Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie d'Alger

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTECHNIQUE

PROJET DE FIN D'ETUDES

INGENIORAT D'ETAT EN ELECTRONIQUE

Procédure de mise au point de Systèmes à Microprocesseurs sur le Tektronix 8002 A

Proposé et suivi par :
A. FARAH

Etudié par :
Chérif AMER YAHIA
et
Abdelmadjid KHENNICHE

JUIN 81

A nos familles.

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie d'Alger

ECOLE NATIONALE POLYTECHNIQUE

DEPARTEMENT D'ELECTRONIQUE ET D'ELECTROTECHNIQUE

PROJET DE FIN D'ETUDES

INGENIORAT D'ETAT EN ELECTRONIQUE

Procédure de mise au point de Systèmes à Microprocesseurs sur le Tektronix 8002 A

Proposé et suivi par :

A. FARAH

Etudie par :

Chérif AMER YAHIA

et

Abdelmadjid KHENNICHE

JUIN 81

- T A B L E D E S M A T I E R E S -

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

Les systèmes à base de microprocesseurs sont de plus en plus appelés à être utilisés dans de nombreuses applications, notamment dans les domaines de l'informatique, de l'instrumentation, de contrôles,... La conception de tels systèmes nécessite une réalisation matérielle et l'élaboration d'un logiciel gérant le fonctionnement de celle-ci. Le TEKTRONIX 8002A est un outil aidant à la mise au point de ces systèmes; il permet de suivre toutes les phases de réalisation, depuis l'écriture du programme jusqu'au produit fini en passant par toutes les étapes de test et d'intégration du logiciel dans le matériel.

Le but de notre projet est de donner la procédure de mise au point de maquettes à microprocesseurs sur le TEKTRONIX 8002A. Dans le premier chapitre où la lecture des schémas est nécessaire pour la compréhension, on a essayé d'exposer la structure et le fonctionnement du microprocesseur MC6800 de MOTOROLA. Le second chapitre présente le microordinateur: TEKTRONIX 8002A, ses modules de gestion et la méthode de mise au point de logiciels. Cette méthode est illustrée, au troisième chapitre, par des programmes exemples. Le chapitre IV comporte une brève étude d'un composant (PIA MC682I) permettant l'échange d'informations entre le microprocesseur et le milieu extérieur. Le cinquième chapitre montre un exemple d'utilisation du MC6800 dans un cas relativement simple: la gestion des feux de signalisation à un carrefour routier. La réalisation matérielle et l'élaboration du logiciel correspondant sont décrites dans ce chapitre.

CHAPITRE PREMIER

- E T U D E D U M C 6 8 0 0 -

I. Généralités

II. Structure de base

III. Modes d'adressage

IV. Exécution d'une séquence d'instructions

I. GENERALITES.

Le microprocesseur MC6800 de MOTOROLA est un composant en technologie LSI(Large Scale Integration = Intégration à grande échelle). Il est monolithique(en un seul boitier) et réalise les fonctions d'une unité centrale, c'est à dire d'une unité arithmétique et logique et d'un module de commande à l'aide de programmes lus en mémoire et au rythme des impulsions d'une horloge.

D'une manière physique, il est réalisé sur une pastille(puce) de silicium, rectangulaire, de quelques millimètres de côté, sur laquelle sont implantés les transistors et autres éléments du circuit logique; cette pastille est montée sur un boitier muni de 40 broches.

La réalisation d'un système minimum autour du MC6800 nécessite l'adjonction :

- de mémoires vive(RAM:MC68I0) et morte(ROM:MC6830);
- de circuits d'entrée/sortie qui assurent l'interface entre le microprocesseur et les organes périphériques tels le PIA:MC682I utilisé pour la transmission de données en mode parallèle et/ou l'ACIA:MC6850 utilisé pour la transmission de données en mode série;
- de bus de liaisons.

Le MC6800 travaille sur des mots de 8 bits, a une capacité d'adressage de 64 Koctets et possède 72 instructions de longueurs variables et 6 modes d'adressage: direct, immédiat, indexé, étendu, relatif et inhérent(implicite ou accumulateur).

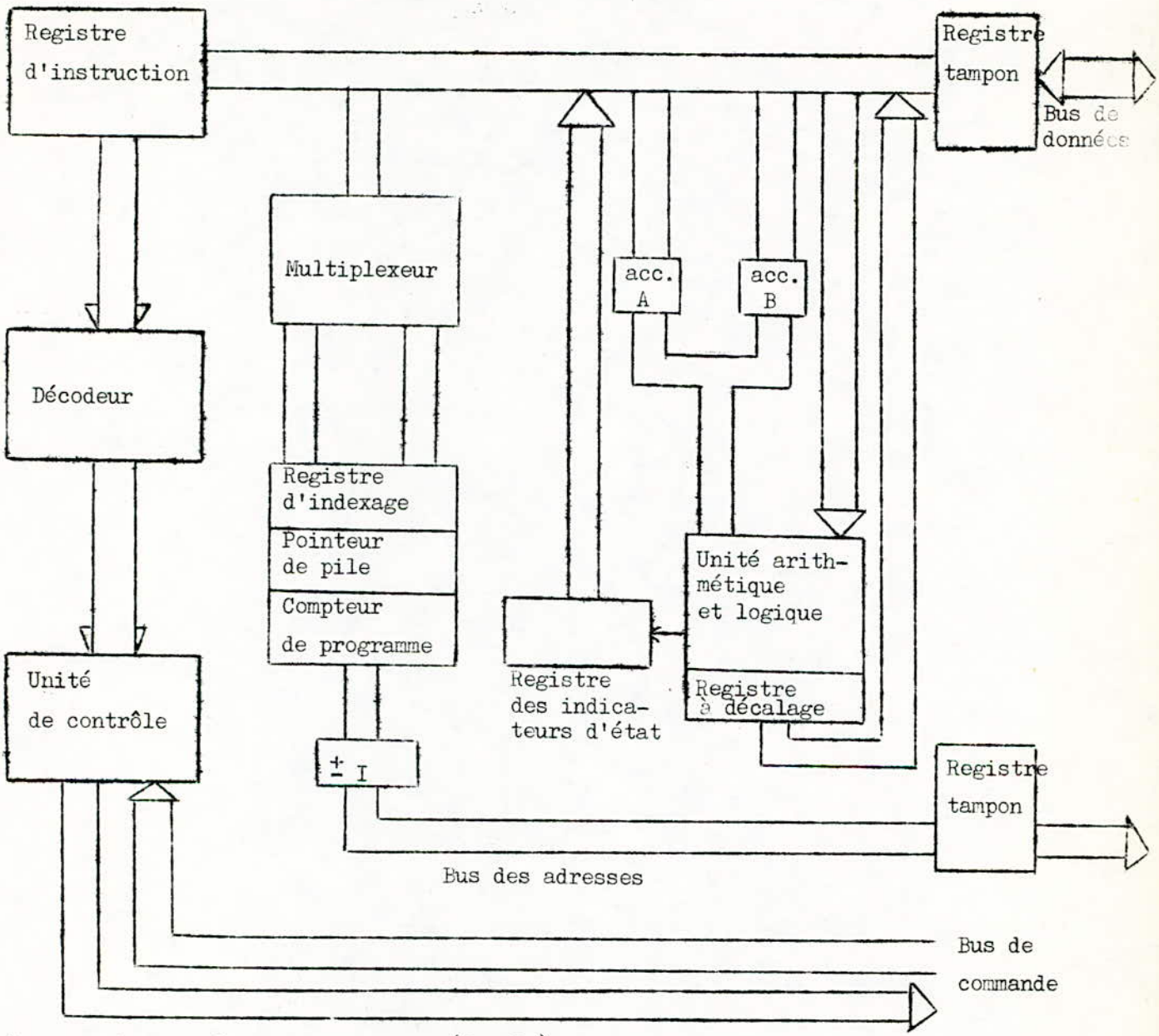
II. STRUCTURE DE BASE.

I°) Organisation interne du MC6800 (figure I).

A la partie supérieure du synoptique, se trouve le bus interne de données. A droite, c'est l'unité arithmétique et logique équipée de deux accumulateurs A et B. A sa sortie, se trouve un registre à décalage, à sa gauche apparait le registre des indicateurs d'état. Au milieu, on remarque le registre d'indexage, en dessous de celui-ci apparaissent les deux registres doubles d'adresses, le pointeur de pile et le pointeur de programme. En dessous de celui-ci figure symboliquement un incrémenteur-décrémenteur pour indiquer le fait que les pointeurs de pile et de programme peuvent-être incrémentés ou décrémentés de manière interne. Ces trois registres sont reliés au bus des adresses. A la gauche de l'illustration apparait le registre d'instructions suivi du décodeur et de l'unité de commande qui assure le séquencement du système. C'est de ce module de commande qu'est issu le bus de commande qui se propage vers la droite de la figure.

Après avoir donné les principaux éléments qui équipent ce microprocesseur standard de technologie NMOS, on va définir le rôle de chacun:

Fig. I Organisation interne du MC6800



a) L'unité arithmétique et logique: (U.A.L.)

C'est l'ensemble des circuits combinatoires capables d'effectuer les opérations arithmétiques et logiques.

b) Les accumulateurs A et B:

Ce sont des registres à 8 bits où se rangent les données intermédiaires en cours de traitement. Ils communiquent avec l'unité arithmétique et logique et les mémoires de données. Ils peuvent également servir de registres à décalage.

c) Le registre à décalage :

Il permet de décaler d'un bit le contenu d'un mot, vers la gauche ou vers la droite.

d) Le registre d'indexage : (IX : Index Register)

Ce registre facilite l'adressage de la mémoire. L'accès au contenu de tableaux emmagasinés dans cette mémoire pourra se faire en une seule instruction.

e) Le pointeur de pile : (SP : Stack Pointer)

Avant de définir le pointeur de pile, on expliquera le rôle d'une pile.

La pile se compose d'un nombre quelconque d'octets en mémoire RAM. Elle permet la mémorisation temporaire et la restitution d'octets successifs d'informations. Celles-ci peuvent représenter l'état actuel du microprocesseur, une adresse de retour ou des données. La pile peut-être utilisée pour une liaison avec un sous programme, une commande d'interruption, le stockage temporaire de donnéesindiqué par le programme ou pour les programmes réentrants.Cette pile est du type L.I.F.O.(Last In-First Out = dernier entré-premier sorti).

Le pointeur de pile, dont la capacité est de I6 bits, contient l'adresse indiquant la première case vide située au sommet de la pile.

La figure ci-dessous nous donne le fonctionnement d'une pile en mémoire.

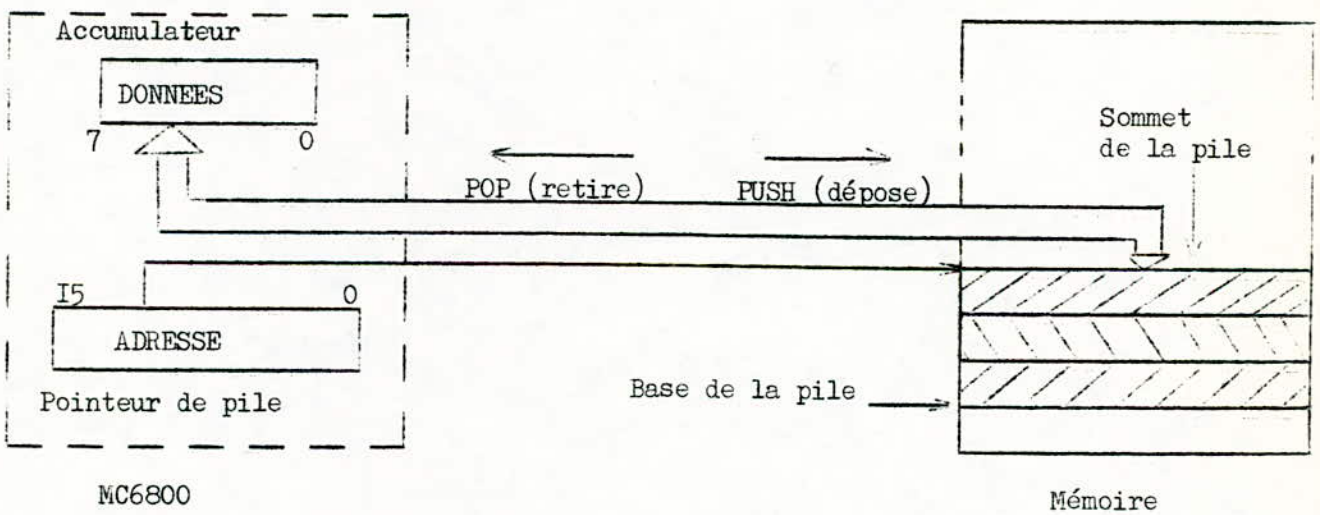


Fig. 2 Fonctionnement d'une pile en mémoire.

Le transfert de données se fait toujours entre l'accumulateur et le sommet de la pile; on peut,soit déposer un mot au sommet de la pile, soit retirer un mot de son sommet. Le pointeur de pile contient l'adresse du haut de la pile. Cette pile est dite pile "SOFT", c'est à dire programmée.

f) Le compteur de programme ou compteur ordinal: (PC:Program Counter)

Ce registre contient toujours l'adresse de l'instruction suivante qui doit-être

exécutée, cette instruction est propagée vers la mémoire via le bus des adresses. La progression du compteur ordinal est fonction de la longueur des instructions traduites.

g) Le registre des indicateurs d'état ou registre code condition: (CCR:Code Condition Register)

C'est un registre à 8 bits qui permet de disposer de 6 informations concernant la gestion du programme. Les 6 indicateurs d'état (C, V, Z, N, I, H) occupent les bits 0 à 5 et sont affectés par les instructions en cours, les résultats des instructions et les interruptions. Les bits 6 et 7 sont systématiquement mis à I.

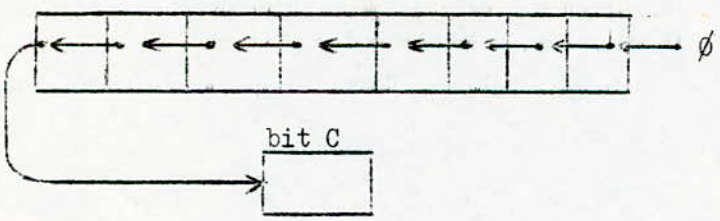
Ci-dessous apparait l'octet correspondant au registre d'état:

1	1	H	I	N	Z	V	C
7	6	5	4	3	2	1	0

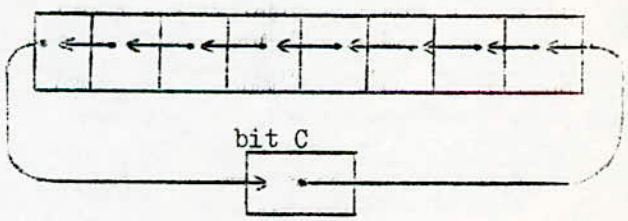
Rôle des indicateurs d'état:

- C: retenue (carry)

Le bit C est utilisé à deux fins; quand le résultat d'une addition donne une retenue, le neuvième bit est mis dans C. Dans le cas d'une rotation ou d'un décalage à droite, le bit le moins significatif passe dans C. Pour une rotation ou un décalage à gauche, c'est le bit le plus significatif qui est emmagasiné dans C.



Décalage à gauche



Rotation à gauche

- V: dépassement ou débordement (overflow)

L'indicateur de débordement est testé lors d'opérations arithmétiques faisant appel à la notation deux-complémenté, car il se pourrait qu'il y ait propagation d'une retenue dans le bit du signe, ceci est indiqué au programme par une mise à I de V.

- Z: zéro

Le bit Z est utilisé par les instructions test lors d'opérations arithmétiques ou logiques. Il estmis à I si le résultat est nul, et à 0 s'il est non nul.

- N:négatif(négative)

Il s'agit du bit du signe qui est mis à I pour indiquer un résultat négatif et à 0 pour indiquer un résultat positif, et celà lors d'opérations arithmétiques en 2-complément.

- I :masquage d'interruption (interrupt mask)

Ce bit est utilisé uniquement lors d'interruptions masquables. Il est mis à I pour éviter la prise en compte d'autres interruptions et à 0 pour accepter une demande d'interruption.

- H :demie-retenue (half carry)

Ce bit est utilisé seulement dans le cadre d'opérations B.C.D. (code binaire décimal). Il détecte la retenue du bit 3 vers le bit 4.

h) Le multiplexeur:

Il permet la communication, en deux temps, des quantités de I6 bits par l'intermédiaire de bus à 8 bits.

i) Leregistre d'instructions:

C'est dans ce registre qu'est mémorisée chaque instruction pendant le temps nécessaire à son exécution.

j) L'unité de contrôle:

C'est un ensemble de circuits logiques qui assure le séquencement des instructions et les envoie à l'unité arithmétique et logique et aux différents registres pour leur exécution, au rythme des impulsions d'une horloge.

k) Le décodeur:

Il permet le décodage des bits de l'instruction.

l) Les bus internes:

Ces bus font communiquer les différents éléments du microprocesseur.

2°) Bus externes et "buffers" d'entrées/sorties.

a) Un bus est un ensemble de lignes de communications groupées par fonction.

Sur le MC6800, on distingue trois bus externes:

- Le bus des données (data bus)

Il est bidirectionnel, à 8 bits, et permet le transfert de données entre le microprocesseur et les éléments externes tels la mémoire et les périphériques.

- Le bus des adresses (bus address)

Il est monodirectionnel, à I6 bits, et permet l'adressage de 2^{I6} registres externes, il est en liaison directe avec le pointeur de programme.

- Le bus de commande (bus control)

Ce bus permet la transmission de signaux de synchronisation et de contrôle du microprocesseur vers le milieu extérieur et inversement. Ces signaux sont les signaux d'état, de lecture, d'écriture, d'interruption, de commande des entrées-sorties, d'horloge,...

b) Connectés sur les bus externes, les buffers d'entrées/sorties servent de tampons entre le microprocesseur et l'extérieur.

3°) Brochage du MC6800.

La figure 3 nous donne un aperçu sur le brochage du 6800 de MOTOROLA:/

I6 broches sont réservées pour le bus des adresses(A0-AI5), 8 pour le bus des données(D0-D7), I pour l'alimentation(5Volts) et 2 pour la masse. 2 broches(NC)ne sont pas connectées. Le MC6800 est également équipé de 9 broches de contrôle et 2 broches de synchronisation. On définira ci-dessous le rôle de chacune de ces II broches.

a) Broche $\overline{\text{RESET}}$(remise au point initial)

Elle permet lors du passage à l'état bas , de remettre à 0 les différents registres du microprocesseur. Le compteur ordinal est chargé par le contenu des deux derniers octets dela mémoire(FFFE et FFFF) qui contiennent l'adresse du début du programme d'initialisation dès que $\overline{\text{RESET}}$ passe à l'état haut, le bit de masquage d'interruption est mis à I.

b) Broche $\overline{\text{NMI}}$ (interruption non masquable)

Le signal arrivant sur la broche $\overline{\text{NMI}}$ est destiné à interrompre le cours d'un programme; dès l'apparition de ce signal, le MC6800 termine l'instruction en cours et prend en compte cette interruption en sauvegardant au préalable le contenu de tous ses registres dans la pile, en chargeant le compteur de programme avec le contenu des adresses FFFC et FFFD.

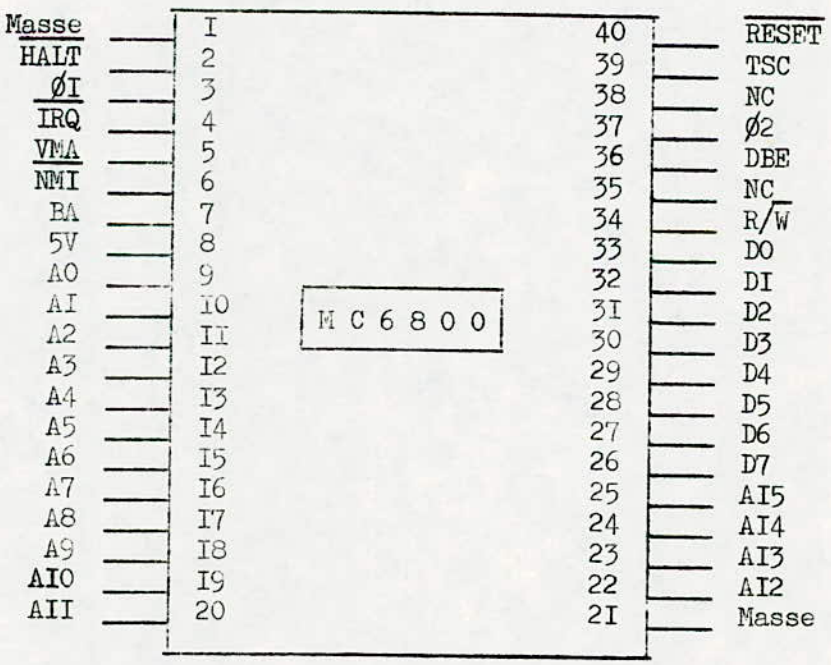


Fig. 3 Brochage du MC6800.

Le bit I du registre code condition est mis à "I" pour éviter la prise en compte d'autres interruptions. Le microprocesseur commence alors l'exécution de la séquence d'interruptions en commençant par l'instruction adressée par le compteur programme. $\overline{NMI}$ est une interruption prioritaire et ne peut-être masquée.

c) Broche $\overline{IRQ}$ (demande d'interruption)

La ligne $\overline{IRQ}$ nous renseigne sur une demande éventuelle d'interruption provenant d'un périphérique; elle est signalée par un niveau bas sur la broche. Cette interruption est masquable par programme, elle n'est prise en compte que si le bit I est à 0. Si c'est le cas, le microprocesseur termine l'instruction en cours, sauvegarde le contenu de ses registres, met à "I" le bit I et charge le compteur programme avec le contenu des octets mémoire d'adresses FFF8 et FFF9 représentant l'adresse contenue dans le pointeur d'interruption masquable.

d) Broche $\overline{HALT}$ (arrêt)

Quand l'entrée $\overline{HALT}$ passe à l'état bas, le microprocesseur passe à l'état arrêt. Toute exécution de programme est arrêtée. Cependant, si une interruption $\overline{IRQ}$ ou $\overline{NMI}$ se produit, elle sera conservée dans le MC6800 et sera prise en compte dès que celui-ci sort de l'état arrêt. Si par contre $\overline{RESET}$ arrive, il y aura changement de l'état de certaines lignes du microprocesseur: les lignes VMA et BA passeront au niveau bas(niveau 0), le bus de données passera à l'état haute impédance, la ligne R/$\overline{W}$ sera àl'état de lecture, le bus d'adresse contiendra l'adresse FFFE tant que $\overline{HALT}$ sera à l'état bas. Le microprocesseur commencera à lire le contenu des adresses FFFE et FFFF dès que $\overline{HALT}$ passe à l'état haut.

e) Broche BA (bus disponible)

L'état haut de la ligne BA indique que le microprocesseur est arrêté et que le bus d'adresses est disponible. Ceci a lieu quand le MC6800 est dans l'état d'attente d'interruption après l'exécution de l'instruction WAIT, ou si $\overline{HALT}$ est à l'état bas.

f) Broche TSC (contrôle de 3 états)

Mis à l'état haut, ce signal met les lignes adresses et la ligne R/$\overline{W}$ à l'état haute impédance, les lignes VMA et BA sont forcées au niveau logique "0" pour éviter la lecture ou l'écriture intempestive de tout circuit activé par VMA.

g) Broche VMA (adresse mémoire valide)

Cette broche active les circuits mémoire et les circuits d'interface. Mise à "I", elle indique que le bus d'adresse est positionné.

h) Broche R/$\overline{W}$ (lecture/écriture)

Elle est utilisée pour contrôler la nature des échanges de données sur le bus des données du microprocesseur. Quand cette ligne est à l'état haut, on est en mode de lecture; si elle est à l'état bas, on est en mode d'écriture.

i) Broche DBE (mise en service du bus de données)

C'est un signal de commande "trois états" pour le bus de données. Mis à"I", il permet au bus de données, le transfert de données. Mis à "0", le bus de données sera à l'état

haute impédance.

j) Broches d'horloge ØI et Ø2

Le MC6800 possède 2 broches d'horloge ØI et Ø2 qui sont les deux phases d' une même horloge.

