THÈSE

présentée à

UNIVERSITÉ DES SCIENCES ET TECHNIQUES DE LILLE

pour obtenir le titre de

DOCTEUR-INGÉNIEUR

par

Kamal QUOTB

Ingénieur I.S.E.N.

"SYSTEMES MULTI-AUTOMATES PROGRAMMABLES : REALISATION MATERIELLE ET PROGRAMMATION A PARTIR D'UNE DESCRIPTION GRAFCET"

Soutenue le 23 Juin 1981 devant la Commission d'Examen :

MM.	Pierre VIDAL	Président
	Jean-Marc TOULOTTE	Rapporteur
	Georges MANESSE	Examinateur
	Etienne DEFFONTAINES	Examinateur
	André LANDAS	Invité

الى جدتي .

ان العلم اداة لبناء العالم ولا
لتخريبه .

A la mémoire de ma Grande-Mère

La science doit être utilisée pour la construction et non pour la destruction du monde.

AVANT - PROPOS

Le travail présenté dans ce mémoire a été effectué au Centre d'Automatique de l'Université des Sciences et Techniques de Lille 1.

J'adresse ici toute ma reconnaissance à Monsieur le Professeur Pierre VIDAL, qui a su nous soutenir et nous aider dans les moments difficiles.

Que Monsieur Jean-Marc TOULOTTE, Professeur à l'Université des Sciences et Techniques de Lille 1, soit assuré de nos remerciements pour l'intérêt qu'il nous a témoigné pour ce travail et pour les conseils qu'il nous a prodigués tout au long de cette recherche.

Je tiens à remercier Monsieur Etienne DEFFONTAINES, Professeur à l'Institut Supérieur d'Electronique du Nord, pour l'honneur qu'il nous fait en acceptent de participer à ce jury.

Je remercie également Monsieur Georges MANESSE, Maître-Assistant à l'Université des Sciences et Techniques de Lille 1, Laboratoire d'Electrotechnique, pour sa participation à ce jury.

Que Monsieur LANDAS, Responsable Technique à la Société LECQ Automation, trouve ici l'expression de mes remerciements, pour avoir bien voulu participer à ce jury de thèse.

Mes remerciements vont aussi à Monsieur Bernard CEURSTEMONT pour son aide sympathique et efficace tout au long de ce travail.

Je tiens aussi à remercier Madame TIPRET-CADART Germaine pour la réalisation dactylographique de ce mémoire.

Enfin je ne saurai terminer cet avant-propos sans remercier l'ensemble du personnel technique et administratif du Centre d'Automatique pour sa collaboration.

S O M M A I R E

...

o

o o

I N T R O D U C T I O N

La logique industrielle a traditionnellement la réputation de ne pas nécessiter de performance élevée en vitesse. Les automates à déroulement cyclique de la mémoire ont, effectivement, des temps de réponse allant jusqu'à 200 ms, pour les plus lents.

Toutefois, la complexité croissante des automatismes abordés nécessite d'une part, la réalisation d'un nombre plus grand d'instructions par unité de temps et, d'autre part, une utilisation plus poussée des descriptions avec un parallélisme important .

Pour satisfaire ces nouvelles exigences, on peut essayer d'augmenter la vitesse des processeurs et d'utiliser des structures multiautomates. Le travail présenté dans ce mémoire est une exploration des possibilités d'association d'automates aussi bien au niveau matériel que logiciel.

Un premier chapitre introduit les notions importantes pour la compréhension du problème sur les automates programmables et le Grafcet pris comme outil de description.

Le deuxième chapitre précise l'architecture à réaliser, décrit le microprocesseur utilisé; sont également abordés les différents modes de parallélisme dans un Grafcet et les problèmes généraux liés aux structures multiprocesseurs

...

Le troisième chapitre aborde la présentation de la réalisation du système multi-automates et précise la composition de chaque carte automate et son fonctionnement ainsi que la carte de contrôle des échanges entre processeurs.

Dans le quatrième chapitre nous détaillons la console de programmation utilisée pour le pilotage du système ainsi que les modalités d'échange opérateur-multiautomates.

Le dernier chapitre traite des méthodes d'implantation du Grafcet sur le système multiprocesseurs. Trois structures y sont employées :

- Un automate monoprocesseur
- Un multi-automates banalisés
- Un système avec automates spécialisés.

o
o o

CHAPITRE I - LES AUTOMATES PROGRAMMABLES

I-1 ETUDE GENERALE SUR LES AUTOMATES PROGRAMMABLES

Après avoir placé notre étude dans son contexte industriel et avant de s'attacher à la description et à la réalisation de notre système, nous présentons des généralités sur la structure, la composition et l'organisation d'un automate programmable. Nous présentons également le modèle que nous avons choisi pour la représentation d'un automatisme quelconque.

I-1-1- Principe

L'automate programmable est une machine capable de recevoir les informations d'état d'un processus caractérisées par des signaux logiques ou numériques et de générer, par un programme défini à priori, les informations de commandes également par signaux logiques ou numériques.

Le programme décrit des relations booléennes et temporelles reliant les informations d'entrée et celles de sortie. Les machines ont été conçues pour remplacer des ensembles câblés de commande à relais électromécaniques ou à circuits logiques électroniques, ceci afin d'éliminer certains inconvénients que présentent de tels ensembles : manque de souplesse, mise au point longue, adaptation et modification très difficiles à réaliser en raison du caractère figé

des circuits, coût important du câblage, encombrement important, difficultés de réalisation de certaines fonctions logiques.

I-1-2 - Organisation des automates programmables

I-1-2-1 - Configuration générale d'un automate programmable [21]

Suivant les fonctions désirées, l'architecture varie d'un constructeur à l'autre, mais la configuration représentée par la figure I, nous montre les éléments fondamentaux qui constituent un automate programmable.

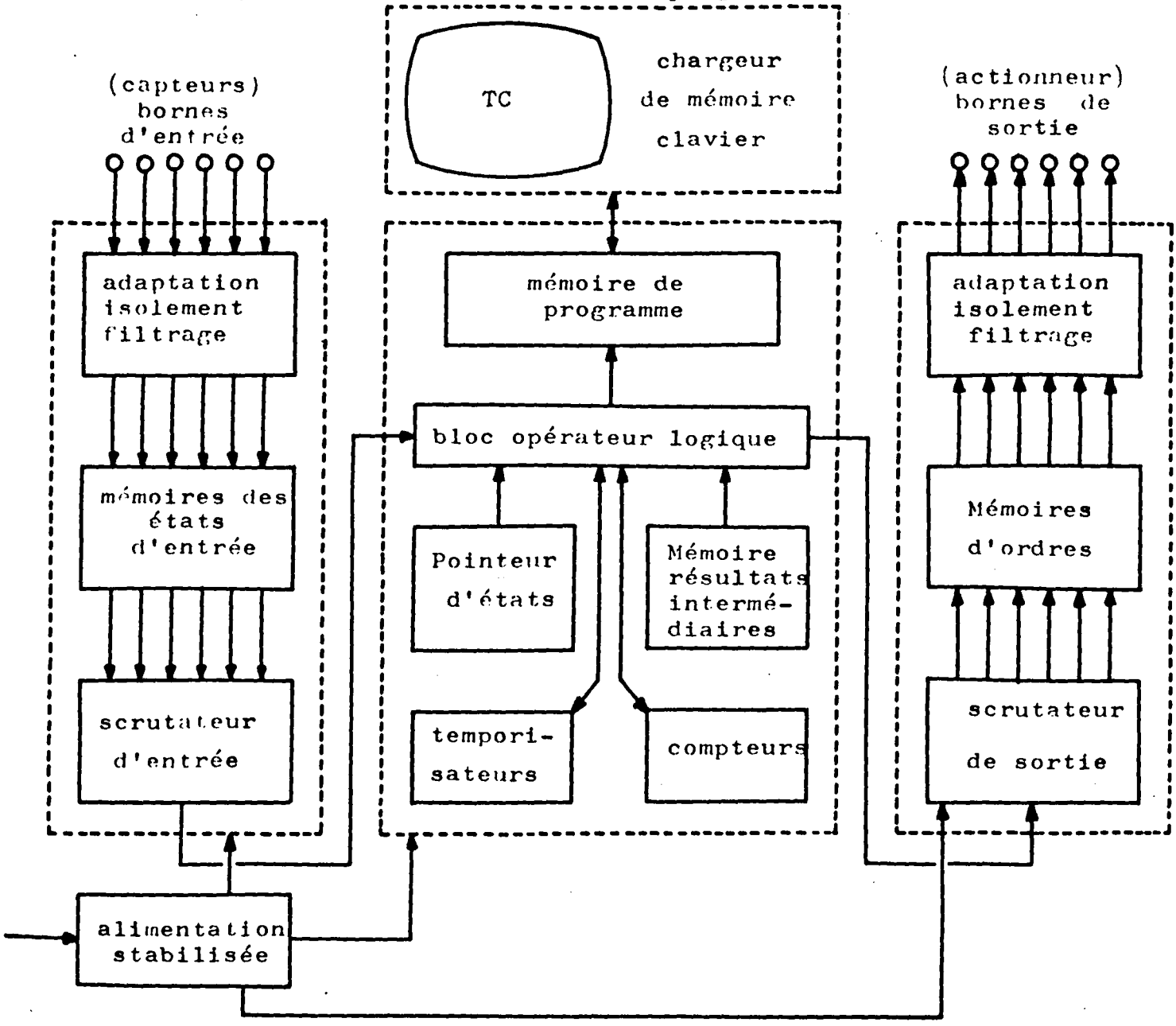


Figure I - Configuration d'un automate programmable ...

I-1-2-2- Nature et importance des entrées-sorties [21]

Les entrées-sorties, constituent un des plus importants modules de l'automate. Le nombre de ces entrées-sorties logiques en binaire (tout ou rien) varie de quelques dizaines à plus d'un millier (extension possible suivant les normes désirées). Des interfaces d'entrée et de sortie, dont le rôle est de transformer les tensions, intensités et la nature des signaux d'entrée ou de sortie, en signaux normalisés utilisés dans l'automate; la figure 2 suivante, montre ce rôle :

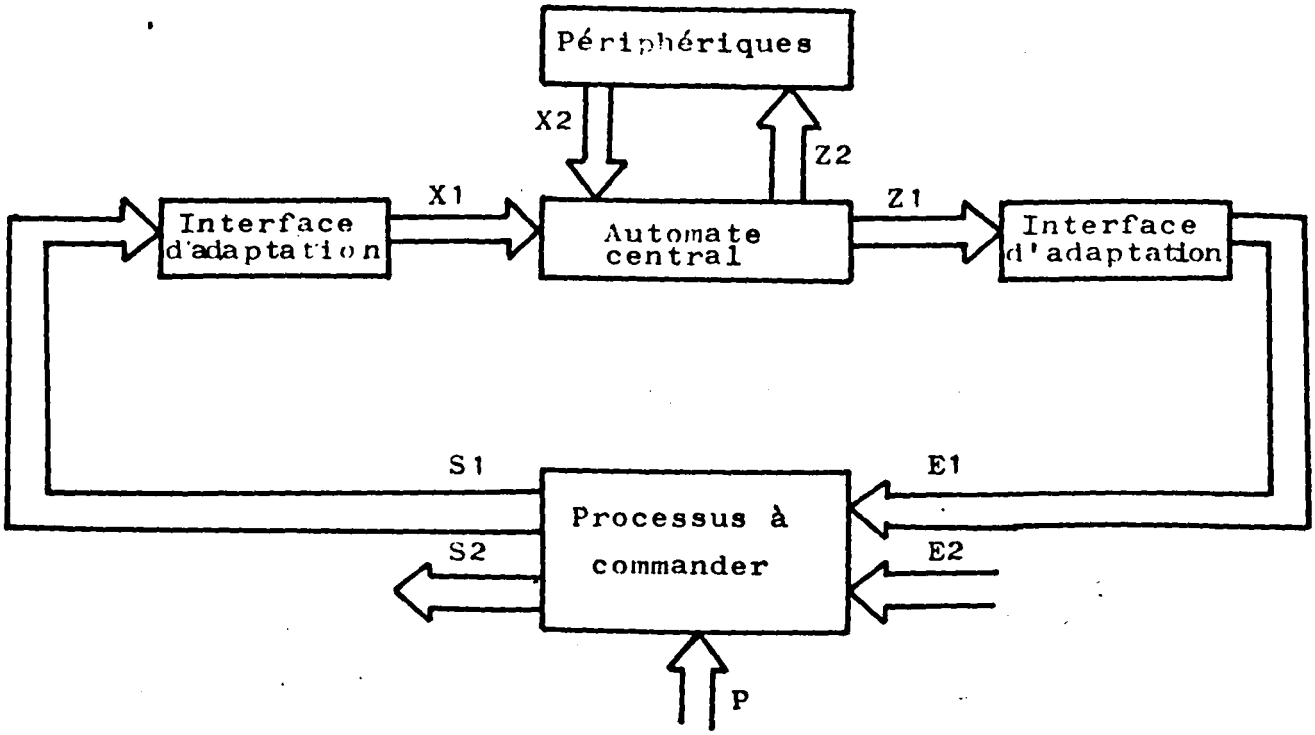


Figure 2

En outre, ces entrées-sorties, doivent être galvaniquement isolées des signaux extérieurs et être à l'abri des parasites. Ces différentes fonctions sont assurées par des filtres, photocoupleurs, transformateurs, relais électromagnétiques, miniatures ou à lames souples.

La majorité des automates sont orientés traitement sur bit. Toutefois, il est possible de trouver sur quelques-uns un traitement sur mot, soit pour des informations numériques, soit pour des signaux analogiques après conversion.

Les entrées peuvent être prises globalement en début de cycle de traitement. Elles sont alors dites synchrones. Quant aux sorties, on les affecte en général en fin de cycle de traitement.

Pour ce qui concerne le traitement, la scrutation du programme se fait, en général, d'une façon cyclique. La durée ne dépend pas toujours du nombre d'instructions dans un programme car, grâce à une instruction spéciale, dite de "saut", cette durée peut se trouver dépendante du nombre d'instructions exécutables au cours du cycle.

I-1-2-3- Unité de traitement [22]

L'unité de traitement a pour rôle de décoder et d'exécuter les instructions contenues dans la mémoire programme.

Les premières utilisent l'unité logique, ses instructions fondamentales : ET, OU, complément, etc... et les instructions spéciales : comptage, temporisation, registre de décalage , etc... Quant à l'unité numérique, elle est rarement utilisée et elle est limitée aux opérations +, -, comparaison, etc...

Les unités de traitement, dites de haute gamme, sont caractérisées par leur rapidité, leur architecture ou leurs

possibilités nouvelles pour l'implantation d'algorithmes.

I-1-2-4- Mémoire [23], [22]

La mémoire du programme contient les différentes séquences et instructions d'un processus donné. Sa capacité varie d'un modèle à l'autre suivant la capacité du programme à y enregistrer 0,25 à 16 K mots de 8 ou 16 bits.

Ces mémoires sont parfois à tores de Ferrites, souvent à semi-conducteurs, soit en RAM, soit en mémoire morte-programmable : PROM, ceci afin d'éviter leur effacement lors des coupures de courant. Le recours à une batterie de secours est une solution satisfaisante en cas d'utilisation de RAM.

I-1-2-5- Console de programmation

Généralement séparée de l'automate, elle est utilisée pour écrire le programme. C'est une interface entre l'opérateur et l'automate plus ou moins performante et remplissant des fonctions plus ou moins complexes : lecture écriture, modification des programmes. Elle comporte un clavier et un dispositif d'affichage à diodes luminescentes ou tuble cathodique et peut être connectée à une imprimante pour éditer le programme au fur et à mesure s'il est écrit au clavier.

Il y a deux types de consoles : celles dites autonomes avec une mémoire vive dans laquelle vient s'enregistrer le programme lors de son élaboration et les consoles qui ne peuvent fonctionner qu'en liaison avec l'automate car, par

économie, elles utilisent la mémoire programme de ce dernier.

Certaines consoles comportent également un enregistreur de mini cassette, bande magnétique ou ruban perforé, ce qui permet de constituer une bibliothèque de programmes.

I-1-2-6- Programmation [23]

La programmation des automates doit être assez simple et accessible à un personnel non spécialisé en informatique; elle se fait en partant des schémas logiques ou à relais et des équations booléennes. On trouve souvent des facilités tels que les parenthèses, le pas à pas, mais aussi des contraintes comme la limite du nombre des termes dans une équation logique.

Pour l'acquisition et la mise en oeuvre d'un automate, le langage de programmation choisi joue un rôle fondamental. Il convient donc de connaître clairement le type de traitement, le temps de traitement, comptage, temporisation, primitives de base et Primitives spéciales.

Nous ne pouvons laisser sous silence l'importance des Primitives spéciales et, en particulier, de l'instruction de saut. En-effet, avec celle-ci, il devient possible d'implanter avec facilité les récents outils de description des grands systèmes logiques (Réseau de Pétri, Grafcet, etc..).

I-2 - REPRESENTATION DU CAHIER DES CHARGES D'UN AUTOMATISME LOGIQUE PAR LE GRAFCET

I-2-1- Automatisme et cahier des charges [18], [1]

Avant de donner un exemple d'utilisation du GRAFCET pour la représentation d'un cahier des charges, nous étudierons, dans ce paragraphe, le contenu, les spécifications et l'élaboration d'un cahier des charges.

I-2-1-1- Système et automatisme

D'une façon générale, un système automatisé peut se décomposer en deux parties qui coopérent :

- la partie opérative : c'est le processus physique à automatiser
- la partie commande : c'est l'automatisme qui élabore en sortie des ordres destinés au processus et des sorties externes destinées à l'opérateur, calcul ou autre automatisme, en fonction des comptes rendus venant du processus et des consignes qu'il reçoit en entrée provenant du monde extérieur.

Par exemple, dans une machine-outil à commande numérique, la partie opérative est la machine-outil proprement dite et la partie commande, l'équipement de commande numérique.

La partie opérative effectue des opérations (trans-

formation des pièces brutes en pièces usinées) lorsqu'elle reçoit l'ordre de la partie commande. Grâce aux comptes rendus fournis par la partie opérative, la partie commande peut être tenue informée de l'état d'avancement des opérations ordonnées :

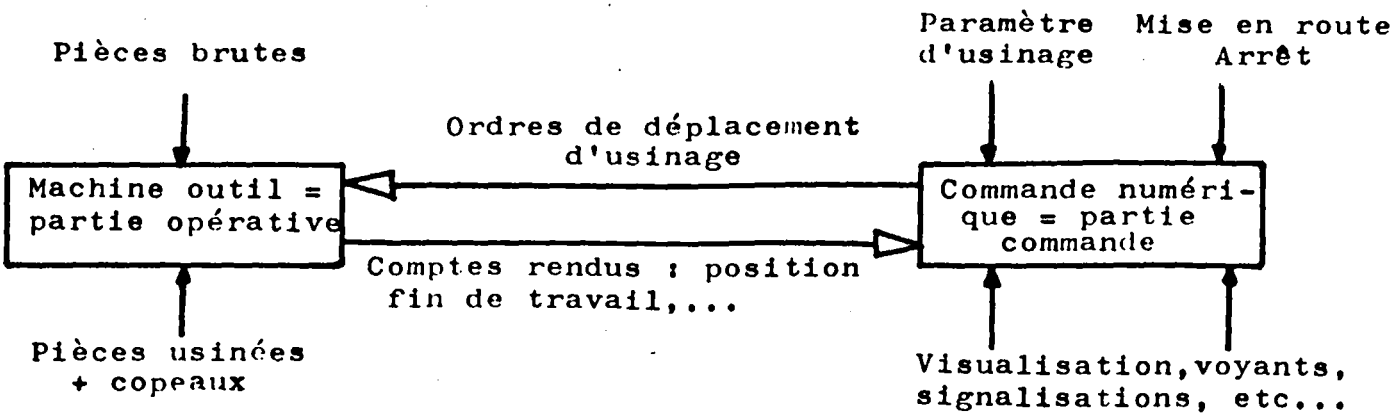


Figure 3 - Machine outil à commande numérique

La partie commande peut recevoir des paramètres d'usinage, des signaux de mise en marche, d'arrêt, etc... Nous nous limiterons aux automatismes logiques pour lesquels les informations traitées présentent un caractère "tout ou rien".

I-2-1-2- Le cahier des charges [18],[1],[

Le cahier des charges va nous décrire, d'une façon systématique et logique, les rapports existants entre la partie opérative (processus) et la partie commande (automatisme) du système à automatiser.

Pour l'élaboration de celui-ci, nous pouvons diviser le problème en deux parties :

- la première partie décrivant les spécifications technologiques, c'est-à-dire la façon avec laquelle l'automatisme

devra physiquement s'insérer dans l'ensemble que constitue le système à automatiser.

- la deuxième partie consiste à décrire les spécifications opérationnelles et fonctionnelles du système : la sûreté du fonctionnement de l'automatisme, la fiabilité absence de pannes dangereuses et aussi la définition du comportement de l'automatisme suivant l'évolution des différents paramètres du processus.

Le rôle de cette deuxième partie est surtout de faire comprendre au concepteur l'automatisme à constituer.

I-2-2 - Modèle de représentation choisi : GRAFCET [2]

Nous avons choisi, comme modèle de représentation, le GRAFCET dont nous allons rappeler succinctement, dans ce paragraphe, les bases .

I-2-3 - Définition du GRAFCET [1].[2].[19]

Le fonctionnement d'un automatisme logique peut être matérialisé par un ensemble :

- d'étapes auxquelles sont associées des ACTIONS
- de TRANSITIONS auxquelles sont associées des RECEPTIVITES
- de LIAISONS (ou ARCS) ORIENTEES, chacune reliant une ETAPE à une TRANSITION ou une TRANSITION à une ETAPE

Un GRAFCET est un graphe orienté défini par l'ensemble suivant :

$$Gr = \left(E, T, A, M_o \right) \quad \text{où}$$

E représente l'ensemble des étapes d'un GRAFCET

T représente l'ensemble des Transitions d'un GRAFCET

A représente l'ensemble des Arcs d'un GRAFCET

M_o représente le Marquega Initial du GRAFCET

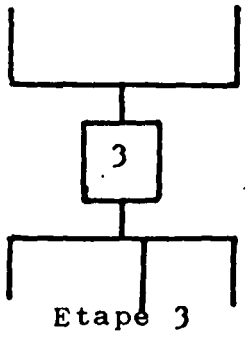
I-2-3-1- Les étapes du GRAFCET

soit l'ensemble

$E = \left(E_1, E_2 \ldots\ E_m \right)$ où $m \in N^*$ l'ensemble des entiers naturels privé de zéro

Les éléments de E sont des étapes représentées par des carrés ou des rectangles repérés numériquement [18]

Figure 4



Une étape est soit active, soit inactive.

A l'état initial on a, obligatoirement, une ou des étapes actives.

L'état d'un automatisme à un instant donné va être

défini par l'ensemble des étapes actives.

Etape 4 active Etape 6 inactive

Figure 5

Pour chaque étape, on fait correspondre des actions à effectuer : ce sont des commandes ou des fonctions, combinatoires et logiques qui font intervenir les entrées, les sorties et même l'état actif ou inactif d'autres étapes de l'automatisme logique.

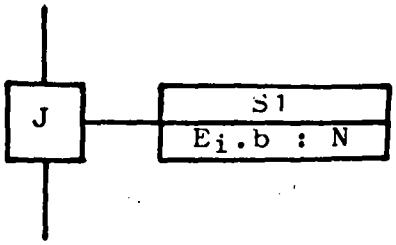


Figure 6

Dans cet exemple, à l'étape j, on fait correspondre la sortie S1 et la variable de Sortie N conditionnée par le produit booléen Ei.b, où b peut être une variable de l'automatisme et Ei une variable caractérisant l'état de l'étape i.

I-2-3-2- Les transitions d'un GRAFCET

soit l'ensemble

$$T = \{ t_1, t_2 \ldots t_q \} \text{ où } q \in N^*$$

N^* l'ensemble des entiers naturels sans zéro

les éléments de cet ensemble sont des transitions représentées par des barres.

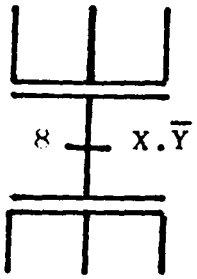


Figure 7

La transition est une phase intermédiaire obligatoire entre deux étapes. En effet, elle indique la possibilité d'évolution entre ces deux dernières.

A toute transition, on fait correspondre une réceptivité, c'est une fonction logique qui fait intervenir, outre les variables d'entrée ou les variations des variables d'entrée, les variables de sortie de l'automatisme et l'état actif ou inactif des variables représentatives de certaines étapes et des prédicats.

Ainsi la réceptivité permet de distinguer, parmi toutes les informations disponibles, uniquement celles qui sont susceptibles de provoquer un changement de comportement du Système.

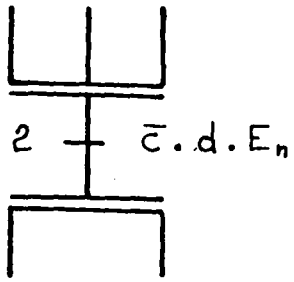


Figure - 8 -

La réceptivité de la transition 2 est caractérisée par la fonction booléenne $\bar{c}.d.E_n$ où c et d sont des variables d'entrée de l'automatisme, E_n caractérise l'état de l'activité de l'étape n; $\bar{c}$ est l'inverse de c.

La réceptivité peut aussi être une fonction de la forme $E_n.(\uparrow d)$ où E_n correspond à l'état de l'étape n et $(\uparrow d)$ correspond au passage de la variable "d" de l'état bas à l'état haut (front montant).

Par analogie, on représente le front descendant d'une variable, par $(\downarrow d)$

L'état d'une variable peut aussi représenter la réalisation d'une temporisation.

I-2-3-3- Les Arcs d'un GRAFCET [2]

soit l'ensemble

$$A = \{ a_1, a_2, a_3 \ldots\ldots a_l \} \text{ où } l \in N^*$$

N^* l'ensemble des entiers naturels sans zéro

Les éléments de cet ensemble sont des arcs orientés dont la fonction est de relier une étape à une transition ou une transition à une étape. Les liaisons sont horizontales

ou verticales.

I-2-3-4- Le marquage d'un GRAFCET

soit l'ensemble

$M = \{ m_1, m_2 \ldots m_n \}$ où $n \in N^*$ ensemble des entiers naturels sans zéro

Les éléments de cet ensemble forment ce qu'on appelle le marquage d'un GRAFCET.

m_i peut prendre 2 valeurs, une à la fois

$m_i = 0$ → étape i inactive

$m_i = 1$ → étape i active

mais l'étape i ne peut pas être à la fois active et inactive.

Graphiquement l'activité d'une étape se traduit par un point appelé marqueur à l'intérieur du cercle (voir figure 5, page I-11).

Dans le cas du marquage initial, il est indiqué par un second carré concentrique

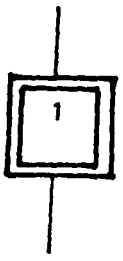


figure 10

A un marquage du GRAFCET, correspond un état de l'automatisme logique.

I-2-4 - Règles de validation et de tir d'une ou de plusieurs transitions

L'aspect évolutif d'un automatisme logique se traduit au niveau de l'outil de description GRAFCET, par une évolution séquentielle du marquage. Il est donc nécessaire de connaître les règles d'évolution d'un GRAFCET au cours d'un cycle.

Les "étapes d'entrée" d'une transition sont les étapes d'où sont issus les arcs orientés vers la transition. Les "étapes de sortie" d'une transition sont les étapes où aboutissent les arcs orientés issus de la transition.

I-2-4-1- Validation d'une transition

Une transition est validée pour un marquage M donné si toutes les étapes d'entrée sont actives.

I-2-4-2- Tir ou franchissement d'une transition [4], [2]

Une transition d'un GRAFCET peut être tirée si elle satisfait aux deux conditions suivantes :

- lorsqu'elle est validée
- si la réceptivité liée à celle-ci est vraie.

Le tir ou le franchissement d'une transition se traduit par la désactivation de toutes les étapes d'entrée et l'activation ou le marquage de toutes les étapes de sortie de cette transition.

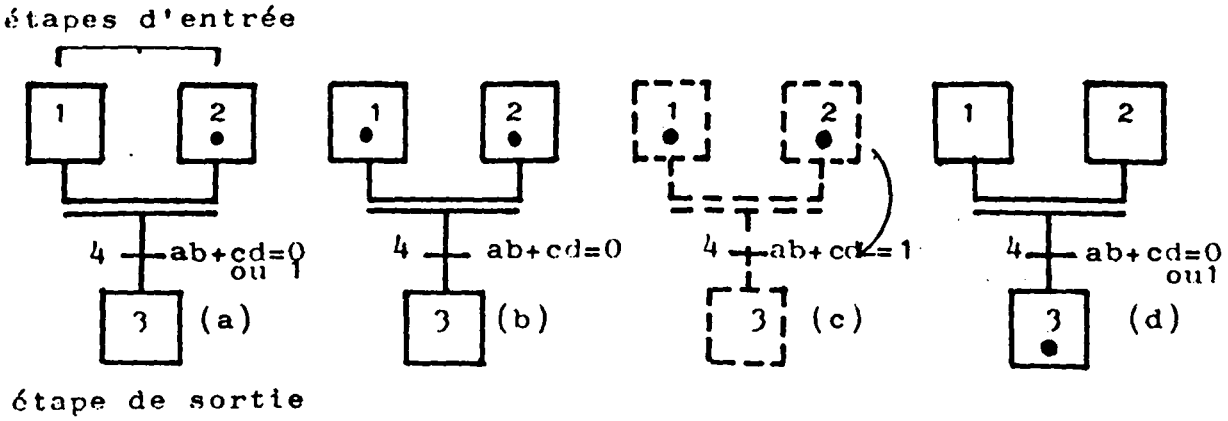


Figure 11

Figure 11-a : Transition non validée, l'étape 1 étant inactive

Figure 11-b : Transition validée, les étapes d'entrée sont actives, mais la transition ne peut être franchie; la réceptivité étant ab+cd=0

Figure 11-c : Lorsque ab+cd=1, la transition est obligatoirement franchie

Figure 11-d : L'étape 3 est active ab+cd=0 ou 1 après le franchissement de la transition

I-2-4-3- Complément sur les règles d'évolution

- Plusieurs transitions simultanément franchissables sont simultanément franchies

- Si au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste active.

I-2-5 - Exemple d'utilisation du GRAFCET pour la représentation du cahier des charges d'un automatisme logique [10]

Nous allons chercher à représenter le système séquentiel associé au dispositif de la figure 12, page suivante

Il s'agit de tracer un triangle rectangle par une plume portée par un bras.

Au départ, la plume P est en A ($H_1=V_1=1$) tous moteurs arrêtés.

En appuyant sur D, nous provoquons le déplacement de la plume suivant le cycle A.C.D.A.

De retour en A, la plume doit s'arrêter si D est relaché.

Traçage d'un triangle rectangle

ACDA par une plume

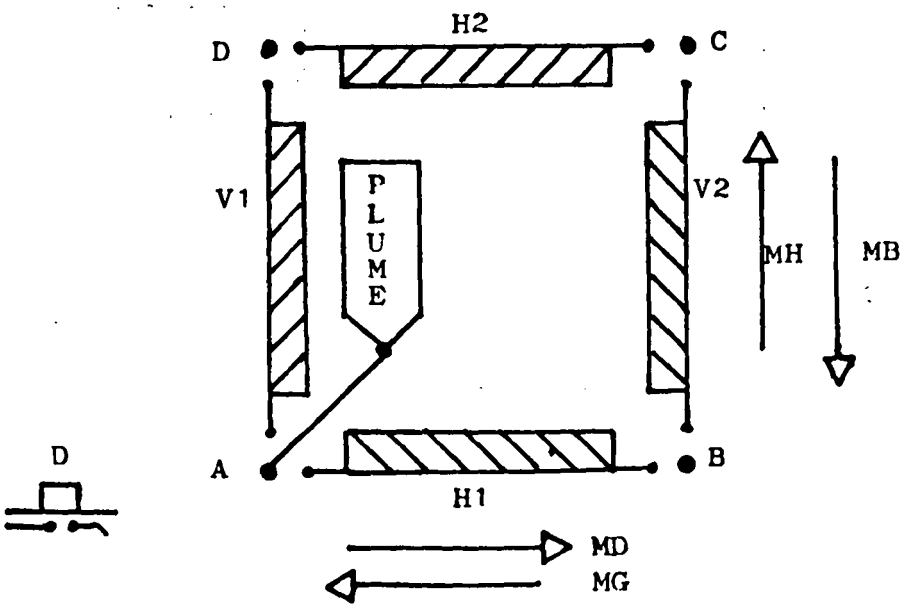


Figure 12

La traduction du processus effectué par la pointe de la plume est donné par le GRAFCET suivant :

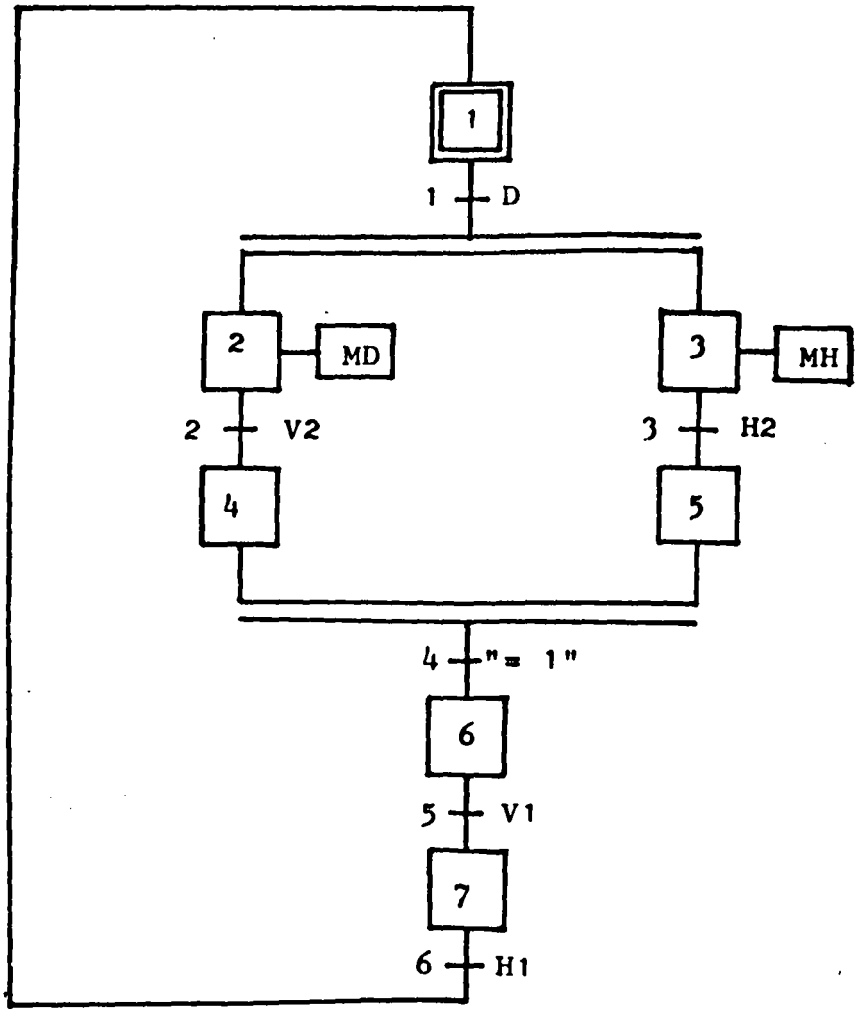


Figure 13

Ce GRAFCET est composé de sept étapes auxquelles sont associées les actions à effectuer :

Etapes	Actions associées
I	Initialisation
2	Déplacement à droite de la plume
3	Déplacement en haut de la plume
4	Etape d'attente ou de synchronisation à laquelle aucune action n'est associée
5	Etape d'attente ou de synchronisation à laquelle aucune action en général n'est associée
6	Déplacement de la plume à gauche
7	Déplacement de la plume en bas

De même, nous pouvons récapituler dans un tableau toutes les transitions et leurs réceptivités respectives.

