MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

U.S.T.H.B

****

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE

****

PROJET DE FIN D'ETUDES

Thèse d'ingénieur en électronique

SUJET: ETUDE D UN MICRO-ORDINATEUR BASÉ AUTOUR DU 6802 : LE KIT D5 de MOTOROLA

C.S.T.N. LABORATOIRE DE TELEDETECTION

Proposé par :

M[r] A ABDELLAOUI

Docteur de spécialité

Etudié par :

ARABI Fatiha

KASSAB Yacine

JANVIER 1982

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

U.S.T.H.B

✲✲✲✲✲✲✲✲✲✲✲

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE

✲✲✲✲✲✲✲✲✲✲✲

PROJET DE FIN D'ETUDES

Thèse d'ingénieur en électronique

SUJET: ETUDE D'UN MICRO-ORDINATEUR BASÉ AUTOUR DU 6802 : LE KIT D5 de MOTOROLA

C.S.T.N. LABORATOIRE DE TELEDETECTION

Proposé par :

M^r A ABDELLAOUI

Docteur de spécialité

Etudié par :

ARABI Fatiha

KASSAB Yacine

JANVIER 1982

D E D I C A C E S

A mon père

A ma mère

A mes soeurs et mon frère

A mes beaux- frères

A mes oncles et mes cousins

A toute la famille

ET à tous mes Amis .

Yacine .

D E D I C A C E S

A mon père

A ma mère

A la mémoire de mon grand-père

A tous mes frères et soeurs

A toute ma famille

A toutes mes amies (-is) .

Pour eux , je dédie cet humble travail .

Fatiha

REMERCIEMENTS

—.—.—.—.—.—

- Nous formulons l'expression de notre profonde reconnaissance à Monsieur ABDELLAOUI , Directeur du groupe de recherche en Télédétection d'ALGER de nous avoir accueilli dans son laboratoire et d'avoir dirigé notre travail .
- Nous remerçions par la même occasion M^elles KAOUA Malika et ZIZI Malika assistantes à l'E-N-P-A pour leur aide précieuse les conseils qu'elles nous ont prodigués et les encouragements incessants qu'elles nous ont apportés .
- Nous ne manquerons pas d'exprimer aussi toute notre gratitude et notre reconnaissance à tous les professeurs de l'Ecole Nationale Polytechnique qui ont contribué à notre formation.

TABLE DES MATIERES .

INTRODUCTION

Le microprocesseur est peut être le développement le plus important que l'industrie électronique ait connu depuis au moins la dernière décennie.Il se prête au remplacement de la logique câblée pour laquelle l'accent est totalement porté sur le hardware : elle exige, en effet,un grand nombre de composants rendants les montages encombrants; elle n'offre guère de souplesse . A l'inverse, les microprocesseurs sont d'une extrême souplesse d'utilisation car ici, l'accent est mis sur le logiciel (software) , modifiable à tout moment à volonté et en un temps réduit.

La nouvelle génération des microprocesseurs s'occupe de réunir les fonctions nécessaires à la réalisation d'un micro-ordinateur complet sur un minimum de puces; ceci permet d'amortir le coût et de réduire au maximum le nombre de circuits total nécessaires à la réalisation d'un système donné. Nous parlons alors de la génération des micro-ordinateurs intégrés tel le " 6802 + 6846 " dont l'étude est faite au chapitre 1 .

Par ailleurs , le "6802" peut s'intégrer facilement dans un système plus complexe utilisant des circuits de la famille 6800 (ROM,RAM,PIA, ACIA ... etc) pour former une carte micro-ordinateur de base.

C'est le cas du KIT d'initiation MEK.6802.D5.MOTOROLA successeur du KIT MEK 6800 D2, récemment apparu sur le marché.

Son étude tant du point de vue HARD que SOFT est détaillée au chapitre 2 . Grâce aux listings du moniteur donnés sur le manuel d'utilisation,nous sommes parvenus à établir son fonctionnement et à l'utiliser pour mettre au point des exemples de programmation.Nous présentons quelques uns de ces exemples au chapitre 3.Nous terminons l'étude du KIT par une application portant sur une acquisition de données.

CHAPITRE 1 : Famille du microprocesseur 6802

I Présentation de la famille 6802:

Les microprocesseurs les plus connus jusqu'en 1977 avaient une architecture dite du type standard à usage universel tel que le 6800 .

Dans cette même année, un nouveau type de MPU et sa famille (6802 + 6846) sont apparus sur le marché avec une architecture plus intégrée du type à " entrées/sorties réparties "(les E/S (entrée/sortie) sont réparties en 2 ou 3 boîtiers LSI).

En effet, les 2 boîtiers 6802 et 6846 intègrent le MPU 6800 la RAM , l'horloge, la ROM , un port d'E/S de 8 bits et un temporisateur programmable (fig 1 et 2)

Le microprocesseur 6802 est un circuit intégré monolithique (sur une seule pastille de silicium). Il est basé sur une des technologies LSI (large scale integration = intégration à grande échelle) qui est la technologie à appauvrissement, canal N et grille silicium la " N MOS " .

Les circuits intégrés NMOS sont fabriqués à base de transistors à effet de champ : les " NMOS FET " (fig 3)

Grâce aux porteurs unipolaires qui sont les électrons , le transistor NMOS possède une rapidité d'éxécution plus performante que celle du type PMOS (porteurs trous). La mobilité des "e^- " étant de 2 à 3 fois plus grande que celle des trous ($\mu e > \mu p$).

Il est dit à appauvrissement (ou charge à déplétion)à cause de la particularité de son canal (existant déjà au repos) qui disparaît lorsque la grille est polarisée .

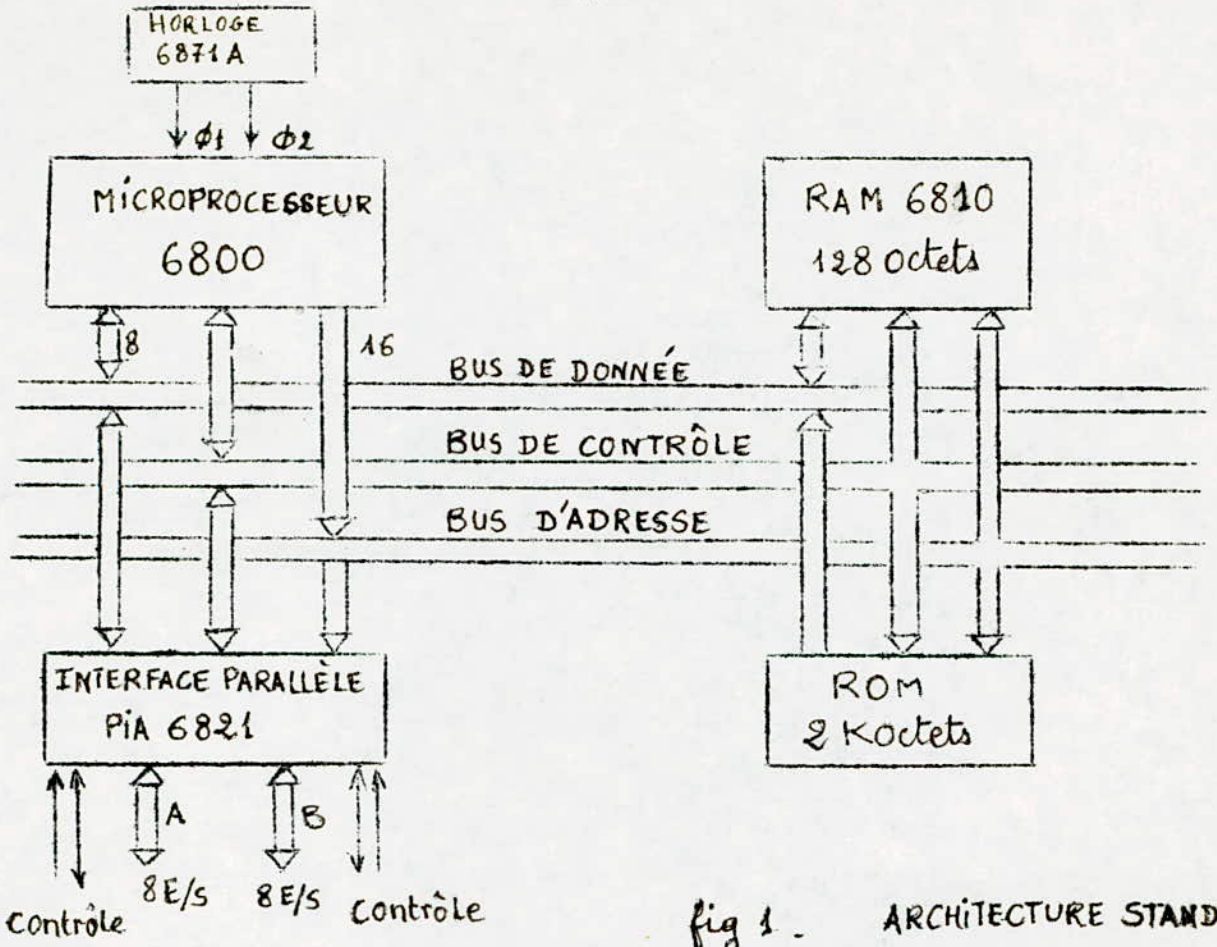


fig 1 - ARCHITECTURE STANDARD D'UN SYSTÈME AUTOUR DU M.P.U. 6800

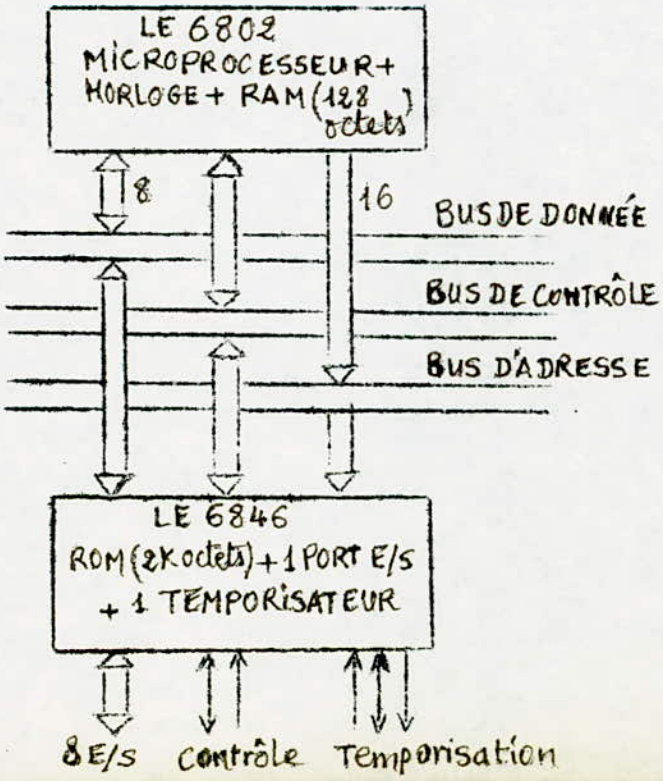


fig 2 - ARCHITECTURE À "E/S REPARTIES" 6802 + 6846

Le 6802 travaille sur des mots de 8 bits. Grâce à ses 16 lignes d'adresse , il a la possibilité d'être extensible dans un système jusqu'à 2^{16} = 65 536 = 64 K adresses mémoire ou périphériques (1K = 2^{10})

En effet , les périphériques avec le 6802 sont adressés comme des positions mémoire ce que l'on désigne en anglo-saxon par " Memory Mapped I/O" .

Le MPU 6802 contient les mêmes registres et accumulateurs que le 6800 avec en plus :

- Une RAM interne de 128 octets situés entre les adresses hexadécimales 0000 et 007F .
- Un oscillateur d'horloge interne .

(voir fig 4)

Les bus d'adresses et de données possèdent la même structure que celle du 6800 ; par contre, les bus de contrôle diffèrent quelque peu .

Les entrées "TSC" (Three State Control) et "DBE" (Data Bus Enable) ont été supprimées sur le 6802 pour être remplacées par des entrées de contrôle de l'horloge et de la RAM .

Rappelons que pour le 6800 , l'entrée "TSC" nous permet de faire un accès direct mémoire sans arrêter le microprocesseur mais en le ralentissant simplement .

La broche "DBE" nous indique la validation du bus de données.

Les microprocesseurs 6800 et 6802 sont compatibles au niveau du logiciel . Ils possèdent le même jeu d'instructions; de ce fait, tous les programmes développés avec le 6800 sont valables pour le 6802 .

Le microordinateur de la figure 5 est constitué essentiellement de 2 boîtiers : le MPU 6802 et le CI 6846 .

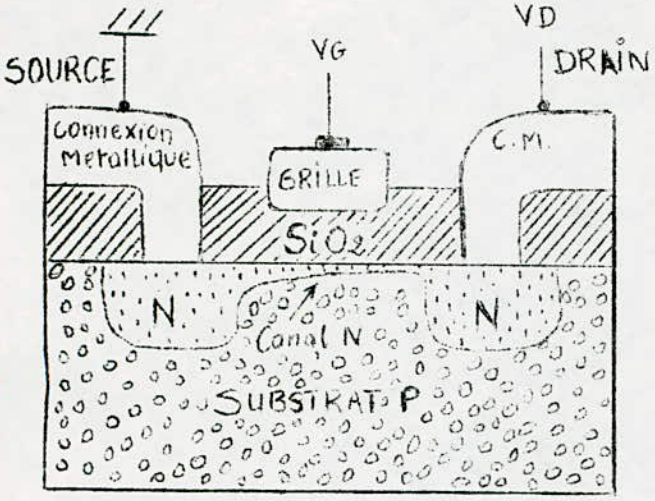


fig a

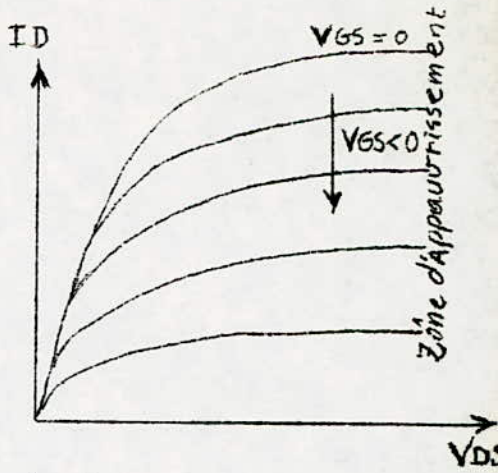


fig b

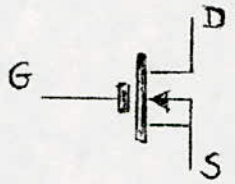


fig c

fig 3 : a - Coupe du transistor MOS FET à canal N
b - Caractéristique $I_D = f(V_{DS})$ du FET
c - Symbole du transistor NMOS à appauvrissement

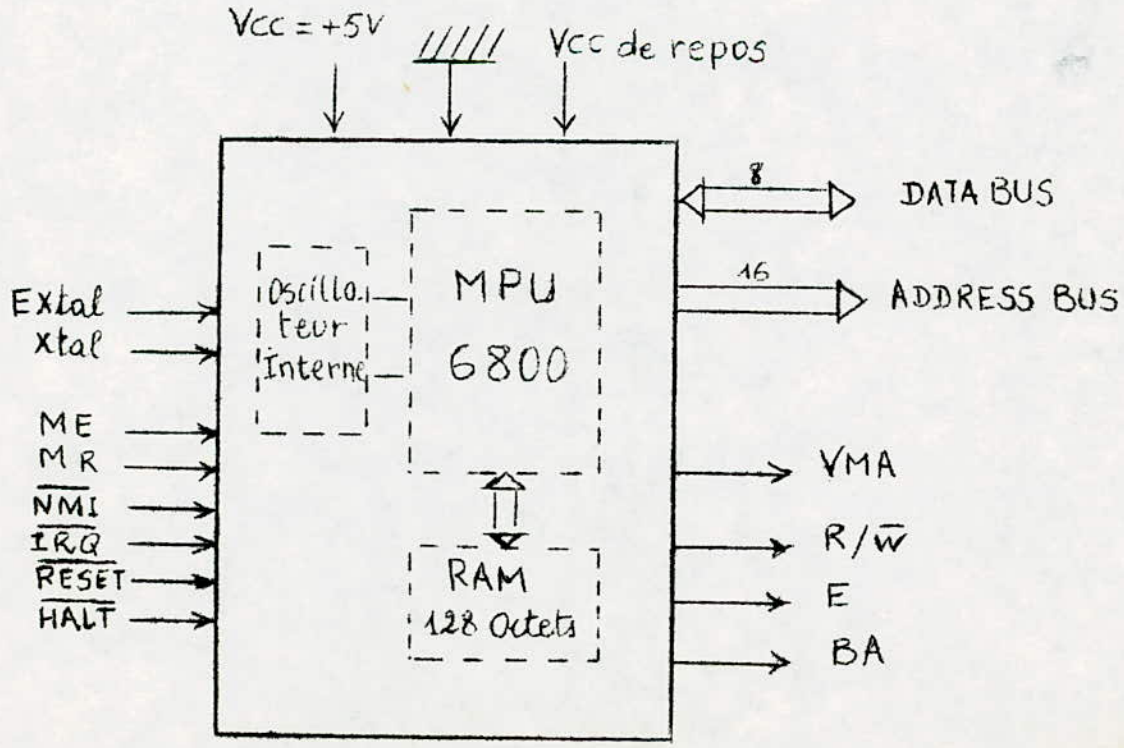


fig 4 - Synoptique global du MPU 6802

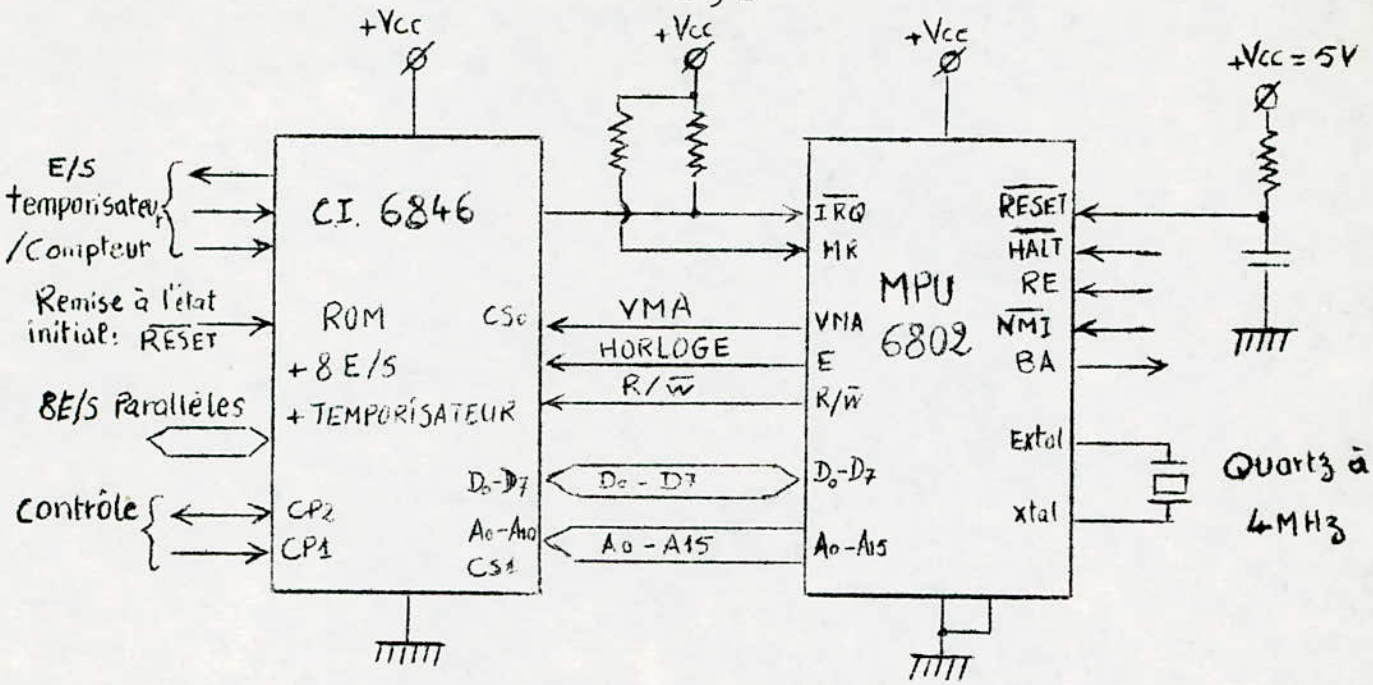
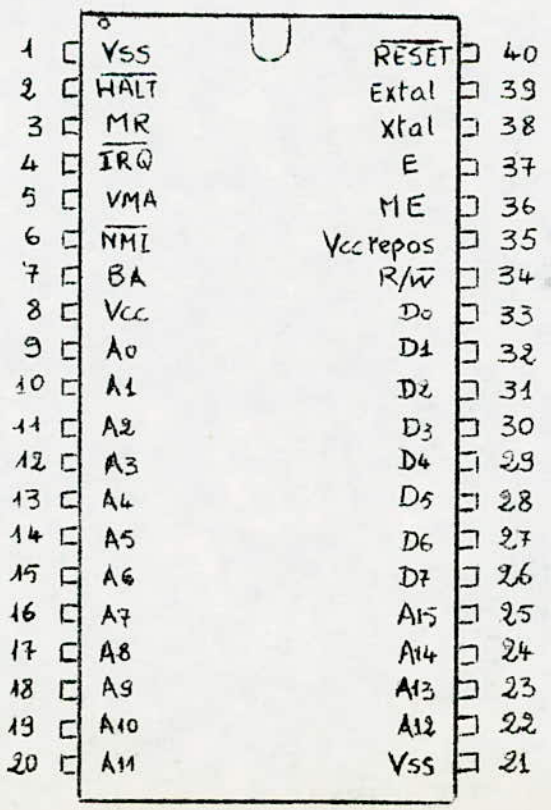


fig 5 - Le microordinateur en 2 boîtiers : 6802 + 6846

fig 6 - Brochage du MPU 6802



Il ne nécessite qu'une seule alimentation Vcc = + 5v , ce qui représente un trés grand avantage pour l'utilisateur.

Ses entrées/sorties sont compatibles à la logique TTL standard

Le MPU 6802 n'est pas uniquement limité à ce type de configuration , mais il peut aussi s'intégrer et s'adapter dans des systèmes plus complexes utilisant des circuits de la famille 6800(RAM, ROM, PIA , ACIA ...etc)

Le circuit 6846 : c'est un circuit intégré composite comportant dans son boîtier :
- une ROM de 2K octets .
- un port de 8 E/S identique au port B du PIA 6820 (ou 6821).
- un temporisateur programmable .

II Etude de l'unité centrale

II.1 Architecture et brochage du MPU 6802 : fig 6 et 7

Tous les registres internes du 6800 figurent sur le 6802.

D'aprés la figure 7 , on distingue :

a) l'unité arithmétique et logique (UAL) :

l'UAL est un ensemble de circuits combinatoires capables d'éffectuer les opérations arithmetiques et logiques nécessaires au traitement de l'information .

b) les registres internes : (voir fig 9)

Le 6802 possède 5 types de registres internes :
- les accumulateurs A et B (registres 8 bits)
- le registre d'index "IX" (registre 16 bits)
- le compteur de programme ou compteur ordinal " PC " (registre 16 bits).
- le pointeur de pile "SP" (registre 16 bits) :

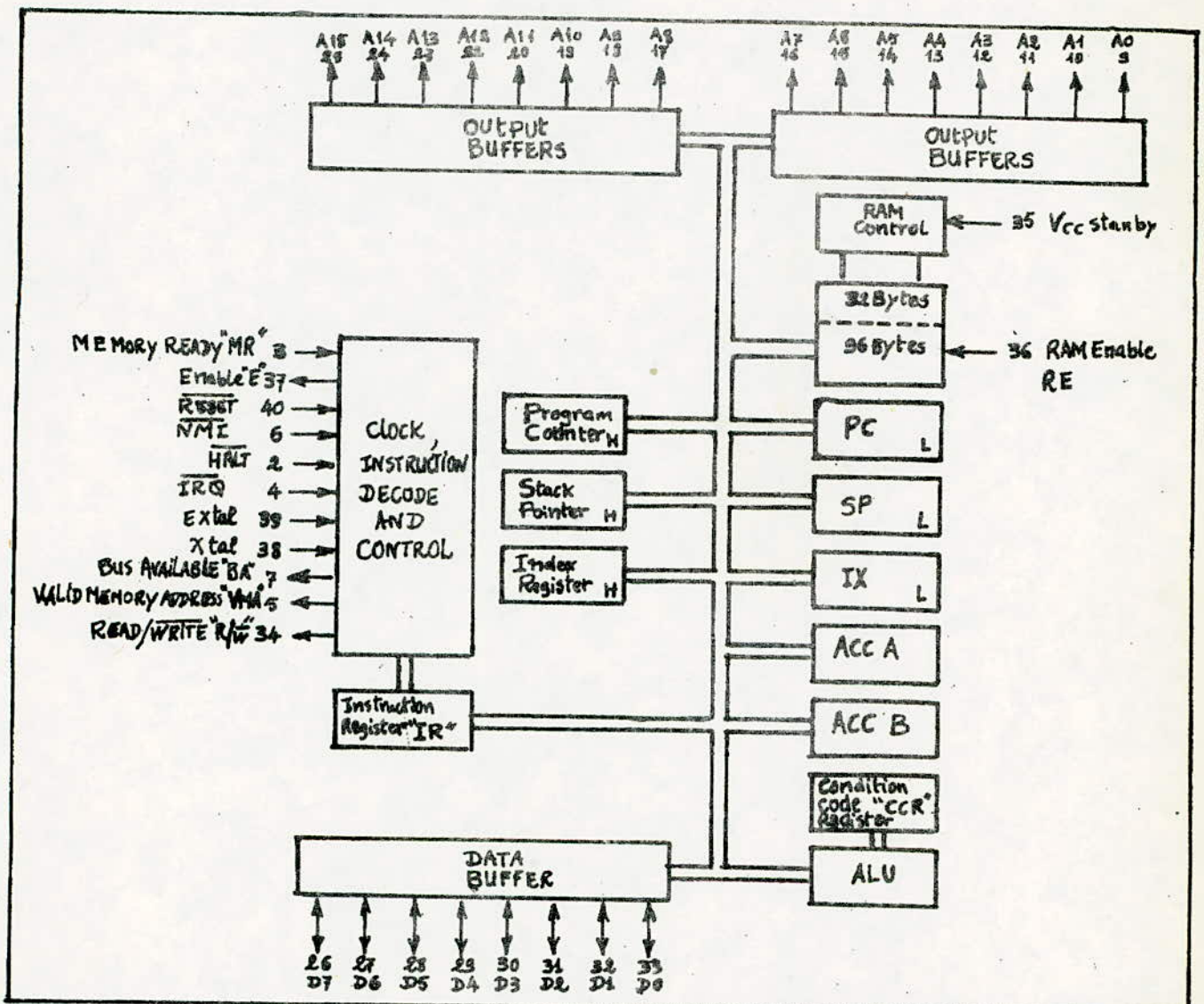
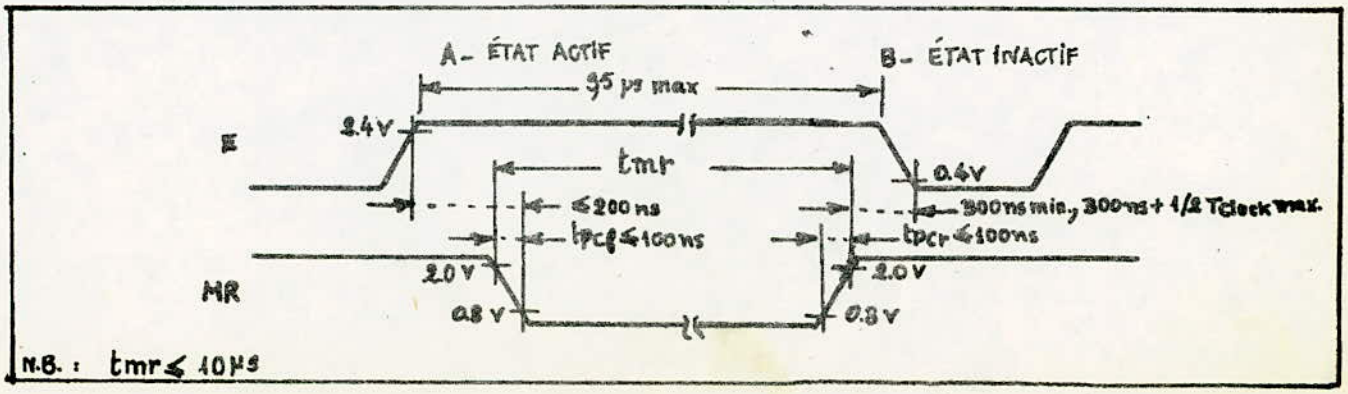


fig 7 - Structure interne du M.P.U. 6802

fig 8 - FONCTION DE COMMANDE MÉMOIRE PRÊTE



- le registre des codes de condition " CCR " (registre 8 bits) :

La plupart des opérations exécutées par le MPU affectent le contenu de son registre de conditions . Celui-ci comprend 6 indicateurs qui se positionnent lorsqu'une condition particulière à chacun apparait : l'indicateur "C" de retenu , l'indicateur"V" de dépassement de la capacité de 8 bits complémentés à deux,l'indicateur "Z" du zéro,l'indicateur"N" d'un resultat négatif ,l'indicateur "H" de demi-retenue .

Le sixième bit de ce registre est le bit d'interruption "I". I = 1 ; le MPU n'est pas autorisé à servir les interruptions masquables ($\overline{IRQ}$ pour le 6802).

c) l'unité de commande :

L'unité de commande assure, à partir du registre d'instruction le séquencement de toutes les opérations logiques et la gestion du système au rythme de l'horloge [illegible] .

d) la RAM interne : Similaire à la mémoire RAM statique 6810 de la famille 6800 . La RAM intégrée dans le MPU 6802 est d'une capacité de 128 octets. L'adressage de ces 128 octets se fait à partir de $ 0000 jusqu'à $ 007F.

Lors [illegible] d'une coupure d'alimentation, les 32 premiers octets peuvent fonctionner en mode faible consommation grâce au Vcc de repos(ou Vcc standby).Ceci offre la possibilité d'une sauvegarde de certaines données dans cette partie de la RAM.

La RAM interne possède une logique de contrôle accompagnée d'une entrée de validation (broche RE).

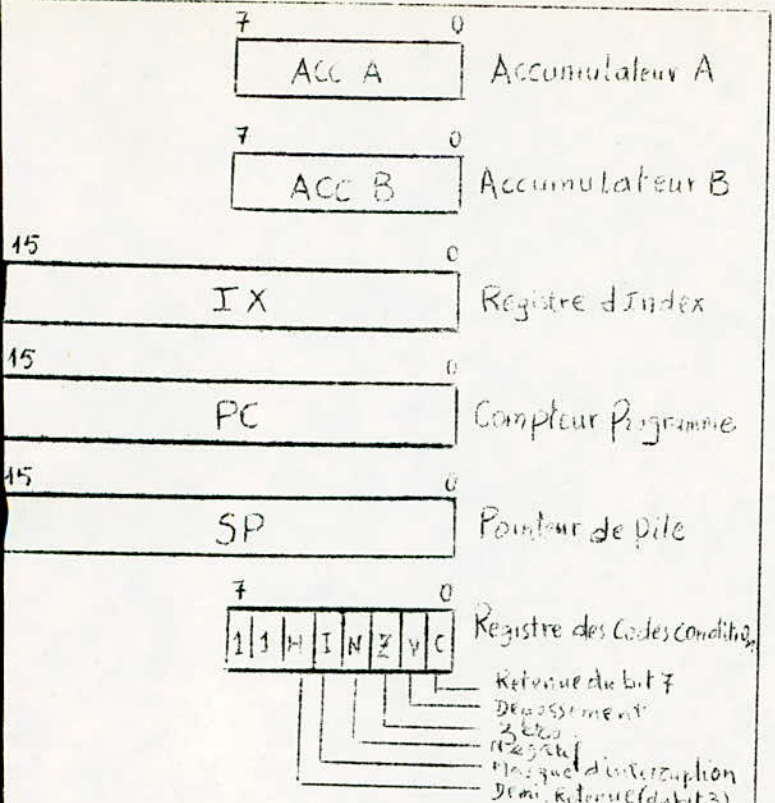


fig 9 - Registres Programmables du Microprocesseur

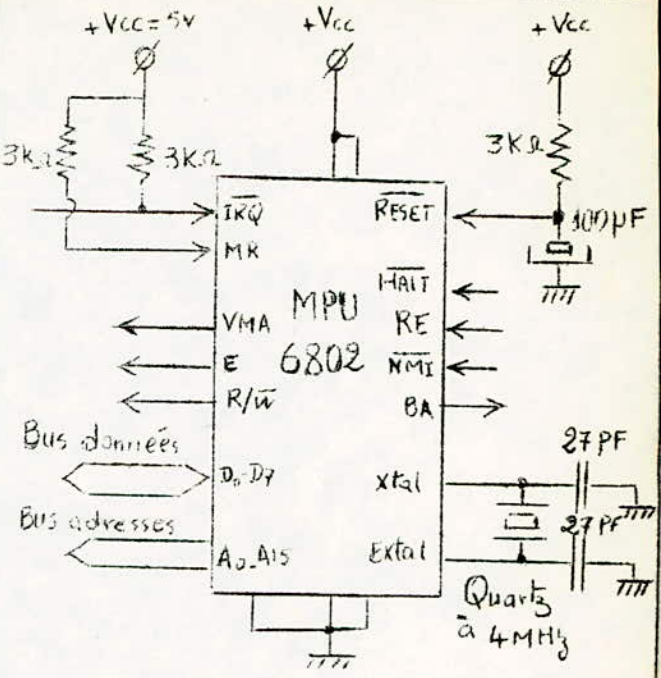
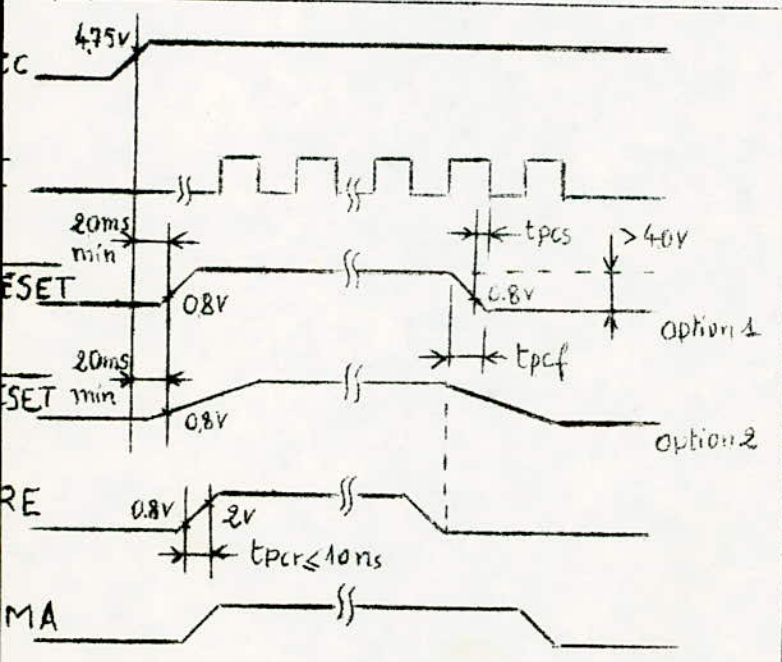


fig 10 - Fonctionnement du micro-processeur à la fréquence fondamentale de 1MHz



option 1 : Lorsque cette option est choisie, les broches RESET et RE peuvent être reliées

option 2 : Correspondant à la fig 12 pour les conditions de mise hors tension

fig 11 - Mise sous tension et Initialisation

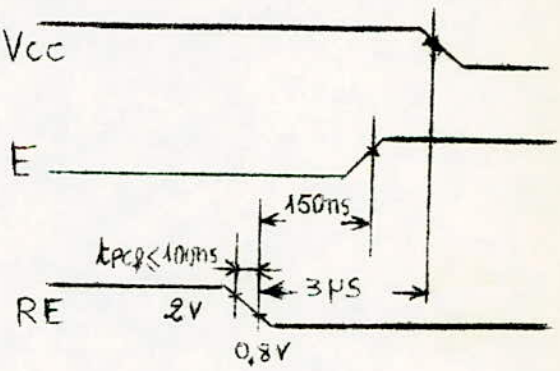


fig 12 - Séquence de mise hors tension

tpcs : temps d'établissement des lignes de commandes du MPU

tpcr : temps de montée

tpcf : temps de descente

e) l'horloge :

Le 6802 possède un oscillateur interne piloté par un quartz externe . Les entrées Extal et Xtal sont prévues pour fonctionner avec un quartz de frequence fondamentale de résonnance de 1 MHZ . Cependant , le diviseur par quatre intégré dans le 6802 permet l'utilisation d'un quartz de 4 MHZ .