- ØI sert à activer le microprocesseur pour des fréquences comprises entre IKhz et IMhz.
- Ø2 synchronise le transfert de données.

III. MODES D'ADRESSAGE.

L'instruction repère l'opérande de différentes manières appelées "modes d'adressage". Le code de l'instruction diffère suivant le mode employé.

On citera les différents modes d'adressage qu'utilise le 6800 de MOTOROLA en illustrant les définitions par des exemples.

I°) Adressage immédiat

Le mot suivant le code opératoire contient une donnée à 8 ou I6 bits directement utilisable par le microprocesseur. L'instruction nécessite donc 2 à 3 octets. Le mot n'est pas contenu dans la mémoire de données mais dans la mémoire programme.

Exemple :

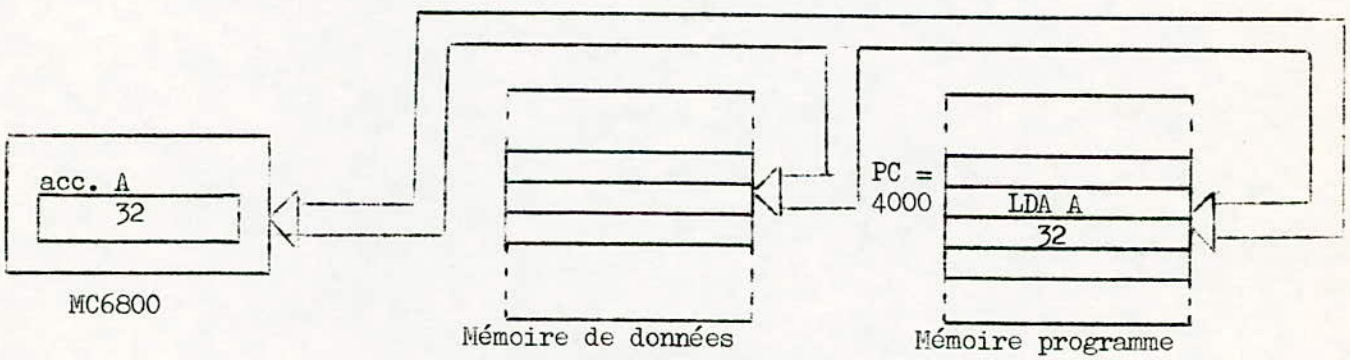


Fig. 4 Adressage immédiat.

" LDA A ##32 " signifie charger l'accumulateur A avec la donnée 32. Cette donnée est mise dans la mémoire programme et par l'intermédiaire du code opératoire " LDA A ", elle est transférée dans l'accumulateur A.

PC = 4000 indique que pour cette instruction, le pointeur de programme est positionné à l'adresse 4000.

Le caractère # signifie l'adressage immédiat.

2°) Adressage direct

Il permet d'adresser des positions mémoire comprises entre 0 et 255. L'instruction

ne prend que 2 octets.

Exemple:

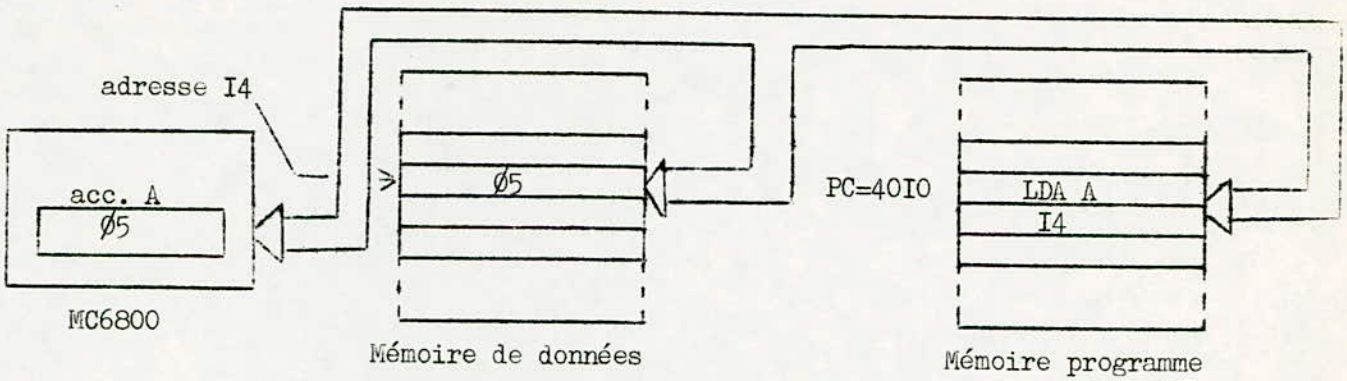


Fig. 5 Adressage direct.

" LDA A I4 " signifie charger dans l'accumulateur A la donnée Ø5 située à l'adresse I4 de la mémoire de données.

3°) Adressage indexé

L'adresse de l'opérande est obtenue en ajoutant le contenu du registre d'indexage au mot qui suit le code opératoire. Cemot est inférieur ou égal à 255, c'est àdire à ØFF en hexadécimal. L'instruction utilise 2 octets. Ce mode permet d'adresser toute la mémoire tout en économisant de la mémoire programme.

Exemple:

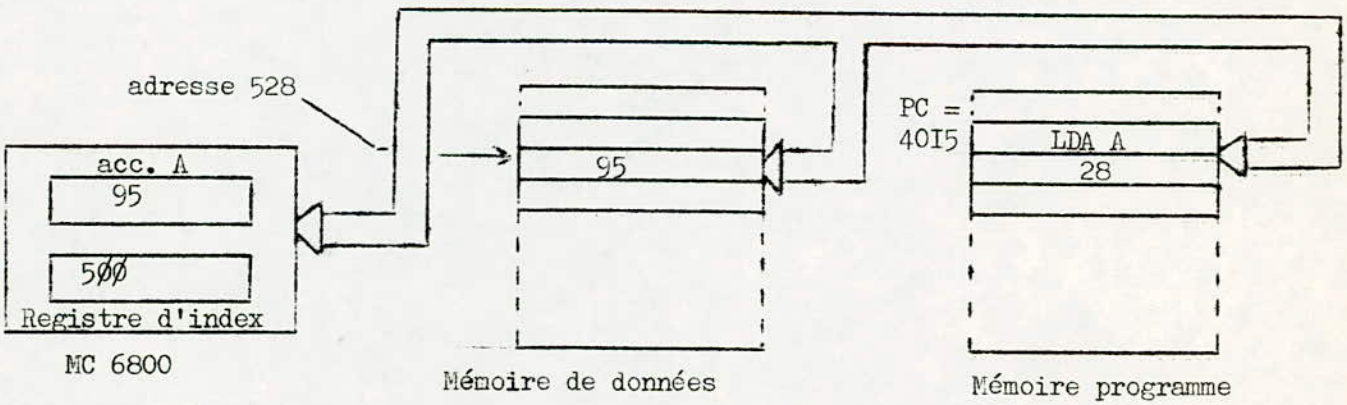


Fig. 6 Adressage indexé.

" LDA A 28,X " signifie charger la donnée 95, située dans la mémoire de données à l'adresse 500+28=528, dans l'accumulateur A. Le caractère X indique le mode d'adressage indexé; dans ce cas X = 500.

4°) Adressage étendu

Cet adressage nécessite 3 positions mémoire, la première contient le code opératoire, les deux dernières contiennent l'adresse de l'opérande. Ce mode ne diffère du mode d'adressage direct que par la longueur des instructions.

Exemple:

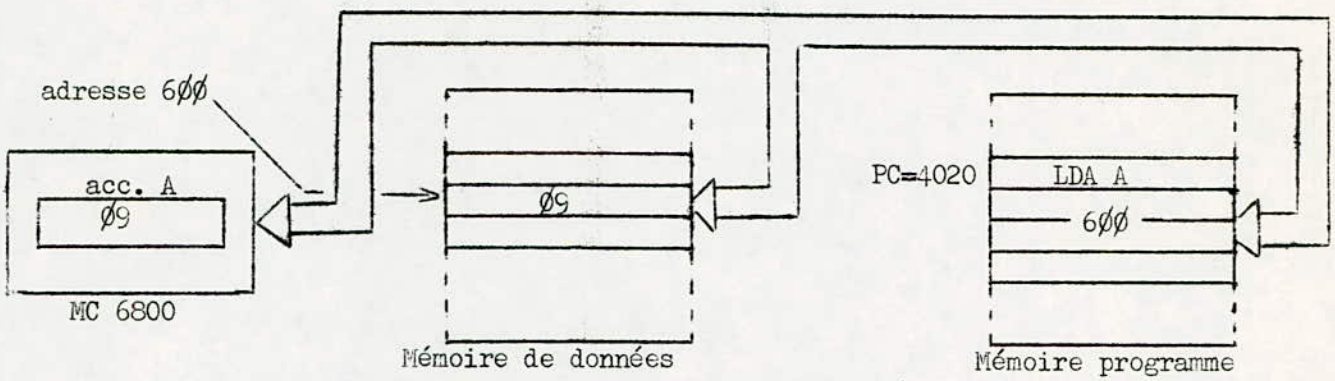


Fig. 7 Adressage étendu.

" LDA A 6ØØ " signifie charger l'accumulateur A avec le contenu Ø9 de la mémoire de données situé à l'adresse 6ØØ; cette adresse prend deux octets dans la mémoire programme.

5°) Adressage relatif

Cet adressage permet le positionnement du pointeur de programme sur la case mémoire désirée. Ce positionnement est obtenu en ajoutant au contenu du pointeur de programme un déplacement limité à I26 positions mémoire vers l'avant et à I27 vers l'arrière. Cet adressage est utilisé uniquement dans les instructions de branchement.

Exemple:

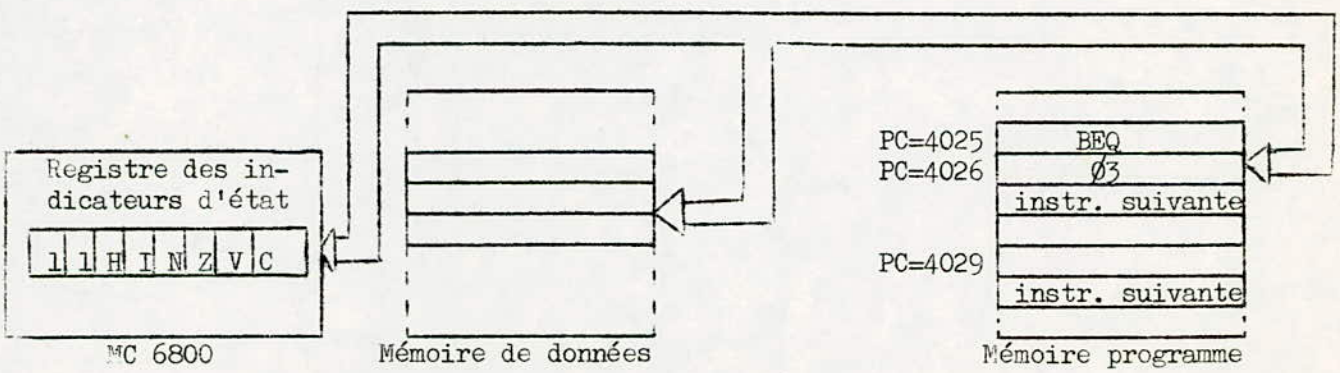


Fig. 8 Adressage relatif

" BEQ Ø3 "signifie brancher si c'est égal à Ø, à l'adresse indiquée par le pointeur de ~~pile~~ programme, c'est à dire à 4026 + 3 = 4029.

6°) Adressage inhérent (accumulateur ou implicite)

C'est l'un des adressages les plus simples. Il se refère en général aux accumulateurs A et B, à la pile et aux registres d'indexage et des indicateurs d'état. L'instruction n'utilise qu'un seul octet.

Exemples:

- " ABA " signifie ajouter au contenu de l'accumulateur A le contenu de l'accumulateur B et garder le résultat dans l'accumulateur A.
- "INX " signifie incrémenter le registre d'indexage.
- " DES " signifie décrémenter le pointeur de pile.
- " CLC " signifie mettre ∅ dans le bit "retenue" du registre des indicateurs d'état.

IV. EXECUTION D'UNE SEQUENCE D'INSTRUCTIONS.

Après avoir parlé de l'organisation interne du MC6800, on étudiera maintenant l'exécution et l'enchainement des instructions d'un programme en sachant toutefois que chaque opération du microprocesseur est successivement validée par un signal périodique d'horloge.

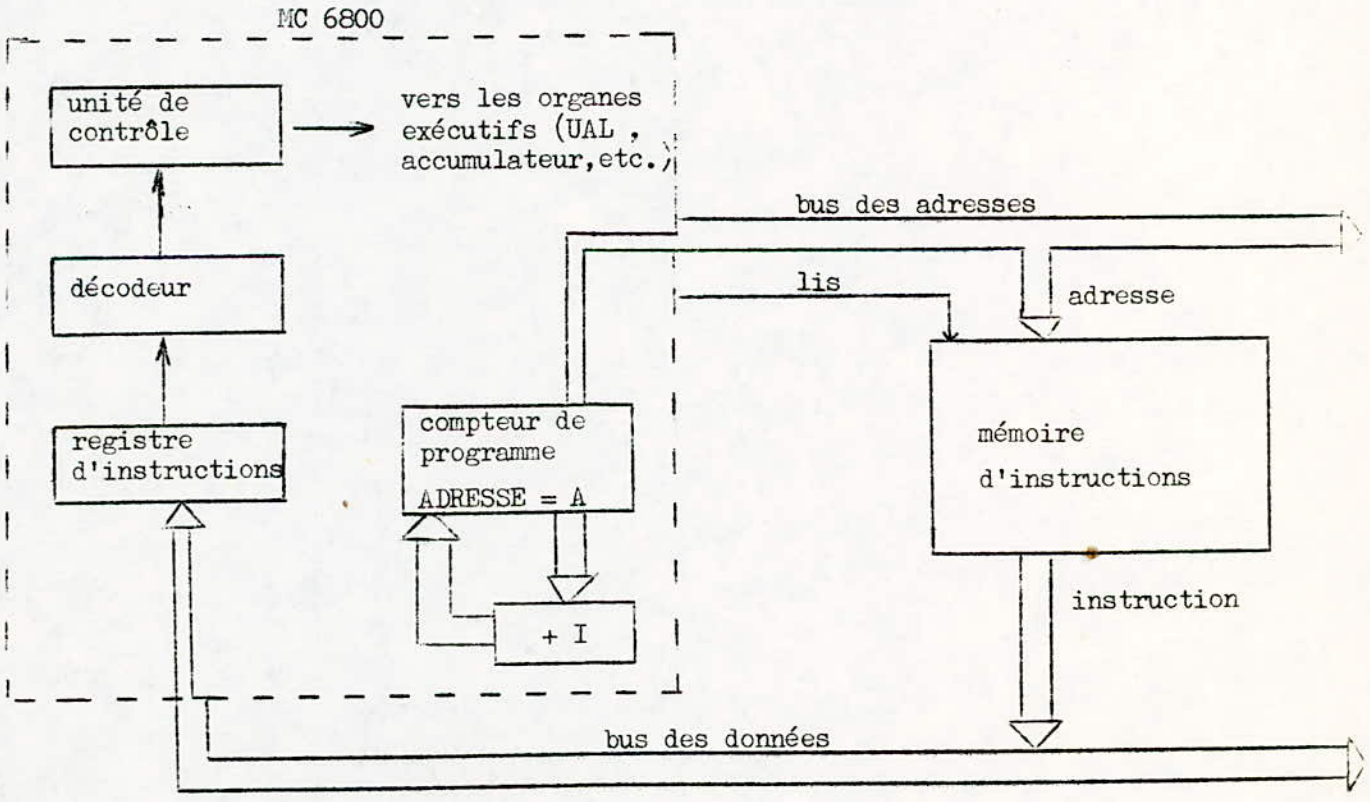


Fig. 9 Exécution d'une séquence d'instructions.

A un instant donné, le compteur de programme contient une adresse A, cette adresse est envoyée vers la mémoire d'instructions à travers le bus des adresses. L'instruction ainsi adressée est transférée dans le registre d'instructions à travers le bus des données et, au même moment, le compteur de programme est incrémenté et contient ainsi l'adresse suivante A + I. Cette première phase concerne " la recherche d'instruction ", elle est appelée FETCH (aller chercher).

La deuxième phase concerne le décodage des bits de l'instruction.

Après que l'unité de contrôle ait interprété ces bits décodés, arrive la troisième et dernière phase qui consiste à exécuter l'instruction par l'intermédiaire des organes exécutifs tels l'unité arithmétique et logique et les accumulateurs.

C'est maintenant au tour de l'instruction suivante, dont l'adresse " A + I " est contenue dans le compteur, à subir l'exécution.

Remarques:

-S'il s'agit d'une instruction à 2 ou 3 octets, l'unité de contrôle assure la totalité de ce séquencement.

-S'il s'agit d'une instruction de branchement, le compteur de programme contiendra l'adresse de l'instruction où l'on se branche.

CHAPITRE II

-ETUDE DU TEKTRONIX 8002A-

I. Présentation du système de développement pour microprocesseurs : TEKTRONIX 8002A

II. Présentation de TEKDOS

III. Ecriture d'un programme source

IV. L'éditeur de texte

V. Le processeur assembleur

VI. Déverminage

VII. L'éditeur de lien

VIII. Service call (SVC)

IX. Analyseur temps réel (RTPA)

I. PRESENTATION DU SYSTEME DE DEVELOPPEMENT POUR MICROPROCESSEURS: TEKTRONIX 8002A.

I°) Introduction.

Un système de développement (Microprocessor Developpement Aid: MDA) est un outil essentiel pour la réalisation d'un prototype de produits à base de microprocesseurs. Il permet la création du logiciel et la mise au point du matériel correspondant. Pour localiser les erreurs d'un programme utilisateur, lors de son exécution, le MDA utilis soit la simulation, soit l'émulation. Un MDA basé sur l'émulation contient un modèle matériel du microprocesseur prototype. Le TEKTRONIX 8002A microprocessor LAB SYSTEM est un système de développement universel (il supporte plusieurs types de microprocesseurs), il utilise l'émulation par substitution, c'est à dire qu'il contient un microprocesseur émulateur identique au microprocesseur du prototype. En simulation, un logiciel lit les instructions et les exécute comme le ferait le microprocesseur du prototype, mais à une vitesse moindre, donc l'avantage de l'émulation par rapport à la simulation est l'exécution d'un programme en temps réel.

2°) Démarche à suivre pour développer un système.

La création d'un nouveau produit nécessite le choix du microprocesseur, actuellement, le 8002A de l'école ne dispose que du MC6800. Les performances de chaque microprocesseur sont évaluées en testant le jeu d'instructions, en comparant les temps d'exécution et les possibilités d'entrée/sortie. Ensuite on déterminera, pour le nouveau produit, ce qui sera logiciel et ce qui sera matériel en tenant compte des défauts et des qualités du microprocesseur choisi.

Le développement de la partie " hard " de la maquette prototype se fait par étapes en parallèle avec la mise au point des programmes correspondants. Tout circuit réalisé est connecté au processeur émulateur via la sonde, puis testé à l'aide de programmes stockés sur disque. Les programmes conçus pour le fonctionnement du prototype sont exécutés sous contrôle d'un programme de déverminage qui vérifie leur logique. Quand tous les différents modules matériels et logiciels sont testés et éventuellement corrigés, on câble toute la maquette et charge le logiciel en entier pour un test global. Si le fonctionnement de l'ensemble est satisfaisant, on charge le code objet dans la mémoire du prototype et on remplace la sonde par un microprocesseur indépendant du système (figure I0).

