الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

CONTRIBUTION A LA CONCEPTION ET A LA REALISATION D'UN MICROORDINATEUR DE TEST RAPIDES DE TRAITEMENT

Proposé par : Mr BESSALAH Hamid

Etudié par : Mr ZENATI Abdelkader
Mr MENACER Mohamed

Dirigé par : Mr BESSALAH Hamid

PROMOTION : JANVIER 85

E.N.P. 10, Avenue Hacen Badi – EL-HARRACH – ALGER

الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

وزارة التعليم و البحث العلمي
MINISTERE DE L'ENSEIGNEMENT ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT ELECTRONIQUE

PROJET DE FIN D'ETUDES

SUJET

CONTRIBUTION A LA CONCEPTION ET A LA REALISATION D'UN MICROORDINATEUR DE TEST RAPIDES DE TRAITEMENT

Proposé par : Mr. BESSALAH Hamid

Etudié par : Mr ZENATI Abdelkader
Mr MENACER Mohamed

Dirigé par : Mr BESSALAH Hamid

PROMOTION : JANVIER 85

E.N.P. 10, Avenue Hacen Badi – EL-HARRACH – ALGER

بسم الله الرحمن الرحيم

يَا أَيُّهَا الَّذِينَ آمَنُوا إِذَا قِيلَ لَكُمْ تَفَسَّحُوا
فِي الْمَجَالِسِ فَافْسَحُوا يَفْسَحِ اللَّهُ لَكُمْ
وَإِذَا قِيلَ انْشُزُوا فَانْشُزُوا

يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ
وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ

وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ

المجادلة ١١

صدق الله العظيم

* UN SUCCES N'EST JAMAIS DEFINITIF , ET UN ECHEC N'EST JAMAIS FATAL , SEUL COMPTE LE COURAGE .

* NUL NE POSSEDE D'AUTRE DROIT QUE CELUI DE TOUJOURS FAIRE SON DEVOIR .

... DEDICACES ...

أهدي هذا المشروع المتواضع مع ازكى التحيات .
إلى اعز الناس علي ، إلى أمي وأختي وإخوتي وإلى
خالي حمود وخالي أحسن وخالي آسماعيل ولد انسي
لهم الجميل الذي شجعني في مرحلة دراستي .
وادام الله الخير وادام فاعلوه . واجر الجميع على الله.

- A ma mere , que j'admire pour son courage .
- A mes oncles , HAMMOUD ? HCENE , SMAIL , qui ont été toujours pret de nous quand on a eu besoin d'eux .
- A ma soeur et mes freres .
- A mes cousins et cousines . N.Z.N.Z.A.L.A.M.K.W.S.
- A tous mes amis sinceres .

M . M E N A C E R

- A mes parents .
- A mes freres et soeurs .
- A mes cousins et cousines .
- A toute ma famille .
- A tous mes amis(es) .

A . Z E N A T I

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

Nous remerçions vivement notre promoteur Mr BESSALAH pour nous avoir bien suivi et et mis à notre disposition tout les moyens necessaire pour notre projet .

Nos remerciements s'adressent aussi à toute l'equipe du laboratoire Architecture des systemes :

MM. S.MENACER , M.KARABERNOU , M.MIHOUBI , M.BOUZID

M.HAMDA , N.REZZOUG , M.LAMALI , Z.ZOUAOUI , M.KRIBES

F.MOUZALI ,

Notre profonde gratitude va egalement à Melle HAMMAD F ZOHRA qui a bien voulu dactylographier cet ouvrage .

S O M M A I R E

* SAVOIR QUE L'ON SAIT CE QUE L'ON SAIT , ET SAVOIR QUE L'ON NE SAIT PAS CE QUE L'ON NE SAIT PAS , VOILA LA VERITABLE SCIENCE .

INTRODUCTION :

L'un des objectifs du laboratoire architecture des systemes du CEN consiste en la conception et la realisation d'un systeme rapide de traitement du signal capable d'effec--tuer à une frequence eleveè un tres grand nombre de trait--ements lies à l'analyse spectrale,au filtrage numerique et à l'interpretation des images .

La realisation d'un systeme d'une telle envergure fait appel à des moyens de developpements,de mesure et de test tres importants et parfois tres couteux .

C'est dans ce contexte qu'il nous a etè confie la realisation d'un microordinateur orienté vers la generation de toutes les fonction implanteès dans le systeme rapide citè ci haut .

La conception et la realisation du microordinateur aeté faite autours d'un microprocesseur du type MOTOROLA MC 6809 à qui nous avons adjoint un clavier et des afficheurs .

La gestion du microprocesseur et de ces peripheriques est effectueé par un moniteur de gestion que nous avons ela--boré et implanté dans une EPROM de 1K . Le logiciel d'app--lication comporte les programmes suivants :

- calcul de l'operation papillon
- generation de donneés pseudo aleatoire
- produit scalaire
- calcul des fonctions elementaires

Ce travail s'articule autours de quatre parties :

- Fonctions fondamentales du traitement du signal
- Realisation de l'unite centrale du processeur
- Realisation du peripherique
- Organigrammes et programmes

CHAPITRE 1

FONCTIONS FONDAMENTALES DU TRAITEMENT DU SIGNAL

1.1. L'OPERATION DE BASE DE LA TRANSFORMEE DE FOURIER :

Les méthodes de traitement de l'information (ou du Signal) ont pour but l'elaboration et l'interpretation des Signaux porteurs d'information, et d'assurer ainsi la perception, la transmission et l'exploitation de ces informations.

La description mathématique des Signaux est l'objectif fondamental de la théorie du Signal qui cherche à analyser la nature des alterations ou modifications subies par les Signaux lors de leur passage au travers des blocs fonctionnels generalement électrique ou electronique.

Par la même , elle fournit les renseignements essentiels necessaires à la conception ou à l'utilisation de ces dispositifs, d'autre part elle permet de déterminer et de tenir compte des limites de fonctionnement imposée à de tels systemes par des perturbations.

Considérons x(t) une fonction continue, cette fonction est échantillonnée à des intervalles égaux dans le temps d'où cette fonction peut être déterminée par la sequence de nombre x(o), x(t), ..., x(nt) et par interpolation on peut déterminer les différentes valeurs de x(t).

La transformée de fourier directe où DFT est une représentation de fourier d'une séquence finie d'échantillonage du Signal x(t), et permet de calculer le spectre de cette suite de valeurs où précisement le spectre d'une séquence d'impulsions de dirac correspondant aux valeurs numériques des éléments de la suite x(t), et dont la relation est :

$$X(n) = \sum_{k=o}^{n-1} x(k) \exp(-j\, 2\pi nk/N)$$

$$n = o,1,\ldots,n-1$$

Donc on voit que la DFT transforme un vecteur x(t) dans le domaine temporel en un vecteur x(n) défini dans le domaine fréquentiel.

1.1.1. LA TRANSFORMEE DE FOURIER RAPIDE : (F.F.T).

L'analyse des divers applications du traitement du Signal fait ressortir l'importance de la transformée de fourier en tant qu'outil irremplaçable dans l'analyse spectrale et le filtrage, seulement l'utilisation de la transformee de fourier n'a pris de l'ampleur qu'avec l'apparition des algorithmes rapides de calcul de la transformee de fourier et de puissants outils de traitement de l'information.

Quand la DFT est calculée à l'aide d'algorithmes rapides, on dit que l'on effectue une transformée rapide de fourier (F.F.T).

La F.F.T consiste en une série de procédés de calculs destinés à reduire le temps d'exécution d'une D.F.T.

Comme on l'a dit avant, il existe plusieur algorithme de calcul de la F.F.T, il a été choisi l'algorithme utilisant l'opération papillon avec le nombre d'échantillonnage N=2 qui est une puissance entière de 2., d'où on a une grande vitesse de calcul avec le minimum d'opération à effectuer.

1.1.2. ALGORITHME DE LA F.F.T :

La figure 1.1 illustre le diagramme de la F.F.T sur 6 points, l'examen de cette figure nous montre que la partie essentielle de ce diagramme est constituée par des motifs appelés itterations qui relient le signal initial à ses états intermédiaires, les coefficients de fourier sont obtenus à la dernière itteration. On reconnait là une géométrie identique de toutes les étapes de traitement composées d'unités algorithmiques semblables qu'on appelle "opération papillon".

Les conventions du "papillon" sont les suivantes ; le cercle(O) définit une opération d'addition et soustraction dans laquelle la somme apparait en haut et la différence en bas, la fleche (——→) décrit une multiplication avec une valeur située au dessus de la fleche.

Donc on voit que la DFT transforme un vecteur x(t) dans le domaine temporel en un vecteur x(n) défini dans le domaine fréquentiel.

1.1.1. LA TRANSFORMEE DE FOURIER RAPIDE : (F.F.T).

L'analyse des divers applications du traitement du Signal fait ressortir l'importance de la transformée de fourier en tant qu'outil irremplaçable dans l'analyse spectrale et le filtrage, seulement l'utilisation de la transformee de fourier n'a pris de l'ampleur qu'avec l'apparition des algorithmes rapides de calcul de la transformee de fourier et de puissants outils de traitement de l'information.

Quand la DFT est calculée à l'aide d'algorithmes rapides, on dit que l'on effectue une transformée rapide de fourier (F.F.T).

La F.F.T consiste en une série de procédés de calculs destinés à reduire le temps d'exécution d'une D.F.T.

Comme on l'a dit avant, il existe plusieur algorithme de calcul de la F.F.T, il a été choisi l'algorithme utilisant l'opération papillon avec le nombre d'échantillonnage N=2 qui est une puissance entière de 2., d'où on a une grande vitesse de calcul avec le minimum d'opération à effectuer.

1.1.2. ALGORITHME DE LA F.F.T :

La figure 1.1 illustre le diagramme de la F.F.T sur 6 points, l'examen de cette figure nous montre que la partie essentielle de ce diagramme est constituée par des motifs appelés itterations qui relie le signal initial à ses états intermédiaires, les coefficients de fourier sont obtenus à la dernière itteration. On reconnait là une géométrie identique de toutes les étapes de traitement composées d'unités algorithmiques semblables qu'on appelle "opération papillon".

Les conventions du "papillon" sont les suivantes ; le cercle(O) définit une opération d'addition et soustraction dans laquelle la somme apparait en haut et la différence en bas, la fleche (⟶) décrit une multiplication avec une valeur située au dessus de la fleche.

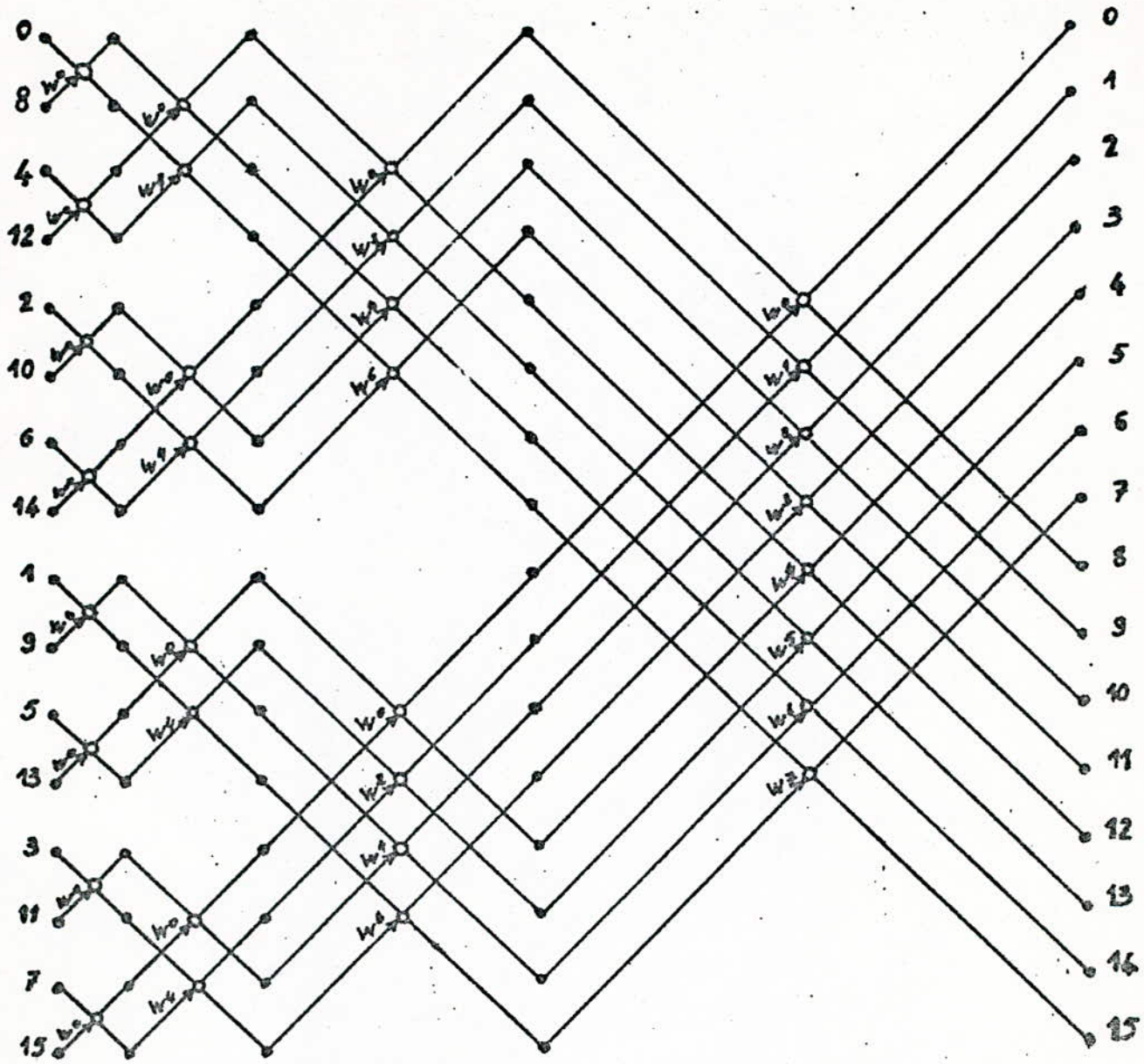


Fig 1.1 . Diagramme de FFT sur 16 points
Entrée désembrouillée _ Sortie ordonnée.

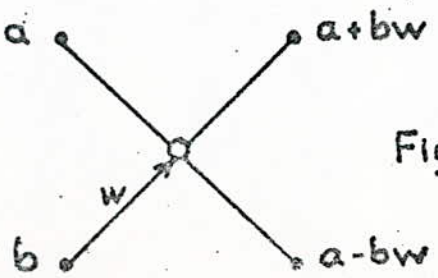


Fig 1.2 . Opération papillon

$C = a + b\,w$

$d = a - b\,w$

Dans le cas de N points et pour $0 \leq K < \frac{N}{2} - 1$,

les expressions liant chaque état intermédiaire (j+1) à l'état j sont :

$$X_{(j+1)}\ (K) = X_j\ (K) + W^p\ X_j \left[k+(N/2j)\right]$$

$$X_{(j+1)}\ \left[K+(N/2j)\right] = X_j\ (K) - W^p\ X_j\ \left[K + (N/2j)\right]$$

En posant K=Ko et K,=K+N/2j , les expressions deviennent :

$$X_{(j+1)}\ (Ko) = X_j\ (Ko) + W^p\ X_j\ (K1)$$

$$X_{(j+1)}\ (K1) = X_j\ (Ko) - W^p\ X_j\ (K1)$$

avec $W^p = e\ X\ p\ (-\ j\ 2\ \pi\ P/N) = \text{Cos}\ \ \theta - j\ \text{Sin}\ \theta$

et $\theta = 2\ \pi.P/N$

Donc on aura pour chaque papillon les expressions suivantes :

$$(1)\quad \begin{cases} RE_{j+1}\ (Ko) = REj\ (ko) + \left[REj\ (K1)\ \text{Cos}\ \theta + IM_j^{(k)} \text{Sin}\theta\right] \\ REj+1\ (K1) = REj(ko) - \left[REj\ (K1).\ \text{Cos}\theta + IMj(K1).\text{Sin}\ \theta\right] \\ IMj+1\ (Ko) = IMj(Ko) + \left[IM\ (K1)..\text{Cos}\theta - REj\ (K1).\text{Sin}\ \theta\right] \\ IM\ j+1\ (K1) = IMj\ (Ko) - \left[LMj\ (K1).\ \text{Cos}\ \theta - REj\ (K1).\ \text{Sin}\ \theta\right] \end{cases}$$

Ce qui nous a été demandé, c'est de calculer où de programmer les expressions de l'opération papillon c-a-d les expressions (1).

l'expression de la transformée de fourier inverse est,

$$(2)\ X(K) = \frac{1}{N} \sum_{n=o}^{N-1} x(n)\ \exp\ (j2.\ \Pi.n.k/N)$$

Les expressions de l'opération papillon déduite de cette expression (2) sont pratiquement les mêmes que les expressions (1), mais en prenant à la place de SIN O, - SIN O dans l'expression (1), où en prenant $-W^p$ au lieu de W^p , et ensuite en divisant les coefficients de fourier de la dernière itteration par N, et comme $N=2^p$, ça revient à decrementer l'exposant de P.

1.2. METHODE DE CALCUL DES FONCTIONS ELEMENTAIRES

Il existe un grand nombre de méthodes de calcul des fonctions élémentaires dont les plus connues sont :

- la méthode de TAYLOR
- la méthode d'approximation polynomiale
- la méthode ittérative
- la méthode de VOLDER.

Chacune de ces méthodes à ses propres caractéristiques et en fonction de l'application, on choisit celle qui convient le plus.

Caractérisons d'une manière succinte chacune des méthodes précitées :

METHODE DE TAYLOR :

La méthode de TAYLOR ne converge pas rapidement, et l'erreur méthodique croit avec la croissance de l'argument.

Du point de vue programmation, chaque fonction est programmée différement, ceci fait que pratiquement tous les programmes sont différents.

METHODE DES APPROXIMATIONS POLYNOMIALES :

Cette méthode est utilisée beaucoup plus souvent, elle est caractérisée par une grande unicité de calcul de toutes les fonctions élémentaires, ce qui est trés appréciable en programmation. Mais il faut conserver en memoire un grand nombre de coefficients de tous les polynomes.

METHODE ITERATIVE :

Dans cette méthode, la capacité mémoire est réduite, l'évaluation de l'erreur est simple ainsi que les formules de recurence des fonctions elementaires. Seulement on a à effectuer pour chaque itération des opérations de multiplication et de division d'où du point de vue programmation, c'est plus complexe.

METHODE DE VOLDER :

Dans cette méthode les coefficients occupent une petite capacité mémoire. Elle se caractérise par une bonne unicité de calcul de ces fonctions élémentaires, on remarque aussi que dans cette méthode, les opérations de multiplication et de division se réduisent à de simples décalages.

D'autre part il est à souligner la convergence rapide de la méthode. De ce fait, on a choisi la méthode de VOLDER pour la programmation des fonctions elementaires; car elle nous permet des simplifications du point de vue programme et une réduction de la capacité mémoire.

Cette méthode à pour principe la conversion des coordonnés carthesiennes en coordonnés polaires que l'on adapte pour le calcul des fonctions élémentaires.

Cette adaptation consiste en la réalisation d'une suite de rotation de tayon vecteur d'angles standards autour de l'origine avec changement simultané de la longueur du vecteur.

1.2.1. CALCUL DES FONCTIONS SIN ET COS :

Pour le calcul de SIN φ et COS φ, on prend le vecteur initial de coordonnée (1,0) et on lui fait subir des rotations d'angle αi (i=o,1,..., n-1) de telle manière que la somme $\sum \alpha_i = \varphi$ et que le vecteur initial coincide avec le vecteur (x,y) à quelques erreurs pres.

Les nouvelles coordonnées sont exprimés par rapport aux précédentes de la façon suivante :

$$Yi+1 = \cos \alpha i \ (Yi + Xi \ tg \ \alpha i)$$
$$Xi+1 = \cos \alpha i \ (Xi - Yi \ tg \ \alpha i)$$

Il a été démontré, que dans un systeme de calcul binaire, le processus converge pour $\alpha i = Artg\ 2^{-i}$ et $|\varphi| < \pi/2$
cece nous permet d'avoir des décalages au lieu des opérations de multiplication, on obtient les formules de reccurence suivantes :

$$Yi+1 = Yi + \delta i\ xi\ 2^{-i}$$
$$Xi+1 = Xi - \delta i\ Yi 2^{-i}$$
$$\varphi i+1 = \varphi i - \delta i\ Artg 2^{-i}$$
$$\delta i = S\ I\ G\ N\ \varphi\ i$$

avec $X = C\ O\ S\ \varphi$ et $Y = S\ I\ N\ \varphi$

On remarque que le terme Cos X i a été négligé , des corrections s'imposent pour avoir un résultat correcte.

Pour chaque itteration on néglige le terme Cos xi d'où / augmentation du module du vecteur de 1/Cosxi, pour N iterations le module du vecteur augmentera d'une constante de valeur :

$$K = \prod_{i=0}^{n-1} 1/\cos\alpha_i = \sqrt{\prod_{i=0}^{n-1} (1 + 2^{-2i})}$$

Pour les valeurs initiales de X et Y on a Xo=1 et Yo=0 donc pour faire les corrections de cette méthode, on prend pour valeurs initiales :

$$xo = 1/k \qquad Yo=0 \qquad \varphi_o=\varphi \text{ avec } |\varphi| < \pi/2.$$

1.2.2. CALCUL DES FONCTIONS ARCTG Y/X et $\sqrt{X^2 + Y^2}$:

Pour le module et l'argument du vecteur (X,Y) , on prend le sens inverse de celui de COS et SIN c-a-d qu'on prend notre vecteur initial (X,Y) et on lui fait subir des rotations d'angles xi = Arctg 2-i, jusqu'à ce que le vecteur coincide avec l'axe des abcises d'où, on aura :

$$Y_n = Arctg\ x/Y \quad et \quad X_n = K.\sqrt{x2 + Y\ 2}$$

Mais comme on a fait une erreur sur le module en négligeant Cos Xi à chaque iteration la correction se fait en diminuant le module du vecteur d'arriver, donc on a :

$$\varphi_n = \text{Arctg } Y/X \quad \text{et} \quad X_{n/k} = \sqrt{x^2 + y^2}$$

Les formules de recurence sont :

$$\begin{cases} y_{i+1} = y_i - \xi_i x_i 2^{-i} \\ x_{i+1} = x_i + \xi_i y_i 2^{-i} \\ \varphi_{i+1} = \varphi_i + \xi_i \text{ Arctg } 2^{-i} \\ \xi_i = SIGN\, y_i \end{cases}$$

$$x_0 = X \qquad y_0 = Y \qquad \varphi_0 = 0$$

1.2.3. CALCUL DU LOGARITHME :

Le processus de calcul de LOGX converge pour des valeurs de X comprises entre 0 et 1.

