UNIVERSITE D'ALGER

Ecole Nationale Polytechnique

Département Eléctricité

THÈSE de FIN d'ÉTUDES
INGÉNIORAT en ÉLECTRONIQUE

Mise en ligne d'un lecteur optique de bande perforée sur MICROPROCESSEUR MC 6800

Sujet proposé par Mr H. TEDJINI, Docteur Ingénieur

Etudié par

Lakhdar AMARA **Abdelhamid AZOUANI**

« Janvier 1979 »

UNIVERSITE D'ALGER

Ecole Nationale Polytechnique

Département Eléctricité

THÈSE de FIN d'ÉTUDES
INGÉNIORAT en ÉLÉCTRONIQUE

Mise en ligne d'un lecteur optique de bande perforée sur MICROPROCESSEUR MC 6800

Sujet proposé par Mr H. TEDJINI, Docteur Ingénieur

Etudié par

Lakhdar AMARA

Abdelhamid AZOUANI

« Janvier 1979 »

R E M E R C I E M E N T S

Nous formulons l'expression de notre profonde reconnaissance à Monsieur Hacène TEDJINI, Docteur Ingénieur à l'Institut des Sciences et de la Technologie Nucléaire, qui a bien voulu nous recevoir dans son service et diriger notre travail.

Que toute l'Equipe du Projet Simulation et Contrôle trouve ici nos remerciements les plus sincères.

Nous ne manquerons pas d'exprimer aussi toute notre gratitude et notre reconnaissance à tous les professeurs de l'Ecole Nationale Polytechnique qui ont contribué à notre Formation.

Nous remercions également Mademoiselle Baya SEGHOUANI pour la Dactylographie du texte ainsi que tout le personnel technique ayant contribué à la publication de cette THESE.

A ma mère

A mon père

Lakhdar AMARA

A ma mère

A mon père

A mon frère

A mes soeurs

Abdelhamid AZOUANI

TABLE DES MATIERES

Chapitre I

E T U D E D U M P U

I. INTRODUCTION AU MICROPROCESSEUR M 6800

I.1- La famille M 6800

Elle est bâtie autour du MPU MC 6800 qui est un Microprocesseur à 8 bits parallèles doté d'une capacité d'adressage de 65536 mots mémoires.
Le microprocesseur MC 6800 constitue l'unité centrale pour la réalisation de micro-ordinateurs. Sur le support du MPU MC 6800 nous disposons de plusieurs autres éléments de la Famille M 6800 :

- des mémoires vives RAM (MCM 6810) à 128 X 8 bits
- des mémoires mortes ROM (MCM 6830) à 1024 X 8 bits
- des circuits d'interface à entrées séries ACIA (MC 6850)
- des circuits d'interface à entrées parallèles PIA (MC 6820)

Les circuits d'interface ont été conçus dans le but d'adapter la logique interne du Microprocesseur aux logiques des différents périphériques. Ces circuits sont entièrement programmables à travers le bus, et leurs status en temps réel est disponible sur le bus.

Le MPU adresse tous ces circuits comme des locations mémoire d'où la simplification de l'interface entre mémoire et les périphériques et l'élimination des instructions spéciales d'entrée/sortie.
Le MPU MC 6800 est compatible TTL ne nécessitant qu'une alimentation de 5 volts. La famille M 6800 est en technologie LSI MOS (canal N).

I.2- Structure Elementaire d'un Système M 6800

Le système minimal bati autour de l'unité centrale MPU MC 6800 comprend :

- une mémoire ROM contenant tous les sous-programmes fixes auquels le microprocesseur fait souvent appel (programme de service).
- une mémoire RAM utilisée pour l'exécution des programmes utilisateurs.
- un circuit adapteur d'interface de périphérique PIA (peripheral interface adapter) servant au couplage des périphériques avec le MPU.

Ces périphériques peuvent être la télétype, lecteur de bande, imprimante, le MPU communique avec les mémoires et les interfaces par l'intermediaire de 3 bus :

1- Bus des données (DATA BUS)

Composé de 8 lignes bidirectionnelles ; elles acheminent les données entre le microprocesseur et les périphériques.

2- BUS adresse

Achemine les adresses du MPU vers les mémoires et les circuits d'interface (16 bits). Ces lignes possèdent 3 états (0, 1, OFF). L'état OFF correspond à une haute impedance et est utilisé pour le mode DMA (Direct Memory Accès).

3- BUS contrôle bidirectionnel

Le bus contrôle assure le cheminement d'un mélange hetérogène de signaux pour l'organisation des opérations du système.

L'utilisation des circuits d'interface d'entrées-sorties permet au bus adresse d'avoir une responsabilité accrue dans le système M 6800. En effet non seulement le bus adresse sélectionne la mémoire mais il devient un outil pour sélectionner les circuits d'entrées-sorties. Par l'intermédiaire de ces 3 bus, l'interface d'entrées-sorties est considéré comme une location mémoire. Comme conséquence, l'utilisateur peut converser avec les entrées-sorties en utilisant n'importe laquelle des instructions faisant référence à la mémoire, en sélectionnant le périphérique désiré à l'aide d'une adresse mémoire.

L'architecture d'un micro-ordinateur organisé autour du M 6800 est donné par la Figure 1.

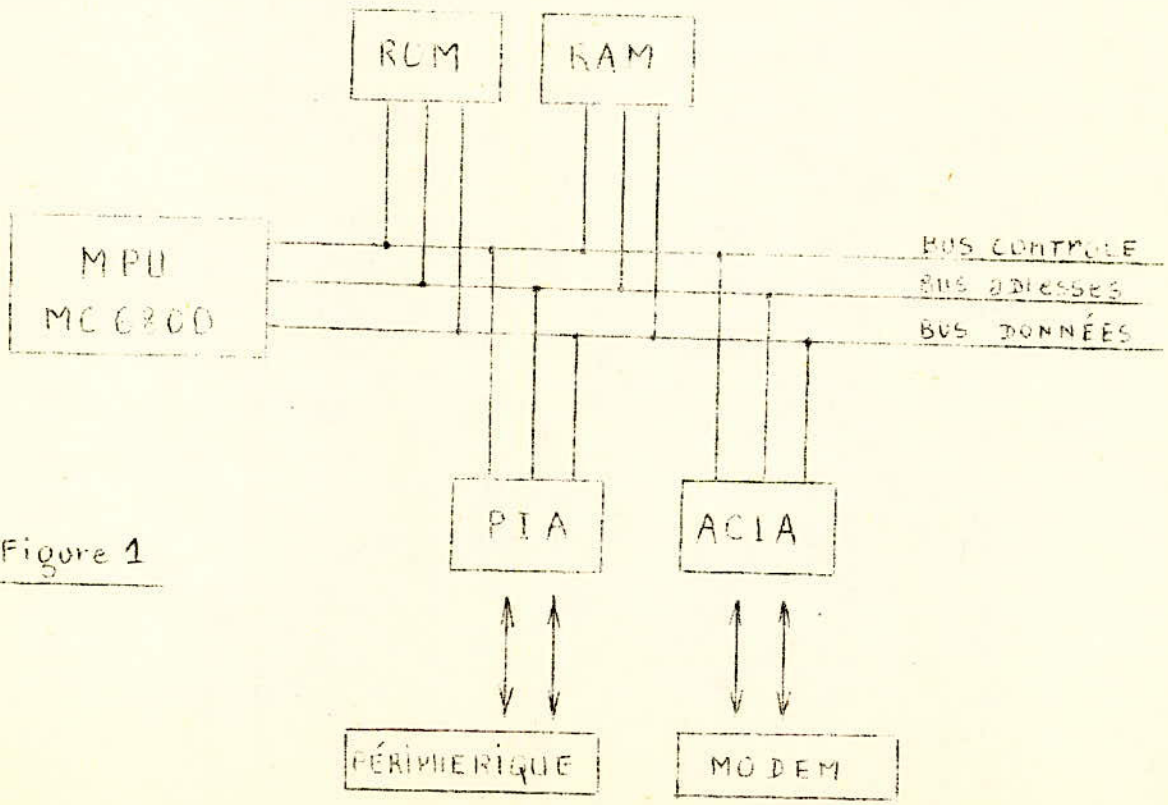


Figure 1

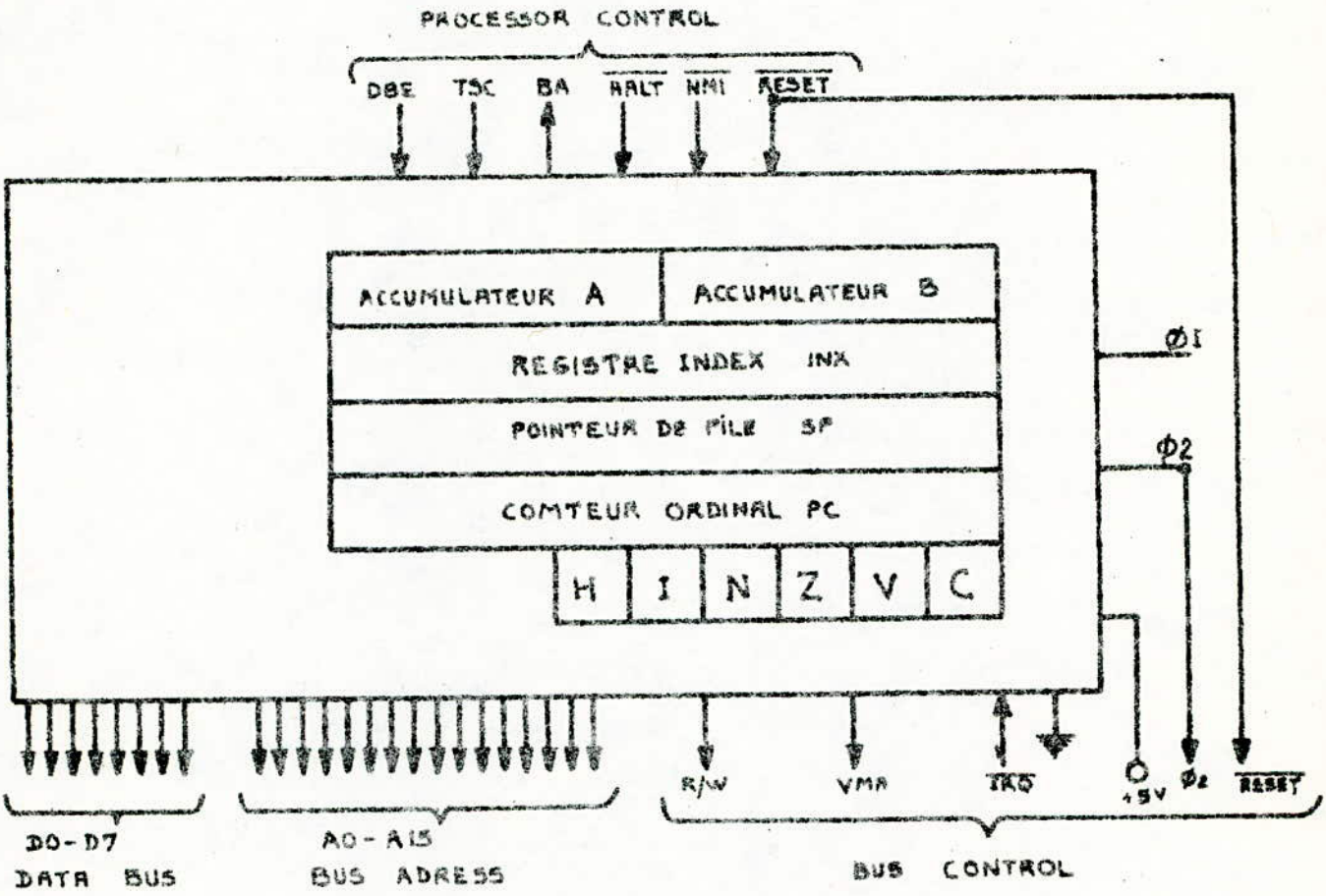


Fig.2 ORGANISATION INTERNE DU MC 6800

II. L'UNITE CENTRALE MPU MC 6800

II.1- Description Générale du MPU

Le MPU possède 6 registres internes : deux accumulateurs A et B, 1 registre index (INX), 1 compteur ordinal (PC), 1 pointeur de pile (SP) et 1 régistre code condition (CCR). Il est piloté par une horloge à deux phases $\emptyset_1$ et $\emptyset_2$ de 1 MHZ. Le MPU est un circuit bi-directionnel basé sur la structure bus, avec 8 bitsdonnées et 16 bits adresses.

Le MPU dispose de 72 intructions de longueurs variables et de 7 modes d'adressage : direct, relatif, immédiat, indexé, étendu, implicité, accumulateur, il est compatible TTL ne nécéssitant qu'une alimentation de 5V.
Le MPU est capable d'interfacer avec 8 circuits de la famille M 6 800 ou une charge TTL à une vitesse d'horloge de 1 µs.
Pour des systèmes necéssitant plus de 8 circuits interface de la famille M 6800 on utilise des "Data Bus extenders (BEX)".

Les lignes de contrôle comprennent $\overline{\text{Reset}}$ qui permet le redémarrage du microprocesseur, Interrupt Request ($\overline{\text{IRQ}}$) et Non-masquable interrupt ($\overline{\text{NMI}}$) pour surveiller le status des périphériques. Les signaux de contrôle TSC, $\overline{\text{Halt}}$ et E peuvent être utilisés pour l'accès direct en mémoire (DMA).

Le diagramme symbolique du MPU est donné par la Figure 2.

II.2- Etude du MPU MC 6800

II.2.1- Registres internes

Le MPU contient 3 registres à 16 bits et 3 registres à 8 bits qui peuvent être utilisés par le programmeur.

a) - deux accumulateurs A et B de 8 bits chacun : ont pour rôle principal de garder les opérandes et les résultats de l'unité arithmétique et logique (ALU). Le fait de disposer de 2 accumulateurs permet de stocker un résultat partiel dans l'accumulateur B tout en réalisant un calcul dans l'accumulateur A.

b) - le registre index (INX) : registre à 16 bits utilisé pour l'adressage en mode indexé, il permet une méthode d'adressage mémoire supplémentaire par le calcul de l'adresse.

c) - le compteur ordinal ou "Program Counter" PC : registre à 16 bits, la séquence d'un programme est contrôlée par le contenu du PC. Sous exécution normale du programme, le PC retient la prochaine adresse de la prochaine instruction que l'on doit aller chercher dans la mémoire contenant le programme et il est incrémenté automatiquement par le contrôle interne du MPU chaque fois que chaque octet d'une instruction a été pris en compte.

d)-le pointeur de pile ou stack pointer (SP) : c'est également un registre à 16 bits contenant l'adresse de la location disponible dans une pile externe. Le SP offre ainsi au MPU des possibilités de stockage en "pile" ou "stack". Il s'agit d'une portion de mémoire traitée en mémoire LIFO (Lastin, First out = dernier entré, premier sorti). Celle-ci est adressée au moyen d'un registre dit "pointeur de pil " et elle est utilisée en premier lieu pour permettre de stocker temporairement le contenu ou les "états" des registres du MPU dans des positions connues lorsque le MPU est forcé de faire une action demandée par une interruption (ou un branchement ou un saut à une soubroutine). Ceci permet au MPU de retrouver et terminer l'exécution du programme qui avait été interrompu.

e) - une unité arithmétique et logique (ALU)

organe d'exécution des fonctions arithmétiques et logiques

f) - le registre des codes de conditions (CCR)

Toutes les opérations effectuées dans l'ALU agissent sur les états du CCR décrivant les résultats de ces opérations. Ceci apparait particulièrement utile lorsque l'on désire changer le déroulement d'un programme de façon conditionnelle, d'après le résultat de l'opération précédente.

CCR	1	1	H	Im	N	Z	V	C

Les bits du CCR sont affectés par des instructions, les résultats des instructions ou des interruptions. La signification des bits est la suivante :

- H = Half Carry (demi-retenue) : mis à 1 s'il y a retenue qui passe du bit 3 au bit 4 par suite d'une instruction ADD, ABA, ABC.

- Im = Interrupt Mask (masque d'interruption)
mis à 1 par une interruption hardware ou software.

- N = négative : la mise à 1 de ce bit indique que le résultat est négatif (b7 du résultat = 1)

- Z = Zéro : la mise à 1 de ce bit indique que le résultat est nul.

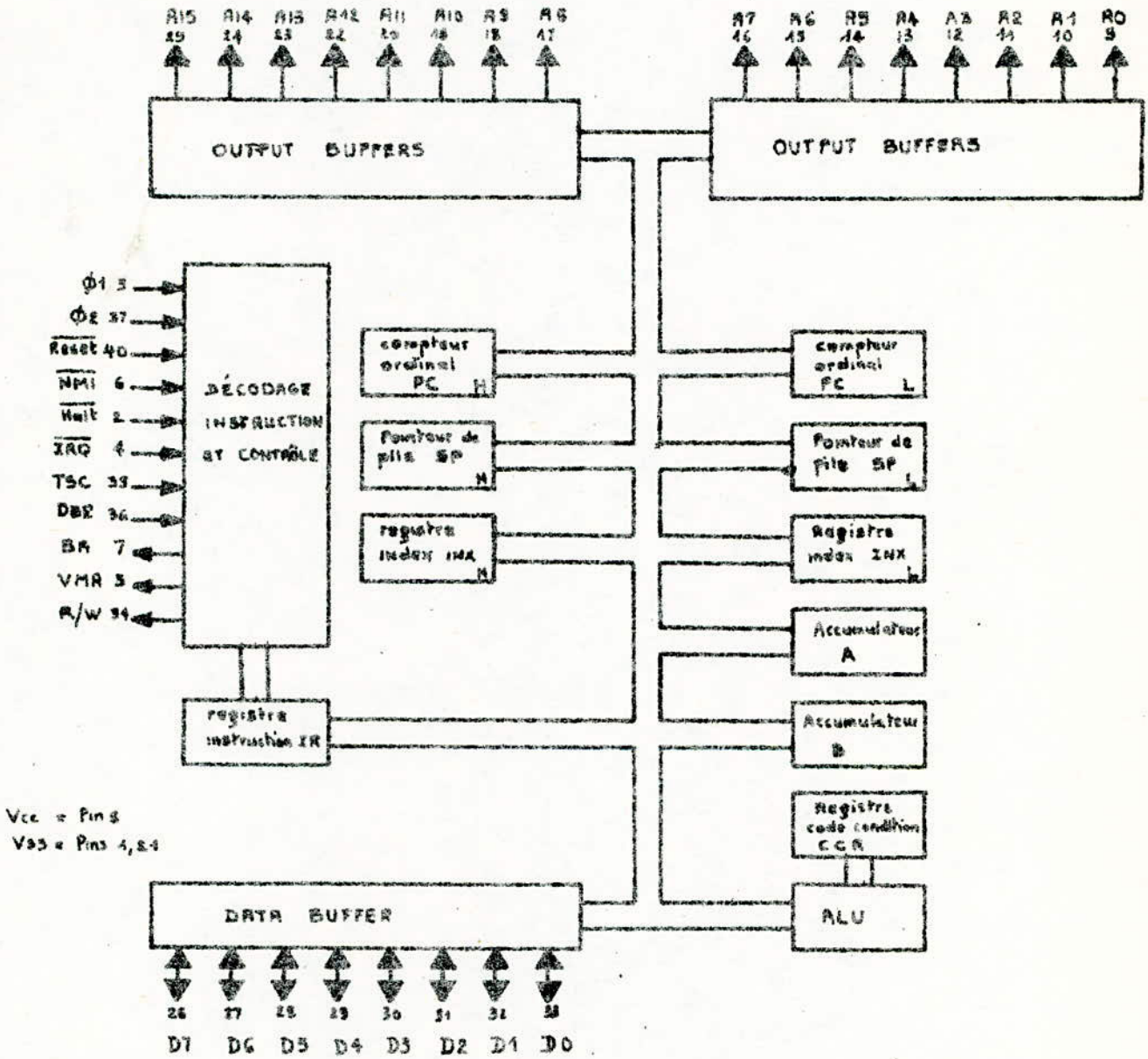


Fig 3. ORGANISATION DU MICROPROCESSEUR M6800

- V = Overflow (dépassement) : la mise à 1 de ce bit indique que le résultat a débordé la capacité de 8 bits complementé à deux des registres A ou B.

- C : Carry (retenue) : ce bit est mis à 1 s'il y a une retenue à partir du bit 7 du résultat à la suite d'une opération entre accumulateurs.

- les bits inutilisés 6 et 7 sont toujours à 1.

La figure 3 détaille la structure interne du M 6800 articulée autour de registres, BUS et ALU. Notons que le registre d'instruction (IR) retient l'octet contenant le code opération de l'instruction qui doit être exécutée une fois que l'on a été chercher celle-ci en mémoire. Celui-ci est décodée et des signaux internes de contrôle sont générés pour donner la séquence à l'ALU et produire toutes les fonctions nécessaires à la manipulation des données afin de réaliser l'instruction.

II.2.2- Les Différentes lignes du MPU

Les lignes du MPU se divisent en trois (3) catégories :

- lignes de données (data bus)
- lignes d'adresses (bus adress)
- lignes de contrôle (bus contrôl)

1°/ Data Bus (D0-D7

D0-D7 sont des lignes données bidirectionnelles qui ont des buffers de sortie avec possibilité "Three state" (trois états). Ces circuits "buffer" de données agissent comme "buffer" d'entrée lorsque le MPU lit une information présente sur le

bus et comme "drivers" de sortie lorsque l'unité centrale émet sur le bus. Ils peuvent également être placés dans un état de haute impédance permettant aux autres modules du système d'utiliser le bus sans impliquer l'unité centrale.

2°/ Bus Adress (A0-A15)

A0-A15 sont des lignes adresse avec possibilité "three state". Ce bus adress de 16 bits permet au MPU d'adresser 64 K octets mémoires ($K = 2^{10}$) quand ces lignes adress sont placées en état de haute impédance (état OFF) c'est essentiellement un circuit ouvert ce qui permet de faire des accès mémoire direct (mode DMA : direct memory accèss).

3°/ Bus Contrôl

Pour le fonctionnement correct du MPU des signaux de contrôle et de synchronisation sont nécessaires pour accomplir des fonctions spécifiques et pour déterminer l'état du processeur. Ces signaux sont les suivants :

a- signaux de synchronisation

$\emptyset_1$ et $\emptyset_2$ sont les phases d'une horloge. Sur ces signaux en opposition de phase sont synchronisés l'adressage et le transfert des données.

$\emptyset_1$ = horloge prévue pour activer le MPU
$\emptyset 2$ servant à mettre en liaison l'élément selectionné du système avec le MPU Via le Data bus et ceci seulement lorsque le bus adress et le VMA sont stables. La fréquence d'utilisation de $\emptyset_2$ est comprise entre 100 KHZ et 1 MHZ.

b- signaux de Contrôle

ils permettent de synchroniser et de coordonner le fonctionnement du microprocesseur avec les organes extérieurs (mémoires ou périphériques) en général plus lents.

b-1- TSC Three state contrôl :
contrôle troisième état. Il met les lignes adresses et la ligne R/W à l'état OFF (haute impédance). TSC n'a aucun effet sur le Data bus. Ce signal permet l'arrêt du MPU pendant un temps très court (max 4,5 µs) sinon les données à l'intérieur du MPU seront perdues.

b-2- Bus Available BA (bus disponible) :
quand il est à 1 il indique que le microprocesseur est stoppé et le bus adresse est disponible.

b-3- $\overline{\text{Halt}}$: ce signal permet l'arrêt du microprocesseur pendant un temps indéfini. Quand ce signal est à l'état bas le microprocesseur est stoppé à la fin de l'instruction en cours, c'est-à-dire tous les signaux caractérisant les 3 états des lignes sont à l'état de haute impédance ; le VMA est à 0 et la sortie Bus available (BA) passe à l'état 1.

b-4- DBE Data Bus enable : validation de bus de données.
C'est un signal caractérisant les 3 états du Data bus qui peuvent être :

- DBE = 1 le "bus data" est dans l'état 0 (état fermé sens rentrant)
- DBE = 1 le "bus data" est dans l'état 1 (état fermé sens sortant)
- DBE = 0 le "bus data" est dans l'état OFF (état ouvert haute impédance).

Le signal dérive généralement de $\emptyset_2$.

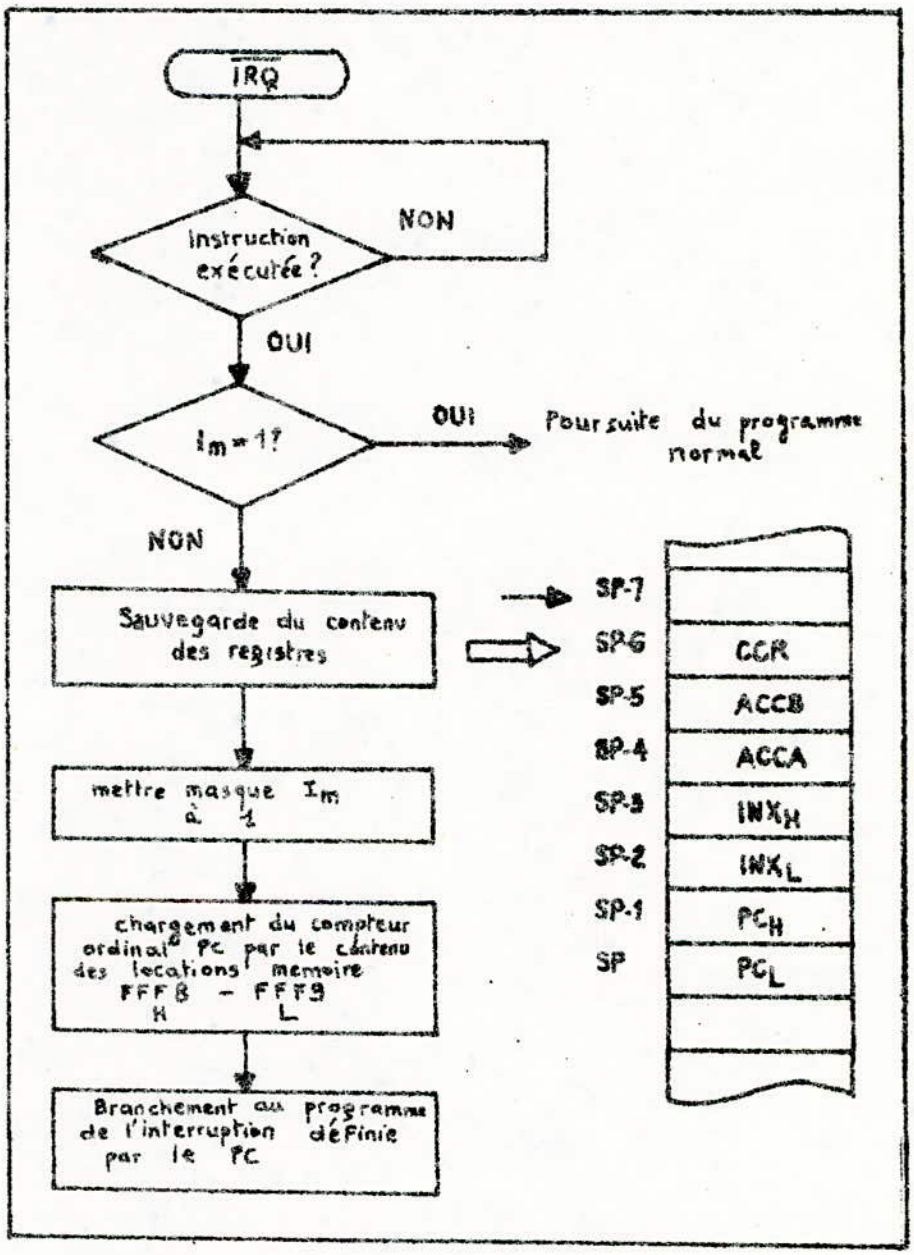


Fig. 4 Traitement de l'interruption Fondamentale IRQ masquable.

b-5- Read/Write (R/W)

c'est un signal indiquant pour chaque cycle opération si le MPU est dans un mode lecture ou écriture. Il indique donc la direction de transfert des données.