3°) Organisation matérielle et logicielle du TEKTRONIX 8002A.

a) Le matériel (Hard):

Le 8002A microprocessor LAB SYSTEM, représenté en synoptique (figure II), est

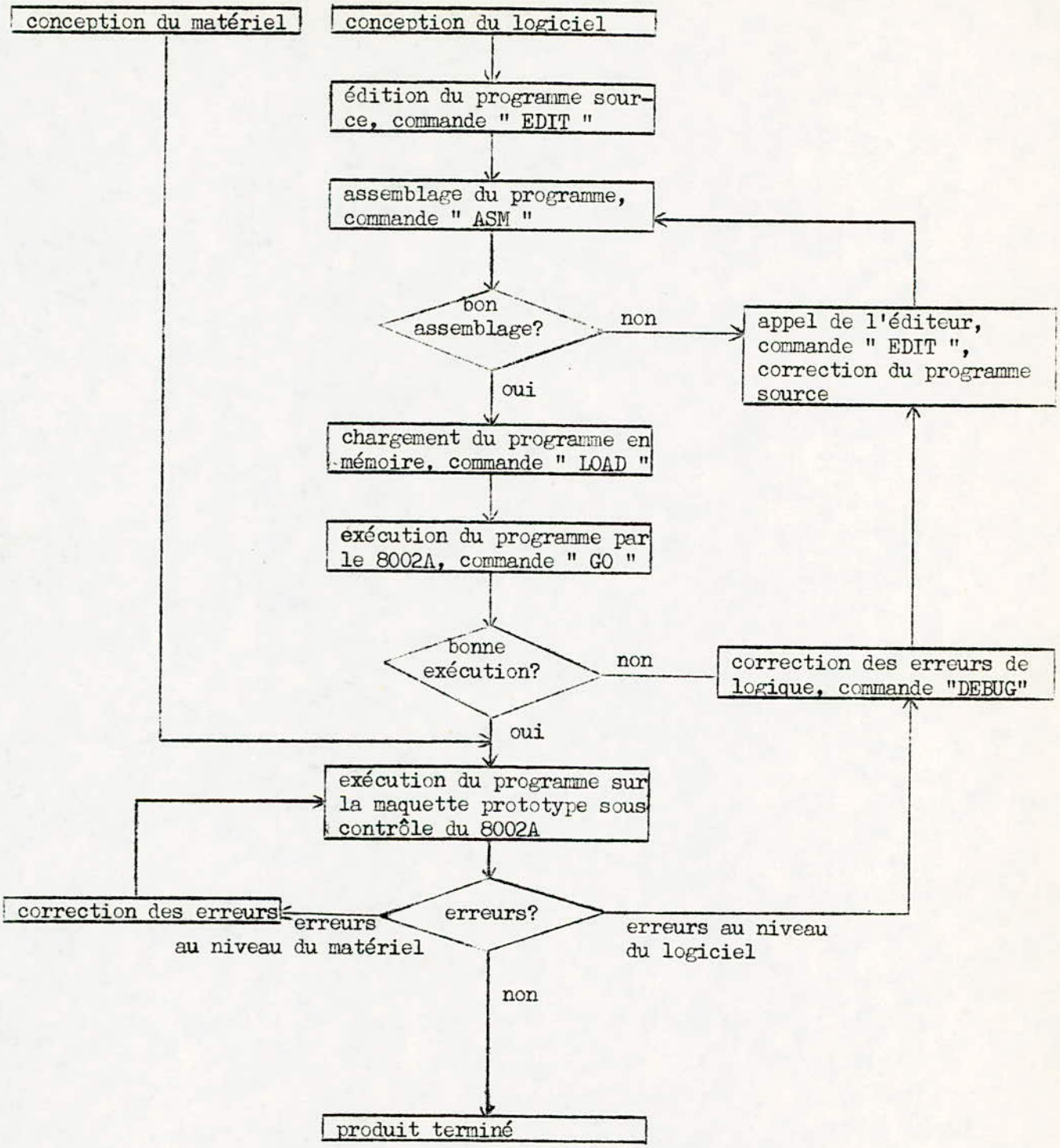


Fig. I0 Organigramme schématisant la procédure de mise au point de maquettes à microprocesseurs sur le TEKTRONIX 8002A.

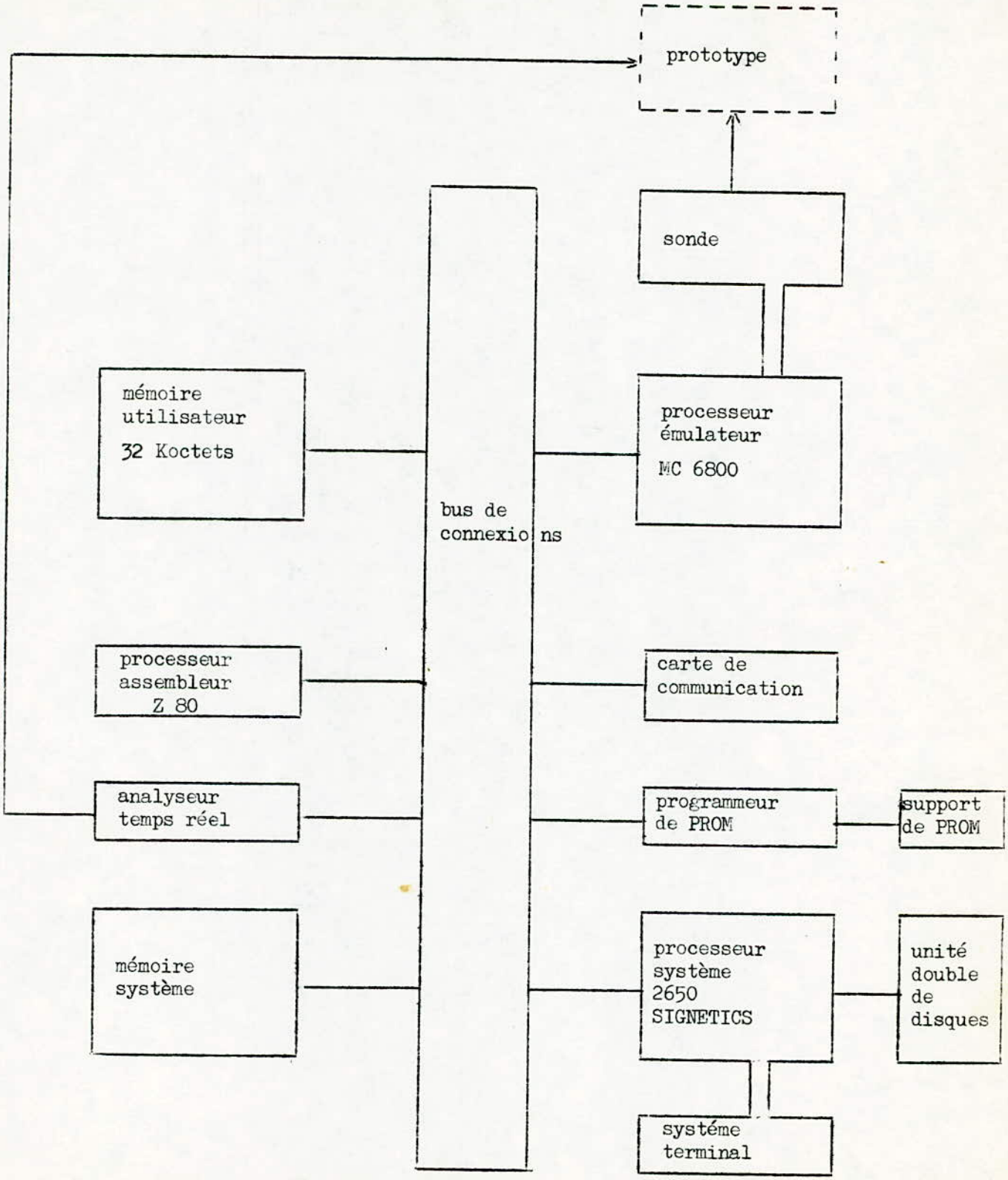


Fig. II Organisation matérielle du TEKTRONIX 8002A.

conçu autour d'un processeur système (le 2650 de SIGNETICS) utilisant d'autres microprocesseurs, et assurant la programmation de PROM, le déverminage et la gestion d'informations. En plus du 2650, le MDA 8002A est également équipé:

- d'un processeur assembleur qui permet d'exécuter l'assemblage et l'édition de lien . Le processeur utilisé est le Z80 choisi pour son jeu d'instructions puissant et sa rapidité d'exécution.

-d'un processeur émulateur qui exécute et assure le déverminage des programmes utilisateurs. Plusieurs cartes processeur-émulateur différentes peuvent exister au sein du 8002A.

-d'une mémoire système composée d'une RAM de I6Koctets et d'une PROM occupant les 256 premiers octets. Le chargeur résidant dans la PROM permet le transfert des sous-modules et certaines commandes de TEKDOS, du disque vers la RAM dès la mise sous tension.

-d'une mémoire utilisateur dont la capacité est 32Koctets. Elle contient les programmes exécutés par le MC6800.

-d'une sonde constituée de deux câbles plats, d'un circuit d'interface et d'une fiche mâle de 40 pattes. Elle assure le lien entre la maquette prototype et le processeur émulateur et contribue ainsi à l'opération de déverminage de la maquette.

-d'un analyseur temps réel composé d'une carte analyseur, d'un interface d'acquisition de données et d'une sonde huit canaux. Il permet de suivre d'une façon dynamique les bus d'adresses et de données ainsi que huit signaux de la maquette, de déceler les incompatibilités logiciel/matériel et les problèmes séquentiels.

-d'un programmeur de PROM utilisé pour le transfert du programme utilisateur de la mémoire programme vers une PROM (qui sera enfichée sur la maquette) et inversement. Il sert aussi pour la comparaison du contenu de la mémoire utilisateur avec celui d'une PROM. Le 8002A possède deux supports de PROM sur sa face avant.

-d'une unité double de disques souples qui constitue la mémoire de masse du 8002A. Elle comprend deux lecteurs/enregistreurs de disques, un contrôleur et une alimentation. Une liaison de type parallèle permet une communication directe entre le processeur système et l'unité de disques.

-d'une carte communication à trois lignes permettant l'échange d'informations entre le 8002A et d'autres systèmes informatiques.

-d'un système terminal constitué d'un clavier et d'une console de visualisation. D'autres équipements peuvent éventuellement faire partie du système terminal (imprimante, perforateur de carte, etc...).

b) Le logiciel (Soft) :

Le logiciel du système est constitué de plusieurs modules. On y trouve un logiciel de base: TEKDOS (Tektronix Disc Operating System), un éditeur de texte permettant la création de textes et les modifications éventuelles; le chargement et le transfert de fi-

chiers entre la mémoire programme et le disque sont assurés par TEKDOS. La conversion du programme source (écrit par l'utilisateur) en code objet exécutable par le microprocesseur se fait à l'aide d'un programme assembleur. La réunion en un seul code objet de plusieurs codesobjets assemblés séparément et pouvant échanger des variables entre eux, s'effectue grâce à l'éditeur de liens. Un programme de déverminage " DEBUG " permet de suivre l'exécution d'un programme, de déceler d'éventuelles erreurs de logique que le programmeur corrigera. Le chargement en mémoire vive d'un code objet à exécuter, le transfert d'un code objet entre la mémoire vive et le disque, le contrôle du contenu des mémoires se font à l'aide du programme d'émulation. Le TEKTRONIX 8002A possède également un programmeur de PROM et un logiciel d'utilisation de l'analyseur temps réel. Des possibilités de communication avec d'autres systèmes informatiques existent par le biais d'un logiciel spécifique et d'un logiciel convertisseur syntaxique.

4°) <u>Précautions à prendre avant la mise sous et hors tension du système</u>.

Avant d'insérer le disque, mettre sous tension :

- l'unité de disque (attendre cinq minutes environ pour permettre aux circuits électroniques d'atteindre une température stable);
- la console de visualisation;
- le 8002A; sa mise en marche entrainera une lecture automatique du disque drive Ø et charge TEKDOS dans la mémoire système.

Insérer le disque système dans le réceptacle de gauche (drive Ø) en s'assurant que l' étiquette collée sur ledisque est face à l'interrupteur POWER et qu'elle correspond à la dernière partie introduite dans le réceptacle.

Avant la mise hors tension du système, retirer le disque.

II. PRESENTATION DE TEKDOS.

Le moniteur du système appelé "TEKDOS" assure la gestion des disques et des fichiers, les fonctions de transfert des données et les fonctions de contrôle des périphériques et du système.

Une ligne de commande TEKDOS est composée du nom de la commande et de ses paramètres souvent spécifiés. Une commande est toujours séparée de son paramètre par un ou plusieurs espaces ou par une virgule. On ne fait entrer la ligne de commande que lorsque le caractère > est affiché. Un message indiquant la fin de l'exécution de la plupart des commandes TEKDOS apparait sur la console de visualisation. L'exécution de toute commande provoque l'affichage du caractère >.

Procédure d'utilisation de TEKDOS.

Après avoir mis sous-tension le système, et dès que TEKDOS est chargé dans la mémoire système, une sonnette retentit et un message d'accueil apparait sur la console:

```
>TEKDOS 68ØØ version 3.I
>-
```

68ØØ indique le type de processeur émulateur utilisé qui est le 68ØØ de MOTOROLA.
version 3.I est le numéro de mise à jour.
Le caractère > en dessous du message d'accueil indique que TEKDOS est prêt à accepter les commandes.

Pour aider le lecteur à se familiariser avec le système, on donnera ci-dessous quelques exemples utilisant les commandes TEKDOS.

a) Liste du répertoire du disque:

TEKDOS étant prêt àaccepter des commandes, on veut par exemple, afficher le répertoire du disque système. Il suffit de faire entrer la commande LDIR, la faire suivre de Ø indiquant le réceptacle où réside le disque système (drive Ø) et d'appuyer sur la touche RETURN (retour chariot), le répertoire apparait sur la console:

```
>L Ø
TEKDOS-COPY
FILE NAME          BLKS
TFKDOS             I6
.                  I74
COPYSYS            I
NEW DISC           I
MULTACCO           I
MULTACCA           4
MULTACC            2
```

```
FEUXO                I
FEUXA                3
FEUX                 2
FEUX*                2
MULTACC*             2
TOTAL FILES          I2
TOTAL BLOCKS USED    2Ø9
BLOCKS AVAILABLE     95
TOTAL BAD BLOCKS     Ø
```

Les trois exemples qui suivent illustreront le formatage et la vérification d'un disque vierge avant son utilisation et montreront également la duplication de fichiers d'un disque système sur un disque vierge formaté et vérifié.

b) Formatage d'un disque:

Le système étant mis en route, insérer un disque système dans le réceptacle Ø et un disque vierge dans le réceptacle I. Entrer la ligne de commande suivante:

```
>FORMAT I, NOUVEAU
```

Le "I" indique le réceptacle qui contient le disque à formater appelé " NOUVEAU ". Après trois minutes environ, la réponse suivante apparait sur la console:

```
*FMT*EOJ
```

Elle indique la fin de l'exécution de la commande FORMAT (EOJ = End Of Job).

c) Vérification du disque:

La vérification d'un disque formaté est nécessaire pour détecter l'existance possible de mauvais secteurs sur ce disque. Faire la commande VERIFY:

```
>VERIFY I
```

Le "I" indique le réceptacle contenant le disque à vérifier. Une fois la vérification terminée, le message suivant apparait indiquant la fin de l'exécution de la commande VERIFY:

```
*VERIFY *EOJ
```

d) Duplication du disque:

Cette procédure est assurée par la commande DUP de la manière suivante:

```
> DUP Ø I NOUVEAU
```

Le "Ø" indique le réceptacle contenant le disque dont on veut recopier les fichiers. Le "I" spécifie le réceptacle où réside le disque à dupliquer. Après quelques minutes, celà dépendra de la quantité d'informations à dupliquer, la fin de cette commande est indiquée par le message suivant:

```
* DUP *  EOJ
```

Ainsi, le contenu du disque système est totalement dupliqué sur le nouveau disque.

D'autres commandes de TEKDOS seront utilisées dans les chapitres qui suivent. En annexe I, on trouvera la liste de toutes les commandes de TEKDOS.

III. ECRITURE D'UN PROGRAMME SOURCE.

Un programme source est mis au point à l'aide d'instructions en mnémonique 6800, des directives d'assemblage et des commentaires. Chaque instruction 6800 doit-être écrite conformément à la mise en page de l'assembleur TEKTRONIX 6800.

I°) Instructions utilisées.

Trois sortes d'instructions peuvent-être utilisées:

a) Instructions 6800:

Ces instructions permettent d'effectuer des opérations arithmétiques, logiques , de décalage, de branchement, de chargement et de stockage. L'annexe 2 donne les 72 instructions du 6800.

b) Directives d'assemblage:

Huit variétés de directives d'assemblages peuvent-être utilisées dans le programme source. En annexe 3, on citera toutes les directives en donnant l'usage de chacune.

c) Macro-instructions:

Une macro-instruction est un groupe de lignes sources susceptible d'être utilisé plusieurs fois, en des endroits différents, au sein d'un même programme source. Ce groupe de lignes est écrit une seule fois, généralement au début d'un programme. Pour pouvoir appeler une macro-instruction, il suffit d'écrire en colonne opération le nom de la macro-instruction défini par la directive d'assemblage MACRO. Des modifications peuvent-être apportées au niveau de la colonne opérande à chaque appel, pour tout paramètre écrit entre apostrophes. Une macro-instruction est définie par la directive MACRO et se termine par la directive ENDM. Le texte de la macro-instruction demeure non assemblé tant qu'on ne lui a pas fait appel.

2°) Mise en page.

Chaque ligne du programme source peut contenir jusqu'à I28 caractères. Quatre colonnes peuvent-être utilisées au niveau de chaque ligne : une colonne étiquette, une colonne opération, une colonne opérande et une colonne commentaire. L'enfoncement de la touche RETURN (retour chariot) au niveau du clavier, termine l'écriture d'une ligne et permet de commencer l'écriture de la ligne suivante.

a) Colonne étiquette:

Lorsqu'elle est utilisée, la colonne étiquette commence au premier caractère de

la ligne et se termine par une tabulation ou un espace. Elle représente des adresses de branchement ou d'appel à des sous-programmes; pour les mnémoniques 6800, l'étiquette représente l'adresse du premier octet de l'instruction. Cette colonne peut contenir jusqu'à huit caractères dont le premier doit-être alphabétique, les autres peuvent-être soit des lettres majuscules de A à Z, soit des nombres de 0 à 9 ou des caractères tels que . (point), - (tiret), $ (dollar), etc... . Chaque étiquette doit-être unique au sein de chaque module source.

b) Colonne opération:

Cette colonne démarre dès la fin de l'espace ou de la tabulation terminant la colonne étiquette. Elle contient soit une mnémonique 6800, soit une directive d'assemblage ou une macro-instruction. Un espace ou une tabulation termine cette colonne.

c) Colonne opérande:

La colonne opérande spécifie les valeurs ou les adresses requises pour les différentes instructions 6800, les directives d'assemblages ou les appels de macro-instructions. Elle peut contenir soit un symbole d'un registre du MC6800 tels que A ou B pour indiquer l'accumulateur A ou B et X pour leregistre d'indexage, soit une expression représentant une donnée ou une adresse en mémoire.

L'expression peut-être:

- un nombre entier ou naturel. Les scalaires sont représentés par les entiers compris entre -32768 et +32767. Les adresses sont définies par les entiers naturels 0 à 65535 . Une constante numérique est un entier possédant toujours un suffixe sauf si le nombre est écrit en base IO. En base I6, le suffixe est désigné par H (ex.I5H); un Ø doit précéder tout nombre écrit en notation hexadécimale et commençant par une lettre alphabétique (ex.ØF4H). Les lettres O et Q sont utilisées pour désigner un nombre écrit en base 8 (ex.450 ou 45Q). En binaire, le suffixe est la lettre B (ex.IOIIOOIIB).

- un symbole qui représente généralement l'adresse d'une donnée ou la donnée elle-même. Si un opérande est désigné par un symbole, celui-ci doit obligatoirement figurer dans la colonne étiquette, à l'exception du symbole $ qui peut-être utilisé uniquement dans la colonne opérande et indique le premier octet de la ligne où il apparait. Des opérandes multiples dans une même colonne doivent-être séparés par des virgules ou un espace. Un point virgule termine la colonne opérande en indiquant le début de la colonne commentaire.

d) Colonne commentaire:

Cette colonne est facultative; toutefois sa présence rend les programmes plus lisibles, donc plus faciles à modifier. Un retour chariot termine la colonne commentaire.

IV. L'EDITEUR DE TEXTE.

Le système de développement possède un éditeur de texte résidant en mémoire programme et aidant à la création et à la modification de programmes écrits en code source. Ces fonctions sont réalisées par l'éditeur qui charge en mémoire programme des lignes de commandes entrées depuis le clavier et transfère les données résultantes sur disque. Chaque ligne de commande précise une ou plusieurs actions que l'éditeur doit entreprendre. La mémoire programme permet le stockage du texte sur lequel opère l'éditeur qui garde un repère de la ligne du texte sur laquelle il travaille, à l'aide d'un pointeur. L'appel de l'éditeur de texte est réalisé par la commande EDIT qui fait partie des commandes TEKDOS mais qui s'exécute dans la mémoire programme après avoir été chargée à partir du disque système.

I°) Les trois formes d'édition de texte.

On peut invoquer l'éditeur avec l'une de ces trois possibilités:

- EDIT ORIGINAL FINAL
- EDIT FINAL
- EDIT

a) EDIT ORIGINAL FINAL:

Cette forme nous permet de modifier le contenu d'un fichier nommé " ORIGINAL " sans pour celà détruire son texte. "FINAL" devient le nom du nouveau fichier contenant le texte corrigé. " ORIGINAL " doit désigner un fichier précédemment créé et stocké sur disque. S'il est nouveau, l'éditeur affiche le message ** PI = NEW FILE ?* * et retourne sous contrôle de TEKDOS. " FINAL " peut désigner un fichier existant ou nouveau. S'il désigne un fichier existant, son contenu sera détruit et remplacé par le contenu modifié de " ORIGINAL " dès qu'on sort de l'éditeur avec la commande FILE. S'il désigne un nouveau fichier, l'éditeur répond avec le message ** NEW FILE ** . La commande FILE permettra à l'éditeur d'écrire dans "FINAL" le texte modifié de " ORIGINAL " et retournera sous contrôle de TEKDOS.

b) EDIT FINAL:

" FINAL " peut désigner un nouveau fichier ou un fichier existant. Si le fichier est nouveau, l'éditeur répond avec le message ** NEW FILE ** . Après l'apparition de ce message sur la console, on fait entrer la commande INPUT et on écrit le texte. Pour garder ce texte et retourner sous contrôle de TEKDOS, on fait la commande FILE. Si on veut retourner sous contrôle de TEKDOS sans garder le texte, on fait la commande QUIT. Si " FINAL " est un fichier existant, le caractère * apparait sur la console. On met le texte dans la mémoire programme avec la commande GET, on le modifie et avec la commande FILE, on retourne sous contrôle de TEKDOS. Si on désire retourner sous contrôle de TEKDOS sans garder ce texte modifié, on fait entrer la commande QUIT au lieu et place de FILE.

c) EDIT:

Cette forme de commande est principalement utilisée pour charger un texte d'un fichier quelconque dans la mémoire programme et également pour stocker un texte contenu dans la mémoire programme dans un fichier quelconque. En faisant entrer cette commande, l'éditeur répond avec le message ** NO FILES SPECIFIED ** , on peut utiliser la commande GET pour transférer le contenu d'un fichier dans la mémoire programme, le modifier et faire entrer la commande PUT pour le remettre dans le fichier. Cette façon de procéder permet de modifier les sous-programmes et aussi de mettre plusieurs fichiers dans un seul plus grand. Un texte écrit dans un fichier existant détruit ce qui existait auparavant dans ce fichier.

2°) Délimiteur espace et caractères utilisés dans une ligne de commande.