Dans ce cas on prendra comme vecteur initial le vecteur Xo=X et Yo = X-1 , et on fait des rotations d'angle

jusqu'a ce que le vecteur final coincide avec l'axe des abcises et on a

Les formules de reccurence sont

$$\begin{cases} y_{i+1} = y_i - \xi_i x_i 2^{-i} \\ x_{i+1} = x_i + \xi_i x 2^{-i} \\ \varphi_{i+1} = \varphi_i + LOG\,(1 + \xi_i 2^{-i}) \\ \xi_i = SIGN\, y_i \end{cases}$$

$$Xo = X \qquad Yo = 1 - X \qquad \varphi_0 = 0$$

Ce processus de calcul du L O G peut être appliquer au L O G des différentes bases c - a - d $\text{Log}_a X$ (base a).

d'où

$$\varphi_{i+1} = \varphi_i + \text{Log}_a\,(1 + \xi_i 2^{-i})$$

1.2.4. DETAIL DE CALCUL :

Lors de la programmation de la methode de VOLDER enfortrau il a été démontré que le nombre d'iterrations nécessaires pour avoir une bonne precision est n = 9 , ce qui nous arrange nous aussi, d'autant plus que le résultat est donné en vergule fixe sur un Octet donc plus de précision ne se verrai pas sur le résultat.

Pour n = 9 on a : K = 1,64651

et

$$1/K = 0{,}60734 \simeq 0{,}5 + 0{,}10734 = 2^{-1} + 2^{-3}$$

donc $Xn / K = X\,n\,2^{-1} + Xn2^{-3}$

On voit que la division Xn par la constante K se réduit à de simples decalages du résultat .

1.2.5. CONCLUSION :

On voit bien que cette méthode de VOLDER a une grande unicité de calcul, surtout du point de vue programmation, le nombre d'itterations n = 9 est petit et donne une bonne précision pour le résultat final.

1.3. GENERATION DE DONNEES PSEU-ALEATOIRES:

Pour les différents programmes étudiés, on a besoin de simuler des données. Ceci nous a amené à élaborer un programme de génération de données pseudo-aléatoires à partir de la relation suivante :

$$Ci+1 = (\lambda C i + \mu) \quad M O D [p]$$ étant la période.

$P = 2^n$

$0 < CO < P$

$\lambda = 2^s + 1$ avec $S = 0,1,\ldots, n-1$

μ étant un nombre impaire le plus grand possible.

1.4. PRODUIT SCALAIRE :

Notre but est de faire le produit scalaire entre deux vecteur x et y de 512 composants.

$X \ (X_1, X_2, \ldots, X_{512})$

$Y \ (Y_1, Y_2, \ldots, Y_{512})$.

d'où

$$X.Y = X_1 . Y_1 + X_2 . Y_2 + \ldots + X_{512} . Y\ 512$$

Les valeurs des 512 coordonnées de X et de Y sont données par le programme de generateur de données.

1.5. LES OPERATIONS EN VIRGULE FLOTTANTE :

Dans notre cas toutes les opérations arithmétique s'effectues avec des nombres représenter en virgule flottante, et ceci pour avoir une meilleur precision pour le résultat.

Un nombre en virgule flottante est composé d'une mantisse et d'un exposant, dans notre cas la mantisse s'écrit sur un Octet et l'exposant sur un Octet (pour plus de précision on pouvait prendre la mantisse sur 2 octets ou plus).

La mantisse d'un nombre en vergule flottante est comprise entre 0,5 et 1

$$0{,}5 \leqslant /M/ < 1$$

Le bit b7 de la mantisse nous indique si le nombre est positif "0" ou négatif "1", le nombre composé de b6 a bo est une partie décimale, la partie entière étant égale à zéro.

L'exposant étant entier donc le bit b7 indique le signe et dont les valeurs maximales et minimales sont 7 F et 8 0 (+ 127 et - 128).

ex : binaire — decimal

6 C . 0 II 0 II 00 $= b_6 x 2^{-1} + b_5 . 2^{-2} + b_4 . 2^{-3} + b_3 . 2^{-4} + b2 . 2^{-5} + b_1 . 2^{-6} + b_o 2^{-7}$

$= 1.2^{-1} + 1.2^{-2} + 0.2^{-3} + 1.2^{-4} + 1.2^{-5} + 0.2^{-6} + 0.2^{-7}$

$= 2^{-1} + 2^{-2} + 2^{-4} + 2^{-4}$

$= 0{,}5 + 0{,}25 + 0{,}00625 + 0{,}03125 = 0{,}84375$

6C 0 II 0 II 00 = 0,84375

ex :

EC I II 0 II 00 = -0,84375

Ceci si nous resonnant un binaire et non en complément à 2. par la suite on va montrer qu'elle est la différence si on resonne en binaire ou en complément à 2, ce qui est trés important pour la suite des calculs.

Si on fait un decalage à gauche, on fait entrer à droite des zeros dans les cas où le nombre est positif ou négatif, mais si on fait un decalage à droite (le bit b7 n'est pas décalé) pour un nombre positif , on fait entrer des zéros à gauche, tandis que pour un nombre négatif on fait entré des "1".

On fait une normalisation pour passer, d'un nombre en virgule fixe à un nombre représenté en vergule flottante
ex : nombre en virgule flottante.

M = 0.00 I II 0 I E = 0 0 0 0 0 0 0 0
1B 00

1er decalage.

M = 0.0II IO IO E = I. 000 000 I
CA 81

2ème decalage

M = 1.00 I 0I00 E = I.000 00I0
94 82

Donc le nombre E5 en virgule fixe s'écrit M = 94 E = 82 en virgule flottante.

Pour la suite des calculs on doit faire notre choix sur le resonnement, soit en binaire, soit en complément à 2 , dans les 2 cas les nombres positifs s'écrivent de la même façons, mais ils s'écrivent differemment pour les nombres négatifs.

ex :

en décimal	en binaire pur	en complement à 2.
-1	1 000 0001=81	IIII IIII = FF
-2	1 000 0010 =82	IIII IIIO = FE
-70	IIII 0000 =F0	1001 0000 =90

On voit bien qu'il y a une grande différence entre écrire un nombre en binaire ou en complément à 2 du moins pour les nombres négatifs.

Notre choix s'est porté sur le resonnement en complément à 2, le resonnement en binaire est aventageux pour la compréhension des résultats et plus simple à manipuler.

* RENDRE CE QUI EST SIMPLE COMPLIQUE EST COURANT ,

RENDRE LE COMPLIQUE SIMPLE , C'EST CA LA CREATIVITE .

CHAPITRE 2

REALISATION DE L'UNITE CENTRALE DU PROCESSEUR

2.1. - DESCRIPTION DE LA FAMILLE 6800 :

2.1.1. LE MICROPROCESSEUR 6809 :

Ce microprocesseur est réalisé en technologie H - MOS c-à-d N - MOS à haute densité d'intégration, ces entrées et sorties sont compatibles avec T T L ce qui facilite la connection avec les circuits logiques classiques. Ces caractéristiques principales sont :

- Alimentation sous tension unique 5 V ± 5 %
- Entrée/sortie compatibles avec T T L
- Signaux de bus identiques à ceux de toutes la famille 6800 permettant l'emploi de tous les périphériques de cette famille
- Signaux de gestion du bus autorisant le DMA ou le fonctionnement en multiprocesseur.
- Possibilité de travailler avec des mémoires lentes
- Instruction de synchronisation avec un évenement extérieur.

2.1.1.1. - BROCHAGE DU 6809

Le brochage du 6809 est le suivant :

- une ligne RESET, qui a pour effet de faire une initialisation du 6809.
- une ligne HALT, qui permet de mettre le 6809 en attente après avoir exécuté l'instruction en cours et libere les bus c-a-d que toutes les sorties passent à l'état haute impedance.
- trois lignes sources d'interruptions externes matérielles, une non masquable NMI, et 2 masquables IRQ , et FIRQ.

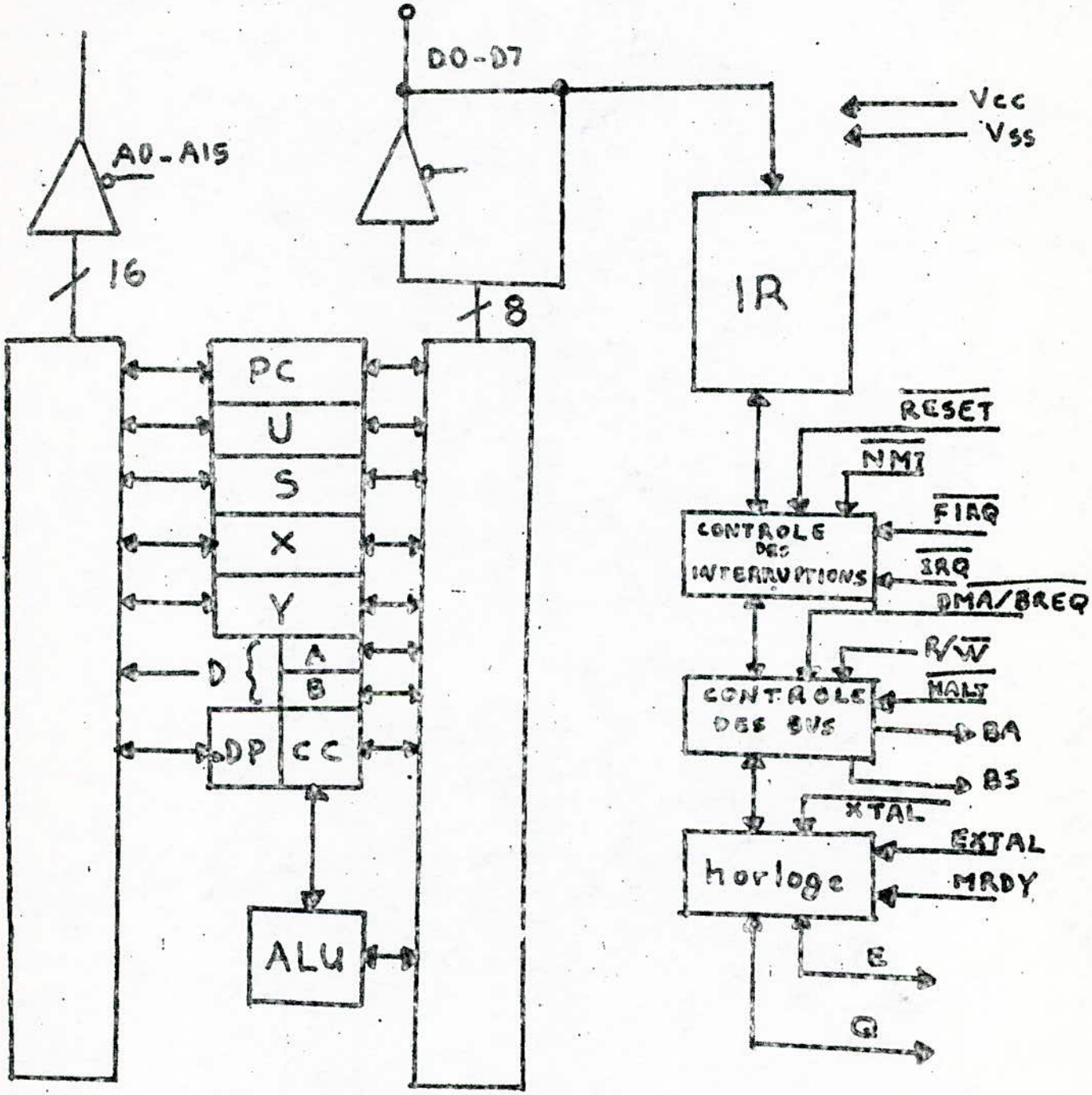


fig 1 Schema fonctionnel du 6809

- deux lignes E et Q, qui sont les horloges du système, il sont en

- deux lignes E et Q, qui sont les horloges du système, il sont en quadrature. Ce sont eux qui rythment tous les échanges avec les éléments du système, ils ne penveut pas être affectés par l'état du processeur. Les adresses sont validées depuis le front montant de Q jusqu'à la fin du cycle, alors que les données sortant du 6809 sont validées à partir du front montant de E jusqu'à la fin du cycle.

- 2 lignes BA, BB qui permettent de savoir dans quel état se trouve le processeur.

- - 2 lignes XTAL et EXTAL servant à connecter un quartz de fréquemce égale à quatre fois la fréquence de fonctionnement du microprocesseur (une division par 4 se fait à l'intérieur), le quartz peut être quasiment quelconque, l'oseillateur étant trés tolérant.

- une ligne R/W pour la lecture et l'écriture du microprocesseur.
- une ligne MRDY ou MEMORY READY, qui permet de ralentir les accés mémoires du 6809 pour les mémoires lentes et ceci en prolongeant l'état haut de E.

- une ligne DMA/BREQ qui permet de demander au 6809 de libérer son bus pour un autre processeur (en multi-processeur) ou pour faire un DMA c-à-d accés directe à la mémoire sans passer par le microprocesseur.

- 8 lignes pour le bus de données, qui sont bidirectionnelles.
- 16 lignes pour le bus adresse, qui sont indirectionnelles.

2.1.1.2. ARCHITECTURE INTERNE DU 6809

Le 6809 comporte 9 registres internes programmables accessibles par l'utilisateur, qui sont :

- 2 index X et Y utilisés principalement pour l'adressage indéxé

ADRESSES	VECTEURS D'INTERRUPTION.
FFFF / FFFE	$\overline{RESET}$
FFFD / FFFC	$\overline{NMI}$
FFFB / FFFA	SWI
FFF9 / FFF8	$\overline{IRQ}$
FFF7 / FFF6	$\overline{FIRQ}$
FFF5 / FFF4	SWI2
FFF3 / FFF2	SWI3
FFF1 / FFF0	RESERVE.

ADRESSAGE DES VECTEURS D'INTERRUPTION.

BA	BS	FONCTIONEMENT DU M.P.U.
0	0	Normal.
0	1	Reconnaissance d'interruption
1	0	Reconnaissance de Synchro Externe.
1	1	Arret ou Bus accordé.

FONCTIONNEMENT DU MICROPROCESSEUR.

- 2 pointeurs de piles U et S, le pointeur U est utilisé par l'opérateur, S est le pointeur systeme ou superviseur, et peuvent être utilisées comme registre d'index.

- un accumulateur D de 16 bits formé par 2 accumulateurs A et B de 8 bits, A étant de poids fort, B de poids faible et on peut utiliser A et B indépendement l'un de l'autre.

- un registre DPR où registre de page, utilisant ainsi le mode d'adressage direct.

- Un registre CCR ou registre d'état, dont chaque bit à une signification particulière . Suivant l'état du processeur et du resultat de l'opération qui vient d'etre réalisé.

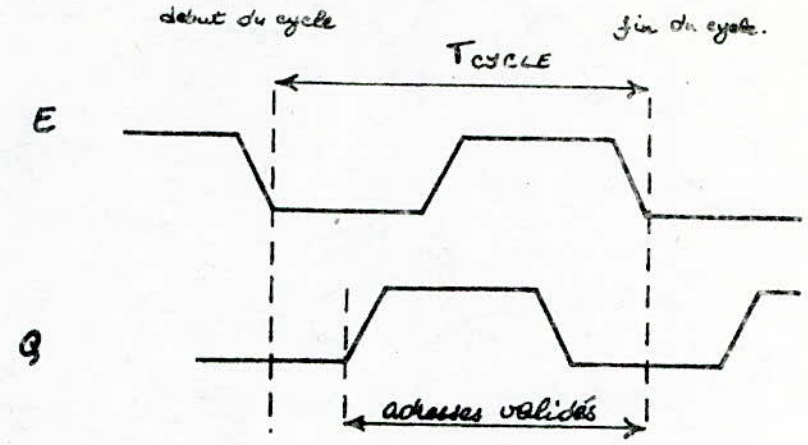
- un registre DC où compteur ordinal, qui pointe sur l'instruction à effectuer.

2.1.2.3. LOGICIEL DU 6809

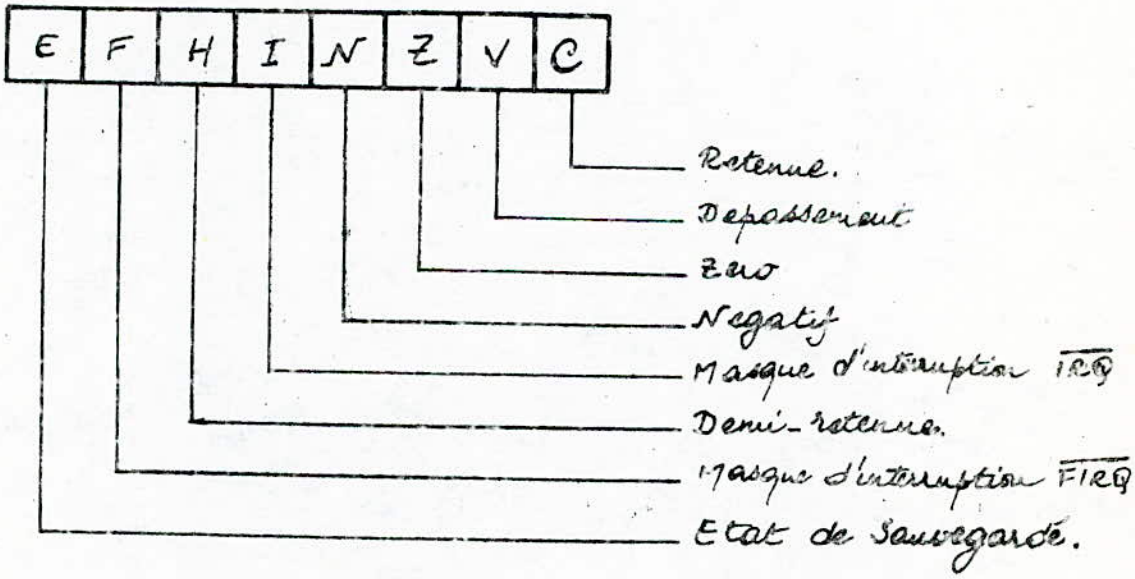
Le 6809 possede un logiciel trés riche par rapport à ces antécédents, il possède :

- 10 modes d'adressage extremement puissant
- 59 mnemoniques d'instructions qui, compte tenu de leurs possibilités sont équivalent à 1464 instructions différentes.

- opérations arithmétiques et logiciels sur 16 bits
- Une multiplication 8 bits par 8 bits en une seule instruction
- un langage assurant une compatibilité ascendente avec le 6800.



CHRONOGRAMME DE E et Q.



REGISTRE D'ETAT DU 6809.

2.1.2. - L'INTERFACE PARALLELE PROGRAMMABLE PIA 6821.

L'interface PIA est destinée aux applications pour lesquelles les données sont envoyés ou reçus en parallèle c-à-d sur 8 lignes de données et permettant la communication entre le microprocesseur et les différents périphériques (clavier, afficheur, imprimante, interface de commande d'automatisme , ...).

2.1.2.1. - BROCHAGE DU PIA :

C'est un circuit réalisé en N-MOS dont l'organisation externe est :

- alimentation monotension + 5V
- 8 lignes pour le bus de donnée Do-Dè pour la transmission et reception des données avec le microprocesseur.
- 16 lignes divisées en 2 ports A et B qui sont pratiquement identiques et qui peuvent être programmées en entrée ou en sortie.
- 3 lignes de validation du PIA, CSo, CS1 et CS2
- une ligne pour la lecture et l'écriture R/W
- une ligne pour l'initialisation RESET
- 2 lignes d'interruptions IRQA etIRQB pour les 2 ports.
- une ligne pour l'horloge E
- 4 lignes pour le controle, CA1, CA2, CBA, CB2 et qui permet tent le dialogue avec l'extérieur. Le fonctionnement de ces lignes est fixé par le registre de controle.

2.1.2.2. ARCHITECTURE INTERNE DU PIA :

LE PIA possède 6 registres :
- CRA - CRB : 2 registres de controle correspondant aux port A et port B et qui fixent le fonctionnement ou le dialogue : PIA - périphérique et PIA- microprocesseur. Ces registres sont à lecture et écriture.