Il est possible d'utiliser une horloge externe. Dans ce cas, la broche Xtal sera mise en l'air et la broche Extal sera reliée à la sortie TTL de l'horloge .

II- 2 Les signaux d'entrée/sortie du MPU

Les broches d'entrée et de sortie du MPU 6802 peuvent se regrouper en cinq parties (fig. 4 et 6):

a - Le bus de données

b - Le bus d'adresses

c - Le bus de commande

d - Les signaux de commande du microprocesseur

e - L'alimentation .

a) Le bus de données ou Data Bus (Do - D_7) :

Le bus de données est de 8 bits, bidirectionnel. Lorsque la RAM interne est sélectionnée, le data bus est en " position sortie" ce qui interdit à toute information externe de reutrer dans le MPU.

b) Le bus d'adresses ou Address Bus (Ao - A15) :

C'est un bus unidirectionnel de 16 lignes.

c) Le bus de commande :

Il est destiné à commander des opérations d'entrée/sortie telles qu'une lecture/écriture mémoire ou une lecture/écriture sur périphérique .

Le bus de commande comprend 4 signaux d'état qui sont :

- le signal VMA (Valid Memory Address) : Ce signal passe à l'état "1" lorsqu'une adresse est valide sur le bus d'adresses.
- la sortie E (Enable ∅2) de l'horloge :

La broche E fournit un signal d'horloge pour le MPU et le reste du systéme . Ce signal, compatible TTL,est un signal à une seule phase (équivalent à la phase ∅2 du 6800).

Cette horloge peut être commandée par le signal MR.

- le signal BA (Bus Available) est activé lors d'une demande de DMA par le signal $\overline{HALT}$ ou lorsque Le MPU est en position WAIT(attente d'interruption).Il indique alors la disponibilité du bus d'adresses.
- le signal R/$\overline{w}$ (Read / $\overline{Write}$) .

d) Les signaux de commande du microprocesseur :

Le 6802 posséde 2 entrées interruption :

- $\overline{IRQ}$: demande d'interruption masquable par programme
- $\overline{NMI}$: interruption non masquable .

L'initialisation du microprocesseur se fait par le signal $\overline{RESET}$.

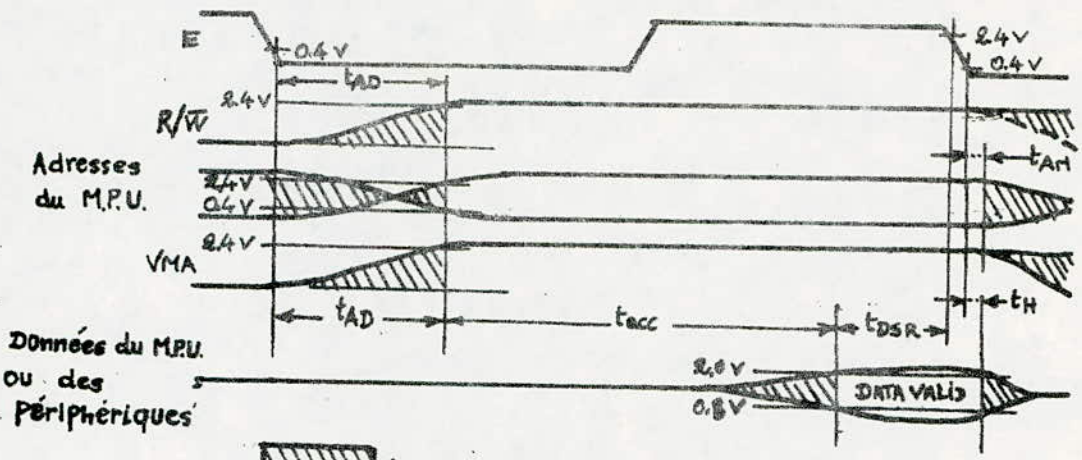
Le signal $\overline{HALT}$ permet un accés direct mémoire (DMA) par arrêt du MPU . Les bus d'adresses et de données ainsi que le signal R/$\overline{w}$ sont alors à l'état haute impédance .

Les broches Extal et Xtal sont réservées pour l'emplacement d'un quartz externe .

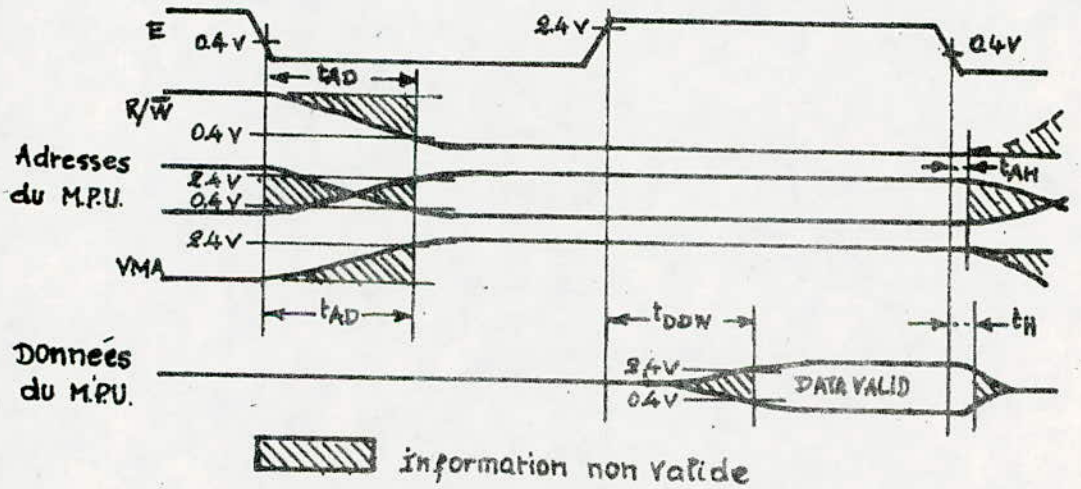
L'entrée RE de validation de la RAM interne (RE = RAM Enable):

Cette entrée compatible TTL valide, par le niveau logique 1, la RAM de 128 octets intégrée dans le 6802 . Un niveau 0 sur cette entrée met la RAM hors circuit .

- LECTURE DES DONNÉES EN MÉMOIRE OU EN PROVENANCE DES PÉRIPHÉRIQUES



- ÉCRITURE DES DONNÉES EN MÉMOIRE OU EN CIRCUITS PÉRIPHÉRIQUES



t_{AD} :	Max 270ns	Temps de retard pour les adresses
t_{acc} :	max 530ns	Temps d'accès à la lecture $t_{acc} = t_{ut} - (t_{AD} + t_{DSR})$
t_{DSR} :	min 100ns	Temps d'établissement des données (en lecture).
t_H :	min 10 ns	Temps de maintien des données (en lecture)
t_H :	min 20 ns	" " " " " (en écriture).
t_{AM} :	min 20ns	Temps de maintien des adresses (adresse, R/$\overline{W}$, VMA).
t_{DDW} :	max 225ns	Temps de retard pour les données (en écriture).
tpcs :	min 200 ns	Temps d'établissement des lignes de commande du M.P.U.
tpcr, tpcf :	max 100ns	Temps de montée et de descente.
t_{BA} :	max 250 ns	Temps de retard pour le signal bus disponible (BA).

L'entrée RE peut être utilisée pour empêcher toute opération de lecture ou d'écriture de la mémoire pendant une diminution de la puissance d'alimentation. RE doit être à l'état bas 3 cycles avant que Vcc ne soit descendu au dessous de 4,75v(voir fig 12)

- L'entrée MR (Memory Ready) : ce signal, compatible TTL, permet l'allongement du signal d'horloge E .

L'entrée MR est mise au niveau logique 1

lorsque le MPU fonctionne normalement(mode synchrone).

Lorsque MR est à l'état bas,E est allongé d'un nombre entier de demi - périodes ce qui permet au microprocesseur l'accès aux mémoires lentes et aux organes d'E/s lents.

(voir fig 8)

c) L'alimentation :

Le 6802 ne nécessite qu'une seule tension d'alimentation Vcc = + 5v . D'où son avantage d'être compatible avec les circuits intégrés TTL .

La broche Vcc standby : Cette broche est l'alimentation des 32 premiers octets de la RAM interne et des circuits de commande de cette RAM . La consommation est de 8mA pour Vcc standby = 5,25 volts

La gestion du MPU s'organise autour de tous les signaux de commande, la figure 13 est un organigramme décrivant les principaux chemins de décision et les vecteurs d'interruption du microprocesseur .

Le tableau 1 donne l'implantation en mémoire des vecteurs d'interruption .

Les interruption du 6802 :

Le 6802 posséde 4 vecteurs d'interruption qui sont :

- $\overline{\text{Reset}}$
- $\overline{\text{NMI}}$

.../...

- SWI

- $\overline{IRQ}$

Initialisation ($\overline{Reset}$) :

La transition de "0" à "1" de cette entrée provoque si $\overline{HALT}$ = 1 une initialisation du MPU :

- la pose du masque d'interruption (I = 1), ce qui inhibe toute demande d'interruption sur l'entrée $\overline{IRQ}$.
- la mise à "0" de BA et le chargement du PC par le contenu des positions mémoire d'adresse FFFE - FFFF .

La ligne $\overline{Reset}$ doit être maintenue à l'état bas au moins pendant 3 cycles d'horloge(indépendant des 20ms nécessaires lors de la remise sous tension) . Ceci permet une bonne initialisation du MPU(voir fig 11).

Interruption Non Masquable ($\overline{NMI}$):

Le bit I d'interruption n'a aucune influence sur la prise en compte de cette interruption .

Un front descendant sur l'entrée $\overline{NMI}$ provoque après éxécution de l'instruction en cours, l'éxécution de la séquence spécifique suivante :

- sauvegarde du contexte dans la pile
- pose du masque d'interruption
- chargement dans le Pc de l'adresse contenue dans les positions mémoires FFFC - FFFD .

Demande d'Interruption ($\overline{IRQ}$):

Cette interruption n'est prise en compte que si le bit d'interruption I est à 0 .

Un niveau "0" sur $\overline{IRQ}$ provoque alors l'éxécution de la séquence suivante :

- sauvegarde du contexte dans la pile
- pose du masque d'interruption
- chargement dans le PC de l'adresse contenue dans les positions-mémoire FFF8 - FFF9 .

Les entrées $\overline{NMI}$ et $\overline{IRQ}$ sont des lignes d'interruption qui sont échantillonnées lorsque le signal E est à l'état haut et la séquence d'interruption ne débute que lorsque le signal E est à l'état bas après l'éxécution de l'instruction en cours .

Interruption software (SWI) :

L'éxécution de cette instruction conduit à la séquence suivante

- sauvegarde du contexte dans la pile
- pose du masque d'interruption
- chargement du PC à l'adresse du programme d'interruption contenue dans FFFA - FFFB .

L'organigramme de gestion du MPU représenté figure 13 fait intervenir l'instruction " WAIT" d'attente d'interruption.

Lorsque le MPU reconnaît cette instruction il procéde à la pose du masque d'interruption et à la sauvegarde du contexte dans la pile .

Il se met ensuite en attente d'une interruption $\overline{NMI}$ ou $\overline{IRQ}$. l'arrivée d'une interruption fait sortir le MPU de l'état WAIT pour le brancher directement au sous programme de l'interruption requise .

II.3 Fonctionnement .

II.3.1 Les instructions du 6802 :

TABLEAU 1

Vecteur	MS LS	Description
$\overline{RESET}$	FFFE FFFF	Redemarrage (Restart)
$\overline{NMI}$	FFFC FFFD	Interruption non Masquable
SWI	FFFA FFFB	Interruption programmée
$\overline{IRQ}$	FFF8 FFF9	Demande d'interruption

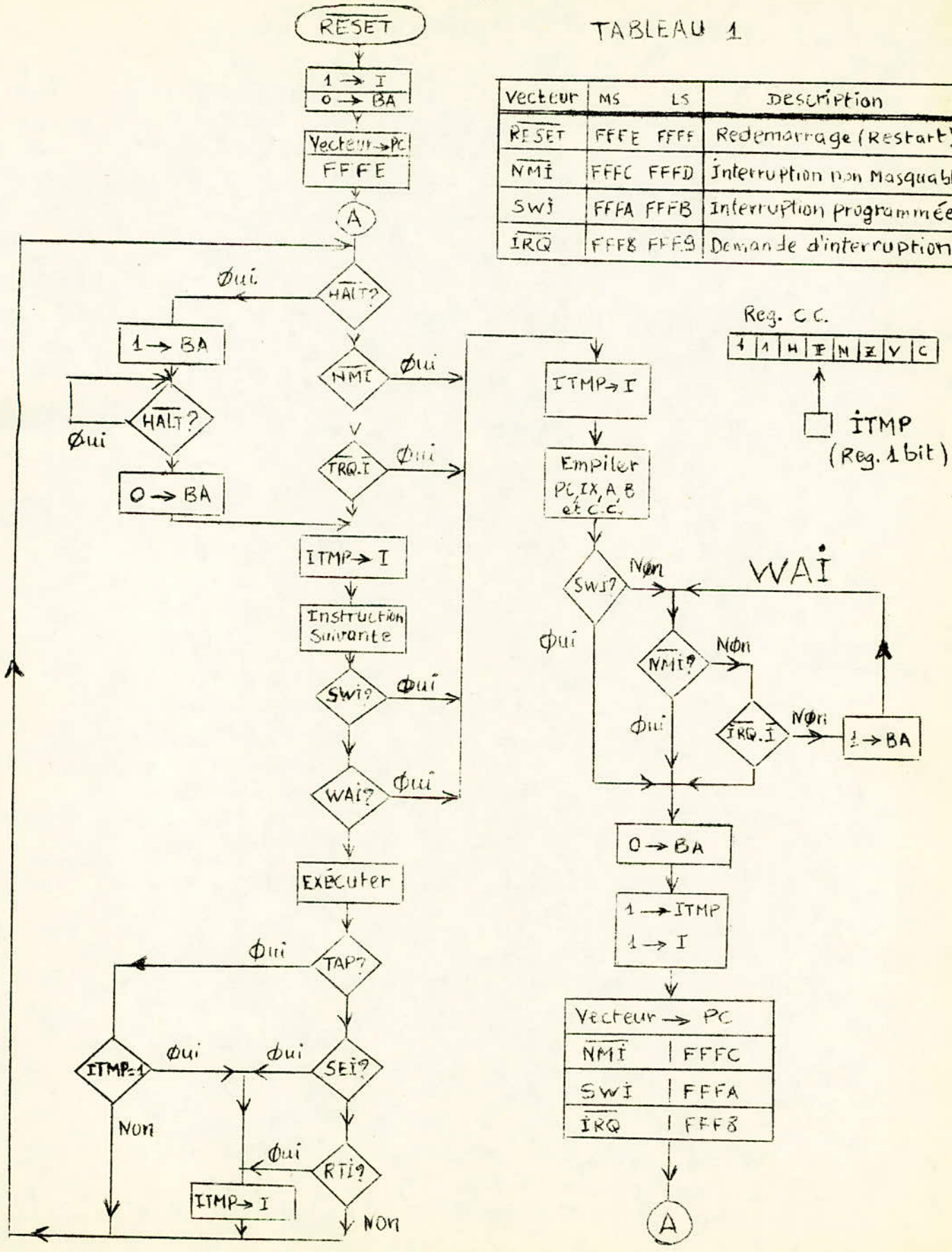


fig 13 - Organigramme de gestion du MPU 6802

Le MPU posséde un jeu de 72 instructions différentes .

Ce jeu comprend les instructions suivantes : arithmétique binaire et décimale, logique, décalages, décalages circulaires, chargements, stockages, branchements conditionnels et inconditionnels, instructions de manipulation de pile et instructions associées aux interruptions . Ce jeu d'instructions est identique à celui du 6800 .

II.3.2 Les modes d'adressage du 6802 :

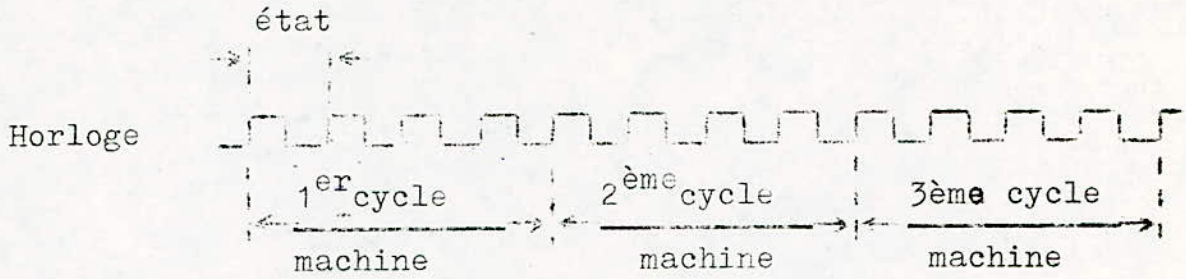
Le 6802 utilise 7 modes d'adressage, les mêmes que ceux du 6800 :

- Adressage accumulateur
- Adressage immédiat
- Adressage direct
- Adressage étendu
- Adressage indexé
- Adressage implicite
- Adressage relatif.

II.3.3 Exécution d'une instruction

L'éxécution d'une instruction se fait en synchronisme avec l'horloge Ø2 du système.Ainsi, pour exécuter une instruction, le MPU doit réaliser une suite d'opérations élémentaires : les micro-instructions . L'horloge synchronise l'ordre d'apparition de ces micro-instructions spécifiées par l'unité de commande

La figure ci dessous montre l'organisation en 3 cycles d'une machine d'une instruction .



Le tableau 2 représente l'éxécution cycle par cycle de l'instruction LDA de chargement d'un accumulateur(AouB) en adressage étendu. Il fournit une description détaillée des informations circulant sur le bus adresses, le bus données, la ligne lecture/écriture (R/$\overline{w}$) et la ligne adresse mémoire valide (VMA) pour chaque cycle d'éxécution de cette instruction.

L'instruction LDA en étendu s'éxécute en 4 cycles machines (ou cycles MPU)

cycle ##	ligne VMA	Bus Adresses	ligne R/$\overline{w}$	Bus Données
1	1	Adresse du code opératoire	1	code opératoire.
2	1	Adresses du code op + 1	1	Adress opérande(octet de poids fort)
3	1	Adresses du code op + 2	1	Adresse opérande(octet de poids faible
4	1	Adresse opérande	1	opérande

- tableau 2 -

1er cycle machine : le MPU va chercher le code opératoire de l'instruction en mémoire pour l'envoyer dans la partie code opératoire du registre d'instruction RI où il sera décodé .

2ème cycle machine : le 1er octet de l'adresse est envoyé dans la partie adresse de RI , octet poids forts .

3ème cycle machine : le 2ème octet de l'adresse est amené dans la partie adresse de RI , octet poids faibles.

4ème cycle machine : la partie adresse de RI est déposée sur le bus adresse (adresse opérande). L'opérande est thansferé, via le bus donnée, dans l'accumulateur .

III - ETUDE DU CIRCUIT INTEGRE 6846 :

III 1 - Organisation de ce circuit (voir fig 5) :

Le 6846 est composé d'une ROM de 2K octets programmable par masque , d'un port d'E/S de 8 bits avec lignes de contrôle(équivalent au port B du PIA 6821) et d'un temporisateur compteur de 16 bits programmable de conception trés proche à celle du Timer 6840.

Il est réalisé en technologie MOS ,canal N , grille Silicium et charge à déplétion .

Le boîtier comporte 40 broches .

III 11- Organisation interne (fig 14) :

Le 6846 peut être décomposé en 3 sections fonctionnelles :

a - La mémoire programmable PROM :

Semblable aux mémoires à lecture seule de la famille 6800, elle est adaptée aux besoins d'un systéme minimum .

Deux entrées Chip Select C_{so} et C_{s1} programmables par masque et définies par l'utilisateur, permettent la séléction de la ROM .

Les 2 K octets de la ROM sont adressés à l'aide des entrées adresses Ao -- A_{10} .

b - Le port "d'E/S parallèles" comprend :

- 8 lignes de données et 2 signaux de dialogue CP_1 et CP_2 dont les commandes et le fonctionnement sont entièrement programmables .

CP_1 est toujours en entrée, CP_2 peut être programmé soit en entrée soit en sortie .

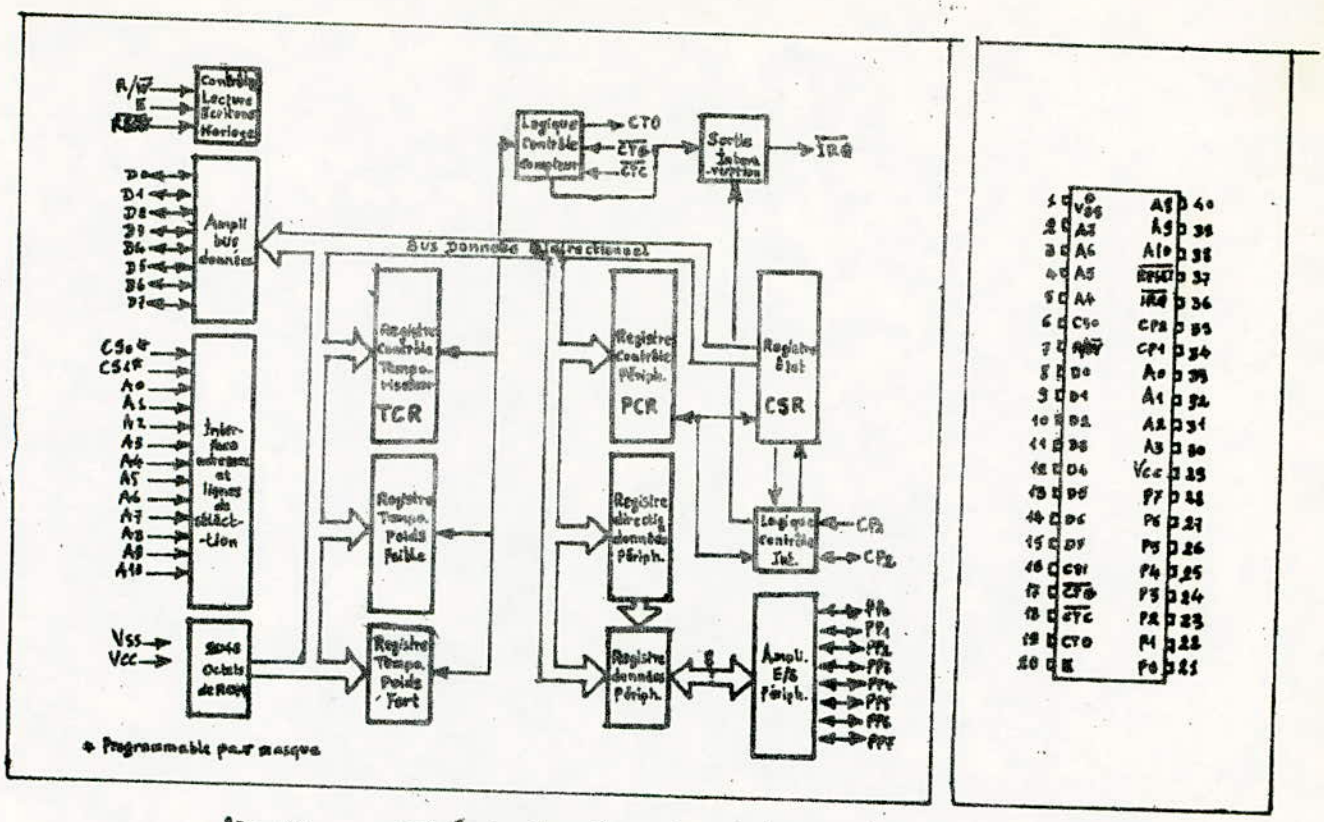


fig 14 - Schéma fonctionnel et brochage du 6846

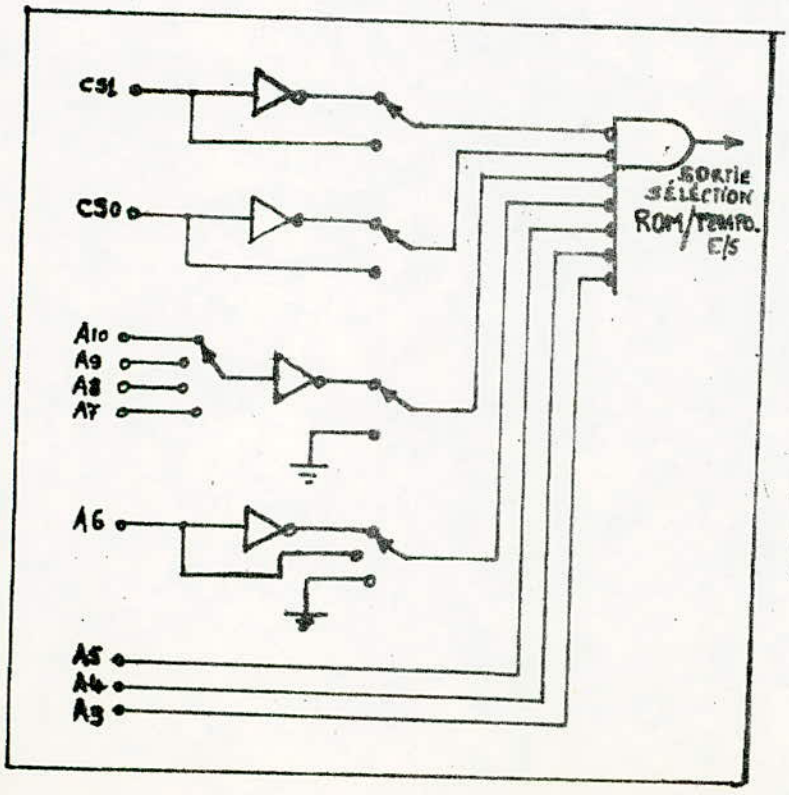


fig 15 - Circuiterie de sélection

A_2	A_1	A_0	REGISTRES SÉLECTIONNÉS
0	0	0	registre d'état Composite (CSR)
0	0	1	registre contrôle périphérique (PCR)
0	1	0	registre direction des données (DDR)
0	1	1	registre données périphériques (PDR)
1	0	0	registre d'état composite (CSR)
1	0	1	registre contrôle temporisateur (TCR)
1	1	0	registre poids fort temporisateur
1	1	1	registre poids faible temporisateur
x	x	x	Adresse ROM

fig 16 - tableau aux adresses des registres du 6846

- 3 registres propres au port d'E/S qui sont :

les registres de commande , de direction et de données .

- 1 registre d'etat commun à l'interface et au temporisateur.

Remarque : Contrairement au PIA ,les registres de direction et de données ont un adressage séparé .

c - Le compteur-temporisateur 16 bits :

Il peut être programmé pour compter des événements, mesurer des fréquences et des intervalles de temps, générer des signaux carrés etc ...

Il comprend 3 registres :

- le registre contrôle temporisateur permettant le contrôle de validation d'interruption, de validation de sortie, de séléction déune source Horloge interne ou externe , d'un précompteur diviseur par 8 et du mode de fonctionnement .

- 2 registres tampons 8 bits (à écriture seule) :

Ce sont deux registres associés au compteur. Le premier (MSB-Buffer registers) est pourvu de l'octet de poids fort, le second (LSB Buffer Register) contient l'octet de poids faible.

La fonction de ces 2 registres tampons est de mémoriser l'équivalent binaire de la valeur du compte désiré moins un .

III$_{12}$- Organisation externe :

Le 6846 s'interface au MPU 6802 via un bus de données bidirectionnel de 8 bits, 2 lignes séléction de boitiers, une ligne de lecture/Ecriture et 11 lignes d'adresses(à cause de la ROM de 2 K octets). Ses signaux avec la sortie VMA permettent au MPU de contrôler le 6846 .

Lorsque le circuit fonctionne en mode ROM(une combinaison parmi les quatres de Cso et Cs_1) toutes les lignes adresses sont utilisées Pour les 3 autres combinaisons de Cso et Cs_1 , le 6846 est séléctionné en temporisateur ou en port d'E/S et seules les lignes d'adresses Ao , A_1 et A_2 sélectionnent leurs registres correspondants : voir figures 15 et 16 .

Les entrées / Sorties du temporisateur :

. Sortie temporisateur compteur CTO (Counter Timer Out put):

La forme de son signal sera programmé par le registre de contrôle temporisateur suivant le mode de fonctionnement désiré. Le mode de fonctionnement dépend du registre de contrôle du Timer, de l'état de l'entrée de déclenchement ($\overline{CTG}$) et de la source Horloge ($\overline{CTC}$) .

. Entrée Horloge Externe $\overline{CTC}$ (external Clock input) :

Cette entrée accepte des signaux asynchrones de niveau TTL. Elle est utilisée comme horloge pour décrémenter le compteur.Elle est synchronisée interieurement par l'horloge E .

3 périodes de E servent à la reconnaissance et à la prise en compte interne d'une transition au niveau bas sur l'entrée $\overline{CTC}$, à la quatrième impulsion de E, le compteur se décrémente .

Un bon fonctionnement nécéssite une bonne synchronisation et une stabilité de l'entrée horloge . Les niveaux hauts et bas de $\overline{CTC}$ doivent être stables pendant au moins une période d'horloge E .

Si le précompteur diviseur par 8 est utilisé , la cadence maximum d'horloge peut être égale à 4 fois la fréquence d'horloge E (4 MH_z) .

Le précompteur sert surtout pour mesurer des temps longs .

. Entrée porte $\overline{CTG}$ (Gate Inputs) :

Elle est utilisée comme signal de déclenchement de l'horloge du temporisateur . De la même manière que $\overline{CTC}$, cette entrée $\overline{CTG}$ est synchronisée par E du système . Une transition de $\overline{CTG}$ est reconnue sur la $4^{ème}$ impulsion de E .

L'entrée $\overline{CTG}$ affecte directement le compteur 16 bits .

La sélection du précompteur */. 8 n'affecte pas cette entrée .

III$_2$ - fonctionnement :

a- principaux registres de programmation : (fig 17)

. Le registre d'état composite (CSR) de 8 bits :

Ce registre à lecture seule intervient lors du fonctionnement du 6846 et il est utilisé par le temporisateur et le port d'E/S .

. Le registre de contrôle du Timer (TCR) de 8 bits :

Ce registre nous détermine totalement le fonctionnement du Timer .

Ses bits 3,4 et 5 sont réservés à la spécification du mode choisi .

. Le Registre de contrôle périphérique (PCR) de 8 bits :

Ce registre est utilisé pour contrôler la fonction de remise à zéro aussi bien que pour la selection des fonctions optionnelles des deux lignes de contrôle périphérique (CP_1 et CP_2) .

b - différents modes de fonctionnement :

Le Timer peut nous fournir 5 modes de fonctionnement :

. Le mode de fonctionnement continu : fig 18.
. Le mode Monocoup normal : fig 19
. Le mode Cascade monocoup: fig 20
. Les modes intervalle de temps :
  - mode comparaison de fréquence
  - mode comparaison de largeur d'impulsion.