Après l'appel de l'éditeur, le caractère * apparait sur la console indiquant que l'éditeur est prêt à accepter une ligne de commande qui comprend le nom de la commande et des caractères. Chaque ligne de commande, composée au maximum de I28 caractères, est suivie d'un retour chariot permettant son exécution. Un blanc, généré par le délimiteur espace, est généralement placé entre le nom d'une commande et son paramètre. Ce blanc ne peut pas être remplacé par une virgule. Le caractère " n " , qui est généralement une valeur numérique et qui revêt deux formes: un nombre absolu " a " ou une plage de lignes "b-c", peut-être accolé à la commande. Les nombres a, b et c doivent-être compris entre I et 32767. Le caractère " : " permet de grouper plusieurs commandes sur une même ligne. Les caractères " < " et " > " permettent l'exécution répétitive d'une commande. Les caractères B, E et C peuvent-être utilisés comme paramètres de commande. Ils représentent respectivement le début, la fin et la ligne en cours d'un texte.

On trouvera en annexe 4, toutes les commandes de l'éditeur avec leurs définitions respectives.

V. LE PROCESSEUR ASSEMBLEUR.

Pour que la machine puisse exécuter un programme source, on fait appel au processeur assembleur qui traduit ce programme en code objet binaire. Le processeur assembleur du TEKTRONIX 8002A est à deux passes. Pendant la premiére passe, l'assembleur traduit les mnémoniques en code machine et range les étiquettes dans la table des étiquettes. A la deuxième passe, il affecte aux opérandes leurs valeurs réelles (figure I2).

Le fichier d'assemblage est constitué de deux parties:

- le programme source avec en regard le code objet généré;
- la table des étiquettes qui contient les symboles utilisés avec en regard leur équivalent en code hexadécimal, éventuellement le nombre d'erreurs de syntaxe dans le

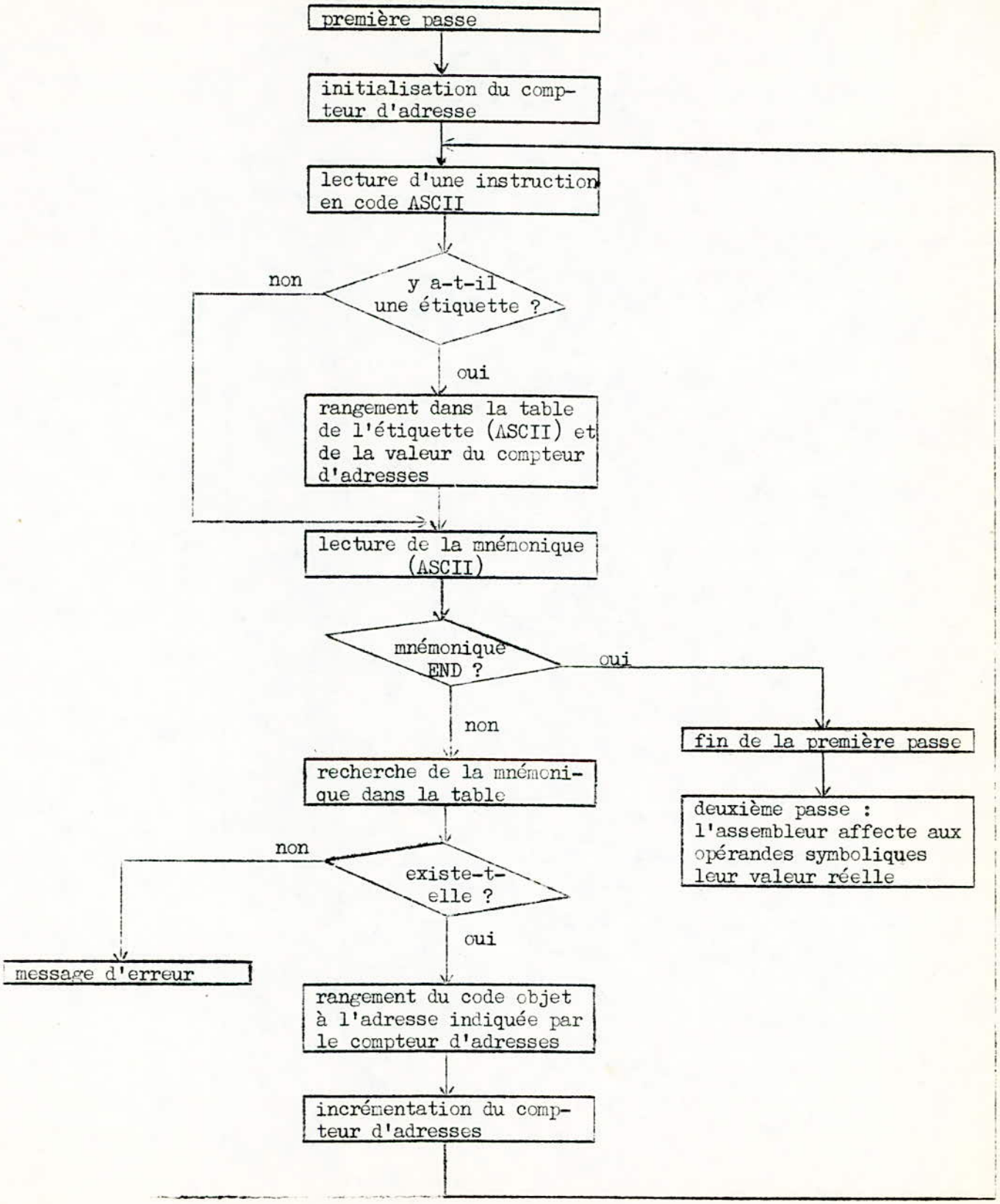


Fig. I2 Organigramme de l'assembleur à deux " passes " du 8002A.

programme source, ainsi qu'un message d'erreur venant juste après chaque ligne où l'erreur a été décelée.

La mise en route du processeur assembleur se fait sous contrôle de TEKDOS à l'aide de la commande ASM écrite suivant la syntaxe ci-dessous:

ASM nom du code objet nom du fichier d'assemblage nom du programme source

Tous les paramètres de la commande ASM sont séparés par un blanc ou une virgule. Les noms du code objet et du fichier d'assemblage peuvent-être omis, par contre, la présence du nom du programme source est obligatoire. Si on omet d'adresser le code objet ou le fichier d'assemblage ou les deux à la fois, on utilise les virgules comme suit:

ASM,, nom du fichier d'assemblage nom du programme source

ASM nom du code objet ,, nom du programme source

ASM,,, nom du programme source

Si le code objet et le fichier d'assemblage sont destinés à un disque ne contenant pas TEKDOS, il faudra spécifier ceci en mettant un slash / et le numéro du réceptacle contenant ce disque juste après les noms du code objet, du fichier d'assemblage et du programme source. Plusieurs programmes sources peuvent-être assemblés par une seule commande ASM; il suffit d'écrire leurs noms sur une même ligne de commande.

Le processeur assembleur travaille avec la mémoire utilisateur (32 Koctets), totalement indépendante de la mémoire système.

VI. DEVERMINAGE.

Le programme de DEBUG permet le contrôle du fonctionnement de la maquette prototype du point de vue logiciel et matériel. Ce programme est chargé en mémoire système à l'aide de la commande DEBUG de TEKDOS. Le logiciel de déverminage possède deux types de commandes qui s'exécutent sous contrôle de TEKDOS : celles d'affichage de l'information sur la console et celles de modification du contenu des registres du microprocesseur émulateur.

La commande TRACE utilise trois formes:

- TRACE ALL assure l'affichage des opérations ayant lieu durant l'exécution d'un programme.
- TRACE JUMP visualise les instructions de branchement, de saut inconditionnel et d'appels de sous-programmes.

L'adjonction du paramètre STEP à ces deux commandes permet l'affichage " pas à pas ".

- TRACE OFF met hors service l'affichage.

L'affichage est donné suivant la mise en page ci-dessous:

LOC INST MNEM R OPER X/PC EADD RA RB XREG SP CC

LOC : adresse de la dernière instruction exécutée.

INST : code hexadécimal de la dernière instruction exécutée.

MNEM : mnémonique de l'instruction.

R : accumulateurs A ou B quand ils sont utilisés comme opérandes.

OPER : opérande.

X/PC : contenu du registre d'index pour les instructions exécutées ou du compteur ordinal pour les instructions de branchement relatif.

EADD : calcul de l'adresse effective pour les instructions d'accès mémoire.

RA : contenu de l'accumulateur A.

RB : contenu de l'accumulateur B.

XREG : contenu du registre d'index.

SP : contenu du pointeur de pile.

CC : contenu du registre code condition.

Tout le contenu de ces colonnes est donné en code hexadécimal sauf la colonne mnémonique.

La commande DSTAT affiche le contenu de tous les registres du microprocesseur émulateur, la dernière instruction exécutée et son adresse ainsi que les points d'arrêt et leurs adresses.

Les points d'arrêt provoquent l'interruption du déroulement d'un programme. Ils sont générés par la commande BKPT (break point) qui s'exécute sous contrôle de TFKDOS . La spécification de l'adresse où l'on veut suspendre le programme est nécessaire.

La commande CLPB annule les points d'arrêt provoqués par BKPT.

La commande SET charge les registres du MC6800 par des valeurs hexadécimales spécifiées par le programmeur.

VII. L'EDITEUR DE LIEN.

Les programmes sources assemblés séparément sont rassemblés en un seul code objet par l'éditeur de lien. Celui-ci réserve un espace mémoire pour chaque section du programme à charger, établit une table des symboles de type GLOBAL, éventuellement translate des adresses dans des zones mémoire réservées et génère un bilan indiquant les différentes sections et les valeurs assignées aux variables de type GLOBAL.

Une section est un code objet issu d'un assemblage à partir d'une même origine . Un module objet peut contenir un maximum de 255 sections ou symboles de type GLOBAL. Une section est définie par SECTION, COMMON ou RESERVE. PAGE, INPAGE et BYTE placés en colonne opérande spécifient le type de translation autorisée. Située dans la colonne opérande, ABSOLUTE interdit toute translation.

Appel de l'éditeur de lien:

La commande de TEKDOS " LINK " permet l'appel de l'éditeur de lien. Trois méthodes sont possibles pour cet appel:

- appel simple : il s'effectue suivant la syntaxe ci-dessous :

LINK fichier de chargement fichier de bilan objet I objet 2 ...

Objet I et objet 2 sont deux codes objets issus de deux assemblages différents, ils doivent-être rassemblés en un seul code objet qui sera rangé dans le fichier de chargement. Dans le fichier de bilan, on y trouve les noms de sections, les symboles GLOBAL et les messages d'erreur. Les noms des fichiers de chargement et de bilan sont facultatifs, ceux des codes objets sont obligatoires.

- appel intéractif : il suffit de faire entrer la commande LINK suivie d'un retour chariot. L'éditeur de lien répond par le caractère * et attend les commandes, à la réception de la commande END, il entreprend l'exécution.

- appel par fichier de commande : la syntaxe suivante définit ce type d'appel :

LINK C fichier de commande

Cet appel permet l'exécution des commandes écrites au préalable dans le fichier spécifié. Celui-ci se termine par la commande END.

Les commandes utilisées en appel intéractif et par fichier de commande sont :

LOG : affiche les messages sur la console et le fichier de bilan s'il est spécifié. Les commandes postérieures à LOG figureront sur le bilan.

NOLOG : supprime l'affichage.

MAP : donne une carte de zones mémoire située dans le fichier de bilan et regroupant les noms, les sections et leurs paramètres, les symboles GLOBAL assignés ou non.

NOMAP : supprime la carte de zones mémoire.

LIST : elle transfère le bilan sur un fichier sur disque ou un périphérique spécifiés dans la ligne de commande.

LOAD : le code objet final est dirigé sur un fichier sur disque spécifié dans la ligne de commande, ensuite il peut-être chargé en mémoire pour exécution à l'aide de la commande LOAD.

DEFINE : il nous permet d'affecter manuellement une valeur à un symbole désigné lors d'un assemblage comme GLOBAL. Dans la ligne de commande, DEFINE est suivie de SYMBOLE = VALEUR.

LOCATE : situe manuellement dans l'espace mémoire une section définie comme telle lors d'un assemblage.

C : suivie du nom du fichier de commande, elle indique le fichier sur disque dans lequel l'éditeur de lien ira chercher les commandes pour l'appel sur fichier de commande.

TRANSFER : suivie d'un symbole ou d'une valeur, elle spécifie l'adresse de lancement d'un programme qui s'affichera sur la console lors de la commande LOAD. Le symbole est de type GLOBAL, la valeur doit-être écrite en code hexadécimal. La commande TRANSFER remplace et annule la directive END.

END : elle interrompt le mode d'entrée de commandes pour l'éditeur de lien.

VIII. SERVICE CALL (SVC).

Pendant la phase de développement du logiciel, en mode d'émulation zéro, on peut utiliser un " service call " (SVC); celui-ci permet au processeur émulateur d'avoir accès aux périphériques du 8002A. On peut par exemple faire entrer des données au sein d'un programme à partir du clavier ou d'un fichier, transférer des données du programme vers un fichier ou la console de visualisation. Il existe au total 26 fonctions qui sont réalisées à l'aide du SVC. Pour activer un SVC, il suffit de mettre sur le bus adresse une des adresses (FEF7, FEF6, FEF5, FEF4, FEF3, FEF2) spécifiques aux six SVC disponibles, avec l'instruction STA A suivie de l'instruction NOP durant laquelle le SVC est effectué.

Lorsqu'un SVC est appelé, TEKDOS ira chercher dans le pointeur du SRB l'adresse de la première instruction du SRB (à chaque adresse FEF7, FEF6, FEF5, FEF4, FEF3, FEF2 correspond une adresse donnée du pointeur du SRB). Le SRB est un bloc de 8 bytes contenant les informations dont a besoin TEKDOS pour effectuer le SVC; les octets I, 2, 6, 7 et 8 sont à écrire dans le programme, ils indiquent respectivement le code de la fonction à réaliser (c'est à dire le type de travail à faire: entrée/sortie, ouverture d'un fichier, lecture ASCII, écriture ASCII, etc...), le canal affecté au périphérique , la longueur en octets du tampon où transite l'information destinée ou provenant du processeur émulateur, l'adresse de départ du tampon (septième et huitième octets). Les octets 3, 4 et 5 sont utilisés de façon interne par TEKDOS, il y indique des informations destinées au processeur émulateur.

Un exemple d'utilisation du SVC sera traité au chapitre suivant.

IX. ANALYSEUR TEMPS REEL (RTPA).

L'analyseur temps réel (RTPA : Real Time Prototype Analyseur) est utilisé pour suivre le déroulement d'un programme en temps réel. Son emploi est nécessaire surtout pendant les phases de déverminage et d'intégration logiciel/matériel de la maquette à réaliser. Constitué d'une carte analyseur contenant deux comparateurs, d'un interface d'acquisition de données et d'une sonde à huit canaux, le RTPA a deux fonctions :

- fonction d'enregistrement : pendant l'exécution d'un programme, l'analyseur peut contenir jusqu'à I28 transactions qui ont précédé une condition pré-programmée, celle-ci peut porter sur une donnée, une adresse, un ou plusieurs bits externes, un certain type d'opérations ou une combinaison de tous ces facteurs. Une transaction comprend au maximum l'enregistrement de 40 bits (I6 bits d'adresse, 8 bits de données, 8 bits externes prélevés au moyen de la sonde, 3 bits pour identifier le type de cycle exécuté: lecture,

écriture, recherche et 5 bits utilisés de façon interne par le RTPA pour l'identification de l'information entrante).

- fonction de comptage : le RTPA peut déterminer le temps (en microsecondes ou en millisecondes) séparant deux événements pré-programmés, tout en conservant la fonction d'enregistrement.

Les commandes du RTPA permettent:

- la pose de comparaison d'événement qui s'effectue avec les commandes EVTI et EVT2 ; l'événement sera spécifié à l'aide des paramètres écrits dans la ligne de commande EVTI ou EVT2;
- l'annulation de celle-ci effectuée par la commande EVT CLR;
- la pose et l'annulation d'un point d'arrêt respectivement à l'aide des commandes BIF et BIF CLR;
- le choix du type de transactions à mémoriser: RTT suivie du type de paramètre à stocker;
- l'affichage sur la console du contenu de la mémoire du RTPA: DRT;
- le lancement du compteur et l'affichage de son contenu: CNT.

Trois modes d'enregistrement sont possibles :

- dans le premier mode, le programme s'arrête quand les conditions mentionnées dans la ligne de commande EVT sont satisfaites. En ce moment, la mémoire du RTPA contient les transactions ayant précédé le point d'arrêt;
- le second mode permet, en utilisant l'option C = 64 dans la ligne de commande de EVT, de stocker ce qui précède et ce qui suit une comparaison spécifiée à l'aide des autres paramètres de EVT;
- le troisième mode permet le stockage de ce qui suit une condition programmée en utilisant l'option C = I28 dans EVT.

Le RTPA peut-être utilisé avec les trois modes d'émulation (Ø, I et 2). Dans le chapitre V, on illustrera l'utilisation du RTPA à l'aide du programme "FEUX" et de la maquette réalisée.

CHAPITRE III

-EXEMPLES D'UTILISATION DU TEKTRONIX 8002A EN MODE D'EMULATION Ø-

I. Introduction

II. Multiplication accélérée

III. Exemple d'utilisation du SVC

I. INTRODUCTION.

L'organigramme ci-dessous schématise les différentes phases nécessaires à la mise au point et au chargement d'un programme en mémoire :

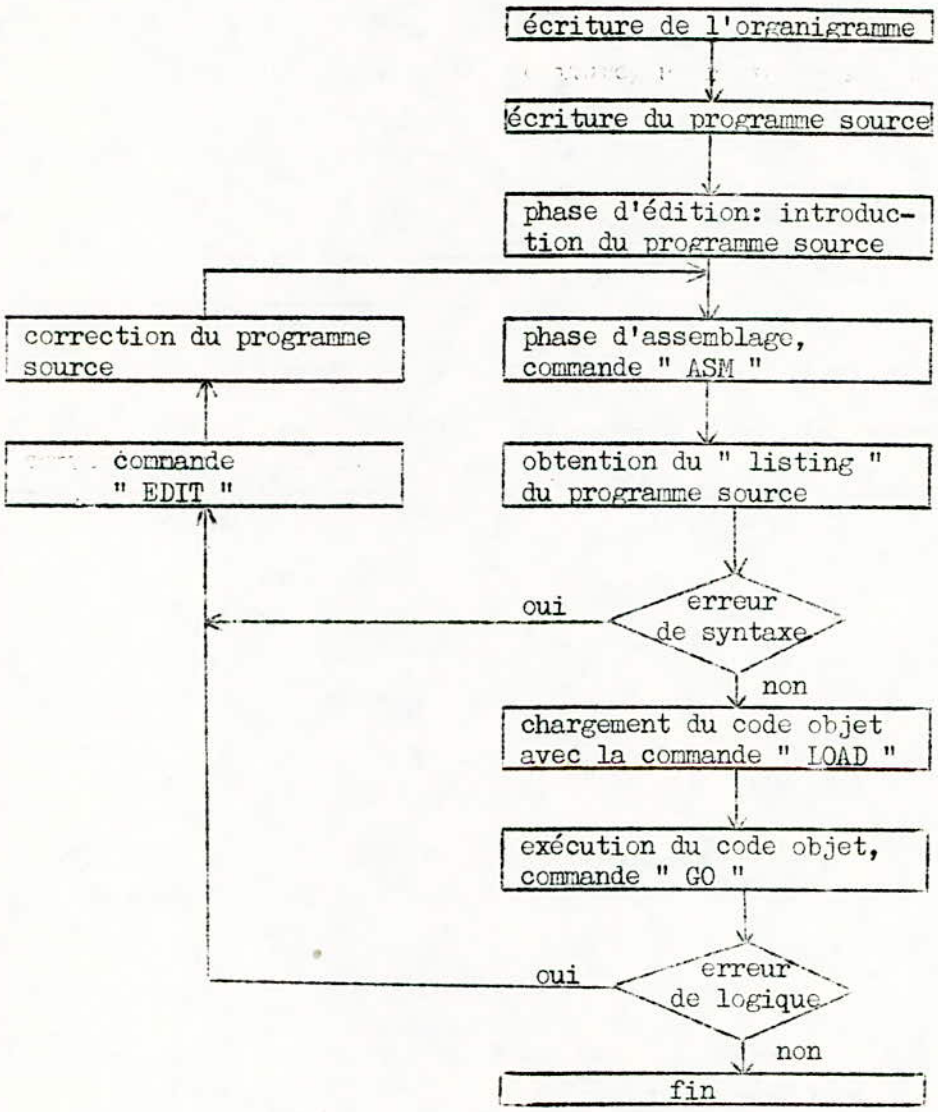


Fig. I3 Organigramme retraçant les phases d'écriture, de mise au point et de chargement d'un programme en mémoire.

Après avoir étudié dans le second chapitre les moyens dont on dispose pour écrire, mettre au point et charger en mémoire un programme pour microordinateur (TEKTRONIX 8002A), ce chapitre illustrera cette étude par deux programmes exemples: la multiplication accélérée et un exemple utilisant le SVC. L'exécution de ces programmes se fera en mode d'émulation Ø qui utilise l'horloge et la mémoire du 8002A.

II. MULTIPLICATION ACCELEREE.

I°) L'organigramme.

