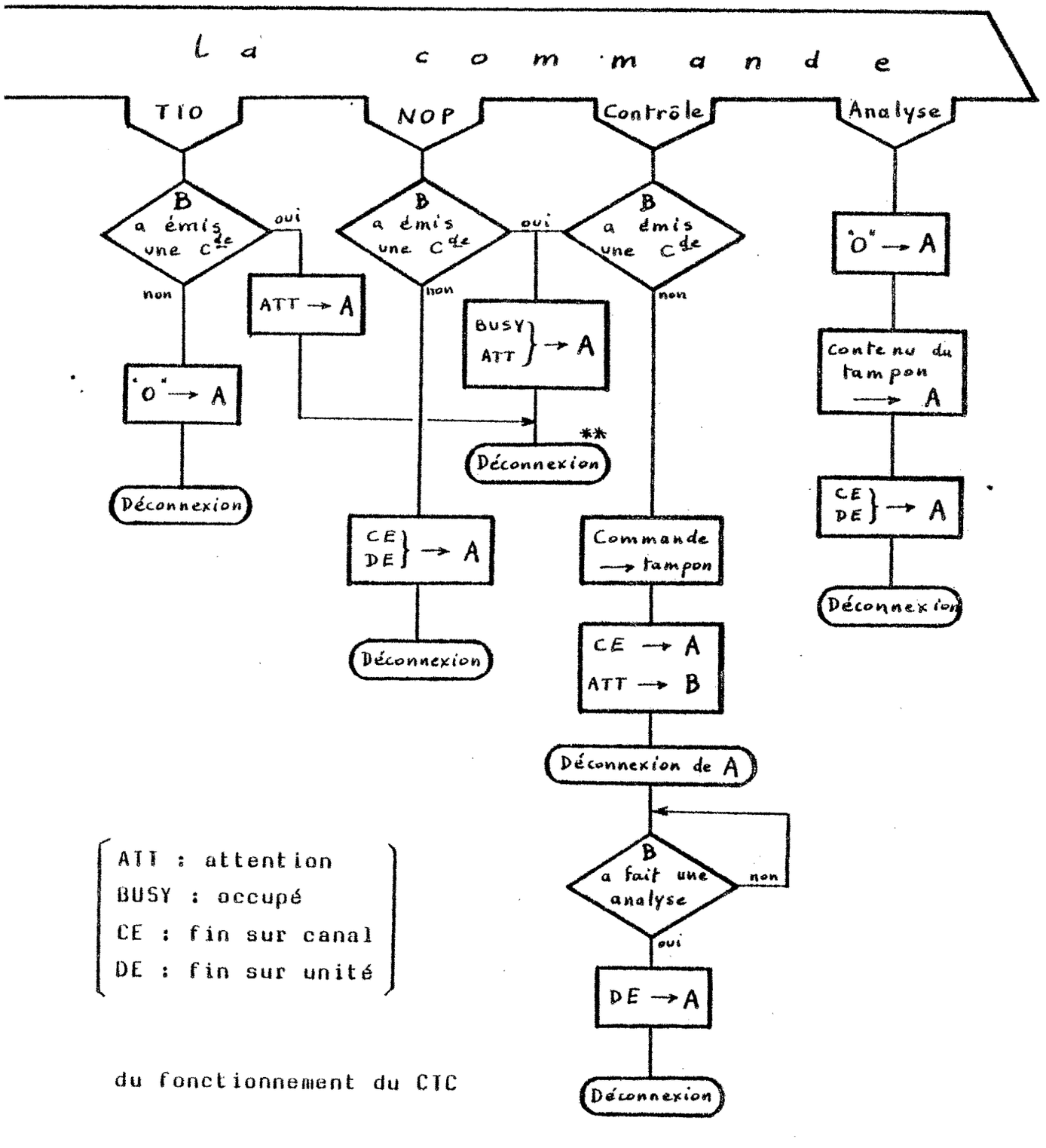


Fig.A4-4 Organigramme général simplifié





ATT : attention
BUSY : occupé
CE : fin sur canal
DE : fin sur unité

du fonctionnement du CTC

temps, de faire communiquer 2 machines virtuelles (MV) seulement, le CTC se présente vis-à-vis du canal comme 2 unités de contrôle, chacune d'elles étant attachée à une MV (par la commande "ATTACH" de CP-67). La procédure indiquée au paragraphe précédent se déroule de la même façon, à la différence que tous les dialogues et tous les échanges ont lieu sur le même interface d'E/S.

La réalisation qui en a été faite se caractérise par :

- l'utilisation d'un microprocesseur standard : le MC 6800 de Motorola

- la programmation de la procédure CTC

- la reconnaissance de 2 adresses d'unités d'E/S

- le respect des spécifications IBM concernant la connexion au canal

- l'implantation de programmes conversationnels. Ceux-ci ont permis, outre la vérification du fonctionnement du CTC, d'acquérir une connaissance de fait du comportement du canal pendant les échanges élémentaires.

A4-5 Architecture du CTC

Dans cette réalisation, le CTC se présente comme une carte coupleur insérée dans une maquette banalisée, bâtie autour du MC 6800 (fig.A4-5). Cette maquette fournit 16 pages de 4 K.octets. Pour ce qui concerne notre application, la mémoire programme est en page 0, la page 14 est attribuée au coupleur et la page 15 à l'interface conversationnel. La maquette offre également un superviseur. Un mécanisme indépendant à base de roues codeuses et d'afficheurs hexadécimaux, permet la visualisation immédiate d'un octet de l'instruction lors du passage à une adresse pré-déterminée.

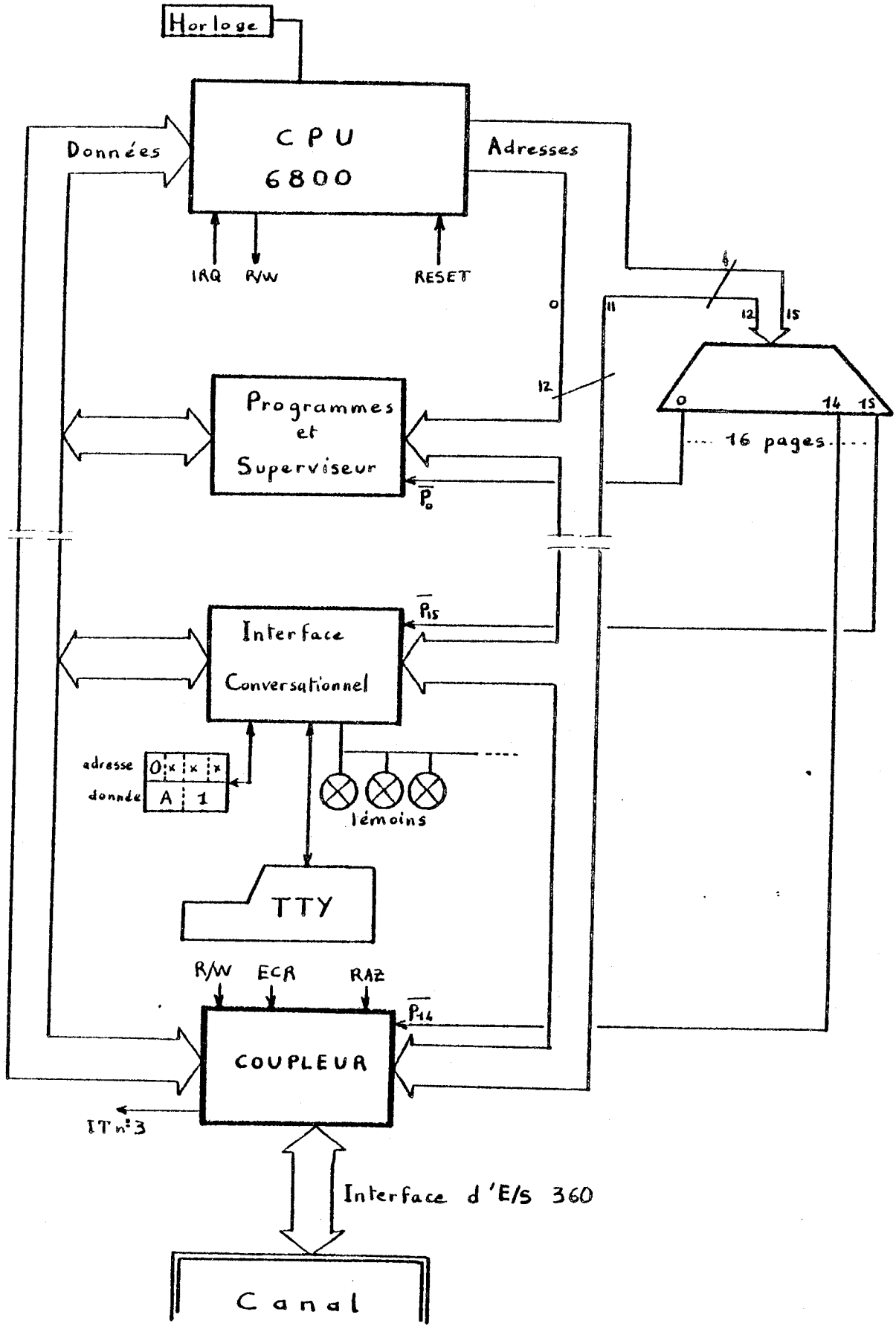


Fig.A4-5 Configuration générale du CIC

L'architecture simplifiée de la carte coupleur est donnée à la fig.A4-6. On y retrouve les fonctions nécessaires à la gestion de l'interface d'E/S :

- génération et réception des Tags-in et des Tags-out
- génération et réception de Bus-in et Bus-out
- reconnaissance des adresses des unités
- génération d'états et de commandes internes
- génération et suppression d'une interruption au CPU.

Dans cette solution à microprocesseur standard, toutes les variables manipulées par le coupleur sont des adresses mémoire (de la page 14) : lignes de contrôle de l'interface, états internes et certaines commandes. Ces adresses sont décodées de la façon suivante :

- les 3 bits de plus faible poids servent à générer les Tags-in et les états internes, et également à reconnaître un Tag-out ou un état interne, parmi 8
- le quatrième bit sert de valeur immédiate dans l'écriture et la lecture des lignes précédentes
- les 3 derniers bits donnent 8 commandes internes : sélection des registres d'E/S, sélection de la mémoire des unités, demande ou suppression d'interruption au microprocesseur.

Après validation des 2 adresses attribuées au CTC, le processus de reconnaissance d'adresse et de propagation de SLO est identique à celui mis en oeuvre dans l'UCB et décrit au Chapitre III. La seule différence est que ce n'est pas le logiciel qui teste l'indicateur RECO, mais ce dernier qui provoque une interruption au CPU.

Remarque : les 2 adresses d'unités sont choisies pour ne se différencier que par la valeur du bit de poids faible.

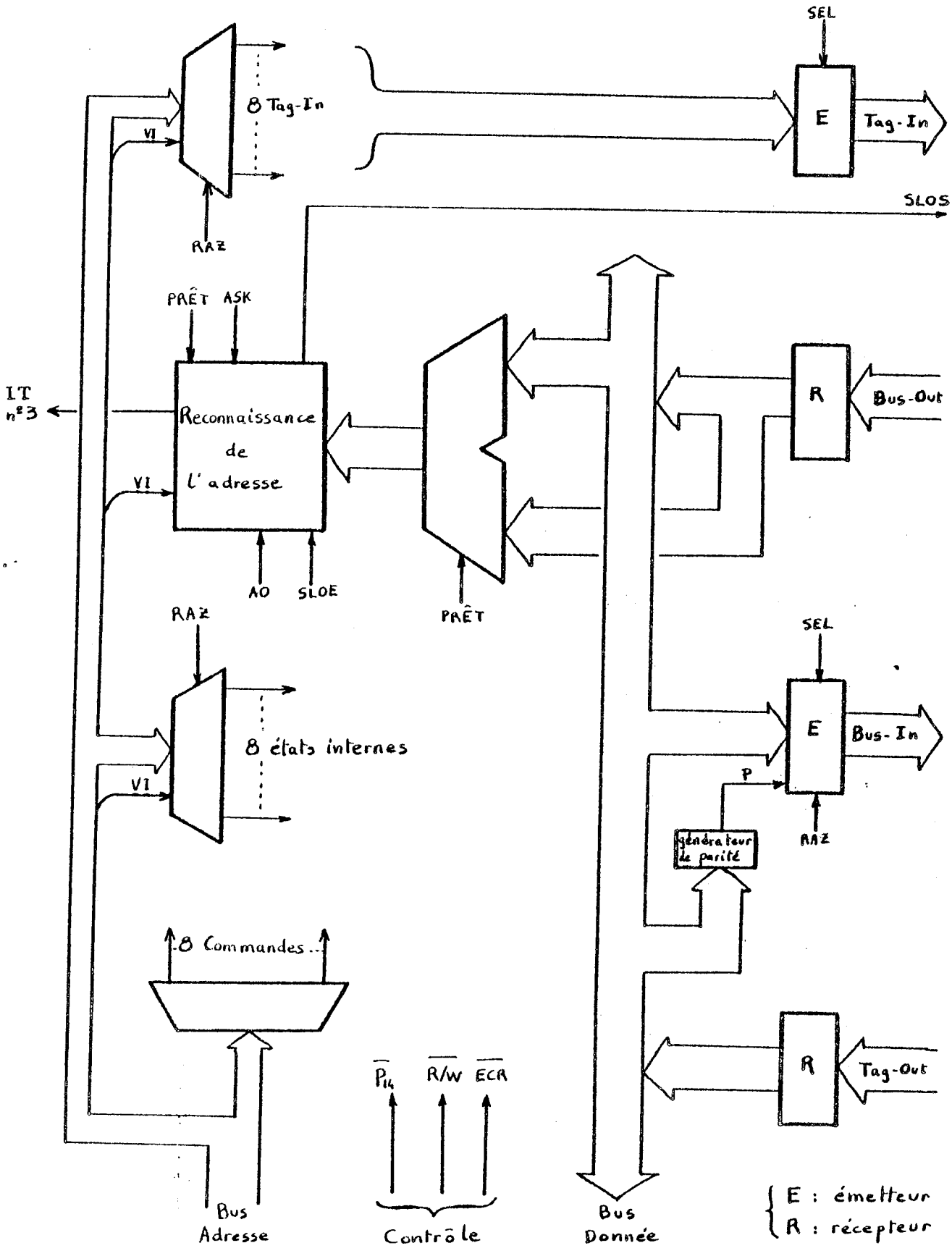


Fig.A4-6 Architecture simplifiée de la carte coupleur du CTC

A4-5 Aspects logiciels

Il a été réalisé successivement 3 logiciels :

1) Un logiciel de test de la carte coupleur.

