REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
« HOUARI BOUMEDIENNE »

ECOLE NATIONALE POLYTECHNIQUE

Département d'Electronique

Filière : D'INGENIORAT D'ETAT EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

ETUDE DU MC 6809
APPLICATION AUX COMMANDES AZIMUTALES

Proposé par :
Mr J. HERRY

Dirigé par :
Mr A. SAIDJ
Mr A. BOURKEB

Etudié par :
Lounis KESSAL
Akila KEMMOUCHE

Promotion Juin 1983

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE
« HOUARI BOUMEDIENNE »

ECOLE NATIONALE POLYTECHNIQUE

Département d'Electronique

Filière : D'INGENIORAT D'ETAT EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

ETUDE DU MC 6809
APPLICATION AUX COMMANDES AZIMUTALES

Proposé par :
Mr J. HERRY

Dirigé par :
Mr A. SAIDJ
Mr A. BOURKEB

Etudié par :
Lounis KESSAL
Akila KEMMOUCHE

Promotion Juin 1983

----oo--REMERCIEMENTS--oo-----

Nous tenons à exprimer ici notre profonde reconnaissance à Mrs: SAIDJ, BOURKE Bet HERRY pour nous avoir aimablement accueillis dans leur Laboratoire, mis à notre disposition une documentation abondante et fait profiter de leurs connaissances qui nous ont permis de mener ce projet à sa fin.

Nous remercions Mr FLAMANT pour nous avoir prodigués d'utiles conseils et Mrs BENDALI, JARMOT pour l'aide précieuse qu'ils nous ont apporté.

Nos remerciments vont à Mr Salem KESSAL pour n'avoir ménagé aucun effort pour nous aider tout au long de notre projet.

Que tous ceux qui ont contribué à la mise en œuvre de ce polycopier, de si peut soit-il, trouvent ici notre sincère gratitude.

L. KESSAL

A. KEMMOUCHE

DEDICACES

A ma mere et mon pere

A mon frere RIAD

A toutes mes soeurs

A toute ma famille et particulierement mes oncles LAKHDAR et ISMAIL

A Fella et tous mes amis (es)

KEMMOUCHE AKILA.

A mes parents pour leur sacrifice
A la mémoire de ma soeur
A mes frères et soeurs
A tous ceux qui me sont chers
A tous mes amis,
je dédie cet ouvrage.

Louis

----°-- SOMMAIRE --°----

INTRODUCTION

Dans le cadre des systèmes développés à base des microprocesseurs de la famille 6800, le laboratoire d'électronique du C.E.N nous a confié pour notre projet de fin d'études, l'étude et la réalisation d'un micro-ordinateur axé autour du "faux" microprocesseur 16 bits : le 6809. Ce microprocesseur est le premier de la troisième génération, beaucoup plus performant que ceux de la première (6800).

La réalisation d'un micro-ordinateur basé autour de ce nouveau microprocesseur constitue une approche valable des systèmes futurs qui travailleront avec des microprocesseurs de la famille 68000.

Le microprocesseur MC 6809 conçu par MOTOROLA est une version améliorée du MC 6800. Il lui est compatible au niveau du langage source (MNEMONIQUE) et offre de nouveaux concepts logiciels.

Le µ P 6809 présente des caractéristiques qu'on ne peut rencontrer que sur les mini-ordinateurs :

- Programme réentrant ;
- Programme translatable ;
- Possibilité de programmation structurée.

Son organisation interne permet de travailler facilement en langage évolué et particulièrement en PASCAL. Le µ P 6809 est plus performant que le 6800 vu la richesse de son jeu d'instructions et l'efficacité de ses modes d'adressage, en effet ces modes d'adressage ont été implantés sur le 6809 pour lui permettre de travailler avec la programmation structurée de haut niveau. Son jeu d'instructions est plus étendu :

"Avec 59 types d'instructions on peut composer 1464 instructions différentes. Les plus remarquables sont les manipulations de données sur 16 bits. Son architecture interne, avec sept registres dont un registre de page direct permettant l'adressage direct dans tout l'espace mémoire, et son organisation externe comportent des éléments qui étendent les possibilités de programmation. Ainsi le MC 6809 est actuellement l'un des plus complets des microprocesseurs "8 bits". Il est classé parmi les microprocesseurs de la troisième génération."

Pour montrer que notre unité centrale est opérationnelle, nous avons simulé la commande séquentielle de dix moteurs servant à positionner les détecteurs et la cible dans une chambre à réaction montée sur une extension d'un accélérateur type VAN DE GRAFF du laboratoire de physique nucléaire expérimentale du C.E.N (Commissariat aux Energies Nouvelles) d'ALGER.

CHAPITRE I

G E N E R A L I T E S

A - MICROPROCESSEURS :

Au début de 1977, on dénombre une quarantaine de type ou famille de microprocesseurs. Ainsi, cette technique, née vers 1970, aura connu, en l'espace de quelques années, une explosion jugée extraordinaire même par ceux qui ont pu suivre l'évolution générale des produits à semi-conducteurs.

De ce fait, le meilleur choix possible ne peut résulter que d'une longue étude préalable.

Il existe 4 grands types de microprocesseurs :

1. Les microprocesseurs réalisés en un seul circuit intégré, exécutant toutes les fonctions de l'unité centrale de traitement d'un ordinateur. C'est le cas de la majorité des microprocesseurs actuels.

2. Les microprocesseurs en tranches justaposables permettant une extension de l'unité arithmétique et logique (microprogrammation). Tel est le cas de microprocesseur bipolaire (IMP-4)

3. Les microprocesseurs avec ROM et ram, horloge et entrées-sorties sur la même et unique puce de circuit intégré. On trouve là le TMS 1000 de Texas, CMOS MC 146805E2 de MOTOROLA.

4. Les microprocesseurs à la demande, c'est à dire spécialisés et destinés à une application donnée (exemple : ITT 7150 pour la commande des machines à laver).

Les principales caractéristiques à considérer sont alors : la longueur des mots, la vitesse de calcul, la puissance, le logiciel, etc.

L'apparition de la technique de système programmable a donné naissance à des microprocesseurs de première génération puis 2ème génération (1973) aux performances considérables.

C'est dans la perspective d'évolution de cette technique que vint la troisième génération pour mettre au point des microprocesseurs très performants travaillant avec des vitesses de calcul élevées et ayant des capacités d'adressage importantes, La longueur des mots peut atteindre 32 bits permettant de concurrencer les mini-ordinateurs (Z80, TMS 9980). Ce critère est très important puisqu'il conditionne la taille des nombres que peut traiter le microprocesseur.

C'est ainsi que le microprocesseur hybride MC 6809, qu'on se propose d'étudier, a été annoncé pour offrir des moyens de programmation de haut niveau donnant une approche valable au microprocesseur 68000 de MOTOROLA aux caractéristiques très enviables :

- Grande vitesse dans l'exécution des instructions (horloge : 8 MHz) et traitant des formats plus grands : 16 et 32 bits (le 6809 travaille sur 8 et 16 bits).

- Son espace mémoire adressable est de 16 Mégabytes. (Celui du 6809 n'est que de 64 K-bytes).

B - RAPPELS SUR LES MICRO-ORDINATEURS

1. Définition :

- L'ordinateur est une machine automatique capable d'exécuter des opérations d'informations, sous le contrôle de séquences d'instruction préalablement fournies.

- Le Micro-Ordinateur est un ordinateur dont les principaux organes constitutifs sont réalisés en circuits L.S.I. (LARGE SCALE INTEGRATION): Intégration à grande échelle.

  * Un circuit L.S.I particulier constituant l'unité centrale est appelé microprocesseur.

  * Des RAM (RANDOM ACCES MEMORY) et des ROM (READ ONLY MEMORY) constituent les mémoires pour stocker les informations et les programmes de traitement.

  * Des interfaces(adaptateurs) Pour l'adaptation du MPU avec les périphériques qui sont des organes de dialogue avec le microprocesseur.

La conception d'un système est totalement différente en logique cablée ou en logique programmée. En logique cablée le concepteur organise son système en assemblant des circuits intégrés logiques accomplissant des opérations logiques suivant la tâche pour laquelle le système est spécialisé ce qui revèle le matériel (HARDWARE).

En logique programmée, le HARD n'est qu'une phase préliminaire puisqu'il nécessite des programmes et des systèmes de programmation (SOFTWARE) qui commande son fonctionnement contrairement à la logique cablée, où une fois les composants fonctionnellement interconnectés, le travail est terminé.

2. Architecture d'un micro-ordinateur :

Dans la plupart des applications, le CPU doit commander et communiquer avec divers systèmes d'entrée/sortie : E/S .

L'information qui circule entre le CPU et les unités périphériques, est soit une donnée soit une commande. Les données sont des informations essentiellement numériques ou alphanumériques codés suivant un code binaire adéquat, tel que le binaire direct le B C D ou le code ASCII (American Standard Code for Information Interchange).

La transmission des informations entre le MPU et les unités d'E/S est une partie délicate de la conception des systèmes.

Un circuit intégré d'adaptation appelé interface est nécessaire entre le CPU et les périphériques : clavier, lecteur de bande perforée, une imprimante ou tout simplement un ensemble de signaux électriques provenant de capteurs qui seront lus par le MPU.

Ces signaux sont mémorisés, traités et ensuite envoyés vers la périphérie pour être visualisés ou enregistrés ou bien agir sur un système industriel (commande numérique de système de machine, contrôle de processus...)

Suivant le mode de transmission de l'information (série ou parallèle) il existe deux modules d'interfaces actuellement parmi les plus performants pour être adaptés directement au MPU :

* Le "P I A (ou PERIPHERAL INTERFACE ADAPTER) pour la transmission en mode parallèle

* L'A C I A (ou Asynchrone Communication Interface) pour la transmission en mode série

CHAPITRE II

ETUDE DU MICROPROCESSEUR MC 6809

A - Rappels sur la famille 6800 :

On rappelle dans ce qui suit les caractéristiques et performances des différentes unités centrales "8 bits" de la famille 6800 existant actuellement et qui peuvent intéresser les concepteurs d'automatismes industriels.

Un certain nombre de caractéristiques sont communes à ces unités centrales, notamment en ce qui concerne les périphériques. Ceux-ci sont toujours considérés comme des positions mémoires, ils ne nécessitent pas d'instructions spécifiques d'E/S.

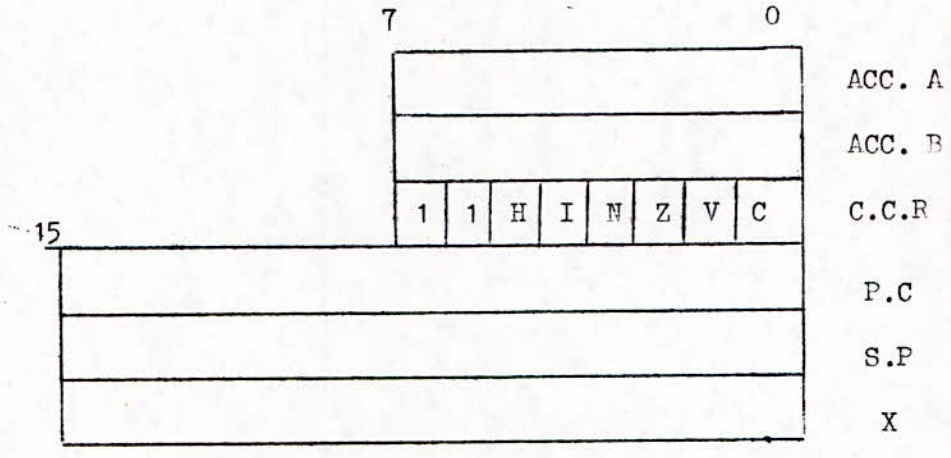
Caractéristiques communes :

- Compatibilité T T L direct ;
- Alimentation mono-tension : + 5 V ;
- Capacité d'adressage : 64 K-octets par bus d'adressage 16 lignes.
- Bus de données ; bidirectionnel à 8 bits ;
- Technologie N - MOS ;
- Compatibilité ascendante entre toutes les unités centrales de la famille 6800.
- Utilisation des mêmes organes d'E/S : PIA 6821, ACIA 6850, Temporisateur 6840,... etc.

Unité centrale 6800 :

C'est la première unité centrale de la famille 6800, aux performances modestes, mais encore largement utilisée dans les applications simples.

La figure suivante montre les six registres internes qui sont à la disposition du programmeur.



1/ LES DIFFERENTS REGISTRES ADRESSABLES :

* Le compteur de programme : contient l'adresse courante du programme

* Le pointeur de pile (Stack pointer) : contient en permanence la première adresse disponible de la pile.

* Le registre d'index (Index Register) : sert de pointeur d'adresse dans le mode d'adressage indexé.

* Le registre d'état ou registre des codes condition (C.C.R) :
Comporte les indicateurs d'états (flag) suivants :

  . H : Half carry ou demi-retenue
  . I : Interrupt flag ou indicateur d'interruption
  . N : Negative flag ou indicateur de signe
  . Z : Zerro flag ou indicateur de zéro
  . V : Overflow ou débordement de capacité
  . C : Carry ou retenu

* Les accumulateurs A et B :ou registres de travail utilisés pour les calculs arithmétiques et logiques et les manipulations de données.

2/ LES DIFFERENTS MODES D'ADRESSAGE :

Le 6800 dispose de 72 instructions et de 7 modes d'adressage.

- Adressage immédiat sur un octet ou deux.
- Adressage direct sur un octet ;
- Adressage étendu ;
- Adressage indexé avec déplacement non signé sur un octet ;
- Adressage implicite ;
- Adressage relatif.

Les principales unités centrales de la famille 6800 sont :
6800 , 6801, 6802, 6808

Compatible, plus performant : le 6809

Le 6809 est une version améliorée du 6800, son architecture interne qui s'appuie sur un bus à 16 bits comporte des éléments qui étendent les possibilités de programmation.

- Les 2 accumulateurs A et B peuvent être associés en un accumulateur D de 16 bits.

- Deux registres d'index X et Y

- Deux pointeur de pile U (utilisateur) et S (système) qui peuvent également être utilisé comme indexe.

- Un registre de page de 8 bits qui permet un adressage de page.

Il possède un mode d'interruption masquable rapide ($\overline{FIRQ}$) dans lequel seuls le compteur de programme (P C) et le registre d'état (C.C.R) sont sauvegardés dans la pile ; une sortie de synchronisation jusqu'à 100 µ s avec des mémoires lentes (MPDY) permet d'envisager des structures de multiprocesseurs.

Les possibilités d'adressage sont plus riches : il y a 4 registres d'index (X, Y, U et S). L'adressage indexé permet des déplacements signés sur 5,8 et 16 bits, des déplacements dans A, B ou D, une incrémentation ou décrémentation de 1 ou 2.

L'adressage relatif peut se faire sur 8 ou 16 bits, les modes indexés et relatif peuvent également être indirects.

En mode d'adressage de page, la mémoire 64 K-octets peut être considérée comme 256 pages de 256 Octets, les instructions utilisant ce mode d'adressage ne mentionnent alors sur un octet qu'un déplacement à l'intérieur de la page.

Les instructions les plus remarquables sont les manipulations de données sur 16 bits (dont l'addition, soustraction, comparaison..), une multiplication de 8 par 8 bits et 3 instructions différentes d'interruption par logiciel.

B - Etude du microprocessur 6809 :

Le µP 6809 , dernier venu sur le marché des µ P 8 bits, se trouve être le plus puissant grâce à son architecture interne sur 16 bits. Sa mise en oeuvre est très simple puisque le bus de données et le bus d'adresses ne sont pas multiplexés.

Caractéristiques du µ P 6809 :

- C'est un µ P réalisé en technologie N - MOS.

- Alimentation sous une tension unique de 5 V.
- Signaux de bus identiques à ceux de toute la famille 6800 permettant l'emploi de tous les périphériques existant dans celle-ci.
- Possibilité de travailler avec des mémoires lentes.
- 3 sources d'interruption externes matériel, une non masquable, une masquable normale et une rapide.
- Instruction de synchronisation avec un évènement externe.
- 10 modes d'adressage extrêmement puissants.
- 59 mnémoniques d'instruction qui, compte tenu de leurs possibilités équivalent à 1464 instructions différentes.
- Opérations arithmétiques et logiques sur 16 bits.
- Multiplication 8 par 8 bits en une seule instruction.
- Langage source assurant une compatibilité ascendante avec µ P 6800 .

1ère Parie : H A R D W A R E

1 - Signaux du µ P 6809 :

Ainsi que nous l'avons dit, le 6809 est simple d'emploi en raison de ses bus non multiplexés ; la fig (2,0) présente une vue globale des signaux du bus 6809 classés par catégorie.

Nous y remarquons :

* 16 lignes d'adresse notées de A0 à A15. Ces lignes sont unidirectionnelles et à 3 états, elles peuvent commander 4 charges TTL LS ou une charge TTL normale.
* 8 lignes de données notées de D_0 à D_7 . Ces lignes sont bidirectionnelles à 3 états et peuvent aussi attaquer 4 charges TTL LS ou une charge TTL normale.

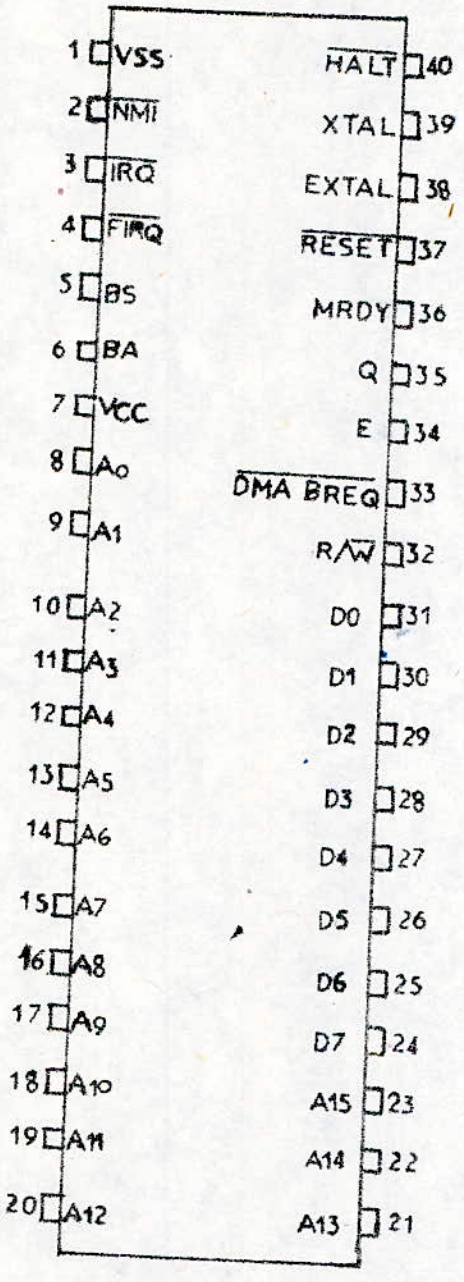


fig.2.0 - MC6809

* 2 lignes d'alimentation : la masse nommée V_{ss} et le + 5 V nommée V_{cc} ou V_{dd} . Cette alimentation doit être égale à + 5 V ± 5 % .

* 2 lignes XTAL et EXTAL : servant à connecter un quartz de fréquence égale à 4 fois la fréquence de fonctionnement du M P U . Il faudra donc un quartz de 4 MHz pour un 6809 travaillant avec une horloge à 1 Mhz. Bien que devant répondre à des critères bien définis dans la fiche techniques du 6809, le quartz peut avoir une fréquence d'oscillation de 4 MHz ± 10 % , l'oscillateur étant très tolérant.

* 1 ligne R/$\overline{W}$ ou lecture-écriture qui indique si le M P U lit (R/$\overline{W}$ à 1) ou écrit (R/$\overline{W}$ à 0) dans la mémoire ou les circuits périphériques. Cette ligne indique le sens de transfert des données, et a la faculté de passer dans le 3ème état.

* 1 ligne $\overline{\text{RESET}}$ ou remise à zéro : une transition négative sur cette entrée permet l'initialisation du M P U et fait charger le P C par le contenu des positions mémoires se trouvant en FFFE et FFFF (en code HEXADECIMAL) qui donne l'adresse sur 16 bits de la première instruction. Les deux octets sont appelés "Vecteur Reset".

* 1 ligne $\overline{\text{MRDY}}$: ou mémoire prête (Memory Ready) qui permet de ralentir les accès mémoires du 6809 pour les mémoires lentes. Un niveau bas sur cette entrée, quand E passe à 1, force la sortie E au niveau haut tant que $\overline{\text{MRDY}}$ reste au niveau bas.
L'allongement de l'horloge ne peut pas dépasser 10µs. La transition négétive de Q n'est pas affectée par $\overline{\text{MRDY}}$, mais les transitions positives de Q sont inhibés pendant le blocage d'horloge.

* E et Q : Les deux lignes E et Q qui sont les horloges du système sont utilisées par tous les éléments du système . Elles travaillent à une fréquence égale au quart de la fréquence du quartz et sont en quadrature l'un par rapport à l'autre. Ces lignes ne sont pas à 3 états et ne sont pas affectées par l'état du processeur ; par contre l'état

haut de E peut être prolongé par le signal $\overline{MRDY}$. Ces deux signaux indiquent en plus l'état stable des lignes d'adresses et de données.

* $\overline{DMA/BREQ}$: Une entrée de demande d'accès au bus permet de faire de l'accès direct en mémoire (Direct Memory Acces) ou un rafraichissement de mémoires dynamiques.
Un niveau bas sur cette entrée bloque l'horloge interne tandis que les sorties E et Q continuent à fonctionner. Les signaux B A (Bus Available) et B S (Bus Status) passent au niveau haut. Les bus d'adresses, de données et la ligne R/$\overline{W}$ passent à l'état haute impédance.

* $\overline{HALT}$: Un niveau logique bas sur l'entrée $\overline{HALT}$ oblige le µ P à s'arrêter à la fin de l'instruction en cours. Le µ P restera figé sans perte de données jusqu'à ce que la ligne $\overline{HALT}$ repasse à l'état haut.
Lorsque le µ P est dans l'état $\overline{HALT}$ les sorties B A et B S passent au niveau haut ce qui indique une reconnaissance du $\overline{HALT}$. Les bus d'adresses et de données passent à l'état haute impédance.

Tant qu'il est dans l'état $\overline{HALT}$, le µ P ne peut pas répondre aux demandes d'interruptions et il ne peut être libéré par un $\overline{reset}$(bien que $\overline{DMA/BREQ}$ soit toujours accepté, et que $\overline{NMI}$ et $\overline{RESET}$ soient mémorisés pour une réponse ultérieure).

Le 6809 a la possibilité d'entrer dans un état d'attente grâce à deux instructions : SYNC et CWAI.

La reconnaissance de synchronisation peut être détectée par les sorties B A et B S qui donnent l'état du microprocesseur.

Le tableau ci-dessous donne les quatres configurations possibles de ces lignes et leurs significations.

B A	B S	ETAT DU MICROPROCESSEUR
0	0	Fonctionnement normal
0	1	Reconnaissance de $\overline{\text{RESET}}$ ou d'interruption
1	0	Reconnaissance de synchronisation
1	1	Bus libéré ou reconnaissance de $\overline{\text{HALT}}$.

La reconnaissance d'interruption se produit pendant la recherche des vecteurs d'interruptions correspondant à $\overline{\text{RESET}}$, $\overline{\text{NMI}}$, $\overline{\text{IRQ}}$, $\overline{\text{FIRQ}}$, SWI, SWI2 et SWI3.

La reconnaissance de synchronisation indique que le microprocesseur attend une synchronisation externe sur une ligne d'interruption.

* $\overline{\text{FIRQ}}$: Demande d'interruption rapide

Cette entrée correspond à une nouvelle interruption du 6809. Un niveau bas sur cette entrée oblige le microprocesseur (quand elle n'est pas masquée) à ranger le contenu du compteur programme PC ainsi que celui du registre d'état C C R sur la pile avant d'aller chercher l'adresse du sous programme de traitement d'interruption $\overline{\text{FIRQ}}$.

* $\overline{\text{IRQ}}$: Demande d'interruption normale :

Un niveau bas sur cette entrée entraine la séquence de traitement de demande d'interruption $\overline{\text{IRQ}}$ à condition que le bit masque (I) du CCR soit à zéro, cette séquence réalisant la sauvegarde de l'état complet du processeur. La réponse sera plus lente que pour le $\overline{\text{FIRQ}}$.

* $\overline{\text{NMI}}$: Interruption Non-Masquable :

Un front descendant appliqué sur cette entrée entraine une séquence d'interruption non-masquable. Une interruption non-masquable ne peut pas être inhibée par programme et possède une priorité supérieure à $\overline{\text{FIRQ}}$, $\overline{\text{IRQ}}$ et aux autres interruptions logicielles
Lors de la reconnaissance de $\overline{\text{NMI}}$, l'état complet du MPU est sauvegardé sur la pile.
On trouvera en annexe A les chronogrammes récapitulant les différents signaux du µ P 6809.

2 - Structure interne du µ P 6809 :

Voir fig (.2)

Dans tout microprocesseur, deux parties distinctes bien qu'étroitement liées sont à considérer : la partie matérielle (HARD) et le logiciel (SOFT).

Nous allons présenter dans ce qui suit les différents registres qui sont à la disposition du pgorammeur : (Fig. (.1)

- Un accumulateur D : 16 bits formé par la concaténation de deux accumulateurs 8 bits A et B. Il est ainsi possible de travailler sur 8 bits dans A et B indépendamment l'un de l'autre et de travailler ensuite sur 16 bits dans D. Le passage d'un mode à l'autre est immédiat et pouvant avoir lieu à n'importe quel moment.

- Deux registres X et Y : utilisés principalement pour l'adressage indexé. Cependant plusieurs opérations arithmétiques et logiques sont possibles sur ceux-ci et entre les indexes et les accumulateurs.

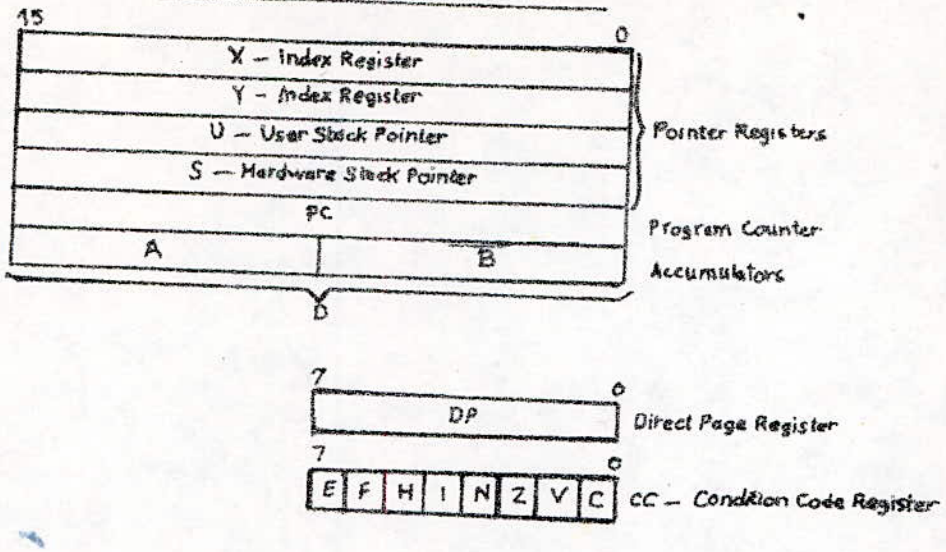
- Deux pointeur de pile U et S :
  * S est le pointeur de pile système superviseur (system-stack) ;
  * U est le pointeur de pile Utilisateur (User stack).

- Un compteur programme P C qui pointe sur l'instruction à effectuer. Ce registre est sur 16 bits puisque la capacité d'adressage du 6809 est de 64 K-octets et fait donc appel à 16 lignes d'adresses.

