UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ALGER
DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTECHNIQUE

FILIERE D'INGENIEUR EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

SIMULATION D'UN PROCESSUS PAR UN SYSTEME MULTIPROCESSEUR

Proposé par : H.TEDJINI Dr Ingénieur

Réalisé par :

OUDJEHANE Badredine

BOUARFA Abdelmajid

JUIN 1982

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE D'ALGER
DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTECHNIQUE

FILIERE D'INGENIEUR EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

SIMULATION D'UN PROCESSUS PAR UN SYSTEME MULTIPROCESSEUR

Proposé par : H.TEDJINI Dr Ingénieur

Réalisé par :

OUDJEHANE Badredine

BOUARFA Abdelmajid

JUIN 1982

P L A N

../..

NOS PLUS SINCERES REMERCIEMENTS A MONSIEUR TEDJINI DE NOUS AVOIR ACCEPTER DANS SON EQUIPE, ET POUR LES CONSEILS PRODIGUES TOUT AU LONG DE NOTRE TRAVAIL.

NOUS REMERCIONS EGALEMENT MONSIEUR HALIMI QUI A BIEN VOULU, PAR SON EXPERIENCE ET SA COMPETENCE, NOUS VENIR EN AIDE.

LES MOYENS HUMAINSET MATERIELS MIS A NOTRE DISPOSITION POUR LA CONCEPTION DE CET OUVRAGE, NOUS ONT ETE GRACIEUSEMENT OFFERTS. NOUS EN SOMMES RECONNAISSANTS ET NOUS REMERCIONS TOUTES LES PERSONNES, DE PRES OU DE LOIN, QUI NOUS ONT PORTE ASSISTANCE.

INTRODUCTION

La simulation, en général, est une expérience artificielle sur toutes sortes de phénomènes physiques, économiques et sociaux.

Celle-ci peut apporter des information précieuses sur les performances des systèmes ou le comportement d'un phénomène physique. Simuler le réel, c'est en concevoir une représentation sur laquelle essais et mesures sont facilement réalisables.

La simulation a pris son essor après l'apparition de l'ordinateur. Cette technique est utilisée à des fins diverses :

ANALYSES DES SYSTEMES, LEURS CONSTRUCTIONS, COMME AIDE A LA FORMATION ET AUSSI BIEN DANS L'ENSEIGNEMENT QUE DANS L'ENTRAINEMENT DU PERSONNEL.

Donc, par définition, la simulation est une expérimentation sur le modèle d'un système. Le modèle peut être un modèle mathématique décrivant le comportement de chaque composant ou grandeur du système. Dans notre cas, le phénomène physique à simuler sera décrit par un modèle mathématique sous forme de système d'équations différentielles.

Plusieurs unités centrales secondaires (dites "Esclaves") travaillent en parallèle sous le contrôle d'un module principal (Maître). Les modules secondaires seront chargés de la résolution des équations régissant

../..

un réacteur nuéclaire, alors que le module principal coordonnera le dialogue entre les différents esclaves et dirigera les sorties (ou entrées) sur les différentes périphéries.

C'est dans le cadre du système multiprocessing qu'intervient notre contribution au projet entrepris par la division V du CSTN, pour la simulation d'un réacteur nucléaire. Nous avons été chargés de la réalisation et mise au point de la carte esclave, de l'étude et réalisation d'une carte maître, et d'établir le logiciel de dialogue dans le système.

CHAPITRE I

SIMULATION ET STRUCTURE MULTIPROCESSING

1ERE PARTIE : SIMULATION

I - SIMULATION ANALOGIQUE -

La réalisation d'une simulation nécessite l'utilisation d'un computer analogique.

Un computer moderne consiste en un ensemble d'éléments électroniques. Il peut effectuer des opérations complexes (somme, intégration, multiplication, opération logique, fonctions non linéaires...) étant capable de grande performance.

Les variables, dans une simulation analogique, sont des tensions continues, qui correspondent aux quantités physiques du processus à simuler. Celles-ci sont continues et le temps en général est une variable indépendante.

Les sorties peuvent s'effectuer sur oscilloscope, table traçante, etc..

Dans une simulation analogique, la précision est limité à 3 ou 4 nombres , décimales près. Celle-ci est déterminée surtout par la précision des périphéries. Néanmoins, il n'est pas vain d'essayer de l'augmenter car l'exactitude (justesse) même de la simulation ne

dépasse généralement pas 2 ou 3 nombres significatifs.

La réalisation d'une simulation d'un système physique consiste donc en la programmation d'un computer analogique afin de résoudre les équations mathématiques (du modèle correspondant) en tenant compte de tous les éléments nécessaires.

Un système analogique peut-être caractérisé par les propriétés suivantes :

- les varaiables dépendantes sont traitées sous forme continue
- l'exactitude est limitée par la qualité des composants qui équipent le computer
- les calculs sont réalisés en parallèle
- grande vitesse de calcul (la vitesse ne dépend pas de la complexité du problème, mais des caractéristiques des éléments du calculateur)
- capacité de réaliser des opérations, sommes, multiplications, intégrations...
- facilité d'introduction de la structure (Hardware) analogique du système étudié.

II - SIMULATION DIGITALE (Numérique)

1 - Présentation -

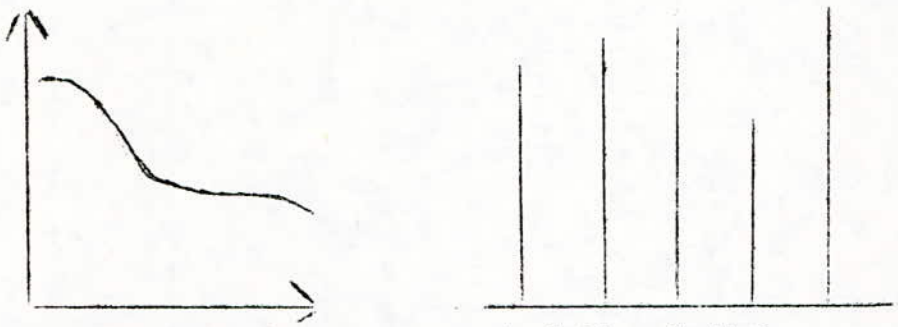
La différence entre analogique et digital peut-être résumé de la façon suivante :

- en analogique, les variables corrélées (dépendantes) sont constamment sous forme continue et peuvent être enregistrées sous forme de figures significatives.

../..

Par contre, en digital, toutes les variables sont sous forme discrète. La précision des données manipulées et enregistrées en sortie, dépend du nombre de digits de la solution et dépend directement de la capacité des registres mémoires.

Les variables sont donc représentées par des nombres discrets (représentant le temps et les paramètres du processus physique).



simulation analogique

simulation digitale

Pour cette simulation, on utilise un computer digital. Certains de ces computers numériques sont conçus pour des applications spécifiques. Les performances de calcul peuvent être alors fixées d'une façon permanente (téléguidage de missiles, contrôle du trafic aérien..)

Tous les computers numériques sont capables de réaliser les opérations élémentaires et quelques manipulations logiques. Pour leur utilisation, on doit écrire un programme d'instruction et le transmettre aux circuits d'entrées. Ces instructions seront alors exécutées en séquence. Le computer digital exécute les calculs en série, à l'inverse de l'analogique qui les exécute en parallèle.

../..

En résumé, la réalisation d'une simulation numérique consiste en la programmation d'une séquence d'expressions arithmétiques dans un computer numérique qui résoudra les équations.

2 - Structures générales -

Un ordinateur numérique est constitué en règle générale de cinq grandes parties :

- Entrée
- Unité de contrôle
- Unité arithmétique
- Mémoire
- Sortie

a) Entrée :

Il y a plusieurs possibilités d'entrée : cartes perforées, rubans perforés, bandes magnétiques, disques souples ou rigides, et aussi par ADC (convertisseur analogique, digital), qui convertit les données analogiques en données numériques.

b) Unité de contrôle :

Cette unité synchronise et dirige pratiquement toutes les autres unités.

c) Unité arithmétique :

Cette unité réalise les opérations arithmétiques de base.

d) Mémoire :

On y conserve les informations nécessaires aux calculs.

e) Sorties :

On peut sortir les résultats par différents moyens : bandes magnétiques, visu, table traçante, imprimante. On peut utiliser aussi un ADC pour sortir les résultats sous forme analogique après conversion.

La précision des computers numériques varie de 6 à 16 décimals digits.

3 - Précision et autres caractéristiques -

Le calcul est réalisé en binaire pure grâce à des circuits électroniques. Les bruits et les signaux n'ont pas d'effets et seule la défection d'un composant entraine la totale défection. Par contre, la précision n'est pas absolue.

La simulation donne seulement une approximation des réponses du système étudié. Elle dépend de chaque calcul, de la manière dont sont organisées les séquences et de l'approximation mathématique du modèle.

La précision reste quand même contrôlable par les constructeurs alors que dans la simulation analogique, elle dépend largement de la précision avec laquelle sont réalisées les opérations de calcul.

Les computers numériques ont à l'heure actuelle, des caractéristiques intéressantes telles que leur haute précision, la souplesse d'utilisation. En outre, l'utilisation d'un système de multiplexage donne à la machine, dans une structure à ressources partagées, la possibilité de simuler plusieurs systèmes simultanément.

../..

Quelques caractéristiques des computers numériques :

- Acquisition de données sous forme de quanta ou sous forme discrète
- Traitement en série des calculs
- L'exactitude est relativement indépendante de la qualité des composants du système. Elle est principalement déterminée par le nombre de bits contenus dans les registres mémoires et par la technique numérique sélectionnée pour résoudre le problème.
- Possibilités d'exécuter un nombre limité d'opérations arithmétiques. Les opérations plus complexes (intégration, différentiation) étant effectuées grâce à des techniques d'approximation.
- Facilité de réaliser des opérations logiques
- Facilité de procéder automatiquement à des changements et à contrôler la topologie des données.

4 - Application de la simulation digitale -

a) Analyse des systèmes :

Systèmes physiques tels que les systèmes aérodynamiques, hydrauliques ...

b) Système d'évaluation : (simulateur)

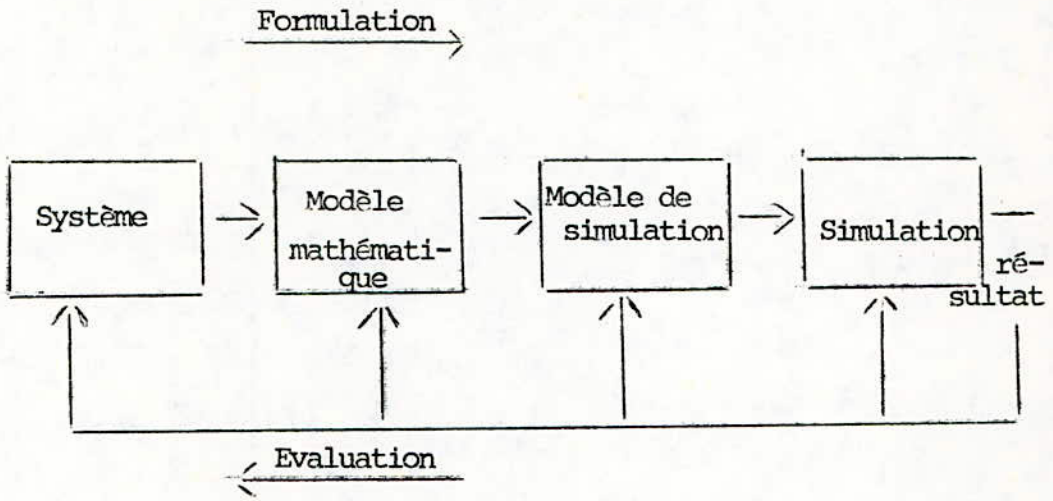
Exemple : système de navigation

Ce système est généralement couplé à un autre computer digital.

c) Simulateur de système discret :

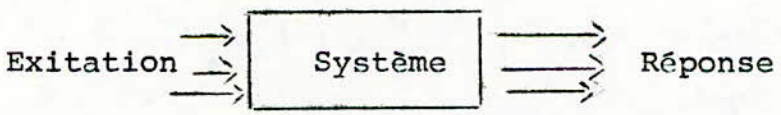
Contrôle du trafic aérien et routier.

../..

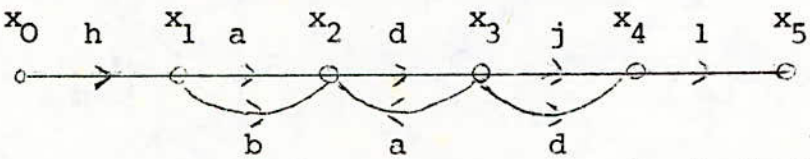


5 - La simulation -

Pour réaliser une simulation, on doit d'abord examiner le système isolé et définir les entrées et sorties variables (Exitation et réponse).



On donne ensuite le modèle mathématique.



Graphe de fluence

../..

Equations :

XO : entrée

X1 = h X O

$X2 = a X_1 + b X_1 + a X_3$

$X3 = d X_2$

$X4 = j X_3 + d X_3$

X5 : sortie

$X5 = X_4$

Pour obtenir ce modèle mathématique, on procède comme suit :

- on définit la forme des équations du système
- on détermine tous les paramètres associés

Pour terminer, on rentre dans le processus de simulation. On donne un modèle discret équivalent au modèle continu, on obtient les équations et les programmes de résolution, qui seront exécutés par le computer. Ainsi la simulation sera réalisée.

6 - Technique de simulation -

a) Temps réel :

La sortie des résultats n' lieu qu'à intervalle de temps bien déterminé. Le temps qui s'écoule entre deux résultats doit être fixé de telle manière, que les réponses ne changent pas de façon significative pendant la période d'échantillonnage.

Le temps réel est performant dans les systèmes complexes représentés par plusieurs équations mathématiques,

../..

car le computer digital travaille par séquence.On devra faire en sorte que le temps d'exécution d'une séquence d'instruction soit plus petit que le temps séparant deux sorties de résultats.

b) Méthode numérique :

Les méthodes numériques (ex : d'itération), avec l'aide de l'ordinateur, permettent d'aborder et de résoudre nombre de problèmes avec de grande performance.

Le choix de la méthode(numérique) doit tenir compte des possibilités du computer utilisé.

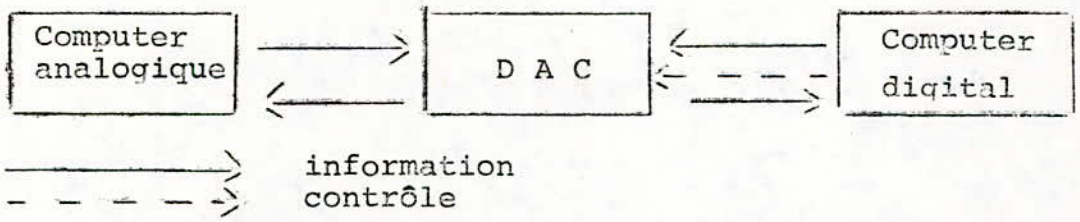
c) Méthodes opérationnelles :

Le principe de ces méthodes est de transformer le calcul à réaliser en un autre plus simple. Ex : transformé en Z.Un point intéressant : le modèle discret est obtenu alors par de simples manipulations algébriques.

III - SIMULATIONS HYBRIDES -

Dans certains cas, seule la simulation hybride permet d'avoir des résultats. Certains systèmes conellés nécessitent ce type de simulation :

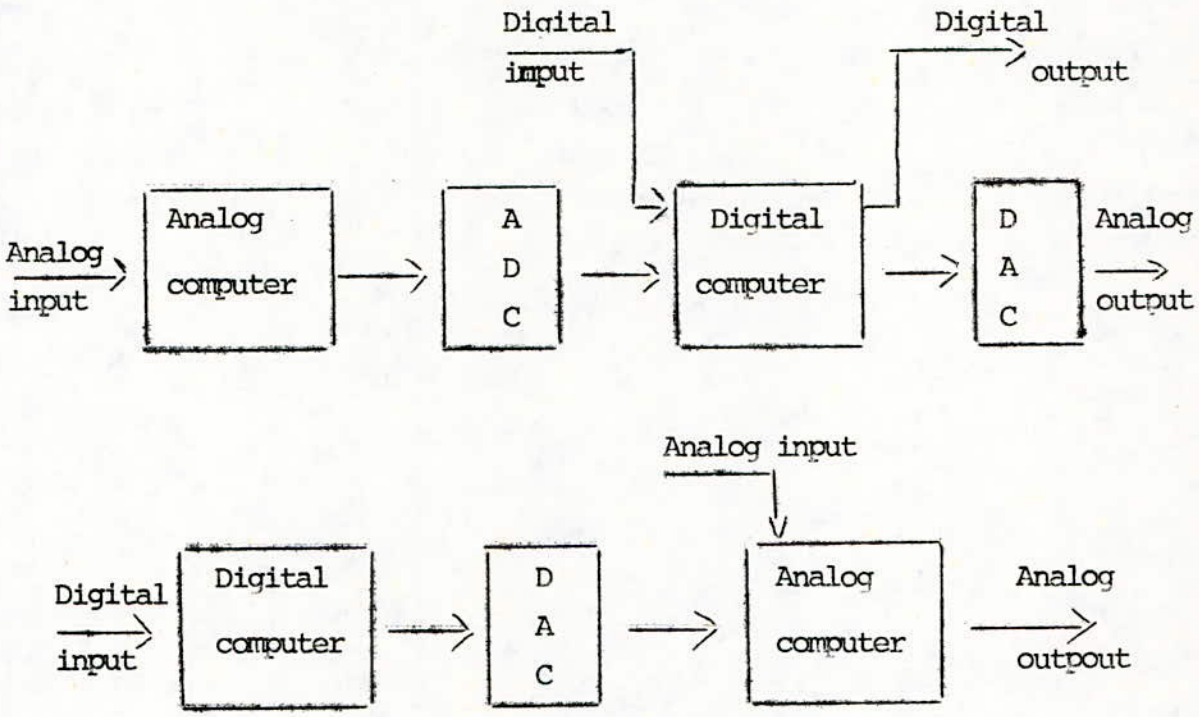
- Les systèmes dont la complexité entraine un temps de calcul long
- Les systèmes dont les réponses sont très rapides
- Certaines applications à temps réel.



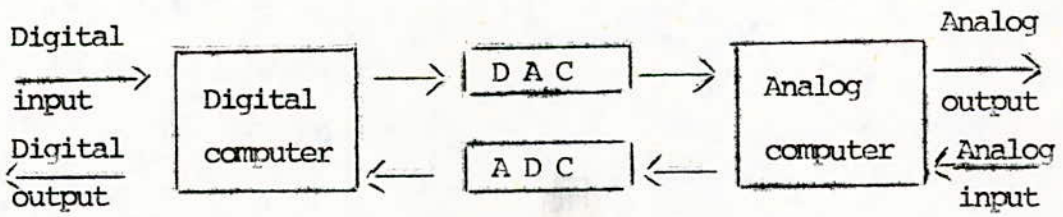
Les connections entre les deux computers sont réalisées par des convertisseurs A/D (analogique/digital). En général, la partie numérique est chargée du contrôle et de la synchronisation.

../..

Pour la distribution des tâches, il existe plusieurs structures possibles.



Deux types de structures de système. hybride:
unilatéral



Structure de système hybride bilatéral

../..

2EME PARTIE : MULTIPROCESSING

I - SYSTEME A MULTIPROCESSEUR -

Au sein du CSTN, dans le cadre du projet de simulation d'un réacteur nucléaire, le système de simulation choisi est à multimicroprocesseur, bâti autour du microprocesseur Motorola MC 6800; C'est un système MIND.

L'intérêt du multiprocesseur réside dans le fait que les CPU et mémoires sont interconnectés entre eux, la communication étant sous un contrôle commun, et le travail réalisé en parallèle. (figure I - 1)

II - CARACTERISTIQUE D'UN SYSTEME MULTIPROCESSEUR -

Un multiprocesseur est un système contenant deux ou plusieurs microprocesseurs, dont les potentialités sont comparables.

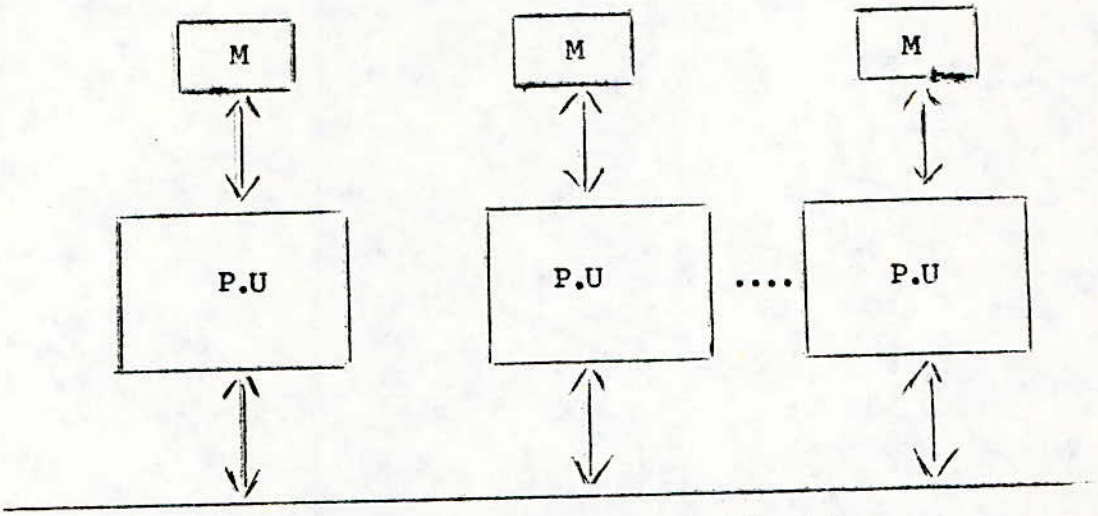
Les processeurs ayant une mémoire locale, partagent l'accès à la mémoire commune, aux canaux entrées/sorties, et à l'unité de contrôle. On a donc certaines ressources qui sont partagées par tous les processeurs (bus commun, mémoire commune ..) (Figure I - 2)

Le système entier est contrôlé par un moniteur, qui gère les inter-actions entre les différents processeurs, chacun étant associé à une variable.

III - TYPES D'ORGANISATION -

On a trois modes possibles d'organisations :

../..



MIND : Multiple Instruction stream multiple Data

Système multiprocesseur

Fig. I - 1

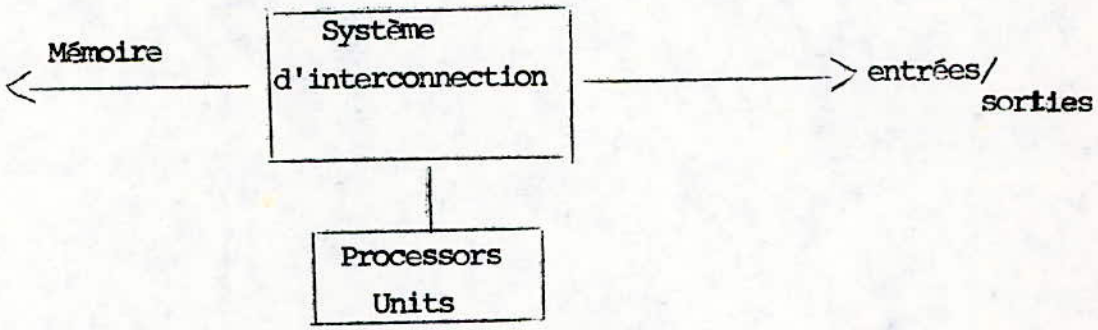


Fig. I - 2

a) Maître - esclave
b) Exécution séparée pour chaque processeur
c) Traitement symétrique pour tous les processeurs.

1 - Mode choisi -

Le mode choisi est le premier.

Son principe sera le suivant :

- le maître est l'unité de contrôle et de synchronisation et a pour tâche, d'empêcher les conflits possibles entres les autres processeurs.

Grâce à l'existence des interconnections entre les différents calculateurs, les échanges se feront à travers une zone mémoire, commune à tous. L'accès à cette zone sera commandé par le maître chargé de la gestion du dialogue. Les esclaves sont vus en définitive comme des périphériques. Cette organisation nécessite donc deux cartes essentielles :

- La carte-maître : carte de commande, comportant aussi le circuit de gestion des interruptions
- La carte-esclave.

2 - Caractère du mode -

Quand un esclave a besoin d'un service, il formule une demande et attend que le programme courant du maître soit interrompu. Ceci fait, la réponse lui est envoyée. Le fait d'avoir un seul processeur (maître) qui gère le tout, diminue les risques de conflits et simplifie le contrôle des tables.

Le système est sujet parfois à des erreurs qui imposent une intervention externe (opérateur) pour un "restart".

La structure est relativement inflexible.

../..

Enfin, dans ce type d'organisation, le support software est simple par comparaison à d'autres types.

IV - CONCEPTION DE LA COMMUNICATION -

Les interconnections entre les variables permettent des échanges de résultats entre les calculateurs à travers la mémoire commune. L'accès à cette mémoire peut créer des conflits.

Pour y remédier, on peut utiliser une matrice Crosbar. Ce choix n'est pas interressant vu la relative simplicité du problème, et la grande complexité de cette matrice. On préfère donc utiliser un système de gestion d'interruption associé au processeur de contrôle. Cette méthode est mieux adaptée aux contraintes imposées.

- Méthode de communication

Deux solutions ont été envisagées pour réaliser la communication (dialogue) entre les unités de calcul.