Pendant son exécution, les lignes "in" et "out" de l'interface d'E/S sont reliées , permettant d'une part un test du câblage et des circuits du coupleur, et d'autre part une simulation de quelques séquences simples.

2) Un programme conversationnel permettant des échanges avec le canal sous le contrôle de l'opérateur. Ce programme comporte :

- un ensemble de sous-programmes de dialogue avec un Télétype : LIRC et PRIN (lecture et impression d'un caractère), LIRO et PRIO (lecture et impression d'un octet de donnée)

- des sous-programmes réalisant les connexions logiques avec le canal : CAND (si l'unité est demandeur) et ITAD (programme de traitement de l'interruption au microprocesseur, suite à une sélection par le canal)

- un ensemble de sous-programmes de dialogue avec le canal : EMIS et RECEP (émission et réception de données), PRES (envoi d'un état-unité asynchrone)

- un sous-programme effectuant le transfert "ping-pong" d'un octet entre les 2 zones mémoire.

L'exécution de ces sous-programmes est lancée par l'opérateur (sauf ITAD) et les résultats sont imprimés sur le Télétype : soit sous forme explicite (écriture de la donnée reçue, par exemple), soit sous forme codée sur un caractère.

Ce programme permet de répondre au logiciel de test écrit sur le 360 (Chapitre V). Nous en montrons des exemples à la fig.A4-7.

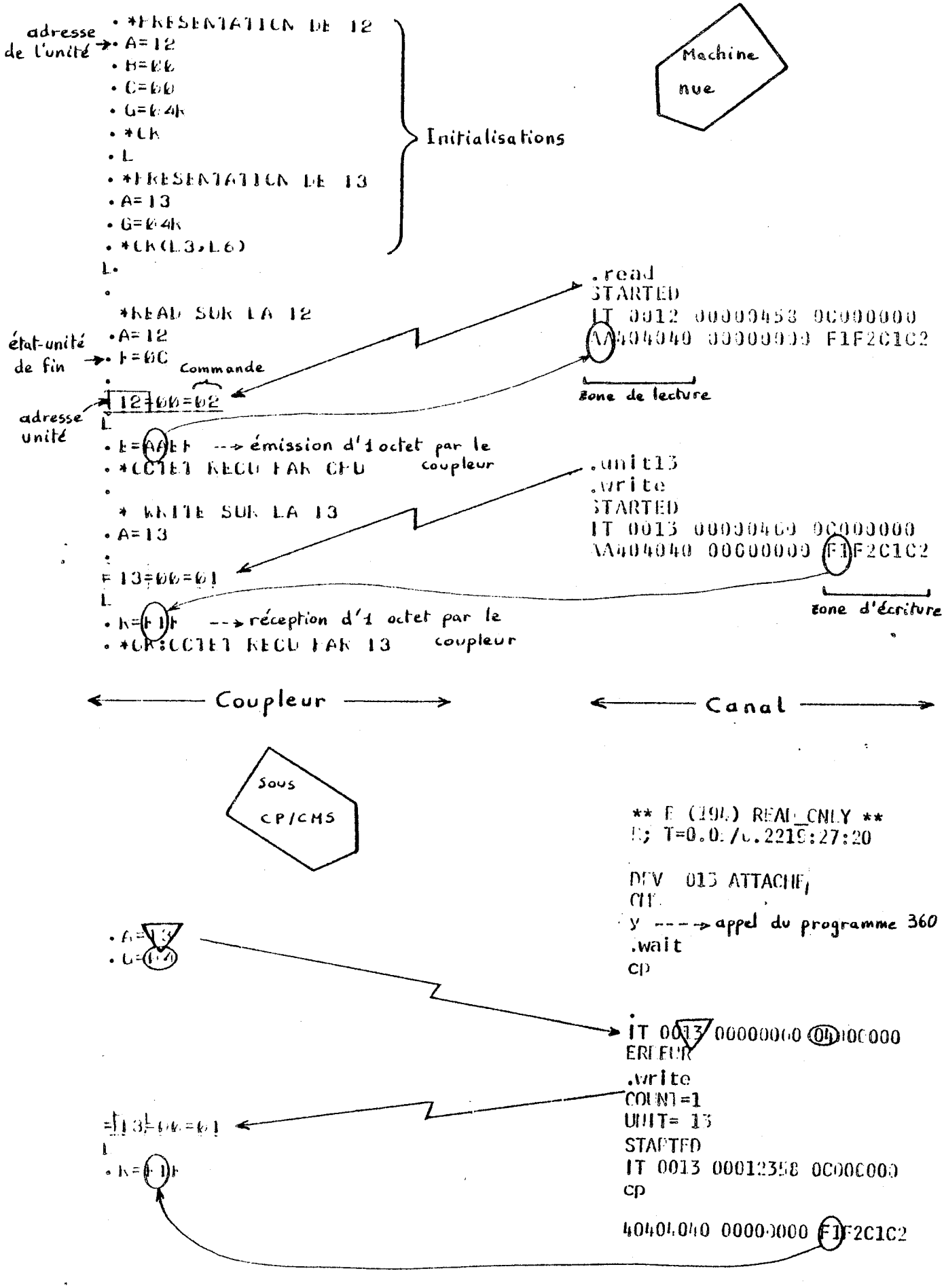


Fig.A4-7 Quelques dialogues entre le canal et le coupleur

3) Un programme qui réalise complètement le fonctionnement "Canal à Canal" décrit au paragraphe A4-2.

Ce programme doit donc simuler le comportement des 2 unités de contrôle du CIC, en respectant l'organigramme de la fig.A4-3. En fait, nous l'avons simplifié en ne tenant pas compte, dans un premier temps, de la commande contrôle.

Il exécute le double automate (un par unité) de la fig.A4-8 dont les états sont :

- état 0 : l'unité sélectée refuse toute commande en envoyant l'état-unité "Occupé" comme état-unité initial

- état 1 : l'unité accepte toutes les commandes (sauf contrôle)

- état 2 : l'unité ne peut recevoir qu'une commande d'analyse ou la commande compatible avec celle qui l'a placée dans cet état : par exemple une lecture, si une écriture a été lancée sur l'autre unité

- état 3 : l'unité accepte seulement une commande de lecture

- état 4 : l'unité accepte seulement une commande d'écriture.

Dans ces 2 derniers états, lorsque la commande attendue arrive, l'unité passe dans l'état 0, et le transfert s'effectue, octet par octet, jusqu'à la détection par une des unités, du signal "stop" de la part du canal. Chacune d'elles envoie alors l'état-unité de fin, c'est-à-dire "fin sur canal et fin sur unité".

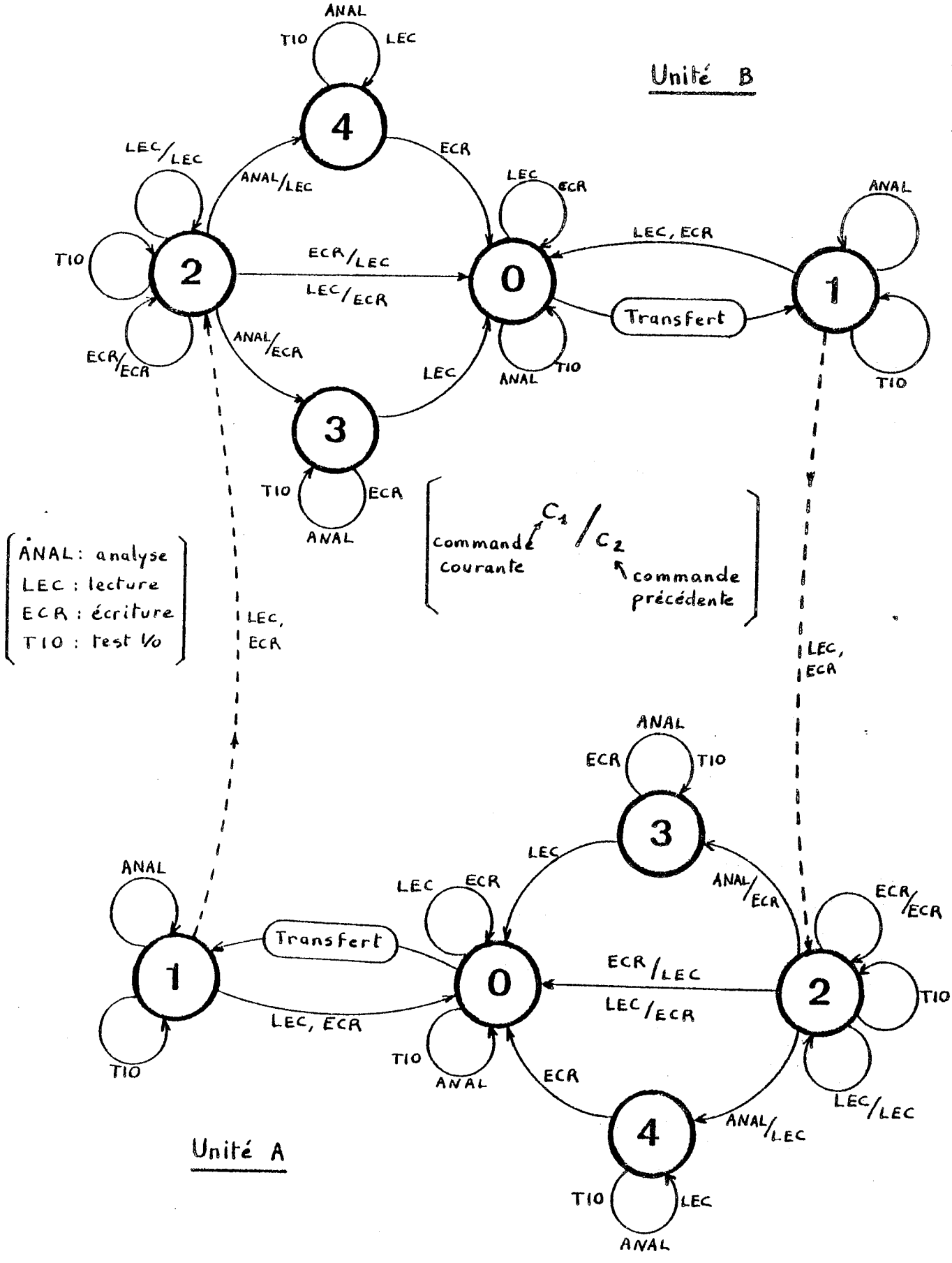


Fig.A4-8 Le double automate du CIC

Remarques :

-1- Quel que soit l'état dans lequel se trouve une unité, une commande d'analyse provoque toujours l'envoi de l'octet de donnée du CTC. Il contient la commande précédemment envoyée à l'autre unité, le cas échéant.

-2- L'unité qui reçoit une commande de lecture ou d'écriture, passe dans l'état 0 et envoie une demande d'interruption au CPU, sous la forme d'un état-unité asynchrone, avec "Attention". Cet état-unité indique qu'une commande convenable (analyse, ou directement la commande compatible) est attendue sur l'autre unité. Un Test I/O ne supprime pas le bit "Attention" dans l'état-unité.

