Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie d'Alger

ECOLE NATIONALE POLYTECHNIQUE D'ALGER

Département d'Electronique

PROJET DE FIN D'ETUDES

Ingéniorat d'Etat en Electronique

CONCEPTION ET REALISATION D'UN LOGICIEL DE SORTIE DE RESULTATS D'UN SIMULATEUR NUMERIQUE

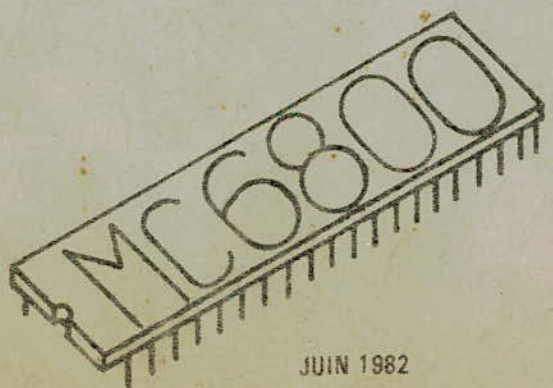
Proposé et suivi par

H.TEDJINI Docteur Ingénieur

Etudié par

Djouher AZZOUZ

Midodzi LASSEY



JUIN 1982

DEDICACES

A mon père, à ma mère:
pour leur confiance, leur courage et leur sacrifice.

A mes frères et soeurs:
pour leur aide.

A la mémoire de mon grand-père.

A mes grands parents.

A mon mari:
pour son aide et ses encouragements.

A ma tante M.
pour son aide.

A toute ma famille, mes beaux parents.

A tous mes amis(-ies).

DJOUHER.

Je dédie ce modeste travail,

à ma mère Hundevimé:
pour son sacrifice et sa confiance.

à la mémoire de mon père.

à Agnelé, Sewa et Agnoko:
pour leur soutien moral.

à mes frères et soeurs.

à Adjoa.

à ma collègue Djouher.

à tous mes amis(-ies).

MIDODZI.

-o-o- REMERCIEMENTS -o-o-
-o-o-o-o-o-o-

Nous remercions Monsieur B.SANSAL,responsable de la division "Simulation et Contrôle" au CSTN où notre projet de fin d'étude a pris forme.

Il nous est particulièrement agréable de remercier Monsieur H.TEDJINI pour l'interêt qu'il a prêté à notre travail,les conseils qu'ils nous a prodigués tout au long de son élaboration.

Nos remerciements vont à tout le personnel du centre de tirage en particulier Messieurs LAAZIB et ABDI pour nous avoir aidé avec beaucoup de compréhension.

Nous tenons à exprimer notre profonde gratitude à Monsieur M.HALIMI dont les suggestions et les conseils nous ont été d'un apport inestimable.

Que tous ceux qui ont contribué à notre formation, trouvent dans ce modeste mémoire l'expressinn de nos vifs remerciements.

Enfin nous remercions tous nos collègues,amis et parents qui n'ont cessé de nous offrir tout au long de notre travail leur support moral.

TABLE DES MATIERES:

.../....

INTRODUCTION.

La simulation numérique,dans le cadre de la réalisation des grands projets industriels,devient l'outil principal de l'ingénieur grâce au développement des calculateurs numéri--ques.

Le hardware(matériel)et le software(logiciel)de tels calcu--lateurs sont d'autant plus compliqués que le système,à simuler,est complexe.Ceci permet de distinguer deux étapes importantes dans la mise au point d'un calculateur numérique:

*Le choix de la structure qui résoud le problème hardware.

*L'élaboration du logiciel approprié.

La structure d'un système multitraitement à ligne omnibus, adoptée dans le projet de la simulation numérique,est la solution hardware.Il faut alors,realiser un logiciel en mesure de gérer ce matériel et de recueillir les résultats du calcu--lateur.

Compte tenu du grand nombre de paramètres d'une simulation, l'observation des résultats est difficile si bien que le logiciel permettant leur stockage sur des supports d'informa--tions est indispensable.Pour cela la technologie a mis à la disposition de ces systèmes,des supports d'informations tels que:

-table traçante.
-enregistreur potentiométrique.
-imprimante.
-bande magnétique.
-disque souple(floppy disk).
-écran de visualisation.
-perforateur de bande à papier.
-etc...

Dans ce projet qui consiste en une réalisation d'un logiciel de sortie de résultats d'un simulateur numérique,nous nous sommes intéressés au stockage des résultats sur trois supports

.../...

différents.Nous exposerons dans les chapitres qui suivent:

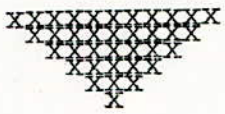
*Une définition du système sur lequel porte l'étude.

*Un programme permettant le stockage sur le disque souple.

*Un programme sortant les valeurs numériques,des paramètres simulés,sur imprimante rapide.

*Enfin un logiciel illustrant ces résultats au moyen d'un enregistreur potentiométrique.

La réalisation de ce logiciel nécessite une connaissance du système.Une étude préliminaire a été effectuée sur la carte esclave.Cette première étude sera présentée en annexe.



CHAPITRE I : Présentation du Système

A-Introduction à la simulation

B-Définition du système

1-Description du système

2-Fonction du système

C-Programmation

1-Logiciel

2-Langages de la programmation

a-binaire

b-héxadécimal

c-langage d'assemblage et assembleur

3-Les différents modes d'adressage

4-La structure de pile

A-Introduction à la Simulation:

C'est une tendance humaine bien naturelle que de tenter de percer un mystère nouveau en le ramenant à un autre problème plus simple ou du moins plus facile à résoudre.

Elle peut se définir comme une méthode de mesure, d'évaluation ou d'étude, qui consiste à substituer à la situation réelle un modèle ou simulateur répondant aux hypothèses et paramètres de la même façon c'est à dire obtenant les mêmes resultats que le processus physique, simulé, dans les mêmes conditions.

La science, depuis toujours et dans de nombreux domaines, a progressé grâce à l'utilisation des procédés de comparaison consistant à simuler la réalité en remplaçant l'objet de l'étude par un modèle plus accessible et plus maniable.

Le plus souvent la simulation se bornera à l'étude d'un nombre limité des aspects de l'objet; ceci permettra d'utiliser le modèle le plus simple possible.

Ce modèle peut être un dispositif particulier donc un matériel qui, le plus souvent mais pas obligatoirement, est relié à un calculateur lequel collecte les resultats, les exploite et, le cas échéant, les transforme en nouvelles instructions de commande.

Il peut être également abstrait; c'est alors un enchainement d'operations logiques et de choix effectués d'après certains critères. Cet enchainement résumé dans un organigramme peut faire l'objet d'un programme.

La simulation d'un reacteur nucléaire qui fait partie des travaux de recherche de la division X du CSTN, est une simulation dite en temps réel.

Ce genre de simulateur est celui qui nécessite l'échange de nombreuses informations sous forme de signaux électriques.

Il exige des moyens de conversion des codes et signaux utilisés par le centre de calcul à qui est connecté ce dispositif particulier(qu'est le simulateur)dúne part et le système extérieur qúest le dispositif d'autre part.

Aussi faut-il que ce système,ces échanges soient correctement gérés,ce qui fera appel à un moniteur de gestion.

Les étapes de l'étude et de la mise eh oeuvre d'un tel système seront:

-Evaluation des besoins et des contraintes du système.

-Etablissement du synoptique général.

Choix(logique cablée,microordinateur,miniordinateur...)

-Choix du microordinateur,du langage et du système de développement(dans le cas où le choix a porté sur ce dernier).

-Conception et réalisation du matériel et du logiciel (programmes de test et simulateurs)

-Mise au point.

B-Définition du système:

1-Description du système:

Le système que nous nous sommes proposés d'étudier est un ensemble de 33 processeurs physiques dont 32 sont des microordinateurs effectuant des operations à savoir la résolution de sous-systèmes dérivant d'un système numérique complet décrivant un processus physique.Le 33ème est conçu de la même façon (c'est un microordinateur) que les 32 autres mais son hardware est plus complexe.Il joue un rôle primordial dans le système.

Toutes ces unités appartenant à un même système et travaillant en parallèle doivent pouvoir se communiquer.Pour cela,des dispositifs de ces différents éléments se multiplient. Parmi ces dispositions on retrouve:

-Communication par réseau de selection matricielle.
-Communication par multibus.
-Communication par ligne omnibus.

Notre système multiprocesseur doit répondre auxqualités que requiert un système de simulation numérique en particulier la RAPIDITE.et donc la fiabilité en temps réel.Cette qualité acquise on arrive à la notion de temps partagé si bien que plusieurs processeurs utilisent le centre de calcul avec une impression de continuité dans les temps de réponse de l'organe de commande.Ceci rejoint aussi la notion de multiprogrammation.

Préoccupés donc par la rapidité,les avantages et inconvénients de chacun des dispositifs cités précédement;le système multiprocesseur dialoguant à l'aide d'une ligne omnibus a été adopté.(Voir schéma à la figure 1).

Les avantages que nous offre cette structure est sa simplicité aussi bien hardware que logicielle.Elle permet de controler,à tout moment de la simulation,chaque processeur P,pour modifier son comportement logiciel,à l'aide d'un seul processeur qui est le "coordinateur" ou(arbitre).Elle nous offre aussi des possibilités d'extension du nombre de processeurs.

Cependant,la gestion du système devient alors complexe au point de vue logiciel.
Rapelons brièvement la définition d'un processeur ou élément du système:
Ce n'est autre qu'un ensemble de composants réunis sur une carte pour assurer la résolution d'un sous-système de la simulation.Une carte minimum necessite:un microprocesseur, une horloge,des ROM,des RAM,coupleur d'E/S et d'autres éléments pour assurer le bon fonctionnement(buffers,bascules, peut-être des éléments discrets...)

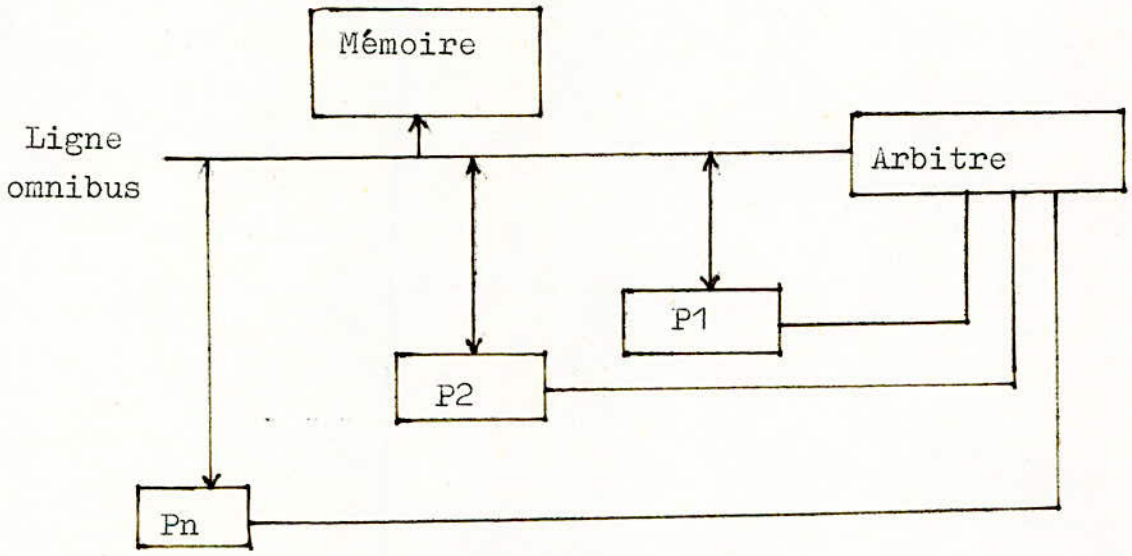


Figure 1 : schéma de communication par ligne OMNIBUS.

2-Fonction du système:

Chaque élément(P) du système a une tâche bien déterminée à exécuter.Conçu pour résoudre les problèmes issus d'un projet de simulation d'un processus continu dont le modèle est régi par 32 systèmes d'équations différencielles, chaque processeurs a besoin des paramètres et des conditions initiales qui se rapportent et qui sont nécessaires à son calcul.Ce n'est qu'après communication de ces paramètres et conditions initiales au processeur que celui-ci est prêt à calculer.

.../...

D'après la structure adoptée,cette distribution de paramètre ne se fait que par l'arbitre pour le processeur à travers la memoire(qui leur est commune).

Il est évident qu'avant le lancement de la simulation, tous les processeurs sont "prêts à calculer".Ces éléments P,pour accéder à la mémoire commune, suivent un ordre de priorité géré par l'arbitre;si bien qu'un P qui termine sa résolutionet qui ne devrait pas rester inactif en attente d'un sifflet général du démarrage d'un nouveau calcul, avertit l'arbitre en lui générant une demande d'accès à la mémoire.Ceci pour optimiser le rendement en temps. L'acquittement lui parvient d'autant plus rapidement qu'il est prioritaire(par rapport aux autres P).L'acquittement reçu,le processeur accède à la MC et dépose son résultat dans une zone réservée à cet effet.

Dans la MC chaque processeur dispose d'une table de laquelle il prend ses données(nécessaires à la suite de son calcul)et sont aussi portés les numéros des P qui ont besoin de son résultat pour la suite de leur résolution.

Une autre table est segmentée entre les 32 processeurs qui y déposent leur résultats.Cette table que nous appelons "BUVAR" sera développée dans les chapitres suivants.

Deux choix sont possibles pour la communication de ces résultats:

a)L'élément P accède à la MC,consulte sa table,dépose son résultat sur les tables des P qui en ont besoin, puis dans BUVAR,pour enfin prendre ses données et reprendre son activité.Ce qui ne peut se faire en un seul accès à la MC.

b)L'élément accède à la MC,dépose son résultat dans BUVAR, prend ses données(de sa table)et retourne à son activité.

Pour optimiser le temps le 2ème choix s'impose.

Comment:

-Il diminue le risque de panne du P à la suite de longs détours.

-Il reduit les accès du P à la MC.

Cependant ce choix n'enchante guère le coordinateur qui se

.../...

voit attribué en plus de ses propres fonctions:
-consulter la table de chaque élément Pi.
-distribuer le résultat de Pi, aux éléments intéressés, à partir de BUVAR(tout en pouvant être interrompu par une demande d'accès par exemple)
Rappelons que le coordinateur avait comme autre fonction:
-l'initialisation et le démarrage des éléments P après leur avoir communiquer les conditions de calcul.
- la gestion de l'accès à la zone commune selon un ordre de priorité.
-etc...

En résumé chaque P résoud un sous système.Le coordi-nateur, lui, veille au bon déroulement de l'exécution des données.

Les résultats des différents P sont maintenant dans la MC.On se pose la question suivante:Comment l'utilisateur va les exploiter?
Pour cela la technologie met à sa disposition différents coupleurs(parallèles, série)d'unité de disque, de visu etc...
Ces coupleurs permettent de sortir ces résultats sur:
-visu
-table traçante
-imprimante
-mémoire de masse (disque, dérouleur de bande...)
-perforateur de bande
-etc...
Tous ces organes de sortie permettent la colection des résultats en vue d'une interprétation .Il faut que l'Homme(l'utilisateur) devant son tableau de commande, derrière lequel se trouve le calculateur effectuant les calculs de la simulation numérique, ait l'impression de suivre exactement le vrai processus(alors qu'en réalité il suit ce processus simulé). Par conséquent cet utilisateur a besoin des résultats rapidement sous les yeux.Ceci fait que la suite des résultats sur l'imprimante, sur le disque etc...sont intéressants pour retracer l'aspect

du système (dans un moment ultérieur)alors que la table traçante qui collecte ces résultats sous forme de courbe, au fur et à mesure du calcul,présente un intérêt particulier pour l'utilisateur.En effet elle illustre directement l'aspect du système.

C-Programmation.

Le but de ce paragraphe n'est pas tant de faire une étude sur la programmation,mais,plutôt de rassembler un certain nombre de renseignements nécessaires pour la comp--rehension de cette étude.C'est pourquoi,nous parlons ici du logiciel,des differents types de langages en informati--que,des modes d'adressage etc...

1-Le logiciel:

Supposons quún processeur Pi demande à l'arbitre un transfert de k octets en memoire;la routine d'interruption doit remplir certaines conditions:
-mettre en place des paramètres nécessaires à ce transfert.
-envoyer une commande au périphérique pour initialiser le transfert.
-retourner au programme interrompu pour relancer son exécution
Pendant ce temps le P effectue son transfert à la fin duquel il avertit l'arbitre de la disponibilité de ses données .Ce dernier donne la main au programme les utilisant.
Le uP est capable de réaliser toutes ces fonctions mais une seul à la fois ,et piusqu'il n'a pas qu'une seule fonction, il faudra lui envoyer un mot binaire de 8 ou 16 bits;chacune des configurations de ce mot correspond à une opération donnée. Ce mot est appelé INSTRUCTION.
Le logiciel est donc une suite ordonnée d'instructions et de programmes relatifs au fonctionnement d'un système de traite--ment de l'information.C'est donc une logique programmée (SOFTWARE)contr--airement à la logique cablée qui relève du HARDWARE.

.../...

2- Langages de la programmation:

L'homme étant extérieur à la machine qu'est le uP et voulant communiquer avec ce dernier,doit utiliser un langage compris par la machine.Autrement dit,il doit communiquer par un programme. Ce programme peut être codé:
Ainsi les codes généralement utilisés sont à base binaire.Dans les uP on utilise les systèmes binaire pur,BCD,ASCII,héxadécimal etc... Il peut ne pas être codé c'est alors une suite de symboles mnémo-niques et il est appelé programme en langage assembleur ou langage d'assemblage.

a- Langage binaire:

Appelé aussi langage machine,C'est le seul que le uP comprend. C'est une suite de 1 et de 0 . A par le contenu
Ainsi pour charger l'accumulateur d'une mémoire on écrira:10110110; pour stocker le contenu de l'accumulateur B dans une mémoire on écrira:11110111;pour faire un saut à une soubroutine 10111101 etc... Situons-nous dans un programme plus complexe(comme un programme de simulation!)on se rend compte alors qu'aligner des 1 et des 0 devient une opération fastidieuse d'une part et risque de se tromper augmente avec la taille du programme d'autre part.

b- Langage héxadécimal:

Pour faciliter la tâche au programmeur,on s'est penché sur le langage héxadécimal qui est un condensé du langage binaire.Ainsi la taille des instructions diminue puisque l'écriture d'un octet en binaire se réduit à l'écriture de 2 caractères en héxadécimal soit 2 chiffres,1 chiffre+1 symbole,2 symboles.
Ainsi,le chargement de l'acc.A par le contenu de M devient: B6; le stockage de l'acc.B devient:F7;le saut à une soubroutine BD etc... Cependant le uP ne pouvant comprendre que le binaire,un traducteur héxadécimal/binaire s'avère nécessaire.Ce dernier est un programme intégré dans le moniteur, fait par le constructeur, et se trouve dans une memoire ROM.

c- Langage d'assemblage Et assembleur:

Malgré le langage héxadécimal les difficultés persistent quand le programme dépasse 100 mots.Etant une suite de symboles ou de chiffres,le programmeur discerne avec peine une instruction qu'il voudrait modifier en cas d'erreur.On fait appel au langage d'assemblage où l'instruction est une expression mnémonique(abreviation de certains mots)qui suggère à l'utilisateur le rôle de l'instruction Ainsi ADDA specifie une addition dans l'acc.A de son contenu avec le contenu de la mémoire en mode d'adressage étendu.L'expression qui suit cette instruction est l'adresse du mot à additionner.Un programme composé de telles instructions est appelé programme en langage d'assemblage et représente le "programme source".
Le µP ne comprend pas plus l'assembleur que l'héxadécimal si bien qu'un traducteur s'impose.L'assembleur est fourni par le constructeur ce traducteur est un programme ayant comme donnée le programme source à partir duquel il produit un programme en langage machine appelé"programme objet".L'assembleur est généralement stocké dans une disquette.Il est fort intéressant;il permet en fait à une 2ème personne de comprendre le programme écrit par le programmeur d'une part et au programmeur de se retrouver dans son propre travail d'autre part.

3-Les différents modes d'adressage:

Par un choix judicieux,parmi les 7 modes d'adressage qu'offre le MC6800,on peut améliorer le programme en réduisant:
-la longueur du programme.
-la capacité d'exécution.
-le temps d'exécution.
par conséquent,donner une grande souplesse à la programmation.
Dans le paragraphe qui suit on définit les différents modes d'adres--sage et on appèlera adresse effective,l'adresse réelle finale de la cellule memoire dans laquelle se trouve la donnée recherchée.

a-Adressage direct:

L'instruction contient au moins 2 octets;l'adresse de l'opéran ande est contenu dans le 2ème octet de l'instruction;le 1er octet est réservé à l'opération.C'est le mode d'adressage le plus utilisé

et le plus simple;il consiste à utiliser les adresses fournies sur le bus adresse pour accéder directement à des données contenues dans les positions correspondantes de la mémoire.

b- Adressage en étendu:

L'instruction contient 3 octets;l'adresse de l'operande est contenue dans les 2ème (poids fort)et le 3ème (poids faible) octet de l'instruction.Ce mode d'adressage permet le balayage de toutes les mémoires de 0000 à FFFF.Pour lire une adresse sur 16 bits on passera 2 tours de lecture en mémoire,c'est évidemment plus long que le mode d'adressage direct.On appèlera donc le mode étendu, un adressage direct mais où le mot adressage est codé sur une lon-gueur double,soit 2 octets.

c-Adressage indirect:

L'instruction se présente au moins sur 2 champs:le code opération et le champ adresse qui est logé dans l'opérande.La cellule désignée par le mot d'instruction ne contient pas la donnée utile, mais simplement une autre adresse qui sera l'adresse effective qui contient la donnée.Grâce à l'adressage indirect, il est facile de stocker un programme complet dnas une mémoire morte ROM.Les adresses utiles de soubroutines qui composent ce programme seront logées dans une RAM.Ainsi à partir de mêmes instructions de programme on peut traiter des données implantées dans toute la mémoire.Cet adressage indirect présente un inconvénient qui est celui du temps de rech-erche de l'information.
Pour obtenir l'adresse effective de l'opérande,il faut 2 cycles d-
de lecture de la mémoire;le temps de traitement est ainsi augmenté d'une durée de cycle;nommée phase d'indirection.

d-Adressage immédiat:

L'opérande est contenu dans le 2ème ou 2ème et 3ème octet de l'instruction,selon que l'on s'adresse aux accumulateurs ou au registre.Le champ adresse de l'operande contient la valeur de l'opérande il est appelé opérande immédiat;donc ce mode d'adressage ne permet de traiter que les opérandes dont les valeurs sont des constantes dans le programme.

e- Adressage indéxé:

L'instruction est composée de l'opération sur le 1er octet et de l'adresse sur le 2ème octet.On fait intervenir le registre d'index de 16 bits ;l'adresse effective de l'instruction est obtenue en ajoutant le contenu du registre d'index à la valeur du 2ème octet de l'instruction.Ce mode d'adressage est particulière--ment adapté au traitement de tableaux de données par incrémenta--tion ou décrémentation des instructions particulières.

f- Adressage en implicite:

L'instruction est composée de 2 ou 3 octets;l'opérande est indiqué par le code opération de l'instruction.Dans l'adressage implicite,on s'adresse implicitement à un registre donné.

g- Adressage inhérent:

C'est parfois l'instruction elle-même plus exactement son code opération,qui contient l'adresse où se trouve la donnée sur laquelle va porter l'operation.

4- La structure de pile:

Le MC6800 dispose d'une pile de registres volatils,qui permet de mémoriser les informations et de les restituer suivant le mode LIFO.La pile peut être constituée par une mémoire spécia--lisée,rapide ou être incluse dans la mémoire centrale de la machine électronique;dans ce cas sa capacité n'est pratiquement pas limitée(sauf par la capacité de la mémoire centrale elle-même).

a- Gestion de la pile :

Un registre,nommé pointeur de pile(SP)contient l'adresse de la 1ère position libre au sommet de la pile;il est modifié à chaque entrée ou sortie d'information,de sorte qu'il désigne toujours ce sommet de pile.
Si l'on considère une pile descendante en mémoire les adresses sont décroissantes vers le sommet de pile,le registre est décrémenté à chaque entrée d'information et incrémentée à chaque sortie.
Une pile consiste donc en une zone mémoire essentiellement

.../...

utilisée pour les stockages temporaires de données et qui est gérée par un mécanisme d'adressage particulier utilisant un registre pointeur de pile.Des instructions de stockage en pile(PUSH) ou de sortie de pile(PULL)permettent son utilisation;cette structure est aussi utilisée pour les appels de retour de sous-programmes.

b- Sous-programme:

Lors d'un appel de sous-programme,le problème essentiel réside dans la mémorisation de l'adresse de retour.Il y a 2 modes principaux de sauvegarde de cette adresse:
-Utilisation d'une position mémoire incluse dans le sous-programme.
-Utilisation d'une pile.

*Utilisation d'une position mémoire de sauvegarde:

L'exécution de l'instruction d'appel provoque le transfert de l'adresse de retour dans la 1ère position mémoire du sous-programme.Cette dernière est simplement le contenu du compteur de programme qui a été incrémenté enfin de décodage de l'instruction d'appel.Dans la 2ème phase,a lieu le transfert de l'adresse de début de sous-programme+1 dans le compteur de programme;en effet les instructions exécutables du sous-programme ne commencent qu'à sa 2ème position;la 1ère étant la mémoire de sauvegarde.

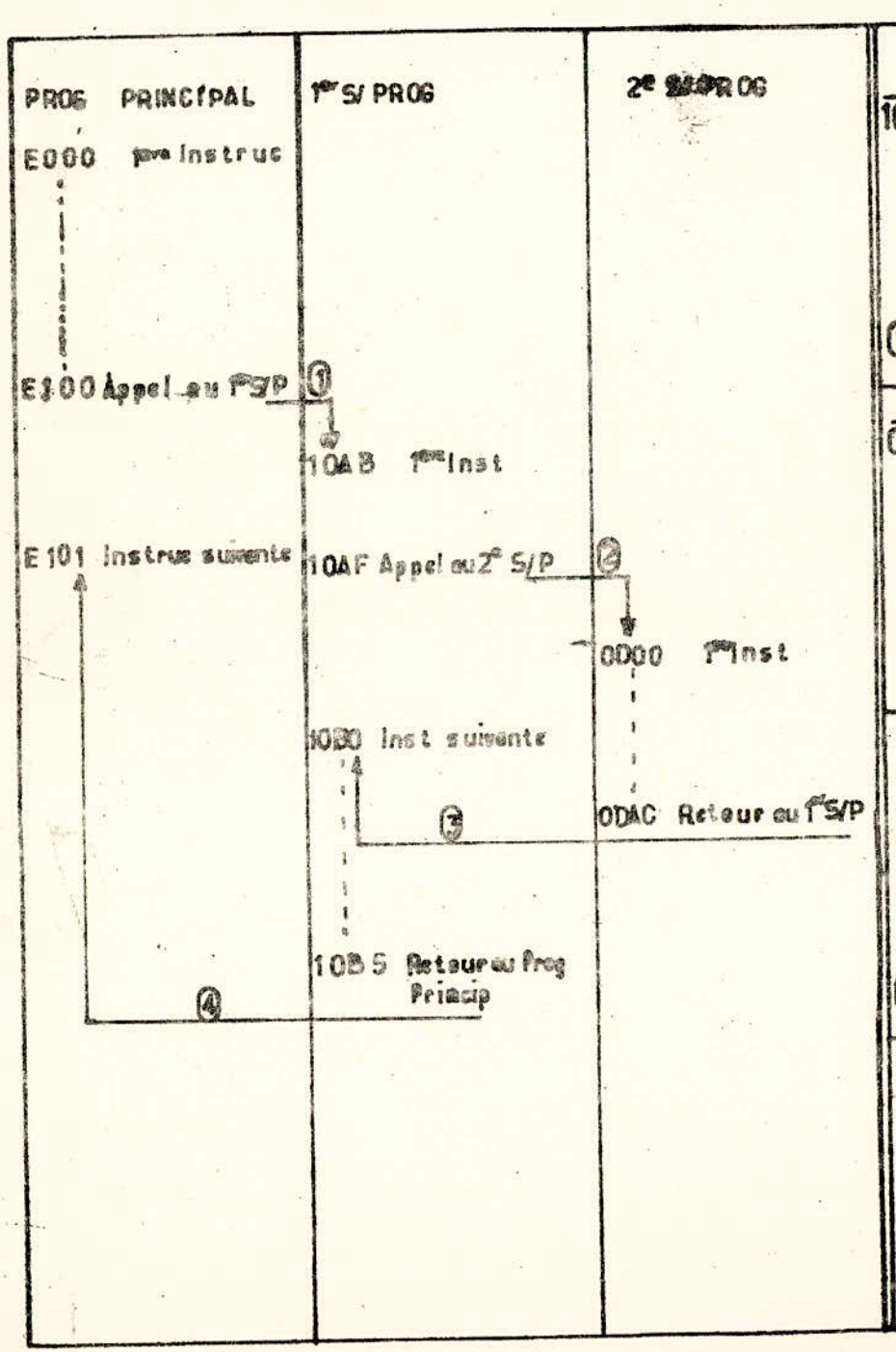
Cette structure ne permet pas la réentrance direct des sous-programmes,tout nouvel appel détruisant le contenu précédent de la mémoire de sauvegarde.De plus,ce mode de sauvegarde est mal adapté aux systèmes à microprocesseurs dont les programmes et sous-programmes sont fréquemment figés en ROM.

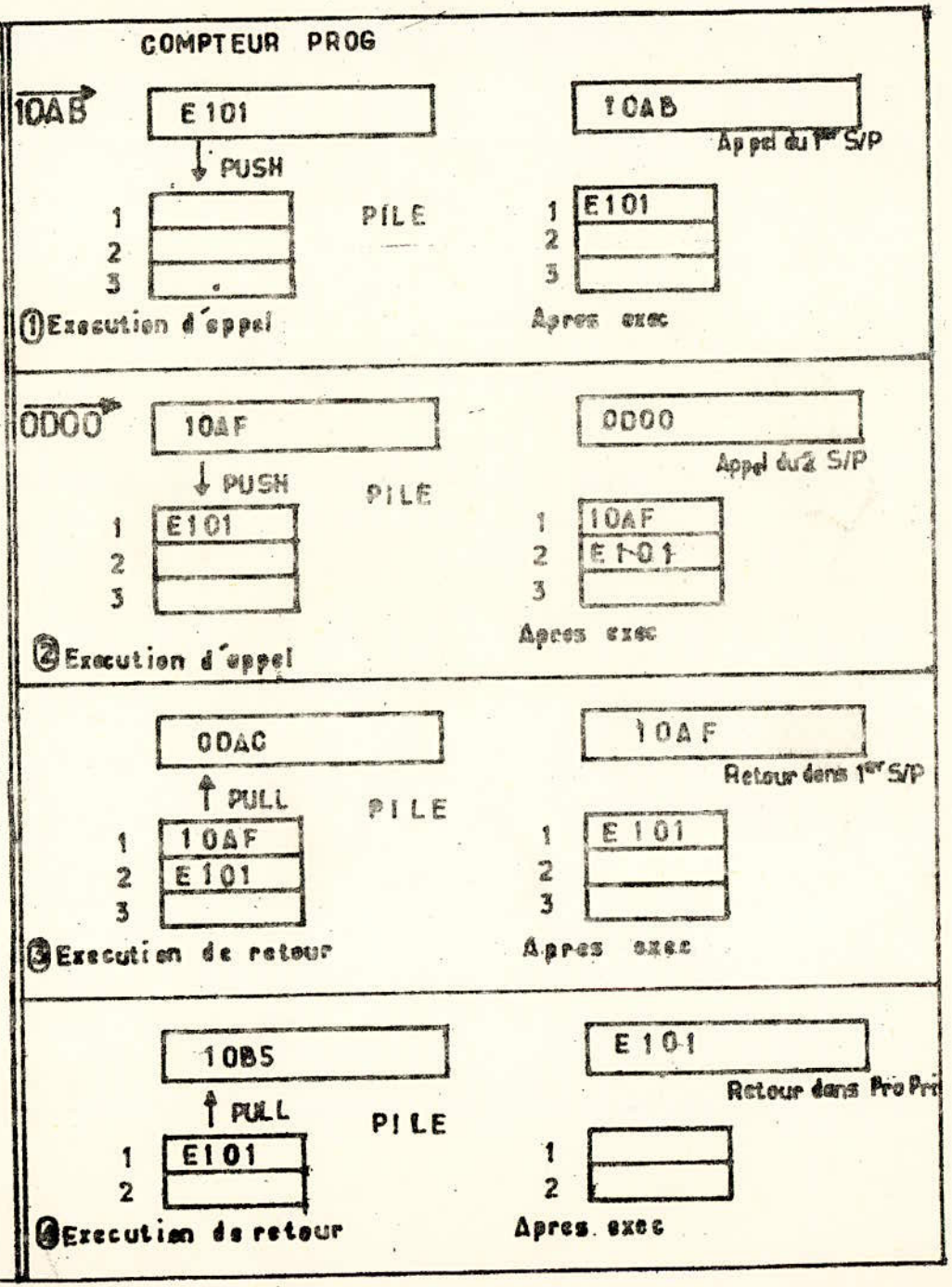
Le retour au sous-programme appelant s'effectue au moyen d'une rupture de séquence par adressage indirect de la mémoire de sauvegarde(ex.BRA:branchement au point de depart du programme).

* Sauvegarde en pile:

Une pile peut être utilisée pour sauvegarder l'adresse de retour d'un sous-programme.Lorsqu'un sous-programme est appelé, l'adresse de retour dans le programme appelant est stockée dans la pile,et restituée dans le compteur de programme à la fin du sous-programme.

EXEMPLE D'UTILISATION D'UNE PILE D'ADRESSE





CHAPITRE II

LOGICIEL DE SORTIE DE RESULTATS SUR DISQUE.

Introduction

A- Etude de l'unité de disque.

A-1- Le contrôleur de l'EXORdisk II .

A-2- Rôle de la mémoire ROM.

A-3- Description physique d'un disque souple.

A-4- Le formattage d'un disque souple.

A-5- Organisation de l'information sur le disque.

A-6- Opération d'écriture.