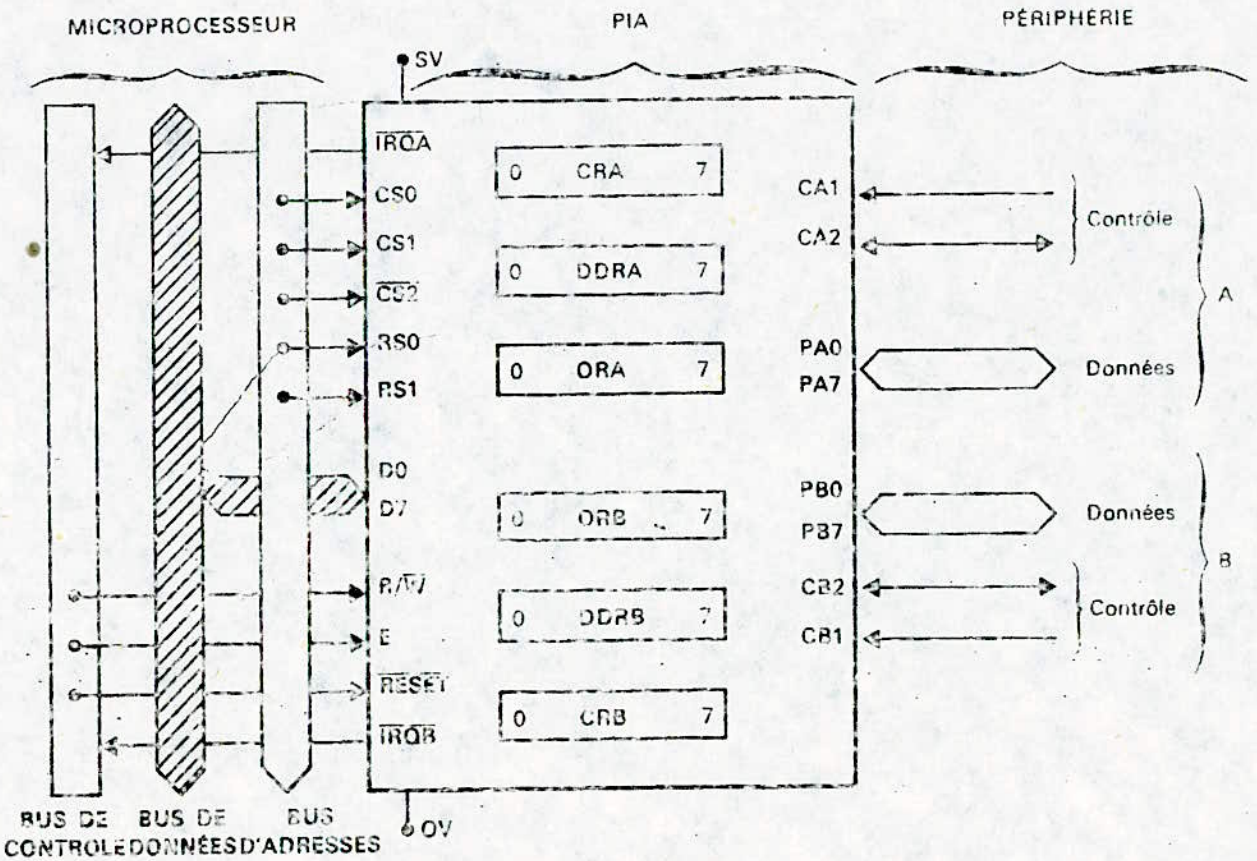


Fig. III.5. — *Organisation interne et externe du PIA.*

2.1.2.3 PROGRAMMATION DU PIA :

Vis à vis du microprocesseur, le PIA se comporte comme étant 4 positions mémoires bien qu'il comporte 6 registres internes.

Les registresdde microprocesseur, le PIA se comporte comme étn étan Les registres de direction de données DDRX et de sorties ORX ont la meme adresse, le bit b2 du registre de controle CRX(x étant A ou B), permettra la distinction entre ces 2 registres, un "0" correspond au registre DDRx, et un "1" au registre ORX.

Donc avant de programmer DDRx ou ORX il faudra programmer CRX, quitte à modifier ce dernier par la suite.

Le plus important dans la programmation du PIA, c'est le registre de controle, et pour une meilleur compréhension de ce qui va suivre, on ne va utiliser que le port A, la programmation du port B étant la même.

Dans le registre de controle on ne peut programmer que les bits de CRA0 à CRA5, les bits CRA6 et CR7 étant commandé par le PIA lui-même.

- Les bits CRA0 et CRA1 déffinissent le fonctionnement de la ligne de dialogue CA1.

. CRA1 : précise le sens du front active attendu par le PIA (front montant ou descendant selon le cas).

. CRA0 : autorise ou non l'interruption du micro par IRQA le front active est reçu par CA1.

- Le bit CRA2, comme il a été indiqué avant permet l'accés à ORA ou DDRA.

- Les bits CRA3, CR4, CR5 déffinissent le fonctionnement de la ligne CA2 qui peut etre en entrée ou en sortie, on distingue 4 modes de fonctionnement :

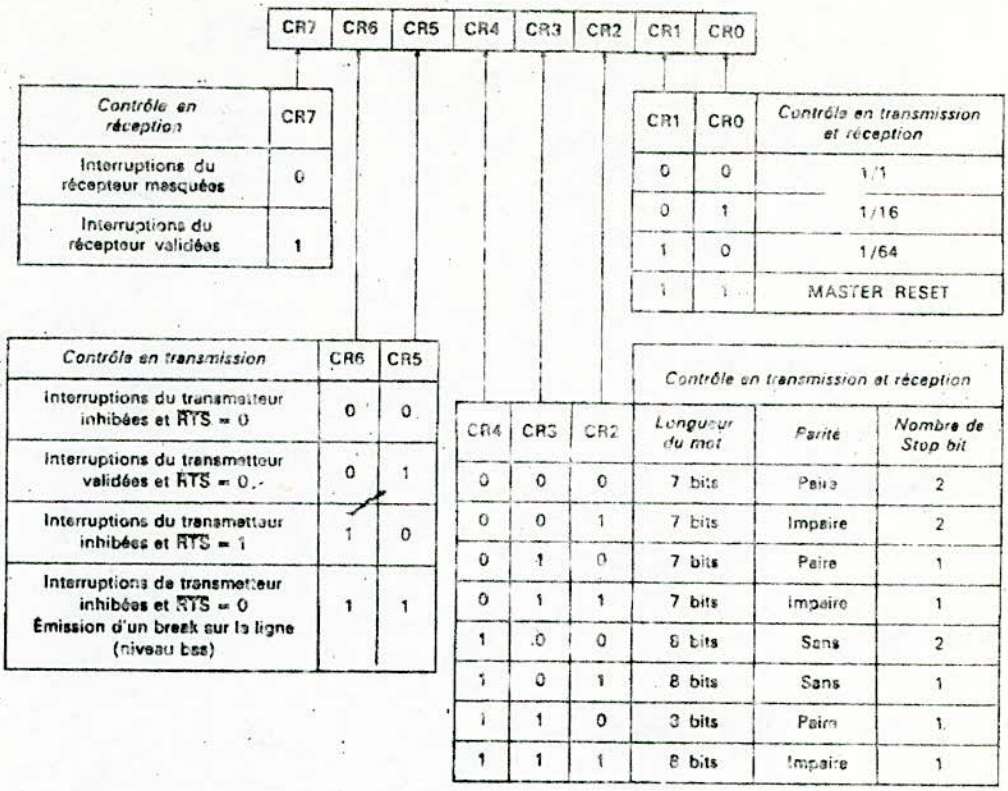


Fig. III.17. — *Programmation du registre de contrôle de l'ACIA.*

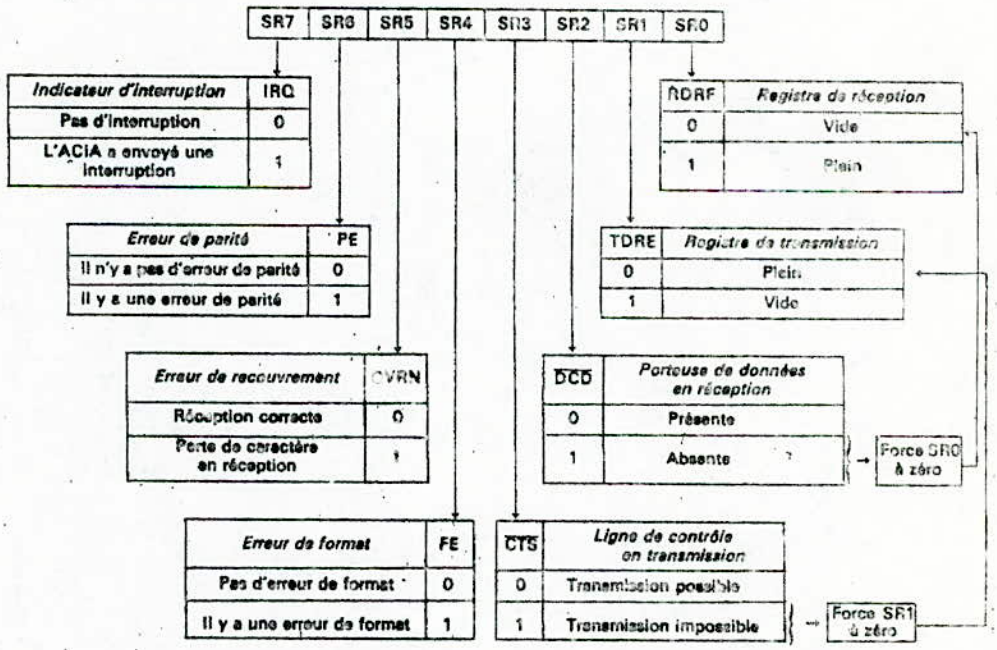


Fig. III.18. — *Rôle du registre d'état de l'ACIA.*

. CRA5 = 0 , CRA4 = x , CRA3 = x : (x etat indifférent).

La ligne CA2 est programmé en entrée, elle fonctionne exactement comme la ligne CA1, et CRA4, CR3 ont un rôle identiqueque CRA1 et CRA0.

. CRA5=1, CRA4=0, CRA3=0 (mode dialogue) :

La ligne CA2 est en sortie, et passe à l'etat bas sur le premier trait descendant de E qui suit un ordre de lecture de ORA si le port A est entrée, ou un ordre d'écriture de ORA si le port A est en sortie, elle repassera à l'etat haut lorsque CA1 recevra le prochain frait active.

.CRA5 = 1, CRA4=0, CR3=1 (mode impulsion) :

La ligne CA2 est en sortie et passe à l'etat bas sur le premier frait descendant de E qui suit l'ordre de lecture de l'ORA si le port A est en entrée ou un ordre d'écriture si le port A est en sortie, elle repassera à l'etat haut sur le prochain trait descendant de E.

. CRA5=1, CRA4=1, CRA3=0/1 (mode programmé) :

la ligne CA2 est en sortie et suit l'etat du bit CRA3, si CRA3 CRA3=0, CA2 est à l'etat bas, si CRA3=1 , CA2 est à l'etat haut donc il faut modifier le contenu de tous le registre CRA pour changer l'etat de CA2.

- Le bit CRA7 est positionné par le PIA à 1 lorsuq'on reçoit le active sur CA1 et il est remis à "0" par lecture de ORA.

- Le bit CRA6 fonctionne exactement comme le CRA7. Si CA2 esten entrée, il est forcé à "0" si CA2 est en sortie.

Aprés un RESET tous les ports sont en entrée, CRA6 et CRA7 à "0", CA2 est en entrée et n'autorise pas d'interruption.

2.1.2.4. APPLICATION

Dans notre carte, on a utilisé 3PIA, le premier a été utilisé pour le clavier et les afficheurs, donc avec le périphérique, les

les 2 autres PIA sont utilisés pour fournir des données du microprocesseur à la carte d'unité de traitement, car une des applications de notre projet consiste à fournir aux unités de traitements 2 formats de données sur 16 bits et sur 8 bits et ceci en même temps; ce qui necessite 2 PIA.

Les ports de ces PIA sont utilisés en sortie et on n'a besoin que d'une seule ligne de commande CA2 pour prevenir l'unité de traitement que les données sont présentées sur les sorties.

La programmation concernant le 2ème et 3ème PIA est donnée dans le chapitre 4. La programmation du 1er PIA est donnée avec les programmes de gestion du clavier et des afficheurs.

2.1.3. L'INTERFACE SERIE PROGRAMMABLE ACIA 6850 :

L'ACIA est l'interface de communication série asynchrone c-à-d que les données reçues ou transmis sont envoyées bit par bit et permet de connecter la carte à un terminal quelconque (teletype, visu, modem, ...).

La liaison de l'ACIA avec le périphérique se fait sur 2 lignes, une ligne pour la transmission et une ligne pour la reception.

On distingue 2 types de fonctionnement :
- en half duplex : dans ce cas on utilise qu'une seule ligne pour la transmission est pour la réception c-à-d que si on utilise un clavier est une visu, les données de la touche enfoncée seront traités d'abord par le microprocesseur puis elles seront envoyées à la visu.

- en full duplex : dans ce cas on utilise les 2 lignes, une pour la transmission, l'autre pour la reception, maintenant les données de la touche enfoncée seront envoyées en même temps à la visu et au microprocesseur. Donc on voit bien que du point de vue rapidité, l'ACIA est en général utilisé en full duplex des que l'application le permet.

2.1.3.1. BROCHAGE DE L'ACIA :

Comme tous les circuits de la famille 6800, l'ACIA estréalisé en technologie N-MOS, dont l'organisation externe est :

- 8 lignes de donnée autorisant le transfert entre le microprocesseur et l'ACIA.

- 3 lignes pour la validation de l'ACIA, CSo , CS1, et CS2
- une ligne pour l'horloge E qui sert à cadenser les échanges de données entre le microprocesseur et l'ACIA.
- une ligne pour la lecture et l'écriture R/W
- une ligne RS pour la selection des registres internes

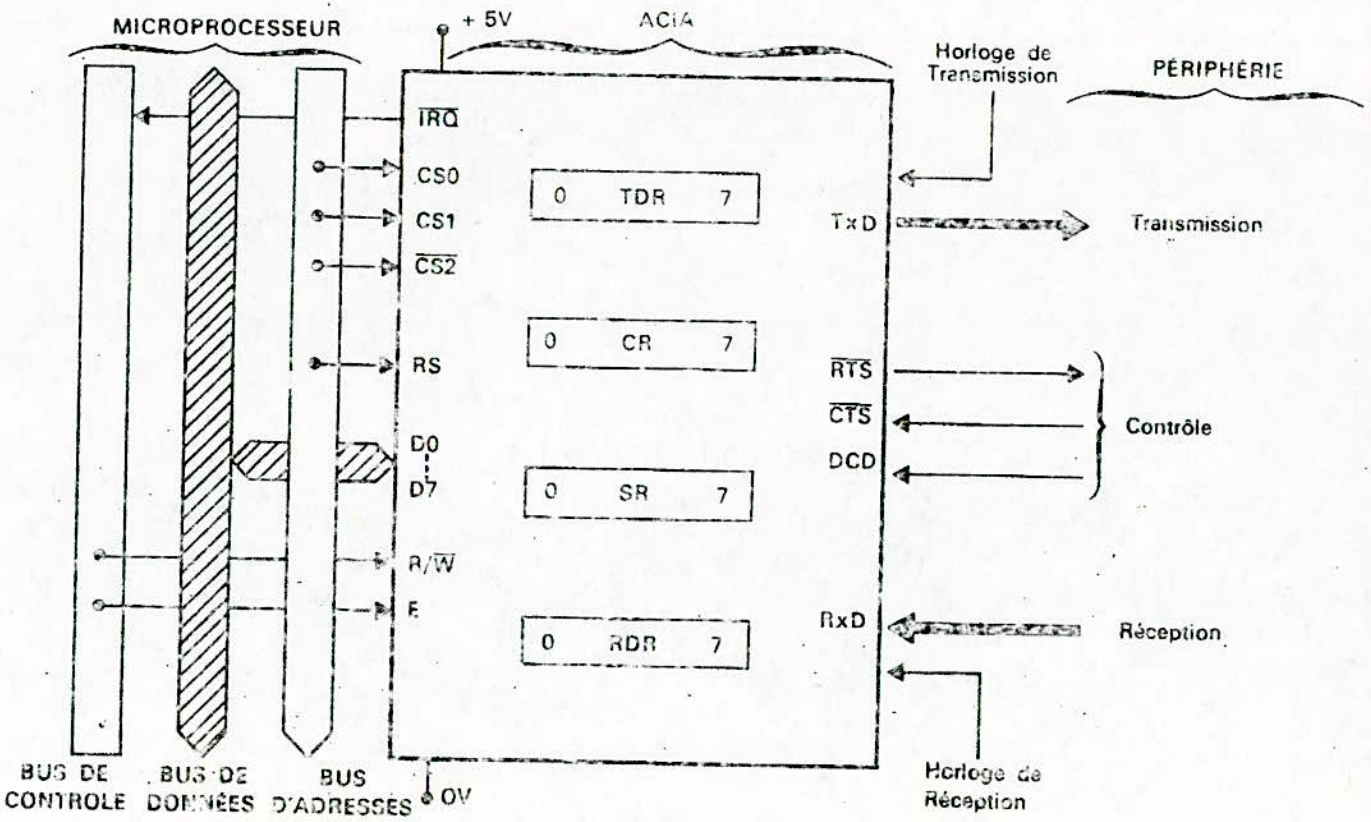


Fig. III.13 — *Organisation interne et externe de l'ACIA.*

- une ligne d'interruption RQ
- une ligne RXD par laquelle les données arrivent en série
- une ligne TXD par laquelle les données sortent en série
- une ligne RXC ou entrée d'horloge de reception qui définie la vitesse de reception de données
- une ligne TXC ou entrée d'horloge de transmission qui définie la vitesse de transmission de données
- une ligne RTS, cette sortie est à zero lors de la transmission d'une donnée en half duplex, et à 1, elle indique qu'on peut envoyé des données et ceci en full duplex.
- 2 lignes CTS et DCD ne sont utilisés que si on a un modem comme périphérique, donc en l'absence de celui-ci, on doit les maintenir à zero.

2.1.3.2. ARCHITECTURE INTERNE DE L'ACIA :

L'ACIA comporte un emetteur et un recepteur de données asyachrone et 4 registres internes, les circuits d'emissions et de receptions peuvent fonctionner simultanement à des vitesses différentes.

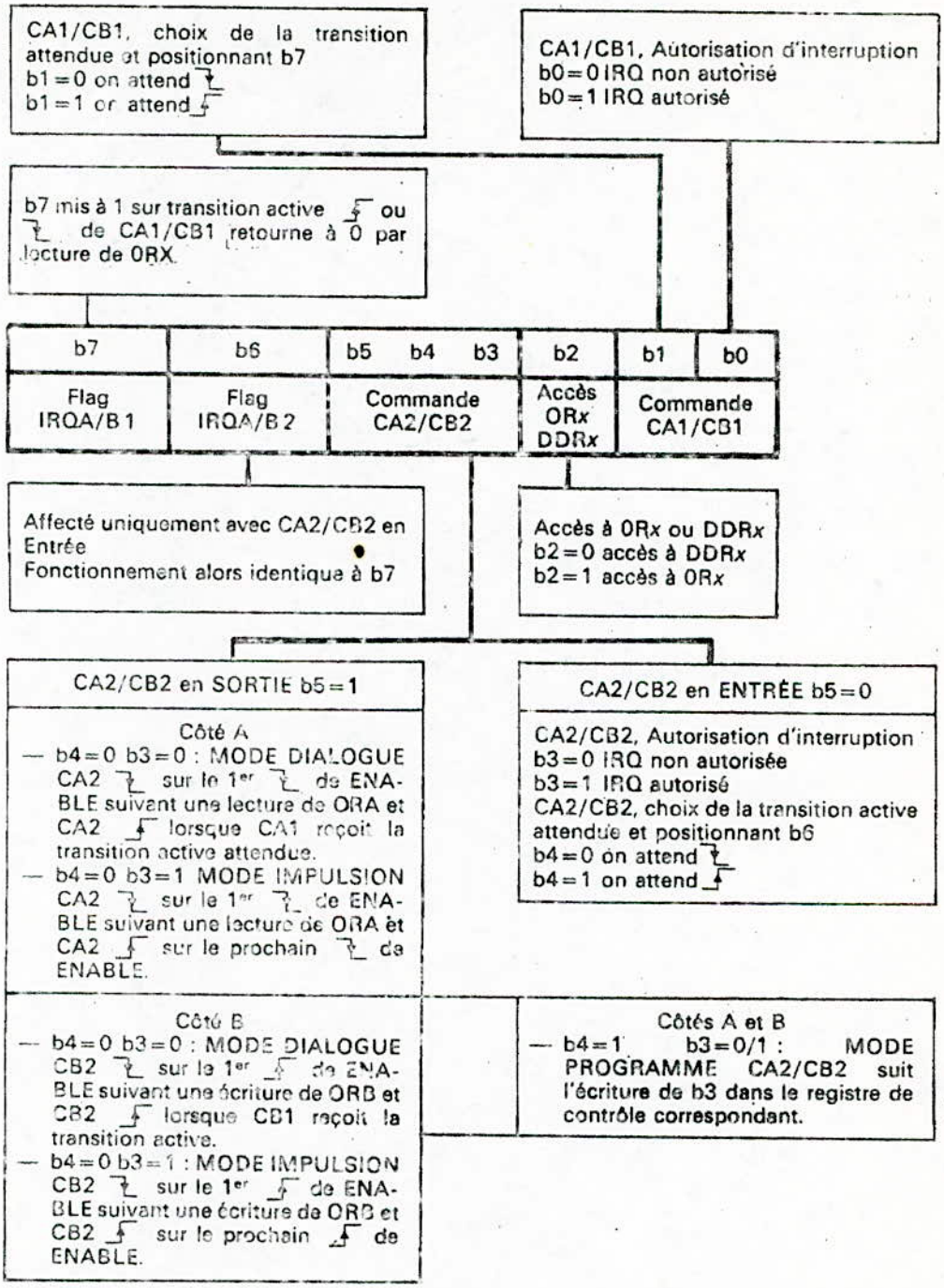
Les données sont converties du mode parallèle au mode série pour la transmission, et du mode série au mode parallèle pour la reception. Du fait de la présence des registres de receptions et de transmissions séparement avec des registres à decalages, l'ACIA peut recevoir où emettre un signal, alors qu'on est entrain de lire où d'écrire le caractere suivant dans le registre approprié.

Les registres internes de l'ACIA sont :
- CR: c'est un registre de controle qui contient les parametres de la transmission et de la reception (format, vitesse, parité,...).

- SR : c'est un registre d'etat, qui renseigne le microprocesseur sur les opérations en court
- TDR : c'est un registre de transmission des données
- RDR : c'est un registre de reception de données qui ont été débarassé de leur bit de Start et Stop.

Les registres à decalage fonctionnent en synchronisme avec une horloge interne.

Tableau 2 — Synthèse de programmation du PIA



2.1.3.3. PROGRAMMATION DE L'ACIA

L'ACIA peut être utilisé en 2 modes, soit le mode programmé et c'est le microprocesseur qui choisi le moment où il faut procédé à la reception ou à la transmission de données, quitte à ce qu'il attend que [illegible] qui interrompt le micro (registre de reception plein où registre de transmission vide).