-3- Quand une unité reçoit une commande autre que celle attendue lorsqu'elle est dans les états 2, 3 ou 4, elle la refuse en envoyant comme état-unité initial : "Occupé et Attention", ou "Attention" seul s'il s'agit d'un TIO. Seule la commande d'analyse est acceptée comme on vient de le voir.

-4- Dans la version réalisée, le transfert des octets a lieu dans le mode "multiplex", alors que le CTC câblé IBM travaille en mode "burst".

A4-6 Résultats obtenus - Conclusion

Le fonctionnement du CTC est illustré par les dialogues de la fig.A4-9. Ces dialogues ont été réalisés hors système d'exploitation et ils montrent que les transferts d'octets ont bien lieu entre les zones mémoire correspondant à chaque commande. Un échange de 64 K.octets s'est même déroulé sans problème.

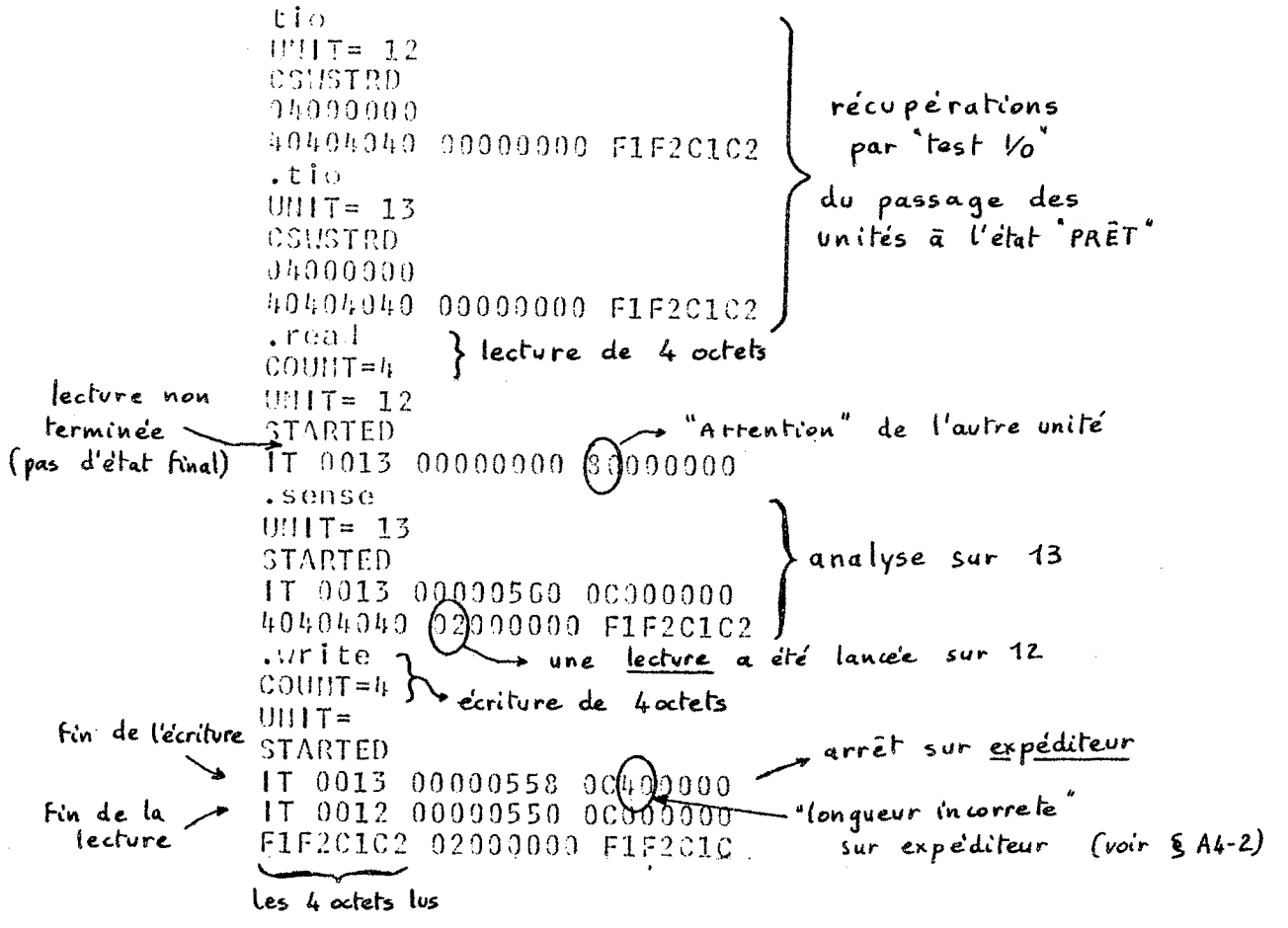
Par contre, sous le système CP/CMS, nous nous sommes heurtés aux difficultés signalées au Chapitre V, dues au fait que le CTC utilise des états-unité (comme "Attention" par exemple) qui ne sont pas valides pour des unités de contrôle de bandes magnétiques, auxquelles il était pourtant assimilé par le système. Cependant, le coupleur lui-même était opérationnel sous ce système (voir fig.A4-7).

En ce qui concerne les débits, nous donnons une valeur maximum théorique qui correspond ici à la valeur pratique, puisque dans cette réalisation, la vitesse de transfert est limitée par le microprocesseur.

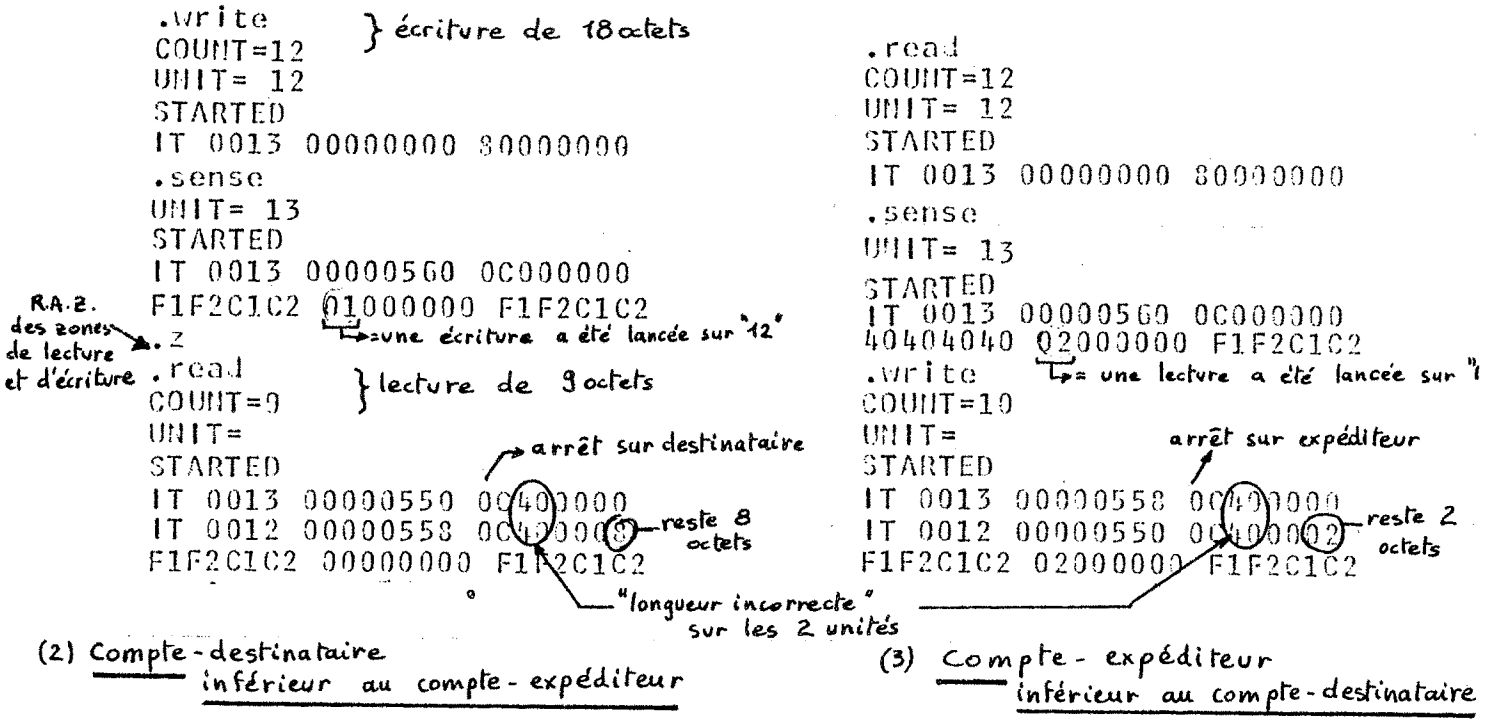
La durée d'un échange élémentaire est de 34 µs, ce qui donne un débit d'environ 30 K.octets/s. Mais la séquence de sélection initiale dure environ 240 µs et la séquence de sélection par l'unité, pour présenter un état-unité par exemple, dure environ 100 µs. De plus, il faut rappeler que nous n'avions prévu ni contrôle d'erreur de parité, ni test du mode de fonctionnement, ni test des séquences "interface disconnect" et suppression de donnée. Ces durées supposent une horloge de 1 MHz, qui est celle utilisée normalement avec le 6800.

Ne disposant pas à l'époque d'analyseur logique, nous ne pouvons pas montrer les différentes séquences. Nous avons seulement pu réaliser une boucle sur une instruction Test I/O pour visualiser une "CISS" à l'oscilloscope, et les durées mesurées étaient bien celles prévues par le programme.

En conclusion, bien que les séquences d'interface avec le canal aient été correctement exécutées, les débits obtenus ont montré la nécessité de réaliser un coupleur plus performant, avec un autre matériel. Par contre l'implantation de l'algorithme de fonctionnement du CTC, comme probablement celui d'une autre application, semble plus facile sur un microprocesseur de ce type.



(1) Transfert avec comptes égaux



(2) Compte-destinataire inférieur au compte-expéditeur

(3) Compte-expéditeur inférieur au compte-destinataire

Fig.A4-9 Quelques dialogues illustrant le fonctionnement du CIC

ANNEXE 5

REALISATION D'UN ASSEMBLEUR POUR LE 8X300

A5-1 Introduction

Une partie des recherches menées actuellement dans le domaine des microprocesseurs concerne la mise au point d'outils de production de logiciels.

La grande diversité des microprocesseurs et donc le grand nombre de langages d'assemblage différents d'une part, la quasi-nécessité d'utiliser un calculateur hôte pour supporter ces outils d'autre part, ont orienté Y.Bekkers et P.Fontanille, au sein de l'Atelier de Micro-informatique de Grenoble, vers la conception d'un générateur d'assembleurs croisés.

La concrétisation de ces travaux dans le Système GAGE (Générateur d'Assembleur de GrenobLE) a heureusement coincidé avec notre besoin de disposer d'un assembleur pour le 8X300 afin de mettre au point les logiciels INICAN et INIDISP.

Dans cette annexe nous décrirons brièvement le système GAGE et plus particulièrement son langage de description d'un assembleur, à partir du cas précis du 8X300 et de ses problèmes spécifiques.

Nous donnerons ensuite la description complète de notre assembleur ainsi que quelques exemples montrant son utilisation.

A5-2 Description simplifiée de GAGE <BEK>

1) Caractéristiques générales

Parmi les principes qui ont guidé la conception du langage de GAGE, on peut citer :

- simplicité de la syntaxe des opérandes
- assembleur entièrement contenu dans sa description
- description claire, dense et rapide.

Ces principes associés à un processus d'assemblage très général, à la banalisation des mnémoniques et à l'acceptation d'instructions multi-formats, ont permis d'obtenir les caractéristiques suivantes :

- rapidité de prise en compte d'un nouveau microprocesseur (1 à 2 hommes.jour)
- portabilité : l'ensemble des programmes est écrit en FORTRAN IV standard
- facilité d'emploi : ne nécessite pas de connaissances poussées en logiciel (ni même en FORTRAN !).

