UNIVERSITE DES SCIENCES
ET DE LA TECHNOLOGIE
U. S. T. H. B.

ECOLE NATIONALE
POLYTECHNIQUE
D'ALGER

Département d'Electronique

PROJET DE FIN D'ETUDES

Thèse d'Ingénieur en électronique

MISE EN ŒUVRE DE TRAVAUX PRATIQUES SUR LE MICROORDINATEUR LE KIT D 5 DE MOTOROLA

Proposé par :
Mlle M. ZIZI

Etudié par :
ASSI Ali

JUIN 1982

DEDICACE

A tous nos camarades dans le Mouvement National Libanais, à l'âme des martyrs de ce mouvement et de tout mouvement de libération dans le monde, je dédie ce modeste travail .

REMERCIEMENT

. Je tiens particulièrement à remercier au nom du Mouvement National Libanais, le parti du Front de Libération National qui nous a aidé, nos camarades et moi, afin de poursuivre nos études sur les bancs de l'université de ce pays frère.

. Je formule l'expression de ma profonde reconnaissance à Mesdemoiselles ZIZI Malika et KAOUA Malika, assistantes à l'ENPA, pour m'avoir guidé dans l'élaboration de ce travail.

. Que le chef de département, les enseignants de l'ENPA et de l'université de TIZI OUZOU qui ont contribué à ma formation trouvent ici l'expression de ma gratitude.

. Enfin je remercie l'UDJL pour l'aide appréciable qu'elle m'a apporté dans le tirage de ce travail.

INTRODUCTION

Le microprocesseur tend sans cesse à s'universaliser, il couvre un vaste domaine d'applications (Industrie, Médecine, gestion etc..)

Son principal avantage réside dans le fait qu'il offre une grande souplesse d'utilisation grâce à son software. D'où le souci de former des personnes qui seront aptes à manipuler convenablement les microprocesseurs .

Les microordinateurs tels que : KIT TEXAS 990C , KIT D2 et KIT D5 de MOTOROLA etc... permettront l'initiation aux microprocesseurs.

Notre travail a été réalisé au laboratoire d'électronique appliquée de l'ENPA. Il consiste en la mise en ouvre de travaux pratiques sur le KIT D5, et il servira de base à la fois théorique et pratique aux etudiants électroniciens, informaticiens, pour la compréhension des microprocesseurs.

Dans un premier chapitre nous donnerons des généralités sur le Mp 6802, principal organe du KIT D5, suivra ensuite la présentation de celui-ci dans un second chapitre.

Dans un troisième chapitre nous proposerons une série de travaux pratiques réparties en trois séries. La première série de TP familiarisera l'étudiant avec le jeu d'instructions du Mp 6802 , suivra ensuite une deuxième série de TP d'initiation à la programmation, dans une troisième série les TP seront rélatifs aux problèmes d'interfaçage d'entées/sorties avec le KIT.

Et enfin tous ceux qui auront à utiliser ou utilisent le KIT D5 trouveront dans le dernier chapitre deux possibilités d'applications du KIT : La première consiste à interfacer le KIT avec un périphérique(Table Tracante), et la deuxième application l'utilisation du KIT comme processeur musical.

TABLE DES MATIERES

CHAPITRE I

GENERALITES SUR LE MPU 6802

A - PRESENTATION DU MPU 6802

Le circuit MC 6802 est un microprocesseur monolithique 8bits contenant, outre les mêmes registres et accumulateurs que le MC 6800, un oscillateur d'horloge interne et 128 octets de mémoires RAM. La figure A.I montre le schema fonctionnel du MC 6802 on distingue :

A.I. L'unité arithmétique et logique UAL :

L'UAL est un ensemble de circuits combinatoires capables d'effectuer les opérations arithmétiques et logiques nécéssaires au traitement de l'information.

A.2. Les registres internes :

Le microprocesseur MC 6802 a trois registres de 16 bits et trois registres de 8 bits accéssibles par programme(voir fig.A.2).

A.2.I. Le compteur ordinal (Program Counter) PC :

Le compteur ordinal ou compteur programme est un registre de 16 bits qui contient l'adresse courante dans le programme.

A.2.2. Le pointeur de pile (Stack pointer) SP :

Le pointeur de pile est un registre de 16 bits qui contient l'adresse de la position disponible dans une pile externe à fonctionnement "dernier entré, premier sorti" ("Last Input, First Output" LIFO).

Cette pile est en général en mémoire lecture ecriture RAM et peut se situer à n'importe quelle adresse.

A.2.3. Le registre d'index (Index Register) IX :

Le registre d'index est un registre de 16 bits qui peut être utilisé pour des transferts de données ou comme indexe dans le mode d'adressage indexé.

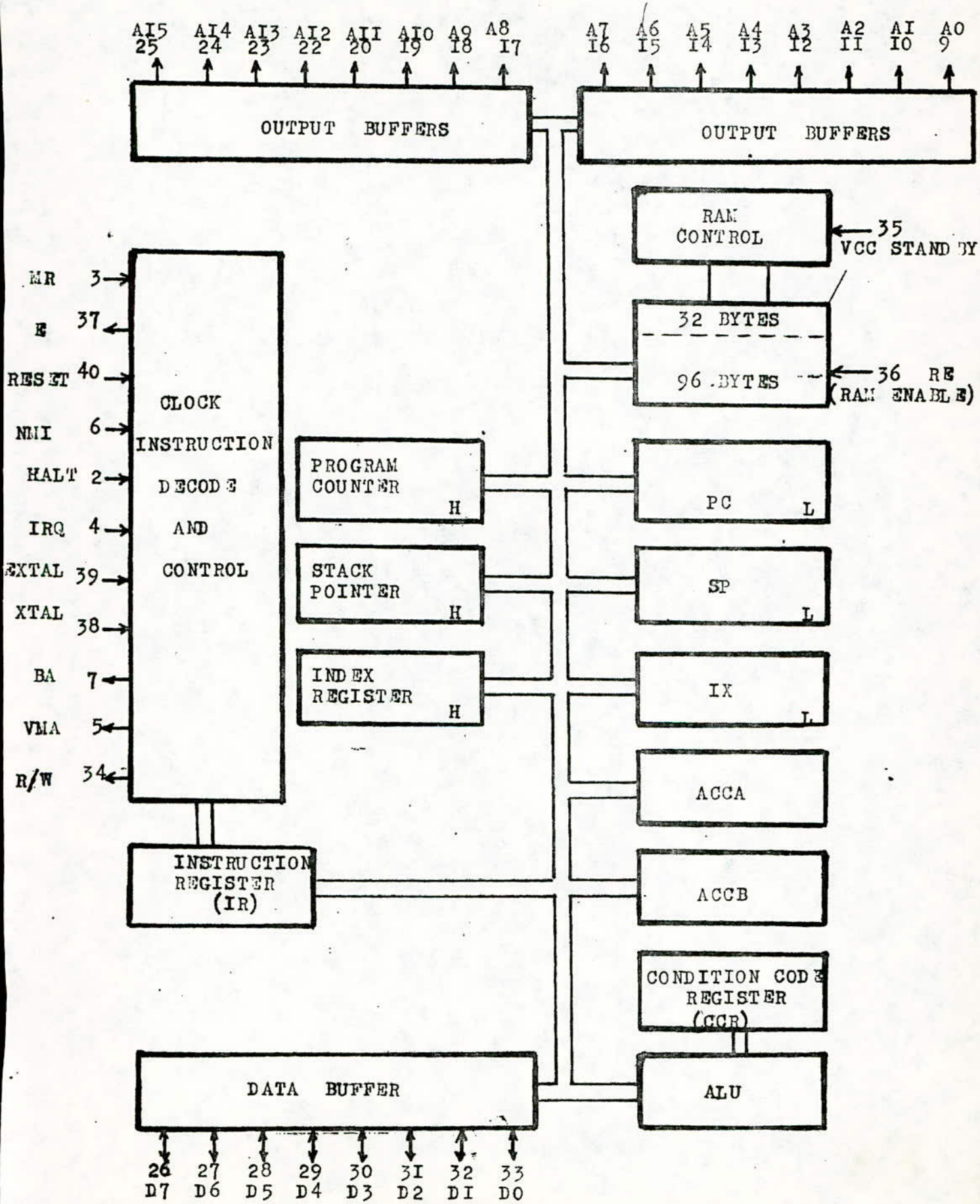


FIG.A.I STRUCTURE INTERNE DU MPU 6802.

A.2.4. Les accumulateurs (Accumulators) ACCA et ACCB :

Le MPU possède deux accumulateurs de 8 bits A et B qui sont utilisés pour contenir des opérandes ou des resultats de l'ALU.

A.2.5. Le registre code de condition (Code Condition Register):

Le registre code de condition est un registre de 8 bits. Il contient le bit masque d' interruption (I), 5 bits indiquant les resultats d'une opération de l'UAL :

CCR			H	I	N	Z	V	C

C : retenue du bit 7
V : dépassement
Z : zéro
N : négatif
I : masque d'interruption
H : demi retenue du bit 3

Les bits 6 et 7 sont à I.
La plupart des opérations executées par le MPU affectent le contenu du registre code de condition (CCR), comme par exemple les instructions de branchement conditionnel.

A.3. 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.

A.4. La RAM interne :

La RAM intégrée dans le MPU 6802 est d'une capacité de 128 octets situés entre les adresses hexadécimales 0000 et 007F Lors d'une coupure de l'alimentation les 32 premiers octets peuvent fonctionner en mode faible consommation grâce au Vcc de repos (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 controle accompagnée d'une entrée de validation (broche RE).

A.5. 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 fréquence de résonnance de IMHZ. Cependant, le diviseur par quatre integré dans le 6802 pernet l'utilisation d'un quartz de 4 MHZ.

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

B - JEU D'INSTRUCTION DU MPU :

Le 6802 possède un jeu de 72 instructions differentes, ce jeu est identique à celui du 6800.

Les instructions sont décrites en détail dans le manuel de programmation de la famille 6800.

Ce jeu conprend les instructions suivantes :

- Arithnétique binaire et décinale
- Logique
- Décalages
- Décalages circulaires
- Chargements
- Stockages
- Branchements conditionnels et inconditionnels
- Instructions de nanipulation de pile
- Instructions associées aux interruptions.

C - MODES D'ADRESSAGE DU 6802 :

Le 6802 utilise 6 nodes d'adressage :

- Adressage innédiat
- Adressage direct
- Adressage étendu
- Adressage indéxé
- Adressage inplicite
- Adressage relatif

Dans le chapitre -3- on donnera plus de détail et nous apprendrons comment utiliser ces differents nodes.

D - SIGNAUX D'ENTEE/SORTIE :

Les broches d'entrée et de sortie du MPU 6802 peuvent être reparties en quatre groupes : (voir fig.D.I et fig.D.2)

D.I. Le bus de données (Data Bus) D0 - D7 :

Le bus de données 8 bits est bidirectionnel et permet les transferts de données entre le MPU et les circuits mémoires ou périphériques.

Lorsque la RAM interne est selectionnée, le Data Bus est en "position sortie" ce qui interdit à toute information externe de rentrer dans le MPU.

D.2. Le bus d'adresse (Address Bus) A0 - AI5 :

C'est un bus unidirectionnel de I6 lignes.

D.3. Le bus de controle :

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

Le bus de controle comprend :

* 4 signaux d'etats :

  - Le signal VMA (Valid Mémory Address) : Ce signal passe à l'etat "I" lorsqu'une adresse est valide sur le bus d'adresses.

  - La sortie E (Enable $\phi 2$) de l'horloge : La broche E fournit un signal d'horloge pour le MPU et le reste du système. Ce signal, compatible TTL , est un signal à une seule phase (équivalent à la phase $\phi 2$ du 6800). Cette horloge peut être commandée par le signal MR.

  - Le signal BA (Bus Available) : Ce signal est activé lors d'une demande de DMA par le signal HALT ou lorsque le MPU est en position WAIT (attente d'interruption). Il indique alors la disponibilitée du bus d'adresses.

  - Le signal R/W (Read/Write) : Indiquant soit l'état de lecture (R/W=I) soit l'état d'écriture (R/W=0).

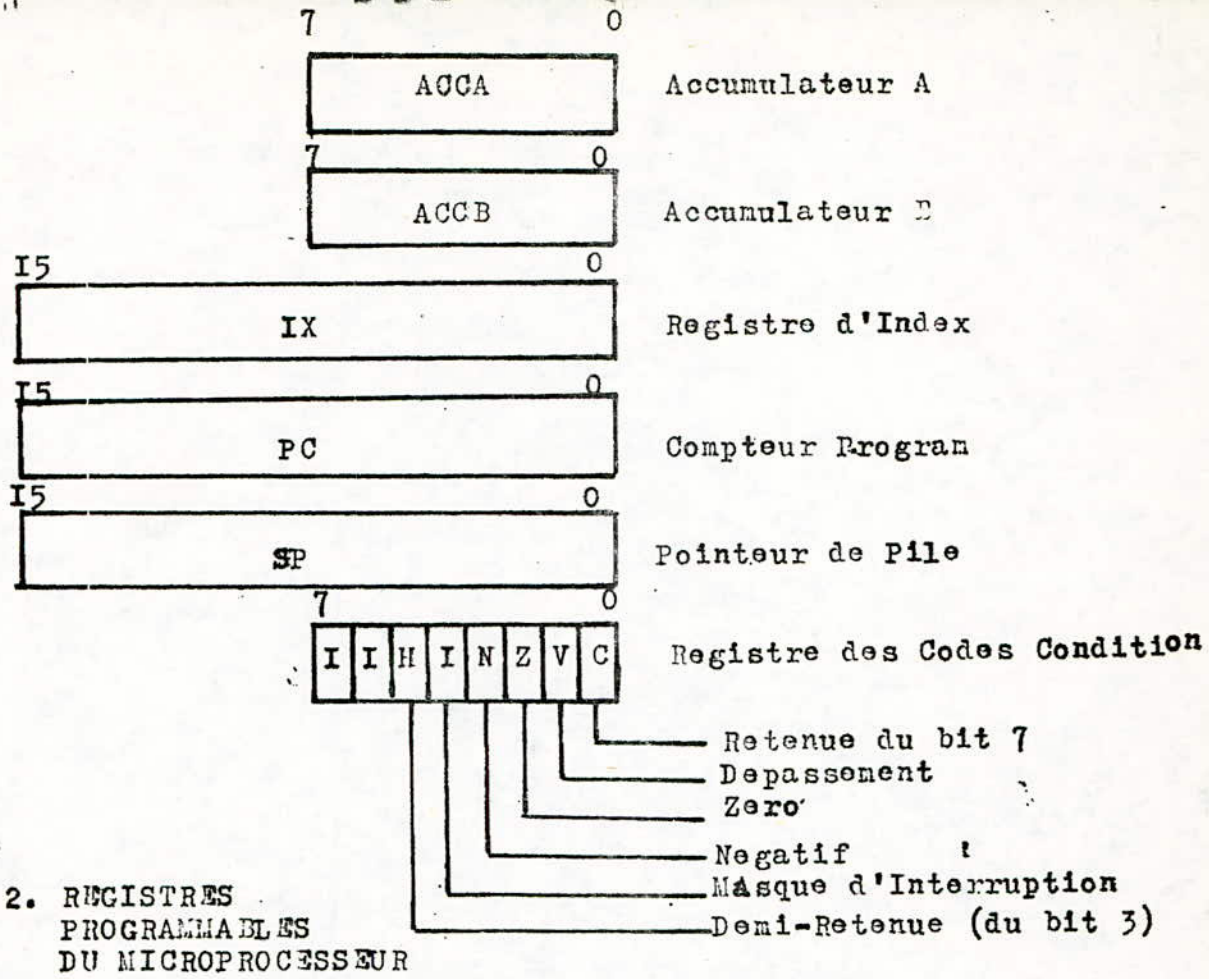


FIGURE 2. REGISTRES PROGRAMMABLES DU MICROPROCESSEUR

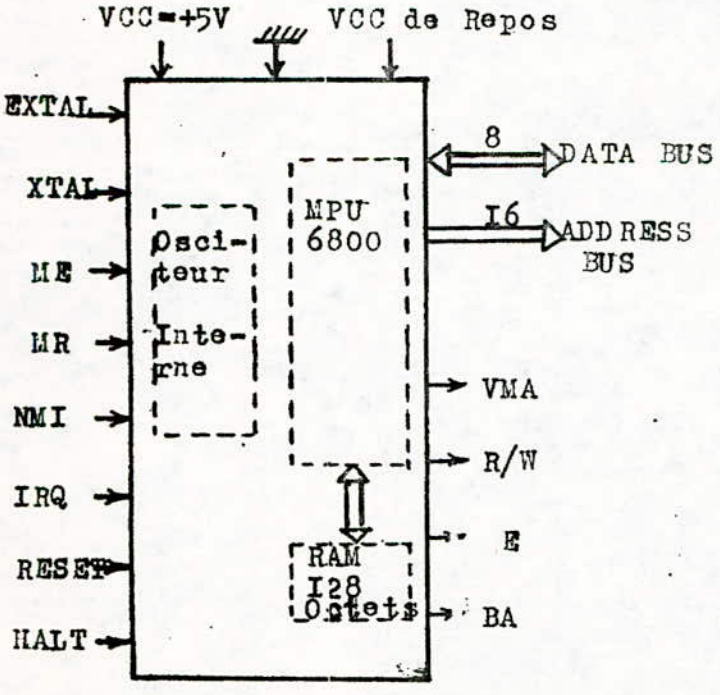


FIGURE D.2 SYNOPTIQUE GLOBAL DU MPU 6802

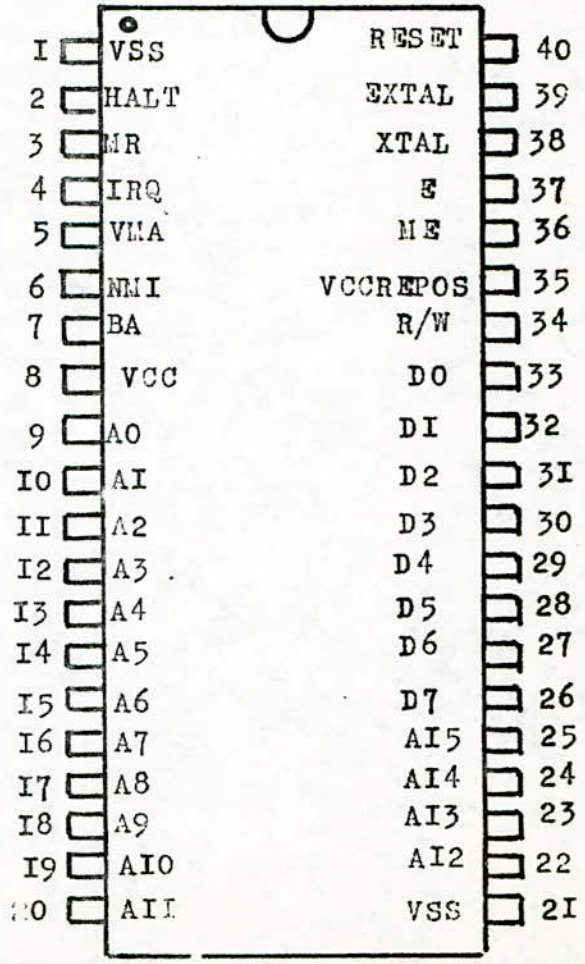


FIGURE D.3. BROCHAGE DU MPU 6802

* Le 6802 possède 2 entrées intrruption :
  - $\overline{IRQ}$: Demande d'interruption nasquable par programme.
  - $\overline{NMI}$: Interruption non nasquable.

* L'initialisation du microprocesseur se fait par le signal RESET.

* Le signal $\overline{HALT}$ permet un accès direct mémoire (DMA) par arrêt du MPU . Les bus d'adresses et de données ainsi que le signal R/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 (RAM Enable). Cette entrée compatible TTL valide par le niveau logique"I", la RAM de 128 octets integrée dans le 6802. Un niveau"0" sur cette entrée met la RAM hors circuit.

* L'entrée MR (Mémory Ready) : Ce signal, compatible TTL, permet l'allongement du signal d'horloge E.
  L'entrée MR est mise au niveau "I" logique lorsque le MPU fonctionne normalement.
  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.

La gestion du MPU s'organise donc autour de tous ces signaux.

Le 6802 possède 4 vecteurs d'interruption qui sont :

- $\overline{RESET}$ Initialisation
- $\overline{NMI}$ Interruption non nasquable
- SWI Interruption software
- $\overline{IRQ}$ Demande d'interruption

Dans le chapitre -3- ces vecteurs seront vus plus en détail lors d'un TP relatif aux interruptions.

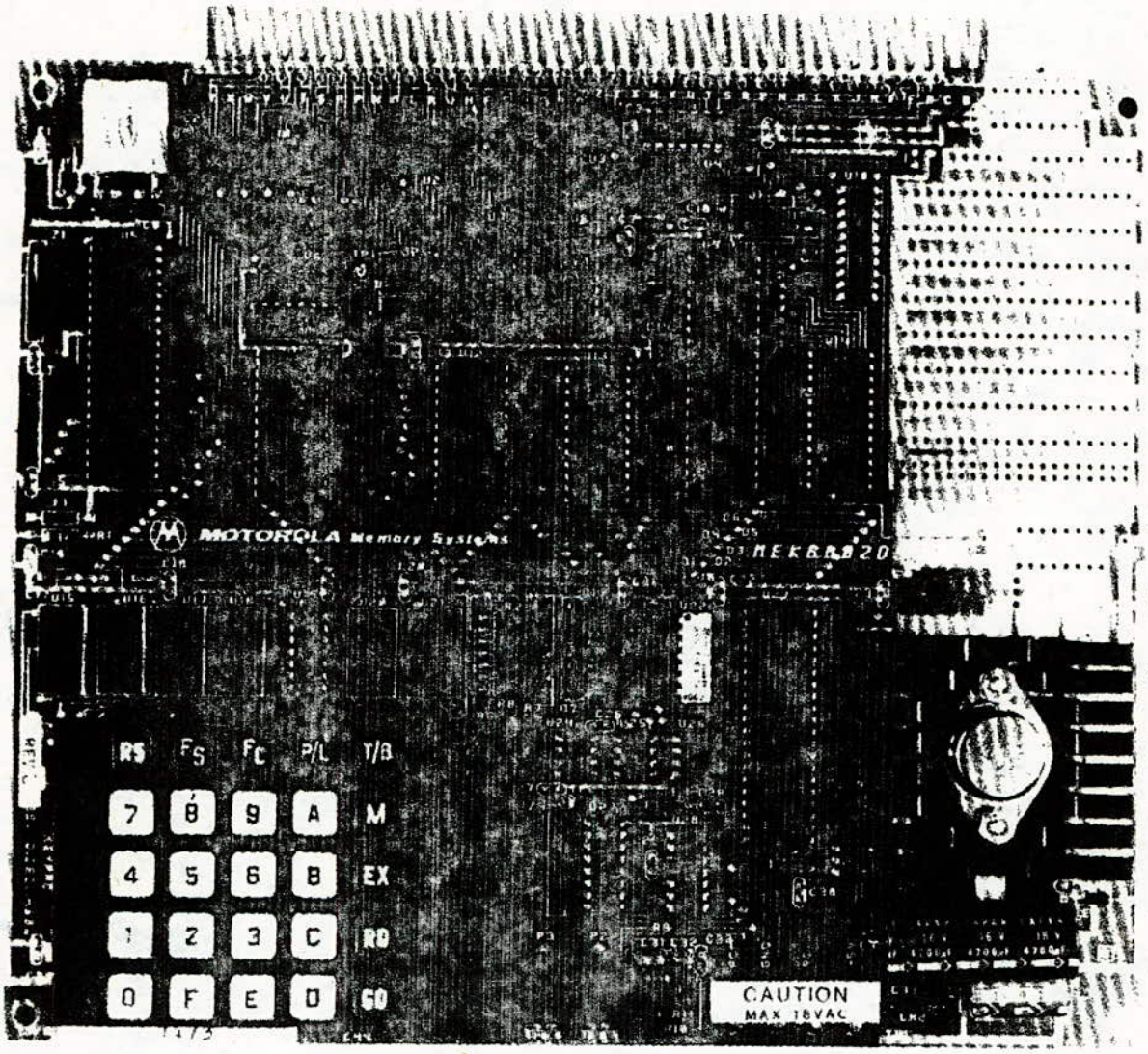
D.4. L'alimentation :

Le 6802 ne nécéssite qu'une seule tension d'alimentation Vcc = + 5V. D'où son avantage d'être compatible avec les circuits integrés 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 8nA pour Vcc standby égale à 5,25 volts.

CHAPITRE II

PRESENTATION DU KIT D5 DE MOTOROLA



MEK6802D5E MICROCOMPUTER EVALUATION BOARD

A - ORGANISATION GENERALE

Le KIT MEK 6802 D5 de MOTOROLA est un systeme de base de faible coût permettant d'utiliser les composants de la famille 6800, il est particulierment adapté pour l'apprentissage des techniques de programmation et les developpements des systemes associés aux microprocessseurs.

Le KIT D5 comprend, le MPU MC 6802, une mémoire morte la ROM D5 BUG(comprenant le moniteur), de la mémoire vive RAM et des organes d'éntrée/sortie. La communication avec l'utilisateur se fait à l'aide d'un clavier à 25 touches en entrée et des afficheurs en sortie. Le language utilisé est l'assembleur codé hexadécimal.

Le KIT D5 est constitué par differents blocs fonctionnels(voir fig.2.I) on presentera successivement les differents blocs :

- Le bloc de commande de tout le systeme:Le MPU 6802
- Le bloc du KEYPAD, du SYSTEM PIA(U23), des AFFICHEURS et de l'interface CASSETTE
- Le bloc à mémoires mortes(ROM)
- Le bloc à mémoires vives (RAM)
- Le décodeur d'adresses (U6)
- Le bloc interface serie
- Le bloc interface parallèle"USER PIA" (le MC 6821 "U9")
- Le bloc BUFFERS (U1, U2,et U3 en option)

B - PRESENTATION DES BLOCS FONCTIONS

La présentation des differents blocs est donnée par la figure 2.I et les diagrammes shematiques (voir annexe)

B.I. Bloc de commande de tout le systeme : Le MPU 6802

(pour une etude plus detaillée voir chapitre.I.)
Toutes les entrées de commande d'interruption et RESET sont à l'état haut.
La broche "MR" est dans son état de repos (+5V).
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 4 = 0,895 MHZ ; celle ci correspond à une période d'horloge E(Ø) = I,II7 .
Le VCC Standby est relié directement à VCC = + 5V, ce qui ne

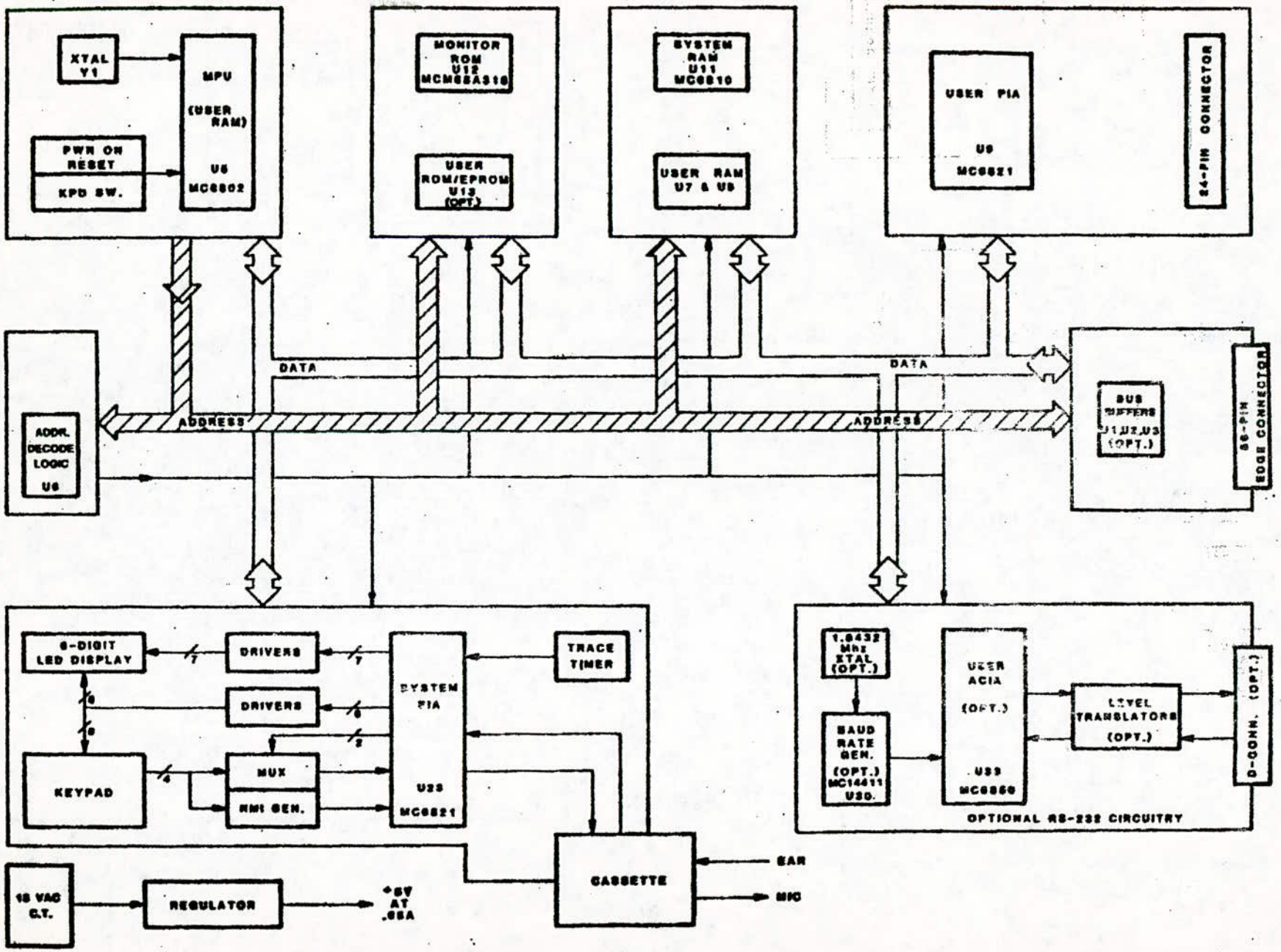


FIGURE 1- BLOCK DIAGRAM

permet pas la sauvegarde des premiers octets de la RAM interne une fois l'alimentation coupée. La RAM interne peut être supprimée par le Strap EI en cas d'utilisation de son adressage pour des circuits externes au KIT .

B.2. Bloc du KEYPAD, du SYSTEM PIA, 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-segments 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" (UI9B), le décodeur de position ou multiplexeur "MUX" (U27) et les drivers en 6 portes NAND (U24A, U24B, U25A, U25B, U26A, U26B).

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 .

Les 6 premières lignes du port B sont reliées à travers des drivers 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 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" = I200 Hz

fréquence du "I" = 2400 Hz

Le "TRACE TIMER" (U29A) sert à déclencher une interruption sur la ligne de commando CB2 . Ce circuit (74LS393) est un compteur qui est synchronisé par l'horloge E(Ø) du système . Les sorties A, B et C sont non connectées, de ce fait il constitue un diviseur de fréquence par I6 (sortie D).

B.3. Bloc à mémoires mortes (ROM) :

C'est un bloc de mémoire à lecture seule dont la "D5 BUG" (UI2) de 2 K octets est indispensable pour la gestion du KIT .

L'"USER ROM" (EPROM "UI3") qui peut aller jusqu'à 2K 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 IK octets, tri-tensions(+5V, -5V, +I2V) .
- soit une 27I6 de 2K octets, monotension (+5V) .

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

B.4. 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 68I0 "U") composée de I28 octets et l'"USER RAM" divisée en boîtiers (U7 + U8) du type MC 2II4 de capacité IK X 4 bits .

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 troisième partie est prévue pour assurer le rôle d'unepile à l'usage de l'utilisateur .

Les 2 RAMs U7 et U8 sont constituées par des cases mémoires de 4 bits . En étant placées en parallèle , elles forment le mot complet de 8 bits .

B.5. Décodeur d'adresses (U6) :

C'est grâce à ce circuit logique du type 74LSI56 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 .

B.6. Bloc interface serie :

Ce bloc en option sur le KIT est composé d'un interface série l'"USER ACIA" (U33) et d'un générateur de fréquence (U30) . Sur notre carte, il pourra être utilisé pour des applications d'Entrée/Sortie en série (télétype, transmission avec MODEM etc ...)

La synchronisation s'effectue par l'horloge "U30"(qui génére plusieurs fréquences) .

B.7. Bloc interface parallèle"USER PIA"(MC 6821 "U9") :

C'est un interface prévu spécialement pour l'utilisateur et son usage se rapporte à faire la liaison entre le MPU et les périphériques (convrtisseur, imprimante etc ...).
Son connecteur à 24 pins nous sera trés utile pour des connections rapides.

B.8. Bloc BUFFERS (U1, U2, et U3 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 .

C- IMPLANTATION MEMOIRE :

La répartition des adresses sur la carte pour chaque boîtier (RAM, ROM;PIA, ACIA) est donnée à la figure.2.2.
Les premieres adresses de $ 0000 à $ 007F sont réservées pour la RAM interne du 6802 . Les adresses de $ 0080 à $ DFFF (entout 56 K bytes) sont utilisées pour l'adressage des circuits externes au KIT . Il est possible de supprimer la RAM interne et de juxtaposer 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.
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 $ F7FF et l'"USER ROM" optionnelle pour les adresses allant de $ E800 à $ EFFF .
Les zônes mémoires $ E488 à $ E6FF , $ E702 à $ E7FF et $ F800 à $ FFFF sont appelées zônes "miroirs" et ne doivent pas être utilisées .

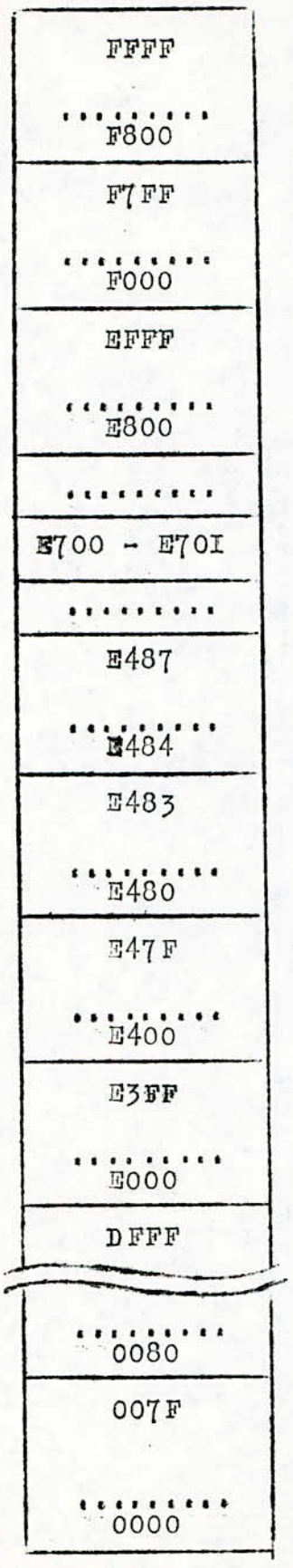


FIG.2.2 MEMORY MAP (Implantation ménoire)

D - COMMANDES DU KIT D5 :

Le clavier dispose de I6 touches hexadecimales 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 fonctionde chacune de ces touches :

D.I. Reset (RS) :

Pour "Reset" remettre à zéro le système entier, enfoncer la touche "RS" et l'affichage résultant sera un tiret dans l'afficheur de gauche.

D.2. Escape (EX) :

Perte du contrôle programme : le programme tourne sur lui-même. Normalement il est possible d'arrêter un programme utilisateur en enfonçant la touche "EX" . Cependant puisque cette fonction dépend de l'état du PIA du clavier; il est possible à un programme utilisateur d'interdire la fonction "ESCAPE" .