Les registres de 8 bits sont :

- Le registre de page direct D P R (Direct Page Register) utilisé pour le mode d'adressage direct.

fig 1 REGISTRES DU 6809



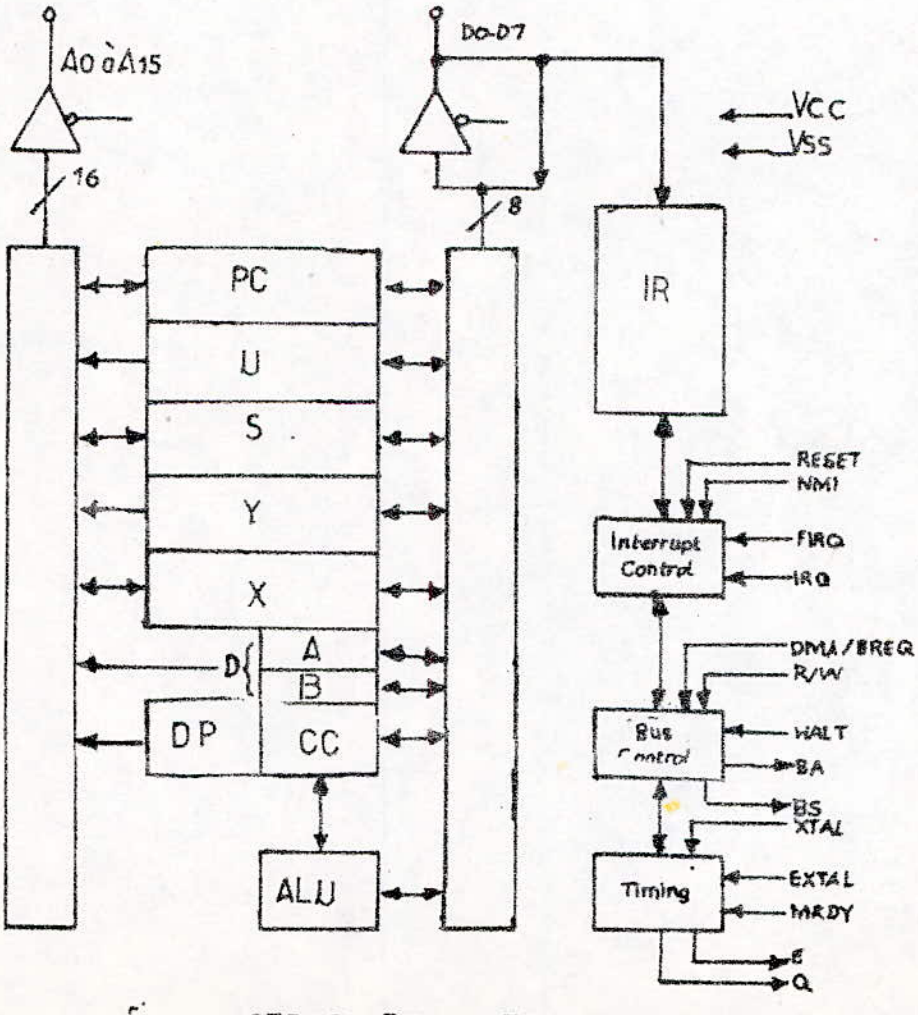


fig 2 STRUCTURE INTERNE DU 6809

- Le registre d'état ou de codes conditions C C R (Condition Code Register) et dont chaque bit a une signification particulière quant à l'état du processeur et du résultat de l'opération qui vient d'être réalisée.

En plus des indicateurs H, I, N, Z, V et C on trouve :

* L'indicateur F ($\overline{\text{FIRQ}}$ mask) :
  C'est le bit pour les interruptions rapides arrivant sur la ligne $\overline{\text{FIRQ}}$. Si F est à 1 les interruptions ne sont pas reconnues. Cet indicateur est positionné également par $\overline{\text{NMI}}$, $\overline{\text{RESET}}$, $\overline{\text{FIRQ}}$, et SWI.

* L'indicateur E (Entire flag) :
  Il nous renseigne sur le nombre de registres rangés dans la pile (variable suivant que l'on a une interruption $\overline{\text{IRQ}}$ ou $\overline{\text{FIRQ}}$). Il est utilisé par l'instruction RTI pour déterminer le nombre d'octets que la pile doit restituer.

Nous verrons lors de la description du jeu d'instructions le rôle de certains de ces bits de façon plus précise.

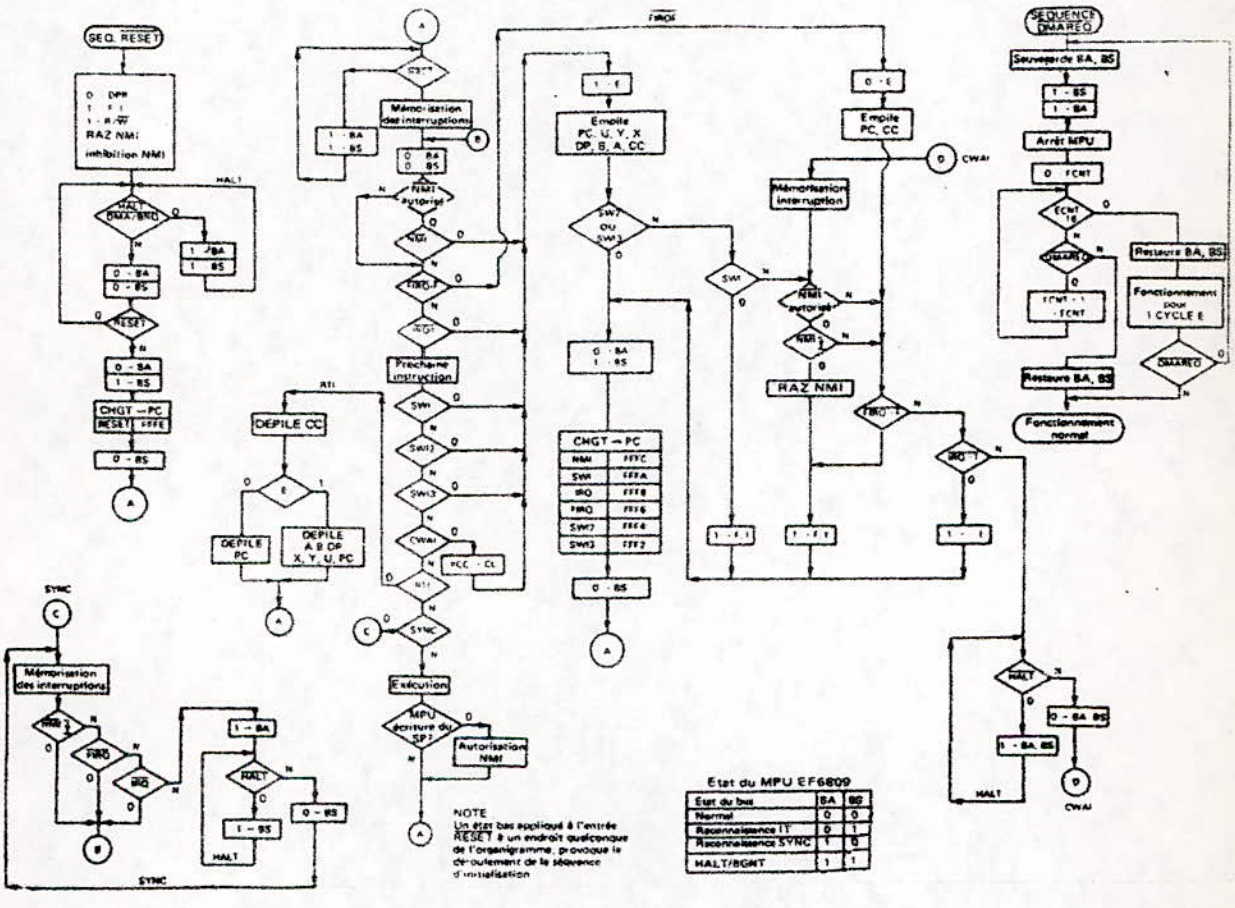
3 - Fonctionnement du M P U :

A) Organigramme de fonctionnement :

En fonctionnement normal, le M P U va chercher une instruction en mémoire puis exécute celle-ci. Cette séquence démarre sur $\overline{\text{RESET}}$ et est répétée indéfiniment sauf si elle est modifiée par une instruction spéciale ou un évènement matériel.

Les instructions logiciels qui modifient le fonctionnement normal du M P U sont : SWI, SWI1, SWI2, CWAI, RTI et SYNC. Une instruction $\overline{\text{HALT}}$ ou $\overline{\text{DMA/BREQ}}$ modifie aussi l'exécution normale des isntructions. La fig. (1.6)

FIGURE 16 – ORGANIGRAMME DU MPU



illustre l'algorithme de fonctionnement du µ P 6809. La moitié gauche de l'organigramme représente un fonctionnement normal ; la moitié droite représente la progression lorsqu'une interruption ou une instruction spéciale survient.

B) INTERRUPTIONS :

a)- Interruptions "HARDWARE" :

Le 6809 dispose de 3 entrées d'interruption HARDWARE

- $\overline{NMI}$: Non Masquable Interrupt.
- $\overline{IRQ}$: Interupt Request
- $\overline{FIRQ}$: Fast Interrupt Request.

Ces 3 interruptions permettent d'avoir 3 vecteurs d'interruption. D'autre part cette vectorisation à 3 niveaux possède une structure de priorité .

$\overline{NMI}$ est prioritaire sur $\overline{FIRQ}$
et $\overline{FIRQ}$ est prioritaire sur $\overline{IRQ}$:

$$\overline{NMI} > \overline{FIRQ} > \overline{IRQ}$$

De plus chaque interruption masque les interruptions moins prioritaires.

Au $\overline{RESET}$ séquentiel $\overline{IRQ}$ et $\overline{FIRQ}$ sont masqués. $\overline{NMI}$ reste masqué tant qu'on a pas initialisé la pile.

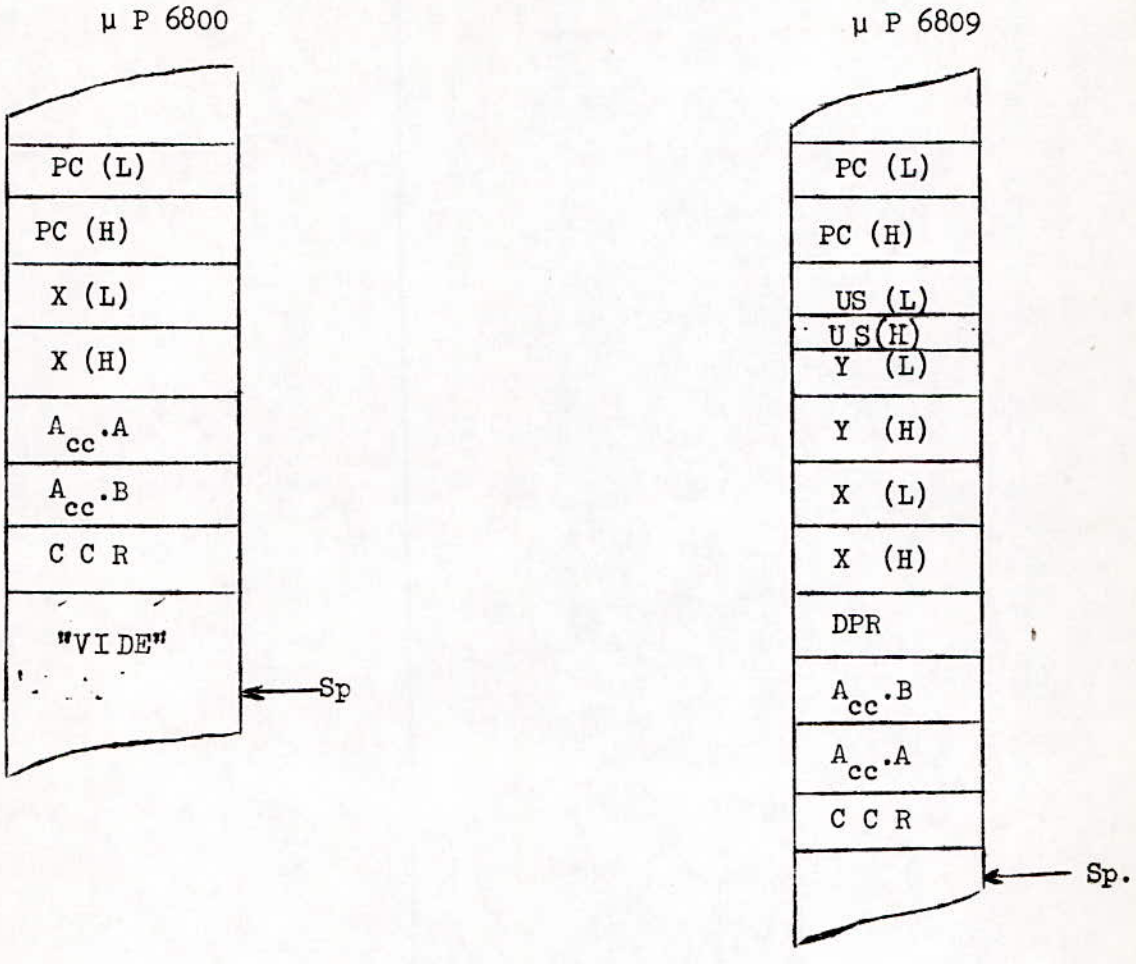
b) - Interruptions "SOFTWARE"

Le 6809 possède 3 interruptions par programmation :

SWI , SWI2 et SWI3.

c) - <u>Sauvegarde de contenu des registres dans la pile</u> :

Différence entre le 6800 et le 6809 :



2ème Partie : S O F T W A R E

A - MODES D'ADRESSAGE

Le µ P 6809 dispose de 10 modes d'adressage dont certains sont extrêmement performants.

On distingue :

1° - Mode d'adressage immédiat ;
2° - Mode d'adressage inhérent ;
3° - Mode d'adressage direct ;
4° - Mode d'adressage étendu ;
5° - Modes d'adressage indexés ;
6° - Mode d'adressage relatif.

On trouvera en annexe B les tableaux donnant les différents modes d'adressage dont dispose le 6809 par comparaison avec le 6800.

1 - Mode d'adressage immédiat :

Le mode d'adressage immédiat permet de placer une valeur dans un registre. On trouve après le code opératoire la valeur à placer dans le registre et non pas une adresse.

Dans le 6809 ce mode est le même qu'avec le 6800, mais on dispose de 4 registres supplémentaires ce qui augmente le nombre d'instructions utilisant ce mode d'adressage.

Exemples :

* LDA ## 03 : Charger A avec la valeur 03
* LDD ## $ 1284 : Charger l'accumulateur D avec la valeur 1284.

Remarque :

Il n'existe pas d'instruction de chargement immédiat du registre pointeur de page direct.

Il faut donc passer par un registre intermédiaire de 8 bits, la séquence à écrire pourra être de la forme suivante :

```
LDA  ##   Valeur
EXG  A , DPR
```

2 - Mode inhérent :

Le mode est le même que dans le 6800. Il est sur un octet et pafois sur deux. Le second est le "POST BYTE"

Exemples:

MNEMONIQUE	CODE MACHINE
* I N C A :	4 C (code opératoire)
* T F R X, Y:	1 F (code opératoire)
	1 2 (Post byte)

3 - Mode d'adressage direct :

Ce mode d'adressage étendu à tout l'espace mémoire est propre au 6809, il peut s'avérer très utile lorsque l'on souhaite réduire l'encombrement mémoire des programmes.

Cet adressage présente l'avantage de ne nécessiter que deux octets pour avoir accès à des données.

Dans le 6800 on ne peut utiliser l'adressage direct que sur les 256 premières cases mémoires (0 à 255), donc l'adressage est limité à la page 0.

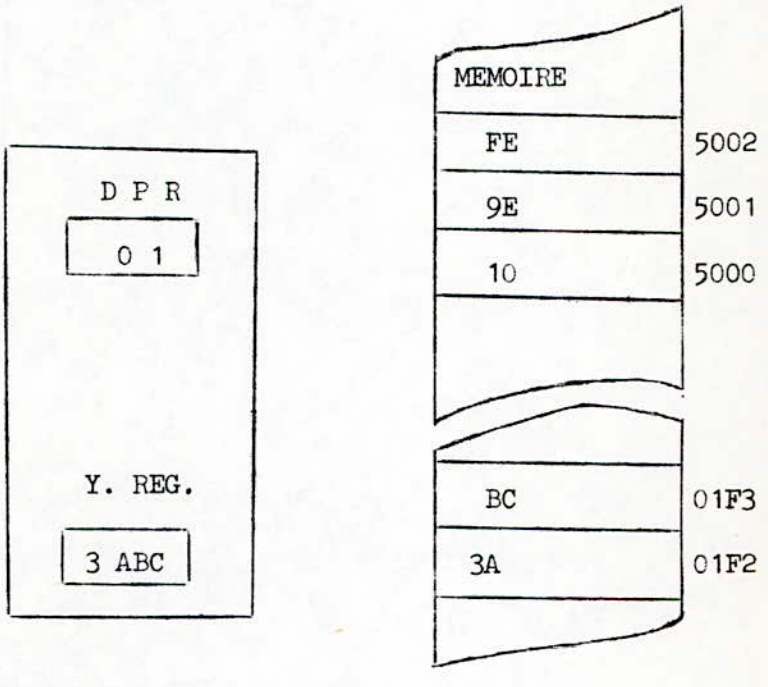
Dans le 6809 un registre de 8 bits a été rajouté. Ce registre contiendra l'octet de poids fort de l'adresse, et on pourra utiliser l'adressage direct sur n'importe quelle page. Il suffira de faire une initialisation du DPR pour pouvoir utiliser tous les avantages de ce mode d'adressage sur une page mémoire.

Le µ P obtiendra l'adresse en prenant l'octet de poids fort dans le DPR et l'octet de poids faible qui se trouvera placé après le code opératoire dans le programme.

A la mise sous tension le DPR est mis à 00, on est donc compatible avec un programme écrit pour un µ P 6800.

Exemple :

* LDY < $ F2



4 - Mode d'Adressage Etendu :

Ce mode d'adressage est identique à celui du 6800, une telle instruction peut tenir sur 3 ou 4 octets.

Dans ce mode l'adresse de l'opérande suit le code opératoire et tient sur 2 octets.

On peut ajouter un niveau d'indirection à ce mode d'adressage, c'est à dire que l'adresse placée après le code opératoire pointe une adresse qui contient l'adresse effective de l'opérande.

Exemples :

* LDA > $ 2A84 - Etendu direct : Charger A par le contenu de la position mémoire 2A84.

* LDA | $ 2A84| - Etendu indirect : Charger A par le contenu d'une position dont l'adresse est contenue dans la position 2A84 .

5 - Modes d'Adressage Indexes :

Avec ce mode d'adressage nous touchons à la famille des modes d'adressage la plus diversifiée et surtout la plus performante du 6809 puisque nous abordons les modes d'adressage indexés. Ce mode qui existe sous une forme simplifiée dans le 6800 (base plus déplacement) prend une autre dimension dans le 6809.

CARACTERISTIQUES PRINCIPALES :

- Le déplacement signé peut être de ± 5,8 ou 16 bits
- Le déplacement peut être le contenu de l'Accumulateur A, B ou D
- Le registre d'index peut être X, Y, U, S ou PC.
- Auto-incrémentation ou décrémentation de 1 ou 2.

Dans tout mode d'adressage indexé, quel qu'il soit, le principe suivant

est appliqué : l'adresse effective où aller chercher la donnée est obtenue en ajoutant une valeur signée, appelée déplacement, à un registre appelé index.

Les différentes options sont sélectionnées par l'octet qui suit le code opératoire (Post byte).

Les bits 5 et 6 sont toujours utilisés pour sélectionner le registre d'index (X, Y, U et S) comme l'indiquent les tableaux donnés en Annexe B.

Le codage de l'instruction est donc un peu particulier, le mot de 8 bits suivant le code opératoire est donc fonction de l'index utilisé et du mode d'adressage indexé choisi.

a) Déplacement constant :

Le premier mode d'adressage indexé est dit à déplacement constant. En effet l'instruction est suivie par un mot de 5,8 ou 16 bits qui représente la valeur du déplacement. Celui-ci permet d'atteindre tout l'espace mémoire du µ P 6809 si l'on choisit de le coder sur 16 bits.

Exemples :

1. Déplacement sur 4 bits :

LDA $ 5,X : Charger A avec le contenu de la position mémoire dont l'adresse est (X) + 5.

2. Déplacement sur 7 bits :

LDY $ 65,X : Charger Y avec le contenu des positions mémoires (X) + 65 (poids faible) et (X) + 66 (poids fort).

3. Déplacement sur 15 bits :

LDA $ 2000,X : l'adresse effective de la donnée est (X) + 2000.

b) Déplacement avec l'accumulateur A, B ou D :

On peut utiliser les accumulateurs A, B ou D comme déplacement dans le mode d'adressage indexé. Dans tout les cas, le contenu de l'accumulateur choisi est ajouté au registre d'index spécifié pour former l'adresse effective.

Exemples :

```
LDA B,X  ; le déplacement est égal au contenu de
           l'accumulateur B, l'adresse effective est
           (B) + (X).
```

c) Indexage par incrémentation/décrémentation automatique :

Le registre utilisé contient l'adresse effective de l'opérande. Après exécution de l'instruction en mode indexé, le registre est incrémenté ou décrémenté de 1 ou 2.

Le choix ± 1 ou ± 2 tient du fait que l'on travaille sur 8 bits, ou sur 16 bits. Dans ce dernier cas on doit aller chercher 2 octets successifs en mémoire.

Cette option permet de sélectionner l'auto-incrémentation/décrémentation de 1 ou 2. L'instruction tient sur deux octets : Code opératoire + Post byte. On précise dans le "post byte" d'une part le registre d'index choisi et d'autre part le mode décrémentation/incrémentation. Cet indexage permet, aisément, de travailler sur des tables.

* LDA 0,X$^{+\ +}$: mettre dans A le contenu de la position mémoire dont l'adresse est le contenu de X et incrémenter X de 2.

* STA 0,X$^{-\ -}$: Stocker A dans la position mémoire d'adresse X et décrémenter X de 2.

d) Indexé Indirect :

Mis à part le mode d'auto-incrémentation/décrémentation par 1 et le mode utilisant un déplacement constant ± 4 bits, tous les modes d'adressage indexés peuvent être utilisés avec un niveau d'indirection. Ainsi la valeur obtenue en ajoutant la valeur du déplacement au contenu du registre d'index choisi pointe sur une case mémoire qui contient l'adresse de la donnée. Le bit 4 du "Post byte" est utilisé pour sélectionner le mode indexé indirect.

Le nombre d'octets pour une instruction donnée est le même que l'on soit dans le mode d'adressage indexé ou indexé indirect.

Exemple :

LDA |B, X| : Charger A avec le contenu de la mémoire dont l'adresse est le contenu de la position mémoire d'adresse (B) + (X).

6 - Mode d'Adressage Relatif :

Le mode d'adressage relatif diffère de celui du 6800 sur deux points :

1° - Le déplacement (en complément à 2) peut être soit de ± 7 bits soit de ± 15 bits. On peut donc réaliser un branchement dans dans tout l'espace mémoire (d'où la création de programmes translatables)

2° - Le mode d'adressage relatif ne se limite pas aux branchements.

Exemple :

2015	LDA $	1FDA,PC
2015	A6	Code Opératoire
2016	8C	Post byte
2017	C2	Déplacement
1FDA		Donnée.

Cette instruction charge l'accumulateur A du contenu d'une position dont l'adresse est donnée par le déplacement C2 (complément à 2) par rapport à l'adresse contenue dans le compteur programme.

Mode d'adressage relatif indirect / PC :

Ce mode d'adressage est le plus impressionnant du µ P 6809, il tient à la fois de l'adressage indexé et de l'adressage relatif. En effet ce mode est l'indexé indirect où le registre d'index utilisé est le compteur de programme P C.

Dans ce mode d'adressage ne figure aucune notion d'adresse absolue puisque tout est référencé par rapport au PC au moyen de déplacement, c'est donc un adressage relatif ; Il signifie que les programmes écrits en utilisant exclusivement ce mode d'adressage pourront tourner n'importe où en mémoire puisqu'il n'est fait nulle part référence à une adresse fixe ; nous pouvons donc écrire des programmes translatables.

Exemple :

```
2015     LDA | $ 1FDA, PCR|
  *          *
2015    A6          Code opératoire
2016    9C          Post byte
2017    C2          Déplacement
2018                Instruction suivante
  *          *
1FDA    01
1FDB    00 }        Nouvelle adresse effective
  *          *
0100    [    ]      donnée
```

Cette instruction met dans A le contenu de la case mémoire d'adresse 0100.

Le déplacement sur 1 octet est calculé de la manière suivante : A la fin de l'instruction le compteur de programme pointe l'instruction suivante et il a donc la valeur 2018.

Le déplacement vaut donc : 2018 + déplacement = 1FDA i.e déplacement = FFC2 . Dans la pratique on ne le code que sur un octet : FFC2 s'écrira simplement C2.

B - JEU D'INSTRUCTIONS DU MICROPROCESSEUR MC 6809

P L A N-

I - Jeu d'instructions :

1. Instructions sur les transferts de données ;
2. Instructions sur les mouvements de données ;
3. Instructions sur les modifications de données ;
4. Instructions arithmétiques ;
5. Instructions de chargement d'adresses effectives ;
6. Instructions de branchements ;
7. Instructions sur les tests de données ;
8. Instructions logiques ;
9. Instructions spéciales ;
10. Instructions sur les interruptions.

II - Explications de quelques instructions :

- Instructions de synchronisation : SYNC
- Chargement de l'adresse effective
- Instructions de transfert et d'échange de registres
- Instructions de rangement et recherche de registre sur la pile.
- Positionnement des bits des codes conditions.
- Instructions de long branchement.

III - Annexe C :

- Liste comparative des instructions 6809 - 6800
- Table d'équivalence entre certaines instructions
- Codes opératoires du MC 6809.

I/ JEU D'INSTRUCTIONS :

Si l'on regarde le nombre d'instructions dont dispose les µ P , on peut être surpris de constater qu'au fur et à mesure que ceux ci deviennent plus performants leur nombre d'instructions se réduit. Ainsi le 6800 dispose de 72 instructions de base et le 6809 n'en a que 59.

Lors de la naissance des µ P ceux ci étaient issus de schémas en logique cablé et disposaient d'un certain nombre de registres très spécialisés. De ce fait ce qui se faisait avec un registre de données ne pourrait pas forcément se faire avec un autre , ceci compliquerait sérieusement la vie des programmeurs qui étaient obligés d'apprendre un nombre assez grand d'instructions et de MNEMONIIQUES.

Les µ P actuels tendent à se libérer de ces contraintes et réduisent donc leur jeu d'instructions en banalisant au maximum leurs registres. Ainsi un maximum d'instructions peuvent utiliser un maximum de modes d'adressage d'où la création de la notion microprocesseur à structure orthogonale.

Nous allons voir que le 6809 s'en approche d'assez près quoi que cette banalisation est limitée par le fait qu'il est un microprocesseur "hybride".

On trouvera en annexe C le jeu complet des instructions du 6809 et toutes les caractéristiques standards mises à part celles qui lui sont spécifiques. Celles-ci seront étudiées un peu plus en détail.

II/ - EXPLICATION DE CERTAINES INSTRUCTIONS :

* Instruction de synchronisation : SYNC :

A/ FONCTIONNEMENT :

Cette instruction permet de synchroniser un programme avec un événement extérieur.

Lorsque le µ P a exécuté cette instruction il s'arrête. Le µ P restera dans cet état tant qu'une interruption ne sera pas reçue. Lorsque l'interruption est reçue 2 cas peuvent se présenter :

1er Cas : Si l'interruption est permise, le µ P exécutera normalement le programme d'interruption.

2è Cas : Si l'interruption est masquée, le µ P continue l'exécution du programme sans traiter le sous-programme d'interruption.

B/ Signaux :

Lorsque le MPU est dans le mode SYNC le signal "BUS AVAILABLE" est à un et le signal "BUS STATUS" est à 0.

Voir en annexe A le chronogramme du signal SYNC.

C/ Utilisation :

Cette instruction permet de réaliser une synchronisation rapide avec des périphériques extérieurs.

Exemple :

Dans le cas où l'interruption est masquée :

```
FAST    SYNC     Attendre l'interruption
        LDA , X  Lire la donnée sur le périphérique
        STA, Y+  ranger la donnée et incrémenter l'adresse
        DECB     FINI ?
        BNE   FAST   NON   RECOMMENCER.
```

REMARQUE :

Cette instruction est similaire à l'instruction (CWAI) mais les registres ne sont pas sauvegardés dans la pile.
Cette synchronisation peut permettre grâce à sa rapidité d'éviter l'utilisation de DMA dans des cas particuliers.

* CHARGEMENT DE L'ADRESSE EFFECTIVE : LEA :

Le 6809 possède des instructions qui permettent de charger une adresse effective dans les registres X, Y, S, U (LOAD EFFECTIVE ADRESS.)

Exemples :

```
LEAX - 1, X  :  Décrémenter X
LEAY 5, X :  On place dans Y le contenu de X plus 5
LEAX 1, X          INCREMENTER X
LEAXB , X          (B) + (X)  →  X
```

1°/ Cette instruction permet d'incrémenter, de décrémenter et d'ajouter des valeurs aux registres X, Y, S, U

2°/ On peut ajouter des données se trouvant dans les accumulateurs à un pointeur : LEAX B , X (B) + (X) → X .

3°/ L'utilisation d'une instruction du genre LEAS-8,S au début d'un sous-programme nous permet de travailler dans une zone où l'on est sûr dene pas détruire d'autres variables dans son programme.

4°/ On peut avoir accès à des tables placées dans des programmes translatables en utilisant le mode d'adressage PC relatif.

LEAX TABLE , PCR

* ECHANGE ET TRANSFERT ENTRE REGISTRE (EXG , TFR) :

Dans ces 2 types d'instruction l'octet qui suit le code opératoire précise le registre d'origine et le registre de destination.

