UNIVERSITÉ DES SCIENCES
ET DE LA TECHNOLOGIE
USTHB

ÉCOLE NATIONALE
POLYTECHNIQUE

Département d'Électronique et d'Électrotechnique
Projet de Fin d'Étude

DIPLÔME D'INGÉNIORAT

ÉTUDE ET RÉALISATION D'UNE UNITÉ DE DIALOGUE - MICRO - MICRO -

Proposé par : H. TEDJINI Dr Ingénieur

Suivi par : A. BOURKEB
H. TEDJINI

Étudiée par :

ZIANE Kamel-Eddine

DAOUD Bachir

JANVIER 1981

UNIVERSITÉ DES SCIENCES
ET DE LA TECHNOLOGIE
USTHB

ÉCOLE NATIONALE
POLYTECHNIQUE

Département d'Électronique et d'Électrotechnique
Projet de Fin d'Étude

DIPLÔME D'INGÉNIORAT

ÉTUDE ET RÉALISATION D'UNE UNITÉ DE DIALOGUE - MICRO - MICRO -

Proposé par : H. TEDJINI Dr Ingénieur

Suivi par : A. BOURKEB
H. TEDJINI

Étudiée par :

ZIANE Kamel-Eddine

DAOUD Bachir

JANVIER 1981

~ Dédicaces ~

Pour :

- Mon très cher et dévoué PÈRE
- Ma douce et bien aimée Mère.
- Celui qui guida mes pas, mon frère El Hadj-Nourdine.
- Mes frères, sœurs, oncles, tantes, beaux frères, belles sœurs.
- Ma femme Rania.
- Tous mes parents, amis et amies.

Kamel Ziane

Pour :

* Mon Oncle
* Mes Frères et Soeur
* Mon Père
* Ma mère
* Mes amis

Daoud. Bachir

REMERCIEMENTS

Ce travail a été réalisé à la division V du C.S.T.N.
Nous remerçions Monsieur B.SANSAL pour l'acceuil chaleureux dans son Service "Contrôle et Simulation".

Nous remerçions également Monsieur H.TEDJINI, Docteur Ingénieur et chargé de cours à l'E.N.P.A. pour nous avoir proposé ce sujet, guidé et conseillé durant notre projet.

Nous nbublions pas d'en remercier et d'exprimer notre profonds gratitude à Monsieur BOURKEB.R , pour l'attention soutenue et les conseils éclairés qu'il n'a cessés de nous prodiguer tout au long de l'élaboration de ce projet.

Nos remerciements vont aussi à Monsieur OUIGUINI.R. et HALIMI.M, qui ont bien voulu s'interesser à notre étude et de nous apporter leur aide précieuse.

Notre vive reconnaissance va également à tous ceux dont la contribution efficace a permis la mise en forme finale de ce projet.

// O M M A I R E

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

La simulation a pour but la construction d'un modèle de travail mathématique ou physique présentant une similitude de propriétés ou de relations avec le système naturel ou technologique faisant l'objet de l'étude .

Cette simulation reproduit à vitèsse accélerée l'évolution temporelle du système en tenant généralement compte d'aléas pour donner sur le comportement de ce système des renseignements que d'autres méthodes ne fournissaient pas , si ce n'est à un prix de revient supérieur .

Les études de simulation et l'utilisation conjointe des ensembles électroniques se sont revelées un outil extrenement précieux dans la recherche scientifique .

La concention et la réalisation du projet " Simulation et Contrôle d'un Réacteur Nucléaire " au centre C.S.T.N. est basée sur le multiprocessing . Plusieurs unités centrales secondaires (MPU esclaves) travaillent en parallèle sous le contrôle d'un module principal (Maître) . Les modules secondaires seront chargés de la résolution des équations régissant un réacteur nucléaire , alors que le module principal coordonnera le dialogue entre les différents esclaves d'une part et les différents periphériques d'autre part .

Notre étude consiste a établir le dialogue entre l'organe principal (Maître) et les différents modules secondaires (Esclaves) pour pouvoir résoudre facilement de nombreux problèmes en gagnant en temp et en espace.

Le choix des processeurs " Maître " et " Esclave " a suivi l'évolution Technologique des composants électroniques et à conduit à l'utilisation des microprocesseurs " MC 6800 " qui permettent actuellement de remplacer avantageusement, tant sur le plan économique que sur le plan performance (rapidité d'exécution) la logique cablée dans le système de traitement .

.../...

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

La simulation a pour but la construction d'un modèle de travail mathématique ou physique présentant une similitude de propriétés ou de relations avec le système naturel ou technologique faisant l'objet de l'étude .

Cette simulation reproduit à vitèsse accélerée l'évolution temporelle du système en tenant généralement compte d'aléas pour donner sur le comportement de ce système des renseignements que d'autres méthodes ne fournissaient pas , si ce n'est à un prix de revient supérieur .

Les études de simulation et l'utilisation conjointe des ensembles électroniques se sont revelées un outil extrenement précieux dans la recherche scientifique .

La conception et la réalisation du projet " Simulation et Contrôle d'un Réacteur Nucléaire " au centre C.S.T.N. est basée sur le multiprocessing . Plusieurs unités centrales secondaires (MPU esclaves) travaillent en parallèle sous le contrôle d'un module principal (Maître) . Les modules secondaires seront chargés de la résolution des équations régissant un réacteur nucléaire , alors que le module principal coordonnera le dialogue entre les différents esclaves d'une part et les différents périphériques d'autre part .

Notre étude consiste a établir le dialogue entre l'organe principal (Maître) et les différents modules secondaires (Esclaves) pour pouvoir résoudre facilement de nombreux problèmes en gagnant en temp et en espace.

Le choix des processeurs " Maître " et " Esclave " a suivi l'évolution Technologique des composants électroniques et à conduit à l'utilisation des microprocesseurs " MC 6800 " qui permettent actuellement de remplacer avantageusement, tant sur le plan économique que sur le plan performance (rapidité d'exécution) la logique cablée dans le système de traitement .

.../...

Suite/

Le principe de fonctionnement dans la résolution d'un système d'équation différentielle à l'aide de l'unité de dialogue entre microprocesseurs est le suivant :

On donne à chacun des 16 esclaves (dans notre cas) une équation bien déterminée à resoudre . Chaque esclave ne peut disposer de toutes les données et tout les paramètres et sera donc obligé de demander à l'organe principal (Maître) de lui transmettre les résultats partiels des autres modules secondaires (autres esclaves) , d'où le principe de dialogue entre esclaves.

CHAPITRE I.

MICROPROCESSEUR MC 6800

I - 1 Introduction

1 - A Mémoires RAM

1 - B Mémoires ROM

1 - C Interfaces PIA

1 - D Interfaces ACIA.

I - 2 Etude de l'Unité centrale (MPU)

2 - A Description générale du MPU

2 - B Organisation interne

2 - C Bus de liaisons.

- 1 Data bus

- 2 Bus adress.

- 3 Bus contrôle

I - 3 Logiciel du MC 6800

3 - A Instructions

3 - B Modes d'adressage.

ETUDE DU 6800

I - INTRODUCTION :

Les microprocesseurs constituent une véritable révolution technologique qui exercera ses effets dans tous les domaines de l'activité humaine.

Le microprocesseur est fonda -mentalement un circuit intégré complexe commandé par un programme et remplissant les fonctions d'une unité centrale de traitement d'ordinateur. L'unité centrale (MPU) ne peut pas dialoguer seule, avec les périphériques pour cela on doit lui adjoindre des mémoires et des interfaces.

Mémoires RAM

Elles permettent l'enregistrement des données mot par mot et une mémorisation globale des mots inscrits. La lecture n'est pas destructrice de l'information le contenu peut être modifié à volonté. Le maintien des informations impose la permanence de sources d'alimentation. Une mémoire RAM comporte des bornes d'adresse du mot choisi d'entrée écriture, de sortie lecture, d'ordres de lire ou d'écrire,d'alimentation et de servitude.

Mémoire ROM

Elles reçoivent des informations une fois pour toutes et dont le contenu ne peut être modifié on rencontre aussi des ROM qui sont des ROM programmable par l'utilisateur, les EPROM ou les REPROM dont

.../...

le contenu peut être occassionnellement modifié. Contrairement à une mémoire PROM, la REPROM est globalement effaçable par rayon Ultra-Violet, et ensuite réinscriptible par l'utilisateur.

- Des interfaces :

* PIA : Adaptateur d'interface de périphérique servant au couplage avec le MPU pour la transmission en parallèle.

* ACIA : Adaptateur d'interface série un Octet parallèle de données sera converti en un octet servi et ensuite transmis.

I - 2. L'UNITE CENTRALE (MPU)

2 - A. Description Générale du MPU :

Le MC 6800 traite des mots de 8 bits et possède une capacité d'adressage de 64 K octets. Le MPU dispose de 72 instructions de longueurs variables et de 7 modes d'adressage. Il est compatible TTL ne nécessitant qu'une alimentation de 5^{v}, sa consommation variée autour de 0,25 w, il travaille à une fréquence de 1 MHZ à deux phases ϕ_1 et ϕ_2 fig(2).

2 - B Organisation Interne :

Le MPU contient 3 registres à 16 bits et trois à 8 bits. Ces registres servent de mémoires temporaires. (Fig.3)

2.- B.1 - Deux Accumulateurs A et B :

Ce sont des registres de 8 bits chacun, ils sont employés pour mémoriser les opérandes (donnés) et les résultats nécessaires à l'unité arithmétique et logique.

.../...

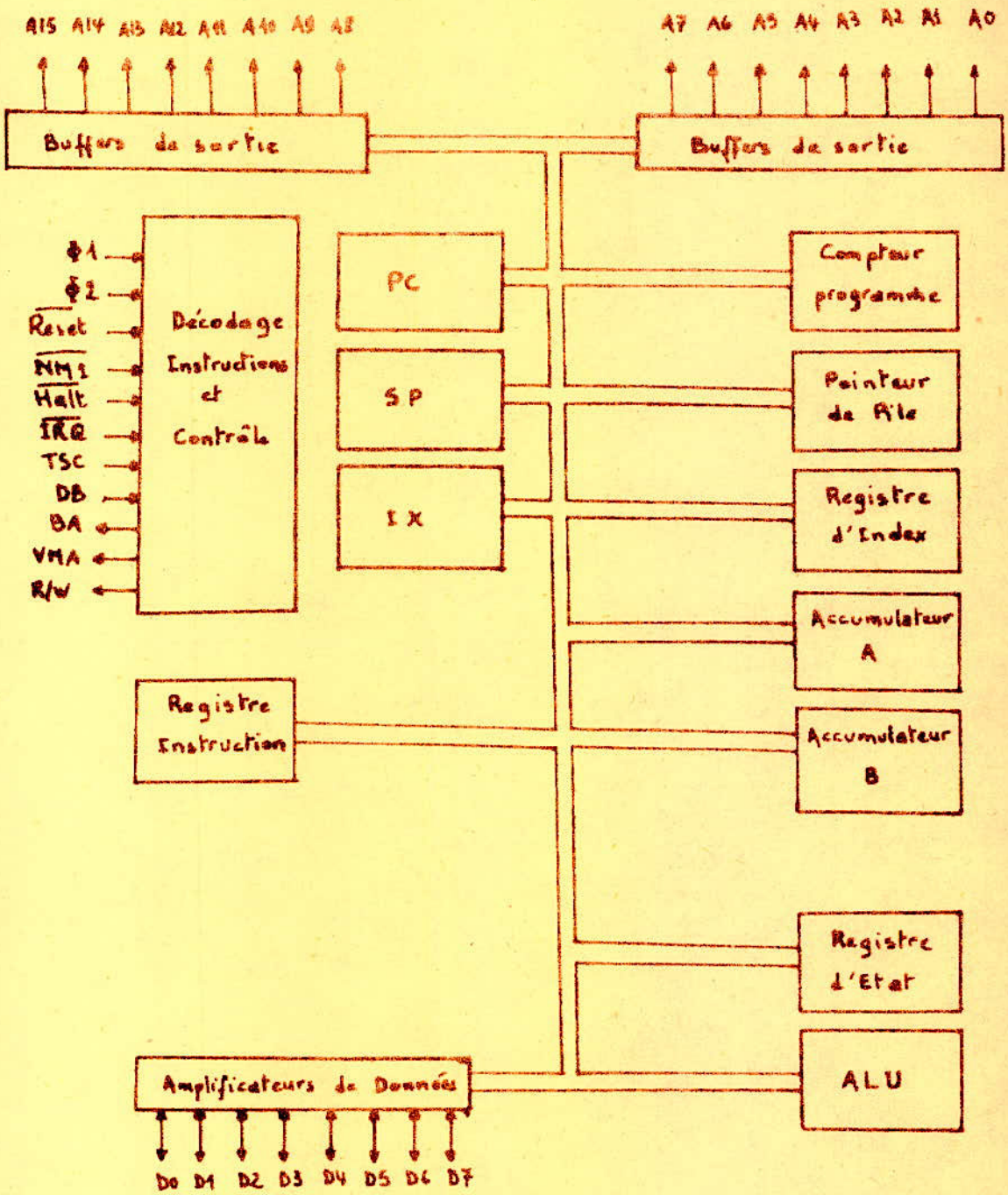


Fig 3 Organisation Générale

2. B - 2. Registre Index (IX)

C'est un registre de 16 bits dont le contenu peut servir d'index dans le mode d'adressage indexé mais aussi peut être utilisé pour le transfert des donnés.

2. B-3. Un pointeur de Pile (SP)

C'est aussi un registre de 16 bits, son rôle c'est de stocker l'adresse de l'emplacement mémoire disponible dans la pile externe.

2.B-4. Compteur Ordinal (PC)

C'est un registre à 16 bits, il est destiné à mémoriser l'adresse de la prochaine instruction à éxécuter lors du traitement du programme.

2.B-5. Registre d'état CCR :

C'est un registre à 8 bits. Il est lié directement à l'ALU (Unité arithmétique et logique) toutes les opérations effectuées dans l'ALU agissent sur les états du CCR. Les bits du CCR sont affectés par des instructions, des résultats on des interruptions, ils permettent de disposer de 6 informations utiles pour la gestion du programme. Cinq d'entres elles concernent les résultats d'une opération effectuée par l'ALU :

1	1	H	I_m	N	Z	V	C

- Overflow : V

mis à 1 lorsque le résultat déborde de la capacité de 8 bits complémenté à deux.

- Zéro : Z

L'indicateur de zéro signale que le résultat d'une opération est nul en se positionnant à 1. Si ce résultat n'est pas nul, il est à zéro.

- Négative N

Ce bit se positionne à 1 si le résultat de l'opération qui vient d'être exécuté a mené à un résultat négatif. Par conséquent si ce bit n'est pas à 1 c'est le résultat nul ou positif, mais en testant le bit Z, on lèvera le doute entre ces 2 cas.

- Interrupt Mask : Im

Ce bit est utilisé dans ce cas des demandes d'interruption Im = 1 L'interruption est prise en compte et interdit l'accès à tout autre interruption . Il est remis à zéro lorsque on autorise de nouveau les interruptions à se manifester.

- Carry C

Ce bit est mis à 1 s'il y a une retenue du bit 7 du résultat.

Half carry.H.

Ce bit est mis à 1 s'il y a une retenue qui passe du bit 3 au bit 4 à la suite d'une opération.

2 - C. Bus de liaisons :

Les lignes du MPU se divisent en 3 catégories (fig 1) :

- Bus de données
- Bus d'adresse
- Bus de contrôle.

1. Data Bus Do - D7 (8 bits)

Ce bus est bidirectionnel, il y a des buffers de sortie avec possibilité "Three Shate" (Trois états).

.../...

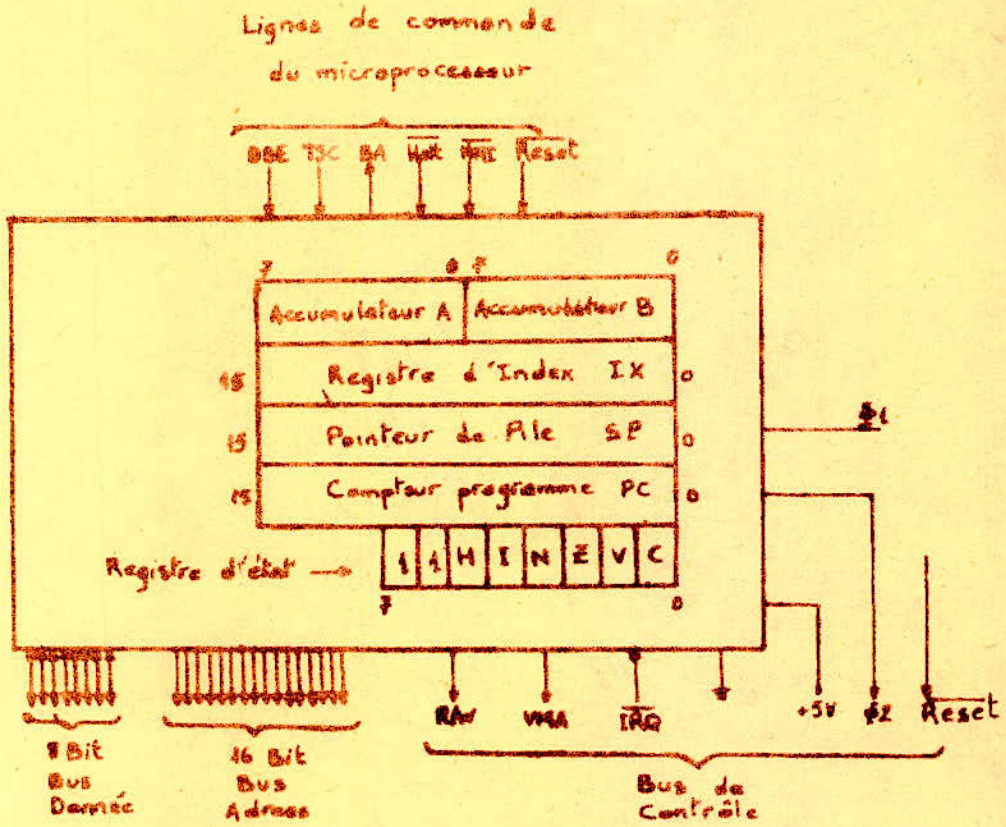


Fig 1 Lignes d'entrée/sortie du 6800

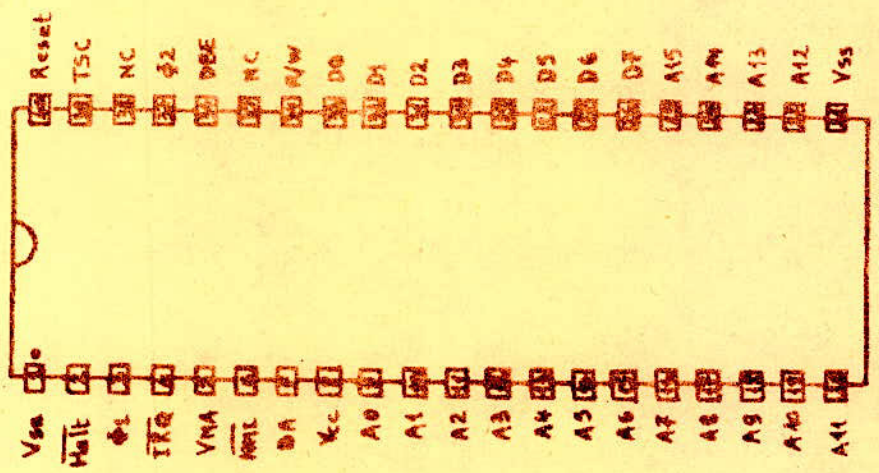


Fig 2 Brochage du 6800 vue de dessus

2 - Bus Adress Ao - A_{15} (8 bits)

Ce bus permet au MPU d'adresser 64K Octets mémoires ($K = 2^{10}$) = 1024 bits. Il est unidirectionnel. Lorsqu'il est en état haute impédance (état OFF) c'est un circuit ouvert ce qui permet de faire accès direct mémoire (DMA).

3 - Bus controle

a. Signaux de contrôle :

Ils permettent de synchroniser et de coordonner le fonctionnement du microprocesseur avec les mémoirs ou les périphériques qui sont généralement plus lents.

- 1. Read / Write

Cette ligne indique le sens de transfert des données si le MPU est dans état d'écriture ou de nature.

RIW = 1 c'est une opération de lecture

RIW = o " " " d'écriture.

- 2. Valid Memory Adress (VMA)

Ce signal indique aux périphériques qu'il y a une adresse disponible sur le bus adresse, le transfert de données ne sera fait que lorsque VMA = 1 -

- 3. Data bus Enable (DBE)

Il caractérise la validation du bus des donnés

DBE = 1 le bus de donnés et à l'état bas on haut (0 ou 1)

DBE = o " " " " " " " haute impédance (OFF)

Ce signal suit celui de la synchronisation $\phi 2$.

.../...

- 4. Bus Available

Si BA = 1 le bus d'adresse est disponible et le MPU arrêté

- 5. Three State control

C'est ce signal qui met à l'état haute impédance les lignes adresses et la ligne RIW.

- b. Signaux d'interruptions :

- 1. RESET :

Ce reset vise à initialiser le système à la mise sous tension. De ce fait il mettra en service un programme dit d'initialisation, qui fournira aux divers registres les informations utiles pour le démarrage. Il interviendra également chaque fois qu'on voudra revenir à la situation initiale en cours de service.

- 2. Halt.

Quant ce signal et à l'état bas le microprocesseur est arrêté à la fin de l'instruction en cours et le VMA et à zéro et BA passe à 1.

- 3. Interrupt Request: IRQ

C'est par cette ligne que la demande d'interruption sera prise en compte par le MPU. Tout d'abord le MPU termine l'instruction en cours, puis il teste le bit Im présent dans le registre des indicateurs d'états. Si celui et à 1 le MPU poursuit l'éxécution de son programme, sinon I_m = 0 le MPU range le contenu de ses registres (compteur ordinal, Ex, ACCA.. CCR) dans la pile de sauvegarde autre fraction de la mémoire RAM dont le niveau de remplissage est indiqué par un pointeur de pile SP après quoi le MPU établi le masque en portant I_m à 1, ce qui permettra éventuellement au programme de déterminer dans quel ordre les instructions seront traités si d'autres interruptions plus prioritaires sont autorisés à intervenir Im sera aussitôt remis à 0.

..../...

- 4. Non masquable Interrupt NME

Lorsque ce signal et à 0, la demande d'interruption non masquable et présente. Le niveau le plus prioritaire est attribué à l'entrée d'interruption non masquable. Par exemple une baisse de la tension d'alimentation annonçant sa coupure de traduira par une commande d'interruption de toute 1ère urgence.

C. Signaux de synchronisation :

L'adressage et le transfert de données sont synchronisés par ϕ_1 et ϕ_2 qui sont en opposition de phases.

ϕ_1 : prévu pour activer le MPU de fréquence comprise entre 1 et 100 khz

ϕ_2 : prévu pour activer les autres éléments du MPU.

I - 3 - L O G I C I E L D U M C 6 8 0 0 :

3.1. Instruction

Le 6800 possède un jeu de 72 instructions d'une longueur de 1 à 3 Octets fig.4. Elles peuvent être :

- Arithmétique binaire ou décimale
- Logique
- Décalage
- Chargement
- Rangement
- Branchements conditionnels et inconditionnels
- D'interruption
- De Pointage.