R/W = 1 : le MPU effectue une opération de lecture
R/W = 0 : le MPU effectue une opération d'écriture

b-6- VMA (Valid memory adress) : Adresse mémoire disponible. Ce signal indique aux périphériques qu'il y a une adresse valide sur le bus adresse. Notons que le transfert des données ne s'effectue entre le data bus et l'élément selectionné que si le VMA est à un niveau haut. VMA peut être aussi utilisé pour valider le PIA, ACIA etc...

c- signaux d'interruptions

Dans les applications typique les périphériques peuvent en continu générer des signaux d'interruptions qui sont pris en compte par le MPU, l'usage des interruptions fournit une souplesse d'emploi considérable. La prise en compte des interruptions doit répondre aux critères suivants :

- sauvegarder les éléments du programme qui est en cours d'exécution
- gérer plusieurs interruptions venant des périphériques différents
- gérer la priorité des interruptions

Le MPU reconnait quatre (4) types d'interruption :
$\overline{\text{Reset}}$, $\overline{\text{IRQ}}$, $\overline{\text{NMI}}$, SWI

c-1- Interrupt Request $\overline{IRQ}$ (Voir figure 4)

c'est une demande d'interruption hardware qui peut être masquée.
Imaginons le MPU effectuant un programme. Si un périphérique veut échanger des informations avec le MPU il faudra que ce dernier abandonne le programme en cours en sauvegardant les registres (données) dans la pile : il y a donc interruption du programme initial. Avant de reconnaitre l'interruption le MPU termine l'instruction en cours. Alors il teste le bit Im du CCR. Si ce bit n'est pas à 1 le MPV va commencer à exécuter une séquence d'interruption en sauvegardant les registres dans la pile. Le MPU répond à la demande d'interruption en mettant Im à 1 de façon à ce qu'une autre interruption n'arrive pas simultanément. A la fin du cycle une adresse de 16 bits sera chargée qui correspond au "vecteur adresse" qui se trouve dans les locations FFF8-FFF-9. Une adresse chargée dans ces locations a pour résultat que le MPU se branche à un sous-programme d'interruption.

c-2- $\overline{Reset}$ (voir figure 6)

utilisé pour redemarrer ou remettre à zéro (0) le MPU après une coupure du secteur ou une initialisation. Quand $\overline{Reset}$ est à son niveau bas donc actif, le bit Im (mask interrupt) est mis à 1 : le compteur ordinal PC pointe à l'adresse FFFE-FFFF de sorte que le MPU puisse exécuter la séquence d'interruption.

c-3- $\overline{Non\ Maskable\ Interrupt\ (NMI)}$ (voir figure 7)

cet interruption est reconnue par le MPU tout de suite. Elle est souvent utilisée comme une détection de panne secteur. Fonctionnement similaire à celui de l'entrée $\overline{IRQ}$ seulement

Fig. 5 Traitement de l'interruption logicielle SWI (software interrupt)

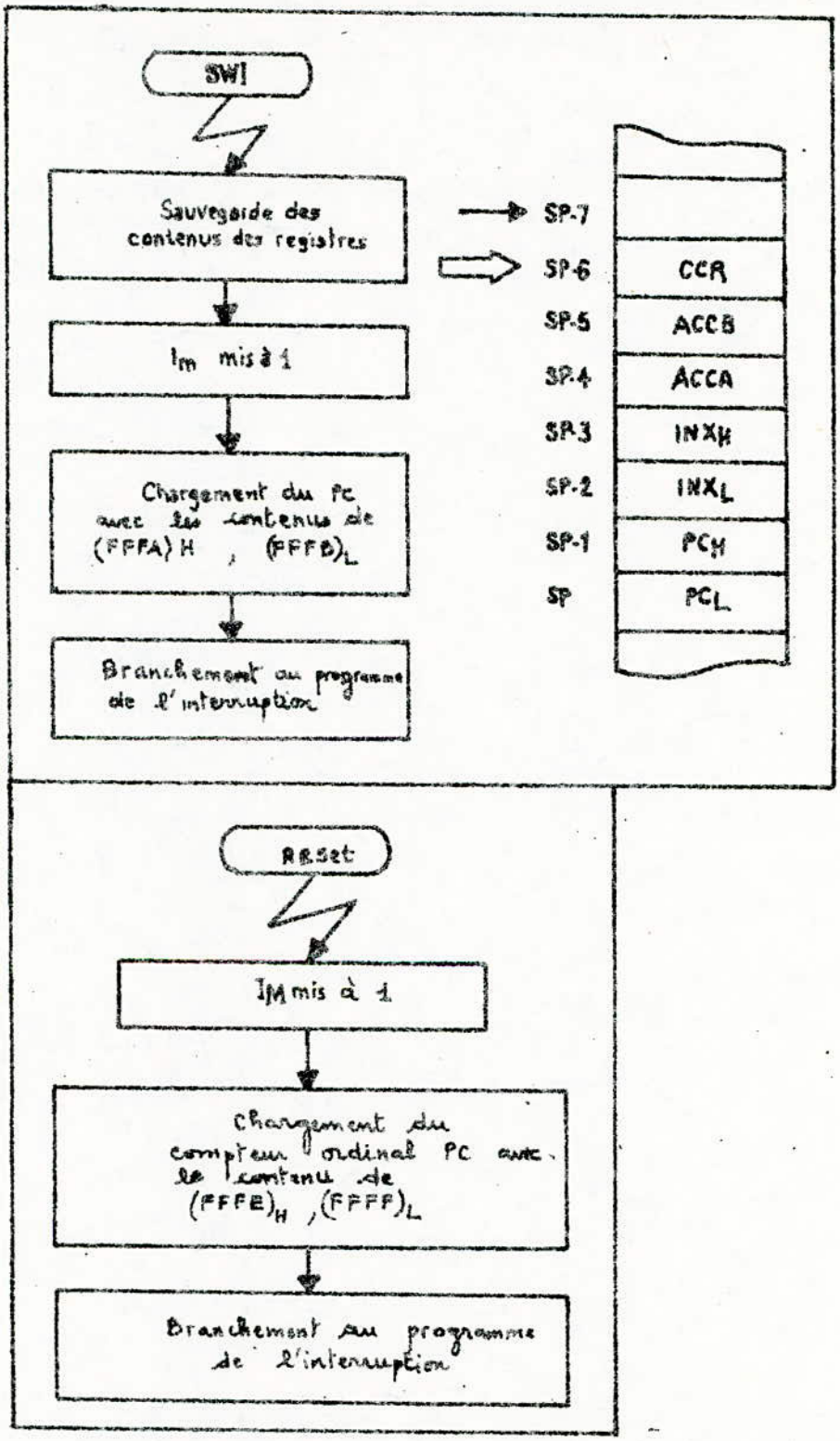


Fig. 6 Traitement de l'interruption du redémarrage Reset

le bit Im n'a pas d'effet sur $\overline{NMI}$, l'adresse de la routine $\overline{NMI}$ doit être chargée dans les locations FFFC et FFFD.

c-4- Software interrupt (SWI) (voir figure 5)

elle est similaire aux interrupt· s hardware et elle est provoquée par instruction. Elle est utilisée pour insérer des points de coupure dans le programme. En effet SWI arrête le programme et sauvegarde les registres.

d- Priorité des interruptions

Le problème des priorités se pose pour le MPU lorsque plusieurs périphériques demandent des interruptions ; la première tâche d'une routine d'interruption est d'identifier la source de l'interruption et décider quelle interruption prendra en compte en première. La méthode la plus courante consiste à tester dès le début chaque périphérique. Si les interruptions sont génerées par des PIA ou ACIA, il suffit de tester le bit d'indicateur d'interruption du registre de contrôle ce qui permet d'identifier la source de l'interruption.
Pour établir une priorité on peut tenir compte du premier indicateur rencontré, éxécuter cette séquence et ensuite procéder au test de l'indicateur suivant.
Il existe une autre méthode dite de vectorisation. La vectorisation évite le balayage des périphériques et par conséquent autorise une plus rapide prise en compte des demandes d'interruptions. Le principe qui prévaut pour gérer la vectorisation des interruptions consiste à réserver une petite partie de la mémoire RAM, interne au système, aux vecteurs. Par exemple, on réservera les premières cellules

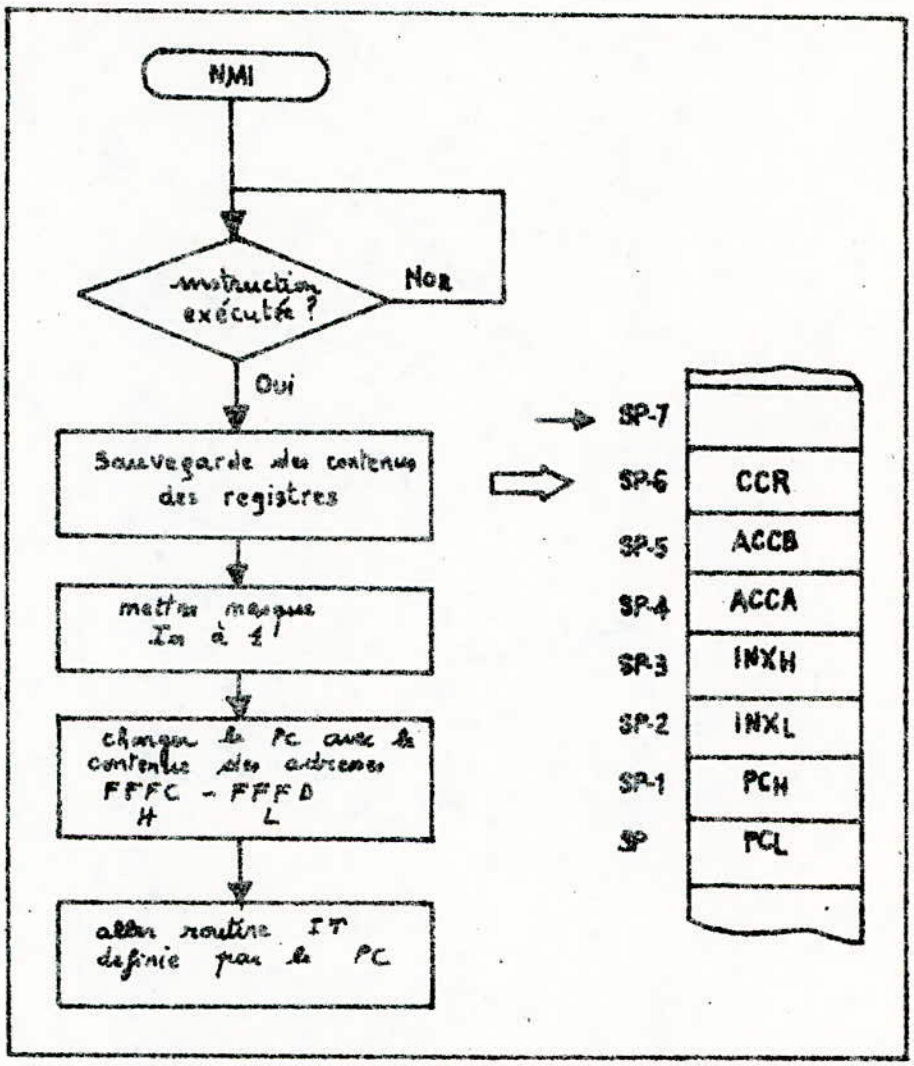


Fig. 7 Traitement de l'interruption non masquable NMI

de la mémoire ou les toutes dernières aux vecteurs, et ce de façon systématique. Avec le 6800 ce sont les dernières cellules qui sont réservées à cet effet, deux par adresse puisque chaque cellule contient un octet alors qu'une adresse complète doit être codée sur 16 bits pour pouvoir atteindre 64K positions. Ces dernières cellules, si l'on retient la numérotation hexadécimale, seront donc par ordre décroissant : FFFF, FFFE, FFFD, FFFC, FFFB etc...

Ainsi le codeur de priorité selectionne l'interruption à reconnaitre et conduit le MPU à l'adresse mémoire dans laquelle se trouve le vecteur.

Exemple du vecteur d'interruption (8 niveaux d'interruption)

	$\overline{\text{RESET}}$	FFFF - FFFE
	$\overline{\text{NMI}}$	FFFD - FFFC
	SNI	FFFB - FFFA
	8	FFF9 - FFF8
	7	FFF7 - FFF6
	6	FFF5 - FFF4
$\overline{\text{IRQ}}$	5	FFF3 - FFF2
	4	FFF1 - FFF0
	3	FFEF - FFEE
	2	FFED - FFEC
	1	FFEB - FFEA

Cet exemple nous montre la localisation des vecteurs d'interruptions dans la mémoire RAM du MPU MC 6 800. Chaque adresse est donnée sur deux mots mémoires, ici, par conséquent l'octet de plus fort poids et l'octet de plus faible poids.

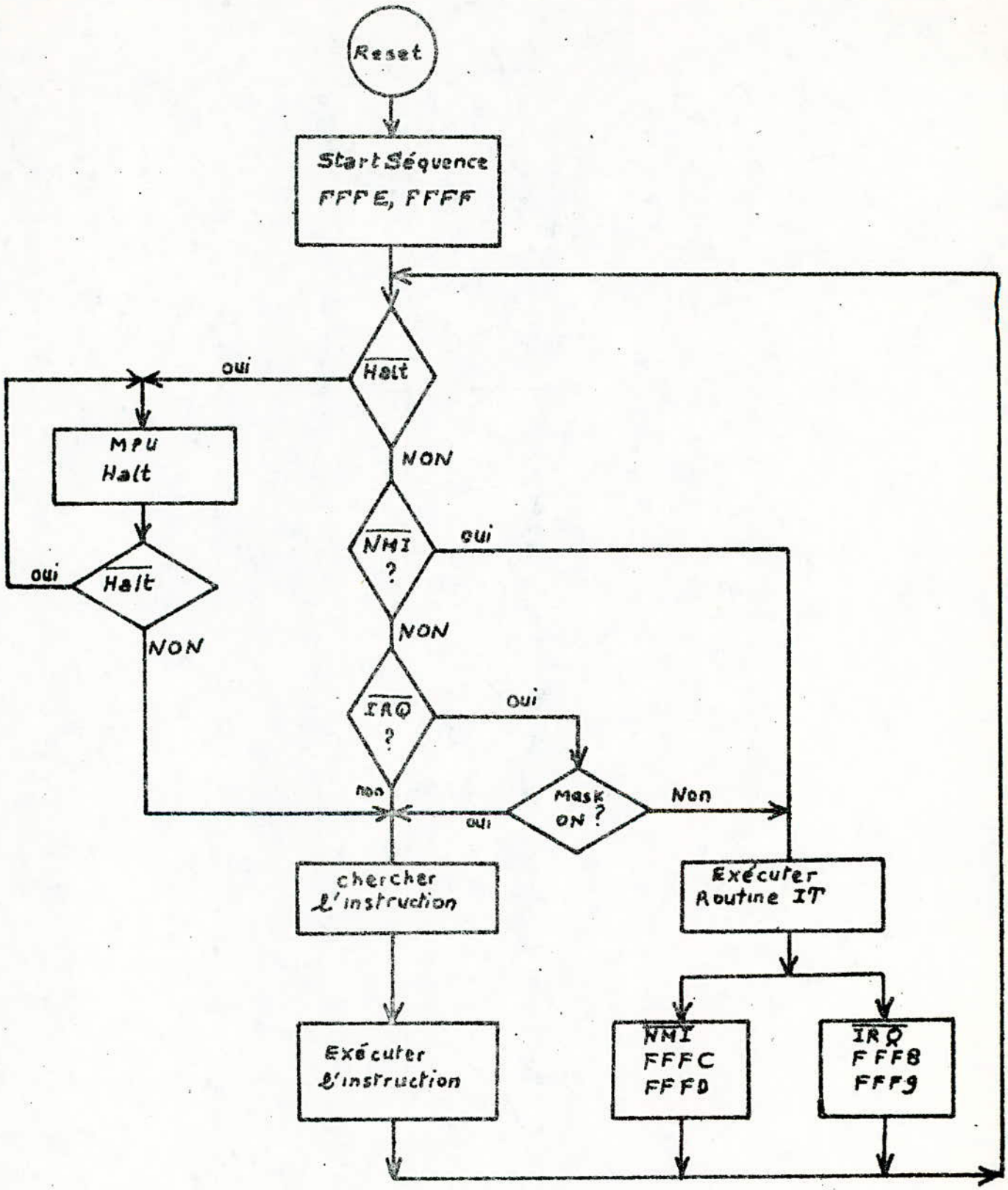


Fig. 8 Organigramme d'une séquence "Start"

e- Séquence démarrage

Avant d'entreprendre une séquence de lecture/écriture le MC 6800 exécute une "Start sequence" dont le diagramme est donné par la figure 8.

II.2.3- Les instructions du M 6800

Le microprocesseur M 6800 possède un jeu de 72 instructions, il y a des instructions logiques, arithmétiques, de notation de déplacement, chargement, rangement, branchements conditionnels et inconditionnels et des instructions d'interruption et de pointage (pointeur de pile).

Les instructions sont classées en quatre (4) grandes catégories :

a- instructions agissant sur les accumulateurs et la mémoire
b- instructions agissant sur le registre index et le pointeur de pile (stack pointer)
c- instructions de branchement et de saut
d- instructions agissant sur le registre code condition (CCR)

Le minimum des cycles mémoire pour une instruction est 2 (LDA) et le maximum 12 (SWI) (voir annexe liste des instructions).

II.2.4- Les modes d'adressage du M 6800

Le microprocesseur exécute le programme en allant chercher une par une, dans l'ordre, les instructions dans la mémoire puis les exécute.

Le premier byte correspond à l'instruction proprement dite, ou opérateur, ce byte est suffisant pour identifier le type d'adressage. Le second byte et éventuellement le troisième contiennent soit un opérande soit une adresse ou un déplacement. La lecture du premier byte engendre une séquence de micro-instruction ; L'utilisateur du MC 6800 dispose de 7 modes d'adressage

- adressage accumulateur
- adressage immédiat
- adressage étendu
- adressage implicité
- adressage indexé
- adressage direct

a- Adressage accumulateur (1 byte)

Ex : ASLA
Il s'agit de déplacer les bits de l'accumulateur A d'une position vers la gauche ainsi le bit 7 se retrouve dans le carry (c)

b- Adressage direct (2 bytes)

L'adresse de l'opérande se trouve dans le second octet. Ce mode ne peut atteindre les adresses supérieures à 255.

Ex. STAA 30 97 30
Stocker le contenu de l'accumulateur A à la position mémoire 0030.

c- Adressage étendu (3 bytes)

Ce mode diffère du mode direct par le nombre de bytes formant l'opérande mais aussi par sa portée car il permet d'adresser les locations mémoire comprises entre 256 et 65535 (0100 et FFFF). Dans ce mode, l'adresse contenue dans le second octet est le MSB et celle du 3è octet est le LSB de l'adresse complète qui est sur deux (2) octets.

Ex. STAA 800B B7 800B
Stocker le contenu de l'accumulateur A à l'adresse 800B

d- Adressage immédiat

l'emplacement de l'opérande est dans le deuxième octet de l'instruction. Cependant deux (2) cas sont à envisager suivant que l'on s'adresse à des registres 8 bits (ACCA et ACCB) où à des registres 16 bits (registre index...). Par exemple pour l'adressage des accumulateurs A et B l'instruction comprend deux bytes :

L DAA 30 86 30

charger immédiatement 30 dans l'accumulateur A.
On utilise 3 bytes pour l'adressage des registres à 16 bits

Ex. LDX 8008 CE 8008

Le registre d'index est chargé immédiatement par 8008

e- Adressage Indexé (2 bytes)

L'adresse contenue dans le second octet est ajouté au contenu du registre index LSB. Le carry est alors ajouté au MSB de l'index. Bien qu'il simplifie la programmation ce mode est exécuté par le MPU avec un temps plus long que celui mis pour l'adressage étendu

LDX	8008	CE	8008
CLR	X,1	6F	01

effacer le contenu de l'adresse 8008 + 1 = 8009

f- Adressage implicité (1 byte)

L'instruction contient l'adresse

g- Adressage relatif (2 bytes)

Ce mode est utilisé dans les branchements. L'adresse contenue dans le deuxième octet est additionné au LSB du compteur ordinal

plus 2 (PC+2). Le carry est ensuite ajouté au MSB. On peut adresser des données dans une gamme de -125 à +125 mots.

Ex: TST	LDAA	00111100	langage machine	2005	B6	3C
	BPL	TST		2008	2A	FB
				↑ PC		

Le deuxième byte de l'instruction "Branch" est un nombre donnant le déplacement que doit effectuer le compteur ordinal PC pour que le MPU puisse continuer son programme ; pour cela il est ajouté au contenu du PC plus 2.

Ex.	2008	0010	0000	0000	1000
+	FB			1111	1011
+	2				10
		0010	0000	0000	0101 = 2005

d'où l'on déduit l'adresse de branchement 2005

II.2.5- Accès direct à la mémoire (DMA)

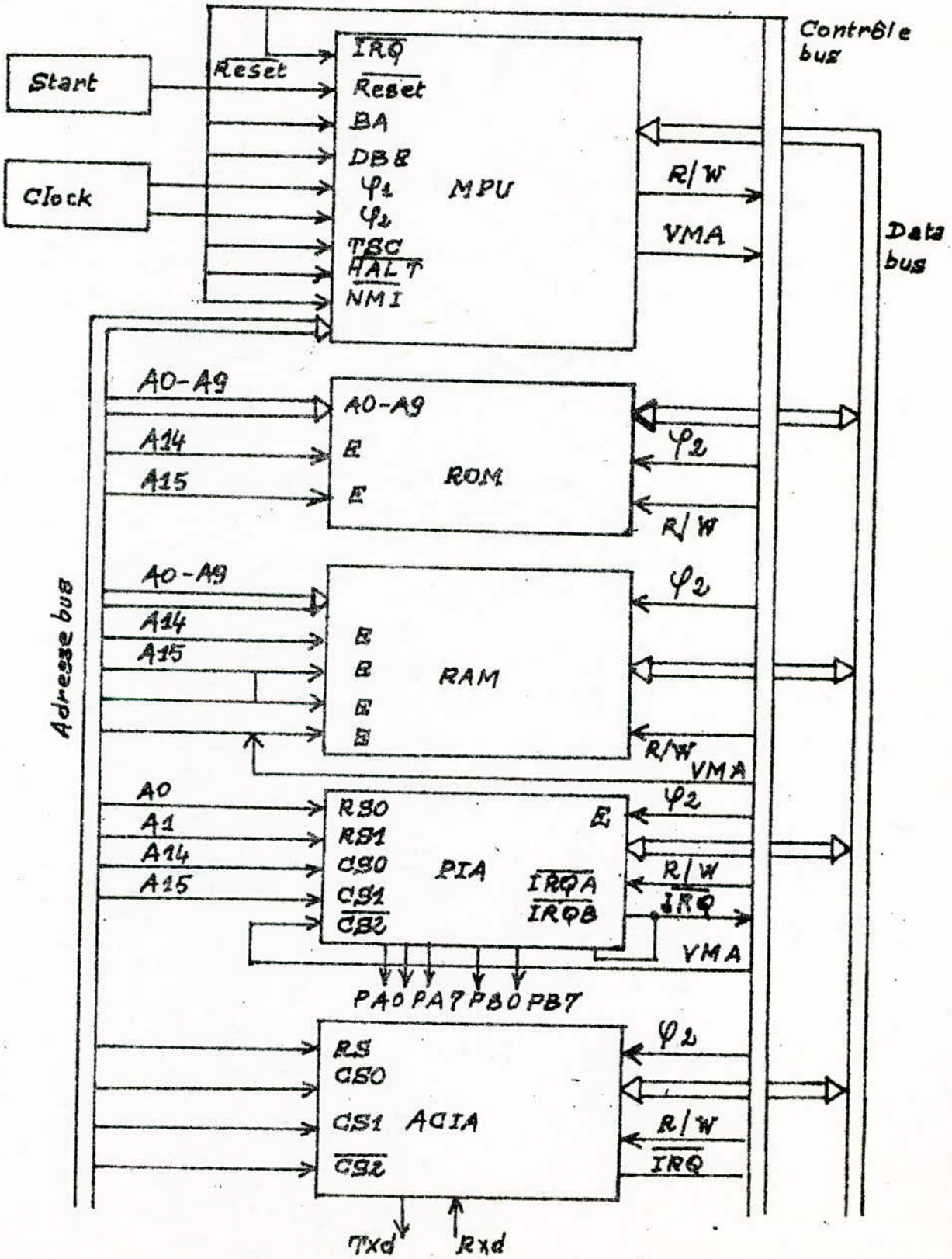
La technique DMA consiste à autoriser les périphériques rapides (ou bien un autre MPU) à accéder à la mémoire du système sans utiliser le temps de travail du MPU. Trois (3) méthodes de DMA couramment utilisées sont par ordre croissant de complexité.

a) Arrêt du processeur (Halt) : le MPU est arrêté
b) Vol de cycle : MPU actif 1 cycle sur 5
c) DMA multiplexés : cycle MPU ralenti.

Les deux méthodes DMA par vol de cycle et DMA multiplexés permettent d'aller chercher ou de stocker une donnée dans la mémoire

alors que le MPU est en fonctionnement, tandis que la méthode DMA par Halte du processeur nécessite que le MPU soit stoppé avant que l'accès mémoire soit effectué. Cette méthode d'arrêt est la plus simple à utiliser mais, ce cas impliquant un temps relativement long avant d'arrêter le MPU, elle peut ne pas être acceptable.

fig. 9 Système minimum M6800



Chapitre II

LES MEMOIRES

II.1. LA MEMOIRE RAM (RAMDOM Accèss MEMORY)

La mémoire RAM est une mémoire vive utilisée pour stocker des informations variables dans un système microprocesseur. Ainsi l'unité centrale, sous contrôle du programme peut à volonté lire ou changer le contenu de la mémoire vive. Les mémoires vives peuvent être de deux sortes : statiques ou dynamiques.
Dans cette dernière, l'information est stockée sous forme de charge électrique dans la capacité de porte d'un transistor MOS. Comme ces condensateurs ne sont pas parfaits, les charges s'écoulent par des fuites et l'information est perdue si la charge n'est pas regénerée périodiquement. Les RAM statiques n'ont pas besoin d'être regénerées car les cellules mémoire sont à 2 états et de conception similaire au Flip-Flop traditionnel. D'une façon générale la mémoire statique consomme plus de puissance que son homologue dynamique.
Si la capacité nécessaire de la RAM est faible, par exemple moins de 1 K bytes il est plus économique d'utiliser des RAM statiques organisées en octets : on emploiera par exemple une mémoire de 128 mots de 8 bits.
De 1 à 4 K bytes il est plus économique d'utiliser des RAM dynamiques. La figure 10 représente la RAM MCM 6810 A semblables à d'autres RAM MOTOROLA disponibles pour le système M 6800. Elle est statique, technologie MOS CANAL N d'une capacité de 128 X 8 bits ; elle est compatible TTL et a un bus donnée bi-directionnel et three-state. Dans la même famille on peut citer la RAM MCM 6604 dynamique.

fig.10 . MCM 6810 RAM - Diagramme.