A-7- Opération de lecture.

A-8- Messages d'erreur.

B- Logiciel.

B-1- Présentation des résultats en mémoire commune.

B-2- Organisation des tables.

B-3- Organigrammes et programmes.

Introduction:

Nous avons vu que les systèmes multipliprocesseurs sont entourés de nombreux organes de sortie.Dans ce chapitre,nous présentons l'organisation des informations(résultats du système) sur le disque.
La disquette est apparue recemment mais elle se développe beau-coup car c'est une mémoire auxiliaire à accès direct et à grande fiabilité.

A- Unité du disque souple "Floppy disk":

L'unité d'entrainement du disque est un dispositif,avec mémoire rapide et à accès direct(aléatoire),utilisé pour le stoc-kage de l'information dans un système de traitement de données.
La mécanique de la disquette nécessite des courants assez impor-tants pour la commande des moteurs.Ces courants sont fournis par des circuits électroniques associés aux organes mécaniques;cet enaemble,circuits mecaniques et électronique,constitue un"DRIVE" "unité de disque souple"apte à recevoir des signaux du niveau TTL.
L'ensemble de ces signaux de commande est réalisé sur une carte électronique qui constitue la carte " CONTROLEUR " . La
La disquette dispose de 2 systèmes d'entrainement ou DRIVES numé-rotés 0 et 1.

A-1- Le contrôleur de l'EXORdisk II :

(Voir la fig.1)pour le schéma de la carte contrôleur.

Celui-ci fourni .,avec l'unité d'entrainement,par le constructeur est une carte sur laquelle on peut remarquer un certain nombre de circuits nécessaires au contrôle des différentes opérations de l'EXORdisk.
Ces circuits sont:
- Le circuit d'écriture de données.
- Le circuit de lecture de données.
- Le circuit de contrôle des buffers de bus.

.../...

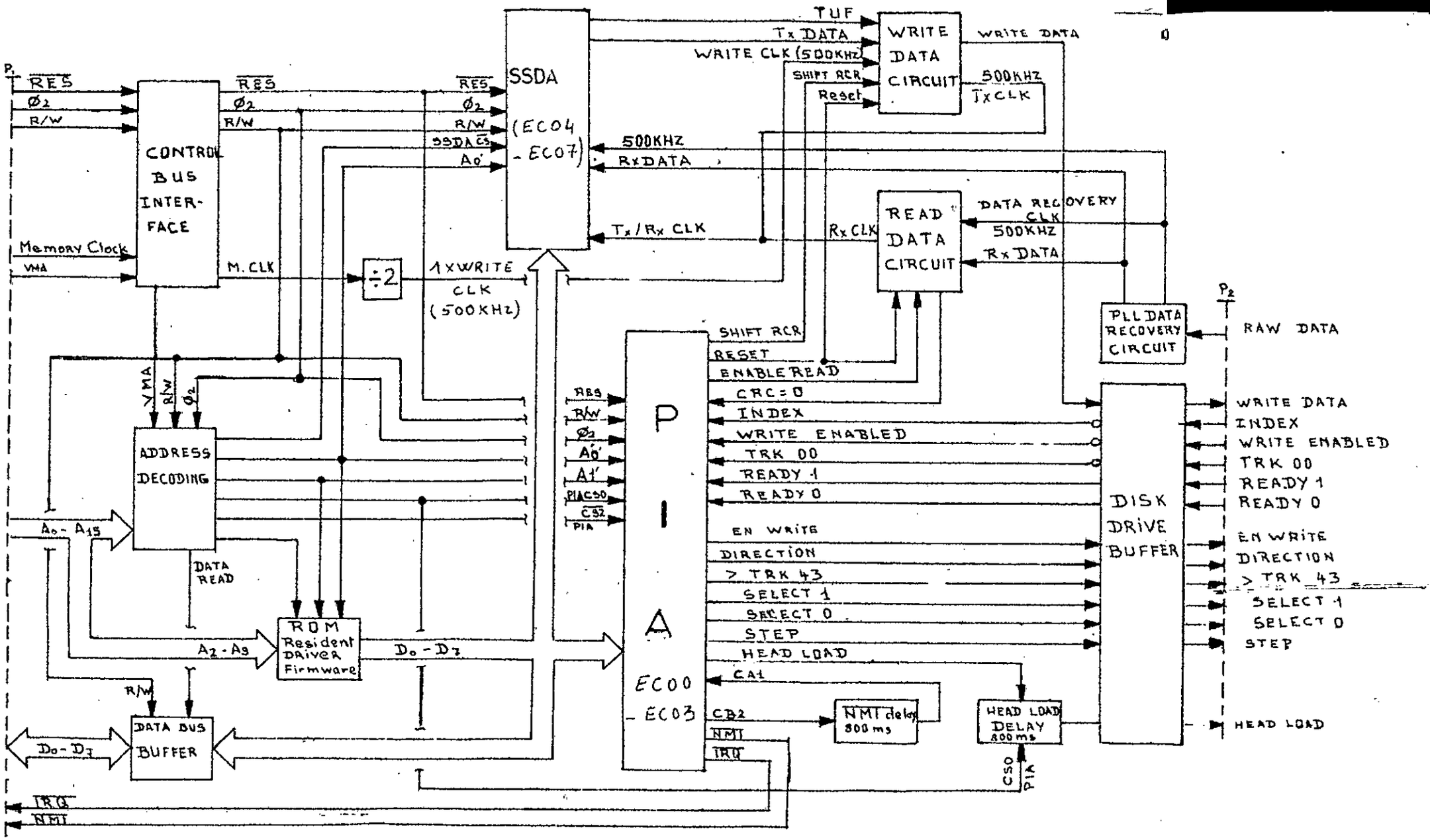


Fig.1: SCHEMA DU MODULE DE CONTROLE

- Les buffers des bus de données.
- Le buffer du bus de commande de l'unité de disque.
- ...

En plus de ces circuits,la carte contrôleur comporte une mémoire ROM,un adaptateur d'interface parallèle(PIA) et un adaptateur de synchronisation des données,série.Cette carte qui est un"module" constitue ce que nous pouvons appeler"interface de l'unité de disque souple".Comme cette dernière peut être utilisée dans de nombreux systèmes de traitement d'informations,il existe différents modes d'emploi du contrôleur.Aussi peut-on l'utiliser sans trop de précautions dans un système de micromodules de la famille parce que ce dernier,réalisé par l'utilisateur,tient compte du contrô--leur.

Cependant quand on l'utilise avec l'EXORciser,la procédure d'uti--lisation du contrôleur est la suivante.

* Mettre l'EXORciser hors tension.
* Installer le module d'interface EXORciser-EXORdisk II.
* Mettre le cable de connection mais en observant bien les pines (qui doivent se correspondre).
* Alimenter l'EXORciser.

A-2- Rôle de la mémoire ROM:

La ROM implantée sur la carte de contrôle,occupe les positions mémoires de E800 à EBFF.

Elle contient:

* Un programme fixe résident,utilisé pour contrôler les opérations au point de vue matériel telle que le contrôle des deux systè--mes d'entrainement.
* Un programme d'initialisation de tout le système qui n'est pas seulement utilisé avec le MDOS(Motorola Disk Operating ou Syst--ème d'opérations du disque de Motorola).
* Un programme standard de 6 paramètres mis à la disposition de l'utilisateur pour l'initialisation.
  Ces paramètres sont:

* CURDRV :permet de choisir l'un des 2 systèmes d'entrainement.
* STRSCT :utilisé pour choisir le secteur sur lequel porte l'opération.
* NUMSCT :contient le nombre de secteurs à utiliser pendant l'opé-ration y compris le secteur à moitié utilisé.
* LSNTLN :utilisé seulement pendant l'opération de lecture,pour indiquer le nombre d'octets à lire sur le dernier sect-eur.
* CURADR :renseigne le système sur l'adresse début de la zone des données sur lesquelles porte l'opération.Sa mise à jour après l'opération sur un secteur se fait automatiquement.
* FDSTAT :Ce paramètre donne l'état et permet à l'utilisateur de vérifier son programme.Lorsqu'il y a une fausse manoeuvre au niveau du programme ou du matériel il avertit,l'uti-lisateur,de l'erreur par un code qu'il visualise sur l'écran.Il existe 9 codes(qui seront développés dans le paragraphe A-8).

A-3- Description physique d'un disque souple:

La disquette,de la taille d'un disque audio 45 tours, est à base d'un disque de mylar flexible d'où son qualificatif de"DISQUE SOUPLE".Ce disque,recouvert d'une substance d'oxyde magnétique et à surfaces planes,tourne à vitesse constante.Il est protégé par une pochette en carton même en fonctionnement.Cet étui est revêtu à l'intérieur d'un enduit plastique ce qui réduit le frottement et lubrifie.
En effet,lorsqu'elle est introduite dans l'unité de lecture/ écriture encore appelé DRIVE,la disquette,entrainée par l'axe d'un moteur tourne en permanence à l'intérieur de sa pochette.

Comme un disque dur,un disque souple comporte des cercles concentriques appelés"PISTES".Une disquette normale en compte 77 sur une surface(SIMPLE FACE) ou sur chacune des faces(DOUBLE FACE) L'enregistrement de données sur un disque se fait en simple den-sité ou en double densité.De plus certaines disquettes munies alors de 2 têtes de lecture/écriture permettent d'enregistrer des données sur les 2 faces.

.../...

Les pistes sont numérotées de 00 à 76.La première (la piste 00), la plus externe,est la piste INDEX.

(Voir la fig.2 et la fig.3)

Comme l'indique la fig.2; on peut remarquer:

* Une ouverture ciculaire qui laisse apparaitre un trou dans le disque.C'est le trou d'INDEX qui signale le début de chaque piste.
* Une fente radiale qui permet à la tête de lecture/écriture d'accéder,à toutes les pistes du disque,sous l'action d'un méca--nisme spécial piloté par une logique externe incluse dans le contrôleur de l'unité d'entrainement du disque.
* Une encoche,au côté inférieur de la pochette,qui permet de sau--vegarder l'information inscrite sur la disquette.Le degagement de cette encoche laisse apparaitre un trou à la place.Ce trou, lors d'une tentative d'une opération d'écriture,inhibe le signal d'écriture et protège ainsi l'information déja enregis--trée.

  D'autre part à l'aide d'une étiquette autocolante,l'utilisateur peut identifier extérieurement son disque souple.

Tableau des caractéristiques des disquettes:

taille	8 pouces soit 20,32 cm
nombre de pistes	76 pistes + la piste d'index
nombre de secteurs/piste	26
nombre d'octets/secteur	128
densité	.3268 bits/pouce en simple densité. .6536 bits/pouce en double densité.
vitesse de rotation	360 tours/minute
capacité	253K octets

.../...

A-4- Le formatage d'un disque souple:

Chaque piste est divisée en un certain nombre de secteurs de longueur fixe.Suivant le formattage(opération logicielle qui permet de diviser une piste en secteurs) on trouve 26,15 ou 8 secteurs de longueur respectivement égale à 128,256 ou 512 octets. Dans notre cas nous avons un disque avec des pistes à 26 secteurs de 128 octets.
Comme pour un disque dur,des informations d'identification doivent être placées sur un disque souple pour repérer et reconnaitre les données.Seulement,cette identification n'est pas pareille à celle du disque dur;elle est programmable avec la donnée sur la disquette lors de l'enregistrement.

A-5- Organisation de l'information: (Voir la fig.4)

Le format physique des données enregistrées sur la disq--uette est une fonction du module de contrôle du disque souple; c'est une technique propre au constructeur de la disquette.On uti--lise généralement la modulation de fréquence.

Pendant l'enregistrement,le bit le plus significatif est le prem--ier transféré et le moins significatif le dernier transféré.De même pour la lecture qui se fait en mode FIFO.

Comme nous l'avons déja dit,le disque souple a 77 pistes(de 00 à 76)et chacune de ces pistes contient 26 secteurs d'informations.
Chaque secteur est précédé par un champ d'identification.
Chaque champ de données est séparé du champ adjacent par un nombre d'octets vides appelés"GAP".
Il existe 4 types de gaps:

- Gap d'identification: situé entre les champs d'idenfification et de données.
- Gap de données: situé entre le champ de données et le prochain champ d'identification.
- Gap de préindex: sépare le dernier champ de données sur la piste et la marque d'adresse d'index.

.../...

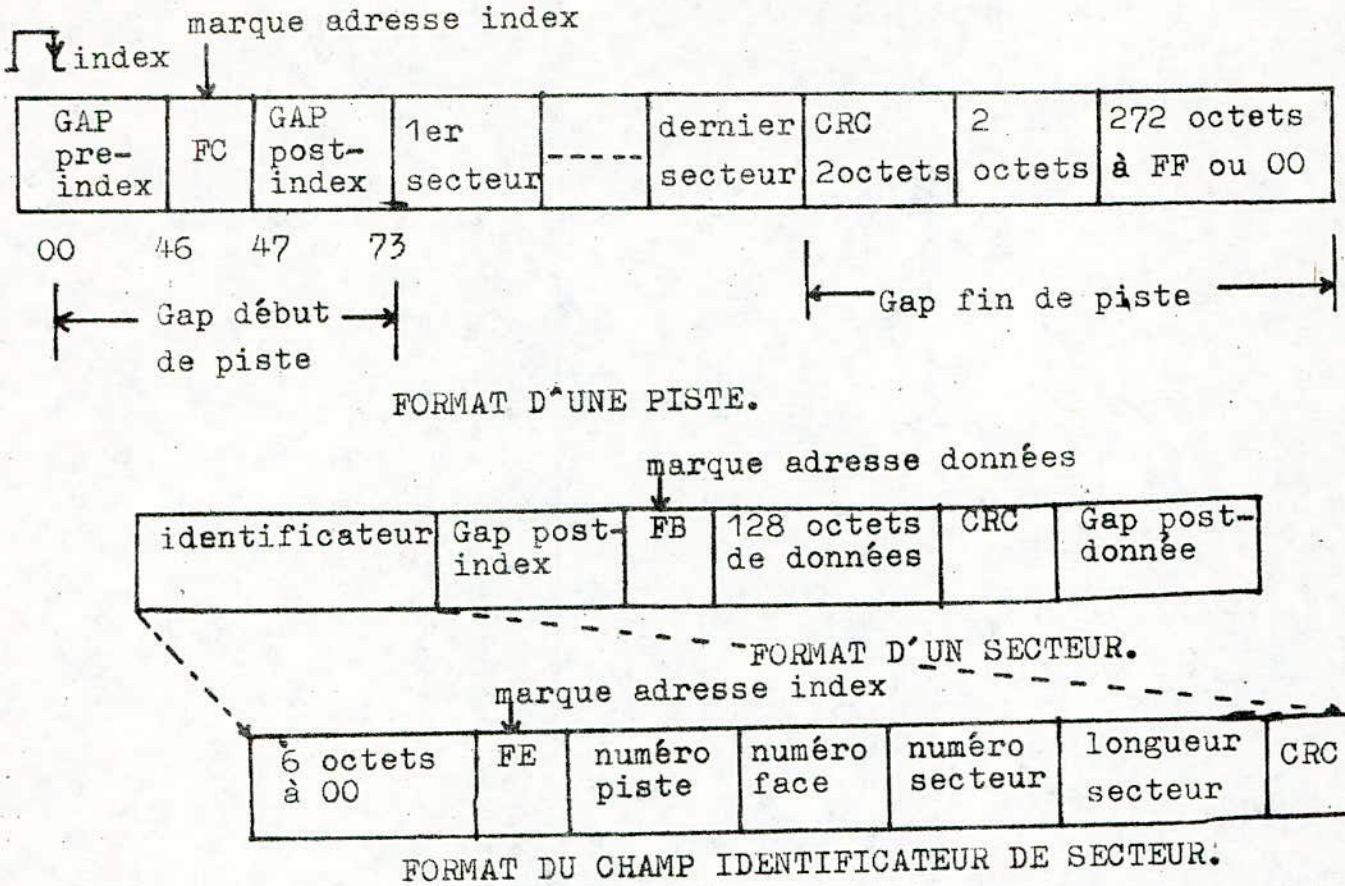


Fig.4: Format des informations sur un disque souple.

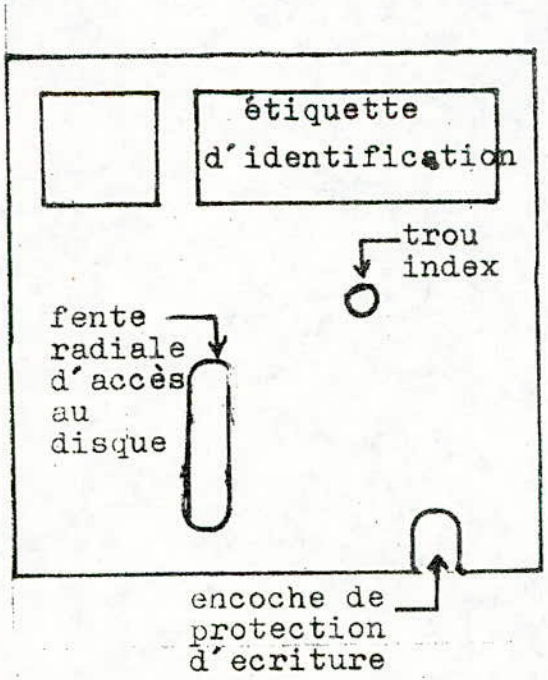


Fig.2: Présentation d'un disque souple.

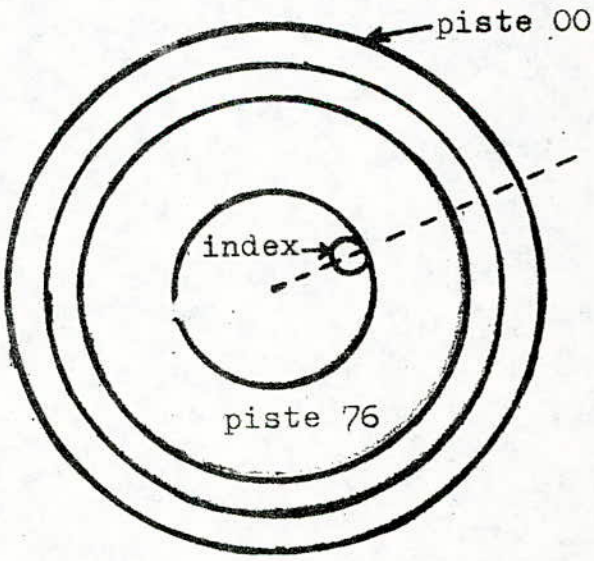


Fig.3: Pistes et index.

- Gap de postindex: défini comme étant l'espace séparant la marque d'adresse et celle de l'identification.

Chaque champ est logé dans la marque adresse. On distingue ainsi 4 types de marques adresses utilisées pour identifier les diffé-rents types de champs:

* Marque adresse d'index: située au début de chaque piste et devant le 1er enregistrement sur la piste.
* Marque adresse d'identification.
* Marque adresse de donnée non rayée.
* Marque adresse de donnée rayée.

En effet, il peut arriver au cours de l'enregistrement, que certaines données soient mal écrites; on les identifie alors pour faciliter u leur lecture.

Cette organisation aussi bonne soit-elle ne nous arrange pas: Elle enregistre les données, à la suite l'une de l'autre, sans tenir compte du bloc d'informations de même nature.

Alors que nous, nous désirons avoir notre information par bloc; car rappelons que nous voulons stocker des résultats d'un système multiprocesseur; nous devrions donc pouvoir stocker les résultats d'un processeur Pi sur un même bloc (qui peut être alors 1 ou plusieurs secteurs). C'est ainsi que nous organisons la disquette comme suit:

Les informations sont inscrites sur la disquette en bloc de 64 secteurs par processeur. Ce qui nous permet d'avoir 1024 résultats d'un processeur Pi (sachant que chaque valeur de résultat s'écrit sur 8 octets).

Chaque bloc est identifié par le pas de calcul et le numéro du processeur; plus les autres modes d'identification déja cités.

A-6- Opération d'écriture:

Comme nous l'avons déja dit, au paragraphe A-5-), l'écriture de la donnée est généralement faite suivant la technique de la modu-lation de fréquence. Par contre ces informations doivent être envo-yées à l'unité de disque souple sous un format bien défini.

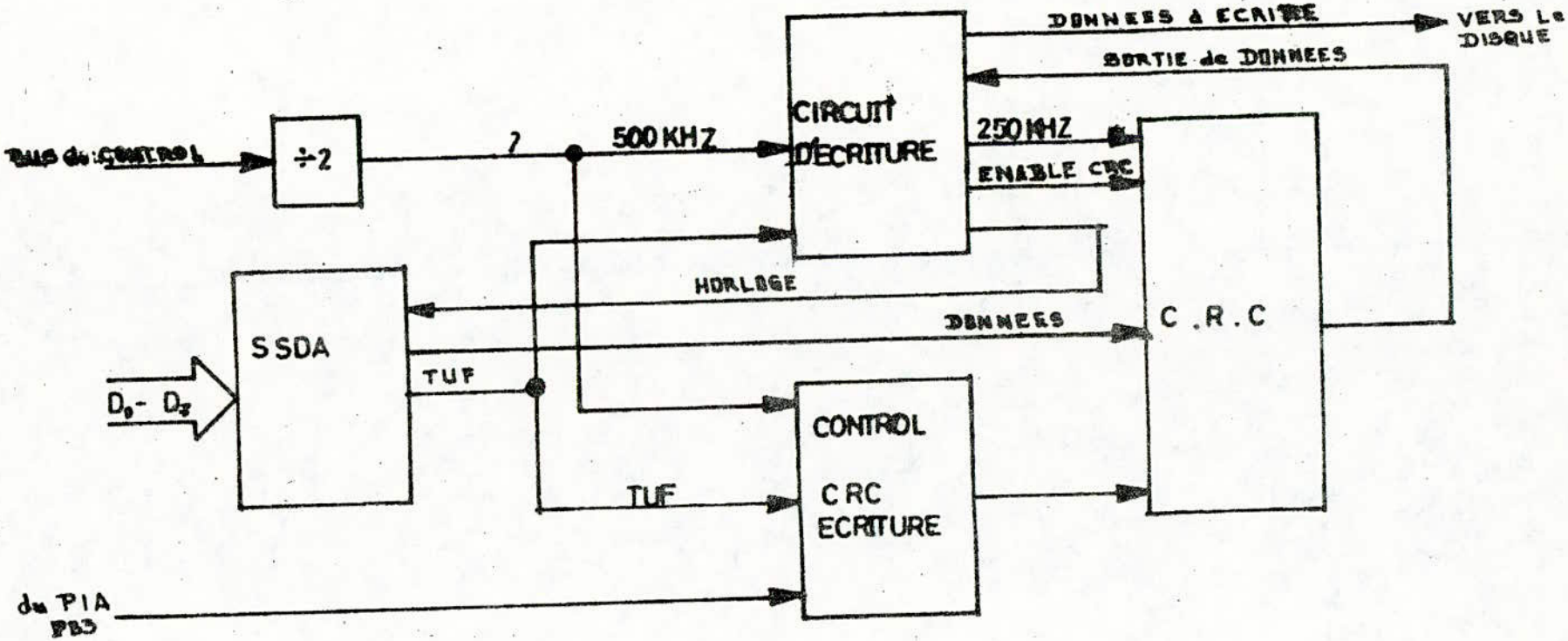


Fig. 6: SCHEMA du CIRCUIT d'ECRITURE

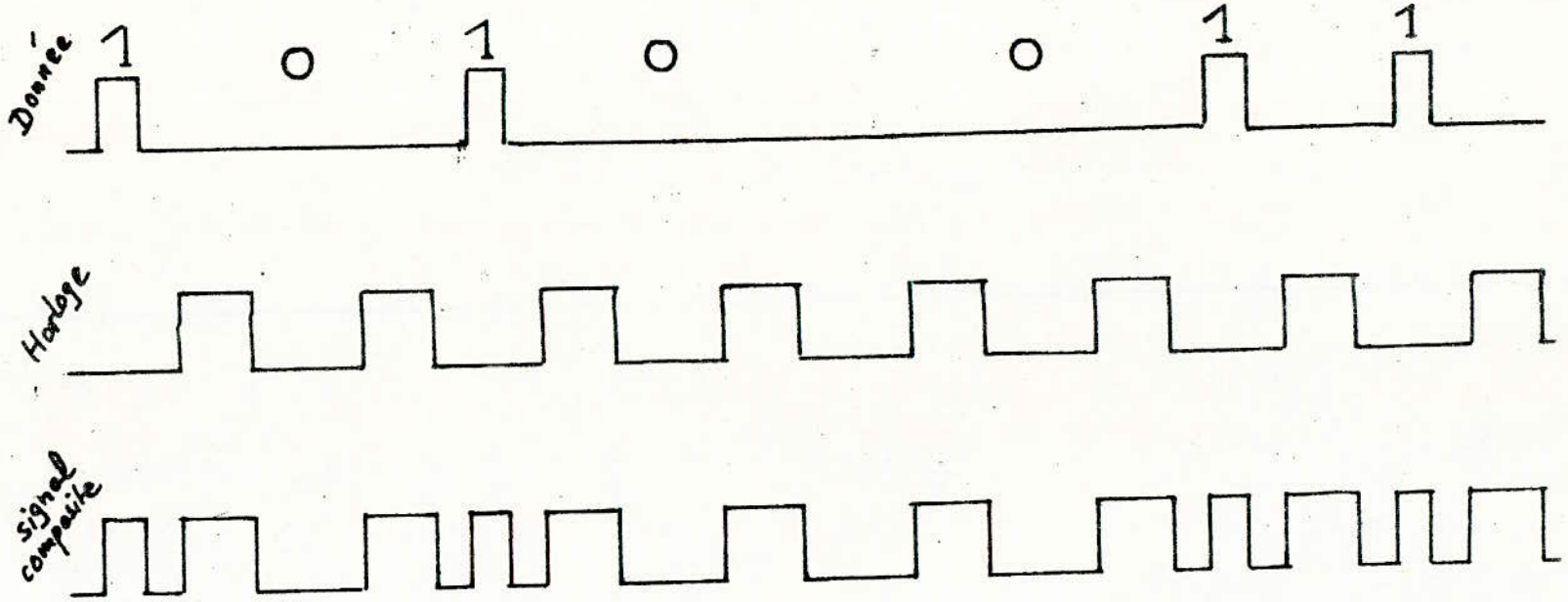


fig. 5 : Format des données.

Ce format consiste à entrelacer les bits de données avec le signal d'horloge au niveau"1" conformément à la fig.5.

Pendant l'écriture,le bit le plus significatif est transféré en 1er;de même à la lecture.

Cette opération d'écriture s'effectue en 7 étapes:

- Chargement du registre de piste.
- Envoi de la commande de recherche du numéro de secteur désiré (SEEK).
- Etape de positionnement de la tête.
- Envoi de la commande d'écriture.
- Chargement des donnees, dès réception de la requête de données.
- Chargement des données restantes.
- Vérifier les indicateurs d'erreur CRC.

Pour le circuit d'écriture:(Voir la fig.6)

A-7- Opération de lecture: (Voir le circuit de lecture fig.7)

Pendant cette opération,comme pour l'écriture,l'adaptateur série de données est mis en jeu pour faire la conversion série-parallèle (en écriture il fait la conversion parallèle série)des bits de données issus du signal composite.

Cette opération s'effectue en 5 étapes:

- Chargement du registre de pistes.
- Envoi du signal de commande de recherche du numéro du secteur désiré.
- Etape de positionnement de la tête.
- Transfert des données vers le µP sous contrôle,par interruption.
- Vérifier que l'opération s'est bien effectuée après le tansfert du nombre correct de données.

Cette dernière étape,aussi bien pendant l'opération de lecture que d'écriture,confirme l'authenticité de l'information sur laquelle porte l'opération.Elle est programmée ou non par l'uti-lisateur.Cependant,surtout à l'écriture,sa programmation augmente le temps d'exécution du programme par le fait que l'information

.../...

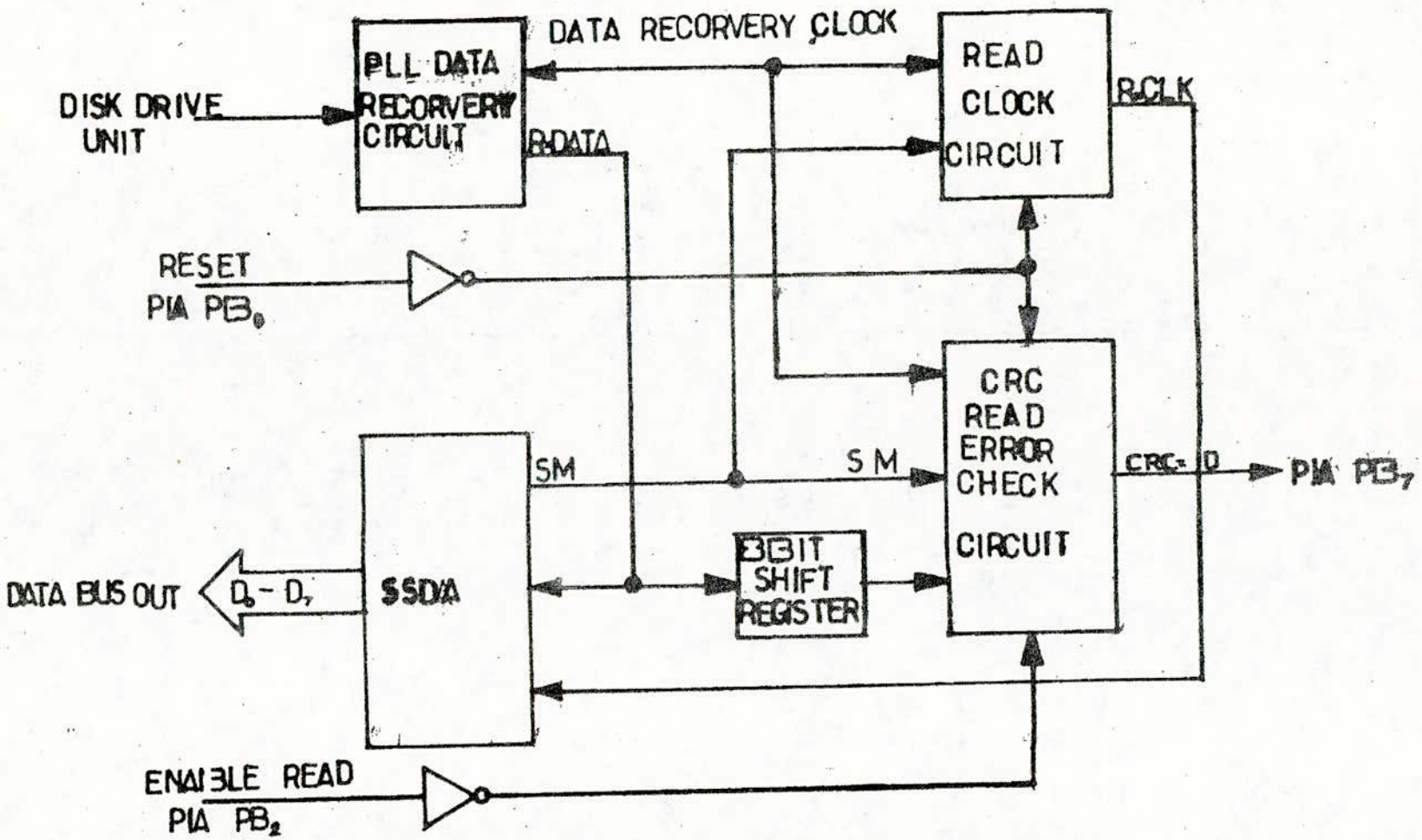


Fig 7 : SCHEMA SYNOPTIQUE du CIRCUIT de LECTURE

écrite est relue et la réponse envoyée à l'utilisateur avant de continuer le programme principal.

A-8- Messages d'erreur:

Toutes ces opérations peuvent être réalisées avec erreur. Comment s'apercevoir de l'erreur aussitôt établie?
Sachant qu'aucun système n'est infaillible et que, quelque soit les précautions prises lors de l'élaboration d'un programme, il est difficile de parvenir au 1er coup à un résultat parfait, le cons-tructeur a prévu un octet signalisateur d'erreur dans le sous-programme d'initialisation du système. Cet octet, connu sous l'éti-quette de FDSTAT (Floppy Disk State ou état du disque souple), contient un message sous forme héxadécimale; ces codes s'étendent 30_{16} jusqu'à 39_{16}.
Si aucune erreur n'a lieu en opérant sur le disque, l'octet de FDSTAT contient le code 30. Par contre la presence d'au moins une erreur change l'état de cet octet en un autre code selon le type d'erreur à signaler. Automatiquement, le disque est arrêté ou du moins l'opération est suspendue et le message d'erreur apparait sur la visu.

Définition des messages d'erreur sur le disque:

(Les messages sont codés en héxadécimal):

30: apparait quand aucune erreur n'a lieu pendant l'opération.
31: indique que le contrôle de redondance cyclique est en erreur.
32: surgit quand le disque est protégé contre l'opération d'écri-ture.
33: indique l'état "NON PRET", du disque, à l'opération.
34: indique que la donnée à lire est rayée ou effacée.
35: parvient quand l'opération sur le disque n'est pas terminée alors que le Timer de l'interface a dépassé le temps requis par l'opération.
36: apparait quand l'adresse du disque n'est pas validée. C'est à dire que la somme des nombres du STRSCT et NUMSCT dépasse le

nombre de secteurs sur le disque.

37: apparait quand la remise de la tête de lecture/écriture est mal faite.

38: indique qu'une donnée devant être lue sur le secteur n'est pas trouvée après 4 essais de commande de lecture.

39: indique une mauvaise écriture de l'adresse.

Il est possible de savoir sur quel secteur porte l'erreur par l'équation suivante:

PSNE = STRSCT + NUMSCT - SCTCNT - 1 .

avec:

PSNE = numéro du secteur en erreur.

STRSCT = numéro du secteur à partir duquel commence l'opération.

NUMSCT = nombre de secteurs sur lesquels porte l'opération.

SCTCNT = contient au début de chaque opération la valeur qui se trouve dans NUMSCT. Pendant l'opération, il se décrémente après l'opération sur chaque secteur.