Transition	Réceptivités associées
I	(D) départ
2	(V_2) Fin de déplacement à droite
3	(H2) Fin de déplacement en haut
4	"1". Transition toujours à 1
5	(V1) Fin de déplacement à gauche
6	(H1) Fin de déplacement en bas

En utilisant le GRAFCET en figure 13, nous pouvons, grâce aux règles d'évolution, suivre facilement l'évolution séquentielle de l'automatisme. Ainsi, nous pourrons examiner toutes les situations possibles et les conditions d'évolution du marquage.

I - 3 - CONCLUSION

Dans ce chapitre, nous avons fait une présentation générale des automates programmables et nous avons abordé la représentation d'un cahier des charges d'un automatisme logique par le GRAFCET.

Ces éléments vont nous permettre de définir maintenant le but de notre projet et les réalisations effectuées.

o
o o

CHAPITRE II - LES AUTOMATES MULTIPROCESSEURS & STRUCTURE A REALISER

II-1 - INTRODUCTION

Les architectures multiprocesseurs sont de plus en plus utilisées, car elles permettent d'obtenir des structures réellement modulaires et un fonctionnement performant. Mais elles posent le problème de la liaison et des échanges entre les différents processeurs qui les composent et celui de la mise en place des programmes.

La disponibilité de microprocesseurs de plus en plus sophisqués permet d'imaginer et de concevoir plusieurs classes de réalisations que nous avons réduites à deux :

- un fonctionnement de type "maître-esclave" entre modules du multiprocesseurs

- un fonctionnement de type "à priorité égale".

Le cahier des charges et la technologie employée vont nous permettre d'utiliser simultanément ou de faire le choix entre les deux types de fonctionnement[6].

La description des évolutions parallèles des automatismes logiques et la complexité de synthèse de ces processus, ont fortement incité à l'utilisation des structures multiprocesseurs.

Les objectifs fondamentaux espérés avec ces structures sont :

- de raccourcir au maximum la durée du cycle de traitement
- d'obtenir une simplicité de synthèse
- d'être plus proche d'une représentation effective de l'automatisme.

II-2 - EVOLUTIONS SIMULTANEES DANS UN GRAFCET [4], [8]

Le GRAFCET permet la description des systèmes logiques à évolutions simultanées.

On parle alors de parallelisme ou d'évolutions simultanées lorsqu'un GRAFCET possède, à un instant donné, plusieurs étapes actives. Dans le cas contraire, on parle de GRAFCET à étape active unique ou (graphe d'état).

II-2-1 - PARALLELISME STRUCTUTAL [1]

C'est la structure du graphe lui-même qui spécifie ce parallelisme quelques soient les réceptivités externes associées aux transitions.

La figure suivante nous donne un exemple de GRAFCET à parallelisme structural.

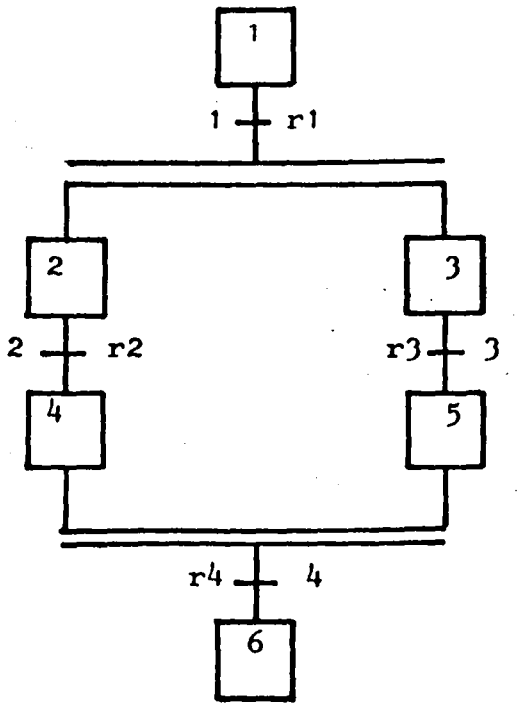


Figure II-1

Dans cette figure, le GRAFCET évolue de la situation (1) à la situation (2,3).

II-2-2 - PARALLELISME INTERPRETE [2],[1]

Rappelons la règle 4 d'évolution dans un GRAFCET "plusieurs transitions simultanément franchissables sont simultanément franchies".

Cette règle introduit un second type de parallélisme que nous appellerons parallélisme interprété, car pour une structure donnée du graphe, il est déterminé par les réceptivités associées aux transitions.

La figure suivante nous donne un exemple de GRAFCET à Parallélisme Interprété :

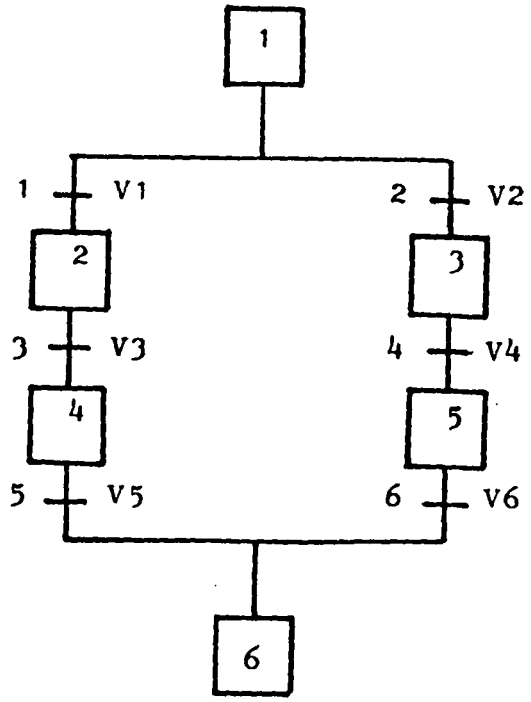


Figure II-2

On a trois cas à envisager :

- Les réceptivités V_1 et V_2 sont incompatibles. Une seule peut être vraie à la fois et nous aurons des évolutions exclusives, on évoluera soit vers la situation (2) si V_1 est vrai, soit vers la situation (3) si V_2 est vrai.

- Les réceptivités V_1 et V_2 sont identiques (vraies simultanément). Dans ce cas nous aurons toujours des évolutions simultanées, c'est-à-dire que le GRAFCET évoluera de la situation (1) vers la situation (2,3) par franchissements simultanés des transitions 1 et 2 lorsque $V_1 = V_2 = 1$

- Les réceptivités V_1 et V_2 sont compatibles, nous aurons alors soit l'activation de l'étape 2 si V_1 seule est vraie, soit l'activation de l'étape 3 si V_2 seule est vraie, soit l'activation des étapes 2 et 3 si V_1 et V_2 sont simultanément vraies.

II -3 - ESPRIT DU PROJET

Le but de notre projet est la conception et la réalisation d'un système multi-automates programmables à partir de modules à base du microprocesseur monobit 14500 B Motorola.

Ce système est destiné à traiter les problèmes de logique combinatoire ou séquentielle. Par ailleurs, nous essaierons de dégager des méthodes systématiques d'implantation des outils de description des automatismes logiques, en tenant compte surtout des problèmes de synchronisation que de tels systèmes induisent.

La programmation et la commande de notre système doivent se faire par l'intermédiaire d'une console spécialisée particulièrement orientée vers l'aide à la conception et à la programmation.

Le tout est réalisé à partir d'un microcalculateur C.B.M. 3032.

Avant de rentrer dans les spécifications des différents modules (ou cartes), nous allons définir la structure réalisée

II-4- - STRUCTURE A REALISER

II-4-1 - STRUCTURE DE BASE

Dans un GRAFCET, on peut remarquer l'existence de deux grandes familles d'opérations :

- les opérations à caractère combinatoire et qui s'effectuent au niveau des transitions.

- les opérations à caractère séquentiel et qui s'effectuent au niveau des étapes afin de faire évoluer l'état du GRAFCET.

La structure de base de notre système, doit permettre :

- soit de faire coopérer des automates classiques
- soit d'utiliser une structure spécialisée répartissant le combinatoire et le séquentiel reliés par une zone commune.

II-4-2 - SCHEMA SYNOPTIQUE

Le schéma synoptique est donné par la figure II-3 de la page suivante.

Cette figure nous décrit, d'une façon globale, la constitution du système multi-automates programmables que nous avons conçu. Il est composé de :

- trois automates programmables P1,P2,P3; on peut étendre ce nombre à "n" automates
- Une mémoire commune accessible à tous les processeurs et qui est le seul lieu d'échange d'informations entre les processeurs.
- Un arbitre qui a pour fonction de régler les problèmes d'accès à la mémoire commune afin d'éviter les con-

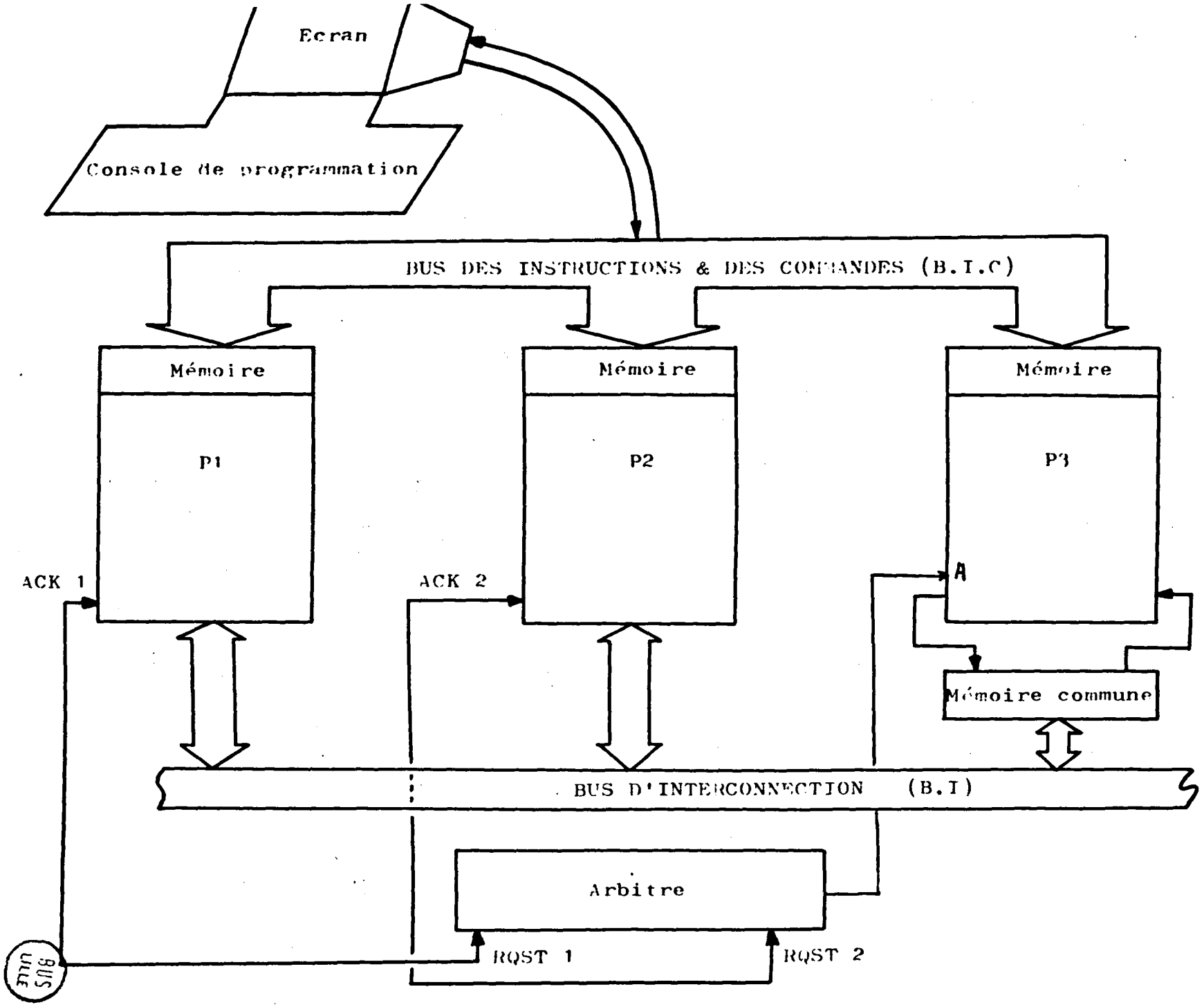

Ecran
Console de programmation
BUS DES INSTRUCTIONS & DES COMMANDES (B.I.C)
Mémoire
P1
Mémoire
P2
Mémoire
P3
ACK 1
ACK 2
A
Mémoire commune
BUS D'INTERCONNECTION (B.I)
Arbitre
RQST 1
RQST 2
BUS SIG

Figure II-3

flits entre les processeurs.

- Une console de programmation qui permet le dialogue entre l'opérateur et les différents automates par l'intermédiaire du BUS d'INSTRUCTIONS et de COMMANDES (B.I.C).

- Un BUS d'INTERCONNECTION qui permet le passage d'informations entre les différents automates.

Une fois vue la constitution générale de notre système, nous étudierons dans le chapitre III, la description détaillée et le fonctionnement de chaque partie en faisant toujours référence à la figure II-3.

Le système Multi-automates programmables a été conçu à partir de l'unité logique du monobit Motorola 14500 B. Il nous paraît donc indispensable, pour la compréhension du fonctionnement de notre système, de présenter rapidement le microprocesseur utilisé et ses caractéristiques de fonctionnement.

II-5 - ETUDE DU MICROPROCESSEUR 14500 B [11],[7]

II-5-1 - LES CARACTERISTIQUES

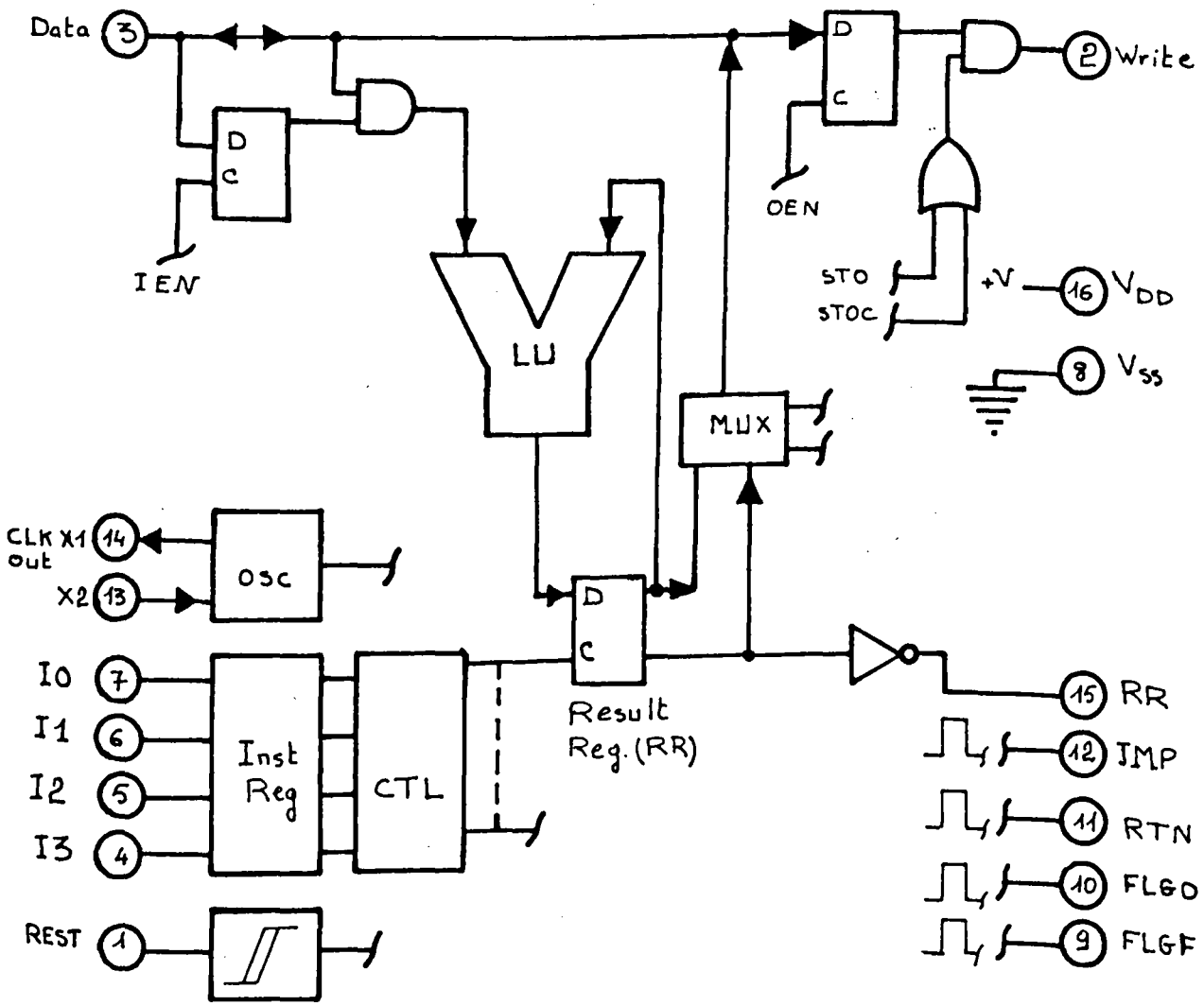
Le 14500 B est un microprocesseur de technologie CMOS. Il travaille sur des données de 1 bit et a été spécialement conçu pour résoudre et traiter des problèmes de logique combinatoire et séquentielle.

L'unité logique est capable d'élaborer et de traiter toutes les fonctions booléennes :

- intersection, intersection complémentée
- union, union complémentée
- union exclusive, etc...

Cette unité est commandée par une horloge produite par un oscillateur interne qui utilise une résistance externe. De même, on peut commander cette unité par une horloge externe; c'est cette deuxième possibilité que nous allons utiliser. La fréquence peut varier de 0 à I Mhz et chaque instruction est exécutée durant une période.

Le diagramme fonctionnel est représenté par la figure suivante :



On distingue 5 grandes parties :

- L'oscillateur
- Le registre d'instruction qui sélectionne l'opération à effectuer.(I_0, I_1, I_2, I_3) sont pris en compte sur le front descendant de l'horloge X_1
- LU ou unité logique, c'est là ou sont traitées toutes les opérations.
- 2 bascules internes, Input Enable Register (I E N) et Output Enable Register (O E N) qui contrôlent le transfert de la donnée (Data) à l'intérieur ou à l'extérieur du microprocesseur.
  Quant IEN est à l'état logique 1, la donnée se trouve appliquée à l'unité logique (LU) et le OEN à l'état 1 crée le signal WRITE.
- Et enfin le RR (Résult Register) c'est un accumulateur à un seul bit qui stocke le résultat de toutes les opérations effectuées par la "LU"

Le bloc CTL a pour fonction le décodage de l'instruction qui se trouve dans "Inst Reg" et envoie la commande logique correspondante à l'unité logique ou aux flags de sortie (JMP, RTN, FLGO, FLGF).

L'ensemble des instructions utilisées peut être résumé dans le tableau suivant :

Code d'instruction		Mnemonic	Action
0	0000	NOP0	No change in registers RR→RR, FLGO→ ⊓
1	0001	LD	Load Résult Reg:Data ⟶ RR
2	0010	LDC	Load Complement:Data ⟶ RR
3	0011	AND	Logical AND, RR.D ⟶ RR (D = Data)
4	0100	ANDC	Logical AND compl. RR. $\overline{D}$ ⟶ RR
5	0101	OR	Logical OR:RR + D→RR
6	0110	ORC	Logical OR compl: RR + $\overline{D}$→RR
7	0111	XNOR	Exclusive NOR:If RR = D, RR ⟶ 1
8	1ooo	STO	Store RR→Data Pin, Write → ⊓
9	1001	STOC	Store Compl. $\overline{RR}$→Data Pin, Write →⊓
A	1010	IEN	Imput Enable D⟶IEN Reg
B	1011	OEN	Output Enable D ⟶OEN Reg.
C	1100	JMP	Jump. JMP Flag ⟶ ⊓
D	1101	RTN	Return, RTN Flag→⊓Skip Nex inst.
E	1110	SKZ	Skip next instruction if RR = 0
F	1111	NOPF	No change in Registers RR→RR, FLGF→⊓

Ce tableau résume l'ensemble de toutes les instructions que l'on peut appliquer au microprocesseur 14500 B.

II - 5-2 - ETUDE DES SIGNAUX D'HORLOGE [11]

On peut diviser l'ensemble des signaux d'horloge suivant 4 groupes d'instructions.

a/ Les signaux correspondant aux instructions NOP, NOPF

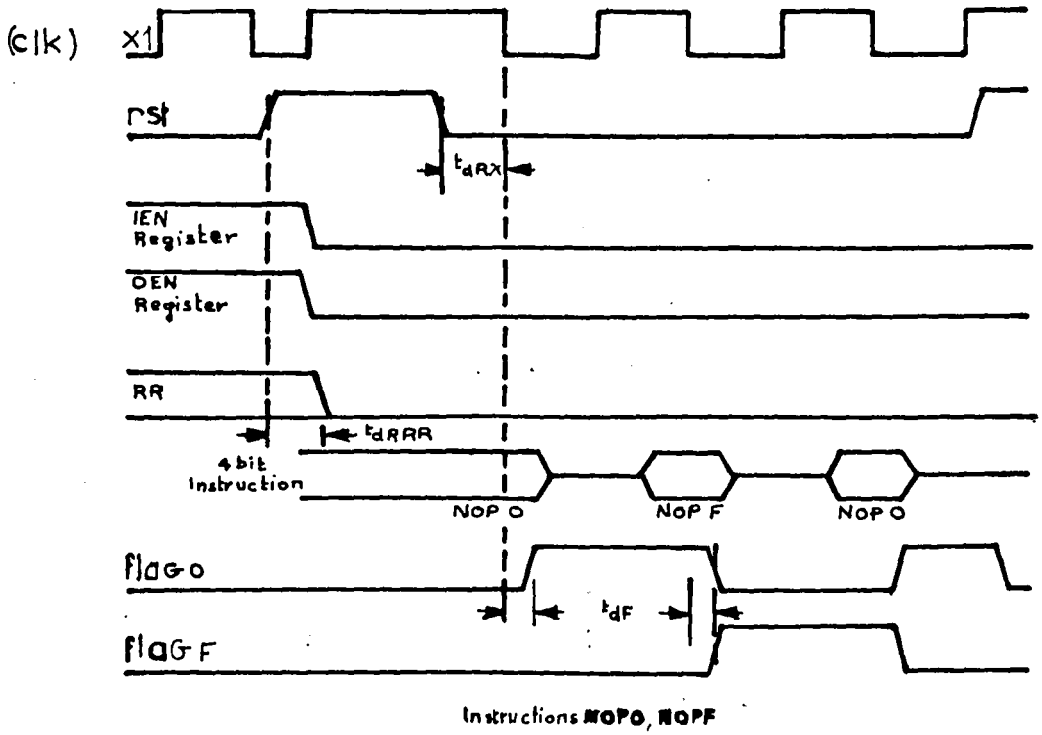


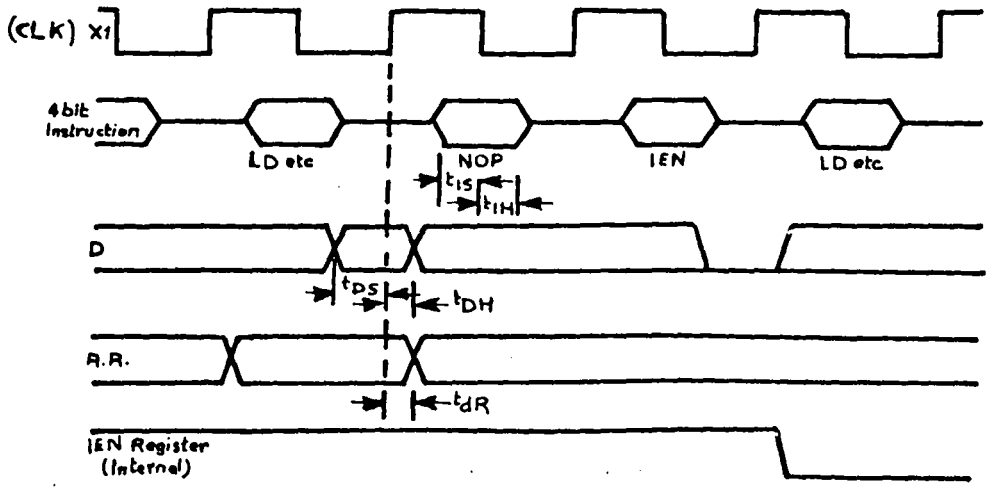
Figure II-5

Ce diagramme a été établi sans application des signaux RR, IEN, et OEN.

Pour les valeurs des différents temps, voir page A-1 en annexe.

b/ Les signaux correspondant aux instructions LD, LDC, AND, ANDC, OR, ORC, XNOR et IEN

Les instructions LD, LDC, etc.. sont prises en compte sur le front descendant du signal X_1, comme le montre le diagramme suivant :



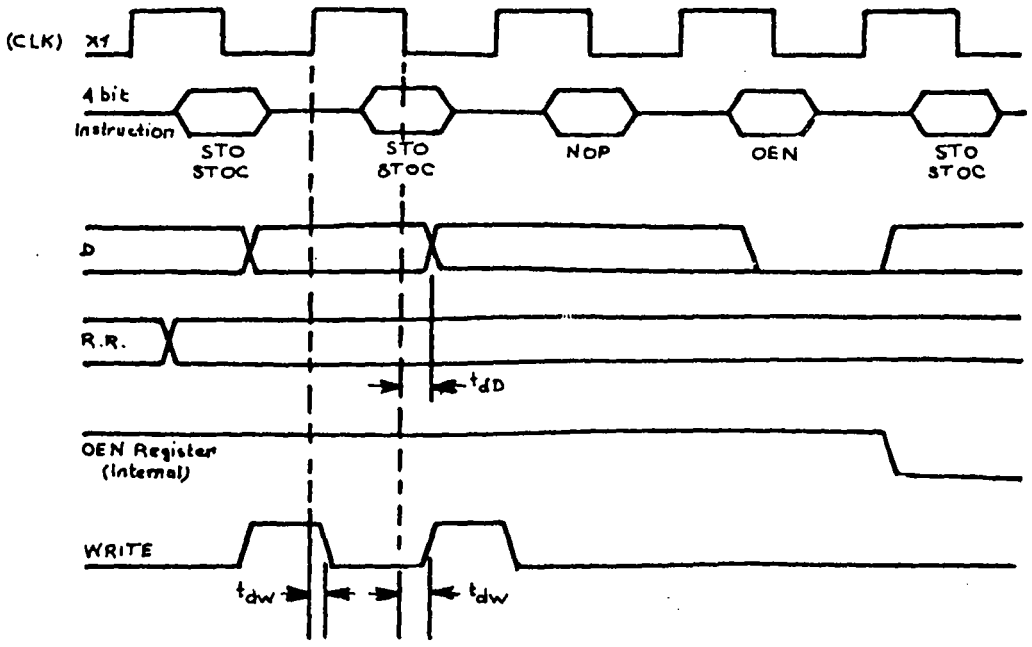
Instructions LD, LDC, AND, ANDC, OR, ORC, XNOR et IEN

Figure II-6

Ce diagramme a été établi pour RST = 0

Pour les valeurs des différents temps, voir page A-1

c/ <u>Les signaux correspondant aux instructions STO, STOC et OEN</u>



Instructions STO, STOC, OEN

Figure II-7

C'est au niveau du front montant de X_1 que nous avons le stockage du résultat de l'opération dans le registre (RR).

Pour les valeurs des différents temps, voir le tableau en Annexe, page A-1

II -5-3 - ETUDE DU MODE DE FONCTIONNEMENT DE L'HORLOGE UTILISE DANS NOTRE SYSTEME MULTI-AUTOMATES

Dans notre étude préalable, nous avions le choix d'utilisation entre deux modes de fonctionnement :

- le mode entrelacé
- le mode normal.

Dans le mode entrelacé, on peut avoir deux instructions par période, une sur l'état bas et l'autre sur l'état haut de l'horloge, ce qui nous fait changer la structure de la mémoire. Malheureusement, ce mode de fonctionnement s'avère impossible d'utilisation à partir d'une certaine fréquence et nous donnons, ci-dessous, les raisons :

Analysons le diagramme suivant :

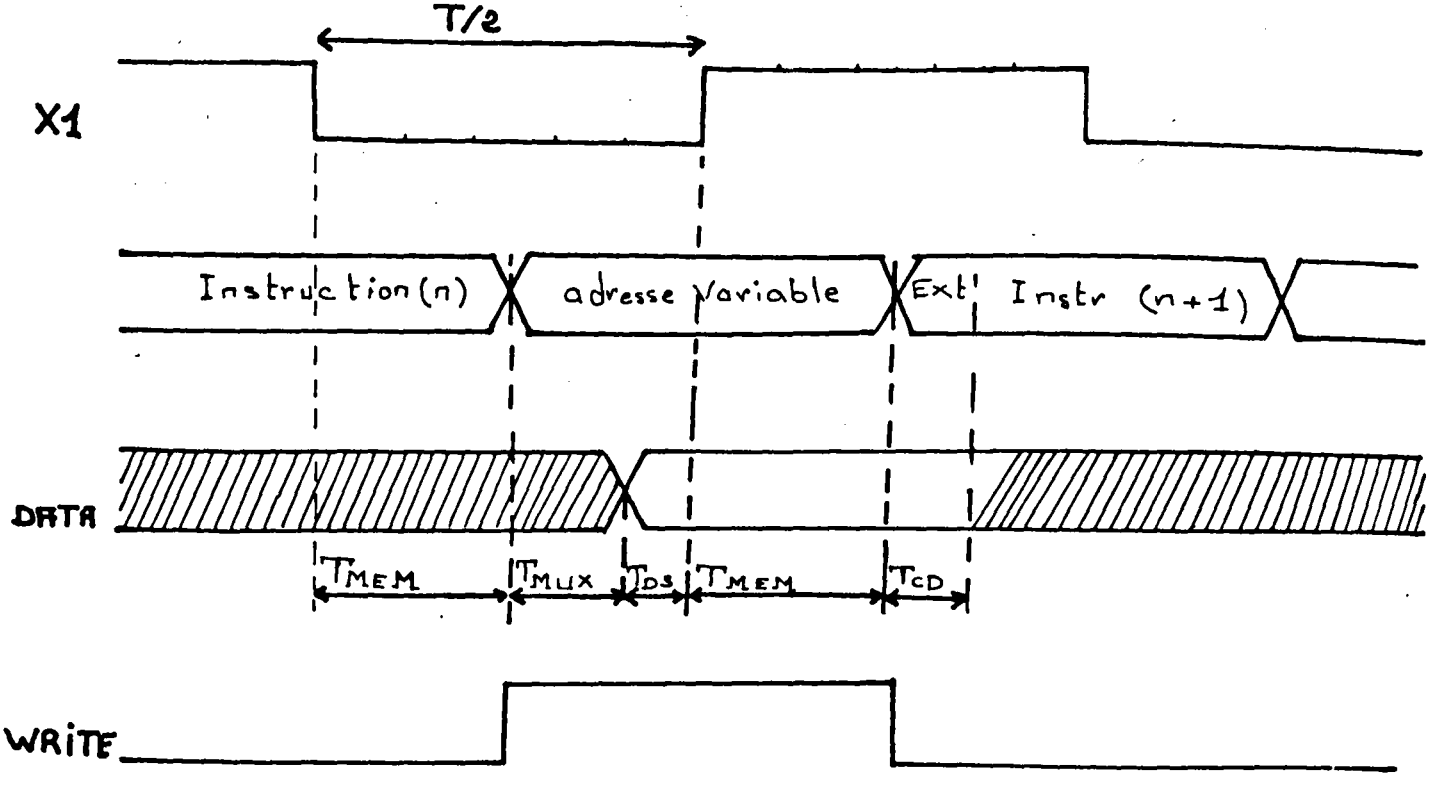


Figure II-8

Pour une alimentation en 5 volts (comptabilité TTL) et une fréquence d'horloge de 1 Mhz (X_1) soit $\frac{T}{2}$ = 500 ns

Le type de mémoire utilisé (P 2112 - 2 voir [8])

Le type de Port de sortie utilisé est le 14512, voir [7]

Nous avons relevé les temps d'établissement suivants

Pour $X_1 = 0$	(Mémoire P 2112-2 .....)	T_{MEM}	250 ns
	(Multiplexeur 14512 .....)	T_{MUX}	150 ns
	(Temps de stabilisation minimum des données .....)	T_{DS}	50 ns
			450 ns
Pour $X_1 = 1$	(Mémoire P 2112-2 .....)	T_{MEM}	250 ns
	(Décodage de la demande d'accès à l'extérieur .....)	T_{CD}	50 ns
			300 ns

En raison de la technologie utilisée, on observe un certain retard entre le signal d'horloge X_1, fourni par le circuit et le signal de commande X_2 appliqué à ce même circuit. Le retard est d'autant plus sensible que la tension d'alimentation est faible.

L'influence du retard n'étant pas la même sur les deux fronts d'horloge, cela se traduit par une modification du rapport cyclique de X_1.