Une seule restriction : Les deux registres doivent avoir la mêre taille .

Exemple : EXG X , Y

Remarque :

Ces 2 instructions n'affectent pas les drapeaux du C.C.P par conséquent l'instruction TAB du 6800 s'écrit avec le 6809 :

TAB ⟺ TFR B ,A (transfert)
TST (positionnement codes).

* INSTRUCTIONS DE RANGEMENT OU DE RECHERCHE DE REGISTRES SUR LA PILE :

On peut ranger ou prendre le contenu des registres de la pile.
Le 6809 possède de puissantes instructions de "PUSHING" (Rangement dans la pile) et de "PULLING" (Extraction de la pile).
Le "Post byte" précise les registres concernés.

Exemples : PULS A, B, X .

Remarques :

1 - Cette puissance dans la manipulation de la pile permet d'écrire aisément des programmes réentrants.

2 - Ces instructions facilitent le passage d'arguments entre un programme principal et des sous-programmes ; ce qui évite de placer un RTS à la fin du sous-programme.

3 - L'ordre de rangement dans la pile est une fonction "HARDWARE" qui range les registres dans un ordre fixe.

* POSITIONNEMENT DES BITS DU REGISTRE DES CODES CONDITIONS :

Les instructions SEC, CLC, SEV, CLV, SEI, CLI ont été remplacées par deux instructions sur 2 octets, elles permettent de mettre à 1 ou à 0 simultanément plusieurs bits.

Ces deux instructions sont les suivantes :

1°) - ORCC ## combinaison binaire :

Cette fonction réalise un "OU" logique entre le CCR et la combinaison binaire, le résultat est placé dans le CCR.

Exemple : ORCC ## 01 .

2°/ - ANDCC ## combinaison binaire :

Identique à la précédente mais c'est un "ET" logique qui est réalisé.

Exemple : ANDCC ## Ø F E .

* INSTRUCTION DE BRANCHEMENT : LONG BRANCHEMENT :

En plus des instructions de branchements que possède le 6800 , le 6809 offre la possibilité de branchement sur tout l'espace mémoire avec des instructions de "Long BRANCH".

Exemple : LBEQ , LBRA ,...

CHAPITRE III

CARTE MICRO-ORDINATEUR MONOCARTE

INTRODUCTION

Le domaine privilégié d'application du µ P 6809 prend tout son intérêt dans un système de développement. Cependant le moniteur de base doit être très évolué pour profiter au maximum des performances et des capacités de traitement que nous offre ce microprocesseur de la 3ème génération.

Le synoptique d'un tel système peut être composé, comme l'indique la figure 3,1 , de :

- Mémoires mortes (ROM) qui contiendraient le moniteur et les différents programmes de gestion et d'aide à la programmation.

- Mémoires vives (RAM) qui peuvent être utilisées par le moniteur ou le programmeur pour des extensions ultérieures ou simplement pour l'exécution des programmes.

- Un interface parallèle (P I A) MC 6820/6821

- Un interface série asynchrone (A C I A) MC 6850

- Un temporisateur programmable ou P T M (Triple Timmer Programmable) MC 6840.

Ce circuit de la famille 6800 peut être utilisé comme :

* Générateur d'impulsion ;
* Générateur de signaux périodiques ;
* Compteur ; fréquencemètre, etc...

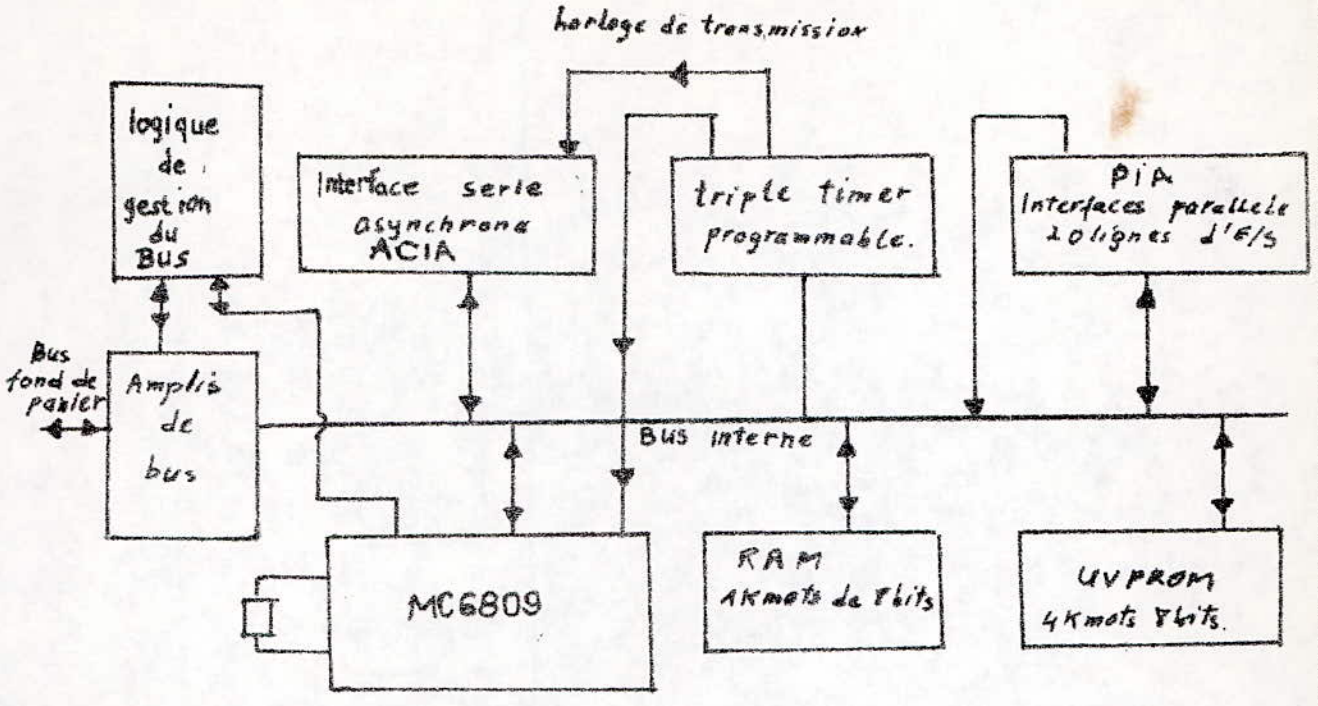


fig 3_1 SYNOPTIQUE DE LA CARTE MICROORDINATEUR CPU09

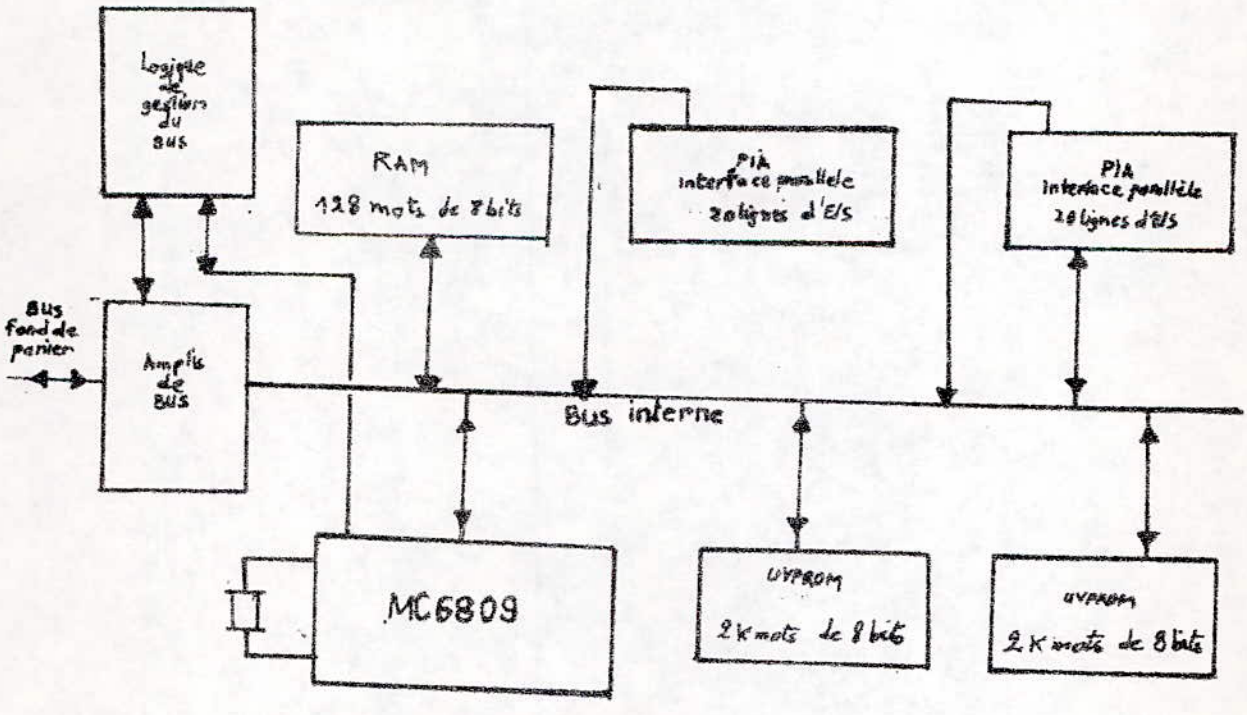


fig 3_2 SYNOPTIQUE DU MICROORDINATEUR REALISE

Il comprend essentiellement 3 compteurs à 16 bits qui peuvent fonctionner simultanément, et générer par programmes des signaux par ses sorties O_1 , O_2 et O_3 . Ce qui donne beaucoup de souplesse à ce circuit. L'une de ces sorties peut être utilisée par l'A C I A comme horloge de synchronisation pour définir la vitesse de transmission.

- Une logique pour le décodage d'adresses qui peut être basé autour d'un démultiplexeur ou une PROM. L'étude de ces deux types de décodage est faite ultérieurement.

- Des amplificateurs de données, d'adresses et des signaux de contrôle (Buffers).

Pour un tel système, le décodage d'adresse dépend de l'espace mémoire adopté et des circuits adressables indispensables au système. Il est étroitement lié au logiciel (outil de développement proposé par les constructeurs ; Nous verrons dans la partie décodage d'adresses que le choix judicieux du type de décodage peut surmonter les problèmes éventuels qui peuvent surgir.

(A) Organisation du système :

Compte tenu de l'application envisagée (VOIR CHAP.IV) la configuration retenue pour réaliser notre micro-ordinateur centré sur le microprocesseur MC 6809, dont le synoptique est donné par la figure 3.2, est la suivante :

- Le MPU avec toute la logique de contrôle ;
- Le décodeur d'adresses ;
- Une EPROM qui contiendrait les programmes de gestion du système ;
- Une RAM ;
- Deux interfaces parallèle : P I A.

Vu la taille du programme nous avons dû utiliser une EPROM MCM 2716 de 2048 x 8 bits. Par contre une RAM MCM 6810 de 128 x 8 bits s'est avérée suffisante.

Nous n'avons pas prévu d'A C I A de Timer. Ces composants n'étant pas indispensables pour notre application. Mais nous avons mis deux P I A à cause des nombreuses commandes à réaliser.

(B) Structure du système :

Présentation du micro-ordinateur monocarte :

Notre carte est organisé autour du MC 6809 dont l'étude est faite dans le CHAPITRE II.

Elle comporte un bus interne qui permet la liaison entre les divers circuits qui la composent ; une RAM, une ROM , deux circuits d'interface parallèle.

Par ailleurs une logique de gestion de bus et un ensemble d'amplificateurs trois états permettent de relier le bus interne au bus de fond de pannier.

(1) Signaux de la carte :

Le C P U présente 16 lignes d'adresses unidirectionnelles notées de A0 à A15, 8 lignes de données D0 à D7 bidirectionnelles à 3 états. On a aussi :

- Deux lignes d'alimentation : la masse V_{ss} et l'alimentation + V_{cc} égale à + 5 V.

- Une ligne R/$\bar{W}$ ou lecture-écriture amplifiée qui a la faculté de passer dans le troisième état pour agir directement sur la commande d'activation du sens de transfert dans les buffers bidirectionnels : 8T26.

- Une ligne $\overline{\text{RESET}}$ ou remise à zéro :

  Cette ligne permet d'initialiser tout le système .
  Un RESET automatique est prévu pour cette carte : A la mise sous tension cette ligne doit être maintenue à l'état bas jusqu'à ce que l'oscillateur d'horloge ait atteint un régime de fonctionnement normal. A cet effet on utilise un réseau R C puisque l'entrée $\overline{\text{RESET}}$ du MC 6809 possède un Trigger de Schmitt ayant une tension de seuil supérieure à celle des interfaces standards.

- Deux lignes E (ϕ_2) et Q(ϕ_1) qui sont utilisées par tous les éléments du système pour pouvoir échanger des données avec le 6809. Ce sont ces signaux qui régissent le séquencement de tous ces échanges.

Les adresses du M P U sont validées sur le front montant de Q, les données sont stables sur le front descendant de E.

D'après le diagramme de temps de E et Q le signal V M A (VALABLE MEMORY ADRESS) nécessaire à la commande des autres circuits est la combinaison de E, Q et B A (BUS AVAILABLE) qui indique l'état du bus. Deux portes NOR suffisent pour générer ce signal à partir de E, Q et B A (VOIR SCHEMA et Table de vérité figure 3,3).

- CIRCUIT DE RAFRAICHISSEMENT :

RAM dynamiques :

La limitation d'intégration des RAM statiques de grande capacité a poussé la création des RAM dynamiques. L'élément mémoire est la capacité de gachette - source d'un transistor MOS, mais cette capacité perd sa charge et il faut la rafraichir toutes les 2 ms environ.

Signaux de commande :

* La demande de rafraichissement est formulée par le signal $\overline{\text{R R}}$

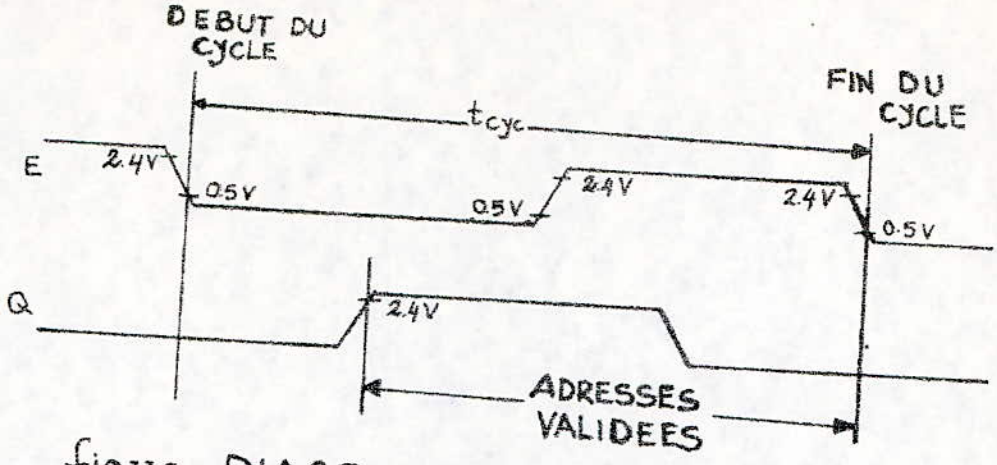


fig3.3a - DIAGRAMMES DES TEMPS DE E et Q

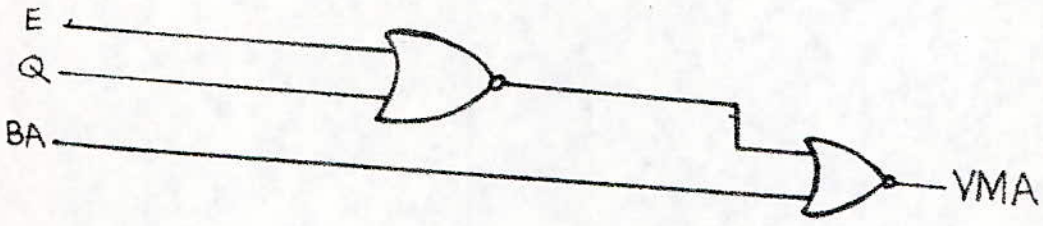


fig3.3b.-SIGNAL VMA

E	Q	BA	VMA
0	0	X	0
0	1	0	1
1	0	0	1
1	1	0	1
X	X	1	0

X : "0" ou "1"

fig:3.3.c-TABLE DE VERITE DU SIGNAL VMA

(Refresh Request) qui est synchronisé avec l'horloge ϕ_2 au moyen d'une bascule D pour attaquer l'entrée D M A du MC 6809.

* Autorisation de rafraichissement : (REFRESH GRANT)
L'état du bus B S, codé avec B A représentant l'état du microprocesseur est synchronisé avec Q et donne au moyen d'une bascule D l'autorisation de rafraissement (R G), en effet le rafraichissement n'a lieu qu'à l'état bas de Q.

- Les lignes d'interruption :

Les lignes $\overline{I R Q}$, $\overline{F I R Q}$ et $\overline{N M I}$ étant actives à l'état bas, nous les avons reliées, de la même façon que $\overline{HALT}$ et $\overline{MEDY}$ au + 5 V par des résistances de rappel pour qu'elles ne soient pas activées accidentellement par un parasite.

- Amplification de bus :

Le microprocesseur MC 6809 utilisé dans le système possède au niveau de ses lignes de sorties (données, adresses et contrôle) une sortance généralement faible. Pour qu'elle ne soit pas dépassée on fait passer tous les signaux issus du MC 6809 par des circuits amplificateurs ou buffers.

Ces buffers sont des circuits logiques inverseurs ou non inverseurs dont l'entrance est faible et la sortance très élevée. Ils ont également pour rôle d'isoler le microprocesseur lorsqu'ils sont en haute impédance ; ceci est nécessaire pour effectuer de l'Accès Direct Mémoire (D M A)

Les buffers utilisés sont de deux types :

- Version Unidirectionnelle (8T95) pour les lignes de contrôle et d'adresses.
- Version bidirectionnelle (8T26) pour les lignes de données puisque ces dernières transitent dans les deux sens.

Les lignes de contrôle qui passent dans des amplis unidirectionnels non inverseurs sont validées en permanence. Quant à l'activation des buffers d'adresses et du signal R/$\overline{W}$ elle fait intervenir le signal R G.

Les sorties de données des mémoires sont toutes reliées entre elles, ce qui est possible grâce au décodage d'adresses : Une seule mémoire est validée à un instant donné. Ces lignes de données aboutissent ensuite sur des amplificateurs inverseurs bidirectionnels. L'entrée de validation de ceux-ci reçoit le signal R/$\overline{W}$ qui détermine le sens de transfert.

On trouve dans la page suivante le brochage des 8T26 ainsi que la logique de commande de ceux-ci et la table de vérité correspondante.

Le signal Refresh Grant (R G) permet de faire passer à l'état haute impédance (3ème état) les buffers d'adresses et de données ce qui protège le microprocesseur des court-circuits accidentels et permet de travailler en multiprocesseurs.

(2) Décodage d'adresses :

INTRODUCTION

Dans un système d'informatique figé, les adresses des cartes sont fixées une fois pour toutes lors de la conception du montage..

Lorsque le C P U place une adresse de 16 bits sur le bus d'adressage elle ne doit activer qu'une position mémoire ou un interface d'E/S ; Le circuit de décodage dépend du type de boitiers utilisés.

Les décodeurs permettent d'acheminer les signaux de sélection vers les différents boitiers de façon telle que le boitier ou l'ensemble des boitiers qui contiennent le mot désiré soit actif.

MC 8T26

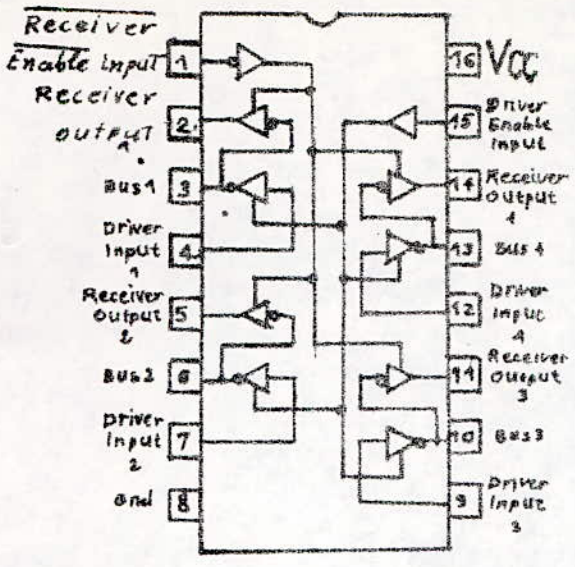
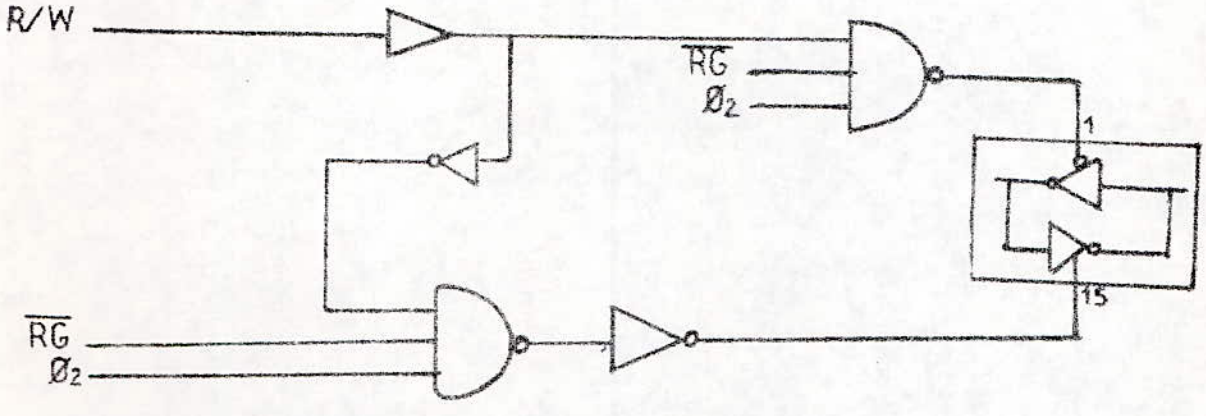


Table de verite de commande des 8T26

R/W	$\emptyset_2$	RG	PIN 1 8T26	PIN 15 8T26
0	0	0	1*	1*
0	1	0	1*	1*
1	0	0	1*	0*
1	1	0	0*	0*
X	X	1	HI*	HI*

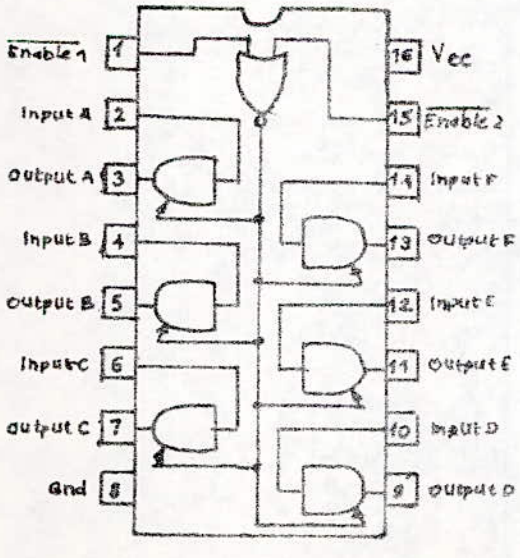
* HI : Haute Impedance

• Lecture ou ecriture : Transfert de donnees

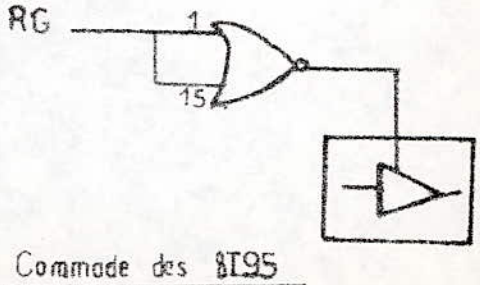


Logique de commande des buffers de donnees

XC8T95



RG	ETAT DU BUS
0	VALIDE
1	HAUTE IMPEDANCE



Commade des 8T95

Nous allons étudier les différents circuits de décodage qui permettent la sélection d'une mémoire ou d'un périphérique auquel le CPU cherche à accéder. Il y ' a plusieurs approches qui sont, en principe,similaires.

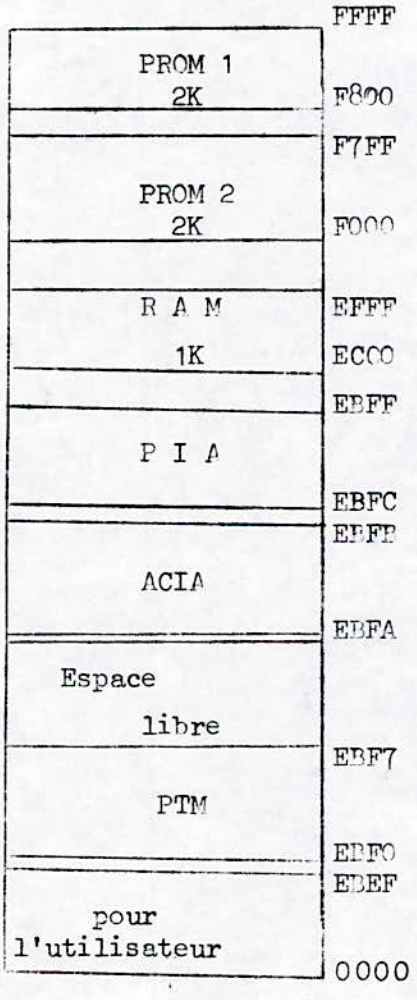
Plutôt que de montrer plusieurs types de décodage, nous avons opté pour développer deux méthodes dont les principes et les idées sont appréciables.

Les deux types de décodage que nous avons utilisés sont :

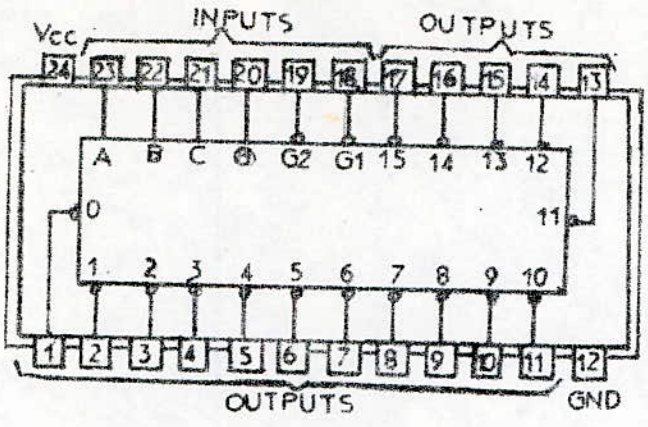
- Le décodage par des techniques classiques, à savoir : le démultiplexage.
- Le décodage par PROM dont les avantages ne sont pas à sous-estimer.

A) DECODAGE PAR MULTIPLEXAGE:

L'espace mémoire adopté est le suivant :



DEMULTIPLEXEUR



SN54154

3.4 TABLE DE FONCTION

INPUTS					OUTPUTS													
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

Principe :

Le démultiplexeur 74154 va servir à décoder les différents boitiers dont les 3 lignes A_{13} , A_{14} , A_{15} sont à l'état haut, d'où leur utilisation pour activer le démultiplexeur en synchronisme avec V M A.

Les lignes A_{12} , A_{11} et A_{10} sont à décoder, elles attaquent respectivement les entrées D C B du décodeur. L'entrée A est attaquée par la combinaison des lignes de A 9 à A 4.

Le tableau 3.4 montre que l'état actif du décodeur 74154 est l'état bas.

Décodage des circuits :

Pour différencier les circuits PIA, ACIA et PTM on utilise les lignes d'adresses A_1 , A_2 et A_3 pour attaquer les autres "select chips".

Le tableau 3.5 donne le décodage des circuits.

CIRCUIT	ADRESSES	COMBINAISON D C B A	SORTIES CORRESPONDANTES
PROM 1	F800 - FFFF	1 1 X X	15 , 14 , 13 , 12
PROM 2	F000 - F7FF	1 0 X X	11 , 10 , 9 , 8
RAM	EC00 - EFFF	0 1 1 X	7 , 6
P I A	EBFC - EBFF	0 1 0 0	4
ACIA	EBFA - EBFB	0 1 0 0	4
P T M	EBF0 - EBF7	0 1 0 0	4

Tableau 3.5

Les sorties correspondantes aux PROM et RAM sont grcupées à l'aide des portes logiques NAND puis inversées pour attaquer les $\overline{C S}$ de ces boitiers.

Les sorties sont maintenues à l'état haut au repos à l'aide de résistances de rappel.