2) Génération d'un assembleur

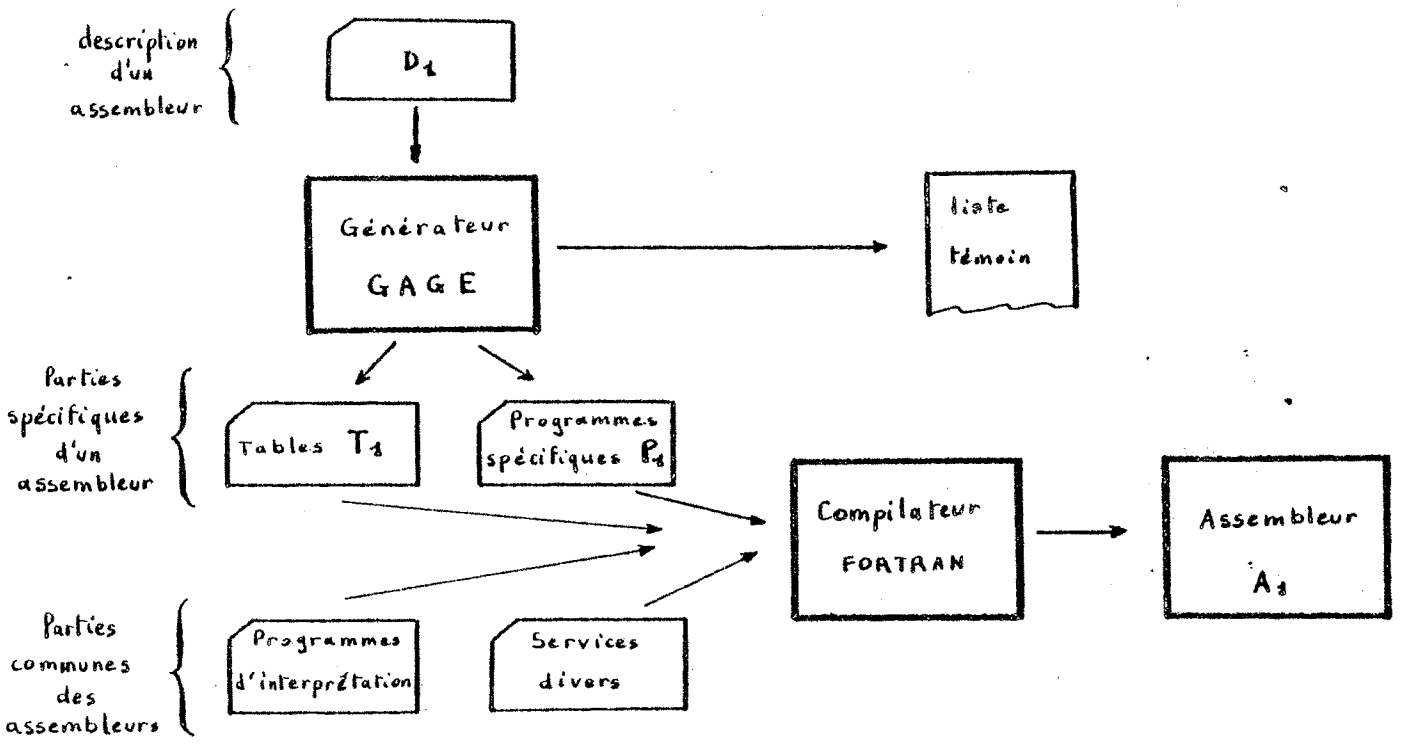
Le système GAGE utilise 3 produits indépendants mais complémentaires :

- un générateur (GAGE lui-même) qui traduit en FORTRAN la description d'un assembleur et fournit une liste témoin en vue de sa mise au point

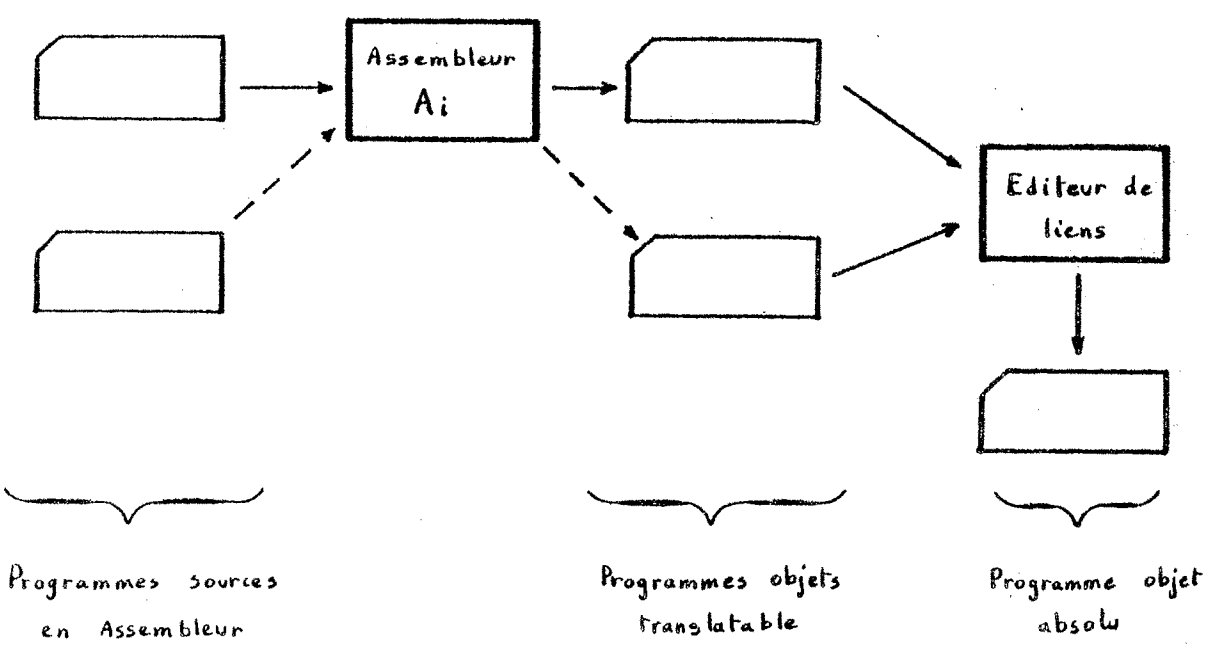
- une librairie de programmes FORTRAN : certains, spécifiques, interprètent les tables produites par le générateur, et d'autres sont des services communs à tous les assembleurs générés (édition du listing, gestion des identificateurs, etc...) ; ces programmes constituent le noyau de chaque assembleur

- un éditeur de liens permet enfin la concaténation des programmes utilisateurs assemblés séparément et qui lui sont fournis en code translatable.

Le processus de génération d'un assembleur est résumé dans le schéma suivant :



Le processus d'assemblage lui, est très classiquement :



3) Le langage GAGE

Il permet l'écriture de la description de l'assembleur. Celle-ci se présente comme une suite de déclarations, et a globalement la structure suivante :

```
TITRE <déclaration de titre>
<déclaration de type>
        -
        -
INSTRUCT <déclaration de symboles>
<déclaration d'instruction>
        -
        -
END
```

Nous retrouverons cette structure dans notre exemple du paragraphe A5-3.

Nous ne décrirons ici que les déclarations de type et d'instruction. Ces dernières ont la syntaxe suivante :

<format source> : <format objet> ;

a) Le format source des instructions :

La syntaxe du langage de description de ce format est très proche de celle d'un langage d'assemblage :

'mnémonique' (opérandes)

Exemple 1 : 'XMIT' (REGD, EXPR(IL)) : <format objet> ;

Remarques :

-1- Les mnémoniques d'instructions ayant même format source et même format objet peuvent être regroupés dans un objet de type "MNEMO" qui sera déclaré dans la liste des déclarations de type :

Exemple 2 : MNEMO INST8X : (MOVE=0, ADD=1. AND=2, XOR=3) ;

Toutes ces instructions sont "multi-format" (voir la définition des formats des types I et II du 8X300 au $ A2-3), et chacun d'eux (8 en tout : 4 sans extension et 4 avec extension) est décrit dans la liste des déclarations d'instruction, par exemple :

Exemple 3 : INST8X (REGS, EXPR(RL), REGD) : <format objet> ;

pour une instruction ayant un format du type I (registre à registre) sans extension.

-2- Les opérandes peuvent être :

- une expression : type pré-défini EXPR
- une expression parenthèsée : type pré-défini INDIR
- une dénotation : . soit une 'chaîne de caractères'
  . soit un objet de type DENOT si les opérandes peuvent être groupés. C'est le cas des objets REGS et REGD qui sont des groupes de registres déclarés ainsi :

Exemple 4 : DENOT REGS (AUX=0, R1=1, ... R11=9, IVR=#F) ;

par exemple, pour le groupe des registres-source dans une instruction 8X300. De même pour les registres-destination REGD.

C'est aussi le cas pour les champs de l'extension :

Exemple 5 : DENOT VA3 (z=0, RI_=1, RI=9, ... OI_=7, OI=#F) ;

où VA3 représente la concaténation des champs VI et A3, avec XI_ quand VI=0 et XI quand VI=1.

- un index : . soit : X'chaîne de caractères'
  . soit un objet de type INDEX pour un groupe d'index (comme DENOT).

-3- il existe également des opérandes de type "composé" et de type "caractère", plus élaborés, et que nous n'avons pas eu à utiliser étant donné le jeu d'instruction simple du 8X300.

-4- GAGE vérifie également qu'il n'y a pas d'ambiguité dans les descriptions.

b) Le format objet des instructions :

Il permet de préciser sous quelle forme les valeurs du mnémonique et des opérandes doivent être générées.

Les valeurs des opérandes sont issues du calcul d'une expression ou de l'assembleur lui-même s'il s'agit d'un opérande appartenant à un groupe de dénotation ou d'index (voir remarque 2 du point "a").

Chaque opérande est numéroté de gauche à droite de 1 à 10 et les valeurs correspondantes sont v1, v2, ... v10 ; la valeur du mnémonique est notée v0.

Le format objet d'une déclaration d'instruction est donc composé :

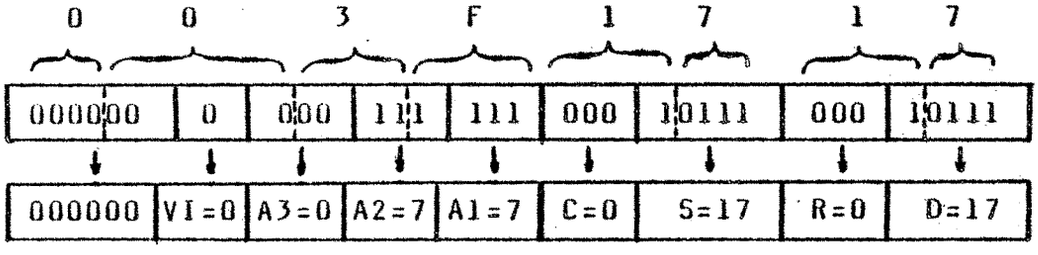
- d'affectations explicites du type "V0=5" qui impose la valeur 5 au mnémonique, ou "V(3)=7" qui donne la valeur 7 au troisième opérande (écriture FORTRAN)

- d'appels à des procédures de génération pré-définies, ou décrites parmi les déclarations de type, et qui définissent la taille des différents champs de l'instruction.

Les procédures prédéfinies BYTE et WORD permettent de générer des valeurs respectivement sur un octet et sur un mot de 16 bits, en complément à 2.

Exemple 6 : 'NOP' : (WORD (#3F) ; WORD (#1717)) ;

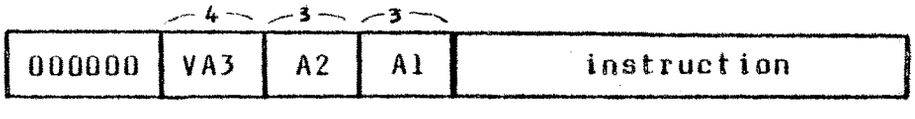
Cette description est celle de l'instuction "étendue" NOP du 8X300 (extension : "z, z, z" et instruction : "MOVE LB7, 0, LB7") ne provoquant aucune opération dans l'UCB, et dont la structure est :



On peut aussi déclarer des procédures de génération conduisant à des structures (de bits) plus élaborées.

Exemple 7 : PROC GENEX1 : ("000000", CPL2 (4), CPL2(3), CPL2(3));

Cette procédure permettra de générer le format de l'extension utilisée par l'UCB, avec VA3 sur 4 bits, A2 et A1 sur 3 bits, les 6 bits poids fort étant pré-initialisés à 0 :



(CPL2 - complément à 2 - est un des 6 "modes de génération" permis, parmi lesquels on trouve aussi : valeur absolue, adresse sur 8 ou 16 bits ...)

Exemple 8 : PROC GI8X : (CPL2(3), CPL2(5), CPL2(3), CPL2(5)) ;

Cette procédure définit les formats de type I et II (voir Annexe 2), et sera appelée dans les déclarations des instructions possédant ces formats. Par exemple, le format objet des instructions de transfert de registre à registre de l'exemple 3 sera :

WORD (#3F) ; GI8X (v0, v1, v2, v3) ;

Exemple 9 : PROC GENJMP : (CPL2(3), ADD16(13)) ;

Cette procédure définit le format de type V, c'est-à-dire celui de l'instruction de saut inconditionnel JMP, qui associé à la déclaration d'instruction :

'JMP' (EXPR) : (WORD(#3F) ; GENJMP(7,v1)) ;

conduira à l'instruction complète suivante :

←1→	←3→	←3→	←3→	←3→	←13→
000000	VI=0	A3=0	A2=7	A1=7	C=7	A=v1=valeur de EXPR

(dans ce cas l'appel à GENEXT n'est pas nécessaire, puisqu'on impose à l'extension la valeur hexadécimale 3F).