Ainsi, les signaux relevés pour une tension d'alimentation de 5 volts, sont :

Pour X_2 = 500 Khz

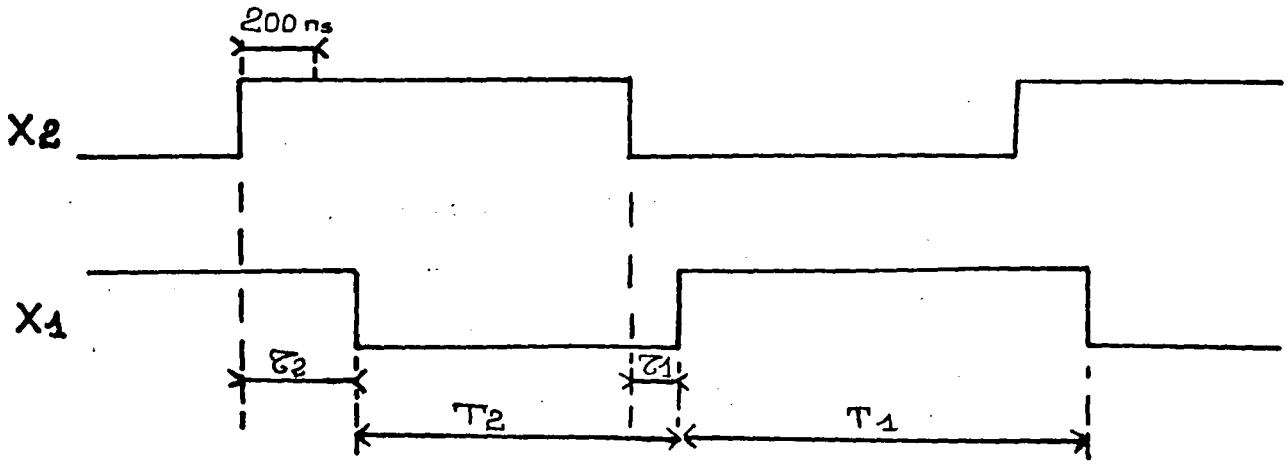


Figure II-9

Pour X_2 = 500 Khz

τ_1 = 160 ns
τ_2 = 360 ns
T_1 = 1,2 μ_s
T_2 = 0,8 μ_s

Pour X_2 = 1 Mhz

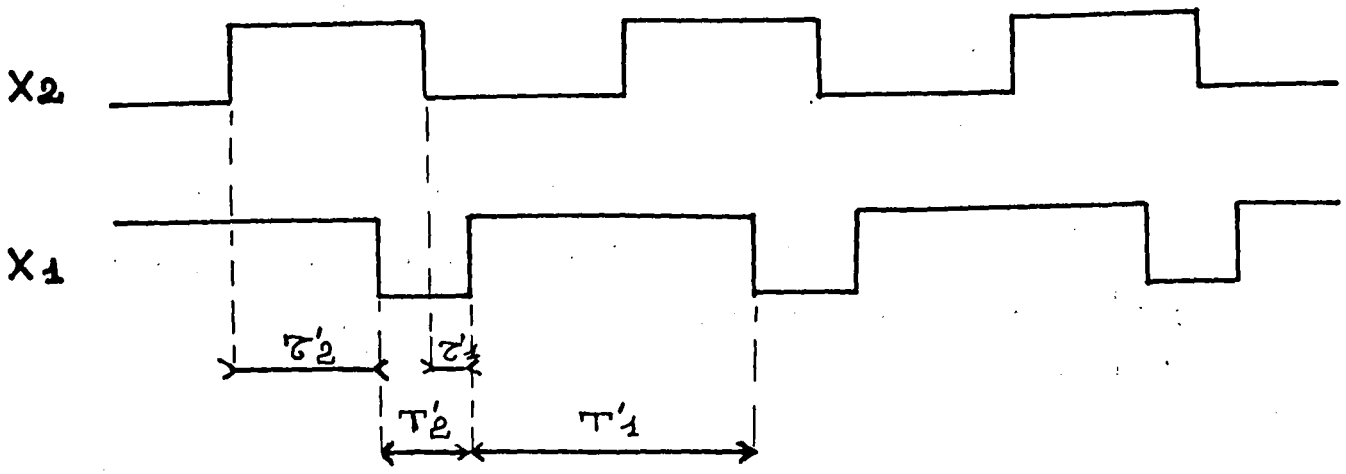


Figure II-10

Pour X_2 = 1 Mhz

τ'_1 = 100 ns
τ'_2 = 360 ns
T_1 = 760 ns
T_2 = 240 ns

II - 5-4 - CONSEQUENCE

Cette modification du rapport cyclique entraîne pratiquement une impossibilité de faire fonctionner notre système multi-automates en mode entrelacé au delà de 600 Khz sous une tension de 5 volts.

Nous allons donc faire fonctionner notre système en mode non entrelacé (une instruction par période).

II - 6 - CONCLUSION

Dans ce chapitre, nous avons abordé d'une façon globale les automates multiprocesseurs ainsi que le problème des architectures pour le traitement des processus parallèles.

Nous avons donc choisi une structure à base de trois automates programmables et un module (arbitre) réglant les problèmes d'accès à la zone commune par les différents processeurs.

Nous avons également défini les différents modes de parallélisme existant dans un GRAFCET, puisqu'il constitue l'outil fondamental pour la représentation des automatismes logiques dans notre étude.

Ce chapitre nous a permis de préciser le but de notre projet, de présenter le microprocesseur utilisé et de faire une étude préalable des modes de fonctionne-

ment de son horloge.

Dans le chapitre qui suit, nous allons décrire les différents processeurs qui constituent notre système multi-automates programmables.

o
o o

...

CHAPITRE III - ETUDE PRATIQUE & REALISATION DU SYSTEME MULTI-AUTOMATES PROGRAMMABLES

III-1 - INTRODUCTION

Ce chapitre comporte quatre parties qui décrivent le système multiprocesseurs. Nous allons nous attacher à l'étude des différents automates que nous avons conçus à partir de l'unité logique du microprocesseur monobit, le 14500 B Motorola.

Ceci nous a conduit à définir une architecture bien adaptée aux microprocesseurs monobit et une structure assez souple permettant l'échange d'informations entre les processeurs fixant les règles d'accès à des ressources communes et facilitant l'obtention d'une structure réellement modulaire et un fonctionnement en performance dégradée.

L'architecture générale est présentée par la figure II-3, page II-7.

Chaque automate travaille suivant le programme contenu dans sa mémoire programme, indépendamment du travail des autres sauf, bien sûr, lorsqu'il désire obtenir une information de la ressource (mémoire) commune aux différents automates.

Notre ensemble multi-automates est composé de trois cartes processeurs : P1, P2, et P3

- d'une carte arbitre (CA)

- d'un bus d'interconnections (BI)

Pour chaque carte nous décrivons la constitution et le fonctionnement de l'ensemble des éléments et nous donnons l'analyse de certains signaux fondamentaux pour la bonne marche du système Multi-automates.

III-2 - DESCRIPTION & FONCTIONNEMENT DES CARTES PROCESSEURS P1, P2 et P3

Les trois automates ont été conçus sur une base identique, néanmoins, l'automate P3, présente des différences au niveau des entrées-sorties, des variables internes et, surtout, des modalités d'accès à la mémoire commune. En-effet, il se trouve priviligié par rapport aux autres automates P1 et P2 qui doivent formuler une demande préalable pour accéder à la mémoire commune, alors que lui peut y avoir accès à tout moment.

III-2-1 - PRINCIPE

La structure des automates P1 et P2 est celle d'un automate programmable classique. La figure III-1, page III.3, montre cette structure.

P3 est un automate ayant pour seule relation à un ensemble d'informations : la mémoire commune. La figure III-2 page III.4, nous montre une telle structure.

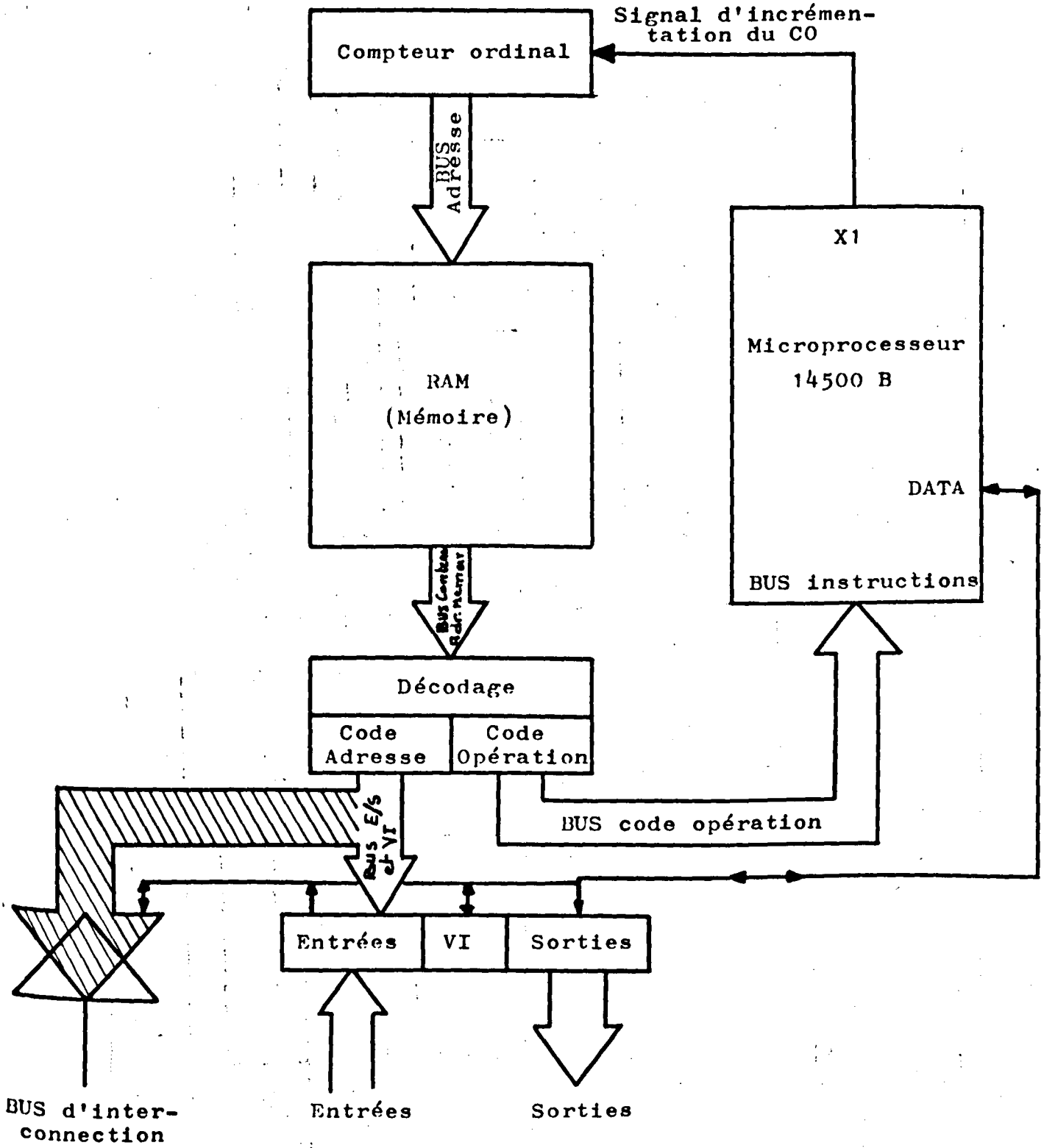


Figure III-1 : Schéma de principe des cartes processeurs P1 et P2

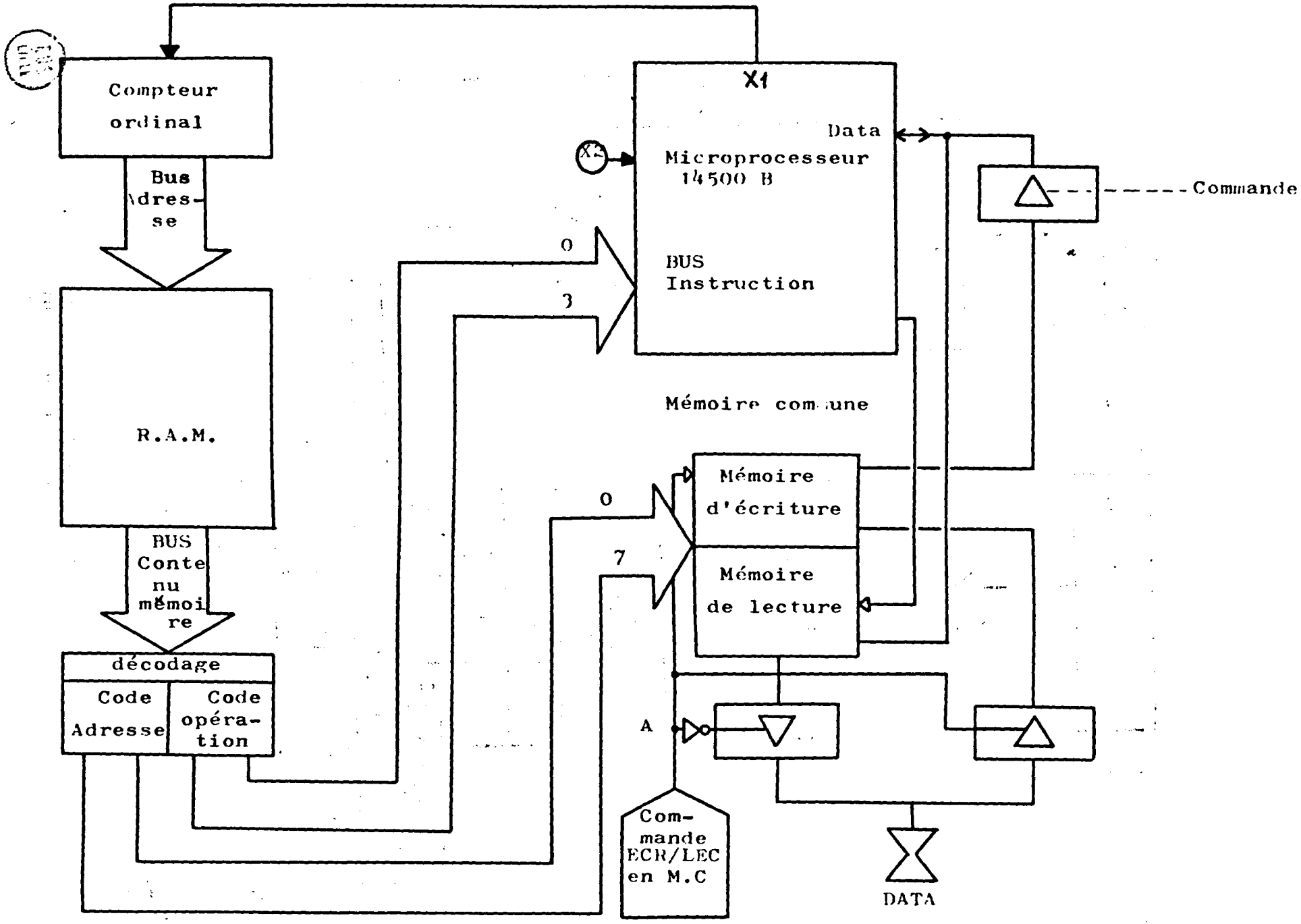

Compteur ordinal
Bus Adresse
R.A.M.
BUS Contenu mémoire
décodage
Code Adresse
Code opération
X1
X2
Data
Microprocesseur 14500 B
BUS Instruction
0
3
Commande
Mémoire commune
Mémoire d'écriture
Mémoire de lecture
0
7
A
Commande ECR/LEC en M.C
DATA

Figure III - 2

III-2-2 - COMPOSITION DES CARTES P1, P2 et P3

Dans cette partie, nous allons décrire comment a été conçu chaque module des différents processeurs (on trouvera les schémas de détail en annexe, pages A.2 - A.3 - A.4)

Chaque carte processeur comprend les éléments fondamentaux suivants :

- 1 mémoire de 256 x 12 bits réalisée à l'aide de 3 circuits RAM (P-2112) extensible à 1K x 12 bits [8]
- 1 microprocesseur 14500B de Motorola utilisé en mode : horloge commandée par l'extérieur [11]
- 1 compteur ordinal réalisé à l'aide de 2 circuits (74193)[9]
- l'aiguillage du bus d'instructions et commandes est réalisé à l'aide de 2 circuits 3 états (P8216) [8]
- ainsi que la logique nécessaire au décodage du contenu adresse et des mots de commandes. [9]

En plus de ces éléments, P1 et P2, possèdent :

- 16 entrées réalisées à l'aide de 2 circuits (MC 14512 CP de Motorola)[7]
- 16 sorties réalisées à l'aide de 2 circuits (MC 14099 B de Motorola)[7]
- 8 variables internes réalisées à l'aide d'un circuit (MC 14599 B de Motorola)[7]
- l'aiguillage des bits d'information processeur combinatoire - mémoire commune est réalisé à l'aide de 3 circuits (74365)[9].

D'autre-part, nous avons implanté la mémoire commune dans P3, elle est réalisée à l'aide d'une mémoire à double accès d'une capacité de 8x1 bit extensible à 64x1 bit. Le circuit utilisé est le (9338 A). De même, l'aiguillage des données arrivant des automates P1 et P2 vers le BUS adresse de la mémoire commune, est réalisé à l'aide de circuits à 3 états (74365).

Après avoir examiné la composition des différents automates, nous allons donner leurs fonctionnements.

III-2-3 - FONCTIONNEMENT DES PROCESSEURS

Comme nous le montre le schéma III-3 page III.7 toute conversation entre la console de programmations et le système multiautomates, transite par le canal des Bus INSTRUCTIONS & COMMANDES sous forme de 2 mots :

- le mot de commande
- le mot d'instruction

Ces deux Bus (bus de commandes et bus instructions) sont communs aux trois processeurs (P1, P2, P3). C'est le mot de commande qui sélectionne le ou les processeurs sur lesquels on travaille.

Pour sa part, le Bus Instructions contient l'information qui transite entre le calculateur et les processeurs.

III-2-3-1 - Le mot des commandes

Ce mot est constitué de la façon suivante

1/0	CS(2)	CS(1)	RESET/NOR	PROG/EXEC	AD/OP	LEC/ECR	Aut/PàP	STEP

(voir annexe, pages A.2 - A.4)

- le STEP est le bit d'incrémentation du compteur ordinal transmettant l'ordre d'écriture en mémoire.

- (PâP/Aut) est le bit qui définit le mode de fonctionnement pas à pas ou automatique

  Pàp/Aut = 0 implique le mode Pàp

  Pàp/Aut = 1 implique le mode Automatique

-(ECR/LEC) est le bit qui indique si on désire charger ou lire la mémoire programmée.

  ECR/LEC = 0 implique le mode écriture

  ECR/LEC = 1 implique le mode lecture

- (OP/AD) est le bit qui indique s'il s'agit de l'octet instruction du "code opération" ou de l'octet instruction du "code adresse"

  OP/AD = 0 Implique octet du code opération

  OP/AD = 1 Implique octet du code instruction

- (EXEC/PRO) est le bit qui indique au système dans quel mode de fonctionnement on est : le mode programmation ou le mode exécution du programme mémoire.

  EXEC/PRO = 0 implique mode exécution du contenu mémoire programme

  EXEC/PRO = 1 implique mode programmation de la mémoire programme

- (NOR/RESET) est le bit qui provoque une remise à zéro générale quand il est à l'état haut.

III - 2-3-2 - LE MOT INSTRUCTION

Ce mot est porté par le bus Instruction qui est bidirectionnel.

Ce mot est formé par les 8 bits suivants :

DI7 / D07	DI6 / D06	DI5 / D05	DI4 / D04	DI3 / D03	DI2 / D02	DI1 / D01	DI0 / D00

(voir annexe, pages A.2 - A.4)

Suivant le mode d'écriture ou lecture, nous allons travailler soit avec les bits DIo ----- DI7 ou DOo -----DO7

- les bits DIo DI1 ----- DI7 représentent le mot instruction à faire rentrer dans la mémoire dans le mode écriture. Ce qui correspond à Ecr/LEC = 0
- les bits DOo, DO1 ----- DO7 représentent les bits du contenu mémoire à lire. Ce qui correspond à ECR/LEC = 1

Ce mot de 8 bits est aiguillé alternativement sur les bits de 8 à 11 et de 0 à 7 de la mémoire suivant qu'on utilise le "code Opération" ou le "code adresse". Le schéma III-3 nous montre cet aiguillage.

III-2-4 - FONCTIONNEMENT DES PROCESSEURS P1 et P2

(schéma en annexe, pages A.2 - A.3)

Nous allons décrire le fonctionnement de la carte processeur P1 ou P2. On suppose le programme déjà introduit

dans la mémoire (RAM)
la procédure de programmation est décrite dans le chapitre IV

La scrutation de la mémoire se fait, soit en automatique, soit en pas à pas.

Dans le cas d'un fonctionnement en automatique, l'incrémentation du compteur ordinal se fait par le signal d'horloge H_2 issu d'un oscillateur.

Dans le cas d'un fonctionnement en pas à pas, l'incrémentation du compteur ordinal se fait par la commande STEP.

PRINCIPE DE SELECTION DE L'AUTOMATE A ACTIVER

CS(2)	CS(1)	Automates
0	0	P3
0	1	P1
1	0	P2
1	1	*

* tous les automates sont sélectionnés

C'est le compteur ordinal qui désigne l'adresse de la case mémoire, 12 bits alors se présentent en sortie; les 4 bits supérieurs (8 à 11) désignent le code opération et sont directement appliqués sur le BUS instruction du microprocesseur 14500 B. Une fois l'instruction traitée, le microprocesseur range le résultat sur le BUS data qui est lui-même appliqué à tous les circuits E/S, VI, et sur

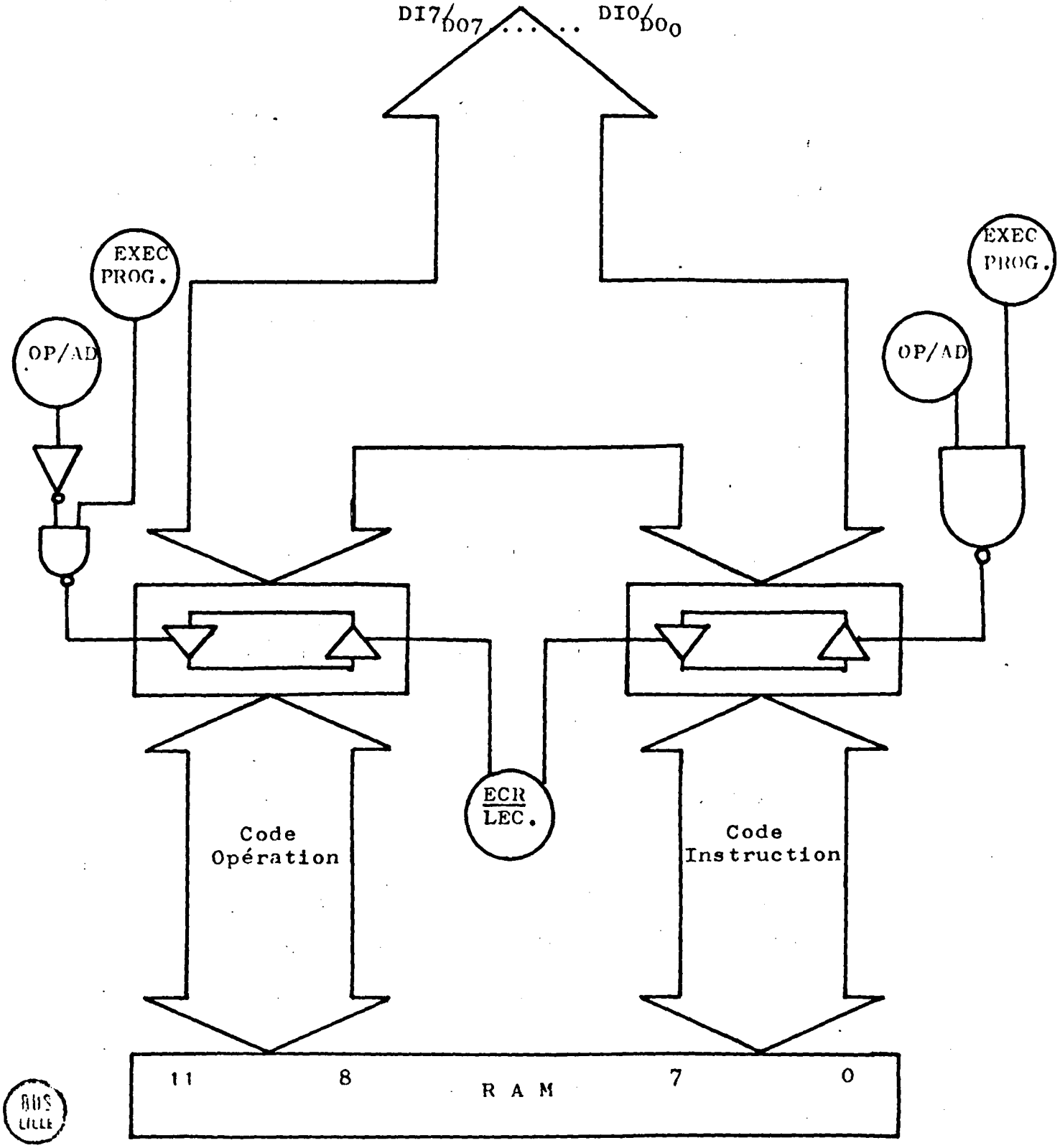
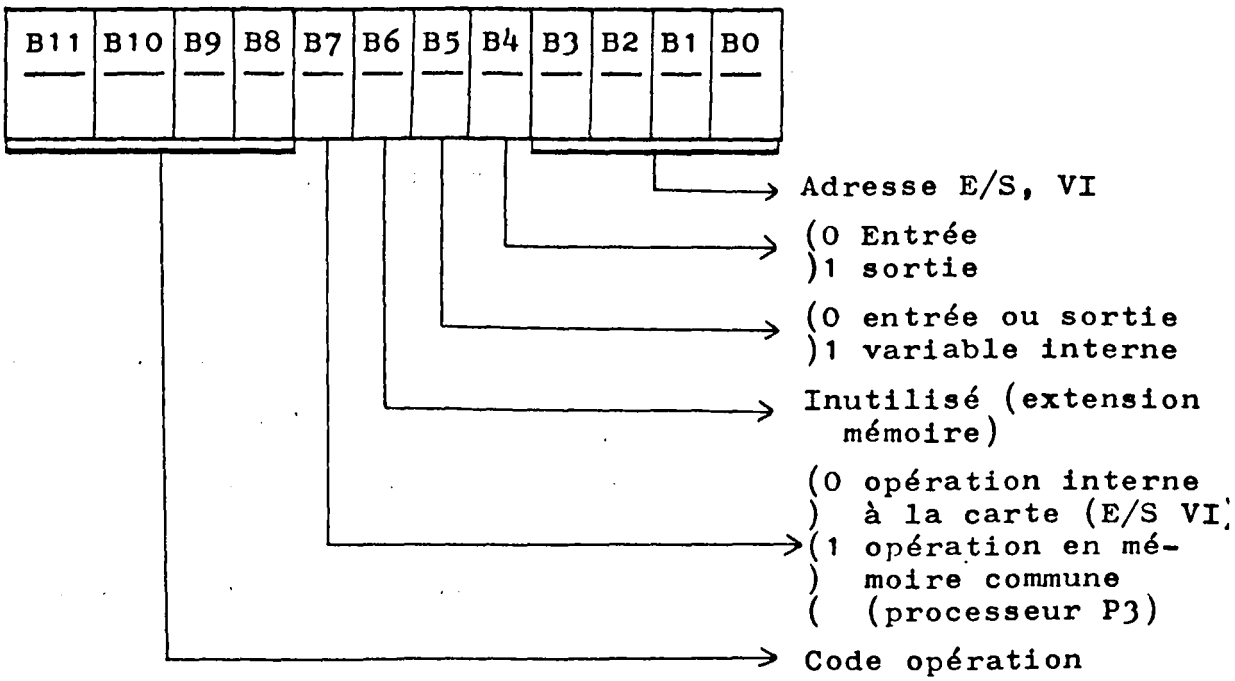

DI7/DO7 . . . . DIO/DO0
EXEC PROG.
OP/AD
EXEC PROG.
OP/AD
ECR/LEC.
Code Opération
Code Instruction
11
8
R A M
7
0

Figure III - 3

le BUS d'Interconnections commun à tous les processeurs.

Les 8 bits inférieurs (0 à 7) sélectionnent l'adresse à laquelle la donnée doit être appliquée. Cette adresse peut être celle d'une entrée, d'une sortie, d'une variable interne ou d'une variable en mémoire commune.



Adresse entrée :

de 0000 à 0111 pour les entrées de 0 à 7
et
de 1000 à 1111 pour les entrées de 8 à 15

Adresse sortie :

de 10000 à 10111 pour les sorties de 16 à 23
et
de 11000 à 11111 pour les sorties de 24 à 31

Variables internes :

de 100000 à 100111 pour les variables internes de 32 à 40

Adresse Mémoire commune

Les adresses en mémoires communes sont codées sur 6 bits, soit une mémoire de capacité de 64 cases. Les

adresses utilisées sont comprises entre 128 et 135.
Le règlement des accès à la mémoire commune sera traité avec la description de la carte arbitre.

III-2-5 - FONCTIONNEMENT DE LA CARTE P3

Le fonctionnement du processeur P3 est différent des deux autres par le fait qu'il ne possède ni entrée, ni sortie, ni variable interne. En plus, P3 à le privilège de pouvoir accéder librement en mémoire commune, il ne converse qu'avec elle.

L'utilisation d'une "mémoire" à double accès, nous donne la possibilité de lire et d'écrire simultanément dans la mémoire commune sans perte de temps.

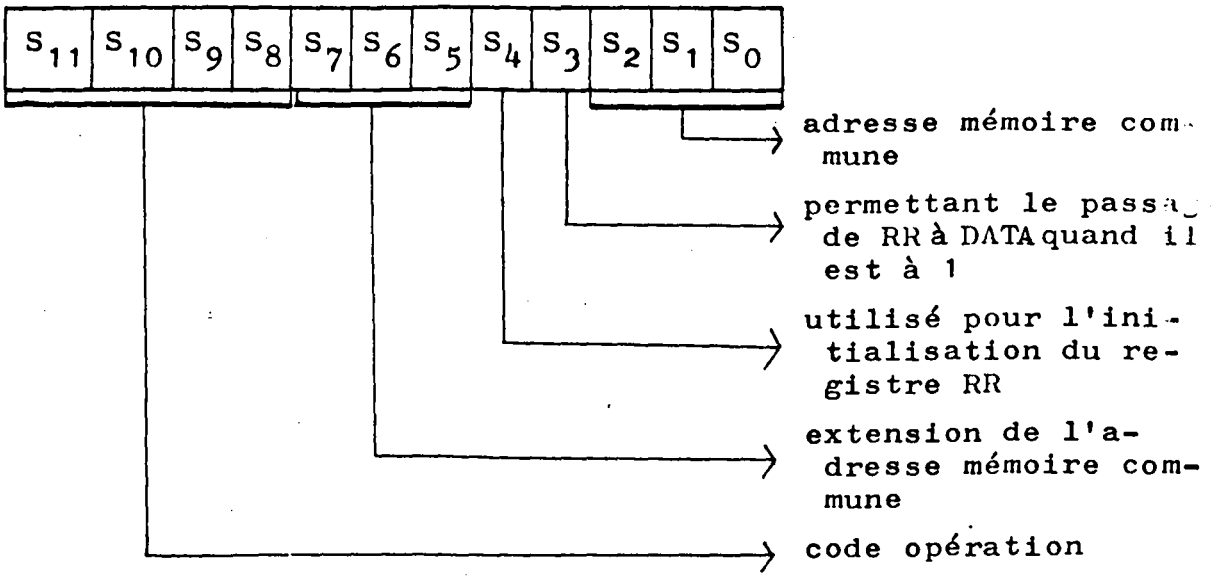
La procédure de programmation sera décrite au chapitre IV.

Le compteur ordinal fournit l'adresse de la case mémoire. 12 Bits se présentent en sortie; les 4 Bits supérieurs (8 à 11) désignent le code opération et sont directement appliqués sur le BUS Instruction du microprocesseur 14500 B.

Une fois l'instruction traitée, le microprocesseur range le résultat sur le Bit DATA qui est, lui-même, appliqué sur le Bus "données" de la mémoire commune.

Les 8 Bits restants de 0 à 7 sélectionnent l'adresse mémoire commune à laquelle doit être appliquée la donnée mais, pour notre étude, nous nous sommes restreints à l'utilisation de 5 des 8 Bits, voir schéma annexe page A.4.

D'autre-part, on trouve le signal A issu de la carte arbitre qui commande l'accès à la mémoire commune pour les autres processeurs (P1 et P2). Ce signal sert aussi pour l'écriture ou la lecture en mémoire commune par les processeurs P1 ou P2.



III - 3 - CARTE ARBITRE

III-3-1 - PRINCIPE ET CAHIER DES CHARGES

Les trois fonctions fondamentales de la carte Arbitre sont :

- La commande du signal d'horloge de chaque processeur par l'intermédiaire du signal H :
  - .H (1) pour le processeur P1
  - .H (2) pour le processeur P2
- La commande d'écriture ou de lecture d'un processeur P1 ou P2 en mémoire commune par l'intermédiaire du signal A
- La commande d'ouverture ou de fermeture de l'accès au

bus d'interconnectionsdes six fils d'adresse de la mémoire commune (B_0, B_1.. B_5) par l'intermédiaire du signal BIBO :

BIBO (1) pour le processeur P1

BIBO (2) pour le processeur P2

L'utilisation et le fonctionnement de plusieurs automates en parallèles pose un problème au niveau de l'utilisation de la mémoire commune. Pour éviter qu'il y ait des conflits entre les processeurs, nous avons conçu un système de priorité.

Dans le cas où deux processeurs demandent l'accès à la mémoire commune en même temps, on donne la priorité au processeur numéro 1. Une fois son traitement avec la mémoire commune terminé, on passe au processeur numéro 2 qui dialoguera à son tour avec la mémoire commune.

Quand un seul processeur demande l'accès, il échange avec la mémoire commune puis libère le bus d'interconnections.

Nous donnons un organigramme montrant les différents cas de fonctionnement qui se présentent. Les rôles des divers signaux sont présentés au paragraphe suivant qui traite du fonctionnement de la carte Arbitre.

Nous présentons maintenant le schéma des liaisons entre les différents blocs qui constituent cette carte Arbitre (figure III-4).

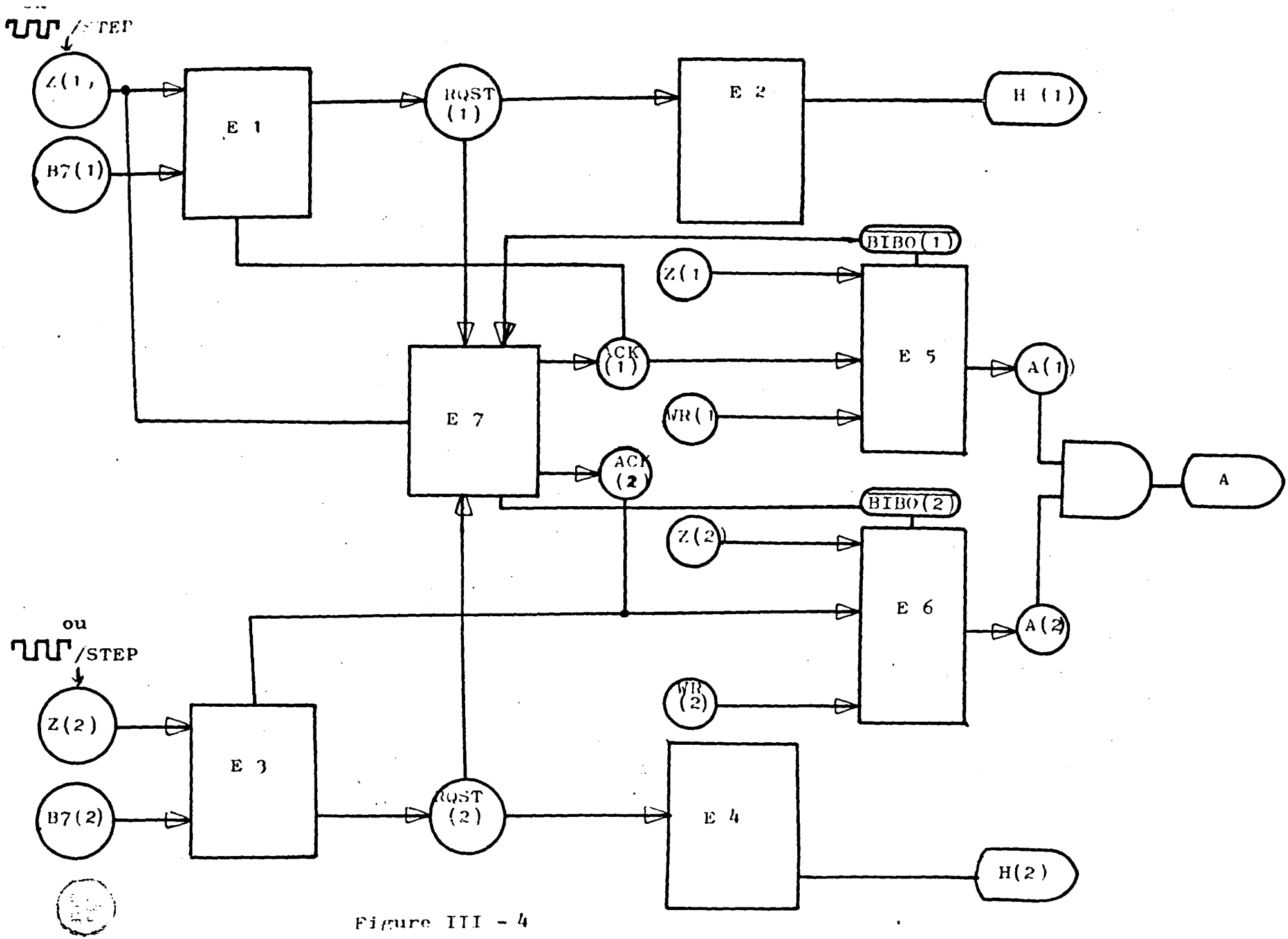

/STEP
Z(1)
B7(1)
E 1
RQST (1)
E 2
H (1)
BIBO(1)
Z(1)
E 5
A(1)
ACK (1)
E 7
WR(1)
ACK (2)
A
BIBO(2)
Z(2)
E 6
A(2)
ou
/STEP
Z(2)
E 3
B7(2)
RQST (2)
E 4
WR (2)
H(2)

Figure III - 4

III-3-2 - FONCTIONNEMENT DE LA CARTE ARBITRE

(Le schéma de la carte Arbitre est donné en page A.5)

D'un point de vue général, la carte Arbitre a pour fonction la gestion des différents signaux provenant des cartes processeurs P1 ou P2 et l'envoi par cette même carte des commandes, soit aux processeurs P1 ou P2, soit au processeur P3 (voir paragraphe BUS d'interconnections entre les différentes cartes).