****

B- Logiciel de sortie sur disque:

B-1- Présentation des résultats:

Les résultats se trouvent dans une table BUVAR,dans la MC, où les processeurs Pi viennent déposer leur résultat à chaque pas de calcul Pc;ce pas étant déterminé par le plus lent.
Au lancement du calcul,on laisse le système se stabiliser et on s'intéresse à son évolution(donc à ses résultats)à partir du pas demandé(Pd correspondant à un temps initial pour nous) où l'on transfère les résultats de BUVAR vers le disque et ceci jusqu'à un temps final(fixé par le nombre de valeurs voulu).
Cependant ce transfert n'est pas aussi rapide non plus systémati--que.On ne peut stocker tous les résultats de la simulation d'une part,et on ne peut pas envoyer un résultat pour le disque d'une manière quelconque,vue sa structure,d'autre part.
Par conséquent,dans notre programme,on force l'utilisateur à ne demander que six résultats parmi les 32 présents en MC.De plus nous nous sommes fixés le nombre de valeurs de chaque résultat (des 6 demandés) à 1024.Chaque Pi occupera une zone résultat de 64 secteurs,sur le disque,comme prévu au paragraphe A-5 de ce chapitre.
Le résultat du processeur Pi est étiqueté de"variable Vi" avec i=0...31.

B-2- Organisation des tables du programme:

L'organisation de la mémoire du programme est la suivante:

- Le début des valeurs de Vi est à BUVADE = BUVAR + 16. Les 14 premiers octets sont réservés au moniteur de gestion;les 2 autres octets sont,eux,réservés au numéro du pas courant(pas de calcul) et sont désignés par Mc.
- Les valeurs de Vi sont d'abord organisées en blocs de 128 octets soit 16 valeurs de Vi(chaque Valeur de Vi étant écrite sur 64 bits flottants).

Ces blocs de 128 octets (au nombre de 6) sont réservés dans une table BUDISC à l'adresse : BUDEB = BUDISC + 15.Les premiers octets de BUDISC se repartissent comme suit:

-1 octet Mb:utilisé comme compteur du nombre de decalages lors du test et porte directement le numéro de la variable demandée.

-2 octets Md:utilisés pour le numéro du pas demandé.

-1 octet M_N:utilisé pour compter le nombre de valeurs de Vi stockées dans le bloc j de 128 octets.

-1 octet Mj:utilisé comme compteur des blocs de 128 octets.

-1 octet Ms:utilisé comme compteur de secteurs remplis de valeurs de Vi.

-6 octets d'identification I1,...I6 : réservés pour contenir le numéro des Vi stockées dans le disque.

-4 octets de test(soit 32 bits)Mo,M1,M2,M3:où l'utilisateur demande les Vi. Il met "1"au bit bi si Vi est à sortir;
"0" sinon.

*Organisation de BUDISC:(voir fig.8)

*Organisation de BUVAR :(voir fig.9)

*Organisation de la page zéro:

Cette page contient les opérandes des opérations élémentaires, les résultats intermédiaires des sous/prog.;pour permettre l'accès direct.

0061 : M_A : 2ème opérande du s/p ADDIC2.
62 : SAUVEB sauvegarde B
63 : So
64 : S1
65 : S2
66 : S3
67 : SAUVX1
69 : SAUVX2
6B : U-3
6C : U-2
6D : U-1
6E : U 1er opérande(=j) } réservées au s/p MULTI2

.../...

006F : V 2ème opérande du MULTI2(=128)
70 : V+1
71 : R-1
72 : R
73 : R+1 } résultat du s/p MULTI2;1er opérande du s/p
74 : R+2 } ADDIC2;résultat de ADDIC2.
75 : R+3 résultat de $8xM_N$
76 : R+4 }
77 : R+5 } résultat du s/p CALDAA(128j+8N+BUDEB)
78 : R+6 }
79 : R+7 } résultat du s/p CALDAD(BUVADE+8i)
7A : TO7T table intermediaire BUVAR/BUDISC
7B : TO7T+1
7C : TO7T+2
7D : TO7T+3
7E : TO7T+4
7F : TO7T+5
80 : TO7T+6
81 : TO7T+7

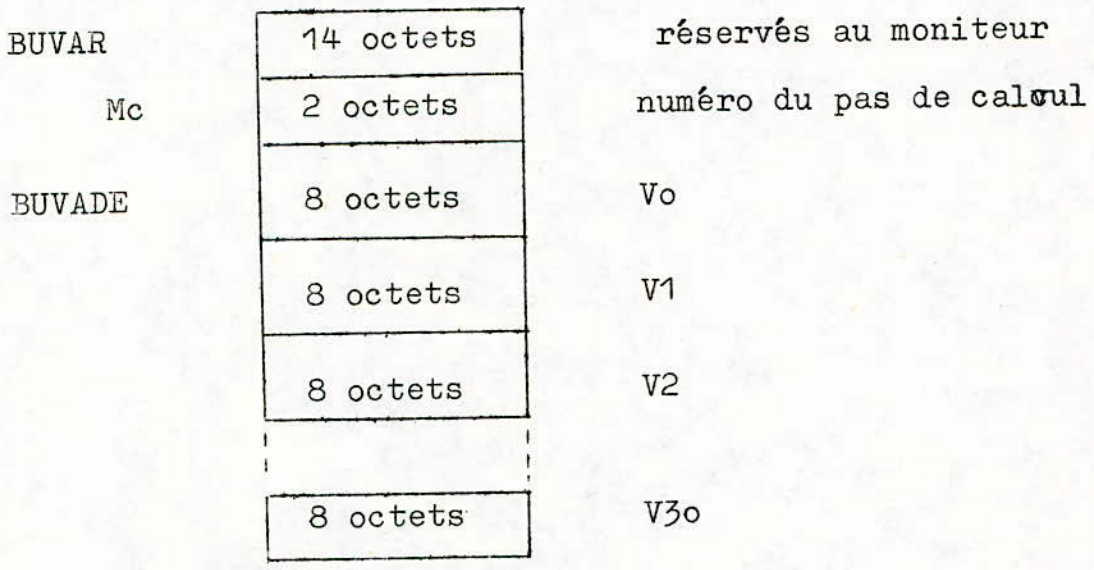


Fig.9: Organisation de la table BUVAR.

.../...

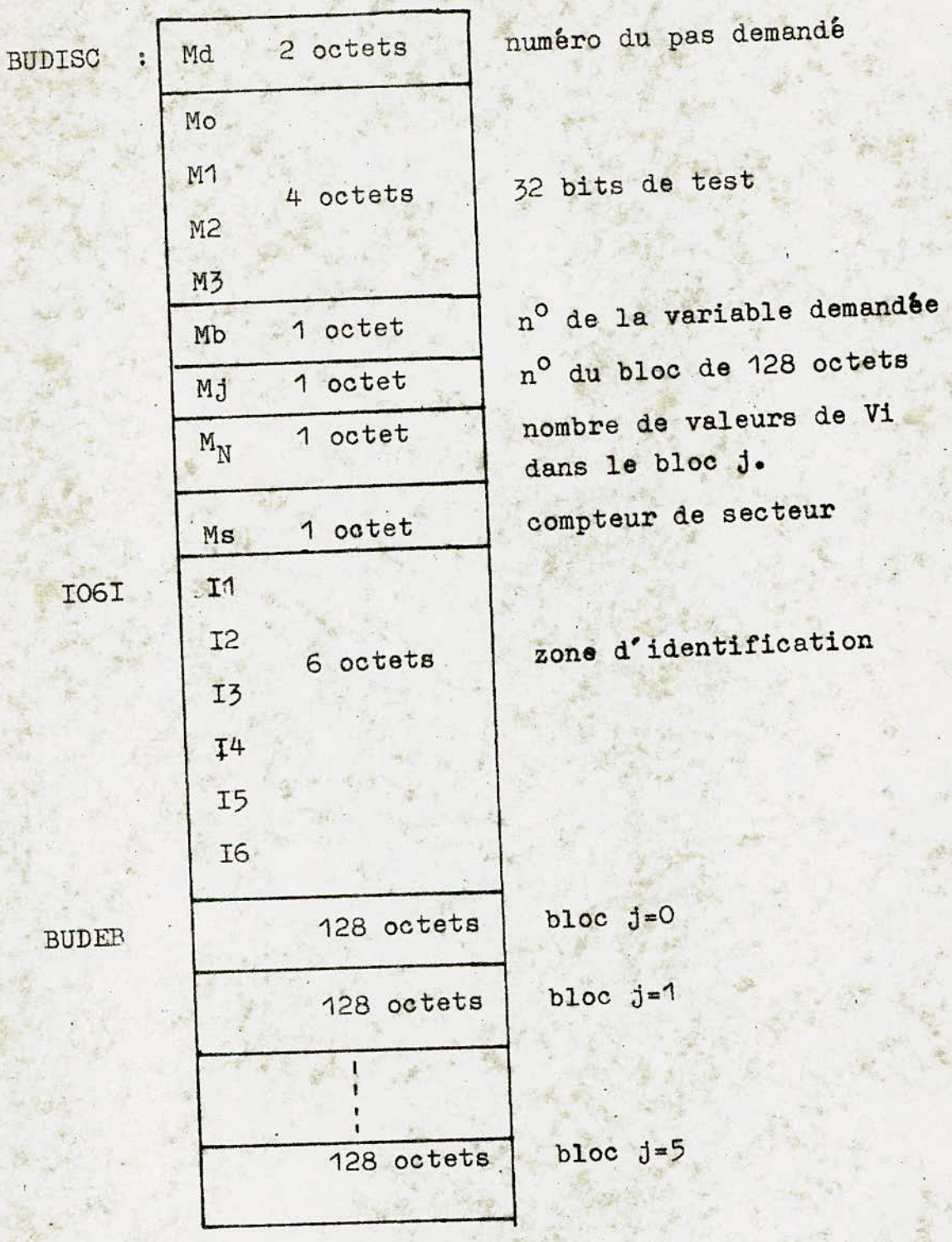
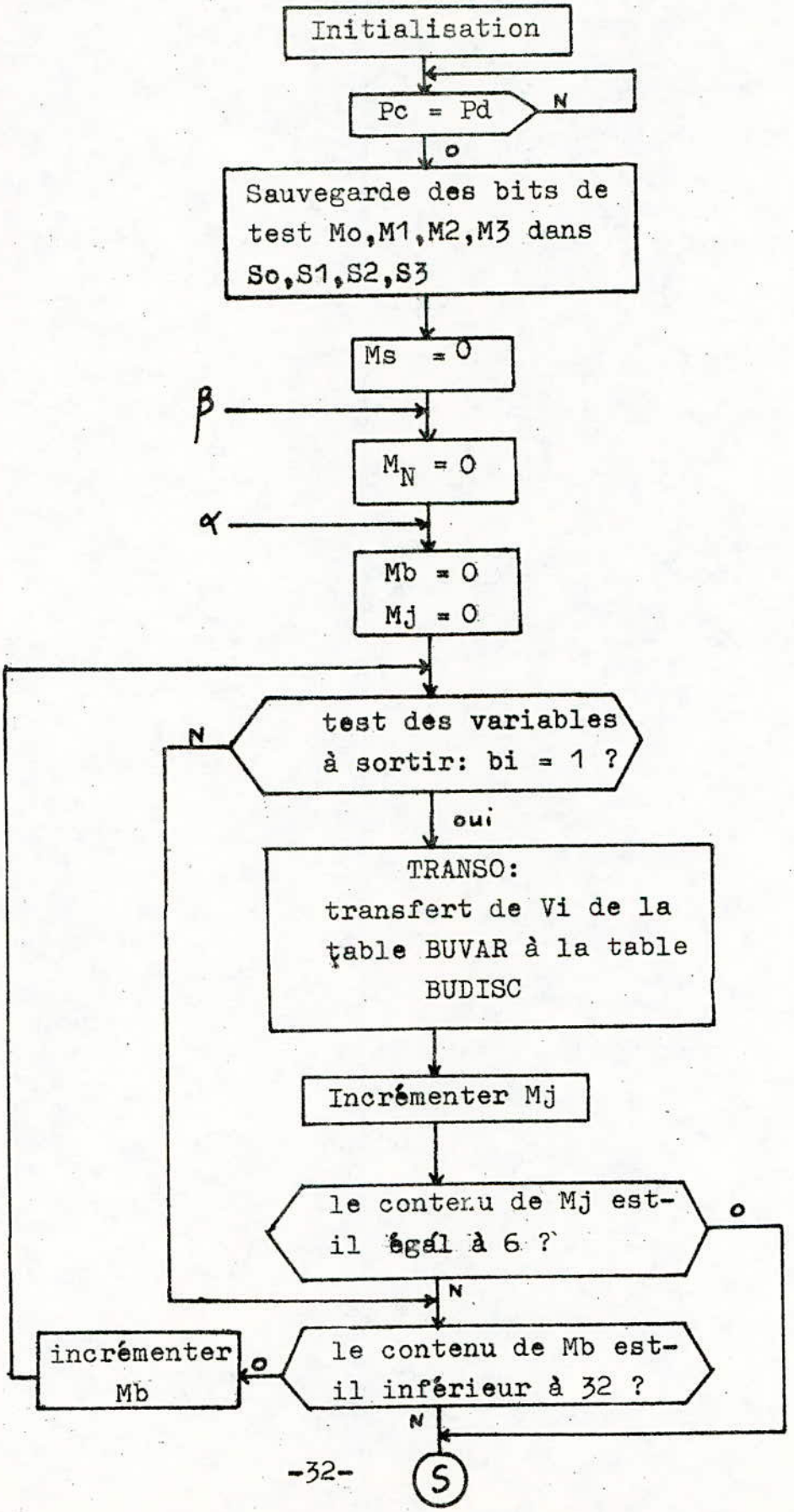


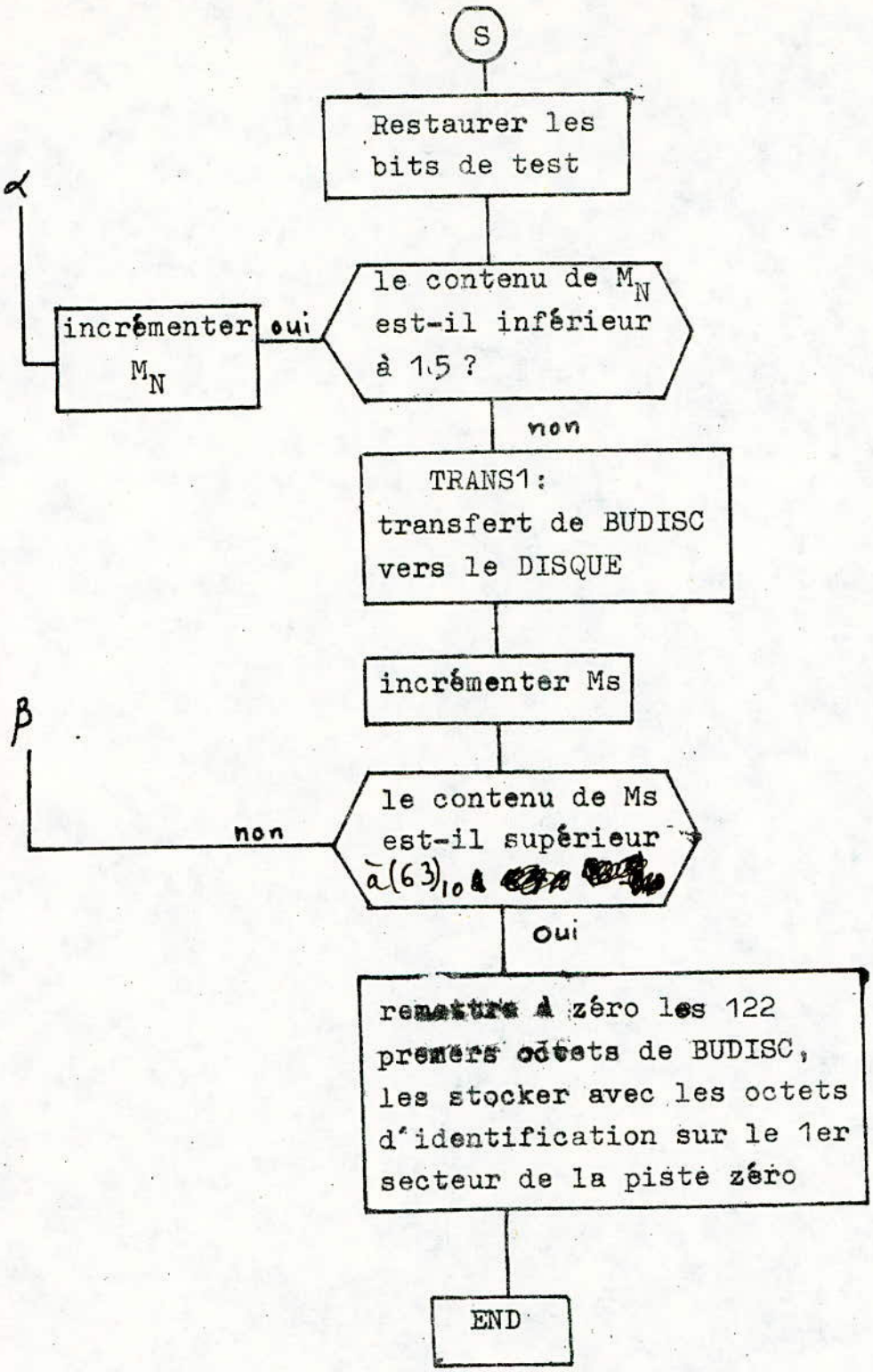
Fig.8: Organisation de la table BUDISC.

.../...

ORGANIGRAMME PRINCIPAL.



.../...


S
Restaurer les bits de test
α
le contenu de M_N est-il inférieur à 1.5 ?
oui
incrémenter M_N
non
TRANS1: transfert de BUDISC vers le DISQUE
incrémenter Ms
β
le contenu de Ms est-il supérieur à $(63)_{10}$?
non
oui
remettre à zéro les 122 premiers octets de BUDISC, les stocker avec les octets d'identification sur le 1er secteur de la piste zéro
END

.../...

B-3- Organigrammes et programmes :

B-3-a- Organigramme principal:(voir a la page 32 et 33).

B-3-b- Programme principal:

```
          * Mo          chargement en immédiat,des octets de test,
          * M1          par l'utilisateur.
          * M2
          * M3
COMPAS    LDX     Md    comparaison du pas demandé au pas de
 BOUCL    DEX           calcul(pas courant).
          CPX   Mc
          BNE   BOUCL
          LDA A  Mo     sauvegarde des bits de test.
          STA A  So
          LDA A  M1
          STA A  S1
          LDA A  M2
          STA A  S2
          LDA A  M3
          STA A  S3
          CLR    Ms     mise à zéro du compteur de secteur
SECTOR    CLR    M_N    mise à zéro du compteur de
                        valeur dans le bloc j.
MISAJO    CLR    Mj     mise à zéro du compteur de blocs de 128
                        octets.
          CLR    Mb     mise à zéro du compteur de décalages
          LDX ## Ø I06I pointer l'adresse de la zone d'identi-
                        -fication.
          CLC           mise à zéro du bit carry
          LDA B ## Ø 8  chargement de l'acc.B par le numéro
                        du bit de poids fort de l'octet à tester.
```

```
ROTATØ   ROR    Mo          test des variables demandées au
         BCC    TOTO        1er octet.
         JSR    TRANSØ      saut au s/p transfert BUVAR/BUDISC.
         INX
         CPX #  Ø I06I+6    tester si les 6 Vi sont déja transférées
         BEQ    TESTN       branchemant a l'adresse TESTN.
TOTO     INC    Mb          tester si le test du 1er octet est
         CMP B  Mb          terminé.
         BNE    ROTATØ      boucler à ROTATØ
         CLC                mise à zéro du bit carry
         LDA B # Ø 10
ROTAT1   ROR    M1          test du 2ème octet
         BCC    NONO
         JSR    TRANSØ
         INX
         CPX #  Ø I06I+6
         BEQ    TESTN
NONO     INC    Mb
         CMP B  Mb
         BNE    ROTAT1      boucler à ROTAT1
         CLC
         LDA B # Ø 18
ROTAT2   ROR    M2          test du 3ème octet
         BCC    MIDO
         JSR    TRANSØ
         INX
         CPX #  Ø I06I+6
         BEQ    TESTN
MIDO     INC    Mb
         CMP B  Mb
         BNE    ROTAT2      boucler en ROTAT2
         CLC
         LDA B # Ø 20
```

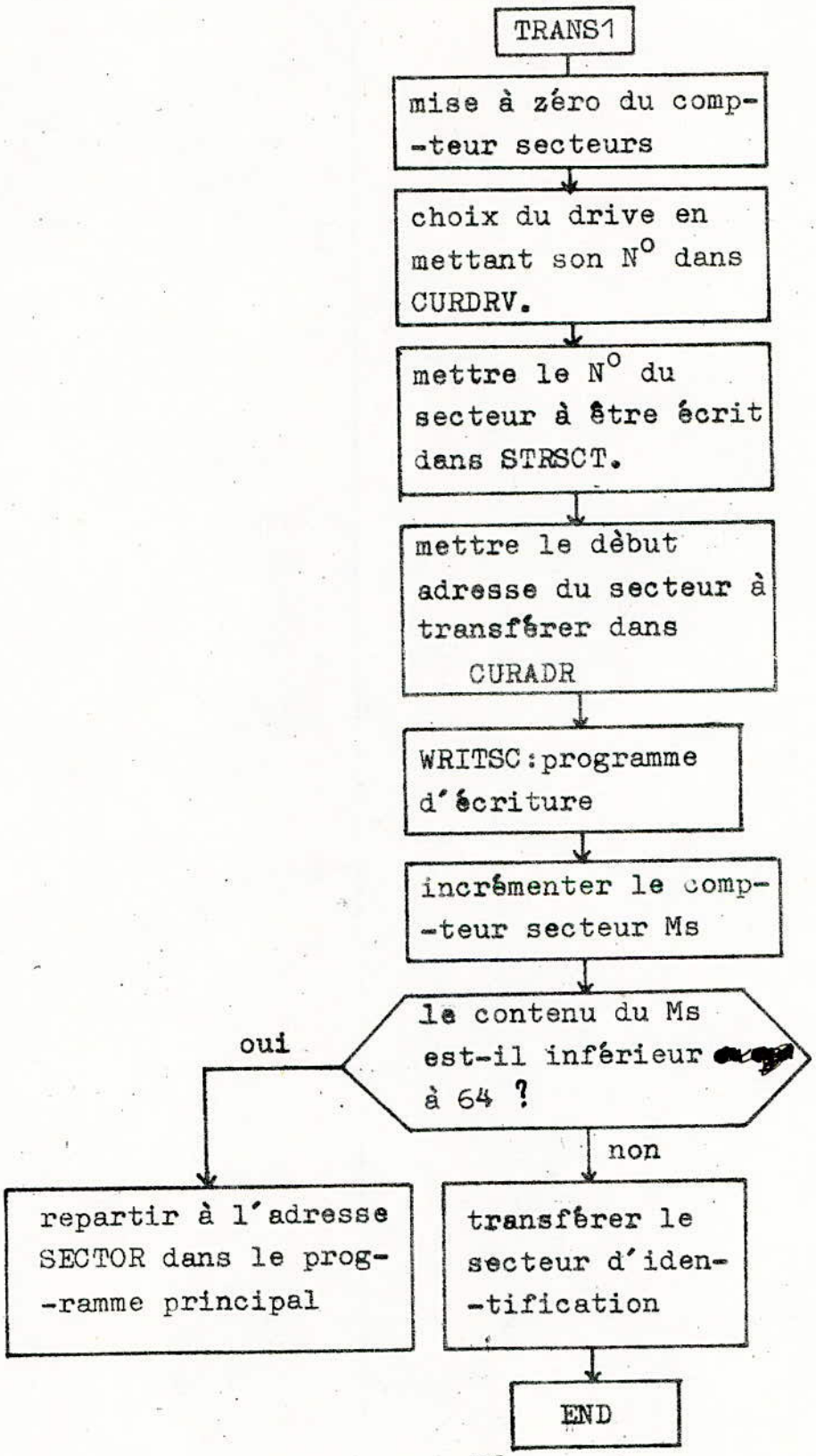
```
ROTAT3   ROR     M3          test du 4ème octet de test.
         BCC     MIMO
         JSR     TRANSØ
         INX
         CPX #Ø  I06I+6
         BEQ     TESTN
MIMO     INC     Mb
         CMP B   Mb
         BNE     ROTAT3
TESTN    LDA A   So          restaurer les bits de test.
         STA A   Mo
         LDA A   S1
         STA A   M1
         LDA A   S2
         STA A   M2
         LDA A   S3
         STA A   M3
         LDA A   M_N         tester le compteur M_N par rapport à
                             15;l'incrémenter s'il est inférieur.
         CMP A #Ø  OF
         BEQ     TRANS1      branchement au transfert BUDISC/DISK
         INC A
         STA A   M_N
         JMP     MISAJO
TRANS1   CLR     M_A
         JSR     OSLOAD      saut au programme d'initialisation
                             du disque.
         LDA A #Ø  00        CHoix du DRIVE "0" .
         STA A   CURDRV
         LDX #Ø  01          nombre de secteur à transférer.
         STX     NUMSCT
         CLR     M_N         mise à zéro du compteur M_N:
         CLR     Mj          mise à zéro du compteur Mj.
```

```
LDX #$01        calcul du numéro du secteur à remplir par
                la 1er bloc(bloc j=0).
STX     R+1
JSR     ADDIC2
LDX     R+1     indiquer le 1er secteur à être écrit en
STX     STRSCT  stockant son numéro dans STRSCT.
JSR     SEEK    saut au programme de positionnement de
                la tête sur le secteur contenu dansSTRSCT.
JSR     CALDAA  saut au prog.du calcul de l'adresse du
                secteur a transférer.
LDX     R+4     chargement de X par le résultat de CALDAA.
STX     CURADR  pointer le début adresse du secteur à
                transférer.
JSR     WRITSC  saut au programme écriture.
JSR     RESTOR  saut au prog.de remise de la tête sur la
                piste zéro.
INC     Mj      pointer le prochain bloc à transférer.
LDX #$41        calcul du n° du secteur à contenir le
                bloc(j=1).
STX     R+1
JSR     ADDIC2
LDX     R+1
STX     STRSCT  mettre dans STRSCT le 1er secteur à être
JSR     SEEK    écrit,positionner la tête sur ce secteur.
JSR     CALDAA
LDX     R+4
STX     CURADR  pointer le secteur à transférer.
JSR     WRITSC
JSR     RESTOR  ramène la tête sur la piste zéro.
INC     Mj
LDX #$81        calcul du n° du secteur à contenir le
STX     R+1     bloc(j=2)
JSR     ADDIC2
LDX     R+1
STX     STRSCT  mettre le n° du secteur à être écrit.
```

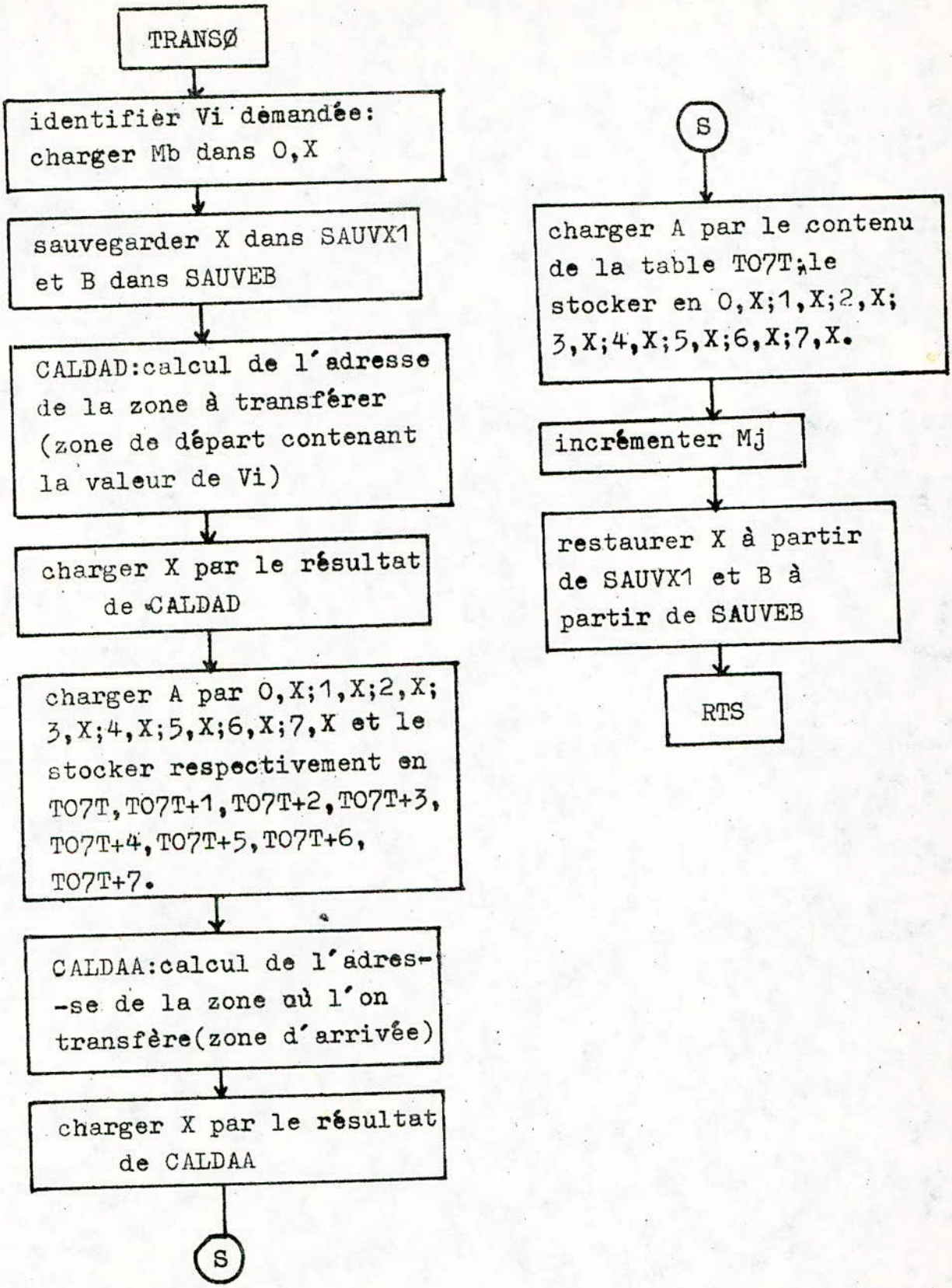
```
JSR     SEEK        positionner la tête sur le secteur contenu dans
JSR     CALDAA      STRSCT.
LDX     R+4
STX     CURADR      mettre dans CURADR le n° du secteur a transférer.
JSR     WRITSC
JSR     RESTOR

INC     Mj          pointer le prochain bloc à transférer.
LDX #$  C1          calcul du n° du secteur à contenir le bloc(j=3).
STX     R+1
JSR     ADDIC2
LDX     R+1
STX     STRSCT      indiquer le N° du secteur à être écrit.
JSR     SEEK        ramener la tête sur le secteur contenu dans
                    STRSCT.
JSR     CALDAA      calculer l'adresse du secteur à transférer,
LDX     R+4         mettre son adresse dans CURADR.
STX     CURADR
JSR     WRITSC      saut au prog.écriture
JSR     RESTOR      ramener la tête sur la piste zéro.

INC     Mj
LDX #$  101         calcul du N° du secteur à contenir le bloc(j=4)
STX     R+1
JSR     ADDIC2
LDX     R+1
STX     STRSCT      indiquer le N° du secteur à être écrit,et y
JSR     SEEK        positionner la tête.
JSR     CALDAA      calculer l'adresse du secteur à transférer,
                    mettre son adresse dans CURADR.
LDX     R+4
STX     CURADR
JSR     WRITSC
JSR     RESTOR

INC     Mj
LDX #$  141         calcul du N° du secteur à contenir le bloc(j=5)
```

```
        STX     R+1
        JSR     ADDIC2
        LDX     R+1
        STX     STRSCT      indiquer le N° du secteur à être
        JSR     SEEK        écrit,positionner la tête.
        JSR     CALDAA      calculer l'adresse du secteur à
        LDX     R+4         transférer et la mettre dans CURADR.
        STX     CURADR
        JSR     WRITSC
        JSR     RESTOR

        INC     Ms          incrémenter le compteur secteurs,le
        LDA A   Ms          comparer à (64)10
        STA A   MA
        CMP A #$ 40
        BEQ     IDENTI      saut à l'adresse IDENTI(transfert
                            du secteur di d'identification).
        JMP     SECTOR
IDENTI  LDX     BUDEB
        CLR A
BOUCLF  STA A   0,X
        INX
        CPX     BUDEB+(6.128-6)10
        BNE     BOUCLF
        LDX #$  00          mettre le N° du secteur (zéro)dans
        STX     STRSCT      STRSCT,son adresse dans CURADR;ce
        JSR     SEEK        secteur portera les octets d'identi-
        LDX     I06I        -fication.
        STX     CURADR

        END
```

Programme TRANS1:Cette partie du programme principal effectue le transfert de la table BUDISC vers le disque.



.../...


TRANSØ
identifier Vi demandée: charger Mb dans 0,X
sauvegarder X dans SAUVX1 et B dans SAUVEB
CALDAD:calcul de l'adresse de la zone à transférer (zone de départ contenant la valeur de Vi)
charger X par le résultat de CALDAD
charger A par 0,X;1,X;2,X;3,X;4,X;5,X;6,X;7,X et le stocker respectivement en TO7T,TO7T+1,TO7T+2,TO7T+3,TO7T+4,TO7T+5,TO7T+6,TO7T+7.
CALDAA:calcul de l'adresse de la zone où l'on transfère(zone d'arrivée)
charger X par le résultat de CALDAA
S
S
charger A par le contenu de la table TO7T;le stocker en 0,X;1,X;2,X;3,X;4,X;5,X;6,X;7,X.
incrémenter Mj
restaurer X à partir de SAUVX1 et B à partir de SAUVEB
RTS

.../...