Pour emettre un caractere il suffit de s'assurer que le registre d'emission soit vide (bit b1 du registre d'etat à 1).

Pour recevoir, il faut s'assuré que le registre de reception est plein (bit bo du registre d'etat à 1).

- TRANSMISSION, un caractere est transmis bit par bit en commençant par Do precedé d'un bit de Start, D7 étant suivi par le bit de parité puis 1 ou 2 bits de Stop. Les bits sont transmis sur une transition négative de l'horloge interne. L'ecriture dans le TDR se fait suivant le trait descendant de E.

- RECEPTION : Les données sont transmises de façon aleatoire, se sont les bits start et stop qui vont permettre une synchronisation des bits du caractere. Les bits de start et stop seront éliminés par les registres à decalage.

2.2.1. SYNOPTIQUE

Le schéma synoptique général d'une carte C P U dont l'élément de base est le 6809, comprend les éléments suivants :

- un decodeur d'adresse, chargé d'aiguiller le microprocesseur vers la zone mémoire adressée et d'activer le buffer de données suivant le niveau du signal R/W.

- Des buffers, par où transitent les lignes d'adresses, de données et de controle, sont destinés à compenser d'éventuelles attennations sur le bus.

- Les mémoires RAM et EPROM permettent de ranger les différents programmes et données utilisés.

- Un moniteur, utilisé pour la gestion du système
- Des interfaces entrée/sortie du type PIA et ACIA :

  . Deux PIA, utilisés pour être connectés avec une carte d'unité de traitement à tester.
  . Un PIA utilisé avec un périphérique clavier-afficheur
  . Un ACIA utilisé pour être connecter à une imprimante où une unité de visualisation.

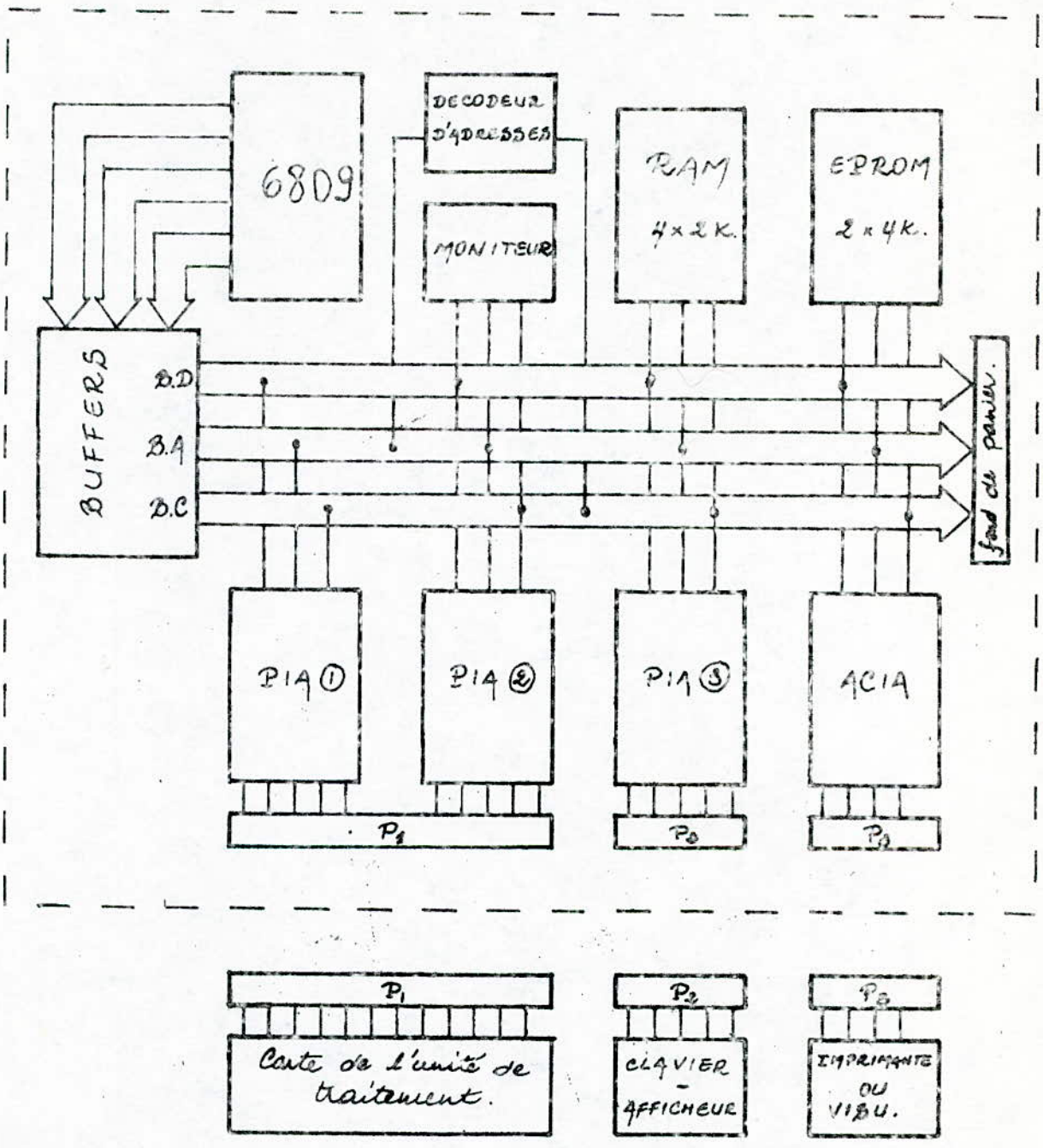
8.2.2. LE MICROPROCESSEUR

Toutes les entrées du 6809 étant active au niveau bas, nous allons relier celles-ci à + 5V par des résistances de rappels pour qu'elles ne puissent être activées accidentellement par un parsite.

Toutes les adresses seront bufferisées ainsi que les donnés et quelques signaux de controles.

2.2.3. LES BUFFERS :

Leurs rôles consistent à amplifier les signaux issus du microprocesseur afin de pouvoir attaquer de nombreux circuits, présentent [illegible] protège ce dernier d'[illegible]



SYNOPTIQUE DE LA CARTE C.P.U

un autre avantage, c'est d'isoler le microprocesseur du bus, ce qui protège celui-ci d'une destruction éventuelle en cas de problème (court-circuit par exemple).

Ces buffers, independement du fait qu'ils servent à éviter des depassements de sortances des circuits logiques présentent plusieurs avantages :

- sortie 3 états
- courant de sortie trés important
- courant d'entrée trés faibles permettant de ne pas avoir à s'occuper des problèmes d'entrance et de sortance dans un systeme équipé de tel amplificateur sur toutes les cartes
- sorties protégées contre les courts circuits de courte durée.

Il existe 2 types de buffers .

2.2.3.1. LES BUFFERS DE DONNEES :

Ce sont des buffers bidirectionnelles , c-à-d que la transmission des informations se fait dans les 2 sens.

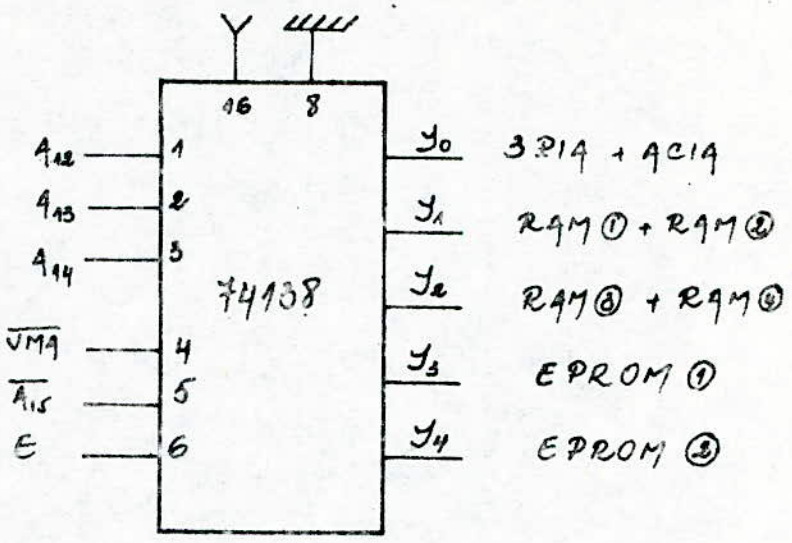
On utilise les SN 74640, ce sont des buffers inverseurs à 8 entrées qui comportent 2 lignes de commandes.

La broche 1 est commandée par le signal R /W qui indique dans quel sens l'information circule.

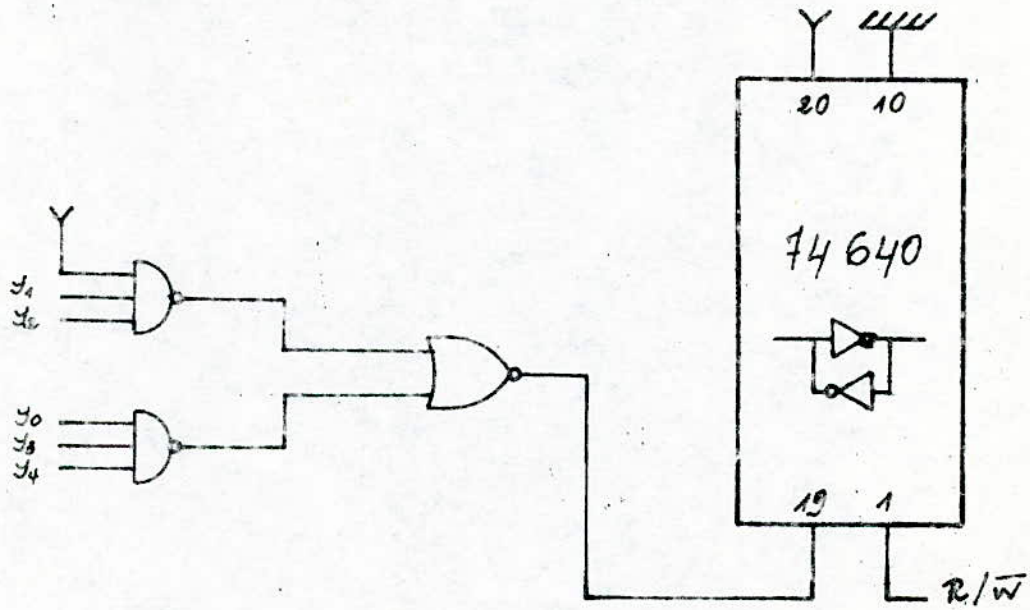
La broche 19 est commandée par tous les selections de boitiers CS réunis à travers une porte AND à 8 entrées et qui permet le fonctionnement du buffer dans les 2 sens dont le cas où un des boitiers est sélectionné.

2.2.3.2. LES BUFFERS D'ADRESSES

Ce sont des buffers unidirectionnels, c-à-d que la transmission des informations se fait dans un seul sens, c'est le cas des adresses et des lignes de contrôles.



CIRCUIT DE DECODAGE.



VALIDATION DU BUFFER DE DONNEES

Les buffers utilisés sont du type MC 8T 95, les broches 1 et 15 sont reliées à la masse , c-à-d que les buffers sont validés tout le temp.

2.2.4. LE DECODAGE D'ADRESSE :

Pour faciliter l'adressage des boitiers, on a utilisé un circuit du type SN 74138, c'est un décodeur 1 parmi 8, donc on peut accéder à 8 boitiers différents. Comme on a plus de 8 boitiers une ligne peut valider 2 où 3, ce qui nous amene à faire un adressage adéquat pour faire la différence entre ces boitiers.

Ce décodeur est validé par la ligne d'adresse A/5 et le signal d'horloge E, et selon la combinaison de A/2, A/12, A/4 on a l'une des sorties qui est selectionnée.

La sortie Y0 du décodeur valide les 3 PIA et l'ACIA, la différentiation entre ces boitiers se fait par les lignes d'adresses A4, A5, A6, A7.

Les sorties Y1, Y2 valident respectivement RAM 1, RAM2 et RAM3, RAM4 et sont différentiées par la ligne d'adresse A

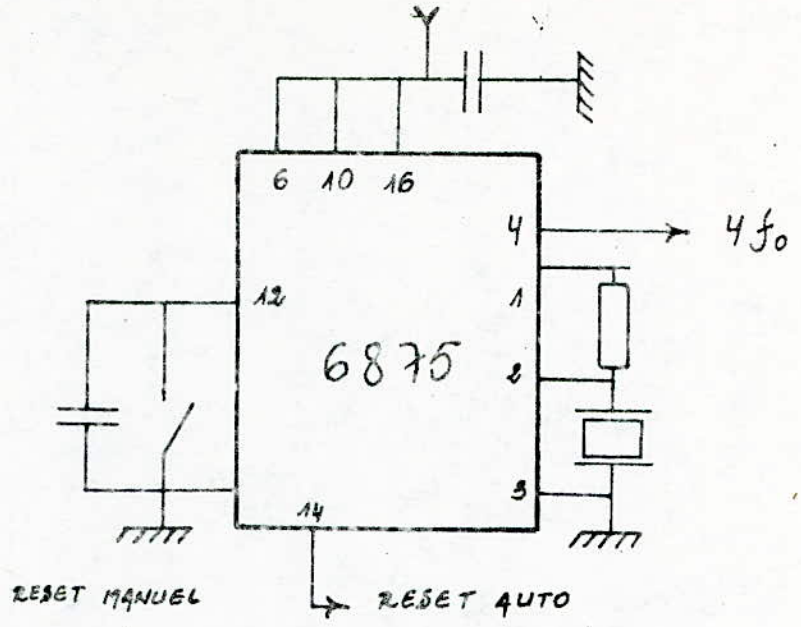
Les EPROM utilisés de capacité de 4K sont directement validés par les sorties Y3, Y4 du décodeur.

2.2.5. CIRCUIT D'HORLOGE ET INITIALISATION

Le circuit intégré utilisé pour le circuit d'horloge est le MC 6875, qui est prévu pour fournir la fréquence 4 f_o pour le microprocesseur.

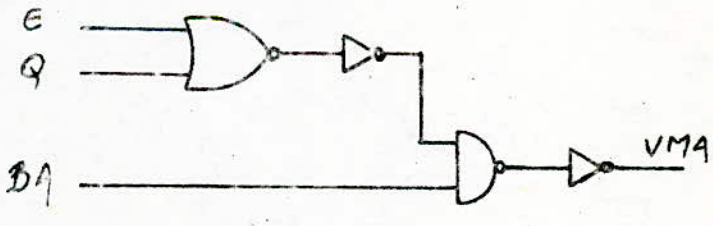
Cette horloge comporte un oscillateur intégré, ainsi qu'un quartz et une résistance déterminant la fréquence de fonctionnement

Le microprocesseur 6809 fonctionne à la fréquence 1 MHZ donc on doit utiliser un quartz de 4 MHZ pour attaquer la broche EXTRAL du microprocesseur.



CIRCUIT D'HORLOGE et D'INITIALISATION.

E	Q	BA	VMA
0	0	x	0
0	1	0	1
1	0	0	1
1	1	0	1
x	x	1	0



CIRCUIT DU SIGNAL VMA

Les entrées innutilisées ont été branchées soit à la masse, soit à Vcc pour qu'elles n'affectent pas le fonctionnement de l'horloge.

La broche 12 ou POWER / RESET du MC 6875 est utilisée pour le circuit de réinitialisation mannelle. Cette broche est reliée à la broche 37 ou RESET du microprocesseur.

Lorsque l'interruption est actionnée, POWER/RESET est mis à la masse, la broche 14 passe à "0" d'où la réinitialisation du système.

2.2.6. SIGNAL VMA :

Ce signal generé à partir de E, Q et BA , permet la validation des adresses à partir des fronts montants de E.

Il sert essentiellement à la commande des périphériques du microprocesseur.

2.2.7. ORGANISATION DE LA MEMOIRE

La zone mémoire occupée est de 17 K octets ;

- 4 boitiers RAM TMM 20/6 de 2 K Octets.
- 2 boitiers EPROM NE 2732. de 4K Octets
- 1 boitier EPROM NE 2708 de 1 K Octets, utilisé pour le moniteur.

Le champ mémoire de 0000 à 7FFF de 32 K Octets est réservé à une extension futur.

La zone occupée est à partir de 8000 à FFFF avec un espace mémoire non utilisé de D000 à FBFF c-à-d 11 K octets.

La zone mémoire occupée par le moniteur est de [illegible] Octets à partir de l'adresse FC00. Il a pour rôle la gestion du clavier afficheurs.

A_{15}	A_{14}	A_{13}	A_{12}	A_{11}	A_{10}	A_7	A_6	A_5	A_4		
0	x	x	x	x	x	x	x	x	x	7FFF	32K non utilisé
1	0	0	0	x	x	0	0	0	1	8010	PIA ①
1	0	0	0	x	x	0	0	1	0	8020	PIA ②
1	0	0	0	x	x	0	1	0	0	8040	PIA ③
1	0	0	0	x	x	1	0	0	0	8080	ACIA
1	0	0	1	0	x	x	x	x	x	9000	RAM ①
1	0	0	1	1	x	x	x	x	x	9800	RAM ②
1	0	1	0	0	x	x	x	x	x	A000	RAM ③
1	0	1	0	1	x	x	x	x	x	A800	RAM ④
1	0	1	1	x	x	x	x	x	x	B000	EPROM ①
1	1	0	0	x	x	x	x	x	x	C000	EPROM ②
1	1	0	1	x	x	x	x	x	x	D000	11K non utilisé
1	1	1	1	1	1	x	x	x	x	FC00	MONITEUR

TABLEAU D'ADRESSAGE et REPARTION DU CHAMP MEMOIRE.

La zone mémoire occupée par les EPROM est de 8 K octets à partir de l'adresse B000. Ces EPROM contiennent les programmes des calculs de , COS, SIN, LOG, ...

La zone mémoire occupée par les RAM est de 8K Octets à partir de l'adresse 9000. Ces RAM sont utilisées pour le stockage des données.

2.2.8. CHAMP MEMOIRE ET ADRESSAGE

L'adressage de la carte mémoire est répartie comme l'indique le tableau.

[illegible] :

* DECOUVRIR , C'EST VOIR CE QUE TOUT LE MONDE A VU ET PENSER CE QUE PERSONNE N'A PENSE .

C H A P I T R E 3

REALISATION DU PERIPHERIQUE

Le périphérique utilisé pour communiquer avec le microprocesseur est un clavier avec des afficheurs, l'un permettant de donner des informations au microprocesseur, l'autre d'afficher les informations reçues du microprocesseur.

Ceci pour la partie matérielle, la partie logicielle permet la gestion du clavier et des afficheurs, ce sont ces parties matérielle et logicielle que nous allons étudier.

Ce dispositif clavier - afficheurs à toutes les possibilités que peut avoir ce type de périphérique avec 24 touches et 6 afficheurs, 4 pour les adresses et 2 pour les données.

3.1. PARTIE MATERIELLE :

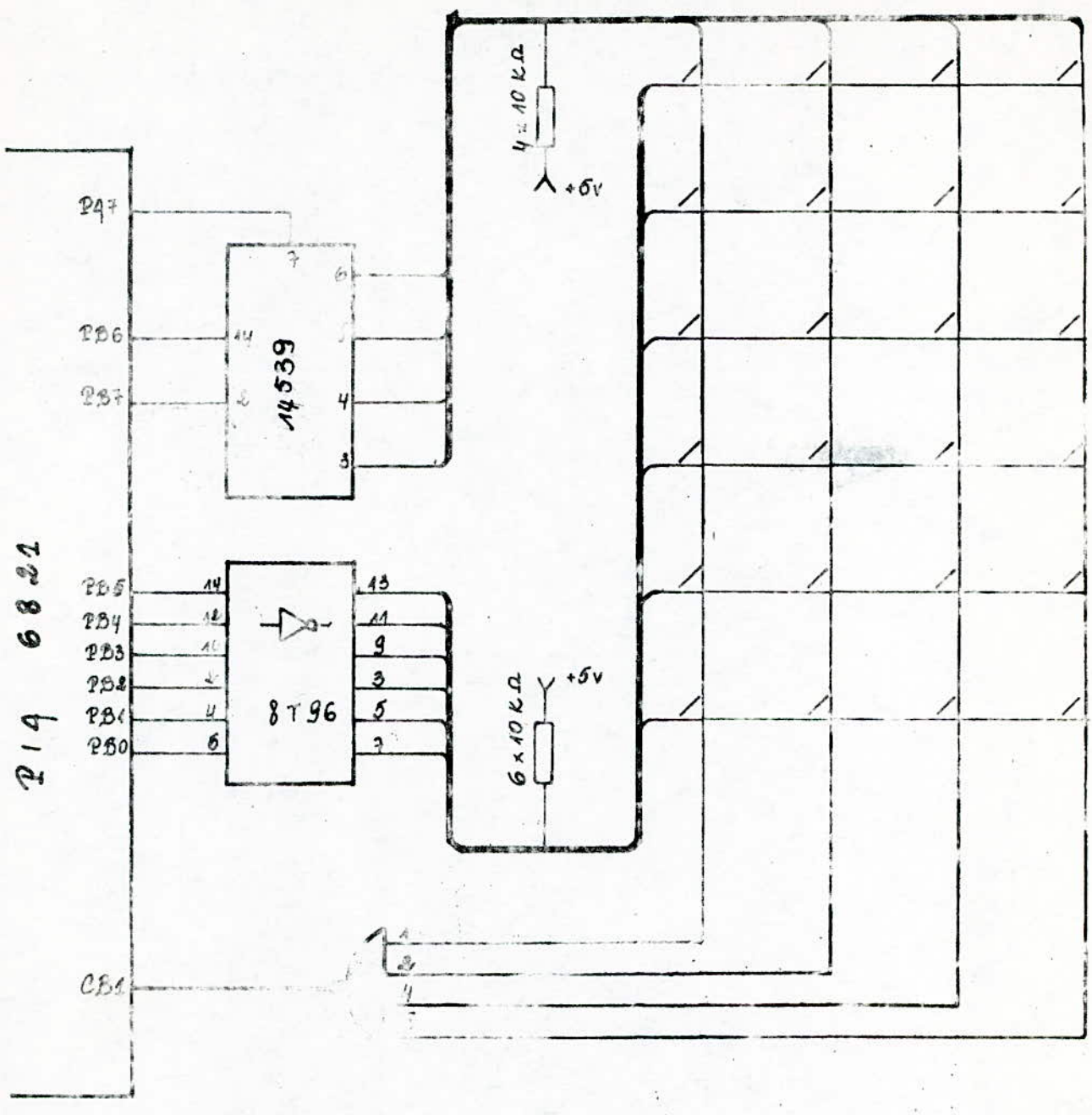
L'interface entrée/ sortie entre le microprocesseur et le clavier-afficheurs est un PIA, qui offre la possibilité de gestion du clavier et des afficheurs.