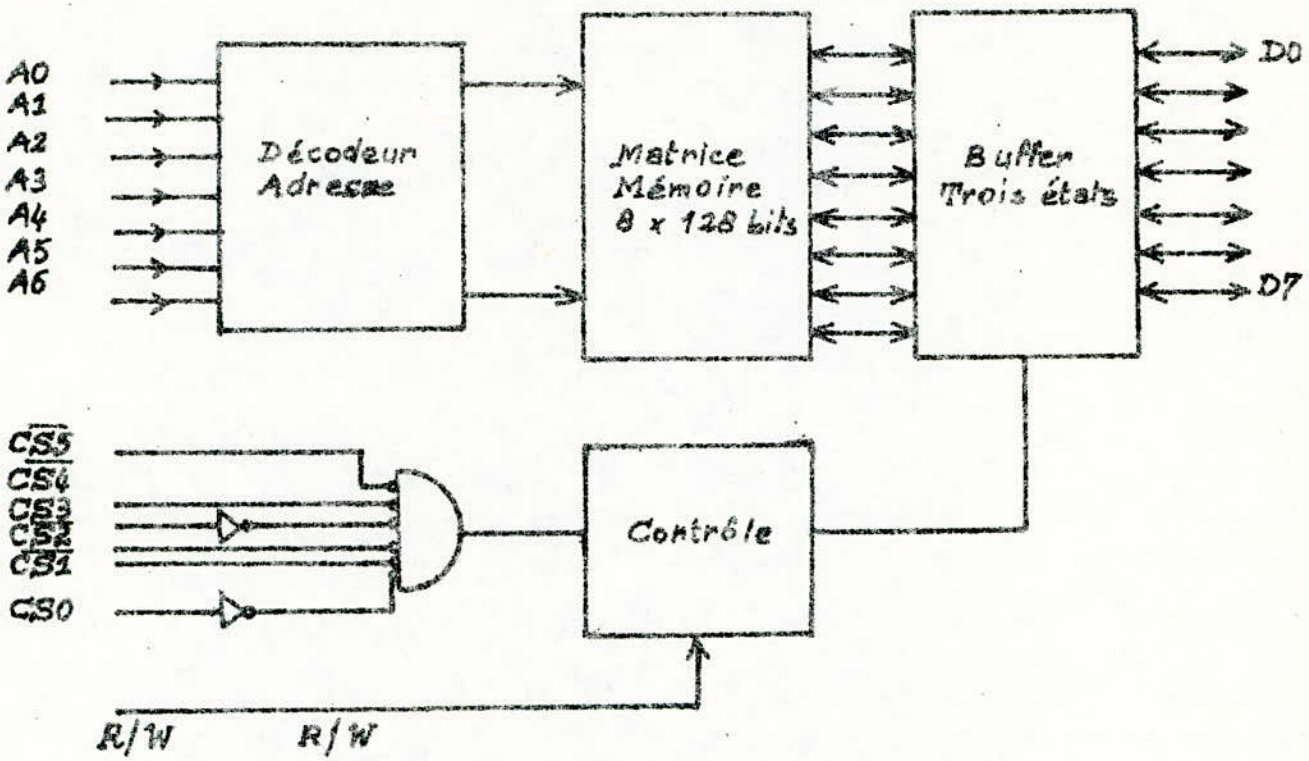
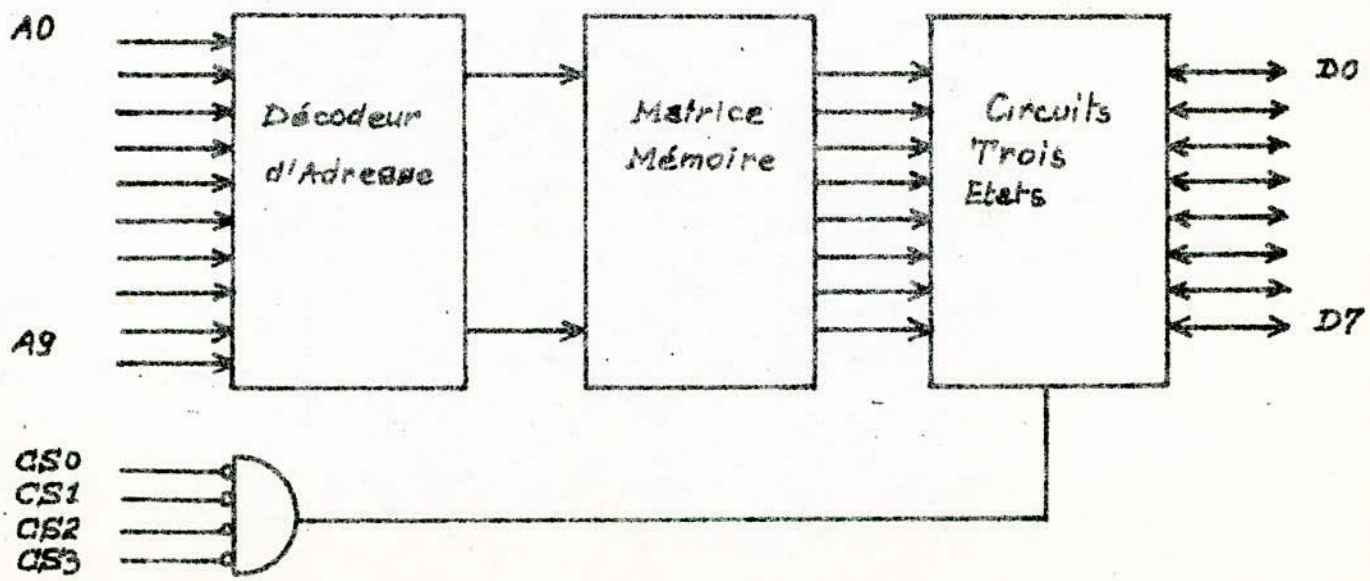


fig.11 . ROM MCM 6830 .



II.2. LA MEMOIRE ROM (Read Only Memory)

Ces mémoires dites mortes sont par construction non volatiles. Une ROM est un circuit intégré contenant une matrice de cellules adressables : chaque cellule ayant été placée de façon définitive dans un état binaire 1 ou 0. Les ROM peuvent être réparties en trois (3) groupes : le premier est composé de ROM programmées par masque par le fabriquant ; le second groupe est celui des mémoires mortes PROM (programmables Read Only Memory) qui sont programmées par l'utilisateur ; le dernier groupe de ROM se compose des EAROM (programmables electriquement) ou RMM (Read Mostly Memory)

La figure 11 représente la mémoire ROM MCM 6830 qui comme la RAM est organisée en octet, de sorte qu'une simple adresse accède directement à un octet d'information.

Chaque ROM a une capacité de 1024 octets et par conséquent requiert dix entrées d'adressage ; quatre (4) entrées "chip select" sont disponibles et peuvent être choisies individuellement comme répondant soit à un niveau haut ; soit à un niveau bas, selon les besoins du système, cela au moyen d'une option sur le masque que le client définit en même temps que la matrice de bits.

Chapitre III

PERIPHERAL INTERFACE ADAPTER (PIA : MC 6820)

I. CARACTERISTIQUES GENERALES

Le dialogue entre le Microprocesseur MC 6800 et les périphériques nécessite la mise en oeuvre d'organes spécialisés qu'on désigne sous le nom d'adaptateurs d'interfaces périphériques ou PIA (MC 6820).

Le PIA donne ainsi la possibilité d'interfacer le MPU avec les périphériques à travers deux bus de 8 bits bi-directionnels et programmables et quatre (4) lignes de contrôle. Les principales caractéristiques du PIA sont :

- Bus 8 bits bi-directionnels pour la communication avec le MPU
- 2 bus bidirectionnels à 8 bits programmables pour l'interface périphérique
- 2 registres de contrôle programmables (CRA et CRB)
- 2 registres direction de données programmables (DDRA et DDRB)
- 4 lignes individuelles d'interruption
- Logique de contrôle handshaking pour l'opération d'entrée-sortie
- lignes périphériques technologie Three-state
- Interruptions programmables
- Capacité C- MOS drive sur les lignes PA0 - PA7

La figure 12 donne le diagramme synoptique du PIA avec les différents registres. Pour la sélection du circuit nous disposons de 3 lignes Chip-select, 2 lignes pour l'adressage des 6 registres,

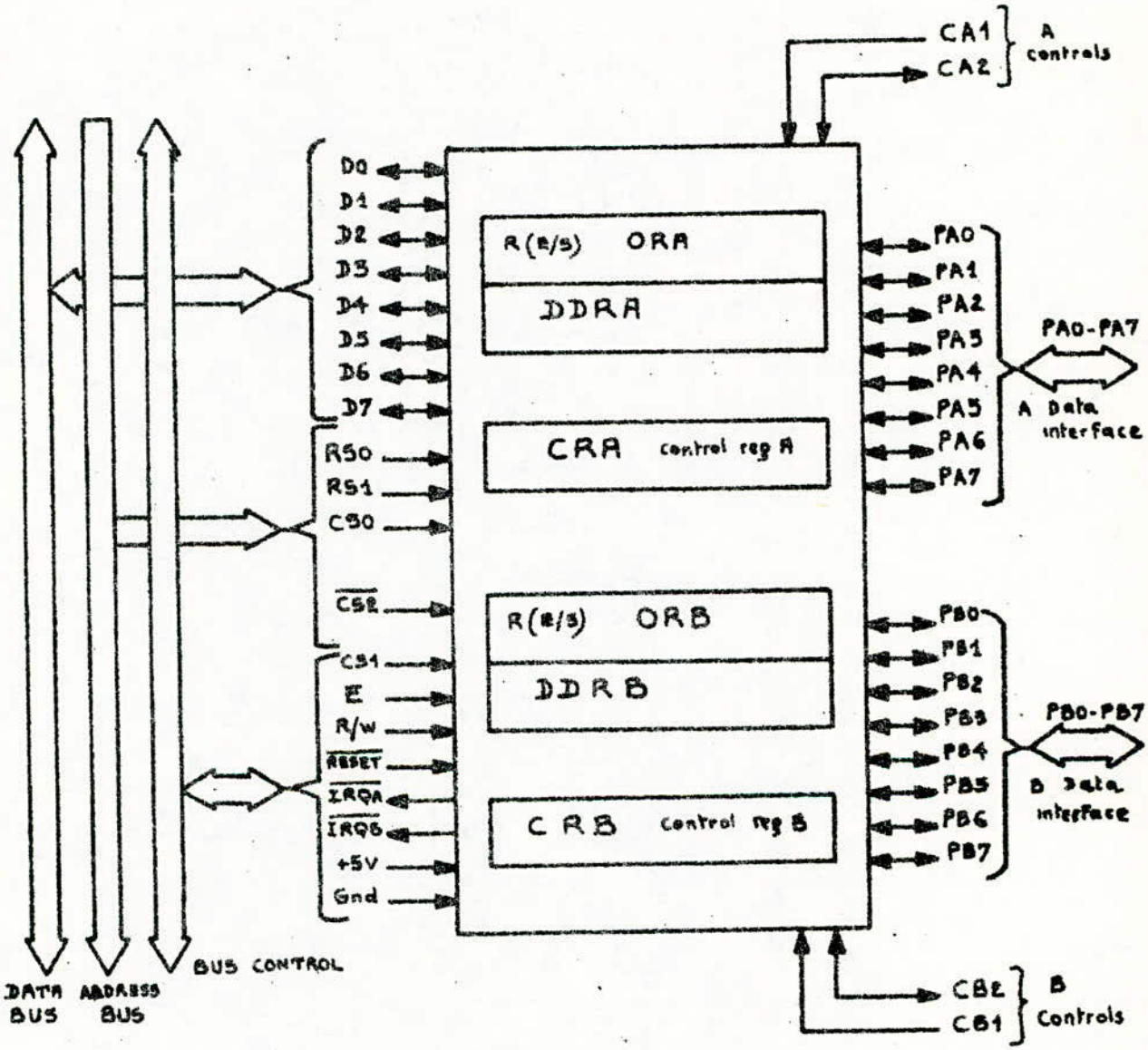


Fig.12 MC 6820 PIA (PERIPHERAL INTERFACE ADAPTER)

une ligne validation (Enable) et une ligne R/W.
Ces signaux en liaison avec le MPU permettent d'avoir un contrôle complet sur le MC 6820 PIA.

II. LIAISON ENTRE LE MPU et le PIA

1- Data bus (D0-D7)

Le Data bus comprend 8 lignes D0-D7 données bidirectionnelles. Elles permettent le transfert de données entre le PIA et le MPU.

2- Adress bus

Cinq lignes utilisées pour l'adressage du MC 6820 PIA

2.1- Chip select ($CS0, CS1, \overline{CS2}$)

Ce sont des lignes d'adressage du PIA. Elles sont utilisées pour la sélection, ces lignes doivent être stables pendant la durée de l'impulsion E. Elles sont reliées à trois lignes du bus Adress (A0-A15).

2.2- Register select (RS0 et RS1)

Ces deux lignes permettent la selection des 6 registres internes du PIA

3- Bus Contrôle

Il se compose de cinq lignes suivantes :

3.1- la ligne Enable E

C'est le signal timing du PIA qui valide le transfert. Cette entrée dérive de $\emptyset_2$: elle contrôle les interruptions provenant

de CA1, CA2, CB1, CB2.

3.2- la ligne R/W (Read/Write)

C'est un signal écriture-lecture généré par le MPU pour contrôler la direction du transfert des données.

3.3- la ligne $\overline{\text{Reset}}$

Ce signal est utilisé pour effacer tous les registres du PIA (remise à 0)

3.4- Interrupt Request : $\overline{\text{IRQA}}$, $\overline{\text{IRQB}}$

Ces deux lignes agissent comme des demandes d'interruption ; chaque ligne $\overline{\text{IRQ}}$ a deux bits "Flag" du registre de contrôle qui peuvent causer la désactivation de la demande d'interruption, ce qui permet d'inhiber une interruption.

III. LIAISON ENTRE LE PIA et le PERIPHERIQUE

La liaison entre le PIA et le périphérique s'effectue au moyen de deux bus de 8 bits chacun bidirectionnels et 4 lignes de contrôle d'interruption.

1- lignes PA0-PA7

Ces lignes peuvent individuellement être programmées en entrées ou sortie. On peut lire les données disponibles sur les lignes, après un transfert des données du MPU vers le périphérique, quand ces lignes sont programmées en sortie, ou le MPU peut lire les données fournies par le périphérique aux lignes programmées en entrées.

2- lignes PB0-PB7

Ces lignes sont identiques aux lignes précédentes PA0-PA7, à la différence que les buffers de sortie sont de technologie "Three state". En outre ces lignes peuvent conduire un courant de 1mA sur 1,5V (pour être branchées directement sur la base d'un transistor).

3- les lignes de contrôle d'interruption

Le MC 6820 PIA possède 4 lignes de contrôle d'interruption (CA1, CA2, CB1 et CB2). CA1 et CB1 sont des entrées d'interruption ; Elles positionnent les interrupt Flags (indicateurs d'interruption) CRA7 et CRB7 des registres de contrôle CRA et CRB. Les lignes CA2 et CB2 peuvent être programmées comme entrées ou sorties ; Elles positionnent les indicateurs d'interruption CRA6, CRB6 des registres de contrôle CRA et CRB lorsque elles sont programmées comme entrées d'interruption.

3.1- la ligne CA1

L'entrée CA1 indique au MPU qu'une information est présente à l'entrée du PIA. Le bit CRA7 du registre de contrôle CRA est associé à CA1. Cet indicateur CRA7 d'interruption est mis à 1 par une transition active sur la ligne CA1 et remis à 0 par une lecture du registre de donnée correspondant et par $\overline{\text{Reset}}$.

3.2- la ligne CB1

Dans le cas où les lignes PB0-PB7 sont programmées en sorties, l'entrée CB1 indique au MPU qu'un périphérique demande une information. Le bit CRB7 du registre de contrôle CRB est associé à

CA1. Cet indicateur CRB7 d'interruption est mis à 1 par une transition active sur la ligne CB1 et remis à 0 par une lecture du registre de donnée correspondant et par Reset.

3.3- la ligne CA2

La présence d'un niveau bas sur cette ligne programmée en sortie indique au périphérique d'entrée que la donnée a été transmise du PIA vers le MPU. Elle fournit donc un signal de reconnaissance.

3.4- la ligne CB2

La présence d'un niveau bas sur cette ligne programmée en sortie indique au périphérique de sortie que le PIA est prêt à lui envoyer une donnée. Elle constitue de cette façon un signal d'acquittement.

IV. STRUCTURE INTERNE DU PIA

Le schéma détaillé du MC 6820 PIA est donné dans la figure 13. Intérieurement le PIA est divisé en deux configurations symétriques et indépendantes A et B. Chaque configuration contient 3 registres:

- un registre direction de données DDRA (B)
- un registre de sortie ORA (B)
- un registre de contrôle CRA (B)

Le MPU traite ces registres comme des locations mémoire, et ils peuvent être lus ou écrits.

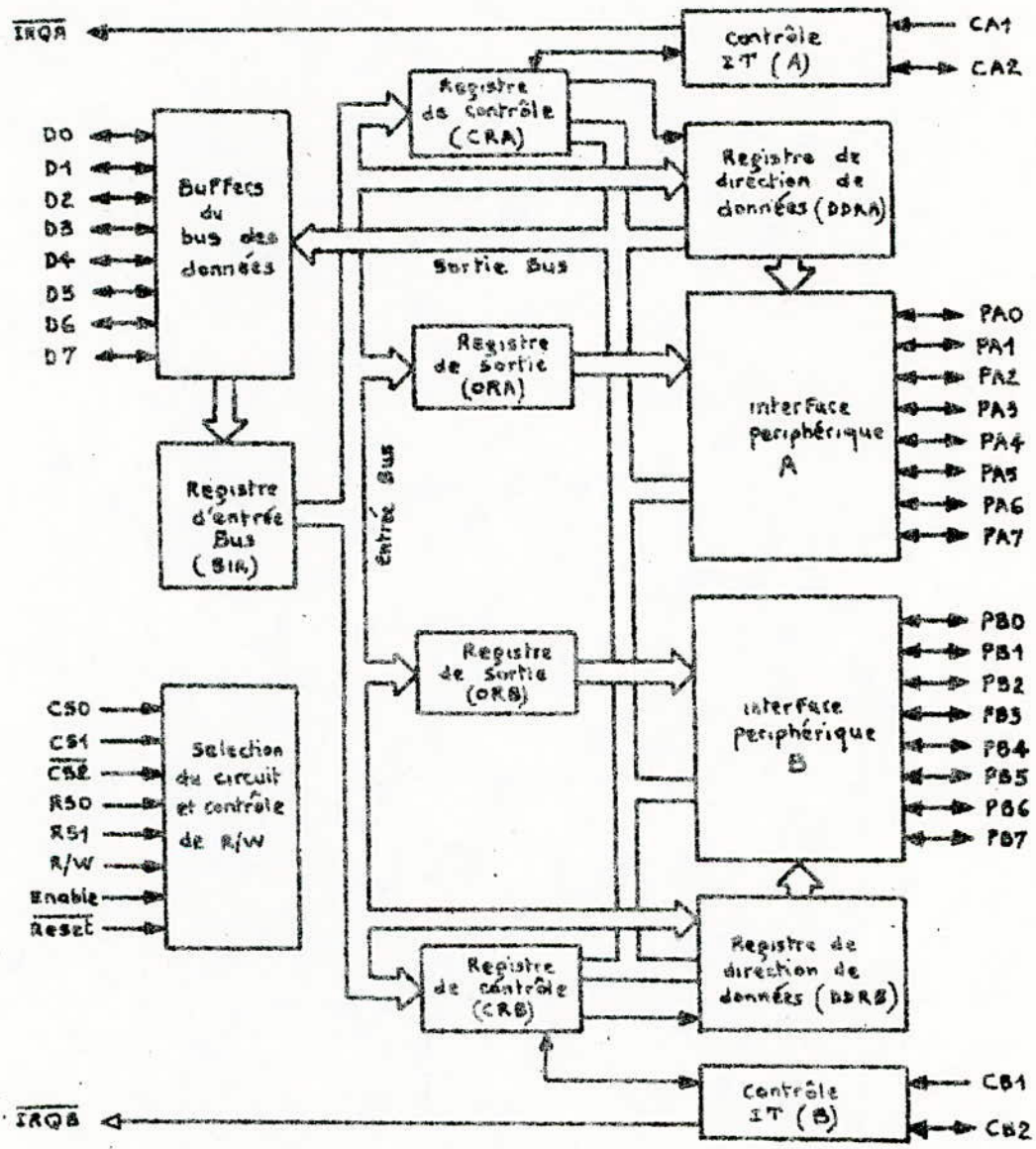


Fig.13 BLOC DIAGRAMME DU PIA

1- Registres direction de données (data direction registers : DDRA et DDRB)

Ces deux registres permettent de définir le sens de transfert des données. A chaque bit de ces registres est associée une ligne donnée. Un "1" définit la ligne correspondante comme une sortie, Un "o" comme une entrée. Si on charge par exemple le DDRA par 00 alors les lignes PA0-PA7 sont des entrées ; Par contre si on le charge par FF toutes les lignes PA0-PA7 sont des sorties.

2- Registres de sortie (Output Registers : ORA et ORB)

Ces registres sont utilisés pour le transfert des données. Quand ils sont adressés ces registres rangent les données présentes au MPU data bus, pendant une opération d'écriture du MPU. Ces données vont aussi apparaître sur celles des lignes PA0-PA7 (PB0-PB7 programmées en sortie).

3- Registres de contrôle (Contrôl Registers : CRA et CRB)

Les registres de contrôle CRA et CRB permettent au MPU de commander par programme les quatre lignes de contrôle d'interruption CA1, CA2, CB1 et CB2. Ils permettent aussi d'autoriser les interruptions sur $\overline{IRQA}$ et sur $\overline{IRQB}$ et de tester sur les bits 6 et 7 l'état des indicateurs d'interruption (registre d'état). Les bits 0 à 5 de ces registres peuvent être écrits ou lus par le MPU, par contre les bits 6 et 7 ne peuvent être que lus, ils sont modifiés par des interruptions externes sur les lignes de contrôle d'interruption CA1, CA2, CB1 et CB2. La configuration de ces registres de contrôle est donnée par la figure 14.

Fig.14 REGISTRES DE CONTRÔLE CRA et CRB

	7	6	5	4	3	2	1	0
CRA	IRQA1	IRQA2	CONTRÔLE CA2			DDRA ou ORA accès	CONTRÔLE CA1	

	7	6	5	4	3	2	1	0
CRB	IRQB1	IRQB2	CONTRÔLE CB2			DDRB ou ORB accès	CONTRÔLE CB1	

RS1	RS2	CONTRÔLE REGISTRE		LOCATION SELECTIONNÉE
		bit CRA2	bit CRB2	
0	0	1	ϕ	registre periphérique ORA
0	0	0	ϕ	registre de direction DDRA
0	1	ϕ	ϕ	registre de contrôle CRA
1	0	ϕ	1	registre périphérique ORB
1	0	ϕ	0	registre de direction DDRB
1	1	ϕ	ϕ	registre de contrôle CRB

Fig.15 ADRESSAGE INTERNE

4- Accès aux Registres

Le registre DDRA et le registre ORA ont la même adresse externe (vue du MC 6800) de même pour les registres DDRB et ORB. Les bits CRA2 et CRB2 permettent d'accéder soit au registre de sortie ORA (B) [bit 2 du registre de contrôle à 1] , soit au registre direction de données DDRA (B) [bit 2 du registre de contrôle à 0] quand la combinaison correcte est appliquée à RS0 et RS1. L'adressage des registres du PIA est donnée par le tableau de la figure 15.

5- Contrôle des lignes CA1 et CB1

Les deux bits 0 et 1 des registres de contrôle CRA et CRB [CRA0, CRA1, CRB0, CRB1] sont utilisées pour commander les entrées d'interruptions CA1 et CB1. Les bits CRA0 et CRB0 autorisent (ou masquent) les demandes d'interruption au MPU respectivement par les broches $\overline{\text{IRQA}}$ et $\overline{\text{IRQB}}$. Les bits CRA1 et CRB1 donnent la possibilité de choisir la transition active du signal d'entrée d'interruption CA1 ou CB1 qui doit générer une interrupt request $\overline{\text{IRQA}}$ ou $\overline{\text{IRQB}}$.

5.1- Indicateurs d'interruption (ou interrupt. Flags) CRA7 et CRB7

Les bits CRA7 et CRB7 sont associées respectivement aux lignes CA1 et CB1 ;Ils renseignent sur l'existence ou l'absence d'interruption de la part de CA1 et CB1. Chaque indicateur CRA7 (CRB7) d'interruption est mis à 1 par une transition active sur la ligne CA1 (CB1).

Ces indicateurs d'interruption ne peuvent pas être directement mis à 1 par le MPU et sont remis à 0, indirectement par une lecture du registre de donnée ORA (B) et par $\overline{\text{Reset}}$.

5-2- Autorisation du signal de demande d'interruption

- CRA0 (CRB0) = 0 masque les demandes d'interruption sur CA1 (CB1)
- CRA0 (CRB0) = 1 autorise les demandes d'interruption sur CA1 (CB1)

5.3- Choix de la transition active de CA1 (CB1)

- CRA1 (CRB1) = 0 l'interrupt : Flaq CRA7 (CRB7) mis à 1 par une transition négative de CA1 (CB1)
- CRA1 (CRB1) = 1 l'interrupt Flag CRA7 (CRB7) mis à 1 par une transition positive de l'entrée d'interruption CA1 (CB1)

Le tableau de la figure 16 illustre le contrôle des lignes CA1 et CB1 par les bits CRA0 (CRB0) ; CRA1 (CRB1).

6. Contrôle des lignes CA2 et CB2

Les bits 5,4,3 du registre de contrôle CRA sont utilisés pour commander la ligne d'interruption CA2. Par contre les bits 5, 4, 3 du registre CRB sont utilisés pour contrôler la ligne d'interruption CB2.

Fig. 16 CONTRÔLE DES ENTRÉES d'INTERRUPTION CA1 et CB1

CRA1 (CRB1)	CRA0 (CRB0)	entrée d'interruption CA1 (CB1)	indicateur d'interruption CRA7 (CRB7)	MPU Interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	↓ active	mis à 1 sur ↓ de CA1 (CB1)	invalide (niveau haut) $\overline{IRQA}$ ($\overline{IRQB}$) = 1
0	1	↓ active	mis à 1 sur ↓ de CA1 (CB1)	niveau bas (active) $\overline{IRQA}$ ($\overline{IRQB}$) = 0 lorsque CRA7 (CRB7) = 1
1	0	↑ active	mis à 1 sur ↑ de CA1 (CB1)	invalide (niveau haut) $\overline{IRQA}$ ($\overline{IRQB}$) = 1
1	1	↑ active	mis à 1 sur ↑ de CA1 (CB1)	niveau bas (active) $\overline{IRQA}$ ($\overline{IRQB}$) = 0 lorsque CRA7 (CRB7) = 1

CRA5 (CRB5)	CRA4 (CRB4)	CRA3 (CRB3)	entrée d'interruption CA2 (CB2)	interrupt Flag CRA6 (CRB6)	MPU interrupt Request $\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓ ACTIVE	mis à 1 sur ↓ de CA2 (CB2)	invalide (niveau haut) $\overline{IRQA}$ ($\overline{IRQB}$) = 1
0	0	1	↓ ACTIVE	mis à 1 sur ↓ de CA2 (CB2)	niveau bas (active) $\overline{IRQA}$ ($\overline{IRQB}$) = 0 lorsque CRA6 (CRB6) = 1
0	1	0	↑ ACTIVE	mis à 1 sur ↑ de CA2 (CB2)	invalide (niveau haut) $\overline{IRQA}$ ($\overline{IRQB}$) = 1
0	1	1	↑ ACTIVE	mis à 1 sur ↑ de CA2 (CB2)	niveau bas (active) $\overline{IRQA}$ ($\overline{IRQB}$) = 0 lorsque CRA6 (CRB6) = 1

Fig. 17 CONTRÔLE DE CA2 et CB2 COMME DES ENTRÉES D'INTERRUPTION
CRA5 (CRB5) = 0

1°) ↑ transition positive
2°) ↓ transition négative

6.1- Bit 5 = 0 [CRA5 (CRB5) = 0]

Si les bits CRA5 et CRB5 sont à zéro, les lignes CA2 et CB2 seront des entrées d'interruption similaires à CA1 et CB1. Ainsi les bits 3 et 4 accomplissent les fonctions semblables aux bits 0 et 1 pour le contrôle de CA2 et CB2.

