UNIVERSITE DES SCIENCES ET DE LA TÉCHNOLOGIE D'ALGER

DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTENIQUE

FILIERE D'INGENIEUR EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

Sujet : Etude et Réalisation d'un interface K.7 avec le M.D.A 8002-A

PROPOSE PAR : M. KAOUA
M. ZIZI

REALISE PAR : RIBA Kamel
HADDAB Saïd

JUIN 1983

UNIVERSITE DES SCIENCES ET DE LA TÉCHNOLOGIE D'ALGER

DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTENIQUE

FILIERE D'INGENIEUR EN ELECTRONIQUE

PROJET DE FIN D'ETUDES

Sujet : Etude et Réalisation d'un interface K.7 avec le M.D.A 8002-A

PROPOSE PAR : M. KAOUA
M. ZIZI

REALISE PAR : RIBA Kamel
HADDAB Saïd

JUIN 1983

D E D I C A C E S

A mes parents

A mes frères et soeurs

A mon frère Farid

A mon ami Mohamed

KAMEL

A mes parents

A mes frères et soeurs

A tous mes amis

HADDAB Saïd

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

Nous formulons l'expression de notre profonde reconnaissance à Melles KAOUA Malika et ZIZI Malika, assistantes à l'ENPA, qui ont bien voulu dirigé notre travail. Nous les remercions pour leur aide précieuse et leurs conseils éclairés.

Nous ne manquerons pas d'exprimer aussi notre gratitude à tous les professeurs et assistants du laboratoire d'électronique appliquée de l'ENPA.

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

Dans ces dernières années, la microinformatique occupe une place de plus en plus importante dans la plupart des domaines tels que : l'industrie, la médecine, l'enseignement, etc... D'où la nécessité d'avoir des systèmes microinformatiques de plus en plus performants pour répondre aux besoins des utilisateurs.

Par exemple, vu que les micro-ordinateurs disposent d'une capacité mémoire (ROM et RAM) limitée, l'adjonction d'un interface cassette, s'est faite sentir ; pour soulager la tâche de l'utilisateur, en mettant à sa disposition une mémoire auxilliaire de capacité plus grande, qu'est la bande magnétique.

Ainsi dans le cadre de notre projet de fin d'études, il nous a été confié d'étudier et de réaliser un interface K7 en liaison avec le système de développement TEKTRONIX 8002A du laboratoire d'électronique appliquée de l'E.N.P.

Nous avons commencé par travailler avec un KIT d'initiation le MEK K5 de MOTOROLA, disponible au labo, pour bien comprendre, si ce n'est que superficiellement, le fonctionnement d'un micro-ordinateur d'une part, et les méthodes de programmation d'autre part.

Dans un premier chapitre, nous donnerons des généralités sur le KIT D5, et le principe général d'un interface K7.

Le second chapitre, comprendra une étude comparative sommaire de deux types d'interface K7 : à savoir ceux du KIT D2 et D5. Cette étude, nous permettra d'opter pour celui du KIT D5, vu qu'il est plus économique et plus souple.

La troisième partie, est consacrée à la réalisation d'un interface K7 à l'aide du MDA 8002A.

Et enfin, on termine par une application, qui consiste en l'élaboration des programmes de gestion de cet interface, à savoir le programme d'enregistrement (ENREG), et le programme de recherche et lecture (LECT).

TABLE DES MATIERES

INTROCUTION PAGES

CHAPITRE I - GENERALITES :

A - Organisation générale du KIT D5 1
B - Principe d'un interface cassette 6

CHAPITRE II - ETUDE DES INTERFACES K7 DES KITS D2 ET D5

A - Etude de l'interface du Kit D2

1 - Format d'enregistrement 10
2 - Principe de fonctionnement 11
3 - Logiciel associé 20

B - Etude de l'interface du Kit D5

1 - Format d'enregistrement 24
2 - Principe de Fonctionnement 25
3 - Logiciel associé 28

C - Comparaison des interfaces K7 des Kit D2 et D5 39

CHAPITRE III - REALISATION D'UN INTERFACE K7 AVEC LE SYSTEME MDA 8002A :

1 - Présentation du systéme de développement (TETRONIX 8002A) 41
2 - Présentation de la maquette 42
3 - Cablage de la maquette 48
4 - Procédure de mise au point de la maquette 48

CHAPITRE IV - LOGICIEL DE GESTION DE L'INTERFACE :

4.1. - Programme d'enregistrement "ENREG" 52
4.2. - Programme de recherche et de lecture d'une file de données "LECT" 60

CONCLUSION
ANNEXE
BIBLIOGRAPHIE

CHAPITRE I

GENERALITES

A - ORGANISATION GENERALE :

Le KIT MEK 6802 D.5 de MOTOROLA est un système de base de faible coût permettant d'utiliser les composants de la famille 6800.
Il comprend, le MPU MC 6802, une mémoire morte la ROM D.5 BUG (contenant le moniteur), de la mémoire vive RAM et des organes d'entrée / 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 KIT D.5 est constitué par différents blocs fonctionnels (voir fig 1.1) on présentera brièvement ces différents blocs (pourplus de détails, se reférer à la bibliographie).

Le bloc de commande de tout le système est représenté par le MPU MC 6802 qui est un microprocesseur monolithique en téchnologie N-MOS se présentant sous forme d'un boîtier de 40 broches (voir brochage fig 1.2)

Le MPU 6802 travaille avec des mots de 8 bits.Grâce à ses 16 lignes d'adresse il a la possibilité d'être extensible dans un système jusqu'à 2^{16} = 64 K adressses mémoires ou périphériques.

Le MPU 6802 a les mêmes registres et accumulateurs que le 6800 (voir fig 1.3) avec en plus.

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

Il possède un jeu de 72 instructions, identiques à celles du 6800.

Le blocs à mémoires mortes est constitué de mémoires à lecture seule dont la "D.5 BUG" de 2.K octets contenant le programme moniteur est indispensable pour la gestion du KIT.

Une "USER ROM" optionnelle qui peut aller jusqu'à 2.K octets est prévue pour stocker des programmes de l'utilisateur.

Le bloc à mémoires vives comprend, en plus de la RAM interne du 6802, deux autres RAM du type volatile qui sont :

a - la"SCRATCH RAM" de composée de 128 octets.

b - l'"USER RAM" de capacité 1.K octets est constituée par 2 boîtiers de type MC.2114.

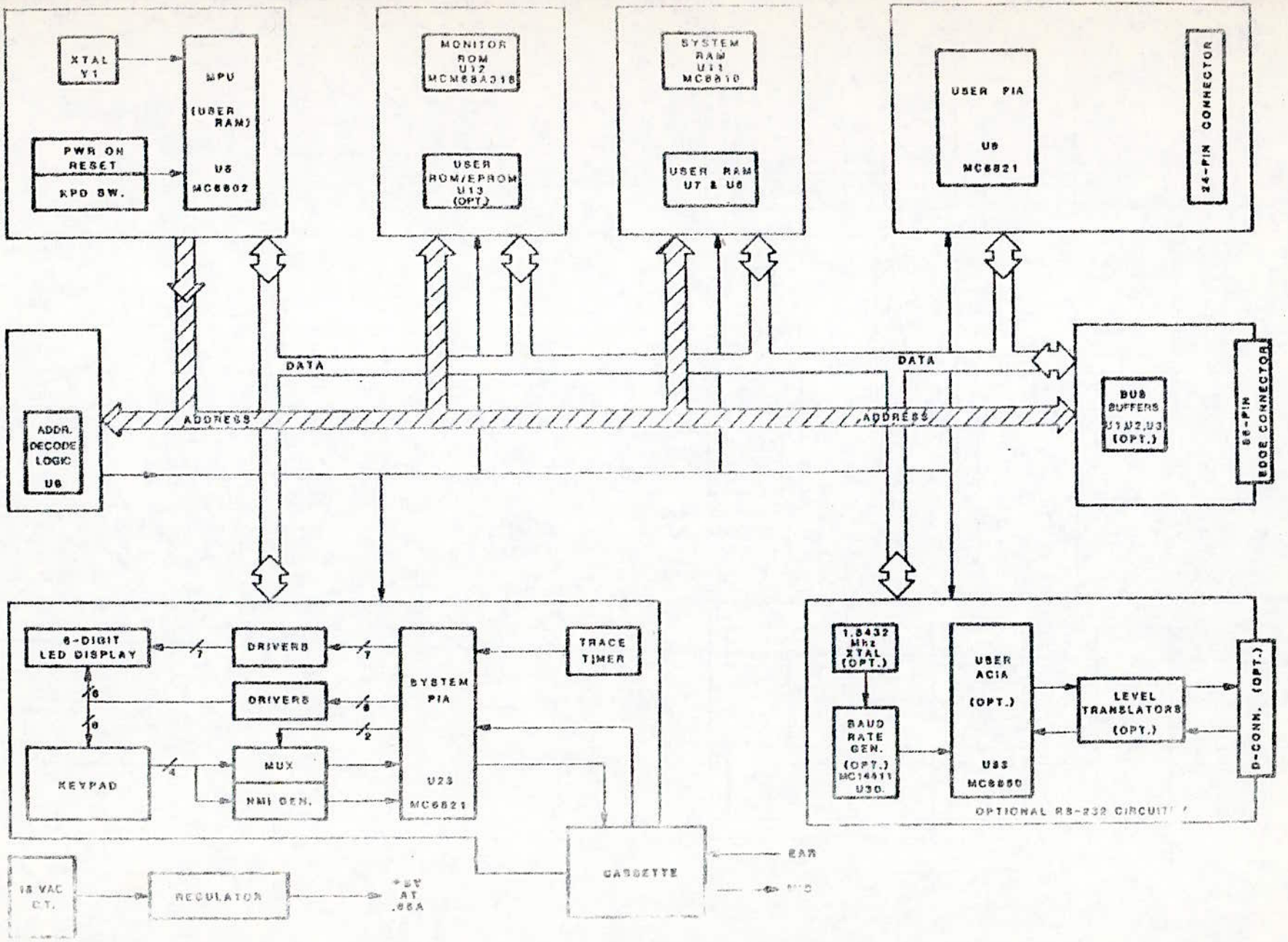


FIGURE 1-1. bloc diagram

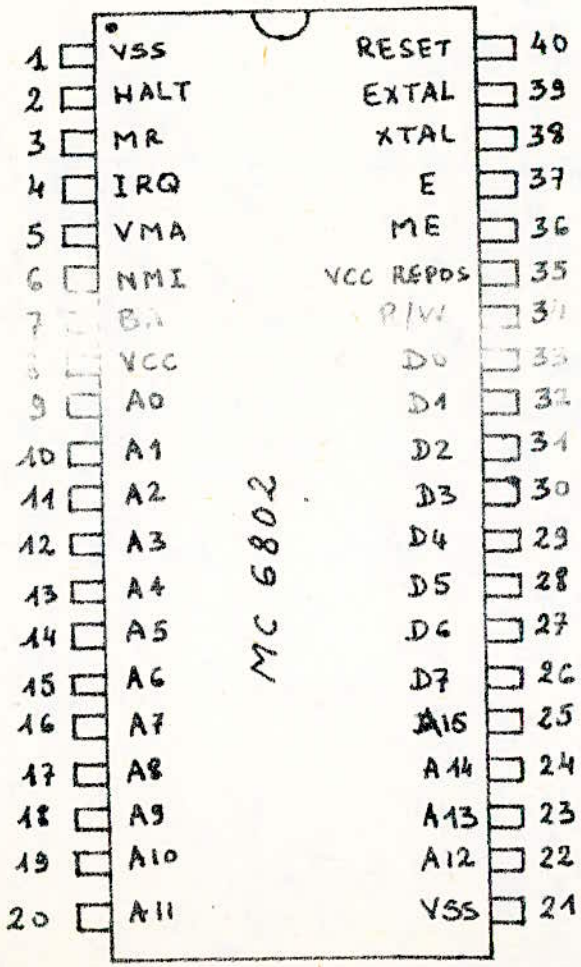


Fig. 1-2. Brochage du MPU 6802.

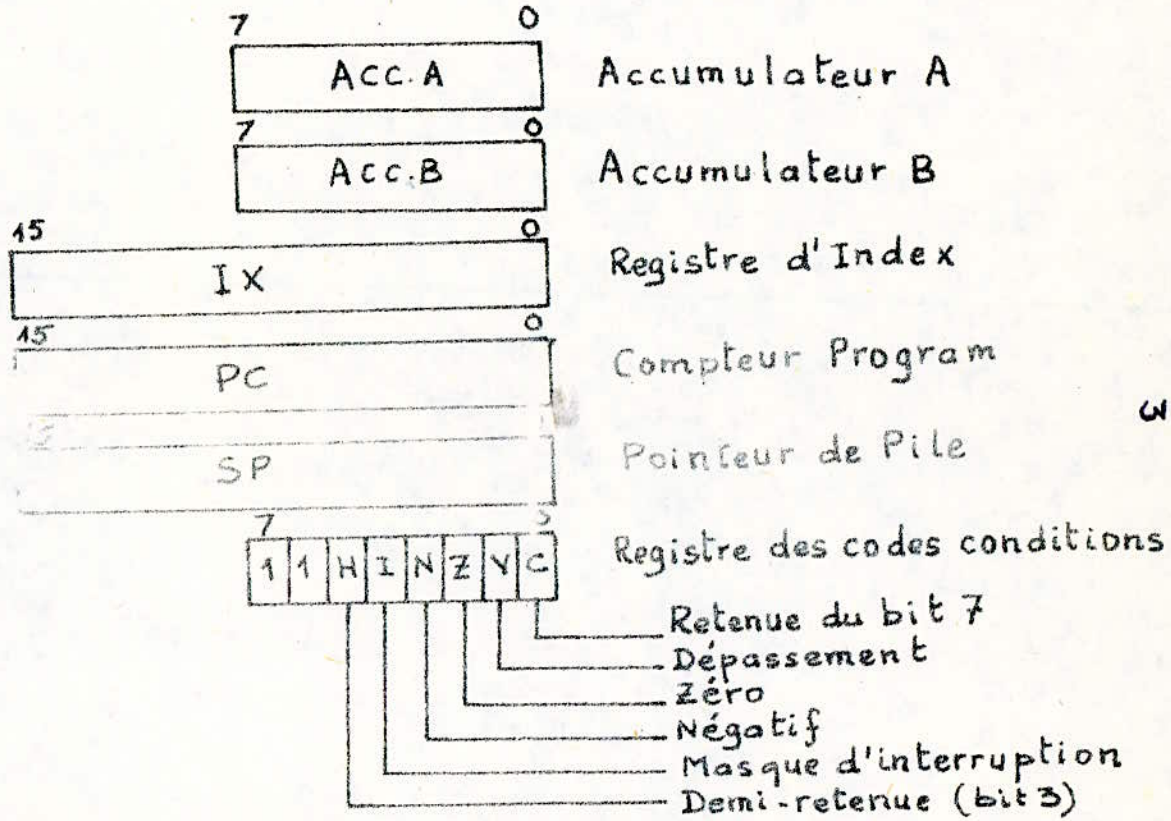
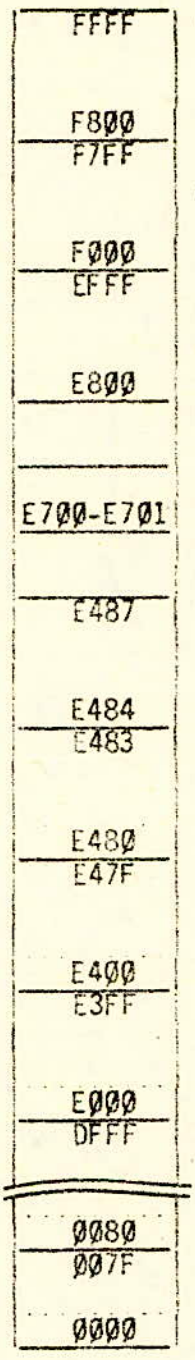


Fig. 1-3 - Registres Programmables du MICROPROCESSEUR.



Operating System Mirror (or optional user ROM)

Operating System (D5BUG)

Optional User ROM

Reserved

System ACIA*

Reserved

System PIA

User PIA

System RAM

User RAM (1K)

External to MEK6802D5

User RAM inside MC6802 (must be disabled if optional Bus buffers are installed)

*ACIA is not supported by D5BUG software.

FIGURE 1-4. MEMORY MAP

Le KIT utilise la méthode d'adressage par décodage.C'est grâce à un circuit logique du type 74 L S 156 que l'on arrive à simplifier au maximum l'adréssage 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.Cette méthode est intéressante, car elle permet d'occuper la plus grande partie possible des 64 K positions mémoires théoriquement disponibles.La figure (1.4)représente la répartition mémoire du KIT.

Le bloc interface parallèle "USER PIA" (MC 6821)est prévu spécialement pour l'utilisateur.Il assure la liaison entre le MPU et les périphériques (convertisseur,imrimante etc.....)

Un bloc interface série en option sur le KIT est composé du "USER ACIA" (MC 6850)et d'un générateur de fréquences (MC 14411).

- Les BUFFERS en option sont utilisés lors de l'extension du KIT P 5
- L'Opérateur peut communiquer avec le microprocesseur à travers un clavier,en entrée,et un système d'afficheurs en sortie.

Le clavier (ou KEYPAD)de 25 touches permet l'introduction de données qui seront prises en compte par le MPU à travers le"SYSTEM PIA" (port B)+PA7. leur affichage s'éffectue par l'envoie sur les lignes du port A du code 7 segments du digit à afficher.

Ce même "SYSTEM PIA" est relié à un interface CASSETTE par les lignes CA 1,pour la lecture,et PB 7 pour l'enregistrement.

Dans le prochain chapitre nous verrons plus en détails le principe de cet interface.

B -- PRINCIPE D'UN INTERFACE CASSETTE:

L'usage de la cassette comme mémoire auxilliaire nécessité l'adjonction d'un interface cassette réalisant la conversion des données numériques en un signal analogique pouvant être enregistré sur la bande magnétique.

Un choix du format d'enregistrement des données doit être défini. Dans ce format le constructeur doit tenir compte:

-- De la précision relative des magnétophones ordinaires, durant l'enregistrement et la restitution d'un signal donné.

-- De leur bande passante qui est réduite à environ 8 KHZ, puisque ces appareils sont destinés à l'enregistrement et la reproduction d'un signal audiofréquence.

La méthode adoptée dans les KIT D2 et D5 répond à ces critères. Dans ces 2 systèmes ou utilise le " KANSAS CITY STANDARD " (KCS).

Cette méthode (KCS)est spécialement désignée pour éliminer les érreurs dues aux fluctuations de vitèsse des magnétophones.Ses principales caractéristiques sont les suivantes:

1 -- Un "1" logique correspond à 8 cycles d'un signal à 2400 Hz.
2 -- Un "0" logique correspond à 4 cycles d'un signal à 1200 Hz.

Donc l'enregistrement des données sur une bande magnétique est fait sous la forme d'une modulation de fréquence.

En combinaison avec un programme de contrôle, l'interface CASSETTE est chargé d'éxécuter toutes les transformations nécessaires.

B.1. FONCTIONNEMENT EN MODE D'ENREGISTREMENT:

Nous voyons sur la figure 1.5 le schéma synoptique d'un interface CASSETTE en mode d'enregistrement.

L'octet stocké en mémoire principale est chargé dans un accumulateur du MPU.

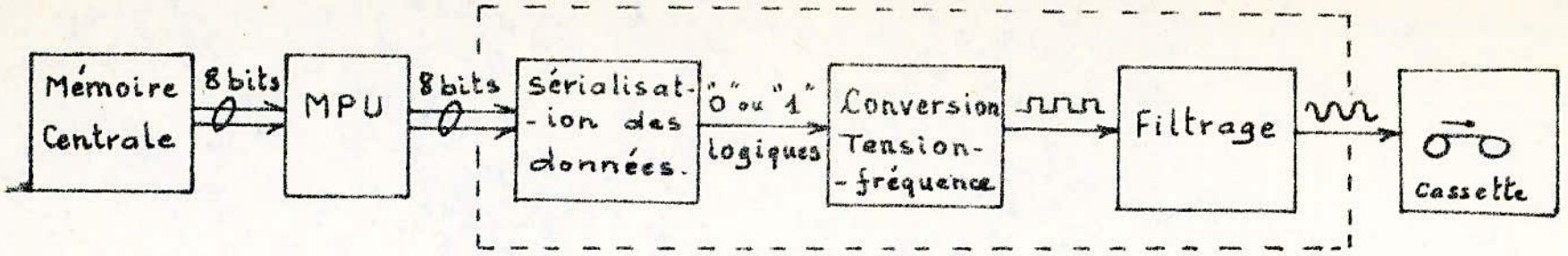


Fig 1-5. En mode d'enregistrement.

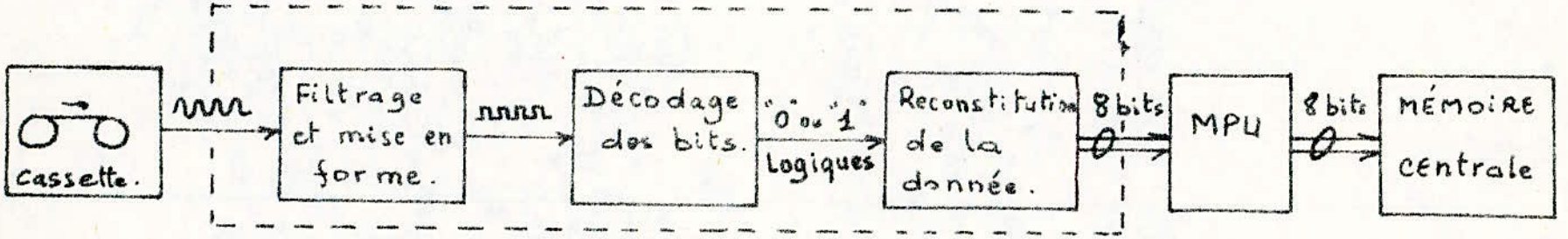


Fig 1-6. En mode Lecture.

SCHÉMA SYNOPTIQUE D'UN INTERFACE CASSETTE.

La donnée,constituée de 8 bits,ne peut être enregistrée d'un trait, car les cassettes ordinaires ne disposent que de 2 pistes seulement.la sérialisation de l'octet s'impose.Cette opération permet l'envoi des bits,les uns aprés les autres.

A la sortie du bloc de sérialisation chaque bit correspond,à un niveau logique devant être converti en un signal de fréquence déterminée.Par exemple pour le format "KCS",chaque niveau logique "1" est converti :en 8 cycles d'un signal carré de 2400 Hz et chaque niveau logique "0" en 4 cycles d'un signal carré de 1200 Hz.

Afin d'éviter l'apparition de distorsions au niveau de l'enregistrement le signal carré est filtré .On obtient en sortie un signal sinusoïdal prêt à être enregistré.

B.2 - FONCTIONNEMENT EN MODE DE LECTURE:

Pendant une opération de lecture l'interface réalise la fonction inverse de l'enregistrement.A savoir,il décode le signal modulé,en provennance de la cassette,afin de reconstituer les informations binaires.

La figure 1.6.représente le schéma SYNOPTIQUE d'un interface CASSETTE en mode de lecture.

Le signal provenant du magnétophone à cassette (signal sinusoïdal modulé en fréquence)est tout d'abord filtré a fin d'en éliminer éventuellement la composante continue et signaux parasites qui peuvent l'affecter.Aprés le filtrage le signal est mis sous forme de signal carré,dont la fréquence varie, pour le "KCS " entre 2400 Hz et 1200 Hz,en fonction de l'enregistrement sur la bande .

Ce signal modulé en fréquence est ensuite converti:en des "0" et "1" logique.

- pour éviter un fonctionnement défectueux une marge d'érreur doit être tolérée durant ce mode.Ainsi les signaux dont la fréquence est inférieure à une certaine valeur,sont décodés comme étant des "0" et ceux dont la fréquence est supérieure à cette valeur comme étant des "1".En guise d'exemple dans le KIT D2 la valeur prise est la fréquence Fs correspondant au milieu de l'intervalle compris entre les fréquences du "1" et du "0"

$$Fs = Fo + \frac{F1 - Fo}{2}$$

F1 .. Fréquence du "1"

Fo .. Fréquence du "0"

En valeur numerique on obtient:

$$Fs = 1200 + \frac{2400 - 1200}{2} = 1800 \text{ Hz}$$

Fo=1200Hz Fs=1800Hz F1=2400Hz

Les bits ainsi décodés vont être stockés les uns aprés les autres dans un registre jusqu'à la formation d'un octet.La dernière opération consiste alors à stocker la donnée lue en mémoire centrale. Dans certains systèmes l'accent est mis plutôt sur le logiciel que sur la logique cablée pour réaliser la conversion de données numériques en signal analogique et vice-vers a.Néanmoins chaque méthode a ses avantages et ses inconvénients.

Ces deux options seront illustrées dans le chapitre 2 par deux exemples d'interfaçes du KIT D2 et du KIT D5.

C H A P I T R E II

ETUDE DES INTERFACES K7 DES KITS D2 ET D5

A - ETUDE DE L'INTERFACE CASSETTE DU KIT D2:

Cet interface donne à l'utilisateur la possibilité de sauver et de relire ses programmes avec un magnéto à cassette ordinaire.La méthode d'enregistrement utilisé est la méthode KANSAS CITY STANDARD qui code un "1" logique par 8 cycles de 2400 HZ et un "0" logique par 4 cycles de 1200 HZ.

A - (1) FORMAT D'ENREGISTREMENT :

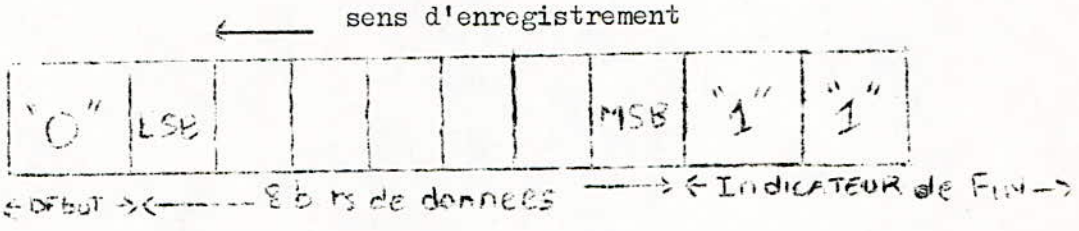
les données à enregistrer sont organisées en blocs,qui à leur tour sont organisés en file de caractères.

a) FORMAT D'UN CARACTERE :

- Un caractère est enregistré sous forme d'un "0" suivi de 8 bits de données et de 2 ou plusieurs "1" logiques qui servent d'indication de fin.

- Entre 2 caractères,on enregistre un nombre indeterminé de "1" qui correspondent à une fréquence de 2400 HZ.

- A l'intérieur d'un caractère: c'est le bit le moins significatif (LSB)qui est transmis en premier,le bit le plus significatif (MSB)est transmis en dernier.



format d'un caractère

b)Format d'un "bloc" de données:

Les données sont organisées en blocs arbitraires de longueurs variables.Chaque bloc est précédé par au moins un enregistrement de 5 secondes de "1" - suivant l'amorce de la bande,des données significatives ne doivent pas être enregistrées pendant les 30 premières secondes.

- Un caracttère "B" début de bloc,est enregistré à la suite des 30 s (1024 "1" logiques).

- Le "B" est suivi d'un octet indiquant la longueur du bloc (longueur au max 256 octets de données).

.../...

- Les 2 octets suivant indiquent l'adresse de début (BEGAD) de la zône mémoire (RAM) contenant initialement les données.

- On peut ensuite enregistrer jusqu'à 256 octets de données et 25 "1" suivent le bloc de données, ainsi que le caractère "G" correspondant à la fin du bloc.

30 s de "1"	"B" Début de bloc	Longueur du bloc (1 oct	Adresse de début (2 octets)	256 octets de données au max	25 s de 1	"G" fin de bloc

format d'un bloc

A - 1) Principe et Fonctionnement de l'Interface Cassette :

Pour bien comprendre cet interface, on donne tout d'abord son synoptique générale, qui sera suivi par une explication des parties enregistrement et lecture.

a) Synoptique Générale :

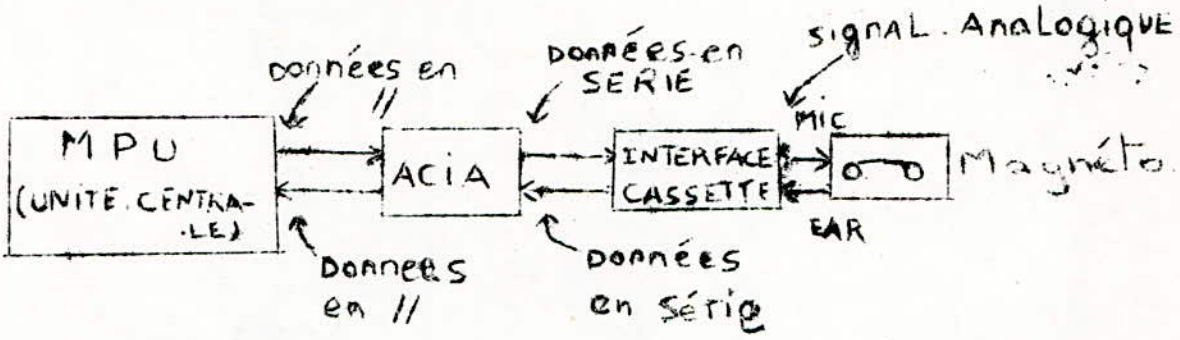


fig 2 - 1 - Schéma Synotique

On sait que le MPU travaillant sur des mots de 8 bits en parallèle ne peut utiliser directement des caractères transmis par le périphérique en mode série.