Si on prend le cas plus général d'une instruction avec extension, par exemple un transfert d'un registre vers le bus-IV, on déclarera :

Exemple 10 : INST8X (REGS, EXPR(RL), EXPR(IL), VA3, A2, A1) :
(GENEXT(v4, v5, v6) ; GI8X(v0, v1, v2, v3) ;

Lors de l'instruction : "MOVE R3, 0, LB7 z, z, BI" l'assembleur génerera la valeur hexadécimale "003D0317" correspondant au mot suivant de la mémoire de programme :

0 0 0 0 0 0	0	0 0 0	1 1 1	1 0 1	0 0 0	0 0 0 1 1	0 0 0	1 0 1 1 1
	VI	A3	A2	A1	C	S	R	D

En effet, d'après GI8X :

- MOVE, de valeur v0=0 (voir MNEMO INST8X) est généré sur les 3 bits poids fort du troisième octet (deuxième "mot" de 16 bits) de l'instruction (champ C)
- R3, de valeur v1=3 (voir DENOT REGS) est généré sur les 5 bits du champ suivant (champ S)
- le deuxième opérande, de valeur v2=0, est généré sur les 3 bits poids fort du quatrième octet (champ R)
- LB7, de valeur v3=#17 (définie par une pseudo-instruction EQU), est généré sur les 5 bits poids faible de l'instruction (champ D)

et d'après GENEXT :

- BI, de valeur v6=5 (voir DENOT A1) est généré sur les 3 bits poids faible du deuxième octet de l'extension (champ A1)
- les champs VA3 et A2 ont leur valeur v4=0 et v5=7 représentées par "z" et correspondant à NOP (voir DENOT VA3 et DENOT A2)
- tandis que les 6 bits poids forts sont pré-initialisés à 0.

Remarque : afin de réduire la taille de la description, le format objet peut être compacté grâce à :

- l'utilisation d'étiquettes dans la partie génération et d'instructions de branchement à ces étiquettes (voir le $ A5-3)
- la définition d'objets de type FORMAT, sortes de super-procédures de génération paramétrées
- l'insertion d'instructions et d'expressions FORTRAN.

Nous n'avons pas eu à utiliser ces 2 dernières facilités sauf dans "V(3)=7" par exemple.

c) Les test sémantiques sur les opérandes :

Ils sont de 2 types :

- sélection d'un format parmi plusieurs, pour une instruction source donnée (par exemple : adressage "direct" ou "étendu" dans le MC 6800 selon la valeur de l'opérande) ; il s'agit de TEST1
- élimination d'instructions ayant des valeurs ou des modes d'opérandes incorrects : TEST2.

Ces 2 objets permettent d'effectuer des calculs et des tests sur les valeurs des variables internes de l'assembleur qui sont :

- SC : numéro de la section courante
- BABS : mode de cette section (variable logique)
- CO : compteur d'emplacement
- VO : valeur associée au mnémonique
- V(I) : valeur des opérandes (I = 1 à 10)
- M(I) : mode des opérandes (I = 1 à 10)
- S(I) : section des opérandes (I = 1 à 10).

Remarque : le mode d'un opérande peut prendre 4 valeurs :

. 1 : absolu
. 2 : translatable
. 3 : externe
. 4 : erroné.

Les tests sont déclarés parmi les déclarations de type de la façon suivante :

TEST1 (ou TEST2) <nom de test> : (<description de test>) ;

Leur langage de description est constitué d'instructions : de calcul, de branchement conditionnel et inconditionnel et d'erreur, dans lesquelles interviennent des instructions et des expressions conditionnelles FORTRAN.

Nous avons donc utilisé quelques tests de validité (TEST2), par exemple celui qui vérifie que le champ R a bien une valeur comprise entre 0 et 8. Il sera déclaré par :

```
TEST2  RL :  (if "M(I).EQ.1.AND.V(I).GT.-1.AND.V(I).LT.8" go OK ;
               err (500) ; return) ;
```

Il sera utilisé dans le format source des déclarations d'instruction comme le montre l'exemple 3.

Un autre test important pour le 8X300 est celui qui permet de vérifier, dans les instructions NZT et XEC que l'adressage d'un opérande se fait bien dans la page courante (de 32 ou 256 instructions selon la longueur du champ I - formats de type III et IV). On déclare alors :

```
TEST2 M32 : (if"M(3).EQ.1.AND.BABS.AND.CO/32.EQ.V(3)/32" go MOD3;
             if "M(3).EQ.2.AND..NOT.BABS.AND.SC.EQ.S(3)" go MOD1;
             err (505) ; return ;
             %MOD1 : if "CO/32.EQ.V(3)/32" go MOD3 ;
             err (506) ; return ;
             %MOD3 : "M(3)=1" ; go OK) ;
```

Il sera utilisé par exemple dans :

```
'NZT'  (EXPR(IC), EXPR(RL), EXPR(M32))  :  <format objet> ;
```

4) Conclusion

La simplicité de ce langage a été largement vérifiée et appréciée lors de la description de notre assembleur. Celui-ci a donc pu être rapidement opérationnel.

Nous avons aussi apprécié le fait qu'il nous a généré du code translatable et a donc permis la mise au point séparée de INTCAN et INTDISP (chacun dans une "section" de 256 instructions).

A5-3 L'assembleur du 8X300

1) La description

déclaration de titre →

```
TITRE "8X300 ET EXTENSION "
```

déclarations de types

directives d'assemblage

```
PSEUDO IDF    : (XDEF=2,XREF=3);
PSEUDO VALEUR : (PAGE=4,SPACE=5,DSECT=1006,ASECT=1007,TSECT=1008);
PSEUDO CHAINE : (TITRE=9,TEXTC=1010,TEXT=1011);
PSEUDO MACRO  : (MACRO=12,FILE=13,COPY=14);
PSEUDO VIDE   : (ENDM=15,ENDF=16);
PSEUDO NORMALE: (ORG=1017,RES=1018,ORGB=1019,SET=2020,EQU=2021);
PSEUDO FIN    : (END=1);
DENOT  FONCTION :(NEG=8,ABS=9,HI=10,LO=11);
```

mnémoniques →

```
MNEMO  INST8X : (MOVE=0,ADD=1,AND=2,XOR=3);
```

dénotations

```
DENOT  REGS   : (AUX=0,R1=1,R2=2,R3=3,R4=4,R5=5,R6=6,OVF=8,R11=9);
DENOT  REGD   : (AUX=0,R1=1,R2=2,R3=3,R4=4,R5=5,R6=6,IVL=7,R11=9,IVR=#F);
DENOT  VA3    : (Z=0,RI_=1,RI=9,MII_=2,MII=#A,SOS_=3,SOS=#B,
                 SI_=4,SI=#C,AI_=5,AI=#D,STI_=6,STI=#E,OI_=7,OI=#F);
DENOT  A2     : (TO=0,BO=1,STA=2,RETI=3,SUPRQ=4,RQST=5,VSOS=6,Z=7);
DENOT  A1     : (DATA=0,ADHI=1,ADLO=2,RZBI=3,TI=4,BI=5,ECR=6,Z=7);
```

tests

```
TEST2  RL     : (IF"M(1).EQ.1.AND.V(1).GT.-1.AND.V(1).LT.8" GO OK;
                 ERR(500); RETURN);
TEST2  IC     : (IF"M(1).EQ.1.AND.V(1).GT.-2.AND.V(1).LT.32" GO OK;
                 ERR(501); RETURN);
TEST2  IL     : (IF"M(2).EQ.1.AND.V(2).GT.-2.AND.V(2).LT.256" GO OK;
                 ERR(502); RETURN);
TEST2  M256   : (IF"M(2).EQ.1.AND.BABS.AND.CO/256.EQ.V(2)/256" GO MOD2;
                 IF"M(2).EQ.2.AND..NOT.BABS.AND.SC.EQ.S(2)" GO MOD1;
                 ERR(503); RETURN;
                 %MOD: IF"CO/256.EQ.V(2)/256" GO MOD2;
                 ERR(504); RETURN;
                 %MOD2: "M(2)=1"; GO OK);
TEST2  M32    : (IF"M(3).EQ.1.AND.BABS.AND.CO/32.EQ.V(3)/32" GO MOD3;
                 IF"M(3).EQ.2.AND..NOT.BABS.AND.SC.EQ.S(3)" GO MOD1;
                 ERR(505); RETURN;
                 %MOD1: IF"CO/32.EQ.V(3)/32" GO MOD3;
                 ERR(506); RETURN;
                 %MOD3: "M(3)=1"; GO OK);
```

procédures de génération

```
PROC GENEXT   : ("000000",CPL2(4),CPL2(3),CPL2(5));
PROC GI8X     : (CPL2(3),CPL2(5),CPL2(3),CPL2(5));
PROC GMODULO  : (CPL2(3),CPL2(5),CPL2(8));
PROC GENJMP   : (CPL2(3),ADD16(13));
```

déclaration des symboles →

```
INSTRUCT (4,;,',<,>,PERLUET,*, )
```

déclarations d'instructions

```
INST8X (REGS,EXPR(RL),REGD)              :(%E1:WORD(#3F);GI8X(V0,V1,V2,V3));
INST8X (EXPR(IC),EXPR(RL),REGD)          : GO E1;
INST8X (REGS,EXPR(RL),EXPR(IC))          : GO E1;
INST8X (EXPR(IC),EXPR(RL),EXPR(IC))      : GO E1;
INST8X (EXPR(IC),EXPR(RL),REGD,VA3,A2,A1) : (%E2:GENEXT(V4,V5,V6);
                                             GI8X(V0,V1,V2,V3));
INST8X (REGS,EXPR(RL),EXPR(IC),VA3,A2,A1) : GO E2;
INST8X (EXPR(IC),EXPR(RL),EXPR(IC),VA3,A2,A1) : GO E2;
INST8X (REGS,EXPR(RL),'IVL',VA3,A2,A1) :     ("V(3)=7"; GO E2);
'XMIT' (EXPR(IC),EXPR(RL),EXPR(IC))      : ("V0=6"; GO E1);
'XMIT' (REGD,EXPR(IL))                   : ("V0=6";
                                           %E3:WORD(#3F);GMODULO(V0,V1,V2));
'XMIT' (EXPR(IC),EXPR(RL),EXPR(IC),VA3,A2,A1): ("V0=6"; GO E2);
'XMIT' (REGD,EXPR(IL),VA3,A2,A1)         : ("V0=6";%E4:GENEXT(V3,V4,V5);
                                             GMODULO(V0,V1,V2));
'XEC'  (EXPR(IC),EXPR(RL),EXPR(M32))     : ("V0=4"; GO E1);
'XEC'  (REGS,EXPR(M256))                 : ("V0=4"; GO E3);
'XEC'  (EXPR(IC),EXPR(RL),EXPR(M32),VA3,A2,A1):("V0=4"; GO E2);
'XEC'  (REGS,EXPR(M256),VA3,A2,A1)       : ("V0=4"; GO E4);
'NZT'  (EXPR(IC),EXPR(RL),EXPR(M32))     : ("V0=5"; GO E1);
'NZT'  (REGS,EXPR(M256))                 : ("V0=5"; GO E3);
'NZT'  (EXPR(IC),EXPR(RL),EXPR(M32),VA3,A2,A1):("V0=5";GO E2);
'NZT'  (REGS,EXPR(M256),VA3,A2,A1)       : ("V0=5"; GO E4);
'JMP'  (EXPR)                            :  (WORD(#3F); GENJMP(7,V1));
'NOP'  (VA3,A2,A1)                       :  (GENEXT(V1,V2,V3);WORD(#1717));
'NOP'                                    :  (WORD(#3F); WORD(#1717));
END
```

Remarques :

-1- En tête de liste, on note la présence d'un certain nombre de directives d'assemblage, communes à pratiquement tous les assembleurs

-2- La déclaration des symboles, après le mot clef INSTRUCT, permet de définir la taille du mot de la mémoire de programme (ici 4 octets). On trouve aussi le symbole de début de commentaires (;), le délimiteur de chaîne de caractères (') et le symbole représentant le compteur d'emplacement (*)

-3- Les 8 déclarations des instructions XEC et NZT auraient pu être ramenées à 4, en ayant déclaré un autre objet de de type MNEMO :

```
MNEMO   XNZ   (XEC = 4, NZT = 5)  ;
```