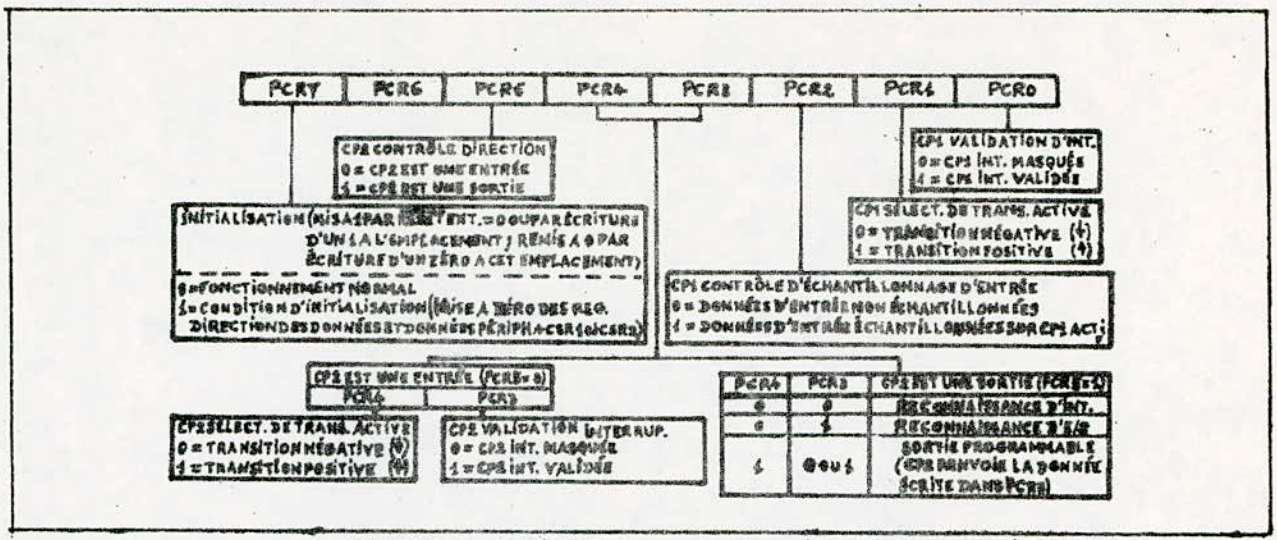


TABLEAU DU REGISTRE DE CONTRÔLE PÉRIPHÉRIQUE (PCR)

TABLEAU DU REGISTRE D'ÉTAT COMPOSITE (CSR)

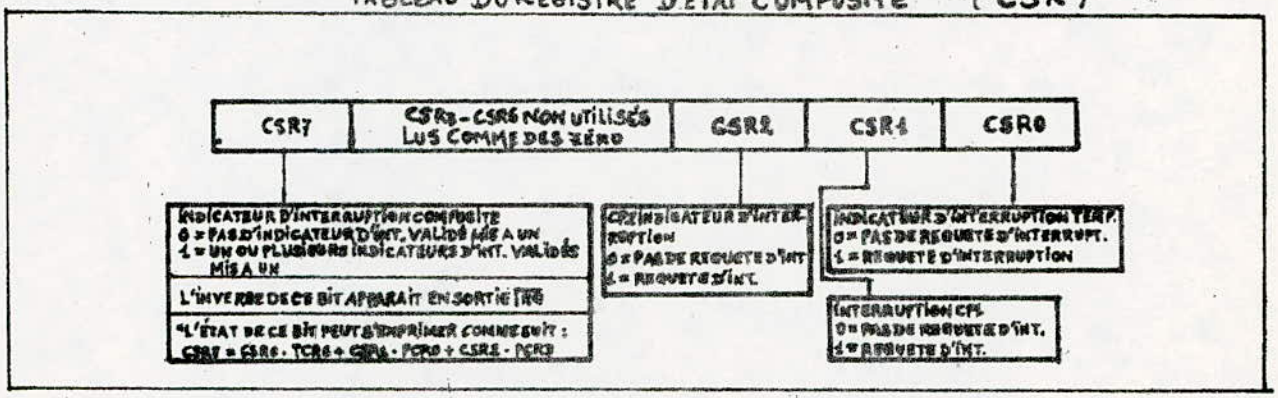
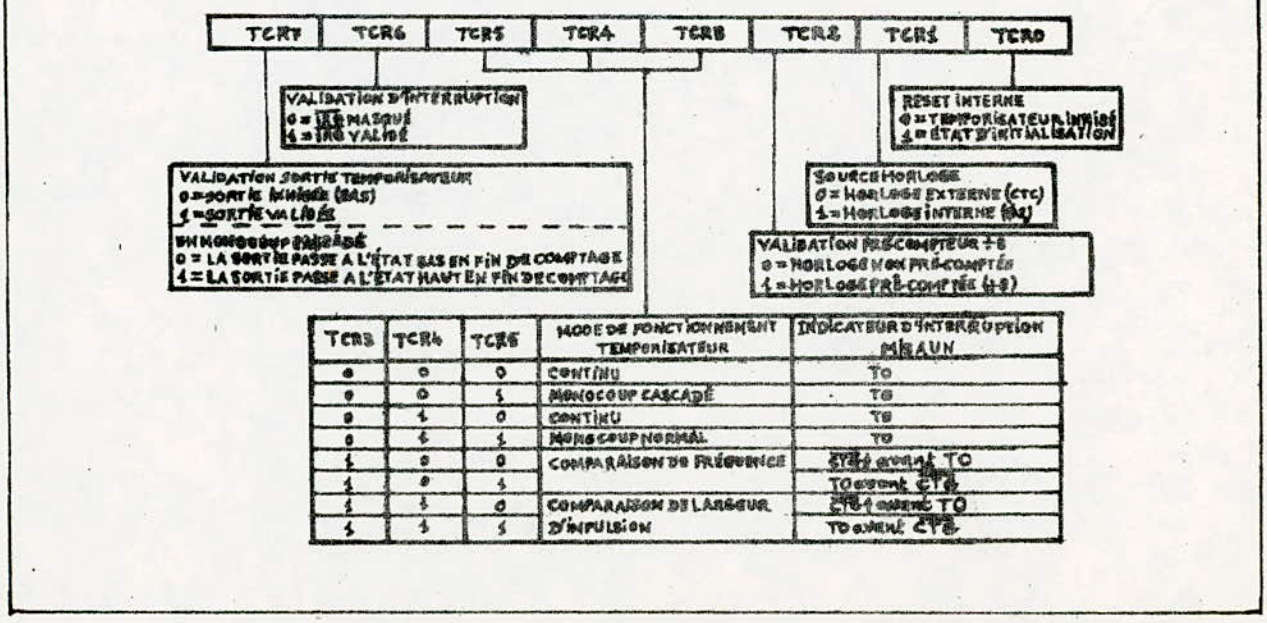


TABLEAU DU REGISTRE DE CONTRÔLE TEMPORISATEUR (TCR)



TCR3	TCR4	TCR5	MODE DE FONCTIONNEMENT TEMPORISATEUR	INDICATEUR D'INTERRUPTION MIS A UN
0	0	0	CONTINU	TO
0	0	1	MONOCOUP CASCADÉ	T0
0	1	0	CONTINU	T0
0	1	1	MONOCOUP NORMAL	T0
1	0	0	COMPARAISON DE FRÉQUENCE	CT0↑ avant TO
1	0	1		TO avant CT0↑
1	1	0	COMPARAISON DE LARGEUR D'IMPULSION	CT0↑ avant TO
1	1	1		TO avant CT0↑

figure 17. PRINCIPAUX REGISTRES DE PROGRAMMATION

fig 18 - MODE CONTINU

TCR3 = 0, TCR7 = 1, TCR5 = 0

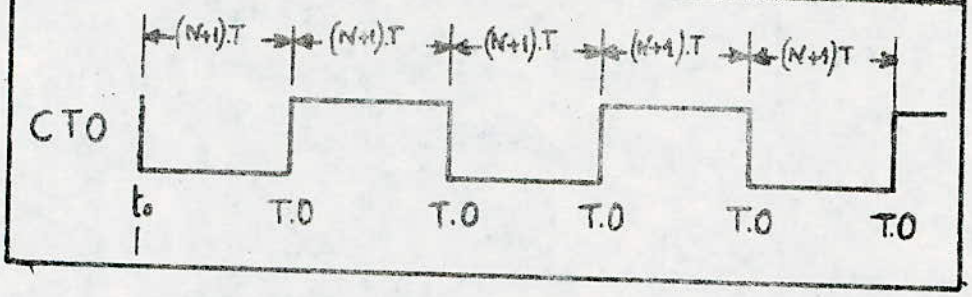


fig 19 - MODE MONOCOUP NORMAL

TCR3 = 0, TCR4 = 1, TCR5 = 1

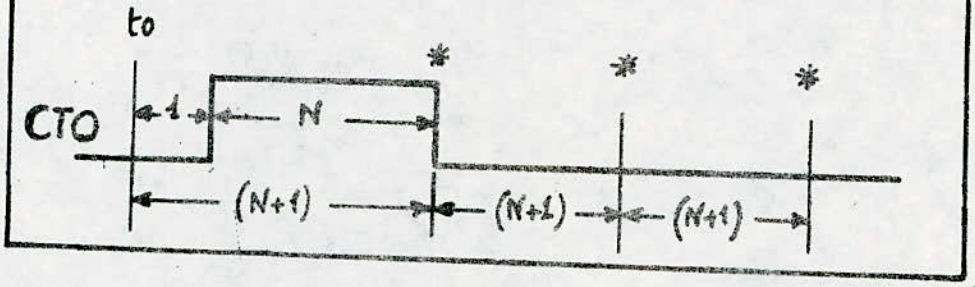
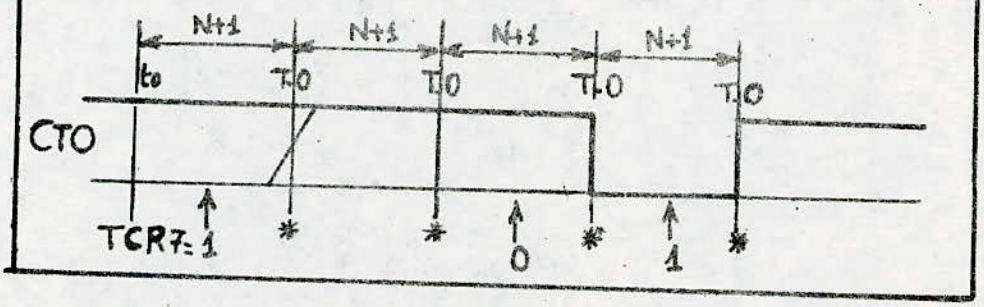


fig 20 - MODE CASCADE MONOCOUP

TCR3 = 0, TCR4 = 0, TCR5 = 1



N = Nombre de 16 bits dans le registre tampon du compteur

T = période

to = Cycle d'initialisation Compteur

T.O = Fin de comptage

* = point sur lequel peut survenir une interruption

C H A P I T R E 2. Etude d'un système a extension autour du 6802 : Le KIT D5 de MOTOROLA

I - Description du KIT :

Le KIT MEK 6802 D5 de MOTOROLA est un micro ordinateur permettant à l'utilisateur de s'initier à la famille des microprocesseurs MC 6800 .

Les 3 composants de base du D5 sont :

Le MPU 6802 (U5), la ROM "D5 BUG " (U12) et le "SYSTEM PIA" (U23) .

Le MPU 6802 permet la gestion du KIT sous le contrôle du Moniteur contenu dans la ROM " D5 BUG " .

Le"SYSTEM PIA " permet le dialogue clavier-Microprocesseur en entrée et Microprocesseur-Afficheurs en sortie .

Le langage utilisé sur le KIT est l'assembleur codé hexadécimal

- présentation des blocs fonctions(fig 1) :

Le KIT D5 est constitué par différents blocs fonctionnels qui sont :

A- Le bloc de commande de tout le système:Le MPU 6802

(pour une étude plus détaillée voir chap 1).

Toutes les entrées de commande d'interruption et $\overline{\text{RESET}}$ sont à l'état haut .

La commande $\overline{\text{RESET}}$ s'éffectue à travers un circuit RC(R1,C17) telle que la routine d'initialisation de tout le système ne démarre que lorsque le potentiel de la broche 40 atteint 0,8 V .

La broche "MR" est dans son état de repos (+ 5 V).

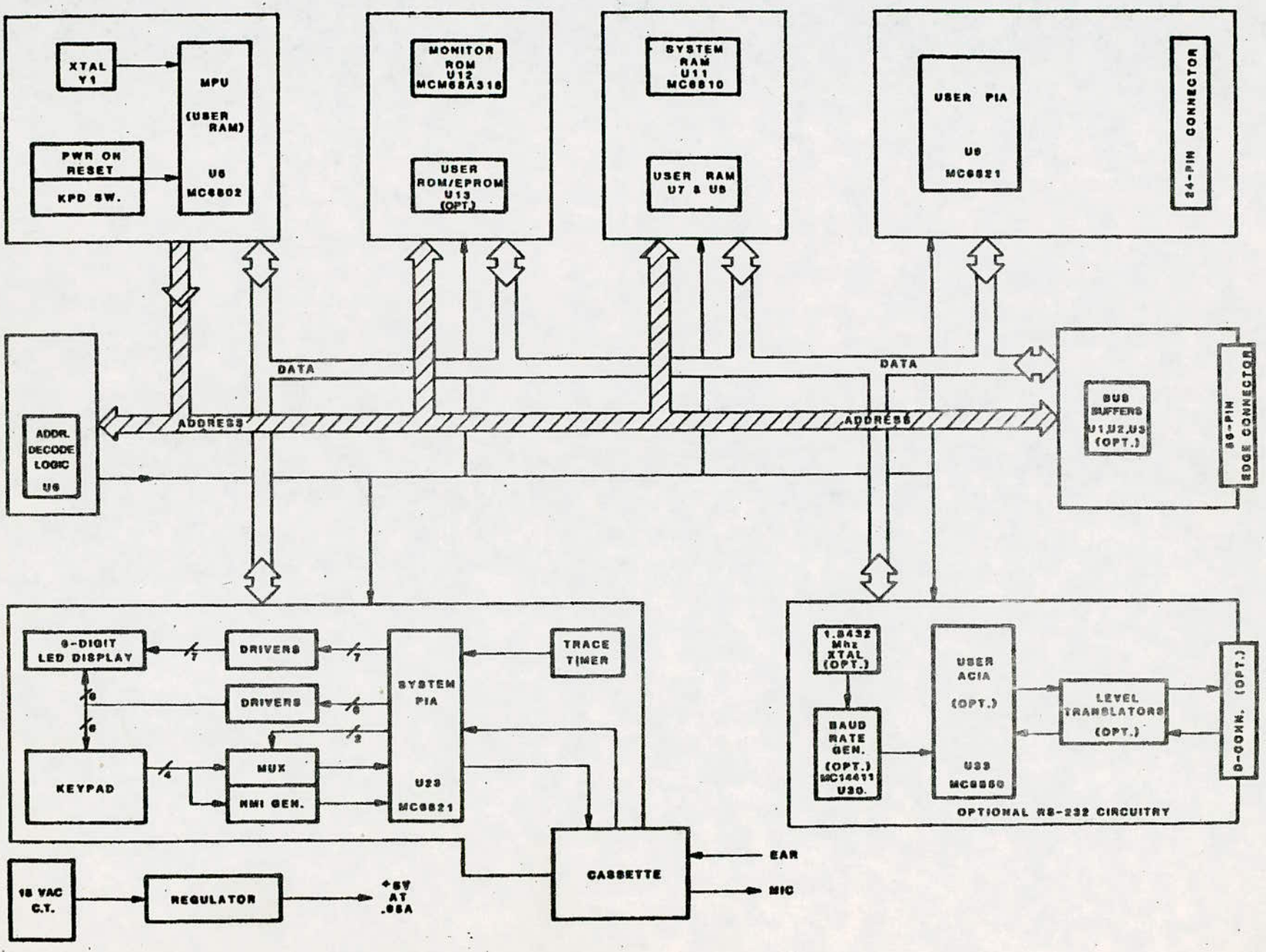


FIGURE 1. BLOCK DIAGRAM

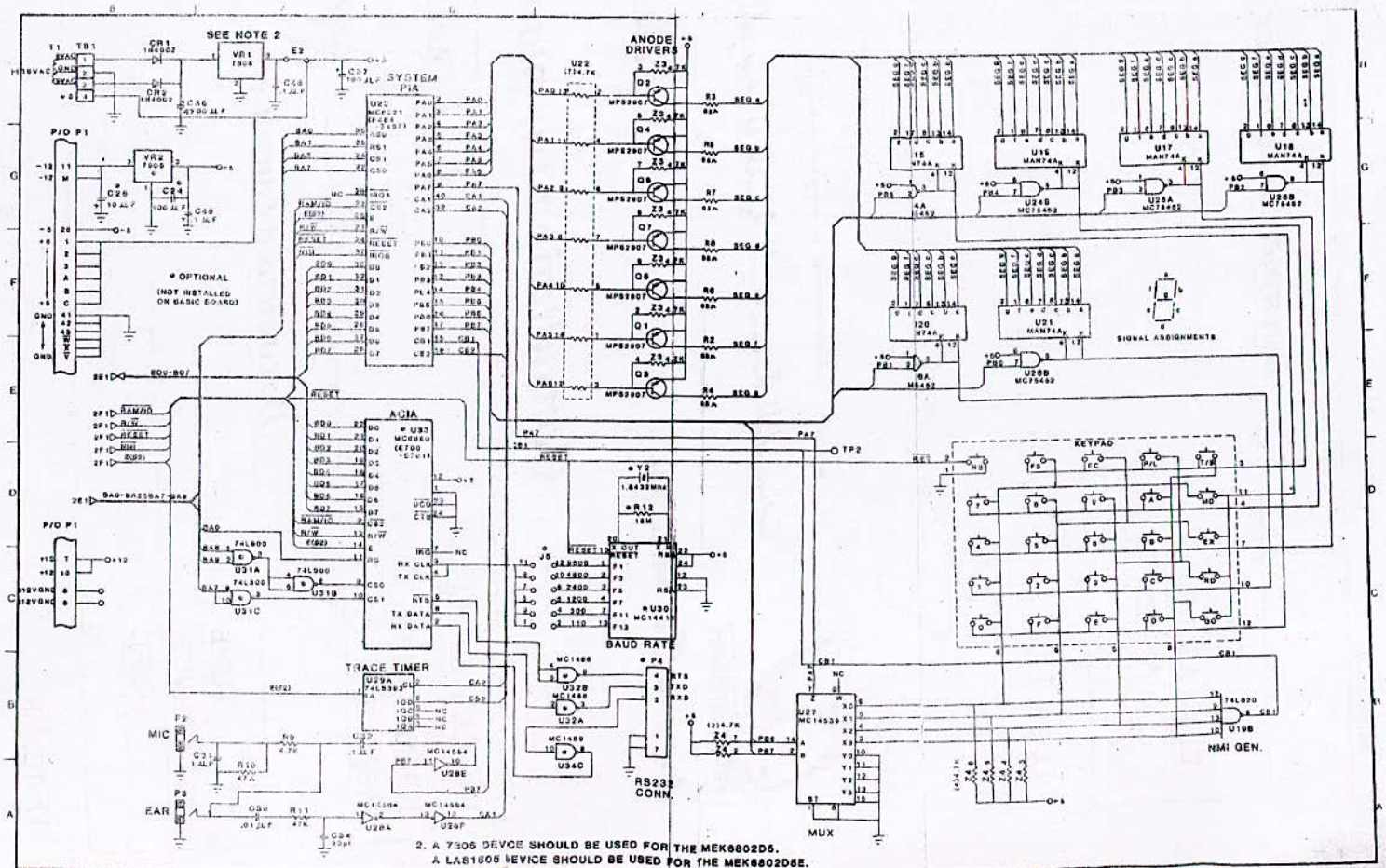


FIGURE 5.1 SCHEMATIC DIAGRAM
(SHEET 3 OF 3)

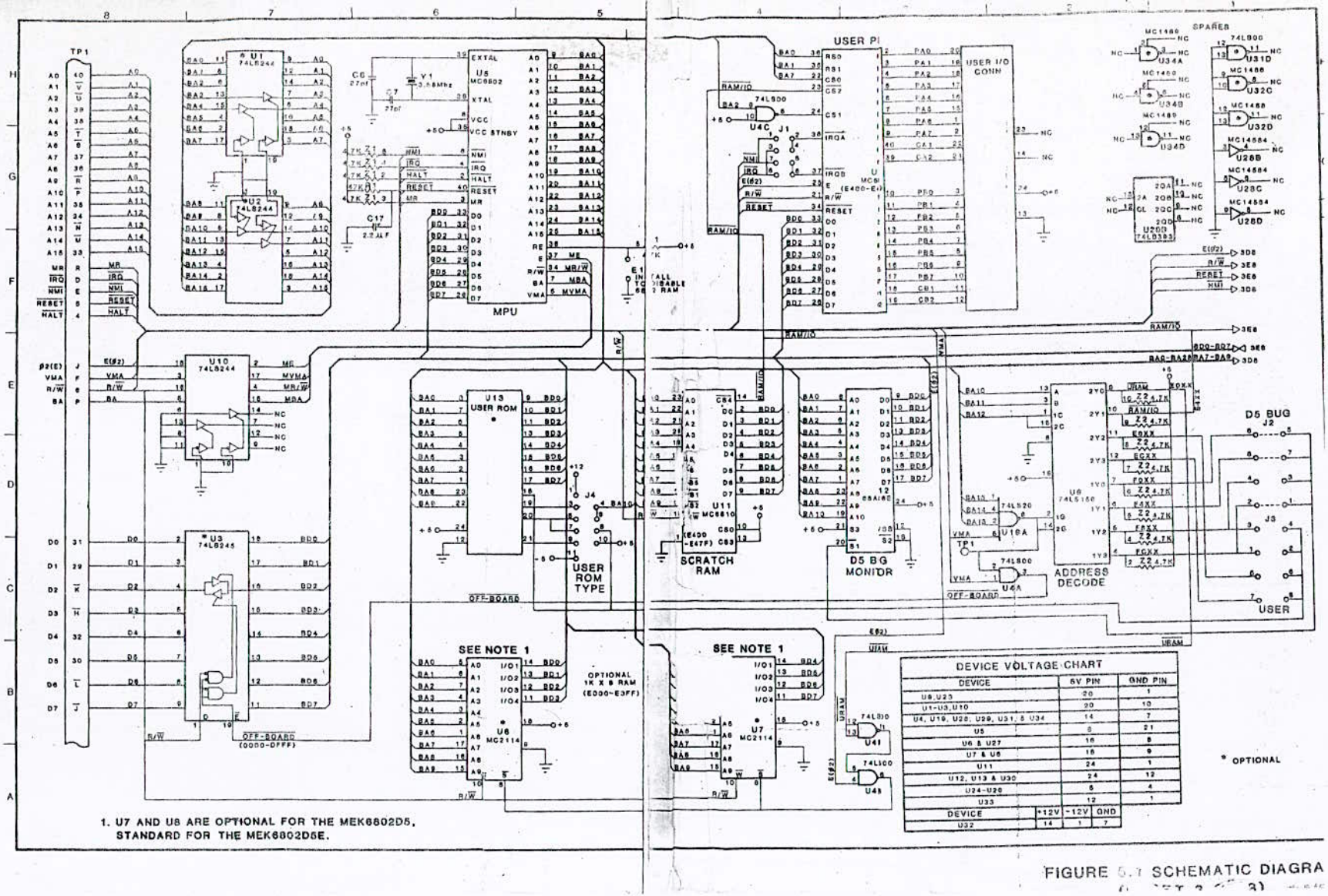


FIGURE 5.1 SCHEMATIC DIAGRA

La carte utilise un quartz de 3,58 MHz ce qui donne une fréquence de fonctionnement (issue de l'oscillateur interne) de 3,58 MHz $\div$ 4 = 0,895 MHz ; celle ci correspond à une période d'horloge E ($\emptyset_2$) = 1,117 µs .

Le Vcc Standby est relié directement à Vcc = + 5 V, ce qui ne permet pas la sauvegarde des 32 premiers octets de la RAM interne une fois l'alimentation coupée. La RAM interne peut être supprimée par le Strap E1 en cas d'utilisation de son adressage pour des circuits externes au KIT .

B- Bloc du KEYPAD, du SYSTEM PIA(U_{23}) des AFFICHEURS et de l'interface CASSETTE :

Le clavier (ou KEYPAD) permet de faire rentrer des données qui seront prises en compte par le MPU à travers le System PIA(port B + PA7) ; leur affichage s'effectue par l'envoi sur les lignes du Port A du code 7-segmets du digit à afficher .

Les lignes du port A attaquent les anodes des LEDS de l'afficheur à travers l'ampli " ANODE DRIVERS" dont chaque transistor travaille en Bloqué - Saturé .

La détection de la position (ligne, colonne) de la touche appuyée sur le clavier se fait par tout un ensemble de circuits logiques qui sont : le "NMI GENERATOR" (U_{19B}), le décodeur de position ou multiplexeur "MUX" (U_{27}) et les drivers en 6 portes NAND(U_{24A}, U_{24B}, U_{25A}, U_{25B}, U_{26A}, U_{26B}).

Pendant la phase de réinitialisation du système, le "SYSTEM PIA" est programmé en sortie pour le port B ; le port A est aussi en sortie à l'exception de PA7 qui sert d'indicateur pendant la phase de recherche de la touche enfoncée (voir au paragraphe III la routine GET) .

Les 6 premières lignes du port B sont reliées à travers des dri -vers aux cathodes des afficheurs .

On utilise PB6 et PB7 pour la recherche de la position de la touche appuyée sur la matrice (clavier).

La Sortie du " NMI GENERATOR" est reliée à l'entrée CB1 du " SYSTEM PIA" ce qui fait qu'on peut déclencher une interruption $\overline{NMI}$ sur le MPU à partir de n'importe quelle touche du clavier (la touche "RS" non comprise).

Ce même "SYSTEM PIA" est relié à un interface CASSETTE capable de détecter ou de délivrer des informations binaires modulées en fréquence. Elles sont d'amplitude 50 mVcc et peuvent être soit lues par le microprocesseur (entrée EAR) , soit enregistrées sur cassette (sortie MIC):

fréquence du "0" = 1200 Hz

fréquence du "1" = 2400 Hz

Le "TRACE TIMER " (U_{29A}) sert à déclencher une interruption sur la ligne de commande CB2 . Ce circuit (74LS393) est un compteur qui est synchronisé par l'horloge E($\emptyset_2$) du système . Les sorties A,B et C sont non connectées, de ce fait il constitue un diviseur de fréquence par 16 (sortie D).

C- Bloc à mémoires mortes (ROM) :

C'est un bloc de mémoire à lecture seule dont la "D5 BUG" (U_{12}) de 2 K octets est indispensable pour la gestion du KIT .

L'"USER ROM" (EPROM "U_{13}") qui peut aller jusqu'à 2 K octets est par contre optionnelle .

La prise de connection J4 nous permet d'utiliser différents types d'EPROM qui peuvent être :

- soit une 2708 de 1 Koctets, tri-tensions(+5V,-5V,+12V)
- soit une 2716 de 2 Koctets,monotension (+5V).

Les connecteurs J2 et J3 sont utilisés pour configurer la "D5 BUG" et la "USER ROM" dans l'espace mémoire (voir paragraphe II_2

D- Bloc à mémoires vives (RAM):

En plus de la RAM interne du 6802 , le KIT possède 2 autres RAM du Type volatile qui sont :

La "SCRATCH RAM" (MC 6810 "U_{11}") composée de 128 octets et l'"USER RAM" divisée en 2 boîtiers(U_7 +U_8) du type MC 2114 de capacité 1K X4bits .

La "SCRATCH RAM" est divisée en 3 parties dont 2 sont réservées spécialement pour la ROM moniteur de gestion D5 BUG (pour son fonctionnement et pour sa propre pile).La 3^{eme}partie est prévue pour assurer le rôle d'une pile à l'usage de l'utilisateur .

Les 2 RAMS U_7 et U_8 sont constituées par des cases mémoires de 4 bits . En étant placées en parrallèle , elles forment le mot complet de 8 bits .

E- le Décodeur d'adresses (U_6):

C'est grâce à ce circuit logique du type 74LS156 que l'on arrive à simplifier au maximum l'adressage sur la carte .

Son rôle principal est de décoder les lignes d'adresse de poids forts pour former 8 sorties dont chacune sélectionnera un boîtier ou des zônes mémoires .

F- bloc interface serie:

Ce bloc en option sur le KIT est composé d'un interface série l'"USER ACIA" (U_{33}) et d'un générateur de fréquence (U_{30}).

Sur notre carte, il pourra être utilisé pour des applications d'Entrée /Sortie en série exemple :
la télétype, la console à écran à tube cathodique,la transmission avec MODEM etc ...

La synchronisation s'éffectue par l'horloge "U_{30}"(qui génére plusieurs fréquences).

G- bloc interface parallèle "USER PIA"(le MC6821 "U_9"):

C'est un interface prévu spécialement pour l'utilisateur et son usage se rapporte à faire la liaison entre le MPU et les périphériques(exemple: convertisseur,imprimante, afficheurs etc...).

Son connecteur à 24 pins nous sera trés utile pour des connections rapides .

H- bloc BUFFERS (U_1 , U_2 et U_3 en option):

Ce sont des amplificateurs des lignes d'adresses et de données.

Ils sont utilisés pour l'extension du KIT D5 à travers un connecteur de 86 pins .

II - Répartition des adresses sur la carte :

II-1 Répartition générale :

La répartition des adresses pour chaque boîtier (RAM,ROM, interfaces parallèles PIA, interface série ACIA) est donnée à la fig 2 . Les premières adresses de $ 0000 à $ 007F sont réservées pour la RAM interne du 6802 . Les adresses de $ 0080 à $DFFF (en tout 56 K bytes) sont utilisées pour l'adressage de circuits externes au KIT . Il est possible de supprimer la RAM interne et de jus taposer son adressage à celui des circuits externes. La "Scratch RAM" , le "User PIA" et le "System PIA" ont un adressage successif allant de $ E400 à $ E487 ceci pour faciliter leur initialisation (leur effacement par la routine $\overline{\text{RESET}}$)

Deux adresses mémoires sont réservées pour une éventuelle utilisation d'un interface série .

La carte D5 est conçue pour fonctionner avec 2 ROMS de 2 K octets chacune : la "D5 BUG" qui occupe la zône allant de $ F000 à $ FFFF et la " User ROM" optionnelle pour les adresses allant de $ E800 à $ EFFF .

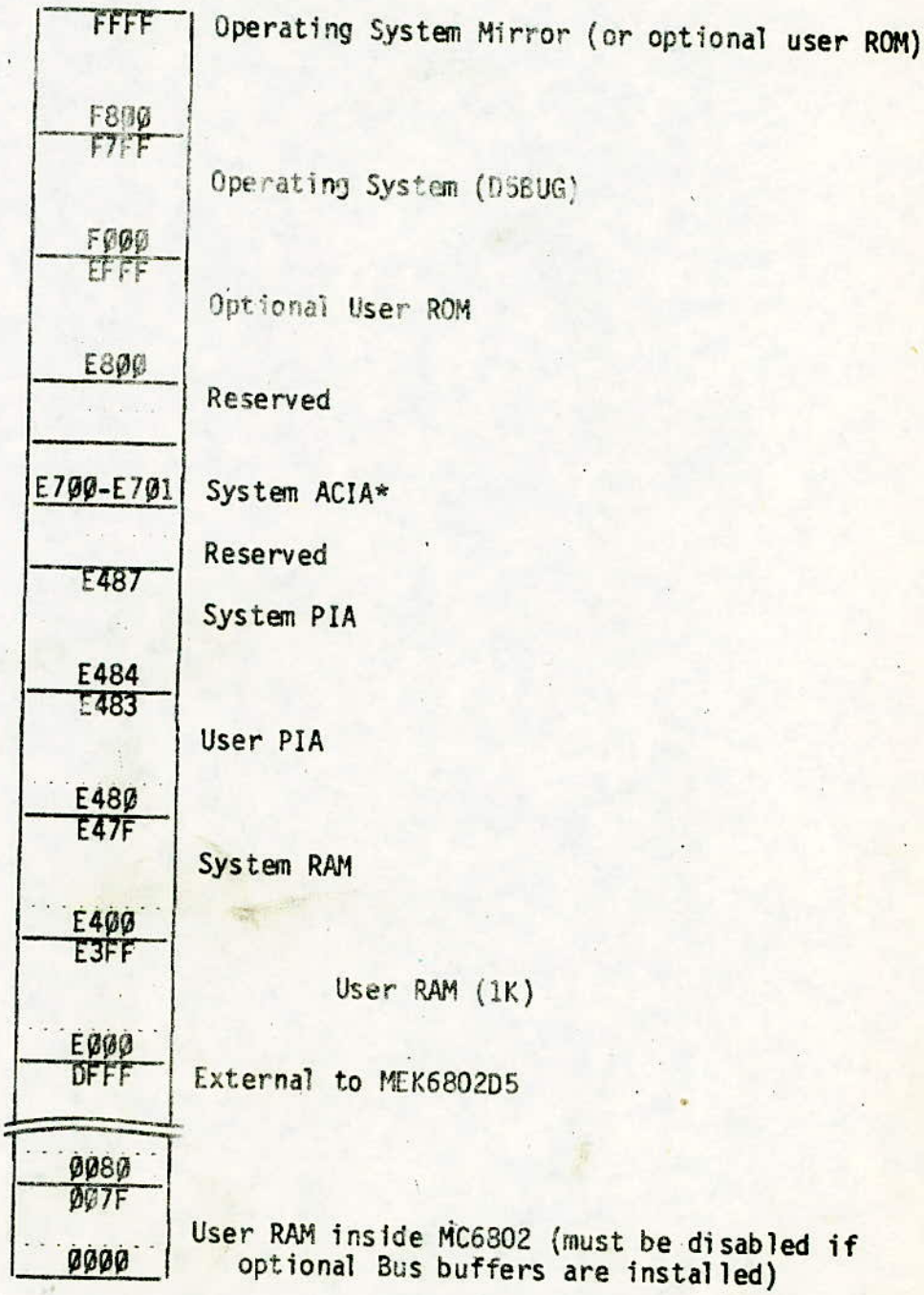
Les zônes mémoire $E488 à $ E6FF , $E702 à $E7FF et $F800 à $ FFFF sont appelées zônes "miroirs" et ne doivent pas être utilisées .

II-2 Le décodage d'adresses :

Le KIT utilise la méthode de d'adressage par décodage .

Cette méthode est intéressante car elle permet d'occuper la plus grande partie possible des 64 K positions mémoire théoriquement disponibles .

Chaque boîtier du KIT est sélectionné à partir du décodeur U6 (circuit 74 LS 156) suivant le tableau de la fig 3 .



*ACIA is not supported by D5BUG software.

fig 2 - MEMORY MAP

fig 3- ADDRESS DECODE MAP

Device	VMA	R/W	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0	U No.
MC68A316E	1	X	1	1	1	1	*	.	.	.	.	.	.	.	.	.	.	.	U12
MCM6810	1	.	1	1	1	0	0	1	0	0	0	.	.	.	.	.	.	.	U11
MC6821 (User)	1	.	1	1	1	0	0	1	X	X	1	X	X	X	X	0	.	.	U9
MC6821 (Syst)	1	.	1	1	1	0	0	1	X	X	1	X	X	X	X	1	.	.	U23
MCM2114	1	.	1	1	1	0	0	0	.	.	.	.	.	.	.	.	.	.	U7,U8
User ROM (Opt. 1)	1	.	1	1	1	0	1	.	.	.	.	.	.	.	.	.	.	.	U13
User ROM (Opt. 2)	1	.	1	1	1	**	1	.	.	.	.	.	.	.	.	.	.	.	U13
MC6802***	1	.	0	0	0	0	0	0	0	0	.	.	.	.	.	.	.	.	U5
MCM6850 (Opt.)	1	.	1	1	1	0	0	1	1	1	0	X	X	X	X	X	X	X	U33
74LS245 (Opt.)	1	.	$0000 - $DFFF (Must install E1)																U3

0 - Logic zero 1 - Logic one . -Both X - Don't Care

* - Normally both,but by removing jumpers it becomes 0 only.

** - 1,but monitor mirror must be disabled.

*** - Inherent to the MC6802 by design, may be disabled for system expansion.

A - Le décodeur d'adresses 74LS 156 : fig 4

Les 8 sorties de ce décodeur ont été étudiées à partir du tableau précédent . Ce sont des combinaisons des lignes d'adresse de poids forts (A10, A11,A12, A13, A14, A15) et de la ligne VMA .

Les branchements sur les entrées du décodeur considèrent que:

$1G_1 = 2G_2 = G$ et $1C_1 = 2C_2 = C$ d'où

$1_{Y0} = \overline{C\,\overline{G}.B.\overline{A}}$ $\qquad$ $2_{Y0} = \overline{\overline{C}\,\overline{G}\,\overline{B}\,\overline{A}}$

$1_{M1} = \overline{C\,\overline{G}\,\overline{B}\,A}$ $\qquad$ $2_{Y1} = \overline{\overline{C}\,\overline{G}\,\overline{B}\,A}$

$1_{Y2} = \overline{C\,G\,B\,\overline{A}}$ $\qquad$ $2_{Y2} = \overline{\overline{C}\,\overline{G}\,B\,\overline{A}}$

$1_{Y3} = \overline{C\,\overline{G}\,B\,A}$ $\qquad$ $2_{Y3} = \overline{\overline{C}\,\overline{G}\,B\,A}$

or A10 = A

A11 = B

A12 = 1C = 2C = C

$\overline{VMA.\ A13.\ A14.\ A15} = \overline{F} = 1G = 2G = G$

d'où:

. $1_{Y0} = \overline{C\,\overline{G}\,\overline{B}\,\overline{A}} = \overline{A12.F.\,\overline{A11}\,.\overline{A10}} = \overline{VMA.A15.A14.A13.A12}$

$\overline{\overline{A11}.\,\overline{A10}} = \overline{VMA\ 1111\ 00} = \overline{VMA\ \ FOXX}$

La sortie 1_{Y0} nous validera donc la partie de la ROM"D5BUG" allant de $ FOOO à $ F3FF (1 K octets)

$1_{Y1} = \overline{C\,\overline{G}\,\overline{B}\,A} = \overline{VMA.A15.A14.A13.A12.\overline{A11}.A10} = \overline{VMA\ 1111\ 01}$

$= \overline{VMA\ \ F4XX}$.