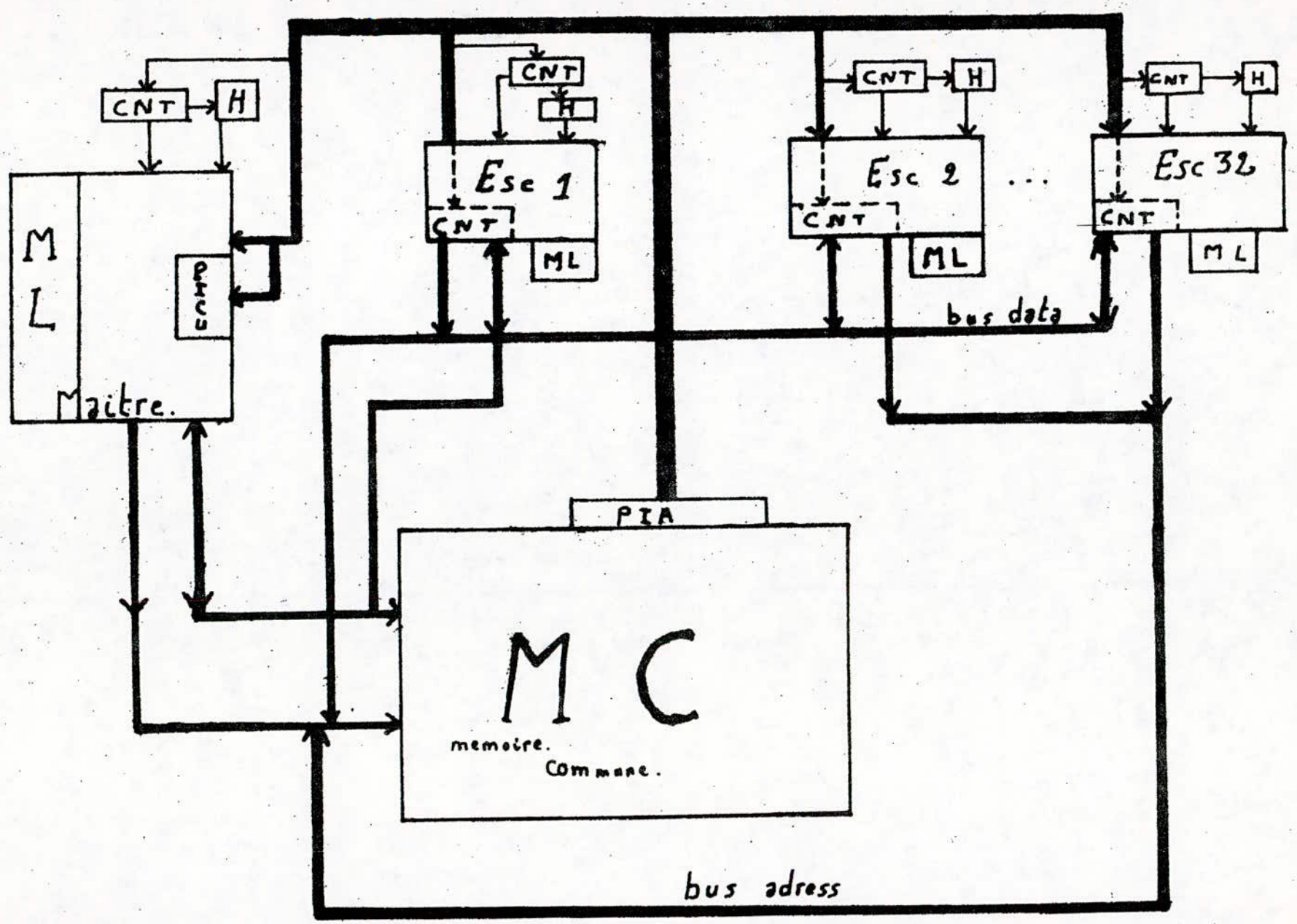
1 - Dialogue par PIA -

Cette méthode consiste en un dialogue maître-esclave à travers des PIA. Elle nécessite donc, pour le maître, autant de PIA qu'il y a d'esclave et un PIA par esclave. Nous n'avons pas retenu cette solution, vu la lourdeur "Hardware" qu'elle engendre.

2 - Utilisation d'une mémoire commune -

Dans ce cas, le dialogue est réalisé à travers une zone mémoire commune, par bus partagé. Cette méthode,par ressources partagées (bus,mémoire), que nous avons choisie, permet une grande économie de circuit.

../..



C.N.T : Contrôle.

CHAPITRE II

- CARTE MAITRE -

1ÉRE PARTIE - HARDWARE

I - INTRODUCTION -

La carte maître forme un micro-ordinateur complet comportant une unité centrale constituée autour d'un micro-processeur Motorola MC 6800. Le maître a la possibilité de travailler indépendemment du système dans sa mémoire locale ou dans la zone commune. De ce fait, son unité centrale sera formée de buffers d'entrées-sorties, pouvant isoler le maître du système, d'une horloge et de RAM et ROM pour sa mémoire locale.

Comme on l'a dit précédemment, le maître coordonne le dialogue entre les différents esclaves. En effet, ceux-ci étant chargé de résoudre chacun une équation différentielle, ils ne pourront commencer leur calcul que si le maître les y autorise, en leur envoyant les paramètres et les variables nécessaires. De même, son calcul terminé, l'esclave a la possibilité de prévenir le maître par le biais d'une interruption. Celui-ci doit donc pouvoir gérer ces interruptions suivant un ordre de priorité bien établi, de façon à éviter des conflits, et générer des appels aux esclaves afin de leur transmettre certaines données, ainsi qu'un signal d'acquitte-

../..

ment lors d'une demande d'interruption.

La carte maître sera donc constituée d'un encodeur de priorité et d'un PIA (interface adaptateur pour périphérique).

II - UNITE CENTRALE -

1 - Description :

Nous ne nous étendrons pas sur le composant principal de cette unité centrale qui est le micro-processeur MC 6800. Nous rappelons que ce circuit intégré, bien connu et très répandu est un micro-processeur monolittique 8 bits, compatible TTL, ne demandant qu'une alimentation de + 5V. Le MC 6800 peut adresser 64 K octets de mémoire grâce à ses 16 lignes d'adresses. Le bus de données, 8 bits, bidirectionnel à sorties trois états permet l'accès direct en mémoire. (Pour le brochage se reporter en annexe).

En plus du 6800, l'unité centrale comporte des circuits d'interface d'adresses et de données. Ces circuits sont commandés par une logique de commande et de contrôle. Celle-ci détermine s'il faut autoriser une entrée ou une sortie de données suivant le cas d'une lecture ou d'une écriture, et activer ou désactiver les interfaces d'adresses suivant que le MPU s'adresse à sa mémoire locale ou à la mémoire commune. Ces circuits sont les MC 8T26 et MC 8T95.

- Buffers MC 8T26 -

Les MC 8T26 sont des transmetteurs-récepteurs de

bus, 3 états, bidirectionnels, inverseurs. Suivant que le micro-processeur reçoit ou transmet une donnée les lignes correspondantes sont entrantes ou sortantes. (Brochage et table de vérité en annexe).

- Buffers MC 8T95 -

Les MC 8T95 sont des amplificateurs, non inverseurs, trois états, unidirectionnels. Ils sont utilisés pour lire ou écrire des données dans une mémoire. (Brochage et table de vérité en annexe).

- Horloge -

L'horloge que nous utiliserons est la MC 6871. Ce générateur d'horloge est prévu pour fournir les signaux d'horloge non chevauchant Ø1 et Ø2 nécessaires au fonctionnement du micro-processeur.

Outre ces deux signaux, cette horloge possède une mémoire prête qui est une entrée asynchrone utilisée pour bloquer les phases d'horloge du micro-processeur dans l'état Ø1 bas, Ø2 haut pour l'utilisation avec une mémoire lente.(Pour le brochage, voir en annexe)

2 - Fonctionnement : Logique de lecture - écriture

Le CPU maître a la possibilité de lire ou d'écrire une information dans sa mémoire locale ou commune. Ces deux opérations se font suivant une logique que nous décrivons ci-dessous :

a) Logique d'écriture :

L'opération d'écriture nécessite que la commande

../..

$R/\overline{W}$ = 1, que le bus de données du MPU soit activé ; soit DBE = 1, et que le bus des adresses soit disponible afin de pouvoir adresser le mot à écrire ; soit BA = 0.
Le signal d'écriture est donc :

$$SE = \overline{R/\overline{W}} \cdot \overline{BA} \cdot \overline{DBE}$$

Il est bon de noter que l'entrée DBE du MPU sera généralement commandée par la phase Ø2 de l'horloge.
Ainsi :

$$SE = \overline{R/\overline{W}} \cdot \overline{BA} \cdot Ø2 \text{ (TTL)}$$

Ce signal sera appliqué sur la borne 15 des 8T26, commande activant les buffers de données dans le sens sortant.

b) Logique de lecture :

Quand le MPU est dans l'état de lecture, la sortie $R/\overline{W}$ est à 1 (état haut). De plus, le bus de données du MPU devant être activé, on doit avoir Ø2 (TTL) = 1 ; on aura donc comme signal de lecture :

$$SL = R/\overline{W} \cdot Ø2 \text{ (TTL)}$$

Ce signal est appliqué sur la borne 1 des 8T26, mais inversé ($\overline{SL}$) car pour ces buffers, le signal d'activation de l'entrée des données doit être au niveau bas (0 logique).

../..

Table de vérité

$R/\overline{W}$	BA	DBE	SE (15)	SL (1)	Fonction
0	0	0	0	1	HI
0	0	1	1	1	Ecriture
0	1	0	0	1	HI
0	1	1	1	1	HI
1	0	0	0	1	HI
1	0	1	0	0	Lecture
1	1	0	0	1	HI
1	1	1	0	0	Lecture

(Logique figure II - 1)

Technique d'aiguillonnage des bus d'adresses

Afin d'éviter tout conflit, il est nécessaire d'isoler le maître du bus commun quand un des esclaves l'utilise pour transmettre ou venir chercher une information en mémoire commune.

L'accès en mémoire commune se faisant à travers les 8T26 pour les données et les 8T95 pour les adresses, on a donc envisagé de les valider si le maître utilise le bus commun et de les mettre en haute impédance dans le cas contraire.

../..

Activation et désactivation des 8T95

Pour ces opérations, on utilisera la logique de blocage du maître que l'on développera un peu plus loin. Lorsque le maître envoie un acquittement ou un appel à un esclave, le MPU maître se bloque automatiquement par l'envoi d'un niveau bas sur son entrée $\overline{\text{Halt}}$. Ce niveau bas provient de la sortie Q d'une bascule D.

Pour isoler le maître du bus commun, on désactive donc les 8T95. La désactivation des 8T95 s'effectue en envoyant un niveau haut (1 logique) sur l'une des 2 entrées de validation ($\overline{\text{validation 1}}$, borne 1 et $\overline{\text{validation 2}}$, borne 15). On applique donc la sortie $\overline{Q}$ sur la borne 15 et le signal BA (bus available) sur la borne 1 . Ainsi le maître se trouve isolé, lorsque l'un des esclaves travaille en mémoire commune.

Inversement, si le maître n'envoie aucun acquittement ni aucun appel, $\overline{Q}$ se trouve à 0, BA retombe à un niveau bas (celui-ci ne passant à 1 que pour un mode interruptible), les bornes 1 et 15 des 8T95 sont à l'état bas, on est donc dans le mode passant.

Activation et désactivation des 8T26

Comme pour les 8T95, on utilisera aussi pour les 8T26, la logique de blocage du maître pour effectuer ces deux opérations.

En zone commune, l'opération de lecture est donnée par l'état 00 sur les bornes 1 et 15 des 8T26. Tandis que l'état 11 caractérise l'opération d'écriture.

../..

Logique de lecture - ecriture

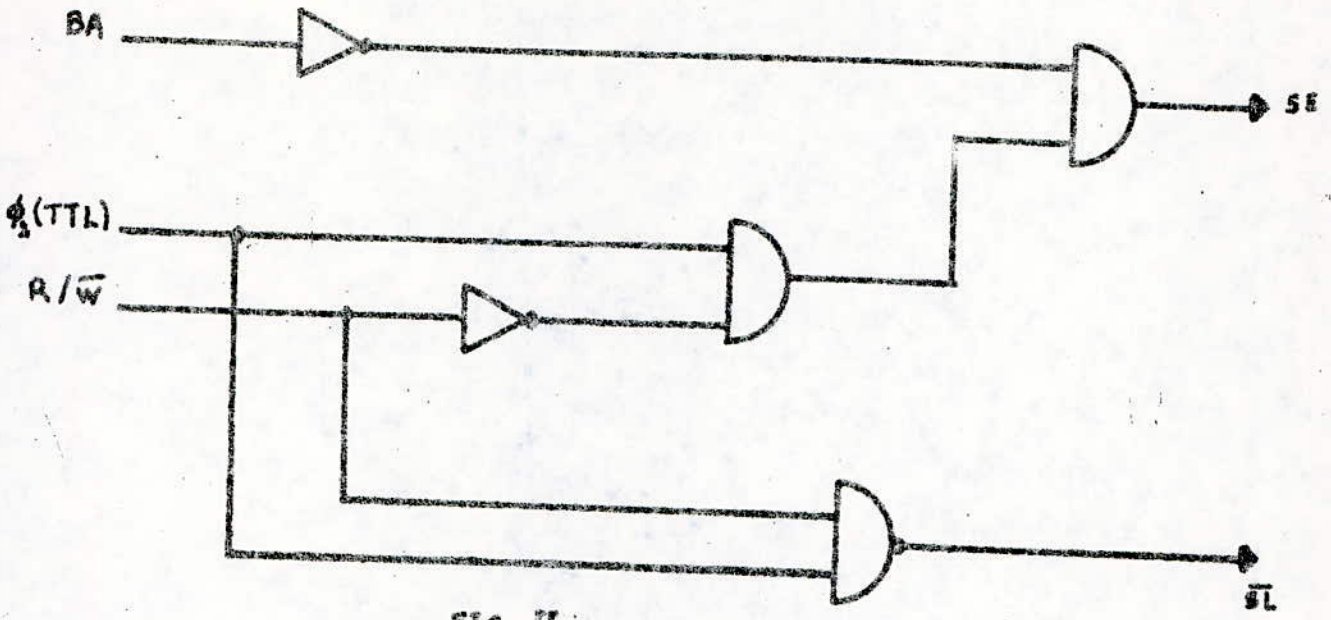


FIG: II 1

aiguillonage des BT26

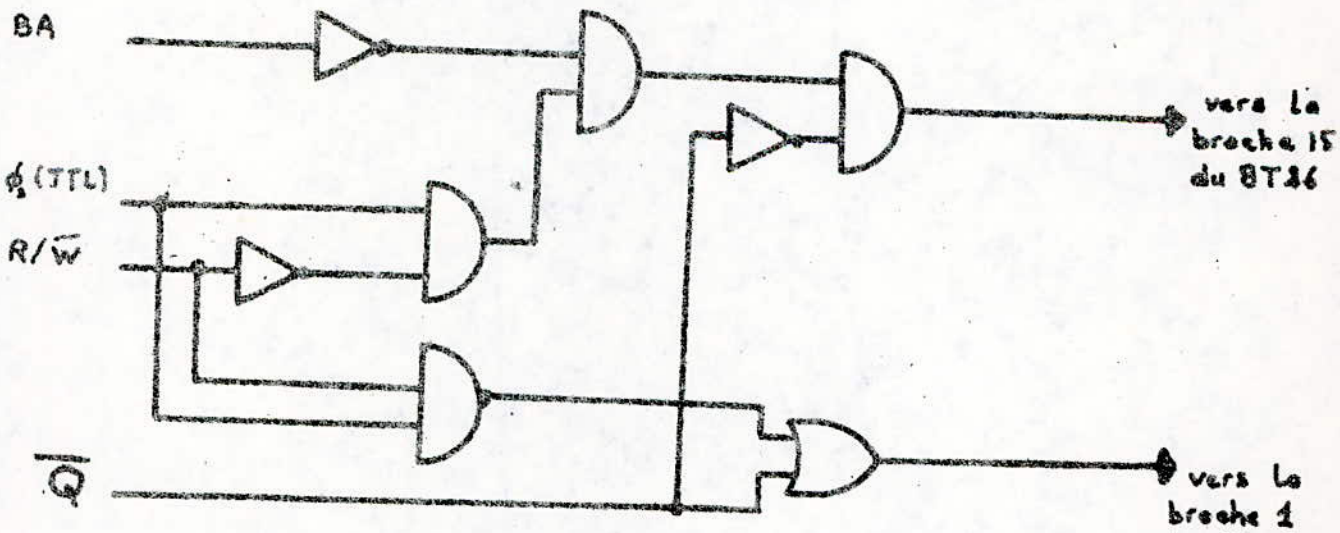


FIG: II 2

On adjoint donc à la logique d'écriture, le signal Q par l'intermédiaire d'une porte "ET" et à la logique de lecture, le signal $\overline{Q}$ par l'intermédiaire d'une porte "OU". Ainsi lors d'un appel ou d'un acquittement du maître, on a la configuration 10 respectivement sur les bornes 1 et 15 des 8T26, les mettant en haute impédance (Logique figure II - 2).

III - ENCODEUR DE PRIORITE -

1 - Introduction :

Le maître devant gérer 32 esclaves, celui-ci peut être interrompu à n'importe quel moment par l'un ou plusieurs d'entre eux. Il y a donc risque de conflit.

Pour y remédier, on établit un ordre de priorité des 32 interruptions, faisant en sorte que le maître ne prenne en compte que la plus prioritaire.

L'encodeur de priorité est donc un module capable de prendre en compte 32 interruptions. Il permet de répercuter toutes les demandes vers le maître, en les mémorisant et en ne laissant passer que la plus prioritaire. Nous fixerons un ordre de priorité décroissant, l'esclave 1 étant le plus prioritaire.

2 - Principe de priorité :

L'ordre de priorité étant fixé de 1 à 32, si, par exemple l'esclave 2 fait une demande d'interruption alors que le maître exécute un sous-programme relatif à l'esclave 1, l'interruption 2 en sera prise en compte que quand le maître aura terminé l'exécution du sous-programme

d'interruption de l'esclave 1. Inversement, si l'esclave 2 interrompt le maître alors que celui-ci exécute un sous-programme venant d'un esclave dont le numéro est supérieur à 2, le maître sauvegarde le sous-programme qu'il exécute et prend en compte l'interruption de l'esclave 2.

Ce principe est celui du contrôleur d'interruption plus communément appelé PICU (Priority Interrupt Control Unit), qui est un circuit de INTEL : le 8214. Ce circuit peut accepter jusqu'à 8 demandes d'interruption, déterminer la plus prioritaire, la comparer à celle écrite dans son registre "CSR" et générer une demande d'interruption avec un vecteur identifiant le sous-programme correspondant.

3 - Description :

Le système possédant 32 esclaves, 32 niveaux d'interruptions sont donc nécessaires. Pour cela, on utilisera quatre 8214 placés en cascade. Pour avoir l'acquisition du 8214 par le MPU, on passe par le 8212 (port d'entrée/sortie à 8 bits), qui joue le rôle de port d'entrée d'information issu de l'encodeur de priorité (8214) vers l'$\overline{IRQ}$ du MPU maître.

Le module chargé de la gestion des interruptions sera donc constitué de quatre PICU (8214), d'un port d'entrée/sortie (8212) et de 8 comparateurs pour le circuit de décodage.

3.1 - Analyse des différents composants :

a) Port d'entrée/sortie (8212)

Le 8212 se compose de 8 bascules D, de buffers de sorties à 3 états et d'une bascule SR (Service request flip-flop) pour générer et contrôler les interruptions

../..

vers le MPU maître.

La sortie Q de ces bascules D suit la donnée D quand l'horloge C est à l'état haut. Quand celle-ci retourne à l'état bas, il y a basculement. Le signal asynchrone $\overline{CLR}$ permet la remise à 0 des bascules données. Les sortis Q des bascules sont reliées à des buffers de sortie 3 états, non inverseurs. Ces buffers sont connectés entre eux par une ligne de contrôle EN qui active ces derniers pour transmettre une donnée ou les désactive en les mettant à l'état haute impédance.

Contrôle logique du 8212 -

Le 8212 a des entrées de contrôle $\overline{DS1}$, DS2, MD et STB.

. Entrée $\overline{DS1}$ et DS2 :

Ces 2 entrées sont utilisées pour sélectionner le circuit. La sélection est établie quand $\overline{DS1}$ = 0 et DS2 = 1

. Entrée MD :

Cette entrée est utilisée pour contrôler l'état des buffers de sorties.

. Entrée STB :

L'entrée STB permet la mise à 0 de la bascule SR.

Bascule SR flip-flop -

Cette bascule est utilisée pour générer et contrôler les interruptions dans le système. Elle est mise à "1" par l'entrée $\overline{CLR}$ d'une façon asynchrone. Quand SR est à l'état "1", on est dans l'état non interruptible.
La sortie Q de la bascule est connectée à l'entrée inversée d'un NOR. L'autre entrée non-inversée est reliée au circuit logique de sélection ($\overline{DS1}$,DS2). La sortie du NOR (INT) est activée à l'état bas et c'est l'état interruptible. (configuration interne, voir annexe).

../..

b) Encodeur de priorité 8214

Les 8 demandes d'interruption, activées à l'état bas arrivent à l'encodeur de priorité. Ce circuit détermine la demande la plus prioritaire, comme fixée par l'utilisateur.

La logique de l'encodeur est telle que si plusieurs demandes d'interruptions arrivent simultanément, la plus prioritaire sera prise en compte et un code binaire (modulo 8) correspondant à la demande activée sera envoyé. L'encodeur possède une bascule pour stocker la demande.

Current Stakes Register CSR -

Le CSR est une simple bascule à deux entrées, considérée comme port adressable pour le micro-processeur. Il est chargé quand $\overline{ECS}$ = 0, quand une interruption est envoyée au système, l'enregistreur envoie un code binaire (modulo 8) qui représente l'interruption activée.

Cette valeur est stockée dans le CSR pour être comparée à toute autre interruption par le comparateur de priorité. Le code binaire de l'interruption courante est inscrit dans le CSR pour être utilisé comme référence pour la comparaison.

On notera que la 4ème entrée $\overline{SGS}$ est une partie de la valeur écrite par le programmateur. Le comparateur de priorité mettra un "1" à la sortie qui indique que le niveau de la demande d'interruption est plus grande que celui du CSR.

La comparaison désactivée, le $\overline{SGS}$ permet au 8214 de générer une interruption au système basé sur la logique

../..

de l'encodeur de priorité.

Signaux de contrôle du 8214 -

. Entrée INTE :

Un "0" sur cette ligne, n'autorise pas les interruptions d'arriver au micro-processeur.

. Entrée $\overline{CLK}$:

Cette entrée déclenche la bascule INT FF. Elle peut être connectée à l'une des horloges du MPU.

. Signaux $\overline{ELR}$,ETLG,ENGL :

Ces trois signaux permettent de placer les 8214 en cascade afin que le système puisse prendre en compte les 32 interruptions.

. Lignes $\overline{A0}$, $\overline{A1}$, $\overline{A2}$:

Ces sorties représentent le complément du niveau d'interruption courante (modulo 8). Grâce à ces signaux, le compteur peut pointer l'adresse du sous-programme spécifique.

. Sortie $\overline{INT}$:

Cette sortie génère une interruption au MPU. Pour ce faire, elle est reliée au STB du 8212.
(Schéma interne : voir annexe).

4 - Fonctionnement :

Quand une demande d'interruption est présentée au 8214, une interruption est envoyée au MPU via le 8212. Le 8214 encodera la demande à travers 3 bits et transmet cette valeur au 8212. Celle-ci va être stockée dans le CSR du 8214 (pour MPU) pour être utilisée comme référence de comparaison à toute autre demande d'interruption. Après la reconnaissance de l'interruption générée, le MPU pointe le compteur de programme à l'adresse du sous-programme d'interruption désiré.

../..

Quand un PICU est sollicité, sa sortie ENLG est à l'état bas (0 logique). Le premier 8214, recevant les interruptions les plus prioritaires, aura son entrée ETLG (Enable This Level Group) reliée à Vcc afin qu'il soit constamment validé. En outre, la sélection du 2ème PICU doit entrainer la désélection du premier. Pour ce faire, la sortie ENGL du premier 8214 sera reliée à l'entrée ETLG du deuxième (ENGL = "1" ⟹ ETLG = "1").

Les adresses des différents sous-programmes d'interruption sont obtenues par complémentation, d'après le tableau page suivante.

5 - Circuit de décodage :

Le circuit de décodage se compose de 8 comparateurs 7485, de portes NAND et d'inverseurs.

Les adresses que nous avons fixé par les comparateurs sont les suivantes :

83F0 pour le 1er 8214
83F1 pour le 2ème 8214
83F2 pour le 3ème 8214
83F4 pour le 4ème 8214
et 83F8 pour le 8212

On présente dans un tableau d'adressage, les lignes d'adresses des 4 encodeurs de priorité (8214) et du port d'entrée/sortie (8212).

On remarque d'après ce tableau que les lignes d'adresses A4 jusqu'à A15 restent inchangées. On utilisera donc 3 comparateurs pour sélectionner la zone 83F dans laquelle se trouve les 8214 et le 8212. Par contre, les lignes A0, A1, A2 et A3 changent. Celles-ci nous permettront donc de sélectionner l'un ou l'autre des 4 encodeurs ou le port d'entrée/sortie.

../..