Il est donc nécessaire de disposer d'un circuit d'interface d'entrée / sortie réalisant la conversion d'un caractère de 8 bits en parallèle en un caractère de 8 bits en série et vice-versa. C'est le cas de l'ACIA qui travaille en mode série asynchrone en lui associant un certain logiciel approprié. Ce circuit permet d'interfacer entre le MPU et l'interface cassette.

Pour l'enregistrement, l'ACIA transmet des données numériques en série vers l'interface cassette chargé de les transformer en signaux analogiques qu'on pourra enregistrer sur bande via l'entrée "MIC" du Magnéto K7.

.../...

Pour la lecture, c'est le fonctionnement inverse : les signaux analogiques issus de la sortie "EAR" sont convertis en données numériques en série, celles-ci sont transmises à l'ACIA qui les désérialise (mise en parallèle) avant de les envoyer au MPU.

Nous rappelons que l'ACIA est considéré par le MPU comme 2 positions mémoires dont chacune correspond à 2 registres.

- Les registres de Transmission et Reception de données.
- Les registres de contrôle et d'état.

L'ACIA dialogue avec la périphérie par l'intermédiaire des broches de transmission et Reception. Les transmission et reception s'opèrent sous contrôle des horloges TxCLK et Rx CLK .

Enfin, notons que l'interface cassette ne fonctionnera que lorqu'on lui associé un certain logiciel. Ainsi les programmes "PUNCH" et "LOAD" en combinaison avec un programme de contrôle, de l'ACIA et du circuit d'interface cassette assurent respectivement. l'enregistrement et la lecture des données sur cassette.

b) Interface d'Enregistrement :

Pour l'enregistrement, on programme l'ACIA en mode de transmission : fig 2-2 / 2 - 3 / 2 - 7.

Dans ce cas, le registre de contrôle contient un mot définissant : la longueur du caractère, le nombre de bits, et le rapport de division. Le MPU lit systèmatiquement le bit T x DRE du registre d'état lui indiquant si le registre de transmission est vide. Une fois que la phase de préparation à la transmission terminée, un caractère est transmis dans le registre de transmission, puis transféré dans un registre à décalage et enfin envoyé en série par la ligne T x Data suivant le format donnée en (1-a).

Ce type d'opération continue jusqu'à ce que tous les caractères soient transmis. Ces derniers qui sont sous forme numérique arrivent à l'interface K7 par la ligne T x Data. Cet interface se charge de les convertir en signaux analogiques de fréquence 2400 Hz ou 1200 Hz suivant qu'il s'agit d'un "1" ou d'un "0" logiques.

Voyons maintenant comment fonctionne cet interface fig 2-7

Le Multiplexeur / Demultiplexeur Mc 14053 (U20) est utilisé pour diriger les signaux vers les p oints désirés pendant les opérations PUNCH et LOAD

.../...

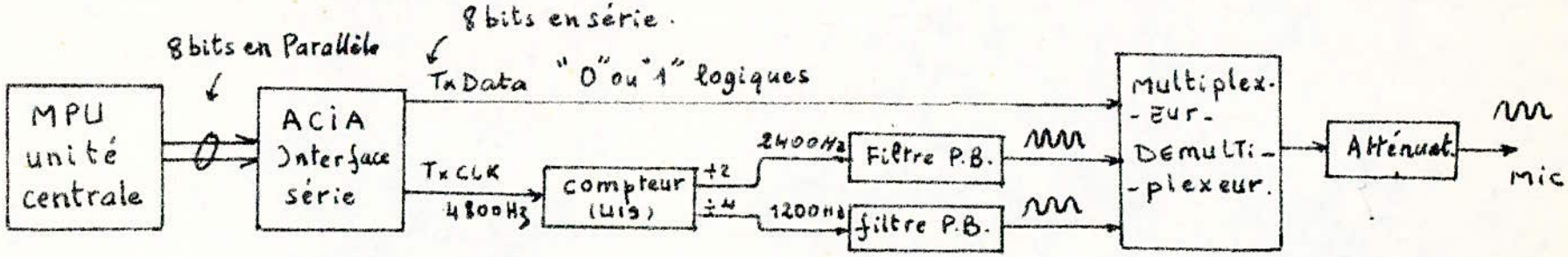


Fig 2-2. Schéma synoptique de la partie Hardware relative à l'enregistrement.

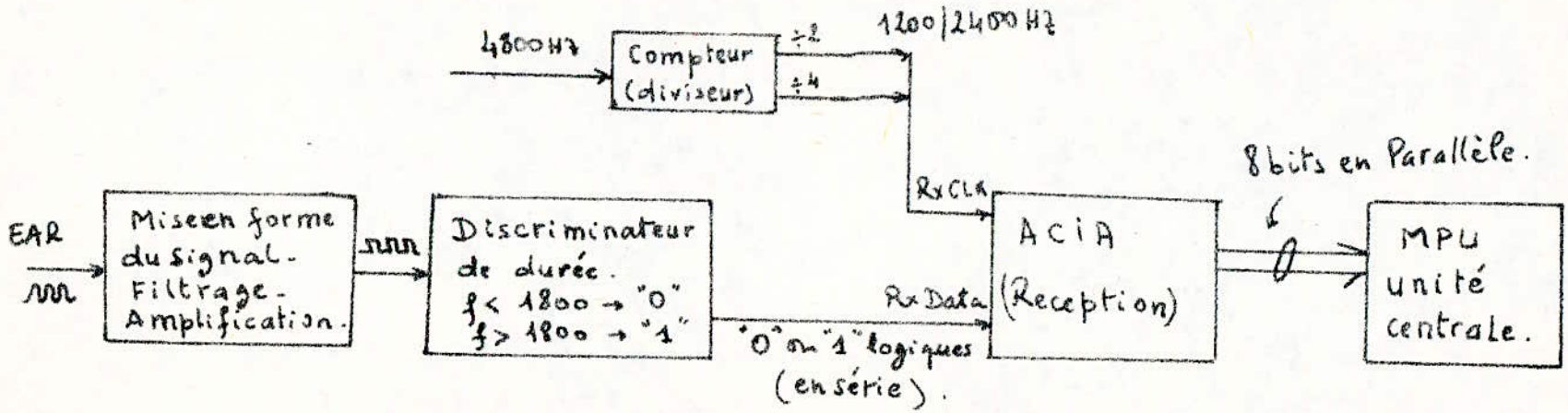


Fig 2-3. Schéma synoptique de la partie Hardware relative à la lecture.

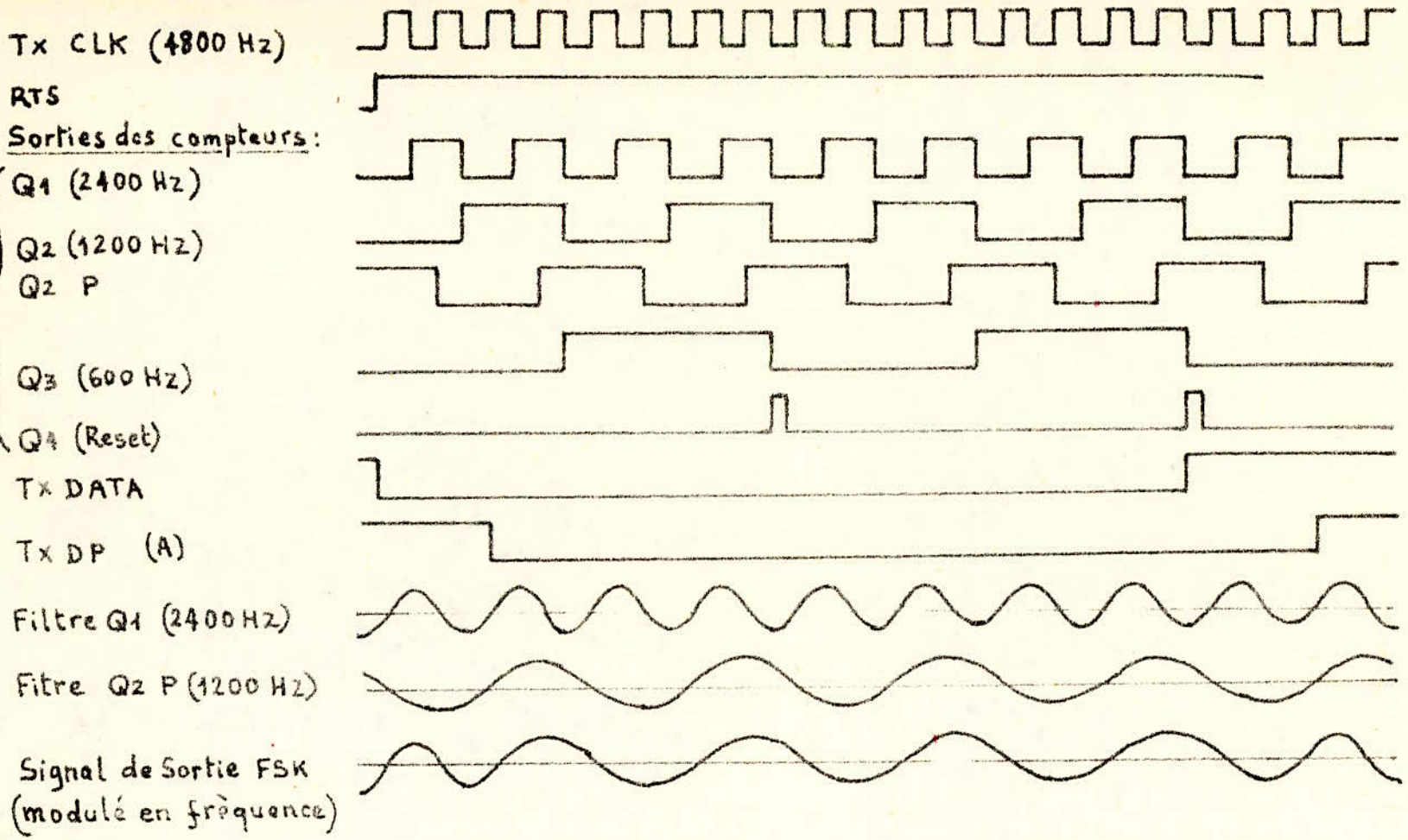


Fig. 2.4. Signaux de Transmission (PUNCH)

Pendant PUNCH : B et C sont au niveau haut, tandis que A est fourni par les données binaires sur T x Data.

Avec cette combinaison des signaux : y est connecté à y1 (parceque B est au niveau haut) ; ainsi le signal d'horloge 4800 Hz (T x CLK) venant de la carte micro-ordinateur, est appliqué à l'entrée d'horloge du compteur MC 14024 (U19).

D'autre part, puisque C est au niveau haut, Z est connecté à Z1, mais ce signal n'est pas utilisé pendant PUNCH.

Pour obtenir les signaux de 2400 Hz et 1200 Hz, on sélectionne soit la sortie $\div$ 2 (Q1), soit le sortie $\div$ 4 (Q2) du compteur qui reçoit une horloge de 4800 Hz. Ainsi les signaux en Xo et X1 sont des sinusoïdes de fréquences respectives 1200 Hz et 2400 Hz obtenus à l'aide des filtres passe-bande (U16 a et U16d). Puis un de ces signaux (en fonction du niveau logique Tx Data en A), sera atténué et appliqué aux sorties du microphone - le signal est ensuite lu par le magnétophone et enregistré sur cassette.

C) Interface de Lecture ;

L'interface réalise le fonctionnement inverse : Il décode le signal modulé en fréquence, en provenance de la cassette afin de reconstituer les informations binaires (voir fig 2-4/2-5/2-6/2-7).

Le signal Rx CLK est tel que l'on doit avoir une transistion positive au milieu de chaque bit et une transition négative à la fin de chaque bit.

Voyons maintenant les differentes phases pour l'obtention de ce signal - fig 2 - 7 :

Le signal provenant de EAR (Ecouteur), est filtré amplifié puis mis sous forme d'un signal carré par U 17 (U17 est un Trigger de schmitt pour amoindrir les problèmes de bruit). on obtient un signal variant entre 0 et 1 dont la fréquence varie entre 2400 Hz et 1200 Hz.

Ce signal est ensuite converti en "0" et "1" logiques grâce à un discriminateur de durée formé par le monostable (U11 = MC 15538) et la bascule D (U18). Ainsi les signaux dont la fréquence est inférieure à 1800 Hz sont décodés comme "0", et les signaux dont la fréquence est supérieure à 1800 Hz sont décodés en "1" pendant "LOAD", le signal 1200 Hz / 2400 Hz est dirigé vers l'entrée horloge à la place du signal 4800 Hz (Tx CLK).

Les sorties du compteur $\div$ 8 (Q3) et $\div$16 (Q4) sont reliées respectivement aux entrées U 14 b et U 14 a. Ces entrées de contrôle sont reliées à la ligne Receive -Data (Rx Data) .

.../...

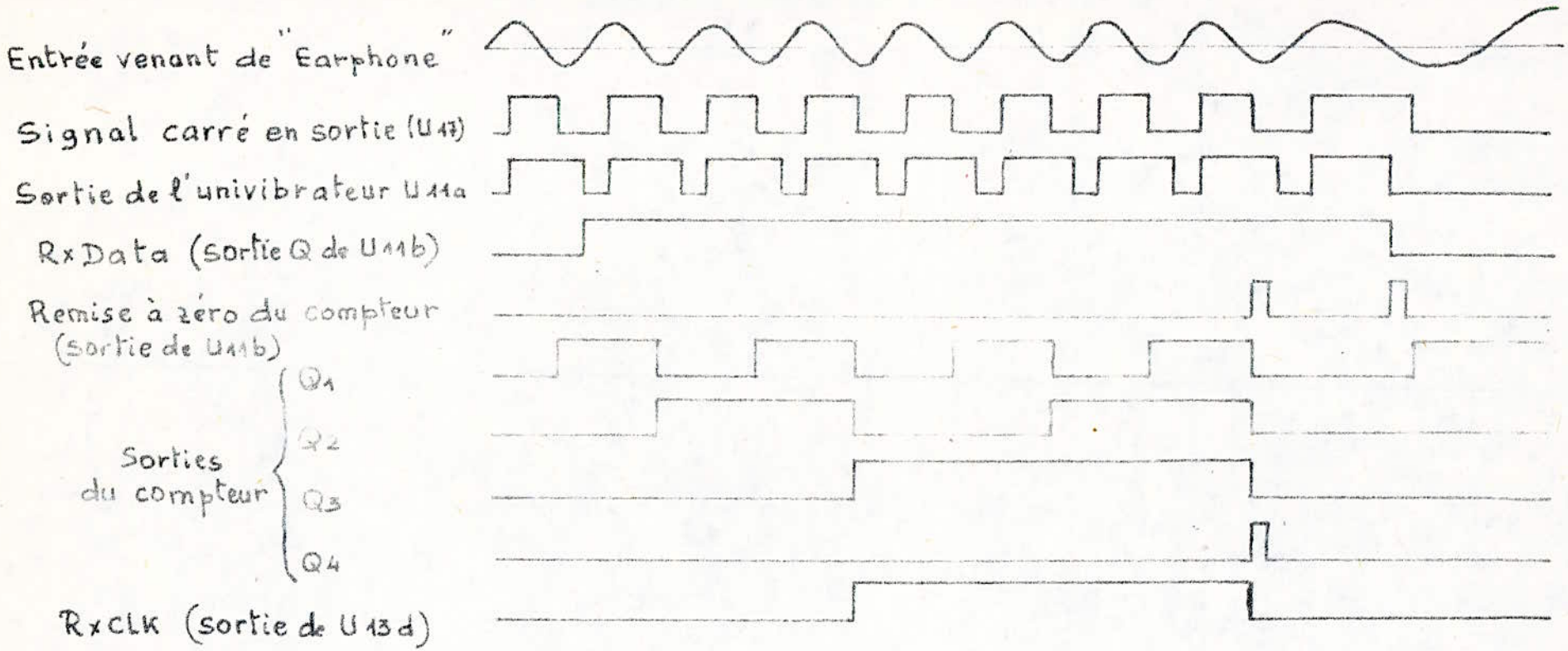


Fig 2-5. Signaux en Reception - Transition de "0" à "1" (LOAD)

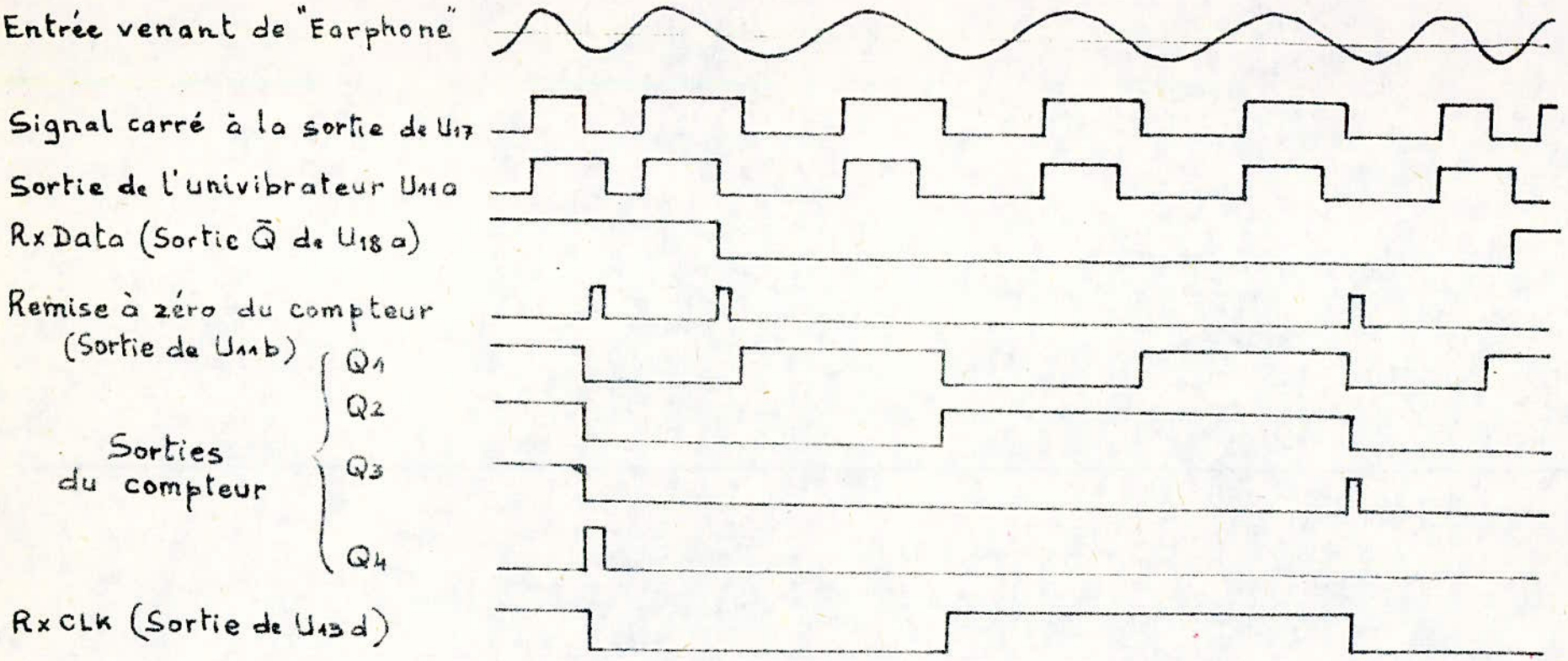


Fig 2-6. Signaux en Reception - Transition de "1" à "0" (LOAD)

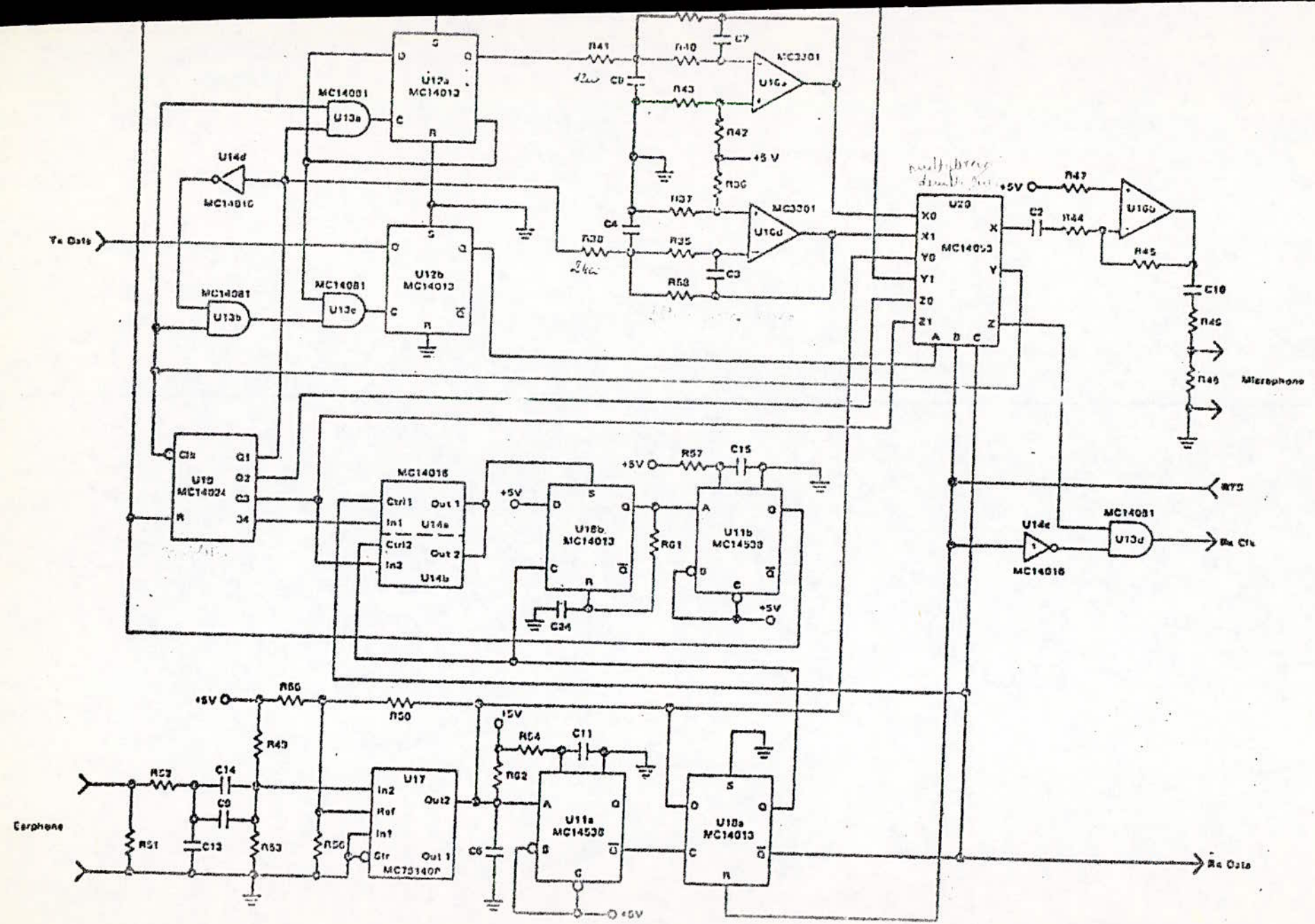


Fig 2-7. Interface pour le magnétophone à cassette (KIT D2).

et à l'entrée de mise à 1 de U 18 b.Ce dernier déclenche le circuit U11 b de remise à zéro du compteur. Par conséquent soit la sortie ÷ 8, soit la sortie ÷ 16 du compteur qui est dirigé (viaX), comme remise à zéro en fonction de l'état "0" ou "1" de la donnée.

Ce compteur est aussi remis à zéro pour chaque transition de "1" à "0" grâce au circuit U11b. Les sorties du compteur ÷ 4 et ÷ 8 sont connectées respectivement à Zo et Z1 du multiplexeur. Ces connections combinées avec les signaux de remise à zéro, fournissent une transition positive en sortie du circuit U20, après 4 cycles à 2400 Hz, ou 2 cycles à 1200 Hz. Ainsi le signal R x CLK a une transition positive au milieu de chaque bit et une transition négative à la fin de chaque bit.

Enfin, les données binaires arrivent à l'ACIA par sa broche d'entrée (Rx Data). Celui-ci est programmé en mode reception Une synchronisation entre l'horloge et les données permet une reception série à la fréquence de l'horloge ou à des fréquences 16 fois ou 64 fois moindre que celle de l'horloge.

La parité paire ou impaire est contrôlée pendant la reception du caractère. Lorsque celui-ci est transmis dans le registre de Réception, l'ACIA génère des signaux d'interruption sur la broche $\overline{IRQ}$ du MPU, qui lit le contenu du registre d'état pour savoir si le registre de reception est plein, et éventuellement pour détecter les erreurs de parité, de format, etc...

La séquence de reception continuera de la même façon jusqu'à ce que tous les caractères soient lus.

Enfin le MPU stockera ces données bilaires dans des cases mémoires prévus à cet effet.

A- (3)- LOGICIEL ASSOCIE A L'INTERFACE K7 (KIT D2) :

a) Enregistrement (PUNCH) :

En appuyant sur la touche P : Punch : on fait appel au programme de transfert de données vers la cassette.

Au départ l'ACIA met le signal $\overline{RTS}$ au niveau bas indiquant qu'on est en mode Punch. On programme l'ACIA pour mettre $\overline{RTS}$ au niveau haut, et pour prendre un mode de transmission de 8 bits, sans parité, avec 2 bits stop.

La transmission est choisie avec une fréquence qui est égale au 1/16e. de la fréquence d'horloge. Tout d'abord 1024 "1" sont transmis à l'aide du sous-programme PNLDR. Puis le programme compare l'adresse de début avec l'adresse de fin. Si la différence est supérieure à 256, le 1er bloc est formé de 256 octets de données. Si la différence est inferieur à 256, la longueur du bloc est égale à la différence entre l'adresse du début et l'adresse de fin .

Ensuite un caractère "B" est inscrit sur la bande, suivi d'un octet indiquant la longueur du bloc.

Les 2 octets suivants comprennent l'adresse de début de la zone mémoire contenant les données à mettre sur la bande. Puis 25 "1" suivent ce bloc de données.

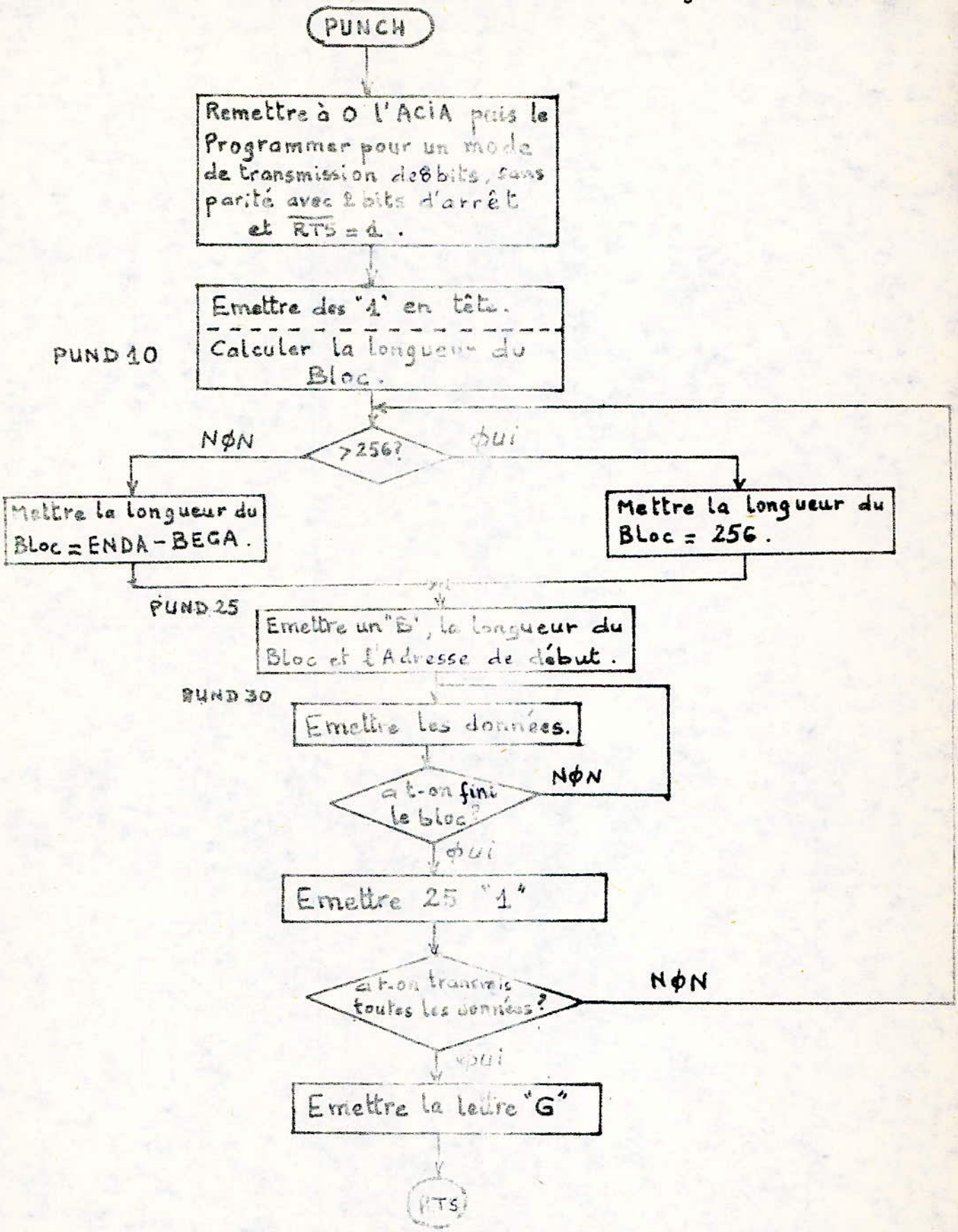
L'adresse de début est de nouveau comparée à l'adresse fin afin de savoir si toutes les données ont été enregistrées.
Si c'est le cas, un caractère "G" indique la fin du bloc.