6.1.1- **indicateurs d'interruption CRA6 et CRB6**

Les bits CRA6 et CRB6 sont associées respectivement aux lignes CA2 et CB2 programmées comme entrées d'interruption. Ces indicateurs d'interruption sont mis à 1 par une transition active de CA2 (CB2). Ils sont remis à zéro (0) par une lecture du registre de donnée ORA (ORB) et par $\overline{\text{Reset}}$.

6.1.2- Autorisation du signal de demande d'interruption

- CRA3 (CRB3) = 0 masque la demande d'interruption sur CA2 (CB2)
- CRA3 (CRB3) = 1 autorise les demandes d'interruption sur CA2 (CB2)

6.1.3- Choix de la transition active de CA2 (CB2)

- CRA4 (CRB4) = 0 l'interrupt Flag CRA6 (CRB6) est mis à 1 par une transition négative de CA2 (CB2)
- CRA4 (CRB4) = 1 l'interrupt Flag CRA6 (CRB6) est mis à 1 par une transition positive de CA2 (CB2).

La figure 17 illustre le comportement de CA2 (CB2) comme entrée d'interruption.

6.2- Bit 5 = 1 [CRA5 (CRB5) = 1]

Si les bits CRA5 et CRB5 sont à 1, CA2 et CB2 seront des sorties de commande de la périphérie par le microprocesseur et fonctionnent suivant l'un des trois modes :

- mode SET-RESET
- mode HAND-SHAKING (Synchronisation des échanges)
- mode Pulse-STROBE

6.2.1- Mode Set-Reset (Bit 5=1, Bit 4=1)

La sortie de commande CA2 (CB2) suit le bit 3 du registre de contrôle CRA (B) de la même façon qu'il a été programmée.

- la sortie CA2 (CB2) passe à l'état bas quand le microprocesseur écrit un 0 dans CRA3 (CRB3)

- la sortie CA2 (CB2) passe à l'état haut quand le microprocesseur écrit un 1 dans le bit CRA3 (CRB3)

6.2.2- Mode Handshaking (Bit 5 = 1, Bit 4 = 0, Bit 3 = 0)

- sortie CA2 : CRA5=1, CRA4=0, CRA3=0

  Envoi d'une impulsion vers le périphérique indiquant l'exécution d'une opération de lecture - CA2 est donc utilisée comme strobe de lecture avec la restitution de CA1. En effet la sortie CA2 passe à l'état bas sur la première transition négative de E ($\emptyset_2$) qui suit une lecture d'une donnée provenant de la périphérie A (ORA).

  CA2 retourne à son état initial (niveau haut) quand l'interrupt · Flag

CRA7 est mis à 1 par une transition positive de CA1 (si le bit CRA1 =1) ou par une transition négative de CA1 (si le bit CRA1 = 0)

- sortie CB2 : CRB 5=1, CRB 4=0, CRB 3 = 0

Envoi d'une impulsion vers le périphérique indiquant l'écriture d'une donnée dans le registre ORB. CB2 est donc utilisée comme strobe d'écriture avec restitution de CB1. En effet la sortie CB2 passe à l'état bas sur la transition positive de la première impulsion E ($\emptyset_2$) qui suit une écriture du registre de donnée ORB. La sortie CB2 passe à l'état haut quand l'indicateur d'interruption CRB7 est mis à 1 par une transition positive de CB1 (si le bit CRB1 = 1) ou par une transition négative de CB1 (si le bit CRB1=0).

6.2.3- Mode Pulse-Strobe (Bit 5=1, Bit 4=0, Bit 3=1)

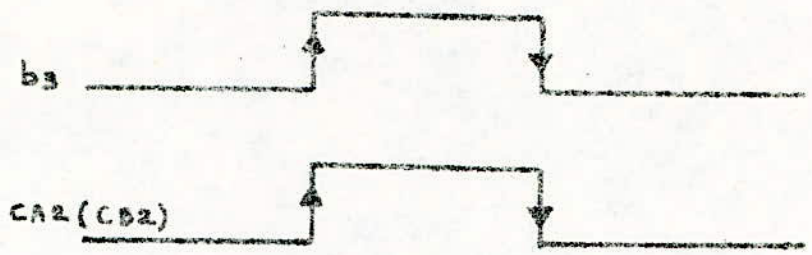
- sortie CA2 : CRA5=1, CRA4=0, CRA3=1

CA2 est utilisée comme strobe de lecture avec restitution de E. La sortie CA2 passe à l'état bas sur la transition négative de la première impulsion E ($\emptyset_2$) qui suit une lecture d'une donnée provenant de la périphérie A (ORA). CA2 retourne à son état initial (niveau haut) à la prochaine transition négative de E ($\emptyset_2$).

- sortie CB2 : CRB5=1, CRAB4=0, CRB3=1

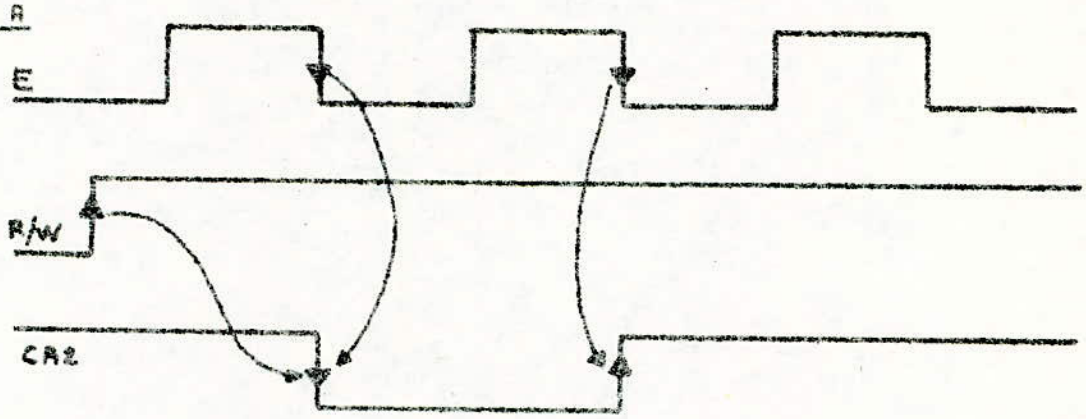
CB2 est utilisée comme strobe d'écriture avec restitution de E. La sortie CB2 passe à l'état bas sur la transition positive de la première impulsion E ($\emptyset_2$) qui suit une écriture du registre de donnée ORB.CB2 passe à l'état haut à la prochaine transition positive de E ($\emptyset_2$).

1. MODE SET-RESET (B5 = 1 et B4 = 1)



2. MODE PULSE-STROBE (b5 = 1 , b4 = 0 , b3 = 1)

a) COTÉ A



b) CÔTÉ B

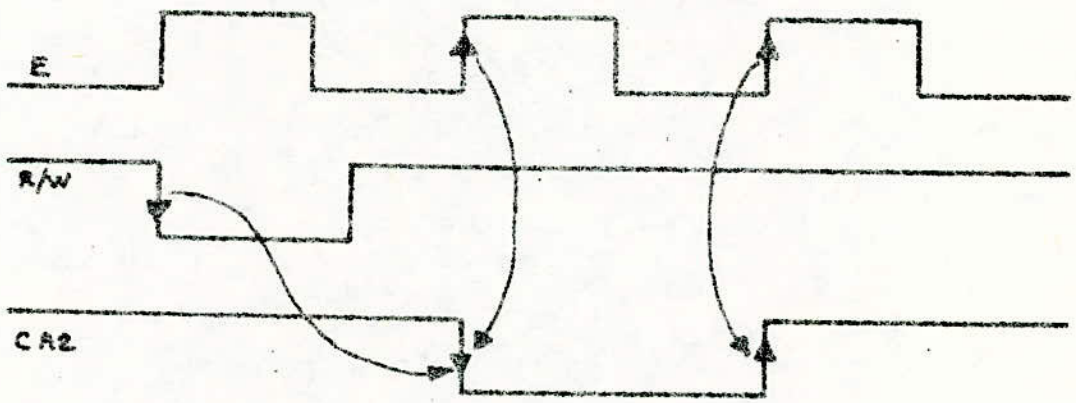
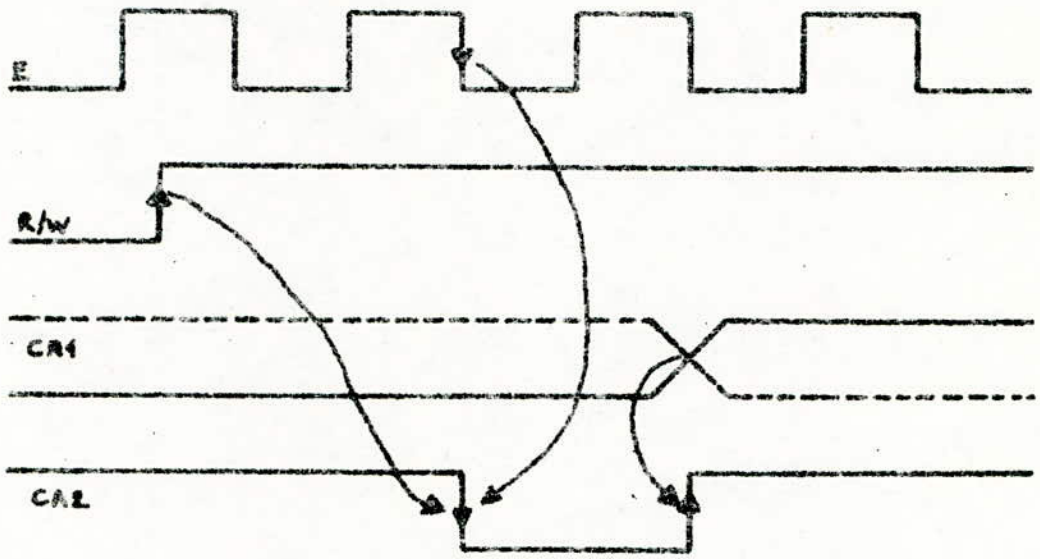


Fig. 18 Diagramme des temps de CA2 et CB2 (en Sortie)

3) MODE HANDSHAKING (b5=1 ; b4=0 ; b5=0)

a) côté A



b) côté B

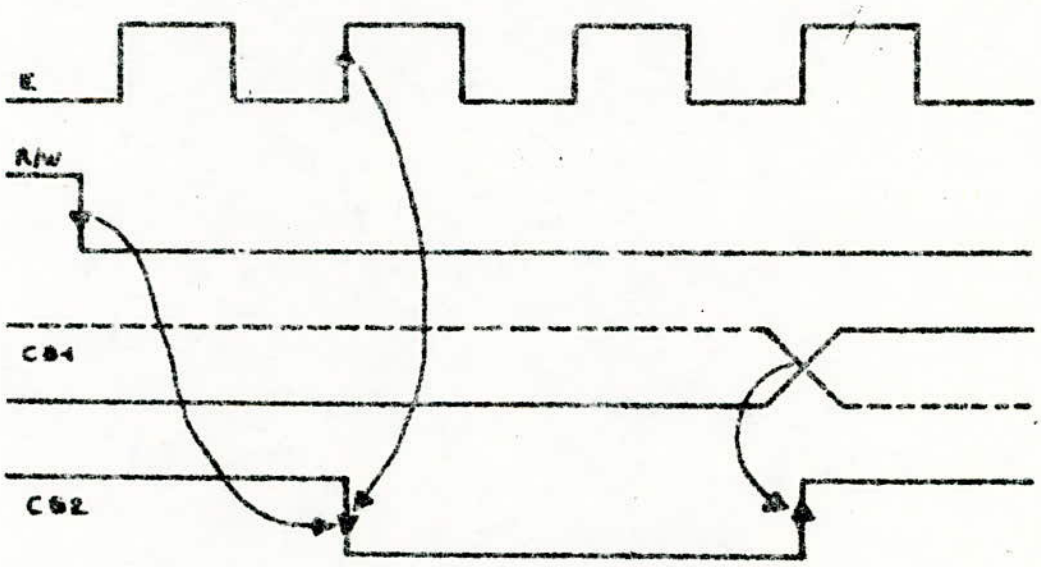


Fig.19 diagramme des temps de CA2 et CB2 (en sortie)

Le diagramme des temps des différents modes est donné par la figure 18 et la Figure 19.

7- Exemple de configuration du PIA

Etudions une configuration fonctionnelle du PIA donnée dans la figure 20. Les 8 bits du registre DDRA et DDRB sont écrits par programme respectivement avec des 0 et des 1 si bien que toutes lignes PA0-PA7 sont des entrées et les lignes PB0-PB7 sont des sorties. Les bits CRA0 et CRB0 des registres de contrôle CRA et CRB sont programmés à 1 autorisant les demandes d'interruption sur les lignes CA1 et CB1.
Ces demandes d'interruption se font sur des transitions négatives (CRA 1=0, CRB 1=0) et entraînent le positionnement à 1 des indicateurs d'interruption CRA7 ou CRB7 suivant que CA1 ou CB1 passe à l'état bas. Le passage à 1 de ces indicateurs d'interruption CRA7 ou CRB7 met à l'état bas les interrupt. request $\overline{\text{IRQA}}$ ou $\overline{\text{IRQB}}$ indiquant au MPU une demande d'interruption.

Les bits CRA2 et CRB2 étant à 1, les registres ORA et ORB sont sélectionnés, le PIA est donc prêt à effectuer des entrées de donnée sur la partie A et des sorties de donnée sur la partie B.

- les bits CRA5=1, CRA4=0, CRA3=1 donc la ligne CA2 agit comme sortie et fonctionne suivant le mode handshaking ; et chaque transfert de donnée vers le MPU fait passer la sortie CA2 à l'état bas. Et cette ligne CA2 repassera à l'état haut (état initial) dès que l'indicateur d'interruption CRA7 sera mis à 1 par une transition négative de CA1.

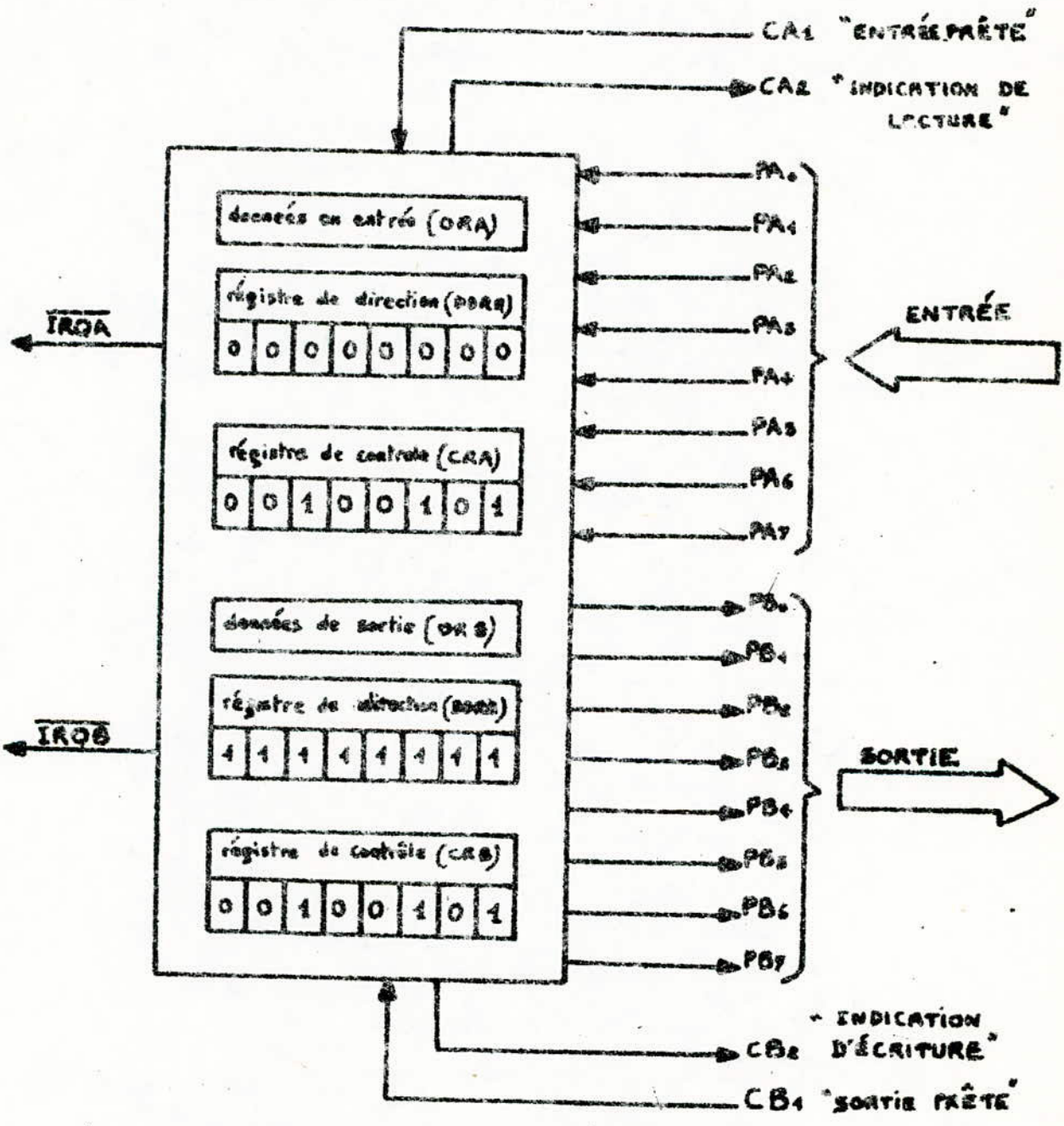


Fig.20 CONFIGURATION FONCTIONNELLE DU PIA

La présence d'un niveau bas sur CA2 indique au périphérique d d'entrée que la donnée a été transmise du PIA vers le MPU.

- les bits CRB 5=1, CRB 4=0, CRB 3=0 par conséquent CB2 agit comme sortie de commande et fonctionne suivant le mode handshaking, chaque transfert de donnée du MPU vers le registre ORB du PIA provoque une transition négative sur la ligne CB2 programmée en sortie. Cette ligne repassera à l'état haut à la prochaine transition négative de la ligne CB1. La présence d'un niveau bas sur la ligne CB2 programmé en sortie indique au périphérique de sortie que le PIA est prêt à lui envoyer une donnée.

Chapitre IV

ETUDE DU LECTEUR OPTIQUE MODEL 601 C-1

I. DESCRIPTION GENERALE

I.1- Format de la bande perforée

Le lecteur optique Model 601 C-1 peut lire des bandes à cinq, six, sept ou huit niveaux de perforations : un adaptateur de bande est prévu pour chaque type de bande. Nous étudierons plus particulièrement la bande à huit perforations.

Le ruban papier est d'un modèle standard de 25 mm de large et huit (8) perforations.

Parmi les huits lignes de perforations de la bande, sept représentent habituellement un caractère codée en ASC II (figure 21) et le huitième trou est utilisé pour vérifier la parité (ou bien, quelque fois, il est prévu pour être uniquement point ou espace). Le pas des caractères (l'espace entre le centre des trous) est de 2,5 mm, de telle sorte que la densité est de dix caractères par 25 mm. Lorsque le ruban est perforé, un neuvième, mais plus petit trou est également perforé : c'est le trou de validation (Sprocket). Le format de la bande perforée se présente comme le montre la figure 22. Les trois premiers caractères ØD, ØA, ØØ représentent respectivement :

- CR : Retour chariot
- LF : Line feed ou interligne
- NULL : Début

Fig.21 TABLEAU DE CODE A 7 ELEMENTS D'INFORMATION ASCII

b_7	b_6	b_5	b_4	b_3	b_2	b_1	Col / Lig.	0	0	0	0	1	1	1	1
								0	0	1	1	0	0	1	1
								0	1	0	1	0	1	0	1
								0	1	2	3	4	5	6	7
			0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
			0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
			0	0	1	0	2	STX	DC2	"	2	B	R	b	r
			0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
			0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
			0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
			0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
			0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
			1	0	0	0	8	BS	CAN	(	8	H	X	h	x
			1	0	0	1	9	HT	EM	)	9	I	Y	i	y
			1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
			1	0	1	1	11	VT	ESC	+	;	K	[	k	{
			1	1	0	0	12	FF	FS	,	<	L	\	l	¦
			1	1	0	1	13	CR	GS	—	=	M	]	m	}
			1	1	1	0	14	SO	RS	.	>	Ñ	^	n	~
			1	1	1	1	15	SI	US	/	?	O	—	o	DEL

La caractère suivant 53 correspondant à S représente le début d'enregistrement.

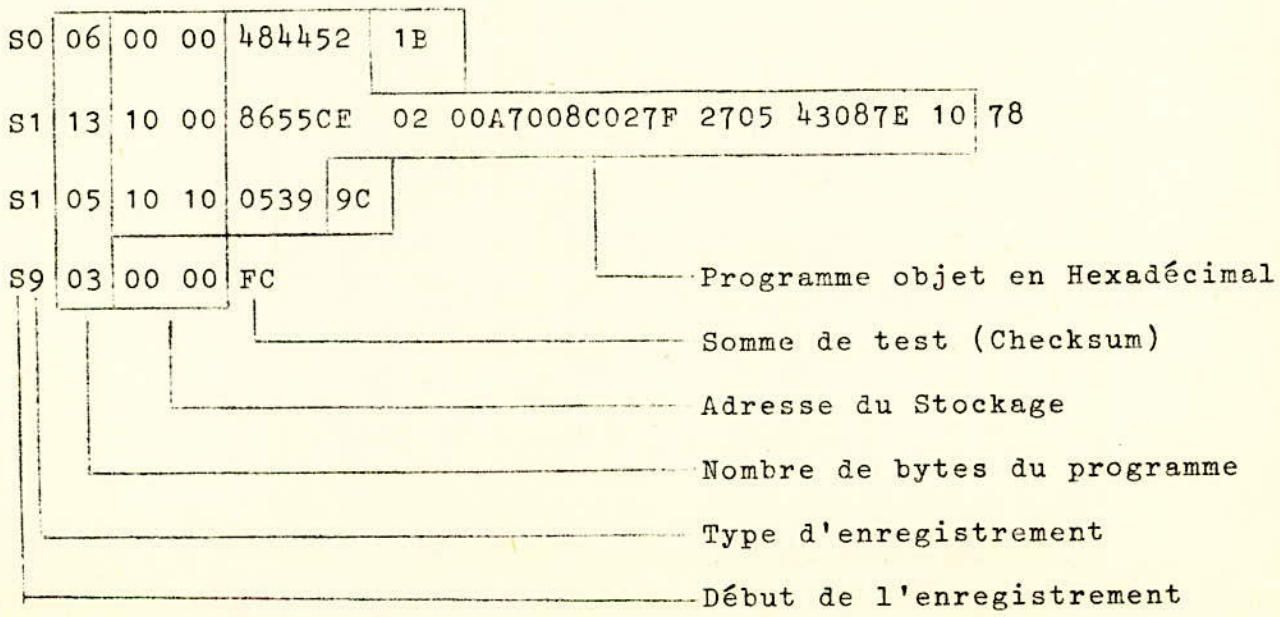
Le caractère CC indique le type d'enregistrement :

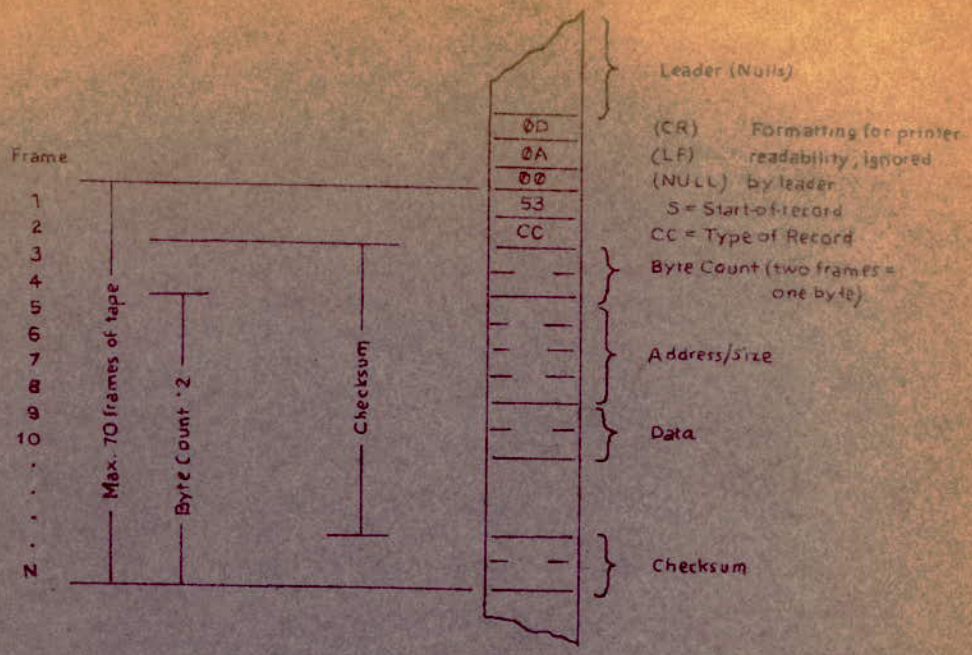
- CC = 30 : Entête
- CC = 31 : Enregistrement des données
- CC = 39 : Fin d'enregistrement

Pour chaque type d'enregistrement on a le Byte Count (compteur de bytes représentant le nombre d'octets de l'enregistrement en hexadécimal), l'adresse du premier byte de données et les données.

A la fin de chaque type d'enregistrement on a le Checksum ou somme de test : il représente le complément à 1 de la somme sur huit bits de tous les octets. Le Checksum sert à dépister les erreurs.

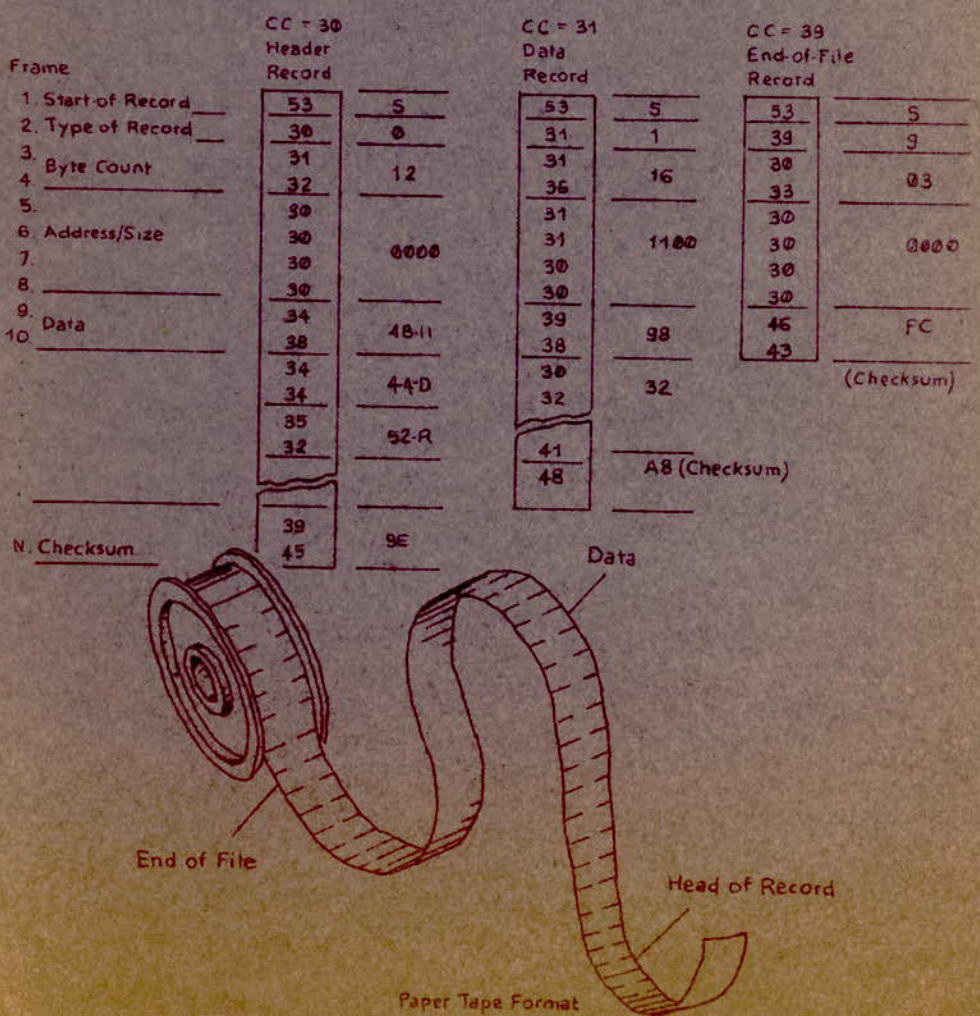
Pour terminer nous donnons un programme objet enregistré sur une bande puis tel qu'il est chargé dans le MPU.