3. 2 - Modes d'adressages :

Les instructions et données (information) sont stockées dans des cellules mémoirs RAM ou ROM. Chaque cellule est caractérisée par son adresse. Pour atteindre une cellule précise, on dispose de plusieurs " modes d'adressage ".

a - Adressage immédiat :

L'opérande se trouve dans le 2° ou 3° octet de l'instruction suivant que l'on s'adresse aux registres de 8 ou 16 bits resp. ACCA ou registres IX Ou SP).

.../...

b - Adressage Direct :

L'adresse de l'opérande se trouve dans le 2° Octet de l'instruction. Ce mode ne peut atteindre les adresses supérieures à 255 -

Adressage étendu.

· Adressage de l'opérande est contenue dans le 2° (MSB) et le 3° (LSB) octets de l'instruction. Ce mode peut adresser les mémoirs comprises entres 256 et 65535.

d - Adressage Indexé :

Le contenu du 2° Octet est ajouté au contenu du registre index pour former l'adresse de l'opérande.

e - Adressage implicite :

Dans ce mode d'adressage l'opérande est indiqué par le code opération de l'instruction.

f - adressage relatif

Le contenu du 2° Octet est ajouté au contenu du compteur ordinal.

g - Adressage des accumulateurs :

La donnée est représentée par le contenu des accumulateurs ACCA ou ACCB.

CHAPITRE II

ETUDE DETAILLEE D'UN P I A
(le compteur d'entrée/sortie - M C 6820)

-=-=-=-•Oo-=-=-=-

II - 1 INTRODUCTION

II - 2 BROCHAGE DU P I A

II - 3 SIGNAUX ECHANGES
- 3 a - Avec le système
- 3 b - Avec la périphérie

II - 4 ORGANISATION INTERNE
- 4 a - Selection et adressage des registres

II - 5 PROGRAMMATION DU P I A
- 5 a - Ecriture des registres DDRA - DDRB
- 5 b - Le bornier A -
- 5 c - Le bornier B -
- 5 d - Ecriture des registres de contrôle
  - . 5 d - 1 Mode de fonctionnement de CA_1
  - 5 d - 2 Mode de fonctionnement de CA_2
  - 5 d - 3 Mode de fonctionnement de CB_2

II - 1 INTRODUCTION

Tous système quels qu'ils soient (Kit d'évaluation, système de mise au point de programmes ou micro-ordinateur) travaillent avec un coupleur d'entrée sortie (PIA).

L'importance particulière dans notre travail de ce composant nous a poussé à étudier de façon détaillée l'un des modèles les plus connus et utilisés le PC 6820 de MOTOROLA ou SF.F 96821 de SESCOSEM.

Compatibles TTL, ces circuits peuvent souvent migrer d'un système microprocesseur à un autre.

Du point de vue hardware ces cricuits LSI (large square intégration) sont très sophistiqués. Leur fonctionnement est programmable et il est nécessaire de maitriser à la perfection tous les bits de commance dont dépend leur mode de travail.

Conscient de l'aspect un peu ardu de leur utilisation, nous avons volontairement illustré notre rappel d'un grand nombre de schémas qui nous l'espérons vous permettrant de tirer le maximum de profits de ce rappel.

Notons aussi que ces coupleurs ont différentes appellations uivant les constructeurs.

- PIA (périphéral interface adapter) pour Motorola et Sescosem.
- PIO (programmable Imput/Outpout) pour Zilog.
- PPI (programmable périphéral Interface) pour Intel et Signétics.

Quant au rôle du coupleur d'entée/ sortie ou PIA c'est d'effectuer la liaison entre le monde extérieur et la machine.

Pour communiquer avec l'extérieur, la logique programmée doit recevoir des informations de l'extérieur, via des entrées, ou fournir des informations vers l'extérieur via des sorties.

II - 2 BROCHAGE DU P I A

Le circuit se présente sous forme d'un boîtier DIL (Dual In Line) à 40 broches voir Fig.1.

- La signification précise de chacune des broches du boîtier est explicitée de façon simplifiée ci-dessous.

II - 3 SIGNAUX ECHANGES

3 a - Avec le système (PIA ——— système)
voir Fig. 2

- CS_0 - CS_1 - $\overline{CS}_2$: permettent la selection du PIA. Quant CS_0 , CS_1 , $\overline{CS}_2$ = 110 le PIA est selectionné

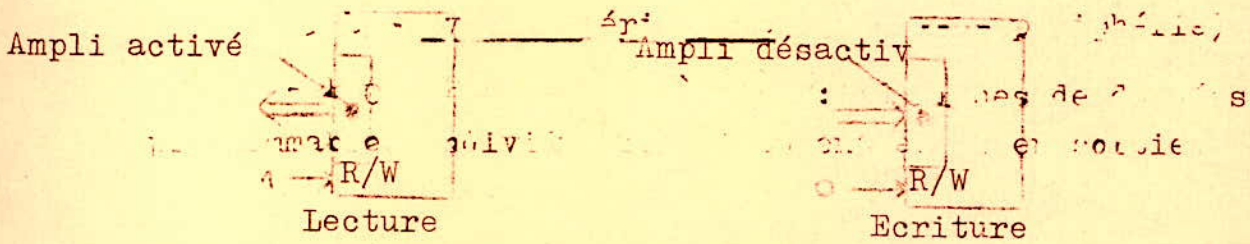
- RS_0 - RS_1: Le PIA étant selectionné, avec les 4 combinaisons de ces 2 bits on adresse les registres internes du PIA. voir Fig. 4

En conséquence, le PIA occupe 4 adresses mémoires.

- E : Signal d'activation des échanges, généralement cette entrée est reliée à $\emptyset_2$ (signal du bus contrôle)

-R/W : Signal de lecture - écriture
"1" = lecture
"0" = écriture

- D_0 à D_7: bus bidirectionnel de données. Il aboutit dans le PIA à un amplificateur qui peut être activé ou mis à l'état haute impétance par le signal R/W et ce si le PIA est selectionné.



et c'est par ces 8 fils qu'arrivent les données à transmettre en sortie (vers périphériques) ou à lire.

- $\overline{RESET}$: Mis à 0, ce signal remet tous les registres internes du PIA à 0.

- $\overline{IRQA}$ - $\overline{IRQB}$: Deux lignes de demandes d'interruption destinées à interrompre l'éxécution d'un programme par le MPU.
Ces lignes sont reliées aux $\overline{IRQ}$ ou $\overline{NMI}$ du MPU ou sont placées sur les entrées du contrôleur prioritaire d'interruption.

3 b - Avec la périphérie (PIA --- périphérie)

- PA_0 à PA_7 ; PB_0 à PB_7 : 16 lignes de données programmables individuellement en entrée ou en sortie.

* Brochage du PIA

fig ①

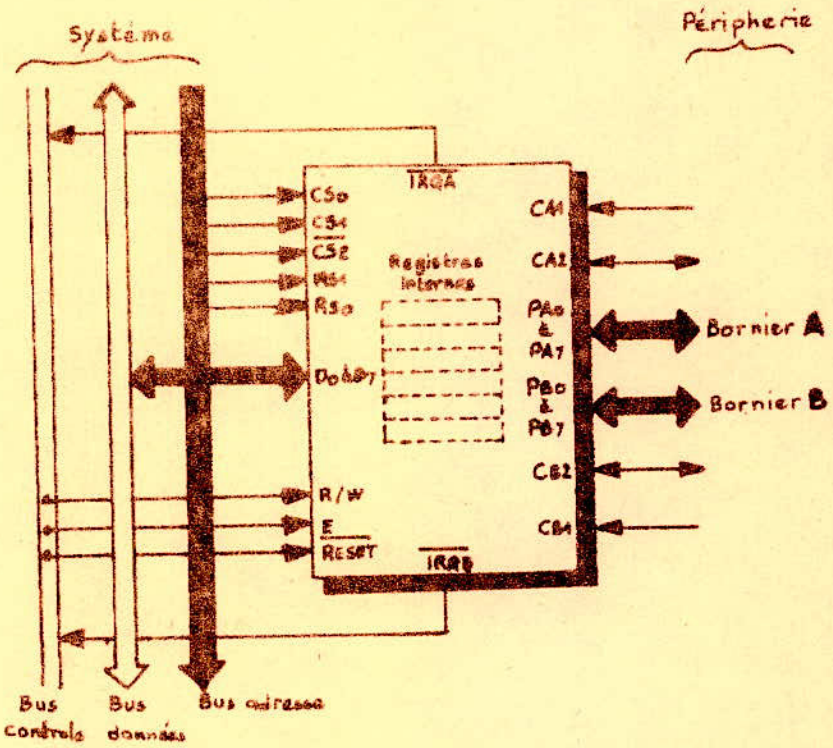


fig ②

Ces deux ports d'entrée/sortie reflètent, en sortie, le contenu des deux registres internes de 8 bits dont l'état binaire apparaît sous forme de tensions de sortie (+ 5 V : "1" logique et 0 V : "0" logique) maintenues tant qu'il n' ya pas de modification dans les registres.

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

Vss ; Vcc : Deux bornes d'alimentation Vcc = 5 V ; Vss = 0 V.

II - 4 ORGANISATION INTERNE :

Le schéma de la Fig. 3 représente le synoptique du coupleur d'entrée/sortie.

Nous remarquons que le microprocesseur peut adresser 6 registres en écriture et en lecture.

Ces 6 registres sont répartis en 2 groupes de trois relatifs à chacun des borniers A et B et qui sont :

- CRA ; CRB : registre de commande. Contient les paramètres de fonctionnement.

- DDRA ; DDRB : Registre de sens de transfert. contient le mot fixant le sens de transfert (entrée ou sortie) pour chacune des lignes de données.

Un état "1" définit une ligne en sortie
Un état "0" définit une ligne en entrée.

- ORA ; ORB : Registre de sortie. Mémorise les données en sortie lors d'une écriture, à la même adresse on peut lire les données présentées en entrée mais elles devront être mémorisées à l'extérieur.

- Deux cricuits de commande d'interruption A et B permettent de traiter CA_1 , CA_2 , CB_1 , CB_2 et de générer $\overline{IRQA}$ et $\overline{IRQB}$

- 4 a - Selection et adressage des registres :

Nous avons vu plus haut que 6 registres internes peuvent être adressés par le microprocesseur qui les considère toutefois comme 4 adresses mémoires.

Par les deux fils d'adressage RS_0 et RS_1 on peut choisir parmi 4 registres.

Un troisième fil aurait permis l'adressage de 8 registres alors qu'ils sont un nombre de 6.

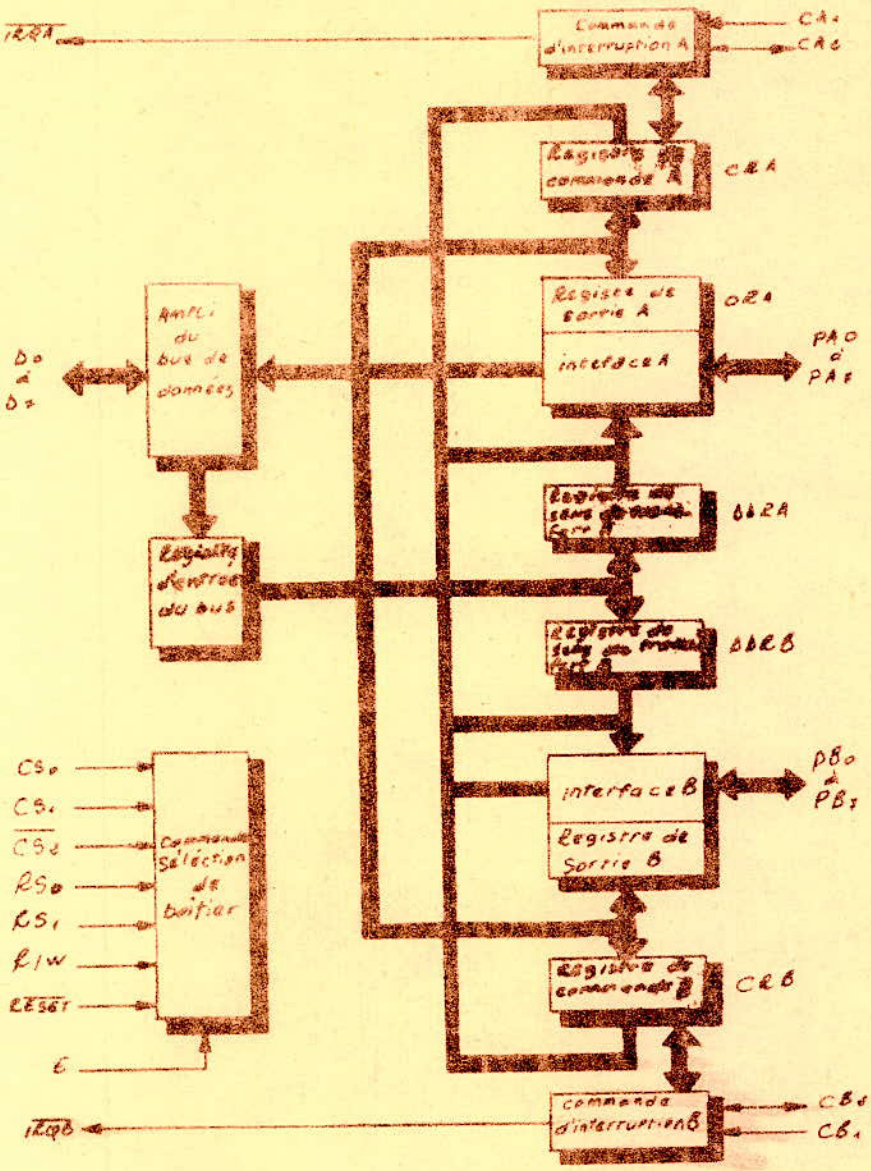


fig ③

RS_1	RS_0	CRA_2	CRB_2	Reg - adressé
0	1	-	-	CRA
0	0	0	-	DDRA
0	0	1	-	ORA et interface
1	1	-	-	CRB
1	0	-	0	DDRB
1	0	-	1	ORB et interface

fig ④

La solution choisie par le constructeur prévoit une économie de broches sur le boîtier. Ainsi les deux registres de commande, CRA et CRB, sont adressés directement comme l'indique la Fig.5 .

Les 4 autres registres DDRA ; DDRB ; ORA et ORB, sont adressés indirectement. Le choix entre ces 4 registres est fonction du bit "2" écrit au préalable dans CRA pour DDRA ou ORA et dans CRB pour DDRB ou ORB. voir Fig. 6 ; 7.

L'adressage de ces 6 registres internes du PIA peut donc se résumer sur le tableau de la Fig. 4.

RS_1	RS_0	CRA_2	CRB_2	REGISTRE ADRESSE
0	1	-	-	CRA
0	0	0	-	DDRA
0	0	1	-	ORA
1	1	-	-	CRB
1	0	-	0	DDRB
1	0	-	1	ORB

II - PROGRAMMATION DU P I A :

5 a - Ecriture des registres DDRA ; DDRB :

Chacune des lignes des deux borniers PA_0 à PA_7 et PB_0 à PB_7 peut être individuellement programmée en entrée ou en sortie.

Ceci est obtenu par l'écriture du mot"sens de transfert" dans DDRA ou DDRB.

- Quand un "0" est écrit dans le bit i du registre DDRA (par exemple) ce bit i de PA et programmé en entrée. Inversement quand un "1" est écrit dans le bit i du DDRA, la ligne PA_I est prggrammée en sortie.

- La Fig. 8 illustre les explications données ci-dessus.

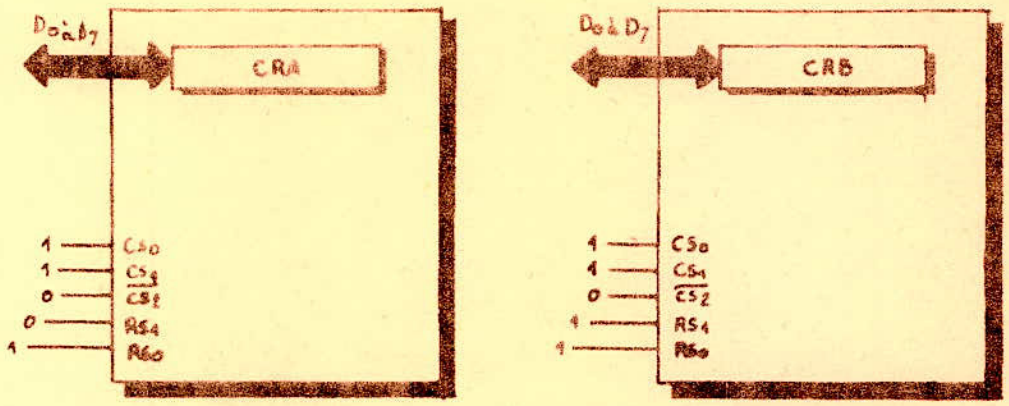


fig ⑤

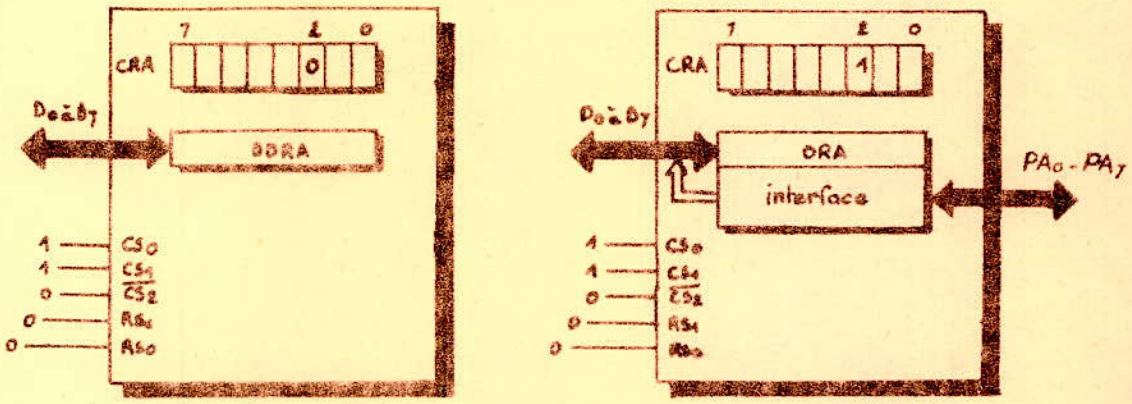


fig ⑥

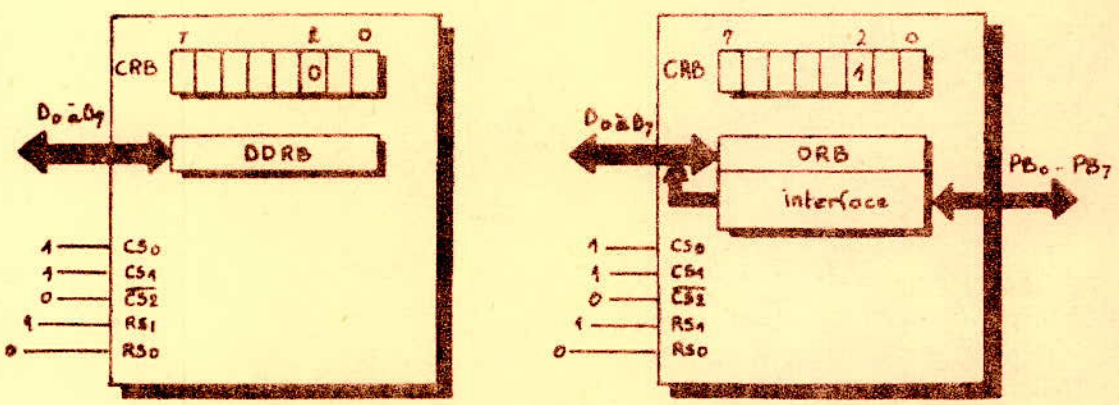


fig ⑦

Dans cet exemple les bits 0, 4 et 6 du DDRA sont à "0" donc les PA_0 ; PA_4 et PA_6 sont des entrées, les autres bits 1, 2, 3, 5, 7 sont à "1" donc PA_1 , PA_2, PA_3 , PA_5 , PA_7 sont des sorties.

On dira alors que les bits 0, 4, 6 sont programmés en entrée et les bits 1, 2, 3, 5 et 7 sont programmés en sortie.

5 -b : Le bornier A : PA_0 - PA_7 :

On a vu précedemment qu'il y a transfert de données sur les lignes programmées en sortie quand les bits correspondant du DDRA sont à "1".

Dans l'exemple de la Fig. 8 le microprocesseur écrit dans les bits D_1 , D_2 , D_3 , D_5 , D_7 à transmettre vers la périphérie dans le registre ORA. Les autres bits n'étant pas pris en compte. Seuls ces bits apparaissent sur les lignes PA_1 , PA_2 , PA_3 , PA_5 et PA_7.et sont disponibles en permanence il y a mémorisation des sorties.

Les données présentes sur les lignes PA_0 ,PA_4 et PA_6 apparaissent alors après amplification sur le bus des données du système afin que le microprocesseur puisse en effectuer la lecture. Mais il n'y a pas de mémorisation des entrées.

Il est à remarquer que lors de la lecture des lignes PAo , PA4 et PA6 le microprocesseur Lit un mot de 8 bits (OCTET) dont seuls D_0 , D_4 et D_6 sont représentatifs.

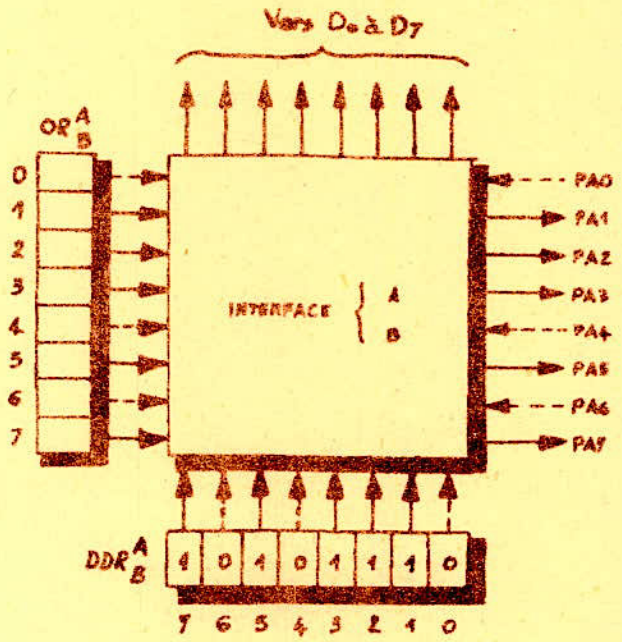
5 - c : Ecriture des registres de contrôle :

On a vu précedemment que parmi les 6 registres internes du PIA, deux d'entrée aux CRA et CRB contiennent les paramètres du fonctionnement du cricuit.

Dans ces deux registres les bits 2 permettent de définir l'adressage des autres registres. Tous les autres bits du CRA et CRB étant relatifs aux lignes d'interruptions CA_1 , CB_1 , CA_2 , CB_2, $\overline{IRQA}$ et $\overline{IRQB}$ disponibles sur le boîtier à 40 broches.

Le format des registres CRA et CRB est donné Fig. 9

Il faut savoir que les bits 6 et 7 ne peuvent être écrit mais lus seulement.