L'orsque l'adresse de début et l'adresse de fin sont différentes, un autre bloc est inscrit,etc ...

Ce cycle continue jusqu'à ce que les adresses de début et de fin soient identiques.

Le contrôle est redonné au moniteur de gestion (JBUG) du KIT D2, par une instruction RTS.

Organigramme de la fonction "PUNCH" : Enregistrement d'u file de caractères



b) LECTURE "LOAD"

En appuyant sur la touche L: LOAD : on fait appel au programme de transfert de données vers la mémoire.

On programme l'ACIA pour avoir le même format que lors du PUNCH (8 bits de données, pas de parité, 2 bits stop).

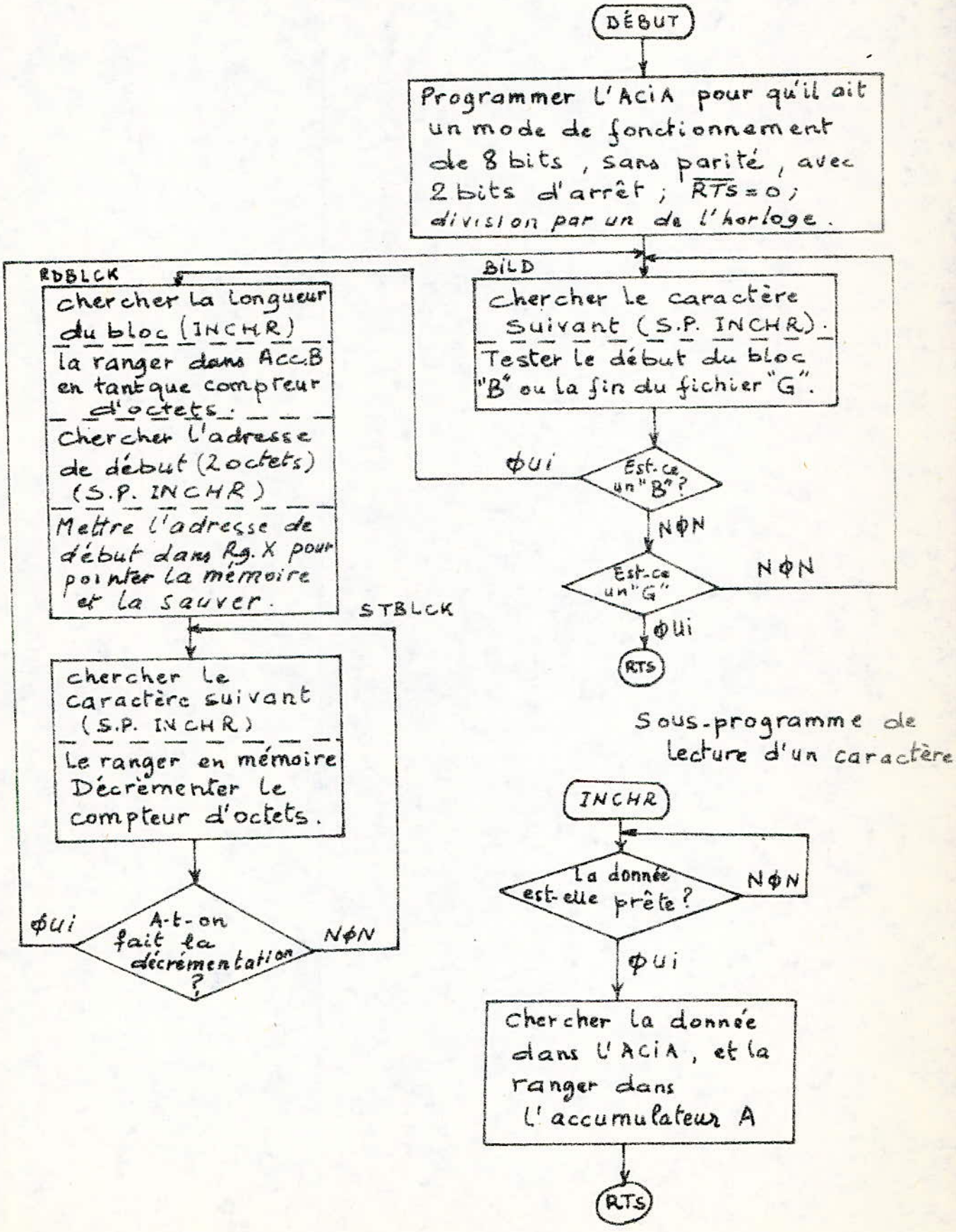
L'horloge de réception est dans le monde de division par un,et $\overline{RTS}$ est mis au niveau bas indiquant que l'ACIA est prêt à recevoir les données venant de l'interface. Chaque Octet de données est lu en appelant le sous-programme INCHR.Celui-ci vérifie d'une manière continuelle le registre d'état de l'ACIA jusqu'à ce que l'indication qu'un octet est prêt à être transmis est détectés.

Le MPU va ensuite chercher l'octet placé dans le registre de réception (ACIA) et il revient au programme LOAD, avec l'octet placé dans l'accumulateur A Cet octet (donnée) est ensuite testé pour déterminer si c'est un caractère "B" ou "G" : si c'est un "B" : le programme se dirige vers le sous-programme de lecture de blocs de données (RDBLCK). Puis la longueur du bloc (octet) est lue et sauvée dans l'accumulateur B, de même que l'adresse de début qui est mise dans des cases mémoires prévues. Une fois ce bloc de données lu, le programme retourne au sous-programme BILD pour déterminer s'il reste un autre bloc.

Lorsque d'autres blocs sont présents dans ce fichier (bande)on les transmet de la même manière.

La fin d'un bloc est reconnue par la lettre "G" dans le programme BILD. La reconnaissance de "G" donne le moyen de quitter le programme de lecture (LOAD) en exécutant l'instruction RTS.

Organigramme "LOAD" : Lecture d'une file de caractères.



B - ETUDE DE L'INTERFACE DU KIT D5 :

A L'instar du KIT D2,le KIT D5 est doté d'un interface cassette, permettant le stockage de données sur une bande magnétique.

B.1 - FORMAT D'ENREGISTREMENT :

a - Format d'un caractère:

Le format du caractère adopté pour le KIT D5 est identique à celui du KIT D2,c'est le "KCS".

Les caractèristiques de ce format ont été definis dans le paragraphe A.1. a-

b - Format d'une file de données:

Avec ce système,les données sont enregistrées sous la forme d'un seul bloc.

A la fin de chaque enregistrement on a l'octet CHECKSUM (ou somme de test) qui représente le complément à 2 de la somme de tous les octets de données et des adresses mémoire de début et de fin de la file.

Le CHECKSUM sert à dépister les érreurs de lecture.

Le format d'enregistrement d'une file de données est organisé de la façon suivante:

- Enregistrement de 30 secondes du caractère "FF"
- Le caractère "53" correspondant à la lettre "S" en code ASC II
- Le "S" est suivi par 4 octets contenant les adresses mémoire de début et de fin de la file de données.
- Enregistrement des données.
- On termine par l'enregistrement de l'octet CHECKSUM.

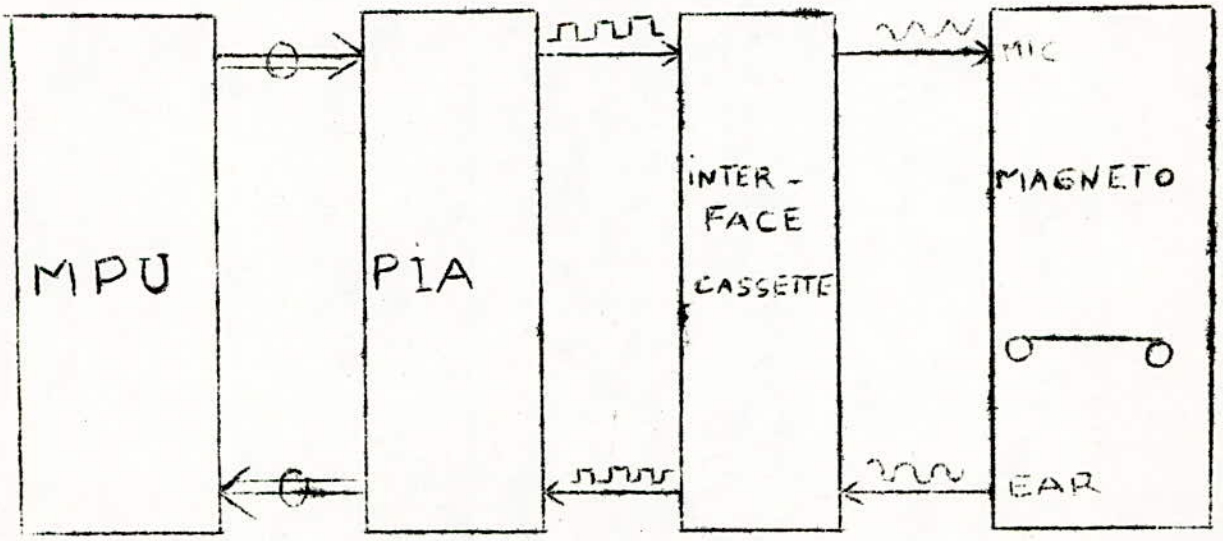
.../...

....	30 secondes de "FF"	caractère "S" début d'une file	Adresse de début de la file: (2 octets)	Adresse de fin de la file (2 octets)	Données	Checksum: (1 octet)	...

Fig...2.7... Format d'enregistrement sur le KIT D5

B - 2 - Principe et fonctionnement de l'interface Cassette du KIT D5 :

a - Schéma synoptique :



- Chaque périphérique necéssite 1 interface assurant l'echange d'informations avec le micropocesseur .

Dans le KIT D5 la communication entre le microprocesseur et l'organe périphérique enregistreur/lecteur de cassettes est assurée par un PIA (périphéral interface adapter).

Le rôle que doit jouer ce circuit d'entrée sortie est déterminé par programme.La définition de la configuration fonctionnelle se fait en écrivant à partir de la mémoire de programme,dans les registres de controle,une information indiquant la fonction que doit réaliser ce PIA.

Une fois que ce circuit est programmé,il est prêt à assurer l'interface entre la périphérie et le microprocesseur.

Le PIA se compose de 2 parties symétriques appelées:

Port A et Port B. Chaque port se compose de 3 registres.

- Registre de données ORA (B)
- Registre de direction de données DDRA (B)
- Registre de controle C R A (B)

Les 6 registres des ports A et B sont considéres par le MPU comme étant 4 adresses mémoire .

b) INTERFACE D'ENREGISTREMENT :

Au moment de l'enregistrement, le PIA transmet vers l'interface des niveaux logiques " o " et"1 ", correspondant respectivement aux tensions 0 et 5 volts chaque niveau a une durée définie par programme.

Un trigger de schmitt de type MC 14584 (voir fig 2.8) sert d'interface de puissance entre la sortie P B7 du PIA et le circuit analogique.

La composante continue du signal issu de l'inverseur est éliminée par la capacité C.32. Avant d'être appliqué à l'entrée " MIC " de l'enregistreur, le signal passe par un filtre passe-bas composé par les résistances R.9 etR.10 et la capacité C 31.

La fonction de transfert de ce filtre étant :

$$FT(P) = \frac{R.10}{R\,9 + R\,10 \;-\; R.9\,R.10\;.\;C.31.\;P}$$

sa fréquence de coupure est Fc = 3388 Hz

Les résistances R.9 et R.10 forment un diviseur de tension, de manière à avoir un signal de niveau convenable à l'entrée MIC.

En effet si on applique à l'entrée 5 volts, on obtient en sortie :

$$V_{MIC} = \frac{R.10}{R.9 + R.10} \;.\; V_{entrée} = \frac{47}{4700 + 47} \times 5$$

$$V_{MIC} = 50\ mV$$

Une fois enregistré, le signal prendra une forme proche d'un signal sinusoïdal du fait de la bande passante du magnétophone et du filtre passe bas.

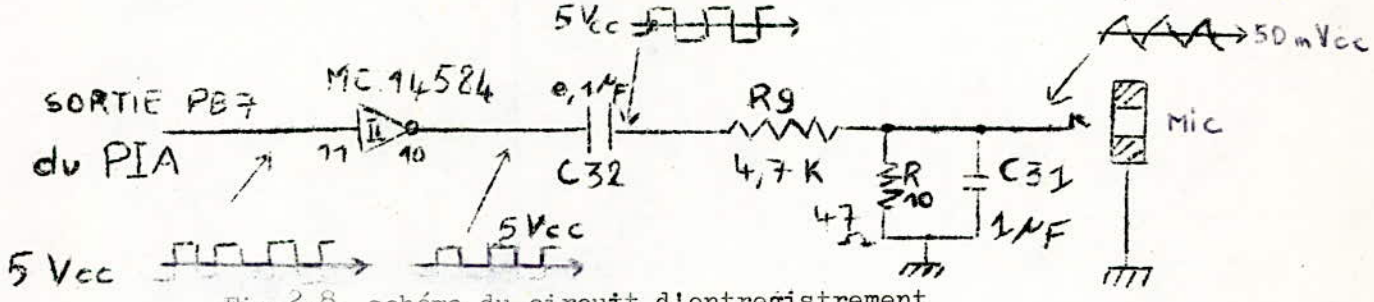


Fig.2.8. schéma du circuit d'entregistrement

C) INTERFACE DE LECTURE :

L'entrée EAR (voir fig.2.9) reçoit le signal enregistré sur la cassette. Ce signal ayant une forme proche d'une sinusoïde passe, tout d'abord, par la capacité C.35, qui éliminera la composante continue, avant d'attaquer l'entrée d'un trigger de schmitt inverseur Mc 14584.

Les 2 inverseurs trigger en série mettent sous forme d'un signal carré, le signal sinusoïdal d'entrée.

La sortie du second inverseur est reliée à l'entrée **C.A.1 du SYSTEM PIA** un logiciel (LOAD/TIN) décode le signal arrivant en CA 1 et le convertit en donnée binaire.

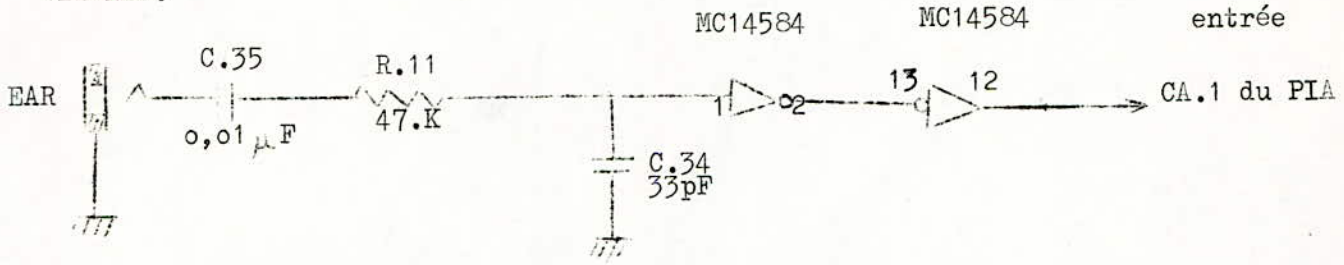


Fig .2.9.. schéma du circuit de lecture

B.3 - LOGICIEL ASSOCIE A L'INTERFACE CASSETTE DU KIT D.5 :

Pour établir le fonctionnement de cet interface, il a fallu étudier les différents listings, construire les organigrammes puis faire le rapprochement entre ces organigrammes pour aboutir au schéma général de fonctionnement de chaque partie.

B.3.1 - PROGRAMME DE GESTION DE L'INTERFACE CASSETTE :
(ROUTINE TAPES)

Lorsqu'on appuie sur la touche P/L du clavier (KEYPAD), le programme principal TAPES est appelé. Ce programme supervise l'éxecution des deux fonctions éssentielles de l'interface : la lecture et l'enregistrement.

Quand TAPES est activée, un test a lieu sur l'indicateur de fonction (FNCFL, ∅ E43E) pour connaitre l'opération commandée (lecture ou enregistrement) (voir fig.2.10)

FNCFL = 0 correspond à un enregistrement
FNCFL ≠ 0 correspond à une lecture
si FNCFL ≠ 0 le programme TAPES fait appel immédiatement à la routine LOAD pour procéder à la lecture de la file de données.

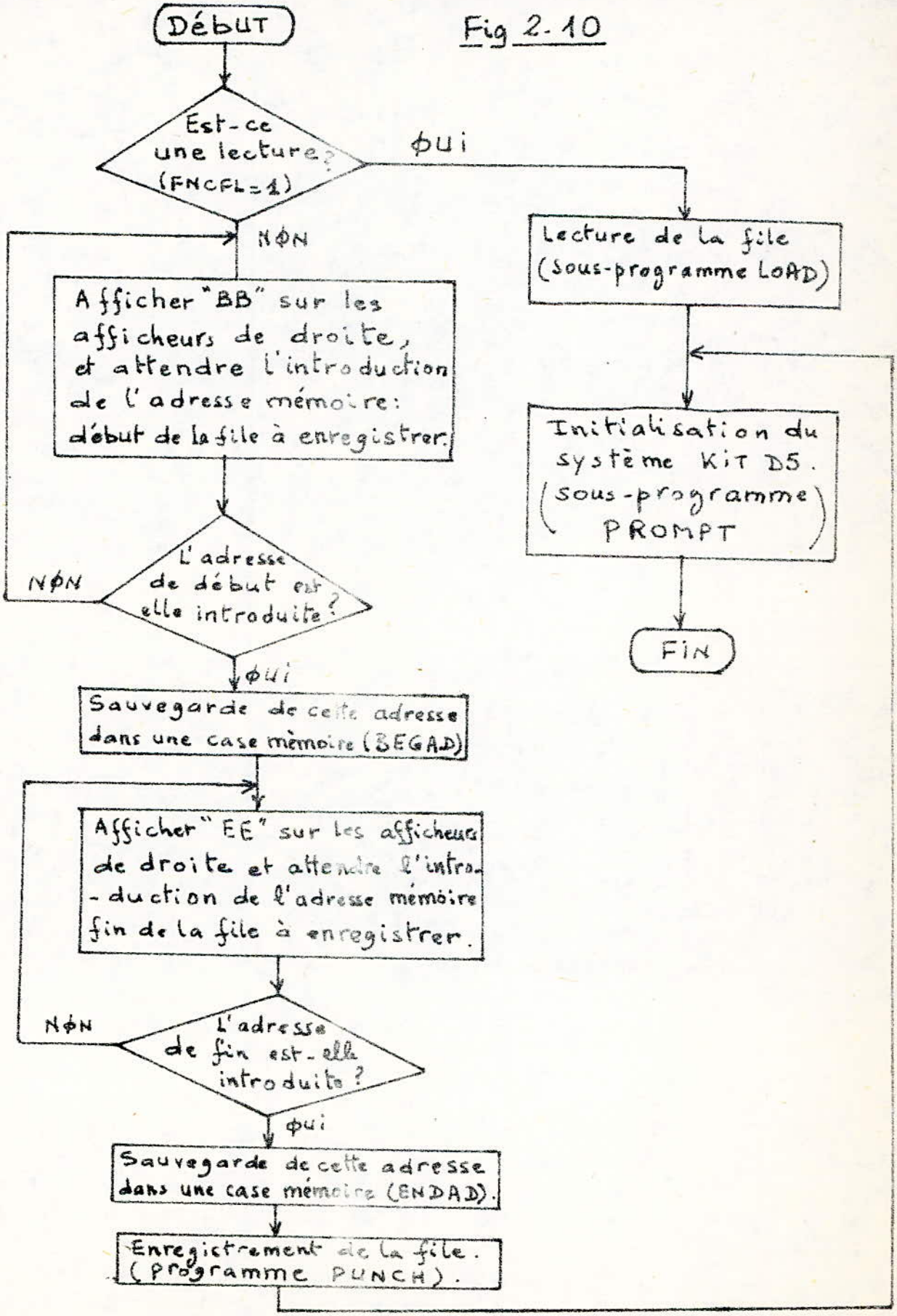
Une fois l'éxecution terminée le contrôle est rendu au moniteur aprés reinitialisation du système KIT D5 par la routine PROMPT

si FNCFL = 0, la location mémoire HEX BUF + 2 (∅ E42E) est chargé avec la valeur héxadécimale " BB", ensuite la routine DYSCOD convertit le contenu des HEX BUFS en code 7 segments avant de le stocker dans les locations mémoires DISBUFS.

On fait alors appel à la routine PUT pour visualiser sur les afficheurs le contenu des DISBUFS. PUT visualise le premier DISBUF pendant 1ms, puis éteint l'afficheur de façon à passer au prochain digit. Pendant la courte période entre l'affichage de 2 digits, une subroutine dont l'adresse est stockée dans la location MNPTR (∅ E419) est éxecutée.

Organigramme "TAPES" : Programme de gestion de l'interface cassette.

Fig 2.10



Durant cette période d'intervalle un test. a lieu sur KYFLG (⌀ E41C) pour savoir si on a appuyé sur une touche du clavier. Si le est positif, le code de la touche est lu par la routine RDKEY.

S'il s'agit d'une touche héxadécimal, le digit est introduit en mémoire et la séquence d'affichage se poursuit. Si par contre c'est une touche de fonction qui à été activée, l'adresse affichée est stockée dans les locations BEGAD (⌀ E460 + ⌀ E461) ; le précedant cycle est repris avec affichage de " EE " au lieu de " BB " Une fois la seconde adresse introduite : l'action sur une touche de fonction entraine le stockage de la seconde adresse dans les locations ENDAD (⌀ EA62 + ⌀ E463). Ensuite TAPES passe le controle à la routine PUNCH pour l'enregistrement des données.

Une fois la fonction PUNCH éxécutée, nous avons une phase de réinitialisation du système par la routine PROMPT, avant de retourner au programme moniteur.

B.3.2. PROGRAMME D'ENREGISTREMENT D'UNE FILE DE DONNES SUR LA CASSETTE : ROUTINE PUNCH

Le programme PUNCH permet l'enregistrement d'une file de caractères sur la cassette. Le format de la file a été défini en B.1.b.

Il s'agit en premier lieu d'enregistrem 30 secondes de caractères " FF " (caractère leader). La valeur correspondant à cette durée est chargée dans le registre d'index du MPU, tandisque l'actet de donnée " FF " est chargé dans l'accumulateur A du MPU. La subroutine d'enregistrement d'un byte PNCHB est établie pour éxécuter la transmission du caractère. Aprés chaque caractère transmis le registre d'index est décrémenté. L'opération est répétée jusqu'à l'enregistrement du dernier byte. On précise que lorsqu'on transmet une série de bytes, il est important de maintenir une synchronisation entre bytes successifs.

Le temps mis à partir du dernier cycle de retour de PNCHB jusqu'au prochain bit " START " à transmettre est éxactement 159 cycles MPU. PNCHB prend éxactement 35 cycles MPU à partir du dernier cycle de l'instruction d'appel jusqu'au premier bit " START " envoyé. Entre le retour de PNCHB et le prochain appel de PNCHB on a 159 - 35 = 124 cycles qui doivent être occupés par un programme par exemple, si 50 cycles sont nécessaires pour établir le prochain caractère, le reste (74 cycles = 124 - 50) doit être utilisé en temps mort avant de rappeler PNCHB.

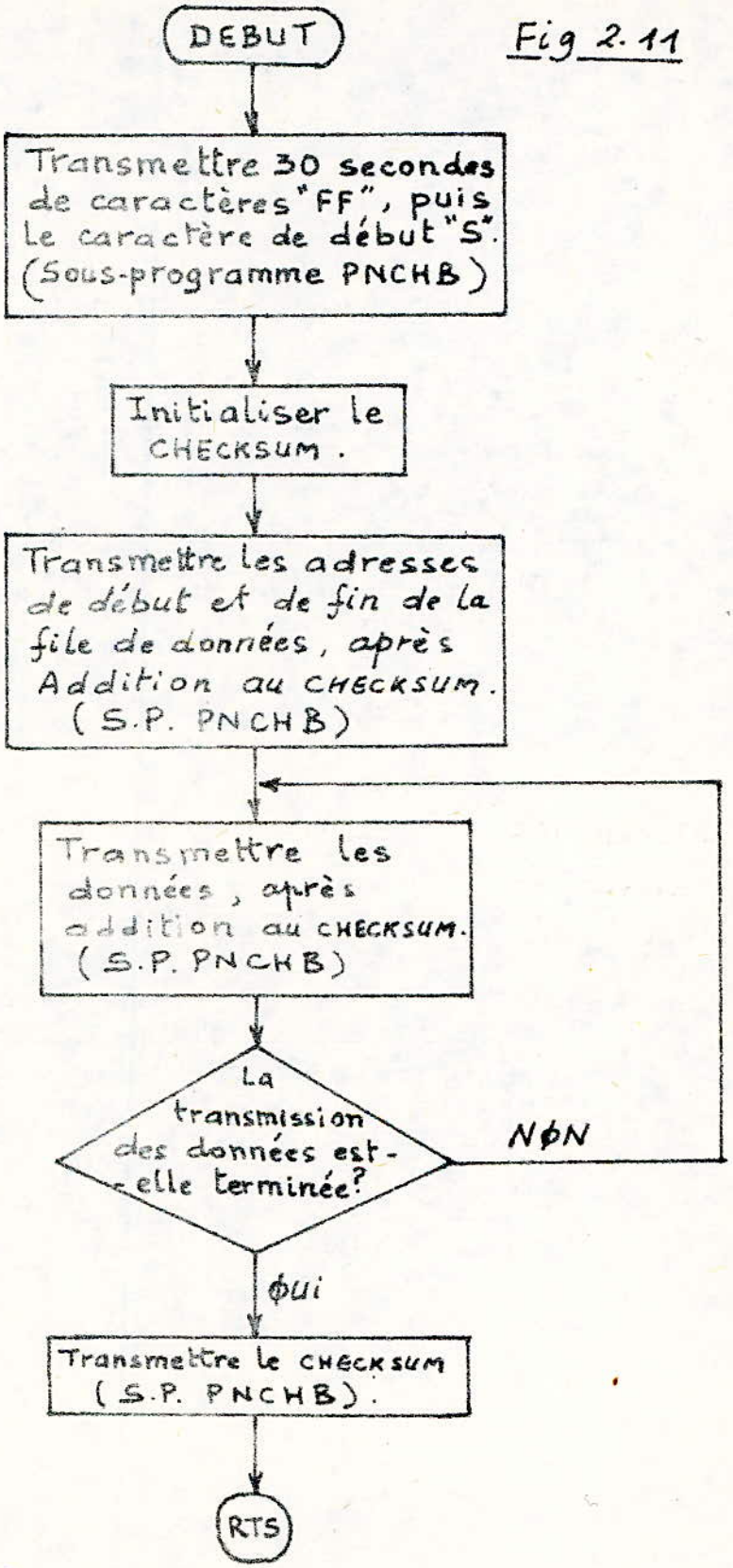
Le premier caractère transmis aprés les 30secondes de leader, est le caractère START " S " début de bloc, suivit par l'envoi des adresses de début et de fin de la file. (voir fig : 2.11)

Rappelons qu'avant d'appeler PUNCH, le programme TAPES avait stocké les adresses de début et de fin dans les locations mémoire BEGAD et ENDAD.

L'initialisation du checksum est éffectuée aprés l'enregistrement du caractère START " S ".

Organigramme "PUNCH" : Enregistrement d'une file de caractères.

Fig 2.11



Chaque octet transmis, indifféremment octet d'adresse ou de donnée, est au préalable additionné au checksum.