B) DECODAGE PAR PROM :

Le décodage par PROM permet de réduire fortement le nombre de composants utilisés dans la circuiterie de décodage tout en autorisant une évolution ultérieure du montage, puisqu'il est toujours possible de remplacer une PROM par une autre ayant un contenu different sans avoir à changer quoique ce soit au circuit imprimé qui le supporte, ce qui ne serait pas le cas avec un décodage à logique cablée.

- Principe

Les lignes d'adresses à décoder (de A_{11} à A_4) issues du bus du système aboutissent respectivement sur les entrées adresses de la PROM (A9 à A0) et les sorties DATA de la PROM aboutissent aux "select chips" des circuits à décoder.

La programmation de la mémoire utilisée dépend de l'espace mémoire adopté pour le système.

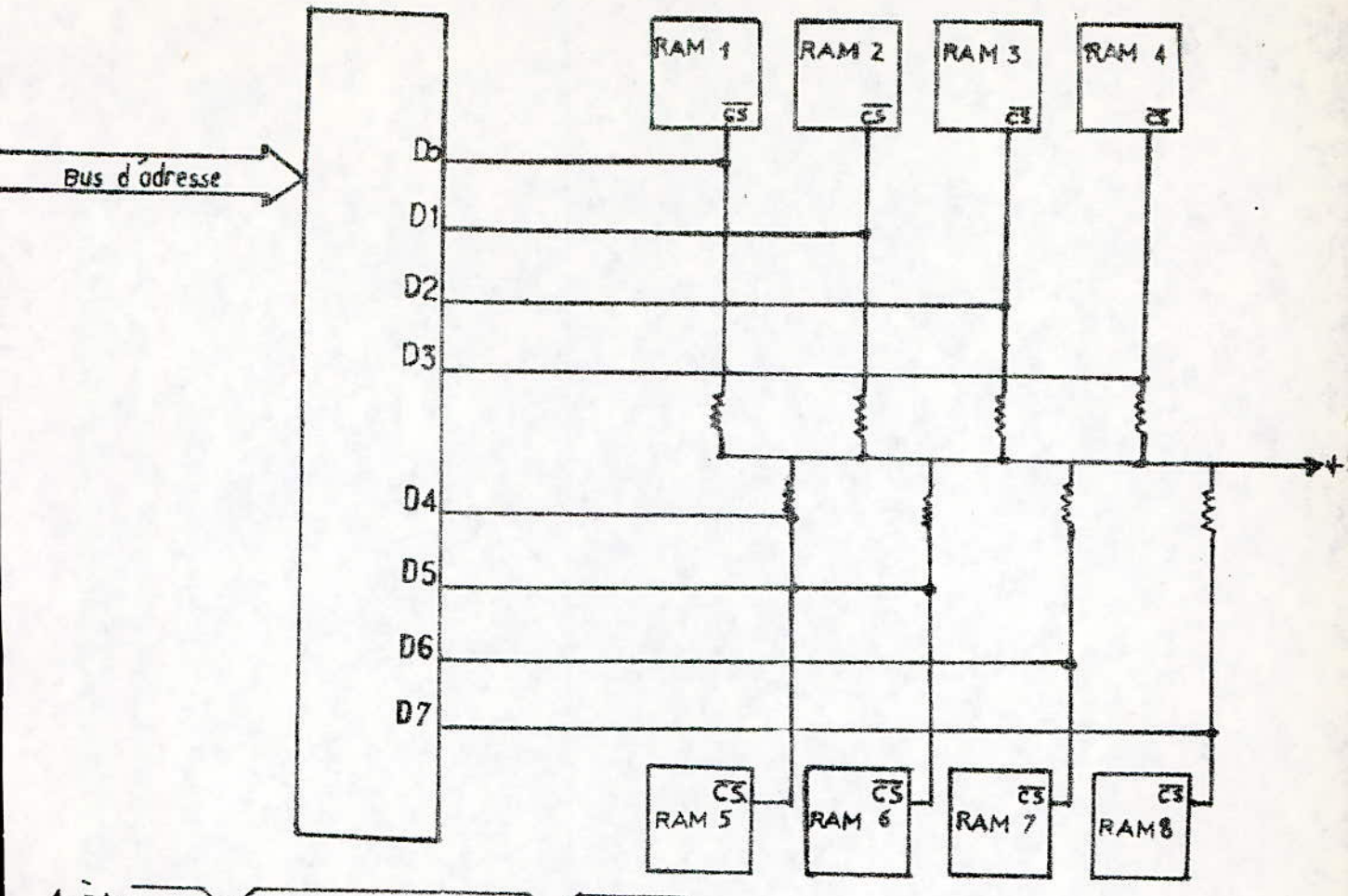
Examinons le schéma présenté figure 3,6 nous y voyons une carte mémoire hypothétique qui peut supporter 8 boitiers classés suivant l'espace mémoire ci-dessous compatible avec celui de l'EXORCISER.

La PROM de décodage reçoit sur sa PIN de validation $\overline{G}$ le signal VMA et les lignes d'adresses A_{15} , A_{14} , A_{13} , A_{12} . Cela permet de n'activer la PROM que lorsque les adresses présentées sur le bus sont valideset incluses dans l'espace mémoire suivant : F000 - FFFF.

* Programmation de la PROM :

Regardons maintenant le tableau 3,6 : Il nous indique le contenu de chacune des 2024 positions mémoire de la PROM qu'on a utilisé (MCM 2716)

Principe de decodage d'adresse par PROM



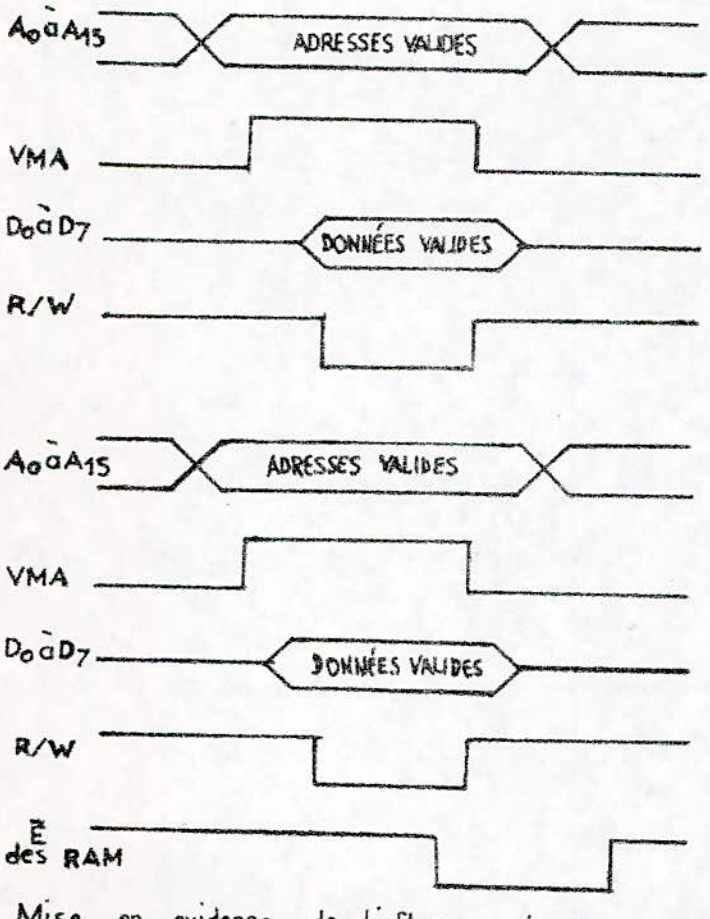


fig 3.6 b

Mise en evidence de l'influence du temps necessaire au décodage d'adresse

Nous voyons sur l'étendue de ses adresses allant de 000 à 1FF que toutes ses sorties sont à "1" sauf D0 : cela signifie que lorsque l'adresse voulue se présente sur cette fourchette sa sortie D0 passe à 0 validant ainsi la PROM 1 de 2 K.

Si nous continuons l'examen du tableau 3,6 nous constatons que de 340 à 3BF toutes ses sorties sont à "1". Ce qui signifie que les mémoires auquelles ces sorties sont reliées ne sont pas activées puisqu'elles dispo - sent d'entrées de validation actives au niveau bas.

TABLEAU 3.6 : - PROGRAMMATION DE LA PROM

ADRESSES PROM	CONTENU EN BINAIRE								CONTENU EN HEXA.	ADRESSE MEMOIRE VALIDEE. ESPACE MEMOIRE EXORCISER
	D7	D6	D5	D4	D3	D2	D1	D0		
000 - 1FF	1	1	1	1	1	1	1	0	F E	F000-F7FF PROM 2
200 - 2FF	1	1	1	1	1	1	0	1	F D	F800-FBFF PROM1
300 - 33A	1	1	1	1	1	1	1	1	F F	NEANT
33B - 33C	1	1	1	1	1	0	1	1	F B	FCEC-FCF3 PTM
33D	1	1	1	1	0	1	1	1	F 7	FCF4-FCF7 ACIA
33E	1	1	1	0	1	1	1	1	E F	FCF8-FCFB PIA
33F	1	1	0	1	1	1	1	1	D F	FCFC-FCFF PROM
340 - 3BF	1	1	1	1	1	1	1	1	F F	NEANT
3C0 - 3DF	1	0	1	1	1	1	1	1	B F	FF00-FF7F RAM2
3ED - 3FF	0	1	1	1	1	1	1	1	7 F	FF80-FFFF RAM1

On peut continuer ainsi le raisonnement et vérifier le contenu du tableau comparativement aux indications d'adresses données.

Ce petit esposé permet de comprendre le principe du décodage d'adresses par PROM, pour compléter cette étude nous allons présenter ci-dessous les avantages qu'on peut trouver à l'utilisation de ce mode de décodage.

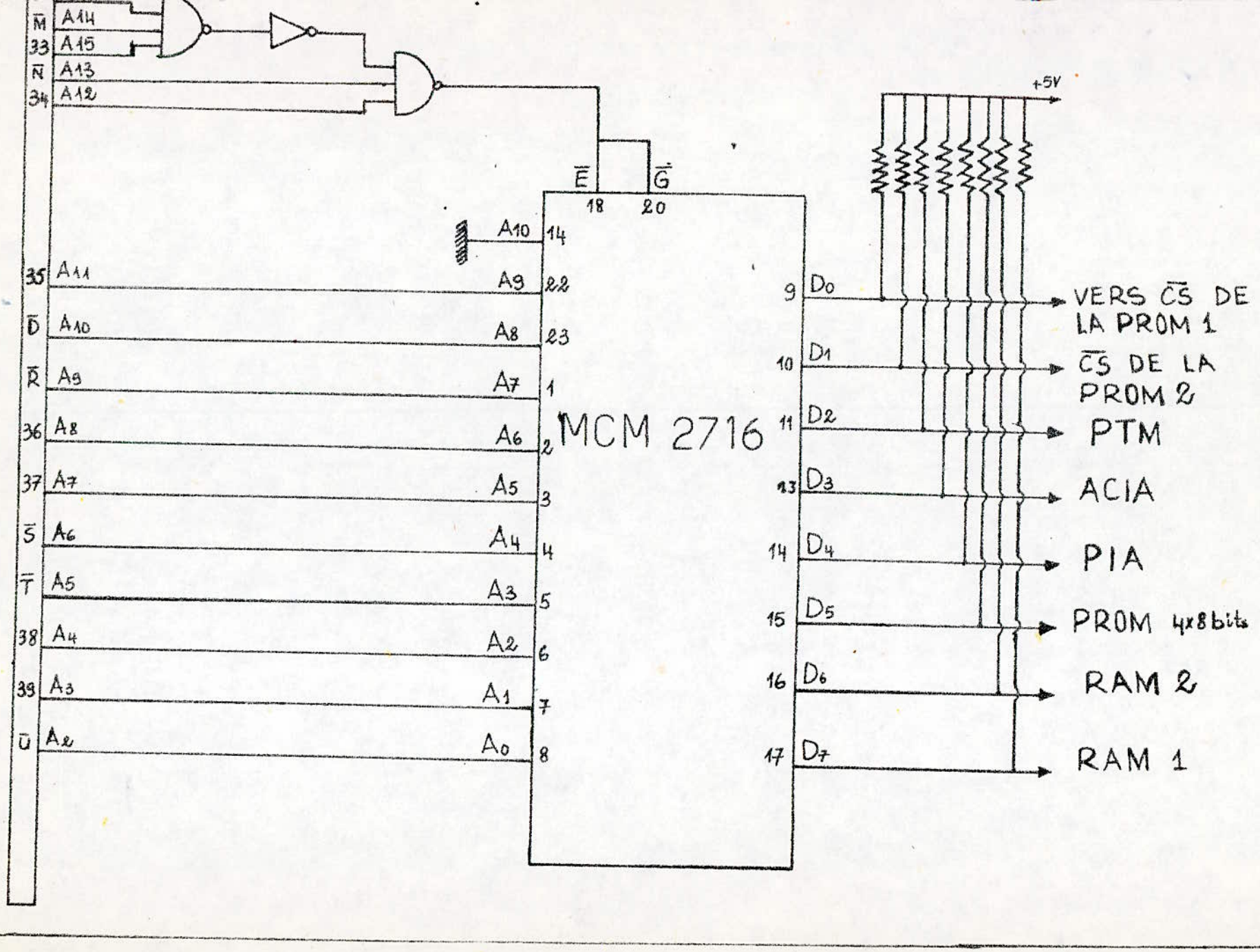
- Réduction du nombre total de boitiers ;
- Simplification du circuit imprimé ;
- Simplification du travail d'étude de la carte puisque une partie de la logique se trouve ainsi réduite à une programmation de mémoires.
- Possibilités d'évolution ultérieure du système sans avoir à repenser le circuit imprimé puisqu'il suffit de changer le contenu de la PROM pour modifier la configuration des adresses, ce qui est impossible avec la logique cablée.

Cette méthode n'échappe pas à quelques inconvénients qui dépendent du type de mémoire utilisée. En effet le paramètre fondamental qui caractérise celle-ci est le temps d'accès relativement faible pour les PROM bipolaires mais élevé (de l'ordre de 350 à 450 ns) pour les EPROM.

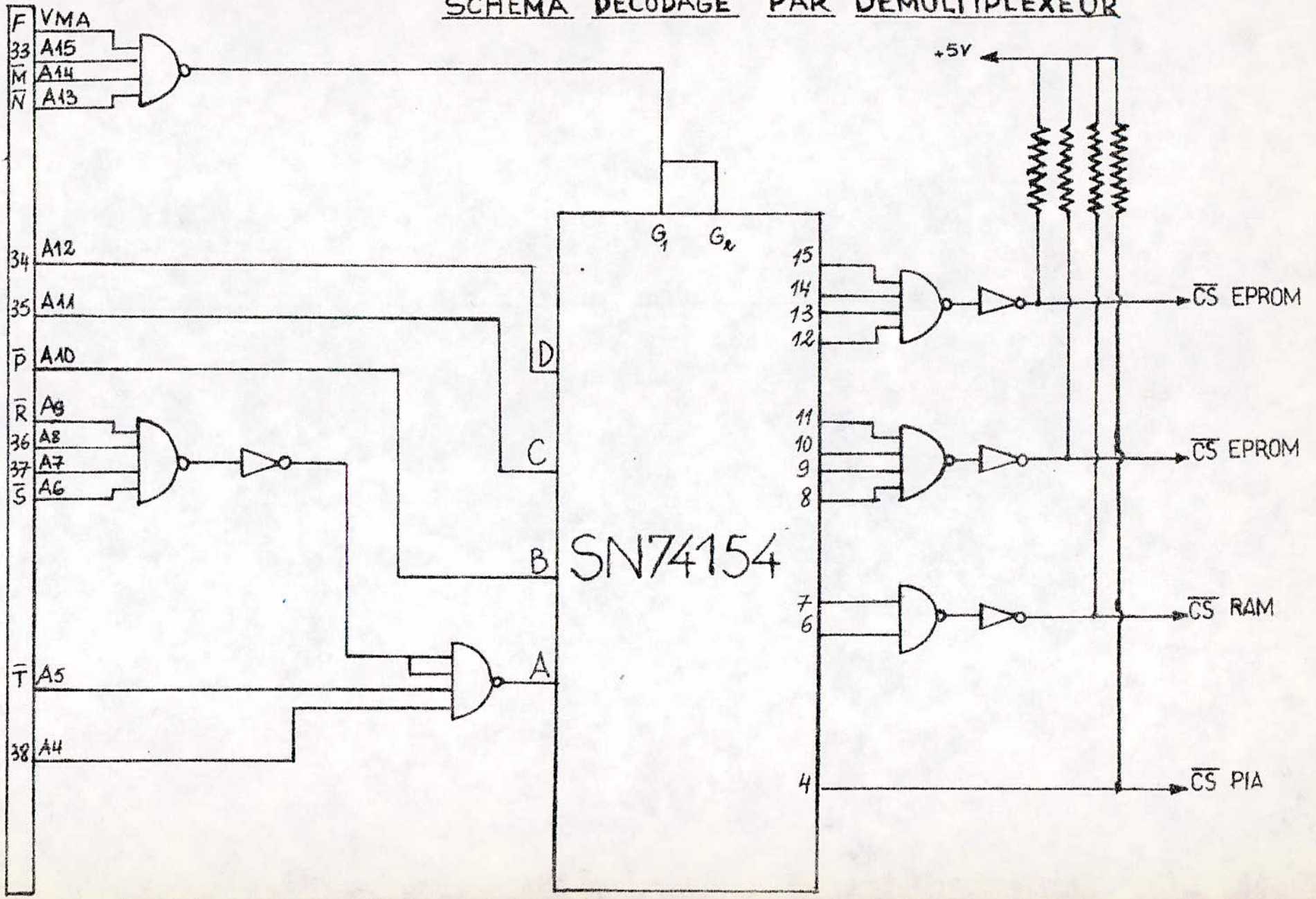
Dans les applications où le temps est un paramètre essentiel l'utilisation des PROM bipolaires est justifiée puisqu'à l'heure actuelle ce sont les seules PROM suffisamment rapides pour ne pas introduire de délais prohibitifs sur les signaux.

Les chronogrammes de la figure 3,6 mettent en évidence l'influence du temps sur le décodage d'adresse.

Les figures de la page suivante donnent les schémas complets des deux types de décodage que nous avons utilisés.


M
A14
33
A15
N
A13
34
A12
E
G
18
20
A10
14
35
A11
A9
22
D
A10
A8
23
R
A9
A7
1
36
A8
A6
2
MCM 2716
37
A7
A5
3
S
A6
A4
4
T
A5
A3
5
38
A4
A2
6
39
A3
A1
7
U
A2
A0
8
+5V
9
D0
10
D1
11
D2
13
D3
14
D4
15
D5
16
D6
17
D7
VERS CS DE LA PROM 1
CS DE LA PROM 2
PTM
ACIA
PIA
PROM 4x8 bits
RAM 2
RAM 1

SCHEMA DECODAGE PAR DEMULTIPLEXEUR



(3) Mémoires :

Les mémoires utilisées sont de deux types :

- RAM ou mémoire vive : MCM 6810 :

C'est une mémoire statique de 128 Octets utilisable dans des systèmes organisés autour d'un bus. Elle est fabriquée dans la technologie MOS Canal N, grille silicium. Ce circuit n'a besoin que d'une seule tension d'alimentation (+ 5 Volts). De plus, il est compatible T T L, et, étant de fonctionnement statique n'a besoin d'aucune horloge ou de signal de rafraichissement.

Le circuit est compatible avec la famille du µ P 6800 l'extension de la mémoire est possible grâce à plusieurs entrées de sélection . Le temps d'accès maximum est de 450 ns.

- ROM ou Mémoire morte : EPROM MCM 2716

L'EPROM MCM 2716 est une mémoire effaçable et reprogrammable électriquement de 2048 x 8 bits. Elles sont utilisables pour des applications demandant une mémoire non-Volatile qui doit être reprogrammée périodiquement. La fenêtre transparente sur le boitier permet d'effacer aux rayons ultra-violets le contenu de la mémoire. Elle possède un mode de consommation statique.

(4) Interface Parallèle d'entrée-sortie : P I A

Le circuit MC 6821 fournit un moyen universel d'interface des appareils périphériques avec les microprocesseurs de la famille MC 6800.

On a ainsi un micro-ordinateur monocarte aux performances modestes mais largement exploitable. En effet le PIA peut servir d'organe de liaison avec la Télétype (T T Y). Nous donnons en annexe une étude simplifiée du PIA ainsi que les programmes qui permettent son utilisation comme adaptateur série.

CHAPITRE IV

A P P L I C A T I O N

INTRODUCTION

Notre réalisation doit servir à commander à distance des moteurs pas à pas utilisés pour positionner des détecteurs et des cibles situés dans une "chambre à réaction" montée en bout d'extension d'un accélérateur électrostatique VAN DE GRAFF du laboratoire de Physique Nucléaire du C.E.N d'Alger.

La tension maximum atteinte pour l'accélerateur est de 3,8 M V (Million de Volt) et l'intensité fournie par la source d'ions varie de quelques n A à plusieurs μ A.

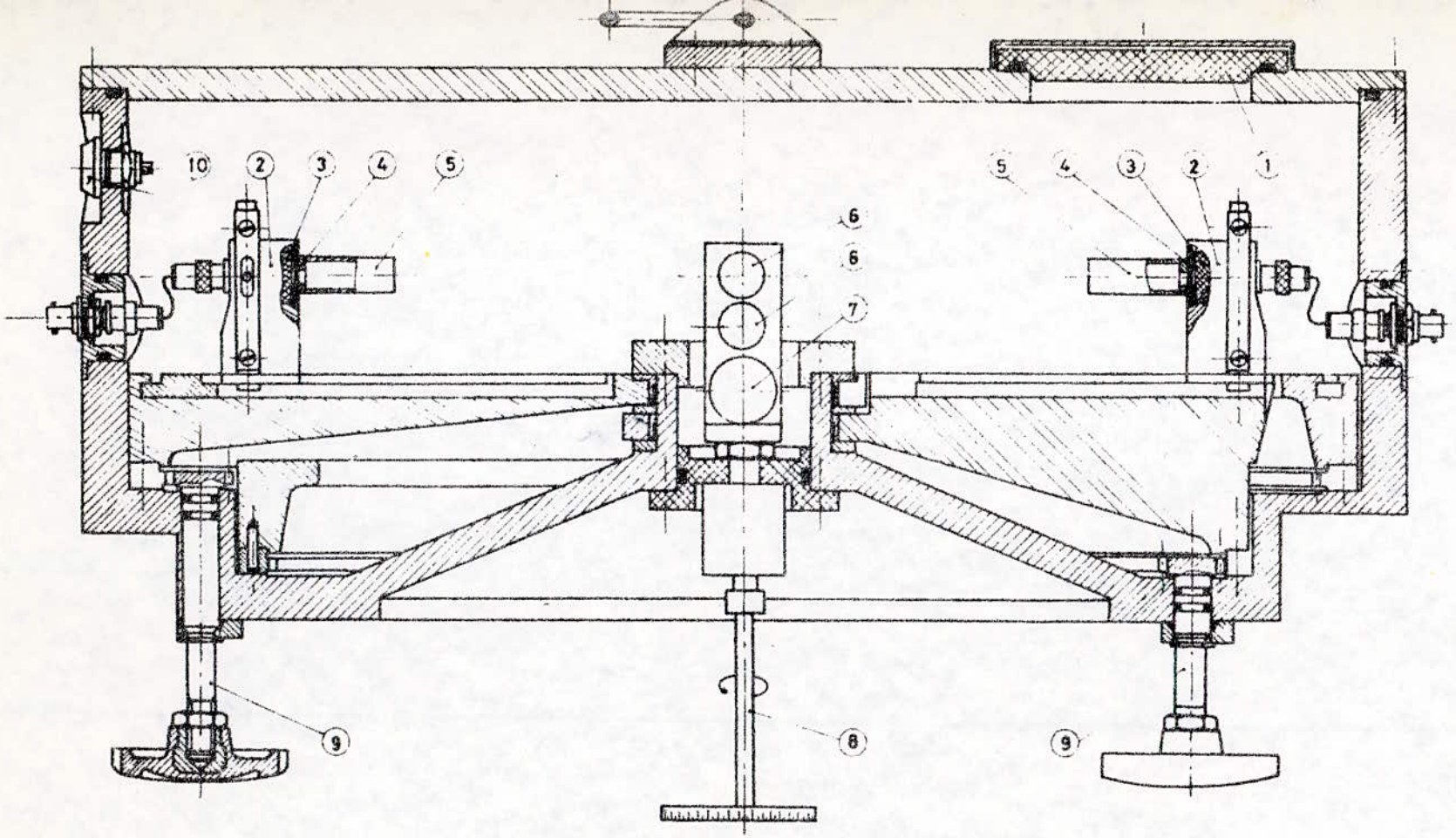
La chambre à réaction est maintenue, en fonctionnement sous un vide poussé de 10^{-5} à 10^{-6} Tor.

Boite à cible ou chambre de réaction :

La boite à cible est constituée d'un cylindre de 500 mm de diamètre. Elle comporte deux couronnes reliées au centre par un rayon. (Se reporter à la figure 7).

Chaque rayon sert de support à 3 détecteurs écartés de 15°. Les couronnes sont dentées sur leur face interne ce qui permet une attaque par un pignon, lui-même guidé à travers un passage étanche (9).

Chaque groupe de 3 détecteurs est mobile en rotation sur un angle d'environ : 300° ce qui permet d'ajuster la situation spatiale la plus propice à la détection.



(1)	Hublot transparent	(7)	Quartz
(2)	Porte détecteur	(8)	Passage tournant du porte cible isolé et piégé
(3)	Détecteur		
(4)	Diaphragme	(9)	Commande extérieure des bras mobiles portant les détecteurs
(5)	Tube de garde isolé et piégé		
(6)	Cible	(10)	Prise Lemo utilisée pour les tensions de piégeage

Figure 7 : Schéma de la chambre à réaction utilisée pour les mesures de coïncidences .

Chaque détecteur est mobile sur sa position radiale ce qui permet le réglage de distance vis-à-vis du centre de la chambre.

Le porte cible qui comporte trois cibles est monté sur un axe (8) qui traverse un passage étanche. Ceci permet un mouvement de rotation et de translation verticale pour permettre un bon positionnement des cibles par rapport au faisceau de particules. La première cible est généralement un quartz (7) qui sert au réglage de la focalisation, de l'intensité du faisceau et d'autres paramètres avant le début de la réaction.

Le faisceau incident bombarde la cible sous un angle dépendant du but de la manipulation et qui peut être modifié en cours d'expérience.

Avant le début de la réaction, les détecteurs et la cible sont positionnés suivant une certaine géométrie. Pour des modifications éventuelles de positions des organes (détecteurs et cibles) l'opérateur est obligé d'arrêter la réaction avec toutes les conséquences que cela entraine et qui dépendent de la nature de l'expérience (entrée d'air, pompage, modification des positions détecteurs, risque de contamination, .. etc,..)

Pour palier à cet inconvénient, la commande des moteurs par microprocesseur, qu'on se propose d'étudier, nous permet de changer le position - nement de la cible et des détecteurs sans modifier aucun autre paramètre : intensité du faisceau incident, focalisation, etc..). Actuellement l'implantation mécanique des moteurs dans la boite à cible n'est pas réalisée. Nous voulons tester le bon fonctionnement de notre système. Pour cela nous avons simulé la commande séquentielle de dix moteurs qui serviront à positionner les détecteurs et la cible.

A - PARTIE HARDWARE

1 - PRESENTATION DE LA CARTE DE COMMANDE

La commande du système nécessite 10 moteurs pas à pas de positionnement qui avancent d'un cran à chaque impulsion.

Au début de l'expérience, tous les organes à commander sont supposés à l'état initial : à savoir 70 mm pour les mouvements de translation et 0 degré pour les mouvements de rotation des portes détecteurs et du porte cible. Dans leurs positions maximales les détecteurs sont distants de 240 mm du centre de la chambre. On suppose que le mouvement de rotation des groupes de détecteurs est limité à 180° (portes détecteurs) au même titre que pourle porte cible. Pour ce dernier, le mouvement vertical est de 70 à 240 mm en prenant comme référence la première cible.

2 - REALISATION DE LA CARTE

Le schéma de la figure (8) nous montre l'utilisation rationnelle des deux P I A . La liaison avec la carte micro-ordinateur se fait à l'aide d'un connecteur.

* Un bouton poussoir BP1 sur la ligne de commande du P I A 1 permet la sélection de l'organe en question qui sont en nombre de 14.

* Les 4 lignes P A 0 - P A 3 du PORT A du P I A 1 aboutissent aux entrées d'un démultiplexeur qui permet d'associer à chacun des organes une L E D.

* Les lignes du PORT B du P I A 1 amplifiées par des 8T26 attaquent les afficheurs par les entrées D C B A qui indiquent la position effective de chacun des organes une fois "sélecté," l'affichage suit le déplacement et permet à l'opérateur de visualiser son positionnement avec exactitude.