Frames 3 through N are hexadecimal digits (in 7-bit ASCII) which are converted to BCD. Two BCD digits are combined to make one 8-bit byte

The checksum is the one's complement of the summation of 8-bit bytes



Le programme objet qui sera executé par le MPU se présente sous la forme de chaînes de caractères et sera logé dans la mémoire du MPU. Certaines valeurs constituent des informations pour le microprocesseur, d'autres représentent des valeurs hexadécimales données.

I.2- Système de détection de perforations

Le lecteur optique possède un réseau de diodes L.E.D (light émitting diode) (figure 23) émettant de la lumière qui balaye la bande de papier. La lumière émise par les diodes est infrarouge donc non visible à l'oeil nu.

Pour éviter tout risque d'interférences pouvant introduire une mauvaise lecture, on a disposé entre la bande de papier et le réseau de photo-transistor une plaque comportant des ouvertures de façon à ce que un rayon de lumière, après avoir traversé une perforation, soit canalisé vers le photo-transistor correspondant à la perforation.

Les perforations de la bande de papier sont détectées optiquement quand la lumière provenant des diodes L.E.D passe à travers les perforations de la bande et les ouvertures de la plaque sensibilisant ainsi le système des photo-transistors. Le courant-photo résultant est amplifié, puis transformé en un signal de sortie compatible TTL.
Lorsqu'un photo-transistor détecte le trou de validation (Sprocket), il génère un signal de validation qui initialise la lecture de la configuration codée. La présence du trou de validation indique que les trous de code sont positionnés correctement en face des têtes de lecture.

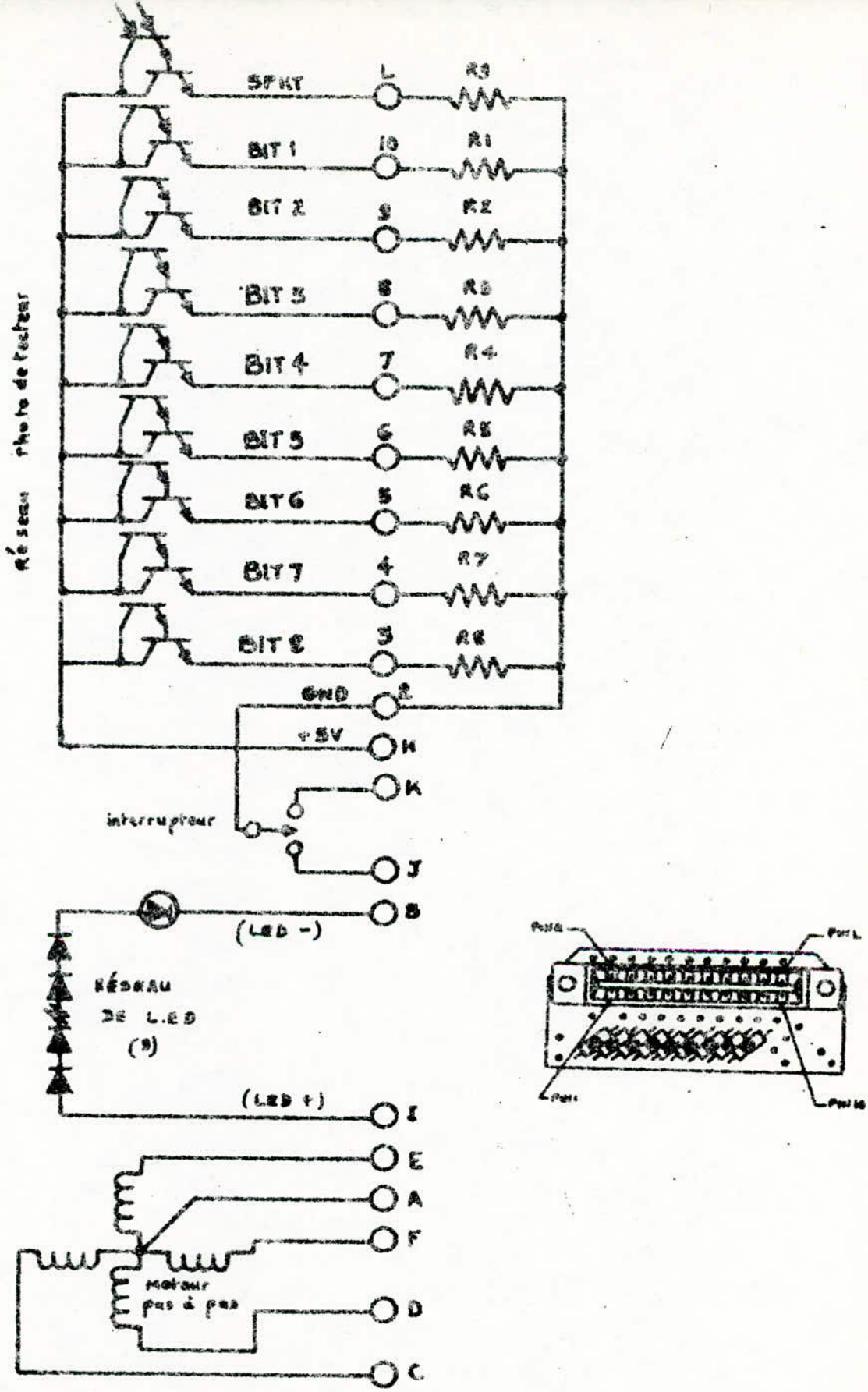


Fig 23 PLANCHE DE JONCTION SCHÉMATIQUE

LECTEUR TYPE 601C-1

La figure 23 représente le réseau de photo-transistors qui détecte les trous correspondant aux huit bits des Data (données) et au signal de validation Sprocket.

I.3- Mouvement de la bande

Le mécanisme d'avance du ruban est constitué par une roue de pincement (Sprocket) qui permet un contrôle précis du mouvement de la bande et une vitesse d'avance très rapide. Le mécanisme d'avance du ruban utilise un moteur pas à pas type Addmaster 4 phase 15°. Le moteur est entrainé dans la phase dual par alimentation séquentielle de deux enroulements adjacents (figure 23).
Le moteur pas à pas utilisé est de classe F donc d'une technologie élevée : meilleure isolation, bonne précision, pouvant même opérer en sûreté à des températures supérieures à 155°C.

I.4- Circuit de commande du moteur : (figure 24)

I.4.1- Bascule D

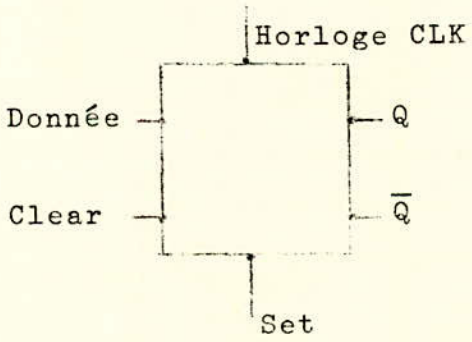


Schéma symbolique

Les entrées asynchrones Set et Clear la mettent en 1 ou 0 indépendamment de l'horloge CLK. Le mode synchrone est défini

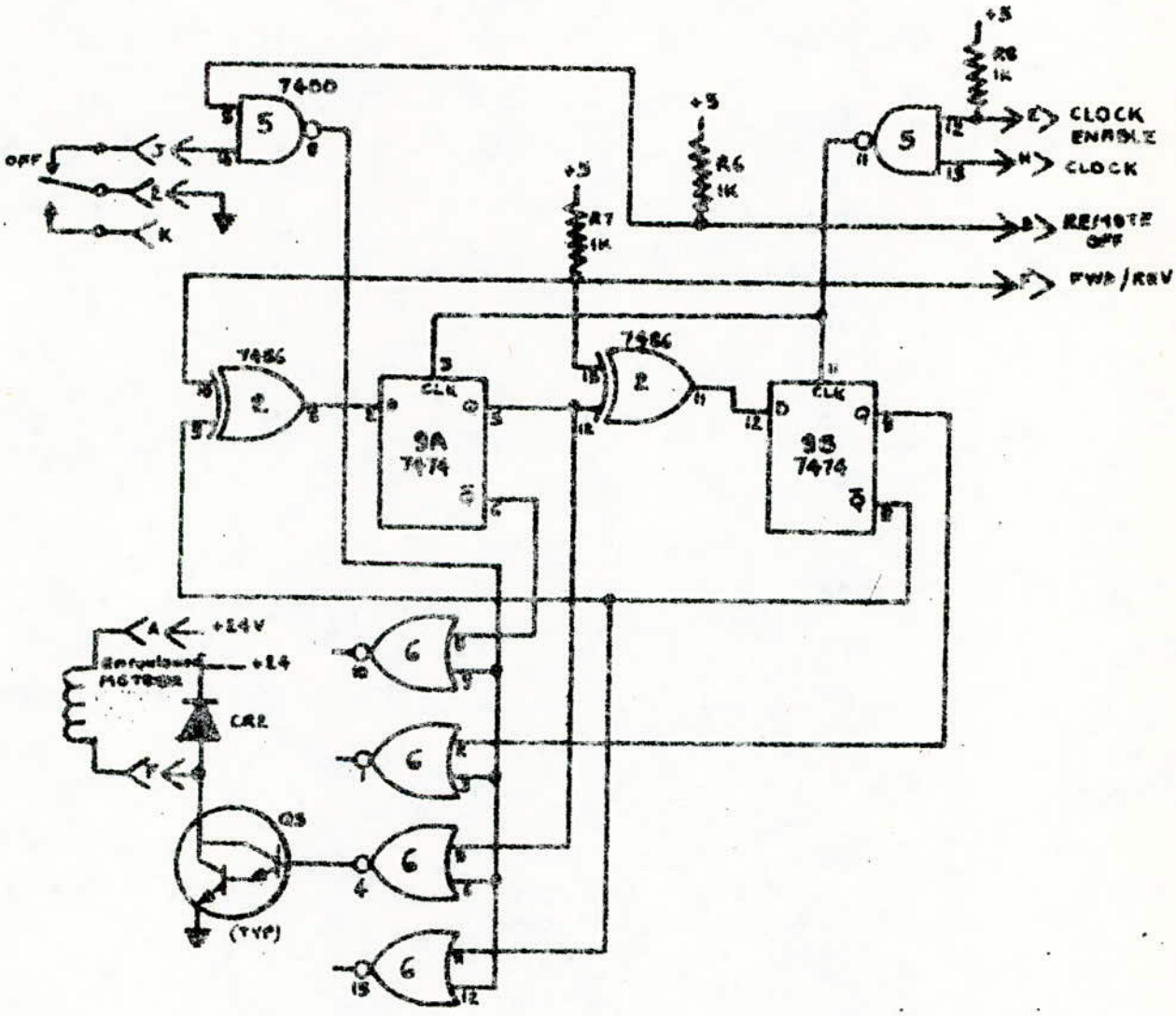


Fig. 24 CIRCUIT DE COMMANDE DU MOTEUR

par l'entrée D et l'horloge. La bascule commute au voisinnage du front montant de l'horloge.

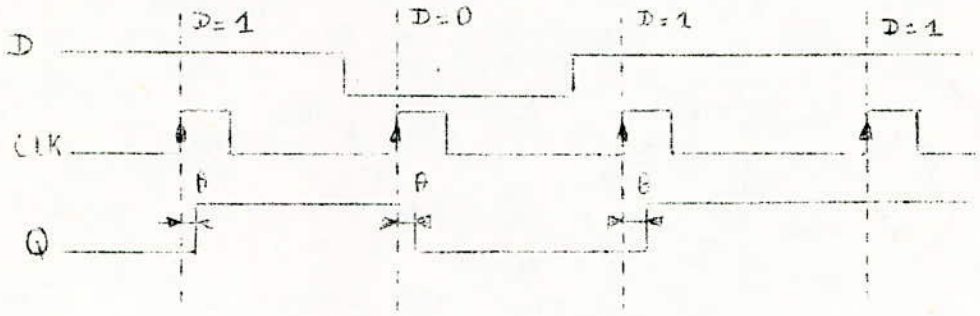
Table de vérité d'une bascule D.

	Entrées				Sorties		Remarques
	Set	Clear	Clock	D	Q	$\overline{Q}$	
mode Asynchrone	0	1	X	X	1	0	Affichage à 1
	1	0	X	X	0	1	Affichage à 0
	0	0	X	X	1*	1*	Etat instable
mode Synchrone	1	1	↑	1	1	0	Affichage à 1
	1	1	↑	0	0	1	Affichage à 0
	1	1	0	X	Q	$\overline{Q}$	Pas de changement

D'après cette table nous voyons que :

- Les entrées Set et Clear sont prioritaires
- L'état de la sortie est instable si Set et Clear sont actifs simultanément
- La sortie Q suit l'état de l'entrée D après action du front de montée de l'horloge.

Diagramme des temps.



La bascule D recopie l'état de l'entrée D après le front montant de l'horloge.

I.4.2- Utilisation d'un compteur à bascules D

Les bascules A et B de type D forment un compteur réversible pour le moteur. Le compteur s'incrémente pendant la transition négative du signal CLK ou se décrémente si la ligne FWD/$\overline{\text{REV}}$ est mise à l'état bas. Le compteur permet au moteur une marche en "dual phase" étant donné qu'il existe toujours deux enroulements adjacents qui sont alimentés.
Pour arrêter l'alimentation du moteur il suffit de mettre l'interrupteur sur la position off ou bien par commande de l'entrée Remote off qui sera mise à 0.

I.5- Circuit des diodes L.E.D : (figure 25)

Tant que l'interrupteur n'est pas dans la position off ou que la ligne Remote off n'est pas mise à "0", un courant collecteur d'intensité nominale 29 mA traverse la résistance R3 pour alimenter les diodes L.E.D.

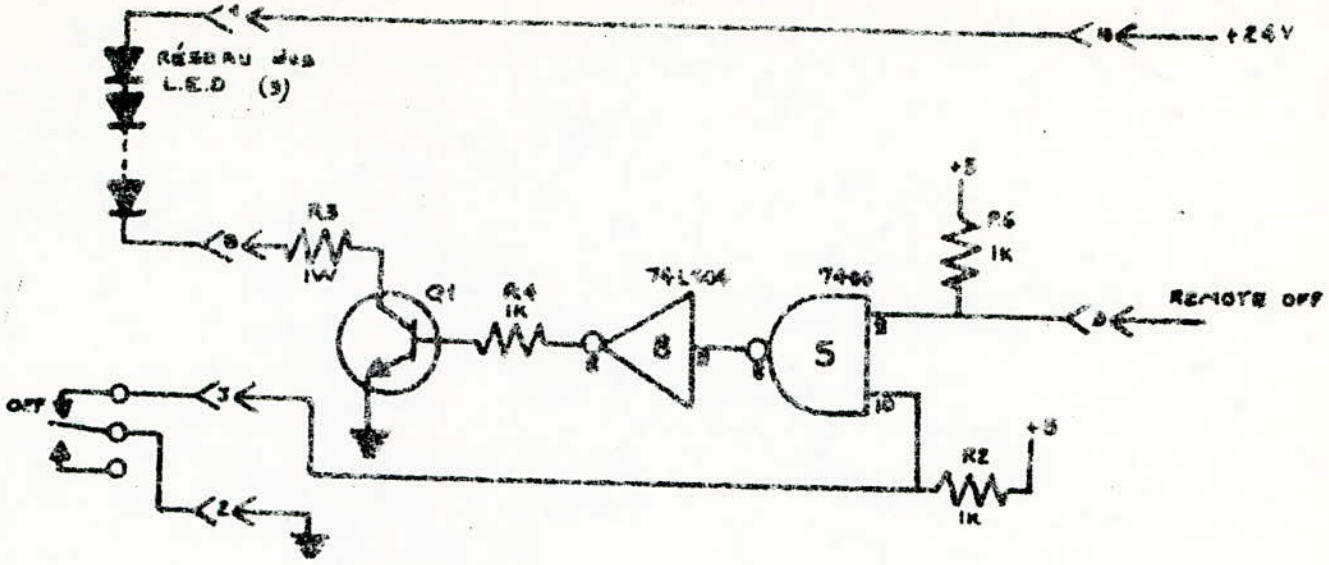


Fig.25 CIRCUIT DES LED

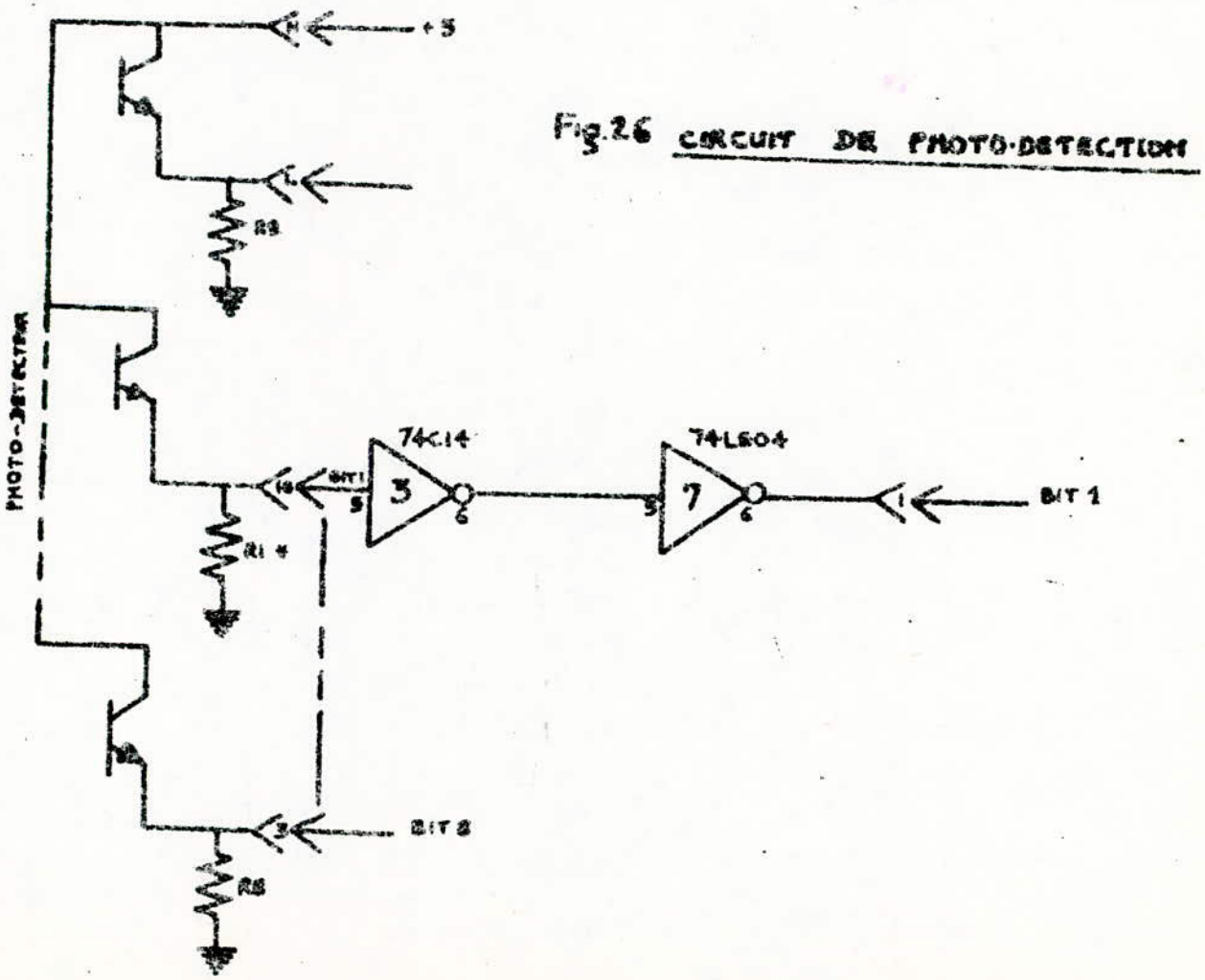
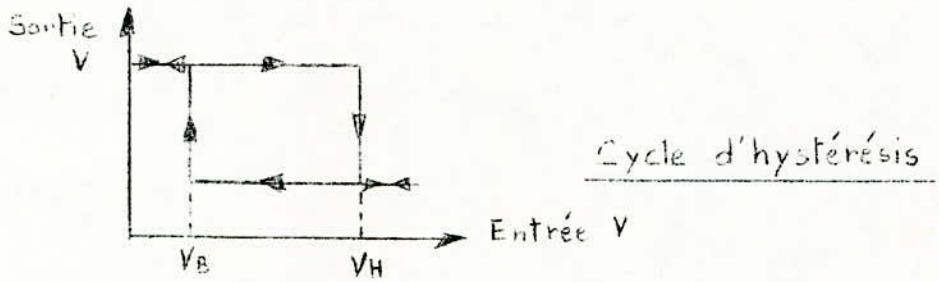


Fig.26 CIRCUIT DE PHOTO-DETECTION

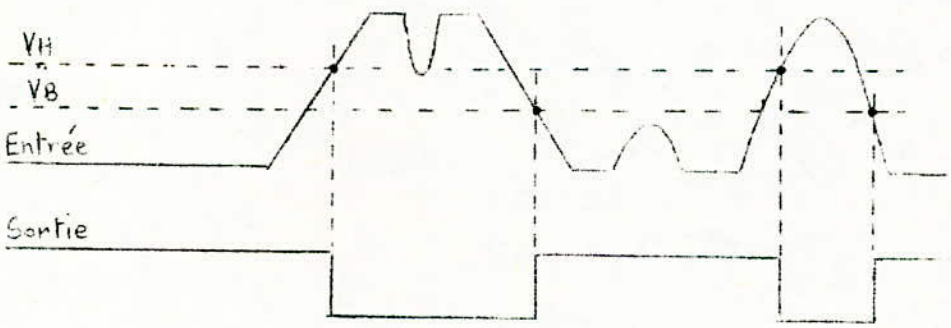
I.6- Circuit détecteur photo électrique

I.6.1- Principe du Trigger de Schmitt :

En général un circuit logique nécessite en entrée des signaux à flancs raides, sinon des oscillations apparaissent en sortie, provoquant des défauts dans la commande des circuits séquentiels. Les circuits dits Trigger de Schmitt, dérivés de la bascule de Schmitt à contre réaction, admettent en entrée des signaux à fronts lents.



Cycle d'hystérésis



Propriétés du Trigger de Schmitt

La figure précédente donne la courbe de transfert de ce circuit. Elle fait apparaitre deux seuils VH (seuil haut) et VB (seuil bas). L'hystérésis (VH-VB) assure une certaine immunité au bruit.

I.6.2- Circuit photo-détecteur : (figure 26)

L'aptitude du Trigger de Scmitt à traiter des signaux à fronts lents est utilisé dans les circuits photo-détecteurs.

Lorsque une cellule photo électrique est sensibilisée (présence d'un trou) il en résulte un courant photo. Le courant photo est alors amplifié par le Trigger de Schmitt (CMOS).

Les résistances R* sont utilisées pour normaliser les oscillations dûes au voltage de chaque ligne (exemple : ligne du bit 1) : elles sont contenues dans un circuit imprimé interchangeable.

I.7- Circuit de contrôle : (figure 27)

Quand l'interrupteur est sur la position "Start" ou que le signal Remote Start est momentanément bas, Reader Ready devient alors actif (mis à 1) par la mise des signaux Clear à 1 des bascules A et B du type D.

Le signal Set de la bascule A est mis à 1 avec une pulsation d'horloge ; ce même signal est mis à 0 quand le signal Sprocket est mis haut. Si par contre il n'y a pas de transition du signal sprocket donc que les sorties de la bascule A restent stables, le signal Set de la bascule B est mis à 1 et reste dans cet état, mettant le signal Reader Ready à 0 donc non actif.

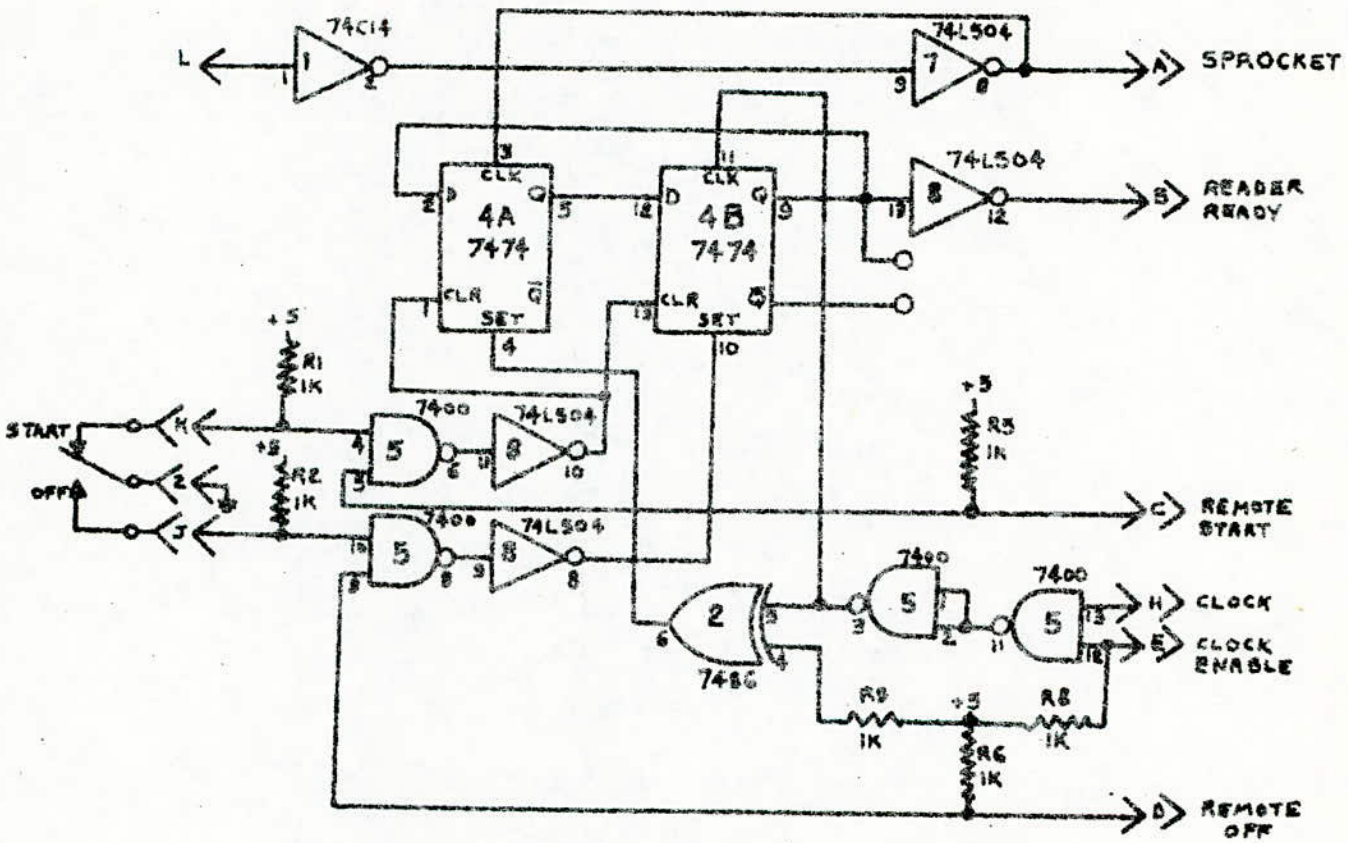
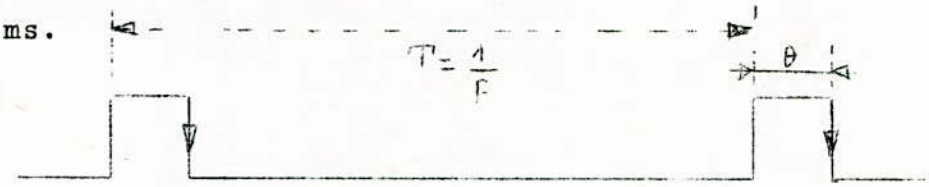


Fig. 2.7 CIRCUIT DE COMMANDE

II. ETUDE DES ENTREES ET SORTIES DU LECTEUR

II.1- Clock : Le signal Clock ou Horloge est une entrée pour le lecteur. Sa fréquence ne doit pas excéder 150 Hertz et la largeur de bande θ du signal doit être comprise entre 1 micro-seconde et 1 ms.