L'enregistrement des données succéde immédiatement à celui des adresses.

Pour terminer le checksum est complémenté à 2 avant d'être transmis.

B.3.3. PROGRAMME DE TRANSMISSION D'UN BYTE : ROUTINE PNCHB

La routine PNCHB envoi vers l'interface cassette un caractère de format " KCS ".

PNCHB forme le caractère en ajoutant à l'octet de donnée, le bit START " 0 " début du caractère et 2 bits SOTP " 1 ".

La routine BIT0 est activée pour envoyer 4 périodes d'un signal de 1200Hz correspondant au "0" logique dans le standard "KCS". (voir fig : 2.12)

La donnée est stockée dans la case mémoire BYTE ($ E459). Les bits de la donnée sont enregistrés les uns aprés les autres, en commençant par le bit de poids faible (LSB), pour terminer avec celui du poids fort (MSB)

L'opération de sérialisation est obtenue grâce à une suite de décalages circulaires. En effet l'instruction ROR décale tous les bits de BYTE d'une position vers la droite. Le bit de retenu C du registre d'état du MPU est chargé avec le bit de poids faible de BYTE. Un test a lieu sur le bit de retenu C, du registre d'état, pour savoir s'il s'agit d'un " 1" ou d'un " 0 ".

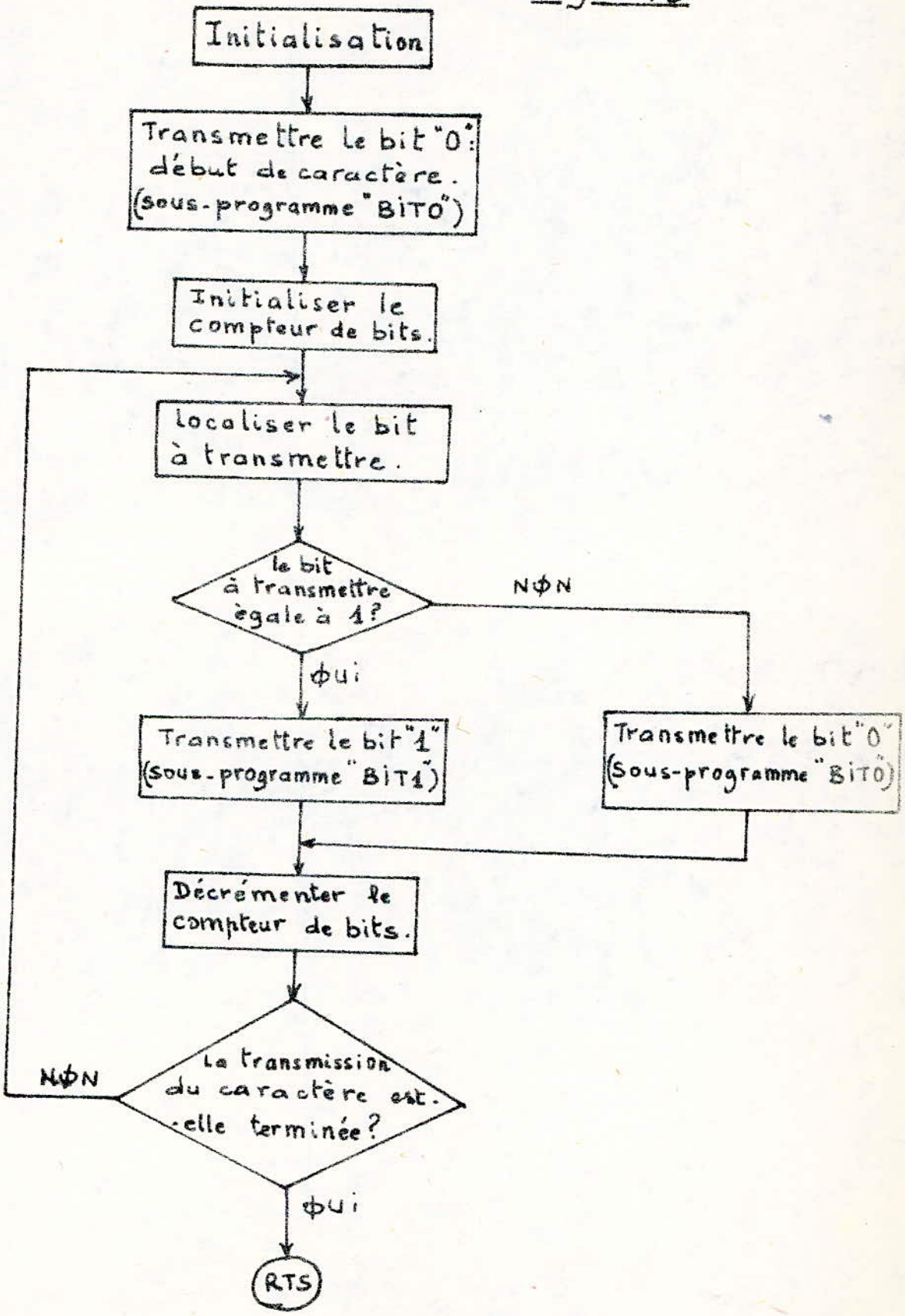
Suivant le résultat du test, c'est la routine BIT0 ou BIT.1 qui est activée Si on a un " 0 " la subroutine BIT0 programmera le "SYSTEM PIA " pour envoyer à travers la sortie P.B7, 4 périodes d'un signal carré de 1200 Hz. Si par contre on a un "1", c'est la subroutine BIT.1 qui envoie, par la sortie PB7, 8 périodes d'un signal carré de 2400 Hz.

PNCHB maintient une synchronisation entre bits pendant la transmission du caractère.

Aprés l'enregistrement de la donnée ; 2 bits STOP " 1 " sont envoyés vers l'interface.

'ganigramme "PNCHB" : Envoi d'un caractère de format "KCS" vers la cassette.

Fig 2-12



B.3.4. - Programme de Chargement ou de Vérification d'une file de données :

Routine LOAD :

La routine LOAD est utilisée pour lire une file de caractéres à partir de la cassette ou pour vérifier un enregistrement vis à vis du contenu de la mémoire. (voir fig : 2.13)

Tout d'abord le programme LOAD recherche le caractére START "S" de début du bloc de données. Par la subroutine de lecture d'un byte TIN. Le byte lu, est comparé. au code ASC II "53" de la lettre "S". Une fois le caractére START "S" retrouvé, on commence la lecture des adresses de début et de fin de la file. Les adresses sont stockées dans les locations mémoires BEGAD et ENDAD.

Notons que chaque octet lu, par TIN, est additionné au checksum. Cette somme servira ultérieurement pour contrôler d'éventuelles erreurs de lecture.

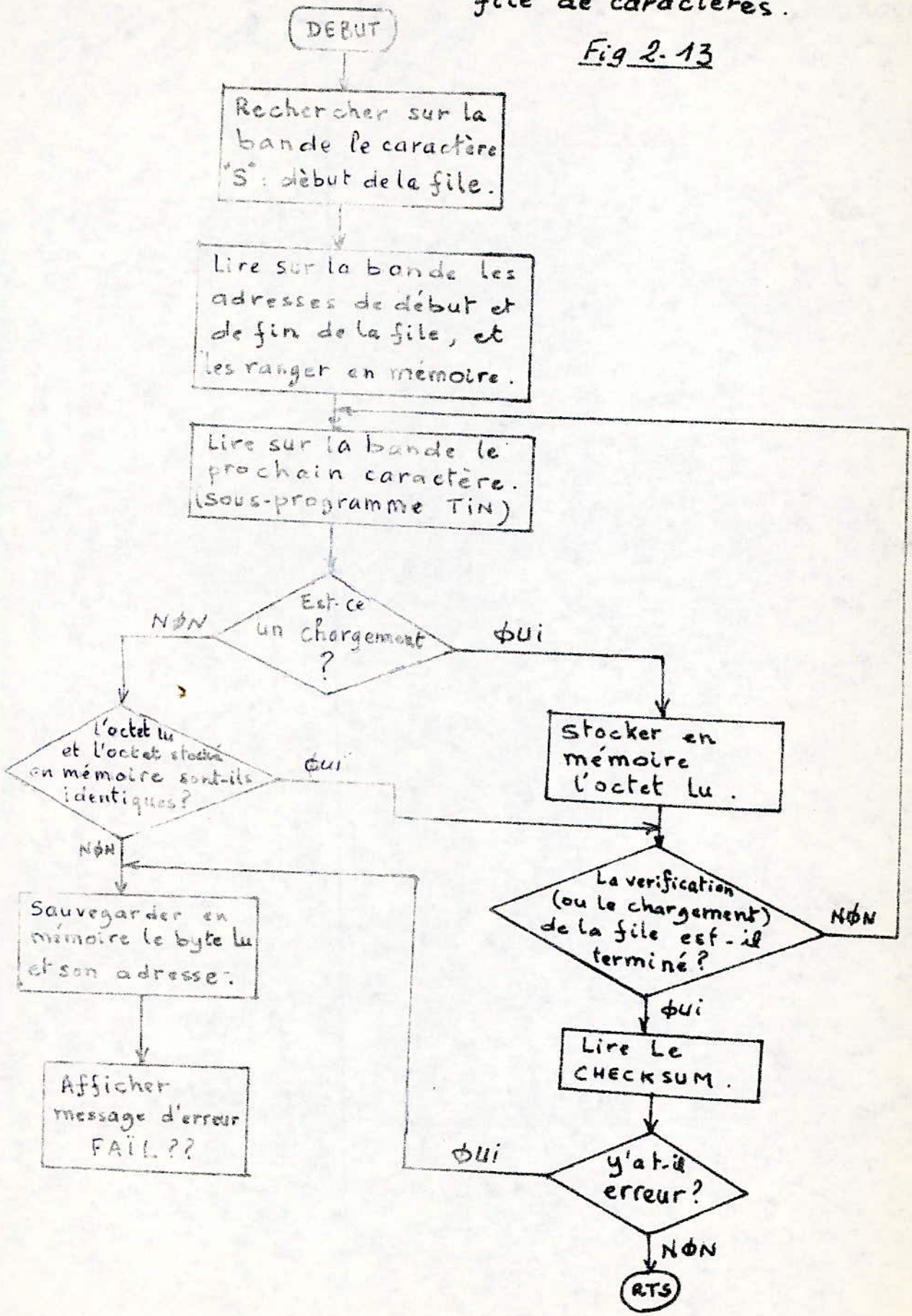
La lecture des données succéde à celle des adresses. Durant cette phase un test est effectué sur le FNCFL (Ø E43E) pour savoir si c'est une opération de lecture qui est demandée, ou une opération de vérification. Dans le premier cas la donnée lue sera stockée en mémoire; dans le second cas la donnée lue sera seulement comparée à celle qui est stockée en mémoire.

Si, pendant une opération de vérification l'octet lu et le byte stocké en mémoire sont différents, le message d'erreur "FAIL?? " apparait sur les afficheurs. L'utilisateur a la possibilité de localiser l'erreur en consultant le contenu des registres du MPU. En éffet l'adresse ou est apparue l'erreur se trouve dans le registre d'index du MPU. De même la donnée lue est stockée dans l'accumulateur A.

Lorsque LOAD éxécute un chargement, les données lues sont stockées en mémoire à partir de l'adresse de début indiquée sur la bande. A la fin de la lecture des données, une comparaison, entre le checksum calculé et le checksum lu sur la bande, est effectuée.

Organi-gramme "LOAD" : programme de lecture d'une file de caractères.

Fig 2.13



Si les deux sommes sont identiques le contrôle est rendu automatiquement au programme moniteur D5BUG. Sinon le message d'erreur " FAIL ??" est visvalisé.

B.3.5. - Programme de lecture d'un byte à partir de la cassette :

Routine TIN

TIN est le programme de lecture d'un caractére de format KCS à partir de la bande magnétique. (voir fig : 2.14)

Cet interface utilise la téchnique "Software Timing". Cette méthode prend la durée d'éxécution des instructions comme unité de mesure du temps.

La routine TIN procède d'abord à la recherche d'une demi-période des bits STOP du précédant caractére. Le registre de contrôle CRA du PIA est alors programmé pour déceler les fronts du signal enregistré présent sur la ligne CA 1, l'arrivée d'un front actif en CA 1 provoque le positionnement de l'indicateur d'interruption (bil 7 du CRA). Une lecture périodique de ce registre permet au MPU de repérer les fronts du signal d'entrée, donc de pouvoir mesurer sa période.

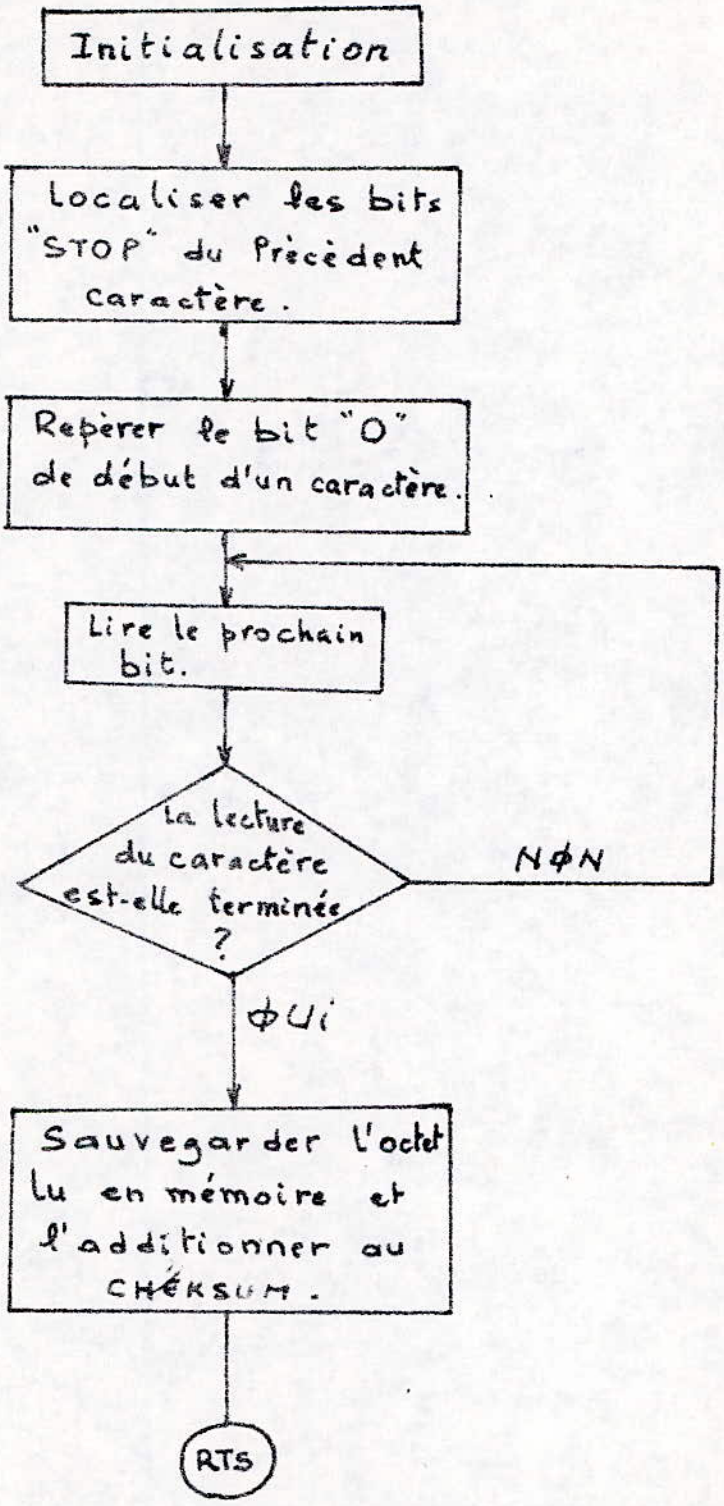
Remarquons que la demande d'interruption de CA 1 est invalidée au niveau du registre CRA.

Aprés la détection d'une demi - période des bits STOP, on re - cherche la première demi - période du bit "0" début d'un caractére. Le bit START "0" est alors lu et stocké dans la location mémoire BYTE (Ø E459), il sera utilisé par la suite comme indicateur de fin de lecture de l'octet.

La lecture des bits de données suit immédiatement celle du bit START.

Organigramme "TIN" (Programme de lecture d'un byte)

Fig 2.14



Nous rappelons que le décodage des "0" et "1" logiques se fait par mesure des périodes de chaque bit. Si cette période est inférieure à 262 Cycles MPU (1710 Hz) il s'agit d'un bit "1". Sinon il s'agit d'un bit "0". Ainsi ou analysera chaque période d'un bit. Un registre GOODAS ($ E45C) enregistre le résultat de chaque test. A la fin de la lecture d'un bit le contenu de GOOD1S représentera le bilan de tous les tests faits. Il ne reste plus qu'à stocker le bit lu dans BYTE ($ E459). Ce processus est répété jusqu'à lecture complète d'un octet.

Avant le retour de la routine TIN l'octet lu est additionné au checksum, puis chargé dans l'accumulateur A.

A la vitesse de 300 Bauds, les bits "STOP" prennent 6,66ms; Durant cette période, le précédant caractére est mis sous le contrôle d'une autre routine. Lorsqu'il s'agit de lire une suite de caractéres, la routine qui contrôle le byte lu doit rappeler TIN au moins un cycle entier des bits "STOP" avant le prochain bit START.

Généralement la durée des bits "STOP" est suffisante pour traiter le précédant caractére.

C- COMPARAISON DES INTERFACES CASSETTES DES KIT D2 et D5.

Tableau comparatif :

	KIT D2	KIT D5
Interface d'entrée Sortie utilisée	ACIA Interface série	PIA Interface parallèle
Horloge utilisée	Externe (MC 6800)	Interne (MC 6802)
Temps de cycle	1 us (MHz)	1,117 us (0,895 MHz)
Circuit d'interface cassette (Enreg - et Lecture)	Complexe	Simple
Conversion du "0" et du "1" logique, en un signal modulé en fréquence et vice-versa	Par Hardware.	Par Softwares
Logiciel de gestion de l'interface K7 (PUNCH et LOAD).	Réduit.	Important.

CONCLUSION :/

D'aprés ce tableau,nous remarquons que le Hardware de l'interface K7 du KIT D2 est plus important que celui du KIT D5.En éffet,le circuit d'interface du KIT D2 utilise en plus d'un ACIA,toute une série de composants tels que des compteurs,des bascules,des portes,etc..

Par contre,celui du KIT D5,non seulement utilise le système PIA du KIT destiné à la gestion des clavier-afficheurs,mais aussi un nombre trés réduit des composants se limitant à 3 capacités,4 résistance et un trigger.

Néanmoins,ce déséquilibre de la partie Hardware est compensé par le software,qui éffectivement est plus réduit pour le KIT D2 que pour le KIT D5,car pour ce dernier,la conversion tension-fréquence se fait entièrement par programme.

Ainsi la tâche du microprocesseur du KIT D2 est allégée lors du déroulement des programmes K7,beaucoup plus qu'elle ne l'est pour celui du KIT D5.

Nous voyons donc,que les 2 interfaces K7 présentent des avantages et des inconvénients,mais d'une manière générale, tous les deux répondent bien aux fonctions qui leur sont demandées.

Ainsi c'est à l'utilisateur de faire un choix judicieux relatif à son application.

Personnellement,nous conseillons l'interface K7 du KIT D5, car celui-ci est:

- Plus économique,du fait qu'il utilise moins de composants.
- Il est assujetti à moins de pannes:donc plus faible.
- Il offre une plus grande souplesse d'utilisation grâce à son software développé.

C H A P I T R E III

REALISATION D'UN INTERFACE K7

AVEC LE SYSTEME MDA 8002A

CHAP - III - REALISATION DE L'INTERFACE CASSETTE AVEC LE TEKTRONIX 8002A

Dans les chapitres précédents nous avons donné le principe général d'un interface cassette, et une étude comparative sur les interfaces cassette de deux micro-ordinateurs le K I T D2 et le KIT D5.

Il nous a été confié de réaliser un interface cassette en liaison avec l'outil de développement : TEKTRONIX 8002A qui se trouve au laboratoire d'électronique appliquée de l'ENP. Cet interface offrera à l'utilisateur, la possibilité de sauvegarder et de relire ses programmes ou données sur cassette.

Pour cela, l'utilisateur n'a qu'à éxécuter un certain programme dont la fonction, est de gérer cet interface. Ce programme de gestion sera stocké sur disque ou EPROM.

Nous commençons donc par donner une brève présentation du système de développement.

1) Présentation du Système de Développement (TEKTRONIX 8002A)

Le TEKTRONIX 8002A est un système de développement qui permet la réalisation d'un prototype de produits à base de microprocesseurs.

Grâce à son fonctionnement en mode "Emulation", le développement de la partie "Hard" de la maquette se fait en parallèle avec la mise au point des programmes correspondants. Dans ce mode de fonctionnement, le M P U de la carte émulateur, se substitue au M P U de la maquette prototype; cette dernière est connectée au processeur émulateur via la sonde on rappelle que l'émulation, permet un test entier de la maquette, et l'éxécution du programme en temps réel

1 - 1) Organisation Matérielle du Tektronix 8002A

Il est conçu autour d'un processeur système(le 2650 de sygnétics), utilisant d'autres MPU et assurant la programmation de PROM, le déverminage, et la gestion de l'information.

De plus, il est équipé de :

- Un processeur "assembleur" qui éxécute l'assemblage d'un programme écrit par l'utilisateur en code source.
- Un processeur "émulateur" qui éxécute et assure le devèrminage des programmes utilisateurs.
- Une mémoire système de 16 K octets (RAM) et d'une PROM occupant les 256 premiers octets.
- Une mémoire utilisateur de 32 K octets.
- Une sonde assurant le lien entre la maquette et le processeur émulateur.

- Un programmeur de PROM permettant le transfert du programme utilisateur, de la mémoire programme vers une PROM (ROM Programmable).

Le système 8002A est relié à une unité double de disque, qui constitue la mémoire de masse du 8002A (la capacité mémoire d'un disque est de 315K octets); et un système terminal, constitué d'un clavier et d'une console de visualisation, permettant le dialogue avec le système

1 - 2) Logiciel du Système :

Il est constitué de plusieurs modules on y trouve un logiciel de base "TEKDOS"; un Editeur de texte permettant la création de textes, et les modifications éventuelles. Le chargement et le transfert de fichiers entre la mémoire programme et le disque, sont assurés par "TEKDOS" qui est le moniteur du système: il assure la gestion des disques et des fichiers; les fonctions de transfert de données, et les fonctions de contrôle des périphériques et du système.

La conversion d'un programme source (écrit par l'utilisateur) en code objet éxécutable par le MPU se fait à l'aide d'un programme "assembleur".

Un programme de "déverminage" DE BUG permet de suivre l'exécution d'unprogramme, et de déceler d'éventuelles erreurs de logique.

Le chargement en mémoire vive d'un code objet à exécuter; le transfert de ce code objet entre la mémoire vive et le disque, le contrôle du contenu des mémoires, se font à l'aide du programme d'émulation.

Le TEKTRONIX 8002A possède en plus un programmeur de PROM et un logiciel d'utilisation de l'analyseur temps réel

Des possibilités de communication avec d'autres système informatiques, éxistent par le biais d'un logiciel spécifique.

2 - Présentation de la Maquette :

2 - 1) Synoptique : (voir fig 3-1)

La maquette comprend :

- Un support du MPU 6800, sur lequel sera implanté la sonde reliée au microprocesseur émulateur du TEKTRONIX.
- Un PIA MC 6821 qui assure l'échange de données entre le MPU et l'interface.
- Une horloge biphasée MC 6871A, assurant l'activation du MPU et la synchronisation du système.
- Un interface de lecture et d'enregistrement.

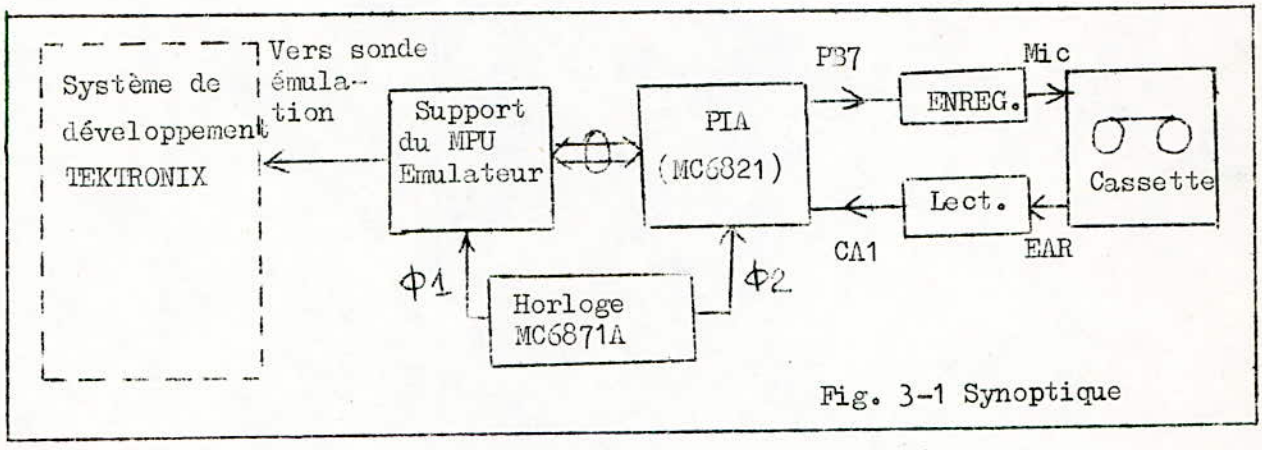


Fig. 3-1 Synoptique

2.2- GENERALITES SUR LES COMPOSANTS UTILISE :

2.2.1. le Microprocesseur MC 6800:

Le Microprocesseur utilisé est celui du système de développement: processeur émulateur.

Il se présente sous forme d'un boitier de 40 broches, monolithique en technologie LSI (N MOS).

Son rôle est celui d'une unité centrale d'un micro-ordinateur, qui travaille sous l'impulsion d'une horloge externe de 1 MHZ (MC 6871 A).

Il traîte des mots de 8 bits en parallèle; et a une capacité d'adressage de 64 K octets (2^{16}). Il possède 6 modes d'adressage (direct, etendu, indexé, relatif, immédiat et implicite) et un jeu de 72 instructions.

Son architecture interne se résume à 2 accumulateurs A et B, un registre d'index (RX), un pointeur de pile (SP), un compteur ordinal (PC) et un registre d'état (CCR).

Il nécessite une alimentation unique de + 5V (TTL)

pour plus de détails sur le MC 6800, voir bibliographie.

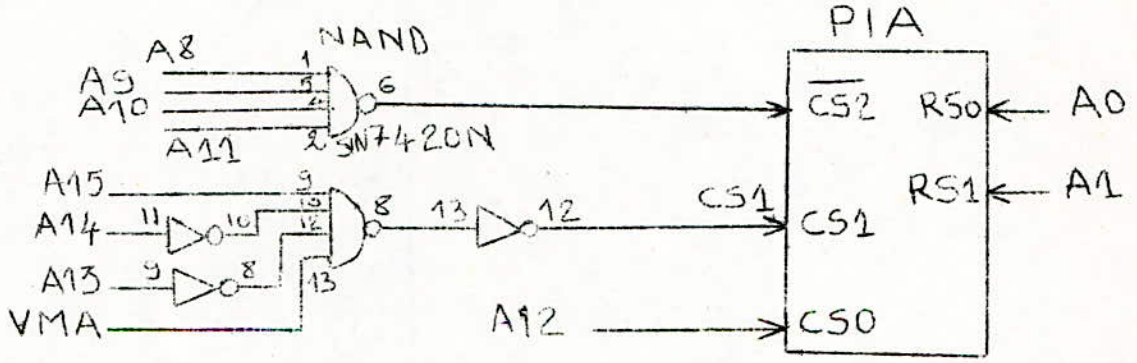
2.2.2. LE PIA MC 6821 :

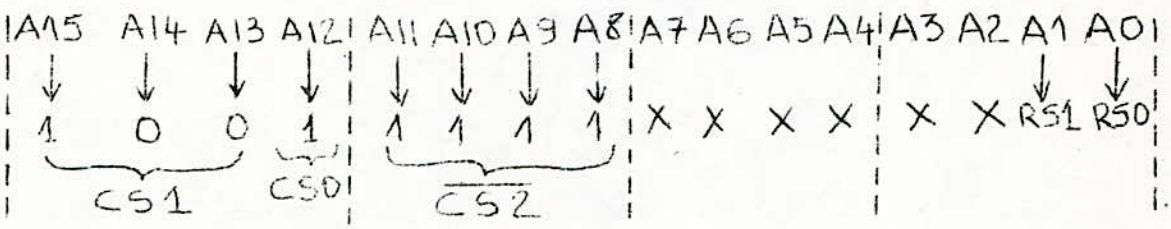
Organe assurant l'échange d'informations entre le microprocesseur et le périphérique. Il travaille en mode parallèle, et est constitué de 2 parties identiques et indépendantes: le port A et le port B. Chaque partie comprend 3 registres à 8 bits chacun.