2) Exemples d'utilisation de l'assembleur et de l'éditeur de liens

```
ass8x rw-5
02:01:46 ASSEMB                                    détection d'erreurs
NUMASS= XX                                          ←
02
$$$$$$$$$$$$$$$$ 197    *** E300 *** CO=0082
 197    *** E200 *** CO=0082
 210    *** E300 *** CO=008C
 210    *** E200 *** CO=008C
$$$$ 474    *** E104 *** CO=00F6

    5 ERREUR(S) ARRET EXECUTION
$
OBJET SUR RW-5 ATEXT
R; T=47.01/49.29 02:02:59
```

```
ass8x rw-5
02:08:51 ASSEMB
NUMASS= XX
02
$$$$$$$$$$$$$$$$$$$$$$

    0 ERREUR(S) ARRET EXECUTION

OBJET SUR RW-5 ATEXT
R; T=46.45/48.71 02:09:58
lnk8x rw-5 rw-7
02:10:28 LINK
$$$
R; T=7.69/8.32 02:10:39
```

assemblage correct et édition de liens →

3) Extrait du listing du logiciel INTCAN

```
ATELIER MICRO - ASSEMBLEUR 8X300 (V3.15/02/79).  INTERFACE CANAL                    PAGE    1

PC     SC M CODE OBJET          LIGNE  PROGRAMME SOURCE

                                  2            ASECT 2
                                  3  *
                                  4  **********************************************
                                  5  *                                            *
                                  6  *              DEUXIEME    PARTIE            *
                                  7  *                                            *
                                  8  *        GESTION DE L'INTERFACE CANAL        *
                                  9  *                                            *
                                 10  **********************************************
                                 11  *
                                 12            CRGB  H'0100'
                                 13  *
                                 14  ***** CISS  (CHANNEL INITIAL SELECTION SEQUENCE)
                                 15  *
0100   2   00178022              16  TESTOCUP    NZT   OCCUPE,1,CUBS     Z,STA,Z          ;TEST ETAT OCCUPE
0101   2   003FE109              17                JMP   CISS                             ;OCCUPE=0,REPONSE NORMALE
                                 18  *
                                 19  ***** CUBS  (CONTROL UNIT BUSY SEQUENCE)
                                 20  *
0102   2   003FC3EF              21  CUBS        XMIT  R3,BUSY                          ;OCCUPE=1,REPONSE ABREGEE
0103   2   00170520              22                XMIT  SEL,1,0          Z,STA,Z       ;SEL=1
0104   2   003D0317              23                MOVE  R3,0,LB7         Z,Z,BI        ;
                                 24  *
0105   2   03848627              25                NZT   SLO,1,*+2        STI,TO,TI     ;STI=1, ATT SLO=0
0106   2   003FE105              26                JMP   *-1                            ;
                                 27  *
0107   2   003FC900              28                XMIT  R11,0
0108   2   003FE178              29                JMP   DECONNECT
                                 30  *
0109   2   03D40520              31  CISS        XMIT  SEL,1,0          OI,STA,TI     ;OI=SEL=1
010A   2   00078520              32                NZT   AO,1,*+3         Z,TO,Z        ;ATT AO=0
010B   2   0007862E              33                NZT   SLO,1,DISC       Z,TO,Z        ;TEST SLO=0
010C   2   003FE10A              34                JMP   *-2
010D   2   003FE117              35                JMP   PRESAER
                                 36  *
010E   2   00398336              37  DISC        NZT   REQ,1,DISC1      Z,Z,ADHI      ;SI REQ=1:INTDISC
010F   2   003FA9C8              38                NZT   R11,INTDISC                    ;SI R11=0:INTDISC
0110   2   00170521              39                XMIT  SEL,1,1          Z,STA,Z       ;SEL=0
0111   2   00181717              40                NOP                    Z,RZTI,RZBI   ;RAZ TAG IN,BUS IN
0112   2   00078534              41                NZT   AO,1,*+2         Z,TO,Z        ;ATT AO=0
0113   2   003FE112              42                JMP   *-1
0114   2   00371717              43                NOP                    Z,VSOS,Z      ;PROP SLO
0115   2 (R) 003FE000              44                JMP   DISPATCH
                                 45  *
0116   2   003FE1C8              46  DISC1       JMP   INTDISC
```

adresse

code généré

numéro de la section

extension

instruction

mnémoniques

opérandes de l'instruction

opérandes de l'extension

commentaires

Référence externe (DISPATCH)

ANNEXE 6

UTILISATION DU MATERIEL D'AIDE AU DEVELOPPEMENT D'APPLICATIONS MICROPROGRAMMEES

A6-1 Introduction

En même temps que GAGE nous avons eu la chance de pouvoir disposer du système MADAM, conçu par Mrs G.BAILLE, J.LAURENT et J.P.SCHOELLKOPF au sein de l'Equipe de Recherche en Architecture des Ordinateurs, qui venait d'être opérationnel (et de plus dans une version "portable").

Ce matériel est un outil de mise au point de toute "application microprogrammée", définie alors comme étant "tout système électronique qui a besoin de lire des instructions dans une mémoire de programme".

Le système MADAM fonctionne donc essentiellement comme un "simulateur de ROM" :

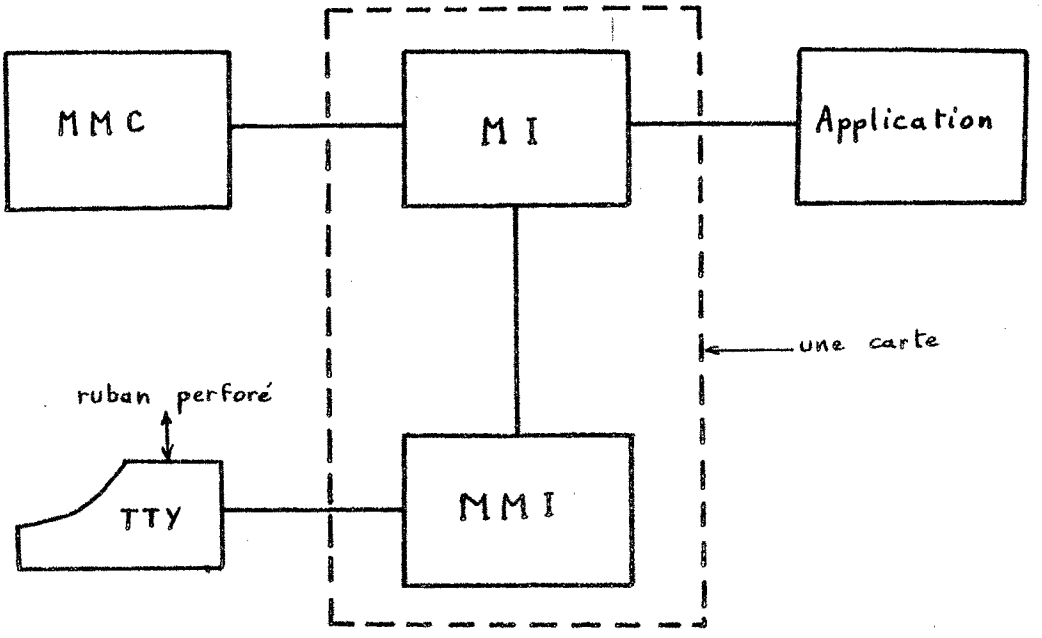
- l'application envoie l'adresse de l'instruction suivante
- MADAM lui délivre alors cette (micro)instruction.

Le système offre en outre les moyens informatiques et électroniques nécessaires à l'édition, la modification et l'exécution du programme.

A6-2 Présentation succinte de MADAM <MAD>

1) Caractéristiques matérielles

MADAM est composé de 3 modules interconnectés de la façon suivante :



- Le Module Microprocesseur (MMI)

Il exécute le programme qui interprète les commandes de MADAM. Ces commandes sont fournies au système par l'intermédiaire d'un "Télétype" muni d'un lecteur-perforateur de ruban.

Ce module est organisé de façon classique autour du microprocesseur MC 6800 , et contient :

- 2 K.octets de mémoire de programme
- 128 octets de mémoire vive
- un interface série (boucle de courant ou V24) pouvant fonctionner à 110 ou 300 bds
- les circuits nécessaires au décodage des adresses, à la gestion des priorités, à l'amplification des bus, etc...

- Le Module Mémoire de Contrôle (MMC)

C'est une mémoire RAM de 4096 mots de 48 bits dans sa version actuelle, décomposée en "cartes" de 512 mots.

Chaque circuit élémentaire est un circuit de 256 fois 1 bit, du type Intel 3106 ou AMD 27LS00, ayant un temps d'accès de l'ordre de 60 ns.

- Le Module Interface (MI)

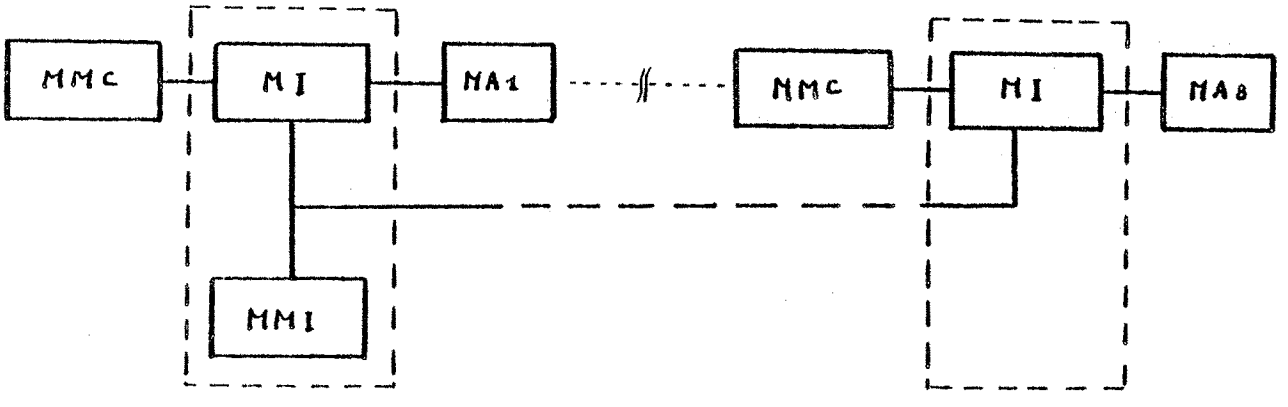
Il assure la liaison entre les 2 modules précédents et une application. Il contient :

- une mémoire RAM de 128 octets contenant la table de description du format des (micro)instructions
- les registres d'échange des données (instructions) avec la mémoire de contrôle
- les registres contenant l'adresse de l'instruction en cours (lecture, chargement) et celle de l'instruction suivante, fournie par l'application
- un compteur de cycles
- les moyens électroniques nécessaires au contrôle de l'exécution du programme de l'application (validation des points d'arrêt, marche/arrêt, arrêt sur compteur, "reset" de l'application...)

2) Extension du système MADAM

L'architecture précédente permet de contrôler jusqu'à 8 applications avec un seul module MMI.

On a alors la configuration suivante :



Le module MI d'une application est sélectionné par le module MMI grâce à un numéro de page (ou d'environnement) positionné par la commande : "E = i".

3) Caractéristiques logicielles

Il s'agit des différents moyens informatiques (commandes) qui permettent à l'utilisateur :

- d'éditer et de modifier le programme de l'application :

  . "F" définit le format de l'instruction (on peut avoir jusqu'à 21 champs de 8 bits nommés par 4 caractères maximum)
  . "I" permet d'entrer les valeurs hexadécimales de ces différents champs
  . "P" imprime le contenu du programme entre 2 adresses.

- de contrôler son exécution :

  . "R" ré-initialise l'application
  . "G" lance l'exécution avec ou sans point d'arrêt, en spécifiant ou non un nombre de cycles
  . "H" arrête l'exécution
  . "K" et "S" permettent respectivement de poser et de supprimer des points d'arrêt
  . "Z" remet à 0 le compteur de cycles
  . "C" positionne des "clés" destinées à l'application
  . "V" permet de visualiser des "voyants" fournis par l'application.

- de charger la mémoire de (micro)programme :

  . "L" lit un ruban perforé généré par un assembleur croisé ou par MADAM lui-même (voir plus loin)

- de perforer un ruban :

  . "D" perfore un ruban dans un format voisin du format "Motorola"
  . "O" perfore un ruban destiné à un programmateur de PROM (type Data I/O).

Remarques :

-1- MADAM détecte certaines erreurs qui peuvent se produire pendant les commandes F et I (bornes et valeurs d'un champ trop grandes, champ inconnu, etc ...), ainsi que lors de la lecture d'un ruban grâce à un octet de contrôle de parité longitudinale ("checksum").

-2- On trouvera à la fin de cette annexe quelques illustrations de ces commandes.