Un processeur P1 ou P2 fait une demande d'accès à la mémoire commune par l'intermédiaire du signal B7 : B7 (1) ou B7 (2); à ce moment là l'horloge H : H (1) ou H (2) se bloque par l'intermédiaire du signal RQST : RQST (1) ou RQST (2). Ce blocage va durer le temps nécessaire pour résoudre le problème de priorité et s'assurer de l'absence de tout conflit. On a alors l'autorisation d'accès à la mémoire commune par l'intermédiaire du signal "ACK" : ACK (1) pour P1 ou ACK (2) pour P2 qui régit un signal BIBO : BIBO (1) pour P1, BIBO (2) pour P2, propre au processeur et qui nous donne le droit d'accès au Bus des adresses mémoires communes (B0 ... B5) et au Bus Donnée. Toutes ces opérations s'opèrent en respectant le diagramme des temps donné au paragraphe suivant en tenant compte des priorités.

III-3-3 - DIAGRAMME DES ECHANGES

Les deux diagrammes que nous présentons ont été relevés respectivement avec une horloge de base 40 KHZ et de 4 MHZ, ce qui nous a permis de vérifier le bon fonctionnement de notre système à une fréquence d'horloge de microprocesseur 14500 B de l'ordre de 1 MHZ.

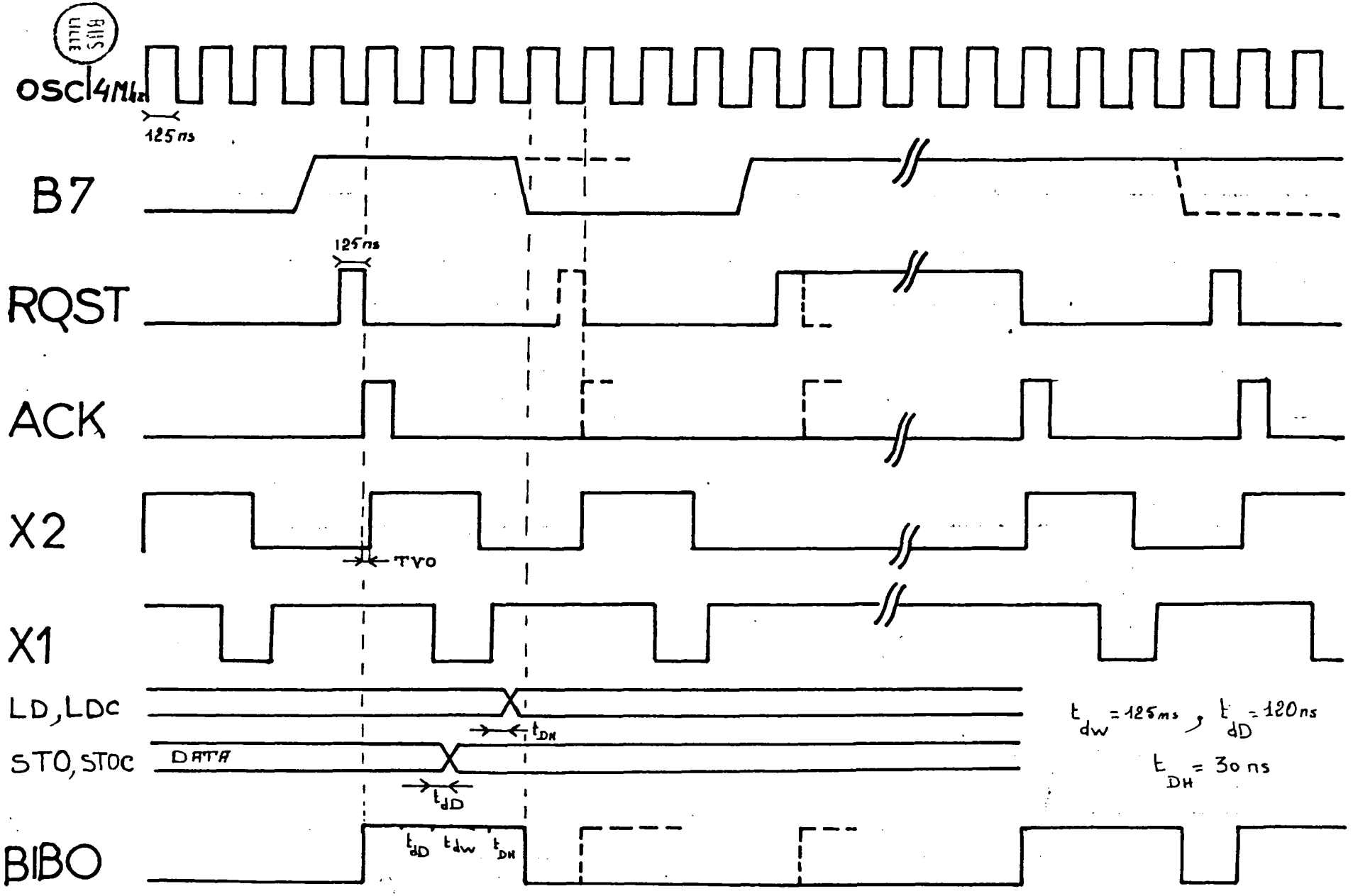


diagramme relevé à 4 MHZ

Figure III - 6

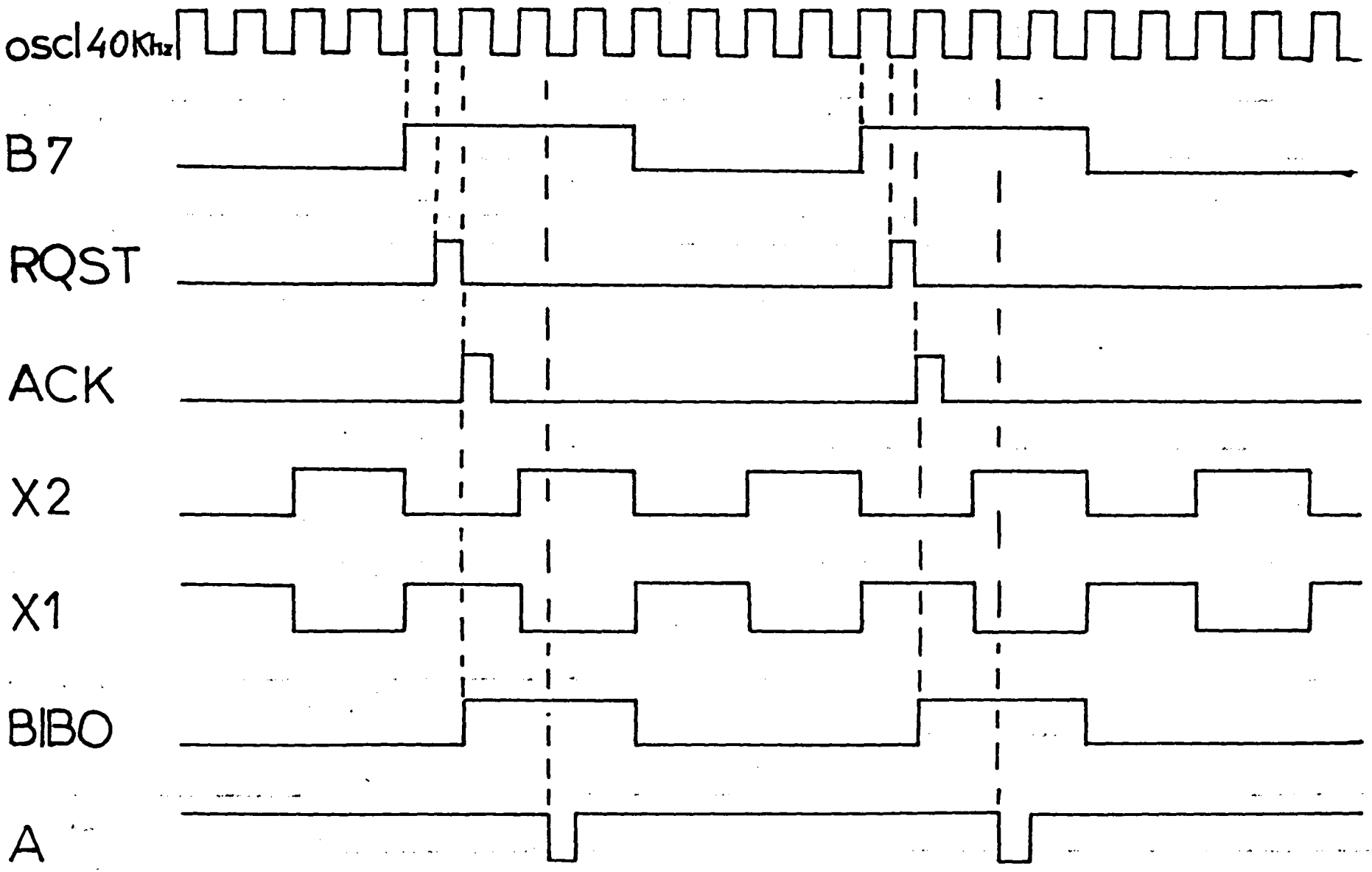


diagramme relevé à 40 khz

Figure III - 5

III-3-4 - ORGANIGRAMME DE PRIORITE UTILISE POUR NOTRE SYSTEME MULTI-AUTOMATES PROGRAMMABLES

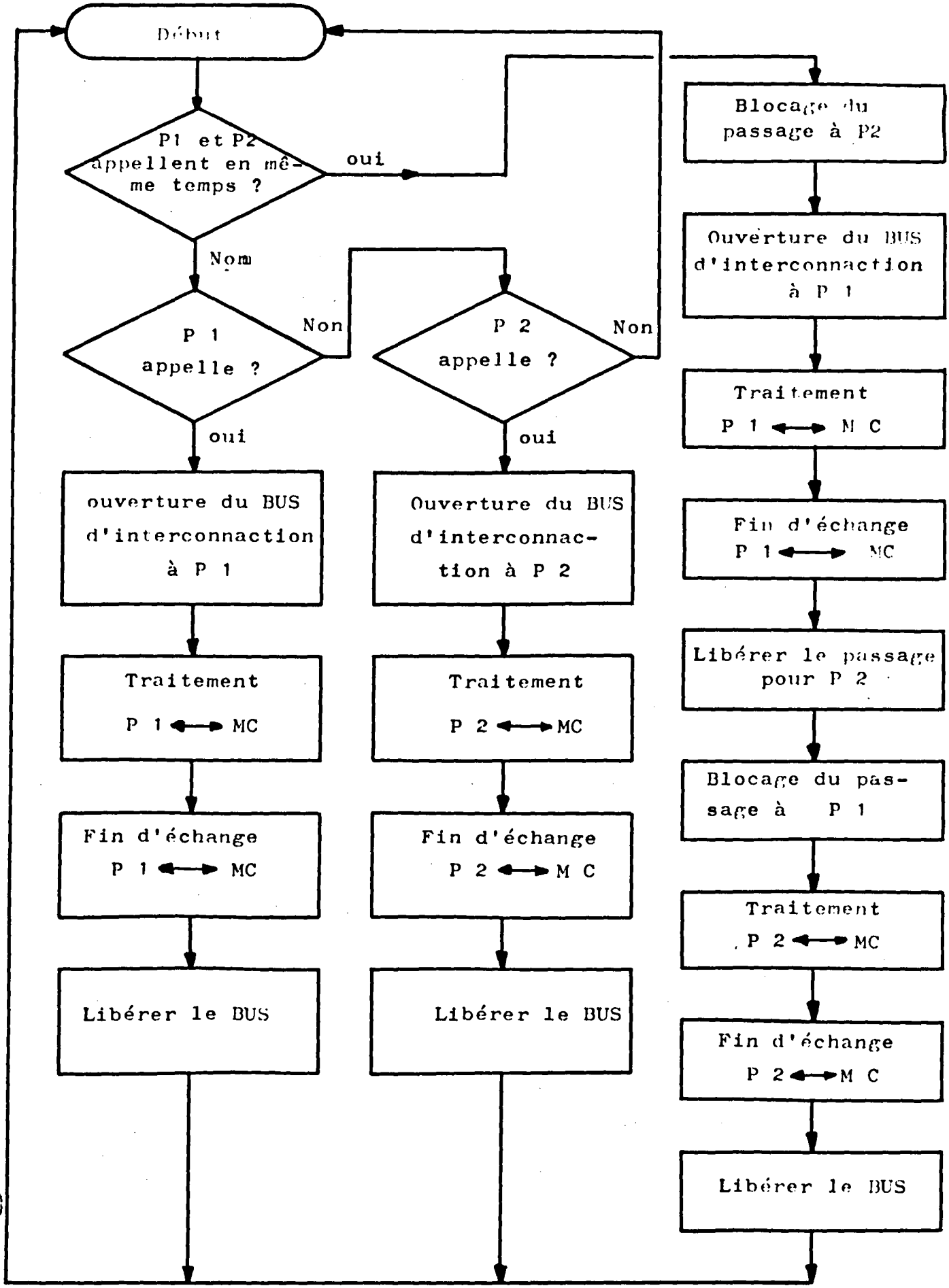
Très vite nous avons abandonné l'idée d'utiliser un circuit de priorité programmable car il n'existe pas sur le marché actuel de circuit adapté à notre système multiprocesseurs ce qui nous a obligé à concevoir un système de priorité cablé répondant aux exigences du système.

Suivant l'architecture du système, deux cas de demande d'accès à la mémoire commune peuvent se présenter :

- le cas où 1 seul automate P1 ou P2 demande l'accès à la mémoire commune
- le cas où les automates P1 et P2 demandent l'accès à la mémoire commune en même temps.

Le premier cas ne présente aucun problème, l'un des deux processeurs traite avec la mémoire commune, l'autre continue son traitement local.

Par contre, dans le deuxième cas où on a une demande simultanée d'accès à la mémoire commune, nous avons réglé ce conflit en donnant la priorité à P1 tout en bloquant P2, puis, après le traitement (P1-mémoire commune) on donne l'accès de la zone commune à P2. Tout ceci est représenté par l'organigramme suivant.



Organigrammme du mode de priorité choisi

Figure III - 7

III-4 - BUS d'INTERCONNECTIONS ENTRE LES DIFFERENTES CARTES

La connection entre les cartes est une partie importante pour le dialogue et le transfert d'informations entre les différents processeurs d'une part (P1, P2 et P3) et la carte Arbitre de l'autre.

La structure utilisée pour cette liaison est celle d'un BUS de signaux.

Les signaux qui sortent des cartes processeur P1 ou P2, prennent deux directions : vers la carte Arbitre et vers la carte Processeur P3.

III-4-1 - SIGNAUX : PROCESSEURS P1 ou P2 ⟷ CARTE ARBITRE

On se restreint ici aux signaux de liaisons entre une carte processeur P1 ou P2 et la carte Arbitre (C.L.)

- Le signal B7 (B7 (1) ou B7 (2))va de la carte P1 ou P2 à la carte C.L.
- Le signal H (H (1) ou H (2))de l'horloge du compteur ordinal, va de la carte Arbitre (C.L.) à la carte processeur (P1 ou P2)
- Le signal WRITE du microprocesseur 14500 B Motorola va de la carte processeur (P1 ou P2) à la carte Arbitre (C.L)

- Le signal BIBO (BIBO (1) ou BIBO (2)) qui permet l'ouverture du BUS d'interconnection au processeur (P1 ou P2). Il va de la carte (C.L) à la carte processeur (P1 ou P2)

- Le signal RESET permet la remise à zéro du système, agit sur la carte P1 ou P2 et C.L

- Le signal Z (i) (Z (1) ou Z (2)) va de la carte processeur (P1 ou P2) à la carte Arbitre (C.L)

Les schémas correspondant à ces signaux sont donnés en figure page

III-4-2 - SIGNAUX : PROCESSEUR P1 ou P2 ⟷ PROCESSEUR P3

Il y a 6 Bits d'adresse de la mémoire commune; on a un ensemble de 6 fils communs à tous les processeurs P1, P2, P3 et qui vont vers la carte processeur P3.

Le signal DATA qui représente la donnée qu'on veut lire ou écrire dans la mémoire commune. Il est commun à tous les processeurs.

III-4-3 - SIGNAUX : PROCESSEUR P3 ⟷ CARTE ARBITRE (C.L)

- Le signal "A" donne l'ordre d'écriture ou de lecture à la mémoire commune, il est généré par le signal BIBO et il va de la carte Arbitre à la carte Proces-

seur P3.

Le schéma d'interconnection est donné ci-dessous :

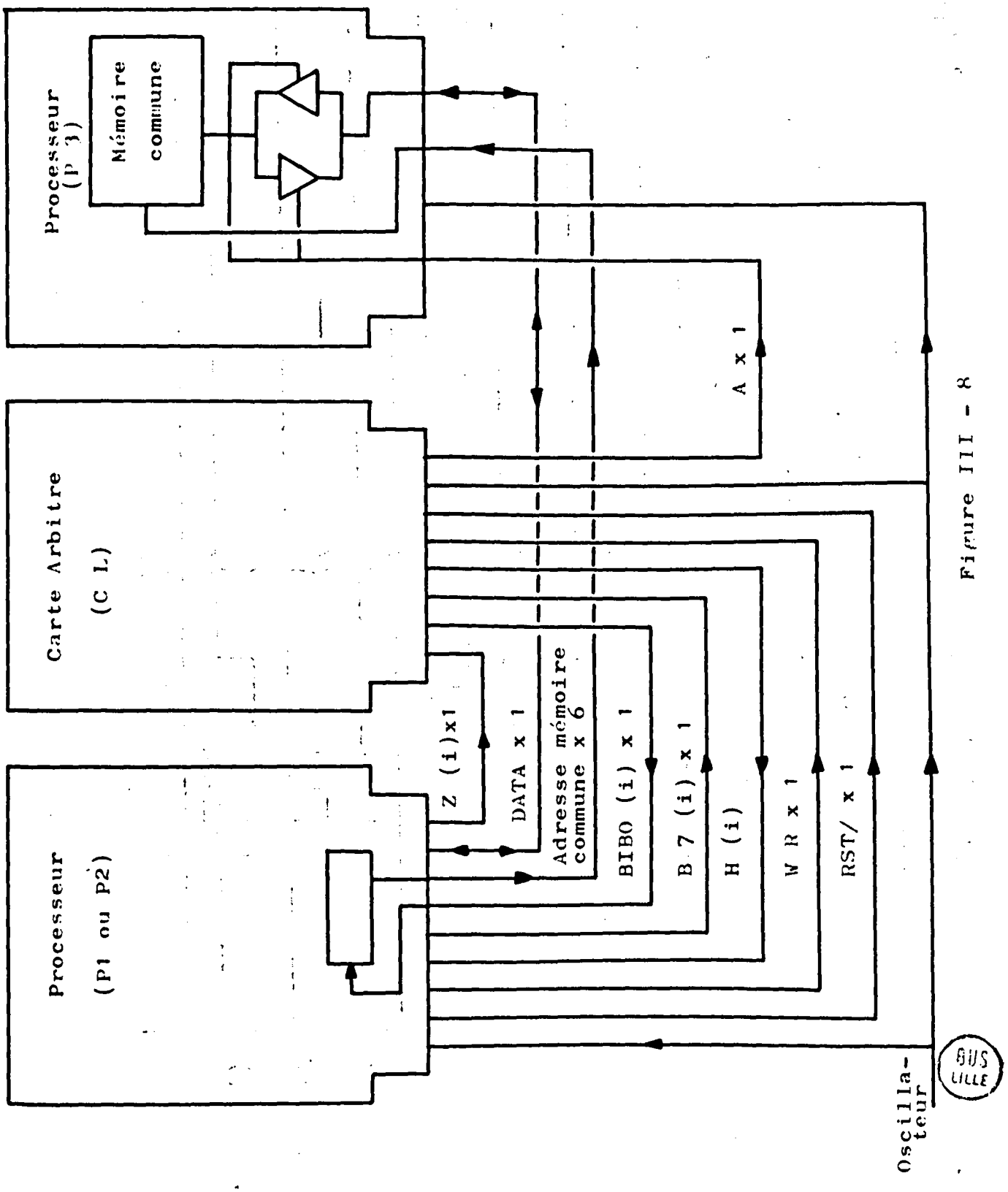
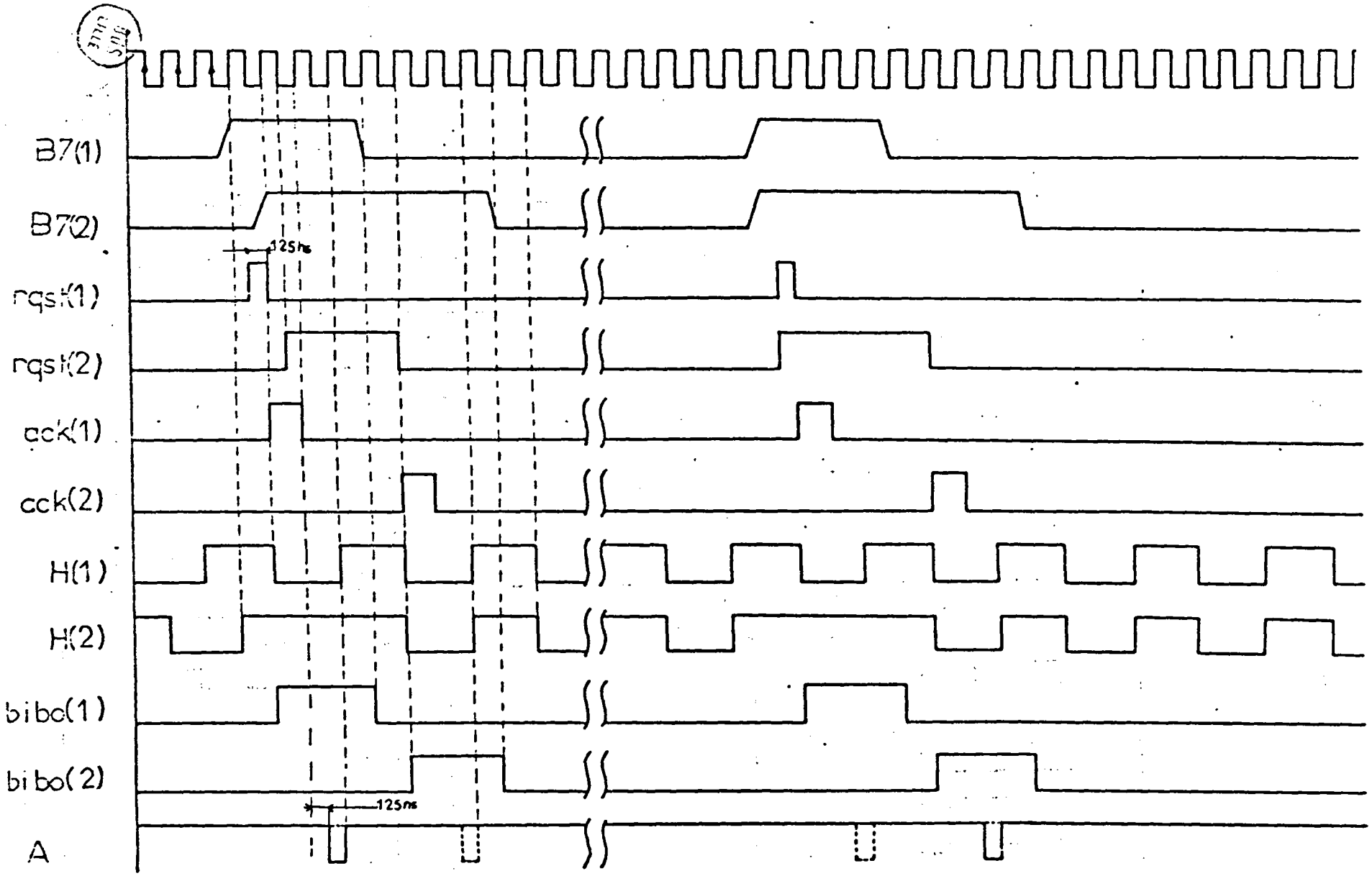


Figure III - 8



On peut substituer les signaux provenant du processeur numero 1 aux signaux provenant du processeur numero 2

diagramme des échanges correspondant à une demande d'accès à la mémoire commune formulée par les 2 processeurs en même temps

III-5 - CONCLUSION

Dans ce chapitre, nous avons présenté les différents modules constituant l'architecture générale du dispositif multiprocesseurs, de même, nous avons examiné les tâches relatives à chacun des automates en explicitant leurs fonctionnements et en présentant des organigrammes d'échanges relatifs à chaque automate tout d'abord, puis à l'ensemble du système.

Il nous reste donc à examiner la façon avec laquelle nous pouvons programmer et piloter notre système; ceci fait l'objet du chapitre IV.

o
o o

CHAPITRE IV - REALISATION & COUPLAGE DU SYSTEME MULTI-AUTOMATES A LA CONSOLE DE PROGRAMMATION

IV-1 - INTRODUCTION

La finalité du système multi-automates programmables est la commande d'automatismes logiques industriels.

Les échanges "multi-automates - PROCEDE" ce font, en général, par :

- des interfaces d'entrée et de sortie, qui réalisent à la fois un isolement galvanique, un filtrage et un changement de niveau d'énergie.

- un système d'alimentation et de servitudes.

Pour avoir une adaptation d'un système multi-automates à un automatisme aussi complexe qu'il soit, il faut ordonner, d'une façon astucieuse et cohérente, les différents programmes implantés dans les mémoires des automates.

Les moyens de dialogue entre l'opérateur et le système multi-automates, sont de deux sortes :

- Un outil de programmation et de mise au point : la console de programmation

- Les éléments spécifiques de test et de maintenance qui permettent à "l'automaticien" de suivre le fonctionnement du processeur.

Dans ce chapitre, nous allons étudier plutôt le premier outil de dialogue.

Les échanges homme-multi-automates programmable se font donc par l'intermédiaire de la console de programmation. Celle-ci est généralement autonome et permet la programmation de la mémoire de chaque automate, les fonctions fondamentales qui lui sont attribuées sont : l'écriture, la lecture et les modifications éventuelles dans les mémoires programmes.

A priori, une console de programmation comporte un clavier et un dispositif d'affichage à diodes luminescentes ou un tube cathodique et peut être reliée à une imprimante pour afficher ou éditer le programme moniteur des processeurs.

Il existe deux types de consoles : celles dites autonomes avec une mémoire vive dans laquelle vient s'implanter le programme de Gestion du Système et les consoles qui ne peuvent fonctionner qu'en liaison directe avec la mémoire programme du système. D'autres consoles comportent également un enregistreur de mini cassettes, bandes magnétiques ou encore de disques souples, mettent aussi à disposition une bibliothèque de programmes.

Certaines consoles contribuent à la surveillance et à l'entretien préventif du processus industriel en rendant compte de l'état des capteurs, des actionneurs et du traitement en cours.

IV-1-1 - CONSOLE DE PROGRAMMATION UTILISEE [10].[22]

Pour notre système Multi-automates, le rôle essentiel de la console de programmation est la préparation des programmes et leur introduction dans les mémoires des différents automates. Toutes les commandes nécessaires pour le pilotage des différents processeurs se font à partir de cette console.

- Choix du processeur avec lequel on converse
- Ecriture dans les différentes mémoires : mémoire de P1, mémoire de P2, mémoire de P3.
- Lecture du contenu mémoire des différents processeurs : P1, P2, P3.
- Donner l'ordre d'exécution à chacun des automates.

Nous avons établi un programme moniteur introduit dans la mémoire vive de la console de programmation qui nous permet de converser avec le système des automates suivant les modes cités ci-dessus.

Nous avons choisi le microcalculateur C.B.M 3032 comme outil de dialogue entre l'opérateur et le système Multi-automates programmables.

Le C.B.M. 3032 dispose d'un clavier semblable à celui d'une machine à écrire et d'un écran cathodique.

Les communications avec l'extérieur se font par quatre voies [10]

- Deux ports J1 et J2
- Un interface de cassette J3
- Un connecteur expansion mémoire du microcalculateur.

Les ports J1 et J2 sont connectés en parallèles et respectivement au bus de commandes et au bus Instructions, comme le montre la figure suivante :

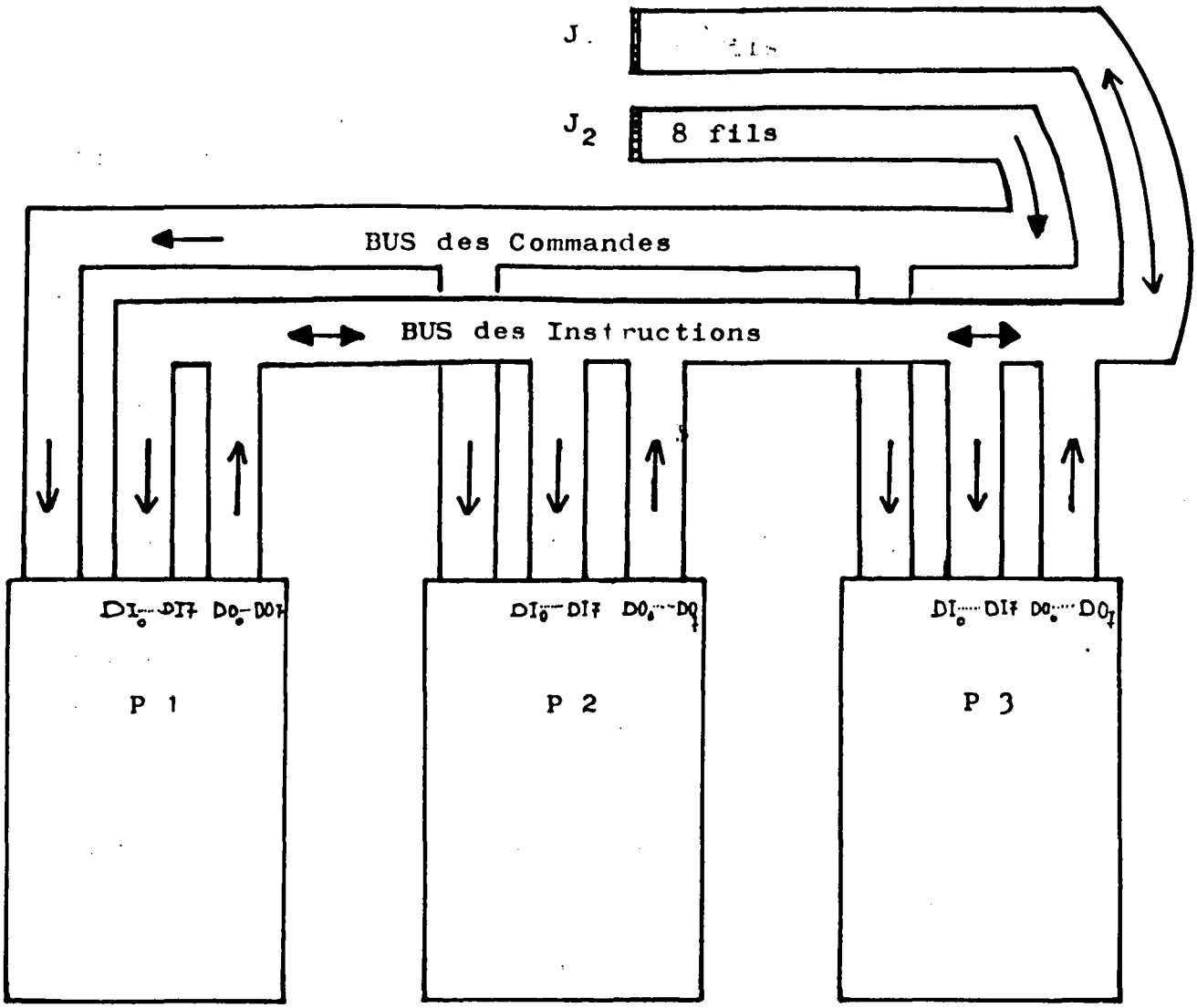


Figure IV - 1

Le bus de commandes est un bus unidirectionnel dans le sens Microcalculateur vers le système Multi-automates. Le bus Instruction est un bus bidirectionnel dont le sens et l'affectation sont régis par le mot de commande issu du programme moniteur.

Après une description sommaire de la console de programmation utilisée, nous allons parler du langage de programmation du microcalculateur.

IV-2 - LANGAGE DE PROGRAMMATION [23]

Le microcalculateur doit recevoir des instructions précises et on doit lui fournir la séquence des opérations à effectuer pour accomplir une tâche spécifique.

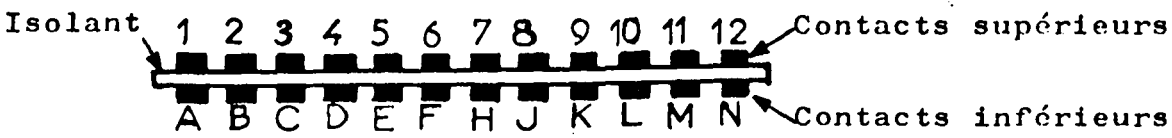
Le langage de programmation du C.B.M. 3032 utilisé pour le pilotage du système multi-automates programmables est le BASIC.

Tous les mots de commandes et d'instructions du système sont introduits en code décimal puis traduits et envoyés en code binaire vers le bus B.I.P (voir figure II-1 page II.7). Par l'intermédiaire des deux connecteurs J1 et J2.

IV-3 - DESCRIPTION DU BUS DE CONNECTIONS "CONSOLE-MULTI-AUTOMATES"

Nous présentons maintenant les divers signaux d'échange au niveau de J1 et J2.