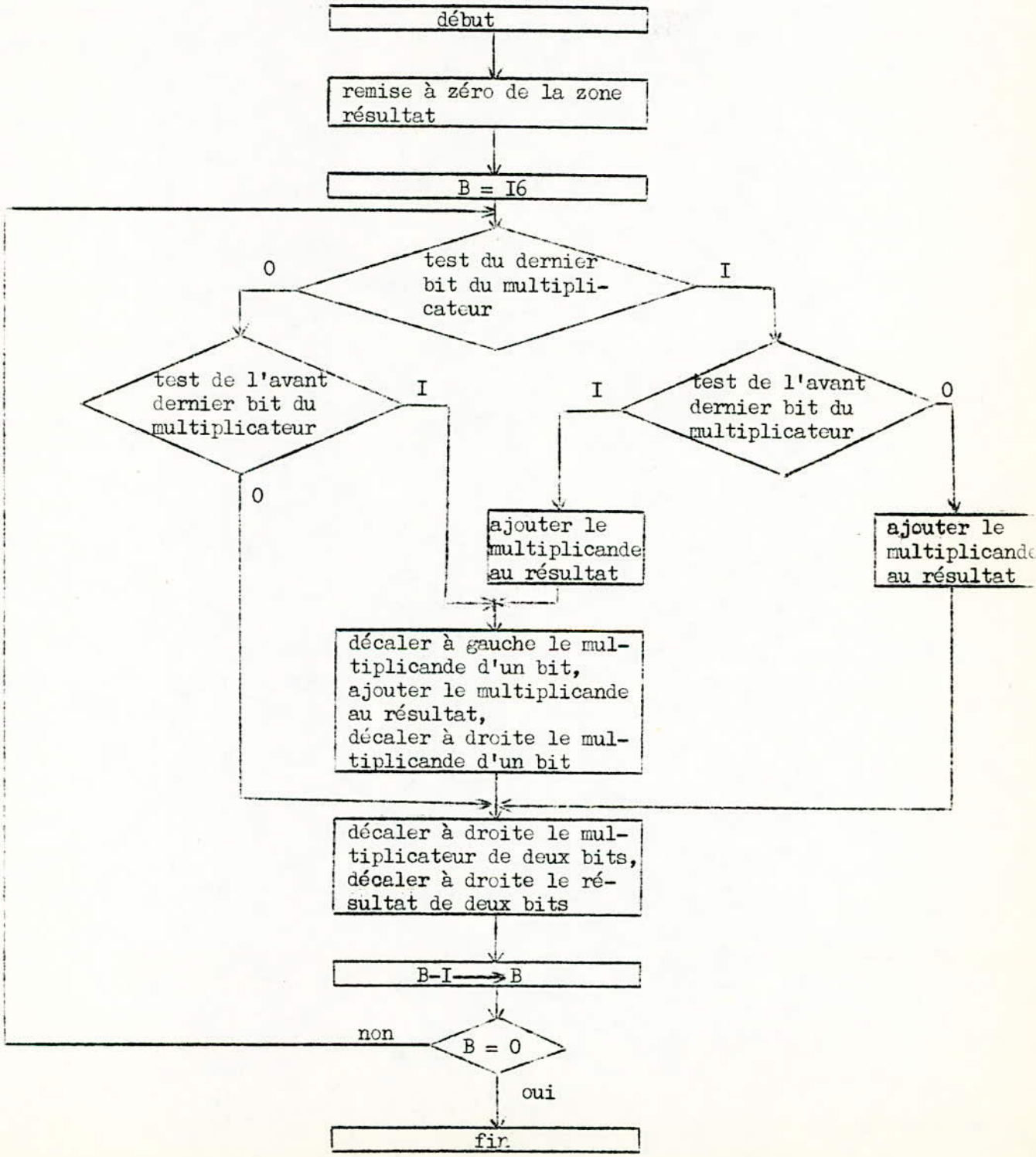


Fig. I4 Organigramme de la multiplication accélérée.

2°) Ecriture du programme source.

Ce programme est conçu pour la multiplication de deux nombres entiers positifs s'écrivant sur 32 bits; la méthode consiste à tester à la fois deux bits du multiplicateur.

Le multiplicande occupe les positions mémoire I4H, I5H, I6H, I7H. La case mémoire I3H étant réservée pour les opérations de décalage.

Le multiplicateur occupe les positions mémoire ØAH, ØBH, ØCH, ØDH.

Le résultat s'écrit sur 64 bits et occupe la zone mémoire IEH, IFH, 2ØH, 2IH, 22H, 23H, 24H, 25H. La case mémoire IDH sert au stockage des retenues éventuelles qui apparaissent lors d'opérations d'addition.

Le programme est le suivant:

étiquette	mnémonique	opérande		commentaires
	ORG	IØØH	;	
	LDX	#Ø9H	;	
	CLR	A	;	
ALPHA	STA	A ICH,X	;	remise à zéro de la zone
	DEX		;	résultat
	BNE	ALPHA	;	
	CLR	I3H	;	
	LDA	B #IØH	;	chargement de l'accumulateur B avec la valeur IØH (I6 en décimal)
BOUCLE	LDA	A ØDH	;	
	LSR	A	;	test du dernier et avant dernier
	BCS	GAMMA	;	bit du multiplicateur
	LSR	A	;	
	BCC	ZETA	;	
	CLC		;	
SUITE	LDX	#I7H	;	
TOURNE	ROL	ØH,X	;	décalage du multiplicande d'un
	DEX		;	bit vers la gauche
	CPX	#I2H	;	
	BNE	TOURNE	;	
	LDX	#Ø5H	;	
RETOUR	LDA	A ICH,X	;	addition du multiplicande
	ADC	A I2H,X	;	et du résultat

```
                STA     A ICH,X         ;
                DEX                     ;
                BNE     RETOUR          ;

                CLC                     ;
                LDX     #I3H            ;
RODE            ROR     ØH,X            ;  décalage du multiplicande
                INX                     ;  d'un bit vers la droite
                CPX     #I8H            ;
                BNE     RODE            ;
                BRA     ZETA            ;

GAMMA           LSR     A               ;  test de l'avant dernier bit
                BCS     TETA            ;  du multiplicateur

                CLC                     ;
                LDX     #Ø5H            ;
REPART          LDA     A ICH,X         ;
                ADC     A I2H,X         ;  addition du multiplicande
                STA     A ICH,X         ;  et du résultat
                DEX                     ;
                BNE     REPART          ;
                BRA     ZETA            ;

TETA            CLC                     ;
                LDX     #Ø5H            ;
VIRE            LDA     A ICH,X         ;
                ADC     A I2H,X         ;  addition du multiplicande
                STA     A ICH,X         ;  et du résultat
                DEX                     ;
                BNE     VIRE            ;
                BRA     SUITE           ;

ZETA            LDA     A #Ø2H          ;
REFAIT          CLC                     ;
                LDX     #ØAH            ;
CERCLE          ROR     ØH,X            ;  décalage du multiplicateur
                INX                     ;  de deux bits vers la droite
```

```
                CPX             #ØEH        ;
                BNE             CERCLE      ;
                DEC             A           ;
                BNE             REFAIT      ;

                LDA             A #Ø2H      ;
REMET           CLC                         ;
                LDX             #IDH        ;
RONDE           ROR             ØH,X        ;  décalage du résultat de deux
                INX                         ;  bits vers la droite
                CPX             #26H        ;
                BNE             RONDE       ;
                DEC             A           ;
                BNE             REMET       ;

                DEC             B           ;  décrémentation du contenu de B
                BNE             BOUCLE      ;  branchement à BOUCLE si B ≠ Ø
                END                         ;  fin
```

3°) Phase d'édition.

Pour introduire le programme source dans la mémoire programme, il suffit:

- d'appeler l'éditeur de textes avec la commande " EDIT " de TEKDOS suivie du nom du fichier (MULTACC dans notre cas);
- de tabuler en faisant entrer la commande"XTABS ON" de l'éditeur;
- d'invoquer la commande " INPUT " de l'éditeur pour écrire le programme source.

Etant sous contrôle de TEKDOS, on fait la manipulation suivante:

```
>EDIT MULTACC
**EDIT VERSION 3.ØA
**NEW FILE
 *XTABS ON
 *I
  INPUT :
          ORG         IØØH
          LDX         #Ø9H
           .            .
           .            .
           .            .
          BNE         BOUCLE
          END
```

Une fois l'écriture du texte achevée, on appuie deux fois sur la touche RETURN (retour chariot) pour pouvoir utiliser d'autres commandes de l'éditeur. On fait entrer la commande " FILE " pour garder le texte dans un fichier.

```
* FILE
 * * END OF TEXT
* EDIT * EOJ
```

On revient ainsi sous contrôle de TEKDOS.

4°) Phase d'assemblage.

La commande " ASM " de TEKDOS, permettra l'assemblage du programme source et l'obtention du programme objet et du listing.

```
> ASM MULTACCO MULTACCA MULTACC
   Tektronix          M68ØØ ASM V3.3
**** Pass 2
     7I Source lines      7I Assembled lines      8Ø2 Bytes available
                    >>> No assembly errors detected <<<
* ASM * EOJ
```

Pour afficher le listing d'assemblage, on utilise la commande de TEKDOS " COPY ".

```
> COP MULTACCA
   Tektronix          M68ØØ ASM V3.3
ØØØØI                ØIØØ            ORG    IØØH       ;                          Page I
ØØØØ2  ØIØØ  CEØØØ9              LDX    #ØØ9H      ;
ØØØØ3  ØIØ3  4F                  CLR    A          ;
ØØØØ4  ØIØ4  A7IC      ALPHA     STA    A ICH,X    ;  remise à zéro de la zone
ØØØØ5  ØIØ6  Ø9                  DEX               ;  résultat
ØØØØ6  ØIØ7  26FB                BNE    ALPHA      ;
ØØØØ7  ØIØ9  7FØØI3              CLR    I3H        ;
ØØØØ8  ØIØC  C6IØ                LDA    B #IØH     ;  Chargement de l'accumulateur B
ØØØØ9  ØIØE  96ØD      BOUCLE    LDA    A ØDH      ;
ØØØIØ  ØIIØ  44                  LSR    A          ;  test du dernier et avant dernier
ØØØII  ØIII  2529                BCS    GAMMA      ;  bit du multiplicateur
ØØØI2  ØII3  44                  LSR    A          ;
ØØØI3  ØII4  2447                BCC    ZETA       ;
ØØØI4  ØII6  ØC                  CLC               ;
ØØØI5  ØII7  CEØØI7    SUITE     LDX    #I7H       ;
ØØØI6  ØIIA  69ØØ      TOURNE    ROL    Øh,X       ;  décalage du multiplicande d'un
ØØØI7  ØIIC  Ø9                  DEX               ;  bit vers la gauche
```

```
ØØØI8  ØIID  8CØØI2          CPX   #I2H     ;
ØØØI9  ØI2Ø  26F8            BNE   TOURNE   ;
ØØØ2Ø  ØI22  CEØØØ5          LDX   #Ø5H     ;
ØØØ2I  ØI25  A6IC  RETOUR    LDA   A ICH,X  ; Addition du multiplicande
ØØØ22  ØI27  A9I2            ADC   A I2H,X  ; et du résultat
ØØØ23  ØI29  A7IC            STA   A ICH,X  ;
ØØØ24  ØI2B  Ø9              DEX            ;
ØØØ25  ØI2C  26F7            BNE   RETOUR   ;
ØØØ26  ØI2E  ØC              CLC            ;
ØØØ27  ØI2F  CEØØI3          LDX   #I3H     ;
ØØØ28  ØI32  66ØØ  RODE      ROR   ØH,X     ; décalage du multiplicande
ØØØ29  ØI34  Ø8              INX            ; d'un bit vers la droite
ØØØ3Ø  ØI35  8CØØI8          CPX   #I8H     ;
ØØØ3I  ØI38  26F8            BNE   RODE     ;
ØØØ32  ØI3A  202I            BRA   ZETA     ;
ØØØ33  ØI3C  44    GAMMA     LSR   A        ; test de l'avant dernier bit
ØØØ34  ØI3D  25ØF            BCS   TETA     ; du multiplicateur
ØØØ35  ØI3F  ØC              CLC            ;
ØØØ36  ØI4Ø  CEØØØ5          LDX   #Ø5H     ;
ØØØ37  ØI43  A6IC  REPART    LDA   A ICH,X  ;
ØØØ38  ØI45  A9I2            ADC   A I2H,X  ; addition du multiplicande
ØØØ39  0I47  A7IC            STA   A ICH,X  ; et du résultat
ØØØ4Ø  ØI49  Ø9              DEX            ;
ØØØ4I  ØI4A  26F7            BNE   REPART   ;
ØØØ42  ØI4C  2ØØF            BRA   ZETA     ;
ØØØ43  ØI4E  ØC    TETA      CLC            ;
ØØØ44  ØI4F  CEØØØ5          LDX   #Ø5H     ;
ØØØ45  ØI52  A6IC  VIRE      LDA   A ICH,X  ;
ØØØ46  ØI54  A9I2            ADC   A I2H,X  ; addition du multiplicande
ØØØ47  ØI56  A7IC            STA   A ICH,X  ; et du résultat
ØØØ48  ØI58  Ø9              DEX            ;
ØØØ49  ØI59  26F7            BNE   VIRE     ;
ØØØ5Ø  ØI5B  2ØBA            BRA   SUITE    ;
ØØØ5I  ØI5D  86Ø2  ZETA      LDA   A #Ø2H   ;
ØØØ52  ØI5F  ØC    REFAIT    CLC            ;
ØØØ53  ØI6Ø  CEØØØA          LDX   #ØAH     ;
```



```
ØØØ54  ØI63  66ØØ  CERCLE    ROR   ØH,X     ; décalage du multiplicateur
```

```
ØØØ55  ØI65  Ø8              INX              ;  de deux bits vers la droite
ØØØ56  ØI66  8CØØØE          CPX    ØEH       ;
ØØØ57  ØI69  26F8            BNE    CERCLE    ;
ØØØ58  ØI6B  4A              DEC    A         ;
ØØØ59  ØI6C  26FI            BNE    REFAIT    ;
ØØØ6Ø  ØI6E  86Ø2            LDA    A #Ø2H    ;
ØØØ6I  ØI7Ø  ØC      REMET   CLC              ;
ØØØ62  ØI7I  CEØØID          LDX    #IDH      ;
ØØØ63  ØI74  66ØØ    RONDE   ROR    ØH,X      ;  décalage du résultat de deux
ØØØ64  ØI76  Ø8              INX              ;  bits vers la droite
ØØØ65  ØI77  8CØØ26          CPX    #26H      ;
ØØØ66  ØI7A  26F8            BNF    RONDE     ;
ØØØ67  ØI7C  4A              DEC    A         ;
ØØØ68  ØI7D  26FI            BNE    REMET     ;
ØØØ69  ØI7F  5A              DEC    B         ;  décrémentation du contenu de B
ØØØ7Ø  ØI8Ø  268C            BNE    BOUCLE    ;  branchement à BOUCLE si B≠Ø
ØØØ7I                        END              ;  fin
   Tektronix          M68ØØ ASM V3.3          Symbol table        Page 3
   % MULTACC (default) Section (ØI82)
      ALPHA- - ØIØ4    BOUCLE - ØIØE    CERCLE - ØI63    GAMMA - - ØI3C
REFAIT - ØI5F    REMET - - ØI7Ø    REPART - ØI43    RETOUR - ØI25    RODE - - - ØI32
RONDE - - ØI74    SUITE - - ØII7    TETA - - - ØI4E    TOURNE - ØIIA
VIRE - - - ØI52    ZETA - - - ØI5D
      7I Source lines          7I Assembled lines          8Ø2 Bytes available
                    >>> No assembly errors detected <<<
 * COPY * EOJ
```

5°) Phase de chargement.

Le chargement en mémoire programme du code objet se fera par la commande de TEK-DOS " LOAD "; l'exécution du programme se fait en mode d'émulation Ø.

```
> EM Ø
* EMULATE * EOJ
> LO MULTACCO
  TRANSFER ADDRESS : ØØØØ
* LOAD * EOJ
```

La commande " EXAM " nous permettra l'introduction du multiplicateur et du multi-

plicande, écrits en notation hexadécimale, dans la mémoire de données.

```
>E ØA
  ØØØA = ØØ-FF  ØØ-FF  ØØ-FF  ØØ-FF
>E I4
  ØØI4 = ØØ-FF  ØØ-FF  ØØ-FF  ØØ-FF
```

Cet exemple est arbitraire; pour contrôler le bon fonctionnement du programme, on a jugé nécessaire d'utiliser tous les bits du multiplicateur et du multiplicande en prenant la valeur FFFFFFFF ([redacted]) pour chacun d'eux.

6°) Phase d'exécution.

Le contrôle de l'exécution du programme objet se fait sous la commande de TEKDOS " DEBUG ". La commande " GO " assure l'exécution du code objet qui se fera en temps réel dans le cas où la commande " TRACE OFF " est utilisée.

```
> DEB
>TR OFF
>G IØØ
  LOC    INST   MNEM   R   OPER   X/PC   EADD   RA   RB   XREG   SP    CC
 ØI82    5E     * * *                          ØØ   ØØ   ØØ26   ØØØØ  D4
 ØI82    BREAK
```

Pour afficher le résultat, on utilise la commande EXAM suivie de la case mémoire IE contenant l'octet le plus significatif.

```
> E IE
  ØØIE = FF  FF  FF  FE  ØØ  ØØ  ØØ  ØI
```

7°) Quelques commandes permettant des corrections au niveau du programme source.

Supposons qu'on ait fait des erreurs de syntaxe et de logique dans le programme source:

a- oubli de ALPHA (sixième ligne)
b- omission de Ø devant l'opérande DH (neuvième ligne)
c- oubli d'une ligne (trente deuxième ligne)
d- présence d'une ligne inutile (quarantième ligne)

a et b étant des erreurs de syntaxe, elles sont indiquées dans le listing d'assemblage.

```
> ASM MULTACCO MULTACCA MULTACC
   Tektronix          M68ØØ ASM V3.3
* * * *  Pass 2
   ØØØØ6      ØI Ø7    ØØØØØØ                BNE
* * * * ERROR 248
   ØØØØ9      ØI ØF    B6ØØØØ      BOUCLE    LDA     A DH
```

```
* * * * ERROR Ø74
    7I Source lines          7I Assembled lines          8Ø2 Bytes available
       2 ERRORS               I UNDEFINED SYMBOL
```

Pour corriger ces erreurs de syntaxe et de logique, on appelle l'éditeur de texte et on utilise les commandes " GET " pour transférer le texte du fichier vers la mémoire programme, "XTABS ON " pour la tabulation, " NUMBER ON " pour numéroter les lignes du texte et " TYPE B-E " pour visualiser tout le texte.

```
  > EDIT MULTACC
* * EDIT VERSION 3.ØA
  * GET IØØ
* * EOF
  * XTABS ON
  * NU ON
  * T B-E
  I              ORG      IØØH
  2              LDX      ##Ø9H
  .               .         .
  .               .         .
  .               .         .
  7Ø             BNE      BOUCLE
  7I             END
```

Le pointeur du " workspace " étant positionné sur la ligne 7I; la correction d'une ligne du texte nécessite le positionnement de ce pointeur sur la ligne à corriger. Les commandes UP et DOWN déplacent le pointeur respectivement vers le haut et vers le bas.

Les corrections sont assurées par les commandes de l'éditeur REPLACE, SUBSTITUTE, INSERT, KILL, etc...

```
 * U 65
      6:                BNE
 * R              BNE          ALPHA
      6:                BNE          ALPHA
 * D 3
      9:     BOUCLE     LDA          A DH
 * S/DH/ØDH/
      9:     BOUCLE     LDA          A ØDH
 * D 24
      33:    GAMMA      LSR          A
 * I              BRA          ZETA
 * D 7
```

```
     4Ø:                    NOP
* K
```

On transfère le texte corrigé sur le fichier avec la commande " FILE " et on assemble de nouveau.

```
* FILE
 * * END OF TEXT
 * * EOF
 * EDIT * EOJ
  > ASM MULTACCO MULTACCA MULTACC
    Tektronix          M68ØØ ASM V3.3
 * * * * Pass 2
        7I Source lines          7I Assembled lines          8Ø2 Bytes available
                  >>> No assembly errors detected <<<
  * ASM * EOJ
```

8°) Exemple d'utilisation des macro-instructions.

Dans l'exemple suivant, on utilise la directive " MACRO " regroupant les instructions ADC A I2H,X , STA A ICH,X et DEX du programme de la multiplication accélérée.

```
            ORG       IØØH          ;
            MACRO     MULTØI        ;  définition de la macro-instruction
            ADC       A I2H,X       ;
            STA       A ICH,X       ;
            DEX                     ;
            ENDM                    ;  termine le bloc défini par MACRO
            LDX       #Ø9H          ;
            CLR       A             ;
ALPHA       STA       A ICH,X       ;  remise à zéro de la zone résultat
            DEX                     ;
            BNE       ALPHA         ;
            CLR       I3H           ;
            LDA       B #IØH        ;  chargement de l'accumulateur B
BOUCLE      LDA       A ØDH         ;
            LSR       A             ;  test du dernier et avant dernier bit du mul-
            BCS       GAMMA         ;  tiplicateur
            LSR       A             ;
            BCC       ZETA          ;
            CLC                     ;
```

```
SUITE    LDX     #I7H          ;
TOURNE   ROL     ØH,X          ;  décalage du multiplicande d'un bit vers la
         DEX                   ;  gauche
         CPX     #I2H          ;
         BNE     TOURNE        ;
         LDX     #Ø5H          ;
RETOUR   LDA     A ICH,X       ;  addition du multiplicande
         MULTØI                ;  appel de la macro-instruction
         BNE     RETOUR        ;
         CLC                   ;
         LDX     #I3H          ;
RODE     ROR     ØH,X          ;  décalage du multiplicande
         INX                   ;  d'un bit vers la droite
         CPX     #I8H          ;
         BNE     RODE          ;
         BRA     ZETA          ;
GAMMA    LSR     A             ;  test de l'avant dernier bit
         BCS     TETA          ;  du multiplicateur
         CLC                   ;
         LDX     #Ø5H          ;
REPART   LDA     A ICH,X       ;
         MULTØI                ;  appel de la macro-instruction
         BNE     REPART        ;
         BRA     ZETA          ;
TETA     CLC                   ;
         LDX     #Ø5H          ;
VIRE     LDA     A ICH,X       ;
         MULTØI                ; appel de la macro-instruction
         BNE     VIRE          ;
         BRA     SUITE         ;
ZETA     LDA     A #Ø2H        ;
REFAIT   CLC                   ;
         LDX     #ØAH          ;
CERCLE   ROR     ØH,X          ; décalage du multiplicateur
         INX                   ;  de deux bits vers la droite
         CPX     #ØEH          ;
         BNE     CERCLE        ;
         DEC     A             ;
```

```
        BNE     REFAIT      ;
        LDA     A #Ø2H      ;
REMET   CLC                 ;
        LDX     #IDH        ;
RONDE   ROR     ØH,X        ;  décalage du résultat de deux bits
        INX                 ;  vers la droite
        CPX     #26H        ;
        BNE     RONDE       ;
        DEC     A           ;
        BNE     REMET       ;
        DEC     B           ;  décrémentation du contenu de B
        BNE     BOUCLE      ;  branchement à BOUCLE si B ≠ Ø
        END                 ;  fin
```

III. EXEMPLE D'UTILISATION DU SVC.

Dans l'exemple ci-dessous, on utilisera le SVCI avec la fonction 8I permettant de faire entrer des données dans un programme à partir du clavier. Le programme consiste à faire l'addition de deux nombres codés en ASCII, le résultat se trouvant dans l'accumulateur A.

```
        TITLE       LECTURE DE DONNEES ASCII
        ORG         ØH          ;
        STA         A           ;  utilisation du SVCI
        NOP                     ;
        CLR         A           ;  début du programme d'addition
        CLR         B           ;
        LDA         A 2ØØH      ;
        LDA         B 2ØIH      ;
        ABA                     ;
        CLR         ØFEF6H      ;  fin du programme d'addition
        NOP                     ;
        ORG         4ØH         ;  adresse du pointeur du SRB Utilisé avec le SVCI
        BYTE        HI (ESSAI), LO (ESSAI) ;
        ORG         IØØH        ;  adresse de départ du SRB
ESSAI   BYTE        8IH         ;  fonction " read ASCII and proceed "
        BYTE        ØØH         ;  sur le canal Ø
```

```
        BYTE    ØØH         ; initialisation des octets réservés (3, 4, 5)
        BYTE    ØØH         ;
        BYTE    ØØH         ;
        BYTE    ADI + I     ; longueur du tampon
        BYTE    HI (TEST), LO (TEST) ;
        ORG     2ØØH        ; adresse de départ du buffer
ADI     EQU     3Ø          ;
TEST    BLOCK   AD + I      ; définition du buffer pour le SVC
```

Pour lancer ce programme, on doit faire les commandes suivantes:

```
>EM Ø
>LO ESSO  (ESSO étant le nom donné au code objet)
>ASSIGN Ø CONI  (assignation du canal Ø au clavier)
>DEB
>TR ALL
>G Ø
```

A l'apparition du symbole " - ", on écrira les deux nombres suivis d'un retour chariot.