Programme TRANSØ:

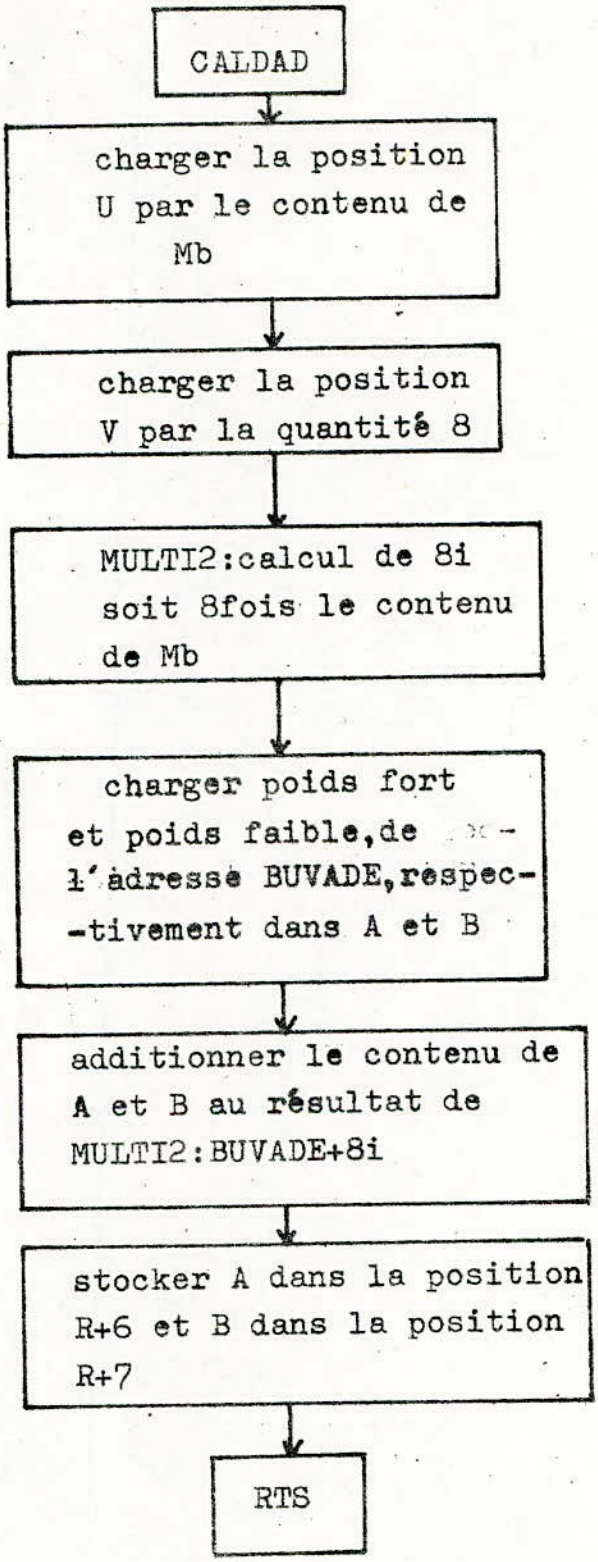
Ce programme effectue le transfert des valeurs des Vi de la table BUVAR vers la table BUDISC.

```
LDA A     Mb        mettre le N° de la variable dans la
STA A     0,X       zone d'identification pointée par X.
STA B     SAUVEB    sauvegarder l'acc.B
STX       SAUVX1    sauvegarder l'index
JSR       CALDAD    saut au s/p du calcul de l'adresse de
                    départ(qui contient Vi à transférer).
LDX       R+6       chargement de X par le résultat de
                    CALDAD.

LDA A     0,X       transfert de Vi dans la zone inter-
STA A     TO7T      -médiaire To7T.
LDA A     1,X
STA A     TO7T+1
LDA A     2,X
STA A     TO7T+2
LDA A     3,X
STA A     TO7T+3
LDA A     4,X
STA A     TO7T+4
LDA A     5,X
STA A     TO7T+5
LDA A     6,X
STA A     TO7T+6
LDA A     7,X
STA A     TO7T+7
JSR       . CALDAA  saut au s/p du calcul de
                    l'adresse de la zone d'arrivée.
LDX       R+4       charger X par le résultat de CALDAA.
```

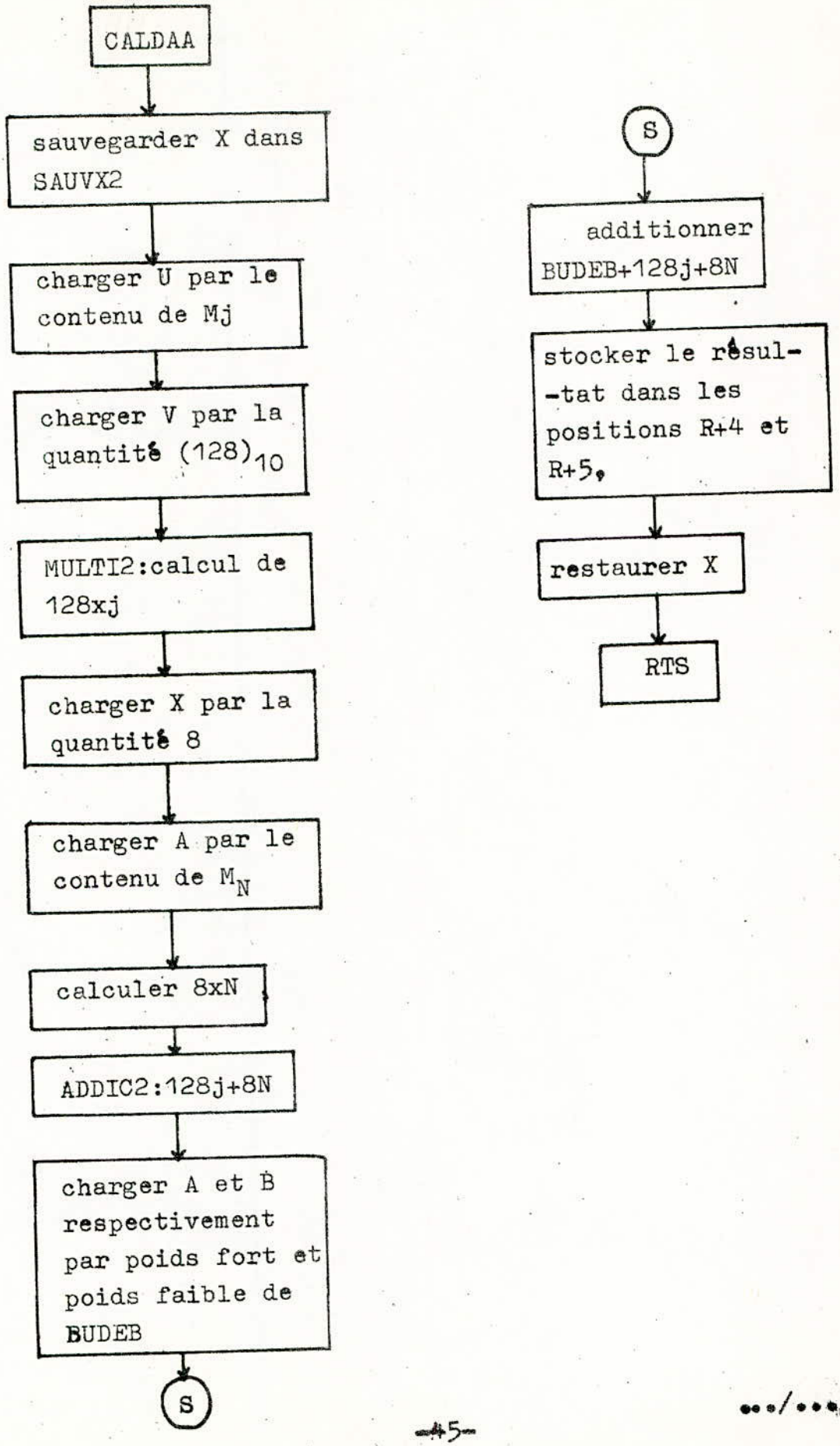
```
LDA A       T07T      transfert de Vi dans BUDISC.à l'dress
STA A       0,X       -se pointée par X.
LDA A       T07T+1
STA A       1,X
LDA A       T07T+2
STA A       2,X
LDA A       T07T+3
STA A       3,X
LDA A       T07T+4
STA A       4,X
LDA A       T07T+5
STA A       5,X
LDA A       T07T+6
STA A       6,X
LDA A       T07T+7
STA A       7,X

INC         Mj        incrémenter le compteur de blocs.
LDX         SAUVX1    restaurer X à partir de SAUVX1.
LDA B       SAUVEB    restaurer B.
RTS                   retour de sous-programme.
```

****

.../...


CALDAD
charger la position U par le contenu de Mb
charger la position V par la quantité 8
MULTI2:calcul de 8i soit 8fois le contenu de Mb
charger poids fort et poids faible,de l'adresse BUVADE,respectivement dans A et B
additionner le contenu de A et B au résultat de MULTI2:BUVADE+8i
stocker A dans la position R+6 et B dans la position R+7
RTS

.../...


CALDAA
sauvegarder X dans SAUVX2
charger U par le contenu de Mj
charger V par la quantité $(128)_{10}$
MULTI2:calcul de 128xj
charger X par la quantité 8
charger A par le contenu de M_N
calculer 8xN
ADDIC2:128j+8N
charger A et B respectivement par poids fort et poids faible de BUDEB
S
S
additionner BUDEB+128j+8N
stocker le résul--tat dans les positions R+4 et R+5,
restaurer X
RTS

.../...

Programme CALDAD:

Ce s/p effectue le calcul de l'adresse de départ(l'adresse où se trouve Vi,à transférer,dans BUVAR).Le résultat est dans les 2 positions mémoires R+6,R+7.Cette adresse est donnée par la relation: BUVADE + 8i ; i étant le contenu de Mb directement.

```
        LDA A       Mb          chargement du 1er opérande (i) dans
        STA A       U           la position mémoire U.
        LDA A #$    8           chargement du 2eme operande (8) dans
        STA A       V           la position V.
        JSR         MULTI2      saut au s/p du calcul de 8i
        LDA B #$    BUVADE(poids faible)
        LDA A #$    BUVADE(poids fort)
        ADD B       R+2         addition avec retenue:BUVADE+8i
        ADCA        R+1
        STA A       R+6         stocker le résultat en R+6,R+7
        STA B       R+7
        RTS                     retour de sous-programme.
```

Programme CALDAA:

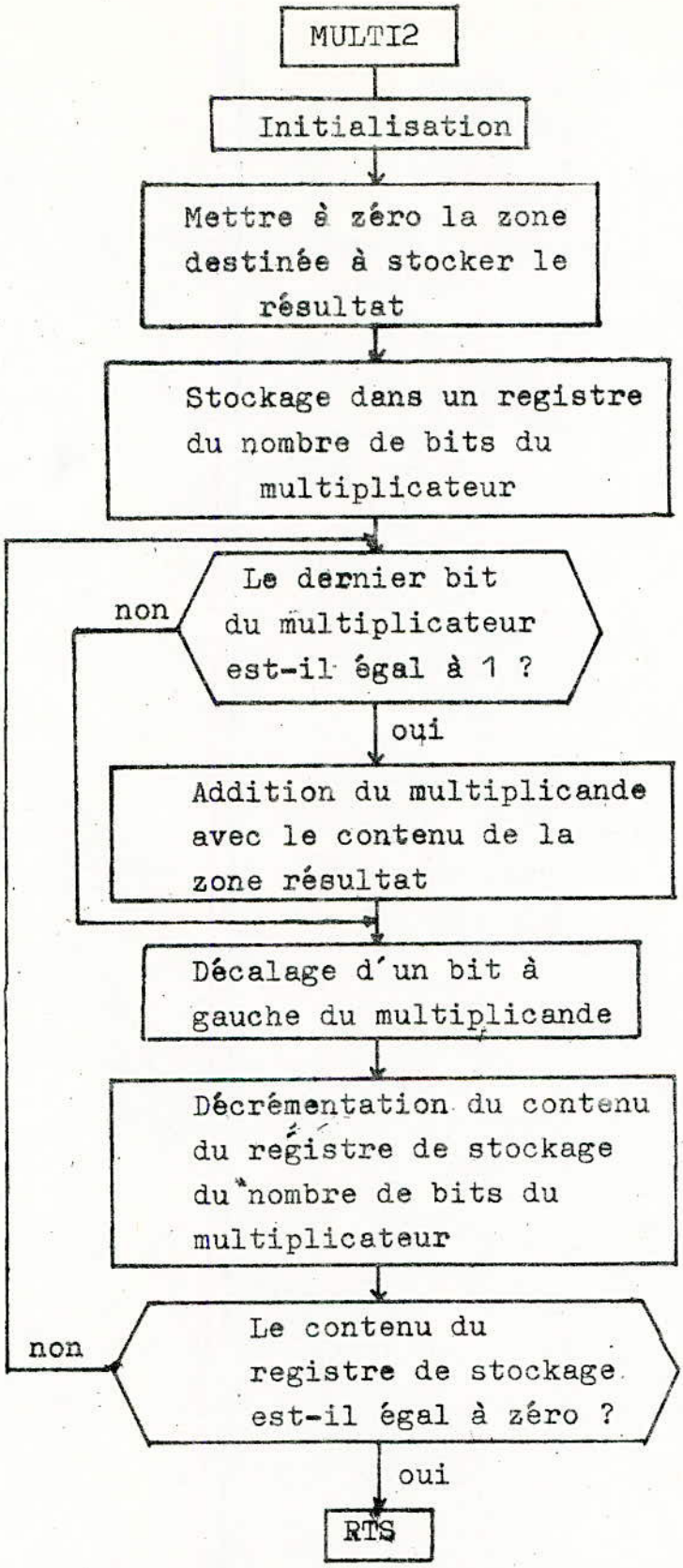
Ce s/p effectue le calcul de l'adresse d'arrivée dans BUDISC.Le résultat est dans les positions R+4,R+5.Cette adresse est calculée à partir de la formule : BUDEB + 128j + 8N .

```
        STX         SAUVX2      sauvegarder X dans SAUVX2.
        LDA A       Mj          charger le 1er opérande(j)dans U.
        STA A       U
        LDA A #$    80          charger le 2eme operande (128 10) dans V.
        STA A       V
        JSR         MULTI2      saut au s/p du calcul de 128j.
MULTI1  CLR A
        LDX #$      8
```

```
BOUCL9   ADD A _   M_N          calcul de 8N.
         DEX
         BNE       BOUCL9
         STA A     R+3          résultat 8N dans R+3,et dans M_A où
         STA A     M_A          il servira d'opérande.
         JSR       ADDIC2       saut au s/p du calcul de 128j+8N;le
                                résultat est dans R+1,R+2.
ADDIC1   LDA A #  ø BUDEB(poids fort)
         LDA B #  ø BUDEB(poids faible)
         ADD B     R+2          calcul de 128j+8N+BUDEB.
         ADC A     R+1
         STA A     R+4          stockage du résultat dans R+4,R+5.
         STA B     R+5
         LDX       SAUVX2       restaurer l'index.
         RTS                    retour de sous-prog.
```

****

.../...



Multiplication 8x8 bits(avec dépassement)
résultat sur 16 bits.

.../...

PROGRAMME MULTI2

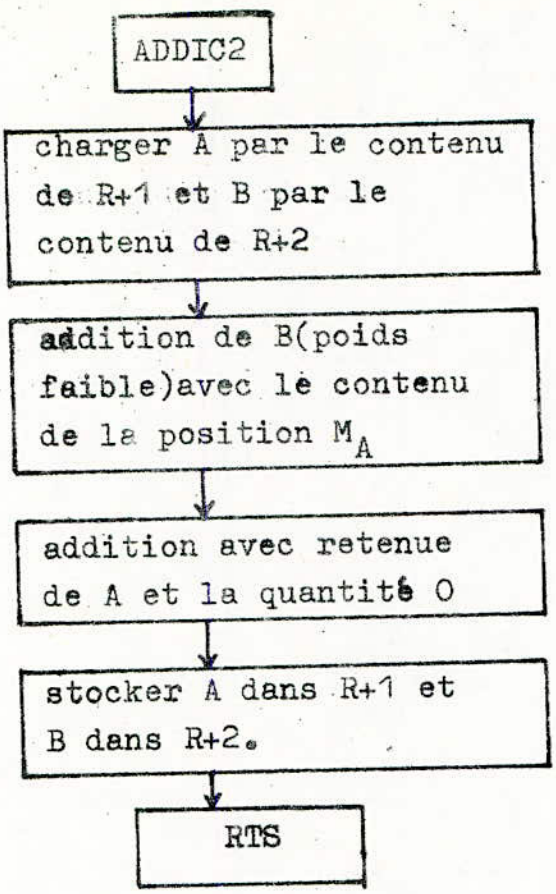
Ce programme effectue la multiplication de 2 nombres binaires sans signe, de 8 bits. Le multiplicande occupe la position mémoire U, le multiplicateur occupe la position V. Les positions U-2, U-3 sont réservées comme zone de manoeuvre permettant le décalage à gauche du multiplicande. La position V+1 est réservée au test de fin de rotation d'un bit à droite du multiplicateur ~~est terminée~~. Le résultat est dans les positions mémoire : R+1, R+2 (16 bits).

```
        CLR A           remise à zéro de l'acc.A
        LDX ## $ 3
                        * remise à zéro de la zone résultat
BOUCL4  STA A R-1,X
        DEX
        BNE   BOUCL4
                        * remise à zéro de la zone de manoeuvre.
        LDX ## $ 2
BOUCL5  STA A U-3,X
        DEX
        BNE   BOUCL5
                        * stockage du nombre de bits du multiplica-
                          -teur.
        LDA B ## $ 8
MULT    LDX ## V
BOUCL6  ROR 0,X
        INX
        CPX ## V+1
        BNE   BOUCL6
        BCC   DEC
                          * addition du multiplicande avec le contenu
                            de la zone résultat.
        LDX ## $ 3
        CLC
```

```
BOUCL7  LDA A U-3,X
        ADCA  R-1,X
        STA A R-1,X
        DEX
        BNE   BOUCL7
                        * décalage à gauche d'un bit du multiplica-
DEC     LDX #$ 2                                          -teur.
BOUCL8  ROL   U-2,X
        DEX
        BNE   BOUCL8
                        * décrémentation du registre de stockage du
                          nombre de bits du multiplicateur,et test
                          de celui-ci.
        DEC B
        BNE         MULT
```

****

.../...



Programme ADDIC2:

Additionne 2 nombres de 16 bits,le byte de poids fort du 2ème opérande étant nul.

LDA B	R+2	charger B par le poids faible du 1er opérande
LDA A	R+1	charger A par le poids fort du 1er opérande.
ADDB ##	M_A	addition de B avec poids faible du 2eme opér--ande.
ADCA ##	$ 00	addition avec retenue de A avec 0.
STA A	R+1	
STA B	R+2	
RTS		

.../...

B-3-c- Programmes fixes, du disque, utilisés:

Programme d'initialisation du disque:

```
                E800  OSLOAD  EQU   *
E800  8E  FF8A          LDS  #STACK   initialise le stack.
E803  8D  1D E822       BSR  FDINIT   initialise PIA,SSDA.
E805  7F  0000          CLR  CURDRV   spécifie le drive 0.
E808  8D  6B E875       BSR  RESTOR   ramène la tête sur piste 00.
E80A  8D  47 E853       BSR  CHKERR   recherche d'erreur
E80C  CE  0017          LDX  #$ 23    spécifie le secteur début.
E80F  DF  03            STX  STRSCT
E811  CE  0002          LDX  #$ 2     spécifie le nombre de secteurs.
E814  DF  03            STX  NUMSCT
E816  CE  0020          LDX  $ BTSTRT spécifie l'adrsse de charge-
E819  DF  06            STX  CURADR   -ment.
E81B  8D  4C E869       BSR  READSC   lecture et chargement.
E81D  8D  34 E853       BSR  CHKERR   contrôle d'erreur
E81F  7E  0020          JMP  BTSTRT   Start Boot.
```

Programme d'initialisation des interfaces:

```
                E822  FDINIT  EQU   *
E822  CE  0000          LDX   $0          Reset PIA
E825  FF  EC02          STX   PIACRA
E828  FF  EC00          STX   PIADRA
                *initialise SSDA-Clear SYNC,Reset TX et RX,
                *disible TIE et RIE,transfer SYNC characters,
                *inhibit SM,select 2 byte transfer,8 bit words
                *transmit SYNC on underflow,disable EIE
                *note:SSDA control et status bits are reversed
                *(left to right...right to left)from SSDA data SHEET.
E82B  CE  DODA          LDX   #$ DODA
E82E  FF  EC04          STX   SSDACR
```

```
                    *SELECT PIA DATA REGISTERS
E831  CE  0404        LDX   #$ 0404
E834  FF  EC02        STX     PIACRA
                    *initialze PIA data output lines-lift head,
                    *Below tack 42,no drive selected,and disable
                    *shift CRC,READ,WRITE,and DATA.
E837  CE  1B02        LDX   #$ 1B02
E83A  FF  EC00        STX     PIACRA
                    *set PIA data direction registers
E83D  CE  0000        LDX   # 0       select direction registers
E840  FF  EC02        STX     PIACRA
E843  CE  1F0F        LDX   #$ 1F0F   set direction registers
E846  FF  EC00        STX     PIADRA
                    *initialize PIA control lines
                    *CA1(Timeout Input)       active on negative
                                              transition.
                    *                         interrupt disabled(for now)
                    *CA2(head step output)    high
                    *CB1(index)               active on pos trans,
                                              intrpt disabled
                    *CB2(timer reset output)  high
E849  CE  3C3E  FDINIT LDX   #$3C3E
E84C  FF  EC02        STX     PIACRA
                    *CLEAR PIA INTERRUPT FLAGS
E84F  FE  EC00        LDX     PIACRA
                    *DONE
E852  39            RTRN    RTS
```

Programme de contrôle d'erreur:

```
                    *ROUTINE TO CHECK FOR POSTED ERROR
          E853      CHKERR EQU   *
E853  24  FD  E852        BCC    RTRN     NO ERROR
E855  8D  03  E85A        BSR    PRNTER   PRINT ERROR NUMBER
E857  7E  F564            JMP    EXBUG    GO TO EXBUG-RESET SP
```

Programme de remise de la tête sur la piste zéro:

```
                        *RESTOR HEAD TO TRACK 0 ENTRY
             E875       RESTOR EQU   *
E875  C6     08                LDAB  #  %00001000
E877         8C                FCB   SKIP2
```

Programme de positionnement de la tête sur le secteur contenu dans STRSCT.

```
                        *SEEK TO TRACK ENTRY
             E878       SEEK   EQU   #
E878  C6     10                LDAB  #  % 00010000
E87A  8C                       FCB   SKIP2
```

Programme d'écriture à partir de la mémoire adressée par CURADR:

```
                        #  WRITE SECTOR ENTRY
             E884       WRITSC EQU   #
E884  C6     80                LDAB  #  % 10000000
```

Programme pour sortir le nombre d'erreurs:

```
                        #  ROUTINE TO PRINT DISK ERROR NUMBER
             E85A       PRNTER EQU   #
E85A  86     45                LDAA  #'E       PRINT E
E85C  8D     08 E866           BSR   XOUTCH
E85E  96     08                LDAA   FDSTAT   PRINT ERROR NUMBER
E860  8D     04 E866           BSR   XOUTCH
E862  86     20                LDAA  #  $ 20   PRINT TWO SPACES
E864  8D     00 E866           BSR   XOUTCH
E866  7E     F018    XOUTCH    JMP    OUTCH
```

****

CHAPITRE III

****

LOGICIEL DE SORTIE DE RESULTATS SUR IMPRIMANTE RAPIDE.

Introduction

1- Caractéristiques de l'imprimante du modèle "Centronics 701"

2- Opération d'impression.

3-Synoptique de base de l'imprimante.

4-Schéma synoptique interne de l'imprimante.

5- Initialisation de l'imprimante.

6- Sélection de l'imprimante.

7- Les lignes de données et de commande.
   a-Les lignes d'entrée.
   b-Les lignes de sortie.

8- Fonction de décodage.

9- Interface de l'imprimante.

10- Quelques sous/prog. ou tables intervenant dans la gestion de l'imprimante.

11- Logiciel.

Organigrammes et programmes.

****

Introduction:

L'imprimante est un organe de sortie ou(périphérique) qui permet de garder une trace du programme et des résultats sous forme de listing.
La mise au point d'imprimantes rapides est une conséquence des grandes vitesses internes atteintes par les ensembles électroniques. Une rentabilité bien comprise exige en effet que les résultats puissent être enregistrés aussi rapidement qu'ils sont produits.
Nous étudions la sortie de résultats d'un système multiprocesseur sur une imprimante rapide du modèle"Centronics 701".

I- Etude de l'imprimante rapide:

1- Caractéristiques de l'imprimante du modèle"701":

Les principales caractéristiques de cette imprimante sont réunies sur le tableau suivant:

-méthode d'impression	par percussion,caractère par caractère,bidirectionnelle.
-vitesse d'impression	60 caractères par seconde
-lignes pleines	26 lignes par minute(132 caractères par ligne)
-lignes courtes	43 lignes par minute(80 caractères par ligne)
-vitesse de transmission parallèle	jusqu'à 75000 caractères par seconde
-donnée d'entrée	standard ASCII 7/8 bits parallèles niveau TTL
-structure des caractères	matrice de points de 5x7; matrice de points de 9x7
-code	US ASCII:64 caractères
-indicateurs	mise sous tension;selection;alerte alarme audio

.../...

-boutons de contrôle	sélection, simple LF, double LF, TOF(top of form)
-contrôle interne	moteur à controle automatique, saut de ligne automatique à chaque retour chariot(CR)
-buffer de caractère	une ligne pleine
-format	horizotalement: 10 caractères/pouce verticalement: 6 lignes/pouce
-papier utilisé	papier à trous d'entrainement sur les côtés: toutes les dimensions inférieures ou égales à 17,3 pouces (soit 439 mm)
-nombre de copies	original+4 copies
-dimensions	épaisseur: 7 pouces(178 mm); longueur: 18 pouces(457 mm); largeur: 24,5 pouces(622 mm)
-poids	27Kg
-tension d'alimentation	115/230V ±'10% à 50/60 Hz
-température:	
en service	de 4,4°C à 37,7°C
emmagasinage	de -37,2°C à 71.1°C
-Humidité:	
en service	de 20% à 90%(N° de condensation)
emmagasinage	de 5% à 95%(N° de condensation)

2-Opération d'impression:

L'imprimante est munie d'une mémoire tampon(buffer de ligne)qui peut recevoir les données, en transmission parallèle, jusqu'à la vitesse de 75000 caractères par seconde.

La tête d'écriture comporte 7 aiguilles en fer, alignées

.../...

verticalement.Chaque aiguille est commandée par un solénoide qui reçoit les impulsions de commande par un générateur de caractère constitué par une ROM.L'écriture des caractères est alors par percussion.Le code de chaque caractère,contenu dans la ROM,est adressé par 7 ou 8 bits,et chaque colonne de la matrice formant le caractère est specifiée par la logique de contrôle d'impression.

Le papier peut être deplacé manuellement ou automatiquement par une des trois commandes:
-line feed (LF)
-vertical tab.(VT)
-form feed (FF)

* Délai d'écriture:

Les solénoides,commandant les aiguilles de la tête d'écriture,ont un temps de réponse qui est de quelques milisecondes.Pour cela il faudrait inclure un délai avant la sortie de chaque caractère. Lorsque l'imprimante est gérée par le MDOS,le délai est inclu dans le temps que met l'unité de disque pour envoyer une ligne d'information vers l'imprimante(on peut le constater par le temps d'arrêt du chariot avant l'écriture de chaque ligne quand l'imprimante est gérée par le MDOS).

3-Synoptique de base de l'imprimante:

(Voir le schéma à la fig.1)

4- Schéma,synoptique,interne de l'imprimante:

(Voir fig.2)

La "701" est une imprimante bidirectionnelle,dont la tête d'impression peut se mouvoir à gauche ou à droite,cherchant le prochain caractère de la nouvelle ligne avant de commencer à écrire.
Les 7 aiguilles de la tête d'impression sont activées par sélection pour former,avec des points,la matrice du caractère spécifié.Leur commande est développée par le générateur de caractère.

.../...

L'imprimante est à base d'un circuit imprimé composé de 4 circuits LSI.

-le circuit de contrôle d'entrée: chip 2009.
-le circuit de synchronisation générale : chip 3011.
-le circuit de logique d'impression: chip 3012.
-le circuit de logique de contrôle et d'impression: chip 3013.

A la mise en marche toute la logique interne est initialisée,cepen--dant l'imprimante reste désélectionnée.Elle peut être sélectionnée à l'envoi du code octal 021 (11 en héxadécimal) ou en appuyant sur le bouton "SELECT".
Quand l'imprimante est sélectionnée et que le MPU présente une donnée sur le bus DATA,il envoie un signal"Data strobe" (DSTA). L'imprimante prend les données,les stocke dans une mémoire vive interne (RAM) et envoie un signal de reconnaissance ($\overline{ACK}$) infor--mant ainsi le MPU qu'elle peut recevoir d'autres informations.

Quand la RAM est entièrement chargée,l'imprimante gènère un signal " BUSY " pour arrêter la transmission des données.A ce moment là, suivant la position du chariot,le compteur d'adresse (qui joue le rôle du registre d'index du MC6800) pointe l'adresse du premier caractère de la nouvelle ligne et s'incrémente ou se décrémente suivant que l'écriture se fasse de gauche à droite ou inverssement.

La RAM peut stocker au maximum 256 caractères.Comme l'impri--mante est dotée d'un"buffer de ligne"pouvant recevoir des caract-tères jusqu'à la vitesse de 75000 caractères par seconde,le temps que mettra l'unité centrale du microordinateur pour remplir la RAM de l'imprimante sera de :

$$\frac{256}{75000} = 33,5 \text{ ms}$$

L'opération d'impression se fait soit,quand la RAM est entièrement chargée soit à la réception du code héxadécimal ØD(CR).
Ainsi,lorsqu'on veut imprimer une ligne dont le nombre de carac--tères est inférieur à 256,il faut terminer celle-ci par un"CR".

Il est à remarquer qu'à chaque réception du code OD (en héxadéc.) l'imprimante fait un saut de ligne (LF) automatiquement.

5- Initilisation de l'imprimante:

Pour que l'imprimante prenne en considération des informations il faut que la logique interne soit initialisée pour la mettre à l'état "prêt"puis sélectionnée (envoyer un 0 sur BUSY) pour permettre la réception des données.
Le signal d'initialisation "PRIM" est généré par le LSI chip 3013 à la suite de:

a/ la mise sous tension
b/ la sélection
c/ l'impression de la fin de la ligne
d/ la réception du code "DELET"(177 en octal,7F en héxadéc.)
e/ la réception de l'impulsion"IP" (Input prime)

6- Sélection de l'imprimante:

Elle se fait en appuyant sur le bouton "SELECT"ou en recevant le code octal 021(11 en héxadéc.) par le bus des données. Elle peut être désélectionnée en appuyant de nouveau sur le bouton "select" ou en recevant le code octal 023(13 en héxadéc.).Une lampe témoin,au dessus du bouton SELECT montre,quand elle est allumée,que l'imprimante est sélectionnée.

7- Les lignes de données et de commande:

a- Les lignes d'entrée:

* 8 lignes parallèles dont 7 les 7 premières représentent les 7 bits du code US ASCII.La 8ième est utilisée comme ligne de comtrôle pour spécifier si l'écriture doit être en caractères gras.
* $\overline{\text{DSTA}}$:"data strobe" active sur le front descendant,est utilisée pour synchroniser la transmission des données du MPU vers l'imprimante.

.../...

* $\overline{IP}$"input prime":active sur le front descendant,est utilisée pour donner l'ordre d'impression.
* "$\overline{BUSY}$":indique que la RAM est complètement chargée ou l'exécution d'une fonction telle que:FF,VT...
*"$\overline{ACK}$":(Acknowledge):ligne de reconnaissance;indique que l'imprimante a chargé une donnée dans la RAM et qu'elle est prête à en recevoir.
*SLCT(select):indique quand elle est à 1,que l'imprimante est sélectionnée.
*$\overline{FAULT}$: signale,quand elle est à l'état bas,qu'il y a erreur:
-imprimante non sélectionnée.
-pas de papier.
-signal vidéo défaillant.

(Voir chronogramme des signaux à la fig.3)

8- Fonction de décodage:

La communication avec l'imprimante se fait par des messages codés en héxadécimal.Dans le tableau suivant nous donnons certains de ces codes,leur fonction et leur action sur l'imprimante.

Fonction	Code héxa.	réponse de l'imprimante
Bell	07	Si l'imprimante est munie d'un avertisseur sonore,ce code déclanche une alarme sonore durant 2 secondes.
LF(line feed)	0A	Avance le papier d'une ligne.
CR(retour chariot)	0D	Déclenche l'opération d'écriture d'une ligne.

.../...