IV-3-1 - SCHEMA SYNOPTIQUE [10]



Vue simplifiée des connecteurs J1 et J2

Le connecteur (J1) a 24 contacts pour lesquels nous dressons ici le tableau du rôle des signaux utilisés.

Broches du connecteur J1 du C.B.M	Mnémonique du signal	Définition du signal
contacts supérieurs		
1	DI01	donnée (E/S) n° 1
2	DI02	donnée (E/S) n° 2
3	DI03	donnée (E/S) n° 3
4	DI04	donnée (E/S) n° 4
12	GND	masse chassis et tresse de protection du câble
contacts inférieurs		
A	DI05	donnée (E/S) n° 5
B	DI06	donnée (E/S) n° 6
C	DI07	donnée (E/S) n° 7
D	DI08	donnée (E/S) n° 8
N	GND	masse pour les données de (DI01 à DI08)

De même, le connecteur J2 à 24 contacts pour lesquels nous donnons, ci-contre, la fonction des signaux utilisés.

Broches du connecteur J2 du C.B.M	Mnémonique du signal	Définition du signal
contacts supérieurs		
1	GND	Masse logique
11	GND	Masse logique
12	GND	Masse logique
contacts inférieurs		
A	GND	Masse logique
B	CA1	entrée de détection de fronts du VIA 6522
C	PA0	Signaux d'E/S vers les périphériques qui peuvent être programmés indépendamment les uns des autres pour se comporter en entrées ou en sorties
D	PA1	
E	PA2	
F	PA3	
H	PA4	
J	PA5	
K	PA6	
L	PA7	
M	CB2	Broche d'E/S spéciale de VIA
N	GND	Masse logique

Les signaux utilisés dans notre structure sont PA0 ..... PA7 qui sont des E/S programmables du C.B.M. Ces signaux bidirectionnels sont issus d'un VIA 6522 de Technologie MOS situé à une adresse déterminée; on se reportera à la description détaillée en [24].

Après avoir présenté les deux connecteurs d'échanges avec l'extérieur du C.B.M, nous allons préciser les

liaisons de ces périphériques avec le système Multi-automates.

IV-3-2 - SIGNAUX de SERVICE & LIAISONS AVEC le SYSTEME MULTIPROCESSEURS

Comme nous montre la figure IV-1 , les deux grandes voies de dialogue entre le système multi-automates et la console de programmation, sont le BUS des Commandes et le BUS Instructions.

Nous avons établi un programme moniteur qui va nous permettre l'envoi ou la réception d'instructions sur ces deux BUS, par l'intermédiaire des deux canaux J1 et J2.

Chaque instruction est envoyée ou reçue sous forme d'un mot de huit bits. Pour la mise en marche du système et pour la compréhension des échanges C.B.M. 3032 Multi-automates, il nous a paru nécessaire de préciser les différents signaux de service ainsi que les interliaisons : voir figure IV-2.

Des Buffers ont été prévus au niveau du BUS de commandes afin d'augmenter le "Fan-out" des différents signaux.

Tous les transferts d'informations se font en binaire.

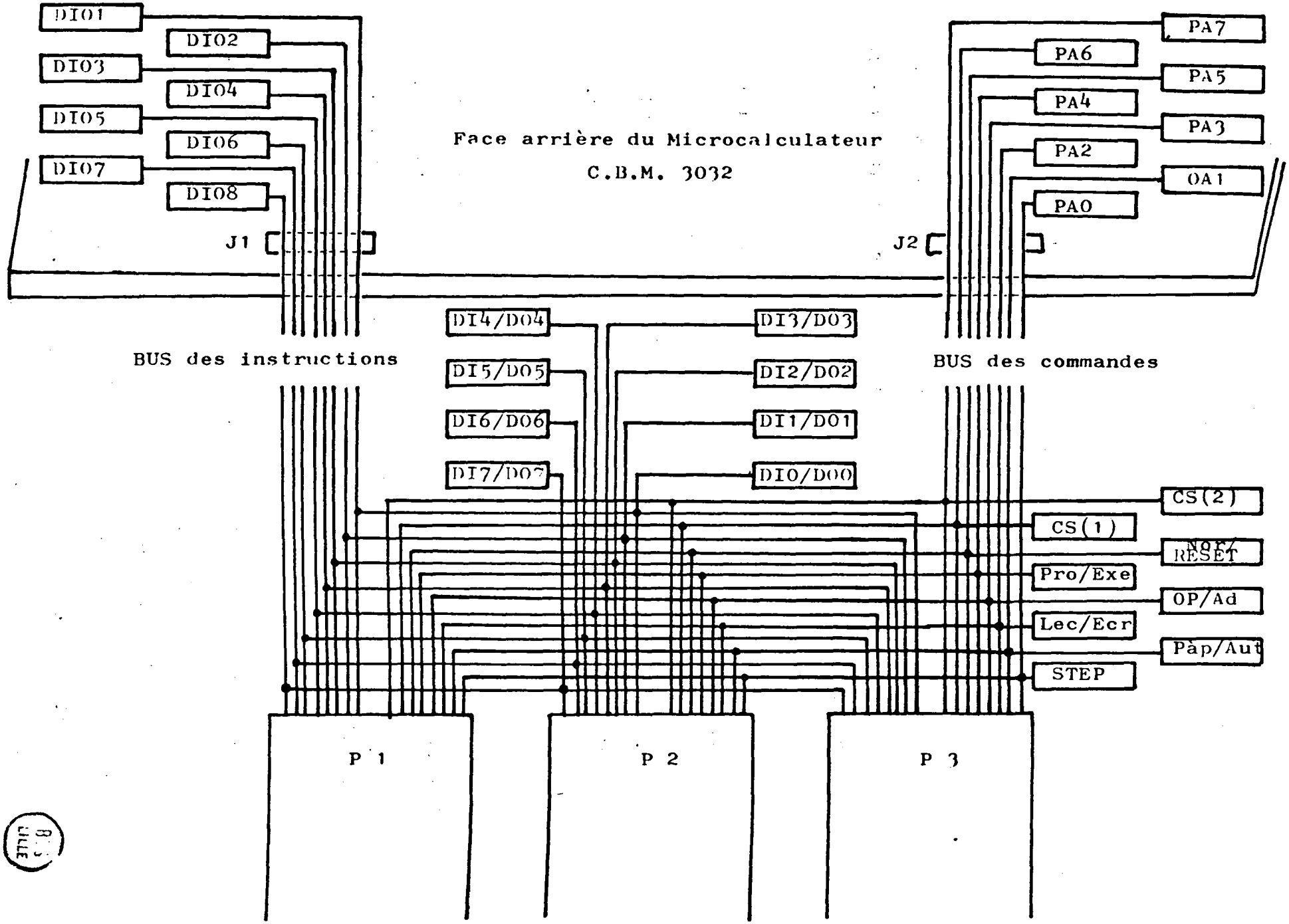

DI01
DI02
DI03
DI04
DI05
DI06
DI07
DI08
J1
Face arrière du Microcalculateur
C.B.M. 3032
PA7
PA6
PA5
PA4
PA3
PA2
PA1
PA0
J2
BUS des instructions
BUS des commandes
DI4/DO4
DI5/DO5
DI6/DO6
DI7/DO7
DI3/DO3
DI2/DO2
DI1/DO1
DI0/DO0
CS(2)
CS(1)
RESET
Pro/Exe
OP/Ad
Lec/Ecr
Pap/Aut
STEP
P 1
P 2
P 3

Figure IV - 2

Les fonctions des différents signaux ont été précisées dans le chapitre III.

IV-4 - DEFINITION DES MODALITES DE DIALOGUE & DE CONVERSATION ENTRE LA CONSOLE & LE SYSTEME MULTI-AUTOMATES

Dans ce paragraphe, nous allons étudier toutes les possibilités de communications qui existent entre la console de programmation et chaque automate.

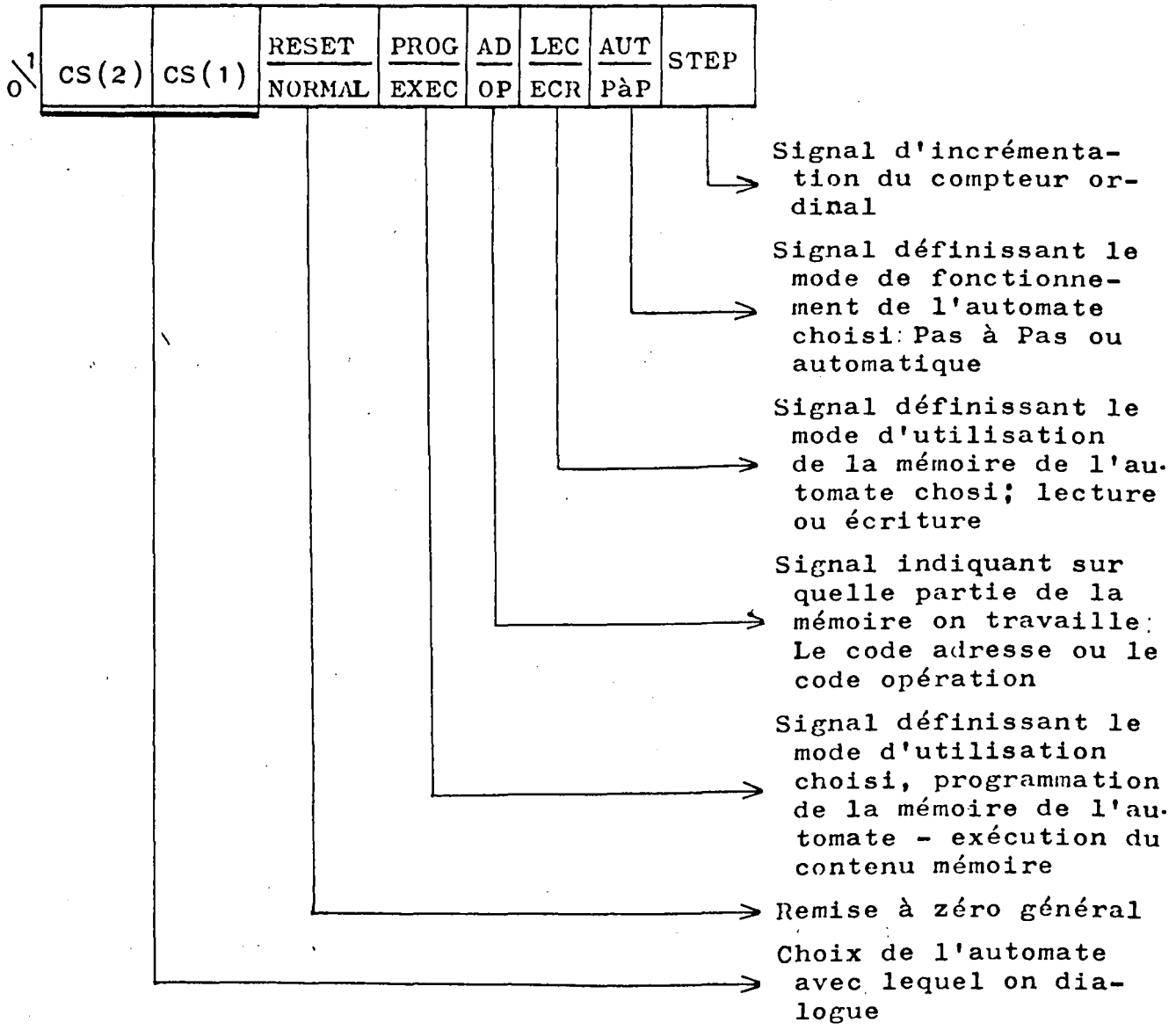
Le message sera caractérisé par un code qui va nous préciser :

- l'automate avec lequel on travaille : P1, P2 ou P3
- le mode de fonctionnement de l'automate : écriture, lecture ou exécution.

Toute cette étude va concerner le BUS de commandes sachant que le BUS d'instructions sera un ensemble de huit données qui vont transiter du connecteur J1 au BUS Instruction ou inversement.

IV-4-1 - MODE ECRITURE

Il s'agit d'écrire dans les différentes mémoires le contenu du BUS Instruction. Le mot de commande correspondant comprend huit signaux dont la signification est donnée par le schéma suivant :



Pour écrire en mémoire Pi, nous procéderons en deux étapes :

- Présentation de l'information à écrire (sous forme de huit bits) sur le BUS Instruction
- Envoi du mot de commande

Deux cas peuvent alors se présenter :

- Ecriture du "code Adresse"
- Ecriture du "code opération"

IV-4-1-1 - ECRITURE DU CODE ADRESSE EN MEMOIRE Pi

Pour écrire le code Adresse en mémoire P i (P1, P2 ou P3), il faut envoyer, en fait, une séquence de trois mots de commandes (voir tableau récapitulatif page IV.13).

Chaque séquence d'écriture incrémente automatiquement le compteur ordinal et le positionne à l'adresse qui suit.

IV-4-1-2 - ECRITURE DU CODE OPERATION EN MEMOIRE Pi; (P1, P2, P3)

Le code opération se présente sous forme de quatre bits (I0,I1, I2, I3). L'information à écrire, dans ce cas, se présente sur les quatre bits de poids faible du BUS Instruction.

Pour écrire un code opération en mémoire Pi (P1, P2, P3), il faut envoyer une séquence de trois mots de commandes (voir tableau récapitulatif, page IV.13).

Chaque séquence d'écriture du code opération, incrémente automatiquement le compteur ordinal.

TABLEAU RECAPITULATIF DES CODES D'ECRITURE DANS

LES MEMOIRES PROCESSEURS (P1, P2, P3)

Processeur	Code à écrire en mémoire processeur	Séquence à envoyer pour l'écriture en mémoire Pi en	
		Hexadécimal	Code décimal
P1	Code Adresse	58-59-58	88-89-88
	Code Opération	50-51-50	80-81-80
P2	Code Adresse	98-99-98	152-153-152
	Code Opération	90-91-90	144-145-144
P3	Code Adresse	18-19-18	24-25-24
	Code Opération	10-11-10	16-17-16

IV-4-2 - MODE DE LECTURE

Il s'agit de lire le contenu des mémoires desdifférents automates : P1, P2 et P3.

En-effet, il est prudent de vérifier le contenu mémoire d'un automate avant l'exécution du programme qu'il contient. La lecture s'effectue en code décimal sur l'écran du C.B.M. 3032.

L'opération de lecture se fait en trois étapes :

- Envoi du mot de commande sur le BUS de commande

- L'information à lire se présente alors sur le BUS Instruction

- L'impression du contenu de périphérique J1 sur l'écran cathodique en code décimal.

Le mot de commande correspondant à la lecture du contenu mémoire d'un automate, comprend huit signaux dont la signification a été donnée en IV-4-1, page IV.11.

Pour chaque automate, deux cas peuvent se présenter, la lecture du code adresse, ou la lecture du code opération. Bien entendu, la lecture du Code opération va se faire sur les quatre bits de poids faible du Bus Instruction.

Nous dressons, ci-joint, le tableau récapitulatif des mots de commandes pour la lecture dans les différents automates.

Automate	Contenu Mémoire à Lire	Mot de commande à envoyer		Séquence d'incrémentation du C.0
		Code Héxadécimal	Code décimal	Code décimal
PC1	Code adresse	5C	92	92-93-92
	Code Opération	54	84	84-85-85
PC2	Code Adresse	9C	156	156-157-156
	Code Opération	94	148	148-149-148
PS	Code Adresse	1C	28	28-29-28
	Code Opération	14	20	20-21-20

Une fois le mot de commande envoyé, l'adresse du Compteur ordinal de l'automate choisi, reste figée. Pour

lire le contenu mémoire de l'adresse suivante. Il suffit d'incrémenter le compteur ordinal suivant les séquences d'incrémentation correspondantes à chaque automate.

IV-4-3 - MODE EXECUTION

Ce mode consiste à exécuter le ou les programmes contenus dans les mémoires des différents automates. Dans le système multi-automates proposé, on peut mettre en exécution un automate seul, deux ensembles en parallèle , ou tous les automates en parallèle.

On distingue deux modes d'exécution :

- le mode d'exécution en automatique
- le mode d'exécution en Pas à Pas.

IV-4-3-1 - EXECUTION EN AUTOMATIQUE

Dans ce mode, le système est livré à lui-même, la scrutation du programme s'effectue d'une façon séquentielle, autonome et cyclique. La vitesse d'exécution est imposée par un oscillateur dont la fréquence varie de zéro à 1 MHZ. L'intervention de l'opérateur se situe au niveau du démarrage du processeur ou de l'arrêt de ce dernier.

IV-4-3-2 - EXECUTION EN PAS-à-PAS

Pour le mode d'exécution en Pas à Pas, c'est l'opérateur qui est maître de la cadence de scrutation du pro-

gramme. Ce mode est très intéressant du point de vue du développement des programmes. En-effet, nous pouvons suivre l'évolution des signaux fondamentaux, Instruction par Instruction et ainsi détecter les anomalies qui peuvent apparaître lors de l'exécution d'un programme.

IV-4-3-3 - LES MOTS DE COMMANDES

A chaque mode d'exécution choisi, correspond un mot de commande.

Dans le système multiprocesseurs, nous avons prévu quatre façons pour l'exécution de l'ensemble des programes dans les différents automates.

- Exécution du programme implanté dans P1 seul
- Exécution du programme implanté dans P2 seul
- Exécution du programme implanté dans P3 seul
- Exécution de l'ensemble des programmes implantés dans P1, P2, P3, en parallèle, deux à deux ou les trois ensemble.

Ce mot de commande est constitué par huit signaux dont nous avons donné la signification au paragraphe IV-4-1 page IV.11.

TABLEAU RECAPITULATIF DES MOTS DE COMMANDE POUR L'EXECUTION DES PROGRAMMES DANS LES DIFFERENTS PROCESSEURS

Automate	Mode d'exécution	Mot de commande à envoyer	
		Code Héxadécimal	Code décimal
PC1	Automatique	46	70
	Pas à pas	44	68
P2	Automatique	86	134
	Pas-à-Pas	84	132
P3	Automatique	06	6
	Pas-à-pas	04	4
P1, P2, et P3 ensemble	Automatique	C6	198
	Pas-à-pas	C4	196

REMARQUE : Les mots de commande ne sont pas uniques. En-effet, pour chaque mode d'exécution, nous donnons en annexe, page une suite de mots qui donne la même fonction.

En exécution automatique, l'incrémentation du compteur ordinal, s'effectue par une horloge dès l'envoi du mot de commande.

Par contre, dans le mode d'exécution en pas-à pas,

l'incrémentation du compteur ordinal se fait par l'envoi sur le BUS de Commandes d'une séquence de mots de commandes. Nous donnons, en annexe, page un récapitulatif des séquences à envoyer sur le BUS de commandes, pour l'incrémentation dans le mode exécution en Pas-à-Pas des différents processeurs.

IV-5 - ORGANIGRAMME DE GESTION DES DIFFERENTS PROCESSEURS

Dans ce paragraphe, nous nous sommes attachés à la construction d'un organigramme de gestion du système multi-automate.

Cette construction a été guidée par tois soucis :

- Possibilité d'arrêt de système multi-automates à n'importe quel moment
- Une implantation rapide des programmes à écrire
- Une rapidité dans le pilotage des différents processeurs

Pour faciliter la compréhension de l'organigramme général, nous l'avons divisé en trois grandes parties :

- Organigramme relatif au pilotage de P1
- Organigramme relatif au pilotage de P2
- Organigramme relatif au pilotage de P3

Le programme correspondant est le programme moniteur du système multi-automates (voir Annexe, page A.6). Nous l'implanterons dans la mémoire vive du microcalculateur C.B.M. 3032 et il doit nous permettre le pilotage et la conversation avec le système multi-automates.

Ce programme sera réalisé en BASIC et sauvegardé sur cassette.

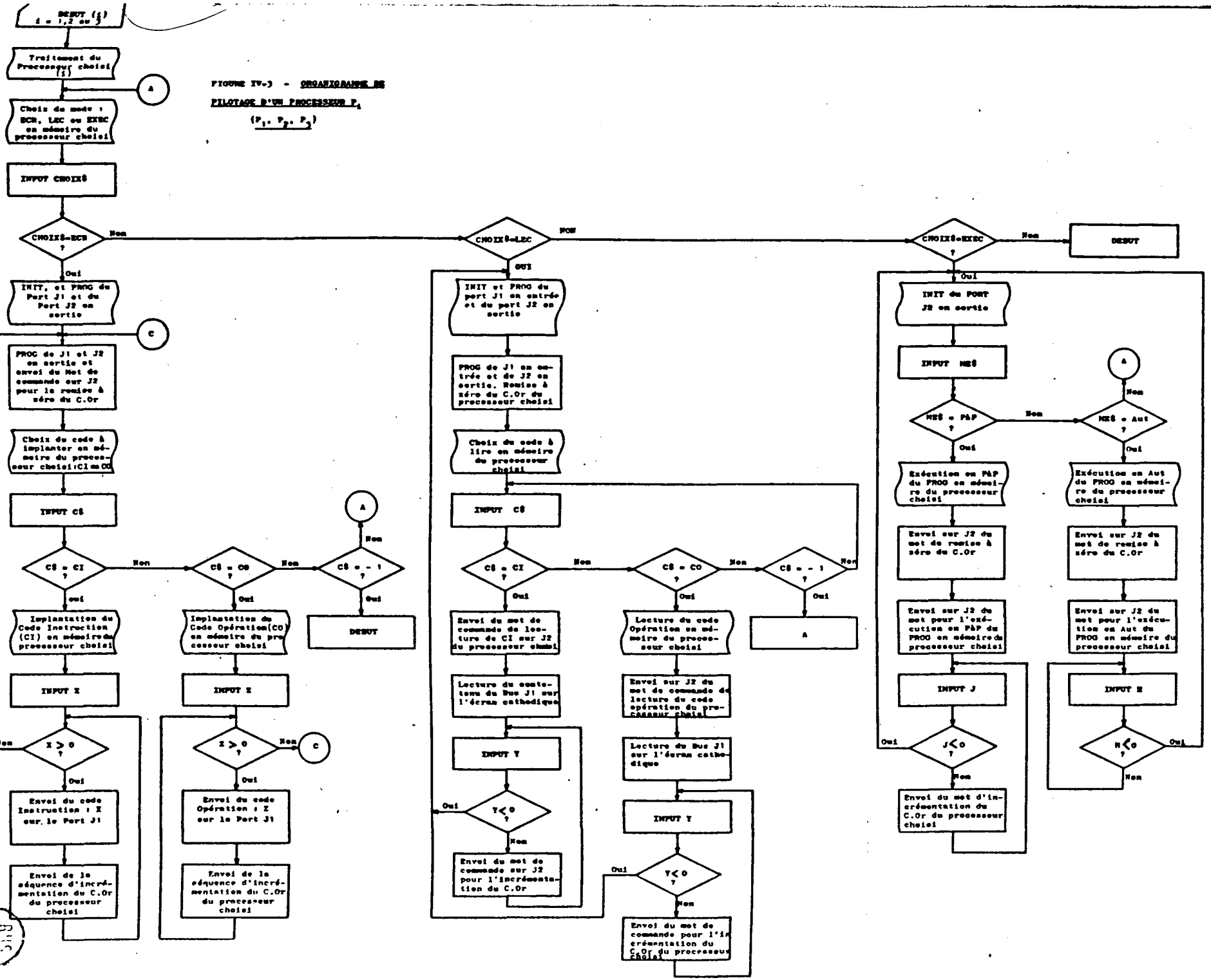


FIGURE IV.7) - ORGANIGRAMME DE PILOTAGE D'UN PROCESSEUR P_i (P_1, P_2, P_3)

Les variables utilisées dans cet organigramme sont :

P1 : Processeur (1) ou automate programmable (1)

J1 : Port d'E/S du microcalculateur C.B.M. 3032 relié au BUS Instructions du multiprocesseur

J2 : Port d'E/S du microcalculateur C.B.M. 3032 relié au BUS de commandes du multiprocesseur

C1 : Code Instructions

CO : Code Opération

P$, CHOIX $, C $, ME $: variables définissant des chaînes de caractères

X, Y, Z : variables quelconques

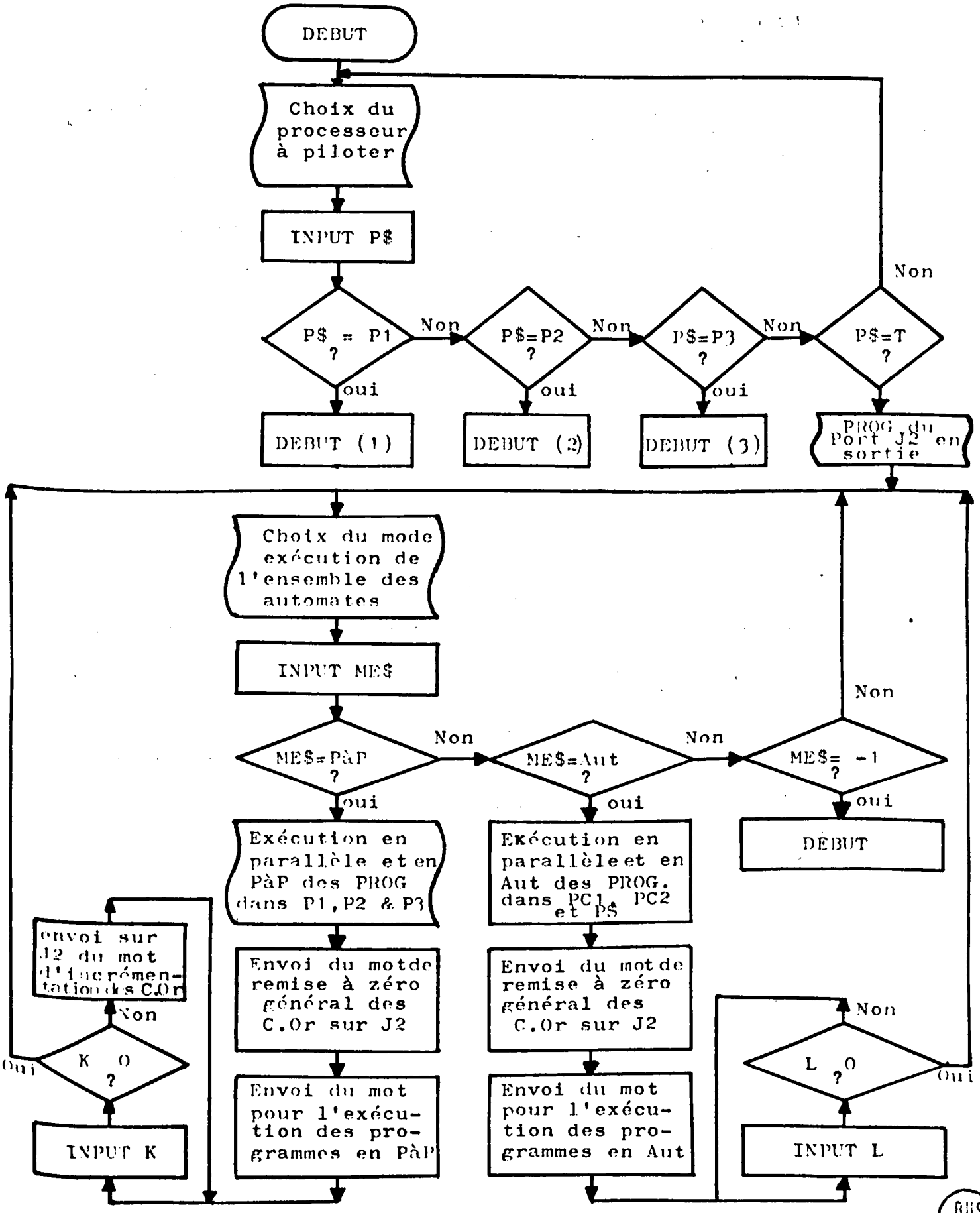
C.Or: Compteur ordinal

P2 : Processeur (2) ou automate programmable (2)

P3 : Processeur (3) ou automate programmable (3)

Nous n'avons pas favorisé les modalités avec tel ou tel processeur, ils sont tous au même niveau. Ce qui diffère d'un processeur à un autre, ce sont surtout les mots de commandes.

IV-6 - ORGANIGRAMME DE PILOTAGE DU MULTIPROCESSEURS



Dans cet organigramme, DEBUT 1, DEBUT 2, DEBUT 3, sont respectivement les organigrammes de pilotage de P1, P2 et P3.

Le Programme "moniteur" correspondant à cet organigramme est donné en annexe, page A.6

Tous les programmes sont donnés en BASIC.

IV-7 - CONCLUSION

Dans ce chapitre, nous avons présenté la console de programmation utilisée ainsi que le langage de programmation.

De même, nous avons défini les modalités de dialogue et de conversation entre la console et le système multiprocesseur.

Dans le chapitre sui suit, nous allons étudier l'implantation d'un outil de description d'un automatisme logique sur un système multiprocesseurs.

o
o o

CHAPITRE V - LES METHODES D'IMPLANTATIONS DES OUTILS DE DESCRIPTION SUR SYSTEMES MULTI-PROCESSEURS

V-1 - INTRODUCTION

Nous avons présenté, dans les chapitres précédents, la structure générale du système multi-automates programmable et le rôle joué par chaque processeur. Le but que nous nous sommes assigné est la mise en oeuvre du Grafcet sur notre dispositif. L'implantation du Grafcet est rendue plus délicate par la notion d'évolution simultanée et dépend de la façon d'utiliser les diverses structures possibles suivantes :

- un automate programmable classique en monoprocesseur
- un automate programmable avec une structure multiprocesseursbanalisés
- et, enfin, une structure multiprocesseurs privilégiant un processeur particulier

Suivant la structure adoptée, il est nécessaire d'effectuer une décomposition du Grafcet, soit dans l'espace, soit dans le temps.

En utilisation monoprocesseur, en effet, il faudra réaliser un partage du temps en fonction des évolutions

éventuellement simultanées. Une structure à processeurs multiples banalisés nous impose, soit une décomposition en graphes d'état, chaque G.E étant implanté sur un processeur, soit une décomposition relativement aux entrées-sorties, ou encore, le produit des deux.

La structure particulière adoptée avec processeur séquentiel et processeurs combinatoires entraîne une décomposition évidente. Encore faut-il décomposer, en partie, les traitements des réceptivités et des sorties.

Le but de la décomposition d'un GRAFCET est de le recouvrir par un ensemble de sous-grafcets permettant d'aboutir à la même relation d'entrées-sorties que le Grafcet initial. Cette décomposition augmente le nombre de sous-machines correspondant à chacun, mais on peut affecter à une seule machine plusieurs sous-grafcets respectant le synchronisme entre les différents processeurs.

D'autres critères de décomposition peuvent être retenus, comme celui permettant un nombre minimum de processeurs et ceci en recherchant des graphes d'état susceptibles d'être implantés sur un même processeur; ce critère réduit le nombre de processeurs, mais il augmente le temps de cycle de traitement.

Pour des raisons technologiques, nous pouvons introduire, dans chaque cas, des contraintes relatives :

- à la minimisation de la densité de traitement (la capacité mémoire)

- à la minimisation du temps de traitement (le temps de réponse)

- à la minimisation du nombre de processeurs utilisés.

Mais il est nécessaire de s'assurer que le comportement global de l'ensemble des sous-grafcets correspond bien à celui du grafcet initial; pour cela, il est parfois nécessaire :

- de modifier les conditions d'évolution qui assurent le synchronisme entre les processeurs

- de créer des variables de synchronisation qui rendent compte de l'état des sous-machines (processeurs)

Dans ce qui suit, nous allons présenter une méthode de décomposition du Grafcet en graphes d'état, puis, nous montrerons les implantations suivant les différentes configurations.

V-2 - RAPPEL D'UNE METHODE DE DECOMPOSITION D'UN GRAFCET EN GRAPHES D'ETAT [3], [5]

Nous donnons, ci-dessous, les aspects essentiels de cette méthode en rappelant quelques définitions et l'algorithme proposés par

V-2-1 - DEFINITION D'UN GRAPHE D'ETAT

Un graphe d'état est un GRAFCET possédant, au maximum, une place active (voir chapitre I)

Un graphe est défini par l'ensemble $G = (E, T, f_E, f_S)$ où E est l'ensemble des étapes, T est l'ensemble des transitions, f_E et f_S sont deux applications définies respectivement de (ExT) et (TxE) dans le doublet (0,1) :

$f_E(e_i, t_j) = 1$ s'il existe un arc reliant (e_i à t_j)

$= 0$ s'il n'existe pas d'arc reliant e_i à t_j

$f_S(t_k, e_l) = 1$ s'il existe un arc reliant t_k à e_l

$= 0$ s'il n'existe pas d'arc reliant t_k à e_l

V-2-2 - SOUS-GRAPHES & REALISATION

Soit le graphe suivant $G = (E, T, f_E, f_S)$ et soit $E' \subset E$ et $T' \subset T$ et soit $f_{E'}$, $f_{S'}$ les restrictions de f_E et f_S respectivement dans E'xT' et T'xE'. Un sous-graphe est défini par $G' = \{(E', T', f_{E'}, f_{S'}), M_o\}$ avec M_o le marquage initial.

On effectue une partition de l'ensemble des étapes E par PT = $(A_1, A_2, \ldots\ldots A_n)$ avec

$$\left\{ \bigcup_{i=1}^{n} A_i = E \text{ et } \forall i, \forall j \quad i \neq j \quad A_i \cap A_j = \emptyset \right\}$$

L'objet de la méthode proposée est de trouver une partition PT tel que à partir du marquage initial chaque bloc ait, au maximum, une place active.

Avant de rappeler l'algorithme de décomposition, nous précisons les éléments fondamentaux utilisés.

soit e une étape, on définit EA l'ensemble des étapes atteintes par e sans traverser de transition "ET"

On établit la partition $PT^{(k)} = \left\{ E^{(k)}, A_1^{(k)}, A_2^{(k)} \ldots A_{q^k}^{(k)} \right\}$

k étant le pas d'examen des transitions; les blocs A_i^k sont des sous-Grafcets ne comportant pas de transition "ET" et $E^{(k)}$ la partie à décomposer au pas k qui peut inclure éventuellement des noeuds "ET".

De même, on définit l'ensemble des transitions à examiner au pas k, et qui sont classées en trois classes :

$$Y_a^{(k)} = \left\{ t_i : t_i \neq \text{Noeud "ET"} \right\}$$

$$Y_b^{(k)} = \left\{ t_j : t_j = \text{Noeud "ET" distributif} \right\}$$

$$Y_c^{(k)} = \left\{ t_l : t_l = \text{Noeud "ET" autre} \right\}$$

A la fin du traitement, on récupère plusieurs blocs sur lesquels on applique les propriétés de constituants connexes.