ADRESSE DES DIFFERENTS SOUS-PROGRAMMES
D'INTERRUPTION

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Code Hexa	Adresses des sous-prog.	N° d'esclaves	
0	1	1	1	0	0	0	0	70	8F	01	1
0	1	1	1	0	0	0	1	71	8E	02	2
0	1	1	1	0	0	1	0	72	8D	03	3
0	1	1	1	0	0	1	1	73	8C	04	4
0	1	1	1	0	1	0	0	74	8B	05	5
0	1	1	1	0	1	0	1	75	8A	06	6
0	1	1	1	0	1	1	0	76	89	07	7
0	1	1	1	0	1	1	1	77	88	08	8
0	1	1	0	1	0	0	0	68	97	09	9
0	1	1	0	1	0	0	1	69	96	0A	10
0	1	1	0	1	0	1	0	6A	95	0B	11
0	1	1	0	1	0	1	1	6B	94	0C	12
0	1	1	0	1	1	0	0	6C	93	0D	13
0	1	1	0	1	1	0	1	6D	92	0E	14
0	1	1	0	1	1	1	0	6E	91	0F	15
0	1	1	0	1	1	1	1	6F	90	10	16
0	1	0	1	1	0	0	0	58	A7	11	17
0	1	0	1	1	0	0	1	59	A6	12	18
0	1	0	1	1	0	1	0	5A	A5	13	19
0	1	0	1	1	0	1	1	5B	A4	14	20

../..

Suite 1

D	D	D	D	D	D	D	D	Code Hexa	Adresses des sous-prog.	N° d'esclaves	
0	1	0	1	1	1	0	0	5C	A3	15	21
0	1	0	1	1	1	0	1	5D	A2	16	22
0	1	0	1	1	1	1	0	5E	A1	17	23
0	1	0	1	1	1	1	1	5F	AO	18	24
0	0	1	1	1	0	0	0	38	C7	19	25
0	0	1	1	1	0	0	1	39	C6	1A	26
0	0	1	1	1	0	1	0	3A	C5	1B	27
0	0	1	1	1	0	1	1	3B	C4	1C	28
0	0	1	1	1	1	0	0	3C	C3	1D	29
0	0	1	1	1	1	0	1	3D	C2	1E	30
0	0	1	1	1	1	1	0	3E	C1	1F	31
0	0	1	1	1	1	1	1	3F	CO	20	32

Tableau d'adressage :

A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0	Code Hexa
1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	83F0 ;1è 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1	83F1 ;2è 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0	83F2 ;3è 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	83F4 ;4è 8214
1	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	83F8 ; 8212

Le schéma du circuit de décodage est représenté (figure II - 3).

IV - MEMOIRE LOCALE -

Le maître dans sa version définitive, possèdera donc une mémoire locale composée de boitiers ROM ou RAM, d'une capacité pouvant varier suivant les besoins du système.

En ce qui nous concerne, cette mémoire occupe les adresses de 32 K à 64 K (8000-FFFF).

L'espace nous faisant défaut sur la carte, on a prévu de placer cette mémoire sur une carte à part, reliée au maître par un ruban fixé grâce à des connecteurs.

Pour la mise en oeuvre d'un système micro-processeur un support logiciel est nécessaire. Nous avons donc envisagé, pour tester notre carte, de remplacer la mémoire locale du maître par un système micro-ordinateur complet : l'EXOciser de Motorola. Ce système nous permettra

../..

CIRCUIT DE DECODAGE

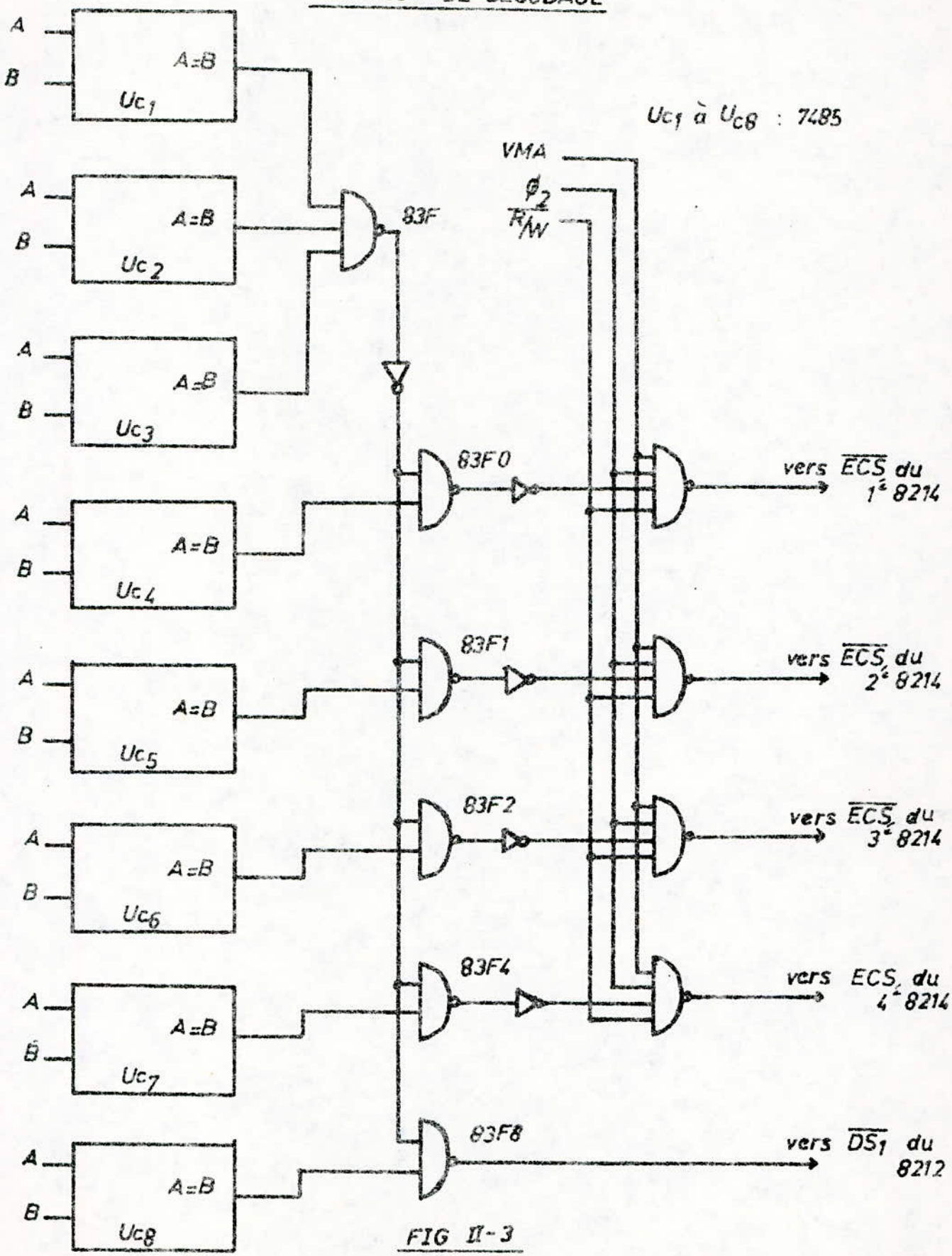


FIG II-3

grâce à sa mémoire et ses interfaces d'échanges, de bénéficier de la visu , de l'imprimante TTY et de la disquette pour inscrire ou sortir des informations. Avec ce procédé, il nous sera donc possible de tester et de mettre au point le dialogue maître-esclave et entre autre, d'effectuer les fonctions suivantes :

- chargement de mémoires
- inscriptions de programme sur visu ou TTY
- changement du contenu d'une mémoire
- opérer avec un PIA pour une interface d'échange (voir schéma figure II - 4)

Pour avoir plus de détails sur l'EXOciser de Motorola, on se reportera au fascicule :

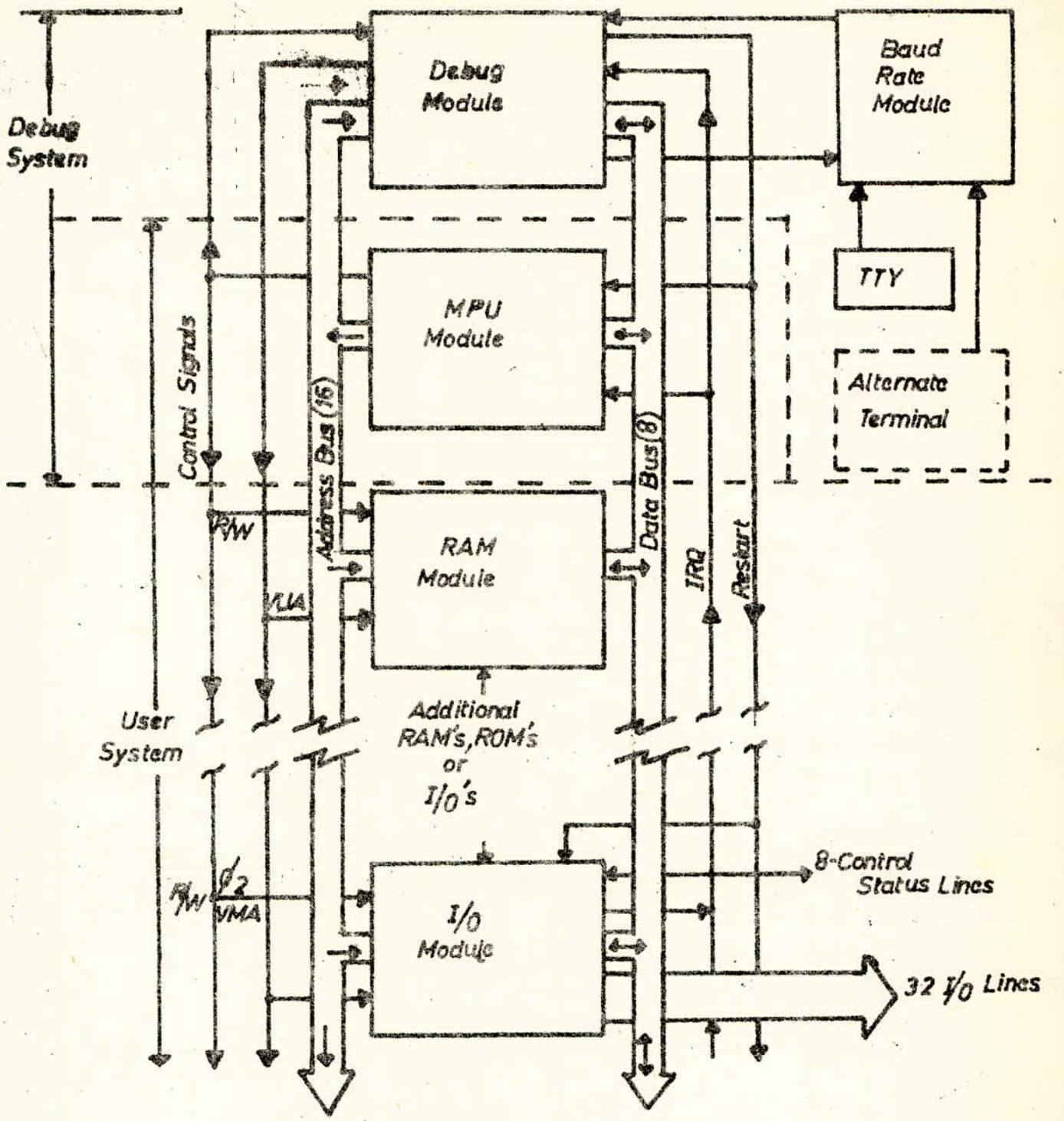
M 6800 Microprocessor - Applications Manual

V - BLOCAGE DU MAITRE :

Les différents processeurs, maître ou esclaves, utilisent évidemment le bus commun lors d'une distribution ou d'un transfert d'information. Lorsque l'un des esclaves l'utilise, le maître ne doit en aucun cas travailler en mémoire commune.

On a donc envisagé non seulement d'isoler le maître du bus commun par l'aiguillonnage des buffers comme vu précédemment, mais aussi de le bloquer par l'envoi d'un niveau bas sur l'entrée $\overline{\text{Halt}}$ de son MPU. Ce bloquage doit être maintenu, tant qu'un esclave travaille en mémoire commune. On réunit donc avec une porte NOR, le signal d'acquittement (ACK) et le signal d'appel (APP). La sortie de cette porte est reliée à l'entrée d'une bascule D, dont la sortie Q est appliquée à l'entrée $\overline{\text{Halt}}$ du MPU du maître.

../..



Typical EXORCISER System Block Diagram

FIG : II- 4

Sachant que la zone commune se trouve de 0000 à 7 FFF, la ligne A15 est à 0. On appliquera donc $\overline{A15}$ à l'entrée $\overline{S_D}$ de la bascule et Vcc (+ 5V) à l'entrée $\overline{C_D}$.

$\overline{S_D}$ et $\overline{C_D}$ étant les deux entrées de forçage de la bascule.

Ainsi, quand un esclave travaille en mémoire commune, $\overline{A15}$ est à 1. On retrouve en sortie ce qui se présente à l'entrée de la bascule, c'est-à-dire un niveau bas dans le cas d'un acquittement ou d'un appel. Le maître se trouve donc bloqué.

On notera que l'on ne peut pas avoir d'appel et d'acquittement en même temps.

La table de vérité et le schéma sont représentés à la figure II - 5.

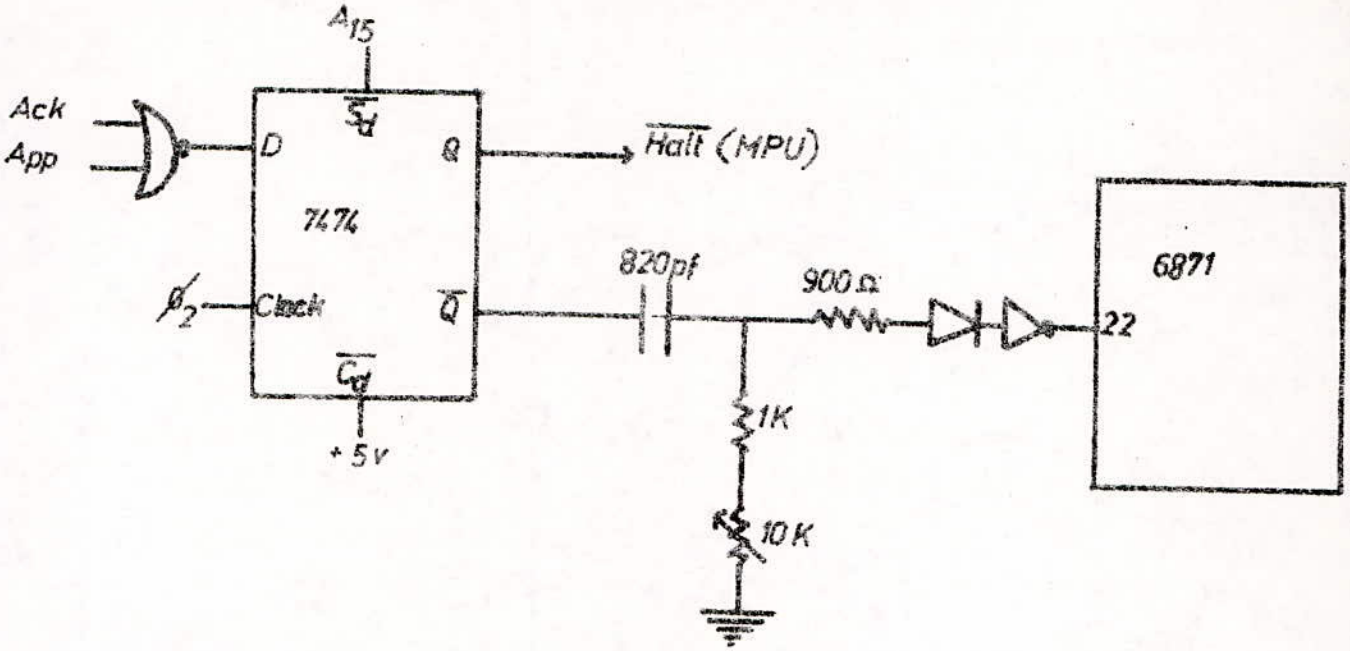
Cependant, un problème important apparaît lors de l'utilisation d'une entrée d'interruption ($\overline{Halt}$). En effet, la propriété d'anticipation du CPU fait que celui-ci termine l'instruction en cours, avant de prendre en compte l'interruption. De même, si celle-ci arrive après le début du dernier cycle d'une instruction, cette propriété fait que l'instruction suivante sera exécutée avant la prise en compte de l'interruption.

Pour y remédier, nous bloquons l'horloge dès l'apparition d'un appel ou d'un acquittement, quant A15 = 0 (c'est-à-dire quand un esclave veut travailler en mémoire commune). L'horloge bloquée, le MPU s'arrête instantanément. Mais l'horloge étant nécessaire au rafraichissement des registres internes du MPU, son temps de blocage ne doit pas excéder 4 micro-secondes.

On utilisera donc un circuit tampon, qui nous permettra d'envoyer une impulsion sur l'entrée "Mémoire prête" de l'horloge. L'entrée de ce circuit sera reliée

../..

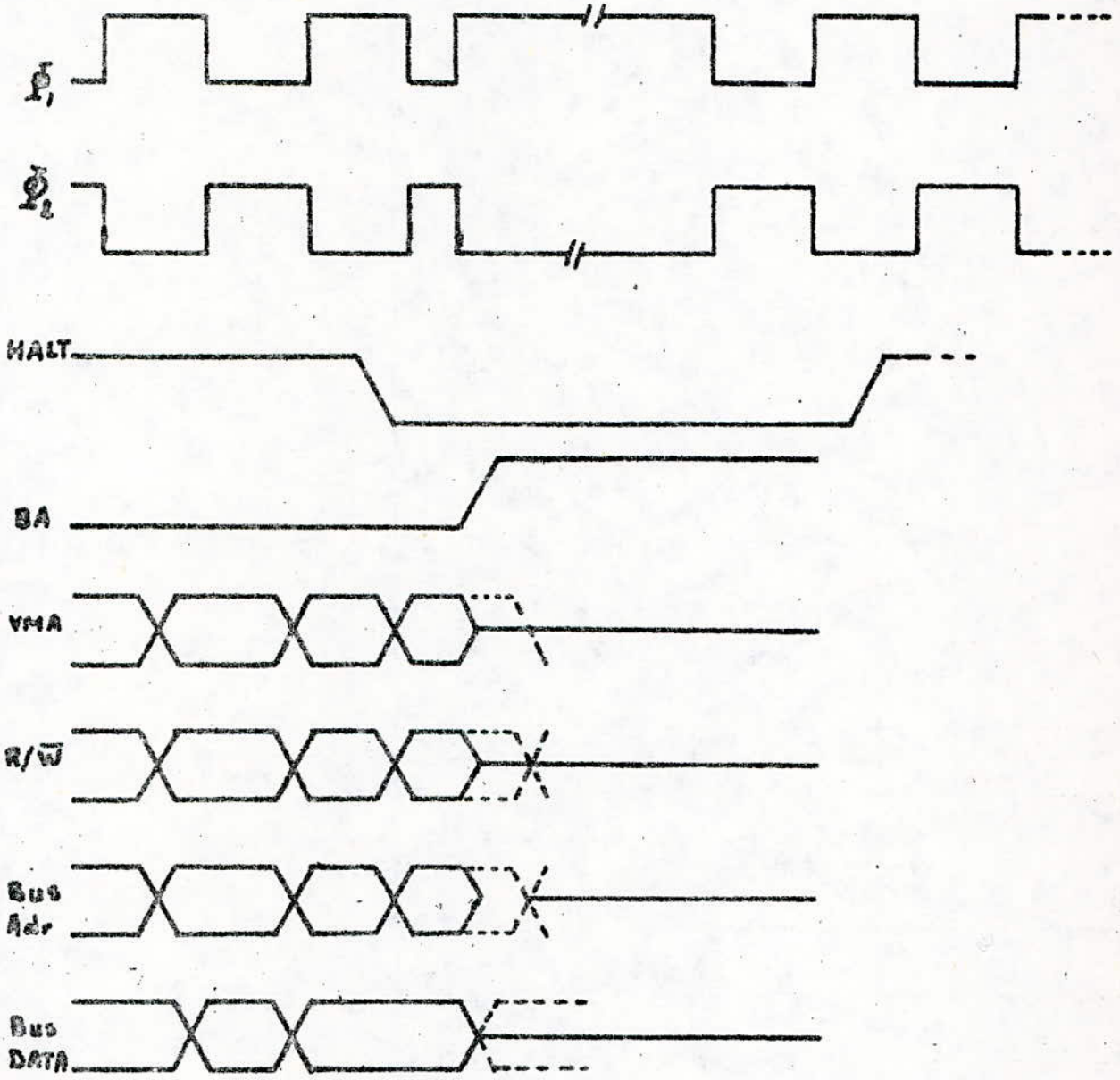
BLOCAGE DU MPU MAITRE ET DE L'HORLOGE



A_{15}	App	Ack	$\overline{App + Ack}$	$\overline{C_d}$	$\overline{S_d}$	Q	$\overline{Q}$	$\overline{Halt}$	Tampon	Commentaire
0	0	0	1	1	1	H	L	H	L	Libre Ack=0 App=0
0	0	1	0	1	1	L	H	L	H	bloqué $A_{15}=0$ Ack=1
0	1	0	0	1	1	L	H	L	H	bloqué $A_{15}=0$ App=1
0	1	1	0	1	1	L	H	L	H	IMPOSSIBLE
1	0	0	1	1	0	H	L	H	L	Libre $A_{15}=1$ M. Locale
1	0	1	0	1	0	H	L	H	L	//
1	1	0	0	1	0	H	L	H	L	//
1	1	1	0	1	0	H	L	H	L	//

FIG : II-5

CHRONOGRAMME DE BLOCAGE DE L'HORLOGE



à la sortie $\overline{Q}$ de la bascule utilisée précédemment. Le schéma est représenté en figure II - 5.

L'entrée asynchrone "Mémoire prête" de l'horloge, permet de bloquer les phases Ø1 à l'état bas et Ø2 à l'état haut pendant 2 micro-secondes. Ce temps dépend d'un réglage du circuit tampon.

En résumé, lors d'un appel ou d'un acquittement, on a simultanément un niveau bas sur l'entrée $\overline{Halt}$ du MPU maître, et un niveau haut sur l'entrée du circuit tampon. Ce niveau haut déclenche une impulsion qui bloque l'horloge, permettant ainsi au MPU de prendre en compte l'interruption $\overline{Halt}$ avant que celui-ci termine l'instruction en cours, ou en exécute une nouvelle.

VI - P. I. A (PERIPHERICAL INTERFACE ADAPTATOR) -

Pour envoyer les signaux d'appel (App) et d'acquittement (Ack) ainsi que l'adresse de l'esclave sollicité, on utilise un coupleur d'entrée/sortie (PIA) : le MC 6821 de Motorola.

Ce circuit,compatible TTL, a un fonctionnement programmable. Son rôle est d'effectuer la liaison entre le monde extérieur et la machine. Le PIA contrôlant le bus commun, sera situé en principe en mémoire commune. Dans notre cas, pour des raisons pratiques, nous l'avons placé sur la carte maître. Cette situation, purement géographique, ne modifie en rien son rôle ou son fonctionnement.

1 - Brochage du PIA :

Le circuit se présente sous forme d'un boitier DIL

../..

(Dual in line) à 40 broches. (Voir schéma de brochage en annexe).

La signification précise de chacune des broches est explicitée ci-dessous.

2 - Signaux d'échanges PIA - système (voir annexe)

- CS0, CS1, $\overline{CS2}$:

Ces lignes permettent la sélection du PIA quand CS0, CS1, $\overline{CS2}$ = 110, le PIA est sélectionné.

- RS0, RS1 :

Le PIA étant sélectionné, avec les 4 combinaisons de ces 2 bits, on adresse ses registres internes.

Le PIA occupe donc 4 adresses mémoires.

- E :

Cette entrée est généralement reliée à Ø2 (signal du bus de contrôle.)

- D0 à D7 :

Bus bidirectionnel de donnée. Il aboutit dans le PIA à un amplificateur pouvant être activé ou mis à l'état haute impédance par le signal $R/_W$ et ce si le PIA est sélectionné.

- $\overline{Reset}$:

Mis à 0, ce signal remet tous les registres internes du PIA à 0.

- $\overline{IRQA}$, $\overline{IRQB}$:

Ce sont deux lignes d'interruption permettant d'interrompre l'exécution d'un programme par le MPU.

3 - Signaux d'échanges (PIA-périphérie)

- PA_0 à PA_7 et PB_0 à PB_7 :

16 lignes de données programmables individuellement en entrée ou en sortie. Ces deux ports d'entrée/sortie nous donnent en sortie, le contenu des deux registres internes de 8 bits.

../..

- CA_1, CB_1 : Deux lignes d'entrées d'interruption
- CA_2, CB_2 : Deux lignes programmables en entrée d'interruption ou en sortie de commande

4 - Registres internes du PIA

Ces registres sont répartis en 2 groupes de 3 registres relatifs aux 2 ports A et B.

- CRA et CRB : Ces registres contiennent les paramètres de fonctionnement.
- DDRA et DDRB : contiennent le mot fixant le sens de transfert (entrée ou sortie) pour les lignes de données.
- ORA et ORB : Registres de sortie. Mémorisent les données en sortie lors d'une écriture. A la même adresse, on peut lire les données présentées en entrées mais elles devront être mémorisées à l'extérieur.
- Deux circuits de commande d'interruption A et B permettant de traiter CA_1, CA_2, CB_1, CB_2 et de générer $\overline{IRQA}$ et $\overline{IRQB}$.

5 - Adressage des registres

Par les deux fils d'adressage RS0 et RS1, on peut choisir parmi 4 registres ORA, ORB, DDRA et DDRB. CRA et CRB étant adressés directement (voir le tableau d'adressage page suivante)

../..