- Le registre de contrôle: CRA (CRB).
- Le registre de sens de transfert des données : DDRA (DDRB).
- Le registre de données : ORA (ORB).

Chacun de ces registres est considéré par le MPU comme une position mémoire adressable.

Il est adressé de la même manière que le seraient quatre positions mémoires: ainsi 5 lignes permettent son adressage: CS_1, CS_0, $\overline{CS_2}$, RS_0; RS_1 - Ces dernières seront commectées aux lignes d'adresse du microprocesseur à travers des port Nand et des inverseurs. Voyons comment s'effectue le décodage d'adresse du PIA:





On obtient les adresses suivantes pour les registres du PIA avec une combinaison de RS_0, RS_1.

RS1	RS0	Registre sélectionné
0	0	DDRA/ORA → 9F00
0	1	CRA → 9F01
1	0	DDRB/ORB → 9F02
1	1	CRB → 9F03

Pour plus de détails sur le PIA, consulter annexe

2.2.3. HORLOGE MC 6871 A :

Ce cirduit délivre les 2 signaux d'horloge $\phi 1$ et $\phi 2$ pour activer le microprocesseur et la synchromisation du système.

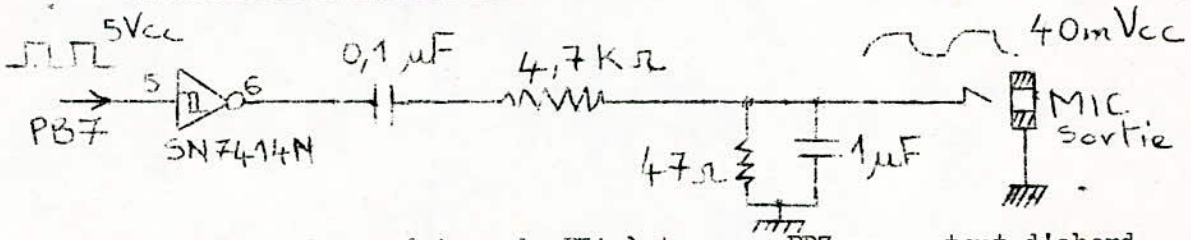
Sa fréquence d'horloge est de 1 MHZ.

2.2.4. LES CIRCUITS D'ENREGISTREMENT ET DE LECTURE :

voir schéma de la maquette Fig 3.2.

Ces circuits transforment le signal afin de l'enregistrer ou de le lire.

a) CIRCUIT D'ENREGISTREMENT :



Le signal carré issu du PIA à travers PB7 passe tout d'abord par une porte (SN 7414) qui assure l'adaptation entre les circuits logiques et l'amplificateur du magnétophone.

Ce signal passe ensuite par les cellules RC qui représente respectivement un filtre et un diviseur de tension, car pour être enregistré sur cassette, le signal doit avoir un niveau convenable de 40 mVcc :

En effet, on obtient bien ce niveau :

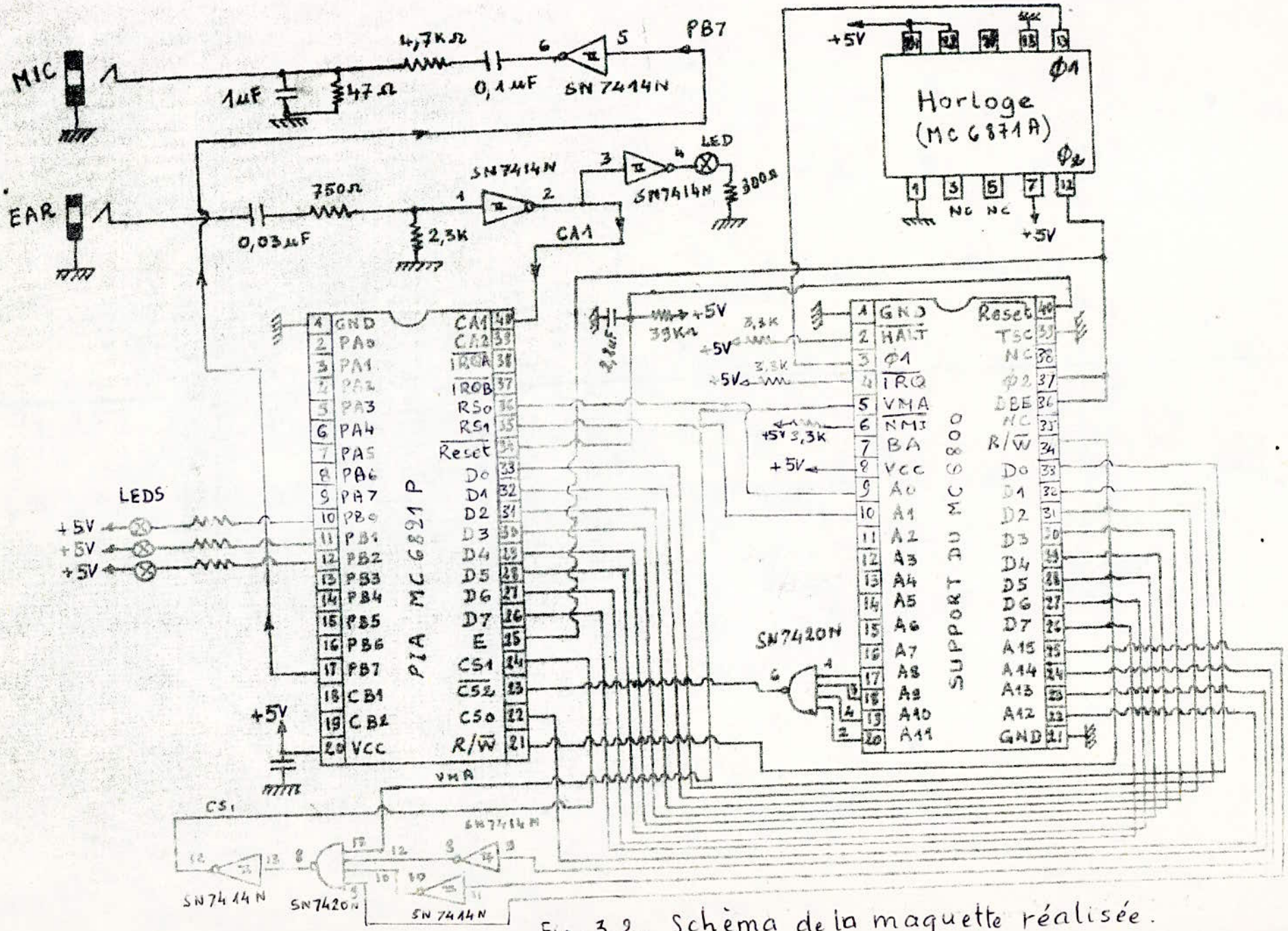


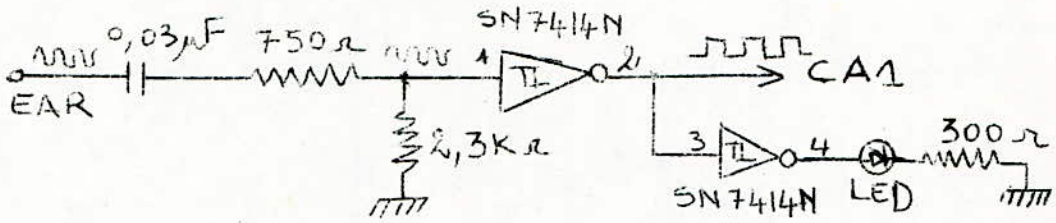
Fig. 3.2. Schéma de la maquette réalisée.

$$V_{MIC} = \frac{47}{4700 + 47} \times 4 = \underline{40\ mV_{cc}}$$

Ce signal aura une forme "arrondi" à cause de la charge et de la décharge de la capacité de 1 uF.

Une fois enregistré,il prendra une forme proche d'un signal sinusoïdal dû à la bande passante du magnétophone.

b) CIRCUIT DE LECTURE :



Le signal enregistré sur cassette,est recueilli à l'entrée EAR (Ecouteur).ce signal passe par une capacité de filtrage de 30 nF (filtrage des basses fréquences et de la composante continue),puis par un diviseur de tension qui fixe le seuil de basculement du trigger.Ce dernier permet la mise sous forme d'un signal carré de 4 volts.Celui-ci attaque le PIA par CA1.

La présence de la LED en sortie (cA1) nous renseigne sur l'éxistence du signal lors de la lecture.

3) CABLAGE DE LA MAQUETTE :

IL suffit d'interconnecter correctement les différents éléments de la maquette.

3.1- LIAISONS MPU - PIA :

Le MPU dialogue avec le PIA par:

- Do - D7:représentent les 8 lignes bidirectionnelles de données,à travers lesquelles le MPU et le PIA échangent des informations.
- R/$\overline{W}$:ligne de lecture/Ecriture.

  R/$\overline{W}$ = 1 ——→ Le MPU va lire

  R/$\overline{W}$ = 0 ——→ Le MPU va écrire.
- $\overline{Reset}$:Remise à zéro du MPU et du PIA.
- ϕ 1 :issu de l'horloge :active le MPU.
- ϕ 2 : issu de l'horloge:relié à la broche E du PIA et aux broches ϕ 2 et DBE du MPU.

  ϕ 2 synchromise le transfert de données.
- D BE relié à ϕ 2:valide le bus de données.
- VMA: validation des positions mémoires du PIA.
- (A11,A10,A9,A8,A15,A14,A13,A12)sont reliés à $\overline{CS_2}$, CS_1,et CSo pour le décodage d'adresse du PIA.

3.2 - LIAISONS PIA - PERIPHERIQUE:

La ligne CA1 programmée en entrée,est reliée au circuit de lecture de l'interface cassette.

La ligne PB7 programmée en sortie,est reliée au circuit d'enregi- t strement.Pour plus de détails sur le câblage voir Schéma de la maquette Fig 3.2.

4- PROCEDURE DE MISE AU POINT DE LA MAQUETTE :

Aprés le câblage de la maquette et sa vérification,on passe à sa mise au point à l'aide du système de développement.

L'utilisation du TEKTRONIX 8002 A en moder d'émulation 1 est nécessaire, du fait que notre maquette utilise une horloge externe au MPU et des positions mémoires (registres du PIA).

On rapelle que l'Emulation est une technique de contrôle des micro-processeurs dans un procédé ou l'éxécution du software peut être observée et controlée.Et que l'émulation doit être en temps réèl :c'est à dire que le procédé sous contrôle continue à fonctionner,sans ralentissement et sans qu'il y'ait d'états d'attente introduits par le système de développement.

Les modes d'Emulation utilisés sont:

- Le mode "Ø" dans lequel toutes les activités prennent part à l'intérieur du système de développement.Dans ce mode l'Emulation utilise la mémoire programme pour éxécuter le programme utilisateur, sans faire appel au prototype (maquette).

Dans le mode "1" :les fonctions d'entrée/sortie,d'horloge et la mémoire (PIA)sont fournis par la maquette,ce qui permet la mise au point de cette maquette et du logiciel qui lui est associé.
qu'est ce que le travail en temps réèl?
C'est enregistrer en temps réèl,c'est à dire à pleine vitèsse de travail de la maquette testée,l'ensemble de l'activité des différents bus du MPU ainsi que la logique associé (tests des entrées/sorties, des interruptions,périphériques logique associée au MPU,etc...).

- LA PROCEDURE EST LA SUIVANTE :

Tout d'abord nous devons introduire notre programme source dans la mémoire programme. On appelle l'Editeur de texte avec la commande "EDIT" de TEKDOS,suivi du Nom du programme (LECT ou ENREG).

De le tabuler,en faisant la commande "X TABS ON" de l'Editeur, d'invoquer la commande INPUT de l'Editeur pour commencer à écrire le programme source.Une fois l'écriture achevée,on appuye 2 fois sur la touche "RETURN" pour pouvoir utiliser d'autres commandes.C'est la phase d'Edition.

De plus,on doit utiliser la commande "file" pour transférer le programme source de la mémoire programme vers l'unité de disque.

- PHASE D'ASSEMBLAGE :

La commande "ASM" de TEKDOS permet l'assemblage du programme source et l'obtentiondu programme en code objet éxécutable par le MPU. Une fois assuré,que le programme est sans erreurs .Nous passons au chargement.

- PHASE DE CHARGEMENT :

Le chargement en mémoire programme du code objet,se fera par la commande de TEKDOS "LOAD" .L'Exécution du programme se fait en mode d'Emulation 1.

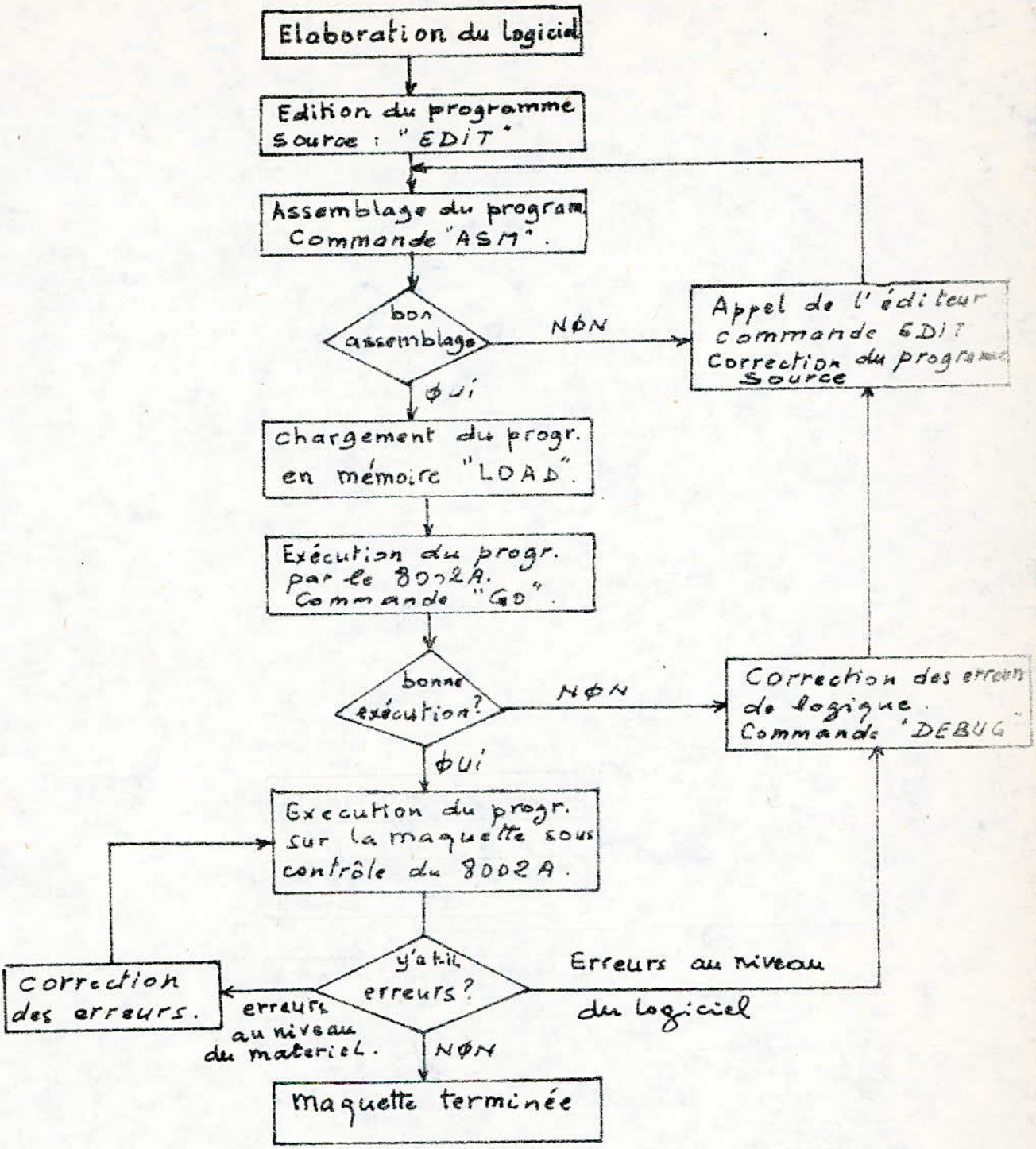


fig. 3.3. Organigramme schématisant la procédure de mise au point de maquettes à microprocesseurs sur le TEKTRONIX 8002A.

- PHASE D'EXECUTION :

Le contrôle de l'éxécution du programme objet se fait sous la commande TEKDOS "DEBUG" .L'Exécution se fait sous la commande "G O" en affichant l'adresse de départ du programme d'application,qu'on verra au chapitre IV.

REMARQUE :/

Pour mener à bien,la procédure de mise au point de la maquette,il faut prendre soin de relier correctement la sonde "Emulateur",sur le support du M C 6800 implanté sur la maquette;de brancher les entrées ou sortie (M I C ou EAR) de la maquette respectivement aux entrées ou sorties (MIC ou EAR) du magnétophone à cassette,que ce soit pour le mode "Enregistrement" ou le mode "Lecture ".

Pour bien comprendre,le procédé de mise au point de la maquette,se referer à la Fig 3.3.

C H A P I T R E IV

LOGICIEL DE GESTION

DE L'INTERFACE

4 - LOGICIEL DE GESTION DE L'INTERFACE :

Un logiciel composé de 2 programmes (ENREG et LECT) gére le fonctionnement de l'interface K7 avec le MDA 8002 A.
Examinons la tâche de chacun de ces deux programmes :

4.1. - Programme d'enregistrement "ENREG" :

4.1. a - Procédure d'éxécution de "ENREG" :

Grâce à ce programme l'opérateur utilisant le systéme MDA 8002A peut stocker sur une cassette les données présentes dans la mémoire utilisateur.

Avant de lancer le programme d'enregistrement on doit faire entrer les informations suivantes :

- L'adresse de début du bloc de données que l'on veut enregistrer dans les locations mémoire BEGAD (⌀ FEFC et ⌀ FEFD).
- L'adresse de fin du bloc de données dans ENDAD (⌀ FEFE et FEFF).
- Le numéro d'enregistrement dans ⌀ FF00. L'opérateur doit numéroter ses enregistrements dans un ordre croissant en commençant par 00 jusqu'à FF.
- Les 5 cases mémoire allant de ⌀ FF01 à ⌀ FF05 sont à la disposition de l'utilisateur pour écrire le nom de la file à enregistrer.

La mémoire utilisateur du systéme MDA 8002A est composée de 32 Koctets de mémoire vive (RAM). Les premiers, 16 Koctets sont situés entre les adresses 0000 et 3 FFF. Les 16 Koctets restant occupent les adresses C000 à FFFF.

Pour enregistrer sur cassette, commencer par connecter la sonde sur le support réservé sur la maquette ; puis brancher l'alimentation à + 5 V, ensuite relier la sortie MIC au magnétophone par une fiche "Jack".

L'étape suivante consiste à éxécuter les commandes ci-après de manière à charger le programme objet ENREGO dans la mémoire de travail :

1 - EM 1 <CR>

2 - LO ENREGO <CR>

3 - MA U 9F00 <CR>

4 - DEB <CR>

5 - BKPT FFF0 <CR>

Après cela le programme "ENREGO" se trouve charger de l'adresse mémoire FF09 à FFF0. On a introduit un point d'arrêt à la fin du programme pour rendre le contrôle au moniteur TEKDOS après l'éxécution du programme.

La dernière opération consiste à introduire les adresses de BEGAD et ENDAD, le numéro et le code de la file à enregister.

Une fois toutes ces informations introduites il ne reste plus qu'à éxécuter l'enregistrement en lançant le programme ENREGO.

- G FF09 <CR>

Notons que durant l'éxécution 3 LEDS s'allument. Elles ne s'éteindrons qu'à la fin d'éxécution de ce programme.

4.1. b - Organigramme de "ENREG" :

La figure 4.1 représente l'organigramme "ENREG".

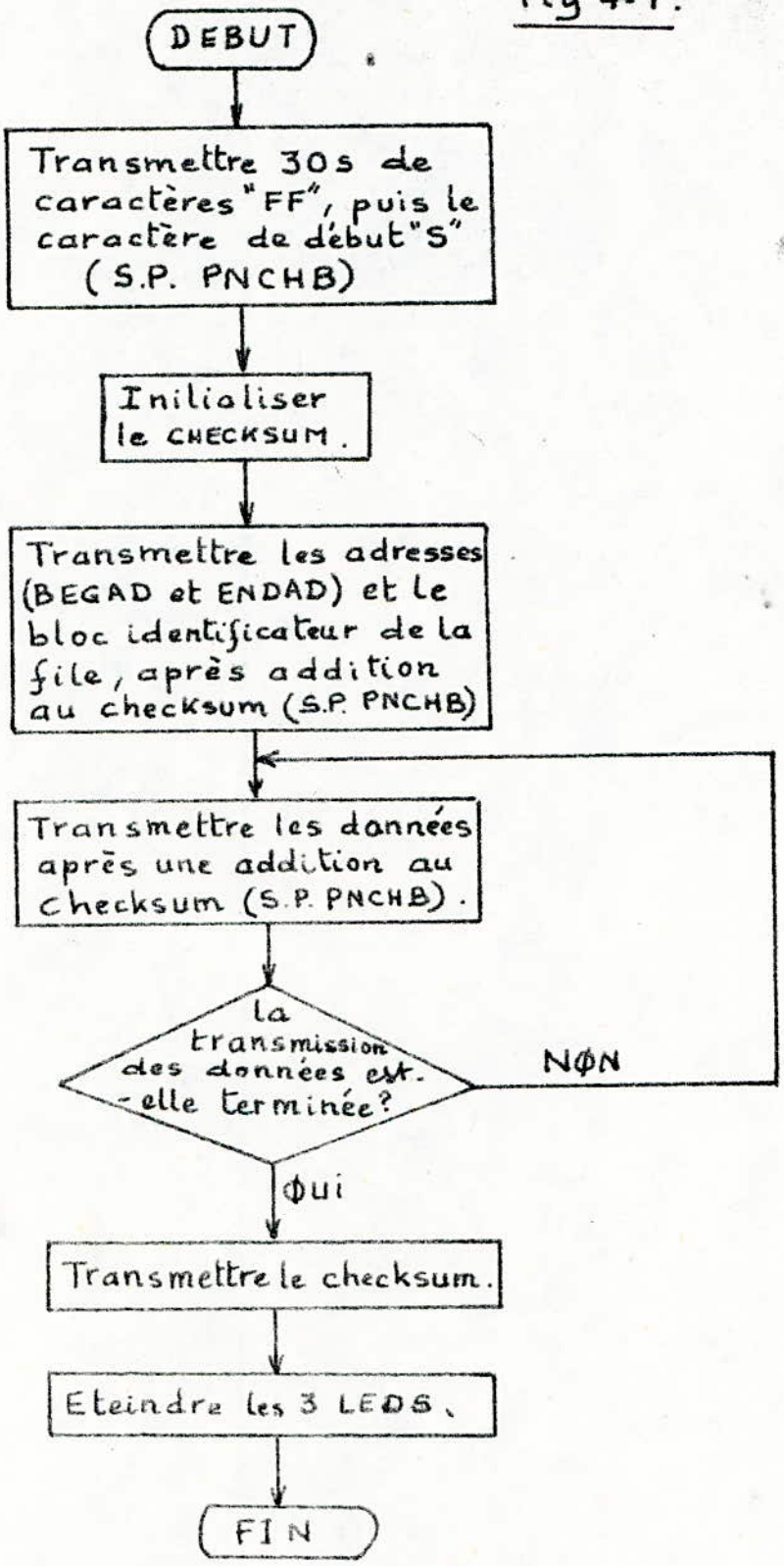
Après une séquence d'initialisation, 30 secondes de caractére leader "FF" et le caractére START "S" début de la file de données sont transmis vers l'interface K7. Pour réaliser cette opération le sous-programme PNCHB est appelé . L'organigramme "PNCHB" est donné dans le chapitre 2 paragraphe B.3.3.

On transmet les adresses BEGAD et ENDAD puis le numéro d'enregistrement ; ensuite le contenu des 5 cases mémoire du nom, ceci constituera le bloc d'identification de la file. On passe ensuite à l'enregistrement des données.

Pour terminer on envoie le CHECKSUM.

Organigramme "ENREG" : Enregistrement d'une file de caractères.

Fig 4-1.



4.1. c - Listing de "ENREG" :

Nous donnons ci-après le listing du programme "ENREG"

4.2. - Programme de recherche et de lecture d'une file de données "LECT" :

4.2. a - Procédure d'éxécution de "LECT" :

Le programme LECT réalise la recherche et la lecture, ou la vérification, de données enregistrées sur cassette.

Avant de commander l'éxécution de ce programme, l'utilisateur doit faire entrer les informations suivantes :

- Le numéro de l'enregistrement que l'on veut lire dans $ FDBB.
- On choisit l'opération à éxécuter
  * FNCFL ($ FDBC) = 00 c'est une vérification
  * FNCFL ($ FDBC) ≠ 00 c'est une lecture.
- La sélection de la zone de stockage des données que l'on va lire, en inscrivant :
  * L'adresse de début dans $ FDBD et $ FDBE
  * L'adresse de fin dans $ FDBF et $ FDCO.

Ce programme apporte plusieurs possibilités concernant la zone de stockage des données. Parmi ces possibilités nous citons :

a - Si l'on veut stocker la file dans la zone d'où elle a été enregistrée, il suffit de mettre "FF" dans $ FDBD, FDBE, FDBF et $ FDCO.

b - On peut donner seulement l'adresse de début et mettre "00" dans $ FDBF et $ FDCO. Les données seront alors chargées à partir de l'adresse de début indiquée .

c - On a la possibilité d'introduire uniquement l'adresse de fin et on met "FF" dans les cases réservées à l'adresse de début.

d - Si on inscrit les deux adresses, de début et de fin, les données seront stocker dans la partie désignée.

ENREG : PROGRAMME D'ENREGISTREMENT D'UNE FILE DE DONNÉES SUR K7.