V-2-3 - ALGORITHME DE DECOMPOSITION D'UN GRAFCET EN GRAPHES D'ETAT

La procédure de décomposition d'un Grafcet en graphes d'état peut être alors résumée par l'algorithme suivant :

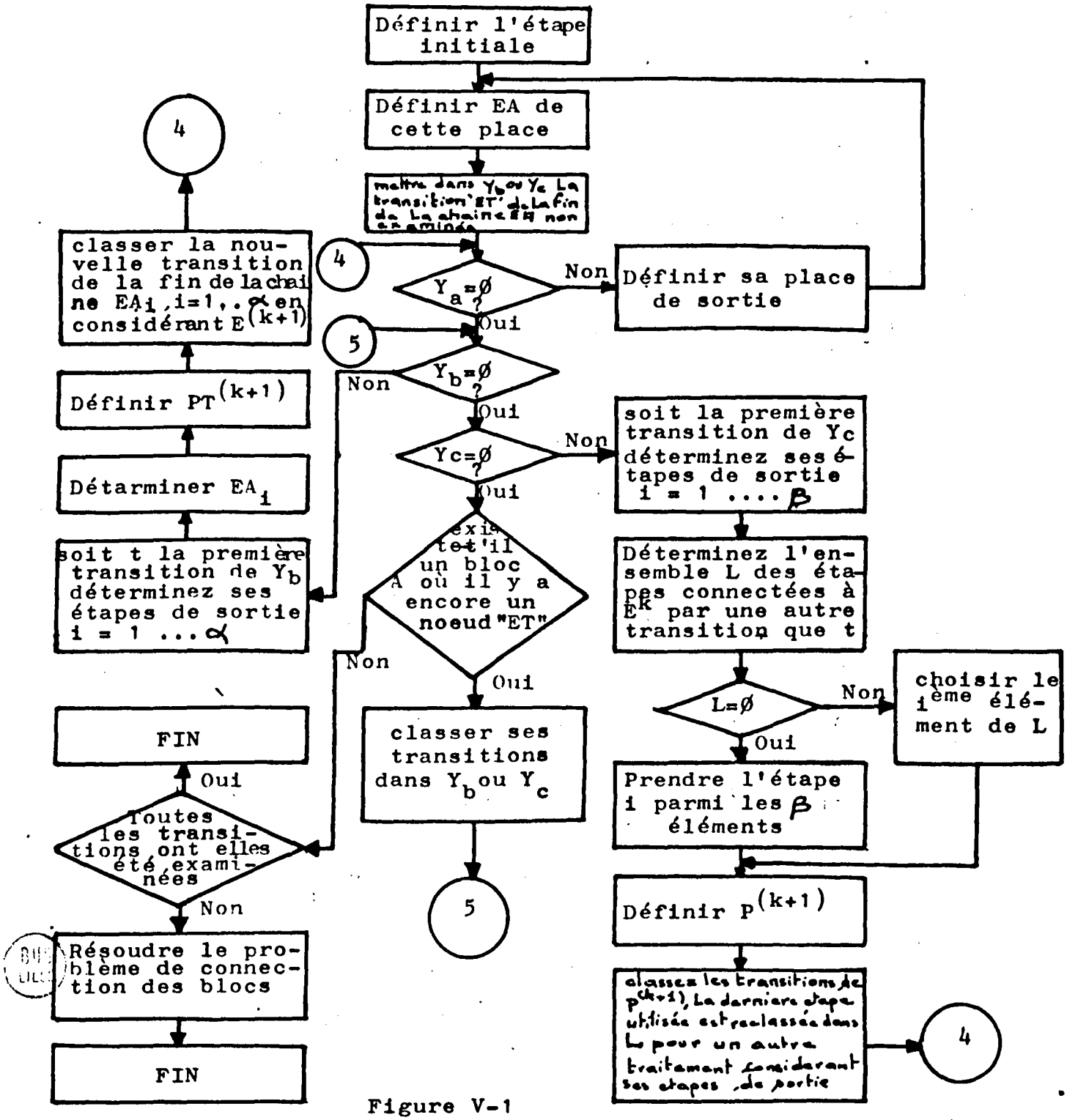


Figure V-1

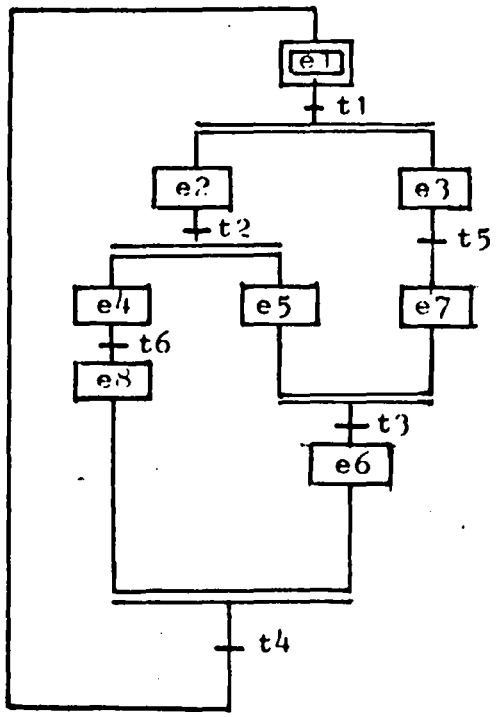
Cet algorithme est basé sur l'examen de toutes les transitions du GRAFCET sans tenir compte des réceptivités liées à ces transitions. Cette méthode de décomposition est applicable pour un Grafcet de type 1 :

qui est un Grafcet sans boucle, sans conflit et où les évènements sont indépendants de l'activité des étapes et tel que l'analyse des diverses possibilités de franchissements prises une à une, sur la structure non interprétée, ne montre aucune réactivation d'étape.

Néanmoins, cette méthode peut être appliquée à un Grafcet général en lui apportant un complément d'informations relatives à l'interprétation du Grafcet.

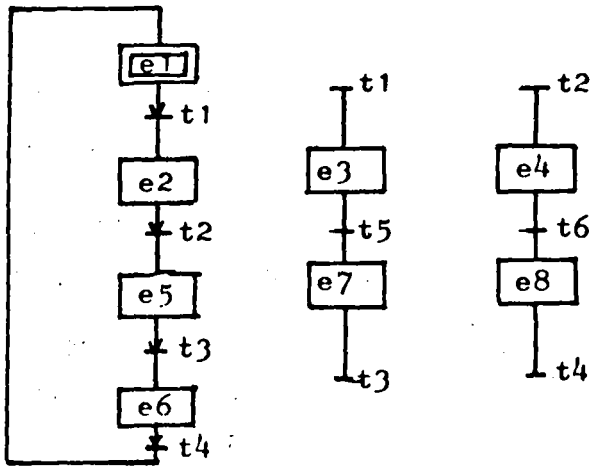
V-2-4 - EXEMPLE 1

Soit à effectuer une décomposition en graphes d'état du Grafcet suivant :



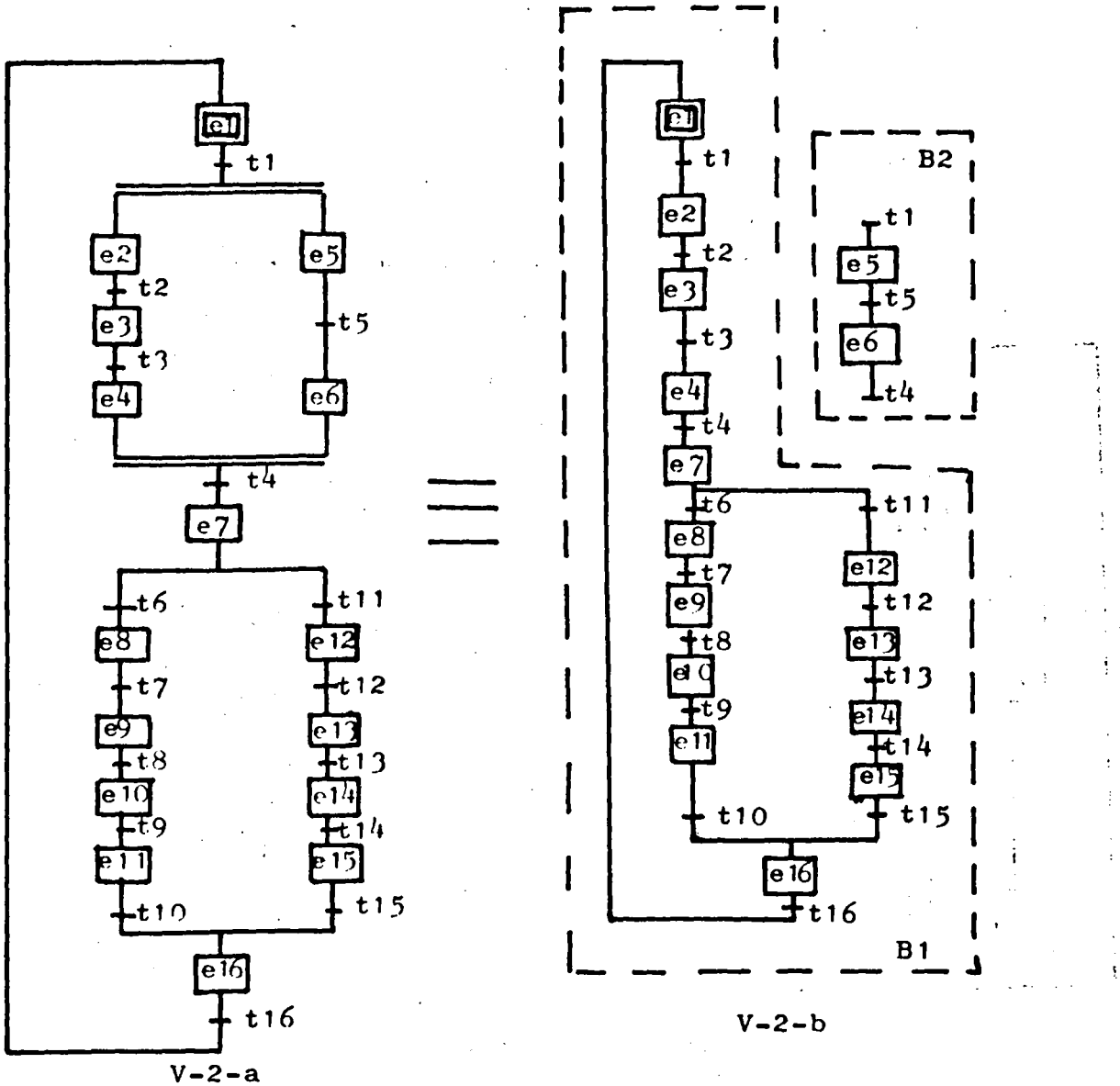
La partition initiale et triviale au pas 1 est $PT^{(1)}=[E]$ l'ensemble de toutes les étapes du Grafcet.
La place initiale est $e_1, t_1 \in Y_b^{(1)}=\{t_1\}$ les places de sortie de t_1 sont $\{e_2, e_3\}$ nous prenons $e_2 \in E^{(2)}$.
soit alors $PT^{(2)}=\{E^{(2)}, (e_3,e_7)\}$
et $t_2 \in Y_b^{(2)}$ et $t_3 \in Y_c^{(2)}$
considérons t_2, ses places de sortie sont e_4 et e_5 nous déduisons donc (e_4, e_8) et $(e_5) \in E^{(3)}$ et $t_4 \in Y_c^{(3)}$
on obtient finalement la partition
$PT=\{(e_1,e_2,e_5,e_6),(e_3,e_7),(e_4,e_8)\}$

Nous déduisons donc les blocs suivants :



V-2-5 - EXEMPLE 2

Soit à décomposer le GRAFCET de la figure V-2-a



V-2-a

V-2-b

Figure V-2

La décomposition de ce Grafcet suivant l'algorithme de la figure V-1 nous donne les deux blocs (B_1, B_2) avec $B_1 = (e_1, e_2, e_3, e_4, e_7, e_8, e_9, e_{10}, e_{11}, e_{16}, e_{12}, e_{13}, e_{14}, e_{15})$ et $B_2 = (e_5, e_6)$

La décomposition d'un tel Grafcet, suivant B_1 et B_2 n'est possible que si les transitions T_6 et T_{11} sont disjonctives.

V-3- SYNCHRONISME et ASYNCHRONISME

Les problèmes de synchronisation sont multiples au niveau d'un Grafcet relativement :

- à la prise en compte des E/S
- à l'évolution du marquage du Grafcet

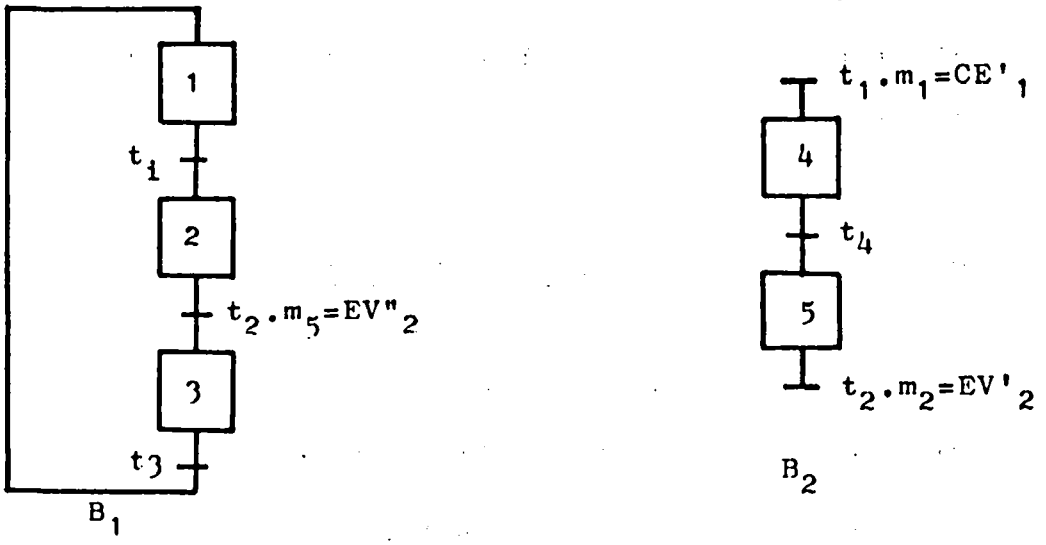
Les structures multiprocesseurs ajoutent encore d'autres difficultés :

- au niveau matériel (pilotage de l'ensemble des processeurs par une horloge unique ou chacun une horloge).
- au niveau de l'acquisition des entrées et de l'affectation des sorties en même temps ou non sur les divers processeurs d'une part et sur un seul processeur, d'autre-part.
- et, enfin, au niveau de la synchronisation des échanges d'informations entre les divers processeurs.

V-3-1 - SYNCHRONISME STRUCTURAL [1],[2]

Nous avons très vite abandonné l'idée d'une structure multi-automates avec une horloge par processeur : structure asynchrone.

En-effet, prenons l'exemple suivant où les deux graphes d'état (B_1 et B_2) en relation, sont implantés sur deux processeurs différents respectivement P_1 et P_2.



Si on suppose que le temps de cycle de P_2 est deux fois plus grand que P_1. Le temps de cycle de traitement de B_2 va être plus petit que celui de B_1. Dans le cas de la configuration : étape 2 de B_1 active et étape 5 de B_2 active et EV'_2 est vraie; l'étape 5 va être désactivée avant que P_1 ne puisse avoir le temps de calculer EV''_2. On se retrouve alors dans une situation de blocage complet de l'activité puisque EV''_2 ne sera jamais vrai et l'étape 2 ne pourra plus être désactivée.

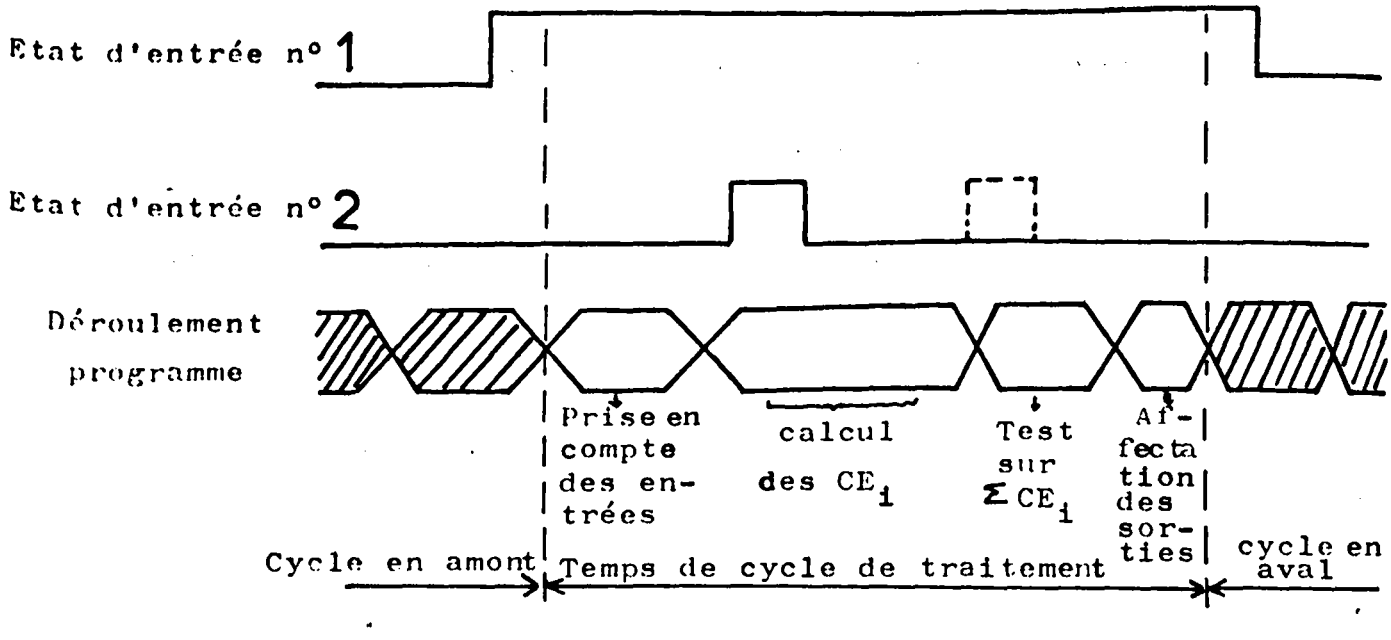
Dans cet exemple, le fait qu'une des deux transitions simultanément franchissables ne soit pas franchie, montre bien l'incohérence des traitements parallèles par une structure matérielle asynchrone, nous devons donc synchroniser le temps de cycle de traitement et retenir

la solution d'un système de modules évoluant en parallèle et pilotés par une horloge principale.

V-3-2 - PRISE EN COMPTE DES ENTREES [12]

La prise en compte de l'état des entrées se fait à un instant donné par rapport à l'exécution du programme. Dans un mode asynchrone la prise en compte d'une entrée se fait au fur et à mesure de son utilisation dans le traitement. Le fait que cette entrée puisse changer d'état dans un même cycle de traitement, rend incompatible l'utilisation de ce mode pour l'implantation du Grafcet dans notre système.

Par contre, dans un mode synchrone la prise en compte de toutes les entrées au début du cycle de traitement élimine cette incompatibilité. Toutefois, le mode synchrone n'est parfaitement valable que dans l'hypothèse où la variation de l'état d'une entrée se fasse dans un temps supérieur au temps du cycle de traitement global (voir la figure suivante).



...

Etant donné la faible vitesse d'évolution de la plupart des processus industriels, le bon fonctionnement de l'automatisme reste valable dans la plupart des cas.

V-3-3 - ARRET D'URGENCE

Sur certaines machines sur lesquelles on peut effectuer un saut d'ensemble d'instructions dans le cycle de traitement, on peut utiliser l'arrêt d'urgence pour raccourcir le temps d'arrêt (l'arrêt d'urgence sera une entrée privilégiée).

V-3-4 - AFFECTATION DES SORTIES

En ce qui concerne les sorties, à tout instant nous avons une image de l'état des sorties représenté par des variables internes. L'affectation des sorties peut être effectuée soit, juste après leur établissement une à une d'une façon asynchrone, soit en bloc en fin de cycle programme ; d'une façon synchrone.

V-4 - PROCEDURE D'IMPLANTATION UTILISANT UN SEUL PROCESSEUR

Notre étude, dans ce paragraphe, est consacrée au traitement sur un même processeur d'un Grafcet décomposé en graphes d'état. Nous effectuons un partage du cycle de traitement en sous-cycles destinés au traitement de chaque graphe.

Pour illustrer cette procédure, nous partons de l'exemple du Grafcet de la figure V-4.

Le traitement global d'un tel Grafcet sur un automate monoprocesseur, consiste à implanter l'organigramme de la figure V-5 sur un seul processeur P_1 ou P_2. Cette implantation s'effectue sans modification préalable de l'organigramme.

Ce traitement se décompose comme suit :

- Initialisation
- Prise en compte des entrées
- Calcul des conditions d'évolution
- Test des conditions d'évolution en vue de l'évolution de l'activité
- Affectation des sorties

Le calcul des conditions d'évolution est suivi de l'affectation des variables internes (V.I) correspondantes. La quatrième phase du traitement consiste à faire des tests sur les sommes logiques des conditions d'évolution (C.E).

Suivant le résultat de ce test, nous affecterons les V.I. ou nous passerons au test suivant[13].

V-4-1 - METHODE D'IMPLANTATION CHOISIE

Il existe plusieurs méthodes d'implantation du Grafcet sur les machines classiques, à savoir : la méthode des appels-réponses, la méthode basée sur les instructions

de saut ..... etc [8], [13]

La méthode que nous employons pour notre système s'applique aussi bien pour les machines monoprocesseurs que pour les systèmes multiprocesseurs; cette méthode généralise la notion du pas-à-pas. Les activités évoluent lorsque les conditions calculées le permettent. Le problème de la simultanéité va être résolu en utilisant plusieurs "pas-à-pas" avec une commande d'évolution différente.

Dans une même unité on ne peut avoir activation multiple car tout doit changer quand un élément évolue. On réalise alors une partition des étapes du Grafcet par rapport à l'activité. Chaque bloc de la partition étant représenté par un "pas-à-pas".

La condition d'évolution pour chaque bloc d'état est régie par la réunion de toutes les conditions d'évolution de cet ensemble. Cette dernière permet :

- soit d'activer l'évolution dans le bloc
- soit de faire évoluer l'activité dans la partition
  soit d'enlever l'activité du bloc.

V-4-2 - RAPPEL SUR LE CALCUL DE LA CONDITION D'EVOLUTION D'UNE TRANSITION [2]

La condition d'évolution d'une transition notée (C.E) est le produit de la transition concernée par l'état

de l'activité de l'étape d'entrée. Dans le cas d'une transition à plusieurs étapes d'entrée nous faisons intervenir le produit des marquages de toutes les étapes amont, exemple :

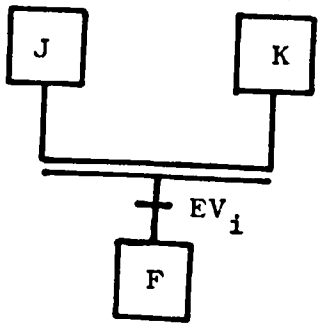


Figure V-4

La condition d'évolution de la transition i sera formulée par l'équation suivante :

$$CE_i = m_j \times m_k \times EV_i$$

soit alors :

$$CE_t = \left(\prod_{e_i \in {}^{\bullet}t} m(e_i)\right) \times EV_t$$

les e_i sont les étapes d'entrée de la transition t.

V-4-3 - TRAITEMENT D'UN GRAFCET

Grâce à l'architecture et à la technologie utilisées dans notre système multi-automates, nous disposons, pour chaque processeur, d'une instruction permettant des sauts conditionnels par masquage. Ce moyen nous permet d'utiliser aisément la méthode des pas-à-pas généralisés. Ainsi, le traitement du Grafcet de la figure V-4 peut être représenté en mettant en séquence les organigrammes de traitement relatif aux blocs (B_1 et B_2)

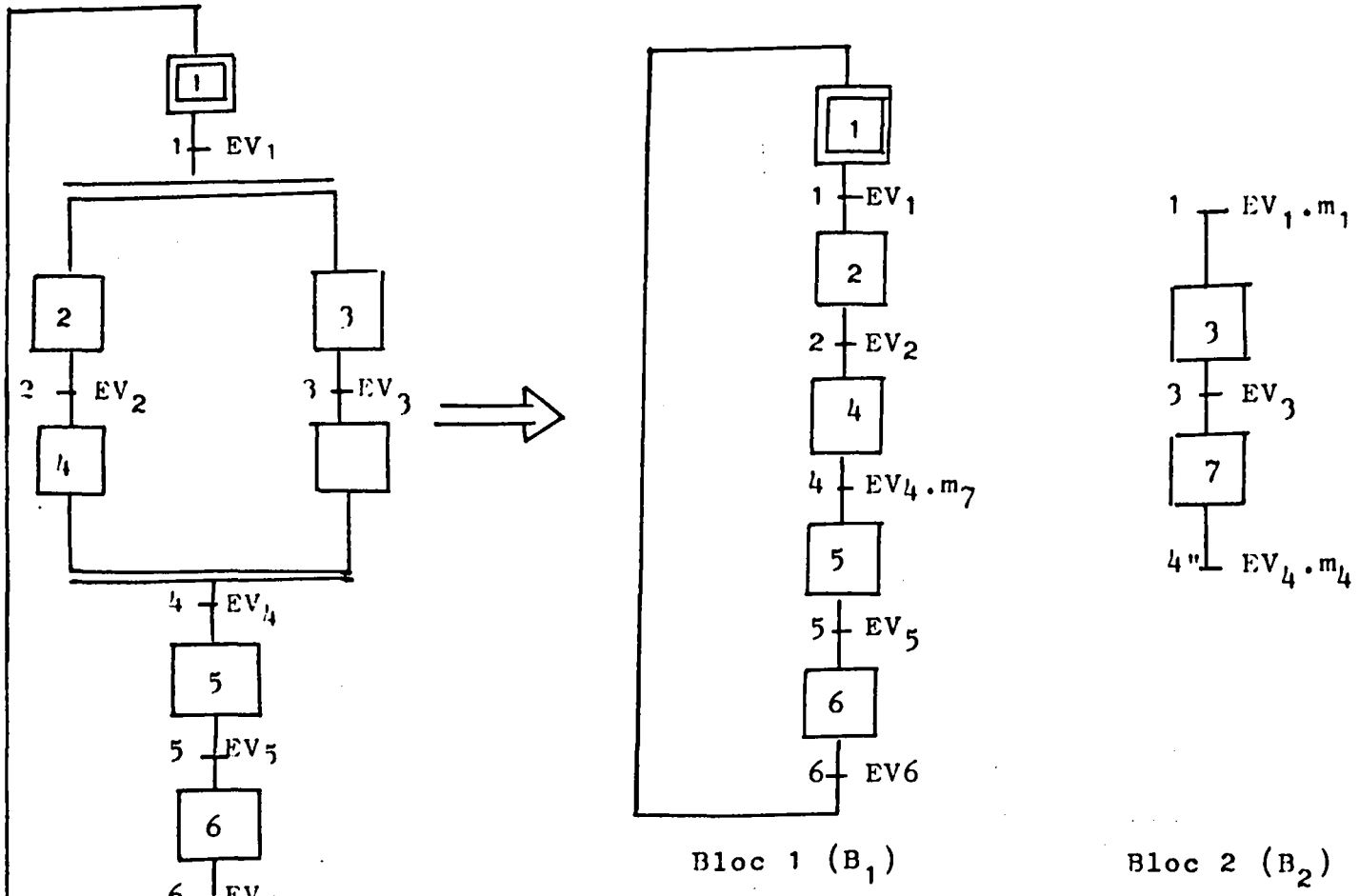


Figure V-4

L'organigramme de traitement global de ce GRAFCET est donné par la figure V-5.

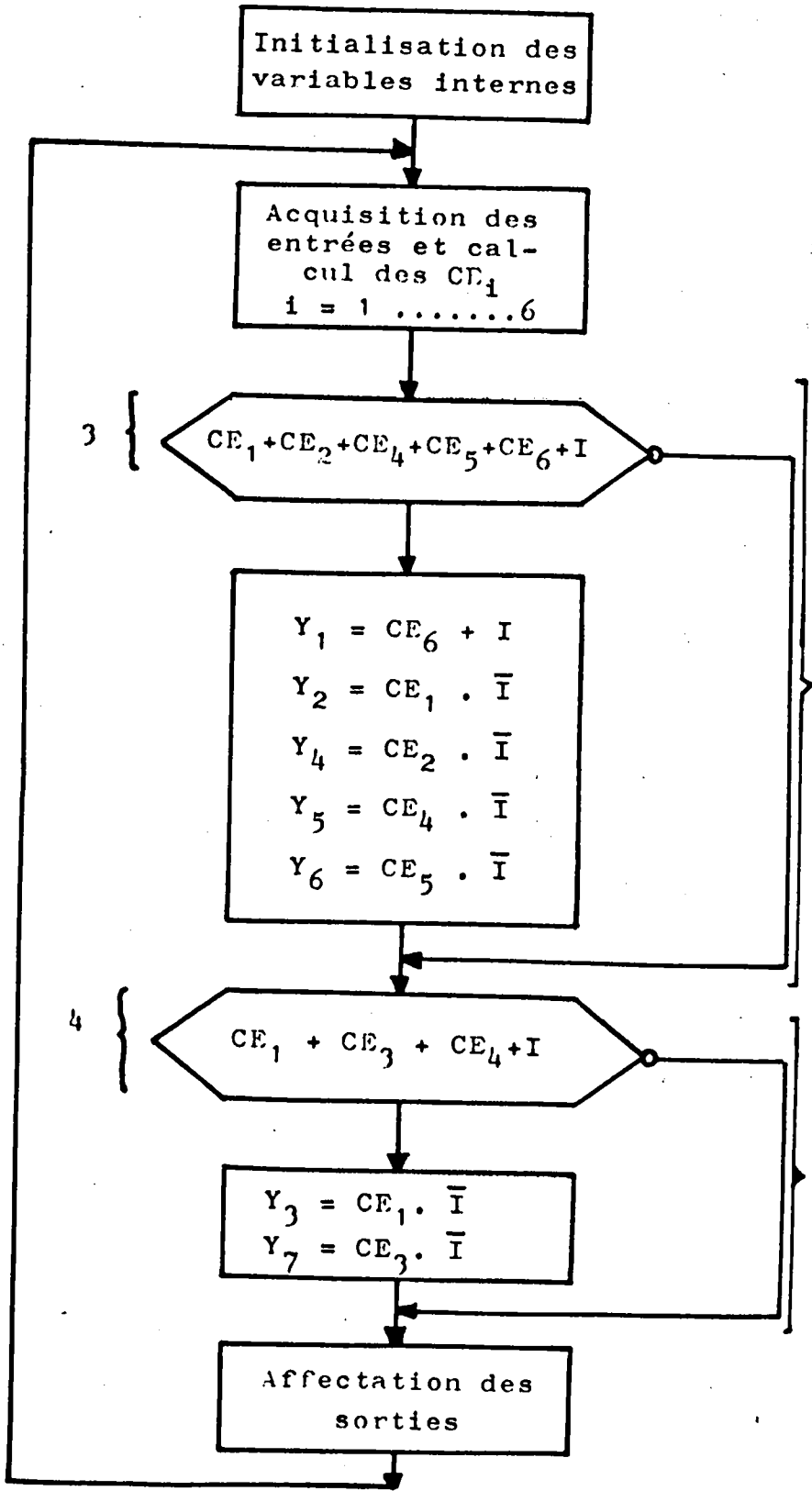


Figure V-5

V-4-3-1 - INITIALISATION

L'initialisation de l'automate est une étape impérative pour le démarrage du traitement dans les meilleures conditions.

En fait, nous avons à initialiser :

- d'une part le microprocesseur
- d'autre part, les variables internes.

V-4-3-1-1 - INITIALISATION DU MICROPROCESSEUR 14500 B [11]

Dans le cas particulier du microprocesseur 14500 B de MOTOROLA, pour valider l'entrée et la sortie sur le Bus DATA (voir chapitre II, page II.9) il faut exécuter respectivement les instructions IEN et OEN avec le registre Résultat (RR) égal à un. (voir page II.IO).

Ne disposant pas d'instruction qui nous permet de porter le RR à 1, nous utiliserons le schéma suivant :

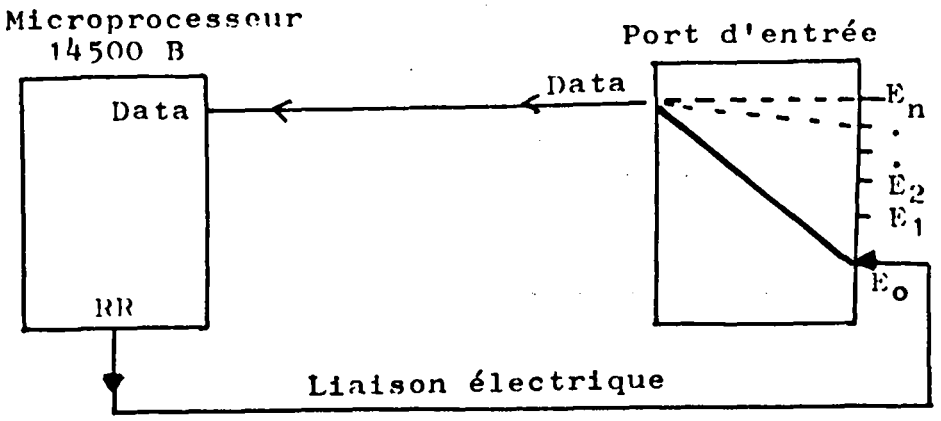


Figure V-6

...

La séquence d'instructions à exécuter est alors la suivante :

- ORC E_o effectue le OU entre RR et $\overline{DATA}$, ce qui a pour effet de mettre RR à 1
- IEN RR validation des entrées
- OEN RR validation des sorties

L'adresse de RR est celle de E_o

V-4-3-1-2 - INITIALISATION DES VARIABLES INTERNES

Chaque variable interne correspondant à une étape initiale doit être initialisée à un; les autres à zéro.

L'initialisation des V.I peut être conditionnée par l'état d'une variable externe I. Cette initialisation s'effectue suivant l'organigramme suivant :

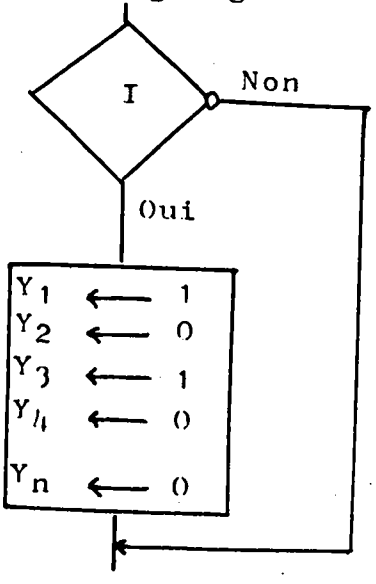


Figure V-7

Y_1 et Y_3 sont des étapes initialement actives. Quand I n'est pas vraie, on saute par masquage l'affectation des variables internes.

Le programme correspondant est alors :

```
 LD I   positionne RR suivant I
 OEN RR masquage en conséquence
 STO Y1 )
        (
STOC Y2 )
        (
 STO Y3 )
        (      Affectation des V.I
STOC Y4 )
   .    (
   .    )
   .    (
   .    )
STOC Yn (
```

V-4-3-2 - PRISE EN COMPTE DES ENTREES & AFFECTATION DES SORTIES

Ces deux étapes de traitement ont été déjà exposées dans les paragraphes V-4-2 et V-4-4.

V-4-3-3 - CALCUL DES CONDITIONS D'EVOLUTION

Le calcul des transitions d'évolution est un traitement combinatoire faisant intervenir les trois opérations logiques de base :

- L'intersection logique ($\cap$)
- L'union logique ($\cup$)
- La complémentation (-)

Une fois la première condition d'évolution (CE_1)

calculée, on l'affecte à une variable interne déterminée et on passe au calcul de la condition d'évolution qui suit, et ceci jusqu'à épuisement total de toutes les conditions d'évolution.

L'ensemble des variables internes qui sont l'image des CE_i nous permet de connaître l'activité du GRAFCET à un instant donné.