RS_1	RS_0	CRA_2	CRB_2	Registre adresse
0	1	-	-	CRA
0	0	0	-	DDRA
0	0	1	-	ORA et interface
1	1	-	-	CRB
1	0	-	0	DDRB
1	0	-	1	ORN et interface

6 - Sélection du PIA

De part ses registres internes, le PIA occupe 4 adresses mémoires. Ces adresses situées en mémoire commune sont les suivantes :

- OFF0
- OFF1
- OFF2
- OFF3

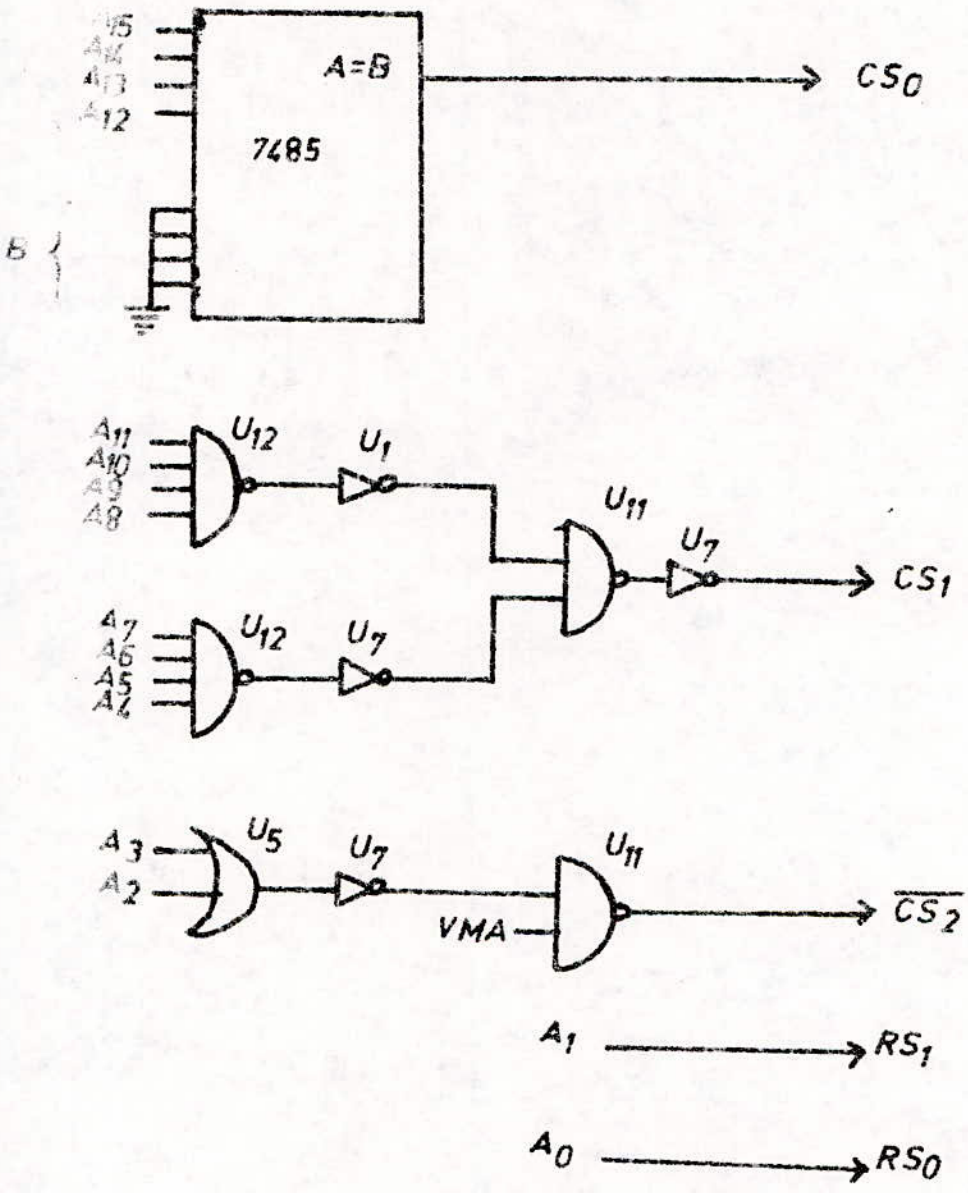
Le PIA est donc sélectionné par OFF. Les combinaisons des deux derniers bits permettent d'adresser les 4 registres internes. Le circuit est représenté en figure II - 6.

7 - Programmation du PIA

a) Principe de fonctionnement :

../..

SELECTION DU PIA



U_1 , U_7 : 7404
U_5 : 7432
U_{11} : 7400
U_{12} : 7420

FIG : II-6

Chacune des lignes des deux borniers A et B peut être individuellement programmée en entrée ou en sortie.

Quand un "O" est écrit dans le bit i du registre DDRA, la ligne i du port A (ou B) est programmée en entrée. Inversement, quand un "1" est écrit dans ce bit la ligne correspondante sera programmée en sortie.

Il est possible aussi, de mettre ces mêmes lignes (PA_0 - PA_7 et PB_0 - PB_7) à l'état haut ou l'état bas en programmant les registres ORA et ORB.

- Un O sur le bit i met la ligne i à l'état bas.
- Un 1 sur le bit i met la ligne i à l'état haut.

La programmation de DDRA, DDRB, ORA et ORB se fera par une soubroutine de chargement de l'accumulateur avec des valeurs répondant à l'utilisation désirée et transfert de celui-ci dans ces registres.

b) Mode de fonctionnement de CA_2

Le CA_2 peut être programmé soit en entrée, soit en sortie en mettant respectivement un "O" ou un "1" dans le bit 5 du CRA.

Si le CA_2 est programmé en sortie, dans ce cas CRA_3 et CRA_4 permettent de définir les modes d'action de CA_2.

Selon la programmation des bits CRA_4 et CRA_3, on distingue 4 modes de fonctionnement :

../..

CRA_4	CRA_3	Modes	
0	0	Dialogue impulsionnel	) associé à une lecture
0	1		
1	0	) Programmé	
1	1		

Dans le mode programmé, la sortie CA_2 suit la programmation du bit CRA_3 du registre CRA. Dans le cas du mode impulsionnel et dialogue, CA_2 est associé à une lecture.

C) Mode de fonctionnement de CB_2

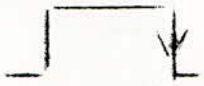
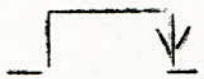
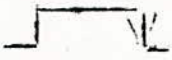
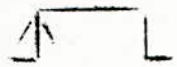
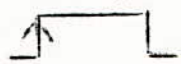
De même que pour CA_2, la programmation de cette ligne en sortie de commande s'effectue en écrivant un "1" dans CRB_5 . CRB_4 et CRB_3 permettent de définir les modes d'action de CB_2 comme le montre le tableau ci-dessous :

CRB_4	CRB_3	Modes	
0	0	Dialogue impulsionnel	) associé à une écriture
0	1		
1	0	) Programmé	
1	1		

../..

d) Mode de fonctionnement de CA_1

La programmation du mode de fonctionnement de la ligne d'interruption CA_1 est décrite dans le tableau ci-dessous.

N°	CRA_1	CRA_0	Transition active de l'entrée d'Int. CA_1	Indicateur d'interruption CRA_7	Sortie d'interruption $\overline{IRQA}$ (vers MPU)
1	0	0		mise à 1 par 	interruption $\overline{IRQA}$ = 1 masquée
2	0	1		mise à 1 par 	passe à 0 quand CRA_7 passe à 1
3	1	0		mise à 1 par 	interruption $\overline{IRQA}$ = 1 masquée
4	1	1		mise à 1 par 	passe à 0 quand CRA_7 passe à 1

2ÉME PARTIE - S O F T W A R E

I - INTRODUCTION -

Nous venons de voir dans la partie précédente toute la structure nécessaire au fonctionnement HARDWARE de la carte maître. Mais celle-ci ne représente qu'un "outil" dans la mise en oeuvre de notre système.

Pour cette mise en oeuvre, nous avons donc établi tout un logiciel permettant la gestion du dialogue Maître-esclave. Les différentes phases de ce dialogue seront développées en détail dans un chapitre suivant.

Avant d'en venir au programme proprement dit, il convient de préciser les différentes opérations que devra effectuer le maître afin de gérer convenablement le système.

II - ROLE DU MAITRE -

Les équations différentielles étant interdépendantes entre elles, le maître doit permettre aux esclaves de dialoguer entre eux à travers la mémoire commune.

Pour ce faire, il devra au préalable initialiser tout le système et transmettre les conditions initiales et les paramètres à calculer aux différents processeurs de calcul (esclaves).

De même, il est possible qu'un ou plusieurs esclaves, pour entreprendre leur calcul, aient besoin d'un

../..

résultat provenant d'un autre processeur. Le maître doit donc effectuer une procédure dite de "mise à jour" qui consiste à prévenir les esclaves intéressés par ce résultat.

Le maître doit être aussi en mesure de contrôler le pas du système, c'est-à-dire qu'il doit veiller à ce qu'aucun esclave n'entreprenne le pas i + 1 tant que les autres processeurs n'aient pas terminer le pas i.

En outre, après chaque pas de calcul, le maître doit tester une éventuelle sortie de résultat sur périphérique. Dans le cas positif, il se doit de l'effectuer.

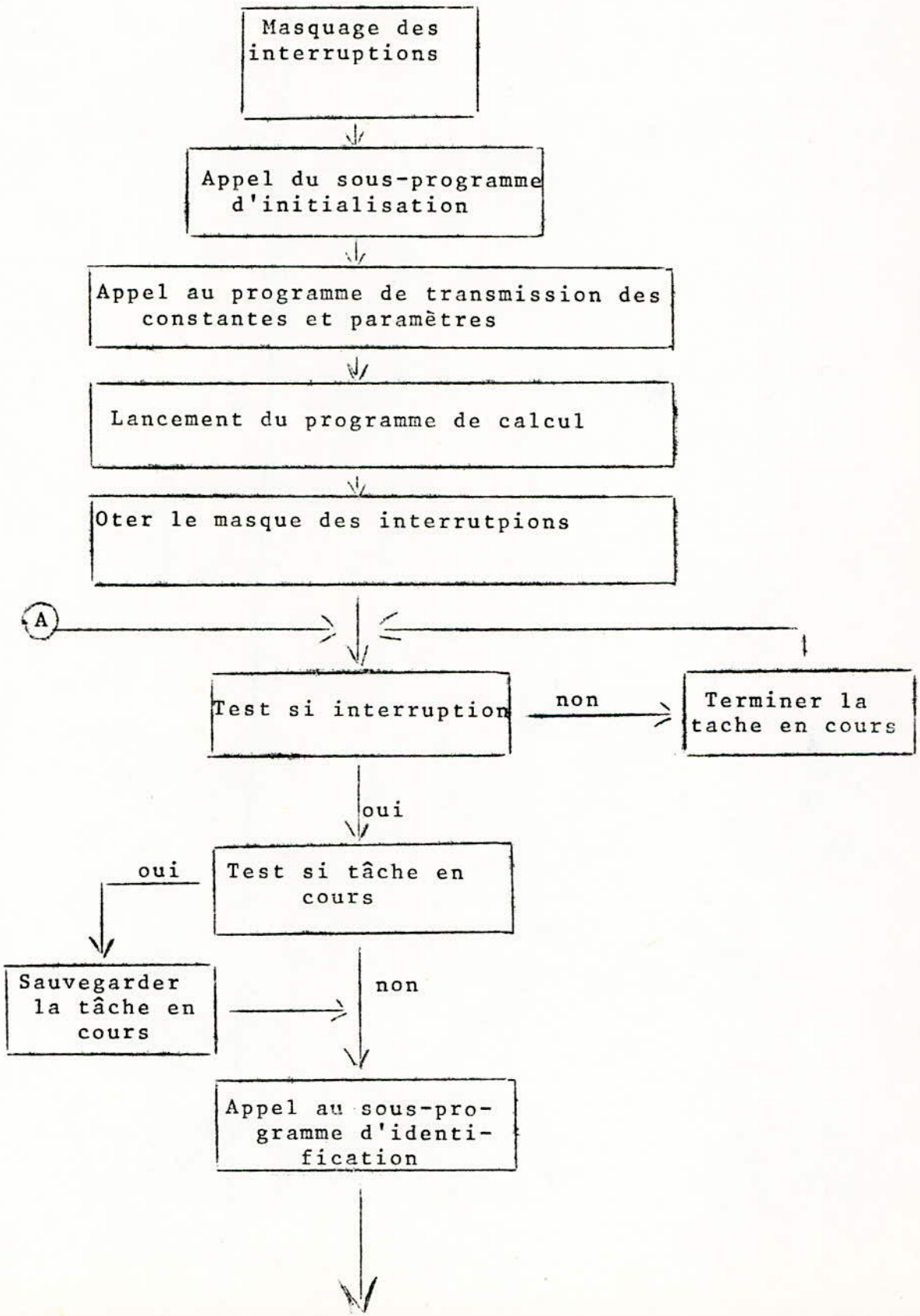
III - ORGANIGRAMME DU MONITEUR -

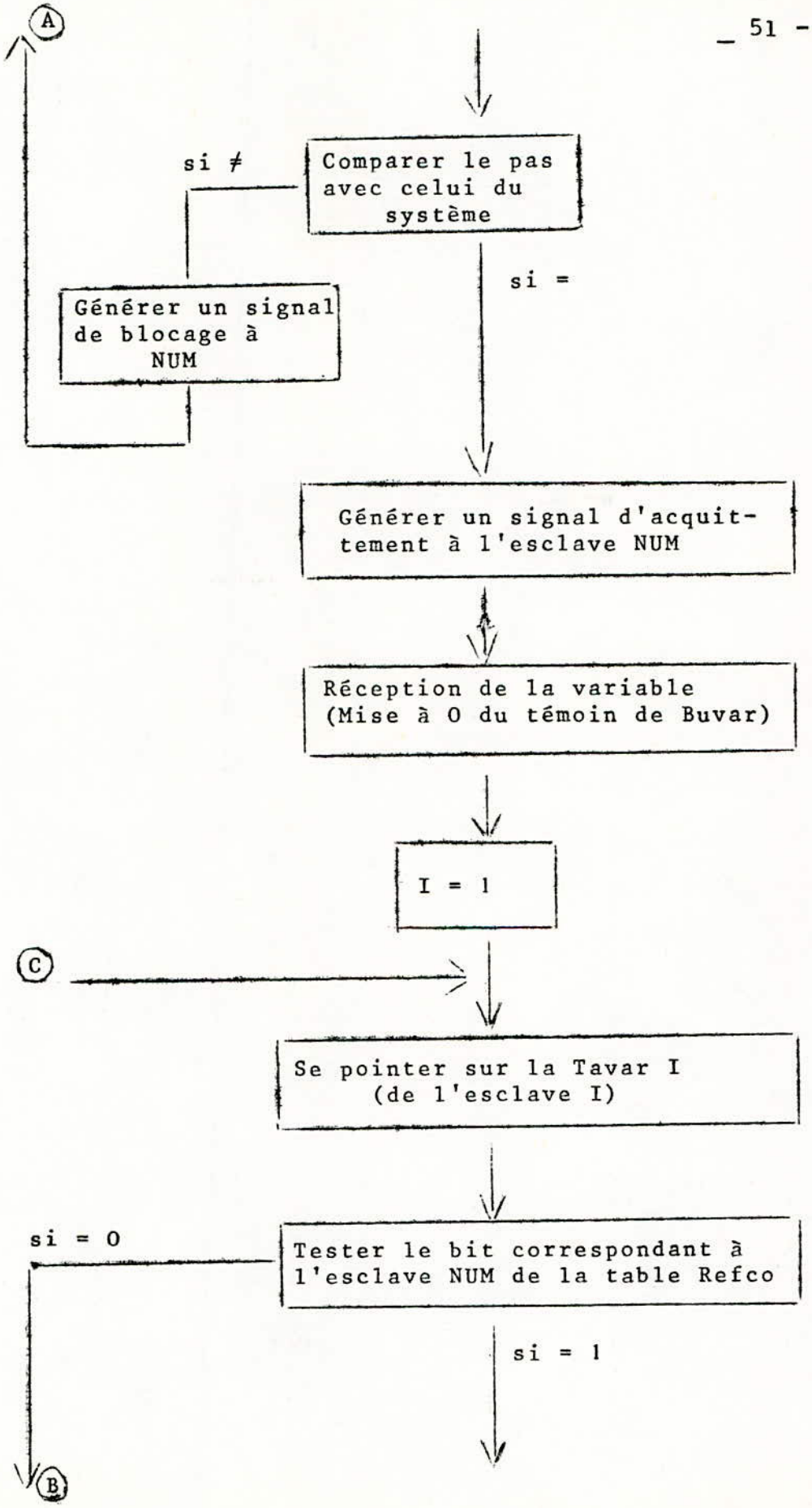
Le programme principal ou moniteur comprend plusieurs parties qui sont :

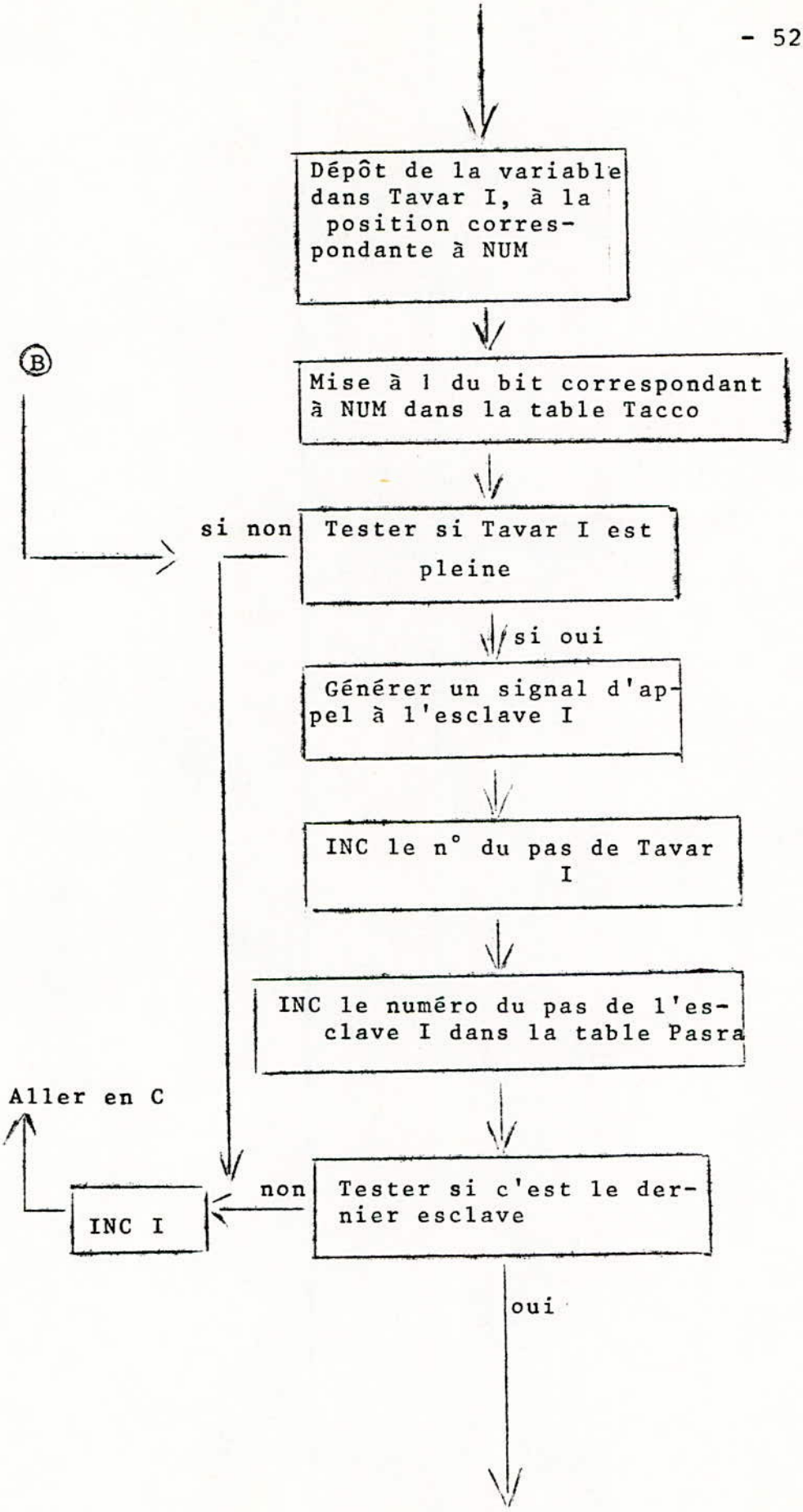
- sous-programme d'initialisation
- sous-programme de chargement des programmes de calcul et des conditions initiales
- distribution des tables
- sous-programme d'identification

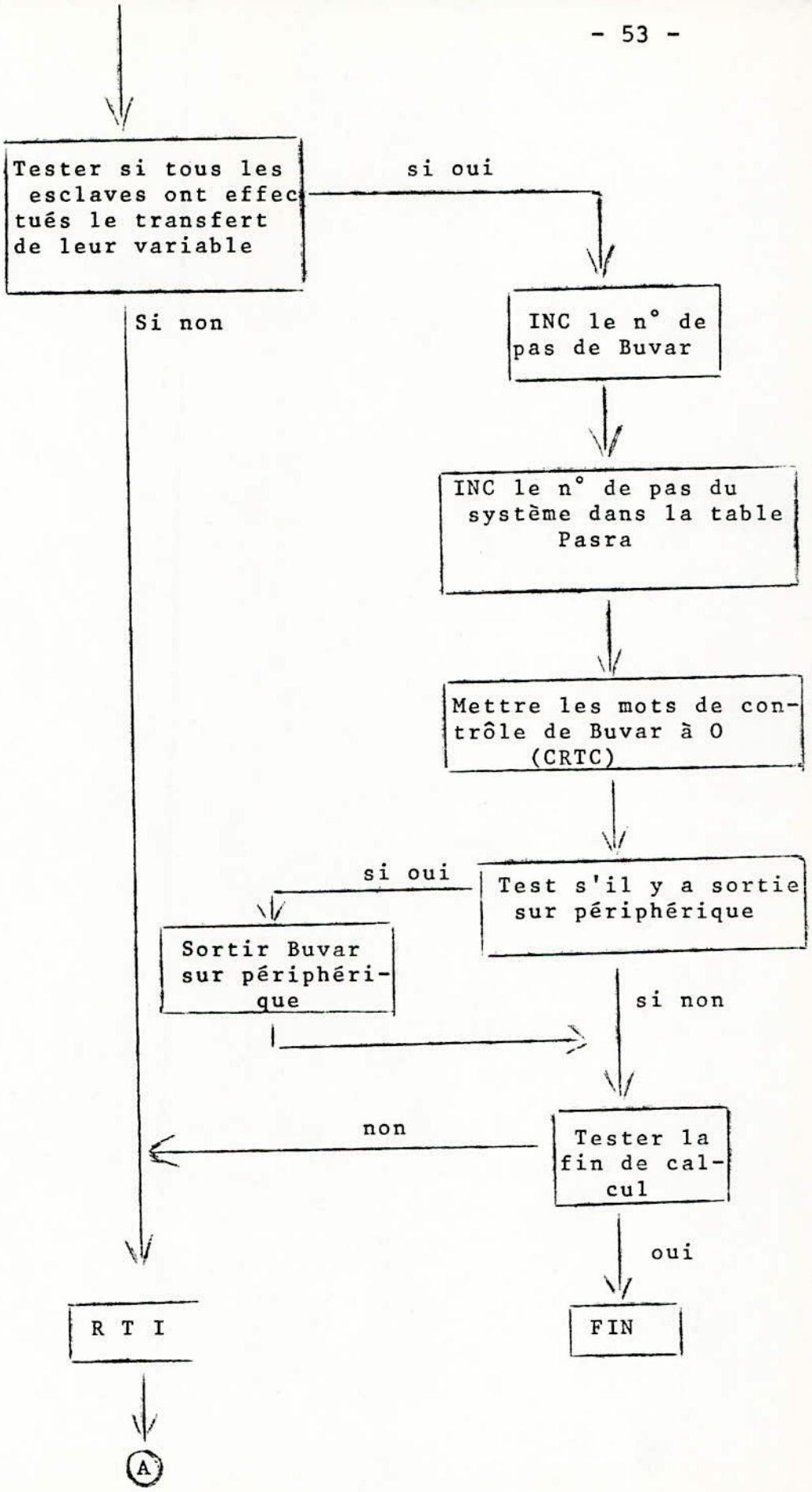
../..