Durant la transition négative du signal Clock, le compteur moteur change d'état, cependant, le moteur ne répond mécaniquement que dans un temps supérieur ou égal à 2 ms.

II.2- Clock Enable : Si ce signal d'entrée est mis à 0, le moteur qui est en marche sera arrêté. Cette ligne peut être laissée flottante si elle n'est pas utilisée.

II.3- Forward/Reverse (F/$\overline{R}$: Cette entrée inverse le sens de rotation du moteur si elle est mise à 0.
Elle peut être laissée flottante si on le désire mais elle ne doit pas changer d'état pendant une période si l'on veut inverser le sens de rotation du moteur.

II.4 - Remote Start et Remote Off : Ces entrées ne sont actives que lorsque elles sont mises à l'état logique 0. Ces deux signaux contrôlent le signal Reader Ready comme nous allons le voir dans l la suite.

La ligne Remote Off, quand elle est mise à 0, supprime le courant dans le moteur et les diodes L.E.D.

II.5- Reader Ready

- Ce signal de sortie est mis à un niveau logique 1 si

  a) - l'interrupteur est dans la position Start ou bien
  b) - le signal Remote Start devient momentanément bas à condition que l'interrupteur ne soit pas dans la position off.

- La sortie Reader Ready sera mise à l'état logique 0 si

  a) - l'interrupteur est mis sur la position off ou
  b) - le signal Remote off est mis momentanément à 0 ou
  c) - lors de la détection de la fin de la bande ou
  d) la bande s'est accrochée

II.6- Data Bits et Sprocket :

Les huit "Data Bits" ou bits des données lisent les informations portées sur les huit canaux de la bande tandis que le signal Sprocket lit le bit de validation.

La présence d'un trou est indiquée par un signal à un niveau logique1 .

Les "Data Bits" et Sprocket lisent continuellement la bande, cependant cette lecture ne doit se faire que si un caractère est positionné. Le positionnement d'un caractère est validé par Sprocket ; la lecture ne se fait que 2 ms à 5 ms après la transition négative du signal Clock.

III. INTERFACE LECTEUR - PIA

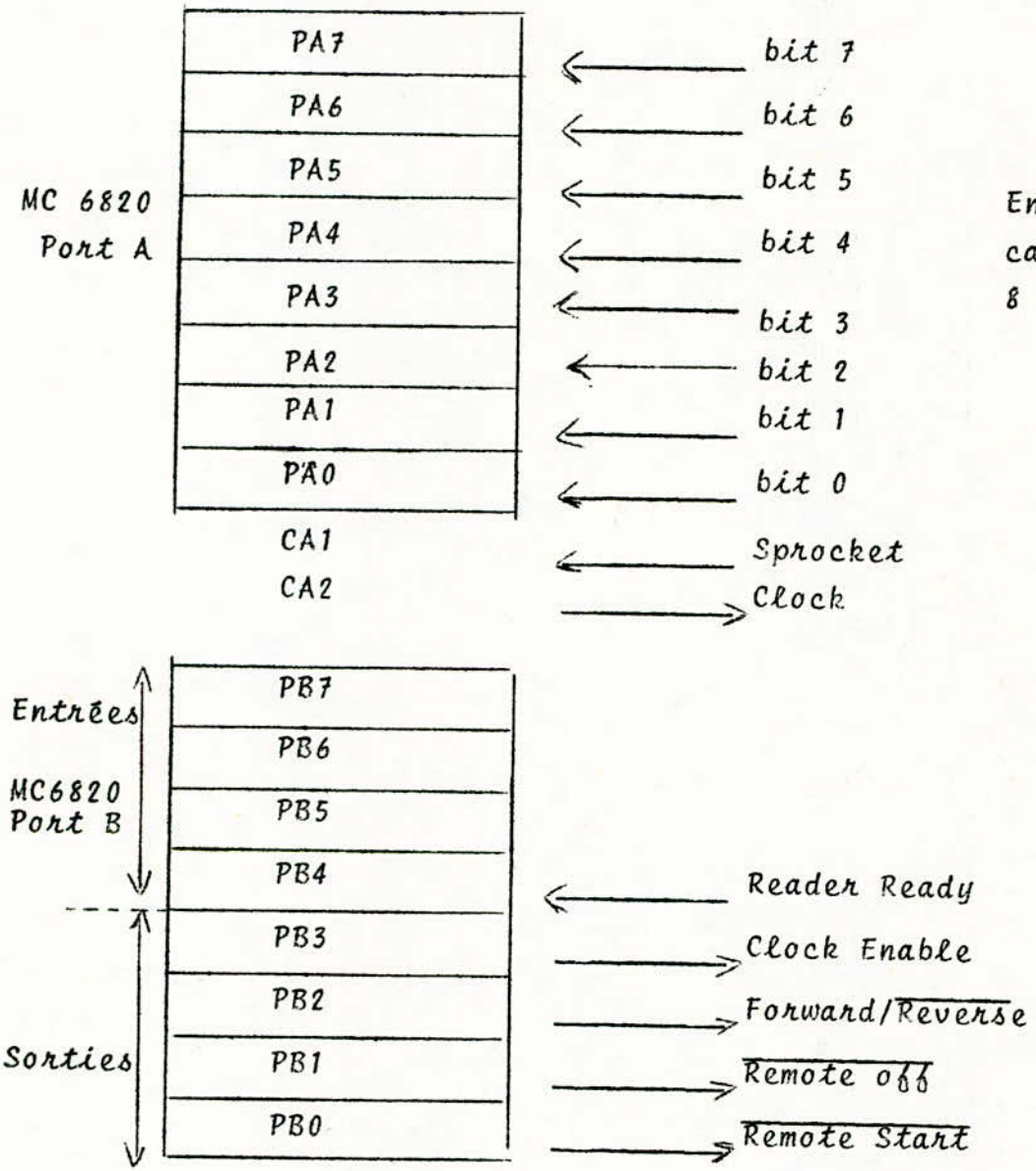
Les signaux d'entrées et sorties du lecteur optique sont tous compatibles T.T.L ce qui permet leurs connexions directes au circuit d'interface microprocesseur (P.I.A) sans recourir à des "buffers" ou circuits logiques additionnels.

III.1- Assignement des signaux vers le PIA

L'interface entre le lecteur optique et le MPU est constitué par quinze lignes : cinq signaux vers le lecteur et dix signaux vers le PIA. Ces quinze signaux sont très efficacement pris en charge par les configurations A et B du circuit adaptateur d'interface de périphérique (P.I.A). Les huit lignes de données PA0-PA7 correspondent aux huit canaux de perforations de la bande de papier. Le signal de commande Clock est généré par la ligne de contrôle CA2 et le signal de validation Sprocket arrive sur l'entrée CA1. Les autres signaux sont assignés de la façon suivante au port B du PIA :

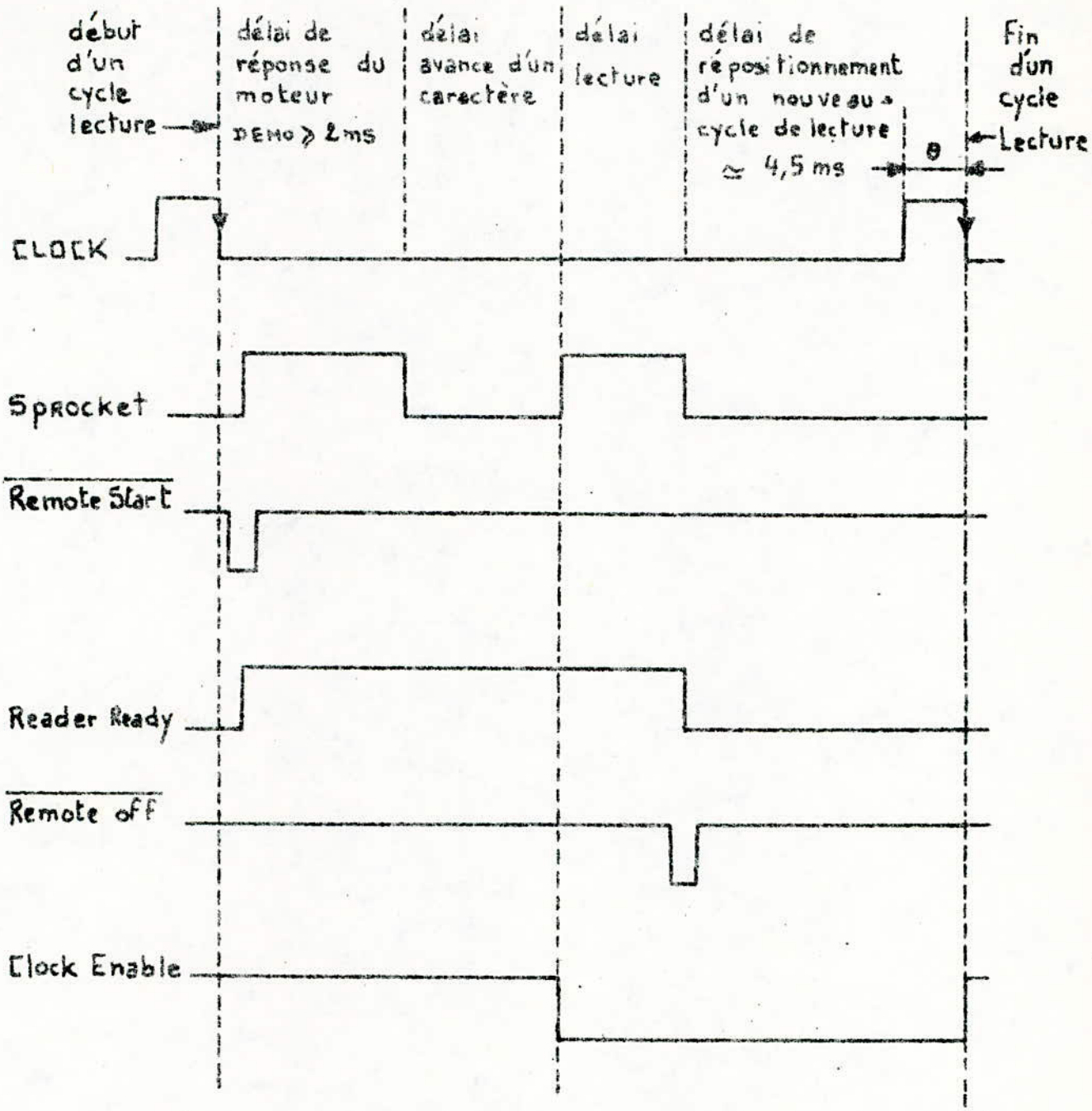
- Reader Ready affecté comme entrée au bit 4 (PB4)
- Clock Enable, Forward/Reverse, Remote off, Remote Start affectés, comme sorties, respectivement aux bits PB3, PB2, PB1 et PB0.

Les caractéristiques de fonctionnement des lignes de contrôle CA1, CA2 et la direction des lignes de données sont déterminés par la programmation respective des régistres de contrôle CRA(B) et des registres de direction des données DDRA (B) du P.I.A.



III.2- Timing de la logique de contrôle du lecteur :

Le lecteur a besoin d'un signal de commande CLOCK pour initialier la lecture de chaque caractère. Les caractéristiques du lecteur indiquent que ce signal doit être une impulsion positive d'une durée comprise entre 1 micro-seconde et 1 ms dont la fréquence



Forward / Reverse peut être haut ou bien bas

TIMING de la LOGIQUE de COMMANDE du LECTEUR

ne doit pas dépasser 150 Hertz.

La solution "Software" aux exigences du signal Clock est de le contrôler directement à partir de CA2 programmé comme sortie suivant le mode Set-Reset c'est-à-dire qu'il prend le même état logique que celui du bit CRA3 du P.I.A.

La transition négative du signal Clock amorce le cycle lecture d'un caractère ; Remote Start momentanément bas (7 cycles M.P.U) met le signal Reader Ready à 1 ceci dans le but de valider les Data bits et Sprocket qui lisent continuellement la bande perforée. Clock Enable est programmé haut.

Forward/Reverse peut être mis soit haut, soit bas, Après la transition négative de Clock, le moteur répond mécaniquement à partir de 2 ms. Le P.I.A. positionne le bit 7 de son registre de contrôle CRA lorsque une transition active (front montant) est détectée sur sa ligne de contrôle CA1 définie préalablement comme entrée.

Dans le cas présent ceci se produit lors de la réception du signal de validation Sprocket qui en fait valide le positionnement d'un caractère en face des têtes de lecture.
Le moteur doit alors être arrêté par une transition négative de Clock Enable : la matrice de trous codée est lue dans l'accumulateur du MPU Via le registre de données du P.I.A.

Pendant le délai de lecture le MPU réceptionne le caractère lue par le lecteur et exécute un sous programme de décodage ASCII-Hexadécimal.

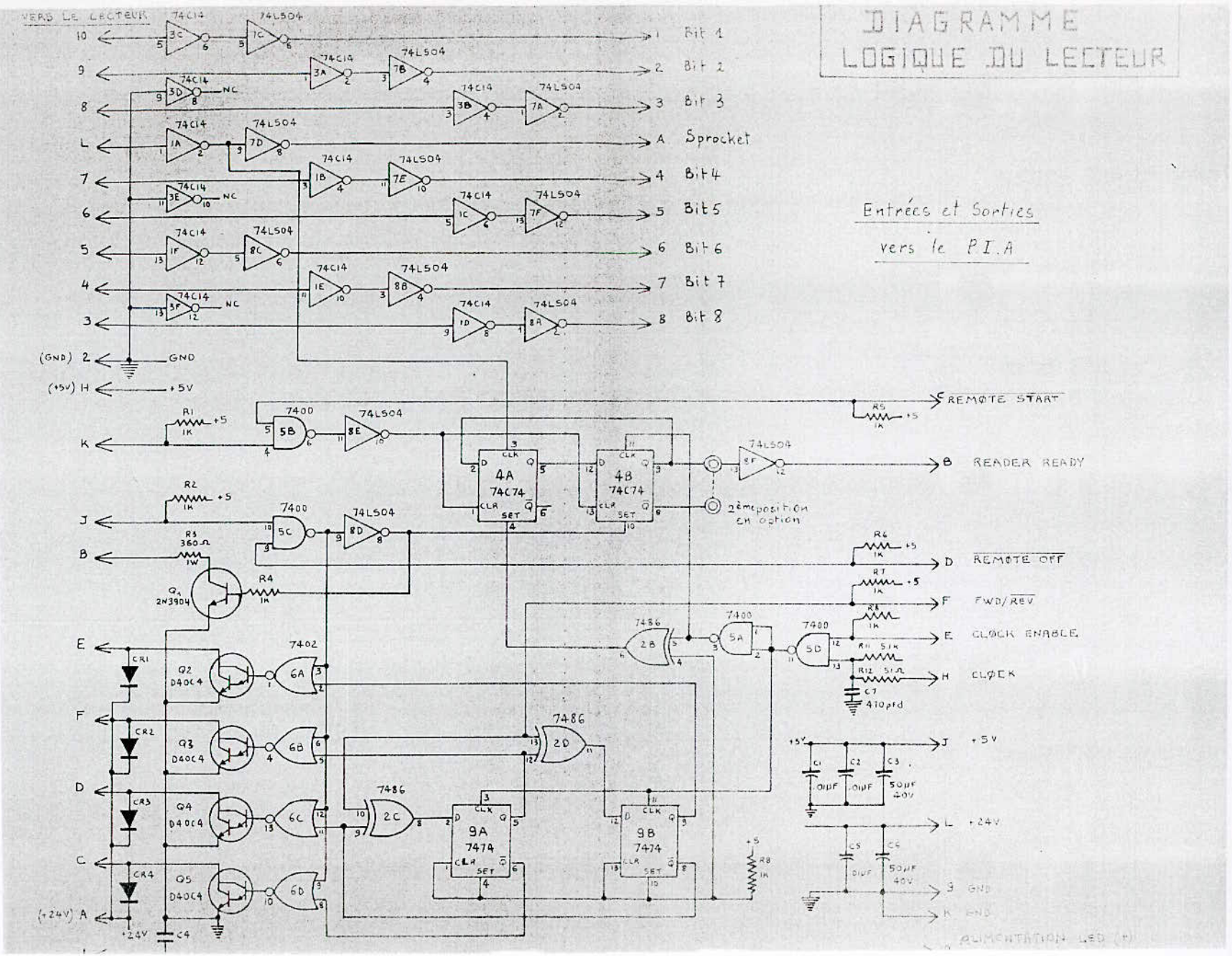
Avant de remettre le signal Clock à 1, un délai de 4,5 ms est exécuté par le MPU. On mettra auparavant, le signal Reader

Ready bas par la mise momentanée (19 cycles MPU) du signal Remote Off à zéro. Lorsque la valeur du nombre chargé dans l'accumulateur pour exécuter le délai de 4,5 ms atteint la valeur zéro, le MPU sort de la boucle du délai et charge un nouveau mot de contrôle dans le registre CRA du PIA qui initialise à nouveau le bit CRA3 et ceci place le signal Clock à 1.

Dans le programme de gestion du lecteur la largeur 0 est évaluée à :

- 20 cycles MPU lorsque le signal FWD/$\overline{\text{REV}}$ est haut
- 36 cycles MPU lorsque le signal FXD/$\overline{\text{REV}}$ est bas.

DIAGRAMME LOGIQUE DU LECTEUR



Entrées et Sorties

vers le P.I.A

Chapitre V

PROGRAMMATION

Le Système M 6800 - lecteur optique étant conçu du point de vue hardware, il s'agit d'élaborer un logiciel (software) permettant la gestion et le contrôle de la logique de commande du lecteur.

Le programme est une série d'instructions mises en mémoire à l'intérieur du MPU de façon séquentielle. Les instructions sont rentrées en code machine hexadécimal octet par octet. Chaque instruction définit une phase de fonctionnement du microprocesseur.

Les programmes sont écrits en hexadécimal et en langage symbolique. En langage symbolique, les codes opérations sont remplacés par des symboles mnémoniques.

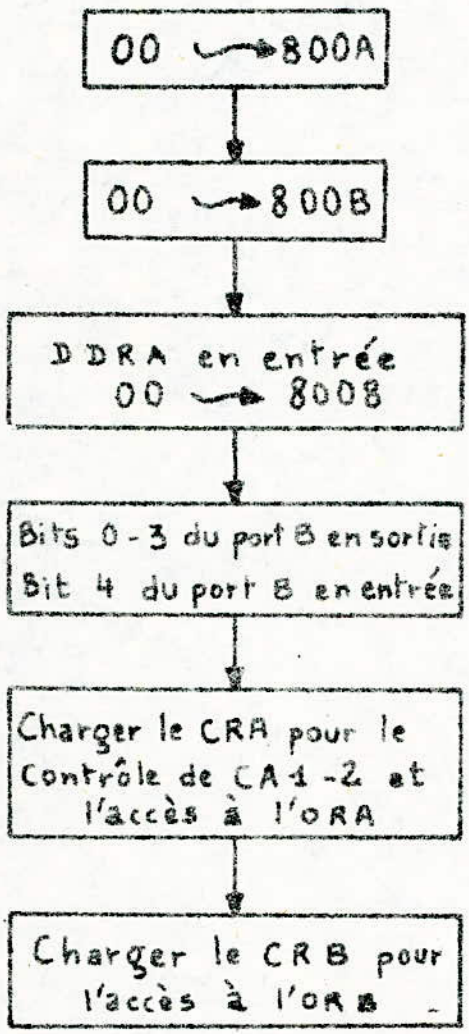
1/ Sous-programme : Initialisation du PIA

8008 : adresse commune aux registres de données (ORA) et de direction de données (DDRA)

8009 : adresse commune aux registres de données (ORB) et de direction de données (DDRB)

800A : registre de contrôle (CRA)

800B : registre de contrôle (CRB)



I. SOUS PROGRAMME : Initialisation du PIA

1- Programme

```
0100   7F 800A    INIT   CLR 800A
0103   7F 800B           CLR 800B
0106   7F 8008           CLR 8008
0109   86 0F           LDAA 0E
010B   B7 8009           STAA 8009
010E   86 3E           LDAA 3E
0110   B7 800A           STAA 800A
0113   86 04           LDAA 04
0115   B7 800B           STAA 800B
```

2- Commentaire

Initialiser le PIA, c'est le préparer à reçevoir et transmettre les données, voire définir la direction de transfert (entrée ou sortie) et programmer les lignes de contrôle CA (B) 1 et CA2 (B) 2.

a) P.I.A côté A

Le DDRA est programmé pour établir le sens entrant des Data Bits. En chargeant le DDRA par 00 les lignes PA0-PA7 sont des entrées. La configuration 3E dans le registre de contrôle CRA aura les conséquences suivantes :

- CRA2 = 1 : Selection du registre de données ORA.

- L'interruption est masquable (CRA0 : 0)

- La ligne de contrôle CA1 sera active sur le front montant (CRA1 = 1)

- La ligne de contrôle CA2 est programmée en mode Set-Reset (CRA5 = CRA4 = 1 ; CRA3 = 1 ou 0) pour le contrôle du signal CLOCK.

b) PIA côté B

Les bits PB0 - PB3 du DDRB sont programmés en sortie ; le bit PB4 en entrée.
La configuration du registre de contrôle CRB permet la selection du registre de données ORB.

II. Initialisation du Compteur CMPT

1- Programme

```
0118    86 03          LDAA 03
011A    B7 028F        STAA 028F
```

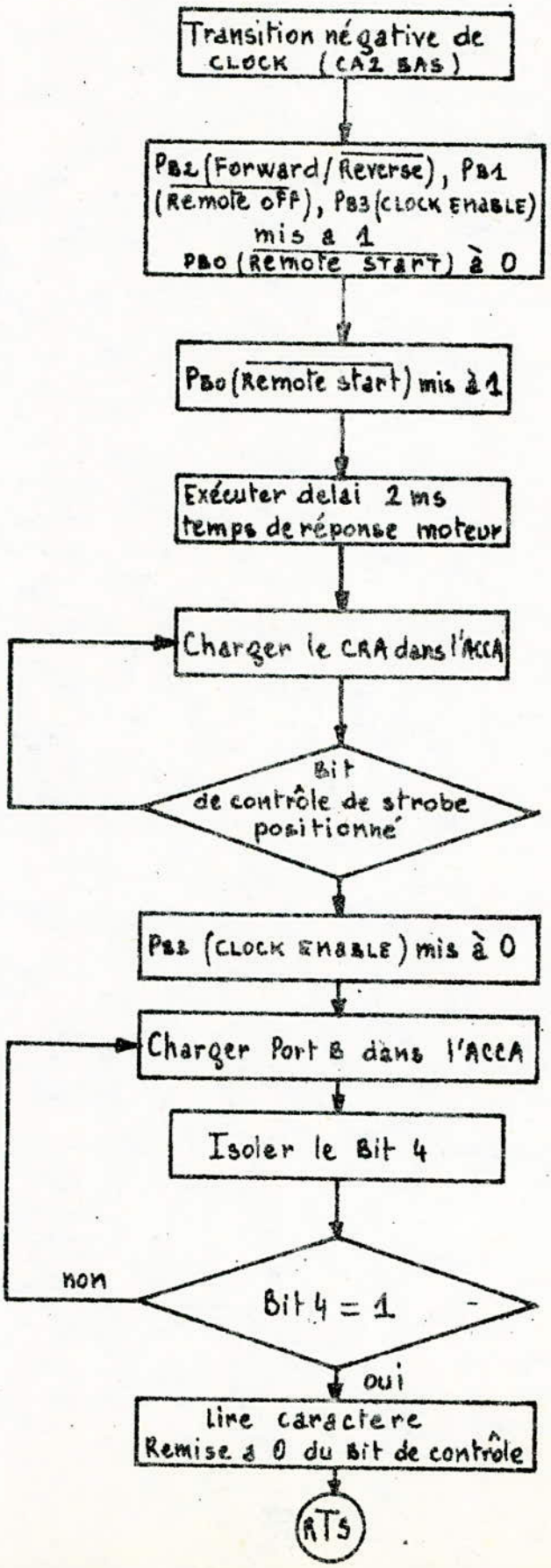
2- Commentaire

Le compteur CMPT est chargé par 3, son utilisation est la suivante : le MPU, après avoir réceptionné les octets d'une série de données avec checksum, calcule son propre checksum correspondant à la somme sur 8 bits de tous les octets de la série.
Le MPU teste le checksum : si une erreur est détectée il fera appel à un sous programme qui lui permettra de relire les données de la série.

A chaque erreur du checksum le compteur est décrémenté. Si l'erreur checksum est reproduite 3 fois de suite pour une même série le contenu du compteur CMPT est nul et un message erreur est émis.

L'explication sera plus explicite aux sous-programmes CPR et CPR3.

III / Sous programme : Avance d'un caractère (AVAC) :



III. SOUS PROGRAMME AVAC : "Avance d'un caractère"

1- Programme

```
01B3   86 32      AVAC   LDAA 32
01B5   B7 800A           STAA 800A
01B8   86 0E             LDAA 0E
01BA   B7 8009           STAA 8009
01BD   86 1F             LDAA 1F
01BF   B7 8009           STAA 8009
01C2   CE FA      DEMO   LDX  FA
01C4   09         LOP1   DEX
01C5   26 FD             BNE  LOP1
01C7   B6 800A    TST    LDAA 800A
01CA   2A FB             BPL  TST
01CC   86 17             LDAA 17
01CE   B7 8009           STAA 8009
01D1   B6 8009    LOP2   LDAA 8009
01D4   84 10             ANDA 10
01D6   27 F9             BEQ  LOP2
01D8   B6 8008           LDAA 8008
01DB   39
```

2- Commentaire

La configuration 32 du registre de contrôle CRA engendre une transition négative du signal Clock (CRA 3 = 0).

Les signaux Forward/Reverse, Remote off et Clock Enable sont positionnés haut lors du chargement de la configuration OE dans le registre de données ORB.

La transition négative du signal Clock fait changer d'état au compteur moteur ; cependant le moteur ne peut répondre mécaniquement qu'à partir de 2 ms d'où l'utilisation d'un délai de 2 ms. Le délai est généré de la façon suivante : le nombre 250 est chargé dans le registre d'index puis décrementé successivement par 1 dans une boucle du sous-programme qui teste si le contenu du registre d'index est nul après chaque passage au travers de la boucle.
Lorsque le contenu est nul, le MPU sort de la boucle et réalise ainsi le délai de 2 ms.