← Ligne programmée en Sortie

←-- Ligne programmée en Entrée

fig 7

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

fig 8a -

. d 1 : Mode de fonctionnement de CA_1 :

La programmation du mode de fonctionnement de la ligne d'interruption CA_1 est décrite dans le tableau ci-dessous.(Voir Fig. 8a.)

N°	CRA_1	CRA_0	TRANS.ACTIV Ent.d'Int CA_1	Indicateur d'Int.CRA_7	Sortie d'Int $\overline{IRQA}$ (vers MPU)
1	0	0	↓	mis à 1 par ↓	$\overline{IRQA}$=1 Inter. Masquée
2	0	1	↓	mis à 1 par ↓	passe à 0 qd CRA_7 passe à 1
3	1	0	↑	mis à 1 par ↑	$\overline{IRQA}$=1 Inter. Masquée
4	1	1	↑	mis à 1 par ↑	passe à 0 qd CRA_7 passe à 1

Prenons par exemple la première ligne (n°1) de ce tableau : Quand les bits CRA_0 et CRA_1 sont "0" une demande d'interruption est prise en compte sur le front descendant du CA_1, l'indicateur d'interruption CRA_7 associé à CA_1, est mis à 1 et le PIA génére un signal de sortie d'interrup. masquée (par CRAo = "0") et cependant mémorisée et devient active quand CRAo passe à "1", $\overline{IRQA}$ est activée à 0 d'ou interruption du programme en cour.

L'illustration graphique de ces modes de fonctionnement est donnée Fig. 10.

REMARQUE : On a vu que le CRA_7 est mis à "1" par le front actif de CA_1 et si le masque d'interruption est enlevé, la mise à "1" de CRA_7 active à "0" l'$\overline{IRQA}$.Et pour désactiver l'$\overline{IRQA}$, c'est-à-dire le remettre à "1", il faut faire effectuer une lecture au microprocesseur qui fera descendre le CRA_7 à "0" et ce dernier fera remonter l'$\overline{IRQA}$ à "1" (voir Fig.10, cas n°2)

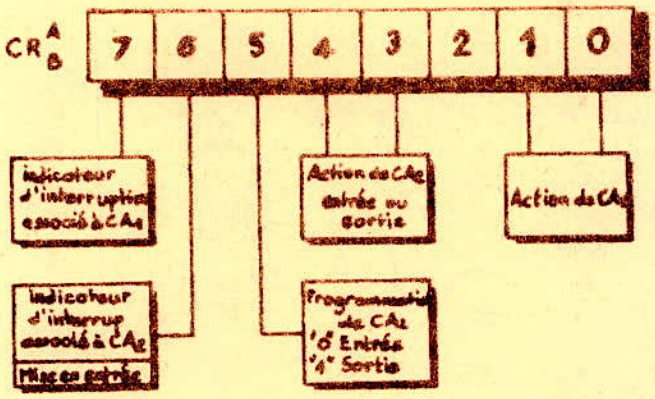

CR A/B
7
6
5
4
3
2
1
0
Indicateur d'interruption associé à CA1
Action de CA2 entrée ou sortie
Action de CA2
Indicateur d'interrup associé à CA2
Mise en sortie
Programmation de CA2 "0" Entrée "1" Sortie

fig 9

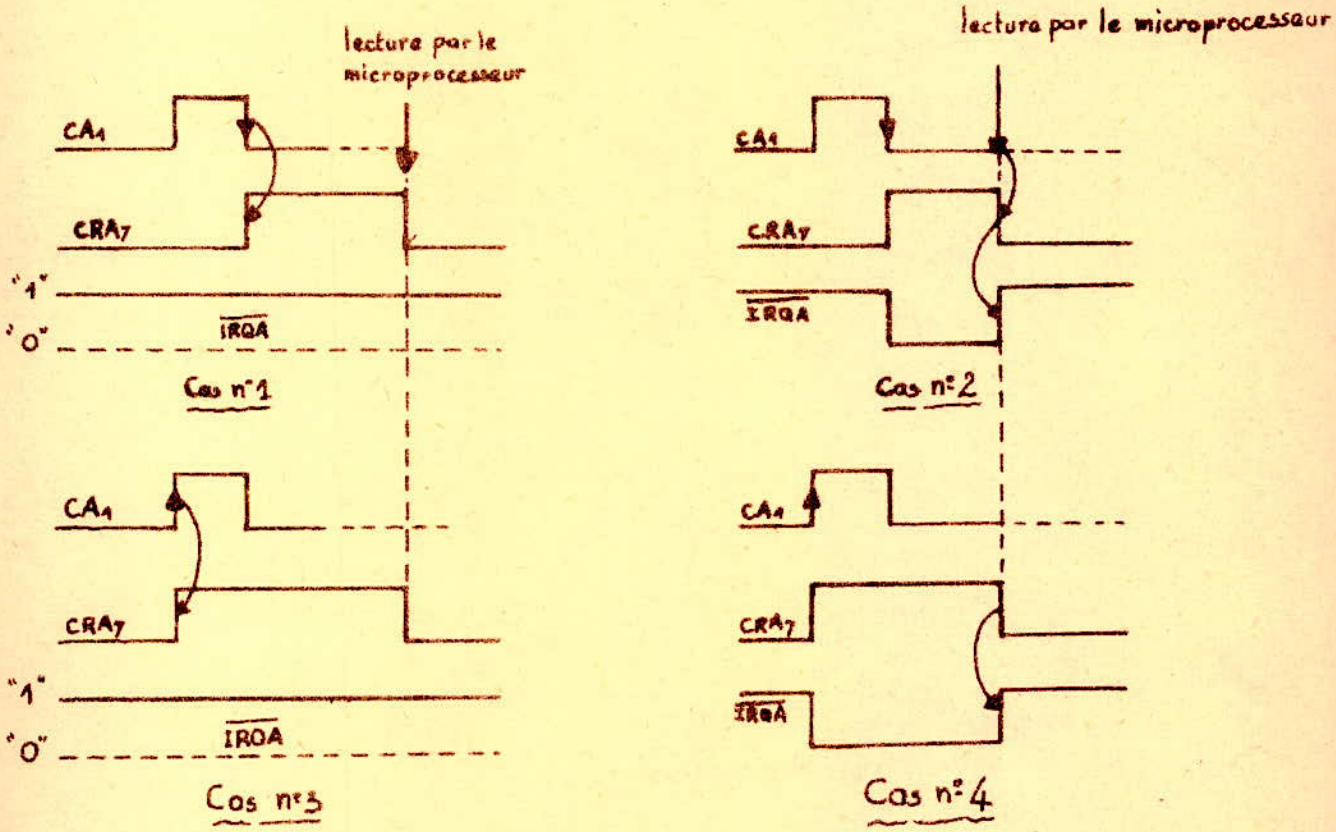

lecture par le microprocesseur
CA1
CRA7
"1"
"0"
IRQA
Cas n°1
CA1
CRA7
"1"
"0"
IRQA
Cas n°3
lecture par le microprocesseur
CA1
CRA7
IRQA
Cas n°2
CA1
CRA7
IRQA
Cas n°4

fig 10

. d - 2 Mode de fonctionnement de CA_2

Le CA_2 peut être programmé soit en entrée ou en sortie en mettant àn"1" ou un "0" dans le bit 5 du CRA ("1" sortie, "0" entrée).

Si CA_2 est programmé en entrée (bit 5 du CRA à "0") CA_2 joue le même rôle que CA_1. Dans ce cas les bits CRA_3, CRA_4 et CRA_6 jouent aussi le même rôle que les bits CRA_0, CRA_1 et CRA_7.

- Si CA_2 est programmé en sortie (bits de CRA à "1") dans ce cas CRA_3 et CRA_4 permettent de définir les modes d'action de CA_2.

Selon la programmation des bits CRA_4 et CRA_3 on distingue 4 modes de fonctionnement (voir tableau 1 ci-dessus).

Tableau : 1

CRA_4	CRA_3	MODES
0	0	dialogue — associé à une lecture
0	1	impulsionnel — associé à une lecture
1	0	programmé
1	1	programmé

Dans le mode programmé la sortie CA_2 suit la programmation du bit CRA_3 du registre CRA.

Dans le cas du mode impulsionnel et dialogue, CA_2 est associé à une lecture.

. d - 3 Mode de fonctionnement de CB2

De même que pour CA_2 la programmation de cette ligne en sortie de commande s'obtient en écrivant un "1" dans CRB_5, de même CRB_4 et CRB_3 permettent de définir les modes d'action de CB_2. Voir tableau 2)

Tableau : 2

CRB_4	CRB_3	M O D E S
0	0	dialogue } associé à une écriture
0	1	impulsionnel }
1	0	Programmé
1	1	

Dans le mode programmé, comme pour le CA_2, la sortie CA_2 suit la programmation du bit CRB_3 du CRB.

Mais contrairement à CA_2, dans le mode dialogue et impulsionnel, le CB_2 est associé à une écriture.

CHAPITRE III

ETUDE ET REALISATION DE L'INTERFACE DIALOGUE

-=-=-=oOo-=-=-=-

III - 1 INTRODUCTION

- 1. 1 Transmission série
- 1. 2 Transmission parallèle

III - 2 METHODE DE DIALOGUE ENTRE MPU

- 2. 1 Interruptions simples
- 2. 2 Interruptions multiples
  - 2 - 2 a Polling
  - 2 - 2 b Interruptions vectorisées

III - 3 UTILISATION DU PIA

III - 4 REALISATION DES INTERFACES

- 4 - a Synoptique carte PIA
- 4 - b Analyse des différents blocs
  - 4. b- 1 : Les buffers 3 états 8T 26
  - 4. b- 2 : " " Adresse 8T 95
  - 4. b- 3 : Circuit de décodage
  - 4. b- 4 : " " de commande
- 4 - c Sélection et dressage du PIA
  - 4. c- 1 : Sélection du PIA
  - 4. c- 2 : Adressage du PIA

III - 5 COUPLAGE DES DEUX CARTES PIA.

III - 1 INTRODUCTION

Un microprocesseur n'est pas fait en général pour dialoguer uniquement avec ses mémoires de programme et de données. Il doit pouvoir communiquer avec la périphérie c'est-à-dire recevoir ou transmettre des informations de ou vers le milieu extérieur, ce qui établit le dialogue entre l'homme et la machine ou comme dans notre cas entre plusieurs machines (système maître-esclave) Voir Fig.I ce qui fait accroitre considérablement le champ d'application des microprocesseurs.

La communication avec l'extérieur, dans les deux sens (entrée/sortie) pose le problème des modes et des moyens de transmission des informations. En ce qui concerne les modes, il existe deux :

- un mode de transmission série
- un mode de transmission parallèle

- 1. 1 Transmission série

On dit transmission série lorsque les bits de l'information se présentent séquentiellement dans le temps. Les bits de poids successifs d'un mot se succédent, séparés par un intervalle de temps dépendant de la fréquence de transmission.

Dans ce cas deux types de transmission sont utilisés :

- Transmission asynchrone série
- " " synchrone série

- 1. 2 Transmission parallèle

On dit transmission parallèle lorsque les bits d'un mot sont émis simultanément en parallèle, sur les huit lignes programmables en entrée ou en sortie. Afin de signaler au recepteur l'envoi d'une information (mot), on crée des signaux de demande d'échange et d'acceptation.

III - 2 METHODE DE DIALOGUE ENTRE LES MICROPROCESSEURS :

L'échange de données entre le microprocesseur et les organes extérieurs (esclaves) doit se faire correctement en sachant la disponibilité du bus. Pour cela, il doit envoyer un signal de disponibilité, c'est le principe des "demandes d'interruption".

Les interruptions sont des évènements qui provoquent l'arrêt d'un programme en cours de traitement et le passage à un sous-programme.

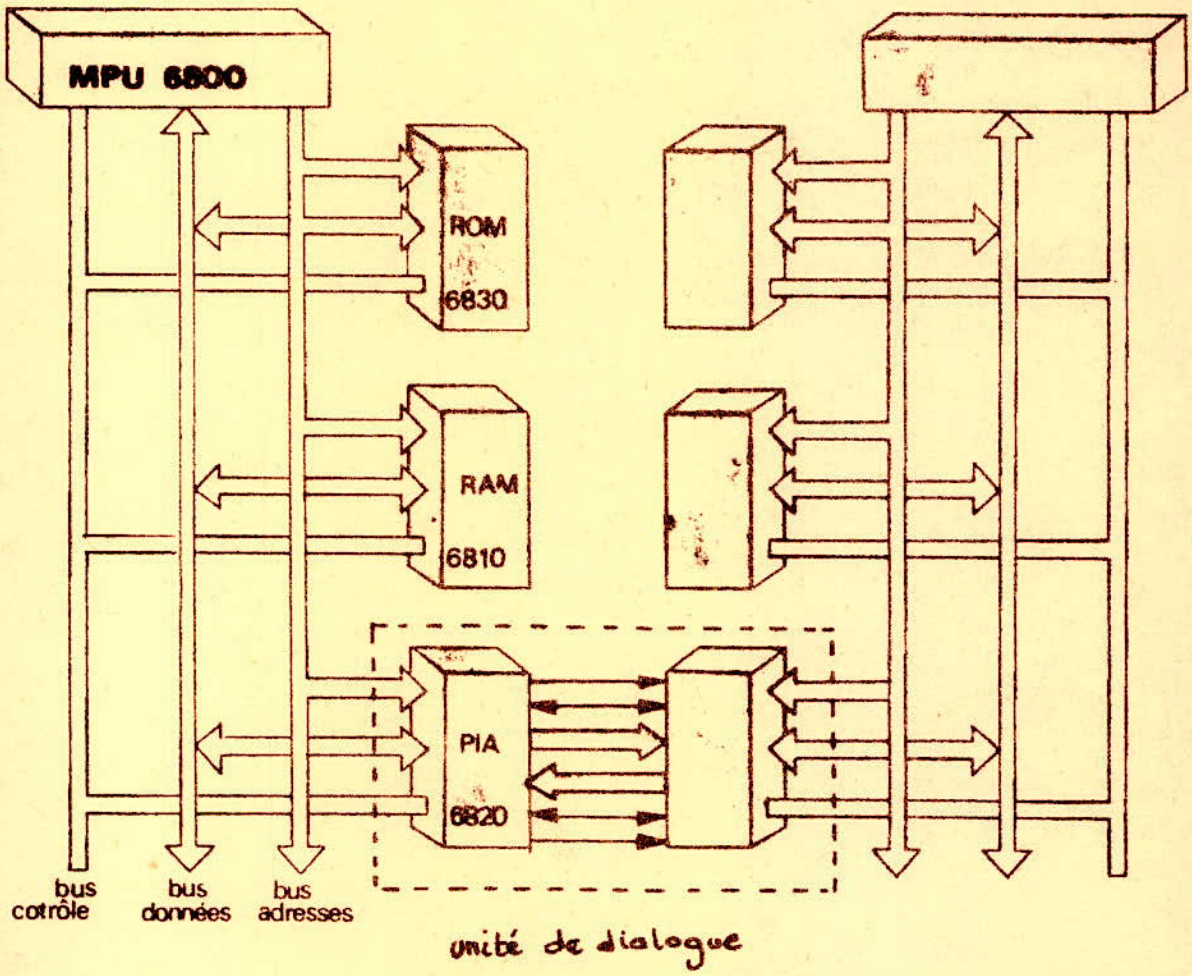

MPU 6800
ROM
6830
RAM
6810
PIA
6820
bus
cotrôle
bus
données
bus
adresses
unité de dialogue

fig I

Une bonne gestion de ces interruptions permet l'exécution de certaines tâches et facilite le travail à l'utilisateur. Les domaines d'utilisation d'un processus d'interruptions sont assez vastes, tels que :

- Les interruptions dûes aux erreurs ou aux pannes de machine (panne d'alimentation, erreur de parité en mémoire).
- Les interruptions de programme (instruction ou adresse incorrecte, opération impossible, division par zéro...).
- Les interruptions d'entrée/sortie (besoin d'échanges d'informations).

C'est ce dernier cas qui permet le dialogue entre le microprocesseur et l'organe extérieur. On peut distinguer deux sortes d'interruptions - interruption simple et lesinterruptions multiples.

2. 1 - Interruption simple :

Pour que le processus d'interruption soit pris en compte, le microprocessus devra être doté de deux lignes supplémentaires - l'une pour la demande d'interruption et l'autre pour l'acceptation. Lorsque le circuit d'entrées/ sorties envoie une "demande d'interruption" au MPU, il lui répond en accusé de reception, par un ordre de "demande d'interruption acquise" correspondant à l'autorisation.

Lors de l'arrivée d'un signal d'interruption, celui-ci est pris en mémoire et déclenche, après achèvement de l'instruction en cours, le transfert le contenu des registres internes (qui regroupent le compteur ordinal- le registre d'index - les accumulateurs A et B et enfin les indicateurs d'états) dans la pile de sauvegarde (STACK) autre fraction de la mémoire RAM dont le niveau de remplissage est assuré par un pointeur de pile noté SP.

Le MPU se branche alors sur l'adresse du vecteur $\overline{IRQ}$, et va lire dans les cellules mémoires qui sont ici FFF8 et FFF9 l'adresse de sous-programme spécifique de l'interruption.

Après l'exécution de ce sous-programme, il y aura restitution du contenu des registres, c'est-à-dire transfert des données des emplacements mémoires (PILE)

vers les registres internes et le programme qui avait été suspendu pourra prendre son exécution grace à une instruction spéciale dite le "retour d'interruption".

2. 2 - Interruptions multiples :

Le microprocesseur ne dispose que d'une entrée $\overline{IRQ}$ (Interruption Request). Or les micro-ordinateurs peuvent dialoguer généralement avec plusieurs organes extérieurs (esclaves). Lorsque les esclaves sont susceptibles de demander une interruption un double problème se pose

- Le microprocesseur doit identifier l'esclave demandeur.
- Dans le cas de plusieurs demandes d'interruptions simultanées, il doit savoir dans quel ordre de priorité il faut les desservir.

Pour résoudre un tel problème, deux solutions sont possibles :
- soit réunir les demandes d'interruption en OU-cablé et rechercher l'esclave qui a fait la demande par serution "Polling".
- soit utiliser un circuit de hierarchisation des priorités c'est le mode interruptible vectorisé "Vectored Interrupt".

2. 2 a - Polling :

C'est le mode interruptible avec test des bits d'états. Dans ce mode, plusieurs esclaves pouvant demander simultanément une interruption, il est nécessaire de pouvoir mémoriser ces demandes, tant que le microprocessus ne les a pas toutes acquittées. Dans ce but, les interfaces des esclaves disposent des bits d'état "Flags", validés lors d'une demande d'interruption. On remarque que cette procédure est d'emploi assez simple mais néanmoins qu'elle ralentit le fonctionnement du système, car le programme Polling sera de nombreuses fois exécuté en cas de demandes d'interruptions répétées.

2. 2 b - Interruptions vectorisées :

On dit que les interruptions sont vectorisées, lorsqu'une interruption provoque un branchement directement à l'adresse du programme du traitement de celle-ci. Il s'agit dans ce cas d'un adressage indirect.

Par conséquent la vectorisation consiste à donner les adresses qui apparaissent sur le bus d'adresses après chaque type d'interruption.

Cette opération s'effectue de plusieurs méthodes dont celle qui consiste à utiliser un controleur de priorité (PICU) qui permet avec un minimum de logiciel, de réaliser des interruptions "dirigées". L'adresse du programme de gestions de l'esclave est générée par le 8212 (dont on trouvera les explications par la suite). Le microprocessus charge le vecteur d'interruption via le dota bus, dans son compteur du programme, ce qui provoque le branchement à l'adresse réelle du programme d'interruption cherché. Mais un problème qui peut surgir c'est celui de l'arrivée d'une interruption de forte priorité en cours d'exécution d'une interruption de faible priorité. L'idéal serait de pouvoir interrompre, dans ce cas, l'éxecution de l'interruption en cours. C'est ce qui se fait grâce à un encodeur de priorité qui compare le niveau de priorité de toute interruption se signalant avec celui de l'interruption en cours d'éxécution et par suite le sous-programme de l'instruction prioritaire est exécuté.

III - 3 UTILISATION DU PIA

Comme on doit travailler en transmission parallèle et que l'on doit utiliser les PIA, nous vous donnons un apperçu pour ce qui est de son utilisateur.

Notons que les deux PIA (PIA interface Maître et PIA interface esclaves) sont utilisés de la même manière qu'ils jouent le même rôle.

1 - Les huit lignes du port A (les PAi) sont utilisées comme entrée parallèle de 8 bits, alors que le port B (les PBi) est utilisé comme sortie. Ces opérations sont réalisables en écrivant respectivement des "1" ou des "0" dans les registres de direction correspondants (DDRA et DDRB), à partir du MPU par une instruction d'écriture.

2 - La ligne CA_1 est utilisée comme signal "information prête" à l'entrée. Une transistion positive ou négative (suivant programmation du registre de contrôle), placera le bit CRA_7 à 1 qui remettra à "0" $\overline{IRQA}$ qui est relié à l'$\overline{IRQ}$ du MPU.

3 - La ligne CA_2 peut être programmée de façon à être remise à "0" par une opération de lecture du PIA et rester dans cet état jusqu'à une nouvelle transition de CA_1. CA_2 constitue donc un signal d'acquitement.

4 - CB_1 constitue une signal "requête d'information", il détecte une transition positive ou négative, met à "1" le bit CRB_7 et remet à "0" l'$\overline{IRQB}$ qui est relié à la ligne d'interruption du MPU.

5 - La ligne CB_2 est programmée de façon à être remise à "0" par une instruction d'écriture dans le PIA, constituant de ce fait un signal "d'information prête" en sortie.

III - 4 REALISATION DES INTERFACES :

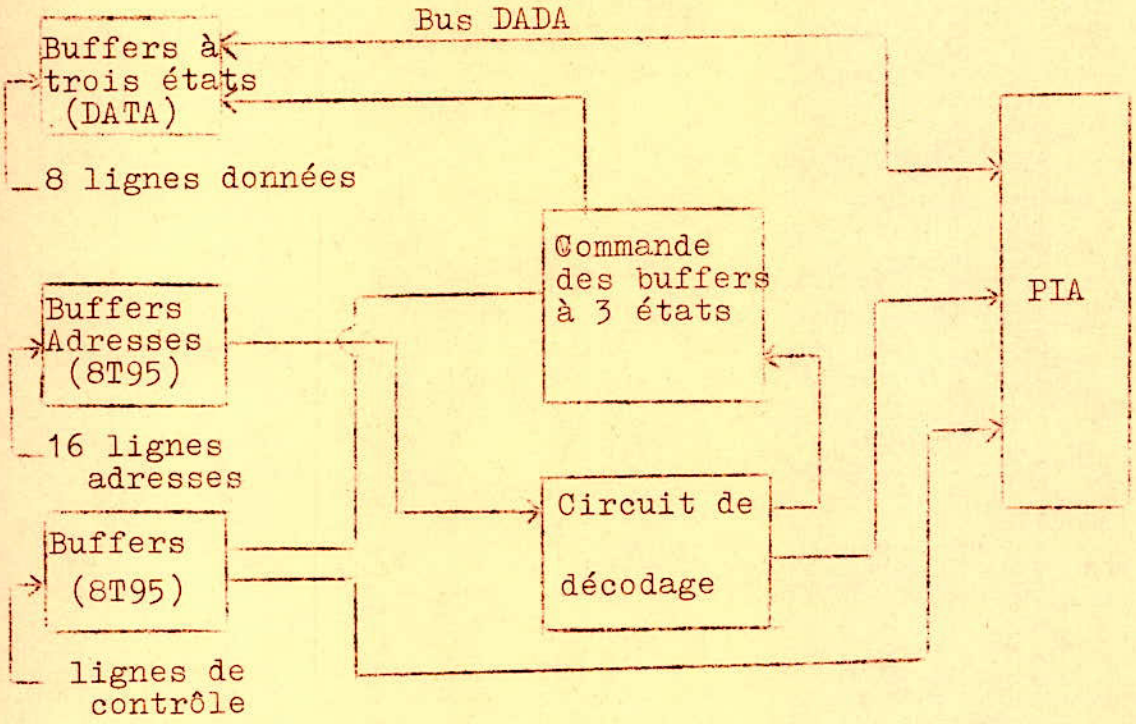
Cette réalisation se compose de deux parties :

1°) Réalisation hardware des deux cartes PIA

2°) " " " " du couplage des 2 cartes PIA

4. a - Synoptique carte PIA.