1 - Programme général

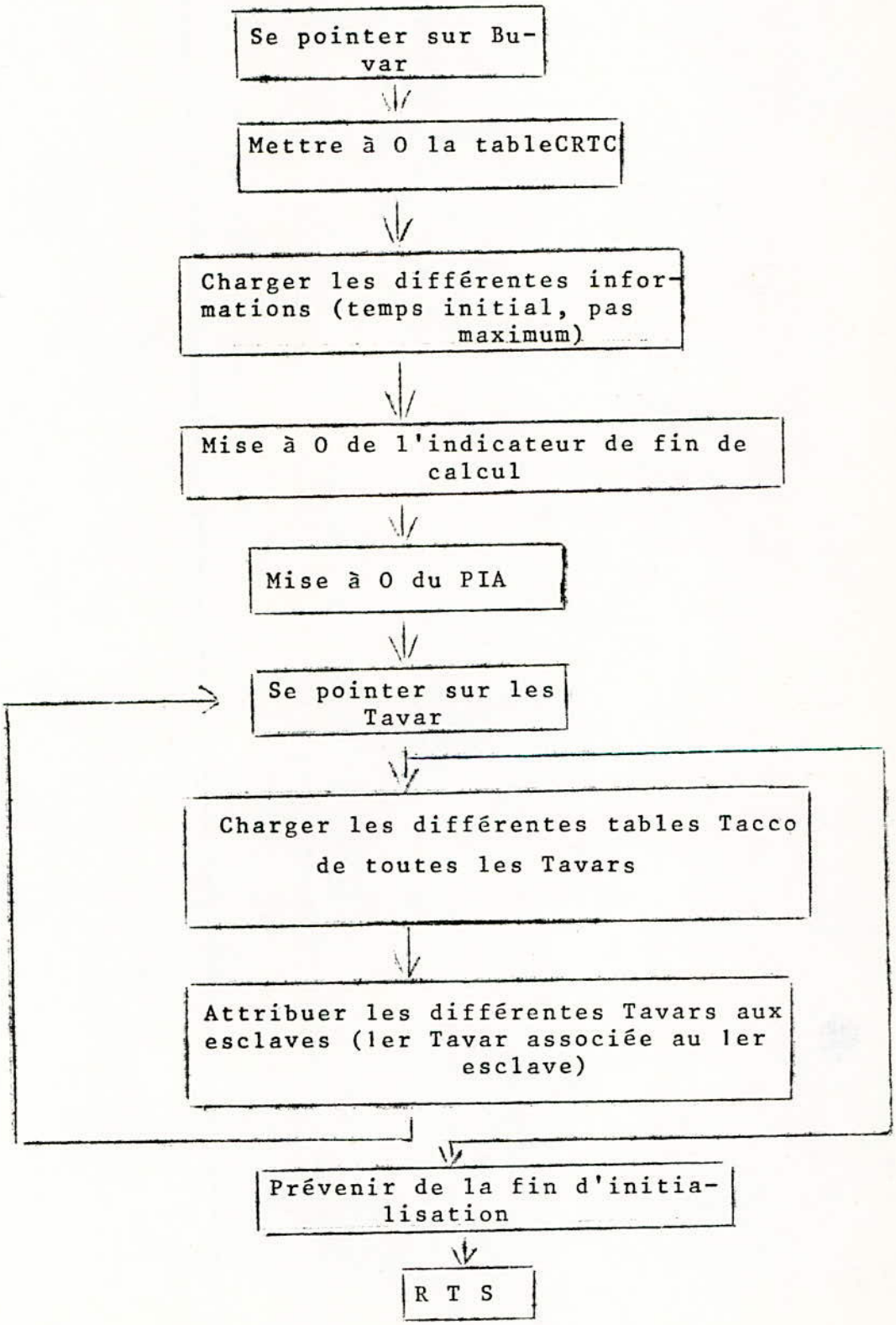



A
si ≠
Comparer le pas avec celui du système
Générer un signal de blocage à NUM
si =
Générer un signal d'acquittement à l'esclave NUM
Réception de la variable (Mise à 0 du témoin de Buvar)
I = 1
C
Se pointer sur la Tavar I (de l'esclave I)
si = 0
Tester le bit correspondant à l'esclave NUM de la table Refco
si = 1
B

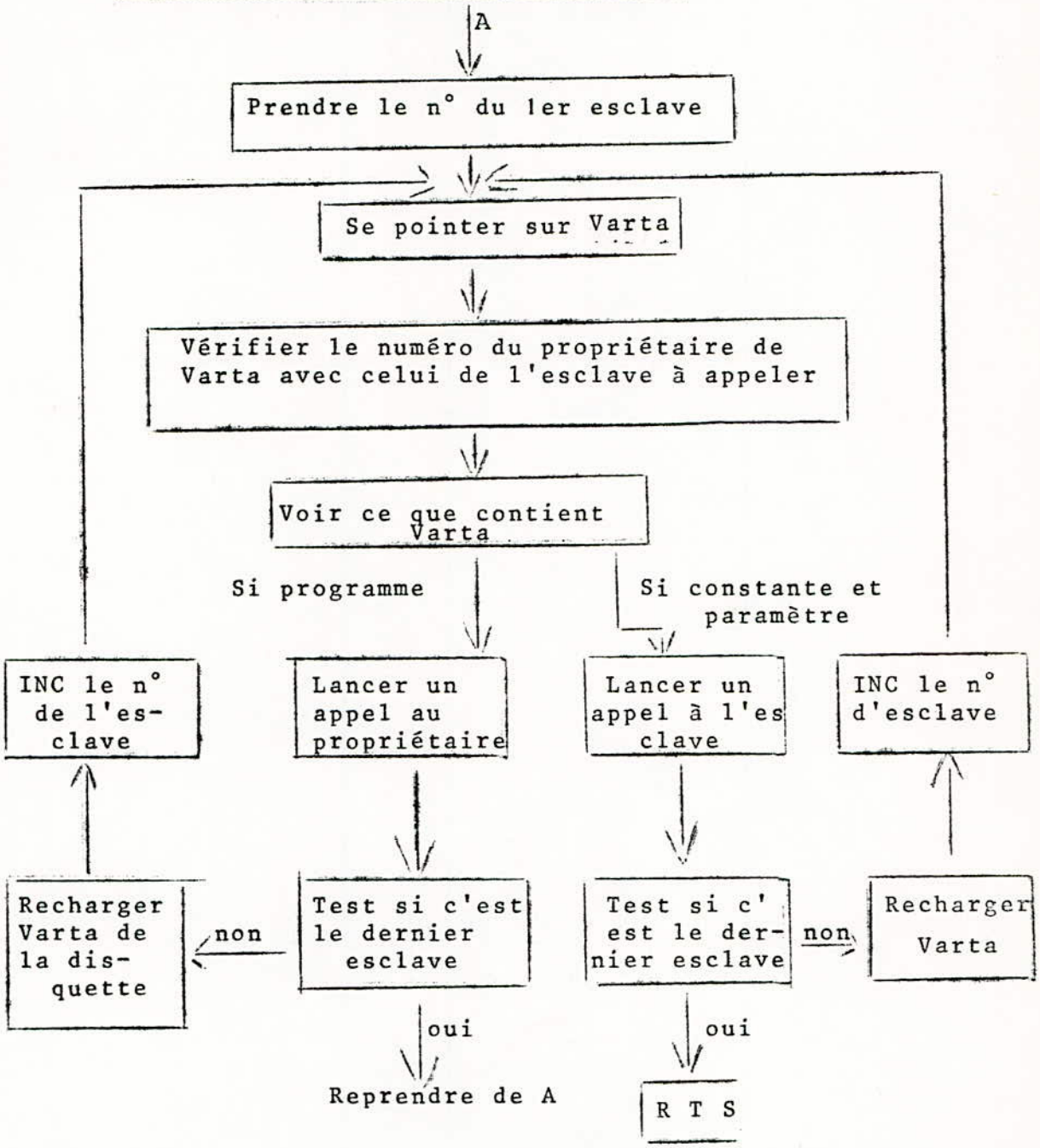

Dépôt de la variable dans Tavar I, à la position correspondante à NUM
Mise à 1 du bit correspondant à NUM dans la table Tacco
B
si non
Tester si Tavar I est pleine
si oui
Générer un signal d'appel à l'esclave I
INC le n° du pas de Tavar I
INC le numéro du pas de l'esclave I dans la table Pasra
Aller en C
non
Tester si c'est le dernier esclave
INC I
oui


Tester si tous les esclaves ont effectués le transfert de leur variable
si oui
Si non
INC le n° de pas de Buvar
INC le n° de pas du système dans la table Pasra
Mettre les mots de contrôle de Buvar à 0 (CRTC)
Test s'il y a sortie sur périphérique
si oui
Sortir Buvar sur périphérique
si non
Tester la fin de calcul
non
oui
R T I
FIN
A

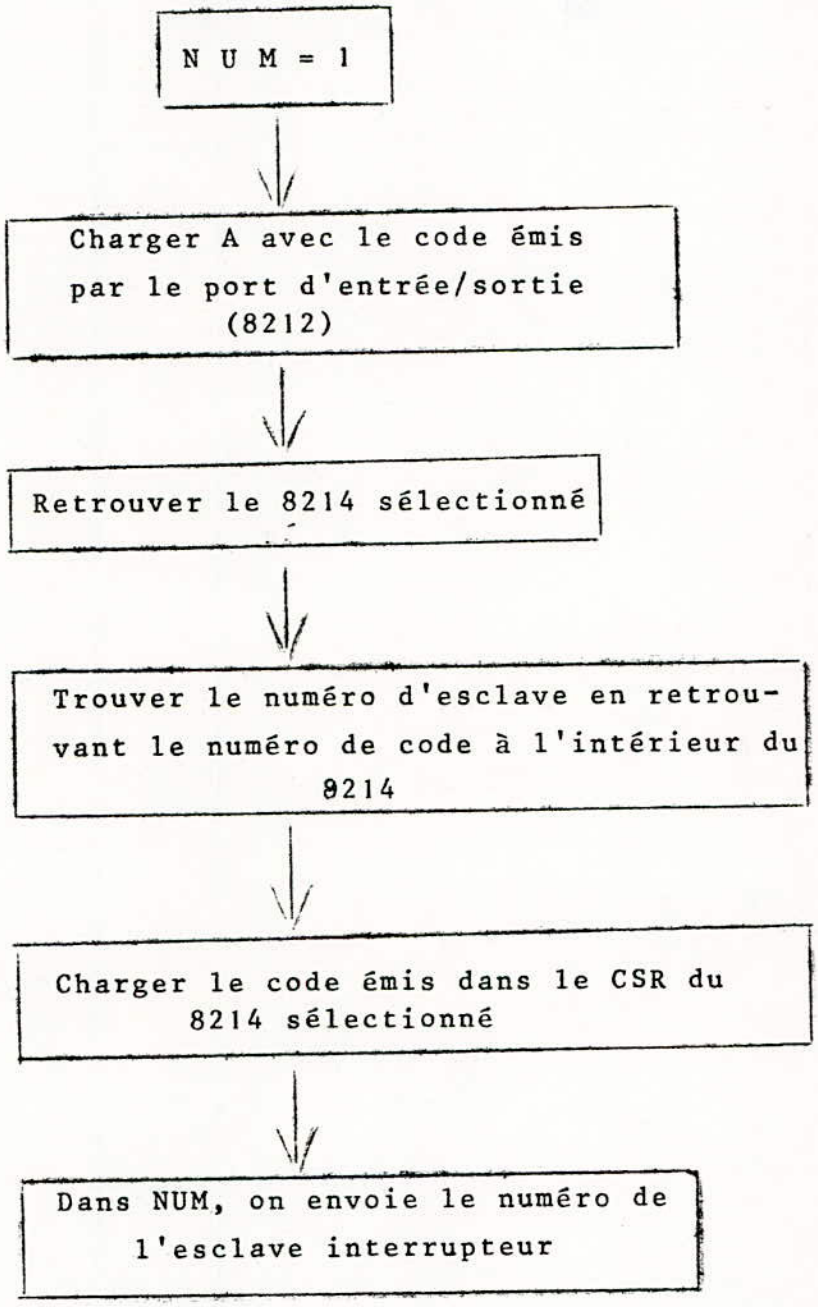
2 - Sous-programme d'initialisation



3 - Sous-programme de chargement des programmes de calcul et des conditions intitiales



4 - Sous-programme d'identification



CHAPITRE III

- CARTE ESCLAVE -

I - INTRODUCTION -

L'étude et la réalisation de la carte esclave ayant été effectuées lors d'un semestre précédent, nous nous bornerons à rappeler les principes généraux de la carte et à présenter les modifications que nous y avons apporté.

II - ROLE DE L'ESCLAVE -

L'esclave est un calculateur.Il a pour tâche la résolution d'une équation différentielles, qu'il résout avec la même méthode que les autres. Il ne commencera son calcul qu'après la réception des conditions initiales et l'ordre de départ du maître.

Lorsque l'esclave s'exécute, celui-ci est isolé. Il travaille indépendamment du système. En fait, il n'utilise le bus commun que lors d'un acquittement du maître, lui permettant de déposer un résultat, ou d'un appel, l'autorisant à venir chercher une information en mémoire commune.

../..

III - DESCRIPTION SOMMAIRE -

La carte esclave possède une unité centrale formée, comme le maître, d'un microprocesseur MC 6800, de buffers d'adresses et de données et d'une horloge. On lui adjoint aussi une logique d'aiguillonnage et de lecture-écriture.

Le reste de la carte est occupé par la mémoire locale de l'esclave, ainsi qu'un circuit de dsélection de la zone et d'un décodeur d'adresse.

IV - MODIFICATION DE LA CARTE -

Les modifications que nous avons apporté concernent les points suivants :

- extension de la mémoire
- autoblocage et commande des buffers de sortie
- adoption du connecteur intra-esclave

1 - Extension de la mémoire :

La place disponible sur la carte nous a permis d'augmenter la capacité mémoire à 10 K octets. Cette mémoire est formée d'1 k octets en 8 boitiers RAM MC 6810 et de 9 k octets en 9 boitiers ROM 2708.

Pour le décodage des adresses, nous avons utilisé un circuit intégré C MOS 14515, comportant 4 entrées (DATA INPUTS) et 16 sorties de sélection (SELECTED OUTPUT) Le brochage et la table de vérité de ce circuit se trouve en annexe.

../..

- Sélection des RAM -

La RAM MC 6810 est une mémoire de 128 octets. Elle possède plusieurs entrées de sélection :
CS_0, $\overline{CS_1}$, $\overline{CS_2}$, CS_3, $\overline{CS_4}$ et $\overline{CS_5}$.

Le tableau ci-dessous, montre comment s'effectuera la sélection de la zone mémoire ainsi que le décodage interne des RAM.

Boitier	ADRESSE	A 15	A 14	A 13	A 12	A 11	A 10	A 9	A 8	A 7	A 6	A 5	A 4	A 3	A 2	A 1	A 0
1	A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	A07F	1	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1
2	A080	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
	A0FF	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1
3	A100	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
	A17F	1	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1
4	A180	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0
	A1FF	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1
5	A200	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
	A27F	1	0	1	0	0	0	1	0	0	1	1	1	1	1	1	1
6	A280	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0
	A2FF	1	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1
7	A300	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
	A37F	1	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1
8	A380	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
	A3FF	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1

Sélection de la zone mémoire (A)

Décodage interne des RAM

../..

On remarque que les lignes A12 à A15 restent inchangées. Nous les utiliserons pour sélectionner la zone mémoire. Les lignes A7 à A11 seront utilisées pour la sélection de chaque boitier, comme le montre le tableau suivant :

Boitiers	CS_0	$\overline{CS_1}$	$\overline{CS_2}$	CS_3	$\overline{CS_4}$	$\overline{CS_5}$
1	$\overline{A_{11}}$	S	A_9	$\overline{A_{10}}$	A_8	A_7
2	$\overline{A_{11}}$	S	A_{10}	A_7	A_8	A_9
3	$\overline{A_{10}}$	S	A_{11}	A_8	A_9	A_7
4	A_8	S	A_{11}	A_7	A_9	A_{10}
5	$\overline{A_{10}}$	S	A_{11}	A_9	A_8	A_7
6	A_9	S	A_{11}	A_7	A_8	A_{10}
7	A_9	S	A_{10}	A_8	A_{11}	A_7
8	A_8	S	A_{10}	A_7	$\overline{A_9}$	A_{11}

S étant la sortie du circuit logique de sélection de la zone mémoire, représenté à la figure III - 1

Sélection des ROM -

Les MC 2708 sont en fait des EPROM, mémoires programmables effaçables. Elles possèdent une entrée de sélection ($\overline{CS}$/WE) pour l'extension de la mémoire. Cette sélection se fera grâce au décodeur 14515 dont on utilisera 9 sorties de sélection sur les 16 disponibles.

../..

2 - Autoblocage et commande des buffers de sortie :

Nous savons que les esclaves doivent être déconnectés du bus commun quand ils n'ont pas accès à la mémoire commune. Seuls les signaux d'acquittement et d'appel du maître leur permettent cette initiative. Quand un esclave envoie une interruption, A_{15} étant à 0 (mémoire commune), il se met en attente, car celui-ci n'est pas nécessairement prioritaire. Cette attente correspond à un blocage du MPU esclave. Pour ce blocage, nous utiliserons une bascule D. Le système fonctionne de la manière suivante :

- Dès que A_{15} passe à 0 (mémoire commune), un niveau bas est envoyé sur le $\overline{\text{Halt}}$ du MPU esclave. Celui-ci se bloque. On applique à l'entrée de forçage de la bascule (mise à 1), les signaux Ack et App. Quand l'un des deux passe à 1, le MPU se débloque automatiquement, permettant ainsi à l'esclave de travailler en mémoire commune.(voir le schéma, figure III - 2).

La table de vérité du système est la suivante :

ACK	APP	D / A15	$\overline{CD}$ / + 5V	$\overline{SP}$ / Ack.App	Q	$\overline{Q}$	
0	0	0	1	1	L	H	ESC bloqué
1	0	0	1	0	H	L	MPU ESC débloqué
0	1	0	1	0	H	L	"
0	0	1	1	1	H	L	"
1	0	1	1	0	H	L	"
0	1	1	1	0	H	L	"

../..

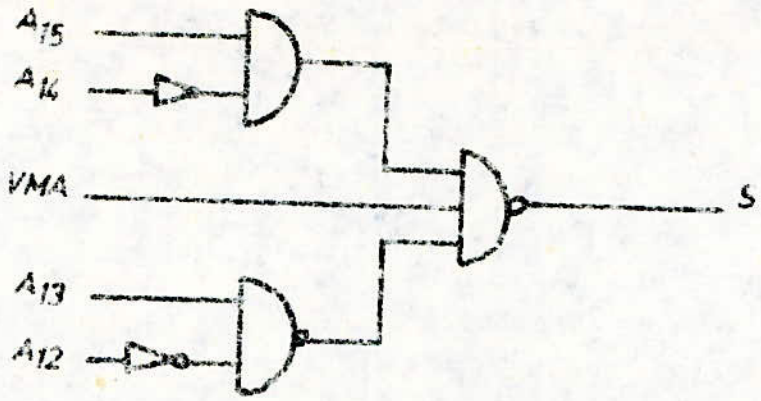


FIG : III - 1

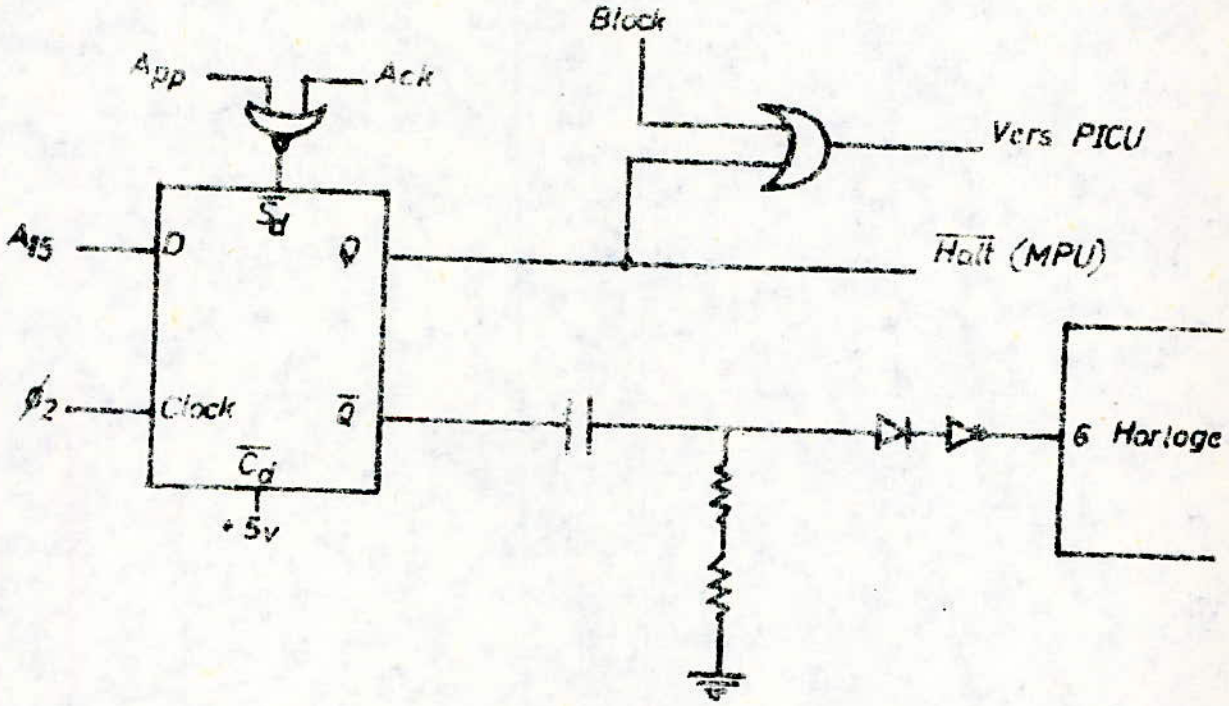


FIG : III - 2

La sortie Q de cette bascule est aussi connectée à une porte OU à deux entrées. Sur l'autre entrée arrive la ligne block, venant du PIA. Dès que l'esclave désire travailler en mémoire commune, A_{15} passe à O ainsi que la sortie Q. Si le PIA émet un niveau bas sur la ligne block, on a :

block + Q = O, une interruption se présente sur le PICU. On notera que cette interruption reste effective, tant qu'il n'y a pas d'acquittement (tant que Q reste à O). (Schéma figure III - 2).

Evidemment, l'accès sur le bus commun s'effectue à travers des buffers d'adresse et de donnée.

Commande des buffers d'adresse (8T95)

On applique A_{15} et $\overline{Q}$ de la bascule D, respectivement sur les bornes 1 et 15 des 8T95. Ainsi, pour un accès en mémoire commune A_{15} et $\overline{Q}$ sont à O. On est dans le mode passant.

On notera que $\overline{Q}$ est le plus souvent à l'état bas. Il ne passe à l'état haut que quand l'esclave est en attente (MPU bloqué). Donc seul A_{15} servira à la désactivation des 8T95. En mémoire locale A_{15} = 1, on a la configuration 1O . Les buffers sont désactivés.

Commande des buffers de données (8T26)

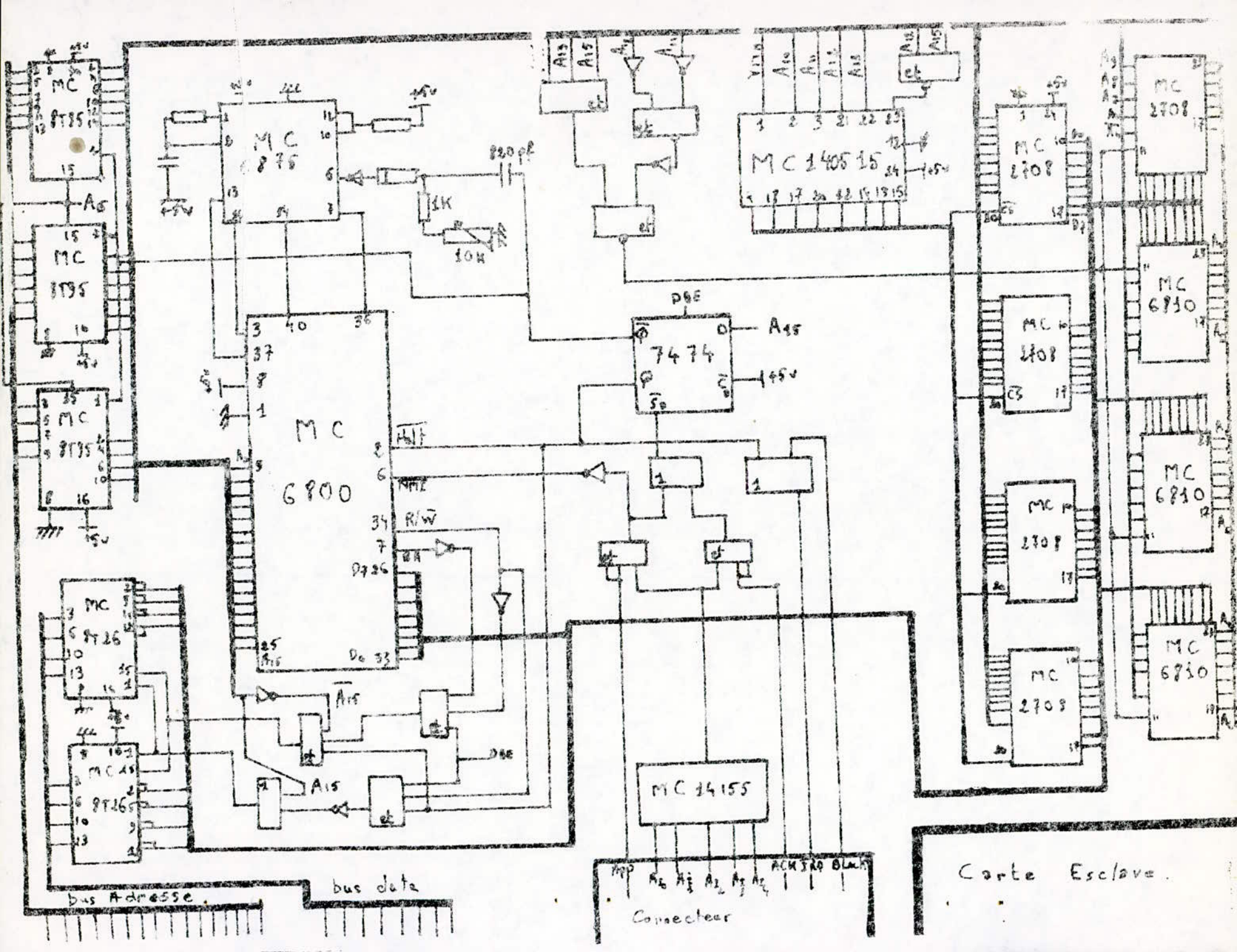
Pour les 8T26, la commande est identique à celle de la carte maître (voir chapitre II, figure II - 2).