$1_{Y2} = \overline{C\ \overline{G}\ B\ \overline{A}} = \overline{VMA.A15.A14.A13.A12.A11.\overline{A10}} = \overline{VMA\ 1111\ 10}$

$= \overline{VMA\ F8\ XX}$.

$1_{Y3} = \overline{C\ \overline{G}\ B\ A} = \overline{VMA\ \ 1111\ \ 11} = \overline{VMA\ FC\ XX}$.

$2_{Y0} = \overline{\overline{C}\ \overline{G}\ \overline{B}\ \overline{A}} = \overline{VMA.A15.A14.A13.\overline{A12}.\overline{A11}.\overline{A10}} = \overline{VMA\ 1110\ 00}$

$= \overline{VMA\ E0\ XX}$.

$2_{Y1} = \overline{\overline{C}\ \overline{G}\ \overline{B}\ A} = \overline{VMA\ 1110\ 01} = \overline{VMA\ E4\ XX}$

$2_{Y2} = \overline{\overline{C}\ \overline{G}\ B\ \overline{A}} = \overline{VMA\ 1110\ 10} = \overline{VMA\ E8\ XX}$

$2_{Y3} = \overline{\overline{C}\ \overline{G}\ B\ A} = \overline{VMA\ 1110\ 11} = \overline{VMA\ EC\ XX}$

Par conséquent :

La sortie $2_{Y0} = \overline{URAM} = \overline{VMA\ E0XX}$ nous permettra de sélectionner les 2 RAMS U7 et U8 de 1K octets placées en parallèle .

La sortie $2_{Y1} = \overline{RAM/IO} = \overline{VMA\ E4XX}$ nous sélectionne la "Scratch RAM" , l'ACIA , le "User PIA" et le " System PIA " .

Les 2_{Y2}, 2_{Y3}, 1_{Y2} et 1_{Y3} sont réservées pour valider la " User ROM ".

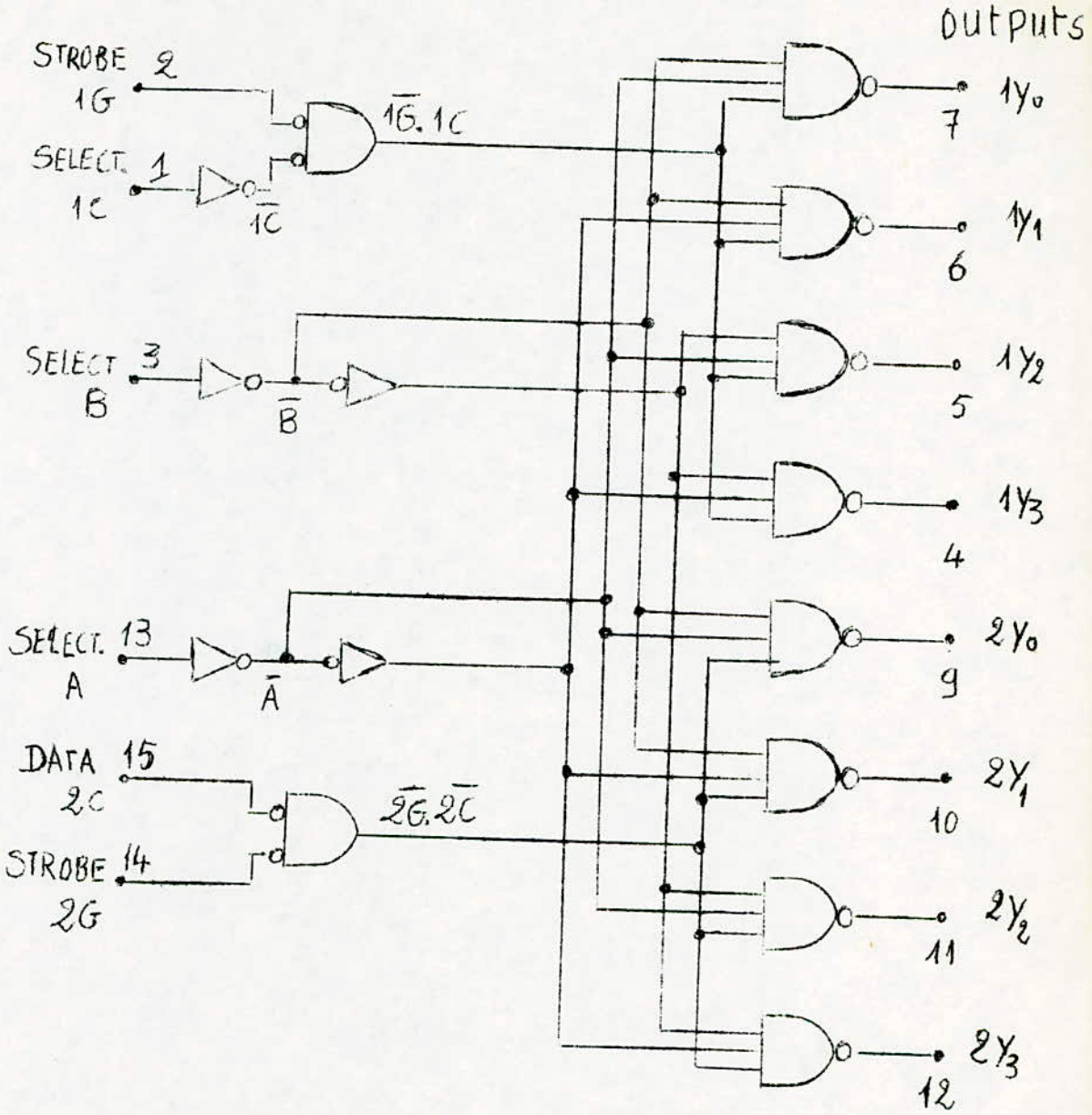
Les sorties 1_{Y0} et 1_{Y1} sont utilisées pour valider l'adressage sur toute la ROM "D5BUG" .

Quand à la sortie 1_{Y3} , elle sert à valider la zône miroir de $ FC00 à $ FFFF . Cette dernière sera exploitée par les vecteurs d'interruption du système .

B - <u>L'adressage externe du KIT :</u>

Pour cet adressage de capacité 56 K octets, le KIT utilise pour la validation en sortie ou en entrée du bus de données une

.../...



$1Y_0 = \overline{1C.1G.\overline{B}.\overline{A}}$

$1Y_1 = \overline{1C.1G.\overline{B}.A}$

$1Y_2 = \overline{1C.1G.B.\overline{A}}$

$1Y_3 = \overline{1C.1G.B.A}$

$2Y_0 = \overline{2C.2G.\overline{B}.\overline{A}}$

$2Y_1 = \overline{2C.2G.\overline{B}.A}$

$2Y_2 = \overline{2C.2G.B.\overline{A}}$

$2Y_3 = \overline{2C.2G.B.A}$

fig 4 - Le DÉCODEUR (U6) : 74LS156 DC

porte NAND (U4A) et le C.I DATA BUFFER (U3 qui est en option)

- à la sortie de U4A on a : $f = \overline{\overline{F}.VMA} = \overline{\overline{VMA.A15.A14.A13}.VMA}$

$= \overline{VMA.(\overline{VMA} + \overline{A15} + \overline{A14} + \overline{A13})} = \overline{VMA.\overline{A15.A14.A13} + VMA.\overline{VMA}}$

or $VMA.\overline{VMA} = 0$ donc $f = \overline{VMA\ (\overline{A15} + \overline{A14} + \overline{A13})}$.

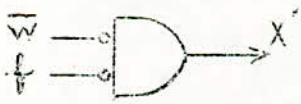
- dans U3 :

pour une lecture , $R/\overline{W} = R =$ " 1 "

$X = R.\overline{f} = R.\ VMA\ (\overline{A15} + \overline{A14} + \overline{A13})$

Ainsi si l'une des lignes A15 , A14 ou A13 est au niveau logique "0" , le DATA BUFFER se trouve validé en entrée pour une lecture du bus de données .

pour une écriture , $R/\overline{W} = \overline{W} =$ "0"



$X' = W.VMA\ (\overline{A15} + \overline{A14} + \overline{A13})$

La fonction X' valide le DATA BUFFER en sortie pour permettre une écriture sur le bus de données à condition que l'une des lignes A15, A14 ou A13 soit à "0" .

C - Les positions d'adresse "miroirs" :

Dans ces positions, on retrouve les contenus de certaines parties de la ROM , de la RAM et des registres PIA .

Prenons l'exemple de la zône "miroir" située de $FC00 à $FFFF .

Cette zône correspond à la zône mémoire $F400 ÷ $F7FF de la ROM "D5BUG" . Voyons comment cette ROM est adressée :

VMA	R/$\overline{W}$	A15	A14	A13	A12	A11	A10	A9	A8	A7	A6	A5	A4	A3	A2	A1	A0
1	X	1	1	1	1	*	.	.	.	.	.	.	.	.	.	.	.

. peut prendre les valeurs 1 ou 0 .

* prend 0 ou 1 suivant les connections de J_2 .

Connection 6-5	F	0	0	0 à F	3	F	F
1erKoctets de la ROM"D5BUG"	1111	0000	0000	0000	1111	0011	1111	1111
Connection 8-7	F	4	0	0	F	7	F	F
2^{e}K octets de la ROM"D5BUG"	1111	0100	0000	0000	1111	0111	1111	1111
Connection 2-1	F	C	0	0 à F	F	F	F
pour adressage"miroir"	1111	1100	0000	0000	1111	1111	1111	1111

Les connections 6-5 et 8-7 ont pour rôle de valider toutes les adresses de la ROM "D5BUG" .

La ligne d'adresse A11 peut prendre soit la valeur "0" soit la valeur "1" .

Par conséquent , lorsque le MPU adresse la zône "miroir"indiqué ci dessus , il adresse en réalité la zône mémoire correspondante de la ROM "D5BUG" .

On exploite cet aspect d'effet miroir pour prendre en compte les vecteurs d'interruption et d'initialisation .Etudions ce dernier vecteur : situé à FFFE - FFFF ,il correspond en fait sur la"D5BUG" aux positions F7FE - F7FF . Celles-ci contiennent l'adresse du programme d'initialisation $F000.

Il en est de même pour les vecteurs d'interruption :

$\overline{NMI}$: (FFFC) = (F7FC) = $F773

SWI : (FFFA) = (F7FA) = $F7D2

$\overline{IRQ}$: (FFF8) = (F7F8) = $F7EC

D- Tableau d'adressage des boîtiers :

boîtier	Description
ROM D5BUG (U12)	$ F000 -F7FF dont une partie($F400-F7FF)apparaît aux adresses $FC00-FFFF . Ce boîtier est validé par le connecteur J2 .
User ROM (U13)	On peut changer le type de ROM grâce au strap J4 Ce boîtier est contrôlé par le connecteur J3.
ScratchRAM (U11)	$E400-E47F .Son champ d'adresses est divisé en 3 sections : . $E400-E418 réservée comme pile pour l'utilisateur . $E419-E463 utilisée par le moniteur D5BUG (tableau Scratch RAM) . $E464-E47F pile du moniteur D5BUG .
User RAM (RAM interne du 6802),	$0000 - 007F . Elle peut être supprimée par le strap E1 lorsque le buffer d'expansion U3 est installé .
RAM optionnelle (U7,U8)	$ E000 -E3FF .
System PIA	$ E484-E487. $E484:registre de données du port A $E485:registre de contrôle du port A $E486:registre de données du port B $E487:registre de contrôle du port B.

.../...

	l'adressage de ce PIA est : 1110 01XX 1XXX X1.. X peut prendre la valeur 0 ou 1 ce qui fait apparaitre 64 images différentes(2^6) avec la $1^{ére}$ image à $E484-E487 et la dernière à $ E7FC - $E7FF.
User PIA (U9)	$E480-E483 . $E480:registre de données du port A $E481:registre de contrôle du port A $E482:registre de données du port B $E483:registre de contrôle du port B. Comme pour le "System PIA",le décodage d'adresses est responsable de l'effet "miroir"apparaissant pour les adresses :1110 01XX 1XXX X0 .. La première image est à $E480 -E483 . La dernière est à $ E7F8 - E7FB.
ACIA (U33) optionnel-non utilisé avec le clavier.	$E700 - E701 . $E700:registre d'état/contrôle. $E701:registre de données son adressage est: 1110 0111 0XXX XXX. Cet adressage fait apparaitre 64 images avec la $1^{ére}$ image à $E700,E701 et la dernière image à $ E77E , E77F
Adressage externe	$0000 - $DFFF .Ces 56 K adresses sont destinées pour l'expansion du KIT .

III Fonctionnement du kit D5 :

III-1 Description du moniteur :

Le fonctionnement de la carte microordinateur D5 est contrôlé à partir du clavier en conjonction avec le moniteur D5BUG .

III-1-1 Description des fonctions :

Le clavier dispose de 16 touches hexadécimales allant de 0 à F et de 8 touches de fonction en ne comptant pas la touche "RS" d'initialisation de tout le système.Examinons la fonction de chacune de ces touches .

a) Escape (Ex) : Cette touche permet de revenir sous le contrôle du moniteur . Si le système exécute un "User program" et que l'on active la fonction Escape, on aura l'affichage de l'état du "User-program counter" UPC et le système sera sous le contrôle de la routine de visualisation des registres "RD" .

Par contre,si le système est sous le contrôle de D5BUG lorsque la touche Ex est appuyée,on a la visualisation du symbole de la routine PROMPT (" — ") sur l'afficheur de gauche .

b) Memory Display / Change (MD) :

Cette touche permet d'examiner et de changer le contenu des cases mémoires .

c) Offset Calculation (FS) :

C'est une commande de calcul du déplacement pour une instruction de branchement utilisant le mode d'adressage relatif. Si l'offset calculé sort de la plage autorisée (+128 à - 127) on a l'affichage de "bAd".

L'offset calculé peut être placé automatiquement à l'emplacement nécessaire dans le programme.

d) Register Display / Alter (RD) :

Cette commande permet de visualiser et de changer le contenu des registres du MPU en commençant par le UPC.

e) Trace Single Step (T/B):

La fonction T/B permet l'exécution d'un programme en pas à pas soit à partir d'un point d'arrêt,soit à partir d'une adresse spécifiée dans le PC.

f) Breakpoints (T/B):

C'est la fonction de pose des points d'arrêt. Elle est trés utile lorsqu'on doit mettre au point un programme.L'utilisateur exécute une partie de son programme puis vérifie les résultats avant de lancer la partie suivante .

On peut mettre jusqu'à 5 points d'arrêt. Les breakpoints ne peuvent être mis que dans une RAM .

Pour supprimer un point d'arrêt,un utilise la touche "FC" .

g) Go to User Program (Go):

Une commande Go permet de lancer le programme à l'adresse désirée . Cette commande utilisée sans préciser d'adresse lance l'exécution à partir de la valeur contenue dans le PC .

Ceci permet par ex.de repartir d'un point d'arrêt .

h) Punch (P/L):

Cette fonction donne à l'utilisateur la possibilité de sauvegarder des zônes mémoires sur des bandes magnétiques en utilisant un magnétophone à cassette ordinaire . La commande demande les adresses de début et de fin de la zône à sauvegarder.

i) Load/Verify (P/L):

C'est la commande de chargement en mémoire d'un programme placé sur une cassette .

Cette commande est assortie d'une possibilité de vérification d'un programme enregistré . En cas de détection d'une erreur,on a l'affichage de "FaiL ??" .

La commande de Load est "FS" suivi de "P/L" tandis que celle de Verify est "FS" - "RD" .

j) User Functions ou fonctions de l'utilisateur :

L'utilisateur a la possibilité d'ajouter 16 fonctions sur celles existant déjà . Les adresses de ces fonctions sont rangées dans une table . Un pointeur (FNCPNT) est prévu pour le stockage de l'adresse de la table .

Une fonction spéciale sera alors exécutée en appuyant sur la touche FS puis sur la touche hexadécimale correspondante .

III-1-2 Description des subroutines (voir fig 5) :

Ce sont des routines de service qui gèrent totalement le fonctionnement du KiT . Néanmoins,on peut les utiliser comme sous programme pour une application particulière du KiT . Le Tableau de la fig 5 nous indique les principales subroutines existant dans la ROM D5 BUG .

a) RESET (ØF000) : Cette routine est appelée lorsqu'on appuie sur la touche "RS" du clavier d'où son appelation "RESET hardware ".

Description :
- pose du masque d'interruption
- effacement de la "scratch RAM" , du "system PIA" et du "user PIA" .
- initialisation du "system PIA"
- initialisation du "User Stack Pointer" USP à ØE418 .

fig 5 -

D5BUG USEFUL PROGRAM ADDRESSES
(Also Refer To Complete D5BUG Listing)

Name of Routine	ADDR in D5BUG	Description
RESET	$F000	Cold Restart
PROMPT	$F024	Warm Start
GET	$F04E	Routine to Read a Key (Uses NMI routine so a user would not normally use this routine directly.
PUT	$F0BB	Display Data on 7-segment readouts and calls functioning subroutines.
DYSCOD	$F120	Decode Hex to 7-segment.
DLY25	$F169	Delay 25 milliseconds.
DLY1	$F171	Delay 1 millisecond.
DLYX	$F179	Delay based on X-Reg (about 1/2 second Max).
ADDAX	$F183	Add A-Register to X-Register.
CLRDS	$F195	Clear display digits per A-Register mask.
ROLL2	$F1AA	Numeric entry routine for 2-digit values (one byte). Address being operated on specified in "HEXBUF".
ROLL4	$F1CC	Numeric entry routine for 4-digit values (2-Bytes).
RDKEY	$F1EF	Read and acknowledge key.
TIN*	$F533	Read 1 byte from tape.
PNCHB*	$F630	Format and punch a whole flie.
LOAD*	$F69C	Load a whole file (if "FNCFL" $\neq$ 0). Verify a whole file (if "FNCFL" = 0).

*Tape routines have critically tuned execution times. Refer to detailed explanations.

b) PROMPT (ØF024) : constitue un RESET software .

Description : - chargement du SP à ØE47E .

-établissement des indicateurs à leur valeur initiale :

ROLPAS =1 ; UPROG=0; ROIFLG=0 ;KYFLG=0; FNCFL=0 .

- établit la routine de sélection des fonctions FUNSEL comme programme principal .
- branchement à la routine PUT d'affichage et d'éxécution du programme principal .

c) GET (ØF04E) : subroutine de lecture et de décodage des touches .

Description : les touches du clavier sont arrangées électriquement en 6 lignes et 4 colonnes, la touche RS mise à part .

GET est appelée lors de l'enfoncement d'une touche .

Pendant qu'une touche du clavier est enfoncée , GET recherche la colonne puis la ligne correspondantes . L'information ligne/Colonne est alors utilisée pour adresser une table des valeurs des codes (voir listing - KYTBL). Le code correspondant à la touche enfoncée est stocké dans la RAM location KEY et reste disponible dans l'accumulateur A .

L'indicateur KYFLG est mis à 1 (Ø01) pour indiquer l'enfoncement d'une touche .

Normalement GET est mise en service à l'aide de $\overline{NMI}$ mais on peut supprimer le signal de sortie d'interruption du"system PIA" et travailler en mode POLLING .

d) PUT (ØFOBB) : programme d'affichage de l'information sur les afficheurs 7-segments et d'éxecution du programme principal comme subroutine une fois chaque milliseconde .

Description : cette routine est trés importante car elle permet d'afficher l'information continuellement . L'information(le code 7 segments du digit à afficher) est présentée sur les lignes parallèles de l'anode . La cathode,quant à celle,est chargée de valider les segments de l'afficheur approprié . Un digit s'affiche pendant 1 ms , puis on éteint l'afficheur de façon à passer au prochain digit (on passe par la même accasion au prochain afficheur). Pendant la courte période entre l'affichage de 2 digits , une subroutine dont l'adresse est stockée dans la location MNPTR de la scratch RAM est exécutée .

Les digits s'affichent donc l'un aprés l'autre .Aprés le 6ème digit , l'affichage reprend du début pour un rafraichissement de tous les digits .

On s'aperçoit donc que PUT alterne entre le rafraichissement de la visualisation et l'exécution d'un certain programme .

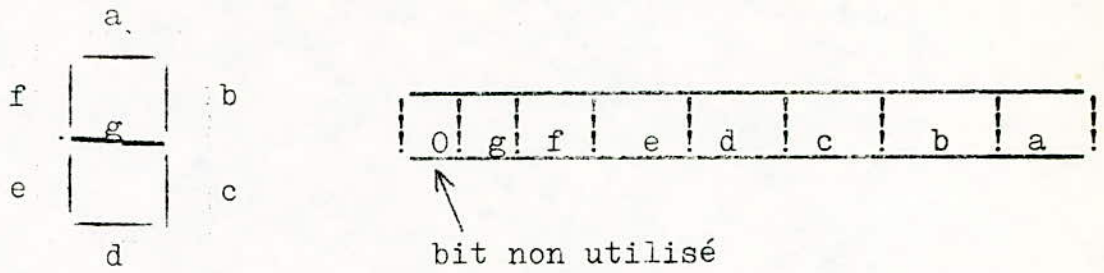
Cependant, la fréquence de rafraichissement des digits est suffisamment élevée pour que la visualisation semble continue .

La routine PUT obtient l'information sur les afficheurs à partir d'un buffer 6 bytes appelé DISBUF . Ce buffer contient le code 7 segments des digits à afficher,le premier byte DISBUF correspondant à l'afficheur le plus à gauche et le sixième byte DISBUF + 5 à l'afficheur le plus à droite .

On valide chacun des afficheurs à partir de l'accumulateur B.

Cette validation se fait en mettant un "1" logique dans le bit qui correspond à l'afficheur . Le bit 5 de l'acc.B, corresponds à l'afficheur le plus à gauche ; le bit 0 à l'afficheur de droite.

Le code 7 - segments



Correspondance DISBUF - Afficheur 7 segments

* un "1" logique dans un bit allume le segment correspondant (à condition que la cathode K soit à 0) .

e) DYSCOD_ (ℰ F120) : Conversion à partir des HEXBUFS de données hexadécimales en code 7-segment et stockage dans les DISBUFS

Description : les 6 digits hexadécimaux contenus dans HEXBUF, HEXBUF+1, HEXBUF+2 sont chacun converti dans le code 7-segments correspondant . Ce dernier est ensuite stocké dans l'un des 6 bytes du buffer d'affichage "DISBUF" .

Les 4 bits les plus significatifs de la location HEX BUF correspondant à la location DISBUF et les 4 bits les moins significatifs de HEXBUF + 2 correspondant à DISBUF + 5 .

On peut alors faire appel à la routine PUT pour visualiser le contenu des 3 registres HEXBUF .

f) DLY _25,_DLY_1_,_DLY_X_ _(ℰF169)_(ℰF171)_(ℰF179)_:_

Ces routines sont des délais de 25 ms et de ms en ce qui concerne les 2 premières . DLY X est un délai variable basé sur le contenu du registre d'index IX .

Description : l'éxecution de DLY X donne une temporisation en accord avec la formule suivante :

(X - 1) (8 cyc) (1,11763 us/cycle) + (27,94 us)

Le délai minimum (27,94 us) est obtenu pour X = Ø 0001
Le délai maximum est de 0,5842 sec (X = ØFFFF).

g) ADDA_X_ (Ø F183) : C'est l'addition de l'accumulateur A au registre d'index IX. Le résultat se trouve dans IX .

h) CLRDS(ØF195): Effacement des afficheurs par un masque contenu dans l'acc A. Description : Chacun des 6 bits de poids faible de l'acc A correspond à un afficheur avec le bit 0 pour l'afficheur le plus à droite.

Pour chaque bit se trouvant à "1" , l'afficheur correspondant se trouve éteint .

i) RDKEY _(_Ø_F1EF)_: Lecture du code de la touche enfoncée

Description : RDKEY est appelée lorsque KYFLG est à 1 . Cette routine transfére le code de la touche enfoncée à partir de KEY dans l'acc A et réinitialise KYFLG à 0 pour s'acquitter de cette touche .

j) ROLL_2_ (Ø F1AA)_: Routine d'entrée de 2 digits

Description : l'adresse de la position mémoire sur laquelle opére Roll 2 est spécifiée dans HEXBUF et HEXBUF + 1 . Un nouveau digit est placé par Roll 2 dans les 4 bits les moins significatifs de l'acc A . Ce digit sera déplacé vers la gauche lorsqu'on rentre un autre digit. Le digit qui se trouvait dans les 4 bits les plus significatifs de l'acc A est perdu. L'indicateur ROLPAS est à 1 avant l'entrée des digits .

Aprés le premier digit rentré, ROLPAS passe à 0 .

k) ROLL_4_(_Ø_F1 CC_)_: Routine d'entrée de 4 digits .

Description : ROLL 4 opére le décalage des digits dans HEXBUF_ HEX BUF + 1 . Un nouveau digit est placé par ROLL 4 dans l'acc A partie LSB puis transféré dans HEXBUF + 1 (partie LSB) .

Pour ce premier digit, ROLPAS = 1 . Il passe à 0 pour la prise en compte des prochains digits .

Un digit est déplacé à partir des 4 bits les moins significatifs de HEXBUF + 1 vers les 4 bits de poids fort de HEXBUF lorsqu'on rentre 3 autres digits . Si on rentre un 5^{éme} digit, le 1er digit rentré sera perdu .

l) TIN (Ø F533) : Lecture d'un byte à partir d'une bande

Description : TIN est utilisé pour lire un caractère de format standard KANSAS CITY à partir de la bande d'une cassette.

Le format du caractére est :

a) "0" logique = 4 cycles de 1200 HZ

b) "1" logique = 8 cycles de 2400 HZ.

c) Un caractére est constitué de :

- un " 0" logique comme bit "Start"
- 8 bits représentant la donnée proprement dite .
- au moins deux "1" logique comme bits "Stop" .

Des techniques de timing software et un algorithme de tolérance d'erreur sont employés pour assurer une bonne lecture des données. A la vitesse de 300 Bauds , les bits "Stop" prennent 13,3 ms . Durant cette période, le précédent caractére est mis sous le contrôle d'une autre routine. Lorsqu'il s'agit de lire une suite de caractères, la routine qui contrôle le caractère lu doit rappeler TIN au moins un cycle avant le prochain bit "Start" . Ceci permet une bonne synchronisation .

m) PNCHB (ØF608) : Envoi d'un byte dans l'interface cassette .

Description : voir TIN pour le format des caractéres .

Lorsqu'on transmet une série de bytes , il [illegible] , il est important de maintenir une synchronisation entre bytes successifs(et pas seulement au niveau du caractére).

PNCHB prend exactement 35 cycles MPU à partir du dernier cycle de l'instruction d'appel de PNCHB jusqu'au premier bit "Start" envoyé La synchronisation est maintenue automatiquement par PNCHB pendant la transmission du caractére .

Aprés la transmission du dernier bit "Stop" , on a le retour de PNCHB . La fin du dernier bit "Stop" d'un caractére marque aussi le début du bit "Start" du prochain caractére .

Le temps mis à partir du dernier cycle de retour de PNCHB jusqu'au prochain bit "Start" à transmettre est exactement 159 cycles MPU .

Entre le retour de PNCHB et le prochain appel de PNCHB on a 159-35 = 124 cycles qui doivent être occupés par un programme . Par exemple,si 50 cycles sont nécessaires pour établir le prochain caractère, le reste (74 cycles) doit être utilisé en temps mort avant de rappeler PNCHB .

n) PUNCH _ (ØF630) : enregistrement d'une file de caractéres sur la cassette .

Description : Avant d'appeler PUNCH , il faut stocker l'adresse de la première donnée à enregistrer dans la location BEGAD (ØE460,1) et l'adresse de la dernière donnée dans ENDAD (ØE462,3) .

Le format de la file est le suivant :

a) 30 secondes du caractére ØFF comme leader .
b) bloc "Start" (Ø53) .
c) adresse de commencement de la file ,octet de poids fort.
d) adresse de commencement de la file,octet de poids faible
e) adresse de fin de la file,octet de poids fort .
f) adresse de fin de la file,octet de poids faible
g) Données binaires commençant à l'adresse de début jusqu'à l'adresse de fin .

h) Un byte checksum . Checksum est la somme complémentée à 2 de tous les bytes de données en plus des adresses de début et de fin.Checksum sert de détecteur d'erreur .

p) LOAD (ØF69C): Chargement ou vérification de la file envoyée de la cassette vers la mémoire .

Description : La routine LOAD est utilisée pour lire une file de caractéres à partir de la cassette ou vérifier l'enregistrement vis à vis du contenu de la mémoire. Si le bit le moins significatif de l'indicateur FNCFL (ØE43E) est mis à 1 , un chargement s'effectue; sinon il s'agit d'une vérification .

III.2 Fonctionnement du KIT D5 :

Pour établir le fonctionnement du KIT ,il a fallu étudier les principaux listings , construire les organigrammes correspondants puis faire le rapprochement entre ces différents organigrammes pour aboutir à un schéma général de fonctionnement .

III-2.1 Principe_général_de_fonctionnement_(fig 7) :

Pour comprendre le fonctionnement du système,il faut se reporter aux organigrammes , plus particulièrement à l'organigramme général.

Le moniteur utilise des cases mémoires de la Scratch RAM pour réaliser un stockage temporaire de données nécessaires au bon fonctionnement du KIT(voir fig 6 tableau de la SCRATCH RAM).Décrivons le fonctionnement général du KIT (voir fig 7) :

Lorsqu'on appuie sur la touche "RESET" du clavier,le moniteur décrit une séquence d'initialisation dans laquelle l'adresse de la routine de sélection des fonctions (FUNSEL : Function Select Routine est stockée dans MNPTR . A la fin de cette séquence,on visualise " _ " pour indiquer que le système est sous le contrôle du moniteur.

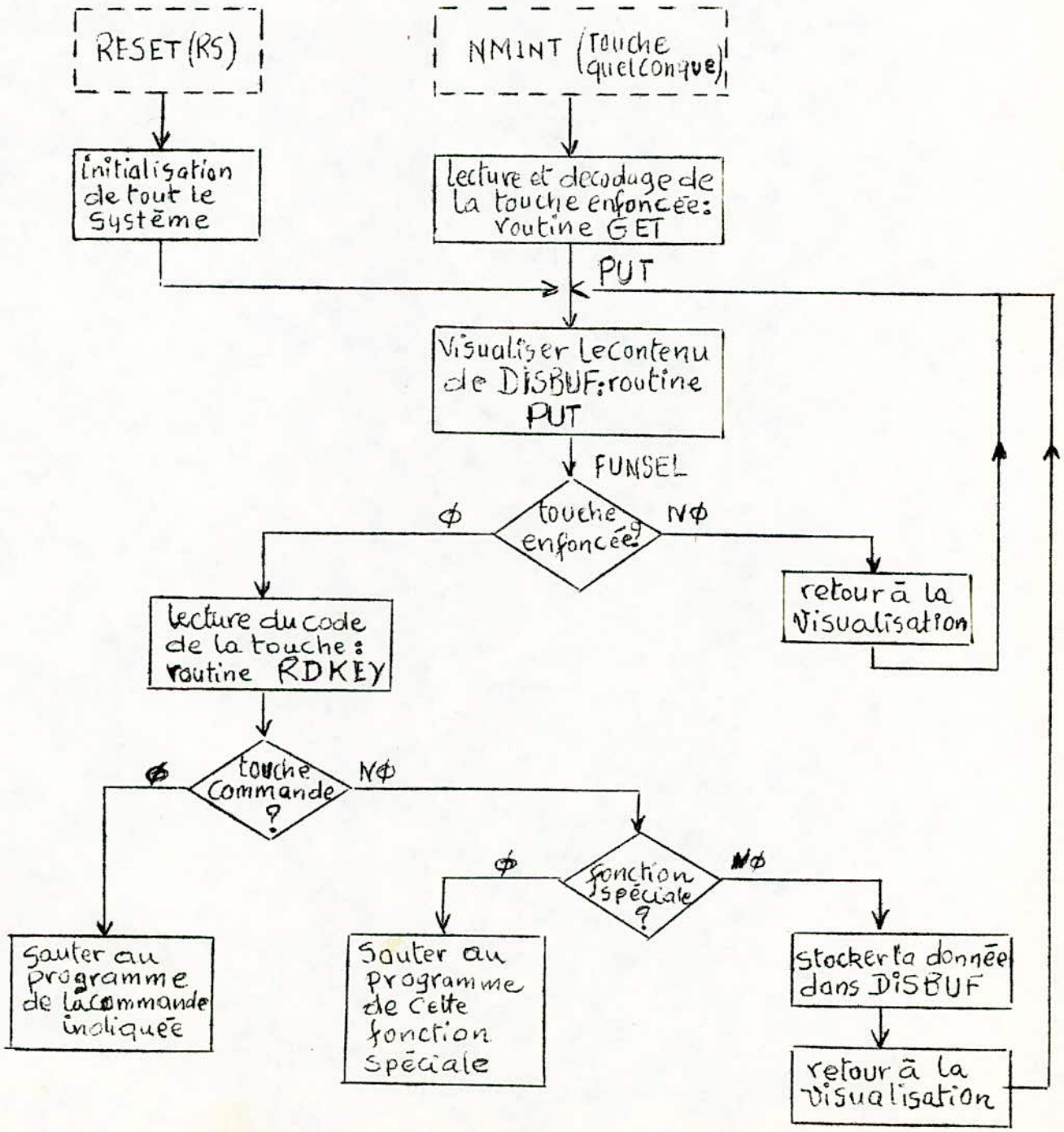
fig 6 -

SUMMARY OF MOST USED D5BUG SCRATCH RAM LOCATIONS
(Also See Complete D5BUG Listing)

Name of Routine	ADDR in D5BUG	Description
MNPTR	$E419,A	Pointer to active sub-program.
KEY	$E41B	Entered key code from keypad.
KYFLG	$E41C	Flag to indicate a key is pending.
DISBUF	$E41D-E422	(6) bytes correspond to (6) 7-segment displays. Contains 7-segment codes.
ROLPAS	$E423	Flag to indicate first digit entry.
HEXBUF	$E42C,D,E	3-byte buffer for hexadecimal information. Each byte corresponds to (2) 7-segment display digits.
USP	$E42F,30	User stack pointer pseudo-register.
UCC	$E431	User condition codes pseudo-register.
UB	$E432	User B-register pseudo-register.
UA	$E433	User A-register pseudo-register.
UX	$E434,5	User X-register pseudo-register.
UPC	$E436,7	User Program Counter pseudo-register.
UIRQV	$E43C,D	Points to user's IRQ service routine.
FNCFL	$E43E	Flag to indicate special (or alternate) function.
FNCPNT	$E43F,40	Points to ADDR of user's special function table.
BYTE	$E459	Data byte read from cassette or to be punched to cassette.
BEGAD	$E460,1	Beginning Address (for punch).
ENDAD	$E462,3	Ending Address (for punch).