V-4-3-4 - TEST DES CONDITIONS D'EVOLUTION

Le test des conditions d'évolution constitue un traitement séquentiel qui a pour but de faire évoluer l'activité d'un GRAFCET, on y trouve, à chaque fois;

- un test sur une somme logique des CE_i
- suivant l'état de ce test, on affecte ou on saute les variables internes concernées

Cette phase de traitement est illustrée par les accolades 3 et 4 de la figure V-5.

V-5- PROCEDURE D'IMPLANTATION UTILISANT PLUSIEURS PROCESSEURS EN PARALLELE [6],[12]

V-5-1 - INTRODUCTION

Dans cette première architecture multiprocesseurs, l'échange entre les processeurs se fait au niveau d'une mémoire commune. Les conflits d'éccès à cette dernière sont

gérés par un arbitre de contrôle.

La figure V-8 nous donne le schéma synoptique de l'ensemble.

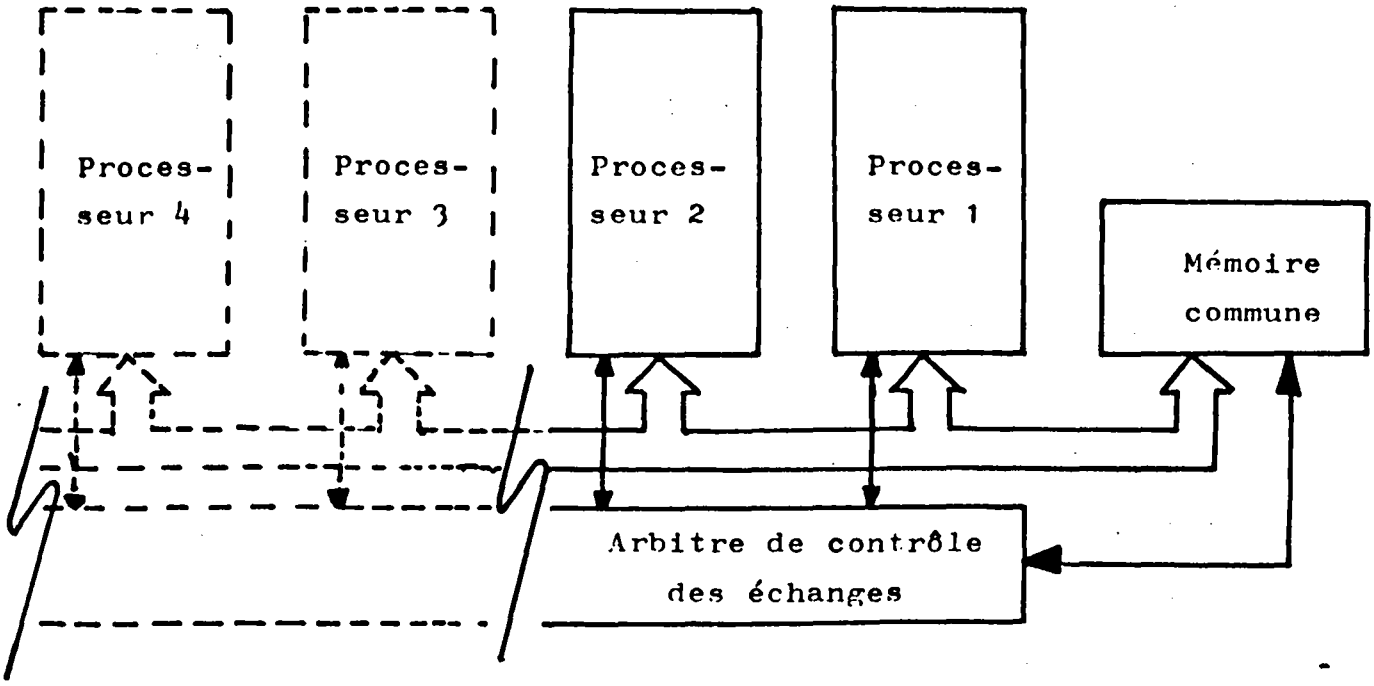


Figure V-8

Nous allons maintenant présenter une méthodologie d'implantation du GRAFCET sur un tel système que nous appellerons "multiprocesseurs banalisés".

V-5-2 - METHODOLOGIE D'IMPLANTATION

Cette implantation consiste à élaborer des sous-traitements sur chaque processeur; chacun de ces sous-traitements représente un graphe d'état issu de la décomposition du GRAFCET[3],[2]

En principe, on essaie d'associer un processeur à chaque graphe d'état. Lorsqu'on a un nombre de graphes

d'état plus grand que celui des processeurs disponibles, il faut faire un choix de façon à éviter les problèmes que nous allons évoquer.

D'un tel partage découle un certain nombre de problèmes pouvant engendrer des anomalies de fonctionnement. Ces problèmes sont dûs à l'évolution simultanée et asynchrone des différents processeurs.[4]

La méthodologie d'implantation devra prévoir des dispositifs particuliers permettant de régler les problèmes de synchronisation entre les processeurs. Ils doivent régler les problèmes :

- de synchronisation des échanges d'informations entre les processeurs
- de synchronisation de la prise en compte des E/S permettant un fonctionnement cohérent et respectant les règles d'évolution du GRAFCET
- et enfin de synchronisation entre les traitements des divers processeurs

V-5-2-1 - TRAITEMENT DU GRAFCET PAR MULTIPROCESSEURS BANALISES

Partant de l'exemple du GRAFCET de la figure V-4, la méthode des pas-à-pas généralisés nous a donné l'organigramme de traitement de la figure V-5. En considérant le produit des partitions par rapport aux graphes d'état et par rapport aux E/S, l'implantation d'un tel diagramme sur

...

un système multiprocesseurs nécessite un partage préalable des tâches en sous-organigrammes qu'on va implanter sur les différents processeurs.

L'éclatement de l'organigramme de la figure V-5 aboutit aux deux sous-organigrammes suivants :

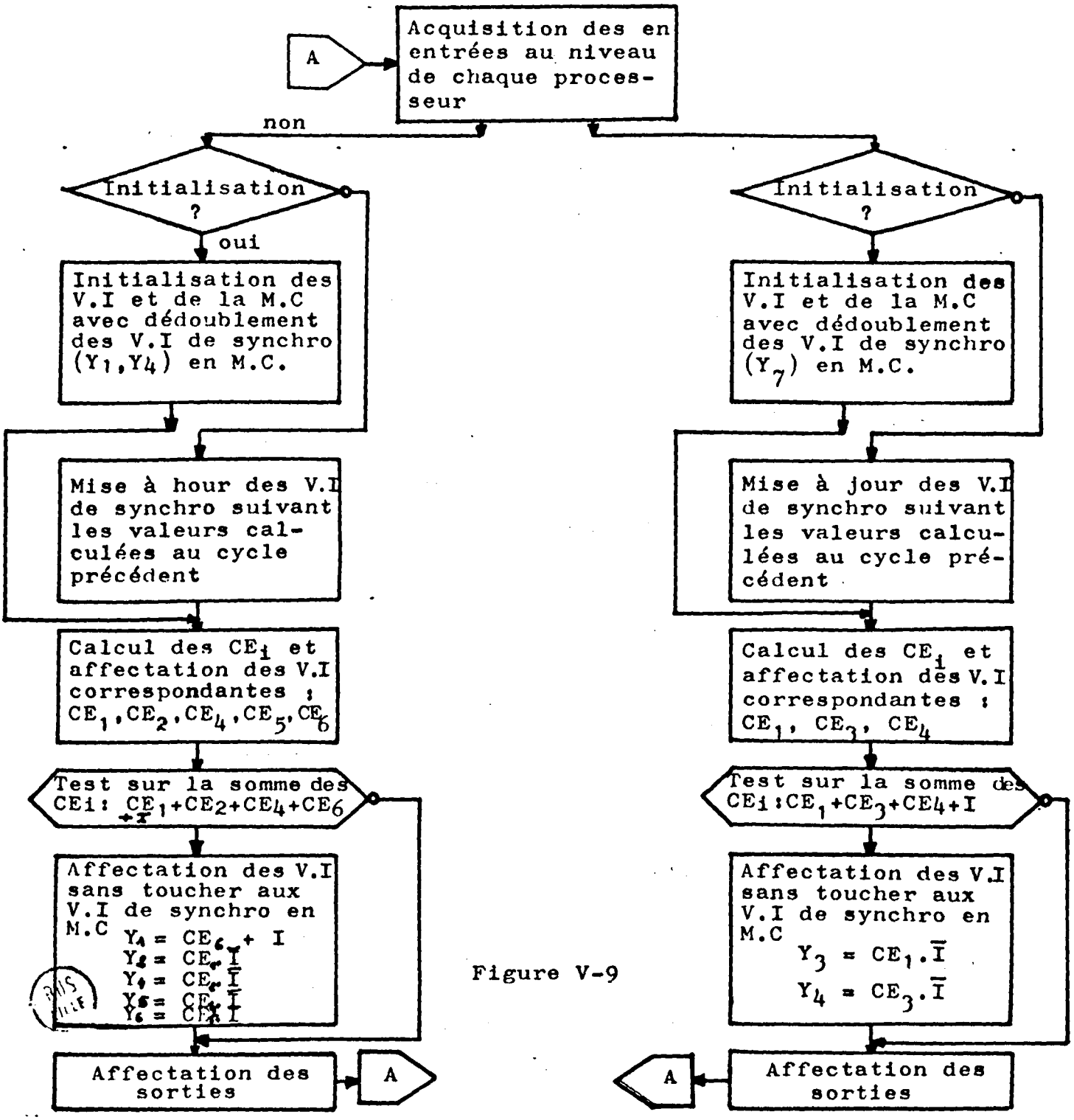


Figure V-9

L'organigramme de traitement effectué par P_1 représente l'implantation du bloc B_1 de la figure V-4.

L'organigramme de traitement effectué par P_2 représente l'implantation du bloc B2 de la figure V-4.

L'exécution des deux programmes s'effectue simultanément sur deux processeurs. On effectue alors un partage du temps de cycle de traitement relatif à l'organigramme général du GRAFCET (figure V-5) en deux temps de sous-cycles relatifs aux deux sous-organigrammes de traitement (t_1 et t_2). Le temps de cycle des multi-traitements simultanés sera égal au plus grand des deux temps t_1 ou t_2.

V-5-2-2 - SYNCHRONISATION DES TRAITEMENTS [2].[1].[16]

Les traitements dans les différents processeurs se font d'une façon asynchrone, ce qui risque de poser des problèmes d'aléas. Nous avons résolu ce problème en créant des variables de synchronisation (Y_1, Y_4 et Y_7) dans notre exemple) qu'on va mémoriser en mémoire commune. Ces variables de synchronisation vont être dédoublées au niveau des variables internes du processeur correspondant; ces dernières vont rester figées et seront remises à jour au début du cycle programme suivant. Tout ceci est décrit par l'organigramme de la figure V-9.

Dans cette structure multiprocesseurs banalisés chaque processeur effectue à la fois un traitement combi-

...

natoire pour le calcul des CE_i et l'affectation des sorties; et un traitement séquentiel relatif à l'évolution de l'activité.

Le traitement global de l'automatisme se décompose en cinq opérations effectuées simultanément sur les différents processeurs :

- Initialisation
- Prise en compte des entrées
- Calcul des conditions d'évolution
- Test des conditions d'évolution en vue de l'évolution de l'activité
- Affectation des sorties

V-5-2-3 - INITIALISATION

Elle concerne :

- L'initialisation des variables internes propres à chaque automate
- L'initialisation des microprocesseurs relatifs à chaque automate
- et enfin l'initialisation des variables de synchronisation.

Cette étape s'effectue au niveau de chaque processeur de la même façon que celle vue au paragraphe V-5-3-1.

Les variables de synchronisation sont initialisées de la même façon que les variables internes.

En dehors du problème de l'initialisation du multi-processeurs, tous les autres traitements sont identiques à ceux du monoprocesseur (voir V-5).

V-5-3 - SYNCHRONISATION DE FIN DE CYCLE DES SOUS-TRAITEMENTS

Les traitements dans les différents automates sont effectués d'une façon autonome, les programmes sont scrutés séquentiellement en parallèle. A la remise à zéro, le compteur ordinal (C.O.) dans chaque automate se met à l'adresse zéro.

La durée du cycle programme varie d'un processeur à l'autre suivant le nombre d'instructions introduites. Pour synchroniser l'ensemble des processeurs, nous attendons qu'il aient tous effectué leur cycle programme avant de remettre le C.O. de chacun à zéro.

Cette synchronisation est effectuée d'une façon matérielle à l'aide de bascules J.K comme nous le montre la figure suivante [9].

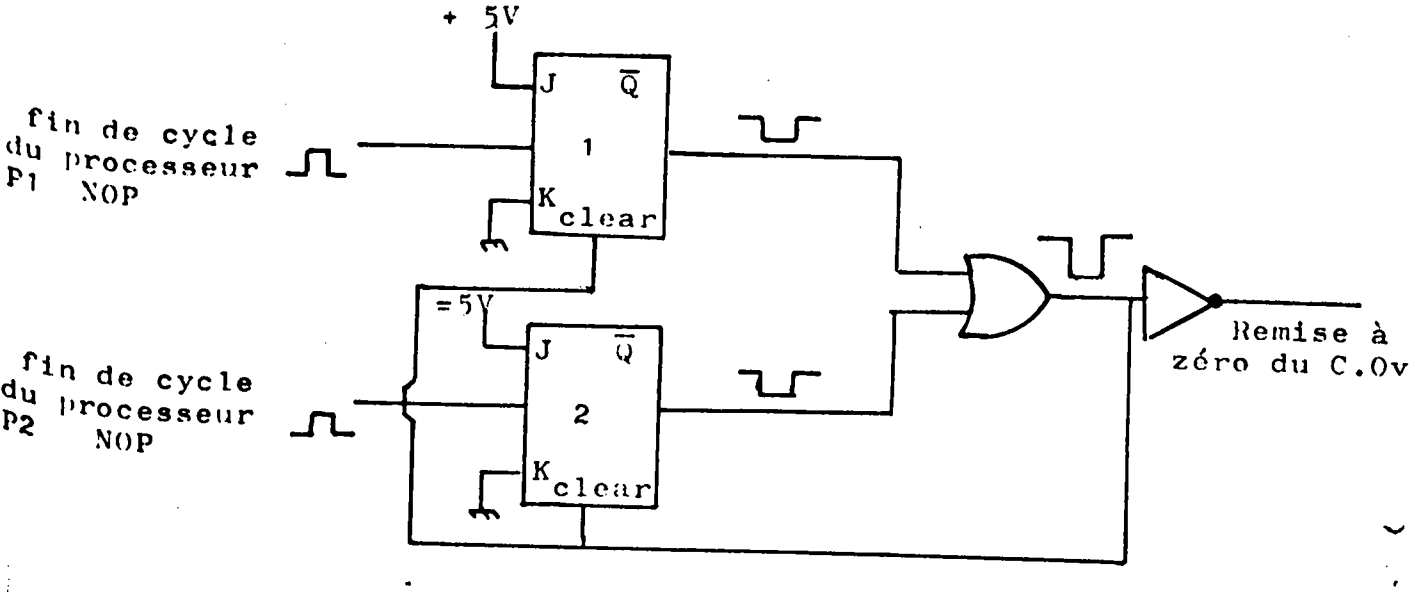


Figure V-10

La durée de cycle programme du système multi-automates est celle du processeur qui possède le plus grand temps de cycle programme.

V-5-4 - TRAITEMENT PAR RAPPORT AUX ENTREES-SORTIES DU SYSTEME MULTIPROCESSEURS

Pour la prise en compte des entrées, ces dernières sont, soit appliquées simultanément à tous les processeurs, soit, après décomposition relative aux E/S, appliquées par paquets sur différents processeurs.

Dans les deux cas, la prise en compte de ces E/S s'effectue comme l'indique les paragraphes V-4-2 et V-4-4

V-6 - PROCEDURE D'IMPLANTATION UTILISANT UN SYSTEME MULTIPROCESSEURS SPECIALISES

V-6-1 - INTRODUCTION

Cette procédure d'implantation s'effectue sur un système multi-automates programmables, où chaque processeur a un traitement spécial à effectuer. Nous avons fait l'étude sur un système de trois processeurs spécialisés mais cette étude peut être étendue à plusieurs.

Les traitements à effectuer par les processeurs sont, soit d'ordre combinatoire, soit d'ordre séquentiel.

Nous avons ainsi choisi la configuration suivante :

- Deux automates programmables effectuant des traitements combinatoires
- Un automate programmable séquentiel qui effectue uniquement un traitement séquentiel
- Une mémoire commune à double accès utilisée par tous les automates, cependant l'automate séquentiel reste privilégié par rapport aux autres; en-effet, il peut y accéder sans formuler de demande préalable, il ne converse qu'avec elle.

Les problèmes posés par un tel système sont les mêmes que ceux examinés au paragraphe V-6-2 et qui sont posés par le système multiprocesseurs banalisés.

Par contre, la méthodologie d'implantation varie suivant la spécialisation des automates programmables.

V-6-2 - METHODOLOGIE D'IMPLANTATION DU GRAFCET SUR UN SYSTEME MULTIPROCESSEURS SPECIALISES

Cette méthode est fondée sur une décomposition préalable du GRAFCET. Le traitement global d'un graphe d'état est constitué d'un traitement combinatoire et d'un traitement séquentiel. Nous implantons les traitements séquentiels correspondant à tous les graphes d'état dans l'automate programmable séquentiel. D'autre-part, le traitement combinatoire de chaque graphe va être implanté dans un automate programmable combinatoire.

Ces différents automates échangent leurs informations respectives par l'intermédiaire de la mémoire commune.

...

Nous illustrons cette implantation en traitant le GRAFCET de la figure V-4.

L'application de la méthode des "pas-à-pas" généralisés et du produit des partitions par rapport aux graphes d'état et aux E/S, nous a conduit à l'organigramme de la figure V-5.

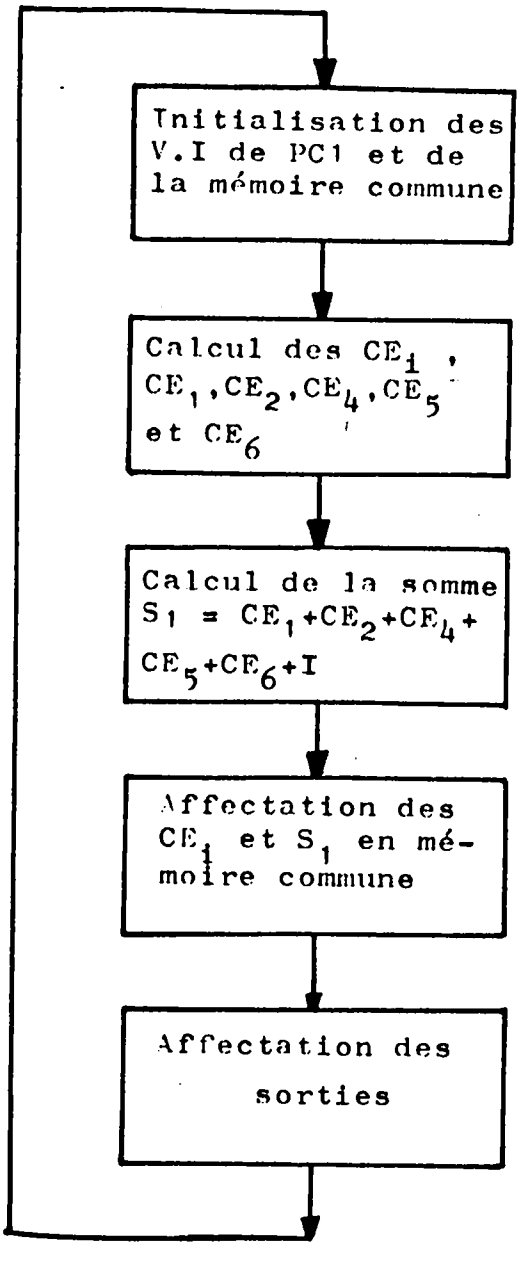
En effectuant une séparation préalable des deux traitements combinatoire et séquentiel on aboutit aux deux organigrammes suivants.

V-6-2-1 - ORGANIGRAMME DE TRAITEMENT RELATIF AUX PROCESSEURS COMBINATOIRES

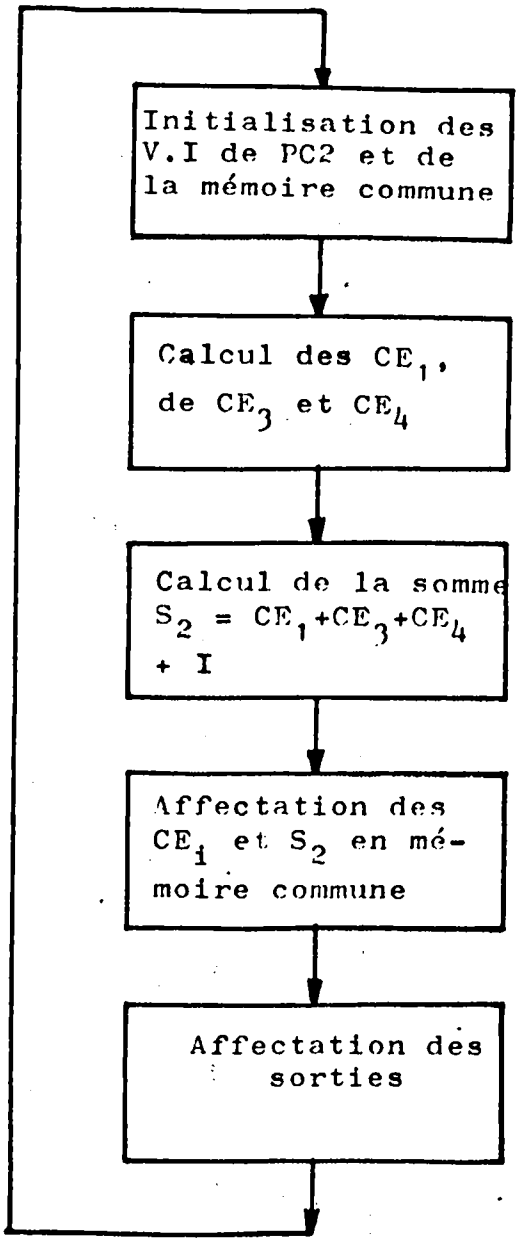
Les traitements effectués par les processeurs PC1 et PC2 vont être d'ordre combinatoire et consistent en le calcul des conditions d'évolution relatives aux graphes d'état de la figure V-4. Les organigrammes de traitement correspondant sont donnés par la figure V-11.

Le lien entre ces deux traitements réside dans les variables de synchronisation dont nous avons, à tout instant, une image en mémoire commune.

Ce sont des traitements purement combinatoires qui ne nous renseignent pas sur l'évolution de l'activité dans le GRAFCET. C'est au niveau de l'automate programmable séquentiel que nous pouvons suivre l'évolution de l'activité.



Traitement effectué par PC1



Traitement effectué par PC2

Figure V-11

V-6-2-2 - ORGANIGRAMME DE TRAITEMENT EFFECTUE PAR L'AUTOMATE PROGRAMMABLE SEQUENTIEL (P.S) [13]

L'automate programmable séquentiel ne possède ni entrée ni sortie, il ne dialogue qu'avec la mémoire commune. Son traitement consiste donc à consulter la mémoire commune en vue de faire évoluer l'activité.

Ce traitement est représenté par l'organigramme de la figure V-12.

...

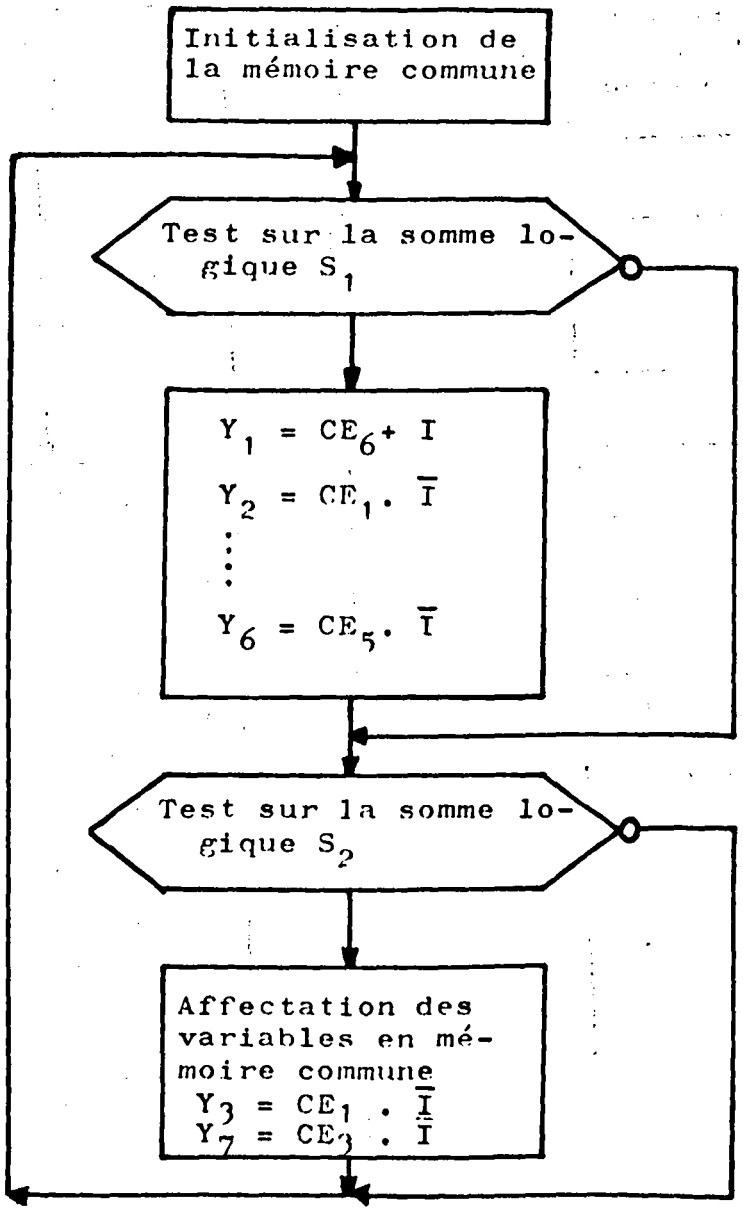


Figure V-12

Le traitement séquentiel est basé sur l'instruction de saut par masquage, quand le test est vrai on réalise les affectations nécessaires, quand le test est faux, on saute par masquage toutes ces affectations.

V-6-3 - TRAITEMENT GLOBAL

Le traitement global du multiprocesseurs spécialisés consiste à faire évoluer l'ensemble des programmes introduits respectivement sur PC1, PC2 et PS, d'une façon simultanée.

L'organigramme du traitement global est donné par la figure V-13.

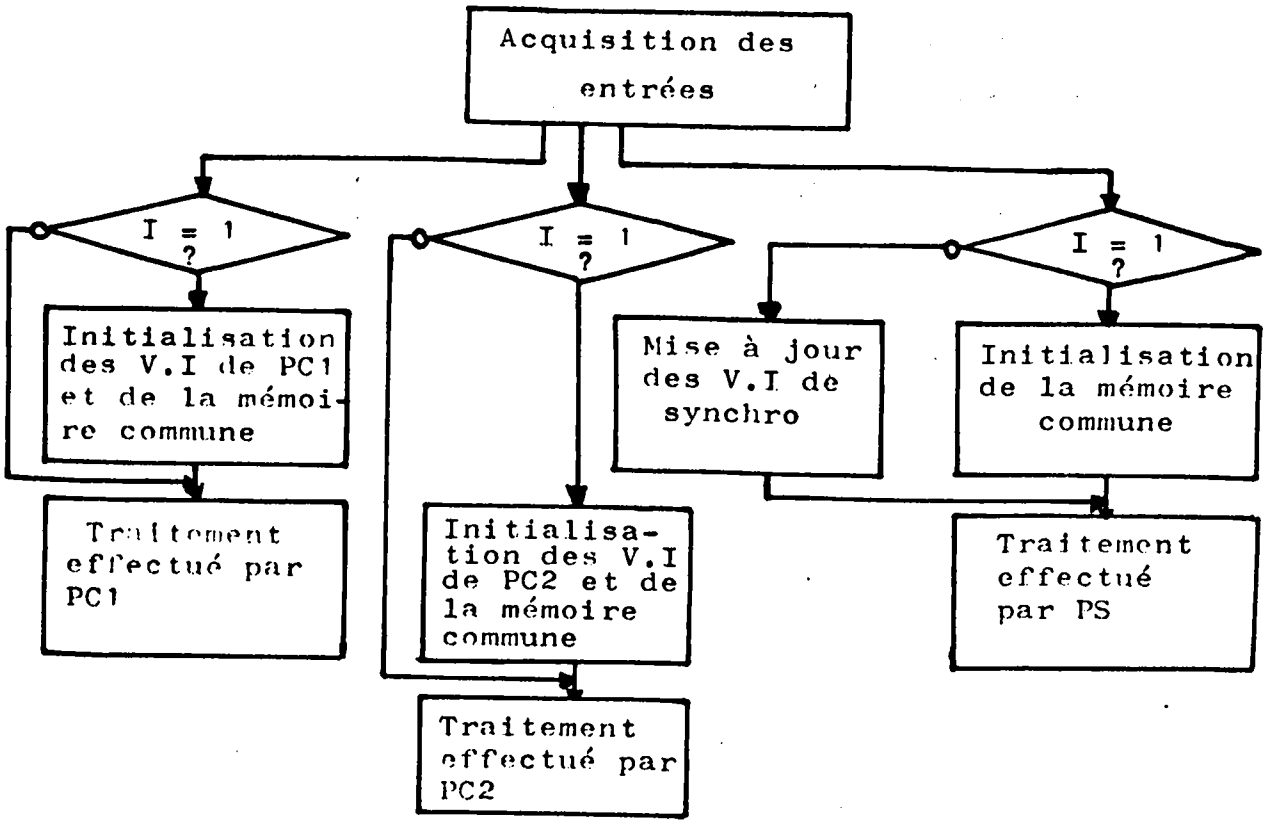


Figure V-13

Les programmes dans les automates sont scrutés d'une façon autonome. Au démarrage du processus, tous les COr des différents processeurs, sont à l'adresse zéro. La durée du cycle programme varie d'un automate à l'autre.

Pour des raisons de synchronisation, nous attendons que tous les processeurs aient terminé leur cycle programme avant de remettre à zéro l'ensemble des COr

Cette synchronisation est effectuée à l'aide de bascules J.K comme le montre la figure V-14.

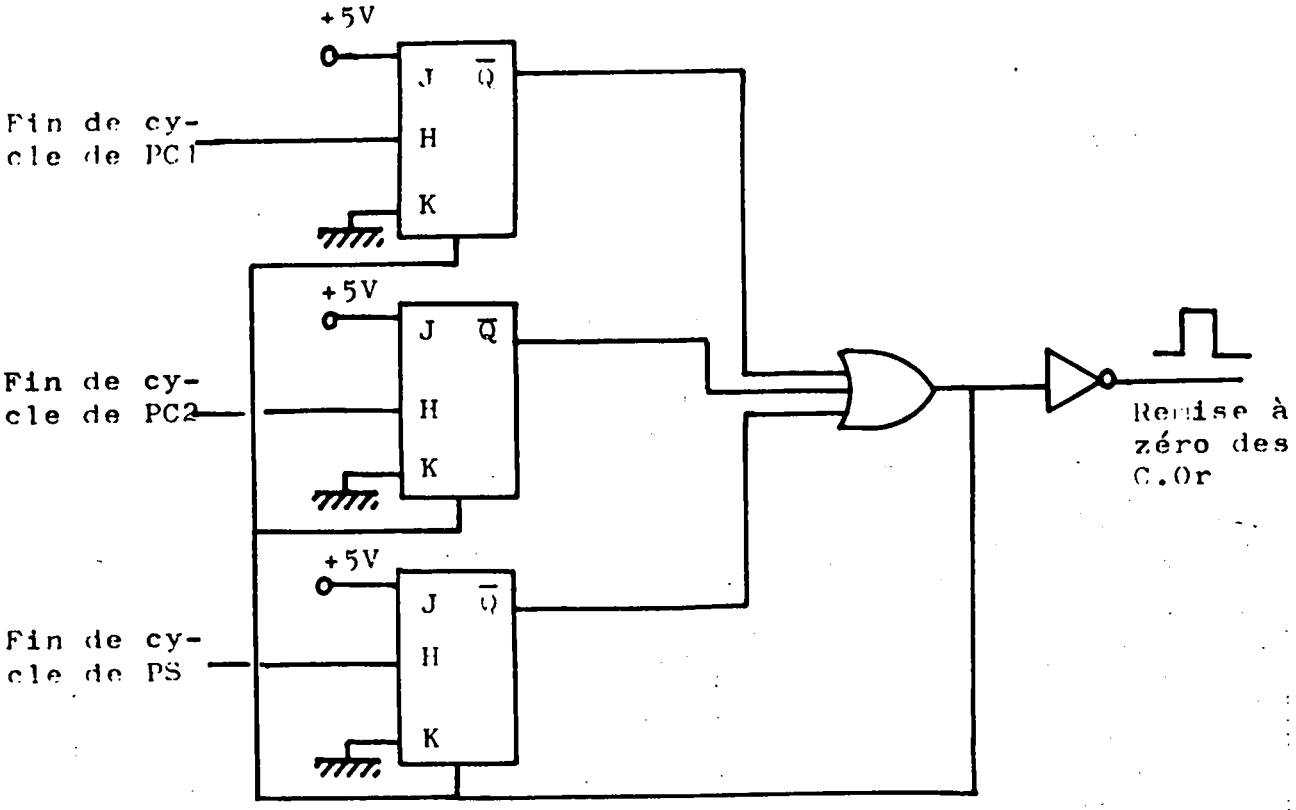


Figure V-14

V-7 - CONCLUSION

Dans ce chapitre, nous avons présenté une méthodologie d'implantation d'un automatisme logique sur un système programmable en se basant sur l'outil de représentation (GRAFCET). Nous avons introduit trois procédures :

- une procédure utilisant un seul processeur
- une procédure utilisant deux ou plusieurs processeurs banalisés
- et une procédure utilisant trois ou plusieurs processeur spécialisés (deux automates programmables combinatoires et un automate séquentiel).

Nous avons étudié les problèmes relatifs au synchronisme introduits par les différentes procédures et par la description GRAFCET.

Pour permettre une implantation multiprocesseurs

et un traitement simultané, nous décomposons, au préalable, le GRAFCET soit par rapport à ses entrées-sorties, soit en graphes d'état.

L'implantation de ces composantes sur différents processeurs travaillant en parallèle, permet un traitement rapide mais suppose le contrôle des échanges d'informations.

o
o o

CONCLUSION GENERALE

Dans cette étude, nous avons réalisé une architecture multiprocesseurs adaptée au traitement des automatismes logiques, en effectuant un partage des tâches sur plusieurs processeurs. Nous avons montré que l'implantation du GRAFCET sur un tel système nécessite une décomposition préalable de ce dernier et que les tâches effectuées par chaque processeur varient suivant la procédure utilisée.

Les performances d'un tel système réside dans la rapidité de traitement, en-effet, les exemples que nous avons traités montrent que le temps de cycle programmé diminue dans de bonne proportions quand on partage les tâches sur différents automates.

La comparaison des implantations suivant les différentes procédures, nous a conduit à classer ces dernières par ordre croissant. La meilleure procédure qui permet un temps de cycle minimum est celle utilisant des processeurs banalisés; la procédure avec des processeurs spécialisés risque, dans certains cas, d'augmenter le cycle programme du processeur séquentiel.