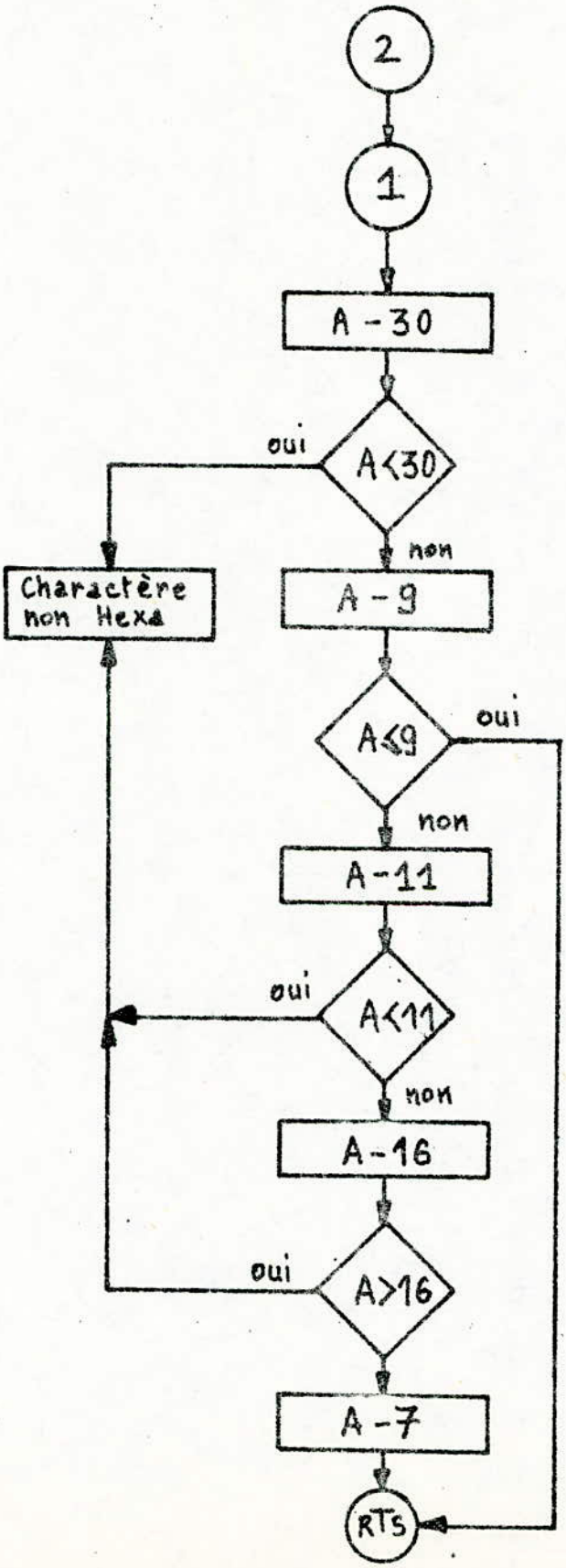
$$FA = (11111010)_2 = (250)_{10}$$

Les instructions DEX et BNE sont exécutées chacune en 4 cycles MPU soit 8 micro-secondes d'où le délai peut être calculé ainsi :

250 X 8 + 3 = 2003 micro-secondes

250 : contenu initial du registre d'index
8 micro-secondes : temps nécessaire pour l'exécution de BNE et DEX
3 micro-secondes : temps d'exécution pour l'instruction LDX

Après ce délai de 2 ms, la réponse du moteur permet d'avancer la bande perforée. La détection optique d'une perforation de validation (Sprocket positionné haut) positionne le bit de contrôle de strobe CRA7 à 1.
Clock Enable est mis immédiatement à 0 pour arrêter le moteur.
Un test de contrôle du signal Reader Ready (bit 4) s'avère nécessaire avant la lecture. Le bit de contrôle de strobe CRA7 est remis à 0 automatiquement par la lecture du contenu du registre de données ORA.


2
1
A - 30
oui
A<30
non
Caractère non Hexa
A - 9
A<9
oui
non
A - 11
oui
A<11
non
A - 16
oui
A>16
A - 7
RTS

IV. SOUS PROGRAMME INHEX : "Entrée d'un caractère hexadécimal"

1- Programme

```
0249   8D 9F      INHEX   BSR  FINI3
024B   80 30               SUBA 30
024D   2B 2C               BMI  NON HEXA
024F   81 09               CMPA 09
0251   2F 0A               BLE  TETA
0253   81 11               CMPA 11
0255   2B 24               BMI  NON HEXA
0257   81 16               CMPA 16
0259   2E 20               BGT  NON HEXA
025B   80 07               SUBA 7
025D   39          TETA    RTS
```

2- Commentaire

Un caractère introduit dans l'accumulateur par le sous-programme FINI 3 est codé en ASCII. Ce caractère ne peut être déchiffré par le MPU alors un décodage ASCII - Hexadécimal est nécessaire. Les caractères stockés sur la bande perforée sont codés en ASCII mais représentent des caractères hexadécimaux. La représentation alpha-numérique de 0-15 en hexadécimal est 0-9, A-F (10-15).
Cependant, les 8 bits en notation ASCII doivent être convertis en un nombre binaire de 4 bits (0000 - 1111).

Code ASCII	30	31	32	33	34	35	36	37	38	39
Code HEXA	0	1	2	3	4	5	6	7	8	9

Code ASCI	41	42	43	44	45	46
Code HEXA	A	B	C	D	E	F

Un caractère est hexadicimal s'il est compris entre 30 et 39 (chiffre) ou bien entre 41 et 46 (lettre).

Le caractère codé en ASCII est introduit dans l'accumulateur où on lui soustrait 30. Le contenu de l'accumulateur est comparé successivement à 0 et 9, 11 et 16.
On considère donc trois (3) cas :

i/ Le contenu de l'ACCA est compris entre 0 et 9 : on a un caractère hexadécimal en chiffre.

ii/ Le contenu de l'ACCA est compris entre 11 et 16 : on a un caractère hexadécimal en lettre

iii/ Le caractère n'appartient à aucun de ces deux intervalles, alors un message "NON HEXA" est émis.

L'instruction SUBA 7 permet le décodage ASCII à l'hexadécimal pour les lettres (A-F) : en effet supposons que l'accumulateur contienne, après l'instruction SUBA 30, le chiffre ; le MPU en executant l'instruction SUBA 7 détecte la lettre F.

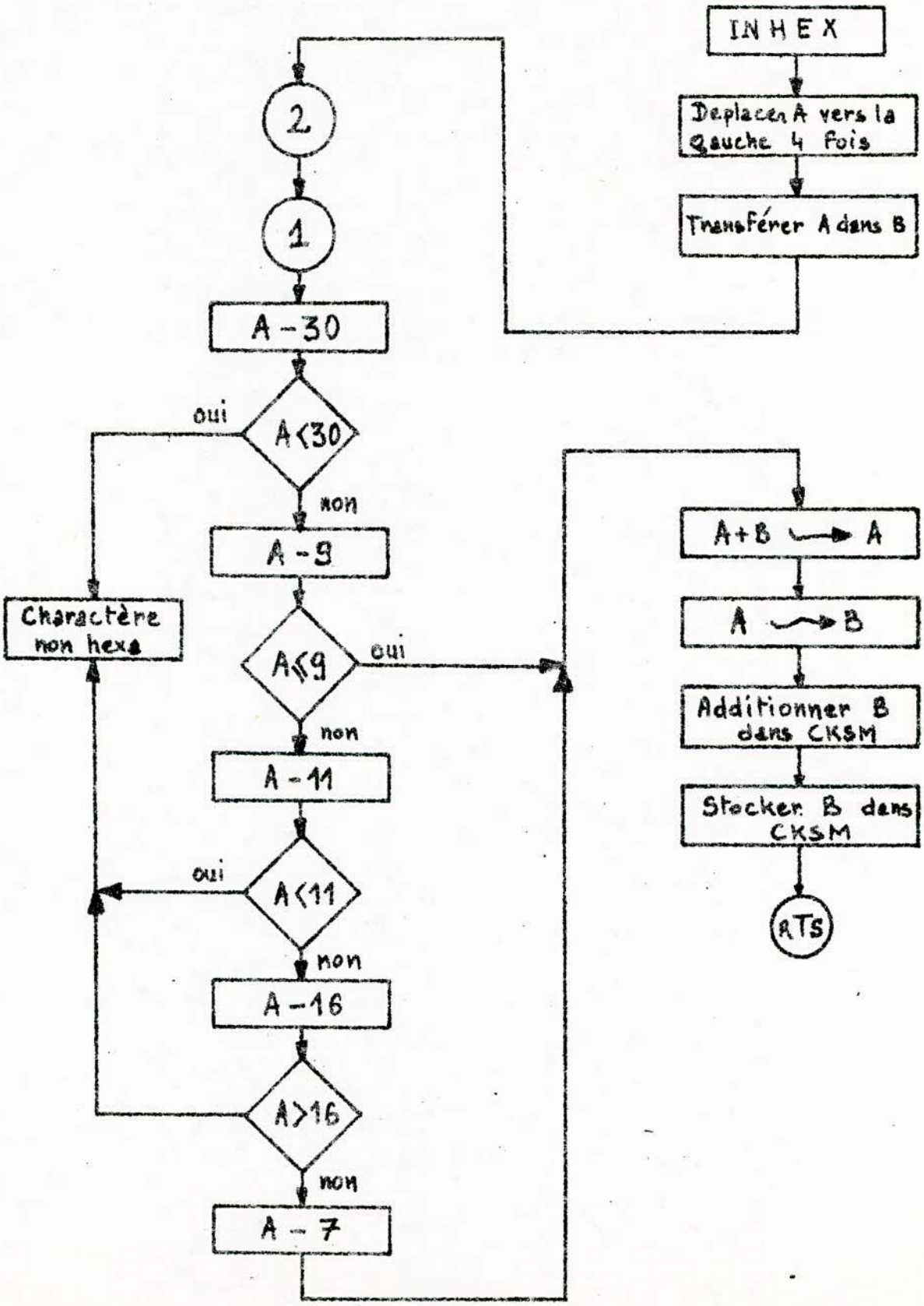
A = 16	00010110
SUBA 7	− 00000111
F	00001111

Pour les chiffres l'instruction SUBA 30 réalise le décodage.

Exemple : Soit N à décoder N = 38
L'instruction SUBA 30 donne :

$$(N)_{16} = N-30 = 8$$

V / Sous programme : Entrée d'un Byte (2 Frames) . BYTE .



V. SOUS PROGRAMME BYTE : "Entrée d'un byte (2 Frames)

1- Programme

```
0237   8D 10      BYTE   BSR   INHEX
0239   48                ASLA
023A   48                ASLA
023B   48                ASLA
023C   48                ASLA
023D   16                TAB
023E   8D 09             BSR   INHEX
0240   1B                ABA
0241   16                TAB
0242   FB 028B           ADDB  CKSM
0245   F7 028B           STAB  CKSM
0248   39                RTS
```

2- Commentaire

La subroutine BYTE assemble 2 caractères hexadécimaux pour constituer un byte. Le premier caractère hexadécimal entré correspond au plus fort poids ; il est transféré dans l'accumulateur A où il subit 4 décalages vers la gauche, ensuite il est transféré dans l'accumulateur B.
Le second caractère hexadécimal est introduit dans l'accumulateur A.
L'addition du contenu des 2 accumulateurs permet d'obtenir le byte constitué par les 2 caractères introduits. Le byte est par la suite additionné au checksum ; le résultat étant stocké dans le checksum.

Exemple : On veut constituer le byte 9C à partir des 2 caractères 9 et C.

i/ Introduction du premier caractère 9 = $(1001)_2$

Le caractère 9 subit 4 décalages vers la gauche dans l'acca : soit 1001 0000 qui est stocké dans l'ACCB.

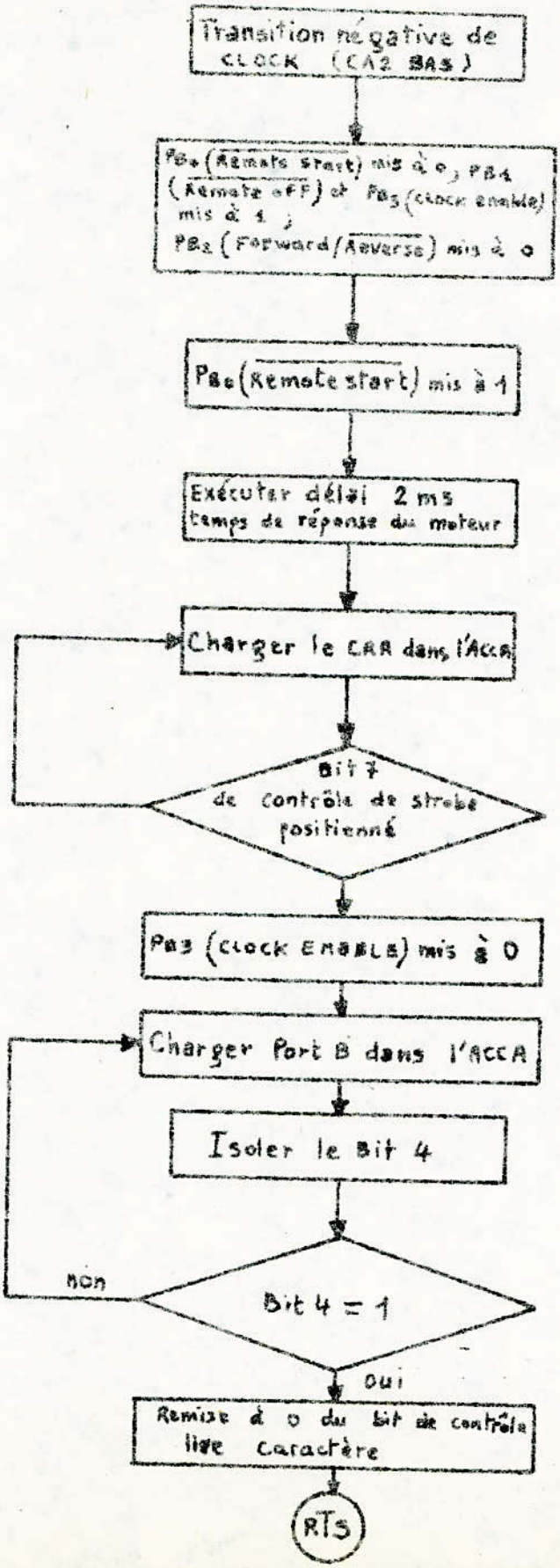
ii/ Introduction du second caractère C = $(1100)_2$ dans l'ACCA

iii/ Addition du contenu des 2 accumulateurs : le résultat qui se trouve dans l'ACCA est transféré dans l'ACCB

```
  10010000
  00001100
 ----------
  10011100 = 9C
```

On a donc constitué le byte 9C.

III / Sous programme : Retour d'un caractère (RETCH)



VI. SOUS PROGRAMME RETCH : Retour d'un caractère

1- Programme

```
020C   86 32      RETCH   LDAA 32
020E   B7 800A            STAA 800A
0211   86 0A              LDAA 0A
0213   B7 8009            STAA 8009
0216   86 1B              LDAA 1B
0218   B7 8009            STAA 8009
021B   CE FA      DEMO    LDX  FA
021D   09         LOP1    DEX
021E   26 FD              BNE  LOP1
0220   B6 800A    TST     LDAA 800A
0223   2A FB              BPL  TST
0225   86 13              LDAA 13
0227   B7 8009            STAA 8009
022A   B6 8009    LOP2    LDAA 8009
022D   84 10              ANDA 10
022F   27 F9              BEQ  LOP2
0231   B6 8008            LDAA 8008
0234   39                 RTS
```

2- Commentaire

Le programme général a été conçu de telle façon qu'une erreur Checksum d'une série de données n'est validée que si elle est détectée trois fois de suite. Donc une nouvelle des données s'avère nécessaire chaque fois que l'erreur Checksum est detectée :

Exemple : Soit une série de données lue par le lecteur et chargé dans le MPU.

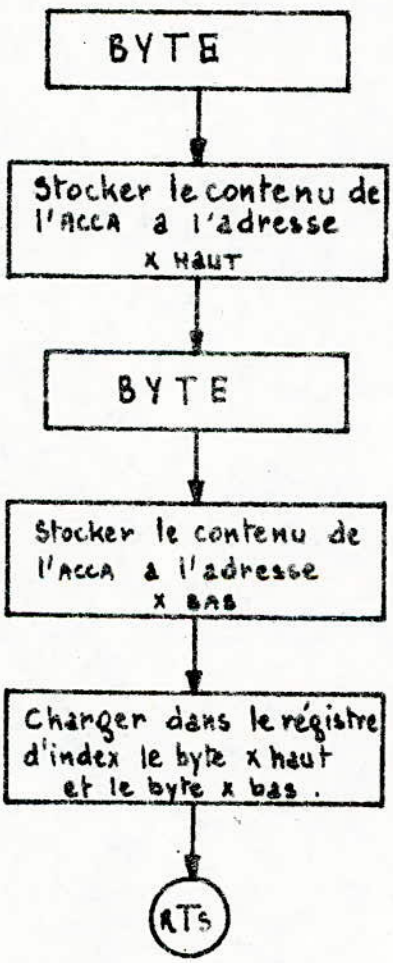
- S1 05 1010 05 39 9C

9C : Checksum S : début d'enregistrement

La sous broutine RETCH permet le retour de chaque caractère et sa lecture ; ceci dans le but de détecter le caractère S.

Quand le caractère S est positionné, une nouvelle lecture de la série de données est effectuée par le lecteur.
La gestion des signaux du lecteur par la subroutine RETCH est la même que dans la subroutine AVAC (avance d'un caractère) : la seule variante est le signal Forward/ Reverse qui doit être positionné bas de façon à inverser le sens de rotation du moteur.

VII / Sous programme : Construction d'adresse : CADDR .



VII. SOUS PROGRAMME CADDR : "Construction d'adresse"

1- Programme

```
01DC   8D 59          CADDR   BSR  BYTE
01DE   B7 0289                STAA XHAUT
01E1   8D 54                  BSR  BYTE
01E3   B7 028E                STAA X BAS
01E6   FE 028D                LDX  X HAUT
01E9   39                     RTS
```

2- Commentaire

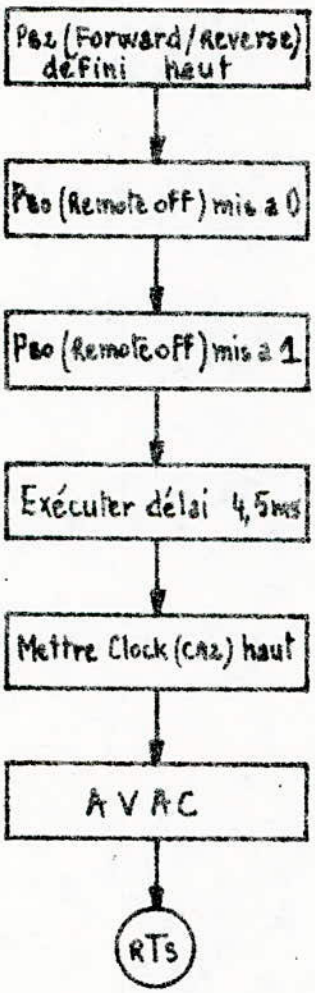
Une adresse est constituée de 2 bytes : pour la constituer on utilisera 2 fois la subroutine BYTE.
Un premier appel à la subroutine BYTE permet de réceptionner le premier byte d'adresse qui est stocké dans la zone mémoire à l'adresse X HAUT. Un second appel au sous programme BYTE permet la réception du second BYTE d'adresse stocké à l'adresse X BAS.

L'instruction LDX permet de charger les contenus des deux locations mémoire selectionnées (X HAUT, X BAS) dans le registre d'index.

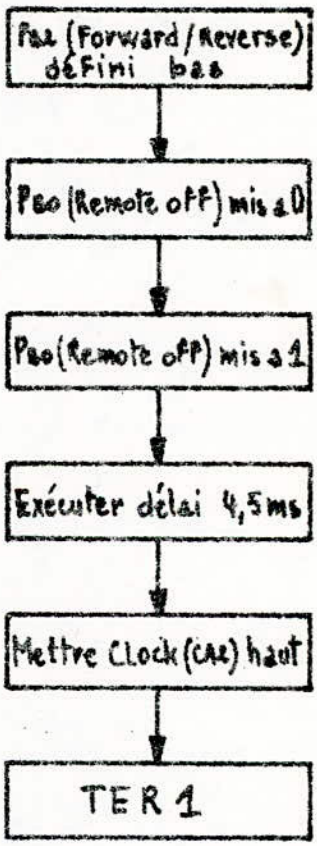
En effet le contenu du byte de la mémoire selectionné est chargé dans le byte le plus signifiant (high byte) du registre d'index. Le contenu du byte de la mémoire qui suit immédiatement le byte de la mémoire selectionnée est chargée dans le low byte (le moins signifiant) du registre d'index.

VIII / Sous Programmes : FINI 1 et FINI 2

FINI 1



FINI 2



VIII. SOUS PROGRAMME FINI 1, FINI 2 et TOP

1- Programmes

```
0172   B6 8009     FINI 1   LDAA 8009
0175   84 FD                ANDA FD
0177   B7 8009              STAA 8009
017A   8D 03                BSR  TOP
017C   8D 35                BSR  AVAC
017E   39                   RTS

01A2   B6 8009     FINI 2   LDAA 8009
01A5   84 F9                ANDA F9
01A7   B7 8009              STAA 8009
01AA   8D D3                BSR  TOP
01AC   8D AE                BSR  TER1

017F   B6 8009     TOP      LDAA 8009
0182   8A 02                ORAA 02
0184   B7 8009              STAA 8009
0187   CE 562      DEL      LDX  562
018A   09           LAS     DEX
018B   26 FD                BNE  LAS
018D   86 3A                LDAA 3A
018F   B7 800A              STAA 800A
0192   39                   RTS
```

2- Commentaire

Un nouveau cycle lecture ne peut être amorcé que si le signal Clock est positionné haut ceci par la mise du signal de contrôle CA2 à 1. Nous devons tenir compte du fait que

la fréquence du signal Clock ne doit pas dépasser 150 Hertz (T est supérieur ou égal à 1,7 ms).

Les sous-programmes FINI 1 ($F/\overline{R} = 1$) et FINI 2 ($F/\overline{R} = 0$) auront, après exécution, les conséquences suivantes :

- Remote Off momentanément bas positionne le signal Reader Ready à 0 donc lecture non validée

- Un délai de 4,5 ms est généré par le MPU ceci afin de repositionner un nouveau cycle lecture.

- CLOCK est positionné haut.

Les sous programmes FINI 1 et FINI 2 comportent pratiquemment les mêmes instructions d'où l'utilisation du sous programme TOP.

La subroutine FINI 1 fait appel au sous programme AVAC (avance d'un caractère) tandis que la subroutine FINI 2 fait appel au sous-programme RETCH (retour d'un caractère).

IX. SOUS PROGRAMME CPR3 : "Remise du compteur à 3"

1- Programme

```
01ED   B6 028F      CPR   LDAA CMPT
01F0   81 03              CMPA 03
01F2   26 03              BNE  CPR3
01F4   8D F4              BSR  FINI 3
01F6   8D 9B              BSR  LECTA 1

01F8   4F           CPR3  CLRA
01F9   7F 028F            CLR  CMPT
01FC   86 03              LDAA 03
01FE   B7 028F            STAA CMPT
0201   8D E7              BSR  FINI 3
0203   8D 8F              BSR  LECTA 1
```

2- Commentaire

Une erreur Checksum ne sera validé que si elle est détectée 3 fois de suite : d'où l'utilisation d'un compteur CMPT initialisé à 3.
Imaginons qu'après une première lecture d'une série de données une erreur Checksum est détectée : le compteur CMPT est alors décrémenté (CMPT = 2) et une nouvelle lecture de la même série est ordonnée par le MPU au lecteur. Si la même erreur est toujours détectée après deux autres tests Checksum (CMPT = 0) un message erreur est émis ; mais si après une deuxième lecture l'erreur n'est plus détectée le compteur doit alors être remis à 3 pour la suite des opérations : c'est le but de la subroutine CPR3.

X. SOUS PROGRAMMES END, NON HEXA, ERREUR:"Messages"

1- Programme

1.1- Message END :

```
025E   CE 0267      END    LDX BETA
0261   BD FA14             JSR FA14
0264   BD E000             JSR E000
0267   45           BETA   E
0268   4D                  N
0269   44                  D
026A   04                  'EOT'
```

1.2- Message Erreur :

```
026B   CE 0274      ERREUR LDX ALPHA
026E   BD FA14             JSR FA14
0271   BD E000             JSR E000
0274   45           ALPHA  E
0275   52                  R
0276   52                  R
0277   45                  E
0278   55                  U
0279   52                  R
027A   04                  'EOT'
```

1.3- Message NON HEXA

```
027B   CE 0282    NON HEXA LDX LANDA
027E   BD FA14             JSR FA14
0281   39                  RTS
0282   4E                  N
0283   4F                  O
```

0284	4E	N
0285	20	SP
0286	48	H
0287	45	E
0288	58	X
0289	41	A
028A	04	'EOT'

1.2- Commentaire

Les messages sont codés en caractères ASCII ; Ils sont chargés dans la zone mémoire sous forme de table. La subroutine localisée à l'adresse FA14 du programme EXBUG, enregistré sur mémoire ROM, affiche après décodage le message contenu dans la table dont l'adresse initiale est chargée dans le registre d'index. L'instruction 'EOT' indique la fin de la table.

Dans notre programme de gestion du lecteur, le message END est émis quand après avoir détecté le début d'enregistrement S, le MPU reçoit le caractère 9 indiquant la fin d'enregistrement. L'exécution du programme doit donc être arrêtée : l'instruction JSR E000 indique le retour au programme EXBUG initié à l'adresse E000.

Il en est de même pour le message ERREUR, émis quand une erreur Checksum est validée.
Le message NON HEXA est émis quand un caractère n'est pas hexadécimal.

XI. PROGRAMME DE GESTION DU LECTEUR

1- Organigramme

2- Programme

```
0100   7F 800A      INIT      CLR  800A
0103   7F 800B                CLR  800B
0106   7F 8008                CLR  8008
0109   86 0F                  LDAA 0F
010B   B7 8009                STAA 8009
010E   86 3E                  LDAA 3E
0110   B7 800A                STAA 800A
0113   86 04                  LDAA 04
0115   B7 800B                STAA 800B
0118   86 03      INIT CMPT   LDAA 03
011A   B7 028F                STAA CMPT

011D   8D 4C      START       BSR  AVAC 1
011F   81 53      LECTA       CMPA S
0121   26 4B                  BNE  ONE
0123   8D 4D                  BSR  FINI 1
0125   81 39                  CMPA 09
0127   27 75                  BEQ  END 2
0129   81 31                  CMPA 1
012B   26 41                  BNE  ONE
012D   7F 028B                CLR  CKSM
0130   8D 67                  BSR  BYTE 2
0132   80 02                  SUBA 02
0134   B7 028C                STAA BYTECT
0137   8D 5D                  BSR  CADDR1

0139   8D 5E      LOAD 1      BSR  BYTE 2
013B   7A 028C                DEC  BYTECT
013E   27 05                  BEQ  LOAD 2
0140   A7 00                  STAA X
0142   08                     INX
0143   20 F4                  BRA  LOAD 1

0145   7C 028B      LOAD 2    INC  CKSM
0148   27 52                  BEQ  CPR1
014A   7A 028F                DEC  CMPT
014D   27 51                  BEQ  ERREUR 3

014F   B6 8009                LDAA 8009
0152   84 FD                  ANDA FD
0154   B7 8009                STAA 8009
0157   8D 26                  BSR  TOP
0159   8D 55        TER 1     BSR  RETCH 1
015B   81 53                  CMPA S
```

```
015D   26 43          BNE  FINI 2
015F   B6 8009        LDAA 8009
0162   84 F9          ANDA F9
0164   B7 8009        STAA 8009
0167   8D 43          BSR  TOP
0169   8D B2          BSR  START
```

3- Commentaire

Le MPU, contrôlant la logique de commande du lecteur, reçoit les caractères enregistrés sur la bande perforée par les lignes PA programmées en entrée. La lecture se fait dans l'ACCA VIA le registre de données du PIA (côté A).