3.1.1. CLAVIER :

C'est un clavier hexadecimal, qui comprend 24 touches, dont 16 pour les données numériques, et une possibilité d'avoir en plus des 8 touches de commandes restantes sur le clavier, 16 autres, touches de commandes ou de controle par l'intermediaire d'une touche "SHIFT" et ceci par logiciel.

Ce clavier est composé de 6 lignes et 4 colonnes, toutes ces lignes et colonnes sont reliés à + 5V à travers une résistance de couplage de 10 KΩ.

Le principe de fonctionnement du clavier est simple, dès qu'une touche est appuyée, un niveau bas "0" apparait sur la ligne et la colonne correspondante et c'est les états de toutes les lignes et les colonnes à cet instant qui déterminent le code de la touche.



LIAISON PIA - CLAVIER

TABLE ①

CODE	TOUCHE
01	0
02	1
42	2
82	3
04	4
44	5
84	6
08	7
48	8
88	9
C8	A
C4	B
C2	C
C1	D
81	E
41	F

TABLE ②

CODE	TOUCHE
10	RESET
50	
90	
D0	
20	
60	
A0	
E0	

TABLE D'EQUIVALENCE

CODE DE LA TOUCHE

- HEXADECIMAL

L'interface entrée/sortie entre le microprocesseur et le clacier étant le P/A, d'où la possibilité d'utiliser plusieurs lignes pour le clavier. Sachant que le port A est utilisé pour l'affichage sauf PA7, donc on a le port B plus la ligne PA7 qui sont disponibles pour le clavier, mais ceci n'est pas suffisant car le clavier a besoin de 10 lignes (6 lignes et 4 colonnes).

Pour resoudre le problème, un multiplexeur est tout indiqué le MC 14539, les 4 colonnes du clavier sont reliées à l'entrée du multiplexeur et selon la combinaison qu'on a aux bits PB6 et PB7, la sortie du multiplexeur ou le bit PA7 passe à "0" pour indiquer la bonne combinaison des colonnes, les 6 lignes de PB0 à PB5 étant reliés aux 6 lignes du clavier par des buffers inverseurs PC 8 T96.

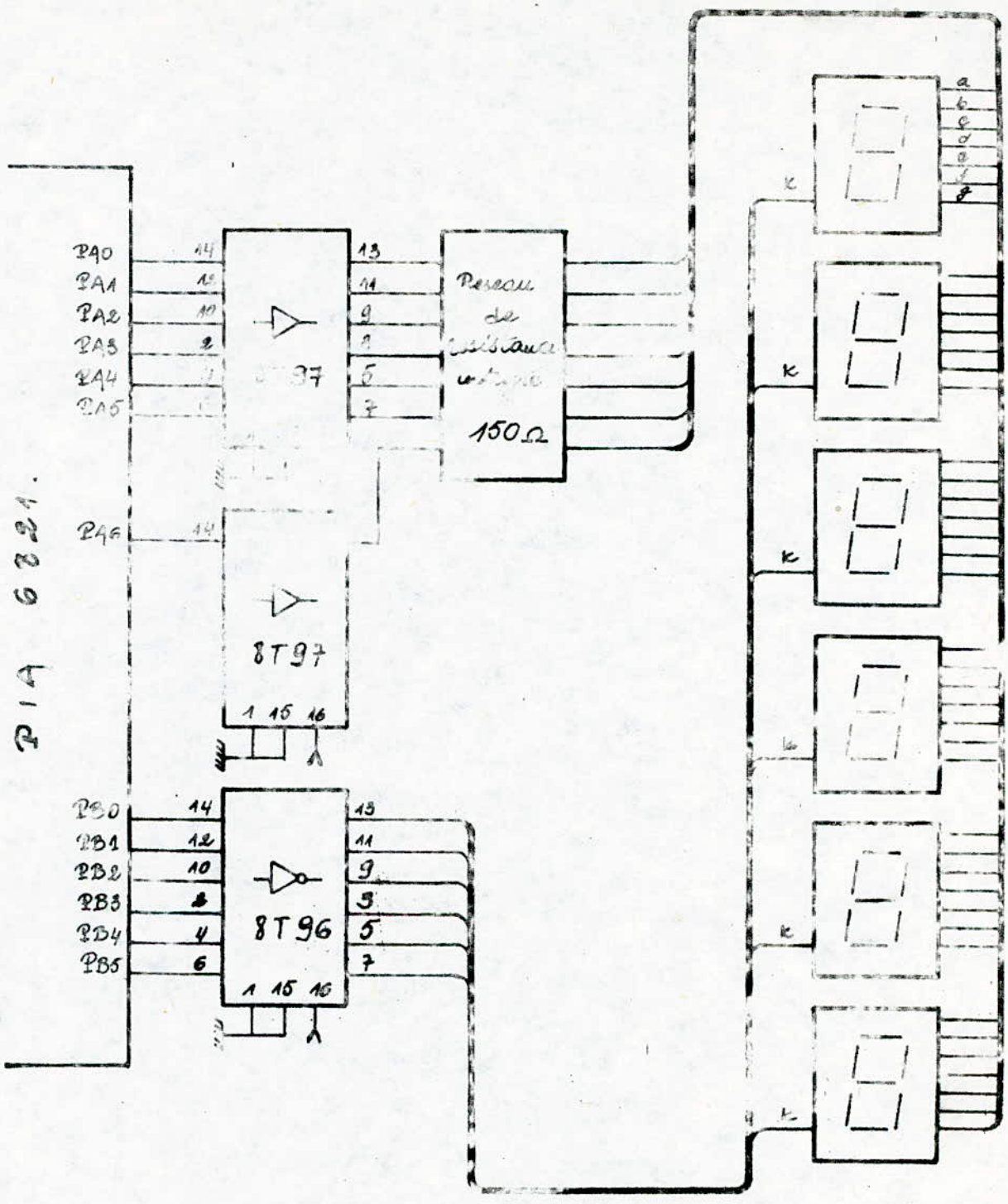
Pour indiquer qu'une touche a été enfoncée, on relie à la ligne d'interruption CB1, les 4 colonnes à traver une porte NAND à 4 entrée le SN 7420, ainsi CB1 passe à 1 lorsqu'une touche est enfoncée.

3.1.2. FONCTIONNEMENT DU CLAVIER

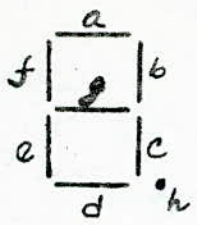
Lorsqu'une touche est enfoncée, CB1 passe à l'etat "1" le microprocesseur est interrompue, pour déterminer le code de la tou touche il commance par chercher la colonne en déterminant la combinaison de PB6 et PB7, puis il détermine la ligne, et ceci en envoyant des niveaux haut successivement de PB0 à PB5 des qu'il y a un changement sur les colonnes c-à-d PA 7=0, il localise la position de la ligne ainsi il détermine le code de la touche en mémorisant le port B.

3.1.3. AFFICHEUR

Pour les afficheur à 7 segments, on a utilisé des DL 500 à cathode commune, un niveau haut appliqué à l'anode d'une provoquerait l'allumage du segment correspondant. Un niveau haut appliqué à la cathode éteindrait tous l'afficheur, c'est par la cathode qu'on sélectionne l'afficheur.



LIAISON PIA - AFFICHEURS



.	g	h	e	d	c	b	a

TABLE (V)

CODE	HEXA
3F	0
06	1
5B	2
4F	3
66	4
6D	5
7D	6
07	7
7F	8
6F	9

CODE	HEXA
77	A
7C	b
39	C
5E	d
79	E
71	F
40	-
50	r
5C	o
73	P

CONVERSION CODE 7 SEGMETS

- HEXADECIMAL.

On utilise 6 afficheurs, 4 pour l'affichage de l'adresse et 2 pour l'affichage des données.

Les cathodes des 6 afficheurs sont reliés au port B de PB0 à PB5 (où aux lignes du clavier) à travers des buffers inverseur MC 8 T 96.

Les anodes des 7 segments sont reliés au port A de PA0 a PA6 à travers des buffers non inverseur MC 8 T 97, ces buffers sont validés tous le temps c-à-d les broches 1 et 15 sont à la masse c'est ce qu'on appelle :"Affichage multiplexé".

3.1.4 FONCTIONNEMENT DE L'AFFICHAGE :

Pour afficher une adresse ou une donnée , il faut valider les afficheurs concernes d'abord et ceci à travers le port B, puis afficher la donnée voulue dont le code 7 segment est sur le port A.

Une temporisation de la donnée affichée est necessaire pour la persistance des impressions retiniennes, ceci fait que l'oeil à l'impression que tous les afficheurs sont allumés simultanement et non sequentiellement.

La temporisation et le decalage à gauche des données dans l'affichage sont fait par ligiciel.

3.1.5. PARTIE REALISATION:

Faute de clavier , on a realisé nous même, les touches utilisés n'etant pas trés solide, d'où on a un effet de rebondissement plus important , qu'on peut d'ailleur l'éliminer avec un système anti-reboud à bascule R5, mais on a préféré l'éliminer de façon logiciel et ceci en mettant un certain temps de l'ordre de 20 ms avant de prendre en compte la touche enfoncée.

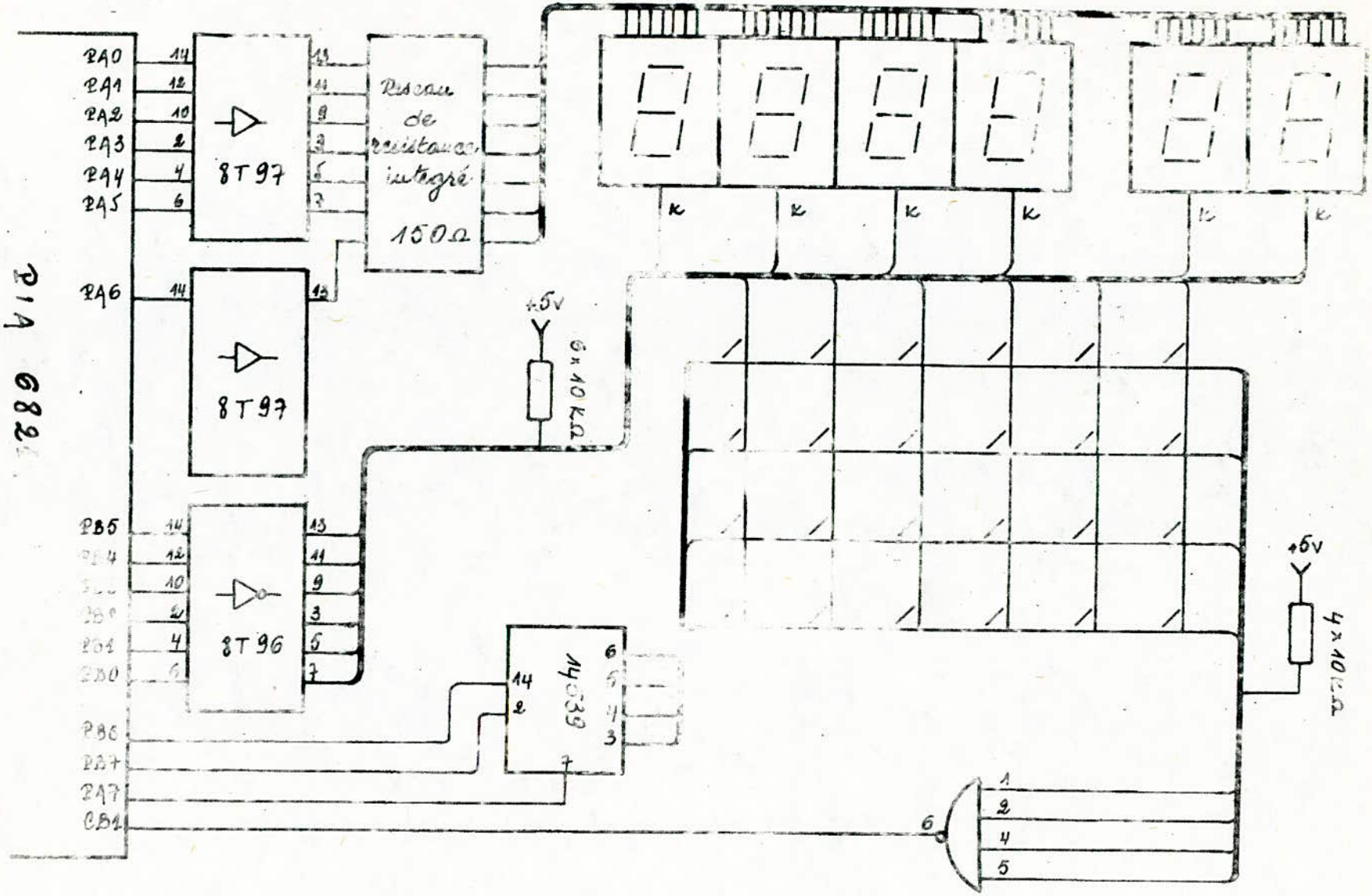
3.2. LOGICIEL DE GESTION :

La partie materielle etant fixé, ce qui nous permet de concevoir un logiciel qui repond aux conditions que nous nous sommes déjà posés pour la realisation d'unt tel dispositif.

Les positions memoires reservées par les différents programmes ainsi que le code des validations des afficheurs sur le port B sont :

port B

M o :



SCHEMA GENERAL DU PERIPHERIQUE.

			PORT B
M 0	: Code 7 segment	1er afficheur adresse	01
M 1	: " "	2ème " "	02
M 2	: " "	3ème " "	04
M 3	: " "	4ème " "	08
M 4	: " "	1er afficheur donné	10
M 5	: " "	2ème " "	20

M I : Compteur du nombre d'interruption

M 2 : Code d'une touche ducclavier

M3 : Chiffre en hexadecimal equivalent au code de la touche.

Les différentes zones memoires reserves aux tables sont :

TABLE I : X1 (equivalent en hexadécimal du code d'une touche)

TABLE II : X2 (correspondance en commande du code d'une touche)

TABLE III : X3 (adresse de debut de la commande).

TABLE IV : X4 (adresse du debut de la commande obtenu par SMIFT)

TABLE V : X5 (equivalent en caractère du code 7 segments).

Les sous programmes utilisés sont :

" RESER" : Initialisation de tous le système

" GEST " : Programme de gestion du périphérique

" DEC " : Sous programme de decodage d'une touche

" ETC " : S/P d'exécution d'une touche de commande

"SHIFT" : S/P d'extension du nombre de touches de commandes

"4 DGT" : S/P d'affichage des adresses

"2 DGT" : S/P d'affichage des données

"ERRO" : S/P d'affichage d'erreur.

"TIRET" : S/P d'affichage du tiret

"DELAI 20": S/P de temporisation de 20ms

"DELAI 1" : S/P de temporisation de 1ms

"POPC" : S/P d'affiche indiquant qu'il n'y a pas d'opération de commande pour cette touche.

3.2.1. INITIALISATION DU SYSTEME "RESET" :

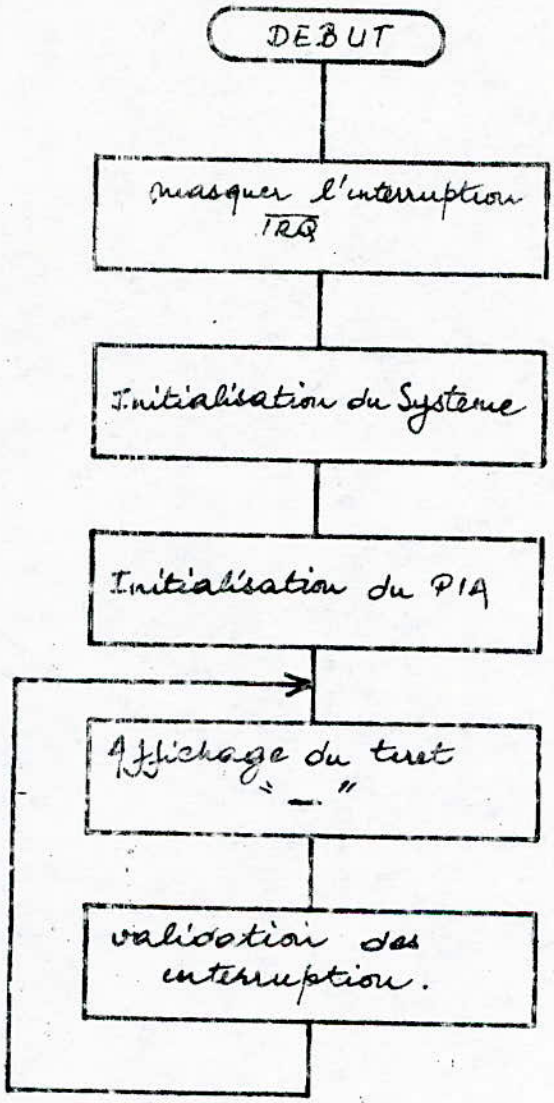
L'initialisation du systeme se fait par le programme de "RESET" qui consiste à :

- mettre à zero toutes les positions memoires utilisés par les programmes de gestion du clavier.
- mettre un masque d'interruption sur RQ pour que la phase d'initialisation ne soit pas programmant.

  . le port B en sortie et en positionnant toutes ces sorites à"1
  . la ligne CA1 active à la transition positive
  . la ligne PA7 en entrée, les autres lignes du port A en sortie

- afficher un tiret sur les 6 afficheurs indiquant à l'opérateur que la phase d'initialisation est terminée et qu'il a le controle du système.
- enlever le masque en validant les interruptions.

ORGANIGRAMME D'INITIALISATION "RESET".

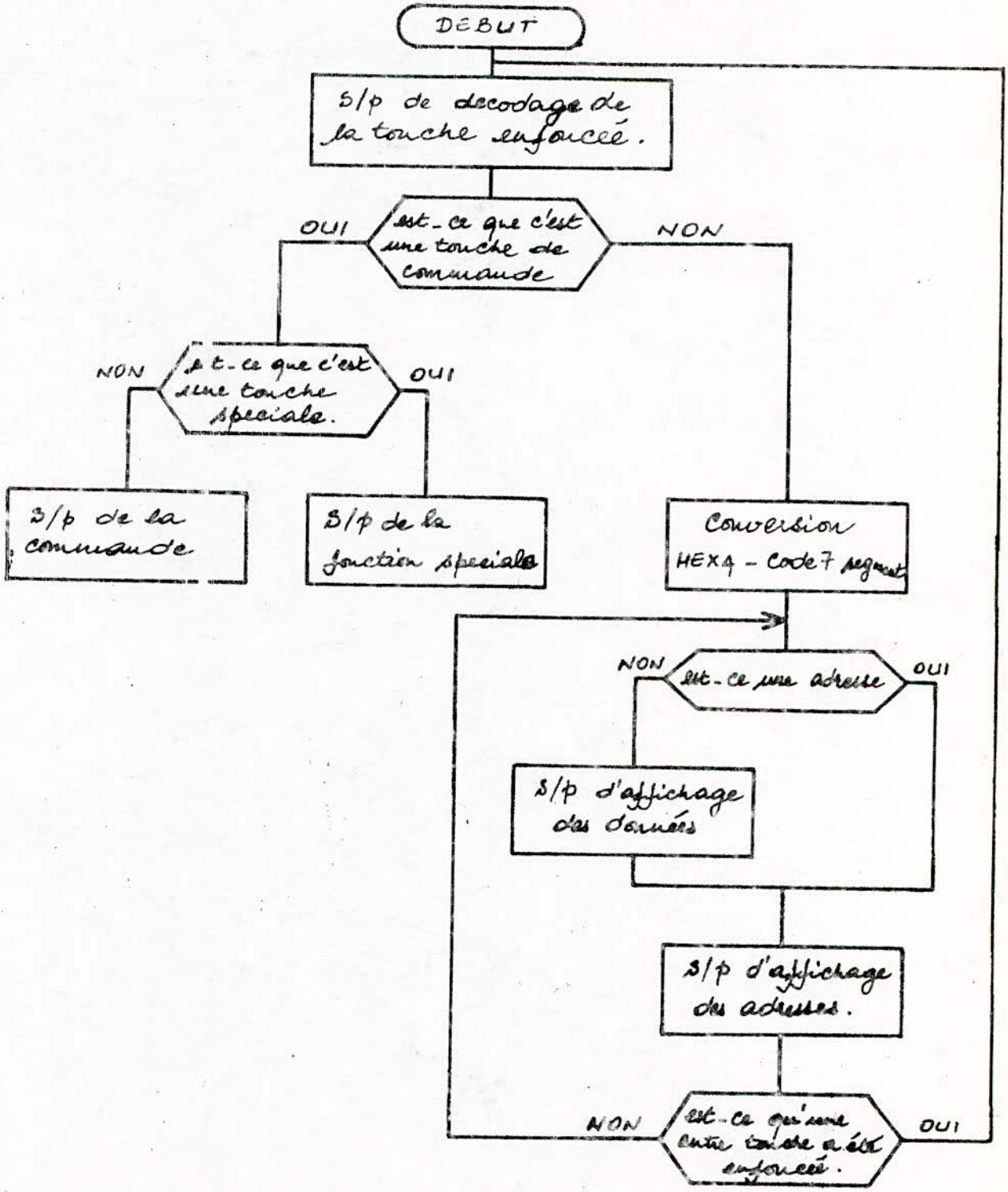


3.2.2. GESTION CLAVIER - AFFICHEURS "GEST"

Cette etape s'effectue sous intérruption, en effet si la touche est enfoncée, la ligne CA1 passe à "1", ce qui interrompt le microprocesseur pour exécuter le programme de l'interruption RQ qui est le programme de gestion et qui consite à :

- Décoder la touche enfoncée faisant appelle au S/P "DEC"
- brancher au S/P "ETC" d'exécution de commande si c'est une touche de commandbe .
- faire le decalage necessaire et l'afficher sur les afficheurs d'adre d'adresse si c'est une adresse.
- l'afficher sur les afficheurs de données si c'est une donnée.
- tester si une touche a été enfoncée, si non il faut réafficher le contenu des afficheurs pour permettre un allumage continu, et attendre jusqu'à la prochaine interruption.