Le schéma synoptique des cartes PIA est donné à la page suivante :



Notons que les deux cartes PIA (carte PIA Maître et Carte PIA esclave) sont identiques.

4. b- Analyse des différents blocs de ce schéma :

4. b- 1 ; les buffers à 3 états (8T 26)

Ce sont des circuits amplificateurs, inverseurs, bidirectionnel. Voir Fig. en annexe). Ils sont connectés aux bus de données. La commande de ces 8T 26 définit le sens de transfert (lecture ou écriture). Ainsi elle synchronise le transfert des données et la sélection du PIA adressé.

Table de vérité de la logique de commande des 8T 26 :

$\phi 2$	R/W	Sortie des comparateurs	Borne1 du 8T26	Borne15 du 8T26	Etat des 8T26
1	1	1	1	1	Lecture
1	0	1	0	0	Ecriture
0	ø	1	1	0	Haute-Impédance

4. b- 2 Buffers Adresse 8T 95 :

Les 8T 95 jouent le même rôle que les 8T 26. Ils ont 3 états, c'est des circuits amplificateurs, mais contrairement aux 8T 26 sont non inverseure. (Voir Fig. en annexe).

Table de vérité du 8T 95

Borde 15 du 8T95	Borne 1 du 8T95	Entrée	Sortie
0	0	0	0
0	0	1	1
0	1	ø	H. Impétance
1	0	ø	H. Impétance
1	1	ø	H. Impétance

4. b-3 Circuit de décodage :

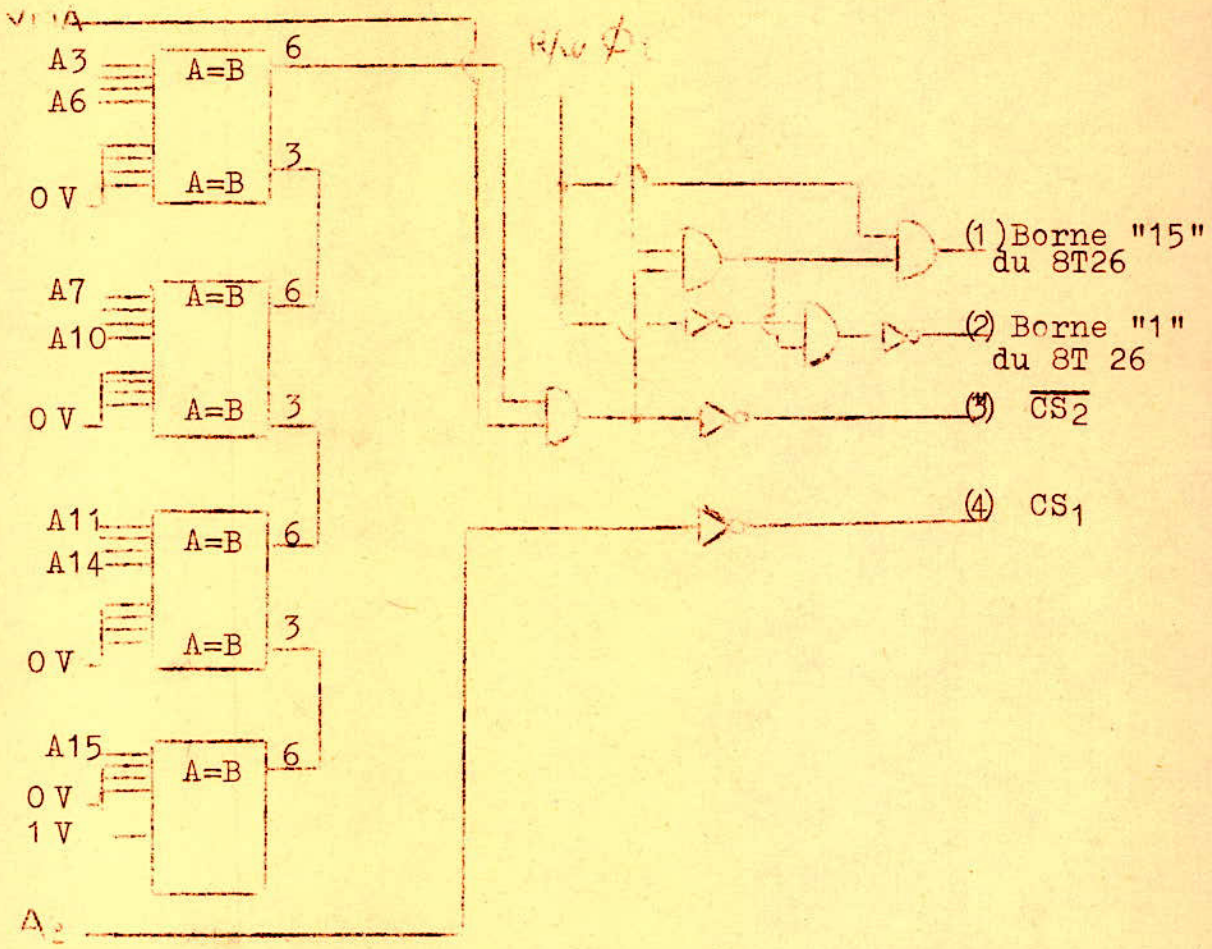
Ce circuit est constitué de quatre comparateurs de bits. Ces comparateurs ont huit entrées et trois sorties (Voir Fig. en annexe).

- Quatre entrées (A) sont fixées par l'utilisateur et les quatre autres (B), à comparer sont reliés aux quatre lignes d'adressage. Il existe trois cas de comparaison (sortie) A = B, A < B, A > B. et suivant le cas désiré ou choisie l'une de ces sorties. Dans notre cas c'est la 1ère A = B qui nous interesse. Donc à l'égalité des états (ceux fixés par l'utilisateur et ceux présentés par le bus d'adresse) la sortie "6" (A = B) du comparateur passe à l'état "1".

Le choix de ce circuit de décodage (comparateur nous permet d'avoir un adressage translatable ce qui donne une large utilisation de l'interface.

4. b- 4 Circuit de commande des 8T26 :

Comme l'indique le schéma ci-après, on a utilisé des ports "ET" et des inverseurs pour avoir la valeur ("1" ou "0") désirée sur chacune des sorties 1,2,3,4.



4. c <u>Sélection et Adressage du PIA</u> :

4. c- 1 : <u>Selection du PIA</u> :

Nous avons vu precédemment (etude du PIA) que la sélection d'un PIA est donnée par les 3 lignes (sélect-ships) CSo, CS_1, $\overline{CS_2}$ ayant respectivement les valeurs 1, 1, 0.

Pour avoir $\overline{CS_2}$ = 0 dans notre cas, on l'a relié à la sortie du comparateur et du VMA via une porte "ET" et un inverseur.

De même pour avoir CS_1 = 1, on le prend à la sortie d'un inverseur dont l'entrée est A2.

Quant au CSo on l'a relié directement à 5 V donc CSo = 1.

4. c- 2 Adressage du PIA :

Comme l'adresse fixée par les comparateurs est 8000 et que nous devons adressé les 4 registres du PIA, on a réservé Ao et A_1 que nous avons reliées respectivement à RSo et RS_1 et qui par leur 4 combinaisons binaires (Voir Fig. 4- Chp.II Etude du PIA) adresseront l'un des 4 registres du PIA.

8000 DDRA ou ORA
8001 CRA
8002 DDRB Ou ORB
8003 CRB

on présente dans le tableau suivant les lignes d'adresse du PIA :

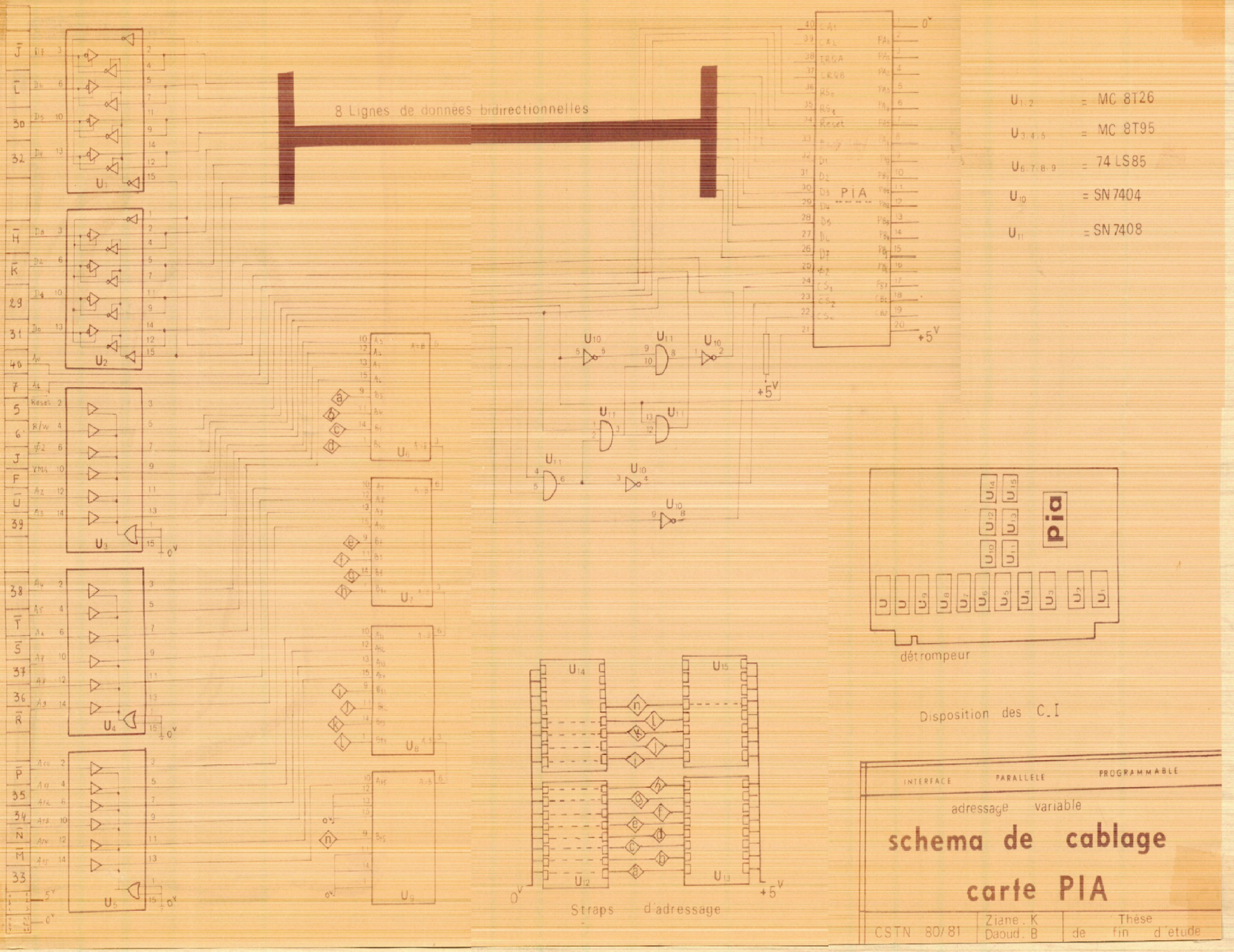
A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0	CODE HEXA
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8000
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	8001
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	8002
1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	8003

Le schéma global se trouve à la page suivante :

III - 5 COUPLAGE DES DEUX CARTES PIA :

Le schéma de principe est donné par la Fig.II

On a réservé les ports A des PIA (Maître et esclave) pour la réception des informations et les ports B pour l'émission. Quant aux lignes de commandes, CA_1, CA_2, CB_1, CB_2, on a relié les CA_1 de l'un aux CB_2 (sortant) de l'autre et les CB_1 de l'un aux CA_2 (sortant) de l'autre, ceci pour satisfaire le principe d'utilisation du PIA énoncé précédemment.



$U_{1,2}$	= MC 8T26
$U_{3,4,5}$	= MC 8T95
$U_{6,7,8,9}$	= 74 LS85
U_{10}	= SN 7404
U_{11}	= SN 7408

Disposition des C.I

INTERFACE PARALLELE PROGRAMMABLE

adressage variable

schema de cablage carte PIA

CSTN 80/81	Ziane. K Daoud. B	These de fin d'etude

~ Schéma de principe de l'unité de dialogue ~

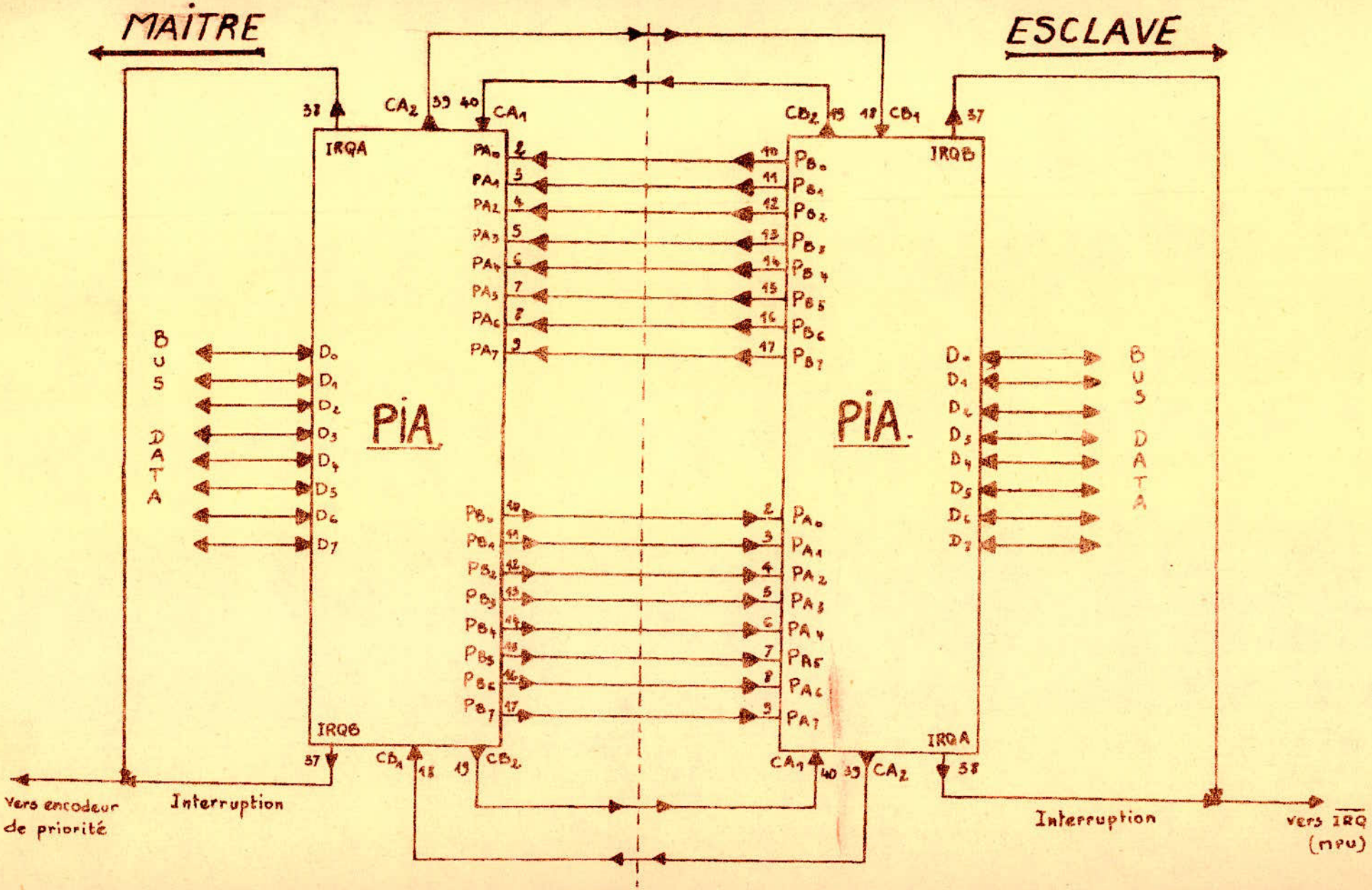


fig II

CHAPITRE IV

ETUDE DU LOGICIEL

-=-=-=-oOo-=-=-=-

IV - 1 PROGRAMMATION DU PIA

1. 1 Sous-programme d'initialisation du PIA

1. 2 Etat des registres

IV - 2 PROGRAMME D'INTERUP.AVEC SAUT AUX S/PROG.D'INTERRUP.

2. 1 Sous-programmes utilisés

IV - 3 PROGRAMME DE TRANSMISSION, RECEPTION D'UN OCTET.

3. 1 Principe.

3. 2 Sous-programmes utilisés

IV - 4 PROGRAMME DE TRANSMISSION, RECEPTION D'UNE TABLE.

4. 1 Principe

4. 2 Sous-programmes utilisés.

IV - 1 PROGRAMMATION DU PIA

Notons que la programmation que nous avons fait se compose de plusieurs sous-programmes, et pour faire exécuter ces sous-programmes dans la hiéarchie désirée, on trace un programme dit "Programme Plan" se composant essentiellement de JSR (Jump to soubroutine). Aussi nous avons donné un nom à chaque sous-programme pour raison de comodité de programmation.

1. 1 - Sous-Programme d'initialisation du PIA "INITA"

Puisque nous avons déterminé les adresses des registres internes du PIA, on peut donc l'initialiser de telle sorte à avoir le port A entrant, le port B sortant et les CA_2, CB_2 programmées en sortie.

- Le sous-programme "INITA" est le suivant :

```
4F          CLR A       00--> A
B7 8001     STA A CRA   A --> CRA
B7 8000     STA A DDRA  A --> DDRA
B7 8003     STA A CRB   A --> CRB
86 FF       LDA A FF    FF--> A
B7 8002     STA A DDRB  A --> DDRB
86 3D       LDA A 3D    3D--> A
B7 8003     STA A CRB   A --> CRA
B7 8001     STA A CRA   A --> CRA
39          RTS
```

1. 2 - Etat des registres du PIA :

0	0	0	0	0	0	0	0	DDRA --> 0 port A entrant

1	1	1	1	1	1	1	1	DDRB --> FF port B sortant

x	x	1	1	1	1	0	1	CRA --> 3D

- 1 : demande d'interrup.non masquée
- CRA_7 mis à "1" par une transition négative de CA_1
- 1 : Sélection de ORA
- CA_2 en mode programmé $CA_2=CRA_3$
- 1 : Programmation de CA_2 en sortie
- x : mis à "1" par une transition négative de CA_1 etremis à "0" par une lecture de ORA ou Reset

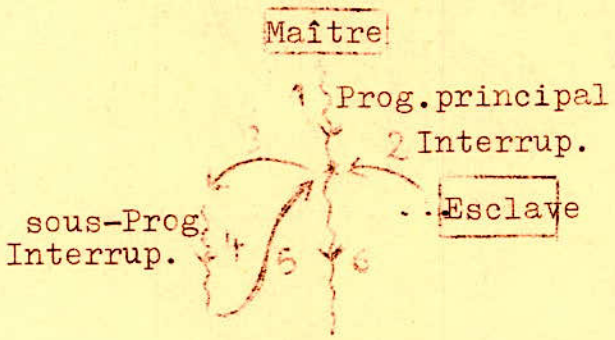
Le registre CRB est programmé de la même manière que le CRA.

IV - 2 PROGRAMME D'INTERRUP. AVEC SAUT AU S/PROGR.D'INTERRUP.

- Plaçons nous dans le cas où le maître exécute un programme principal et qu'un esclave vient l'interrompre.

Pour bien illustré ce cas, prenons comme exemple: Le Maître déroule un programme (écriture de "PROGRAMME" sur visue) et quand il est interrompu, il saute au sous/ programme d'interruption (écriture de "SOUS-PROGRAMME" dur visu).

Le schéma synoptique suivant nous donne, suivant la numérotation des flèches, le déroulement des étapes.



1 - Le Maître exécute le programme principal.

2 - L'esclave envoi une interruption

3 - Le Maître arrête son programme principal (il est interrompu) et saute au S/Prog. d'interruption.

4 - Le Maître exécute le s/prog. d'interrup. (écriture "SOUS-PROGRAMME")

5 - A la fin du s/prog. d'interrup., le Maître trouve un RTI (Return From interrupt) et repart au point où il a quitté son programme principal.

6 - Le Maître continue l'éxécution de son programme principal.

Ce que nous obtenons sur la visu est :

PROGRAMME
PROGRAMME
PROG
SOUS-PROGRAMME
RAMME
PROGRAMME

2. 1 Sous-programmes utilisés

1- a Inistialisation : INITA
1- b Programme principal : PROGA
1- c Sous-programme d'interruption : PROGB
1- d Interruption : INTERUP.
1- e Programme Plan: PROGP

1 - a INITA :

C'est le sous-programme d'initialisation donné précedemment à qui on ajoute l'adresse ou doit se placer le vecteur d'interruption FFF8, FFF9.

1 - b PROGA :

C'est le sous-programme d'écriture "PROGRAMME" sur visu. (Voir Chp. V).

1 - c PROGB:

C'est le sous-programme d'écriture "SOUS-PROGRAMME" sur visu (voir chp. V).

1 - d INTERUP :

C'est le sous-programme que l'esclave exécute pour interrompre le Maître. L'interruption se fait à l'aide des CA_1, CA_2, CB_2, CB_1 (Voir timing). Le CB_2 est porté à "1" à l'initialisation puis à "0" par le sous-programme INTERUP. Comme CA_1 est relié au CB_2, donc il sera lui aussi activé à "0" ce qui fera monter le CRA_7 à "1" et mettre l'$\overline{IRQ}$ à "0" d'où interruption. Pour remettre l'$\overline{IRQ}$ à "1", il faut faire une lecture du registre DATA considéré du PIA.