INITIALISATION DU PIA

```
0001  FFØ9                        ORG  Ø FFØ9 H
0002  FFØ9  8E FFF7               LDS # ØFFF7H
0003  FFØC  7F 9FØ3               CLR  Ø9FØ3H  - Initialisation PIA
0004  FFØF  86 FF                 LDA A# ØFFH
0005  FF11  B7 9FØ2               STA A 9FØ2H
0006  FF14  86 Ø4                 LDA A# Ø4H
0007  FF16  B7 9FØ3               STA A Ø9FØ3H
0008  FF19  86 8Ø                 LDA A # 8Ø H
0009  FF1B  B7 9FØ2               STA A 9FØ2H
0010  FF1E  7E FF 8Ø              JMP  PUNCH
```

Transmission d'un 1 logique

```
0011  FF21  C6 ØF      BIT1     LDA B # ØF H
0012  FF23  BD FF4F    LOPB1    JSR INVRT
0013  FF26  86 18               LDA A # 18H
0014  FF28  4A        BIT 1Ø   DEC A
0015  FF29  2A FD               BPL  BIT1Ø
0016  FF2A  2Ø ØØ               BRA  BIT11
0017  FF2C  5A        BIT 11   DEC  B
0018  FF2E  26 F3               BNE  LOPB1
0019  FF3Ø  BD FF 4F            JSR  INVRT
0020  FF33  39                  RTS
```

Transmission d'un 0 logique

```
0021  FF34  C6 Ø7      BITØ     LDA B # Ø7H
0022  FF36  BD FF4F    LOPBØ    JSR INVRT
0023  FF39  86 38               LDA A # 38H
0024  FF3B  4A        BITOØ    DEC A
0025  FF3C  2A FD               BPL  BITOØ
0026  FF3E  Ø1                  NOP
0027  FF3F  5A                  DEC B
0028  FF4Ø  26 F4               BNE  LOPBØ
0029  FF42  BD FF.4F            JSR  INVRT
0030  FF45  86 1D               LDA A # 1D H
0031  FF47  4A        BITO1    DEC A
0032  FF48  2A FD               BPL  BITO1
0033  FF4A  7E FF4D             JMP  B1
```

```
0034  FF4D  01        B1     NOP
0035  FF4E  39               RTS
```

Routine de transmission d'un front montant ou descendant vers la cassette.

```
0036  FF4F  86 80     INVRT  LDAA #80H
0037  FF51  B8 9F02           EOR A 9F02H
0038  FF54  B7 9F02           STA A 9F02H
0039  FF57  39               RTS
```

Enregistrement d'un octet sur la bande: y compris le bit start, données, et les bits stop.

```
0040  FF58  B7 FF07  PNCHB  STA A 0FF07H
0041  FF5B  8D D7           BSR  BIT0
0042  FF5D  86 09           LDA A #09H
0043  FF5F  B7 FF08         STA A 0FF08H
0044  FF62  7D FF62         TST
0045  FF65  86 13     LPOUT LDA A #13H
0046  FF67  4A         PN0   DEC A
0047  FF68  2A FD           BPL  PN0
0048  FF6A  0D               SEC
0049  FF6B  76 FF07         ROR  0FF07H
0050  FF6E  25 05           BCS  D01
0051  FF70  8D C2           BSR  BIT0
0052  FF72  7E FF7A         JMP  NDBIT
0053  FF75  8D AA     D01   BSR  BIT1
0054  FF77  7E FF7A         JMP  NDBIT
0055  FF7A  7A FF08 NDBIT   DEC  0FF08H
0056  FF7D  2A E6           BPL  LPOUT
0057  FF7F  39               RTS
```

Enregistrement d'une file de caractères: 30 s de FF, caractère de début; adresses de début et de fin, nom de la file, octet CHKSUM.

```
0058  FF80  CE 0348  PUNCH  LDX #0348H
0059  FF83  86 FF   LLOOP  LDA A #0FFH  : 30 s de "FF"
0060  FF85  C6 10           LDA B #10H
0061  FF87  5A       PUN0   DEC B
0062  FF88  2A FD           BPL  PUN0
0063  FF8A  BD FF58         JSR  PNCHB
0064  FF8D  09               DEX
0065  FF8E  26 F3           BNE  LLOOP
```

Enregistrement du caractère de début, des adresses et du Nom de la file.

```
0066  FF9Φ  86 53          LDA A # 53H  - caractere de début "S
0067  FF92  C6 1Φ          LDA B # 1ΦH
0068  FF94  5A      PUN1   DEC B
0069  FF95  2A FD          BPL PUN1
0070  FF97  BD FF 5B       JSR PNCHB
0071  FF9A  Φ1              NOP
0072  FF9B  7F FFΦ6        CLR ΦFFΦ6H  - Initialisation CHECKSUM
0073  FF9E  CE FEFC        LDX #ΦFEFCH - Transmission des
0074  FFA1  A6 ΦΦ   ADLOP  LDA Φ,X          adresses BEGAD et END
0075  FFA3  16              TAB
0076  FFA4  FB FFΦ6        ADD B ΦFFΦ6H - Addition CHECKSUM.
0077  FFA7  F7 FFΦ6        STA B ΦFFΦ6H
0078  FFA7  Φ1              NOP
0079  FFAB  C6 ΦD          LDA B # ΦDH
0080  FFAD  5A      PUN2   DEC B
0081  FFAE  2A FD          BPL PUN2
0082  FFBΦ  BD FF58        JSR PNCHB
0083  FFB3  ΦB              INX
0084  FFB4  8C FFΦ6        CPX # ΦFFΦ6H
0085  FFB7  26 E8          BNE ADLOP
```

Enregistrement des données

```
0086  FFB9  Φ1              NOP
0087  FFBA  Φ1              NOP
0088  FFBB  FE FEFC        LDX ΦFEFCH
0089  FFBE  A6 ΦΦ   DLOOP  LDA A Φ,X
0090  FFCΦ  16              TAB
0091  FFC1  FB FFΦ6        ADD B ΦFFΦ6H - Addition au checksu
0092  FFC4  F7 FFΦ6        STA B ΦFFΦ6H   des données transm
0093  FFC7  F7 FFΦ6        STA B ΦFFΦ6H
0094  FFCA  C6 ΦB          LDA B # ΦBH
0096  FFCC  5A      PUN3   DEC B
0096  FFCD  2A FD          BPL PUN3
0097  FFCC  BD FF58        JSR PNCHB
0098  FFD2  7E FFD5        JMP B4
0099  FFD5  BC FEFE   B4   CPX ΦFEFEH - Transmission terminé
0100  FFD8  27 Φ3          BEQ DUNDA
0101  FFDA  Φ8              INX
0102  FFDB  20 E1          BRA DLOOP
```

Enregistrement du byte CHECKSUM.

```
0103  FFDD   7Ø FFØ6   DUNDA  NEG ØFFØ6H
0104  FFEØ   B6 FFØ6          LDA A ØFFØ6H  - Transmettre checksum.
0105  FFE3   C6 14            LDA B # 14H
0106  FFE5   5A        PUN 4  DEC B
0107  FFE6   2A 5D             BPL PUN 4
0108  FFE8   BD FF58           JSR PNCHB
0109  FFEB   86 ØF             LDA A # ØFH
0110  FFED   B7 9FØ2           STA A 9FØ2H  - Eteindre les 3 LEDS.
0111  FFFØ   3F                 SWI
                                END
```

o

Pour toutes ces options le programme contrôle les adresses de début et de FIN. Dans le cas où il ne peut pas stocker dans la zone sélectionnée, par suite d'un dépassement, il allume la LED "ADRESSE" pour prévenir l'utilisateur.

Pour lire ou vérifier des données sur cassette. On commence par connecter la sonde, on alimente les circuits puis on branche le lecteur de cassette à l'entrée EAR.

On charge le programme objet LECTO dans la mémoire utilisateur en écrivant sur la console du systéme MDA 8002A les commandes suivantes :

1 - EM 1 <CR>

2 - LO LECTO <CR>

3 - MA U 9FØØ <CR>

4 - DEB <CR>

5 - BKPT FFF1 <CR>

Le programme "LECTO" est alors chargé de l'adresse mémoire Ø FDD5 à Ø FFF1. On a introduit un point d'arrêt à la fin du programme.

La dernière opération consiste à introduire le numéro d'enregistrement, l'opération à éxécuter (FNCFL) et bien sûr les adresses de début et de fin.

Après l'introduction de ces informations, on éxécute le programme en écrivant :

- G FDD5 <CR>

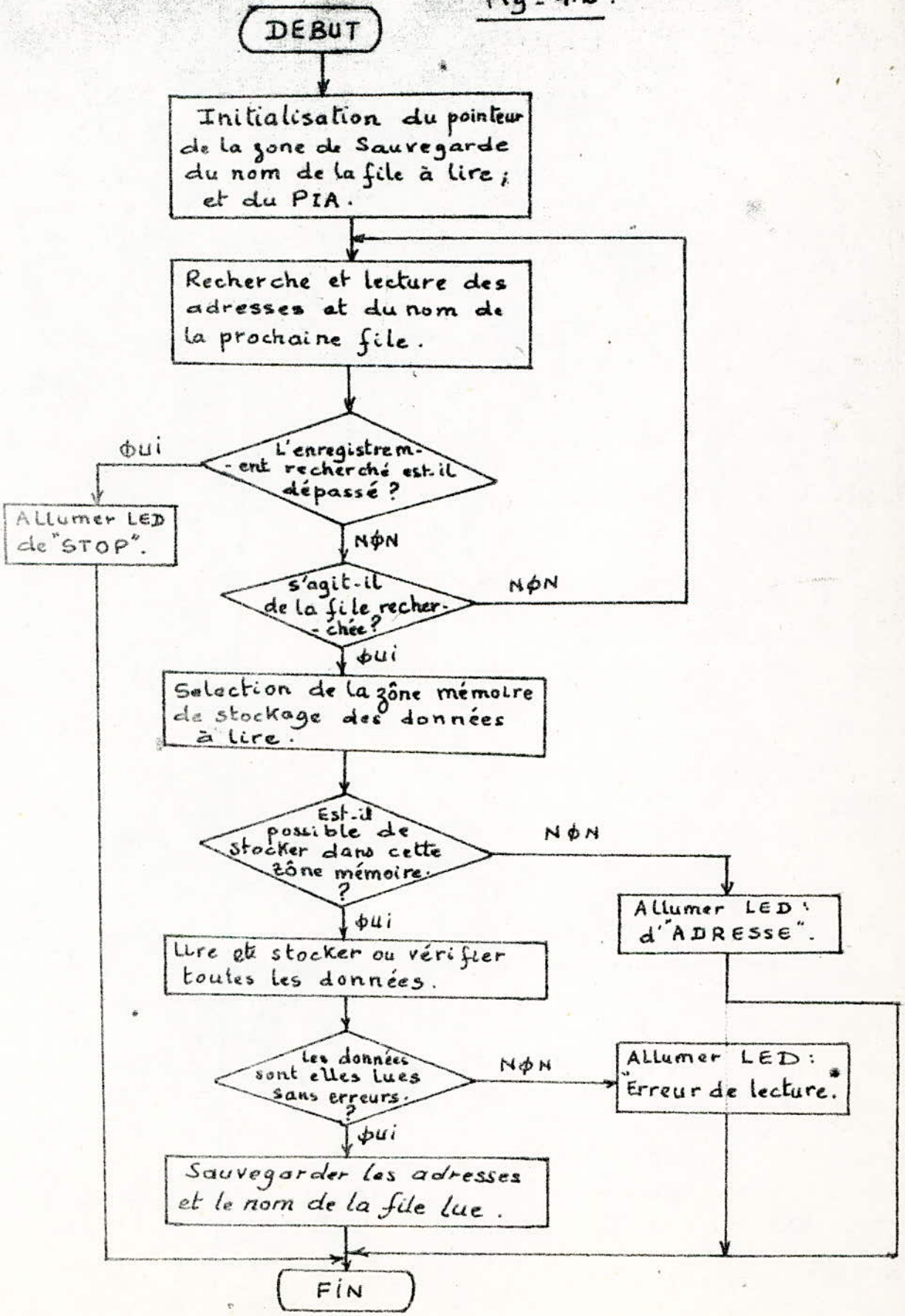
Une LED "TEMOIN" nous indique si les données sont transmises ou non. Il est recommandé au début de réduire le niveau du signal issu du magnétophone puis de l'augmenter progressivement jusqu'à ce que la LED "TEMOIN" s'allume sans palpitations.

4.2. b - Organigramme "LECT" :

La figure 4.2 représente l'organigramme "LECT".

Organigramme "LECT" : Recherche et lecture ou vérificati d'une file de donnée sur K7.

Fig - 4.2 .



Après une phase d'initialisation du PIA et du pointeur de la zone de sauvegarde du nom de la file à lire ; on procéde à la recherche de la prochaine file en repérant 255 octets de caractére leader "FF" et le caractére "S" début de la file. Pour réaliser cette opération le sous-programme TIN est appelé (l'organigramme "TIN" a été décrit dans le chapitre 2 paragraphe B.3.5.)

Une fois le début de la file trouvé, on lit et on stocke les adresses et le bloc d'identification.

Des tests sont effectués. D'abord pour savoir si l'enregistrement que l'on recherche est dépassé. Si c'est le cas la LED "STOP" s'allumera.

Un autre test détermine si l'on est arrivé aux données recherchées.

Une fois que l'on trouve l'enregistrement ; on teste la zone sélectionnée pour stocker les données, si par exemple la zone est située hors de la RAM utilisateur, la LED "ADRESSE" s'allumera. Sinon on stocke les données, si c'est une lecture, ou on fait une comparaison avec le contenu de la mémoire pour le cas d'une vérification. Une erreur survenant durant cette phase provoque l'allumage de la LED "LECT".

Après une lecture ou une vérification sans erreur les adresses et le nom de la file sont sauvegardés dans une zone réservée pouvant contenir 4 noms. Cette zone va de l'adresse $ FD93 à $ FDBA. L'utilisateur consultera cette zone pour connaitre les adresses de ses programmes.

4.2. c - Listing de "LECT" :

LECT : PROGRAMME DE RECHERCHE ET DE LECTURE D'UNE FILE DE DONNÉES SUR K7.

INITIALISATION DU PIA.

```
0001  FDD5                    ORG ØFDD5H
0002  FDD5  8E FFF8           LDS #ØFFF8H  - Initialisation du poin
0003  FDD8  7F 9FØ3           CLR 9FØ3H    - Initialisation PIA.
0004  FDDB  86 FF              LDA A #ØFFH
0005  FDDD  B7 9FØ2           STA A 9FØ2H
0006  FDEØ  86 Ø4              LDA A #Ø4H
0007  FDE2  B7 9FØ3           STA A 9FØ3H
0008  FDE5  86 Ø7              LDA A #Ø7H
0009  FDE7  B7 9FØ2           STA A 9FØ2H
0010  FDEA  86 Ø4              LDA A #Ø4H
0011  FDEC  B7 9FØ1           STA A 9FØ1H
0012  FDEF  B6 9FØØ           LDA A 9FØØH
```

Initialisation du pointeur de la zone mémoire de sauvegarde du nom, et des adresses de début et de fin, de la file.

```
0013  FDF2  FE FDC1           LDX ØFDC1H   - Recherche et lecture
0014  FDF5  8C FD93           CPX #ØFD93H    du nom de la file et
0015  FDF8  27 1D             BEQ FILE       des adresses.
0016  FDFA  8C FD9D           CPX #ØFD9DH
0017  FDFD  27 18             BEQ FILE
0018  FDFF  8C FDA7           CPX #ØFDA7H
0019  FEØ2  27 13             BEQ FILE
0020  FEØ4  8C FDB1           CPX #ØFDB1H
0021  FEØ7  27 ØE             BEQ FILE
0022  FEØ9  CE FD93           LDX #FD93H
0023  FEØC  FF FDC1           STX ØFDC1H
0024  FEØF  6F ØØ    EFFAC   CLR Ø,X
0025  FE11  Ø8                INX
0026  FE12  8C FD BB          CPX #ØFDBBH  - Enregistrement est-il
0027  FE15  26 F8             BNE EFFAC      depassé ?
```

Recherche de la prochaine file.

```
0028  FE17  7F FDC9  FILE    CLR ØFDC9H
0029  FE1A  BD FF 2Ø  BCLF    JSR TIN
0030  FE1D  81 FF             CMP A #ØFFH
0031  FE1F  26 F6             BNE FILE
0032  FE21  7C FDC9           INC ØFDC9H
```

```
0033  FE24   F6 FDC9          LDA B ϕFDC9
0034  FE27   C1 FF            CMP B #ϕFFH
0035  FE29   26 EF            BNE BCLF
0036  FE2B   BD FF2ϕ  START   JSR TIN
0037  FE2E   81 53            CMP A #ϕ53H
0038  FE30   26 F9            BNE START
```

Lecture des adresses de début et de fin, et du Nom de la file.

```
0039  FE32   CE FDC5          LDX #ϕFDC5H
0040  FE35   7F FDD4          CLR ϕFDD4H
0041  FE38   BD FF2ϕ  BCLAD   JSR TIN
0042  FE3B   A7 ϕϕ            STA A ϕ,X
0043  FE3D   ϕ8               INX
0044  FE3E   8C FDCF          CPX #ϕFDCFH
0045  FE41   26 F5            BNE BCLAD
0046
```

Reconnaissance de l'enregistrement.

```
0046  FE43   B6 FDC9          LDA A ϕFDC9H   - Reconnaissance
0047  FE46   B1 FD BB         CMP A ϕFDBBH     de l'enregistrement.
0048  FE49   22 ϕ4            BHI STOP
0049  FE4B   27 ϕA            BEQ ZONE
0050  FE4D   20 C8            BRA FILE
```

Message "STOP" de dépassement de la file recherchée.

```
0051  FE4F   86 ϕ6    STOP    LDA A #ϕ6H     . Message STOP si
0052  FE51   B7 9Fϕ2          STA A 9Fϕ2H      Enregistrement
0053  FE54   7E FF F1         JMP FIN          recherchée est depassé.
```

Selection de la zone mémoire reservée au stockage des données lues sur K7.

```
0054  FE57   B6 FD C7  ZONE   LDA A ϕFDC7H   - Selection de la
0055  FE5A   F6 FD C8          LDA B ϕFDC8H     zone mémoire de
0056  FE5D   Fϕ FD C6          SUB B ϕFDC6H     stockage des donnée
0057  FE6ϕ   B2 FD C5          SBC A ϕFDC5H     lues.
0058  FE63   B7 FD C3          STA A ϕFDC3H
0059  FE66   F7 FD C4          STA B ϕFDC4H
0060  FE69   5F                CLR B
0061  FE6A   CE FD BD          LDX #ϕFDBDH
0062  FE6D   A6 ϕϕ      BCL   LDA A ϕ,X
0063  FE6F   81 FF            CMP A #ϕFFH
```

```
0064  FE71  26 09          BNE TSTAD
0065  FE73  5C             INC B
0066  FE74  08             INX
0067  FE75  8C FD C1       CPX #FDC1H
0068  FE78  26 F3          BNE BCL
0069  FE7A  20 30          BRA ORG1
0070  FE7C  C1 01   TSTAD  CMP B #01H
0071  FE7E  22 36          BHI ADFIN
0072  FE80  B6 FD BD       LDA A FDBDH
0073  FE83  F6 FD BE       LDA B 0FDBEH
0074  FE86  BD FF DA       JSR SPAD
0075  FE89  B7 FD C5       STA A 0FDC5H
0076  FE8C  F7 FD C6       STA B 0FDC6H
0077  FE8F  FB FD C4       ADD B 0FDC4H
0078  FE92  B9 FD C3       ADC A 0FDC3H
0079  FE95  25 41          BCS MES
0080  FE97  BD FF DA       JSR SPAD
0081  FE9A  B7 FD C7       STA A 0FDC7H
0082  FE9D  F7 FD C8       STA B 0FDC8H
0083  FEA0  FE FD BF       LDX 0FDBFH
0084  FEA3  27 07          BEQ ORG1
0085  FEA5  B1 FD BF       CMP A FDBFH
0086  FEA8  22 2E          BHI MES
0087  FEAA  27 03          BEQ TO
0088  FEAC  7E FE E0  ORG1  JMP ORG2
0089  FEAF  F1 FD C0  TO    CMP B 0FDC0H
0090  FEB2  22 24          BHI MES
0091  FEB4  26 F6          BRA ORG1
0092  FEB6  B6 FD BF  ADFIN LDA A 0FDBFH
0093  FEB9  F6 FD C0       LDA B 0FDC0H
0094  FEBC  B7 FD C7       STA A 0FDC7H
0095  FEBF  F7 FD C8       STA B 0FDC8H
0096  FEC2  BD FF DA       JSR SPAD
0097  FEC5  F0 FD C4       SUB B 0FDC4H
0098  FEC8  B2 FD C3       SBC A 0FDC3H
0099  FECB  25 0B          BCS MES
0100  FECD  BD FF DA       JSR SPAD
0101  FED0  B7 FD C5       STA A 0FDC5H
0102  FED3  F7 FD C6       STA B 0FDC6H
0103  FED6  20 D4          BRA ORG1
```

Message d'erreur d'adresse.

```
0104  F4D8   86 05     MES   LDA A #05H     - Message d'erreur
0105  FEDA   B7 9F 02        STA A 9F02H       d'adresse dans le
0106  FEDD   7E FF F1        JMP FIN           cas ou on ne peut pas
                                               stocker dans la zone
```

Lecture ou Verification des données.

```
0007  FEE0   FE FD C5  ORG2  LDX FDC5H
0108  FEE3   BD FF 20  LOPDA JSR TIN
0109  FEE6   7D FD BC        TST FDBCH
0110  FEE9   27 04           BEQ VERF
0111  FEEB   A7 00           STA A 0,X
0112  FEED   20 04           BRA LOPBO
0113  FEEF   A1 00     VERF  CMP A 0,X
0114  FEF1   26 13           BNE BAD
0115  FEF3   BC FDC7 LOPBO  CPX FDC7H
0116  FEF6   27 03           BEQ CHK
0117  FEF8   08              INX
0118  FEF9   20 E8           BRA LOPDA
0119  FEFB   BD FF 20  CHK   JSR TIN
0120  FEFE   7D FD D4        TST FDD4H   - Test du checksum
0121  FE01   26 03           BNE BAD       detecteur d'erreur
0122  FF03   7E FFBE         JMP F1
```

Message d'erreur de lecture ou de Verification des données.

```
0123  FF06   C6 03     BAD   LDA B #03H    - Message d'erreur
0124  FF08   F7 9F 02        STA B 9F02H
0125  FF0B   7E FF F1        JMP FIN
```

Routine de localisation d'un front montant ou descendant et determination de la longueur en cycles (période).

```
0126  FF0E   B6 05     FEDGE LDA A #05H
0127  FF10   F6 9F00          LDA B 9F00H
0128  FF13   01              NOP
0129  FF14   4C       LOOPF  INC A
0130  FF15   F6 9F01          LDA B 9F01H
0131  FF18   2A FF14          BPL LOOPF
0132  FF1B   C8 02           EOR B #02H
0133  FF1D   F7 9F01          STA B 9F01H
0134  FF20   39              RTS
```

Tin : Lecture d'un octet à partir de la K7.

```
0135  FF24  86 FF     TIN     LDA A #ØFFH
0136  FF23  B7 FDCF             STA A ØFDCFH
0137  FF26  7F FDDØ             CLR ØFDDØH
0138  FF29  7F FDD1             CLR ØFDD1H
0139  FF2C  7F FDD2             CLR ØFDD2H
0140  FF2F  8D DE               BSR FEDGE
0141  FF31  7D FF31            TST
0142  FF34  7D FF34            TST
0143  FF37  B7 FDD3           STA A ØFDD3H
0144  FF3A  8D D3             BSR FEDGE
0145  FF3C  81 1B             CMP A #1BH
0146  FF3E  2C F4             BGE NOTSH
0147  FF4Ø  B7 FDD3   LOOPS   STA A FDD3H
0148  FF43  8D CA             BSR FEDGE
0149  FF45  16                TAB
0150  FF46  FB FDD3           ADD B FDD3H
0151  FF49  C1 2B             CMP B #2BH
0152  FF4B  2F F3             BLE LOOPS
0153  FF4C  7E FF4F           JMP P1
0154  FF4F  F6 9FØØ    P1     LDA B 9FØØH
0155  FF52  8B Ø5             ADD A #Ø5H
0156  FF54  2Ø 1Ø             BRA SYN
0157  FF56  86 ØØ    LPOUT   LDA A #ØØH
0158  FF58  2Ø ØØ             BRA LPMID
0159  FF5A  7F FDDØ  LPMID   CLR ØFDDØH
0160  FF5D  B7 FDD1           STA A FDD1H
0161  FF6Ø  7F FDD2           CLR ØFDD2H
0162  FF63  86 ØA     LPIN    LDA A #ØAH
0163  FF65  4C        LOOP1   INC A
0164  FF66  F6 9FØ1   SYN     LDA B 9FØ1H
0165  FF69  2A FA             BPL LOOP1
0166  FF6B  F6 9FØØ           LDA B 9FØØH
0167  FF6E  7D FF6E           TST
0168  FF71  Ø1                NOP
0169  FF72  81 34             CMP A #34H
0170  FF74  2D Ø5             BLT SHRT
0171  FF76  7C FDD2           INC ØFDD2H
0172  FF79  2Ø Ø5             BRA WITH
0173  FF7B  7A FDD2   SHRT    DEC ØFDD2H
0174  FF7E  2Ø ØØ             BRA WITH
```

```
0175  FF8Ø    F6 FDDØ   WITH  LDA B ØFDDØH
0176  FF83    BB FDD1         ADD A ØFDD1H
0177  FF86    B7 FDD1         STA A ØFDD1H
0178  FF89    C9 ØØ           ADC B # ØØH
0179  FF8B    F7 FDDØ         STA B ØFDDØH
0180  FF8E    26 Ø3           BNE CHOKO
0181  FF9Ø    Ø1              NOP
0182  FF91    2Ø Ø4           BRA NOTO
0183  FF93    81 17   CHOKO   CMP A # 17H
0184  FF95    2C ØA           BGE BITOV
0185  FF97    C6 Ø5   NOTO    LDA B# Ø5H
0186  FF99    5A        P2    DEC B
0187  FF9A    2A FD           BPL P2
0188  FF9C    7E FF9F         JMP P3
0189  FF9F    2Ø C2     P3    BRA LPIN
```

Fin d'une durée d'un bit.

```
0190  FFA1    78 FDD2 BITOV   ASL ØFDD2H
0191  FFA4    76 FDCF         ROR ØFDCFH
0192  FFA7    24 Ø8           BCC TINDU
0193  FFA9    81 5D           CMP A # 5DH
0194  FFAB    2D A9           BLT LPOUT
0195  FFAD    8C 24           LDA A # 24H
0196  FFAF    2Ø A9           BRA LPMID
Ø1Ø
```

Addition du byte lue au checksum.

```
0197  FFB1    B6 FDCF TINDU   LDA A ØFDCFH
0198  FFB4    BB FDD4         ADD A ØFDD4H
0199  FFB7    B7 FDD4         STA A ØFDD4H
0200  FFBA    B6 FDCF         LDA A ØFDCFH
0201  FFBD    39              RTS
```

Sauvegarde du Nom, et des adresses de début et de fin, de la file lue.

```
0202  FFBE    CE FDC5   F1    LDX # ØFDC5H
0203  FFC1    A6 ØØ     SAVN  LDA A Ø,X
0204  FFC3    Ø8              INX
0205  FFC4    FF FDC3         STX Ø FDC3H
0206  FFC7    FE FDC1         LDX Ø FDC1H
0207  FFCA    A7 ØØ           STA A Ø,X
0208  FFCC    Ø8              INX
0209  FFCD    FF FDC1         STX Ø FDC1H
```

- Sauvegarde du nom et des adresses de la file lue.

```
0210  FFDØ   FE FD C3        LDX #ØFDC3H
0211  FFD3   ØC FDCF         CPX #ØFDCFH
0212  FFD6   26 E3           BNE SAVN
0213  FFD8   2Ø 17           BRA FIN
```

Contrôle des Adresses mémoires : BEGAD et ENDAD

```
0214  FFDA   81 FF     SPAD  CMP A #3FH   - Contrôle des
0215  FFDC   23 ØA           BLS RET       adresses
0216  FFDE   81 BF           CMP A #BFH    mémoires
0217  FFEØ   23 ØC           BLS MES1      BEGAD et ENDAD
0218  FFE2   81 FD           CMP A #FDH
0219  FFE4   22 Ø8           BHI MES1
0220  FFE6   27 Ø1           BEQ T
0221  FFE8   39       RET    RTS
0222  FFE9   C1 92    T      CMP B #92H
0223  FFEB   22 Ø1           BHI MES1
0224  FFED   39              RTS
0225  FFEE   7E FED8  MES1   JMP MES
0226  FF F1  3F       FIN    SWI
                             END
```

o

CONCLUSION

L'interface K7 que nous avons réalisé se distingue par la simplicité de son HARDWARE. Son logiciel lui confère une grande souplesse d'utilisation, car il est modifiable à tout moment. Tel qu'il a été réalisé il peut être adapté à n'importe quel système à base de microprocesseur de la famille 6800.

Par ailleurs notre étude est une première approche du problème de la recherche automatique de données sur K7, car les différentes commandes du magnétophone sont actuellement manuelles. Le fonctionnement serait entièrement automatique si l'on arrive à commander électriquement l'enclenchement et le déclenchement des touches de l'appareil enregistreur/lecteur de cassettes.

ANNEXE 1
Programmes de l'interface K7 du kit D2

PROGRAMME " PUNCH " :

```
PUNCH    LDA A # $ 0101 0001 .....8 bits avec2 bits d'arrêt RTS=1
         STA A ACIAS
         LDX # $ 03 FF
         BSR  PNLDR ..................Emettre des "1"

PUND 10  LDA.B  ENDAD + 1 ..........)  Calcul de la longueur
         SUB B  BEGAD + 1 ..........)  du bloc
         LDA A  ENDAD               )
         SBC A  BEGAD ..............)
         BEQ PUND 25.................  Difference < 256
         LDA B, $ FF.................  oui bloc = 256

PUND 25  LDA A    B ..............)  Enregistre "B" la longueur
         BSR OUTCH                )  du bloc et l'adresse
         PSH B                    )  de début
         TSX                      )
         BSR PUN                  )
         PUL A                    )
         INC A                    )
         STA A TEMP 2             )
         LDX #  BEGAD ..............)On enregistre l'adresse début

PUND 30  BSR PUN..................... Emettre les données
         DEC TEMPS 2
         BNE PUND 30 ................ a t'on fini le bloc ?
         STX BEGAD
         LDX # $ 00 19
         BSR PNLDR .................. oui, emettre 25 " 1 "
         LDX BEGAD
         DEX
         CPX ENDAD .................. a t on transmis toutes les données ?
         BNE PUND 10 ................ non, revenir au SP. PUND 10
         LDA A # 'G ................. enregistre un " G"
         RTS

OUTCH    PSH B.....................)  sous programme d'enregistrement d'un byte
OUTC 1   LDA B  ACIAS             )
         ASR B                    )
         ASR B                    )
         BCC OUT C.1              )
         STA A ACIAD              )
         PUL B                     )
         RTS.......................)

PUN      LDA A 0,X
         BSR OUTCH
         INX
         RTS

PNLDR    LDA A # $ FF .............)S.P d'enregistrement des 30 s de " 1 "
         BSR OUTCH                )
         DEX                      )
         BNE PNLDR                )
         RTS.......................)
```

PROGRAMME "LOAD" :

```
LOAD LDAA #% 0C01.0000 ..... Division par 1. Programmation
     STAA ACIAS              de L'ACIA.
BILD BSR  INCHR  ..........  Lire le 1e caractère.
     CMP A #'B   ..........  Est ce un "B"? Début de Bloc.
     BEQ RDBLCK  ..........  Si oui, chercher la long du Bloc.
     CMPA #'G    ..........  Si non, est-ce un G ?
     BNE BILD    ..........  Si non, chercher le caract.
                             suivant.
     RTS         ..........  Si oui, RTS
RDLBLCK BSR INCHR ........  Prendre le caract. suivant.
     TAB ..................  Le ranger dans Acc B.
     INCB .................  Incrementer B (compteur d'octets)
     BSR INCHR ............  Lire l'octet de poids fort de
     STAA BEGAD              Adr.Début.
     BSR INCHR ............  Lire l'octet de poids faible
     STAA BEGAD + 1          de Adr. Début.
     LDX BEGAD ............  Mettre adresse début dans Rg.Index
STBLCK BSR INCHR             Lire l'octet suivant
       S T A A 0,X .......   Le ranger dans adresse indiquée
       INX                   par Rg Index.
       DEC B .............   Décrémenter le compteur d'octets..
       BNE STBLK .........   et se brancher à BILD s'il n'est
       BRA BILD              pas égale à 0.
```

SOUS-PROGRAMME INCHR : (Recherche du caractère)

```
INCHR LDA A ACIAS        )   Donnée est elle prête ?
      ASR A              )

      BCC INCHR          )   si oui, aller chercher la donnée
      LDA A ACIAD        )   dans l'ACIA et la ranger dans
                         )   ACC A.

      RTS
```

ANNEXE 2

Le PIA 6821

Le registre de contrôle CRA (CRB) assurant la commande des 2 lignes CA_1 et CA_2 (CB_1 et CB_2), la gestion des intérruptions, ainsi que la sélection des 2 autres registres.