ORGANIGRAMME DE GESTION DU PERIPHERIQUE "GEST".

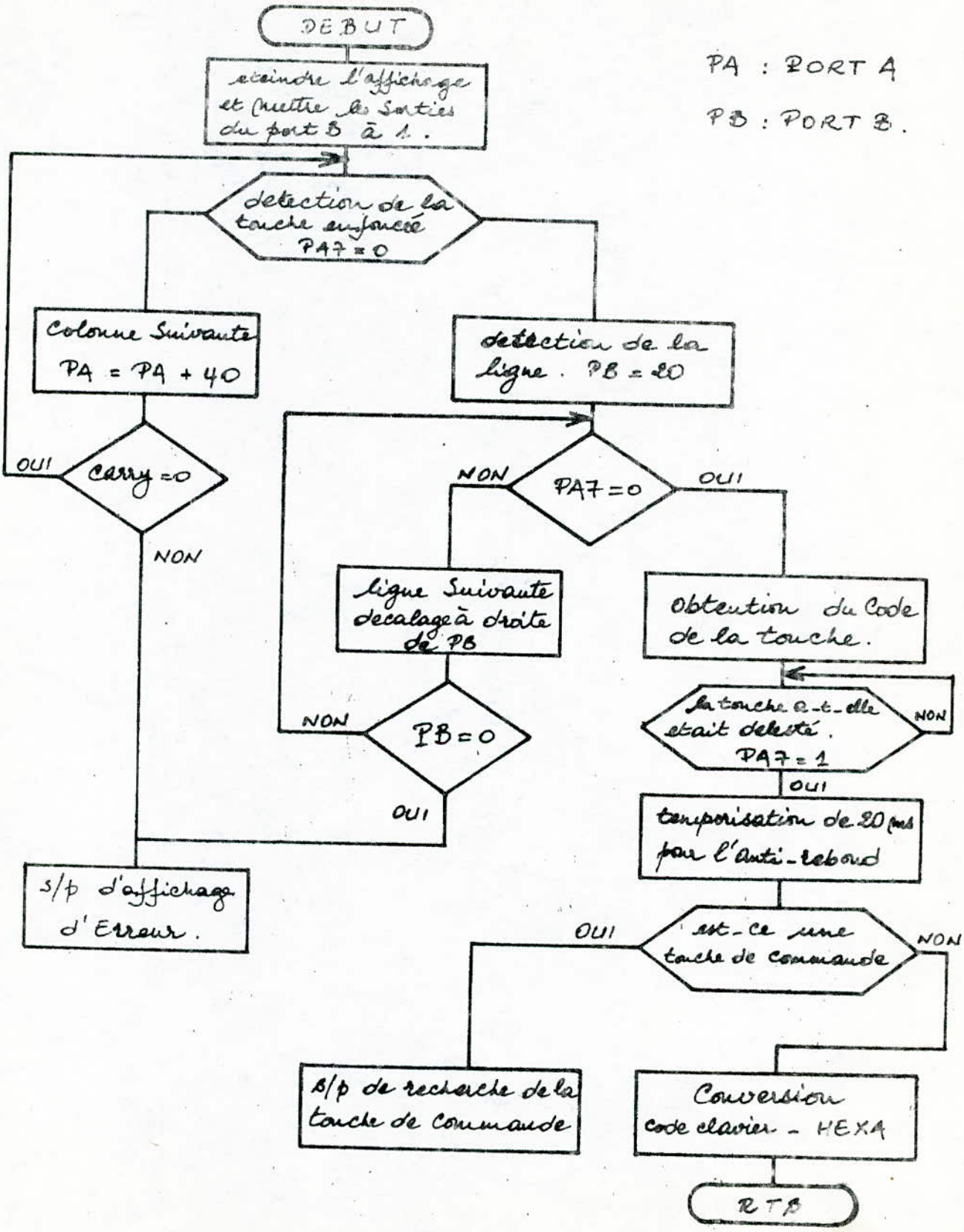


3.2.3. S/P DE DECODAGE "DEC" :

L'analyse de la touche se fait en plusieurs etapes , ou procede de la façon suivante :

- on determine la colonne, en cherchant la combinaison exacte de PB6 et PB7, qui correspond à la colonne, si le bit PA7 passe "0" ceci nous indique que la combinaison PB6 et PB7 correspond bien à la colonne qui est passée à "0".
- On détermine la ligne, et ceci en ne mettant qu'une seule ligne à zero "0" , ce qui fait que si cette ligne n'est pas celle de la touche enfoncée, la colonne est à "1" donc PA7 est à "1" , donc on passe à la ligne suivante jusqu'à avoir PA7 a "0".
- On attend jusqu'à ce que la touche soit désesté et ceci pour ne pas prendre en compte plusieurs fois la même touche.
- On fait une temporisation d'environ 20 ms afin de parer au phénomène de rebondissement dû à la touche, ce qui evitera au systeme de recueillir des f qui induiront le programme en erreur.
- on teste la touche
  - si c'est une touche de commande ou se branche sur le S/P "ETC".
  - Si c'est une touche de donnée, on détermine l'equivalent hexadecimal de la touche appuyée.

ORGANIGRAMME DE DECODAGE D'UNE TOUCHE "DEC".



3.2.4. S/P D'EXECUTION DE COMMANDE "ETC"

Pour toutes les touches de fonctions, on a des touches de commandes et de controles, pour une exécution du nombre des touches de commandes, on a utilisé une touche SHIFT pour permettre ça.

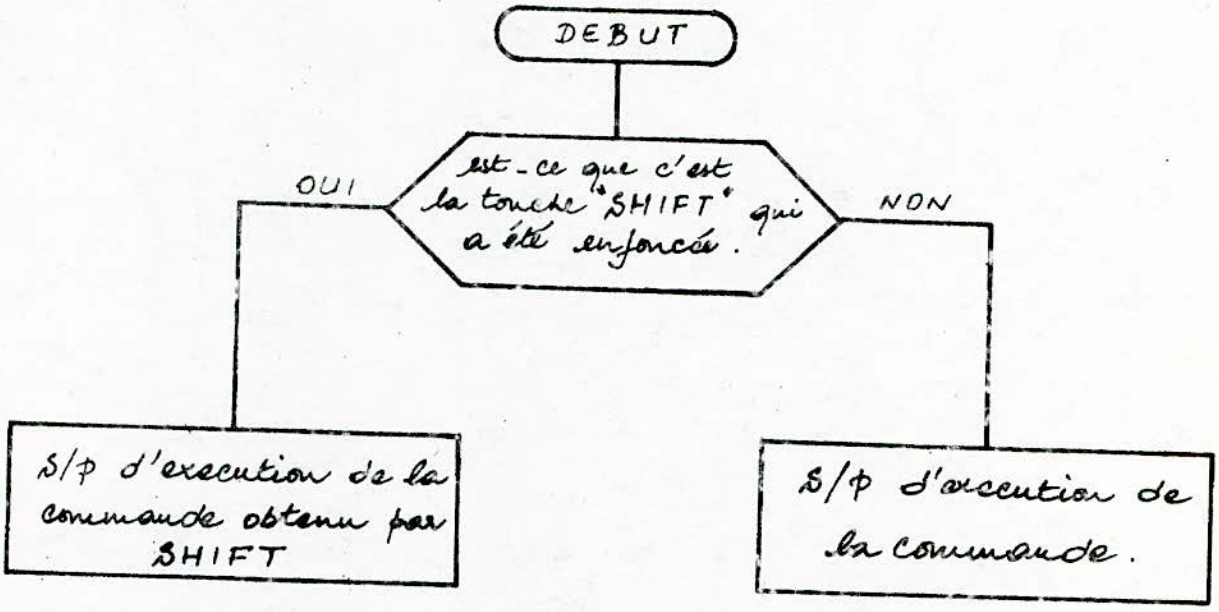
Les étapes suivies pour l'exécution de la touche de commande sont :

- on teste si la touche SHIFT a été enfoncée.

  . si oui on attend jusqu'à ce qu'une seconde touche soit enfoncée pour déterminer la commande à exécuter, puis on cherche l'adresse de début de programme de cette commande dans la table IV pour son exécution.

  . Si non on cherche directement l'adresse de début de programme de cette commande dans la table III pour son exécution.

ORGANIGRAMME D'EXECUTION DE COMMANDE " ET(.



3.2.5. S/P D'AFFICHAGE DE DONNEES

Il y a 2 S/P programme d'affichage :

- le S/programme d'affichage des adresses "4DGT" et qui consiste à afficher les 4 premiers chiffres introduit sur le clavier et si l'opérateur s'est trompé, il devra faire une réinitialisation du systeme puis il réintroduit ses chiffres et fait la correction qu'il faut. Ce S/programme peut être controlé par les différents programmes de commandes.

- le S/programme d'affichage des données '2DGT" consiste à afficher le 5ème et le 6ème chiffre introduit après les 4 premiers de l'adresse, ceci si aucune commande n'a été actionnée . Si non, ce S/programme sera controlé par les différents programmes de commandes.

Ces 2 S/programme d'affichage appelle un autre S/Programme de temporisation de 1ms qui est nécessairepour que l'on voie la donnée s'afficher et pour que l'affichage nous parait allumés continuellement il faut faire le raffraichissement des données dans les 60ms (1/15 secondes) qui suit l'affichage.

3.2.6. S/P D'AFFICHAGE DES CARACTERES :

On utilise 3 S/programmes :

- l'affichage des tirets sur les 6 afficheurs pour le S/P "RESET"
- l'affichage de ERRO sur les 4 afficheurs d'adresses en cas de message d'erreur.

- l'affichage de POPC sur les 4 afficheurs d'adresses en cas où on appuie sur une touche de commande, ceci nous indique que pour to cette touche il n'y a pas d'opération de commande.

Le principe d'affichage de ces 3 S/programmes est pratiquement le même, on valide avant l'afficheur concerner, puis on affiche le caractere correspondant en le maintenant pendant 1ms puis on passe au suivant et ainsi de suite.

3.2.7. S/P DE TEMPORISATION :

On a 2 types de temporisation , celle de 20 ms pour éviter l'effet de rebondissement, et celle de 1ms nécessaire pour l'affichage de donnée.

Le programme de ces 2 temporisations est le même à part que le nombre de boucle de base qui est dans l'index X qui sera différent.

- fréquence d'horloge = 1 µs
- durée d'une boucle de base = (4+4+2) = 10 µs

Pour avoir environ 20 ms = 2000.10 µs, il faut 2000= $ 700 boucles de base , donc l'index X=700

Pour avoir environ 1 ms = 100.10 µs, il faut 100= $64 boucles de base donc x=64.

3.2.8. - CONCLUSION :

Nous pensons avoir donner avec ces différents programmes au clavier - afficheurs, les pleines possibilitées que peut avoir ce type de dispositif, il reste à indiquer les différentes touches de commandes et controles pour compléter ce périphérique.

Nous avons essayés de faire un clavier afficheurs standardisé c-à-d que sa manipulation ne differe pas beaucoup des autres micro-ordinateur, pour quelqu'un qui a déjà manipuler ce genre de système il ne trouvera aucun problème pour son utilisation.

Si l'opérateur venait à se tromper sur l'adresse, il doit éteindre l'affichage avec un "RESET" et réintroduire ses données, hormis ce détail son fonctionnement est pratiquement le même que les autres micro-ordinateur.

CONCLUSION :

Les objectifs que nous nous sommes fixes avant d'en-
-tamer ce travail ont été pratiquement atteints .

Nous xxavons elaboré les programmes de COSX , SINX;...
Et ont été mis au point les programmes fixxix de gestion
du clavier et des afficheurs . Cet ensemble de programmes
finis et implantés dans une carte maitraisse qui comporte
le CPU , les buffers ainsi que la logique de commande .

La deuxiemme carte contient 3 PIA destinés à assurer
l'echange de l'information entre le CPU et les peripheriques
ainsi que le transfert de donneés vers la carte à tester,
sur la méme carte est implanté un ACIA et des memoires
RAM etEPROM , dans ces dernieres,sont implantés les progra-
-mmes des applications cités ci haut . Le clavier et les
afficheurs sont implantés sur une carte exterieur .

La realisation de la carte CPU nous a posé quel-
ques problemes , en premier lieu des de synchronisation
entre notre carte CPU et la carte DEBUG du systeme de
devellopement qui gere tous le systeme ainsi que les peri-
-pheriques dans notre cas la visualisation .

Par la suite la methode d'emulation offerte par
l'EXORCISER nuos a permis de verifier la bonne marche de
la carte .

Ceci n'etant pas tres pratique notre promoteur nous
a proposé de faire notre propre moniteur et le peripheri-
-que necessaire pour son fonctionnement .

La gestion du clavier et des afficheurs existante

dans le moniteur est suffisante pour la confirmation de la bonne marche de la carte .

Il est evident que le microordinateur realise estun prototype qui demande certains ameliorations , dans le sens d'une implantation de tous les composants ainsi que les peripheriques sur une meme carte et sur circuit imprimé .

* DEFINITION D' UN BON PROFESSEUR : QUELQU'UN QUI SE REND PROGRESSIVEMENT INUTILE .

4.ORGANIGRAMMES et PROGRAMMES :

Ce chapitre traite les programmes et organigrammes utilises dans notre projet,dont les principales parties sont:

- les organigrammes et programmes du moniteur,qui traite la gestion du clavier et des afficheurs .
- les organigrammes et programmes sur les operations a virgule flottante.
- les programmes des fonctions elementaires,operation papillons,generation de donnees et produit scalaire.

4.1.OPERATION EN VIRGULE FLOTTANTE :

Les differentes positions memoires utilisees pour les programmes a virgule flottante sont:

- Premier operande	:	Ra	M1	B001
		Pa	M2	B002
- Deusieme operande	:	Rb	M3	B003
		Pb	M4	B004
- Resultats	:	Rc1	M5	B005
		Rc2	M6	B006
		Pc	M7	B007
- Signe du 1 operande	:		M8	B008
- Signe du 2 operande	:		M9	B009
- Signe du resultat	:		MA	B00A
- Registre intermediaire	:		MB	B00B
			MC	B00C
			MD	B00D
			ME	B00E

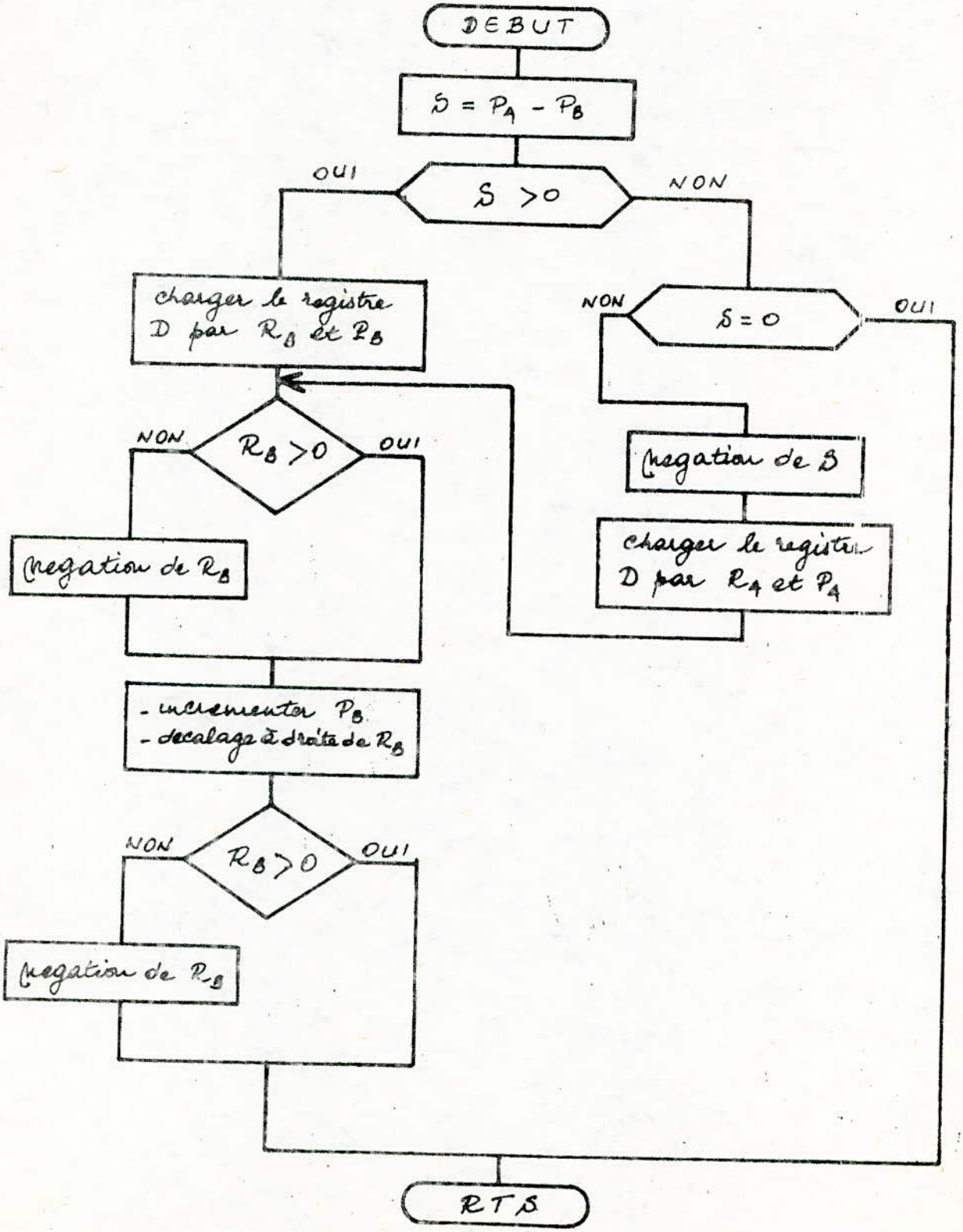
R : Mantisse
P : Exposant

Les zones memoires occupees par les differents s/programmes sont :

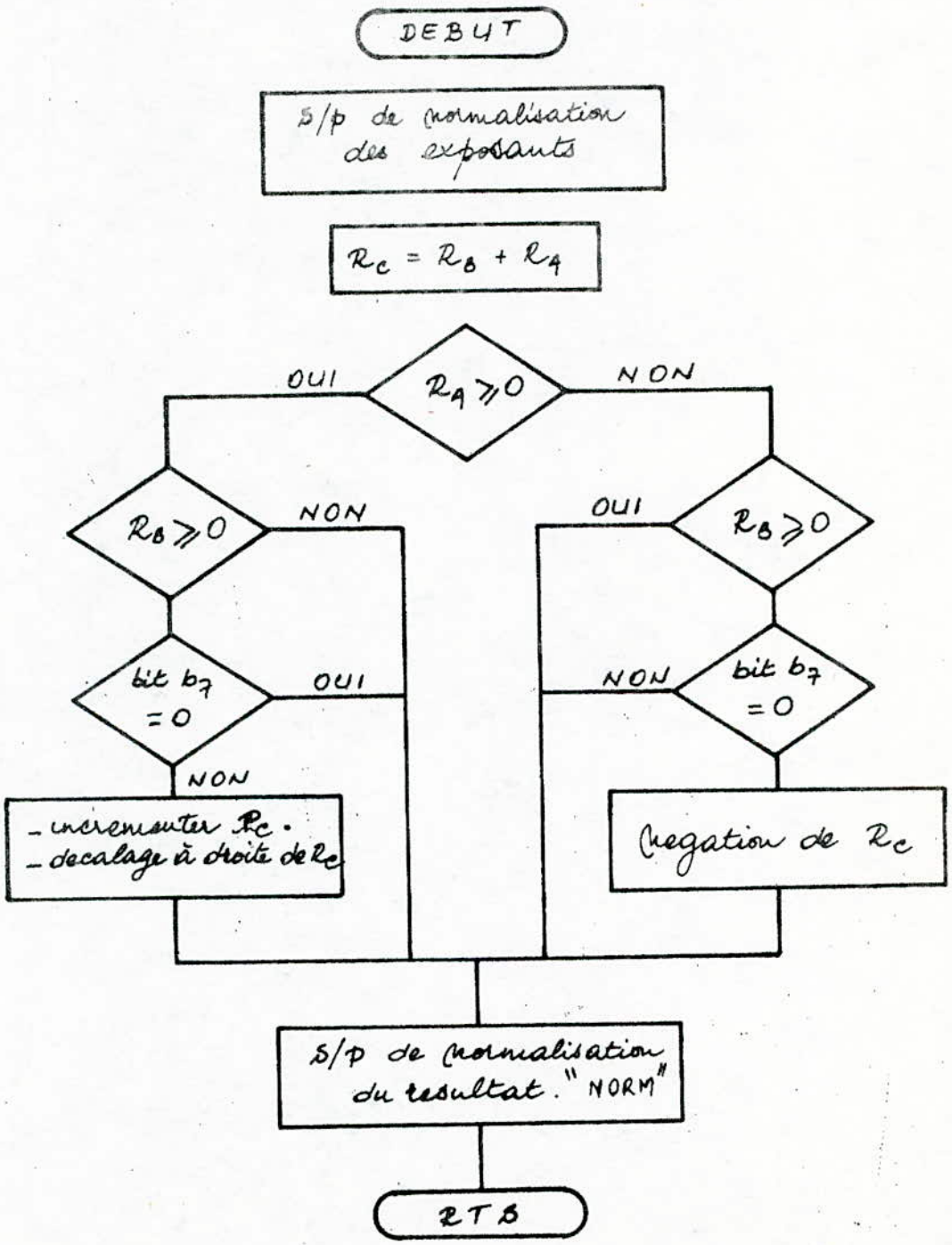
"NOREX"	:	B100
"ADD"	:	B160
"SOMME"	:	B1C0
"MUL"	:	B200
"NORM"	:	B260