../..

3 - Connecteur intra-esclave :

En plus du bus commun, le système comporte un bus reliant le PIA aux 32 esclaves. Ce connecteur intra-esclave est constitué de 8 lignes :

- lignes AI_0 à AI_4 : Ces 5 lignes permettent l'identification de l'esclave concerné.
- Ack : signal d'acquittement
- App : signal d'appel
- Pablo: ligne de blocage


MC 8T95
MC 8T26
MC 6875
MC 14405 15
MC 2708
MC 6810
MC 2709
MC 7474
MC 6800
MC 14155
1K
10K
820 pF
+5V
A15
R/W
HALT
NMI
DBE
bus Adresse
bus data
Connecteur
Carte Esclave

C H A P I T R E IV

- GESTION DU DIALOGUE -

I - INTRODUCTION -

Nous allons présenter dans ce chapitre, la manière dont s'effectue la gestion du dialogue entre les différents calculateurs.

Pour cela, on peut envisager plusieurs possibilités d'organisation :

- Dialogue par PIA
- Dialogue suivant le principe des boites aux lettres.

Le principe retenu sera le deuxième.

Lorsqu'un processeur calculateur a une variable à communiquer aux autres processeurs, il la dépose dans la mémoire commune (celle-ci jouant le rôle de boite postale). Le maître ayant connaissance des utilisateurs de ce résultat (ce qui n'est pas le cas des esclaves), va en faire la distribution dans les différentes tables associées aux esclaves.

Il nous faudra donc gérer toutes ces tables, dites tables de correspondance et mettre au point le moniteur

../..

qui en sera chargé.

Tout le travail a été basé sur les deux idées suivantes :

- les temps d'accès des processeurs doivent être minimalisés.
- l'accès en mémoire commune n'est pas interruptible.

II - AUTOBLOCAGE DU MAITRE ET DE L'ESCLAVE -

Comme nous l'avons dit précédemment, le bus commun est partagé par tous les processeurs du système.

Tous les esclaves étant déconnectés du bus, quand l'un d'eux veut accéder à la mémoire commune, il ne se branche sur le bus, qu'après l'acquittement du maître. Sans cette autorisation, l'esclave est en Halt (MPU bloqué). La réception d'un appel (App) branche automatiquement l'esclave sur le bus commun. Ce signal ainsi que l'acquittement reste à l'état haut. Le calculateur remettra lui-même ces signaux à l'état bas quand il aura terminé son transfert.

Ainsi, il s'interdira l'accès en même temps qu'il libèrera le bus.

Pour le maître, le problème se pose en des termes différents. Le maître peut accéder à la mémoire commune à n'importe quel moment à la condition, bien entendu, qu'il n'ait donné auparavant ni appel, ni acquittement. L'occupation du bus par un esclave est caractérisé par l'état haut d'une des deux lignes Ack ou App. On les utilisera donc pour interdire au maître l'accès en mémoire commune (voir chapitre II "Blocage du Maître")

../..

Le maître n'est mis en Halt (bloqué) que si les deux conditions suivantes se trouvent réunies en même temps :
- le bus est utilisé par un calculateur
- le maître veut accéder à la mémoire commune.

Si l'une des deux conditions n'est pas remplie, le maître n'est pas bloqué. Par contre, si seulement la première se trouve vérifiée, le maître est déconnecté du bus, mais peut travailler dans sa mémoire locale. S'il désire accéder en mémoire commune, la deuxième condition sera alors vérifiée. Le maître est donc automatiquement bloqué.

III - DIALOGUE -

Dans la structure choisie, nous avons en fait deux types de dialogues :

a) le dialogue maître-esclave

b) le dialogue esclave-esclave.

a) Dialogue maître-esclave :

Le système possède un connecteur direct, reliant tous les esclaves, le maître et le PIA.

Le rôle de cette liaison est très important. C'est grâce à ce connecteur que sont données toutes les autorisations d'accès. Le maître peut donc adresser n'importe quel esclave, directement.

Le connecteur contient :
- Toutes les lignes d'interruptions ($\overline{IRQ}$), allant des esclaves vers le PICU, chaque ligne d'$\overline{IRQ}$ étant spécifique à un esclave.

../..

- La ligne d'appel (App) permettant à un esclave de venir chercher une information en mémoire commune
- La ligne d'acquittement permettant à l'esclave de déposer un résultat
- Les lignes de sélection de l'esclave
- La ligne de blocage

Pour générer ces différents signaux, on utilise un PIA.

Quand le maître veut lancer un acquittement à un calculateur, il programme le PIA de telle sorte que celui-ci génère l'adresse de l'esclave concerné et le signal (Ack) à transmettre.

A la réception, chaque esclave dispose d'un décodeur pouvant reconnaitre l'adresse émise. Si c'est le cas, le signal d'acquittement est pris en compte. Dans le cas contraire, il est ignoré.

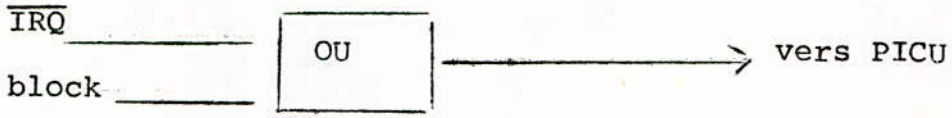
A la fin d'accès, l'esclave remettra lui-même ces lignes (Ack ou App) à leur état initial (état bas). Celles-ci sont donc les témoins du bus commun. Leur état (1 ou 0) permettra au maître de savoir si le bus est occupé ou non. Le rôle particulier de la ligne de blocage (block) sera développé un peu plus loin.

Déroulement des opérations :

Nous allons voir le déroulement des opérations successives. Supposant que l'esclave i vient de terminer son calcul, il doit donc déposer son résultat dans la mémoire commune. Pour ce faire, il envoie une demande d'accès ($\overline{IRQ}$) à l'encodeur de priorité. Celui-ci compare les priorités, permet le passage de la demande la plus prioritaire.

../..

La demande $\overline{IRQ}_i$ arrive donc au maître. Celui-ci se branche au sous-programme relatif à l'interruption i. Le signal d'acquittement est envoyé que si le pas de l'esclave i correspond à celui du système. Si ce n'est pas le cas, le maître génère un block, signal qui désactive la demande d'interruption au niveau même de l'esclave, grâce au circuit suivant :



Ce signal de blocage est fait de telle manière que la demande ne soit plus transmise à l'encodeur, ne génant pas celle d'un autre esclave moins prioritaire. Le signal block retombe à 0 quand le maître génère à nouveau un signal à travers le PIA. La demande de l'esclave i peut à nouveau atteindre le PICU.

Dans le cas où le pas est le même, le maître lance un acquittement permettant à l'esclave d'effectuer son transfert. Celui-ci terminé, le calculateur remet à zéro le PIA libérant ainsi le bus commun.

b) Dialogue esclave-esclave :

La mémoire commune représente l'élément de dialogue esclave-esclave.

Chaque esclave possède dans cette mémoire, une table qui est en quelque sorte sa "boite aux lettres" :

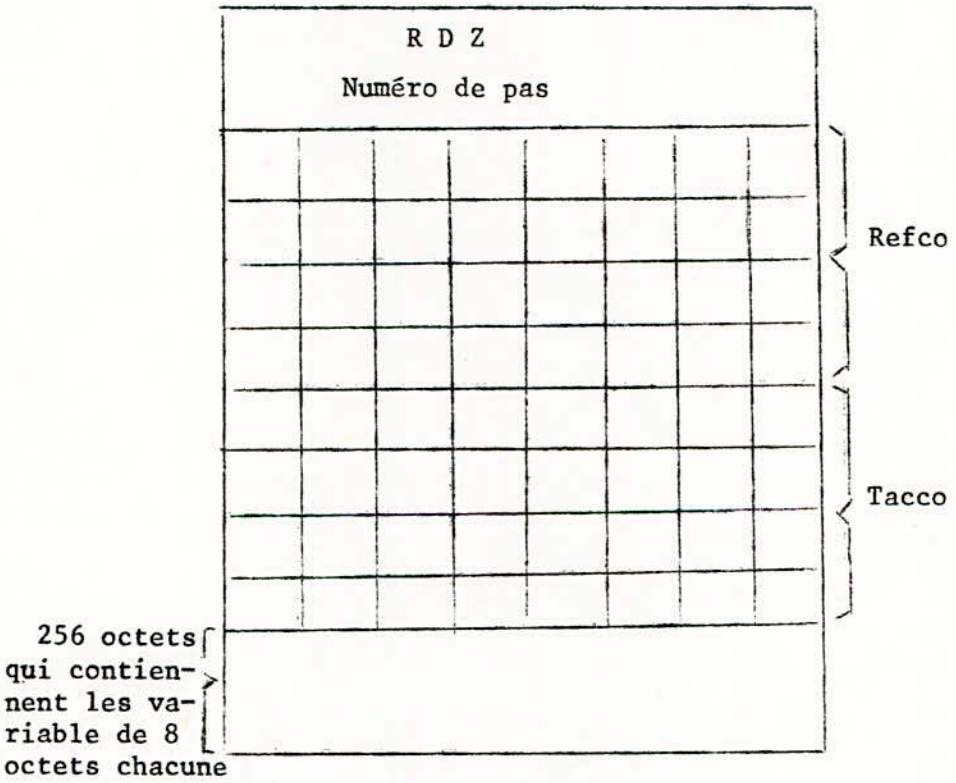
../..

c'est la table Tavar.

La Tavar contient la table des correspondants de cet esclave (Refco) et la table Tacco servant de contrôle de la Tavar.

On peut donc se représenter la mémoire commune comme une boite postale. Le moniteur, jouant le rôle de "facteur" distribue les variables dans les boites aux lettres associées à chaque esclave.

Table_Tavar -



R D Z : octet d'identification de la table Tavar. Il contiendra le numéro de l'esclave propriétaire de la table.

../..

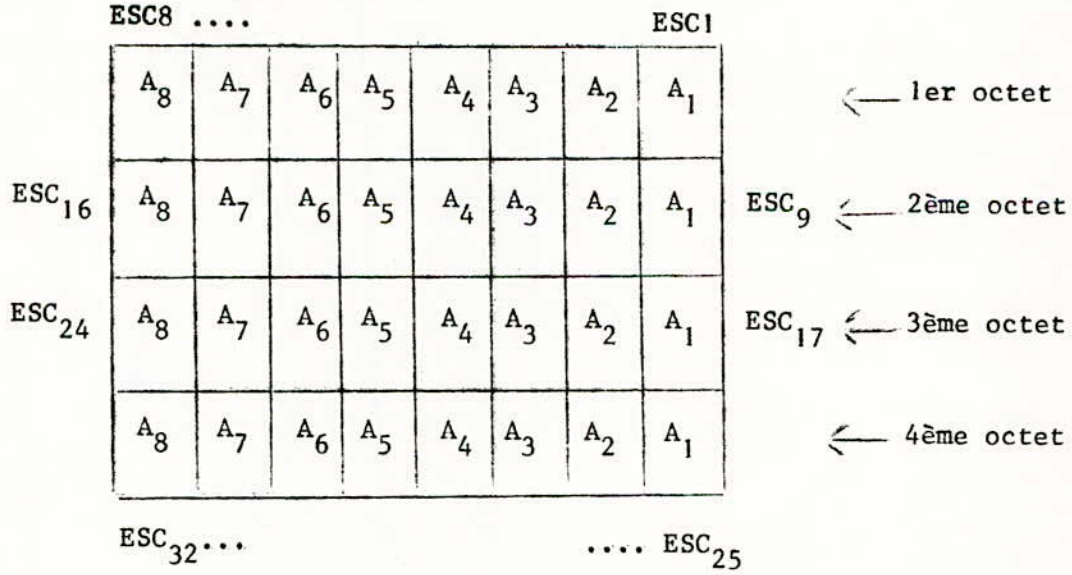
b.1 La table Refco

La table Refco, forte de 4 octets sert de référence au maître, en ce qui concerne les correspondants du propriétaire de la table. Elle contient 32 bits, chaque bit étant relatif à un esclave.

Par exemple, si le bit n° j de la table, appartenant à l'esclave i, est à 1, l'esclave i a donc besoin du résultat calculé par le processeur j.

Refco est donc la table des correspondants du propriétaire de la Tavar. Elle est remplie avant le début de la simulation et reste valable durant toute la simulation.

Structure de la table Refco :



../..

b.2 La table Tacco :

La table Tacco est une table de contrôle. Chaque bit est associé à chaque variable distribuée et déposée par le moniteur dans la table Tavar.

Tacco est initialisée à zéro avant le lancement de la simulation.

Dès qu'une variable est déposée dans la table, le bit correspondant est mis à 1. En comparant cette table à la table Refco, le moniteur saura quand la Tavar sera pleine (c'est-à-dire que toutes les variables attendues auront été distribuées). Le maître lancera alors un appel au propriétaire pour que celui-ci effectue le transfert vers sa mémoire locale.

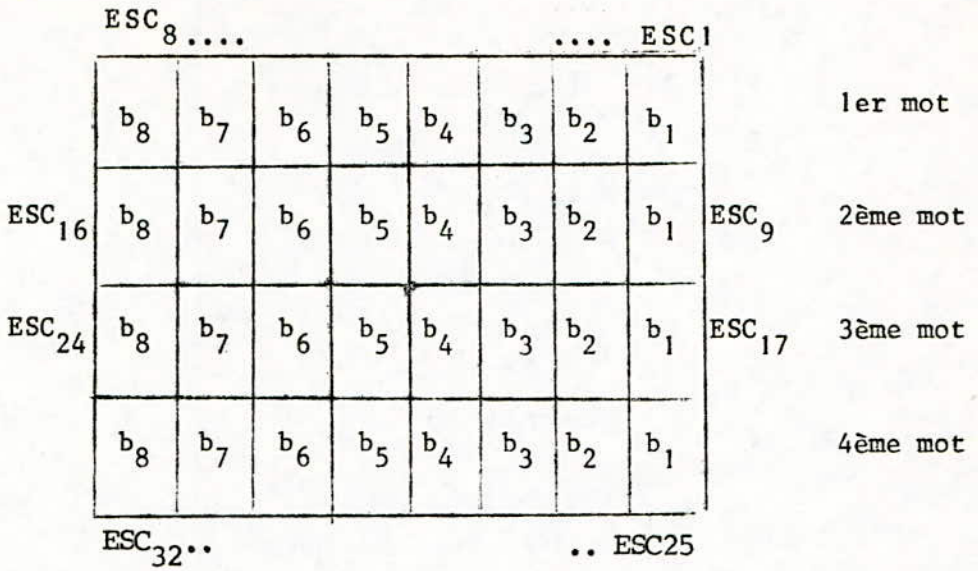
Exemple :

Le bit n° 4 du deuxième mot (octet) correspond à la variable n° 12. Dès que le moniteur dépose cette variable, (en supposant que celle-ci est attendue), il mettra le bit 4 à 1 et comparera alors les tables Tacco et Refco. Si elles sont identiques, un appel est lancé au propriétaire (de la Tavar). Si par contre, elles sont différentes, le moniteur continuera la distribution.

Structure de la table Tacco : (voir page suivante)

Quand on dépose la variable 23, le bit b7 du 3ème octet (mot) est mis à 1. Il était initialement à 0.

../..



b.3 La_table_Buvar :

Une autre table importante est celle de réception des variables. C'est dans cette table que sont déposés les résultats calculés par les processeurs calculateurs, ceux-ci n'ayant accès qu'à leur propre table de variable Tavar.

Quand l'esclave reçoit une autorisation d'accès (Ack), il dépose sa variable dans une position définie (dont il a l'adresse) et retourne dans sa mémoire locale, libérant le bus commun. Le moniteur peut donc effectuer la distribution de la variable et les mises à jour des différentes tables.

La table Buvar dispose d'une table de contrôle (CRTP) qui donne le nombre de variables arrivées.

Dès qu'un esclave dépose son résultat, il mettra l'octet témoin de Buvar à OO. Ce témoin sera remis à FF

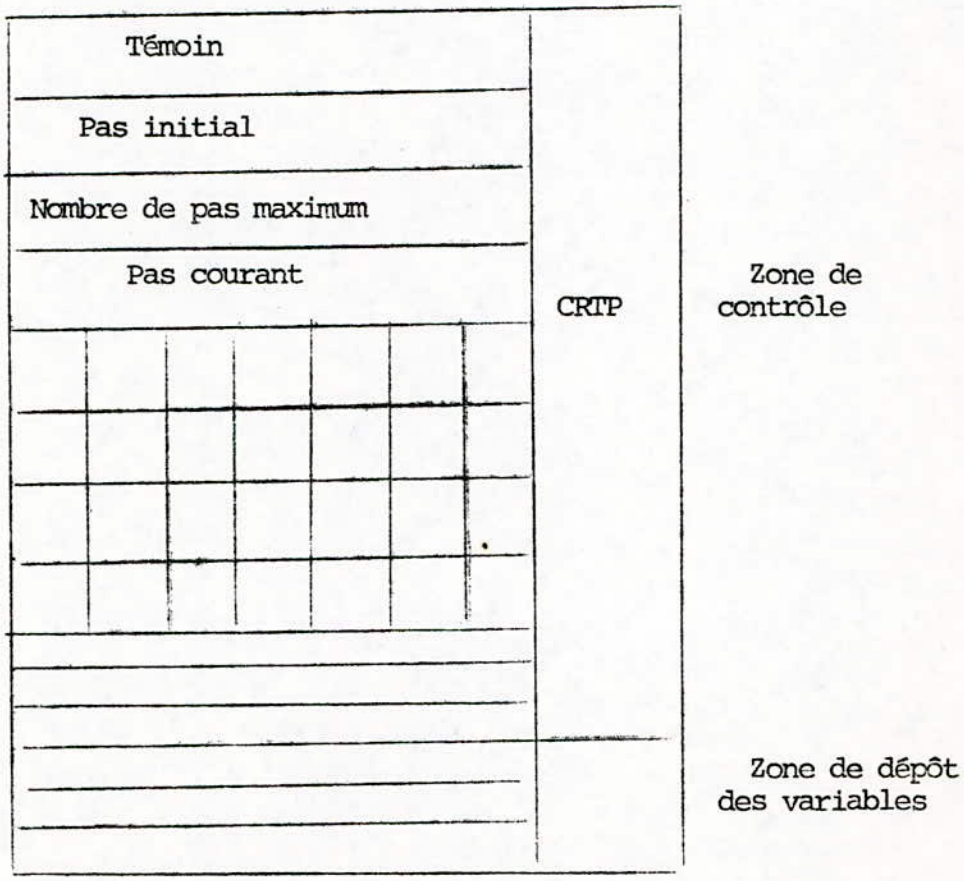
../..

par le moniteur lors de la mise à jour qui suit chaque départ. Ainsi le prochain esclave autorisé à accéder en mémoire commune trouvera le témoin à FF et le remettra à OO.

Dès que le pas du système change, la table de contrôle de la table Buvar (CRTP) est remise à O, ainsi un nouveau cycle de calcul recommence.

La table Buvar étant aussi utilisée pour les entrées/sorties, on devra y reporter plusieurs renseignements sur l'état de la simulation en cours : le pas de calcul, le temps initial, la fin de calcul, le nombre de pas.

Structure de la table Buvar :



. ./..

Après chaque dépôt, le moniteur mettra un 1 dans le bit de la table CRTP, correspondant à l'esclave qui vient de déposer sa variable.

Cette table sert de contrôle des dépôts effectués. Quand elle est égale à FF (quand tous les mots de la table sont égaux à FF), le moniteur saura que pour le même pas, toutes les variables ont été calculées. Il pourra donc effectuer la sortie des variables sur les différents périphériques (Imprimante, visu, table traçante..) En outre, cette table nous permet de choisir la longueur du pas du système, suivant deux possibilités :

1) le pas correspondra au temps que met l'esclave le moins prioritaire à calculer sa variable et à la déposer dans Buvar.

2) le système sera à temps réel, c'est-à-dire que le pas du système est fixé extérieurement (on lui impose une cadence de travail)

Dès que le pas du système change, le moniteur remettra le CRTP à zéro (tous les mots sont remis à 00).
Le système dispose d'une autre table située en mémoire locale du maître, contenant tous les pas des esclaves et du système. Elle nous servira à connaitre l'état du système. Ce sera la table Pasra.

Une autre table, Task Table, contiendra toutes les tâches exécutables par le moniteur. C'est la table des tâches sur laquelle on reviendra un peu plus loin.

IV - TRANSMISSION DES PROGRAMMES DE TRAVAIL ET DES CONSTANTES ET PARAMETRES -

Avant de commencer la simulation, certaines opérations sont nécessaires :

../..

- transmission des programmes d'initialisation
- transfert des programmes de calcul vers les esclaves ainsi que les données nécessaires à leur calcul.

Pour ce faire, nous utiliserons les différentes lignes qui existent entre le processeur de contrôle et les esclaves.

Les programmes seront transférés dans la mémoire commune à partir de disques par exemple et devront être envoyés vers leurs processeurs respectifs.

L'appel lancé par le maître au premier esclave, étant un signal représentant une demande d'interruption non masquable ($\overline{NMI}$) au micro-processeur esclave, celui-ci se branchera à une certaine adresse.

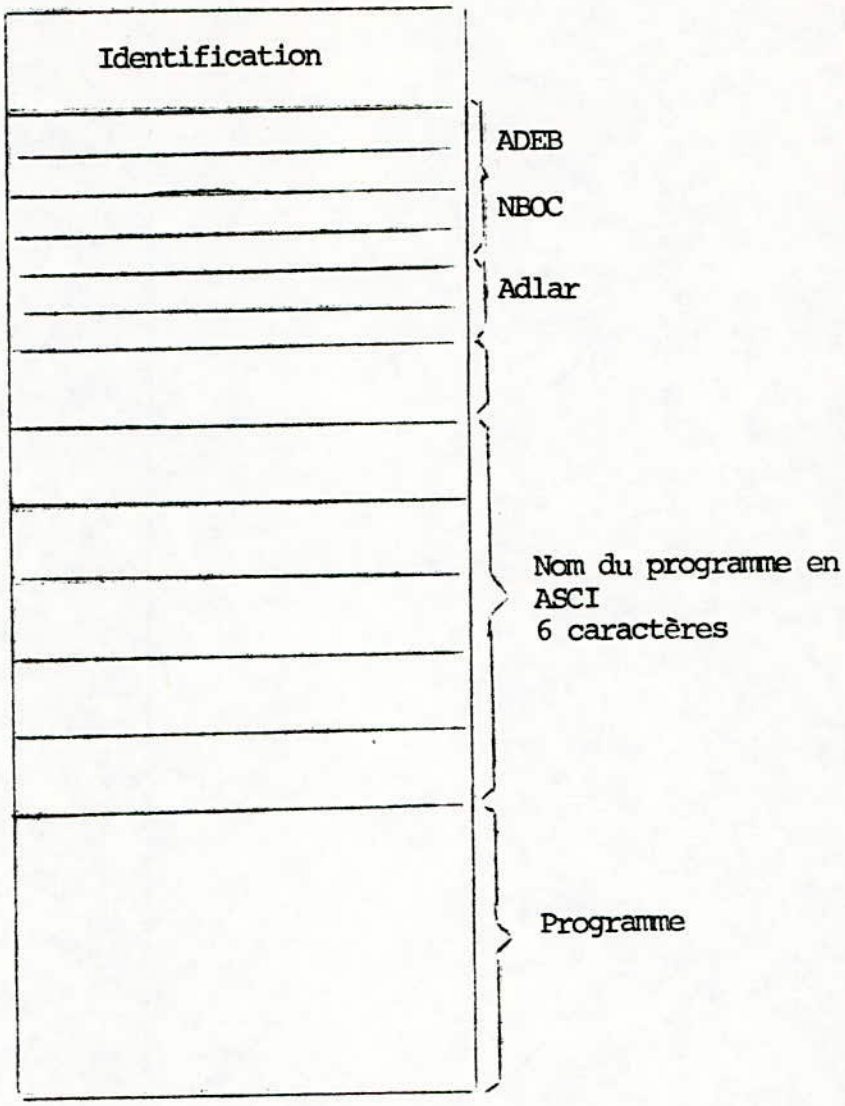
Cette adresse correspond à un sous-programme de réception de la table contenant le programme (ou les conditions initiales). Cette table sera appelée Varta.

Dans le cas d'une transmission de programme de calcul, la table Varta aura la structure suivante : (voir page suivante).

ADEB : adresse début d'implantation
NBOC : nombre octets (longueur de la table)
Adlar : adresse de lancement.

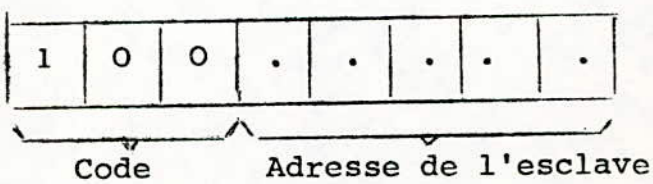
Le premier octet permet l'identification. Il contient le numéro de l'esclave à qui est destiné le programme et un code permettant au maître et aux esclaves de différencier la transmission d'un programme ou de constantes.

../..