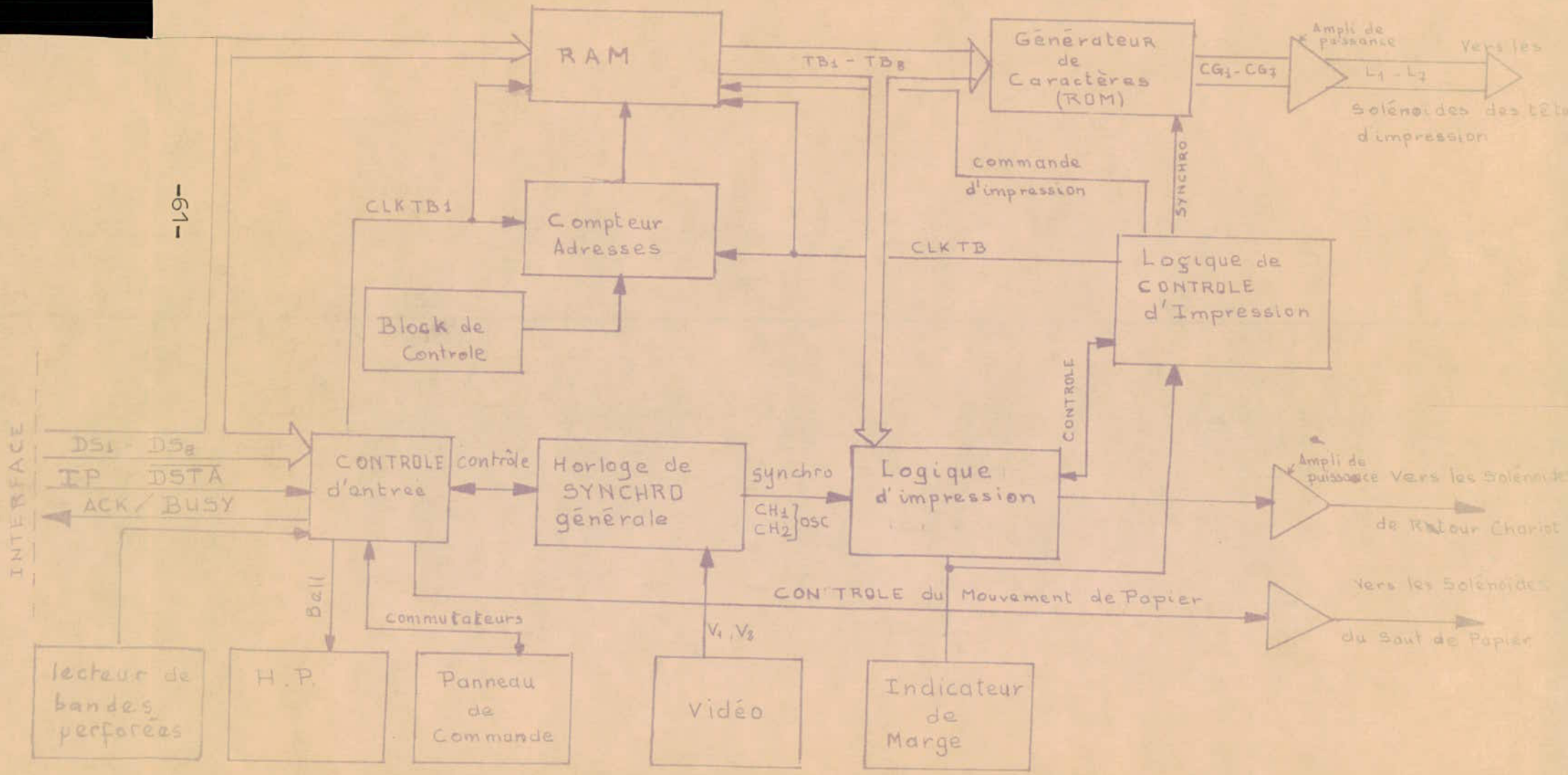


fig. 2 : SCHEMA SYNOPTIQUE interne DE L'IMPRIMANTE

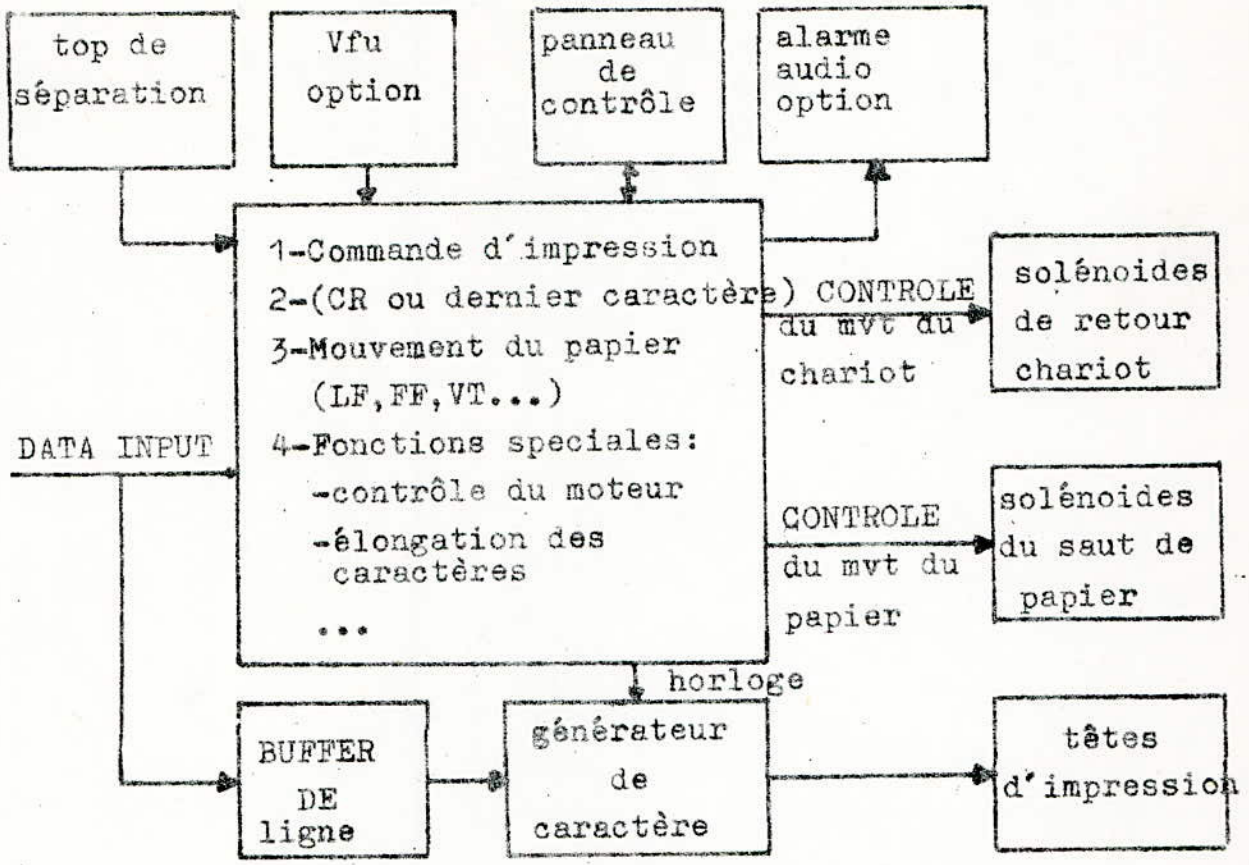


fig.1- Schéma du synoptique de base de l'imprimante.

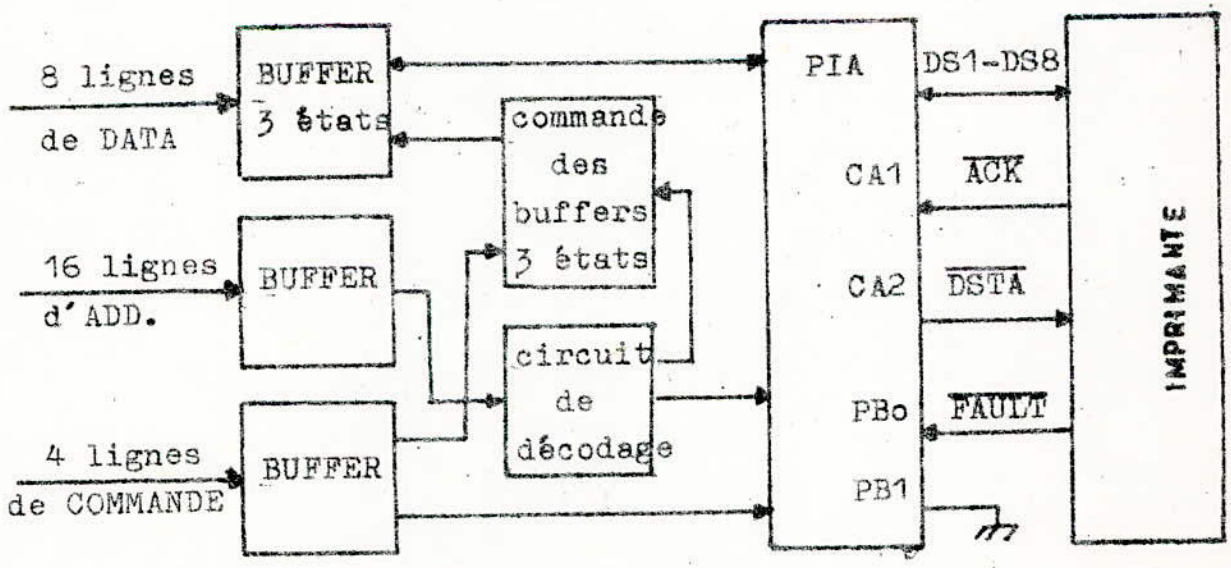


fig.4: schéma synoptique de l'interface.

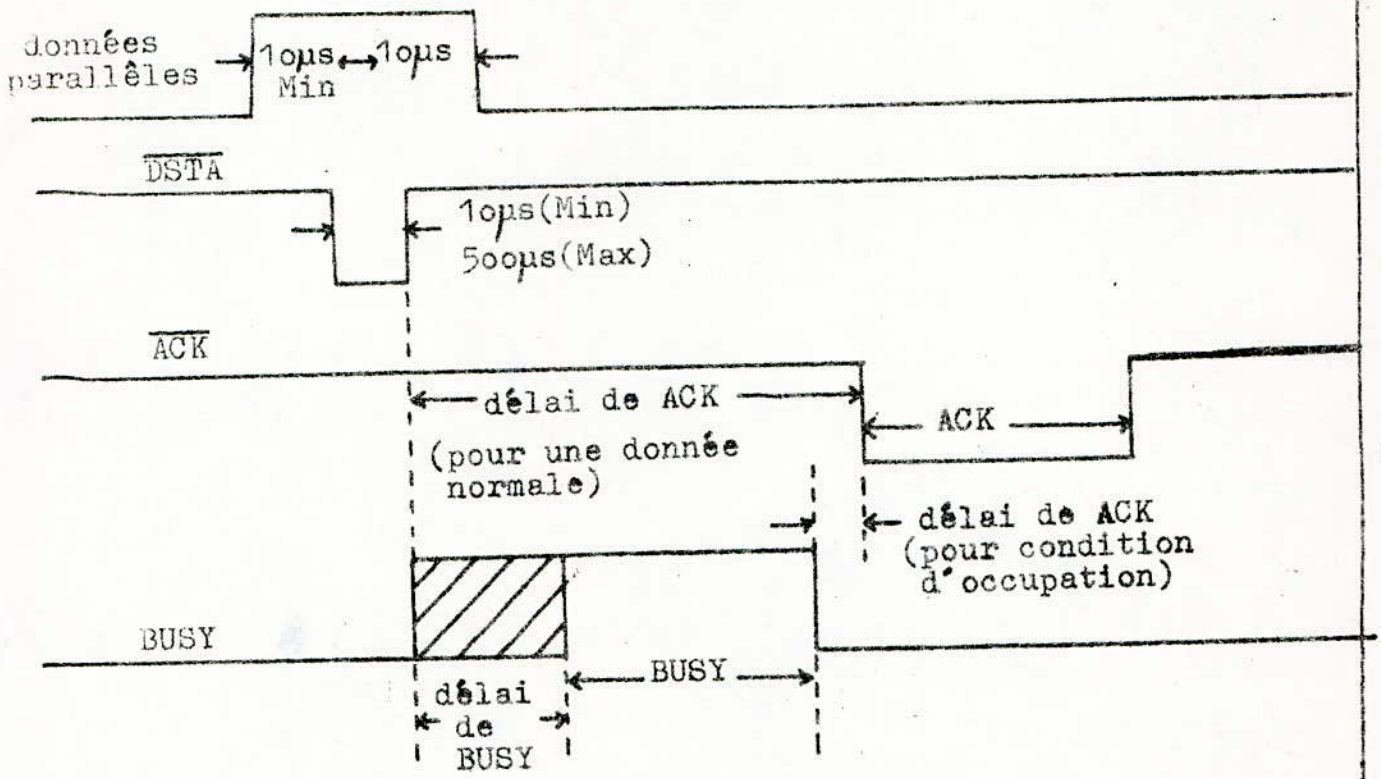


fig: 3 : CHRONOGRAMMES DES SIGNAUX DE DONNEES ET DE COMMANDE

DUREE D'ENTREE DES DONNEES NORMALES

délai de ACK	2,5 à 10,0 µs
ACK	2,5 à 5,0 µs

DUREE DU SIGNAL BUSY POUR LES DIFFERENTES FONCTIONS

délai de ACK	0 à 10,0 µs
délai de BUSY	0 à 1,5 µs
ACK	2,5 à 5,0 µs
-saut de ligne	75 à 105 ms
-saut de table	240 à 270 ms
-saut de page(éjection du papier)	2,07 à 2,11 s
-initialisation de la logique interne	100 à 400 µs
-appel sonore	0
-sélection	100 à 400 µs
-désélection	jusqu'à ce que l'imprimante soit selectionnée de nouveau
-commande d'impression	16,7 ms/char

.../...

Ecriture en caractères gras(elongated chara--ter).	OE	àla réception de ce code toute la ligne sera écrite en caractères gras;OE est actif tant qu'il n'y a pas OD.
Sélection	11	Sélection de l'imprimante(par les "jumpers"optionnels,on peut ne pas initialiser la logique interne après sélection).
Délete	7F	initialise l'imprimante en mettant à zéro la RAM.
Déselection	13	déselectionne l'imprimante.

9- Interface de l'imprimante:

L'imprimante utilise la même interface que le disque.Pour sortir un listing il faudrait donc adresser le PIA de l'imprimante puis l'initialiser.

a-Adressage du PIA de l'imprimante:

```
EC10    DATA
EC11    CNTRL1(contrôle 1)
EC12    STAT
EC13    CNTRL2 (contrôle 2)
```

b- Initialisation du PIA:

```
EBC0    LDX # FF2E
EBC3    STX   DATA
EBC6    LDA A ## # 3C
EBC8    STA A  CNTRL2
EBCB    RTS
```

c- Synoptique de l'interface:(Voir la fig.4)

10- Quelques sous programmes et tables intervenant dans la gestion de l'imprimante:

ADRESSE du S/p	FONCTION	PARAMETRES
F5C2	saut à EXBUG 1.2	
FA14	sortie, sur la visu, sur la TTY d'une table terminée par 04 avec un CR+LF	mettre dans X l'adresse du 1er octet de la table
F9CF	sortie du mot contenu dans l'acc. A sur la visu ou la TTY.	mettre dans X l'adresse du 1er octet de la table
F61B	délai	
FA16	sortie d'une table terminée par 04, sans LF ni CR.	

* Quelques adresses de commande:

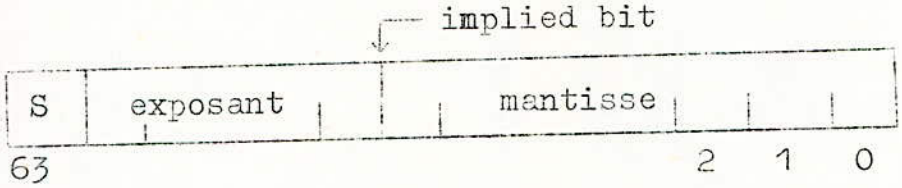
ADRESSE	COMMANDE	FONCTION
A189	.PRT	commande pour sortir un print (PRNT) sur l'imprimante.
A44D	.TCT	commande pour sortir un texte sur l'imprimante.
A422	.MOV	commande pour le transfert du contenu d'une zone mémoire à une autre.

.../...

11- Logiciel de sortie de résultats sur l'imprimante:

Les résultats se présentent de la même façon que pour le disque c'est à dire ,déposés dans la table BUVAR par les Pi.
Dans cette table les valeurs sont écrites en double précision (soit 64 bits flottants).Pour les imprimer,ces valeurs sont d'abord converties en décimal sous le format: ±XXXXXX E ± XX.

a-Rappel sur la représentation des nombres en double précision:



Dans ce type de représentation(en virgule flottante),un nombre N s'écrit de la façon suivante:
$N = M.2^E$ où M est la mantisse et E l'exposant. Le nombre de bits de la mantisse fixe la précision limite des calculs et le nombre de bits de l'exposant détermine l'etendu du codage.La forme dite normalisée,permet de conserver le maximum de bits significatifs.
-le bit 63=S est le signe de la mantisse.
-les bits 52 à 62:ces 11 bits représentent l'exposant codé dont la valeur est supérieure à la valeur réelle de celle du bias exposant.Le bias exposant est égal à : $2^{10}-1=1023$; donc l'exposant codé est: $E+(2^{10}-1)$.
-les bits 0 à 51:représentent la mantisse.Le bit 51 est précédé de l'implied bit égal à 1 et sous entendu.
Un nombre en double précision est représenté par:

$$N = (-1)^S \; 2^{\overbrace{E+(2^{10}-1)}^{\text{exposant codé}}} \; (1,M)$$

(1,M) : virgule

b- Programme:

Les valeurs binaires sont converties en décimal et codées en ASCII.
Comme nous sortons 6 résultats,les valeurs codées ASCII sont

stockées dans une table DETAVA avant d'être imprimées.Cette table contiendra les valeurs prises à un pas donné(une valeur de chacune des variables)soit une ligne d'impression.

Avant d'imprimer les valeurs des Vi(la table DETAVA) on imprime d'abord la table d'identification des variables auxquelles on s'intéresse.Cette table est remplie après le test des octets où sont demandées les Vi et désignée par TAIDVA(table d'identité des variables).Cette table se présente après impression de la manière suivante:

I_1:xx:VARIA A
I_2:xx:VARIA B
I_3:xx:VARIA C
I_4:xx:VARIA D
I_5:xx:VARIA E
I_6:xx:VARIA F

où:
xx est le n° de la Vi demandée et VARIA A,...,F sont les noms donnés aléatoirement à ces variables.

Après impression de la table d'identification est dressée la matrice destinée à contenir les différentes valeurs qui seront prises et dont la première ligne est constituée par la valeur du pas les noms des variables.Cette table sera imprimée ainsi:

h=.... VARIA A VARIA B VARIA C VARIA D VARIA E VARIA F

les autres lignes comprendront le n° du pas est les valeurs des variables:

h=....	VARIA A	VARIA B	VARIA C	VARIA D	VARIA E	VARIA F
00000h	±xxxxxxE±xx	.....	.....	....	....	.....
00001h	±xxxxxxE±xx	.....	.....	....	....	.....
⋮						
01024h	±xxxxxxE±xx	...	....	....	...	....

Ce programme de l'imprimante utilise certains des sous/prog. du

.../...

disque:MULTI2,CALDAD.

Organisation de la page zéro:

0061/ [illegible] M_M utilisé dans le s/p MISFOR
62/ [illegible] S_o 4 octets de sauvegarde des octets de test.
63/ S1
64/ S2
65/ S3
66/ SAVEX4 2 octets de sauvegarde de X
68/ SAVEB sauvegarde de B
69/ SAVEX1 2 octets de sauvegarde de X
6B/ U-3 réservés à MULTI2
6C/ U-2
6D/ U-1
6E/ U
6F/ V
70/ V+1
71/ R-1
72/ R
73/ R+1 résultat de MULTI2
74/ R+2
75/SAVEA sauvegarde de A
76/ SAVEX sauvegarde de X (2 octets)
78/ R+6 résultat de CALDAD
79/ R+7
7A/ RO5R résultat de CON[illegible]BD5 (5 octets)
7B/ RO5R+1
7C/ RO5R+2
7D/ RO5R+3
7E/ RO5R+4
7F/ Po 4 octets utilisés dans MISFOR
80/ P1
81/ P2
82/ P3
83/ Md 2 octets pour le n° du pas demandé

0085/ Mo 4 octets de test
  86/ M1
  87/ M2
  88/ M3
  89/ Mb octet compteur du nombre de decalage et contenant le n° de la Vi demandée.
  8A/ Mp 2 octets compteur du nombre de pas.
  8C/ K10K 10 octets contenant les constantes de conversion
  8D/ K10K+1 de CONBD5.
  8E/ K10K+2
  8F/ K10K+3
  90/ K10K+4
  91/ K10K+5
  92/ K10K+6
  93/ K10K+7
  94/ K10K+8
  95/ K10K+9
  96/ K10K+10

Programmes fixes utilisés par l'imprimante:

Sortie d'une table terminée par 04 sur l'imprimante:

```
A223 A6 00      LDA A 0,X
A225 81 04      CMP A #$04
A227 27 05      BEQ $A22E
A229 BD DA      BSR $A205
A22B 08         INX
A22C 20 F5      BRA $A223
A22E 39         RTS
```

Sortie d'un caractère contenu dans l'accA sur l'imprimante, délai inclu:

```
A205 8D F6   DELSOR   BSR $A1FD saut au s/p DELAI
A207 BD EBCC          JSR $EBCC saut au s/p sortie de caractère
A20A 39               RTS       à partir de l'accA.
```

programme délai:

```
A1FD 36    DELAI   PSH A
A1FE 86 0F         LDA A #$ 0F
A200 4A            DEC A
A201 26 FD         BNE $A200
A203 32            PUL A
A204 39            RTS
```

programme de sortie d'un caractère contenu dans l'acc.A:

```
EBCC B7 EC10          STA A   DATA
EBCF B6 EC10          LDA A   DATA
EBD2 37       LIST3   PSH B
EBD3 F6 EC12          LDA B   STAT
EBD6 C4 03            AND B   #3
EBD8 5A               DEC B
EBD9 33               PUL B
EBDA 26 06 EBE2       BNE     ERROR PAS DE PAPIER OU NON SELECTIONNEE.
EBDC 7D EC11          TST     CNTRL1
EBDF 2A F1 EBD2       BPL     LIST3
EBE1 39               RTS

EBE2 0D    ERROR SEC
EBE3 39    RTS
```

Organisation de la table d'identité des variables(TAIDVA):

0E	spécifie l'écriture en caractères gras.
20	4 espaces
20	
20	
20	
49	code de I1:xx:VARIA A
31	
3A	
-	NO1N
-	
3A	
56	
41	
52	
49	
41	
20	
42	
0D	CR
0A	LF
0E	
20	
20	
20	
20	
49	code de I2:xx:VARIA B
32	
3A	
-	NO2N
-	
3A	
56	
41	
52	
49	
41	

20	
42	
0D	
0A	
0E	
20	
20	
20	
20	
49	code de I3:xx:VARIA C
33	
3A	
-	NO3N
-	
3A	
56	
41	
52	
49	
41	
20	
43	
0D	
0A	
0E	
20	
20	
20	
20	
49	code de I4:xx:VARIA D
34	
3A	

```
-    NO4N
-
3A
56
41
52
49
41
20
44
0D
0A
0E
20
20
20
20
49    code de I5:xx:VARIA E
35
3A
-    NO5N
-
3A
56
41
52
49
41
20
45
0D
0A
0E
20
20
```

```
20
20
49    code de I6:xx:VARIA F
36
3A
-    NO6N
-
3A
56
41
52
49
41
20
46
0D
0A
04
```

Organisation de la table nominative des variables(1ère ligne de la matrice)cette table est désignée par TANOVA:

0E	
20	4 espaces
.	
.	
20	
68	pas h=0,1s
3D	
30	
2C	
31	
73	
20	8 espaces
.	
.	
.	
20	
56	code VARIA A
41	
52	
49	
41	
20	
41	
20	12 espaces
.	
.	
20	
56	code VARIA B
41	
52	
49	
41	
20	
42	
20	12 espaces
.	
.	
20	

56	code VARIA C
41	
52	
49	
41	
20	
43	
20	12 espaces
.	
.	
20	
56	code VARIA D
41	
52	
49	
41	
20	
44	
20	12 espaces
.	
.	
20	
56	code VARIA E
41	
52	
49	
41	
20	
45	
20	12 espaces
.	
.	
20	
56	code VARIA F
41	
52	
49	

41	
20	
46	
20	12 espaces
.	
.	
20	
0D	CR + LF
0A	
04	fin de table

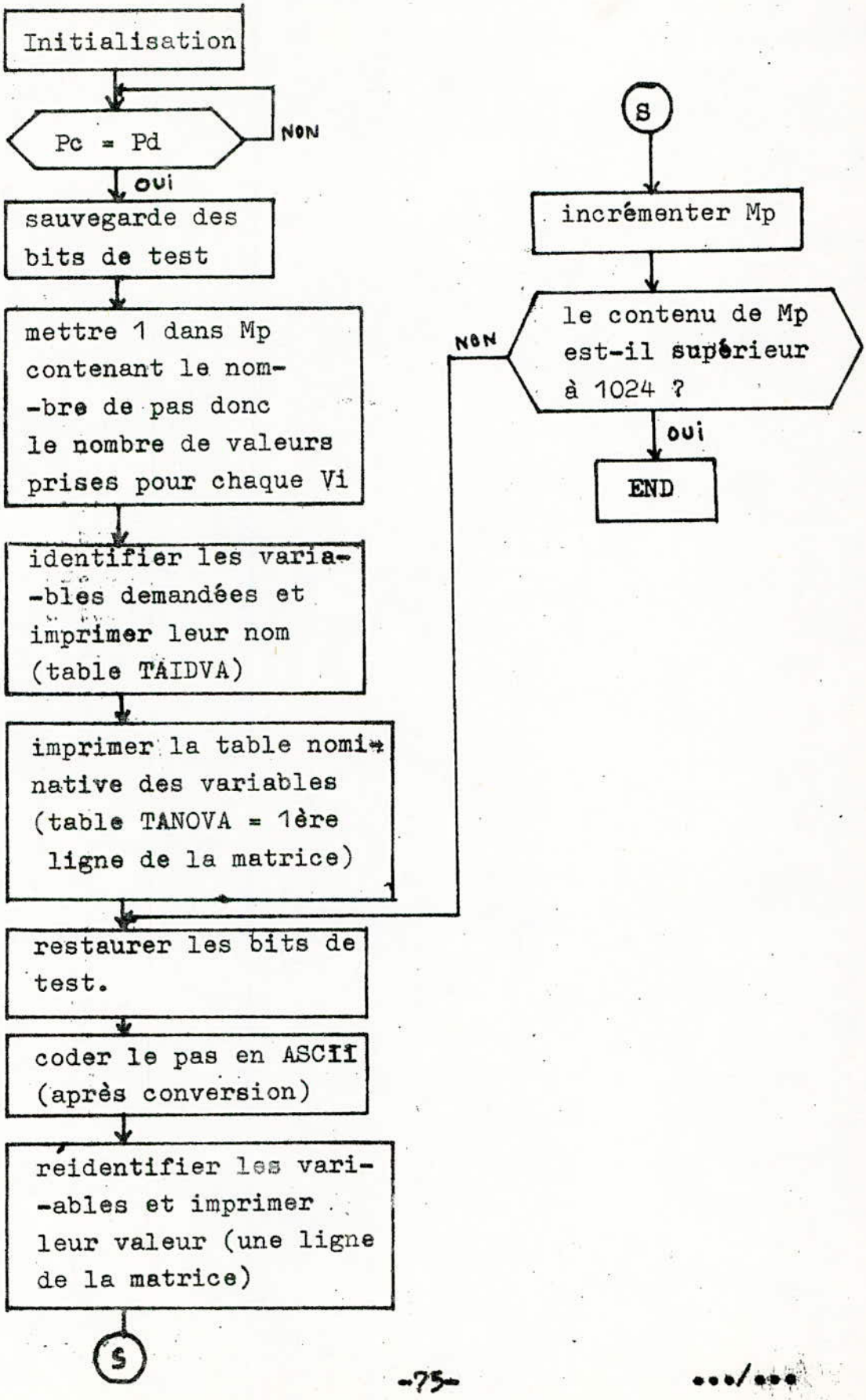
Organisation de la table contenant à un pas donné une valeur de chacune des Vi demandées, le n° du pas. Cette table représente une ligne de la matrice, elle est designée par DETAVA.

```
      20  4 espaces
      .
      20
      .   P05P(pas)
      .   5 caractères
      .
      .
      .
      68
      20  6 espaces
      .
      20
      .   SIGNM1:signe
      2C  de la 1ère
M01M  .   mantisse codé
      .
      .
      .
      .
      45
      .   SIGNE1:signe
      .   exposant codé
      .
      20  8 espaces
      .
      20
      .   SIGNM2
      2C
M02M  .   valeur de la
      .   mantisse sur
      .   5 caractères
      .
      .
      45
```

```
      .   SIGNE2
      .   valeur de
      .   l'exposant
      20  8 espaces
      .
      20
      .   SIGNM3
      2C
      .   M03M
      .
      .
      .
      .
      45
      .   SIGNE3
      .
      .
      20  8 espaces
      .
      20
      .   SIGNM4
      2C
      .   M04M
      .
      .
      .
      .
      45
      .   SIGNE4
      .
      .
      20  8 espaces
      .
      20
```

```
      .   SIGNM5
      2C
      .   M05M
      .
      .
      .
      .
      45
      .   SIGNE5
      .
      .
      20  8 espaces
      .
      20
.     SIGNM6
      2C
      .   M06M
      .
      .
      .
      .
      45
      .   SIGNE6
      .
      .
      20  10 espaces
      .
      20
      0D  CR+LF
      0A
      04  fin de table
```

Organigramme principal:



.../...

Programme principal:

```
        *Mo     remplissage en immédiat des octets de test par
        *M1     l'utilisateur.
        *M2
        *M3
        LDA A   Mo      sauvegarde des bits de test
        STA A   So
        LDA A   M1
        STA A   S1
        LDA A   M2
        STA A   S2
        LDA A   M3
        STA A   S3
        LDX #Ø  NO1N    identification des variables demandées;
        LDA B #  Ø8     et codage de celles-ci dans la table
ROTATØ  ROR     Mo      TAIDVA.
        BCC     TOTO    test des Vi demandées au 1er octet.
        STX     SAVEX4
        STA B   SAVEB
        LDA B   Mb
        CLR A
        JSR     CONBD5  saut au s/p conversion binaire/décimal
        LDA B   SAVEB
        LDX     SAVEX4
        LDA A   RO5R+3  stockage du résultat de CONBD5 à la position
        STA A   0,X     pointée par X.
        LDA A   RO5R+4
        STA A   1,X
        CPX     NO6N    test si l'identification des 6 Vi est
        BEQ     IMIDVA  terminé  auquel cas on imprime TAIDVA.
        LDA A #Ø 14     poimter par X l'adresse où sera codé le
DEC1    INX             n° de la prochaine Vi.
        DEC A
        BNE     DEC1
TOTO    INC     Mb      test si le test du 1er octet est terminé.
        CMP B   Mb
        BNE     ROTATØ
```

```
        LDA B #$10
ROTAT1  ROR     M1          test des Vi demandées au 2ème octet.
        BCC     NONO
        STX     SAVEX4
        STA B   SAVEB
        CLR A
        LDA B   Mb
        JSR     CONBD5
        LDA B   SAVEB
        LDX     SAVEX4
        LDA A   RO5R+3
        STA A   0,X
        LDA A   RO5R+ 4
        STA A   1,X
        CPX     NO6N
        BEQ     IMIDVA
        LDA A #$14
DEC2    INX
        DEC A
        BNE     DEC2
NONO    INC     Mb          tester si le test du 2ème octet est
        CMP B   Mb          terminé.
        BNE     ROTAT1
        LDA B #$18          test du 3ème octet
ROTAT2  ROR     M2
        BCC     MIDO
        STX     SAVEX4
        STA B   SAVEB
        CLR A
        LDA B   MB
        JSR     CONBD5
        LDA B   SAVEB
        LDX     SAVEX4
        LDA A   RO5R+3
        STA A   0,X
        LDA A   RO5R+4
        STA A   1,X
        CPX     NO6N
        BEQ     IMIDVA
        LDA A #$14
DEC3    INX
        DEC A
        BNE     DEC3
MIDO    INC     Mb
        CMP B   Mb
        BNE     ROTAT2
```

```
LDA B #$20          test du 4ème octet.
ROTAT3  ROR    M3
        BCC    MIMO
        STX    SAVEX4
        STA B  SAVEB
        CLR A
        LDA B  Mb
        JSR    CONBD5
        LDA B  SAVEB
        LDX    SAVEX4
        LDA A  RO5R+3
        STA A  0,X
        LDA A  RO5R+4
        STA A  1,X
        CPX    NO6N
        BEQ    IMIDVA
        LDA A #$14
DEC4    INX
        DEC A
        BNE    DEC4
MIMO    INC    Mb          voir si le test du 4ème octet est
        CMP B  Mb          terminé.
        BNE    ROTAT3

IMIDVA  LDX    TAIDVA      imprimer la table d'identité des
  CODE  LDA A #$00          variables:TAIDVA.
        JSR    OUTCH
        BRA    CODE

        LDA A  So          restaurer les bits de test.
        STA A  Mo
        LDA A  S1
        STA A  M1
        LDA A  S2
        STA A  M2
        LDA A  S3
        STA A  M3

 CODE1  LDX    TANOVA      imprimer la table nominative des
        LDA A #$00          variables soit la 1ère ligne de la
        JSR    OUTCH
        BRA    CODE1       matrice.

COMPAS  LDX    Md          comparaison du pas de calcul au pas
 BOUCL  DEX                 demandé.
        CPX    Mc
        BNE    BOUCL

CODPAS  LDX #  $01          initialiser le compteur de pas à 1;
        STX    Mp

  STOCK LDA B  Mp(poids faible)    coder le n° du pas,et le
        LDA A  Mp(poids fort)      stocker dans la table à
        JSR    CONBD5              imprimer:DETAVA.
        LDA A  RO5R
        STA A  PO5P
```

```
         LDA A    RO5R+1
         STA A    PO5P+1
         LDA A    RO5R+2
         STA A    PO5P+2
         LDA A    RO5R+3
         STA A    PO5P+3
         LDA A    RO5R+4
         STA A    PO5P+4
TEVADE   LDX      SIGNM1     test des Vi demandées pour coder leur
         LDA B    #$08       valeur dans la table à imprimer:DETAVA.
ROTAT4   ROR      Mo         test du 1er octet.
         BCC      TOTO1
         JSR      MISFOR     saut au s/p de mise en forme des Vi.
         JSR      CODASC     saut au S/p qui convertit et code Vi.
         CPX      SIGN6      voir si les 6 valeurs sont prises,
         BEQ      IMPRIM     auquel cas on imprime la table DETAVA.
         LDA A    #$13
DEC5     INX
         DEC A
         BNE      DEC5
TOTO1    INC      Mb         voir si le test du 1er octet est fini.
         CMP B    Mb
         BNE      ROTAT4
ROTAT5   LDA B    #$10
         ROR      M1         test du 2ème octet.
         BCC      NONO1
         JSR      MISFOR
         JSR      CODASC
         CPX      SIGN6
         BEQ      IMPRIM
         LDA A    #$13
DEC6     INX
         DEC A
         BNE      DEC6
NONO1    INC      Mb         est ce que le test du 2ème octet est fini.
         CMP B    Mb
         BNE      ROTAT5
         LDA B    #$18       test du 3ème octet.
ROTAT6   ROR      M2
         BCC      MIDO1
         JSR      MISFOR
         JSR      CODASC
         CPX      SIGN6
         BEQ      IMPRIM
         LDA A    #$13
DEC7     INX
         DEC A
         BNE      DEC7
MIDO1    INC      Mb         est ce que le test du 3ème octet est fini.
         CMP B    Mb
         BNE      ROTAT6
```

```
         LDA B #$20        test du 4ème octet.
ROTAT7   ROR     M3
         BCC     MIMO1
         JSR     MISFOR
         JSR     CODASC
         CPX     SIGNM6
         BEQ     IMPRIM
         LDA A #$13
DEC8     INX
         DEC A
         BNE     DEC8

MIMO1    INC     Mb        tester si la rotation du 4ème octet est
         CMP B   Mb        terminée.
         BNE     ROTAT7

IMPRIM   LDX     DETAVA    imprimer la table DETAVA=une ligne de
BOUCLF   LDA A #$00        la matrice.
         JSR     OUTCH     saut au s/p de sortie de caractère.
         BRA     BOUCLF

         LDX     Mp        tester si l'on a sorti 1024 valeurs
         CPX #$0400        sinon incrementer le compteur de pas
         BEQ               et remplir de nouveau DETAVA,

         BEQ     END

         INX

         STX     Mp

         JMP     STOCK     saut à l'adresse STOCK

         END
```