A6-3 MADAM comme mémoire de programme de l'UCB

1) Capacité utilisée

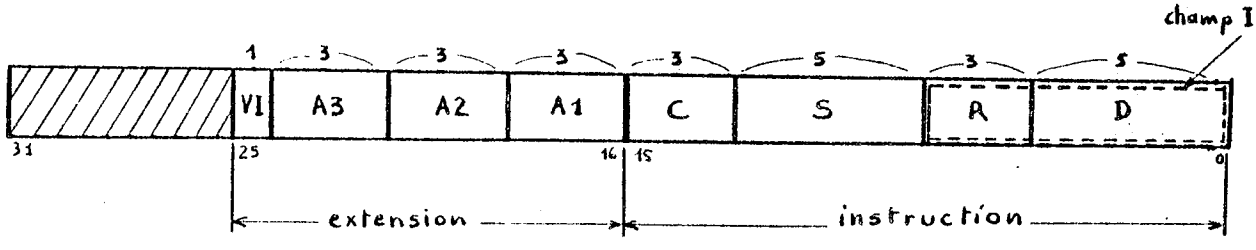
Comme nous l'avons vu au Chapitre IV, l'application de l'UCB à PIAR a nécessité une mémoire de programme de 512 mots de 26 bits. Nous avons donc réduit le module MMC à 512 mots de 32 bits.

2) Découpage de l'instruction

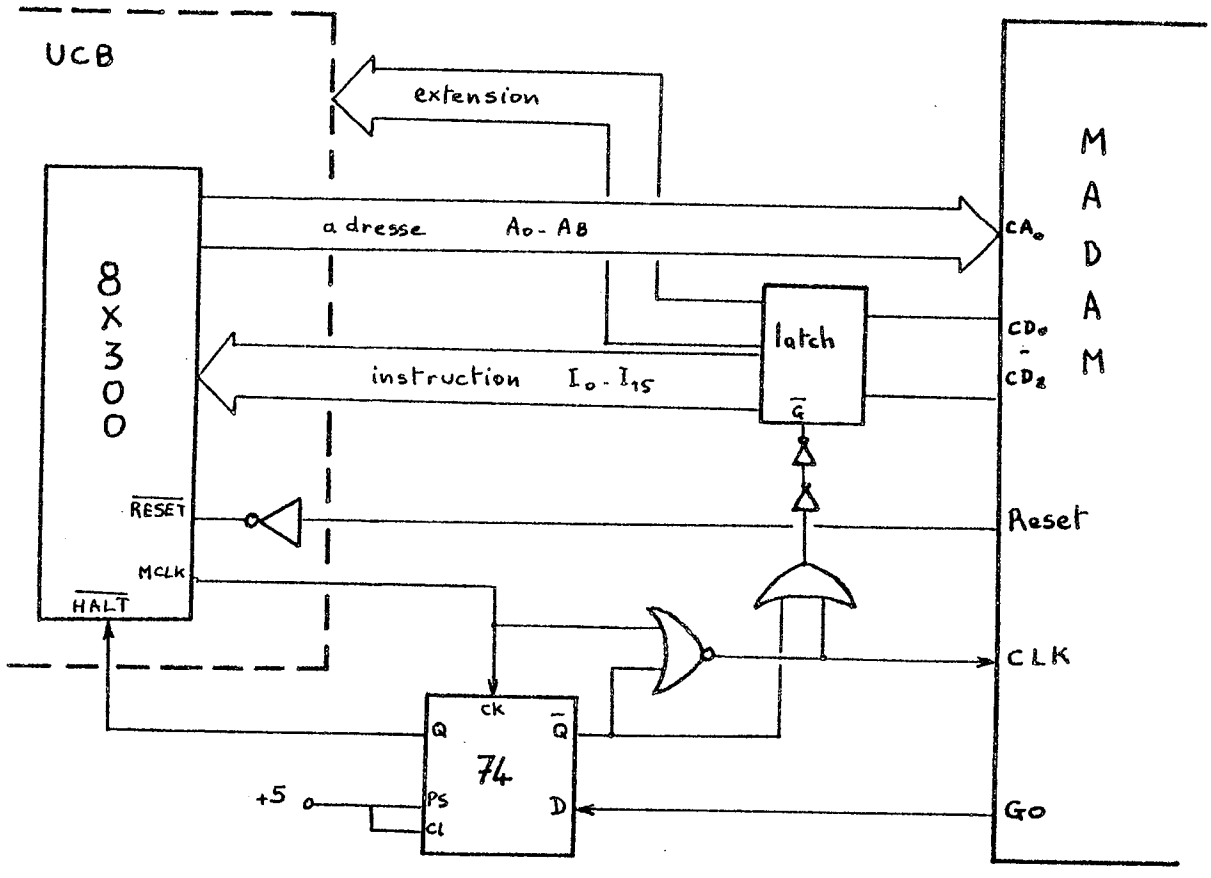
Le paragraphe 3 de l'annexe 2 décrit le jeu d'instructions du 8X300 ainsi que les différents types de formats.

En ce qui concerne le format de l'extension de ces instructions, il a été décrit dans les paragraphes 3-1 et 3-4 du chapitre III.

Nous rappellerons donc seulement les noms donnés dans MADAM aux différents champs issus du découpage des 32 bits du mot mémoire du module MMC.

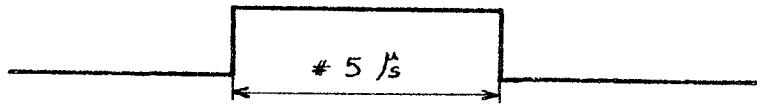


3) Liaison matérielle entre MADAM et l'UCB



MADAM fournit :

- le signal Reset positionné par la commande R et qui nécessite une inversion puisqu'il est de la forme :

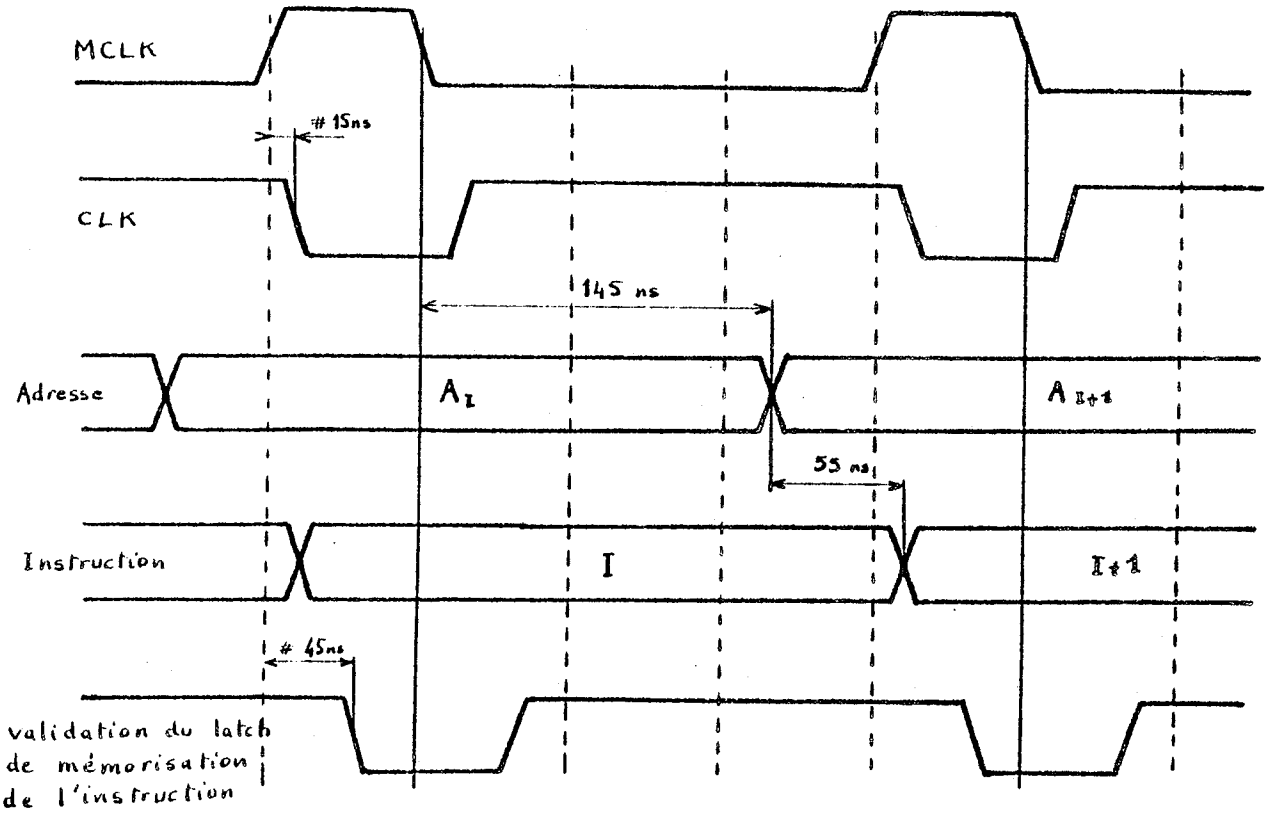


- le signal Go qui permet l'arrêt (et le lancement) du programme. Il est positionné à 0, dans MADAM, par la commande H, le passage sur un point d'arrêt ou le passage à 0 du compteur de cycles.

Ce signal est utilisé par l'UCB, d'une part pour générer le signal HALT pour le 8X300 et d'autre part pour inhiber le signal CLK que l'application doit fournir à MADAM pour rythmer l'accès au module MMC ainsi que pour synchroniser le contrôle de l'exécution.

Remarque : les entrées $\overline{RESET}$ et $\overline{HALT}$ du 8X300 sont également positionnées respectivement par l'Interface Canal et l'Interface Dispositif comme nous l'avons vu aux chapitres III et IV.

4) Chronogramme des accès à la mémoire de programme



A6-4 Exemples d'utilisation des commandes de MADAM

1) Définition des formats

[illegible] ← travail dans l'environnement "0"

[illegible]

2) Remise à 0 du compteur de cycles
Execution d'un pas de programme

```
 A=000 C=F002
*?
*H
 A=000 C=0000
*G A=X N=1
*
*** I=0 A=001 N=FFFF
```

{ A = adresse de l'instruction suivante
C = valeur du compteur de cycles

3) Chargement, suppression et pose de points d'arrêt
Lancement du programme avec validation des points d'arrêt

```
*L
*?
*?
*S 000 1FF
*H
 A=001 C=0000
*K 000 013
*G A=X K

*** I=0 A=014 N=040F
*G A=X
*H
 A=078 C=826C
G A=X K

*** I=0 A=001 N=0311
*G A=X K

*** I=0 A=014 N=071F
```

fin du chargement

passage à l'adr. 13

points d'arrêt non validés

passage à l'adr. 0

passage à l'adr. 13

4) Entrée des valeurs de certains champs

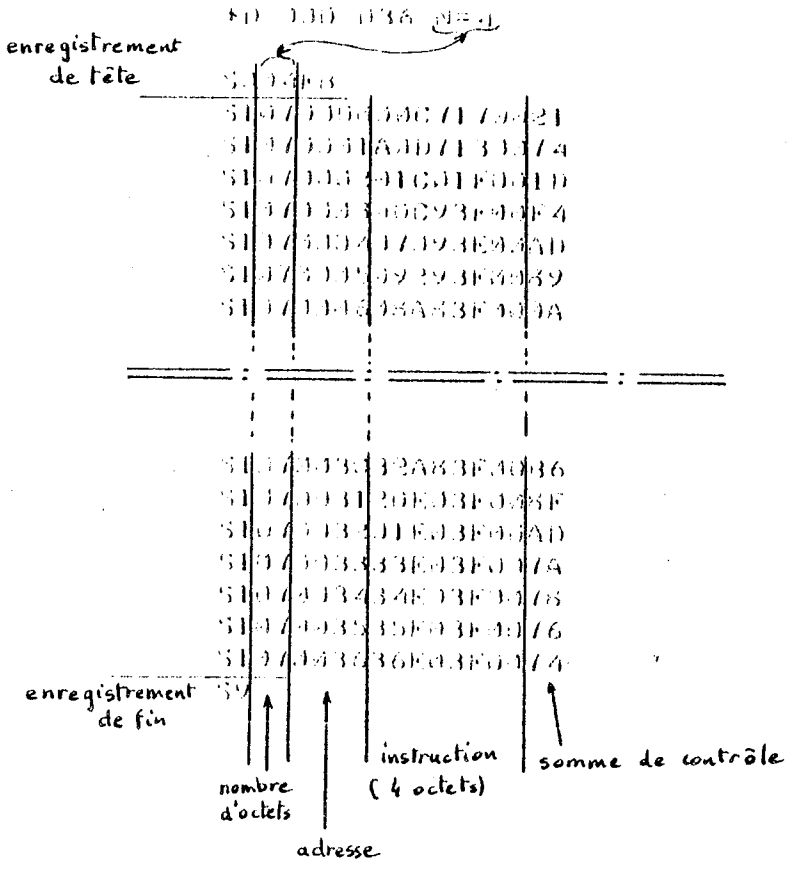
```
*I 049
C=07;(N)
049
S=03,D=0F;N
04A
A2=01,S=17,D=1F;N
04B
A3=04,A2=00,A1=04,C=05,S=14,R=01,D=0D;N
04C
C=07,I=49;N
04D
A3=07,A2=02,A1=04,C=06,S=15,R=01;
```