Les rôles des bits bo à b7 de CRA (CRB) sont :

b° : autorise ou masque (invalide) la demande d'intérruption arrivant sur la ligne CA_1 (CB_1) sous forme d'impulsions :

bo = o ---) interruptions invalidées
bo = 1 ---) interruptions autorisées

dans les 2 cas B7 du CRA (CRB) est positionné à 1.

b1 permet de définir sur quel front actif de l'impulsion de demande d'interruption appliquée sur CA.1 (CB.1)

b1 = o ----) transition négative
b1 = 1 ----) transition positive

b2 permet l'adressage des registres ORA (ORB) et DDRA (DDRB)

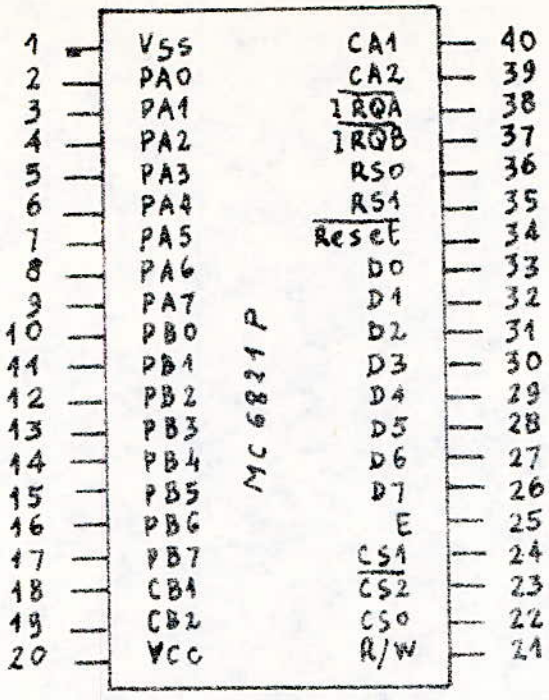
b2 = o --- séleetion de DDRA (DDRB)
b2 = 1 ---)selection de ORA (ORB)

b5 = 0 : CA2 (CB2) : programmée en entrée, dans ce cas b3, b4 et b6 ont respectivement le même rôle que bO, b1 et b7.
b5 = 1 : CA2 (CB2) : programmée en sortie.

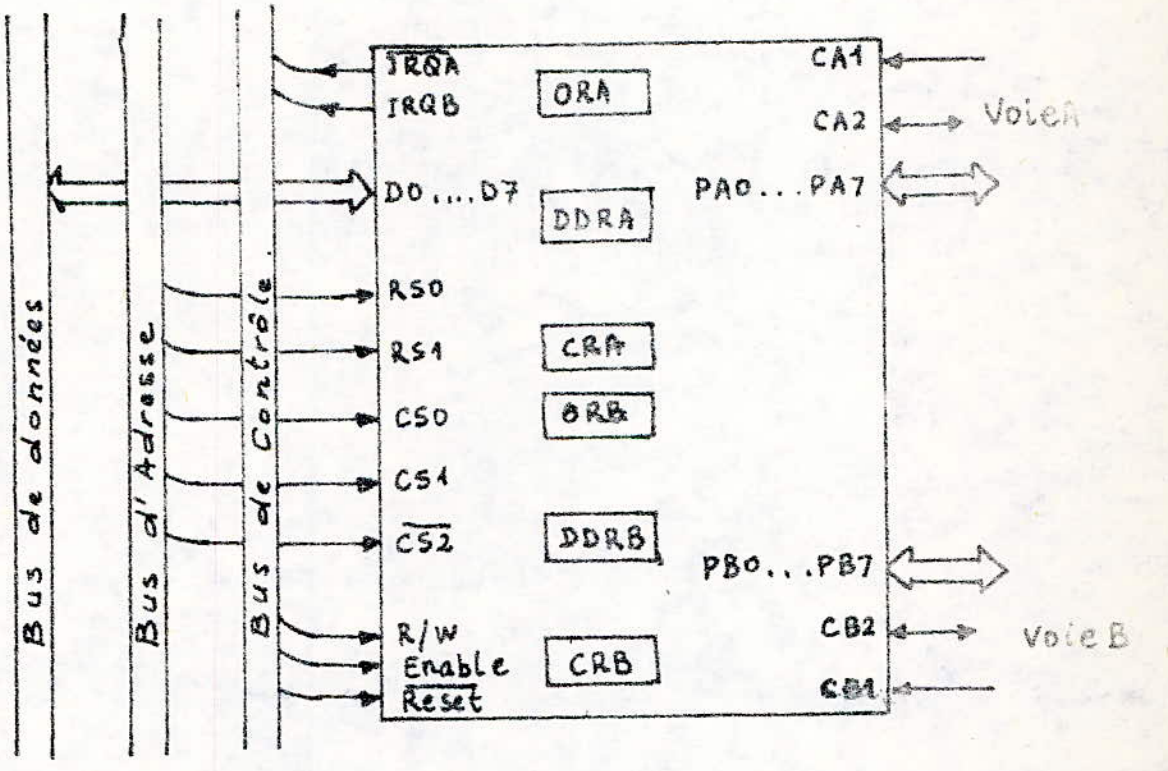
- Le registre de sens de transfert de données DDRA (DDRB) sert à fixer le sens de l'échange de données chaque bit de ce registre détermine le sens (entrée ou sortie) de la ligne correspondante de ORA (ORB) - l'écriture d'un 1 ou d'un 0 définit respectivement la ligne en sortie ou en entrée
- Le registre de données ORA (ORB) sert de " tampon " d'échange de données entre le MPU et le périphérique.

$\overline{CS2}$	CS1	CSO	RS1	RSO	b2 (CR)	Registre sélectionné
0	1	1	0	0	0	DDRA
0	1	1	0	0	1	ORA
0	1	1	0	1	X	CRA
0	1	1	1	0	1	DDRB
0	1	1	1	0	0	ORB
0	1	1	1	1	X	CRB

Annexe 2



BROCHAGE DU PIA
MC 6821 P



Microprocesseur

Organisation interne et externe du PIA 6821 P.

Périphériques

ANNEXE 3

Programme de l'interface K7

du KIT D5

TAPES

```
0778                    ********************************************************
0779                    *
0780                    * TAPES - SOFTWARE CASSETTE TAPE INTERFACE
0781                    *
0782                    ********************************************************
0783A F4D7 7D E43E  A TAPBEG TST     FNCFL     SEE IF PUNCH OR LOAD
0784A F4DA 27 06 F4E2        BEQ     PCH
0785A F4DC BD F69C  A LDTAP  JSR     LOAD      DO LOAD (OR VERF)
0786A F4DF 7E F024  A        JMP     PROMPT    WHEN DONE
0787                    *
0788A F4E2 CE F4EC  A PCH    LDX     #BEGEND   POINT AT BEGEND ROUTINE
0789A F4E5 FF E419  A        STX     MNPTR     ACTIVATE
   0A F4E8 86 BB    A        LDAA    #$BB
0791A F4EA 20 1D F509        BRA     CONOUT    DISPLAY BB IN LAST DISPLAYS
0792                    *
0793A F4EC 7D E41C  A BEGEND TST     KYFLG     SEE IF KEY PENDING
0794A F4EF 26 01 F4F2        BNE     ADNOW
0795A F4F1 39                RTS               ** RETURN ** NO KEY
0796                    *
0797A F4F2 BD F1EF  A ADNOW  JSR     RDKEY     READ & ACKNOWLEDGE KEY
0798A F4F5 2B 05 F4FC        BMI     FUNK      FUNCTION KEY
0799A F4F7 BD F1CC  A        JSR     ROLL4     ENTER NEW NUMBER
   ...A F4FA 20 16 F512      BRA     DYSOUT    CONVERT TO 7-SEG & LEAVE
0801                    *
0802A F4FC 86 EE    A FUNK   LDAA    #$EE
0803A F4FE B1 E42E  A        CMPA    HEXBUF+2  END ADDR DONE ?
0804A F501 27 12 F515        BEQ     DOPCH     GO DO PUNCH
0805A F503 FE E42C  A        LDX     HEXBUF    SAVE ENTERED ADDR
0806A F506 FF E460  A        STX     BEGAD
0807A F509 B7 E42E  A CONOUT STAA    HEXBUF+2  'EE' OR 'BB' TO LAST DISPLAYS
0808A F50C 7F E42C  A        CLR     HEXBUF    CLEAR FIRST FOUR NIBBLES
   9A F50F 7F E42D  A        CLR     HEXBUF+1
0810A F512 7E F120  A DYSOUT JMP     DYSCOD    CONV & RETURN
0811                    *
0812A F515 FE E42C  A DOPCH  LDX     HEXBUF    SAVE ENTERED ADDR
0813A F518 FF E462  A        STX     ENDAD
0814A F51B BD F630  A        JSR     PUNCH     PUNCH TAPE
0815A F51E 7E F024  A        JMP     PROMPT    WHEN DONE
0816                    *
```

```
0818                    ****************************************************************
0819                    * FEDGE - ROUTINE TO LOCATE AN EDGE (POS OR NEG)
0820                    *         AND DETRMINE DISTANCE TO IT (TIME)
0821                    *              EXECUTION TIME TUNED
0822                    ****************************************************************
0823                    *                                    8 FOR BSR
0824A F521 86 05     A FEDGE  LDAA   #5          2 START COUNT=FIXED (-1)
0825A F523 F6 E484   A        LDAB   PIADP       4 CLEAR INTERRUPT
0826A F526 01                 NOP                2 DELAY
0827A F527 4C                 INCA               2 DURATION COUNT IN A-REG
0828A F528 F6 E485   A        LDAB   PIACR       4 CHECK FOR EDGE FOUND
0829A F52B 2A FA F527         BPL    LOOPF       4 IF NOT;KEEP LOOKING
0830A F52D C8 02     A        EORB   #$02        2 INVERT EDGE SENSE CONTROL
0831A F52F F7 E485   A        STAB   PIACR       5 PIA LOOKS FOR OTHER EDGE
0832A F532 39                 RTS                5 **RETURN**


0834                    ****************************************************************
0835                    * TIN - READ 1 BYTE FROM TAPE
0836                    *       TIME TUNED
0837                    *
0838                    ****************************************************************
0839                    *                                    9 FOR JSR
0840A F533 86 FF     A TIN    LDAA   #$FF        2
0841A F535 B7 E459   A        STAA   BYTE        5 INITIALIZE BYTE
0842A F538 7F E45A   A        CLR    CYCNT       6
0843A F53B 7F E45B   A        CLR    CYCNT+1     6 INIT BIT-TIME COUNT
0844A F53E 7F E45C   A        CLR    GOOD1S      6 INIT LOGIC SENSE
0845A F541 8D DE F521         BSR    FEDGE       [22/21+-5] SYNC TO AN EDGE
0846A F543 7D F543   A        TST    *           6 DELAY
0847A F546 7D F546   A NOTSH  TST    *           6 DELAY
0848A F549 B7 E45D   A        STAA   OLD         5 "
0849A F54C 8D D3 F521         BSR    FEDGE       [22/21+-5] MEASURE TO NEXT EDGE
0850A F54E B1 1B     A        CMPA   #27         2 <1.5 SHORT HALF ?
0851A F550 2C F4 F546         BGE    NOTSH       4 MUST FIND SHORT FIRST
0852A F552 B7 E45D   A LOOPS  STAA   OLD         5 SAVE LAST COUNT
0853A F555 8D CA F521         BSR    FEDGE       [22/21+-5] MEASURE TO NEXT
0854A F557 16                 TAB                2 MAKE EXTRA COPY
0855A F558 FB E45D   A        ADDB   OLD         4 SUM OF LAST 2
0856A F55B C1 2B     A        CMPB   #43         2 >2.33 NOM. SHORTS?
0857A F55D 2F F3 F552         BLE    LOOPS       4 KEEP LOOKING FOR LONG
0858                    *
0859                    * EDGE SENSE SET-UP TO SENSE TRAILING EDGE OF CYCLES
0860                    * & YOU ARE IN THE MIDDLE OF THE FIRST LONG CYCLE
0861                    *
0862A F55F 7E F562   A        JMP    *+3         3 DELAY
0863A F562 F6 E484   A        LDAB   PIADP       4 CLEAR INTERRUPT FLAG
0864A F565 8B 05     A        ADDA   #5          2 COMPENSATE FOR PROCESSING
0865A F567 20 10 F579         BRA    SYNCIN      4 BRANCH INTO COUNT LOOP
0866A F569 86 00     A LPOUT  LDAA   #0          2 INIT BIT-TIME COUNT
0867A F56B 20 00 F56D         BRA    LPMID       4 DELAY
0868A F56D 7F E45A   A LPMID  CLR    CYCNT       6
0869A F570 B7 E45B   A        STAA   CYCNT+1     5 ESTABLISH BIT-TIME COUNT
0870A F573 7F E45C   A        CLR    GOOD1S      6 INIT LOGIC SENSE
0871A F576 86 0A     A LPIN   LDAA   #10         2 FIXED TIME (-1)= INIT COUNT
0872A F578 4C                 LOOP1  INCA               2 A-REG HOLDS DURATION COUNT
0873A F579 F6 E485   A SYNCIN LDAB   PIACR       4 EDGE YET?
```

```
F57C 2A FA F578          BPL     LOOP1     4 IF NOT;KEEP LOOKING
F57E F6 E484  A          LDAB    PIADP     4 CLEAR INTERRUPT FLAG
F581 7D F581  A          TST     *         6 DELAY TO MAKE PASS TIME...
F584 01                  NOP               2 EVEN MULTIPLE OF LOOP TIME
F585 81 34    A          CMPA    #52       2 <1.4 SHORT ?
F587 2D 05 F58E          BLT     SHRT      4
F589 7C E45C  A          INC     GOODIS    6 GOODIS POS MEANS 0
F58C 20 05 F593          BRA     WITHIN    4
F58E 7A E45C  A SHRT     DEC     GOODIS    6 GOODIS NEG MEANS 1
F591 20 00 F593          BRA     WITHIN    4 DELAY
F593 F6 E45A  A WITHIN   LDAB    CYCNT     4 HIGH BYTE
F596 BB E45B  A          ADDA    CYCNT+1   4 ADD CURRENT TO BIT-TIME COUNT
F599 B7 E45B  A          STAA    CYCNT+1   5 UPDATE
F59C C9 00    A          ADCB    #0        2 ADD IN CARRY
F59E F7 E45A  A          STAB    CYCNT     5 UPDATE HIGH BYTE
F5A1 26 03 F5A6          BNE     CHKOVR    4 IF CARRY; BIT MAY BE OVER
F5A3 01                  NOP               2 DELAY
F5A4 20 04 F5AA          BRA     NOTOVR    4 BIT NOT OVER
F5A6 81 17    A CHKOVR   CMPA    #23       2 (279-256)
F5A8 2C 0A F5B4          BGE     BITOVR    4 BIT-TIME EXPIRED
F5AA C6 05    A NOTOVR   LDAB    #5        [38] 2
F5AC 5A                  DECB              *    2
F5AD 2A FD F5AC          BPL     *-1       *    4
F5AF 7E F5B2  A          JMP     *+3       3
F5B2 20 C2 F576          BRA     LPIN      4

                         *
                         * END OF A BIT-TIME
                         *
F5B4 78 E45C  A BITOVR   ASL     GOODIS    6 LOGIC SENSE TO CARRY
F5B7 76 E459  A          ROR     BYTE      6 SHIFT NEW BIT INTO BYTE
F5BA 24 08 F5C4          BCC     TINDUN    4 DONE WHEN START FALLS OUT
F5BC 81 5D    A          CMPA    #93       2 >2.5 NOM. SHORTS ?
F5BE 2D A9 F569          BLT     LPOUT     4 NO; BIT-TIME STARTS AT 0
F5C0 86 24    A          LDAA    #36       2 YES; TRY MAINTAIN FRAMING
F5C2 20 A9 F56D          BRA     LPMID     4 NEXT BIT-TIME

                         *
                         * DATA BYTE READ; CLEAN-UP AND LEAVE
                         *
F5C4 B6 E459  A TINDUN   LDAA    BYTE      4 GET CURRENT BYTE
F5C7 BB E45E  A          ADDA    CHKSM     4 ADD TO CHECKSUM
F5CA B7 E45E  A          STAA    CHKSM     5 UPDATE
F5CD B6 E459  A          LDAA    BYTE      4 GET RECEIVED DATA IN A-REG
F5D0 39                  RTS               5 ** RETURN **
```

```
0918                    ********************************************************
0919                    * BIT1 - SEND A LOGIC 1 BIT-TIME
0920                    *        LESS 177 CLOCK CYCLES
0921                    *           TIME TUNED
0922                    ********************************************************
0923                    *                              8 FOR BSR
0924A F5D1 C6 0F    A BIT1   LDAB   #15          2 # SHORT H-CYCS (-1)
0925A F5D3 BD F5FF  A LOOPB1 JSR    INVRT        [20/5] TRANSMIT :DGE
0926A F5D6 86 18    A        LDAA   #24          [152] 2 DELAY
0927A F5D8 4A                DECA                " 2
0928A F5D9 2A FD F5D8        BPL    *-1          " 4
0929A F5DB 20 00 F5DD        BRA    *+2          4 DELAY
0930A F5DD 5A                DECB                2 1 LESS HALF CY :LE
0931A F5DE 26 F3 F5D3        BNE    LOOPB1       4 TILL 2ND LAST :DGE
0932A F5E0 BD F5FF  A        JSR    INVRT        [20/5] 15TH EDGE IN BIT-TIME
0933A F5E3 39                RTS                 5 **RETURN** 177 CYC TO NXT


0935                    ********************************************************
0936                    * BIT0 - SEND A LOGIC 0 BIT-TIME
0937                    *        LESS 177 CLOCK CYCLES
0938                    *           TIME TUNED
0939                    ********************************************************
0940                    *                              8 FOR BSR
 941A F5E4 C6 07    A BIT0   LDAB   #7           2 # LONG H-CYCS  -1)
 942A F5E6 BD F5FF  A LOOPB0 JSR    INVRT        [20/5] TRANSMIT :DGE
0943A F5E9 86 38    A        LDAA   #56          [344] 2 DELAY
0944A F5EB 4A                DECA                " 2
0945A F5EC 2A FD F5EB        BPL    *-1          " 4
0946A F5EE 01                NOP                 2 DELAY
0947A F5EF 5A                DECB                2 1 LESS TO GO
0948A F5F0 26 F4 F5E6        BNE    LOOPB0       4 TILL 2ND LAST :DGE
0949A F5F2 BD F5FF  A        JSR    INVRT        [20/5] 7TH EDGE :N BIT-TIME
0950A F5F5 86 1D    A        LDAA   #29          [182] 2 DELAY
0951A F5F7 4A                DECA                " 2
0952A F5F8 2A FD F5F7        BPL    *-1          " 4
0953A F5FA 7E F5FD  A        JMP    *+3          3 DELAY
0954A F5FD 01                NOP                 2 "
0955A F5FE 39                RTS                 5 **RETURN** 177 CYC TO NXT


0957                    ********************************************************
0958                    * INVRT - ROUTINE TO TRANSMIT A RISING
0959                    *         OR FALLING EDGE TO THE CASSETTE
0960                    *            TIME TUNED
0961                    ********************************************************
0962                    *                              9 FOR JSR
0963A F5FF 86 80    A INVRT  LDAA   #$80         2
 964A F601 B8 E486  A        EORA   PIADPB       4
0965A F604 B7 E486, A        STAA   PIADPB       5 INVERT OUTPUT
0966A F607 39                RTS                 5 ** RETURN **
```

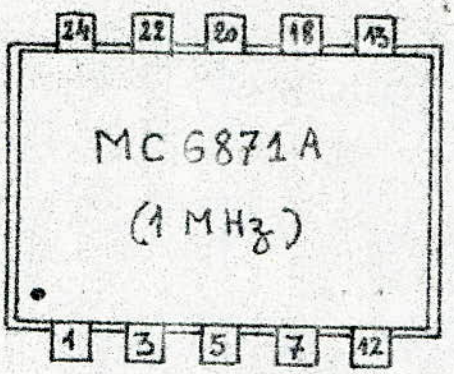
```
0968                    ************************************************************
0969                    * PNCHB - PUNCH 1 BYTE TO TAPE. INCLUDES
0970                    *         START BIT,DATA,AND ALL BUT LAST HALF-CYCLE
0971                    *         OF STOP BITS
0972                    *                      TIME TUNED
0973                    ************************************************************
0974                    *                              9 FOR JSR
0975A F608 B7 E459  A PNCHB   STAA   BYTE      5 SAVE BYTE TO PUNCH
0976A F60B 8D D7 F5E4         BSR    BIT0      [30/<177>] SEND START BIT
0977A F60D 86 09    A         LDAA   #9        2 # BITS IN BYTE (+2 STOP) (-1)
0978A F60F B7 E45F  A         STAA   NBITS     5 ESTABLISH BIT COUNT
0979A F612 7D F612  A         TST    *         6 DELAY
0980A F615 86 13    A LPPOUT  LDAA   #19        [122] 2 DELAY
0981A F617 4A                 DECA             " 2
0982A F618 2A FD F617         BPL    *-1       " 4
0983A F61A 0D                 SEC              2 SO LAST 2 BIT TIMES = 1'S
0984A F61B 76 E459  A         ROR    BYTE      6 LOGIC SENSE TO CARRY
0985A F61E 25 05 F625         BCS    DO1       4 IF LOGIC 1
0986A F620 8D C2 F5E4         BSR    BIT0      [30/<177>] XMIT A 0 BIT-TIME
0987A F622 7E F62A  A         JMP    ENDBIT    3
0988A F625 8D AA F5D1 DO1     BSR    BIT1      [30/<177>] XMIT A 1 BIT-TIME
0989A F627 7E F62A  A         JMP    ENDBIT    3 MATCHING DELAY
0990A F62A 7A E45F  A ENDBIT  DEC    NBITS     6 1 LESS BIT-TIME TO GO
0991A F62D 2A E6 F615         BPL    LPPOUT    4 CONTINUE FOR BYTE+STOP BITS
0992A F62F 39                 RTS              5 ** RETURN ** 159 CYC TO NXT
```

```
0994                    ************************************************************
0995                    * PUNCH - FORMAT AND PUNCH A CASSETTE DATA FILE
0996                    *         INCLUDING LEADER AND CHECKSUM
0997                    *            EXECUTION TIME TUNED
0998                    ************************************************************
0999                    *                                  9 FOR JSR
1000A F630 CE 0348  A PUNCH  LDX    #840        3 COUNT FOR 30-SEC LEADER
1001A F633 86 FF    A LLOOP  LDAA   #$FF        2 LEADER CHARACTER
1002A F635 C6 10    A        LDAB   #16         [104] 2 DELAY
1003A F637 5A                DECB               " 2
1004A F638 2A FD F637        BPL    *-1         " 4
1005A F63A BD F608  A        JSR    PNCHB       [44/<159>] PUNCH A LEADER CHAR
1006A F63D 09                DEX                4
1007A F63E 26 F3 F633        BNE    LLOOP       4 CONTINUE FOR 30-SEC
1008                    *
1009                    * LEADER FINISHED
1010                    *
1011A F640 86 53    A        LDAA   #'S         2 BLOCH START CHAR
1012A F642 C6 10    A        LDAB   #16         [104] 2 DELAY
1013A F644 5A                DECB               " 2
1014A F645 2A FD F644        BPL    *-1         " 4
1015A F647 BD F608  A        JSR    PNCHB       [44/<159>] PUNCH START CHAR
1016A F64A 01                NOP                2 DELAY
1017A F64B 7F E45E  A        CLR    CHKSM       6 INITIALIZE CHECKSUM
1018A F64E CE E460  A        LDX    #BEGAD      3 POINT AT FIRST ADDR BYTEOLT
1019A F651 A6 00    A ADLOOP LDAA   0,X
1020A F653 16                TAB                2 EXTRA COPY
1021A F654 FB E45E  A        ADDB   CHKSM       4 ADDR IS PART OF CHECKSUM
1022A F657 F7 E45E  A        STAB   CHKSM       5 UPDATE
1023A F65A 01                NOP                2 DELAY
1024A F65B C6 0D    A        LDAB   #13         [86] 2
1025A F65D 5A                DECB               " 2
1026A F65E 2A FD F65D        BPL    *-1         " 4
1027A F660 BD F608  A        JSR    PNCHB       [44/<159>] PUNCH ADDR BYTE
1028A F663 08                INX                4 ADV TO NXT ADDR BYTE
1029A F664 8C E464  A        CPX    #BEGAD+4    3 DONE YET ?
1030A F667 26 E8 F651        BNE    ADLOOP      4 CONTINUE FOR 4 ADDR CHARS
1031                    *
1032                    * READY TO PUNCH DATA
1033                    *
1034A F669 01                NOP                2 DELAY
1035A F66A 01                NOP                2 DELAY
1036A F66B FE E460  A        LDX    BEGAD       5 GET BEG ADDR OF DATA
1037A F66E A6 00    A DLOOP  LDAA   0,X         5 GET A DATA BYTE
1038A F670 16                TAB                2 EXTRA COPY
1039A F671 FB E45E  A        ADDB   CHKSM       4 ADD TO CHECKSUM
1040A F674 F7 E45E  A        STAB   CHKSM       5 UPDATE
1041A F677 F7 E45E  A        STAB   CHKSM       5 DELAY
1042A F67A C6 0B    A        LDAB   #11         [74] 2
1043A F67C 5A                DECB               " 2
1044A F67D 2A FD F67C        BPL    *-1         " 4
1045A F67F BD F608  A        JSR    PNCHB       [44/<159>] PUNCH DATA BYTE
1046A F682 7E F685  A        JMP    *+3         3 DELAY
1047A F685 BC E462  A        CPX    ENDAD       5 SEE IF DONE
1048A F688 27 03 F68D        BEQ    DUNDAT      4 IF FINISHED
1049A F68A 08                INX                4 ELSE ADV TO NXT
1050A F68B 20 E1 F66E        BRA    DLOOP       4 AND CONTINUE LOOP
```

```
1052                        *
1053                        * READY TO PUNCH CHECKSUM
1054                        *
1055A F68D 70 E45E  A DUNDAT NEG     CHKSM     6 SUM INCL, CHECK WILL BE 0
1056A F690 B6 E45E  A        LDAA    CHKSM     4 PREPARE TO SEND
1057A F693 C6 14    A        LDAB    #20       [128] 2
1058A F695 5A                DECB              " 2
1059A F696 2A FD F695        BPL     *-1       " 4
1060A F698 BD F608  A        JSR     PNCHB     [44/<159>] PUNCH CHECKSUM
1061A F69B 39                RTS               5 ** RETURN **
```