* 3 touches reliées au DATA A du PORT A du P I A 2 servent à commander les moteurs pour l'ajustement de la position des organes, le micro-ordinateur répond par une série d'impulsions sur les lignes du PORT B du P I A 2 et les lignes P A 4 et P A 5 du P I A 1. On affecte à chaque ligne la commande d'un moteur qui se fera par l'intermédiaire d'un étage de puissance.

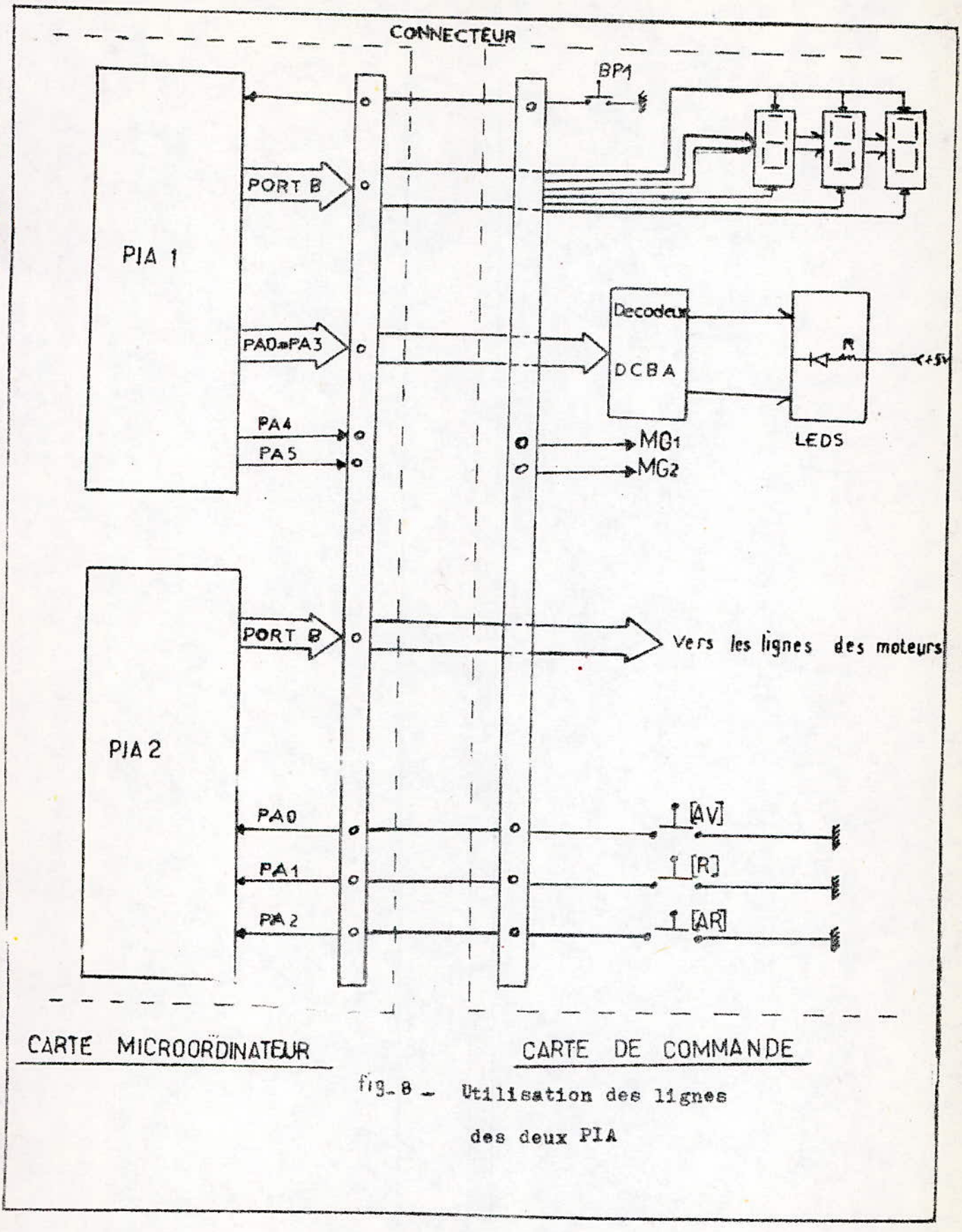


fig. 8 – Utilisation des lignes des deux PIA

- Les commandes |AV| et |AR| reliées respectivement à PAD et PA 1 permettent le positionnement lent des moteurs (déplacement lent des détecteurs, portes détecteurs et cibles). La LED 1 s'allume sous l'action de la commande |AR| pour indiquer qu'il s'agit d'un mouvement de recul.

- La commande |R| est reliée à la ligne PA 2 du P I A 2. L'action sur cette commande suivie de l'une des deux précédentes permet un déplacement rapide des organes.

* Enfin le deuxième bouton poussoir B P 2 permet l'initialisation de tout le système. Une bascule R S réalisée à l'aide de deux portes NAND élimine les risques de rebondissements qui sont provoqués par l'action sur ce bouton poussoir. L'une des sorties de la bascule R S est reliée directement à la broche $\overline{\text{RESET}}$ du MC. 6809 commune à tous les circuits de la carte micro-ordinateur.

3 - UTILISATION DE LA CARTE COMMANDE

Les touches qu'on a utilisées sont du type fugitif c'est à dire que le contact fermé ne durera que tant que le doigt appuiera sur la touche : |AV|, |AR| et |R|. Le bouton poussoir BP 1 a une fonction totalement différente puisqu'il agit directement sur la ligne de controle CA 1 du P I A 1 et positionne le bit 7 du registre de contrôle correspondant. Celui-ci permet la sélection de l'organe à déplacer. Un compteur digit est associé pour mémoriser le nombre de fois que le bouton BP1 est actionné (VOIR PROGRAMMES ET ALGORITHMES de gestion du système).

Nous avons vu précédemment qu'il y avait 14 organes différents à commander ; de ce fait le compteur digit (C.D1) prend toutes les valeurs de 1 à 14. Chaque chiffre est affecté à un organe et allume une LED qui lui est associée. Par principe, il faut reboucler sur le premier organe dès que l'on a dépassé le chiffre 14. Le tableau ci-après donne les valeurs de CD1 correspondantes aux 14 organes.

COMPTEUR DIGIT : CD1	ORGANE CORRESPONDANT	MOUVEMENT DE L'ORGANE	MOTEUR ET LIGNE DU P I A
1	DETECTFUR D1	TRANSLATION	M1 ; PB0
2	D2	"	M2 ; PB1
3	D3	"	M3 ; PB2
4	D'1	"	M4 ; PB3
5	D'2	"	M5 ; PB4
6	D'3	"	M6 ; PB5
7	CIBLES	TRANSLATION	M7 ; PB6
8	CIBLES	ROTATION	M8 ; PB7
9	D1	"	MG1 : Moteur du groupe 1 ligne : PA4
10	D2	"	
11	D3	"	
12	D'1	"	MG2 : Moteur du groupe 2 ligne : PA5
13	D'2	"	
14	D'3	"	

Explicitons maintenant par étapes la façon de procéder pour commander les détecteurs et la cible après initialisation.

Nous devons :

* Sélectionner l'organe à déplacer à l'aide de B P 1.

* Vérifier la sélection en examinant les LED associées et l'affichage automatique de la position réelle.

* Deux possibilités de modification de position :

a) Lente : - Touche |AV| pour l'avance tant que le doigt appuie sur celle-ci.

- |AR| pour le recul.

b) Rapide : Sur action simultanée de |R| et de l'une des deux précédentes.

Quand la position maximale ou minimale est atteinte les afficheurs commencent à clignoter.

Affichage :

Pour l'affichage de la position effective d'un des 14 organes on a utilisé la commande multiplexée de 3 afficheurs de type T I L 308 (Schéma interne Fig. 9) à partir d'un code d'affichage rangé en mémoires :

- 1 pour les unités ;
- 1 pour les dizaines ;
- 1 pour les centaines .

Commande multiplexée des afficheurs :

Les entrées de commande des afficheurs sont en parallèle et l'on validera séquentiellement les commandes des cathodes.

Les 4 lignes PB0 à PB3 du P I A 1 attaquent, après amplification par des buffers, les entrées D C B A des afficheurs. La conversion binaire B C D s'avère nécessaire. On verra plus loin que cette conversion se fait par logiciel.

Les afficheurs possèdent une entrée de validation qui est le "LATCH STROBE INPUT". Cette entrée permet aussi la mémorisation d'une donnée sur l'afficheur ; Elle est active à l'état bas :

48. SCHEMATIC DRAWINGS

IN DRAWING NUMBER SEQUENCE

EA6

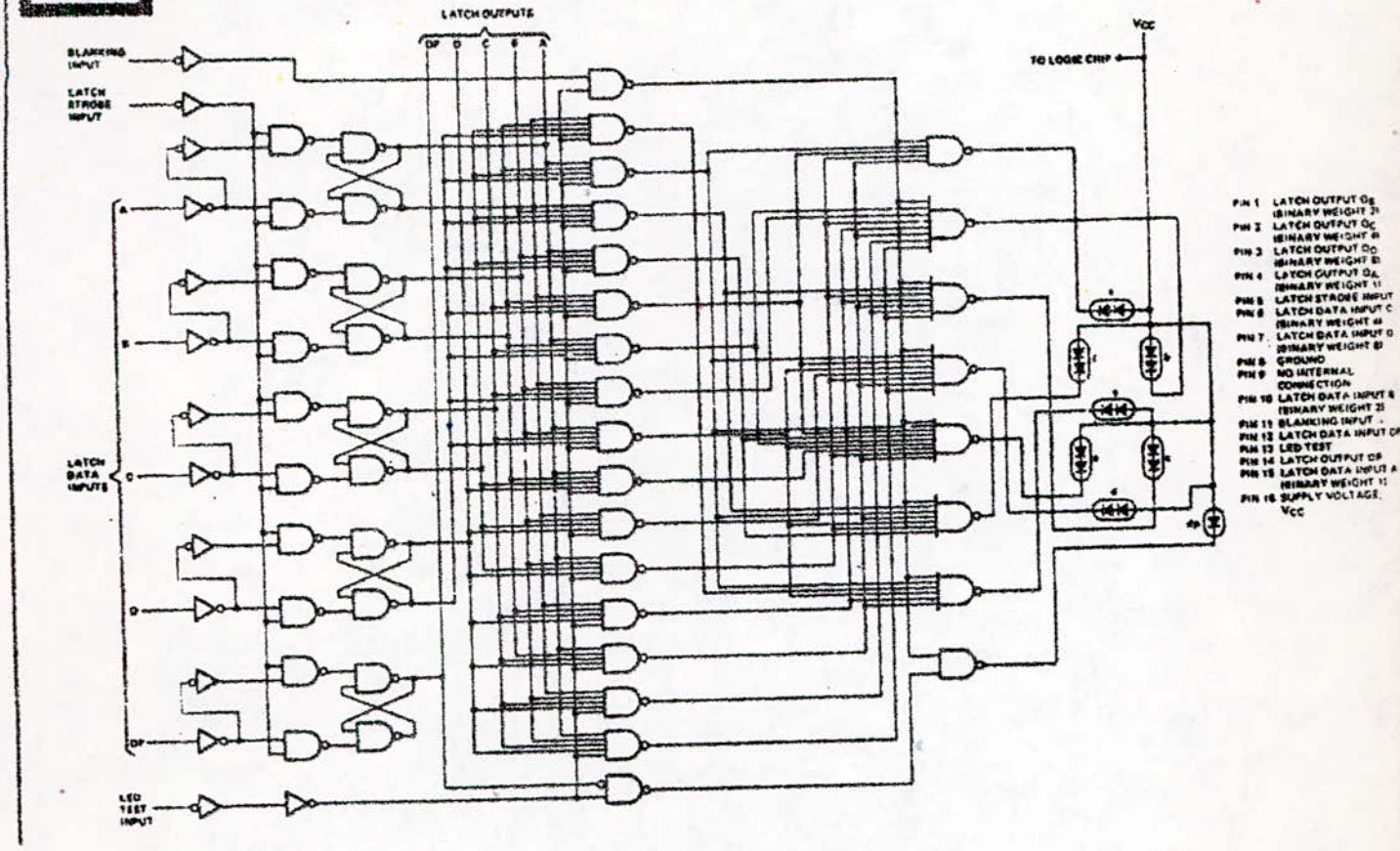


FIG.9

- Un état haut sur cette entrée permet de mémoriser la valeur précédemment affichée ;

- Un état bas valide l'afficheur qui prend en compte la combinaison se présentant sur les entrées D C B A (LATCHS STROBE INPUT), les afficheurs possèdent un décodeur intégré qui permet la conversion B C D - 7 segments.

La commande de 3 afficheurs nécessite 3 lignes. A cet effet on utilise les lignes PB4, PB5, et PB6 du P I A 1. Ces lignes véhiculent le code de validation des afficheurs, elles aboutissent sur des inverseurs de type 7404. Le tableau suivant donne les combinaisons qui valideront ces 7 segments.

PB7	PB6	PB5	PB4	Afficheur sélecté
1	0	0	1	Unités
1	0	1	0	Dizaines
1	1	0	0	Centaines
0	X	X	X	Extinction des 3 afficheurs

Ces codes sont rangés en mémoire à l'initialisation, d'ailleurs on verra dans la suite que ces positions mémoires sont les mêmes qu'on utilise pour la conversion Binaire - B C D.

Enfin la ligne PB7 du PIA 1 attaque les entrées "BLANKING INPUT" des 7 segments à travers un inverseur. Cette ligne éteint les afficheurs quand elle est à l'état bas. Son rôle sera explicité plus loin (VOIR PARTIE SOFTWARE).

B - PARTIE SOFTWARE

Le logiciel du système comporte un certain nombre de subroutines qu'on va expliciter avant de donner l'Algorithme général.

Ce logiciel comprend :

- Subroutine d'affichage et de conversion Binaire - B C D
- Subroutine de commande des moteurs
- Subroutine de temporisation appelée DELAY
- Subroutine de clignotement appelée BLANK

1. Subroutine d'affichage et de conversion Binaire-BCD :

a - Principe :

Le nombre à convertir est au maximum égal à $(240)_{10}$ en base 10.

On retranche 100 et on incrémente parallèlement une position correspondant aux centaines, jusqu'à ce que le résultat soit négatif, alors on ajoutera 100.

On continue ainsi avec le nombre 10 ; le reste sera mis dans la position mémoire réservée aux unités.

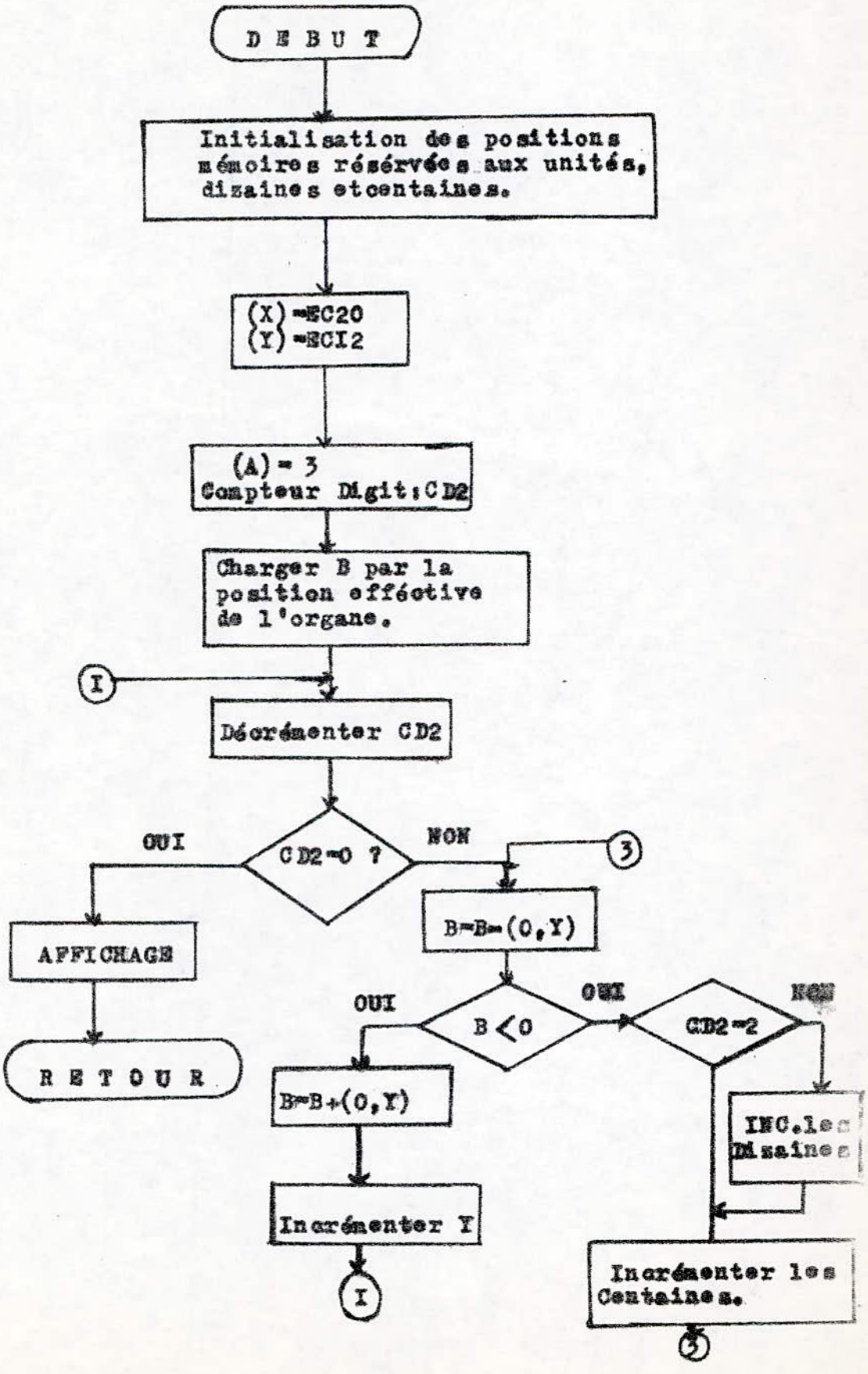
L'initialisation des positions mémoires des centaines, des dizaines et des unités permet la commande multiplexée des afficheurs.

b - Algorithme :

Le nombre à convertir va être cherché dans une position mémoire réservée à chaque organe. L'initialisation du registre d'index X à EC00 permet l'utilisation du mode d'adressage indexé. Le registre d'index Y permet aussi de retrancher en indexé l'équivalent en binaire de $(100)_{10}$ et de $(10)_{10}$ respectivement (64) et (0 A) en HEXADECIMAL qui sont le contenu des positions EC12 et EC13.

SUBROUTINE DE CONVERSION BINAIRE-BCD: SB CNV

ALGORITHME:



2. Subroutine de commande des moteurs :

On utilise 2 positions mémoires pour générer les impulsions :

a) Position EC11 :

On l'initialise à $(80)_{16}$ quand le premier moteur est "sélecté," puis on "shifte" cette position à chaque action sur B P 1.

b) Position EC18 :

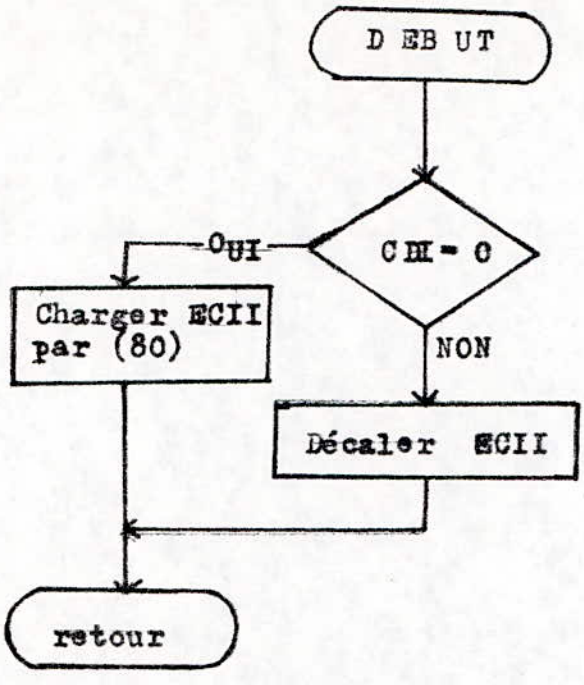
Pour les moteurs commandant la rotation des portes détecteurs on utilise l'accumulateur A et le compteur digit C D 1 ; Ceci évite de modifier la combinaison sur les LEDS.

A chaque action sur |AV| et |AR| l'une de ces positions est transférée sur le port du P I A correspondant pour actionner l'un des moteurs.Après un certain temps, qui correspond à la durée d'impulsion, les lignes du port doivent retomber à l'état bas.

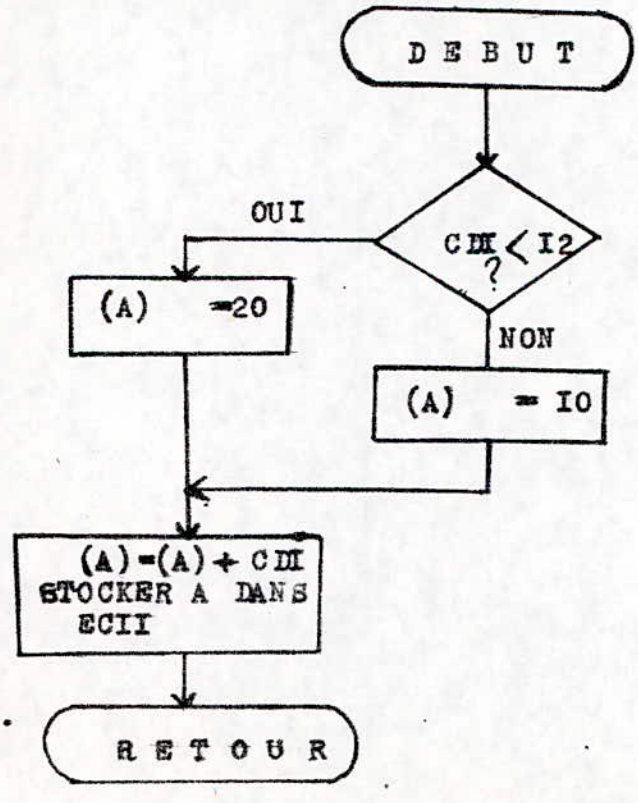
SUBROUTINE DE COMMANDE:

ORGANIGRAMMES:

a) MOTEURS 1à8:



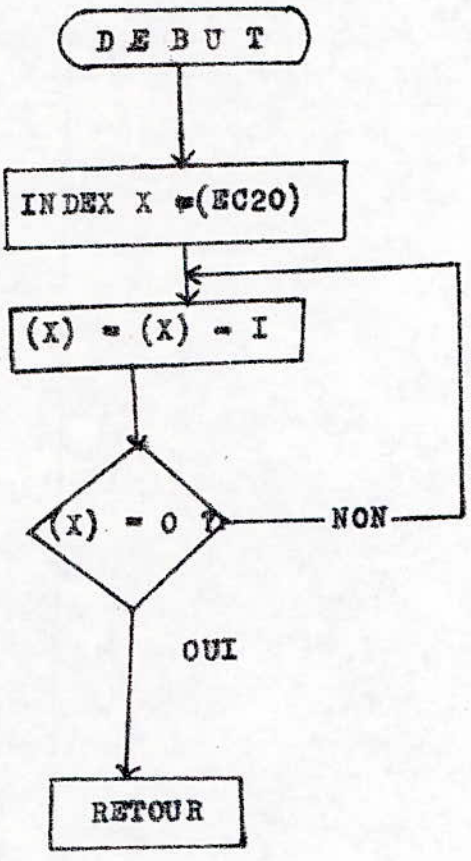
b) MOTEURS MG1 et MG2:



3. SUBROUTINE DE TEMPORISATION: SB DELAY

Le principe est simple:
On charge l' index X par la valeur de temporisation et on le decrémente à zéro. Le chargement de X se fait à partir d' une position mémoire EC20 ce qui permet l'utilisation de la subroutine pour des tailles de temporisation differentes.

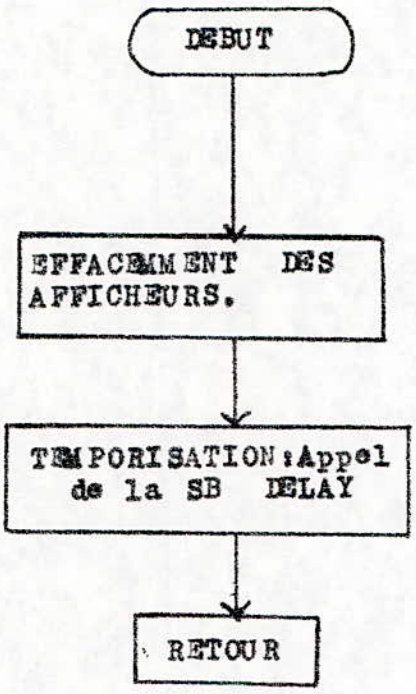
ALGORITHME :



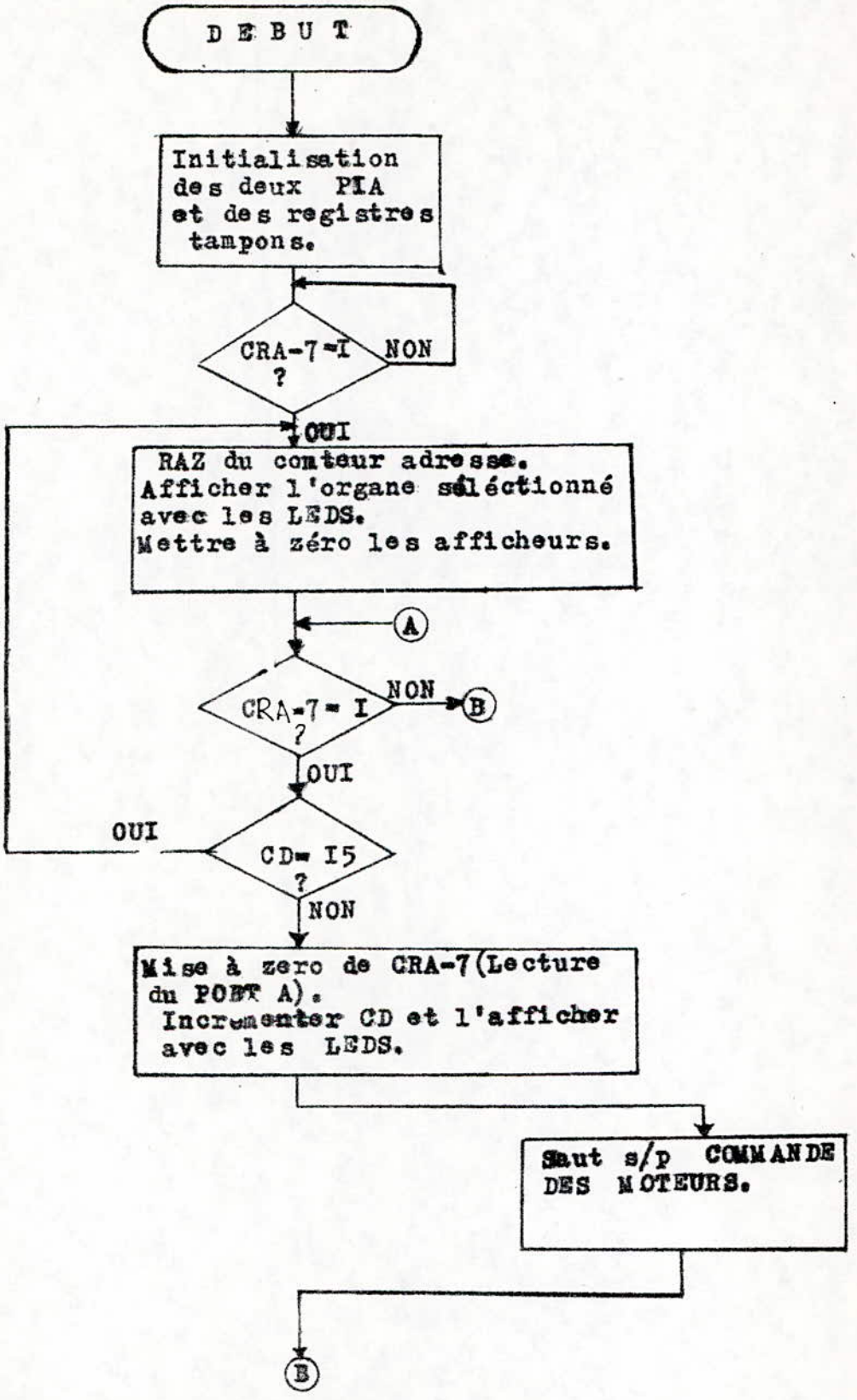
4.SUBROUTINE DE CLIGNOTEMENT: SB BLANK

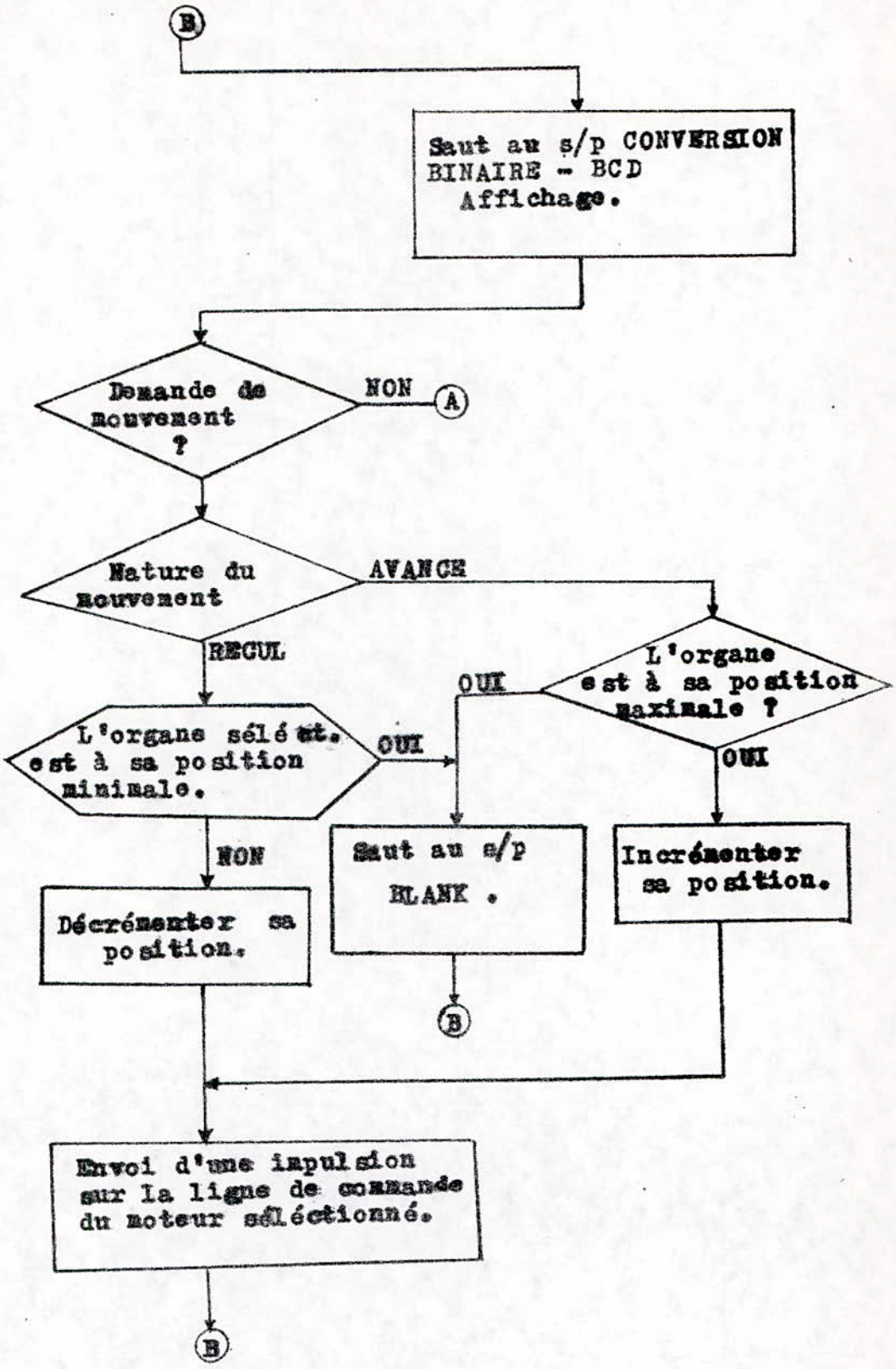
On envoie un "0" sur la broche "BLANKING INPUT" des afficheurs par la ligne PB7 duPIAI, on charge la position memoire EC20 par une valeur de temporisation TEMP3 et on se branche sur la SB DELAY pour refaire le cycle d'affichage.