- Dans le cas d'une transmission de programme :

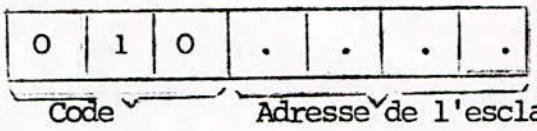
octet d'identification



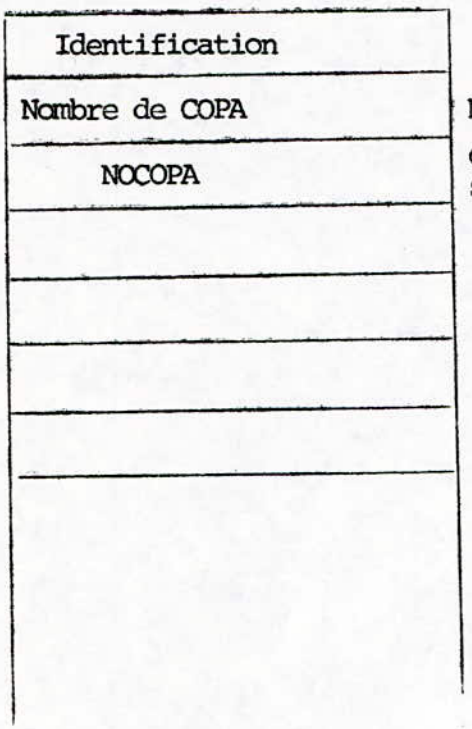
../..

- Cas d'une transmission de constantes :

octet d'identification



Organisation de Buvar dans le cas d'une transmission de constantes :

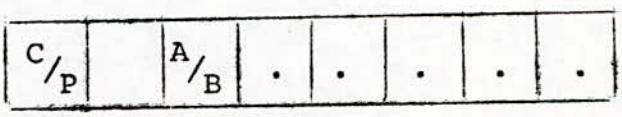


-

Nombre de constantes ou para.

octet donnant tous les renseignements sur la contante qui suit.

NOCOPA



C/P = 1 On a une constante qui suit NOCOPA

= 0 On a un paramètre

../..

A/B = 1 La constante qui suit est un élément de la matrice A

= 0 La constante qui suit est un élément de la matrice B.

Les 5 derniers bits donnent le numéro de la constante.

Représentation des phénomènes :

Tous phénomènes peuvent être régis par une équation d'état, ayant la forme suivante :

$$\begin{cases} \dot{X} = AX + BU \\ Y = CX \end{cases}$$

A, B, C étant des matrices

X, $\dot{X}$: vecteur représentant les états

Y : sortie

U : commande

Les matrices dépendent de la complexité du phénomène.

Exemple :

$$\begin{pmatrix} \dot{X}_1 \\ \dot{X}_2 \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} + \begin{pmatrix} B_1 & B_2 \\ B_3 & B_4 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$$

$$\begin{pmatrix} Y \end{pmatrix} = \begin{pmatrix} C_1 & C_2 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

Chaque processeur a pour travail, le calcul d'une variable d'état.

../..

Notre exemple nécessite deux calculateurs :

E_1 : affecté au calcul de $\dot{X}_1$

E_2 : affecté au calcul de $\dot{X}_2$

Pour effectuer ces calculs, E1 et E2 ont besoin du programme de calcul mais aussi des différentes conditions initiales et des constantes. Il faudra donc leur préciser :

Pour E_1 : A_{11}, A_{12}, B_1 et B_2

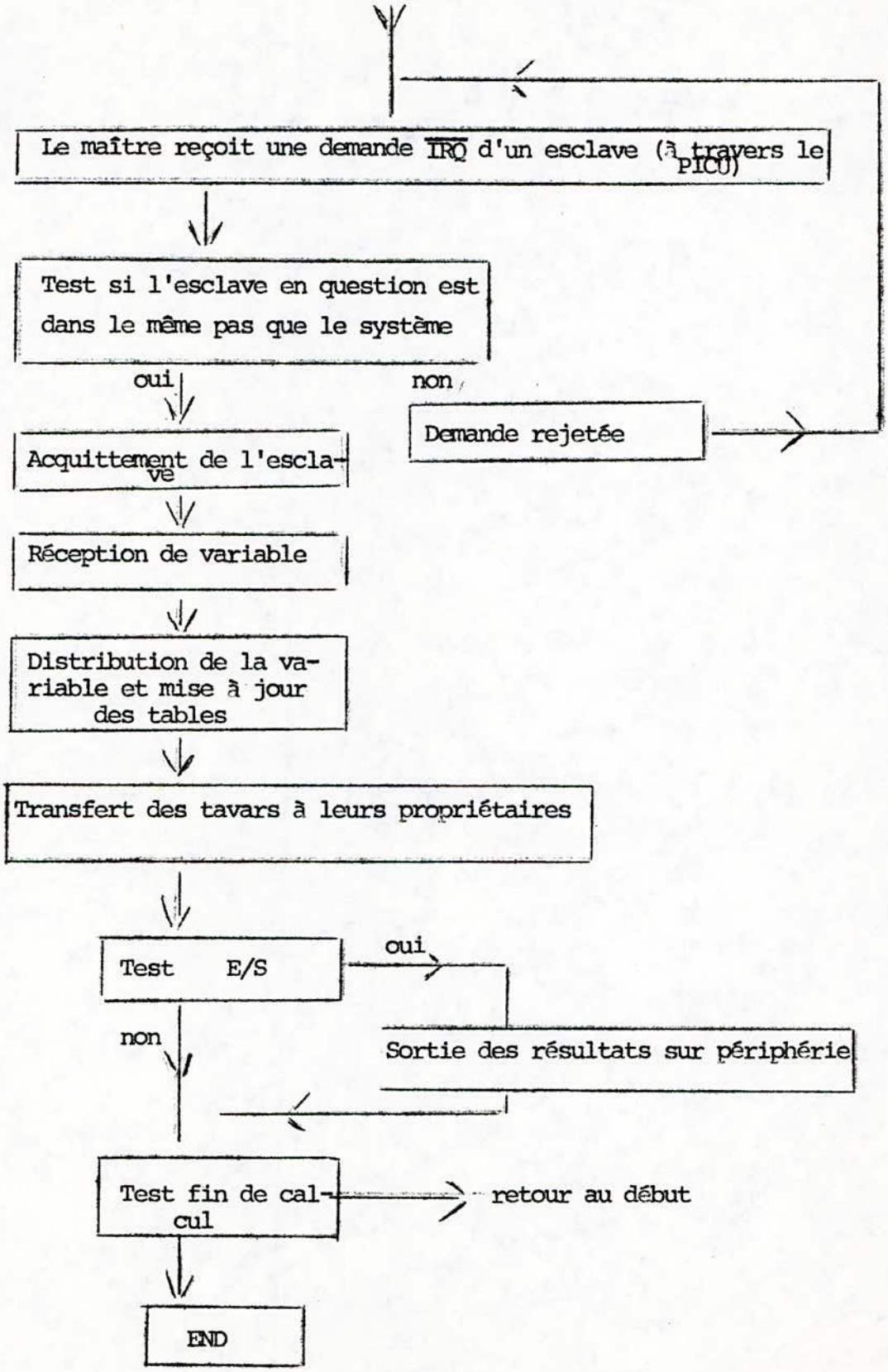
Pour E_2 : A_{21}, A_{22}, B_3 et B_4

ainsi que X_1 (t_i) et X_2 (t_i).

On peut remarquer que la 1ère ligne des matrices A et B contient tout ce dont a besoin E_1 pour effectuer son calcul. En généralisant cette méthode à notre système celui-ci est en mesure d'effectuer la résolution de 32 variables.

Dans notre cas, les constantes a i j sont, soit égales à 1, soit égales à 0. La matrice parait évidemment plus simple.

Ces constantes représentent l'interconnection entre les variables i et j. C'est à partir de celle-ci que l'on pourra établir les tables de correspondance Refco.


Le maître reçoit une demande IRQ d'un esclave (à travers le PICU)
Test si l'esclave en question est dans le même pas que le système
oui
non
Demande rejetée
Acquittement de l'esclave
Réception de variable
Distribution de la variable et mise à jour des tables
Transfert des tavars à leurs propriétaires
Test E/S
oui
non
Sortie des résultats sur périphérie
Test fin de calcul
retour au début
END

CONCLUSION

Le travail réalisé, nous a permis de découvrir un domaine dans lequel nous n'avions auparavant que de légères notions.

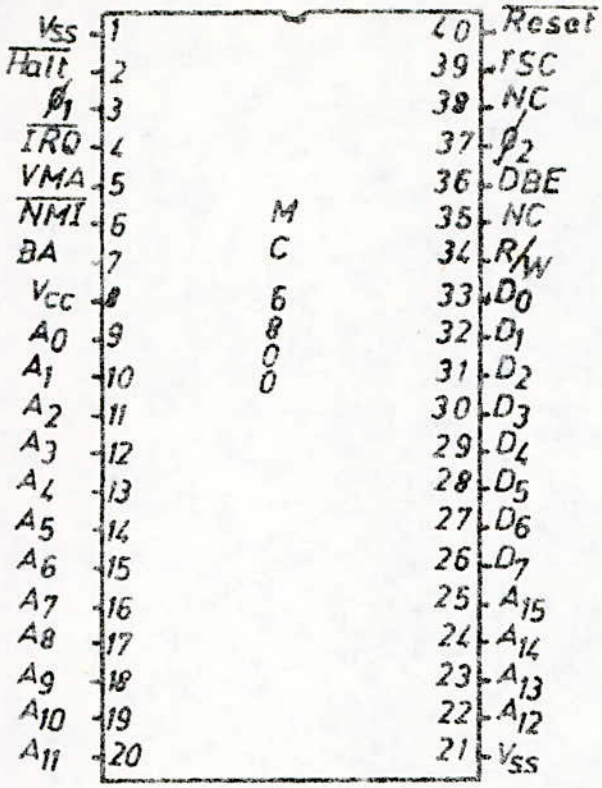
LA PREMIERE PARTIE CONSTITUAIT EN L'ETUDE ET LA REALISATION HARDWARE D'UN SYSTEME MULTIPROCESSEUR. La réalisation de nos *CARTES MAITRE ET ESCLAVE* a été pour nous, l'occasion d'aborder des problèmes pratiques, qui se posent dans ce domaine et d'apprendre certaines techniques d'utilisation des composants, de tests et de mise au point des cartes.

LA CONCEPTION DU MONITEUR DE GESTION DU DIALOGUE REPRESENTAIT LA SECONDE PARTIE DE NOTRE TRAVAIL. Cette étude *SOFTWARE* a été enrichissante à plus d'un titre. Il nous a fallu organiser un type de communication entre microprocesseur en utilisant le principe des ressources partagées.

En espérant que cette étude sera utilisée dans l'avenir, à des applications concrètes, nous ajouterons que nous-mêmes avons bénéficié de l'excellent travail effectué par nos collègues des semestres précédents, nous procurant ainsi, une bonne base de départ à la réalisation de ce projet de fin d'études.

ANNEXE

BROCHAGE DU 6800



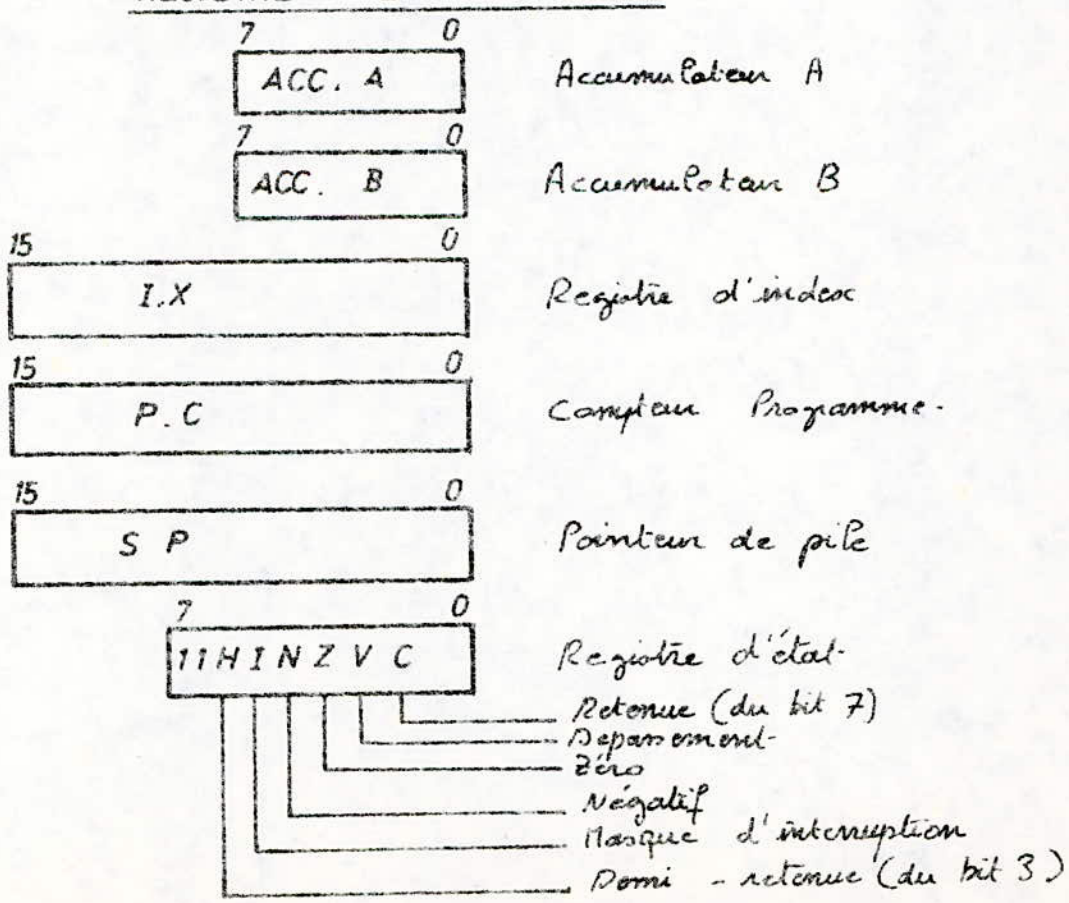
Lignes de commandes

- $\overline{Reset}$: Remise au point initial
- $\overline{HALT}$: Arrêt
- $\overline{NMI}$: Interruption non masquable
- $\overline{IRQ}$: Interruption masquable
- R/W : Lecture - écriture
- DBE : Activation du bus données
- VMA : Adresse mémoire validé
- TSC : Commande trois états.
- BA : Bus disponible

Autres lignes

- A_0 à A_{15} : Lignes d'adresses
- D_0 à D_7 : Lignes de données
- Φ_1 et Φ_2 : Phases d'horloge non chevauchantes

REGISTRE INTERNE DU 6800



8T26

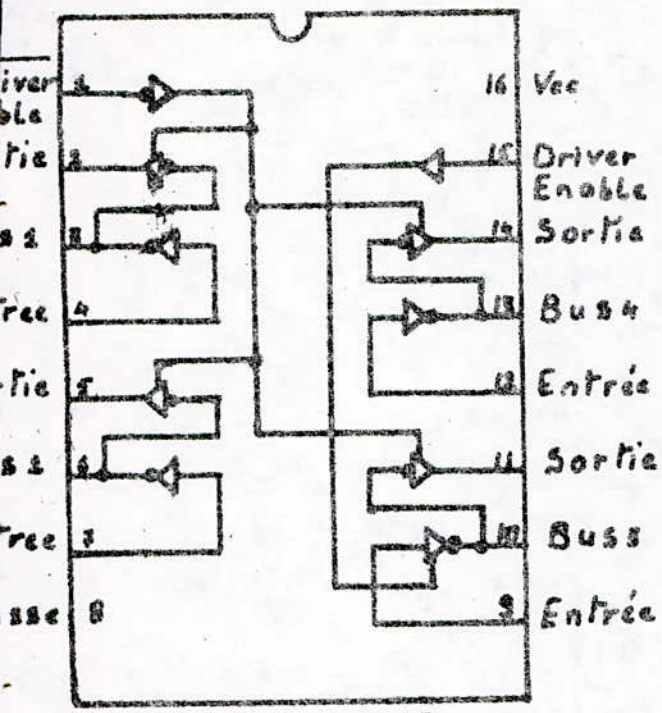


FIG : 13

8T95

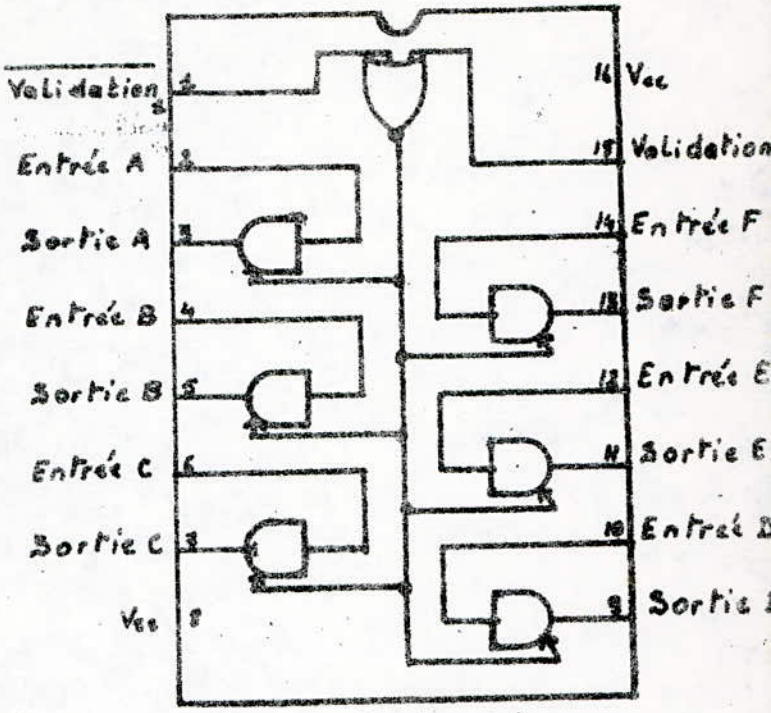


FIG : 14

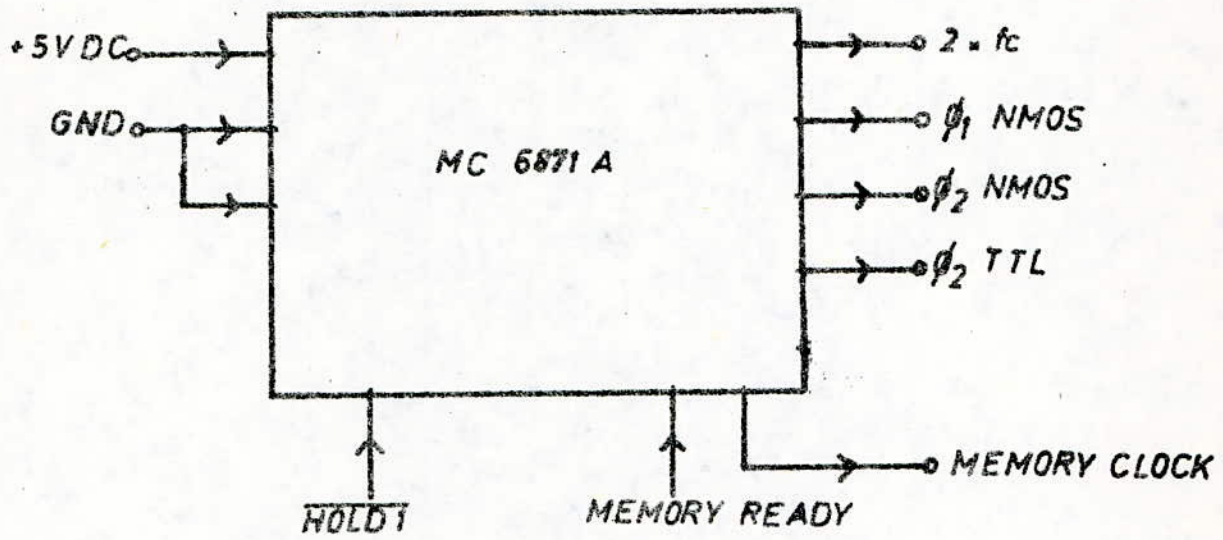
Table De verite : 8T26

Driver Enable	Receiver Enable	Sortie	Entree	Bus
L	L	L	X	L
L	L	H	X	H
L	H	X	X	isolé
H	H	X	L	L
H	H	X	H	H

Table De verite 8T95

$\overline{\text{Valid 2}}$	$\overline{\text{Valid 1}}$	Entrée	Sortie
L	L	L	L
L	L	H	H
L	H	X	Z
H	L	X	Z
H	H	X	Z

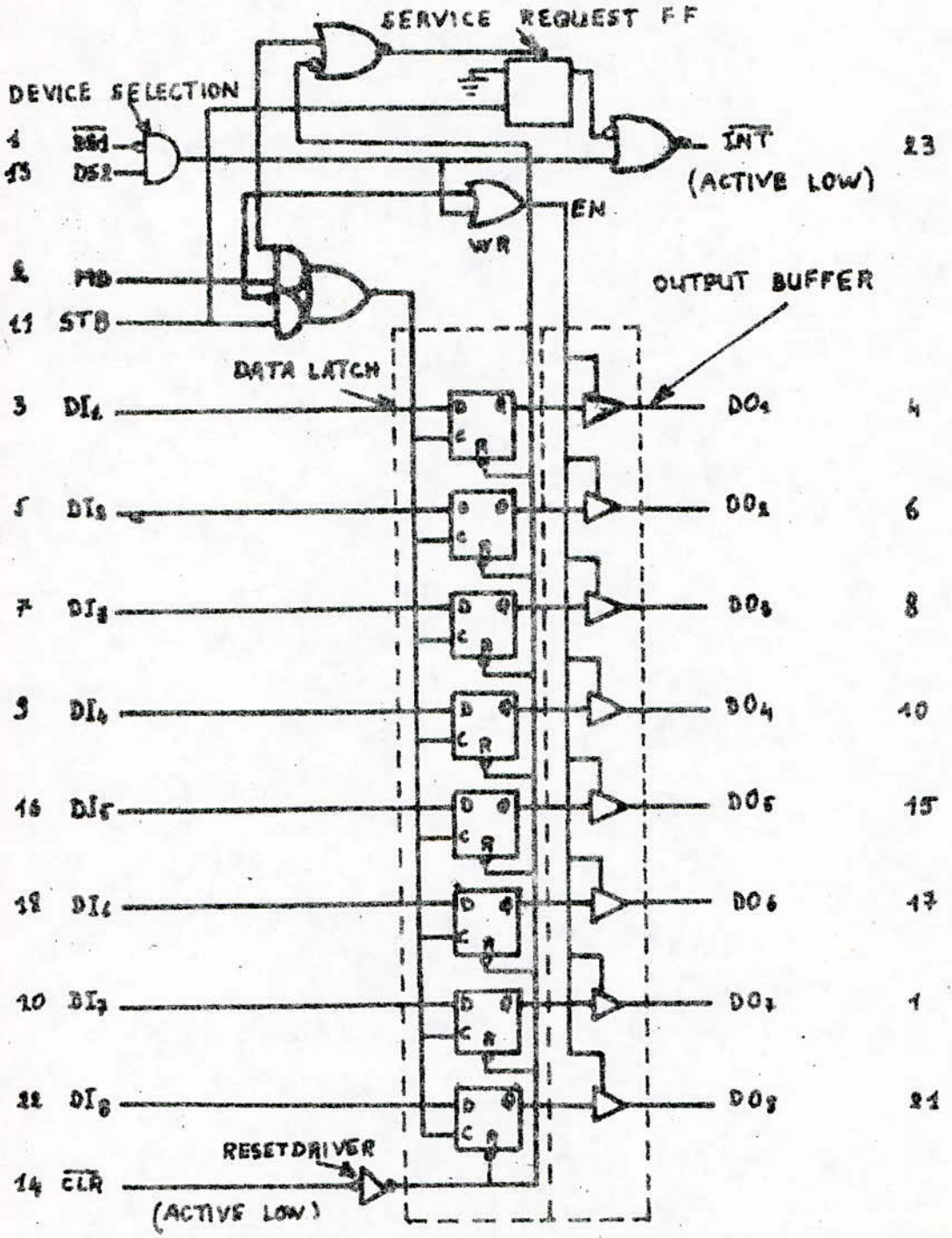
BROCHAGE DE L HORLOGE MC 6871 A



PIN	CONNECTION
1	GND
3	MEMORY CLOCK
5	ϕ_2 TTL
7	V_{cc} (+5VDC)
12	ϕ_2 NMOS
13	ϕ_1 NMOS
18	GND
20	$\overline{HOLD\ 1}$
22	MEMORY READY
24	2 x fc