IV - 3 PROGRAMME DE TRANS/RECEP D'UN OCTET :

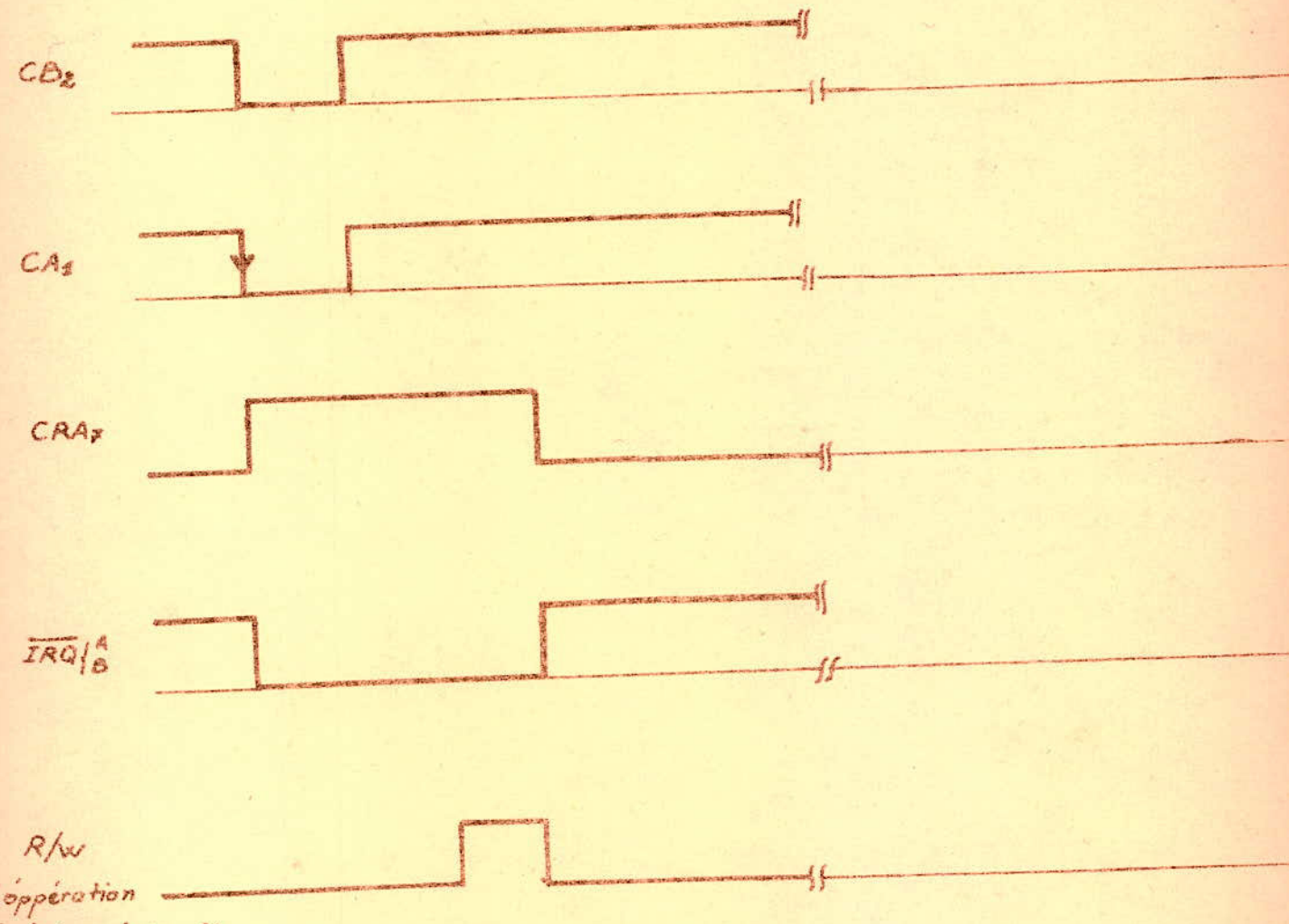
On a vu précedemment que la transmission d'information se fait par Octet.

3 -1 Principe :

Prenons le cas où l'esclave envoi un octet au Maître.

On veut transmettre un octet se trouvant dans une position mémoire de l'esclave à une position mémoire du Maître. Pour cela, on écrit l'octet à transmettre dans une adresse que nous fixons dans le Programme Plan de l'esclave "TRAOCT" dans notre cas "1500",une fois l'octet transmis par l'esclave arrive au maître, ce dernier le lit et le stock dans une adresse que nous avons fixé dans le Programme Plan du Maître "RECOCT" qui est "700" dans notre cas.

TIMING


CB2
CA2
CRA7
IRQ A/B
R/W
(opération de lecture du registre DATA par le MPU).
"1"
"0"

C'est deux adresses seront par la suit e dans le pragraphe suivant (transmission/reception d'une table), les adresses début des tables à transmettre ou à recevoir.

- Le déroulement des étapes se fait comme suit :

Pour l'esclave :

Après l'exécution du sous-programme d'initialisation, il passe au sous-programme d'écriture et de transmission de l'octet à transmettre puis au sous-programme d'interruption "INTERUP"

Pour le Maître:

Après le sous-programme d'initialisation, il exécute le programme de visualisation "VISU" et quand il sera interrompu, il saute au sous-programme d'interruption qui est la lecture de l'octet reçu, puis retour au sous-programme"VISU"

3. 2 - Sous-programmes utilisés :

Pour le Maître :

a - INITIALISATION : INIT.
b - Programme de visualisation : VISU.
c-- Sous-programme d'interuption : RECEPO
d - Programme Plan : RECOCT.

Pour l'esclave :

a - Initialisation : INIT.
b - Sous-programme d'écriture et transmi-:TRANSO.
c-- Sous-programme d'interuption : INTERUP.
d-- Programme Plan : TRAOCT.

ces sous-programmes sont donnés au Chp. V.

IV - 4 PROGRAMME DE TRANS/RECEP d'UNE TABLE :

On se propose de transmettre une table de "n" Octets avec $n \leq 256$.

4. 1 Principe :

Le principe étant le même que celui de la transmission - réception d'un octet, à part que dans ce cas, on transmet plusieurs octets ce qui nécessite un autre sous-programme que nous appelons"DECOMPT" qui a pour rôle d'arrêter la transmission quand le nombre n d'octet désirant être transmis est atteint.

4. 2 Sous-programmes utilisés :

a - Initialisation : INIT
b - Programme de visualisation : VISU.
c - Sousprogramme d'interruption : RECEP.
d - Programme Plan : RECTAB.

Pour l'esclave :

a - Initialisation : INIT
b - Sous-programme de transmission : TRANS.
c - Sous-programme d'interruption : INTERUP.
d - Sous-programme de décompte : DECOMPT.
e - Programme Plan : TRATAB.

Ces sous-programmes sont donnés au chapitre V.

- Rappelons que pour le cas que nous venons d'étudier, c'est l'esclave qui envoit une table au Maître, pour avoir l'inverse, il suffit de mettre les sous-programmes utilisés par le Maître dans l'esclave et inversement.

Mais comme on a besoin des deux cas c'est-à-dire, que la transmission/réception peut se faire du Maître vers l'esclave ou de l'esclave au Maître, on met tous les sous-programmes utilisés dans le Maître et dans l'esclave et suivant qu'on veut transmettre ou recevoir une table on utilise les programmes Plan TRATAB ou RECTAB (voir chp.V). Dans ce cas il faut prévoir deux zones mémoires, l'une pour écrire la table à transmettre et l'autre pour l'écriture de la table reçue. On a prévu les adresses suivantes :

-"700" adresse début table de réception
-"600" " " " de transmission

Donc pour transmettre une table on écrit les informations en "1500" qui seront stocker en "600" puis envoyer.

Et pour lire les informations de la table qui fut transmise par l'un des esclaves ou le Maître (suivant les cas),on se place en "700".

CHAPITRE V

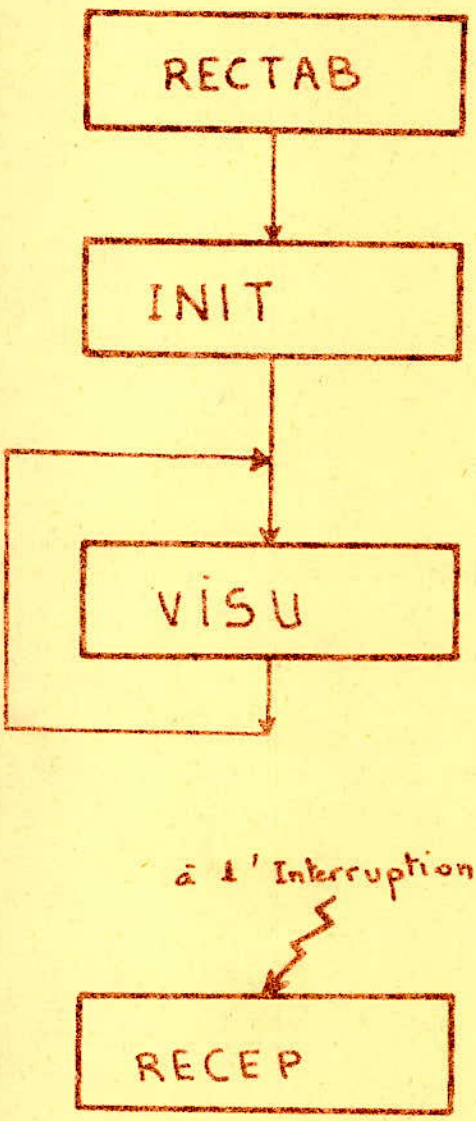
(_)RGANIGRAMMES ET

PROGRAMMES DE GESTIONS.

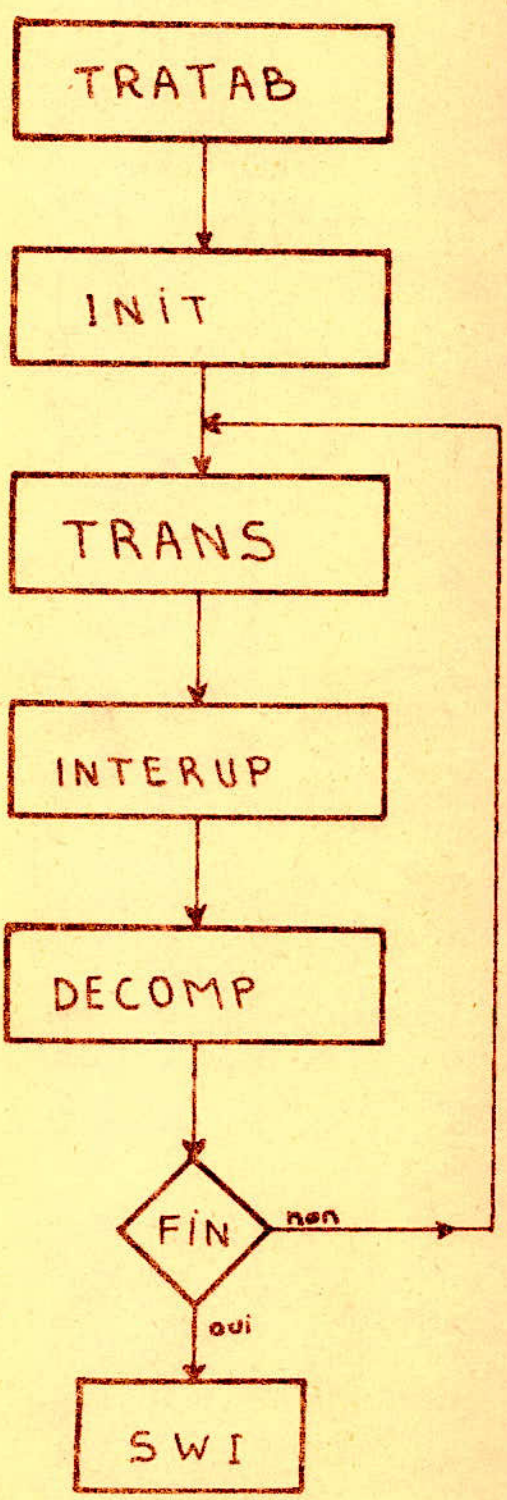
-=-=-oOo-=-=-

Organigramme de Transmission et Reception d'une table

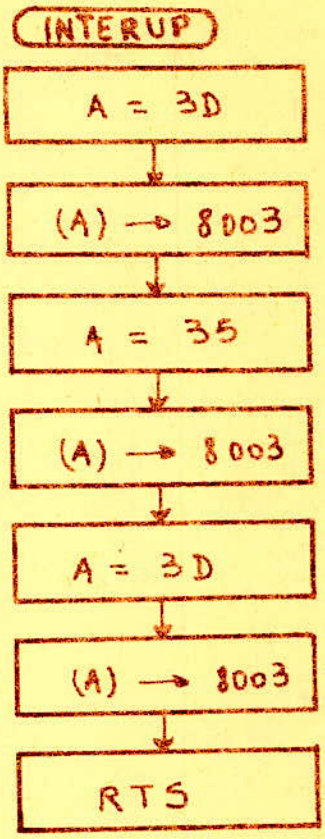
Reception d'une Table

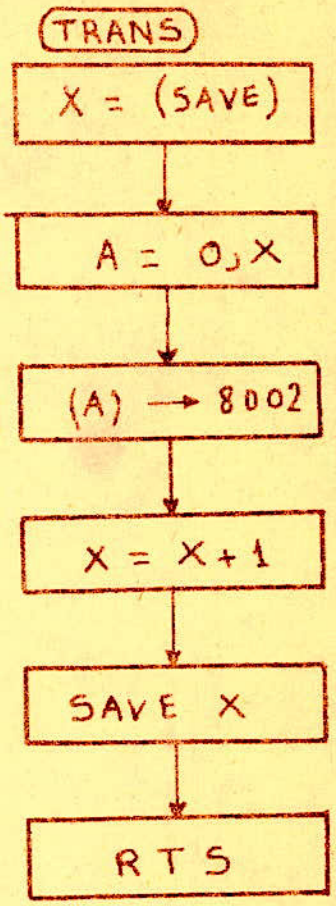


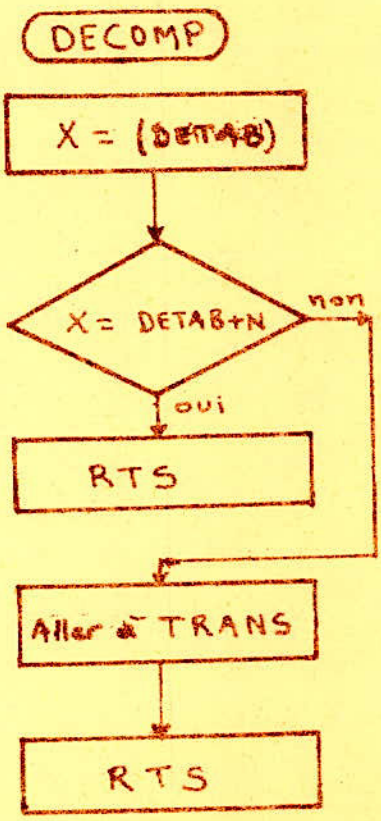
Transmission d'une Table

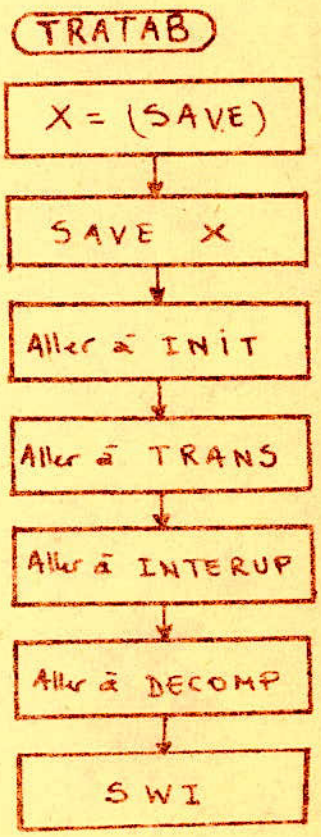


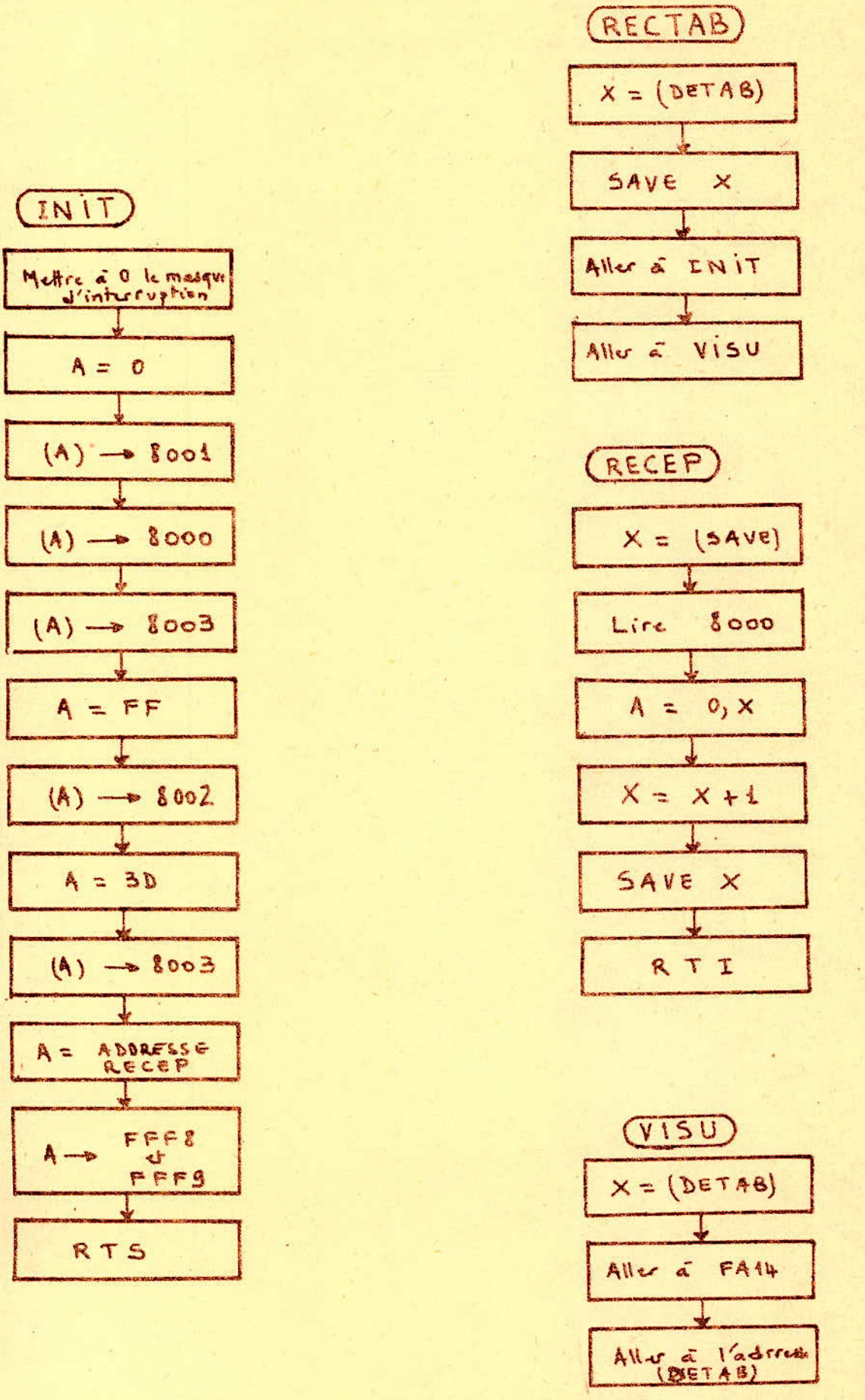
Sous programmes de Transmission d'une Table











Sous-programmes de Reception d'une Table

" Programme d'interruption avec saut au S/prog d'interrup "

--- S/programme d'initialisation:

```
PAGE  001   INITA

00010                          NAM     INITA
00020 0400                     ORG     $400
00030                          OPT     S
00040       8001    CRAPIA     EQU     $8001
00050       8000    DDRAPI     EQU     $8000
00060       8003    CRBPIA     EQU     $8003
00070       8002    DDRBPI     EQU     $8002
00080 0400 4F                  CLR A
00090 0401 B7 8001             STA A   CRAPIA    ACCES AU REG. DDRA
00100 0404 B7 8000             STA A   DDRAPIA   DECLARATION PA0-PA7 EN ENTRE
00110 0407 B7 8003             STA A   CRBPIA    ACCES AU REG. DDRB
00120 040A 86 FF               LDA A   #$FF
00130 040C B7 8002             STA A   DDRBPIA   DECLARATION PB0-PB7 EN SORTIE
00140 040F 86 3D               LDA A   #$3D
00150 0411 B7 8001             STA A   CRAPIA
00160 0414 B7 8003             STA A   CRBPIA
00170 0417 86 02               LDA A   #$2
00180 0419 B7 FFF8             STA A   $FFF8
00190 041C 86 00               LDA A   #$00
00200 041E B7 FFF9             STA A   $FFF9
00210 0421 39                  RTS
00220       0000               END
CRAPIA 8001 DDRAPI 8000 CRBPIA 8003 DDRBPI 8002
```

--- S/programme d'écriture "PROGRAMME":

```
PAGE  001   PROGA

00010                          NAM     PROGA
00020 0100                     ORG     $100
00030 0100 0F                  CLI
00040 0101 CE 010D             LDX     #TOTO
00050 0104 BD FA14             JSR     $FA14
00060 0107 86 FF               LDA A   #$FF
00070 0109 4A       BOUCLE     DEC A
00080 010A 26 FD               BNE     BOUCLE
00090 010C 39                  RTS
00100 010D 50       TOTO       FCC     /PROGRAMME/
00110 0116 04                  FCB     4
00120       0000               END
```

--- S/prog d'interruption : écriture "SOUS PROGRAMME":-

```
PAGE  001   PROGB

00010                          NAM     PROGB
00020 0200                     ORG     $200
00030       8000    ORAPIA     EQU     $8000
00040       8002    ORBPIA     EQU     $8002
00050 0200 B6 8000             LDA A   ORAPIA   LECTURE DU REG. ORA
00060 0203 B6 8002             LDA A   ORBPIA   LECTURE DU REG. ORB
00070 0206 0F                  SEI
00080 0207 CE 0218             LDX     #TITI
00090 020A BD FA14             JSR     $FA14
00100 020D 86 FF               LDA A   #$FF
00110 020F 4A          BOU     DEC A
00120 0210 26 FD               BNE     BOU
00130 0212 86 FF               LDA A   #$FF
00140 0214 4A          CLE     DEC A
00150 0215 26 FD               BNE     CLE
00160 0217 3B                  RTI
00170 0218 53          TITI    FCC     /SOUS-PROGRAMME/
00180 0226 04                  FCB     4
00190       0000               END
```

--- S/prog d'execution d'interruption utilisé par l'Esclave:

```
PAGE  001

00010 0500                     ORG     $500
00020                          NAM     INTERUP
00030       8001    CRAPIA     EQU     $8001
00040 0500 86 3D               LDA A   #$3D
00050 0502 B7 8001             STA A   CRAPIA
00060 0505 86 35               LDA A   #$35
00070 0507 B7 8001             STA A   CRAPIA
00080 050A 86 3D               LDA A   #$3D
00090 050C B7 8001             STA A   CRAPIA
00100 050F 39                  RTS
```

--- Programme Plan d'execution du Prog d'interrup avec Saut au S/prog d'interrup.

```
PAGE  001   PROGP

00010                          NAM     PROGP
00020 0300                     ORG     $300
00030       0400    INITA      EQU     $400
00040       0100    PROGA      EQU     $100
00050       0200    PROGB      EQU     $200
00060 0300 BD 0400             JSR     INITA
00070 0303 BD 0100 SIR         JSR     PROGA
00080 0306 7F 0303             JMP     SIR
00090       0000               END
```