Si la touche "EX" est enfoncée, alors qu'un programme utilisateur tourne, et que la fonction "ESCAPE" est active, l'affichage donnera l'état du compteur ordinal d'utilisation courante et le système sera séquence d'édition d'affichage de registres .

Si le système était en train d'exécuter un programme moniteur lorsqu'on a appuyé sur "EX" l'affichage résultant est un tiret dans l'afficheur de gauche .

D.3. Divers :

Lorsque l'on rentre des informations, l'entrée du premier chiffre remet à zéro les registres modifiés et le nouveau chiffre est introduit dans les 4 bits de poids faibles du registre. Les entrées suivantes se font par la droite et les nombres sont décalés vers la gauche. Il n'y a pas de limite au nombre de chiffres entrées. Quand la touche de commande suivante est enfoncée, la valeur prise en compte par le microprocesseur est celle qui est affichée à cet instant.

D.4. Echange entre mémoire et affichage :

Pour afficher ou changer le contenu d'un mot mémoire, il faut d'abord entrer l'adresse du mot à changer. Les zéros en tête ne sont pas nécessaires et les chiffres peuvent être entrés tant que l'affichage est correct. Quand l'adresse est

complète, appuyer sur la touche "M" . Les quatre premiers chiffres indiquent l'adresse du mot mémoire sur lequel on travaille. Les cinquième et sixième afficheurs donnent le contenu actuel du mot mémoire adressé, et si l'affichage ne correspond pas à la donnée entrée, cela signifie que l'adresse choisie ne peut pas accepter de données (zone ROM moniteur) .

Pour passer au mot mémoire suivant, appuyer sur "GO". Pour revenir en arrière au mot mémoire précédent, appuyer sur "M" .

D.5. Calcul de decalage (déplacement) :

Pour calculer un déplacement relatif, appuyer sur"FS" après l'affichage de la mémoire; L'adresse mémoire affichée au moment où l'on a appuyé sur "FS" est considérée comme l'adresse du mot déplacement. L'affichage en cet instant sera un A dans l'afficheur le plus à droite, précédant l'entrée de l'adresse de destination.

Après que cette adresse ait été entrée, appuyer sur "GO". LE déplacement sera calculé et vérifié pour voir s'il peut s'exprimer sur 8 bits. Le résultat sera affiché dans les deux afficheurs de droite, s'il est correct (possible). Sinon le mot "BAD" indique que le déplacement est trop grand pour être exprimé sur 8 bits.
Noter ce déplacement sur le listing avant de continuer.

Appuyer sur "GO". Le déplacement calculé est mis en mémoire et le contrôle revient au programme d'échange mémoire/affichage. L'adresse mémoire affichée, sera celle suivant immédiatement l'octet de déplacement mis en mémoire.

Si pour quelque raison le déplacement calculé est mis en mémoire et le contrôle revient au programme d'échange entre mémoire et affichage. L'adresse mémoire affichée sera la même que celle où on a lancé le programme de calcul de déplacement. Si le calcul n'était pas possible(affichage BAD), appuyer sur"M" et le moniteur reprendra les procédures d'échange entre mémoire et affichage.

D.6. Affichage-Changement des registres :

Pour faire afficher les registres, appuyer avant ou après le début ou la fin d'un programme, ou après un point d'arrêt (Microprocesseur à l'arrêt registres stables), appuyer sur "RD". Le premier registre affiché sera le compteur ordinal (pointeur de programme utilisateur).

Pour passer au registre suivant, appuyer sur "GO" et pour revenir en arrière au registre précédent, appuyer sur "M". Les registres se suivent cycliquement : si on appuie sur "GO" alors que CC est affiché, l'affichage donnera alors le PC (pointeur de programme).

Pour changer le contenu d'un registre, afficher le registre voulu et entrer la valeur désirée.

PC	PC	PC	PC	P	C	Compteur ordinal PC
		a	a		A	Registre A(accumulateur)
		b	b		B	Registre B(accumulateur)
X	X	X	X	I	d	Registre d'index Id
SP	SP	SP	SP	S	P	Moniteur de pile SP
		c	c	C	C	Registre de contrôle CC

D.7. Programme en pas à pas :

Pour faire exécuter une instruction unique par le MC 6802, appuyer sur "T/B" à partir d'une séquence d'affichage des registres. Après l'exécution de cette unique instruction, le moniteur reprend le contrôle dans une séquence d'affichage des registres et tous les registres contiennent les résultats dus à l'instruction exécutée. On peut appuyer "T/B" aussi souvent qu'on le désire pour faire exécuter plusieurs instructions .

D.8. Lancer un programme utilisateur :

Pour lancer un programme utilisateur, on commence au point d'arrêt (un tiret dans l'afficheur le plus à gauche), on entre l'adresse de départ de programme et on appuie sur "GO" .

D.9 Continue :

C'est la même chose que "GO" pour les programmes utilisateur, mis à part qu'au lieu de spécifier l'adresse de départ, l'adresse de départ prise en compte est celle contenue dans le registre compteur-ordinal. Pour utiliser cette fonction, appuyer sur "GO" à partir du point d'arrêt(tiret). Il n'est pas nécessaire d'entrer plusieurs nombres avant d'appuyer sur "GO" pour passer en fonction "CONTINUE".

D.I0. Point d'arrêt :

Pour passer en mode "pose de points d'arrêt", appuyer sur "FS" puis sur "T/B" à partir du tiret de départ. Pour quitter ce mode de fonctionnement, appuyer sur "EX". Pour avancer jusqu'au point d'arrêt suivant, appuyer sur "GO". La fonction d'avance tourne cycliquement à chaque fois qu'on appuie sur "GO". On peut placer jusqu'à cinq points d'arrêt simultanés.

Pour supprimer un point d'arrêt existant, avancer jusqu'à l'adresse où il est affiché et appuyer sur "FC". Le dernier afficheur "nombre de points d'arrêt" sera décrémenté et un autre point d'arrêt sera affiché. Pour supprimer tous les points d'arrêt appuyer sur "FC" autant de fois qu'il le faut, pour que le nombre de points d'arrêt soit à zéro .

Pour placer un nouveau point d'arrêt, entrer son adresse hexadécimale puis appuyer sur "FS". Les points d'arrêt ne peuvent être placés qu'à des adresses situées en RAM. Si le nombre de points d'arrêts ne change pas, cela signifie que le point d'arrêt (# 3F) ne peut pas être mis en mémoire à l'adresse proposée .

D.II. Enregistrement :

Pour enregistrer sur une cassette, commencer au point de départ (tiret à gauche); Appuyer sur "P/L" et l'affichage devient 0000 bb, demandant une adresse de départ.

Par exemple, presser sur I, 2, 3, l'affichage devient 0I23 bb ; Appuyer sur "GO" pour entrer cette adresse. L'affichage est maintenant 0000 EE, comme pour demander l'adresse de la dernière donnée à enrigistrer. Par exemple, appuyer sur 4, 5, 6, 7. L'affichage est 4567 EE . Mettre le magnétophone en route sur le mode enrigistrement. Appuyer sur "GO". Une suite de mots #FF durant 30 secondes, va précéder le début du transfert de données(de la mémoire à la cassette).

Quand la commande d'enregistrement est terminée le microprocesseur revient au point de repos (tiret) et la D5 est prête pour la commande suivante.

D.I2. Chargement à partir de la cassette- Verification ;

La commande de chargement est "FS" puis "P/L". La commande de vérification est : "FS" puis "RD".

Quand le chargement ou la vérification est achevé et sans erreur, le microprocesseur se remet au repos (tiret) et est prêt à exécuter la commande suivante.

Si une erreur apparait pendant le chargement ou la vérification, le message "FAIL ??" apparait sur les afficheurs. L'utilisateur peut intervenir à ce point et faire afficher les registres de façon à rechercher l'erreur. L'adresse où est apparue l'erreur se trouve dans le registre d'index. De même;la donnée contenue dans le registre A (accumulateur) est le dernier mot-mémoire lu sur la bande enregistrée.

D.I3. Fonctions utilisateur :

Des fonctions utilisateur peuvent être obtenues en utilisant la touche de mise en place de fonctions et le pointeur de fonction (FNCPNT).

Les fonctions d'utilisateur sont appelées en appuyant sur "FS" suivi d'un chiffre hexadécimal.

Les fonctions utilisateur sont créés en formant un tableau d'adresses de fonctions fournies à l'utilisateur et en mettant l'adresse de début de tableau en mémoire dans le FNCPNT.

On peut ajouter jusqu'à I6 fonctions à celles existant déja sur la carte D5 .

E - ADRESSES DES PROGRAMMES UTILES DU D5 BUG

Nom du programme	Adresse dans D5 BUG	DESCRIPTION
RESET	$ F000	Remise à zéro initiale
PROMPT	$ F024	Départ après initialisation
GET	$ F04E	Programme de lecture clavier (emploi la procédure d'interruption non masquable et ainsi un utilisateur n'emploiera pas ce programme directement).
PUT	$ F0BB	Affiche les données sur les afficheurs 7 segments et appelle les programmes en fonctionnement.
DYSCOD	$ FI20	Décode l'hexadécinal pour l'affichage sur 7 segments.
DLY 25	$ FI69	Temporisation de 25 millisecondes.
DLY I	$ FI7I	Temporisation de I millisecondo.
DLY X	$ FI79	Temporisation réglable avec le registre X I/2 S au maximum.
ADDAX	$ FI83	Additionne le registre A au registre X.
CLRDS	$ FI95	Supprime l'affichage de certains afficheurs par un masque contenu dans le registre.
ROLL 2	$ FIAA	Programme d'entrée numérique de deux valeurs de données(I octet) l'adresse de l'operation est spécifiée dans "HEXBUF".
ROLL 4	$ FICC	Programme d'entrée numérique de 4 données (2 octets).
RDKEY	$ FIEF	Lit reconnait les touches du clavier.
TIN	$ F533	Lit un octet à partir d'un magnéto-cassette.
PNCHB	$ F630	Formate et fait enregistrer une file entière.
LOAD	$ F69C	Charge une file entière(FNCFL≠0). Vérifie une file entière(FNCFL=0).

Les programmes relatifs à l'interface cassette nécessitent des durées de séquencement précises.

CHAPITRE III

TP SUR LE KIT D5

TP I : PRELIMINAIRES

Dans ce prenier TP nous allons nous familiariser avec les principales touches du KIT.

Nous vous proposons dans un début un programne qui consiste en "la visulisation d'un not" (de 6 lettres au nax) sur des afficheurs 7 segnents .

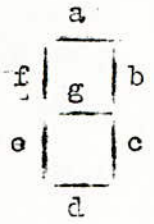
Le fornat général d'un listing pour un programne donné se présente conne suit :

Adresses du programne	Instructions en language nachine	Mnenoniques et données codées hexadécinal	Conmentaire
0000	86 77	LDA A # $77	Chargement des lettres du not à visualiser dans les buffers.
0002	B7 E4 ID	STA A DISBUF	
0005	86 6D	LDA A # $6D	
0007	B7 E4 IE	STA A DISBUF+I	
000A	86 6D	LDA A # $6D	
000C	B7 E4 IF	STA A DISBUF+2	
000F	86 30	LDA A # $30	
00II	B7 E4 20	STA A DISBUF+3	
00I4	86 40	LDA A # $40	
00I6	B7 E4 2I	STA A DISBUF+4	
00I9	86 40	LDA A # $40	
00IB	B7 E4 22	STA A DISBUF+5	
00IE	CE F0 A2	LDX #/DIDDLE	Appel du subroutine de visualisation
002I	FF E4 I9	STX HNPTR	
0024	7E F0 BB	JMP PUT	

END

Le mot à visualiser etant "ASSI --"

DISBUF, DISBUF+1,..., DISBUF+5 sont des cases memoires où sont stockés respectivement les codes des lettres A,S,S,I,-,-



Par exemple pour décoder la lettre A, les segments a, b, c, d, f et g doivent être allumés.

0	g	f	e	d	c	b	a

Le "I" logique dans un bit allume le segment correspondant.

bit non utilisé

0	I	I	I	0	I	I	I

Donc "77" est le code de "A".

Nous allons maintenant charger ce programme dans le KIT, pour cela :

- Appuyer sur "RS" pour avoir le tiret.
- Entrer la première adresse "0000" puis appuyer sur "M" le contenu de la case mémoire d'adresse "0000" est alors visualisé sur les 2 afficheurs de droite (afficheur de donnée). Entrer alors "86" qui remplace le contenu précedant.
- Pour passer à l'adresse suivante, appuyer sur "GO" 0001 est donc visualisé sur les 4 premiers afficheurs, ensuite appuyer sur "M" afin de faire entrer la donnée suivante c'est à dire "77".

  Proceder de la même manière pour faire entrer tout le programme en mémoire .

- Appuyer ensuite sur "EX", d'où la reapparition du tiret.

Avant l'execution d'un programme, il est conseillé de le verifier instruction par instruction pour cela utiliser les touches :

- "GO" pour passer d'une case mémoire (MEM) à la case mémoire suivante (MEM+I).
- "M" pour passer d'une case mémoire (MEM) à la case mémoire précedante (MEM-I).

- Pour executer le programme faites entrer l'adresse du début du programme, appuyer ensuite sur "GO". Le mot "ASSI ..." sera donc visualisé sur les afficheurs 7 segment.

Exercice I : En utilisant le même programme afficher un mot de votre choix. (voir tableau page 23)

Exercice 2 : Dans ce premier programme vous aviez utilisé l'ACCA, (pour charger les données afin de les stoker ensuite dans les DISBUFS). Vous pouviez aussi bien utiliser l'ACCB il suffit de remplacer :

LDA A # $ XX 86 XX Par LDA B # $ XX C6 XX

STA A DISBUF B7 .... Par STA B DISBUF F7 ....

Ou encore dans ce même programme utiliser les deux accumulateurs A et B .

Exercice 3 : De la même manière vous pouvez utiliser le registre d'index (IX) de I6 bits, ce qui reduit le nombre d'instructions du programme.

Donné	Codage binaire	Codage hexadécimal
	0 0 0 0 0 0 0 0	$ 00
0	0 0 I I I I I I	$ 3F
I	0 0 0 0 0 I I 0	$ 06
2	0 I 0 I I 0 I I	$ 5B
3	0 I 0 0 I I I I	$ 4F
4	0 I I 0 0 I I 0	$ 66
5	0 I I 0 I I 0 I	$ 6D
6	0 I I I I I 0 I	$ 7D
7	0 0 0 0 0 I I I	$ 07
8	0 I I I I I I I	$ 7F
9	0 I I 0 I I I I	$ 6F
A	0 I I I 0 I I I	$ 77
b	0 I I I I I 0 0	$ 7C
C	0 0 I I I 0 0 I	$ 39
d	0 I 0 I I I I 0	$ 5E
E	0 I I I I 0 0 I	$ 79
F	0 I I I 0 0 0 I	$ 7I
H	0 I I I 0 I I 0	$ 76
L	0 0 I I I 0 0 0	$ 38
I	0 0 I I 0 0 0 0	$ 30
O	0 0 I I I I I I	$ 3F
P	0 I I I 0 0 I I	$ 73
q	0 I I 0 0 I I I	$ 67
S	0 I I 0 I I 0 I	$ 6D
U	0 0 I I I I I 0	$ 3E

Le programme qui visualise le mot "ASSI --" sur les afficheurs 7 segments s'écrit comme suit : (en utilisant le RX)

Adresses du programme	Instructions en langage machine	Mnemoniques et données codées hexadécimal	Commentaire
0000	CE 77 6D	LDX # $776D	Chargement
0003	FF E4 ID	STX DISBUF	des lettres
0006	CE 6D 30	LDX # $6D30	du mot à
0009	FF E4 IF	STX DISBUF+2	visualiser
000C	CE 40 40	LDX # $4040	dans les
000F	FF E4 2I	STX DISBUF+4	buffers.
00I2	CE F0 A2	LDX #DIDDLE	Appel du
00I5	FF E4 I9	STX MNPTR	subroutine de
00I8	7E F0 BB	JMP PUT	visualisation
		END	

Faites entrer ce programme en mémoire et vérifiez si le mot "ASSI --" est visualisé à la fin de son execution.

Toujours dans le cadre de la familiarisation avec les touches du KIT nous vous proposons 2 petits programmes afin d'illustrer la touche "RD". Celle-ci permet non seulement de visualiser le contenu des différents registres du MPU mais aussi de le changer.

Programme I : Visualisation des registres du MPU.

Adresses du programme	Instructions en langage machine	Mnemoniques et données codées hexadecimal	Commentaire
0000	86 II	LDA A # $II	Charger l'ACCA par $II

0002	C6 FF	LDA B # $FF	Charger l'ACCB par $FF.
0004	CE IF IF	LDX # $IFIF	Charger le RX par $IFIF.
0007	3F	SWI	Interruption du programme.

- Faites entrer le programme dans les cases mémoires.
- Executez-le : "0000 " puis "GO".
- Le programme s'est arrêté en "0007 PC", on remarquera que le registre PC (Pointeur programme) est déjà visualisé.

Appuyer successivement sur la touche "GO" afin de visualiser les registres. D'abord le contenu de A(egale à II) est visualisé vient ensuite celui de B(egale à FF), celui de IX(egale à IFIF), celui de SP(egale à E4I8) et en dernier vient le contenu du registre CC avant de recycler sur PC.

Programme 2 : Changement du contenu des registres

Adresses du programme	Instructions en langage machine	Mnemoniques et données codées hexadécimal	Commentaire
0000	B7 00 I0	STA A ADR0	Stoker l'ACCA dans ADR0.
0003	F7 00 II	STA B ADRI	Stoker l'ACCB dans ADRI.
0006	FF 00 I2	STX ADR2	Stoker le RX dans ADR2.
0009	3F	SWI	Interruption du programme

Ce petit programme permettra à l'etudiant d'introduire des données dans les registres du MPU par l'intermediaire de la touche "RD".

- Faites entrer le programme en mémoire.
- Avant de l'executer faites entrer des données dans

les principaux registres A, B et X pour cela :

* Appuyez sur "RD", le contenu du PC est alors visualisé (sur les 4 premiers digits).
* Appuyez ensuite sur "GO", en premier le contenu de l'ACCA est visualisé, faites alors entrer par exemple la donnée "AA", le contenu précédent de l'ACCA est donc écrasé par "AA".
* De la même manière appuyez sur "GO", faites entrer dans ACCB la donnée BB, dans RX la donnée ABAB.
- Executez le programme.
- Verifiez le contenu des adresses 00I0, 00II, 00I2 et 00I3; vous deviez trouver respectivement :

00I0	AA
00II	BB
00I2	ABAB
00I3	AB

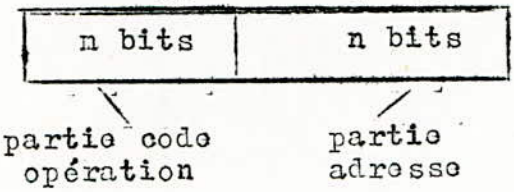
Exercice 4 : Utilisez les données de votre choix et procédez de la même manière.

TP 2 : UTILISATION DES DIFFERENTS TYPES D'ADRESSAGE

A - RAPPELS :

Un programme est une suite d'instructions. Une instruction se trouve rangée en mémoire sous la forme d'une ou plusieurs informations binaires.

En général une instruction comporte deux parties :



1. Code opération : Pour le 6802 la partie code opération est sur 8 bits (ou 1 octet) ; elle indique la nature de l'opération à effectuer.

2. Adresse : La partie adresse comporte au maximum deux octets, elle permet de :

- Localiser les nombres à traiter (les opérandes)
- Preciser les adresses de branchement
- Selectionner des périphériques ...

Pour le 6802 on distingue 3 grands types d'instructions :

1. Instructions de manipulation :

- Chargements des registres (LDAA, LDAB, LDX, LDS)
- Stockages en mémoires (STAA, STAB, STX, STS)

2. Instructions de traitement de l'information :

- Traitement arithmétique (ADDA, ADDB, SUBA, SUBB ...)
- Traitement logique ("ET", "OU", "OU exclusif" ...)

3. Instructions de saut et de branchement :

BRA, BCC, BCS, BNE, JMP, JSR, RTS,...

DIFFERENTS MODES D'ADRESSAGE :

Les principaux modes d'adressage du 6802 sont les suivants :

a) Adressage direct :

L'adresse (sur I octet) spécifiée dans la partie adresse designe la case où se trouve effectivement l'opérande.

b) Adressage étendu :

Identique à l'adressage direct sauf que l'adresse spécifiée dans la partie adresse est étendue à deux octets.

c) Adressage relatif :

Dans ce cas l'adresse de l'opérande est obtenue en tenant compte d'un déplacement (positif ou négatif) par rapport à l'adresse de l'instruction en cours d'execution.

La relation entre l'adresse relative et l'adresse absolue de la destination d'une instruction de branchement est :

$$D = (PC + 2) + R$$

où :

PC est l'adresse du prenier octet de l'instruction de branchement.

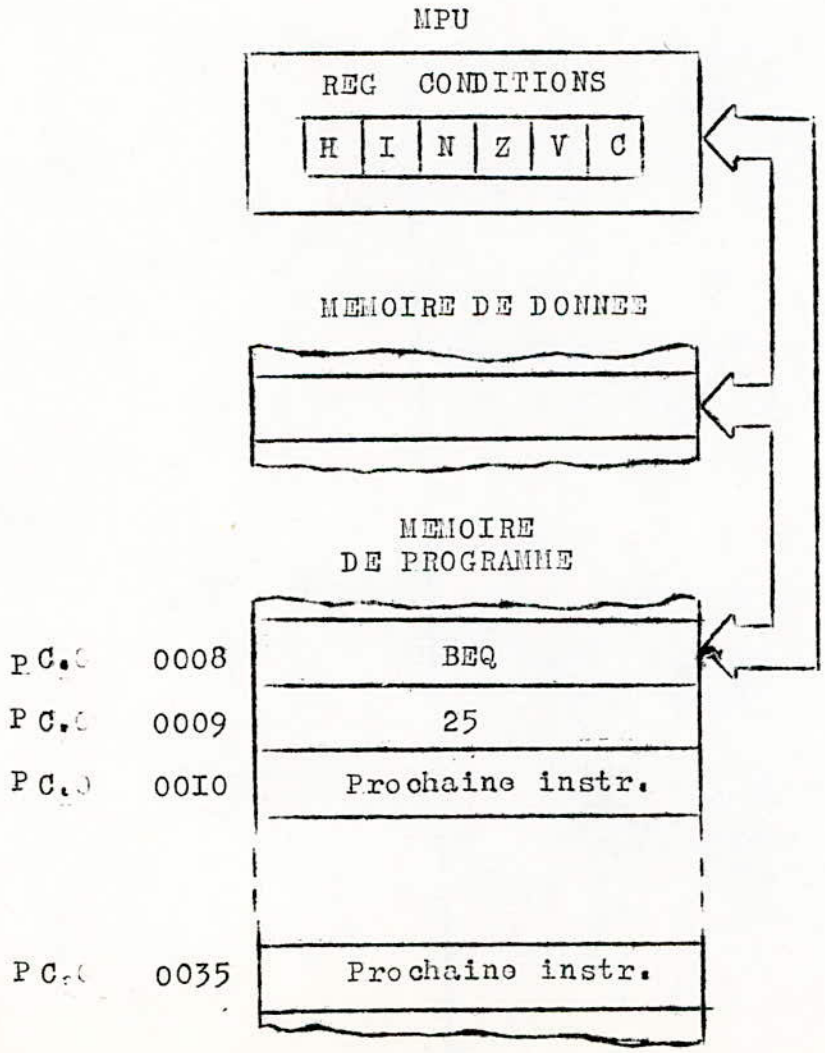
D est l'adresse de branchement.

R est un nonbre binaire en complément à 2 correspondant au deuxième octet de l'instruction de branch.

Pour que l'adressage relatif soit valide il faut que :

$$(PC + 2) - I28 \leqslant D \leqslant (PC + 2) + I27$$

Soit un exemple d'adressage relatif :



d) Adressage indexé :

Dans ce cas l'adresse de l'opérande auquel on veut accéder est calculée en ajoutant à l'adresse designée dans l'instruction le contenu d'un registre appelé "Index", ce registre peut être incrementé ou decrementé après chaque opération ce qui permettra donc à une même instruction de lire ou de charger une table complète de données rangées à des adresses successives. (voir exercice 3)

e) Adressage immédiat :

En faite l'instruction ne comprend pas une partie adresse, cette dernière est remplacée par l'opérande lui-même. Ce type d'adressage permet au programmeur d'introduire des constants dans son programme (Ex : LDAA # 34)

f) Adressage implicite :

Il n'ya pas de partie adresse, le code opération est, dans ce cas, une information suffisante pour faire executer l'instruction ; celle-ci n'occupe donc qu'un seul octet. (Ex : ASLA, ASLB, INX, DEX, SWI, RTS ...)

B - MANIPULATIONS :

OBJECTIF : Au cours de ce T.P nous allons voir à travers des programmes elementaires non seulement les différents modes d'adressage mais aussi les instructions essentielles du 6802 .

EXERCICE I : Addition de deux nombres en utilisant l'adressage immédiat.

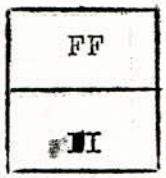
```
LDA A # $33        Charger l'ACCA par $33
LDA B # $44        Charger l'ACCB par $44
ABA                Additionner les Acc. A et B
SWI                Fin du programme.
```

TRAVAIL DEMANDE :

1. Traduisez ce programme en language machine.
2. Executez le programme sur le KIT D5.
3. Verifiez le resultat. Comment procedez-vous ?
4. Que contient le registre "CC" ? Explicitez :
   - Que signale l'indicateur carry "C"
   - Que signale l'indicateur overflow "V"
   - Que signale l'indicateur zéro "Z"
   - Que signale l'indicateur négatif "N"
   - Que signale l'indicateur demi-retenue "H"
5. Donner des exemples qui positionnent :
   - Le carry "C"
   - Le carry "C" et l'overflow "V"
   - Le carry "C" , le zéro "Z" et le demi-retenue "H"
6. Modifiez le programme (toujours la somme de deux nombres) en utilisant les instructions soit ADDA soit ADDB .

EXERCICE 2 : Permutation du contenu de deux positions nénoires.

On se propose de permuter le contenu de 2 positions nénoires (ADRI et ADR2) de telle sorte qu'après exécution du programme, le contenu de ADRI soit "II" et celui de ADR2 "FF".



LDA A	ADRI	Charger l'ACCA par le contenu de ADRI
LDA B	ADR2	Charger l'ACCB par le contenu de ADR2
STA A	ADR2	Stoker l'ACCA dans ADR2
STA B	ADRI	Stoker l'ACCB dans ADRI
SWI		Fin du programme.

A travers l'exercice 2 nous utilisons l'adressage étendu et les instructions de manipulation telle que LDA et STA .

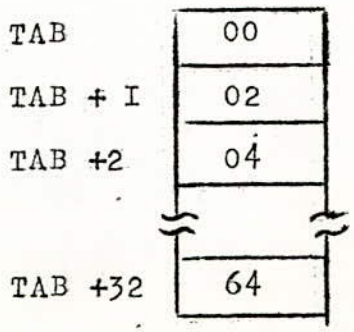
TRAVAIL DEMANDE :

I. Traduisez ce programme en language machine

2. Testez le programme sur le KIT D5.

EXERCICE 3 : Il s'agit d'un stockage , en nenoire centrale à des adresses successives, de tous les nonbres décinaux pairs de 0 à I00.(en hexadécinal il s'agit des nonbres pairs de $00 à $64).

Après l'execution du programme le contenu de la zone nénoire de stockage se presente conne ci-contre.



```
DEBUT   CLRA          Initialiser l'ACCA
        LDX # 0       Initialiser le RX
BOUCLE  STAA TAB,X    Stoker le contenu de A dans TAB
        INCA          Incrementer le contenu de A
        INCA          Incrementer le contenu de A
        INX           Incrementer le contenu de RX
        CPX TAB+33    Detecter la fin de la zone
        BNE BOUCLE    Sauter à BOUCLE si X ≠ TAB+33
FIN     SWI           Fin du programme.
```

Cet exercice illustre :

- 2 autres modes d'adressage : Indexé et Relatif
- Differentes types d'instructions (CPX, INCA ...)

TRAVAIL DEMANDE :

I. Reperez les 2 modes d'adressage utilisés dans ce programme, puis faites la traduction en language machine.

2. Comment procedez-vous pour calculer le déplacement "BOUCLE" à l'aide du KIT D5. Representez l'etat des afficheurs après chaque touche utilisée (se réferer au chapitre II paragraphe D.5)

EXERCICE 4 : Soit à realiser l'opération suivante :

$$2(ADRI - ADR2)$$

Nous savons que la multiplication par deux d'un nombre binaire, consiste à le décaler d'un cran vers la gauche.

Le programme qui realise cette opération est trés simple

ceci grâce à l'instruction de décalage arithmétique vers la gauche ASL .

LDA A	ADRI	Charger l'ACCA par le contenu de ADRI
SUBA	ADR2	Realiser l'opération (ADRI - ADR2).
ASL A		Décaler à gauche le contenu de A realisant ainsi l'opération2(ADRI - ADR2).
STA A	ADR3	Stoker le contenu de A dans ADR3
SWI		Fin du programme.

TRAVAIL DEMANDE :

1. Traduisez ce programme en language machine
2. Testez le programme sur le KIT D5 , en prenant par exemple :

ADRI	05
ADR2	02

3. On se propose de placer un point d'arrêt "Breakpoint" avant l'execution de l'instruction "ASLA".
   - Comment procedez-vous ?
   - Inspectez le contenu de la case mémoire ADR2 et de l'ACCA. Que concluez-vous ?

EXERCICE 5 : Instructions logiques : ANDA, ORAA, EORA

a) Annuler les 4 bits de droite d'un mot.

Le programme realisant la mise à zéro de quelque bits d'un mot est trés facile grâce à l'instruction"AND".

On suppose que le mot dont on se propose d'annuler les 4 bits de droite, se trouve dans une case mémoire ADR.

LDA A ADR	Charger A par le contenu de ADR
AND A # % IIII0000	Masquer A à l'aide de $F0
STA A ADR	Stoker A dans ADR
SWI	Fin du programme

TRAVAIL DEMANDE :

I. Traduisez ce programme en language machine

2. Testez le programme sur le KIT D5 en prenant pour exemple "MOT" = % IOIOIOIO = $ AA .

3. Quel sera le resultat obtenu en remplaçant ANDA par ORAA dans le programme ? conclure .

b) Complementer le bit "7" d'un mot.

Le programme qui realise cette opération est simple grâce à l'utilisation de l'instruction EORA ("OU" exclusif).

LDA A ADR	Charger A par le contenu de ADR
EOR A # % I0000000	Complementation du bit "7"
STA A ADR	Stoker A dans ADR
SWI	Fin du programme

TRAVAIL DEMANDE :

I. Traduisez ce programme en language machine

2. Testez le programme sur le KIT D5 en prenant "MOT" = % IOIOIOIO = $ AA .

3. Modifiez le programme dans le but de :

- Complementer le bit "4"
- Complementer le bit "5"
- Complementer les bits "4" et "5"

TP 3 : PRINCIPE DE PROGRAMMATION

Nous avons vu :

- Les touches essentielles et suffisantes pour introduire un programme dans le KIT D5.
- Les differents modes d'adressage et les instructions usuelles.

Nous allons voir :

A travers des exercices quels sont les principes à suivre pour realiser un programme.

DIFFERENTES PHASES DE MISE AU POINT D'UN PROGRAMME :

La redaction d'un programme et sa mise au point comporte différentes phases qui sont dans l'ordre chronologique :

1. L'analyse du problème à resoudre :
   - Description détaillée du problème
   - Fonctions à executer
   - Objectifs à atteindre ...
2. Etablissement d'un organigramme :
   On presente sous forme graphique toutes les séquences à executer pour la resolution du problème demandé.
   (voir annexe les différents symboles).
3. Programmation en langage mnémonique
4. Introduction du programme en héxadécimal
5. Execution du programme

MANIPULATION :

Realiser un programme n'est pas toujours si simple nous allons donc illustrer 2 exercices complets avant de vous demander de realiser votre propre organigramme (pour un problème donné).

EXERCICE I : Comparaison de 3 nombres

Soit 3 nombres non égaux entre eux, stockés aux adresses X, Y, et Z . Nous voulons les sortir en séquence ascendante et les stocker dans les cases mémoires : ADRI; ADR2, ADR3 .

MARCHE A SUIVRE :

I. On teste (X) - (Y), si (X) - (Y) > 0 on va à l'etape 2 , autrement on va à l'etape 3.

2. Mettre le contenu de la mémoire (Y) dans la case mémoire ADRI et celui de (X) dans ADR2.

3. Mettre le contenu de (X) dans ADRI et celui de (Y) dans ADR2.

4. On teste (Z) - (ADR2), si (Z) - (ADR2) > 0 on passe à l'etape 5 , sinon on passe à l'etape 6.

5. Mettre le contenu de Z dans ADR3 et faire sortir le resultat.

6. Mettre le contenu de la mémoire ADR2 dans ADR3 et le contenu de Z dans ADR2.

7. Tester (ADR2) - (ADRI), s'il est positif, on passe à l'etape 8 , autrement on passe à l'etape 9 .

8. Faire sortir le resultat.

9. Mettre le contenu de ADRI dans ADR4, (ADR2) dans ADRI, et (ADR4) dans ADR2.

IO. Faire sortir le resultat.