passage à l'instruction suivante

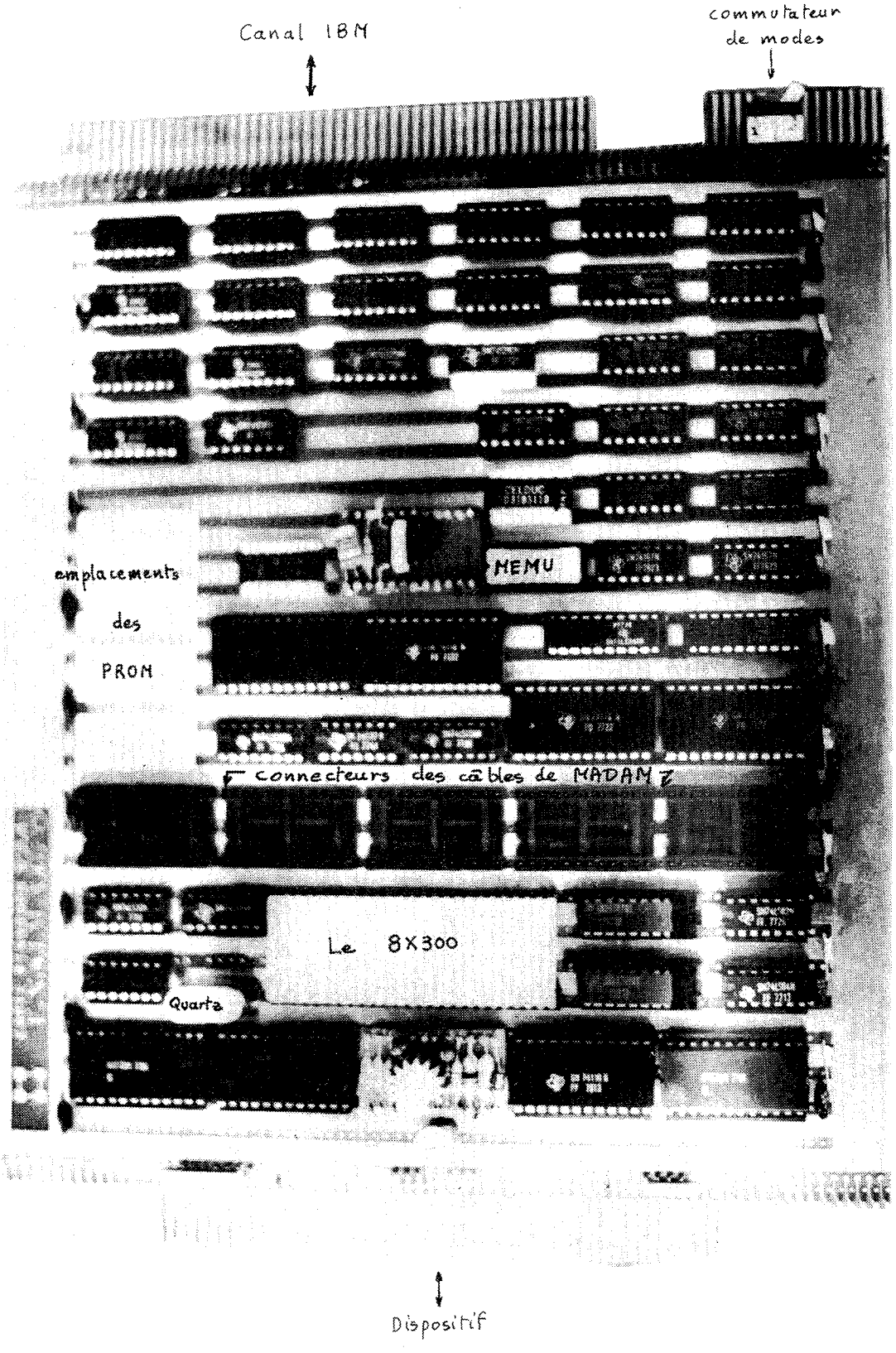
5) Impression du programme
Modification des champs erronés

```
P 000 136
000 VI=44 A3=00 A2=02 A1=07 C=06 S=07 R=00 D=04 I=00
001 VI=00 A3=14 A2=02 A1=03 C=06 S=17 R=05 D=04 I=00
002 VI=14 A3=00 A3=03 A1=07 C=06 S=00 R=00 D=01 I=01
003 VI=01 A3=10 A2=17 A1=07 C=06 S=09 R=00 D=00 I=00
→ 004 VI=05 A3=00 A2=07 A1=06 C=00 S=07 R=04 D=09 I=09
005 VI=04 A3=00 A2=07 A1=07 C=01 S=09 R=03 D=09 I=09
006 VI=00 A3=00 A2=07 A1=07 C=05 S=08 R=00 D=08 I=08

↓
S1 004
S=09, D=07;

→ MP 004 004
004 VI=00 A3=01 A2=07 A1=06 C=00 S=09 R=04 D=07 I=07
```

6) Perforation d'un ruban

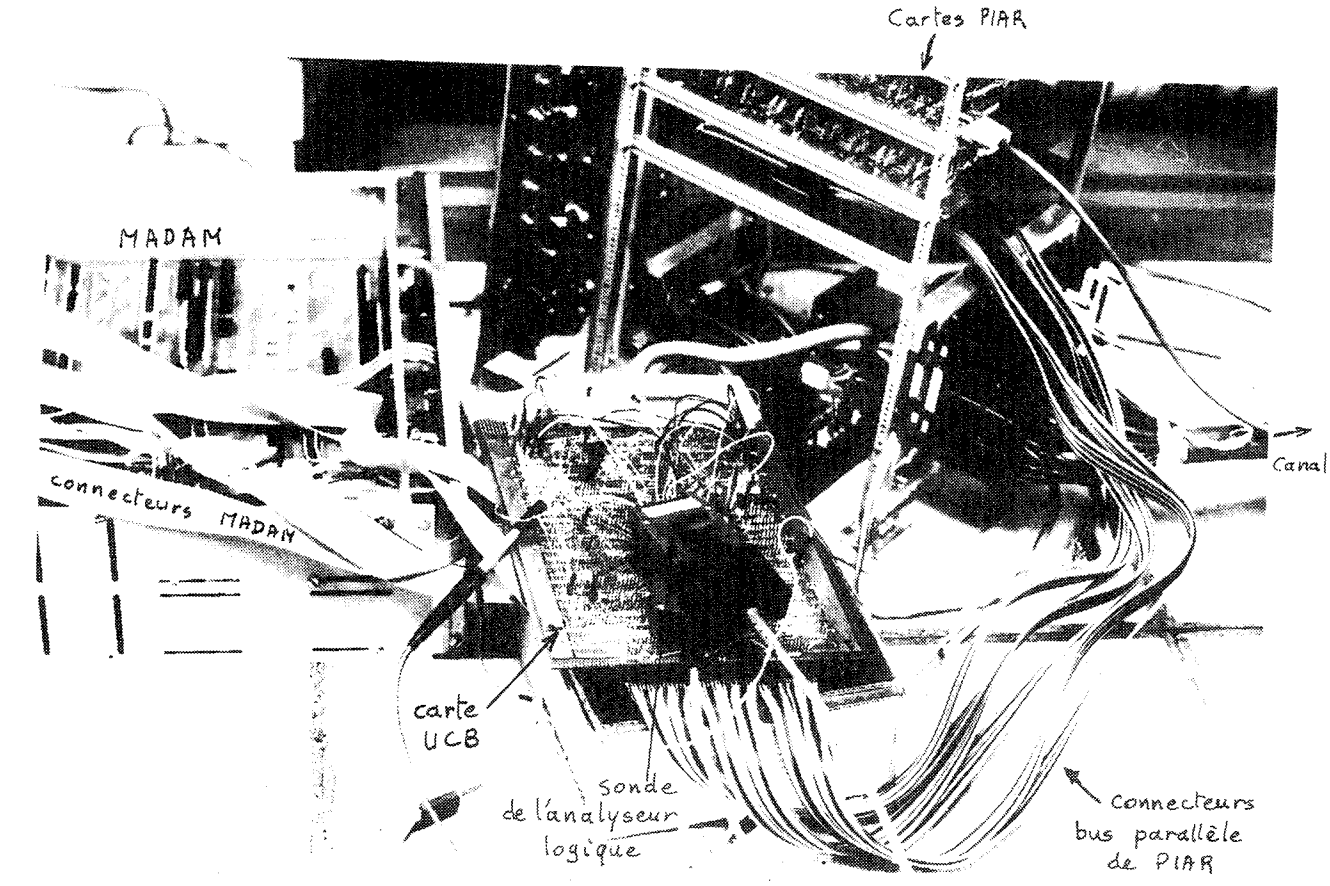


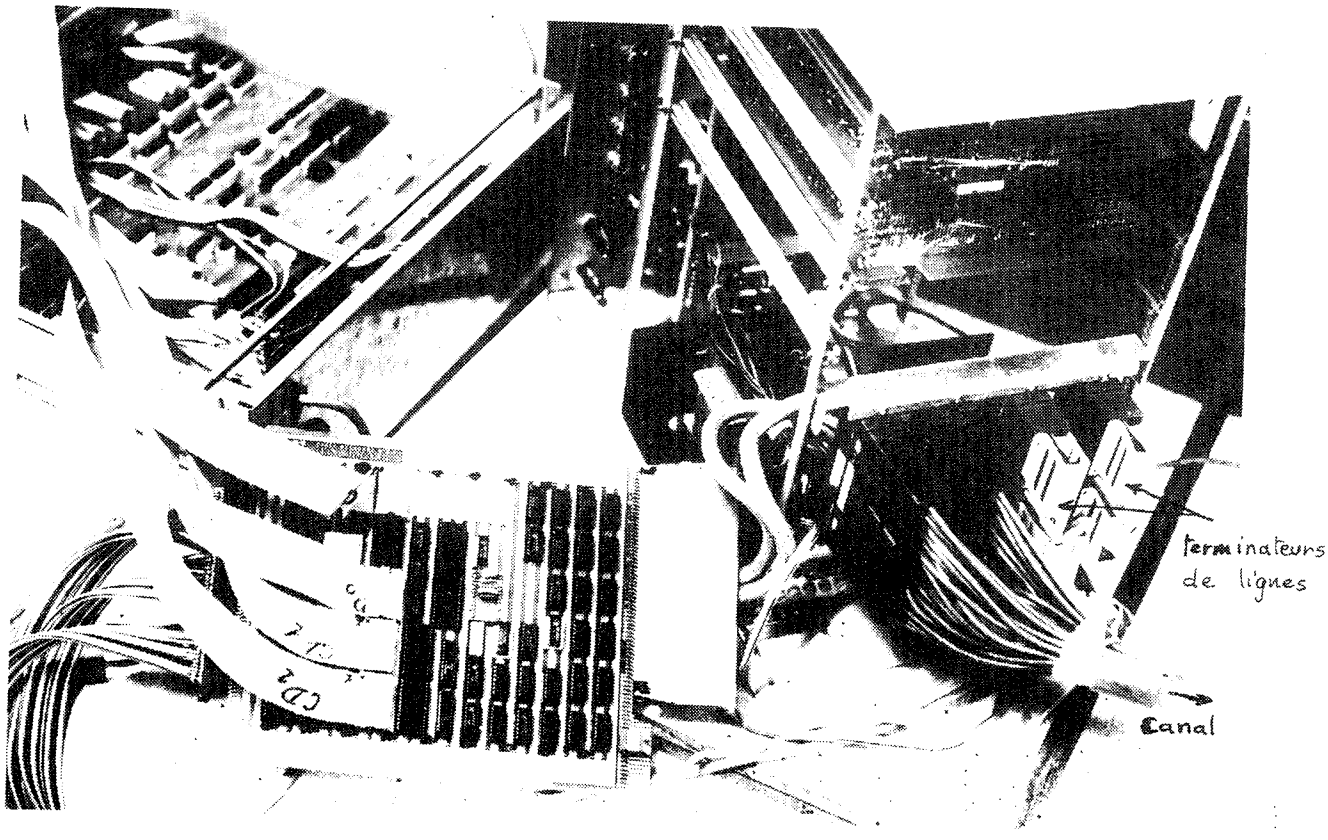
PHOTOS - SOUVENIRS



La carte de l'UCB

Vue d'ensemble du matériel pendant la mise au point





Vues détaillées des connexions MADAM - UCB - PIAR - Canal IBM 360

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, Maître de Conférences à l'Institut National Polytechnique de GRENOBLE
- G. NOGUEZ, Professeur à l'Université de PARIS VI

Monsieur Serge A R N A U D

est autorisé à présenter une thèse en soutenance pour l'obtention du diplôme de DOCTEUR-INGENIEUR, spécialité "Génie Informatique".

Grenoble, le 9 Octobre 1979
Le Président de l'I.N.P.G.

Ph. TRAYNARD
Président
de l'Institut National Polytechnique