CHAPITRE IV

- L E P I A M C 6 8 2 I -

I. Définition du PIA

II. Description du MC682I

III. Exemples de programmation

I. DEFINITION DU PIA.

Le PIA (Peripheral Interface Adapter) est un organe qui assure l'échange d'informations entre le microprocesseur et l'extérieur. L'extérieur englobe les périphériques permettant la communication entre le microprocesseur et un utilisateur ou entre le microprocesseur et un système quelconque tels que les feux de signalisation qui feront l'objet de notre exemple dans le cinquième chapitre.

Le microprocesseur 6800 de MOTOROLA dialogue avec la périphérie à travers des circuits d'entrée/sortie programmables. Parmi ces circuits, on peut citer l'ACIA et le SSDA qui sont utilisés pour la transmission série, le PIA qui assure la transmission parallèle.

II. DESCRIPTION DU MC682I.

I°) Registres du MC682I.

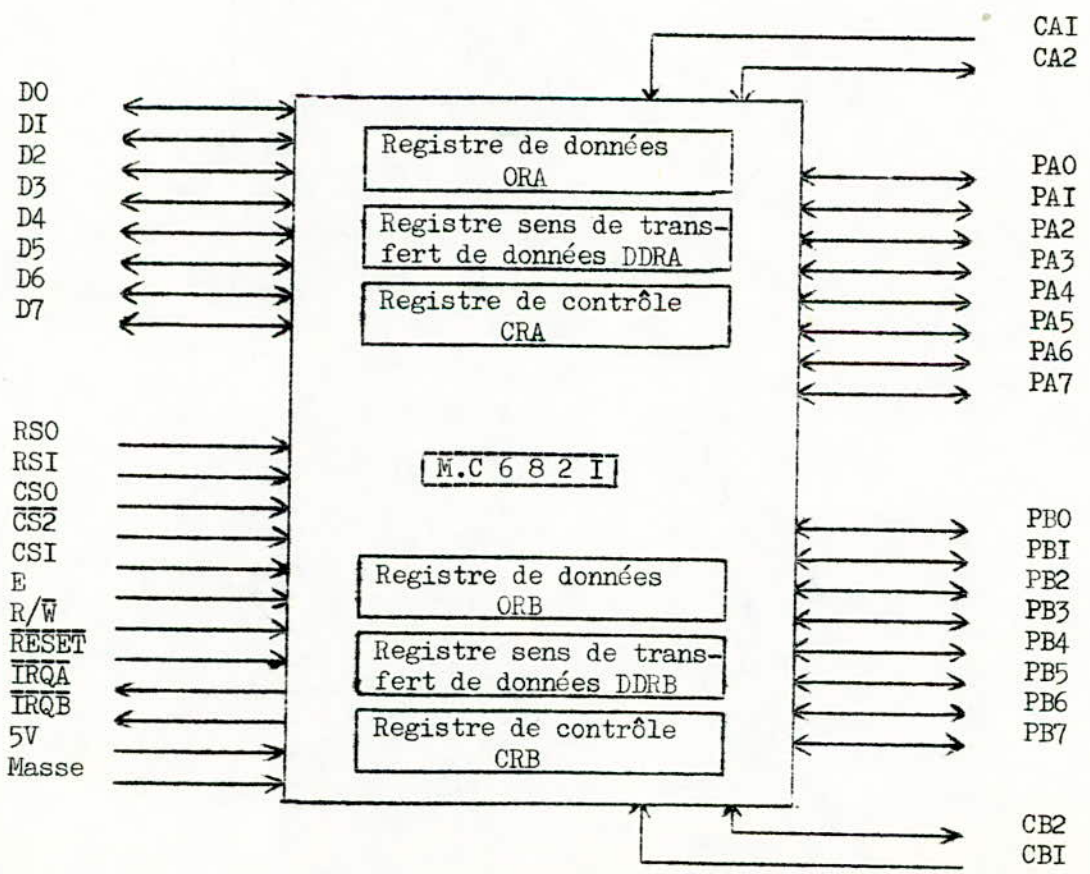


Fig.I5 Registres du MC682I

Le PIA est constitué de deux parties A et B identiques et indépendantes. Chaque partie comprend trois registres à huit bits chacun:

- le registre de contrôle CRA (CRB) qui assure la commande des deux lignes CAI et CA2 (CBI et CB2), la gestion des interruptions ainsi que la sélection des deux autres registres;
- le registre sens de transfert de données DDRA (DDRB) qui sert à fixer le sens de l'échange de données. Chaque bit de ce registre détermine le sens (entrée ou sortie) de la ligne correspondante. L'écriture d'un 0 ou d'un I définit respectivement la ligne en entrée ou en sortie;
- le registre de données ORA (ORB) qui sert de " tampon " d'échanges de données entre le microprocesseur et les organes périphériques.

Chacun de ces registres internes est considéré par le microprocesseur comme un emplacement mémoire adressable.

2°) Liaisons MPU-PIA et PIA-Périphérique.

Le MC682I se présente sous forme de boitier à 40 broches.

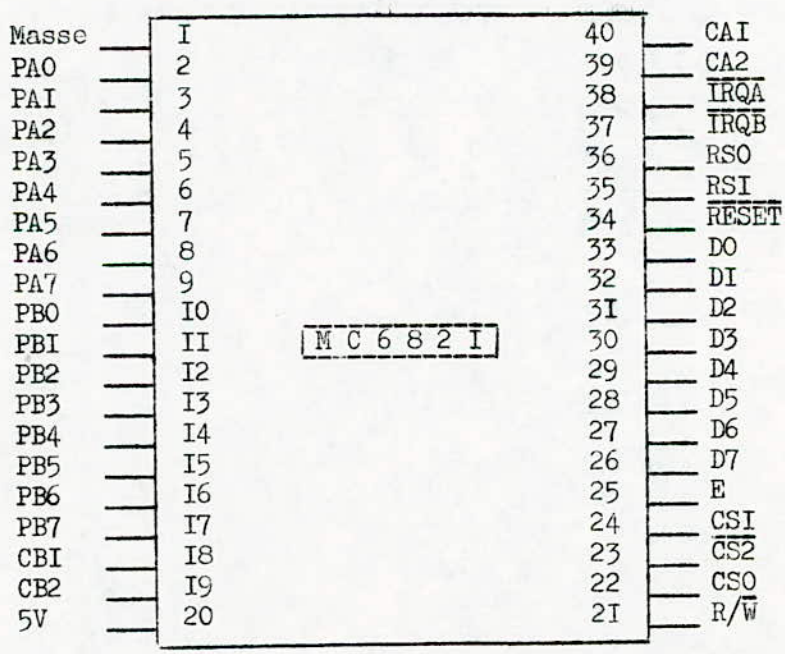


Fig. I6 Brochage du MC682I.

A travers toutes ces lignes, le PIA organisera le transfert de données entre le microprocesseur et les périphériques. Pour celà deux types de liaisons sont utilisées :

a) Liaisons MPU-PIA :

Ces liaisons sont assurées par :

- le bus de données (D0-D7) qui assure le transfert de l'information entre le MPU et le PIA;

-les lignes de sélection "chips selects " (CS0, CSI, $\overline{CS2}$) qui permettent la sélection du boitier du PIA. Les lignes CS0 et CSI doivent-être maintenues au niveau "I" et la ligne $\overline{CS2}$ au niveau "0";

-les lignes de sélection des registres (RS0, RSI) qui assurent l'adressage des registres internes du PIA. Seuls les registres CRA et CRB sont adressables directement grâce à ces deux lignes. Les autres seront adressés en fonction de la valeur du bit 2 écrit au préalable dans CRA ou CRB;

- la ligne lecture/écriture (R/$\overline{W}$) qui permet au MPU d'effectuer une lecture ou écriture sur le PIA suivant l'état logique "I" ou "0";

- l'enable (E) qui correspond à la phase d'horloge. Cette ligne est utilisée pour indiquer à tous les boitiers l'instant où les données doivent-être prises en compte;

- la ligne $\overline{RESET}$ qui permet l'initialisation des registres du PIA;

- la ligne $\overline{IRQA}$ ($\overline{IRQB}$) qui provoque une demande d'interruption quand elle passe du niveau haut au niveau bas.

b) Liaisons PIA-Périphériques :

Elles sont assurées par :

- les ports A et B (PA0-PA7 et PB0-PB7) qui, chacun constitué de huit lignes bidirectionnelles, assurent le transfert de l'information entre le PIA et les périphériques ;

- les lignes de contrôle (CAI, CBI, CA2, CB2) où la ligne CAI (CBI) est uniquement utilisée en entrée pour une demande d'interruption correspondant à la mise à "I" du bit 7 du CRA (CRB). La ligne CA2 (CB2) peut-être utilisée soit en entrée, dans ce cas son fonctionnement est analogue à CAI (CBI), soit en sortie pour le contrôle des périphériques .

3°) Rôle des bits b0 à b7 du registre de contrôle CRA (CRB).

Les bits b0 à b5 du registre de contrôle peuvent-être lus ou écrits par le MPU ; tandis que les bits b6 et b7 sont uniquement lus. L'arrivée sur les lignes CAI, CBI, CA2 et CB2 d'une interruption extérieure ou la lecture du registre de sortie ORA (ORB) par le MPU modifient les bits b6 et b7.

Dans le registre de contrôle CRA (CRB), l'état logique du bit b0 autorise ou masque (empêche) la demande d'interruption qui arrive sous forme d'impulsion sur la ligne CAI (CBI). Si b0 = 0, les interruptions sont masquées et la ligne $\overline{IRQA}$ ($\overline{IRQB}$) est inhibée. Si b0 = I, les interruptions sont autorisées et la ligne $\overline{IRQA}$ ($\overline{IRQB}$)est activée. Dans les deux cas, le bit b7 du CRA (CRB) est positionné à " I "

Le bit bI permettra au programmeur de définir le front actif de l'impulsion de demande d'interruption appliquée sur CAI (CBI). Quand bI =0 ou bI = I, respectivement la

transition négative ou positive sera prise en compte. Le bit b7 est également mis à "I".

La demande d'interruption $\overline{IRQ}$ qui arrive au microprocesseur ne sera prise en compte que lorsque le bit I (bit d'interruption masquable) du registre code condition (CCR) est à zéro.

Le bit b2 permet l'adressage des registres ORA (ORB) et DDRA (DDRB) du PIA . L'état logique "0" ou "I" du bit b2 du CRA (CRB) sélectionne respectivement le registre DDRA (DDRB) ou ORA (ORB). Le registre CRA (CRB) est directement adressable grâce aux lignes RS0 et RSI. Le tableau ci-dessous résume l'adressage de tous ces registres.

RSI	RS0	CRA2	CRB2	Registres sélectionnés
0	0	I	-	ORA
0	0	0	-	DDRA
0	I	-	-	CRA
I	0	-	I	ORB
I	0	-	0	DDRB
I	I	-	-	CRB

Si b5 = o, la ligne CA2 (CB2) est programmée en entrée; dans ce cas, les bits b3, b4 et b6 ont respectivement le même rôle que les bits b0, bI et b7. Si b5 = I, la ligne CA2 (CB2) est programmée en sortie; ce cas génère trois modes de fonctionnement :

- b5 = I b4 = I : c'est le mode SET/$\overline{RESET}$.
- b5 = I b4 = 0 b3 = 0 : c'est le mode HAND-SHAKING.
- b5 = I b4 = 0 b3 = I : c'est le mode PULSE STROBE

III. EXEMPLES DE PROGRAMMATION.

Pour faciliter la compréhension du rôle de chacun des bits du CRA (CRB), on se propose de donner quelques exemples de programmation.

I°) Programme fixant le port A en lecture, le port B en écriture (Initialisation du PIA).

Avant d'écrire le programme, on fixe les registres internes du PIA à des adresses choisies arbitrairement. Ainsi :

- CRA est mis à l'adresse AØØ5.
- DDRA et ORA partagent l'adresse AØØ4.
- CRB est à l'adresse AØØ7.
- DDRB et ORB partagent l'adresse AØØ6.

Le programme est le suivant :

```
DEBUT
CLR     ØAØØ5H      ; adressage du registre de contrôle CRA, sélection du registre
CLR     ØAØØ4H        de direction de transfert DDRA, programmation du port A en
                      entrée (lecture).
CLR     ØAØØ7H      ; adressage du registre de contrôle CRB, sélection du registre
LDA     A #ØFFH       de direction de transfert DDRB, programmation du port B en
STA     A ØAØØ6H      sortie (écriture).

LDA     A #Ø4H      ; positionnement à "I" des bits b2 des registres CRA et CRB .
STA     A ØAØØ5H      Sélection des registres de données ORA et ORB .
STA     A ØAØØ7H
```

Explication :

L'adressage du registre de contrôle CRA (CRB) et la sélection du registre sens de transfert de données DDRA (DDRB) sont assurés par l'instruction CLR ØAØØ5H (CLR ØAØØ7H) qui permet la mise à zéro du contenu du registre CRA (CRB); de ce fait, le bit b2 du CRA (CRB) se retrouve à zéro et permet l'accès au registre DDRA (DDRB). L'instruction CLR ØAØØ4H met à zéro les bits du registre DDRA, ainsi, le port A est programmé en entrée.

La mise à "I" des bits du registre DDRB, effectuée à l'aide des instructions LDA A #ØFFH et STA A ØAØØ6H, programme le port B en sortie.

L'échange de l'information entre le microprocesseur et l'organe périphérique est assuré par la séquence d'instructions LDA A #Ø4H, STA A ØAØØ5H et STA A ØAØØ7H. La valeur Ø4 est chargée dans l'accumulateur A puis rangée aux adresses AØØ5 et AØØ7 fixant ainsi à "I" les bits b2 des registres CRA et CRB. La mise à "I" de ces deux bits permet la sélection des registres de données ORA et ORB.

2°) Programme permettant le transfert d'une information mémorisée à l'adresse ØØAA jusqu'à un organe périphérique connecté au port B du PIA dont les lignes seront programmées en sortie .

On garde les mêmes adresses que celles utilisées dans le premier exemple, c'est à dire le registre CRB est à l'adresse AØØ7, les registres DDRB et ORB partagent l'adresse AØØ6.

La figure ci-dessous illustre cet échange:

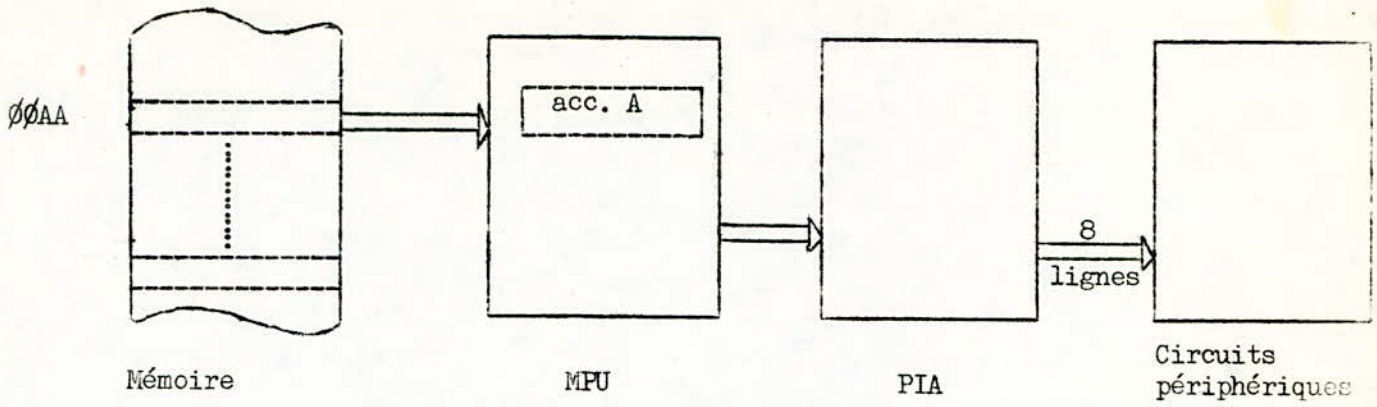


Fig.I7 Echange d'une information entre le MPU et un périphérique.

Le programme est le suivant :

```
CLR     ØAØØ7H      ; adressage du CRB et sélection du DDRB.

LDA     A #ØFFH     ; programmation du port B en sortie.
STA     A ØAØØ6H

LDA     A #Ø4H      ; sélection du registre ORB.
STA     A ØAØØ7H

LDA     A ØØAAH     ; sortie de l'information mémorisée à l'adresse ØØAA sur le port B.
STA     A ØAØØ6H
```

Explication :

L'instruction CLR ØAØØ7H adresse le registre de contrôle CRB et sélectionne le registre sens de transfert de données DDRB par le biais du bit b2 du CRB qui est positionné au niveau logique "0".

La paire d'instructions LDA A #ØFFH et STA A ØAØØ6H met à "I" les bits du registre DDRB permettant ainsi la programmation du port B en sortie.

La quantité binaire 00000I00 (ou Ø4H) met à "I" le bit b2 du registre CRB et permet donc au microprocesseur d'accéder au registre ORB. Cette séquence est effectuée par les instructions LDA A #Ø4H et STA A ØAØØ7H.

LDA A ØØAAH donne l'ordre à l'unité centrale de charger l'accumulateur A avec le contenu de l'adresse ØØAA. L'instruction STA A ØAØØ6H range ce contenu dans le registre ORB, registre qui est en liaison avec les circuits périphériques par l'intermédiaire du port B

c) Programme permettant de savoir si le bit b7 du registre de contrôle a été positionné "I" lors d'une demande d'interruption.

On considérera ici le port A. La même séquence d'instructions est utilisée dans le cas du port B. On gardera également les mêmes adresses qu'auparavant. Le registre CRA occupera l'adresse AØØ5, les registre DDRA et ORA partageront l'adresse AØØ4.

Après l'initialisation du PIA, arrive la suite d'instructions ci-dessous qui nous éclaircira sur l'état logique du bit b7 du CRA en testant le bit Z du registre d'état du 6800.

```
BETA    LDA     A ØAØØ5H        ; charge l'accumulateur A avec le contenu du CRA.
        BIT     A #8ØH          ; teste le bit b7 du CRA.
        BNE     ALPHA           ; branche à ALPHA si Z = 0.
        BRA     BETA            ; branche à BETA si Z = I.
ALPHA   LDA     A AØØ4H         ; charge l'accumulateur A avec le contenu de ORA.
        STA     A ØØAAH         ; range le contenu de l'accumulateur A à l'adresse ØØAA.
```

Applications :

L'instruction BIT A #8ØH effectue un "ET" logique entre le contenu de l'accumulateur A et l'opérande précisé dans l'instruction, soit en binaire I0000000. Dans le cas où tous les bits du résultat du "ET" logique sont à zéro, l'indicateur d'état Z est mis à "I"; dans le cas contraire, il est mis à "0". Ainsi, si le bit b7 du CRA était à "I" , le résultat du "ET" logique serait différent de "0", ce qui positionnerait Z à "0".

CHAPITRE V

- R E A L I S A T I O N D ' U N E M A Q U E T T E A M I C R O P R O C E S S E U R (M C 6 8 0 0) S U R L E 8 0 0 2 A -

I. Introduction

II. Elaboration du logiciel

III. Conception de la maquette prototype

IV. Exécution du programme sur la maquette prototype sous contrôle du 8002A

V. Quelques exemples utilisant le RTPA

I. INTRODUCTION.

Dans ce chapitre, on se propose de mettre au point une maquette à microprocesseur (MC6800) sur le TEKTRONIX 8002A illustrant les notions développées auparavant.

Le problème qu'on se pose ici est celui du contrôle des feux de signalisation tricolores (rouge, orange,vert) à un carrefour routier qui comporte deux rues à trafic inégal. On supposera que le trafic sur la rue I est plus important que celui de la rue 2. On se propose de régler le trafic de la manière suivante:

Etat I: le feu de la voie I est vert, celui de la voie 2 est rouge.

Etat 2: le feu de la voie I est orange, celui de la voie 2 est rouge.

Etat 3: le feu de la voie I est rouge, celui de la voie 2 est vert.

Etat 4: le feu de la voie I est vert, celui de la voie 2 est orange.

Les états 2 et 4 étant des états transitoires et brefs, leur durée est fixée à 2 secondes. Le feu de la rue I dont le trafic est plus dense, doit rester vert plus longtemps que celui de la rue 2, on convient donc d'une durée de 30 secondes pour l'état I et de I5 secondes pour l'état 3. Toutefois, une interruption voulue venant de l'extérieur déclenchera un cinquième état permettant un clignotement des feux oranges des deux rues. Cette possibilité est généralement utilisée pendant la nuit où le trafic est très diminué. La durée de l'allumage des feux oranges est fixée à une seconde. Leur extinction dure une demie seconde.

Le système à mettre en œuvre nécessite une partie matérielle et une partie logicielle.

La partie matérielle comporte:

- un support du MC6800 sur lequel sera implanté la sonde reliée au processeur émulateur;
- un interface (PIA 682I) qui assure l'échange de données entre le MPU et l'extérieur (LED);
- une horloge biphasée (687IA) assurant l'activation du MPU et la synchronisation du système;
- des circuits TTL SN7400N, SN7432N, SN7408N permettant le positionnement au niveau logique " 0 " ou "I" de certaines broches du MPU et du PIA.

La partie logicielle est constituée d'un programme gérant le contrôle des feux de signalisation.

II. ELABORATION DU LOGICIEL.

I°) Organigrammes.