ORGANIGRAMME :

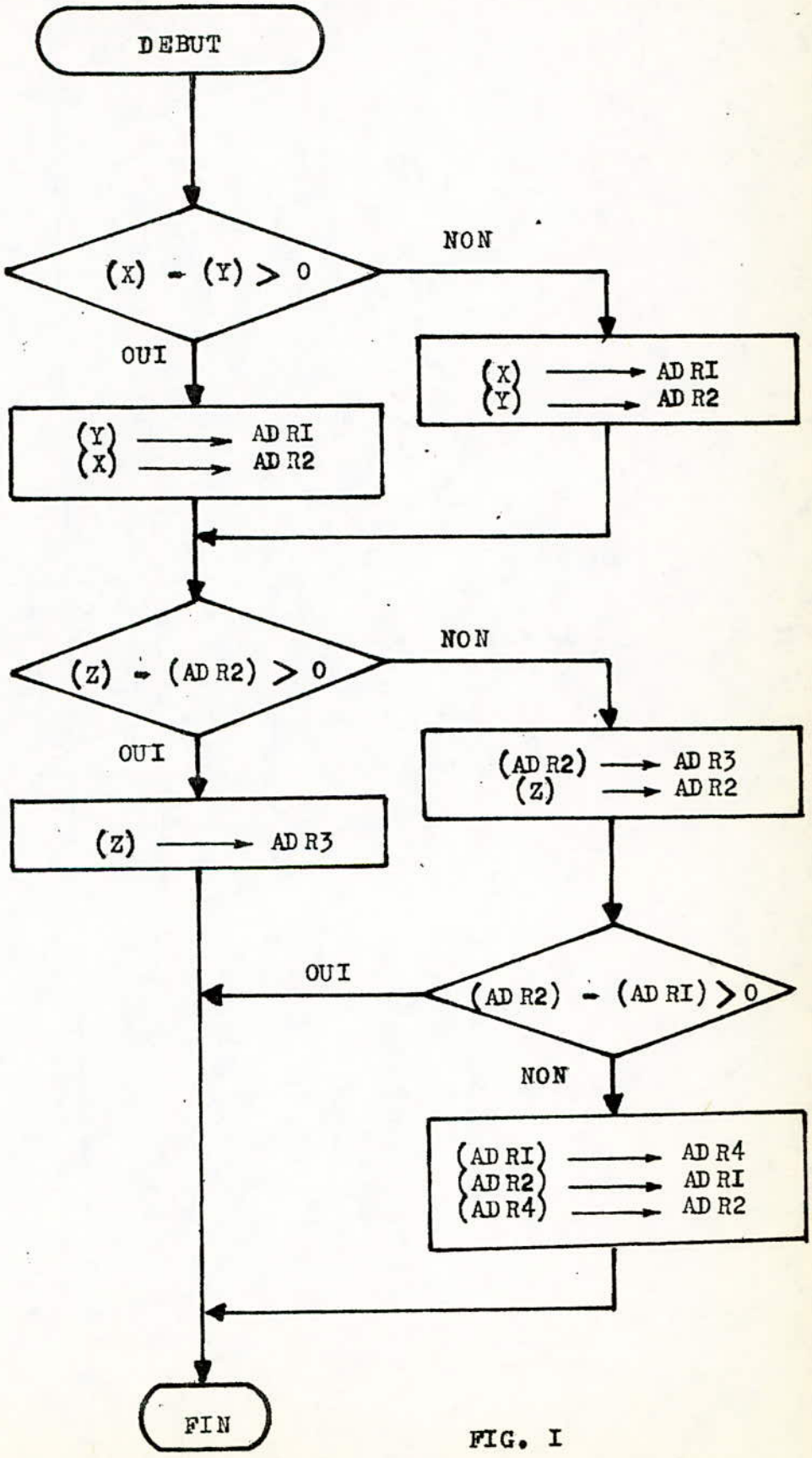


FIG. I

PROGRAMME :

```
DEBUT     LDA A   X          Charger A par le contenu de X
          CMP A   Y          Comparer A avec le contenu de Y
          BLT   SUITE I      Saut à SUITE I si (A) < (Y)
          LDA B   Y          Charger B par le contenu de Y
          STA B   ADRI       Stocker B dans la case ADRI
          STA A   ADR2       Stocker A dans la case ADR2
SUITE I   STA A   ADRI       Stocker A dans la case ADRI
          LDA B   Y          Charger B par le contenu de Y
          STA B   ADR2       Stocker B dans la case ADR2
          LDA A   Z          Charger A par le contenu de Z
          CMP A   ADR2       Comparer A avec le contenu de ADR2
          BLT   SUITE 2      Saut à SUITE 2 si (A) < (ADR2)
          STA A   ADR3       Stocker A dans la case ADR3
          BRA   FIN          Saut à l'adresse FIN
SUITE 2   LDA B   ADR2       Charger B par le contenu de ADR2
          STA B   ADR3       Stocker B dans la case ADR3
          STA A   ADR2       Stocker A dans la case ADR2
          LDA A   ADR2       Charger A par le contenu de ADR2
          CMP A   ADRI       Comparer A avec le contenu de ADRI
          BLT   SUITE 3      Saut à SUITE 3 si (A) < (ADRI)
          BRA   FIN          Saut à l'adresse FIN
SUITE 3   LDA B   ADRI       Charger B par le contenu de ADRI
          STA B   ADR4       Stocker B dans la case ADR4
          STA A   ADRI       Stocker A dans la case ADRI
          LDA A   ADR4       Charger A par le contenu de ADR4
          STA A   ADR2       Stocker A dans la case ADR2
FIN       SWI                Fin du programme.
```

TRAVAIL DEMANDE :

I. Traduisez ce programme en langage machine

2. Testez le programme sur le KIT D5 en prenant :

X = 0000 avec (X) = 02
Y = 000I avec (Y) = 03
Z = 0002 avec (Z) = 0I

ADRI = 0003
ADR2 = 0004
ADR3 = 0005
ADR4 = 0006

3. Visualisez le contenu des cases 0003, 0004 et 0005, que constatez-vous ?

EXERCICE 2 : Recopie d'une zone de mémoire dans une autre zone.

Soit à realiser le programme qui consiste à transferer le contenu d'une zone mémoire dans une autre zone mémoire :

ORGANIGRAMME : L'organigramme qui realise cette opération est donné par la figure 2 .

Deux cas peuvent se présenter :

a) Cas ou la longueur de la zone à recopier est supérieur à 256 octets :

Principe de la recopie :

- Transfert d'un octet par l'intermediaire de l'ACCA;
- Décrémentation du pointeur X (INDEX). La décrémentation du contenu du pointeur de pile SP se fait automatiquement à chaque transfert;
- Test de X.

La zone de départ et la zone d'arrivée sont à des emplacement quelquonques .

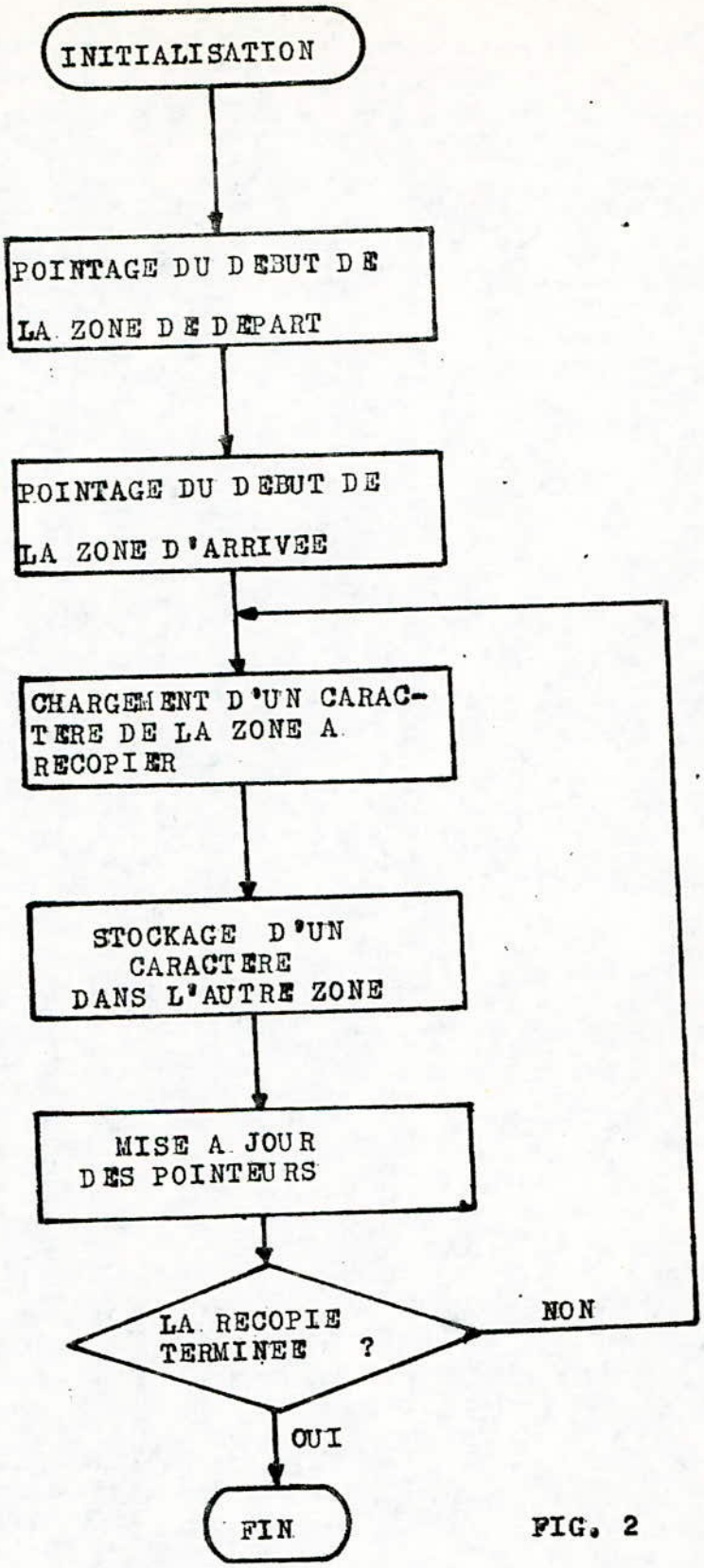
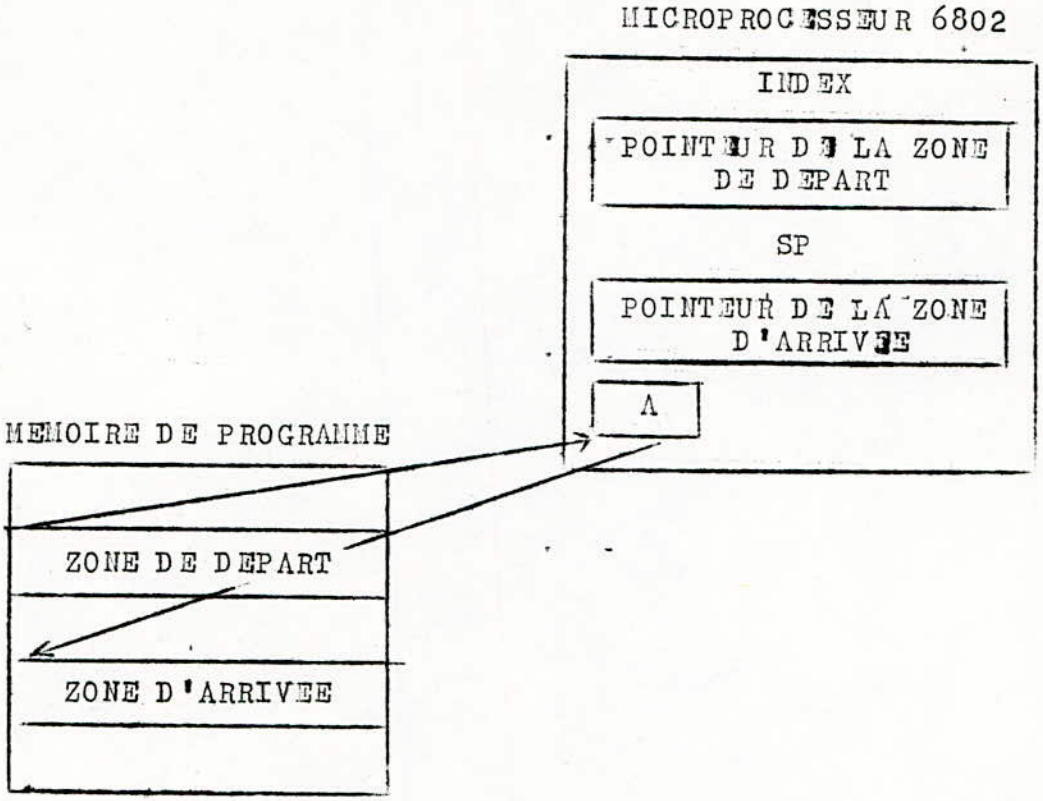

INITIALISATION
POINTAGE DU DEBUT DE LA ZONE DE DEPART
POINTAGE DU DEBUT DE LA ZONE D'ARRIVEE
CHARGEMENT D'UN CARACTERE DE LA ZONE A RECOPIER
STOCKAGE D'UN CARACTERE DANS L'AUTRE ZONE
MISE A JOUR DES POINTEURS
LA RECOPIE TERMINEE ?
NON
OUI
FIN

FIG. 2



PROGRAMME :

DEBUT	LDX # FIN I	Chargement en immédiat de l'adresse de la fin de zone de départ.
	STS M	Sauvegarde du contenu du pointeur de pile.
	LDS # FIN 2	Chargement en immédiat de l'adresse de la fin de zone d'arrivée.
BOUCLE	LDA A 0,X	Chargement de l'ACCA avec la position mémoire définie par X.
	PSH A	Transfert du contenu de l'ACCA vers la position mémoire définie par SP.
	DEX	Décrémentation du contenu de X.
	CPX # ZFIN	Détection de fin de zone.
	BNE BOUCLE	Saut à l'adresse "BOUCLE" si la recopie n'est pas finie.
	LDS M	Rechargement du pointeur de pile avec le contenu avant transfert.
FIN	SWI	Fin du programme.

Vu la repartition ménoire du KIT D5 (voir CHAPITRE 2) nous transferons le contenu des cases ménoires de E000 à EIAB dans la zone ménoire de E200 à E3AB.

Soit : FIN I = EIAB

FIN 2 = E3AB

Z FIN = DFFF

M = 0000

LISTING DU PROGRAMME :

DEBUT	00I0	CE EI AB	LDX # FINI
	00I3	9F 00	STS M
	00I5	8E E3 AB	LDS # FIN 2
BOUCLE	00I8	A6 00	LDA A 0,X
	00IA	36	PSH A
	00IB	09	DEX
	00IC	8C DF FF	CPX # ZFIN
	00IF	26 F7	BNE BOUCLE
	002I	9E 00	LDS M
	0023	3F	SWI

b) Cas ou la longueur de la zone ménoire est inférieur ou égale à 256 octets :

Soit à realiser l'organigramme et le programme d'une recopie d'une zone ménoire; dans le cas où la longueur est inférieur ou égale à 256 octets.

TRAVAIL DEMANDE :

I. Realisez l'organigramme, et ecrivez le programme correspondant.
2. Testez le programme sur le KIT D5 pour une zone ménoire de votre choix.

EXERCICE 3 : Soit 5 nombres se trouvant dans des cases mémoires d'adresses consécutives , constituant ainsi une table :

0000	09
0001	21
0002	0F
0003	02
0004	1A

TRAVAIL DEMANDE :

1. Cherchez le plus grand élément de cette table, et stockez-le dans une case mémoire appelée "MAX" :

   a) Etablir l'organigramme qui realise cette opération

   b) En déduire le programme

   c) Tester ce programme sur le KIT D5.

2. Faites la somme des 5 éléments de cette table :

   a) Etablir l'organigramme qui realise cette opération

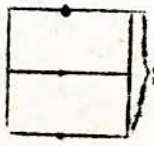
   b) En déduire le programme

   c) Tester ce programme sur le KIT D5.

REMARQUE :

Après l'éxecution du programme, la somme des éléments de la table ci-dessus sera stockée aux adresses 0005 et 0006 (c'est à dire la somme occupe 16 bits) :

SOMBAS = 0005
SOMHTE = 0006



(SOMBAS) : Partie basse de la somme
(SOMHTE) : Partie haute de la somme

EXERCICE 4 : Soit le nonbre N codé héxadécinal on donande de le convertir en BCD.

TRAVAIL DEMANDE :

1. Etablir l'organigranme qui realise cette opération
2. En déduire le programme
3. Tester ce programme sur le KIT D5 pour :
   - N = $A8
   - N = $64
   - N = $2D

TP 4 : MULTIPLICATION ET DIVISION

A - MULTIPLICATION DE 2 OCTETS SANS SIGNE :

Examinons l'exemple d'une multiplication binaire .Elle est effectuée exactement de la même manière que la multiplication décimale :

(7)				I	I	I		Multiplicande
(5)				I	0	I		Multiplicateur
				I	I	I		
			0	0	0			
		I	I	I				Produits Partiels
(35)	I	0	0	0	I	I		Résultat

On multiplie chaque bit du multiplicateur par le multiplicande, chaque produit partiel ainsi obtenu sera zéro ou le multiplicande lui-même . Si le multiplicande a n bits et le multiplicateur n bits , le resultat comprendra (n + n)bits. Il ya n produits partiels qu'on ajoutera 2 à 2 après avoir effectuer les décalages correspondants , la somme ainsi obtenue represente la succession des sommes partielles.

ORGANIGRAMME : L'organigramme qui correspond à la multiplication effectuée plus haut est donné par la figure I.

TRAVAIL DEMANDE :

EXERCICE I :

a) A partir de l'organigramme (fig.I) , déduire le programme.

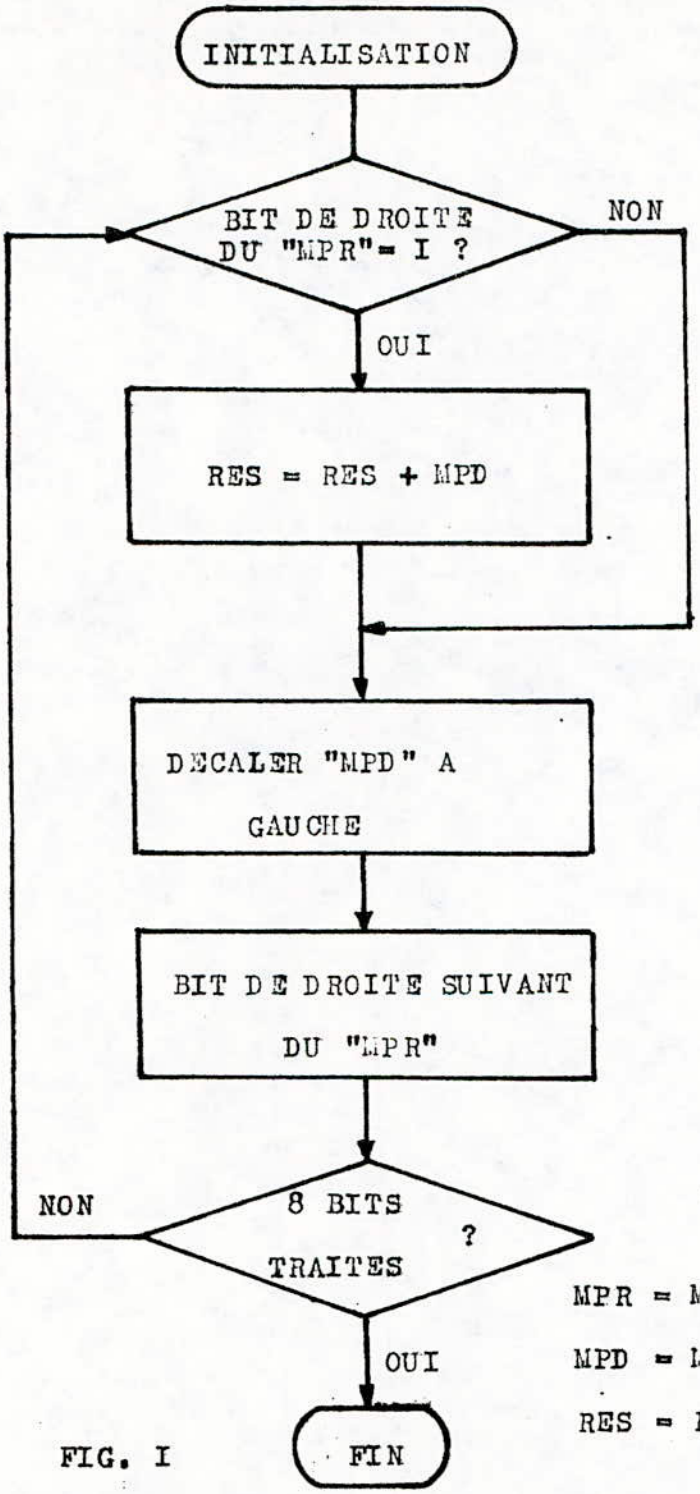


FIG. I

MPR = MULTIPLICATEUR

MPD = MULTIPLICANDE

RES = RESULTAT

b) Tester le programme sur le KIT D5 en prenant pour exemple :

A A	Multiplicande
B B	Multiplicateur
2 E 7 C	Resultat

Le programme que vous avez developper dans l'exercice I n'est qu'un des nombreuses formes sous lequelles il pourrait avoir été écrit. Tout programmeur peut trouver des façons de modifier, et parfois d'améliorerun pfogramme.

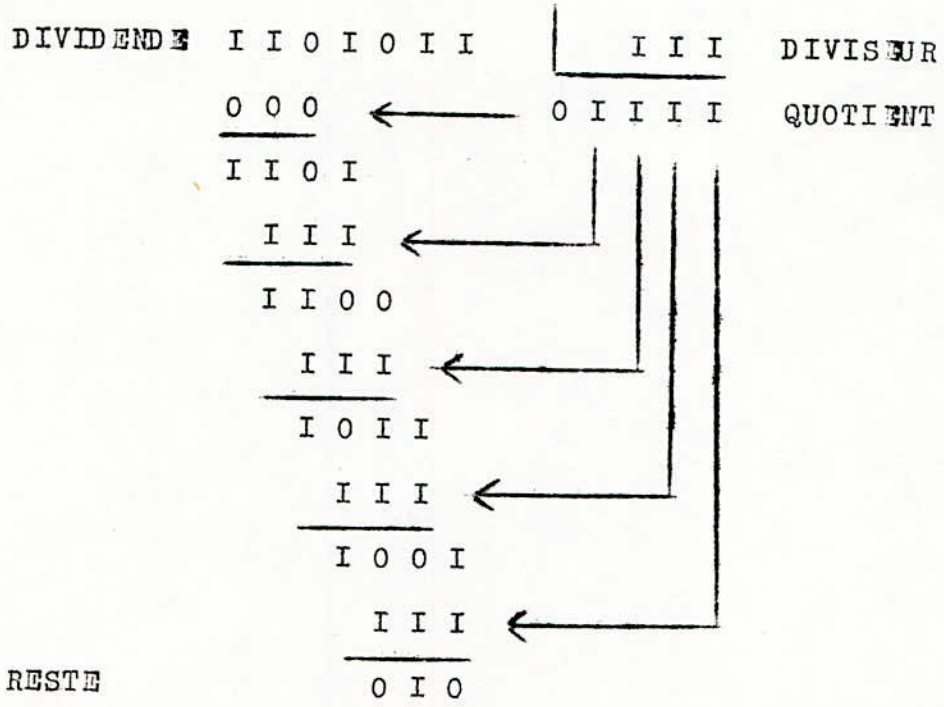
Par exemple , vous avez decalé à gauche le multiplicande avant l'addition. Il aurait été mathématiquement équivalent de decaler le resultat à droite avant de l'ajouter au multiplicande.

EXERCICE 2 : Soit à effectuer maintonant une multiplication 8X8 à l'aide du même algorithme mais en décalant le resultat d'un cran à droite au lieu de décaler le multiplicande d'un cran à gauche.

a) Ecrire L'organigramme qui correspond à cette méthode
b) En déduire le programme
c) Tester le programme sur le KIT D5 en prenant AA comme multiplicande et BB comme multiplicateur
d) Comparer ce programme à celui de l'exercice I et determiner si cette nouvelle approche est plus rapide ou plus lente que la précédente.

B - DIVISION :

Examinons l'exemple numérique suivant :



Nous remarquons que dans la division , on retranche au dividende soit le diviseur, soit zéro suivant que le bit correspondant du quotient est à "I" ou "0".On recommence au pas suivant après avoir fait subir un décalage à droite du diviseur par rapport au dividende.

Ainsi, multiplication ou division sont des opérations très comparables. Il suffit de changer les additions en soustractions et d'inverser le sens des décalages.

Pourtant une difficultée complémentaire apparait dans le cas de la division. Il faut ajouter à chaque étape une opération de comparaison entre les bits de fort poids du dividende et le diviseur afin de determiner le bit correspondant du quotient.

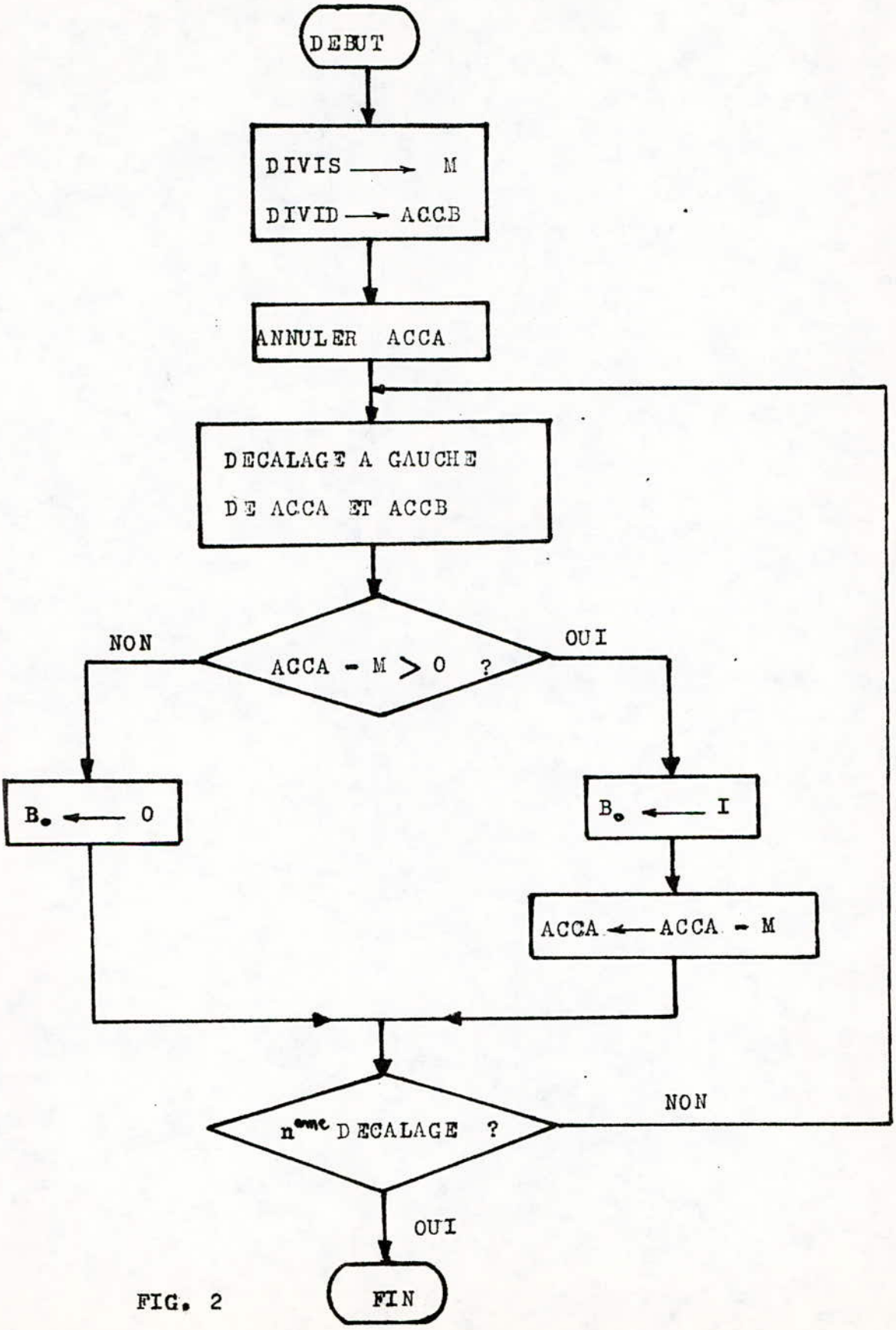

DEBUT
DIVIS ⟶ M
DIVID ⟶ ACCB
ANNULER ACCA
DECALAGE A GAUCHE
DE ACCA ET ACCB
ACCA - M > 0 ?
NON
OUI
B. ⟵ 0
B. ⟵ I
ACCA ⟵ ACCA - M
n^eme DECALAGE ?
NON
OUI
FIN

FIG. 2

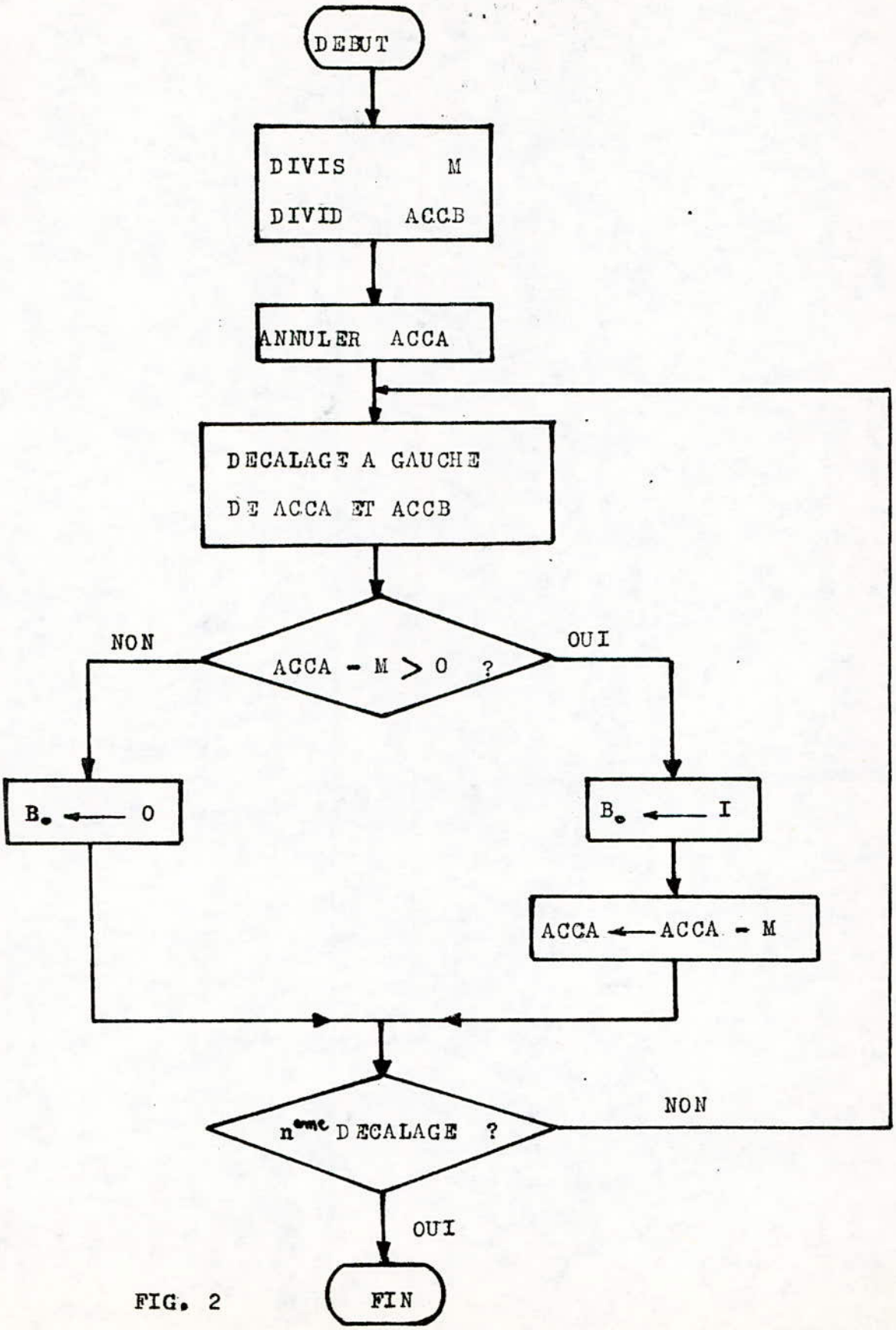

DEBUT
DIVIS M
DIVID ACCB
ANNULER ACCA
DECALAGE A GAUCHE
DE ACCA ET ACCB
NON
ACCA - M > 0 ?
OUI
B. ← 0
B. ← I
ACCA ← ACCA - M
n^eme DECALAGE ?
NON
OUI
FIN

FIG. 2

Nous allons etudier le cas de la division d'un nombre de 8 bits par un nombre de 8 bits sans tenir compte du signe. Pour ce faire nous proposons la methode suivante illustrée par l'organigramme (fig. 2).

TRAVAIL DEMANDE :

EXERCICE 3 :

a) Faites le programme correspondant à l'organigramme de la figure 2.

b) Testez votre programme sur le KIT D5 en prenant pour exemple :

DIVIDENDE	6B	07	DIVISEUR
		0F	QUOTIENT
RESTE	02		

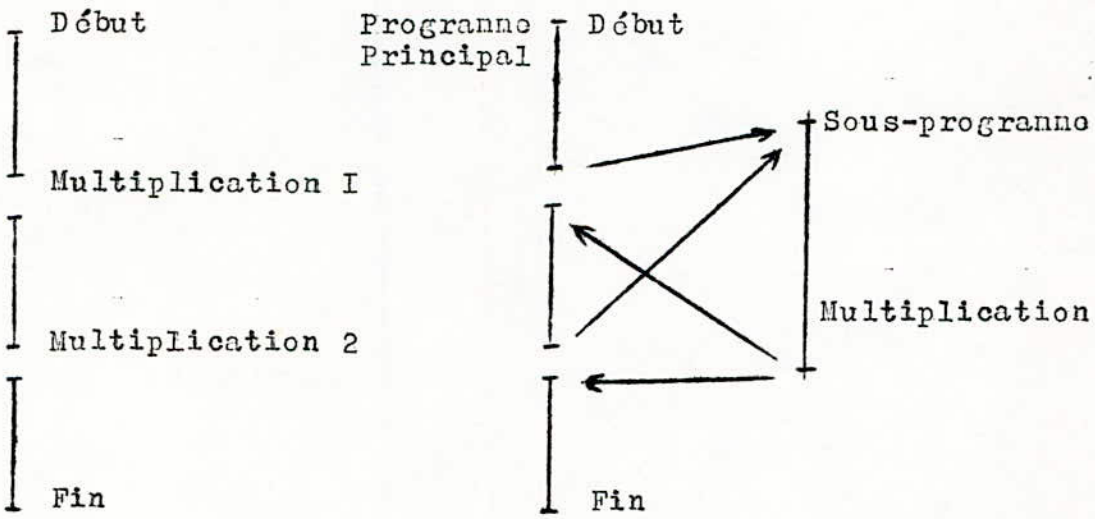
En fin de l'opération le quotient apparait dans l'ACCB et le reste dans l'ACCA .

TP 5 : UTILISATION DES SOUS-PROGRAMME DU MONITEUR D5-BUG

GENERALITES :

I - NOTION DE SOUS-PROGRAMME :

Au cours d'un programme, on peut retrouver à plusieurs reprises une même séquence d'instructions réalisant une opération donnée(multiplication). Plutôt que de répéter chaque fois cette séquence, ce qui encombrerait inutilement la mémoire, il est intéressant de l'implanter une seule fois en mémoire sous forme de sous-programme.

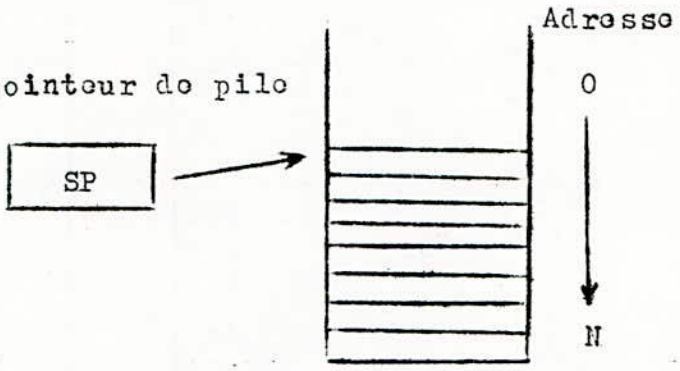


Quand ce sous-programme doit être exécuté, on fait appel de sous-programme (BSR, JSR) et quant on a terminé son exécution, on revient dans le programme principal par un retour de sous-programme (RTS). Les instructions d'appel de sous-programme ressemblent à celles de saut, mais nécessitent

néanmoins la sauvegarde dans la "pile" de l'adresse de retour pour pouvoir revenir par la suite au programme principal à l'endroit où il a été abandonné.