--- S/prog utilisés par le maître

```
PAGE  001   INITA

00010                          NAM     INITA
00020 0400                     ORG     $400
00030                          OPT     S
00040       8001     CRAPIA    EQU     $8001
00050       8000     DDRAPI    EQU     $8000
00060       8003     CRBPIA    EQU     $8003
00070       8002     DDRBPI    EQU     $8002
00080 0400 4F                  CLR A
00090 0401 B7 8001             STA A   CRAPIA    ACCES AU REG. DDRA
00100 0404 B7 8000             STA A   DDRAPIA   DECLARATION PA0-PA7 EN ENTRE
00110 0407 B7 8003             STA A   CRBPIA    ACCES AU REG. DDRB
00120 040A 86 FF               LDA A   #$FF
00130 040C B7 8002             STA A   DDRBPIA   DECLARATION PB0-PB7 EN SORTIE
00140 040F 86 3D               LDA A   #$3D
00150 0411 B7 8001             STA A   CRAPIA
00160 0414 B7 8003             STA A   CRBPIA
00170 0417 86 04               LDA A   #$4
00180 0419 B7 FFF8             STA A   $FFF8
00190 041C 86 39               LDA A   #$39
00200 041E B7 FFF9             STA A   $FFF9
00210 0421 39                  RTS
00030 0422 0F                  CLI
00040 0423 CE 042F             LDX     #TOTO
00050 0426 BD FA14             JSR     $FA14
00060 0429 86 FF               LDA A   #$FF
00070 042B 4A          BOUCLE  DEC A
00080 042C 26 FD               BNE     BOUCLE
00090 042E 39                  RTS
00100 042F 50          TOTO    FCC     /PROGRAMME/
00110 0438 04                  FCB     4
00030       8000     ORAPIA    EQU     $8000
00040       8002     ORBPIA    EQU     $8002
00050 0439 B6 8000             LDA A   ORAPIA    LECTURE DU REG. ORA
00060 043C B6 8002             LDA A   ORBPIA    LECTURE DU REG. ORB
00070 043F 0F                  SEI
00080 0440 CE 0451             LDX     #TITI
00090 0443 BD FA14             JSR     $FA14
00100 0446 86 FF               LDA A   #$FF
00110 0448 4A          BOU     DEC A
00120 0449 26 FD               BNE     BOU
00130 044B 86 FF               LDA A   #$FF
00140 044D 4A          CLE     DEC A
00150 044E 26 FD               BNE     CLE
00160 0450 3B                  RTI
00170 0451 53          TITI    FCC     /SOUS-PROGRAMME/
00180 045F 04                  FCB     4
00030       0400     INITA     EQU     $400
00040       0422     PROGA     EQU     $422
00050       0439     PROGB     EQU     $439
00060 0460 BD 0400             JSR     INITA
00070 0463 BD 0422     SIR     JSR     PROGA
00080 0466 7E 0463             JMP     SIR
00090       0000               END
CRAPIA 8001 DDRAPI 8000 CRBPIA 8003 DDRBPI 8002 BOUCLE 042B
TOTO   042F ORAPIA 8000 ORBPIA 8002 BOU    0448 CLE    044D
TITI   0451 INITA  0400 PROGA  0422 PROGB  0439 SIR    0463
```

-Transmission/Reception d'un octet-

--- S/programme plan d'execution de transmission d'un octet:

```
PAGE  001

00010 0000                        ORG     $0
00020                             NAM     TRANOCT
00030       0100      INIT    EQU     $100
00040       0400      TRANS   EQU     $400
00050       0500      INTERU  EQU     $500
00070 0000 CE 1500            LDX     #$1500
00080 0003 FF 0600            STX     $600
00090 0006 BD 0100            JSR     INIT
00100 0009 BD 0400 ROI        JSR     TRANS
00110 000C BD 0500            JSR     INTERUP
00130 000F 3F                 SWI
```

--- S/programme de transmission d'un octet:

```
PAGE  001

00010 0400                        ORG     $400
00020                             NAM     TRANSO
00030       8000      ORAPIA  EQU     $8000
00040 0400 FE 0600            LDX     $600
00050 0403 A6 00              LDA A   0,X
00060 0405 B7 8000            STA A   ORAPIA
00070 0408 FF 0600            STX     $600
00080 040B 39                 RTS
00090       0000              END
```

--- S/programme d'execution d'une interruption:

```
PAGE  001

00010 0500                        ORG     $500
00020                             NAM     INTERUP
00030       8001      ORAPIA  EQU     $8001
00040 0500 86 3D              LDA A   #$3D
00050 0502 B7 8001            STA A   ORAPIA
00060 0505 86 35              LDA A   #$35
00070 0507 B7 8001            STA A   ORAPIA
00080 050A 86 3D              LDA A   #$3D
00090 050C B7 8001            STA A   ORAPIA
00100 050F 39                 RTS
```

--- Programme plan d'éxecution de reception d'un octet:

```
PAGE  001

00010 0900                        ORG     $900
00020                             NAM     RECEPOCT
00030       0100      INIT        EQU     $100
00040       0200      VISU        EQU     $200
00050 0900  CE 1000               LIX     #$1000
00060 0903  FF 0700               STX     $700
00070 0906  BI 0100               JSR     INIT
00080 0909  7E 0200               JMP     VISU
00090       0000                  END
```

--- S/programme de visualisation de l'octet transmis:

```
PAGE  001

00010 0200                        ORG     $200
00020                             NAM     VISU
00030 0200  0F                    CLI
00040 0201  CE 1000               LIX     #$1000
00050 0204  BI FA14               JSR     $FA14
00060 0207  7E 0200               JMP     $200
```

?

--- S/programme de reception d'un octet:

```
PAGE  001

00010 0300                        ORG     $300
00020                             NAM     RECEPO
00030       8000      OFAFIA      EQU     $8000
00040 0300  FF 0700               LIX     $700
00050 0303  F6 8000               LIA A   OFAFIA
00060 0306  A7 00                 STA A   0,X
00080 0308  39                    RTS
```

" - Transmission/Reception d'une Table - "

--- b/programme reception d'une table:

```
PAGE  001

00010 0300                        ORG     $300
00020                             NAM     RECEP
00030       8000     ORAPIA EQU     $8000
00040 0300 FE 0700                LDX     $700
00050 0303 B6 8000                LDA A   ORAPIA
00060 0306 A7 00                  STA A   0,X
00070 0308 7C 0701                INC     $701
00080 030B 39                     RTS
```

--- b/programme de transmission d'une table:

```
PAGE  001

00010 0400                        ORG     $400
00020                             NAM     TRANS
00030       8000     ORAPIA EQU     $8000
00040 0400 FE 0600                LDX     $600
00050 0403 A6 00                  LDA A   0,X
00060 0405 B7 8000                STA A   ORAPIA
00070 0408 08                     INX
00080 0409 FF 0600                STX     $600
00090 040C 39                     RTS
```

--- b/programme d'éxecution d'une interruption:

```
PAGE  001

00010 0500                        ORG     $500
00020                             NAM     INTERUP
00030       8001     CRAPIA EQU     $8001
00040 0500 86 3D                  LDA A   #$3D
00050 0502 B7 8001                STA A   CRAPIA
00060 0505 86 35                  LDA A   #$35
00070 0507 B7 8001                STA A   CRAPIA
00080 050A 86 3D                  LDA A   #$3D
00090 050C B7 8001                STA A   CRAPIA
00100 050F 39                     RTS
```

--- Propramme Plan d'éxecution d'une transmission de table

```
PAGE  001

00010 0000                      ORG     $0
00020                           NAM     TRATAB
00030        0100     INIT      EQU     $100
00040        0400     TRANS     EQU     $400
00050        0500     INTERU    EQU     $500
00060        0800     DECOMP    EQU     $800
00070 0000 CE 1500              LDX     #$1500
00080 0003 FF 0600              STX     $600
00090 0006 BD 0100              JSR     INIT
00100 0009 BD 0400 ROI          JSR     TRANS
00110 000C BD 0500              JSR     INTERUP
00120 000F BD 0800              JSR     DECOMP
00130 0012 3F                   SWI
```

--- S/programme d'initialisation pour Trans/Recep d'une table.

```
PAGE  001

00010 0100                      ORG     $100
00020                           NAM     INIT
00030        8001     CRAPIA    EQU     $8001
00040        8000     DDRAPI    EQU     $8000
00050        8003     CRBPIA    EQU     $8003
00060        8002     DDRBPI    EQU     $8002
00070 0100 4F                   CLR A
00080 0101 B7 8001              STA A   CRAPIA
00090 0104 B7 8000              STA A   DDRAPIA
00100 0107 B7 8003              STA A   CRBPIA
00110 010A 86 FF                LDA A   #$FF
00120 010C B7 8002              STA A   DDRBPIA
00130 010F 86 3D                LDA A   #$3D
00140 0111 B7 8001              STA A   CRAPIA
00150 0114 B7 8003              STA A   CRBPIA
00160 0117 0E                   CLI
00170 0118 86 03                LDA A   #$3
00180 011A B7 FFF8              STA A   $FFF8
00190 011D 86 00                LDA A   #$00
00200 011F B7 FFF9              STA A   $FFF9
00210 0122 39                   RTS
```

--- S/programme de Visualisation de la table transmise:

```
PAGE  001

00010 0200                      ORG     $200
00020                           NAM     VISU
00030 0200 0E                   CLI
00040 0201 CE 1000              LDX     #$1000
00050 0204 BD FA14              JSR     $FA14
00060 0207 7E 0200              JMP     $200
```

--- S/programme de décompte :

```
PAGE  001

00010 0800                      ORG     $800
00020                           NAM     DECOMP
00030       0009     ROI        EQU     $009
00040 0800 FE 0600              LDX     $600
00050 0803 8C 0096              CPX     #150F
00060 0806 26 01                BNE     BOU
00070 0808 39                   RTS
00080 0809 BD 0009 BOU          JSR     ROI
00090 080C 39                   RTS
00100       0000                END
```

--- Programme Plan d'exécution de reception Table.

```
PAGE  001

00010 0900                      ORG     $900
00020                           NAM     RECTAB
00030       0100     INIT       EQU     $100
00040       0200     VISU       EQU     $200
00050 0900 CE 1000              LDX     #$1000
00060 0903 FF 0700              STX     $700
00070 0906 BD 0100              JSR     INIT
00080 0909 7E 0200              JMP     VISU
00090       0000                END
```

--- S/programmes utilisés par le Maitre et les esclaves pour la Trans/Recep de table

PAGE 001

```
00010 0100                          ORG     $100
00020                               NAM     INIT
00030       8001     CRAPIA EQU     $8001
00040       8000     DDRAPI EQU     $8000
00050       8003     CRBPIA EQU     $8003
00060       8002     DDRBPI EQU     $8002
00070 0100 4F                       CLR A
00080 0101 B7 8001                  STA A   CRAPIA
00090 0104 B7 8000                  STA A   DDRAPIA
00100 0107 B7 8003                  STA A   CRBPIA
00110 010A 86 FF                    LDA A   #$FF
00120 010C B7 8002                  STA A   DDRBPIA
00130 010F 86 3D                    LDA A   #$3D
00140 0111 B7 8001                  STA A   CRAPIA
00150 0114 B7 8003                  STA A   CRBPIA
00160 0117 0E                       CLI
00170 0118 86 03                    LDA A   #$3
00180 011A B7 FFF8                  STA A   $FFF8
00190 011D 86 00                    LDA A   #$00
00200 011F B7 FFF9                  STA A   $FFF9
00210 0122 39                       RTS
00010 0200                          ORG     $200
00020                               NAM     VISU
00030 0200 0E                       CLI
00040 0201 CE 1000                  LDX     #$1000
00050 0204 BD FA14                  JSR     $FA14
00060 0207 7E 0200                  JMP     $200
00010 0300                          ORG     $300
00020                               NAM     RECEP
00030       8000     ORAPIA EQU     $8000
00040 0300 FE 0700                  LDX     $700
00050 0303 B6 8000                  LDA A   ORAPIA
00060 0306 A7 00                    STA A   0,X
00070 0308 7C 0701                  INC     $701
00080 030B 39                       RTS
00010 0400                          ORG     $400
00020                               NAM     TRANS
00040 0400 FE 0600                  LDX     $600
00050 0403 A6 00                    LDA A   0,X
00060 0405 B7 8000                  STA A   ORAPIA
00070 0408 08                       INX
00080 0409 FF 0600                  STX     $600
00090 040C 39                       RTS
00010 0500                          ORG     $500
00020                               NAM     INTERUP
00040 0500 86 3D                    LDA A   #$3D
00050 0502 B7 8001                  STA A   CRAPIA
00060 0505 86 35                    LDA A   #$35
00070 0507 B7 8001                  STA A   CRAPIA
00080 050A 86 3D                    LDA A   #$3D
00090 050C B7 8001                  STA A   CRAPIA
00100 050F 39                       RTS
00010 0800                          ORG     $800
00020                               NAM     DECOMP
00040 0800 FE 0600                  LDX     $600
00050 0803 8C 0096                  CPX     #150F
00060 0806 26 01                    BNE     BOU
00070 0808 39                       RTS
```

PAGE 002

```
00050 0809 BD 0009 BOU     JSR     ROI
00090 080C 39              RTS
00010 0000                 ORG     $0
00020                      NAM     TRATAB
00030          0100  INIT  EQU     $100
00040          0400  TRANS EQU     $400
00050          0500  INTERU EQU    $500
00060          0800  DECOMP EQU    $800
00070 0000 CE 1500         LDX     #$1500
00080 0003 FF 0600         STX     $600
00090 0006 BD 0100         JSR     INIT
00100 0009 BD 0400 ROI     JSR     TRANS
00110 080C BD 0500         JSR     INTERUP
00120 080F BD 0800         JSR     DECOMP
00130 0012 3F              SWI
00010 0900                 ORG     $900
00020                      NAM     RECTAB
00040          0200  VISU  EQU     $200
00050 0900 CE 1000         LDX     #$1000
00060 0903 FF 0700         STX     $700
00070 0906 BD 0100         JSR     INIT
00080 0909 7E 0200         JMP     VISU
00090          0000        END
```

CHAPITRE VI

GESTION DES INTERRUPTIONS

-=-=-=-=-oOo-=-=-=-=-

VI - 1 INTRODUCTION :

Comme le Maître doit gérer 16 esclaves, et qu'il peut être interrompu à n'importe quel moment par ces derniers, il ne saura plus quel est l'esclave qui le demande. Pour remédier à ce problème on a étudié un procédé qui fixera un ordre de priorité des 16 interruptions et qui fera en sorte que le Maître ne prendra en compte qu'une seule interruption à la fois (l'interruption la plus prioritaire).

La gestion de ces interruptions est assurée par un module capable de prendre en compte 32 interruptions, mais on ne travaillera qu'avec 16.

Ce module permet de répercuter toutes les demandes d'interruption vers le Maître en le mémorisant et en ne laissant passer que la plus prioritaire.

L'ordre de priorité que nous avons fixé dans notre cas est décroissant c'est-à-dire, l'interruption la plus prioritaire et celle venant de l'esclave "1" et la moins prioritaire celle venant de l'esclave "16".

VI - 2 PRINCIPE DE PRIORITE :

Comme l'ordre de priorité est fixé de 1 à 16, si par exemple l'esclave "3" envoit une interruption alors que le Maître exécute un sous-programme d'interruption dont l'esclave ayant le n° 3 (1, ou 2) aurait fait la demande, l'interruption 3 ne sera prise en compte que quant le Maître aura terminé l'éxécution du sous-programme d'interruption de l'esclave (1 ou 2). Par contre, si l'esclave "3" interrompt le Maître alors que celui-ci exécute un sous-programme venant d'un esclave dont le n° est supérieur à 3, le Maître sauvegarde le sous-programme qu'il exécute et prend en compte l'interruption de l'esclave n° 3. Ce principe est celui de l'encodeur de priorité (8214) d'où son utilisation dans notre module.

Pour avoir l'acquisition du 8214, par le MPU, on passe par un 8212 (Port d'entrée/sortie à 8 bits) qui joue le rôle de port d'entrée d'information issues de l'encodeur de priorité (8214) vers l'$\overline{IRQ}$ du MPU.

VI - 3 FONCTIONNEMENT

Quand une demande d'interruption est présentée au 8214, une interruption est envoyée au MPU via le 8212. Le 8214 encodera la demande à travers 3 bits (module 8) et transmet cette valeur au 8212, celle-ci va être stocker dans CSR du 8214 (pour MPU) pour être utilisée comme

référence de comparaison à toute autre demande d'interruption.

Après la reconnaissance de l'interruption générée, le MPU pokte le compteur de programme à l'adresse du sous-programme d'interruption désiré.

Pour qu'une demande du deuxième 8214 soit prise en compte, il faut que celui-ci soit sélectionné ce qui entraine la déselection du premier 8214 d'où la sortie ENGL du premier PICU (8214) = "1" ---→ l'entrée ETLG du deuxième (8214) = "1".

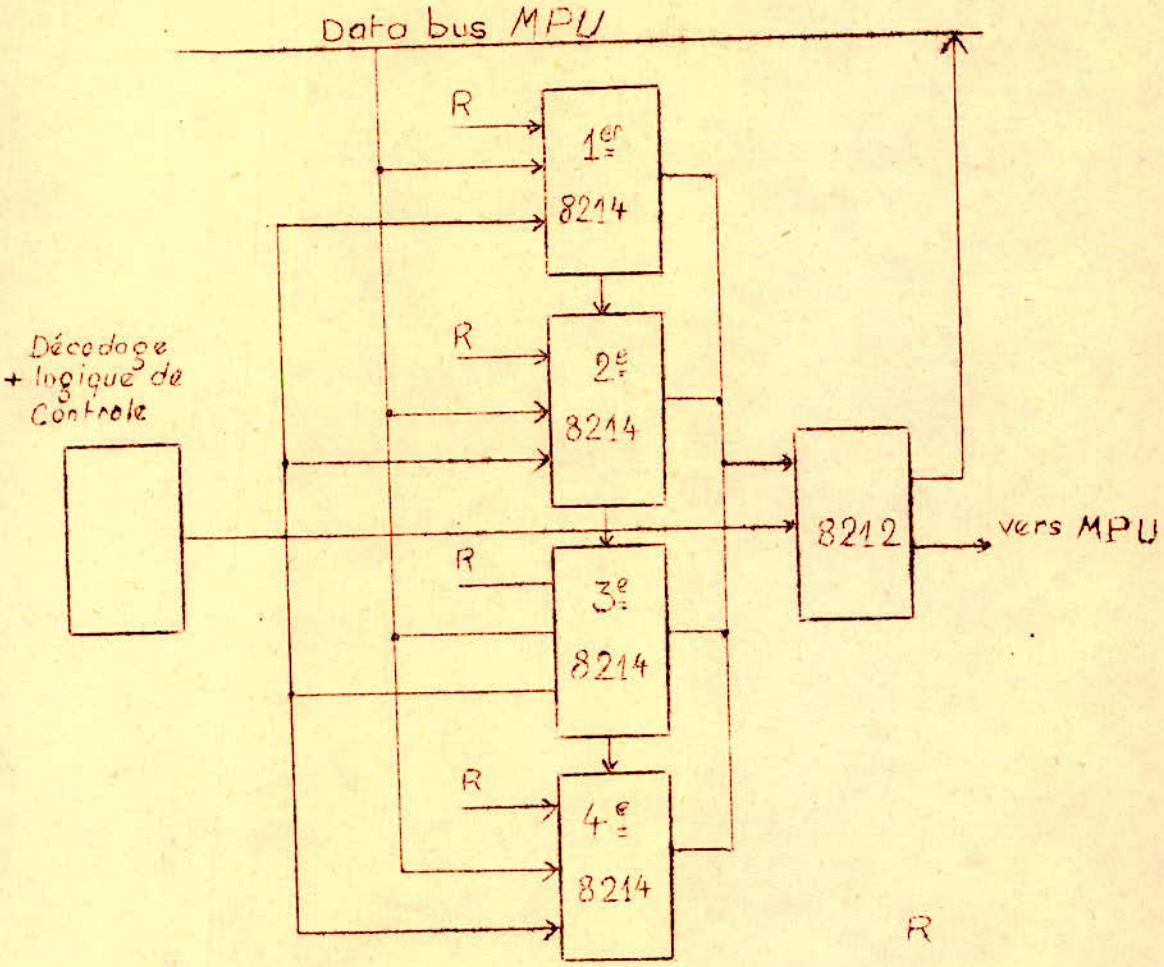
Les adresses des différents sous-programmes d'interruptions sont obtenues, après avoir complémentées, à partir du tableau suivant :

D_7	D_6	D_5	D_4	D_3	D_2	D_1	D_0	Code HEXA	Adresses desS/Prog. d'Interup.
0	0	0	0	1	0	0	0	08	F7
0	0	0	0	1	0	0	1	09	F6
0	0	0	0	1	0	1	0	0A	F5
0	0	0	0	1	0	1	1	0B	F4
0	0	0	0	1	1	0	0	0C	F3
0	0	0	0	1	1	0	1	0D	F2
0	0	0	0	1	1	1	0	0E	F1
0	0	0	0	1	1	1	1	0F	F0
0	0	0	1	0	0	0	0	10	EF
0	0	0	1	0	0	0	1	11	EE
0	0	0	1	0	0	1	0	12	ED
0	0	0	1	0	0	1	1	13	EC
0	0	0	1	0	1	0	0	14	EB
0	0	0	1	0	1	0	1	15	EA
0	0	0	1	0	1	1	0	16	E9
0	0	0	1	0	1	1	1	17	E8

VI - 4 REALISATION DE LA CARTE ENCODEUR DE PRIORITE

4. 1 - Schéma synoptique :

le schéma synoptique est donné à la page suivante :



4. 2 - Analyse des différents blocs :

4. 2. 1 - Port d'entrée/sortie à 8 bits (8212)

Le port d'entrée/sortie du 8212 se compose de huit bascules du type D et des buffers de sortie à trois états chacun avec contrôle et selection logique du circuit. Il comprend aussi une bascule SR (Service Request Flip-Flop) pour générer et contrôler les interruptions vers le microprocesseur.

Description du fonctionnement : Fig. C

Les 8 bascules qui composent ce circuit sont du type D. La sortie "Q" suivra la donnée "D" quand l'horloge C est à l'état haut "1". Le basculement aura lieu quand l'horloge retourne à l'état bas. La remise à zéro des bascules données se fait par un signal asynchrone $\overline{CLR}$. Les sorties "Q" de la bascule donnée sont connectées à des buffers de sorties à 3 états, non inverseuses. Ces buffers ont une ligne de contrôle commune EN, cette ligne soit elle active le buffer à transmettre la donnée venant des sorties "Q", soit le désactive en mettant la sortie à l'état haute-impédance.