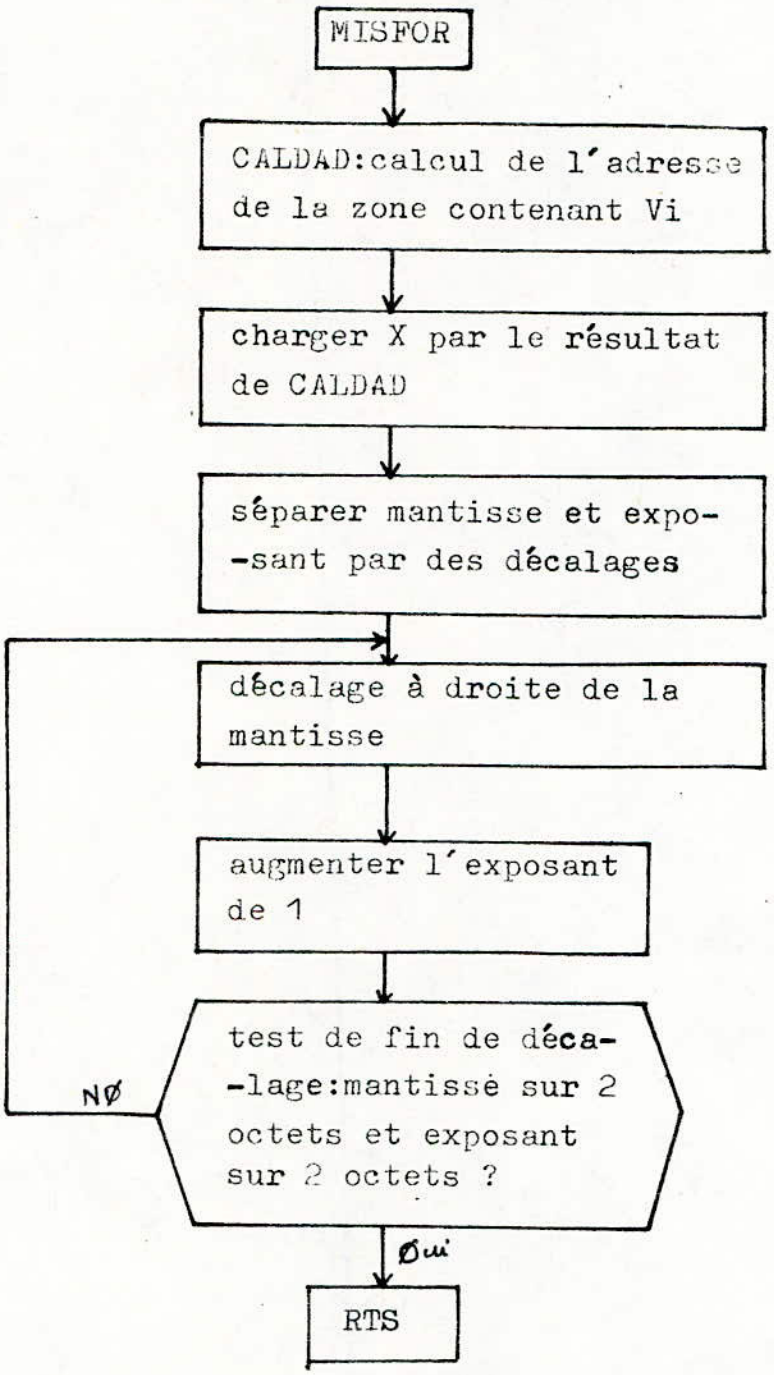
SOUS PROGRAMME MISFOR:

Ce sous/prog. réduit la valeur, écrite sur 64 bits flottants, sur 4 octets(exposant 2 octets et mantisse 2 octets)en augmentant l'exposant. L'exposant se trouve ensuite en position Po et P1, la mantisse en position P2 et P3.

```
         STX     SAVEX4    sauvegarder X
         STA B   SAVEB     sauvegarder X
         JSR     CALDAD    saut au calcul de l'adresse de Vi.
         LDX     R+6       pointer l'adresse de Vi.
         CLR B
         LDA A   1,X
         LSR A
         BCC     BOUCL1
         ADD B #$01

BOUCL1   LSR A
         BCC     BOUCL2
```

```
            ADD B  #$02
BOUCL2      LSR A
            BCC    BOUCL3
            ADD B  #$04
BOUCL3      LSR A
            BCC    BOUCL4
            ADD B  #$08
BOUCL4      ASLA
            ASLA
            ASLA
            ASLA
            STA A  1,X
            STA B  M
                   M
            CLC
            LDA B  #$04
BOUCL5      ROR    M
                   M
            ROR    2,X
            ROR    3,X
            ROR    4,X
            ROR    5,X
            ROR    6,X
            ROR    7,X
            DEC B
            BNE    BOUCL5

            LDA B  #$08
BOUCL6      ROR    2,X
            ROR    3,X
            ROR    4,X
            ROR    5,X
            ROR    6,X
            ROR    7,X
            DEC B
            BNE    BOUCL6

            LDA B  #$08
BOUCL7      ROR    3,X
            ROR    4,X
            ROR    5,X
            ROR    6,X
            ROR    7,X
            DEC B
            BNE    BOUCL7

BOUCL8      LDA B  #$08
            ROR    4,X
            ROR    5,X
            ROR    6,X
            ROR    7,X
            DEC B
            BNE    BOUCL8

            LDA B  #$08
BOUCL9      ROR    5,X
            ROR    6,X
            ROR    7,X
            DEC B
            BNE    BOUCL9
            BCC    MIMI
            LDA B  7,X
            LDA A  6,X
           ADD B  #$01
            ADD

            ADC A  #$00
            STA A  P2
            STA B  P3

MIMI        LDA B  1,X
            ADD B  #$ 24
            STA B  P1
            LDA B  0,X
            STA B  Po
            LDA B  6,X
            STA B  P2
            LDA B  7,X
            STA B  P3

            LDA B  SAVEB   restaurer B
            LDX    SAVEX4  restaurer X

            RTS
```

Organigramme MISFOR:



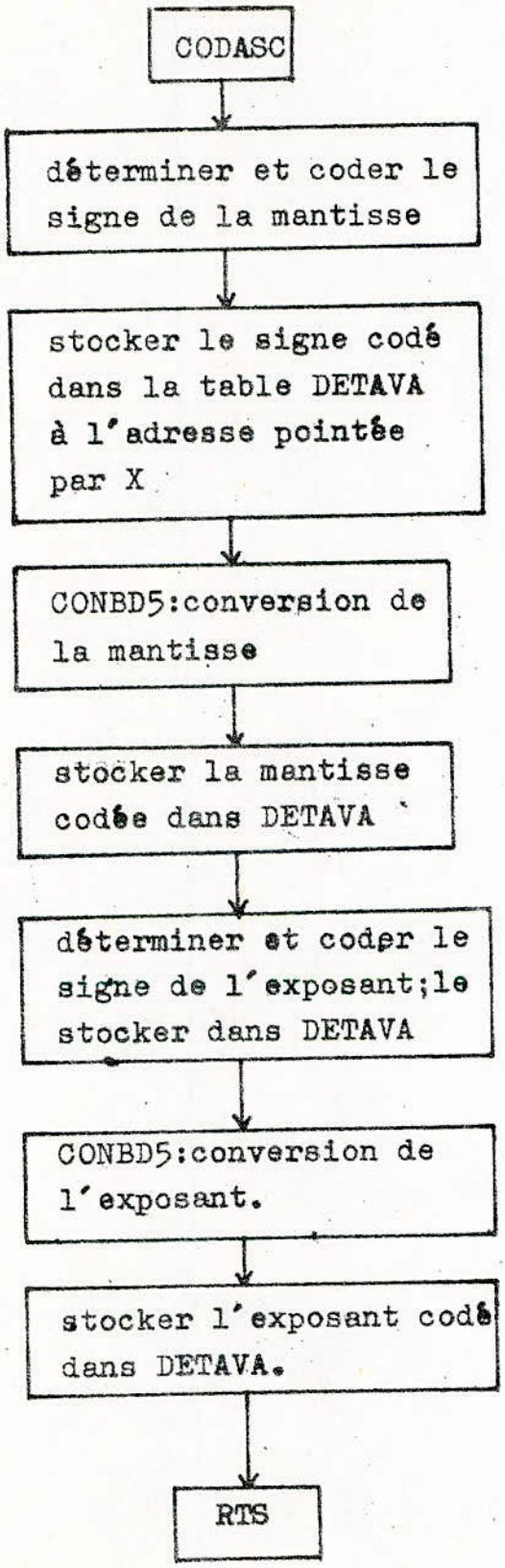
SOUS PROGRAMME CODASC:

Ce programme permet de convertir la valeur binaire en décimal.Les Vi étant en virgule flottante,on détermine d'abord le signe de l'exposant et on le code ensuite sa valeur qui est aussi codée; enfin la mantisse et son signe codés.Les signes et les valeurs codés en ASCII sont rangés dans la table à imprimer:DETAVA.

```
STA B   SAVEB       sauvegarder B
SIGNEM  ASL   Po      déterminer le signe de la mantisse et
        BCC   POSITI  le coder.
        LDA B #$2D    coder le signe -.
        STA B 0,X
        BRA   CONMAN
POSITI  LDA B #$2B    coder le signe +.
        STA B 0,X
CONMAN  LDA B P3      conversion de la mantisse.
        LDA A P2
        JSR   CONBD5  saut au s/p de conversion binaire/déc.
        LDA A R05R    ranger le résultat de CONBD5 dans
        STA A 2,X     DETAVA.
        LDA A R05R+1
        STA A 3,X
        LDA A R05R+2
        STA A 4,X
        LDA A R05R+3
        STA A 5,X
        LDA A R05R+4
        STA A 6,X
SIGNEE  CLC           déterminer le signe de l'exposant et le
                      coder.
        LSR   Po
        LDA B P1      mise en forme unbiased de l'exposant.
        LDA A Po
        SUB B #$ F1   Exp - (1023)10
        SBC A #$ 03
```

```
                BCS     NEGATI
STA A SAVEA →   LDA A   #$2B        coder le signe +.
                STA A   8,X
                BRA     MOMO
NEGATI          LDA A   #$2D        coder le signe - .
                STA A   8,X
LDA A SAVEA →
MOMO            COM A               complément à 1 du MSB.
                NEG B               comp.à 2 du LSB.
                BCS     CONEXP
                ADD A   #$01
CONEXP          JSR     CONBD5      conversion de l'exposant.
                LDA A   R05R+3
                STA A   9,X
                LDA A   R05R+4
                STA A   1, A,X
                LDA B       SAVEB   restaurer B.
                RTS         retour de sous programme.
```

Organigramme CODASC:

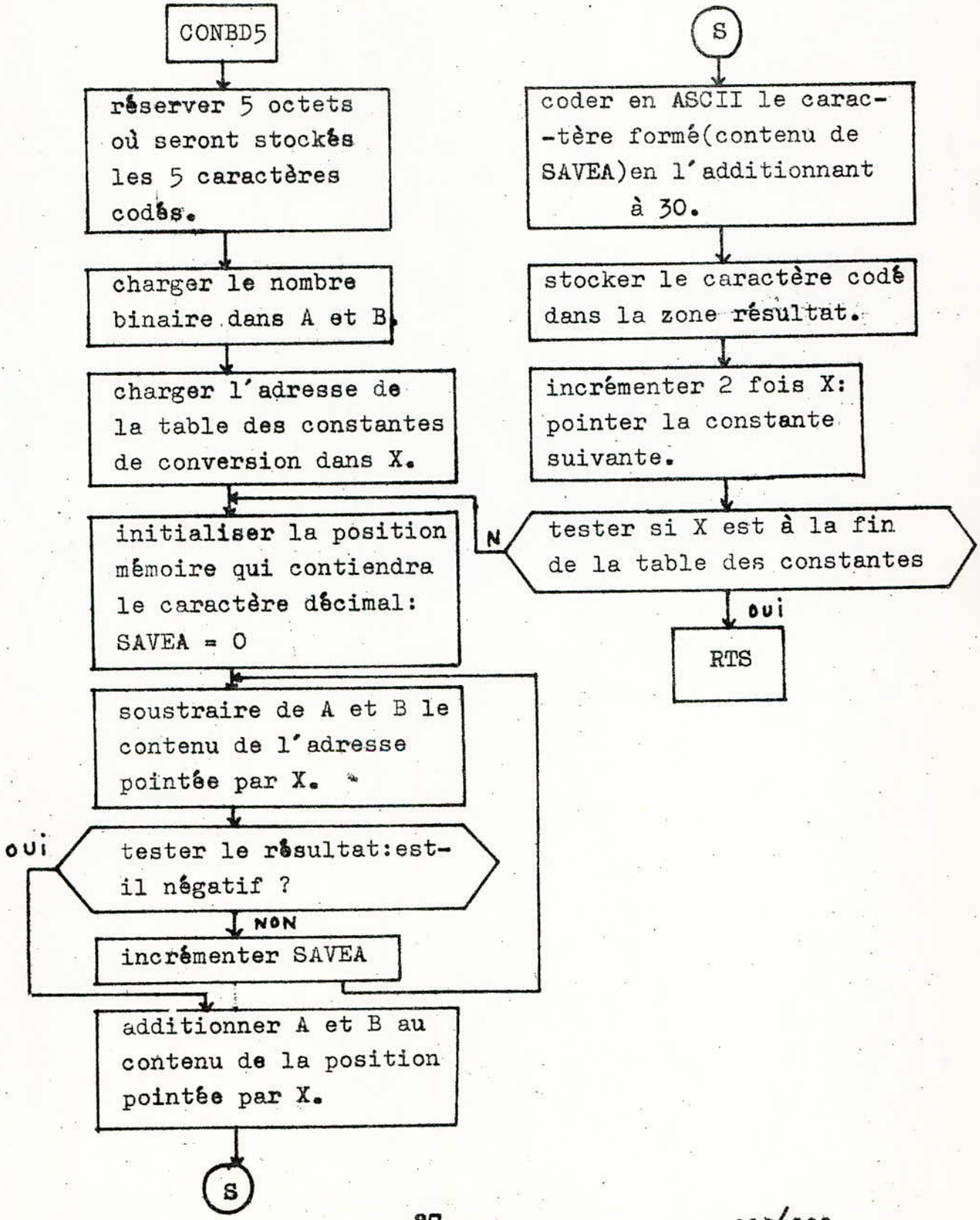


Programme CONBD5:

Ce sous programme convertit un nombre binaire de 16 bits, écrit dans A et B, en un nombre décimal de 5 caractères en utilisant les cons--tantes de conversion qui se trouve dans les positions K10K à K10K+9. Le résultat (5 caractères décimaux) est stockés dans les positions RO5R à RO5R+4.

```
        LDX #   RO5R        pointer la zone résultat.
        STX     SAVEX
        LDX #   K10K        pointer la table des constantes.
CVDEC1  CLR     SAVEA       initialiser la mémoire des caract.déc.
CVDEC2  SUB B   1,X
        SBC A   0,X
        BCS     CVDEC5      bit carry à 1.
        INC     SAVEA       incrémenter le caractère à former.
        BRA     CVDEC2
CVDEC5  ADD B   1,X         restaurer le résultat partiel.
        ADC A   0,X
        PSH A
        STX     SAVEX1
        LDX     SAVEX       stockage du caractère formé.
        LDA A   SAVEA
        ADD A #$30          coder le caractère en ASCII.
        STA A   0,X
        PUL A               restaurer A
        INX
        STX SAVEX
        LDX     SAVEX1      pointer table des constantes.
        INX
        INX
        CPX #K10K+10
        BNE     CVDEC1      branchement à l'adresse CVDEC1.
        RTS                 retour de sous programme.
```

Organigramme de CONBD5:



CHAPITRE IV

LOGICIEL DE SORTIE DE RESULTATS SUR ENREGISTREUR POTENTIOMETRIQUE:

Introduction.

A-Enregistreur potentiométrique.

A-1-Description.

A-2-Caracteristiques techniques de l'enregistreur 6 voies:

A-3-Système d'inscription.

A-4-Déroulement du papier.

B-Convertisseur numérique analogique.

B-1-Définition.

B-2-Différentes familles de CNA(Convertisseurs numérique/ analogique).

a-Le CNA parallèle.

b-Temps de comversion.

C-Logiciel.

-organigrammes.

-programmes.

Introduction:

Le tracé de courbe est particulièrement intéressant.Il illustre,à l'utilisateur qui suit la simulation,l'aspect dyna-mique du processus aussitôt les résultats calculés.Ainsi l'uti-lisateur a l'impression de suivre le vrai processus physique.
On s'est intéressé à l'évolution de 6 paramètres(6 variables).
Cependant la table traçante à 2 plumes ne permettant pas de tracer les 6 courbes simultanément,on a recours à l'enregistreur potentio-métrique à 6 voies.

A-1-Enregistreur potentiométriquex à 6 voies:

L'enregistreur utilisé est l'enregistreur du modèle 316 de BRYANS SOUTHERN INSTRUMENTS LIMITED(BSIL).C'est un appareil très performant,équipé de 6 plumes distantes de deux(2) millimètres l'une de l'autre et présente 16 vitesses de déroulement du papier.
Il est commandé par le CPU de MOTOROLA qui gère les échanges avec les interfaces analogiques.
Cet appareil quoique performant présente un défaut qui est le décalage entre les tracés d'un pas croissant rendant difficile l'interprétation des courbes dépendantes du temps.Ainsi faudrait-il se rappeler de combien de millimètres tel diagramme est séparé du scripteur de référence;tout en sachant que ce retard varie avec la vitesse de défilement du papier.
L'appareil possède plusieurs commandes controlées à partir du MPU. Parmi ces commandes nous pouvoms utiliser celle du lève plume et celle de l'avance par pas du déroulement du papier.
On distingue 3 sortes de tracés de courbes: avec cet appareil:

-tracé par point.
-tracé par tirets.
-tracé continu.

Dans ce chapitre on élabore un programme qni permet le tracé point par point.

.../...

A-2-Caractéristiques techniques:

L'enregistreur potentiométrique multivoies du modèle 316 de BRYANS SOUTHERN INSTRUMENTS LIMITED(BSIL)possède la fiche technique:

-Nombre de canaux: 6
-hauteur hors-tout: 262 mm
-dimensions hors-tout: 424x262x320
-poids: 15 Kg
-système d'écriture: plume à pointe fibre.
-écart entre les plumes: 2mm
-largeur de piste: 250mm+2%(répartis de part et d'autre)
-largeur total du papier:290mm
-temps de traversée de l'échelle: o,3s(soit 1oo cm/s).
-bande passante: 1,7 Hz (à -3 dB)
-fidélité: 0,15% de la pleine échelle(0,4mm)
-seuil d'écriture: 0,15% de la pleine échelle(0,4mm)
-linéarité: 0,1% de la pleine échelle(0,25mm)
-16 vitesses de deroulement du papier:
60,40,20,10,8,4,2,1 cm/mn et cm/h.
-entrainement du papier:
.moteur pas à pas(résolution 0,066 mm/pas)
.télécommande du déroulement par impulsion TTL 0,75 V.
.télécommande marche-arrêt du papier par court circuit.
.télécommande marche avant-arrière.

-sortie des amplificateurs : 0 à 1 V à l'arrière du chassis.
-cadrage de zéro: sur toute l'étendue de l'échelle.
-alimentation secteur: 220V/115V (+20% 115V/220v (+20%;-10%)
-fréquence secteur: 50-60 Hz

A-3- Système d'inscription:

Chaque scripteur,du système,d'inscription,est disposé sur une barre métallique perpendiculaire au défilement du papier et séparé,de son voisin,de 2mm.Ceci permet d'avoir,sur un même rouleau d'enregistrement,6 systèmes d'asservissement potentiométriques pouvant se déplacer sur la totalité de la largeur du

.../...

papier et permettant une grande dynamique de mesure.

A-4-Déroulement du papier:

Le déroulement du papier influe sur la synchronisation des tracés de courbe sur les 6 voies.Pour résoudre le problème nous avons établi un programme en conséquence c'est à dire permettant le rattrapage du zéro,et ceci de la manière suivante:
Chaque scripteur ne doit inscrire son point que lorsque le papier aura défilé de 2mm(distance minimale séparant 2 scripteurs voisins) après que le scripteur précédent ait inscrit le sien.
Cette solution consiste à retarder le signal appliqué au servo-mécanisme d'inscription d'un temps proportionnel à la vitesse de défilement du papier.Chaque plume attendra le temps que met le papier pour venir de la plume précédente à celle-ci.
Ainsi si la plume n°1 est prise comme référence,la plume n°2 attendra un temps T tel que:

$$T = \frac{\text{distance séparant les 2 plumes}}{\text{vitesse de défilement}} = \frac{X}{V}$$

La plume n°3 attendra 2T,la plume n°4 attendra 3T etc...
Autrement dit,chaque plume attend un temps T après que la précé-dente ait inscrit son point.

B-Convertisseur numérique/analogique:

Le tracé des courbes à partir de valeurs numériques n'est possible qu'après un traitement préliminaire qui consiste en une conversion numerique/analogique.Pour une grande rapidité de conver-sion,un convertisseur numérique/analogique est utilisé.La conver-sion N/A se fait par l'envoi d'une suite de nombres au convertis-seur qui les transforme en une suite d'échantillons d'amplitude discrète.

B-1-Définition:

Un convertisseur numérique analogique (CNA) est un dispositif qui reçoit une information numérique sous forme d'un mot de n bits et la transforme en un signal analogique. Un CNA fait correspondre à l'une des 2^n combinaisons binaires possibles à l'entrée, une parmi 2^n tensions discrètes obtenues à partir d'une tension de référence Uref. Le nombre N à convertir s'exprime en fonction des puissances entières de 2 comme suit:

$$N = d1.2^{n-1} + d2.2^{n-2} + \ldots + dn.2^{0} .$$

$$U = N.Uref.$$

$$U = d1.Uref.2^{n-1} + d2.Uref.2^{n-2} + \ldots + dn.Uref.2^{0} .$$

Si N est inférieur à l'unité on exprime:

$$N' = \frac{N}{2^n}$$

$$U = N'.Uref = N.\frac{Uref}{2^n}$$

$$U = Uref.\sum_{i=1}^{n} \frac{di}{2^i}$$

On définit ainsi une fonction de transfert pour le CNA.
La tension de référence peut être constante; on a alors un CNA classique, ou variable on parle alors d'un CNA multiplieur.

B-2- Différentes familles de CNA:

On peut distinguer 2 sortes de convertisseurs:
-CNA indirects.
-CNA directs.
Les premiers, comme leur nom l'indique, utilisent une référence intermédiaire qui est une grandeur analogique par exemple le temps ou une densité de probabilité.
Les CNA directs font correspondre au mot binaire le signal de sortie analogique sans aucun intermédiaire.
Les CNA directs, eux mêmes, se subdivisent en CNA parallèles et CNA séquenciels suivant que la conversion des bits s'effectue en même temps ou successivement.

Les CNA parallèles sont les plus utilisés vue leur rapidité. Notre étude se limitera à ce dernier type de CNA.

a- Le CNA parallèle:

Nous avons vu que la fonction de transfert utilisée pour la conversion directe est la suivante:

$$U = Uref. \sum_{i=1}^{n} \frac{di}{2^i} \qquad \text{avec} \quad di = \begin{cases} 0 \\ \text{ou} \\ 1 \end{cases}$$

et Uref = tension de référence.

A cette fonction de transfert répond le schéma de principe, du CNA parallèle, qui comprend:

-une grandeur de référence donnée par le constructeur: Uref.

-un système de pondération: multiplication par les coefficients $\frac{1}{2^i}$.

-une commande numérique: multiplication par les di.

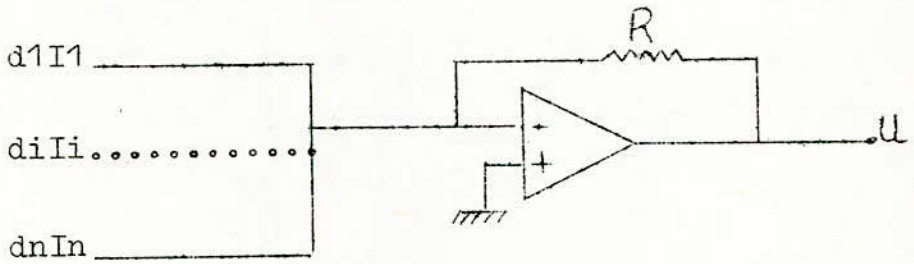
-une sommation de ces différents signaux.

-une transformation éventuelle de courant en tension ou l'inverse. Généralement il est plus facile d'additionner des courants; ainsi on multiplie et on divise par une résistance comme suit:

$$U = Uref\left(\frac{d1}{2R} + \frac{d2}{2^2 R} + \ldots + \frac{dn}{2^n R}\right)$$

$$= R(I1d1 + I2d2 + \ldots + dnIn)$$

Cette transformation après la sommation des courants pondérés se fait à l'aide du montage suivant:



b-Temps de conversion:

.../...

Ce facteur est important vu qu'il détermine la vitesse de conver-sion et la fréquence maximale.Pour notre cas le CNA utilisé,le DAC 02,à l'entrée de l'enregistreur a un temps de conversion de 1,5 µs.

Ses caractéristiques sont les suivantes:

-entrée compatible TTL,DTL,C-MOS.
-tension d'alimentation : +12V à 18V.
-résolution 10 bits+bit de signe.
-monotonie garantie à 0-70°C.
-stabilité dans toute la gamme de température 60ppm/°C max.
-sortie bipolaire : ±10V.

C-Schéma synoptique de l'enregistreur:

Voir schéma à la page 94:

D-Logiciel:

Ce programme utilise des sous programmes établis pour le disque et l'imprimante:MULTI2,CALDAD,MISFOR...
Les valeurs des Vi se trouvant dans BUVAR sur 64 bits flottants sont traitées par le prog.MISFOR pour séparer exposant et mantisse (chacun sur 2 octets) puis par le prog.ALEXPO qui permet d'aligner tous les exposants sur la même valeur de façon à garder de la mantisse que le nombre de bits attaquant l'entrée des CNA.
Lorsque la valeur est ramenée à 12 bits(la valeur de chaque Vi), l'enregistreur est initialisé pour le tracé d'un point correspon-dant à chaque Vi.A un pas donné les 6 valeurs ou points à tracer sont rangés dans la table BUVTAT.

SCHEMA SYNOPTIQUE

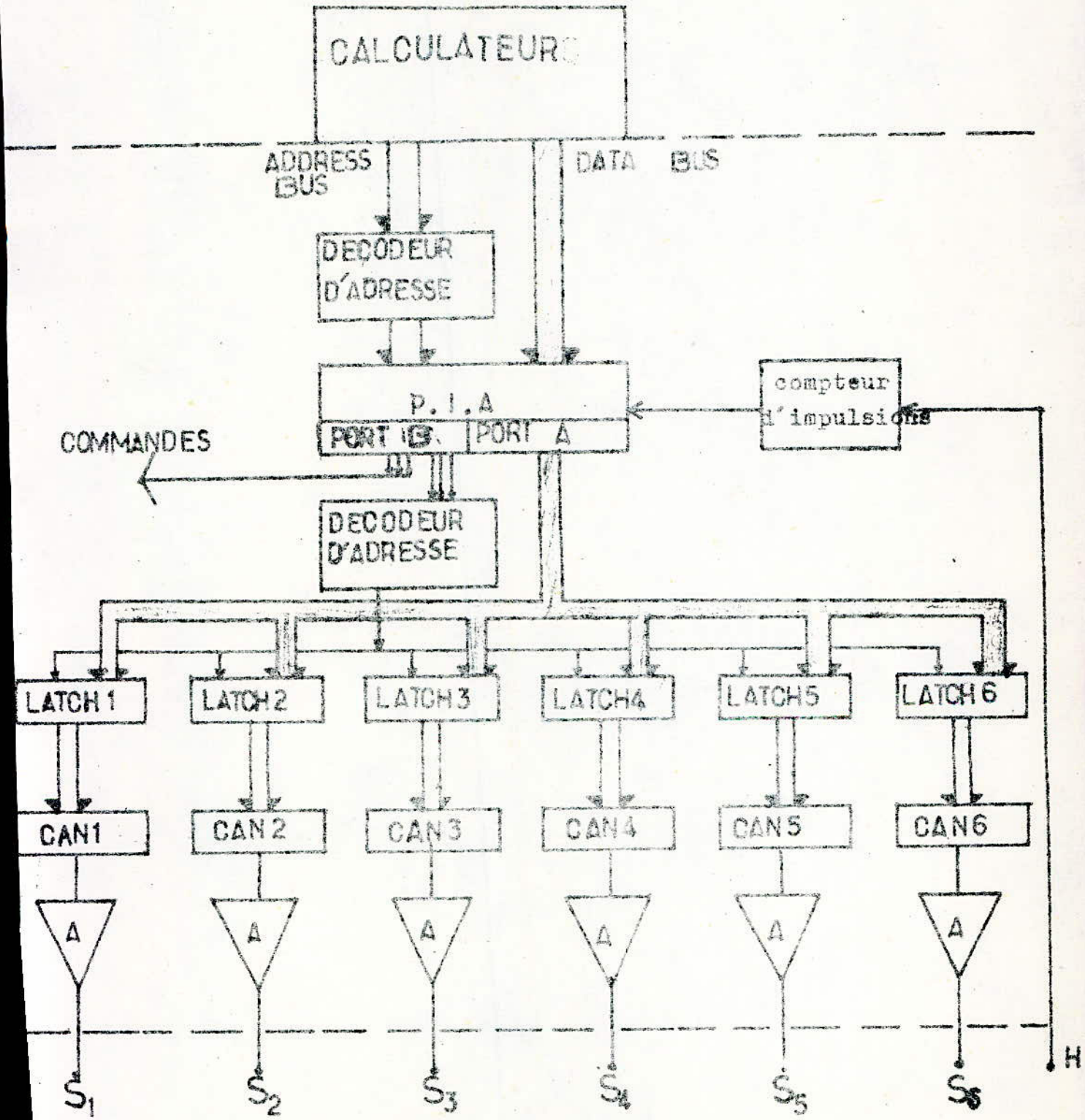


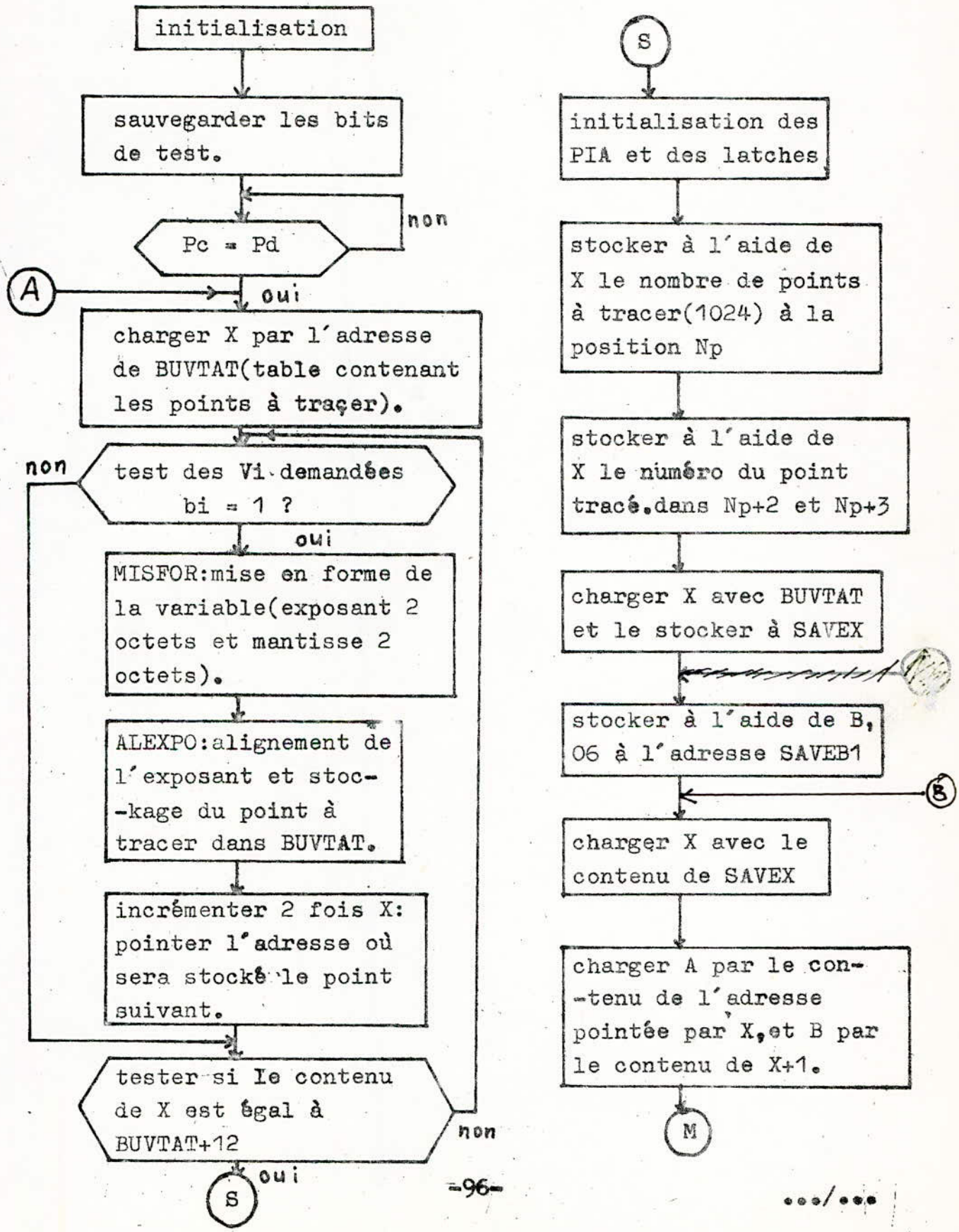
Fig. 1 : ENREGISTREUR

Organisation de la page zéro:

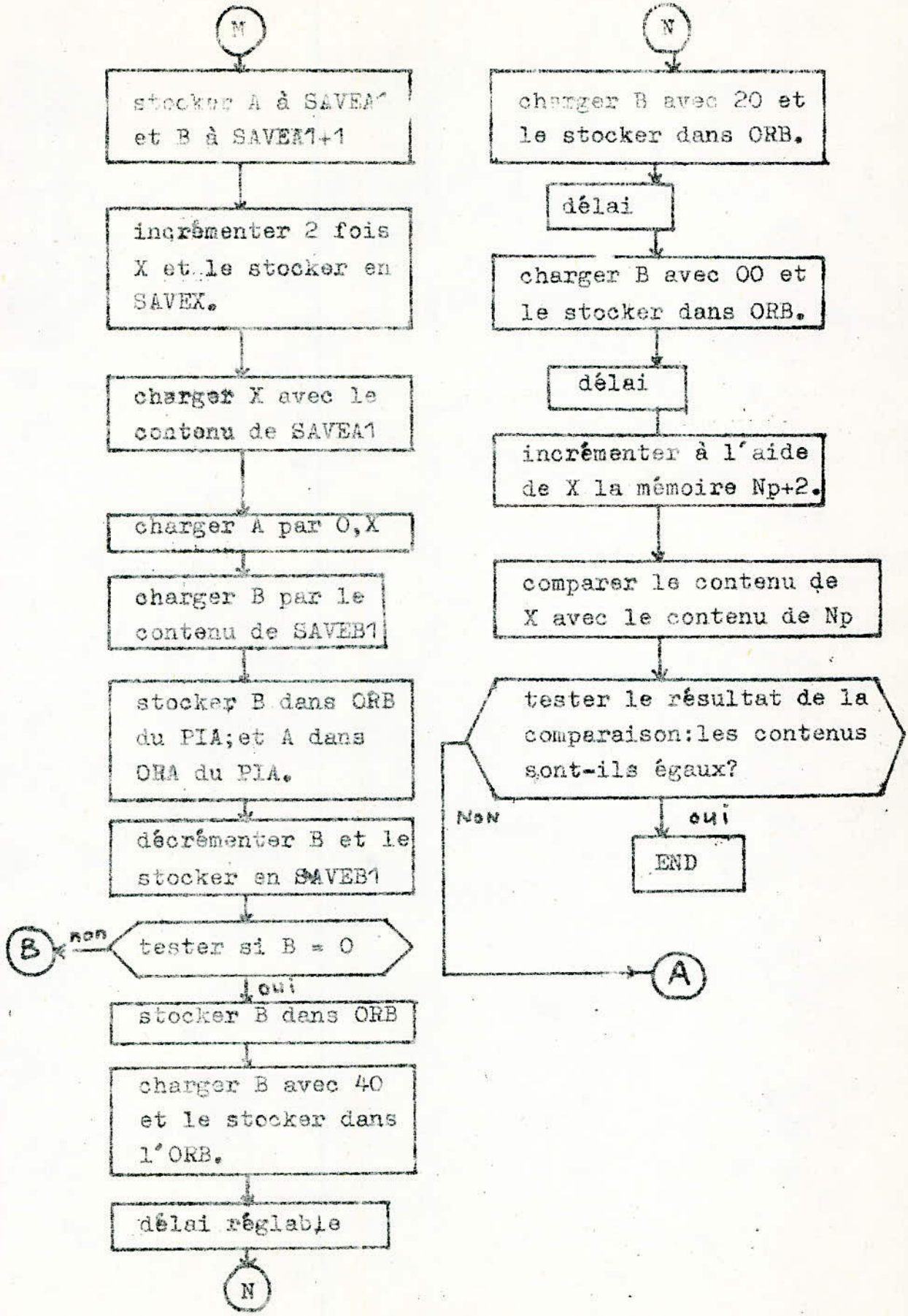
0061/ M_M utilisée dans le s/p MISFOR.
62/ So 4 octets de sauvegarde des octets de test.
63/ S1
64/ S2
65/S3
66/ SAVEX4 2 octets de sauvegarde de X.
68/ SAVEB 1 octet de sauvegarde de B.
69/ Po 4 octets contenant le résultat de MISFOR.
6A/ P1
6B/ P2
6C/ P3
6D/ U-3 octets réservés à MULTI2.
6E/ U-2
6F/ U-1
70/ U
71/ V
72/ V+1
73/ R-1
74/ R
75/ R+1 } résultat de MULTI2
76/ R+2 }
77/ R+6 } résultat de CALDAD.
78/ R+7 }
79/ Mb contient le n° de la Vi demandée.
7A/ Md 2 octets pour le n° du pas demandé.
7B/
7C/ Mo 4 octets de test.
M1
M2
M3

La table BUVTAT a 12 octets sachant qu'on sort 6 Vi .

Programme principal:



.../...


M
stocker A à SAVEA1
et B à SAVEA1+1
incrémenter 2 fois
X et le stocker en
SAVEX.
charger X avec le
contenu de SAVEA1
charger A par 0,X
charger B par le
contenu de SAVEB1
stocker B dans ORB
du PIA;et A dans
ORA du PIA.
décrémenter B et le
stocker en SAVEB1
B
non
tester si B = 0
oui
stocker B dans ORB
charger B avec 40
et le stocker dans
l'ORB.
délai réglable
N
N
charger B avec 20 et
le stocker dans ORB.
délai
charger B avec 00 et
le stocker dans ORB.
délai
incrémenter à l'aide
de X la mémoire Np+2.
comparer le contenu de
X avec le contenu de Np
tester le résultat de la
comparaison:les contenus
sont-ils égaux?
NON
oui
END
A

Programme principal:

```
        LDA A   Mo      sauvegarde des bits de test.
        STA A   So
        LDA A   M1
        STA A   S1
        LDA A   M2
        STA A   S2
        LDA A   M3
        STA A   S3
COMPAS  LDX     Md      comparer le pas demandé au pas de calcul
  DEC   DEX
        CPX     Mc
        BNE     DEC
  FIFO  LDX     BUVTAT  pointer l'adresse de la table où seront
        CLR     Mb      rangés les points à tracer.
        LDA B   #08     test des Vi demandées au 1er octet.
ROTATØ  ROR     Mo
        BCC     TOTO
        JSR     MISFOR  saut au s/p MISFOR.
        JSR     ALEXPO  saut au S/p ALEXPO.
        INX
        INX
        CPX     BUVTAT+12   tester si les 6 Vi sont prises.
        BEQ     COURBE  branchement à l'adresse COURBE.
  TOTO  INC     Mb      voir si le test du 1er octet est fini.
        CMP B   Mb
        BNE     ROTATØ
        LDA B   #10     test du 2ème octet.
ROTAT1  ROR     M1
        BCC     NONO
        JSR     MISFOR
        JSR     ALEXPO
        INX
        INX
        CPX     BUVTAT+12
        BEQ     COURBE.
```

```
NONO    INC     Mb          est-ce que le test du 2eme octet est fini.
        CMP B   Mb
        BNE     ROTAT1
        LDA B   #$18        test du 3ème octet.
ROTAT2  ROR     M2
        BCC     MIDO
        JSR     MISFOR
        JSR     ALEXPO
        INX
        INX
        CPX     BUVTAT+12
        BEQ     COURBE
MIDO    INC     Mb          voir si le test du 3ème octet est fini.
        CMP B   Mb
        BNE     ROTAT2
        LDA B   #$20        test du 4ème octet.
ROTAT3  ROR     M3
        BCC     MIMO
        JSR     MISFOR
        JSR     ALEXPO
        INX
        INX
        CPX     BUVTAT+12
        BEQ     COURBE
MIMO    INC     Mb          le test du 4ème octet est-il fini
        CMP B   Mb
        BNE     ROTAT3
COURBE  CLR     CNTRL1      initialisation du PIA(de l'enregistreur)
        CLR     CNTRL2
        LDA A   #$FF
        STA A   ORA
        STA A   ORB
        LDA B   #$06
        STA B   CNTRL1
        STA B   CNTRL2
        CLR A
        STA A   ORA
        LDA B   #$06
```


.../...

```
Bo  LDA A     CNTRL1
    BPL       Bo
              LDA A     ORA
              STA B     ORB
              DEC B
              BNE       Bo
              STA B     ORB
              LDX #$ 400
              STX       Np          mettre le nombre de points à tracer dans Np
              LDX #$ 00
              STX       Np+1        mettre 0 dans la mémoire où sera le n° du point tracé.
              LDX       BUVTAT      pointer l'adresse de la table où seront
              STX       SAVEX       rangés les points à tracer.
              LDA B #$06            charger B par le nombre de courbes à
              STA B     SAVEB1      tracer.
    BOUCLY    LDX       SAVEX       pointer la debut de BUVTAT.
              LDA A     0,X
              LDA B     1,X
              STA A     SAVEA1
              STA B     SAVEA1+1
              INX
              INX
              STX       SAVEX
              LDX       SAVEA1
              LDA A     0,X
              LDA B     SAVEB1
              STA A     ORA
              STA B     ORB
              DEC B
              STA B     SAVEB1
              CMP B #$ 00
              BNE       BOUCLY
              STA B     ORB
              LDA B #$ 40           envoi de la commande"avance du papier".
              STA B     ORB
              LDX # $ FFFF          délai réglable
```


.../...

```
DEC1    DEX
        BNE     DEC1
        LDA B #$ 20     envoi de la commande"lève plume"
        STA B   ORB
        LDX #$ FFFF     délai
DEC2    DEX
        BNE     DEC2
        LDX #$ FFFF
DEC3    DEX
        BNE     DEC3
        LDX #$ FFFF
DEC4    DEX
        BNE     DEC4
        LDA B #$ 00     remise à zéro des commandes.
        STA B   ORB
        LDX #$ FFFF     délai
DEC5    DEX
        BNE     DEC5
        LDX     Np+1    incrémenter le n° dp point et le compa-
        INX             -rer au nombre de poimts à tracer.
        STX     Np+1
        CPX     Np
        BEQ     END
        JMP     FIFO

        END
```

Sous-Programme ALEXPO:

Ce s/p aligne les valeurs sur un même exposant.

```
        ASL     Po          déterminer le signe de
        BCC     JOJO        de la mantisse.
        LDA A   P2
        LDA B   P3
        COM A               complémenter la
        NEG B               mantisse.
        BCS     JOJO
        ADD A   #$01
JOJO    LSR     Po
        LDA A   Po          déterminer si l'exposant
        LDA B   P1          unbiased est égal à 8.
        SUB B   #$07
        SBC A   #$04
        BCC     ROTATB
ROTATA  CLC                 augmenter l'exposant s'il est inférieur
        ROR     P2          à 8.
        ROR     P3
        LDA A   P1
        LDA B   Po
        ADD B   #$01
        ADC A   #$00
        LDX     Po
        CPX #$ 08
        BNE     ROTATA
        BRA     MOMO1
ROTATB  CLC                 diminuer l'exposant s'il est supérieur
        ROL     P3          à 8.
        ROL     P2
        LDA A   P1
        LDA B   Po
        SUB B   #$01
        SBC A   #$00
        LDX     Po
        CPX #$08
        BNE     ROTATB
MOMO1   LDX #$ 04           décaler la mantisse a droite en aumentant
        LSR     P2          l'exposant.
        ROR     P3
        ADD B   #$01
        ADC A   #$00
        DEX
        BNE     MIMO1
        LDA A   P2          stocker la valeur à la position pointée
        STA A   0,X         par X à la table BUVTAT.
        LDA A   P3
        STA A   1,X
        RTS                 retour de sous-prog.
```


.../...

ÇONCLUSION.

La sortie de résultats sur des supports d'informations est indispensable dans les systèmes de simulation en l'occu--rence les systèmes multiprocesseurs.Ces derniers traitent plusieurs variables à la fois qu'il est difficile d'inter--préter simultanément.Aussi faut-il d'abord les stocker.
A cet effet,le disque souple grâce à sa grande capacité de stockage ,sa fiabilité et sa grande vitesse d'échange (250 Kbits/s) est parmi les supports les plus utilisés.
L'imprimante est aussi importante malgré q'elle soit un peu lente(60 caractères/s ou 480 bits/s).Elle permet de vérifier la concordance des résultats,stockés sur les différents supports,en donnant une valeur numérique de ceux-ci.
L'enregistreur potentiométrique,lui,choisi pour son intérêt transcriptif de la valeur numérique en signal analogique, donne à l'ingénieur un aspect dynamique du processus aussitôt après le calcul de la variable.Ainsi l'utilisateur a l'impression de suivre le vrai processus.
Nous pensons que ces trois supports sont importants dans ce cadre.

ANNEXE

****

PERFECTIONNEMENT D'UN ELEMENT DU SYSTEME:

I-Introduction:

La connaissance du microordinateur aussi bonne soit-elle n'est pas suffisante lorsqu'il s'agit de concevoir ou de réaliser un système à microprocesseur.Il faut encore bien connaitre tout ce qui constitue son environnement tant en ce qui concerne le hardware que le software.

Ainsi pour bien aborder notre projet,nous nous sommes propo--sés de faire une étude et réalisation d'un élément de notre système;quoique cette dernière ait déja fait l'objet d'une thèse d'ingeniorat.

Il s'agit de la réalisation d'une carte esclave.En effet, considérée comme une initiation,cette étude nous a permis de toucher concrètement à la notion d'adressage des mémoires et de modifier,compte tenu de notre but et de nos hypothèses,la réali--sation de la carte esclave.

Cette carte sera l'un des 33 CPU composant notre système multiprocesseur"MAITRE'-ESCLAVE".On rapelle que le système est composé de 32 esclaves et d'un 33ème CPU appelé "MAITRE " qui arbitrera le conflit d'accès,des esclaves,à la mémoire commune (donc la communication entre eux).

Cette carte ayant fait l'objet d'une étude nous nous limite--rons à une présentation brève de l'esclave conçu et des modifica--tions apportées à savoir:

-Augmentation de la capacité mémoire(ce qui rend la carte opérationnelle).

-logique lecture/écriture.

II- Réalisation:

Cette carte est un microordinateur construit autour du µP MC6800.

Les principaux éléments composant cette carte sont:

-Le uP qui constitue l'unité centrale.

-Les mémoires ROM et RAM.

.../...

-L'interface entrée/sortie qui est l'organe de dilogue entre le ordinateur et les périphériques tels que :

.lecteur de ruban

.TTY

.lecteur de disque

A- DESCRIPTION DU MATERIEL UTILISE:

-Le uP MC6800

-Les 8T26 et 8T95

-L'horloge MC6875

-Les RAM MC6810

-Les ROM 2708

-Le PIA MC6821

A-1- Le microprocesseur MC6800:

A-1-1- Présentation du MC6800:

LE MC6800 est un microprocesseur monolithique 8 bits, réalisant la fonction d'unité centrale pour la famille 6800. Compatible T.T.L, le MC6800 ne demande qu'une alimentation de +5V et n'a besoin d'aucun circuit externe T.T.L pour l'interface avec le bus.

.../...

Sa consommation varie autour de o,25w.Il se présente sous forme de boitier DIL(Dual In Line)de 4o broches constituant ses lignes d'E/S.C'est un circuit intégréLSI réalisé en thchnologie MOS à canal N(N-MOS).

Le MC68oo est un µp d'usage général.Les bus de données et d'adresses sont séparés.La capacité d'adressage est de 65536 octets.Les registres d'E/S inclus dans les coupleurs sont traités comme des positions de mémoire.Le 68oo dispose de 7 modes d'adressage:registre,page zéro,étendu, indéxé,relatif,immédiat et implicite.IL a 6 registres programmables par l'utilisateur:

-2 accumulateurs de 8 bits.
-1 index de 16 bits.
-1 pointeur de pile de16 bits.
-1 compteur ordinal de 16 bits.
-1 registre d'état de 8 bits.

La configuration minimale requiert 5 boitiers:

-horloge
-microprocesseur.
-RAM
-ROM
-coupleurs.

Il existe 3 versions du 68oo:

-Le 68oo fonctionnant à 1MHz(temps minimum pour une instruction 2µs)
-Le 68Aoo fonctionnant à 1,5 MHz(temps minimum pour une instruction 1,3µs).
Le 68Boo fonctionnant à 2 MHz(temps minimum pour une instruction 1µs).

A-.1.2 Jeu d'instructions:

Le 68oo a un jeu de 72 types d'instructions.Compte tenu des modes d'adressage, il possède 197 instructions. Ces instructions peuvent être classéssen 5 groupes:

-instructions de transfert.

-instructions arithmétiques(binaires et décimales).
-instructions logiques(ET,OU...)

-instructions de saut systématique ou conditionnel(boucles saut...)et appel de sous-programmes.
-instructions de contrôle du µp(relatives aux interruptions:SWI ,manipulation dans la pile ...)
La longueur des instructions est de 1,2 ou 3 octets. La plupart des instructions opèrent sur l'accumulateur et un opérande.

A...1.3 Lignes d'entrées/sorties:

a/ Bus de données:
C'est un bus bidirectionnel de 8 bits fonctionnant en logique 3 états:

-2 états bas:"1" et "o" logique
-1 état haut: haute impédance

b/ Bus d'adresses:
C'est un bus unidirectionnel de 16 bits permettant l'adressage de 64K octets et pouvant être en HI.
c/ Bus de contrôle et de commande(voir tableau 1)

nom des broches	explications
Do-D7	bus de données
R/W	selection lecture/ecriture ou E/S
VMA	adresse memoire valide
Ao-A15	bus d'adresses
Reset	Remise à l'etat initial
Vcc	tension d'alimentation +5V
GND	masse
DBE	validation du bus de données
TSC	commande de mise à l'etat 3 états
BA	prise en compte d'une demande d'acces direct à la memoire.
$\overline{Halt}$	demande d'acces direct à la mémoire
$\overline{NMI}$	demande d'interruption non masquable
$\overline{IRQ}$	demande d'interruption masquable
Ø1,Ø2	phases d'horloge

Tableau 1: brochage du 6800

Ce bus englobe:les signaux de symcronisation(Ø1 et Ø2),les signaux de contrôle(R/$\overline{W}$,VMA,DBE,BA,TSC),les signaux d'interruption($\overline{Halt}$, $\overline{Reset}$,$\overline{IRQ}$,$\overline{NMI}$).Ces 3 bus sont récapitulés sur le tableau 1.

.../...

A-1-4- Organisation externe du 6800:(fig.1)

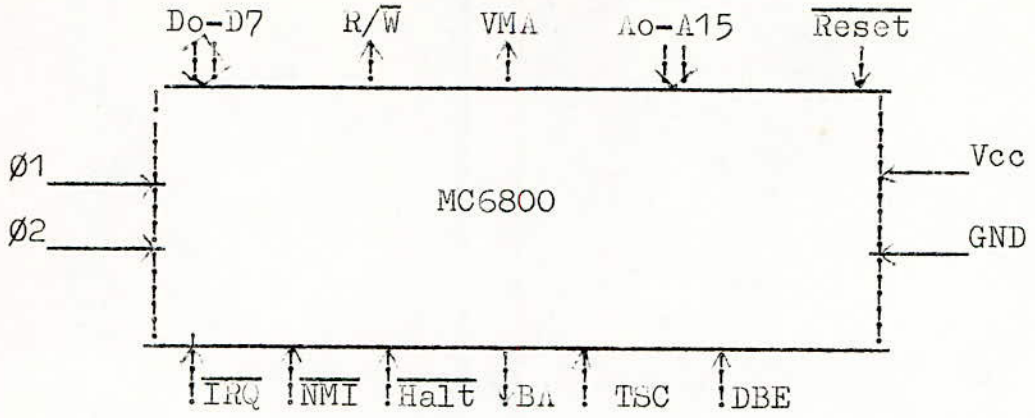


fig.1:Organisation externe du 6800.

A-1-5-Organisation interne du 6800:(fig.2)

a-ALU(Arithmetic Logic Unit).

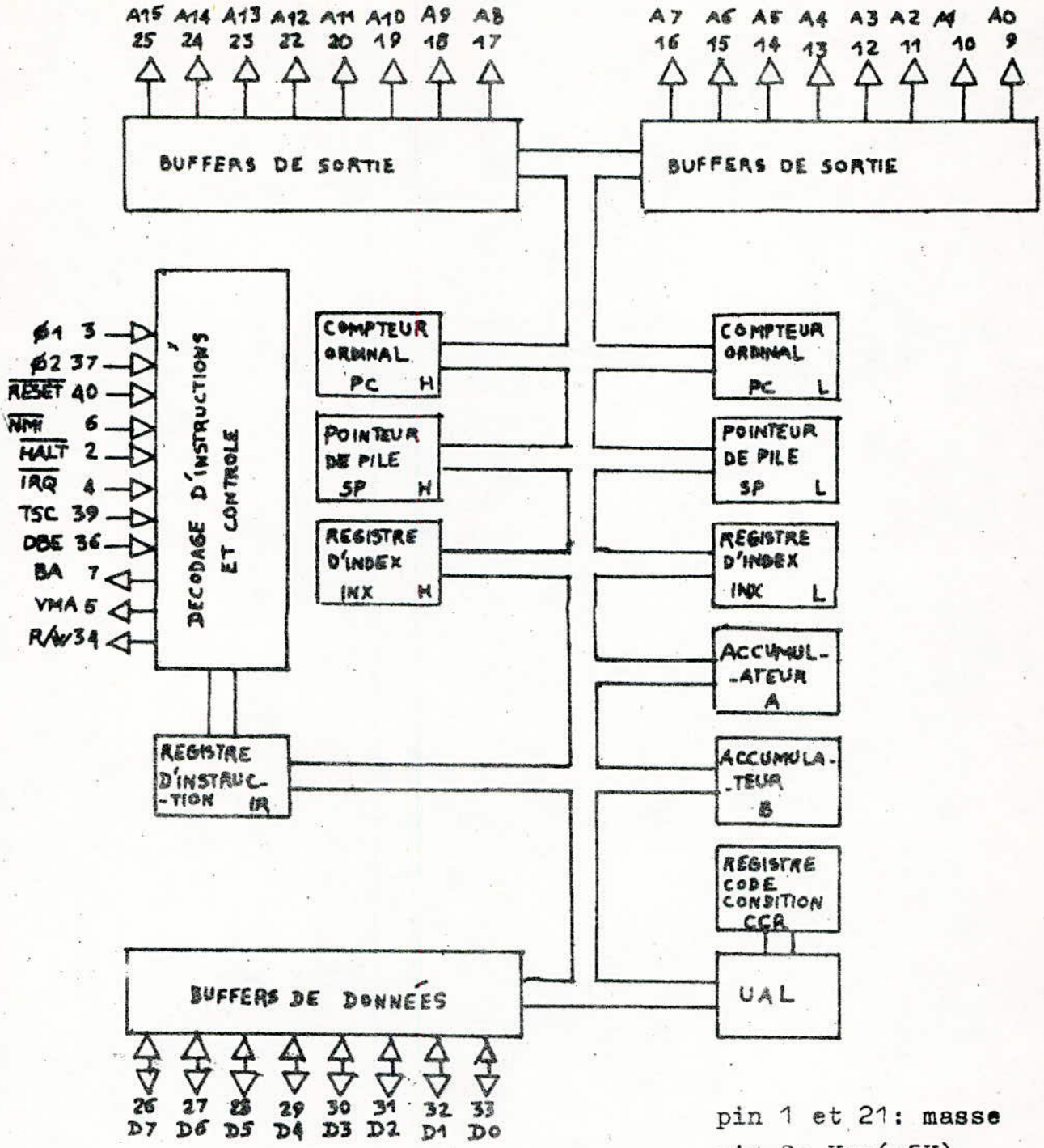
C'est l'ensemble des circuits combinatoires capables d'effectuer des opérations arithmétiques et logiques du traitement des informations.L'ALU gère et manipule les données.

b-Unité de contrôle et de décodage:

Analyse l'information du programme,décode les instruc--tions et les fait exécuter par lALU,l'acc....

c-Registres internes:

-Compteur ordinal(Program Couter PC):registre à 16 bits,déter--mine la séquence des instructions apres leur décodage;s'in--crémente à chaque instruction.
-Accumulateurs A et B:registres à 8 bits où sont rangées les données intermédiaires et les résultats nécessaires à l'ALU.
-Registre d'index:registre à 16 bits;comtient l'adresse en mode indéxé.
-Registre d'instruction RI:c'est un registre à 16 bits où est rangée l'instruction,reçue de la mémoire,pendant le temps nécessaire à son exécution.
-Pointeur de pile(SP:stack pointer):registre à 16 bits,utilisé lors d'un passage à un sous-programme;il stocke l'adresse de



pin 1 et 21: masse
pin 8: Vcc(+5V)

fig: 2: ORGANISATION INTERNE DU MC6800

retour au programme initial.

-Registre d'état(CCR:Code Condition Register):registre à 8 bits; contient les informations concernant les résultats d'une opération exécutée par l'ALU.

signification de ses bits:

N:résultat négatif

Z: // nul

V:dépassement

C:retenue

H:demi-retenue

I:bit de masquage d'interruption

Les 2 autres bits sont constamment égaux à 1.

A-2-Les Buffers 8T26 et 8T95:

Chaque ligne du bus adresses,données,ou de contrôle ne peut commander qu'une charge TTL.Pour délivrer un courant supérieur à cette charge,ces bus sont amplifiés par des buffers 8T26 et 8T95 à 3 états.Ces buffers jouent le rôle d'interface d'adaptation,d'amplification,de protection et surtout d'isoler les bus d'adresses,de données,de lignes de commande ou de contrôle.

A-2-1- Les buffers 8T26:

Le bus de données doit être amplifié dans les 2 sens, écriture et lecture car,selon que le uP reçoit ou transmet une donnée ,les lignes correspondantes sont entrantes ou sortantes. Pour cela on utilise deux 8T26,bidirectionnel.

brochage et table de vérité des 8T26: voir fig.3

A-2-2-Les buffers 8T95:

Pour lire ou écrire des données dans une mémoire,on place l'adresse sur un interface dit d'adresse.Le circuit utilisé à cet effet est le buffer 8T95(au nombre de 3)unidirectionnels, 3 états.

brochage et table de vérité des 8T95:voir fig.4

A-3- L'Horloge MC6875:

Le circuit d'horloge doit générer les signaux d'horloge requis par le système conçu autour du MC6800:
-Ø1(N-MOS)et Ø2(N-MOS)nécessaire au MC6800.
-Ø2(TTL)nécessaire à la logique de commande.
-signaux de synchronisation(cas de mémoire lente).
Ce générateur d'horloge est compatible avec les versions 1,0-1,5-2,0 MHz du MC6800.

brochage de la MC6875:(voir fig.5)

A-4- Les RAM 6810:

Ce circuit,fabriqué en technologie N-MOS,n'a besoin que d'une seule alimentation de +5V.C'est une mémoire de 128 octets; compatible TTL et DTL;à fonctionnement statique;n'a besoin d'aucune horloge ou de signal de rafraichissement.
Schéma et brochage de la 6810:(voir fig.6)

A-5- Les ROM 2708:

Ce sont en fait des EPROM utilisables pour la mise au point de système demandant une mémoire non volatile.Elles sont effaçables aux UV;capacité de 1k octets;fonctionnement statique; compatible TTL;sortie 3 états.
Schéma et brochage de la 2708:(voir fig.7)

A-6-Le P.I.A MC6821:(Periphecal Interface Adapter)

Le µP ne peut dialoguer avec les unités périphériques. Il est nécessaire de prévoir un étage tampon(interface);pour cela a été utilisé le 6821.C'est un interface parallèle qui assure la transmission en parallèle des informations.A cet effet le MC6821 possède 16 sorties groupées en 2 ports:le port A et le port B. Chacun de ces ports est régi par:
-un registre de contrôle(Control Register:CR)
-un registre de direction de transfert des données(Data Direction Rgister):DDR.
-un registre de sortie(Output Register:OR)
-2 lignes de contrôle d'interruption.

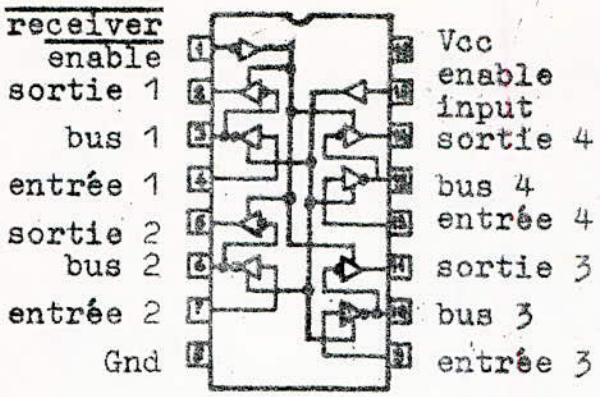


fig.3a-brochage du 8T26

driver enable	receiver enable	sorti	entrée	bus
L	L	L	X	L
L	L	H	X	H
L	H	X	X	
H	H	X	L	L
H	H	X	H	H

fig.3b-table de vérité du 8T26

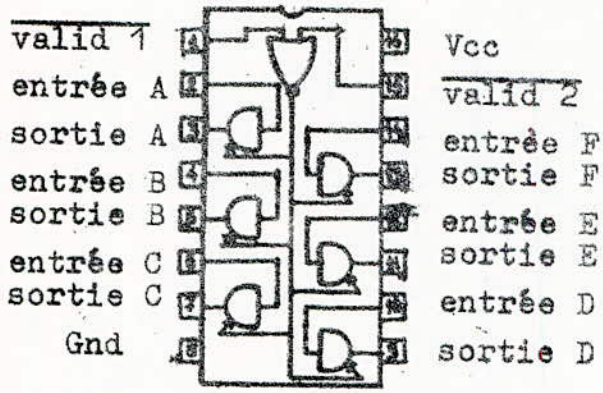


fig.4a-brochage du 8T95

Valid2	Valid1	entrée	sortie
L	L	L	L
L	L	H	H
L	H	X	Z
H	L	X	Z
H	H	X	Z

fig.4b-table de vérité du 8T95

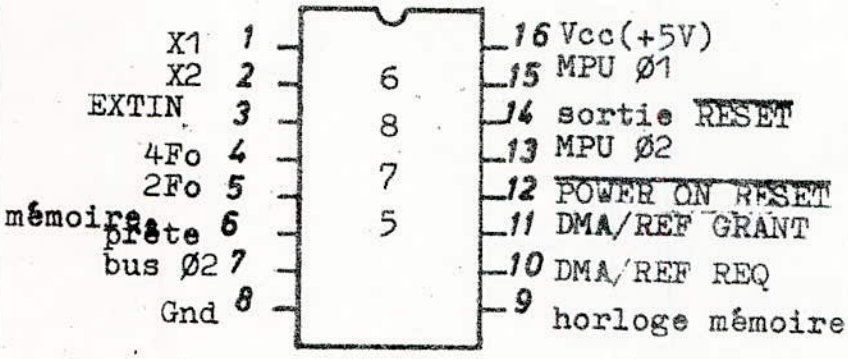


fig.5-brochage de l'horloge 6875.

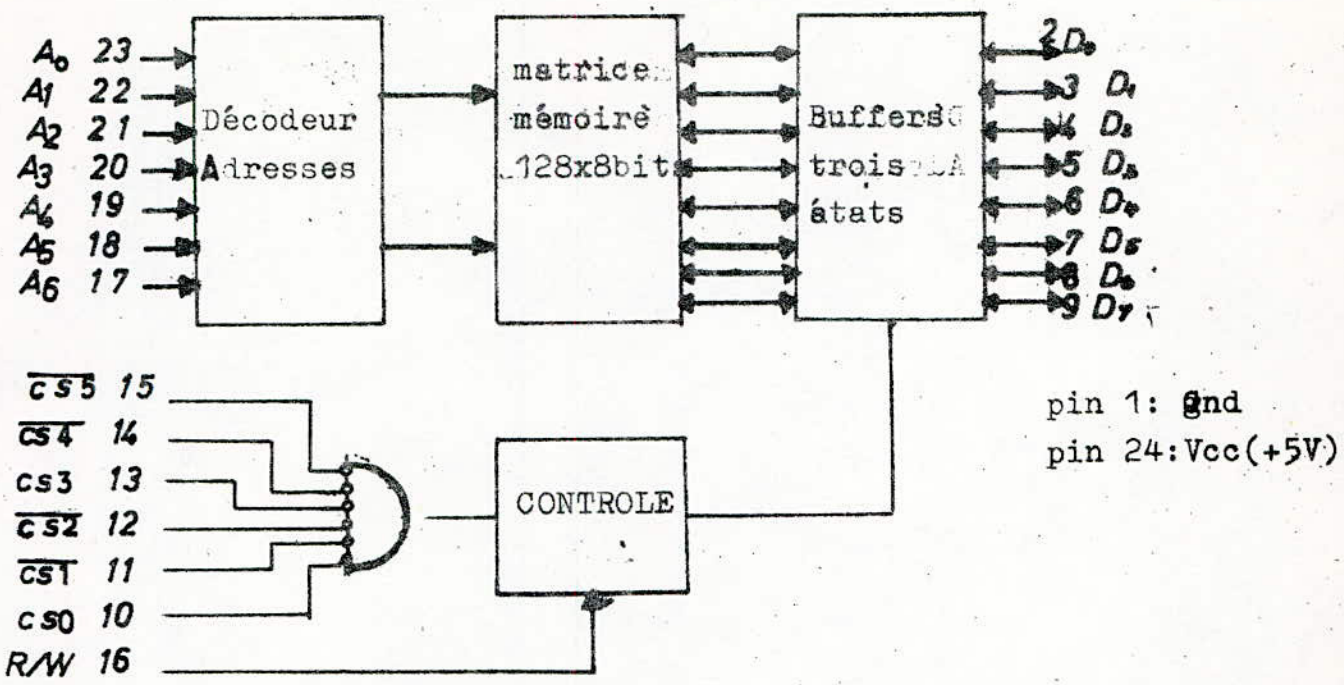


fig.6: Organisation interne et brochage de la RAM 6810.