2 - NOTION DE PILE :

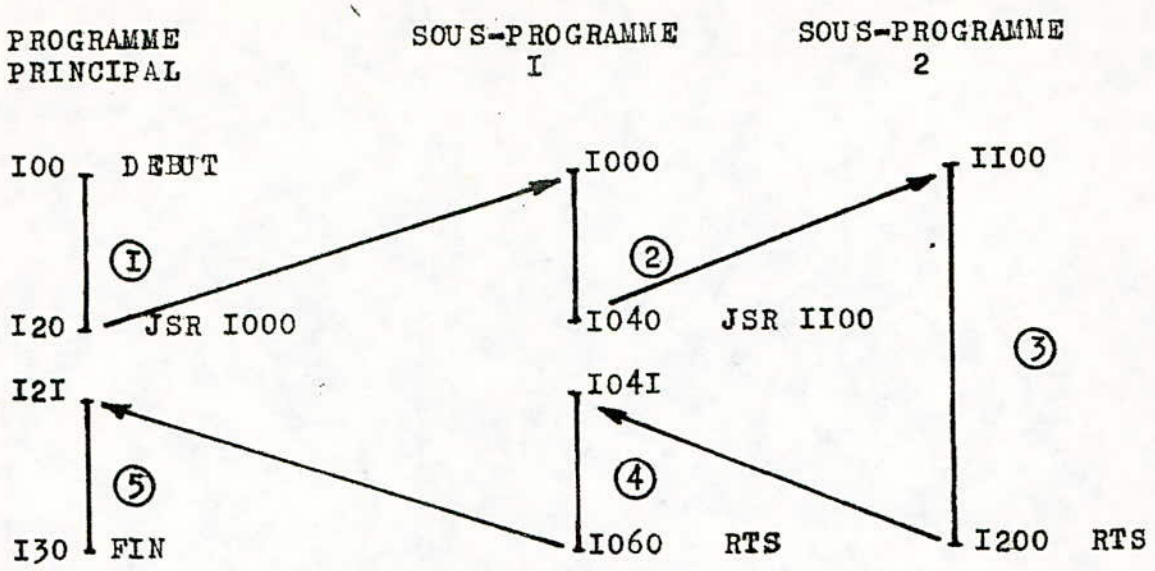
Une pile est un ensemble de cases mémoires consécutives, accessible grâce à un pointeur de pile (SP).



Nous avons vu plus haut que la pile peut être utilisée implicitement lors de l'appel de sous-programmes. Examinons l'exemple de la figure.I, les états successifs de la pile lors de l'exécution de deux sous-programmes imbriqués.

Dans l'exemple, les adresses sont écrites en décimal et l'on suppose que toutes les adresses peuvent être stockées dans une seule case mémoire.

En plus de cette utilisation automatique de la pile, il est très commode de s'en servir pour la sauvegarde temporaire du contenu de certains registres internes du microprocesseur, ce qui évite, dans beaucoup de cas simples, l'utilisation dans le système d'une autre mémoire vive.


PROGRAMME PRINCIPAL
SOUS-PROGRAMME 1
SOUS-PROGRAMME 2
I00 DEBUT
I20 JSR I000
I21
I30 FIN
I000
I040 JSR II00
I041
I060 RTS
II00
I200 RTS
①
②
③
④
⑤

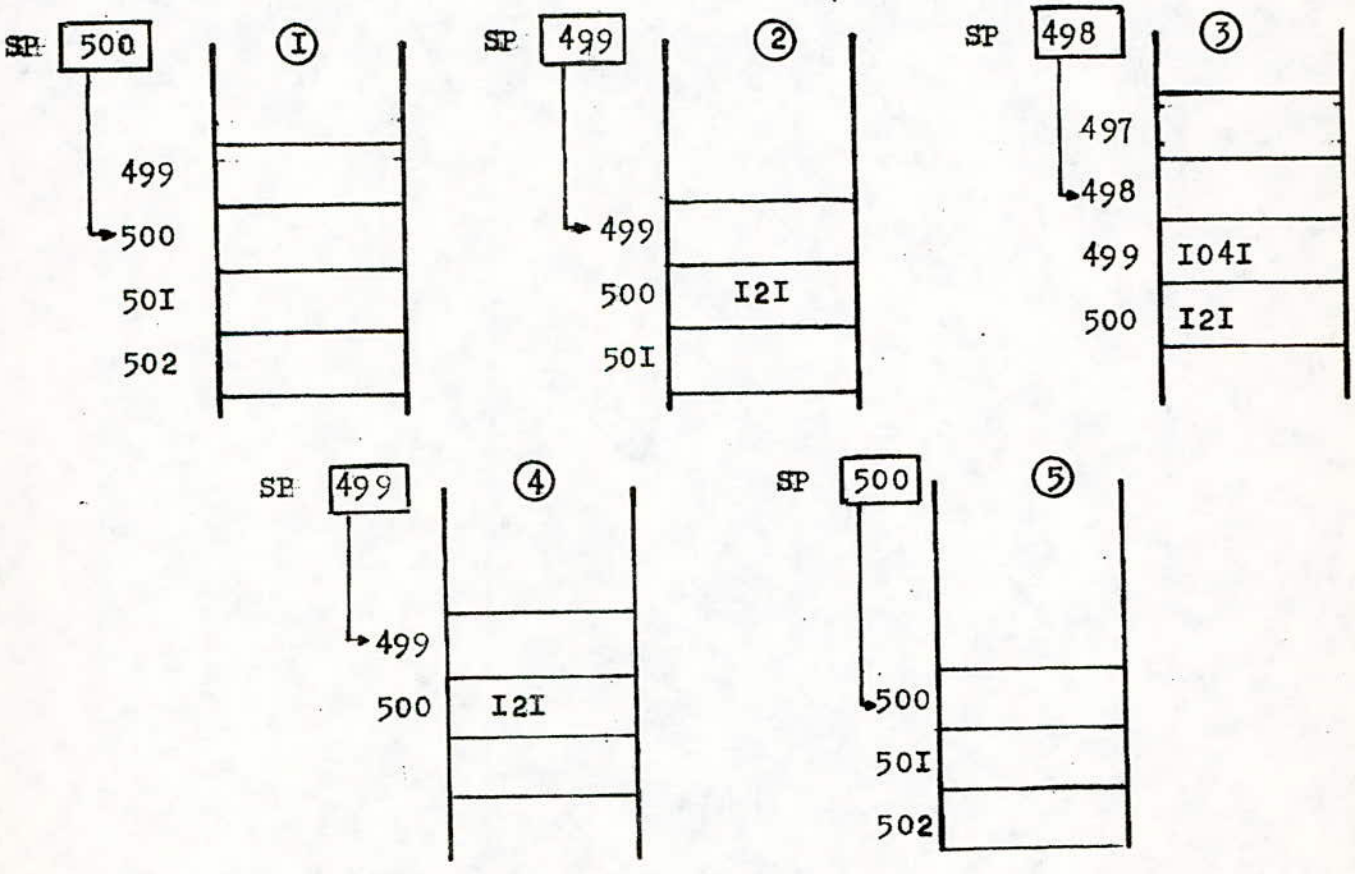

SP 500
499
500
501
502
SP 499
499
500 I21
501
SP 498
497
498
499 I041
500 I21
SP 499
499
500 I21
SP 500
500
501
502

FIG.I

SOUS-PROGRAMME DELAI :

Le moniteur D5 BUG dispose de 3 types de délais (sub-routines) utilisable par le programmeur, il s'agit de ;

DLY 25 ($FI69)

DLY I ($FI7I)

DLY X ($FI79)

Ces routines sont des délais de 25 ns et I ns en ce qui concerne les 2 premières. DLY X est un délai variable grâce au contenu du registre d'index IX.

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

(X - I) (8 cyc) (I,II763 µs/cyc) + (27,94 µs)

Délai minimum = 27,94 µs pour X = $000I

Délai maximum = 0,5842 sec pour X = $FFFF

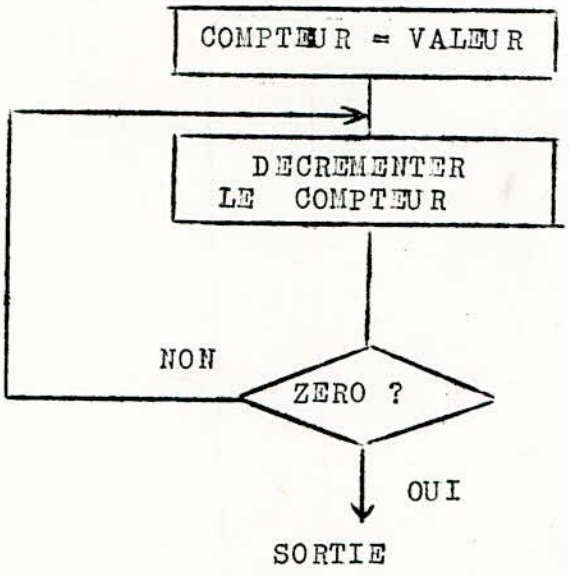
Puisque le délai maximum prévu sur le KIT est de 0,5842 seconde, il est nécéssaire au programmeur de concevoir son propre programme délai pour obtenir un délai plus long.

Les délais programmés sont obtenus par comptage. On charge un registre avec une valeur, puis on le décrémente, le programme boucle sur lui-même et continu à décrémenter le registre jusqu'à ce que celui ci atteigne la valeur "0".

Le temps total requis par ce processus va realiser le délai souhaité. Par exemple, realisons un délai de 50 µs :

```
DELAI     LDA B # 8        L'ACCB sert de compteur
ENCORE    DEC B            Décrémenter l'ACCB
          BNE ENCORE       Test
```

La logique de ce programme est très simple et elle est conforme à l'organigramme de la figure qui suit :



Calculons maintenant le délai effectif qui sera réalisé par ce programme. Se référer à l'annexe pour le nombre de cycles correspondant à chacune des instructions .

LDA , en mode immédiat demande : 2 cycles

DEC prend : 2 cycles

BNE prend : 4 cycles

Délai = 2 + 6X8 = 50 cycles

Si l'on suppose un temps de cycle de I μs, ce délai sera donc de 50 μs

TRAVAIL DEMANDE :

EXERCICE I ;

a) Quel est le délai minimum et maximum qu'on peut réaliser avec le programme précédent.

b) Modifier le programme pour obtenir un délai de 300 µs

c) Quel est le délai minimum et maximum qu'on peut obtenir en utilisant cette fois-ci le registre d'index.

EXERCICE 2 : On veut générer des délais encore plus longs :

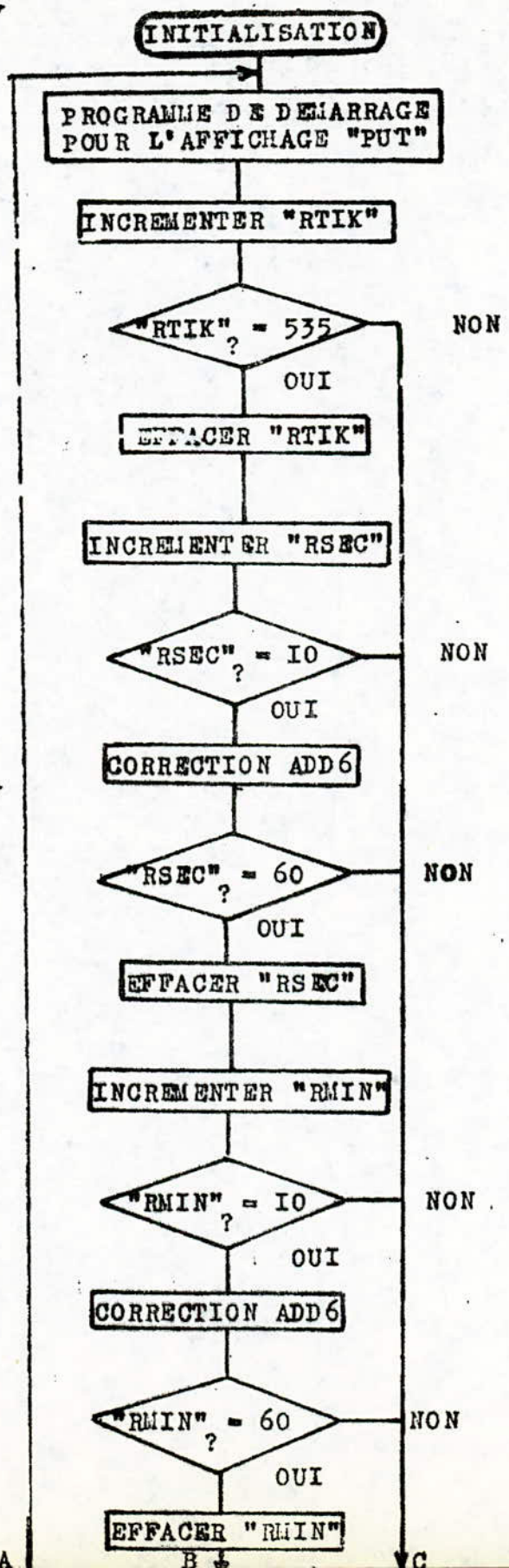
- Comment procédez-vous ?
- Faites l'organigramme et le programme correspondant pour générer un délai de 30 secondes
- Testez le programme sur le KIT D5, à l'aide d'une montre chrono.

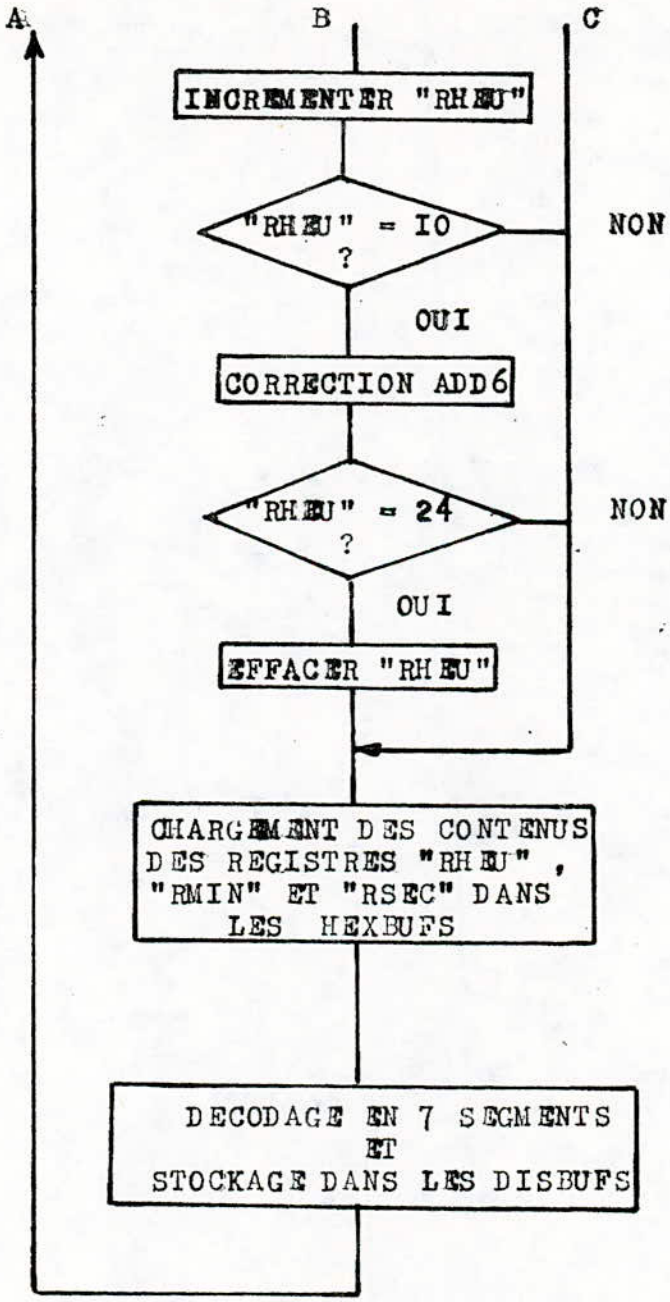
SOUS-PROGRAMME HORLOGE :

On transforme 5 cases mémoires en horloge (heures , minutes , secondes), le système d'affichage du KIT D5 se chargera de visualiser ces cases mémoires ("RHEU" , "RMIN" , "RSEC" , "RTIK H" , "RTIK L") dans les 6 digits.

Le principe de cette horloge est d'incrémenter le registre TICK "RTIK" (constitué de deux cases mémoires "RTIK H" et "RTIK L") après chaque rafraichissement des digits (qui dure environ 1,87 ms). Après avoir généré 1 seconde (535 itérations) on incrémente le registre seconde "RSEC" et on initialise "RTIK" pour un nouveau comptage. Arrivé à 10 (0A) on doit incrémenter les dizaines de seconde après avoir mis à zéro "RSEC". On teste les dizaines à 06 et si c'est égale on met à zéro et on incrémente "RMIN" , et ainsi de suite , l'affichage se fait en DCB .

ORGANIGRAMME :




A
B
C
INCREMENTER "RHEU"
"RHEU" = IO ?
NON
OUI
CORRECTION ADD6
"RHEU" = 24 ?
NON
OUI
EFFACER "RHEU"
CHARGEMENT DES CONTENUS DES REGISTRES "RHEU", "RMIN" ET "RSEC" DANS LES HEXBUFS
DECODAGE EN 7 SEGMENTS ET STOCKAGE DANS LES DISBUFS

PROGRAMME :

```
INITIA      CLR A
            STA A  RHEU
            STA A  RMIN
            STA A  RSEC          Initialisation
            STA A  RTIKH
            STA A  RTIKL
START       LDX #  TIME          Charger X avec l'adresse TIME
            STX    MNPTR         Preparation d'un saut à PUT
            JMP    PUT           Saut à PUT
TIME        LDX    RTIK          Enregistrer le TICK de l'horloge
            INX                  Incrémenter  RTIK
            STX    RTIK          Nouvelle valeur de RTIK
            CPX #  535($0217)    Ya t-il I seconde ?
            BNE    VISU
            LDX #  0000
            STX    RTIK          Effacer RTIK
            INC    RSEC          Enregistrer I seconde
            LDA A  RSEC
            ORA A  #$F5          Ya t-il IO secondes?
            COM A
            BNE    VISU
            LDA A  RSEC
            ADD A  # 06          Correction ADD6
            STA A  RSEC
            CMP A  # 60          Ya t-il 60 secondes ?
```

```
        BNE   VISU
        CLR   RSEC          Effacer RSEC
        INC   RMIN          Enregistrer I minute
        LDA A RMIN
        ORA A # $F5         Ya t-il I0 minutes ?
        COM A
        BNE   VISU
        LDA A RMIN
        ADD A # 06          Correction ADD6
        STA A RMIN
        CMPA # 60           Y-a-t-il 60 minutes ?
        BNE   VISU
        CLR   RMIN          Effacer RMIN
        INC   RHEU          Enregistrer I heure
        LDA A RHEU
        ORA A # $F5         Y-a-t-il I0 heures ?
        COM A
        BNE   CHECK
        LDA A RHEU
        ADD A # 06          Correction ADD6
        STA A RHEU
CHECK   LDA A RHEU
        CMP A # 24          Y-a-t-il 24 heures ?
        BNE   VISU
        CLR   RHEU          Effacer RHEU
VISU    LDX   RHEU
```

```
STX  HEXBUF        Chargement des HEXBUFS
LDA  A RSEC        par les contenus de RSEC,
STA  A HEXBUF+2    RMIN et RHEU.
JMP  DYSCOD        Décodage 7 segments
RTS                Retour à l'affichage.
```

TRAVAIL DEMANDE :

EXERCICE I :

a) Traduisez le sous-programme horloge en language machine

b) Testez le sous-programme sur le KIT D5

c) Vous remarquez sans doute que ce programme fonctionne en chronomètre, aussi faites entrer l'heure actuelle pour obtenir une horloge en temps réel.

EXERCICE 2 : En modifiant quelque peu le programme précédent on vous demande de visualiser l'heure toute les minutes.

TP 6 INITIALISATION DU PIA

Le but de ce TP est de vous apprendre comment initialiser le PIA (Périphéral Interface Adapter) pour toute échange entre le MPU et un périphérique.

GENERALITES SUR LE PIA 6821 :

Le PIA MC 6821 est un adapteur d'interface périphérique programmable, organisé de deux moitiés synétriques A et B (voir fig.I.).

Chaque partie comporte 3 registres :

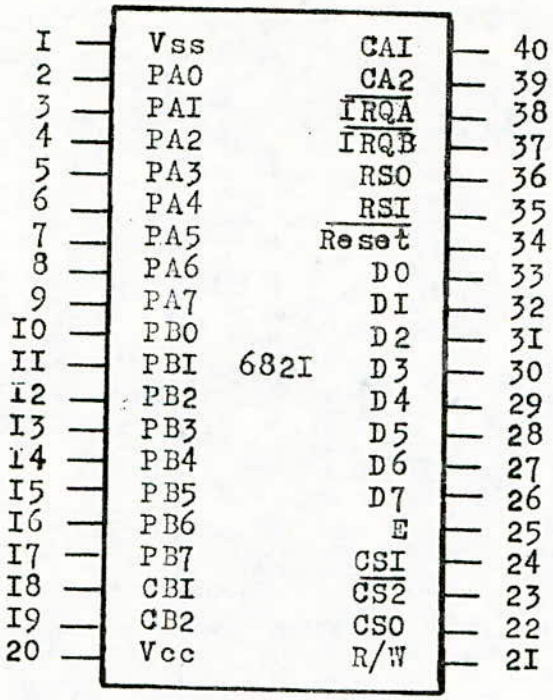
- Registre de données ORA (B)
- Registre de direction de données DDRA (B)
- Registre de contrôle CRA (B)

ORA (B) et DDRA (B) ont une même adresse, la selection se faisant par un bit du registre de contrôle.

Ces 6 registres sont donc considérés par l'extérieur comme 4 adresses différentes.

. Le registre de direction de données détermine l'utilisation de chaque bit du registre de données en entrées ou en sorties. Un zéro dans le bit de rang n du registre de direction de données entraine la programmation du bit de rang n du registre de données en entrée et inversement.

. Le registre de données est relié aubus externe PA (ou PB) qui est donc bidirectionnel.



BROCHAGE DU 682I

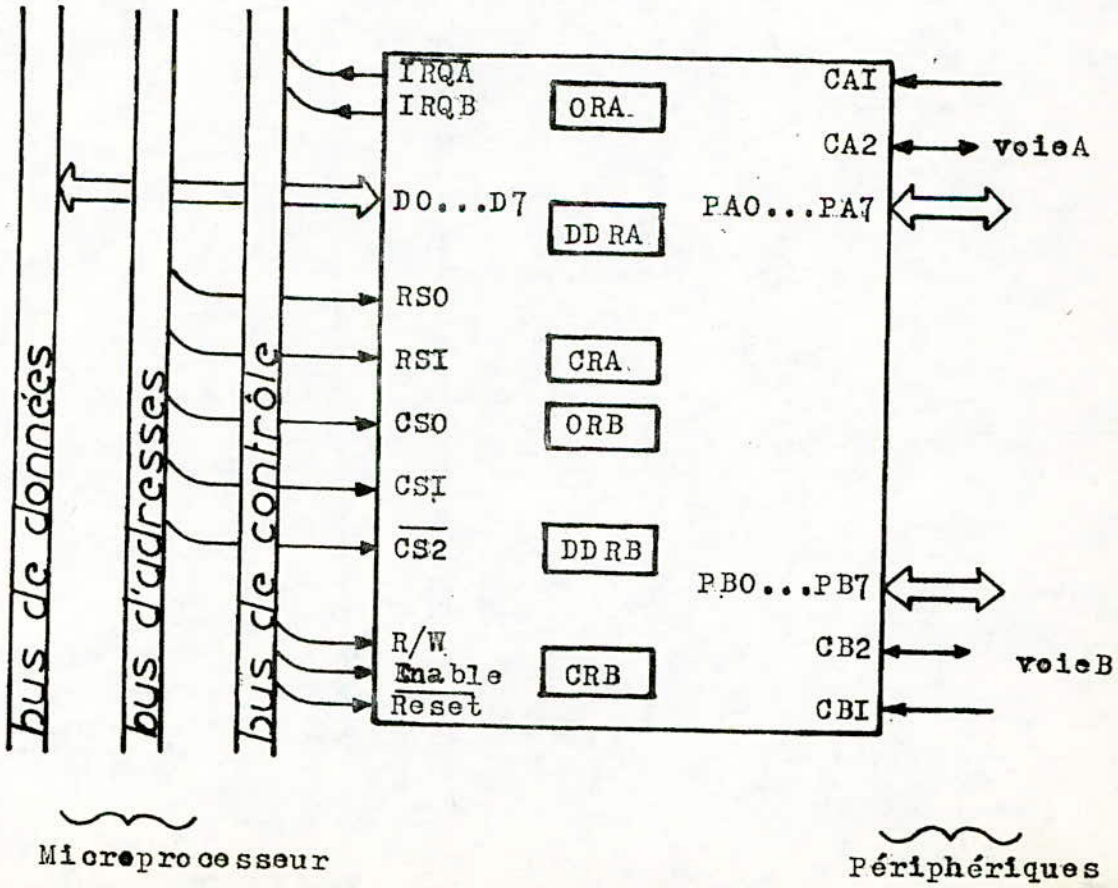


FIG.I. ORGANISATION INTERNE ET EXTERNE DU PIA 682I.

. Le registre de contrôle permet de programmer les modes de fonctionnement du I/2 PIA, chaque bit possède une fonction précise, comme par exemple, la sélection de DDR ou OR, le mode de fonctionnement des lignes de contrôle du périphérique (CA ou CB) et des sorties d'interruption.

La figure 2 résume le rôle de chacun des bits du registre de contrôle.

Les lignes CS0, CSI et $\overline{CS2}$ sont des entrées qui permettent de sélectionner le PIA.

Les lignes RS0 et RSI permettent en combinaison avec le bit 2 du registre de contrôle, de sélectionner les registres internes du PIA , ceci se résume dans le tableau suivant:

RS0	RSI	CRA2	CRB2	Registre selectionné
0	0	I	X	Registre de sortie (ORA)
0	0	0	X	Registre de direction (DDRA)
0	I	X	I	Registre de sortie (ORB)
I	0	X	X	Registre de controle (CRA)
0	I	X	0	Registre de direction (DDRB)
I	I	X	X	Registre de contrôle (CRB)

Nous rappelons que le KIT D5 dispose d'un connecteur à 24 pins "USER PIA" CONNECTOR qui sert à communiquer avec le PIA 682I. (pour le brochage du "USER PIA" CONNECTOR voir figure 3).

REGISTRE DE CONTROLE
DU PIA

b7	b6	b5	b4	b3	b2	b1	b0
IRQA(B)1	IRQA(B)2	CA2(CB2)			DDR	CAI(CBI)	
INDICATEUR	INDICATEUR	CONTROLE			ACCES	CONTROLE	

bI determine quel front de CAI (CBI) positionnera [illegible] indicateur d'interruption :

bI=0 : de CAI(CBI) → IRQA(B)I

bI=I : de CAI(CBI) → IRQA(B)I

validation/invalidation de la demande d'interruption de CBI (CAI)

b0=0 invalide IRQA(B) vis à vis de la transition active de CAI(CBI).

b0=I valide IRQA(B) vis à vis de latransition active de CAI(CBI).

.Si une demande d'interruption parvient à CAI(CBI) avec b0=0, IRQA(B) sera positionné à la prochaine transition de b0 (de 0 à I).

ndicateur d'interruption positionné à I par les transitions de CAI(CBI). Remise à zéro automatique par l'opération de lecture du registre de sortie ORA(B). Peut être remis à zéro par cablage

ndicateur d'interruption :

si CA2 en entrée : (b5=0) IRQA(B)2 passe de 0 à I à la transition active de CA2(CB2). Remise à zéro automatique par une lecture du registre ORA(B) ou par cablage.

si CA2 en sortie : alors IRQA(B)2 égale à zéro et ne sera pas affecté par les transitions deCA2(CB2).

determine lequel du registre de direction de données ou du registre de sortie est adressé.

b2=0 Registre de direction de données

b2=I Registre de sortie selectionné

CA2(CB2) établi comme sortie

par b5=I

b5 b4 b3

I 0

→ I) **CA2 :**

b3=0 : **signal de lecture avec retour par CAI**

CA2 passe de I à 0 à la transition ↓ de E suivant une opération de lecture du registre de sortie A, puis revient à I par la prochaine transition active de CAL

b3=I : **signal de lecture avec retour par E**

idem sauf que CA2 revient à I par la prochaine transition ↓ de E.

→ 2) **CB2 :**

b3=0 : **signal d'écriture avec retour par CBI**

CB2 passe de I à 0 à la transition ↓ de E suivant une opération d'écriture dans le registre de sortie B, revient à I par la prochaine transi- active de CB2.

b3=I : **signal d'écriture avec retour par E**

idem sauf retour par la prochaine transition ↑ de E

b5 b4 b3

I I → **mise à I ou 0 de CA2(CB2)**

CA2 recopie le bit b3 après chaque écriture par le MPU dans le registre de contrôle.

CA2(CB2) établi comme entrée

par b5=0

b5 b4 b3

0

→ **validation/invalidation de demande d'interruption par CA2(CB2) :**

b3=0 invalide l'indicateur d'interruption IRQA(B) demandée par la transition active de CA2(CB2).

b3=I valide l'indicateur d'interruption demandée par la transition active de CA2.

.Si une transition active de CA2(CB2) intervient pendant une phase d'invalidation, IRQA(B) sera positionné à la prochaine transition > 0 de b3.

→ **détermine la transition active de CA2(CB2) qui positionnera IRQA(B)2**

b4=0 ⟹ CA2 ↓ IRQA(B)2 0 → I

b4=I ⟹ CA2 ↑ IRQA(B)2 0 → I

FIG.2.

La répartition mémoire des registres internes du PIA sur le KIT D5 se présente comme suit :

Adresse	Registre sélectionné
E480	on est dans le port A CRA2=0 , on adresse DDRA CRA2=I , on adresse ORA
E48I	on est dans le port A et exactement dans le CRA
E482	on est dans le port B CRB2=0 , on adresse DDRB CRB2=I , on adresse ORB
E483	on est dans le port B et exactement dans le CRB

Pour mieux saisir l'initialisation du PIA nous allons etudier une application pratique :

Soit à réaliser le programme qui allume des LEDS par rotation circulaire, les LEDS sont disposées à la sortie du PIA (port B) dont l'adresse est E482 (pour DDRB et ORB) et E483 (pour CRB). Le shéma de la naquette est donné par fig.3.

Indications :

- Initialiser le PIA en vue d'obtenir la configuration désirée c.à.d port B en sortie.
- Réaliser le programme en créant un délai de I seconde avant toute nouvelle allumage.

MAQUETTE DE CAIBLAGE PIA LEDS ET INTERRUPTEURS

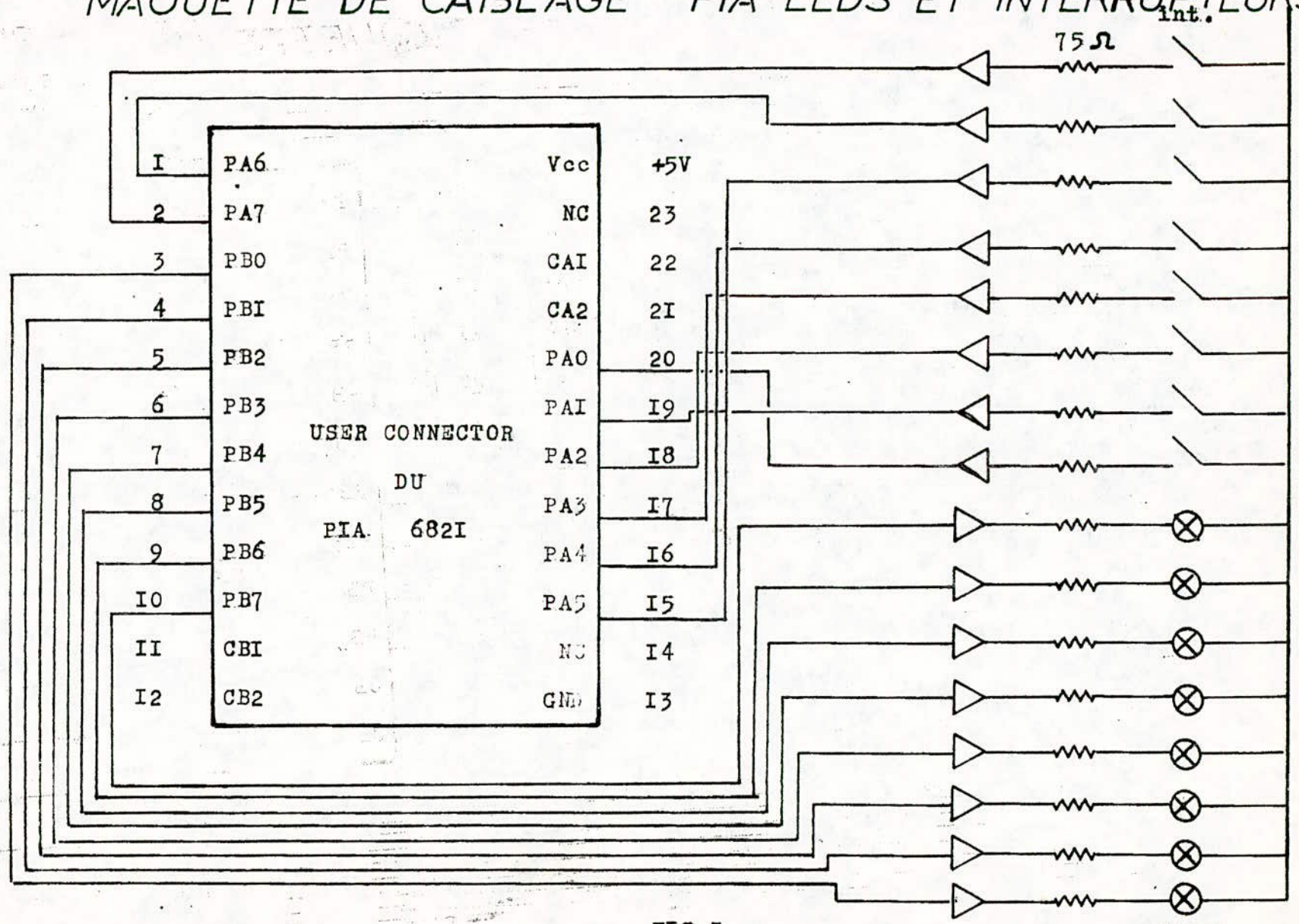
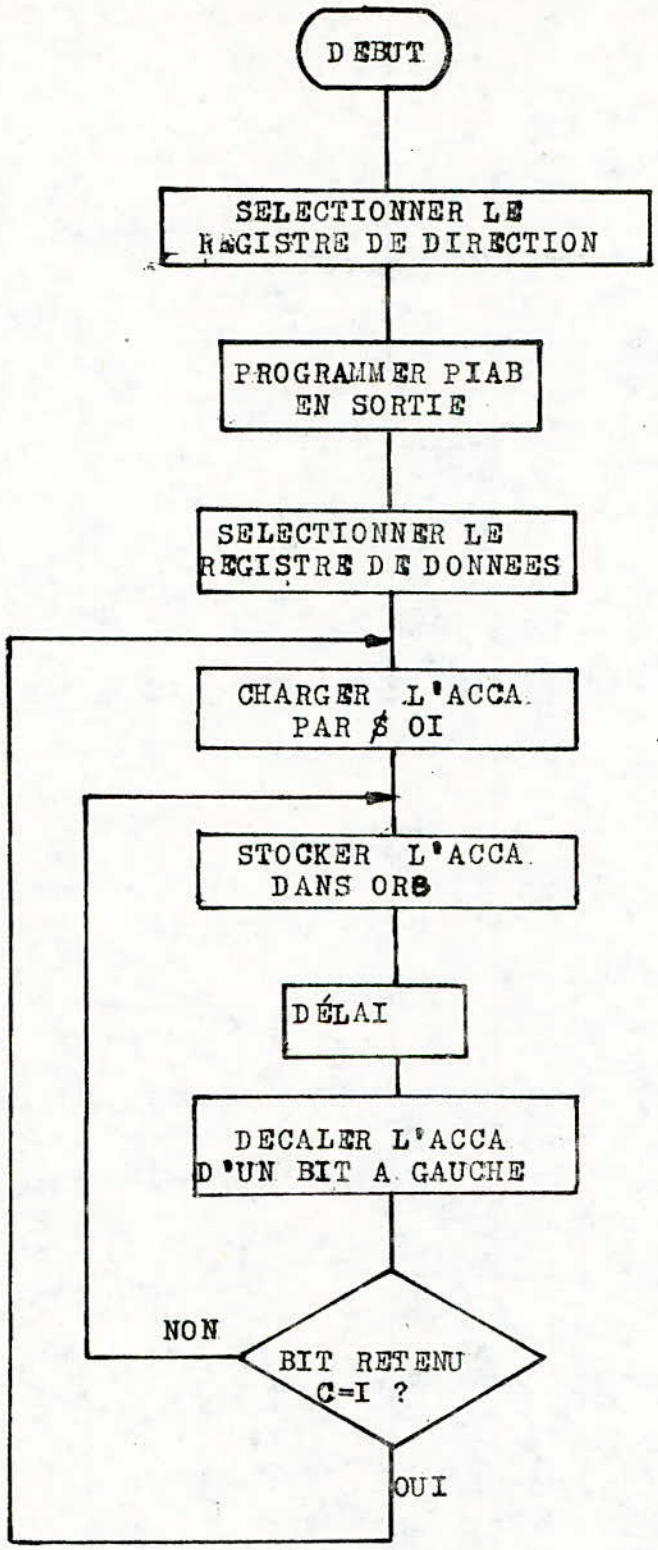


FIG. 3

ORGANIGRAMME :



PROGRAMME :

```
DEBUT     CLR  PIACRB        Sélection du registre de direction
          LDA A # $FF
                             Le PIA B est programmé en sortie
          STA A DDRB
          LDA A # $04
                             Sélection du registre de données
          STA A  PIACRB
ENCORE    LDA A # $0I        Charger l'ACCA par $0I
BOUCLE    STA A PIAORB       Sortie de l'octet sur le port B
          LDX # DELAI
TEMPS     DEX
                             Délai
          CPX # 0000
          BNE  TEMPS
          ASLA               Décaler l'ACCA d'un bit à gauche
          BCS  ENCORE        Test
          BRA  BOUCLE        Saut à BOUCLE .
```

TRAVAIL DEMANDE :

I. Traduisez ce programme en language machine, en chargeant le RX par la valeur qui vous assure un délai de I seconde.

2. Réalisez le cablage

3. Testez le programme sur le KIT D5.

A- INTERFACE PIA LEDS

EXERCICE I: Réaliser un compteur à l'aide du KIT D5 et faire apparaitre le resultat sur les diodes LEDS.