PORT D'ENTREE/SORTIE : 8212

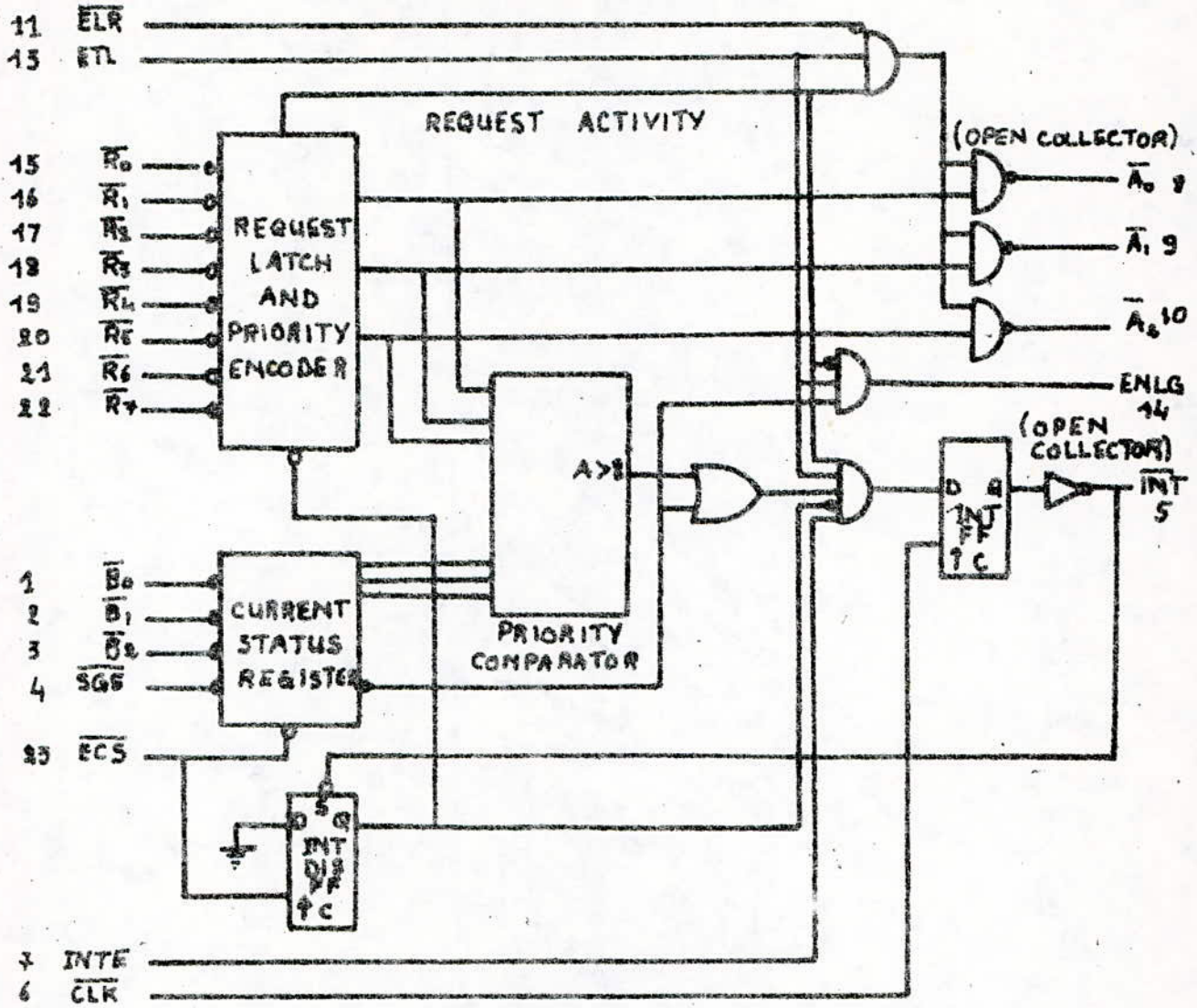
LOGIC DIAGRAM



PIN NAMES	
$DI_1 - DI_8$	DATA IN
$DO_1 - DO_8$	DATA OUT
$\overline{DS1} - DS2$	DEVICE SELECT
MD	MODE
STB	STROBE
$\overline{INT}$	INTERRUPT (ACTIVE LOW)
$\overline{CLR}$	CLEAR (ACTIVE LOW)

CONTROLEUR DE PRIORITE S : 8214

LOGIC DIAGRAM



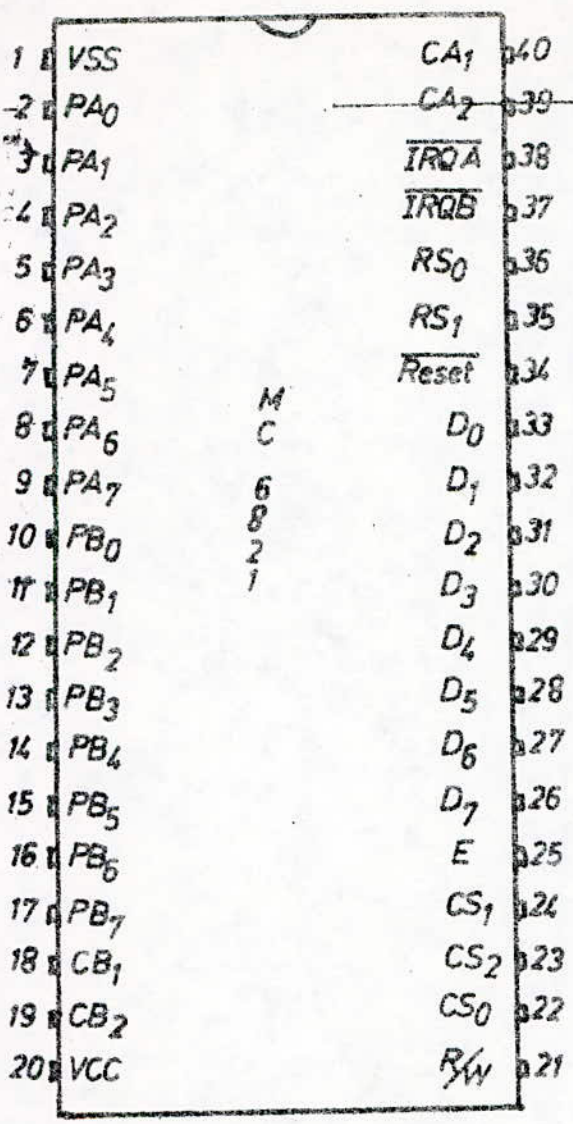
PIN NAMES

INPUTS:

R_0 - R_7	REQUEST LEVELS (R_7 HIGHEST PRIORITY)
B_0 - B_2	CURRENT STATUS
SGS	STATUS GROUP SELECT
ECS	ENABLE CURRENT STATUS
INTE	INTERRUPT ENABLE
CLK	CLOCK (INT F.F)
ELR	ENABLE LEVEL READ
ETLG	ENABLE THIS LEVEL GROUP

OUTPUTS:

A_0 - A_2	REQUEST LEVELS	} OPEN COLLECTOR
INT	INTERRUPT (ACT. LOW)	
ENLG	ENABLE NEXT LEVEL GROUP	



BROCHAGE DU PIA

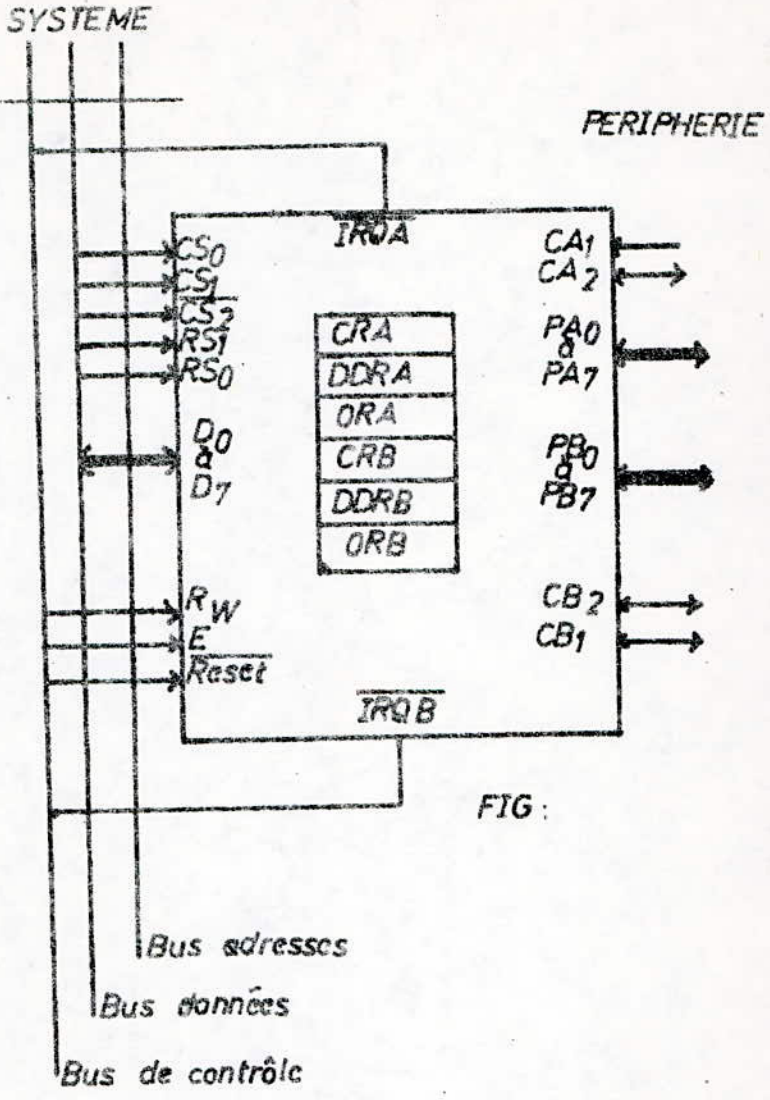


FIG :

RS_1	RS_0	CRA_2	CRB_2	Registre adresse
0	1	-	-	CRA
0	0	0	-	DDRA
0	0	1	-	ORA et INTERFACE
1	1	-	-	CRB
1	0	-	0	DDRB
1	0	-	1	ORB et INTERFACE

Tableau N° Adressage du PIA

BROCHAGE DU DECODEUR MC 14515

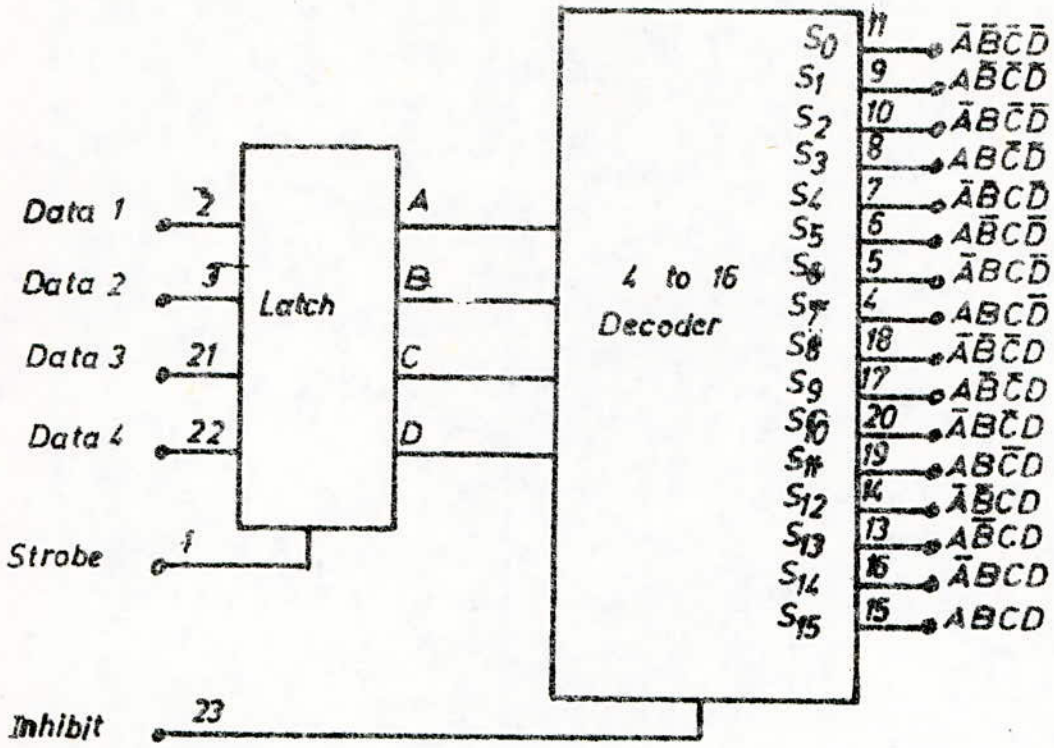


TABLE DE VÉRITE

INHIBIT	DATA INPUTS				SELECTED OUTPUT MC 14515 = Logic "0"
	D	C	B	A	
0	0	0	0	0	S_0
0	0	0	0	1	S_1
0	0	0	1	0	S_2
0	0	0	1	1	S_3
0	0	1	0	0	S_4
0	0	1	0	1	S_5
0	0	1	1	0	S_6
0	0	1	1	1	S_7
0	1	0	0	0	S_8
0	1	0	0	1	S_9
0	1	0	1	0	S_{10}
0	1	0	1	1	S_{11}
0	1	1	0	0	S_{12}
0	1	1	0	1	S_{13}
0	1	1	1	0	S_{14}
0	1	1	1	1	S_{15}
1	"	"	"	"	All Outputs = 1

ACCUMULATOR AND MEMORY OPERATIONS	MNEMONIC	ADDRESSING MODES: IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	COND. CODE REG. 5	4	3	2	1	0
																		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	8	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	8	3				$\overline{M} \rightarrow M$	•	•	↕	↕	S	S
	COMA													43	2	1	$\overline{A} \rightarrow A$	•	•	↕	↕	R	S
	COMB													53	2	1	$\overline{B} \rightarrow B$	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	8	3				00 − M → M	•	•	↕	↕	②	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	②	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	②	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	③	③
Decrement	DEC							6A	7	2	7A	8	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	8	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	$A \rightarrow M_{SP}$, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	$B \rightarrow M_{SP}$, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, $M_{SP} \rightarrow A$	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, $M_{SP} \rightarrow B$	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	8	3				M	⑥	•	↕	↕	⑦	⑧
	ROLA													49	2	1	A	⑥	•	↕	↕	⑦	⑧
	ROLB													59	2	1	B	⑥	•	↕	↕	⑦	⑧
Rotate Right	ROR							66	7	2	76	8	3				M	⑥	•	↕	↕	⑦	⑧
	RORA													46	2	1	A	⑥	•	↕	↕	⑦	⑧
	RORB													56	2	1	B	⑥	•	↕	↕	⑦	⑧
Shift Left, Arithmetic	ASL							68	7	2	78	8	3				M	⑥	•	↕	↕	⑦	⑧
	ASLA													48	2	1	A	⑥	•	↕	↕	⑦	⑧
	ASLB													58	2	1	B	⑥	•	↕	↕	⑦	⑧
Shift Right, Arithmetic	ASR							67	7	2	77	8	3				M	⑥	•	↕	↕	⑦	⑧
	ASRA													47	2	1	A	⑥	•	↕	↕	⑦	⑧
	ASRB													57	2	1	B	⑥	•	↕	↕	⑦	⑧
Shift Right, Logic	LSR							64	7	2	74	8	3				M	⑥	•	R	↕	⑦	⑧
	LSRA													44	2	1	A	⑥	•	R	↕	⑦	⑧
	LSRB													54	2	1	B	⑥	•	R	↕	⑦	⑧
Store Acmltr	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	8	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R

- Code d'instruction du 6800 (1).

INDEX REGISTER AND STACK POINTER OPERATIONS	MNEMONIC	IMMED OP	~	#	DIRECT OP	~	#	INDEX OP	~	#	EXTND OP	~	#	INHER OP	~	#	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$(X_H/X_L) - (M/M+1)$	•	•	②	‡	⑥	•
Decrement Index Reg	DEX													09	4	1	$X - 1 \rightarrow X$	•	•	‡	‡	•	•
Decrement Stack Pntr	DES													34	4	1	$SP - 1 \rightarrow SP$	•	•	•	•	•	•
Increment Index Reg	INX													08	4	1	$X + 1 \rightarrow X$	•	•	‡	‡	•	•
Increment Stack Pntr	INS													31	4	1	$SP + 1 \rightarrow SP$	•	•	•	•	•	•
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	6	2	FE	5	3				$M \rightarrow X_H$ $(M+1) \rightarrow X_L$	•	•	⑤	‡	R	•
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	6	2	BE	5	3				$M \rightarrow SP_H$ $(M+1) \rightarrow SP_L$	•	•	⑤	‡	R	•
Store Index Reg	STX				DF	5	2	EF	7	2	FF	6	3				$X_H \rightarrow M$, $X_L \rightarrow (M+1)$	•	•	⑤	‡	R	•
Store Stack Pntr	STS				9F	5	2	AF	7	2	BF	6	3				$SP_H \rightarrow M$, $SP_L \rightarrow (M+1)$	•	•	⑤	‡	R	•
Indx Reg → Stack Pntr	TXS													35	4	1	$X - 1 \rightarrow SP$	•	•	•	•	•	•
Stack Pntr → Indx Reg	TSX													30	4	1	$SP + 1 \rightarrow X$	•	•	•	•	•	•

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE OP	~	#	INDEX OP	~	#	EXTND OP	~	#	INHER OP	~	#	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	•	•	•	•	•	•
Branch If Carry Clear	BCC	24	4	2										C = 0	•	•	•	•	•	‡
Branch If Carry Set	BCS	25	4	2										C = 1	•	•	•	•	•	‡
Branch If = Zero	BEQ	27	4	2										Z = 1	•	•	•	‡	•	•
Branch If ≥ Zero	BGE	2C	4	2										$N \oplus V = 0$	•	•	•	•	‡	•
Branch If > Zero	BGT	2E	4	2										$Z + (N \oplus V) = 0$	•	•	•	•	‡	•
Branch If Higher	BHI	22	4	2										C + Z = 0	•	•	•	•	•	•
Branch If ≤ Zero	BLE	2F	4	2										$Z + (N \oplus V) = 1$	•	•	•	•	‡	•
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	•	•	•	•	•	•
Branch If < Zero	BLT	2D	4	2										$N \oplus V = 1$	•	•	•	•	‡	•
Branch If Minus	BMI	2B	4	2										N = 1	•	•	•	•	•	•
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	•	•	•	•	•	•
Branch If Overflow Clear	BVC	28	4	2										V = 0	•	•	•	•	•	•
Branch If Overflow Set	BVS	29	4	2										V = 1	•	•	•	•	•	•
Branch If Plus	BPL	2A	4	2										N = 0	•	•	•	•	•	•
Branch To Subroutine	BSR	8D	8	2											•	•	•	•	•	•
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	•	•	•	•	•	•
Jump To Subroutine	JSR				AD	8	2	BD	9	3					•	•	•	•	•	•
No Operation	NOP										01	2	1	Advance Prog Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1	⑪						
Return From Subroutine	RTS										39	5	1	See special Operation	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1		•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1		•	⑪	•	•	•	•

CONDITION CODE REGISTER OPERATIONS	MNEMONIC	INHER OP	~	#	BOOLEAN OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Clear Carry	CLC	0C	2	1	$0 \rightarrow C$	•	•	•	•	•	R
Clear Interrupt Mask	CLI	0E	2	1	$0 \rightarrow I$	•	R	•	•	•	•
Clear Overflow	CLV	0A	2	1	$0 \rightarrow V$	•	•	•	•	R	•
Set Carry	SEC	0D	2	1	$1 \rightarrow C$	•	•	•	•	•	S
Set Interrupt Mask	SEI	0F	2	1	$1 \rightarrow I$	•	S	•	•	•	•
Set Overflow	SEV	0B	2	1	$1 \rightarrow V$	•	•	•	•	S	•
Acmltr A → CCR	TAP	06	2	1	$A \rightarrow CCR$	———	⑫	———			
CCR → Acmltr A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

LEGEND

OP	Operation Code (Hexadecimal)	00	Byte = Zero
~	Number of MPU Cycles	H	Half-carry from bit 3
#	Number of Program Bytes	I	Interrupt mask
+	Arithmetic Plus	N	Negative (sign bit)
−	Arithmetic Minus	Z	Zero (byte)
•	Boolean AND	V	Overflow, 2's complement
M_{SP}	Contents of memory location pointed to be Stack Pointer	C	Carry from bit 7
		R	Reset Always
+	Boolean Inclusive OR	S	Set Always
⊕	Boolean Exclusive OR	‡	Test and set if true cleared otherwise
M	Complement of M	•	Not Affected
→	Transfer Into	CCR	Condition Code Register
0	Bit = Zero	LS	Least Significant
		MS	Most Significant

CONDITION CODE REGISTER NOTES

(Bit set if test is true and cleared otherwise)

(1) (Bit V) Test: Result = 10000000 ?
(2) (Bit C) Test: Result = 00000000 ?
(3) (Bit C) Test: Decimal value of most significant BCD Character greater than nine ? (Not cleared if previously set)
(4) (Bit V) Test: Operand = 10000000 prior to execution ?
(5) (Bit V) Test: Operand = 01111111 prior to execution ?
(6) (Bit V) Test: Set equal to result of N⊕C after shift has occured
(7) (Bit N) Test: Sign bit of most significant (MS) byte of result = 1 ?
(8) (Bit V) Test: 2's complement overflow from subtraction of LS bytes ?
(9) (Bit N) Test: Result less than zero ? (Bit 15 = 1)
(10) (All) Load Condition Code Register from Stack. (See Special Operations)
(11) (Bit I) Set when interrupt occurs. If previously set, a Non Maskable interrupt is required to exit the wait state.
(12) (ALL) Set according to the contents of Accumulator A

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

- M 6800 MICROPROCESSOR APPLICATION MANUAL
(THOMSON)

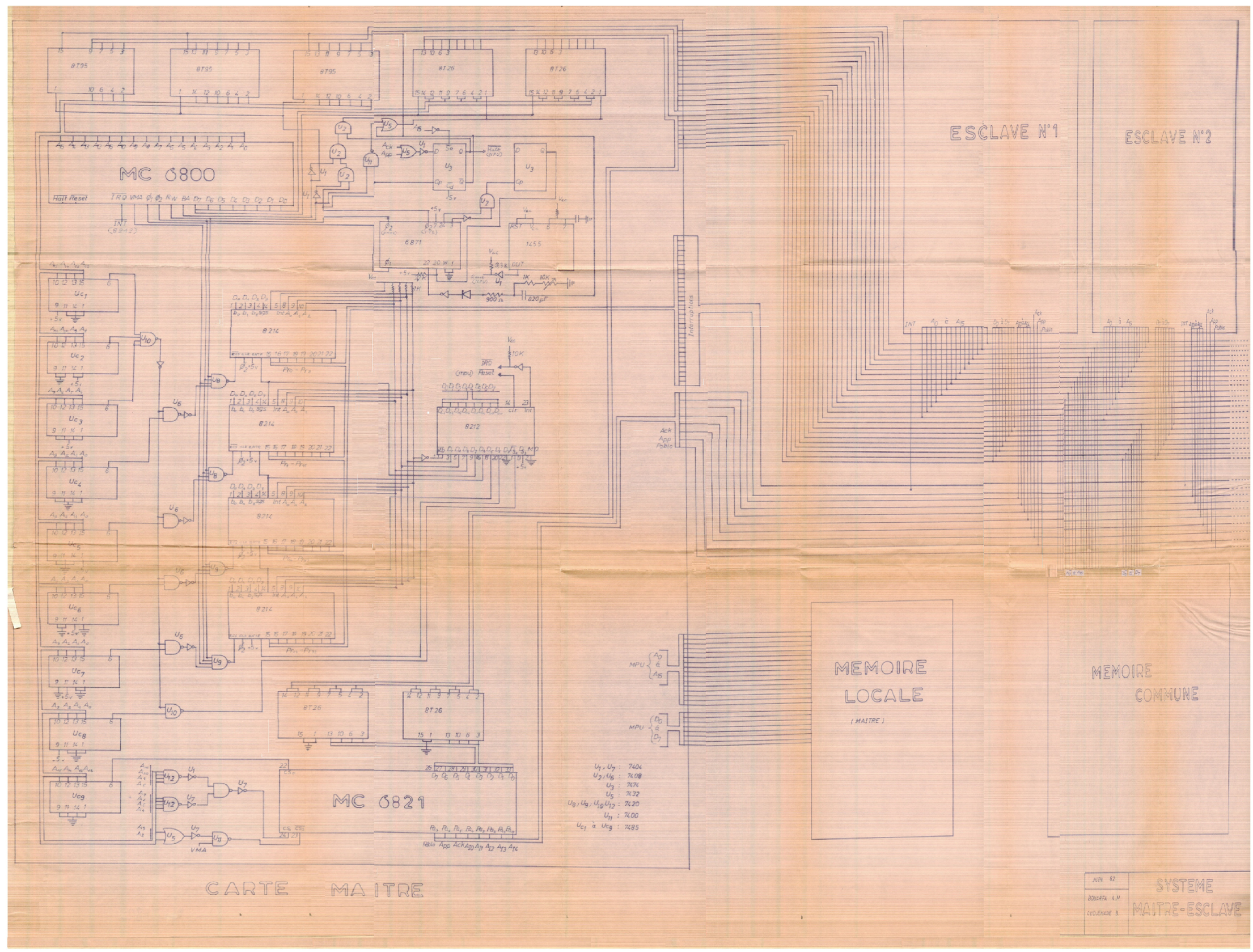
- DIGITAL SIMULATION OF PHYSICAL SYSTEM
(JOSEPH S.ROSKO)

- HYBRID COMPUTATION
(BEKEY AND KARPLUS)

THESES :

1) ETUDE D'UN SYSTÈME MULTIPROCESSEUR A RESSOURCE PARTAGEE

2) ETUDE ET REALISATION D'UNE UNITE DE DIALOGUE MICRO - MICRO.


MC 6800
MC 6821
8T95
8T26
6871
1455
8214
8212
ESCLAVE N°1
ESCLAVE N°2
Interruptions
MEMOIRE LOCALE
(MAITRE)
MEMOIRE COMMUNE
U_1, U_7 : 7404
U_2, U_6 : 7408
U_3 : 7474
U_5 : 7432
U_8, U_9, U_{10}, U_{12} : 7420
U_{11} : 7400
U_{c1} à U_{c9} : 7485
CARTE MAITRE
JUIN 82
BOUARFA A.M.
GOUDJIANE B.
SYSTEME MAITRE-ESCLAVE