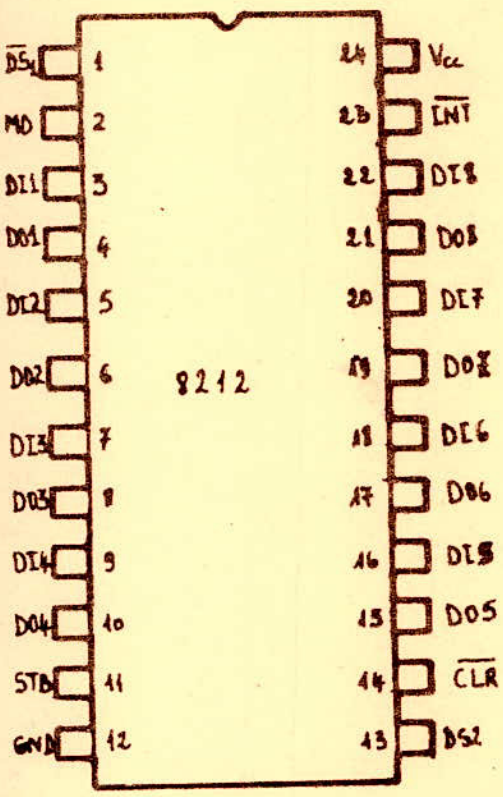


fig b - Brochage du 8212

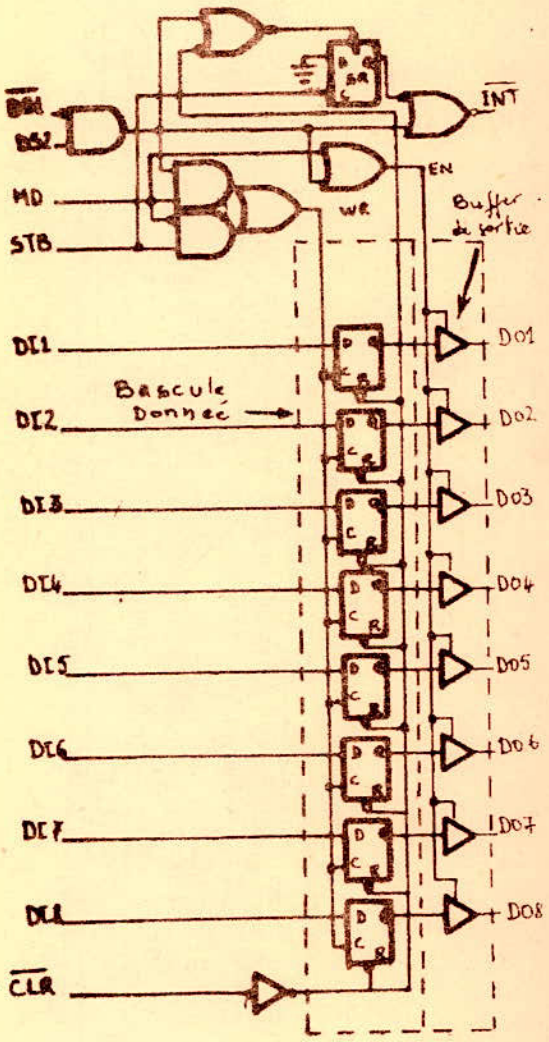


fig c - Configuration Interne du 8212

Contrôle logique

Le 8212 a des entrées de contrôle $\overline{DS_1}$, DS_2, MD et STB. Ces entrées sont utilisées pour contrôler la selection du circuit, les bascules données, l'état des buffers de sorties et la bascule SR Flip-Flop.

- 1 Entrées $\overline{DS_1}$ et DS_2 :

Ces deux entrées utilisées pour selectionner le circuit. Quand $\overline{DS_1}$ est bas et DS_2 est haut ($\overline{DS_1}$, DS_2 = 01)lz circuit est sélectionner. Dans cet état la sélection, les buffers de sortie sont activés et la bascule SR est mise à "1" d'une façon asychrone.

- 2 Entrée MD :

L'entrée MD est utilisée pour contrôler l'état des buffers de sortie et pour déterminer la source de l'horloge (C) de la bascule donnée.

Quant MD est haut (mode sortant) les buffers de sorties sont activés et la source de l'horloge "C" de la bascule donnée provient du circuit logique de selection ($\overline{DS_1}$, DS_2).

Quand MD = 0 (mode entrant) l'état des buffers de sortie est déterminé par le circuit logique de sélection ($\overline{DS_1}$-DS_2) et l'horloge "C" de la bascule donnée est l'entrée STB.

- 3 Entrée STB (strobe) :

Cette entrée est utilisée pour remettre à zéro la bascule SR.

- 4 Bascule SR Flip-Flop :

Cette bascule est utilisée pour générer et contrôler les interruptions dans le système micro-ordinateur. Elle est mise à "1" par l'entrée $\overline{CLR}$ d'une façon asynchrone. Quand SR est à "1" c'est l'état non-interruptible.

La sortie "Q" de la bascule SR est connectée à l'entrée inverseuse d'un NOR. L'autre entrée est non inversée et elle est connectée au circuit logique de sélection (DS_1 - DS_2). La sortie du NOR (INT) est activée à l'état bas et c'est l'état interruptible.

4. 2. 2 Encodeur de priorité 8214.

Le 8214 est un circuit de contrôle des priorités d'interruptions à 8 niveaux. Le PICU (Priority Interrupt Control Unit) peut accepter 8 demandes d'interruptions ; détermine la plus prioritaire, et compare cette

priorité à un software CSR et générer une interruption au système accompagné d'un vecteur d'interruption pour identifier le sous-programme spécifique.

Le PECU est désigné pour supporter une large variété de structure d'interruptions vectorisées et réduit le coût dans la gestion des interruptions.

Descriptions du fonctionnement : Fig a

1. Encodeur de priorité :

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

La logique de l'encodeur de priorité est telle que si deux ou plusieurs demandes d'interruptions arrivent simultanément alors c'est celle qui a la plus haute priorité qui sera prise en compte, et un code binaire (module 8), correspondant à la demande activée, sera envoyé. L'encodeur de priorité contient aussi une bascule pour stocker la demande.

2/ Current Stakes Register : CSR :

Dans la gestion des interruptions, l'importance n'est pas de rendre prioritaire l'arrivée d'une demande mais de constater si cette dernière est plus prioritaire que celle déjà servie.

Le CSR est une simple bascule à deux entrées qui est considérée comme un port adressable par le micro-ordinateur. Il est chargé quand $\overline{ECS}$ est à l'état bas.

Quant une interruption est envoyée au système, le programmeur (enregistreur) sort un code binaire (module 8) qui représente l'interruption activée. Cette valeur est stockée dans le CSR et est comparée à toute autre interruption par le comparateur de priorité.

Le code binaire de l'interruption courante est inscrit dans le CSR pourra être utilisé comme référence pour la comparaison.

Il n'y a pas de restriction pour le maintien, d'autres valeurs peuvent être écrites dans le CSR et servent comme référence.

Notons que la quatrième entrée $\overline{SGS}$ est une partie de la valeur écrite par le programmeur et remplie une fonction spéciale. Le comparateur de priorité mettra un "1" à la sortie qui indique que le niveau de la demande d'interruption est plus grande que celui du CSR.

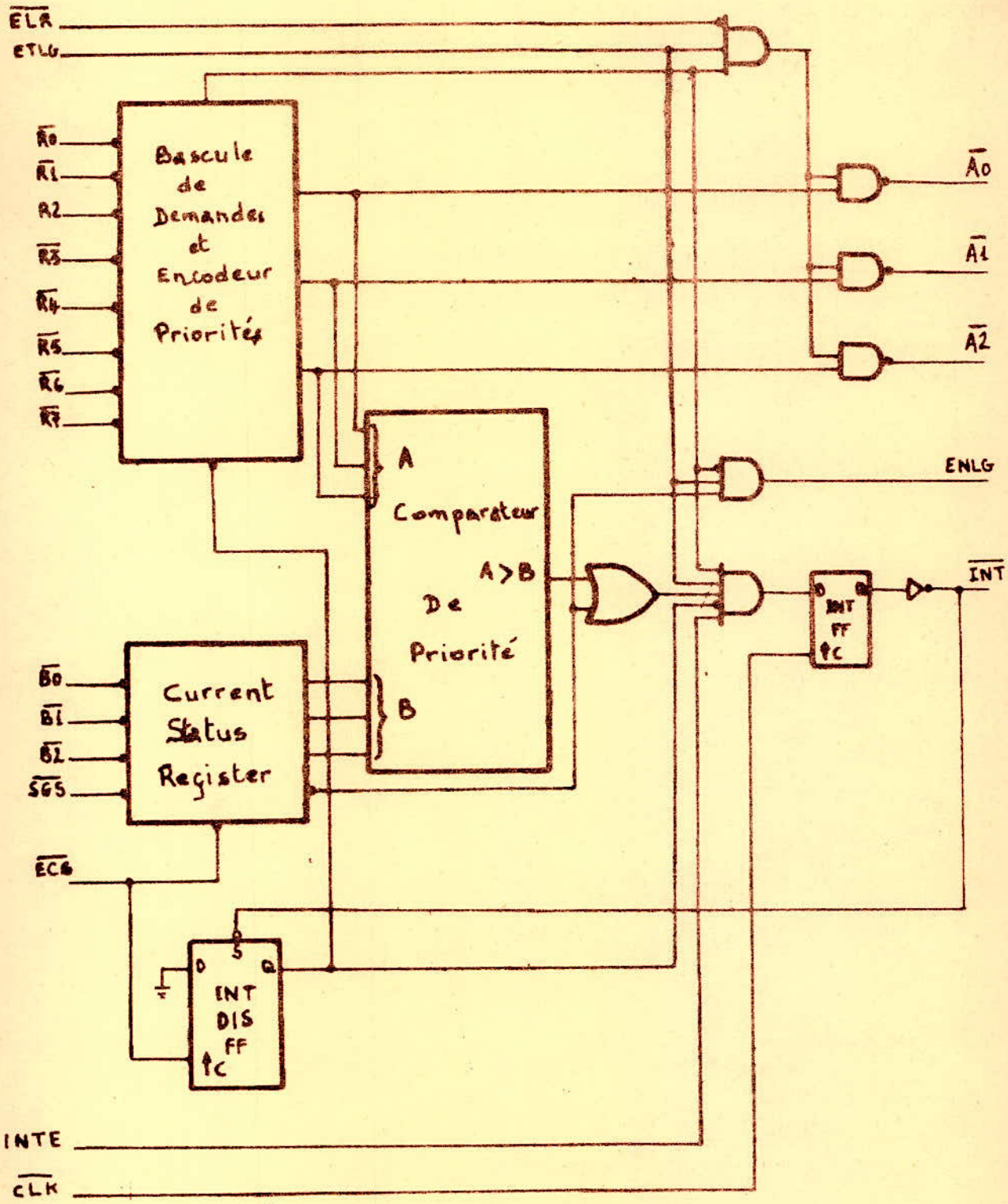


fig 8 - Configuration Interne du 8214

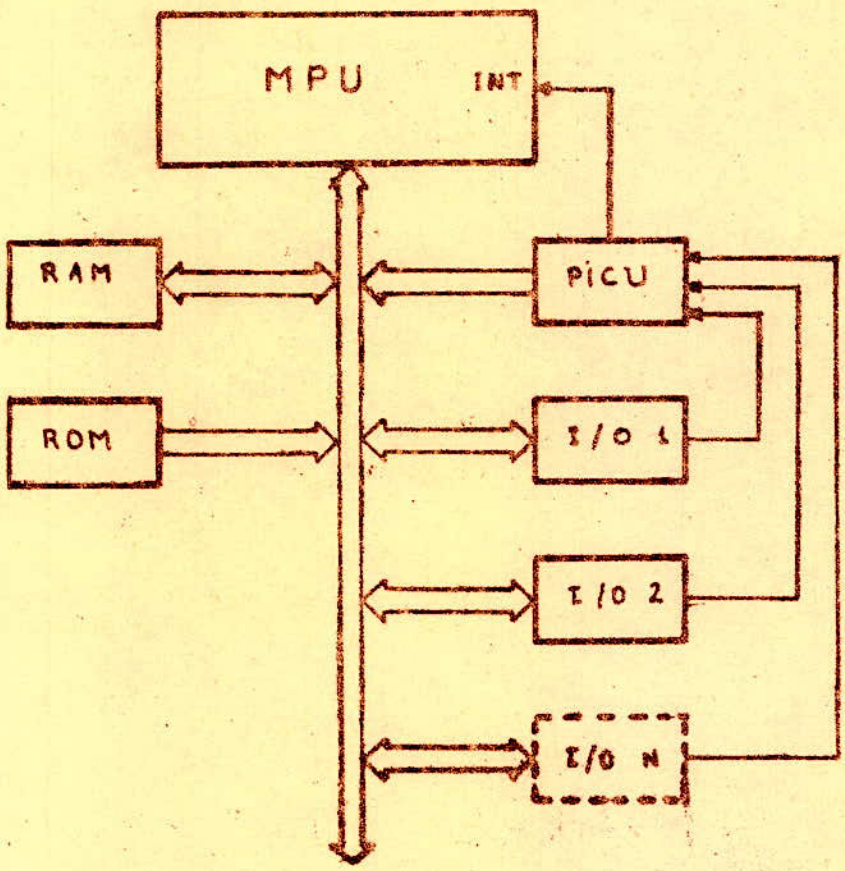


fig-c Interruptions Vectorisées

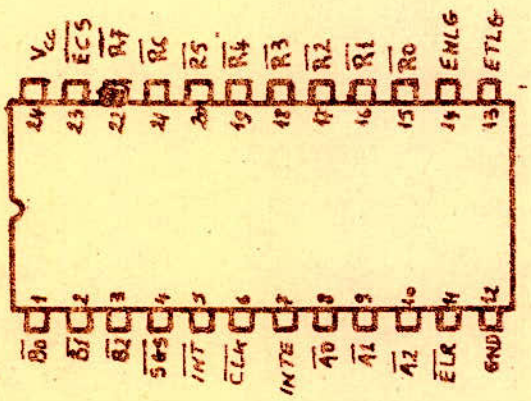


fig-d. Brochage du 8214

LE $\overline{SGS}$ autorise alors le programmeur de désactiver cette comparaison et permet au 8214 de générer une interruption au système qui est basée sur la logique de l'encodeur de priorité.

3. Signaux de contrôle

a - Entrées INTE et $\overline{CLK}$

Un zéro sur la ligne INTE n'autorise pas les interruptions d'arriver vers le microprocesseur. L'entrée $\overline{CLK}$ est actuellement celle qui déclenche la bascule INT FF, elle peut être connectée à l'une des horloges du microprocesseur.

b - Signaux $\overline{ELR}$, ETLG, ENGL

Ces trois signaux autorisent les 8214 d'être mis en cascade afin que plus de 8 demandes d'interruptions puissent être générées.

La sortie ENLG de l'un des 8214 est connectée à l'entrée du deuxième et ainsi de suite, avec la premier PICU ayant son ETLG mis à "1" et à la plus haute priorité. Quant la sortie ENLG est à "1" ça indique qu'il n'y a pas d'interruption à travers ce circuit mais elles peuvent être générées par le prochain 8214 qui a le moins de priorité.

6 - Lignes $\overline{A_0}$ - $\overline{A_1}$ - $\overline{A_2}$

Les sorties $\overline{A_0}$, $\overline{A_1}$, $\overline{A_2}$ représentent le complément du niveau de l'interruption courante (module 8). Par l'usage de ces signaux le compteur de programme peut pointer l'adresse du sous-programme spécifique.

d - Sortie $\overline{INT}$:

La sortie INT du 8214 est le signal qui génére une interruption au microprocesseur. Dès qu'une interruption est activée, la bascule INT dis FF est mise à 1, inibilant toute autre demande. C'est cette sortie qui est reliée au STB du 8212.

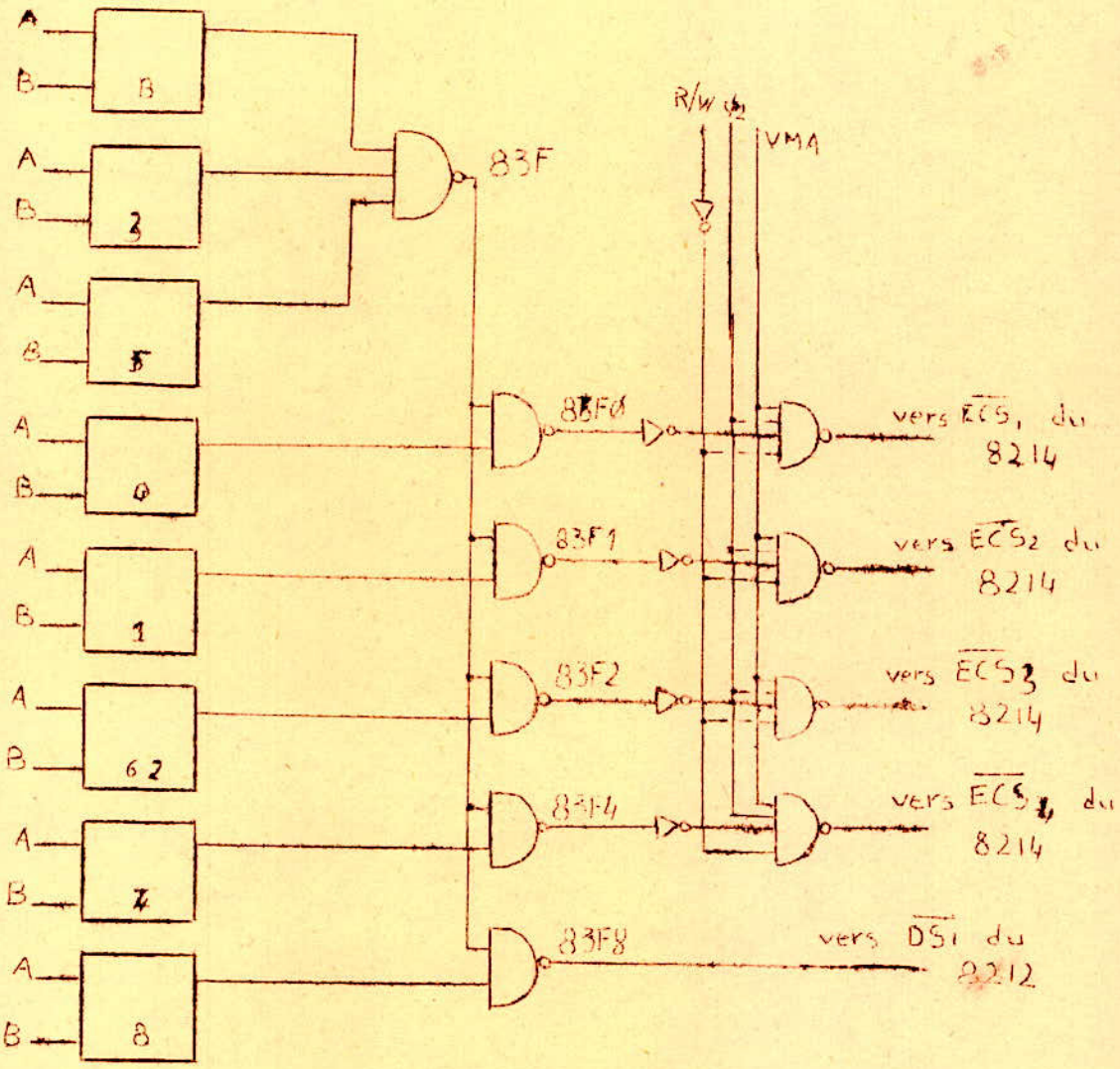
4.2. 3 Circuit de décodage :

Ce circuit est constitué de 8 comparateurs de bits identiques à ceux utilisés pour la carte PIA.

4.2. 4 Circuit logique de contrôle :

Il se compose de 8 portes NAND et de 6 inverseurs

Voir schéma à la page suivante :



4. 3 - Adressage des 8214 et 8212 :

Les adresses que nous avons fixées par les comparateurs sont :

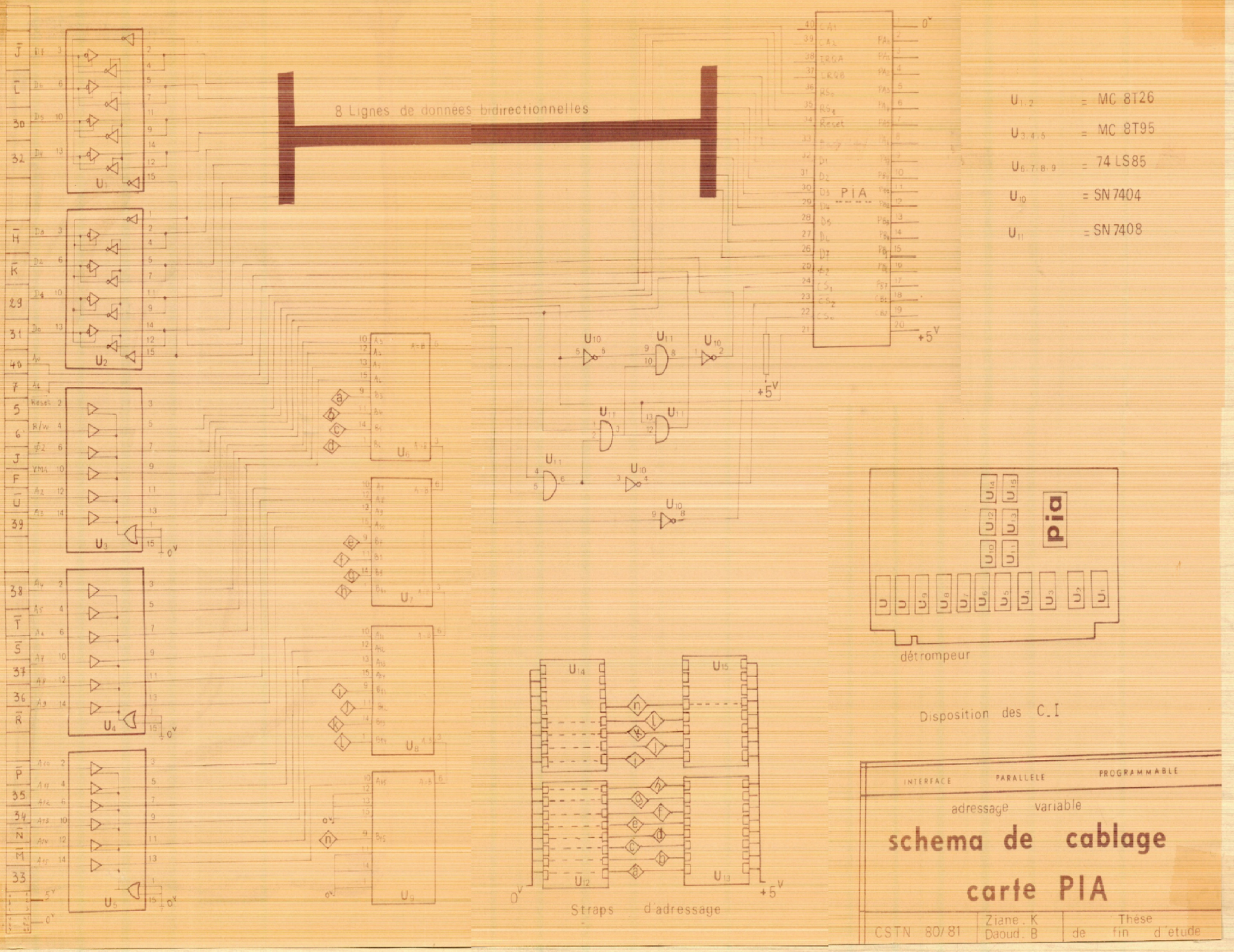
83F0 Pour le 1er 8214
83F1 Pour le 2ème 8214
83F2 Pour le 3ème 8214
83F4 Pour le 4ème 8214
et 83F8 Pour le 8212

Le choix de ces adresses successives des 4 8214, nous a permit de simplifier le circuit de décodage.

On présente dans le tableau suivant les lignes d'adresses des 4 encodeurs de priorité (8214) et du port d'entrée/sortie à 8 bits (8212).