INDICATIONS :

- Utiliser le port B en sortie
- Visualiser le comptage sur les LEDS en créant un délai de I/2 seconde avant toute nouvelle valeur affichée .

TRAVAIL DEMANDE :

I. Faire l'organigramme qui réalise ce compteur
2. En déduire le programme
3. Tester sur le KIT D5

EXERCICE 2 : Les interrupteurs disposés suivant la figure 3 vont vous permettre de réaliser une acquisition de données.

Visualiser sur les LEDS, les valeurs affichées par les interrupteurs

INDICATIONS :

- Utiliser le port A en entrée et le port B en sortie
- Afficher à l'aide des interrupteurs les valeurs 00, FF, AA ...

TRAVAIL DEMANDE :

I. Faire l'organigramme qui réalise cette opération
2. En déduire le programme
3. Tester sur le KIT D5

EXERCICE 3 : Brogrannor, à travers le registre de contrôle CRB , CB2 en node Set/Reset c'est à dire que CB2 suit le bit CRB3. Relier la sortie CB2 à une des diodes LEDS afin de la faire clignoter à des fréquences différentes.

TRAVAIL DEMANDE :

1. Faire l'organigranne de clignotement
2. En déduire le programne
3. Tester sur le KIT D5

TP 7 : LES INTERRUPTIONS

Les interruptions sont en général des signaux venant du monde exterieur au système . Le microprocesseur 6802 traite quatre types d'interruptions classés par ordre de priorité decroissante :

Provenance	Action	Nom	Type
Signaux de l'exterieur	Mise à l'état initial	RESET	Inconditionnelle
	Interruption nonmasquable	NMI	Inconditionnelle
	Interruption masquable (demandée)	IRQ	Conditionnelle
Instruction	Interruption logicielle	SWI	Inconditionnelle

I. LES POINTEURS D'INTERRUPTION POUR 6802 :

Après la prise en compte d'une interruption, il se déroule toujours un programme appelé programme de traitement de l'interruption. Les pointeurs d'interruption sont des cases mémoires reservées et contenant les adresses des programmes de traitement correspondants.

Par construction ces pointeurs sont situés aux adresses suivantes :

.Pointeurs d'interruption.

Nature de l'interruption	Adresse des pointeurs	Contenu des pointeurs
IRQ	FFF8 et FFF9	F7EC
SWI	FFFA et FFFB	F7D2
NMI	FFFC et FFFD	F773
RESET	FFFE et FFFF	F000

2. LES DIFFERENTES INTERRUPTIONS

• Initialisation du microprocesseur (RESET)

Quand le système est mis sous tension ou quand un programme en cours est abandonné, le microprocesseur est prévenu par l'entrée RESET. C'est une rupture de séquence au profit d'un programme privilégié, le moniteur. (voir fig.I)

Pour le KIT D5 , D5 BUG déroule un programme dont l'adresse (F000) est donnée par les pointeurs (FFFE et FFFF).

• Interruption non-masquable (NMI)

Un front descendant sur l'entrée NMI du microprocesseur entraine automatiquement l'éxecution d'une séquence d'interruption, une fois l'instruction en cours terminée.

Le traitement consiste à sauvegarder le contexte programme puis, à interdire momentanément toute interruption IRQ moins prioritaire et enfin à dérouler le programme de traitement indiqué par le pointeur NMI. (voir fig.I)

• Demande d'interruption (IRQ)

Un niveau logique 0 sur l'entrée IRQ du microprocesseur correspond à une demande d'interruption. Avant d'examiner cette demande le microprocesseur finit l'instruction en cours.

La prise en compte est effective si le masque I n'est pas actif (si I=0). Il s'agit de sauvegarder le contexte du programme puis d'interdire momentanément toute interruption IRQ et enfin de dérouler le programme du taitement indiqué par le pointeur IRQ. (voir fig.I)

C'est dans cette catégorie d'interruption que le bit de masque I du registre d'état est employé, deux instructions permettent de le positionner à I ou à 0 :

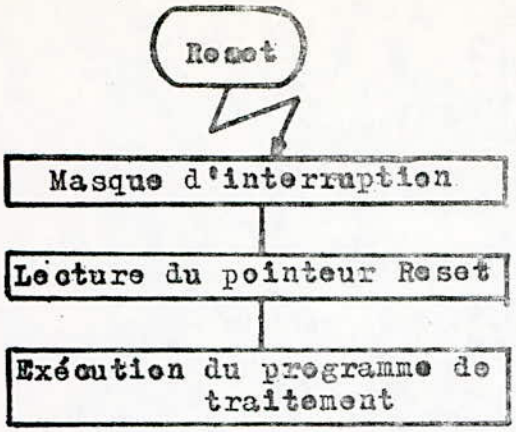
- SEI (SEt Intrrupt mask) met I à "un" et masque donc toute demande IRQ.
- CLI (CLear Interrupt mask) met I à "0" et par conséquent autorise (démasque) toute interruption IRQ.

- Interruption logicielle (SWI)

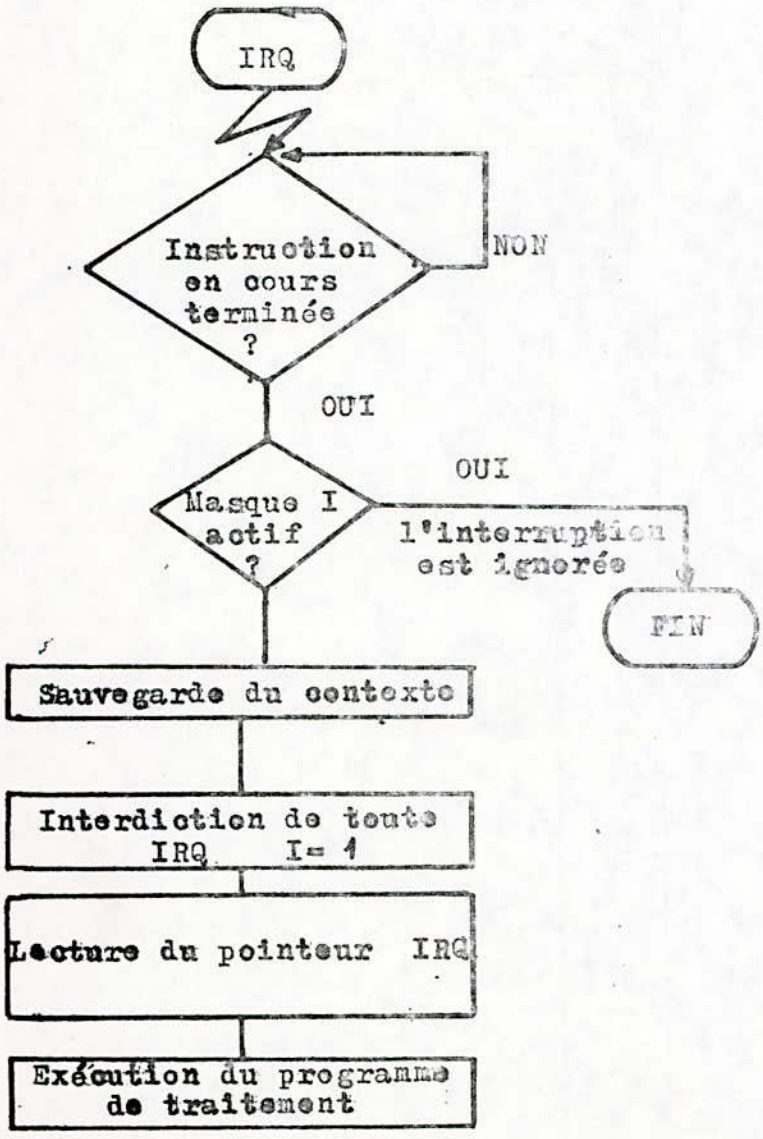
D'une façon intentionnelle, le programmeur peut agir sur le microprocesseur à tout moment dans un programme, afin de dérouler une séquence particulière. A cet effet, il existe l'instruction SWI (Soft Ware Interrupt) qui procède d'une façon analogue aux interruptions matérielles externes. La différence réside seulement dans les petis détails que nous mettons en evidence en comparant les organigrammes des séquences.

Par rapport à la séquence MII, le test de fin de l'instruction en cours est inexistant car SWI est l'instruction qui provoque la séquence correspondante. (voir fig.I)

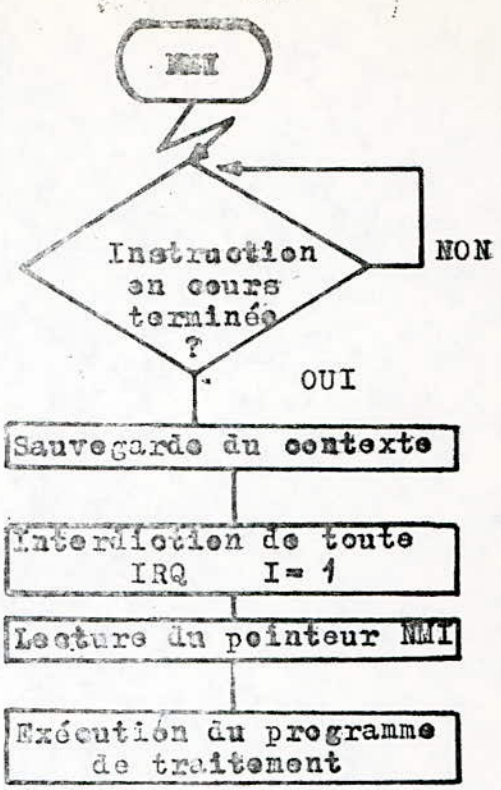
Dans le KIT D5, SWI est reservée à l'abandon ou à la mise au point des programmes en cours. Dans tous les cas le traitement va figer l'état du microprocesseur et du programme interrompu, puis doit afficher le contexte d'avant l'interruption (, ceci permet de savoir dans quelles conditions un programme a été abandonné et d'aider l'utilisateur à mettre au point son programme.



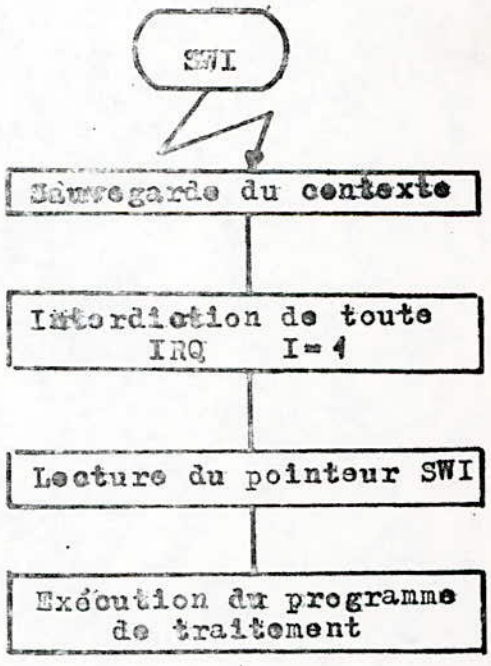
•Séquence d'interruption RESET



•Séquence d'interruption IRQ



•Séquence d'interruption NMI



•Séquence d'interruption SWI

FIG.I

TRAITEMENT D'UNE DEMANDE D'INTERRUPTION : IRQ

I. Principe :

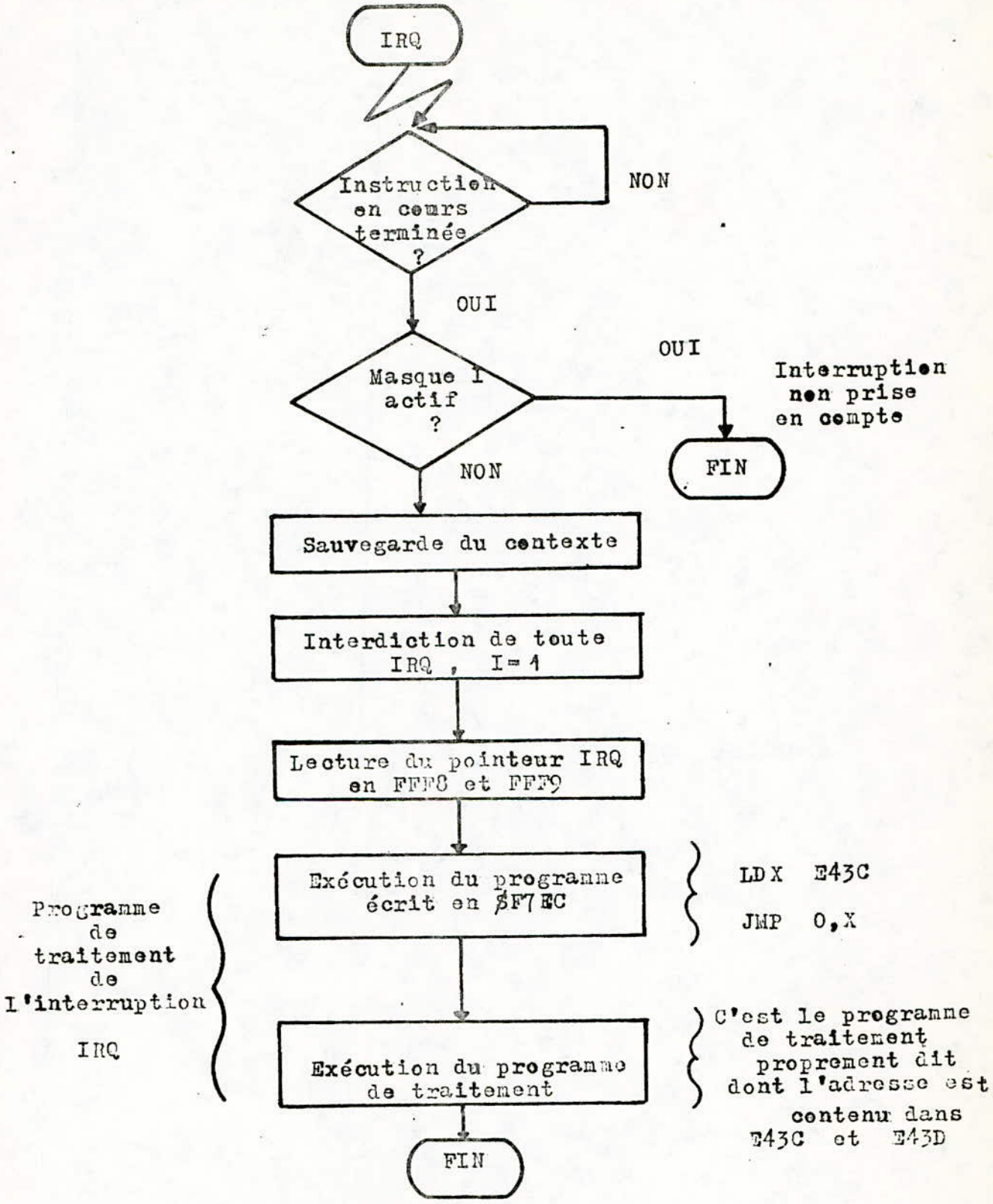
L'interruption IRQ est d'origine externe au système; Nous savons qu'elle peut servir à prévenir le microprocesseur, afin qu'il déroule une séquence d'instructions spécifiques à L'application.C'est le programme lié à l'interruption IRQ ; on le nomme aussi programme de traitement de l'interruption.

Comme nous l'avons vu auparavant, il est possible d'interdire ou non la prise en compte de ce signal de demande d'interruption. C'est L'action de masquage ou non de l'interruption IRQ. Acet effet, il existe le bit I (Interrupt mask) du registre d'état CC , l'autorisation d'une demande IRQ doit se trduire par la mise d'un 0 dans le bit de masquage I.

Sur le KIT D5 à tout signal d'interruption IRQ correspond un pointeur qui renvoie à l'adresse ($ F7EC). C'est le début du traitement de l'interruption par le moniteur D5 BUG (voir listing moniteur D5 BUG) :

```
F7EC   FE E43C      LDX E43C
F7EF   6E 00        JMP 0,X
```

Ces instructions renvoient à leur tour à un programme dont l'adresse est contenu dans les cases mémoires E43C et E43D, le programmeur doit donc initialiser ces cases avec L'adresse de la suite du traitement de l'interruption IRQ.



. Séquence d'interruption externe IRQ

2. Exemple :

Le PIA utilise pour solliciter le MPU la procédure IRQ. Dés qu'un signal d'interruption se presente, le PIA le transmet au MPU sur le fil IRQ. Nous voulons par ce moyen commander la lecture d'un mot sur les fils du PIA.

Soient les données de l'exemple :

- Signal d'interruption reçu sur CAI programmé sur transition I ⟶ 0
- Mot à lire sur PIAA
- Adresse de rangement du code reçu : MEMO
- Adresse de traitement de l'interruption : INTER

L'organigramme de cet interruption est donné par(fig.2)

PROGRAMME :

```
                    . Préparation à l'interruption
        CLR  PIACRA     Sélection DDRA
        CLR  PIDDRA     PIAA en entrée
        LDA A # $05     CAI actif et sélection ORA
        STA A PIACRA
        LDX # $ INTER   Préparation de l'adresse du
                        programme d'interruption
        STX  E43C
        CLI             Autorisation des interruptions
        WAI             Attente de la demande IRQ
        SWI             Fin du programme
                    . Traitement de l'interruption
INTER   LDA A PIAORA    Lecture PIAA
        STA A  MEMO     Rangement donnée
        RTI             Retour d'interruption
```

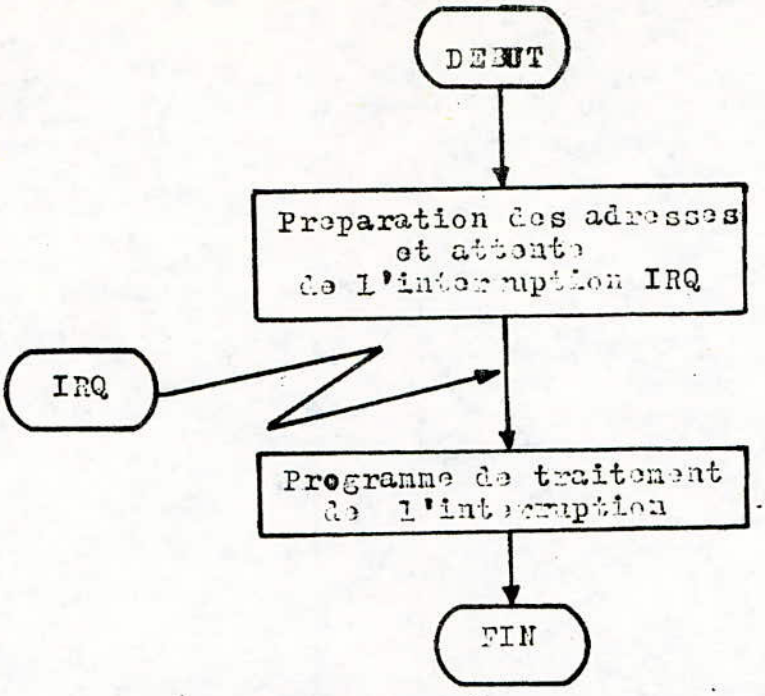


FIG.2 Programmation d'interruption

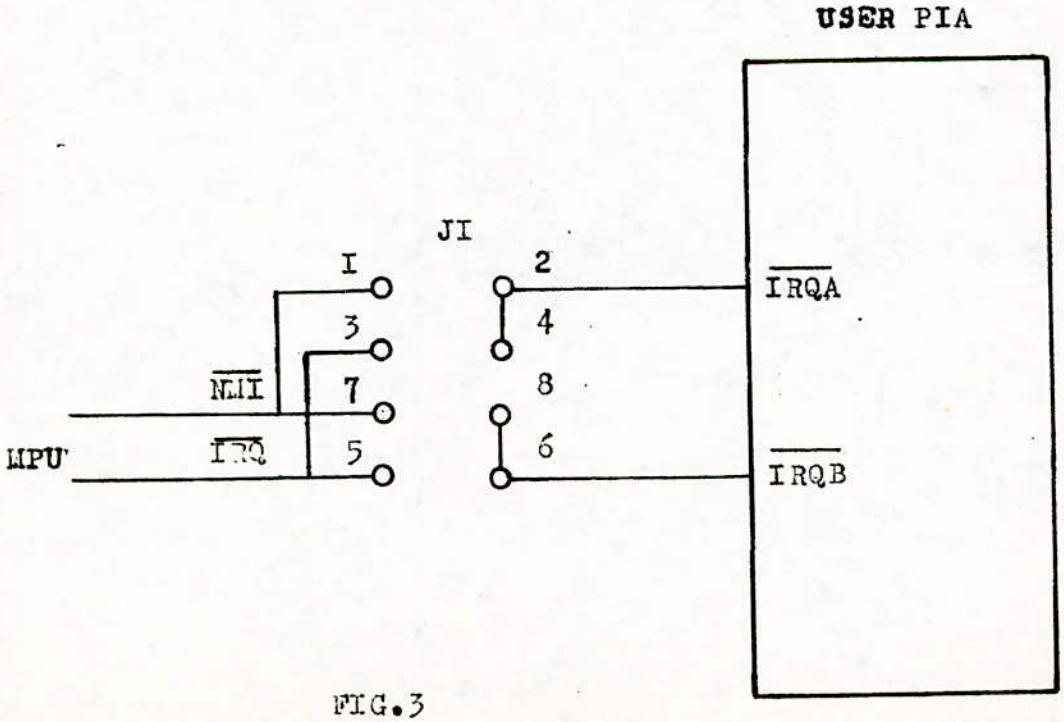


FIG.3

TRAVAIL DEMANDE :

I. Traduisez ce programme en language machine

2. Testez le programme sur le KIT D5

REMARQUE : Les connections entre les lignes d'interruption du "USER PIA" et celle du MPU se font à l'aide de STRAP sur le connecteur JI (voir fig.3)

EXERCICE I : On dispose d'une bascule D 7474

I. Brancher le circuit externe d'interruption (bascule D) sur CAI .

2. Initialiser le PIA de telle façon qu'une interruption présentée sur CAI soit masquée . Vérifier qu'elle n'est pas prise en compte .

3. Démasquer l'interruption au niveau du PIA. Qu'elle précaution doit-on prendre au niveau du MPU pour que l'interruption présente sur CAI soit prise en compte.

- Faites les différents programmes nécéssaires
- Testez-les sur le KIT D5.

EXERCICE 2 : Même questions que l'exercice I en prenant l'interruption NMI.

EXERCICE 3 : Le MPU doit prendre une décision vis-à-vis de deux interruptions demandées en même temps.

Par conséquent il est nécéssaire de faire un programme de telle façon à rendre prioritaire un programme par rapport à un autre

I. Faites un programme de priorité tel que PROG I soit prioritaire sur PROG 2 .

On précise que les deux programmes PROG I et PROG 2 doivent être éxécutés.

Prendre par exemple les deux programmes suivants :

PROG I : Addition de 2 nombres et stockage du résultat dans la case mémoire MEM I

```
LDA A # $03      Charger l'ACCA par  $03
LDA B # $06      Charger l'ACCB par  $06
ABA              Addition A + B
STA A  MEM I     Ranger le resultat
```

PROG 2 : Soustraction de 2 nombres et stockage du résultat dans la case mémoire MEM 2

```
LDA A # $09      Charger l'ACCA Par  $09
LDA B # $05      Charger l'ACCB par  $05
SBA              Soustraction A - B
STA A  MEM 2     Ranger le résultat
```

2. Testez votre programme sur le KIT D5

TP 8 : COMMANDE DE CAN ET CNA A TRAVERS LE "USER PIA"

Le but de ce TP est d'étudier les problèmes d'interfaçage analogique en liaison avec le PIA

Le CAN utilisé est : l'ADC 0804

Le CNA utilisé est ; le DAC 0830

L'ADC 0804 et le DAC 0830 sont des circuits CMOS 8bits, facilement interfaçable avec les microprocesseurs tels que 6800, 6802, Z80 ...

CARACTERISTIQUES

I. ADC 0804 :

- Resolution de 8 bits
- Tension de réference Uref = +5V
- Temps de conversion t_c = 100 µs
- Temps d'accès t_{acc} = 135 ns
- La tension de l'échlon correspondant au bit du poids le plus faible (LSB) ; sera :

$$e = \frac{Uref}{2^n} = \frac{Uref}{2^8} = \frac{5V}{256} \# 19{,}53 \text{ nV}$$

L'erreur commise sur la conversion, est donné par la formule :

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

$$\varepsilon_c \leqslant 9{,}76 \text{ nV}$$

Le brochage et le diagramme des temps sont donnés par (fig.I)

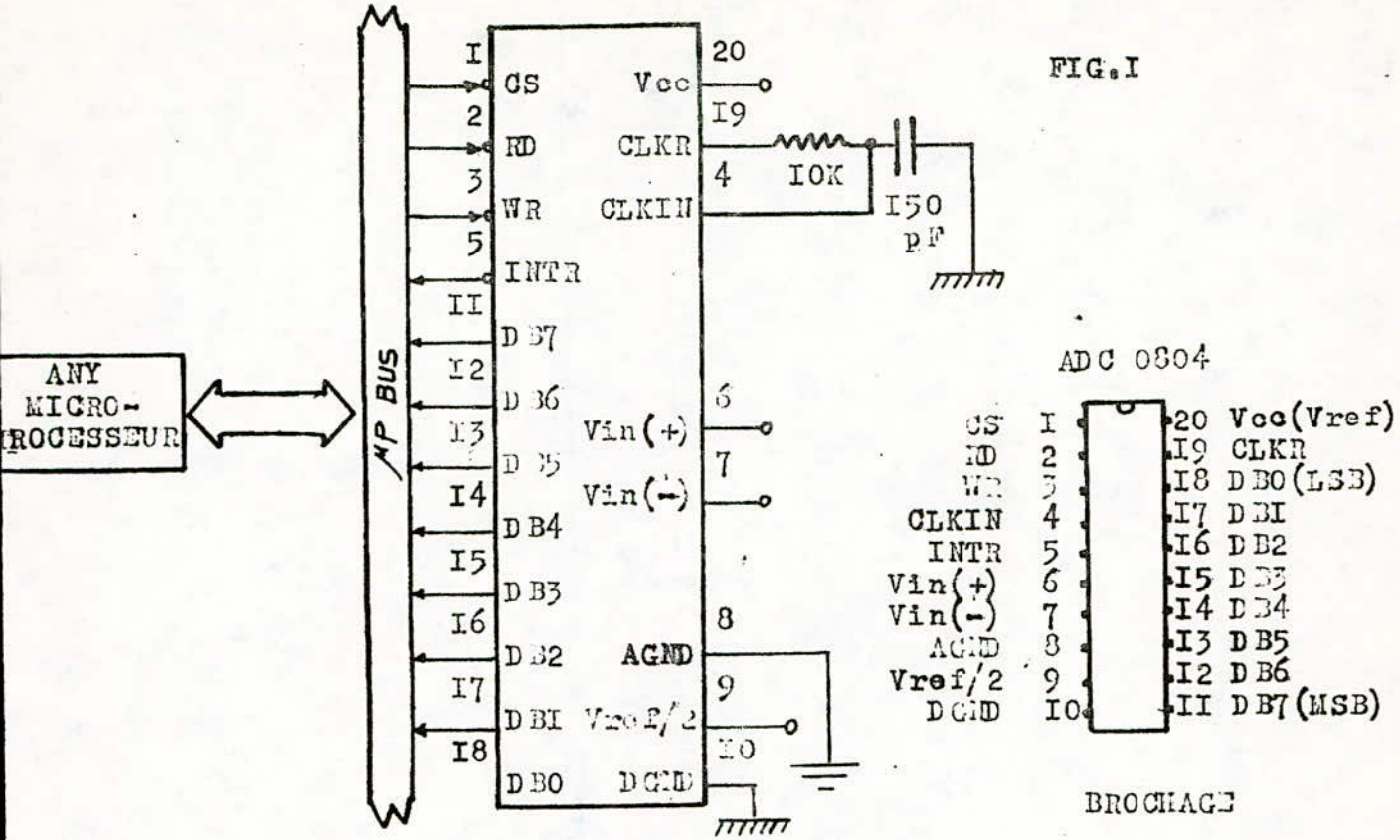
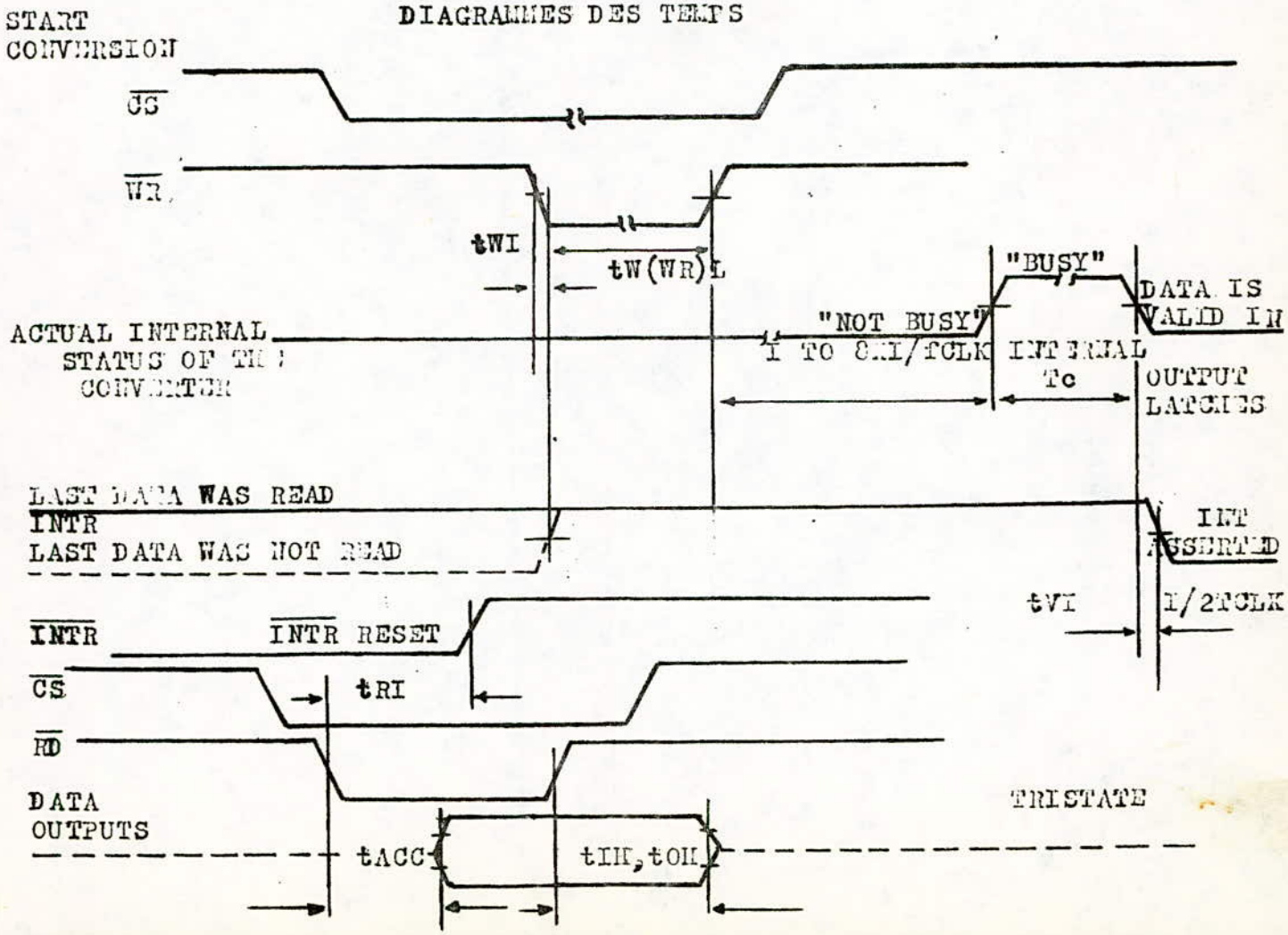