*Tape routines have critically tuned execution times. Refer to detailed explanations.

fig 7- Organigramme général de fonctionnement du KIT



Ainsi , aprés la séquence d'initialisation,c'est la séquence d'affichage "PUT" qui est exécutée. PUT visualise le premier DISBUF(Ø E41D) pendant 1 ms. Elle éteint ensuite tous les segments ,valide toutes les lignes du clavier (en mettant les cathodes K à 0) avant de passer à FUNSEL.

Il faut préciser ici que l'appui sur les autres touches autres que "RS" aboutit aussi à la routine PUT mais en ayant éxecuté la routine GET de lecture et de décodage de la touche enfoncée .
Au niveau de FUNSEL , tous les indicateurs adéquats sont déjà positionnés .

Un test a lieu sur KYFLG pour savoir si on a appuyé sur une touche. Si le test est positif, le code de la touche est lu par la routine RDKEY . Ce dernier nous renseigne sur la nature de la touche:

N = 1 (N=bit 7 du code) correspond à une touche de fonction.

N= 0 correspond à une touche hexadécimale .

Si N= 1 FUNSEL passe le contrôle à la fonction demandée

Si N= 0 Un test sur FNCFL a lieu pour déterminer s'il s'agit simplement de l'entrée d'un digit ou bien d'une fonction spéciale.Si on se trouve dans le premier cas,les routines ROLL 4 de prise en comp-te de 4 digits et DYSCOD de conversion de l'hexadécimal en code 7-segment sont activées avant de revenir finalement à PUT et visualiser ainsi le digit rentré.

Une fonction spéciale est traitée sur le même principe qu'une fonction du système .

Une fois que la fonction demandée est exécutée,on revient à la routine PUT pour visualiser les résultats .

```
0131                    ********************************************************************
0132                    *
0133                    * PUT - DISPLAYS DATA IN DISBUF & CALLS THE
0134                    *       FUNCTIONING SUBROUTINE
0135                    *
0136                    ********************************************************************
0137A F0BB C6 20     A PUT     LDAB   #%00100000 INIT DIG ENABLE PATTERN
0138A F0BD CE E41A   A LP1P    LDX    #DISBUF-3 POINT AT DISPLAY BUFFER
0139A F0C0 17                  TBA               MAKE EXTRA COPY
0140A F0C1 08          LP2P    INX               POINT AT NXT DIGIT INFO
0141A F0C2 48                  ASLA              ADD 1 TO 'X' FOR EACH SHIFT
0142A F0C3 24 FC F0C1          BCC    LP2P       LOOP DEVELOPS DIGIT INFO ADDR
0143A F0C5 A6 00     A         LDAA   ,X         GET SEG INFO
0144A F0C7 43                  COMA              ANODE DRIVERS ARE GND TRUE
0145A F0C8 B7 E484   A         STAA   ANOD       STORE ANODE INFO TO PIA
0146A F0CB F7 E486   A         STAB   CATH       ENABLE DIGIT CATHODE
0147A F0CE BD F171   A         JSR    DLY1       ON FOR 1 MILLISECOND
0148A F0D1 86 FF     A         LDAA   #%11111111 1'S TURN OFF SEGS
0149A F0D3 B7 E484   A         STAA   ANOD       TURN OFF ALL SEGS
0150A F0D6 B7 E486   A         STAA   CATH       ENABLE ALL KPD ROWS
0151A F0D9 37                  PSHB              HAS ROTATING DIGIT ENABLE
0152A F0DA FE E419   A         LDX    MNPTR      GET ADDRESS OF ACTIVE MAIN PROG
0153A F0DD AD 00     A         JSR    ,X         EXECUTE IT
0154                    ****
0155                    **** SEE MANUAL
0156                    ****
0157A F0DF 33                  PULB              RECOVER DIGIT ENABLE
0158A F0E0 54                  LSRB              NEXT DIGIT
0159A F0E1 26 DA F0BD          BNE    LP1P       NOT THRU WHOLE CYCLE
0160A F0E3 20 D6 F0BB          BRA    PUT        PAST LAST DIGIT
0161                    *
```

```
0164                    ****************************************************************
0165                    *
0166                    * FUNSEL - ROUTINE TO SELECT A FUNCTION FROM A KEY INPUT
0167                    *
0168                    ****************************************************************
0169A F0E5 7D E41C  A FUNSEL TST     KYFLG     KEY PENDING ?
0170A F0E8 26 01 F0EB        BNE     KEYNOW    YES,TEST IT
0171A F0EA 39                RTS               ** RETURN ** NO KEY PENDING
0172                    *
0173A F0EB BD F1EF  A KEYNOW JSR     RDKEY     GET & ACKNOWLEDGE KEY
0174A F0EE 2B 15 F105        BMI     FUNKY     IF FUNCTION KEY
0175A F0F0 7D E43E  A        TST     FNCFL
0176A F0F3 26 0B F100        BNE     UFNK
0177A F0F5 BD F1CC  A        JSR     ROLL4     # ENTRY SO ROLL IT IN
0178A F0F8 BD F120  A        JSR     DYSCOD    CONVERT TO 7-SEG
0179A F0FB 86 03    A        LDAA    #%00000011
0180A F0FD 7E F195  A        JMP     CLRDS     BLANK LAST 2 DIGITS
0181                    *
0182A F100 FE E43F  A UFNK   LDX     FNCPNT    POINT AT USER FUNCTION TABLE
0183A F103 20 03 F108        BRA     HASH      .
0184                    *
0185A F105 CE F110  A FUNKY  LDX     #SYSFNC   POINT AT SYSTEM FUNCTION TBL
0186A F108 48          HASH  ASLA              2 BYTES PER ENTRY
0187A F109 BD F183  A        JSR     ADDAX     DEVELOP POINTER
0188A F10C EE 00    A        LDX     ,X        GET JMP ADDR
0189A F10E 6E 00    A        JMP     ,X        ** GO THERE **
0190                    *
0191                    *
0192A F110     F1F6  A SYSFNC FDB     MEMBEG    'MD'
0193A F112     F024  A        FDB     PROMPT    'EX'
0194A F114     F2CA  A        FDB     REGBEG    'RD'
0195A F116     F6F3  A        FDB     GO        'GO'
0196A F118     F4C5  A        FDB     FSET      'FS'
0197A F11A     F4D1  A        FDB     FCLR      'FC'
0198A F11C     F4D7  A        FDB     TAPBEG    'P/L'
0199A F11E     F388  A        FDB     BRKBEG    'T/B'
0200                    *
0201
```

III -2.2 Programme de mise en route et d'initialisation (fig 8 et 9, Routine $\overline{\text{RESET}}$) :

Ce programme initialise tout le système . Il inhibe d'abord les interruptions $\overline{\text{IRQ}}$ puis il efface la "Scratch RAM ",le "User PIA" et le "Systeme PIA" . Il programme alors le "System PIA" : les lignes PA0-PA6 sont en sortie , PA7 est en entrée . Le port B est programmé en sortie . Le mot de commande est le même pour les 2 ports : $ 06 ou % 0000 0110 .

Les lignes de commande CB1 et CB2 constituent des entrées d'interruption $\overline{\text{NMI}}$. La ligne CA1 est en entrée , elle est utilisée par l'interface cassette .

La ligne CA2 est en entrée, elle sera mise en sortie pour commander l'entrée CLEAR du TRACE TIMER .

La validation des interruptions $\overline{\text{NMI}}$ est faite plus loin par la subroutine ENNMI . Elle ne concerne que la ligne de commande CB1 . ENNMI permet aussi de valider toutes les lignes du clavier.

Les "User" et"System" stack pointers sont initialisés respectivement à $E418 et $E47E . Les indicateurs sont mis à leur valeur initiale .

On établit FUNSEL comme routine principale puis on active la routine PUT d'affichage et d'exécution de la routine principale.

Le symbole " _ " de PROMPT est visualisé indiquant que la phase d'initialisation est terminée .

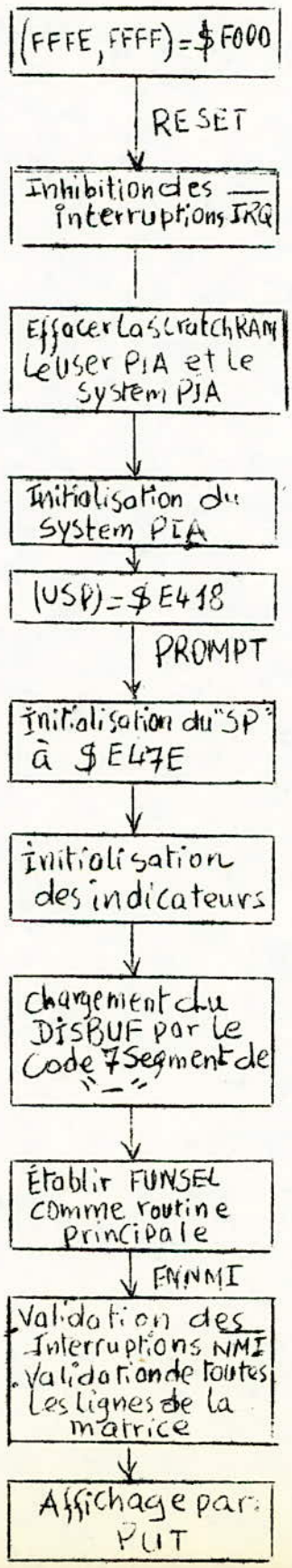
III-2-3 Programme de lecture et de décodage d'une touche du clavier (fig 10 et 11 ,Routine GET) :

La routine GET de lecture d'une touche est activée par le signal d'interruption $\overline{\text{NMI}}$. GET procéde à la recherche de la touche enfoncée en sélectionnant d'abord la colonne puis la ligne sur la matrice Le décodeur "MUX"le MC14539 de la fig 12 est utilisé pour cette

.../...

fig8 - RESET . routine d'initialisation de tout le système

Organigramme



```
0001                            NAM     RESET
0002                            OPT     CREF,LLEN=80
0003A F000                      ORG     $F000
0004                    ********************************************************************
0005                    *
0006                    * RESET - COLD START ROUTINE
0007                    *
0008                    ********************************************************************
0009A F000 01           RESET   NOP             SET INTERRUPT MASK
0010A F001 0F                   SEI             .
0011A F002 CE E3FF  A           LDX     #$E3FF  CLEAR RAM
0012A F005 08           CLRLOP  INX             .
0013A F006 6F 00    A           CLR     0,X     .
0014A F008 8C E487  A           CPX     #$E487  .
0015A F00B 26 F8 F005           BNE     CLRLOP  .
0016A F00D CE E484  A           LDX     #$E484  INITIALIZE SYSTEM PIA
0017A F010 86 7F    A           LDAA    #$7F    .
0018A F012 A7 00    A           STAA    0,X     .
0019A F014 86 FF    A           LDAA    #$FF    .
0020A F016 A7 02    A           STAA    2,X     .
0021A F018 86 06    A           LDAA    #$06    .
0022A F01A A7 01    A           STAA    1,X     .
0023A F01C A7 03    A           STAA    3,X     .
0024A F01E CE E418  A           LDX     #$E418  DEFAULT USER STACK
0025A F021 FF E42F  A           STX     USP     .
0026                    ********************************************************
0027                    *
0028                    * PROMPT - ROUTINE TO SET UP PROMPT CONDITIONS
0029                    *
0030                    ********************************************************
0031A F024 8E E47E  A PROMPT LDS     #STKTOP INIT SYSTEM STACK
0032A F027 86 01    A           LDAA    #1      SET FIRST PASS
0033A F029 B7 E423  A           STAA    ROLPAS  .
0034A F02C 7F E43B  A           CLR     UPROG   INIT FLAGS
0035A F02F 7F E438  A           CLR     ROIFLG  .
0036A F032 7F E41C  A           CLR     KYFLG   .
0037A F035 7F E43E  A           CLR     FNCFL   .
0038A F038 86 40    A           LDAA    #$40    DISPLAY PROMPT
0039A F03A B7 E41D  A           STAA    DISBUF  .
0040A F03D 86 1F    A           LDAA    #%00011111 .
0041A F03F BD F195  A           JSR     CLRDS   .
0042A F042 CE F0E5  A           LDX     #FUNSEL EXECUTE FUNCTION SELECT
0043A F045 FF E419  A           STX     MNPTR   .
0044A F048 BD F7AE  A           JSR     ENNMI   ENABLE NMI
0045A F04B 7E F0BB  A           JMP     PUT     & GO
```

fig 9-

sélection .

GET utilise le principe suivant : elle envoie sur les lignes PB6 et PB7 (entrées A et B de "MUX") le numéro de colonne(00,01,10,-11) en commençant par la colonne 0 et à chaque fois,un test nous renseigne sur l'etat de la sortie Z ,sortie reliée à la ligne PA7 du "System PIA " .

- Remarquons que les colonnes de la matrice constituent les entrées X0,X1,X2 et X3 du décodeur .

Si PA7 =0 (Z=0) le n° de colonne envoyé sur PB6 -PB7 correspond à celui de la matrice .

On sauvegarde l'information colonne puis on passe à la sélection de la ligne; On envoie sur les lignes PB6-PB7 du port B l'information colonne trouvée et on commence par valider la 6ème ligne de la matrice (PB5=1) . On teste toujours PA7: PA7=0 la ligne recherchée est celle qui a été validée .

Si PA7 =1 on valide la ligne suivante .

On utilise alors l'information ligne/colonne pour trouver le code de la touche à partir de la table des valeurs des codes KYTBL.

Ce dernier est sauvegardé dans la case mémoire KEY.L'indicateur KYFLG est mis à 1 pour informer FUNSEL de l'entrée d'une touche .

Avant de pointer à la table des codes,on attend le moment où la touche n'est plus enfoncée puis on exécute un délai de 25ms.Ceci permet d'éviter les problèmes de rebondissement .

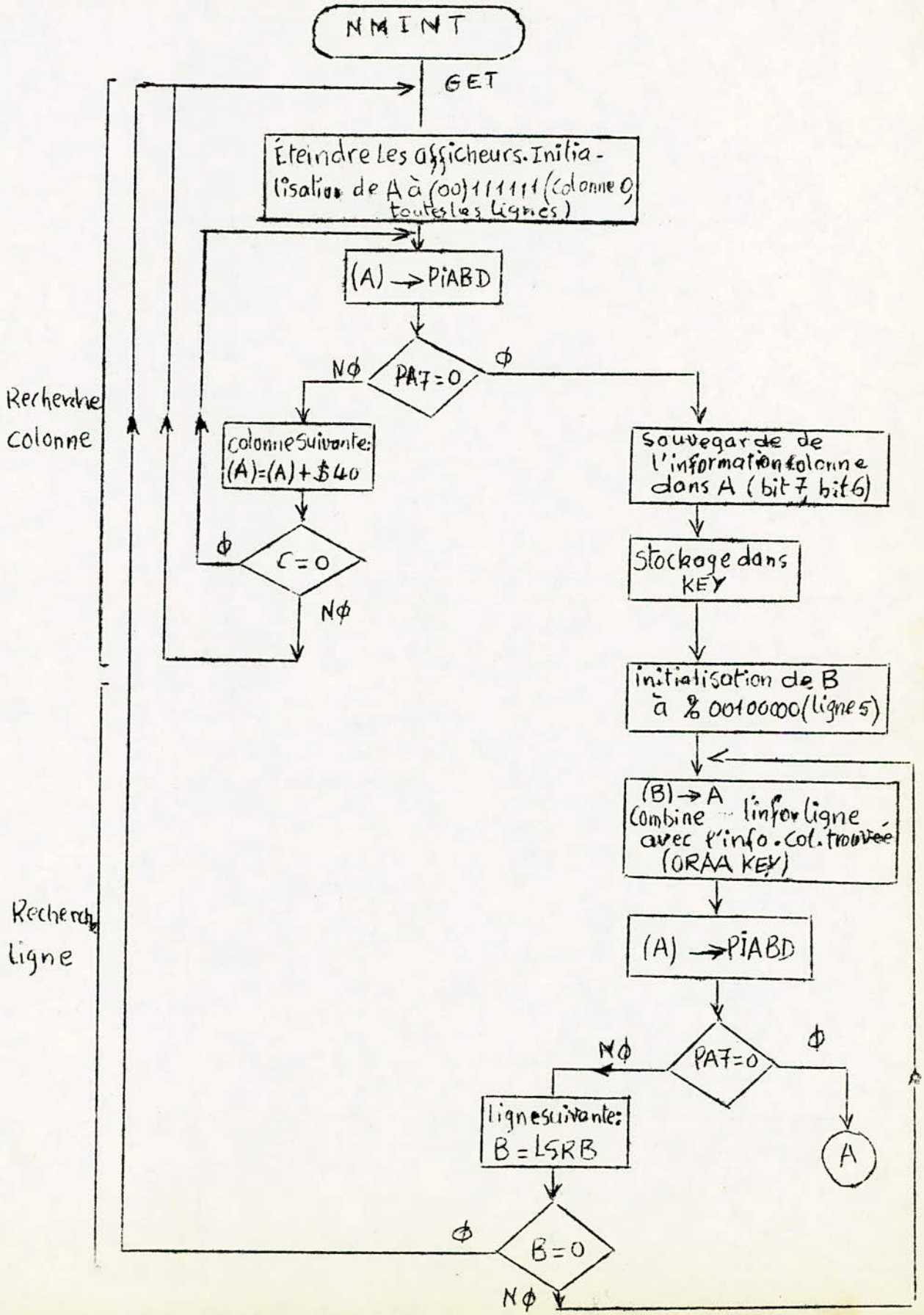
II-2-4 Traitement_des_interruptions_: (fig.13 et 14)

Le programme D5BUG utilise 3 types d'interruption :

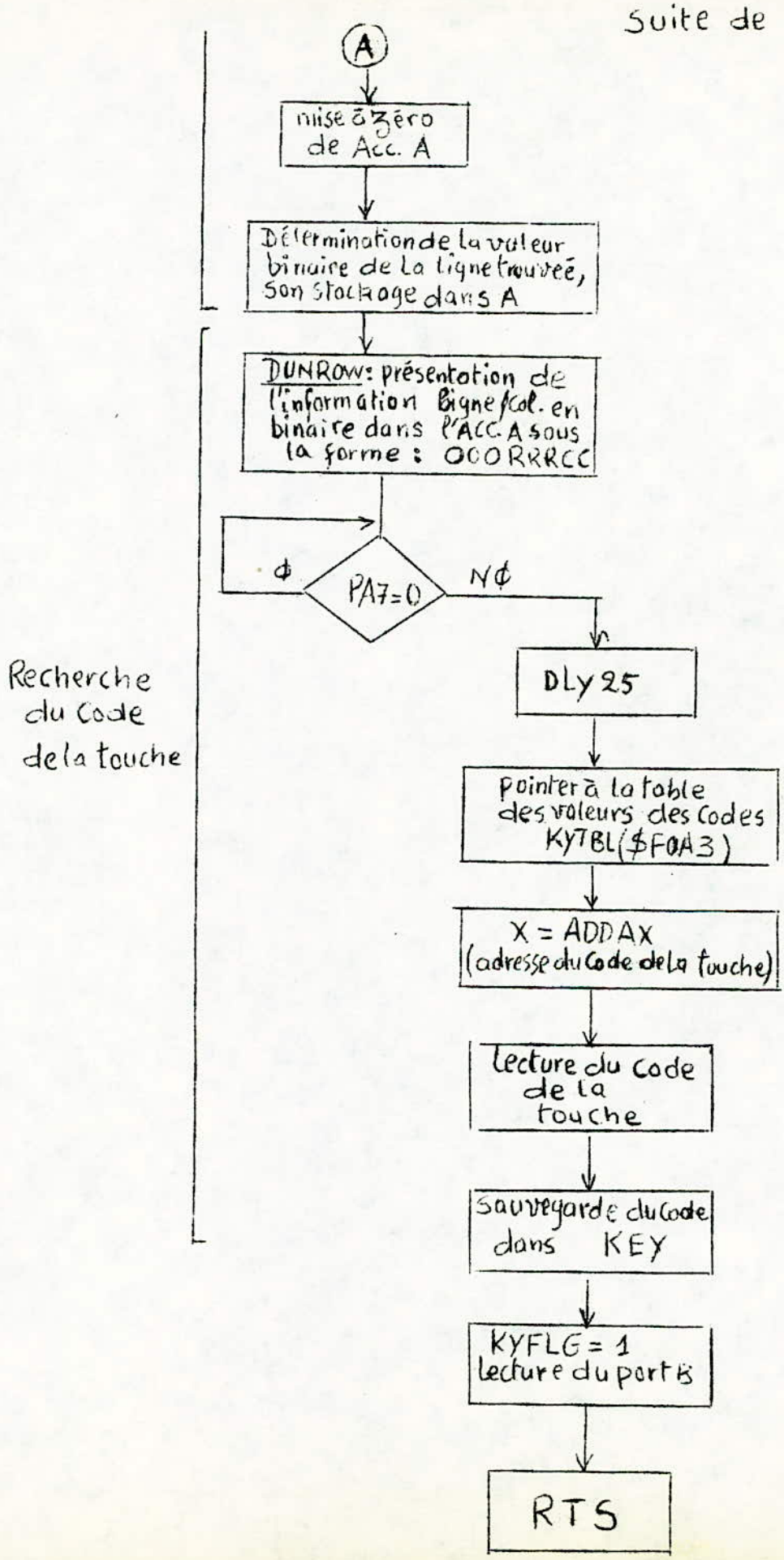
- SWI : interruption software
- $\overline{\text{IRQ}}$: interruption masquable
- $\overline{\text{NMI}}$: interruption non masquable.

fig 10 - GET - Routine de lecture et de décodage d'1 touche

Organigramme



Suite de GET


A
mise à zéro de Acc. A
Détermination de la valeur binaire de la ligne trouvée, son stockage dans A
DUNROW: présentation de l'information Ligne/col. en binaire dans l'ACC.A sous la forme : OOORRRCC
Ø
PA7=0
NØ
DLY 25
pointer à la table des valeurs des codes KYTBL($F0A3)
X = ADDAX (adresse du Code de la touche)
lecture du code de la touche
sauvegarde du Code dans KEY
KYFLG = 1 lecture du port B
RTS
Recherche du code de la touche

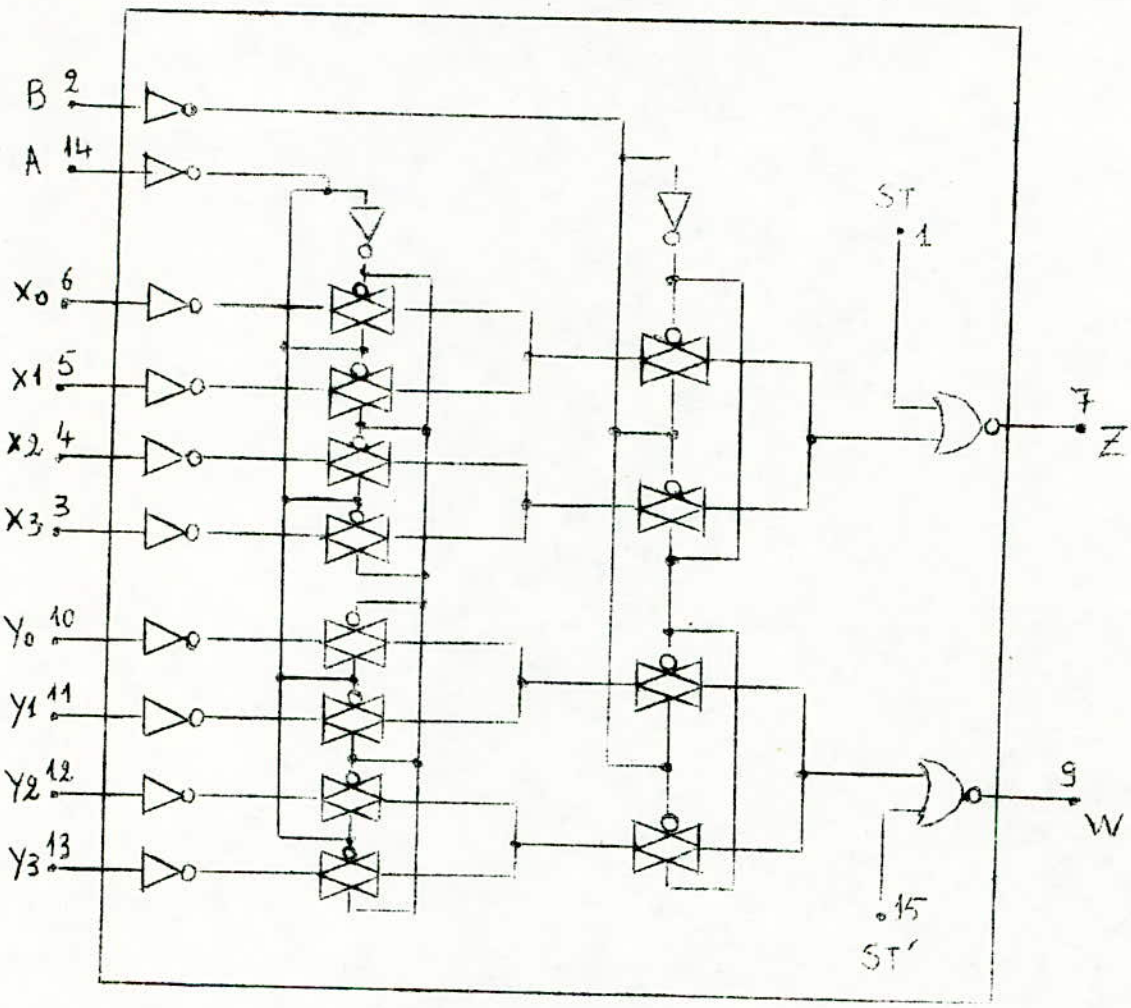
```
0048                    ********************************************************************
0049                    *
0050                    * GET - ROUTINE TO READ A KEY
0051                    *
0052                    ********************************************************************


0054A F04E CE E484  A GET     LDX    #PIA       POINT AT PIA
0055A F051 86 FF    A         LDAA   #$FF       .
0056A F053 A7 00    A         STAA   KPCOL,X    TO TURN OFF DISPLAYS
0057A F055 86 3F    A         LDAA   #%00111111 COL 0, ALL ROWS
0058A F057 A7 02    A LPCOL   STAA   KPROW,X    STORE INFO TO KEY MATRIX
0059A F059 6D 00    A         TST    KPCOL,X    MSB IS MUX BIT
0060A F05B 2A 06 F063         BPL    COLFND     BIT-7 LOW MEANS COL FOUND
0061A F05D 8B 40    A         ADDA   #$40       INC COL BITS TO MUX
0062A F05F 24 F6 F057         BCC    LPCOL      CONTINUE FOR ALL COLS
0063A F061 20 EB F04E         BRA    GET        KEY BOUNCED, START OVER
0064A F063 84 C0    A COLFND  ANDA   #%11000000 MASK TO SAVE ONLY COL
0065A F065 B7 E41B  A         STAA   KEY        WILL UPDATE LATER; JUST TEMP SAV
0066A F068 C6 20    A         LDAB   #%00100000 ROW 5
0067A F06A 17           LPROW TBA                COPY ROW INFO TO A-REG
0068A F06B BA E41B  A         ORAA   KEY        COMBINE WITH COL INFO
0069A F06E A7 02    A         STAA   KPROW,X    DRIVE KEY MATRIX
0070A F070 6D 00    A         TST    KPCOL,X    MSB LOW = CLOSURE
0071A F072 2A 05 F079         BPL    ROWFND
0072A F074 54                 LSRB              NEXT LOWER ROW BIT
0073A F075 26 F3 F06A         BNE    LPROW      LOOP TILL ALL ROWS TRIED
0074A F077 20 D5 F04E         BRA    GET        KEY BOUNCED; START OVER
0075A F079 4F           ROWFND CLRA             PREPARE TO FIND BINARY ROW #
0076A F07A 54           LPFND  LSRB             LOOP BUILDS BINARY ROW #
0077A F07B 25 03 F080         BCS    DUNROW     WHEN BIT FALLS OFF; A-REG HAS #
0078A F07D 4C                 INCA
0079A F07E 20 FA F07A         BRA    LPFND
0080A F080 79 E41B  A DUNROW  ROL    KEY
0081A F083 49                 ROLA
0082A F084 79 E41B  A         ROL    KEY
0083A F087 49                 ROLA              A-REG IS 000RRRCC
0084                    * A-REG NOW CONTAINS OFFSET FOR KEY LOOK-UP
0085A F088 6D 00    A CLOP    TST    KPCOL,X    SEE IF KEY STILL DOWN
0086A F08A 2A FC F088         BPL    CLOP       WAIT TILL LET UP
0087A F08C BD F169  A         JSR    DLY25      DELAY TO DEBOUNCE
0088A F08F CE F0A3  A         LDX    #KYTBL     POINT AT TOP OF TABLE
0089A F092 BD F183  A         JSR    ADDAX      CALC ADDR OF KEY CODE
0090A F095 A6 00    A         LDAA   ,X         GET KEY CODE
0091A F097 B7 E41B  A         STAA   KEY        SAVE KEY VALUE
0092A F09A C6 01    A         LDAB   #1         .
0093A F09C F7 E41C  A         STAB   KYFLG      INDICATE KEY PENDING
0094A F09F F6 E486  A         LDAB   PIAROW     TO CLEAR NMI
0095A F0A2 39           DIDDLE RTS              ** RETURN **
0096                    *
0097                    * THIS RTS IS USED AS A DO-NOTHING SUB
0098                    * SO SYST CAN BE DISABLED EXCEPT DISPLAY
0099                    *
```

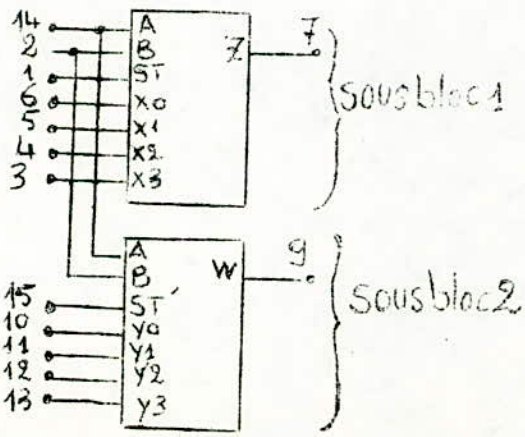
fig 11a

```
0101                    ********************************************************************
0102                    *
0103                    * KYTBL - KEY VALUE TABLE
0104                    *
0105                    ********************************************************************
0106A F0A3     00     A KYTBL  FCB     $00       '0' KEY
0107A F0A4     0F     A        FCB     $0F       'F'
0108A F0A5     0E     A        FCB     $0E       'E'
0109A F0A6     0D     A        FCB     $0D       'D'
0110A F0A7     01     A        FCB     $01       '1'
0111A F0A8     02     A        FCB     $02       '2'
0112A F0A9     03     A        FCB     $03       '3'
0113A F0AA     0C     A        FCB     $0C       'C'
0114A F0AB     04     A        FCB     $04       '4'
0115A F0AC     05     A        FCB     $05       '5'
0116A F0AD     06     A        FCB     $06       '6'
0117A F0AE     0B     A        FCB     $0B       'B'
0118A F0AF     07     A        FCB     $07       '7'
0119A F0B0     08     A        FCB     $08       '8'
0120A F0B1     09     A        FCB     $09       '9'
0121A F0B2     0A     A        FCB     $0A       'A'
0122A F0B3     84     A        FCB     $84       'FS' FUNCTION SET
0123A F0B4     85     A        FCB     $85       'FC' FUNCTION CLEAR
0124A F0B5     86     A        FCB     $86       'P/L' PUNCH/LOAD
0125A F0B6     87     A        FCB     $87       'T/B' TRACE/BREAK
0126A F0B7     80     A        FCB     $80       'MD' MEMORY DISPLAY
0127A F0B8     81     A        FCB     $81       'EX' ESCAPE
0128A F0B9     82     A        FCB     $82       'RD' REGISTER DISPLAY
0129A F0BA     83     A        FCB     $83       'GO' GO
```

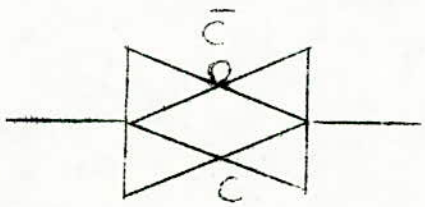
fig 11-6

fig 12 - le Décodeur ou Multiplexeur MC 14539





Porte Analogique



C	Transmission gate Input / Output
0	OFF
1	ON

Le sous- programme NMINT relatif à l'interruption $\overline{NMI}$ est placé à l'adresse Ø F773 du moniteur . NMINT est exécuté soit aprés commande "TRACE" (les points d'arrêt peuvent être actifs ou non) soit à la suite d'une interruption provenant du clavier (signal d'interruption CB1) .

La fonction "TRACE" est réalisée par hardware. Elle permet à l'utilisateur d'exécuter son programme instruction par instruction.

Le "TRACE TIMER" (circuit 74LS 393) est commandé à partir du "System PIA" . La ligne CA2 est relié à l'entrée CLEAR(CL)du compteur ; l'initialisation du compteur se fait alors en mettant CA2 à "1" . Lorsque CA2 passe à "0" , le comptage commence .

Les sorties QA, QB, QC du compteur ne sont pas connectées,QD constitue l'entrée d'interruption CB2 du "System PIA" .

Celle-ci ne bascule donc à "0" que sur le front descendant de la 16^{éme} impulsion d'horloge E .

Lorsque la fonction"TRACE" apparait ,le systéme est généralement sous le contrôle de la routine "RD" de visualisation des registres du MPU .

Les valeurs des contenus des registres sont dans la pile.

La valeur de UPC placée dans la pile correspond à l'instruction suivante du "User program" . Cette fonction s'exécute comme suit : le MPU fait démarrer le compteur à travers la ligne CA2 du "System PIA ". Une instruction RTS est ensuite exécutée pour recharger les valeurs rangées dans la pile dans les registres internes du MPU .

Cette instruction prend 6 cycles d'horloge.L'instruction suivante du "User program" est alors exécutée . Le programme NMINT est activé à la 16^{éme} impulsion d'horloge "NMINT " vérifie si des point d'arrêt sont actifs par un test sur l'indicateur "ROIFLG" .