ORGANIGRAMME:



ALGORITHME DE COMMANDE DE LA CHAMBRE A CIBLE.

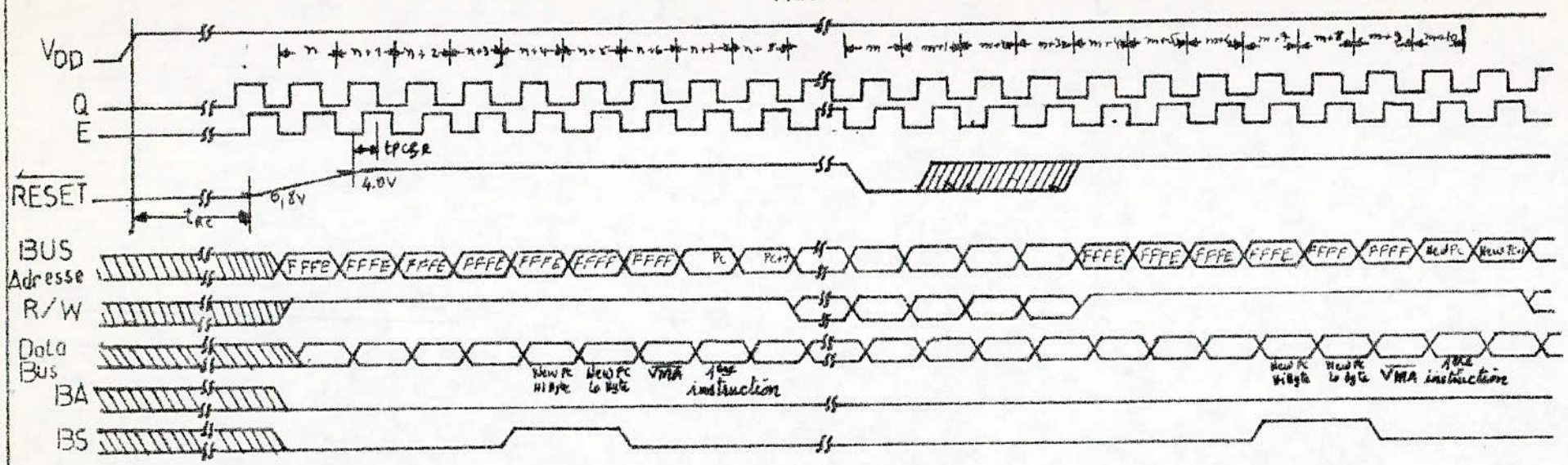


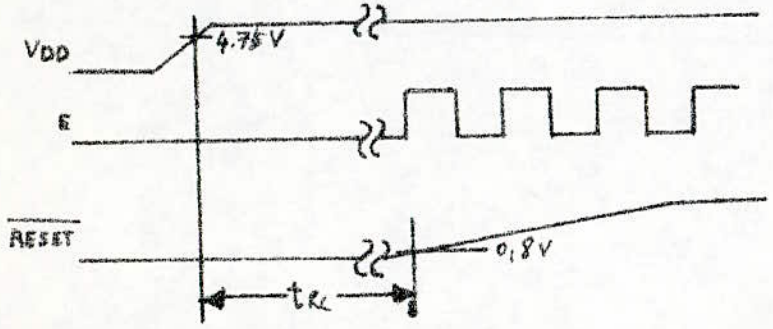

B
Saut au s/p CONVERSION BINAIRE - BCD Affichage.
Demande de mouvement ?
NON
A
Nature du mouvement
AVANCE
RECUL
L'organe sélect. est à sa position minimale.
OUI
NON
L'organe est à sa position maximale ?
OUI
OUI
Saut au s/p BLANK .
Décrémenter sa position.
Incrémenter sa position.
B
Envoi d'une impulsion sur la ligne de commande du moteur sélectionné.
B

ANNEXES

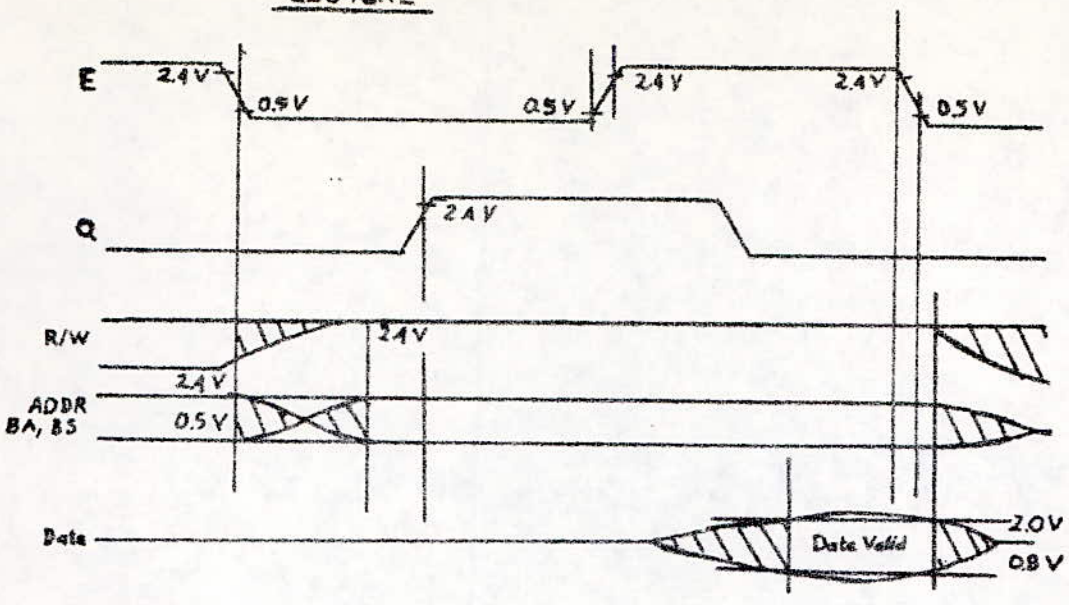
ANNEXE_A

RESET

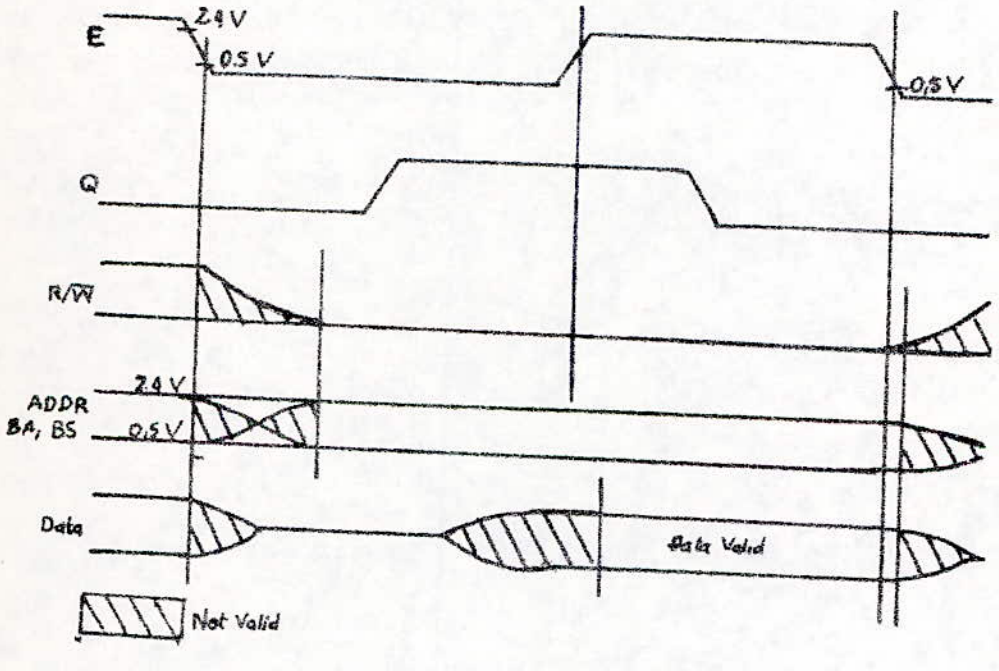




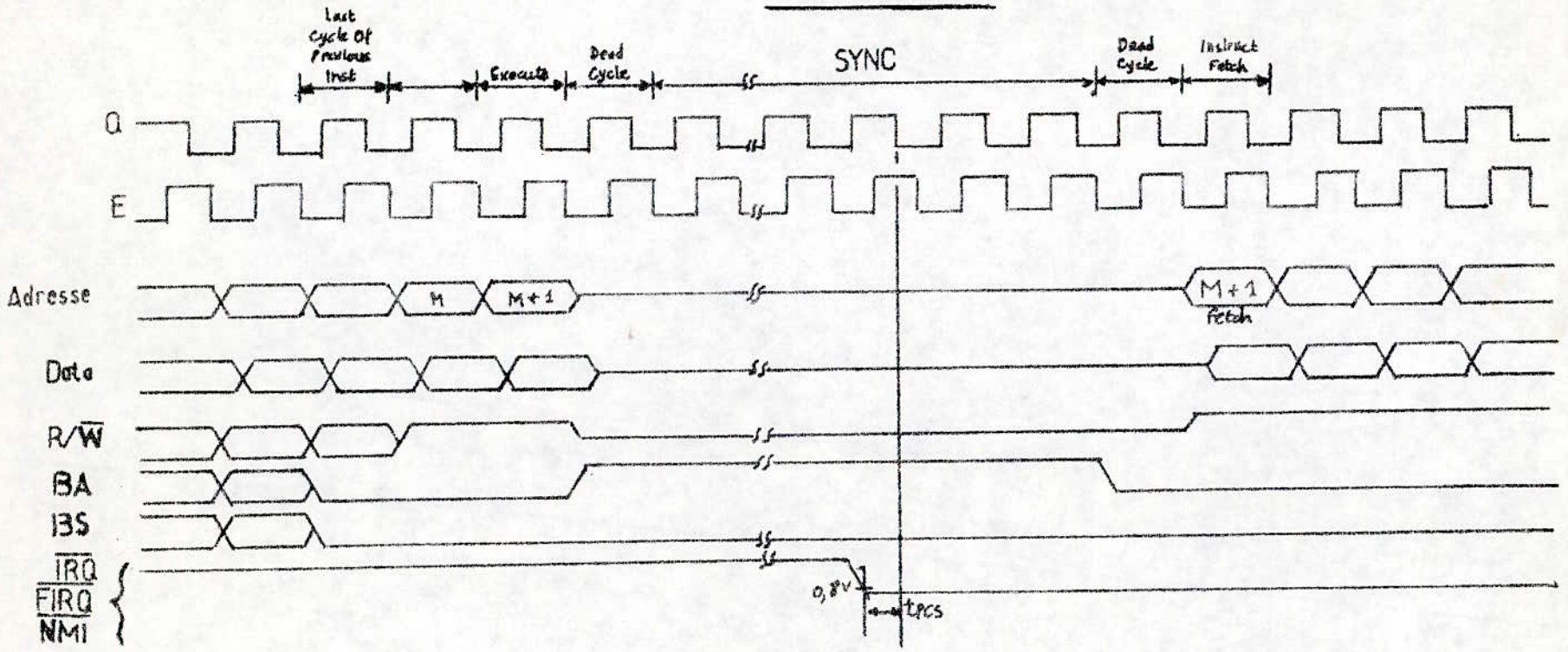
LECTURE



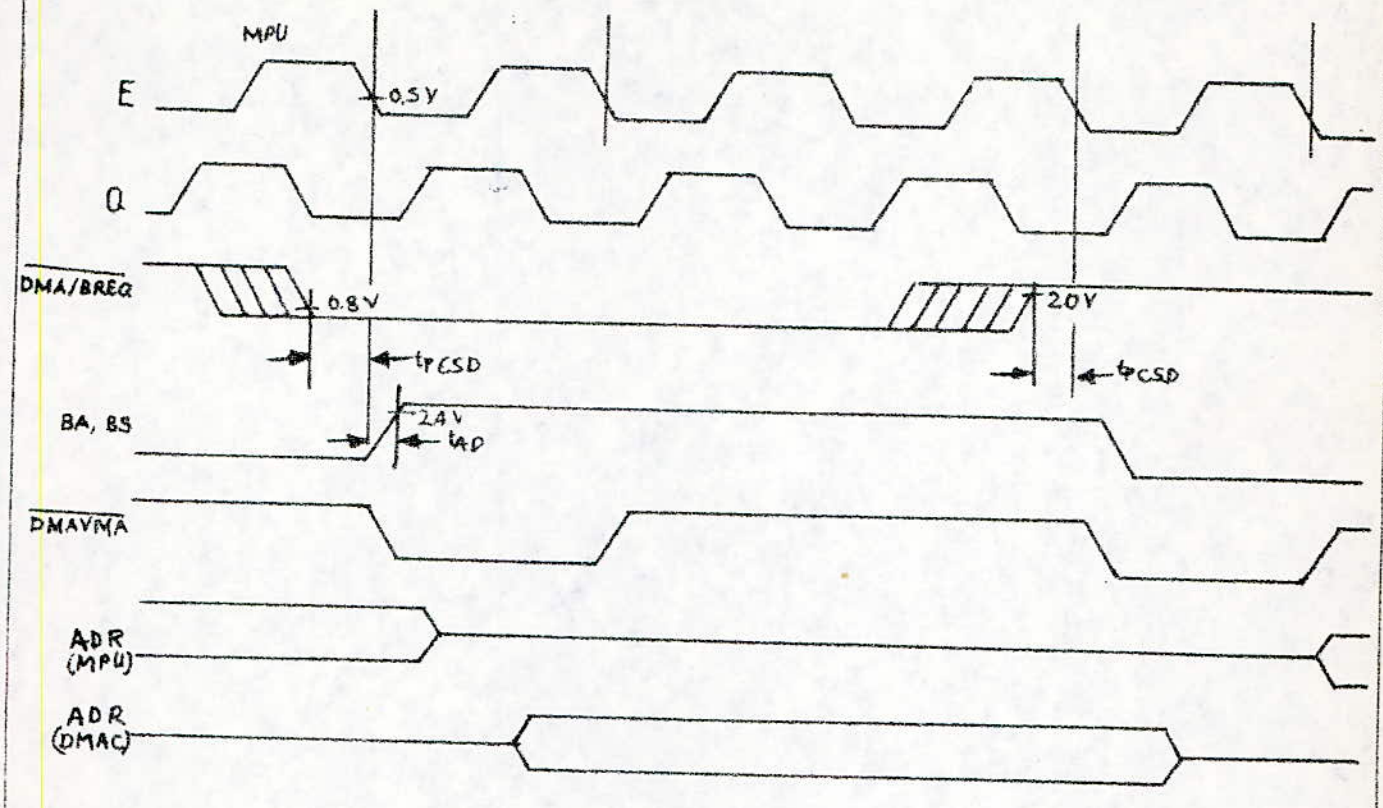
ECRITURE



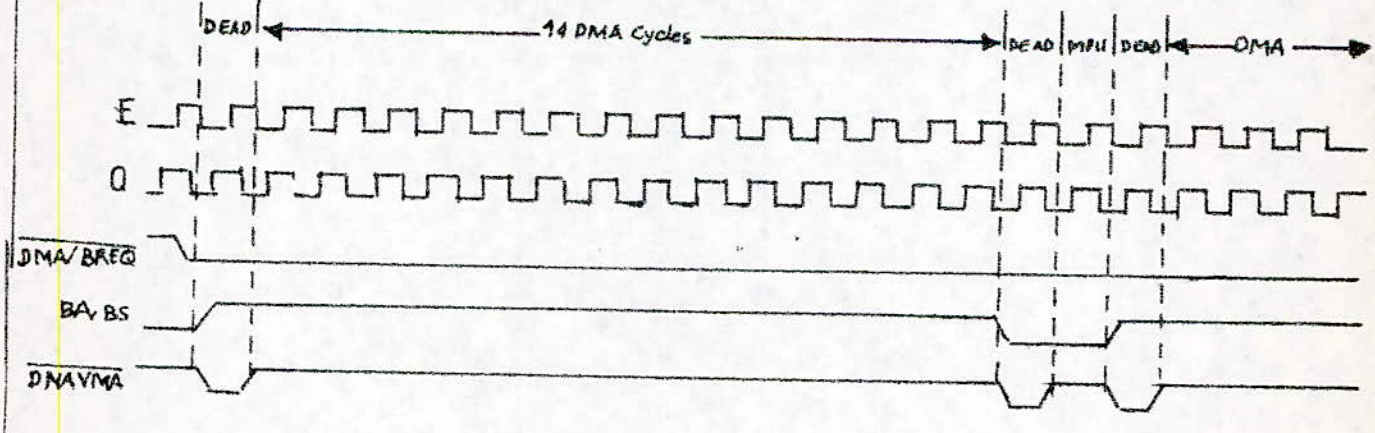
SYNC TIMING


Last Cycle Of Previous Inst
Execute
Dead Cycle
SYNC
Dead Cycle
Instruct Fetch
Q
E
Adresse
M
M+1
M+1 Fetch
Data
R/W
BA
BS
IRQ
FIRQ
NMI
0,8V
tpcs

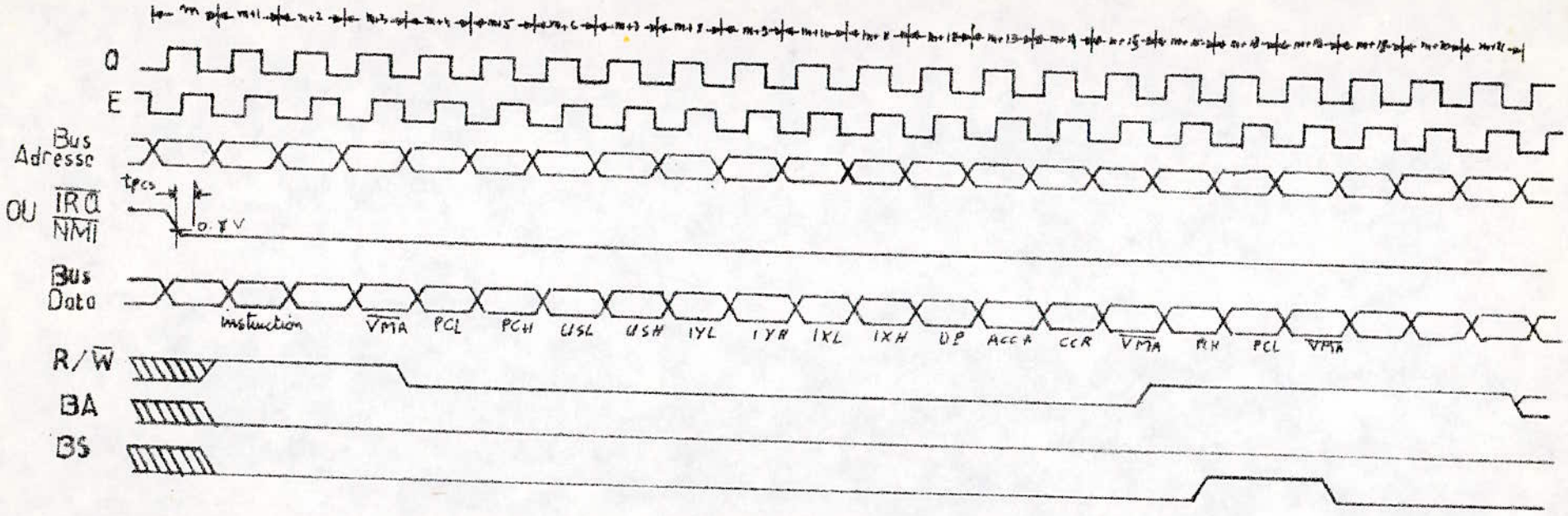
DMA TIMING



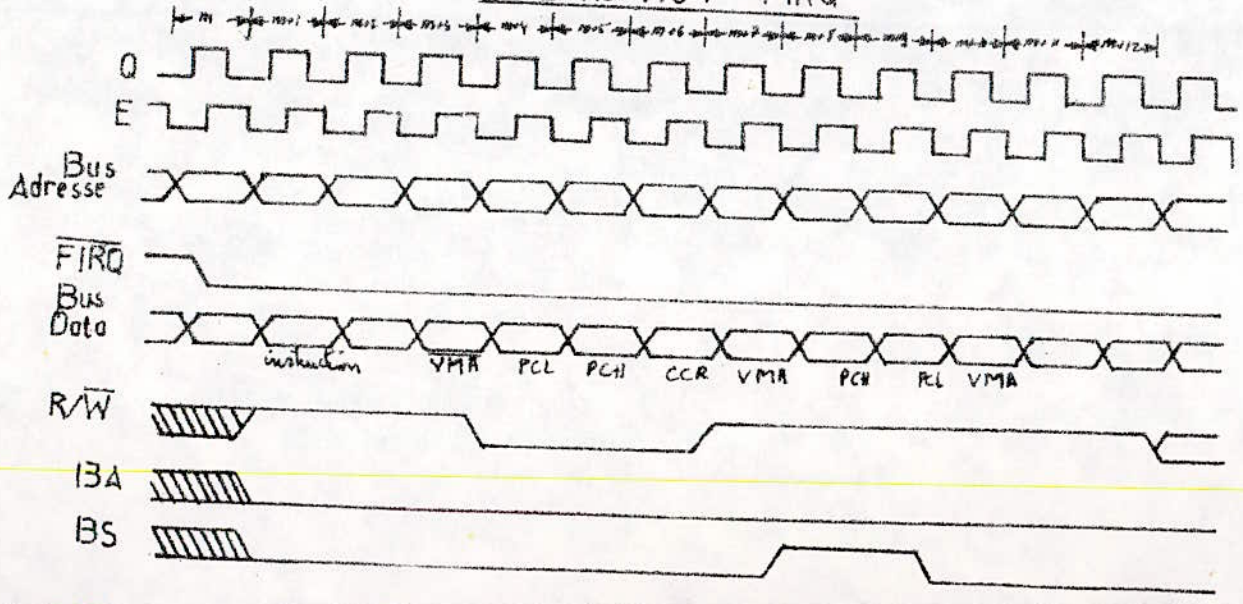
AUTO_REFRESH DMA TIMING



INTERRUPTIONS $\overline{IRQ}$ & $\overline{NMI}$



INTERRUPTION $\overline{FIRQ}$



ANNEXE_B

FEUTRIER SUD-EST — FEUTRIER ILE-DE-FRANCE

MODES D'ADRESSAGE 6809

Immédiat

le même que pour 6800

cependant le 6809 a plus de registres

Implicite

le même que pour 6800

pour le 6809 le code opératoire peut tenir sur 2 octets

Direct

le même que pour 6800

le 6809 possède un registre de page direct

Etendu

le même que pour 6800

possibilité d'un niveau d'indirection

Indirect étendu

nouveau pour le 6809

l'adresse pointe la nouvelle adresse à utiliser

code opératoire sur 2 octets le 2ème octet indique le mode indirect

le 2ème octet est 9 F sauf pour ST, TST, SUB (F 9)

le 1er octet est le code opératoire du mode indexé

Indexé

existe sous une forme simplifiée dans le 6800

plus puissant dans le 6809

le déplacement peut être ± 5, 8, 16 bits

Les registres A, B, D peuvent être utilisés comme déplacement

X, Y, U, S ou PC peuvent être utilisés comme registres d'index

auto incrémentation, décrémentation + 1, ± 2 avec un déplacement nul

codes opératoires sur 2 octets le 2ème octet indique un des

Indirect indexé

nouveau pour le 6809

l'adresse obtenue dans le mode indexé est utilisée pour pointer l'adresse à utiliser

ne peut pas être utilisé avec l'auto incrémentation par 1 (± 1)