FIG.I

ADC 0804

BROCHAGE

DIAGRAMMES DES TEMPS



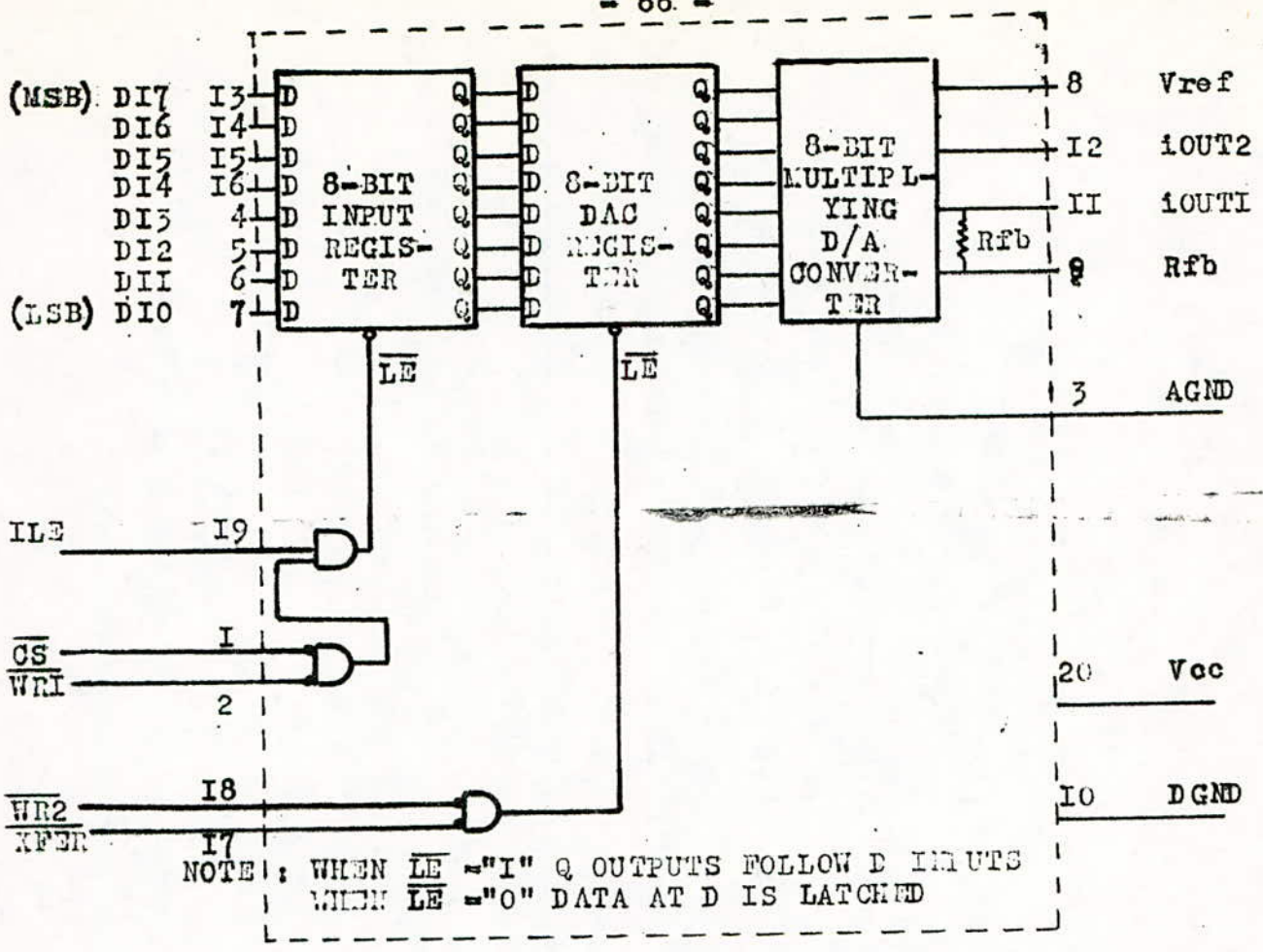
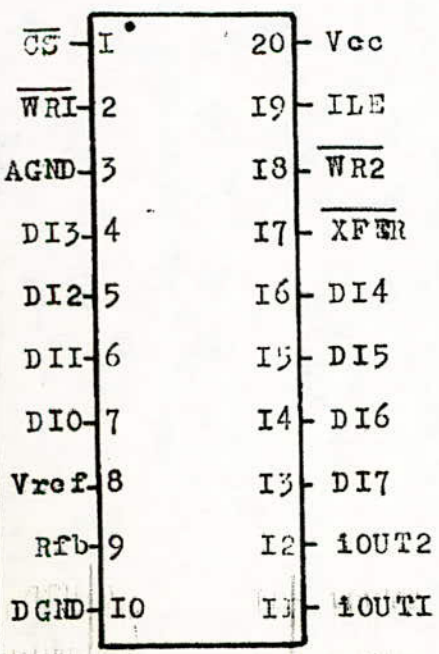
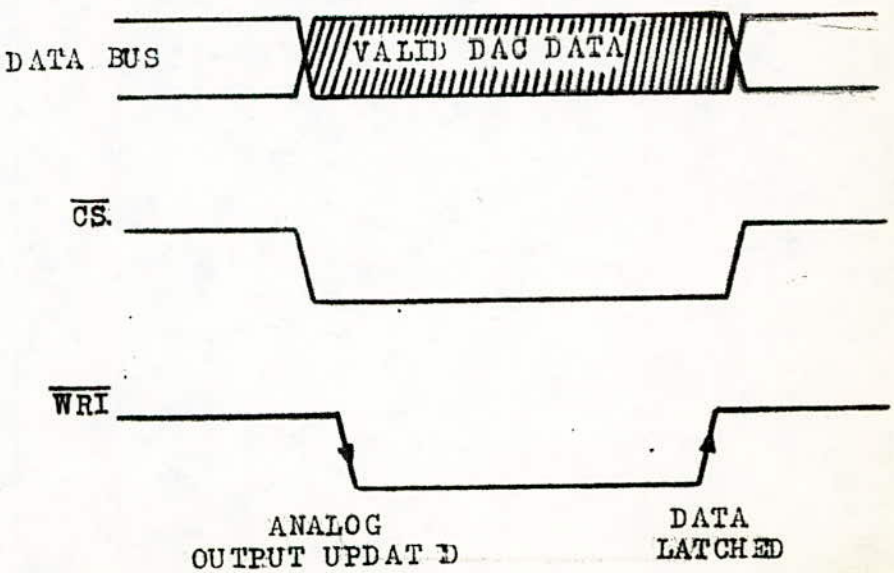


DIAGRAMME FONCTIONNEL DU DAC0830 **FIG.2**



BROCHAGE DU DAC0830

DIAGRAMME DES TEMPS



ILE = LOGIC"I" ; WR2 and XFER GROUNDED

2. DAC 0830 :

- Resolution de 8 bits
- Tension d'alimentation entre +5V et +I5V
- Tension de réference : $-10V \leqslant Uref \leqslant +10V$
- Temps d'établissement : I µs

Le brochage et le diagramme de temps sont donnés par (fig.2)

PROBLEME I : Interface ADC-PIA

La figure 3 presente le shema de cablage .
On se propose d'attaquer l'ADC avec une tension inférieur à +5V en utilisant un potentiomètre PI . Cette tension sera convertie par l'ADC avant d'être transmise au port B du PIA.

MARCHE A SUIVRE :

- Initialiser le PIA (port B) en entrée
- programmer un front actif sur CB2
- Lire les données sur le PIAB
- Stocker les données lues dans des cases mémoires

QUESTIONS :

1. Faire le cablage du montage
2. Faire l'organigramme et le programme de l'acquisition de 5 données (informations) , avec une période de 30 secondes.

Remarques :

- Nous avons choisi une période de 30 secondes de manière à ce que l'étudiant ait le temps de faire varier le potentiomètre pour la prochaine acquisition.

- Les données doivent être stockées aux adresses : 000I, 0002, 0003, 0004 et 0005 . (après execution du programme)

3. Testez votre programme sur le KIT D5. Vérifiez les cases mémoires 000I, 0002, 0003, 0004 et 0005 . Que remarquez-vous ?
4. On veut vérifier les tensions lues sur le voltmètre avec les données stockées en mémoires, pour cela vous deviez convertir les données en BCD. Quelles modifications deviez-vous apporter au programme précedent.

PROBLEME 2 : Interface PIA - DAC

La figure 4 presente le shéma de cablage.

L'utilisation de deux A.O /A 74I permet l'obtention d'une tension de sortie (Vout) bipolaire en accord avec la formule suivante :

$$Vout = Vref \cdot \frac{(DIGITAL\ INPUT)_{10} - I28}{I28}$$

Le MSB du "DIGITAL INPUT" donne le signe du digital.

$$\pm Vout = \pm Vref \times \pm DIGITAL\ CODE$$

La correspondance entre ce qui est affiché sur les pins DI0, DI1,...DI7 du DAC (informations numériques), et ce qui est recupéré en sa sortie Vout (informations analogiques) se résume par le tableau de la figure 5 .

On se propose de convertir des données (informations numériques) stockées aux adresses : 00I1, 00I2, 00I3, 00I4 et 00I5 , en des informations analogiques (tensions) recuellis sur un voltmètre .

INPUT CODE		IDEAL Vout	
HEXA	BINAIRE MSB ..... LSB	+Vref	-Vref
FF	I I I I I I I I	Vref - I LSB	- Vref + I LSB
C0	I I 0 0 0 0 0 0	Vref/2	- Vref /2
80	I 0 0 0 0 0 0 0	0	0
7F	0 I I I I I I I	- I LSB	+ I LSB
3F	0 0 I I I I I I	- Vref/2-ILSB	Vref /2 + I LSB
00	0 0 0 0 0 0 0 0	- Vref	+ Vref

FIG.5

Avec I LSB = $\frac{Vref}{I28}$

QUESTIONS :

I. Faire le cablage du montage

2. Faire l'organigramme et le programme de cette conversion, de manière à ce que toutes les I0 secondes il y a apparition d'une donnée analogique sur le voltmètre.

3. Testez votre programme sur le KIT 85 en prenant Vref = +5V et les données suivantes : (00II) = 80

(00I2) = A0

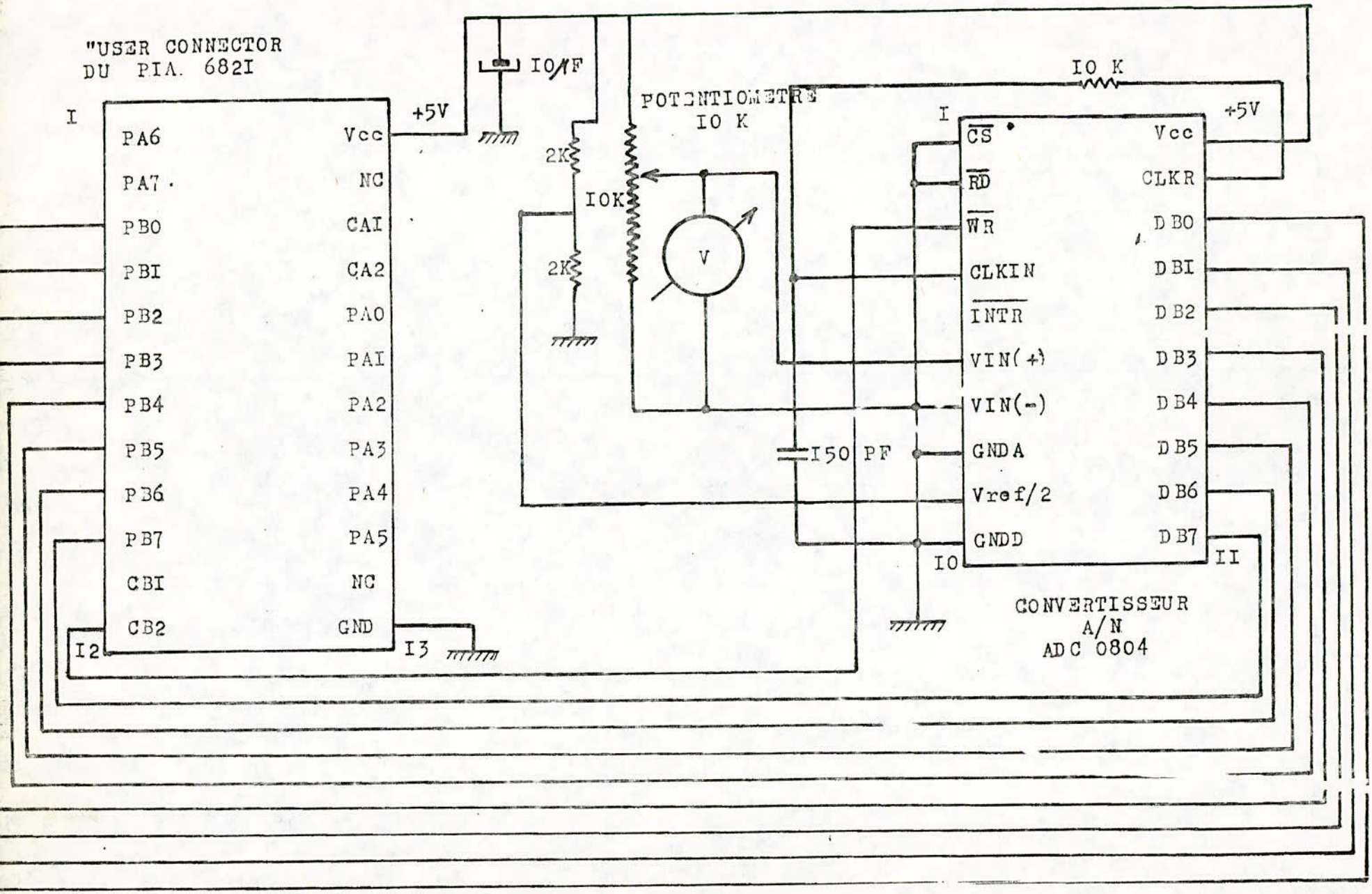
(00I3) = C0

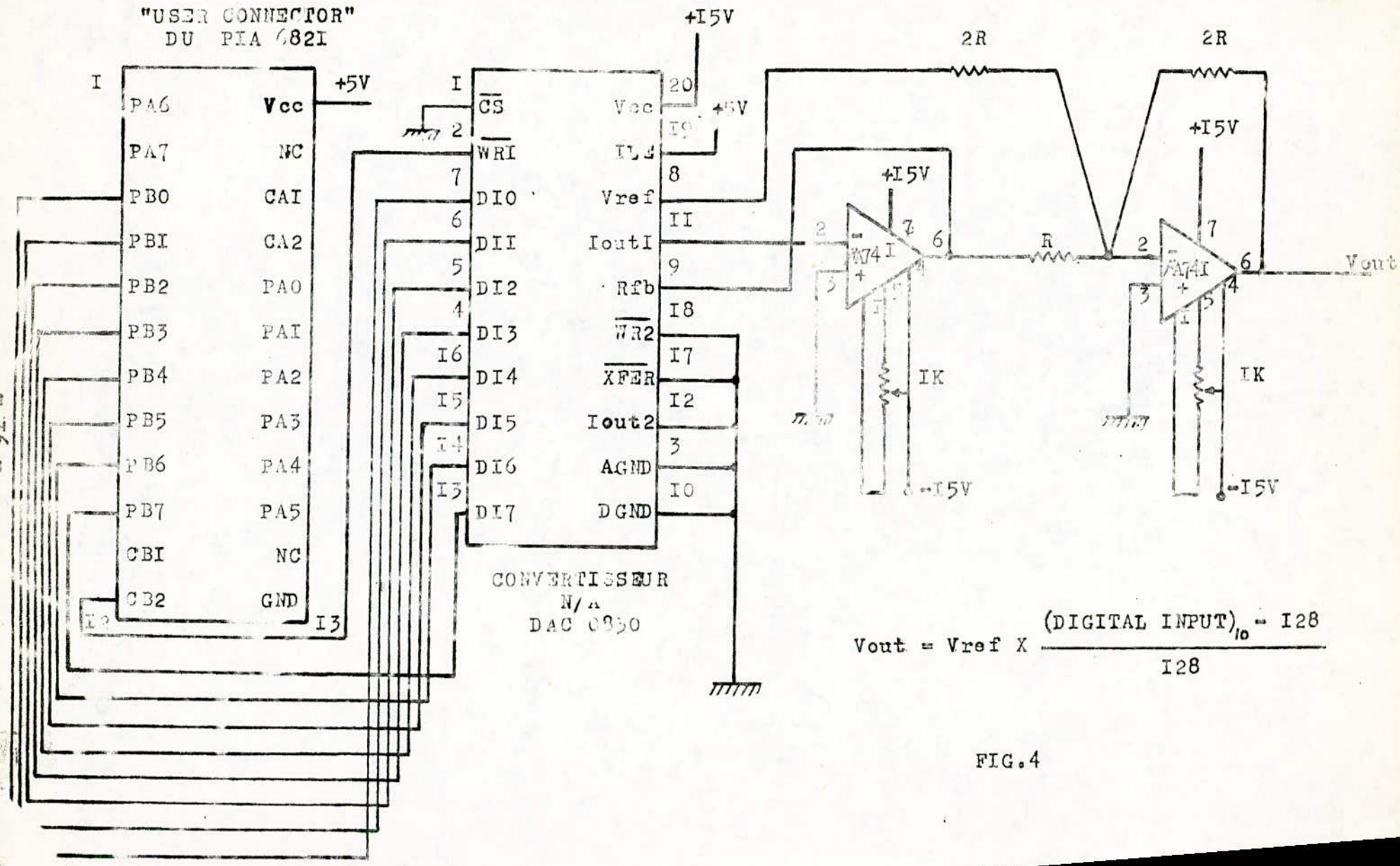
(00I4) = E0

(00I5) = FF

Comparez avec les résultats théoriques du tableau (fig.5)

FIG.3


"USER CONNECTOR
DU PIA. 6821
I
PA6
PA7
PB0
PBI
PB2
PB3
PB4
PB5
PB6
PB7
CBI
CB2
Vcc
NC
CAI
CA2
PA0
PAI
PA2
PA3
PA4
PA5
NC
GND
I2
I3
+5V
I0/1F
2K
I0K
2K
POTENTIOMETRE
I0 K
V
I0 K
I
CS
RD
WR
CLKIN
INTR
VIN(+)
VIN(-)
GNDA
Vref/2
GNDD
Vcc
CLKR
DB0
DBI
DB2
DB3
DB4
DB5
DB6
DB7
+5V
I50 PF
I0
II
CONVERTISSEUR
A/N
ADC 0804



$$Vout = Vref \times \frac{(DIGITAL\ INPUT)_{10} - I28}{I28}$$

FIG.4

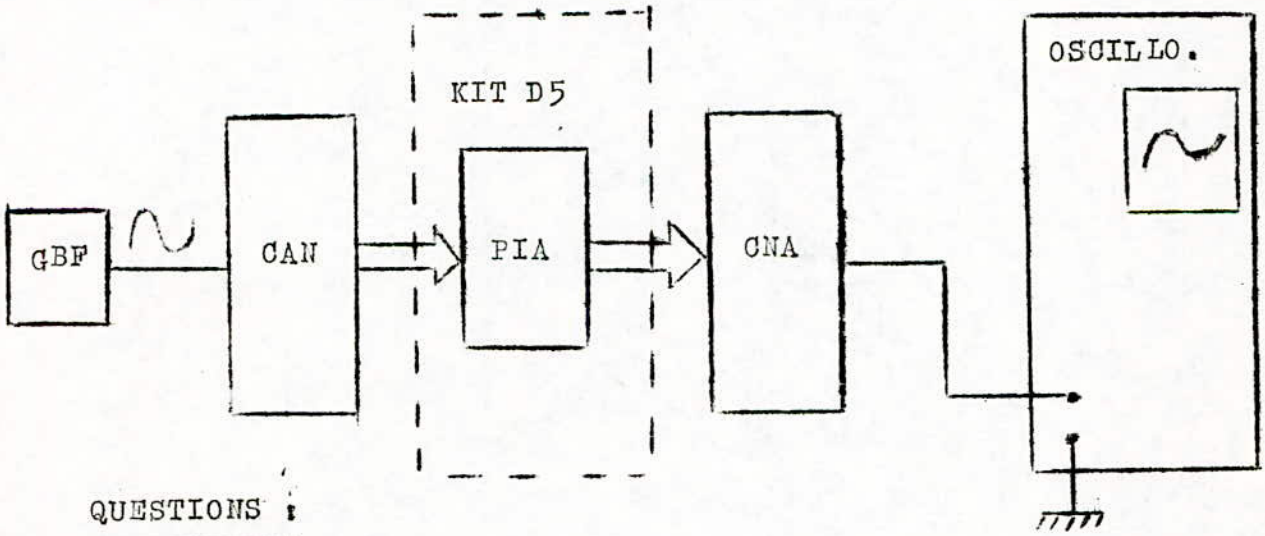
PROBLEME 3 : Interface ADC - PIA - DAC

Nous disposons du matériel suivant :

- Un générateur BF
- Un oscilloscope
- La maquette du TP

Soit le signal sinusoidal (4V , I KHz) en provenance du G.B.F, celui-ci est converti par le CAN (ADC 0804) avant d'être transmis au port A du PIA.La donnée (signal) est numérisée et stockée en mémoire avant d'être retransmise au CNA (DAC 0830) par l'intermédiaire du port B. Elle sera ensuite visualisée sur l'oscilloscope.

Shéma de principe :



QUESTIONS :

1. Réaliser le shéma de cablage
2. Faire l'organigramme et le programme de cette opération
3. Visualiser le resultat sur l'oscilloscope . Conclusion

TP : 9 DECODAGE D'ADRESSES COMMANDE DE CNA ET CAN PAR LE BUS DU MICROPROCESSEUR

OBJECTIF : Le but de ce TP est de :

a) Etudier le décodage d'adresses de deux circuits : ADC 0804 et DAC 0830 à partir d'un décodeur 74LSI38 .

b) Etudier les problèmes d'interfaçage analogique.

Les circuits utilisés :

- CAN : ADC 0804 ; Entrée analogique 0 ⟶ +5V
- CNA : DAC 0830 ; Sortie analogique -5V ⟶ +5V

Les brochages et les diagrammes de temps de ces circuits sont donnés dans le TP 8 .

I - LOGIQUE DE DECODAGE :

Les circuits, tels que ADC 0804 et DAC 0830 sont vus vis à vis du MPU comme des cases mémoires. Comme toute case mémoire, ceux ci doivent avoir une adresse bien spécifique.

Nous voyons donc la nécéssité d'un décodeur d'adresses(74LSI38) comme nous le montre la figure 2.

EXERCICE :

a) Déterminer les adresses de chaque circuits (ADC 0804 et DAC 0830) à partir du schéma global (fig 2) et celui du décodeur (fig I). Conclusion ?

b) Donner le schéma de cablage de manière à avoir le décodage d'adresse suivant : ADC 0804 décodé par AF04

DAC 0830 décodé par AF06.

2 - FONCTIONNEMENT DU ADC :

Nous avons vu dans le TP précédant l'interfaçage du DAC et de l'ADC à travers le PIA. Nous nous proposons d'étudier le fonctionnement de l'ADC connecté directement au bus du microprocesseur comme nous le montre la figure 2.

EXERCICE : Soit le programme suivant qui consiste à convertir une information analogique :

```
STA A   CF00
LDA A   CF01
BPL     FB
LDA A   CF00
STA A   EI00
SWI
```

a) Retrouver l'organigramme de ce programme, en vous aidant des diagrammes des temps du TP 8.

b) Traduiser le programme en language machine et le tester sur le KIT D5.

c) Calculer le temps minimum pour l'éxecution de ce programme.

d) Sachant que l'interruption de fin de conversion (INTR) est lue sur le bit D7 (voir fig. 2), et connaissant la plage de temps de conversion de l'ADC, determiner le nombre de boucles réalisées par le programme.

3 - FONCTIONNEMENT DU DAC :

EXERCICE : Soit le programme suivant qui consiste à convertir une information numérique : (valeur dans EI01)

```
LDA A   E1O1
STA A   CF02
STA A   CF03
SWI
```

a) Retrouver l'organigramme de ce programme, en vous aidant des diagrammes des temps du TP 8.

b) Traduiser le programme en language machine et le tester sur le KIT D5.

c) Calculer le temps minimun pour l'éxécution de ce programme.

4 - AFFINER LA PROGRAMMATION :

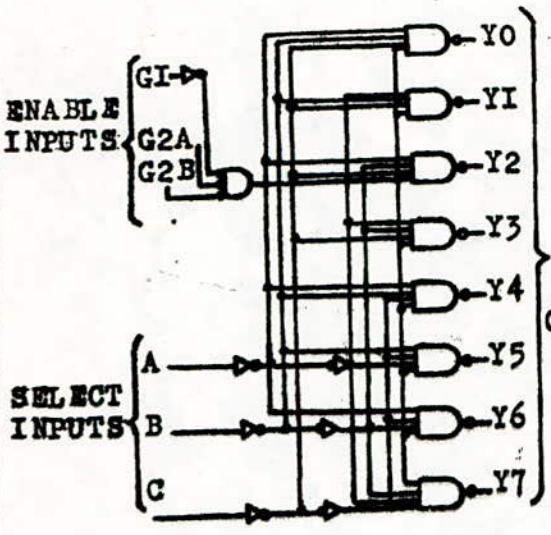
a) Mettre une sinusoide 0 sur l'ADC et la ressortir sur le DAC

b) Mettre un signal triangulaire sur l'ADC et la ressortir sur le DAC .

c) Etudier les défauts de conversion .

d) Remplacer la lecture de la fin de conversion par une temporisation : DLY... du monitour du 6802 (D5 BUG)

e) Un petit traitement de signal : Calculer la tension d'offset de la sinusoide d'entrée, et la ressortir en temps réel.



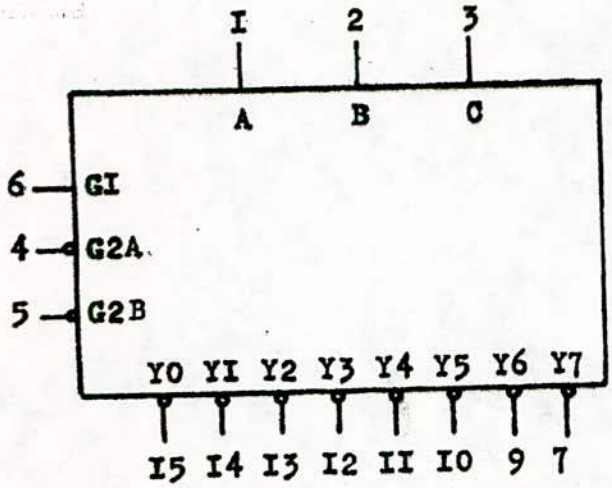
LOGIQUE DU 74I38

INPUTS					OUTPUTS							
ENABLE		SELECT										
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

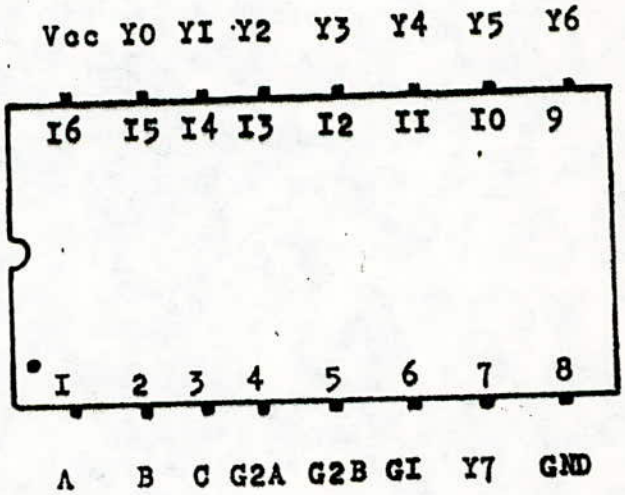
. G2 = G2A + G2B

H=HIGH L=LOW X=Don't care

TABLE DE VERITE DU 74I38



SYMBOLE LOGIQUE DU 74I38



BROCHAGE DU 74I38

FIG.I.

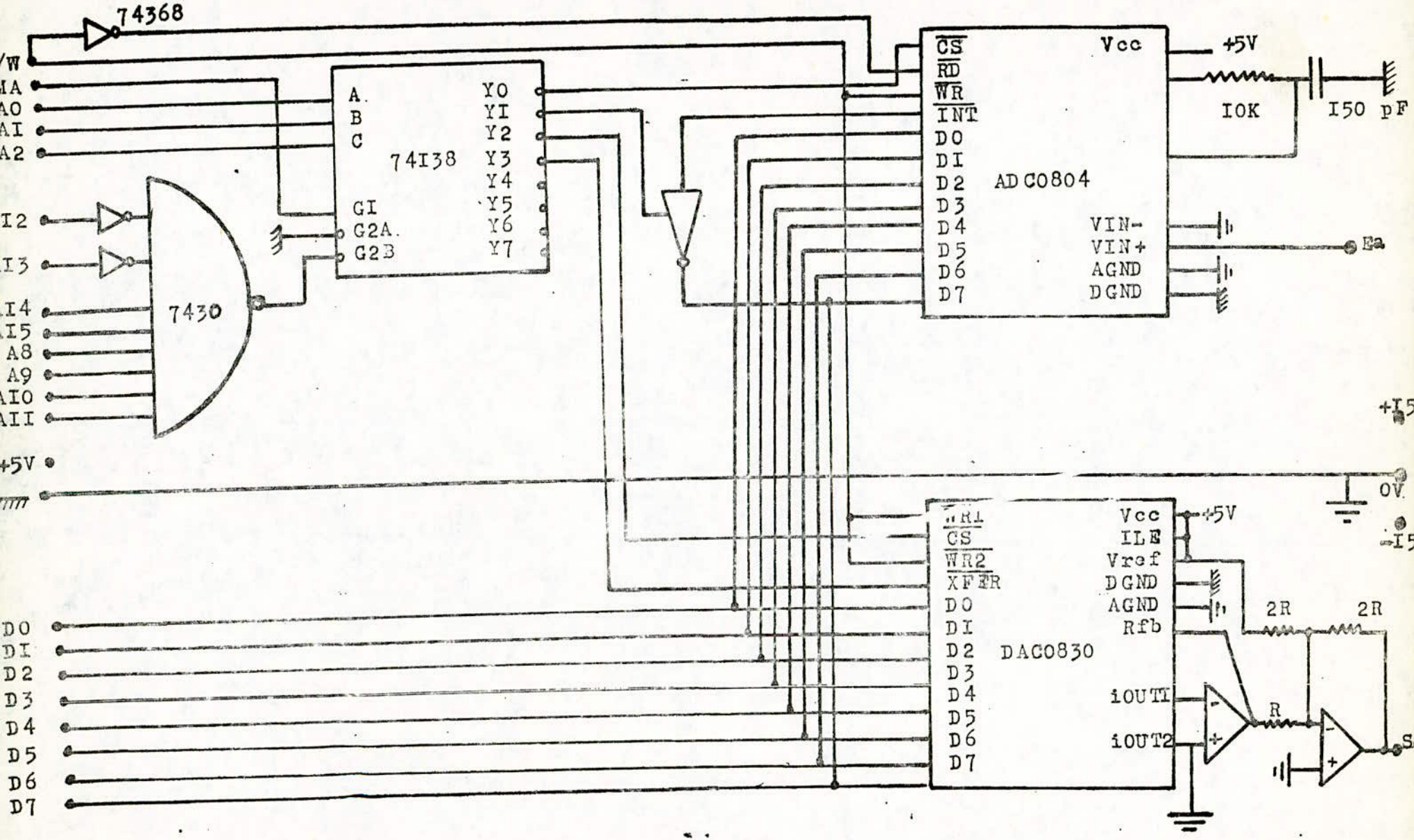


FIG.2. INTERFACE NUMERIQUE-ANALOGIQUE

CHAPITRE IV

APPLICATIONS AVEC LE KIT D5

Nous vous présentons deux exemples d'applications avec le KIT D5 pour guider l'étudiant à concevoir sa propre application pour résoudre un problème donné.

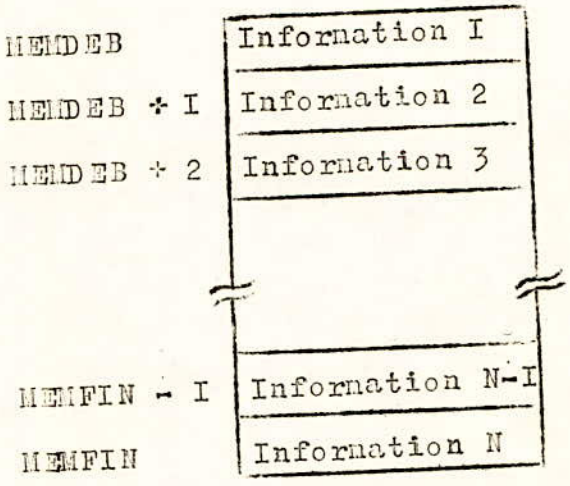
Nous avons choisi comme première application l'interface du KIT D5 avec un périphérique : "la table traçante", et comme deuxième application l'étude et la réalisation d'un processeur musical.

APPLICATION I : Interface du KIT D5 avec un périphérique : "Table Traçante".

Cette application consiste à prendre des informations contenues dans une zone mémoire, les faire sortir à travers le PIA (port B). Ces informations numériques (PB0...PB7) vont attaquer un DAC qui à son tour transformera les informations en des informations analogiques. Ces dernières vont être reproduites sous forme de courbe à l'aide d'une table traçante.

Soit à réaliser l'organigramme , d'en déduire ensuite le programme pour la sortie des courbes sur table traçante.

Les informations à convertir se trouvent dans la zone mémoire comprise entre "MEMDEB" et "MEMFIN".

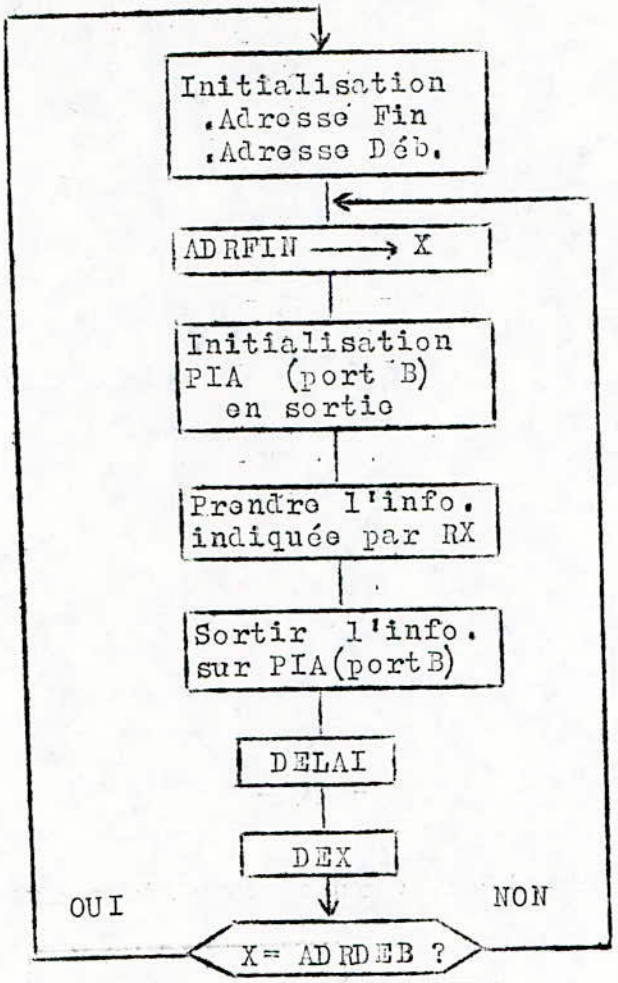


Pour adapter le KIT D5 avec un organe d'enregistrement lent tel que la table traçante il est nécessaire d'utiliser dans le programme un délai adéquat.

Selon ce que l'on met dans la zone mémoire de données, nous pouvons obtenir plusieurs types de courbes.

Pour illustrer notre application nous avons choisi de tracer des fonctions usuelles telles que la sinusoide dans un premier essai dans un second il s'agira du tracé du signal carré, puis ensuite vient le tracé d'une courbe en escalier, et enfin quelques formes géométriques.

I. Organgramme pour sortir une courbe sur table traçante :



2. Programme :

```
DEBUT    0010   CE ....   LDX # MEMDEB      Initialisation
         0013   FF 0000   STX   ADRDEB      Adresses
         0016   CE ....   LDX # MEMFIN
         0019   FF 0002   STX   ADRFIN
BOUCLE   001C   7F E482   CLR   PIADRB      Initialisation
         001F   86 FF     LDAA# $FF         PIA
         0021   B7 E482   STAA  PIADRB
         0024   C6 34     LDAB# $34
         0026   F7 E483   STAB  PIACRB
         0029   A6 00     LDAA  0,X         Prendre l'information
         002B   B7 E482   STAA  PIADRB      Sortir  l'information
         002E   FF 0004   STX   ADRINT
         0031   CE ....   LDX # DELAI       DELAI
ENCORE   0034   09        DEX
         0035   8C 0000   CPX # 0000
         0038   26 FA     BNE   ENCORE
         003A   FE 0004   LDX   ADRINT
         003D   09        DEX
         003E   BC 0000   CPX   ADRDEB      Tracé d'une courbe
         0041   26 D9     BNE   BOUCLE      terminé ?
         0043   20 CB     BRA   DEBUT       Retour
```

3. Exemples de tracé de courbes :

Zone mémoire	Essai I Sinusoide	Essai 2 Signal carré	Essai 3 Escalier
E000	FF	FF	FF
E00I	F0	80	F9
E002	FF	FF	F3
E003	F0	80	ED
E004	FF	FF	E7
E005	F0	80	EI
E006	FF	FF	D B
E007	F0	80	D 5
E008	FF	FF	CF
E009	F0	80	C9
E00A	FF	FF	C3
E00B	F0	80	BD
E00C	FF	FF	B7
E00D	F0	80	BI
E00E	FF	FF	AB
E00F	F0	80	A5

Trois variables vont determiner les formes de ces courbes :

- Base de temps de la table tracante : X
- Sensibilité de la table tracante : Y
- Délai utilisé dans le programme

Lors de nos essais sur la table tracante nous avons utilisé les variables suivantes :

Pour la sinusoide	Pour le signal carré	Pour l'escalier
Délai = 0FFF	Délai = FFFF	Délai = FFFF
X = 0,2s/Cm	X = 0,5s/Cm	X = I s/Cm
Y = 4mV/Cm	Y = 40 mV/Cm	Y = 4 mV/Cm

Les resultats obtenus sont donnés à la figure I.