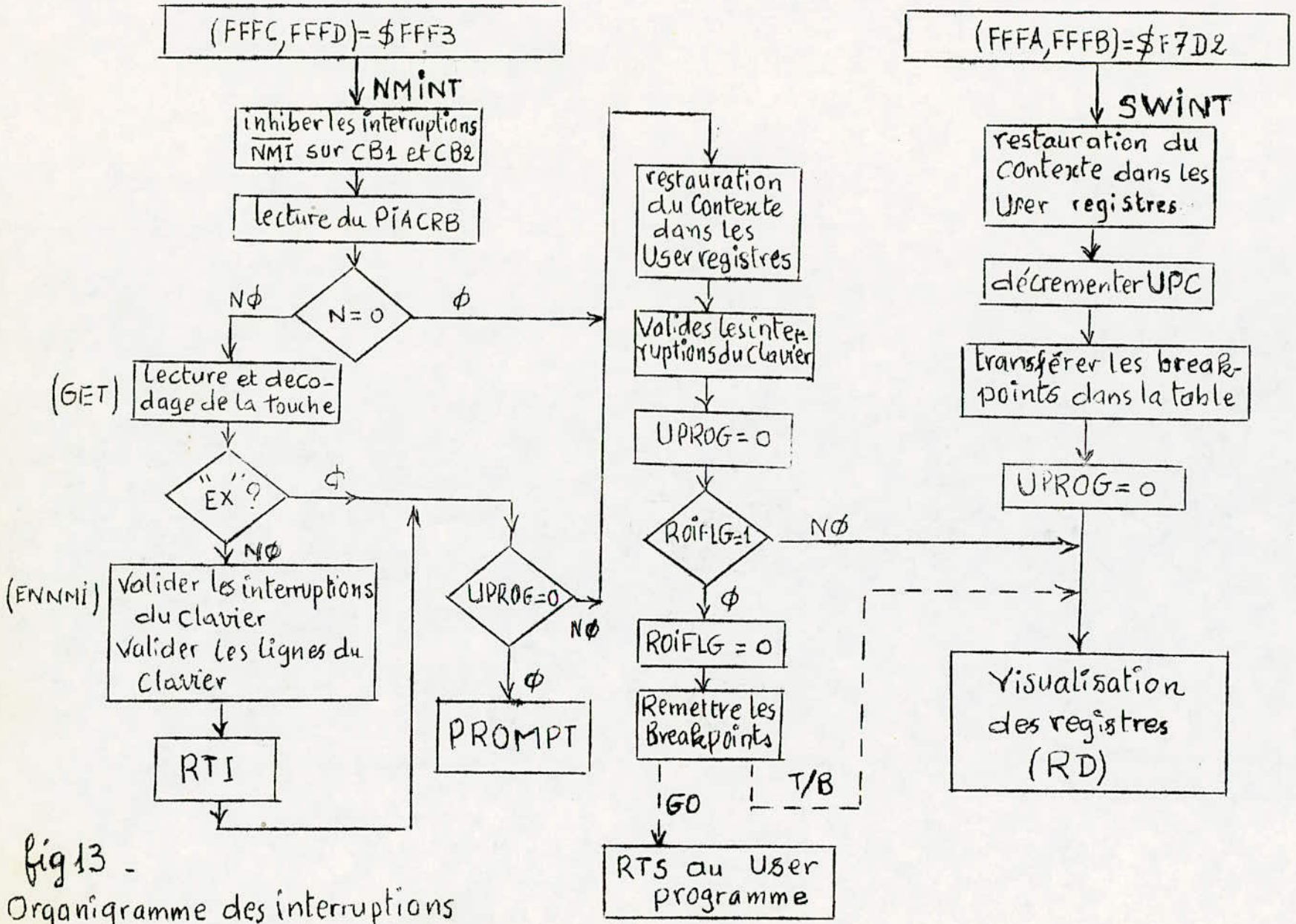


fig 13 -
Organigramme des interruptions NMI et SWI

```
1184                    ********************************************************
1185                    *
1186                    * INTERRUPTS - INTERRUPT HANDLING ROUTINES
1187                    *
1188                    ********************************************************
1189A F773 01          NMINT   NOP             SET IRQ FLAG
1190A F774 0F                   SEI             .
1191A F775 86 04    A           LDAA    #$04    PIA DISABLE CODE
1192A F777 B7 E487  A           STAA    PIACRB  DISABLE NMI'S DURING SERVICE
1193A F77A B6 E487  A           LDAA    PIACRB  READ INT STATUS
1194A F77D 2A 12 F791           BPL     SAVE    IF RETURN FROM TRACE
1195                    * KEY CLOSURE CAUSED NMI
1196A F77F BD F04E  A           JSR     GET     FIND AND DEBOUNCE KEY
1197A F782 81 81    A           CMPA    #$81    'EX' ?
1198A F784 27 03 F789           BEQ     ABORT
1199A F786 8D 26 F7AE           BSR     ENNMI   RE-ENABLE INTERRUPT
1200A F788 3B                   RTI             *** DONE: RETURN ***
1201                    * 'EX' KEY; PROMPT OR ABORT
1202A F789 7D E43B  A ABORT     TST     UPROG   ESCAPE FROM USER PROG ?
1203A F78C 26 03 F791           BNE     SAVE    IF ESCAPE FROM USER PROG
1204A F78E 7E F024  A           JMP     PROMPT  *** ALREADY IN OP-SYST ***
1205A F791 BF E42F  A SAVE      STS     USP     SAVE POINTER TO USER REGS
1206A F794 8E E47E  A           LDS     #STKTOP INIT TO SYST AREA
1207A F797 8D 23 F7BC           BSR     SVSTAT  RECOVER STATUS AT 'EX' TIME
1208A F799 8D 13 F7AE           BSR     ENNMI   RE-ENABLE KEY NMI
1209A F79B 7F E43B  A           CLR     UPROG   SIGNAL NOT IN USER PROGRAM
1210A F79E 7D E438  A           TST     ROIFLG  IS THIS RETURN FROM TRACE ?
1211A F7A1 27 08 F7AB           BEQ     NOTROI  IF NOT
1212A F7A3 7F E438  A           CLR     ROIFLG  SIGNAL NOT ROI NOW
1213A F7A6 FE E439  A           LDX     ROIBAK  GET RETURN ADDR
1214A F7A9 6E 00    A           JMP     0,X     AND RETURN FROM ROI
1215A F7AB 7E F2CA  A NOTROI    JMP     REGBEG  *** TO REG DISPLAY ***
1216                    *
1217                    *
1218A F7AE B6 E486  A ENNMI     LDAA    PIAPB   TO CLEAR FLAGS
1219A F7B1 86 07    A           LDAA    #$07    ENABLE KEY INTERRUPT CODE
1220A F7B3 B7 E487  A           STAA    PIACRB  TO PIA CONTROL REG
1221A F7B6 86 FF    A           LDAA    #$FF
1222A F7B8 B7 E486  A           STAA    PIAPB   ENABLE ALL KEY ROWS
1223A F7BB 39                   RTS             ** RETURN **
1224                    *
1225                    *
1226A F7BC BE E42F  A SVSTAT    LDS     USP     POINT AT STACKED STATUS
1227A F7BF CE E431  A           LDX     #UCC    POINT AT PSEUDO REG AREA
1228A F7C2 32           SVLOOP  PULA            GET STACKED BYTE
1229A F7C3 A7 00    A           STAA    ,X      STORE AT PSEUDO REG RAM LOC
1230A F7C5 08                   INX             POINT AT NEXT REG LOC
1231A F7C6 8C E438  A           CPX     #UPC+2  PAST END ?
1232A F7C9 26 F7 F7C2           BNE     SVLOOP  IF NOT CONTINUE LOOP
1233A F7CB BF E42F  A           STS     USP     SAVE USER SP AT INTERRUPT TIME
1234A F7CE 8E E47C  A           LDS     #STKTOP-2 SET FOR RETURN
1235A F7D1 39                   RTS             ** RETURN **
1236                    *
1237                    *
1238A F7D2 01           SWINT   NOP             SET IRQ FLAG
1239A F7D3 0F                   SEI             .
1240A F7D4 BF E42F  A           STS     USP     POINTER TO USER'S REGS
1241A F7D7 8E E47E  A           LDS     #STKTOP INIT TO SYST AREA
```

fig 14

```
1242A F7DA 8D E0 F7BC          BSR    SVSTAT   RECOVER BREAK STATUS
1243A F7DC FE E436  A          LDX    UPC      BACK UP PROG CNTR
1244A F7DF 09                  DEX             .
1245A F7E0 FF E436  A          STX    UPC      .
1246A F7E3 BD F485  A          JSR    OUTBKS   TAKE OUT BREAKPOINTS
1247A F7E6 7F E43B  A          CLR    UPROG    SIGNAL NOT IN USER PROG
1248A F7E9 7E F2CA  A          JMP    REGBEG   *** TO REG DISPLAY ***
1249                    *
1250                    *
1251A F7EC FE E43C  A UIRQ     LDX    UIRQV    GET USER IRQ VECTOR
1252A F7EF 6E 00    A          JMP    0,X      *** GO TO USER SERVICE ROUTINE ***
1253                    *
```

Supposons que la commande "TRACE"exécute un programme à partir d'un point d'arrêt (breakpoint)."ROIFLG" est alors à 1 ."NMINT" fait appel au programme de remise des points d'arrêt à partir de la table . Ce dernier aboutit au programme de visualisation des registre "RD" .

Si aucun point d'arrêt n'était posé ,"ROIFLG" est à "0"; la commande "TRACE" aboutit directement sur "RD" .

Les interruptions SWI sont utilisées par le moniteur pour poser des points d'arrêt. Le moniteur pointe à l'adresse spécifiée pour remplacer le code opératoire par le code du breakpoint $3F(instruction SWI) .

Ainsi, lorsque le "User program" s'arrête sur un point d'arrêt, le sous-programme SWINT relatif à l'instruction SWI s'éxecute .

Le contenu du pointeur de pile de l'utilisateur (USP) est transféré dans le SP . Le contexte est restauré à partir de la User pile puis transféré dans les User registres (UCC,UB,UA,UX,UPC) . Le UPC est décrémenté ; le compteur de programme pointe à l'adresse du breakpoint .

L'adresse du point d'arrêt et son code opératoire sont sauvegardés dans la table des breakpoints .

Le programme de visualisation des registres est ensuite appelé On peut alors utiliser la touche "GO" pour examiner le contenu des "User registres" .

Si on désire continuer l'exécution du programme à partir de l'endroit où l'on s'est arrêté,on revient sous le contrôle du moniteur en appuyant sur la touche"EX" puis on appuie sur la touche"GO".

L'interruption $\overline{IRQ}$ est réservée à l'utilisateur .

Deux cases mémoire (UIRQV) sont réservées pour ce vecteur d'interruption . L'utilisateur peut y mettre l'adresse de son program -me d'interruption .

La fonction "GO" : (fig 15 et 16)

La touche "GO" réalise 3 fonctions dépendant du mode de fonction -nement en cours .

Si on est dans le mode d'examen de la mémoire ou dans le mode de visualisation des registres , la commande "GO" entraine la visualisation de la case mémoire suivante . Si on ne se trouve pas dans un de ces modes de fonctionnement , la touche "GO" peut être utilisée pour lancer le programme de l'utilisateur à partir d'une adresse ou d'un point d'arrêt .

L'indicateur "ROLPAS" est testé pour déterminer si une nouvelle adresse de lancement a été rentrée .

ROLPAS = 0 : le programme remplace le contenu de UPC par la valeur de l'adresse de lancement mise dans HEXBUF ET HEXBUF + 1 .
ROLPAS = 1 : le "User programm" s'exécute à partir de la valeur de UPC .
"UPROG" est mis à Ø 01 pour indiquer un "User program"

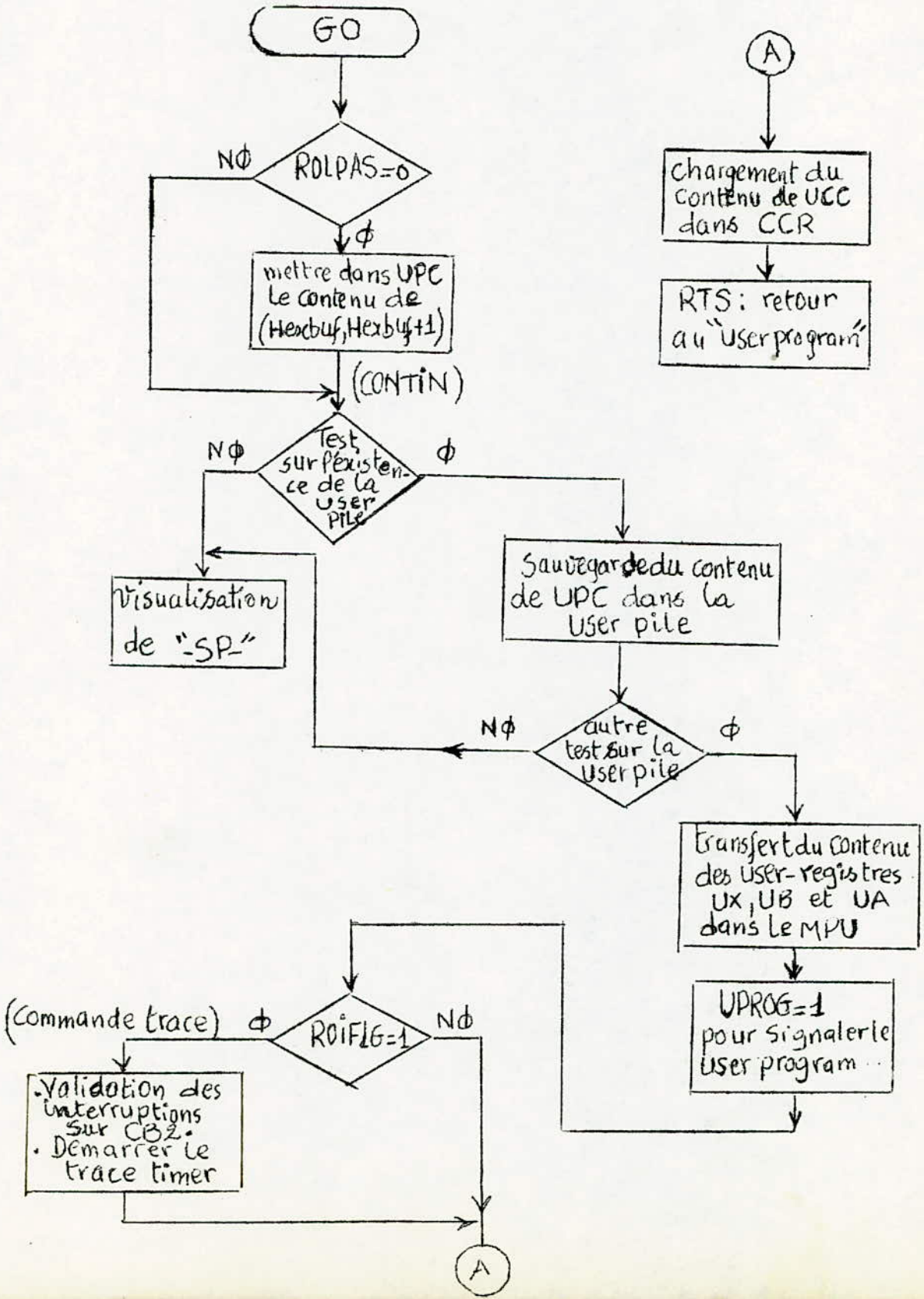
Un test sur "ROIFLG" détermine si des points d'arrêt ont été posés . S'il n'y a pas de breakpoints, le MPU exécute une instruction RTS de retour au "User program" . Celui-ci s'exécute à partir de l'adresse contenue dans le UPC .

Si des breakpoints ont été posés (ROIFLG =1) on fait appel à la commande "trace" pour exécuter une instruction (celle correspondant à l'adresse du point d'arrêt) . Recevant l'interruption $\overline{\text{NMI}}$, le programme NMINT vérifie si les indicateurs pour les points d'arrêt et la fonction "trace" sont tous deux actifs(ROIFLG=1 et N=0) .

.../...

fig 15 _ Routine GO de lancement d'un User program

Organigramme



```
1119                        ************************************************
1120                        *
1121                        * GO - GO TO USER PROGRAM
1122                        *
1123                        ************************************************
1124A F6F3 7D E423  A GO      TST    ROLPAS   HEX DATA PRIOR TO 'GO' ?
1125A F6F6 26 06 F6FE       BNE    CONTIN   IF NOT; ASSUME UPC
1126A F6F8 FE E42C  A       LDX    HEXBUF   GET ENTERED VALUE
1127A F6FB FF E436  A       STX    UPC      STORE AS GO ADDR
1128A F6FE CE F70B  A CONTIN LDX    #GO1     RETURN ADDR AFTER ROI
1129A F701 FF E439  A ROI   STX    ROIBAK   SAVE IN RAM
1130A F704 86 01    A       LDAA   #1
1131A F706 B7 E438  A       STAA   ROIFLG   SIGNAL SINGLE TRACE
1132A F709 20 03 F70E       BRA    GOTO     EXIT (NO BREAKS)
1133                        * COME HERE AFTER RUNNING ONE INSTRUCTION
1134A F70B BD F45F  A GO1   JSR    INBKS    INSTALL BREAKPOINTS
1135A F70E BE E42F  A GOTO  LDS    USP      GET USER'S STACK POINTER
1136A F711 86 55    A       LDAA   #$55     START TEST FOR EXISTANCE OF STK
1137A F713 36               PSHA
1138A F714 32               PULA
1139A F715 81 55    A       CMPA   #$55     DID IT GO ?
1140A F717 26 10 F729       BNE    BADSTK   NO; STACK IS BAD
1141A F719 B6 E437  A       LDAA   UPC+1    LOW BYTE
1142A F71C 36               PSHA            STACK FOR RTS
1143A F71D B6 E436  A       LDAA   UPC      HIGH BYTE
1144A F720 36               PSHA
1145A F721 86 AA    A       LDAA   #$AA     SEE IF STACK STILL OK
1146A F723 36               PSHA
1147A F724 32               PULA
1148A F725 81 AA    A       CMPA   #$AA
1149A F727 27 1E F747       BEQ    GOEXIT   OK; FINAL EXIT SEQ
1150A F729 CE 406D  A BADSTK LDX   #$406D   MESSAGE "-SP- ??" TO 7-SEGS
1151A F72C FF E41D  A       STX    DISBUF
1152A F72F CE 7340  A       LDX    #$7340
1153A F732 FF E41F  A       STX    DISBUF+2
1154A F735 CE 5353  A ALTBAD LDX   #$5353
1155A F738 FF E421  A       STX    DISBUF+4
1156A F73B 8E E47E  A       LDS    #STKTOP  INIT TO GOOD AREA
1157A F73E CE F0A2  A       LDX    #DIDDLE  DO-NNOTHING SUB
1158A F741 FF E419  A       STX    MNPTR    STORE AS MAIN PROG
1159A F744 7E F0BB  A       JMP    PUT      ONLY ESCAPE IS RESET OR 'EX'
1160                        *
1161A F747 FE E434  A GOEXIT LDX    UX       RECOVER USER STATUS
1162A F74A F6 E432  A       LDAB   UB
1163A F74D B6 E433  A       LDAA   UA
1164A F750 36               PSHA            TEMP SAVE ON USER'S STACK
1165A F751 86 01    A       LDAA   #1
1166A F753 B7 E43B  A       STAA   UPROG    FLAG SIGNALS IN USER PROG
1167A F756 7D E438  A       TST    ROIFLG   TRACE EXIT ?
1168A F759 27 12 F76D       BEQ    ABSOUT   IF NOT;; JUST GET GOING
1169A F75B 86 3C    A       LDAA   #$3C
1170A F75D B7 E485  A       STAA   PIACRA   HOLDS TRACE COUNTER RESET
1171A F760 B6 E486  A       LDAA   PIAPB    READ TO CLEAR ANY INT FLAG
1172A F763 86 0E    A       LDAA   #$0E
1173A F765 B7 E487  A       STAA   PIACRB   ENABLE TRACE NMI
1174A F768 86 34    A       LDAA   #$34
1175A F76A B7 E485  A       STAA   PIACRA   RELEASE TIMER
1176A F76D B6 E431  A ABSOUT LDAA   UCC      TIMED EXIT TO USER PROG
1177A F770 06               TAP             SET USER COND CODES
1178A F771 32               PULA            GET USER A-REG; DON'T MESS 'CC'
1179A F772 39               RTS             *** EXIT TO USER PROG ***
1180                        *
```

Si c'est le cas , D5BUG insére les points d'arrêt dans le "User - program" à partir de la table des breakpoints avant de revenir au programme de l'utilisateur . C'est de cette manière qu'un programme est exécuté à partir d'un point d'arrêt avec la touche "GO" .

III.3 Extension du KIT D5 :

Grâce aux 56 Koctets réservés aux circuits externes, le KIT D5 présente de grandes possibilités dans son extension .

Le bas de la carte est un connecteur de format standard pour l'Exorciser ce qui permet au KIT de recevoir sans problèmes de nouvelles cartes en parallèle dont chacune peut présenter une ou plusieurs fonctions qui sont soit données par le constructeur soit réalisées par l'utilisateur.

On peut par exemple implanter une carte de visualisation pour pouvoir dialoguer avec un terminal vidéo ou bien avec un simple téléviseur muni d'un clavier ASC II .

De plus,on peut utiliser des moniteurs de mise au point de programmes en Hexa pour visu,imprimante,programmateur d'EPROM ...etc.

Il est possible de travailler sur des cartes de PROMS contenant les languages basic et d'éditeur assembleur.

Il ne faut pas oublier qu'on peut travailler avec la K7 grâce à l'interface cassette donné sur le KIT .

Tout ceci évoque les différentes possibilités d'extension du KIT D5; celles-ci ne demandent qu'à être exploitées pour rendre ce KIT un véritable outil de travail et de recherche.

CHAPITRE 3 : Exemples de programmation avec le KiT D5

Dans ce chapitre, nous présentons dans une première partie quelques exemples de programmation avec le KiT D5 .

La seconde partie consiste en une application du KiT pour une acquisition de données .

I - Exemples

I_1 - l'horloge minute-seconde analogique

Cette application nécessite une réservation de 4 cases mémoires une case pour le registre Minute " Rmin ", une case pour le registre seconde " Rsec " et 2 cases pour le registre "RTick " . Le principe de cette horloge est d'incrémenter "RTick " aprés chaque rafraichissement des digits (qui dure environ 1,87 ms).

Au bout de 535 itérations on incrémenté "Rsec " et on initialise "RTick" pour un nouveau comptage. Le comptage peut aller jusqu'à 99 mn 59s.

L'Affichage se fait en DCB .

- L'organigramme et le programme sont donnés respectivement fig 1 et fig 2.

I_2 - visualisation des contenus de positions mémoire par - délai :

Le KiT D5 n'offre pas la possibilité de vérifier qu'un programme est rentré correctement ou du moins pour le faire, il faut appuyer chaque fois sur la touche "GO", ce qui n'est guère pratique

Dans ce but, nous avons élaboré un programme d'affichage par délai byte par byte du programme à vérifier .

Deux cases mémoires sont réservées pour spécifier l'adresse du programme à vérifier et 2 autres cases "Tmin et Tsec " sont prévues pour le délai d'affichage. Le comptage du délai DCB peut aller

.../...

fig 1 - Organigramme de l'horloge Minute-Seconde

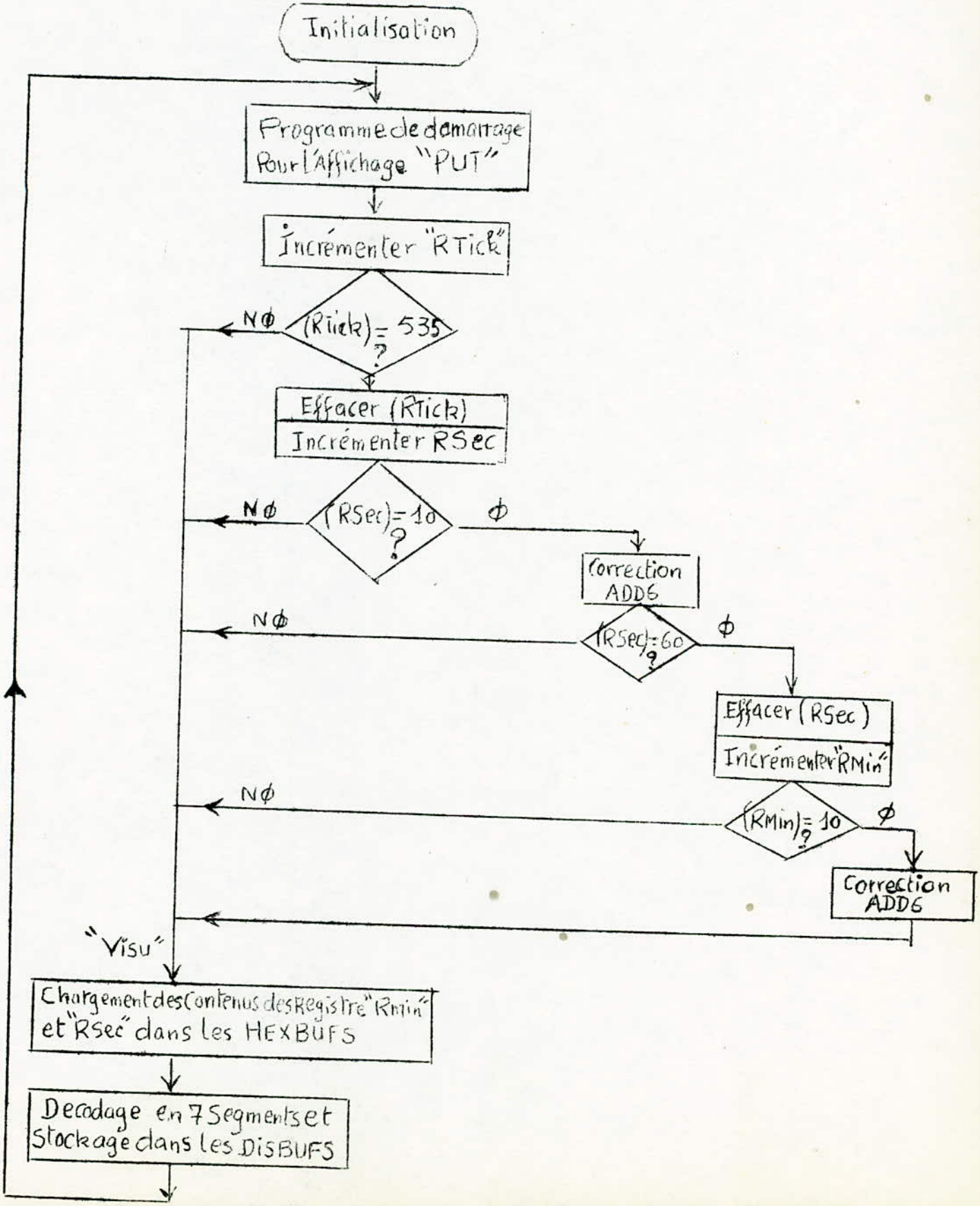


fig 2 - programme de l'Horloge Minute-Seconde

$ 0005

```
                                    ——— o ——— initialisation
0001        00          RMin.
0002        00          RSec.
0003        00
0004        00          ] RTick.

                                    ——— o ——— démarrage
0005        CE  000E            LDX # TIME
0008        FF  E419            STX MNPTR
000B        7E  F0BB            JMP PUT

                                    ——— o ——— programme TIME
000E        DE  03      _TIME_  LDX RTick           - enregistrer le Tick
0010        08                  INX                   de l'HORLOGE.
0011        DF  03              STX RTick
0013        8C  0217            CPX # 535 (#0217)   - y a t-il 1 seconde?
0016        26  2C              BNE VISU
0018        CE  0000            LDX # 0000
001B        DF  03              STX RTick           - effacer RTick.
001D        7C  0002            INC RSec.           - enregistrer 1 Sec.
0020        96  02              LDA A RSec.
0022        8A  F5              ORA A # F5          - y a t-il 10 sec?
0024        43                  COM A
0025        26  1D              BNE VISU
0027        96  02              LDA A RSec.         - correction
0029        8B  06              ADD A # 06            ADD6.
002B        97  02              STA RSec.
002D        81  60              CMP A # 60          - y a t-il 60 Sec.?
002F        26  13              BNE VISU
0031        7F  0002            CLR RSec.           - effacer Rsec.
0034        7C  0001            INC RMin.           - enregistrer 1 mn.
0037        96  01              LDA A RMin.
0039        8A  F5              ORA A # F5          - y a t-il 10 mn?
003B        43                  COM A
003C        26  06              BNE VISU
003E        96  01              LDA A RMin.         - correction
0040        8B  06              ADD A # 06            ADD6.
0042        97  01              STA A RMin.
0044        DE  01      _VISU_  LDX RMin.           - Charge des HEXBUFS par
0046        FF  E42C            STX HEXBUF            les contenus de Rmin et RSec.
0049        BD  F120            JSR DYSCOD          - Décodage 7 Segments.
004C        86  03              LDA A # %00000011   - effacer les 2 premiers
004E        BD  F195            JSR CLRDS             digits.
0051        39                  RTS                 - Retour à l'affichage.
```

fig 3 - Organigramme de visualisation des contenus de positions mémoires par délai

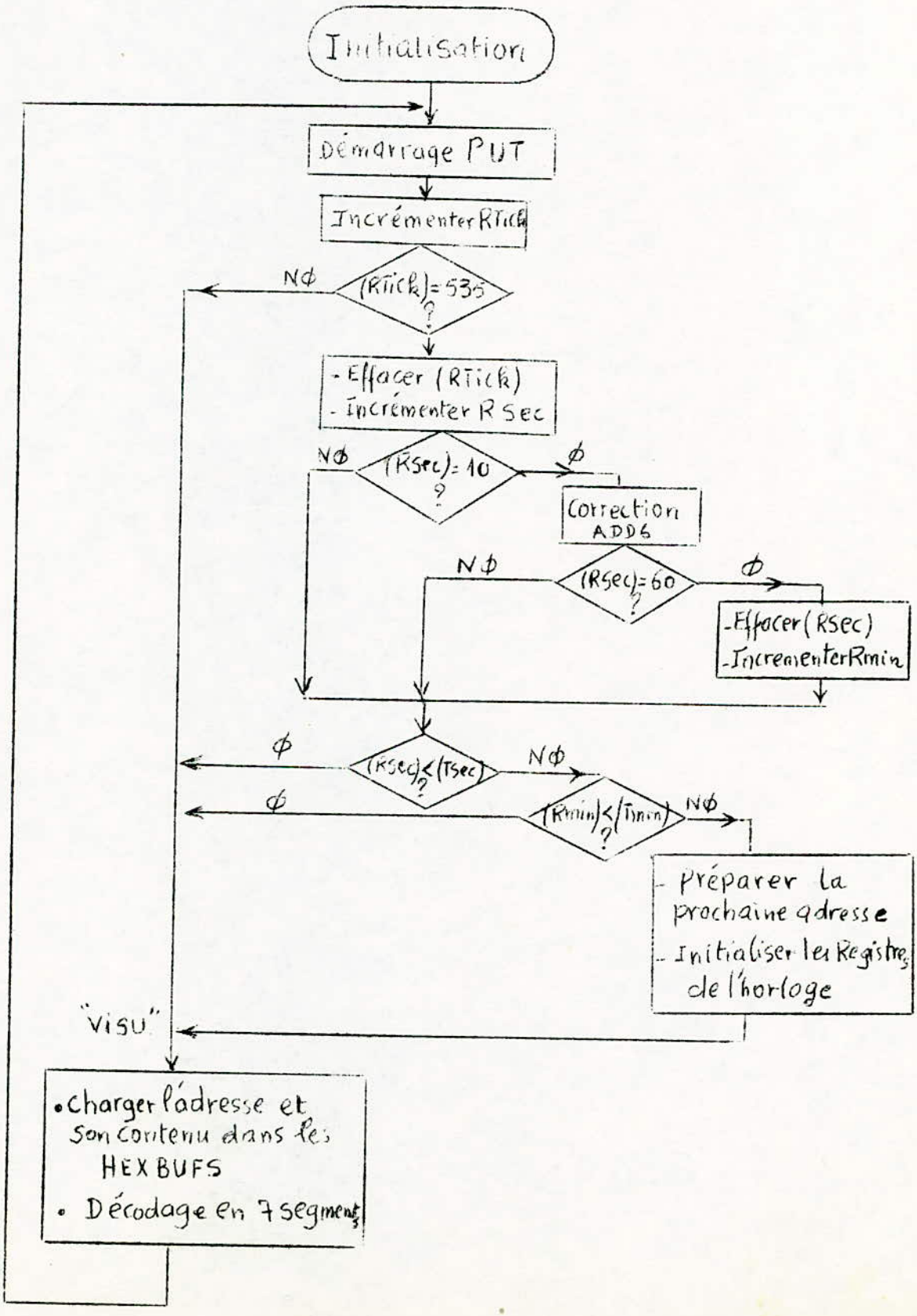


fig 4 - programme de VISUALISATION des contenus de positions mémoire par délai.

$0008

```
0000    00          Rmin.                          Initialisation
0001    00          RSec.
0002    00          ]                              //
0003    00          ]RTick                         //
0004      TSec.
0005      Tmin.                                    Réservation
0006    ] adresse de                               //
0007    ] départ                                   //

                    ——— o ——— démarrage

0008    CE 0011              LDX # PROG
000B    FF E419              STX  MNPTR
000E    7E F0BB              JMP  PUT

                    ——— o ——— programme

0011    DE  02     _PROG_    LDX  RTick           - engistrer le Tick
0013    08                   INX                    de l'horloge.
0014    DF  02               STX  RTick
0016    8C  0217             CPx # 535($0217)     - yat-il 1 Seconde?
0019    26  41               BNE  VISU
001B    CE  0000             LDX # 0000           - RAZ de Rtick.
001E    DF  02               STX  RTick
0020    7C  01               INC  RSec.           - enregistrer 1 sec.
0022    96  01               LDA A RSec.          - yat-il 10 sec?
0024    8A  F5               ORA # F5
0026    43                   COM A
0027    26  10               BNE  TEST
0029    96  01               LDA A RSec.          - correction
002B    8B  06               ADD A # 06             ADD6.
002D    97  01               STA A  RSec.
002F    81  60               CMPA # 60            - yat-il 60 sec?
0031    26  06               BNE  TEST
0033    7F  0001             CLR  RSec.           - éffacer RSec.
0036    7C  0000             INC  Rmin.           - engegistrer 1mn.
0039    B6  0001   _TEST_    LDA A RSec.          - Periode
003C    B1  0004             CMPA TSec.             d'affichage.
003F    2D  1B               BLT  VISU
0041    B6  0000             LDA A Rmin.
0044    B1  0005             CMPA Tmin.
0047    2D  13               BLT
0049    FE  0006             LDX  0006            - Passer à la case
004C    08                   INX                    mémoire suivante.
004D    FF  0006             STX  0006
0050    7F  0000             CLR  Rmin.           - RAZ des
0053    7F  0001             CLR  RSec.             Registres
0056    7F  0002             CLR  0002 ]            HORLOGE.
0059    7F  0003             CLR  0003 ]RTick
005C    FE  0006   _VISU_    LDX  0006            - stockage dans
005F    FF  E42C             STX  HEXBUF            les
0062    A6  00               LDA A 00,X             HEXBUFS.
0064    B7  E42E             STA A HEXBUF+2
0067    BD  F120             JSR  DYSCOD          - Décodage en 7 segts
006A    39                   RTS                  - Retour à l'affichage
                                                    par PUT.
```

jusqu'à 9mn 59s .

- L'organigramme et le programme sont donnés en fig 3 et fig 4 .