4.1. OPERATIONS EN VIRGULE FLOTTANTE .

4.1.1. S/programme de normalisation des exposants "NOM



4.1.2. S/ programme d'addition en virgule flottante "ADD".



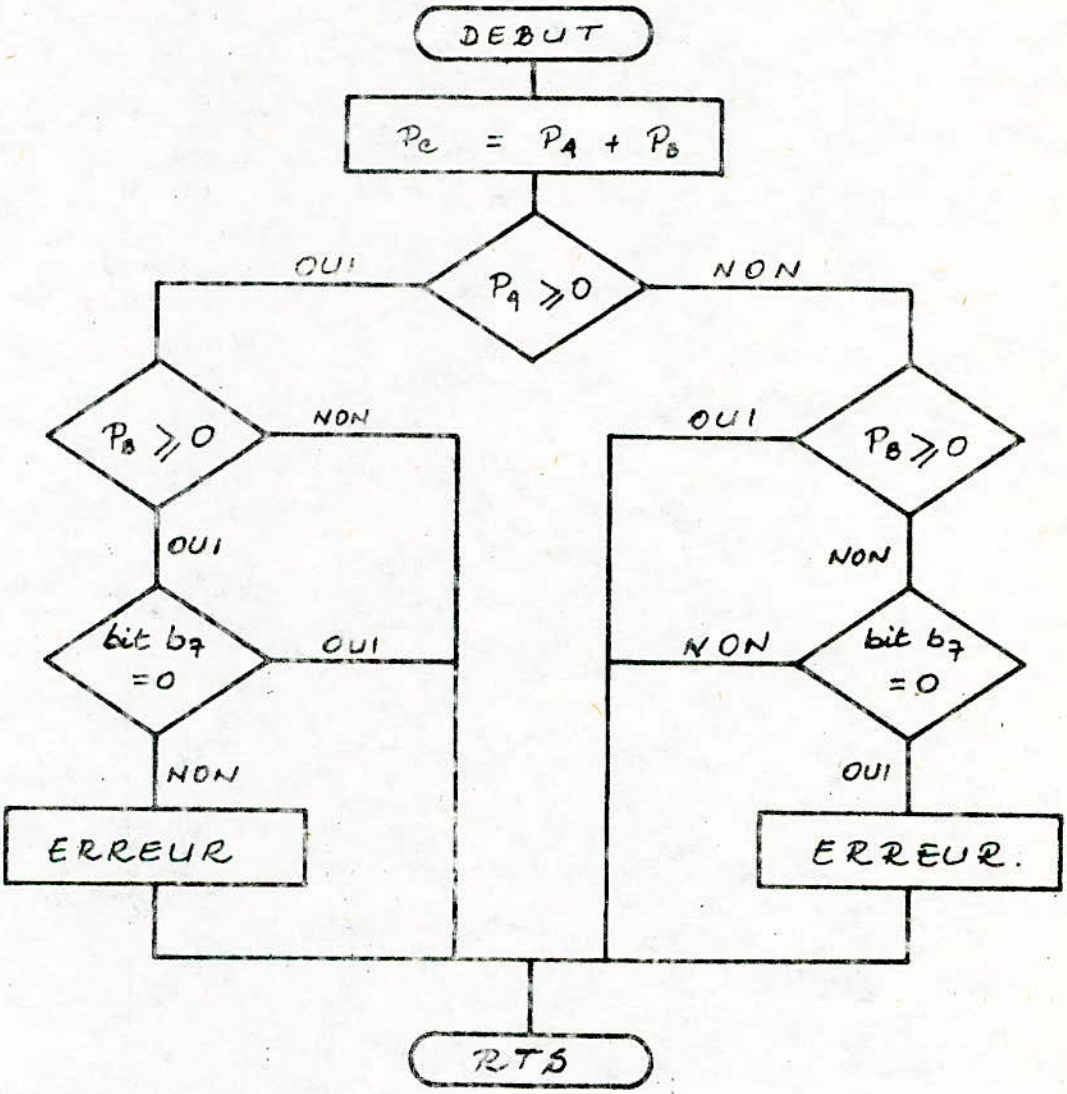
S/Prosramme "NOREX" : B100 1

```
        LDA     $M2
        SUBA    $M4         difference des exposants
        STA     $MB
        STA     $MC
        BLE     LP1         est ce que la difference est positive
        LDD     $M3
        TFR     D,X
LP6     BGE     LP2         est ce que la mantisse est positive
        NEGA
LP2     INCB                decalage de la mantisse a droite et
        LSRA                incrementation de l'exposant.
        DEC     $MB
        BNE     LP2
        CMPX    #$0000
        BGE     LP3         est ce que MB est nulle.
        NEGA
        STD     $MD
LP3     STD     $MD
        LDA     $MC
        BLE     LP4
        LDD     $MD
        STD     $M3
        BRA     LP5
        LDD     $MD
        STD     $M1
LP5     RTS
LP1     BEQ     LP5     est ce la difference est nulle
        NEG     $MB
        LDD     $M1
        BRA     LP6
```

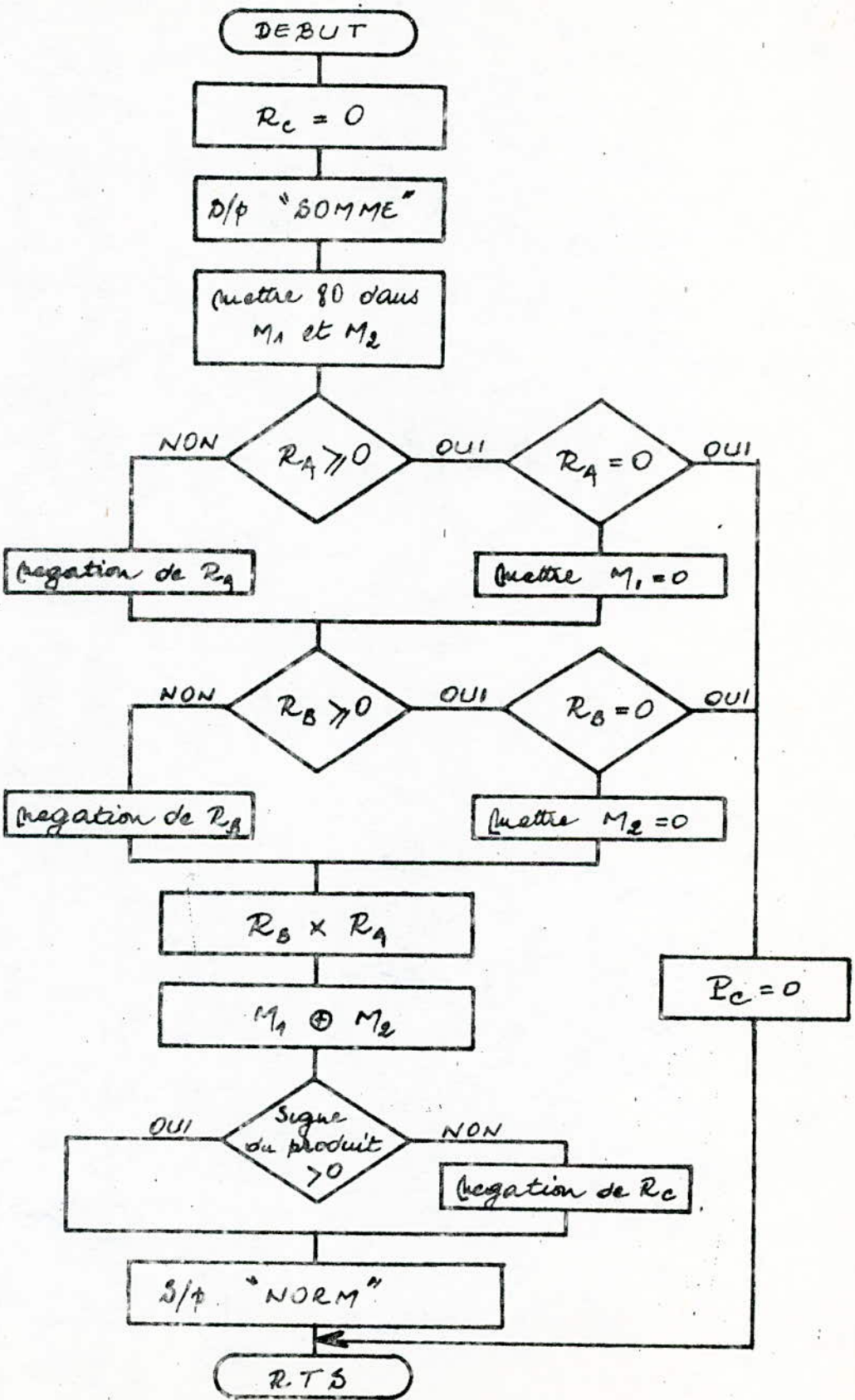
S/Prosramme "ADD" : B160

```
        CLR     $M6
        BSR     "NOREX"     S/Prosramme de normalisation des
        LDA     $M2         exposant.
        STA     $M7
        LDA     $M1
        ADDA    $M3         la somme des mantisses stocker dans M5
        STA     $M5
        LDA     $M1
        BGE     LP1         est ce que le 1 operande est positif
        LDA     $M3
        BGE     LP2         est ce que le 2 operande est positif
        LDA     $M5
        BITA    #$80
        BNE     LP2
        NEG     $M5
```

4.1.3. 3/programme de la Somme des Exposants "SOMME".



1.4. S/programme de multiplication en virgule flottante.



```
LP2 BSR    "NORM"      S/Programme de normalisation du resultat
LP3 RTS
LP1 LDA    $M3
    BLT    LP2
    LDA    $M5
    BITA   #$80        Est ce qu'il ya depassement
    BEQ    LP2
    LSRA
    STA    $M5
    INC    $M7
    BRA    LP3
```

S/Programme "SOMME" : B1C0

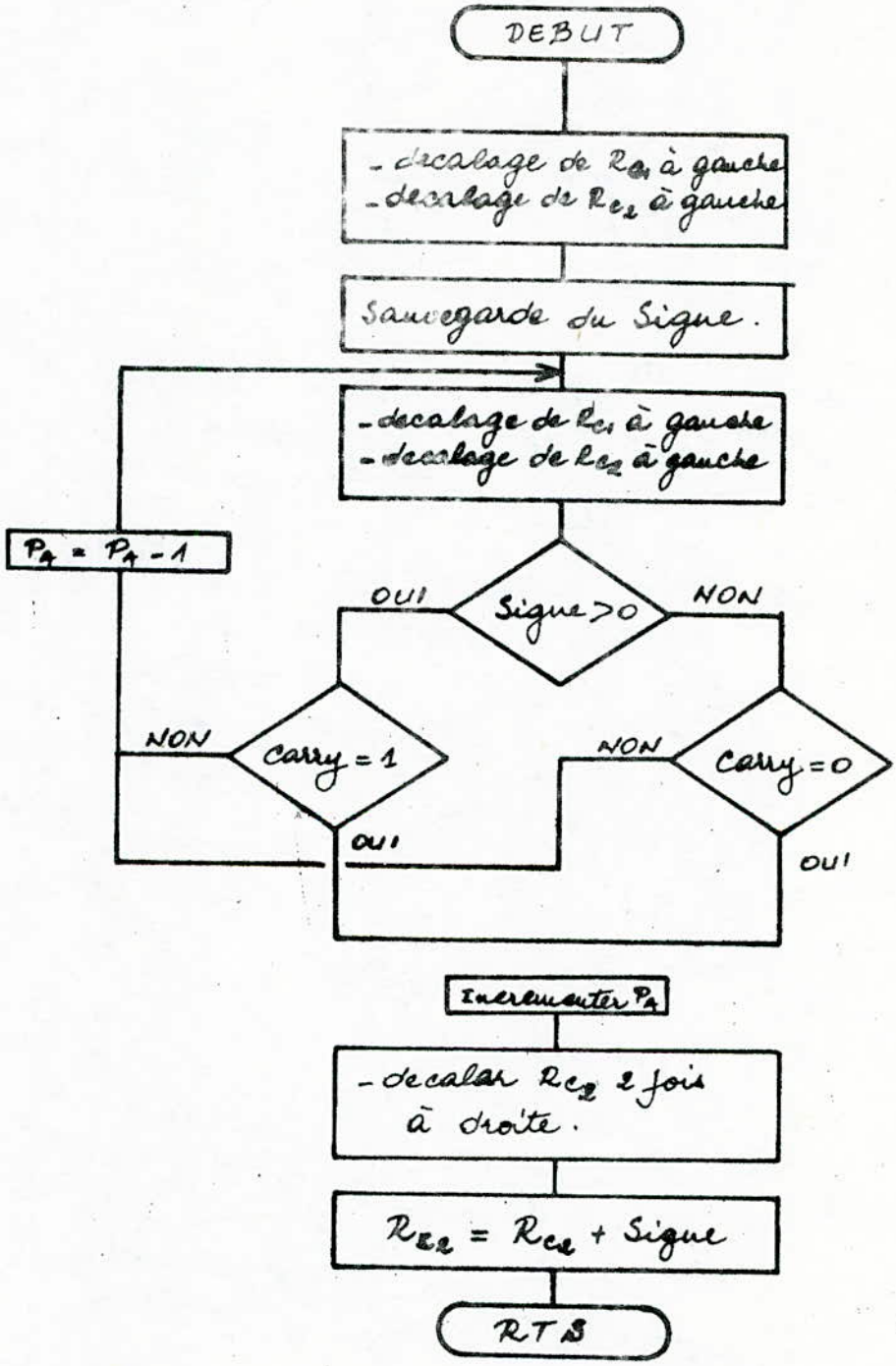
```
    LDA    $M2
    ADDA   $M4           Somme des exposants
    STA    $M7
    LDA    $M2           Est ce que l'exposant du premier
    BGE    LP1           operande est positif
    LDA    $M4
    BGE    LP2           Est ce que l'exposant du deuxieme
    LDA    $M7           operande est positif
    BITA   #$80
    BNE    LP2
    STA    $M7
LP2 RTS
LP1 LDA    $M4
    BLT    LP2
    LDA    $M7
    BITA   #$80
    BEQ    LP2           S'il y a depassement mettre a zero
                         l'exposant du resultat en signe d'erreur
```

S/Programme "MUL" : B200

```
    CLR    $M5
    CLR    $M6
    BSR    "SOMME"       S/Programme de la somme des exposants
    LDD    #$8080
    STD    $M8
    LDA    $M1
    BGE    LP1           Est ce que le premier operande est positif
    NEGA
    BRA    LP7
LP1 BEQ    LP2
    CLR    $M8
```

FF

4.1.6. S/ programme de conversion, virgule fixe – virgule flottante "NORM".



```
LP7   LDB   $M3      est ce que le 2 operande est positif
      BGE   LP3
      NEGB
      BRA   LP6
LP3   BEQ   LP2      est ce que le 2 operande est nulle
      CLR   $M9
LP6   BSR   "MUL"    S/Programme de multiplication
      STD   $M5
      LDA   $M8
      EORA  $M9      signe du resultat
      STA   $MA
      BEQ   LP4      est ce que le signe est positif
      LDD   $M5      le resultat est negatif , on prend
      COMA           sa negation
      NEGB
      STD   $M5
LP4   BSR   "NORM"   S/Programme de normalisation du resultat
      BRA   LP5
LP2   CLR   $M7
LP5   RTS
```

S/Programme "NORM" : B260

```
      CLR   $MB
      LDD   $M5
      BEQ   LP1
      LSL   $M6
      ROLA
      ROR   $MB      sauvegarde du signe
      LDB   $MB
      BNE   LP2
LP3   LSL   $M6      decalage a gauche de l'operande
      ROLA
      DEC   $M7
      BCC   LP3      est ce que la retenue est nulle
      BRA   LP4
LP2   LSR   $M6
      ROLA
      DEC   $M7      decrementer l'exposant
      BCS   LP2
LP4   INC   $M7
      RORA
      LSRA
      ADDA  $MB      ajouter le signe a l'operande
      STD   $M5
Pr    LP1   RTS
```

4.2 PROGRAMMES DES FONCTIONS ELEMENTAIRES :

Les differentes positions memoires reservees pour les differents programmes sont :

```
M20  : B650   abcisse du vecteur ( X )
M22  : B652
M24  : B654   ordonnee du vecteur ( Y )
M26  : B656
M28  : B658   arsument du vecteur ( Q )
M30  : B65A
M32  : B65C
M34  : B65E                                                    -i
M40  : B660  zone reservee pour les coefficients ARCTG 2
M50  : B670   \\      \\      \\          \\
                          -i
              LOG ( 1+&2   ) avec &>0

M70  : B690    \\      \\      \\          \\
                          -i
              LOG ( 1+&2   ) avec &<0
```

X0 , Y0 , Q0 : valeurs initiales du vecteur.

Les zones memoires reservees aux differents programmes sont :

"COS et SIN" : B300

"ARCTG Y/X et $\sqrt{X2 + Y2}$" : B460

"LOGARITHME" : B640

PROGRAMME "COS et SIN" : B300

```
        LDY    #$M40
        LDD    #$X0
        STD    $M20
        STD    $M22
        CLR    $M24
        CLR    $M25
        LDD    #$Q0
        STD    $M28
        CLR    $M30
LP5     LDA    $M28
        ANDA   #$80
        BEQ    LP1
        LDA    $M20
        NEGA
        STA    $M22
        LDD    Y++
        STD    $M1
        LDD    $M28
        STD    $M3
        BSR    "ADD"
        LDA    $M5
        LDB    $M7
        STD    $M28
        LDA    $M24
        NEGA
        STD    $M26
```

```
        BRA     LP2
LP1     LDD     $M20
        STD     $M22
        LDD     $M24
        STD     $M26
        LDD     Y++
        NEGA
        STD     $M1
        LDD     $M28
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M28
LP2     LDA     $M30
        BEQ     LP3
LP4     DEC     $M23
        DEC     $M27
        DECA
        BNE     LP4
LP3     LDD     $M24
        STD     $M1
        LDD     $M22
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M24
        LDD     $M26
        NEGA
        STD     $M1
        LDD     $M20
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M20
        LDA     $M30
        INCA
        CMPA    #$9
        BEQ     LP6
        JMP     LP5
LP6     END
```

PROGRAMME "ARCTG Y/X et $\sqrt{\overline{X2}^{-}+\overline{Y2}^{-}}$: B460

```
        LDY     $M40
        LDD     ##X0
        STD     $M20
        STD     $M22
        LDD     ##Y0
        STD     $M24
        CLR     $M28
        CLR     $M29
        CLR     $M30
LP5     LDA     $M24
        ANDA    #$80
        BEQ     LP1
        LDA     $M20
        NEGA
        STA     $M22
        LDD     Y++
        NEGA
        STD     $M1
        LDD     $M28
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M28
        LDA     $M24
        NEGA
        STD     $M26
        BRA     LP2
LP1     LDD     $M20
        STD     $M22
        LDD     $M24
        STD     $M26
        LDD     Y++
        STD     $M1
        LDD     $M28
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M28
LP2     LDA     $M30
        BEQ     LP3
LP4     DEC     $M23
        DEC     $M27
        DECA
        BNE     LP4
LP3     LDD     $M22
        NEGA
        STD     $M1
        LDD     $M24
        STD     $M3
        BSR     "ADD"
```

```
        LDA     $M5
        LDB     $M7
        STD     $M24
        LDD     $M20
        STD     $M1
        LDD     $M26
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M20
        LDA     $M30
        INCA
        CMPA    #$9
        BNE     LP5
        LDD     $M20
        DEC     $M21
        STD     $M1
        DEC     $M21
        DEC     $M21
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M20
        END
```

PROGRAMME "LOGARITHME" : B640

```
        LDY     $M70
        LDX     $M50
        LDD     #$X0
        STD     $M20
        LDD     #$Y0
        STD     $M24
        CLR     $M28
        CLR     $M29
        CLR     $M30
LP5     LDD     #$X0
        STD     $M32
        NEGA
        STD     $M34
        LDA     $M24
        ANDA    #$80
        BEQ     LP1
        LDA     $M30
        LDU     Y,A
        STU     $M1
        LDD     $M30
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M28
        LDD     $M20
        STD     $M22
        BRA     LP2
LP1     LDD     $M32
        STD     $M34
        LDD     $M20
        NEGA
        STD     $M22
        LDA     $M30
        LDU     X,A
        STU     $M1
        LDD     $M28
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M5
        STD     $M28
LP2     LDA     $M30
        BEQ     LP3
LP4     DEC     $M23
        DEC     $M35
        DECA
LP3     LDD     $M24
        STD     $M1
        LDD     $M22
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M24
        LDD     $M20
        STD     $M1
        LDD     $M34
        STD     $M3
        BSR     "ADD"
        LDA     $M5
        LDB     $M7
        STD     $M20
        LDA     $M30
        CMPA    #$9
        BNE     LP5
        LDD     $M28
        NEGA
        STD     $M28
        END
```