Des zones mémoires convenablement remplies, avec des variables convenablement choisies permettent d'obtenir les formes géometriques données à la figure 2.

FIG.I

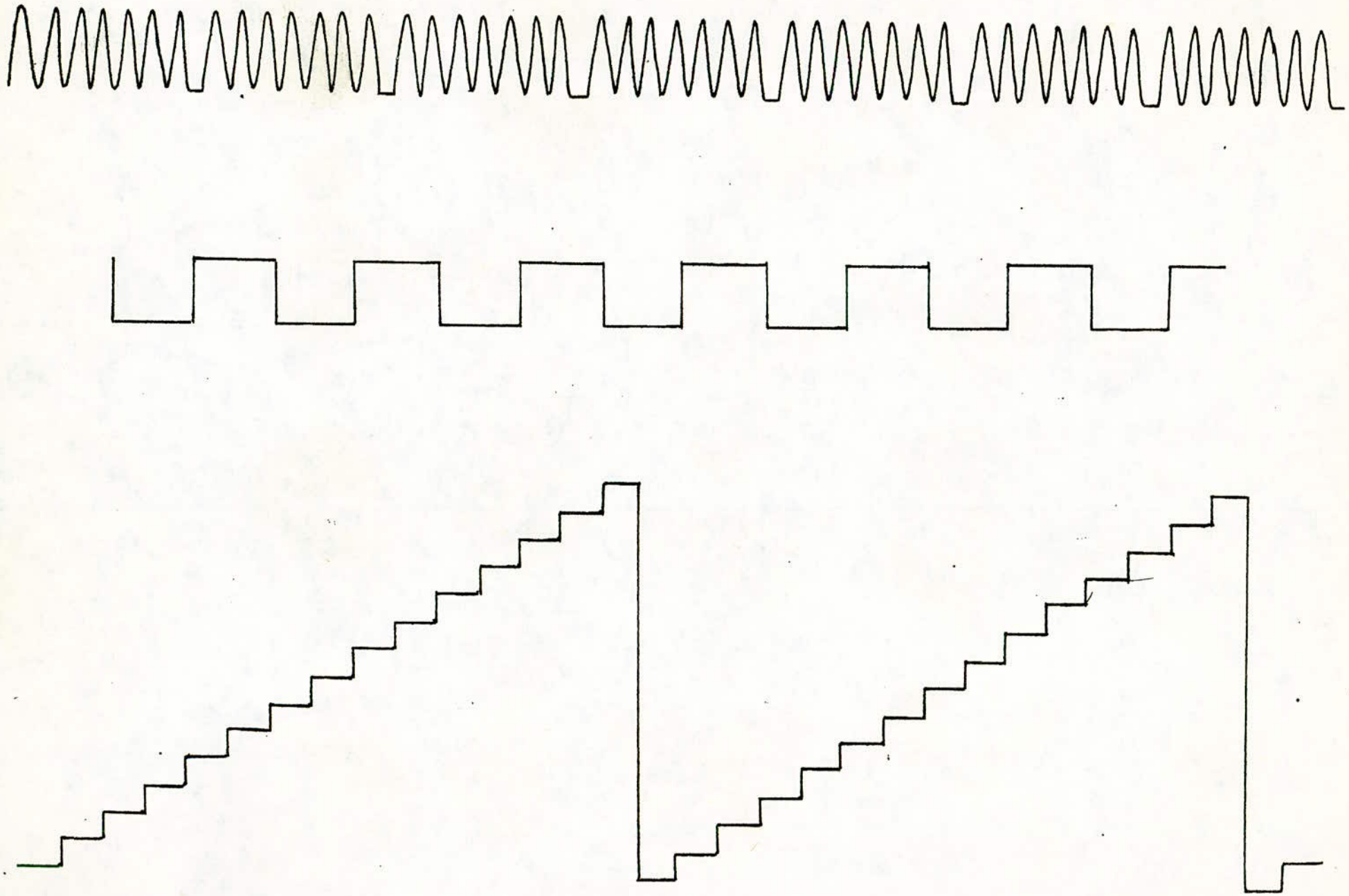
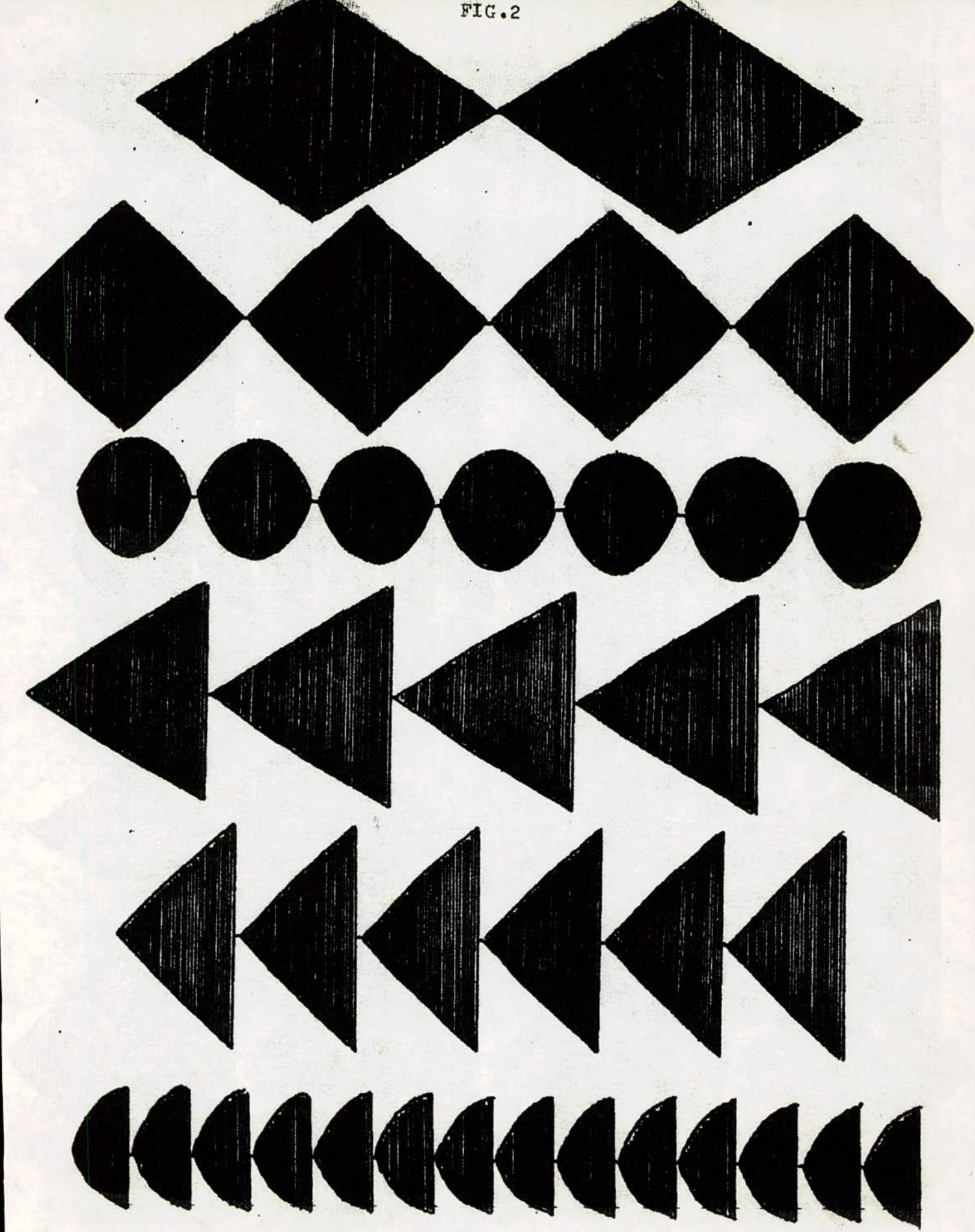


FIG.2

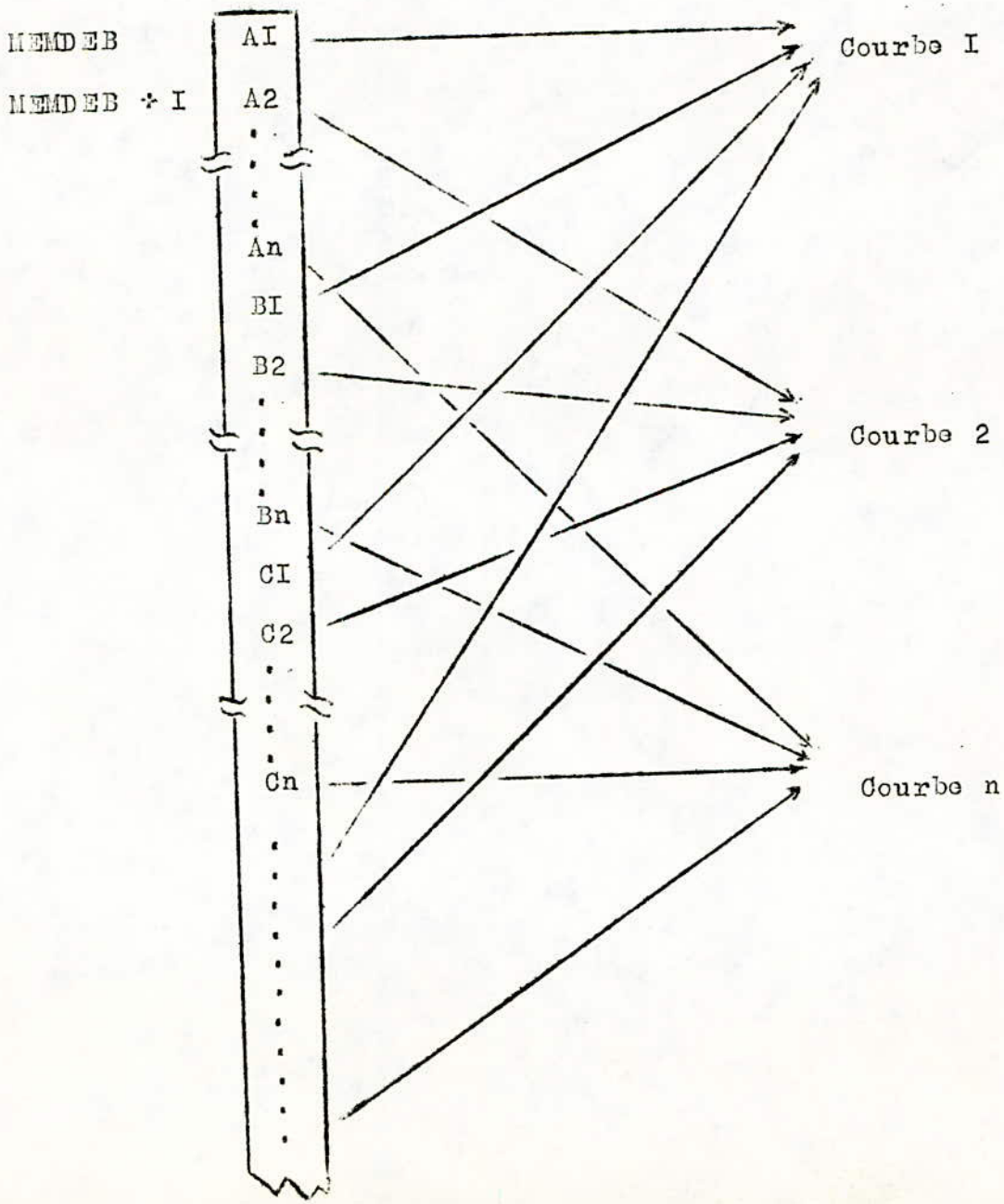


Amélioration du programme :

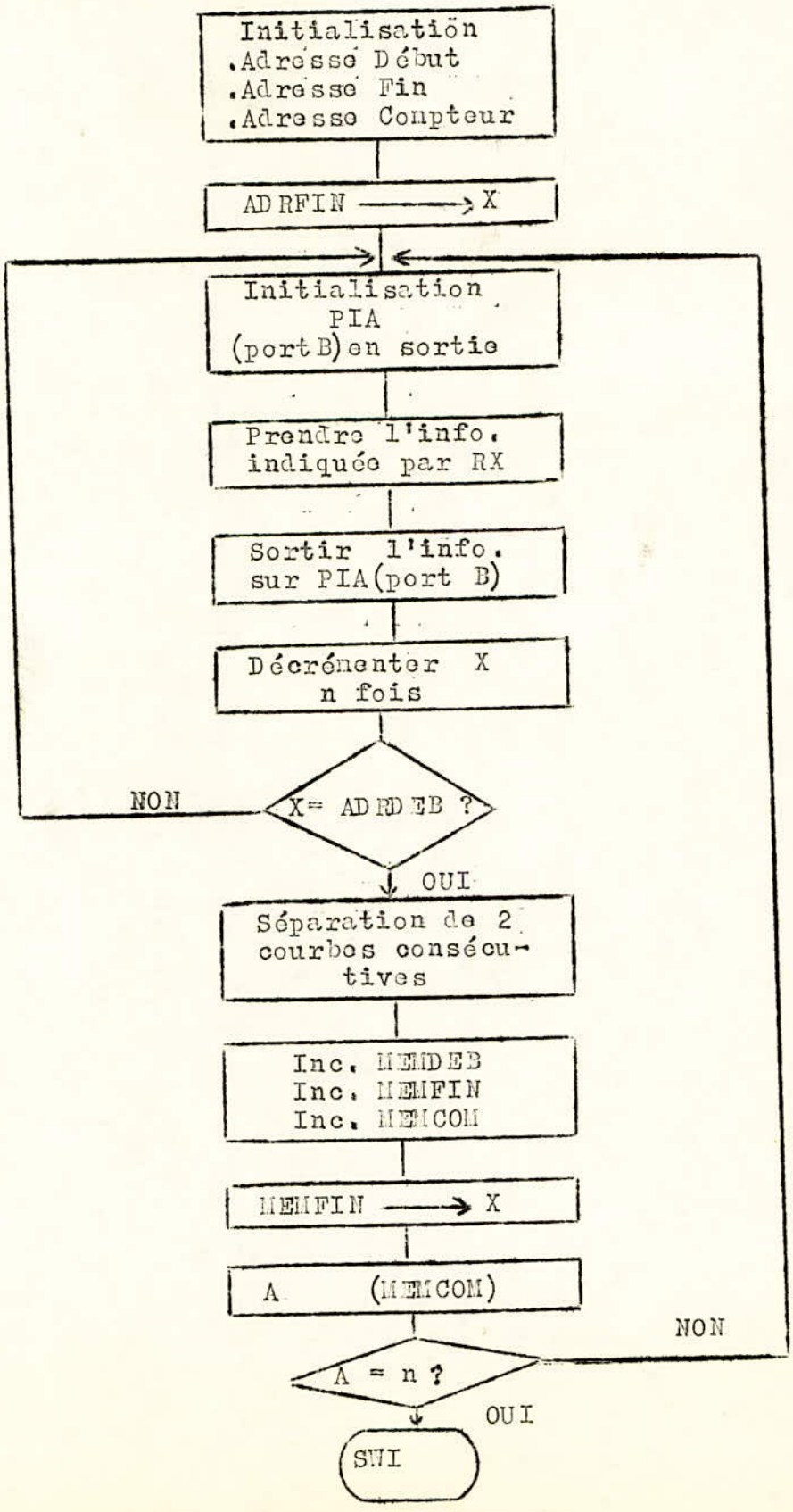
Le programme précédent permet de reproduire seulement une courbe sur la table traçante.

Parfois on a besoin de tracer plusieurs courbes en exécutant le programme une seule fois, pour cela on a pensé à améliorer le programme précédent, de telle façon qu'il sera capable de reproduire une ou plusieurs courbes.

Les informations sont disposées dans une zone mémoires comme suit :



Organigramme pour le tracé de n courbes :



Programme pour le tracé de 3 courbes : (n = 3)

```
DEBUT     00I0  7F 0007   CLR   ADRCOM
          00I3  CE ....   LDX # MEMDEB
          00I6  FF 0000   STX   ADMDEB      Initialisation
          00I9  CE ....   LDX # MEMFIN        Adresses
          00IC  FF 0002   STX   ADRFIN
BOUCLE    00IF  7F E482   CLR   PIADRB
          0022  86 FF     LDAA# $FF
          0024  B7 E482   STAA  PIADRB      Initialisation
          0027  C6 34     LDAB# $34           PIA
          0029  F7 E483   STAB  PIACRB
          002C  A6 00     LDAA  0,X         Prendre l'information
          002E  B7 E482   STAA  PIADRB      Sortir  l'information
          003I  FF 0004   STX   ADRINT
          0034  C6 ..     LDAB# délai
ENCORE2   0036  CE ....   LDX # DELAI
ENCORE3   0039  09        DEX
          003A  8C 0000   CPX # 0000
          003D  26 FA     BNE   ENCORE3         DELAI
          003F  5A        DECB
          0040  CI 00     CMPB# 00
          0042  26 F2     BNE   ENCORE2
          0044  FE 0004   LDX   ADRINT
          0047  09        DEX                 Décrémenter 3 fois
          0048  09        DEX                 le registre d'index
          0049  09        DEX
          004A  BC 0000   CPX   ADMDEB
          004D  26 D0     BNE   BOUCLE
          004F  4F        CLRA                Séparation de 2 courbes
          0050  B7 E482   STAA  PIADRB            consécutives
          0053  CE FFFF   LDX # RETARD
ENCOREI   0056  09        DEX
          0057  8C 0000   CPX # 0000
          005A  26 FA     BNE   ENCOREI
          005C  7C 0007   INC   ADRCOM      Incrementer  ADRCOM
          005F  7C 000I   INC   ADR DEB     Incrementer  ADMDEB
          0062  7C 0003   INC   ADRFIN      Incrementer  ADRFIN
          0065  FE 0002   LDX   ADRFIN      Charger X par ADRFIN
          0068  B6 0007   LDAA  ADRCOM      Charger A par ADRCOM
          006B  8I 03     CMPA# $03         A = 3 ?
          006D  26 B0     BNE   BOUCLE      Saut à "BOUCLE"
          006F  3F        SWI               Fin du programme
```

APPLICATION II

LE PROCESSEUR MUSICAL

A- L'ETUDE THEORIQUE :

I. Fabrication d'une note de musique

2. Fabrication du délai spécifique de chaque note

3. Le temps de base

4. Codage des notes dans une partition

5. Traitement de la partition.

B- REALISATIONS :

I. Organigramme général

2. Interface PIA.

C- EXEMPLE D'INTERPRETATION :

L'éducation sentimentale (musique de Maxime le Forestier), édition de Misère E.I2.M.

I. Analyse de la partition

2. Codage des notes

3. Gestion des couplets

4. Utilisation du programme

5. Mode d'emploi du KIT D5

A - L'ETUDE THEORIQUE

Faire de la musique, c'est en fait émettre successivement des signaux de fréquences différentes.

Ces signaux sont émis de façon courante par des instruments de musique. Ils sont généralement composés d'une fondamentale sinusoïdale et d'harmoniques donnant le timbre de l'instrument. Le signal carré d'un générateur d'impulsions, une fois filtré et amplifié peut produire un son comparable

Or notre microprocesseur, muni de ses cordes vocales numériques, prétend aussi pouvoir fredonner des airs de musique.

I. Fabrication d'une note de musique :

Chaque note est identifiée par sa fréquence donc par une période bien définie, que nous pouvons produire de la manière suivante :

1) Sortie d'un niveau haut sur le PIA;
2) Délai spécifique de la note;
3) Sortie d'un niveau bas sur le PIA;
4) Délai de même durée que le précédent.

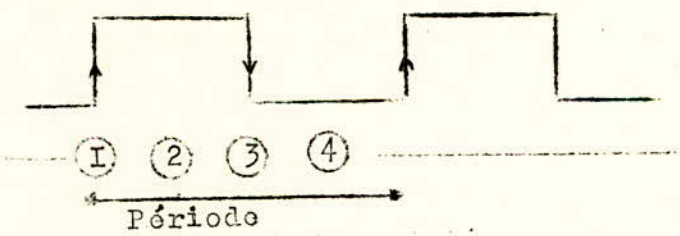


FIG.I.- Signal généré sur le PIA

2. Fabrication du délai spécifique de chaque note :

Le programme de délai mis au point fait intervenir deux variables XX et YY. Un calcul précis, qui tient compte de cycles utilisés par chaque instruction, nous donne la correspondance (fig.2.) :

3. Le temps de base :

La durée d'une note est considérée comme un multiple d'une durée minimale appelée temps de base. Celui-ci doit être le même pour toutes les notes.

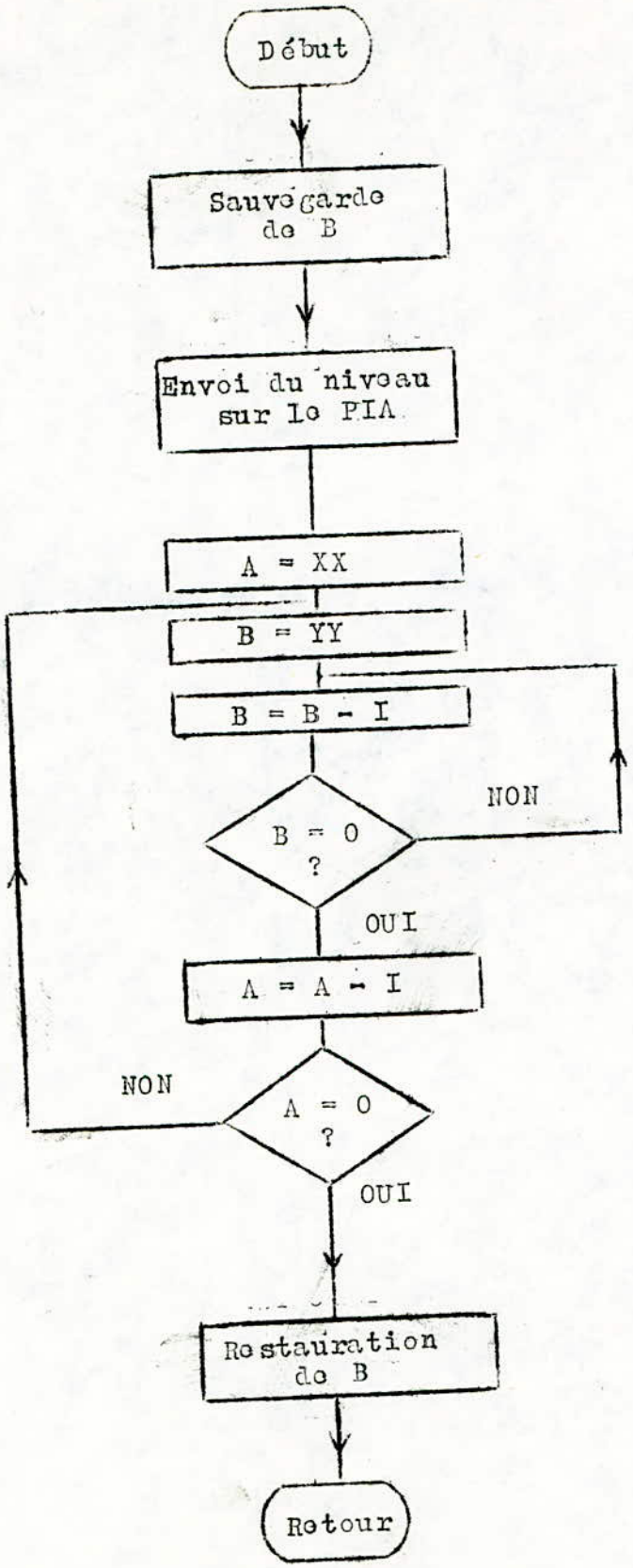


FIG.2. Génération du délai des notes

L'inconvénient du principe énoncé en I. est que pour un nombre égal de périodes, une note aigue (de fréquence plus élevée ou de période plus petite) sera émise en un temps beaucoup plus court qu'une note grave (de fréquence moindre donc de période plus longue).

Pour chaque note, une troisième variable appelée BASE permet de déterminer le nombre de périodes contenues dans un temps de base.

Calcul de BASE :

Soient F la fréquence de la note considérée et t le temps de base (I/I6, I/20, ou I/24 de seconde). Nous avons la relation :

$$t = BASE \times \frac{I}{F}$$

D'où : $\boxed{BASE = Ft.}$

Ainsi le codage, permettant d'identifier une note, porte sur trois variables : XX, YY et BASE. (voir le tableau des notes.)

4. Codage des notes dans une partition :

Pour identifier les paramètres de la note à jouer nous avons décidé d'utiliser le format suivant :

FIG.3- Structure des codes

Ce codage permet d'avoir un jeu de 32 notes (2^5) et huit volumes différents. Les temps sont codés par exemple comme suit :

Temps	Dessin	Nombre de temps de base	Code hexadécimal
Double croche		I	0I
Croche		3	03
Croche pointée		4	04
Noire		6	06
Noire pointée		9	09
Blanche		I2	0C
Blanche pointée		I8	I2
Ronde		24	I8

TABLE.2- Codage des durées de notes

Nom de la note	Fréquence (hertz)	Code Note/ Volume	Adresse dans le tableau des notes TABNOT	Variables spécifiques				
				BASE			XX	YY
				I/24	I/20	I/I6		
Ré 5	II75	07	EI00	3I	3B	49	0I	24
Do 5	I046	0F	EI03	2C	34	4I	0I	2A
Si 4	988	I7	EI06	29	3I	3E	0I	2C
La 4	880	IF	EI09	25	2C	37	0I	33
Sol# 4	83I	27	EI0C	23	2A	34	0I	36
Sol 4	784	2F	EI0F	2I	27	3I	0I	3A
Fa # 4	740	37	EII2	IF	25	2E	0I	3E
Fa 4	698	3F	EII5	ID	23	2C	0I	42
Mi 4	659	47	EII8	IB	2I	29	0I	46
Ré # 4	622	4F	EIIB	IA	IF	27	0I	4B
Ré 4	587	57	EIIE	I8	ID	25	0I	50
Do # 4	554	5F	EI2I	I7	IC	23	0I	55
Do 4	523	67	EI24	I6	IA	2I	0I	5B
Si 3	494	6F	EI27	I5	I9	IF	0I	60
Si b 3	466	77	EI2A	I3	I7	ID	0I	67
La 3	440	7F	EI2D	I2	I6	IB	0I	6D
Sol# 3	4I5	87	EI30	II	I5	IA	0I	74
Sol 3	392	8F	EI33	I0	I4	I8	0I	7B
Fa # 3	370	97	EI36	0F	I2	I7	0I	83
Fa 3	349	9F	EI39	0E	II	I6	0I	8B
Mi 3	330	A7	EI3C	0D	I0	I5	0I	94
Ré # 3	3II	AF	EI3F	0C	I0	I3	0I	9E
Ré 3	294	B7	EI42	0C	0F	I2	0I	A7
Do # 3	277	BF	EI45	0B	0E	II	0I	B2
Do 3	262	C7	EI48	0B	0D	I0	0I	BC
Si 2	247	CF	EI4B	0A	0C	0F	0I	C9
Si b 2	233	D7	EI4E	0A	0C	0F	0I	D5
La 2	220	DF	EI5I	09	0B	0E	0I	E2
Sol# 2	208	E7	EI54	09	0A	0D	0I	F0
Sol 2	I96	EF	EI57	08	0A	0C	0I	FF
Fa 2	I75	F7	EI5A	07	09	0B	02	8E

TABLE.I. Paranétrage des notes

Le volume a huit niveaux codés par les trois bits du prenier octet.

Exemples : Volume 2 ⟹ [0 I 0]

Volume naxinal ⟹ [I I I]

Silence (volume nul) ⟹ [0 0 0]

5. Traitement de la partition :

Notre partition une fois traduite en code hexadécimal est une suite d'octets.

Exemple : 7F 03 représente un La 440 de la durée d'une croche en volume naxinal

La partition codée est pointée par un index appelé IPAR (Index de PARtition), qui balaie toute la nénoire partition lors de l'exécution du norceau de musique proposé.

Or de nonbreux airs possèdent des refrains. Il est donc souhaitable d'avoir un outil simple pour éviter la recopie des séquences identiques en nénoire. Nous sonnes appelés à convenir d'un code spécial réservé aux SAUTS à l'intérieur d'une partition :

Mnémonique	Connande (nacro-instruction	Codage prenier Octet	Codage deuxième Octet	Signification de la connande
SAR	Saut arrière	FC	déplacement Δ	IPAR - (2Δ - 2) ⟶ IPAR décrémente PRECOM.
SAV	Saut avant	FD	déplacement Δ	IPAR + (2Δ - 2) ⟶ IPAR décrémente PRECOM.
PCS	Prise en compte des sauts	FE	inutilisé	Cette connande incrémente la variable PRECOM, tandis que les autres la décrénentent. Les sauts ne sont autorisés que dans la nesure où PRECOM est nulle. Sinon la partition est exécutée en séquence.
FIN	Fin de la partition	FF	inutilisé	On arrête le programne.

TABLE.3. Macroinstructions

Ces 4 connandes introduisent un language de programnation ayant pour support la partition. Connne elles sous-entendent des traitements à effectuer, nous les nonnons des "nacroinstructions".

Le problème revient à établir pour chaque partition un organigramme. Nous disposons alors de 5 éléments différents qui en feront partie.

FIN	Fin de la partition
PCS SAV SAR	Prise en compte des sauts Saut avant Saut arrière
P.X	Exécution d'un élément de partition ne contenant aucune macroinstructions

TABLE.4. Eléments d'une partition

B - REALISATIONS

I. Organigramme général : (voir figure 4)

2. Interface PIA :

Le programme de la figure 4 peut émettre des sons de volume réglable. Il existe 8 niveaux, codés de 0 à 7.

Il s'agit d'injecter un courant variable avec le niveau choisi, dans la charge représentée par le haut-parleur de 8Ω.

Les circuits 7404 à collecteur ouvert sont adaptés à ce genre de montage. Les sorties sont réunies au même point M qui est ensuite relié à une alimentation de 5V à travers le haut-parleur.

V Volume	Sorties PIA B (I= actif 0= repos) PB6	PB5	PB4	PB3	PB2	PBI	PB0
0	0	0	0	0	0	0	0
I	0	0	0	0	0	0	I
2	0	0	0	0	0	I	I
3	0	0	0	0	I	I	I
4	0	0	0	I	I	I	I
5	0	0	I	I	I	I	I
6	0	I	I	I	I	I	I
7	I	I	I	I	I	I	I

TABLE.5. Modulation du volume

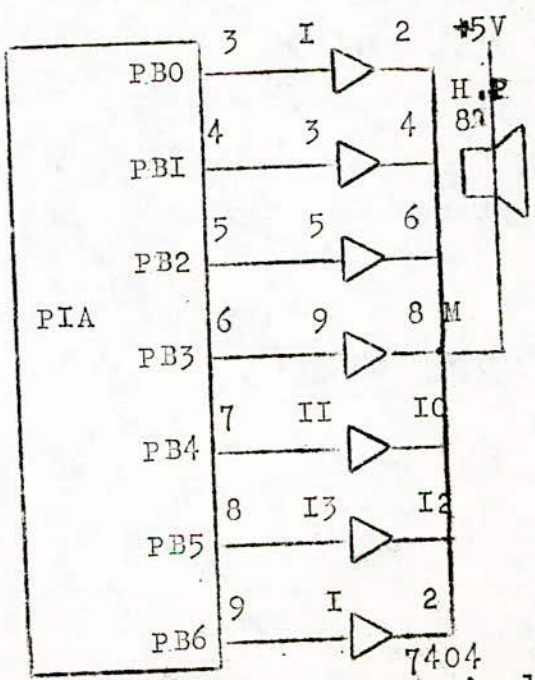
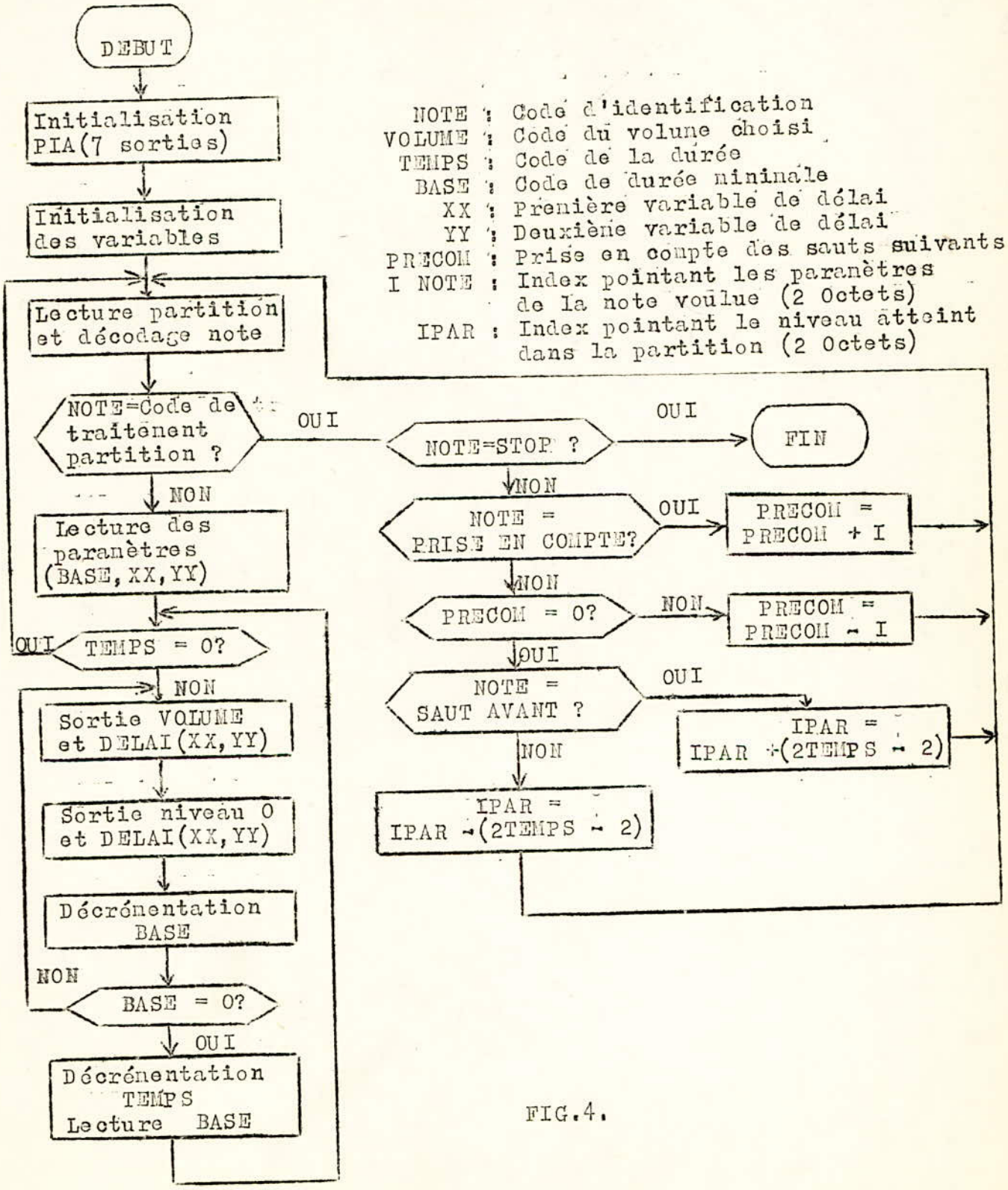


FIG.5 Processeur musical: Interface PIA

Organigramme général :



NOTE : Code d'identification
VOLUME : Code du volume choisi
TEMPS : Code de la durée
BASE : Code de durée minimale
XX : Prenière variable de délai
YY : Deuxième variable de délai
PRECOM : Prise en compte des sauts suivants
I NOTE : Index pointant les paranètres de la note voulue (2 Octets)
IPAR : Index pointant le niveau atteint dans la partition (2 Octets)

FIG.4.