II - Utilisation du KiT D5 pour une acquisition de données :

II.1. Synoptique d'une chaîne d'acquisition :

La figure 5 représente le synoptique d'une acquisition de données .

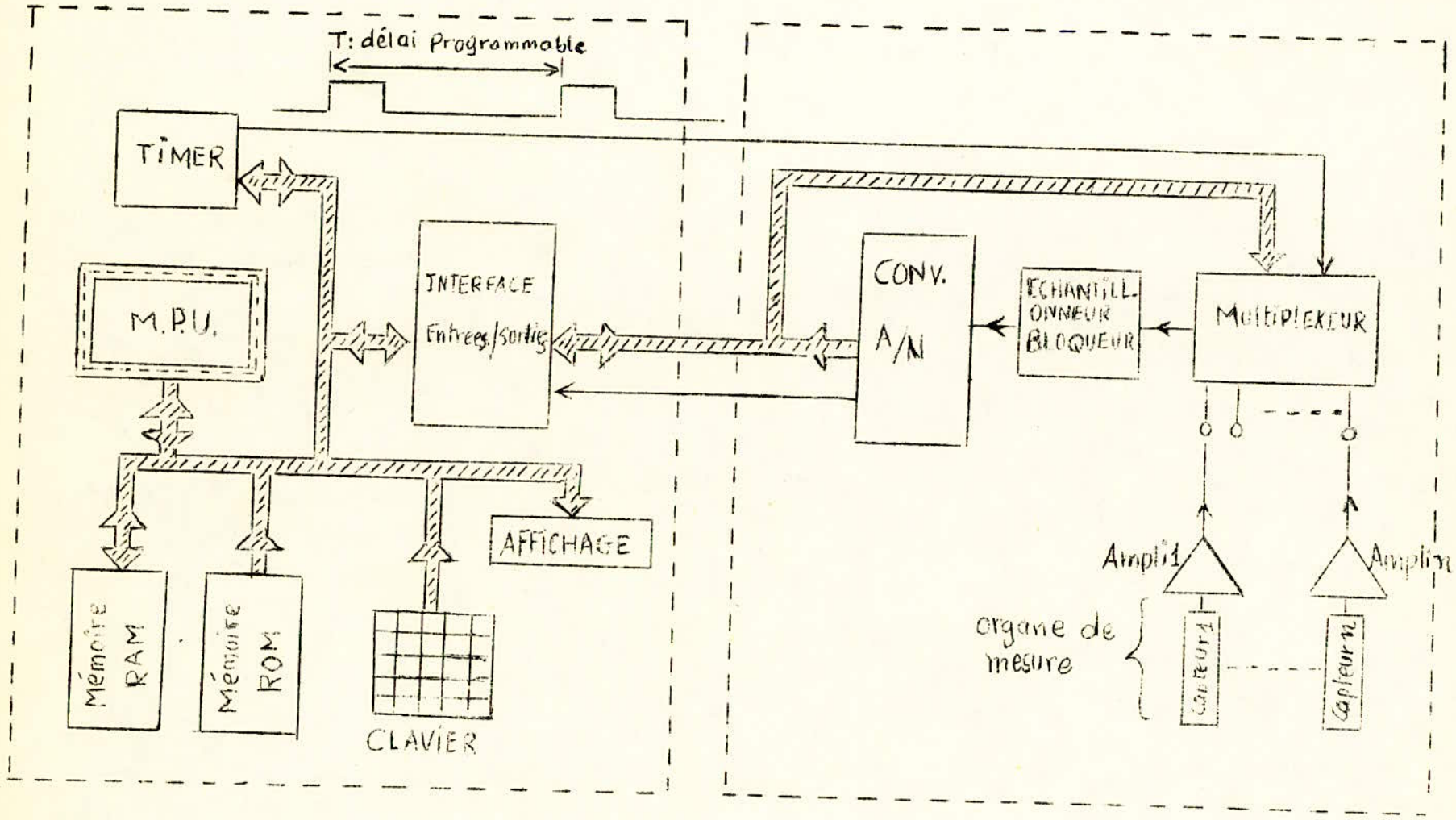
L'élément central de cette chaîne est un microordinateur qui doit être couplé à un interface analogique comprenant successivement :

- les capteurs et leurs amplis associés.
- un multiplexeur (n voies ou canaux) qui permet la sélection des différentes voies
- un échantillonneur - bloqueur
- un convertisseur Analogique /Numérique

Le microordinateur réalise l'acquisition de l'information analogique apparaissant sur les canaux en les explorant séquentiellement . Le résultat de cette acquisition peut être traité puis stocké ou envoyé sur un organe de sortie(visualisation,enregistrement sur K7 ,imprimante ...etc).

Un cycle complet d'acquisition comprend : la commande de multiplexage (avec la possibilité de validation du multiplexeur par timing),la conversion et le stockage de la donnée; ces mêmes opérations étant répétées n fois pour une prise en compte de toutes les entrées.

Pour notre étude,vu que nous ne disposons pas de capteurs ni de multiplexeur, nous avons été amené à ne considérer qu'une seule voie.Le capteur correspondant est simulé à l'aide d'une tension variable .



Environnement <<NUMÉRIQUE>> :
Le MICROORDINATEUR

Environnement << ANALOGIQUE >>

fig 5.

Comme microordinateur,nous disposons du KiT D5 .

Le convertisseur A/N utilisé est du type ADC 0804 de 8bits,il se - ra commandé à travers le " USER PIA " du KiT.

Le timing est réalisé par " SOFT " .

L'organigramme de la figure 6 décrit les différentes phases de notre étude .

La première phase est réservée à l'acquisition de données,la seconde au traitement de données,la troisième à la visualisation du résultat .

. Notre application consiste à réaliser la première phase qui est l'acquisition de données puis de visualiser le resultat (3^{eme} phase).

. Pour la phase de traitement (representée en pointillé), nous presentons au paragraphe II_5 un exemple qui se rapporte à une sonde de température.

II_2 - Etude de l'interface parallèle PiA 6821 et du convertisseur Analogique /Numérique :

* Le PiA 6821 (L'USER PiA "U_9") :

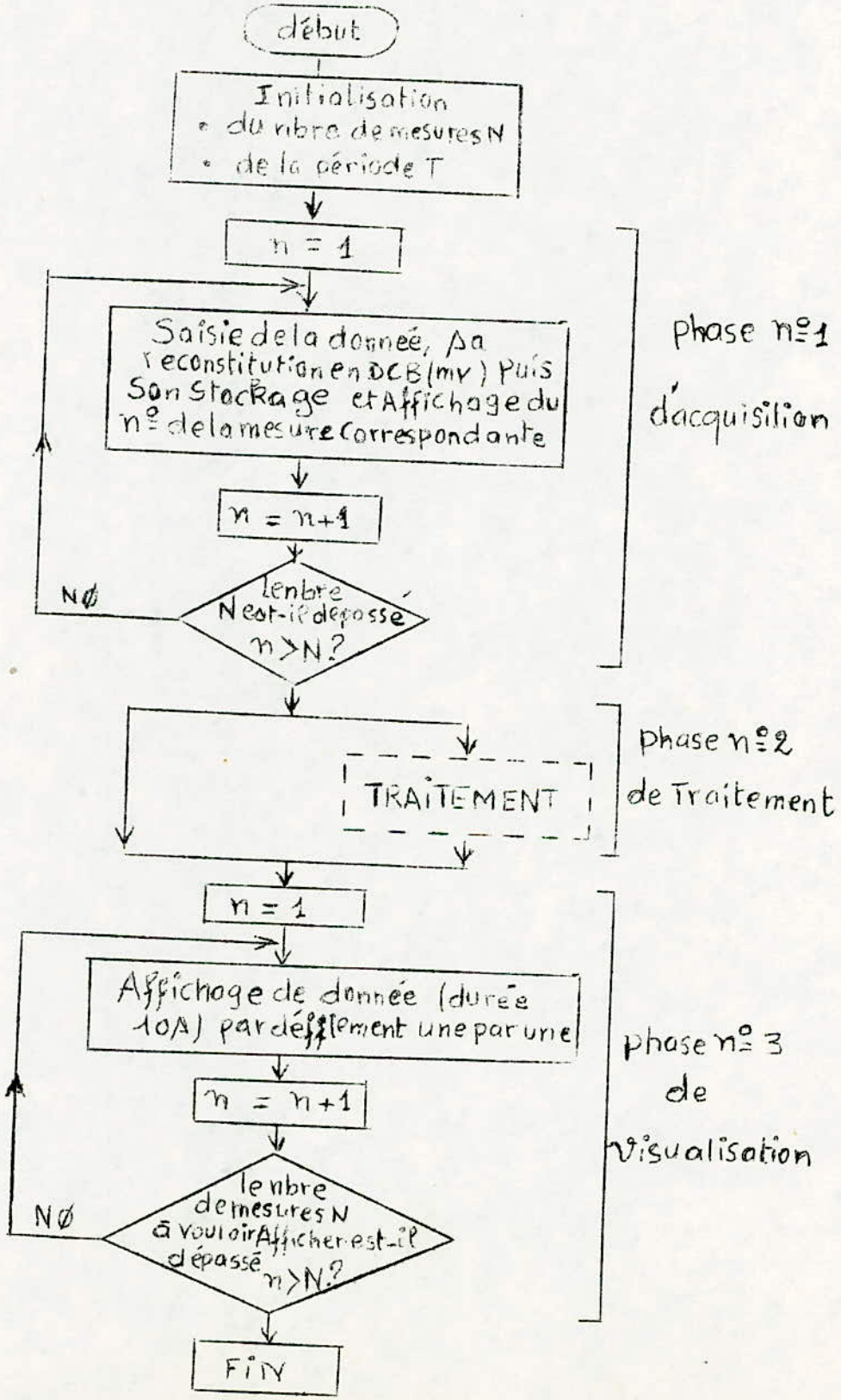
Dans ce paragraphe nous ne nous limiterons qu'à la programmation de cet interface .

Pour une étude plus détaillée,se référer à la bibliographie ligne N° 1 et 2 .

L'interface PiA 6821 possède 2 ports A et B de 8 Entrées/Sorties . Chaque port a un registre de contrôle un registre de donnée, un registre de direction et 2 signaux de commande(fig 7) .

Chaque registre est séléctionné par les premières lignes d'adresses Ao et A_1 correspondant sur le PiA à RSo et RS_1 :

fig 6 - ORGANIGRAMME GÉNÉRAL (pour une seule voie)



RS_1	RS_0	Registre	Port	Designation
0	0	de données	A	PiA AD
0	1	de contrôle	A	PiA AC
1	0	de données	B	PiA BD
1	1	de contrôle	B	PiA BC

Le fonctionnement des 2 ports A et B est identique.

Pour notre application,on utilise le port B

Le signal de commande CB_2 est bidirectionnel et peut donc être programmé soit en entrée, soit en sortie.Par contre CB_1 est toujours en entrée .

Les fronts actifs de CB_1 et CB_2 sont programmables.

. mode de fonctionnement du PiA voir fig 8

Lorsque le bit 5 est à 1 , le signal de commande CB_2 est déclaré en sortie . Il est possible alors de programmer le port B suivant 3 modes qui sont :

- Le mode "positionnement à 0 ou 1 ": b5=1,b4=1,b3=0ou1
- Le mode "Sortie impulsion": b5=1,b4=0,b3=1
- Le mode "dialogue " : b5=1,b4=0,b3=0

Pour notre application on opte le 1er mode.

On génére un front positif sur CB_2 par positionnement à zéro puis à 1 .

Le signal CB_2 active l'entrée $\overline{WRITE}$ ($\overline{WR}$) du convertisseur .

Le basculement de "0" à "1" de cette entrée provoque le transfert de la donnée convertie à partir du buffer du convertisseur vers le registre du port B (PiA BD) .

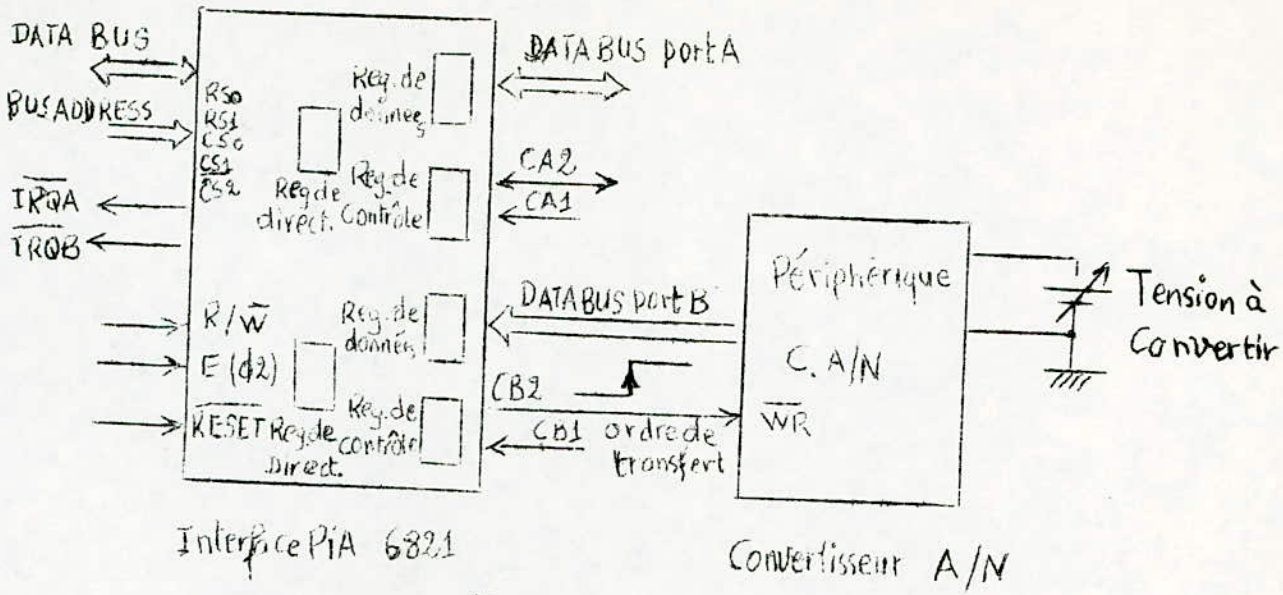


fig 7.

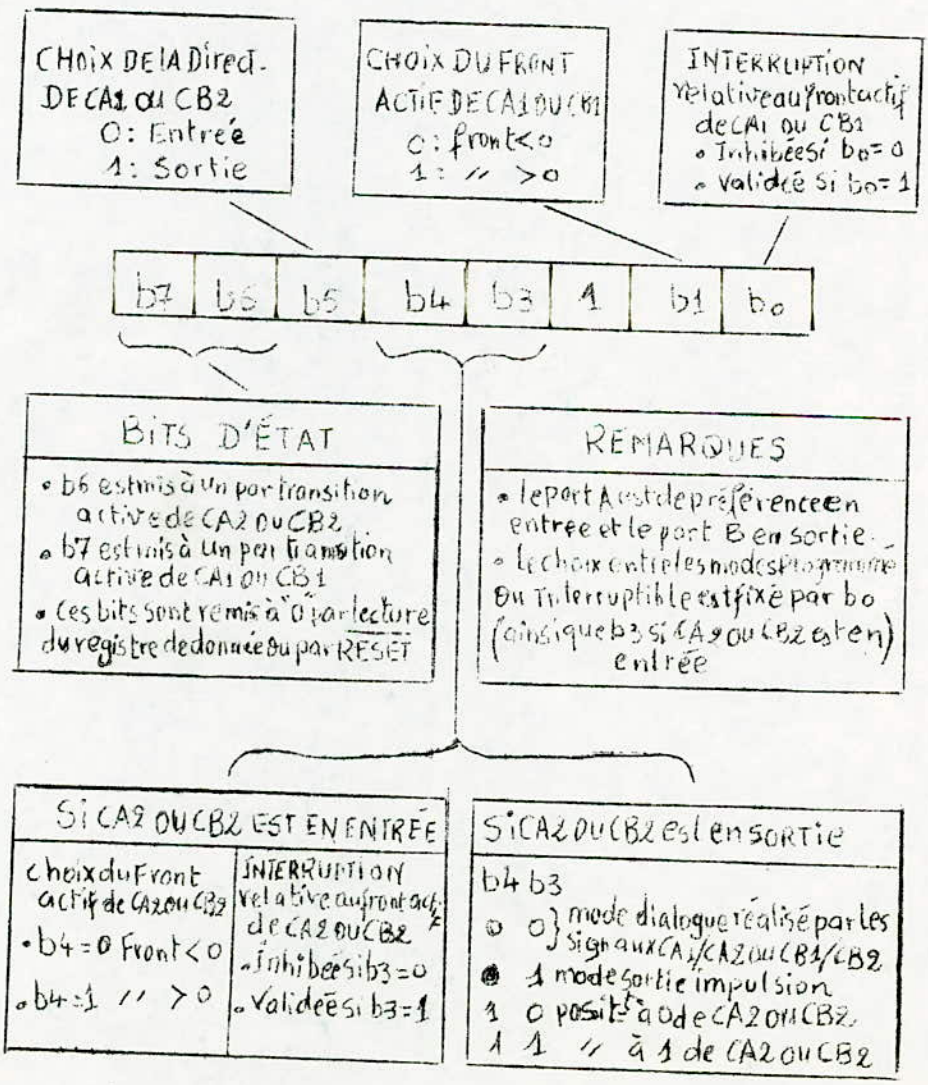


fig 8 - Synoptique de la détermination du mot de Commande et d'état du PIA

* Le convertisseur Analogique/Numérique : ADC 0804 .

Le ADC 0804 est un circuit CMOS, travaillant par approximations successives .

Son alimentation est de + 5 V.

Il est parfaitement compatible avec la logique du MPU .

Ses caracteristiques sont :

- une résolution de n=8 bits .
- une tension de référence Uref = + 5 V .
- un temps de conversion t_C = 100 µs
- un temps d'accés tacc = 135 n s.
- La tension de l'échelon correspondant au bit du poids le plus faible (LSB) ;sera :

$$e = \frac{Uref}{2^n} = \frac{Uref}{2^8} = \frac{5\ V}{256} = 19{,}53\ mv .$$

e = 19,53 mv

- L'erreur commise sur la conversion, donnée par la formule :

$$\varepsilon_c \leqslant \frac{1}{2} \cdot \frac{Uref}{2^n}$$

sera : **$\varepsilon_c \leqslant$ 9,76 mv.**

II 3 - Schéma de montage :

Notre schéma comporte le convertisseur A/N dont son entrée de séléction de boîtier est toujours validée($\overline{CS}$ est au niveau bas). L'A.D.C. travaille continuellement de telle manière que le résultat de la conversion de la tension analogique se trouve en permanence sur le Buffer tri-state de sortie ($\overline{RD}$ au niveau bas) .

Un front actif montant sur " $\overline{WR}$ " (voir fig 10).entraine le transfert du résultat sur le port B de l'interface (PiA BD).

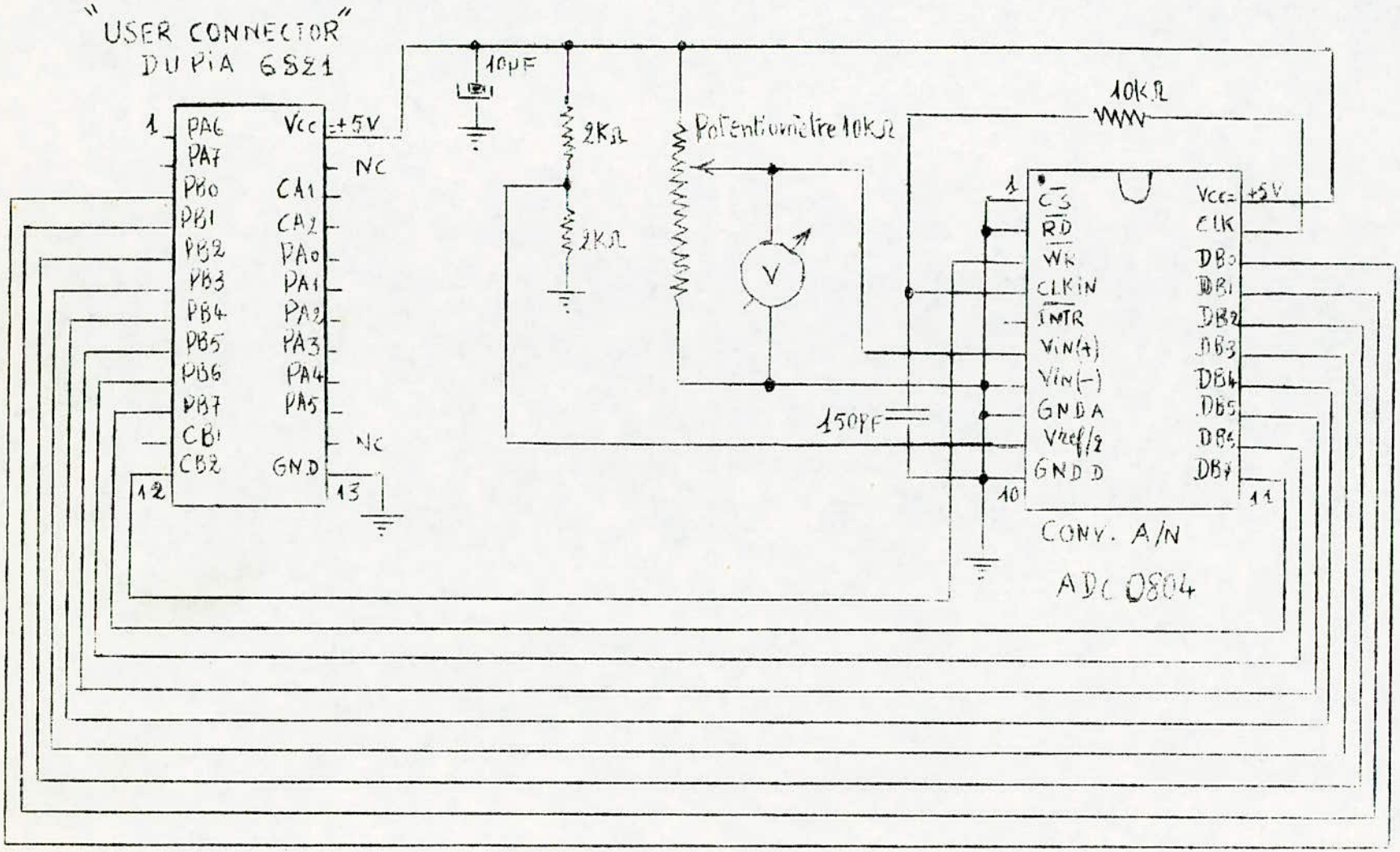
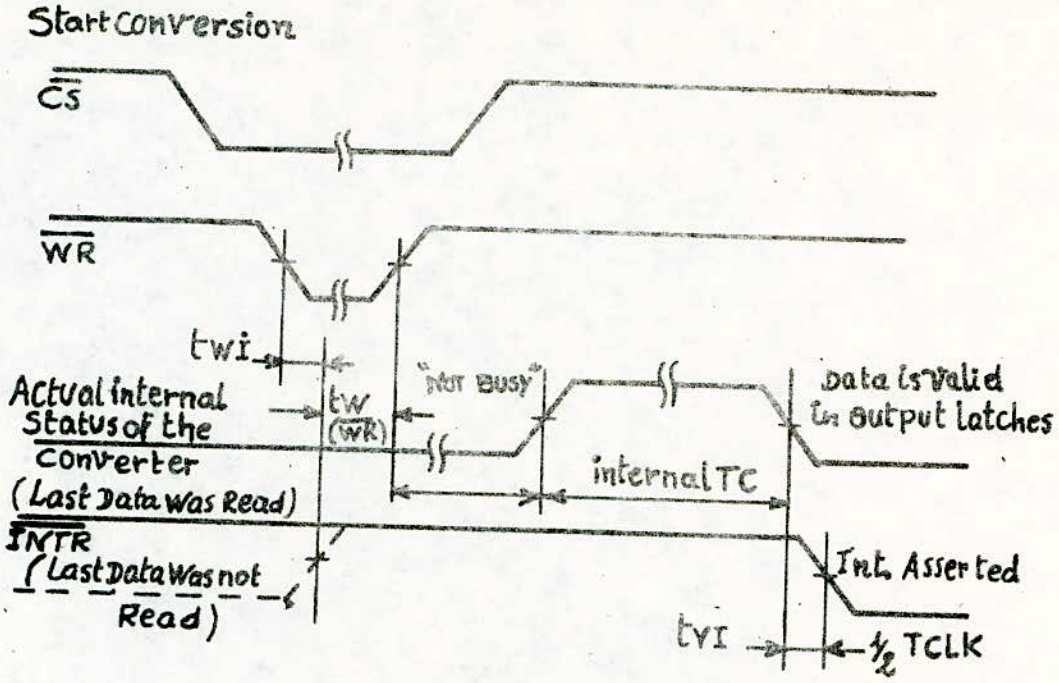


fig 9- Schéma de montage



Output Enable and Reset INTR

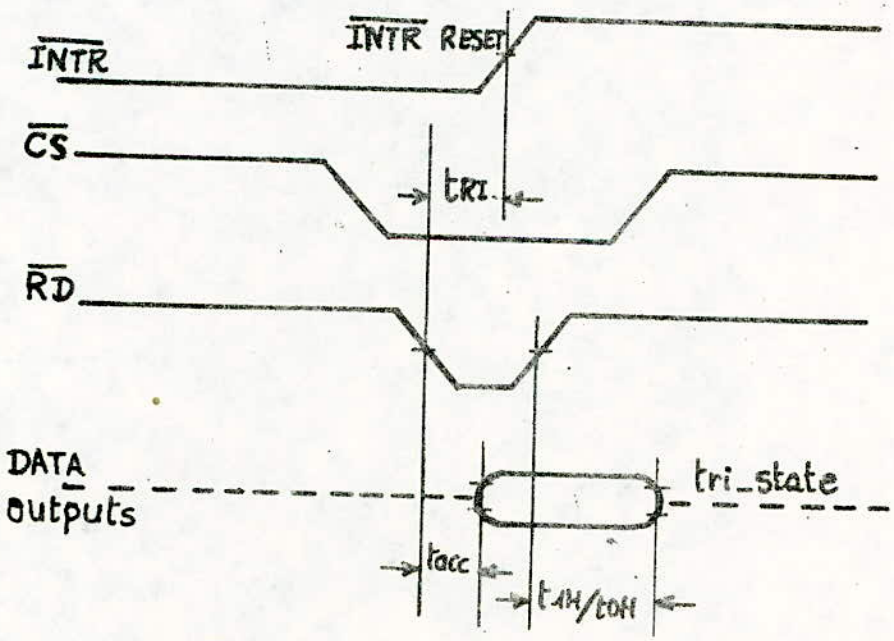


fig 10 - Signaux de fonctionnement du convertisseur donnés par le Constructeur

II_4 - programmes :

II 41 - Sous programmes :

A. Sous programmes de l'horloge et de son initialisation :

- Sous programme de l'horloge ($. E00E) :

Ce sous programme ressort de l'application precedente(parag. I_4) sauf qu'il ne posséde pas de sortie pour affichage .

. L'organigramme et le programme sont donnés respectivement en fig 11 et fig 12 .

- Sous programme d'initialisation des registres horloge($ E07C) :

Etant donné que nous faisons souvent appel à cette initialisation des registres horloge , nous avons jugé utile de l'employer en sous programme .

. Son programme est donné par le listing fig 15

B - Sous programmes "RESULt" et "-ENd --" :

Le principe est éxactement le même pour les 2 mots .

Les 2 ,sont utilisés pour le programme principal 2 d'affichage

Le 1er s'affiche avec le delai du programme principal 2 d'affichage 10 s .

Le second représente la fin d'affichage des données.

- Le sous programme " RESULt " est donné par le listing fig 13 .
- le sous programme "ENd " est donné par le listing fig fig 14 .

C - Sous programme "DATA" ($ E089):

C'est le sous programme le plus important faisant la liaison du "Hard " au "programme principal 1" de lecture des données .

Au départ, il initialise tout le port B puis il génére un front o sur CB2 , le convertisseur sera donc averti et transfèrera son résultat au PiA BD . Le microprocesseur vient lire l'information sur le port B .Il reconstitue la valeur lue du binaire à sa vraie valeur en "DCB" (mV) en passant par une multiplication.

Il stocke le résultat dans 2 cases mémoires,il réinitialise le port B de l'interface et revient au "programme principal 1" .

. L'organigramme et le programme sont representés respecti-vement en fig 16 et fig 17 .

II 42 . " programme principal 1 " ($ E0D6):

C'est le programme le plus important nous permettant l'acquisition de la donnée d'une manière séquentielle .

Le delai est programmable . Les cases mémoires $ E008 et $E009 representent respectivement registre de la période partie minute"Tmin" et registre de la période partie seconde "Tsec " .

La période { "Tmin" , "Tsec " } s'écrit normalement lors de sa programmation (en DCB).

Le nombre de mesures "N" sera aussi programmable.

Il sera contenu dans la case mémoire $E004 et s'écrira en Hexadecimal.

Il peut aller jusqu'à $ FF (c'est à dire 256-1 en decimal).

Dans notre application, la capacité de la RAM reserveé pour le stockage ne dépasse guère les 128 octets par conséquent le nombre de mesures "N" qu'on pourra atteindre sera: $N = \frac{128}{2} - 1 = 63$

(le " _1" est reservé pour l'adresse du sous programme "RESULt").

Le"programme principal 1" demarre à partir de $E0D6

. L'organigramme et le programme sont donnés respectivement aux fig.18 et fig 19.

II 43 . "programme principal "2 (∅ E12D):

C'est le programme qui nous permet de visualiser le résultat de la série de mesures "N" .

Ainsi,lorsque le "programme principal 1 " entamme cette serie "N",l'affichage des résultats s'éffectue automatique et jusqu'à visualiser toutes les mesures .

Si on veut démarrer le "programme principal 2" aprés une 1ère visualisation,on aura ainsi le choix pour programmer N une nouvelle fois en une valeur N' .

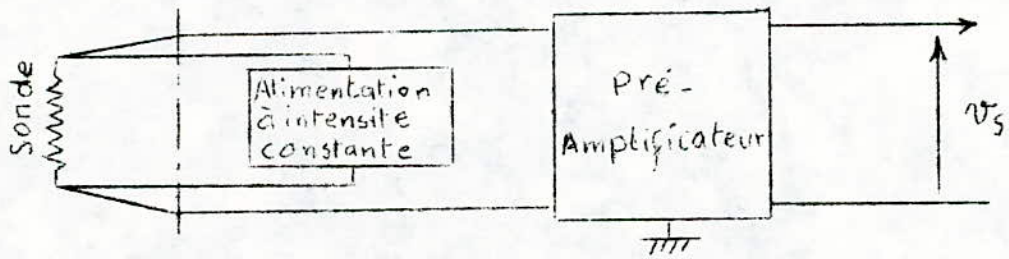
L'adresse de démarrage de ce programme est : ∅E1.2D.

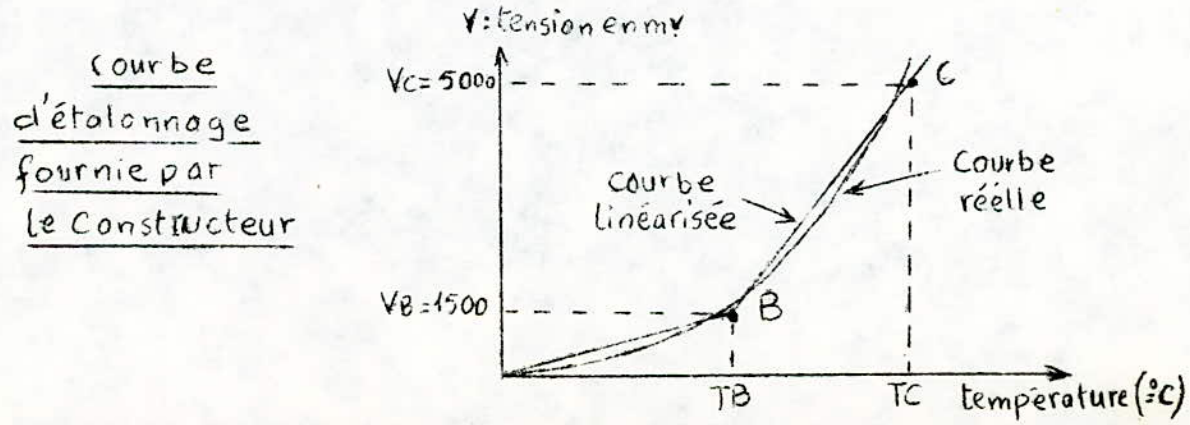
. L'organigramme et le programme sont donnés respectivement aux fig 20 et fig 21 .

II 5 - Exemple de traitement de données de temperature :

Pour illustrer un exemple de traitement de donnée ,nous avons utilisé les caractéristiques d'une sonde de température de resistance en Nickel et fabriquée par Wishay - Micromesures .

Il s'agit donc de reconstituer la courbe réellë des température à partir des valeurs mesurées en tension.(mV).





Ainsi pour se rapprocher de la courbe réélle nous utilisons 3 points A,B et C .

-Si la valeur de la donné "V" en millivolt se trouve dans la pa partie linéaire A B , nous appliquerons la formule suivante :

$$T = V. \frac{TB - TA}{VB - VA}$$

- Si la valeur "V" se trouve entre B et C.