Quand le caractère S (début d'enregistrement sur la bande) est détecté, le prochain caractère qui suit est soit 9 soit 1.

- 9 (S9) : Fin d'enregistrement ; le MPU émet un message END

- 1 (S1) : Ce caractère indique l'enregistrement des bytes de données.

Cas général où S1 est détecté :

Le MPU initialise son Checksum qui est mis à 0. En effet lors de la réception des bytes de données, le MPU fait leur somme sur 8 bits : il calculera donc son propre checksum.

La subroutine BYTE introduit le premier byte indiquant le nombre d'octets en hexadécimal de la chaine de caractères qui suit S1.

Le MPU retranche 2 à ce nombre : le résultat est stocké dans le compteur de Bytes (BYTECT). BYTECT contient le nombre d'octets d'une série de données avec Checksum. Les 2 bytes d'adresse ne sont pas comptés.

L'appel de la subroutine CADDR1 permet de charger l'adresse du premier byte de données dans le registre d'index. Les opérations suivantes, du MPU, consistent à réceptionner les bytes de données.

Pour chaque octet stocké par le MPU :

- le compteur de bytes est décrémenté
- son contenu est comparé à 0

i/ BYTECT ≠ 0

- le byte est stocké à son adresse
- le registre d'index est incrementé ; l'adresse positionnée sera celle du prochain byte à stocker

Une boucle de programme permet au MPU de stocker tous les octets de données.

ii/ BYTECT = 0

Tous les bytes de données sont stockés à leurs adresses respectives. Le Ckecksum est incrementé ; cette opération amènera le MPU a détecter une erreur checksum si elle existe. Le test chexksum consiste à comparer son contenu à 0.

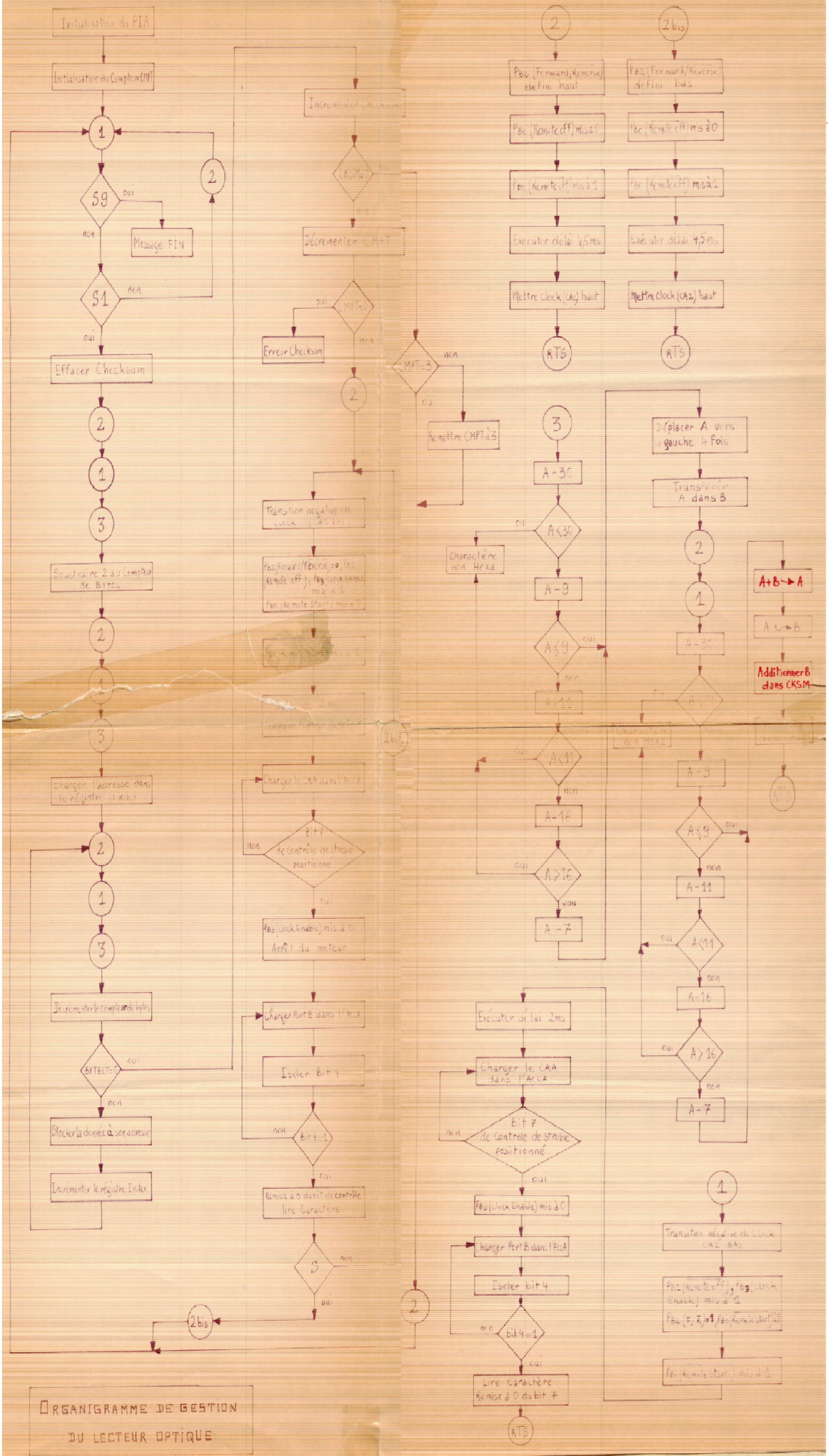
i/ CKSM ≠ 0

L'erreur existe ; Le compteur CMPT est decrémenté. Si son contenu n'est pas nul, le MPU fera revenir les caractères

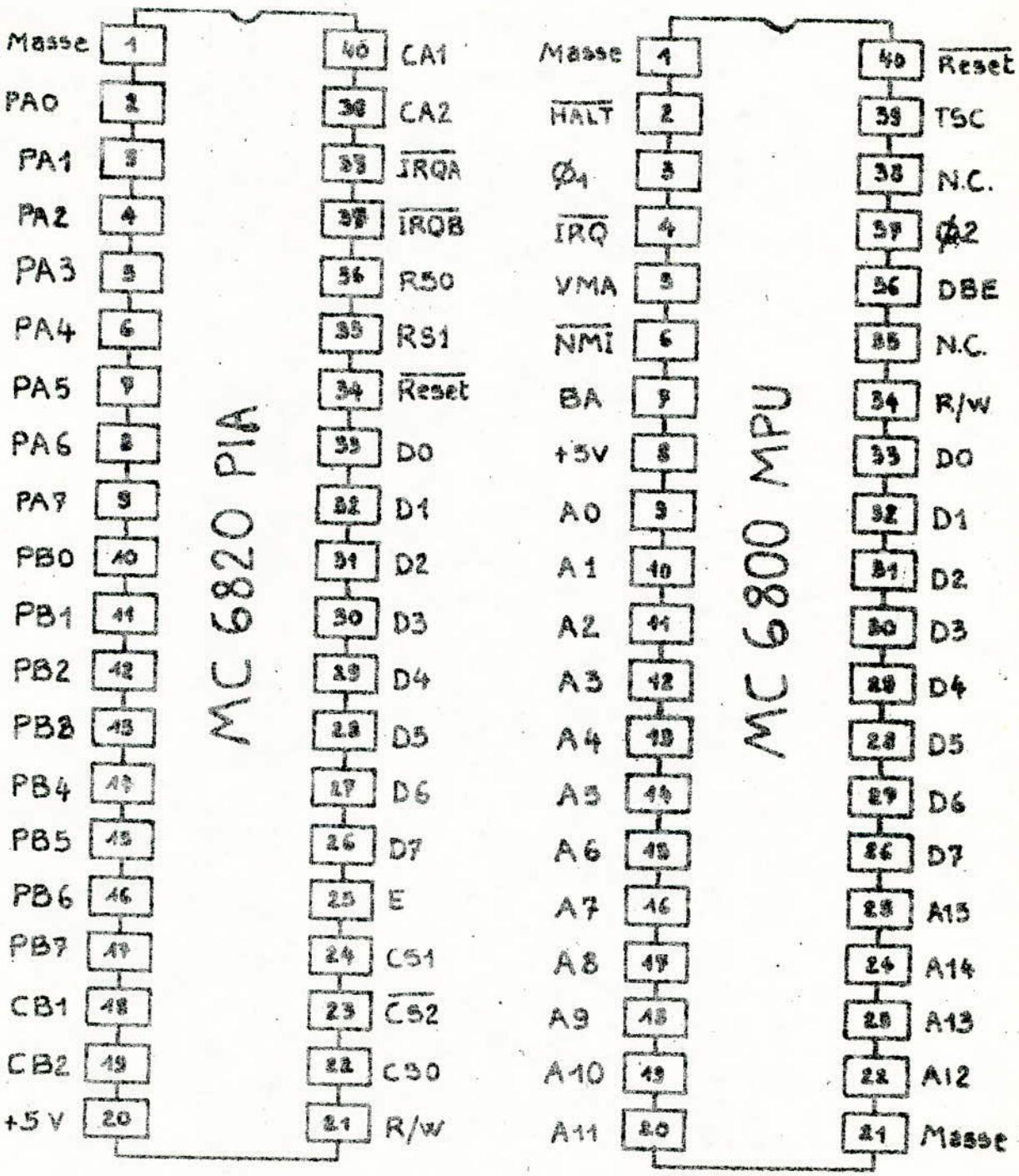
déjà lus de la bande jusqu'à la détection de leur début d'enregistrement S. Le retour de chaque caractère se fait par la subroutine RETCH.
Le MPU peut donc ordonner, au lecteur, une nouvelle lecture des données. Un nouveau test Checksum est effectué par le MPU. Le message ERREUR est émis quand la même erreur Checksum est détectée 3 fois de suite : dans ce cas le contenu de CMPT est nul.

ii/ CKSM = 0

Dans ce cas il n'y a pas d'erreur checksum. Cependant le MPU teste le compteur CMPT qui doit toujours être dans ce cas égal à 3. La subroutine CPR1 remet le compteur à 3.



ORGANIGRAMME DE GESTION DU LECTEUR OPTIQUE



BROCHAGE DU MPU ET DU PIA

Chapitre VI

II- ESSAIS

Les essais sont faits en utilisant un EXORCISER et un module optionnel I/O (entrée, sortie) qui permet l'interface avec le lecteur. Ce module contient 2 PIA dont un seul sera utilisé.

L'exorciser est un système qui permet à l'utilisateur d'un M 6800 de développer ses programmes.
Le software contenue dans l'Exorciser nous donne un système fonctionnel qui permet d'éditer, assembler et modifier les programmes en temps réel.

L'exorciser de base comprend un module MPU et un module d'évaluation (DEBUG).
Le module MPU comprend le MC 6800 MPU et le système CLOCK.
Le module DEBUG fournit à l'exorciser la capacité d'évaluer le programme utilisateur. Il contient 3 ROM dans lesquelles se trouve le programme EXBUG et 2 RAM.

Le programme de gestion du lecteur est chargé à partir d'un clavier dans une RAM du module DEBUG.
Pour son exécution, on affiche l'adresse initiale 0100 et on appuie sur la touche G du clavier.

Enfin nous donnons le listing du programme de gestion du lecteur enregistré dans une PAM du module DEBUG.

```
0100   7F 800A      INIT    CLR  800A  Initialisation du P.I.A.
0103   7F 800B              CLR  800B
0106   7F 8008              CLR  8008
0109   86 0F                LDAA 0F
010B   B7 8009              STAA 8009
010E   86 3E                LDAA 3E
0110   B7 800A              STAA 800A
0113   86 04                LDAA 04
0115   B7 800B              STAA 800B
0118   86 03      INII CMPT LDAA 03    Initialisation du compteur
011A   B7 028F              STAA CMPT

011D   8D 4C        START   BSR  AVAC1 Avancer un caractère
011F   81 53        LECTA   CMPA S
0121   26 4B                BNE  ONE   PRE.CARACT.NON (S)
0123   8D 4D                BSR  FINI1
0125   81 39                CMPA 09
0127   27 75                BEQ  END2  FIN
0129   81 31                CMPA 01
012B   26 41                BNE  ONE   2ND. CARACT. NON (1)
012D   7F 028B              CLR  CKSM  Zéro Checksum
0130   8D 67                BSR  BYTE2 Lire 1 Byte
0132   80 02                SUBA 02
0134   B7 028C              STAA BYTECT Compteur de Bytes
0137   8D 5D                BSR  CADDR1 Construction d'adresses

0139   8D 5E        LOAD1   BSR  BYTE2
013B   7A 028C              DEC  BYTECT
013E   27 05                BEQ  LOAD2 Zéro Compteur de Byte
0140   A7 00                STAA X     Stocker Donnée
0142   08                   INX
0143   20 F4                BRA  LOAD1

0145   7C 028B      LOAD2   INC  CKSM
0148   27 52                BEQ  CPR1  Checksum nul
014A   7A 028F              DEC  CMPT
014D   27 51                BEQ  ERREUR3 Erreur Checksum

014F   B6 8009              LDAA 8009
0152   84 FD                ANDA FD
0154   B7 8009              STAA 8009
0157   8D 26                BSR  TOP
0159   8D 55        TER1    BSR  RETCH1  Retour Caractère
015B   81 53                CMPA S
015D   26 43                BNE  FINI2
015F   B6 8009              LDAA 8009
0162   84 F9                ANDA F9
0164   B7 8009              STAA 8009
0167   8D 43                BSR  TOP
0169   8D B2                BSR  START
```

```
016B  8D 46     AVAC1   BSR   AVAC     Avance d'un caractère
016D  39                RTS

016E  8D 02     ONE     BSR   FINI1
0170  8D AD             BSR   LECTA

0172  B6 8009   FINI1   LDAA  8009
0175  84 FD             ANDA  FD
0177  B7 8009           STAA  8009
017A  8D 03             BSR   TOP
017C  8D 35             BSR   AVAC
017E  39                RTS

017F  B6 8009   TOP     LDAA  8009
0182  8A 02             ORAA  02
0184  B7 8009           STAA  8009
0187  CE 562    DELAI   LDX   562      Délai 4,5 ms
018A  09         LAS     DEX
018B  26 FD             BNE   LAS
018D  86 3A             LDAA  3A
018F  B7 800A           STAA  800A
0192  39                RTS

0193  8D 8A     LECTA1  BSR   LECTA
0195  02                NOP

0196  8D 44     CADDR1  BSR   CADDR
0198  39                RTS

0199  8D 70     BYTE2   BSR   BYTE1
019B  39                RTS

019C  8D 4F     CPR1    BSR   CPR

019E  8D 67     END2    BSR   END1

01A0  8D 63     ERREUR3 BSR   ERREUR2

01A2  B6 8009   FINI2   LDAA  8009
01A5  84 F9             ANDA  F9
01A7  B7 8009           STAA  8009
01AA  8D D3             BSR   TOP
01AC  8D AB             BSR   TER1
01AE  02                NOP
01AF  02                NOP
```

```
01B0   8D 5A      RETCH1  BSR   RETCH    Retour d'un caractère
01B2   39                 RTS

01B3   86 32      AVAC    LDAA  32       Avance d'un caractère
01B5   B7 800A            STAA  800A
01B8   86 0E              LDAA  0E
01BA   B7 8009            STAA  8009
01BD   86 1F              LDAA  1F
01BF   B7 8009            STAA  8009
01C2   CE FA      DEMO    LDX   FA       Délai moteur 2 ms
01C4   09         LOP1    DEX
01C5   26 FD              BNE   LOP1
01C7   B6 800A    TST     LDAA  800A
01CA   2A FB              BPL   TST
01CC   86 17              LDAA  17
01CE   B7 8009            STAA  8009
01D1   B6 8009    LOP2    LDAA  8009
01D4   84 10              ANDA  10
01D6   27 F9              BEQ   LOP2
01D8   B6 8008            LDAA  8008
01DB   39                 RTS

01DC   8D 59      CADDR   BSR   BYTE     Lire 2 Frames
01DE   B7 028D            STAA  XHAUT
01E1   8D 54              BSR   BYTE
01E3   B7 028E            STAA  X BAS
01E6   FE 028D            LDX   X HAUT   Adresse construite
01E9   39                 RTS

01EA   8D 86      FINI3   BSR   FINI1
01EC   39                 RTS

01ED   B6 028F    CPR     LDAA  CMPT
01F0   81 03              CMPA  03
01F2   26 03              BNE   CPR3
01F4   8D F4              BSR   FINI3
01F6   8D 9B              BSR   LECTA1

01F8   4F         CPR3    CLRA
01F9   7F 028F            CLR   CMPT
01FC   86 03              LDAA  03
01FE   B7 028F            STAA  CMPT     Remise du compteur à 3
0201   8D E7              BSR   FINI3
0203   8D 8E              BSR   LECTA1

0205   8D 2E      ERREUR2 BSR   ERREUR1

0207   8D 55      END1    BSR   END
```

```
0209   8D 2C      BYTE1   BSR   BYTE     Entrée d'un Byte (2 frames)
020B   39                 RTS
020C   86 32      RETCH   LDAA  32       Retour d'un caractère
020E   B7 800A             STAA  800A
0211   86 0A               LDAA  0A
0213   B7 8009             STAA  8009
0216   86 1B               LDAA  1B
0218   B7 8009             STAA  8009
021B   CE FA      DEMO    LDX   FA
021D   09         LOP1    DEX
021E   26 FD               BNE   LOP1
0220   B6 800A    TST     LDAA  800A
0223   2A FB               BPL   TST
0225   86 13               LDAA  13
0227   B7 8009             STAA  8009
022A   B6 8009    LOP2    LDAA  8009
022D   84 10               ANDA  10
022F   27 F9               BEQ   LOP2
0231   B6 8008             LDAA  8008
0234   39                 RTS

0235   8D 34      ERREUR1 PSR   ERREUR

0237   8D 10      BYTE    BSR   INHEX
0239   48                 ASLA
023A   48                 ASLA
023B   48                 ASLA
023C   48                 ASLA
023D   16                 TAB
023E   8D 09      BSR     INHEX
0240   1B                 ABA
0241   16                 TAB
0242   FB 028B            ADDB  CKSM
0245   F7 028B            STAB  CKSM
0248   39                 RTS

0249   8D 9F      INHEX   BSR   FINI3    Entrée d'un caractère Hexa
024B   80 30               SUBA  30
024D   2B 2C               BMI   NON-HEXA
024F   81 09               CMPA  09
0251   2F 0A               BLE   TETA
0253   81 11               CMPA  11
0255   2B 24               BMI   NON-HEXA
0257   81 16               CMPA  16
0259   2E 20               BGT   NON-HEXA
025B   80 07               SUBA  7
025D   39         TETA    RTS
```

```
025E   CE 0267    END      LDX   BETA
0261   BD FA14             JSR   FA14   Afficher message
0264   BD E000             JSR   E000   Exbug

0267   45         BETA     E             Message END
0268   4D                  N
0269   44                  D
026A   04                 'EOT'

026B   CE 0274    ERREUR   LDX   ALPHA
026E   BD FA14             JSR   FA14   Afficher message
0271   BD E000             JSR   E000   Exbug

0274   45         ALPHA    E             Message Erreur
0275   52                  R
0276   52                  R
0277   45                  E
0278   55                  U
0279   52                  R
027A   04                 'EOT'

027B   CE 0282    NON-HEXALDX   LANDA
027E   BD FA14             JSR   FA14   Afficher message
0281   39                  RTS

0282   4E                  N             Message NON-HEXA
0283   4F                  O
0284   4E                  N
0285   20                  SP
0286   48                  H
0287   45                  E
0288   58                  X
0289   41                  A
028A   04                 'EOT'

028B   0001                RMB 1 CHECKSUM
028C   0001                RMB 1 BYTECT (Compteur de Bytes)
028D   0001                RMB 1 X HAUT
028E   0001                RMB 1 X BAS
028F   0001                RMB 1 CMPT (Compteur)
```

CONCLUSION

Le Système M 6800 avec ses possibilités de programmation, interruption, DMA et les circuits d'interface des périphériques trouve de nombreuses applications dans beaucoup de domaines techniques.
Le système du M 6800 avec sa capacité et souplesse d'interface est très indiqué pour le contrôle de processus dans les systèmes industriels.
En fait les applications des microprocesseurs sont innombrales. Outre le grand public, elles couvrent : l'instrumentation et le test, les télécommunications, l'informatique, le médical... Ce vaste éventail explique l'importance des microprocesseurs. D'après les prévisions actuelles, on peut retenir les quelques points suivants dans l'évolution des microprocesseurs pour les prochaines années :

- Amélioration de l'intégration : apparition de nouveaux microprocesseurs 16 bits et de microprocesseurs intégrant des mémoires et des interfaces.

- Amélioration des performances actuelles, notamment de la vitesse.

- Décroissance rapide du coût des microprocesseurs.

APPENDIX A
MPU INSTRUCTIONS

Accumulator and Memory Instructions

OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	5 H	4 I	3 N	2 Z	1 V	0 C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	6	3				00 → M	•	•	R	S	R	R
	CLRA													4F	2	1	00 → A	•	•	R	S	R	R
	CLRB													5F	2	1	00 → B	•	•	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	5	2	B1	4	3				A − M	•	•	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	5	2	F1	4	3				B − M	•	•	↕	↕	↕	↕
Compare Acmltrs	CBA													11	2	1	A − B	•	•	↕	↕	↕	↕
Complement, 1's	COM							63	7	2	73	6	3				M → M	•	•	↕	↕	R	S
	COMA													43	2	1	A → A	•	•	↕	↕	R	S
	COMB													53	2	1	B → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add. of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	④	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	④	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	④	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	⑥	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑥	↕
	ROLB													59	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑥	↕
	RORA													46	2	1	A	•	•	↕	↕	⑥	↕
	RORB													56	2	1	B — C b7 b0	•	•	↕	↕	⑥	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑥	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑥	↕
	ASLB													58	2	1	B — 0 ← C b7 b0	•	•	↕	↕	⑥	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑥	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑥	↕
	ASRB													57	2	1	B — b7 b0 → C	•	•	↕	↕	⑥	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑥	↕
	LSRA													44	2	1	A	•	•	R	↕	⑥	↕
	LSRB													54	2	1	B — 0 → b7 b0 → C	•	•	R	↕	⑥	↕
Store Acmltr.	STAA				97	4	2	A7	6	2	B7	5	3				A → M	•	•	↕	↕	R	•
	STAB				D7	4	2	E7	6	2	F7	5	3				B → M	•	•	↕	↕	R	•
Subtract	SUBA	80	2	2	90	3	2	A0	5	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	5	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subtract Acmltrs.	SBA													10	2	1	A − B → A	•	•	↕	↕	↕	↕
Subtr. with Carry	SBCA	82	2	2	92	3	2	A2	5	2	B2	4	3				A − M − C → A	•	•	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	5	2	F2	4	3				B − M − C → B	•	•	↕	↕	↕	↕
Transfer Acmltrs	TAB													16	2	1	A → B	•	•	↕	↕	R	•
	TBA													17	2	1	B → A	•	•	↕	↕	R	•
Test, Zero or Minus	TST							6D	7	2	7D	6	3				M − 00	•	•	↕	↕	R	R
	TSTA													4D	2	1	A − 00	•	•	↕	↕	R	R
	TSTB													5D	2	1	B − 00	•	•	↕	↕	R	R
																		H	I	N	Z	V	C

LEGEND:

- OP Operation Code (Hexadecimal);
- ~ Number of MPU Cycles;
- # Number of Program Bytes;
- + Arithmetic Plus;
- − Arithmetic Minus;
- · Boolean AND;
- M_{SP} Contents of memory location pointed to be Stack Pointer;
- + Boolean Inclusive OR;
- ⊕ Boolean Exclusive OR;
- M Complement of M;
- → Transfer Into;
- 0 Bit = Zero;
- 00 Byte = Zero;

Note – Accumulator addressing mode instructions are included in the column for IMPLIED addressing

CONDITION CODE SYMBOLS:

- H Half-carry from bit 3;
- I Interrupt mask
- N Negative (sign bit)
- Z Zero (byte)
- V Overflow, 2's complement
- C Carry from bit 7
- R Reset Always
- S Set Always
- ↕ Test and set if true, cleared otherwise
- • Not Affected

Index Register and Stack Manipulation Instructions

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

Jump and Branch Instructions

OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BRANCH TEST	5 H	4 I	3 N	2 Z	1 V	0 C
Branch Always	BRA	20	4	2										None	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Carry Clear	BCC	24	4	2										C = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Carry Set	BCS	25	4	2										C = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If = Zero	BEQ	27	4	2										Z = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If ≥ Zero	BGE	2C	4	2										N ⊕ V = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If > Zero	BGT	2E	4	2										Z + (N ⊕ V) = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Higher	BHI	22	4	2										C + Z = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If ≤ Zero	BLE	2F	4	2										Z + (N ⊕ V) = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Lower Or Same	BLS	23	4	2										C + Z = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If < Zero	BLT	2D	4	2										N ⊕ V = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Minus	BMI	2B	4	2										N = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Not Equal Zero	BNE	26	4	2										Z = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Overflow Clear	BVC	28	4	2										V = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Overflow Set	BVS	29	4	2										V = 1	⇕	⇕	⇕	⇕	⇕	⇕
Branch If Plus	BPL	2A	4	2										N = 0	⇕	⇕	⇕	⇕	⇕	⇕
Branch To Subroutine	BSR	8D	8	2											⇕	⇕	⇕	⇕	⇕	⇕
Jump	JMP				6E	4	2	7E	3	3				See Special Operations	⇕	⇕	⇕	⇕	⇕	⇕
Jump To Subroutine	JSR				AD	8	2	BD	9	3					⇕	⇕	⇕	⇕	⇕	⇕
No Operation	NOP										02	2	1	Advances Prog. Cntr. Only	⇕	⇕	⇕	⇕	⇕	⇕
Return From Interrupt	RTI										3B	10	1		⑩					
Return From Subroutine	RTS										39	5	1		⇕	⇕	⇕	⇕	⇕	⇕
Software Interrupt	SWI										3F	12	1	See Special Operations	⇕	⇕	⇕	⇕	⇕	⇕
Wait for Interrupt	WAI										3E	9	1		⇕	⑪	⇕	⇕	⇕	⇕

Condition Code Register Manipulation Instructions

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

CONDITION CODE REGISTER NOTES:

(Bit set if test is true and cleared otherwise)

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

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

1- MOTOROLA M 6800 MICROPROCESSEUR PROGRAMMING MANUAL (1975)

2- MOTOROLA M 6800 MICROCOMPUTER SYSTEM DESIGN DATA (1976)

3- 6800 PROGRAMMING FOR LOGIC DESIGN BY ADAM OSBORNE SYBEX (1977)

4- MOTOROLA ENGINEERING NOTE 100 MIKBUG (1975)

5- AN INTRODUCTION TO MICROCOMPUTERS VOLUME II SOME REAL PRODUCTS BY ADAM OSBORNE SYBEX (1977)

6- MOTOROLA SYSTEMES REFERENCE AND DATA SHEETS M 6800 (1975)

7- THEORIE ET PRATIQUE DES MICROPROCESSEURS H. LILEN EDITIONS RADIO (1977)