C - EXEMPLE D'INTERPRETATION

L'éducation sentimentale (musique de Maxime le Forestier), édition De Misère E.I2.M

I) Analyse de la partition :

Nous pouvons à partir de la partition originale, reconnaitre l'existence de trois blocs que nous baptisons PA, PB, PC.

L'élément PA constitue le couplet, tandis que PB et PC sont les enchainements. Le couplet PA se répète quatre fois; PB enchaine les trois peniers couplets tandis que PC finit la mélodie.

2) Codage des notes :

Al'aide des tableaux de paramétrage, il est simple de coder la patition. Une fois celle-ci effectée, séparons les couplets des enchainements.

3) Gestion des couplets :

Avec les macro instructions, il est relativement facile de reprendre un couplet ou un enchainement.

Traitons en détail la gestion des élément de cette partition.

Les macroinstructions déterminent la valeur de la variable PRECOM à chaque instant. Consultons le tableau de la figure 6. Les pointillés horizontaux correspondent à l'action des macroinstructions sur la variable PRECOM.

Premier passage :

Les trois premières"macros" PCS mettent 3 dans PRECOM.
Le couplet PA est joué une fois.
SAV n'est pas exécutée, mais décrémente PRECOM.
L'enchainement PB est joué une fois.
SAR I n'est pas exécutée, mais décrémente PRECOM. De même pour SAR 2.
SAR 3 est autorisée puisque PRECOM est nul (prise en compte).Le branchement I est réalisé en PCS 2.

Deuxième passage :

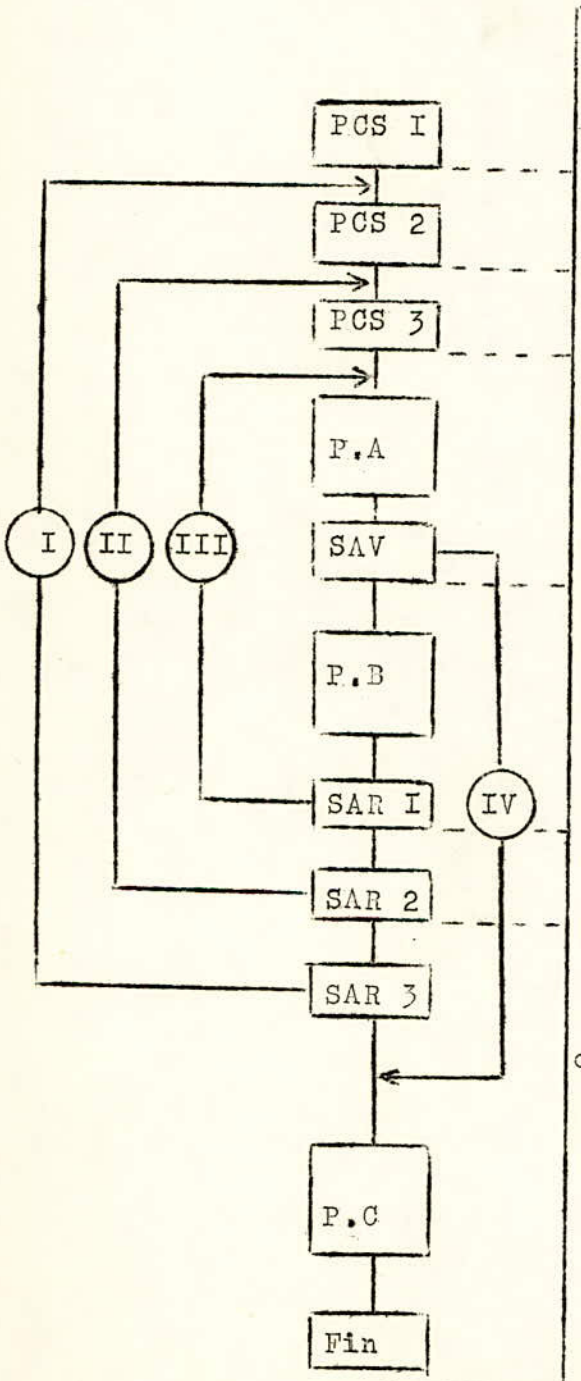
Cette fois-ci, PCS 2 et PCS 3 mettent 2 dans PRECOM.
Donc, après l'exécution du couplet PA at de l'enchainement PB, les macros SAV et SAR I vont autoriser SAR 2 en décrémentant PRECOM. Le branchement II est réalisé en PCS 3.

Troisièmepassage :

La macro PCS 3 est suivie du couplet PA qui est exécuté. PCS 3 et SAV autorisent le branchement III après exécution de PB pour la troisième fois.

Quatrième passage :

La partie PA est exécutée pour la quatrième fois.
La macro SAV est exécutée pour la première fois. Le saut en avant IV permet de finir la partition avec PC.



	Valeur de PRECOM			
	Premier passage	Deuxième passage	Troisième passage	Quatrième passage
PCS I	I	0		
PCS 2	2	I	0	
PCS 3	3	2	I	0
SAV	2	I	0	macro exécutée
SAR I	I	0	macro exécutée	
SAR 2	0	macro exécutée		
SAR 3	macro exécutée			
Branchement exécuté	I	II	III	IV

FIG.6. Utilisation des macroinstructions

Début	ADRESSE	CODE	NOTE ou macro	ADRESSE	CODE	Note	
Début	EI60	FE 00	PCS	EIC0	6F 03	Si	
	EI62	FE 00	PCS	EIC2	8F 09	Sol	
	EI64	FE 00	PCS	EIC4	8F 03	Sol	
	EI66	A7 03	Mi	EIC6	7F 03	La	
	EI68	9F 03	Fa	EIC8	8F 03	Sol	
	EI6A	A7 03	Mi	EICA	8F I2	Sol	
	EI6C	B7 03	Ré	EICC	87 03	Sol#	
	EI6E	C7 03	Do	EICE	8F 03	Sol	
	EI70	CF 03	Si	EID0	8F 03	Sol	
	EI72	DF 03	La	EID2	7F 03	La	
	EI74	E7 03	Sol#	EID4	6F 03	Si	
	EI76	DF 03	La	EID6	8F 03	Sol	
	EI78	CF 03	Si	EID8	7F 03	La	
	EI7A	C7 03	Do	EIDA	87 03	Sol#	
	EI7C	DF 03	La	EIDC	7F 03	La	
	EI7E	B7 03	Ré	EIDE	6F 03	Si	PA
	EI80	A7 03	Mi	EIE0	67 03	Do	
	EI82	9F 03	Fa	EIE2	7F 03	La	
	EI84	A7 03	Mi	EIE4	57 03	Ré	
	EI86	B7 I2	Ré	EIE6	67 03	Do	
	EI88	8F 03	Sol	EIE8	6F 09	Si	
	EI8A	7F 03	La	EIEA	67 03	Do	
	EI8C	8F 03	Sol	EIEC	67 I2	Do	
PA	EI8E	9F 03	Fa	EIEE	6F I2	Si	
	EI90	A7 03	Mi	EIF0	7F I2	La	
	EI92	B7 03	Ré	EIF2	8F I2	Sol	
	EI94	C7 03	Do	EIF4	9F I2	Fa	
	EI96	CF 03	Si	EIF6	8F I2	Sol	
	EI98	C7 03	Do				
	EI9A	B7 03	Ré	EIF8	FD 05	SAV	PB
	EI9C	A7 03	Mi	EIFA	C7 I2	Do	
	EI9E	C7 03	Do	EIFC	FC 4B	SAR	
	EIA0	9F 03	Fa				
	EIA2	A7 03	Mi	EIFE	FC 4D	SAR	
	EIA4	B7 09	Ré	E200	FC 4F	SAR	
	EIA6	C7 03	Do	E202	C7 24	Do	PC
	EIA8	C7 I2	Do	E204	FF 00		
	EIAA	7F 03	La				
	EIAC	87 03	Sol#				
	EIAE	7F 03	La				
	EIB0	6F 03	Si				
	EIB2	67 03	Do				
	EIB4	7F 03	La				
	EIB6	6F 03	Si				
	EIB8	7F 03	La				
	EIBA	6F 03	Si				
	EIBC	67 03	Do				
	EIBE	57 03	Ré				

L'Education sentimentale de Maxime le Forestier.

TABLE.6. Codage de la partition

4) Utilisation du programme :

Notre programme utilise les dix premières adresses RAM du KIT pour la gestion des différents paramètres. Il ne faut en aucun cas y écrire le code de la partition.

Programme :

Adresses en RAM des variables utiles

PHIPAR	0000	Index partition (2 Octet)
PBIPAR	0001	
PHINOT	0002	Index tableau note
NOTE	0003	Code de la note
VOLUME	0004	Volume de la note
TEMPS	0005	Temps de la note
BASE	0006	Temps de base de la note
XX	0007	Première variable de délai
YY	0008	Deuxième variable de délai
PRECOM	0009	Prise en compte sauts

Implantation RAM/KIT

ORG	E00A	Début du programme
TABNOT	E100	Début tableau notes
PIACRB	E483	Registre controle PIAB
PIADRB	E482	Registre E/S PIAB

Adresses du programme	Instructions en language machine	Mnémoniques et données codées hexadécimal	Commentaire
		Initialisation PIA	
INIPIA E00A	7F E483	CLR PIACRB	Accès a DDRB
E00D	86 7F	LDAA # $7F	Sept sorties utiles
E00F	B7 E482	STAA PIADRB	
E012	86 04	LDAA # $04	PIA prêt en E/S
E014	B7 E483	STAA PIACRB	
		Initialisation des variables	
INIVAR E017	7F 0009	CLR PRECOM	Prise en compte= 0
E01A	86 E1	LDAA # $E1	Index tableau notes
E01C	97 02	STAA PHINOT	(poids fort)

Déoodage d'une note

```
DECODE  E01E    DE 00    LDX  PHIPAR   Restauration index part.
        E020    A6 00    LDAA 0,X      Lect.Premier Octet
        E022    I6       TAB           Sauvegarde dans B
        E023    84 F8    ANDA # $F8    Décodage note : Masque
        E025    44       LSRA          2 décalages à droite :
        E026    44       LSRA            2(CODE) ---> A
        E027    97 03    STAA NOTE     NOTE=2(CODE)
        E029    44       LSRA          (CODE) ---> A
        E02A    9B 03    ADDA NOTE     3(CODE) ---> A
        E02C    97 03    STAA NOTE     Rangement NOTE=3(CODE)
        E02E    4F       CLRA          Init. Reg. A
        E02F    C4 07    AND B # $07   Décodage volume: Masque
DECALE  E03I    27 05    BEQ  RGTVOL   Test sur B = (CODE)
        E033    0D       SEC           CARRY = I
        E034    49       ROLA          Décalage à gauche
        E035    5A       DECB          Décrémentation B=(CODE)
        E036    20 F9    BRA  DECALE   Nouveau test
RGTVOL  E038    97 04    STAA VOLUME   Rangement VOLUME = A
        E03A    08       INX           Incrément. Index Partition
        E03B    E6 00    LDAB 0,X      Lect, Deuxième Octet
        E03D    D7 05    STAB TEMPS    Rangement TEMPS NOTE
        E03F    08       INX           Increment. Index Part.
        E040    DF 00    STX  PHIPAR   Sauvegarde Index Part.
```

Test identification note

```
TNOTE   E042    96 03    LDAA NOTE     Chargement NOTE
        E044    8I 5D    CMPA # $5D    Saut si TRAITEMENT
        E046    27 26    BEQ  TRAPAR           PARTITION
```

Détermination des paranètres de note

```
PARAM   E048    DE 02    LDX  PHINOT   Index tableau note
        E04A    E6 00    LDAB 0,X      Lect. Temps de Base
        E04C    D7 06    STAB BASE     Rangement TEMPS DE BASE
        E04E    A6 0I    LDAA I,X      Lect.Premier var.DELAI
        E050    97 07    STAA XX       Rangement
        E052    A6 02    LDAA 2,X      Lect.Deuxième var.DELAI
        E054    97 08    STAA YY       Rangement
        E056    96 05    LDAA TEMPS    Saut si NOTE terminée
TEMPO   E058    27 C4    BEQ  DECODE
```

Exécution d'une période

```
SETPIA  E05A    96 04    LDAA VOLUME   Niveau haut voulu
        E05C    BD E0BD  JSR  DELAI    Attente DELAI
RESPIA  E05F    86 00    LDAA # $00    Niveau bas
        E06I    BD E0BD  JSR  DELAI    Attente DELAI
```

Test temps de base écoulé

```
TEMMIN  E064   5A        DECB          Saut si TEMPS DE BASE
        E065   26 F3      BNE  SETPIA      non terminé
```

Test temps de la note écoulée

```
STEMPS  E067   D6 06      LDAB BASE     Rechargement BASE
        E069   7A 0005    DEC  TEMPS    Saut si TEMPS écoulé
        E06C   20 EA      BRA  TEMPO
```

Traitement partition

```
TRAPAR  E06E   96 04      LDAA VOLUME   Lect.Type traitement
        E070   8I 7F      CMPA # $7F    Saut si non STOP
        E072   26 0I      BNE  VALSAU

STOP    E074   3F        SWI           Fin de PARTITION
VALSAU  E075   8I 3F      CMPA # $3F    Saut si non PRISE EN
        E077   26 05      BNE  TESTIG        COMPTE
        E079   7C 0009    INC  PRECOM   Si oui incrémentation
        E07C   20 A0      BRA  DECODE   Saut décodage suivant
TESTIG  E07E   D6 09      LDAB PRECOM   Saut si PRECOM = 0
        E080   27 05      BEQ  SAUTAV

        E082   7A 0009    DEC  PRECOM   Si oui décrémentation
        E085   20 97      BRA  DECODE   Saut décodage suivant
SAUTAV  E087   8I IF      CMPA # $IF    Saut si non SAUTAVANT
        E089   26 0D      BNE  SAUTAR

        E08B   BD E0A2    JSR  ADDIT    INDEX + TEMPS
        E08E   BD E0A2    JSR  ADDIT    Nouvel INDEX+TEMPS
OPCOM   E09I   C6 02      LDAB # $02    Opération commune aux2
        E093   BD E0B0    JSR  SOUST    Nouvel INDEX - 2
        E096   20 86      BRA  DECODE   Saut décodage suivant
SAUTAR  E098   D6 05      LDAB TEMPS    Saut arrière
        E09A   BD E0B0    JSR  SOUST    Nouvel INDEX-TEMPS
        E09D   BD E0B0    JSR  SOUST    Nouvel INDEX-TEMPS
        E0A0   20 EF      BRA  OPCOM    Saut opération commune
```

Sous-programme d'addition Index Partition

```
ADDIT   E0A2   96 0I      LDAA PBIPAR   Charg.INDEX poids faible
        E0A4   0C         CLC           CARRY = 0
        E0A5   9B 05      ADDA TEMPS    PBIPAR=PBIPAR + TEMPS
        E0A7   97 0I      STAA PBIPAR   Rang. INDEX poids faible
        E0A9   96 00      LDAA PHIPAR   Charg.INDEX poids fort
        E0AB   89 00      ADCA # $00    Addition carry s'il y a
        E0AD   97 00      STAA PHIPAR   Rang. INDEX poids fort
        E0AF   39         RTS           Retour PROG. PRINCIPAL
```

Sous-programme de soustraction Index Partition

```
SOUST   E0B0    96 0I     LDAA PBIPAR   Charg. INDEX poids faible
        E0B2    0C        CLC           CARRY = 0
        E0B3    I0        SBA           PBIPAR = PBIPAR - TEMPS
        E0B4    97 0I     STAA PBIPAR   Rang. INDEX poids faible
        E0B6    96 00     LDAA PHIPAR   Charg. INDEX poids fort
        E0B8    82 00     SBCA # $00    Soust. Carry s'il y a
        E0BA    97 00     STAA PHIPAR   Rang. INDEX poids fort
        E0BC    39        RTS           Retour PROG. PRINCIPAL
```

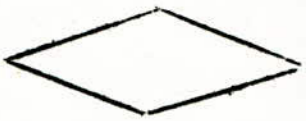
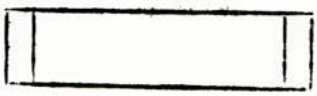
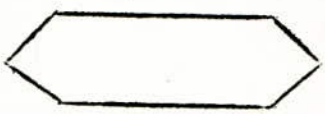
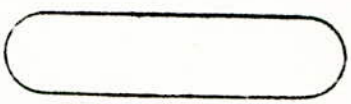
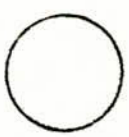
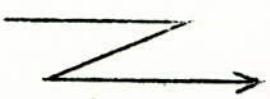
Sous-programme de délai

```
DELAI   E0ED    37        PSHB          Sauvegarde ACCB
        E0BE    B7 E482   STAA PIADRB   Envoi niveau sur PIA
        E0C1    96 07     LDAA XX       Prenière var. DELAI   A
BXX     E0C3    D6 08     LDAB YY       Deuxième var. DELAI   B
BYY     E0C5    5A        DECB          B = B - I
        E0C6    26 FD     BNE  BYY      Saut
        E0C8    4A        DECA          A = A - I
        E0C9    26 FB     BNE  BXX      Saut
        E0CB    33        PULB          Restauration ACCB
        E0CC    39        RTS           Retour PROG. PRINCIPAL
```

5) Mode d'emploi du KIT : Implanter la partition entièrement codée en zone RAM selon la table 7

0000	Variables utiles
0009	
E00A	Programme Processeur musical
E0CC	
EI00	Tableau des notes
EI5A	
EI60	Partition codée
E204	

Lancer le programme en E00A avec IPAR = EI60

TRAITEMENT : Le symbole traitement indique l'exécution d'opérations portant sur des informations. (L'opération effectuée est inscrite dans le symbole).	
EMBRANCHEMENT : Ce symbole définit un choix d'une séquence logique parmis plusieurs possibles. (instruction de rupture de séquence conditionnelle).	
SOUS-PROGRAMME : Ce symbole indique l'exécution d'un sous-programme.	
ENTREE-SORTIE : Ce symbole définit une instruction d'échange avec un organe périphérique.	
DEBUT, FIN : Ce symbole indique le début ou la fin d'un programme.	
RENVOI : Symbole utilisé pour indiquer la continuité d'une séquence lors d'un changement de page.	
INTERRUPTION : Symbole utilisé pour indiquer l'arrivée d'un signal d'interruption.	

Liste des pricipaux symboles graphiques utilisés pour la représentation d'organigrammes.

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				M̄ → M	•	•	↕	↕	R	S
	COMA													43	2	1	Ā → A	•	•	↕	↕	R	S
	COMB													53	2	1	B̄ → B	•	•	↕	↕	R	S
Complement, 2's (Negate)	NEG							60	7	2	70	6	3				00 − M → M	•	•	↕	↕	①	②
	NEGA													40	2	1	00 − A → A	•	•	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	•	•	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts Binary Add of BCD Characters into BCD Format	•	•	↕	↕	↕	③
Decrement	DEC							6A	7	2	7A	6	3				M − 1 → M	•	•	↕	↕	4	•
	DECA													4A	2	1	A − 1 → A	•	•	↕	↕	4	•
	DECB													5A	2	1	B − 1 → B	•	•	↕	↕	4	•
Exclusive OR	EORA	88	2	2	98	3	2	A8	5	2	B8	4	3				A⊕M → A	•	•	↕	↕	R	•
	EORB	C8	2	2	D8	3	2	E8	5	2	F8	4	3				B⊕M → B	•	•	↕	↕	R	•
Increment	INC							6C	7	2	7C	6	3				M + 1 → M	•	•	↕	↕	⑤	•
	INCA													4C	2	1	A + 1 → A	•	•	↕	↕	⑤	•
	INCB													5C	2	1	B + 1 → B	•	•	↕	↕	⑤	•
Load Acmltr	LDAA	86	2	2	96	3	2	A6	5	2	B6	4	3				M → A	•	•	↕	↕	R	•
	LDAB	C6	2	2	D6	3	2	E6	5	2	F6	4	3				M → B	•	•	↕	↕	R	•
Or, Inclusive	ORAA	8A	2	2	9A	3	2	AA	5	2	BA	4	3				A + M → A	•	•	↕	↕	R	•
	ORAB	CA	2	2	DA	3	2	EA	5	2	FA	4	3				B + M → B	•	•	↕	↕	R	•
Push Data	PSHA													36	4	1	A → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
	PSHB													37	4	1	B → M_{SP}, SP − 1 → SP	•	•	•	•	•	•
Pull Data	PULA													32	4	1	SP + 1 → SP, M_{SP} → A	•	•	•	•	•	•
	PULB													33	4	1	SP + 1 → SP, M_{SP} → B	•	•	•	•	•	•
Rotate Left	ROL							69	7	2	79	6	3				M	•	•	↕	↕	↕	↕
	ROLA													49	2	1	A	•	•	↕	↕	↕	↕
	ROLB													59	2	1	B — C b7 — b0	•	•	↕	↕	↕	↕
Rotate Right	ROR							66	7	2	76	6	3				M	•	•	↕	↕	↕	↕
	RORA													46	2	1	A	•	•	↕	↕	↕	↕
	RORB													56	2	1	B — C b7 — b0	•	•	↕	↕	↕	↕
Shift Left, Arithmetic	ASL							68	7	2	78	6	3				M	•	•	↕	↕	↕	↕
	ASLA													48	2	1	A	•	•	↕	↕	↕	↕
	ASLB													58	2	1	B — C b7 b0 — 0	•	•	↕	↕	↕	↕
Shift Right Arithmetic	ASR							67	7	2	77	6	3				M	•	•	↕	↕	↕	↕
	ASRA													47	2	1	A	•	•	↕	↕	↕	↕
	ASRB													57	2	1	B — b7 b0 — C	•	•	↕	↕	↕	↕
Shift Right Logic	LSR							64	7	2	74	6	3				M	•	•	R	↕	↕	↕
	LSRA													44	2	1	A	•	•	R	↕	↕	↕
	LSRB													54	2	1	B 0 — b7 b0 — C	•	•	R	↕	↕	↕
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:

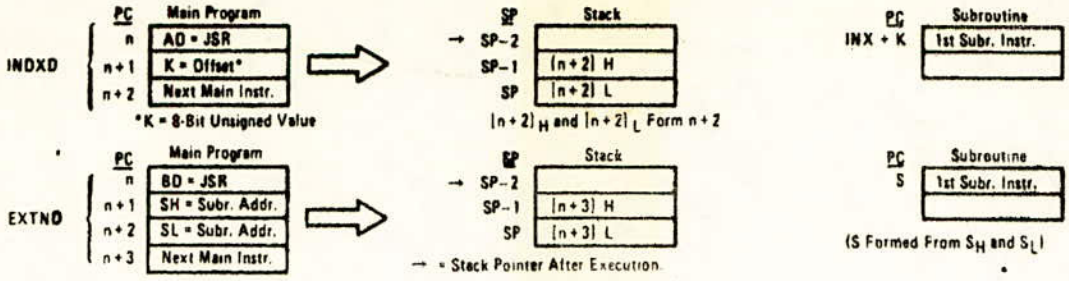
- OP Operation Code (Hexadecimal).
- ~ Number of MPU Cycles.
- # Number of Program Bytes.
- + Arithmetic Plus.
- − Arithmetic Minus.
- · Boolean AND.
- + Boolean Inclusive OR.
- ⊕ Boolean Exclusive OR.
- M̄ Complement of M.
- → Transfer Into.
- 0 Bit Zero.
- 00 Byte Zero.
- M_{SP} Contents of memory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

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

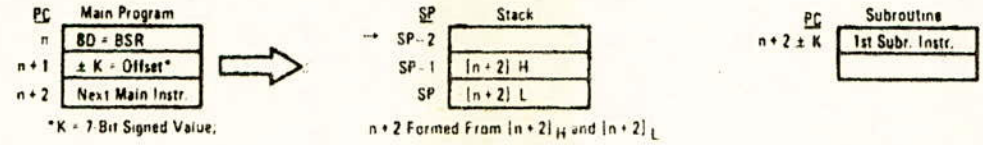
CONDITION CODE SYMBOLS:

- 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

JSR, JUMP TO SUBROUTINE:

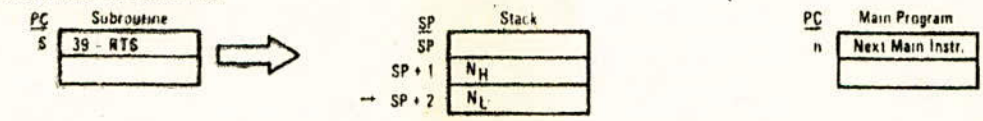


BSR, BRANCH TO SUBROUTINE:



JMP, JUMP:

RTS, RETURN FROM SUBROUTINE:



RTI, RETURN FROM INTERRUPT:

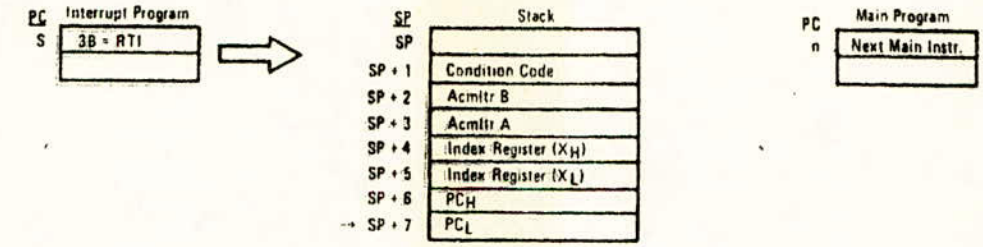
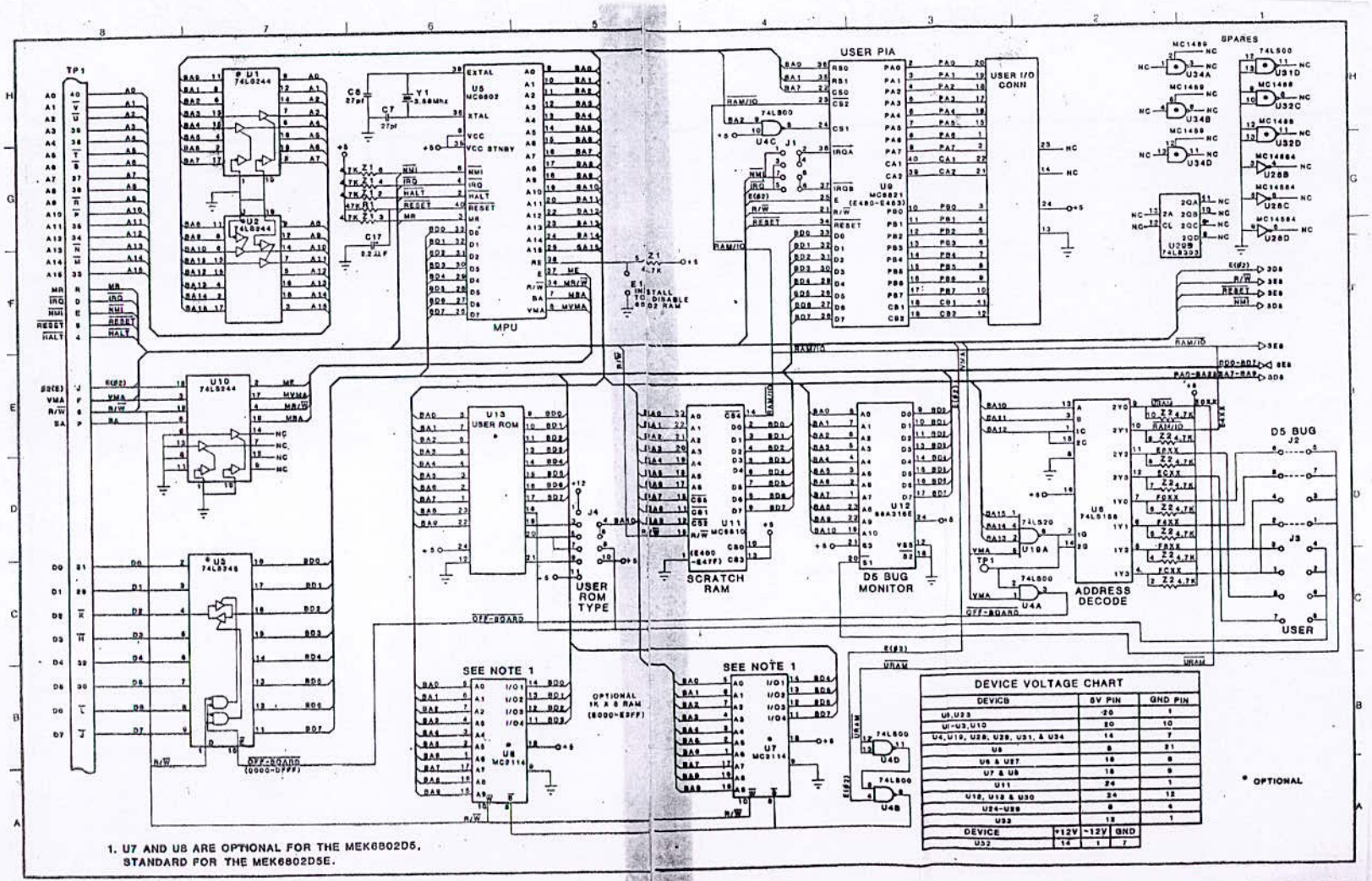


TABLE 6 – CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

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

CONDITION CODE REGISTER NOTES: (Bit set if test is true and cleared otherwise)

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



1. U7 AND U8 ARE OPTIONAL FOR THE MEK6802D5. STANDARD FOR THE MEK6802D5E.

FIGURE 5.1 SCHEMATIC DIAGRAM (SHEET 2 OF 3) 5-5/5-6

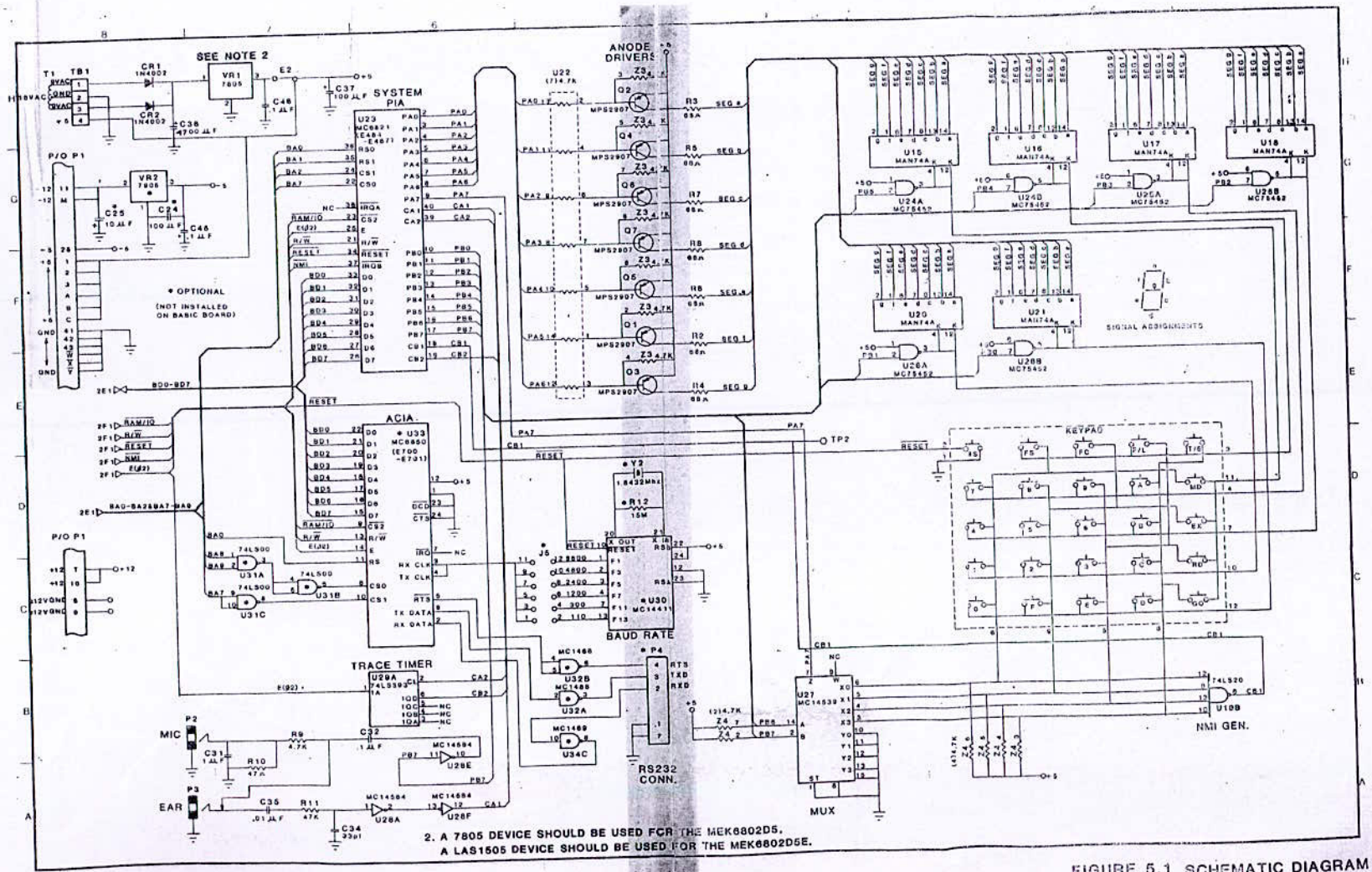


FIGURE 5.1 SCHEMATIC DIAGRAM
(SHEET 3 OF 3) 5-7/5-8

CONCLUSION

. Cette étude aidera à comprendre tout système à base de microprocesseurs.

La facilité de mise en oeuvre du KIT, son faible prix de revient et sa conception justifient son utilisation intense dans les écoles et les universités.

Néanmoins nous pouvons constater qu'une application plus complexe avec le KIT fait appel à l'utilisation d'un système de développement.

VOEUX

. Je souhaite que les générations futures trouveront interêt dans l'étude des microprocesseurs, que leurs utilisations n'ira pas à l'encontre de l'humanité et qu'ils serviront pour la protection des pouples opprimés et à l'installation de la PAIX dans le monde.

BIBLIOGRAPHIE

1 - R. AROUETE - H. LILEN
Théorie et pratique des microprocesseurs
Editions RADIO.

2 - CL. PARIOT
Introduction aux microprocesseurs et aux microordinateurs
Edition DUNOD informatique.

3 - D. GIROD et R. DUBOIS
Au cœur des microprocesseurs
Edition EYROLLES

4 - RODNAY ZAKS - PIERRE LE BEUX
Programmation du 6800

5 - " MICROPROCESSEUR " - Université de LAURAINE

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

7 - Projet de fin d'études :

Etude d'un microordinateur basé autour du 6802 :

Le KIT D5 de MOTOROLA