4.3. 1 Tableau d'adressage :

Les 12 premières lignes du bus d'adresse restent inchangées, ce qui nous permet d'utiliser trois comparateurs au lieu de 16. Seules les 4 dernières lignes A12, A_{13}, A_{14}, A15 changent, et c'est ce qui nous permet de selectionner l'un ou l'autre des 4 encodeurs de priorité (8214) ou du 8212.



$U_{1,2}$	= MC 8T26
$U_{3,4,5}$	= MC 8T95
$U_{6,7,8,9}$	= 74 LS85
U_{10}	= SN 7404
U_{11}	= SN 7408

Disposition des C.I

INTERFACE PARALLELE PROGRAMMABLE

adressage variable

schema de cablage carte PIA

CSTN 80/81	Ziane. K Daoud. B	These de fin d'etude

TABLEAU :

A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	Code Hexa
1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	83 F0 1° 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	0	0	1	83 F1 2° 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	0	1	0	83 F2 3° 8214
1	0	0	0	0	0	1	1	1	1	1	1	0	1	0	0	83 F4 4° 8214
1	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0	83 F8 8212

VI - 5 PROGRAMMATION

- Comme on ne dispose pas de 16 exorcisers pour pouvoir tester la gestion des priorités d'interruptions des 16 esclaves, on les a assimilés à 16 boutons poussoirs.

Ces boutons poussoires jouent le rôle d'esclave. A chaque fois qu'ils sont mis à l'état bas, c'est comme si l'$\overline{IRQ}$ de l'esclave considéré est activé et le maître est interrompu suivant la priorité fixée.

- Afin de pouvoir vérifier si la priorité fixée fonctionne bien on a tracé un programme dont le principe est le suivant :

- Le Maître déroule un programme principal (écriture PROGRAMME) et des dès qu'il est interrompu (par l'un des boutons poussoires) il saute au sous-programme d'interruption qui est : écriture "INTERRUPTION" n° i"ACTIVEE" sur visu (i étant le n° du B.P. qui a interrompu le Maître). Une fois l'éxécution du sous-programme d'interrup. termineé il retourne au point où il a quitté son programme principal et continue son traitement.

Notons que si le Maître exécute un sous-programme d'interrup. et qu'un autre B.P. de moindre priorité est activé, le Maître n'en tiendra compte que quand il aura terminé la tâche qu'il assume, par contre si un B.P. de plus haute priorité est activé, le Maître saute au sous-programme d'interrup. du B.P. prioritaire après avoir sauvegarder ce qu'il entreprenait.

5.1. - Sous-programme utilisé :

1 - INITC Initialisation des 8212 et 8214
2 - PROGA Prog. Principal écriture "PROGRAMME" sur visu
3 - IDETI sous-prog.d'identification de l'interrup.envoyée.
4 - PROGB sous-prog. d'interrup.écriture"INTERRUP.n°1 ACTIVEE"
5 - ECRIT Sous-prog. d'entrée caractère.

5-2 - Programme de Gestion des interruptions -

--- S/programme d'identification :

```
PAGE  001   IIENTI

0001Ø                              NAM     IIENTI
00050 0300                         ORG     $300
00060 0300 86 00                   LDA A   #$0
00070 0302 B7 3000                 STA A   $3000
00080 0305 B6 83F8                 LDA A   $83F8
00090 0308 B7 3001                 STA A   $3001
00100 030B FE 3000                 LDX     $3000
00110 030F 9C F7                   CPX     $F7
00120 0310 27 08                   BEQ     BOU1
00130 0312 9C F6                   CPX     $F6
00140 0314 27 07                   BEQ     BOU2
00150                    *
00160                    *
00170                    *
00180                    *
00190 0316 9C F8                   CPX     $F8
00200 0318 27 06                   BEQ     BOU16
00210 031A 7E 0450 BOU1            JMP     $450
00220 031D 7E 0500 BOU2            JMP     $500
00230 0320 7E 0C00 BOU16           JMP     $C00
00240      0000                    FNI
```

--- S/programme d'interruption : écriture " INTERRUPTION N° i ACTIVÉE " :

```
PAGE  001   PROGI

00010                              NAM     PROGI
00020 0450                         ORG     $450
00030 0450 86 F7                   LDA A   #$F7
00040 0452 B7 83F0                 STA A   $83F0
00050 0455 CE 1388                 LDX     #5000
00060 0458 BD FA14                 JSR     $FA14
00070 045B 3B                      RTI
00080 0500                         ORG     $500
00090 0500 86 F6                   LDA A   #$F6
00100 0502 B7 83F0                 STA A   $83F0
00110 0505 53                      FCC     /SOUS-PROGRAMME ASSOCIE/
00120                    *
00130                    *
00140                    *
00150                    *
00160 0C00                         ORG     $C00
00170 0C00 96 F8                   LDA A   $F8
00180 0C02 B7 83F0                 STA A   $83F0
00190 0C05 53                      FCC     /SOUS-PROGRAMME ASSOCIE/
00200      0000                    FNI
```

--- S/programme d'entrée caractères :

```
PAGE  001   ECRIT

00010                          NAM     ECRIT
00020 0100                     ORG     $100
00030 0100 CE 5000             LDX     #$5000
00040 0103 BD FA7F             JSR     $FA7F
00050 0106 A7 00               STA A   0,X
00060 0108 08                  INX
00070 0109 20 F8               BRA     $103
00080 010B 06                  TAP
00090      0000                ENI
```

100;C INTERUPTION N1 ACTIVEE

INTERUPTION N2 ACTIVEE

.

.

.

INTERUPTION N16 ACTIVEE

--- S/programmes d'initialisation et écriture "PROGRAMME" :

```
PAGE  001   INITC

00010                          NAM     INITC
00020 0200                     ORG     $200
00030 0200 86 00               LDA A   #$00
00040 0202 B7 83F0             STA A   $83F0
00050 0205 86 03               LDA A   #$3
00060 0207 B7 FFF8             STA A   $FFF8
00070 020A 86 00               LDA A   #$00
00080 020C B7 FFF9             STA A   $FFF9
00090 020F 86 00               LDA A   #$00
00100 0211 B7 83F1             STA A   $83F1
00110 0214 0E                  CLI
00120 0215 CE 021E             LDX     #PRO
00130 0218 BD FA14             JSR     $FA14
00140 021B 7F 0200             JMP     $200
00150 021E 50          PRO     FCC     /PROGRAMME/
00160 0227 04                  FCB     4
00170      0000                END
```

// O N C L U S I O N

/)/ous dirons en conclusion que le but de notre projet etait de faire dialoguer 2 ou plusieurs micro-ordinateurs (par transfert de tables).

Quand au moniteur (programme que doit gerer le maitre pour faire dialoguer les esclaves), c'est une partie qui fait l'objet d'un autre projet de fin d'études et qui est en cours de réalisation.

L'interface réalisée, à "adressage variable" présente l'avantage d'être utilisable pour l'adaptation d'autres peripheriques (à transmission paralléle) au systeme .

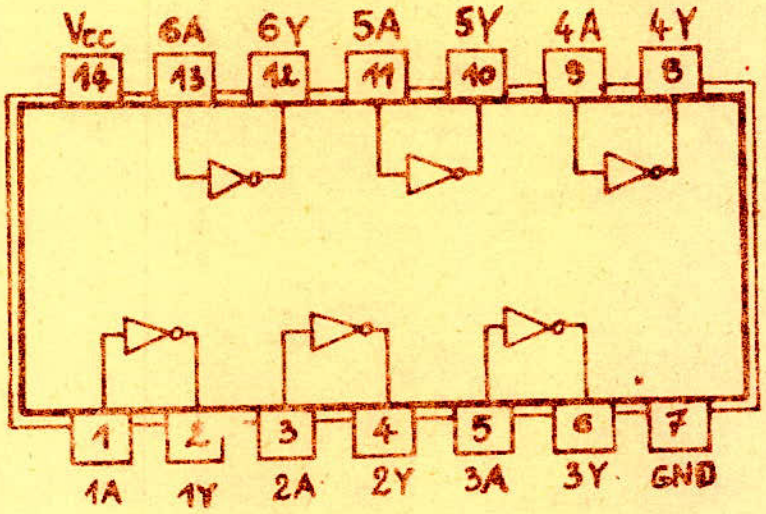
Les résultats obtenus font du systeme un outil de développement plus puissant dont l'utilisation est plus aisée

Enfin nous dirons également que ce projet nous a permis d'assimiler le fonctionnement d'un Exorcieser et d'une gamme de périphérique et de maitriser de ce fait des techniques d'interfaces entre un micro-ordinateur et un systéme d'entrée-sorties.

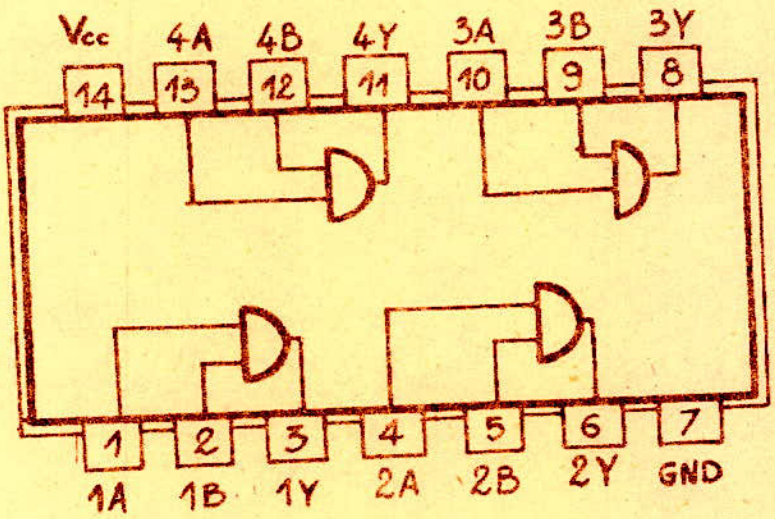
ANNEXE.

CODE ASCII

LS caractère b_4	b_3	b_2	b_1	MS caractère b_7 b_6 b_5	000	001	010	011	100	101	110	111
					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	EDT	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	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

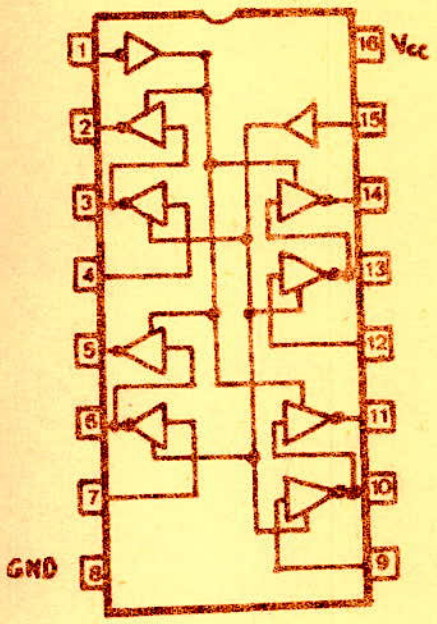


SN 7404 = 6 Inverseurs



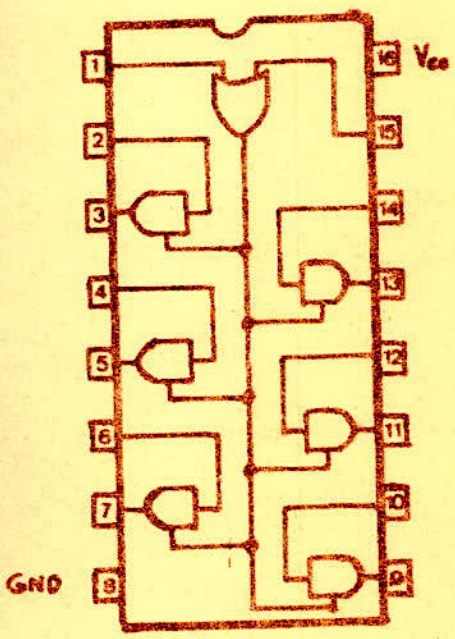
SN 7408 = 4 Portes "AND"

Brochage des C.I:



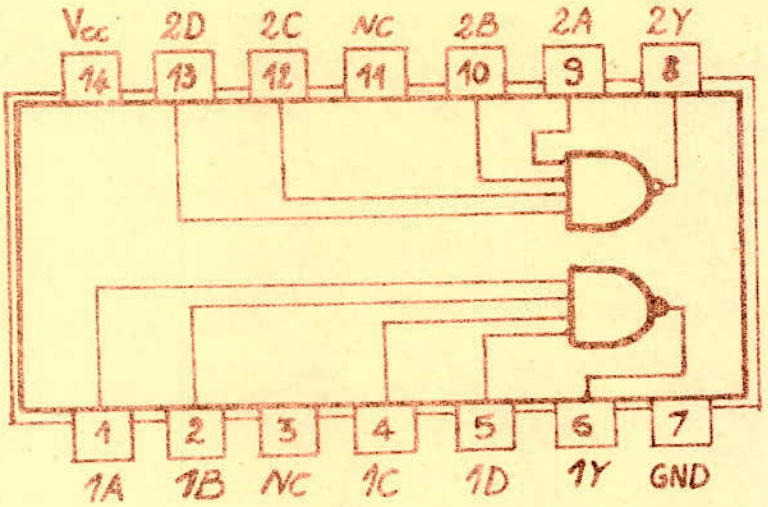
1. Receiver enable input
2. Receiver output 1
3. Bus 1
4. Driver Input 1
5. Receiver output 2
6. Bus 2
7. Driver input 2
8. GND
9. Driver input 3
10. Bus 3
11. Receiver output 3
12. Driver input 4
13. Bus 4
14. Receiver output 4
15. Driver enable input
16. Vcc

MC.8T26

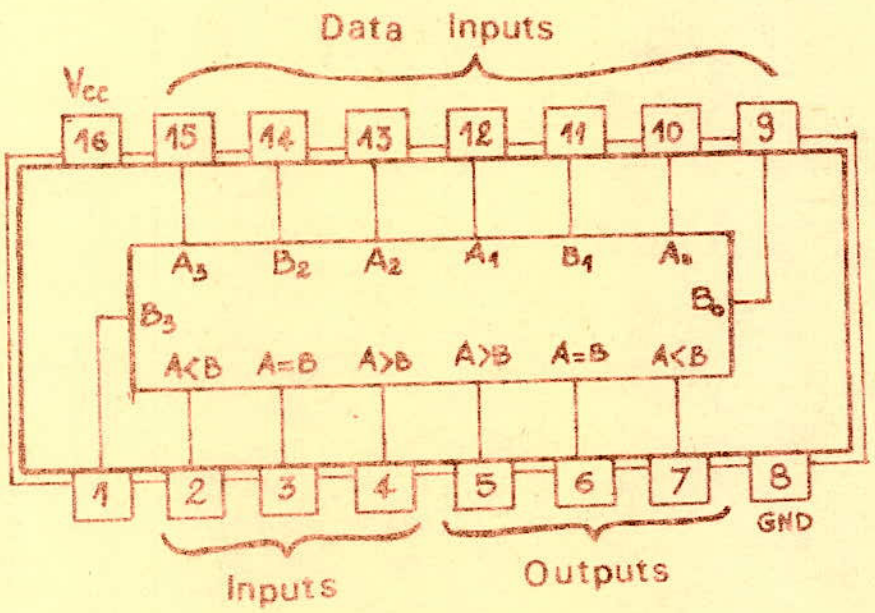


1. Enable 1
2. Input A
3. Output A
4. Input B
5. Output B
6. Input C
7. Output C
8. GND
9. Output D
10. Input D
11. Output E
12. Input E
13. Output F
14. Input F
15. Enable 2
16. Vcc

MC.8T95



SN 7420 = 2 Portes "NAND"



SN 74LS85 = COMPARATEUR

Tableau : 2

ACCUMULATOR AND MEMORY OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	BOOLEAN/ARITHMETIC OPERATION (All register labels refer to contents)	COND. CODE REG. H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	5	2	BB	4	3				A + M → A	↕	•	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	5	2	FB	4	3				B + M → B	↕	•	↕	↕	↕	↕
Add Acmltrs	ABA													1B	2	1	A + B → A	↕	•	↕	↕	↕	↕
Add with Carry	ADCA	89	2	2	99	3	2	A9	5	2	B9	4	3				A + M + C → A	↕	•	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	5	2	F9	4	3				B + M + C → B	↕	•	↕	↕	↕	↕
And	ANDA	84	2	2	94	3	2	A4	5	2	B4	4	3				A · M → A	•	•	↕	↕	R	•
	ANDB	C4	2	2	D4	3	2	E4	5	2	F4	4	3				B · M → B	•	•	↕	↕	R	•
Bit Test	BITA	85	2	2	95	3	2	A5	5	2	B5	4	3				A · M	•	•	↕	↕	R	•
	BITB	C5	2	2	D5	3	2	E5	5	2	F5	4	3				B · M	•	•	↕	↕	R	•
Clear	CLR							6F	7	2	7F	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	•	•	↕	↕	⑤	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	⑤	•
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	•	•	↕	↕	⑤	↕
	ROLA													49	2	1	A	•	•	↕	↕	⑤	↕
	ROLB													59	2	1	B — C b_7 … b_0	•	•	↕	↕	⑤	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	⑤	↕
	RORA													46	2	1	A	•	•	↕	↕	⑤	↕
	RORB													56	2	1	B — C b_7 … b_0	•	•	↕	↕	⑤	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	⑤	↕
	ASLA													48	2	1	A	•	•	↕	↕	⑤	↕
	ASLB													58	2	1	B — C b_7 … b_0 0	•	•	↕	↕	⑤	↕
Shift Right, Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	⑤	↕
	ASRA													47	2	1	A	•	•	↕	↕	⑤	↕
	ASRB													57	2	1	B — b_7 … b_0 C	•	•	↕	↕	⑤	↕
Shift Right, Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	⑤	↕
	LSRA													44	2	1	A	•	•	R	↕	⑤	↕
	LSRB													54	2	1	B — 0 b_7 … b_0 C	•	•	R	↕	⑤	↕
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	6	2	B0	4	3				A − M → A	•	•	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	6	2	F0	4	3				B − M → B	•	•	↕	↕	↕	↕
Subract 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

Code d'instruction du 6800 (1)

Tableau : 3

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

JUMP AND BRANCH OPERATIONS	MNEMONIC	RELATIVE OP	RELATIVE ~	RELATIVE #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	INHER OP	INHER ~	INHER #	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										01	2	1	Ad Prog Cntr. Only	•	•	•	•	•	•
Return From Interrupt	RTI										3B	10	1	}	⑩					
Return From Subroutine	RTS										39	5	1	} See special Operation	•	•	•	•	•	•
Software Interrupt	SWI										3F	12	1	}	•	S	•	•	•	•
Wait for Interrupt	WAI										3E	9	1	}	⑪	•	•	•	•	•

CONDITION CODE REGISTER OPERATIONS	MNEMONIC	INHER OP	INHER ~	INHER #	BOOLEAN OPERATION	5 N	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 → Acmlt A	TPA	07	2	1	$CCR \rightarrow A$	•	•	•	•	•	•

LEGEND

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

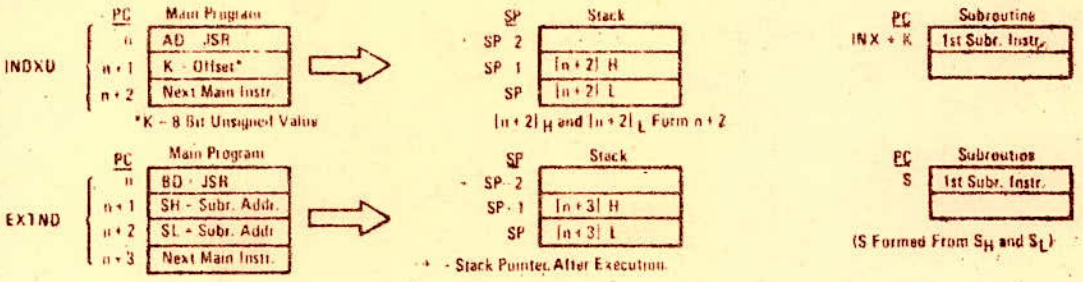
CONDITION CODE REGISTER NOTES

(Bit set if test is true and cleared otherwise)

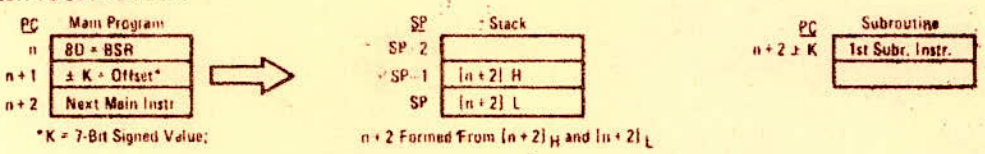
① (Bit V) Test Result = 10000000 ?
② (Bit C) Test Result = 00000000 ?
③ (Bit C) Test Decimal value of most significant BCD Character greater than nine ? (Not cleared if previously set)
④ (Bit V) Test Operand = 10000000 prior to execution ?
⑤ (Bit V) Test Operand = 01111111 prior to execution ?
⑥ (BIT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

SPECIAL OPERATIONS

JSR, JUMP TO SUBROUTINE:

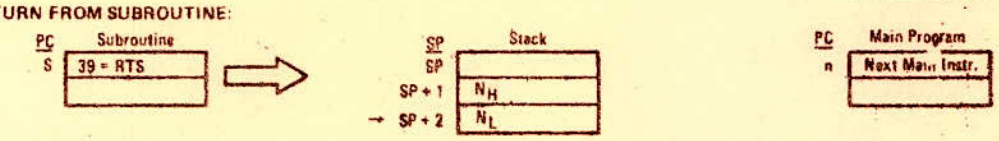


BSR, BRANCH TO SUBROUTINE:



JMP, JUMP:

RTS, RETURN FROM SUBROUTINE:



RTI, RETURN FROM INTERRUPT:

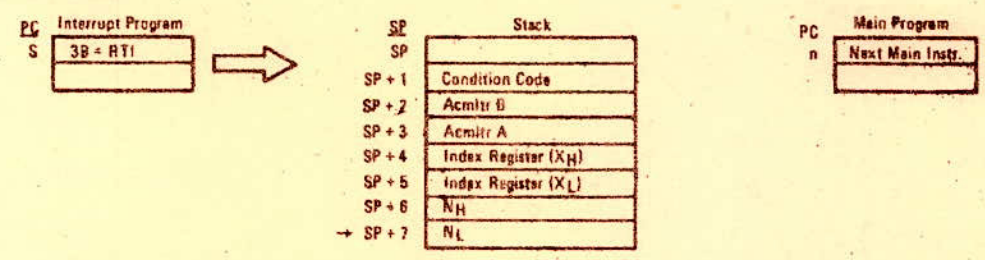


TABLE 6 – CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

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

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

- L'EMPLOI DES MICRO-PROCESSEURS (AUMIAUX)

- INTRODUCTION AU MICRO PROCESSEURS ET MICRO ODINATEURS (PARIOT)

- PROGRAMMATION DE MICRO PROCESSEURS (H-LILEN)

REVUES/

- MICROSYSTEME N°5

- MICROSYSTEME N°4

- L'ONDE ELECTRIQUE N°3

- ELECTRONIQUE APPLICATION N°8

BROCHURES/

- INTEL 8080 MICRO COMPUTER SYSTEM USER'S MANUEL.

- 6800 MICRO PROGRAMMING FOR LOGIC DESIGN (OSBORNE).