Relatif

le même que pour le 6800 mais 1 cycle plus rapide

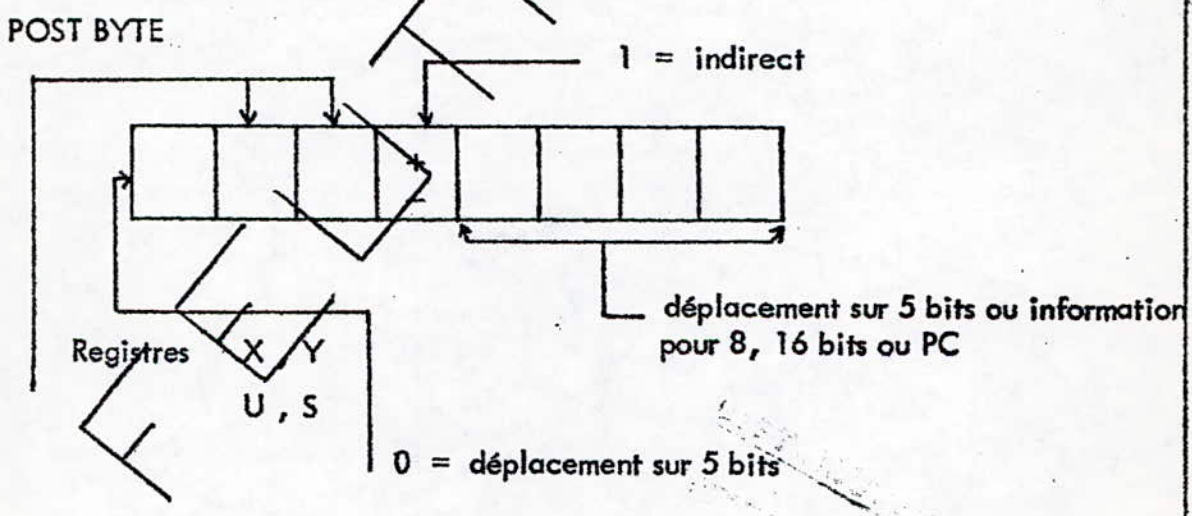
Relatif long

nouveau pour le 6809

le déplacement est en complément à 2 sur 2 octets

code opératoire sur 2 octets le premier octet est toujours 10_{16} le 2ème octet est le même que dans le mode relatif

temps de cycle variable



MODES D'ADRESSAGE INDEXE DU 6809

TYPE	FORMS	NON INDIRECT ASSEMBLER FORM	NON INDIRECT POST-BYTE OPCODE	NON INDIRECT + ~	NON INDIRECT + #	INDIRECT ASSEMBLER FORM	INDIRECT POST-BYTE OPCODE	INDIRECT + ~	INDIRECT + #
CONSTANT OFFSET FROM R	NO OFFSET	,R	1RR00100	0	0	[,R]	1RR10100	2	0
	5 BIT OFFSET	n, R	ORRNNNNN	1	0	DEFAULTS TO 8-BIT			
	8 BIT OFFSET	n, R	1RR01000	2	1	[n, R]	1RR11000	4	1
	16 BIT OFFSET	n, R	1RR01001	4	2	[n, R]	1RR11001	6	2
ACCUMULATOR OFFSET FROM R	A-REGISTER OFFSET	A, R	1RR00110	1	0	[A, R]	1RR10110	3	0
	B-REGISTER OFFSET	B, R	1RR00101	1	0	[B, R]	1RR10101	3	0
	D-REGISTER OFFSET	D, R	1RR01011	4	0	[D, R]	1RR11011	6	0
AUTO INCREMENT/DECREMENT R	INCREMENT BY 1	R+	1RR00000	2	0	NOT ALLOWED			
	INCREMENT BY 2	R+ +	1RR00001	3	0	[R+ +]	1RR10001	5	0
	DECREMENT BY 1	−R	1RR00010	2	0	NOT ALLOWED			
	DECREMENT BY 2	− − R	1RR00011	3	0	[− −R]	1RR10011	5	0
CONSTANT OFFSET FROM PC	8 BIT OFFSET	n, PC	1XX01100	3	1	[n, PC]	1XX11100	5	1
		n, PCR	1XX01100	3	1	[n, PCR]	1XX11100	5	1
	16 BIT OFFSET	n, PC	1XX01101	5	2	[n, PC]	1XX11101	7	2
		n, PCR	1XX01101	5	2	[n, PCR]	1XX11101	7	2

RR → 00 = X
01 = Y
10 = U
11 = S

OPTIONS DU MODE D'ADRESSAGE INDEXE

Mode d'adressage non - indirect

type d'adressage indexé	écriture assembleur	fonctionnement R = X, Y, U, S	octets supplémentaires	cycles supplémentaires
déplacement nul R non modifié	, R	prendre l'opérande pointé par R	0	0
déplacement nul R modifié	, R +	prendre l'opérande pointé par R et ensuite incrémenter R	0	+2
	, R ++	prendre l'opérande pointé par R et ensuite incrémenter R de 2	0	+3
	, - R	décrémenter R de 1 et prendre l'opérande	0	+2
	, - - R	décrémenter R de 2 et ensuite prendre l'opérande	0	+3
déplacement fixe	déplacement, R	ajouter temporairement le déplacement signé (signe plus valeur) à R et puis prendre l'opérande		
		- 16 ≤ déplacement ≤ + 15	0	+1
		-128 ≤ déplacement ≤ +127	+1	+1
		-65536 ≤ déplace. ≤ +65536	+2	+4
déplacement variable	A , R B , R D , R	ajouter temporairement le déplacement signé (mis dans l'accumulateur en complément à 2) à R et puis prendre l'opérande	+0 +0 0	+1 +1 +4
adressage relatif	étiquette , PCR	utilise l'adressage relatif pour accéder à une constante ayant une étiquette		
		- localisée à ± 128 octets de l'instruction suivante	+1	+1
		- localisée n'importe où en mémoire	+2	+5
	déplacement, PC	ajouter temporairement un déplacement signé (signe plus valeur) à l'adresse de l'instruction suivante et prendre l'opérande.		
		- 128 ≤ déplacement ≤ + 127	+1	+1
		-65536 ≤ déplace. ≤ + 65536	+2	+5

Adressage indirect Utilise l'opérande comme pointeur de l'opérande.

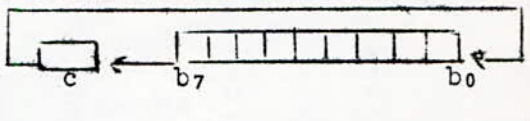
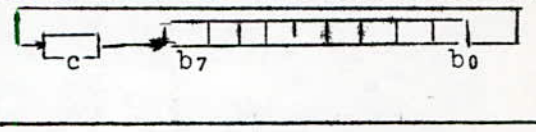
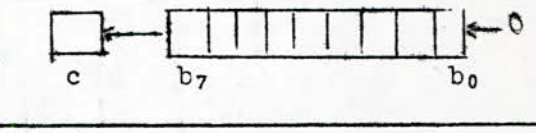
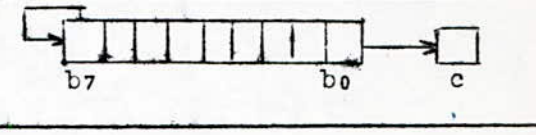
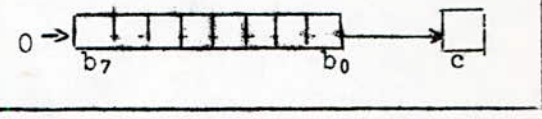
type d'adressage indexé	écriture assembleur	fonctionnement R = X, Y, U, S	octets supplémentaires	cycles supplémentaires
déplacement nul, R non modifié	[, R]		0	+3
déplacement nul R modifié	[, R + +] [, - - R]		0 0	+6 +6
déplacement fixe	[déplacement, R]	- 128 ⩽ déplacem. ⩽ + 127 - 65536 ⩽ déplacem. ⩽ + 65536	+1 +2	+4 +7
déplacement variable	[A, R] [B, R] [D, R]		0 0 0	+4 +4 +7
adressage relatif	[étiquette, PCR]	pour étiquette localisée . + 128 octets de l'instruction suivante . n'importe où	 +1 +2	 +4 +8
	[déplacement, PC]	- 128 ⩽ déplacement ⩽ + 127 - 65536 ⩽ déplacement ⩽ + 65536	+1	+4

ANNEXE-C

INSTRUCTIONS SUR LES TRANSFERTS DE DONNEES

INST.	FONCTION	MNEM.	OPERATION
L D	CHARGER ACC.	LDA LDB	(M) → A (M) → B
	CHARGER REG. 16 BIT	LDD LDX LDY LDS LDU	(M) → R_4 (M +1) → R_L
S T	STOCKER ACC.	STA STB	A → M B → M
	STOCKER REG. 16 BIT	STD STX STY STS STU	R_H → M R_L → M + 1
P S H	RANGER REG. (S) DANS LA PILE SYSTEME (S)	PSHS	SP-1 → SP R → (M SP)
	RANGER REG. (S) DANS LA PILE UTILISATEUR (U)	PSHU	USP-1 → USP R → (MUS)
P U L	DECHARGER REG. (S) DE LA PILE SYSTEME (S)	PULS	(Msp)⟶R (SP + 1) → SP
	DECHARGER REG. (S) DE LA PILE UTILISATEUR (U)	PULU	(M_{usp}) → R U_{sp} + 1 → U_{sp}
	| Le post byte suivant ces deux types d'instructions précise le ou les registres concernés |.		
T F R	TRANSFERT ENTRE REG. DE MEME FORMAT . | Le post byte donne le registre destinataire et le registre source|.	T F R	R_1 → R_2
E X G	ECHANGE ENTRE REG. DE MEME FORMAT	E X G	R_1 ↔ R_2

INSTRUCTIONS SUR LES MVTS DES DONNEES :

INST.	FONCTION	MNEMONOQIE	OPERATION
ROL	ROTATION A GAUCHE DE UN BIT SUR A,B ou M.	ROL A ROL B ROL	A B M 
ROR	ROTATION A DROITE	ROR A ROR B ROR	A B M 
LSL(ASL)	DECALAGE LOGIQUE A GAUCHE	LSL A LSL B LSL	A B M 
ASR	DECALAGE ARITHMETIQUE A DROITE	ASR A ASR B ASR	A B M 
LSR	DECALAGE LOGIQUE A DROITE	LSR A LSR B LSR	A B M 

INSTRUCTIONS DE MODIFICATIONS DES DONNEES

INST.	FONCTION	MNEMONIQUE	OPERATION
CLR	METTRE L'ACC.A A"0" L'ACC.B A"0" LA MEMOIRE A"0"	CLR A CLR B CLR	00 → A 00 → B 00 → M
DEC	DECREMENTER UNE MEMOIRE ACC. A ACC. B	DEC DECA DECB	(M) - 1 → M (A) - 1 → A (B) - 1 → B
INC	INCREMENTER : MEMOIRE ACC. A ACC. B	INC INC A INCB	(M) + 1 → M (A) + 1 → A (B) + 1 → B
COM	COMPLEMENTER A "UN : MEMOIRE ACC. A ACC. B	 COM COM A COM B	 $(\bar{M}) \rightarrow M$ $(\bar{A}) \rightarrow A$ $(\bar{B}) \rightarrow B$
NEG	COMPLEMENTER A "DEUX" MEMOIRE ACC. A ACC. B	 NEG NEGA NEGB	 00 - (M) → M 00 - (A) → A 00 - (B) → B

INSTRUCTIONS ARITHMETIQUES

INST.	FONCTION	MNEMONIQUE	OPERATION
ADD	ADDITION ACC AVEC MEMOIRE M	ADDA ADDB	(A) + (M) → A (B) + (M) → B
	ADD. DOUBLE ACC AVEC MEMOIRE M	ADD	$(D_H) + (M)$ $(D_H) \rightarrow A$ $(D_L) + (M + 1) \rightarrow (D_L)$ B
ADC	ADD. ACC. AVEC BIT DE RETENUE	ADCA ADCB	(A) + (M) + C → A (B) + (M) + C → B
ABX	ADD. B ACC AVEC LE REG. X	ABX	$(B) + (X_L) \rightarrow X_L$ $0 + C + (X_H \rightarrow X_H$
D A	AJUSTER ADD EN DCB	DAA	CONVERTIT BINAIRE ADD. EN FORMAT BCD
SUB	SOUSTRACTION D'UNE MEMOIRE D'UN ACC.	SUBA SUBB	(A) (M) → A (B) (M) → B
	SOUSTRACTION DE MEMOIRE DU DOUBLE ACC.	SUBD	(D_M) (M) → (D_H) A $(D_L) - (M + 1) \rightarrow (D_L)$ B
SBC	SOUSTRACTION DE MEMOIRE ET LE BIT CARRY DE L'ACC.	SBCA SBCB	(A) - (M) - C → A (B) (M) C → B
SEX	EXTENDRE LE SIGNE DES DANS A	SEX	A_{cc} B BIT7 = 1 A = F F A_{cc} B BIT7 = 0 A = 0

INSTRUCTIONS SUR LE REGISTRE DE CODE CONDITION

INST	FONCTION	MNEMONIQUE	OPERATION
O R	MISE A "UN" DES BITS DU CCR [OU ENCLUSIF AVEC LE CONTENU DU C C R]	ORCC	(CC ; + (M) → (C C) E F H I N Z V C 0 0 0 0 0 0 0 0 + M = 0 0 0 0 1 1 1 1 E F H I N Z V C 0 0 0 0 1 1 1 1
AND	MISE A "ZERO" DES BITS DU C C R [ET LOGIQUE AVEC LE CONTENU DU CCR]	AND CC	(CC) . (M) → C C E F H I N Z V C 0 0 0 0 1 1 1 1 M = 1 1 1 1 0 0 0 0 E F H I N Z V C 0 0 0 0 0 0 0 0

INSTRUCTIONS DE CHARGEMENT D'ADRESSES EFFECTIVES :

INST.	FONCTION	MNEMONIQUE	OPERATION
L E A	CHARGER LE REGISTRE AVEC L'ADRESSE EFFECTIVE	LEAX LEAY LEAS LEAU	EA → X EA → Y EA → S EA → U

INSTRUCTIONS DE BRANCHEMENTS :

INST	FONCTION	MNEMONIQUE	BRANCH TEST
B R A	BRANCHEMENT INCONDITIONNEL	B P A	AUCUN
B R N	BRANCHEMENT INHIBE	B R N	AUCUN
B C C (BHS)	BRANCHEMENT SI CARRY NUL (SUPERIEUR OU EGAL)	B C C	C = 0
B C S(BLO)	BRANCHEMENT SI CARRY EST A "UN" |Si INFERIEUR|	B C S	C = 1
B E Q	BRANCHEMENT SI EGAL A "ZERO"	B E Q	Z = 1
BNE	BRANCH. S'IL N'EST PAS EGAL A "ZERO"	B N E	Z = 0
B G E	BRANCH. SI $\geq$ ZERO*	B G E	(N (+) V) = 0
B L T	BRANCH. SI < ZERO*	B L T	(N (+) V) = 1
B G T	BRANCH. SI > ZERO*	B G T	Z + (N(+)V)=0
B L E	BRANCH. SI $\leq$ ZERO*	B L E	Z + (N(+)V)=1
BH I	BRANCH. SI SUPERIEUR	B H I	(C + Z) = 0
B L S	BRANCH. SI INFERIEUR OU EGAL	B L S	(C + Z) = 1
B M I	BRANCH. SI NEGATIF	B M I	N = 1
B P L	BRANCH. SI POSITIF	B P L	N = 0
B V C	BRANCH. SI OVERFLOW EST NUL	B V C	V = 0
B V S	BRANCH. SI OVERFLOW EST A "UN"	B V S	V = 1
B S R	BRANCH. A UN SOUS PROGRAMME	B S R	AUCUN

*DONNEES SIGNEES SEULEMENT

INSTRUCTIONS DE SAUT :

INST.	FONCTION	MNEM.	OPERATION
J M P	SAUT A UNE ADRESSE	J M P	(M) → P C_H ; (M + 1) → P C_L
J S R	SAUT A UNE ADRESSE DE SUBROUTINE	J S R	SP −1→SPPC_L → (SP) (M) → P C_H SP−1→SPPC_H → (SP) (M+1) → P C_L
R T S	RETOUR DE SUBROUTINE	R T S	(SP) →PC_H SP + 1 → S P (SP) → PC_L SP + 1 → S P

INSTRUCTIONS DE TEST SUR LES DONNEES

INST.	FONCTION	MNEMONIQUE	TEST
BIT	BIT TEST (AND) ENTRE LE CONTENU de l'ACC. ET LA MEMOIRE	BIT A BIT B	(A) . (M) (B) . (M)
CMP	COMPARER L'ACC. AVEC MEMOIRE	CMP A CMP B	(A) - (M) (B) - (M)
	COMPARE MEMOIRE AVEC UN REGISTRE 16 BITS.	CMP D CMP X CMP Y CMP U CMP S	$(R_L) - (M+1)$ $(R_H) - (M_H)$
TST	COMPARE MEMOIRE A "ZERO" COMPARE ACC. A "ZERO"	TST TST A TST B	

INSTRUCTIONS LOGIQUES :

INST.	FONCTION	MNEM.	OPERATION
AND	ET LOGIQUE ENTRE LE CONTENU DE LA MEMOIRE ET L'ACC.	AND A AND B	(A) . (M) → A (B). (M) → B
EOR	OU LOGIQUE EXCLUSIF ENTRE M ET ACC.	EOR A EOR B	(A) (+) (M) → A (B) (+) (M) → B
O R	OU LOGIQUE ENTRE M ET ACC.	OR A OR B	(A) + (M) → A (B) + (M) → B

INSTRUCTIONS SPECIALES :

INST.	FONCTION	MNEMONIQUE	OPERATION
MUL	MULTIPLICATION NON SIGNE DES ACCUMULATEURS	MUL	$A^* B \rightarrow D$
NOP	AUCUNE OPERATION (FONCTION : NE RIEN FAIRE)	NOP	PC + 1→PC

INSTRUCTIONS SUR LES INTERRUPTIONS

INST	FONCTION	MNEMONIQUE	OPERATION
SWI	INTERRUPTION PAR LOGICIEL	S W I	E = 1 REG → M_{SP} SP - C → S P I = 1 F = 1 (M) = FFFA → P_{CH} (M) - FFF B → P_{CL}
		S W I 2	E = 1 REG S → M_{SP} SP - C → S P (M) FFF4 = P_{CH} (M) FFF5 = P_{CL}
		S W I 3	E = 1 REG S → M_{SP} SP - C → S P (M) FFF2 = P_{CH} (M) FFF3 = P_{CL}
RTI	RETOUR D'INTERRUPTION	R T I	(M SP) → C C SP + 1 → S P I F A = 1 (M_{SP}) → REG S SP + C → S P I F E = C (M SP) → P_{CH} SP + 1 → S P M_{SP} → P_{CL} SP + 1 → S P
CWA I	METTRE A ZERO ET ATTENDRE L'INTERRUPTION	C W A I	C C = M E = 1 REG S → (M_{SP}) SP - C → SP
SYNC	ATTENDRE UNE INTERRUPTION	SYN C	ARRETER MPU ET ATTENDRE UNE INTERRUPTION

6809 Instruction Codes, Memory Requirements, and Execution Times

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative		
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement		
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes
ABX	3A	3	1															
ADCA				89	2	2	99	4	2	B9	5	3	A9	4+	2+			
ADCB				C9	2	2	D9	4	2	F9	5	3	E9	4+	2+			
ADDA				8B	2	2	9B	4	2	BB	5	3	AB	4+	2+			
ADDB				CB	2	2	DB	4	2	FB	5	3	EB	4+	2+			
ADDD				C3	4	3	D3	6	2	F3	7	3	E3	6+	2+			
ANDA				84	2	2	94	4	2	B4	5	3	A4	4+	2+			
ANDB				C4	2	2	D4	4	2	F4	5	3	E4	4+	2+			
ANDCC				1C	3	2												
ASL							08	6	2	78	7	3	68	6+	2+			
ASLA	48	2	1															
ASLB	58	2	1															
ASR							07	6	2	77	7	3	67	6+	2+			
ASRA	47	2	1															
ASRB	57	2	1															
BCC																24	3	2
BCS																25	3	2
BEQ																27	3	2
BGE																2C	3	2
BGT																2E	3	2
BHI																22	3	2
BHS																24	3	2
BITA				85	2	2	95	4	2	B5	5	3	A5	4+	2+			
BITB				C5	2	2	D5	4	2	F5	5	3	E5	4+	2+			
BLE																2F	3	2
BLO																25	3	2
BLS																23	3	2
BLT																2D	3	2
BMI																2B	3	2
BNE																28	3	2
BPL																2A	3	2
BRA																20	3	2
BRN																21	3	2
BSR																8D	7	2
BVC																28	3	2
BVS																29	3	2
CLR							0F	6	2	7F	7	3	6F	6+	2+			
CLRA	4F	2	1															
CLRB	5F	2	1															
CMPA				B1	2	2	91	4	2	B1	5	3	A1	4+	2+			
CMPB				C1	2	2	D1	4	2	F1	5	3	E1	4+	2+			
CMPD				10 83	5	4	10 93	7	3	10 B3	8	4	10 A3	7+	3+			

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative			notes
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement			
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	
CMPS				11 8C	5	4	11 9C	7	3	11 BC	8	4	11 AC	7+	3+				
CMPU				11 83	5	4	11 93	7	3	11 B3	8	4	11 A3	7+	3+				
CMPX				8C	4	3	9C	6	2	BC	7	3	AC	6+	2+				
CMPY				10 8C	5	4	10 9C	7	3	10 BC	8	4	10 AC	7+	3+				
COM							03	6	2	73	7	3	63	6+	2+				
COMA	43	2	1																
COMB	53	2	1																
CWAI				3C	20	2													
DAA	19	2	1																
DEC							0A	6	2	7A	7	3	6A	6+	2+				
DECA	4A	2	1																
DECB	5A	2	1																
EORA				88	2	2	98	4	2	B8	5	3	A8	4+	2+				
EORB				C8	2	2	D8	4	2	F8	5	3	E8	4+	2+				
EXG				1E	8	2													
INC							0C	6	2	7C	7	3	6C	6+	2+				
INCA	4C	2	1																2
INCB	5C	2	1																
JMP							0E	3	2	7E	4	3	6E	3+	2+				
JSR							9D	7	2	BD	8	3	AD	7+	2+				
LBCC																10 24	5(6)	4	1
LBCS																10 25	5(6)	4	1
LBEQ																10 27	5(6)	4	1
LBGE																10 2C	5(6)	4	1
LBGT																10 2E	5(6)	4	1
LBHI																10 22	5(6)	4	1
LBHS																10 24	5(6)	4	1
LBLE																10 2F	5(6)	4	1
LBLO																10 25	5(6)	4	1
LBLS																10 23	5(6)	4	1
LBLT																10 2D	5(6)	4	1
LBMI																10 2B	5(6)	4	1
LBNE																10 26	5(6)	4	1
LBPL																10 2A	5(6)	4	1
LBRA																16	5	3	
LBRN																10 21	5	4	
LBSR																17	9	3	
LBVC																10 28	5(6)	4	1
LBVS																10 29	5(6)	4	1
LDA				86	2	2	96	4	2	B6	5	3	A6	4+	2+				
LDB				C6	2	2	D6	4	2	F6	5	3	E6	4+	2+				
LDD				CC	3	3	DC	5	2	FC	6	3	EC	5+	2+				

Address Mode →	Inherent			Immediate			Direct			Extended			Indexed/Indirect			Relative			notes
Operand Form →				data8 or data16			adr8			adr16			See Appendix B			label or displacement			
Instruction Mnemonic ↓	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	Object Code	No. of Cycles	No. of Bytes	
LDS				10 CE	4	4	10 DE	6	3	10 FE	7	4	10 EE	6+	3+				
LDU				CE	3	3	DE	5	2	FE	6	3	EE	5+	2+				
LDX				8E	3	3	9E	5	2	BE	6	3	AE	5+	2+				
LDY				10 8E	4	4	10 9E	6	3	10 BE	7	4	10 AE	6+	3+				
LEAS													32	4+	2+				
LEAU													33	4+	2+				
LEAX													30	4+	2+				
LEAY													31	4+	2+				
LSL																			
LSLA	48	2	1				08	6	2	78	7	3	68	6+	2+				
LSLB	58	2	1																
LSR																			
LSRA	44	2	1				04	6	2	74	7	3	64	6+	2+				
LSRB	54	2	1																
MUL	3D	11	1																
NEG																			
NEGA	40	2	1				00	6	2	70	7	3	60	6+	2+				
NEGB	50	2	1																
NOP	12	2	1																
ORA																			
ORB				8A	2	2	9A	4	2	BA	5	3	AA	4+	2+				
ORCC				CA	2	2	DA	4	2	FA	5	3	EA	4+	2+				
PSHS				1A	3	2													
PSHU				34	5+	2													2, 3
PULS				36	5+	2													2, 3
PULU				35	5+	2													2, 3
ROL				37	5+	2													2, 3
ROLA	49	2	1				09	6	2	79	7	3	69	6+	2+				
ROLB	59	2	1																
ROR																			
RORA	46	2	1				06	6	2	76	7	3	66	6+	2+				
RORB	56	2	1																
RTI	3B	6/15	1																
RTS	39	5	1																
SBCA				82	2	2	92	4	2	B2	5	3	A2	4+	2+				
SBCB				C2	2	2	D2	4	2	F2	5	3	E2	4+	2+				
SEX	1D	2	1																
STA							97	4	2	B7	5	3	A7	4+	2+				
STB							D7	4	2	F7	5	3	E7	4+	2+				
STD							DD	5	2	FD	6	3	ED	5+	2+				
STS							10 DF	6	3	10 FF	7	4	10 EF	6+	3+				
STU							DF	5	2	FF	6	3	EF	5+	2+				
STX							9F	5	2	BF	6	3	AF	5+	2+				
STY							10 9F	6	3	10 BF	7	4	10 AF	6+	3+				
SUBA				80	2	2	90	4	2	B0	5	3	A0	4+	2+				
SUBB				C0	2	2	D0	4	2	F0	5	3	E0	4+	2+				
SUBD				83	4	3	93	6	2	B3	7	3	A3	6+	2+				
SWI	3F	19	1																
SWI2	10 3F	20	2																
SWI3	11 3F	20	2																
SYNC	13	2	1																
TFR				1F	7	2													
TST							0D	6	2	7D	7	3	6D	6+	2+				2
TSTA	4D	2	1																
TSTB	5D	2	1																

Note 1: The cycle count in parentheses applies if the branch is taken.

Note 2: The immediate data in this instruction's object code is always an encoded register specification. See the description of this instruction in Chapter 22 for details.

Note 3: A PSH or PUL instruction requires 5 cycles plus one cycle for each byte pushed or pulled.