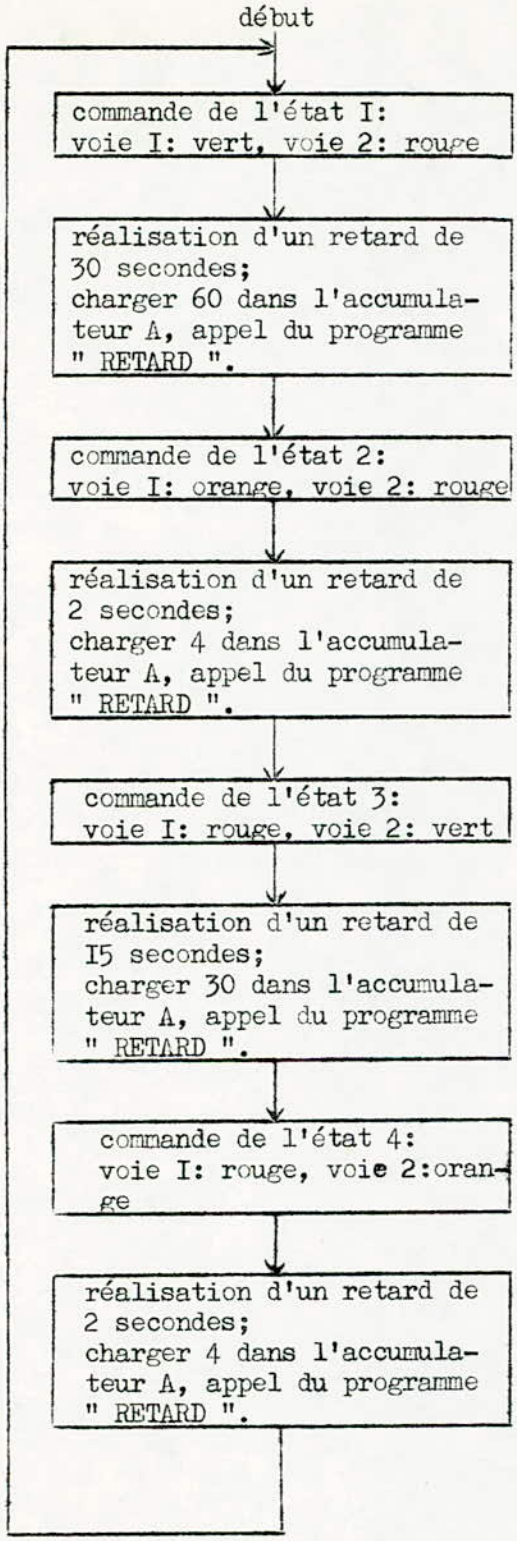


Fig. I8 Organigramme du cycle " 3 feux ".

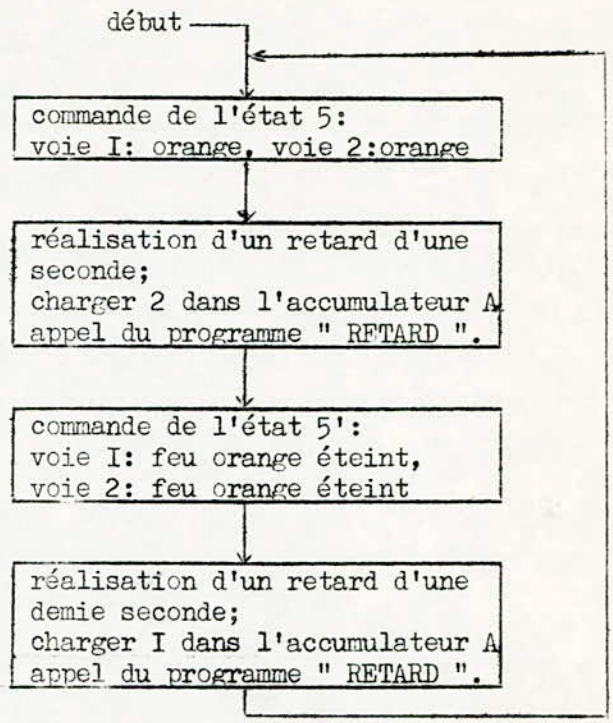


Fig. I9 Organigramme des feux clignotants.

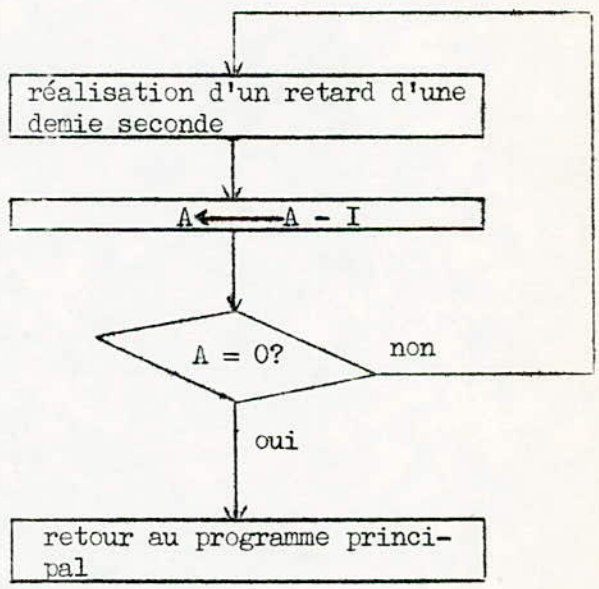


Fig. 20 Organigramme du retard.

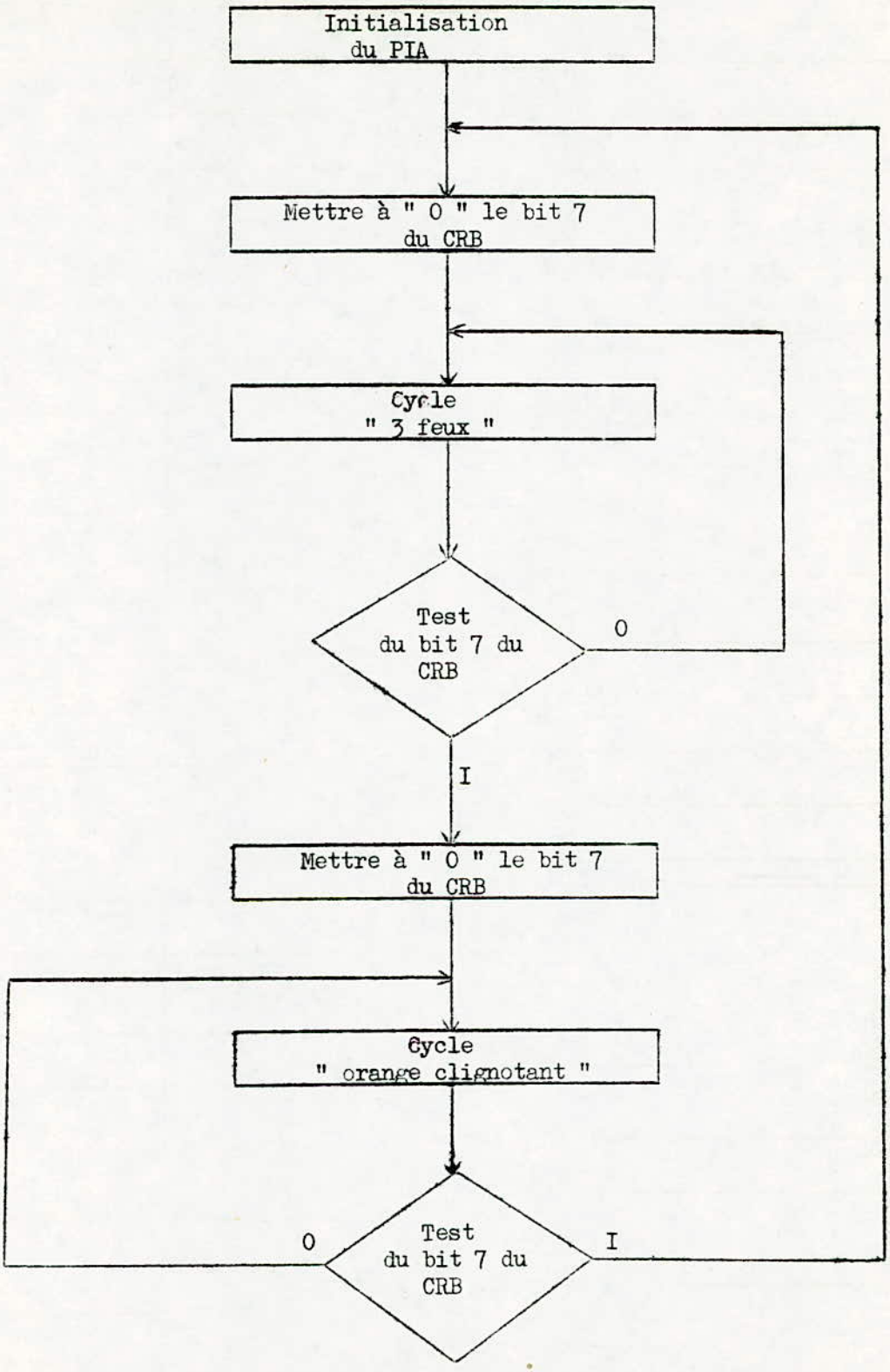


Fig. 2I Organigramme général.

2°)Ecriture du programme source.

```
          ORG   ØFFH          ;
          LDS   #6ØØH         ;
          CLR   ØAØØ7H        ; adressage du CRB, sélection du DDRB
          LDA   A #ØFFH       ;
          STA   A ØAØØ6H      ; mise à "I" du DDRB, programmation du port B en sortie
          LDA   A #Ø6H        ;
          STA   A ØAØØ7H      ; sélection du CRB, transition positive active
BOUCLE    CLR   ØAØØ6H        ;
          LDA   A #ØCH        ;
          STA   A ØAØØ6H      ; voie I = vert, voie 2 = rouge
          LDA   A #3CH        ;
          BSR   RETARD        ; délai de 3Ø secondes
          LDA   A #I4H        ;
          STA   A ØAØØ6H      ; voie I = orange, voie 2 = rouge
          LDA   A #Ø4H        ;
          BSR   RETARD        ; délai de 2 secondes
          LDA   A #2IH        ;
          STA   A ØAØØ6H      ;  voie I = rouge, voie 2 = vert
          LDA   A #IEH        ;
          BSR   RETARD        ;  délai de I5 secondes
          LDA   A #22H        ;
          STA   A ØAØØ6H      ; voie I = rouge, voie 2 = orange
          LDA   A #Ø4H        ;
          BSR   RETARD        ; délai de 2 secondes
          LDA   A ØAØØ7H      ;
          BIT   A #8ØH        ; test du bit 7 du CRB
          BNE   CLIGN         ;
          BRA   BOUCLE        ;
CLIGN     CLR   ØAØØ6H        ;
          LDA   A #I2H        ;
          STA   A ØAØØ6H      ; voie I = orange, voie 2 = orange
          LDA   A #Ø2H        ;
          BSR   RETARD        ; délai d'une seconde
          LDA   A ØAØØ7H      ;
          BIT   A #8ØH        ; test du bit 7 du CRB
          BNE   BOUCLE        ;
```

```
            CLR   ØAØØ6H        ; voie I = feu orange éteint, voie 2 = feu orange éteint
            LDA   A #ØIH        ;
            BSR   RETARD        ; délai d'une demie seconde
            LDA   A ØAØØ7H      ;
            BIT   A #8ØH        ;  test du bit 7 du CRB
            BNE   BOUCLE        ;
            BRA   CLIGN         ;
RETARD      LDX   #ØØH          ;
TOURNE      INX                 ;
            CPX   #ØBI9H        ; compare le contenu du registre d'indexage à la valeur
            BNE   TOURNE        ; ØBI9IH
            DEC   A             ;
            BIT   A #ØFFH       ;  compare le contenu de l'accumulateur A à ØFFH
            BNE   RETARD        ;
            RTS                 ;   retour au programme
```

3°) Test du programme au niveau du système de développement.

Le test au niveau du système de développement, en mode d'émulation Ø, du programme d'application, consiste en l'édition de ce programme, son assemblage pour détecter et corriger les erreurs de syntaxe éventuelles, son chargement en mémoire et finalement son exécution. Si des erreurs de logique sont détectées, on corrige le programme, puis on le refait assembler et l'on recommence cette procédure itérative jusqu'à l'obtention d'un programme correct.

III. CONCEPTION DE LA MAQUETTE PROTOTYPE (figure 22).

Il suffit d'interconnecter correctement les différents éléments du système, c'est à dire relier entre elles les broches du PIA et du MPU réservées aux données pour constituer le bus de données, relier respectivement les broches d'adresses du MPU (AO,AI,A2, AI3,AI4) aux broches RSO, RSI, CSO, CSI, $\overline{CS2}$ du PIA formant le bus d'adresses, relier les broches $\overline{IRQ}$, $\overline{RESET}$, $\overline{R/W}$ du MPU aux broches correspondantes du PIA réalisant le bus de commande et finalement relier les broches PBO, PBI, PB2, PB3, PB4, PB5 aux diodes électroluminescentes (LED).

IV. EXECUTION DU PROGRAMME SUR LA MAQUETTE PROTOTYPE SOUS CONTROLE DU 8002A.

L'exécution du programme sur la maquette prototype sous contrôle du 8002A

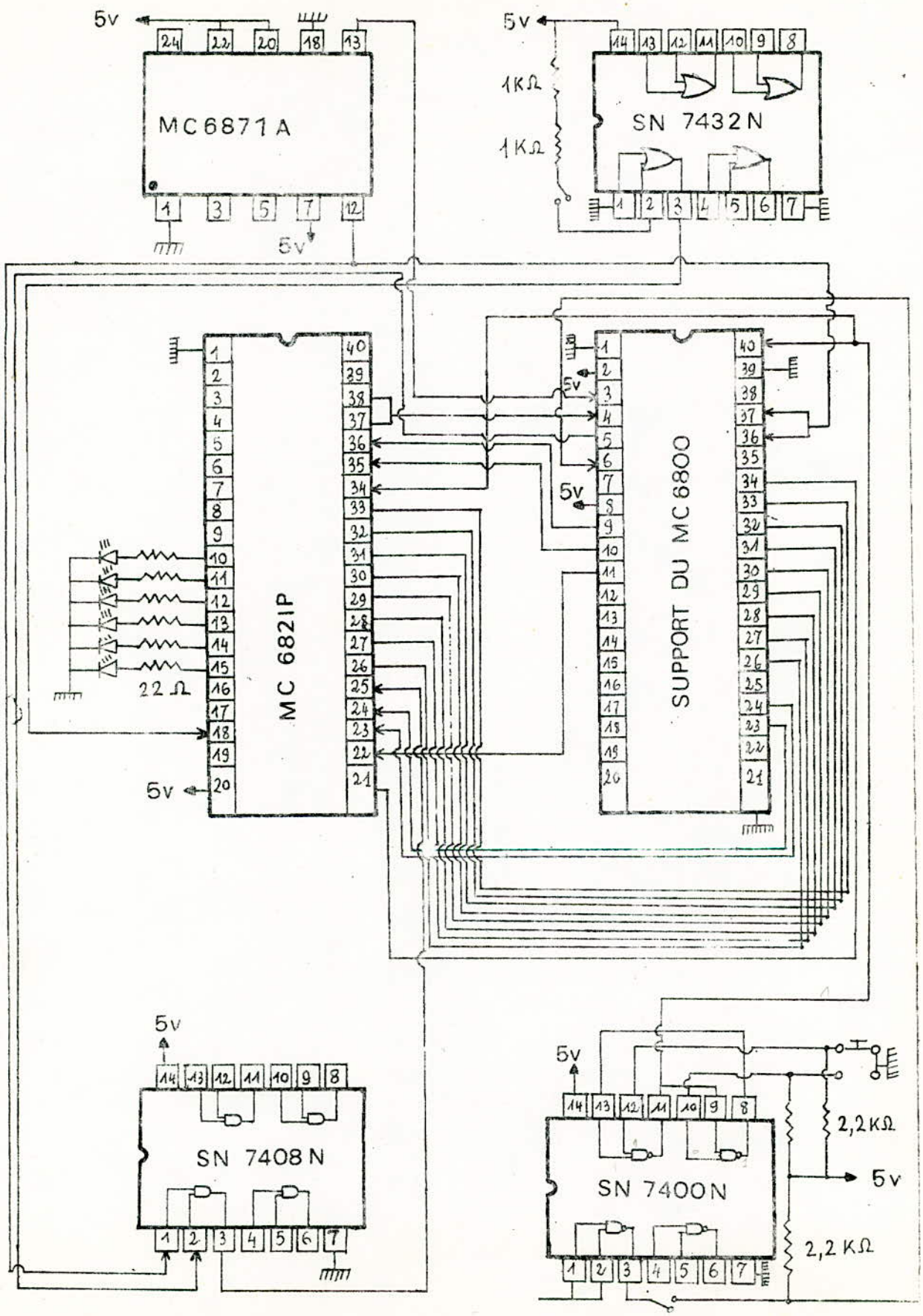


Fig.22 SCHEMA DE LA MAQUETTE REALISEE

s'effectuera en mode d'émulation I. Ce mode d'émulation utilise l'horloge du prototype et les mémoires de la maquette (registre du PIA) et du 8002A.

Les commandes ci-dessous assureront la mise en marche de la maquette prototype:

```
  > EM I
 * EMULATE * EOJ
   > LO FEUXO
 TRANSFER ADDRESS: ØØØØ
 * LOAD * EOJ
   > MA P FF-I75
   > MA U AØØ4-AØØ7
   > FILL I75 6ØØ ØØ
   > DEB
   >TR OFF
   > G FF
```

La ligne de commande MA P FF-I75 réserve, en mémoire programme, les cases mémoire FF à I75. MA U AØØ4-AØØ7 réserve, en mémoire extérieure (registres du PIA), les cases mémoire AØØ4 à AØØ7. FILL I75 6ØØ ØØ met à zéro les cases mémoire I75 à 6ØØ réservées à la pile.

V. QUELQUES EXEMPLES UTILISANT LE RTPA.

Pour l'utilisation du RTPA, on a besoin de certaines adresses affectées aux instructions du programme; pour celà on donne ci-dessous le " listing " d'assemblage.

```
Tektronix                          M68ØØ      V3.3                          Page I
ØØØØI                 ØØFF                 ORG   ØFFH      ;
ØØØØ2      ØØFF       8EØ6ØØ               LDS   #6ØØH     ;
ØØØØ3      ØIØ2       7FAØØ7               CLR   ØAØØ7H    ; adressage du CRB, sélection du
ØØØØ4      ØIØ5       86FF                 LDA   A #FFH    ; DDRB
ØØØØ5      ØIØ7       B7AØØ6               STA   A ØAØØ6H  ; mise à "I" du DDRB, programmation
ØØØØ6      ØIØA       86Ø6                 LDA   A #Ø6H    ; du port B en sortie
ØØØØ7      ØIØC       B7AØØ7               STA   A ØAØØ7H  ; sélection du ORB, transition posi-
ØØØØ8      ØIØF       7FAØØ6   BOUCLE      CLR   ØAØØ6H    ; tive active
ØØØØ9      ØII2       86ØC                 LDA   A #ØCH    ;
ØØØIØ      ØII4       B7AØØ6               STA   A ØAØØ6H  ; voie I = vert, voie 2 = rouge
ØØØII      ØII7       863C                 LDA   A #3CH    ;
ØØØI2      ØII9       8D47                 BSR   RETARD    ; délai de 3Ø secondes
ØØØI3      ØIIB       86I4                 LDA   A #I4H    ;
```

```
ØØØI4   ØIID   B7AØØ6          STA   A ØAØØ6H   ; voie I = orange, voie 2 = rouge
ØØØI5   ØI2Ø   86Ø4            LDA   A ##Ø4H    ;
ØØØI6   ØI22   8D3E            BSR   RETARD     ; délai de 2 secondes
ØØØI7   ØI24   862I            LDA   A ##2IH    ;
ØØØI8   ØI26   B7AØØ6          STA   A ØAØØ6H   ; voie I = rouge, voie 2 = vert
ØØØI9   ØI29   86IE            LDA   A ##IEH    ;
ØØØ2Ø   ØI2B   8D35            BSR   RETARD     ; délai de I5 secondes
ØØØ2I   ØI2D   8622            LDA   A ##22H    ;
ØØØ22   ØI2F   B7AØØ6          STA   A ØAØØ6H   ; voie I = rouge, voie 2 = orange
ØØØ23   ØI32   86Ø4            LDA   A ##Ø4H    ;
ØØØ24   ØI34   8D2C            BSR   RETARD     ; délai de 2 secondes
ØØØ25   ØI36   B7AØØ7          LDA   A ØAØØ7H   ;
ØØØ26   ØI39   858Ø            BIT   A ##8ØH    ; test du bit 7 du CRB
ØØØ27   ØI3B   26Ø2            BNE   CLIGN      ;
ØØØ28   ØI3D   2ØDØ            BRA   BOUCLE     ;
ØØØ29   ØI3F   7FAØØ6  CLIGN   CLR   ØAØØ6H     ;
ØØØ3Ø   ØI42   86I2            LDA   A ##I2H    ;
ØØØ3I   ØI44   B7AØØ6          STA   A ØAØØ6H   ; voie I = orange, voie 2 = orange
ØØØ32   ØI47   86Ø2            LDA   A ##Ø2H    ;
ØØØ33   ØI49   8DI7            BSR   RETARD     ; délai d'une seconde
ØØØ34   ØI4B   B6AØØ7          LDA   A ØAØØ7H   ;
ØØØ35   ØI4E   858Ø            BIT   A ##8ØH    ; test du bit 7 du CRB
ØØØ36   ØI5Ø   26BD            BNE   BOUCLE     ;
ØØØ37   ØI52   7FAØØ6          CLR   ØAØØ6H     ; voie I = feu orange éteint,
ØØØ38   ØI55   86ØI            LDA   A ##ØIH    ; voie 2 = feu orange éteint
ØØØ39   ØI57   8DØ9            BSR   RETARD     ; délai d'une demie seconde
ØØØ4Ø   ØI59   B6AØØ7          LDA   A ØAØØ7H   ;
ØØØ4I   ØI5C   858Ø            BIT   A ##8ØH    ; test du bit 7 du CRB
ØØØ42   ØI5E   26AF            BNE   BOUCLE     ;
ØØØ43   ØI6Ø   2ØDD            BRA   CLIGN      ;
ØØØ44   ØI62   CEØØØØ RETARD   LDX   ##ØØØØ    ;
ØØØ45   ØI65   Ø8      TOURNE  INX              ;
ØØØ46   ØI66   8CBI9I          CPX   ##ØBI9IH   ; compare le contenu du registre
ØØØ47   ØI69   26FA            BNE   TOURNE     ; d'indexage à la valeur ØBI9IH
ØØØ48   ØI6B   4A              DEC   A           ;
ØØØ49   ØI6C   85FF            BIT   A ØFFH     ; compare le contenu de l'accumula-
ØØØ5Ø   ØI6E   26F2            BNE   RETARD     ; teur A à ØFFH
ØØØ5I   ØI7Ø   39              RTS              ; retour au programme
ØØØ52
```

```
Tektronix      M68ØØ        ASM V3.3                Symbol Table              Page 2
% FEUXO (default)  Section (ØI7I)
     BOUCLE - ØIØF          CLIGN - - ØI3F          RETARD - ØI62
TOURNE - ØI65
     52 Source lines        52 Assembled lines        935 Bytes available
                    >>> No assembly errors detected <<<
```

I°) Exemple I.