4.3 PROGRAMME DE L'OPERATION PAPILLON : B6C0

les equations de l'operation papillon sont :

$$REJ+1(K0) = REJ(K0) + [REJ(K1).COS@ + IMJ(K1)SIN@]$$

$$REJ+1(K1) = REJ(K0) - [REJ(K1).COS@ + IMJ(K1)SIN@]$$

$$IMJ+1(K0) = IMJ(K0) + [IMJ(K1).COS@ + REJ(K1)SIN@]$$

$$IMJ+1(K1) = IMJ(K0) - [IMJ(K1).COS@ - REJ(K1)SIN@]$$

```
M80 :   K
M81 :   @
M82 :   N/2
M83 :   COS@
M84 :   SIN@
M85 :   Rj+1(K0)
M86 :   Ij+1(K0)
M87 :   Rj+1(K1)
M88 :           registre ou sont stoches les resultas
M89 :           intermedeaire
M90 :
M100:           adresse de debut de zone ou sont stoches
                les valeures initiales Rej(K0),Ij(K0) et
                qui sont disposees de la maniere suivante :

M100 :  mantisse de REj(K0)
M102 :  mantisse de IMj(K0)
M101 :  exposant de REj(K0)
M103 :  exposant de IMj(K0)

                et ainsi de suite.

CLRA
LDB     $M80
ADDD    $MRE
TFR     D,Y
LDB     $M80
ADDB    $M82
CLRA
ADDD    $MRE
TFR     D,U
LDD     0,U
STD     $M1
LDD     $M83
STD     $M03
BSR     "MUL"
LDA     $M5
```

FF

```
LDB     $M7
STD     $M30
LDD     2,U
STD     $M1
LDD     $M84
STD     $M3
BSR     "MUL"
LDA     $M5
STA     $M1
LDA     $M7
STA     $M2
LDD     $M30
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     $M40
STD     $M1
LDD     0,Y
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     $M85
LDD     $M40
NEGA
STD     $M1
LDD     #$Y
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     $M87
LDD     $M84
STD     $M1
LDD     0,U
STD     $M3
BSR     "MUL"
LDA     $M5
LDB     $M7
NEGA
STD     $M89
LDD     $M83
STD     $M1
LDD     2,U
STD     $M3
BSR     "MUL"
LDA     $M5
LDB     $M7
STD     $M1
```

```
LDD     $M89
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     $M88
STD     $M1
LDD     2,Y
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     $M90
LDD     $M88
NEGA
STD     $M1
LDD     2,Y
STD     $M3
BSR     "ADD"
LDA     $M5
LDB     $M7
STD     2,U             IMJ+1(K1)
LDD     $M90
STD     2,Y             IMJ+1(K0)
LDD     $M87
STD     0,U             REJ+1(k1)
LDD     $M85
STD     0,Y             REJ+1(K0)
```

.../

4.4 PROGRAMME DE GENERATION DE DONNEES : B700

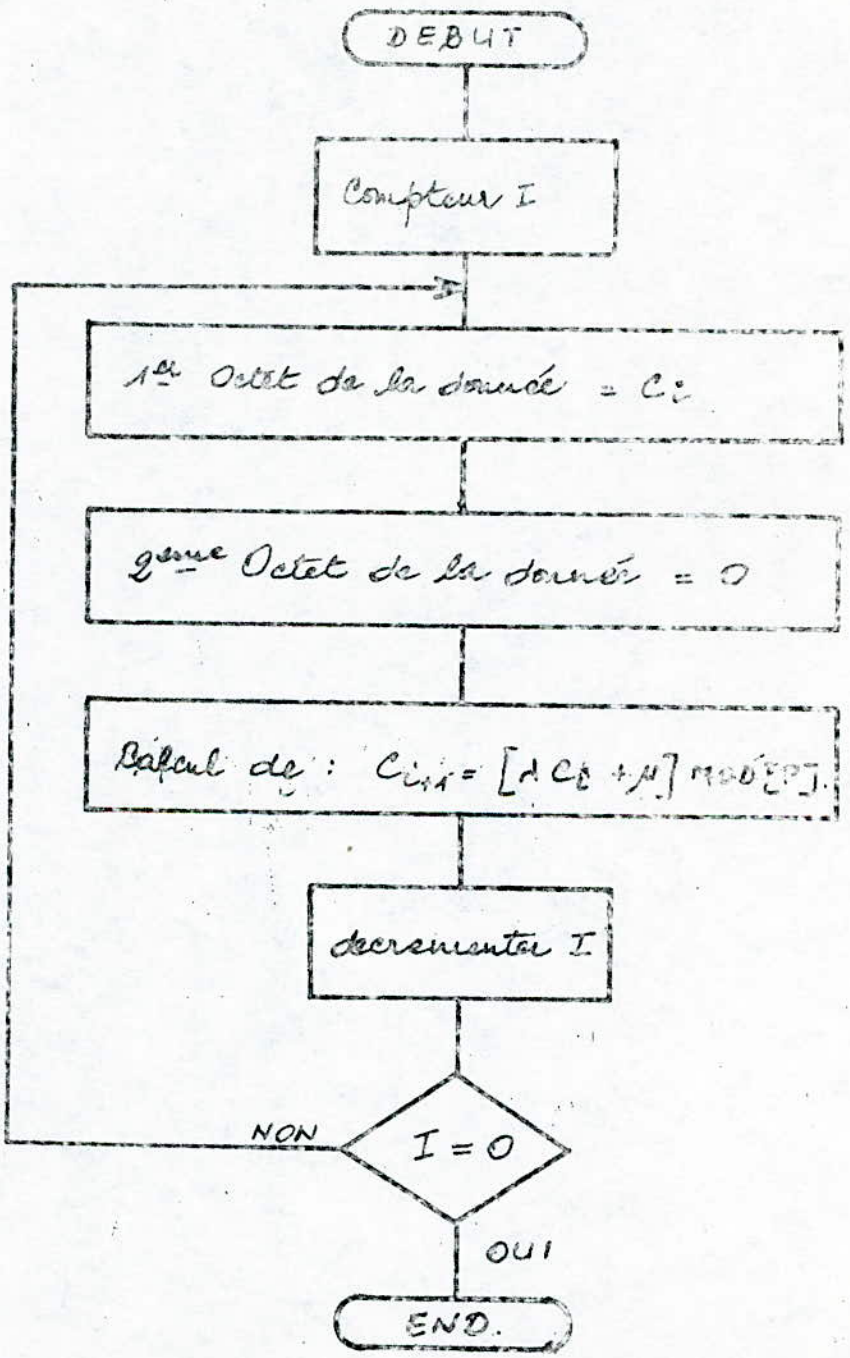
```
        LDA     $$MA    le nombre de valeur a seneree ou compteur
        STA     $MB
        LDX     Z1      Z1:zone ou sont stockes les donnees
LP1     LDA     $M1
        STA     0,X     stocker Ci dans le premier octet
        CLR     1,X     stocker 0 dans le deuxieme octet
        LDB     $M3
        MUL
        ADDD    $M5     calcul de C(i+1)
        STA     $M1
        LEAX    2,X
        DEC     $MB
        BNE     LP1     est ce que le compteur est a 0
        END
```

4.5 PROGRAMME DE TRANSMISSION DE DONNEES A L'UNITE DE TRAITEMENT : B720

```
        PIA1    :  8040
        PIA2    :  8010

        CLRA
        STA     CRA1
        STA     CRB1
        STA     CRA2    acces aux DDRA1 , DDRB1 ,DDRA2
        COMA
        STA     DDRA1   portA en sortie
        STA     DDRB1   portB en sortie
        STA     DDRA2   portA en sortie
        LDA     #$2C
        STA     CRA1
        STA     CRB1
        STA     CRA2
        LDX     #$Z2    Z2:adresse de debut de zone ou sont
                           stockes les donnees.
LP1     LDA     X+
        STA     ORA2    charger ORA2 par le poid fort de la partie
                        reelle ou imasinaire.
        LDA     X+
        STA     ORB1    charger ORB1 par le poid faible de la
                        partie reelle ou imasinaire.
        LDA     X+
        STA     ORA1    charger ORA1 par COS ou SIN .
        CMPX    #$Z3    Z3:adresse de fin de la zone
        BNE     LP1
        END
```

4.4. PROGRAMME DE GENERATION DE DONNEES PSEUDO-ALEATOIRES.



4.6 PROGRAMME DU PRODUIT SCALAIRE : B740

Z3:debut d'adresse des coordonnees du vecteur (X)
z4:debut d'adresse des coordonnees du vecteur (Y)

```
      LDX    #$Z3
      LDY    #$Z4
      LDU    #$512      charger U par le nombre de coordon-
      STU    $MA        nees du vecteur (compteur)
      CLR    $MD
      CLR    $ME
LP1   LDD    X++        charger D par les coordonnees de X
      STD    $M1
      LDD    Y++        charger D par les coordonnees de y
      STD    $M3
      BSR    "MUL"
      LDA    $M5
      LDB    $M7
      STD    $M1
      LDD    $MD        charger d par le resulta des operations
                        precedentes
      BSR    "ADD"
      LDA    $M5
      LDB    $M7
      STD    $MD
      LDU    $MA
      LEAU   -1,X       decrementer le compteur
      STU    $MA
      CMPU   #$0000
      BNE    LP1        est ce que le compteure est nul
      END
```

4.7 PROGRAMME DU MONITEUR

les differentes positions memoires reservees pour les differents programmes sont:

M10: code 7 segment du premier afficheur adresse FFE5
M11: \\ \\ \\ \\ deuxieme \\ \\ FFE6
M12: \\ \\ \\ \\ troisieme \\ \\ FFE7
M13: \\ \\ \\ \\ quatrieme \\ \\ FFE8
M14: \\ \\ \\ \\premier \\ donnee FFE9
M15: \\ \\ \\ \\deuxieme \\ \\ FFEA
M1 : compteur du nombre d'interruption.
M2 : code d'une touche du clavier
M3 : chiffre en hexadecimal equivalent au code de la touche FFE3.

Les zones memoires reservees aux tables sont:

FF00 : Table 1
FF20 : Table 2
FF40 : Table 3
FF60 : Table 4
FF80 : Table 5

Les zones memoires reservees aux differents sous-programmes sont:

FC00 : "RESET"
FC50 : "GEST"
FCC0 : "DEC"
FD50 : "ETC"
FD80 : "SHIFT"
FDA0 : "4DGT"
FDD0 : "2DGT"
FE00 : "ERRO"
FE80 : "POPC"
FE40 : "TIRER"
FE60 : "DELAI 20"
FE70 : "DELAI 1"

L'adrsse du PIA est: 8020.

4.7.1.PROGRAMME "RESET" :

```
LDA     #$10
TFR     A,CCR
JSR     "RESET"
CLRA
STA     CRA
STA     CRB
LDA     #$7F        7 lignes du port A en sortie et 1 en
                    entree.
STA     DDRA
```

```
        LDA     #$FF
        STA     DDRB
        LDA     #$4         port A ne tient pas compte des
        STA     CRA         interruptions
        LDA     #$7         interruptions validees au front
        STA     CRB         montant.
        LDA     #$3F
        STA     $M10
        STA     $M11
        STA     $M12
        STA     $M13
        STA     $M14
        STA     $M15
        CLR     $M1
        CLR     $M15
        CLR     $M10
        CLR     $M11
        CLR     $M12
        CLR     $M13
LP0     JSR     "TIRET"
        ANDCC   #$00
        BRA     LP0
```

4.7.2 PROGRAMME "GEST" : FC50

```
SR4     JSR     "DEC"
        LDB     $M1
        DECB
        BNE     LP0
LP3     LDX     #$X5        charger X par l'adresse de la table 5
        LDA     $M3
        LDA     X,A         charger A par le code 7 segment du no-
                            mbre correspondant.
        STA     $M10        stocher A dans le 1 afficheur adresse.
        BRA     LP1
LP0     DECB
        BNE     LP2
LP5     LDA     $M10        decalage a gauche du nombre du 1
        STA     $M11        afficheur adresse au 2.
        BRA     LP3
LP2     DECB
        BNE     LP4
LP7     LDA     $M11        decalage a gauche du 2 afficheur.
        STA     $M12
        BRA     LP5
LP4     DECB
        BNE     LP6
        LDA     $M12
        STA     $M13
        BRA     LP7
PR      LP6     DECB
```

```
        BNE     LP8
LP9     LDX     #$X5        charsement de la table 5.
        LDA     $M3
        LDA     X,A         charsement de A par le code 7 sesment
                            du nombre correspondant.
        STA     $M14        stochase de A dans le 1 afficheur de
                            donnee.
        BRA     LP1
LP8     LDA     $M14        decalase a sauche du nombre du 1 affich
        STA     $M15        eur de donnee au 2.
        BRA     LP9
LP1     LDA     $M1
        CMPA    #$5         est ce que le nombre dinterruption a
                            depasse 4
        BGE     SP1
SP3     JSR     "DGT4"      S/programme d'affichage des adresses
        BRA     SP2
SP1     JSR     "DGT2"      S/programme d'affichage des donnees
        BRA     SP3
SP2     LDA     CRB
        ANDA    #$80
        BEQ     LP1
        BRA     SR4
```

4.7.3 S/PROGRAMME "DEC" FCC0

```
        LDA     #$00
        STA     ORB         eteindre les afficheurs
        LDA     #$3F
        STA     ORB         mettre les sorties du port B a 1
LP1     LDB     ORA
        ANDB    #$80
        BEQ     LP0         recherche de la colonne,est ce que PA7
                            est nul.
        LDA     ORB
        ADDA    #$40
        BCC     LP1         s'il ya depassement du carry il ya erro
LP3     JSR     "ERREUR"
LP0     ANDA    #$C0
        STA     $M1         le code de la colonne est stoche ds M1
        LDB     #$20
        ORB     $M1
        STB     ORB
LP4     LDA     ORA
        ANDA    #$80
        BEQ     LP2         recherche de la ligne est ce que PA7
                            est nul.
        ANDB    #$3F
        LSRB
        BEQ     LP3
        ORB     $M1
        BRA     LP4
LP2     STB     $M2
```

```
LP5  LDA    ORA
     ANDA   #$80
     BEQ    LP5      est ce que la touche a ete deleste
     JSR    "DELAI20"
     LDA    $M2
     ANDA   #$0F
     BNE    LP8      ets ce que c'est une touche de commande
     JSR    "ETC"
LP8  INC    $M1      incrementer le compteur.
     CLR    $M3
     LDX    #$X1     charger X avec la table 1
     LDA    $M2
LP7  CMPA   0,X      recherche de l'equivalent du code de la
                     touche en hexa.
     BEQ    LP6
     LEAX   1,X
     INC    $M3
     BRA    LP7
LP6  RTS
```

4.7.4 S/PROGRAMME "ETC" FD50

```
     LDB    #$7
     STB    $M20
     LDX    #$X2     charsement dans X de l'adresse de la
                     table 2
     LDA    $M2
LP2  CMPA   0,X      recherche du code de la touche dans
                     la table 2.
     BEQ    LP1
     LEAX   2,X
     DEC    $M20
     BNE    LP2
     JSR    "ERREUR"
LP1  LDX    20,X     recherche de l'adresse d'execution de
                     la commande .
     JSR    0,X      saut a l'adresse de la commande.
```

4.7.5 S/PROGRAMME "SHIFT" FD80

```
     CWAI   #$EF     attendre la prochaine interruption
     JSR    "DEC"
     LDA    $M3
     LDX    #$X4     charger X par la table 4.
     LDX    A,X
     LDX    A,X
     JSR    0,X      saut a l'adresse du programme a execute.
```

4.7.6 S/PROGRAMME "4DGT" FDA0

```
        LDA     #$03
        STA     $M2
        LDA     #$01
        STA     ORB         affichage du 1 afficheur adresse
        LDY     #$M10
LP1     LDB     0,Y
        STB     ORA
        JSR     "DELAI 1"
        LSLA
        STA     ORB
        LEAY    1,Y         affichage du prochain afficheur
                            d'adresse.
        DEC     $M2
        BNE     LP1
        RTS
```

4.7.7 S/PROGRAMME "2DGT" FDD0

```
        LDA     #$10
        STA     ORB
        LDB     $M15        affichage du 1 afficheur de donnee.
        STB     ORA
        JSR     "DELAI 1"
        LSLA
        STA     ORB
        LDB     $M14        affichage du 2 afficheur de donnee.
        STB     ORA
        JSR     "DELAI 1"
        RTS
```

4.7.8 S/PROGRAMME "ERRO" FE00

```
        LDA     #$8
        STA     ORB
        LDB     #$79        affichage de E sur le 4 afficheur.
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
        LDB     #$50        affichage  de r sur le 3 afficheur
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
```

```
        LDB     #$50      affichage de r sur le 2 afficheur.
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
        LDB     #$5C      affichage de 0sur le 1 afficheur.
        STB     ORA
        JSR     "DELAI 1"
        BRA
```

4.7.9 S/PROGRAMME "POPC" FE80

```
    LP1 LDA     #$8
        STA     ORB
        LDB     #$73      affichage de P sur le 4 afficheur.
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
        LDB     #$3F      affichage de 0 sur le 3 afficheur
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
        LDB     #$73      affichage de P sur le 2 afficheur
        STB     ORA
        JSR     "DELAI 1"
        LSRA
        STA     ORB
        LDB     #$39      affichage de C sur le 1 afficheur
        STB     ORA
        JSR     "DELAI 1"
        BRA     LP1
```

4.7.A S/PROGRAMME "TIRET" FE40

```
LP1  LDA   #$3F
     STA   ORB      valider les 6 afficheurs
     LDA   #$40
     STA   ORA      afficher le tiret
     JSR   "DELAI 1"
     BRA   LP1
```

4.7.B S/PROGRAMME "DELAI 20" FE60

```
LP1  LDX   #$720    charger X par le nombre de cycle
     LEAX  -1,X
     CMPX  #$0000
     BNE   LP1
     RTS
```

4.7.B S/PROGRAMME "DELAI 1" FE70

```
     LDA   #$7D
LP1  DECA
     CMPX  #$0000
     BNE   LP1
     RTS
```

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

THESE :

- Etude du microprocesseur juin 85
- Programmateur de proms et reproms juin 84
- Etude et realisation d'une centrale piloteè per microprocesseur . seba juin 84

REVUES :

- Haut parleur
- micro systeme

OUVRAGES:

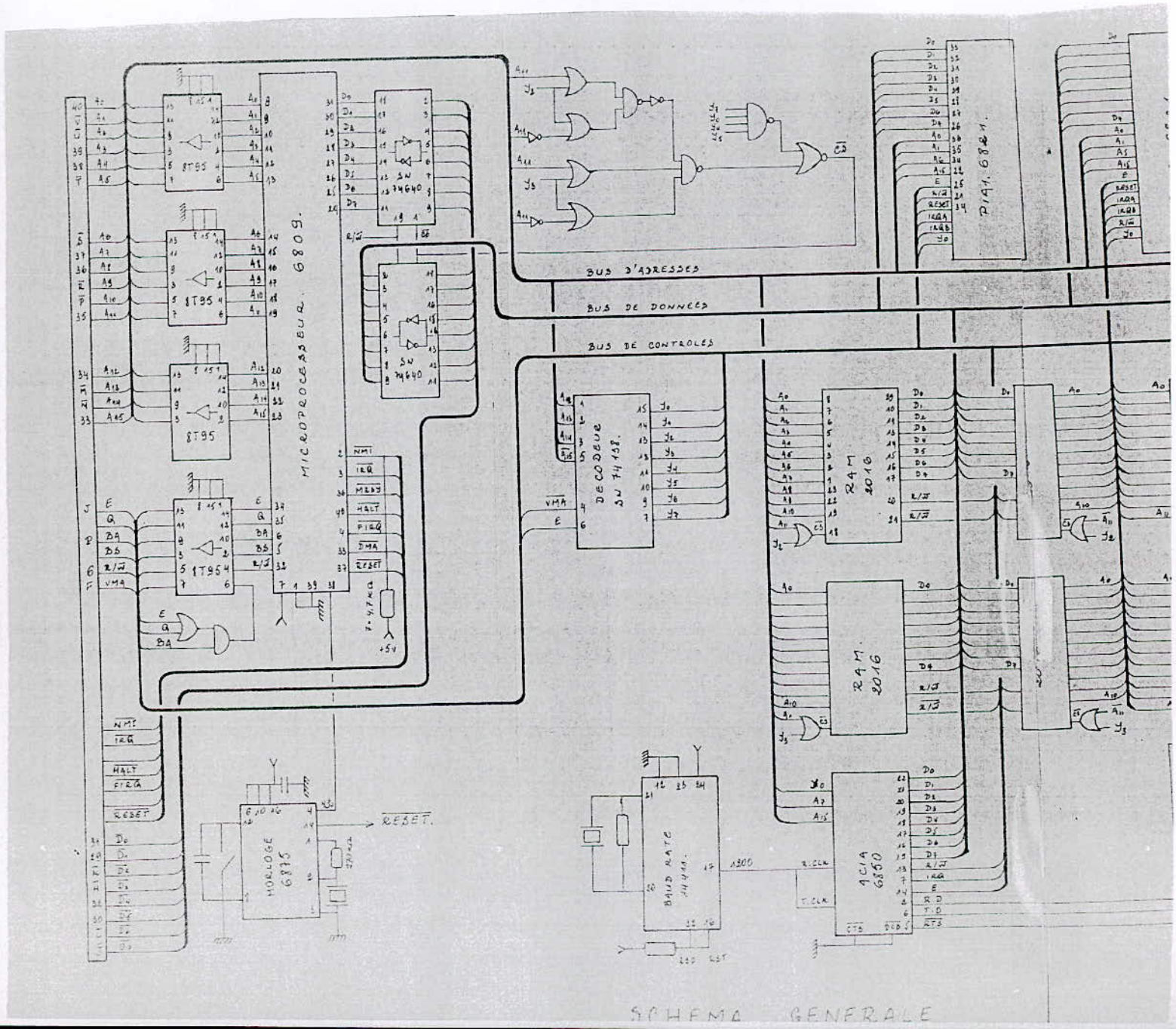
- Les systemes à microprocesseur (AUMIAUX)
- Le microprocesseur 6809 (DARDANE)
- The fast fourier transform (BRIGHAM)

* LA BEAUTE SANS LA BONTE EST UNE SOURCE SANS EAU .

* LE MEILLEUR MOMENT C'EST TOUT DE SUITE .

* ON PEUT EPROUVER UNE TELLE JOIE A FAIRE PLAISIR A QUELQU'UN QU'ON AIT ENVIE DE LE REMERCIER .

* JE N'AIME PAS QU'UN HOMME SOIT TROP COMPETENT IL RISQUE DE MANQUER D'HUMANITE .



SCHEMA GENERALE