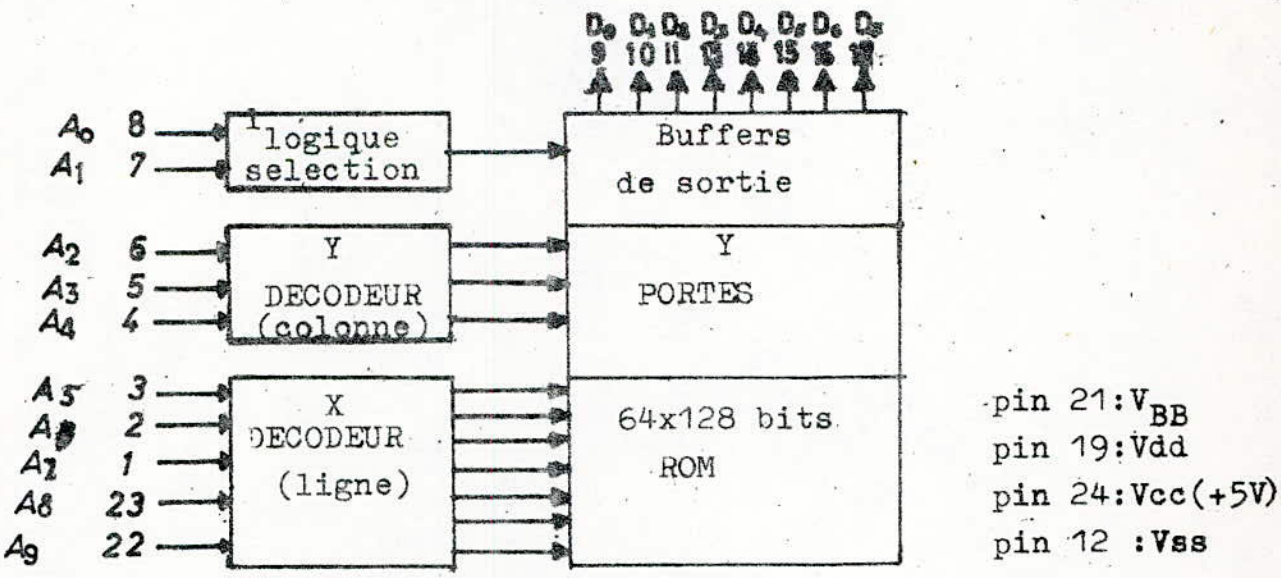


fig.7 : Organisation interne et brochage de l'EPROM 2708.

Ces 2 ports sont reliés à travers ces registres au registre d'en-
-trées du bus des données(Bus Input Register:BIR)!
Le PIA communique avec le MPU par:les bus:de données,d'adresses,
de contrôle.

Signaux MPU ——→ PIA:

CS0,CS1,$\overline{CS2}$:reliés au bus adresses du MPU,valident le PIA lorsque
CS0 CS1 $\overline{CS2}$ = 110

RS1,RS0:Le PIA étant sélectionné,les 4 combinaisons de ces 2 bits adressent les registres internes.En conséquence,le PIA occupe 4 positions mémoires dont les adresses doivent être différentes des autres adresses de RAM,ROM ou autre coupleur.

E:signal d'activation des échanges,sert aussi de synchronisation pour le PIA.Cette entrée est reliée à l'horloge Ø2 du MPU.

R/$\overline{W}$:signal de lecture-écriture:à 1=lecture,à 0=écriture.

Do-D7:bus bidirectionnel de données,il aboutit dans le PIA,à un amplificateur qui peut être activé ou mis à l'état HI par le R/$\overline{W}$ si le PIA est selectionné.Par ces 8 entrées arrivent les echanges d'informations entre le MPU et le PIA.

$\overline{Reset}$:mis à 0,ce signal remet à 0 tous les registres internes.

$\overline{IRQA}$;$\overline{IRQB}$:2 lignes de demande d'interruption dans l'exécution d'un programme du MPU.

Signaux PIA ——→ Périphérique:

PA0-PA7)
PB0-PB7) 16 lignes de données programmables,en entrée ou en sortie, et groupées en 2 ports.Ces 2 ports reflètent,en sortie, le contenu de 2 registres internes de 8 bits maintenus tant qu'il n'y a pas de modifications sur ces registres.

CA1,CB1: 2 lignes d'entrée d'interruption.

CA2,CB2: 2 lignes progammables en entrée d'interruption ou en sor-
-tie de commande.Dans ce cas elles reflètent directement l'état d'un bit du registre interne de contrôle.

Lignes d'alimentation:

Vss=0V

Vcc=+5V

La consommation est d'environ 110mA et la puissance dissipée de 550mW.

.../...

*Registres internes du PIA:

Ils sont répartis en 2 groupes de 3 relatifs aux ports A et B.

CRA,CRB:contiennent les paramètres de fonctionnement.

DDRA,DDRB:contiennent le mot fixant le sens du transfert(entrée ou sortie)pour chacune des lignes de données.

- Un état"1"définit une sortie.
- Un état"0"définit une entrée.

ORA,ORB:mémorisent les données en sortie lors d'une écriture.A la même adresse,on peut lire les données présentes en entrée, mais elles devront être mémorisées à l'exterieur.

*Adressage des registres:

Deux lignes RS0 et RS1 permettent de choisir l'un des registres ORA,ORB,DDRA,DDRB.Les registres de contrôle sont adressés directement.

Bloc diagramme du MC6821:(fig.8)

Tableau d'adressage des registres:(fig.9)

B-Adressage de la mémoire commune et de la mémoire locale:

Les interfaces de données sont activées en sortie ou en entrée suivant que le MPU reçoit l'ordre de lecture ou d'ecriture.

B-1- Circuit de lecture/écriture:

Logique de lecture/écriture:(Voir fig.10a)

Table de vérité:(Voir fig.10b)

B-1-a- Ecriture:

Cette opération requiert:

-la validation du signal d'écriture(non mis en HI)et $R/\overline{W} = 0$.

-le bus adresse disponible(BA=0).

-le bus de données activé(DBE=1).

Alors;le signal d'écriture a pour expression:

$$S_E = \overline{R/\overline{W}.\overline{BA}.DBE}$$

B-1-b- Lecture:

Cette opération requiert:

.../...

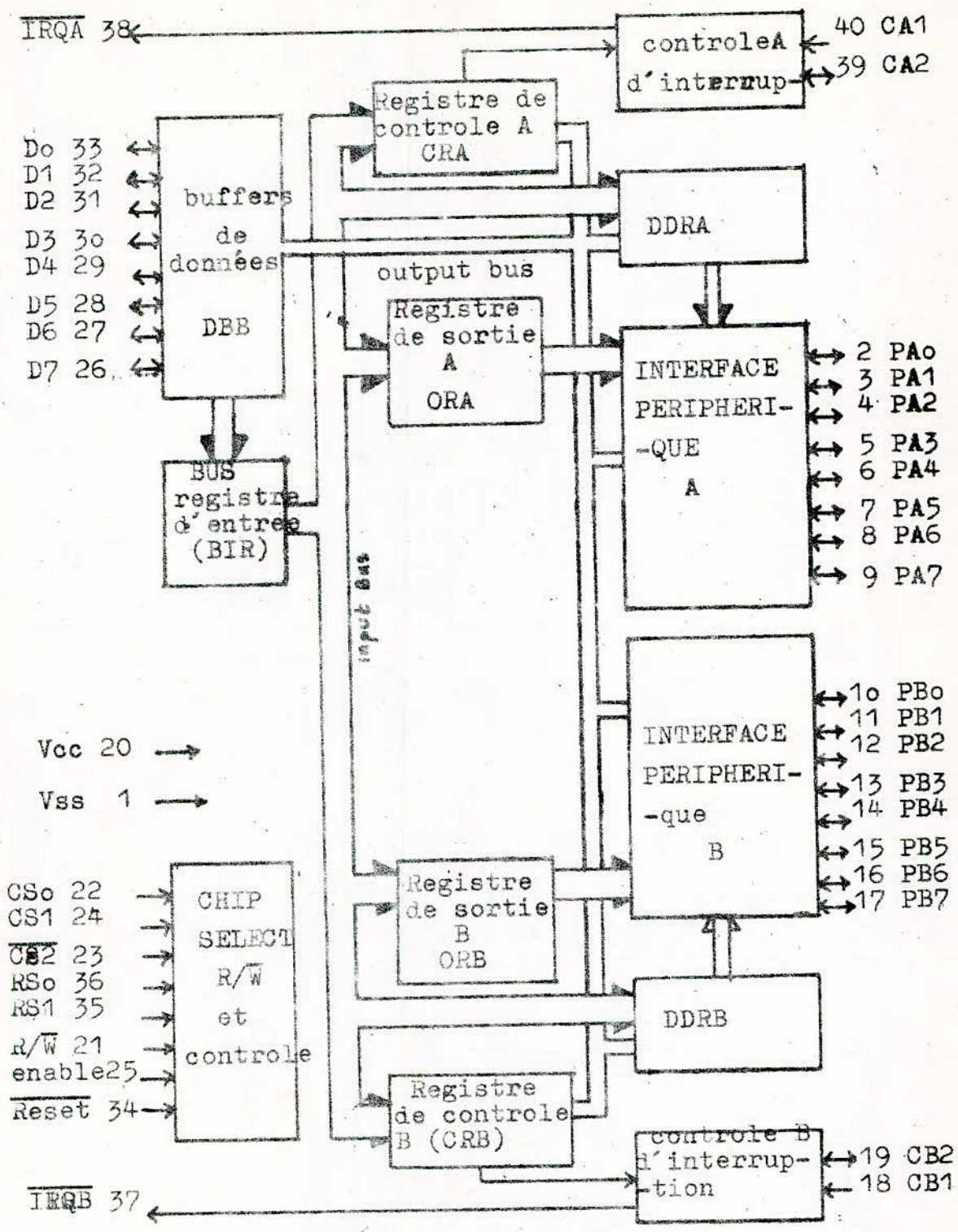


fig.8: Bloc diagramme du MC 6821.

RS1	RSO	CRA-2	CRB-2	REGISTRE SELECTIONNE
0	0	0	-	DDRA
0	0	1	-	ORA et INTERFACE
0	1	-	-	CRA
1	0	-	0	DDRB
1	0	-	1	ORB et INTERFACE
1	1	-	-	CRB

fig.9 : Tableau d'adressage des registres du PIA.

R/$\overline{W}$	BA	DBE	S_E	$\overline{S_L}$	FONCTION
0	0	0	0	1	HI
0	0	1	1	1	ECRITURE
0	1	0	0	1	HI
0	1	1	1	1	HI
1	0	0	0	1	HI
1	0	1	0	0	LECTURE
1	1	0	0	1	HI
1	1	1	0	0	

fig.10-b : Table de vérité de la lecture/écriture.

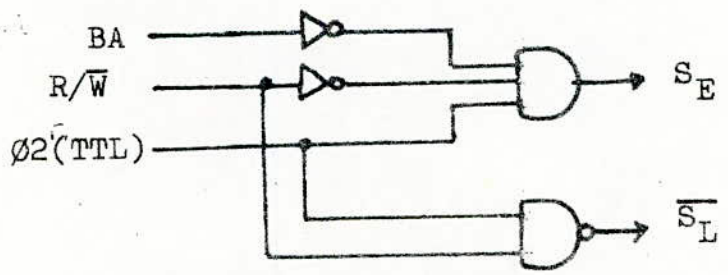


fig.10-a : Logique de lecture/ecriture.

-$R/\overline{W}$ = 1

-DBE = 1.Ce signal est pratiquement identique à Ø2 TTL(sinon plus large).

Le signal de lecture a pour expression:

$$S_L = R/\overline{W}.Ø2$$

Comme les 8T26 sont activés par un niveau bas on prend: $\overline{S_L}$

B-2- Activation,desactivation et aiguillonnage des 8T26:

Chaque esclave a une mémoire locale qui occupe de 8000 à FFFF(32K octets)et se partage une mémoire commune,avec les autres esclaves,qui occupe de 0000 à 7FFF.L'état de la ligne A15(0 ou 1) indiquera le passage d'une zone à l'autre.Cette ligne,adjointe à la logique lecture/ecriture,constituera le circuit d'aiguillomnage des 8T26 en zone commune(A15=0)Ou en zone locale(A15=1 les mettant en HI).

Aiguillonnage des 8T26:(Voir fig.11a)

Table de vérité d'aiguillonnage des bus adresses:(Voir fig.11b)

Activation,désactivation des 8T95:

D'après la table de vérité du 8T95,le raccordement de $\overline{valid.1}$ et $\overline{valid.2}$ à l'état bas permet de retrouver l'adresse d'entrée en sortie alors que le raccordement à l'état haut place le buffer en HI.

B-3- Blocage du CPU:

L'esclave n'accède en MC qu'après l'acquittement du mai-tre;en attendant ce signal,il est mis en arrêt par application de la A15 à l'entrée $\overline{Halt}$.

Le signal d'acquittement appliqué à une bascule JK,comme la A15, basculera l'état de celle-ci et mettra le $\overline{Halt}$ à l'état haut et le CPU travaille en zone commune.

Seulement,vue sa propriété d'anticipation le CPU risque d'exécuter l'instruction dont l'adresse a A15=0 avant de prendre en considér-ation le $\overline{Halt}$.Or cette adresse se trouve en zone commune et son

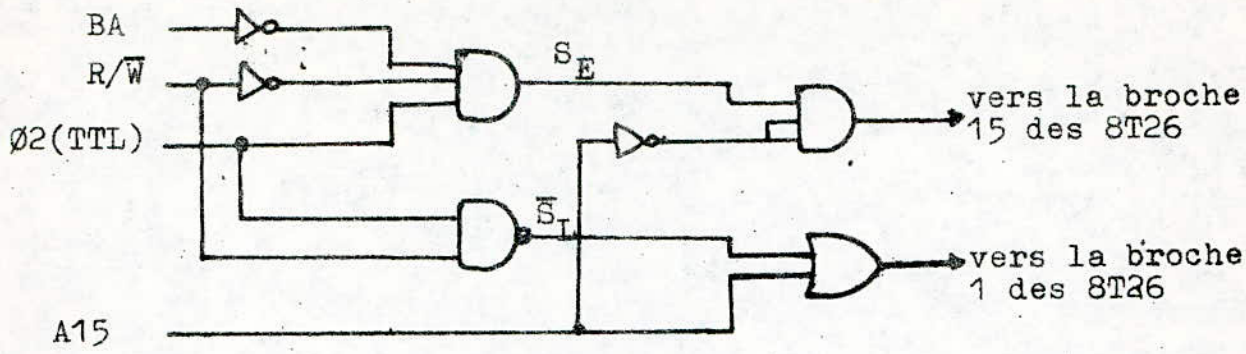


fig.11-a : Aiguillonnage des 8T26.

S_E	$\overline{S_L}$	A15	broche 15	broche 1	
0	0	1	0	1	8T26 en HI; travail en zone locale.
1	1	1	0	1	
0	1	1	0	1	
0	0	0	0	0	← lecture en zone commune
1	1	0	1	1	← écriture en zone commune
0	1	0	0	1	

fig.11-b : table de vérité.

exécution n'est permise qu'après l'acquittement.Pour cela,il est nécessaire de bloquer l'horloge(Ø1 et Ø2 :

)àtravers un circuit tampon;le signal d'acquittement étant appliqué à clear de la JK pendant tout le temps que l'esclave travaille en MC pour neutraliser la A15.

Chronogramme de blocage de l'horloge:(voir fig.12)
Schéma du circuit tampon:(voir fig.13)
L'esclave générera un signal fin d'exécution à travers un PIA pour reprendre le travail en mémoire locale.

C- Décodage des mémoires:

La carte réalisée a une capacité de 10K octets mémoire répartis comme suit:
-1K octets de RAM MC6810 de 128 octets chacune soit 8 boitiers.
-9K octets de Rom MC2708 de 1K octets chacune soit 9 boitiers.
Les RAM occupent la zone de A000 à A3FF.
Les ROM occupent la zone de DC00 à FFFF.

Ce choix de zone n'est pas aléatoire.Cet emplacement nous permet de couvrir la ROM et la RAM MIKBUG par la ROM/RAM du système(esclave).
En effet,on rappelle que la RAM MIKBUG se situe en A000 à A03F; et la ROM MIKBUG qui elle,n'est pas figée,peut occuper toutes les positions de E000 à FFFF.

C-1- Adressage des RAM: (voir tableau 2)

D'après ce tableau on constate que:
Ao à A6: servent à l'adressage interne des mémoires.
A7,A8,A9 :servent à la sélection du boitier.
A10,A11:servent au verrouillage à A3FF.
A12 à A15:servent à la sélection de la zone mémoire(zone A).

Pour cette sélection:(voir: -fig.14 pour la sélection de la zone A
-tableau 3 pour sélection du boitier
-tableau 2 pour adressage des RAM)

C-2- Adressage des ROM:(voir tableau 4)

.../...

fig.13 : Circuit tampon.

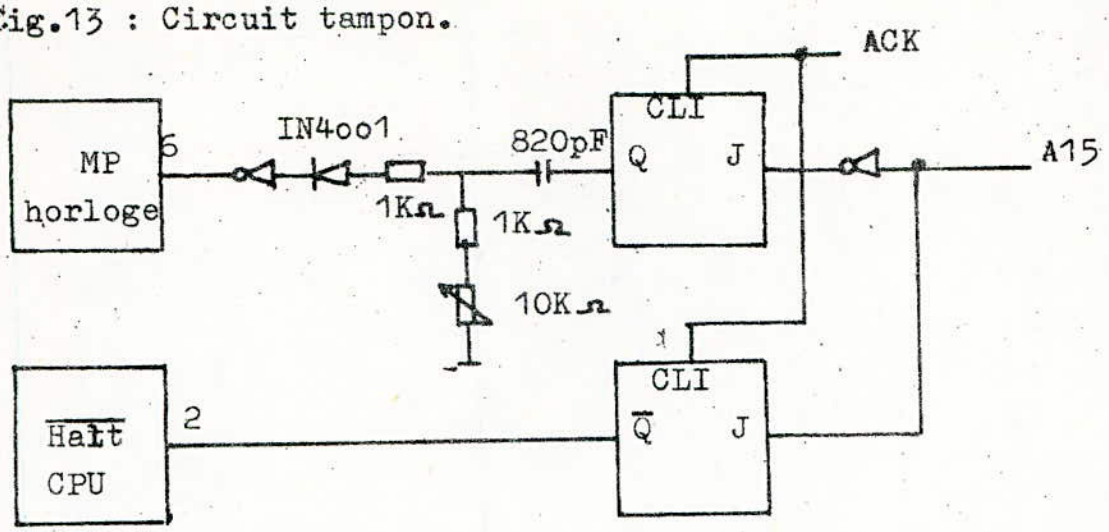
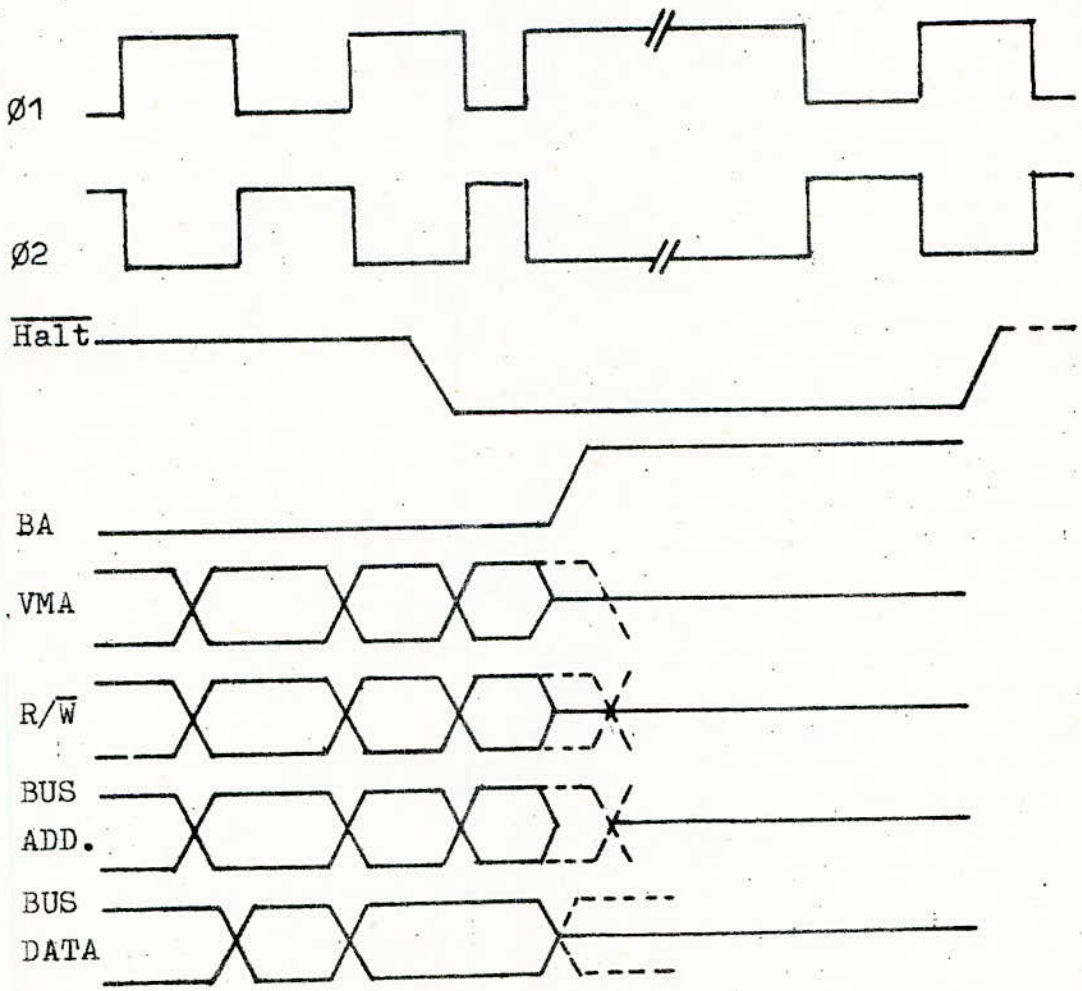


fig.12 : Chronogramme de blocage de l'horloge.



CSo	$\overline{CS1}$	$\overline{CS2}$	CS3	$\overline{CS4}$	$\overline{CS5}$	boitier sélectionné
$\overline{A10}$	A9	A8	$\overline{A11}$	A7	(A)	1
A7	A9	A8	$\overline{A11}$	A10	(A)	2
A8	A7	A11	$\overline{A10}$	A9	(A)	3
A8	A9	A10	A7	A11	(A)	4
A9	A8	A7	$\overline{A10}$	A11	(A)	5
A9	A8	A10	A7	A11	(A)	6
A9	A7	A10	A8	A11	(A)	7
A8	A11	A10	A9	$\overline{A7}$	(A)	8

Tableau 3 : Sélection des boitiers RAM.

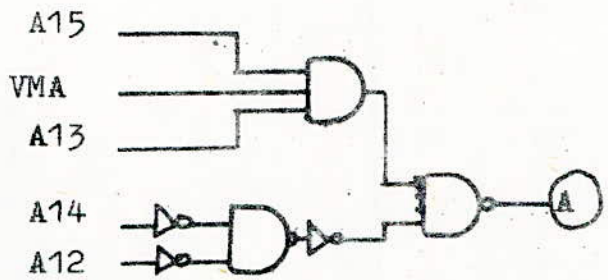


fig.14: Sélection de la zone A.

	A15	A14	A13	A12	A11	A1o	A9	A8	A7	A6	A5	A4	A3	A2	A1	Ao
A000	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A07F	1	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1
A080	1	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0
A0FF	1	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
A100	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
A17F	1	0	1	0	0	0	0	1	0	1	1	1	1	1	1	1
A180	1	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0
A1FF	1	0	1	0	0	0	0	1	1	1	1	1	1	1	1	1
A200	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
A27F	1	0	1	0	0	0	1	0	0	1	1	1	1	1	1	1
A280	1	0	1	0	0	0	1	0	1	0	0	0	0	0	0	0
A2FF	1	0	1	0	0	0	1	0	1	1	1	1	1	1	1	1
A300	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
A37F	1	0	1	0	0	0	1	1	0	1	1	1	1	1	1	1
A380	1	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
A3FF	1	0	1	0	0	0	1	1	1	1	1	1	1	1	1	1

Tableau 2 : Adressage des mémoires RAM.

	A15	A14	A13	A12	A11	A1o	A9	A8	A7	A6	A5	A4	A3	A2	A1	Ao
DCOO	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0
DFFF	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
EOOO	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
E3FF	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1
E400	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0
E7FF	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1	1
E800	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0
EBFF	1	1	1	0	1	0	1	1	1	1	1	1	1	1	1	1
ECOO	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0
EFFF	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
FOOO	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
F3FF	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1
F400	1	1	1	1	0	1	0	0	0	0	0	0	0	0	0	0
F7FF	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
F800	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
FBFF	1	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1
FCOO	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
FFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Tableau 4 : Adressage des mémoires ROM.

D'après ce tableau on constate que:

Ao-A9 :servent à l'adressage interne des mémoires.

A10 à A13 :servent à la sélection du boitier.

A14,A15:servent à la sélection de la zone(de DC00 à FFFF).

Pour cette sélection a été utilisé le décodeur SN74154.

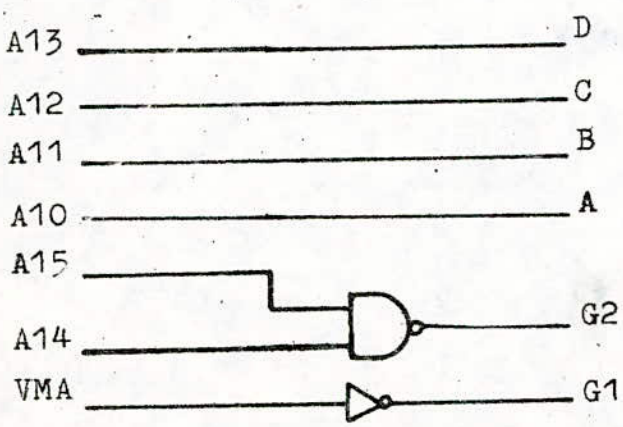
Le décodeur SN74154:(table de vérité voir fig. tableau5).

C'est un décodeur monolithique 4 lignes parmi 16; utilise un circuit TTL pour décoder 4 entrées codées binaire en une parmi les 16,mutuellement exclusives,sorties quand les 2 entrées strobes(enables)G1 et G2 sont au niveau bas.La fonction de démultiplexage,par l'utilisation de 4 lignes d'entrée pour adresser la ligne de sortie,passant la donnée d'une des entrées strobes avec l'autre strobe au niveau bas,est performante.Quand un des strobes(enable)est au niveau haut,toutes les sorties sont a l'état haut.Ces démultiplexeurs sont utilisés pour le décodage des mémoires performant.Ce circuit est compl--ètement compatible TTL et DTL.Toutes les entrées sont amplif--iées.Sa consommation varie de 85 à 170mW.

****

INPUTS						OUTPUTS															
G1	G2	D	C	B	A	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
L	L	L	L	L	L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	L	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	L	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H
L	L	L	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H	H
L	L	L	H	H	L	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H	H
L	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H	H
L	L	H	L	L	L	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H	H
L	L	H	L	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H	H
L	L	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H	H
L	L	H	H	L	L	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H	H
L	L	H	H	H	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	L
L	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	L	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H
H	H	X	X	X	X	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H	H

Tableau 5 : Table de vérité du décodeur SN54154



CODE ASCII

CARACTERE MS / LS CARACTERE				b7 b6 b5	000	001	010	011	100	101	110	111
b4	b3	b2	b1		0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0		P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EDT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	//
1	1	0	1	13	CR	GS	-	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

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

- Code d'instruction du 6800 (1).

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

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

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

LEGEND

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

CONDITION CODE REGISTER NOTES

(Bit set if test is true and cleared otherwise)

① (Bit V) Test: Result = 10000000 ?

② (Bit C) Test: Result = 00000000 ?

③ (Bit C) Test: Decimal value of most significant BCD Character greater than nine ? (Not cleared if previously set)

④ (Bit V) Test: Operand = 10000000 prior to execution ?

⑤ (Bit V) Test: Operand = 01111111 prior to execution ?

⑥ (Bit V) Test: Set equal to result of N⊕C after shift has occured

⑦ (Bit N) Test: Sign bit of most significant (MS) byte of result = 1 ?

⑧ (Bit V) Test: 2's complement overflow from subtraction of LS bytes ?

⑨ (Bit N) Test: Result less than zero ? (Bit 15 = 1)

⑩ (All) Load Condition Code Register from Stack. (See Special Operations)

⑪ (Bit I) Set when interrupt occurs. If previously set, a Non Maskable interrupt is required to exit the wait state.

⑫ (ALL) Set according to the contents of Accumulator A

Bibliographie:

M.AUMIAUX

-Les systèmes à microprocesseurs. MASSON(1980)

-Au coeur des microprocesseurs.

EFCIS(THOMSON-CSF)

-Microprocesseurs et Mémoires catalogue 1980.

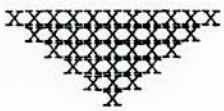
-document technique du M68SFDC Motorola 1977.

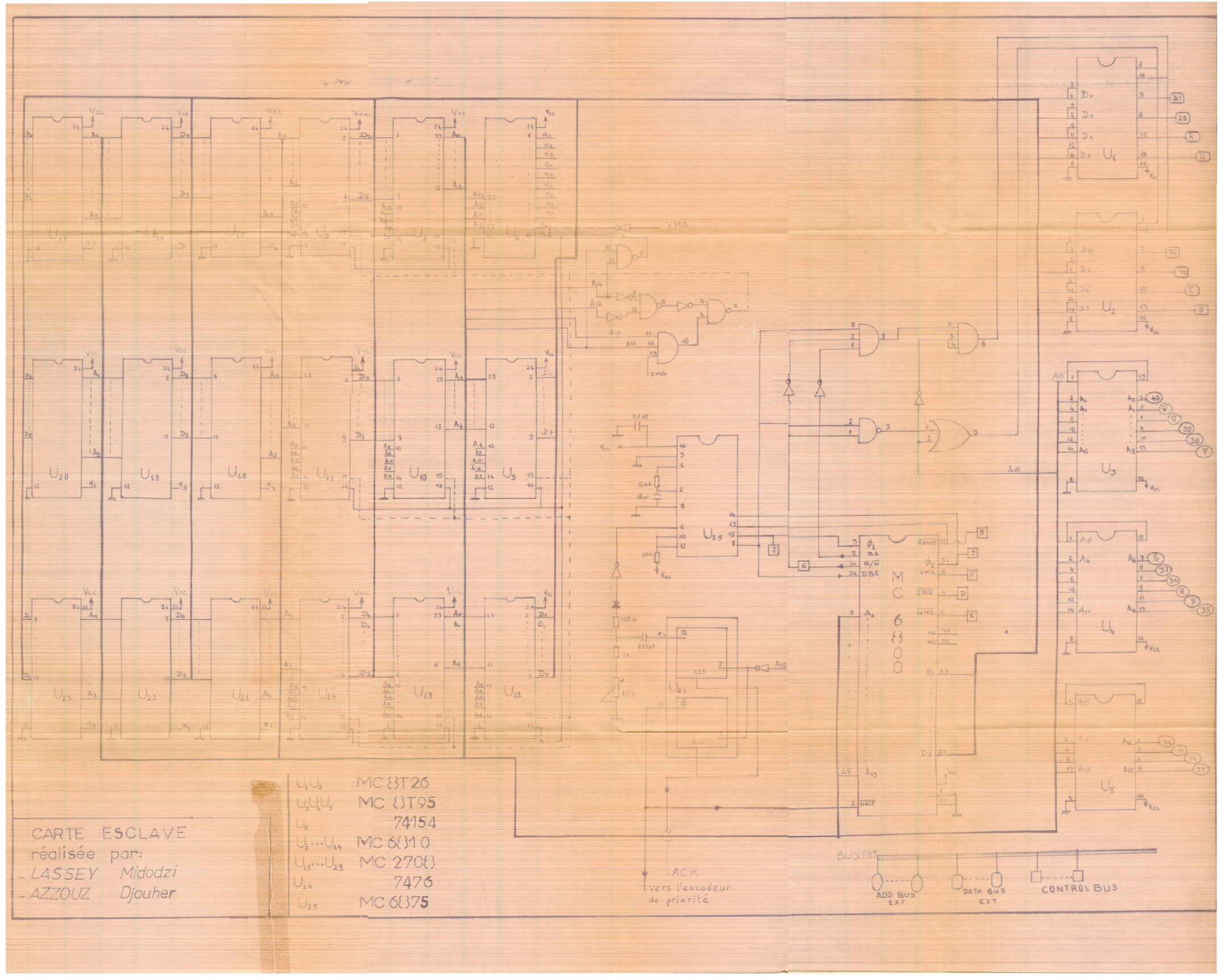
THESES:

-Mise en ligne d'un enregistreur potentiometrique 6 voies.sur un calculateur.

-Etudes des entrées/sorties d'un microordinateur et applications.

-Etude d'un système multiprocesseur à ressources partagées.




CARTE ESCLAVE
réalisée par:
- LASSEY Midodzi
- AZZOUZ Djouher
U1 U2 MC 68T26
U3 U4 U5 MC 68T95
U6 74154
U7 ... U44 MC 6810
U15 ... U23 MC 2708
U24 7476
U25 MC 6875
VMA
ACK
vers l'encodeur de priorité
MC 6800
BUS EXT
ADD BUS EXT
DATA BUS EXT
CONTROL BUS