PROGRAMME DE COMMANDE DES MOTEURS:

Programme source:

```
          EBF8  PIADAI   EQU   $EBF8
          EBF8  PIADDAI  EQU   $EBF8
          EBF9  PIACAI   EQU   $EBF9
          EBFA  PIADBI   EQU   $EBFA
          EBFA  PIADDBI  EQU   $EBFA
          EBFB  PIACBI   EQU   $EBFB
          EBFC  PIADA2   EQU   $EBFC
          EBFC  PIADDA2  EQU   $EBFC
          EBFD  PIACA2   EQU   $EBFD
          EBFE  PIADB2   EQU   $EBFE
          EBFE  PIADDB2  EQU   $EBFE
          EBFF  PIACB2   EQU   $EBFF
                *
0000                     ORG   $0000
0000 86   EC             LDA   #%EC
0002 IE   8B             EXG   A,Dp      Initialiser le DPR
0004 IOCE EC70           LDS   #%EC70    Inititialiser le pointeur S
0008 IC   AF             ANDCC #%AF      Valider les interruptions
000A 7F   EBFA           CLR   PIACA2
000D 7F   EBFF           CLR   PIACB2
00I0 7F   EBF9           CLR   PIACAI
00I3 7F   EBFB           CLR   PIACBI
00I6 7F   EBFC           CLR   PIADDA2 PortAdu PIA2 Entrant
00I9 86   FF             LDA   #%FF
00IB B7   EBFE           STA   PIADDB Port B duPIA2 Sortant
00IE B7   EBF8           STA   PIADDAI Port A du PIAI Sortant
002I B7   EBFA           STA   PIADDBI  Port B du PIAI Sortant
0024 86   04             LDA    %04
0026 B7   EBF9           STA   PIACAI
0029 B7   EBFB           STA   PIACBI
002C B7   EBFA           STA   PIACA2
002F B7   EBFF           STA   PIACB2
0032 86   64             LDA   #%64
0034 97   I2             STA    $I2
0036 86   0A             LDA   #%0A
0038 97   I3             STA    $I3
003A 86   46             LDA   #%46
003C 8E   EC00           LDX    %EC00
003F A7   80     Boucle  STA    0,X     Initialisation des positions
004I 8C   EC07           CMPX #%EC07    mémoires servant à stocker
0044 26   F9             BNE   Boucle   la position effective de
0046 0F   07             CLR    $07     chaque organe.
0048 0F   08             CLR    $08
004A 0F   09             CLR    $09
004C 86   0F             LDA   #%0F
004E 97   09             STA    $09
0050 97   0C             STA    $ 0C
0052 86   IE             LDA   #%IE
0054 97   0A             STA    $0A
0056 97   0D             STA    $0D
```

```
0058 B6 EBF9   Init.  LDA   PIACAI   Début de la commande
005B 84 80            ANDA  #%80
005D 27 F9            BEQ   Init     Oui,attente
005F 0F I0     Réinit. CLR   $ I0
006I 7F EBFA          CLR   PIADBI   RAZ des afficheurs
0064 86 0F            LDA   #% 0F
0066 B7 EBF8          STA   PIADAI
0069 8E EBFF          LDX   #%EBFF
006C B6 EBF9   TSEL   LDA   PIACAI   Demmande de séléction?
006F 84 80            ANDA  #%80
007I 27 27            BEQ   Cnv/aff
0073 86 08            LDA   #%08
0075 B7 EBF8          STA   PIADAI
0078 0C I0            INC   $ I0
007A 96 I0            LDA   $ I0
007C 8I 0F            CMPA  #% 0F
007E 27 DF            BEQ   Réinit
0080 F6 EBF8          LDA   PIADAI   Remettre CRA-7 àzéro
0083 48               ASLA
0084 48               ASLA
0085 48               ASLA
0086 48               ASLA
0087 97 ID            STA   $ ID
0089 B7 EBFA          STA   PIADBI   Allumer la LED qui
008C 30 IF            LEAX  -I,X     correspond à l'organe
008E 8I I0            CMPA  #%I0
0090 27 04            BEQ   IMP
0092 04 IB            LSR   $ IB
0094 20 04            BRA   Cnv/aff
0096 86 80     IMP    LDA   #%08
0098 97 IB            STA   $ IB
009A 0F I7     Cnv/aff CLR  $ I7
009C 0F I8            CLR   $ I8
009E 0F I9            CLR   $ I9
00A0 86 09            LDA   #%09
00A2 97 I4            STA   $ I4
00A4 86 0A            LDA   % 0A
00A6 97 I5            STA   $ I5
00A8 86 0C            LDA   #% 0C
00AA 97 I6            STA   $ I6
00AC I08E ECI2        LDY   %ECI2
00B0 E6 00            LDB   0,X      Chercher la valeur de
00B2 C5 80     Lp2    BITB  #% 80    la position de l' organe
00B4 26 I9            BNE   LpI      séléctionné et l'afficheur
00B6 86 03            LDA   #%03
00B8 4A        Lp3    DECA
00B9 27 IA            BEQ   Lp7
00BB A0 A4     Lp4    SUBB  0,Y
00BD 2B 0C            BMI   Lp6
00BF 8I 02            CMPA  #% 02
00CI 27 04            BEQ   Lp5
00C3 0C I8            INC   $ I8     Incrémenter les dizaines
00C5 20 F4            BRA   Lp4
00C7 0C I9     Lp5    INC   $ I9     Incrémenter les centaines
00C9 20 F0            BRA   Lp4
00CB EB A0     Lp6    ADDB  0,Y
```

```
00CD E0   A4      LpI     SUBB   0,Y
00CF 0C   I9              INC    $ I9     Incrémenter les centaines
00DI 20   DD              BRA    Lp2
00D3 D7   I7      Lp7     STB     $ I7
00D5 86   08      Affich  LPA    #% 08
00D7 B7   EBFA            STA    PIADAI
00DA E6   23              LDB     3,X
00DC DB   ID              ADDB    $ ID
00DE F7   EBFA            STB    PIADBI
00EI E6   A0              LDB     ,Y
00E3 F7   EBF8            STB    PIADAI
00E6 I08C ECI7            CMPY   #%ECI7
00EA 26   E9              BNE    Affich
00EC B7   EBF8            STA    PIADAI
00EF B6   EBFC            LDA    PIADA2
00F2 8I   FF              CMPA   #% FF    Demande de mouvement?
00F4 I027 FF74            LBEQ    TSEL    Non, tester la séléction
00F8 8I   F9              CMPA   #%F9     Oui,tester le type de Mvt
00FA 27   0F              BEQ    Lp8      Avance RAPIDE?
00FC 8I   FC              CMPA   #%FC     Recul RAPIDE?
00FE 27   37              BEQ    LpII
0I00 8I   FB              CMPA   #%FB     Avance LENTE?
0I02 27   5F              BEQ    LpI4     Non,c'est un Recul LENT.
0I04 CC   4000            LDD    #%4000   Valeur de Temporisation I
0I07 DD   20              STD     $ 20
0I09 20   3I              BRA    TMTR     Mvt de RECUL
0I0B CC   FFFF    Lp8     LDD    #%FFFF   Valeur de Temporisation 2
0I0E DD   20              STD     $ 20
0II0 96   I0      TMTA    LDA     $ I0
0II2 8I   08              CMPA   #% 08    Test sur l'organe séléct.
0II4 22   IC              BHI    LpI6
0II6 E6   00              LDB     0,X
0II8 8I   08              CMPA   #%08
0IIA 27   0D              BEQ    LpI0
0IIC CI   F0              CMPB   #%F0     Positon max. des DETECT.
0IIE 25   05              BLO    Lp9      en mouvement de translat
0I20 BD   0IF2            JSR    SB BLANK Clignoter les afficheurs
0I23 I6   FF72            LBRA    Cnv/aff
0I26 6C   00      Lp9     INC     0,X
0I28 20   4I              BRA     IMP I    Envoi d'une impulsion
0I2A CI   B4      LPI0    CMPB   #%B4     Positionmax. de la cible
0I2C 25   F8              BLO    Lp9      en rotation?
0I2E BD   0IE7            JSR    SB BELAY Temporisation
0I3I I6   FF65            LBRA    Cnv/aff
0I34 BD   0ID6    LpI6    JSR     SB COM   Génerer une impulsion
0I37 20   47              BRA     TMTGA    Test des DETECT.de Groupe A
0I39 CC   FFFF    LpII    LDD    #%FFFF
0I3C DD   20              STD     $ 20
0I3E 96   I0      TMTR    LDA     $ I0     Test des Mvts de rotation
0I40 8I   08              CMPA   #%08
0I42 27   0D              BEQ     1PI2
0I44 CI   46              CMPB   #%46
0I46 22   07              BHI     LpI3
0I48 BD   0IF2            JSR     SB BLANK
0I4B I6   FF44            LBRA    Cnv/aff
0I4E 6A   00      LpI3    DEC     ,X
```

```
0I50 20    I5              BRA    IMPI      Temporisation
0I52 CI    00    LpI2      CMPB   # % 00
0I54 22    F8              BHI    LpI3
0I56 BD    0IF2            JSR    SB BLANK
0I59 I6    FF36            LBRA   Cnc/aff
0I5C BD    0ID6            JSR    SB COM
0I5F 20    45              BRA    TMTGR     Test de Mvt de Groupe
oI6I CC    4000  LpI4      LDD    #%4000
0I64 DD    20              STD    $ 20
0I66 20    A9              BRA    TMTA
0I68 BD    0IE7  IMP I     JSR    SB DELAY  Temporisation
0I6B 96    II              LDA    $ II
0I6D B7    EBFE            STA    PIADB2
0I70 CC    0FFF  LpEI      LDD    #%0FFF    Duree de l'ipulsion
0I73 DD    20              STD    $ 20
0I75 BD    0IE7            JSR    SB DELAY  Temporisation
0I78 7F    EBFE            CLR    PIADB2
0I7B I6    FFI4            LBRA   Cnv/aff
0I7E 96    I0    TMTGA     LDA    $ I0      Test de l'AVANCE des
oI80 8I    0C              CMPA   #% 0C     moteurs de Groupe de
0I82 25    II              BLO    LpC2      DETECTEURS
0I84 96    0B              LDA    #% 0B     Test de la position de
0I86 8I    8C              CMPA   #% 8C     DETECTEURS en rotation
oI88 25    05              BLO    LpC3
0I8A BD    0IF2  LpCI      JSR    SB BLANK
0I8D I6    FF03            BRA    Cnv/aff
oI90 0C    0B    LpC3      INC    $ 0B
0I92 0C    0C              INC    $ 0C
0I94 0C    0D              INC    $ 0D
0I96 20    36              BRA    IMP 2
0I98 96    08    LpC2      LDA    $08
0I9A 8I    8C              CMPA   #% 8C
0I9C 25    02              BLO    LpC4
0I9E 20    DE              BRA    LpCI
0IA0 0C    08              INC    $ 08
0IA2 0C    09              INC    $ 09
0IA4 0C    0A              INC    $ 0A
0IA6 20    26              BRA    IMP 2     Génerer une impulsion pour
0IA8 96    I0    TMTGR     LDA    $ I0      les moteurs de Groupe de
0IAA 8I    0C              CMPA   #%0C      DETECTEURS
0IAC 25    0D              BLO    LpDI
0IAE 96    0B              LDA    $ 0B      Test de la position du GR.2
oIB0 8I    00              CMPA   #% 00     de DETECTEURS EN ROTATION
0IB2 22    02              BHI    LpC2
oIB4 20    2C              BRA    LpCI
0IB6 0A    0B              DEC    $ 0B
0IB8 0A    0C    LpD2      DEC    $ 0C
0IBA 0A    0D              DEC    $ 0D
0IBC 20    I0              BRA    IMP 2
0IBE 96    08    LpDI      LDA    $ 08
0IC0 8I    00              CMPA   #% 00
0IC2 22    02              BHI    LpD3
0IC4 20    3C              BRA    LpCI
0IC6 0A    08    LpD3      DEC    $ 08
0IC8 0A    09              DEC    $ 09
0ICA 0A    0A              DEC    $ 0A
```

```
0ICC BD   01E7  IMP 2       JSR    SB DELAY   Temporisation
0ICF 96   I8                LDA     $ I8
0ID1 B7   EBF8              STA    PIADAI     Générer une
0ID4 20   99                BRA    LpEI        impulsion
                *
          0ID6  SB COM      EQU*    $ 0ID6
          0IE7  SB DELAY    EQU*    $ 0IE7
          0IF2  SB BLANK    EQU     $ 0IF2
                *
0ID6                        ORG     $ 0ID6
0ID6 96   I0                LDA     $ I0
0ID8 8I   0C                CMPA    #% 0C
0IDA 25   04                BLO    LpI9
0IDC 86   I0                LDA    #% I0
0IDE 20   02                BRA    Lp20
0IE0 86   22          LpI9  LDA    #% 20
0IE2 8B   08          Lp20  ADDA   #% 08
0IE4 97   I8                STA     S I8
0IE6 39                     RTS
                *
0IE7                        ORG     $ 0IE7
0IE7 34   I0          DELAY PSHS      X
0IE9 9E   20                LDX     $ 20
0IEB 30   IF            Dly LEAX    -I,X
0IED 26   FC                BNE    Dly
0IEF 35   90                PULS   PC,X
0IFI 39                     RTS
                *
0IF2                        ORG      $ 0IF2
0IF2 7F   EBF8              CLR     PIADAI
0IF5 CC   0FFF              LDD    #% 0FFF     Durée de l'impu-
0IF8 DD   20                STD     $ 20        lsion
0IFA BD   0IE7              JSR    DELAY
0IFD 39                     RTS
                *
```

INTERFACE PARALLELE PROGRAMMABLE

PIA 6820 et 6821

GENERALITES :

Le P I A (PERIPHERAL INTERFACE ADAPTOR) est l'interface universalle programmable permettant de communiquer avec les microprocesseurs. Il permet l'interface avec les circuits suivants :

- Claviers, codés ou non :
- Afficheurs LED ou 7 segments
- Convertisseur N - A
- Tables traçantes, etc.

C'est un circuit à 40 broches, réalisé en technologie N-MOS et monotension : 5 V

C'est un circuit pratiquement symétrique comportant 2 ports de communication appelée PORT A et PORT B. Chaque port comprend 8 lignes programmables en entrée-sortie, et ceci une à une le sens des échanges est fixé par le contenu d'un registre de direction de données (DDR). Quatre lignes de contrôle (CA1 et CA2, CB1 et CB2) permettent le dialogue avec l'extérieur, leur fonctionnement est fixé par les registres de contrôle (CR). En fin les données transmises à l'extérieur sont mis temporaiement dans un registre de sortie (DR).

Les échanges avec le µ P se font par l'intermédiaire de :

- bus de données D0 - D7 pour programmer les registres de contrôle (CRA et CRB), les registres de direction de données (DDRA et DDRB) et lire les données reçues sur un port ou transmettre des données,

- 3 lignes de validation de boitiers CS0, CS1, $\overline{CS2}$.

- 2 entrées de sélection de registres RS0 et RS1 permettent de distinguer les registres internes.

- L'entrée ENABLE reçoit un signal d'horloge généralement ϕ_2 pour assurer les échanges synchrones.

- L'entrée R/W fixe le sens de transferts : lecture du PIA ou écriture

- Une entrée $\overline{\text{RESET}}$, sensible au niveau bas, reçoit le signal $\overline{\text{RESET}}$ général de la carte microsystème, ce qui permet la mise à zéro de tous les registres internes du PIA.

- Deux lignes d'interruption IRQA et IRQB permettent d'interrompre le programme en cours et d'appeler un programme de traitement d'interruption.

INTERCONNECTION DU BOITIER ET ADRESSAGE

Le schéma de la fig. 1 montre l'interconnection du PIA avec les bus du µ P et l'extérieur.

Adressage :

Vis à vis du µ P le PIA se comporte seulement comme 4 positions mémoire bien qu'il comporte 6 registres internes.

On peut donc appliquer au PIA toutes les instructions utilisables avec les mémoires, en effet les registres DDRx et DRx ont la même adresse, le bit b_2 du registre de contrôle correspondant permet la distinction entre ces deux registres. TALBEAU 1).

Il en résulte qu'avant de programmer DDRx ou DRx il faudra programmer CRx, quitte à modifier ces derniers par la suite.

Les adresses étant consécutives, RS0 et RS1 reçoivent respectivement AD et A1 du bus d'adresse.

TABLEAU 1.

RS1	RS0	CRA-2	CRB-2	ADRESSE	REGISTRE ADRESSE
0	0	0	X	ADR	Registre de direction (DDRA)
0	0	1	X	ADR	Registre de sortie A (DRA)
0	1	X	X	ADR + 1	Registre de contrôle (CRA)
1	0	X	0	ADR + 2	Registre de direction B (DDRB)
1	0	X	1	ADR + 2	Registre de sortie B (ORB)
1	1	X	X	ADR + 3	Registre de contrôle (CRB)

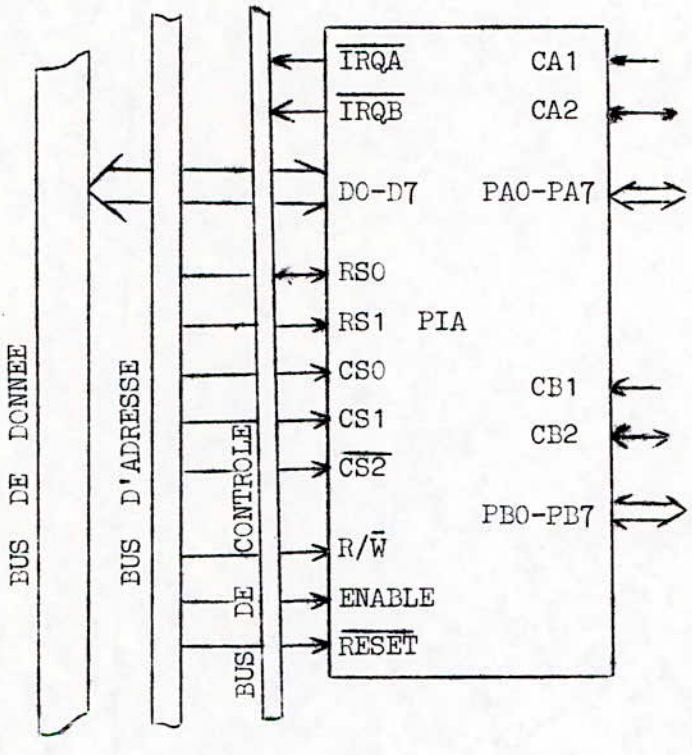


Fig. 1

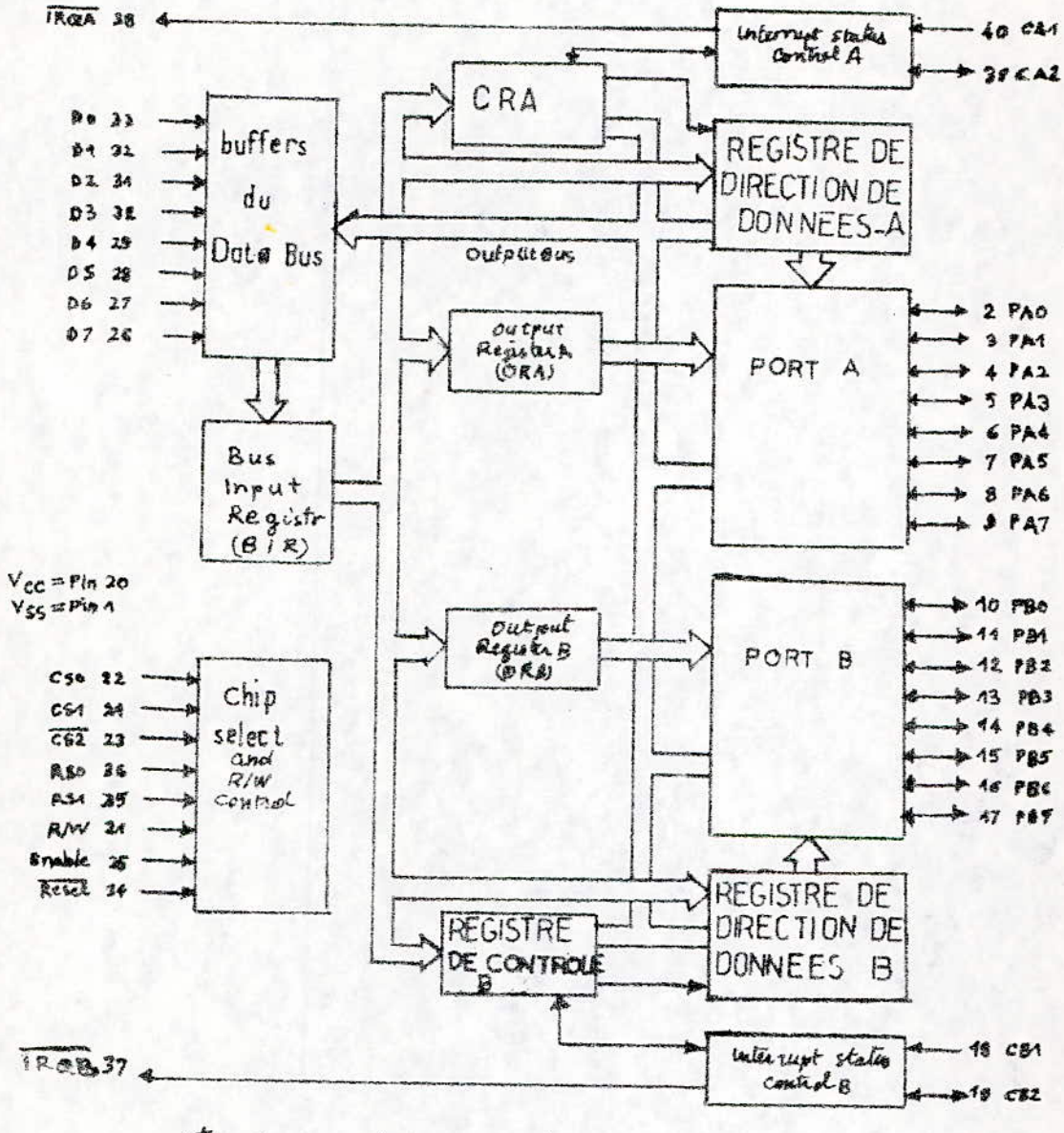
ORGANISATION INTERNE DU PIA :

(VOIR SCHEMA DE LA FIG. 2)

Il comporte essentiellement 6 registres :

* CRA - CRB : Registres de commande qui fixent le fonctionnement des lignes CA1 - CA2 et CB1 - CB2.

FIG·2



structure interne du PIA MC6821

* DDRA - DDRB : Registres de direction de données à écriture seule

* ORA - ORB : Registres de sorties des ports A et B. Ils mémorisent les informations envoyés à l'extérieur sur les ports A et B.

les tableaux suivants montrent la programmation des différents registres.

Tableau : Registres de contrôle et d'état ; Rôle et signification des bits CRXX.

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CONTROLE DE CA2			ACCES DDRA	CA1 CONTROLE	

	7	6	5	4	3	2	1	0
CRB	IRQB1	IRQB2	CONTROLE DE CB2			ACCES DDRB	CB1 CONTROLE	

Tableau : Contrôle de l'entrée d'interruption CA1 et CB1

CRA-1 (CRB-1)	CRA-0 (CRB-0)	Entrée CA1 (CB1)	Bit indicateur CRA7 (CRB-7)	Demande d'interruption $\overline{IRQ}$
0	0	Active ↓	Mis à 1 sur ↓ de CA1 (CB1)	Inhibe l'interruption $\overline{IRQ}$
0	1	Active ↓	Mis à 1 sur ↓ de CA1 (CB1)	IRQ mis à 0 quand CRA-7 est à 1 (CRB-7)
1	0	Active ↑	Mis à 1 sur ↑ de CA1 (CB1)	Inhibe $\overline{IRQ}$
1	1	Active ↑	Mis à 1 sur ↑ de CA1 (CB1)	IRQ mis à 0 quand CRA-7 est à 1 (CRB.7)

Notes :
1. ↑ indique une transition positive
2. ↓ indique une transition négative
3. Le bit CRA-7 (CRB-7) est remis à zéro par une lecture de registre DATA A (ou B).

CONTROLE DE CA2 et CB2 COMME
ENTREES D'INTERRUPTION

CRA-5 (CRB-5)	CRA-4 (CRB-4)	CRA-3 (CRB-3)	Entrée d'int. CA2 (CB2)	Bit indicateur CRA-6(CRB-6)	Demande d'interruption du MPU $\overline{IRQA}$ ($\overline{IRQB}$) -
0	0	0	↓ Active	Mis à 1 sur ↓ de CA2 (CB2)	Inhibé - $\overline{IRQ}$ passe à l'état haut
0	0	1	↓Active	Mis à 1 sur ↓ de CA2 (CB2)	Passe à l'état bas quand CRA-6(CRB-6) passe à 1.
0	1	0	↑ Active	Mis à 1 sur ↑ de CA2 (CB2)	Inhibé - $\overline{IRQ}$ passe à l'état haut.
0	1	1	↑ Active	Mis à 1 sur ↑ de CA2 (CB2)	Passe à l'état bas quand CRA-6 (CRB-6) passe à 1

CONTROLE DE CB2 COMME
SORTIE

CRB-5	CRB-4	CRB-3	CB2 MIS A 0	CB2 MIS A UN
1	0	0	BAS SUR LA PREMIRE TRANSITION POSITIVE de E qui suit une écriture du registre de données B	HAUT QUAND CRB-7 est mis à 1 pour une transition active du signal CB1
1	0	1	BAS SUR LA TRANSITION positive de E qui suit une écritu - re du registre de données B	HAUT SUR LA TRANSITION POSITIVE DE LA PROCHAINE IMPULSION DE E
1	1	0	BAS quand CRB-3 est mis à ZERO pour une écriture du registre de contrô le B	BAS TANT QUE CRB3 est bas Il passe à l'état haut quand CRB-3 sera mis à 1 pour une écriture dans le registre de CR-B.
1	1	1	Toujours HAUT TANT QUE CRB-3 est haut passe à l'état bas quand CRB-3 est mis à 0 par une écriture.	HAUT QUAND CRB-3 est mis à 1 par une écriture du registre de contrôle B (CRB)

CONTROLE DU CA2 COMME

SORTIE

CRA-5	CRA-4	CRA-3	CA2 MIS A ZERO	CA2 MIS A UN
1	0	0	BAS SUR LA TRANSITION NEG. DE LA PREMIERE IMPULSION de E qui suit une lecture du Registre D.A	HAUT SUR LA TRANSITION ACTIVE DU SIGNAL CA1
1	0	1	BAS imméd. après une lecture du MPU du Registre Don. A	HAUT SUR LA TRANSITION NEG. DE LA PROCHAINE IMPULSION DE E.
1	1	0	BAS QUAND CRB-3 EST MIS A ZERO PAR UNE ECRITURE DE REG. CON. A	TOUJOURS BAS TANT QUE CRA-3 est bas.
1	1	1	TOUJOURS HAUT TANT QUE CRB-3 est haut.	HAUT QUAND CRA-3 passe à 1 comme l'effet d'une écriture dans le registre de contrôle A.

PROGRAMMES PERMETTANT L'UTILISATION D'UN PIA COMME ORGANE DE LIAISON AVEC LA TTY

Programme I: LECTURE DE DONNEES DE LA TTY

On utilise le bit 7 du PIA PORT A et on place la donnée dans une position mémoire EC00.

```
                EBF9    PIACA   EQU     $ EBF9
                EBF8    PIADDA  EQU     $ EBF8
                EBF8    PIADA   EQU     $ EBF8
                        *
0000                            ORG     $ 0000
0000 7F         EBF9            CLR     PIACA
0003 7F         EBF8            CLR     PIADDA  Mettre toutes les
0006 86         04              LDA     #%04    lignes sortantes
0008 B7         EBF9            STA     PIACA
000B B6         EBF8    WTSTB   LDA     PIADA   Y'a-t'il le bit START?
000E 2B         FB              BMI     WTSTB   NON,attendre
0010 BD         0030            JSR     DLY2    Oui, Délai d'un temps
                                                d'un demi bit
0013 86         80              LDA     #% 80   Compter avec 'I'bit MSB
0015 BD         0035    TTYRCV  JSR     DELAY   Attendre un temps-bit
0018 79         EBF8            ROL     PIADA   Prendre le bit suivant
0019 46                         RORA            Combiner avec le prece-
001C 24         F7              BCC     TTYRCV  dent bit
001E 97         60              STA     $ 60    Transfert trmine'
0020 3F                 *       SWI

(Programme de DELAY)
0030 8E     0236        DLY2    LDX     #%0236  Temporisation de 4.55ms
0033 20     03                  BRA     DLY
0035 8E     046C        DELAY   LDX     #%046C  Temporisation de 9.1ms
0038 30     1F          DLY     LEAX    -1,X
003A 26     FC                  BNE     DLY
003C 39                         RTS
                        *
```

Programme 2: ECRITURE DE DONNEES SUR TTY

La donnée est dans la mémoire 0060
On suppose que la parité n'est pas utile à générer.

```
0000                            ORG
0000 7F         EBF9            CLR     PIACA
0003 86         FF              LDA     #%FF
0005 B7         EBF8            STA     PIADDA
0008 86         04              LDA     #%04
000A B7         EBF9            STA     PIACA
000D 96         60              LDA     $ 60    Prendre la donnée
000F C6         0B              LDB     #11     COMPTEUR=11 caracteres
0011 7F         EBF8            CLR     PIADA   Envoi du bit START
0014 9D         35      TBIT    JSR     DELAY   Attendre un temps-bit
0016 1A         01              ORCC    #%01    Mettre le Carry à 'I'pour
                                                former le bit STOP
0018 46                         RORA            Prendre le bit suivant
0019 79         EBF8            ROL     PIADA   Envoi du bit suivant
001C 5A                         DECB
001D 26         F5              BNE     TBIT
001F 3F                         RTS
```

C O N C L U S I O N

La réalsiation d'un micro-ordinateur autour du µ P MC 6809 dans le cadre de notre projet de fin d'étude, nous a aidé à acquérir une connaissance suffisante des microprocesseurs et à assimiler des notions sur les méthodes de leur programmation plus évoluées.

Cette réalisation n'a pas été sans nous poser de sérieuses difficultés,d'autant plus que le µ P 6809 n'a jamais été utilisé dans le cadre des activités du laboratoire d'Electronique du C.E.N.

En s'appuyant sur la compatibilité partielle avec le 6800, on a pu réaliser un micro-ordinateur monocarte possédant tous les signaux de base nécessaires aux modules qu'on pourrait éventuellement associer.

Ne disposant pas d'un système de développement à base du 6809, nous avons en premier lieu fait tourner les programmes de test avec l'EXORCISER. A cause de la non compatibilité et de la non équivalence totale du logiciel des deux microprocesseurs nous les avons ensuite adapter au 6809 ce qui nous a permis de nous familiariser avec le 6800, mais ceci ne va pas sans augmenter les sources d'erreurs dans la programmation.

L'utilisation de notre carte pour la commande à distance de la chambre à cible n'est qu'une première application loin de refléter les capacités réelles de traitement du 6809 ; ceci nous permet entre autre de tester le bon fonctionnement de cette carte. Le système de commande ainsi réalisé sera directement exploitable en mettant un étage de puissance aux lignes aboutissants aux moteurs.

Enfin, nous espérons que cet ouvrage contribuera à acquérir des connaissances utiles des µ P et aider toute personne intéressée par le 6809 dans le but d'ouvrir la voie pour son intégration dans des systèmes à microprocesseurs et préparer ainsi l'utilisation du µ P 68000 le plus puissant de sa génération.

BIBLIOGRAPHIE

LANCE . A . LEVENTHAL
- 6809 Assembly language Programming

G. REVELLIN
- Microprocesseurs : du 6800 au6809 etmodes d'interfaçage

D. GIROD (2° Edition)
- Au coeur des microprocesseurs

H. LILEN
-Guide mondial des microprocesseurs

REVUES
-Haut parleur
-Eléctronics

DOCUMENTS TECHNIQUES

FEUTRIER
- Cours du 6809
- Data book TTL (TEXAS INSTRUMENTS)

EFCIS (Thomson -CSF)
-Catalogue sur les microprocesseurs et les mémoires (.1980)

MOTOROLA
-MC 6800 Microcomputer Sysetme Design Data
-MICROCOMPUTER