Par ailleurs, l'emploi des structures multiprocesseurs est à l'ordre du jour et l'architecture proposée est telle qu'elle peut s'étendre à (n) processeurs; de ce fait, c'est un moyen matériel pour approfondir la multi-programmation.

Suivant les problèmes industriels posés, ce moyen permettrait de trouver le nombre optimal de processeurs à utiliser.

D'autre-part, il serait utile et nécessaire de compléter cette étude en introduisant un processeur numérique en parallèle avec les autres processeurs logiques pour le traitement des temporisations, du comptage ... etc.

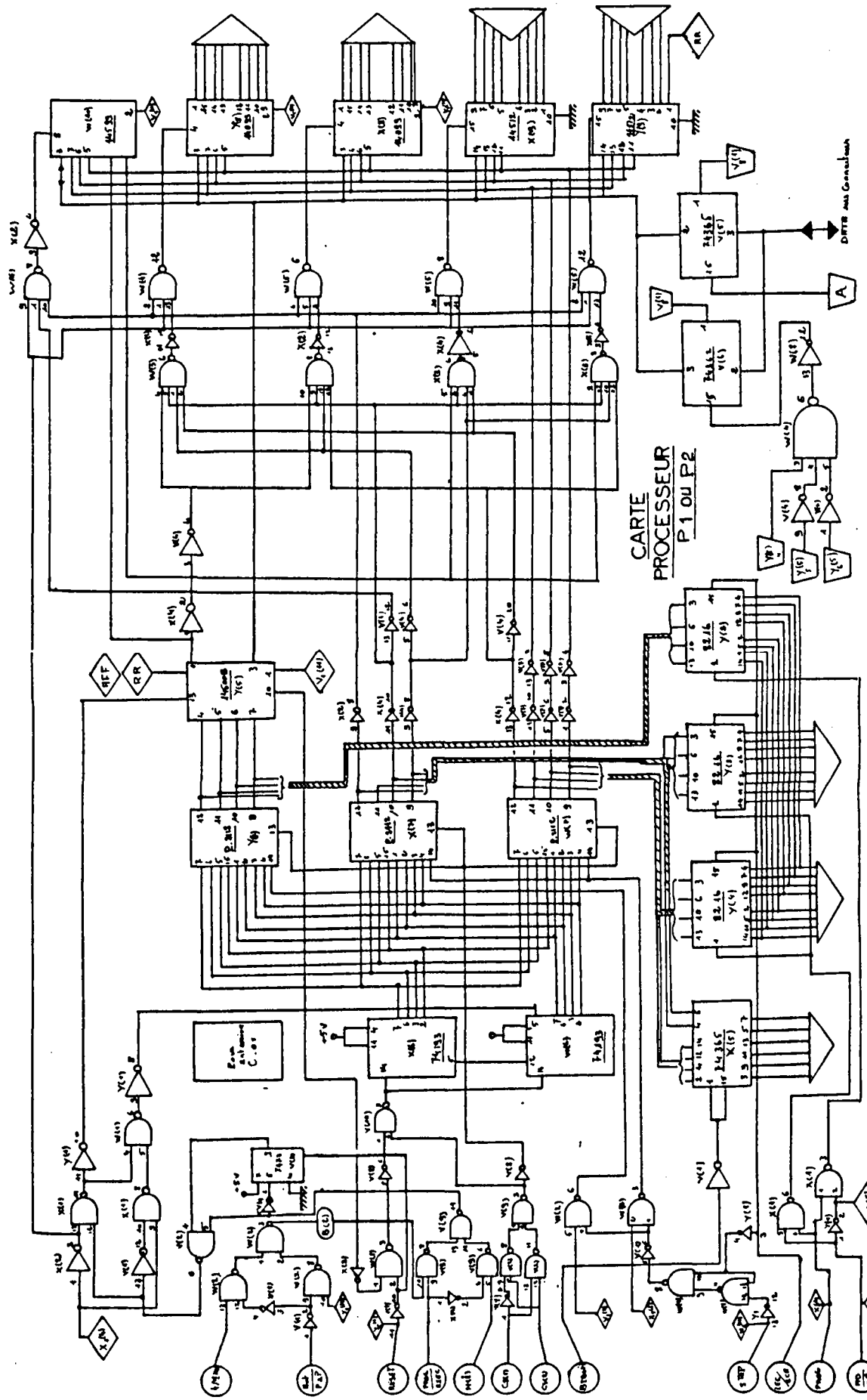
o
o o

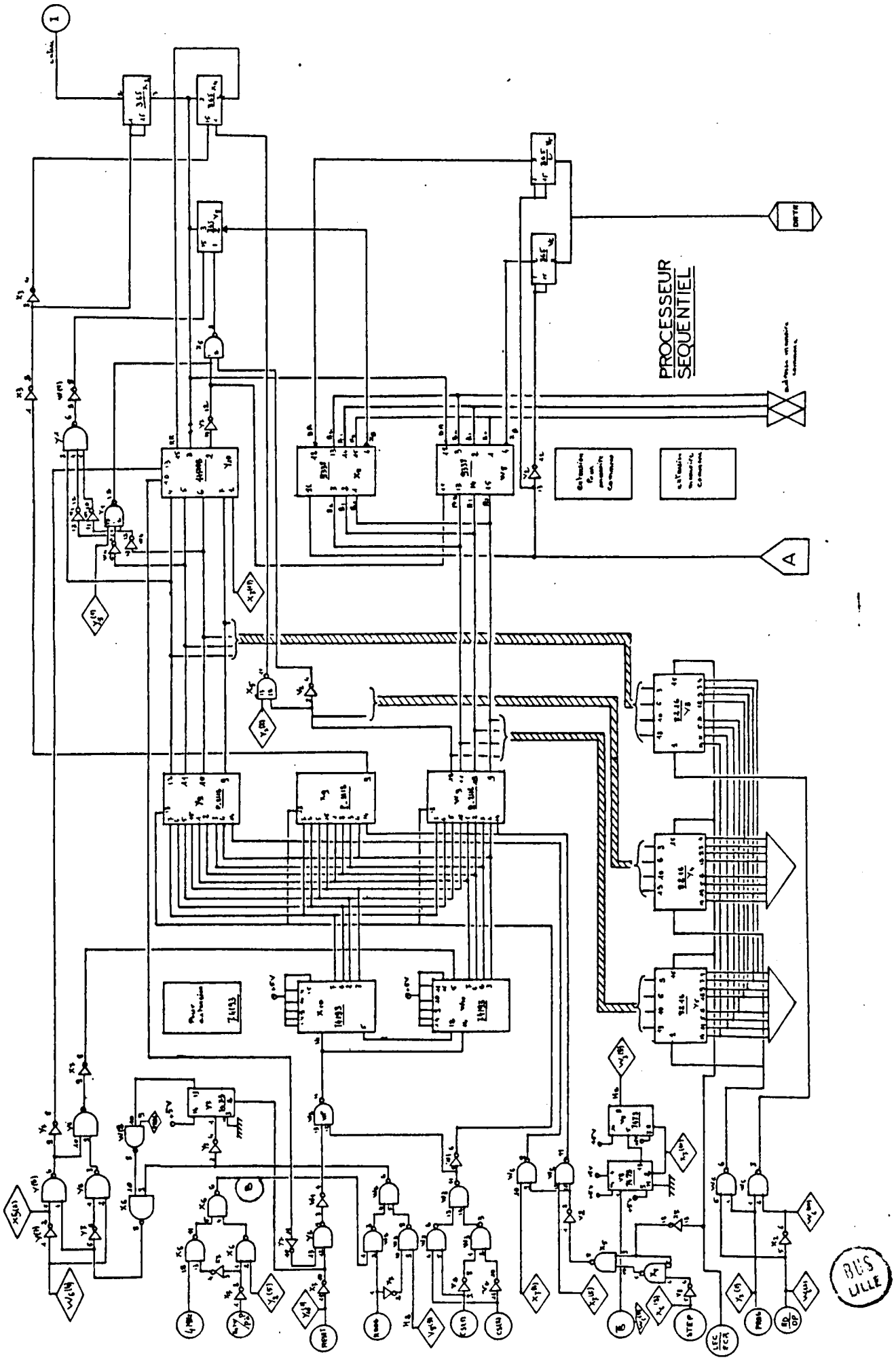
A N N E X E

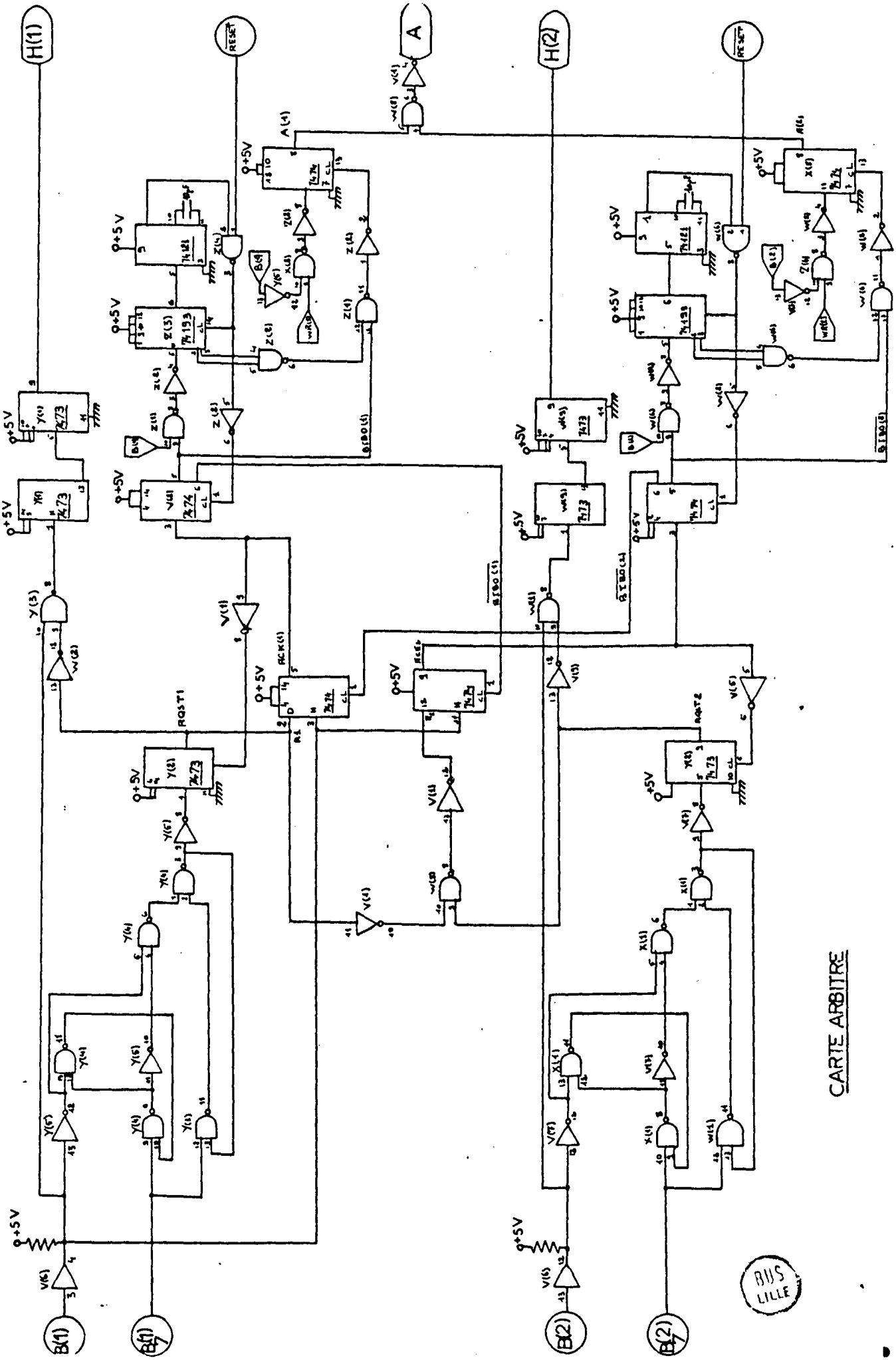
Dans cette annexe, nous donnons le tableau des différents temps d'établissement des signaux du 14500B, les schémas techniques des différentes cartes réalisées et le programme moniteur de pilotage du multi-automates programmables à partir du CBM 3032.

Tableau des différents temps d'établissement des signaux.

Caractéristiques	Symbole	VDD Vdc	Typ	Unité
Propagation Delay Time X1 to RR	tdR	10	110	ns
X1 to FLAGF, FLAGO, RTN, JMP	tdF	10	100	ns
X1 to WRITE	tdW	10	125	ns
X1 to DATA	tdD	10	120	ns
RST to RR	tdRRR	10	110	ns
RST to X1	tdRX	10	120	ns
RST to FLAGF, FLAGO, RTN, JMP	tdRF	10	90	ns
RST to WRITE, DATA	tdRW	10	90	ns
Minimum Clock Pulse Width,X1	PWC	10	50	ns
Setup Time Instruction	tIS	10	125	ns
DATA	tDS	10	50	ns
Hold Time Instruction	tIH	10	0	ns
DATA	tDH	10	30	ns




PROCESSEUR
SEQUENTIEL
A
BUS
LILLE



CARTE ARBITRE

Nous donnons, ci-dessous, le programme de gestion et de pilotage des différents automates.

```
10 PRINT"CHOIX DU PROCESSEUR A PILOTER"
20 INPUT P$
30 IF P$="PC1" THEN 60
40 IF P$="PC2" THEN 740
50 IF P$="PS" THEN 1520
60 IF P$="T" THEN 2350
70 GOTO 10
80 PRINT"CHOIX DU MODE ECRITURE,LECTURE OU EXECUTION DU PROG EN M(PC1)"
90 INPUT CHOIX$
100 IF CHOIX$="ECRITURE" THEN 140
110 IF CHOIX$="LECTURE" THEN 350
120 IF CHOIX$="EXECUTION"THEN 550
130 GOTO 10
140 PRINT"INITI ET PROG DU PORT IEEE ET DUPORT A EN SORTIE"
150 GOSUB 3000
160 GOSUB 3060
170 PRINT"CHOIX DU CODE A IMPLANTER EN M(PC1):CI OU CO"
180 INPUT C$
190 IF C$="CI" THEN 230
200 IF C$="CO" THEN 290
210 IF C$="-1" THEN 10
220 GOTO 170
230 PRINT"IMPLANTATION EN M(PC1) DU CODE CI"
240 INPUT X
250 IF X<0 THEN 5980
260 POKE 59426,X
270 GOSUB 3090
280 GOTO 240
290 PRINT"IMPLANTATION EN M(PC1) DU CODE CO"
300 INPUT X
310 IF X<0 THEN 160
320 POKE 59426,X
330 GOSUB 3130
340 GOTO 300
350 PRINT"INITI ET PROG DU PORT IEEE EN ENTREE ET DU PORT A EN SORTIE"
360 GOSUB 3170
370 PRINT"CHOIX DU CODE A LIRE EN M(PC1):CI OU CO"
380 INPUT C$
390 IF C$="CI" THEN 430
400 IF C$="CO" THEN 430
410 IF C$="-1" THEN 80
420 GOTO 370
430 PRINT"LECTURE DU CODE INSTRUCTION EN M(PC1)"
440 POKE 59471,92
445 A=PEEK(59424)
447 PRINT A
450 INPUT X
460 IF X<0 THEN 350
470 GOSUB 3230
480 GOTO 450
490 PRINT"LECTURE DU CODE OPERATION EN M(PC1)"
500 POKE 59471,84
503 A=PEEK(59424)
507 PRINT A
510 INPUT X
```

```
520 IF X<0 THEN 350
530 GOSUB 3290
540 GOTO 510
550 PRINT"INIT DU PORT A EN SORTIE"
560 GOSUB 3050
570 PRINT"CHOIX DU MODE D'EXECUTION DU PROGRAMME EN M(PC1)"
580 INPUT ME$
590 IF ME$="PAP" THEN 620
600 IF ME$="AUT" THEN 680
610 IF ME$="-1" THEN 80
620 PRINT"EXECUTION EN PAP DU PROG EN M(PC1)"
630 POKE 59471,68
633 POKE 59471,100
634 POKE 59471,100
635 POKE 59471,68
637 PRINT"ADR MAX AEXECUTER"
640 INPUT J
650 IF J<0 THEN GOTO 550
655 FOR K=0 TO J
660 GOSUB 3380
670 NEXT K
675 GOTO  640
680 PRINT"EXECUTION EN AUT DU PROG EN M(PC1)"
690 POKE 59471,254
695 POKE 59471,102
700 POKE 59471,70
710 INPUT A
720 IF A<0 THEN 550
730 GOTO 680
740 PRINT"CHOIX DU MODE ECRITURE,LECTURE OU EXECUTION DU PROG EN M(PC2)"
750 INPUT CHOIX$
760 IF CHOIX$="ECRITURE" THEN 800
770 IF CHOIX$="LECTURE" THEN 1090
780 IF CHOIX$="EXECUTION"THEN 1350
790 GOTO 10
800 PRINT"INITI ET PROG DU PORT IEEE ET DUPORT A EN SORTIE"
810 GOSUB 3000
820 GOSUB 3060
830 PRINT"CHOIX DU CODE A IMPLANTER EN M(PC2):CI OU CO"
840 INPUT C$
850 IF C$="CI" THEN 890
860 IF C$="CO" THEN 1030
870 IF C$="-1" THEN 10
880 GOTO 830
890 PRINT"IMPLANTATION EN M(PC2) DU CODE CI"
900 INPUT X
910 IF X<0 THEN 4020
1000 POKE 59426,X
1010 GOSUB 3440
1020 GOTO 900
1030 PRINT"IMPLANTATION EN M(PC2) DU CO"
1040 INPUT X
1050 IF X<0 THEN 4000
1060 POKE 59426,X
1070 GOSUB 3480
1080 GOTO 1040
1090 PRINT"INITI ET PROG DU PORT IEEE EN ENTREE ET DU PORT A EN SORTIE"
1100 GOSUB 3170
1110 PRINT"CHOIX DU CODE A LIRE EN M(PC2):CI OU CO"
1120 INPUT C$
1130 IF C$="CI" THEN 1170
1140 IF C$="CO" THEN 1250
1150 IF C$="-1" THEN 740
1160 GOTO 1110
```

```
1170 PRINT"LECTURE DU CODE INSTRUCTION EN M(PC2)"
1180 POKE 59471,156
1190 A=PEEK(59424)
1200 PRINT A
1210 INPUT X
1220 IF X<0 THEN 1090
1230 GOSUB 3520
1240 GOTO 1210
1250 PRINT"LECTURE DU CODE OPERATION EN M(PC2)"
1260 POKE 59471,148
1270 A=PEEK(59424)
1280 PRINT A
1290 INPUT X
1300 IF X<0 THEN 1090
1310 GOSUB 3580
1320 GOTO 1290
1330 PRINT"INIT DU PORT A EN SORTIE"
1340 GOSUB 3350
1350 PRINT"CHOIX DU MODE D'EXECUTION DU PROGRAMME EN M(PC2)"
1360 INPUT ME$
1370 IF ME$="PAP" THEN 1400
1380 IF ME$="AUT" THEN 1460
1390 IF ME$="-1" THEN 80
1400 PRINT"EXECUTION EN PAP DU PROG EN M(PC2)"
1410 POKE 59471,132
1420 INPUT A
1430 IF A<0 THEN GOTO 1330
1440 GOSUB 3640
1450 GOTO 1420
1460 PRINT"EXECUTION EN AUT DU PROG EN M(PC2)"
1470 POKE 59471,255
1480 POKE 59471,134
1490 INPUT A
1500 IF A<0 THEN 80
1510 GOTO 1460
1520 PRINT"CHOIX DU MODE ECRITURE,LECTURE OU EXECUTION DU PROG EN M(PS)"
1530 INPUT CHOIX$
1540 IF CHOIX$="ECRITURE" THEN 1580
1550 IF CHOIX$="LECTURE" THEN 1830
1560 IF CHOIX$="EXECUTION"THEN 2070
1570 GOTO 10
1580 PRINT"INITI ET PROG DU PORT IEEE ET DUPORT A EN SORTIE"
1590 GOSUB 3000
1600 GOSUB 3060
1610 PRINT"CHOIX DU CODE A IMPLANTER EN M(PS):CI OU CO"
1620 INPUT C$
1630 IF C$="CI" THEN 1710
1640 IF C$="CO" THEN 1770
1650 IF C$="-1" THEN 1520
1700 GOTO 1610
1710 PRINT"IMPLANTATION EN M(PS) DU CODE CI"
1720 INPUT X
1730 IF X<0 THEN 4040
1740 POKE 59426,X
1750 GOSUB 3700
1760 GOTO 1720
1770 PRINT"IMPLANTATION EN M(PS) DU CODE CO"
1780 INPUT X
1790 IF X<0 THEN 4060
1800 POKE 59426,X
1810 GOSUB 3740
1820 GOTO 1780
1830 PRINT"INITI ET PROG DU PORT IEEE EN ENTREE ET DU PORT A EN SORTIE"
1840 GOSUB 3170
```

```
1850 PRINT"CHOIX DU CODE A LIRE EN M(PS):CI OU CO"
1860 INPUT C$
1870 IF C$="CI" THEN 1910
1880 IF C$="CO" THEN 1990
1890 IF C$="-1" THEN 1520
1900 GOTO 1850
1910 PRINT"LECTURE DU CODE INSTRUCTION EN M(PS)"
1920 POKE 59471,28
1930 A=PEEK(59424)
1940 PRINT A
1950 INPUT X
1960 IF X<0 THEN 1830
1970 GOSUB 3780
1980 GOTO 1950
1990 PRINT"LECTURE DU CODE OPERATION EN M(PS)"
2000 POKE 59471,20
2010 A=PEEK(59424)
2020 PRINT A
2030 INPUT X
2040 IF X<0 THEN 1830
2050 GOSUB 3820
2060 GOTO 2030
2070 PRINT"INIT DU PORT A EN SORTIE"
2080 GOSUB 3350
2090 PRINT"CHOIX DU MODE D'EXECUTION DU PROGRAMME EN M(PS)"
2100 INPUT ME$
2200 IF ME$="PAP" THEN 2230
2210 IF ME$="AUT" THEN 2290
2220 IF ME$="-1" THEN 80
2230 PRINT"EXECUTION EN PAP DU PROG EN M(PS)"
2240 POKE 59471,4
2250 INPUT A
2260 IF A<0 THEN GOTO 2070
2270 GOSUB 3860
2280 GOTO 2250
2290 PRINT"EXECUTION EN AUT DU PROG EN M(PS)"
2300 POKE 59471,255
2310 POKE 59471,6
2320 INPUT A
2330 IF A<0 THEN 80
2340 GOTO 2290
2350 PRINT"CHOIX DU MODE EXECUTION DU CONTENU M DE L'ENSEMBLE DES AUTOMATES"
2355 GOSUB  3350
2360 INPUT ME$
2370 IF ME$="PAP" THEN 2400
2380 IF ME$="AUT" THEN 2460
2390 IF ME$="-1" THEN 80
2400 PRINT"EXECUTION EN PAP DU PROG EN M(PC1),M(PC2)ET EN M(PS)"
2410 POKE 59471,196
2420 INPUT A
2430 IF A<0 THEN GOTO 2350
2440 GOSUB 3920
2450 GOTO 2420
2460 PRINT"EXECUTION EN AUT DU PROG EN M(PC1),M(PC2) ET EN M(PS)"
2470 POKE 59471,255
2480 POKE 59471,198
2490 INPUT A
2500 IF A<0 THEN 80
2510 GOTO 2490
3000 POKE 59427,000
3010 POKE 59426,255
3020 POKE 59427,004
3030 POKE 59426,000
3040 POKE 59459,255
3050 RETURN
```

```
3060 PRINT"REMISE A ZERO GENERALE"
3070 POKE 59471,254
3080 RETURN
3090 POKE 59471,68
3100 POKE 59471,69
3110 POKE 59471,68
3120 RETURN
3130 POKE 59471,80
3140 POKE 59471,81
3150 POKE 59471,80
3160 RETURN
3170 POKE 59459,255
3180 POKE 59471,255
3190 POKE 59427,000
3200 POKE 59426,000
3210 POKE 59427,004
3220 RETURN
3230 POKE 59471,92
3240 POKE 59471,93
3250 POKE 59471,92
3260 A=PEEK(59424)
3270 PRINT A
3280 RETURN
3290 POKE 59471,84
3300 POKE 59471,85
3310 POKE 59471,84
3320 A=PEEK(59424)
3330 PRINT A
3340 RETURN
3350 POKE 59459,255
3360 POKE 59471,254
3370 RETURN
3380 FOR I=1 TO 4 STEP 1
3390 POKE 59471,68
3400 POKE 59471,69
3410 NEXT I
3420 RETURN
3430 POKE 59471,152
3440 POKE 59471,153
3450 POKE 59471,152
3460 RETURN
3470 POKE 59471,144
3480 POKE 59471,145
3490 POKE 59471,144
3500 RETURN
3510 POKE 59471,156
3520 POKE 59471,157
3530 POKE 59471,156
3540 A=PEEK(59424)
3550 PRINT A
3560 RETURN
3570 POKE 59471,148
3580 POKE 59471,149
3590 POKE 59471,148
3600 A=PEEK(59424)
3610 PRINT A
3620 RETURN
3630 FOR I=1 TO 4 STEP 1
3640 POKE 59471,132
3650 POKE 59471,133
3660 POKE 59471,132
3670 NEXT I
3680 RETURN
3690 POKE 59471,24
3700 POKE 59471,25
```

```
3720 POKE 59471,24
3730 RETURN
3740 POKE 59471,16
3750 POKE 59471,17
3760 POKE 59471,16
3770 RETURN
3780 POKE 59471,28
3790 POKE 59471,29
3800 POKE 59471,28
3802 A=PEEK(59424)
3804 PRINT A
3810 RETURN
3820 POKE 59471,20
3830 POKE 59471,21
3840 POKE 59471,20
3842 A=PEEK(59424)
3844 PRINT A
3850 RETURN
3860 FOR I=1 TO 4 STEP 1
3870 POKE 59471,4
3880 POKE 59471,5
3890 POKE 59471,4
3900 NEXT I
3910 RETURN
3920 FOR I=1 TO 4 STEP 1
3930 POKE 59471,196
3940 POKE 59471,197
3950 POKE 59471,196
3960 NEXT I
3970 RETURN
3980 POKE 59471,92
3990 GOTO 160
4000 POKE 59471,148
4010 GOTO 820
4020 POKE 59471,156
4030 GOTO 820
4040 POKE 59471,28
4050 GOTO 1600
4060 POKE 59471,20
4070 GOTO 1600
5000 PRINT"CHOIX DU PROCESSEUR A PILOTER EN AUT"
5010 INPUT A$
5020 IF A$="PC1" THEN 5060
5050 GOTO 5
5060 PRINT"CHOIX DU MODE:ECRITURE/LECTURE ENPC1"
5070 INPUT CHOIX$
5080 IF CHOIX$="LECTURE"THEN 5350
5090 IF CHOIX$="ECRITURE"THEN 5110
5100 GOTO 500
5110 PRINT"ECRITURE EN AUT DU PROG ESSAI EN M(PC1)"
5120 GOSUB 3000
5140 GOSUB 3060
5150 DIM A(116)
5170 FOR I=0 TO 115
5180 READ A(I)
5190 POKE 59426,A(I)
5200 GOSUB 3090
5210 NEXT I
5220 POKE 59471,92
5230 PRINT"FIN D'ECRITURE EN AUT DU CI"
5240 GOSUB 3060
5250 DIM B(116)
5270 FOR I=0 TO 115
5280 READ B(I)
5290 POKE 59426,B(I)
```

```
GOSUB 3130
POKE &347F,34
PRINT"FIN D'ECRITURE EN AUT DU PC"
GOSUB 3000
GOTO 3000
PRINT"LECTURE AUT DU PROG EN MKFC1)"
GOSUB 3170
PRINT"LEC AUT DU CI EN MKFC1)"
POKE &347F,92
```

°

° °

Nous donnons, dans cette partie, les autres mots de commandes qu'on peut utiliser pour l'exécution, l'écriture ou la lecture d'un programme dans un processeur.

MOTS de COMMANDES CORRESPONDANT A L'EXECUTION D'UN PROGRAMME EN P1

Ce mot de commande est constitué par huit signaux dont nous avons donné la signification au paragraphe IV-4-1 page IV-11.

Automate	Mode d'exécution	Mot de commande à envoyer (Code décimal)
P1	Automatique	70 *
	Pas à Pas	68 **

* signifie que ce mot de commande n'est pas unique, il en existe d'autres : 78-71-79, ils ont la même fonction : exécution en automatique du programme en P1

** signifie que ce mot de commande n'est pas unique, il en existe d'autres : 64-72-76, ils ont la même fonction : exécution en Pas-à-Pas du programme en PC1.

En mode de Pas-à-Pas, l'incrémentation du compteur ordinalde P1 se fait par l'envoi sur le BUS de commande d'une séquence de mots de commandes donnée en code décimal 68-69-68-69-68-69-68-69.

MOTS DE COMMANDES POUR L'EXECUTION DU PROGRAMME en PC2

Le mot de commande est constitué par huit signaux dont nous avons donné la signification au paragraphe IV-4-1, page IV-11.

Automate	Mode d'exécution	Mot de commande à envoyer (Code décimal)
PC2	Automatique	134 *
	Pas-à-Pas	132 **

* signifie que ce mot de commande n'est pas unique, il en existe d'autres : 130-138-142, ils ont la même fonction : exécution en automatique du programme en PC2

** signifie que ce mot de commande n'est pas unique, il en existe d'autres : 128-136-140, ils ont la même fonction : exécution en Pas-à-Pas du programme en PC2.

Par contre, en mode Pas-à-Pas, l'incrémentation du compteur ordinal se fait par l'envoi d'une séquence, de mots de commandes donnée en code décimal : 132-133-132-133-132-133-132-133.

...

MOT DE COMMANDE POUR L'EXECUTION DES PROGRAMMES EN PC1, PC2 et PS, ensemble en parallèle

Le mot de commande est constitué par huit signaux dont nous donnons la signification au paragraphe IV-4-1, page IV.11

Automate	Mode d'exécution	Mot de commande à envoyer
		Code décimal
PC1, PC2 et PS ensemble en même temps	Automatique	198 *
	Pas-à-Pas	196 **

* signifie que ce mot de commande n'est pas unique, il en existe d'autres : 194-202-206, ils ont la même fonction : exécution en parallèle en mode automatique des programmes en PC1, PC2 et PS.

** signifie que ce mot de commande n'est pas unique, il en existe d'autres : 192-200-204, ils ont la même fonction : exécution en parallèle en mode Pas-à-Pas des programmes de PC1, PC2 et PS.

En mode Pas-à-Pas, l'incrémentation du compteur ordinal se fait par l'envoi, sur le BUS de commandes, d'une séquence de mots de commandes donnée en code décimal 196-197-196-197-196-197-196-197

MOTS DE COMMANDE POUR L'EXECUTION D'UN PROGRAMME en PS seul

Ce mot de commande est décrit par huit signaux dont nous donnons la signification au paragraphe IV-4-1, page IV.11.

...

Automate	Mode d'exécution	Mot de commande à envoyer
		Code décimal
PS	Automatique	6 *
	Pas-à-Pas	4 **

* signifie que ce mot de commande n'est pas unique, il en existe d'autres ayant la même fonction (2,10,14) exécution en automatique

** signifie que ce mot de commande n'est pas unique, il en existe d'autres ayant la même fonction (0,8,12) exécution en Pas-à-Pas.

En exécution Pas-à-Pas, l'incrémentation du compteur ordinal se fait par l'envoi sur le BUS de commandes d'une séquence de mots de commandes donnée en code décimal : 4-5-4-5-4-5-4-5.

° ° ° °

B I B L I O G R A P H I E

[1] M. BLANCHARD. Comprendre, maîtriser et appliquer le GRAFCET. Cépadues Editions 1979.

[2] SYLVAIN THELLIEZ et J.M. TOULOTTE. GRAFCET et logique industrielle programmée. Editions Eyrolles 1980.

[3] J.P. PARSY et J.M. TOULOTTE. A Method for decomposing Interpreted Petri nets And Its Utilization. Digital Processes, 5 (1979) 223-234

[4] Charles ANDRE, Sur une méthode de conception assistée par ordinateur des systèmes logiques à évolutions simultanées. Thèse de 3ème cycle, NICE, Juin 1975.

[5] Michel AUGUIN, Conception des systèmes de commande à l'aide de réseaux logiques programmables, Thèse de 3ème cycle, NICE, Novembre 1979.

[6] Pierre VERNEL, Conception et Réalisation d'un multi-calculateur temps réel à Grande Sûreté de Fonctionnement. Thèse de Docteur d'Etat (Sciences) Polytechnique de Lorraine. Juin 1975

[7] THE EUROPEAN CMOS SELECTION (Motorola Semiconductors) 1977

[8] INTEL DATA , Catalog 1977

[9] THE TTL DATA BOOK FO DESIGN ENGINEERS

...

[10] CBM.2001 - 16, - 32, 3016, 3032, PROFESSIONAL Computer User MANUAL : Juin 1979

[11] INDUSTRIAL CONTROL UNIT HANBOOK MC 14500B MOTOROLA

[12] Jean DEFRENNE, Implantation de réseaux de Pétri sur Automate biprocesseur à haute sûreté de Fonctionnement. Thèse 3ème cycle, Juin 1979, LILLE

[13] J.M. TOULOTTE, Réseaux de PETRI et Automates programmables. Automatisme, Tome XXIII n° 7-8. Juillet-Août 1978, pp. 200-211.

[14] T. MAURIN & M. ROBIN, GRAFCET, une traduction microprogrammée. Mesures, Régulation, Automatisme, Mai 1979, pp. 75-80.

[15] Claude LAURGEAU, Considérations sur la place des automates programmables en Informatique Industrielle Nouvel Automatisme. Avril 1980, PP 39-44.

[16] J.P. MUSSE, J. P. DRAPIER, Olivier DOUCHIN. Système Modulaire multitâches pour commande de processus. Nouvel Automatisme. Sept-Oct.1979; pp. 51-58.

[17] C. ANDRE, F. BOERI et J. MARIN, Synthèse et réalisation des systèmes logiques à Evolutions Simultanées. RAIRO n° Avril 1976, pp. 67 à 86.

[18] AFCET, Rapport de la Commission de normalisation de la représentation du cahier des charges d'un automatisme logique (Groupe de travail AFCET "systèmes logiques") Revue Automatisme, Mars-Avril 1978 pp.66-83.

[19] R. VALETTE, Etude Comparative de deux outils de représentation GRAFCET et Réseau de PETRI. Nouvel Automatisme. Décembre 1978, pp. 377-382

[20] F. CAVAZZINI, Exemple d'utilisation du GRAFCET pour la spécification et la synthèse d'un automatisme Automatismes Industriels n° 77 Mai 1979, PP. 47-52.

[21] Kamal QUOTB, ETUDE et Réalisation d'un automate programmable à l'aide d'un microprocesseur à tranches (Série 3000). DEA d'Automatique . 1978/1979. LILLE.

22 Claude BRIE, Panorama des processus en tranches; les unités de traitement. Avril 1979. Nouvel automatisme pp 39.48

23 Jean-Marc TOULOTTE, La fiche technique idéale d'un automate programmable. Nouvel automatisme Septembre Octobre 1979

24 SYNERTEK, 3050 Conorado Drive, Santa Clara, CA 95051