On fera : $V = T . \frac{Vc - V_B}{T_c - TB} + VB - TB. \frac{Vc - V_B}{Tc - TB}$

$$T = \frac{Tc - TB.}{Vc - V_B} \left(V - V_B + TB . \frac{Vc - V_B}{Tc - TB} \right).$$

L'organigramme de cet exemple est donné à la figure 22 .

fig 11- Organigramme de l'horloge Minute-Seconde.

max. 99 minutes 59 secondes
(environ $1^H 40 min$)

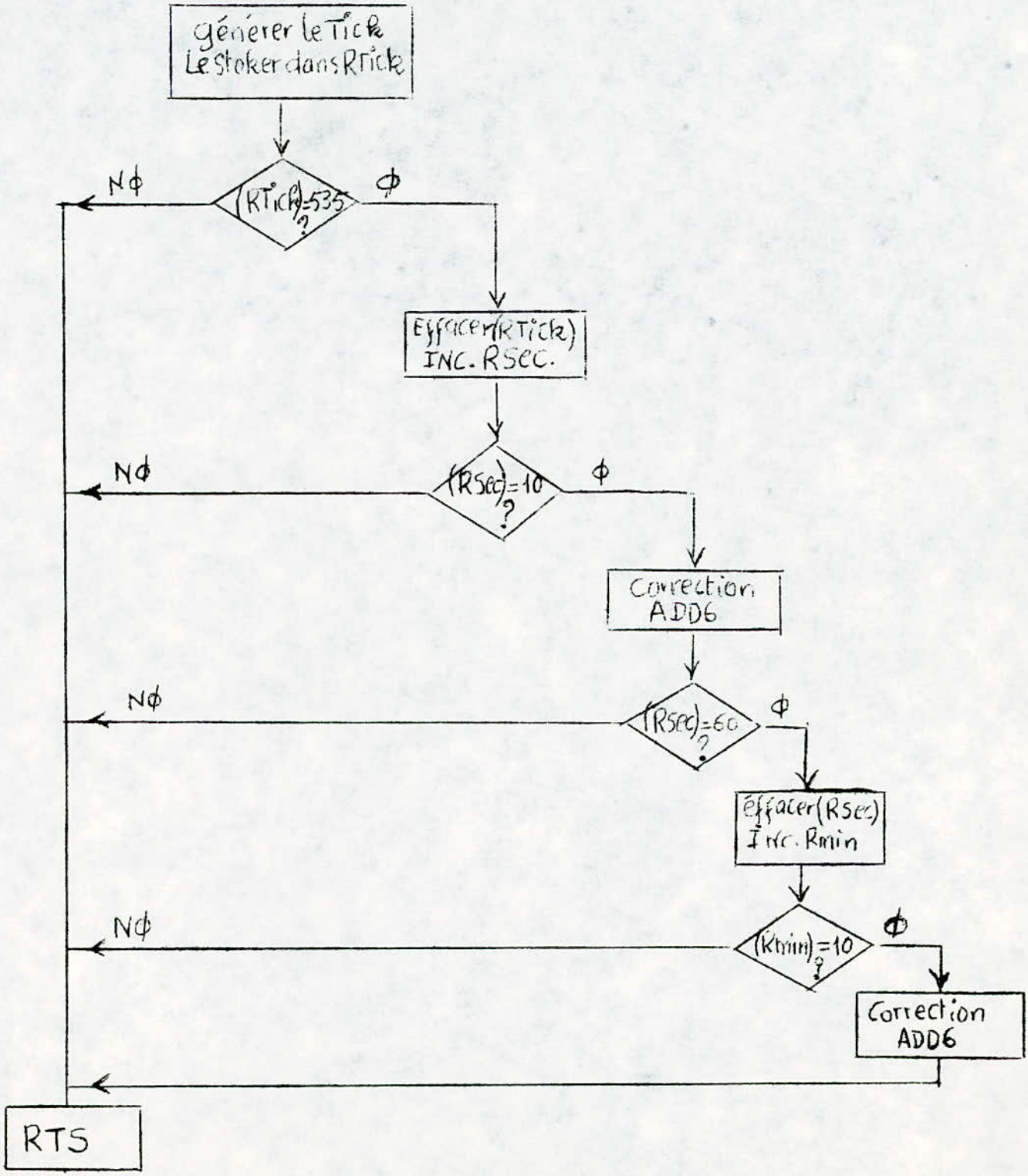


fig 16 - Organigramme "DATA". Ordre à la prise en compte de la donnée, sa reconstitution en sa vraie valeur en DCB et son stockage; Puis initialisation du PIA (port B).

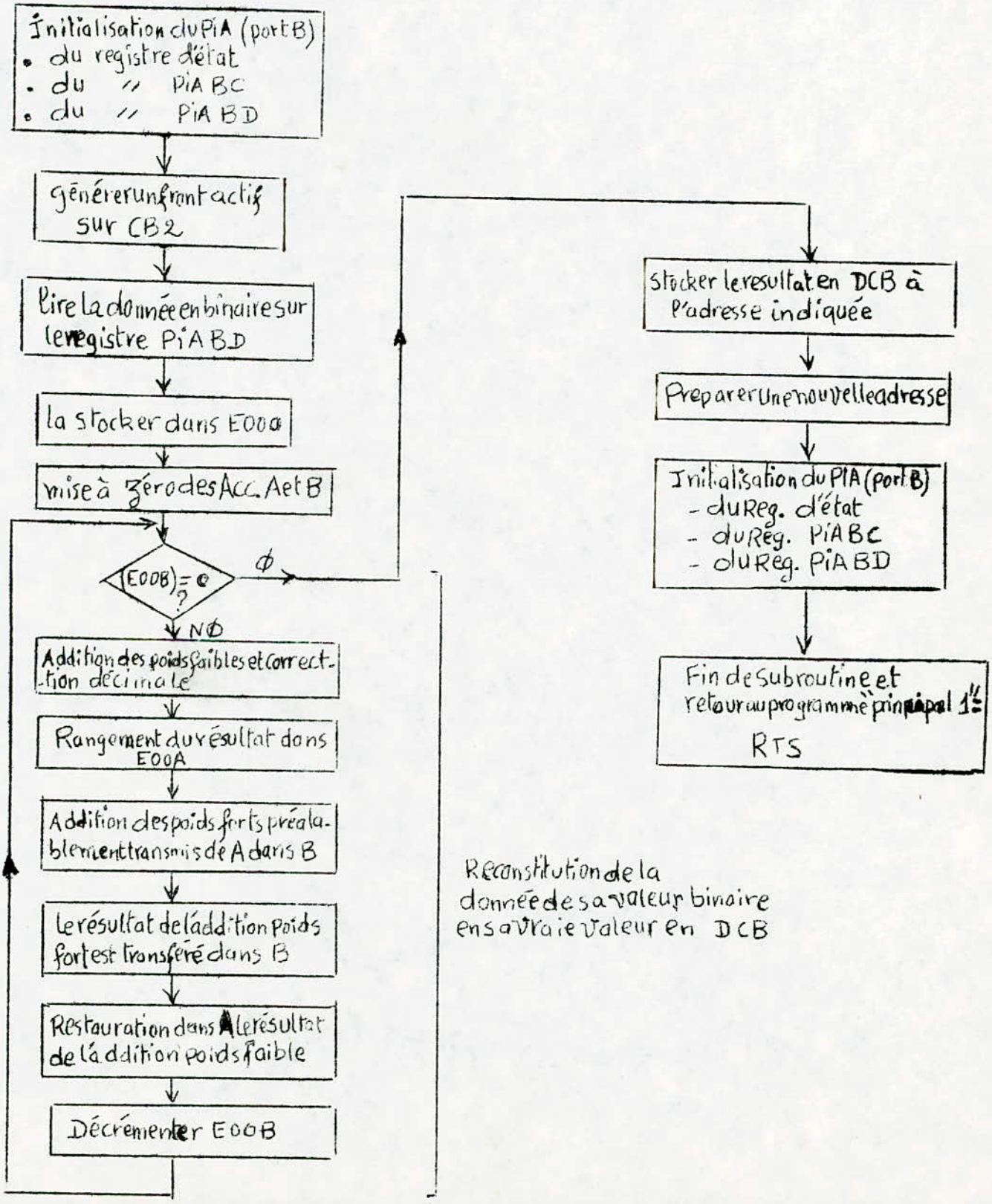


fig 18 - Organigramme principal 1 de lecture de la donnée à partir du DATA BUS du Convertisseur A/N jusqu'à son stockage en sa Vraie Valeur en DCB avec un délai "T"

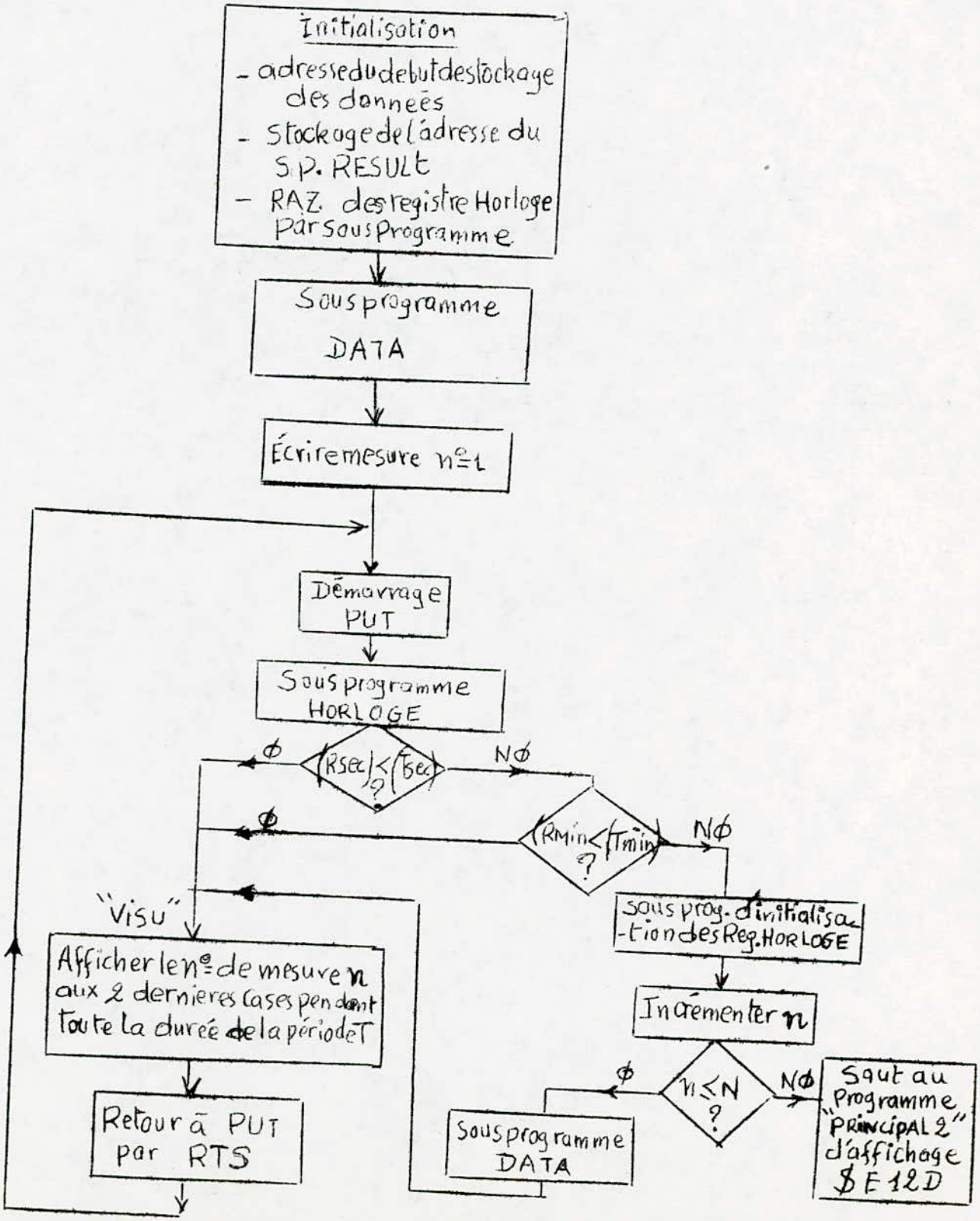
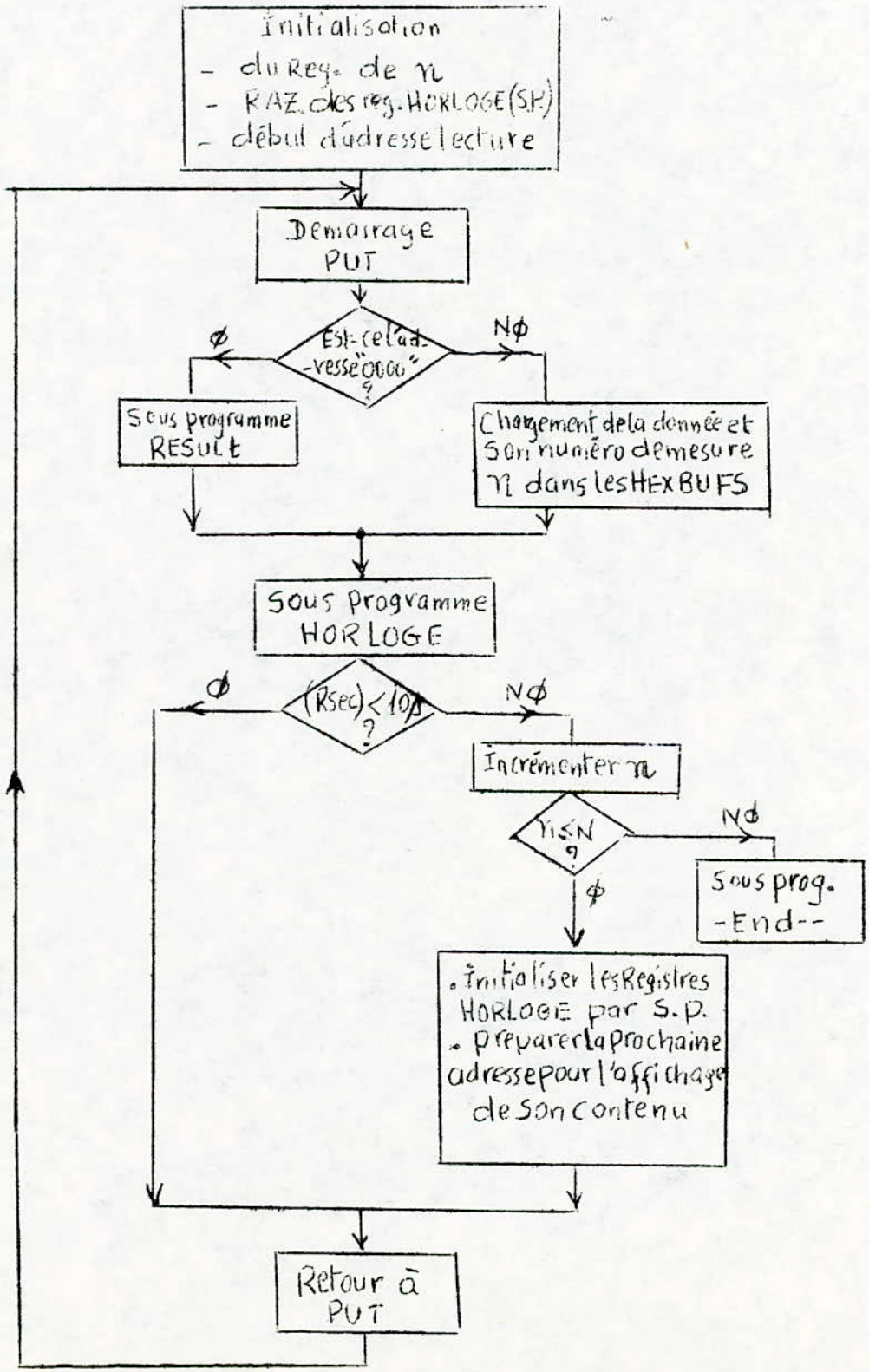


fig 20- Organigramme principal 2
d'affichage de données uneparune



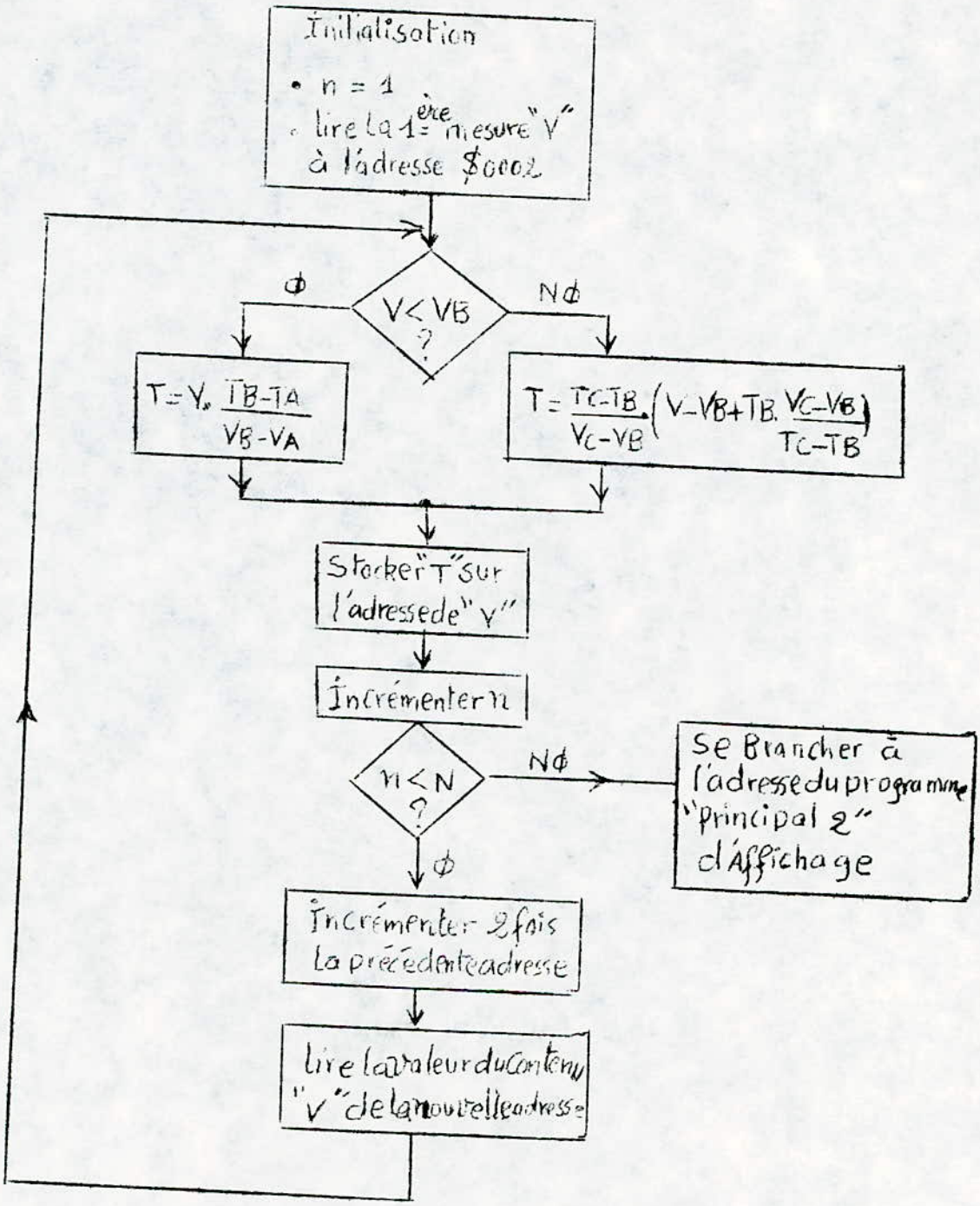


fig 22 - Organigramme de traitement de données pour une sonde de température

* RÉSERVATION DES CASES MÉMOIRES :

0000 } - pour l'adresse du sous-programme **RESULT**
0001 }

E000 - pour RMinute

E001 - pour RSeconde

E002 } - pour RTick
E003 }

E005 - début d'adressage (0002) pour IX du programme principale 1

E007 - pour l'enregistrement du nbre de mesures "n".

E00A } - pour la reconstitution de l'information de sa valeur
E00B } binaire en sa valeur décimale

E00C } - pour l'adresse à visualiser son contenu. Réservé au
E00D } prog. principal 2. L'adresse $0000 contient le S.P. RESULT.

* VARIABLES :

E004 : "N" nbre de mesures que l'on veut effectuer ou lire (en écriture HEXADECIMALE jusqu'à FF=256).

E008 : TMin } période de la série de mesures que l'on
E009 : TSec } veut se fixer.
En écriture DECIMALE (valeur de l'heure proprement dite)

fig 12. Sous Programme HORLOGE

$ E00E

```
E00E  FE E002        LDX  RTick      Enregistrer le
E011  08             INX             Tick de
E012  FF E002        STX  RTick      l'horloge.
E015  8C 0217        CPX# 535 ($0217) y a-t-il 1 Seconde?
E018  26 33          BNE .RETOUR.
E01A  CE 0000        LDX # 0000
E01D  FF E002        STX  RTick      RAZ de RTick.
E020  7C E001        INC  RSec.      Enregistrer 1 seconde.
E023  B6 E001        LDAA RSec.
E026  8A F5          ORA # F5        Est-ce qu'on a 10 sec?
E028  43             COM A
E029  26 22          BNE .RETOUR.
E02B  B6 E001        LDAA RSec.
E02E  8B 06          ADDA # 06       correction ADD6.
E030  B7 E001        STA A RSec
E033  81 60          CMPA # 60       Est-ce qu'on a 60 sec.?
E035  26 16          BNE .RETOUR.
E037  7F E001        CLR  RSec.      Effacer RSec.
E03A  7C E000        INC  RMin.      Enregistrer 1 minute.
E03D  B6 E000        LDAA RMin.
E040  8A F5          ORA # F5        Est-ce qu'on a 10 min.?
E042  43             COM A
E043  26 08          BNE .RETOUR.
E045  B6 E000        LDAA RMin.
E048  8B 06          ADDA # 06       Correction ADD6.
E04A  B7 E000        STA A RMin.
E04D  39       .RETOUR. RTS          retour de Subroutine.
```

fig 13. Sous-programme RESULt

$ E04E

```
E04E  CE 7779   LDX # 7779      "RE" code 7Segments
E051  FF E41D   STX  DISBUF
E054  CE 6D3E   LDX # 6D3E      "SU"  ,,
E057  FF E41F   STX  DISBUF+2
E05A  CE 3878   LDX # 3878      "Lt"  ,,
E05D  FF E421   STX  DISBUF+4
E060  39        RTS             Retour de Subroutine.
```

fig 14. Sous-programme -End--

$ E061

```
E061  CE 4079   LDX # 4079      "-E" code 7 Segts
E064  FF E41D   STX  DISBUF
E067  CE 375E   LDX # 375E      "nd"  ,,
E06A  FF E41F   STX  DISBUF+2
E06D  CE 4040   LDX # 4040      "--"  ,,
E070  FF E421   STX  DISBUF+4
E073  CE F0A2   LDX # DIDDLE    adresse de retour à
E076  FF E419   STX  MNPTR       l'affichage
E079  7E F0BB   JMP  PUT
```

fig 15 - Sous-programme d'initialisation des registres de l'Horloge

$ E07C

```
E07C  7F E000   CLR  RMin
E07F  7F E001   CLR  RSec
E082  7F E002   CLR  E002   } RTick
E085  7F E003   CLR  E003   }
E088  39        RTS
```

fig 17. Sous-programme DATA

$ E089

```
E089  B6 E482        LDAA PIABD    - initialisation du PIA port B.
E08C  4F             CLRA
E08D  B7 E483        STAA PIABC
E090  B7 E482        STAA PIABD
E093  C6 34          LDAB # 34     - Générer le front montant sur CB2.
E095  86 3C          LDAA # 3C
E097  F7 E483        STAB PIABC
E09A  B7 E483        STAA PIABC
E09D  B6 E482        LDAA PIABD    - prendre l'information.
E0A0  B7 E00B        STAA E00B
E0A3  4F             CLRA
E0A4  5F             CLRB
E0A5  B1 E00B        CMPA E00B
E0A8  27 14    .BCL2. BEQ  ST1
E0AA  8B 19          ADDA # 19     - valeur du LSB du C.A/N
E0AC  19             DAA
E0AD  B7 E00A        STAA E00A
E0B0  17             TBA
E0B1  89 00          ADC  # 00
E0B3  19             DAA
E0B4  16             TAB
E0B5  B6 E00A        LDAA E00A
E0B8  7A E00B        DEC  E00B
E0BB  7E E0A8        JMP  BCL2
E0BE  FE E005  -ST1- LDX  E005     - Stockage de notre information (en DCB) à l'adresse indiquée.
E0C1  A7 01          STAA 01,X
E0C3  17             TBA
E0C4  A7 00          STAA 00,X
E0C6  08             INX           - préparer une nouvelle adresse.
E0C7  08             INX
E0C8  FF E005        STX  E005
E0CB  B6 E482        LDAA PIABD    - initialiser de nouveau notre PIA.
E0CE  4F             CLRA
E0CF  B7 E483        STAA PIABC
E0D2  B7 E482        STAA PIABD
E0D5  39             RTS           - Retour au prog. principal 1.
```

fig 19 - PROGRAMME PRINCIPAL 1

$ E0D6

——— o ——— Initialisation

```
E0D6   CE 0002     LDX # 0002      - début d'adresse pour
E0D9   FF E005     STX  E005         stockage des données.
E0DC   CE E04E     LDX # E04E      - adresse du S.P. RESULT
E0DF   FF 0000     STX  0000         dans 0000 .
E0E2   BD E07C     JSR  E07C       - S.P. de R.A.Z. des Reg. Horloge.
E0E5   BD E089     JSR  E089       - S.P. DATA .
E0E8   7F E007     CLR  E007
E0EB   7C E007     INC  E007       - n=1 .
```

——— o ——— démarrage

```
E0EE   CE E0F7     LDX # PROG1
E0F1   FF E419     STX  MNPTR
E0F4   7E F0BB     JMP  PUT
```

——— o ——— programme 1.

```
E0F7   BD E00E  _PROG1_ JSR  E00E      - S.P. HORLOGE
E0FA   B6 E001         LDAA RSec.
E0FD   B1 E009         CMPA TSec.      - Comparaison de
E100   2D 1C           BLT  VISU         la période
E102   B6 E000         LDAA RMin.        "T".
E105   B1 E008         CMPA TMin.
E108   2D 14           BLT  VISU
E10A   BD E07C         JSR  E07C       - S.P. de RAZ. des Reg. Horloge.
E10D   7C E007         INC  E007       - incrémenter n.
E110   B6 E007         LDAA E007
E113   B1 E004         CMPA E004       - Comparer n à N.
E116   2F 03           BLE  NEWMES
E118   7E E12D         JMP  PROG.2     - ad. du prog. principal 2
E11B   BD E089 _NEWMES_ JSR  E089       - S.P. DATA .
E11E   B6 E007 _VISU_  LDAA E007
E121   B7 E42E         STA A HEXBUF+2
E124   BD F120         JSR  DYSCOD     - décodage 7 Segments.
E127   86 3C           LDA A#% 00111100 - masque pour affichage
E129   BD F195         JSR  CLRDS        sur 2 premiers Digits.
E12C   39              RTS             - Retour à PUT.
```

fig 21 - PROGRAMME PRINPAL 2

$ E12D

——o—— Initialisation

```
E12D    BD E07C          JSR E07C        - S.P. RAZ des Reg. HORLOGE.
E130    86 00            LDAA #00
E132    B7 E007          STAA E007       - initialiser le Reg. de n.
E135    CE 0000          LDX #0000       - première adresse à
E138    FF E00C          STX E00C          lire.
```

——o—— démarrage

```
E13B    CE E144          LDX # PROG 2
E13E    FF E419          STX MNPTR
E141    7E F0BB          JMP PUT
```

——o—— programme 2

```
E144    FE E00C  .PROG2. LDX  E00C
E147    8C 0000          CPX #0000.
E14A    26 06            BNE  DONNEE
E14C    BD E04E          JSR  E04E       - S.P. RESULT
E14F    7E E165          JMP  TEST
E152    A6 00    _DONNEE_ LDAA 00,X       - Chgt de donnée, poids
E154    B7 E42C          STAA HEXBUF       fort.
E157    A6 01            LDAA 01,X       - Chgt de donnée, poids
E159    B7 E42D          STAA HEXBUF+1     faible.
E15C    B6 E007          LDAA E007       - Chgt du nº de la
E15F    B7 E42E          STAA HEXBUF+2     mesure.
E162    BD F120          JSR  DYSCOD     - décodage en 7 segts
E165    BD E00E  _TEST_  JSR  E00E       - SousProg. HORLOGE
E168    B6 E001          LDAA RSec.
E16B    81 10            CMPA #10        - période d'affichage (10/Sec)
E16D    2D 19            BLT  AFFICH
E16F    7C E007          INC  E007       - incrémenter le numéro
E172    B6 E007          LDAA E007         de la mesure.
E175    B1 E004          CMPA E004       - Est ce que n=N ?
E178    2F 03            BLE  INITIAL.
E17A    BD E061          JSR  E061       - S.P de -End-
E17D    BD E07C  _INITIAL_ JSR E07C      - S.P de RAZ des Reg. HORLOGE.
E180    FE E00C          LDX  E00C       - Préparation de la
E183    08               INX                prochaine
E184    08               INX                adresse.
E185    FF E00C          STX  E00C
E188    39       _AFFICH_ RTS             - Retour à PUT.
```

II-6 Conclusions :

. Les programmes de l'application que nous avons présentée peuvent être stockés dans une EPROM.

La zône mémoire allant de $E000 à $E009 réservée à l'initialisation et à la programmation des paramétres N et T doit être située alors dans une partie de mémoire RAM .

. Le circuit intégré "6846" étudié au chapitre 1 peut être utile dans notre application.En effet, il possède une PROM, un port d'E/S et un timer; le port B remplacera le "User PIA " ,le timer remplacera le timing par SOFT (et donc libèrera le microprocesseur de ce fonctionnement).

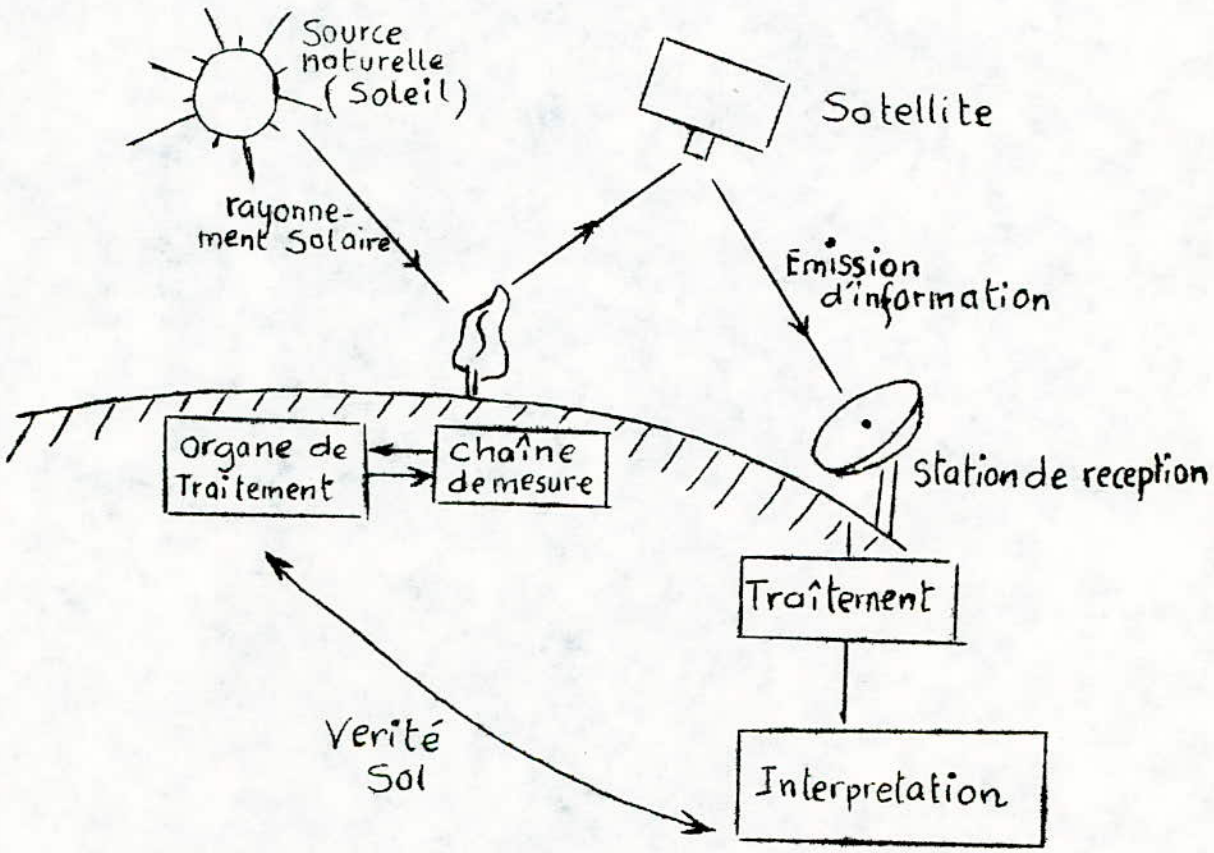
La PROM, vu sa capacité de 2K octets pourra contenir tout le programme d'acquisition et aussi plusieurs sous-programmes de traitement suivant le type de capteur utilisé . L'utilisation de ces sous programmes se fera en correspondance avec la mesure effectuée.

. Notre application peut être améliorée et étendue au domaine de la télédétection.Elle rentre plus précisément dans une aide à la mise au point de modèle de caractérisations des objets au sol par télédétection.

Le synoptique d'une chaîne de télédétection est donné ci-dessous

Les centres d'interprétation ont pour but principal d'établir une corrélation entre le rayonnement mesuré et la nature des "objets" au sol émettant ce rayonnement.

Dans le cadre de cette phase d'interprétation,on est amené à connaitre certains paramétres de l'objet en question tels que température de surface,éclairement,taux d'absorption et de réflexion... etc . L'acquisition et le traitement de ces informations sont effectués à l'aide d'un micro ordinateur.Les résultats sont transmis au centre d'interprétation où ils seront exploités.



Système de Télédetection

Conclusion

Cette étude du KIT D5 est utile pour la compréhension d'un système à microprocesseur en l'occurence un micro ordinateur.

La facilité de mise en oeuvre du KIT , son faible prix de revient et surtout ses grandes possibilités d'extension grâce à son connecteur compatible EXORCISER en font un excellent outil d'initiation voire même de travail, pour de multiples applications.

Par ailleurs, de par sa conception , nous le voyons très bien utilisé lors de travaux pratiques sur microprocesseurs dans les écoles et les universités.

Nous espérons que cet ouvrage sera très instructif pour les étudiants désirant s'initier au microprocesseur 6802 .

TABLE 3 – ACCUMULATOR AND MEMORY INSTRUCTIONS

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

LEGEND:

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

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

CONDITION CODE SYMBOLS:

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

TABLE 4 – INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

POINTER OPERATIONS	MNEMONIC	IMMED OP	IMMED ~	IMMED #	DIRECT OP	DIRECT ~	DIRECT #	INDEX OP	INDEX ~	INDEX #	EXTND OP	EXTND ~	EXTND #	IMPLIED OP	IMPLIED ~	IMPLIED #	BOOLEAN/ARITHMETIC OPERATION	5 H	4 I	3 N	2 Z	1 V	0 C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	6	2	BC	5	3				$X_H - M, X_L - (M+1)$	•	•	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	$SP + 1 \rightarrow X$	•	•	•	•	•	•

COND. CODE REG.

TABLE 5 – JUMP AND BRANCH INSTRUCTIONS

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

COND. CODE REG.

*WAI puts Address Bus, R/W, and Data Bus in the three-state mode while VMA is held low.

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

1- "EMPLOI DES MICROPROCESSEURS" ,AUMIAUX édition MASSON .

2- "MICROPROCESSEURS ET MEMOIRES" , THOMSON - EFCIS Catalogue 1980.

3- MEK 6802 D5 E - MICROCOMPUTER EVALUATION BOARD USER'S MANUAL .

4- "PROGRAMMATION DU 6800" , RODNAY ZAKS - PIERRE LE BEUX.

5- "LES MICROPROCESSEURS" ,RODNAY ZAKS - PIERRE LE BEUX. TECHNIQUES et APPLICATIONS , 3ème édition .

6- "INTERFAçAGE DES MICROPROCESSEURS" , M.ROBIN-TH.MAURIN. édition DUNOD TECHNIQUE.

7- "UN FIL D'ARIANNE"

8- MICROSYSTEMES n° 15 , article:introduction aux microprocesseurs

n° 16 , article : le microprocesseur et son environnement.

n° 8 , article : une serrure à microprocesseur SESAME 6802 .