On vérifie la durée du premier état du cycle " 3 feux ".

```
> EVT I A = IIB B = ALL
> DEB
>TR OFF
>BIF IS
>CNT M
>G FF
```

Le point d'arrêt est mis à l'adresse IIB, l'option B = ALL spécifie la "surveillance" de toutes les opérations (écriture, lecture,recherche). BIF IS permet d'arrêter le programme quand l'adresse IIB est atteinte, l'option S redonne la main au système terminal (cette option est prise par défaut).CNT déclenche le compteur à partir du lancement du programme, l'option M indique que l'unité de comptage est la milliseconde.

A l'arrêt du programme, on obtient l'affichage suivant:

```
LOC    INST   MNEM   R   OPER   X/PC   EADD   RA   RB   XREG   SP     CC
ØIIB   86I4   LDA    A   #I4                   I4   ØØ   BI9I   Ø5FE   DØ
ØIIB   BREAK
```

L'affichage de la durée d'exécution s'effectuera enfaisant entrer la commande CNT sans paramètres.

```
> CNT
  COUNT = 29949 MSEC
```

2°) Exemple 2.

Pour calculer la durée d'un cycle complet " 3 feux ", on utilise les deux comparateurs du RTPA.

```
> EVT I A = IØF
> EVT 2 A = IØF
> DEB
> BIF ARM
>CNT M
>G FF
```

Dans la ligne de commande de EVT I, on a spécifié l'adresse du début de la boucle du cycle " 3 feux "; le compteur commencera à fonctionner quand apparaitra à l'entrée du RTPA l'adresse IØF. Le compteur s'arrêtera quand l'option spécifiée dans la ligne de commande de EVT 2 est satisfaite, c'est à dire quand on aura fait un cycle complet. Ce mode de fonctionnement du compteur est utilisé avec l'option ARM de la commande BIF.

A l'arrêt du programme, on aura sur la console les lignes suivantes:

```
LOC    INST    MNEM    R    OPER    X/PC    EADD    RA    RB    XREG    SP    CC
IØF    7FAØØ6  CLR          AØØ6            =AØØ6   46    ØØ    BI9I    Ø6ØØ  C4
IØF  BREAK
```

Pour avoir la durée du cycle complet, on fait la commande CNT sans paramètres, on aura:

```
> CNT
  COUNT = 489I7 MSEC
```

3°) Exemple 3.

Pour arrêter le programme après qu'il ait fait deux fois le cycle complet:

```
> EVT 2 A = IØF P = 3
> DEB
>BIF 2
>G FF
```

L'option P = 3 est utilisée pour forcer le programme à passer trois fois par l'adresse IØF, c'est à dire faire deux cycles complets. BIF 2, en liaison avec EVT 2, permet d'arrêter le programme après deux cycles complets.

On aura l'affichage suivant:

```
LOC    INST    MNEM    R    OPER    X/PC    EADD    RA    RB    XREG    SP    CC
ØIØF   7FAØØ6  CLR          AØØ6            =AØØ6   Ø6    ØØ    BI9I    Ø6ØØ  C4
ØIØF  BREAK
```

Si on ajoute l'option C = 64 dans la ligne de commande de EVT 2 , le programme s'arrêtera 64 microsecondes plus tard après avoir fait deux cycles complets, on aura sur la console les lignes suivantes:

```
LOC    INST    MNEM    R    OPER    X/PC    EADD    RA    RB    XREG    SP    CC
ØI66   8CBI9I  CPX          #BI9I                   32    ØØ    I66F    Ø5FE  C6
ØI66  BREAK
```

4°) Exemple 4.

Le programme s'arrêtera quand apparaitra sur le bus de données la configuration ØØIØØØIØ (voie I = rouge, voie 2 = orange).

```
>EVT I T = ØØIØØØIØ
>DEB
>TR OFF
>BIF I
>RTT W
>G FF
```

La commande RTT W permet de stocker dans la mémoire du RTPA seulement les opérations d'écriture en mémoire.

A l'arrêt du programme, on aura l'affichage suivant:

LOC	INST	MNEM	R	OPER	X/PC	EADD	RA	RB	XREG	SP	CC
ØI2F	B7AØØ6	STA	A	AØØ6		= AØØ6	22	ØØ	BI9I	Ø6ØØ	CØ
ØI2F	BREAK										

5°) Exemple 5.

Recherche de l'adresse d'une donnée connue située entre deux adresses connues.

```
>EVT I A  FF D = ØC
>EVT 2 A  I2Ø
>DEB
>BIF LIM
>G FF
```

BIF LIM est utilisée pour une comparaison combinée.

A l'arrêt du programme, on aura sur la console de visualisation l'affichage ci-dessous indiquant que la donnée ØC se trouve à l'adresse ØII2.

LOC	INST	MNEM	R	OPER	X/PC	EADD	RA	RB	XREG	SP	CC
ØII2	86ØC	LDA	A	##ØC			ØC	ØØ	A27E	Ø6ØØ	CØ

- C O N C L U S I O N -

Par le biais de cette étude, nous avons essayé de montrer la grande souplesse qu'offre le TEKTRONIX 8002A dans la conception de systèmes à microprocesseurs. Pour la mise au point de ces systèmes, le 8002A utilise trois modes d'émulation dont l'avantage est qu'il est désormais possible :

- de tester les programmes d'application afin de détecter les erreurs de logique (mode d'émulation Ø).
- de tester parallélement la maquette prototype en cours d'élaboration.
- de tester progressivement le logiciel de façon conjointe au matériel (mode d'émulation I).
- de corriger rapidement les erreurs.
- d'effectuer le test final et la mise au point en temps réel (mode d'émulation 2 utilisant les mémoires et l'horloge de la maquette prototype).

ANNEXE I

- COMMANDES DE TEKDOS -

TEKDOS possède six variétés de commandes. Ci-dessous est donnée la définition de chaque commande; l'introduction des caractères soulignés suffit.

I°) Les commandes du système:

- ABORT:annule toutes les commandes entrées précédemment.
- ASM: crée un fichier code objet et un listing d'assemblage à partir d'un programme source.
- ASSIGN: assigne des canaux à des fichiers ou des périphériques. Cette commande est utilisée lors des services call .
- CLOSE: ferme les canaux ouverts par la commande ASSIGN.
- CMPF: compare le contenu de deux fichiers et affiche la différence sur un fichier ou périphérique.
- CONT: continue l'exécution d'un programme suspendu.
- CONVERT: traduit un programme écrit en langage assembleur de MOTOROLA en un programme écrit en langage assembleur de TEKTRONIX.
- DELETE: efface le contenu de fichiers sur disque.
- DEVICE: informe TEKDOS si l'état des périphériques est disponible ou pas.
- EDIT: invoque l'éditeur de textes.
- EMULATE: choisit le mode d'émulation à utiliser (Ø, I ou 2).
- FETCH: charge en mémoire programme un fichier écrit en binaire.
- FORMAT: permet de formater un disque vierge.
- HELP: affiche un message explicant l'erreur se trouvant au niveau de la ligne de commande.
- LDIR: affiche le contenu du répertoire sur disque.
- LIBGEN: crée ou modifie les fichiers de bibliothèque.
- LINK: appelle l'éditeur de lien, lie deux ou plusieurs programmes code objet assemblés séparément.
- LOAD: charge en mémoire le code objet d'un programme source.
- MAP: choisit la pagination mémoire.
- MODULE: stocke sur disque un programme (binaire) se trouvant dans la mémoire programme.
- NEWDISC:permet de formater et vérifier le disque se trouvant dans le réceptacle drive I et de copier TEKDOS du disque contenu dans le réceptacle drive Ø sur le disque situé dans le réceptacle drive I.

- RENAME: change le nom d'un fichier ou d'un disque.
- STATUS: affiche les états du processeur émulateur, des canaux, du programme en exécution.
- SUSPEND: suspend l'exécution de programmes.
- SYSTEM: identifie le réceptacle contenant le disque système.
- TYPE: affiche ou supprime l'affichage d'une commande en exécution.
- VERIFY: vérifie l'état du disque.
- XEQ: charge en mémoire programme et exécute un programme écrit en binaire et stocké sur disque avec la commande MODULE.

2°) Les commandes de transfert de données:

- COMM: assure la communication entre le 8002A et un autre ordinateur.
- COPY: transfère un texte d'un fichier ou circuit vers un autre fichier ou circuit.
- COPYSYS: copie le programme moniteur TEKDOS d'un disque sur un autre.
- DUMP:transfère le contenu de la mémoire sur un fichier ou sur la console.
- DUP: copie le contenu d'un disque sur un autre.
- EXAM: affiche et permet de modifier le contenu de la mémoire.
- FDUMP: affiche le contenu d'un fichier dans sa représentation hexadécimale et ASCII.
- FILL: permet de remplir une zone mémoire avec une donnée.
- MOVE: transfère des données d'une zone mémoire vers une autre.
- PATCH: permet de faire entrer des expressions en hexadécimal en mémoire.
- PRINT: recopie le contenu d'un fichier sur une imprimante.
- PRINTL: recopie les données d'un fichier ou périphérique sur un autre en numérotant les lignes.
- RHEX: lit et convertit le code hexadécimal tektronix en code binaire et stocke en mémoire.
- WVEX: traduit le contenu de la mémoire programme en notation hexadécimale, affiche la traduction sur un périphérique.
- WHEX: recopie en traduisant en hexadécimal le contenu de la mémoire sur un fichier ou une imprimante.

3°) Les commandes de déverminage:

- BKPT: permet de mettre un pointd'arrêt à un programme en exécution.
- CAL: permet des calculs arithmétiques en hexadécimal au niveau de la console.
- CLBP: annule le point d'arrêt mis par BKPT.
- DEBUG: met en service le programme de déverminage.
- DISM: traduit un code objet en mnèmoniques.Cette commande peut être utilisée quand on fait entrer directement un programme écrit en code objet, sans passer par l'éditeur.

-DSTAT: affiche les régistres du microprocesseur émulateur

- GO: lance l'exécution d'un programme.

- SET: force les régistres du microprocesseur émulateur avec les valeurs désirées (sauf le compteur ordinal).

- TRACE : TRACE ALL permet l'affichage d'un programme en exécution.
  TRACE OFF annule l'affichage.
  TRACE ALL STEP affiche pas à pas.
  TRACE JMP affiche seulement les instructions de saut.

4°) Les commandes du RTPA:

- BIF: met un point d'arrêt, affiche, ou annule les options de la ligne de commande.
- CNT: permet de lancer le compteur.
- DRT: affiche sur la console le contenu de la mémoire du RTPA.
- EVT: permet la pose et l'annulation de comparaisons d'évènements.
- RTT: permet de choisir les transactions à stocker dans la mémoire du RTPA.

5°) Les commandes du programmeur de PROM:

-CPROM: compare le contenu de la mémoire programme avec celui d'une PROM.

- RPROM: transfére le contenu d'une PROM vers la mémoire programme.
- WPROM: recopie les données de la mémoire programme sur une PROM.
- CSMS: compare le contenu de la mémoire avec celui d'une bande perforée.

6°) Les touches du clavier:

- BACK SPACE: efface le dernier caractère écrit.
- ESC: utilisée une fois, elle efface la ligne courante; utilisée deux fois elle suspend l'exécution d'un programme.
- RUB OUT:annule le dernier caractère entré.
- SPACE BAR: insère un blanc dans une ligne, suspend et remet en marche l'affichage.

ANNEXE II

SIGNIFICATION DES MNEMONIQUES ET DUREE DES INSTRUCTIONS
(durée exprimée en périodes d'horloge)
Microprocesseur MOTOROLA 6800

		ACCY	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA	Add Accumulators						2	
ADC	Add with Carry		2	3	4	5		
ADD	Add		2	3	4	5		
AND	Logical And		2	3	4	5		
ASL	Arithmetic Shift Left	2			6	7		
ASR	Arithmetic Shift Right	2			6	7		
BCC	Branch if Carry Clear							4
BCS	Branch if Carry Set							4
BEQ	Branch if Equal to Zero							4
BGE	Branch if Greater or Equal Zero							4
BIT	Bit Test		2	3	4	5		4
BLE	Branch if Less or Equal							4
BLS	Branch if Lower or Same							4
BLT	Branch if Less than Zero							4
BMI	Branch if Minus							4
BNE	Branch if Not Equal to Zero							4
BPL	Branch if Plus							4
BRA	Branch Always							4
BSR	Branch to Subroutine							8
BVC	Branch if Overflow Clear							4
BVS	Branch if Overflow Set							4
CBA	Compare Accumulators						2	
CLC	Clear Carry						2	
CLI	Clear Interrupt Mask						2	
CLR	Clear	2			6	7		
CLV	Clear Overflow						2	
CMP	Compare		2	3	4	5		
COM	Complement	2			6	7		
CPX	Compare Index Register		3	4	5	6		
DAA	Decimal Adjust						2	
DEC	Decrement	2			6	7		
DES	Decrement Stack Pointer						4	
DEX	Decrement Index Register						4	
EOR	Exclusive OR		2	3	4	5		
INC	Increment	2			6	7		
INS	Increment Stack Pointer							
INX	Increment Index Register							
JMP	Jump				3	4		
JSR	Jump to Subroutine				9	8		
LDA	Load Accumulator		2	3	4	5		
LDS	Load Stack Pointer		3	4	5	6		
LDX	Load Index Register		3	4	5	6		
LSR	Logical Shift Right	2			6	7		
NEG	Negate	2			6	7		
NOP	No Operation						2	
ORA	Inclusive OR Accumulator		2	3	4	5		
PSH	Push Data						4	
PUL	Pull Data						4	
ROL	Rotate Left	2			6	7		
ROR	Rotate Right	2			6	7		
RTI	Return from Interrupt						10	
RTS	Return from Subroutine						5	
SBA	Subtract Accumulators						2	
SBC	Subtract with Carry		2	3	4	5		
SEC	Set Carry						2	
SEI	Set Interrupt Mask						2	
SEV	Set Overflow						2	
STA	Store Accumulator			4	5	6		
STS	Store Stack Register			5	6	7		
STX	Store Index Register			5	6	7		
SUB	Subtract		2	3	4	5		
SWI	Software Interrupt						12	
TAB	Transfer Accumulators						2	
TAP	Transfer Accumulators to Condition Code Reg.						2	
TBA	Transfer Accumulators						2	
TPA	Transfer Condition Code Reg. to Accumulator						2	
TST	Test	2			6	7		
TSX	Transfer Stack Pointer to Index Register						4	
TXS	Transfer Index Register to Stack Pointer						4	
WAI	Wait for Interrupt						9	

In " l'emploi des microprocesseurs " M. AUMIAUX

ANNEXE III

- DIRECTIVES D'ASSEMBLAGE -

I°) Les directives de contrôle du fichier d'assemblage:

- LIST: accès au fichier d'assemblage autorisé.
- NOLIST: accès au fichier d'assemblage interdit.
- PAGE: la prochaine ligne du programme source commence une nouvelle page du fichier d'assemblage.
- SPACE: insère des lignes blanches dans le fichier d'assemblage.
- TITLE: donne un titre au programme.
- STITLE: donne un sous titre au programme.
- WARNING: lorsqu'on suspecte une erreur dans le programme source, on utilise cette directive pour visualiser le code d'erreur.

2°) Les directives de définition de symboles:

- EQU: associe définitivement une valeur au symbole de la colonne étiquette.
- SET: associe ou réassocie une valeur au symbole de la colonne étiquette.
- STRING: avant d'utiliser un symbole comme variable concatenaire, on doit le définir avec cette directive.

3°) La directive de contrôle du compteur mémoire de l'assembleur:

- ORG: spécifie l'origine d'un programme; si cette directive n'est pas précisée, le programme débutera à l'adresse Ø.

4°) Les directives d'écriture de données:

- BYTE: insère une ou plusieurs constantes données dans le code objet à raison d'un octet de mémoire par constante.
- WORD: insère une ou plusieurs constantes données dans le code objet à raison de deux octets de mémoire par constante.
- ASCII: incorpore en code ASCII dans le code objet une expression concatenaire située dans la colonne opérande.
- BLOCK: réserve un nombre d'octets, spécifié dans la colonne opérande, en mémoire.

5°) Les directives de définition de macro-instructions:

- MACRO: donne un nom à un bloc de définition de macro-instruction.
- ENDM: termine le bloc de définition de macro-instruction.
- REPEAT: provoque l'assemblage répétitif des lignes du programme source.
- ENDR: termine le bloc défini par REPEAT.
- INCLUDE: incorpore au sein d'un programme en cours le texte d'un fichier sur disque. Ce texte ne doit pas comporter la directive INCLUDE. Celle-ci ne doit pas se trouver juste avant ENDM, ENDR ou ENDIF.

6°) Les directives d'assemblage conditionnel:

- IF: le bloc de lignes du programme source situé entre IF et ENDIF ne sera assemblé que si l'opérande est non nul.
- ENDIF: indique la fin d'un bloc commencé par IF.
- ELSE: permet l'assemblage d'un bloc en lieu et place du bloc IF-ENDIF quand celui-ci n'est pas assemblé (cas où l'opérande est nul).
- EXITM: achève la macro-expansion (traduction en code objet de la macro-instruction) en cours avant la directive ENDM.

7°) Les directives d'assemblage translatable:

- SECTION: déclare une section au sein d'un programme, lui donne un nom et définit ses caractéristiques.
- COMMON: réalise la même fonction que SECTION excepté que le même nom peut identifier des sections communes dans plus d'un module source.
- RESERVE: réserve une zone mémoire lors de l'édition de lien où les zones réservées portant le même nom sont mises en une seule zone. On spécifiera dans la colonne opérande le nom de la section et le nombre d'octets à réserver.
- RESUME: reprend la définition de la section en cours.
- GLOBAL: définit un ou plusieurs symboles comme variables globales.
- NAME: donne un nom au code objet.

8°) La directive de fin de module:

- END: termine un programme source.

ANNEXE IV

- COMMANDES DE L'EDITEUR DE TEXTES -

Il existe quatre types de commandes de l'éditeur de textes.

I°) Commandes de déplacement du pointeur du workspace: BEGIN, DOWN, END, FIND et UP.

2°) Commandes de transfert et d'affichage: COPY, FILE, GET, LIST, PUT, PUTK et TYPE.

3°) Commandes de modification de textes: INPUT, INSERT, KILL, REPLACE et SUBSTITUTE.

4°) Commandes d'utilisation: AGAIN, BRIEF, MACRO, QUIT, SUSPEND, TAB, TABS, XTABS ON, MOVE, NUMBER et la touche ESC.

On donnera ci-dessous une brève définition de chacune de ces commandes; seule l'introduction des caractères soulignés est nécessaire.

- AGAIN: elle répète l'exécution de la dernière commande exécutée; seules les commandes BRIEF, QUIT, FILE, TAB et TABS peuvent-être répétées. Il en est de même pour la dernière commande " répétable " contenue dans la définition de MACRO.

- BEGIN:déplace le pointeur du workspace vers la première ligne du texte et affiche cette ligne.

- BRIEF: permet ou supprime l'affichage provoqué par l'exécution de certaines commandes. L'affichage est annulé si on fait entrer une première fois BRIEF, il reprend dès la seconde introduction de BRIEF.

- COPY: réalise la copie du contenu d'un fichier sur un autre fichier ou sur la console.

- DOWN: déplace le pointeur du workspace vers le bas; le nombre de lignes à sauter est spécifié comme paramètre.

- END: déplace le pointeur vers la fin du texte.

- FILE: termine la phase d'édition en stockant sur disque le texte créé, redonne la main à TEKDOS.

- FIND: utilisée pour chercher dans le texte une expression spécifiée dans la ligne de commande; l'éditeur commence à chercher à partir de la ligne en cours jusqu'à la rencontre de l'expression recherchée.

- GET: déplace un texte se trouvant dans un fichier sur disque vers le workspace pour permettre des manipulations (corrections, adjonction d'autres lignes); à la suite de GET, on spécifie le nombre de lignes constituant le texte.

- INPUT: permet d'écrire dans le workspace à partir du clavier. Cette commande est utilisée chaque fois qu'on crée un texte.

- INSERT: insère une seule ligne avant la ligne courante.

- KILL: sans paramètre, cette commande efface la ligne courante. Si on veut effacer plusieurs lignes successives, il faudra spécifier le nombre à partir de la ligne courante.
- LIST: utilisée pour sortir un texte sur une imprimante.
- MACRO: elle permet de stocker des lignes de commandes destinées à être répétées; elle sert aussi à faire exécuter, afficher ou effacer ces commandes, selon les paramètres utilisés dans la ligne de commande de MACRO.
- PUT: recopie le contenu du workspace sur un fichier ou une imprimante sans détruire le contenu du workspace.
- PUTK: fait le même travail que PUT mais efface le contenu du workspace.
- QUIT: termine avec l'éditeur et redonne la main à TEKDOS; si on sort avec la commande QUIT, le contenu du workspace est perdu, ce qui fait la différence avec la commande FILE.
- REPLACE: remplace la ligne pointée par une nouvelle ligne; on laisse un blanc entre cette commande et la ligne qui la suit.
- SUBSTITUTE: remplace une expression par une autre au niveau de la ligne en cours.
- SUSPEND: suspend l'éditeur et redonne la main à TEKDOS.
- TAB: définit un caractère de tabulation; si on ne spécifie pas de caractère, c'est la lettre I qui jouera ce rôle.
- TABS: assigne des positions de tabulation différentes de celles prises par défaut.
- XTABS ON: permet d'utiliser la tabulation (8-16-24-32-48-56-64) ne necessitant pas de définition de caractère de tabulation.
- TYPE: affiche un texte sur la console à partir du workspace; on indique le nombre de lignes pour avoir le texte en entier.
- UP: déplace le pointeur du workspace, vers le haut, du nombre de lignes indiqué dans la ligne de commande.
- NUMBER: NUMBER ON numérote les lignes du texte; NUMBER OFF ôte la numérotation.
- MOVE: permet la transposition des lignes du texte.
- La touche ESC efface et permet ainsi de faire entrer une nouvelle ligne; si elle est pressée deux fois, elle efface la ligne et suspend l'éditeur.

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

I°) Livres:

RODNAY ZAKS et PIERRE LE BEUX. - Les microprocesseurs. SYBEX-EUROPE 1977, 1978.

R.LYON-CAEN et J.M.CROZET. - Microprocesseurs et microordinateurs. MASSON 1978.

2°) Manuels:

Manuels TEKTRONIX.

Manuel de programmation SESCOSEM.

3°) Revues:

Micro-système N°12.

Micro-système N°13.