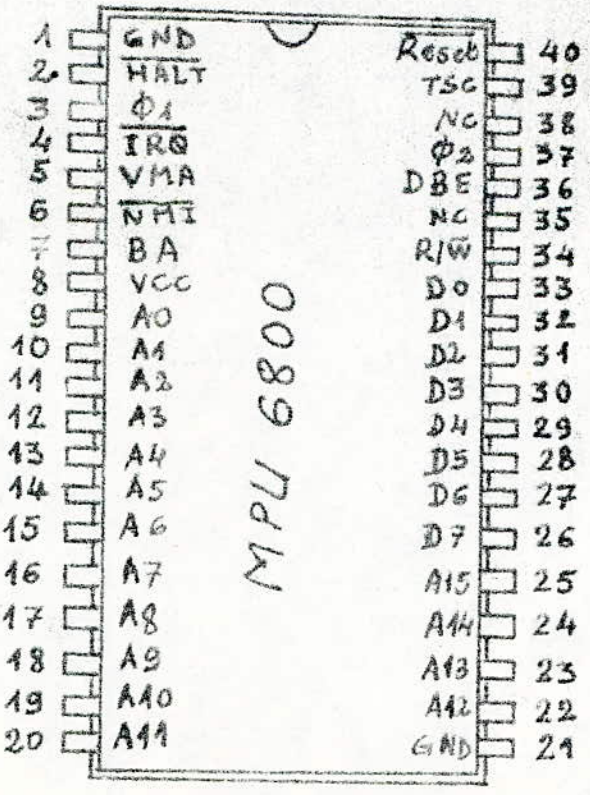
```
                        ********************************************************************
1063                    *
1064                    * LOAD - LOAD OR VERIFY A DATA FILE FROM
1065                    *        CASSETTE TAPE
1066                    *
1067                    ********************************************************************
1068                    *                                  9 FOR A JSR
1069
1070A F69C BD F533  A LOAD   JSR    TIN       [56/101+-5] READ A BYTE FROM TAPE
1071A F69F 81 53    A       CMPA   #'S       2 BLOCK START ?
1072A F6A1 26 F9 F69C       BNE    LOAD      4 NO; TRY AGAIN
1073                    *
1074                    * BLOCK START FOUND; NOW READ BEG & END ADDR'S
1075                    *
1076A F6A3 CE E460  A       LDX    #BEGAD    3 POINT AT ADDR AREA
1077A F6A6 7F E45E  A       CLR    CHKSM     6 INITIALIZE CHECKSUM
1078A F6A9 BD F533  A LOPAD  JSR    TIN       [56/101+-5] GET ADDR CHAR
1079A F6AC A7 00    A       STAA   0,X       6 STORE RECIEVED ADDR CHAR
1080A F6AE 08               INX              4 POINT AT NEXT ADDR LOC
1081A F6AF BC E464  A       CPX    #BEGAD+4  3 DONE GETTING ADDR'S ?
1082A F6B2 26 F5 F6A9       BNE    LOPAD     4 NO; CONTINUE
1083                    *
1084                    * READY TO READ DATA
1085                    *
1086A F6B4 FE E460  A       LDX    BEGAD     5 POINT TO WHERE DATA GOES
1087A F6B7 BD F533  A LOPDAT JSR    TIN       [56/101+-5] GET DATA FROM TAPE
1088A F6BA 7D E43E  A       TST    FNCFL     6 SEE IF LOAD OR VERF ?
1089A F6BD 27 04 F6C3       BEQ    VERF      4 IF NOT SET; IT'S VERF
1090A F6BF A7 00    A       STAA   0,X       6 IT'S LOAD SO STORE DATA
1091A F6C1 20 04 F6C7       BRA    LOPBOT    4 GO TO BOTTOM OF LOOP
1092A F6C3 A1 00    A VERF   CMPA   0,X       5 JUST COMPARE TO MEM
1093A F6C5 26 11 F6D8       BNE    BAD       4 IF NON-COMPARE; SIGNAL ERROR
1094A F6C7 BC E462  A LOPBOT CPX    ENDAD     5 DONE ?
1095A F6CA 27 03 F6CF       BEQ    CHKCHK    4 IF SO; CHECK CHECKSUM
1096A F6CC 08               INX              4 POINT AT NEXT DATA LOC
1097A F6CD 20 E8 F6B7       BRA    LOPDAT    4 AND CONTINUE LOAD/VERF
1098                    *
1099                    * DATA FINISHED... NOW CHECK CHECKSUM
1100                    *
1101A F6CF BD F533  A CHKCHK JSR    TIN       [56/105+-5] GET CHECKSUM
1102A F6D2 7D E45E  A       TST    CHKSM
1103A F6D5 26 01 F6D8       BNE    BAD       4 IF NOT ZERO; BAD CHECKSUM
1104A F6D7 39               RTS              5 ** RETURN **
1105                    *
1106A F6D8 FF E434  A BAD    STX    UX        6 SO USER CAN SEE END ADDR
1107A F6DB B7 E433  A       STAA   UA        5 SO USER CAN CHECK IT
1108A F6DE 7D E43E  A       TST    FNCFL     CHECK FOR ERROR OVERRIDE
1109A F6E1 2A 01 F6E4       BPL    STOP
1110A F6E3 39               RTS              ** RETURN ** NO MESSAGE
1111                    *
1112A F6E4 CE 7177  A STOP   LDX    #$7177    "FA"
1113A F6E7 FF E41D  A       STX    DISBUF
1114A F6EA CE 0638  A       LDX    #$0638    "IL"
1115A F6ED FF E41F  A       STX    DISBUF+2
1116A F6F0 7E F735  A       JMP    ALTBAD    PRINT "FAIL ??"
```

Annexe 4

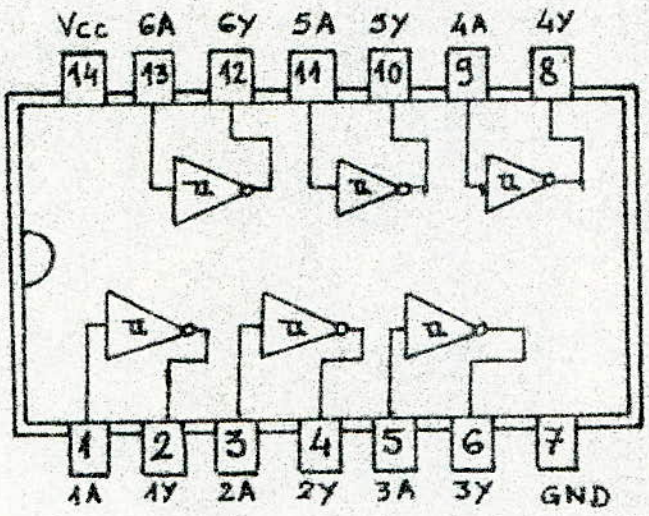


PIN	connection
1	GND
3	Memory clock
5	Φ_2 TTL
7	Vcc (+5Vdc)
12	Φ_2 NMOS
13	Φ_1 NMOS
18	GND
20	$\overline{HOLD1}$
22	Memory ready
24	$2 * f_c$

Brochage de l'Horloge MC 6871A.



Brochage du MPU 6800.

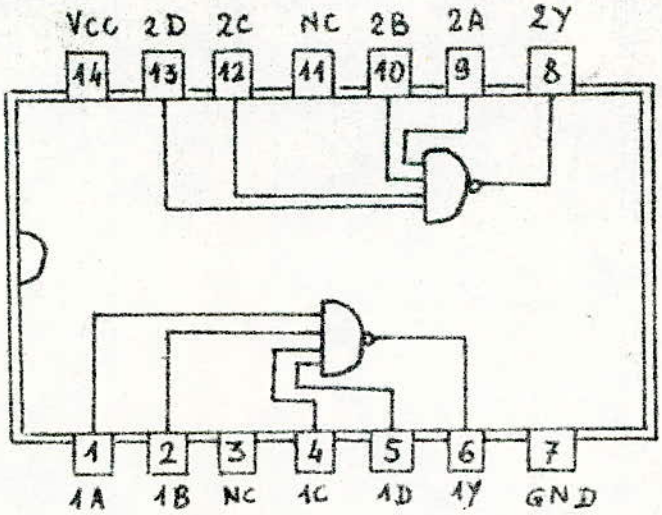


positive logic:

$Y = \bar{A}$

Brochage du SN 7414 N

HEX SCHMITT-TRIGGER INVERTERS.

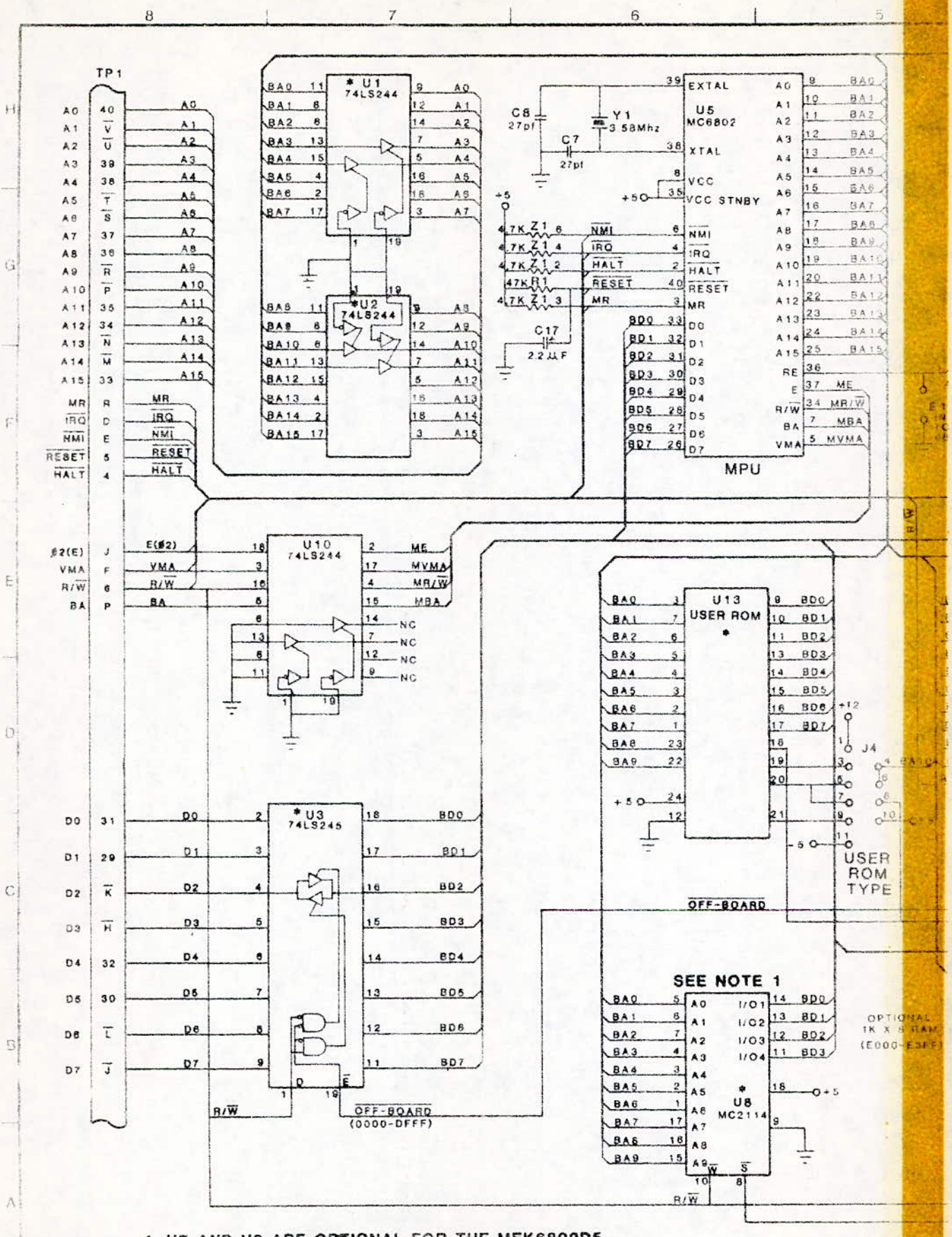


positive logic:

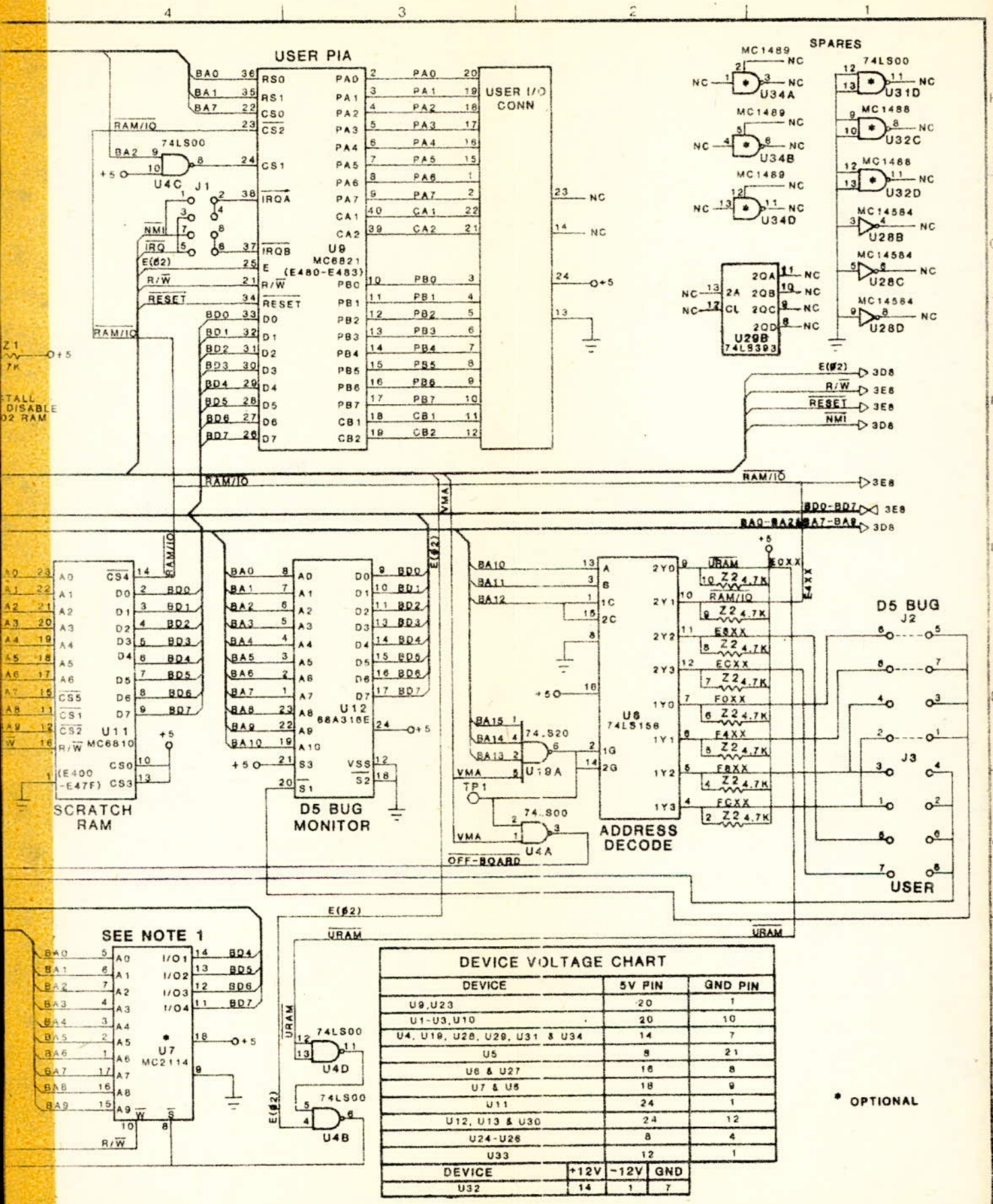
$Y = \overline{ABCD}$

Brochage du SN 7420 N

DUAL 4-INPUT POSITIVE NAND GATES.



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



DEVICE VOLTAGE CHART		
DEVICE	5V PIN	GND PIN
U9,U23	20	1
U1-U3,U10	20	10
U4, U19, U28, U29, U31 & U34	14	7
U5	8	21
U6 & U27	16	8
U7 & U8	18	9
U11	24	1
U12, U13 & U30	24	12
U24-U26	8	4
U33	12	1

DEVICE	+12V	-12V	GND
U32	14	1	7

* OPTIONAL

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

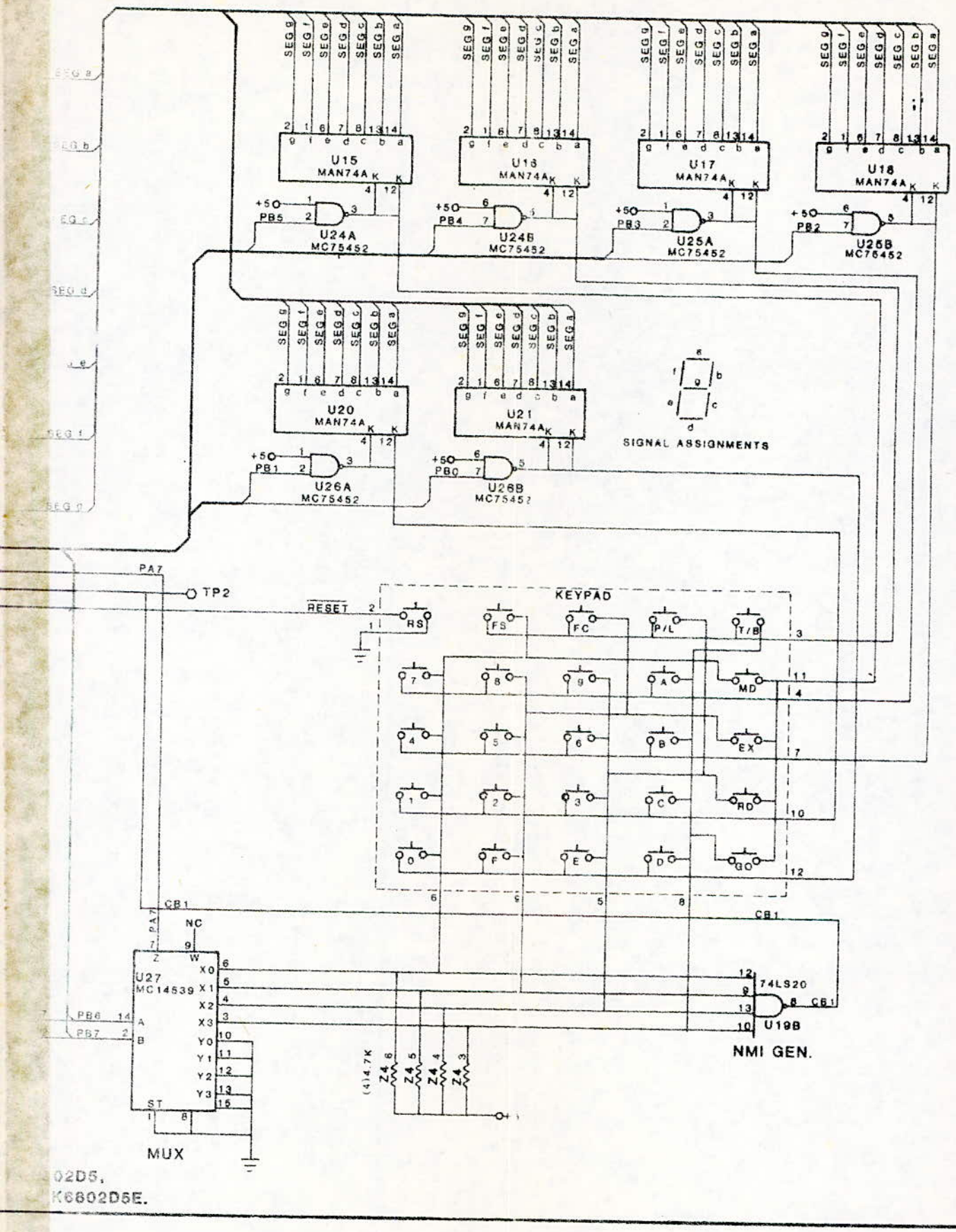


FIGURE 5.1 SCHEMATIC DIAGRAM
(SHEET 3 OF 3) 5-7/5-8

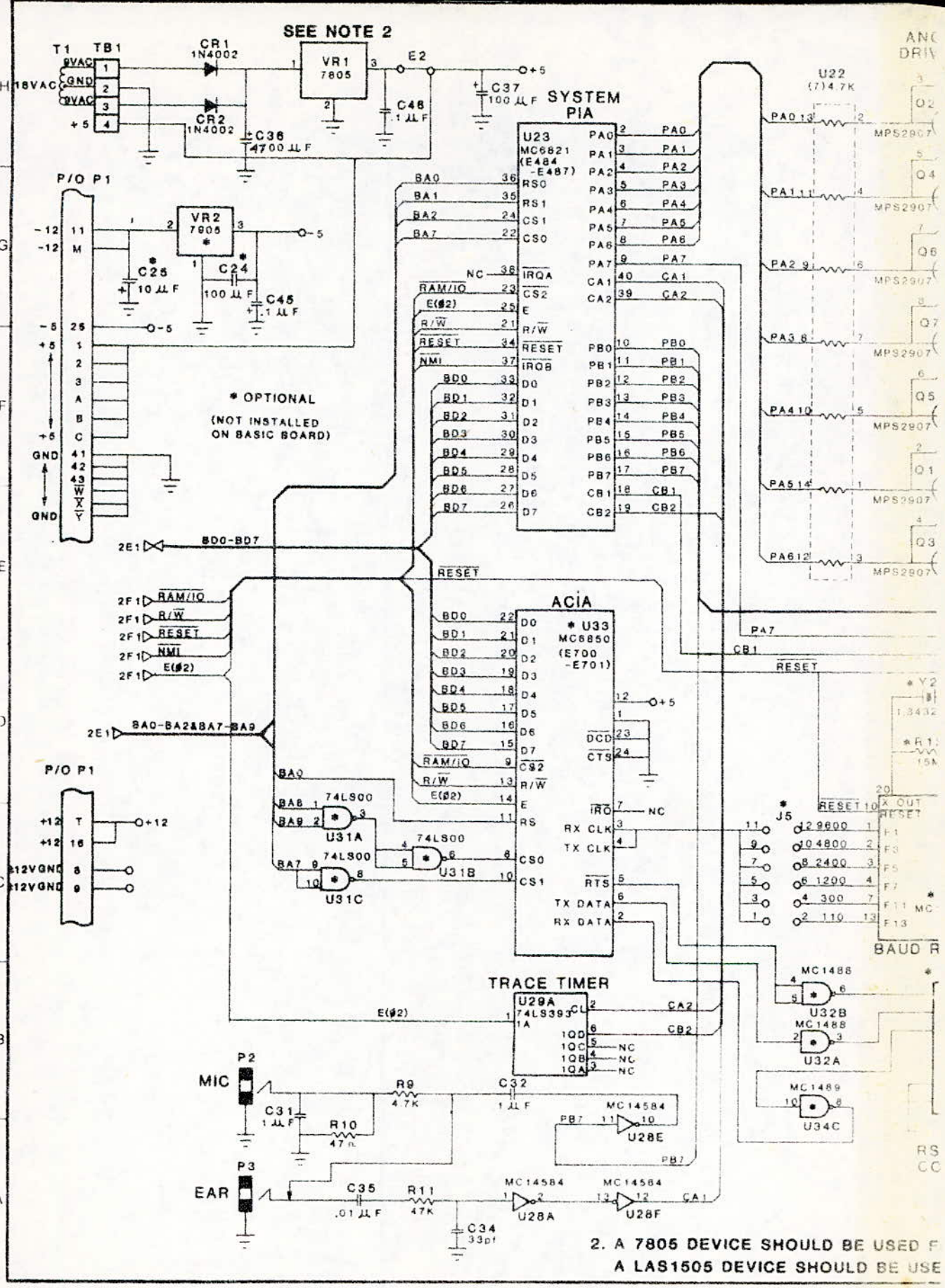

SEE NOTE 2
T1
TB1
18VAC
9VAC
GND
9VAC
+5
CR1
1N4002
CR2
1N4002
VR1
7805
C36
4700 µF
C46
.1 µF
E2
+5
C37
100 µF
SYSTEM
PIA
U23
MC6821
(E484
-E487)
P/O P1
VR2
7905
C25
10 µF
C24
100 µF
C45
.1 µF
-12
-12
-5
+5
+5
GND
GND
* OPTIONAL
(NOT INSTALLED
ON BASIC BOARD)
BA0
BA1
BA2
BA7
RAM/IO
E(Ø2)
R/W
RESET
NMI
IRQA
IRQB
CS2
CS1
CS0
BD0
BD1
BD2
BD3
BD4
BD5
BD6
BD7
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
CA1
CA2
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
CB1
CB2
2E1
BD0-BD7
2F1
RAM/IO
R/W
RESET
NMI
E(Ø2)
2E1
BA0-BA2&BA7-BA9
ACIA
* U33
MC6850
(E700
-E701)
DCD
CTS
IRQ
RX CLK
TX CLK
RTS
TX DATA
RX DATA
RS
CS0
CS1
74LS00
U31A
U31B
U31C
P/O P1
+12
+12
+12VGND
+12VGND
TRACE TIMER
U29A
74LS393
1QD
1QC
1QB
1QA
NC
NC
NC
MIC
P2
C31
1 µF
R10
47 Ω
R9
4.7K
C32
1 µF
EAR
P3
C35
.01 µF
R11
47K
C34
33pf
MC14584
U28E
U28A
U28F
U22
(7)4.7K
MPS2907
Q1
Q2
Q3
Q4
Q5
Q6
Q7
J5
BAUD R
MC1488
U32B
U32A
MC1489
U34C
2. A 7805 DEVICE SHOULD BE USED F
A LAS1505 DEVICE SHOULD BE USE

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

ADDRESSING MODES

COND. CODE REG.

LEGEND:

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

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

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

TABLE 4 – INDEX REGISTER AND STACK MANIPULATION INSTRUCTIONS

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

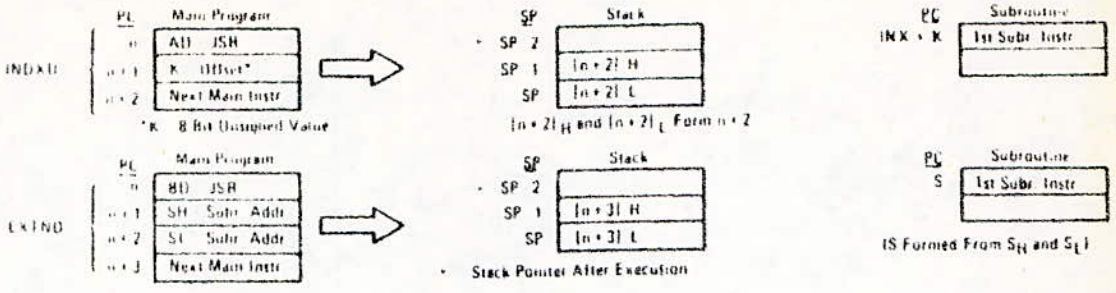
TABLE 5 – JUMP AND BRANCH INSTRUCTIONS

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

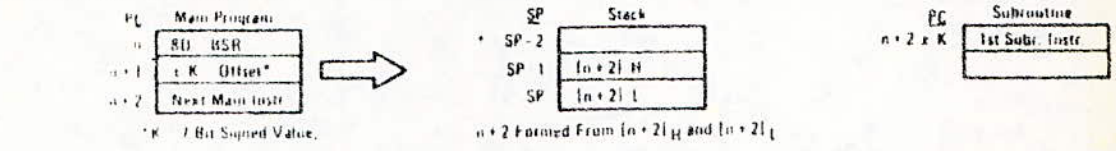
*WAI puts Address Bus, R/W, and Data Bus in the three state mode while VMA is held low

SPECIAL OPERATIONS

JSR, JUMP TO SUBROUTINE

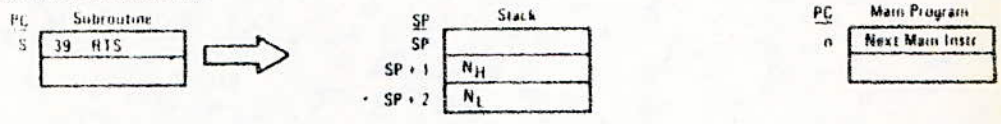


BSR, BRANCH TO SUBROUTINE



JMP, JUMP

RTS, RETURN FROM SUBROUTINE



RTI, RETURN FROM INTERRUPT

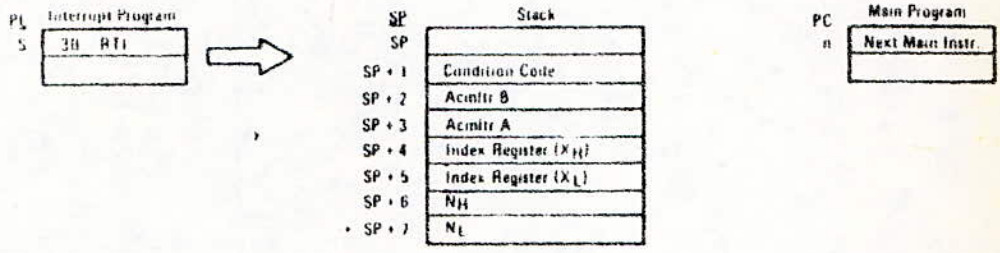


TABLE 6 – CONDITION CODE REGISTER MANIPULATION INSTRUCTIONS

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

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

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

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

1 - Projet de fin d'études :

- "Etude d'un micro-ordinateur basé autour du 6802 le KIT D5 de MOTOROLA" Janvier 1982
- "Mise en oeuvre de travaux pratiques sur le micro-ordinateur le KIT D5 de MOTOROLA" Juin 1982
- "Procédure de mise au point de systèmes à microprocesseurs sur le TEKTRONIX 8002A" Juin 1981.

2 - MEK6802D5E - Microcomputer evaluation board User's manuel.

3 - MEK6800D2 - Evaluation KIT II manual.

4 - D. GIROD et R. DUBOIS

"Au coeur des microprocesseurs"

Edition EYROLLES